Neural Mesh-Based Graphics

Shubhendu Jena, Franck Multon, Adnane Boukhayma

Neural Mesh-Based Graphics

Shubhendu Jena, Franck Multon, Adnane Boukhayma

Inria, Univ. Rennes, CNRS, IRISA, M2S, France

Abstract. We revisit NPBG [2], the popular approach to novel view synthesis that introduced the ubiquitous point feature neural rendering paradigm. We are interested in particular in data-efficient learning with fast view synthesis. We achieve this through a view-dependent mesh-based denser point descriptor rasterization, in addition to a foreground/background scene rendering split, and an improved loss. By training solely on a single scene, we outperform NPBG [2], which has been trained on ScanNet [9] and then scene finetuned. We also perform competitively with respect to the state-of-the-art method SVS [42], which has been trained on the full dataset (DTU [1] and Tanks and Temples [22]) and then scene finetuned, in spite of their deeper neural renderer.

1 Introduction

Enabling machines to understand and reason about 3D shape and appearance is a long standing goal of computer vision and machine learning, with implications in numerous 3D vision downstream tasks. In this respect, novel view synthesis is a prominent computer vision and graphics problem with rising applications in free viewpoint, virtual reality, image editing and manipulation, as well as being a corner stone of building an efficient metaverse. The introduction of deep learning in the area of novel view synthesis brought higher robustness and generalization in comparison to earlier traditional approaches. While the current trend is learning neural radiance fields (e.g. [29, 56, 61]), training and rendering such implicit volumetric models still presents computational challenges, despite recent efforts towards alleviating these burdens (e.g. [17, 25, 49, 62]). An appealing alternate learning strategy [2, 41, 42, 53], achieving to date state-of-the-art results on large outdoors unbounded scenes such as the Tanks and Temples dataset [22], consists of using a pre-computed geometric representation of the scene to guide the novel view synthesis process. As contemporary successors to the original depth warping techniques, these methods benefit from a strong geometry prior to constrain the learning problem, and recast its 3D complexity into a simpler 2D neural rendering task, providing concurrently faster feed-forward inference.

Among the latter, NPBG [2] is a popular strategy, being core to several other neural rendering based methods (e.g. [35, 37, 60, 64]). Learnable descriptors are appended to the geometry points, and synthesis consists of rasterizing then neural rendering these features. It is practical also as it uses a lightweight and relatively simpler architecture, compared to competing methods (e.g. [41, 42]).

Our motivation is seeking a data-efficient, fast, and relatively lightweight novel-view synthesis method. Hence we build on the idea of NPBG [2], and we introduce several improvements allowing it to scale to our aforementioned goals. In particular, and differently from NPBG [2]: We introduce denser rasterized feature maps through a combination of denser point rasterization and face rasterization; We enforce view-dependency explicitly through anisotropic point descriptors; As the foreground and background geometries differ noticeably in quality, we propose to process these two feature domains separately to accommodate independently for their respective properties; Finally, we explore a self-supervised loss promoting photo-realism and generalization outside the training view corpus. The improvement brought by each of these components is showcased in Table 3.

By training simply on a single scene, our method outperforms NPBG [2], even though it has been additionally trained on ScanNet [9] and then further fine-tuned on the same scene, in terms of PSNR, SSIM [58] and LPIPS [66], and both on the Tanks and Temples [22] (Tab. 1) and DTU [1] (Tab. 2) datasets. The performance gap is considerably larger on DTU [1]. Our data efficiency is also illustrated in the comparison to state-of-the-art method SVS [42] (Tab. 1 2). We achieve competitive numbers despite their full dataset trainings, their deeper convolutional network based neural renderers, and slower inference. We also recover from some of their common visual artifacts as shown in Figure 6.

2 Related Work

While there is a substantial body of work on the subject of novel view synthesis, we review here work we deemed most relevant to the context of our contribution.

Novel view synthesis. The task of novel view synthesis essentially involves using observed images of a scene and the corresponding camera parameters to render images from novel unobserved viewpoints. This is a long explored problem, with early non deep-learning based methods [7, 11, 15, 24, 46, 69] using a set of input images to synthesize new views of a scene. However, these methods impose restrictions on the configuration of input views. For example, [24] requires a dense and regularly spaced camera grid while [10] requires that the cameras are located approximately on the surface of a sphere and the object is centered. To deal with these restrictions, unstructured view synthesis methods use a 3D proxy geometry of the scene to guide the view synthesis process [5, 23]. With the rise of deep-learning, it has also come to be used extensively for view synthesis, either by blending input images to synthesize the target views [16, 54], or by learning implicit neural radiance fields followed by volumetric rendering to generate the target views [29], or even by using a 3D proxy geometry representation of the scene to construct neural scene representations [2, 32, 41, 42, 53].

View synthesis w/o geometric proxies. Early deep-learning based approaches combine warped or unwarped input images by predicting the corresponding blending weights to compose the target view [8, 54]. Thereafter, several approaches came up leveraging different avenues such as predicting camera poses [67], depth maps [19], multi-plane images [12, 68], and voxel grids [20]. Recently, implicit neural radiance fields (NeRF) [29] has emerged as a powerful representation for novel view synthesis. It uses MLPs to map spatial points to volume density and view-dependent colors. Hierarchical volumetric rendering is then performed on the predicted point colors to render the target image. Some of the problems associated with NeRF [29] include higher computational cost and time of rendering complexity, the requirement of dense training views, the lack of across-scene generalization, and the need for test-time optimization. A number of works [3, 13, 18, 26, 30, 40, 59, 62, 63] have tried addressing these limitations. In particular, Spherical Harmonics [51] have been used to speed up inference of Neural Radiance Fields by factorizing the view-dependent appearance [62]. Nex [59] introduced a related idea, where several basis functions such as Hemispherical harmonics, Fourier Series, and Jacobi Spherical Harmonics were investigated, and concluded that learnable basis functions offer the best results. Some other methods, like pixelNeRF [63], GRF [55], IBRNet [57] and MVSNeRF [6] proposed to augment NeRFs [29] with 2D and 3D convolutional features collected from the input images. Hence, they offer forward pass prediction models, i.e. test-time optimization free, while introducing generalization across scenes. While these methods are promising, they need to train on full datasets to generalize well, while at the same time evaluating hundreds of 3D query points per ray for inference similar to NeRF [29]. Hence, both training and inference often takes quite long for these methods. We note that besides encoders [6, 55, 57, 63], implicit neural representations can also be conditioned through meta-learning e.g. [33, 48, 50].

View synthesis using geometric proxies. Different from the work we have discussed so far, several recent methods utilize a geometric reconstruction of the scene to construct neural scene representations and consequently use them to synthesize target views. These geometric proxies can either be meshes [41, 42, 53] or point clouds [2, 27, 34, 52]. SVS [42] and FVS [41] utilize COLMAP [44, 45] to construct a mesh scaffold which is then used to select input views with maximum overlap with the target view. FVS [41] then uses a sequential network based on gated recurrent units (GRUs) to blend the encoded input images. SVS [42] operates on a similar principle except that the geometric mesh scaffold is used for on-surface feature aggregation, which entails processing or aggregating directional feature vectors from encoded input images at each 3D point to produce a new feature vector for a ray that maps this point into the new target view. Deferred neural rendering (DNR) [53] proposes to learn neural textures encoding the point plenoptic function at different surface points alongside a neural rendering convolutional network. It is infeasible for very large meshes due to the requirement of UV-parametrization. On the other hand, NPBG [2] operates on a similar idea by learning neural descriptors of surface elements jointly with the rendering network, but uses point-based geometry representation instead of

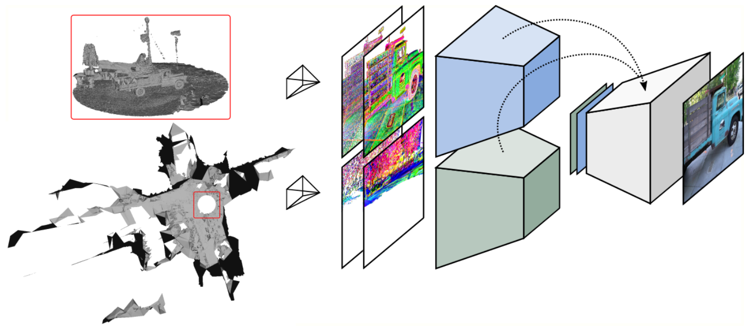


Fig. 1: Overview: An automatic split of the scene geometry is used to rasterize foreground/background mesh-borne view-dependent features, through both point based and mesh based rasterizations. A convolutional U-Net [43] maps the feature images into the target image.

meshes. Similarly, [27, 52] use COLMAP [44, 45] to reconstruct a colored point cloud of the scene in question, which is used alongside a neural renderer to render target images. Our method combines both approaches in the sense that it uses both point cloud and mesh representation of the scene as geometric proxies. Like NPBG [2], we also learn neural descriptors of surface elements, but since point clouds of large unbounded scenes are often sparse, using meshes helps in enhancing the richness of the rasterized neural descriptors. Hence, using both point clouds and meshes help us in achieving a balance between accuracy and density of the rasterized scene neural descriptors, which we will explain in detail in the upcoming sections and also through an ablative analysis.

3 Method

Given a set of calibrated images of a scene, our goal is to generate novel views from the same scene through deep learning. To this end, we design a forward pass prediction method, expected to generalize to target views including and beyond input view interpolation. Using a geometry proxy of the scene, we set view-dependent learnable descriptors on the vertices, and we split the scene automatically into a dense foreground and a sparser background. Each of these areas are rasterized through PyTorch3D's point based and mesh based rasterizations [39]. A convolutional neural renderer translates and combines the resulting image features into the target color image. Figure 1 illustrates this pipeline. Our method can be trained on a single scene by fitting the point descriptors and learning the neural renderer weights jointly. It can also benefit from multi-scene training through the mutualization of the neural renderer learning. We present in the remaining the different components of our method.

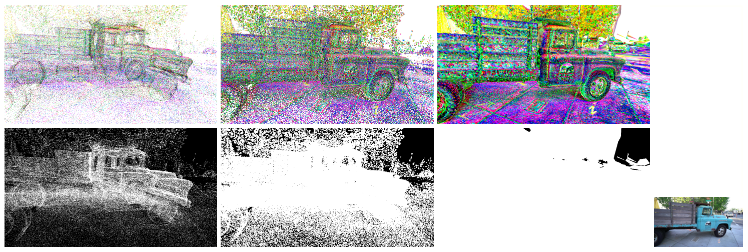


Fig. 2: We introduce denser feature images compared to NPBG [2]. Left: NPBG's rasterization. Center: Our point based rasterization. Right: Our mesh based rasterization. Top row shows feature PCA coefficients. Bottom row shows the resulting rasterization mask (Occupied/unoccupied pixels).

Preprocessing We need to obtain a geometry representing the scene from the training images as a preprocessing stage. Standard structure-from-motion (SfM) and multi-view stereo (MVS) pipelines can be used to achieve this [44, 45]. In this respect, we chose to use the preprocessed data from SVS [42] and FVS [41], and so the preprocessing steps are identical to these methods. The first step involves running structure-from-motion [44, 45] on the images to get camera intrinsics $\{K\}$ and camera poses as rotation matrices $\{R\}$, and translations $\{T\}$. The second step involves running multi-view stereo on the posed images, to obtain per-image depth maps, and then fusing these into a point cloud. Finally, Delaunay-based triangulation is applied to the obtained point cloud to get a 3D surface mesh $\mathcal{M}$. These steps are implemented following COLMAP [44, 45].

3.1 Dual Feature Rasterization

Given a target camera pose $R \in SO(3)$, $T \in \mathbb{R}^3$, the rendering procedure starts with the rasterization of learnable mesh features into the target image domain. The mesh $\mathcal{M}$ consists here of vertices (i.e. points) $\mathcal{P} = \{p_1, p_2, ..., p_N\}$ with neural point descriptors $\mathcal{K} = \{k_1, k_2, ..., k_N\}$, and faces $\mathcal{F} = \{f_1, f_2, ..., f_M\}$.

We noticed initially that having denser feature images improves the performance of the neural rendering. Hence, differently from NPBG [2], we use the PyTorch3D [39] renderer which allows us to obtain denser feature images for point cloud based rasterization. Furthermore, we notice additionally that using PyTorch3D's mesh based rasterization provides even denser feature images (c.f. Fig.2). Hence, we propose to rasterize the scene geometry features using both modes.

Point Cloud based Rasterization PyTorch3D [39] requires us setting a radius r in a normalized space which thresholds the distance between the target view pixel positions $\{(u, v)\}$ and the projected 3D points of the scene onto the target view, i.e. $\{\Pi(p) : p \in P\}$. If this distance is below the threshold for a given pixel (u, v), we consider this point to be a candidate for that pixel, and we finally pick the point $p_{u,v}$ with the smallest z coordinate in the camera coordinate frame. This writes:

$$\mathcal{P}_{u,v} = \{p \in \mathcal{P} : ||\Pi(p) - (u, v)||_2 \leq r\}. \tag{1}$$

$$p_{u,v} = \underset{p \in \mathcal{P}_{u,v}}{\operatorname{argmin}} \, p_z, \quad \text{where} \quad p = (p_x, p_y, p_z). \tag{2}$$

The neural descriptor of the chosen point $p_{u,v}$ is projected onto the corresponding pixel position to construct the rasterized feature image. We set a radius of $r = 0.006$ to achieve a balance between accuracy of the projected point positions and the density of the rasterized feature images. The feature descriptors for each pixel are weighed inversely with respect to the distance of the projected 3D point to the target pixel, which can be expressed as $w_{u,v} = (1 - ||\Pi(p_{u,v}) - (u, v)||_2^2)/r^2$ where $||\Pi(p_{u,v}) - (u, v)||_2 < r$. Finally the point feature image $\{k^{\text{pt}}_{u,v}\}$ can hence be expressed as follows:

$$k^{\text{pt}}_{u,v} = w_{u,v} \times \mathcal{K}(p_{u,v}), \tag{3}$$

where $\mathcal{K}(p_{u,v})$ is the neural descriptor of mesh vertex $p_{u,v}$.

Mesh based Rasterization On the other hand, the mesh rasterizer in PyTorch3D [39] finds the faces of the mesh intersecting each pixel ray and chooses the face with the nearest point of intersection in terms of the z-coordinate in camera coordinate space, which writes:

$$\mathcal{F}_{u,v} = \{f \in \mathcal{F} : (u, v) \in \Pi(f)\}. \tag{4}$$

$$f_{u,v} = \underset{f \in \mathcal{F}_{u,v}}{\operatorname{argmin}} \, \hat{f}_z, \quad \text{where} \quad \hat{f} = (\hat{f}_x, \hat{f}_y, \hat{f}_z). \tag{5}$$

$\hat{f}$ represents here the intersection between ray (u, v) and face f in camera coordinate frame. Finally, to find the feature corresponding to each pixel, the barycentric coordinates of the point of intersection $\hat{f}_{u,v}$ of the face $f_{u,v}$ with the corresponding pixel ray are used to interpolate the neural point descriptors over the face. By noting (p_i, p_j, p_k) as the vertices making up face $f_{u,v}$, the mesh feature image $\{k^{\text{mesh}}_{u,v}\}$ can be expressed as follows:

$$k^{\text{mesh}}_{u,v} = \alpha\mathcal{K}(p_i) + \beta\mathcal{K}(p_j) + \gamma\mathcal{K}(p_k), \tag{6}$$

where (α, β, γ) are the barycentric coordinates of $\hat{f}_{u,v}$ in $f_{u,v}$.

While we expect the point cloud rasterization to be more accurate, the mesh rasterization provides a denser feature image, albeit less accurate and fuzzier due to the dependence on the quality of the geometric triangulation being used. The final feature image combines the best of both worlds as it consists of the mesh and point feature rasterized images concatenated together, i.e. $k_{u,v} = \left[k^{\text{pt}}_{u,v}, k^{\text{mesh}}_{u,v}\right]$.

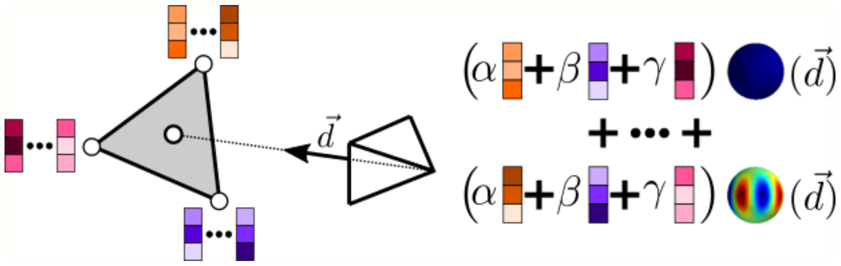

$\vec{d}$
$(\alpha \; + \beta \; + \gamma \;) \; (\vec{d})$
$+ \cdots +$
$(\alpha \; + \beta \; + \gamma \;) \; (\vec{d})$

Fig. 3: Differently from NPBG [2], we introduce mesh-based view-dependent feature rasterization. We learn per-point Spherical Harmonic coefficients, interpolated via the barycentric coordinates of the ray-face intersection.

3.2 Anisotropic Features

Novel view synthesis entails learning scene radiance functions as perceived appearance can vary according to the viewing angle. While the neural point descriptors in NPBG [2] are not view direction dependent per se, the neural rendering convolutional network could in theory model such view-dependency, even without taking the camera parameters of the target view explicitly as input. That is, the spatial disposition and neighborhood of these rasterized descriptors in image domain depends on the target view. However, incorporating view dependency in the geometry descriptors by design is bound to improve this aspect within such novel view synthesis strategy. Hence, we define anisotropic neural point descriptors in this work, and we implement this idea efficiently using Spherical Harmonics (SH). Spherical Harmonics have been long used as a popular low-dimensional representation for spherical functions to model e.g. Lambertian surfaces [4, 38] or even glossy surfaces [51]. They have been also recently introduced as a means of alleviating the computational complexity of volumetric rendering of implicit neural radiance fields (NeRFs) [62].

We adapt our point descriptors to auto-decode Spherical Harmonic coefficients rather than point features directly. Reformulating our earlier definition, a descriptor for a point (i.e. vertex) p can be expressed now as the set of coefficients:

$$\mathcal{K}(p) = \left(k_p^{l,m}\right)_{0 \leq l \leq l_{max}}^{-l \leq m \leq l}, \tag{7}$$

where $k^{l,m} \in \mathbb{R}^8$, 8 being the desired final feature dimension. We use 3 SH bands hence $l_{max} = 2$.

Evaluating a view dependent point feature consists of linearly combining the Spherical Harmonic basis functions $\Phi_l^m : [0, 2\pi]^2 \rightarrow \mathbb{R}$ evaluated at the viewing angle corresponding to a viewing direction $\vec{d}$. The rasterized point feature at pixel (u, v), as introduced in equation 3, can thus be finally expressed as:

$$k_{u,v}^{\text{pt}} = w_{u,v} \times \sum_l \sum_m k_{p_{u,v}}^{l,m} \Phi_l^m(\vec{d}). \tag{8}$$

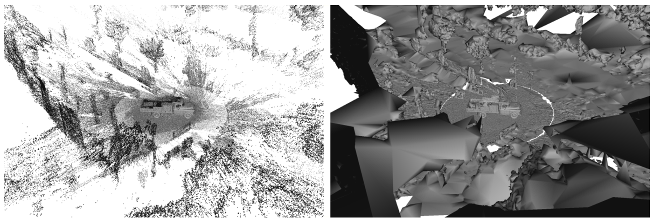


Fig. 4: COLMAP [44,45] geometry for scene "Truck" of Tanks and Temples [22]. Left: Point cloud. Right: Mesh.

We recall that in concordance with definitions in the previous section (3.1), $p_{u,v}$ is the point-rasterized vertex at location (u, v).

Similarly, the rasterized mesh feature at pixel (u, v), as introduced in equation 6, can be expressed as:

$$k_{u,v}^{\text{mesh}} = \sum_l \sum_m (\alpha k_{p_i}^{l,m} + \beta k_{p_j}^{l,m} + \gamma k_{p_k}^{l,m}) \Phi_l^m(\vec{d}). \tag{9}$$

Here again, (p_i, p_j, p_k) is the triangle rasterized at pixel (u, v), and α, β, γ are the barycentric coordinates of the ray intersection with that triangle. Figure 3 illustrates the former equation.

We note the the view direction $\vec{d}$ for a pixel (u, v) can be expressed as a function of the target camera parameters as follows:

$$\vec{d} = RK^{-1} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} + T. \tag{10}$$

The rasterized view-dependent feature images ($k_{u,v} = \left[k_{u,v}^{\text{pt}}, k_{u,v}^{\text{mesh}}\right]$) are subsequently fed to a U-Net [43] based convolutional renderer to obtain the final novel rendered images. We will be detailing upon this neural renderer next.

3.3 Split Neural Rendering

Upon observing the geometry obtained from running COLMAP [44, 45], especially on large unbounded scenes, such as the scenes in the Tanks and Temples dataset [22], the reconstruction is considerably more dense, detailed and precise for the main central area of interest where most cameras are pointing, as can be seen in Figure 4. The remaining of the scene geometry is sparse and less accurate. Hence, we argue that feature images rasterized from these foreground and background areas lie in two relatively separate domains, as the former is richer and more accurate than the latter.

As such, we propose to split the proxy geometry into a foreground and background sub-meshes (c.f. Figure 1). The split is automatically performed following

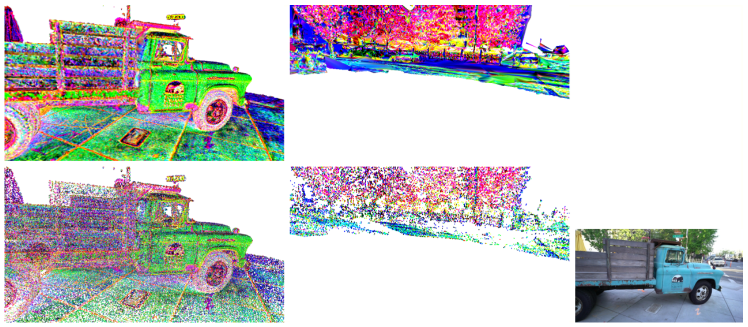


Fig. 5: Left: Foreground rasterization. Right: background rasterization. Top row: Mesh based rasterization. Bottom row: Point based rasterization. We visualize feature PCA coefficients.

NeRF++ [65]. Essentially, the center of the foreground sphere is approximated as the average camera position ($\bar{T}$), and its radius is set as 1.1 times the distance from this center to the furthest camera: $r_{fg} = \max(||T - \bar{T}||_2)$. We separately rasterize the features from both areas (c.f. Figure 5), and we process these foreground and background feature images with two separate yet identical encoders, focusing each on processing feature images from their respective domains. While there are 2 separate encoders, the features share a single decoder with skip connections coming in from both encoders. Other than these aspects, our neural renderer follows the multi-scale architecture in NPBG [2].

3.4 Hybrid Training Objective

We experiment with a mix of supervised and self-supervised losses for training our method: $L = L_{\text{rec}} + L_{\text{GAN}}$. The loss is used to perform gradient descent on the parameters θ of the neural renderer f_θ and the scene point descriptors $\mathcal{K}$ jointly.

Supervised loss The supervised loss follows the perceptual reconstruction loss in NPBG [2], where we urge the network to reproduce the available groundtruth images (I_{gt}) in feature space, i.e.:

$$L_{\text{rec}} = \sum_l ||\Psi_l(f_\theta(\{k_{u,v}\})) - \Psi_l(I_{gt})||_1. \tag{11}$$

$f_\theta(\{k_{u,v}\})$ is the output image from our network using rasterized features $\{k_{u,v}\}$. Ψ_l represents the l^{th} feature map from a pretrained VGG19 network [47], where $l \in \{$'conv1_2','conv2_2','conv3_2', 'conv4_2','conv5_2'$\}$.

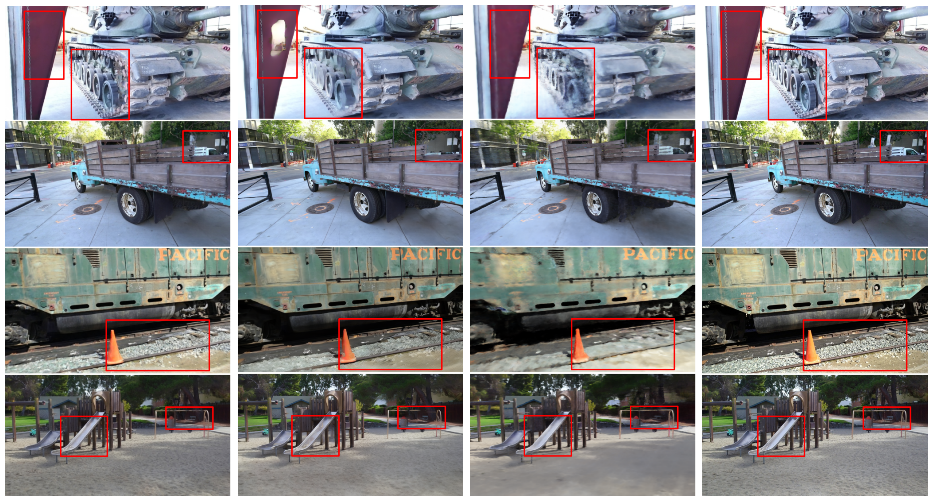
Unsupervised loss We introduce a GAN [14] loss to encourage the photorealism of the generated images and also improve generalization outside the training camera view-points. Specifically, with our entire model so far being the generator, we use a discriminator based on the DCGAN [36] model. Besides sampling from the training cameras, we additionally sample artificial views following RegNeRF [31]. We note that for these augmented views we do not have a target ground-truth image, and hence only the GAN loss is back-propagated. To obtain the sample space of camera matrices, we assume that all cameras roughly focus on a central scene point $\bar{T}$, as described in Section 3.3. The camera positions $\tilde{T}$ are sampled using a spherical coordinate system as follows: $\tilde{T} = \bar{T} + \tilde{r}[\sin\theta \sin\phi, \cos\theta, \sin\theta \cos\phi]$, where $\tilde{r}$ is sampled uniformly in $[0.6r_{fg}, r_{fg}]$, r_{fg} being the foreground radius as defined in Section 3.3. ϕ is sampled uniformly in $[0, 2\pi]$. θ is sampled uniformly around the mean training camera elevation within $\pm$ 1.5 its standard deviation. For a given camera position, the camera rotation is defined using the camera "look-at" function, using target point $\bar{T}$ and the up axis of the scene.

4 Results

Datasets To demonstrate the effectiveness of our approach for novel view synthesis in the context of large unbounded scenes, we choose the Tanks and Temples dataset [22]. It consists of images in Full HD resolution captured by a high-end video camera. COLMAP [44,45] is used to reconstruct the initial dense meshes/point clouds as well as to obtain the camera extrinsics and intrinsics for each scene. For quantitative evaluation, we follow the protocol in FVS [41] and SVS [42]. 17 of the 21 scenes are used for training and for the rest of the scenes, i.e., "Truck", "Train", "M60", and "Playground", the images are divided into a fine-tuning set and a test set. We also compare our method to prior approaches on the DTU [1] dataset, which consists of over 100 tabletop scenes, with the camera poses being identical for all scenes. DTU [1] scenes are captured with a regular camera layout, which includes either 49 or 64 images with a resolution of 1200×1600 and their corresponding camera poses, taken from an octant of a sphere. We use scenes 65, 106 and 118 for fine-tuning and testing purposes and the others are used for training. For scenes 65, 106 and 118, of the total number of images and their corresponding camera poses, 10 are selected as the testing set for novel view synthesis and the rest are used for fine-tuning.

Training procedures We perform two types of training. In full training, we follow FVS [41] and SVS [42] and jointly optimize the point neural descriptors of the training scenes along with the neural renderer. To test on a scene, we use the pretrained neural renderer and fine-tune it while learning the point descriptors of that scene, using the fine-tuning split. In single scene training, we perform the latter without pretraining the neural renderer.

For single scene trainings, we train our network for 100 epochs regardless of the scene. For full training, we pretrain our neural renderer for 15 epochs on all training scenes followed by fine-tuning the entire network for 100 epochs as before on specific scenes. For all our experiments, we use a batch size of 1 with



(a) Ours (b) SVS [42] (c) NeRF++ [65] (d) GT

Fig. 6: Qualitative comparison on Tanks and Temples [22].

Adam [21] optimizer, and learning rates of 10^{-1} and 10^{-4} for the point neural descriptors and the neural renderer respectively.

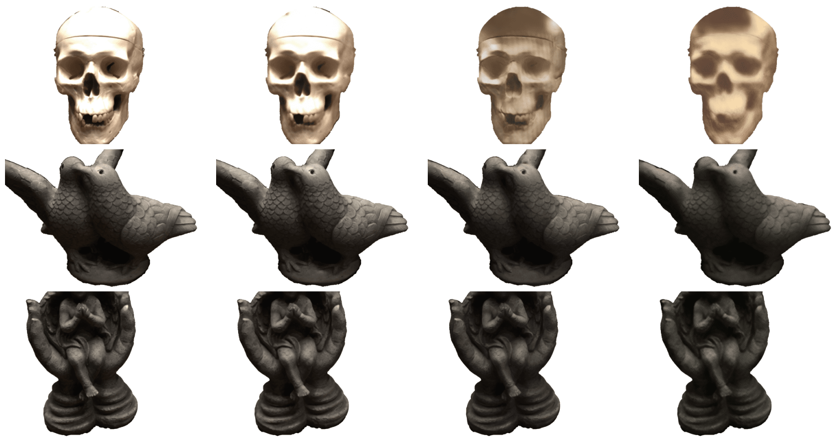
Methods	Truck			M60			Playground			Train		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
LLFF [28]	10.78	0.454	60.62	8.98	0.431	71.76	14.40	0.578	53.93	9.15	0.384	67.40
EVS [8]	14.22	0.527	43.52	7.41	0.354	75.71	14.72	0.568	46.85	10.54	0.378	67.62
NeRF [29]	20.85	0.738	50.74	16.86	0.701	60.89	21.55	0.759	52.19	16.64	0.627	64.64
NeRF++ [65]	22.77	0.814	30.04	18.49	0.747	43.06	22.93	0.806	38.70	17.77	0.681	47.75
FVS [41]	22.93	0.873	13.06	16.83	0.783	30.70	22.28	0.846	19.47	18.09	0.773	24.74
SVS [42]	23.86	0.895	9.34	19.97	0.833	20.45	23.72	0.884	14.22	18.69	0.820	15.73
NPBG [2]	21.88	0.877	15.04	12.35	0.716	35.57	23.03	0.876	16.65	18.08	0.801	25.48
Ours (Single)	23.88	0.883	17.41	19.34	0.810	24.13	23.38	0.865	23.34	17.35	0.788	23.66
Ours (Full)	24.03	0.888	16.84	19.54	0.815	23.15	23.59	0.870	22.72	17.78	0.799	24.17

Table 1: Quantitative comparison on Tanks and Temples [22]. Deeper shades represent better performance.

Metrics We report our performance for view synthesis, in line with previous seminal work, using three image fidelity metrics, namely Peak signal-to-noise ratio (PSNR), structural similarity (SSIM) [58] and learned perceptual image patch similarity (LPIPS) [66].

Quantitative comparison For our method, we show both single scene training (Ours (Single)) and full dataset training (Our (Full)) results. We relay the performances of methods [2, 8, 28, 29, 41, 42, 65] as reported in SVS [42].

Table 1 shows a quantitative comparison of our method with the recent state-of-the-art on Tanks and Temples [22]. Most methods underperform on these chal-



(a) Ours (Full) (b) Ours (Single) (c) NPBG [2] (Full) (d) NPBG [2] (Single)

Fig. 7: Qualitative comparison on DTU [1].

Methods	65			106			118		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
LLFF [28]	22.48/22.07	0.935/0.921	9.38/12.71	24.10/24.63	0.900/0.886	13.26/13.57	28.99/27.42	0.928/0.922	9.69/10.99
EVS [8]	23.26/14.43	0.942/0.848	7.94/22.11	20.21/11.15	0.902/0.743	14.91/29.57	23.35/12.06	0.928/0.793	10.84/25.01
NeRF [29]	32.00/28.12	0.984/0.963	3.04/8.54	34.45/30.66	0.975/0.957	7.02/10.14	37.36/31.66	0.985/0.967	4.18/6.92
FVS [41]	30.44/25.32	0.984/0.961	2.56/7.17	32.96/27.56	0.979/0.950	2.96/6.57	35.64/29.54	0.985/0.963	1.95/6.31
SVS [42]	32.13/26.82	0.986/0.964	1.70/5.61	34.30/30.64	0.983/0.965	1.93/3.69	37.27/31.44	0.988/0.967	1.30/4.26
NPBG [2]	16.74/15.44	0.889/0.873	14.30/19.45	19.62/20.26	0.847/0.842	18.90/21.13	23.81/24.14	0.867/0.879	15.22/16.88
Ours (Single)	26.78/20.85	0.957/0.925	9.64/12.91	29.98/25.40	0.931/0.909	12.75/13.70	31.43/26.52	0.946/0.931	11.73/11.13
Ours (Full)	28.98/22.90	0.970/0.943	7.15/11.13	30.67/25.75	0.939/0.917	12.10/13.10	32.39/27.97	0.956/0.941	10.62/10.07

Table 2: Quantitative comparison on DTU [1]. Left/Right: View interpolation/extrapolation. Deeper shades represent better performance.

lenging large unbounded scenes apart from NPBG [2], FVS [41] and SVS [42]. Although they could achieve promising results with only single scene training, methods based on volumetric neural rendering (NeRF [29] and NeRF++ [65]) are famously computationally expensive to train and render. Even by training on a single scene only, our method outperforms NPBG [2] in almost all scenes. We note that the neural renderer of NPBG [2] here was pretrained on ScanNet dataset [9]. Our method produces competitive results with respect to the best performing methods on this benchmark, i.e. FVS [41] and SVS [42]. It is interesting to observe in particular that with merely single scene training, our method outperforms the state-of-the-art SVS on scene "Truck", while coming as a close second in almost all other scenes in PSNR and SSIM [58]. FVS [41] and SVS [42] perform exceedingly well in all metrics but require training on the entire dataset, with a considerably larger training time than our single scene training. In particular, their lower LPIPS values could be attributed to the use of a considerably deeper neural renderer than ours, consisting of 9 consecutive U-Nets [43]. Ours is a much lighter single U-Net [43].

Table 2 reports quantitative comparison on the DTU [1] dataset. We use the view interpolation and extrapolation setting, as adopted by SVS [42] and

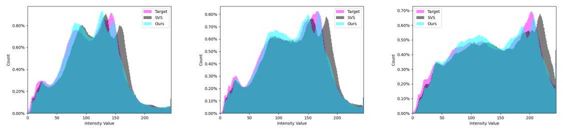


Fig. 8: Red, Blue and Green histograms for images in the "Truck" scene of Tanks and Temples [22] for our method, SVS [42] and the groundtruth.

FVS [41] for a fair comparison with other methods. The setting comprises of 6 central cameras to evaluate view interpolation and 4 corner cameras to evaluate view extrapolation. As has been observed from the experiments conducted in SVS [42], LLFF [28] and EVS [8] perform decently on DTU [1], while NeRF [29], FVS [41] and SVS [42] excel on it. NPBG [2], on the other hand, performs very poorly due to lack of data per scene (approximately 39 images) making the point feature autodecoding less efficient. Our method, despite using point feature autodecoding equally, gains considerably on NPBG [2] and performs relatively close to FVS [41]. Again, as observed for the Tanks and Temples [22] dataset, our method is able to rapidly train for a single scene and perform relatively well, and close to methods which have been trained and finetuned on the entire dataset such as FVS [41] and SVS [42], which makes it very practical in the sense that it is able to achieve reasonable results with limited training time and data.

Qualitative comparison Qualitative results on Tanks and Temples [22] of some of the competitive contemporary methods are summarized in Figure 6. In general, we notice that SVS [42] excels in the synthesis quality of the novel views. This is also reflected in the LPIPS [66] metric, where SVS [42] performs better than the competition. NeRF++ [65] on the other hand tends to underperform and produces blurs and artifacts, particularly if we look at the results of the "M60" and "Train" scenes.

Although SVS [42] performs well overall, especially in background regions, it tends to display an unnatural "smoothing effect" on the image in certain regions, such as the rocks indicated in the "Train" scene or in the track of the tank in the "M60" scene. We suspect that this is due to their neural renderer which contains 9 consecutive U-Nets [43], which might cause the view-dependent feature tensor to be oversmoothed. Furthermore, SVS [42] results sometimes contain major artifacts like holes, missing structures or transparent parts in certain regions, such as the ones indicated in the "M60", "Truck" and "Playground" scenes respectively in Figure 6. Another potential issue that we observed was an apparent color shift in some of the results of SVS [42]. To investigate this, we plot separate color histograms of the synthesized outputs of SVS [42], our method and the target images for the "Truck" scene in Figure 8. We notice in this figure that the histogram of the images synthesized by our method matches that of the target images more closely than those synthesized by SVS [42]. Overall, our method tends to display less of the color shifts and holes/artifacts that SVS [42] seems to exhibit despite lower scores with respect to the evaluation metrics, particularly for LPIPS [66].

Methods	Truck			M60			Playground			Train		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
Original NPBG	21.55	0.807	27.58	17.48	0.757	33.39	22.26	0.808	28.79	16.08	0.697	34.58
Bigger radius	22.26	0.832	23.55	18.76	0.784	28.73	22.61	0.821	26.62	16.15	0.717	30.89
With mesh	22.70	0.850	20.76	**19.95**	0.811	24.26	22.78	0.838	25.22	15.75	0.730	29.01
Dir. features	23.40	0.872	18.57	19.26	0.810	24.05	23.27	0.857	24.42	17.13	0.782	24.79
Split scene	23.64	0.880	17.99	19.03	**0.812**	**24.04**	23.32	0.865	**23.11**	**17.37**	0.787	24.13
Ours (Single)	**23.88**	**0.883**	**17.41**	19.34	0.810	24.13	**23.38**	**0.865**	23.34	17.35	**0.788**	**23.66**

Table 3: Quantitative ablation on Tanks and Temples [22]. Best/second best performances are emboldened/underlined respectively.

Next, we present some qualitative results on DTU [1] in Figure 7. We notice that compared to NPBG [2] which generates very poor quality novel views (both for single-scene finetuning and full training) presumably due to lack of adequate training data, our method offers way better visual results which we believe is due to a denser input feature image fed to the neural renderer, on account of point based rasterization with bigger radius and the mesh based rasterization. In fact, our method, with just single-scene finetuning performs better than NPBG [2] even when fully trained, which is also supported by the quantitative results presented in Table 2.

Ablation studies In this section, we conduct an ablative analysis to justify the choice of our final architecture. We ablate in the single-scene training scenario using all testing scenes of Tanks and Temples [22]. We progressively add components to our baseline architecture (i.e. NPBG [2]) until we reach our final model to demonstrate their individual contributions to our performance. The results are summarized in Table 3. "Original NPBG" is NPBG [2] in the single-scene setting. "Bigger radius" is the "Original NPBG" with a bigger rasterization radius as discussed in Section 3.1, as opposed to NPBG [2] which has a rasterization radius of half a pixel. "With mesh" includes the mesh rasterized and interpolated feature image mentioned in Section 3.1. These two previous components lead to denser feature images. "Directional features" incorporates view dependency in the geometry descriptors using Spherical Harmonics (SH) as discussed in Section 3.2, while "Split scene" splits the proxy geometry into foreground and background, rasterizes and encodes each features separately. Our final model uses an additional GAN [14] loss during training as described in Section 3.4. Overall throughout all scenes, the numbers witness the consistent improvement brought by the various components.

5 Conclusion

We improved in this work on the Neural Point Based Graphics (i.e. NPBG [2]) model for novel view synthesis, by providing a new data-efficient version that can achieve superior results by training solely on a single scene. SVS [42] still produces Superior LPIPS [66] synthesis performance. As future work, we will investigate and improve on this aspect of our method, while attempting to maintain memory and compute efficiency.

References

1. Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision **120**(2), 153–168 (2016) 1, 2, 10, 12, 13, 14
2. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In: European Conference on Computer Vision. pp. 696–712. Springer (2020) 1, 2, 3, 4, 5, 7, 9, 11, 12, 13, 14
3. Barron, J.T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., Srinivasan, P.P.: Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5855–5864 (2021) 3
4. Basri, R., Jacobs, D.W.: Lambertian reflectance and linear subspaces. IEEE transactions on pattern analysis and machine intelligence **25**(2), 218–233 (2003) 7
5. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proceedings of the 28th annual conference on Computer graphics and interactive techniques. pp. 425–432 (2001) 2
6. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021) 3
7. Chen, S.E., Williams, L.: View interpolation for image synthesis. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques. pp. 279–288 (1993) 2
8. Choi, I., Gallo, O., Troccoli, A., Kim, M.H., Kautz, J.: Extreme view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7781–7790 (2019) 3, 11, 12, 13
9. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5828–5839 (2017) 1, 2, 12
10. Davis, A., Levoy, M., Durand, F.: Unstructured light fields. In: Computer Graphics Forum. vol. 31, pp. 305–314. Wiley Online Library (2012) 2
11. Debevec, P.E., Taylor, C.J., Malik, J.: Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 11–20 (1996) 2
12. Flynn, J., Neulander, I., Philbin, J., Snavely, N.: Deepstereo: Learning to predict new views from the world's imagery. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 5515–5524 (2016) 3
13. Garbin, S.J., Kowalski, M., Johnson, M., Shotton, J., Valentin, J.: Fastnerf: High-fidelity neural rendering at 200fps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14346–14355 (2021) 3
14. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014) 9, 14
15. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 43–54 (1996) 2

16. Hedman, P., Philip, J., Price, T., Frahm, J.M., Drettakis, G., Brostow, G.: Deep blending for free-viewpoint image-based rendering. ACM Transactions on Graphics (TOG) **37**(6), 1–15 (2018) 2
17. Hedman, P., Srinivasan, P.P., Mildenhall, B., Barron, J.T., Debevec, P.: Baking neural radiance fields for real-time view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5875–5884 (2021) 1
18. Jain, A., Tancik, M., Abbeel, P.: Putting nerf on a diet: Semantically consistent few-shot view synthesis. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5885–5894 (2021) 3
19. Kalantari, N.K., Wang, T.C., Ramamoorthi, R.: Learning-based view synthesis for light field cameras. ACM Transactions on Graphics (TOG) **35**(6), 1–10 (2016) 3
20. Kar, A., Häne, C., Malik, J.: Learning a multi-view stereo machine. Advances in neural information processing systems **30** (2017) 3
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 11
22. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017) 1, 2, 8, 10, 11, 13, 14
23. Kopf, J., Cohen, M.F., Szeliski, R.: First-person hyper-lapse videos. ACM Transactions on Graphics (TOG) **33**(4), 1–10 (2014) 2
24. Levoy, M., Hanrahan, P.: Light field rendering. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 31–42 (1996) 2
25. Li, Q., Multon, F., Boukhayma, A.: Learning generalizable light field networks from few images. arXiv preprint arXiv:2207.11757 (2022) 1
26. Liu, L., Gu, J., Zaw Lin, K., Chua, T.S., Theobalt, C.: Neural sparse voxel fields. Advances in Neural Information Processing Systems **33**, 15651–15663 (2020) 3
27. Meshry, M., Goldman, D.B., Khamis, S., Hoppe, H., Pandey, R., Snavely, N., Martin-Brualla, R.: Neural rerendering in the wild. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6878–6887 (2019) 3, 4
28. Mildenhall, B., Srinivasan, P.P., Ortiz-Cayon, R., Kalantari, N.K., Ramamoorthi, R., Ng, R., Kar, A.: Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. ACM Transactions on Graphics (TOG) **38**(4), 1–14 (2019) 11, 12, 13
29. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. In: European conference on computer vision. pp. 405–421. Springer (2020) 1, 2, 3, 11, 12, 13
30. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. arXiv preprint arXiv:2112.00724 (2021) 3
31. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S., Geiger, A., Radwan, N.: Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5480–5490 (2022) 10
32. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3504–3515 (2020) 2

33. Ouasfi, A., Boukhayma, A.: Few'zero level set'-shot learning of shape signed distance functions in feature space. arXiv preprint arXiv:2207.04161 (2022) 3
34. Pittaluga, F., Koppal, S.J., Kang, S.B., Sinha, S.N.: Revealing scenes by inverting structure from motion reconstructions. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 145–154 (2019) 3
35. Prokudin, S., Black, M.J., Romero, J.: Smplpix: Neural avatars from 3d human models. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 1810–1819 (2021) 1
36. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 (2015) 10
37. Raj, A., Tanke, J., Hays, J., Vo, M., Stoll, C., Lassner, C.: Anr: Articulated neural rendering for virtual avatars. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 3722–3731 (June 2021) 1
38. Ramamoorthi, R., Hanrahan, P.: On the relationship between radiance and irradiance: determining the illumination from images of a convex lambertian object. JOSA A **18**(10), 2448–2459 (2001) 7
39. Ravi, N., Reizenstein, J., Novotny, D., Gordon, T., Lo, W.Y., Johnson, J., Gkioxari, G.: Accelerating 3d deep learning with pytorch3d. arXiv preprint arXiv:2007.08501 (2020) 4, 5, 6
40. Reiser, C., Peng, S., Liao, Y., Geiger, A.: Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14335–14345 (2021) 3
41. Riegler, G., Koltun, V.: Free view synthesis. In: European Conference on Computer Vision. pp. 623–640. Springer (2020) 1, 2, 3, 5, 10, 11, 12, 13
42. Riegler, G., Koltun, V.: Stable view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12216–12225 (2021) 1, 2, 3, 5, 10, 11, 12, 13, 14
43. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015) 4, 8, 12, 13
44. Schonberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4104–4113 (2016) 3, 4, 5, 8, 10
45. Schönberger, J.L., Zheng, E., Frahm, J.M., Pollefeys, M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision. pp. 501–518. Springer (2016) 3, 4, 5, 8, 10
46. Seitz, S.M., Dyer, C.R.: View morphing. In: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques. pp. 21–30 (1996) 2
47. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 9
48. Sitzmann, V., Chan, E.R., Tucker, R., Snavely, N., Wetzstein, G.: Metasdf: Meta-learning signed distance functions. In: NeurIPS (2020) 3
49. Sitzmann, V., Rezchikov, S., Freeman, B., Tenenbaum, J., Durand, F.: Light field networks: Neural scene representations with single-evaluation rendering. Advances in Neural Information Processing Systems **34**, 19313–19325 (2021) 1
50. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. Advances in Neural Information Processing Systems **32** (2019) 3

51. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In: Proceedings of the 29th annual conference on Computer graphics and interactive techniques. pp. 527–536 (2002) 3, 7
52. Song, Z., Chen, W., Campbell, D., Li, H.: Deep novel view synthesis from colored 3d point clouds. In: European Conference on Computer Vision. pp. 1–17. Springer (2020) 3, 4
53. Thies, J., Zollhöfer, M., Nießner, M.: Deferred neural rendering: Image synthesis using neural textures. ACM Transactions on Graphics (TOG) **38**(4), 1–12 (2019) 1, 2, 3
54. Thies, J., Zollhöfer, M., Theobalt, C., Stamminger, M., Nießner, M.: Ignor: Image-guided neural object rendering. arXiv preprint arXiv:1811.10720 (2018) 2, 3
55. Trevithick, A., Yang, B.: Grf: Learning a general radiance field for 3d representation and rendering. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15182–15192 (2021) 3
56. Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., Wang, W.: Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. arXiv preprint arXiv:2106.10689 (2021) 1
57. Wang, Q., Wang, Z., Genova, K., Srinivasan, P.P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: Ibrnet: Learning multi-view image-based rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4690–4699 (2021) 3
58. 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), 600–612 (2004) 2, 11, 12
59. Wizadwongsa, S., Phongthawee, P., Yenphraphai, J., Suwajanakorn, S.: Nex: Real-time view synthesis with neural basis expansion. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8534–8543 (2021) 3
60. Wu, M., Wang, Y., Hu, Q., Yu, J.: Multi-view neural human rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1682–1691 (2020) 1
61. Yariv, L., Gu, J., Kasten, Y., Lipman, Y.: Volume rendering of neural implicit surfaces. Advances in Neural Information Processing Systems **34**, 4805–4815 (2021) 1
62. Yu, A., Li, R., Tancik, M., Li, H., Ng, R., Kanazawa, A.: Plenoctrees for real-time rendering of neural radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5752–5761 (2021) 1, 3, 7
63. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021) 3
64. Zakharkin, I., Mazur, K., Grigorev, A., Lempitsky, V.: Point-based modeling of human clothing. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14718–14727 (2021) 1
65. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020) 9, 11, 12, 13
66. Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 586–595 (2018) 2, 11, 13, 14

67. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1851–1858 (2017) 3
68. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. arXiv preprint arXiv:1805.09817 (2018) 3
69. Zitnick, C.L., Kang, S.B., Uyttendaele, M., Winder, S., Szeliski, R.: High-quality video view interpolation using a layered representation. ACM transactions on graphics (TOG) **23**(3), 600–608 (2004) 2

Neural Mesh-Based Graphics
– Supplementary Material –

Shubhendu Jena, Franck Multon, Adnane Boukhayma

Inria, Univ. Rennes, CNRS, IRISA, M2S, France

1 Additional qualitative results

Results on Tanks and Temples [3]: Figures 1, 2, 3, 4 show additional novel view synthesis qualitative results of our method for more test views on the Tanks and Temples [3] dataset, not seen during training.

Results on DTU [1]: Figures 5, 6, 7 show additional novel view synthesis qualitative results of our method for more test views on the DTU [1] dataset, not seen during training.

2 Additional qualitative comparisons

Comparison on Tanks and Temples [3]: Figure 8 shows additional novel view synthesis qualitative result comparison between our method for more test views on the Tanks and Temples [3] dataset, not seen during training against SVS [4] and NeRF++ [5].

Comparison on DTU [1]: Figure 9 shows additional novel view synthesis qualitative result comparison between our method for more test views on the DTU [1] dataset, not seen during training against SVS [4] and NeRF++ [5].

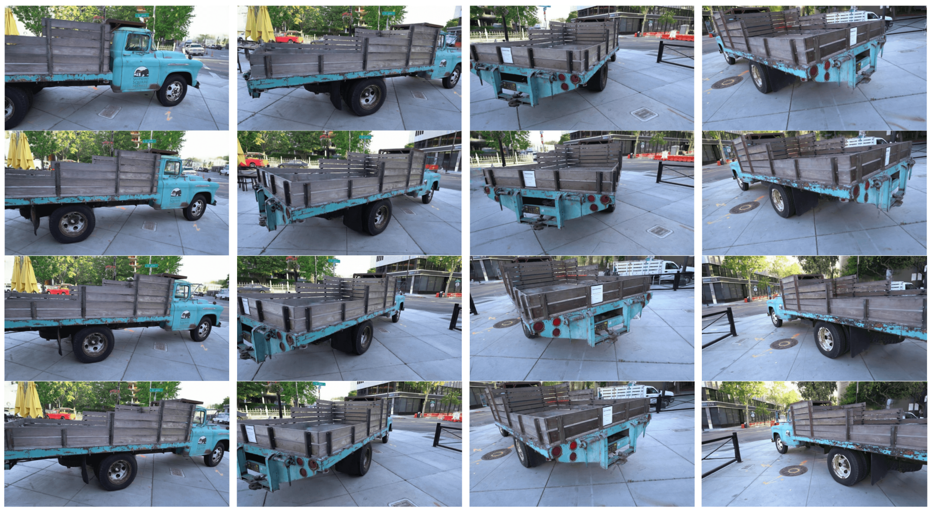


Fig. 1: Additional results on “Truck” scene of the Tanks and Temples [3] dataset.

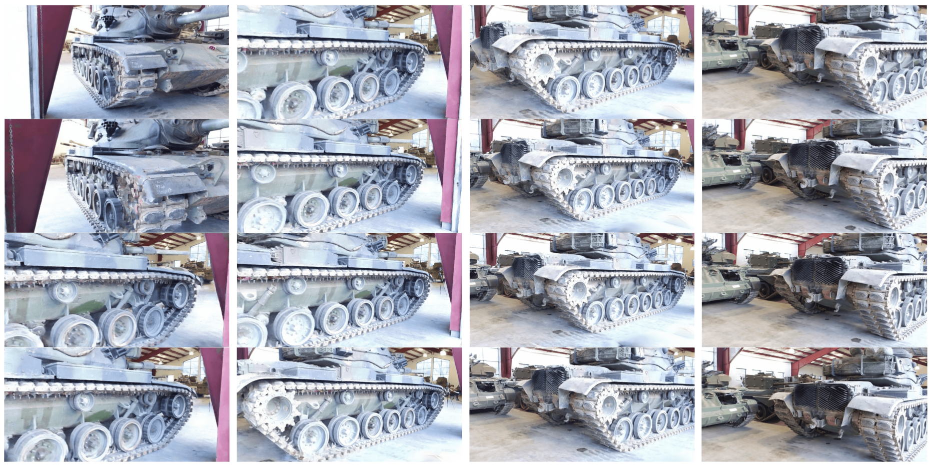


Fig. 2: Additional results on “M60” scene of the Tanks and Temples [3] dataset.

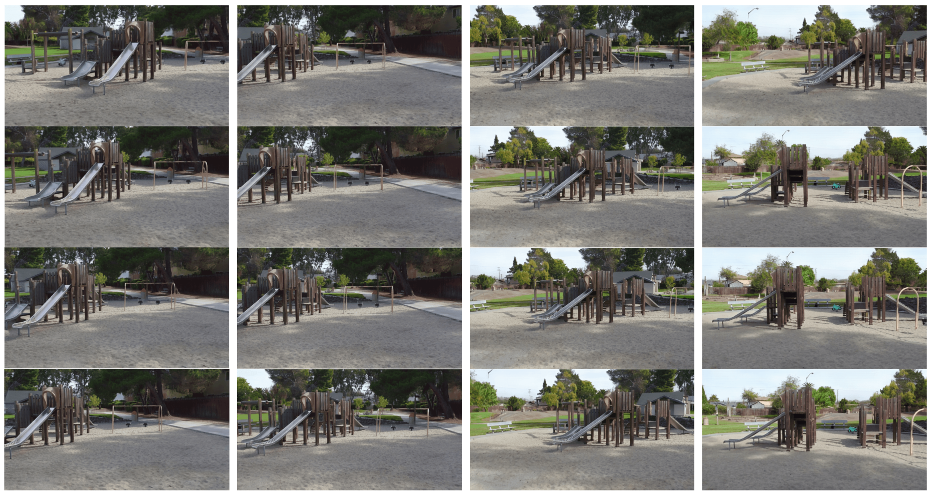


Fig. 3: Additional results on “Playground” scene of the Tanks and Temples [3] dataset.

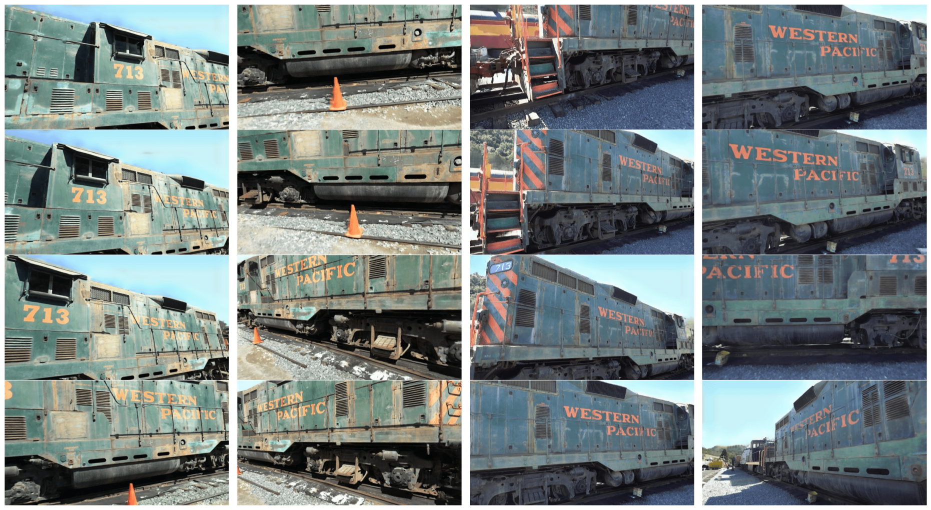


Fig. 4: Additional results on “Train” scene of the Tanks and Temples [3] dataset.

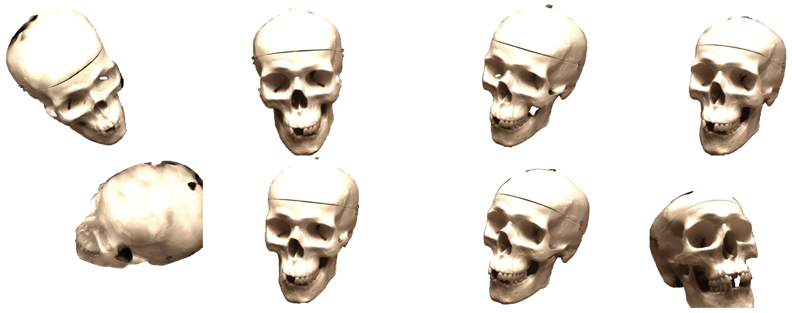


Fig. 5: Additional results on scene “65” of the DTU [1] dataset.

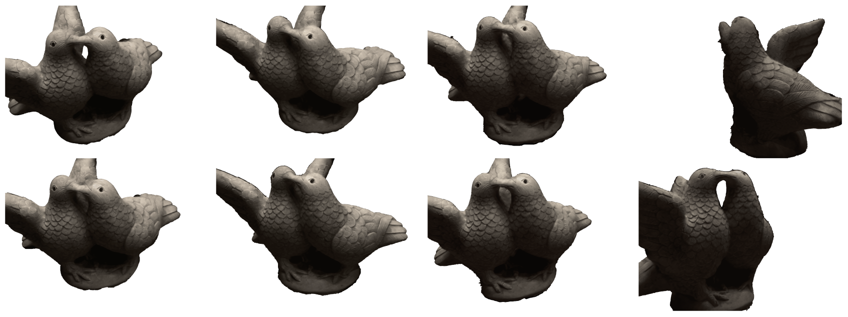


Fig. 6: Additional results on scene “106” of the DTU [1] dataset.

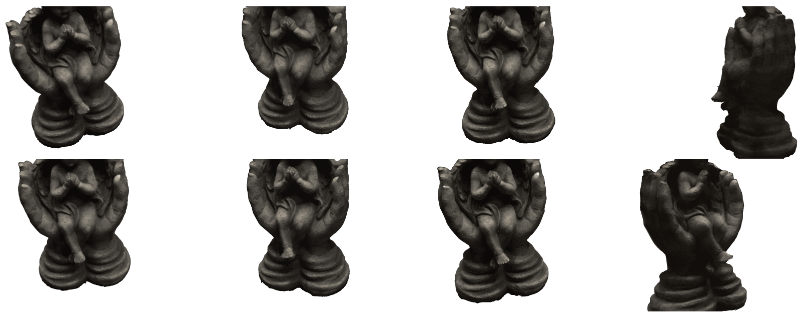


Fig. 7: Additional results on scene “118” of the DTU [1] dataset.

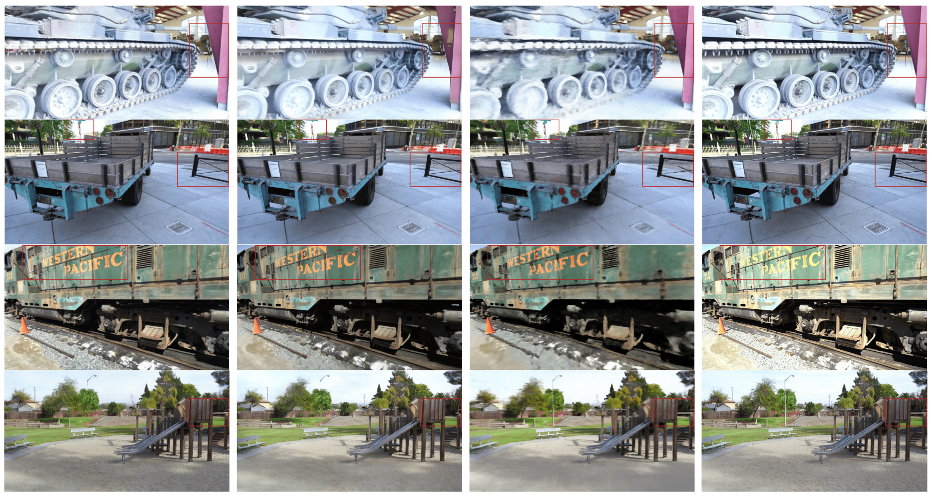


Fig. 8: Additional qualitative comparisons on Tanks and Temples [3].

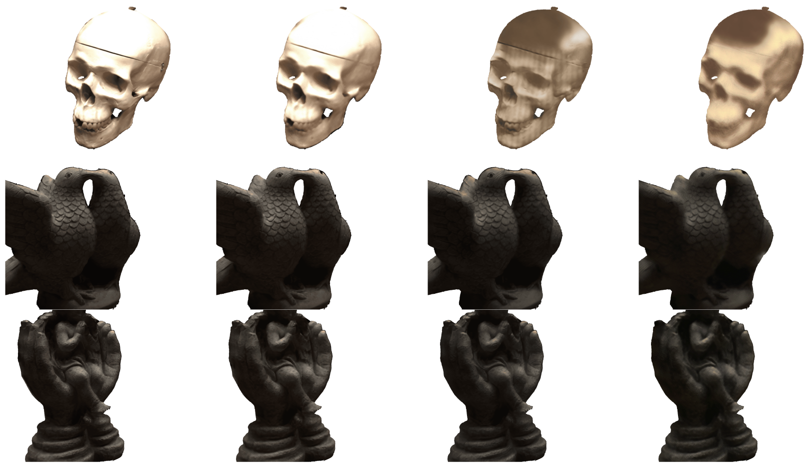


Fig. 9: Additional qualitative comparisons on DTU [1].

References

1. Aanæs, H., Jensen, R.R., Vogiatzis, G., Tola, E., Dahl, A.B.: Large-scale data for multiple-view stereopsis. International Journal of Computer Vision **120**(2), 153–168 (2016) 1, 4, 5
2. Aliev, K.A., Sevastopolsky, A., Kolos, M., Ulyanov, D., Lempitsky, V.: Neural point-based graphics. In: European Conference on Computer Vision. pp. 696–712. Springer (2020) 5
3. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics (ToG) **36**(4), 1–13 (2017) 1, 2, 3, 5
4. Riegler, G., Koltun, V.: Stable view synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12216–12225 (2021) 1, 5
5. Zhang, K., Riegler, G., Snavely, N., Koltun, V.: Nerf++: Analyzing and improving neural radiance fields. arXiv preprint arXiv:2010.07492 (2020) 1, 5