DReg-NeRF: Deep Registration for Neural Radiance Fields

Yu Chen
Department of Computer Science, National University of Singapore
chenyu@comp.nus.edu.sg

Gim Hee Lee
Department of Computer Science, National University of Singapore
gimhee.lee@nus.edu.sg

Abstract

Although Neural Radiance Fields (NeRF) is popular in the computer vision community recently, registering multiple NeRFs has yet to gain much attention. Unlike the existing work, NeRF2NeRF [14], which is based on traditional optimization methods and needs human annotated keypoints, we propose DReg-NeRF to solve the NeRF registration problem on object-centric scenes without human intervention. After training NeRF models, our DReg-NeRF first extracts features from the occupancy grid in NeRF. Subsequently, our DReg-NeRF utilizes a transformer architecture with self-attention and cross-attention layers to learn the relations between pairwise NeRF blocks. In contrast to state-of-the-art (SOTA) point cloud registration methods, the decoupled correspondences are supervised by surface fields without any ground truth overlapping labels. We construct a novel view synthesis dataset with 1,700+ 3D objects obtained from Objaverse to train our network. When evaluated on the test set, our proposed method beats the SOTA point cloud registration methods by a large margin with a mean RPE $= 9.67^\circ$ and a mean RTE $= 0.038$. Our code is available at https://github.com/AIBluefisher/DReg-NeRF.

1. Introduction

Scene reconstruction has many applications in the real world, for example, in augmented reality, ancient culture preservation, 3D content generation, etc. Recently, rapid progress has been made in increasing the reconstruction quality using neural radiance fields (NeRF) [26]. While previous works mostly focused on synthesizing images at the object level or unbounded scenes within a small area, Block-NeRF [34] extends NeRF to city-scale scenes by splitting data into multiple intersected blocks. Specifically, Block-NeRF trains multiple NeRF models in the same coordinate frame with the ground-truth camera poses provided by fusing multiple high-precision sensors. However, images can be collected without absolute pose information in some cases, e.g., when images are captured with digital cameras or in GPS-denied areas. In such cases, Block-NeRF cannot work since NeRF models are trained on different coordinate frames. Consequently, NeRF registration [14] is necessary for synthesizing consistent novel views from multiple NeRFs trained in different coordinate frames.

Point cloud registration is a classic problem in 3D computer vision, which aims at computing the relative transformation from the source point cloud to the target point cloud.
However, NeRF registration is under-explored since existing works focus mostly on point cloud registration. Unlike point clouds that are simple explicit representations, NeRF encodes scenes implicitly, which makes registering multiple NeRFs more challenging. NeRF2NeRF [14] is the first work that tried to solve registering NeRFs by a traditional optimization-based approach. However, it requires human-annotated keypoints for initialization, which limits its application in the real world where human annotations can be impractical. In view of the above-mentioned challenges, we focus on the study of the NeRF registration problem by answering the following two questions: 1) Can we register two or multiple NeRFs together where only pre-trained models are accessible? 2) How to register NeRFs without any human annotations and initializations?

We further use the following settings in our endeavor to answer the two challenging questions on NeRF registration: 1) Images are collected into different blocks and no images are associated with known absolute position information. 2) Multiple NeRF blocks are trained individually where ground truth camera poses in each block are in their local coordinate frames. 3) Only the trained NeRF models are accessible, and all training images are removed and therefore not available due to plausible privacy-preserving issues or disk limitations. We emphasize that NeRF registration is a challenging task, and we focus more on the object-centric scenes in this paper. See Fig. 1 for the illustration of our task setting and dataset construction.

To solve the NeRF registration problem, we first utilize an occupancy grid along with each NeRF model to extract a voxel grid. The voxel grid is then fed into a 3D Feature Pyramid Network (FPN) [21] to extract features. The resulting voxel feature grids are further processed by a transformer module. In the transformer network, we first adopt a self-attention layer to enhance the intra-feature representations within each voxel feature grid. We further utilize a cross-attention layer to learn the inter-feature relations between the source feature grid and the target feature grid. Finally, we use an attention head to decode the source features and target features into correspondences and confidence scores. Unlike SOTA point clouds registration methods [16, 42], we utilize NeRF as geometric supervision and thus do not rely on pre-computed overlapping scores to mask correspondences outside the overlapping areas.

The main contributions of our work are:

- A dataset for registering multiple NeRF blocks, which is created by rendering 1,700+ 3D objects that are downloaded from the Objaverse dataset.
- A novel network for registering NeRF blocks which do not rely on any human annotation and initializations.
- Exhaustive experiments to show the accuracy and generalization ability of our method.

To the best of our knowledge, this is the first work on registering NeRFs without a) any initializations from keypoints or other registration methods and b) precomputed ground-truth overlapping labels.

2. Related Work


Block-NeRF [34] partition the collected images into different street blocks, each block is trained individually along with jointly optimizing camera poses [20, 5]. The final images can be synthesized by fusing from multiple nearby NeRF blocks. Mega-NeRF [37] also adopted the same divide-and-conquer strategy as BlockNeRF, but focus more on aerial images and exploiting the sparse network structures. To train their network, Block-NeRF obtained camera poses by fusing data from different sensors, while Mega-NeRF used Structure from Motion (SfM) [23, 9, 7, 8] tools to recover camera poses. Both of them assume camera poses are in the same coordinate frames.

Point Cloud Registration. The Iterative Closest Point (ICP) [2, 6] algorithm has been widely applied to the industry community and research community for years. Given a rough initialization, ICP tries to align the source point cloud to the target point cloud. The global point cloud registration methods can align point clouds without initialization by extracting geometric features such as the Fast Point Feature Histograms (FPFH) [30]. To solve the long-time issue of the global registration method in evaluating candidate models within RANSAC [12], Zhou and Park [45] proposed a fast approach that does not need to evaluate the candidate models at each iteration. Deep point cloud registration methods also gained much attention nowadays. Deep Closest Point [41] is a learned variant of the classical ICP. Inspired by SuperGlue [32], which is a deep learning method for matching
2D image correspondences, Predator \[16\] and REGTR \[42\] adopted the self-attention and cross-attention mechanisms from SuperGlue to learn the correlation for pairwise low-overlapping point clouds. The ground-truth overlapping scores are computed from dense point clouds and used to mask out the correspondences outside the overlapping regions. We also follow the previous attention mechanisms, but do not rely on the pre-computed overlapping labels.

**NeRF Registration.** NeRF2NeRF \[14\] is the first work that tries to register multiple NeRFs. The initial transformation is estimated from human-annotated keypoints, and then refined by the surface fields from NeRF. To reduce the number of useless samples, NeRF2NeRF adopts Metropolis-Hasting sampling to maintain an active set. The whole framework is based on traditional optimization methods and needs human interaction. ZeroNeRF \[28\] claims it can register NeRF without overlap. However, it still requires a global registration method for initialization. Unlike the previous works, which rely heavily on traditional optimization methods, we try to register multiple NeRF blocks by learning methods without human interaction.

### 3. Our Method

Fig. 2 shows an illustration of our framework. Our network takes a source NeRF model and a target NeRF model as input, and outputs correspondences in the source NeRF and target NeRF. We first train multiple NeRF blocks in different coordinate frames. For each NeRF model, we associate it with an occupancy voxel grid, where each voxel indicates whether it is occupied or not. After training, we extract a 3D voxel grid for each NeRF model and then feed it into a 3D CNN backbone to extract features. Subsequently, we use a transformer with self-attention and cross-attention layers to learn the relations between the pairwise feature vectors. We then adopt a decoder to decode the resulting features \( F_{\text{source}}, F_{\text{target}} \) into correspondences \( \{X_{\text{source}}, X_{\text{target}}\} \) and the corresponding confidence scores \( \{S_{\text{source}}, S_{\text{target}}\} \).

Finally, the relative transformation can be solved by the weighted Kabsch-Umeyama algorithm \[38\] from the correspondences.

### 3.1. Background

**Neural Radiance Fields.** Neural Radiance Fields (NeRF) aims at rendering photo-realistic images from a new view point. For a 3D point \( X \), the density field \( \sigma_t \) of \( X \) is defined as the differential probability of a ray \( r = o + td \) hitting a particle, where \( o \) is the camera center, \( d \) is the view direction. The transmittance \( T(t) \) denotes the probability of ray without hitting any particles when traveling a distance \( t \), and the discrete form of \( T(t) \) is:

\[
T_n = T(0 \rightarrow t_n) = \exp \left( \sum_{k=1}^{n-1} -\sigma_k \delta_k \right). \tag{1}
\]

Given a set of points \( \{X_n = o + t_n d | n \in [0, K]\} \), NeRF predicts the view-independent volume density \( \sigma_n \) and view-dependent radiance field by:

\[
\sigma_n, e_n = F(X_n; \Theta),
\]
\[
c_n = F(r, e; \Theta), \quad \tag{2}
\]

where \( e_n \) is an embedding vector and \( \Theta \) denotes the network parameters. The final color of an image pixel can be rendered by:

\[
C(t_{N+1}) = \sum_{n=1}^N T_n \cdot \left( 1 - \exp(-\sigma_n \delta_n) \right) \cdot c_n. \quad \tag{3}
\]

We recommend interested readers to refer to \[26, 33\] for the detailed derivation.

### 3.2. Querying Radiance Fields from NeRF

To extract features for the latter learning modules, we first construct a 3D volume from NeRF. Specifically, we assume each NeRF is trained within a bounding box with a 3D voxel grid of resolution \([x_{\text{ref}}, y_{\text{ref}}, z_{\text{ref}}]\). We then obtain the point locations \( \{X\} \) of voxel centers. We also obtain a binary occupancy mask \( M_{\text{occ}} \) from the occupancy grid, where voxels that are not occupied denote empty space and thus are ignored. The occupancy grid is the acceleration structure used in InstantNGP \[27\] to skip empty space in NeRF training. Each grid has a resolution of \(128^3\) that is centered around \((0, 0, 0)\) and stores occupancy as a single bit. During ray marching, a sample point is skipped if the bit of a grid cell is low. We can query the density fields \( \{\sigma\} \) and radiance fields \( \{e\} \) using Eq. \(2\) with the coordinates \( \{X\} \).

Since density fields can be noisy, we further obtain a density mask \( M_{\text{df}} = \sigma > \sigma_t \) by setting a threshold \( \sigma_t \) (we use \( \sigma_t = 0.7 \)). We then obtain our mask as \( M = M_{\text{occ}} \cap M_{\text{df}} \).

One issue with radiance fields is that they are view-dependent. However, the training views are different for each NeRF block. Suppose the NeRF is trained by \( N \) views with camera poses \( P = [R | t] \) which project points from camera frame to world frame. To obtain the radiance fields, we form \( N \) viewing rays \( r = o + td \), where \( d = \frac{X - o}{\|X - o\|} \) for each query point and average the queried radiance fields over all rays. The final color \( C \) can be obtained by volume rendering of Eq. \(3\). Furthermore, we compute the alpha compositing value by \( \alpha = 1 - \exp(-\delta \delta) \), where \( \delta \) is a chosen small value. We associate each point in the voxel grid with \( [X, C, \alpha] \) and feed the voxel grid \( G \in \mathbb{R}^{x_{\text{ref}} \times y_{\text{ref}} \times z_{\text{ref}} \times C} \) and a mask \( M \in \mathbb{R}^{x_{\text{ref}} \times y_{\text{ref}} \times z_{\text{ref}} \times 1} \) into a 3D CNN to extract backbone features, where \( C = 7 \) with 3 channels from the point coordinate, three from the color channels, and one from the scalar \( \alpha \).

### 3.3. Feature Extraction

Given the voxel grid, we adopt the feature pyramid network \[21\] to extract features. We use ResNet \[15\] as the
feature backbone, where all the 2D modules are replaced by their corresponding 3D parts. The feature pyramid network enables the learning of high-level semantic features at a multi-scale and thus is suitable to extract the voxel grid features in our task settings. We note the difference from the original feature pyramid network which utilizes different scale features for object detection, we adopt only the features output from the last layer. Since the dimension of the feature grid from the last layer can be different from the original voxel grid, we rescale it to the size of the original voxel grid \( \mathbf{G} \) and resulted in a voxel feature grid \( \mathbf{G}_f \in \mathbb{R}^{N_f \times \text{dim} \times \text{dim} \times C_f} \).

We cannot use \( \mathbf{G}_f \) as the input to our transformer since the voxel feature grid can contain too many points to enable the transformer to run on a single GPU. To solve this issue, we iteratively downsample \( \mathbf{G}_f \) by the spherical neighborhood [35] as done in KPConv [36] to obtain the downsampled voxel points \( \mathbf{X} \). The downsample iteration is terminated when the total number of occupied voxels in the current sampled voxel feature grid is less than 1.5K. Lastly, we reshape the occupied voxel features into a tensor \( \mathbf{F} \in \mathbb{R}^{N \times C} \). The weights of the feature extraction network are shared across the source and the target NeRFs. We denote the downsampled voxel points and extracted features of the source and target NeRFs as: \( \mathbf{X}_{\text{source}}, \mathbf{F}_{\text{source}} \) and \( \mathbf{X}_{\text{target}}, \mathbf{F}_{\text{target}} \), respectively.

### 3.4. Transformer

We then feed the resulting feature \( \mathbf{F}_{\text{source}} \) and \( \mathbf{F}_{\text{target}} \) into a \( L \)-layer transformer with self-attention and cross-attention layers. We follow the same intuition of Predator [16] and REGTR [42], where a self-attention layer is applied to both the source feature \( \mathbf{F}_{\text{source}} \) and the target feature \( \mathbf{F}_{\text{target}} \) to enhance the intra-contextual relations, and a cross-attention layer is applied to both \( \mathbf{F}_{\text{source}} \) and \( \mathbf{F}_{\text{target}} \) to learn the inter-contextual relations. We follow the classical transformer architecture [40, 19, 42] with input to be voxel features.

**Multi-Head Attention.** The multi-head attention operation in each layer is defined as:

\[
\text{MH}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concat}(\text{head}_1, \cdots, \text{head}_n)\mathbf{W}^O, \tag{4}
\]

where \( \text{head}_i = \text{Attention}(\mathbf{QW}^i, \mathbf{KW}^i, \mathbf{VW}^i) \). The attention function is adopted as the scaled dot-product:

\[
\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}. \tag{5}
\]

In the self-attention layers, \( \mathbf{Q} = \mathbf{K} = \mathbf{V} \) represents the same feature tensor in each block. In the cross-attention layers, the keys, and the values are the feature tensors from the other block. The self-attention mechanism enables the network to learn the relationship inside the same feature points, while the cross-attention mechanism enables the communication of the different feature points.

**Decoder.** After encoding features by transformer, we further adopt a single-head attention layer to predict the corresponding point locations \( \mathbf{X}_{\text{source}} \) and confidence scores \( S_{\text{source}} \) of the source voxel points \( \mathbf{X}_{\text{source}} \) in the target NeRF’s coordinate frame. Similarly, we also predict the corresponding point locations \( \mathbf{X}_{\text{target}} \) and confidence scores.

---

**Figure 2:** Network architecture of our method. The pipeline of our method is: 1) We first extract the pairwise voxel grid \( \mathbf{G} \in \mathbb{R}^{N \times \text{dim} \times \text{dim} \times C_f} \) and a binary mask \( \mathbf{M} \in \mathbb{R}^{N \times \text{dim} \times \text{dim} \times 1} \) from the source NeRF and the target NeRF. 2) The voxel grid \( \mathbf{G} \) and a binary mask \( \mathbf{M} \) are fed into the 3D feature pyramid network to extract voxel features. 3) The extracted voxel grid features are downsampled to a 2-dimensional tensor \( \mathbf{F} \in \mathbb{R}^{N \times C_f} \) by their spherical neighborhood. 4) The resulting source features \( \mathbf{F}_{\text{source}} \) and target features \( \mathbf{F}_{\text{target}} \) are strengthened by a transformer, where a self-attention layer is used to enhance the intra-contextual relations, and a cross-attention layer is used to learn the inter-contextual relations. 5) Finally, we use a single-head attention layer to decode the features into correspondences and their corresponding confidence scores.
\( \hat{S}_{\text{target}} \) of the target voxel points \( \hat{X}_{\text{target}} \) in the source NeRF’s coordinate frame. Finally, we utilize the predicted correspondences to compute the relative rigid transformation. The confidence scores are used as weights that mask out the irrelevant correspondences and are visible in the source NeRF and the target NeRF.

### 3.5. Training Loss

**Surface Field Supervision.** To train the network, we encourage the predicted correspondences to have the same features which are invariant in the corresponding NeRF model. The naive way is to adopt density fields as supervision instead of radiance fields since density fields are invariant to view points. However, density fields can be very noisy. We thus utilize the surface fields [14] as supervision. The surface field is defined as the differential probability of the ray hitting a surface at \( X_n \) given by:

\[
S(t) = \mathcal{T}(t) \cdot (1 - \exp(-2\sigma_\delta)).
\]

**Proof:** The differential probability of a ray hitting a surface at point \( X \) is the product of the probability of a ray traveling over \([0, t_n)\) without hitting any particle before \( t_n \) times the differential probability of the ray hitting exactly at point \( X(t_n) \). The surface field can then be written as:

\[
S(t) = \int_{t-\delta}^{t+\delta} \mathcal{T}(s) \cdot \sigma(s) \, ds
\]

(7)

We derive the exact form of \( S(t) \) as follow:

\[
S(t) = \int_{t-\delta}^{t+\delta} \mathcal{T}(s) \cdot \sigma(s) \, ds
\]

\[
= \int_{t-\delta}^{t+\delta} \mathcal{T}(0 \rightarrow t - \delta) \cdot \mathcal{T}(t - \delta \rightarrow s) \cdot \sigma(s) \, ds
\]

\[
= \mathcal{T}(t - \delta) \cdot \int_{t-\delta}^{t+\delta} \mathcal{T}(t - \delta \rightarrow s) \cdot \sigma(s) \, ds
\]

\[
= \mathcal{T}(t - \delta) \cdot \sigma_t \cdot \int_{t-\delta}^{t+\delta} \left( \exp \left( - \int_{t-\delta}^{s} \sigma(\mu) \, d\mu \right) \right) \, ds
\]

\[
= \mathcal{T}(t - \delta) \cdot \sigma_t \cdot \int_{t-\delta}^{t+\delta} \left( \exp \left( - \sigma_t(s - t + \sigma) \right) \right) \, ds
\]

\[
= \mathcal{T}(t - \delta) \cdot \sigma_t \cdot \left( \frac{1}{\sigma_t} \right) \cdot \exp \left( - \sigma_t(s - t + \delta) \right) \right|_{t-\delta}^{t+\delta}
\]

\[
= \mathcal{T}(t - \delta) \cdot (1 - \exp(-2\sigma_t \cdot \delta)).
\]

(8)

The second term holds since transmittance is multiplicative (c.f. Eq.(18) of [33]). The 4th term holds since we can assume the density \( \sigma_t \) is a constant within a small region \([t - \delta, t + \delta]\). To produce the view-independent field, we first form \( N \) viewing rays for each point. Subsequently, we take the maximum value of all rays as the density field value instead of averaging the value (as done in Sec. 3.2) for the radiance field. Finally, we obtain the surface field mask \( M_{sf} \) by checking \( S(t) > \eta \) (we use \( \eta = 0.5 \)). We then update our mask by \( M = M_{occ} \cap M_{df} \cap M_{sf} \).

**Confidence Loss.** We adopt the cross-entropy loss to supervise the confidence score, with the surface fields \( \hat{S}_{\text{source}} = S(\hat{X}_{\text{source}}), \hat{S}_{\text{target}} = S(\hat{X}_{\text{target}}) \) queried from NeRF as the ground truth label:

\[
\mathcal{L}_{\text{conf}} = \text{BCE}(\hat{S}_{\text{source}}, \hat{S}_{\text{source}}) + \text{BCE}(\hat{S}_{\text{target}}, \hat{S}_{\text{target}}).
\]

(9)

**Surface Field Loss.** In addition to the confidence loss, we also encourage the predicted correspondences to have consistent surface field values:

\[
\mathcal{L}_{\text{sf}} = \frac{1}{N} \| S(\hat{X}_{\text{source}}, \hat{X}_{\text{target}}) - S(\hat{X}_{\text{source}}, \hat{X}_{\text{target}}) \|_1,
\]

(10)

where \( N \) is the total number of the concatenated source voxel points and target voxel points.

**Correspondence Loss.** We further use a correspondence loss to constrain the predicted locations of correspondences:

\[
\mathcal{L}_{\text{corr}} = \sum_i \rho(\| T^*(x_i) - y_i \|; \eta, \gamma),
\]

(11)

where \( T^* \) is the ground truth relative transformation between the source NeRF and target NeRF, \( \rho(\cdot; \eta, \gamma) \) is the adaptive robust loss function [1] and \( \{\eta, \gamma\} \) are respectively a smooth interpolation value and a scale parameter, which are hyperparameters used to control the shape of the robust function. We use \( \eta = 1, \gamma = 0.5 \) in our experiments.

Moreover, we adopt the feature loss [39, 42] to leverage the geometric properties when computing the correspondences. Our final loss is therefore defined as:

\[
\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{conf}} + \lambda_1 \mathcal{L}_{\text{sf}} + \lambda_2 \mathcal{L}_{\text{corr}} + \lambda_3 \mathcal{L}_{\text{feat}},
\]

(12)

where \( \lambda_1, \lambda_2, \lambda_3 \) are the weights associated with the corresponding loss functions. In our experiments, we use \( \lambda_1 = 1.0, \lambda_2 = 0.1, \lambda_3 = 1.0 \).

### 4. Experiments

**Datasets.** We aim at registering multiple NeRFs. Due to the lack of a suitable dataset for our task, we downloaded the 3D mesh models of 1,700+ objects from Objaverse [10] to construct our dataset. Objaverse [10] is a massive dataset that contains 800K+ annotated 3D Objects. It is created for the text-to-3D task. We utilize it to construct our dataset for NeRF registration. Specifically, we randomly selected
30+ categories and each category contains 40 – 80 objects. As Objaverse contains only 3D objects, we render 120 images for each object where the distribution of camera poses can be seen from Fig. 1 (a). We then split the images into 2 blocks by KMeans. We also add a randomly generated transformation to the original camera poses after splitting data into separate blocks, such that NeRF blocks are trained in different coordinate frames. Each NeRF block is trained in 10K iterations. See Fig. 3 for an overview of our selected training data. The NeRF models trained on all objects are used to train our NeRF registration neural network, and we randomly select 44 objects that are not seen during training for the test.

**Implementations.** We render images for all 1,700+ objects in a computer with an Intel i7 CPU and an NVIDIA GTX 4090 GPU. We run 8 processes concurrently for downloading the mesh models and the job is finished within a week. We use an occupancy grid with a resolution of 128 × 128 × 128 associated InstantNGP [27] as our NeRF representation. To train NeRF, we sample 1024 points for each ray. We use Adam [17] as the NeRF optimizer. We set the initial learning rate to $1 \times 10^{-2}$ and decay it at step $5K, 7.5K, 9K$ with a multiplicative factor being $0.33$. Except for storing the NeRF models, we also store the camera poses and intrinsics as metadata. For our NeRF registration network, we use AdamW [25] as the optimizer with weight decay $1 \times 10^{-4}$. The learning rate is set to $1 \times 10^{-4}$ and halved every $34K$ iteration. The batch size is 1. Our network is trained for 60 epochs, which took about 48 hours to finish. For our transformer, we use $L = 6$ layers and $h = 8$ heads.

**Evaluation.** NeRF2NeRF [14] needs human annotated keypoints for initialization, which are not available on the dataset. Therefore, we do not evaluate NeRF2NeRF on this dataset and use Fast Global Registration (FGR) [45] as the baseline. We also compared it against the state-of-the-art deep point cloud registration method REGTR [42]. For FGR and REGTR, we extract the voxel grid of each NeRF block to a point cloud and use the pairwise point clouds as input to them. For REGTR, we use the model that is pre-trained on the 3DMatch [44] dataset provided by the author. We do not retrain REGTR on the Objaverse dataset since the ground-truth overlapping labels are not available on this dataset. We also evaluated our method Ours$_{df}$ with the surface fields replaced by the density fields as a comparison.

**Results.** The quantitative results can be seen from Tab. 1 and Tab. 2. We report the relative rotation errors (RRE) $\Delta R$ (in degree) and the relative translation errors (RTE) $\Delta t$ as metrics. Note that the scale of translation is unknown and we multiply $\Delta t$ by $1e2$. As we can see, FGR [45] failed in most of the scenes. We think it is the low resolution of our voxel grids that makes FGR [45] fail to find the correspondences. REGTR [42] also fails to find the correct transformations in almost all the scenes. It is worse than FGR in both rotations and translations. We also find very poor generalization ability of Ours$_{df}$. We conjecture that it is due to the density fields being too noisy and not unique for identifying per-scene geometry. In contrast, “Ours” achieves the best results among almost all the scenes – since the network can be regularized to focus on the scene geometry properties by leveraging the surface fields.

We present some qualitative results in Fig. 4. To visual-
align the camera poses to obtain the target coordinate frame and visualize the result in the mations. Moreover, we transform the source prediction to poses in target NeRF
concatenate the transformed camera poses with the camera P to obtain the transformed camera poses
render the images, we first transform the camera poses matching method, Ours
}

Table 1: Quantitative results (first part) of registration on the Objaverse dataset. \( \Delta R \) denotes the relative rotation errors in degree, \( \Delta t \) denotes the relative translation errors multiplied by 1e2 with unknown scales. FGR [45] denotes the fast global matching method, Ours\textsubscript{sf} denotes our method with surface fields replaced by density fields.

<table>
<thead>
<tr>
<th>Plane</th>
<th>Piano 90d</th>
<th>Piano 96e</th>
<th>Truck 1431</th>
<th>Guitar 1554</th>
<th>Guitar 1408</th>
<th>Guitar 96eb</th>
<th>Guitar 00a0</th>
<th>Lantern 023I</th>
<th>Lamp 0230</th>
<th>Bench 0605</th>
<th>Shield 02a7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta R )</td>
<td>FGR [45]</td>
<td>13.84</td>
<td>137.09</td>
<td>7.06</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
</tr>
<tr>
<td>REGTR [42]</td>
<td>13.96</td>
<td>139.48</td>
<td>7.16</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
</tr>
<tr>
<td>Ours\textsubscript{sf}</td>
<td>13.83</td>
<td>137.09</td>
<td>7.06</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
</tr>
<tr>
<td>Ours</td>
<td>13.84</td>
<td>137.09</td>
<td>7.06</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
</tr>
</tbody>
</table>

\( \Delta t \)

<table>
<thead>
<tr>
<th>Fireplug 006d</th>
<th>Fireplug 006c</th>
<th>Fireplug 0512</th>
<th>Shoe 1B63</th>
<th>Shoe 1627</th>
<th>Shoe 0949</th>
<th>Shoe 024c</th>
<th>Teddy 0447</th>
<th>Elephant 138a</th>
<th>Elephant 164b</th>
<th>Elephant 139p</th>
</tr>
</thead>
<tbody>
<tr>
<td>FGR [45]</td>
<td>13.84</td>
<td>137.09</td>
<td>7.06</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
</tr>
<tr>
<td>REGTR [42]</td>
<td>13.96</td>
<td>139.48</td>
<td>7.16</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
</tr>
<tr>
<td>Ours\textsubscript{sf}</td>
<td>13.83</td>
<td>137.09</td>
<td>7.06</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
</tr>
<tr>
<td>Ours</td>
<td>13.84</td>
<td>137.09</td>
<td>7.06</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
<td>11.91</td>
<td>104.03</td>
</tr>
</tbody>
</table>

Table 2: Quantitative results (second part) of registration on the Objaverse dataset. \( \Delta R \) denotes the relative rotation errors in degree, \( \Delta t \) denotes the relative translation errors multiplied by 1e2 with unknown scales. FGR [45] denotes the fast global matching method, Ours\textsubscript{sf} denotes our method with surface fields replaced by density fields.

Ablation Studies. We further show the mean of RRE and RTE in Tab. 3 to ablate our method. “w.o. conf” denotes training our method without the confidence loss, “w.o. sf” denotes training our network without the surface field loss, “Ours\textsubscript{sf}” denotes our method with the surface fields replaced by the density fields. We can see that the surface fields are critical to our network training.

Table 3: Ablations studies of our method. The results are averaged on the 44 test objects.

<table>
<thead>
<tr>
<th></th>
<th>FGR [45]</th>
<th>REGTR [42]</th>
<th>w.o. conf</th>
<th>w.o. sf</th>
<th>Ours\textsubscript{sf}</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Delta R ) &amp; (%)</td>
<td>61.59</td>
<td>113.78</td>
<td>71.84</td>
<td>101.17</td>
<td>86.22</td>
<td>9.67</td>
</tr>
<tr>
<td>( \Delta t ) &amp; (%)</td>
<td>13.50</td>
<td>43.31</td>
<td>14.97</td>
<td>20.35</td>
<td>16.06</td>
<td>3.85</td>
</tr>
</tbody>
</table>
Figure 4: **The qualitative results on Objaverse [10] dataset after NeRF registration.** From left to right are respectively the rendered images by the source NeRF model, the rendered images by the target NeRF model, the side view (SV) of the aligned camera poses, the birds-eye-view (BEV) of the aligned camera poses, the concatenated predictions by transforming the source prediction to the target NeRF’s coordinate frame. **red** and **green**, respectively, denote the results from source NeRF and target NeRF.

Figure 5: Our method failed on unbounded scenes where noisy points are extracted from the occupancy grid.

grid. Instead, we pre-compute the voxel grids $G$ and the corresponding binary mask $M$ for all NeRF blocks and store them on disks. The voxel grid $G$ and binary mask $M$ are loaded into memory at each iteration. During training, our network takes about 2.8 seconds per iteration. The bottleneck on the training time is from loading $G$, $M$, and the source and target NeRF models. During inference, our model takes about 0.4 seconds with the input voxel grid containing about 10K points. Further acceleration can be achieved by pre-downsampling the voxel grid to a lower resolution.

4.1. Further Discussion

**Limitations.** Our method has shown good performance in registering NeRF blocks. However, registering NeRF is still a challenging problem in large-scale scenes. We summarize the limitations of our work as follows (more discussions are given in the supplementary):
• **Generalizability vs. out-of-distribution (OOD).** While our method is generalizable to unseen in-distribution scenes during testing, we postulate that performance would drop when tested on OOD scenes/object classes, e.g., training on indoor and testing on outdoor scenes, etc.

• **Application to real-world data & unbounded scenes.** We emphasize that our training data contains real-world objects (e.g., Shoes in Fig. 3). Our method currently cannot be applied to unbounded scenes since NeRF is not good at geometry estimation. Consequently, incorrect geometries like floaters can influence the performance of our model. It means that our method can fail if the extracted occupancy grid contains too many noisy points (See Fig. 5). We argue that better results can be obtained by applying RANSAC [12] to filter outliers based on our predicted correspondences, or training better NeRF blocks by utilizing depth supervision [11, 29], or utilizing a robust loss [31] to ignoring floaters during training NeRF blocks. In addition, for real-world data that contain background, techniques like [3] can also be applied to our method to get the interested objects. We leave this as our future work.

• **Scale in the relative transformation.** We follow the assumption of NeRF2NeRF that the scales for two NeRFs are the same, which can be violated in real-world settings. Nonetheless, additional sensors such as IMU, wheel encoders, etc., are easily available to get the absolute scale. For settings where only RGB images are available, the scale can be a problem. Additional scene priors are needed to fix the scale for RGB images as input.

**Why localization methods based on SfM tools are not compared?** A simple solution is to first synthesize images using NeRF. We can then use SfM to get 2D-2D correspondences from keypoints matcher and do triangulation to recover the 3D scene points. Consequently, localization-based methods such as perspective-n-points (PnP) [18] or iterative closest point (ICP) [2, 6] can be applied on the 3D scene points to register the NeRF models. However, we argue that SfM is fragile in scenes where keypoint correspondences are difficult to establish. One failure case is given in Fig. 7. As a result, all methods that rely on keypoint correspondences can potentially fail due to wrong matches. Particularly, it is often hard to obtain enough matches for texture-less scenes/objects. False matches can also occur due to changes in image appearance. We circumvent this problem by learning the correspondences from NeRF representations, i.e. the density field, which is shown robust to image appearance changes in the experiments. Moreover, our method is an end-to-end solution and therefore can be much faster than other methods that rely on keypoints, e.g., iNeRF [22] takes more than 50 secs to register an image in an existing NeRF model (c.f. Fig. 4 of the iNeRF paper), while ours only takes 0.4 secs to register two NeRFs.

![Figure 7: COLMAP failed on synthetic dataset due to wrong correspondences.](image-url)

**Why not register images in the same coordinate frame by global bundle adjustment (BA)?** We argue that there are cases where using BA to recover all poses may not be the best option:

• **Robustness.** BA relies on good keypoint correspondences which can be challenging to obtain in texture-less scenes, etc. In contrast, our DReg-NeRF leverages NeRF features for registration without explicit correspondence search on the images.

• **Scalability and efficiency.** Images of a large scene can be collected in smaller batches. It is more scalable and efficient to build smaller NeRF models on each batch of images and subsequently do NeRF registration to get the NeRF model of the full scene.

• **Modularity.** It is easier to update a modular NeRF model. Any module can be easily replaced or added via NeRF registration.

5. **Conclusion**

In conclusion, we have proposed a novel network architecture that registers NeRF blocks into the same coordinate frame. Unlike existing methods, our method does not rely on any initialization and human-annotated keypoints. We constructed a dataset with 1,700+ objects where images are rendered from 3D textured meshes of the Objaverse dataset. We train our method on our constructed dataset. Our method surpasses the state-of-the-art traditional and learning-based point cloud registration methods when evaluated on the test set.

**Acknowledgement.** This research work is supported by the Agency for Science, Technology and Research (A*STAR) under its MTC Programmatic Funds (Grant No. M23L7b0021).
References


[29] Barbara Roessle, Jonathan T. Barron, Ben Mildenhall, Pratul P. Srinivasan, and Matthias Nießner. Dense depth


