

# Screen Space Ambient Occlusion Based Multiple Importance Sampling for Real-Time Rendering

Abd El Mouméne Zerari  · Mohamed Chaouki Babahenini

Received: 9 August 2017 / Revised: 20 November 2017 / Accepted: 3 December 2017 / Published online: 14 December 2017  
© 3D Research Center, Kwangwoon University and Springer-Verlag GmbH Germany, part of Springer Nature 2017

**Abstract** We propose a new approximation technique for accelerating the Global Illumination algorithm for real-time rendering. The proposed approach is based on the Screen-Space Ambient Occlusion (SSAO) method, which approximates the global illumination for large, fully dynamic scenes at interactive frame rates. Current algorithms that are based on the SSAO method suffer from difficulties due to the large number of samples that are required. In this paper, we propose an improvement to the SSAO technique by integrating it with a Multiple Importance Sampling technique that combines a stratified sampling method with an importance sampling method, with the objective of reducing the number of samples. Experimental evaluation demonstrates that our technique can produce high-quality images in real time and is significantly faster than traditional techniques.

**Keywords** Soft shadows · GPU · Global illumination · Real-time rendering · SSAO · Multiple importance sampling

## 1 Introduction

Real-time rendering of global illumination effects is a challenging problem. Currently, we can achieve sufficient image rates only by global illumination approximation, such as Ambient Occlusion (AO), which is a shading method. When light travels through a scene, some places are more occluded than others. The effect of making these areas appear is called AO, which takes visibility into account but not light. The calculation of AO depends on the local geometry of the scene, which is called the AO object-space, and is performed by projecting the rays in all directions of a point of the object-space that is centered in the top hemisphere to compute the ambient occlusion factor. The problem with this method is that the high number of rays that are launched from the object-space can be expensive to calculate.

AO allows the simulation of the soft shadows that occur in the fissures of 3D objects when indirect lighting is cast in the scene. Soft shadows that are calculated from AO can help define the separation between objects in the scene and add another level of realism to the rendered scene.

In general, AO methods are not practical for real-time rendering. Ambient occlusion in real time was out of reach until Screen-Space Ambient Occlusion (SSAO) was proposed. AO is approximated by the SSAO method in the screen-space. First introduced in games by Crytek [1] and extensively used in real-time

---

A. E. M. Zerari (✉) · M. C. Babahenini  
Department of Computer Science, LESIA, University of  
Mohamed Khider, B.P 145, 07000 Biskra, Algeria  
e-mail: a.zerari@live.fr

M. C. Babahenini  
e-mail: chaouki.babahenini@gmail.com

applications, SSAO is entirely computed in screen-space, straightforward to implement and independent of the scene geometry. Therefore, SSAO is faster to calculate than AO and can be added to a wide range of computer graphics applications. Several SSAO techniques [1–6] and [7] are available for image synthesis. They all have the same fundamental principle, which is based on approximating the points of the visible world-space by sampling the depths of neighboring pixels in a screen-space. These techniques have the power to accelerate the rendering step and are suitable for real-time applications. However, the effectiveness of the traditional SSAO method depends primarily on the number of samples per pixel and the sampling method. A reduced number of samples is used to achieve the desired performance, but noise appears in the result. This noise can be eliminated by a filtering process or by adding additional samples to converge to a better-quality result. This addition leads directly to degradation of performance. Traditional SSAO methods also select uniform random samples, which leads to a problem with the sample distribution. We try to find a compromise between visual quality and performance through our approach.

This article presents a novel technique for accelerating the computation time of the SSAO algorithm without affecting the visual quality of the results. Many Monte Carlo integration models exist, but with different sampling strategies, such as stratified sampling [8], importance sampling [9] and multiple importance sampling [10]. Our technique aims to reduce this number of samples by privileging the most significant samples. We replaced the uniform random sampling strategy by the Multiple Importance Sampling (MIS) method, which allowed us to improve the visual quality by adding more detail to areas where direct light does not reach. We applied our optimization method to different popular techniques, such as Crytek [1], Mittring [2], and Alchemy [6], which allowed us to obtain better visual results in less time.

The remainder of the paper is organized as follows: In Sect. 2, we present briefly the work that is related to our approach. In Sect. 3, we outline our generalized SSAO for the illumination of meso-structures, and we explain our extensions of the initial method for improving the visual quality and the integration of our technique into a complete global illumination simulation. Section 4 presents and discusses the results that were obtained and the comparative study

that we conducted to evaluate the performance of the proposed approach. Finally, in Sect. 5, we present our conclusions and directions for future work.

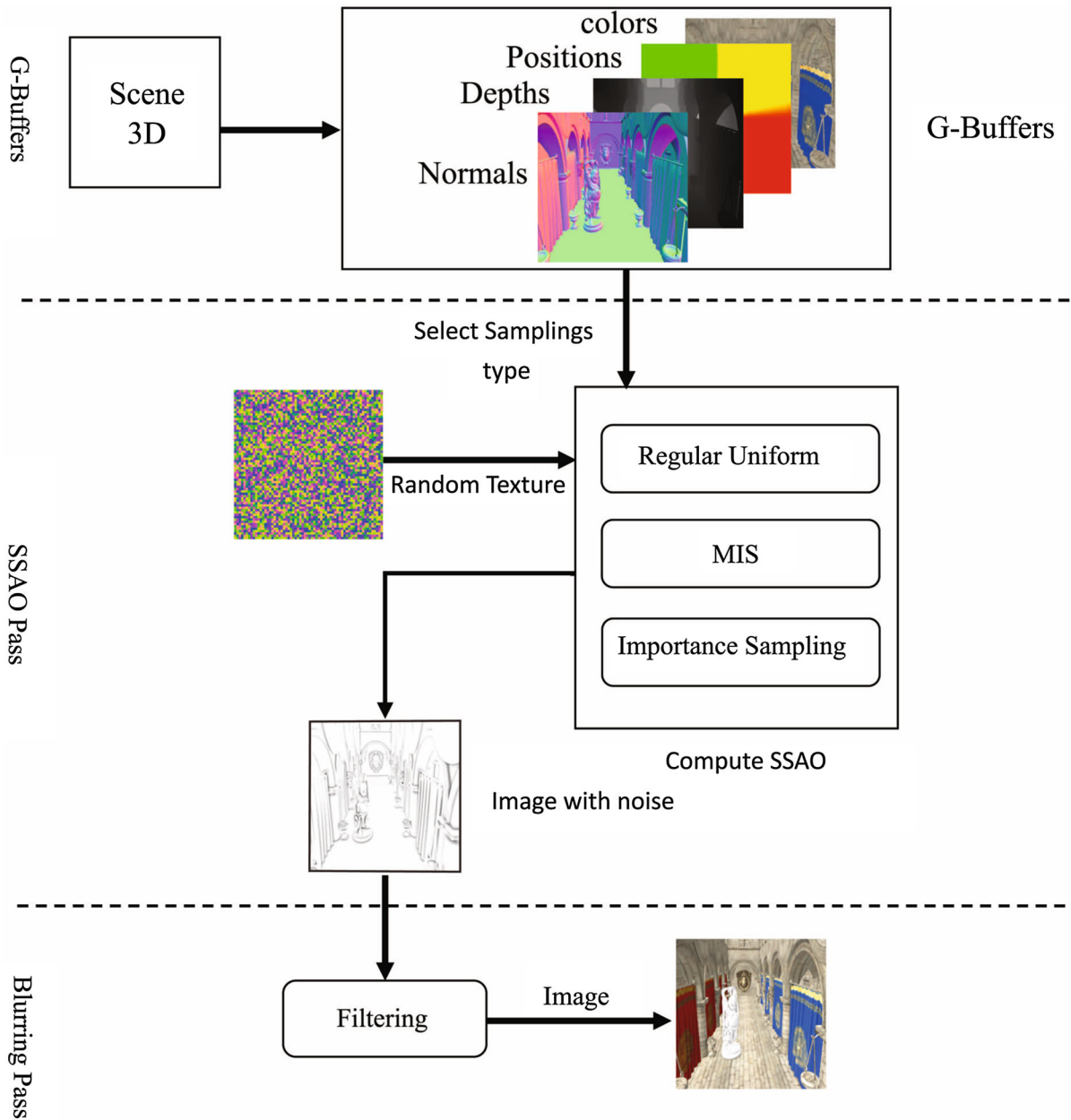
## 2 Related Work

Thomas Luft et al. [11] were the first to introduce the screen-space concept, where the scene is present in the depth buffer. A screen-space representation, which includes the normal, position, depth and attributes of each pixel, is an efficient representation of a 3D scene. Rasterization can easily create this representation, and it can make the complexity of rendering almost geometrically independent. The geometry is represented as 3D points that are rasterized by the GPU from a selected viewpoint. Usually, screen-space methods consist of two or more steps, where the scene is rendered from the light source position or the camera position, and the attributes of visible points are calculated in the first step. This method was first used for the calculation of ambient occlusion for dynamic scenes in real-time in the methods of [1–6] and [7]. The SSAO method uses a depth map to store the depth information and uses other buffers to store other information that is useful for calculating the AO factor of the SSAO. This factor is only an estimate of a set of samples that are distributed in the hemisphere around the location of the current pixel. This simulates the traditional method of ray tracing in the screen-space.

The approximation of the ambient lighting was introduced by Zhukov et al. [12] to consider the local geometry of the scene. Landis [13] calculated the weight of the AO in progressive depth perception using contact geometry and soft shadows; however, this method is limited to static scenes. Two types of algorithms are used to calculate ambient occlusion: object-space algorithms and screen-space algorithms. Object-space ambient occlusion techniques include all the geometric information in the AO calculation. Bunnell [14] presented an entirely geometric method, where the scene is pre-processed into a set of disks; then, it computes ambient occlusion analytically between these disks. Several object-space techniques exist in the literature [15, 16] and [17]. While offering high-quality results, the complexity of these methods depends on the number of polygons in the scene and the number of projected radii.

The first Screen-Space Ambient Occlusion technique was introduced in the CryEngine 2 game engine, which was developed by Crytek in 2007 [1]. It works by sampling points that are randomly distributed in a sphere at each pixel. The main feature of this technique is the grayish images that it produces.

Filion and McNaughton [4] propose a similar method to that of Crytek [1], which is called the Starcraft II SSAO method. They use a hemisphere instead of a sphere to calculate the ambient occlusion factor and the sample points are in the coordinate system of the screen, not in the global coordinate system.



**Fig. 1** Overview of our SSAO technique for computing the interactive global illumination at each point using different types of variance reduction methods

Another method is called Horizon-Based Ambient Occlusion (HBAO), which was proposed by Bavoil et al. [7] and uses the angle of the visible horizon to approximate AO. The idea is to find a maximum horizon angle at which light can reach the sample point. From each point that is centered in the hemisphere of the screen-space, multiple regularly spaced directions are generated. Then, using the ray-marching method [18], several samples are taken along these directions. For each sample, a test is performed with the depth buffer to find the value of the horizon angle in the tangent space. Then, the final average horizon angle is used to estimate the value that contributes to the AO. Mittring proposed the Another Horizon-Based Approach in Unreal Engine 4 SSAO method [2]. Instead of attempting to approach the horizon angle with ray-marching (as in work of Bavoil), several sampling angles are generated and averaged; such samples are produced in a uniformly random manner. This allows more realistic results to be obtained compared to the Crytek method.

McGuire et al. [6] present a work entitled “The Alchemy Screen-Space Ambient Obscurance algorithm”. From the current point, a set of image-space disk samples is generated and then projected to the depth buffer to estimate the value of the AO. This method intelligently chooses a falloff function to cancel certain terms in the calculation of the ambient occlusion factor.

Marc Sunet et al. [3] in 2016 presented a comparative study of SSAO techniques, in which four SSAO methods were considered and their strengths and weaknesses were cited. The Alchemy method [6] is the best of these four methods. The Mittring method [2] also provides excellent results in terms of quality and visualization. Recently, efforts were made to port Ambient Occlusion to mobile devices [3]: Crytek [1], Starcraft [4], Alchemy [6] and HBAO [7]. In addition, a new formulation of the ambient occlusion technique was introduced, which is called “Ground-Truth Ambient Occlusion (GTAO)” [5]. The primary purpose of this method is to use an alternative formulation of the ambient occlusion equation and efficiently implement the distribution of the computation by using spatio-temporal filtering, thereby guaranteeing better realism and faster computing speed. This method allowed the generation of an image with a very high level of visual quality. Because of spatio-

temporal filtering, this method is more time-consuming than other techniques [1, 2, 4, 6] and [7].

Importance Sampling is introduced into the SSAO method in the approach of Multiple Depth Layers by Vardis et al. [19]. They improve the estimate of ambient occlusion by using depth information from different views, which was obtained using the shadow map [20] for each view. Importance sampling is used according to the importance of each view, rather than according to the importance of each sample. This method yields good visual results, but at the expense of computing speed.

The goal of this paper is to implement a screen-space ambient occlusion method that is based on Monte Carlo sampling methods and selects a set of samples with a higher contribution through variance reduction methods. For this purpose, our contribution is a high-quality real-time rendering technique that combines the stratified sampling method and the importance sampling method through the multiple importance sampling method to efficiently choose the most relevant samples. We compare our technique to those that were proposed by Mittring [2] and McGuire [6].

### 3 Proposed Technique

#### 3.1 Overview of Our Technique

In this section, we present our technique (Fig. 1). Our main contribution is based on the SSAO model, which aims to reduce the number of samples and avoid artifacts by introducing variance reduction techniques such as stratified sampling, importance sampling, and multiple importance sampling, and increase visual quality by showing more details where direct light does not reach, without a loss of performance. First, the scene is rendered from the camera viewpoint. We create a G-Buffer, which contains the normals, positions and colors of all visible points. Then, we calculate the SSAO value at each point. Afterward, we apply a shading model and, finally, filter the result.

The diagram in (Fig. 1) shows a general overview of our system, which illustrates the relationships between the different phases of our algorithm. Our system requires three shader programs: a shader for generating the G-Buffer, a shader that calculates the

SSAO and shading, and a shader for the filtering step. The steps are detailed in the next subsections.

### 3.2 Description of Our Technique

In this section, we propose a technique that provides a faster and smoother approximate global illumination solution compared to other methods. Algorithm 1 shows an overview of the proposed solution. We introduce our strategy for sampling the hemisphere and, more importantly, our sampling evaluation criterion. Additionally, we discuss possible ways to reduce the (potentially) significant amounts of kernel sampling. We begin with a description of our algorithm. Then, we summarize the key terms and equations that are used for calculating indirect illumination.

---

#### Algorithm 1 General technique of our application

---

```

1:First rendering pass: Generate the G-Buffer from the eyes
  view.
2:Second rendering pass: Compute SSAO and Shading.
3:for Each visible fragment of the scene do
4:   Generate a Sample Kernel.
5:   Retrieve the geometry (Depth, normal).
6:   Calculate SSAO factor.
7:   Choose the lighting model.
8:   Save the result in texture.
9:end for
10:Third rendering pass: Blurring stage.

```

---

#### 3.2.1 First Rendering Pass: Create the G-Buffer

This pass aims at approximating the 3D geometry as a set of 2D maps, i.e., computing G-Buffer. In a post-processing pass and from the viewers position in the scene, we safeguard the depth information, normal, position, and color in the Frame Buffer as different textures. Then, we compute the G-Buffer using a shader program. These values will be communicated to the next steps in the form of a uniform variable. Next, we use this information for the computation of SSAO, lighting, and filtering in various rendering passes. This pass is fully implemented in the GPU (see Fig. 1).

#### 3.2.2 Second Rendering Pass: Compute SSAO and Shading

The SSAO calculation is the most important pass in this work in terms of our contribution. We begin by presenting our hemisphere sampling strategies in Algorithms (2, 3, 4, 5 and 6) for estimating the SSAO value. We describe the shading model that is used to refine our scene. Then, we detail the type of filter that is used to reduce the noise that is caused by the under-sampling step.

##### 3.2.2.1 Generate a Sample Kernel Based on MIS

We want to produce samples that are distributed in a hemisphere that is oriented around the normal to the surface. Since it is difficult to generate a sample kernel for each direction of the normal to the surface, we produce a sample kernel in the tangent space, with a normal vector pointing in the positive z-direction. We need a minimum number of samples for realistic results. Banding artifacts may appear due to the distribution of samples. By introducing a sample rotation kernel for each fragment, we can significantly reduce the banding artifacts. Most SSAO methods are based on a uniform random distribution of samples.

To estimate the Monte Carlo integral, samples must be taken from the hemisphere [13]. The speed and efficiency of the calculation of SSAO are related to the sampling techniques that are used. The main problem with the Monte Carlo method by uniform sampling is its computational cost due to slow convergence. A good Monte Carlo estimator needs to reduce the variance. There are several popular techniques for reducing the variance of the estimator, such as stratified sampling, preferential sampling [9] and multiple importance sampling [10].

Our technique aims to further reduce the number of samples by introducing our sampling strategies: importance sampling (Algorithm 2), stratified sampling (Algorithm 3) and our multiple importance sampling technique (Algorithm 4), which combines the first two methods.

The estimator for importance sampling as described by [9]:

$$\langle I \rangle = \frac{1}{N} \sum_{i=1}^N \frac{f(x_i)}{p(x_i)} \quad (1)$$

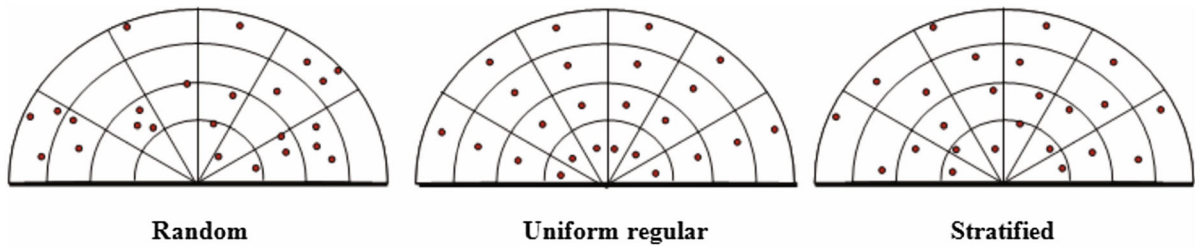


Fig. 2 Different sampling strategies with 24 samples

where  $x_i$  is a sample from the integration domain,  $N$  is the number of samples,  $p(x)$  is probability density function (PDF) in  $[0,1]$ , and  $f(x)$  is a function to be sampled, which is defined over the integration domain. Thus, the areas of the function  $f(x)$  that have the highest values will be privileged during the sampling. For this, we need to calculate the cumulative distribution function (CDF)  $P(x)$  of a real-valued random variable  $X : P(x) = Pr\{X \leq x\}$ . This can be done by uniformly choosing a value  $\xi \in [0, 1]$  and computing  $P^{-1}(\xi)$  [9],  $P(x \in [a, b]) = \int_a^b p(x)dx$  [10].

Stratified sampling [8, 21] consists of dividing the integration domain  $D \in [0,1]$  into  $n$  non-overlapping regions  $D_1, D_2, D_3, \dots, D_n$ . Each region is called a stratum. We generate a random sample within each of these strata. These regions must completely cover the area of origin. In each of the sub-domains, the integral must be evaluated separately. The Monte Carlo estimate is given by:

where  $N_i$  is the number of subdomains,  $N_j$  is the number of samples in subdomain  $i$ , and  $v_i = \int_{D_i} (\frac{f(x)}{p(x)})p(x)dx$ .

(a) Multiple Importance Sampling

Multiple importance sampling can significantly increase the robustness of Monte Carlo integration. It uses several sampling techniques to estimate an integral and then combines these samples to converge to the optimal [10].

We use two Monte Carlo estimators of the integral of  $f(x)$  : the first with sampling distribution PDF  $p_1(x)$  and the second with sampling distribution PDF  $p_2(x)$ :

1. Importance Sampling: using Monte Carlo method sampling with a cosine PDF (see 6); to sample a hemisphere that is placed above a current point.
2. Stratified Sampling: using Monte Carlo method sampling with  $1/2\pi$  as the PDF (see 7);

---

**Algorithm 2** ImportSampl(STRATUM stratum)

---

*// Generate samples according to the Importance Sampling strategy, where PDF = cos(θ)/π.*

```

1: for the current stratum do
2:   Csi1 = random();
3:   Csi2 = random();
4:   // (θ, φ) are the spherical coordinates
5:   THETA = arcsin(Csi1);
6:   PHI = (2 * π * Csi2) ;
7: end for
8: return vec2 (THETA ,PHI) ; // vec2: vector in 2D

```

---

$$\langle I \rangle = \sum_{i=1}^{n_i} \frac{v_i}{N_i} \sum_{j=1}^{N_j} \frac{f(x_j)}{p(x_j)} \tag{2}$$

Algorithm 2 presents the importance sampling strategy, in which  $\cos(\theta)/\pi$  is used as a PDF [22], and  $P(x)$  denotes the cumulative distribution function. We choose uniformly values  $\xi_i \in [0, 1]$  and compute  $P^{-1}(\xi_i)$ . The current stratum is the area between

angles 1 and 2 of the hemisphere. The pairs  $(\theta, \phi)$  are the spherical coordinates (elevation angle, azimuthal angle) that represent the direction of a ray.

We use samples pairs  $(\theta, \phi)$  (elevation angle, azimuthal angle) to represent the direction of a ray.

**Algorithm 3** StratifSampl(STRATUM stratum)

// Generate samples according to the Stratified Sampling strategy, where PDF =  $1/2\pi$ .

```

1: for the current stratum do
2:   Csi1 = random();
3:   Csi2 = random();
4:   //  $(\theta, \phi)$  are the spherical coordinates
5:   THETA = arccos (1 - Csi1);
6:   PHI = (2 *  $\pi$  * Csi2) ;
7: end for
8: return vec2 (THETA , PHI) ; // vec2: vector in 2D
    
```

Algorithm 3 presents the stratified sampling strategy that is used in our approach for generating privileged samples, in which  $1/2\pi$  is used as a probability density function [22]. The point  $(x, y)$  represents the sample position in the hemisphere, e.g., as shown in Fig. 2 on the right.

We are inspired by the Veach strategy [10], which consists of combining the two Monte Carlo estimators to obtain  $n_i$  samples of  $p_i(x)$  among  $n$  pdfs. In our case, the multiple importance sampling estimator is simply:

$$F = \frac{1}{n_1} \sum \omega_1(X_{1,j}) \frac{f(X_{1,j})}{p_1(X_{1,j})} + \frac{1}{n_2} \sum \omega_2(X_{2,j}) \frac{f(X_{2,j})}{p_2(X_{2,j})} \tag{3}$$

To combine the two estimators, we use a weighting function [10]. The weights ( $\omega_1$  and  $\omega_2$ ) that are given by this function make it possible to generate samples  $X_{1,j}$  or  $X_{2,j}$  whose purpose is to reduce the variance. The goal is to find the estimator  $F$  with minimal variance by choosing the weights appropriately. Veach [10] suggests the use of the two heuristic equilibrium weights that are associated with each strategy:

$$\omega_1(X_{1,j}) = \frac{p_1(X_{1,j})}{p_1(X_{1,j}) + p_2(X_{1,j})} \tag{4}$$

$$\omega_2(X_{2,j}) = \frac{p_2(X_{2,j})}{p_1(X_{2,j}) + p_2(X_{2,j})} \tag{5}$$

where the sum of the weighting functions must be equal to one.  $X_{1,j}$  is the sample of the random variable  $x$  that is generated with PDF  $p_1$ , and  $X_{2,j}$  is the sample of the random variable  $x$  that is generated with PDF  $p_2$ .

$$p_1(X_{1,j}) = \frac{\sin(\theta)\cos(\theta)}{\pi} \tag{6}$$

$$p_2(X_{2,j}) = \frac{\sin(\theta)}{2\pi} \tag{7}$$

where  $\theta$  is the polar angle that is formed by the normal and the sample ray at the current point.

(b) Description of our technique based on MIS

In this section, we explain our technique, which uses the MIS method to weight the samples. We show how to combine the two estimators (the importance sampling estimator and the stratified sampling estimator). We use the balanced heuristic approach, as described in Sect. (3.2.2.1), to calculate the weights. Our main aim is to compute the PDF of each strategy when we generate samples ( $p_1(X_{2,(i,k)})$  and ( $p_2(X_{1,(i,k)})$ )).

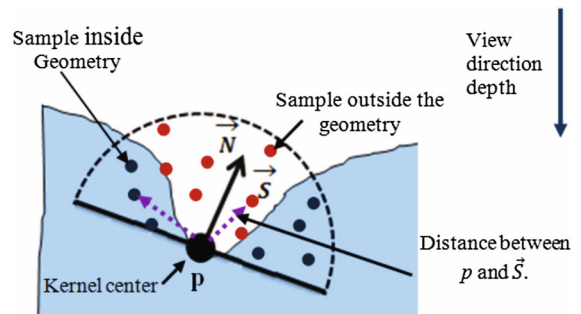


Fig. 3 Sampling around the hemisphere

Algorithm 4 is a general overview of our proposed technique.

Algorithm 6 shows how to calculate  $(p_1(X_{2,(i,k)}))$  and calculate the weight  $\omega_2$  from two different PDFs,

---

**Algorithm 4** Multiple Importance Sampling strategy

---

```

1: Divide the hemisphere into subdomains of  $n$  strata.
2: for each stratum of the domain do
3:   for each point  $(i,k)$  from the current stratum do
4:     //  $pdf1 = p_1(X_{1,(i,k)})$  ,  $pdf2 = p_2(X_{1,(i,k)})$ 
5:     ContribImportS = MISImportSampl(pdf1,pdf2);
6:     //  $pdf3 = p_2(X_{2,(i,k)})$  ,  $pdf4 = p_1(X_{2,(i,k)})$ 
7:     ContribStratifS = MISStratif(pdf3,pdf4);
8:     // Final contribution
9:     MISSample[i] = ContribImportS + ContribStratifS;
10:  end for;
11: end for;

```

---

Algorithm 5 illustrates how to calculate  $p_2(X_{1,(i,k)})$  and the weight  $\omega_1$  from two different PDFs, namely (pdf1, pdf2). First, we use Algorithm 2 to generate a direction that is defined by polar coordinates  $(\theta, \phi)$  and calculate the value of pdf1 according to  $(\theta, \phi)$ . Second, we use Algorithm 3 to generate another direction that is defined by polar coordinates  $(\theta_1, \phi_1)$  and we calculate the value of pdf2 according to  $(\theta, \phi)$ . Third, we calculate the value of the first weight  $\omega_1$  (see Formula 4).

namely (pdf3, pdf4). First, we use Algorithm 3 to generate a direction that is defined by polar coordinates  $(\theta, \phi)$  and we calculate the value of pdf3 according to  $(\theta, \phi)$ . Second, we use Algorithm 2 to generate another direction that is defined by polar coordinates  $(\theta_1, \phi_1)$ , and we calculate the value of pdf4 according to  $(\theta, \phi)$ . Third, we calculate the value of the first weight  $\omega_2$  (see Formula 7).

---

**Algorithm 5** MISImportSampl(PDF pdf1, PDF pdf2)

---

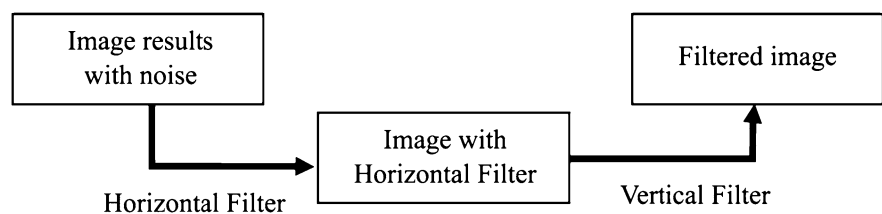
```

1: //  $pdf1 = p_1(X_{1,(i,k)})$  ,  $pdf2 = p_2(X_{1,(i,k)})$ 
2: vec2(THETA,PHI) = ImportSampl(current stratum) ; // Algorithm 2
3: pdf1 = (cos(THETA) × sin(THETA))/π; // see (6)
4: vec2(THETA1,PHI1) = StratifSampl(current stratum) ; // Algorithm 3
5: pdf2 = (sin(THETA1))/2π ; // see (7)
6:  $\omega_1 = \text{BalanceHeuristic}(pdf1,pdf2)$ ; // see (4)
7: ContribImportS =  $\omega_1 \times \text{estimator1}$ ; // see (1)
8: return ContribImportS;

```

---

**Fig. 4** Bilateral Blur Filter steps





**Algorithm 6** MISStratif(PDF pdf3, PDF pdf4)

---

```

1: // pdf3 = p2(X2,(i,k)) , pdf4 = p1(X2,(i,k))
2: vec2(THETA,PHI) = StratifSampl(current stratum) ; // Algorithm 3
3: pdf3 = ( sin(THETA))/ 2π ; // see (7)
4: vec2(THETA1,PHI1) = ImportSampl(current stratum) ; // Algorithm 2
5: pdf4 = (cos(THETA1) × sin(THETA1))/π ; // see (6)
6: ω2 = BalanceHeuristic(pdf4,pdf3); // see (7)
7: ContribStratifS = ω2 × estimator2; // see (2)
8: return ContribStratifS;

```

---

**3.2.2.2 Compute SSAO** In this step, we compute the ambient occlusion factor by using our sampling strategies. This stage is fully implemented in the GPU with the shader program through the following steps:

*In the Vertex Shader* We perform the geometric transformations to determine the position of each vertex in the proper space, and we compute the texture coordinates that are required to retrieve the G-Buffer information. We calculate the vertex positions in world-space. The value of the position map is represented by an RGB color, where each component is in the range [0, 1]. Each vector component is in the range [- 1, 1], so the conversion of a normal RGB texel is performed by the following formula: (RGB × 2.0 - 1.0).

*In the Fragment Shader* The G-Buffer contents (Normal, Depth, and Position) are sent to the fragment shaders as uniform variables to convert the geometric information of the screen-space to the world-space. Then, a sampling in the hemisphere is performed according to the selected sampling method to calculate the ambient occlusion factor.

$$AOF = \frac{1}{N} \sum_{i=1}^N V(p, \omega) \quad (8)$$

where  $N$  is the number of samples,  $AOF$  is the ambient occlusion factor, and  $V(p, \omega)$  is the (inverse) binary visibility function of a ray from  $p$  in direction  $\omega$ .

We use AO as defined by Landis [13] (Eq. 8) to estimate the SSAO factor value (SSAOF), where we approach the projected rays in the hemisphere using a selected sampling strategy. For each fragment, we

obtain the depth, normal and position information from the G-Buffers. The values of the normal and position that are stored in the G-Buffers are in the range [0, 1] (texture coordinates), and they must be transformed into vectors that are in the range [- 1, 1] in world-space coordinates. We obtain the depth, normal and restored position information in the world-space, which will be used to calculate the ambient occlusion factor.

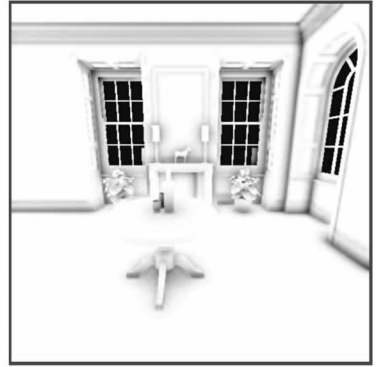
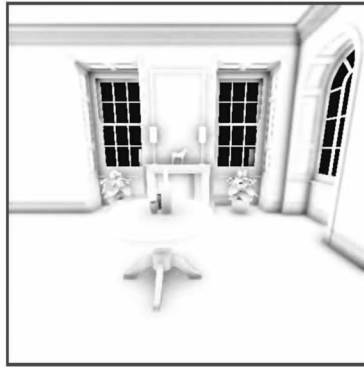
For each sample in the hemisphere and according to the chosen sampling strategy, we retrieve the depth of each sample from the Depth-Buffer. Then, we compute the position of each sample with respect to the kernel center position (current vertex  $p$ ). We obtain the position for each sample in the hemisphere that is centered at point  $p$ . Next, we normalize the normal vector  $\vec{N}$  and the distance vector  $\vec{S}$  (between the center kernel  $p$  and the sample point) and compute the scalar product between them, as shown in (Fig. 3). From this value, we can determine whether the sampled point is inside or outside the geometry. If this point is inside the geometry, we increment the ambient occlusion factor value; otherwise, this point does not contribute to the value of the AO factor.

After calculating the AO factor, we need to illuminate our scene to improve the realism. We apply the Phong illumination model [23] for local illumination because it is fast, simple and widely used by many image synthesis systems. It combines three elements: ambient light, the diffuse model, and the specular model. We apply this illumination model for two different cases: First, we use Phong illumination model with local illumination using uniform ambient light (as illustrated in Fig. 6 on the left). Second, we apply Phong illumination model with indirect

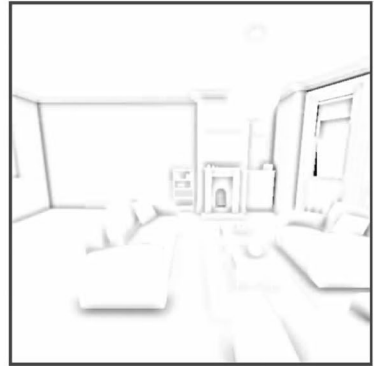
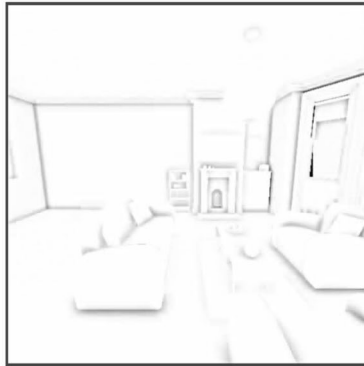
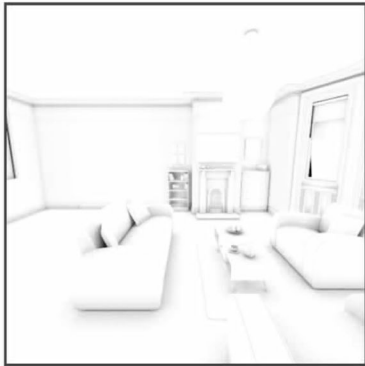
**Dragon**



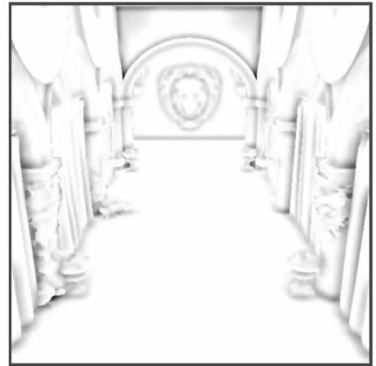
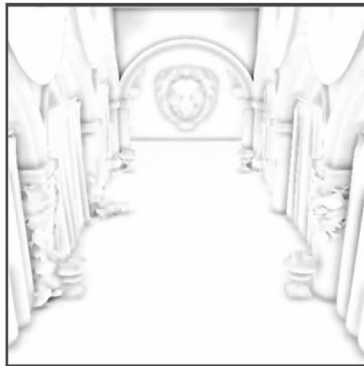
**Fireplace Room**



**Living Room**



**Sponza**



◀ **Fig. 5** Selected qualitative results for several scenes and techniques. Left: AO computed by path tracing (reference image) with 256 spp. Middle: SSAO with regular uniform sampling with 24 spp. Right: our SSAO technique with multiple importance sampling with 24 spp

illumination by replacing the value of the uniform ambient light by the ambient occlusion factor AOF (as illustrated in Fig. 6 on the right). After calculating the ambient occlusion factor AOF and lighting, we save the result in a Frame Buffer for filtering and display it on the screen.

### 3.2.3 Third Rendering Pass: Blurring Stage

The visual quality of the results increases with the number of samples that are generated; however, the number of samples negatively affects the frame rate. To obtain a suitable frame rate, the number of samples must be minimized. However, this reduction in the

number of samples produces banding artifacts (noise) in the result. It is simple to remove this noise using a blur rendering pass. The Bilateral Blur Filter [24] is used to solve this problem because this filter takes into account the depth values and does not blur the edges. Two rendering passes are necessary to apply this filter: a first pass through the horizontal filter and a second pass through the vertical filter, as described in Fig. 4.

## 4 Results and Discussion

To evaluate our technique, we want to discuss the results in terms of quality and performance. The system on which we tested our technique consisted of an Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz, Nvidia GeForce GTX 1070/PCIe/SSE2, with 32 GB RAM on a Microsoft Windows 8.1 64-bit OS using OpenGL 4.5. To compare our results, we chose the standard FPS (Frame Per Second) and time

**Table 1** Specifications of each scene and computing times of different objects using the path tracing method with 256 spp

| Scene             | Number of triangles | Calculation time using path tracing with 256 spp (s) |
|-------------------|---------------------|--|
| Sibenik Cathedral | 75,284              | 39   |
| CrytekSponza      | 262,195             | 87   |
| Fireplace room    | 143,173             | 51   |
| Living room       | 580,637             | 221  |
| Dragon            | 100,002             | 44   |
| Bunny             | 69,688              | 36   |
| Buddha            | 100,000             | 43   |
| Lucy              | 33,446              | 25   |



**Fig. 6** With 24 spp. Left: Illuminated by Phong shading using uniform ambient light (51 FPS). Middle: Illuminated by Phong shading using the Mittring [2] SSAO method (47 FPS). Right: Our SSAO MIS technique with Phong shading (47 FPS)

calculation criteria for the generation of an image. Performance changes were measured as the number of sampling kernels was varied from 16 to 256. The images that were generated by our technique were compared to the reference images, which were calculated with the path tracing method (Ground Truth) using different numbers of samples per pixel (spp). All the images that were generated by our method are HDR images; the resolution of the calculated images is  $512 \times 512$  (HDR). Our HDR images were mapped by the Reinhard operator [25] for visualization on LDR screens.

We evaluate our techniques by using the RMSE (Root Mean Square Error) metric [26] and the HDR-VDP-2 [27] metric.

We validate our results qualitatively and quantitatively on images of scenes to be rendered in real time. Our test scenes, which contain various objects, are shown in [28, 29].

Figure 5 shows selected results from multiple methods and scenes. The images in the left-most column are the path-tracing results versus (reference images), based on which we measure the others. The reference images use the same number of samples (256 spp). The second column contains images that were obtained with the Mittring method [2] SSAO technique with 24 spp (Regular Uniform Sampling). The right column shows images that were obtained by our proposed technique, which uses multiple importance sampling with 24 spp. Note that for the Mittring method and our technique, the computation time for each scene is the same. The calculation times for the reference images are listed in Table 1. The soft shadow approximation is the result of this technique; the soft shadows appear clearly and add extra realism to our 3D scene. Our technique achieves better visual quality compared to the Mittring method. In the right image, the soft shadows are more realistic than in the middle image. To take into account the interactions between light and 3D objects, we must add a light source and a shading model to increase the realism of our scene. Shading with the Phong model in real time is a possible solution (see Fig. 6).

Figure 6 shows a scene that is illuminated by Phong shading with 24 spp, with uniform ambient light

**Fig. 7** Comparison of rendered images using 24 spp, obtained by different methods. **a** Mittring method. **b** Original Alchemy method. **c** Original HBAO. **d** Our technique

running at 51 FPS (left), with the Mittring method running at 47 FPS (middle) and with our technique running at 47 FPS (right). The addition of textures in this scene remarkably improves the aesthetics. The approximation of soft shadows is very easy to identify in the right image. Our technique produces results of higher visual quality than the Mittring method, with the same frame rate.

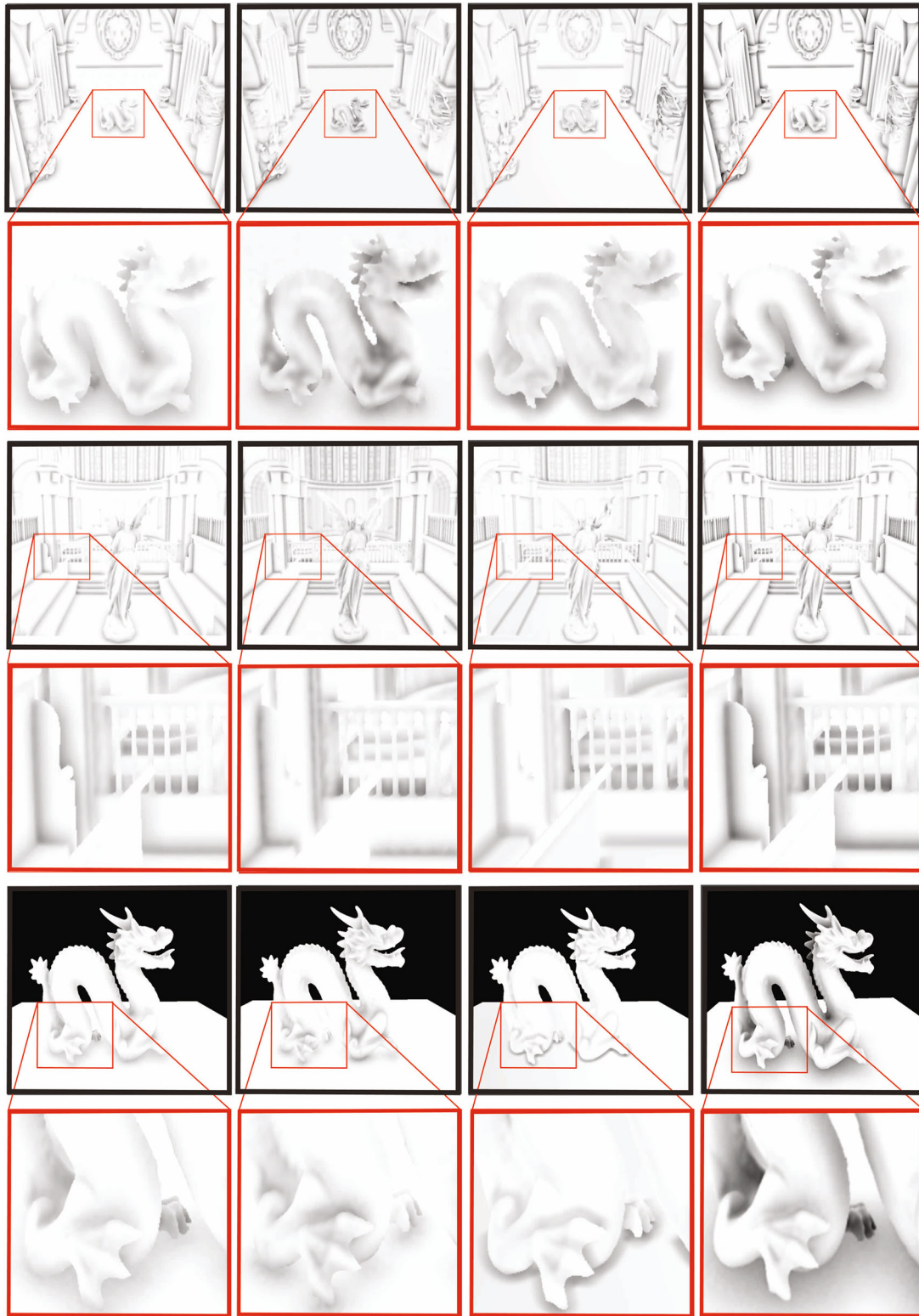
The images rendered with a small number of samples may reveal highly visible artifacts in areas where ambient occlusion occurs. The bilateral filter is used to reduce this noise (artifacts) (see Sect. 3.2.3).

A comparative study of the images rendered by different methods (Mittring, original Alchemy, original HBAO and our SSAO MIS technique) and by different scenes is illustrated in Fig. 7. According to these images, the visual quality of our results is better than other works. The soft shadows generated by our technique are more visible and more realistic.

Now we evaluate our technique by using the RMSE metric and the HDR-VDP-2 metric, which is a perceptual metric that is applied to HDR (High Dynamic Range) images.

Figure 8 shows the RMSE values that were obtained by comparing a reference image for each scene with an image that was generated by the Mittring method and an image that was created by our MIS technique. Note that the reference image is generated by path tracing using 256 spp and that the images that were rendered by the Mittring method and by our MIS technique use 24 spp. The RMSE reaches a smaller value for our technique than for the Mittring method on most scenes. Thus, our technique outperforms the Mittring method.

In Table 2, we evaluate our high-sampling technique using the RMSE metric, which represents the sample standard deviation of the differences between the predicted values and the observed values for our images. All images that were generated by our technique are compared to images that were created by the Mittring method (using a uniform sampling



Same FPS= 49

Same FPS= 129

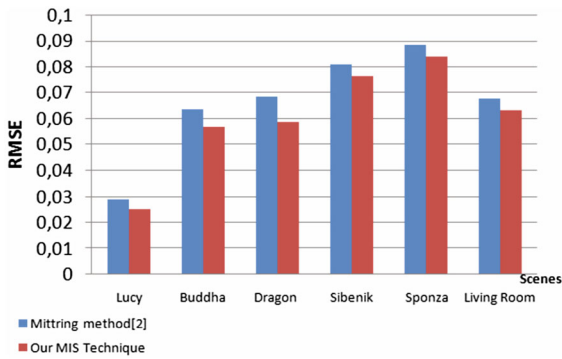
Same FPS= 181

(a)

(b)

(c)

(d)



**Fig. 8** RMSE results for different methods using 24 spp, according to various rendering scenes. The reference images are generated by the path tracing method using 256 spp

method). Note that in this case, the images that were generated by the Mittring method are considered the reference images, which use 128 spp. All images that were generated by our technique use 16 spp. A visual evaluation of these results is presented in Fig. 9. According to the values of the RMSE for most rendered scenes, we conclude that images that were obtained by our technique, which uses 16 spp, closely resemble the reference images, which were rendered using 128 spp. In addition, the computation time of our technique is smaller than the computation time of the Mittring method.

Figure 9 shows the perceptual differences between the images that were generated with our SSAO MIS technique and the reference images. We evaluate the perceptual differences by applying the HDR-VDP-2 perceptual metric to the HDR images. Notice that the

**Fig. 9** HDR-VDP-2 metric between the reference images and those obtained with our technique for the four test scenes with 24 spp. Left: Reference images. Middle: Images that were generated with our MIS technique. Right: Images that represent the HDR-VDP-2 metric between the reference images (left) and the images that were generated by our SSAO MIS technique (middle)

reference images, in this case, are produced with the regular uniform sampling method by using many samples (128), while the images that are rendered by our technique use only 16 samples. Table 2 lists the frame rates of these different images that are obtained with different methods. We observe that most of the images of the test scenes have low error according to the metric HDR-VDP-2 (the green and blue fields in the images in the right columns).

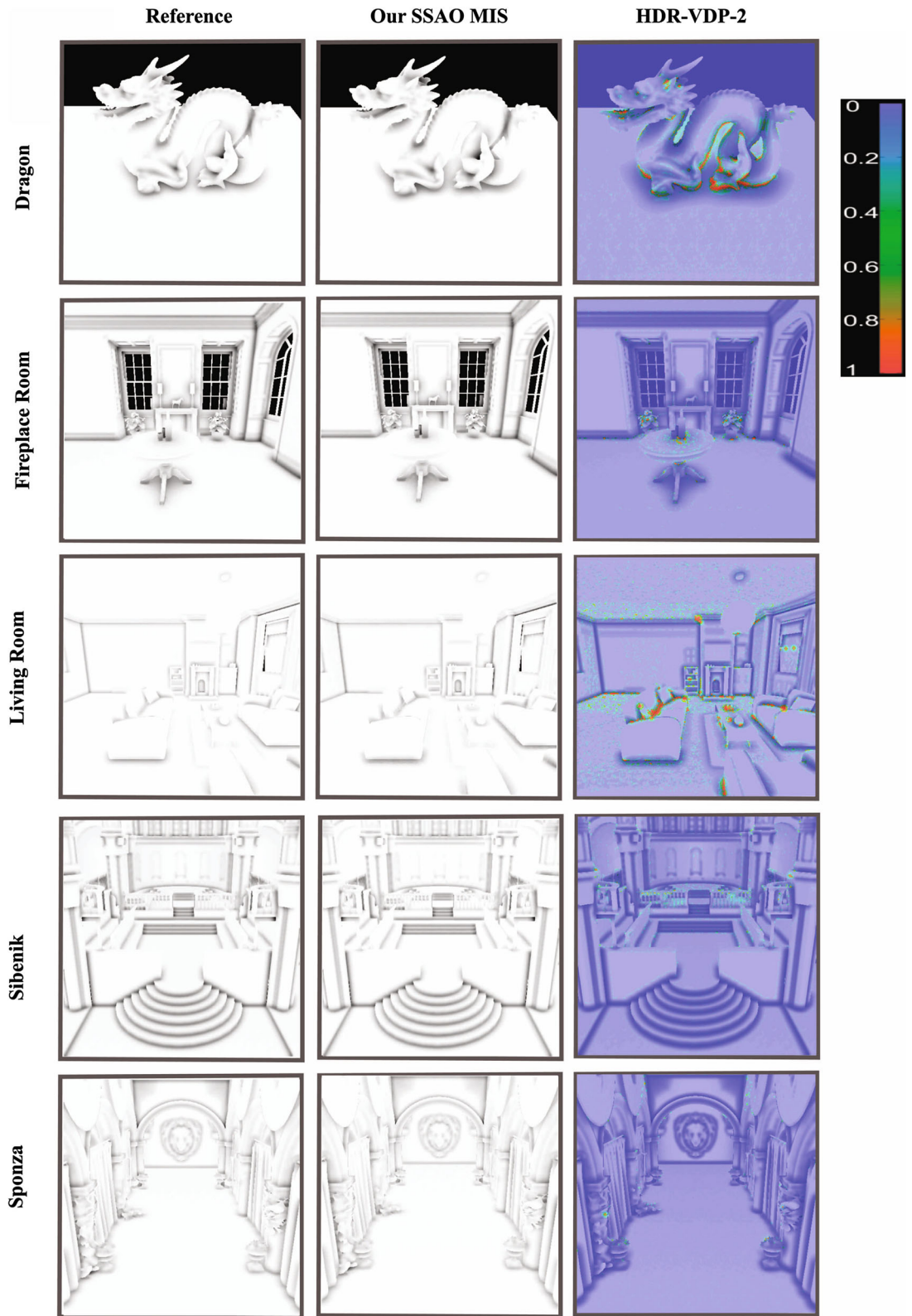
The frame rate that is required for our technique is higher than the frame rate of the reference image (Mittring method). Consequently, our technique is faster and generates images that are very similar to the reference images.

### 5 Conclusions and Future Work

In this paper, we have improved the perceptual effect of the SSAO method on the quality of the image while also positively affecting the calculation time and generating a visual approximation of soft shadows, which improves the visual quality of the image. The integration of the multiple importance sampling

**Table 2** RMSE values between reference images and the images of our technique

| Method    | Scene          | Number of sampling | FPS | RMSE     |
|-----------|----------------|--------------------|-----|----------|
| Reference | Sponza         | 128                | 41  |          |
| Our MIS   | Sponza         | 16                 | 61  | 0.009226 |
| Reference | Living room    | 128                | 18  |          |
| Our MIS   | Living room    | 16                 | 28  | 0.008032 |
| Reference | Sibenik        | 128                | 98  |          |
| Our MIS   | Sibenik        | 16                 | 145 | 0.018461 |
| Reference | Dragon         | 128                | 181 |          |
| Our MIS   | Dragon         | 16                 | 210 | 0.014294 |
| Reference | Fireplace room | 128                | 71  |          |
| Our MIS   | Fireplace room | 16                 | 103 | 0.018788 |



method into the calculation of the SSAO improves the visual quality without decreasing the frame rate, which enables us to generate scenes in real time. Our technique can be integrated into several interactive areas, such as the latest video games, 3D reconstructions, medical visualization, architectural design, computer vision, and any application in which indirect lighting is required.

A limitation of our proposed technique is that it cannot be applied to point-based rendering and image-based rendering; it is only applicable to polygon-based rendering. As the scene complexity changes, the processing time grows exponentially, thereby rendering our algorithm unsuitable for interactive use.

Future works will seek to extend our study to more complex scenes by using pre-computed Ambient Occlusion textures to accelerate the calculations and improve the realism. Moreover, uniform regular sampling around the pixel can be replaced with another type of sampling to further reduce the number of samples. We also plan to replace the bilateral filter with another one that can give better visual results without affecting the computational performance. Temporal coherence can also accelerate the rendering process by saving computational time and achieving more accurate results that are beyond the capabilities of traditional SSAO techniques.

## References

- Mittring, M. (2007). Finding next gen: Cryengine 2. In *Proceeding of SIGGRAPH 2007, advanced real-time rendering in 3D graphics and games course*, Chapter 8, ACM, New York, NY, USA, (pp. 97–121). <https://doi.org/10.1145/1281500.1281671>.
- Mittring, M. (2012). The technology behind the unreal engine 4 elemental demo. In *Proceeding of SIGGRAPH 2012: 39th international conference and exhibition on computer graphics and interactive techniques*, Advances in Real-Time Rendering in 3D Graphics and Games course, ACM, Los Angeles Convention Center.
- Sunet, M., & Vazquez, P. (2016). Optimized screen-space ambient occlusion in mobile devices. In *The proceedings of the 21st international conference on Web3D technology* (pp. 127–135), Anaheim, California. ISBN:978-1-4503-4428-9. <https://doi.org/10.1145/2945292.2945300>.
- Filion, D., & McNaughton, R. (2008). Effects and techniques. In *Proceeding ACM SIGGRAPH 2008 games* (pp. 133–164). <https://doi.org/10.1145/1404435.1404441>.
- Jimenez, J., Wu, X., Pesce, A., & Jarabo, A. (2016). Practical real-time strategies for accurate indirect occlusion. In *Proceeding of SIGGRAPH 2016 Courses: Physically based shading in theory and practice* (pp. 112–161).
- McGuire, M., Osman, B., Bukowski, M., & Hennessy, P. (2011). The alchemy screen-space ambient obscurity algorithm. In *Proceeding: HPG '11 proceedings of the ACM SIGGRAPH symposium on high performance graphics* (pp. 25–32). Vancouver, British Columbia, Canada August 05–07, 2011. ISBN:978-1-4503-0896-0. <https://doi.org/10.1145/2018323.2018327>.
- Bavoil, L., Sainz, M., & Dimitrov, R. (2008). Image-space horizon-based ambient occlusion. In *Proceeding: ACM SIGGRAPH 2008 talks* (pp. 22:1–22:1) Los Angeles, California, . ISBN:978-1-60558-343-3. <https://doi.org/10.1145/1401032.1401061>.
- Rubinstein, R. Y. (1981). *Simulation and the Monte Carlo method*. New York, NY, USA: Wiley.
- Owen, A., & Zhou, Y. (2000). Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449), 135–143. <https://doi.org/10.2307/2669533>. (Taylor & Francis, Ltd. on behalf of the American Statistical Association).
- Veach, E. (1997). Robust Monte Carlo methods for light transport simulation. Ph.D. thesis. Stanford University. ISBN:0-591-90780-1.
- Luft, T., Coloditz, C. & Deussen, O. (2006). Image enhancement by unsharp masking the depth buffer. In *Journal of ACM Transactions on Graphics (TOG)—Proceedings of ACM SIGGRAPH 2006* (Vol. 25(3), pp. 1206–1213). ISBN:1-59593-364-6. <https://doi.org/10.1145/1141911.1142016>.
- Zhukov, S., Iones, A., & Kronin, G. (1998). An ambient light illumination model. In G. Drettakis & N. Max (Eds.), *Proceedings of eurographics rendering workshop 98* (pp. 45–56). Springer: Berlin, Heidelberg, New York.
- Landis, H. (2002). *Production-ready global illumination. RenderMan in production. Course 16: ACM SIGGRAPH 2002 course notes*. Boston: ACM.
- Bunnell, M. (2005). Dynamic ambient occlusion and indirect lighting. In *GPU Gems: Programming techniques for high-performance graphics and general-purpose computation*, Chapter 14 (Vol. 2(2), pp. 223–233), Addison-Wesley Professional. ISBN 0-321-33559-7.
- McGuire, M. (2010). Ambient occlusion volumes. In *Proceedings of the conference on high performance graphics* (pp. 47–56). Saarbrücken, Germany.
- Hernell, F., Ljung, P., & Ynnerman, A. (2010). Local ambient occlusion in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 16(4), 548–559. <https://doi.org/10.1109/TVCG.2009.45>.
- Grottel, S., Krone, M., Scharnowski, K. & Ertl, T. (2012). Object-space ambient occlusion for molecular dynamics. In *Pacific visualization symposium (PacificVis), 2012 IEEE* (pp. 209–216). <https://doi.org/10.1109/PacificVis.2012.6183593>.
- Amanatides, J., Woo, A., et al. (1987). A fast voxel traversal algorithm for ray tracing. *Eurographics*, 87(3), 3–10. (ISSN:1017-4656).
- Vardis, K., & Gaitatzes, G. (2013). Multi-view ambient occlusion with importance sampling. In *Proceeding I3D '13 proceedings of the ACM SIGGRAPH symposium on interactive 3D graphics and games* (pp. 111–118). Orlando,



- Florida March 21–23, 2013. ACM New York, NY, USA 2013. ISBN:978-1-4503-1956-0. <https://doi.org/10.1145/2448196.2448214>.
20. Williams, L. (1978). Casting curved shadows on curved surfaces. In *ACM SIGGRAPH '78 proceedings of the 5th annual conference on computer graphics and interactive techniques* (Vol. 12(3), pp. 270–274).
  21. Cox, P., Yang, P., Mahant-Shetti, S. S., & Chatterjee, P. (1985). Statistical modeling for efficient parametric yield estimation of MOS VLSI circuits. *IEEE Transactions Electron Devices*, 20(1), 471–478. <https://doi.org/10.1109/JSSC.1985.1052319>.
  22. Dutré, P. (2003). Global illumination compendium. In *The Journal of Computer Graphics*, Cornell University: Department of Computer Science, Katholieke Universiteit Leuven.
  23. Phong, B. T. (1975). Illumination for computer generated images. *Communications of the ACM*, 18(6), 311–317.
  24. Paris, S., Kornprobst, P., Tumblin, J., & Durand, F. (2009). Bilateral filtering: Theory and applications. *Journal of Foundations and Trends, in Computer Graphics and Vision*, 4(1), 1–73. <https://doi.org/10.1561/06000000020>.
  25. Reinhard, E., Stark M., Shirley P., & Ferwerda J. (2002). Photographic tone reproduction for digital images. In *Proceedings of SIGGRAPH 2002, ACM transactions on graphics (TOG homepage)* (Vol. 21(3), pp. 267–276). <https://doi.org/10.1145/566570.566575>.
  26. Willmott, C., & Matsuura, K. (2005). Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. *Published in Climate Research*, 30, 7982. <https://doi.org/10.3354/cr030079>.
  27. Mantiuk, R., Kim, K. J., Rempel, A. G., & Heidrich, W. (2011). Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions. In *Journal ACM transactions on graphics (TOG)—Proceedings of ACM SIGGRAPH 2011 (TOG Homepage)*, (Vol. 30(4), p. 40). ISBN:978-1-4503-0943-1. <https://doi.org/10.1145/1964921.1964935>.
  28. McGuire, M. (2017). Computer graphics archive, July 2017. <http://casual-effects.com/data/index.html>.
  29. Stanford University: The stanford 3D scanning repository, 1996. <http://graphics.stanford.edu/data/3Dscanrep/>.