

## Supplementary Material

Yuxin Huang<sup>1</sup> Andong Yang<sup>2</sup> Yuantao Chen<sup>1,3</sup> Runyi Yang<sup>1,4</sup>  
 Zhenxin Zhu<sup>1,5</sup> Chao Hou<sup>1,6</sup> Hao Zhao<sup>1,†</sup> Guyue Zhou<sup>1</sup>

<sup>1</sup>AIR, Tsinghua University

<sup>2</sup>University of Chinese Academy of Sciences <sup>3</sup>Chinese University of Hong Kong (Shenzhen)

<sup>4</sup>Imperial College London <sup>5</sup>Beihang University <sup>6</sup>The University of Hong Kong

572142867olive@gmail.com, zhaohao@air.tsinghua.edu.cn

### A. More on the Network Architectures of the Time-Pose Function

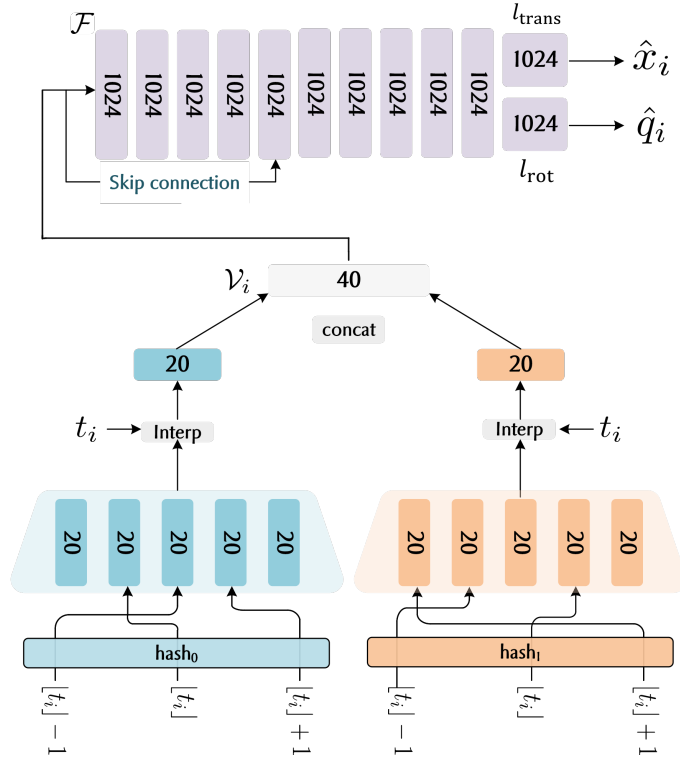


Figure 1. **Architecture.** Our implementation of the Time-Pose Function with a multi-resolution hash grid.

**Multi-resolution Hash Grid** This part additionally introduces the details of our implementation of the time-pose function. For the queried time-stamp  $t_i$ , the hash encoding of  $\lfloor x_i \rfloor - 1$ ,  $\lfloor x_i \rfloor$ , and  $\lfloor x_i \rfloor + 1$  are extracted in each layer  $\mathcal{G}^{(l)}$  of the multi-resolution hash grid. We perform quadratic interpolation on the extracted hash feature:

$$\begin{aligned} \mathcal{V}_i^l = & \frac{1}{2}(t_i - \lfloor x_i \rfloor)(t_i - \lfloor x_i \rfloor - 1)\mathcal{G}^{(l)}(\lfloor x_i \rfloor - 1) \\ & - (t_i - \lfloor x_i \rfloor + 1)(t_i - \lfloor x_i \rfloor - 1)\mathcal{G}^{(l)}(\lfloor x_i \rfloor) \\ & + \frac{1}{2}(t_i - \lfloor x_i \rfloor + 1)(t_i - \lfloor x_i \rfloor)\mathcal{G}^{(l)}(\lfloor x_i \rfloor + 1). \end{aligned} \quad (1)$$

The interpolated feature are then concatenated together  $\mathcal{V}_i = \text{concat}\{\mathcal{V}_i^l\}_{l=1}^L \cdot (1)$ .

**MLP-based** This part introduces the details of the MLP-based implementation of the time-pose function. We use a 10-layer MLP with 1024 dimensions in each layer, including a skip connection that concatenates the timestamp input to the 5th layer. The input timestamp is encoded by the MLP to obtain feature encoding  $\mathcal{V}_i$ . Two fully connected layers are placed after the shared MLP to decode the feature vector  $\mathcal{V}_i$  to the camera pose. (Fig. 2).

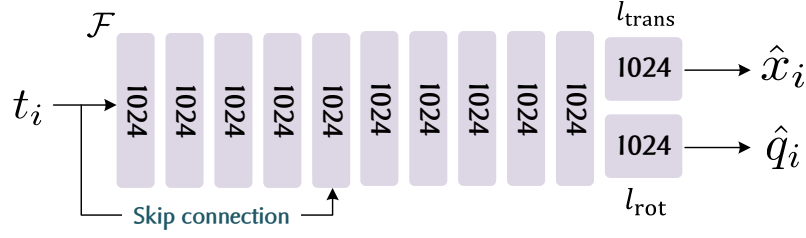


Figure 2. **Architecture.** The MLP-based implementation of the Time-Pose Function.

**1-D Feature Grid** The 1-D feature grid version of the implementation has the same feature grid representation and interpolation functions as our multi-resolution hash feature grid, except for the hash encoding and multiple grid layers. (Fig. 3).

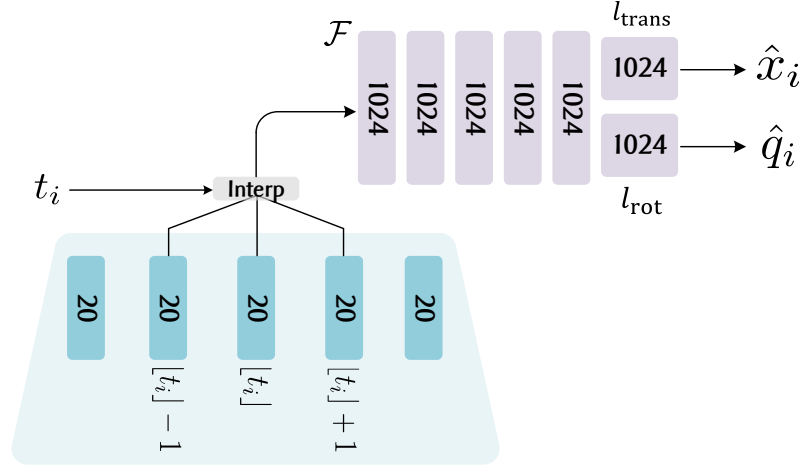


Figure 3. **Architecture.** The 1-D feature grid-based implementation of the Time-Pose Function.

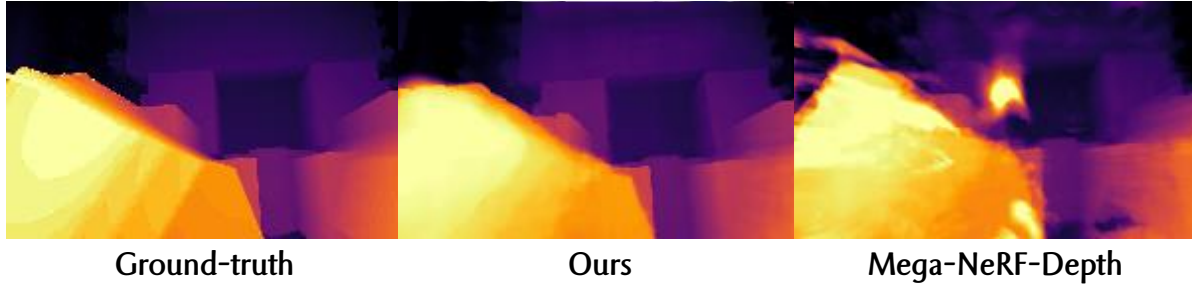


Figure 4. Qualitative results of the comparison of ours with Mega-NeRF-Depth.



Figure 5. **Data Example.** RGB images and the corresponding timestamps in each AUS trajectory.

## B. Comparison with Mega-NeRF-Depth

The estimated depth camera pose from the time-pose function is acceptable, but not sufficient for direct use to supervise scene geometry. We compared our method against Mega-NeRF [5] with depth supervision that takes the time-pose function output as depth camera poses. The qualitative results (Fig. 4) and the quantitative results (Table. 3 in the main paper) show that the direct use of outputted camera pose may help increase the performance of Mega-NeRF in depth prediction. However, the depth prediction results obtained in this way have significant artifacts at the edges of the scene, and in addition, Mega-NeRF-Depth is even inferior to the baseline method that uses only RGB as supervision in rendering RGB images.

## C. The AUS Dataset

**Capture Settings** Our data were collected using a simulator [3] on several city scene models [1, 4] that have been built. We took RGB pictures and depth maps of the scene from an overhead view by crossing the scene with a virtual drone following a given path.

We showed some of the captured images along with their timestamps in Fig. 5.

Since the output of our simulator is a dense RGB image and depth map, we randomly down-sample the dense depth map to a sparser depth map during training to better simulate the real drone shooting data, and in the evaluation process, we use the full depth map to calculate the evaluation metrics for model depth prediction.

**Data Scale** Table. 1 presents the number of images across all scenes. The image resolution is 960x480 pixels and the trajectories are presented in Fig. 6 and Fig. 7.

**More results** Due to the large size of the dataset, we have selected a few representative samples to showcase our results in the paper. Full results in all scenes are presented in Fig. 8.

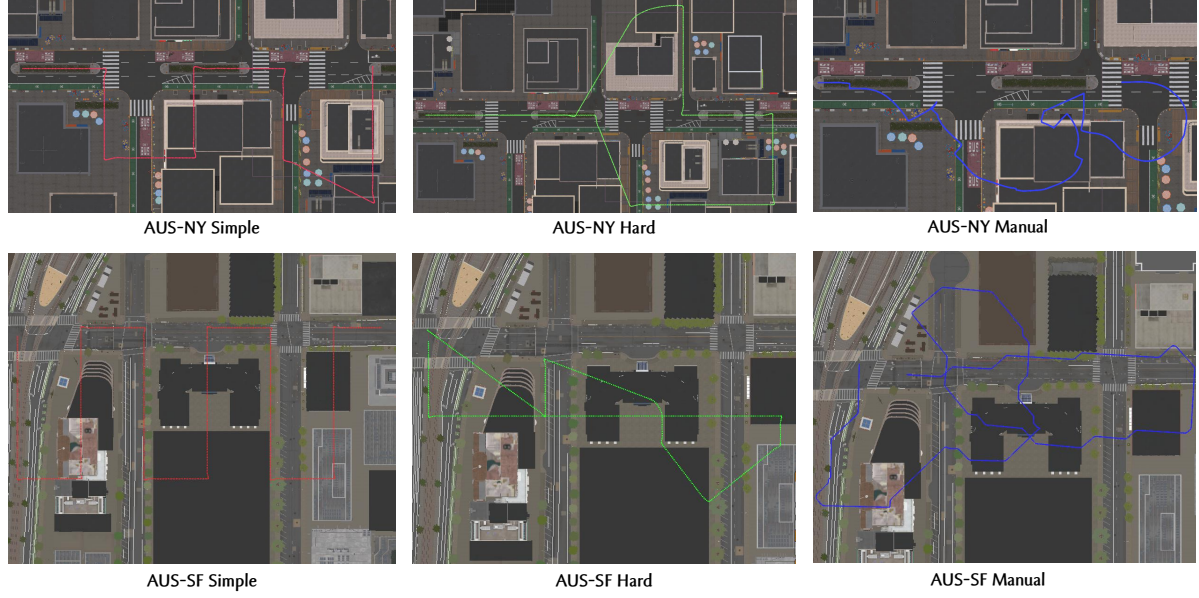


Figure 6. A top-view of the trajectories covered in AUS-NY and AUS-SF.



Figure 7. A top-view of the trajectories covered in AUS-Virtual scenes.

Table 1. Data scale of each scene in AUS Dataset.

Scene	Train set	Validation set
AUS-NY Simple	45	12
AUS-NY Hard	73	11
AUS-NY Manual	100	11
AUS-SF Simple	151	11
AUS-SF Hard	101	11
AUS-SF Manual	262	11
Bridge	213	11
Town	108	11
School	156	11
Castle	62	11

## D. Real-world Dataset

Fig. 9 shows four sets of pictures collected from the real-world dataset using a DJI M300 UAV, equipped with a high-definition RGB camera and LiDAR. The RGB camera captures images at a frame rate of 30 fps, while the LiDAR collects depth information at 240 Hz. In this community scene, the UAV flies in a zigzag pattern to gather data, with a maximum flight



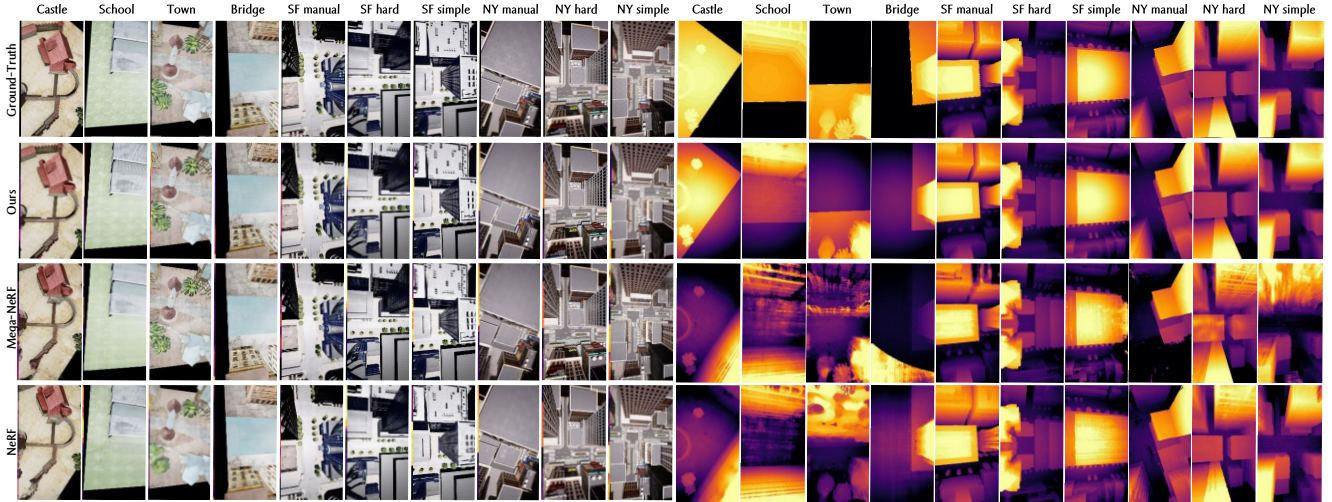


Figure 8. Qualitative Results of all scenes in AUS Dataset.

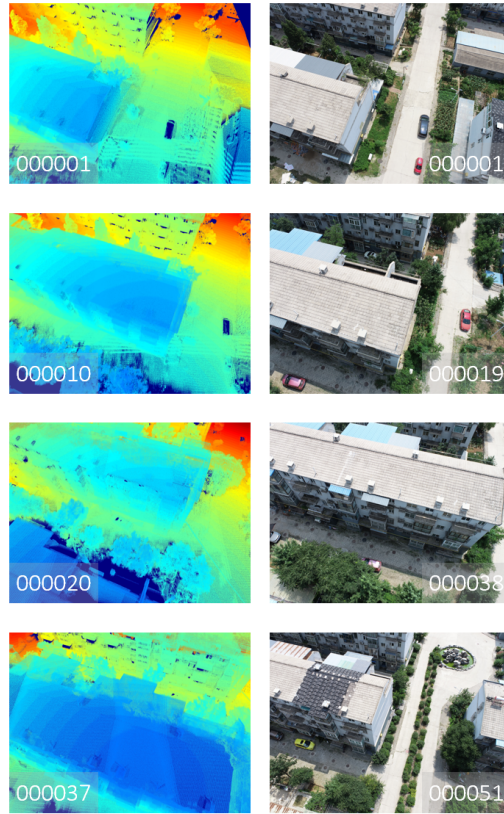


Figure 9. Real-world dataset.

speed of approximately 23 m/s.

## E. Limitations

Despite the promising results of the time-pose function for localization and the realistic rendered images, our system still has limitations:

- The time-pose function learns only from the time-pose trajectory prior, failing to leverage the imagery features. Combining the time-pose function with traditional pose regressors may be a good future work;
- Our scene representation has similar limitations to those of the original NeRF like pose accuracy sensitive and slow converging. However, recent advances in improving NeRF are easy to transfer to our model as the model structure in our scene representation is similar to the original NeRF [2].

## **F. Acknowledgement**

We thank Kirill Sibriakov for the realistic city scene model of New York and San Francisco [4], and the Visual Computing Research Center at Shenzhen University for their open accessed Urbanscene3D dataset [1].

## References

- [1] Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. Capturing, reconstructing, and simulating: the urbanscene3d dataset. In *ECCV*, 2022.
- [2] 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 Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pages 405–421, Cham, 2020. Springer International Publishing.
- [3] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. In *Field and Service Robotics, Results of the 11th International Conference, {FSR} 2017, Zurich, Switzerland, 12-15 September 2017*, volume 5 of *Springer Proceedings in Advanced Robotics*, pages 621–635. Springer, 2017. titleTranslation:.
- [4] Kirill Sibiriakov. Artstation page <https://www.artstation.com/vegaart>, 2022.
- [5] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable Construction of Large-Scale NeRFs for Virtual Fly- Throughs. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12912–12921, June 2022. ISSN: 2575-7075.