Robust Camera Pose Refinement for Multi-Resolution Hash Encoding

Hwan Heo [1] Taekyung Kim [2] Jiyoung Lee [2] Jaewon Lee [1] Soohyun Kim [1] Hyunwoo J. Kim [1] Jin-Hwa Kim [2 3]

Abstract

Multi-resolution hash encoding has recently been proposed to reduce the computational cost of neural renderings, such as NeRF. This method requires accurate camera poses for the neural renderings of given scenes. However, contrary to previous methods jointly optimizing camera poses and 3D scenes, the naïve gradient-based camera pose refinement method using multi-resolution hash encoding severely deteriorates performance. We propose a joint optimization algorithm to calibrate the camera pose and learn a geometric representation using efficient multi-resolution hash encoding. Showing that the oscillating gradient flows of hash encoding interfere with the registration of camera poses, our method addresses the issue by utilizing smooth interpolation weighting to stabilize the gradient oscillation for the ray samplings across hash grids. Moreover, the curriculum training procedure helps to learn the level-wise hash encoding, further increasing the pose refinement. Experiments on the novel-view synthesis datasets validate that our learning frameworks achieve state-of-the-art performance and rapid convergence of neural rendering.

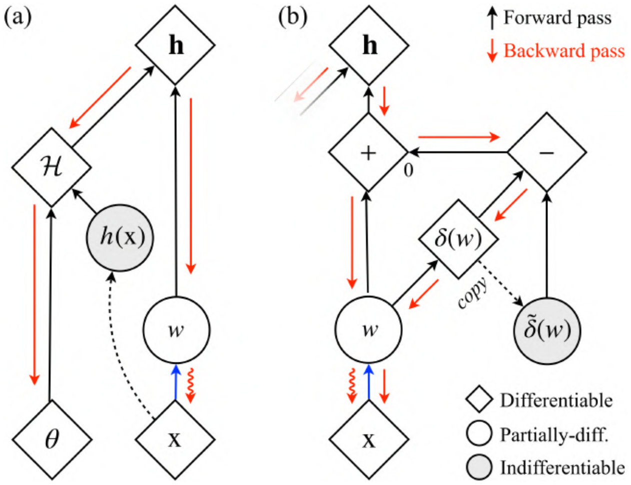

Figure 1. Gradient smoothing (a) → (b) to attenuate the gradient fluctuation (jiggled red arrow) of the hash encoding $\mathbf{h}$. Notice that we omitted the overlapped region in (b) being transparent. For the camera pose refinement, the error back-propagation passes through the d-linear interpolation weight w; however, its derivative is determined by the sign of relative position of the input coordinate $\mathbf{x}$ to the corners of the hash grid. The gradient fluctuation from this makes it difficult to converge. Please refer to Sec. 3 for the implementation details and the definitions of other symbols.

1. Introduction

A great surge in neural rendering has emerged in the last few years. Specifically, the Neural Radiance Fields (Mildenhall et al., 2020) (NeRF) has shown remarkable performances in the novel view synthesis. NeRF leverages a fully connected network to implicitly encode a 3D scene as a continuous signal and renders novel views through a differentiable volume rendering. However, when the rendering of NeRF performs, a large number of inferences are inevitable, making the computational burden of training and evaluation heavier.

Aware of this problem, related works have circumvented the shortcomings by introducing grid-based approaches (Liu et al., 2020; Yu et al., 2021a; Hedman et al., 2021; Sun et al., 2022; Wu et al., 2021; Fridovich-Keil et al., 2022; Karnewar et al., 2022), which store view direction-independent representations in dense grids. While these methods explicitly encode the whole scene simultaneously, they face a trade-off between the computational cost of the model size and its performance. Therefore, delicate training strategies such as pruning or distillation are often required to preserve the view-synthesis quality and reduce the model size. Recently, Instant-NGP (Müller et al., 2022) addressed these problems by proposing multi-resolution hash encoding for positional encoding, which combines a multi-resolution decomposition with a lightweight hash grid. The multi-resolution hash encoding achieved state-of-the-art performance and the fastest convergence speed of NeRF.

[1] Department of Computer Science, Korea University, Republic of Korea [2] NAVER AI Lab, Republic of Korea [3] AI Institute of Seoul National University, Republic of Korea. Correspondence to: Jin-Hwa Kim & Hyunwoo J. Kim <j1nhwa.kim@navercorp.com; hyunwoojkim@korea.ac.kr>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Despite the impressive performance of multi-resolution hash encoding, the volume rendering procedure (emission-absorption ray casting (Kajiya & Herzen, 1984)) used in Instant-NGP depends largely on the accurate camera poses. This method samples the points along the ray defined by direction and origin, which are determined by the camera pose. However, obtaining accurate camera poses in real-world scenarios might be unavailable, so most existing works utilize an off-the-shelf algorithm such as Structure-from-Motion (SfM), or COLMAP (Schönberger & Frahm, 2016). The previous works (Wang et al., 2021c; Jeong et al., 2021; Lin et al., 2021) have attempted to resolve this issue by jointly optimizing camera poses and scene representations with the original NeRF. However, applying this approach to multi-resolution hash encoding leads to severe deteriorations in pose refinement and scene representation.

Based on the gradient analysis of the naïve joint optimization of pose parameters and multi-resolution hash encodings, we demonstrate that the non-differentiability of the hash function and the discontinuity of the d-linear weights as a function of the input coordinate leads to the fluctuation in the Jacobian of the multi-resolution hash encodings. We investigate a novel learning strategy for jointly optimizing the camera pose parameters and the other parameters when the camera poses are noisy or unknown, utilizing the outstanding performance of multi-resolution hash encoding.

Given that, we propose to use a non-linear activation function in our straight-through estimator for smooth gradients in the backward pass, consistently maintaining the d-linear interpolation in the forward pass (*ref.* Figure 1). Moreover, we propose multi-level curriculum training that regulates the convergence speed of each level-wise encoding. We also empirically show that a small decoder compared to the size of the hash table (Müller et al., 2022) converges to suboptimal when the camera poses are noisy. The ablation studies on the depth and wide of the decoding networks and the core components of the proposed learning framework firmly validate our proposed method for robust camera pose refinement for multi-resolution hash encoding.

In summary, our contributions are three-fold:

- We analyze the derivative of the multi-resolution hash encoding, and empirically show that the gradient fluctuation negatively affects the pose refinement.
- We propose an efficient learning strategy jointly optimizing multi-resolution hash encodings and camera poses, leveraging the smooth gradient and the curriculum learning for coarse-to-fine adaptive convergences.
- Our method achieves state-of-the-art performance in pose refinement and novel-view synthesis with a faster learning speed than competitive methods.

2. Related Work

2.1. Neural Rendering

Mildenhall et al. (2020) first introduced the Neural Radiance Fields (NeRF) which parameterizes 3D scenes using neural networks. They employed a fully differentiable volume rendering procedure and a sinusoidal encoding to reconstruct high-fidelity details of the scene representations. The necessity of sinusoidal encoding was examined from the perspectives of kernel regression (Tancik et al., 2020), or the hierarchical structure of a natural scene reconstruction task (Landgraf et al., 2022).

Subsequently, in order to improve the reconstruction quality of NeRF, various modifications have been proposed such as replacing the ray casting with anti-aliased cone tracing (Barron et al., 2021), disentangling foreground and background models through non-linear sampling algorithms (Zhang et al., 2020; Neff et al., 2021; Barron et al., 2022), or learning implicit surface instead of the volume density field, *e.g.*, signed distance function (Oechsle et al., 2021; Wang et al., 2021b; Yariv et al., 2021). Also, there are several applicative studies with decomposition of NeRF (Pumarola et al., 2021; Martin-Brualla et al., 2021; Srinivasan et al., 2021; Boss et al., 2021; Rebain et al., 2021; Park et al., 2021), composition with generative works (Schwarz et al., 2021; Niemeyer & Geiger, 2021; Wang et al., 2021a; Jain et al., 2022), or few-shot learning (Yu et al., 2021b; Jain et al., 2021; Rebain et al., 2022; Xu et al., 2022; Chibane et al., 2021; Wei et al., 2021; Chen et al., 2021).

2.2. Accelerating NeRF

One crucial drawback of NeRF is its slow convergence and rendering speed. To accelerate the training speed of NeRF, previous works have combined grid-based approaches which store view-direction-independent information on voxel grids. Liu et al. (2020) introduce a dense feature grid to reduce the computation burden of NeRF and progressively prunes the dense grids. The other works pre-compute and store a trained NeRF to the voxel grid, increasing rendering speed (Yu et al., 2021a; Hedman et al., 2021). On the other hand, rather than distilling the trained NeRF to voxel grids, direct learning of features on the voxel has been proposed (Sun et al., 2022; Wu et al., 2021; Wang et al., 2022; Fridovich-Keil et al., 2022).

While these methods have been successful in achieving near real-time neural rendering, they also come with drawbacks such as the increased model size and lower reconstruction quality caused by pre-storing the scene representation. To overcome these limitations, Müller et al. (2022) recently proposed Instant-NGP, which utilizes spatial hash functions and multi-resolution grids to approximate dense grid features and maximizes the hierarchical properties of 3D scenes.

This approach allows for state-of-the-art performance and the fastest convergence speed simultaneously.

2.3. NeRF with Pose Refinement

For the majority of neural rendering, it is crucial to have accurate camera intrinsic and extrinsic parameters. In an effort to address this issue, Yen-Chen et al. (2021) proposed a method for combining pose estimation and NeRFs by utilizing an inverted trained NeRF as an image-to-camera pose model. Subsequently, various methods for jointly optimizing camera pose parameters and 3D scene reconstruction have been proposed. Wang et al. (2021c) proposed a joint optimization problem in which the camera pose is represented as a 6-degree-of-freedom (DoF) matrix and optimized using a photometric loss. Building upon this, Xia et al. (2022) proposed a method that replaces ReLU-based multi-layer perceptrons (MLPs) with sine-based MLPs and employs an efficient ray batch sampling.

In addition to directly optimizing camera parameters, geometric-based approaches (Jeong et al., 2021; Lin et al., 2021; Chng et al., 2022) have also been suggested. For example, Lin et al. (2021) proposed BARF, which optimizes the warping matrix of the camera pose with a standard error back-propagation algorithm, utilizing curriculum training to adjust the spectral bias of the scene representation.

Unlike previous methods based on the original NeRF structure, our method is designed for grid-based approaches, especially for multi-resolution hash encoding, which shows outstanding performance in novel-view synthesis and its training speed. The common NeRF structure and its variants are prone to slowly converge, but our method can be converged significantly faster with state-of-the-art reconstruction performance under the circumstance of noisy or unknown camera poses.

3. Method

As mentioned in Section 1, we observed that a naïve error back-propagation for the camera pose refinement with multi-resolution hash encoding leads to inferior results compared to the use of sinusoidal encoding, (*e.g.*, Jeong et al., 2021; Lin et al., 2021). To further understand the observation, we analyze the derivative of the multi-resolution hash encoding (Section 3.1). We point out that the gradient fluctuation of the multi-resolution hash encoding makes it difficult to learn the pose refinement and scene reconstruction jointly (Section 3.2). To address these, we propose a method for calibrating inaccurate camera poses in multi-resolution hash encoding (Section 3.3). Additionally, we find that the multi-level decomposition of a scene induces the different convergence rates of multi-level encoding, which results in limited camera pose registration (Section 3.4).

3.1. Multi-Resolution Hash Encoding

This section describes the multi-resolution hash encoding presented by Müller et al. (2022), which we focus on.

3.1.1. Multi-Resolution Hash Encoding

As the combination of the multi-resolution decomposition and the grid-based approach with hashing mechanism, multi-resolution hash encoding is defined as a learnable mapping of input coordinate $\mathbf{x} \in \mathbb{R}^d$ to a higher dimension. The trainable encoding is learned as multi-level feature tables independent of each other.

The feature tables $\mathcal{H} = \{\mathcal{H}_l \mid l \in \{1, \dots, L\}\}$ are assigned to the L levels and each table contains T trainable feature vectors with the dimensionality of F. Each level consists of the d-dimensional grids where each dimension has N_l sizes considering multi-resolution. The number of grids for each size exponentially grows from the coarsest N_{min} to the finest resolutions N_{max}. Therefore, N_l is defined as follows:

$$b := \exp\left(\frac{\ln N_{\text{max}} - \ln N_{\text{min}}}{L-1}\right) \tag{1}$$

$$N_l := \lfloor N_{\text{min}} \cdot b^{l-1} \rfloor. \tag{2}$$

For given a specific level l, an input coordinate $\mathbf{x}$ is scaled by N_l and then a grid spans to a unit hypercube where $\lfloor \mathbf{x}_l \rfloor := \lfloor \mathbf{x} \cdot N_l \rfloor$ and $\lceil \mathbf{x}_l \rceil := \lceil \mathbf{x} \cdot N_l \rceil$ are the vertices of the diagonal. Then, each vertex is mapped into an entry in the level's respective feature table. Notice that, for coarse levels where the number of the total vertices of the grid is fewer than T, each vertex corresponds one-to-one to the table entry. Otherwise, each vertex corresponds to the element of the l^{th} table $\mathcal{H}_l$, whose table index is the output of the following spatial hash function (Teschner et al., 2003):

$$h(x) = \left(\bigoplus_{i=1}^{d} x_i \pi_i\right) \mod T, \tag{3}$$

where $\bigoplus$ denotes bitwise XOR and π_i are unique and large prime numbers. In each level, the 2^d feature vectors of the hypercube are d-linearly interpolated according to the relative position of $\mathbf{x}$. However, the interpolation enables us to get the gradient of the table entry since the interpolating weights are the function of $\mathbf{x}$. We will revisit this in the following section for analysis.

The output $\mathbf{y}$ of the multi-resolution hash encoding is the concatenation of the entire level-wise interpolated features and its dimensionality is $L \times F$. For simplicity, we denote as $\mathbf{y} = f(\mathbf{x}; \theta)$ with its trainable parameter θ. Similar to the other neural renderings using differentiable volume rendering (emission-absorption ray casting), the decoding MLP $m(\mathbf{y}; \phi)$ predicts the density and the non-Lambertian color along the ray. All trainable parameters are updated via photometric loss $\mathcal{L}$ between the rendered ray and the ground-truth color.

3.1.2. Derivative of Multi-Resolution Hash Encoding

For the gradient analysis, we derive the derivative of the multi-resolution hash encoding with respect to $\mathbf{x}$. Let $\mathbf{c}_{i,l}(\mathbf{x})$ denote the corner i of the level l resolution grid in which $\mathbf{x}_l$ is located, and let $h_l(\cdot)$ represent the hash function for the l-level, as defined in Equation (3).

Next, consider a function $\mathbf{h}_l : \mathbb{R}^d \rightarrow \mathbb{R}^F$, whose output is a l^{th} interpolated feature vector with 2^d corners, as given by,

$$\mathbf{h}_l(\mathbf{x}) = \sum_{i=1}^{2^d} w_{i,l} \cdot \mathcal{H}_l\big(h_l(\mathbf{c}_{i,l}(\mathbf{x}))\big), \tag{4}$$

where $w_{i,l}$ denotes the d-linear weight, which is defined by the *opposite* volume in a unit hypercube with the relative position of $\mathbf{x}$:

$$w_{i,l} = \prod_{j=1}^{d} \left(1 - |\mathbf{x}_l - \mathbf{c}_{i,l}(\mathbf{x})|_j\right) \tag{5}$$

where the index j indicates the j-th dimension in the vector. We can redefine the multi-resolution hash encoding vector $\mathbf{y}$ as follows:

$$\mathbf{y} = f(\mathbf{x};\theta) = \big[\mathbf{h}_1(\mathbf{x}); \ldots; \mathbf{h}_L(\mathbf{x})\big] \in \mathbb{R}^{F'} \tag{6}$$

where the dimension of the output vector is $F' = L \times F$ after the concatenation.

The Jacobian $\nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x}) \in \mathbb{R}^{F \times d}$ of the l^{th} interpolated feature vector $\mathbf{h}_l(\mathbf{x})$ with respect to the $\mathbf{x}$ can be derived using the chain-rule as follows:

$$\begin{aligned} \nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x}) &= \left[\frac{\partial \mathbf{h}_l(\mathbf{x})}{\partial x_1}, \ldots, \frac{\partial \mathbf{h}_l(\mathbf{x})}{\partial x_d}\right] \\ &= \sum_{i=1}^{2^d} \mathcal{H}_l\big(h_l(\mathbf{c}_{i,l}(\mathbf{x}))\big) \cdot \nabla_{\mathbf{x}} w_{i,l}, \end{aligned} \tag{7}$$

where $\mathcal{H}_l(g_l(\mathbf{c}_l(\mathbf{x})))$ is not differentiable with respect to $\mathbf{x}$ and the k-th element of $\nabla_{\mathbf{x}} w_{i,l} \in \mathbb{R}^{1\times d}$ is defined by,

$$\frac{\partial w_{i,l}(\mathbf{x})}{\partial x_k} = s_k \cdot \prod_{j \neq k} \left(1 - |\mathbf{x}_l - \mathbf{c}_{i,l}(\mathbf{x})|_j\right), \tag{8}$$

where s_k denotes $\mathit{sign}\big(\mathbf{c}_{i,l}(\mathbf{x}) - \mathbf{x}_l\big)_k$.

As seen in Equation (7), the Jacobian $\nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x})$ is the weighted sum of the hash table entries corresponding to the nearby corners of $\mathbf{x}$. However, the gradient $\nabla_{\mathbf{x}_k} w_{i,l}$ is not continuous at the corners due to the variable s_k, causing the direction of the gradient to flip. This oscillation of the gradient $\nabla_{\mathbf{x}} w_{j,l}$ is the source of gradient fluctuation, independently from $\mathcal{H}$. For a detailed analysis of the derivatives and further discussion, please refer to Appendix A.1.

3.2. Camera Pose Refinement

Camera pose can be represented as a transformation matrix from the camera coordinate to the world coordinate. Let us denote the camera-to-world transformation matrix as $\big[\mathbf{R}|\mathbf{t}\big] \in \mathrm{SE}(3)$, where $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^{3\times 1}$ are rotation matrix and translation vector, respectively.

3.2.1. Pose Refinement with the Sinusoidal Encoding

The pose refinement using error back-propagation in neural rendering is jointly optimizing the 6 DoF pose parameters and neural scene representation through the differentiable volume rendering:

$$\phi^*, \psi^* = \arg\min_{\phi,\psi} \mathcal{L}(\mathcal{I}, \hat{\mathcal{I}}; \phi, \psi), \tag{9}$$

where ϕ and ψ denote model parameters and trainable camera parameters, $\hat{\mathcal{I}}$ and $\mathcal{I}$ denote reconstructed color and its ground-truth color respectively.

Note that, to our knowledge, all previous works (Yen-Chen et al., 2021; Wang et al., 2021c; Xia et al., 2022; Jeong et al., 2021; Lin et al., 2021; Chng et al., 2022) of pose refinement in neural rendering utilize fully differentiable encoding with respect to the input coordinate (*e.g.*, sinusoidal or identity). However, they have limited performance compared to multi-resolution hash encoding (Müller et al., 2022).

3.2.2. Pose Refinement with Multi-Resolution Hash Encoding

Now, we present the optimization problem of pose refinement with multi-resolution hash encoding. Based on the Equation (9), we also directly optimize the camera pose parameters with multi-resolution hash encoding,

$$\phi^*, \theta^*, \psi^* = \arg\min_{\phi,\theta,\psi} \mathcal{L}(\mathcal{I}, \hat{\mathcal{I}}; \phi, \theta, \psi), \tag{10}$$

where θ is a trainable parameter for multi-resolution hash encoding, *i.e.*, the entries of the hash tables $\mathcal{H}_l$. However, we observe that the pose refinement and reconstruction quality from the above optimization problem is much worse than the previous works (Refer to (e) of Table 3).

To explain the poor performance, we assume that the gradient fluctuation of Equation (10), or Equation (8), negatively affects pose refinement. Since the input coordinate $\mathbf{x}$ is defined as a rigid transformation of the camera pose $\big[\mathbf{R}|\mathbf{t}\big]$ and image coordinate (projected in homogeneous space $z = -1$), the gradient fluctuation propagates through the gradient-based updates of the camera poses. We speculate that this fluctuation makes the joint optimization of the pose refinement and the scene reconstruction difficult. In Appendix A.2, we present more details of the camera pose refinement with the gradient-based optimization.

3.3. Non-linear Interpolation for Smooth Gradient

To mitigate the gradient fluctuation, we propose to use a *smooth gradient* for the interpolation weight $w_{i,l} \in [0, 1]$ maintaining forwarding pass, inspired by the straight-through estimator (Bengio et al., 2013).

For the smooth gradient, we use the activation function $\delta(w_{i,l})$ whose derivative is zero at the corners of the hypercube, and $w_{i,l} \in [0, 1]$,

$$\delta(w_{i,l}) = \frac{1 - \cos(\pi w_{i,l})}{2}, \tag{11}$$

where the activation value $\delta(w_{i,l})$ is ranged in $[0, 1]$.

As a result, the gradient of $\delta(w_{i,l})$ with respect to $\mathbf{x}$ is derived as follows:

$$\nabla_{\mathbf{x}}\delta(w_{i,l}) = \frac{\pi}{2}\sin(\pi w_{i,l}) \cdot \nabla_{\mathbf{x}} w_{i,l}. \tag{12}$$

Remind that $\nabla_{\mathbf{x}} w_{i,l}$ is not continuous and flipped through the boundary of a hypercube. In Equation (12), the weighting by the sine function effectively makes the gradient smooth and continuous (*ref.* Figure 2b). Moreover, the gradient of $\mathbf{x}$ near the boundary is relatively shrunk compared to the middle of the grids, which may prevent frequent back-and-forth across the boundary after camera pose updates.

However, we do not directly use this in the interpolation forward pass. The cosine function in Equation (11) unintentionally scatters the sampled points in a line toward the edges of the grids. This phenomenon, which we refer to as the "zigzag problem," can be addressed by the *straight-through* estimator (Bengio et al., 2013). It maintains the results of the linear interpolation in the forward pass by the cancel-out of the last two terms in Equation (13), and *partially* uses the activation value $\delta(w_{i,l})$ in the backward pass as follows:

$$\hat{w}_{i,l} = w_{i,l} + \lambda\delta(w_{i,l}) - \lambda\tilde{\delta}(w_{i,l}), \tag{13}$$

where λ is a hyperparameter that adjusts the smooth gradient and the zigzag problem, $\tilde{\delta}$ denotes the detached variable from the computational graph. The steps involved in the straight-through estimator are illustrated in Figure 1.

For an additional discussion, we present an illustration of the zigzag problem in Appendix A.3 (See Figure 4). Although this straight-through estimator does not perfectly make the gradient smooth and continuous with the addition in Equation (13), it is empirically more effective than other mixing variants (see Appendix A.3 and Table 7).

3.4. Curriculum Scheduling

As argued by Tancik et al. (2020); Landgraf et al. (2022), NeRFs exhibit a hierarchical structure, *i.e.*, the coordinate-based MLPs can suffer from spectral bias issues, in which different frequencies converge at different rates. Lin et al. (2021) further address this issue in the pose refinement. The research showed that the Jacobian of the k^{th} positional encoding amplifies pose noise, making the naïve application of positional encoding inappropriate for pose refinement.

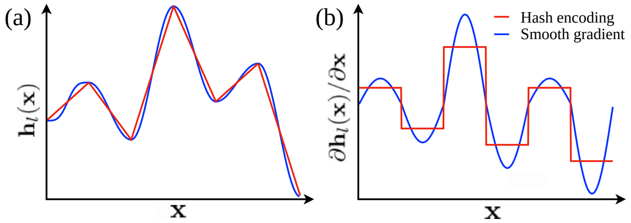

Figure 2. Illustration on the *smooth gradient* induced by Equation (12). We visualize the 1D case of the multi-resolution hash encoding $\mathbf{h}_l(\mathbf{x})$ and its derivative $\partial\mathbf{h}_l(\mathbf{x})/\partial\mathbf{x}$ in (a) and (b), respectively. The y-axis illustrates a table entry, *i.e.*, 1D hash encoding. For further discussion, please refer to the text and Appendix A.1.

We observe that the multi-resolution hash encoding, which leverages the multi-level decomposition of scenes, exhibits a similar problem. To resolve the problem, we propose a curriculum scheduling strategy to regulate the convergence rate of the level-wise encoding. We weight the l^{th} level of multi-resolution hash encoding $\mathbf{h}_l$ by

$$\tilde{\mathbf{h}}_l = r_l(t) \cdot \mathbf{h}_l, \tag{14}$$

where the weight of the encoding $r_l(t)$ is defined as

$$r_l(t) = \begin{cases} 0 & \alpha(t) < l \\ \frac{1-\cos((\alpha(t)-l)\pi)}{2} & 0 \le \alpha(t) - l < 1 \\ 1 & \text{otherwise}, \end{cases} \tag{15}$$

and $\alpha(t) = L \cdot \frac{t-t_s}{t_e-t_s} \in [0, L]$ is proportional to the number of iterations t in the scheduling interval $[t_s, t_e]$.

This weighting function is similar to the coarse-to-fine method proposed by Park et al. (2021) and Lin et al. (2021). This method allows the decoding network receives the encodings from all levels, while high-level encodings are more slowly updated than the coarse levels. We empirically found this multi-level coarse-to-fine strategy is effective in multi-resolution hash encoding.

4. Experiment

In this section, we validate our proposed method using the multi-resolution hash encoding (Müller et al., 2022) with inaccurate or unknown camera poses.

4.1. Implementation Details

4.1.1. Dataset

We evaluate the proposed method against the two previous works, BARF (Lin et al., 2021) and GARF (Chng et al., 2022). Since the implementation of GARF is unavailable, we re-implement GARF. Our re-implemented GARF has the same structure as BARF except for sinusoidal encoding and Gaussian activation. Following Lin et al. (2021) and Chng et al. (2022), we evaluate and compare our method on two public novel-view-synthesis datasets.

NeRF-Synthetic. NeRF-Synthetic (Mildenhall et al., 2020) has 8 synthetic object-centric scenes, which consist of 100 rendered images with ground-truth camera poses (intrinsic and extrinsic) for each scene. Following Lin et al. (2021), we utilize this dataset for the noisy camera pose scenario. To simulate the scenario of imperfect camera poses, we adopt the approach in Lin et al. (2021) synthetically perturbing the camera poses with additive Gaussian noise, $\delta\psi \sim \mathcal{N}(\mathbf{0},\ 0.15\mathbf{I})$.

LLFF. LLFF (Mildenhall et al., 2019) has 8 forward-facing scenes captured by a hand-held camera, including RGB images and camera poses that have been estimated using the off-the-shelf algorithm (Schönberger & Frahm, 2016). Following previous works, we utilize this dataset for the *unknown* camera pose scenario. Unlike the synthetic datasets, we initialize all camera poses with the *identity* matrix. Note that, the camera poses provided by **LLFF** are the estimations obtained using the COLMAP algorithm (Schönberger & Frahm, 2016). As such, the pose error measured in our quantitative results only indicates the agreement between the learned pose and the estimated pose using the classical geometry-based approach.

4.1.2. Implementation Details

For the multi-resolution hash encoding, we follow the approach of Instant-NGP (Müller et al., 2022), which uses a table size of $T = 2^{19}$ and a dimensionality of $F = 2$ for each level feature. Each feature table is initialized with a uniform distribution $\mathcal{U}[-1\text{e-}4,\ 1\text{e-}4]$. Note that we reproduce the entire training pipeline in PyTorch for pose refinement instead of using the original C++ & CUDA implementation of Instant-NGP for fair comparison [1].

The decoding network consists of 6-layer MLPs with ReLU (Glorot et al., 2011) activation and 256 hidden dimensions, including density network branch and color. We utilize the tiny-cuda-nn (tcnn) (Müller et al., 2021) framework for the decoding network. We present the other implementation details in Appendix B.1. While we set $\lambda = 1$ by default for the *straight-through estimator*, the other options are explored in Appendix A.3.

4.1.3. Evaluation Criteria

In conformity with previous studies (Lin et al., 2021; Chng et al., 2022), we evaluate the performance of our experiments in two ways: 1) the quality of view-synthesis for the 3D scene representation and 2) the accuracy of camera pose registration. We measure the PSNR, SSIM, and LPIPS scores for view-synthesis quality, as employed in the original NeRF (Mildenhall et al., 2020). The rotation and translation errors are defined as follows:

$$E(\mathbf{R}) = \cos^{-1}\left(\frac{tr(\mathbf{R}' \cdot \mathbf{R}^{\mathrm{T}}) - 1}{2}\right), \qquad (16)$$

$$E(\mathbf{t}) \ = |\mathbf{t}' - \mathbf{t}|_2^2, \qquad (17)$$

where $\left[\mathbf{R}'|\mathbf{t}'\right] \in \mathrm{SE}(3)$ denotes the ground-truth camera-to-world transformation matrix and $tr(\cdot)$ denotes trace operator. Like the Lin et al. (2021), all the metrics are measured after the pre-alignment stage using the Procrustes analysis. In experiments, all the camera poses ψ are parameterized by the $\mathfrak{se}(3)$ Lie algebra with known intrinsics.

4.2. Quantitative Results

4.2.1. Synthetic Objects in NeRF-Synthetic

Table 1 demonstrates the quantitative results of the NeRF-Synthetic. In Table 1, the proposed method achieves state-of-the-art performances in both pose registration and reconstruction fidelity across all scenes. The results align with Müller et al. (2022) showing impressive performance on the scenes with high geometric details.

On the other hand, Müller et al. (2022) previously demonstrated that multi-resolution hash encoding is limited to the scenes with complex and view-dependent reflections, *i.e.*, *Materials*. Although they attributed this limitation to their shallow decoding networks, we observed similar performance when utilizing deeper decoding networks. We hypothesize that frequency-based encodings, such as sinusoidal or spherical harmonics, might be more appropriate for addressing complex and view-dependent reflections. We will further investigate this issue in future work.

4.2.2. Real-World Scenes in LLFF

We report the quantitative results of the LLFF dataset in Table 2. Note that GARF utilizes 6-layer decoding networks for this dataset. In Table 2, the proposed method outperforms the previous methods regarding reconstruction fidelity and pose recovery, especially for translation. These results suggest that the learned pose from our method is closely

[1] While our re-implementation performs almost the same with the original, it takes slightly longer training time due to PyTorch's execution latency. The performance of our re-implemented Instant-NGP is reported in Appendix B.1.

Table 1. Quantitative results of the NeRF-Synthetic dataset.

Scene	Camera Pose Registration						View Synthesis Quality								
	Rotation (°) ↓			Translation ↓			PSNR ↑			SSIM ↑			LPIPS ↓		
	GARF	BARF	Ours	GARF	BARF	Ours	GARF	BARF	Ours	GARF	BARF	Ours	GARF	BARF	Ours
Chair	0.113	0.096	**0.085**	0.549	0.428	**0.365**	31.32	31.16	**31.95**	0.959	0.954	**0.962**	0.042	0.044	**0.036**
Drum	0.052	0.043	**0.041**	0.232	0.225	**0.214**	24.15	23.91	**24.16**	0.909	0.900	**0.912**	0.097	0.099	**0.087**
Ficus	0.081	0.085	**0.079**	**0.461**	0.474	0.479	26.29	26.26	**28.31**	0.935	0.934	**0.943**	0.057	0.058	**0.051**
Hotdog	0.235	0.248	**0.229**	**1.123**	1.308	**1.123**	34.69	34.54	**35.41**	0.972	0.970	**0.981**	0.029	0.032	**0.027**
Lego	0.101	0.082	**0.071**	0.299	0.291	**0.272**	29.29	28.33	**31.65**	0.925	0.927	**0.973**	0.051	0.050	**0.036**
Materials	**0.842**	0.844	0.852	**2.688**	2.692	2.743	**27.91**	27.84	27.14	**0.941**	0.936	0.911	0.059	**0.058**	0.062
Mic	0.070	0.071	**0.068**	0.293	0.301	**0.287**	31.39	31.18	**32.33**	0.971	0.969	**0.975**	0.047	0.048	**0.043**
Ship	**0.073**	0.075	0.079	0.310	0.326	**0.287**	27.64	27.50	**27.92**	0.862	0.849	**0.879**	0.119	0.132	**0.110**
Mean	0.195	0.193	**0.189**	0.744	0.756	**0.722**	28.96	28.84	**29.86**	0.935	0.930	**0.943**	0.063	0.065	**0.056**

Table 2. Quantitative results of the LLFF dataset.

Scene	Camera Pose Registration						View Synthesis Quality								
	Rotation (°) ↓			Translation ↓			PSNR ↑			SSIM ↑			LPIPS ↓		
	GARF	BARF	Ours	GARF	BARF	Ours	GARF	BARF	Ours	GARF	BARF	Ours	GARF	BARF	Ours
Fern	0.470	0.191	**0.110**	0.250	**0.102**	**0.102**	24.51	23.79	**24.62**	0.740	0.710	**0.743**	0.290	0.311	**0.285**
Flower	0.460	**0.251**	0.301	0.220	0.224	**0.211**	**26.40**	23.37	25.19	**0.790**	0.698	0.744	**0.110**	0.211	0.128
Fortress	**0.030**	0.479	0.211	0.270	0.364	**0.241**	29.09	29.08	**30.14**	0.820	0.823	**0.901**	0.150	0.132	**0.098**
Horns	**0.030**	0.304	0.049	0.210	0.222	**0.209**	22.54	22.78	**22.97**	0.690	0.727	**0.736**	0.330	0.298	**0.290**
Leaves	**0.130**	1.272	0.840	0.230	0.249	**0.228**	**19.72**	18.78	19.45	**0.610**	0.537	0.607	0.270	0.353	**0.269**
Orchids	0.430	0.627	**0.399**	0.410	0.404	**0.386**	19.37	19.45	**20.02**	0.570	0.574	**0.610**	0.260	0.291	**0.213**
Room	**0.270**	0.320	0.271	**0.200**	0.270	0.213	31.90	31.95	**32.73**	0.940	0.949	**0.968**	0.130	0.099	**0.098**
T-Rex	**0.420**	1.138	0.894	**0.360**	0.720	0.474	22.86	22.55	**23.19**	0.800	0.767	**0.866**	0.190	0.206	**0.183**
Mean	**0.280**	0.573	0.384	0.269	0.331	**0.258**	24.55	23.97	**24.79**	0.745	0.723	**0.772**	0.216	0.227	**0.197**

related to that of the classical geometric algorithm, indicating that our proposed method can learn camera poses from scratch using the multi-resolution hash encoding.

In terms of rotation angle registration, our method outperforms BARF, achieving comparable performance to GARF. Still, notice that our method achieves the best view-synthesis quality compared to the other methods. Also, in Table 4, we investigate the interaction with the COLMAP camera pose initialization and our method. Please refer to Appendix B.2 for the details. Here, the underbar denotes runners-up.

4.3. Ablation Study

We present additional ablation studies to examine the proposed method's effectiveness. Similar to the Instant-NGP (Müller et al., 2022), all the following experiments are conducted on the *Hotdog* in the NeRF-Synthetic dataset for comparison. Note that other scenes behave similarly.

4.3.1. Component Analysis

In Table 3, we perform the ablation study for our method to examine the role of each element. As shown in row (b) compared with (c), the *smooth gradient* significantly helps with pose refinements, resulting in more accurate pose registration and higher view-synthesis quality. Also, from (a) and (b), we observe that the *straight-through estimator* prevents unintentional jittering from the non-linear weighting showing outperformance. Lastly, as shown in (a) and (d), our proposed multi-level learning rate scheduling reasonably enhances pose estimation and scene reconstruction qualities.

4.3.2. Time Complexity

In Table 5, we visualize the comparison of the training speed between the proposed method and the previous works (Lin et al., 2021; Chng et al., 2022). By utilizing fast convergence of multi-resolution hash encoding, the proposed method achieves more than 20× faster training speed compared to the previous works. Remind that the proposed method outperforms previous methods both in pose registration and view synthesis.

4.3.3. Decoder Size

Here, we examine the design criteria for decoding networks $m(\mathbf{y};\phi)$ in terms of model capacity. The original implementation of Instant-NGP (Müller et al., 2022) utilizes shallow decoding networks, resulting in the feature table $\mathcal{H}$ having a relatively larger number of learnable parameters than the decoding networks, *i.e.*, $|\theta| \gg |\phi|$. We find that this often leads to the suboptimal convergence of both the multi-resolution hash encoding and the camera pose registration.

Figure 3 presents the view-synthesis quality with respect

Table 3. Ablation study on the components of the proposed method. Experiments are conducted on the *Hotdog* in the NeRF-Synthetic dataset. Three components are the straight-through estimator in Equation (13), the smooth gradient with cosine activation in Equation (11), and the curriculum scheduling in Sec. 3.4.

	Component Ablation			**Evaluation Metric**		
	w/ Straight-Through	*w/ Smooth Grad.*	*w/ Curriculum Scheduling*	Rotation (°) ↓	Translation ↓	PSNR ↑
(a)	✓	✓	✓	**0.234**	**1.124**	**35.41**
(b)		✓	✓	0.245	1.130	35.03
(c)			✓	0.977	3.210	29.89
(d)	✓	✓		0.447	1.921	32.19
(e)				2.779	6.423	25.41

Table 4. Quantitative results of the proposed method in the LLFF dataset with the COLMAP initialization (PSNR ↑).

	Experimental Setting		LLFF								
	w/ COLMAP	*w/* Pose Refinement	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Rex	*Average*
(a)	✓		25.83	26.56	28.00	26.46	18.89	20.15	31.96	26.51	25.55
(b)		✓	24.62	25.19	30.14	22.97	19.45	20.02	32.73	23.19	24.79
(c)	✓	✓	**26.41**	**28.00**	**30.99**	**27.35**	**19.97**	**21.26**	**33.02**	**26.83**	**26.73**

Table 5. Comparison of the time complexity on the *Hotdog* in the NeRF-Synthetic dataset. In the averaged training time, our method takes only 10.8 ms, significantly faster than the previous works, GARF and BARF, which are 213 ms and 252 ms, respectively.

Method	Total Training Time	Time/Iteration
BARF	14 hours	252 ms
GARF	12 hours	218 ms
Ours	40 mins	10.8 ms

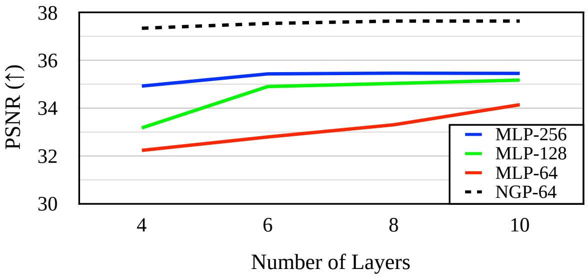

Figure 3. Performance depends on the decoder size. We plot as the depth of the decoder increases, varying the hidden size from 64 to 256. The dashed line denotes the NGP's with the hidden size of 64 using the ground-truth camera poses as the upper bound.

to varying model sizes of the decoding network. Unlike the findings of Müller et al. (2022), who did not observe improvement with deeper decoder MLPs (as shown by the dashed line in the Figure), we observe that the decoder size heavily impacts both the view synthesis and the pose registration. Therefore, in cases where the camera pose is inaccurate, we assume a sufficient number of parameters in the decoder is necessary. Informed by this analysis, we employ deeper and wider decoding networks than the original Instant-NGP: 6-layer MLPs with 256 neurons. Note that competitive methods (Lin et al., 2021; Chng et al., 2022) utilize a deeper decoder network with 8-layer MLPs with 256 neurons having more parameters.

4.4. Noise Robustness

In the previous section, we mainly validated LLFF with an unknown camera pose and NeRF-Synthetic with an imperfect camera pose. However, the variability in camera poses is relatively insignificant due to the inherent constraints of LLFF, which only allows for forward-facing perspectives. It is also for the NeRF-Synthetic, which synthetically perturbs the camera pose with a fixed certain level of the Gaussian distribution of $\delta\psi \sim \mathcal{N}(\mathbf{0},\ 0.15\mathbf{I})$. Therefore, we recognize the significance of validating our method's robustness against varying noise levels. We additionally evaluated the performance of the Hotdog (NeRF-synthetic) dataset under different noise levels.

Table 6. Quantitative results of the proposed method with varying noise levels.

$\delta\psi$	**Evaluation Metric**		
	Rotation (°) ↓	Translation ↓	PSNR ↑
$\mathcal{N}(\mathbf{0},\ 0.15\mathbf{I})$	0.234	1.124	35.41
$\mathcal{N}(\mathbf{0},\ 0.30\mathbf{I})$	0.462	2.111	32.03
$\mathcal{N}(\mathbf{0},\ 0.60\mathbf{I})$	61.90	105.0	7.61

As shown in Table 6, the pose reconstruction fails to converge beyond 60 degrees of noise. At such a high noise level, the deviation from the ground-truth pose is severe to cause significant updates to the hash encoding as rendered image changes, resulting in inaccurate pose refining that does not concord with scene reconstruction and pose refinement.

4.5. Qualitative Results

We present the qualitative results of our method compared with competitive methods. Please refer to Appendix B.5.

5. Conclusion

We investigate the joint optimization of camera poses and scene reconstruction using multi-resolution hash encoding. Based on the careful analysis of the gradient fluctuation of the hash encoding, we propose a simple yet effective method of the straight-through estimator for gradient smoothing. Additionally, we consider the spectral bias of multi-level decomposition and adopt a multi-level learning rate scheduling varying convergence rates of the multi-level encodings. Our extensive experiments show that the proposed method successfully recovers the camera poses with state-of-the-art performance on NeRF-Synthetic and LLFF. Nevertheless, with its state-of-the-art performance and fast convergence, we believe our method is a reasonable choice for real-world problems with imperfect or unknown camera poses.

Limitations. The proposed method shows limited performance in the scenes with complex reflections inherited from the multi-resolution hash encoding (Müller et al., 2022). For a large deviation of the pose noise, *e.g.*, more than 60 degrees, our method did not converge appropriately. In the absence of additional methods such as COLMAP (Schönberger & Frahm, 2016), or the use of geometric loss as introduced in SC-NeRF (Jeong et al., 2021), only the gradient-based pose refinement remains a challenging problem in NeRF. This issue shall be addressed in our future research endeavors.

6. Acknowledgements

Most of this work was done while Hwan Heo and Soohyun Kim were research interns at NAVER AI Lab. The NAVER Smart Machine Learning (NSML) platform (Kim et al., 2018) has been used in the experiments.

This research was partly supported by the MSIT (Ministry of Science and ICT), Korea, under the ICT Creative Consilience program (IITP-2023-2020-0-01819) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

References

Barron, J. T., Mildenhall, B., Tancik, M., Hedman, P., Martin-Brualla, R., and Srinivasan, P. P. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

Barron, J. T., Mildenhall, B., Verbin, D., Srinivasan, P. P., and Hedman, P. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

Bengio, Y., Léonard, N., and Courville, A. C. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, 2013.

Boss, M., Braun, R., Jampani, V., Barron, J. T., Liu, C., and Lensch, H. P. Nerd: Neural reflectance decomposition from image collections. In *ICCV*, 2021.

Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., and Su, H. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *ICCV*, 2021.

Chibane, J., Bansal, A., Lazova, V., and Pons-Moll, G. Stereo radiance fields (srf): Learning view synthesis from sparse views of novel scenes. In *CVPR*, 2021.

Chng, S.-F., Ramasinghe, S., Sherrah, J., and Lucey, S. Garf: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation. In *ECCV*, 2022.

Fridovich-Keil, S., Yu, A., Tancik, M., Chen, Q., Recht, B., and Kanazawa, A. Plenoxels: Radiance fields without neural networks. In *CVPR*, 2022.

Glorot, X., Bordes, A., and Bengio, Y. Deep sparse rectifier neural networks. In *NeurIPS*, 2011.

Hedman, P., Srinivasan, P. P., Mildenhall, B., Barron, J. T., and Debevec, P. Baking neural radiance fields for real-time view synthesis. In *CVPR*, 2021.

Jain, A., Tancik, M., and Abbeel, P. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021.

Jain, A., Mildenhall, B., Barron, J. T., Abbeel, P., and Poole, B. Zero-shot text-guided object generation with dream fields. In *CVPR*, 2022.

Jeong, Y., Ahn, S., Choy, C., Anandkumar, A., Cho, M., and Park, J. Self-calibrating neural radiance fields. In *ICCV*, 2021.

Kajiya, J. T. and Herzen, B. P. V. Ray tracing volume densities. *ACM Trans. Graph.*, 1984.

Karnewar, A., Ritschel, T., Wang, O., and Mitra, N. Relu fields: The little non-linearity that could. *ACM Trans. Graph.*, 2022.

Kim, H., Kim, M., Seo, D., Kim, J., Park, H., Park, S., Jo, H., Kim, K., Yang, Y., Kim, Y., et al. Nsml: Meet the mlaas platform with a real-world case study. *arXiv preprint arXiv:1810.09957*, 2018.

Landgraf, Z., Hornung, A. S., and Cabral, R. S. Pins: Progressive implicit networks for multi-scale neural representations. In *ICML*, 2022.

Lin, C.-H., Ma, W.-C., Torralba, A., and Lucey, S. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021.

Liu, L., Gu, J., Lin, K. Z., Chua, T.-S., and Theobalt, C. Neural sparse voxel fields. *NeurIPS*, 2020.

Martin-Brualla, R., Radwan, N., Sajjadi, M. S. M., Barron, J. T., Dosovitskiy, A., and Duckworth, D. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021.

Mildenhall, B., Srinivasan, P. P., Ortiz-Cayon, R., Kalantari, N. K., Ramamoorthi, R., Ng, R., and Kar, A. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Trans. Graph.*, 2019.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

Müller, T., Rousselle, F., Novák, J., and Keller, A. Real-time neural radiance caching for path tracing. *ACM Trans. Graph.*, 2021.

Müller, T., Evans, A., Schied, C., and Keller, A. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 2022.

Neff, T., Stadlbauer, P., Parger, M., Kurz, A., Mueller, J. H., Chaitanya, C. R. A., Kaplanyan, A. S., and Steinberger, M. DONeRF: Towards Real-Time Rendering of Compact Neural Radiance Fields using Depth Oracle Networks. *Computer Graphics Forum*, 2021.

Niemeyer, M. and Geiger, A. Giraffe: Representing scenes as compositional generative neural feature fields. In *CVPR*, 2021.

Oechsle, M., Peng, S., and Geiger, A. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *ICCV*, 2021.

Park, K., Sinha, U., Barron, J. T., Bouaziz, S., Goldman, D. B., Seitz, S. M., and Martin-Brualla, R. Nerfies: Deformable neural radiance fields. In *ICCV*, 2021.

Pumarola, A., Corona, E., Pons-Moll, G., and Moreno-Noguer, F. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021.

Rebain, D., Jiang, W., Yazdani, S., Li, K., Yi, K. M., and Tagliasacchi, A. Derf: Decomposed radiance fields. In *CVPR*, 2021.

Rebain, D., Matthews, M., Yi, K. M., Lagun, D., and Tagliasacchi, A. Lolnerf: Learn from one look. In *CVPR*, 2022.

Schwarz, K., Liao, Y., Niemeyer, M., and Geiger, A. Graf: Generative radiance fields for 3d-aware image synthesis. In *CVPR*, 2021.

Schönberger, J. L. and Frahm, J.-M. Structure-from-motion revisited. In *CVPR*, 2016.

Srinivasan, P. P., Deng, B., Zhang, X., Tancik, M., Mildenhall, B., and Barron, J. T. Nerv: Neural reflectance and visibility fields for relighting and view synthesis. In *CVPR*, 2021.

Sun, C., Sun, M., and Chen, H. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022.

Tancik, M., Srinivasan, P. P., Mildenhall, B., Fridovich-Keil, S., Raghavan, N., Singhal, U., Ramamoorthi, R., Barron, J. T., and Ng, R. Fourier features let networks learn high frequency functions in low dimensional domains. In *NeurIPS*, 2020.

Teschner, M., Heidelberger, B., Müller, M., Pomerantes, D., and Gross, M. H. Optimized spatial hashing for collision detection of deformable objects. In *VMV*, 2003.

Wang, C., Chai, M., He, M., Chen, D., and Liao, J. Clip-nerf: Text-and-image driven manipulation of neural radiance fields. In *CVPR*, 2021a.

Wang, L., Zhang, J., Liu, X., Zhao, F., Zhang, Y., Zhang, Y., Wu, M., Xu, L., and Yu, J. Fourier plenoctrees for dynamic radiance field rendering in real-time. In *CVPR*, 2022.

Wang, P., Liu, L., Liu, Y., Theobalt, C., Komura, T., and Wang, W. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *NeurIPS*, 2021b.

Wang, Z., Wu, S., Xie, W., Chen, M., and Prisacariu, V. A. NeRF−−: Neural radiance fields without known camera parameters. *arXiv*, 2021c.

Wei, Y., Liu, S., Rao, Y., Zhao, W., Lu, J., and Zhou, J. Nerfingmvs: Guided optimization of neural radiance fields for indoor multi-view stereo. In *ICCV*, 2021.

Wu, L., Lee, J. Y., Bhattad, A., Wang, Y., and Forsyth, D. Diver: Real-time and accurate neural radiance fields with deterministic integration for volume rendering. In *CVPR*, 2021.

Xia, Y., Tang, H., Timofte, R., and Van Gool, L. Sinerf: Sinusoidal neural radiance fields for joint pose estimation and scene reconstruction. In *BMVC*, 2022.

Xu, D., Jiang, Y., Wang, P., Fan, Z., Shi, H., and Wang, Z. Sinnerf: Training neural radiance fields on complex scenes from a single image. In *ECCV*, 2022.

Yariv, L., Gu, J., Kasten, Y., and Lipman, Y. Volume rendering of neural implicit surfaces. In *NeurIPS*, 2021.

Yen-Chen, L., Florence, P., Barron, J. T., Rodriguez, A., Isola, P., and Lin, T.-Y. iNeRF: Inverting neural radiance fields for pose estimation. In *IROS*, 2021.

Yu, A., Li, R., Tancik, M., Li, H., Ng, R., and Kanazawa, A. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021a.

Yu, A., Ye, V., Tancik, M., and Kanazawa, A. pixelNeRF: Neural radiance fields from one or few images. In *CVPR*, 2021b.

Zhang, K., Riegler, G., Snavely, N., and Koltun, V. Nerf++: Analyzing and improving neural radiance fields. *arXiv*, 2020.

Zhou, Y., Barnes, C., Lu, J., Yang, J., and Li, H. On the continuity of rotation representations in neural networks. In *CVPR*, 2018.

A. The Derivative of Multi-Resolution Hash Encoding

We describe the details of the gradient-based optimization of the camera poses in the multi-resolution hash encoding.

A.1. Jacobian of the Interpolated Feature Vector

Borrowed from the Equation (7) in the manuscript, the Jacobian $\nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x}) \in \mathbb{R}^{F\times d}$ of the l^{th} interpolated feature vector $\mathbf{h}_l(\mathbf{x})$ with respect to $\mathbf{x}$ can be derived using the chain-rule as follows:

$$\begin{aligned}\nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x}) &= \left[\frac{\partial \mathbf{h}_l(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial \mathbf{h}_l(\mathbf{x})}{\partial x_d}\right] \\ &= \sum_{i=1}^{2^d} \mathcal{H}_l\big(h_l\big(\mathbf{c}_{i,l}(\mathbf{x})\big)\big) \cdot \left[\frac{\partial w_{i,l}(\mathbf{x})}{\partial x_1}, \dots, \frac{\partial w_{i,l}(\mathbf{x})}{\partial x_d}\right].\end{aligned} \tag{18}$$

Let $\bar{i}$ be one of the nearest corner indices from $\mathbf{c}_{i,l}$ in a unit hypercube, where $\mathbf{c}_{i,l}$ and $\mathbf{c}_{\bar{i},l}$ make an edge of the unit hypercube. Among the 2^d corners, we have 2^{d-1} pairs like that. Then, we have the relation for $w_{\bar{i}_k,l}$ as follows:

$$\frac{\partial w_{\bar{i}_k,l}(\mathbf{x})}{\partial x_k} = -\frac{\partial w_{i,l}(\mathbf{x})}{\partial x_k}, \tag{19}$$

which can be inferred from Equation (8) since the relative positions of $\mathbf{x}$ are different for the two cases.

Using this relation and the appropriate choice of the indices, the k^{th} element of Jacobian $\nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x})$ can be rewritten as follows:

$$\begin{aligned}\frac{\partial \mathbf{h}_l(\mathbf{x})}{\partial x_k} &= \sum_{i=1}^{2^d} \mathcal{H}_l\big(h_l\big(\mathbf{c}_{i,l}(\mathbf{x})\big)\big) \cdot \frac{\partial w_{i,l}(\mathbf{x})}{\partial x_k} \\ &= \sum_{i=1}^{2^{d-1}} \Big(\mathcal{H}_l\big(h_l\big(\mathbf{c}_{i,l}(\mathbf{x})\big)\big) - \mathcal{H}_l\big(h_l\big(\mathbf{c}_{\bar{i}_k,l}(\mathbf{x})\big)\big)\Big) \cdot \frac{\partial w_{i,l}(\mathbf{x})}{\partial x_k} \\ &= \sum_{i=1}^{2^{d-1}} \Big(\mathcal{H}_l\big(h_l\big(\mathbf{c}_{i,l}(\mathbf{x})\big)\big) - \mathcal{H}_l\big(h_l\big(\mathbf{c}_{\bar{i}_k,l}(\mathbf{x})\big)\big)\Big) \cdot \prod_{j\neq k}\left(1 - |\mathbf{x}_l - \mathbf{c}_{i,l}(\mathbf{x})|_j\right),\end{aligned} \tag{20}$$

where $\prod_{j\neq k}\left(1 - |\mathbf{x}_l - \mathbf{c}_{i,l}(\mathbf{x})|_j\right)$ and the differences between the hash table entries are constant to the x_k, which make $\partial\mathbf{h}_l(\mathbf{x})/\partial x_k$ is constant along with the k^{th} axis of the unit hypercube. Notice that the last term can be seen as the weights defined as:

$$\sum_{i=1}^{2^{d-1}} \prod_{j\neq k}\left(1 - |\mathbf{x}_l - \mathbf{c}_{i,l}(\mathbf{x})|_j\right) = 1, \tag{21}$$

where $\partial\mathbf{h}_l(\mathbf{x})/\partial x_k$ is the convex combination of the differences between two hash table entries.

Our speculation from this analysis is that $\partial\mathbf{h}_l(\mathbf{x})/\partial x_k$ is the linear slope defined by the differences among the hash table entries consist of each grid. However, the slope is sharply changed when crossing the boundary of the grid. For the one-dimensional case, this is a kind of *triangular wave* [2], which is a periodic, piecewise linear, continuous real function. This function has sharp local minima and maxima at the vertices of the wave, hindering appropriate gradient steps for accurate pose refinement.

Our method is proposed to remedy this problem. We empirically show that the smaller gradient for the positions near the boundaries compared with the other positions is significantly helpful for accurate pose refinement.

A.2. Camera-to-World Transformation

This section shows that the gradient fluctuation of the Jacobian $\nabla_{\mathbf{x}}\mathbf{h}_l(\mathbf{x})$ also affects the gradient-based optimization of the camera poses by the chain-rule.

[2] https://en.wikipedia.org/wiki/Triangle_wave

Remind that we denote the camera-to-world transformation matrix as $\left[\mathbf{R}|\mathbf{t}\right] \in \mathrm{SE}(3)$, where $\mathbf{R} \in \mathrm{SO}(3)$ and $\mathbf{t} \in \mathbb{R}^{3\times1}$, which are rotation matrix and translation vector, respectively. Typically, the translation vector $\mathbf{t}$ is expressed as a 3D vector in Euclidean space. Whereas for the rotation matrix on $\mathrm{SO}(3)$ space, previous works adopt the 3 DoF axis-angle representation with the Rodrigues' formula or the 6D continuous representation (Zhou et al., 2018; Jeong et al., 2021).

Consider an input coordinate of the image space as $\mathbf{x}_I \in \mathbb{R}^2$. To transform the image coordinate to the world coordinate, the image coordinate is projected to its homogeneous space $z = -1$. Then, we get the corresponding world coordinate $\mathbf{x} \in \mathbb{R}^{3\times1}$ by the rigid transformation of the camera pose as follows:

$$\mathbf{x} = \mathbf{R}\begin{bmatrix}\mathbf{x}_I \\ -1\end{bmatrix} + \mathbf{t}. \tag{22}$$

In volume rendering, the emission-absorption ray casting utilizes the transformed coordinate $\mathbf{x}$ to sample the points $\mathbf{x}'$ in a casted ray. Therefore, the derivative of camera pose parameters with respect to the $\mathbf{x}'$ is derived as follows:

$$\nabla_{\mathbf{R}}\mathbf{h}_l(\mathbf{x}') = \frac{\partial \mathbf{h}_l(\mathbf{x})}{\partial \mathbf{x}'} \cdot \frac{\partial \mathbf{x}'}{\partial \mathbf{R}}, \tag{23}$$

$$\nabla_{\mathbf{t}}\mathbf{h}_l(\mathbf{x}') = \frac{\partial \mathbf{h}_l(\mathbf{x}')}{\partial \mathbf{x}'} \cdot \frac{\partial \mathbf{x}'}{\partial \mathbf{t}}. \tag{24}$$

As shown in the above equation, the relations between $\nabla_{\mathbf{x}'}\mathbf{h}_l(\mathbf{x}')$ and, the Jacobian of $\nabla_{\mathbf{R}}\mathbf{h}_l(\mathbf{x}')$ and $\nabla_{\mathbf{R}}\mathbf{h}_l(\mathbf{x})$ are linear. Therefore, the gradient fluctuation of the $\nabla_{\mathbf{x}'}\mathbf{h}_l(\mathbf{x}')$ directly propagates to the gradient with respect to the camera poses, resulting in inappropriate updates for the pose refinement.

Comparison with Sinusoidal Encoding. The main difference of the pose refinement using multi-resolution hash encoding from sinusoidal encoding is *unsmoothness*. In the case of the sinusoidal encoding (Lin et al., 2021; Jeong et al., 2021), the encoding output of the Equation (4) is replaced with a differentiable encoding function $\gamma(\mathbf{x})$. Rather than linearly approximating the feature of $\mathbf{x}$ within a unit hypercube, it is directly encoded using the sinusoidal encoding as follows:

$$\mathbf{h}_l(\mathbf{x}) = \gamma_l(\mathbf{x}) = \left[\cos(2^l\pi\mathbf{x}),\ \sin(2^l\pi\mathbf{x})\right]. \tag{25}$$

Unlike the multi-resolution hash encoding, it induces the smooth gradient with respect to the input coordinate $\mathbf{x}$.

A.3. Zigzag Problem and the Straight-Through Estimator

As discussed in the manuscript, if we use smooth interpolation weight directly, the cosine function in Equation (11) unintentionally scatters the sampled points, which we refer to as the zigzag problem. We present an illustration of the zigzag problem in Figure 4. As shown on the right side of the figure, if we directly use the smooth interpolation in a hypercube, the sampled points get closer to the edges.

To remedy this problem, we leverage the straight-through estimator where the hyperparameter λ adjusts the "zigzag problem" between the original weight and the activation value. We explore the λ in Table 7 to investigate this impact.

Table 7. Hyperparameter Search for the λ. Experiments are conducted on the *Hotdog* in the NeRF-Synthetic.

	λ	**Evaluation Metric**		
		Rotation (°) ↓	Translation ↓	PSNR ↑
(a)	0.5	0.433	2.003	32.14
(b)	1	0.234	1.124	35.41
(c)	2	0.229	1.113	35.48
(d)	4	0.241	1.128	35.15

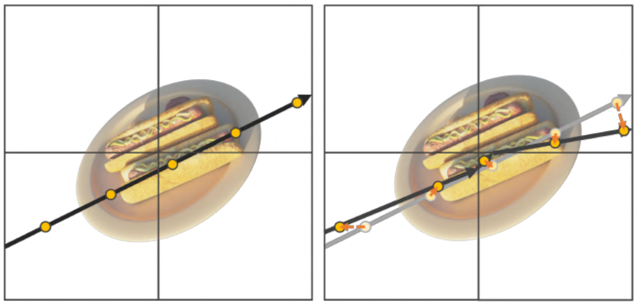

Figure 4. Illustration about "zigzag problem" of the smooth interpolation. Left: ray casting with linear interpolation, Right: ray casting with smooth interpolation in Equation (11). As shown on the right, if we directly use smooth interpolation in a certain hypercube, the sampled points are getting closer to the edges.

B. Additional Experiments

B.1. Re-implementation of Instant-NGP

B.1.1. Implementation Details

We follow the implementation settings from the original Instant-NGP (Müller et al., 2022) and BARF (Lin et al., 2021) except for the number of layers (Refer to Section 4.3). For the NeRF-Synthetic and the LLFF, we resize the images for each dataset to 400×400 and 480×640, respectively. We use the sinusoidal encoding with 4 levels for view-direction encoding, employed by Mildenhall et al. (2020). For ray casting, we randomly sample 1024 rays and $N = 128$ samples per ray at each optimization step. Notice that the sum of all weights normalizes all the reparameterized weights to ensure that their sum equals one. We use the Adam optimizer and train all models for 200K iterations, with a learning rate of 5×10^{-4} that exponentially decays to 1×10^{-4}. For multi-level learning rate scheduling, we set the scheduling interval $[t_s, t_e] = [20\text{K}, 100\text{K}]$, which is between 10% to 50% progress of the entire training.

B.1.2. Quantitative Results

This section presents the performance of our re-implemented version of multi-resolution hash encodings. The following experiments were conducted with ground-truth camera poses to compare it with the original Instant-NGP (Müller et al., 2022). In Table 8 and Table 9, we confirm that our implemented version gets similar results compared with the original.

Table 8. Quantitative results of our re-implemented multi-resolution hash encoding (PSNR ↑) on the NeRF-Synthetic.

Model	NeRF-Synthetic								
	Chair	Drum	Ficus	Hotdog	Lego	Materials	Mic	Ship	*Average*
Müller et al. (2022)	**35.00**	26.02	33.51	**37.40**	**36.39**	**29.78**	**36.22**	**31.10**	**33.18**
Ours	34.91	**26.13**	**33.52**	37.19	36.23	29.69	36.17	31.08	33.12

Table 9. Quantitative results of our re-implemented multi-resolution hash encoding (PSNR ↑) on the LLFF.

Model	LLFF								
	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Res	*Average*
Müller et al. (2022)	**25.87**	26.52	27.96	**26.60**	**18.93**	**20.31**	**31.96**	26.50	**25.58**
Ours	25.83	**26.56**	**28.00**	26.46	18.89	20.15	**31.96**	**26.51**	25.55

B.2. Real World Scenes from COLMAP Initialization

The manuscript investigates the challenging problem of jointly optimizing unknown camera poses and scene reconstruction using multi-resolution hash encoding. In this section, we conduct additional experiments on the LLFF dataset with the COLMAP (Schönberger & Frahm, 2016) camera pose initialization. Given that the estimated poses from COLMAP are inaccurate, we further refine the provided poses through the joint optimization of camera poses and scene reconstructions.

Table 4. Quantitative results of the proposed method in the LLFF dataset with the COLMAP initialization (PSNR ↑).

	Experimental Setting		LLFF								
	w/ COLMAP	*w/* Pose Refinement	Fern	Flower	Fortress	Horns	Leaves	Orchids	Room	T-Rex	*Average*
(a)	✓		25.83	26.56	28.00	26.46	18.89	20.15	31.96	26.51	25.55
(b)		✓	24.62	25.19	30.14	22.97	19.45	20.02	32.73	23.19	24.79
(c)	✓	✓	**26.41**	**28.00**	**30.99**	**27.35**	**19.97**	**21.26**	**33.02**	**26.83**	**26.73**

Table 4 shows the quantitative results on the LLFF dataset with the COLMAP-initialized poses and the further pose refinement using the proposed method. Here, '*w/* Pose Refinement' denotes the joint optimization of camera poses and scene reconstructions by the proposed method.

As shown in row (b) compared with (a), the results show that the proposed method achieves better view synthesis quality for some scenes, such as *Fortress, Leaves, Room*. This supports that pose registration (from scratch) can often align more accurate poses for scene reconstruction, even better than the classical geometry-based approaches (Schönberger & Frahm, 2016). Additionally, comparing rows (a) and (c) in the Table, it is observed that the pose registration from the COLMAP initialization significantly outperforms the COLMAP (fixed) or the from-scratch. Informed by this analysis, since the entire pipeline is lighter than the other vanilla NeRF architectures, the proposed method is expected to be useful in real-world problems of novel-view synthesis with imperfect camera poses, even for quite accurate ones from the COLMAP.

B.3. Multi-Resolution Hash Encoding with BARF and GARF

To ensure a comprehensible and fairer comparison, we conducted additional experiments incorporating BARF and GARF within a multi-resolution hash encoding framework, referred to as BARF-NGP and GARF-NGP, respectively. These models utilized a 6-layer network architecture, with each layer having 256 neurons. BARF-NGP employed a coarse-to-fine strategy and a trainable pose parameter, while GARF-NGP replaced the activation function of BARF-NGP with a Gaussian activation function.

Table 11. Comparison of the various pose reconstruction method with Instant-NGP. Experiments are conducted on the *Hotdog* in the NeRF-Synthetic.

Model	**Evaluation Metric**		
	Rotation (°) ↓	Translation ↓	PSNR ↑
BARF-NGP	0.942	2.991	30.38
GARF-NGP	0.939	2.992	30.41
Ours	**0.234**	**1.124**	**35.41**

Table 11 shows the reconstruction performance of BARF-NGP and GARF-NGP with imperfect camera poses. As shown in the table, BARF-NGP and GARF-NGP exhibited relatively poor performance due to their inability to resolve gradient fluctuations in multi-resolution hash encodings.

Moreover, this additional experiment failed to uncover meaningful performance gains for GARF-NGP over BARF-NGP. GARF has argued that its use of a Gaussian activation function is particularly well-suited for pose reconstruction, given its strong approximation of the first derivative. However, we hypothesize that such abilities may be compromised when applied to multi-resolution hash encoding, given the prevalence of gradient fluctuation. This hypothesis is supported by our observation that the performance of GARF-NGP did not improve over that of BARF-NGP.

B.4. Proposed-Method with Another Grid-Based Approach

Although we limited the scope of this work to the Instant-NGP, our proposed method is not limited to Instant-NGP and can be applied to the other grid-based methods as well. To demonstrate this, we experimented by replacing Instant-NGP with Plenoxels (Fridovich-Keil et al., 2022). The performance of Plenoxels in jointly reconstructing the scene and pose is presented below.

Table 12. Comparison of the various pose reconstruction method with Plenoxels (Fridovich-Keil et al., 2022). Experiments are conducted on the *Hotdog* in the NeRF-Synthetic.

Model	**Evaluation Metric**		
	Rotation (°) ↓	Translation ↓	PSNR ↑
BARF-Plenoxels	0.421	2.289	33.23
GARF-Plenoxels	0.409	2.077	33.54
Ours-Plenoxels	**0.299**	**1.460**	**34.68**

The results presented in Table 12 demonstrate that the proposed method outperforms BARF and GARF even when applied to a different grid-based representation. However, the performance gains are relatively smaller compared to the gains obtained when BARF and GARF are directly applied to NGP (refer to Table 11).

We speculate that the difference in performance between Plenoxels and Instant-NGP is due to differences in their learnable encoding. Instant-NGP uses a hash function to map a table entry to a vertex, making the feature non-differentiable with respect to position. It results in the pose update relying solely on interpolation weights. On the other hand, Plenoxels (Fridovich-Keil et al., 2022) and other grid-based methods, such as DVGo (Sun et al., 2022), DIVeR (Wu et al., 2021), and Fourier-Plenoctrees (Wang et al., 2022), use a one-to-one mapping between the learnable encoding (e.g., coefficients of spherical harmonics) and a vertex, enabling differentiation over the position. Therefore, the pose reconstruction problem with Instant-NGP is more challenging, which is the primary focus of our research.

Lastly, it is noteworthy that multi-resolution hash encoding outperforms other grid-based representations, including Plenoxels, which further demonstrates the effectiveness of the proposed method for pose reconstruction.

B.5. Qualitative Results

In Figure 5, we present the qualitative results of the proposed method in the fourth column. For comparison, the ground-truth image and the rendering of two previous works, BARF (Lin et al., 2021) and GARF (Chng et al., 2022), are also presented in the first, second, and third columns, respectively. The ground-truth is shown at the beginning of each row. The qualitative results generally show that our method gives the finer-grained novel-view synthesis with the camera pose refinement.

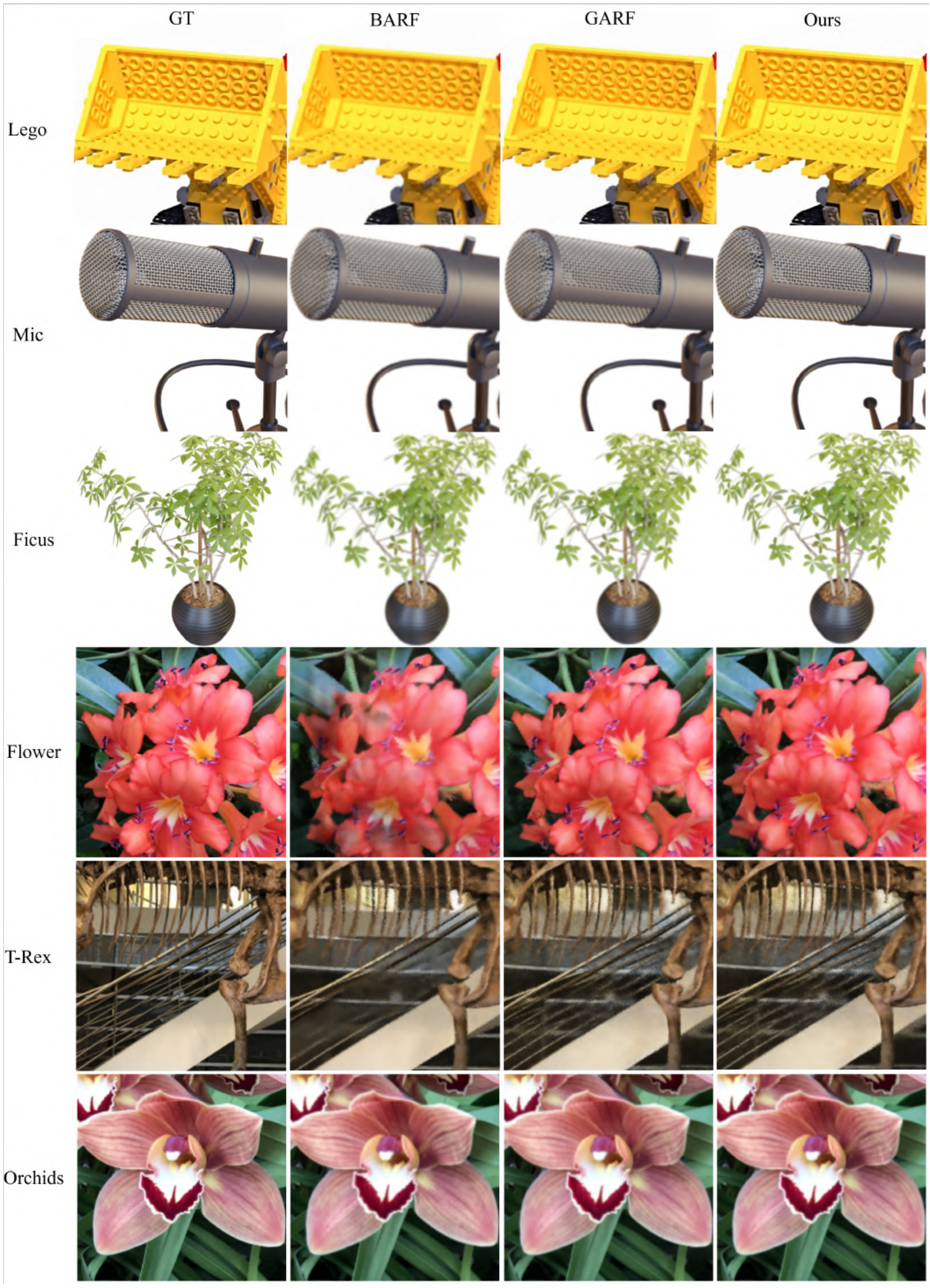

Figure 5. Qualitative results of the proposed method.