BluNF: Blueprint Neural Field

Robin Courant\textsuperscript{1*} \quad Xi Wang\textsuperscript{1*} \quad Marc Christie\textsuperscript{2} \quad Vicky Kalogeiton\textsuperscript{1}

\textsuperscript{1}LIX, Ecole Polytechnique, IP Paris \quad \textsuperscript{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 Blueprint Neural Field (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.

1. Introduction

The demand for realistic and tailored 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 \cite{40, 23}, recent advancements in deep learning, such as Neural Radiance Fields (NeRF) \cite{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 \cite{54}, manipulation of rendering properties \cite{21}, deformations through mesh-to-NeRF mappings \cite{55}, and object removal using user-provided masks \cite{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 \cite{18}, navigation \cite{1, 14}, and cinematographic staging \cite{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

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.
semantic editable blueprint from multiple viewpoints of a scene by leveraging prior semantic and depth information. Our novel **Blueprint Neural Field** 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-view 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),...
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 pre-trained 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 **Blueprint Neural Field** (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 module** II and (ii) a **neural field module** $B_\theta$. To begin, we employ a geometric projection module II to map pixel coordinates $(u, v)$ to 2D blueprint coordinates $(x, y)$. Next, the neural field module $B_\theta$ 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 2: Overview of BluNF training pipeline. BluNF leverages camera parameters $K, R, t$ 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 II. The resulting blueprint coordinates $(x, y)$ are then fed as inputs to the neural field module $B_\theta$, which predicts the corresponding semantic label $\hat{s}$. We use the semantic label $s$ associated to pixel coordinates $(u, v)$ as supervision target.](image)

**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 II, becomes feasible when the intrinsic matrix $K \in \mathbb{R}^{3 \times 3}$ and extrinsic matrix $[R|t] \in \mathbb{R}^{3 \times 4}$ associated with the camera view are known, along with the depth. By leveraging the projection matrix II and the depth value $d \in \mathbb{R}$ at coordinates $(u, v)$, we derive the 3D coordinates $(x, y, z) \in \mathbb{R}^3$:

$$
\begin{bmatrix}
  x \\
  y \\
  z \\
 1
\end{bmatrix} = II \begin{bmatrix}
  u, d \\
  v, d \\
  d \\
 1
\end{bmatrix}, \text{ with } II^{-1} = \begin{bmatrix}
  K & 0 \\
  0 & 1
\end{bmatrix} \begin{bmatrix}
  R | t
\end{bmatrix}, \tag{1}
$$

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 $B_\theta$ is designed to predict a semantic label $\hat{s}$ associated with the projected blueprint coordinates $(x, y)$. $B_\theta$ consists of two sub-modules: (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-
To evaluate the performance of BluNF, we employ 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. 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).

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), *MatterPort3D - 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.
BluNF outperforms MVR 100%.

R1 (room 0) R2 (room 2)

<table>
<thead>
<tr>
<th>Method (depth)</th>
<th>pACC ↑</th>
<th>fwIoU ↑</th>
<th>comp. ↑</th>
<th>pACC ↑</th>
<th>fwIoU ↑</th>
<th>comp. ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>MVR (GT)</td>
<td>88.1</td>
<td>87.0</td>
<td>65.4</td>
<td>82.7</td>
<td>81.7</td>
<td>61.3</td>
</tr>
<tr>
<td>BluNF (GT)</td>
<td>92.6</td>
<td>92.7</td>
<td>90.0</td>
<td>86.7</td>
<td>86.9</td>
<td>82.3</td>
</tr>
<tr>
<td>NeRF (top-view)</td>
<td>82.2</td>
<td>80.7</td>
<td>69.2</td>
<td>72.6</td>
<td>71.4</td>
<td>58.4</td>
</tr>
<tr>
<td>MVR (NeRF)</td>
<td>88.3</td>
<td>86.8</td>
<td>63.1</td>
<td>81.0</td>
<td>79.5</td>
<td>57.2</td>
</tr>
<tr>
<td>BluNF (NeRF)</td>
<td>90.6</td>
<td>90.5</td>
<td>85.5</td>
<td>83.6</td>
<td>83.6</td>
<td>76.6</td>
</tr>
</tbody>
</table>

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.

![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.](image)

### 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.

#### 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).
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 fwiIoU. For MatterPort3D M1, the impact of depth is more prominent: the fwiIoU 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

<table>
<thead>
<tr>
<th>Encoding type</th>
<th>pACC ↑</th>
<th>fwiIoU ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>No encoding</td>
<td>90.4</td>
<td>85.5</td>
</tr>
<tr>
<td>Hash</td>
<td>91.8</td>
<td>91.6</td>
</tr>
<tr>
<td>SIREN</td>
<td>91.0</td>
<td>91.0</td>
</tr>
<tr>
<td>PE</td>
<td>92.6</td>
<td>92.7</td>
</tr>
</tbody>
</table>

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

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

up to 19.6% in fwiIoU 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 fwiIoU 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 fwiIoU of 73.9% and 75.2% respectively. In contrast PE seems much less impacted with an pACC of 90.0% and a fwiIoU 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.

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-
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.

**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

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