LEIA: Latent View-invariant Embeddings for Implicit 3D Articulation Supplementary Material

Archana Swaminathan[1], Anubhav Gupta[1], Kamal Gupta[1], Shishira R Maiya[1], Vatsal Agarwal[1], and Abhinav Shrivastava[1]

University of Maryland, College Park
https://archana1998.github.io

1 Training Details

1.1 LEIA Network Architecture

NeRF Architecture Our NeRF architecture is based on InstantNGP [4] and leverage NerfAcc [2] for ray marching accleration and approximation. We keep base details of our NeRF implementation the same as InstantNGP and PARIS, specifically the hash encoding, spherical harmonics encoding, hash grid levels and layers of the MLPs for geometry and texture. To summarize, the multi-resolution hash encoding represents the point position as a 16-dimensional feature and the viewing direction utilizes spherical harmonics encoding of the order 4. The hash grid is configured with 16 levels, with a maximum of 2^{19} entries per level.

Hypernet Architecture In addition to the NeRF, we use a Hypernet to modulate the NeRF weights. The Hypernet gives gives low rank matrix weights, and then the NeRF weights for both the geometry and texture models are updated. The geometry network has a two-layer fully connected MLP, and the texture network has a one-layer fully connected MLP, both from the tiny-cudann framework. The hypernet in addition uses a multi-layer perceptron with one hidden layer of 64 dimensions, influencing all layers by default. The output generates soft masks of the weights for the target network. The `tanh` nonlinearity is used. These soft masks have a rank of 10 and are normalized to give the final low-rank weights. The latent network used for learning the embedding, learns the latent dictionary of size $N \times 512$, where N is the number of latent vectors learnt, which is equal to the number of states chosen. For all our experiments, we have $N = 4$ states used for training. Throughout the geometry and texture network, ReLU activation units are used.

1.2 Training time and compute

We use a single GPU for each training instance. The training time is about 3 hours on the RTX A5000 and 2 hours on the RTX A6000, for a full training run of 20,0000 iterations, which is until convergence occurs. We train across 4 states, learning a latent embedding representation for each articulation state.

1.3 Losses and Optimization

We use the SmoothL1 Loss for the RGB image which is described in the methods section of the main paper, as well as a foreground mask loss. The mask loss is calculated by taking the binary cross entropy loss between the predicted opacity and the ground truth foreground mask as denoted below:

$$\text{BCELoss } (\hat{y}, y) = -\frac{1}{N}\sum_{i=1}^{N} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i) \tag{1}$$

where $\hat{y}$ is the predicted opacity and y denotes the ground truth foreground mask. In addition to these two main training losses, we use the latent manifold loss and depth and occlusion regularizations, as described in the main paper.

2 Data Curation Details

2.1 Synthetic Data - SAPIEN

We render objects from the PartNet-Mobility Dataset [1,3] using SAPIEN library [5] that it is shipped with. We set the near and far bounds of the camera to be 0.1 and 100 respectively, and and also use ambient and directional and point-based lighting. We render images of high resolution with a size of 800×800 with a camera with the field of view set as 75 degrees. We render images at 20 linearly spaced steps of the range of motion of the articulation (unless otherwise specified). In total, we captured 20 rendered images, encapsulating the range of articulation for each object, across 100 camera views, across 12 objects. From this, we choose **four states** for training, which are in the middle of the set of the 20 rendered images, spaced with a distance of two states.

2.2 Real-World Data Capture

We captured 68 images from a real-world scene of the oven of a toy kitchen, with an iPhone and a tripod, and moved it in a rough semicircular sweep, capturing pictures at 17 different camera positions. The images are post-processed to remove background and then used for training.

3 Extended Ablations

We show quantitative results (see Table 1) of some extended ablations. Different dimensions of latent embedding, number of layers in hypernetwork architecture, and nearest neighbours for manifold loss was tested. We see that overall, LEIA works better with less hypernetwork layers, showing that the modulation of the NeRF is strong with just one layer in the hypernet. Similarly, we see lesser number of nearest neighbors for the manifold loss helps drastically in increasing

Table 1: Ablation Study. We test out different latent embedding size, layers in hypernet and nearest neighbors in our extended ablations. We show that for our final experiments, we go with embedding size 512, only one hypernet layer and 2 neighbors for manifold loss, for better visual quality and numerical accuracy.

Latent Embedding Size	**PSNR**	**SSIM**	**LPIPS**
256	27.48	0.94	0.06
512	**29.59**	**0.96**	**0.05**
1024	28.84	0.94	0.06
Num layers in Hypernet	**PSNR**	**SSIM**	**LPIPS**
3	29.08	0.95	0.06
2	28.60	0.94	0.06
1	**29.55**	**0.96**	**0.05**
Nearest Neighbors for Manifold Loss	**PSNR**	**SSIM**	**LPIPS**
2	**29.40**	**0.95**	**0.05**
3	25.38	0.94	**0.05**

PSNR. This indicates that reducing the distance between just two nearest neighbors of the query state in training helps create a manifold where the latents are spaced apart just enough to establish a linear relationship between them. We also see that LEIA works better with latent embedding dimension size 512, which is an empirically found result. We do not ablate over number of cameras as it is a well-established fact that NeRF models work better with more views. We chose the same number of cameras that the baseline PARIS trains with for fair comparison, which was a 100 cameras.

4 Supplementary Webpage

We show videos and GIFs of gradually interpolated, linearly spaced states in the following link `https://archana1998.github.io/leia/results_index.html`. Interpolation for both PARIS and LEIA was done with 5 equally spaced timesteps in between the start and end state. We show how PARIS fails to predict the intermediate states and motion correctly for multi-part objects (specifically ones where the two parts of the object move differently or have both types of articulation), whereas LEIA can do it correctly for both single-part and multi-part. This empirically substantiates and emphasizes our claim that LEIA is versatile, scalable, and robust to multiple articulations occuring in multi-part objects.

Additional Real World result. We provide one more real-world reconstruction using our method in Fig. 1. We highlight the fact that ours is the first method to handle multiple articulation joints with no 3D supervision, which demonstrates reasonable generalization on real-world data.

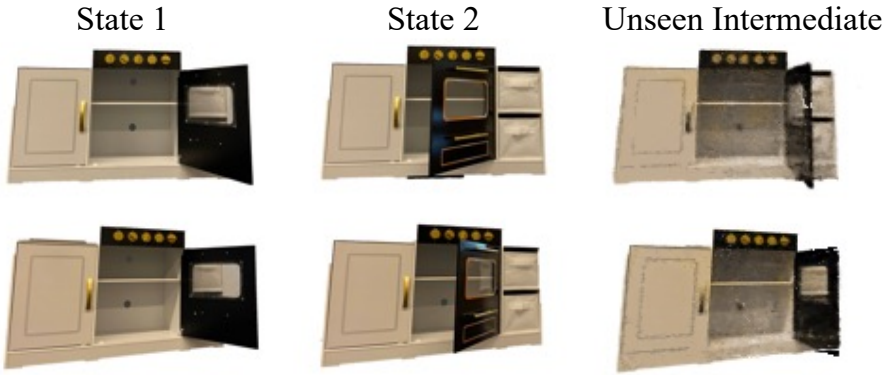

Fig. 1: Additional Real World Result. Shown above are the start, end and unseen intermediate, interpolated state of a toy kitchen object, captured in the real world with a mobile camera.

References

1. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015)
2. Li, R., Gao, H., Tancik, M., Kanazawa, A.: Nerfacc: Efficient sampling accelerates nerfs. arXiv preprint arXiv:2305.04966 (2023)
3. Mo, K., Zhu, S., Chang, A.X., Yi, L., Tripathi, S., Guibas, L.J., Su, H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2019)
4. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Trans. Graph. **41**(4), 102:1–102:15 (Jul 2022). https://doi.org/10.1145/3528223.3530127, https://doi.org/10.1145/3528223.3530127
5. Xiang, F., Qin, Y., Mo, K., Xia, Y., Zhu, H., Liu, F., Liu, M., Jiang, H., Yuan, Y., Wang, H., Yi, L., Chang, A.X., Guibas, L.J., Su, H.: SAPIEN: A simulated part-based interactive environment. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)