DeepMapping2: Self-Supervised Large-Scale LiDAR Map Optimization

Chao Chen[1,*], Xinhao Liu[1,*], Yiming Li[1], Li Ding[2], Chen Feng[1,†]
[1]New York University [2] University of Rochester
https://ai4ce.github.io/DeepMapping2/

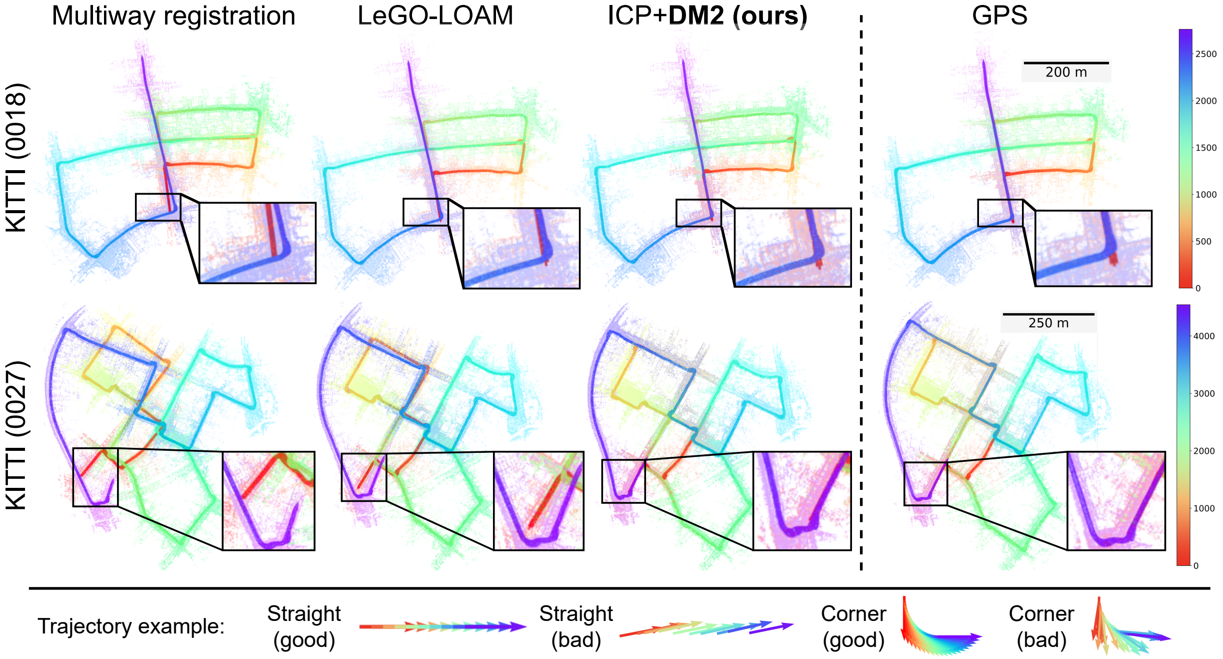


Figure 1. **Mapping result on the KITTI dataset.** The bird's eye view map is shown together with the estimated sensor pose of each frame. Each pose is represented by an arrow indicating the xy-coordinate and heading (yaw angle) as shown in the bottom examples. The color indicates the frame index in the trajectory. Challenging regions are zoomed in for a clear view. Best viewed in color.

Abstract

LiDAR mapping is important yet challenging in self-driving and mobile robotics. To tackle such a global point cloud registration problem, DeepMapping [1] converts the complex map estimation into a self-supervised training of simple deep networks. Despite its broad convergence range on small datasets, DeepMapping still cannot produce satisfactory results on large-scale datasets with thousands of frames. This is due to the lack of loop closures and exact cross-frame point correspondences, and the slow convergence of its global localization network. We propose DeepMapping2 by adding two novel techniques to address these issues: (1) organization of training batch based on map topology from loop closing, and (2) self-supervised local-to-global point consistency loss leveraging pairwise registration. Our experiments and ablation studies on public datasets such as KITTI, NCLT, and Nebula demonstrate the effectiveness of our method.

1. Introduction

Mapping is a fundamental ability for autonomous mobile agents. It organizes an agent's local sensor observations into a map, i.e., a global spatial representation of the environment. A pre-built map is useful in robotics, self-driving, and augmented reality for agents to localize themselves [2–6]. Various simultaneous localization and mapping (SLAM) methods can create maps of new environments from 2D and/or 3D sensors [7–13]. In particular, LiDAR-based mapping is often adopted to build large-scale maps in self-driving and mobile robotics due to LiDAR's direct and accurate 3D point cloud sensing capability.

[*]Equal contribution
[†]The corresponding author is Chen Feng cfeng@nyu.edu

Similar to visual SLAM, LiDAR SLAM methods typically contain front-end and back-end modules [14–17]. The front-end module tracks sensor movements by LiDAR/inertial/wheel odometry and provides constraints between sequential frames of point clouds by either iterative closest point (ICP) or 3D feature detection and correspondence matching algorithms. The back-end uses those constraints in a pose/pose-landmark graph optimization [18,19] to minimize the odometry drift, similar to the bundle adjustment in visual SLAM and Structure-from-Motion (SfM).

However, without accurate GNSS/IMU as odometry, large-scale LiDAR mapping results could be unsatisfactory (see Fig. 1), due to errors in LiDAR odometry and difficulties in correspondence matching and loop closing, especially outdoors. To tackle these issues, researchers start to explore deep learning methods. Some of them focus on replacing sub-tasks in LiDAR mapping with deep networks [20–23], following the common machine learning paradigm: `train-then-test`. Yet such methods could face generalization issues when the training dataset domain is different than the testing one.

Differently, DeepMapping [1] proposes a new paradigm: `training-as-optimization` for point cloud mapping. It encapsulates the global registration in a point-cloud-based PoseNet [24] (L-Net), and evaluates the map quality using another binary occupancy network (M-Net) with a binary cross-entropy (BCE) loss. This converts the continuous map optimization into a self-supervised training of binary classifications. Since *no testing is needed*, it does not face any generalization issues because mapping is done once training is finished.

However, despite its superior performance on small-scale datasets, we found DeepMapping often fails on large-scale datasets due to the following challenges:

(1) *No-explicit-loop-closure*: DeepMapping gradually optimizes L-Net using frames in each mini-batch that are temporal neighbors, and only relies on M-Net to control the global map consistency. This is like incremental registration that is doomed to drift when the number of frames is large. SLAM solves this by loop closing, which is not yet clear how to be incorporated into DeepMapping.

(2) *No-local-registration*: Although previous works have shown local registration to be locally accurate [25–28], DeepMapping only uses it in the ICP-based pose initialization but not in the optimization. This is due to a common problem faced by all LiDAR registration methods, the lack of point correspondences in LiDAR point clouds: the same 3D point rarely appears again in another scan, because of the sparse sensor resolution and long-range sensing.

(3) *Slow-convergence-in-global-registration*: L-Net regresses a single frame of point cloud into its global pose, which is supervised only by the M-Net and BCE loss. Unlike pairwise registration, this global registration lacks enough inference cues to output correct poses, thus leading to slow convergence when the dataset is large.

We propose DeepMapping2 that is able to effectively optimize maps on large-scale LiDAR datasets. It extends DeepMapping with two novel techniques. The first one addresses challenge (1) by organizing data frames into training batches based on map topology from loop closing. This allows a frame with its topological/spatial neighbors to be grouped into the same batch. We find this to be the best way of adding loop closing into DeepMapping which uses free-space inconsistencies via M-Net and BCE loss to generate self-supervision, because such inconsistencies happen mostly between unregistered neighboring frames.

The second technique is a novel self-supervised local-to-global point consistency loss that leverages precomputed pairwise registration. For each point in a frame, we can compute this new consistency as the L2 distance between different versions of its global coordinate calculated using a neighboring frame's global pose and the relative pose between the two frames from the pairwise registration. This allows us to address challenge (2) without relying on point correspondences between different frames: even if two neighboring frames do not have enough common points as correspondences for pairwise local registration, we can still incorporate the local registration's results during training. It also addresses the challenge (3) because now L-Net is supervised by stronger gradients from not only the BCE loss, but also the new consistency loss.

Our contributions are summarized as follows:

- Our DeepMapping2 is the first self-supervised large-scale LiDAR map optimization method as far as we know, and this generic method achieves state-of-the-art mapping results on various indoor/outdoor public datasets, including KITTI [29], NCLT [30], and the challenging underground dataset Nebula [31].
- Our analysis reveals why DeepMapping fails to scale up and leads to the two novel techniques–batch organization and local-to-global point consistency loss–to incorporate loop closing and local registration in the DeepMapping framework. Their necessity and effectiveness are further validated in our ablation study.

2. Related Work

Pairwise registration. Pairwise registration computes the transformation from a source point cloud to a target point cloud. A global pose of each point cloud can be obtained by incrementing the pairwise transformation sequentially. Iterative closest point (ICP) [32] and its variants [25, 33] are widely-used pairwise registration methods by iteratively estimating and minimizing closest point correspondence. More recently, learning-based algorithms use a pre-trained model to predict pairwise transformation [27, 28]. However, all these methods suffer from aggregation errors when the registration is done incrementally to solve a mapping problem. Though DeepMapping2 relies on pairwise registration for consistency loss, it only benefits from the relatively accurate pairwise transformation but does not suffer aggregation errors because registration is not done incrementally.

Multiple registration. In addition to pairwise registration, several methods for multiple point cloud registration have been proposed [9, 34, 35]. Global SfM [36, 37] divides one scene into smaller partitions and iteratively does motion averaging until convergence. [38] transforms the problem into pose-graph optimization and uses partitioning to reduce computation. [39] follows a similar practice but with a supervised end-to-end learnable algorithm. On the other hand, DeepMapping [1] does not require any scene partitioning or supervised training. Instead, it approaches the problem by transforming global map optimization into self-supervised training of deep networks. DeepMapping2 follows the same line of thought and inherits the general framework of DeepMapping. It improves the mapping result of DeepMapping, especially in large-scale scenes, and overcomes challenges faced by DeepMapping by the two proposed novel techniques.

Loop closure. Loop closure is a significant component in SLAM algorithms. It aims to decide whether a place has been visited before by the agent. Most classical loop closure methods [40] rely on hand-crafted features and descriptors to compare different frames. Deep learning approaches like PointNetVLAD [41] combine the feature extraction from PointNet [42] and the supervised contrastive loss from NetVLAD [43]. In contrast, TF-VPR [44] trains a similar deep-learning based network without supervision by mining the information between temporal and feature neighbors. It is a challenge, however, to incorporate loop closing in the training of DeepMapping because the detected loop closure cannot be described in a differentiable manner. DeepMapping2 solves this problem by organizing the training batches with spatially adjacent frames and incorporating loop closure information into the self-supervised training process.

Simultaneous Localization and Mapping. SLAM is the computational problem of building a map while keeping track of an agent's location. Both visual [45–47] and LiDAR SLAM [14, 15, 32, 48, 49] are well-studied in this domain. Particularly, LiDAR SLAM provides relatively accurate geometric information and can be classified into train-then-test and train-as-optimization. The majority of methods fall into the train-then-test category. Such LiDAR SLAM methods perform point-level matching [32], feature-level matching [48, 49], and point feature to edge/plane matching [14, 15] to find correspondences between scans. However, existing LiDAR SLAM algorithms are prone to large errors, particularly in estimating the sensor rotation of each frame. DeepMapping [1], on the other hand, belongs to the train-as-optimization category, and the network trained does not generalize in other scenes because the self-supervised loss functions in DeepMapping2 are designed for same scene optimization.

3. Method

3.1. Overview

Problem setup. We aim to solve the problem of registering multiple point clouds into a single global frame. Formally, the input point cloud is denoted as $\mathcal{S} = \{S_i\}_{i=1}^K$, where K is the total number of point clouds. Each point cloud S_i is represented by a $N_i \times 3$ matrix where N_i is the number of points in S_i, *i.e.* $S_i \in \mathbb{R}^{N_i \times 3}$. The goal is to estimate the sensor pose $\mathcal{T} = \{T_i\}_{i=1}^K$ for each S_i, where $T_i \in SE(3)$.

Common ground with DeepMapping. Inspired by DeepMapping [1], we use deep neural networks, represented as a function f with learnable parameters, to estimate sensor poses $\mathcal{T}$. The registration problem is then transformed into finding the optimal network parameters that minimize the objective function

$$(\theta^*, \phi^*) = \arg\min_{\theta,\phi} \mathcal{L}_\phi(f_\theta(\mathcal{S}), \mathcal{S}), \tag{1}$$

where $f_\theta : S_i \mapsto T_i$ is a localization network (L-Net) that estimates the global pose for each point cloud, and ϕ is the parameter of a map network (M-Net) m_ϕ that maps from a global coordinate to its corresponding occupancy probability. $\mathcal{L}_\phi$ is the self-supervised binary cross entropy loss to measure the global registration quality:

$$\mathcal{L}_\phi = \frac{1}{K}\sum_{i=1}^{K} B\left[m_\phi\left(G_i\right), 1\right] + B\left[m_\phi\left(s\left(G_i\right)\right), 0\right], \tag{2}$$

where the global point cloud G_i is a function of L-Net parameters θ, and $s(G_i)$ is a set of points sampled from the free space in G_i. $B[p, y]$ in Eq. (2) is the binary cross entropy between the predicted occupancy probability p and the self-supervised binary label y:

$$B[p, y] = -y\log(p) - (1-y)\log(1-p). \tag{3}$$

Moreover, Chamfer distance is another loss to help the network converge faster in DeepMapping [1]. It measures the distance of two global point clouds X and Y by:

$$\begin{aligned} d(X,Y) = &\frac{1}{|X|}\sum_{\mathbf{x}\in X}\min_{\mathbf{y}\in Y}\|\mathbf{x}-\mathbf{y}\|_2 \\ &+ \frac{1}{|Y|}\sum_{\mathbf{y}\in Y}\min_{\mathbf{x}\in X}\|\mathbf{x}-\mathbf{y}\|_2. \end{aligned} \tag{4}$$

Difference from DeepMapping. DeepMapping introduces a ”warm start” mechanism to transform each raw

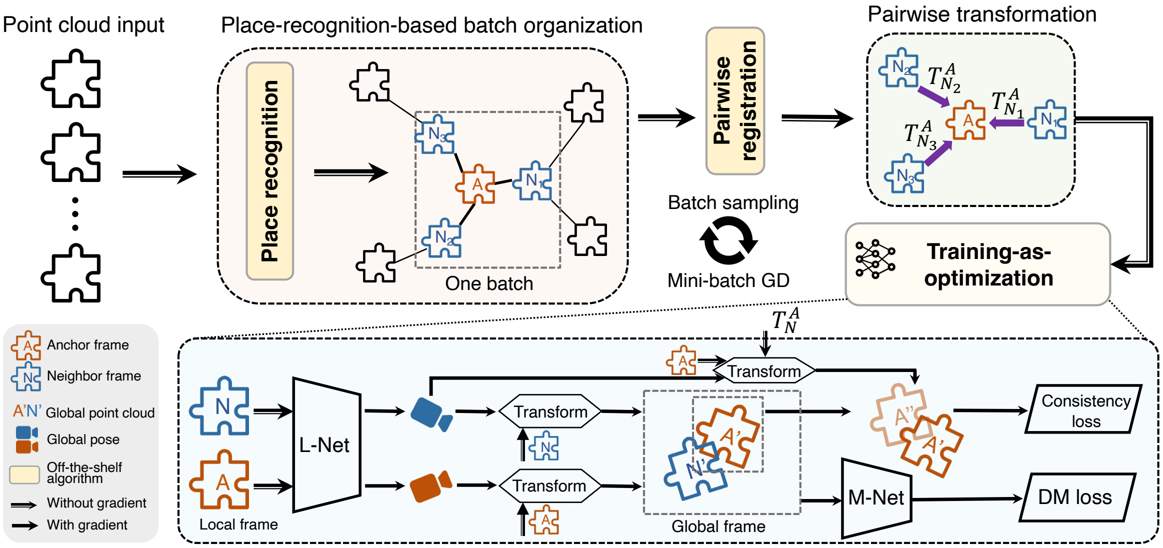


Figure 2. **Pipeline of DeepMapping2.** The pipeline mainly consists of place-recognition-based *batch organization* and learning-based *optimization*. In batch organization, the input point clouds are organized into mini-batches by topological map attained from place recognition. Each batch contains an anchor frame A and several spatially closed neighbor frames N. The transformation between the anchor frame and each neighbor frame T_N^A is obtained by pairwise registration. In optimization, each batch is fed into L-Net to estimate the global pose. The transformed global anchor frame is then obtained in two ways: A' directly from the global pose of the anchor frame (T_A^G) and A'' from the global pose of the neighbor frame (T_N^G) and the pairwise registration (T_N^A). The consistency loss penalizes the differences between A' and A''. The global frames are also fed into M-Net for computing DeepMapping loss. Best viewed in color.

point cloud to a sub-optimal global pose via existing registration methods like ICP. This optional step will accelerate the convergence of DeepMapping in the small-scale dataset. However, a reasonable initialization is required in our method for large-scale datasets, because it would be difficult to converge if starting from scratch.

Limitation of DeepMapping. Despite its success in small datasets, DeepMapping cannot scale up to large datasets because of the aforementioned challenges: (1) *no-explicit-loop-closure*, (2) *no-local-registration*, and (3) *slow-convergence-in-global-registration*.

In Sec. 3.2, we will introduce the pipeline of our method. We will introduce the solution to challenge (1) in Sec. 3.3, and the solution to challenges (2) and (3) in Sec. 3.4

3.2. Pipeline

Batch organization. The pipeline of DeepMapping2 has two main stages as shown in yellow and blue blocks in Fig. 2. The first stage is the organization of training batches based on map topology from place recognition. The input is a sequence of point clouds that is initially registered to an intermediate state between scratch and the ground truth. Each anchor frame A is organized with its neighbor frame N into one batch using the map topology from off-the-shelf place recognition algorithms [41,44] (A and N are formally defined in Sec. 3.3).

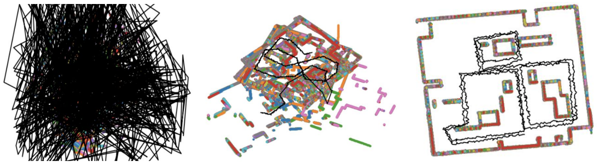
Pairwise registration. Before the optimization in the second stage, pairwise registration is calculated between the anchor frame and the neighbor frames in each batch by finding the transformation from the neighbors to the anchor. This can also be achieved by any off-the-shelf pairwise registration algorithms.

Training as optimization. The second stage of the pipeline is learning-based optimization. Besides those important components that have been introduced in Sec. 3.1, the consistency loss is another key component of DeepMapping2. The loss function is designed with the following idea: for each point in the anchor frame A, we can compute its consistency as the L2 difference between different versions (A' and A'') of its global coordinates. These versions are calculated from the global pose of each neighboring frame N and the relative transformation between the anchor and the neighbor. A more detailed explanation of the loss function will be given in Sec. 3.4.

3.3. Batch organization

Due to the lack of loop closure in DeepMapping, extending the method to a large-scale environment is challenging. Though loop closure can be easily integrated into a SLAM system (*e.g.* through bundle adjustment), it is hard to implement in deep-learning-based mapping methods because of its non-differentiability.

Effect of different batch organization. Dividing the whole trajectory into multiple mini-batches is inevitable. Then what is the correct way of such division? We tested several batch arrangements to split the dataset as shown



(a) Random (b) Temporal (c) Spatial topology

Figure 3. Illustration of various batch organization methods. We show the mapping results and the trajectories in a large-scale toy dataset [1]. Using a spatial group would produce the best mapping results compared to other mapping methods. Other methods fail because the network wrongly registers frames from different areas together. Note that (b) is adopted in DeepMapping [1].

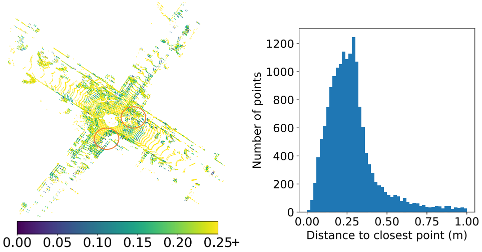
in Fig. 3. In Fig. 3a, a random organization of batches leads to an inferior mapping result because the network tends to pull random pairs of frames together and register them. In Fig. 3b, the batches are organized sequentially, as in DeepMapping. They can be registered well locally, but the global mapping quality is still inferior. Not surprisingly, in Fig. 3c, using spatially adjacent neighbors to form a batch would yield the best result. The reason is that M-Net would pull together the frames in a batch and register them together. Meanwhile, loop closure is incorporated into the training process because the loop is closed when all spatially adjacent frames are registered correctly in the loop.

Loop-closure-based batch organization. We construct a batch that contains spatially adjacent neighbors (A and Ns) using map topology from off-the-shelf place recognition algorithms. The map topology is a connected graph where each node represents a frame, and each edge connects two spatially adjacent nodes. To construct the batches, given an anchor frame A, we find the top k closest frames connected to the anchor frame in map topology and organize them into one batch. By doing so, we can include loop closure information in the training process by letting M-Net register those spatially adjacent frames. This also makes it possible to construct the pairwise constraint in the local-to-global point consistency loss explained in Sec. 3.4.

3.4. Local-to-global point consistency loss

Slow convergence. Even with the new batch organization, DeepMapping can still converge slowly, especially in large-scale scenes. Although initialization can give a "warm start" to the network, this information becomes decrepit as the training gets farther away from the initialization because it does not provide enough constraint on global pose estimation. Therefore, we want to incorporate the information from the initialization into the training process to provide a global inference cue for the networks. This can be done by constraining the pairwise relations of two adjacent point clouds to minimize their distance in the global frame.

Point correspondence. However, point-level correspondence is needed to compute the distance between points.



(a) Color-coded distance (m) (b) Histogram of distance

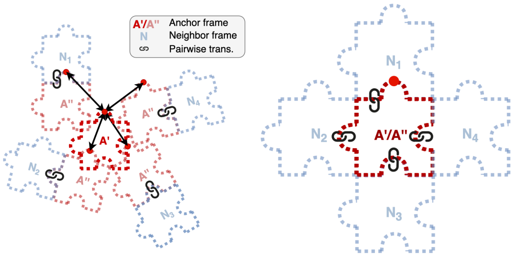
Figure 4. **Example of LiDAR point clouds lacking point-to-point correspondence**. We select two consecutive frames from the KITTI [29] dataset and register them with ground truth transformation. The distance between the two sensor poses is 1.09 m. For each point in the point cloud, we compute its distance to the closest point after registration. The color-coded distance is shown in (a) and the histogram in (b). The place where points have a distance smaller than 0.05 m is marked by red circles.

This correspondence is rare in large-scale and/or outdoor scenarios where mapping algorithms are often applied. As shown in Fig. 4, most points are about 0.25 m away from their closest points in the real-world dataset. Although alternative distance can be calculated by finding the closest points or abstractly described by hand-crafted descriptors [50] or learning-based features [27], they can be inaccurate because point-to-point correspondence may not exist in the first place. When two point clouds are scanned at different sensor poses, it is very unlikely that the same point would exist in both point clouds due to the sparsity of LiDAR scanning. This makes the correspondence found by the methods above inaccurate. Thus, we ask the question: How can we constrain the pairwise relations of globally registered point clouds while not relying on inaccurate point-to-point correspondence?

Distance metric. We approach the question by considering the points in a single point cloud. When the point cloud S is transformed by different transformation matrices T, the point-to-point correspondence is preserved because they are from the same local point cloud. After transformation, the distance between the two point clouds can be easily calculated by averaging the L2 distance between each corresponding point. The metric can be defined as a function of two transformations T, T' and one single point cloud S:

$$d(T, T', S) = \frac{1}{|S|} \sum_{s \in S} \|Ts - T's\|_2. \tag{5}$$

Note that Eq. (5) not only measures the distance between the two point clouds after transformation but also reflects the difference (inconsistency) between the two transformations. This is desired in our problem because a relatively accurate pairwise transformation is available from each neighbor frame to the anchor frame. We want to constrain the estimation of L-Net so that the pairwise relations are preserved in the global sensor poses.



(a) Poor global registration (b) Perfect global registration

Figure 5. **Illustration of consistency loss.** Red puzzle represents different versions of anchor frames and blue puzzle represents the neighbor frames. The loss measures the euclidean distance of all the corresponding points of different versions of anchor frames that transformed from different poses, as indicated by the two-way arrows. Red points in the figure show one of the points for illustration.

Fig. 5 depicts the idea of this metric. Two versions of anchor frames are shown. A' is transformed by the global pose estimated by L-Net. A'' is transformed by each neighbor frame's global pose and relative transformation. In Fig. 5a, the point clouds are poorly registered so the distances indicated by the arrows are large. In Fig. 5b, however, all corresponding points overlap so the distance is zero.

Consistency loss. Following this line of thought, we design the local-to-global consistency loss to measure the inconsistency between the local and global registrations. We denote the pairwise transformation between S_i and S_j as T_i^j, and the global pose of S_i as T_i^G. Also, recall that neighbor $\mathcal{N}_i$ is defined as the indices of neighbor frames of S_i. We formulate the consistency loss as

$$\mathcal{L}_C = \frac{1}{K|\mathcal{N}_i|}\sum_{i=1}^{K}\sum_{j\in\mathcal{N}_i} d(T_j^G T_j^i, T_i^G, S_i). \tag{6}$$

It is worth noting that loss in Eq. (6) solves the two challenges mentioned above altogether. Global inference cue is provided while avoiding point-to-point correspondence. Because each T_j^i is computed pairwisely and is considered more accurate than pairwise relations in the global estimation, by minimizing $\mathcal{L}_C$, the network can get more information from the initialization and have a faster convergence.

4. Experiment

Dataset. We use three datasets for a comprehensive evaluation: (1) KITTI dataset [29] for evaluations in outdoor scenarios, (2) NeBula odometry dataset [31] for evaluations in indoor environments where GPS signal is not available, and (3) NCLT dataset [30] for evaluations in both scenarios.

Metrics. For quantitative evaluations, we use the absolute trajectory error (ATE) [51] following DeepMapping [1]. For qualitative evaluations, we visualize both the registered map and the trajectory in the bird's eye view.

Baselines. We compare our method with baselines including ICP [32], multiway registration [38], DGR [28], HRegNet [52], GeoTransformer [53], and KISS-ICP [54], and LeGO-LOAM [15]. We also perform ablation studies to compare the effectiveness of each proposed techniques.

Warm start. Following DeepMapping [1], this step has the same function as the front-end initialization in SfM before bundle adjustment (BA). In fact, DeepMapping2 can be seen as the deep learning version of BA for LiDAR mapping. Note that just like BA in SfM is affected by the initial solution's quality, DeepMapping2 is also affected by the "warm start" quality. Nonetheless, DeepMapping2 can be seamlessly integrated into almost any mapping front-end such as KISS-ICP, and Lego-LOAM, and always improves ATE without manual hyperparameter tuning.

Implementation details. We use the same network structure as in [1]. The orientations are represented by Euler angles. We also tried other rotation representations (such as 6D representation [55]) and obtained similar results.

4.1. KITTI dataset

KITTI [29] is a widely-used authoritative benchmark to evaluate SLAM-based algorithms [56]. We employ the two most complex and challenging scenes from the dataset, where lots of places are revisited multiple times and the explored area is relatively large. Meanwhile, there are dynamic objects on the road, further increasing the difficulties.

Quantitative comparisons. From Tab. 1, we see that DeepMapping2 can perform well even with very trivial map initialization, like incremental ICP. There are significant improvements in the map quality when comparing the optimized map and the initialization, *e.g.*, DeepMapping2 improves LeGO-LOAM by 16.6% in terms of translation on drive_0018. In general, our method is robust to initialization. No matter what category of methods is used to initialize the map, the performance of our method is consistent.

Qualitative comparisons. As shown in Fig. 1, although multiway registration and DGR perform well at the start of the trajectory, the error is accumulated as the number of frames increases, leading to a noticeable drift. While the loop closure module in LeGO-LOAM is supposed to correct the drift, we can still see the errors on the trajectory, especially when the number of frames in drive_0027 exceeds 4,000. On the other hand, the qualitative results produced by our approach do not have a noticeable difference from those provided by the GPS sensor.

4.2. NCLT dataset

NCLT [30] is a large-scale dataset collected at the University of Michigan's North Campus. The point clouds are collected using a Velodyne HDL-32E 3D LiDAR mounted on a Segway robot. In total, the NCLT dataset has a robot trajectory with a length of 147.4 km and maintains 27 dis-

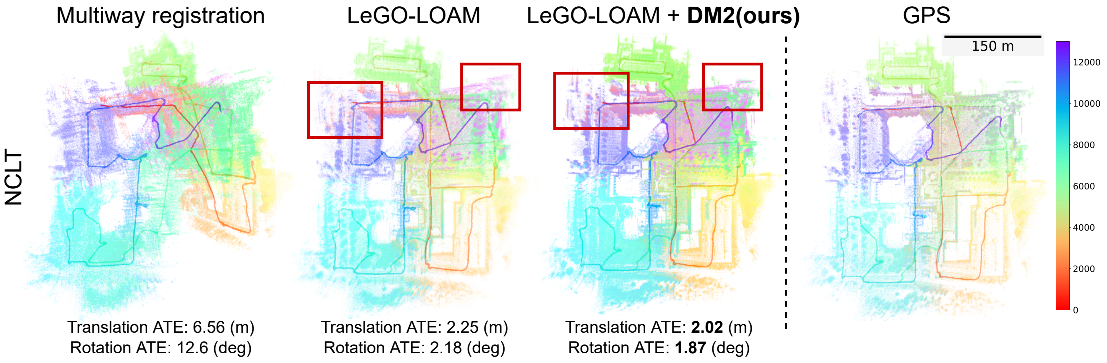


Figure 6. **Mapping results on NCLT dataset.** The red boxes indicate the places where our method has a better mapping quality than LeGO-LOAM. The ATEs are also listed. Other labels and legends follow Fig. 1. Best viewed in color.

Table 1. **Quantitative result on the KITTI dataset.** The quality of the predicted trajectory is measured by absolute trajectory error (ATE) compared with the GPS trajectory. Results on two trajectories are listed. "T-ATE" means translation ATE (m) "R-ATE" means rotation ATE (deg). The last two rows are our results.

Method	**Drive_0018**		**Drive_0027**	
	T-ATE (m)↓	**R-ATE (°)↓**	**T-ATE (m)↓**	**R-ATE (°)↓**
Incremental ICP	4.38	4.61	3.53	2.67
Multiway [38]	2.24	1.75	4.70	5.93
DGR [28]	3.15	4.09	4.12	1.59
LeGO-LOAM [15]	1.90	1.36	2.96	2.36
HRegNet [52]	30.61	94.90	45.49	85.36
GeoTransformer [53]	4.03	3.02	10.15	15.34
ICP+DM [1]	3.42	1.66	3.39	2.70
KISS-ICP [54]	2.10	0.68	6.25	1.21
ICP+**DM2**	1.81	0.72	**2.29**	1.57
KISS-ICP+**DM2**	1.78	**0.68**	2.30	**1.17**
LeGO-LOAM+**DM2**	**1.63**	1.18	2.59	2.27

Table 2. **Quantitative results on the NCLT dataset**. The notation is the same as in Tab. 1. The last two rows are our results.

Method	**T-ATE (m)↓**	**R-ATE (°)↓**
Incremental ICP	6.20	12.95
Multiway [38]	6.56	12.6
DGR [28]	8.89	42.9
LeGO-LOAM [15]	2.25	2.18
ICP+**DM2**	3.73	6.27
LeGO-LOAM+**DM2**	**2.02**	**1.87**

crete mapping sessions over the year. Each mapping session includes both indoor and outdoor environments. We select an interval of the trajectory for better illustration.

Qualitative results As shown in Fig. 6, although LeGO-LOAM can produce a relatively satisfactory map, it still misaligns point clouds in some areas of the trajectory. In the red boxes in Fig. 6, the optimized map by DeepMapping2 has better alignment than that from LeGO-LOAM, which is also proved by the ATEs reported under the maps.

Quantitative results The translation and rotation ATEs of different methods on NCLT are reported in Tab. 2. Incremental ICP does not have good map quality. Hence, when using it as initialization, our method does not have a better map than LeGO-LOAM, despite the fact that it reduces the errors by nearly one-half. Nevertheless, our method can still improve the map from LeGO-LOAM and have lower ATEs.

It is worth noting that the slightly better map quality by [15] mainly results from the smaller area and more frames in the trajectories (shown by the scale in Fig. 6). However, this is not always expected in all implementations. Comparing the results in KITTI and NCLT, we find our method having a consistent improvement to the initialization regardless of the sampling density of the environment.

4.3. NeBula dataset

NeBula odometry [31] dataset is provided by Team CoSTAR. The dataset is used to test the multi-robot system in real-world environments such as the DARPA Subterranean Challenge [57]. In each scene of the dataset, a ground-truth survey-grade map is provided by DARPA and the trajectory is produced by running LOCUS 2.0 [58]. Visual odometry and kinematic odometry are also included. We test different methods on the data collected in Lava Beds National Monument. The whole trajectory is 590.85 meters long and contains 37,949 LiDAR scans in total.

Improvement to the odometry. Nebula is very challenging because it contains a long indoor trajectory without any marker for loop closure detection. We first run initialization on the dataset with incremental ICP and LeGO-LOAM. However, the mapping result is very poor (see Fig. 7) and cannot be used as a "warm start" for DeepMapping2. We find, however, the provided kinematic odometry gives a good mapping result but can still be optimized. Hence, we run DeepMapping2 with the initialization from the kinematic odometry. As shown in Fig. 7, the optimized map corrects the large misalignments at the two ends of the tunnel. It is very accurate qualitatively compared to the survey-grade map. This dataset does not provide the complete trajectory so we cannot do a quantitative analysis.

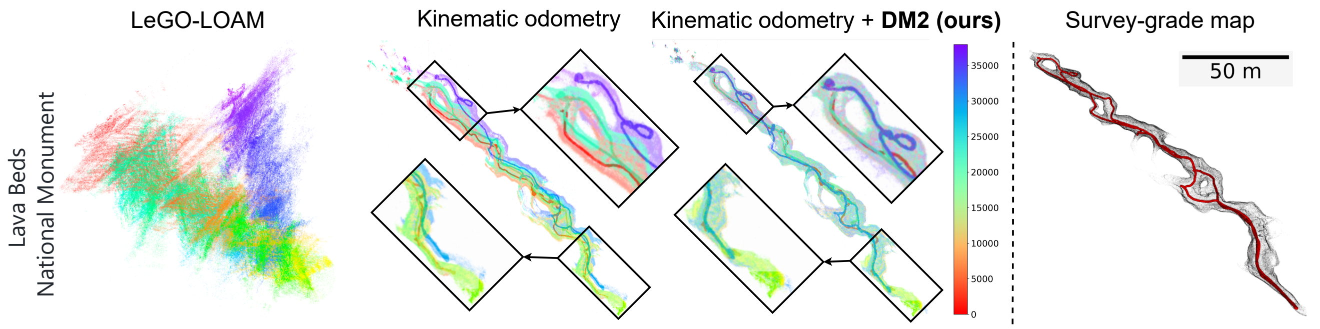


Figure 7. **Mapping result on NeBula dataset.** Several parts of the map are zoomed in for a clearer demonstration. As far as we know, we are the first to show this quality of mapping result of the trajectory in the Nebula dataset. Other labels and legends follow Fig. 1.

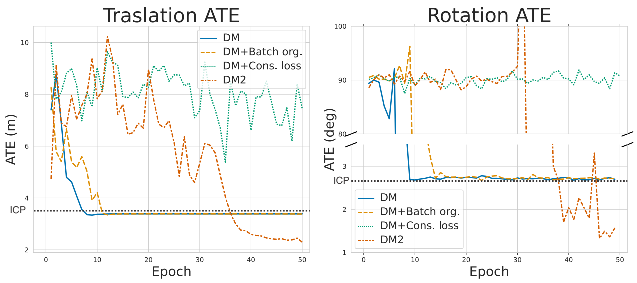


Figure 8. **ATE versus training epoch on drive_0027 in KITTI.** The legend indicates the ATEs with different components in Tab. 3.

4.4. Ablation study

We do experiments on the KITTI dataset to analyze how the proposed techniques influence the mapping quality. We compare the quantitative results when one or two of the components are missing from the pipeline.

Batch organization As shown in Tab. 3, batch organization plays an important role in mapping. The first version of DeepMapping (DM) constructs a map with batches organized sequentially. Drifts will occur when encountering large-scale scenarios with many revisits. On the other hand, our method can have a better mapping quality with batch organization due to implicit loop closure.

Consistency loss. Despite the benefits of local-to-global consistency loss mentioned in Sec. 3.4, it requires accurate pairwise registrations to work. Row 1 and row 3 in Tab. 3 show that adding consistency loss alone to DeepMapping loss does not lead to a significant improvement. This is because the pairwise registration from off-the-shelf algorithms is prone to errors for non-adjacent point clouds. Comparing row 2 to row 5 in Tab. 3, only when batch organization groups spatially adjacent frames together, can consistency loss work ideally and have the best mapping quality. It can also be seen from Fig. 8 that consistency loss speed-ups the convergence.

Table 3. **Ablation study on the KITTI dataset.** The experiment is done by combining different components in the pipeline of DeepMapping2. All the results reported are from the 50th epoch of the training. The method without DeepMapping loss fails to converge so its result is not reported.

	Components			
DM loss	Batch org.	Con. loss	**T-ATE (m)**↓	**R-ATE (°)**↓
✓			1.88	4.72
✓	✓		1.65	2.07
✓		✓	1.88	4.70
	✓	✓	—	—
✓	✓	✓	**1.63**	**1.81**

5. Conclusion

Limitation. DeepMapping2 requires good loop closure to arrange batches. However, sometimes in large-scale environments, using a self-supervised loop closing method such as TF-VPR [44] is time-comsuming. One alternative is to use pretrained PointNetVLAD [41] with some geometric verification, such as ICP. Another option is to obtain loop closure from the pre-built map used in the warm-start step.

Also, although our convergence rate is improved by the consistency loss, the computation time of our method is still longer than other state-of-the-art methods like LeGO-LOAM. However, our framework allows *GPU-based parallel optimization* for speed boosting via distributed data parallel training [59] as in the supplementary.

Summary. DeepMapping2 adds loop closure-based batch organization and self-supervised consistency loss to DeepMapping, which can achieve excellent mapping performance in large-scale scenes. We believe our work can motivate more research in self-supervised LiDAR map optimization. The current pipeline can be used as a generic learning-based map optimization method in the back-end of almost any point cloud SLAM methods. Future works include adopting ideas like keyframe-based SLAM, and generalizing DeepMapping2 to multi-agent SLAM.

Acknowledgment. This work is supported by NSF CMMI-1932187, CNS-2121391, EEC-2036870, and IIS-2238968.

References

[1] Li Ding and Chen Feng. Deepmapping: Unsupervised map estimation from multiple point clouds. In *CVPR*, pages 8650–8659, 2019. 1, 2, 3, 5, 6, 7

[2] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183. IEEE, 2014. 1

[3] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-net: Towards learning based lidar localization for autonomous driving. In *CVPR*, pages 6389–6398, 2019. 1

[4] Matteo Palieri, Benjamin Morrell, Abhishek Thakur, Kamak Ebadi, Jeremy Nash, Arghya Chatterjee, Christoforos Kanellakis, Luca Carlone, Cataldo Guaragnella, and Ali-akbar Agha-Mohammadi. Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time. *IEEE Robotics and Automation Letters*, 6(2):421–428, 2020. 1

[5] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In *2013 IEEE international conference on robotics and automation*, pages 5182–5189. IEEE, 2013. 1

[6] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L. Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. LaMAR: Benchmarking Localization and Mapping for Augmented Reality. In *ECCV*, 2022. 1

[7] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *PAMI*, 29(6):1052–1067, 2007. 1

[8] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007. 1

[9] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011. 1, 3

[10] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *ECCV*, pages 834–849. Springer, 2014. 1

[11] Mathieu Labbé and François Michaud. Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics*, 36(2):416–446, 2019. 1

[12] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021. 1

[13] Lipu Zhou, Daniel Koppel, and Michael Kaess. Lidar slam with plane adjustment for indoor environment. *IEEE Robotics and Automation Letters*, 6(4):7073–7080, 2021. 1

[14] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, pages 1–9. Berkeley, CA, 2014. 2, 3

[15] Tixiao Shan and Brendan Englot. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4758–4765. IEEE, 2018. 2, 3, 6, 7

[16] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020. 2

[17] Kamak Ebadi, Yun Chang, Matteo Palieri, Alex Stephens, Alex Hatteland, Eric Heiden, Abhishek Thakur, Nobuhiro Funabiki, Benjamin Morrell, Sally Wood, et al. Lamp: Large-scale autonomous mapping and positioning for exploration of perceptually-degraded subterranean environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 80–86. IEEE, 2020. 2

[18] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. isam: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. 2

[19] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g^{2}o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011. 2

[20] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *ECCV*, pages 607–623, 2018. 2

[21] Chenghao Shi, Xieyuanli Chen, Kaihong Huang, Junhao Xiao, Huimin Lu, and Cyrill Stachniss. Keypoint matching for point cloud registration using multiplex dynamic graph attention networks. *IEEE Robotics and Automation Letters*, 6(4):8221–8228, 2021. 2

[22] Qing Li, Shaoyang Chen, Cheng Wang, Xin Li, Chenglu Wen, Ming Cheng, and Jonathan Li. Lo-net: Deep real-time lidar odometry. In *CVPR*, pages 8473–8482, 2019. 2

[23] Xieyuanli Chen, Thomas Läbe, Andres Milioto, Timo Röhling, Olga Vysotska, Alexandre Haag, Jens Behley, and Cyrill Stachniss. Overlapnet: Loop closing for lidar-based slam. *arXiv preprint arXiv:2105.11344*, 2021. 2

[24] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, pages 2938–2946, 2015. 2

[25] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: A globally optimal solution to 3d icp point-set registration. *PAMI*, 38(11):2241–2254, 2015. 2

[26] Ashutosh Singandhupe, Hung La, Trung Dung Ngo, and Van Ho. Registration of 3d point sets using correntropy similarity matrix. *arXiv preprint arXiv:2107.09725*, 2021. 2

[27] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *CVPR*, pages 3523–3532, 2019. 2, 3, 5

[28] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration. In *CVPR*, 2020. 2, 3, 6, 7

[29] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 2, 5, 6

[30] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016. 2, 6

[31] Ali Agha, Kyohei Otsu, Benjamin Morrell, David D Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021. 2, 6, 7

[32] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. Spie, 1992. 2, 3, 6

[33] Haggai Maron, Nadav Dym, Itay Kezurer, Shahar Kovalsky, and Yaron Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):1–12, 2016. 2

[34] Pascal Willy Theiler, Jan Dirk Wegner, and Konrad Schindler. Globally consistent registration of terrestrial laser scans via graph optimization. *ISPRS journal of photogrammetry and remote sensing*, 109:126–138, 2015. 3

[35] Georgios D Evangelidis, Dionyssos Kounades-Bastian, Radu Horaud, and Emmanouil Z Psarakis. A generative model for the joint registration of multiple point sets. In *ECCV*, pages 109–122. Springer, 2014. 3

[36] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. In *CVPR*, pages 864–872, 2015. 3

[37] Siyu Zhu, Runze Zhang, Lei Zhou, Tianwei Shen, Tian Fang, Ping Tan, and Long Quan. Very large-scale global sfm by distributed motion averaging. In *CVPR*, pages 4568–4577, 2018. 3

[38] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *CVPR*, pages 5556–5565, 2015. 3, 6, 7

[39] Zan Gojcic, Caifa Zhou, Jan D Wegner, Leonidas J Guibas, and Tolga Birdal. Learning multiview 3d point cloud registration. In *CVPR*, pages 1759–1769, 2020. 3

[40] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311. IEEE, 2010. 3

[41] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *CVPR*, pages 4470–4479, 2018. 3, 4, 8

[42] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, pages 652–660, 2017. 3

[43] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307, 2016. 3

[44] Chao Chen, Xinhao Liu, Xuchu Xu, Yiming Li, Li Ding, Ruoyu Wang, and Chen Feng. Self-supervised visual place recognition by mining temporal and feature neighborhoods. *arXiv preprint arXiv:2208.09315*, 2022. 3, 4, 8

[45] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. 3

[46] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017. 3

[47] Ruihao Li, Sen Wang, and Dongbing Gu. Deepslam: A robust monocular slam system with unsupervised deep learning. *IEEE Transactions on Industrial Electronics*, 68(4):3577–3587, 2020. 3

[48] Michael Bosse and Robert Zlot. Keypoint design and evaluation for place recognition in 2d lidar maps. *Robotics and Autonomous Systems*, 57(12):1211–1224, 2009. 3

[49] Robert Zlot and Michael Bosse. Efficient large-scale 3d mobile mapping and surface reconstruction of an underground mine. In *Field and service robotics*, pages 479–493. Springer, 2014. 3

[50] Bastian Steder, Giorgio Grisetti, and Wolfram Burgard. Robust place recognition for 3d range data based on point features. In *2010 IEEE International Conference on Robotics and Automation*, pages 1400–1405. IEEE, 2010. 5

[51] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 7244–7251. IEEE, 2018. 6

[52] Fan Lu, Guang Chen, Yinlong Liu, Lijun Zhang, Sanqing Qu, Shu Liu, and Rongqi Gu. Hregnet: A hierarchical network for large-scale outdoor lidar point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16014–16023, 2021. 6, 7

[53] Zheng Qin, Hao Yu, Changjian Wang, Yulan Guo, Yuxing Peng, and Kai Xu. Geometric transformer for fast and robust point cloud registration. In *CVPR*, pages 11143–11152, 2022. 6, 7

[54] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 2023. 6, 7

[55] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019. 6

[56] Athanasios Chalvatzaras, Ioannis Pratikakis, and Angelos A Amanatiadis. A survey on map-based localization techniques for autonomous vehicles. *IEEE Transactions on Intelligent Vehicles*, 2022. 6

[57] Tomáš Rouček, Martin Pecka, Petr Čížek, Tomáš Petříček, Jan Bayer, Vojtěch Šalanskỳ, Daniel Heřt, Matěj Petrlík, Tomáš Báča, Vojěch Spurnỳ, et al. Darpa subterranean challenge: Multi-robotic exploration of underground environments. In *International Conference on Modelling and Simulation for Autonomous Systems*, pages 274–290. Springer, 2019. 7

[58] Andrzej Reinke, Matteo Palieri, Benjamin Morrell, Yun Chang, Kamak Ebadi, Luca Carlone, and Ali-Akbar Agha-Mohammadi. Locus 2.0: Robust and computationally efficient lidar odometry for real-time 3d mapping. *IEEE Robotics and Automation Letters*, 2022. 7

[59] Shen Li, Yanli Zhao, Rohan Varma, Omkar Salpekar, Pieter Noordhuis, Teng Li, Adam Paszke, Jeff Smith, Brian Vaughan, Pritam Damania, et al. Pytorch distributed: Experiences on accelerating data parallel training. *arXiv preprint arXiv:2006.15704*, 2020. 8