Pacific Graphics 2019
C. Theobalt, J. Lee, and G. Wetzstein
(Guest Editors)

Volume 38 (2019), Number 7

High Dynamic Range Point Clouds for Real-Time Relighting

Manuele Sabbadin[1], Gianpaolo Palma[1], Francesco Banterle[1], Tamy Boubekeur[2], Paolo Cignoni[1]

[1]Visual Computing Lab - ISTI CNR, Pisa, Italy
[2]Telecom Paris, Institut Polytechnique de Paris & Adobe

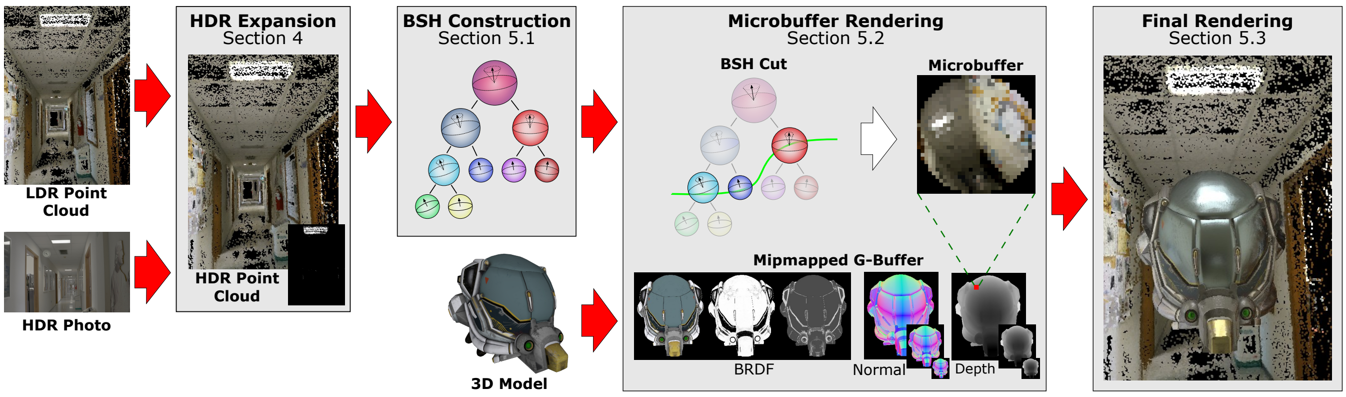


Figure 1: *Our relighting framework can be split into two parts: the preprocessing of the LDR point cloud and the real-time PBGI rendering of a 3D model inside the acquired environment. In the preprocessing step, our algorithm expands the dynamic range of the point cloud using an input HDR photo (the inset shows the difference map from the LDR version) and computes the BSH to use in the following PBGI algorithm. The proposed PBGI algorithm takes advantage of the computation capabilities of the Geometry Shader and of a new mipmapping operator for the G-Buffer to speed-up the computation of the micro-buffers of each pixel of the viewport. Finally, the collected micro-buffers are convolved by the BDRF of each unprojected pixel to render the final image.*

Abstract

Acquired 3D point clouds make possible quick modeling of virtual scenes from the real world. With modern 3D capture pipelines, each point sample often comes with additional attributes such as normal vector and color response. Although rendering and processing such data has been extensively studied, little attention has been devoted using the light transport hidden in the recorded per-sample color response to relight virtual objects in visual effects (VFX) look-dev or augmented reality (AR) scenarios. Typically, standard relighting environment exploits global environment maps together with a collection of local light probes to reflect the light mood of the real scene on the virtual object. We propose instead a unified spatial approximation of the radiance and visibility relationships present in the scene, in the form of a colored point cloud. To do so, our method relies on two core components: High Dynamic Range (HDR) expansion and real-time Point-Based Global Illumination (PBGI). First, since an acquired color point cloud typically comes in Low Dynamic Range (LDR) format, we boost it using a single HDR photo exemplar of the captured scene that can cover part of it. We perform this expansion efficiently by first expanding the dynamic range of a set of renderings of the point cloud and then projecting these renderings on the original cloud. At this stage, we propagate the expansion to the regions not covered by the renderings or with low-quality dynamic range by solving a Poisson system. Then, at rendering time, we use the resulting HDR point cloud to relight virtual objects, providing a diffuse model of the indirect illumination propagated by the environment. To do so, we design a PBGI algorithm that exploits the GPU's geometry shader stage as well as a new mipmapping operator, tailored for G-buffers, to achieve real-time performances. As a result, our method can effectively relight virtual objects exhibiting diffuse and glossy physically-based materials in real time. Furthermore, it accounts for the spatial embedding of the object within the 3D environment. We evaluate our approach on manufactured scenes to assess the error introduced at every step from the perfect ground truth. We also report experiments with real captured data, covering a range of capture technologies, from active scanning to multiview stereo reconstruction.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Computer graphics; Rendering; Rasterization; Image processing;*** *Point-based models; Mixed / augmented reality;*

1. Introduction

Relighting a virtual 3D object in the context of a real scene is a challenging but mandatory process for the realistic integration of digital assets in augmented reality (AR) applications. One of the main steps of such a process consists of shading the virtual object with a model of the lighting that is as close as possible to the real light field of the scene. Currently, this model takes typically the form of one or several High Dynamic Range (HDR) environment maps, reconstructed from several photos or videos of the scene and used as infinitely distant light emitters [ZL17, RPAC17]. Although efficient at approximating incoming light in outdoor environments, this Image-Based Lighting (IBL) falls short when nearby light transport effects and visibility relationships cannot be ignored. Scenarios as simple as moving an object in a corridor reveals the lack of dimensionality of a 2D map quickly when used as a substitute for the ideal – but intractable – plenoptic light field. Eventually, one needs to represent the scene's lighting in three dimensions, with a lighting model that can capture (dis)occlusion events when moving the object in 3D space. Interestingly, numerous technologies exist for quick and automatic acquisition of large 3D point clouds of environments, either using low budget hardware such as RGB-Depth cameras (Microsoft Kinect, Intel RealSense, StructureSensor, etc.) [WSG*16, DNZ*17] or even simple RGB cameras together with multiview reconstruction algorithms [SF16]. These devices provide a dense sampling of the 3D scene, where each sample carries several attributes (e.g., position, normal vector and color). Assuming a diffuse scene, we can use the colored point cloud as an approximate radiance cache and computing its reflection on a virtual object ends up effectively relighting it.

Starting from this observation, we propose a framework to relight virtual objects in real time using a 3D point sampling of a real scene (Fig. 1). To do so, we address the two main challenges induced by the use of a point cloud as a relighter. First, as captured colored point clouds typically come in Low Dynamic Range (LDR) format, we propose a new method to reconstruct an HDR colored point cloud from an LDR one using a single representative HDR image of the scene. Second, at rendering time, we need to solve, in real time, for the visibility between the point cloud and the virtual object. In particular, given a location on the virtual object, we seek for the visible subset of the point cloud that should be considered to shade it. We address this problem by developing a new variant of the Point-Based Global Illumination (PBGI) algorithm. Combined with IBL for distant direct illumination and screen-space reflections for local color bleeding, our approach provides a convincing real-time solution for a collection of AR scenarios such as games, visual special effects previz, and computer-aided design.

Contributions. The paper presents two complementary contributions:

- a method to expand the dynamic range of a colored 3D point cloud using a single HDR photograph covering a small part of the environment and without any calibration data; i.e., without the estimation of the intrinsic and extrinsic camera parameters;
- a rendering pipeline inspired from the PBGI algorithm that can shade virtual objects using the HDR point cloud in real time, exploiting the Geometry Shader stage to traverse a spatial hierarchy, indexing the point cloud, efficiently and introducing a G-Buffer mipmapping mechanism to speed-up the indirect visibility determination.

We report experiments conducted with our framework on captured and synthetic data to perform ground truth comparisons. We evaluate each stage of our framework individually by comparing the output of the proposed HDR expansion algorithm numerically and measuring the differences introduced by the HDR point cloud in the rendering. We also report an exhaustive performance analysis of the proposed PBGI algorithm, illustrating its use in an AR application scenario where the rendering of the object is composited with the original media used for the acquisition of the point cloud (Fig. 11). In our framework, we never "render" the HDR point cloud but use it as a radiance cache for relighting virtual objects.

2. Related Work

2.1. Inverse Tone Mapping

Inverse/reverse tone mapping operators (ITMOs/RTMOs) [RWP*10, BADC17] expand the dynamic range of LDR images/videos to obtain content that can be employed in HDR applications such as HDR visualization or Image-Based Lighting [Deb98]. Typically, they can be classified into three main classes: global, expand map-based, and user-based. A global ITMO expands the dynamic range of LDR content using a per-pixel function, such as a linear scale [AFR*07] or power function [Lan02, MSG15], with global statistics. Expand map-based operators [BLDC06, RTS*07, KO14] increase the dynamic range using a global function (i.e., linear, inverse sigmoid, etc.) that varies locally using an expand map (i.e., a map with values in $[0,1]$), which controls the areas need to be expanded and the relative intensity. Finally, user-based operators expand the dynamic range of LDR content with user inputs such as classifying areas [DMHS08] or cloning details from well-exposed areas into under-exposed and over-exposed regions [WWZ*07]. Recently, deep learning has been employed with success for expanding LDR content [EKD*17, EKM17, ZL17, MBRHD18]. However, these approaches are not straightforward to apply to point clouds because large training sets (i.e., point clouds with HDR color data) are challenging to acquire and there are very few publicly available datasets. A general framework for enhancing (including dynamic range) videos using reference photos was proposed by Baht et al. [BZS*07]. Starting from a set of reference photos of the entire scene, the framework requires Structure-from-Motion and a dense Multi-View Stereo to transfer the details in the video. In our case, we have only a single HDR input photo to reduce acquisition time with a lightweight framework without camera calibration.

2.2. Point Based Global Illumination

For a survey on real-time global illumination methods, we refer the reader to the state-of-the-art report by Ritschel et al. [RDGK12] and to the work of Silvennoinen and Lehtinen [SL17] for a recent overview. In our context, working with dense point clouds naturally led us to adopt the *Point-Based Global Illumination* framework. Introduced by Christensen [Chr08], who built upon surfel-based ambient occlusion [Bun05] and *LightCuts* [WFA*05], the PBGI algorithm generalizes the idea of z-buffered rasterization in a 2-step process. At caching time, the 3D scene is densely sampled, the sample set is shaded and a multiresolution structure, e.g., bounding sphere hierarchy or octree, is generated over it. At rendering time, for each receiver, e.g., unprojected final image pixel, **(i)** a so-called *micro-buffer* (a low resolution color+depth hemispherical buffer aligned to the shaded point normal) is generated, **(ii)** an adaptive light cut is searched in the PBGI tree, and **(iii)** the retrieved cut nodes are splatted into the micro-buffer, solving for visibility using the depth component. The final receiver response, e.g., pixel color, then sums the convolution of its micro-buffer by its BRDF with its direct illumination response. This algorithm is free from noise, accounts for long-range indirect lighting and reproduces an important subset of GI effects. Its evolutions demonstrate high scalability for parallel architectures [REG*09, HREB11] and out-of-core execution [Tab12], robustness to compression [BB12] and factorization [WHB*13], the ability, to a certain extent, to cope with non-diffuse effects [WMB15], and scalability to render complex scenes from a very large number of viewpoints [KBLE19]. Our key observation is that a 3D scanning colored point cloud already provides the input of a PBGI tree avoiding the significant amount of work requested at caching time. At the same time, its 3D spatial embedding allows accounting for near-field effects and local visibility relationship when used to relight a virtual object.

2.3. Relighting

Relighting synthetic objects using natural real-world lighting (e.g., HDR environment maps) is an important topic in computer graphics, sparking a vast literature from the seminal work on IBL [Deb98]. Over the years, researchers have worked on different subtopics: increasing the realism of classic IBL to have local effects (e.g., local shadows, shading, and caustics) by densely sampling the environment [UKL*13], editing the lighting [Pel10, BCD*13], inserting virtual objects in single photographs [KHFH11, KSH*14, GSY*17, HSH*17] or in a single RGB-D frame [XLL*18]. Many methods are limited to the portion of the scene which is visible in the photograph. They are based on the extraction of the most important light sources, visible or invisible, ignoring the lighting contribution of the other part of the scene, that can be critical for the lighting of, e.g., glossy objects. Moreover, these methods focus on the estimation of the lighting environment, relying on a final rendering, based on ray tracing, that does not aim at real-time performance. For a complete overview of this topic, we point the reader to the survey by Kronander et al. [KBG*15].

Research more focused on our work is MR360 by Rhee et al. [RPAC17], an interactive mixed reality system based on an LDR 360 panoramic video. They employed a simple inverse tone mapping operator to enhance the LDR panoramic video reproducing only lighting effects due to directional lights. Zhang et al. [ZCC16] presented a system for capturing scenes using RGBD scanners tailored for emptying and refurbishing indoors. Starting from an RGBD scan, it produces a scene model of the empty room, its lighting emitters, and its materials. Although they integrate a set of LDR photos with auto-exposure, the method cannot return the entire dynamic range of the scene. Finally, Whelan et al. [WSG*16] introduced ElasticFusion, a real-time dense visual SLAM algorithm with the estimation of the position of the main light sources exploiting the scene's geometry and specular regions. This estimation makes possible the insertion and rendering, with coherent lighting, of synthetic objects in augmented reality applications.

3. Algorithm Overview

Our goal is the real time relighting of a virtual object inside a real scene using an acquired colored point cloud of the environment. The inputs of our method are **(i)** P_{LDR}, the colored point cloud of the environment, **(ii)** I_{HDR}, an HDR photograph of a representative portion of the scene free from registration (intrinsic and extrinsic camera parameters are unknown), and **(iii)** O, the 3D polygon mesh of the object to relight equipped with material properties. P_{LDR} can be acquired automatically using inexpensive hardware such as an RGBD camera [DNZ*17], or with a simple RGB camera together with a multiview stereo software [SF16]. Since the output clouds of these devices come with LDR color data, the first component of our framework is a procedure designed to expand the dynamic range of the overexposed regions of P_{LDR} using I_{HDR} (Sec. 4). Then, at rendering time, the resulting HDR point cloud P_{HDR} is used for relighting the 3D model of the virtual object O using the second component of our framework: a new PBGI algorithm bootstrapped directly from P_{HDR} and designed to achieve real-time performance (Sec. 5). Fig. 1 shows an overview of the proposed algorithm.

4. HDR Expansion of Point Clouds

To expand the dynamic range of P_{LDR}, we assume that I_{HDR} provides a representative distribution of the dynamic range in the 3D scene, acquiring some of the overexposed areas. We do not estimate the intrinsic and extrinsic camera parameters to align I_{HDR} to the point cloud, making the capture process easy, since I_{HDR} can be taken completely independently of P_{LDR}. For the same reason, we do not calibrate the response function of the acquisition devices, but we remove only the gamma correction from the input images and point clouds assuming $\gamma = 2.2$. To inject the dynamic range of I_{HDR} in P_{LDR}, we propose a new algorithm (Alg. 1) which leverages an efficient matching method running in 2D and propagate the dynamic expansion to the entire 3D point cloud. In the first step (Sec. 4.1), we extend the dynamic range of a set S of LDR renderings of P_{LDR} using a randomized matching method before backprojecting the resulting HDR images on P_{LDR}. In the second step (Sec. 4.2), we fill the missing or corrupted HDR values in the gradient domain, yielding P_{HDR}.

Algorithm 1 Point Cloud HDR Expansion

```
Input: LDR point cloud $P_{LDR}$, HDR photo $I_{HDR}$
Output: HDR point cloud $P_{HDR}$
1: $S \leftarrow$ CREATELDRRENDERINGS($P_{LDR}$)
2: $I_{LDR} \leftarrow$ FINDBESTEXPOSURE($I_{HDR}, P_{LDR}$)
3: $P^*_{HDR} \leftarrow P_{LDR}$
4: for all $S_i \in S$ do
5:     $NNF_i \leftarrow$ GENERALIZEDPATCHMATCH($S_i, I_{LDR}$)
6:     $S_{HDR_i} \leftarrow$ HDRMAPPING($NNF_i, I_{HDR}$)
7: $P^*_{HDR} \leftarrow$ PROJECT($S_{HDR}$)
8: $P_{HDR} \leftarrow$ GRADIENTDOMAININFILLING($P^*_{HDR}$)
```

4.1. HDR Expansion and Backprojection

In this step, we want to estimate the HDR data in the overexposed regions of a set of unlit LDR renderings S of P_{LDR}. The renderings have to be taken from reasonable points of view so that they cover as much as possible the overexposed areas of the scene. Even if there exist solutions to extract these points of view automatically [DBGBR*14], we merely proceed manually, letting the user navigate the scene and generate freely S (Alg. 1 line 1). We perform the actual rendering by point splatting P_{LDR} onto 1024×1024 viewports, storing the camera matrix for future backprojection needs.

In the next stage, we seek an LDR version I_{LDR} of I_{HDR} with an exposure as close as possible to the one used for the color acquisition in P_{LDR}, so that I_{LDR} can act as a good indirection to relate I_{HDR} and P_{LDR} (Alg. 1 line 2). We proceed by looking for the best exposure value e that minimizes the Wasserstein distance $\mathcal{W}$ between the histogram $\mathcal{H}$ of P_{LDR} and the histogram $\mathcal{H}$ of I_{LDR}, resulting from a simple tone-mapping of I_{HDR} based on the application of the exposure e and a simple clamping operation (see Eq. 2). Computing the histograms on the luminance channel of the XYZ color space using 128 bins, we set the following minimization problem:

$$\underset{\log_2(e)\in[a,b]}{\arg\min} \; \mathcal{W}(\mathcal{H}(P_{LDR}), \mathcal{H}(I_{LDR}(e))) \tag{1}$$

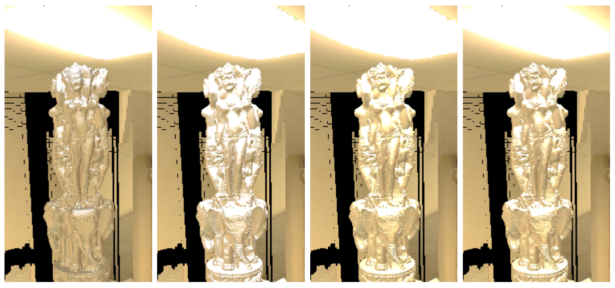
$$I_{LDR}(e) = \text{clamp}(e \cdot I_{HDR}, 0.0, 1.0). \tag{2}$$

We find the solution of Eq.1 with a simple iterative procedure, where at each step we sample uniformly (100 samples) the log-space of the exposure range $[a,b]$, and we select the exposure value that produces the minimum Wasserstein distance. At each iteration, we half the range search, centering it around the exposure value selected in the previous one. We iterate the procedure until convergence i.e., the error does not decrease or a maximum number of iterations has been reached (10 in practice). We initialize the whole process using the exposure range $[a,b]$ of the images retrieved using the exposure fusion operator [MKR07] on I_{HDR}. As a result, we obtain I_{LDR} with a color range that is closer to P_{LDR}, increasing the performance of the upcoming matching process.

Then, we expand the dynamic range of each rendering in S with I_{HDR} through I_{LDR}, without any calibration data. We propose a new alternative strategy to the state-of-the-art inverse tone mapping solutions, that our experiments proved to be more robust to the incomplete and noisy nature of a point cloud (see Sec. 6.2 for a comparison). In particular, we run the generalized PatchMatch algorithm [BSGF10] between each rendering S_i and I_{LDR}, leveraging its robustness to rotation and scale due to a multi-resolution approach (Alg. 1 line 5). For each pixel in S_i, we compute the offset in I_{LDR} to retrieve the most similar patch, defining the nearest neighbor field (NNF), and a quality field as the Sum of Squared Differences (SSD) between patches. Then we use the (LDR-defined) NNF to fetch HDR values from I_{HDR} and substitute LDR values in S_i, yielding expanded HDR renderings (Alg. 1 line 6). Note that the matching procedure can create blurry images S_i due to the size of the patches. This effect is indeed not a problem for our framework, because we are interested in the global distribution of the real radiance in the point cloud to obtain a coherent object relighting.

Finally, we backproject the expanded HDR values from S_i to the point cloud, using the camera matrix stored for each image S_i (Alg. 1 line 7). We consider only over-exposed areas, i.e., points with an original LDR luminance above the threshold $t_l = 0.9$ where the camera sensor starts to acquire poorly, as shown in [RTS*07]. For all the other points, we assign the original LDR color as HDR color assuming that the input point cloud was acquired with a reasonable exposure, as is usual for video acquisition devices. Since over-exposed areas have the dominant contribution during relighting, we expand these regions only, leaving unchanged the under-exposed ones. For points visible in more renderings, we compute a weighted average of the candidate HDR values, using the quality field to weight more heavily candidates with a low SSD. In the backprojection, we apply the exposure value computed for generating I_{LDR} to align the black values of I_{HDR} to P_{LDR}. This avoids abrupt color/luminance changes near the boundary between regions with the original LDR color and regions with the projected HDR color.

Our HDR expansion can run in two distinct modes by either extending the luminance range or the entire color from I_{HDR}. In the latter case, the overexposed areas remain closer to the original ones instead of leaning toward boosted whites, which is instrumental when the acquisition hardware introduces chromatic color shift artifacts. For instance, expanding only the luminance range is useful when PatchMatch introduces artifacts due to colors in the point cloud which are not present in the HDR image. Fig. 2 shows the differences between both modes.



(a) *LDR* **(b)** *HDR lum* **(c)** *HDR color* **(d)** *Ref.*

Figure 2: *Expanding the color as in (c) leads to results closer to the ground truth (d) than expanding only the luminance (b). (a) shows the rendering with the LDR point cloud.*

4.2. Gradient-domain HDR Infilling

The output of the previous steps is a partially HDR point cloud P^*_{HDR} where some regions may lack valid HDR colors either because there are not visible in any rendering or because the 3D points receive HDR colors with low confidence, detected by a high patch error (SSD above a threshold $t_h = 1$ in all our experiments). Therefore, we propose to reconstruct the full P_{HDR} by propagating the HDR color from valid neighbors in the gradient domain [PGB03] (Alg. 1 line 8). Let Ω be the set of invalid HDR points, $\partial\Omega$ the set of the points on the boundary of the target regions (with valid HDR color), G the k-nearest neighbors graph of the points (in our experiments we always use $k = 16$), N_p the set of neighbors of the point p

in the graph G, $H(p)$ and $L(p)$ respectively the HDR and the LDR color of the point p. The final HDR color $H(p)$ for the points in Ω is computed by solving a Poisson problem:

$$\forall p \in \Omega \qquad |N_p| H(p) - \sum_{q \in N_p \cap \Omega} H(q) = \sum_{q \in N_p \cap \partial\Omega} H(q) + \sum_{q \in N_p} \nabla_{pq}, \tag{3}$$

where ∇_{pq} is the local gradient of the LDR colors between the points p and q defined as $\nabla_{pq} = (L(p) - L(q))$. Again, we can propagate either the luminance only or the entire RGB color. Fig. 3 shows a rendering of P^*_{HDR}, the invalid area and P_{HDR} after gradient-domain propagation.

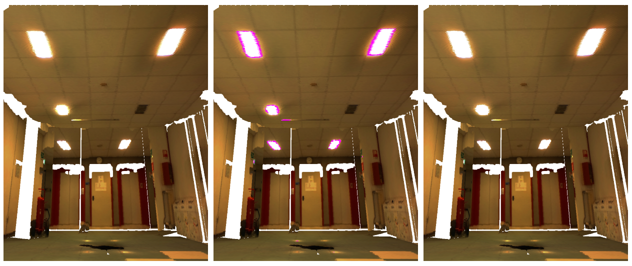


Figure 3: *After the projection of the HDR data to the point cloud (left image), some points can miss HDR values or receive invalid HDR color (magenta points in the center image) stemming from low PatchMatch confidence. The image on the right shows the final result, after the gradient-domain reconstruction.*

5. Real-Time PBGI

We can now use P_{HDR} to lit a synthetic object O using a new real-time PBGI algorithm. Since P_{HDR} already contains an approximation of the diffuse indirect lighting of the environment, we skip the expensive shading caching phase of the PBGI and run the algorithm directly from P_{HDR}. Our method runs in three steps:

1. the construction of the bounding sphere hierarchy H structuring P_{HDR}, at loading time (Sec. 5.1),
2. the adaptive tree cut search in H to fill every per-pixel micro-buffer (Sec. 5.2),
3. the convolution of the resulting micro-buffer by the pixel's BRDF, yielding the final pixel color value (Sec. 5.3).

To achieve real-time performance in the second step, our PBGI algorithm exploits intensively the GPU geometry shading stage and a new G-Buffer mipmapping mechanism. In this way, it alleviates the main PBGI bottleneck, the pixel shading, where each pixel requires filling a specific micro-buffer. In the following, after explaining our implementation of the classical PBGI algorithm (Sec. 5.1 and 5.2), we will present our contributions: MIP-PBGI and Cross-PBGI (Sec. 5.2.1).

5.1. Hierarchy Construction

We compute a Bounding Sphere Hierarchy (BSH) H of P_{HDR} from which per-receiver cuts will be gathered to fill micro-buffers at rendering time. Using the data of all the points in the node, each internal node A stores the average color $\mathbf{c}_A$, the average position $\mathbf{p}_A$ and the radius r_A of the bounding sphere, the bounding cone containing the normal vectors (stored as direction $\vec{\mathbf{n}}_A$ and half cone aperture α_A), and two indices for the children nodes. A leaf node contains the position, color, normal and radius. Alg. 2 shows the main steps for the tree construction.

Algorithm 2 BSH Construction

```
Input: HDR point cloud $P_{HDR}$
Output: Bounding Sphere Hierarchy $H$
1: $V \leftarrow$ GETMORTONCODES($P_{HDR}$)
2: $V^* \leftarrow$ PARALLELSORT($V$) (by [HKS09])
3: $H \leftarrow$ CREATELEAFNODES($V^*$)
4: $H \leftarrow$ CREATEINNERNODES($H$) (by [Kar12])
5: $H \leftarrow$ FILLINNERNODES($H$)
6: $H \leftarrow$ PRUNING($H$)
```

First, we voxelize the point cloud in its bounding box using a 64 bits Morton code and average the data for the cell containing more than one point. Second, the resulting nodes are Morton-sorted with the GPU method proposed by Ha et al. [HKS09], a parallel prefix sums [HSO07]. Third, we create the inner nodes using the algorithm of Karras et al. [Kar12]. Fourth, we propagate the leaves' attributes to the inner nodes: the color is propagated as a simple average of the children colors, while the position and the radius are computed with an approximate smallest sphere [FGK03], forcing the parent node always to contain its children. For the normal cone, we use the algorithm by Barequet et al. [BE05] to compute the minimum cone bounding a set of vectors. Finally, we prune the hierarchy by collapsing the sub-trees containing points with similar color and normal to speed up the visit of H at rendering time. In all our experiments, we use the same pruning thresholds: a color variance below $t_c = 0.01$ and an average dot product of the normals above $t_n = 0.98$. As shown in Fig 4, the approximation introduced by the pruning step is negligible in the final rendering.

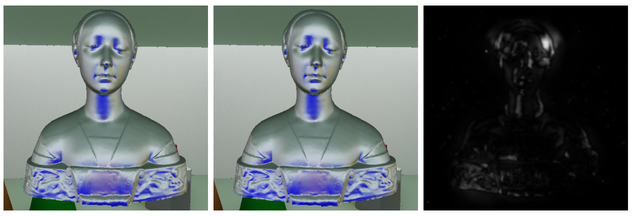


Figure 4: *Comparison of the renderings using the original BVH (left) and the clustered version (center). On the right the map with the probability to perceive differences between them computed with HDR-VDP-2.2.*

5.2. Micro-buffer Rasterization

At rendering time, we use H to gather, in real time, the incoming radiance from the surrounding scene over each rasterized point of O. To do so, we extend the concept of micro-buffers [REG*09] with a new mipmapping operator designed for the G-Buffer resulting from the rasterization of O. The G-buffer contains, for each pixel (i, j), the position $\mathbf{p}_{ij}$, the normal $\vec{\mathbf{n}}_{ij}$ and the material attributes $\mathbf{brdf}_{ij}$ of the rasterized object. Intuitively, the algorithm stores the incoming radiance in a set of small environment maps, the micro-buffers, one for each rasterized object point in the G-Buffer. The

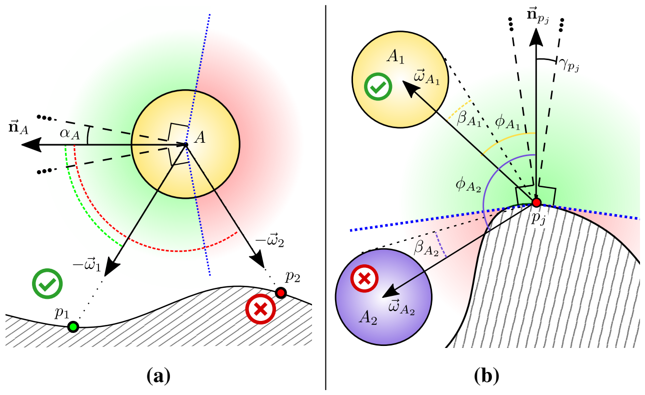


Figure 5: *Visibility conditions for a BSH node used in the MIP-PBGI. (a) A node is discarded if its normal cone is not facing the view direction of the pixel. For instance, A is discarded by p_2 but not by p_1. (b) The node is discarded when located outside the visibility cone of the pixel . Here, A_1 is inside the visibility cone of p_j, A_2 is not.*

incoming radiance is gathered from the nodes of H in the form of the optimal tree cut w.r.t. the resolution of the micro-buffer and the distance of the object point to the scene. The algorithm starts the traversal of H from the root node A and decides between (i) pursuing the visit on the children nodes, (ii) rasterizing the node in the micro-buffer or (iii) discarding it because it is not visible from the receiver. This decision depends on the solid angle $\Omega(A)$ subtended by the node A and the solid angle Ω subtended by a pixel (i, j) of the micro-buffer (from now on called m-pixel). More precisely, $\Omega(A)$ is the solid angle of the cone stemming from the 3D location $\mathbf{p}_{ij}$ of the pixel and the position $\mathbf{p}_A$ of the node:

$$\Omega(A) = 2\pi\left(1 - \sqrt{1-(r_A/dist(A))^2}\right), \tag{4}$$

where $dist(A) = \|\mathbf{p}_A - \mathbf{p}_{ij}\|$ and r_A is the node radius. The solid angle of each m-pixel is approximated as constant, subdividing the solid angle of the hemisphere by the resolution of the micro-buffer. If $\Omega(A) > \Omega$, we continue the visit in the children nodes, otherwise we rasterize the node in the micro-buffer using a paraboloid mapping [HS98] along the direction $\vec{\omega} = \mathbf{p}_A - \mathbf{p}_{ij}$. Each m-pixel stores the color of the rasterized node $\mathbf{c}_A$ and the value $dist(A)$ to perform the depth test. The node A is discarded if its normal cone is not facing the view direction of the considered point (see Fig. 5a)

$$-\vec{\omega}\cdot\vec{\mathbf{n}}_A < \cos(\alpha_A + \pi/2). \tag{5}$$

We refer to this version of the algorithm as "Classic PBGI".

5.2.1. MIP-PBGI and Cross-PBGI

To speed up the rendering time, we propose two new extensions of PBGI: MIP-PBGI, a multiresolution approach based on a new mipmapping operator for the G-Buffer; and Cross-PBGI (or X-PBGI) which merges the MIP and Classic approaches. MIP-PBGI parallelizes the traversal of H in a multi-resolution fashion following two observations. As pointed out by Hollander et al. [HREB11], the GPU Geometry Shader (GS) and the hardware Transform Feedback (TF) feature can be used to parallelize the visit of several branches of the BSH for the same pixel, instead of executing a depth-first visit in a single shader. The method works iteratively, with the input of the current GS iteration stored in the TF buffer by the previous GS iteration. The second observation by Wang et al. [WHB*13] points out that the BSH cuts gathered to fill the micro-buffers end up to be very similar for receivers which are close in position and normal. We further develop this idea by traversing H using a mipmapped version of the G-buffer proposing a new mipmapping operator for normal vectors (see Fig. 6). Given an internal level i, for each pixel, we store the minimum cone of directions (direction $\vec{n}$ and half aperture angle γ) containing the normals of all the pixels in the first level projected in that pixel at level i. The cone is computed using the algorithm by Barequet et al. [BE05].

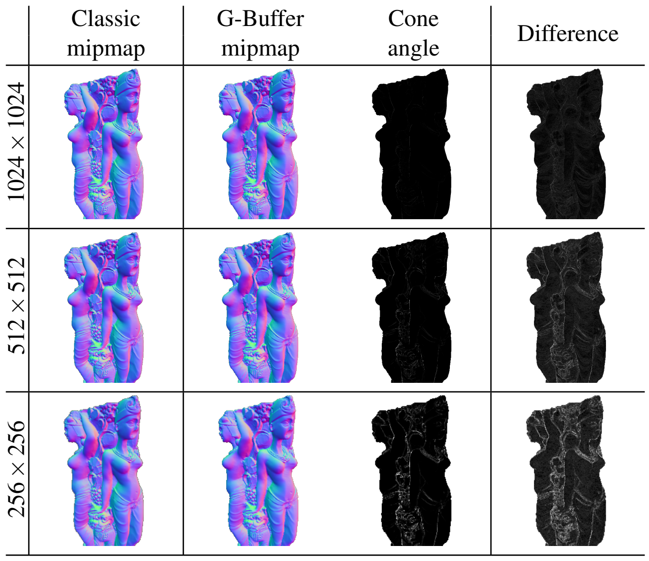


Figure 6: *Three mipmap levels of the normal data in the G-Buffer. The first column shows the classical mipmapping. The second and the third columns show the result of the new operator, respectively the cone direction and the aperture of the cone. The last column shows the differences between the two approaches computed as the angle between the vectors. The main variations are near the depth discontinuities.*

Our MIP-PBGI (Alg. 3) starts by generating a buffer of indices pairs which is sent to the GS, where each pair contains the index of the root node of H and the index of a non-empty pixel in the smaller mipmap level of the G-buffer. Let $[A_k, p_j]_i$ the input pair of the GS at level i of the mipmapped G-buffer and let $\Omega(p_j)$ be the solid angle of the normal cone stored in the pixel p_j computed as $\Omega(p_j) = 2\pi(1-\cos\gamma_j)$. If $i = 0$, i.e., first mipmap level of the G-buffer, the algorithm works as Classic PBGI. Otherwise, four alternative paths occur: **(i)** to continue the visit of the BSH in the children nodes; **(ii)** to continue the visit in the next mipmap level of the G-buffer; **(iii)** to discard the node because not visible from the receiver; **(iv)** to rasterize the current node in the micro-buffer of the pixel. In particular, if the solid angle of A_k is greater than

Algorithm 3 MIP-PBGI - Geometry Shader

```
Input:
    BSH $H$, mipmapped G-Buffer $G$, micro-buffer solid angle $\Omega$
    Pair $[A_k, p_j]_i$ with node of $H$ and pixel in the level $i$ of $G$
1:  $\Omega(A_k) \leftarrow 2\pi(1-\sqrt{1-(r_{A_k}/dist(A_k))^2})$        ▷ solid angle of $A_k$
2:  $\vec{\omega} \leftarrow \mathbf{p}_{A_k} - \mathbf{p}_{p_j}$
3:  if $i = 0$ then                                  ▷ In the first G-buffer mipmap level
4:      if $-\vec{\omega} \cdot \vec{\mathbf{n}}_{A_k} < \cos(\alpha_{A_k} + \pi/2)$ then            ▷ node not visible
5:          DISCARD($A_k$)
6:      else if $\neg$ISLEAF($A_k$) $\wedge$ $\Omega(A_k) > \Omega$ then
            ▷ continue in the $A_k$ children
7:          return $\{[\text{LEFT}(A_k), p_j]_0,\ [\text{RIGHT}(A_k), p_j]_0\}$
8:      else
9:          RASTERIZE($A_k$)
10: else                                             ▷ In the next G-buffer mipmap level
11:     $\Omega(p_j) \leftarrow 2\pi(1-\cos\gamma_j)$                   ▷ solid angle of $p_j$ normals cone
12:     $\beta_{A_k} \leftarrow \arcsin(r_{A_k}/dist(A_k))$
13:     $\phi_{A_k} \leftarrow \arccos(\vec{\omega} \cdot \vec{n}_{p_j})$
14:     if $(-\vec{\omega} \cdot \vec{\mathbf{n}}_{A_k} < \cos(\alpha_{A_k} + \pi/2)) \wedge (\phi_{A_k} - \beta_{A_k} > \gamma_{p_j} + \pi/2)$ then
            ▷ node not visible
15:         DISCARD($A_k$)
16:     else if $\neg$ISLEAF($A_k$) $\wedge$ $\Omega(A_k) > \Omega(p_j)$ then
            ▷ continue in the $A_k$ children
17:         return $\{[\text{LEFT}(A_k), p_j]_i,\ [\text{RIGHT}(A_k), p_j]_i\}$
18:     else if $\neg$ISLEAF($A_k$) $\wedge$ $\Omega(A_k) \leq \Omega(p_j)$ then
            ▷ continue in the next mipmap level
19:         return $\{[A_k, p_{j0}]_{i+1},\ [A_k, p_{j1}]_{i+1},\ [A_k, p_{j2}]_{i+1},\ [A_k, p_{j3}]_{i+1}\}$
20:     else
21:         RASTERIZE($A_k$)
```

the solid angle of the G-buffer pixel $\Omega(A_k) > \Omega(p_j)$, we continue the visit in the children of A_k, pushing a new pair in the TF buffer for each of them, with the same pixel. In the opposite case, i.e., $\Omega(A_k) \leq \Omega(p_j)$, our algorithm keeps the same BSH node and continues the visit by generating four new pairs in the TF buffer, one for each child of the pixel p_j in the next mipmap level $i+1$. With this procedure, the first part of the BSH visit is shared by neighboring pixels in the G-Buffer. Using normal cones in the G-buffer guarantees that all the nodes of H required to shade a pixel in the lowest mipmap level are gathered. During the visit, a node is discarded if its normal cone is not facing the normal of the pixel (Eq. 5) or if its view direction cone does not intersect the visibility cone of the pixel (see Fig. 5b):

$$\phi_{A_k} - \beta_{A_k} > \gamma_{p_j} + \pi/2, \tag{6}$$

where $\beta_{A_k} = \arcsin\left(\frac{r_{A_k}}{dist(A_k)}\right)$ is half the aperture of the view direction cone of A_k, defined by its distance from p_j and its radius r_A, and $\phi_{A_k} = \arccos\left(\vec{\omega} \cdot \vec{n}_{p_j}\right)$ is the angle between the pixel's normal cone direction $\vec{n}_{p_j}$ and the direction $\vec{\omega}$ that connects p_j at the center of A_k. Finally, when A_k is a leaf or the solid angle of the node is smaller than the solid angle of the m-pixel ($\Omega(A_k) \leq \Omega$), we reached the first mipmap level and we rasterize the node into the m-pixels covered by its visibility cone. During rasterization, we compute a simple ray-plane intersection using the plane described by the position and the direction of the normal cone of the node.

Our MIP-PBGI induces a significant speedup compared to Classic PBGI for large viewports with large micro-buffer, while for small ones it has worse performance (see Fig. 10 and the additional material). Moreover, the allocation of the output buffer used to store all primitives emitted by a GS iteration quickly becomes a bottleneck even for low-resolution viewports. To solve these problems, we introduce X-PBGI where MIP-PBGI is used only until $\Omega(A_k) > \Omega_{tr}$, with a new threshold $\Omega_{tr} > \Omega$. When this condition is false, the algorithm refines further the cut of the input pair using a depth-first search visit in a single shader, such as the classic approach. This hybrid version splits the visit of each pixel into several shaders, each one acting on a different sub-tree of H. The choice of an appropriate value for the threshold Ω_{tr} speeds-up the algorithm and reduces the memory usage. Experimentally, we noticed that the threshold $\Omega_{tr} = 32\Omega$ provides the best trade-off between memory and time in our scenes.

5.3. Micro-buffer Rendering

The output of the previous step is a texture filled with the micro-buffers data. Each micro-buffer can be used to compute the final pixel color by convolving it with the pixel's BRDF model $\mathbf{brdf}(\vec{\omega}_{in}, \vec{\omega}_{out})$, using a quadrature numerical integration on the discrete set of directions of the micro-buffer. The outcoming radiance of the pixels along the view direction $\vec{\omega}_{out}$ is equal to:

$$L_{out}(\vec{\omega}_{out}) = \sum_{i,j} \left[\mathbf{brdf}(\vec{\omega}_{ij}, \vec{\omega}_{out})\, C_{ij}\, \Omega_{i,j}\, (\vec{\mathbf{n}} \cdot \vec{\omega}_{ij})\right], \tag{7}$$

where C_{ij} is the color response stored in the m-pixel, $\vec{\omega}_{ij}$ is the associated input direction, and $\Omega_{i,j}$ is the solid angle of the m-pixel (i, j) according to the paraboloid mapping. Since the input cloud could have a non uniform point distribution or area with missing data (especially for clouds generated by multi-view stereo reconstruction), the micro-buffer can end up with many empty m-pixels in extreme scenarios. To limit this issue, preserving real-time performances, we optionally make the rasterization stopping criteria "over conservative", using a threshold that is four times the solid angle of the m-pixel ($\Omega(A_k) \leq 4\Omega$). This translates into a larger node coverage in the micro-buffer. Last, for the remaining empty m-pixels, we set them to the average color of the entire cloud.

5.4. Implementation Details

We implemented our rendering algorithm in OpenGL 4.4 using GS and TF capabilities and storing the BSH in a Shader Storage Buffer Object. Each node encompasses the indices of the children node (2 unsigned int), the position (3 floats), the color (3 floats), the radius (1 float) and the normal cone (encoded as an unsigned int in RGBA color format, where the RGB channels code the cone direction and the alpha the half aperture angle). Each node requires 40 bytes in total. We store the micro-buffers in a 2D texture with size equal to the viewport multiplied by the size of the micro-buffer. Each pixel in the texture is a half float RGBA color, where the alpha channel contains the depth values. The texture is read and written inside the shaders using the Image Load/Store functionalities. Classic PBGI visits the BSH for each pixel inside a single shader and does not need additional memory. On the contrary, MIP-PBGI and X-PBGI visit the BSH in an iterative and parallel way using the GS, together with the TF to store the output of the GS, which are pairs of indices (2 unsigned int). Each pair contains the index of a node of the BSH

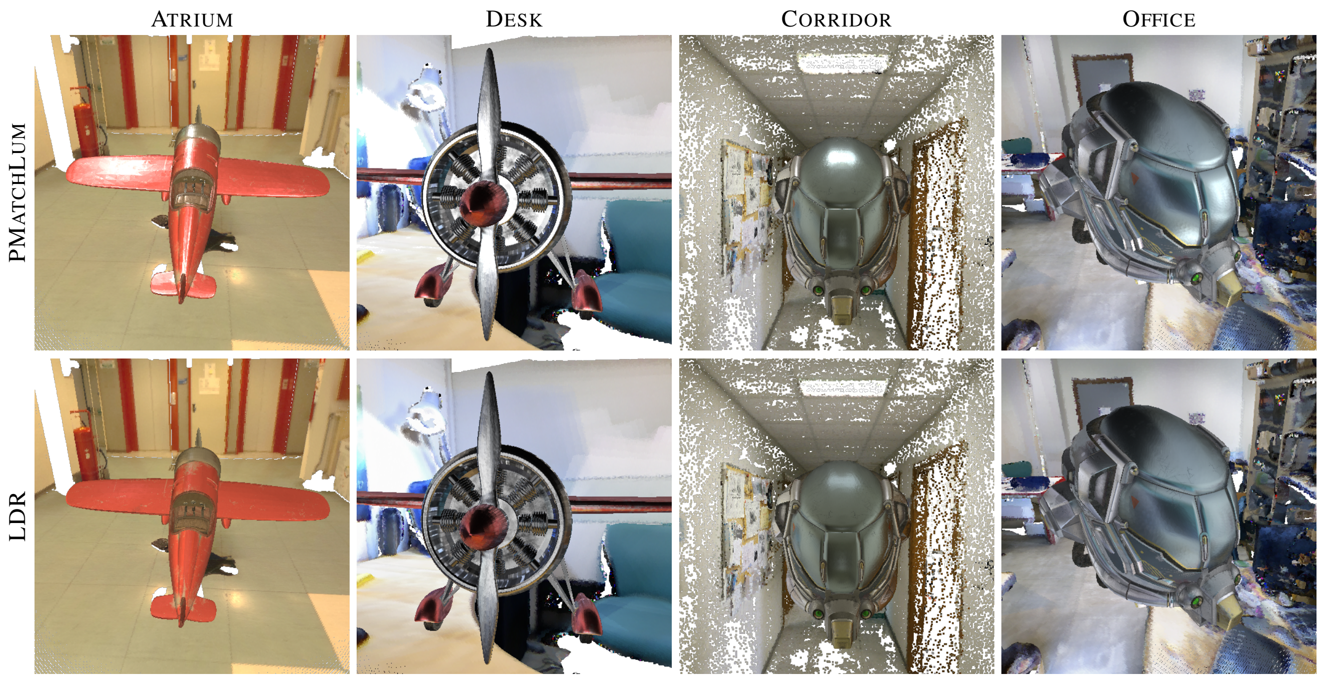


Figure 7: *Rendering results of the X-PBGI with the LDR point cloud (bottom) and with the cloud expanded with* PMATCHLUM *(top) in four acquired real scenes. The 3D models, an airplane and a helmet, are equipped with a spatially varying Disney Principled BRDF.*

and the Z-order index of the pixel in the G-Buffer. At each iteration, the algorithm takes in input the output pairs of the previous iteration and generates the new pairs by refining the visit in the BSH or in the next level of the G-Buffer. The algorithm stops when the output TF buffer is empty, which means that all the needed nodes were rasterized in the micro-buffer. Finally, X-PBGI requires an additional stream in the TF to store the pairs where the cut in the BSH must be refined using Classic PBGI.

6. Results

We tested the proposed pipeline with different datasets comparing the results of both the steps, HDR expansion (Sec. 6.2) and PBGI rendering (Sec. 6.3), with state-of-the-art methods. We performed our tests on a PC with an Intel i7-6700 CPU, 32GB of RAM, and an NVIDIA GeForce GTX1080 GPU with 8GB of video memory.

6.1. Dataset

For our tests, we employed a set of point clouds obtained with different technologies (Tab. 1). The tested clouds are:

- SPONZA, SIBENIK and FIREROOM, a Monte Carlo point sampling of three synthetic scenes after having baked the HDR diffuse color response using path tracing;
- ATRIUM, BUILDING and KITCHEN, three scenes reconstructed from a single HDR panoramic image with a user-assisted method [BCD*13] (with HDR color);
- CORRIDOR, a scene reconstructed with the multiview stereo software COLMAP [SF16] using the 1065 video frames of a walk in a corridor (with LDR color);
- OFFICE and DESK, scenes with several light sources acquired with a Kinect using BundleFusion [DNZ*17] (with LDR color);
- TOYROOM, a simple modeled room obtained with a Monte Carlo sampling of the 3D model after having baked the LDR diffuse color response using path tracing; this dataset is only used to evaluate the new PBGI algorithm.

Regarding point clouds with HDR ground truth color data, the LDR version was obtained by simply applying an exposure using Eq. 2. The rendered objects present different BRDF data such as pure diffuse, GGX [WMLT07], and Disney Principled [Bur12].

	Points	**Shots**	**HDR**	**BSH**	**N.Nodes**	**N.Cluster**
SPONZA	3.0M	3	180.5s	14.6s	6.0M	1.3M
SIBENIK	4.0M	2	185.3s	18.0s	8.0M	2.2M
FIREROOM	1.0M	1	56.1s	4.3s	2.0M	1.2M
ATRIUM	818k	1	50.4s	3.3s	1.6M	450K
BUILDING	871k	1	50.1s	4.3s	1.7M	125K
KITCHEN	1.25M	1	47.4s	5.5s	2.4M	440K
CORRIDOR	2.87M	3	171.3s	13.8s	5.8M	3.7M
OFFICE	3.54M	1	59.5s	20.0s	7.1M	2.6M
DESK	2.81M	1	57.0s	13.1s	5.6M	1.2M

Table 1: *For each scene, the table reports the number of points, the number of shots used for the HDR expansion (each shot is a cubemap), the time for the expansion, the time to build the BSH hierarchy, and the number of nodes of the BSH (before and after the clustering).*

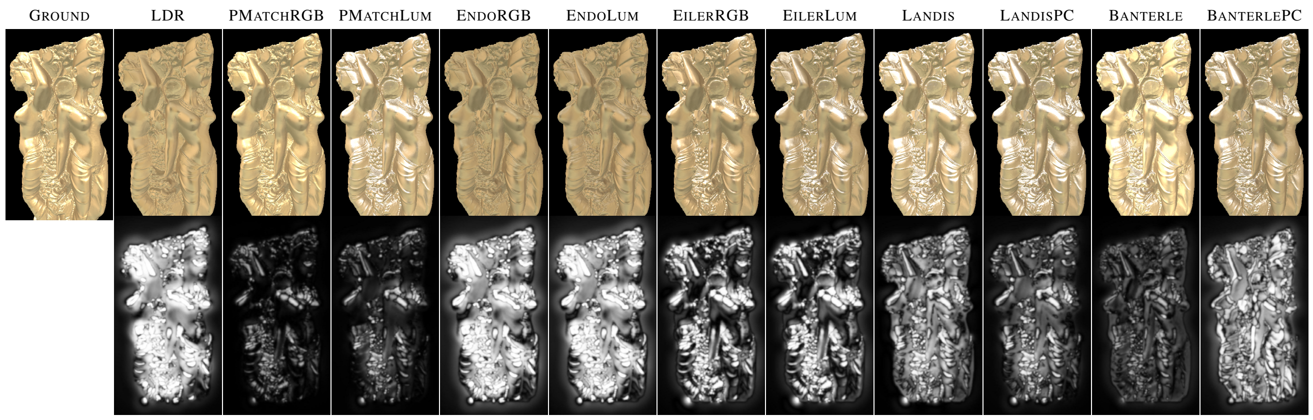


Figure 8: *Ground truth comparison of the renderings obtained with the different HDR expanded versions of the point cloud* ATRIUM. *For each rendering, the bottom image shows the probability map (computed with HDR-VDR-2.2) to detect differences from the ground truth rendering. More comparisons are available in the additional material.*

6.2. HDR Expansion Evaluation

To test the HDR expansion method presented in Sec. 4, we used the six clouds with ground truth HDR color (SPONZA, SIBENIK, FIREROOM, ATRIUM, BUILDING and KITCHEN). Starting from the LDR version of these clouds, we compare our method (PMATCH) with two alternative solutions. The first one is based on the substitution, inside the proposed method, of the PatchMatch algorithm with a state-of-the-art ITMOs, to do the expansion of the renderings before their backprojection on the cloud. We tested two image ITMOs, the global operator by Landis et al. [Lan02](LANDIS) and the expand-map based operator by Banterle et al. [BLDC06](BANTERLE), and two recent deep learning networks proposed by Eilertsen et al. [EKD*17](EILER) and by Endo et al. [EKM17](ENDO). For classical ITMOs, we focused on non-linear operators because they typically provide high-quality results for IBL [BDA*09]. Regarding deep learning methods, we used the pre-trained network provided by the authors. The second class of solutions is based on the extension of the classical image ITMOs (LANDIS-PC and BANTERLE-PC) to the structure of a point cloud, by applying the operators directly to the color of each point. For deep learning and PatchMatch methods, we tested either the possibility to transfer only the luminance (method with the suffix LUM) or the entire color (method with the suffix RGB) of the expanded renderings. For the ITMOs, we used only the luminance because these operators work only on it, and the maximum luminance value of I_{HDR} to determine the maximum value to reach after the expansion.

We evaluated all methods numerically in two ways: (i) measuring the distance of the expanded cloud from the ground truth HDR data using the Root Mean Square Error (RMS) on the per-point color; (ii) comparing the relighting effect obtained on a virtual object placed inside the scene and rendered with the proposed X-PBGI algorithm. In the last case, we measure the differences between the renderings with the expanded point cloud and the rendering with the ground truth HDR cloud using three different errors: the RMS error; the quality value of the HDR-VDP-2.2 [MN15]; the Structure Similarity (SSIM) [WBSS04] applied to the PU encoding [AMS08] of the luminance values. For each dataset, we selected two different viewpoints. In the rendering, we remove the background to avoid wrong perceptual impressions. Fig. 8 shows the rendering obtained using the point cloud ATRIUM expanded with the tested methods together with the relative probability maps of detecting differences from the ground truth. The probability maps are computed using HDR-VDP-2.2. Tab. 2 contains the numerical values of this comparison. The images and data of the other scenes are provided in the additional material. The PMATCH methods show the best results for almost all the tests. When they do not have the best score they are really close to the best ones. Looking at the RMS error on the point data only, the PMATCH methods are less competitive. This is due to the trivial computation of the RMS that does not take into account the distribution of the error in space, differently from the evaluation of the relighting effects. Also, the probability maps of HDR-VDP-2.2 visually show that the PMATCH methods produce the best results with a lower probability of detected differences from the ground truth.

6.3. PBGI Performance

We compared the performance of the PBGI algorithms introduced in Sec. 5.2.1 (MIP-PBGI and X-PBGI) with Classic PBGI [REG*09]. We used two test scenes (TOYROOM and SPONZA) with two different rendering viewpoints: the first one with a detail of the object that gets all the viewport; the second one where the entire object is visible in the viewport. Fig. 10 shows the charts with the time and the memory of the first view in the TOYROOM scene by changing the viewport and the micro-buffer size. The charts of the other views and scenes are provided as additional material. MIP-PBGI shows its benefits only for large viewports and large micro-buffers. Then, due to the intensive use of GPU memory to store the output of the GS in the TF, in some case MIP-PBGI is not able to complete. The X-PBGI approach is the fastest method, reaching a speed-up of 10x compared to Classic PBGI. Furthermore, its memory occupancy is always lower than MPI-PBGI. The

	RENDERING			POINT CLOUD
	RMS	**HDR-VDP**	**SSIM**	**RMS**
LDR	0.231	78.55	0.832	3.649
BANTERLE	0.114	**81.82**	0.938	2.383
LANDIS	0.175	80.03	0.932	3.349
BANTERLEPC	0.193	80.01	0.804	5.099
LANDISPC	0.150	80.52	0.948	2.834
EILERLUM	0.257	79.10	0.881	5.554
EILERRGB	0.240	79.11	0.881	5.554
ENDOLUM	0.245	78.87	0.824	3.549
ENDORGB	0.245	78.85	0.824	3.549
PMATCHLUM	0.109	81.33	**0.969**	**2.145**
PMATCHRGB	**0.068**	81.69	**0.969**	**2.145**

Table 2: *Numerical comparison of the renderings in Fig. 8 with the ground truth rendering. The error metrics are the RMS error, the quality of HDV-VDP-2.2, and the Structure Similarity (SSIM). The green text highlights the best results (for HDR-VDP and SSIM higher values are better). The last column shows the RMS error computed directly on the cloud color data. More comparisons are available in the additional material.*

additional material contains some renderings of the same scene with the three PBGI algorithms. Note that they differ only in how the BVH is traversed, but they fill the microbuffers in a very similar way. This is also confirmed by the high PSNR values, that are always above the 46dB.

To evaluate the approximation introduced by the proposed X-PBGI, we compared its relighting results with a path tracing (ground truth) and with Voxel Cone Tracing [CNS*11] (VCT). Fig. 12 shows the renderings of the same object in the TOY ROOM scene with the relative error maps. The additional material contains more data for this comparison. For the rendering, we use a viewport 512×512 pixels. The X-PBGI uses 32×32 micro-buffer. For the VCT we test two different versions: VCT16 uses 16 cones uniformly distributed on the visible hemisphere of each surface point with 40° aperture, plus a cone in the specular mirror direction with an aperture depending on the material roughness; VCT1024 uses 1024 cones distributed like the 32×32 micro-buffer with the paraboloid mapping used in our PBGI methods and an aperture depending on the solid angle of the pixel that the cone represents. Numerically, the error introduced by X-PBGI, measured as RMS and PSNR, is lower in all the tests. VCT16 obtains reasonable renderings in a short time but with a higher error than X-PBGI. Regarding VCT1024, the error is lower than VCT16 at the cost of higher rendering time, but the results remain inferior to X-PBGI. In particular, VCT introduces more approximation on the ornament on the bottom of the bust, where the contribution of the color objects in the scene is overestimated. Then, the main limitation of VCT methods is the right weight to give to each cone color contribution during the final convolution with the BRDF. In our test, we use the solid angle covered by each cone. The main advantages of the proposed PBGI are the real-time/interactive performance and the ability to capture more faithfully close and mid-range illumination effects, such as the color bleeding on the front of the bust.

6.4. Limitations

The main limitation of our PBGI evolution is the rendering of very shiny surfaces (i.e., low roughness) due to the approximation introduced by the size of the micro-buffers. For small micro-buffers, the algorithm may not capture the important reflection directions properly, and this may lead to higher errors, e.g., banding artifacts, as shown in Fig. 9. The use of larger micro-buffers mitigates these issues, but this solution increases the rendering time. An alternative solution could be the use of an importance sampling strategy during the construction of a micro-buffer as proposed by Ritschel et al. [REG*09] or adaptive micro-buffers [WMB15]. Our results lack of any kind of self occlusion effect. We can overcome this issue using a state-of-the-art method, such as Screen Space Soft Shadow [GCS10] or Precomputed Radiance Transfer [SKS02] to precompute the visibility function using spherical harmonics. Some of the algorithms used in the proposed framework are affected by inherent limitations that directly influence our results. For example, the PatchMatch algorithm [BSGF10] gives poor results if I_{HDR} and the set of LDR renderings S are incoherent. Using the quality value returned by the PatchMatch mitigates this limitation but cannot completely overcome the problem. For this reason, it is important to choose carefully both I_{HDR} and the set S.

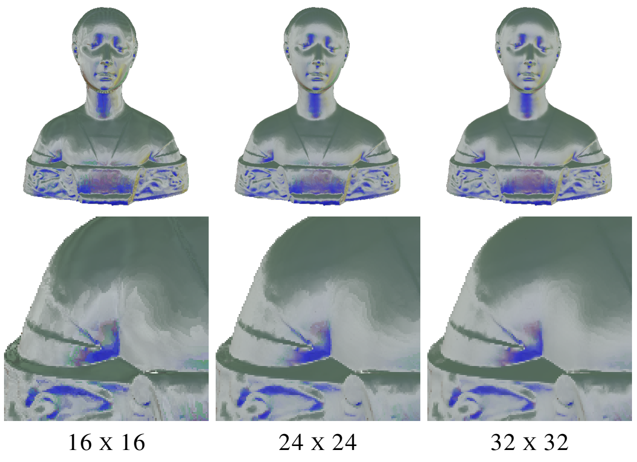


Figure 9: *Comparison of the renderings of the same object with roughness* 0.1 *(GGX) by changing the micro-buffer size. In this case, the high specularity of the material leads to banding artifacts for small micro-buffer.*

6.5. Applications

Fig. 7 shows the benefit of our framework in a look-dev scenario. The figure presents the rendering of an airplane and a helmet with a spatially varying Disney Principled BRDF placed in different real scenes using the LDR point cloud and the HDR cloud expanded with the method PMATCHLUM. The contribution of the HDR expansion increases the realism of the rendering especially on the shiny regions of the object. Fig. 11 shows the results in an AR application scenario where the rendering of the object in the scene CORRIDOR is combined with a video frame used for the acquisition of the point cloud. The renderings with the HDR cloud show

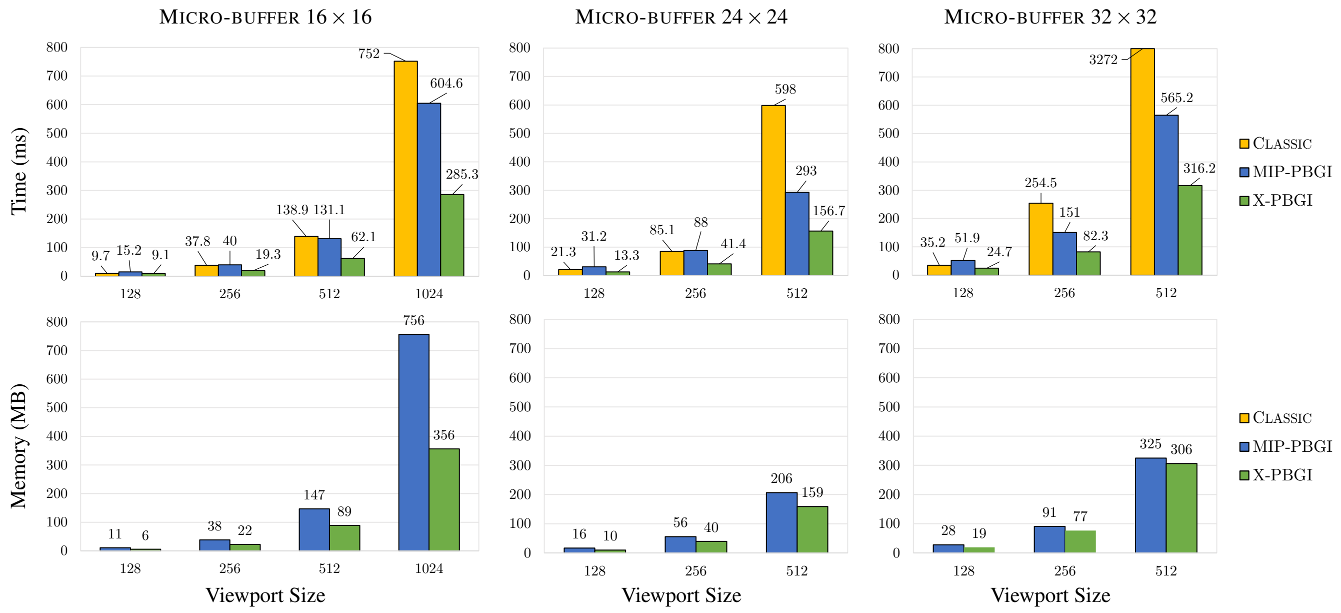


Figure 10: *Performance comparison (time and memory occupancy) of the three PBGI algorithms – Classic [REG*09], MIP-PBGI and X-PBGI (Sec. 5.2) – varying the viewport and the micro-buffer size. These tests are performed on the point clouds* TOYROOM *with a* 512×512 *viewport where the object gets all the viewport. For the memory occupancy, we report the additional memory required to store the output primitives of the Geometry Shader in the Transform Feedback buffers. More comparisons are available in the additional material.*

more coherent lighting with the environment. The additional material contains a video with small animations of these objects inside the HDR expanded clouds compared with the result obtained with the LDR cloud.

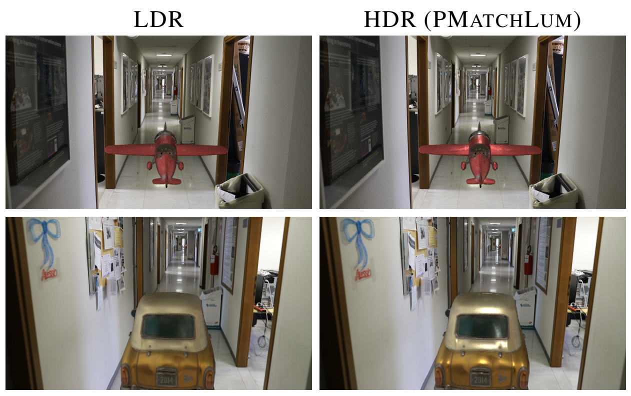


Figure 11: *AR compositing of the rendering of two objects in the scene* CORRIDOR *with a video frame used for the computation of the point cloud.*

7. Conclusion

We have presented a framework to exploit the data encoded into a captured 3D point cloud (radiance, spatial, and visibility information) of a real-world scene to perform real-time relighting of a virtual object inside the scene. Our framework is based on two main contributions: an HDR expansion method to create an HDR 3D point cloud from an LDR one using a single HDR picture – to our knowledge, this is the first 3D ITMO; a real-time relighting engine based on a new PBGI variant equipped with a G-Buffer mipmapping mechanism. We have experimented our framework with several datasets with and without HDR ground truth color data, showing good numerical performance and convincing visual results. Furthermore, we have shown the results obtained in an AR scenario with animated virtual objects. Although our two contributions are presented as an integrated framework, both can be of interest independently for other applications such as the simple visualization of enhanced cloud on new HDR devices or the previsualization of visual special effects.

In future works, we plan to investigate the preprocessing steps to increase the quality of the input LDR point cloud in terms of density and point distribution, accounting for the data source. This step should help to populate regions with no information, improving the quality of the rendering. Finally, on-the-fly HDR expansion could help using our framework in a dynamic AR scenario, where the real environment may change over time.

Acknowledgements

This work was partially supported by the EU H2020 Programme EMOTIVE: EMOTIve Virtual cultural Experiences through personalized storytelling (grant no. 727188), the French National Research Agengy (ANR) under grant ANR 16-LCV2-0009-01 ALLEGORI, and by BPI France, under grant PAPAYA.

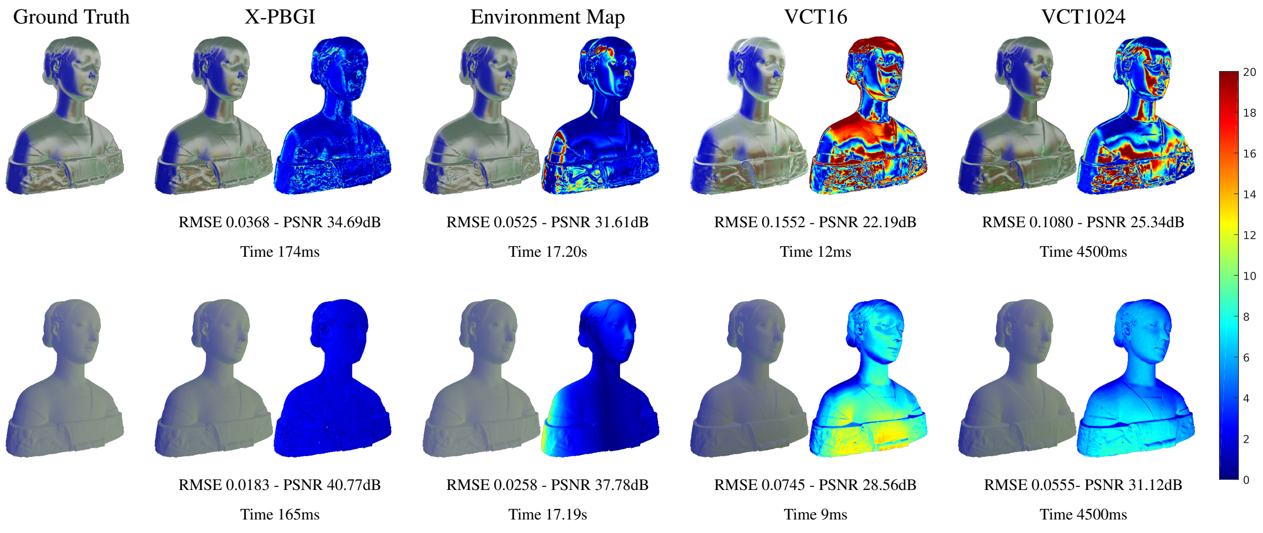

Ground Truth
X-PBGI
Environment Map
VCT16
VCT1024
RMSE 0.0368 - PSNR 34.69dB
Time 174ms
RMSE 0.0525 - PSNR 31.61dB
Time 17.20s
RMSE 0.1552 - PSNR 22.19dB
Time 12ms
RMSE 0.1080 - PSNR 25.34dB
Time 4500ms
RMSE 0.0183 - PSNR 40.77dB
Time 165ms
RMSE 0.0258 - PSNR 37.78dB
Time 17.19s
RMSE 0.0745 - PSNR 28.56dB
Time 9ms
RMSE 0.0555- PSNR 31.12dB
Time 4500ms
20
18
16
14
12
10
8
6
4
2
0

Figure 12: *Comparison of X-PBGI rendering with the ground truth obtained with a path tracing, the classical environment map algorithm, and two different versions of the Voxel Cone Tracing (VCT) algorithm (16 cones plus a specular cone for VCT16 and 1024 cones for VCT1024). Each rendering shows the relative error map from the ground truth. (Top) GGX BRDF with roughness* 0.1. *(Bottom) Pure diffuse BRDF. More comparisons are available in the additional material.*

References

[AFR*07] Akyüz A. O., Fleming R., Riecke B. E., Reinhard E., Bülthoff H. H.: Do HDR Displays Support LDR Content? A Psychophysical Evaluation. *ACM Trans. Graph. 26*, 3 (2007), 38. 2

[AMS08] Aydin T. O., Mantiuk R., Seidel H.-P.: Extending quality metrics to full luminance range images. In *Human Vision and Electronic Imaging XIII* (2008), vol. 6806, International Society for Optics and Photonics, p. 68060B. 9

[BADC17] Banterle F., Artusi A., Debattista K., Chalmers A.: *Advanced High Dynamic Range Imaging: Theory and Practice (2nd Edition).* AK Peters (CRC Press), July 2017. 2

[BB12] Buchholz B., Boubekeur T.: Quantized point-based global illumination. *Comp. Graph. Forum (Proc. EGSR 2012) 31*, 4 (2012), 1399–1405. 3

[BCD*13] Banterle F., Callieri M., Dellepiane M., Corsini M., Pellacini F., Scopigno R.: Envydepth: An interface for recovering local natural illumination from environment maps. *Comput. Graph. Forum 32*, 7 (2013), 411–420. 3, 8

[BDA*09] Banterle F., Debattista K., Artusi A., Pattanaik S. N., Myszkowski K., Ledda P., Chalmers A.: High dynamic range imaging and low dynamic range expansion for generating HDR content. *Comput. Graph. Forum 28*, 8 (2009), 2343–2367. 9

[BE05] Barequet G., Elber G.: Optimal bounding cones of vectors in three dimensions. *Inf. Process. Lett. 93*, 2 (2005), 83–89. 5, 6

[BLDC06] Banterle F., Ledda P., Debattista K., Chalmers A.: Inverse tone mapping. In *GRAPHITE* (2006), ACM, pp. 349–356. 2, 9

[BSGF10] Barnes C., Shechtman E., Goldman D. B., Finkelstein A.: The generalized PatchMatch correspondence algorithm. In *ECCV* (2010). 4, 10

[Bun05] Bunnell M.: Dynamic ambient occlusion and indirect lighting. *GPU Gems 2* (2005), 223–233. 2

[Bur12] Burley B.: Physically based shading at disney. In *ACM SIGGRAPH 2012 Courses:Practical physically-based shading in film and game production* (2012), ACM, p. 26. 8

[BZS*07] Bhat P., Zitnick C. L., Snavely N., Agarwala A., Agrawala M., Cohen M., Curless B., Kang S. B.: Using photographs to enhance videos of a static scene. In *EGSR* (2007), Eurographics Association, pp. 327–338. 2

[Chr08] Christensen P.: Point-based approximate color bleeding. *Pixar Technical Notes 2*, 5 (2008), 6. 2

[CNS*11] Crassin C., Neyret F., Sainz M., Green S., Eisemann E.: Interactive indirect illumination using voxel cone tracing. *Computer Graphics Forum 30*, 7 (2011), 1921–1930. 10

[DBGBR*14] 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. *Comp. Graph. Forum 33*, 2 (2014), 459–468. 4

[Deb98] Debevec P.: Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH* (1998), ACM Press, pp. 189–198. 2, 3

[DMHS08] Didyk P., Mantiuk R., Hein M., Seidel H.-P.: Enhancement of bright video features for HDR displays. *Comp. Graph. Forum 27*, 4 (2008), 1265–1274. 2

[DNZ*17] Dai A., Niessner M., Zollhöfer M., Izadi S., Theobalt C.: Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Trans. Graph. 36*, 3 (2017), 24:1–24:18. 2, 3, 8

[EKD*17] Eilertsen G., Kronander J., Denes G., Mantiuk R., Unger J.: Hdr image reconstruction from a single exposure using deep cnns. *ACM Trans. Graph. 36*, 6 (2017). 2, 9

[EKM17] Endo Y., Kanamori Y., Mitani J.: Deep reverse tone mapping. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 177:1–177:10. 2, 9

[FGK03] Fischer K., Gärtner B., Kutz M.: Fast smallest-enclosing-ball computation in high dimensions. In *European Symposium on Algorithms* (2003), Springer, pp. 630–641. 5

[GCS10] Gumbau J., Chover M., Sbert M.: *Screen Space Soft Shadows.* GPU Pro. A.K. Peters/CRC Press, 2010, pp. 477–491. 10

[GSY*17] Gardner M.-A., Sunkavalli K., Yumer E., Shen X., GAMBARETTO E., GAGNÉ C., LALONDE J.-F.: Learning to predict indoor illumination from a single image. *ACM Trans. Graph. 36*, 6 (2017), 176:1–176:14. 3

[HKS09] HA L., KRÜGER J., SILVA C. T.: Fast four-way parallel radix sorting on gpus. *Comp. Graph. Forum 28*, 8 (2009), 2368–2378. 5

[HREB11] HOLLANDER M., RITSCHEL T., EISEMANN E., BOUBEKEUR T.: Manylods: Parallel many-view level-of-detail selection for real-time global illumination. *Comp. Graph. Forum 30*, 4 (2011), 1233–1240. 3, 6

[HS98] HEIDRICH W., SEIDEL H.-P.: View-independent Environment Maps. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware* (1998), Spencer S. N., (Ed.), The Eurographics Association. 6

[HSH*17] HOLD-GEOFFROY Y., SUNKAVALLI K., HADAP S., GAMBARETTO E., LALONDE J.: Deep outdoor illumination estimation. In *CVPR* (2017), pp. 2373–2382. 3

[HSO07] HARRIS M., SENGUPTA S., OWENS J. D.: Parallel prefix sum (scan) with cuda. *GPU gems 3*, 39 (2007), 851–876. 5

[Kar12] KARRAS T.: Maximizing parallelism in the construction of bvhs, octrees, and k-d trees. In *HPG Conference* (2012), pp. 33–37. 5

[KBG*15] KRONANDER J., BANTERLE F., GARDNER A., MIANDJI E., UNGER J.: Photorealistic rendering of mixed reality scenes. *Comp. Graph. Forum 34*, 2 (2015), 643–665. 3

[KBLE19] KOL T. R., BAUSZAT P., LEE S., EISEMANN E.: Megaviews: Scalable many-view rendering with concurrent scene-view hierarchy traversal. *Computer Graphics Forum 38*, 1 (2019), 235–247. 3

[KHFH11] KARSCH K., HEDAU V., FORSYTH D., HOIEM D.: Rendering synthetic objects into legacy photographs. *ACM Trans. Graph. 30*, 6 (2011), 157:1–157:12. 3

[KO14] KOVALESKI R. P., OLIVEIRA M. M.: High-quality reverse tone mapping for a wide range of exposures. In *SIBGRAPI* (August 2014), IEEE Computer Society, pp. 49–56. 2

[KSH*14] KARSCH K., SUNKAVALLI K., HADAP S., CARR N., JIN H., FONTE R., SITTIG M., FORSYTH D.: Automatic scene inference for 3d object compositing. *ACM Trans. Graph. 33*, 3 (2014), 32:1–32:15. 3

[Lan02] LANDIS H.: Production-ready global illumination. In *SIGGRAPH Course Notes* 16 (2002), pp. 87–101. 2, 9

[MBRHD18] MARNERIDES D., BASHFORD-ROGERS T., HATCHETT J., DEBATTISTA K.: Expandnet: A deep convolutional neural network for high dynamic range expansion from low dynamic range content. *Comp. Graph. Forum 37*, 2 (2018), 37–49. 2

[MKR07] MERTENS T., KAUTZ J., REETH F. V.: Exposure fusion. In *Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (2007), IEEE Computer Society, pp. 382–390. 4

[MN15] MANISH NARWARIA RAFAL MANTIUK M. P. D. S. P. L. C.: Hdr-vdp-2.2: a calibrated method for objective quality prediction of high-dynamic range and standard images. *Journal of Electronic Imaging 24* (2015), 24 – 24 – 3. 9

[MSG15] MASIA B., SERRANO A., GUTIERREZ D.: Dynamic range expansion based on image statistics. *Multimedia Tools and Applications* (2015), 1–18. 2

[Pel10] PELLACINI F.: envylight: An interface for editing natural illumination. *ACM Trans. Graph. 29*, 4 (July 2010), 34:1–34:8. 3

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. *ACM Trans. Graph. 22*, 3 (July 2003), 313–318. 4

[RDGK12] RITSCHEL T., DACHSBACHER C., GROSCH T., KAUTZ J.: The state of the art in interactive global illumination. *Comp. Graph. Forum 31*, 1 (2012), 160–188. 2

[REG*09] RITSCHEL T., ENGELHARDT T., GROSCH T., SEIDEL H.-P., KAUTZ J., DACHSBACHER C.: Micro-rendering for scalable, parallel final gathering. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 132:1–132:8. 3, 5, 9, 10, 11

[RPAC17] RHEE T., PETIKAM L., ALLEN B., CHALMERS A.: MR360: mixed reality rendering for 360° panoramic videos. *IEEE Trans. Vis. Comput. Graph. 23*, 4 (2017), 1379–1388. 2, 3

[RTS*07] REMPEL A. G., TRENTACOSTE M., SEETZEN H., YOUNG H. D., HEIDRICH W., WHITEHEAD L., WARD G.: Ldr2hdr: On-the-fly reverse tone mapping of legacy video and photographs. *ACM Trans. Graph. 26*, 3 (2007), 39. 2, 4

[RWP*10] REINHARD E., WARD G., PATTANAIK S. N., DEBEVEC P. E., HEIDRICH W.: *High Dynamic Range Imaging - Acquisition, Display, and Image-Based Lighting (2. ed.)*. Academic Press, 2010. 2

[SF16] SCHÃŰNBERGER J. L., FRAHM J. M.: Structure-from-motion revisited. In *CVPR* (June 2016), pp. 4104–4113. 2, 3, 8

[SKS02] SLOAN P.-P., KAUTZ J., SNYDER J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph. 21*, 3 (July 2002), 527–536. 10

[SL17] SILVENNOINEN A., LEHTINEN J.: Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Trans. Graph. 36*, 6 (2017), 230:1–230:13. 2

[Tab12] TABELLION E.: Point-based global illumination directional importance mapping. In *ACM SIGGRAPH Talk* (2012). 3

[UKL*13] UNGER J., KRONANDER J., LARSSON P., GUSTAVSON S., LÖW J., YNNERMAN A.: Spatially varying image based lighting using hdr-video. *Computers and Graphics 37*, 7 (2013). 3

[WBSS04] WANG Z., BOVIK A. C., SHEIKH H. R., SIMONCELLI E. P.: Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing 13*, 4 (2004), 600–612. 9

[WFA*05] WALTER B., FERNANDEZ S., ARBREE A., BALA K., DONIKIAN M., GREENBERG D. P.: Lightcuts: A scalable approach to illumination. *ACM Trans. Graph. 24*, 3 (2005), 1098–1107. 2

[WHB*13] WANG B., HUANG J., BUCHHOLZ B., MENG X., BOUBEKEUR T.: Factorized point based global illumination. *Comp. Graph. Forum 32*, 4 (2013), 117–123. 3, 6

[WMB15] WANG B., MENG X., BOUBEKEUR T.: Wavelet point-based global illumination. *Comp. Graph. Forum 34*, 4 (2015), 143–153. 3, 10

[WMLT07] WALTER B., MARSCHNER S. R., LI H., TORRANCE K. E.: Microfacet models for refraction through rough surfaces. In *Eurographics Conference on Rendering Techniques* (2007), pp. 195–206. 8

[WSG*16] WHELAN T., SALAS-MORENO R. F., GLOCKER B., DAVISON A. J., LEUTENEGGER S.: Elasticfusion: Real-time dense SLAM and light source estimation. *I. J. Robotics Res. 35*, 14 (2016), 1697–1716. 2, 3

[WWZ*07] WANG L., WEI L.-Y., ZHOU K., GUO B., SHUM H.-Y.: High dynamic range image hallucination. In *SIGGRAPH '07: ACM SIGGRAPH 2007 Sketches* (2007), ACM, p. 72. 2

[XLL*18] XING G., LIU Y., LING H., GRANIER X., ZHANG Y.: Automatic spatially varying illumination recovery of indoor scenes based on a single rgb-d image. *IEEE TVCG* (2018), 1. 3

[ZCC16] ZHANG E., COHEN M. F., CURLESS B.: Emptying, refurnishing, and relighting indoor spaces. *ACM Trans. Graph. 35*, 6 (2016), 174:1–174:14. 3

[ZL17] ZHANG J., LALONDE J.: Learning high dynamic range from outdoor panoramas. In *IEEE ICCV* (2017), pp. 4529–4538. 2