Real-Time Global Illumination in Point Clouds

Reinhold Preiner*
Supervised by: Michael Wimmer†

Institute of Computer Graphics And Algorithms
Vienna University of Technology
Vienna / Austria

Abstract

In this paper we present a real-time global illumination approach for illuminating scenes containing huge point clouds. Our GI approach is based upon the distribution of Virtual Point Lights (VPLs) in the scene, which are then used for the indirect illumination of the scene geometry, using Imperfect Shadow Maps for visibility calculation of the VPLs. We are able to render multiple indirect light bounces at real-time rates, where each light bounce handles the transport of both the diffuse and the specular fraction of the reflected light.

Keywords: Global Illumination, Point Clouds, ISM, Imperfect Shadow Maps, VPL, Virtual Point Light

1 Introduction

Point clouds are a convenient type of geometry representation when huge amounts of geometrical data are to be displayed quickly, or when geometrical data is given in this way (e.g. gathered from a 3D scanning device) and has to be displayed immediately and without a time-consuming triangulation preprocessing step. The Scanopy application[1] developed at the Vienna University Of Technology and at Imagination[2] in Vienna is able to show huge point clouds in real-time and was originally developed to display scanned objects.

There are various approaches for global illumination (GI) in scenes containing conventional mesh geometry. Virtual Point Lights (VPLs) were introduced by Keller in 1997 [4] for indirect illumination, and Ritschel et al [7] proposed Imperfect Shadow Maps (ISMs) as an efficient way for visibility calculation even for a high number of VPLs. The methods and algorithms in our approach base on the thesis of Knecht [6], who implemented GI using VPLs for indirect illumination and ISMs for visibility.[3]

In order to render the ISMs, a number of sample points covering the scene's surfaces, are taken from the meshes, and then are splatted onto the maps. Since we already deal with point clouds describing our geometry, it is an obvious step to take advantage of this fact by applying this GI method on point clouds.

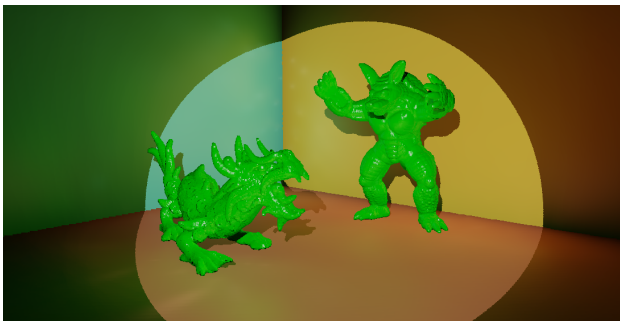

Figure 1: Point cloud scene containing over 5 million points rendered with our global illumination algorithm

Contribution. In this paper, we present our GI-algorithm on point clouds and introduce our approach in illumination computation over several light bounces. Our approach combines the advantages of fast point rendering and efficient ISM rendering for visibility in GI. Further, we advance the illumination method implemented by Knecht by increasing the number of VPLs available for indirect illumination in multiple bounces by simultaneously incorporating one direct- and one indirect bounce with one VPL.

In Section 2 we give a short overview of related work and the background in point rendering, Virtual Point Lights and Imperfect Shadow Maps. In Section 3 and 4, we introduce our approach and describe our algorithm in detail, showing how the implementation of Knecht [6] was advanced in our approach in order to implement real-time GI for point clouds. Finally, in Sections 5 and 6, we present the results of our work and subsequently discuss its restrictions and future work to be done.

2 Related Work

Considering a correct simulation of global illumination in a scene, the rendering equation, first introduced by Kajiya [3] in 1986, represents an ideal and complete description of the illumination process.

*reinholdpreiner@gmx.at
†wimmer@cg.tuwien.ac.at
[1] www.cg.tuwien.ac.at/research/projects/Scanopy
[2] www.imagination.at
[3] Note: Knecht also applies temporal coherency, which we do not.

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

Equation 1 shows a common notation of the rendering equation. The energy radiating from a point p to a direction ω_o equals the sum of energy emitted from p in that direction (represented by emittance-term $L_e(p,\omega_o)$), and the energy from all light from the whole hemisphere above p that is reflected in direction ω_o. The latter is given by the BRDF $\rho(p,\omega_i,\omega_o)$, and the integrand of the incoming light direction $L_i(p,\omega_i)$ incorporates all light incident from the hemisphere Ω over p.[4] θ is the angle between incident light direction and the surface normal at p, and $\cos\theta$ represents a geometry factor that influences the amount of reflected light based on the incident light direction.

Several offline techniques exist, that simulate light propagation as described by the rendering equation, but, since too time-consuming, are inapplicable for real-time rendering. Real-time Global Illumination is a challenging task and with current hardware can often only be achieved by introducing speed-gaining approximations that result in acceptable images.

Keller introduced Virtual Point Lights (VPLs) in his Instant Radiosity approach in 1997 [4]. Those VPLs are seeded over the scene geometry and act as "virtual" light sources for the overall illumination computation of a screen pixel. They represent a sampled subset of all points contributing incident energy on a given point p in the rendering equation. The main issue with the use of many VPLs is the visibility information for all possible outgoing light directions of each single VPL.

In 2008, Ritschel et al [7] proposed so called Imperfect Shadow Maps (ISMs) as a method for efficiently computing indirect illumination. The main idea behind ISMs is, that it is sufficient to create fast and inaccurate (imperfect) shadow maps, if used for a large number of Virtual Point Lights. With growing number of VPLs used for indirect illumination, visible errors or artifacts from the shadow maps' imperfectness get more and more obliterated.

We apply global illumination on point clouds, implemented in a renderer that is able to render enormous point clouds at interactive frame-rates. The technique bases on Wimmer and Scheiblauer's Instant Points approach [8], introduced in 2006. They use a new out-of-core algorithm that uses nested octrees for efficiently building a hierarchy on the point set, significantly reducing the memory-overhead for the data-structure. Figure 2 shows a 3d-scanned point cloud scene rendered in the Scanopy application.

[4]Note: When rendering translucent objects, we have to extend to BSDFs and integrate over the whole sphere around p

Figure 2: Point cloud scene of St. Stephan's Cathedral in Vienna, rendering over 20 million points (over 420 million points in model) with the Instant Points approach in Scanopy (no illumination computation).

3 Overview

Our scene is represented by a point cloud and illuminated by a spot-lightsource. Illumination of the scene geometry mainly consists of two parts: direct and indirect illumination. The direct illumination part is straightforward: The geometry within the spotlight-cone is shaded, and shadow-mapping is performed. Indirect illumination is the more sophisticated part. The term indirect illumination describes all illumination from light rays, which do not come directly from the light source, but rather come from some surface point where the ray was reflected (bounced). Of course, light rays in real world can be bounced several times until their energy is totally absorbed by the reflecting surfaces. Therefore, a good GI implementation also includes multiple light bounces (while maintaining an accordant high frame rate).

To perform indirect illumination, the bulk of light rays from the light source that are reflected at different surface points and from there are illuminating other parts of a scene, is approximated by a number of Virtual Point Lights (VPLs). Those VPLs are seeded over the directly illuminated area in the scene, and from there on each one acts as a point light illuminating the geometry within the whole hemisphere over that point. In an own render pass, those parts of the scene which are visible to the camera are shaded with respect to the previously seeded VPLs. This shading is performed in image space. We use a Camera G-Buffer which stores necessary data of the scene per pixel - i.e. only for those points of the scene which are visible to the camera - and each consecutive shading is performed per pixel on this G-Buffer.

When rendering multiple light bounces, those steps are repeated, i.e. first the VPLs have to be redistributed (originating from the current VPL positions), and then the G-Buffer is again shaded from the new VPL positions, accumulating illumination in the G-Buffer over several bounces.

In order to improve speed, we perform interleaved shad-

ing on the accumulation buffer, as proposed by [5]. Only a interleaved subset of pixels is shaded per VPL, resulting in fewer computations, at costs of reduced quality. After all indirect illumination computations - probably iterative over multiple light bounces - are finished, the interleaved accumulation buffer is merged to an image of original size. This merged image can show local differences in the shading of the pixels (see Figure 7). Therefore, we apply an geometry-aware filter kernel in order to smooth the result image [5]. Finally, direct illumination with shadow-mapping is added, and the resulting HDR scene image is tone-mapped in order to obtain an LDR image of the scene.

Basically, the implementation of indirect illumination in our approach represents an extension to the method described by [6]. While [6] just follows an indirect light ray over several bounces reducing its energy with each reflection, this implementation additionally takes a possible direct illumination of each single reflecting point into account. A comparison of these approaches is given in figure 3.

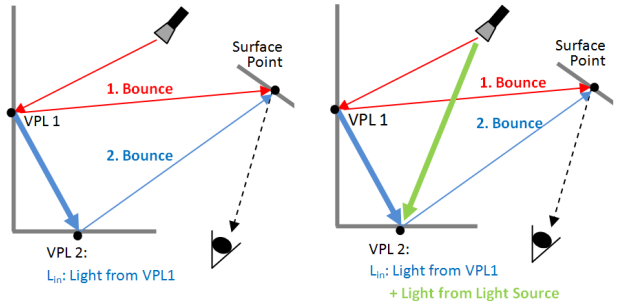

Figure 3: Comparison between Indirect Illumination by Knecht (left side) and our implementation (right side). At multiple bounces, Knecht only reflects the light coming from the last VPL at following VPLs in direction of the surface point to be shaded. The new approach implemented in Scanopy reflects both the light from the last VPL and additionally the possibly incident light coming directly from the light source.

Specular bounces. We are able to correctly render multiple specular bounces. At each bounce, we incorporate both diffuse and specular components of light incident when shading specular reflections[5], considering that the color of specular reflected light deflects over several bounces due to mixed diffuse light (see Figure 4). To accomplish this, for each VPL at each bounce, we cache the incoming light from the previous VPL scaled by the cosine of the light incident angle (geometry term) to lookup the diffuse contribution of any incident light ray.

[5]Note that this is the same as for diffuse reflections

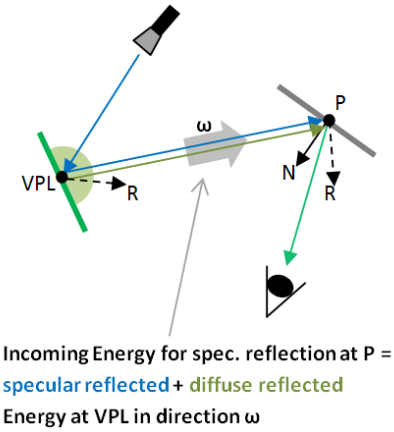

Figure 4: Illustration of our method for computing specular reflections. At each reflection point, N is the surface normal, and R is the direction of max. specular reflection. The incident light reflected at P is the sum of diffuse reflected light (green) and the specular reflected light (blue) at a VPL, resulting in a slightly changed color of the incident light reflected at P.

4 Detailed Algorithm

To perform global illumination in point clouds, our algorithm incorporates several rendering passes, which are illustrated in Figure 5. This sections describes the several steps of the algorithm in more detail. Figure 7 illustrates the intermediate buffers (G-Buffers, ISMs, Accumulation Buffer), and their place in the rendering chain.

4.1 Camera- and Light-G-Buffer

In the first step, the whole scene is rendered from the camera's point of view into a Camera G-Buffer. The G-Buffer stores necessary information of the accordant geometry or surface behind each pixel in image space. This information is distributed over several textures, and consists of the RGB color of the surface, its material properties (diffuse intensity, specular intensity and shininess), the surface normal and the linear depth of the geometry in view-space.

In the second pass, the whole scene is rendered again, but this time from the spot-light-source's point of view. The scene information rendered in this pass is stored into the Light G-Buffer, which contains the same per-pixel information as the Camera G-Buffer, but further stores an importance value. This value correlates with the intensity of the surface color and the surface's shininess (specular power), and is needed for importance sampling of when distributing VPLs in a following step.

4.2 Distribute VPLs

As mentioned before, indirect illumination is calculated by shading scene geometry with respect to Virtual Point

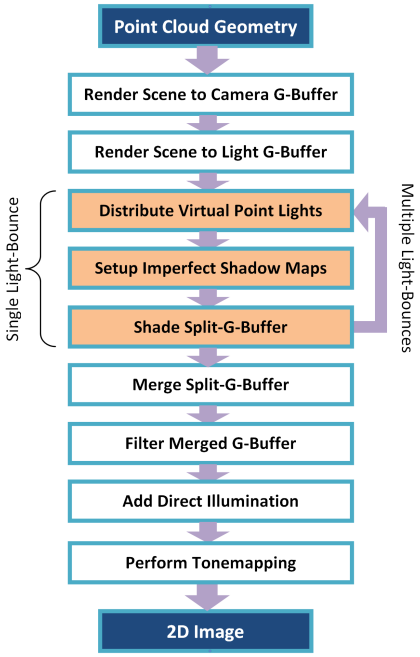

Figure 5: algorithm overview of the global illumination rendering chain

Lights (VPLs), which represent a number of surface points that are directly (or already indirectly) illuminated from some previous light source, and from there illuminate other parts of the scene by reflecting their incident light.

The first set of VPLs is distributed on the surface area illuminated directly by the spot light. The quality of the resulting image increases with the number of VPLs used. However, the number of VPLs significantly influences the frame rate. Therefore, an importance sampling approach is used in order to achieve good results even with fewer VPLs. We start at a pseudo-random distribution of the VPL positions over the spot-light-illuminated area stored in the Light G-Buffer. A 2d-Halton distribution is used in order to achieve a controlled homogeneous distribution, since simple random distributions contain a higher level of noise and locally inhomogeneous areas. Based upon this distribution, hierarchical warping proposed by [2] is used to relocate the VPL positions to obtain denser VPL distributions where needed, and less VPLs where they do not contribute much to the final scene illumination anyway.

When sampling of the VPL positions is done, all necessary data of the VPLs is stored into a VPL-Buffer, which is represented by several 1d-textures. This VPL data consists of the VPL's world space position, the surface normal and material properties (color, reflection indicators) of the surface point the VPL is located on, and two values indicating (or related to) the illuminance of the VPL from both its previous VPL (from a previous bounce) and directly from the light source (see figure 3). The latter values are needed later when calculating the VPL's diffuse reflection of light from both the previous VPL and the light source itself. Since both the sampling of the VPL positions and the hierarchical warping can be done in image space of the light source, all other VPL data can simply be looked up in the Light G-Buffer.

4.3 Setup Imperfect Shadow Maps

The main issue for indirect illuminating of scene points is VPL-visibility. We have to know, whether a currently shaded surface point is visible to a VPL or not. If the point is occluded by some other object, light reflected at the VPL doesn't reach this point and no indirect illumination takes place. Such indirect shadows have a high contribution to scene realism. Figure 6 shows a simple scene setup demonstrating shadows cast from an indirectly illuminated object.

To store the necessary visibility information for each VPL, simple shadow-mapping is insufficient, since the area of the scene visible to a VPL covers the whole hemisphere over the VPL's surface point. Therefore, parabolic maps introduced by Brabec et al [1] are used.

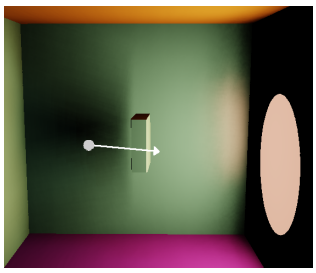

Figure 6: Indirect shadow realized by Imperfect Shadow Maps. The white sphere and arrow indicate the position and direction of the spot-lightsource. Only the wall on the right side is directly illuminated, resulting in the centered cube to cast a smooth indirect shadow.

For each VPL, we create an Imperfect Shadow Map (ISM), as proposed by [7]. To create an ISM, the parabolic projected points of the scene geometry are (box-)splatted onto a map (e.g. by rendering point sprites). The size of a box splat is thereby quadratic proportional to the depth of the point. This imperfect approximation of a perfect shadow map is sufficient for our needs and rendering is much faster this way, especially when using many VPLs. To support large numbers of VPLs, we use one huge ISM buffer which contains all ISMs for each VPL in the scene. Figure 7 illustrates a part of a huge ISM buffer (upper right side of the figure).

To be able to create all ISMs for the whole set of VPLs in the scene in one render pass, we do not render all points of the scene into each single ISM. Rather the whole scene is rendered once, and the points of the scene geometry are distributed over the ISMs as they are passed through the vertex shader. Therefore, each ISM contains only of a subset of the total number of points within the scene, and each point is rendered to only one ISM in fact. This

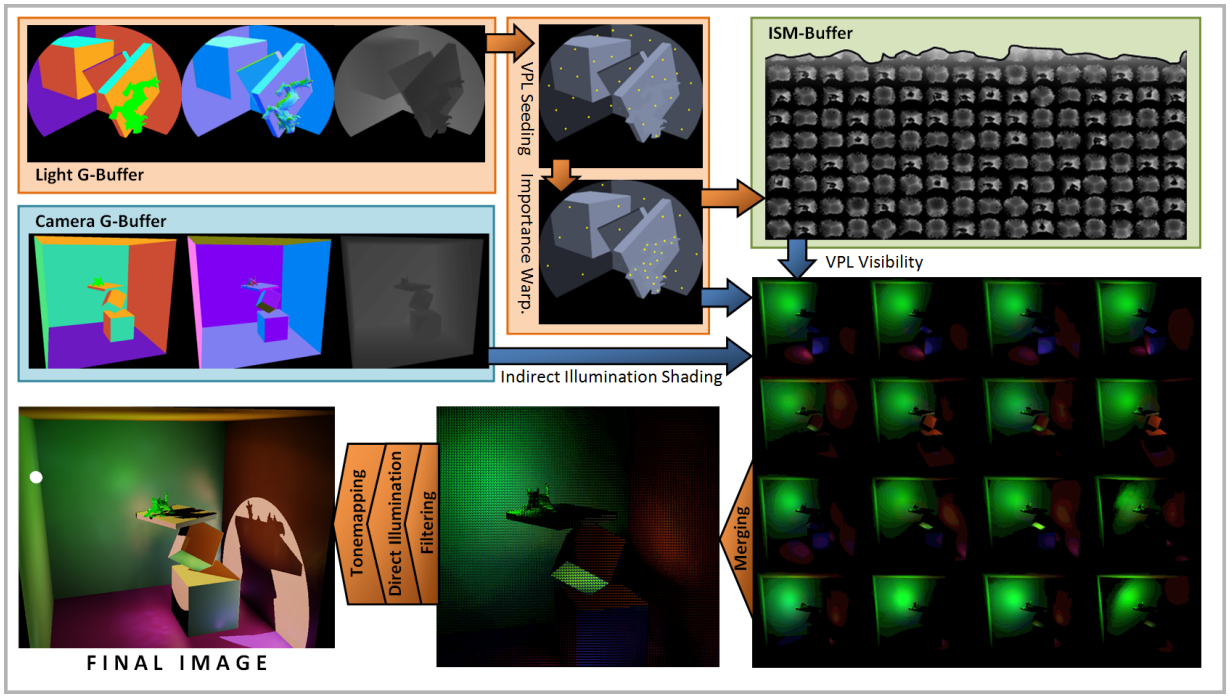

Figure 7: Overview of the GI image synthesis chain and its buffers involved.

further approximation is sufficient for the creation of an ISM, as long as a point cloud of a model within the scene doesn't consist of too few points and the relative number of VPLs is not too high. This point distribution approach makes the time consumed by the ISM setup pass independent of the number of VPLs, which allows for high image quality (high number of VPLs) at real-time frame rates. However, for models with a too low number of points, this distribution approach could certainly yield to unusable, perforated mappings within the ISMs. This problem could be addressed by e.g. rendering multiple passes on the ISM creation stage (using different point-to-ISM assignment offsets per pass), or finding a convenient ISM splat-size scaling factor per point cloud depending on the number of points per ISM and the distance to the ISM's VPL-position.

Since ISMs are created by box-splatting a subset of the points in the scene, the resulting parabolic image can contain holes. To improve the quality of the ISMs, we perform a pull-push-operation on the ISM buffer in order to fill those holes, as described by [6]. The quality of the result depends on the number of pull-push-iterations, but already a number 2-3 iterations can gain good improvements (see Figure 8).

4.4 Shading Split-G-Buffer

Based upon the visibility information encoded for each VPL in the ISM buffer, indirect illumination shading can be performed. As already mentioned, shading of indirect (and, also of direct) illumination is performed in image space, i.e. only for the pixels in the camera G-buffer. In general, for each pixel we would have to calculate the transported energy from each single VPL to the surface point corresponding with that pixel. Since this can be very time consuming with a large number of VPLs, interleaved sampling introduced by [5] is used.

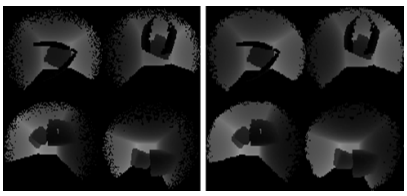

Figure 8: Comparison between a raw ISM buffer (left) and a improved ISM buffer after a pull-push-operation with 2 iterations (right).

4.5 Redistributing VPLs

When rendering multiple indirect light bounces, the VPLs have to be redistributed after each indirect illumination shading pass, in order to obtain a new set of VPLs representing surface reflection points for the next light bounce.

Similar to the method implemented by [6], we assign one new VPL to each current VPL, so that the number VPLs in the scene remains constant. All points of the scene are passed through a shader which distributes all points among the current VPLs and renders a VPL buffer containing the new set of VPLs. To select the best candidate for a new VPL per existing VPL, our shader assigns

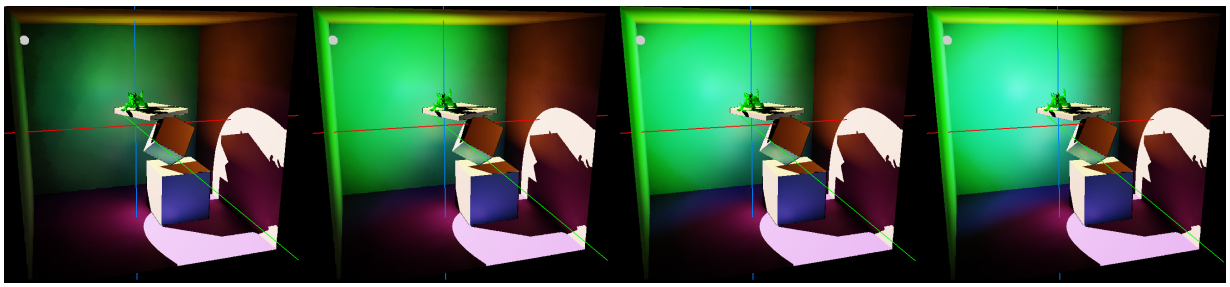

Figure 9: An object stack illuminated by a spot-light inside a Cornell Box, rendered with 256 VPLs and (from left to right) with 1, 2, 3 and 4 light bounces. Considering a distortion by the tone-mapping operator, we can clearly see that the biggest difference in the light situation is given between the first and the second bounce, while the third and fourth bounce only contributes minimal additional brightness.

to each incoming point a z-value that corresponds with the VPL quality of that point. This value is dependent on whether the new point is visible to the current VPL at all, and on how much luminance this new VPL can contribute to the result in the next bounce. Using the right setup, for each current VPL the GPU depth-test automatically leaves those scene points in the buffer, which are best suited as new VPLs. This buffer is then used as new VPL buffer for the next light bounce.

5 Results

Our algorithm supports multiple indirect light bounces at interactive frame-rates. However, in our test scenes, rendering 2-3 bounces already covered the dominant part of the global scene illumination (refer to Figures 9 and 12).

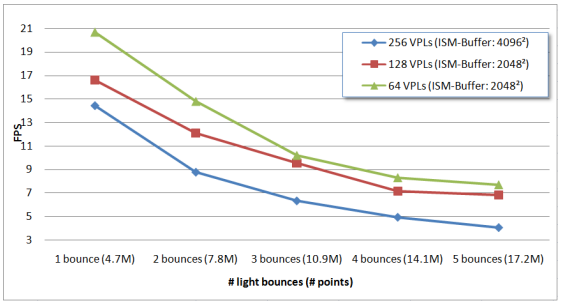

Figure 10: Frame-rate dependency from the number of light-bounces and the number of VPLs in the scene shown in Figure 9.

Figure 9 compares several light bounces in a small Cornell-Box point cloud scene (4.7M points at 1 bounce and 256 VPLs) containing a few boxes and an asian dragon model. Figure 10 illustrates the dependency of the frame-rate from the number of light bounces, the number of VPLs and the size of the ISM-buffer used in this scene. At 128 and 64 VPLs, we used a smaller ISM buffer size (with a different resolution per ISM). Note that with increasing number of light bounces, the total number of points per frame rendered increases too.

Figure 12 shows our GI render mode in the scene of the scanned-dataset of St. Stephan's Cathedral shown in Figure 2, comparing 1, 2 and 3 light bounces at both 4x4 interleaved and non-interleaved VPL shading.[6] The scene was rendered using 256 VPLs. Performance is strongly depending on the number of indirect light bounces, image quality (number of interleaving sub-panes), number of VPLs a.s.o. Figure 11 compares the frame-rates achieved for the St. Stephan's Cathedral scene in Figure 12, rendering 20.6 million points at one light bounce (increases with additional bounces).

The frame-rates shown for both the Cornell-Box and the Cathedral scene were achieved on a platform with a Intel Xeon X5550 2.67GHz CPU, with 72 GB RAM and a GeForce GTX 285 GPU with 2.8 GB RAM.

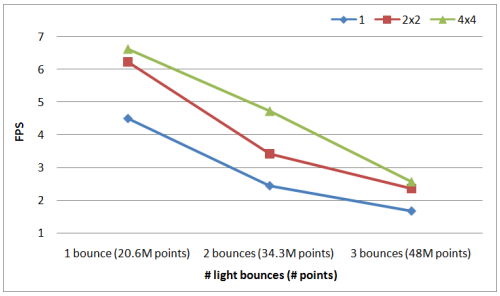

Figure 11: Comparison of frame-rates with 1, 2 and 3 light bounces using 1x1, 2x2 and 4x4 subdivisions for interleaved sampling for the scene shown in Figure 12 (20.6M points).

Due to the importance sampling of the VPL positions by hierarchical warping (concentrating more VPLs on shiny surfaces), we are also able to reproduce caustics from curved surfaces (as shown in figure 13), without the use of additional VPLs. Further, the method of hierarchically warping the VPLs works against a weakness of Virtual Point Lights: the tendency to create sparkles when

[6]Note: The holes in the scene due to an imperfect/incomplete dataset, which was acquired by 3D-scanning. We used an imperfect normal-estimation algorithm, occasionally leading to wrong or missing normals and thus to to local illumination errors.

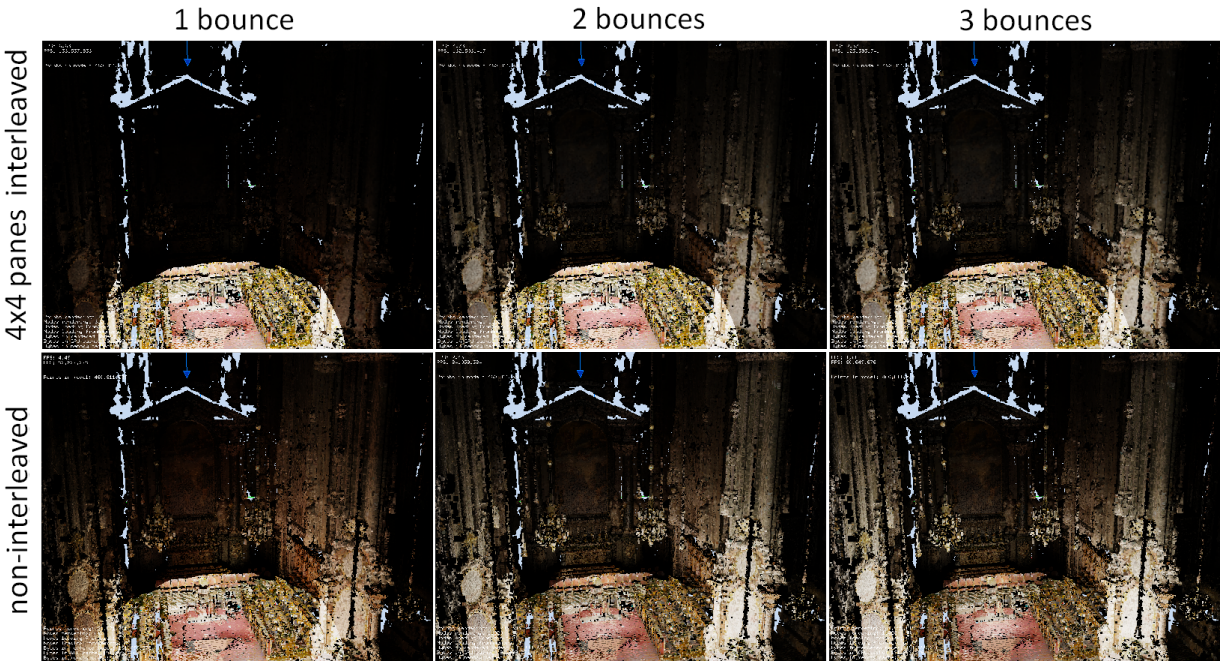

Figure 12: Comparison of different GI-parameters rendering the scanned inside of St. Stephan's Cathedral in Vienna. In the dark scene, we placed a big spotlight at the ceiling pointing at the floor. The upper row shows screens with 4x4 panes interleaved indirect illumination shading. The series in the lower row is rendered without interleaving (consuming more time), which leads to a higher overall-illumination, since each pixel is illuminated by each VPL. Note that with increasing number of indirect light bounces, the higher areas of the walls get more and more illuminated (converging at 3-4 bounces).

placed on surfaces with a high specular component (see figure 14). The appearance of those sparkles often due to an undersampling of VPLs in highly shiny areas, which can not be avoided in cases of too few VPLs in a scene with balanced reflection behavior.

We support transport of specular reflected light over several bounces, allowing for highly specular scenes. However, those situations often require a higher VPL density to avoid the mentioned sparkle effects, while a proper diffuse illumination suffices less VPLs.

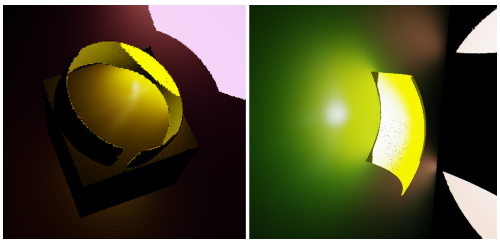

Figure 13: Left: caustic created by a ring. Right: Illumination of a parabolic surface, causing a highlight at its focal point at a nearby wall.

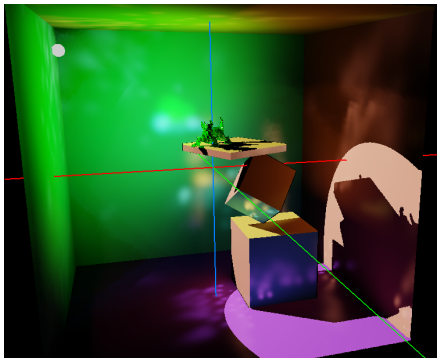

Figure 14: Scene rendered with global diffuse intensity 0.5, specular intensity 0.95 and shininess 1000. At the highly glossy surfaces, the distribution of VPLs lead to the appearance of sparkles.

6 Conclusions and Future Work

We have shown a way to perform global illumination on point cloud scenes at real-time frame-rates. It benefits from the efficiency of Imperfect Shadow Maps for visibility calculation of Virtual Point Lights in our scenes. We are able to calculate diffuse and specular reflections for multiple indirect light bounces. Our approach works best in scenes containing geometry with a high diffuse reflection component and lower specular intensity, since surfaces with too high specular intensity can result in unintended sparkle artifacts.

Our implementation yet handles only one spot-light source, since this eases the way of performing impor-

tance sampling of VPLs over a limited area (in light view space) for the first bounce. In fact, directional light sources would work the same way using just orthogonal instead of perspective projection. Point lights on the other hand should be handled differently, since they would require eight Light-G-Buffers using the same G-Buffer setup. For this light source type, the use of two parabolic maps would be more encouraged, each one storing one of its hemispheres.

Extending to multiple lightsources, a linear relation between light source count and per-lightsource VPL density is expected, when maintaining equal frame-rates. However, some kind of importance sampling over the different light sources, which adjusts the number of VPLs seeded by a single light source depending on its total scene contribution (light source intensity, distance to the viewer), could be applied to speed up rendering.

All test scenes used in this paper, even those containing plane surfaces like the Cornell Box, are fully represented by point clouds. The implementation does not support polygon models for Global Illumination rendering yet. But since the integration of a polygon model support would require the sampling for surface points on this models anyway (ISM splatting), it would be more expedient to create a preprocessed point cloud representation of the model if possible.

References

[1] Stefan Brabec, Thomas Annen, and Hans peter Seidel. Shadow mapping for hemispherical and omnidirectional light sources. In *In Proc. of Computer Graphics International*, pages 397–408, 2002.

[2] Petrik Clarberg, Wojciech Jarosz, Tomas Akenine-Möller, and Henrik Wann Jensen. Wavelet importance sampling: Efficiently evaluating products of complex functions. In *Proceedings of ACM SIGGRAPH 2005*, 2005.

[3] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[4] Alexander Keller. Instant radiosity. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 49–56, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[5] Alexander Keller and Wolfgang Heidrich. Interleaved sampling. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 269–276, London, UK, 2001. Springer-Verlag.

[6] Martin Knecht. Real-time global illumination using temporal coherence. Master's thesis, Vienna University of Technology, 2009.

[7] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.*, 27(5):1–8, 2008.

[8] Michael Wimmer and Claus Scheiblauer. Instant points, July 2006.