BluNF: Blueprint Neural Field

Robin Courant[1*] Xi Wang[1*] Marc Christie[2] Vicky Kalogeiton[1]
[1]LIX, Ecole Polytechnique, IP Paris [2]Inria, IRISA, CNRS, Univ. Rennes

Abstract

*Neural Radiance Fields (NeRFs) have revolutionized scene novel view synthesis, offering visually realistic, precise, and robust implicit reconstructions. While recent approaches enable NeRF editing, such as object removal, 3D shape modification, or material property manipulation, the manual annotation prior to such edits makes the process tedious. Additionally, traditional 2D interaction tools lack an accurate sense of 3D space, preventing precise manipulation and editing of scenes. In this paper, we introduce a novel approach, called **Bl**ueprint **N**eural **F**ield (BluNF), to address these editing issues. BluNF provides a robust and user-friendly 2D blueprint, enabling intuitive scene editing. By leveraging implicit neural representation, BluNF constructs a blueprint of a scene using prior semantic and depth information. The generated blueprint allows effortless editing and manipulation of NeRF representations. We demonstrate BluNF's editability through an intuitive click-and-change mechanism, enabling 3D manipulations, such as masking, appearance modification, and object removal. Our approach significantly contributes to visual content creation, paving the way for further research in this area.*

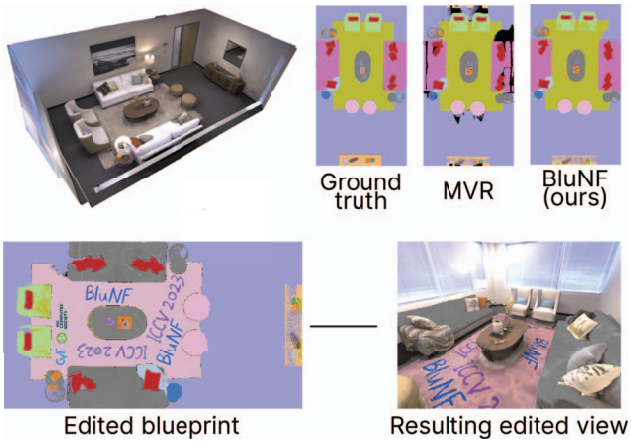


Figure 1: Generating a blueprint from multiple views of a scene can be challenging. (top) Traditional multiple-view reconstruction methods (MVR) depend on the diversity of views, and suffer from occlusions (black areas in MVR blueprint). Instead, our proposed BluNF addresses these by leveraging an implicit neural representation of the scene. (bottom) BluNF associated with NeRF enables scene editing, such as changes in appearance or texture.

1. Introduction

The demand for realistic and taylored 3D scenes is increasing for various applications, ranging from artistic purposes like movie and video game creation to more practical uses in architecture and design. While traditional methods, like photogrammetry and light-field encoding, have facilitated the reconstruction of 3D scenes from multiple viewpoints [40, 23], recent advancements in deep learning, such as Neural Radiance Fields (NeRF) [30, 3, 4, 48] have shown impressive capabilities in capturing and rendering realistic scenes. However, editing and manipulating NeRF representations remains challenging.

Existing works propose various NeRF editing techniques, such as scene composition using multiple NeRFs [54], manipulation of rendering properties [21], deformations through mesh-to-NeRF mappings [55], and object removal using user-provided masks [51]. However, selecting and editing NeRF views present specific challenges, particularly when relying on traditional 2D interaction tools that lack a true sense of 3D space (e.g. 2D screens, keyboards and mice). Especially when considering the needs of designers/users who often rely on simplified or partial views of scenes for higher-level semantic manipulation. Blueprints, commonly used in architectural designs, establish a clear link between the semantic representation of entities and their underlying geometric structure, providing a foundation for constructing real scene layouts. Beyond architectural applications, blueprints find utility in diverse computer vision and graphics applications, including motion planning [18], navigation [1, 14], and cinematographic staging [29]. Their use in all these domains highlights the enduring significance of blueprints as a means of conveying abstract scene information for various purposes.

In this paper, we present an approach for constructing a

*Equal contribution.

semantic editable blueprint from multiple viewpoints of a scene by leveraging prior semantic and depth information. Our novel **Bl**ueprint **N**eural **F**ield module, BluNF, generates a 2D semantic-aware editable blueprint that is robust to noise and sparse observations. Notably, BluNF enables user-friendly editing of NeRF representations. To the best of our knowledge, we are the first to employ an implicit neural field to construct a 2D semantic-aware blueprint for editing purposes, as most bird's-eye-view methods rely on CNNs or transformers [25].

First, BluNF is capable of generating blueprints even in scenarios with incomplete depth or semantic information, without explicit geometric constraints. It improves robustness compared to multi-view-based methods [16] (MVR) or straightforward NeRF orthogonal projections (see top-part Figure 1). Second, BluNF introduces a novel and intuitive editing approach for NeRF. By providing a concise and informative blueprint, BluNF enables easy editing and manipulation, including changes in appearance and object removal, achieved through a simple *click-and-change* mechanism, as depicted in bottom-part Figure 1.

Our contributions are: (i) BluNF, a novel module that generates a blueprint of a scene through an implicit neural field without explicit geometric constraints or supervision; (ii) an underlying representation that is robust to sparse and noisy observations, enabling reliable semantic layer identification and outperforming multi-view-based or classical NeRF methods; and (iii) a combination of BluNF with NeRF representations of the same scene, enabling intuitive user manipulations on the generated 2D blueprint (masking, appearance, removal), and enabling direct view rendering with NeRF incorporating the edits.

2. Related work

Implicit Neural Representation. Over the past few decades, implicit neural representations (INRs) have become popular for representing complex scenes due to their high efficiency, broad capacity, and diverse applications [27, 30, 24]. Their core idea is to exploit neural network representations to directly fit the target output by learning from input data without explicit parameterization. INRs have been successfully applied to various areas, including meshes [56, 35], voxels and point clouds [27, 19], and light fields [42]. Recently, NeRF [30] has shown success in realistic viewpoint reconstructions, synthesizing photorealistic images of unseen views, while preserving geometric consistency and handling reflective lighting conditions.

To enhance the learning process and improve the representation performance of INRs, numerous techniques have been introduced: SIREN [43] proposes a periodic activation function to fit complex and generic natural signals and derivative information such as images, acoustics, and spatial data. [46] propose a frequency-based Positional Encoding (PE) to extract features for improving fitting performance, especially in high-frequency areas. HashGrid [32] leverages hash encoding to address the ambiguity and smoothing issues, achieving gain in both fitting performance and training speed. In our work, we show that choosing the appropriate encoding scheme can have a substantial impact on the INR performance (Section 4.2).

NeRF. The original proposal for NeRF was made by Mildenhall et al [30], with the main contribution of a Positional Encoding-supported MLP network to encode 5D spatial and view angle information. The pixel-wise multi-views image is constrained by computing the ray-based volumetric rendering. Subsequent improvements and derivatives have been proposed in order to enhance the system's image synthesis performance [4, 3], robustness to sparse views [33], and support for dynamic scenes [39, 36], etc. Moreover, NeRF has become a powerful system capable of producing not only photorealistic rendering images, but also fitting various genres of applications. For instance, Zhi et al. [58] propose adding a semantic head parallel to the RGB one for semantic digit rendering. A similar task is also investigated in [49], where the main idea is to exploit the density field during the NeRF training.

Recently, [54] shows how to edit disentangled objects by constructing multiple NeRF models dedicated to each object and then combining them for scene rendering. This enables translating, rotating and scaling pre-defined objects in the scene. A similar concept is adapted in [13, 22, 57] by combining composition functionality with dynamic or semantic scene representations for more detailed user manipulations. GAN-based NeRF systems [41, 34, 8] provide another fresh perspective on the task of generating NeRF, yet do not enable manipulating pre-trained NeRF. In our work, we use the proposed BluNF for manipulating and editing a pre-trained NeRF representation of a scene.

Layout. Scene reconstruction is a longstanding problem in computer vision and graphics [9, 38, 59], traditionally tackled with 3D geometry techniques such as SfM [40] and SLAM [52, 50]. However, with the recent development of deep neural networks (DNNs), the reconstruction problem has evolved to encompass a variety of applications: from explicit supervised reconstruction [47, 10] to 2D BEV (Bird-Eye-View) extraction for autonomous driving applications [25, 15], and to INR-based reconstruction [5]. For reconstruction, various systems are proposed for different targets, scales [11, 53, 48], and usage scenarios, ranging from terrain [11, 53, 48] and urban [45] to indoor scenes [37] or surface reconstruction applications [2]. For editing, various systems allow for customizations, such as geometric editing or object removal [54, 55], color or style changing [17, 21], and generative manipulation [41, 34].

Nevertheless, most editing methods suffer from non-intuitive handling of 3D content (particularly in NeRF).

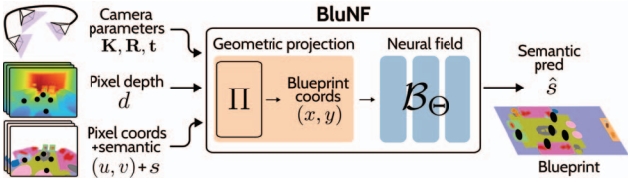


Figure 2: **Overview of BluNF training pipeline.** BluNF leverages camera parameters $\mathbf{K}, \mathbf{R}, \mathbf{t}$ and pixel depth d to establish a mapping from pixel coordinates (u, v) to blueprint coordinates (x, y). This mapping is achieved through the geometric projection module Π. The resulting blueprint coordinates (x, y) are then fed as inputs to the neural field module $\mathcal{B}_\Theta$, which predicts the corresponding semantic label $\hat{s}$. We use the semantic label s associated to pixel coordinates (u, v) as supervision target.

This requires users to possess prior knowledge, to provide semantic masks for each training image [54], manually annotate object contours [26, 31] or indicate a pretrained latent code that lacks detailed manipulation [12, 34, 41], or could involve batch operations [17, 21]. Previous works [12, 7] partially mention the idea of 2D layout for NeRF scene representation. However, they only focus on acceleration or latent controlled generation and not specifically intended for editing or semantic understanding tasks. Instead, we propose a blueprint neural representation that enables intuitive and efficient manipulation of the layout when combined with NeRF.

3. Method

In this work, we introduce the **Blu**eprint **N**eural **F**ield (BluNF) that generates a 2D semantic-aware blueprint of a 3D scene from sparse semantic views. For a given scene, a *blueprint* refers to a top-view semantic floorplan, as illustrated in Figure 1. In Section 3.1, we show that BluNF builds on neural implicit learning techniques by mapping pixel-wise semantic information to corresponding blueprint semantic labels constrained on a ray-based loss. In Section 3.2, we elaborate on how BluNF improves the efficiency and reliability of scene editing tasks.

3.1. BluNF

Figure 2 provides an overview of the BluNF training pipeline, illustrating the two main modules: (i) a *geometric projection* $\mathbf{\Pi}$ and (ii) a *neural field* $\mathcal{B}_\Theta$. To begin, we employ a *geometric projection module* $\mathbf{\Pi}$ to map pixel coordinates (u, v) from input views to 2D blueprint coordinates (x, y). Next, the *neural field module* $\mathcal{B}_\Theta$ predicts a semantic label $\hat{s}$ from the input blueprint coordinates. Thereby, BluNF effectively captures the underlying scene's semantic information, enabling to build an implicit representation of the blueprint. Below we detail these two modules.

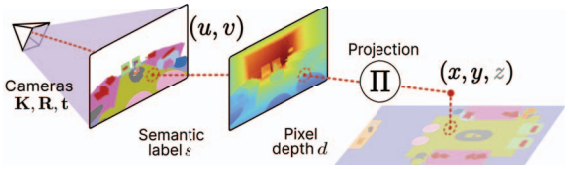


Figure 3: **View-to-blueprint projection.** Given an image view captured by a camera with its intrinsic and extrinsic parameters $\mathbf{K}, \mathbf{R}, \mathbf{t}$, and a associated depth, the projection module $\mathbf{\Pi}$ projects one pixel's coordinates (u, v), incorporating the associated depth d back to 3D space (x, y, z). We then project it onto the XY-plane (floor) to produce the 2D blueprint coordinates (x, y).

Projection module. The goal of the projection module is to transform the camera view coordinates (u, v) to blueprint coordinates (x, y). Figure 3 illustrates this process. Given semantic views, we project each pixel coordinate $(u, v) \in [\![H, W]\!]$ along with its associated semantic label $s \in [\![1, C_s]\!]$ onto the 2D semantic blueprint plan, where C_s represents the number of semantic classes. This projection, denoted as $\mathbf{\Pi}$, becomes feasible when the intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times3}$ and extrinsic matrix $[\mathbf{R}|\mathbf{t}] \in \mathbb{R}^{3\times4}$ associated with the camera view are known, along with the depth. By leveraging the projection matrix $\mathbf{\Pi}$ and the depth value $d \in \mathbb{R}$ at coordinates (u, v), we derive the 3D coordinates $(x, y, z) \in \mathbb{R}^3$:

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \mathbf{\Pi} \begin{bmatrix} u.d \\ v.d \\ d \\ 1 \end{bmatrix}, \text{ with } \mathbf{\Pi}^{-1} = \begin{bmatrix} \mathbf{K} & 0 \\ 0 & 1 \end{bmatrix} [\mathbf{R}|\mathbf{t}] \quad , \quad (1)$$

Next, by projecting the 3D coordinates onto the floor – i.e., the XY plane –, we obtain the corresponding blueprint coordinates $(x, y) \in \mathbb{R}^2$. Consequently, for all pixel coordinates (u, v) of scene views, we generate the inputs (x, y) used as training data for the neural field module.

Neural field module. The neural field module $\mathcal{B}_\Theta$ is designed to predict a semantic label $\hat{s}$ associated with the projected blueprint coordinates (x, y). $\mathcal{B}_\Theta$ consists of two submodules: (i) an encoding block that encodes blueprint coordinates $(x, y) \in \mathbb{R}^2$ into a higher-dimensional vector, and (ii) an MLP that maps the encoded input coordinates to a predicted semantic label $\hat{s}$. As depicted in Figure 2, after the projection of semantic labels from scene views, the neural field module learns an implicit representation of the scene's semantic blueprint. We choose to rely on an implicit representation due to its robustness and the inherent ability to handle sparse inputs, as explained in prior works [30, 43].

Training and loss. The BluNF pipeline is trained with a cross-entropy loss that compares the projected semantic la-

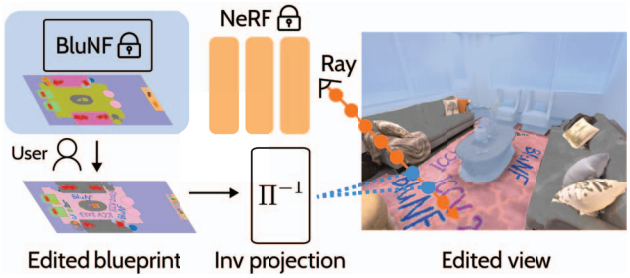


Figure 4: **Overview of BluNF editing pipeline.** It shows the process of leveraging a pre-trained BluNF model in conjunction with user interactions to edit a blueprint representation. During view rendering with a pre-trained NeRF, the NeRF samples (*orange dots*) impacted by the edits are replaced by back-projected samples from the edited blueprint (*blue dots*) using the inverse projection $\mathbf{\Pi}^{-1}$.

bel s associated with the pixel coordinates (u, v) and the predicted semantic label $\hat{s}$ generated by the neural field module. The parameters Θ of the MLP are then updated using standard gradient descent optimization.

Ambiguities. Geometric ambiguities may arise when projecting pixel coordinates from different views, resulting in the possibility of two different semantic labels being projected onto close blueprint coordinates. We incorporated the coordinate encoder in the neural field module to disentangle these geometric ambiguities, inspired by the encoding introduced in the NeRF paper [30]. By mixing both high- and low-frequency components, the encoder enables the MLP to distinguish points more accurately (high-frequency) while also smoothing out ambiguities (low-frequency).

3.2. BluNF editing

Our BluNF representation is designed to enable user interactions on the blueprint, and when combined with the corresponding NeRF representation, it enables rendering any views with the edits. Figure 4 illustrates this process. The blueprint consists of connected components, representing groups of pixels belonging to the same semantic class, forming distinct shapes. Through user selection, these connected components can be modified by assigning new textures or completely removed from the 3D scene. To incorporate the edits, we use the inverse of the projection $\mathbf{\Pi}^{-1}$, as defined in Section 3.1, to back-project the blueprint coordinates affected by the edits onto the 3D space, creating a set of 3D samples. During rendering, with a pre-trained NeRF model, we replace affected samples along the rays based on the back-projected edited samples. These sample replacements can involve adjustments to the color value, density value, or both. For instance, removing an object entails cancelling the density value of the corresponding sample.

4. Experiments

Metrics. To evaluate the performance of BluNF, we employ standard segmentation metrics [28]: pixel accuracy (*pAcc*) and frequency-weighted IoU (*fwIoU*). Additionally, we report the completeness factor (*comp.*) that measures the ratio of the valid output area to the total blueprint plan to assess the completeness of the reconstructed blueprint. To investigate the robustness of each method against partial observations, common in reconstruction problems [33, 16, 40], we report results for each dataset and metric with varying numbers of input frames $(90, 45, 9)$.

Datasets. We test on four scenes from two 3D indoor datasets: Replica [44] - *room 0* referred to as *R1*, and *room 2* referred to as *R2* -, and MatterPort3D [6] - *gZ6f7yhEvPG* referred to as *M1*, and *pLe4wQe7qrG* referred to as *M2*. Replica is a synthetic dataset, while MatterPort3D contains real-world RGB-D information. For both datasets, we manually collect the ground truth blueprints from the given 3D models by computing the orthogonal projection with carefully hand-selected area and culled objects to avoid vertical occlusion (Figure 5). See supplementary for more details.

4.1. Reconstruction comparison

Compared methods. We compare BluNF to two other methods: First, the multi-view reconstruction (MVR) method that directly projects and gathers all pixels in the dataset to generate the blueprint. Second, the *NeRF (top)* method that computes the blueprint by sampling orthogonal rays vertically to the floor in a pre-trained NeRF under the same input images. *NeRF (top)* is sensitive to the height selection: too low height leads to over-trimmed objects; too high height risks of being influenced by in-the-air or corner artifacts which are less supervised. For a fair comparison, we report the best fwIoU result among different sampled heights. For BluNF, we report results with two input depth sources: the ground-truth (*GT*) and the NeRF-estimated depth (*NeRF*). Additionally, for all methods, we use the semantic maps provided in the datasets.

Quantitative results. We compare our proposed BluNF to both methods mentioned above and report the results in Table 1. Overall, we observe that for the four scenes, BluNF outperforms the other methods for all metrics and all options, i.e., number of input frames and nature of depth.

More precisely, for **Replica scenes**: (i) *R1* (subtable top-left): BluNF performs the best for both depth options: for instance, with 90 frames as input, it reaches pACC of 92.6% and 90.6% with and without ground truth depth, respectively, against 88.1% and 88.3% for MVR. (ii) *R2* (subtable top-right): MVR and BluNF using the ground-truth depth achieve relatively high scores for all metrics and all numbers of frames. However, BluNF generally outperforms MVR, achieving higher scores for pACC, fwIoU, and completeness across all numbers of frames. For example, us-

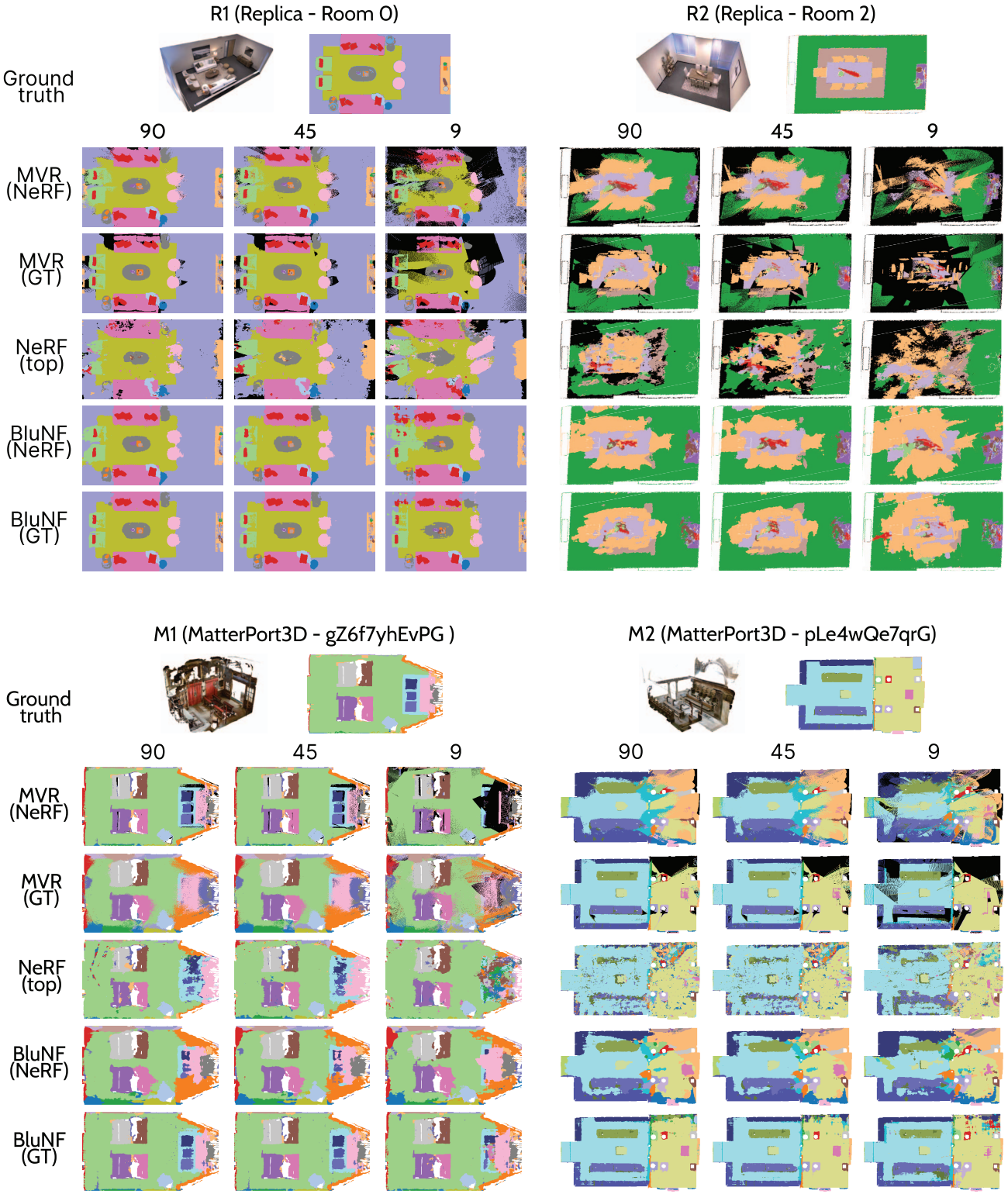


Figure 5: **Visualizations of generated blueprints for different methods** on the *Replica - R1* (top-left), *Replica - R2* (top-right), textitMatterPort3D - M1 (bottom-left) and *MatterPort3D - M2* (bottom-right) datasets. Each row corresponds to a method: NeRF top-view, MVR with or without ground-truth depth, and BluNF with or without ground-truth depth. Each column displays results for a different number of input frames.

Method (depth)	R1 (room 0)									R2 (room 2)								
	pACC ↑			fwIoU ↑			comp. ↑			pACC ↑			fwIoU ↑			comp. ↑		
# frames	90	45	9	90	45	9	90	45	9	90	45	9	90	45	9	90	45	9
MVR (GT)	88.1	87.0	65.4	82.7	81.7	61.3	95.2	94.1	72.2	60.6	56.6	31.1	53.2	50.3	29.5	51.8	47.9	26.1
BluNF (GT)	**92.6**	**92.7**	**90.0**	**86.7**	**86.9**	**82.3**	**100**	**100**	**100**	**71.9**	**71.9**	**58.6**	**60.9**	**60.9**	**49.6**	**100**	**100**	**100**
NeRF (top-view)	82.2	80.7	69.2	72.6	71.4	58.4	97.7	97.3	93.1	51.5	40.4	31.2	44.8	34.6	27.3	56.6	49.3	47.4
MVR (NeRF)	88.3	86.8	63.1	81.0	79.5	57.2	99.1	97.8	74.3	46.9	44.3	26.0	38.7	36.4	22.3	48.4	46.2	31.4
BluNF (NeRF)	**90.6**	**90.5**	**85.5**	**83.6**	**83.6**	**76.6**	**100**	**100**	**100**	**62.1**	**63.9**	**50.6**	**51.9**	**52.9**	**41.8**	**100**	**100**	**100**

(a) **Replica dataset**.

Method (depth)	M1 (gZ6f7yhEvPG)									M2 (pLe4wQe7qrG)								
	pACC ↑			fwIoU ↑			comp. ↑			pACC ↑			fwIoU ↑			comp. ↑		
# frames	90	45	9	90	45	9	90	45	9	90	45	9	90	45	9	90	45	9
MVR (GT)	87.8	86.8	73.1	82.8	81.9	69.4	90.3	89.1	74.7	81.5	80.3	65.5	76.5	75.3	61.6	77.4	76.5	64.4
BluNF (GT)	**89.0**	**89.1**	**84.2**	**82.9**	**82.9**	**76.5**	**100**	**100**	**100**	**87.4**	**87.6**	**82.8**	**80.7**	**80.9**	**75.5**	**100**	**100**	**100**
NeRF (top-view)	**80.3**	**80.0**	**73.1**	**73.0**	**72.0**	**64.9**	**100**	**100**	**100**	**62.5**	**62.8**	**59.5**	49.0	49.5	**47.4**	**100**	**100**	**100**
MVR (NeRF)	62.0	61.3	54.5	54.5	53.9	47.7	95.5	95.2	90.5	52.3	52.5	45.9	42.9	43.1	36.9	87.3	86.8	76.9
BluNF (NeRF)	67.9	67.9	64.2	59.0	59.4	56.5	**100**	**100**	**100**	62.0	62.3	57.3	**53.0**	**53.1**	47.2	**100**	**100**	**100**

(b) **MatterPort3D dataset**.

Table 1: **Comparison to the state of the art on *Replica* (a) and *MatterPort3D* (b) for varying number of frames (90, 45,9).** For both subtables, we report results for 2 scenes (left and right), the top part corresponds to results with ground-truth depth (GT), and the bottom part without, i.e., it uses the estimated-by-NeRF depth.

ing 90 frames, BluNF achieves pACC, fwIoU, and completeness scores of 71.9%, 60.9%, and 100%, respectively, while MVR achieves scores of 60.6%, 53.2%, and 51.8%, respectively. For **MatterPort3D scenes**: (i) *M1* (subtable bottom-left): with ground-truth depth (GT), BluNF is also better than MVR for both pACC and fwIoU; for instance, at 90 frames it reaches 89.0% and 82.9% against 87.8% and 82.8%, for both metrics respectively. When using NeRF-estimated depth, even though BluNF outperforms MVR by +9.7% in pACC and +8.8% in fwIoU for 9 frames; *NeRF (top)* performs the best with 73.1% of pACC and 64.9% for 9 frames as well. (i) *M2* (subtable bottom-right): with ground-truth depth, the results are similar, with BluNF again outperforming MVR for all metrics and numbers of frames. Interestingly, the scores are in most cases lower for MatterPort3D scenes than for Replica scenes. We attribute this discrepancy to the intrinsic quality differences between the datasets. The MatterPort3D datasets rely on RGB-D acquisitions that vary in quality, resulting in a relatively poorer quality overall. For a more detailed discussion on this matter, please refer to the supplementary material.

Qualitative results. Figure 5 shows visual results of all methods for both datasets. It highlights the failure of other methods, notably MVR, and especially when the number of frames is low (blacks areas in blueprints in rows 1, 2, 6, 7; columns 3, 6). Moreover, we note the high quality of BluNF-generated blueprint (row 5, 10; columns 1, 4), in contrast to the others (rows 2, 3, 7, 8; columns 1, 4).

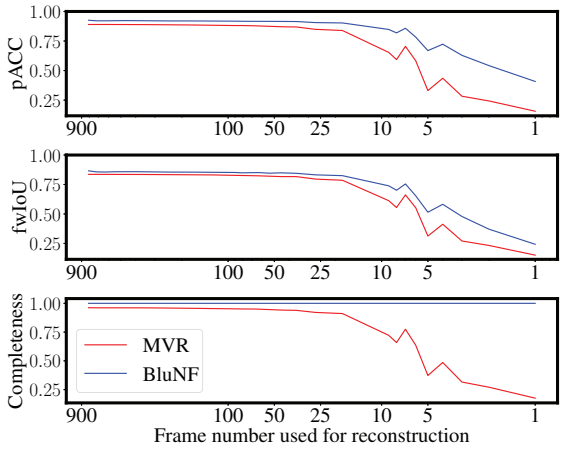


Figure 6: **Ablation of number of input frames for BluNF and MVR on Replica-R1 [44].** BluNF outperforms MVR in all metrics for all input frame numbers. Their gap is more notable when reducing to few input frames, i.e., less than 5.

4.2. Ablation study

In this section, we present the results of four ablation studies to highlight the influence of: (i) the number of frames; (ii) the impact of depth; (iii) the analysis of completeness; (iv) the different types of encoding. Most discussions and ablations are presented with the Replica R1 dataset, unless stated otherwise.

Number of Frames. The number of frames is a critical factor for almost all reconstruction methods, including both geometric-based and learning-based approaches [33, 40]. A low overlapping of multi-view information can lead to missing parts or erroneous estimations. In Figure 6, we compare BluNF against MVR for different number of frames gradually reduced from 900 to 1 for Replica *R1*. We observe that our BluNF outperforms MVR for all metrics. Specifically, when the number of frames is high (e.g. some hundreds), both methods perform similarly for all metrics. However, the more the number of frames reduces, the more the gap between the two methods increases. This gap becomes even more pronounced when the number of frames is very low (e.g., fewer than 5 frames). This highlights the fact that leveraging the filling capacity of implicit representations used by BluNF leads to significantly better performance. We also observe this trend in Table 1a.

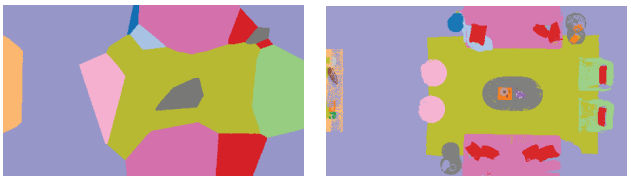
Impact of depth. Here, we examine the impact of depth on our proposed BluNF by comparing the performance of BluNF with GT and NeRF-based depths, i.e., rows 2 and 5 in Table 1a and 1b. For **Replica dataset *R1***, the nature of depth (GT or NeRF-estimated) does not influence much the performances of BluNF. Between ground-truth and estimated depth, BluNF drop around 5% of pACC and fwIoU. For **MatterPort3D *M1***, the impact of depth is more prominent: the fwIoU of BluNF with 45 frames decreases from 82.9% with ground truth depth to 59.4% without. Those two behaviours can be explained by the contrasted quality between the synthetic and real-acquired depth information. Refer to the supplementary material for more details.

Analysis of completeness. Regarding completeness, BluNF exploits a continuous neural field and therefore its completeness is always maximum (100%). Nevertheless, note that for MVR this is an important drawback, and it gets worse when the number of frames decreases. For Replica *R1* in Table 1a, *MVR (GT)* completeness decreases from 95.2% with 90 frames, to 72.2% with 9 frames. In addition, it is interesting to see that NeRF-estimated depth increases the completeness of MVR. For instance, on MatterPort3D *M1* in Table 1b with 9 frames, it increases from 74.7% with ground truth, to 90.5% without. It is explained by the error on the estimated depth, making the projection fuzzy, yet less accurate. Note that *NeRF (top)* achieves full completeness on MatterPort3D as there is no ceiling semantic class, on the contrary of Replica, polluted by ceiling predictions.

BluNF input encoding. We compare in Table 2 the effectiveness of different types of encoding: (i) no encoding; (ii) the hash encoding from Instant-NGP [32]; (iii) a SIREN-based BluNF [43]; and (iv) the standard positional encoding (PE) introduced in NeRF [30]. Firstly, the results validate our claim in Section 3.1 that encoding helps to disentangle the geometric ambiguities associated with the projection module. Specifically, there is a significant gap of up to 19.6% in fwIoU between with and without PE encoding schemes when using 9 input frames. In addition, these results show that with a high number of frames, the encoding type does not affect the results; the difference of pACC and fwIoU between the different encoding is around ±1%. Nevertheless, when the number of frames decreases, *Hash* and *SIREN* are highly impacted. For instance for 9 frames, we report a pACC of 84.8% and 84.5% and a fwIoU of 73.9% and 75.2% respectively. In contrast *PE* seems much less impacted with an pACC of 90.0% and a fwIoU of 82.3%. Figure 7 presents a visual comparison of the influence of input encoding on the resulting blueprints. We compare blueprints generated with and without positional encoding. The comparison provides evidence supporting our claim from Section 3.1 that positional encoding helps disentangle projection ambiguities. Blueprint with PE exhibits enhanced detail and accuracy, while those without encoding lack detail and are dominated by ambiguous shapes.

Encoding type	pACC ↑			fwIoU ↑		
Number of frames	90	45	9	90	45	9
No encoding	85.4	85.5	73.3	77.0	76.2	63.0
Hash	91.8	91.6	84.8	85.2	84.9	73.9
SIREN	91.0	91.0	84.5	84.1	84.2	75.2
PE	**92.6**	**92.7**	**90.0**	**86.7**	**86.9**	**82.3**

Table 2: **Ablations of coordinate encoding** in BluNF on Replica *R1* with ground-truth depth.



(a) w/o PE (b) w/ PE

Figure 7: **Visual comparison of different input encodings.** (a) Result with positional encoding (PE), and (b) result without positional encoding (w/o PE).

4.3. Editing comparison

Comparison to baselines. In this work, we compare our BluNF editing pipeline with two baselines based on semantic-NeRF [58]. The first baseline involves the user choosing a semantic class to edit for the entire scene and using the rendered semantic view of semantic-NeRF as a mask to apply edits on the RGB view. However, as shown in Figure 8, this approach has several drawbacks compared to our proposed method BluNF. Specifically, as demonstrated by the blue arrow the two armchairs are selected due to belonging to the same semantic class. In contrast, BluNF en-

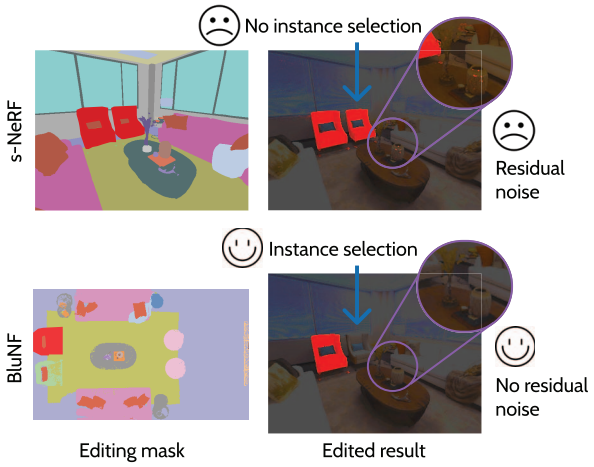


Figure 8: **Editing result comparison.** (Top) Semantic-NeRF-based editing results and (bottom) our proposed BluNF editing result.

ables us to select specific instances within the same semantic class. Additionally, the purple zoom illustrates that using every pixel of the same semantic class leads to artifacts intrinsic to semantic-NeRF, while BluNF, relying on connected components, avoids selecting these artifacts, highlighting the flexibility of BluNF. The second baseline involves using the same mask selection as in BluNF, requiring the user to select connected components for each semantic view. However, this approach is impractical as it would demand the user to manually select connected components for all frames, emphasizing the generalization power of BluNF.

Qualitative results. Here we present the results of the BluNF editing pipeline introduced in Section 3.2. Figure 9 illustrates the outcomes, where the first column shows the selected blueprint area for each application, while the second and third columns illustrate two different synthesized views. The applications we showcase are **Object selection/masking**: Our system enables the user to select instances for editing by clicking on the generated blueprint. Grouped instance is able to back-project to 3D as a high-quality dynamic mask; **Object recoloring**: The user can change the color of the selected area by simply applying color blending; **Prompt Re-texturing**: The user can generate a 2D texture from a text-prompt with an external model [20]. Our system supports direct 2D textured mapping on the blueprint, which can be printed on the surface of selected objects; **Sketching**: Our system supports sketching or painting for editing instances; **Instance Removal**: Our system supports primitive instance removal by setting the density of the selected area to zero without requiring re-training of the NeRF. In the last row, we show that we remove a vase from the table by selecting it on the blueprint.

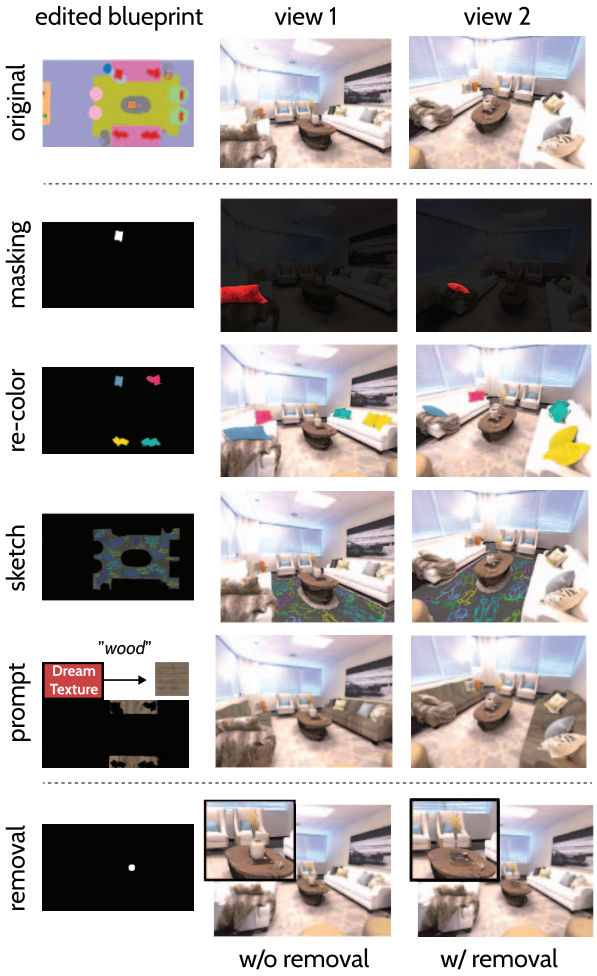


Figure 9: **Show cases of applications enabled by BluNF.** Examples of masking, recoloring, prompting, hand-sketching and instance removal.

5. Conclusion

In this work, we introduced BluNF, a novel module capable of generating 2D semantic-aware editable blueprints of scenes. By leveraging implicit learning with semantic and depth priors, BluNF demonstrates superior performance compared to geometric and semantic NeRF-based methods. BluNF combined with NeRF allows various intuitive editing applications. It bridges the gap left by standard editing methods, limited by their lack of 3D spatial understanding when relying on traditional 2D interaction tools.

6. Acknowledgment

We would like to thank Nicolas Dufour for proofreading and the anonymous reviewers for their feedback. This work was supported by ANR-22-CE23-0007 project and the Hi!Paris collaborative project and scholarship.

References

[1] Eloi Alonso, Maxim Peter, David Goumard, and Joshua Romoff. Deep reinforcement learning for navigation in aaa video games. *IJCAI*, 2020. 1

[2] Dejan Azinović, Ricardo Martin-Brualla, Dan B Goldman, Matthias Nießner, and Justus Thies. Neural rgb-d surface reconstruction. In *CVPR*, 2022. 2

[3] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratuk P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 1, 2

[4] Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022. 1, 2

[5] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *ECCV*, 2020. 2

[6] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from rgb-d data in indoor environments. *3DV*, 2017. 4

[7] Anpei Chen, Zexiang Xu, Andreas Geiger, Jingyi Yu, and Hao Su. Tensorf: Tensorial radiance fields. In *ECCV*, 2022. 3

[8] Yuedong Chen, Qianyi Wu, Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. Sem2nerf: Converting single-view semantic masks to neural radiance fields. In *ECCV*, 2022. 2

[9] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, 2015. 2

[10] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 2

[11] Dawa Derksen and Dario Izzo. Shadow neural radiance fields for multi-view satellite photogrammetry. In *CVPR*, 2021. 2

[12] Terrance DeVries, Miguel Angel Bautista, Nitish Srivastava, Graham W Taylor, and Joshua M Susskind. Unconstrained scene generation with locally conditioned radiance fields. In *ICCV*, 2021. 3

[13] Xiao Fu, Shangzhan Zhang, Tianrun Chen, Yichong Lu, Lanyun Zhu, Xiaowei Zhou, Andreas Geiger, and Yiyi Liao. Panoptic nerf: 3d-to-2d label transfer for panoptic urban scene segmentation. *3DV*, 2022. 2

[14] Sukkpranhachai Gatesichapakorn, Jun Takamatsu, and Miti Ruchanurucks. Ros based autonomous mobile robot navigation using 2d lidar and rgb-d camera. In *2019 First international symposium on instrumentation, control, artificial intelligence, and robotics (ICA-SYMP)*, 2019. 1

[15] Nikhil Gosala and Abhinav Valada. Bird's-eye-view panoptic segmentation using monocular frontal view. *IEEE Robotics and Automation Letters*, 2022. 2

[16] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2, 4

[17] Yi-Hua Huang, Yue He, Yu-Jie Yuan, Yu-Kun Lai, and Lin Gao. Stylizednerf: consistent 3d scene stylization as stylized nerf via 2d-3d mutual learning. In *CVPR*, 2022. 2, 3

[18] Brian Ichter, Edward Schmerling, Tsang-Wei Edward Lee, and Aleksandra Faust. Learned critical probabilistic roadmaps for robotic motion planning. In *ICRA*, 2020. 1

[19] Yue Jiang, Dantong Ji, Zhizhong Han, and Matthias Zwicker. Sdfdiff: Differentiable rendering of signed distance fields for 3d shape optimization. In *CVPR*, 2020. 2

[20] Carson Katri. Dreamtextures. https://github.com/carson-katri/dream-textures, 2023. 8

[21] Zhengfei Kuang, Fujun Luan, Sai Bi, Zhixin Shu, Gordon Wetzstein, and Kalyan Sunkavalli. Palettenerf: Palette-based appearance editing of neural radiance fields. *CVPR*, 2022. 1, 2, 3

[22] Abhijit Kundu, Kyle Genova, Xiaoqi Yin, Alireza Fathi, Caroline Pantofaru, Leonidas J Guibas, Andrea Tagliasacchi, Frank Dellaert, and Thomas Funkhouser. Panoptic neural fields: A semantic object-aware neural scene representation. In *CVPR*, 2022. 2

[23] Marc Levoy and Pat Hanrahan. Light field rendering. In *SIGGRAPH*, 1996. 1

[24] Lei Li, Siyu Zhu, Hongbo Fu, Ping Tan, and Chiew-Lan Tai. End-to-end learning local multi-view descriptors for 3d point clouds. In *CVPR*, 2020. 2

[25] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *ECCV*, 2022. 2

[26] Hao-Kang Liu, I Shen, Bing-Yu Chen, et al. Nerf-in: Free-form nerf inpainting with rgb-d priors. *arXiv preprint arXiv:2206.04901*, 2022. 3

[27] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM TOG*, 2019. 2

[28] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 4

[29] Amaury Louarn, Marc Christie, and Fabrice Lamarche. Automated staging for virtual cinematography. In *SIGGRAPH*, 2018. 1

[30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ECCV*, 2021. 1, 2, 3, 4, 7

[31] Ashkan Mirzaei, Tristan Aumentado-Armstrong, Konstantinos G Derpanis, Jonathan Kelly, Marcus A Brubaker, Igor Gilitschenski, and Alex Levinshtein. Spin-nerf: Multiview segmentation and perceptual inpainting with neural radiance fields. *CVPR*, 2023. 3

[32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM TOG*, 2022. 2, 7

[33] Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022. 2, 4, 7

[34] Michael Niemeyer and Andreas Geiger. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021. 2, 3

[35] Merlin Nimier-David, Sébastien Speierer, Benoît Ruiz, and Wenzel Jakob. Radiative backpropagation: an adjoint method for lightning-fast differentiable rendering. *ACM TOG*, 2020. 2

[36] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021. 2

[37] Georgios Pavlakos, Ethan Weber, Matthew Tancik, and Angjoo Kanazawa. The one where they reconstructed 3d humans and environments in tv shows. In *ECCV*, 2022. 2

[38] Stefan Popov, Pablo Bauszat, and Vittorio Ferrari. Corenet: Coherent 3d scene reconstruction from a single rgb image. In *ECCV*, 2020. 2

[39] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2

[40] Johannes L. Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 1, 2, 4, 7

[41] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. Graf: Generative radiance fields for 3d-aware image synthesis. *NeurIPS*, 2020. 2, 3

[42] Jinglei Shi, Xiaoran Jiang, and Christine Guillemot. Learning fused pixel and feature-based view reconstructions for light fields. In *CVPR*, 2020. 2

[43] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *NeurIPS*, 2020. 2, 3, 7

[44] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, et al. The replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019. 4, 6

[45] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul Srinivasan, Jonathan T. Barron, and Henrik Kretzschmar. Block-NeRF: Scalable large scene neural view synthesis. *CVPR*, 2022. 2

[46] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *NeurIPS*, 2020. 2

[47] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *CVPR*, 2017. 2

[48] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, 2022. 1, 2

[49] Suhani Vora, Noha Radwan, Klaus Greff, Henning Meyer, Kyle Genova, Mehdi SM Sajjadi, Etienne Pot, Andrea Tagliasacchi, and Daniel Duckworth. Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. *TMLR*, 2022. 2

[50] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martí n, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *CVPR*, 2019. 2

[51] Silvan Weder, Guillermo Garcia-Hernando, Á ron Monszpart, Marc Pollefeys, Gabriel Brostow, Michael Firman, and Sara Vicente. Removing objects from NeRFs. In *CVPR*, 2023. 1

[52] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, 2015. 2

[53] Yuanbo Xiangli, Linning Xu, Xingang Pan, Nanxuan Zhao, Anyi Rao, Christian Theobalt, Bo Dai, and Dahua Lin. Bungeenerf: Progressive neural radiance field for extreme multi-scale scene rendering. In *ECCV*, 2022. 2

[54] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, 2021. 1, 2, 3

[55] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: geometry editing of neural radiance fields. In *CVPR*, 2022. 1, 2

[56] Cheng Zhang, Bailey Miller, Kai Yan, Ioannis Gkioulekas, and Shuang Zhao. Path-space differentiable rendering. *ACM TOG*, 2020. 2

[57] Jiakai Zhang, Xinhang Liu, Xinyi Ye, Fuqiang Zhao, Yanshun Zhang, Minye Wu, Yingliang Zhang, Lan Xu, and Jingyi Yu. Editable free-viewpoint video using a layered neural representation. *ACM TOG*, 2021. 2

[58] Shuaifeng Zhi, Tristan Laidlow, Stefan Leutenegger, and Andrew J Davison. In-place scene labelling and understanding with implicit scene representation. In *ICCV*, 2021. 2, 7

[59] Michael Zollhöfer, Patrick Stotko, Andreas Görlitz, Christian Theobalt, Matthias Nießner, Reinhard Klein, and Andreas Kolb. State of the art on 3d reconstruction with rgb-d cameras. In *Comput. Graph. Forum*, 2018. 2