
Supplementary Material

Anonymous Author(s)
Affiliation
Address
email

1 Our code is publicly available at Anonymous GitHub.

2 A Time-Pose Function

3 **Qualitative Results** As shown in figure S1, the RGB camera and depth camera covers almost the
4 same trajectory in a single flight, the underlying trajectory prior can be learned implicitly in the RGB
sequence and can be directly transferred into predicting depth camera poses.

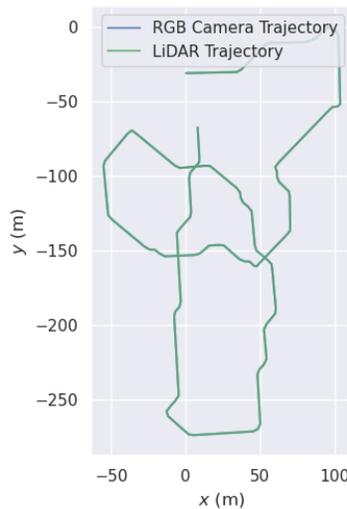


Figure S1: A visual demonstration on the trajectory alignment between RGB and depth sequence.

5

6 To better demonstrate our model performances, we qualitatively show the differences between our
7 estimated trajectories and the ground-truth trajectories in figure S2.

8 B More on the Network Architectures of the Time-Pose Function

9 **Multi-resolution Hash Grid** This part additionally introduces the details of our implementation
10 of the time-pose function. For the queried time-stamp t_i , the hash encoding of $[x_i] - 1$, $[x_i]$, and
11 $[x_i] + 1$ are extracted in each layer \mathcal{G}_θ^l of the multi-resolution hash grid. We perform quadratic

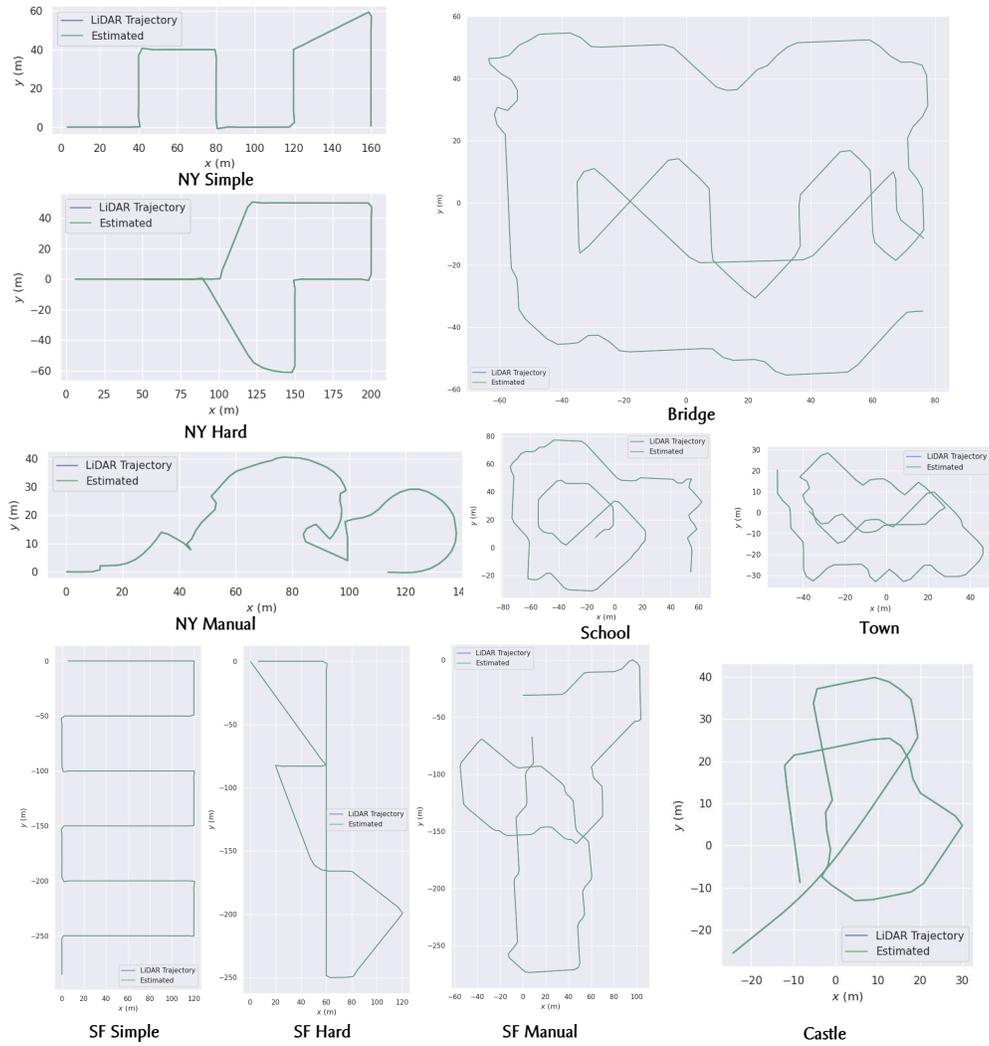


Figure S2: Qualitative result of the Time-Pose Function on the AUS datasets.

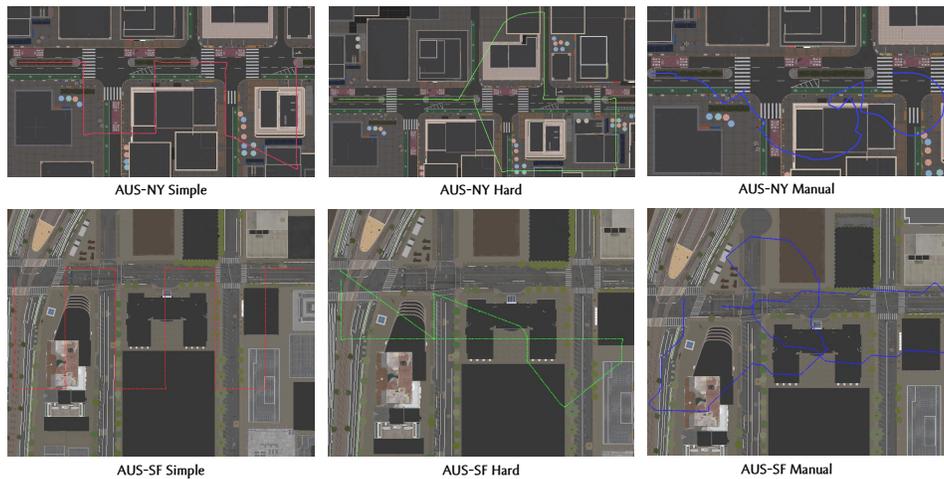


Figure S3: A top-view of the trajectories covered in AUS-NY and AUS-SF.



Figure S4: A top-view of the trajectories covered in AUS-Virtual scenes.

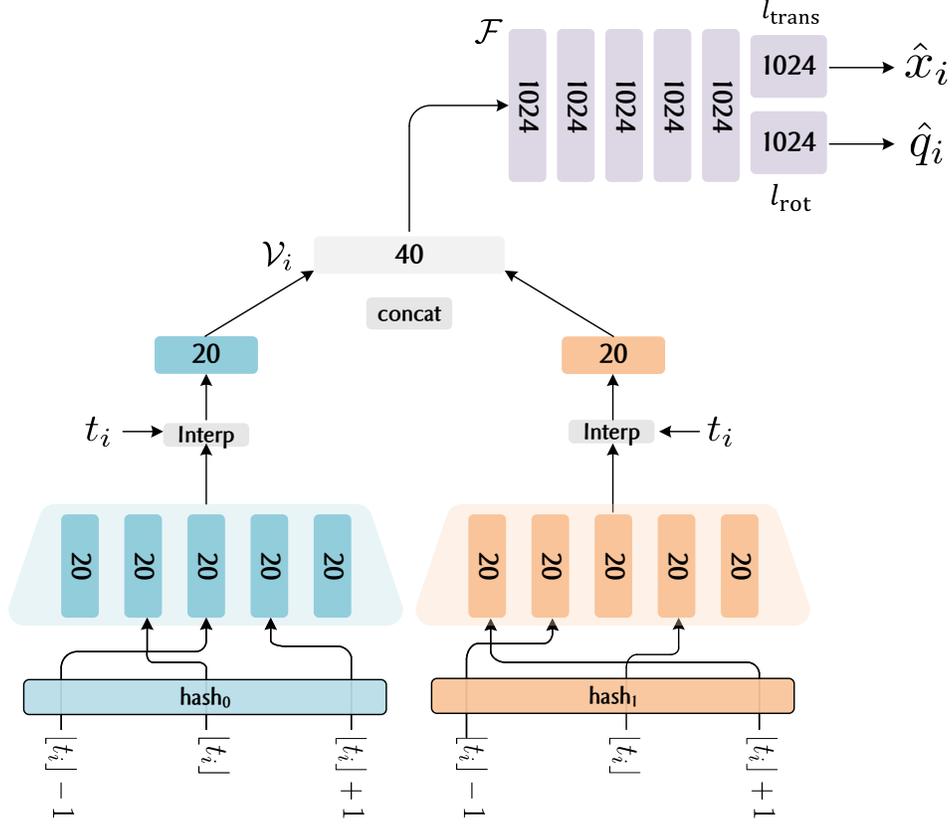


Figure S5: **Architecture.** Our implementation of the Time-Pose Function with a multi-resolution hash grid.

12 interpolation on the extracted hash feature:

$$\begin{aligned}
 f_i^l = & \frac{1}{2}(t_i - \lfloor x_i \rfloor)(t_i - \lfloor x_i \rfloor - 1)\mathcal{G}_\theta^l(\lfloor x_i \rfloor - 1) \\
 & - (t_i - \lfloor x_i \rfloor + 1)(t_i - \lfloor x_i \rfloor - 1)\mathcal{G}_\theta^l(\lfloor x_i \rfloor) \\
 & + \frac{1}{2}(t_i - \lfloor x_i \rfloor + 1)(t_i - \lfloor x_i \rfloor)\mathcal{G}_\theta^l(\lfloor x_i \rfloor + 1).
 \end{aligned} \tag{S1}$$

13 The interpolated feature are then concatenated together: $f_i = \text{concat}\{f_i^l\}_{l=1}^L$. The concatenated
 14 feature vector is then processed through the MLP to calculate the output pose vector \hat{x}_i and \hat{q}_i . (Figure
 15 S5).

16 **MLP-based** This part introduces the details of the MLP-based implementation of the time-pose
 17 function. We use a 10-layer MLP with 1024 dimensions in each layer, including a skip connection
 18 that concatenates the timestamp input to the 5th layer. The input timestamp is encoded by the MLP
 19 to obtain feature encoding \mathcal{V}_i . Two fully connected layers are placed after the shared MLP to decode
 20 the feature vector \mathcal{V}_i to the camera pose. (Figure S6).

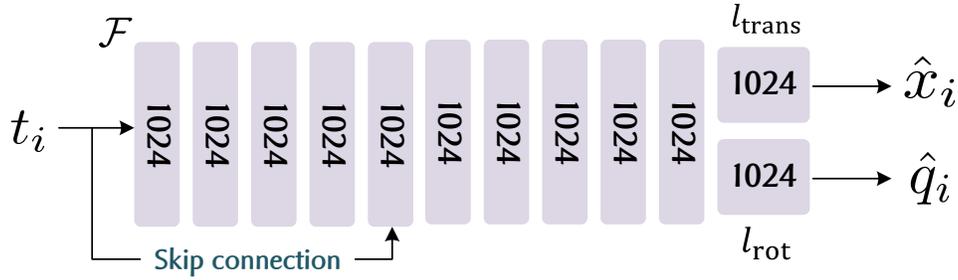


Figure S6: **Architecture.** The MLP-based implementation of the Time-Pose Function.

21 **1-D Feature Grid** The 1-D feature grid version of the implementation has the same feature grid
 22 representation and interpolation functions as our multi-resolution hash feature grid, except for the
 23 hash encoding and multiple grid layers. (Figure S7).

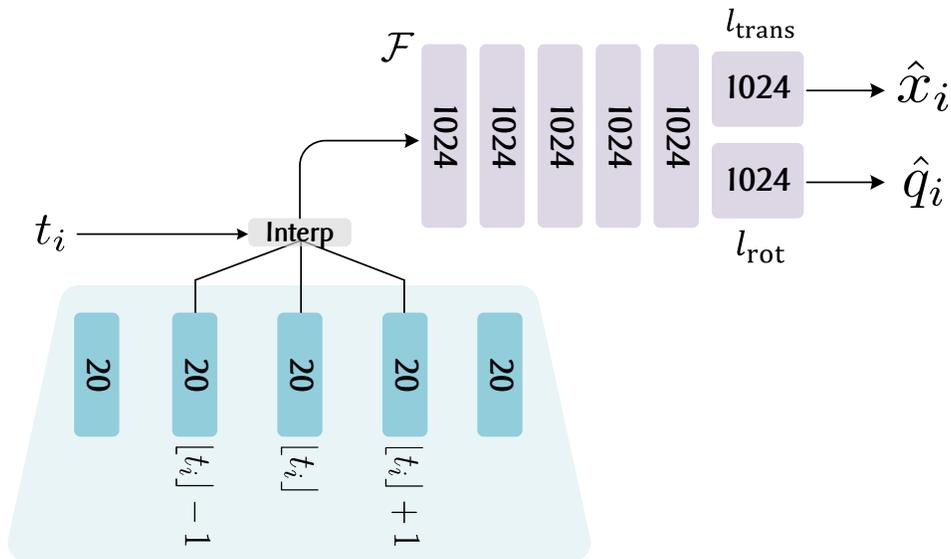


Figure S7: **Architecture.** The 1-D feature grid-based implementation of the Time-Pose Function.

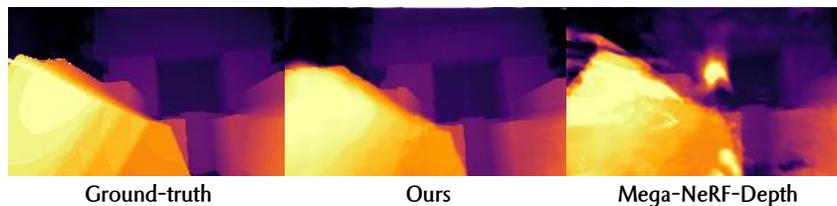


Figure S8: Qualitative results of the comparison of ours with Mega-NeRF-Depth.

24 C Comparison with Mega-NeRF-Depth

25 The estimated depth camera pose from the time-pose function is acceptable, but not sufficient for
 26 direct use to supervise scene geometry. We compared our method against Mega-NeRF [5] with
 27 depth supervision that takes the time-pose function output as depth camera poses. The qualitative
 28 results (Figure S8) and the quantitative results (Table 3 in the main paper) show that the direct use
 29 of outputted camera pose may help increase the performance of Mega-NeRF in depth prediction.

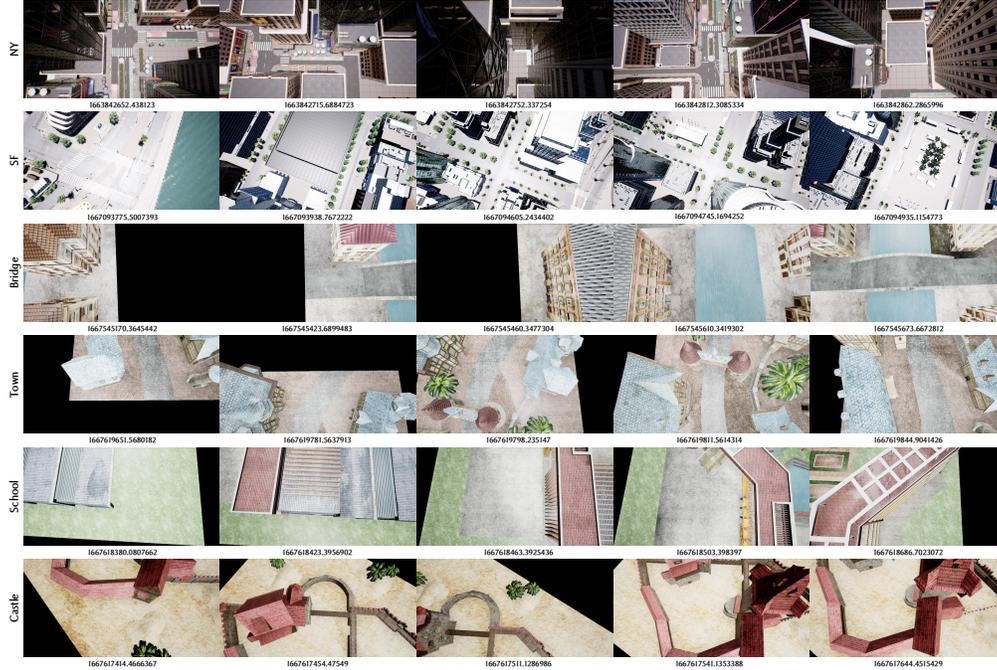


Figure S9: **Data Example** RGB images and the corresponding timestamps in each AUS trajectory.

30 However, the depth prediction results obtained in this way have significant artifacts at the edges of
 31 the scene, and in addition, Mega-NeRF-Depth is even inferior to the baseline method that uses only
 32 RGB as supervision in rendering RGB images.

33 D The AUS Dataset

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

37 We showed some of the captured images along with their timestamps in figure S9.

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

42 E Limitations

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

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

52 **F Acknowledgement**

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

56 **References**

- 57 [1] Liqiang Lin, Yilin Liu, Yue Hu, Xingguang Yan, Ke Xie, and Hui Huang. Capturing, reconstructing, and
58 simulating: the urbanscene3d dataset. In *ECCV*, 2022.
- 59 [2] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren
60 Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In Andrea Vedaldi, Horst
61 Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, Lecture Notes in
62 Computer Science, pages 405–421, Cham, 2020. Springer International Publishing.
- 63 [3] Shital Shah, Debadepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-Fidelity Visual and Physical
64 Simulation for Autonomous Vehicles. In *Field and Service Robotics, Results of the 11th International
65 Conference, {FSR} 2017, Zurich, Switzerland, 12-15 September 2017*, volume 5 of *Springer Proceedings in
66 Advanced Robotics*, pages 621–635. Springer, 2017. titleTranslation:.
- 67 [4] Kirill Sibiriyakov. Artstation page <https://www.artstation.com/vegaart>, 2022.
- 68 [5] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-NeRF: Scalable Construction of
69 Large-Scale NeRFs for Virtual Fly- Throughs. In *2022 IEEE/CVF Conference on Computer Vision and
70 Pattern Recognition (CVPR)*, pages 12912–12921, June 2022. ISSN: 2575-7075.