NeRFuser: Large-Scale Scene Representation by NeRF Fusion

Jiading Fang*, 1, Shengjie Lin*, 1, Igor Vasiljevic 2, Vitor Guizilini 2, Rares Ambrus 2, Adrien Gaidon 2, Gregory Shakhnarovich 1, Matthew R. Walter 1

Abstract

A practical benefit of implicit visual representations like Neural Radiance Fields (NeRFs) is their memory efficiency: large scenes can be efficiently stored and shared as small neural nets instead of collections of images. However, operating on these implicit visual data structures requires extending classical image-based vision techniques (e.g., registration, blending) from image sets to neural fields. Towards this goal, we propose NeRFuser, a novel architecture for NeRF registration and blending that assumes only access to pre-generated NeRFs, and not the potentially large sets of images used to generate them. We propose registration from re-rendering, a technique to infer the transformation between NeRFs based on images synthesized from individual NeRFs. For blending, we propose sample-based inverse distance weighting to blend visual information at the ray-sample level. We evaluate NeRFuser on public benchmarks and a self-collected object-centric indoor dataset, showing the robustness of our method, including to views that are challenging to render from the individual source NeRFs.

Code available at https://github.com/ripl/nerfuser

*Equal contribution.

1Toyota Technological Institute at Chicago, Chicago, IL.
2Toyota Research Institute, Los Altos, CA.

Figure 1: NeRFuser. Starting from separately constructed input NeRFs, NeRF A and NeRF B, we render images at novel viewpoints, including those not well-covered by either input NeRF. Our proposed registration through re-rendering followed by a novel blending procedure leads to a higher quality result (lower right) than renders from the original NeRFs (A or B).

1. Introduction

The transmission of visual data relies on efficient video and image encoding and decoding, technologies with decades of development behind them. Most computer vision methods and tools are specialized to this type of data—methods that align and blend images are ubiquitous [32] and fundamentally designed to work on explicit 2D representations of images.

However, a new learning-based representation for visual data has emerged in recent years: neural fields [39]. Pioneered by neural radiance fields (NeRFs) [16], these implicit representations allow for efficient visual compression and impressive view synthesis, generating a potentially infinite set of possible views from a fixed set of training images. Despite the promise of this representation as a storage and communication format, there is a lack of tools that treat NeRFs as data, much like common image processing tools treat images.

Towards expanding the utility of NeRFs as a data representation, we propose NeRFuser (Fig. 1), a NeRF fusion framework for the registration and blending of pre-trained NeRFs. Treating input NeRFs as black boxes (i.e., raw data), without access to the images that generate them, our method can register NeRFs (in both pose and scale) and render images from blended NeRFs. Removing the need for source images also greatly reduces memory consumption.
A typical scene may be captured by 100 images, each about 1 MB in size. In contrast, NeRF, which acts as a compression of the individual images, provides an implicit representation of the scene that takes up approximately 5 MB, a $20 \times$ reduction from the set of original images. Directly transferring this implicit representation makes it possible to build real-time 3D capturing applications (e.g., NeRF streaming).

NeRFuser fuses NeRFs in two steps: registration and blending. For the first step, we propose registration from re-rendering. It takes advantage of the ability of modern NeRFs to synthesize high-quality views, which enables us to make use of a 2D image matching pipeline for the 3D NeRF registration task. For the second step, inspired by BlockNeRF [33], we propose a fine-grained sample-based blending method, including a novel compatible weighting method. In summary, we propose (i) a novel registration from re-rendering method, that aligns uncalibrated implicit representations and solves for relative scale and pose; and (ii) a new blending method to composite predictions from multiple NeRFs, resulting in blended images that are better than images rendered by any individual NeRF.

2. Related Work

2.1. Neural Radiance Fields

A Neural Radiance Field (NeRF) [33] is a parametric representation of a 3D scene. It optimizes a neural network composed of MLPs to encode the scene as density and radiance fields, which can be used to synthesize novel views through volumetric rendering. Since its introduction, many follow-up works [2, 17, 3, 41, 31, 34] have improved over the original implementation.

One line of improvement involves the reconstruction of large-scale NeRFs [38, 43, 33, 30, 46, 45, 35, 21, 33]. However, most of these works focus on reconstructing the entire scene with a single model. While progressive training [38, 30] and carefully designed data structures [43, 46, 45] have helped to expand the expressivity of a single model, other works [21, 33] have shown that a collection of many small models can perform better, while maintaining the same number of parameters. Our method provides a novel way to reason over many small models, combining them to improve performance.

2.2. NeRF Registration

NeRFs are optimized from posed images, with poses usually obtained using a structure-from-motion (SfM) method [28, 29, 26, 27, 8, 22, 9]. Because these methods are scale-agnostic, the resulting coordinate system will have an arbitrary scale specific to each NeRF. Jointly using multiple NeRFs requires NeRF Registration, i.e., solving for the relative transformation between their coordinate systems.

Note that the setting is different from “NeRF Inversion” [40, 14] that estimates the 6-DoF camera pose relative to the pre-trained NeRF given an image, a technique that has been used for NeRF-based visual navigation and localization [1, 15, 6]. However, these tasks can potentially be handled by NeRFuser if formulated as NeRF-to-NeRF pose estimation. Also relevant are works that jointly optimize NeRF representations along with the poses and intrinsics [13, 37, 12]. However, NeRFuser only uses SfM on re-rendered images, and does not modify the pre-trained NeRFs themselves.

While there is a large body of work on registration for explicit representations (e.g., point-clouds) [20], there are few works on NeRF registration. To the best of our knowledge, there are two works related to ours: nerf2nerf [10] and Zero-NeRF [19]. Both approaches perform registration in a purely geometric way—extracting surfaces from the NeRF representations, and thus do not take full advantage of the rich encoded radiance information and its rendering capability. Further, nerf2nerf is only capable of local registration under a known scale, and even then requires a reasonable initialization in the form of human annotations. On the other hand, our method performs scaled global registration and does not require any human annotations.

2.3. NeRF Blending

Image blending is a highly researched topic in computational photography [5, 4], but few works have discussed blending in terms of NeRFs. Relevant is Block-NeRF [33], which blends NeRFs in both image- and pixel-wise manner. It introduces two ways to measure the blending weights of NeRFs: inverse distance weighting (IDW) and visibility prediction. IDW computes the contribution of each NeRF according to:

$$w_i \propto d_i^{-\gamma},$$

where $d_i$ is some notion of distance between the camera and elements of NeRF $i$, $\gamma \in \mathbb{R}$ is a positive hyper-parameter that modulates the blending rate.

Building on this work, we propose a blending approach that operates on ray samples, as well as a new IDW method for sample-wise blending. NeRFuser provides a principally more refined way of blending, which we show leads to sharper images.

3. Methodology

In this section we will first describe our NeRF registration method registration from re-rendering and then our blending technique IDW-Sample.

3.1. Background

In this section we include background information on volumetric rendering and inverse distance weighing
3.1.1 Volumetric Rendering

Given a 3D point $p$ and a viewing direction $d$, NeRF $R$ predicts the scene density $\sigma$ at that point and its color $c$ when viewed along direction $d$

$$\left[\sigma(p), c(p, d)\right] = R(p, d).$$ (2)

To render novel views, consider the image pixel corresponding to camera ray $r = (o, d)$, where $o$ is the camera’s optical center and $d$ is the ray direction.

In practice, non-overlapping samples are proposed along the ray at locations with significant predicted density. Assuming $K$ intervals of length $\delta_k$ sampled along ray $r$ at a distance of $t_k$ from $o$, a NeRF predicts the density and color for each sample as

$$\sigma_k = \sigma(o + t_k d),$$ (3a)

$$c_k = c(o + t_k d, d).$$ (3b)

The accumulated color is

$$C(r) = \sum_{k=1}^{n} T_k \alpha_k c_k,$$ (4)

where $\alpha_k = 1 - \exp(-\sigma_k \delta_k)$ is the probability that light is blocked in this sampled interval of length $\delta_k$ along the ray at location $t_k$. The probability of light reaching this interval (i.e., not being blocked along the way) is then

$$T_k = \prod_{l=1}^{k-1} (1 - \alpha_l) = \exp\left(-\sum_{l=1}^{k-1} \sigma_l \delta_l\right).$$ (5)

Additionally, termination probability is defined as

$$p_k = T_k \alpha_k,$$ (6)

which is the probability of light traveling along $r$ and getting blocked at sample $k$. Equation 4 thus becomes

$$C(r) = \sum_{k=1}^{n} p_k c_k.$$ (7)

3.1.2 NeRF Blending with IDW

When using multiple NeRFs to render an image, the contribution of each NeRF can be determined using inverse distance weighting (IDW)

$$w_i \propto d_i^{-\gamma},$$ (8)

where $d_i$ is the Euclidean distance of the blending element (e.g., ray sample in IDW-Sample) to NeRF $i$’s origin, $\gamma \in \mathbb{R}^+$ is a hyper-parameter that modulates the blending rate. Block-NeRF [33] proposes two variants of IDW, namely IDW-2D and IDW-3D, which we discuss below.

**IDW-2D** blends images using an image-wise weighting

$$I = \sum_i w_i I_i,$$ (9)

where we use the distance between the query camera center and NeRF $i$’s origin as $d_i$ to compute $w_i$. While IDW-2D works well when the query camera is much closer to one of the NeRFs than the others, it suffers when the query camera is roughly of the same distance from all source NeRFs. In
NeRFs is straightforward. When using two NeRFs, a similarity transform is required that transforms a 3D point \( p_i \) of pixel \( i \) determined using the expected depth predicted by NeRF \( i \),
\[
d_i^{(j)} = \| x_i - p_i^{(j)} \|_2.
\]
Each pixel \( j \) is then rendered by substituting \( d_i^{(j)} \) into Equation 8 as
\[
I^{(j)} = \sum_i w_i^{(j)} I_i^{(j)}.
\]
The major problem with IDW-3D is that, to accurately obtain the expected point of ray termination \( p_i^{(j)} \), it requires NeRF \( i \) to faithfully predict the depth for pixel \( j \). This is not always fulfilled since (i) NeRFs are not known to accurately reconstruct the scene geometry; and moreover, (ii) source NeRFs will be focusing on different portions of the scene by design, leading to invalid blending weights. Empirically, IDW-3D usually performs the worst among all blending methods.

### 3.2. Registration from Re-Rendering

The first step of our pipeline is to estimate the relative transformations between two or more input NeRFs.

We assume that each NeRF is trained on a separate set of images, capturing different, yet overlapping, portions of the same scene (i.e., each input NeRF has at least one neighbor). We do not assume a specific type of training data, e.g., the poses used to train the NeRFs may have come from KinectFusion [18] and have metric scale, or they may be the result of a SfM pipeline (e.g., COLMAP [29]) and thus have arbitrary scale.

As a result, each NeRF may have a unique coordinate system that is inconsistent with the others, in terms of translation and rotation as well as scale. Without loss of generality, the following discussion focuses on the registration of two NeRFs, \( A \) and \( B \), as the extension to more than two NeRFs is straightforward.

Our goal is to find the transformation \( T_{BA} \in \text{SIM}(3) \) that transforms a 3D point \( p_B \) in NeRF \( B \) to its corresponding point \( p_A \) in NeRF \( A \) as \( p_A = T_{BA} p_B \). Note that \( T_{BA} = \begin{bmatrix} R_{BA} & t_{BA} \\ 0 & 1 \end{bmatrix} \) \( S_{BA} \) can be decomposed into a rotation \( R_{BA} \), translation \( t_{BA} \), and uniform scaling of factor \( s_{BA} \), where \( S_{BA} = \text{diag}(s_{BA}, s_{BA}, s_{BA}, 1) \).

First, we assume that the NeRFs are produced with sufficient training views, so that they can generate high quality novel views. We then sample a set of poses (e.g., uniformly on the upper hemisphere), which we use as local poses to query both NeRFs to get re-renderings.

We re-purpose off-the-shelf structure-from-motion methods (SuperPoint as the feature [8] and SuperGlue as the matcher [27]) on the union of re-rendered images from the two NeRFs in order to recover their poses in the same coordinate system, which we then use for registration, as discussed next.

**Procedure and Notation** Given the trained model of NeRF \( A \), we query it with sampled camera poses \( \{G_{BA}^i\}_i \) to synthesize images \( \{I_A\}_i \). \( G_{BA}^i \in \text{SE}(3) \) is specified as a pose matrix that transforms a point from the coordinate system of camera \( A_i \) to that of NeRF \( A \). \( I_A \) is the image synthesized from NeRF \( A \) using the query pose \( G_{BA}^i \). Likewise, we query NeRF \( B \) with \( \{G_{BA}^B_i\}_i \) to synthesize images \( \{I_B\}_i \). We then feed images \( \{I_A\}_i \cup \{I_B\}_i \) as input to SfM, and obtain poses \( \{G_{BA}^C_i\}_i \) and \( \{G_{BA}^C_i\}_i \) as output. \( G_{BA}^C \in \text{SE}(3) \) is the recovered pose of image \( I_A \) from SfM. It is specified as a pose matrix that transforms a point from the coordinate frame of camera \( A_i \) to \( C \), where \( C \) is the coordinate system determined by this SfM execution. Note that an \( \text{SE}(3) \) pose matrix does not involve scale, so the induced camera-to-world transformation always assumes a specific camera instance that shares the same scale as the world.

**Recovering Scale** Let \( S_{AC} = \text{diag}(s_{AC}, s_{AC}, s_{AC}, 1) \) be the scale matrix from NeRF \( A \) to \( C \), meaning that one unit length in NeRF \( A \) equals \( s_A \) units in \( C \). Considering \( G_{ij} \in \text{SE}(3) \) as the pose of camera \( A_i \) relative to camera \( A_j \) when specified in \( C \)'s scale, we have
\[
G_{ij} = G_{Aj}^{-1} G_{Ai} \quad \text{using } C \text{ as bridge}
\]
\[
= S_{AC} G_{Ai}^{-1} s_{AC} G_{Ai}^{-1} \quad \text{using } A \text{ as bridge}
\]

If we further dissect \( G_{Ai}^{C} \) as \( \begin{bmatrix} R_{Ai}^C & t_{Ai}^C \\ 0 & 1 \end{bmatrix} \) and repeat this for \( G_{Aj}^C \), \( G_{Ai}^A \), \( G_{Aj}^A \), Equation 12 becomes
\[
\begin{bmatrix}
R_{Ai}^C \quad R_{Ai}^C \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
R_{Aj}^A \quad t_{Aj}^A \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
t_{Aj}^C - t_{Ai}^C \\
1
\end{bmatrix}
= \begin{bmatrix}
1 & 0 & s_{AC} R_{Aj}^A (t_{Aj}^A - t_{Ai}^A) \\
0 & 1
\end{bmatrix}
\]

Note that all components involved in Equation 13 are either determined when sampling camera poses or given by SfM with the exception of \( s_{AC} \), which is what we want to recover. Specifically, equating the L2 norm of the translation part from both sides of Equation 13 gives
\[
s_{AC} = \frac{\| t_{Ai}^C - t_{Aj}^A \|_2}{\| t_{Aj}^A - t_{Ai}^A \|_2}
\]

In practice, we use the median over all \( i, j \) pairs to construct \( S_{AC} \). \( S_{BC} \) is recovered similarly.
Recovering Transformations Let \( T_{AC} \in \text{SIM}(3) \) be the transformation from NeRF \( A \) to \( C \). Using camera \( A_i \) as bridge, we have \( T_{AC} = G_{A}^{S}S_{AC}G_{A}^{-1}, \) \( i \). In practice, we compute \( T_{AC} \) over all instances of camera \( A_i \), and choose the closest valid \( \text{SIM}(3) \) transformation to the median result. \( T_{BC} \) is recovered similarly. We then compute NeRF \( B \) to NeRF \( A \) transformation as \( T_{BA} = T_{AC}^{-1}T_{BC}. \)

Robustness to pose estimation errors While our proposed registration method works better with more accurate SfM results on NeRF-synthesized images, it is also robust to SfM’s errors. When computing the relative scale, we only need to recover at least two poses (so that at least one pair is formed to be used in Equation 14) from each NeRF’s re-renderings. This is easily achievable with a reasonably sampled set of query poses. Moreover, since we consider the median result as the final estimation, the impact of erroneous poses will be minimal. A similar analysis also holds for transformation recovery, except that only a single pose is needed for the estimation.

3.3. NeRF Blending

Given two or more registered NeRFs and a query camera pose, NeRF blending [33] aims to combine predictions from the individual NeRFs with the goal of high-quality novel view synthesis. Without loss of generality, we consider again the two-NeRF setting: \( A \) and \( B \) with relative transformation \( T_{BA} \in \text{SIM}(3). \) Let \( G_B \in \text{SE}(3) \) be a pose defined in NeRF \( B \)’s coordinate system that can be used to query NeRF \( B \). To get the corresponding pose \( G_A \) to query NeRF \( A \), we first decompose \( T_{BA} = G_{BA}S_{BA}, \) and compute \( G_A = T_{BA}G_BS_{BA}^{-1}. \)

For blending, there are three key concepts to consider: (i) when to blend: in what case should it be used; (ii) what to blend: at what granularity should it happen; and (iii) how to blend: in which way should we compute blending weights? Block-NeRF [33] answers (i) with visibility thresholding, where if the mean visibility of a frame (predicted by a visibility network) is above a certain threshold then blending is activated. Afterwards, it answers (ii) by introducing image- and pixel-wise blending. Finally, it handles (iii) by inverse-distance-weighting (IDW) and predicted visibility weighting. Importantly, to achieve any of these results, a visibility prediction network has to be trained jointly with the NeRF and used during inference. In our setting, we do not assume access to a visibility network, since we are dealing with black-box uncalibrated NeRFs not generated for this particular purpose.

In this paper, we answer (i) by proposing a simpler threshold that is solely based on distance. We answer (ii) by proposing a novel sample-based blending method recognizing the fact that the color of a pixel is computed using samples along the ray in NeRF during volumetric rendering. We answer (iii) by proposing an IDW method for our sample-based blending. Since we use IDW with sample-based blending, we coin our method \textit{IDW-Sample}.

Without loss of generality, we discuss the blending of two registered NeRFs, \( A \) and \( B \). Our findings easily extend to an arbitrary number of NeRFs and also to any volumetric representation.

3.3.1 Distance Test for NeRF Selection

The decision of when to render using blended NeRFs, rather than just one NeRF, is an important question, because NeRFs can only render with high-quality within their effective range. Rendering using distant NeRFs, whose rendering quality is poor, can only be harmful. Hence, we introduce a test based on the distance between the origin of the query camera and the NeRF centers. Denoting the distances from NeRF \( A \) and NeRF \( B \) as \( d_A \) and \( d_B \), the test value is \( \tau = \max \left( \frac{d_A}{d_B}, \frac{d_B}{d_A} \right) \). If \( \tau \) is greater than a threshold, it means that the second-closest NeRF is sufficiently far, in which case it is better to simply use the rendering of the closest NeRF to the query camera; otherwise, IDW-based blending is enabled.

3.3.2 IDW-Sample for NeRF Blending

During NeRF’s volumetric rendering stage, a pixel’s color is computed using samples along the ray. Recognizing this fact, we propose a sample-wise blending method that calculates the blending weights for each ray sample using IDW. We show that the original volumetric rendering methodology can be easily extended to take advantage of these new sample-wise blending weights, resulting in our proposed \textit{IDW-Sample} strategy. Merge Ray Samples Consider a pixel to be rendered, which gets unprojected into a ray. Since ray samples are separately proposed according to the density field of each source NeRF, we need to merge them...
4. Experiments

In this section, we describe our registration and blending experiments on ScanNet, Block-NeRF, and an object-centric scene dataset we collect, which we will make available.

4.1. Datasets

Object-Centric Indoor Scenes  We created a dataset consisting of three indoor scenes, using an iPhone 13 mini in video mode. Each scene consists of three video clips — we choose two objects in each scene, and collect two (overlapping) video sequences that focus on each object. Then, we collect a third (test) sequence that observes the entire scene.

We extract images at 3.75 fps from all three video clips and feed them jointly to an structure-from-motion (SfM) tool (we use the hloc toolbox [26, 27]) to recover their poses. These poses are defined in a shared coordinate system, which we denote as $C$. We then center and normalize the poses for each training set, so that the processed poses are located within the bounding box $[-1, 1]^3$. Note that this step induces a local coordinate system for each NeRF, which we denote as $A$ and $B$ respectively. We record the resulting transformations $\{T_{CA}, T_{CB}\} \subset \text{SIM}(3)$ and treat them as ground-truth. NeRFS $A$ and $B$ are then trained separately from the corresponding training set of images. We test NeRFuser on this dataset and report results of both registration and blending in Section 4.2.

ScanNet Dataset  Since the “ground-truth” poses of our dataset are estimated using SfM based only on RGB images, they are potentially not as accurate as what could be obtained using RGB-D data. Hence, we use the ScanNet dataset to further test registration performance. The ScanNet dataset provides a total of 1513 RGB-D scenes with annotated camera poses, from which we use the first 218 scenes. We downsample the frames so that roughly 200 posed RGB-D images are kept from each scene. We then split the images into three sets: two for training NeRFS and one for testing. Specifically, images are split according to their temporal order. We first randomly select 10% of all images as the test set. Of the remaining images, we label the first 25% as training set $A$, the last 25% as training set $B$, and randomly label the middle 50% as either $A$ or $B$. This splitting strategy creates a moderate spatial overlap among $A, B$ and the test sets. Once we have the splits, we center and normalize the training poses the same way as in Section 4.1. The resulting transformations $T_{CA}, T_{CB}$ are recorded as ground-truth. After generating the NeRFS, we check their quality according to validation PSNR and keep the best 25 scenes. We test NeRF registration on this dataset and compare with point-cloud registration in Section 4.3.

Mission Bay Dataset  To further test the rendering quality of different blending methods, we run experiments on the Mission Bay dataset from Block-NeRF [33], which features a street scene from a single capture. The dataset is collected using 12 cameras that capture the surround of a car that drives along a straight street with a quarter turn.
Table 1: Blending results on Object-Centric Indoor Scenes. IDW-Sample works the best for all metrics with both ground-truth and estimated transformations. Results with estimated $\hat{T}_{BA}$ are only marginally worse than those with ground-truth $T_{BA}$, which demonstrates that our proposed NeRF registration is accurate enough for the downstream blending task.

<table>
<thead>
<tr>
<th>Blending</th>
<th>Ground-truth $T_{BA}$</th>
<th>Estimated $\hat{T}_{BA}$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PSNR ↑</td>
<td>SSIM ↑</td>
</tr>
<tr>
<td>NeRF</td>
<td>20.92</td>
<td>0.716</td>
</tr>
<tr>
<td>Nearest</td>
<td>23.81</td>
<td>0.779</td>
</tr>
<tr>
<td>IDW-2D</td>
<td>24.70</td>
<td>0.795</td>
</tr>
<tr>
<td>IDW-3D</td>
<td>23.48</td>
<td>0.776</td>
</tr>
<tr>
<td>IDW-Sample (Ours)</td>
<td><strong>24.91</strong></td>
<td><strong>0.813</strong></td>
</tr>
</tbody>
</table>

Figure 5: NeRF blending with IDW-based methods on the Mission Bay dataset. Per-pixel errors are visualized as heat maps. Individual NeRFs renderings have large artifacts on either side, which are best resolved by IDW-Sample blending.

4.2. NeRF Fusion

We test NeRFuser including both NeRF registration and NeRF blending on Object-Centric Indoor Scenes. For registration, we generate 32 poses that are roughly uniformly placed on the upper hemisphere of radius 1, with elevation from 0 to 30°. We use them as local poses to query NeRFs $A$ and $B$, and feed the 64 synthesized images jointly to SfM. NeRF $B$ to NeRF $A$ transformation $\hat{T}_{BA}$ is recovered using our proposed procedure. To evaluate its accuracy, given ground-truth and estimated transformations $\{T, \hat{T}\} \subset \text{SIM}(3)$, we first compute $\Delta T = \hat{T}T^{-1}$. It is then decomposed into $\Delta G \in \text{SE}(3)$ and $\Delta S \in \text{SIM}(3)$ as $\Delta T = \Delta G \Delta S$. Rotation error $r_{err}$ is computed as the angle (in degrees) of $\Delta G$'s rotation matrix. Translation error $t_{err}$ is computed as the L2 norm of $\Delta G$'s translation vector. Note that by definition, $t_{err}$ is measured in NeRF $A$’s unit. For scale error, we extract $\Delta s = |\Delta S|^{1/3}$ and compute $s_{err} = |\log \Delta s|$. We report $T_{BA}$ errors against ground-truth: $r_{err} = 0.031^\circ$, $t_{err} = 0.0013$, $s_{err} = 0.0045$.

For blending, we set distance test ratio $\tau = 1.8$ and blending rate $\gamma = 5$. Since it depends on the NeRF $B$ to NeRF $A$ transformation $\hat{T}_{BA}$, we report results in two settings: (i) using ground-truth $T_{BA}$ and (ii) using estimated $\hat{T}_{BA}$. The second setting is specifically used to showcase the compound performance of the full NeRFuser framework. In addition to IDW-based methods, we also include
Figure 6: Effect of re-rendering poses on NeRF registration. With more sampled poses, the registration errors go down while success rate improves (green curve). Using additional hemispheric poses besides the training ones also proves helpful (orange line vs. blue line). More interestingly, with a large enough ratio $\rho$, registration with hemispherically sampled poses outperforms training poses when using the same number or fewer poses in total. It shows that it is beneficial to have a larger spatial span of re-rendering poses for registration as illustrated in Figure 8.

NeRF and Nearest as baselines. NeRF directly uses NeRF-synthesized images, while Nearest uses the rendering from the closer NeRF to the query pose. We evaluate blending results against ground-truth images on three metrics: PSNR [11], SSIM [36] and LPIPS [42]. We report numbers averaged over test images of all three scenes from our dataset in Table 1.

<table>
<thead>
<tr>
<th>Registration</th>
<th>$r_{err}$ ($^\circ$)</th>
<th>$t_{err}$</th>
<th>$s_{err}$</th>
<th>Success</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF-extracted point-cloud</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICP [23]</td>
<td>3.027</td>
<td>0.1151</td>
<td>N/A</td>
<td>0.13</td>
</tr>
<tr>
<td>FGR [44]</td>
<td>4.549</td>
<td>0.1844</td>
<td>N/A</td>
<td>0.04</td>
</tr>
<tr>
<td>FPFH [24]</td>
<td>2.805</td>
<td>0.0381</td>
<td>N/A</td>
<td>0.17</td>
</tr>
<tr>
<td>RGB-D-fused point-cloud</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ICP [23]</td>
<td>1.598</td>
<td>0.0816</td>
<td>N/A</td>
<td>0.17</td>
</tr>
<tr>
<td>FGR [44]</td>
<td>1.330</td>
<td>0.0372</td>
<td>N/A</td>
<td>0.71</td>
</tr>
<tr>
<td>FPFH [24]</td>
<td>0.049</td>
<td>0.0205</td>
<td>N/A</td>
<td>0.79</td>
</tr>
<tr>
<td>NeRFuser</td>
<td>0.588</td>
<td>0.0315</td>
<td>0.0211</td>
<td>0.84</td>
</tr>
</tbody>
</table>

Table 2: Registration results on ScanNet. We compare to point-cloud registration methods on both NeRF-extracted point-cloud and ground-truth RGB-D-fusion point-cloud. Due to the noisy geometry of NeRF reconstructions, registration performance on NeRF-extracted point-clouds is inferior. However, NeRFuser is comparable to the registration performance on RGB-D-fused methods in terms of $r_{err}$ and $t_{err}$, while having the highest success rate. Our method also recovers the relative scale, while point-cloud baselines work on the relaxed problem of SE(3) pose recovery. Bold numbers are the best, italic numbers are second best.

4.3. NeRF Registration

To further test the registration performance on a large-scale dataset, we use the ScanNet dataset [7] as prepared according to Section 4.1. We repeat the same registration procedure as detailed in Section 4.2, except that 60 hemispheric poses are sampled instead of 32. During experiments, we notice failure cases due to NaN or outlier values. To report more meaningful numbers, we treat cases that meet any of the following conditions as failure: (i) is NaN or (ii) $r_{err} > 5^\circ$ or (iii) $t_{err} > 0.2$ or (iv) $s_{err} > 0.1$.

We also compare our method against various point-cloud registration (PCR) baselines using both (i) point-clouds extracted from NeRFs and (ii) point-clouds fused from ground-truth posed RGB-D images. We describe the point-cloud data preparation for each scene as below. While there are scale-adaptive methods [25] for PCR, most available and well-tested implementations presume that the two point-clouds to be registered are measured in the same unit. Since $T_{BA}$ is measured in NeRF $A$’s unit, we make sure that the measuring unit of both point-clouds is the same as NeRF $A$’s. For (i) NeRF-extracted point-clouds, the unit conversion is achieved by applying $S_{BA}$ to point-cloud $B$. For (ii) RGB-D fused point-clouds, the unit conversion is achieved by applying $S_{CA}$ to both point-clouds. Additionally, note that in this case all ground-truth poses are defined in the same world coordinate system. To enable a fair comparison, we further transform RGB-D fused point-cloud $A$ by $G_{BA}$ after unit conversion. After these processing steps, we get point-clouds ready for registration, whose ground-truth solution is $G_{BA}$ for both (i) and (ii). We report in Table 2 the results of our registration method and various PCR baselines averaged over all successfully registered scenes, as well as the success rate.

4.4. NeRF Blending

To further test our blending performance, we use the outdoor Mission Bay dataset as described in Section 4.1, with ground-truth transformations. We set the distance test ratio $\tau = 1.2$, and the blending rate $\gamma = 10$. Quantitative results averaged over test images of all scenes are reported in Table 3. Qualitative results are visualized in Figure 5.
Figure 7: Effect of blending rate $\gamma$ in IDW-based blending ranging from $[0.01, 1000]$. For all blending methods, quality initially increases with $\gamma$, but then decreases as $\gamma$ increases further.

Blending results on Mission Bay dataset. IDW-Sample performs the best for all metrics.

<table>
<thead>
<tr>
<th>Blending</th>
<th>PSNR ↑</th>
<th>SSIM ↑</th>
<th>LPIPS ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>NeRF</td>
<td>17.306</td>
<td>0.571</td>
<td>0.502</td>
</tr>
<tr>
<td>Nearest</td>
<td>19.070</td>
<td>0.657</td>
<td>0.398</td>
</tr>
<tr>
<td>IDW-2D</td>
<td>19.692</td>
<td>0.659</td>
<td>0.413</td>
</tr>
<tr>
<td>IDW-3D</td>
<td>18.806</td>
<td>0.636</td>
<td>0.433</td>
</tr>
<tr>
<td>IDW-Sample (Ours)</td>
<td>19.986</td>
<td>0.678</td>
<td>0.388</td>
</tr>
</tbody>
</table>

4.5. Ablation Studies

Ablation on re-rendering poses for NeRF registration

We study the registration performance on ScanNet dataset with the number of sampled poses. To account for the fact that each scene may be of a different scale, we introduce $\rho$ as the ratio of the number of sampled poses over the number of training views. We geometrically sample $\rho \in [0.167, 1.3]$, and generate the hemispheric poses accordingly. We evaluate the performance of NeRF registration w.r.t. $\rho$ averaged over all ScanNet scenes. In addition, we include 2 more settings: (i) training poses only: instead of hemispheric poses, use NeRF’s training poses for re-rendering; (ii) hemispheric + training poses: use NeRF’s training poses together with hemispherically sampled ones ($\rho = 0.3$) for re-rendering. Results are reported in Figure 6. We want to additionally highlight that, since registration using a relatively small number of sampled poses for re-rendering can still work well, it implies that the registration procedure will not take long. In practice, it typically only takes minutes to finish.

Ablation of $\gamma$ in IDW-based blending

We study the effect of blending rate $\gamma$ in IDW-based blending on Object-Centric Indoor Scenes. Specifically, we use ground-truth transformations and set distance test ratio $\tau = 1.8$. We geometrically sample $\gamma$ in $[10^{-2}, 10^3]$. For each sampled $\gamma$, we blend NeRFs with all IDW-based methods and report the results averaged over test images of all 3 scenes from Object-Centric Indoor Scenes. Since Nearest and NeRF are not affected by $\gamma$, we draw dotted horizontal lines for comparison. The results are shown in Figure 7. In $\gamma \to 0$ case, all IDW-based methods become the same as using the mean image. In $\gamma \to \infty$ case, IDW-2D becomes the same as Nearest, while IDW-Sample becomes analogous to KiloNeRF [21] (more details in appendix 6.2). We find the optimal $\gamma$ in between the extremes for any IDW-based method. Moreover, our proposed IDW-Sample almost always performs the best for any given $\gamma$.

Ablation on Blending Performance over Query Poses

We provide a qualitative ablation study to showcase the performance of our proposed blending method IDW-Sample against baselines for different test poses with respect to two NeRF centers in Figure 9. The study is supposed to provide a geometric sense of where the blending method gives the most benefits.

5. Conclusion

We have introduced NeRFuser, a NeRF fusion pipeline that registeres and blends arbitrary many NeRFs treated as input data. To address the problem of registration, we propose registration from re-rendering, taking advantage of NeRF’s ability to synthesize high quality novel views. To address the problem of blending, we propose IDW-Sample, leveraging the ray sampling nature of NeRF rendering. While we have demonstrated NeRFuser’s robust performance in multiple scenarios, it inherits any of the failure cases of the input NeRFs. Variants that use structured priors can easily be integrated into our framework, since it
Figure 9: A visualization of how IDW-Sample performs against Nearest. The axes denote the reference frames for two input NeRFs, while the green camera frusta denote the test camera poses. The darker the camera frusta, the better IDW-Sample performs compared to Nearest. The best-performing poses are those that evenly observe the scene.

is agnostic to the source NeRFs. We believe this tool will help enable the increased proliferation of implicit representations as raw data for future 3D vision applications.

References


6. Ablation

6.1. Ablation of $\gamma$ in IDW-based Blending on the Mission Bay Dataset

We additionally present the blending rate $\gamma$ ablation on Mission Bay Dataset. Specifically, we use ground-truth transformations and set distance test ratio $\tau = 1.2$. We geometrically sample $\gamma$ in $[1, 10^3]$. For each sampled $\gamma$, we blend NeRFs with all IDW-based methods and report the results averaged over test images of all scenes from the Mission Bay Dataset. Since Nearest and NeRF are not affected by $\gamma$, we draw dotted horizontal lines for comparison. The results are shown in Figure 10.

6.2. Connection to KiloNeRF

In this section we establish a relationship between IDW-Sample and KiloNeRF [21]. Within the framework of our IDW-Sample method, KiloNeRF is a special case when the blending rate $\gamma \to \infty$. Intuitively, KiloNeRF employs a grid of small NeRFs within the axis-aligned bounding box of the scene, where each small NeRF is only responsible for the spatial cube it occupies. Specifically, given a sample $(\delta, t)$ on ray $(o, d)$, KiloNeRF first determines which grid it falls into based on the ray sample location $o + td$. The NeRF corresponding to this grid will be given a weight of 1, while all other NeRFs will be weighted 0. In IDW-Sample, if we set the power $\gamma$ to infinity, it means that, for each ray sample, it will only take the field information from the closest NeRF, but not a weighted sum of information from all NeRFs. Thus, only the closest NeRF becomes responsible for that sample, which resembles the case for KiloNeRF. Our method generalizes this approach, since we can freely choose a $\gamma$ smaller than infinity to tune the range that each NeRF is responsible for, which results in better rendering quality (see how IDW-Sample in Figure 10 degrades as $\gamma \to \infty$). An ablation study of $\gamma$ is provided on both the Object-centric Indoor Scenes (in the main document) and the Mission Bay Dataset (in subsection 6.1).