CGI2021 manuscript No. 7
(will be inserted by the editor)

Efficient real-time dynamic diffuse global illumination using signed distance fields

Jinkai Hu · Milo K. Yip · Guillermo Elias Alonso · Shihao Gu · Xiangjun Tang · Xiaogang Jin ✉

Abstract We present SDFDDGI, a novel approach to compute dynamic diffuse global illumination in real time using Signed Distance Fields (SDF). For an input scene, we first construct its compact representation using SDF. Different from traditional SDF which are stored by discrete voxels, our approach approximates the scene by a set of simple primitive shapes, which facilitates real-time computation and dynamic changes. Then, we reconstruct the irradiance function in the space domain by discrete samples (referred to as probes), which are positioned heuristically for real time performance. The probe irradiance can be updated and interpolated effectively supported by our compact SDF representation. Subsequently, a screen-space refinement method is developed to enhance rendering details and visual quality. We validate our approach by comparing the performance and quality of our method to other state-of-the-art real-time global illumination methods. Our approach is able to calculate real-time diffuse global illumination for both dynamic geometry and dynamic lighting efficiently without any precomputation, while also supporting multi-bounced light. It is also hardware free, and can manage both large open scenes and indoor high-detailed scenes.

Keywords Real-time rendering · Global illumination · Signed distance fields

Xiaogang Jin, jin@cad.zju.edu.cn

Jinkai Hu · Guillermo Elias Alonso · Xiangjun Tang · Xiaogang Jin
State Key Lab of CAD&CG, Zhejiang University, Hangzhou 310058, China

Milo K. Yip
MoreFun Studio Group, Tencent, China

Shihao Gu
Independent Researcher, China

1 Introduction

Global Illumination (GI) has been a long standing goal in pursuing more photorealism in the heavy real-time rendering industry. However, the high expense of computing the rays traversing the scene, as well as the complexity of estimating physical reflection models, and reconstructing noise-free renderings, necessitate faster approximation approaches than naïve path tracing. To this end, many approaches have emerged in the past few decades, such as, the baking approach that records the precomputed lighting information into static lightmaps; extension of the baking approach that supports dynamic changes of scene elements [17]; *Precomputed Radiance Transfer (PRT)* [20], which dynamically captures soft shadows, interreflections, and caustics in low-frequency environments. In addition, researchers have begun to explore some deep-learning-based methods, such as *Deep Illumination*, which approximates GI with offline rendering quality at interactive rates [22]. Moreover, all of these approaches have limitations, either regarding variation of scene geometry or dynamic light. Existing methods also usually rely on heavy precomputation, which substantially constrains artistic choices.

Recently, some methods have been developed to manage fully dynamic global illumination, including *Voxel-Based Global Illumination* (VXGI) [4], *real-time Ray Traced Global Illumination* (RTGI) [18] and *Screen Space Global Illumination* (SSGI) [15, 19, 21]. Indeed, these methods offer achievable solutions to real-time global illumination. However, as approximation approaches, these methods pose challenges to engineering robustness. For instance, they potential produce artifacts, including light leakage and noise corruption, as well as limitations on traverse depths of light and performance issues.

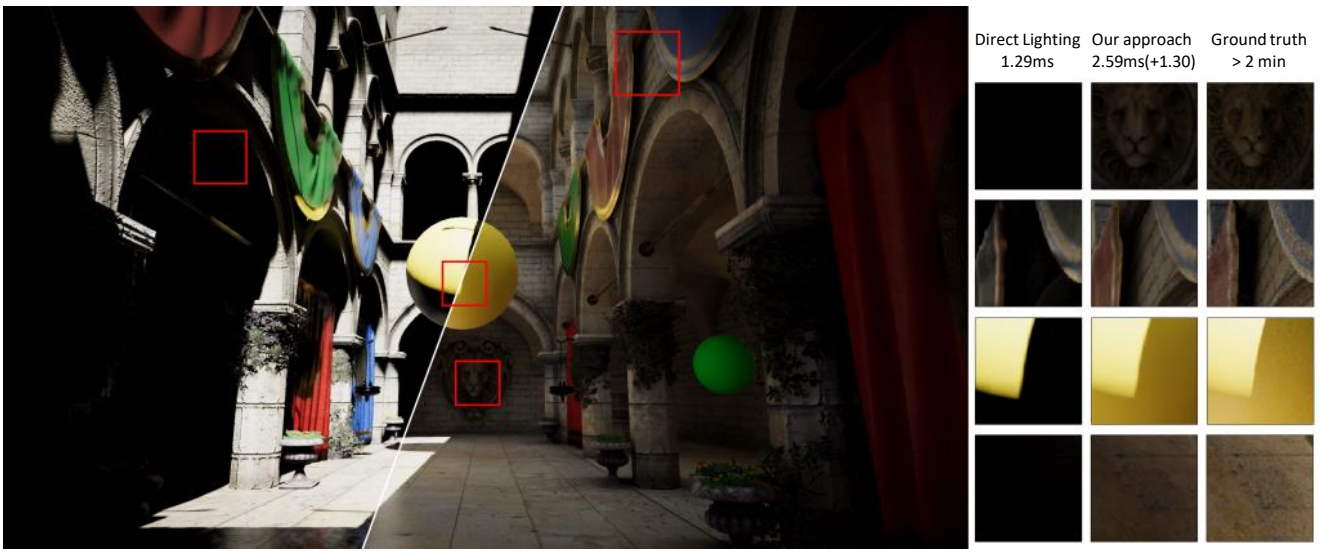


Fig. 1: *Images rendered with global illumination off/on using our SDFDDGI method for the Sponza scene. It costs only 1.3 milliseconds extra time for computing the indirect global illumination, comparing to the ground truth result rendered by an offline path tracer which costs more than 2 minutes. Our method is able to reconstruct multi-bounce diffuse global illumination and is noise-free.*

In this paper, we propose an approach to address these challenges, by employing SDF as a representation of the scene, which, as substitute for of conventional triangle meshes, accelerate the computation of GI.

In Sect. 3.1, we introduce a compact representation of the SDF. In traditional applications, the SDF is stored by discrete voxels in volume textures, which requires enormous storage and consumes a considerable expense on dynamic changes. Our compact representation, however, offers a solution to resolve this issue. Then, in Sect. 3.2, we present a probe positioning scheme based on our compact SDF, which also determines which probe should be updated. In Sect. 3.3, we describe how to update probe irradiance effectively and retain stability. In Sect. 3.4, we provide a feasible approach to perform probe visibility tests in a more accurate and less expensive manner when performing probe interpolation. In Sect. 3.5, a screen-space method is developed to enhance the final result. In Sect. 4, we compare the performance and quality of our algorithm to other state-of-the-art real-time global illumination methods. In summary, our paper makes the following contributions:

- A novel scene-dependent probe positioning method based on SDF that captures the spatial distribution of the irradiance function.
- A new solution to prevent light leakage, which can handle the artifacts presented in previous methods.
- An efficient screen-space refinement approach that enhances detail and quality of probe-based GI.

2 Related work

Ever since Kajiya first pioneered research of the rendering equation [9], the goal of photorealism through GI has been shaping the development of rendering. We list several recently emerged methods that attempt to meet the challenge of real-time GI.

Virtual Point Lights (VPL) [12] and Reflective Shadow Maps (RSM) [5] are two of the earliest real-time GI techniques. The main idea of VLP is the substitution of GI computation by adding virtual point light sources at areas illuminated by direct lighting. RSM, on the other hand, is based on the same idea as shadow maps. RSM not only stores depth information but also light source directions and radiant flux, and uses this reflective shadow map as global illumination. However, these possess have some known limitations.

RSM and VPL ultimately evolved into Light Propagation Volumes (LPV) [10]. LPV introduced the concept of volume in VPL and transferred illumination data across space. This method solved many of the problems of VPL and RSM, but light leakage remained a serious concern as well as some certain accuracy issues.

There are also voxelization approaches [4], which first voxelize the scene into a sparse voxel tree and then inject lighting data. This allows estimation of GI at real-time frame rates, but it also produces light leakage. Furthermore, in scenes with highly varying dynamic ge-

ometry, the computational cost of voxelization is prohibitively high.

High-end global illumination approaches, such as Ray Tracing GI [18] reconstruct world coordinates and normals out of G-buffers, and then sample the hemisphere to compute global illumination. The number of computations of this kind of approach is very large, and thus it is only viable for a small number of samples and high-end hardware. For this reason, it requires a final denoising stage [16] [11] to achieve an acceptable result. The main limitation of this method is its performance, in which it is almost impossible to calculate multi-bounce illumination, apart from the cost of additional denoising.

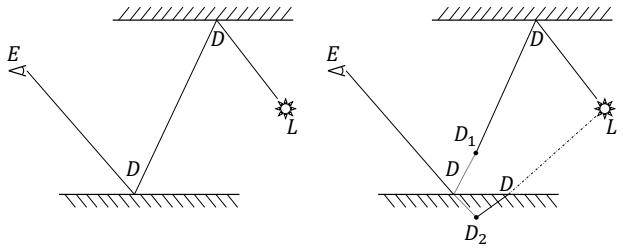
Until now, the best choice in terms of performance has been screen space GI, such as screen space diffuse GI [15] [19] or screen space specular reflections (SSR) [21]. Comparatively, necessary information for specular reflections normally resides inside of the screen space; where as diffuse GI often lacks most of the needed lighting information, and is thus unable to provide an optimal result.

Last year, NVIDIA proposed a new approach RTXGI [13] using its ray tracing accelerated hardware, by means of discretizing the spatial distribution of the irradiance function. Compared to common probe-based GI [14], its main contribution is the use of depth information and Variance Shadow Maps (VSM) [7] in order to prevent light leakage artifacts that arise from the discretization of irradiance. However, the effect of GI concerning detailed geometry is not ideal, light leakage is not completely solved and it possesses hardware limitations.

SDF, however, can be utilized to simplify the scene representation, and is useful for low-frequency GI, such as diffuse GI. SDF is a scalar field in the space domain, which represents the distance from a point in space to the nearest surface in the scene. A positive value is assigned if the point is in the outer region of the nearest surface and negative if it is inside. In this way, it produces a compact representation of the geometry information of a scene.

On the basis of RTXGI and using SDF, we propose a novel approach SDFDDGI, which overcomes the aforementioned constraints and possesses the following advantages:

- It can manage dynamic geometry, dynamic lighting and animations.
- It provides interframe stability and low delay response for dynamic changes.
- It completely eradicates light leakage problems by performing pixel-level visibility tests.
- The proposed technique is not limited to ray tracing accelerated hardware.



(a) Common LDDE path (b) Deviation of LDDE path

Fig. 2: *The discretization of space in our diffuse GI model produces deviations on the LDDE path. Instead of taking the point D on the actual path, we interpolate irradiance at nearby points D_1 and D_2 on the irradiance's discrete space domain.*

3 Approach

Our approach is inspired by the framework of RTXGI. In the pipeline of RTXGI, it first updates the scene's bounding volume hierarchy (BVH), and then updates probes by traversing the BVH to compute radiance. At the same time, it saves hit point depth information. Subsequently, RTXGI interpolates probes to compute each pixel's irradiance. In order to prevent light leakage, RTXGI uses low-resolution depth information to carry out probe visibility tests.

We adopt the same expression notation for paths as [23], but we use L and E to represent any type of light and camera, respectively. RTXGI makes approximations on the D-D part of the LDDE path by moving the first D vertex to the position of the probe, which causes light leakage problems since visibility may be different from these points, as shown in Figure 2. To avoid such artifacts, RTXGI uses VSM with the aforementioned low-resolution depth information. However, it has limitation of resolution in that one probe only has a resolution of 16×16 to map the whole sphere of directions. Moreover, VSM itself produces light leaking artifacts. As a consequence, it cannot completely eradicate the problem.

As Figure 3 shows, our SDF representation, which is composed of simple SDF primitives, can be considered as an approximation of the scene geometry. These primitives are culled according to the position of the camera, and put together into clusters by proximity. This SDF representation is also used to adjust the positions of probes to prevent unsuitable sampling points, and to decide which probes are turned on or off. Then, we traverse the SDF to obtain intersections from the probe, and sample RSM to compute the radiance. In

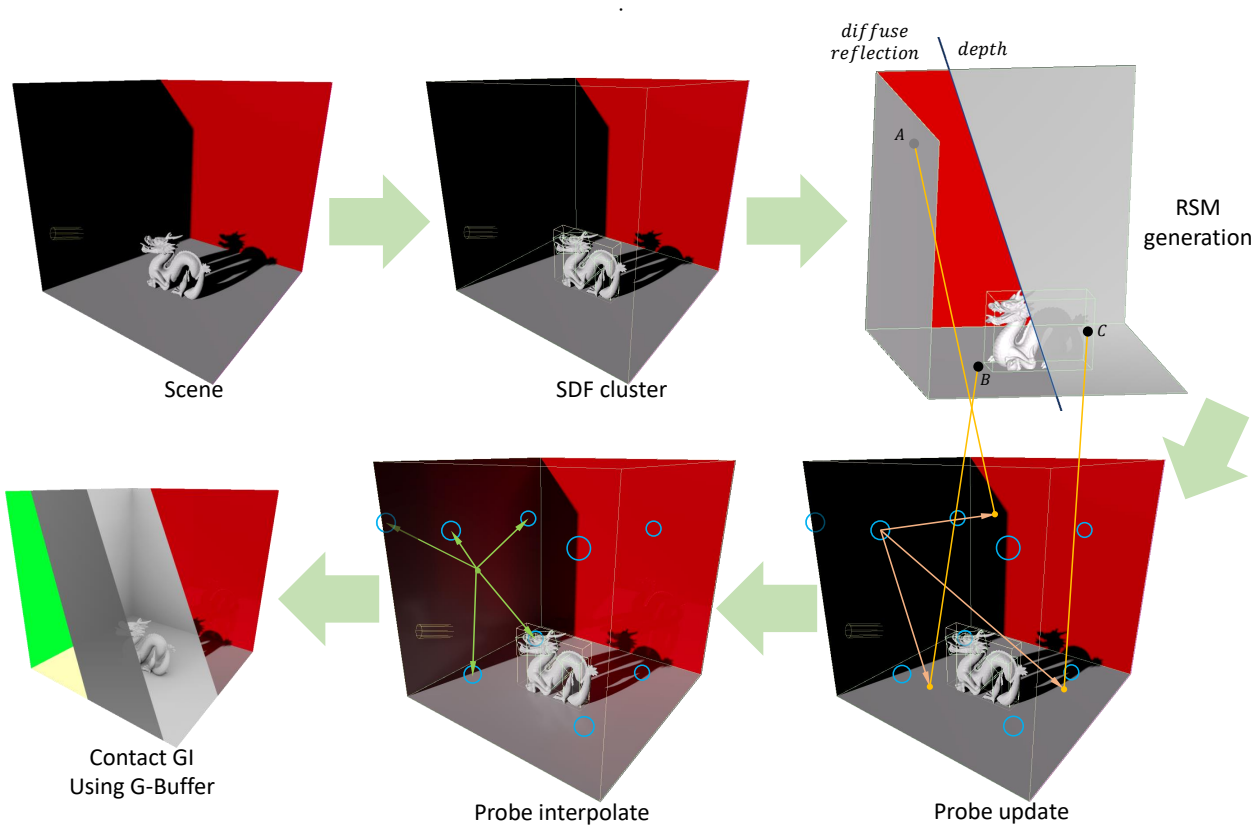


Fig. 3: *To calculate the dynamic diffuse GI in SDFDDGI, we first gather and pack the SDF primitives representing the scene into clusters. Then, the generated RSM together with the SDF primitives are used to update the radiance on the probes, whose position may be adjusted according to scene geometry. Finally, we computer the shading as a weighted sum (according to ambient occlusion) of the result of interpolating probes and screen-space Contact GI.*

our approach, in an LDDE path, L-D is implemented with RSM, D-D with the traversal of SDF, and D-E with the G buffers. Finally, for each pixel, we interpolate probes to obtain the pixel's irradiance, with probe weights determined by SDF soft shadow tests to eliminate light leakage. To compensate for the loss of detail produced by the simplified SDF geometry and the discretization of the irradiance function, we use G buffer data to add Contact GI to enhance the details, which is based on the horizon-based Ambient Occlusion.

3.1 SDF Clusters

SDF is usually stored in volume textures. However, such an approach necessitates a trade-off between the resolution and details of the scene. In addition, it is difficult to use volume textures to store dynamic scenes. Therefore, we represent our scene with a SDF composed of different simple SDF primitives, such as rectangular blocks, spheres, planes, cylinders, or other primitives whose SDF values can be computed analytically and efficiently instead of performing voxelization of the scene. Consequently, our simplified SDF representation can be stored at a small storage cost. Taking the Sponza Palace scene as an example (see Figure 1), only 4 KB is needed to store its SDF representation.

In our implementation, we create the SDF representations of the test scenes manually for simplicity. SDFDDGI is not very sensitive to the accuracy of the SDF representation (see Sect. 4.5) unless the maximum 2-D projection area of an object is very high. For most cases, we can also directly use the bounding volume (used for collision detection) associated with each object as its SDF primitive. For small objects, whose projection area is small, we do not even create any corresponding SDF primitives. This is because the main contribution to indirect illumination of them is provided by Contact GI, and the cost of SDF primitives could be saved, what will be detailed in Sect. 3.5. As can be seen in the Sponza scene, we can use very simple SDF primitives (see Figure 4) while still achieving good results.

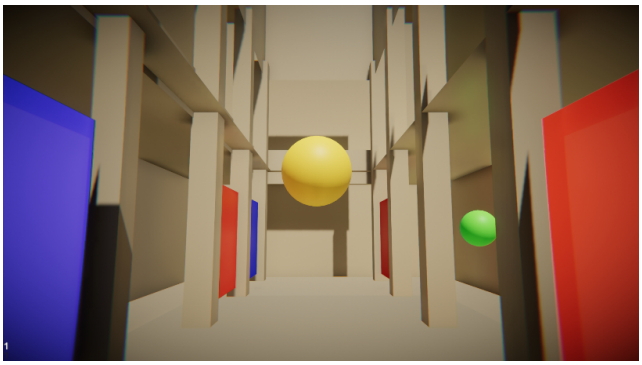


Fig. 4: *Example of the SDF primitives that we used for our experiments for the Sponza scene. Curtains are represented as planes, most of the building structure as cubes, and experimental objects as spheres. Shading in this image is only for visual cohesion.*

The performance of our method remains relatively stable as long as the SDF primitives can be fitted into the GPU L1 cache memory. Most of our experiments were carried on an RTX 2080Ti GPU (see performance on lower-end hardware in Sect. 4.4), in which the increase in cost for larger scenes is negligible even when processing 200 SDF primitives, which is sufficient for a large scene after camera-dependent culling (the total number of primitives that we used for the Sponza scene is 43). Apart from camera-dependent culling, SDF primitives LOD for far objects and other culling are beyound the scope of this work.

In each frame, we cull SDF primitives near the camera and then generate a cluster structure to speed up the SDF query. We use a standard clustering algorithm on the CPU to pack near SDF primitives into a cluster. K-means is performed by using the primitive center. For general hardware, these distance-based clustering algorithms are sufficiently fast and will not create any performance bottleneck. Such an approach achieves performance that is superior to BVH [24] for small-scale data and its cost is negligible in comparison to other stages of our approach. Using this algorithm, we can achieve a 10% to 100% speed-up in different scenes. See Sect. 4.5 for more information.

Simultaneously, since necessary data are mainly in the L1 cache, global memory is available for other parts that have higher requirements on memory access. As a consequence, shading stages that have high requirements on memory but are computationally inexpensive (e.g. volumetric effects or shadow map generation), can be run in parallel with SDFDDGI, thus increasing the GPU general utilization rate.

3.2 Probe choice

Since irradiance is a $\mathbb{R}^5$ function, we split its domain into two parts: space coordinates $\mathbb{R}^3$ and the direction $\mathbb{R}^2$ as shown in Equation 1:

$$P = \begin{bmatrix} x & y & z \end{bmatrix}^\top, D = \begin{bmatrix} \omega & \theta \end{bmatrix}^\top, \text{Irradiance} = E(P, D). \quad (1)$$

Our method discretizes the spatial domain of the irradiance function E into many sampling points P, which are referred to as probes. Every probe stores the irradiance on a sphere of directions around its position. We create a spatially arranged probe volume around the camera and use these probes to interpolate global illumination.

Irradiance is not a continuous function in its spatial domain. For example, there are potential discontinuities in walls or at occluded points, which are the main cause of light leakage in most real-time global illumination algorithms. To represent the irradiance distribution in space reasonably, it is necessary to place these probes carefully.

In a traditional production-ready GI baking system, users will place probes manually or use some time-consuming optimization methods to optimize the positions of probes [6]. However, such a placement approach is too slow to be used in real-time applications. It is worth noting that two probes far away from an object will have a similar sampling result. In addition, a probe will lead to dark leakage when it is very close to an object or inside of the object. Therefore, probes should not be positioned too far away from an object, while they should maintain a smallest distance from the object in order to prevent them from falling into the object. Fortunately, by using our compact SDF representation, such a positioning process can be performed at a very fast speed. Moreover, no baking is required in our framework.

To this end, we first calculate the SDF value at the original probe location. If the value is smaller than a threshold value, which means that the probe is too near to other objects or even inside of an object, this probe position will negatively impact the quality of the sampling. In this case, we query the gradient of SDF and use the gradient descent method to obtain an acceptable sampling point near the original position. If the displacement between the positions of last and current frame exceeds a threshold, the irradiance at last frame will need to be rejected, so we allocate more rays to this probe to ensure a more stable result. The pseudocode of the algorithm is listed in Figure 5.

In order to reduce the number of probe updates, in each frame we first choose which probes need to be updated and divide the updates between different frames for better performance. Probes are assigned different

```
1: procedure UpdateProbePos
2:     lastPos ← position of probe in last frame
3:     pos ← position of probe in the uniform grid
4:     if querySdf(pos) < threshold1 then
5:         pos ← gradientDescent(SDF, pos)
6:     if distance(pos, lastPos) > threshold2 then
7:         markRejectHistory(probe)
8:     return pos
```

Fig. 5: *Pseudocode for the algorithm to find a suitable position for a probe.*

weights according to their distances and directions to the camera in order to determine which probes should be updated.

Due to the random probe choice, this method may produce jitter between frames. This phenomenon, however, is not as evident as in the original RTXGI since the performance of our approach enables us to perform a more extensive sampling. Furthermore, to reduce this artifact, we can perform an interpolation from the surrounding probes to make the result stable.

In contrast to RTXGI, our method does not need human intervention to have a better probe distribution, and it can rapidly and accurately respond to scene changes. This also reduces the leaking caused by dynamic objects.

3.3 Probe update

We use compute shaders to sample the radiance L over a sphere of directions at each probe to calculate the irradiance E:

$$E(P, D) = \int_{4\pi} \max(0, \cos(D, D')) \cdot L(P, D') \mathrm{d}D'. \quad (2)$$

In this phase, we use an 8x8 thread block to update each chosen probe. Prior to sampling, all threads in a block need to cooperate to move the SDF primitives (and cluster structures) into the L1 cache, since they will be queried frequently.

When performing the SDF query, we first calculate the distance from a primitive to clusters and reject all the primitives in a cluster if the distance to the cluster is greater than the current distance. When performing the SDF sphere tracing, many clusters can also be rejected using twice the previous SDF query result as the initial distance. This is because the Lipschitz constant of SDF is always 1 and, therefore, the SDF value of the current point will be always less than double the value at the last point. This can be explained mathematically as follows. As we have

$$\begin{aligned} &|sdf(A) - sdf(B)| \leq 1 \times ||A - B||, \\ &|sdf(A)| - |sdf(B)| \leq |sdf(A) - sdf(B)|, \end{aligned} \quad (3)$$

and therefore we have

$$|sdf(A)| \leq ||A - B|| + |sdf(B)| \leq 2|sdf(B)|. \quad (4)$$

This especially increases performance when the SDF value is smaller. Therefore, this works fairly well for sphere tracing, in which query density is very high near object boundaries, i.e. where the SDF value is small. Moreover, this algorithm is both stack-free and GPU-friendly.

We sample the irradiances at random low-discrepancy directions and store them in the L1 cache. To calculate the intersections with the scene, we perform an accelerated version of sphere tracing [2] for our scenarios.

After the intersection point is calculated, RSM [5] is employed to obtain its flux. Additionally, the emission value stored inside of the primitive is used in order to support area lights and self-emission.

By reusing the probe GI data of the last frame, we can achieve multi-bounce global illumination. Since multi-bounce diffuse GI is generally of lower frequency than first bounce diffuse GI, we limit its sample number to a lower level in order to decrease the computational expense. In addition, we can speed up GI response by multiplying multi-bounce GI by a coefficient between 0 and 1, in order to decrease loop gain effects.

After synchronizing all threads in a block, each thread calculates the irradiance in its own direction D using Equation 2, and uses octahedral mapping [3] to write irradiance data into a probe atlas texture. We maintain a balance between the number of samples and the degree of temporal mixing. If there are too few samples, we apply a stronger temporal mixing in order to prevent jittering. By storing the radiance samples to the L1 cache, we can reuse them across threads to compute the irradiance, and thus stabilize the result.

3.4 Per-pixel GI shading

Probe visibility tests play an important role to prevent light leakage triggered by the discretization of irradiance in the spatial domain.

RTXGI uses probe depth buffers generated by ray tracing and VSM [7] to perform visibility tests. Such an approach is limited to the resolution of the depth buffer, and thus it is unable to eliminate light leakage completely, especially for leakage caused by thin objects. Inspired by the easy generation of soft shadows

using SDF [1], we employ the SDF shadow trace to run visibility tests, which can naturally produce soft indirect shadows and transitions.

Each pixel needs to interpolate 8 probes. This means performing sphere tracing 8 times, making it unacceptable as it is too expensive to compute. Fortunately, we only need to perform a probe visibility test instead of sampling radiance, enabling us to markedly reduce computation cost.

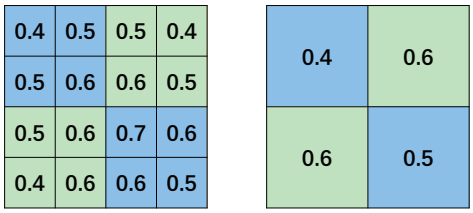


Fig. 6: *Example of min/max checkerboard downsampling algorithm. Green areas take the highest value and blue areas take the minimum value.*

First, we downsample the screen depth buffer according to a min/max checkerboard to obtain a half-resolution buffer. As shown in Figure 6, for checkerboard green pixels, we obtain the maximum value in an area of 4 pixels in the full resolution depth buffer; otherwise, we take the minimum for blue pixels. This assures that we have valid samples that cover the whole depth range. This algorithm was developed in 2019 by the Red Dead Redemption team and it has proven to be highly effective for downsampling in the domain of low-frequency rendering.

Subsequently, in the min/max checkerboard downsampled depth buffer, we choose one pixel out of every 2×2 pixel block to formulate a $1/4$ resolution buffer as follows. (1). For different frames, we choose a different pixel; (2). Cover the whole depth extent as much as possible.

This can be accomplished simply by first choosing a pixel according to the frame index, and then changing some of them by checking the neighbouring chosen pixels.

We then perform a duplicate removal on each 2×2 block unit in the $1/4$ resolution buffer. We assume that if the starting points are close, they will have the same visibility test result for a specific probe. In this way, we can avoid some duplicate computations.

After duplicate removal, visibility tasks are distributed uniformly for each 4 threads assigned for every 2×2 block, making every thread's visibility tasks diminish from 8 to 4 on average. We use shared memory for the pixels to exchange information. In our experiments, we use a 4×4 size block to obtain an optimal speedup ratio, because an oversized block hinders the merging of visibility test tasks, thus decreasing the effectiveness of this algorithm.

After completing the visibility tests, we write the result back into the L1 cache and distribute it to every corresponding pixel in order to be used for probe interpolation. Since our approach can accurately obtain the visibility of the probe, we do not need to perform an extra cosine weighting or any other additional weighting terms like RTXGI to further avoid leakage artifacts.

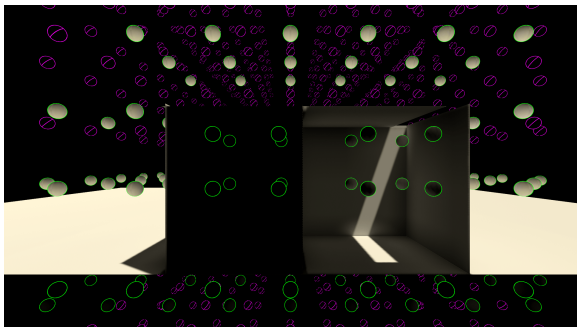


Fig. 7: *Dark room without light leakage even with thin walls. Our approach is independent of probe resolution, and thus will not produce light leakage for objects of any thickness.*

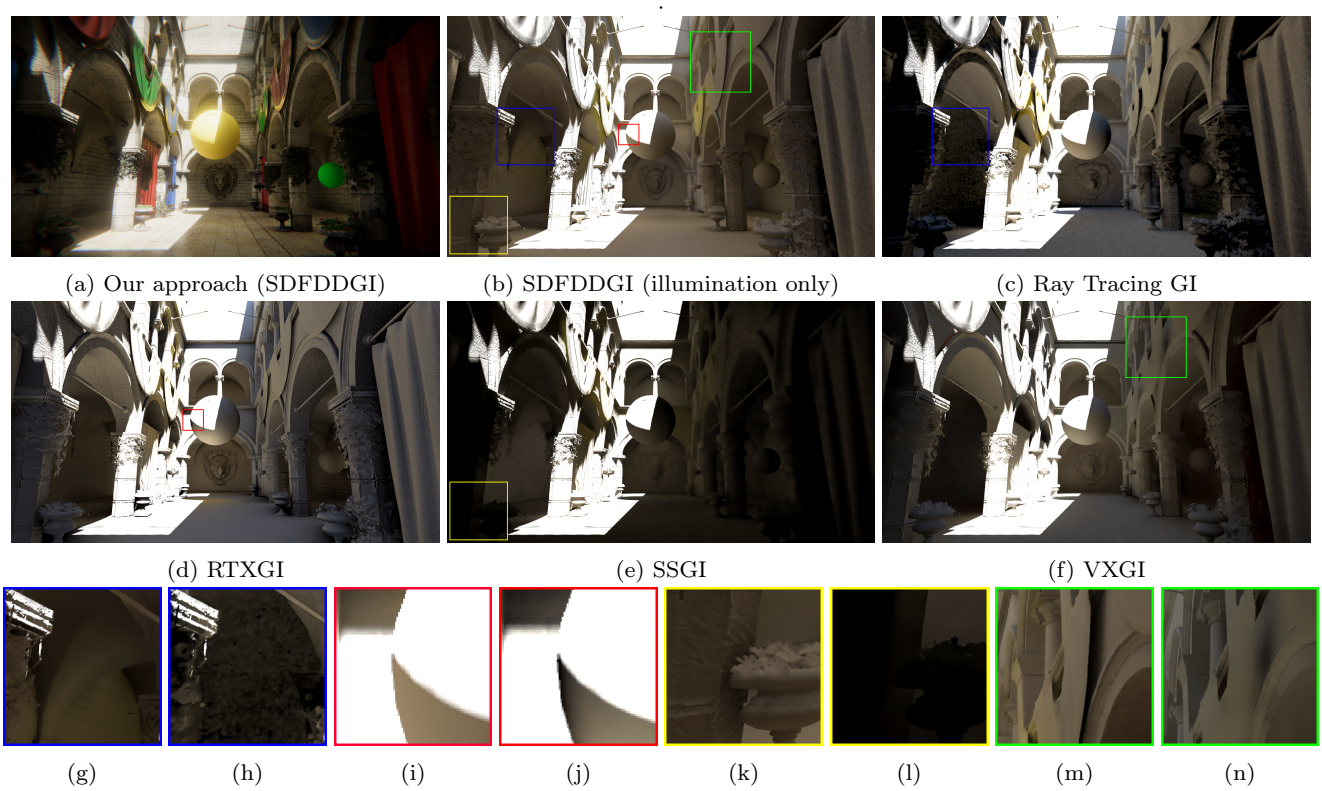
In the upsampling stage, if a pixel does not have a valid result, we assign it a value according to its surrounding pixels or the last frame's reprojection result using the motion vector. Although finding representative pixels in the second phase is more important for the quality of the final result, we can reuse past results and Neighborhood Clipping to reduce its influence until it is negligible.

This method greatly compensates for the overwhelming strain produced by the per-pixel visibility tests. It reduces 8 tests per pixel to 0.25 tests per pixel or lower, which greatly increases the performance of our algorithm without a negative impact on the result.

RTXGI uses a low-resolution depth buffer to perform visibility tests, which leads to light leakage artifacts. We employ a per-pixel probe visibility test instead to eradicate light leakage completely and optimize the process with effective downsampling to further reduce the computational cost, as shown in Figure 7.

3.5 Contact GI

Although our method can greatly reduce the computational cost of GI by discretizing the irradiance on the



(a) Our approach (SDFDDGI) (b) SDFDDGI (illumination only) (c) Ray Tracing GI

(d) RTXGI (e) SSGI (f) VXGI

(g) (h) (i) (j) (k) (l) (m) (n)

Fig. 8: *The full result of the Sponza Palace scene rendered by our real-time SDFDDGI method (a). We compare the illumination only result of our approach (b) to that of other real-time GI methods (c)(d)(e)(f). Our method is able to overcome limitations in prior methods such as extra noises and the lacking of multi-bounce lighting in real-time Ray Tracing GI (h), light leakage on the dynamic yellow sphere in RTXGI (j), darker regions and incomplete GI due to the use of only screen space information in SSGI (l), and rougher detailing in Voxel-Based GI (n). Our zoomed-in counterpart results are shown in (g), (i), (k) and (m), respectively.*

space domain, it may result in loss of detail. Therefore, we present a screen space refinement approach that can enhance the rendering details, dubbed Contact GI.

Ambient occlusion represents how much ambient lighting an object surface receives according to the occlusion received by surrounding objects. In the per pixel GI shading stage, we can enhance its details by multiplying the GI shading result with AO. This works reasonably well in dark regions. However, such an approach does not take into account the effect of multi-bounce indirect illumination among other problems. For instance, the effect of diffuse GI should be greater in directly lit regions. However, we would incorrectly weaken the indirect illumination by simply multiplying the result by the AO mask, as illustrated in Figure 9.

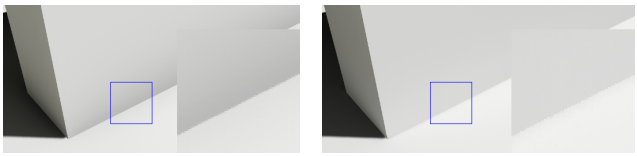
Inspired by horizon-based ambient occlusion (GTAO) [8], we propose Contact GI to enhance the details of diffuse GI without producing overshadowing by taking into account incoming irradiance E_i (Equation 5):

$$\begin{aligned} E_i &= \int_{\omega_i} L_w \times (\vec{n_c} \cdot \omega) d\omega \approx C_i \times \int_{\omega_i} (\vec{n_c} \cdot \omega) d\omega \\ &\approx C_i \times \sqrt{1 - (\vec{n_c} \cdot \vec{n_i})^2} \times \int_{\omega_i} d\omega \\ &\approx C_i \times \frac{\sqrt{1 - (\vec{n_c} \cdot \vec{n_i})^2}}{distance(P_c, P_i)^2} A_i, \end{aligned} \tag{5}$$

where E_i is the incoming irradiance contributed by pixel i to pixel P_c. C_i is the color of pixel i, $\vec{n_i}$ is its normal vector, and A_i is its corresponding actual area.

We assume that all objects have a completely rough surface, which means that its previously shaded screen color can be regarded as its output radiance. When searching for horizon angles in GTAO, we add the current sampling pixel's contribution to irradiance if it is not occluded (see Figure 10). With such a method, we can obtain an irradiance value when computing ambient occlusion.

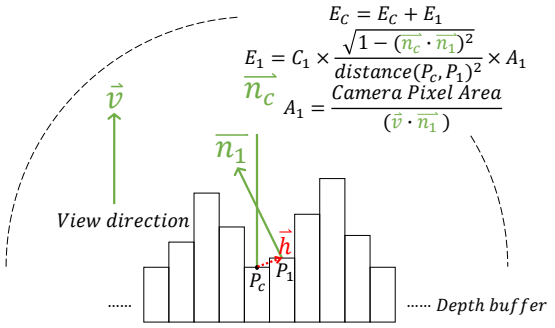
Subsequently, as in GTAO, we apply spatial and temporal denoising to the obtained Contact GI (AO



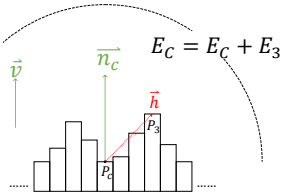
(a) A naïve AO mask, which incorrectly overshadows some regions.

(b) Contact GI takes into account the effect of indirect illumination.

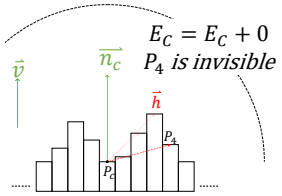
Fig. 9: *The corner formed by the wall and the floor is overshadowed, since it does not take into account the effect of multi-bounced indirect illumination.*



(a) Current pixel P_1 is visible by pixel P_c, and thus we calculate its contribution to incoming irradiance.



(b) Current pixel P_3 is also visible by pixel P_c, and thus we add its contribution to incoming irradiance.



(c) Currently searched pixel P_4 is **not** visible by pixel P_c, and thus it does not contribute to incoming irradiance.

Fig. 10: *We take advantage of the sampling stage when searching horizon angles to calculate incoming irradiance's contribution for contact GI. The bars represent the depth buffer, $\vec{n_i}$ is the normal of pixel i, and $\vec{h}$ is current horizon vector. Visibility can be checked simply by assuring that the angle between vectors $\vec{P_iP_c}$ and $\vec{n_c}$ is smaller than that of the current horizon vector.*

and the screen space irradiance are packed in one vector4). Strong ambient occlusion means that the average distance of light bounce is small, thus screen space information is more complete for better quality GI. Therefore, the AO value is used as a weight to interpolate screen space irradiance and probe irradiance linearly to compute the final result. When the AO value is low, i.e., those areas which receive more GI contributions from near-by surfaces, Contact GI will have a higher weight. Such a design allows us to take both large-scale GI and rich detailed GI into account.

4 Results

We tested the performance and GI quality of our method with different hardware and scenes. Most of the experiments were performed in comparison to other state-of-the-art real-time global illumination approaches.

4.1 Probe volume resolution

Since the positions of the probes are assigned at run time according to the SDF gradient, this method does not require any previous human intervention for a reasonable distribution of the probes. We first observe the effect of the resolution of the probe volume on the resulting GI, and then we compare it to an offline path tracer as the ground truth. In Figure 11, it can be seen that when the resolution of probe volume is very low, our method can still prevent the light leakage problem, although the resulting GI will not be very accurate.

4.2 Effect of Contact GI

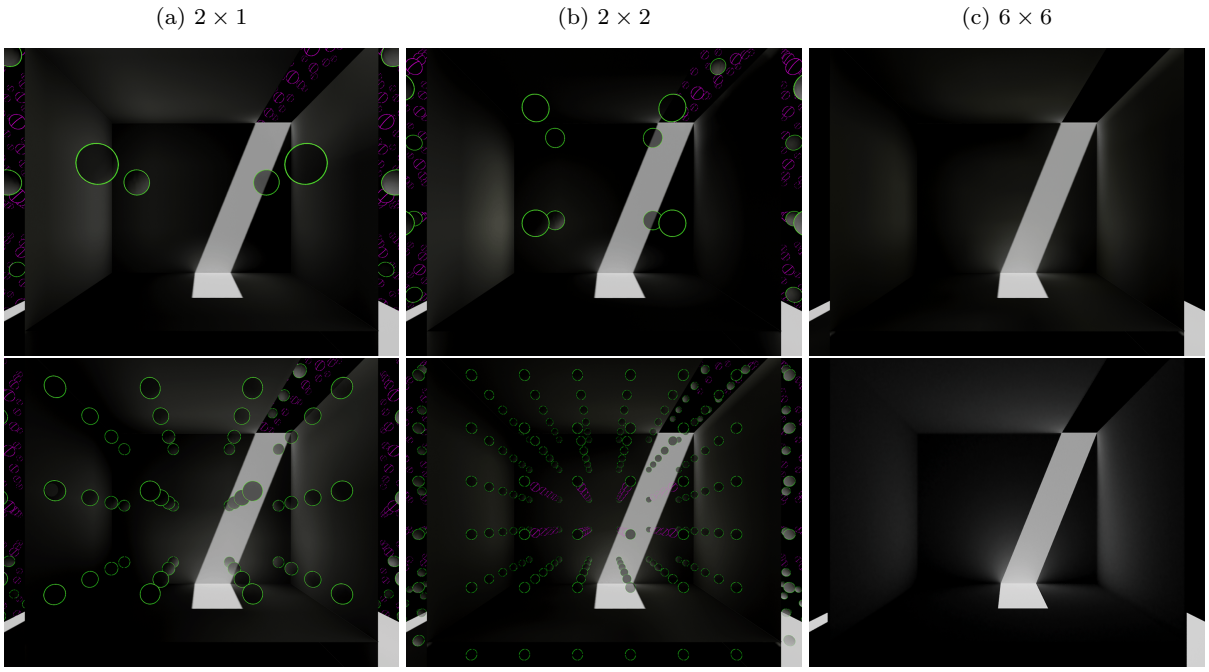
Our Contact GI compensates for the loss of detail of diffuse GI produced by the sampling of the irradiance function. In Figure 12, we can observe the difference of using Contact GI on a basic lighted room and the Sponza scene.

Contact GI brings about significant improvements to the effect of minor traits of the scene to the diffuse global illumination. It solves some of the loss of detail resultant from the disparity of the position of the probes and the real position taken into account for shading.

4.3 GI comparison to other methods

We first compare our method to the offline path tracer, as shown in Figure 1. We then compare our method to other real-time GI techniques using the Sponza scene. This scene has 250,000 triangles, 26 different materials, and 48 different resolution textures. We employ 43 SDF primitives for the dynamic representation of the structure of the scene, and cluster them into 8 clusters. The probe volume of SDFDDGI, as well as RTXGI, has a resolution of $22 \times 14 \times 32$ probes. The VXGI voxel resolution is $64 \times 64 \times 64$.

It can be seen from Figure 8 that with the same probe volume, our approach fits the subtleties of the



(a) 2×1 (b) 2×2 (c) 6×6

(d) 4×4 (e) 6×6 (f) Ground Truth

Fig. 11: *SDFDDGI results with different probe volume resolutions. We take path tracing as the ground truth for reference. With too few probes, as shown in the room with* 2×1 *probes (a), we may not be able to produce satisfactory global illumination but light leakage is still absent. In addition, with a denser but still relatively rough probe volume grid (c), we can obtain similar global illumination as in a path tracing reference (f).*

scene better. In comparison to Ray Tracing GI, our method is able to support multiple-bounce illumination and does not require a denoiser, which could blur some details. VXGI with a higher resolution is not able to achieve the same level of detail while costing more computation time.

As shown in Figure 13, our approach generates no light leakage even for a low-resolution probe volume. Light leakage will arise in RTXGI, however, especially for thin objects such as the curtains. Instead, our method correctly manages the occlusion of curtains.

4.4 Performance

We test the performance of our method with the Sponza scene as the test scene under the same configuration as in the previous section, and analyze the time consumed for every stage of the algorithm. We also perform comparisons to other aforementioned methods with different graphics cards.

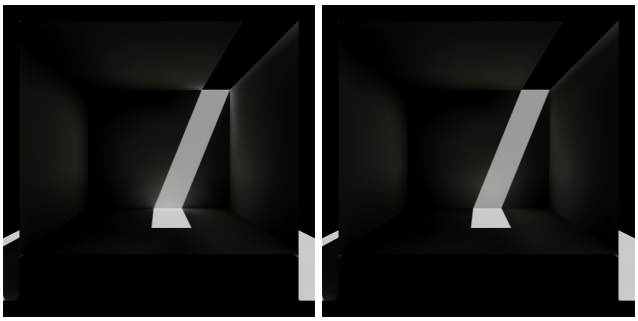
As shown in Table 1, our method achieves the best performance while retaining the same or better quality. Time per frame is only shorter for Screen Space GI, which makes substantial quality compromises in order to achieve this performance.

	Consumed time	
Method	Total	Per stage
SDFDDGI	1.30 ms	*Probe Update*: 0.18 ms *Contact GI*: 0.57 ms *Shading*: 0.54 ms
SSGI (diffuse only)	1.17 ms	*Depth Mipmap*: 0.06 ms *HiZ trace*: 0.69 ms *Denoise*: 0.42 ms
RTXGI	3.98 ms	*Probe Update*: 3.28 ms *Shading*: 0.70 ms
Ray Tracing GI	4.13 ms	*Trace ray*: 2.23 ms *Denoising*: 1.90 ms
VXGI (diffuse only)	5.24 ms	*Voxelize*: 3.31 ms *Cone trace*: 1.93 ms

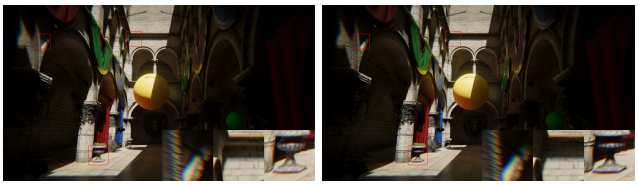
Table 1: *Stage by stage performance comparison of state of the art real-time GI approaches to our method. All comparisons are run on RTX 2080Ti and i7 9700k, with a render resolution of* 1920×1080 *on the Sponza scene.*

4.5 Effect of SDF representation on GI

Since diffuse global illumination is based on the underlying SDF representation of the scene, different SDF for the same scene will necessarily have an impact on GI. Figure 14 shows the effect on GI of different SDF for

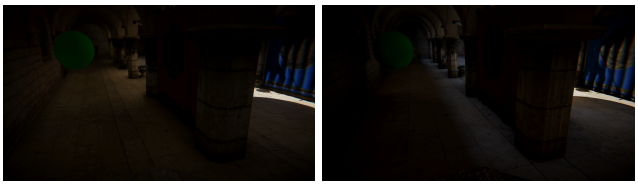


(a) Contact GI on (b) Contact GI off



(c) Contact GI on (d) Contact GI off

Fig. 12: *Examples of the effect of Contact GI on a simple lighted room (a) (b) and the Sponza Palace (c) (d). Contact GI greatly compensates for the loss of detail resultant from discretization of the irradiance and enhances the realism of details.*



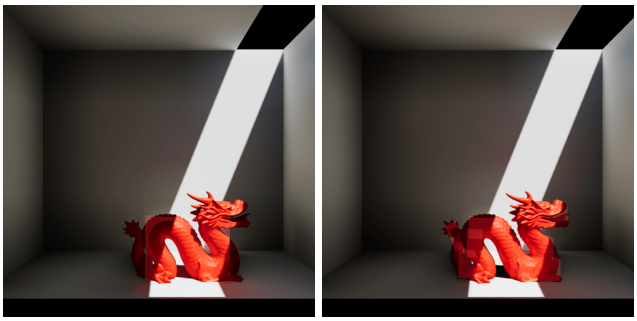
(a) SDFDDGI (b) RTXGI

Fig. 13: *Comparison of light leakage artifacts on SDFDDGI and RTXGI. Our method (a) is able to eradicate light leakage. However, on thin objects, such as the curtain in the Sponza scene, RTXGI may encounter light leakage issues (b).*

the same underlying geometry. Even for such a coarse SDF, we can see that its visual impact is much less recognizable due to the low-frequency nature of the diffuse GI. Low detail is partially compensated with Contact GI as well. Table 3 gives a performance comparison of different amount of primitives representing the Sponza Scene.

5 Conclusions

We have proposed a novel approach to calculate real-time diffuse global illumination using SDF. Our method supports fully dynamic lighting and geometry for a scene while requiring no baking. It also provides superior quality global illumination and achieves higher performance than other prior real-time approaches, and thus has manifold potential applications with dynamic scenes.



(a) Coarse SDF (b) Fine SDF

Fig. 14: *Comparison of the GI produced by a coarser and a finer SDF representation of the Stanford dragon model, computed with a probe resolution of* $6 \times 5 \times 3$*. SDF in both cases are formed by the union of a set of boxes, represented as squares in the picture.*

Our method still has room for improvement. First, instead of using a fixed density of probe volume, we could sample the irradiance space with more probes for regions near the camera and less probes for regions far from the camera. This would make the algorithm feasible for scenes with different scales. It would also offer a better balance between small indoor scenes, mid-ranged building scenes, and large open worlds as well. We could also use importance sampling for probe ray generation according to the relative placement of the camera and the probes in order to further stabilize global illumination since only polygons whose normals point to the camera are visible.

	Consumed time by graphics card	
Graphics card	Total	Per stage
GeForce RTX 2080 Ti	1.30 ms	*Probe Update*: 0.18 ms *Contact GI*: 0.57 ms *Shading*: 0.54 ms
GeForce GTX 1070	3.52 ms	*Probe Update*: 0.48 ms *Contact GI*: 1.24 ms *Shading*: 1.80 ms
GeForce GTX 970M	9.05 ms	*Probe Update*: 1.45 ms *Contact GI*: 3.46 ms *Shading*: 4.14 ms

Table 2: *Stage by stage performance comparison of our method running on different graphics cards. The shading stage costs more time in low-end graphics cards because we only adjusted optimization parameters for high-end graphics cards.*

Amount	Consumed time	
	Total	Per stage
43 prims	1.30 ms	*Probe Update*: 0.18 ms *Contact GI*: 0.57 ms *Shading*: 0.54 ms
92 prims	1.61 ms	*Probe Update*: 0.30 ms *Contact GI*: 0.57 ms *Shading*: 0.74 ms
130 prims	1.92 ms	*Probe Update*: 0.43 ms *Contact GI*: 0.57 ms *Shading*: 0.92 ms

Table 3: *Performance comparison for different amount of primitives.*

Second, our research focuses on the dynamic diffuse GI. For specular GI, we still have to rely on a mixed approach using other methods such as SSR [21] or Ray Tracing. The L(D*)SE path can be handled with zero extra cost when using SSR together with SDFDDGI. Finally, our approach uses simplified SDF primitives to represent the scene. Our current implementation relies on a partially manual approach to create a simplified SDF representation, which might be more labor intensive for large and complex scenes. Determination of how to approximate a model with simplified SDF primitives automatically [25] constitutes a worthwhile direction for future research.

Acknowledgements We thank the anonymous reviewers for their constructive comments. Xiaogang Jin was supported by the National Key R&D Program of China (Grant No. 2017YFB1002600), the National Natural Science Foundation of China (Grant Nos. 62036010, 61732015), the Ningbo Major Special Projects of the "Science and Technology Innovation 2025" (Grant No. 2020Z007), and the Key Research and Development Program of Zhejiang Province (Grant No. 2020C03096).

References

1. Aaltonen, S.: Gpu-based clay simulation and ray-tracing tech in claybook. In: Game Developers Conference (2018)
2. Bálint, C., Valasek, G.: Accelerating sphere tracing. In: Eurographics (Short Papers), pp. 29–32 (2018)
3. Cigolle, Z.H., Donow, S., Evangelakos, D., Mara, M., McGuire, M., Meyer, Q.: Survey of efficient representations for independent unit vectors. Journal of Computer Graphics Techniques **3**(2), 1–30 (2014)
4. Crassin, C., Neyret, F., Sainz, M., Green, S., Eisemann, E.: Interactive indirect illumination using voxel cone tracing. Comput. Graph. Forum **30**(7), 1921–1930 (2011)
5. Dachsbacher, C., Stamminger, M.: Reflective shadow maps. In: I3D, pp. 203–231 (2005)
6. Di Benedetto, M., Ganovelli, F., Balsa Rodriguez, M., Jaspe Villanueva, A., Scopigno, R., Gobbetti, E.: Exploremaps: Efficient construction and ubiquitous exploration of panoramic view graphs of complex 3d environments. Comput. Graph. Forum **33**(2), 459–468 (2014)
7. Donnelly, W., Lauritzen, A.: Variance shadow maps. In: I3D, pp. 161–165 (2006)
8. Jiménez, J., Wu, X., Pesce, A., Jarabo, A.: Practical real-time strategies for accurate indirect occlusion. In: SIGGRAPH 2016 Courses (2016)
9. Kajiya, J.T.: The rendering equation. In: ACM SIGGRAPH, pp. 143–150 (1986)
10. Kaplanyan, A., Dachsbacher, C.: Cascaded light propagation volumes for real-time indirect illumination. In: I3D, pp. 99–107 (2010)
11. Koskela, M., Immonen, K., Mäkitalo, M.J., Foi, A., Viitanen, T., Jääskeläinen, P., Kultala, H., Takala, J.: Blockwise multi-order feature regression for real-time path-tracing reconstruction. ACM Trans. Graph. **38**(5), 138 (2019)
12. Laine, S., Saransaari, H., Kontkanen, J., Lehtinen, J., Aila, T.: Incremental instant radiosity for real-time indirect illumination. In: EGSR, pp. 277–286 (2007)
13. Majercik, Z., Guertin, J.P., Nowrouzezahrai, D., McGuire, M.: Dynamic diffuse global illumination with ray-traced irradiance fields. Journal of Computer Graphics Techniques **8**(2), 1–30 (2019)
14. McAuley, S.: Rendering the world of far cry 4. In: Game Developers Conference, pp. 143–146 (2015)
15. Ritschel, T., Grosch, T., Seidel, H.: Approximating dynamic global illumination in image space. In: I3D, pp. 75–82 (2009)
16. Schied, C., Kaplanyan, A., Wyman, C., Patney, A., Chaitanya, C.R.A., Burgess, J., Liu, S., Dachsbacher, C., Lefohn, A.E., Salvi, M.: Spatiotemporal variance-guided filtering: real-time reconstruction for path-traced global illumination. In: Proceedings of High Performance Graphics, pp. 2:1–2:12 (2017)
17. Seyb, D., Sloan, P., Silvennoinen, A., Iwanicki, M., Jarosz, W.: The design and evolution of the uberbake light baking system. ACM Trans. Graph. **39**(4), 150 (2020)
18. Shyshkovtsov, O., Karmalsky, S., Archard, B., Zhdan, D.: Exploring raytraced future in metro exodus. In: Game Developer's Conference (2019)
19. Silvennoinen, A., Timonen, V.: Multi-Scale Global Illumination in Quantum Break. In: ACM SIGGRAPH Advances in Real-Time Rendering Course. ACM (2015)
20. Sloan, P.J., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Trans. Graph. **21**(3), 527–536 (2002)
21. Sousa, T., Kasyan, N., Schulz, N.: Secrets of cryengine 3 graphics technology. In: SIGGRAPH Courses (2011)
22. Thomas, M.M., Forbes, A.G.: Deep illumination: Approximating dynamic global illumination with generative adversarial network. CoRR **abs/1710.09834** (2017)
23. Veach, E.: Robust Monte Carlo methods for light transport simulation. Stanford University PhD thesis (1997)
24. Wald, I., Boulos, S., Shirley, P.: Ray tracing deformable scenes using dynamic bounding volume hierarchies. ACM Trans. Graph. **26**(1), 6 (2007)
25. Wu, J., Kobbelt, L.: Structure recovery via hybrid variational surface approximation. Comput. Graph. Forum **24**(3), 277–284 (2005)