Supplementary for Progressive Multi-Scale Light Field Networks

David Li Amitabh Varshney
University of Maryland, College Park

1. Additional Model and Training Hyperparameters

Our full training pipeline is written in PyTorch. As mentioned in the main text, we utilize a multi-layer perceptron with 9 hidden layers and one output layer totalling 10 layers for each of our models. Depending on the target level of detail, we vary the number of neurons used at each hidden layer. This can be 128, 256, 384 or 512. Each hidden layer is followed by a layer normalization (LayerNorm) layer with learned affine parameters providing a learned scale and bias for each node. For our progressive model, along with taking subsets of the weight and bias parameters of the linear layer, we also take the appropriate subsets of the scale and bias parameters for each LayerNorm layer.

In addition to the primary light field networks, we also train an auxiliary network which encodes the occupancy. The auxiliary network is made up of two hidden layers with 16 neurons each and one output layer. It is trained alongside the primary light field network using the same optimizer with an squared L2 loss.

Our models are trained using the Adam optimizer with the learning rate set to 0.001. Additionally, the learning rate is decayed by $\gamma = 0.98$ at the end of each epoch using an exponential learning rate scheduler. All runs are performed with an identical seed for the random number generator. We use the same hyper-parameters across all datasets.

2. Additional Qualitative Results

Qualitative results for all of our datasets comparing single-scale LFNs and progressive multi-scale LFNs are shown on the following page. Rendered video of each static light field from different views are provided on our supplementary webpage.

3. Training Ablation Details

3.1. Training Strategy

To evaluate our training strategy, we perform an ablation against progressive training. For progressive training, we use the same progressive multi-scale LFN model where subsets of coefficients are used to represent different levels of detail. Levels of detail trained sequentially from the lowest level to the highest at their target resolutions. Once an LOD is trained, it's parameters are frozen so they do not get updated during the training of subsequent levels. However, each subsequent LOD still shares parameters from lower levels. We train LODs one, two, three, and four for 2133, 533, 133, and 100 epochs respectively on images downsampled to their target resolution to approximately match the expected number of rays sampled using our training strategy. Additionally, we do not use learning rate scheduling due to the additional epochs. We observe that progressive training can achieve similar PSNR results but has worse SSIM results and takes an average of 24.20 hours to train compared to 17.78 hours for our training strategy.

3.2. Number of Training Views

To determine how the amount of training views affects the resulting render quality, we conduct experiments training our progressive multi-scale light field networks with fewer training views. For each dataset, we load every other training view of the original 216 training views for a total of 108 views. Then the same 16 test split views are used to compute the PSNR and SSIM metrics to evaluate the visual quality. Since light field networks do not have as strong a 3D prior as NeRFs, utilizing a NeRF-based teacher model may be needed for setups with much fewer cameras.

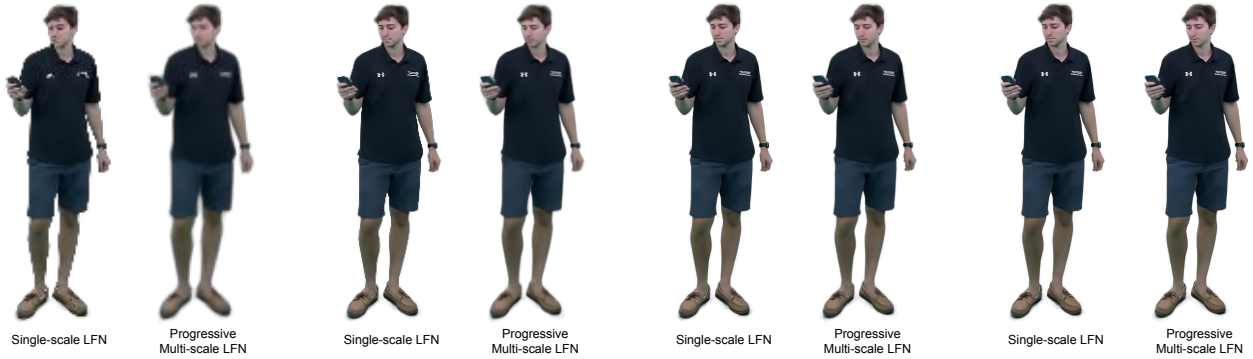
4. Areas for Future Work

4.1. Dynamic Light Field Networks

Our current work does not utilize dynamic light fields, also known as light field videos. One method of achieving dynamic light field networks involves simply inputting a time coordinate to the light field as used in SIGNET [2]. Another potential method could be predicting Fourier coefficients similar to Fourier PlenOctrees [5]. In both cases, we believe our method would be applicable to dynamic light field networks though we leave exploration of these to future work.

4.2. View Synthesis Quality

As noted by Sitzmann *et al*. [4], LFNs have worse multi-view consistency compared to neural radiance fields [3],

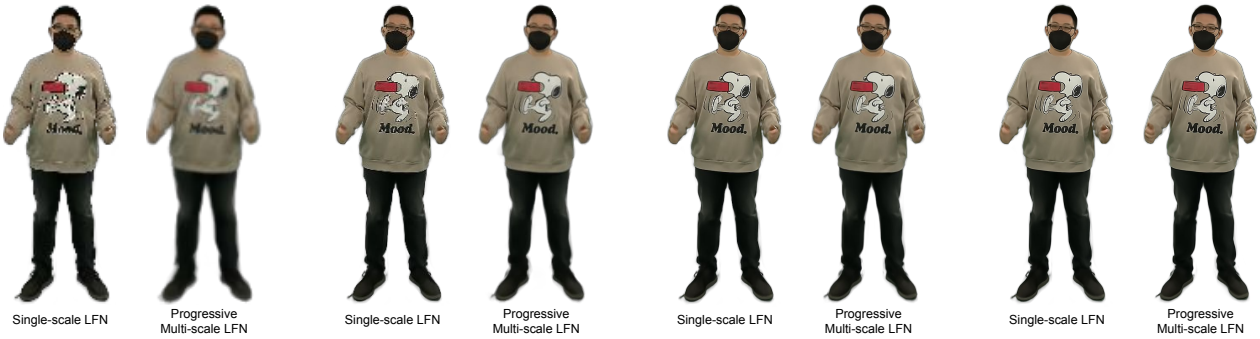


(a) A at 1/8 scale (b) A at 1/4 scale (c) A at 1/2 scale (d) A at full scale

Figure 1: Qualitative results for dataset A.

(a) B at 1/8 scale (b) B at 1/4 scale (c) B at 1/2 scale (d) B at full scale

Figure 2: Qualitative results for dataset B.



(a) C at 1/8 scale (b) C at 1/4 scale (c) C at 1/2 scale (d) C at full scale

Figure 3: Qualitative results for dataset C.

(a) D at 1/8 scale (b) D at 1/4 scale (c) D at 1/2 scale (d) D at full scale

Figure 4: Qualitative results for dataset D.

(a) E at 1/8 scale (b) E at 1/4 scale (c) E at 1/2 scale (d) E at full scale

Figure 5: Qualitative results for dataset E.

hence requiring many more views for training. When adding positional encoding to the Plücker coordinates, LFNs can better render high-frequency detail from training views but largely fail at view synthesis. In our experiments, our light field networks directly input Plücker coordinates into the network without any positional encoding. This is enabled by our dense camera setup featuring 240 camera views. Future work may focus on closing the gap in view synthesis quality and dataset requirements between LFNs and NeRFs.

Table 1: Average Model Rendering Quality

Model	LOD 1	LOD 2	LOD 3	LOD 4
Multi-scale LFN	29.37	29.88	29.01	28.12
Mip-NeRF	23.45	24.32	24.46	24.26

(a) PSNR (dB) at 1/8, 1/4, 1/2, and 1/1 scale.

Model	LOD 1	LOD 2	LOD 3	LOD 4
Multi-scale LFN	0.8834	0.8819	0.8626	0.8570
Mip-NeRF	0.6681	0.6830	0.6704	0.6624

(b) SSIM at 1/8, 1/4, 1/2, and 1/1 scale.

To determine how well our datasets are represented with NeRFs, we compare our method against Mip-NeRF [1] on our 240 camera setup datasets. Following our existing training setup, we use 216 training poses, 12 validation poses, and 12 test poses. We train Mip-NeRF for $1,000,000$ iterations using a batch size of 1024 rays with our dataset reader sampling 67% of rays from the foreground and 33% from the background in each batch. Our quantitative results are shown in Table 1. On our datasets, we observe that Mip-NeRF produces softer edges and artifacts in unoccupied regions which lead to worse PSNR and SSIM scores. Note that further hyper-parameter tuning may improve Mip-NeRF results.

References

[1] Jonathan T. Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P. Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields, 2021. 3

[2] Brandon Yushan Feng and Amitabh Varshney. SIGNET: Efficient neural representation for light fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 14224–14233, October 2021. 1

[3] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1

[4] Vincent Sitzmann, Semon Rezchikov, William T. Freeman, Joshua B. Tenenbaum, and Fredo Durand. Light field networks: Neural scene representations with single-evaluation rendering. In *arXiv*, 2021. 1

[5] Liao Wang, Jiakai Zhang, Xinhang Liu, Fuqiang Zhao, Yanshun Zhang, Yingliang Zhang, Minye Wu, Lan Xu, and Jingyi Yu. Fourier plenoctrees for dynamic radiance field rendering in real-time, 2022. 1