Screen-Space Virtual Point Light propagation for Real-Time Global Illumination

Vitor M. Aranha
Department of Computer Science
Federal University of Bahia
Salvador, Bahia, Brazil
Email: vitormoraesaranha@gmail.com

Márcio C. F. Macedo
Department of Computer Science
Federal University of Bahia
Salvador, Bahia, Brazil
Email: marciocfmacedo@gmail.com

Antônio L. Apolinário Jr.
Department of Computer Science
Federal University of Bahia
Salvador, Bahia, Brazil
Email: antonio.apolinario@ufba.br

***Abstract*—Proper light-transport simulation adds both realism and aesthetically pleasing effects to virtual 3D scenes. However, the cost of computing complex light interactions is prohibitive for real-time applications. Indirect diffuse lighting is a low frequency component of global illumination that can greatly enhance the quality of an image. Virtual point lights are commonly used to reproduce diffuse bounces by tracing light paths through the scene and creating proxy light sources at the intersections between a path and geometry. In this work[1] we extend clusterization-based methods for virtual point lights, allowing for the reproduction of up to two bounces of light with a projection-aware sampling method in real-time. We show that plausible images can be obtained in real-time rates for low to mid-end commodity GPUs.**

I. Introduction

The multitude of visual effects that arise from the interaction between light and the environment in a synthetic scene is popularly called global illumination. Their representation may yield realistic and pleasing images, as well as visual cues of the many materials that compose a scene. However, reproducing these effects may come with a high computational cost. In the movie industry, where render times are not usually a problem and the development of the final product may take hundreds of hours, the task of computing each frame may be forwarded for clusters of computers in render farms. On the other hand, in real-time media like video games, global illumination effects often require optimization and ad-hoc methods to meet time constraints.

Global illumination algorithms for real-time applications must be able to reproduce lighting effects plausibly under strict time and hardware budgets. For this reason, algorithms may be specialized to compute only a smaller subset of visual phenomena efficiently. One of these commonly reproduced effects is the indirect diffuse lighting, which represents the many scattered interreflections of light rays between surfaces throughout the scene. This low-frequency component of global illumination is multi-bounce by nature, this means that a light ray will hit more than a single object, getting attenuated and carrying color information from one surface to another.

The multiple bounces involved in this process usually make it prohibitive to represent dynamically in real time, as the multiple ray-triangle intersection tests between geometry stored non-contiguously will incur in performance hotspots (e.g. pointer chasing). Techniques used to represent this phenomenon may approach the problem in two ways, either in world-space or in screen-space.

The world-space solution tries to compute the diffuse component of light transport between the objects in world space. Approaches like this include many-light methods [1] and Path Tracing [2] for example. These techniques achieve accurate results but at a higher computational cost that grows linearly with the number of objects. On the other hand, screen-space methods like the Deep G-Buffer [3] and Directional Occlusion [4] focus only on data present in screen-space. These approaches tend to be faster but highly viewer dependent and prone to artifacts between frames.

An interesting approach to the problem of computing diffuse lighting in real-time was proposed in Clustered Visibility [5]. In that work the cost of computing indirect lighting in world-space is mitigated by reducing the number of visibility queries using clusterization techniques. However, this method is limited to single bounce diffuse illumination, leaving further bounces still an open problem.

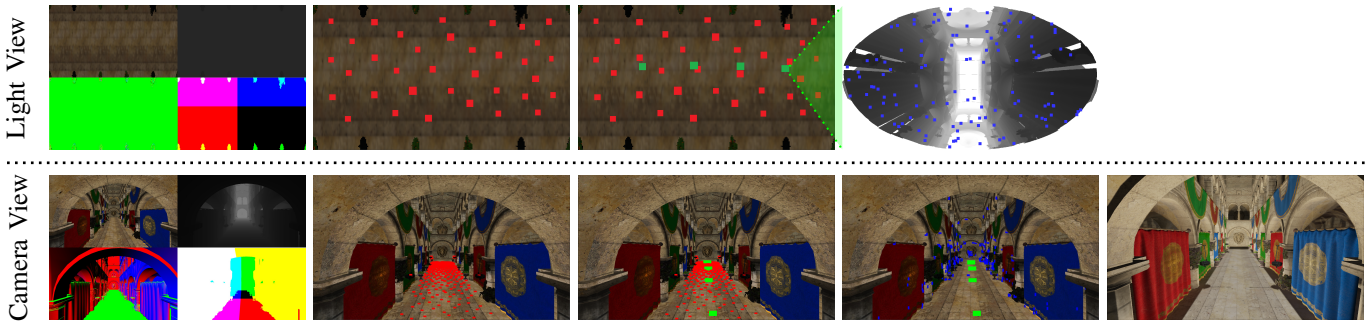
In this work we propose Screen-Space Virtual Point Light Propagation (SSVP), our global-illumination method that merges screen-space algorithms with cluster-based world-space approaches. SSVP uses a pseudo-random sampling method for paraboloidal projections that approximates the projection of an elliptical paraboloid in a 2D texture. That pseudo-random sampling strategy reduces the probability of samples falling under empty or incoherent areas of the projected image, consequently allowing for plausible real-time propagation of up to two indirect diffuse bounces of light in the scene in fast, real-time framerates. The contributions of our work are the following:

- A fast way to reproduce up to two bounces of diffuse indirect lighting for 3D scenes.
- An efficient method of sampling from paraboloid shadow maps.

II. Screen-Space VPL Propagation

The main pipeline of SSVP can be seen in Figure 1. The idea behind it is to leverage the 2D projection representing

[1] M.Sc. Dissertation



(a) G-Buffer Rendering (b) First-bounce VPL Sampling (c) First-bounce VPL Clustering (d) Second-bounce VPL Sampling (e) Final Rendering

Fig. 1. An overview of our SSVP pipeline. First, the scene is rendered from the camera's viewpoint to a G-buffer. Next, we render the scene from the viewpoint of the main light source to generate our RSM. In both cases we store depth, world-space position, albedo and normals (a). First-bounce VPLs (red squares) are sampled over the scene from the RSM (b). Afterwards, clusters (green squares) are generated by grouping similar VPLs (c). For each cluster, we generate a paraboloid map and perform a unit-disk sampling to distribute second-bounce VPLs (blue squares) over the scene (d). Shading is performed by gathering the contribution of the VPLs in the final stage (e).

the visibility of a Virtual Point Light (VPL) generated by an elliptical paraboloid, with a sampling method that matches the circular shape of its horizontal cross section. Our algorithm may be divided in two main steps and a final presentation stage: creation of first-bounce VPLs; creation of second-bounce VPLs; a final step that just gathers contribution from each light and creates the final image that is presented to the screen. The first-bounce VPLs are obtained with a traditional Reflective Shadow Map (RSM) [6]. To achieve the goal of propagating the second-bounce VPLs, we reuse previously clustered VPLs and devise a sampling scheme for their visibility maps.

A. First-bounce VPL creation

This stage requires two rasterization passes over geometry. Besides rendering from the viewpoint of the main camera, a render pass from the perspective of the main directional light source is also required. Position, normals, albedo and depth information are stored in 2D textures (Figure 1a) to generate an RSM. Each texel from this texture is considered to be a VPL placed upon a small surface patch, with its positive hemisphere oriented by the normal of this surface (Figure 1b). We sample from this buffer with a 2D Hammersley sequence [7] defined in Equation 1, for the first n elements where γ is the Van der Corput sequence.

$$\left(\frac{k}{n}, \gamma(k)\right) \text{ for k = 0, 1, 2 ... } n\text{-1} \tag{1}$$

Thinking ahead to performance, we aim at bringing the number of visibility checks to a minimum. To this end we perform K-Means clusterization over the set of sampled VPLs (Figure 1c), assigning each point light to the cluster with the nearest mean. The prototype of each cluster is itself another VPL, its position being the average of child VPLs and its orientation being the summation of each normal belonging to this set. The goal here is to use each centroid's shadow map as the representative visibility of a VPL.

The computation of the distance metric between a cluster and a VPL requires some attention. The traditional Euclidean distance is subject to many artifacts leading to clusters completely occluded, inside geometry and therefore leaking light. This can happen when VPLs are positioned closely together but in surfaces that are not coplanar. To account for these geometric properties, the normals of each VPL are introduced in the computation too, attenuating these artifacts. This distance formula was proposed by [5] and is shown in Equation 2 below.

$$D_{ij} = w_1 \|\mathbf{V}_i - \mathbf{C}_j\| + w_2(\mathbf{n}_i \cdot \mathbf{n}_j)^+ \tag{2}$$

The distance D_{ij} between VPL $\mathbf{V}_i$ and Cluster $\mathbf{C}_j$ is the weighted sum of the Euclidean distance $\|\mathbf{V}_i - \mathbf{C}_j\|$ and the clamped dot product $(\mathbf{n}_i \cdot \mathbf{n}_j)^+$ between the VPL normal n_i and Cluster normal n_j. Weights w_1 and w_2 correspond to the factors that modulate the relevance of each factor to the distance metric. We used the values of w_1 = 0.6 and w_2 = 0.4, empirically tuned for our test scenes. At this point it is important to make clear that if the scene corresponds to the interior of a convex or spherical shape, then cluster centroids may appear as floating in space due to the steady and smooth change in position between different points. Such configurations may require careful consideration when implementing the algorithm.

B. Second-bounce VPL creation

At this stage we add the main contribution of SSVP, second-bounce diffuse lighting. We render the scene from the viewpoint of each centroid to 2D textures in the same fashion as the RSM in the first step, consisting of position, normals, albedo and depth. Once again, considering the existence of several clusters to be rendered we would like to avoid the expensive rendering of cubemaps for each one. To perform a single raster pass for each hemisphere of visibility, we use paraboloid mappings [8] (Figure 1d). We intend to sample from these paraboloid reflective shadow maps in the same

fashion as did in the previous step of the algorithm. This step corresponds to the actual VPL propagation of our method.

However, as shown in Figure 1d (top) the projection obtained by the paraboloid mapping does not fit evenly inside a unit box. Instead it fills an ellipse oriented around the center of the rectangular image.

Here we are guided by an interesting observation with respect to the nature of the paraboloid used to obtain this projection. Let Equation 3 be our paraboloid of interest :

$$f(x, y) = \frac{1}{2} - \frac{1}{2}(x^2 + y^2) \tag{3}$$

An elliptic paraboloid has the form:

$$z = \frac{x^2}{a^2} + \frac{y^2}{b^2} \tag{4}$$

Rearranging Equation 3:

$$z = \frac{1}{2} - \frac{x^2}{2} - \frac{y^2}{2} \tag{5}$$

Equation 5 defines an elliptical paraboloid translated up by a factor of $\frac{1}{2}$ and with its concavity opening towards the negative z axis. Note that terms a^2 and b^2 are equal here ($a^2 = -2$ and $b^2 = -2$), what reveals another interesting property. The elliptical paraboloid is also a circular paraboloid as well. This means that any horizontal cross section of the paraboloid defines a circle. In fact the only reason it appears elliptical in Figure 1d is because of the rectangular viewport it was clipped against. This understanding guided our sampling method. Regular uniform random variables on the interval $[0, 1[$, defined on the unit square would cause samples to fall outside of the circular region of the projection, becoming wasteful and using unneeded bandwidth. Hence, we apply a unit circle uniform sampling method shown in [9] to concentrate samples on the inner boundaries of the texture inside the actual projection.

By the end of this stage we obtain second-bounce VPLs in the same fashion as done for the first-bounce. We store these point lights in arrays on the GPU and proceed to the shading step. This approach also allows us to easily toggle the further bounces on and off for visualization.

C. Shading the scene

To efficiently shade the scene with many lights, we adhere to an interleaved shading approach [10], splitting the framebuffer into smaller, equal-sized tiles with resolution directly proportional to the number of VPLs. Each tile is shaded by a different subset of lights and then interleaved back together. Because each tile represents only a fraction of the total framebuffer resolution, the number of fragments to shade is reduced considerably per light, with a speed up inversely proportional to the number of light sources per tile.

Finally, a post processing tone mapping operator [11] is applied to clamp lighting values to the 3-Channel 8 bit per color image format of the screen. Figure 1e shows the final image obtained by our method for the scene Sponza.

	4 Clusters	8 Clusters	16 Clusters	32 Clusters
Gbuffer	1.03	1.03	1.03	1.03
Misc	0.26	0.29	0.30	0.30
Paraboloid	1.16	2.50	7.90	56.29
Shade	10.44	10.90	11.08	11.13
Total(ms)	12.89	14.72	20.31	68.75

TABLE I

RELATIVE TIME OF EACH SSVP PIPELINE STAGE AS THE NUMBER OF CLUSTERS CHANGE. TIME IS GIVEN IN MILLISECONDS. MISC CORRESPONDS TO THE TIME NEEDED TO GENERATE A SHADOW MAP, TO SAMPLE FIRST-BOUNCE VPLS, TO CLUSTER THOSE VPLS AND TO SAMPLE SECOND-BOUNCE VPLS. SHADING ACCOUNTS FOR BOTH PIXEL SHADING OPERATIONS AND DENOISING.

III. RESULTS AND DISCUSSION

We ran our tests on a personal computer under the Windows 10 operational system. The hardware specifications consisted of a total of 16 gigabytes of RAM, an Intel i7 4790K CPU at stock clocks and an Nvidia GTX 1060 with 6 gigabytes of VRAM. The code prototype was developed in C++ using the OpenGL 4.6 API. For ground truth comparisons, a base image was generated by path-tracing the scenes with 256 ray samples per pixel. Three aspects are considered for evaluation: rendering performance, visual quality and temporal coherence.

We applied our method in four popular scenes from computer graphics literature: Cornell Box, Conference Room, Sibenik Cathedral and Crytek Sponza, all provided by [12]. By default we used FULL-HD resolution (1080p) together with 256 first-bounce VPLs and 256 second-bounce VPLs.

A. Rendering Performance

In order to obtain a comprehensive frametime analysis of SSVP running on different scenes, we benchmarked a test run from a fixed point of view inside each model. Figure 3 shows the average rendering times for increasing number of clusters for each test scenario.

Notice the sharp increases in rendering times as the number of clusters goes up. For all test scenes 16 Clusters appears to be the hard threshold for hard real-time scenarios, around 16 milliseconds per frame [13]. To better understand the phenomenon behind this steep growth we breakdown an example frame for the scene Sponza in Table I.

Each time slice corresponds to a specific step of our pipeline. Notice as the number of clusters grow, the main culprit for the increased rendering times is the Paraboloid Creation step. The time required to render paraboloids grows by a factor seemingly proportional to the raise in the number of clusters (4, 8 and 16), beyond this point (at the change to 32) time grows steeper in a spike. Each paraboloid pass implies in rendering the geometry as many times as the number of clusters. Preliminary investigation of this behavior implies in increasing cache misses and points synchronization as we grow the number of render passes in geometry shaders, but no experiments with different hardware configurations were performed therefore a decisive evaluation can not be provided here.

Because this behavior is consistent between scenes, we advise the reader to employ caution when implementing the

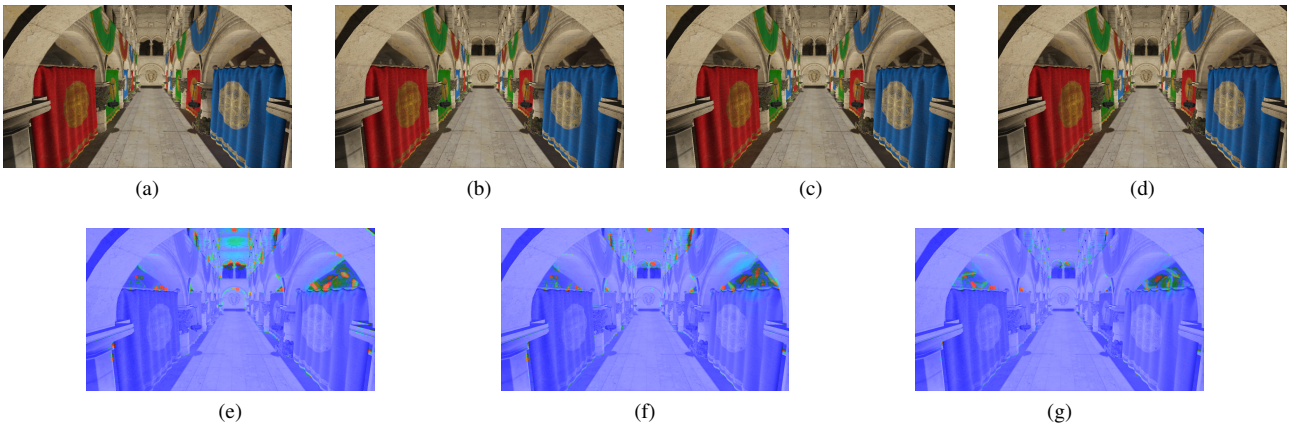


Fig. 2. HDR-VDP-2 metric applied to images generated with 4, 8 and 16 clusters (Figures 2a, 2b and 2c respectively) in comparison to 32 clusters in Figure 2d. Figures 2e, 2f and 2g show the perceptual error metric for 4, 8 and 16 clusters when compared to the base image with 32 clusters.

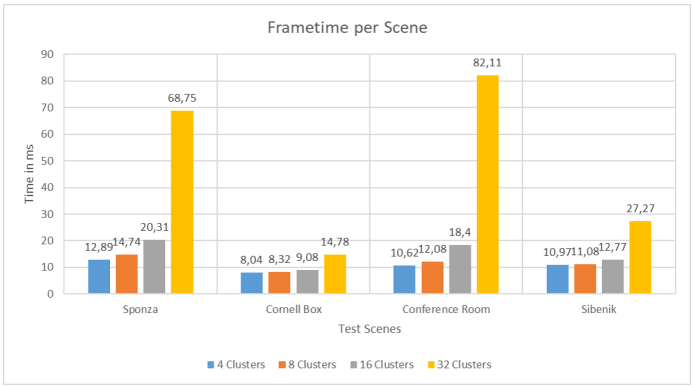


Fig. 3. Average frametimes of our algorithm for four different scenes, running on a viewport with 1920x1080 resolution, 4x4 interleaved sampling pattern and increasing cluster configurations.

technique, picking a number of clusters that offers a good trade-off between performance and quality is paramount to maintain good framerates.

B. Visual Quality

We carry the visual quality tests with a ground truth comparison. Each reference image was generated with the industry standard Path-Tracing technique. The comparison shows the impact of allowing direct light to bounce two times through the scene, and compare the final image to the ground truth. Figures 4, 5, 6 and 7 show how the addition of second-bounce VPLs introduces some effects such as color bleeding, which can be seen nearby the curtains in Sponza for example and make the shadows brighter, giving a proper sense of indirect lighting.

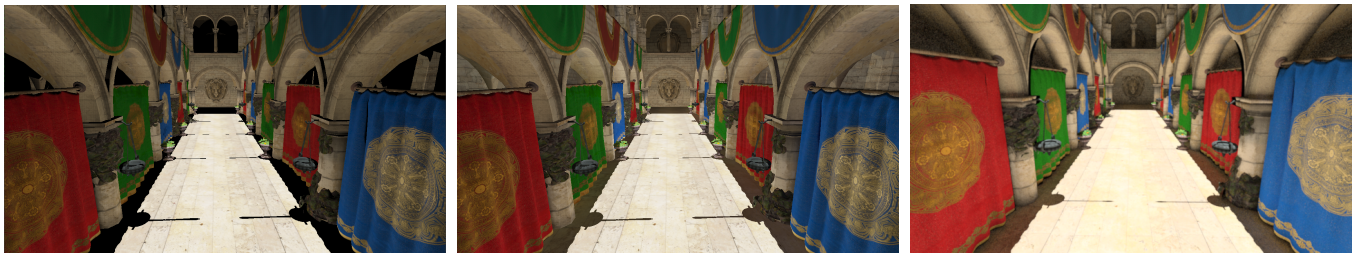
Two different lighting conditions were used. In the Sponza scene, directional lighting, similar to sun rays come from the top of the atrium and reach the floor of the main hall. Because the other scenes consist mainly of closed spaces, Sibenik, Conference Room and Cornell Box, we shoot a spotlight against a wall and let the VPLs propagate from it.

As observed in Table I, adding more clusters impacts the performance negatively, on the other hand the quality of the final image is expected to increase due to more precise visibility computations. For the hypothetical scenario where we have as many clusters as VPLs, maximum quality is achieved and the technique is indistinguishable from the original Instant Radiosity [14]. This would obviously beat the purpose of SSVP, that is reproducing plausible indirect lighting in real-time. For this reason we also conduct an experiment to evaluate the visual impact of increasing cluster numbers up to the tested maximum of 32. The visual perception metric HDR-VDP-2 [15] is employed in this test to quantify the quality of an image against another one of better quality.

The root-mean-square error obtained for 4 clusters is 0.1259, 8 clusters is 0.1020 and 16 clusters is 0.0806. The smaller the value, smaller the difference between an image and the reference being used. The main reasoning here is that the more clusters, the higher is the overall quality of the algorithm, therefore we try to obtain a score of how close lesser configurations are to higher ones, better approaching the ground truth for example.

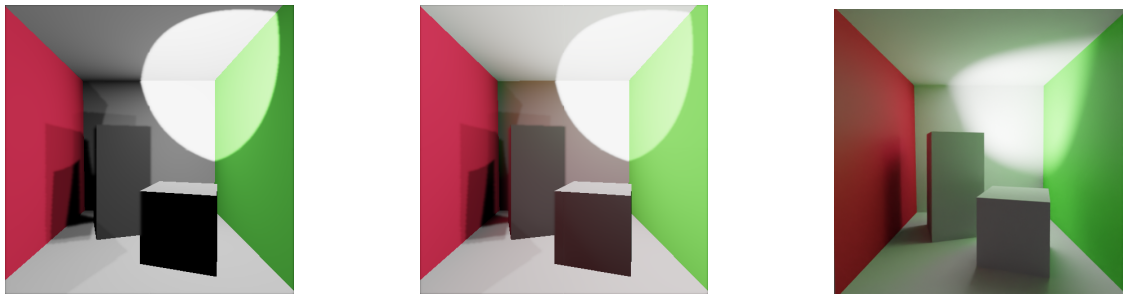
C. Temporal Coherence

The problem of temporal coherence is a complex field that has its own specific literature [16], for this reason we did not devise any specific method to deal with it in SSVP. Mitigation of possible temporal artifacts are handled by a few algorithmic decisions during the course of the pipeline. First, we reuse the sampling patterns between frames in order to prevent VPLs from moving too much. Second, cluster seeds are initialized from previous positions, preventing a jittered motion that could cause quickly changing visibility maps. As a result, SSVP is a stable technique for camera movements and smooth slow light transitions. However, as a drawback of the low amount of paraboloid maps used, some artifacts similar to black stripes can happen during quick light movement as a side effect of cluster centroids moving along surfaces.



(a) SSVP (One Bounce) (b) SSVP (Two Bounces) (c) Path-Tracing (Ground Truth)

Fig. 4. Difference between single-bounce (a) and double-bounce (b) lighting rendered by our technique when compared to the ground-truth (c) on the scene Crytek Sponza.

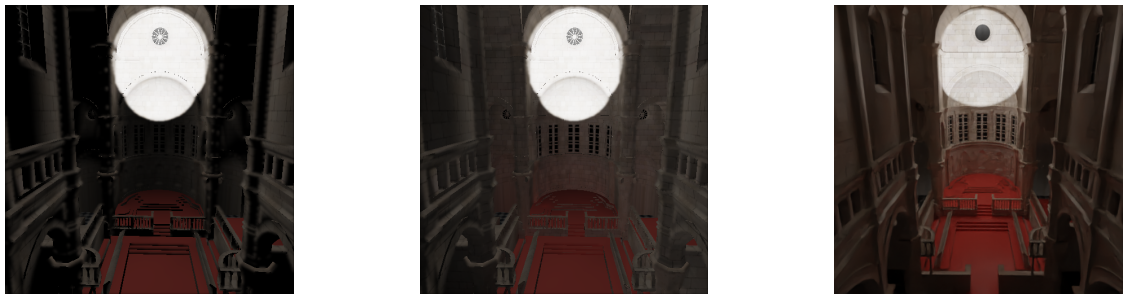


(a) SSVP (One Bounce) (b) SSVP (Two Bounces) (c) Path-Tracing (Ground Truth)

Fig. 5. Difference between single-bounce (a) and double-bounce (b) lighting rendered by our technique when compared to the ground-truth (c) on the scene Cornell Box.

(a) SSVP (One Bounce) (b) SSVP (Two Bounces) (c) Path-Tracing (Ground Truth)

Fig. 6. Difference between single-bounce (a) and double-bounce (b) lighting rendered by our technique when compared to the ground-truth (c) on the scene Conference Room.



(a) SSVP (One Bounce) (b) SSVP (Two Bounces) (c) Path-Tracing (Ground Truth)

Fig. 7. Difference between single-bounce (a) and double-bounce (b) lighting rendered by our technique when compared to the ground-truth (c) on the scene Sibenik Cathedral.

D. Discussion

To understand the results obtained by our tests and how they relate to the technique we must restate our goals with this work: extend cluster based indirect illumination methods to up to two diffuse bounces quickly for hard real-time scenarios. With this in mind, the algorithm alone must not exhaust the entire budget of 16 milliseconds, otherwise there would be no time for other tasks such as logics of physics updates.

Evaluation of the performance data present in Table I, shows that SSVP is able to achieve frame rates of 13 milliseconds for no more than 8 clusters, in a scene like Sponza for example at FULL-HD resolution. The inclusion of a visual perception metric such as HDR-VDP-2 confirms this configuration is indeed the best trade-off in our case. We extend this observation to Conference Room and Sibenik Cathedral as well, both performing well below the threshold for the same 8-cluster configuration. The exception here being Cornell Box, its low polygon count allowed it to perform well even for 32 Clusters.

To reach a conclusive answer for which configuration is the optimal these values should be contextualized with the frametime analysis. Whenever performance must be favored instead of visual quality, only a few clusters (4 in this case) are enough. If the user is aiming at a compromise solution, 8 Clusters still offer rates below 16ms and better quality.

The visual impact of adding the second-bounces is evident even for low cluster counts. Our visual comparisons show how much more detail is added to the scene, but our empiric experiments suggest maintaining the proportion Second-Bounces per Cluster equal or above 32, that means at least 32 new second-bounce VPLs must be sampled per cluster to obtain a good distribution of lights in the scene. That means for 4 clusters, 128 total second bounces are the minimum, for 8 clusters its 256, for 16 clusters its 512 VPLs and so on.

IV. Conclusion

Our method offers an extension to cluster-based methods for many-light techniques. In this work we were able to reproduce up to two diffuse bounces of light with a clusterization method for VPLs with the help of a sampling approach that closely matches the 2D projection of a circular paraboloid map.

The method proposed here achieves real-time frame rates, proving itself to be fast enough to be applicable to hard real-time conditions such as video games and capable of performing under 16 milliseconds per frame while still offering plausible quality when compared to a ground truth image. The results of our method also present reasonably stable temporal coherence for smooth and slow light transitions and good coherence for moving cameras. Finally, because of its specialization in a specific effect of global illumination, possible future works to be explored include the addition of participating media and other visual phenomena as well.

Finally, we consider the current generation of hardware enabled ray-tracing to bring many possibilities for hybrid global illumination algorithms. Because rasterization still dominates real-time rendering applications, general advances in this field will probably require ray-tracing methods to coexist with them. Next advances could particularly be employed in areas where ray-tracing thrives such as mirror reflections and ray-traced shadows, to extend and complement SSVP for instance.

V. Publications

During the development of this M. Sc. dissertation, we obtained the publication [17], awarded one of the three best conference papers at Sibgrapi 2020.

Acknowledgment

The authors would like to thank the National Council for Scientific and Technological Development (CNPq) and its project INCT-MACC for the financial support provided to carry this research.

References

[1] C. Dachsbacher, J. Křivánek, M. Hašan, A. Arbree, B. Walter, and J. Novák, "Scalable realistic rendering with many-light methods," in *Computer Graphics Forum*, vol. 33, no. 1. Wiley Online Library, 2014, pp. 88–104.

[2] J. T. Kajiya, "The rendering equation," in *Proceedings of SIGGRAPH*. ACM, 1986, pp. 143–150.

[3] M. Mara, M. McGuire, D. Nowrouzezahrai, and D. P. Luebke, "Deep g-buffers for stable global illumination approximation." in *High Performance Graphics*, 2016, pp. 87–98.

[4] T. Ritschel, T. Grosch, and H.-P. Seidel, "Approximating dynamic global illumination in image space," in *Proceedings of the I3D*. ACM, 2009, pp. 75–82.

[5] Z. Dong, T. Grosch, T. Ritschel, J. Kautz, and H.-P. Seidel, "Real-time Indirect Illumination with Clustered Visibility," in *Proceedings of the VMV*, vol. 9, 2009, pp. 187–196.

[6] C. Dachsbacher and M. Stamminger, "Reflective shadow maps," in *Proceedings of the I3D*. ACM, 2005, pp. 203–231.

[7] T.-T. Wong, W.-S. Luk, and P.-A. Heng, "Sampling with hammersley and halton points," *Journal of graphics tools*, vol. 2, no. 2, pp. 9–24, 1997.

[8] W. Heidrich and H.-P. Seidel, "View-independent Environment Maps." in *Workshop on Graphics Hardware*, 1998, pp. 39–45.

[9] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. Morgan Kaufmann, 2016.

[10] B. Segovia, J. C. Iehl, R. Mitanchey, and B. Péroche, "Non-interleaved deferred shading of interleaved sample patterns," in *Graphics Hardware*, 2006, pp. 53–60.

[11] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda, "Photographic tone reproduction for digital images," in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, 2002, pp. 267–276.

[12] M. McGuire, "Computer Graphics Archive," July 2017. [Online]. Available: https://casual-effects.com/data

[13] T. Akenine-Möller, E. Haines, and N. Hoffman, *Real-time rendering*. Crc Press, 2019.

[14] A. Keller, "Instant Radiosity," in *Proceedings of the SIGGRAPH*. ACM, 1997, pp. 49–56.

[15] R. Mantiuk, K. J. Kim, A. G. Rempel, and W. Heidrich, "Hdr-vdp-2: A calibrated visual metric for visibility and quality predictions in all luminance conditions," *ACM Transactions on graphics (TOG)*, vol. 30, no. 4, pp. 1–14, 2011.

[16] D. Scherzer, L. Yang, O. Mattausch, D. Nehab, P. V. Sander, M. Wimmer, and E. Eisemann, "Temporal coherence methods in real-time rendering," in *Computer Graphics Forum*, vol. 31, no. 8. Wiley Online Library, 2012, pp. 2378–2408.

[17] V. M. Aranha, M. C. Macedo, and A. L. Apolinario, "Screen-space vpl propagation for real-time indirect lighting," in *2020 33rd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. IEEE, 2020, pp. 46–53.