High Dynamic Range Expansion of Point Clouds for Real-Time Relighting

Manuele Sabbadin[1,3], Gianpaolo Palma[1], Francesco Banterle[1], Tamy Boubekeur[2], Paolo Cignoni[1]

[1] Visual Computing Lab - ISTI - CNR, Pisa, Italy
[2] LTCI, Telecom ParisTech, Paris-Saclay University, Paris, France
[3] Department of Computer Science - University of Pisa, Italy

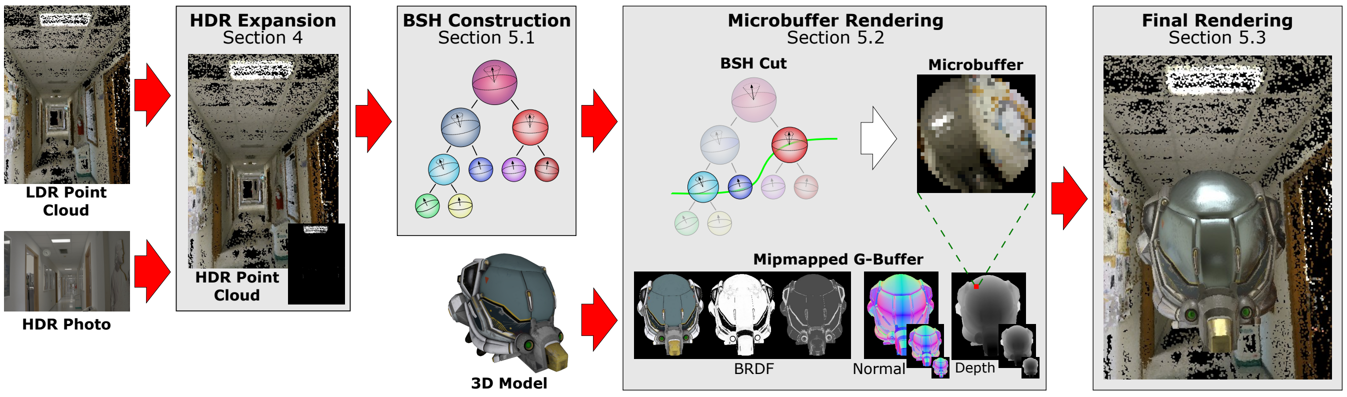


Figure 1: *The relighting framework can be split into two independent parts: the preprocessing of the LDR point cloud (HDR expansion and BSH construction) and the real-time rendering of a 3D model inside the acquire environment. In the preprocessing step, the algorithm expands the dynamic range of the point cloud using an input HDR photo (the images shows the difference map between the computed HDR cloud and 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 microbuffers of each pixel of the viewport. Finally, the collected microbuffers are combined with the BDRF data of the 3D model to obtain the final rendering.*

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 genuine light transport hidden in the recorded per-sample color response to relight virtual objects in visual effects (VFX) look-dev or augmented reality 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 of all, 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 may only cover part of it. We perform efficiently this expansion 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 which are not covered by the renderings or with low quality dynamic range by solving a Poisson's 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 with respect to the perfect ground truth. We also report experiments on real captured data, covering a range of capture technologies, from active scanning to multiview stereo reconstruction.

CCS Concepts

• ***Computing methodologies*** $\rightarrow$ ***Computer graphics; Rendering;*** *Point-based models;*

1. Introduction

Nowadays several solutions are available for acquiring large 3D point clouds of objects or environments in a fast and completely automatic way. These solutions are based on low budget hardware such as an RGB-Depth camera for the active acquisition systems (Microsoft Kinect, Intel RealSense, StructureSensor, etc.) or a simple camera for multiview reconstruction algorithms. These devices allow the acquisition of the geometric information of the scene (3D position and surface normal) and of the RGB color of each sampled point. Typically, this data is used for different purposes; from the simple visualization to a more complex processing to compute dense triangular model or to infer more semantic information. Even if the color data is acquired with a Low Dynamic Range (LDR) sensor, getting a limited range of the output radiance of each sampled point, together with the geometric information, it can be used in the context of the relighting of a virtual object inside the acquired scene in a realistic way. Typical application scenarios are the Augmented Reality (AR) and Visual Effects (VFX) where the goal is the integration of a virtual object into a movie or a picture with a realistic illumination as similar as possible to the scene. In particular, in the relighting context, the colored point cloud can be used to compute the mid-range lighting effects, while the classical environment map rendering and screen-space reflection can approximate respectively the long range/direct illumination effects and the local inter-reflection.

This paper presents an innovative framework to use the radiance, spatial and visibility information of a captured colored point cloud of a real-world scene to compute the contribution of the lighting on a virtual object inserted into such scene. Since we obtain LDR colored point cloud from the most common 3D acquisition solutions, we propose a preprocessing step to extend the dynamic range of the point cloud and obtain a High Dynamic Range (HDR) color for each point sample. Then, this HDR point cloud is used to compute the lighting over the virtual object using an innovative real-time Point-Based Global Illumination (PBGI) method. The main contributions are:

- the development of an innovative framework to extend the dynamic range of the color in a point cloud using a single HDR photograph covering a small part of the environment and without any calibration data (i.e., without estimation of the intrinsic and extrinsic camera parameters). The algorithm is based on a first step to transfer the dynamic range from the HDR photo to the point cloud using a set of renderings testing different procedures (i.e., inverse tone mapping, deep learning, and Patch-Match), following by the solving of the Poisson's equation to propagate the HDR data on the regions not covered by the renderings or with a low quality extracted HDR values;
- an innovative PBGI method that exploits the geometry shader capabilities and the mipmapping of the G-buffer of the virtual object to speed-up the computation and to obtain real-time performances.

We tested both the steps with a synthetic ground truth and several real captured scenes, evaluating numerically the differences among the proposed HDR expansion algorithms and among the rendering with the LDR and the HDR point cloud. We report an exhaustive performance analysis of the proposed PBGI algorithm, and we show the rendering results obtained in an AR application scenario.

2. Related Work

In this section, we present the state-of-the-art for the three main topics related to our proposed method: the inverse/reverse tone mapping (Section 2.1), the PBGI rendering algorithm (Section 2.2), and the virtual relighting of a virtual object for AR applications (Section 2.3).

2.1. Inverse Tone Mapping

Inverse/reverse tone mapping operators (ITMOs/RTMOs) [RWP*10, BADC17] expand the dynamic range of LDR images/videos in order to obtain content that can be employed in HDR applications such as HDR visualization (i.e., showing this content on an HDR television) 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 which areas need to be expanded and the relative intensity. Finally, a user-based operator expands the dynamic range of LDR content with user inputs such as classifying areas to be expanded [DMHS08], or cloning details from well-exposed areas into under-exposed and over-exposed areas [WWZ*07].

Recently, deep learning has been employed with success for expanding LDR content [EKD*17, EKM17, 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 difficult to acquire and there are very few publicly available datasets.

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 i.e., unprojected final image pixel, (i) a so-called *microbuffer* i.e., 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 microbuffer, solving for visibility using the depth component. The final receiver response then sums the convolution of its microbuffer with its BRDF and 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 improvements demonstrate high scalability for parallel architectures [REG*09, HREB11] and out-of-core execution [Tab12], robustness to compression [BB12] and factorization [WHB*13] as well as the ability, to a certain extent, to cope with non-diffuse effects [WMB15]. Our key observation is that a scanned 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 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 and has sparked a vast literature from the seminal work on Image-Based Lighting (IBL) by Debevec [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 [GSY*17, HSH*17] or video stream (e.g., AR). For a more 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 M360 by Rhee et al. [RPAC17]. This system provides interactive mixed reality using LDR 360 panoramic videos for lighting virtual objects 360 videos. They employed inverse tone mapping to enhance LDR panoramic videos and achieved convincing results. The use of panoramic video limits the rendering of the lighting effects due to directional lights. Zhang et al. [ZCC16] presented a system for capturing indoor 3D scenes with color data using RGBD scanners tailored for emptying/refurnishing indoors. Starting from an RGBD scan, it produces a scene model of the empty room, its lighting emitters, and its materials. The main differences with our approach are the use of a triangular mesh instead of a point cloud and the computation of the color data by integration of a set of LDR photos with auto-exposure that do not return the entire dynamic range of the scene. Finally, Whelan et al. [WSG*16] introduced ElasticFusion; a real-time dense visual simultaneous localization and mapping (SLAM) algorithm. Such SLAM has a module for estimating light sources exploiting scene geometry and specular regions. This estimation makes possible the insertion and rendering with a coherent lighting of synthetic objects in augmented reality applications.

3. Algorithm Overview

Our goal is to relight virtual objects in real time inside a real scene using a scanned colored point cloud of the environment. The inputs of our method are (i) P_{LDR}, a scanned colored point cloud of an 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, we start with a preprocessing step to expand the dynamic range of P_{LDR} using I_{HDR} (Section 4), especially in the overexposed regions. Then, at rendering time, the resulting HDR point cloud P_{HDR} is used for relighting the 3D model of the virtual object O in real time using a new PBGI algorithm (Section 5). Fig. 1 shows an overview of the proposed algorithm.

4. HDR Expansion of Point Clouds

In order to expand the dynamic range of the point cloud P_{LDR}, we assume that the HDR photo I_{HDR} provides a representative distribution of the dynamic range of the 3D scene, with no constraint on being aligned in any form to the point cloud. This makes the capture process easy because I_{HDR} can be taken completely independently from P_{LDR}. In order to boost P_{LDR} dynamic range from I_{HDR}, we propose a new 2-step framework. In the first step (Section 4.1), the algorithm extends the dynamic range of a set of LDR renderings of the point clouds, testing different approaches such as the classical inverse tone mapping operators for images ([Lan02, BLDC06]), the recent deep learning architectures ([EKD*17, EKM17]), and randomized match methods based on PatchMatch ([BSFG09, BSGF10]). The output of this step is a set of renderings with HDR color values. Each rendering is then projected on the original point cloud using the camera parameters used during their acquisition. The second step (Section 4.2) uses a Poisson strategy [PGB03] to fill the HDR values on samples that are not covered by the renderings or have with a low-quality HDR mapping. Since all the proposed algorithms work in a linear color space, we remove the gamma correction from the input images and point cloud.

4.1. HDR Expansion

The general idea is to estimate the HDR data in the overexposed regions of a set of unlighted renderings S_i of the point cloud obtained using the cloud LDR color. The renderings have to be taken from reasonable positions so that they cover as much as possible of the overexposed areas of the scene. Even if there exist solutions that can be used to automatically extract these rendering positions (for example [DBGBR*14]), we decide to use a manual approach, allowing a user to navigate the scene and acquire renderings from the best positions; i.e., to see the most important overexposed areas. A cubemap is captured from each of these positions and used in the following steps. For the rendering we use a simple point splatting algorithm with viewport 1024×1024. For each captured rendering the algorithm stores the camera intrinsic and extrinsic parameters for the following reprojection of the image data to the point cloud.

In the first step, the algorithm finds an LDR version of I_{HDR} with an exposure as close as possible to the one used for the acquisition of the color in P_{LDR}. To find this exposure the algorithm proceeds in the three steps. Firstly, it retrieves a set of fixed exposure images using the exposure fusion operator [MKR07]. Secondly, it looks for the image with the closer luminance distribution to the point cloud. In particular, it selects the image with the closest histogram to the histogram of the entire LDR point cloud, using as measure the Wasserstein distance [CEN07]. The histograms are computed only on the luminance channel in the CIE XYZ color space using 128 bins. Thirdly, the selected exposure is refined with a binary search in the range defined by the nearest exposures of the tested images. The algorithm proceeds to look for a better exposure until the error stops to decrease or up to a limit of 50 iterations. Every time the algorithm picks an exposure value α, it applies the exposure to the HDR image and clamps the result in $[0,1]$:

$$I_{LDR} = \text{clamp}(\alpha \cdot I_{HDR}, 0.0, 1.0) \quad (1)$$

The final output is an LDR version I_{LDR} of the input HDR image with a color range that is the closest to the input point cloud. This increases the quality of the correspondences extracted by the PatchMatch step.

At this point, the algorithm finds the correspondences between each rendering S_i of the point cloud and the HDR image I_{HDR}, using its LDR version I_{LDR}, without using any calibration data about the camera parameters of the HDR photograph. We tested three different strategies for the HDR expansion of the renderings (the comparison is in Section 6.2). The first two strategies are based on a trivial application of two well-known methods for HDR expansion in images: the inverse tone mapping operators and the deep learning architecture for HDR reconstruction for a single photo. The most interesting results are instead obtained with the third strategy based on the use of the generalized PatchMatch algorithm [BSGF10]. We use the generalized PatchMatch [BSGF10] between S_i and I_{LDR} because it is more robust at the rotation and scale. In particular, for each pixel of the rendering S_i, it computes the offset in the image I_{LDR} to retrieve the most similar patch, defining the nearest neighbor field (NNF), and the value of the distance between the patches according to an error metric defining a quality field. In our implementation, the Sum of Squared Differences (SSD) error was employed. Exploiting the computed NNF of each rendering, the algorithm transfers the HDR data from I_{HDR} to the renderings S_i (I_{HDR} and I_{LDR} have the same size so the computed offset in the LDR image can be used to retrieve the data from the HDR image).

The last step of all strategies (i.e., inverse tone mapping, deep learning, and PatchMatch) is the projection of the HDR data from HDR expanded renderings to the point cloud, using the camera parameters previously stored. In this step, we limit the expansion to the overexposed areas of the point cloud. We consider overexposed all the points with an input LDR color luminance above a threshold $t_l = 0.9$. For all the other points, the algorithm assigns the original well exposed LDR color as HDR color. During the projection, the algorithm applies the exposure value α computed in the first step for the HDR image to align the black value of the HDR image to the black value LDR point cloud, avoiding edge-step transaction around the luminance threshold t_l. If some points are visible in more than one rendering, the algorithm computes the average of the projected color samples. In the case of the inverse tone mapping operator or of the deep learning approach, it computes the simple average. On the contrary, in the case of the PatchMatch, the algorithm computes the weighted average with the quality field of the matching giving more weight to the samples with lower SSD.

When we project the HDR information to the point cloud, we can transfer either the luminance alone (i.e., just extending the LDR range) or the entire color from I_{HDR}. In the latter case, we can obtain a color for the overexposed areas (usually light sources) closer to the original one instead of boosted white color. This is useful when the hardware for the acquisition of the point cloud introduced some strange chromatic shift to the captured colors. The choice of what information to transfer depends on the specific application. In Figure 2, we show the differences between the two methods.

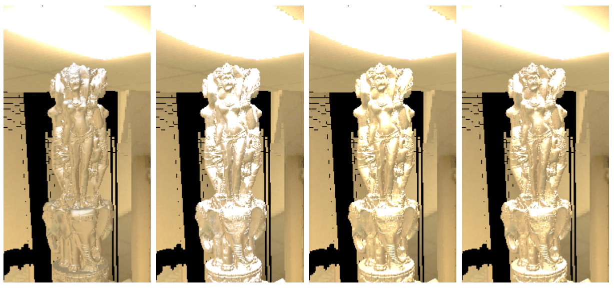

(a) *LDR* **(b)** *HDR lum* **(c)** *HDR color* **(d)** *Ground Truth*

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. Poisson Color Editing

The output of the HDR expansion framework is a point cloud P_{HDR} with HDR color values. However, some critical regions without valid HDR colors may be created. This is due to two different reasons: the points in such regions are not visible in any renderings; or in the case of the PatchMatch approach, the points receive HDR colors very different from the original ones, resulting in a high patch SSD. In the first case, the point does not have projected HDR color, while in the second case the HDR color is discarded because the SSD is above a threshold t_h. In all our experiments, we always use the same threshold $t_h = 1$. In both cases, to solve the problem the general idea is to propagate the HDR color from the neighbor valid points using a Poisson strategy [PGB03].

Let Ω be the set of the points in the target regions (without a valid HDR color), $\partial\Omega$ 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 the following system of linear equations:

$$\forall p \in \Omega \qquad |N_p| H(p) - \sum_{q \in N_p \cap \Omega} H(q) = \sum_{q \in N_p \cap \partial\Omega} H(q) + \sum_{q \in N_p} v_{pq}, \quad (2)$$

where v_{pq} is the local gradient of the LDR colors of the points p and q defined as:

$$v_{pq} = (L(p) - L(q)). \quad (3)$$

In this step, we can also propagate only the luminance or the entire RGB color. In the first case, the algorithm solves a single system of linear equation, while in the second it solves a system for each color channel independently. Figure 3 shows a rendering of the point cloud with the HDR color after the projection without Poisson editing, the area interested in the Poisson Editing and the results obtained with the Poisson editing.

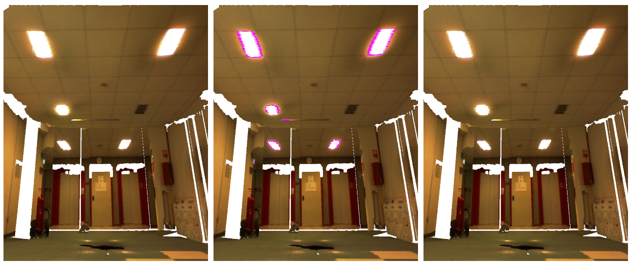


Figure 3: *After the projection of the six shots to the point cloud (left image), some points can receive invalid HDR color (cyan point in the center image) due to missing information in the shots or a bad quality value given by the PatchMatch algorithm. The image on the right shows the final result, after the Poisson editing.*

5. Real-Time PBGI

After the computation of the HDR point cloud P_{HDR}, the goal is to render a synthetic object O inside the cloud using a new PBGI algorithm that runs in real time. Since P_{HDR} already contains an approximation of the diffuse indirect lighting of the environment, the proposed method avoids the expensive phase for the computation of the shading information of each point. The proposed algorithm exploits intensively the GPU geometry shading stage and a new G-Buffer mipmapping method to speed-up the rendering alleviating the pixel shading, the PBGI bottleneck as each pixel requires filling a specific microbuffer. Our method runs in three steps:

1. the GPU construction of H, a bounding sphere hierarchy structuring P_{HDR}, at loading time (Section 5.1),
2. the cut search in H for each pixel of the G-buffer rasterized from O, to fill its specific microbuffer (Section 5.2),
3. the computation of the final rendering of each pixel, applying any given BRDF to the relative microbuffer (Section 5.3).

The second step represents a tremendous amount of computation, which usually prevents real-time performances at high framerate. We tackle this issue using our new G-Buffer mipmapping strategy.

5.1. Hierarchy Construction

In this step, we compute a Bounding Sphere Hierarchy (BSH) H of the HDR point cloud for which per-receiver light cuts will be gathered to fill micro-buffers at rendering time. Each node A in the hierarchy H stores the average color of the points inside the node $\mathbf{c}_A$, the average position $\mathbf{p}_A$, the radius r_A of the bounding sphere that contains all the points in the node, the bounding cone containing the normal vectors of the points inside the node, represented as a direction $\vec{\mathbf{n}}_A$ and half cone aperture α_A and the two indices pointing to the left and right child of the node. A leaf node contains the color, the normal and the position of a single point. The tree construction is organized in four steps: the creation and sorting of the leaf nodes, the creation of inner nodes, the propagation of the leaves' attributes to its inner nodes and tree vectorization.

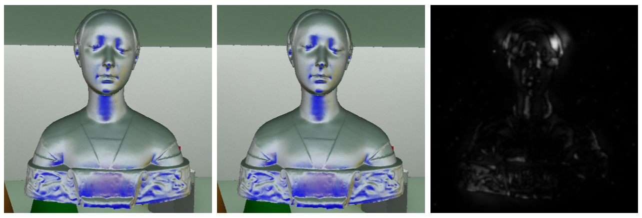


Figure 4: *Comparison of the renderings using the original BVH (left) and the clustered version (center). On the right the map with the probability to perceive differences between them computed with HDR-VDP-2.2.*

In order to preserve the spatial locality, we organize the leaf nodes using Morton codes, interleaved bits of the points 3D coordinates. We voxelize the point cloud in its bounding box using a 64 bits Morton code and obtain a leaf node for each non-empty cell of the voxelization. When the cell contains more points, the leaf node averages the data of all the points in the cell. The resulting nodes are Morton-sorted using a compute shader following the approach presented by Ha et al. [HKS09], using a parallel prefix sums [HSO07]. For the creation of inner nodes, we use the fast algorithm proposed by Karras et al. [Kar12]. Once the structure of H is initialized, we propagate the leaves' attributes to inner nodes. While color is propagated as a simple average of the children colors, the position and the radius is computed with the approximate smallest sphere by Fischer et al. [FGK03], making the approximation conservative by forcing the parent node to always contain its children. For the computation of the normal cone, for the first level above the leaves, we use the algorithm in [BE05] to compute the minimum cone for a set of vectors. For the other levels, we find the minimum cone of normals that contains the cone of the children. We reduce this computation to the previous problem by finding four vectors on the border of the cones of the children and computing the minimum cone surrounding them. These four vectors are obtained by rotating the cone direction of each child around the cross product between this direction and the difference vector between the cones. The rotation angle is the cone aperture. The obtained hierarchy is pruned by removing all the sub-tree where the color variance of all the points in the sub-tree is below t_c and the normal variation is above t_n. The idea is to reduce the number of nodes by clustering all the point in the cloud with very similar color and normal in order to speed up the visit of the hierarchy during the rendering. The variances are computed as the average vector distance from the node color and average dot product from the node normal. In all our experiments, we use the pruning thresholds $t_c = 0.01$ and $t_n = 0.98$. The visual effects on the final rendering are negligible as is shown in Figure 4. Finally, we linearize the pruned hierarchy H into a vector using a depth-first visit.

5.2. Micro-buffer Rendering

We use H at rendering time to gather in real time the incoming radiance from the surrounding scene over each point of O. To do so, we

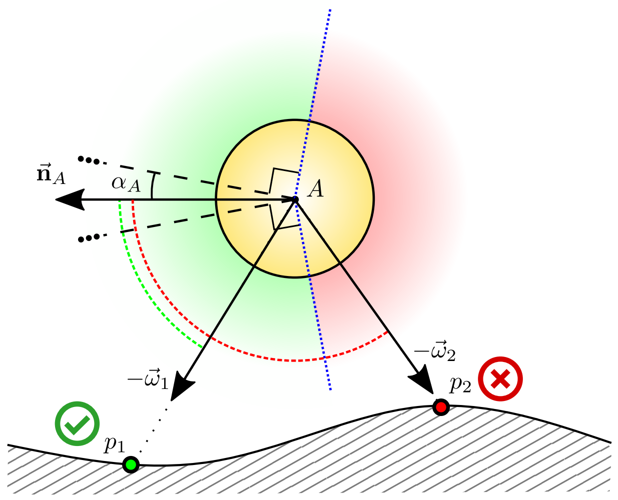


Figure 5: *If a node faces a direction opposite with respect to the pixel we are considering, it is discarded with all its sub-tree. The cone of visibility is given by the cone of normals of the node (direction $\vec{\mathbf{n}}_A$ with aperture $2\alpha_A$) plus $\frac{\pi}{2}$ radians. For example, the point p_1 is inside the cone of visibility of A, while the point p_2 is not.*

extend the micro-buffer rendering approach proposed by Ritschel et al. [REG*09] with a new mipmapping operator designed for the G-Buffer resulting from the rasterization of O. This 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 points. The general idea of our algorithm is to store the incoming radiance in a set of small environment map, the micro-buffers, one for each rasterized object point in the G-Buffer, where the incoming radiance is gathered from the nodes of the BSH on the optimal cut of the hierarchy computed in function of the resolution of the micro-buffer and the distance of the object point from the scene. The algorithm starts a depth-first search on the BSH from the root node A and it decides if to continue the visit on the children nodes, to rasterize the node in the microbuffer or to discard it because it is not visible from the point. The decision on the rasterization of the node in the micro-buffer is based 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. The solid angle of a node $\Omega(A)$ is the solid angle of the cone generated by the pixel and the node sphere:

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

where $dist(A) = |\mathbf{p}_A - \mathbf{p}_{ij}|$ is the distance between the center of the sphere of the node A and the 3D position of current pixel $\mathbf{p}_{ij}$, and r_A is the radius of the sphere. The solid angle of each micropixel has been approximated as constant:

$$\Omega = \frac{2\pi}{m_{width}\, m_{height}}, \tag{5}$$

where m_{width} and m_{height} are the width and the height of the microbuffer. If $\Omega(A) > \Omega$, we continue the visit in the children nodes, otherwise we rasterize the node in the micro-buffer using a standard hemispherical projection $\Phi(x,y) = (x, y, \sqrt{1-x^2-y^2})$, that maps a pixel (x,y) (with $x, y \in [0,1]$) of the micro-buffer to the direction $\vec{\omega} = \mathbf{p}_A - \mathbf{p}_{ij}$ that connects the center of the sphere of the node to the surface point. In the microbuffer, we store the color of the node $\mathbf{c}_A$ and the value $dist(A)$ to do the depth test during the rasterization of the node. The node and its sub-tree are discarded during this visit if the cone of normals of the node is not facing the direction of the considered point (see figure 5), that is if $-\vec{\omega} \cdot \vec{\mathbf{n}}_A < \cos(\alpha_A + \frac{\pi}{2})$, where α_A is the normal cone angle of the node A. In the rest of the paper, we call this version of the rendering algorithm as Classic PBGI.

The following sections present two versions of the proposed PGBI algorithm: the Mipmapped PBGI (Section 5.2.1), a multiresolution approach based on a new mipmapping operator for the G-Buffer; the Hybrid PBGI (Section 5.2.2) that merges the Mipmapped version with the classical approach for the micro-buffer rendering, based on a visit of the BSH done in the same shader for single pixel.

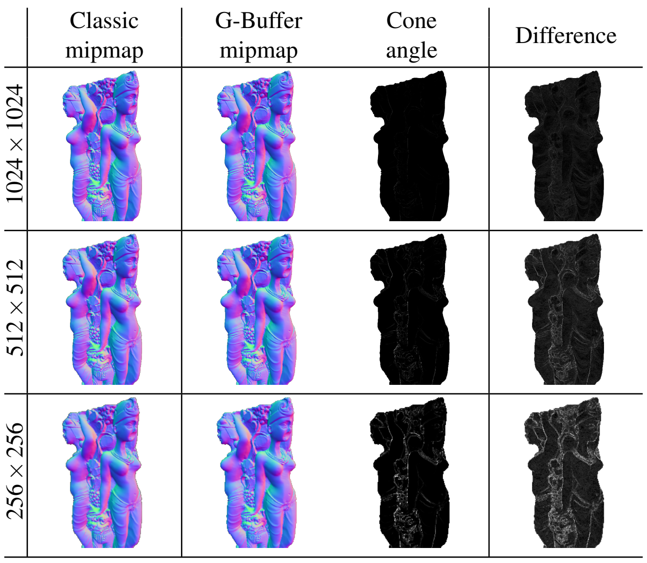


Figure 6: *The first three mipmap levels of the normal data in the G-Buffer. The first column shows the classical mipmapping while the second and the third columns show the result of our algorithm, respectively the cone direction and the aperture of the cone. To improve the visualization, the last column shows the sine of the angle between the normal computed traditionally and with our method. The main differences are near the depth discontinuities.*

5.2.1. Mipmapped PBGI

We present a new PBGI algorithm that parallelizes the BSH traversal in a multi-resolution way using two observations from state-of-the-art methods. As pointed out by Hollander et al. [HREB11], the GPU geometry shader and the OpenGL Transform Feedback can be used to parallelize the visit of several branches of the BSH for the same pixel, instead of executing a sequential depth-first visit in a single shader. The method works in an iterative way where the input of the next step is stored in the Transform Feedback buffer by the previous step. Given in input at the geometry shader a pair made

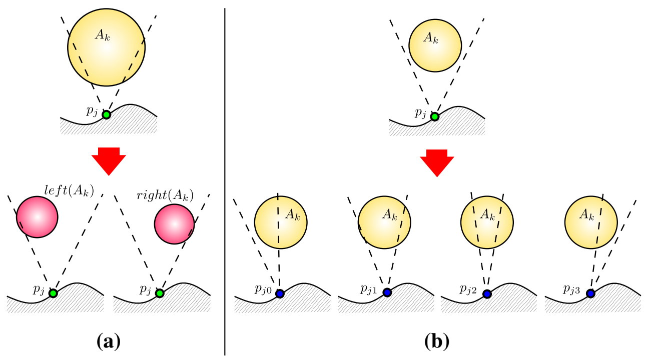


Figure 7: *The image shows two of the three cases that the geometry shader can manage during the processing of a pair $[A_k, p_j]$ composed by a node of the BSH and a non-empty pixel of the G-buffer. (a) When the node is bigger than the cone of normals of the pixel, the algorithm proceeds the visit generating two new pairs for the children of A_k. (b) When the node is smaller than the cone of normals of the pixel and the pixel is not in the first mipmap level, the algorithm generates four new pairs for the next iteration. The third option happens when the node is smaller than the cone of normals of the actual pixel p_j and the pixel is in the last mipmap level or when the node is a leaf. In this case, the algorithm rasterizes the node into the microbuffer.*

by the current node of the hierarchy and the pixel of the viewport, the shader decides or to rasterize the node in the microbuffer or to continue the visit with another iteration by sending in the Transform Feedback buffer two new pairs with the children of the nodes. The second observation in Wang et al. [WHB*13] points out that, using small microbuffers, the cuts on the BSH to render the microbuffers result to be very similar for receivers near in position and normal. The new contribution of our algorithm is the traversal of the hierarchy using a mipmapped version of the G-buffer of the rasterization of the 3D object. To be robust in the traversal of the BSH, we implement a new operator for the mipmapping of the normal vector (see figure 6). For a level i after the first one, it stores a cone of directions, stored as direction $\vec{n}$ and cone angle γ. This contains all the normals of the pixels of the first level that are projected into one pixel of the level i. For the computation of these cones of directions, we use the same procedure for the estimation of the normal cone of internal nodes of the BSH described in Section 5.1, based on [BE05].

The mipmapped algorithm starts by generating a buffer of pairs, where each pair $[A, p]$ is composed by two integer indices, the index of the root node of the BSH and the index of a not-empty pixel in the smaller mipmapping level of the G-buffer. The index of the pixel is generated with the classical 2D z-filling curve. When a pair is sent to the geometry shader, there are three alternative processing paths: to continue the visit of the BSH in the children nodes (Figure 7a); to continue the visit in the next level of mipmapping in the G-buffer (Figure 7b); to rasterize the current node of the pair in the micro-buffer of the pixel. Take for example the pair $[A_k, p_j]_i$ at the level i of the mipmapped G-buffer. Let $\Omega(p_j)$ be the solid angle associated with the cone of normal vectors stored in the pixel p_j computed with the following formula $\Omega(p_j) = 2\pi(1-\cos\alpha_j)$. If pixel p_j is not in the first mipmapping level ($i > 0$) and the solid angle of the BSH node is greater than the solid angle of the pixel $\Omega(A_k) > \Omega(p_j)$, or if p_j is in the first level $i = 0$ and the solid angle of the BSH node is greater than the solid angle of the pixel of the micro-buffer $\Omega(A_k) > \Omega$, then the algorithm must visit the two children nodes of A_k. In both cases, it outputs two new pairs with the children of A_k for the next iteration of the geometry shader ($[left(A_k), p_j]_i$ and $[right(A_k), p_j]_i$). Otherwise, if p_j is not in the first mipmapping level and the solid angle of the BSH node is smaller of the solid angle of the pixel $\Omega(A_k) \leq \Omega(p_j)$, the algorithm keeps the same BSH node but it continues the visit from the next mipmapping level, generating four pairs, one for each pixel in which p_j is split: $[A_k, p_{j0}]_{i+1}$, $[A_k, p_{j1}]_{i+1}$, $[A_k, p_{j2}]_{i+1}$ and $[A_k, p_{j3}]_{i+1}$. In this way, the first part of the BSH traversal is shared by the four pixels p_{j0}, p_{j1}, p_{j2} and p_{j3} and it is refined when the solid angle of the node becomes smaller. In all cases, before storing the new pairs in the Transform Feedback buffer for the input of the next iteration, the algorithm checks the visibility of the nodes (see figure 8) taking into account that now we have a cone of view directions for the BSH node, determined by its distance from the object surface and its sphere radius, with direction $\vec{\omega}_{A_k}$ and angle:

$$\alpha_{A_k} = \arcsin\left(\frac{r_A}{dist(A_k)}\right), \tag{6}$$

and the cone of normals in the pixel of the G-Buffer with direction $\vec{\omega}_{p_j}$ and angle γ_{p_j}. Let's indicate as:

$$\phi_{A_k} = \arccos\left(\vec{\omega}_{p_j} \cdot \vec{\omega}_{A_k}\right), \tag{7}$$

the angle between the normal at p_j and the vector $\vec{\omega}_{A_k}$, which connects p_j to the center of the sphere of the node A_k. The algorithm discards the node if its cone of normals does not intersect the cone of visibility of the point p_j, that is:

$$\phi_{A_k} - \alpha_{A_k} > \gamma_{p_j} + \frac{\pi}{2}. \tag{8}$$

Finally, when the pixel p_j is in the first mipmapping level and the solid angle of the node is smaller the solid angle of the pixel ($\Omega(A_k) \leq \Omega$) or when the node A_k is a leaf of the hierarchy, the shader rasterizes the BSH node into the pixel of the micro-buffer covered by the visibility cone of the node. For the rasterization, we use a simple ray-sphere intersection test. We approximate the depth with the depth of the intersection of the ray with the plane described by the center and the direction of the normal cone of the node.

5.2.2. Hybrid PBGI

Even if the previous approach can benefit of a great speedup compared to the Classic PBGI especially for high-resolution viewports, it requires a lot of memory (Table 3). The allocation of the output buffer, used to store all primitives emitted by a geometry shader iteration, becomes a bottleneck even for low-resolution viewports. To solve this problem, we implement a hybrid algorithm where the mipmapping PBGI version is used only until the solid angle of the node is above a threshold Ω_{tr} greater than the pixel solid angle Ω, and the pixel of the viewport is not in the first mipmap level. When one of these conditions are false, for each remaining pair in the Transform Feedback, the algorithm refines further the cut using a

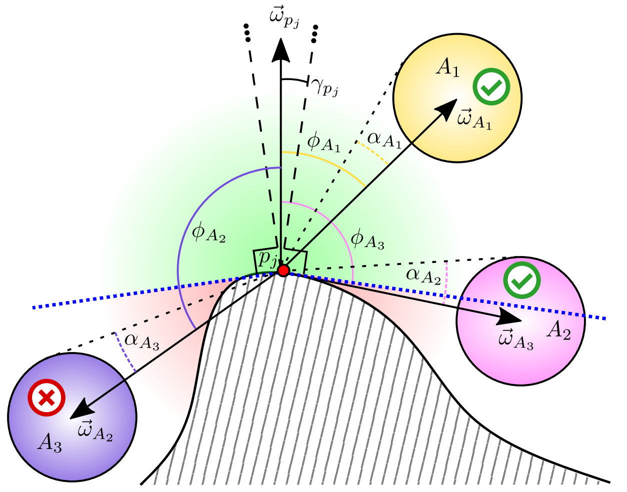


Figure 8: *The visibility of a node w.r.t. a pixel of the G-buffer depends on the cone of normals of the pixel and the position and size of the node. In particular, the node is inside the visibility cone of the pixel if the angle $\phi_{A_k} - \alpha_{A_k}$ is less than $\gamma_{p_j} + \frac{\pi}{2}$. In figure, the nodes A_1 and A_2 are inside the visibility cone of p_j, A_3 is not.*

depth-first search visit in a single shader execution, as in the classic approach. Differently from the Classic PBGI, where the visit of the BSH of each pixel is done in a single shader, with this hybrid version the visit of each pixel is split into several shaders each one on a different sub-tree of the BSH. The choice of an appropriate value for the threshold Ω_{tr} speeds-up the algorithm and reduces its memory usage. Experimentally, we noticed that the threshold $\Omega_{tr} = 32\Omega$ guarantees the best speed-up for the algorithm.

5.3. Micro-buffer Final Rendering

All previous approaches lead to a texture filled with the micro-buffers information. Each micro-buffer can be used to compute the final color of the pixel by convolution of its information with BRDF data $f(\vec{\omega}_{in}, \vec{\omega}_{out})$ of the rendered object. Since a micro-buffer represent a discrete set of directions (typical size is 24×24 pixels) over the visible hemisphere, we can weigh the incoming radiance of each pixel p_{ij} by the subtended solid angle Ω_{ij} of the saved nodes in the pixel. The outcoming radiance of the pixels along the view direction $\vec{\omega}_{out}$ is equal to:

$$L_{out}(\vec{\omega}_{out}) = \sum_{i,j} \left[f(\vec{\omega}_{ij}, \vec{\omega}_{out}) C_{i,j} \Omega_{ij} (\vec{\mathbf{n}} \cdot \vec{\omega}_{ij}) \right], \tag{9}$$

where $C_i j$ is the stored color in the pixel of the microbuffer, and $\vec{\omega}_{ij}$ is the associated input direction. Using this formula, we obtain darker rendering w.r.t. the reference created with a Montecarlo path tracing. The reason is due to the stopping condition that permits to rasterize a node of the BSH in the micro-buffer when the solid angle subtended by the node is smaller of the solid angle of one pixel in the micro-buffer. This condition could create two problems: the node does not cover all the solid angle of the pixel Ω, so the sum of the all the solid angles of the rasterized nodes is less than the portion of visible hemisphere of the pixel; some pixels of the microbuffer do not receive any information creating holes in microbuffer because any node has a view direction that can be rasterized in the pixel according to the hemispherical projection. To solve the first problem, we simply substitute the solid angle of the node $\Omega_{i,j}$ with the solid angle of the pixel Ω. To alleviate the second problem, preserving the real-time performance, we change the stopping criteria for the rasterization using a bigger threshold that is four times the solid angle of the pixel ($\Omega(A_k) \leq 4\Omega$). In this way, the sphere of the node can be rasterized in all pixels covered by the visibility cone of the node using a simple ray-sphere intersection test.

5.4. Implementation Details

We implemented all the presented algorithms in OpenGL4.4 using the Geometry Shader and the Transform Feedback capabilities. In particular, the BSH is stored in a Shader Storage Buffer Object where each node contains the indices of the children node (2 unsigned int), the position (3 float), the color (3 float), the radius (1 float) and the cone of normals (encoded in an unsigned int as RGBA color where the RGB channels contain the direction and the alpha contains half the aperture angle of the cone). Each node requires 40 bytes. All the three versions of the algorithm (Classic, Mipmapped, and Hybrid PBGI), store the microbuffers in a 2D texture with size equal to the viewport multiple by the size of the microbuffer. 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 with the Image Load/Store. While the Classic PBGI visits the BSH for each pixel inside a single shader and does not need additional memory, the Mipmapped and Hybrid PBGI visit the BSH in an iterative and parallel way using the geometry shader. We use the Transform Feedback to store the output of the geometry shaders, pairs of indices (2 unsigned int) that store the index of the node of the BSH and the Z-order index of the pixel in the G-Buffer. At each iteration, the algorithm takes in input the pairs of the previous one 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 array buffer of the Transform Feedback with the generated pair is empty. This means that all the pairs have been rasterized in the microbuffer texture. Finally, the Hybrid PBGI requires an additional stream in the Transform Feedback where to store the pairs where the cut of the BSH must be refined with the Classical version. In this case, the algorithm runs a different shader for each pair of the additional stream computing the optimal cut for the rasterization as the Classic PBGI.

6. Results

We tested the proposed pipeline with different datasets comparing the results of both the steps of the algorithm with state-of-the-art methods. We performed the tests on a PC with an Intel i7-6700 CPU, 32GB of RAM, and an NVIDIA GeForce GTX1080 GPU with 8GB of dedicated video memory. In the following sections, we present the used dataset (Section 6.1), the numerical and visual comparisons results for the HDR expansion (Section 6.2) and the PBGI rendering (Section 6.3), and finally the rendering results using some PBR models (Section 6.4).

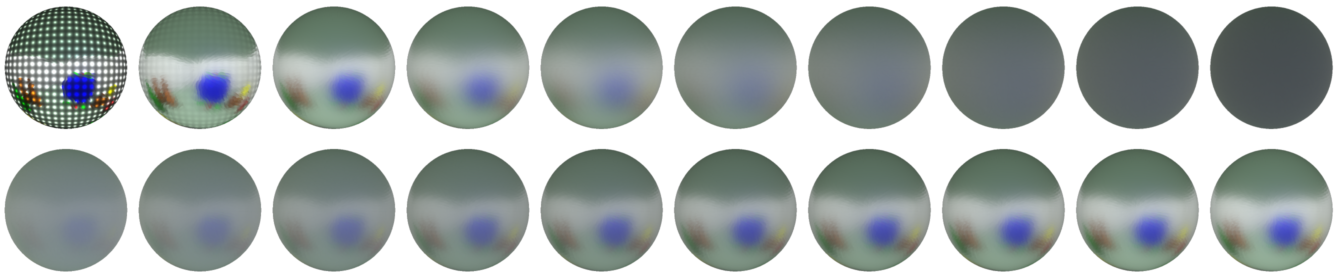


Figure 9: *Hybrid PBGI rendering of a sphere inside the scene* TOYROOM *by varying the parameters of a Disney Principled BRDF. (Top) Rendering with increasing roughness (from 0.1 to 1) with fixed metalness (1). (Bottom) Rendering with increasing metalness (from 0.1 to 1) with fixed roughness (0.3).*

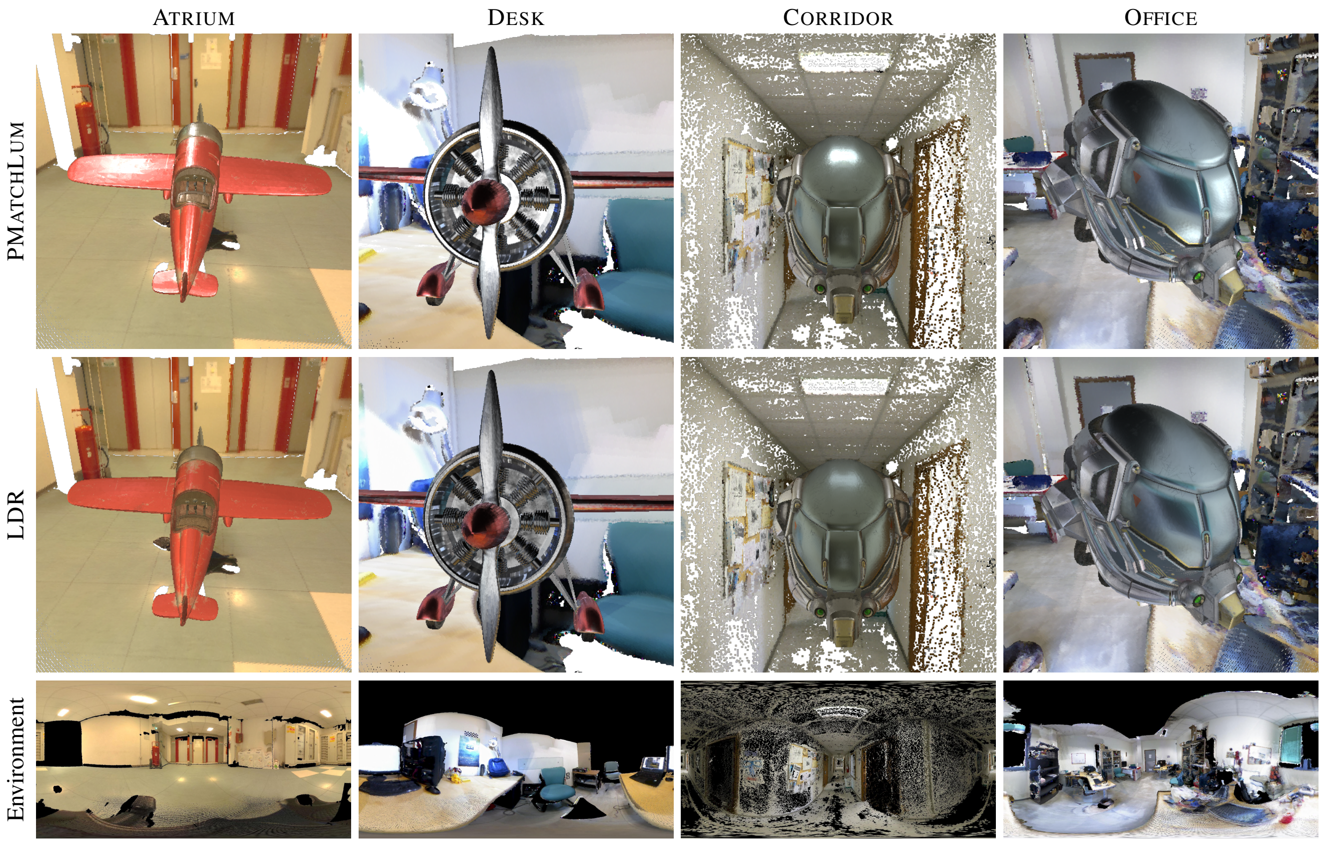


Figure 10: *Rendering results of the Hybrid PBGI with the LDR point cloud (central row) and the cloud expanded with the* PMATCHLUM *method (top row) in four different scenes. The rendered 3D models, an airplane and a helmet, has a spatially varying Disney Principled BRDF [Bur12]. The last row shows an equirectangular map of the point cloud acquired from the point of view of the model.*

6.1. Dataset

For our tests, we employed a set of several point clouds obtained with different technologies, some of them with ground truth HDR data. In particular, we tested the following seven scenes:

- SPONZA, a Montecarlo point sampling of the Crytek Sponza scene after the baking of the color data computed with a Path Tracing rendering (with HDR color data);
- ATRIUM and BUILDING, respectively an atrium and an outdoor scene reconstructed from a single HDR panoramic image acquired with a 360 deg camera where the point cloud was created with the user-assisted method proposed in [BCD*13] (with HDR color data);
- CORRIDOR, a scene reconstructed with the multiview stereo software COLMAP [SF16] using the 1065 video frames of a walking in the corridor (without HDR color data);

- OFFICE and DESK, respectively a scene of an office and of a desk with several light sources acquired with a Kinect using BundleFusion [DNZ*17] (without HDR color data);
- TOYROOM, a point cloud of a simple room with some colored objects modeled with Blender obtained with a Montecarlo sampling of the 3D model after the baking of the color data computed with a Path Tracing rendering (without HDR data). This dataset was only used to evaluate the performance and the quality of the new PBGI algorithm.

For the point clouds with HDR color data (SPONZA, ATRIUM and BUILDING), the LDR version was obtained by simply applying an exposure and a clamp operator to the HDR color using Equation 1. For rendering, we used several 3D models with different BRDF data such as a pure diffuse BRDF, the GGX BRDF [WMLT07], and the Disney Principled BRDF [Bur12].

	Points	**Shots**	**HDR**	**BSH**	**N.Nodes**	**N.Cluster**
SPONZA	3.0M	3	180.5s	13.82s	4.8M	1.6M
ATRIUM	818k	1	50.4s	3.62s	1.4M	430K
BUILDING	871k	1	50.1s	4.61s	450K	90K
CORRIDOR	2.87M	3	171.3s	13.96s	1.1M	860K
OFFICE	3.54M	1	59.5s	16.67s	3M	1.3M
DESK	2.81M	1	57.0s	14.57s	2M	590K

Table 1: *For each scene, the table reports the number of points, the number of cubemaps used for the HDR expansion, the time for the HDR expansion, the time to build the BSH hierarchy before the rendering with the proposed PBGI algorithm, and the number of nodes of the BSH (before and after the clustering).*

6.2. HDR Expansion Evaluation

To test the HDR expansion framework presented in Section 4, we used the tree clouds with ground trust HDR data, SPONZA, ATRIUM and BUILDING. Starting from the LDR version of these point clouds, we tested the three different approaches described in Section 4.1. The first one is the application of two different ITMOs: the global operator by Landis et al. [Lan02](LANDIS) and the expand-map based operator by Banterle et al. [BLDC06](BANTERLE). We limited our focus to these non-linear operators because non-linearity typically provides high-quality results for IBL [BDA*09]. In addition, we tested the results obtained by the trivial extension of these ITMOs to the structure of a point cloud (LANDIS-PC and BANTERLE-PC), by applying directly the operator to each point of the cloud. The second approach is based on two deep learning networks for inverse tone mapping proposed by Eilertsen et al. [EKD*17](EILER) and by Endo et al. [EKM17](ENDO). We use the pre-trained network of the authors. The last approach is the proposed randomized match method based on the PatchMatch algorithm (PMATCH). For the deep learning and the PatchMatch approaches, we tested both the possibility to transfer only the luminance (method with the suffix LUM) or the entire color (method with the suffix RGB) of the HDR expanded images. For the ITMOs we use only the luminance because these operators work only on this data.

We evaluated the output of these methods in two different ways. The first one computes the Root Mean Square Error (RMS) of

SPONZA

	View1			View2			Points
	RMS	**HDR-VDP**	**SSIM**	**RMS**	**HDR-VDP**	**SSIM**	**RMS**
LDR	1.137	78.86	0.796	0.040	80.59	0.978	**3.454**
BANTERLE	10.58	78.03	0.843	0.469	78.30	0.896	8.314
LANDIS	16.97	77.60	0.833	0.820	77.70	0.871	12.97
BANTERLEPC	4.940	78.18	0.865	0.225	79.18	0.936	9.147
LANDISPC	6.844	78.11	0.857	0.351	78.69	0.909	17.15
EILERLUM	0.572	79.51	0.909	0.020	82.14	0.996	3.507
EILERRGB	0.567	79.50	0.909	0.019	82.10	0.996	3.507
ENDOLUM	1.045	79.00	0.811	0.038	80.80	0.982	**3.454**
ENDORGB	1.058	79.03	0.811	0.039	80.80	0.982	**3.454**
PMATCHLUM	0.442	**79.73**	**0.941**	0.013	**83.30**	**0.998**	3.555
PMATCHRGB	**0.425**	79.68	**0.941**	**0.012**	83.29	**0.998**	3.555

ATRIUM

	View1			View2			Points
	RMS	**HDR-VDP**	**SSIM**	**RMS**	**HDR-VDP**	**SSIM**	**RMS**
LDR	0.231	78.55	0.832	0.127	79.32	0.938	3.649
BANTERLE	0.114	**81.82**	0.938	0.061	**82.53**	0.982	2.383
LANDIS	0.175	80.03	0.932	0.101	80.45	0.973	3.349
BANTERLEPC	0.193	80.01	0.804	0.100	80.75	0.941	5.099
LANDISPC	0.150	80.52	0.948	0.088	80.93	0.974	2.834
EILERLUM	0.257	79.10	0.881	0.137	79.65	0.964	5.554
EILERRGB	0.240	79.11	0.881	0.126	79.72	0.964	5.554
ENDOLUM	0.245	78.87	0.824	0.136	79.55	0.937	3.549
ENDORGB	0.245	78.85	0.824	0.136	79.54	0.937	3.549
PMATCHLUM	0.109	81.33	**0.969**	0.060	81.86	**0.990**	**2.145**
PMATCHRGB	**0.068**	81.69	**0.969**	**0.035**	82.35	**0.990**	**2.145**

BUILDING

	View1			View2			Points
	RMS	**HDR-VDP**	**SSIM**	**RMS**	**HDR-VDP**	**SSIM**	**RMS**
LDR	0.129	82.13	0.833	0.080	82.93	0.890	0.064
BANTERLE	0.182	81.63	0.913	0.065	82.78	0.933	0.087
LANDIS	0.352	80.87	0.845	0.080	81.80	0.909	0.137
BANTERLEPC	0.107	83.08	0.911	0.090	82.83	0.815	0.493
LANDISPC	0.128	82.35	0.930	0.064	82.99	0.938	0.049
EILERLUM	0.310	80.79	0.880	0.073	82.10	0.917	0.130
EILERRGB	0.278	80.78	0.880	0.071	82.16	0.917	0.130
ENDOLUM	0.154	80.77	0.835	0.076	82.52	0.893	0.100
ENDORGB	0.157	80.69	0.835	0.078	82.49	0.893	0.100
PMATCHLUM	0.059	83.38	**0.985**	**0.065**	83.61	**0.938**	**0.046**
PMATCHRGB	**0.043**	**83.54**	**0.985**	0.067	**83.68**	**0.938**	**0.046**

Table 2: *The table contains the error measures of the expanded point clouds with respect to the ground truth HDR cloud. The columns "View1" and "View2" contain the error measures of the renderings obtained with the Hybrid PBGI (Section 5.2.2) for two different rendering viewpoints. The used error metrics are the RMS error, the quality of HDV-VDP-2.2, and the Structure Similarity (SSIM). The last column contains the RMS error computed directly on the point cloud color data. The green text highlights the best result for each test (for HDR-VDR and SSIM higher the better).*

the HDR expanded point cloud with respect to the ground truth HDR data. The second evaluation compares the indirect effect produced by the cloud on a virtual object inserted in the scene and rendered with the proposed Hybrid PBGI in Section 5.2.2. In particular, given the renderings with the expanded HDR clouds, we compare them numerically with respect to the rendering with the ground truth HDR cloud using three error measures: the RMS error, the quality value of the HDR-VDP-2.2 [NMPDSLC15] and the Structure Similarity (SSIM) [WBSS04]. For each dataset, we used two different viewpoints for the camera. Table 2 contains the numerical values of this comparison. Figures 12 and 13 show the produced renderings with the different expanded point clouds compared and the renderings with the HDR ground truth and the LDR point cloud. We removed the image background to avoid wrong perceptual impressions. These two figures show in the bottom a panoramic view of the tested scene taken from the centroid of the rendered object. Comparing these renderings, the PMATCH methods show the best results according to all the error measures and in particular, the method based on the transfer of all the RGB color generates renderings with slightly lower errors. Also according to the RMS error on the point cloud data, the PMATCH methods are the best ones but the differences from the LDR cloud is lower. Consider that the trivial computation of the RMS on the cloud data does not take account of the distribution of the error, on the contrary of the evaluation of the rendering effects. This is the reason because the LDR point cloud of the scene SPONZA has a lower point RMS lower than the other method but it produces visually worse rendering results. Table 1 contains the timings for the expansion of the different point clouds with the PMATCHLUM method.

6.3. PBGI Performance

We tested the performance of the proposed PBGI algorithms in Sections 5.2.1 and 5.2.2 (Mipmapped and Hybrid PBGI) with the classic microbuffer rendering by Ritschel et al. [REG*09]. Table 3 contains the rendering time in milliseconds of the three versions of the PBGI rendering (Classic, Mipmapped and Hybrid) by changing the viewport size and the microbuffer size. We tested two scenes (TOYROOM and SPONZA) with two different rendering viewpoint: the first one where is visible a detail of the object that gets all the viewport; the second one where the entire object is visible in the viewport. The table in the bottom contains the maximum memory occupied by the additional buffer to store the output primitives of the Geometry Shader inside the Transform Feedback with respect to the Classic PBGI. The letter 'x' means that the algorithm fails due to memory issue (the texture to store the microbuffers or the buffer for the output of the Transform Feedback are too big). The Hybrid approach is the faster one reaching also a speed-up of 10x with respect to the Classic microbuffer rendering, especially for big viewport and microbuffer. Furthermore, the memory occupancy is always less than the Mipmapped version.

To evaluate the approximation introduced by the proposed Hybrid PBGI, we compared the rendering results of the proposed algorithm with the outputs of a path tracing and the output of an environment mapping rendering. Figure 14 shows the renderings of the same object in the TOY ROOM scene by changing the roughness parameter of a GGX BRDF. For the environment mapping, the maps were taken from the position of the relighted object, the optimal situation for this algorithm. The last columns show the maps of the luminance differences from the path tracing rendering with the relative RMS error. Numerically, the error introduced by the Hybrid PBGI is lower or comparable to the error of the environment mapping rendering. The main advantages of the proposed method are the real-time rendering and the possibility to capture also close and mid-range illumination effects such as the green color bleeding on the front of the bust.

The main limitation of our method is the rendering of very specular surfaces (i.e., low roughness) due to the approximation introduced by the size of the microbuffers. For small microbuffer, the algorithm may not capture the main/optimal reflection directions, and it may lead to a higher error, which may create visible artifact as shown in Figure 9. This happens for roughness parameters below 0.2. The use of an importance sampling strategy during the rendering can solve these artifacts.

6.4. HDR Rendering

Figure 11 shows a comparison of the renderings computed with LDR point cloud and the HDR cloud expanded with the method PMATCHLUM of a new object placed inside the scene. The figure shows also the difference map between the two renderings computed only on the new object placed in the scene. In all the test it is visible the additional contribution of the saturated areas of the cloud that are expanded with the proposed method. The proposed PBGI framework is also able to manage different type of BDRF. Figure 10 shows the rendering of an airplane and a helmet with a spatially varying Disney Principled BRDF in different scenes from two rendering point of views. Also in this case, the contribution of the HDR expansion is visible on the specular regions of the object.

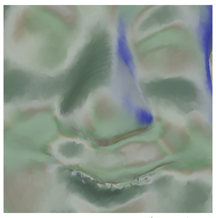
7. Conclusion

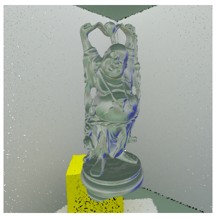
We presented an innovative algorithm to use the data encoded into a captured point cloud (radiance, spatial and visibility information) of a real-world scene to compute the real-time relighting of the virtual object inside the scene. The framework is based on two components: the HDR expansion of the input point cloud, usually captured with LDR devices; the real-time relighting of the virtual object using a new PBGI algorithm. For the HDR expansion, starting from a single exemplary HDR photo of the environment, we propose a framework based on the HDR reconstruction of a set of LDR rendering of the point cloud. We tested three different strategies to expand the LDR rendering (image inverse tone mapping, deep learning architecture and randomized matching), showing numerically that the methods based on the PatchMatch between the HDR photos and the LDR renderings are the best solution. After the reprojection of the HDR data from the rendering to the point cloud, the algorithm computes a Poisson HDR infilling for the areas not covered by the rendering or with a low-quality HDR reconstruction. The expanded HDR point cloud is used as input for the proposed PGBI algorithm to compute in real-time the final rendering of the object. Starting from the construction of a BSH of the cloud in a preprocessing step, the proposed method takes advantage of the geometry shader and of a new mipmapping operator for the

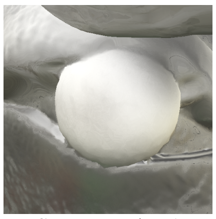
Microbuffer size

Time Perf.	ToyRoom						Sponza					
(ms)	**View1**			**View2**			**View1**			**View2**		
Classic	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**
128x128	9.7	21.3	35.2	7.5	16.7	27.0	32.7	79.9	135.9	23.8	58.9	90.8
256x256	37.8	85.1	254.5	15.6	35.2	81.2	146.3	374.9	953.3	36.8	90.3	161.9
512x512	138.9	598.0	3272.0	41.1	173.5	920.6	584.9	2093.5	10k	126.5	380.6	1680.0
1024x1024	752.0	x	x	306.2	x	x	3025.8	x	x	890.1	x	x
Mipmapped	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**
128x128	15.2	31.2	51.9	9.9	16.4	24.8	45.4	115.8	194.7	24.1	53.1	81.0
256x256	40.0	88.0	151.0	22.6	47.9	77.9	144.8	370.4	606.9	75.7	184.7	292.5
512x512	131.1	293.0	565.2	64.5	148.4	281.6	515.5	x	x	242.8	640.0	1237.1
1024x124	604.6	x	x	244.7	x	x	x	x	x	930.7	x	x
Hybrid	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**
128x128	9.1	13.3	24.7	6.5	8.8	11.5	19.2	42.3	64.6	9.6	15.8	22.1
256x256	19.3	41.4	82.3	9.9	16.9	29.8	58.2	150.0	233.4	20.1	41.2	67.8
512x512	62.1	156.7	316.2	24.2	53.1	108.5	205.1	568.1	927.6	58.5	132.7	243.7
1024x1024	285.3	x	x	115.8	x	x	874.1	x	x	227.0	x	x

Memory Perf.	ToyRoom						Sponza					
(MB)	**View1**			**View2**			**View1**			**View2**		
Mipmapped	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**
128x128	10	16	27	4	6	11	30	84	128	13	26	45
256x256	38	55	90	15	24	39	113	303	435	47	101	174
512x512	146	206	325	50	81	135	440	x	x	156	360	642
1024x124	756	x	x	183	x	x	x	x	x	603	x	x
Hybrid	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**	**16x16**	**24x24**	**32x32**
128x128	5	10	19	1	2	4	20	45	80	3	7	13
256x256	22	39	76	4	8	15	82	183	319	14	29	52
512x512	89	158	306	18	32	61	329	734	1279	56	115	208
1024x1024	356	x	x	71	x	x	1.3k	x	x	227	x	x


ToyRoom-View1


ToyRoom-View2


Sponza-View1

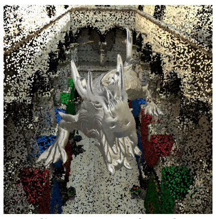

Sponza-View2

Table 3: *Performance comparison (time and memory occupancy) of the three PBGI algorithms, Classic [REG*09], Mipmapped (Section 5.2.1) and Hybrid (Section 5.2.2), varying the viewport and the microbuffer size. The tests were performed on the point clouds* ToyRoom *and* Sponza *using the two viewpoints on the right: in the first one the object to relight gets all viewport; in the second one the entire object is visible in the viewport. The 'x' letter means that the algorithm was unable to run for memory issues. For the memory occupancy, the table reports only the additional memory required to store the output primitive of the geometry shader in the transform feedback buffers.*

G-buffer of the rendered object to speed up the rendering. In particular, we proposed a hybrid solution that splits the visit of the BSH into two parts. In the first one, the visit of the BSH is done in parallel and in an iterative way for each pixel with multiple instances of the geometry shader. During this visit, the geometry shader can decide to refine the visit in the BSH or in the next level of the G-Buffer. In the second part, when the solid angle of the node is below a multiple of the solid angle of a pixel in the microbuffer, the algorithm refines the visit for each sub-tree reached by the first visit in a single shader until the rasterization of the node in the microbuffer. In this way the algorithm can share the visit of the tree for near pixels in the viewport, reducing the rendering time at the cost of a minimum memory overhead. We tested the algorithm with several datasets with and without HDR ground truth information, showing very good numerical and very convincing visual results. We show also the result obtained in an AR scenario with some small animation of PBR models in the video of the supporting information.

In future works, we plan to investigate some preprocessing steps to increase the quality of the input LDR point cloud in term of better density and distribution of points, taking also account of the source of the data that produced it. This step should help to reduce areas in the cloud without information, improving the quality of the rendering. A further future research direction is the proposal of an interactive HDR expansion method that could help to use the framework in a dynamic AR scenario where the lighting environment can change.

Acknowledgements

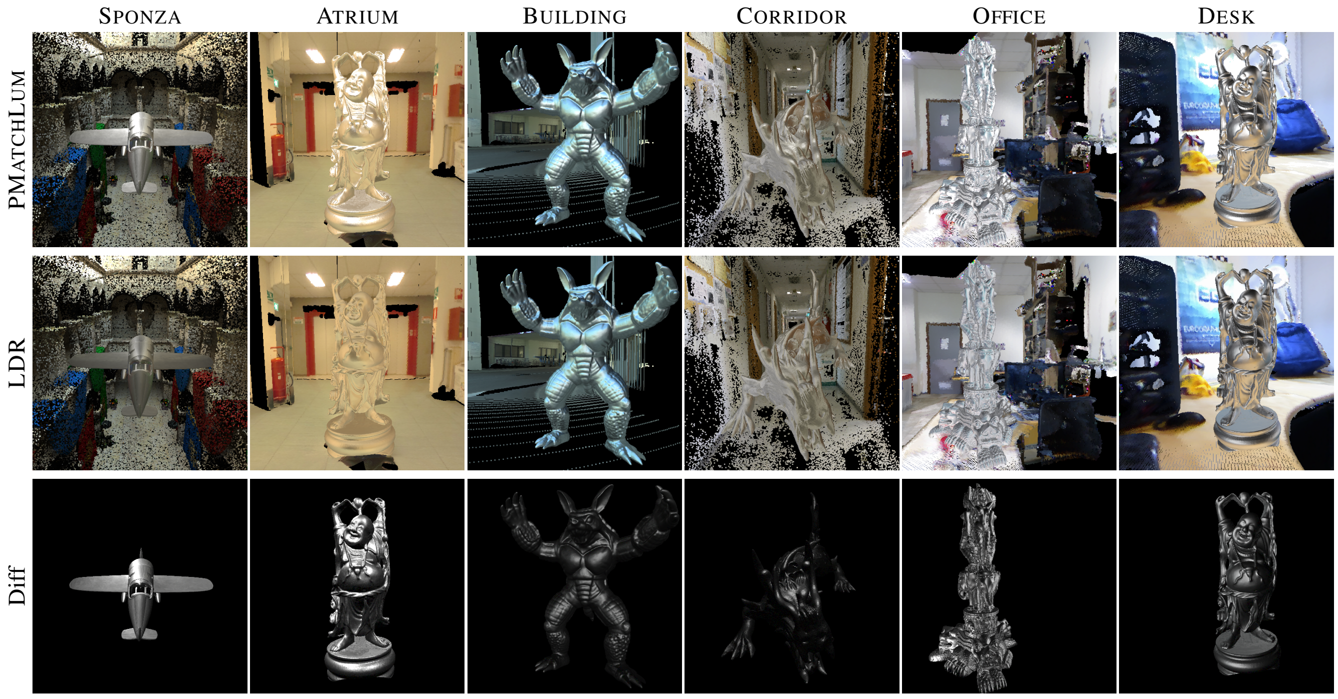


Figure 11: *Comparison of rendering results obtained with the HDR cloud expanded with the method* PMATCHLUM *(top row) and the input LDR point cloud (central row). For all objects, we used a GGX BRDF with roughness equal to 0.25. The last row shows the differences of the rendered object in the images.*

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

[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. *Computer Graphics Forum 32*, 7 (October 2013), 411–420. 3, 9

[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. 10

[BE05] BAREQUET G., ELBER G.: Optimal bounding cones of vectors in three dimensions. *Inf. Process. Lett. 93*, 2 (Jan. 2005), 83–89. 5, 7

[BLDC06] BANTERLE F., LEDDA P., DEBATTISTA K., CHALMERS A.: Inverse tone mapping. In *GRAPHITE '06: Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia* (New York, NY, USA, 2006), ACM, pp. 349–356. 2, 3, 10

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D. B.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (Aug. 2009). 3

[BSGF10] BARNES C., SHECHTMAN E., GOLDMAN D. B., FINKELSTEIN A.: The generalized PatchMatch correspondence algorithm. In *European Conference on Computer Vision* (Sept. 2010). 3, 4

[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. 9, 10

[CEN07] CHAN T., ESEDOGLU S., NI K.: Histogram based segmentation using wasserstein distances. In *Scale Space and Variational Methods in Computer Vision* (Berlin, Heidelberg, 2007), Sgallari F., Murli A., Paragios N., (Eds.), Springer Berlin Heidelberg, pp. 697–708. 4

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

[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. *Computer Graphics Forum 33*, 2 (2014), 459–468. 3

[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 '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (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. *Computer Graphics 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 (May 2017), 24:1–24:18. 3, 10

[EKD*17] Eilertsen G., Kronander J., Denes G., Mantiuk R., Unger J.: Hdr image reconstruction from a single exposure using deep cnns. *ACM Transactions on Graphics (TOG) 36*, 6 (2017). 2, 3, 10

[EKM17] Endo Y., Kanamori Y., Mitani J.: Deep reverse tone mapping. *ACM Trans. Graph. 36*, 6 (Nov. 2017), 177:1–177:10. 2, 3, 10

[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

[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 (Nov. 2017), 176:1–176:14. 3

[HKS09] Ha L., Krüger J., Silva C. T.: Fast four-way parallel radix sorting on gpus. *Computer Graphics 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. *Computer Graphics Forum 30*, 4 (2011), 1233–1240. 3, 6

[HSH*17] Hold-Geoffroy Y., Sunkavalli K., Hadap S., Gambaretto E., Lalonde J.: Deep outdoor illumination estimation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017* (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 *Proceedings of the Fourth ACM SIGGRAPH / Eurographics Conference on High-Performance Graphics* (2012), EGGH-HPG'12, pp. 33–37. 5

[KBG*15] Kronander J., Banterle F., Gardner A., Miandji E., Unger J.: Photorealistic rendering of mixed reality scenes. *Computer Graphics Forum 34*, 2 (May 2015), 643–665. 3

[KO14] Kovaleski R. P., Oliveira M. M.: High-quality reverse tone mapping for a wide range of exposures. In *27th SIBGRAPI Conference on Graphics, Patterns and Images* (August 2014), IEEE Computer Society, pp. 49–56. 2

[Lan02] Landis H.: Production-ready global illumination. In *SIGGRAPH Course Notes* 16 (2002), pp. 87–101. 2, 3, 10

[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. *Computer Graphics 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), PG '07, IEEE Computer Society, pp. 382–390. 3

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

[NMPDSLC15] Narwaria M., Mantiuk R., P. Da Silva M., Le Callet P.: 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. 11

[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. 3, 4

[RDGK12] Ritschel T., Dachsbacher C., Grosch T., Kautz J.: The state of the art in interactive global illumination. *Computer Graphics 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, 6, 11, 12

[RPAC17] Rhee T., Petikam L., Allen B., Chalmers A.: MR360: mixed reality rendering for 360Âř panoramic videos. *IEEE Trans. Vis. Comput. Graph. 23*, 4 (2017), 1379–1388. 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

[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 *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2016), pp. 4104–4113. 3, 9

[SL17] Silvennoinen A., Lehtinen J.: Real-time global illumination by precomputed local reconstruction from sparse radiance probes. *ACM Transactions on Graphics (Proceedings of SIGGRAPH Asia) 36*, 6 (Nov. 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 (November 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 (April 2004), 600–612. 11

[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. *Computer Graphics Forum 32*, 4 (2013), 117–123. 3, 7

[WMB15] Wang B., Meng X., Boubekeur T.: Wavelet point-based global illumination. *Computer Graphics Forum 34*, 4 (2015), 143–153. 3

[WMLT07] Walter B., Marschner S. R., Li H., Torrance K. E.: Microfacet models for refraction through rough surfaces. In *Proceedings of the 18th Eurographics Conference on Rendering Techniques* (2007), EGSR'07, Eurographics Association, pp. 195–206. 10

[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. 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

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

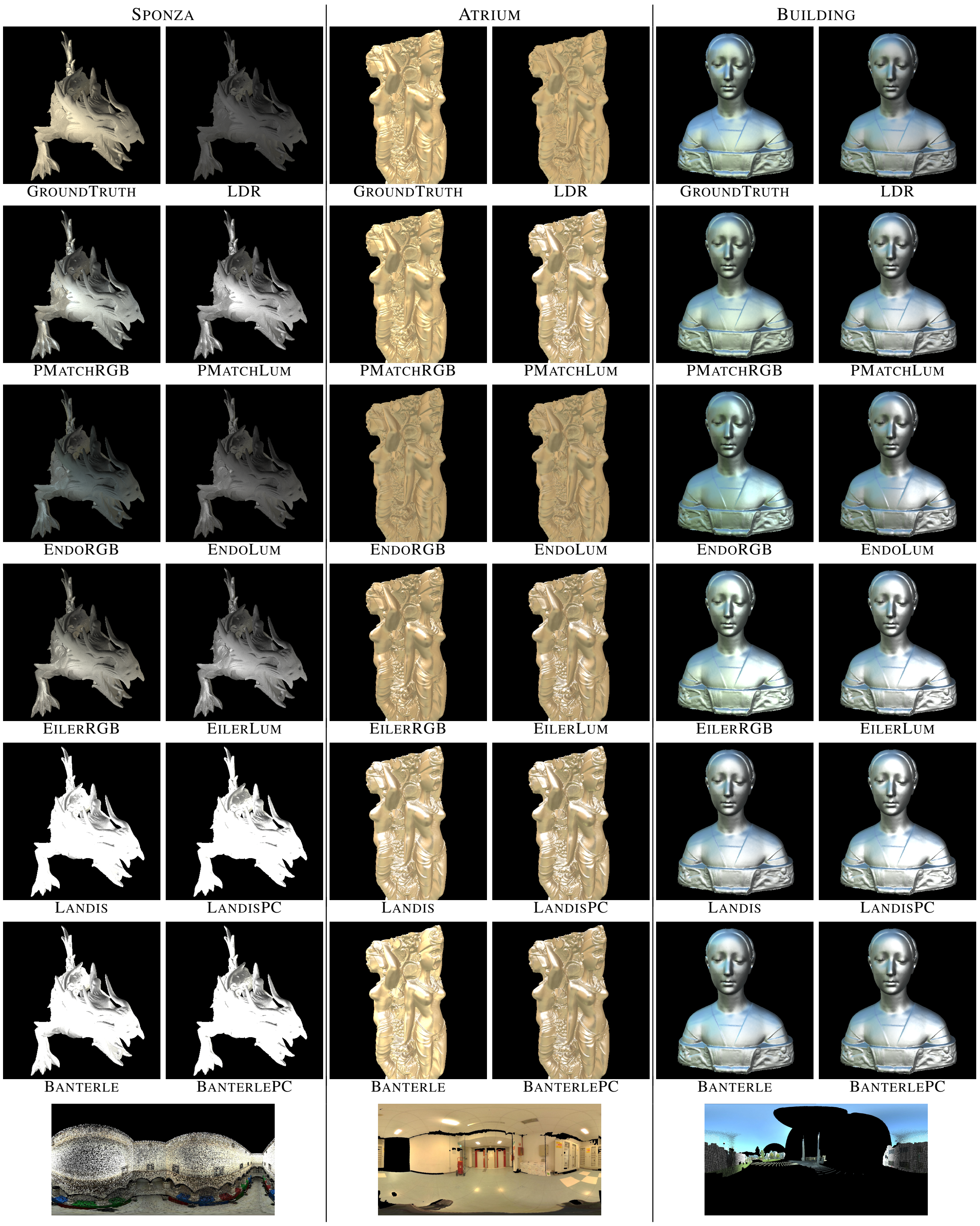


Figure 12: *Comparison of the renderings using the point cloud expanded with the method tested in Section 6.2 for the scene* SPONZA, ATRIUM *and* BUILDING. *The figure shows also the renderings obtained with the ground truth cloud (*GROUNDTRUTH*) and with the LDR cloud (LDR) and the panorama of the point cloud from the point of view of the object centroid.*

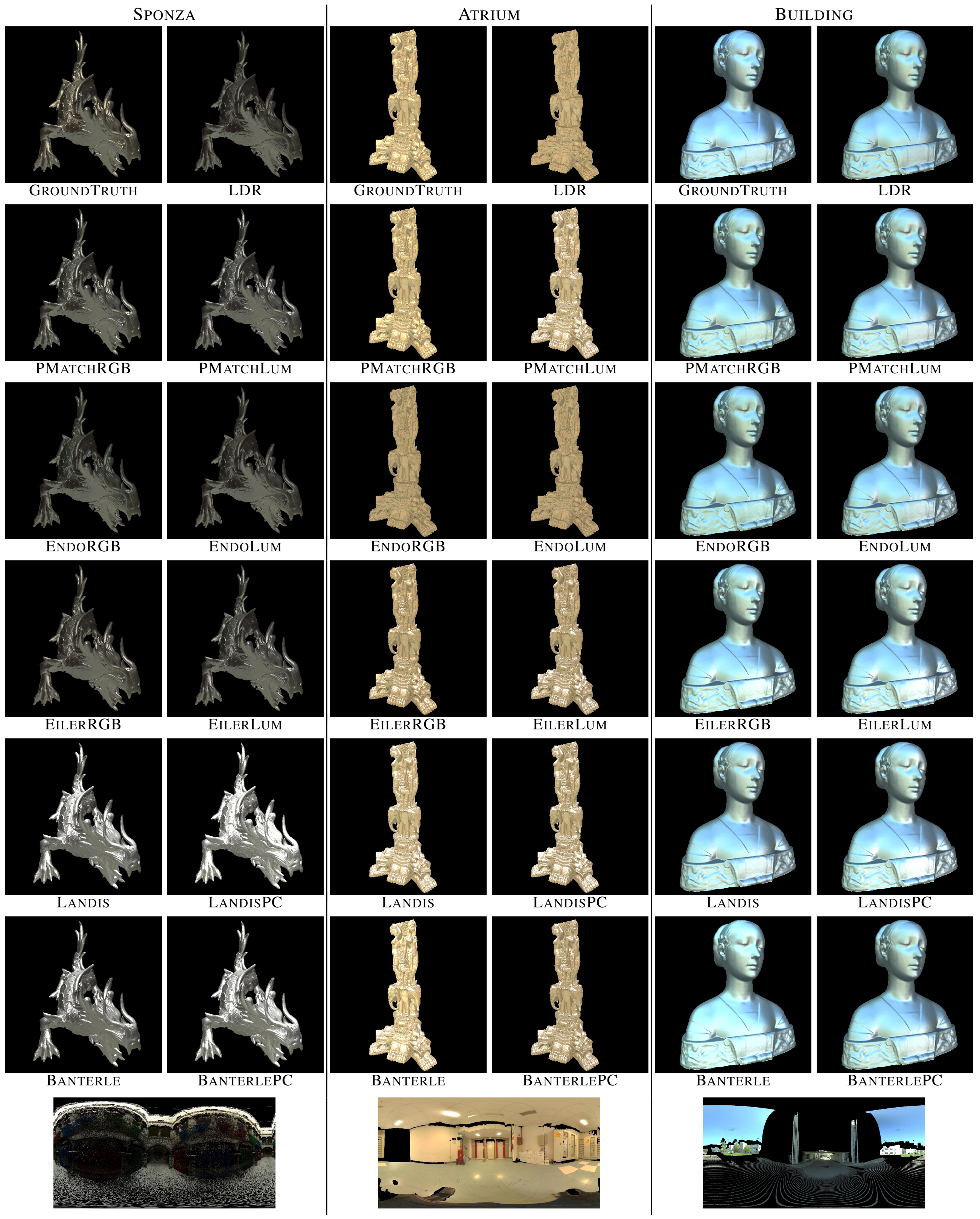


Figure 13: *Comparison of the renderings using the point cloud expanded with the method tested in Section 6.2 for the scene* SPONZA, ATRIUM *and* BUILDING. *The figure shows also the renderings obtained with the ground truth cloud (*GROUNDTRUTH*) and with the LDR cloud (LDR) and the panorama of the point cloud from the point of view of the object centroid.*

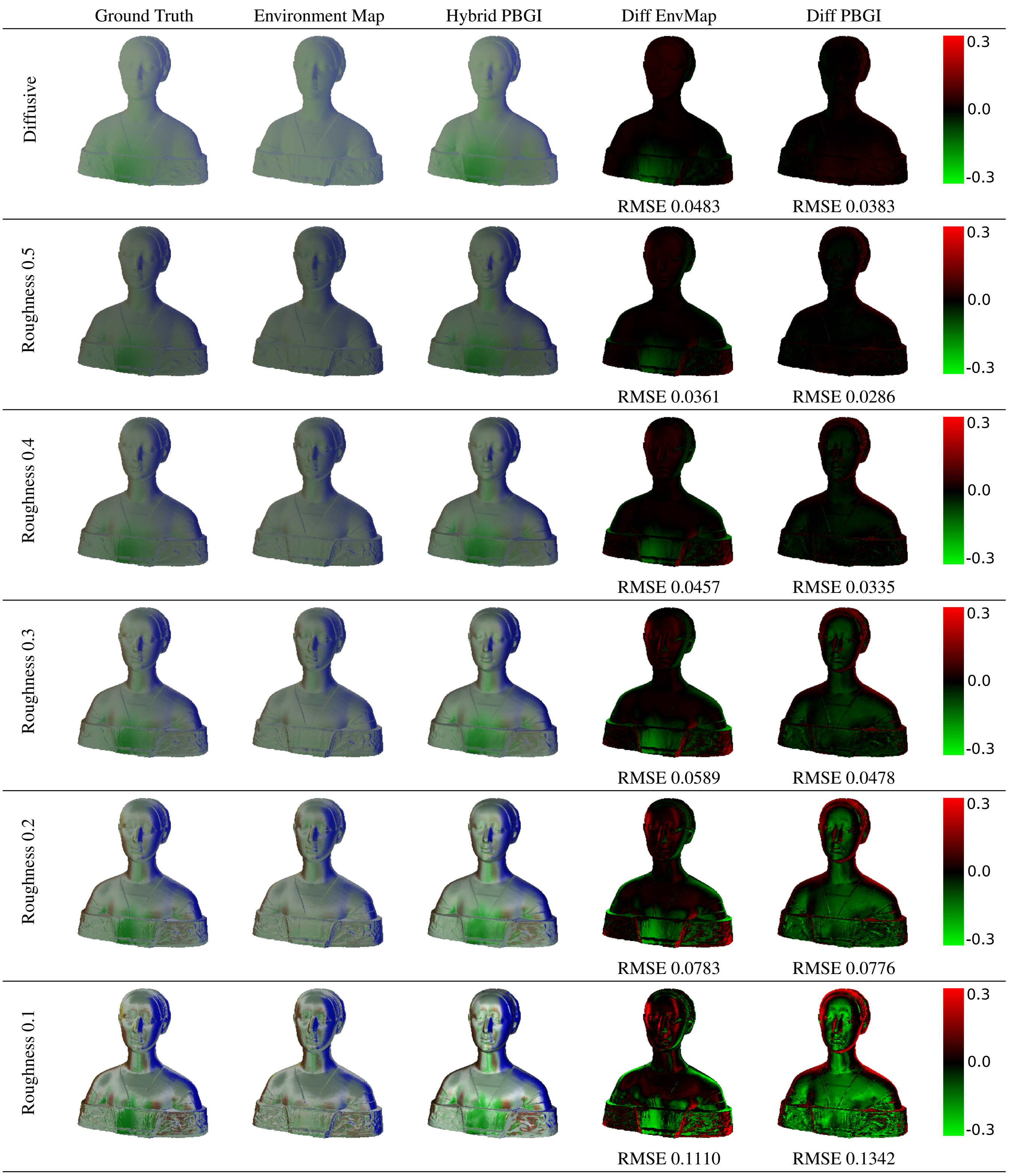


Figure 14: *Comparison of the Hybrid PBGI rendering with a path tracing and the classical environment mapping. The last two columns show the difference map from the path tracing result of the environment mapping and the proposed PBGI approach with the relative RMS error. The difference is on the luminance channel. For each column, we show the object varying the roughness parameter of the GGX BRDF.*