LU-NeRF: Scene and Pose Estimation by Synchronizing Local Unposed NeRFs
— Supplementary Material

Zezhou Cheng$^{1,2}$ Carlos Esteves$^2$ Varun Jampani$^2$ Abhishek Kar$^2$
Subhransu Maji$^1$ Ameesh Makadia$^2$
$^1$University of Massachusetts, Amherst $^2$Google Research

Contents

1. Experimental details 1
   1.1. Training GNeRF 1
   1.2. Test-time optimization for view synthesis 1
   1.3. Qualitative and quantitative comparisons 1

2. Supplementary analysis 1
   2.1. Pose synchronization and refinement 1
   2.2. Case study on Drums 2
   2.3. Effect of depth regularization 2
   2.4. Computational cost 2

3. Implementation details 3
   3.1. Building connected graphs 3
      3.1.1 Distance functions 3
      3.1.2 Analysis 3
   3.2. LU-NeRF architecture and training details 4
   3.3. Synchronization and refinement details 4
   3.4. Dataset release and open sourcing 5

1. Experimental details

1.1. Training GNeRF

We trained GNeRF [7] using the official codebase. The scores of GNeRF in our experiments are overall better than those reported in VMRF [11], likely because we trained GNeRF for more iterations. In our experiments, we train GNeRF for 60K iterations which take 48 hours on a single NVIDIA GeForce GTX 1080Ti. We notice that GNeRF is prone to mode collapse in the adversarial training stage, i.e., the generator produces the same or similar sets of outputs with negligible variety, which is a well-known issue for GAN-based models [3]. To achieve similar performance reported in GNeRF and VMRF, we train 5 GNeRF models per prior pose setting and report the results from the best one selected according to the performance on the validation set. Specifically, 35% of the training trials (26 out of 75) suffered from the mode collapse issue on the unordered image collections.

1.2. Test-time optimization for view synthesis

Following the procedure established by prior works [6, 7] for evaluating novel view synthesis, we register the cameras of the test images using the transformation matrix obtained via Procrustes analysis on the predicted and groundtruth cameras of the training images; we then optimize the camera poses of the test images with the photometric loss to factor out the effect of the pose error on the view synthesis quality. In the GNeRF pipeline, the test-time optimization is interleaved with the training process, so the total number of iterations depends on the number of training steps seen for the best checkpoint. In our experiments, GNeRF runs approximately 4230 test-time optimization iterations.

1.3. Qualitative and quantitative comparisons

In the main text, we quote the scores from VMRF where the results on Hotdog are missing. Here we also train GNeRF using the official codebase and report the results in Table 1. This allowed us to generate GNeRF results on the Hotdog scene, and observe the mode collapse reported in the previous section.

Overall, our method outperforms both GNeRF and VMRF under an unconstrained pose distribution, while also being more general – our method works with arbitrary 6 degrees-of-freedom rotations (6DOF), while the baselines assume upright cameras (2DOF) on the sphere, even when the range of elevations is not constrained.

2. Supplementary analysis

2.1. Pose synchronization and refinement

Table 2 demonstrates the necessity of our pose synchronization and refinement steps. The pose synchronization aggregates the local pose estimation from LU-NeRF and provides a rough global pose configuration, and the pose refinement step further improves the global poses.

---

The LU-NeRFs for different mini-scenes are independent. The computational cost of the proposed framework. We randomly sample 30 mini-scenes and report the average training time for LU-NeRF-1 and LU-NeRF-2. Table 4 presents the computational cost of the proposed framework. We randomly sample 30 mini-scenes and report the average training time for LU-NeRF-1 and LU-NeRF-2. The LU-NeRFs for different mini-scenes are independent viewpoints. We find that such depth regularization is crucial to the success of LU-NeRF. Table 3 shows that incorporating the depth regularization significantly improves the pose estimation accuracy of LU-NeRF – the median relative rotation errors decrease from 11.08° to 4.69° while the mean drops from 11.93° to 8.26°. Even though the maximum relative rotation error is smaller without the depth regularization, the Shonan averaging [5] fails to converge to a reasonable global pose configuration.

2.4. Computational cost

Table 4 presents the computational cost of the proposed framework. We randomly sample 30 mini-scenes and report the average training time for LU-NeRF-1 and LU-NeRF-2. The LU-NeRFs for different mini-scenes are independent.
and thus can be trained in parallel. The training of LU-NeRFs and the global pose refinement with BARF [6] can be significantly accelerated with some recent advances in learnable scene representations (e.g. PlenOctrees [10], InstantNGP [8]).

### 3. Implementation details

#### 3.1. Building connected graphs

Given a distance function \( \text{dist}(\cdot, \cdot) \), image descriptors \( \{f_i\}_N \) and corresponding cameras \( \{C_i\}_N \) where \( i \in \{0, \ldots, N - 1\} \) is the image index and \( N \) is the total number of images, as the first step, we build a minimal spanning tree (MST) using Kruskal’s algorithm. Each node on the MST represents a camera and the weight \( W_{ij} \) on the edge that connects camera \( C_i \) and \( C_j \) is the distance \( \text{dist}(f_i, f_j) \) between the descriptors of image \( I_i \) and \( I_j \). To ensure each mini-scene contains at least \( K \) images (\( K = 5 \) by default), we augment the MST by adding nearest neighbors for nodes that have less than \( K - 1 \) connected nodes in the MST. We also ensure that each edge appears in both mini-scenes centered at the endpoints of the edge, such that there are at least two measurements for the relative pose between two connected cameras. Having multiple measurements allows for estimating the confidence of the predicted relative poses and identifying LU-NeRF failures (see Sec. 3.2).

#### 3.1.1 Distance functions

**Motivation.** We intentionally experiment with simplistic approaches to compute image similarity in our graph-building procedure since our primary contribution in this work is the local-to-global pose and scene estimation starting with LU-NeRF on mini-scenes. In practice, depending on the application context, there are likely different cues or weak supervision that can be exploited for graph building (as we do for ordered sequences). We leave it to future work to explore more sophisticated unsupervised/self-supervised techniques for building neighborhood graphs.

In our experiments, we tried two different features to build the connected graph: self-supervised DINO features [4] and raw RGB values with \( \ell_1 \) distance.

#### DINO features

We extract semantic object parts by applying K-means clustering on the image collections [2]. The number of parts is 10 by default in our experiments. We build an image descriptor in a similar way as the Histogram-of-Gradients (HoG). Specifically, we evenly split the part segmentation maps into \( 4 \times 4 \) grid and compute a part histogram within each cell. We then normalize the histogram per cell into a unit vector and use the concatenated 16 histograms as the image descriptors. The cosine distance between these descriptors is used to build the MST and final graph.

**\( \ell_1 \) on RGB values.** We estimate the distance between two images as the minimum \( \ell_1 \) cross-correlation over a set of rotations and translations. Formally, we compute

\[
\text{dist}(I_1, I_2) = \arg\min_{t, \theta} \sum_x |I_1(x) - I_2(R_\theta x + t)|,
\]

where \( x \) is the pixel index, \( t \) is in a \( 5 \times 5 \) window centered at 0, \( R_\theta \) is an in-plane rotation by \( \theta \) and \( \theta \in \{0, \pi\} \). When the minimum is found at \( \theta \neq 0 \), we augment the input scene with the in-plane rotated image and correct its estimated pose by \( \theta \).

We found that considering the in-plane rotations here is useful because of symmetries – some scenes of symmetric objects can contain similar images from cameras separated by large in-plane rotations. This is problematic for LU-NeRFs because they initialize all poses at identity. Augmenting the scene with the closest in-plane rotations makes the target poses closer to identity and helps convergence.

**Metric selection.** In experiments with unordered image collections, we used the \( \ell_1 \)/RGB metric for Lego and Drums, and the DINO metric for Chair, Hotdog, and Mic. The RGB metric fails to build useful graphs for Hotdog, Mic and Ship, while the DINO metric fails for Lego, Drums, and Ship. No graph-building step is necessary on ordered image collections since the order determines the graph.

#### 3.1.2 Analysis

Figure 2 presents the MST, the connected graph, and image pairs that are connected in the graph on the Chair scene from the NeRF-synthetic dataset when using the DINO features. Surprisingly, the self-supervised ViT generalizes well on unseen objects and the learned representations are robust to viewpoint change (see the last column of Figure 2).

Figure 3 presents an analysis of the connected graphs built with DINO and RGB features. Both features provide outlier-free connected graphs on Chair. The graphs built with DINO contain much fewer outliers on Hotdog and Mic, while RGB features induce clearer graphs on Drums and Lego. Both DINO and RGB features produce more outliers on Ship than other scenes.

**Optimal graph vs noisy graph.** To analyze the effect of graph building on the unordered image collection, we build

---

Table 4. **Computational cost.** We report the mean time for training a single LU-NeRF-1, a single LU-NeRF-2, and the final refinement step on an NVIDIA Tesla P100. The pose synchronization step runs on CPU and has a negligible running time.

<table>
<thead>
<tr>
<th>Stage</th>
<th>Running time</th>
</tr>
</thead>
<tbody>
<tr>
<td>LU-NeRF-1</td>
<td>1.08 hours</td>
</tr>
<tr>
<td>LU-NeRF-2</td>
<td>0.89 hours</td>
</tr>
<tr>
<td>Pose synchronization</td>
<td>3.18 seconds</td>
</tr>
<tr>
<td>Pose refinement</td>
<td>4.40 hours</td>
</tr>
</tbody>
</table>
Minimum spanning tree
Connected Graph
Samples of connected nodes

56.49º  23.18º  22.96º  17.51º  13.15º

Figure 2. Graph built with DINO features on Chair. The minimum spanning tree (left), the connected graph (middle), and samples of connected image pairs (right). In the right panel, each column presents two images that are connected on the graph (1st and 3rd row), the corresponding part co-segmentation maps [2] (2nd and 4th row), and rotation distance between the two views (bottom).

Figure 3. Graph statistics. We compare the rotation distance between mini-scene neighbors on the optimal graph built with groundtruth camera poses, the graph built with DINO features, and the one built with RGB features. For most scenes both DINO and RGB mini-scenes include outlier images (close to 180° distance) which our pipeline needs to deal with.

Table 5. Optimal graphs vs noisy graphs. The outliers in the noisy connected graph built with DINO/RGB features may hurt the performance of our method in camera pose estimation. The clean graph is built from the ground-truth camera poses.

<table>
<thead>
<tr>
<th>Scenes</th>
<th>Rotation Error [°]</th>
<th>Translation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair</td>
<td>Optimal graph</td>
<td>Noisy graph</td>
</tr>
<tr>
<td>4.24</td>
<td>2.64</td>
<td>0.16</td>
</tr>
<tr>
<td>Hotdog</td>
<td>0.23</td>
<td>0.24</td>
</tr>
<tr>
<td>Lego</td>
<td>0.07</td>
<td>0.09</td>
</tr>
<tr>
<td>Mic</td>
<td>0.84</td>
<td>6.68</td>
</tr>
<tr>
<td>Drums</td>
<td>0.05</td>
<td>12.39</td>
</tr>
</tbody>
</table>

3.2. LU-NeRF architecture and training details

In the training of LU-NeRF, we do not apply the coarse-to-fine strategy proposed in BARF [6]; we sample 2048 rays per mini-batch and adopt the learning rate schedule for pose and MLP parameters from BARF; we remove the positional encoding and view-dependency, and use a compact 4-layer MLP to reduce the memory cost and speed up the training. We set the initial camera poses to identity. We have experimented with random initialization around identity but observed no significant difference. We terminate the training of LU-NeRF-1 if the average change of the camera rotations in a mini-scene is less than 0.125° within 5k iterations. We train LU-NeRF-2 for 5k iterations with frozen initial poses and then jointly optimize camera poses and neural fields for 45k iterations. We apply depth regularization on small patches (2 × 2 by default) in both LU-NeRF-1 and LU-NeRF-2.

3.3. Synchronization and refinement details

In the pose synchronization step, we apply the off-the-shelf Shonan averaging\(^2\) [5], which solves a convex relaxation of the problem described in Eqn. (1) of the main text, while iteratively converting it to higher dimensional special-orthogonal spaces SO(n) until it converges. We then optimize the translation with fixed camera rotation.

The input to the Shonan averaging is the relative pose estimations from LU-NeRF and the confidence of these pose estimations. Each pair of cameras may have multiple measures of the relative poses, as each camera may appear in multiple mini-scenes. We apply a simple heuristic to pick one measure from these multiple candidates: given two

\(^2\)https://github.com/dellaert/ShonanAveraging

an oracle outlier-free connected graph with groundtruth camera poses. Table 5 compares the performance of our method with the optimal graphs and noisy graphs built with DINO/RGB features. Outliers in the connected graph may hurt the performance of LU-NeRF. Nevertheless, with our simple graph-building methods based on DINO/RGB features, our method outperforms the baselines when they are not given prior constraints on the camera pose distributions.

We notice that the performance with the optimal graph is worse than that with the noisy graph on Chair. The “optimal” graph minimizes camera distances, but it is not guaranteed to be the best choice for LU-NeRF, e.g., issues like mirror symmetry ambiguity (Sec. ??) can arise more often when cameras are in close proximity, and there is randomness inherent in training neural networks.
Table 6. Camera baselines. We report the average rotation [°] and translation distance between all camera pairs in the sequential data sampled from the Objectron dataset [1].

<table>
<thead>
<tr>
<th></th>
<th>Bike</th>
<th>Chair</th>
<th>Cup</th>
<th>Laptop</th>
<th>Shoe</th>
<th>Book</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rotation</td>
<td>24.41</td>
<td>5.61</td>
<td>13.41</td>
<td>23.13</td>
<td>19.36</td>
<td>45.78</td>
</tr>
<tr>
<td>Translation</td>
<td>4.09</td>
<td>1.20</td>
<td>0.63</td>
<td>1.79</td>
<td>0.82</td>
<td>1.43</td>
</tr>
</tbody>
</table>

To resolve the scale ambiguity across different mini-scenes, we first scale each mini-scene so that the MST edges connecting different mini-scenes are scale-consistent (MST construction is described in Sec. 3.1). Specifically, we establish a reference scale by fixing it in one mini-scene and propagating it to others through the MST. We focus on edges linking mini-scene centers and rescale the mini-scenes so that overlapping edges share a consistent scale.

We then obtain the translation by solving a linear system $t_j - t_i = R_i t_{ij}$ where $R_i$ is the rotation of camera $i$ from the Shonan method and $t_{ij}$ is the relative translation between camera $i$ and $j$ from LU-NeRF. Similar to the relative rotations, each pair of cameras may have multiple measures of the relative translation. We use the same heuristic described above to pick one from the multiple measures. The translation optimization has also been implemented in the off-the-shelf Shonan averaging.

In the global pose refinement stage, we closely follow the default setting of BARF [6] which jointly trains the MLP and refines the poses for 200k iterations with a coarse-to-fine positional encoding strategy.

We utilize the camera visualization toolkit from BARF [6] in our main paper and the supplementary material.

3.4. Dataset release and open sourcing.

We will release the newly ordered Blender sequences and open-source the code on our project website.

For the sequential data sampled from the Objectron dataset [1], Table 6 reports the average rotation and translation distance between all camera pairs as a reference for our quantitative evaluations in Table 6 in the main text.

References


3https://zezhoucheng.github.io/lu-nerf/