

Appendix

Augmenting with NeRFs: Fast Relocalization on Densified Datasets

Paper ID: 1129

A. Additional Implementation Details

Our implementations of model architectures varied minimally from the chosen off-the-shelf backbones with the exception of our attempted reimplementation of CoordiNet [10] for which source code is unavailable. No depth priors are used in any experiments, nor do we compare against RGB-D relocalizers.

A.1. UniRepLKNet

For UniRepLKNet [6] we use the A variant in all experiments unless otherwise noted. We append a regression head with which we replace UniRepLKNet’s ”head” module, pulling features into three sequential 1024-dimensional linear layers with ReLU activation functions, and finally a 7-dimensional output layer to predict 6-DOF pose (parametrized as XYZ WXYZ). The output layer’s quaternion values are normalized before final output. The number of features entering the custom head module varies with the size of the images used due to adaptive pooling - for 224x224 images as in all experiments unless otherwise noted, the linear layer accepts 320 feature dimensions. UniRepLKNet is loaded with ImageNet-1k pretrained weights when possible.

A.2. ResNet

We implement ResNet [9] by loading the ResNet-152 model with pretrained weights as is available from PyTorch [13]. We remove the final classification layer from ResNet-152, and attach in sequence: 1x1 adaptive pooling, flatten, and two 1024-dimensional linear layers with ReLU activations and 20% dropout, followed by a 512-dimensional linear layer, one more ReLU, and the 7-dimensional output layer for which the quaternion outputs are once again normalized.

A.3. CoordiNet

Our implementation of CoordiNet is a PyTorch-based recreation using a pre-trained EfficientNet-B3 [14] from the timm library as a feature encoder, as prescribed in [11]. Our implementation demonstrably falls well short of the success published in the CoordiNet paper, and serves only as a demonstration of a more complex alternative architecture functioning well within our larger pipeline. For rigorous benchmarks, please see our synthetic section which highlights direct comparisons to works with published code (particularly DFNet [5], DSAC* [3]) on our faraway-relocalization (FR) scenes designed to illustrate shortcomings of competing approaches.

B. Offline Performance

Our paper largely omits mention of the offline performance of our pipeline due to our emphasis on practical deployment to lightweight, portable robotics. However, our offline performance, while uncompetitive with approaches focused on optimizing for map building time (such as AceZero [4]), remains very tractable. We provide the following details on offline performance to aid reproducibility.

B.1. NeRF Fitting

Training Instant-NGP [12] is constrained primarily by the VRAM cost of loading the training dataset, rather than the available compute. That is, the thousands of high resolution images available in the Cambridge and 7Scenes dataset were prohibitive to work with on a local machine, as Instant-NGP is designed to load the entire training dataset simultaneously. If not for this constraint, it would be very feasible to train on local hardware, but the 16GB of VRAM in our 4080 was not enough for the larger of the 7Scenes datasets. We therefore train Instant-NGP models on single A6000s for each scene. These generally fit each scene to full convergence in under 20 minutes of time, with most scenes converging in under a minute. The resulting NeRF model checkpoints varied from 20MB to 50MB.

B.2. NeRF Sampling

Our model, unlike some prior works such as [15], does not require smooth depth gradients or smooth images (i.e. it is tolerant to single-pixel noisiness). This means it is free to benefit from advancements in NeRF rendering driven by hash encoding, which has become ubiquitous among NeRF models since the introduction of Instant-NGP (INGP) [12]. Regardless, we chose INGP for its accessibility and simplicity, but faster models should further accelerate this stage.

The sampling of additional images from each scene took varying amounts of time depending on resolution used and the quantity of images sampled. We performed this both locally and on a cluster of 8x A6000s depending on the demands of the scene. On the A6000s, we can render as fast as 25 FPS at 1 sample per pixel (SPP), or 18FPS at 5 SPP. After discovering that much render time was consumed by GPU Scheduling / Dead Time / CPU Interop, we began running multiple processes on the same GPU, bringing total FPS much higher. At 2x Instant-NGPs rendering per GPU, we achieve a net FPS of closer to 40, and with 3x instances on a single A6000 we achieve 50 FPS. 4 instances yields 55 FPS, and 5 instances 61 FPS.

Rendering 100k frames with 5 instances on a single A6000 at 1SPP and a resolution of 256^2 px consumes roughly 30 minutes. At 224^2 px this takes closer to 20 minutes. A 4080 has moderately higher clock speeds and memory bandwidth, exceeding this performance, but succumbs to memory pressure with fewer models simultaneously loaded, leading to comparable performance.

B.3. Pose Regressor Training

Training our various regression models also scaled with the size of the training dataset and the image resolution, but generally took between 1h to 3h on a single A6000. Notably, we omit information about DFNet’s training speed relative to our own in our comparison table. This is because DFNet performs its novel view synthesis in a periodic manner every 20 epochs (by default), whereas it’s a separate process in our pipeline, complicating timing. Regardless, this omission is to DFNet’s favor, which if not for early stopping due to lack of convergence on the challenging synthetic scenes, would train for over 8 hours on a single A6000, compared to our 2.5h even when combined with the 5m-30m view-synthesis stage.

C. Dataset Details

Below we describe training and dataset details for traditional and synthetic scenes.

C.1. Traditional Scenes

The number of training images ranges from 200 (Cambridge - Shop Facade) to 7000 (7Scenes - RedKitchen). Neither dataset provides camera calibration parameters, such as focal length, FOV, or distortion.

The Cambridge Landmarks dataset reliably failed to produce a convergent NeRF across most scenes, due to flawed ground-truth reconstructions and several isolated problematic poses. These included black silhouettes of structures, an image of a thumb (GreatCourt), and cameras posed facing backwards or underground. This motivated a re-run of SfM on the Cambridge dataset, and the omission of a few images that could not be localized. To preserve the integrity of our results, we aligned our uncovered camera positions from the SfM reruns to the original data, using a combination of singular value decomposition and multiscale ICP refinement to achieve the correct rotation and scale of our pose point cloud. As a final step, we minimized the average between original and our SfM-rerun bounding box dimensions, to find a suitable scale factor, ensuring our errors are not globally scaled relative to competing results.

We believe that prior works mitigated the effects of these outliers by reporting median rather than mean accuracy. Furthermore, works like DFNet refrain from training on the entire dataset, picking specific sequences from each scene instead. Finally, some scenes of Cambridge Landmarks are so problematic that they are broadly ignored in benchmarks (Street, Great Court). In our case, these flawed poses were a crucial obstacle to constructing our NeRF, and since reliable ground-truth poses fall outside the scope of this work, we found it acceptable to correct them. This is not uncommon practice - 7Scenes, for example, has notably received corrections used in related works as [1, 2, 5, 15]. Furthermore, as we too report median errors, these minimal omissions (23 total frames representing less than 0.4% of data) of erroneous ground-truth poses in Cambridge-StMarysChurch (10 train-set omissions), Cambridge-GreatCourt (5 train-set omissions), and Cambridge-OldHospital (8 train-set omissions) do not meaningfully impact reported results.

Regarding resampling, for the 7Scenes [7] dataset, we use radii/orientation perturbations of 20cm/10°, 40cm/20°, 60cm/30°, and 80cm/40°. For Cambridge Landmarks, the sampling radius is increased proportionately to the scene domain to 75cm, 150cm, 225cm, and 300cm (orientation remains unchanged).

C.2. Synthetic Scenes

The "Hangar" scene depicts a large (34m x 160m x 110m) rectangular scene exhibiting perfect symmetry on the XZ plane with the exception of a few colored primitives placed around the scene (to give a reasonable model a chance).

- **Training Data** consists of 5000 views at 1280x720 resolution, captured from a human eye-level perspective, moving around the periphery of the hangar, with the camera rotating randomly about the up (Z) axis. A simple undistorted pinhole camera model with a 120°FOV is used.
- **Test Data** simulates a plausible FPV drone flight path through the center of the hangar, traveling in a broad loop. The test images are 512x512 resolution, with identical intrinsics to the train set. On average, each test view is 25m away from its nearest train view.
- **Implementation and Ablation** Our best-performing trajectory was achieved with a train/test resolution of only 224², using UniRepLKNet-N. Similar performance was observed with the smaller UniRepLKNet-P and the larger UniRepLKNet-B pose regression backbone variants. A unique feature of the environment is the rectangular domain, allowing us to experiment with a uniform volumetric sampling of camera poses. This lets us more easily ablate on density, as we show in the paper.

The Dronerace dataset shows a smaller warehouse environment (21m x 45m x 17m) designed to model demands of autonomous drone racing, where high framerates and significant rotations are present.

- **Training Data** consists of only a very sparse 200 ground-truth images of high resolution (1024²). These are captured through the same high-FOV pinhole camera model as before (120 °), and are placed at human-accessible eye-level position around the notably tall scene.
- **Testing Data** is a sequence of 3000 256² images showing a drone race flight path with realistic physics. This trajectory has an average distance to nearest training pose of 5m, and 46°, emphasizing the diverse rotations a drone finds itself in relative to the upright human photography-like training images of the scene / race course.
- **Evaluation** involves first training our NeRF to a surprisingly high PSNR of 22.8 for the limited ground-truth poses. We then generate 100,000 novel views of which over 90% are discarded per our filtering method as illustrated in our synthetic scenes figure. This naturally preserves poses within the scene domain for training. We use UniRepLKNet-A, more in line with our traditional experiments, with a 224² resolution. For IMU fusion, we derive baseline accelerations by finite difference of the drone flightpath test data. We then apply noise with characteristics matching that described in [8] to authentically demonstrate a common drone IMU. We then tune a simple KF to the model, tweaking hyperparameters in order to recover an ideal tuning as one might with prior knowledge of IMU or model behavior.
- **DFNet comparison** required some minor modifications to the embedding dimensionality of DFNet in order to account for errors arising from differently-sized train/test imagery. DFNet converged to a respectable 22.26 PSNR NeRF after 601 iterations, per the default KingsCollege configuration provided in the code. We are careful to do performance and accuracy measurements in a comparable fashion to our own model, with a 1-sample batch size. Notably, map-building was exceedingly slow, and pose-regressor training would have lasted approximately 8 hours on a single A6000 if not for the built-in early stopping (set at the default 20 iterations of nondrecreasing validation loss) due to the model's failure to converge.

References

- [1] Eric Brachmann, Tommaso Cavallari, and Victor Adrian Prisacariu. Accelerated coordinate encoding: Learning to relocalize in minutes using rgb and poses. In *CVPR*, 2023. 2
- [2] Eric Brachmann, Martin Humenberger, Carsten Rother, and Torsten Sattler. On the limits of pseudo ground truth in visual camera re-localisation, 2021. 2
- [3] Eric Brachmann and Carsten Rother. Visual camera re-localization from RGB and RGB-D images using DSAC. *TPAMI*, 2021. 1
- [4] Eric Brachmann, Jamie Wynn, Shuai Chen, Tommaso Cavallari, Áron Monszpart, Daniyar Turmukhambetov, and Victor Adrian Prisacariu. Scene coordinate reconstruction: Posing of image collections via incremental learning of a relocalizer. In *ECCV*, 2024. 1
- [5] Shuai Chen, Xinghui Li, Zirui Wang, and Victor Prisacariu. Dfnet: Enhance absolute pose regression with direct feature matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022. 1, 2

- [6] Xiaohan Ding, Yiyuan Zhang, Yixiao Ge, Sijie Zhao, Lin Song, Xiangyu Yue, and Ying Shan. Unireplknet: A universal perception large-kernel convnet for audio, video, point cloud, time-series and image recognition, 2024. [1](#)
- [7] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 173–179, 2013. [2](#)
- [8] Rodrigo Gonzalez and Paolo Dabov. Performance Assessment of an Ultra Low-Cost Inertial Measurement Unit for Ground Vehicle Navigation. *Sensors (Basel)*, 19(18):3865, Sep 2019. [3](#)
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. [1](#)
- [10] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. Coordinet: uncertainty-aware pose regressor for reliable vehicle localization, 2021. [1](#)
- [11] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. LENS: localization enhanced by nerf synthesis. *CoRR*, abs/2110.06558, 2021. [1](#)
- [12] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Trans. Graph.*, 41(4):102:1–102:15, July 2022. [1](#), [2](#)
- [13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. [1](#)
- [14] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020. [1](#)
- [15] Qunjie Zhou, Maxim Maximov, Or Litany, and Laura Leal-Taixé. The nerfect match: Exploring nerf features for visual localization, 2024. [2](#)