Neural Lens Modeling
Supplementary Material

Wenqi Xian[1,*], Aljaž Božič[2], Noah Snavely[3] and Christoph Lassner[4]
Meta Reality Labs Research[1,2,4]
Cornell University[1,3]
wx97@cornell.edu[1], aljaz@meta.com[2], snavely@cs.cornell.edu[3], classner@meta.com[4]

A. Overview

This supplementary document provides implementation and training details in Section B and additional evaluation results on the SynLens dataset in Section C.

B. Implementation and Training Details

	FC layer width				
FC layer depth	128	256	512	1024	2048
4	10.82	9.12	1.18	**0.07**	0.07
10	9.43	9.18	0.92	0.09	0.10
15	9.21	9.23	0.45	0.12	0.13

Table 1. **Reprojection error (RMS) on SynLens with different model size.**

Optimize NeuroLens. We performed a hyperparameter search on the depth and width of the lens model and present the results in Tab. 1. We found that fully connected layers with a width of 1024 and depth of 4; we had instead consistently used a depth of 4 for all experiments which produces the best results. Specifically, we built an invertible ResNet with two residual blocks, each with two levels of internal layers. In all experiments on the SynLens dataset, we set the learning rate to 1e-4 and the maximum number of epochs to 50 (each epoch iterates through all frames in the captured sequence). The average runtime of the optimization process is 5 minutes on one NVIDIA V100 GPU machine. We activate the photometric loss at the 15th epoch and and fix the weight ratio between the geometric and photometric loss to 10:1.

Optimizing the Keypoint Marker Design and Detection. We initiate the marker itself as a $21 \times 21 \times 3$ image with random uniform initialization that is being processed by a Sigmoid function to remain within range during optimization. The detector model is a MobileNet-v3 [1] followed by two fully-connected layers 1000×386, ReLU and 386×3. The MobileNet-v3 is initialized as pretrained from the PyTorch model zoo. The first two of the three outputs represent a [0;1] normalized mean for the Gaussian distribution of the marker center, the third one represent the standard deviation. We optimize the negative log-likelihood of the data under the predicted distributions; the predicted standard deviation can be used for filtering points for which the model is too uncertain, for all experiments in the paper we use an empirically selected threshold of 0.2.

We use three learning rates for the optimization: 0.00005677 for the MobileNet-v3 backend, 0.4034 for the fully connected layers, and 67.18 for the marker values themselves. We use a batch size of 50 randomly sampled transformations applied to the marker. To train a marker/detector for general purpose RGB camera calibration we use the transformations of:

1. motion blur with probability 0.2 and blur kernel size 3;
2. affine transformation with rotation up to 180 deg. (pos. and neg.), translation of up to 5% (pos. and neg.), scale of up to 5% (up and down), shearing of up to 2 deg. (in both directions);
3. sensor noise in the form of random Gaussian noise with mean 0.0 and std 0.1

implemented using the Kornia [4] library. The transformations can be adapted to the capture scenario and sensor at hand. The hyperparameters for the optimization have been optimized using Ray Tune [3] and the async hyperband scheduler [2]. The marker print size, resolution and density has to be sensibly chosen depending on the printed board size, camera focal length and intended recording distance.

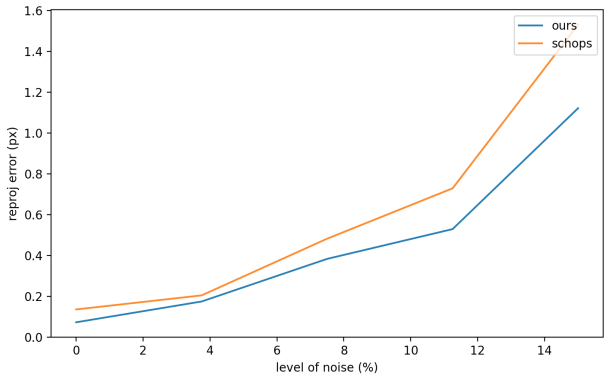

Figure 1. Reprojection error (RMS) on the SynLens dataset by baseline method and our method with noisy keypoints.

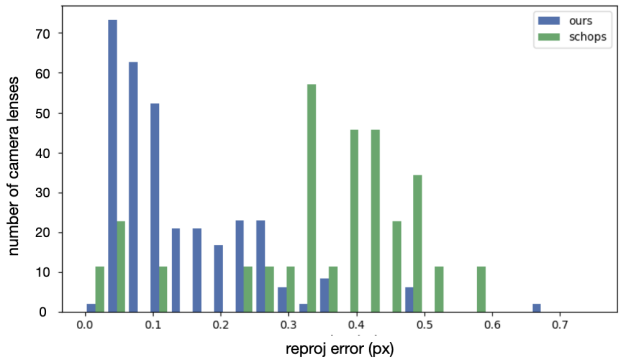

Figure 2. Reprojection error (RMS) histogram on the SynLens dataset by baseline method and our method with ground-truth keypoints.

C. Additional Evaluations on the SynLens Dataset

In Fig. 1, we show how different levels of artificially added noise in keypoints affects the calibration performance of the different methods. We added Gaussian noise with zero mean to the center location of each uniformly sampled keypoint, and the standard deviation of the added noise varies from 1% to 15% of the average distance between each keypoint. The error of both methods increases with the level of added noise. However, our method has a slower increase in reprojection error, meaning it is more robust to noisy keypoints. In Fig. 2, we show a detailed comparison with Schöps et al. [5] on the result distribution of the SynLens dataset.

References

[1] Andrew Howard, Mark Sandler, Bo Chen, Weijun Wang, Liang-Chieh Chen, Mingxing Tan, Grace Chu, Vijay Vasudevan, Yukun Zhu, Ruoming Pang, Hartwig Adam, and Quoc Le. Searching for mobilenetv3. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1314–1324, 2019. 1

[2] Liam Li, Kevin Jamieson, Afshin Rostamizadeh, Ekaterina Gonina, Jonathan Ben-tzur, Moritz Hardt, Benjamin Recht, and Ameet Talwalkar. A system for massively parallel hyperparameter tuning. In I. Dhillon, D. Papailiopoulos, and V. Sze, editors, *Proceedings of Machine Learning and Systems*, volume 2, pages 230–246, 2020. 1

[3] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 1

[4] E. Riba, D. Mishkin, D. Ponsa, E. Rublee, and G. Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. 1

[5] Thomas Schöps, Viktor Larsson, Marc Pollefeys, and Torsten Sattler. Why having 10,000 parameters in your camera model is better than twelve. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2535–2544, 2020. 2