Appendix

We provide in this supplementary more ablation studies, method analysis, and additional visualizations that could not fit in the paper. In particular, we include (1) the accuracy of various place recognition algorithms, (2) analysis of the computation time of our method and all baselines, (3) more visualizations including heat map and clearer mapping result, (4) and (5) video visualizations of trajectory estimation throughout the training processes.

A. Robustness on map topology

As mentioned in Sec. 3.2, the map topology used for organizing training batches can be obtained by any off-the-shelf algorithms. All results reported in Sec. 4 are based on TF-VPR [44], which is a self-supervised place recognition algorithm. In this supplementary, we compare the mapping accuracy of DeepMapping2 when different approaches are used to provide map topology.

We compute the map topology from a pre-trained PointNetVLAD [41] model and GPS. The mapping results are shown in Tab. I. For drive_0018, there is no significant difference between the mapping results from TF-VPR and PointNetVLAD. Also, using GPS provides a marginally better mapping result, but this is expected given that GPS is used as the ground truth. Due to PointNetVLAD's low-quality map topology, PointNetVLAD for drive_0027 does not produce a good mapping result. In summary, **our method is robust regardless of the map topology used**, as long as it is relatively accurate to reflect the adjacency relationships in the environment.

Table I. **Robustness on map topology.** The table lists the mapping result of DeepMapping2 when running with the map topology attained by different place recognition methods. GPS theoretically provides the most ideal map topology.

PR algo.	Drive_0018		Drive_0027	
	T-ATE (m)↓	R-ATE (°)↓	T-ATE (m)↓	R-ATE (°)↓
TF-VPR [44]	1.81	0.72	2.29	1.57
PointNetVLAD [41]	1.82	0.80	4.79	7.80
GPS (ground truth)	1.62	0.62	2.07	1.42

B. Time analysis

Table II. **Computation time of different methods**. The three baseline methods are run on CPU. DM2 is run on RTX3090 GPU. Note that there is no NVLink when 2 GPUs are used.

Method	Time consumption (s)	
	Drive_0018	Drive_0027
Multiway [38]	113	141
DGR [28] (on CPU)	70200	108522
LeGO-LOAM [15]	287	470
DM2 (1 GPU)	21600	29900
DM2 (2 GPUs)	12085	18587

We compare the computation time of different methods on two trajectories from the KITTI [29] dataset. Due to the nature of training-as-optimization, as we mentioned in Sec. 5, our method takes longer to compute than some of the baselines. When we apply DeepMapping2 to larger datasets, we can apply parallel training to reduce computation time. In Tab. II, it shows that if two GPUs are used for training, the training time is almost decreased by half. It is worth noting that our hardware has no NVLink, which is frequently used in distributed multi-GPU systems to speed up data transmission among GPUs. As a result, the theory and the actual scalability should be very similar. When given sufficient computational resources, the time needed by our method can be significantly reduced, *i.e.*, the time required is expected to be **inversely proportional** to the number of GPUs used. Thus, DeepMapping2 should support a multi-agent setup where the point clouds are scanned by multiple agents and do not have a sequential order.

C. More experiments

We conduct more tests on the simulated dataset [1]. The original DeepMapping pipeline fails on the large-scale dataset, because the drift cannot be correct as described in Sec. 3.1. DeepMapping2 successfully estimates multiple trajectories with different numbers of frames on the simulated point dataset as visualized in Fig. V

We also conduct more experiments on other KITTI sequences [29]. The detailed result is shown in Tab. III.

Table III. **Additional results on the KITTI dataset.**

More sequence	Sequence 02		Sequence 08	
	T-ATE (m)↓	R-ATE (°)↓	T-ATE (m)↓	R-ATE (°)↓
Incremental ICP	8.06	4.57	4.38	4.46
Multiway Registration	4.96	3.37	2.40	0.90
ICP+DM2	**2.56**	**1.31**	**1.85**	**0.81**

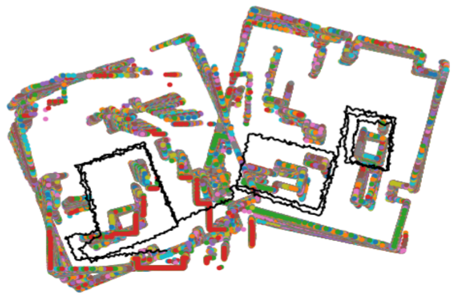


Figure I. **Original DeepMapping results on simulated point cloud dataset.** The black line represents the trajectory, while the color block represents the occupancy grid.

D. More visualization

In order to clearly demonstrate the optimization capability of DeepMapping2, we also offer heat map visualization. Results from `drive_0018`, `drive_0027`, and NCLT are included. It can be shown from Figs. II and III that DeepMapping2 generally has smaller errors compared to other methods. Also, Fig. IV demonstrates how DeepMapping2 improves map from LeGO-LOAM, particularly for the areas indicated by the red box.

We also include a larger and clearer visualization in Fig. VI for the mapping result on the NeBula dataset. It is clear that kinematic odometry fails to align the same locations when they are visited at different times, particularly at two ends of the map. On the other hand, DeepMapping2 significantly improves the map's quality.

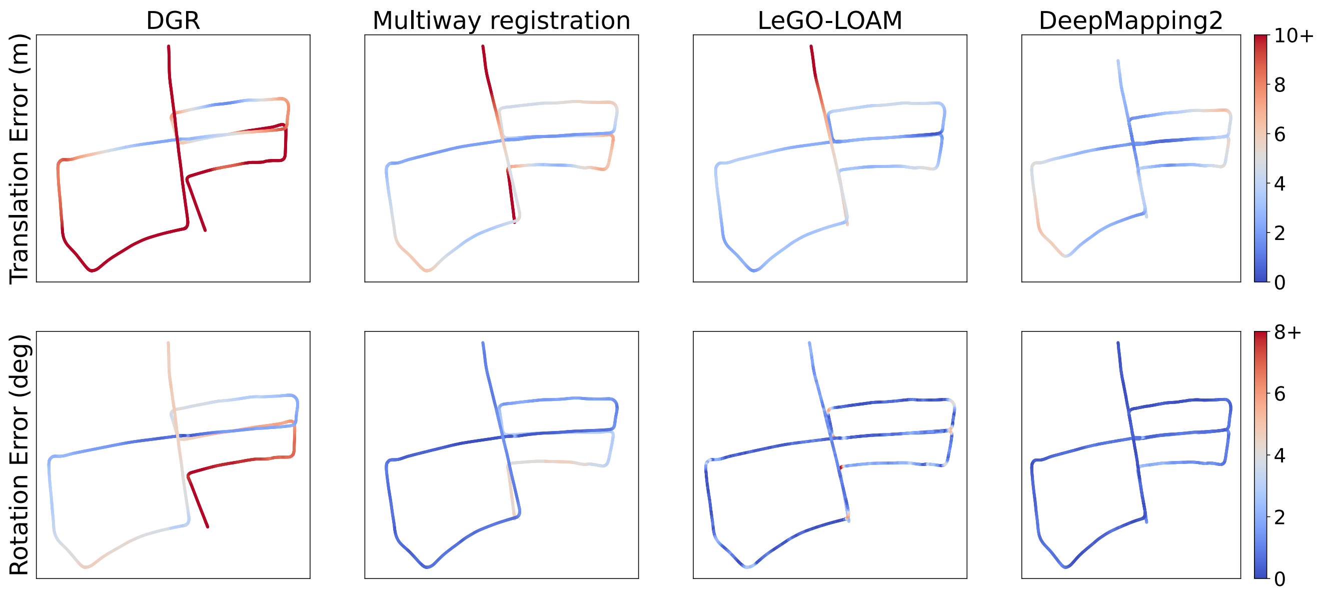


Figure II. **KITTI drive_0018.** Heat map visualization of both translation and rotation ATE for each frame in the dataset. Note that the color bar is clipped for better visualization. Best viewed in color.

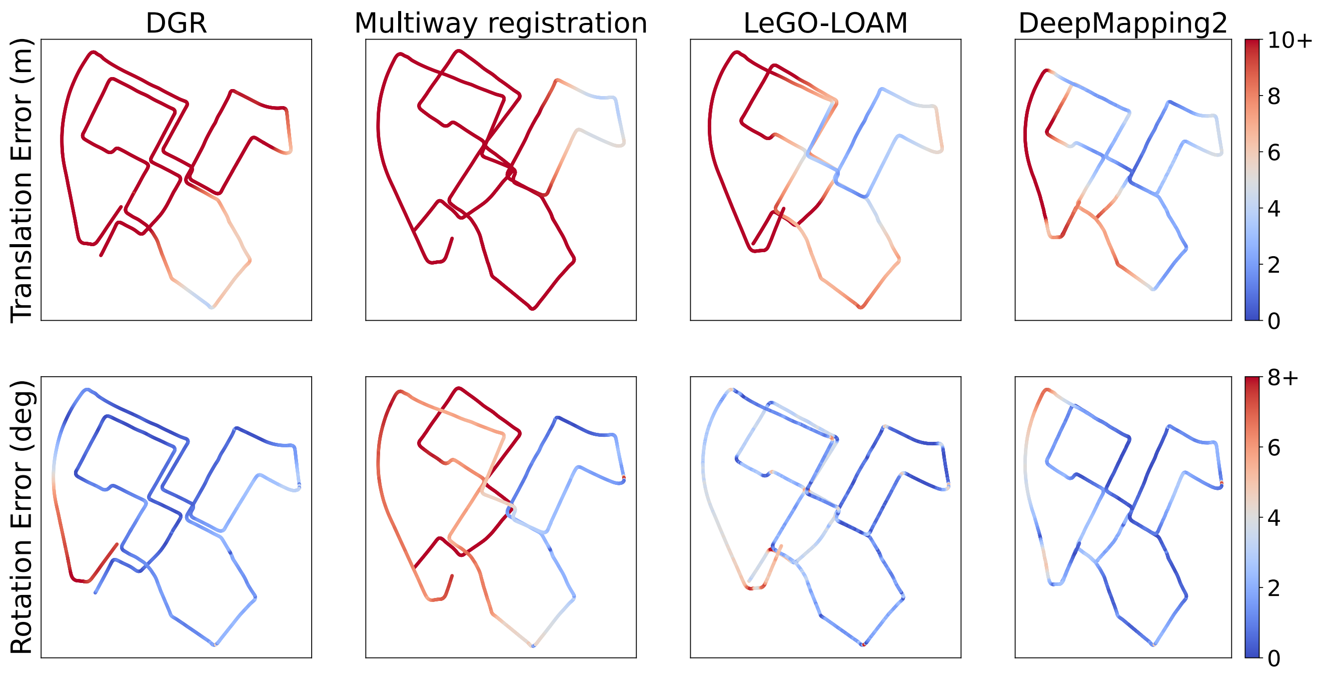


Figure III. **KITTI drive_0027.** Heat map visualization.

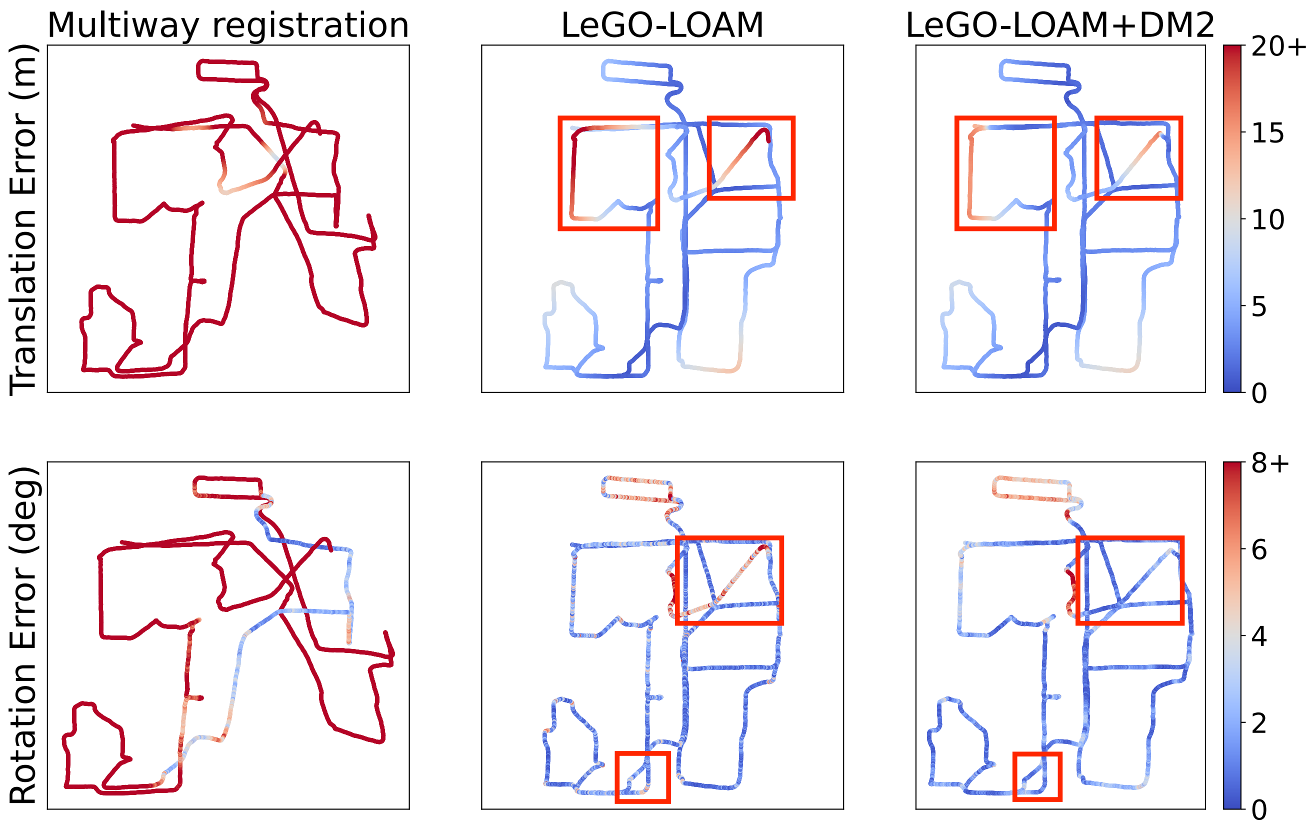
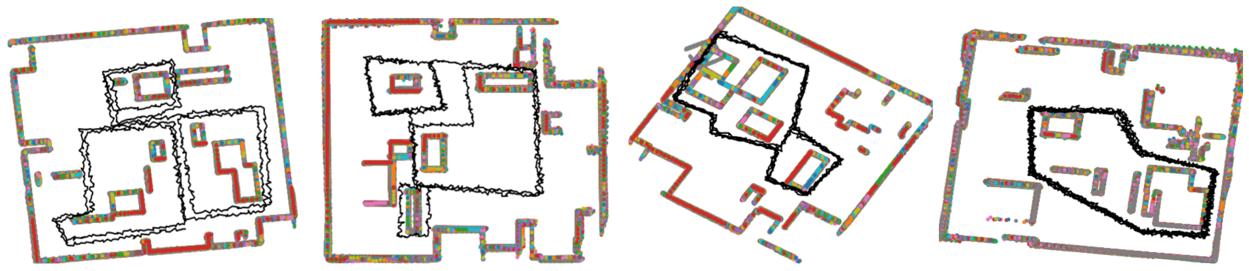
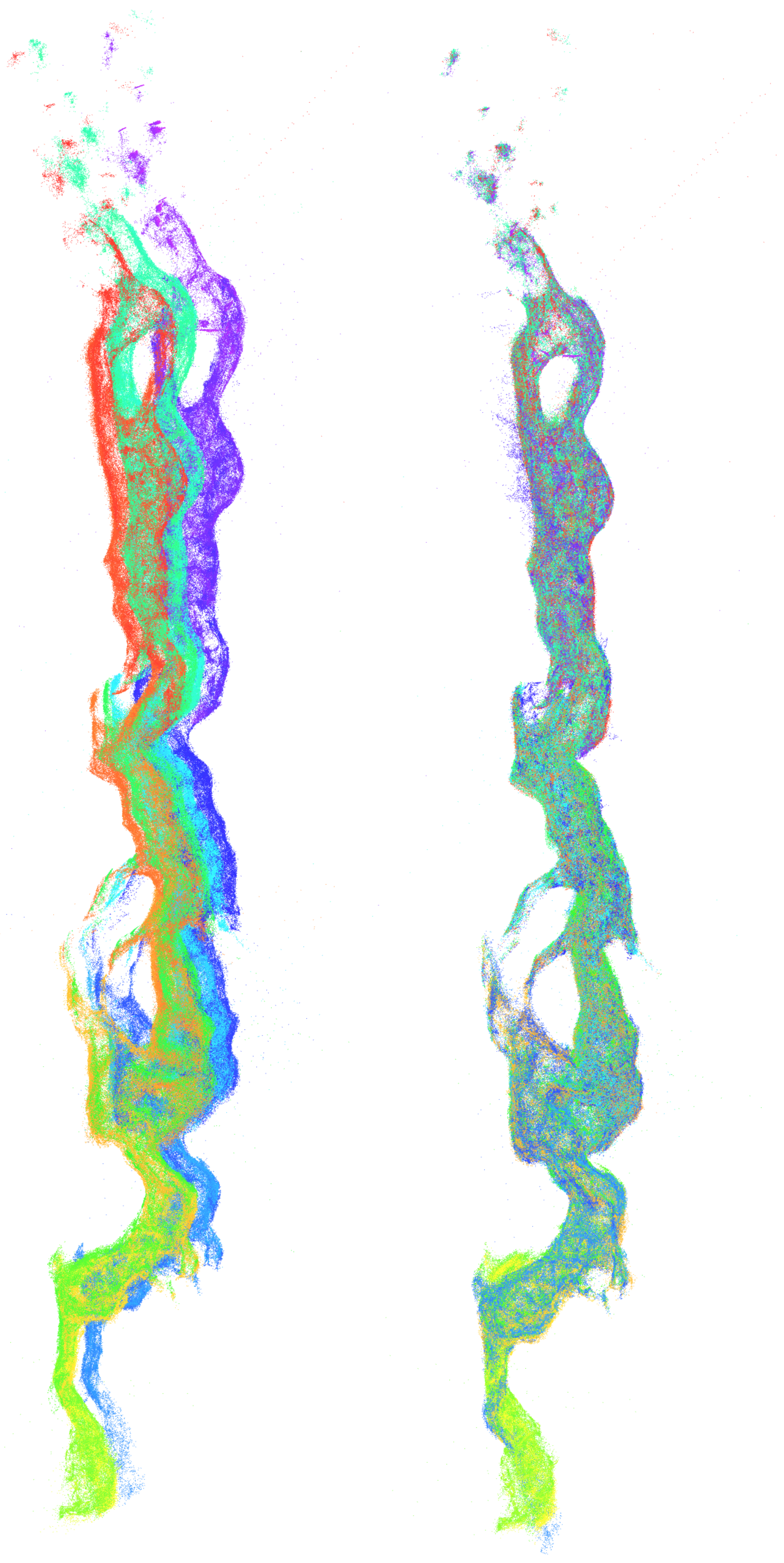


Figure IV. **NCLT.** Heat map visualization. The red boxes highlights the regions where DM2 improves over LeGO-LOAM.



(a) Scene1 (1024 frames) (b) Scene2 (1024 frames) (c) Scene3 (1024 frames) (d) Scene4 (2048 frames)

Figure V. **Mapping and trajectory plot on Simulated point cloud dataset [1]** We include five mapping results including (a)(b)(c) DeepMapping2 mapping results on three different trajectories with 1024 frames (d) DeepMapping2 mapping results on a trajectory with 2048 frames.



(a) Kinematic odometry (KO)

(b) KO+DeepMapping2

Figure VI. **Mapping result on NeBula.** The color of point indicates the frame index in the trajectory.