A Comparative Assessment Of Implicit and Explicit Plenoptic Scene Representations

Davi R. Freitas
INRIA Rennes
Rennes, France
davi-rabbouni.de-carvalho-freitas@inria.fr

Ricardo L. de Queiroz
Universidade de Brasília
Brasília, Brazil
queiroz@ieee.org

Ioan Tabus
Tampere University
Tampere, Finland
ioan.tabus@tuni.fi

Christine Guillemot
INRIA Rennes
Rennes, France
christine.guillemot@inria.fr

***Abstract*—3D scene representation has been a central theme of study for a wide range of applications, and the representation of light behavior is one of the relevant topics when producing realistic models. In this work, we create a framework to assess the representation of non-Lambertian scenes by generating a pipeline to create plenoptic point clouds (PPCs) systematically and evaluating them against implicit solutions, such as Neural Radiance Fields (NeRF)-like models. We compare such approaches according to rendering quality and compression efficiency. On the compression side, we propose an encoding scheme for PPC, leveraging the occlusion masks of the points and the Moving Picture Expert Group's (MPEG) Geometry-Based Solid Content Test Model (GeS-TM). Rendering results over the training views show that the uncompressed PPC outperforms 3D Gaussian Splatting (3DGS) by 1.51 dB, on average, for the 8 scenes of the NeRF Synthetic 360 dataset. In compression efficiency, 3DGS outperforms the compressed PPCs by 0.7 dB in BD-PSNR on average. Our occlusion-aware encoding scheme reduces the size of uncompressed PPCs up to 800 times, outperforming current encoding schemes for PPC by 1.9 dB in BD-PSNR.**

***Index Terms*—NeRF, 3D Gaussian Splatting, Plenoptic Point Clouds, Compression**

I. Introduction

The modeling of three-dimensional (3D) scenes has been a focal point of study across interdisciplinary domains. In this context, understanding and analyzing the complex behavior of light in 3D scenes plays a central role in the task of modeling them realistically. Particularly, the ability to model scenes and objects that have different brightness levels depending on the viewer's position, *i.e.* non-Lambertian, is highly desired. As such, the plenoptic function aims to represent the intensity of light at any given point by a 7-dimensional function

$$P(x, y, z, \theta, \phi, \lambda, t), \tag{1}$$

where (x, y, z) are space coordinates, (θ, ϕ) the viewing direction, λ is the light wavelength and t is the time [1]. Such function can be incorporated differently in the modeling of 3D scenes according to the type of scene representation.

Implicit scene representations refer to methods that implicitly define a scene as a continuous function of the space coordinates through learned or inferred parameters, generally via a neural network. Neural Radiance Fields (NeRF) [2] is a scene-specific, 3D implicit representation that models a scene through the use of multi-layer perceptrons (MLP). The MLPs are an implementation of a continuous function F that maps the 3D coordinates of point $\mathbf{x} = (x, y, z)$ from a viewing direction $\mathbf{n}$ with azimuth and elevation of $\mathbf{n} = (\theta, \phi)$ into a color $\mathbf{c}$ and density σ:

$$(\mathbf{c}, \sigma) = F_{\Theta}(\mathbf{x}, \mathbf{n}). \tag{2}$$

Hence, the plenoptic function in Eq. 1 is naturally represented by NeRF, with the coordinates and viewing direction being represented through $\mathbf{x}$ and $\mathbf{n}$ in Eq. 2 to obtain the view-dependent color $\mathbf{c}$, which models the dependency on the wavelength λ of the plenoptic light intensity function in Eq.1.

With explicit scene representations, the information of the scene is directly encoded in a structured and interpretable format, typically given by the geometry and a set of appearance attributes of the scene. As such, point clouds (PC) have gained traction due to their simplicity compared to meshes concerning their real-time capture and rendering capability. For these structures, the plenoptic function can be represented through a plenoptic point cloud (PPC) [3], where the coordinates (x, y, z) can be discretized into voxel positions, the wavelength λ can be represented by the red-green-blue (RGB) components, and the viewing direction (θ, ϕ) can be sampled by a set of RGB triplets due to the finite number of camera rigs at the time of capture. The amount of data required to represent these volumes fomented the interest of organizations, such as the Moving Picture Expert Group (MPEG), in advancing the PC compression standards not only for vanilla PCs, but also for PPCs [4]. Nonetheless, one of the main constraints with these compression activities is the reduced number of PPCs [5] due to the non-trivial process of capturing them, limiting the standardization experiments to a constrained dataset.

Therefore, the contributions of our work are:

1) Propose a way to compare the plenoptic modeling of both explicit (PPCs) and implicit (NeRF-like) 3D schemes in an encoding context, in terms of:
   a) rendering quality
   b) compression efficiency
2) Provide a pipeline to generate PPCs systematically, using the colorless point cloud and images of the scene,
3) Propose an efficient encoding scheme of the PPCs based on their occlusion maps over the training views.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 956770.

We show that, over a set of training views, PPCs excel in scenes with a higher degree of specularity, while implicit methods outperform explicit ones in compression efficiency.

II. Method

PPCs provide a way to enhance the photorealism of explicit representations by taking into account the specularity of the points through the modeling of view-dependent colors. This section details the steps that were taken to generate PPCs in a systematic manner, enabling their assessment against state-of-the-art (SOTA) neural implicit methods.

The bottom branch in Fig. 1 shows the step-by-step generation of our PPCs. Considering a mesh inside a Blender model and a set of camera poses $\mathbf{d} = \{d_i\}_{i=1}^{k}$, we generate with the Blender renderer (upper branch in Fig. 1) a set of RGB images $\mathbf{V} = \{V_i\}_{i=1}^{k}$ corresponding to each $\mathbf{d}$, such that we create a PPC of k colors by projecting the pixel colors into the 3D scene. In order to obtain the geometry, we sample and voxelize the mesh into an $N \times N \times N$ colorless PC, also known as depth-N PC. Moreover, we generate a set of rendered views $\mathbf{V}' = \{V_i'\}_{i=1}^{k}$ for the same set of poses $\mathbf{d}$ over the generated PPCs to provide a manner to compare to the SOTA neural implicit methods. Since the latter uses a set of images and poses as input for training in order to output images for any given direction, this comparison can be achieved by training the models over $\mathbf{d}$ and $\mathbf{V}$ (top branch). In this work, the goal is to assess how the explicit and implicit models fare in an encoding context, hence a render-based evaluation over the k training views is desired.

A. Extraction of Point Clouds from the Blender Models

The first step to generate the PPCs is to extract a dense set of points from the Blender models. Since such models have an underlying mesh representation for their geometry, it is necessary to sample these structures into "occupied" voxels out of the mesh polygonal faces, where voxels are defined as nodes in a uniform cubic $N \times N \times N$ grid. For that, we evaluated three different sampling approaches over triangulated meshes.

The first method, referred to here as "Triangle Split", discretizes the mesh directly to a voxel volume by splitting the triangles into smaller ones until the longest edge is smaller than 0.5 voxel. Once this condition is reached, the voxel beneath the vertice coordinates is set to one.

The Poisson disk sampling [6] uniformly distributes randomly the points that are generated on the mesh's surface with a minimum geodesic distance constraint between them of $2r$ given a disk of radius r. Its blue noise properties reduce the effects of aliasing in the later steps of the resulting model.

We call the third method "Area-Based" [7], since the number of sampled points is dependent on the triangle area. Based on a surface density parameter – number of points per square unit –, each triangle will have a different number $m \in \mathbb{R} \setminus \mathbb{Z}$ of sampled points according to their respective area. The fractional part of m is then used as a probability to pick another point in the triangle.

B. Colorization of the Plenoptic Point Clouds

Once we have the colorless voxelized PC, we can leverage $\mathbf{V}$ to create the K RGBs for each point of the PPC. Let $\mathbf{X}$ be a point in 3D space with its position in the World Coordinate System given by $\mathbf{X_w}$. We forward project $\mathbf{X}$ to each Camera Coordinate System through $\tilde{\mathbf{X}}_c = \mathbf{d}\tilde{\mathbf{X}}_w$, where $\tilde{\mathbf{X}}_w$ and $\tilde{\mathbf{X}}_c$ denote $\mathbf{X}$ in homogeneous coordinates in the world and camera coordinate system, respectively. Additionally, $\mathbf{d} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix}$ corresponds to the $4{\times}4$ extrinsic matrix containing each camera rig's translation $\mathbf{t}$ and rotation $\mathbf{R}$. The 2D pixel coordinates $\mathbf{x}_c$ are then obtained by normalizing the homogeneous image coordinates $\tilde{\mathbf{x}}_c$ by the depth, where $\tilde{\mathbf{x}}_c = \mathbf{K}\tilde{\mathbf{X}}_c$. The camera intrinsic matrix $\mathbf{K}$ is known during the generation of $\mathbf{V}$. As the PC is voxelized, the center of each of the occupied voxels is considered as the starting position to be projected, in order to reduce potential aliasing artifacts.

Furthermore, the forward projection is also used to estimate the occlusion of points for each view. Let $\mathcal{P}_{uv} = \{\mathbf{P}_i | \pi(\mathbf{P}_i) = (u, v)\}_i$ be the set of 3D points that are projected by a function π to the same pixel of coordinates (u, v). We identify the points with minimum z-value in the camera coordinate system, *i.e*, $\mathbf{P}_{uv}^* = \arg\min_{\mathbf{P}_i \in \mathcal{P}_{uv}} z_i$, and mark them as "non-occluded". Then, we filter out the remaining points, that is, the ones that fall into the "occluded" class. Thus, points $\mathbf{P}_i$ having their z_i larger than $\mathbf{P}_{uv}^*$ are marked as occluded. We use the occlusion mask $\mathbf{O}$ to fill the occluded voxel at a certain viewpoint with the average color of all the non-occluded views for the same point. The same operation is performed to create a "RGB main" triplet, suitable for renderers that do not provide support for representing plenoptic colors.

C. Rendering of the Plenoptic Point Clouds

One of the main challenges when comparing implicit (top branch in Fig. 1) and explicit (bottom branch in Fig. 1) scene representations in this work lies in the differences in the underlying modeling of the 3D scene and their respective rendering schemes. NeRF-like solutions have a rendering scheme based on ray-marching via the Volume Rendering equation, $\hat{C}(\mathbf{r}) = \sum_{i=1}^{N} T_i(1 - exp(-\sigma_i\delta_i))\mathbf{c}_i$, with $T_i = exp(\sum_{j=1}^{i-1} \sigma_j\delta_j)$ and $\delta_i = t_{i+1} - t_i$ being the step-size. On the other hand, the explicit models via PPCs are rendered by projecting to a viewpoint of interest the occupied voxels that are visible in that viewing direction. The renderer block for PPCs in Fig. 1 can be done using the MPEG's Software Renderer for PCs [8], with support for the generation of new views. Nevertheless, given the context of assessing the encoding capability of these methods, we obtain the rendered views by directly projecting the points of the PPC over each of the training viewpoints, in the same projection scheme explained in Sec. II-B. In this work, we refer to this method as "Direct rendering".

Although both the rendered views $\mathbf{V}$ generated as ground-truth (GT) and the $\mathbf{V}''$ generated by the PPCs adopt an explicit representation, the fact that the geometry from the GT is sampled and voxelized from a mesh introduces errors to the PPC generation framework even before performing

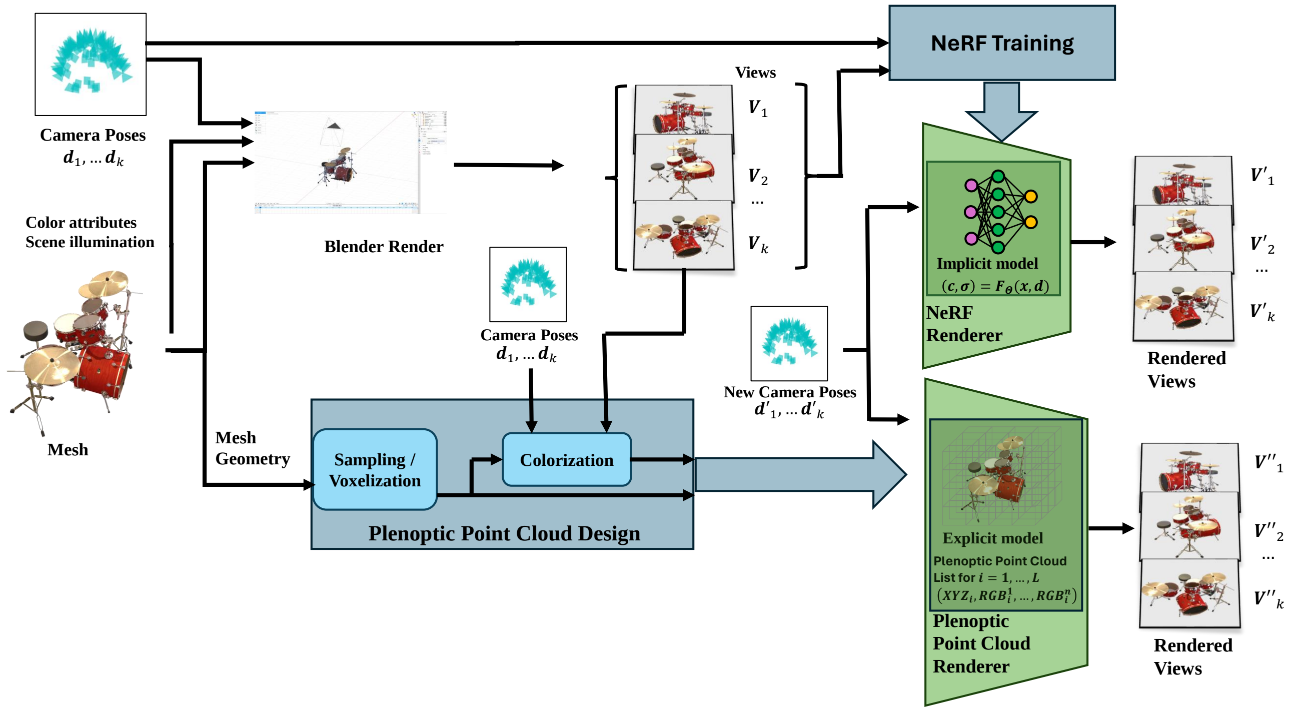

Fig. 1: Block diagram illustrating the framework for plenoptic representation for both the implicit (top branch) and explicit (bottom branch) methods, including the pipeline for the generation of novel plenoptic point clouds. Both approaches can be evaluated according to a render-based comparison of their outputs for a set of views.

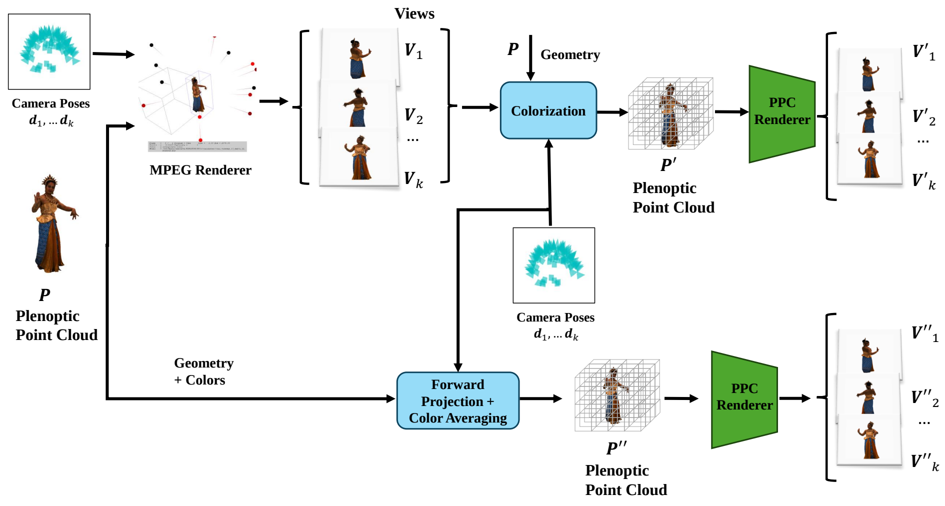

Fig. 2: Diagram illustrating the colorization as is done in our work (top branch) as opposed to a "best case" scenario, where both the geometry and the plenoptic colors for a given scene are known.

the Colorization block in the pipeline. Hence, we design an additional comparison pipeline in Fig. 2 to obtain a systematic assessment of the errors introduced by the colorization step, removing the geometry distortions by the mesh sampling and voxelization stage and additional image distortions caused by the usage of a different rendering algorithm (as it is between the set of images $\mathbf{V}$ and $\mathbf{V}''$ in Fig. 1).

The colorization at the top branch of Fig. 2 is exactly as it is done at the bottom branch of Fig. 1 and explained in Sec. II-B. The main distinction lies in the starting point used as a reference; while the GT rendered views from Fig. 1 have an underlying mesh that is rendered via Blender, the GT images in Fig. 2 are generated from a PPC $\mathbf{P}$ with the same geometry as our generated $\mathbf{P}'$ and rendered through MPEG's Software

Renderer. Therefore, the distortions between $\mathbf{V}$ and $\mathbf{V}'$ consist of the colorization step of the colorless geometry from $\mathbf{P}$ with its rendered views $\mathbf{V}$ and the difference between the rendering scheme, since $\mathbf{V}'$ are rendered via Direct rendering. Additionally, we create a "best-case" scenario in the bottom branch, where both the geometry and the plenoptic RGB triplets from the reference $\mathbf{P}$ are known. Here, the points are projected to the k virtual images in order to identify which points fall into the same pixel coordinates. The color for $\mathbf{P}''$ is obtained by assigning to each point the average of the GT colors of the points from $\mathbf{P}$ that are projected to the same virtual pixel. However, the averaging of the GT colors in the points of $\mathbf{P}$" works as an anti-aliasing filter.

D. Compression

PPCs have a linear increase in terms of size if one wishes to have a more dense representation in the angular domain. As a result, providing a solution to reduce the model's size without significant degradation becomes paramount. Among solutions that have been proposed are those where the colors are transformed via either a Karhunen-Loève (KLT) or a Discrete Cosine transform before being compressed with MPEG's standards for PC compression, the Video-based [9] and the Geometry-based [10] Point Cloud Compression standards.

Here, we apply the same KLT over the k YUV-converted attributes, but we use the Geometry-Based Solid Content Test Model (GeS-TM) platform to compress the models. The attributes are lossy compressed via the region adaptive hierarchical transform (RAHT), while the geometry is either losslessly encoded via octree quantization, or encoded with losses using the triangle soup (Trisoup) coding.

Additionally, we propose to use the occlusion mask $\mathbf{O}$ to achieve further bitrate savings. This "Occlusion Aware" approach encodes for each color triplet of the PPC only the colors of the non-occluded voxels at each viewpoint k. This is done by compressing via the GeS-TM k sub-PCs $\mathbf{B}$ of non-occluded points of the PPC $\mathbf{P}$, that is $\mathbf{B}_k = \{p \in \mathbf{P} \mid \mathbf{O}_k(p)\}$. Since the geometry is losslessly encoded, the geometry of each $\mathbf{B}_k$ does not need to be conveyed. Instead, $\mathbf{O}$ can be reconstructed on the decoder side if the k poses $\mathbf{d}$ and intrinsics $\mathbf{K}$ are transmitted. On the encoder side, since $\mathbf{O}$ was already generated during the colorization step, it does not need to be computed a second time.

E. Generation of the Dataset

The dataset used in this work for benchmarking is based on NeRF's Synthetic Blender Dataset [2], which provides the Blender files of the model and a training/testing/validation split of rendered images for 8 scenes. However, the generated views that were made available to the public were from projections over modified versions of the models, which could not be disclosed by the authors. As such, the precise GT geometry from which the images that are generally used for benchmarks in the SOTA is unknown, which does not comply with our pipeline in Fig. 1. Hence, we generate the same set of rendered views and splits of training, validation, and testing for the 8 synthetic scenes, but over the meshes of which we now have knowledge of their geometry. Consequently, the models that we use for benchmarking on the implicit representation branch were all re-trained over these newly rendered images. The re-generated scenes are available on the project page, at https://drcfts.github.io/c-ppc/.

III. Experiments

We use in our experiments a regenerated version (see Sec. II-E) of the *NeRF-Synthetic 360* – also known as Blender – dataset from the original NeRF [2], which contains 8 sequences with training sets of 100 images with dimensions of 800 × 800 each. For the state-of-the-art comparisons, we compare our explicit model against implicit ones. For rendering quality benchmark, our solution is compared against the following: NeRF, a modified version of Plenoctrees aimed for compression [11] – which we refer to here as C-PO –, the 3-Dimensional Gaussian splatting (3DGS) [12]. For the compression efficiency evaluation, the compression version of C-PO and the Compressed 3D Gaussian Splatting (C-3DGS) [13] were assessed.

A. Colorization Block Analysis

We assess the results from the PPC generation schemes from Fig. 2, where the main source of distortions of the top branch (Branch 1) stem from the colorization block since the GT images are rendered from a model with the exact same geometry and rendering pipeline. On the other hand, the bottom branch (Branch 2) provides a "best-case" scenario where the aliasing from the projection of the voxels on the pixels can be reduced by averaging the true points from the colors that are projected into the same pixel.

Table I presents the comparisons of the 2 branches over the "thaidancer" sequence from the PPC dataset of 8i [5]. For Branch 1, we compare the results of colorizing the new PC P' from images of dimensions 800×800 and 2000×2000. This increase in the image resolution results in a gain of almost 4 dB, which indicates that the size of the voxels is too high for the given image resolution. Results for Branch 2 with a virtual image of resolution 800×800 show that the anti-aliasing from the color averaging reduces these artifacts almost to the quality level of Branch 1 with 2000×2000 resolution. Nonetheless, since this kind of knowledge from the GT is not available in the pipeline at the time of projection, our approach against aliasing consists of super-resolving the image to a higher resolution. This is done via a $8\times$ bicubic upsampling of the 800 resolution image. The results, as seen in Table I, provide marginal gains over the low-resolution version.

B. Analysis of the different sampling methods

Results from Table II show the average over 20 frames from the training set of "drums" between the three different point sampling methods from Sec. II-A. For all methods, the quality from the rendered views increases significantly from resolution 9 to 10, which is explained by the presence of holes due to the bigger average distance between points. On the other hand, choosing resolution 11 instead of 10 provides only

TABLE I: Projection Experiments for sequence "thaidancer", evaluating the color projection between the top (Branch 1) and bottom (Branch 2) approaches from Fig. 2 of projecting plenoptic colors into the colorless PC.

Method	Resolution	PSNR ↑	SSIM ↑	LPIPS ↓
Branch 1	800	33.62	0.972	0.011
	2000	37.35	0.990	0.006
	6400 (SR)	34.59	0.979	0.010
Branch 2	800	37.05	0.990	0.009
	2000	38.50	0.994	0.006

TABLE II: Comparison for different sampling methods for different resolutions over 20 training frames of the "drums".

Sampling Method	Depth	Metric			
		PSNR ↑	SSIM ↑	LPIPS ↓	Number of Points ↓
Triangle Split	9	**24.48**	**0.934**	**0.086**	738758
	10	**36.03**	**0.992**	**0.019**	3098013
	11	**36.1**1	**0.993**	0.019	12264813
Poisson Disk [6]	9	24.02	0.930	0.091	697079
	10	35.64	0.991	0.022	2524045
	11	35.52	0.991	**0.017**	**4028562**
Area Based [7]	9	23.98	0.930	0.091	**686846**
	10	31.36	0.978	0.028	**2448258**
	11	35.16	0.991	0.019	4927955

marginal gains at the cost of an increased number of points, which is undesirable from a compression standpoint. Thus, the PCs in our experiments are generated with the "Poisson-Disk" approach for a bit depth of 10 for the resolution, due to the combination of rendering quality and model size.

C. Explicit vs Implicit Scene Representation Benchmark

1) Rendering Quality: Table III shows the quality metrics for each of the sequences of the *NeRF Synthetic-360* dataset for the average over the 100 training views. One can observe how, in an encoding context, the evaluation of the training views favors the usage of the direct rendering of the PPCs over the views rendered by taking the PPC's closest color with MPEG's Software Renderer. This is due to the software's features for splatting the points when rendering the images, which causes additional distortions to the final rendered pixels when compared to the direct projection as is done in the colorization block of the pipeline.

Moreover, the PPCs rendered via direct rendering outperform the ones in 3DGS on average for all quality metrics, in particular, due to scenes that exhibit higher specularity, such as the reflections in "drums", "materials" and "ship", and transparencies as in "drums" and "ship". Scenes such as "ficus" also contain a certain degree of specularity in addition to regions with fine geometry. Nevertheless, the difference between the uncompressed sizes of the PPCs and 3DGS models (800 MB vs 70MB, respectively) foment an assessment of both approaches taking into account their sizes.

2) Rate-Distortion-Based Comparison: Table IV shows the BD-PSNR results for the average of 100 training views over all dataset's scenes. The compressed Plenoctrees method (C-PO) was used as the anchor. The distortion points in C-3DGS were generated by varying the densification threshold

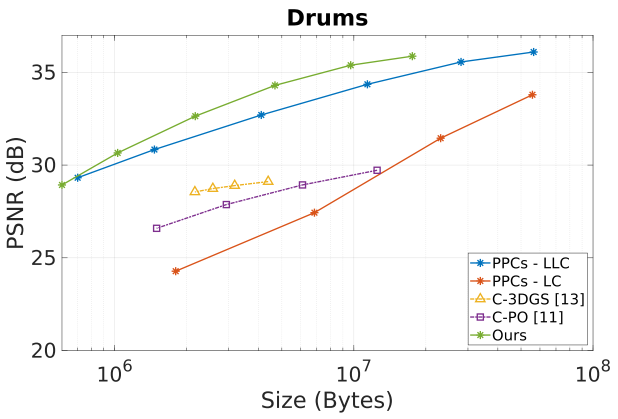

Fig. 3: RD results for "*drums*" in terms of PSNR (dB) vs the size (in Bytes) comparing our occlusion-aware compression scheme to PPC-based solutions, C-3DGS [13] and C-PO. [11]

on the gradient for the 2D position of the uncompressed 3DGS model, which defaults to 2.10^{-5}. The additional rates were generated with values of $\{3, 4, 5\}.10^{-5}$ before compression. For the explicit representations, two curves are obtained. While the attributes are encoded lossily by applying RAHT over the KLT-transformed coefficients in YUV space, the geometry is encoded either losslessly via octree quantization (LLC), or with losses with Trisoup from MPEG's GeS v5.0 (LC). Encoding parameters are chosen according to the Test Model's common test conditions. Thus, for the attribute coding in LLC, the QPs used were $\{22, 28, 34, 40, 46, 51\}$. We also assess our proposed "Occlusion Aware" solution, encoding the geometry losslessly and compressing lossily only the colors from non-occluded points via GeS' RAHT.

Table IV shows the reduction in bitrate of the "Occlusion Aware" scheme, as it gives a 1.9 dB advantage in BD-PSNR when compared to sending all the colors transformed via KLT, as in LLC. Average results reveal a favorable performance on average for the C-3DGS, although the more specular scenes, such as "drums", display an advantage of the plenoptic point cloud representations, which is illustrated in Fig. 3. These plots show a reduction from 40 to 800 times of our compressed models relative to their uncompressed counterparts.

The visualization for a training view for "ship" in Fig. 4 also displays the discrepancies in representing reflections between the 3DGS (Fig. 4 - b) and PPCs (Fig. 4 - c). When compressing the PPC with a QP value of 34 (Fig. 4), the representation of the specular highlights still remains on the scene.

IV. Conclusion

We propose a method to generate explicit plenoptic representations through PPCs in a systematic manner, as well as an approach to assess these structures against implicit methods, such as the NeRF-like approaches. Additionally, we also propose an encoding scheme that leverages the occlusion of the PPC's points for the training views to compress the attributes. Benchmarks regarding the rendering quality suggest that the direct rendering with PPCs outperforms the implicit

TABLE III: Benchmark comparison to the PPC-based solutions against the state of the art for implicit methods over the average of the 100 frames from the training set of the sequences from the Synthetic Blender Dataset.

Method		*chair*	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	Average
NeRF [2]	PSNR ↑	32.90	27.30	32.59	37.74	33.66	32.53	33.72	29.21	32.46
	SSIM ↑	0.971	0.937	0.977	0.981	0.971	0.967	0.984	0.849	0.955
	LPIPS ↓	0.041	0.100	0.048	0.047	0.046	0.053	0.028	0.191	0.069
C-PO [11]	PSNR ↑	37.48	31.22	36.50	41.14	37.01	36.78	37.49	33.10	36.34
	SSIM ↑	0.989	0.971	0.989	0.990	0.987	0.986	0.994	0.936	0.980
	LPIPS ↓	0.015	0.047	0.018	0.025	0.018	0.028	0.009	0.097	0.032
3DGS [12]	PSNR ↑	**38.49**	29.49	38.12	**41.73**	**39.71**	35.99	**39.07**	33.27	36.98
	SSIM ↑	**0.992**	0.978	0.995	0.991	**0.992**	0.989	**0.996**	0.917	0.981
	LPIPS ↓	**0.009**	0.025	0.006	**0.015**	**0.009**	0.016	**0.003**	0.107	0.024
PPCs - MPEG Rendering	PSNR ↑	32.17	31.79	35.67	37.21	32.03	35.78	33.75	30.78	33.65
	SSIM ↑	0.982	0.986	0.992	**0.996**	0.971	0.991	0.988	0.949	0.982
	LPIPS ↓	0.036	0.026	0.018	0.021	0.064	0.024	0.012	0.093	0.037
PPCs - Direct Rendering	PSNR ↑	36.11	**36.53**	**44.08**	40.95	37.76	**39.43**	37.85	**35.18**	**38.49**
	SSIM ↑	0.989	**0.994**	**0.998**	0.994	0.991	**0.995**	**0.996**	**0.978**	**0.992**
	LPIPS ↓	0.020	**0.014**	**0.004**	**0.015**	0.035	**0.015**	0.007	**0.038**	**0.019**

TABLE IV: Benchmark, in BD-PSNR (in dB), for the PPC-based solutions against the state-of-the-art implicit methods over the average of the training frames from the sequences from the Synthetic Blender Dataset. C-PO [11] was used as anchor.

Method	**BD-PSNR (dB)**								
	chair	*drums*	*ficus*	*hotdog*	*lego*	*materials*	*mic*	*ship*	Average
C-3DGS [13]	**7.03**	0.92	8.67	**8.67**	**6.95**	4.30	**8.37**	**5.20**	**6.26**
PPCs - LC	-1.35	-2.01	2.21	0.69	-3.49	0.50	0.03	2.18	-0.16
PPCs - LLC	1.40	4.41	7.17	4.10	0.96	4.11	4.12	3.45	3.72
Occl. Aware (Ours)	2.30	**5.79**	**9.77**	5.69	3.40	**7.25**	6.15	4.46	5.60

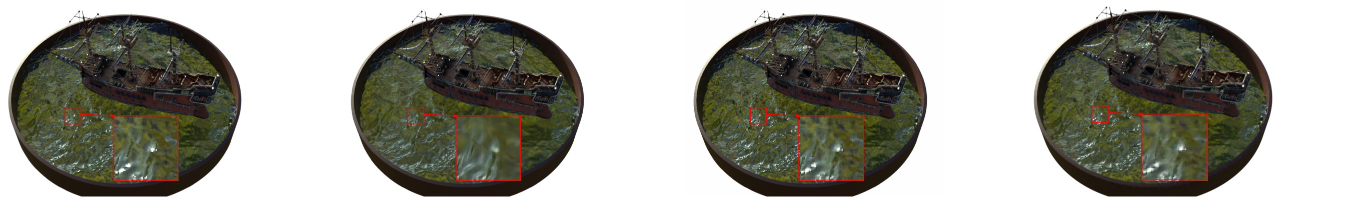
(a) Ground truth (b) 3DGS [12] (29.61 dB) (c) PPCs (34.64 dB) (d) Ours - QP = 34 (31.49 dB)

Fig. 4: Training view of *ship* from (b) 3DGS [12], (c) uncompressed PPC, (d) (Ours) compressed PPC with QP = 34 (attributes).

3DGS representation in PSNR, SSIM, and LPIPS, with a PSNR gain of 1.51 dB on the training views of the NeRF-Synthetic 360 dataset. For scenes with a higher degree of specularity, the gains can reach up to 7.04 dB. Nonetheless, the compression efficiency assessment shows an advantage of 0.7 dB in BD-PSNR for a compressed version of 3DGS over the compressed PPCs, although our compression scheme gives a 1.9 dB advantage over existing compression schemes for PPCs and a size reduction from up to 800 times relative to the uncompressed models.

References

[1] M. Landy and J. A. Movshon, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, pp. 3–20. 1991.

[2] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J. Frahm, Eds., Cham, 2020, pp. 405–421, Springer International Publishing.

[3] G. Sandri, R. L. de Queiroz, and P. A. Chou, "Compression of Plenoptic Point Clouds," *IEEE Transactions on Image Processing*, vol. 28, no. 3, pp. 1419–1427, 2019.

[4] R. Higa et al., "Multi-Attribute Implementation for Multiple Colors Per Point in the TMC2," ISO/IEC JTC1/SC29 Joint WG11/WG7 (MPEG/JPEG), document M55144, Oct. 2020.

[5] M. Krivokuća, P. A. Chou, and P. Savill, "8i Voxelized Surface Light Field (8iVSLF) Dataset," ISO/IEC JTC1/SC29 Joint WG11/WG1 (MPEG/JPEG), input document M42914, Jul. 2018.

[6] M. Corsini, P. Cignoni, and R. Scopigno, "Efficient and flexible sampling with blue noise properties of triangular meshes," *IEEE Transactions on Visualization and Computer Graphics*, vol. 18, no. 6, pp. 914–924, 2012.

[7] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: Measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, pp. 167 – 174, 06 1998.

[8] J.E. Marvie O. Mocquard J, Ricard, "[3dg] mpeg-pcc-renderer new functionalities," doc. m59060, ISO/IEC MPEG JTC 1/SC 29/WG 11, Online, January 2022.

[9] D. C. Garcia, C. Dorea, R. U. B. Ferreira, D. R. Freitas, R. L. de Queiroz, R. Higa, I. Seidel, and V. Testoni, "Differential transform for video-based plenoptic point cloud coding," *IEEE Transactions on Image Processing*, vol. 31, pp. 1994–2003, 2022.

[10] D.R. Freitas, G.L. Sandri, and R.L. De Queiroz, "Geometry-based compression of plenoptic point clouds," in *2022 IEEE 24th MMSP*, 2022, pp. 1–5.

[11] D. R Freitas, C. Guillemot, and I. Tabus, "Compression of plenoctree model attributes enabling fast communication and rendering of neural radiance fields," in *2023 31st European Signal Processing Conference (EUSIPCO)*. IEEE, 2023, pp. 590–594.

[12] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, 2023.

[13] S. Niedermayr, J. Stumpfegger, and R. Westermann, "Compressed 3d gaussian splatting for accelerated novel view synthesis," *arXiv preprint arXiv:2401.02436*, 2023.