Improving Robustness for Joint Optimization of Camera Poses and Decomposed Low-Rank Tensorial Radiance Fields
Supplementary Material

Bo-Yu Cheng, Wei-Chen Chiu, Yu-Lun Liu

National Yang Ming Chiao Tung University
tomas1999.ee06@nycu.edu.tw, walon@cs.nctu.edu.tw, yulunliu@cs.nycu.edu.tw

1 Overview

This supplementary material presents additional results to complement the main manuscript. First, we provide details on the spectrum analysis of 1D signal alignment in Sec. 2. Second, we describe the connection between 1D and 3D Gaussian filtering, which is the main motivation behind our proposed method, in Sec. 3. Next, we derive the equivalence of separable component-wise convolution and naïve 3D convolution in Sec. 4. Finally, we provide the complete training process and implementation details in Sec. 5 and Sec. 6, respectively. Alongside this document, we provide additional video results and compare our results with state-of-the-art methods.

2 Complete Version of Spectrum Analysis of 1D Signal Alignment

In Eq. 5 of Sec. 3.2, we formulated the 1D signal alignment problem as:

$$\begin{aligned}\mathcal{L}_{\text{1d}}(g, q_1, q_2) &= \sum_{i\in[1,2]} \int \|\mathcal{W}_{\text{1d}}(g, q_i)(x) - f_i(x)\|^2 dx \\ &= \sum_{i\in[1,2]} \int \|g(x) - f_{GT}(x - p_i + q_i)\|^2 dx,\end{aligned} \tag{1}$$

where $\mathcal{W}$ is the translation operator on signal defined as $\mathcal{W}_{\text{1d}}(g, q_i)(x) = g(x - q_1)$, f_{GT} is the ground truth 1D signal, p_1, p_2 are predetermined translation values, and g is the signal we are trying to optimize, and q_1, q_2 are the reconstructed translation values. We optimize $\mathcal{L}_{\text{1d}}$ with gradient descent and attempt to reach one of the global minima, where $q_1 - q_2 = p_1 - p_2$ and g = $\mathcal{W}_{\text{1d}}(f_{GT}, p_1 - q_1)$

2.1 Theorem 1: Simplifying Joint Optimization to Pure Alignment on Shifted GT Signals

To simplify the complex dynamic interaction of joint optimization in Eq. 1, here we assume that g rapidly converges to temporary optima (with respect to the current translation) before the translation parameters q_1, q_2 are further refined. Note that this assumption is reasonable because voxel-based architectures (which we focus on) easily overfit to current supervision. Due to the property that the minima of squared error are achieved at average value, this allows us to replace g by the average of two translated ground truth signals. (Note that the partial optimization of g is convex and is guaranteed to converge to $g^*_{q_1,q_2}$).

$$\begin{aligned} g^*_{q_1,q_2} &= \underset{g}{\text{argmin}}\, \mathcal{L}_{\text{1d}}(g, q_1, q_2) \\ &= \underset{g}{\text{argmin}} \sum_{i\in[1,2]} \int \|g(x) - f_{GT}(x - p_i + q_i)\|^2 dx \\ &= \frac{f_{GT}(x - p_1 + q_1) + f_{GT}(x - p_2 + q_2)}{2}. \end{aligned} \tag{2}$$

Substituting Equation 2, the loss $\mathcal{L}_{1d}$ with respect to q_1 and q_2 under this assumption can be simplified as follows.

$$
\begin{aligned}
\mathcal{L}_{\text{1d}}(q_1, q_2) &= \mathcal{L}_{\text{1d}}(g^*_{q_1,q_2}, q_1, q_2) \\
&= \sum_{i\in[1,2]} \int \|g^*(x - q_i) - f_i(x)\|^2 dx \\
&= \sum_{i\in[1,2]} \int \|g^*(x)_{q_1,q_2} - f_{GT}(x - p_i + q_i)\|^2 dx \\
&= \sum_{i\in[1,2]} \int \|\frac{f_{GT}(x - p_1 + q_1) - f_{GT}(x - p_2 + q_2)}{2}\|^2 dx \\
&= \int \|f_{GT}(x - p_1 + q_1) - f_{GT}(x - p_2 + q_2)\|^2 dx \\
&= \int \|f_{GT}(x) - f_{GT}(x + u)\|^2 dx \\
&= \int \|f_{GT}(x) - \mathcal{W}(f_{GT}, -u)\| dx,
\end{aligned}
\tag{3}
$$

where $u = (p_1 - p_2) - (q_1 - q_2)$ is the translation value of the pure alignment problem between two shifted ground truth signals f_{GT} and $\mathcal{W}(f_{GT}, -u)$.

2.2 Theorem 2: Spectral Property of Gradient in 1D Signal Alignment

$$
\begin{aligned}
\text{Let } u &= (p_1 - p_2) - (q_1 - q_2) && \text{(a). replace variable} \\
\mathcal{L}_{1d} &= \int \|f_{GT}(x) - f_{GT}(x + u)\|^2 dx \\
&= \int \| \mathfrak{F}[\ f_{GT}(x) - f_{GT}(x + u)\ ] \|^2 dk && \text{(b). Parseval's theorem} \\
&= \int \| \mathfrak{F}[\ f_{GT}\ ] - e^{iku}\mathfrak{F}[\ f_{GT}\ ] \|^2 dk && \text{(c). shift property} \\
&= \int \| \mathfrak{F}[\ f_{GT}\ ] \|^2 \cdot \| 1 - e^{iku} \|^2 dk \\
\frac{d}{du}\mathcal{L}_{1d} &= \frac{d}{du} \int \| \mathfrak{F}[\ f_{GT}\ ] \|^2 \cdot \| 1 - e^{iku} \|^2 dk \\
&= \int \| \mathfrak{F}[\ f_{GT}\ ] \|^2 \cdot \frac{d}{du} \| 1 - e^{iku} \|^2 dk && \text{(d). swap operators by Leibniz integral rule} \\
&= \int \| \mathfrak{F}[\ f_{GT}\ ] \|^2 \cdot H(u, k)\ dk,
\end{aligned}
\tag{4}
$$

where $H(u, k) = 4\pi k\ sin(2\pi ku)$, $\mathfrak{F}[\ f_{GT}\ ]$ represents the Fourier transform of the spatial domain function $f_{GT}(x)$, and k is the wavenumber in the frequency domain. In the final form, the function $H(u, k)$ transfers the spectrum $\mathfrak{F}[\ f_{GT}\ ]$ into the derivative $\frac{d}{du}\mathcal{L}_{1d}$. The value of $H(u, k)$ is plotted in the left part of Fig. 1. We can see that the sign of the transfer function H is well-behaved when the magnitude of k is small. Here "well-behaving" means that the sign of the gradient is able to help u descend to 0. (i.e., positive when $u > 0$ and negative when $u < 0$). However, when the magnitude of k increases, the sign of H quickly begins to alternate with increasing magnitude. If the spectrum of f_{GT} is too wide, the sign of $\frac{d}{du}\mathcal{L}_{1d}$ will be affected by the flipping sign of $H(u, k)$, and joint optimization will get stuck in local optima.

2.3 Theorem 3: Effect of Gaussian Kernel on the 1D Signal Alignment

To deal with this flipping gradient issue of $H(u, k)$ when k departs from 0, we try to shrink the bandwidth of f_{GT} so that it won't reach too much into the flipping area of the transfer function H. Applying the 1D Gaussian filter $\mathcal{N}(x)$ on f_{GT} is a natural solution. Here we show that applying a Gaussian filter to f_{GT} is effectively the same as modulating the transfer function

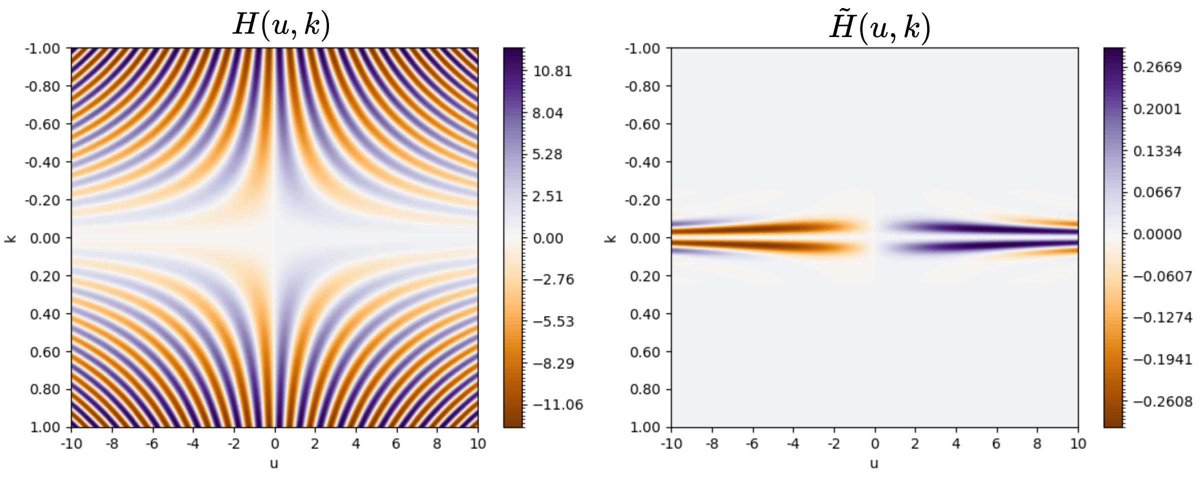

Figure 1: **Copy of Fig. 3(b)** is placed here for readability. Left: The value of $H(u, k)$ is plotted. We can see that the sign of the transfer function H is well-behaved when the magnitude of k is small. Here, "well-behaving" means that the sign of the gradient is able to help u descend to 0, i.e., positive when $u > 0$ and negative when $u < 0$. However, when k departs from 0, the sign of H quickly begins to alternate and the magnitude increases, which causes the gradient to be large and noisy. Hence, high-frequency signals with a spreading spectrum can easily get stuck in local optima. Right: Filtering the f_{GT} with Gaussian kernel is effectively the same as modulating the function $H(u, k)$ with a Gaussian mask, resulting in a more desirable function $\tilde{H}(u, k)$

.

H.

$$\begin{aligned}
\text{Let } \tilde{f}_{GT} &= \mathcal{N} *_{1\text{d}} \tilde{f}_{GT} \text{ be the filtered signal} \\
\tilde{\mathcal{L}}_{1d} &= \mathcal{L}_{1d} \text{ calculated with } \tilde{f}_{GT} \\
\frac{d}{du}\tilde{\mathcal{L}}_{1d} &= \int \| \mathfrak{F}[\, \mathcal{N} *_{1\text{d}} f_{GT} \,] \|^2 \cdot H(u,k)\, dk && \text{(a) Substitute Eq. 4} \\
&= \int \| \mathfrak{F}[\, \mathcal{N} \,] \cdot \mathfrak{F}[\, f_{GT} \,] \|^2 \cdot H(u,k)\, dk && \text{(b) convolution property} \\
&= \int \| \mathfrak{F}[\, f_{GT} \,] \|^2 \cdot \| \mathfrak{F}[\, \mathcal{N} \,] \|^2 \cdot H(u,k)\, dk && \text{(c) } \mathfrak{F}[\, \mathcal{N} \,] \text{is real} \\
&= \int \| \mathfrak{F}[\, f_{GT} \,] \|^2 \cdot \tilde{H}(u,k)\, dk,
\end{aligned} \tag{5}$$

where $\tilde{H}(u,k) = \| \mathfrak{F}[\, \mathcal{N} \,] \|^2 \cdot H(u,k)$, and $*_{1\text{d}}$ denotes the 1D convolution operator. Step (a) comes directly from Equation 4, and step (b) applies the convolution property of the Fourier transform. In step (c), we use the identity $\mathfrak{F}[\mathcal{N}(x)] = \mathfrak{F}[e^{-ax^2}] = -\sqrt{\frac{\pi}{a}} e^{-\pi^2 k^2/a}$, which says that the Fourier transform of Gaussian function is another Gaussian function; hence $\mathfrak{F}[\mathcal{N}(x)]$ is real-valued and can be separated out of the 2-norm.

In Fig 3(b)(bottom) in the main paper (same as the right half of Figure 1), we plot the modulated transfer function $\tilde{H}(u, k)$ (filter $N(x)$ is generated by Eq.11 with $\sigma = 4.0$). We can see that the misbehaved region is suppressed and the gradient descent is very likely to converge to $u = 0$ as long as the initial magnitude of u is less than 6.0. (Actually, the region in which $\frac{d}{du}\tilde{\mathcal{L}}_{1d}$ is well-behaved is *quasi-convex* and is guaranteed to converge to global optima given the suitable learning rate that prevents us from getting stuck at saddle points.)

3 Connection of Gaussian Filtering in 1D and 3D (Motivation for Proposed Techniques In Sec. 3.6)

Here we discuss the connections between the 1D analysis in Sec.3.2 and the 3D joint optimization techniques proposed in Sec. 3.6.

3.1 Connection to Smoothed 2D Supervision & 3D Gaussian Filtering

Note that in the previous section, we only considered blurring the input signals f_{GT} (which in turn affects f_1 and f_2 in Eq. 1). If we strictly map this setting into the joint optimization in the 3D case as described in Sec. 3.2.2, we should only blur 2D input training images (*Smoothed 2D Supervision* in Sec. 3.6). However, we found empirically that restricting the spectrum of both 2D training images and the 3D radiance field gives the best reconstruction quality (as shown in Tab. 4).

3.2 Motivation for Randomly Scaled Kernel & Edge Guided Loss

From the previous spectral analysis, one may have the impression that a larger kernel leads to stronger modulation, and hence always results in more robust pose registration. However, this is not always true, because the magnitude of $H(u, k)$ decreases linearly as k approaches 0. Notice that in Fig. 1(Right) the magnitude of modulated $\tilde{H}$ is weaker than that of H, which means that $\frac{d}{du}\tilde{\mathcal{L}}_{1\text{d}}$ is weaker than $\frac{d}{du}\mathcal{L}_{1\text{d}}$ and therefore is more easily influenced by noise. In the 3D case, this **weak and noisy gradient problem** caused by overly aggressive filtering corresponds to the excessive blur effect that destroys important edge signals in the training images, causing pose alignment to fail. See Fig. 2(b) for a visualization of the image blurred by an over-strength kernel, in which the thin edge information is eliminated, causing the camera pose to randomly drift.

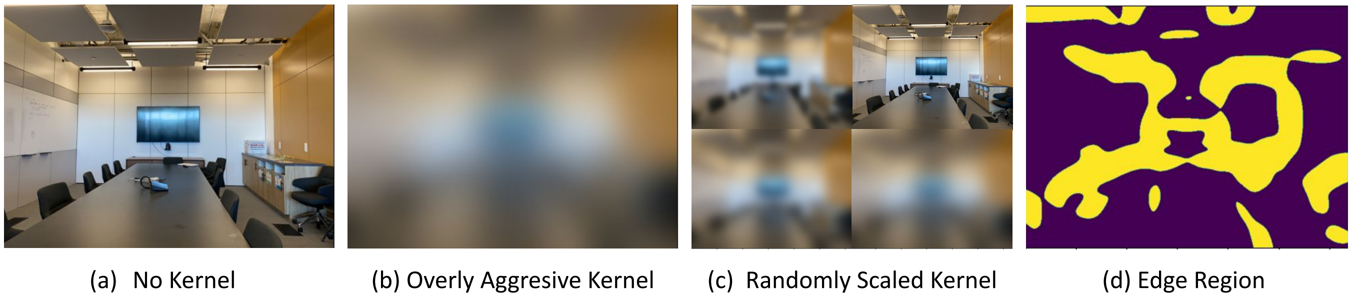

Figure 2: **Visualization of 2D Randomly Sampled Kernel and Edge Guided Loss**. (a) Input supervision without kernel. Joint optimization using unblurred images easily overfit to high-frequency noises (b) Input supervision blurred by an overly aggressive kernel. Notice that the edge information is largely destroyed by the blurring process, resulting in weak and noisy gradients, causing the poses to drift around easily. (c) Same input supervision blurred by four randomly scaled kernels. We empirically found that mixing different filtering strengths results in a more robust joint optimization. (d) Edge areas of the blurred image selected by the Sobel filter with a threshold set to 1.25x of the average value of the filtered map.

Real-world scenes are composed of edge structures of various scales; it is insufficient to use a single-size kernel on all these different scene structures (in which the same kernel may be overly aggressive in one scene, but overly gentle in another scene). Therefore, we introduce *randomly scaled kernel* in Sec. 3.6, which randomly scales the kernel by a factor uniformly sampled from $[0, 1]$. See Fig 2(c) for a visualization of the same input image filtered by a range of randomly sampled kernels. We observe that the training schedule becomes more robust when we alternate between these randomly sampled kernel scales.

Another way to mitigate the weak and noisy gradient problem is the *edge guided loss* introduced in Sec. 3.6, in which we increase the learning rate (and hence amplify the gradient signal) on pixels in the edge area. See visualization in Fig. 2 (d), where we color the edge area (which is detected using the Sobel filter on the filtered 2D images) in yellow. Edge-guided rendering loss helps joint optimization focus more on the edge areas of the training images, resulting in more robust pose optimization.

4 Theorem 4: Equivalence of Separable Component-Wise Convolution and Naive 3D Convolution

In Eq. 15 of Sec. 3.5, we used the following identity to simplify the computation of Naive 3D convolution into separable component-wise convolution.

$$\tilde{\mathcal{T}}_\sigma = \sum_{r=1}^{\mathbf{R}} \tilde{\mathbf{v}}^X_{\sigma,r} \otimes \tilde{\mathbf{M}}^{Y,Z}_{\sigma,r} + \tilde{\mathbf{v}}^Y_{\sigma,r} \otimes \tilde{\mathbf{M}}^{X,Z}_{\sigma,r} + \tilde{\mathbf{v}}^Z_{\sigma,r} \otimes \tilde{\mathbf{M}}^{X,Y}_{\sigma,r}, \tag{6}$$

where $\tilde{\mathcal{T}}_\sigma = (\mathcal{N}_{3\text{d}} *_{3\text{d}} \mathcal{T}_\sigma)$ denotes the 3D Gaussian convoluted tensor volume, $\tilde{\mathbf{v}}_{\sigma,r} = (\mathcal{N}_{1d} *_{1d} \tilde{\mathbf{v}}_{\sigma,r})$ denotes the 1D Gaussian convoluted vector component, and $\tilde{\mathbf{M}}_{\sigma,r} = (\mathcal{N}_{2\text{d}} *_{2\text{d}} \tilde{\mathbf{M}}_{\sigma,r})$ denotes the 2D Gaussian convoluted matrix component.

To prove Eq. 6, we first notice that the outer product between the lower-dimensional component $\otimes$ can be viewed as convolution $*$ in a higher-dimensional space. We can view the outer product as a convolution with full padding, where the kernel and the signal are embedded into the high-dimensional space before performing the convolution. For example, the outer product between two 1D vectors $\mathbf{u} \in \mathbb{R}^n$ and $\mathbf{v} \in \mathbb{R}^m$ is a $m \times n$ matrix. We can obtain the same result by performing 2D convolution on 2D discrete embedded functions $\mathbf{u}^X, \mathbf{v}^Y : \mathbb{Z}^2 \to \mathbb{R}$.

$$
\begin{aligned}
\mathbf{u}^X[x,y] &= \begin{cases} \mathbf{u}[x]\delta[y] & \text{if } 0 \le x < n \\ 0 & \text{otherwise} \end{cases} \\
\mathbf{v}^X[x,y] &= \begin{cases} \mathbf{v}[y]\delta[x] & \text{if } 0 \le y < m \\ 0 & \text{otherwise} \end{cases} \\
(\mathbf{u}^X *_{2\text{d}} \mathbf{v}^Y)[x,y] &= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \mathbf{u}^X[i,j] \cdot \mathbf{v}^Y[x-i, y-j] && \text{(a) expand definition} \\
&= \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} \mathbf{u}[i]\ \delta[j] \cdot \mathbf{v}[y-j]\ \delta[x-i] && \text{(b) only add turns where } j = 0 \text{ and } x - i = 0 \\
&= \mathbf{u}[x]\ \delta[0] \cdot \mathbf{v}[y-0]\ \delta[0] && \text{(c) substitute values} \\
&= \mathbf{u}[x] \cdot \mathbf{v}[y] \\
&= (\mathbf{u} \otimes \mathbf{v})[x,y], && \text{(e) definition of } \otimes
\end{aligned}
\tag{7}
$$

where in (c) we assume $\mathbf{u}[x]$ and $\mathbf{v}[y]$ returns 0 when index is out of range of the vector. Now we have proved that the result of the outer product $\otimes$ between 1D vectors is identical to the convolution $*_{2\text{d}}$ of two embedded vectors in the 2D space.

Similar properties can be obtained for vector-matrix products:

$$
(\mathbf{v}^X *_{3\text{d}} \mathbf{M}^{Y,Z})[x,y,z] = (\mathbf{v} \otimes \mathbf{M})[x,y,z], \quad \text{(a) proof is similar to Eq. 7,} \tag{8}
$$

where $\mathbf{v}^X$ is the embedded version of the 1D vector $\mathbf{v}$ along the X axis in the 3D space, and $\mathbf{M}^{Y,Z}$ is the embedded version of the 2D matrix $\mathbf{M}$ along the Y, Z axes in the 3D space.

Now with Eq. 7 and Eq. 8 we can replace $\otimes$ in 3D Gaussian definition and Eq. 6 by convolution $*_{3\text{d}}$, after which we apply the commutative and associative property of the convolution operator $*_{3\text{d}}$ to distribute the separable Gaussians into each tensor component.

$$
\begin{aligned}
\tilde{\mathcal{T}}_\sigma &= \mathcal{N}_{3\text{d}} *_{3\text{d}} \mathcal{T}_\sigma \\
&= (\mathcal{N}_{1\text{d}}^X \otimes \mathcal{N}_{1\text{d}}^Y \otimes \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} (\sum_{r=1}^{\mathbf{R}} \mathbf{v}_{\sigma,r}^X \otimes \mathbf{M}_{\sigma,r}^{Y,Z} + \mathbf{v}_{\sigma,r}^Y \otimes \mathbf{M}_{\sigma,r}^{X,Z} + \mathbf{v}_{\sigma,r}^Z \otimes \mathbf{M}_{\sigma,r}^{X,Y}) && \text{(a) expand definition} \\
&= (\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathcal{N}_{1\text{d}}^Y *_{3\text{d}} \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} (\sum_{r=1}^{\mathbf{R}} \mathbf{v}_{\sigma,r}^X *_{3\text{d}} \mathbf{M}_{\sigma,r}^{Y,Z} + \mathbf{v}_{\sigma,r}^Y *_{3\text{d}} \mathbf{M}_{\sigma,r}^{X,Z} + \mathbf{v}_{\sigma,r}^Z *_{3\text{d}} \mathbf{M}_{\sigma,r}^{X,Y}) && \text{(b) apply Eq. 7 and Eq. 8} \\
&= \sum_{r=1}^{\mathbf{R}} (\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathcal{N}_{1\text{d}}^Y *_{3\text{d}} \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} (\mathbf{v}_{\sigma,r}^X *_{3\text{d}} \mathbf{M}_{\sigma,r}^{Y,Z}) + \\
&\qquad (\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathcal{N}_{1\text{d}}^Y *_{3\text{d}} \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} (\mathbf{v}_{\sigma,r}^Y *_{3\text{d}} \mathbf{M}_{\sigma,r}^{X,Z}) + \\
&\qquad (\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathcal{N}_{1\text{d}}^Y *_{3\text{d}} \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} (\mathbf{v}_{\sigma,r}^Z *_{3\text{d}} \mathbf{M}_{\sigma,r}^{Z,Y}) && \text{(c) linearity of } *_{3\text{d}} \\
&= \sum_{r=1}^{\mathbf{R}} (\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathbf{v}_{\sigma,r}^X) *_{3\text{d}} ((\mathcal{N}_{1\text{d}}^Y *_{3\text{d}} \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} \mathbf{M}_{\sigma,r}^{Y,Z}) + \\
&\qquad (\mathcal{N}_{1\text{d}}^Y *_{3\text{d}} \mathbf{v}_{\sigma,r}^Y) *_{3\text{d}} ((\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathcal{N}_{1\text{d}}^Z) *_{3\text{d}} \mathbf{M}_{\sigma,r}^{X,Z}) + \\
&\qquad (\mathcal{N}_{1\text{d}}^Z *_{3\text{d}} \mathbf{v}_{\sigma,r}^Z) *_{3\text{d}} ((\mathcal{N}_{1\text{d}}^X *_{3\text{d}} \mathcal{N}_{1\text{d}}^Y) *_{3\text{d}} \mathbf{M}_{\sigma,r}^{X,Y}) && \text{(c) commutivity , associativity, of } *_{3\text{d}} \\
&= \sum_{r=1}^{\mathbf{R}} \tilde{\mathbf{v}}_{\sigma,r}^X *_{3\text{d}} \tilde{\mathbf{M}}_{\sigma,r}^{Y,Z} + \tilde{\mathbf{v}}_{\sigma,r}^Y *_{3\text{d}} \tilde{\mathbf{M}}_{\sigma,r}^{X,Z} + \tilde{\mathbf{v}}_{\sigma,r}^Z *_{3\text{d}} \tilde{\mathbf{M}}_{\sigma,r}^{X,Y} && \text{(d) definition rewrite} \\
&= \sum_{r=1}^{\mathbf{R}} \tilde{\mathbf{v}}_{\sigma,r}^X \otimes \tilde{\mathbf{M}}_{\sigma,r}^{Y,Z} + \tilde{\mathbf{v}}_{\sigma,r}^Y \otimes \tilde{\mathbf{M}}_{\sigma,r}^{X,Z} + \tilde{\mathbf{v}}_{\sigma,r}^Z \otimes \tilde{\mathbf{M}}_{\sigma,r}^{X,Y}. && \text{(e) apply Eq. 8.}
\end{aligned}
\tag{9}
$$

Now the proof is complete, and we have verified that **the 3D convoluted tensor can be expressed as the composition of**

individually convoluted components. With a similar process, we can prove that the 3D feature tensor also has the property:

$$\begin{aligned}\tilde{\mathcal{T}}_c &= \mathcal{N}_{\text{3d}} *_{\text{3d}} \mathcal{T}_c \\ &= \sum_{r=1}^{\mathbf{R}} \tilde{\mathbf{v}}_{c,r}^X \otimes \tilde{\mathbf{M}}_{c,r}^{Y,Z} \otimes \mathbf{b}^X + \tilde{\mathbf{v}}_{c,r}^Y \otimes \tilde{\mathbf{M}}_{c,r}^{X,Z} \otimes \mathbf{b}^Y + \tilde{\mathbf{v}}_{c,r}^Z \otimes \tilde{\mathbf{M}}_{c,r}^{X,Y} \otimes \mathbf{b}^Z.\end{aligned} \tag{10}$$

5 Complete Training Process

The training loss of our method in joint optimization of the 3D running field can be summarized as follows

$$\begin{aligned}\tilde{\mathcal{L}}_{\text{joint}}(F_\sigma, F_c, \mathbf{P}, \mathcal{N}_\sigma, \mathcal{N}_c, \mathcal{N}_I) &= \sum_{i=1}^{L} \sum_{u\in\mathbf{U}} \mathrm{E}_i u \cdot \|\mathbf{V}(\tilde{F}_\sigma, \tilde{F}_c, \mathcal{W}_{\text{3d}}(P_i, s(\vec{0}, \vec{d_u}))) - \tilde{I}_{iu}\| \\ \tilde{F}_\sigma(\mathbf{x}) &= \tilde{\mathcal{T}}_\sigma(\mathbf{x}), \text{ where } \tilde{\mathcal{T}}_\sigma \text{ is component-wise convoluted with } \mathcal{N}_\sigma \\ \tilde{F}_c(\mathbf{x}, \vec{d}) &= s(\tilde{\mathcal{T}}_c(\mathbf{x}), \vec{d}), \text{ where } \tilde{\mathcal{T}}_c \text{ is component-wise convoluted with } \mathcal{N}_c \\ \tilde{I} &= \{\mathcal{N}_I^X *_{\text{2d}} \mathcal{N}_I^Y *_{\text{2d}} I_i\}_{i=1,2,\cdots,L},\end{aligned} \tag{11}$$

where E_{iu} is the edge-guided rendering weight of pixel u on image i (In the first few iterations, $E_{iu} = 1.5$ for pixels on the edge regions, after that, $E_{iu} = 1.0$ for all training pixels.), $\mathcal{N}_c, \mathcal{N}_\sigma$ are 1D Gaussian kernel for convolving $\mathcal{T}_c, \mathcal{T}_\sigma$ respectively, $\mathcal{N}_I$ is the 1D Gaussian kernel that is used for smoothing the supervision images I.

The complete training loss can finally be expressed as

$$\mathcal{L}_{\text{3d}} = w_1 \cdot \mathcal{L}_{\text{joint}} + w_2 \cdot \mathcal{L}_{\text{L1}} + w_3 \cdot \mathcal{L}_{\text{TV}}, \tag{12}$$

where $\mathcal{L}_{\text{joint}}$ is described in Eq. 11, $\mathcal{L}_{\text{L1}}, \mathcal{L}_{\text{TV}}$ are the L1 loss and TV loss on tensor components $\mathbf{v}_{\sigma,r}, \mathbf{M}_{c,r}, \mathbf{M}_{\sigma,r}, \mathbf{v}_{c,r}$ respectively. w_1, w_2, w_3 are loss weights.

Now, the total training process can be described as Algorithm 1, where the kernel width is first sampled from a predefined kernel schedule, then randomly scaled by a factor sampled uniformly from $[0, 1]$, then Gaussian kernels are constructed and used to calculate our proposed rendering loss $\mathcal{L}_{\text{joint}}$ in Eq. 11. Finally, the total loss $\mathcal{L}_{\text{3d}}$ in Eq. 12, based on which the camera poses and the radiance field are jointly updated.

Algorithm 1: Conceptual Training Process for Our Proposed 3D Joint Optimization Training

```
$\mathcal{T}_\sigma, \mathcal{T}_c \leftarrow$ Initialize Voxel Grid
$\mathbf{P} \leftarrow$ Initialize Camera Poses
**for** $s$ = 1 **to** train_iters **do**
    $i \leftarrow$ 2D_kernel_sched($s$)
    $\sigma$, $c \leftarrow$ 3D_kernel_sched($s$)
    $u_\sigma \leftarrow$ randomly sample density kernel scale
    $u_I \leftarrow$ randomly sample 2D kernel scale
    $\mathcal{N}_\sigma \leftarrow$ sample discrete gaussian kernel with variance $(\sigma \cdot u_\sigma)^2$
    $\mathcal{N}_c \leftarrow$ sample discrete gaussian kernel with variance $c^2$
    $\mathcal{N}_I \leftarrow$ sample discrete gaussian kernel with variance $(i \cdot u_I)^2$
    $\tilde{\mathcal{L}}_{\text{joint}} \leftarrow \tilde{\mathcal{L}}_{\text{joint}}(\mathcal{T}_\sigma, \mathcal{T}_c, \mathbf{P}, \mathcal{N}_\sigma, \mathcal{N}_c, \mathcal{N}_I)$ (with randomly selected pixels in all training views)
    $\mathcal{L}_{\text{L1}}$ = $\mathcal{L}_{\text{L1}}(\mathbf{v}_{\sigma,r}, \mathbf{M}_{c,r}, \mathbf{M}_{\sigma,r}, \mathbf{v}_{c,r})$
    $\mathcal{L}_{\text{TV}}$ = $\mathcal{L}_{\text{TV}}(\mathbf{v}_{\sigma,r}, \mathbf{M}_{c,r}, \mathbf{M}_{\sigma,r}, \mathbf{v}_{c,r})$
    $\mathcal{L}_{\text{3d}} = w_1 \cdot \mathcal{L}_{\text{joint}} + w_2 \cdot \mathcal{L}_{\text{L1}} + w_3 \cdot \mathcal{L}_{\text{TV}}$
    back propagation
    update $\mathcal{T}_\sigma, \mathcal{T}_c, \mathbf{P}$
**end for**
```

6 Implementation Details

We evaluate our proposed method against three previous works BARF (Lin et al. 2021), GARF (Chng et al. 2022), and HASH (Heo et al. 2023). Since the implementations of GARF and HASH are unavailable, we directly use the results reported in their paper for comparison. For the planar image alignment task, we compare our result with (Lin et al. 2021) under the same settings. To reconstruct the radiance field, we follow (Lin et al. 2021; Chng et al. 2022; Heo et al. 2023) to evaluate and compare our method on **NeRF-Synthetic** dataset and **LLFF** dataset.

Planar Image Alignment is performed under settings identical to those of BaRF (Lin et al. 2021), where a sample image is chosen from ImageNet (Deng et al. 2009) , from which $L_{2\text{d}} = 5$ patches are cropped with different homography transform parameters $P_0, P_1, \cdots, P_5 \in \mathbb{R}^8$. We jointly reconstruct the homography parameters and the 2D image represented with the decomposed 2D tensor. The warp parameters are parameterized in $sl(3)$ and initialized to $\vec{0}$ before training.

NeRF-Synthetic dataset is proposed by (Yen-Chen et al. 2020), and consists of 8 object-centric scenes. For each scene, the training data consist of 100 images along with the camera extrinsic and intrinsic. Following (Lin et al. 2021), we simulate camera noise with Gaussian noise $N(0, 0.15I)$ on $\mathfrak{se}(3)$ pose embedding. The noises are composed of the ground truth extrinsic parameter, and our joint optimization process aims to cancel the noise and restore accurate camera poses. We follow BaRF (Lin et al. 2021) to resize training and testing images to 400×400.

LLFF dataset is proposed by (Mildenhall et al. 2019), and consists of 8 forward-facing scenes captured with a handheld camera. The ground truth camera poses in the datasets are estimated by COLMAP (Schönberger et al. 2016; Schönberger and Frahm 2016). Following BaRF (Lin et al. 2021), we optimize camera poses from scratch with identity initialization, and images are resized to 480×640 before being used.

6.1 Implementation Details for Planar Image Alignment

We use 500×500 2D decomposed tensor with 100 components. The 2D tenor and homography warp parameters are jointly optimized with *Adam* optimizer (Kingma and Ba 2014) with learning rates 0.001 and 0.01 respectively. The total training iteration is set to 15000, Gaussian kernel schedule shrinks exponentially with a pise-wise linear curve that starts with 128 and reaches 0 at 6000 iteration.

6.2 Implementation Details for NeRF (3D) Synthetic Object

We follow the implementation of (Chen et al. 2022) and sample 2048 rays per iteration across all training images with sample density equal to 2X current tensor grid resolution. We decrease the hidden width of the decoder *MLP* to 32 and postpone the input of the viewing direction to the last layer. The pose parameters, the tensor volume, and the MLP decoder are jointly optimized with *Adam* optimizer with learning rates 0.001, 0.01, and 0.0005, respectively, and the learning rates are exponentially degraded. *Smooth 2D supervision* in Section 3.6 is used in synthetic scenes. The total training iteration is set to 40000. The 2D and 3D Gaussian kernel schedule shrinks exponentially with a pice-wise linear curve, which starts with 0.3 (in 3D coordinate) and 0.025(in 2D coordinate) and degrades to effectively 0 at 10000 iteration. The 3D tensor grid is scaled from 64^3 to 300^3 with 5 up-sample steps, and kernel parameters are adjusted to adapt to the current tensor grid resolution. We postpone the alpha mask update in (Chen et al. 2022) to allow blurry scenes in the early stage of training. The total training time is 80 minutes on a single RTX3090 GPU.

6.3 Implementation Details for NeRF (3D) Real World Scenes

Most settings in Real World Scenes are the same as the Synthetic Scenes. We describe the difference here, where we follow (Yen-Chen et al. 2020) and (Chen et al. 2022) to use the Normalized Device Coordinate (NDC) in joint optimization of the forward-facing scenes. We found that the absolute coordinates (before Procruste analysis to align with GT) of reconstructed poses are often more densely gathered than GT poses, causing the absolute size of the reconstructed scene to be smaller and clipped by the near plane in NDC coordinate; we solve this issue by moving the sampling near the plane of NDC from 1.0 to -1. Most other settings are the same as the synthetic setting. The total training iteration is set to 50000. The 3D tensor grid is scaled from 128^3 to 800^3 with 5 up-sample steps. The Gaussian kernel schedule is the same as the synthetic NeRF setting, except that we start with 0.4 in 3D coordinates for the 3D kernel and 0.07 in 2D coordinates for the 2D kernel. To accelerate the training process in increase the stability, we increase the number of rays per iteration to 20480 before 6000 iterations and restore to 4096 rays per iteration afterward. The learning rate of the pose parameters warms up for 500 iterations, and poses are reset to identity at 2500 iterations. *Randomly scaled Kernel Parameters* and *Edge-Guided Loss* described in Section 3.6 are used in real-world scenes. Note that to prevent input scale instability of the *MLP* decoder, we use randomly scaled Gaussian kernel only for density volume, the Gaussian kernel for the color volume is not randomized. The total training time is about 3 hours on a single RTX3090 GPU.

6.4 Evaluation Criteria

Following previous works (Lin et al. 2021; Chng et al. 2022; Heo et al. 2023), we measure our results in two aspects: pose error for registration and view-synthesis quality for the scene representation. Since the reconstructed poses and scenes are variable to the ground truth up to a similarity transform, we perform the Procrustes analysis to align the training pose parameter and the GT poses before calculating the rotation error and translation error. To report the view-synthesis quality independent of tiny registration noise, before calculating PSNR, SSIM, and LPIPS on the trained scene w.r.t testing image, we perform test-time optimization on each testing pose to prevent tiny pose error from contaminating the view-synthesis quality. In the 2D case, warp error (distance between homography parameter and the ground truth warp) and PSNR are reported.

References

Chen, A.; Xu, Z.; Geiger, A.; Yu, J.; and Su, H. 2022. Tensorf: Tensorial radiance fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Chng, S.-F.; Ramasinghe, S.; Sherrah, J.; and Lucey, S. 2022. Gaussian activated neural radiance fields for high fidelity reconstruction and pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heo, H.; Kim, T.; Lee, J.; Lee, J.; Kim, S.; Kim, H. J.; and Kim, J.-H. 2023. Robust Camera Pose Refinement for Multi-Resolution Hash Encoding. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lin, C.-H.; Ma, W.-C.; Torralba, A.; and Lucey, S. 2021. BARF: Bundle-Adjusting Neural Radiance Fields. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*.

Mildenhall, B.; Srinivasan, P. P.; Ortiz-Cayon, R.; Kalantari, N. K.; Ramamoorthi, R.; Ng, R.; and Kar, A. 2019. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*.

Schönberger, J. L.; and Frahm, J.-M. 2016. Structure-from-Motion Revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Schönberger, J. L.; Zheng, E.; Pollefeys, M.; and Frahm, J.-M. 2016. Pixelwise View Selection for Unstructured Multi-View Stereo. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Yen-Chen, L.; Florence, P.; Barron, J. T.; Rodriguez, A.; Isola, P.; and Lin, T.-Y. 2020. INeRF: Inverting Neural Radiance Fields for Pose Estimation. arXiv e-prints, page. *arXiv preprint arXiv:2012.05877*.