DeepPointMap: Advancing LiDAR SLAM with Unified Neural Descriptors

Xiaze Zhang[1], Ziheng Ding[1], Qi Jing[1], Yuejie Zhang[1,3], Wenchao Ding[2,*], Rui Feng[1,2,3,*]

[1]School of Computer Science, Shanghai Key Laboratory of Intelligent Information Processing, Fudan University
[2]Academy for Engineering and Technology, Fudan University
[3]Shanghai Collaborative Innovation Center of Intelligent Visual Computing
{xzzhang22, zhding22, jingq22}@m.fudan.edu.cn, {zhangyuejie, dingwenchao, fengrui}@fudan.edu.cn

Abstract

Point clouds have shown significant potential in various domains, including Simultaneous Localization and Mapping (SLAM). However, existing approaches either rely on dense point clouds to achieve high localization accuracy or use generalized descriptors to reduce map size. Unfortunately, these two aspects seem to conflict with each other. To address this limitation, we propose a unified architecture, *DeepPointMap*, achieving excellent preference in both aspects. We utilize neural networks to extract highly representative and sparse neural descriptors from point clouds, enabling memory-efficient map representation and accurate multi-scale localization tasks (*e.g.*, odometry and loop-closure). Moreover, we showcase the versatility of our framework by extending it to more challenging multi-agent collaborative SLAM. The promising results obtained in these scenarios further emphasize the effectiveness and potential of our approach.

Introduction

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics and autonomous driving, which aims to reconstruct the map of the explored environment while simultaneously estimating the location of the agent within it. It plays a vital role in enabling autonomous agents to navigate and understand their surroundings. Point clouds have gained prominence as a powerful representation for capturing intricate 3D structures. Taking advantage of this, LiDAR SLAM has become a prominent approach for achieving accurate localization and generating high-quality maps.

Currently, several SLAM methods (Shan and Englot 2018; Chen et al. 2019; Pan et al. 2021; Vizzo et al. 2021; Dellenbach et al. 2022) based on hand-crafted features have demonstrated impressive performance on specific benchmarks. However, hand-crafted features are often either too sparse or over redundant, due to inefficacy in capturing semantic information. Sparse features may result in low-fidelity maps while dense features may be memory inefficient and problematic for large-scale reconstruction. Some methods (Qin et al. 2022; Yew and Lee 2022) use deep learning-based approaches to extract more compact features and achieve global registration. Although yielding promising results on point cloud registration benchmarks, they are hard to directly adapt to SLAM tasks since they can only deal with the registration task of single input size (*e.g.*, scan-level), and can be heavily impacted by cumulative errors. Furthermore, most of the existing methods adopt pure geometric pipeline to compose the SLAM system, *e.g.*, involving hand-crafted feature extraction, iterative-based odometry and projection-based loop-closure. The geometric pipeline may lose considerable information from sensor inputs, which makes a neural information processing pipeline a more promising direction.

To tackle these challenges, we present *DeepPointMap* (DPM), a novel deep learning-based LiDAR SLAM framework including two neural networks: (1) the DPM Encoder which extracts unified neural descriptors to represent the environment efficiently, and (2) the DPM Decoder which performs multi-scale matching and registration (*i.e.*, odometry and loop-closure) based on the aforementioned neural descriptors. As illustrated in Fig. 1, the highly compact neural descriptors provide a novel solution for environment encoding and facilitate high-fidelity mapping and information sharing (*e.g.*, in multi-agent cooperative SLAM), while the DPM Decoder renders a unified neural information processing pipeline for neural descriptors. Unlike other neural descriptor-based methods which can be only used in limited scenarios or specific SLAM components, our descriptor is used to accomplish several sub-tasks of SLAM in a unified manner, yielding exceptional localization accuracy, memory efficiency, map fidelity, and real-time processing.

In summary, we introduce an efficient LiDAR SLAM framework predominantly based on neural networks, offering a promising solution for advancing the field of LiDAR SLAM. It achieves a new *state-of-the-art* (SOTA) in localization accuracy, preserving a high-fidelity map reconstruction, with smaller memory consumption. These advantages can be further exploited in more challenging applications such as multi-agent cooperative SLAM which has limited communication bandwidth for feature sharing among agents. Our contributions can be summarized as:

- We propose using neural descriptors for online LiDAR SLAM. Compared to the traditional geometric descriptors, our neural descriptors serve as a compact map representation and facilitate a unified SLAM architecture, achieving desirable localization accuracy, memory efficiency, and map fidelity.

*Corresponding authors: Rui Feng and Wenchao Ding.

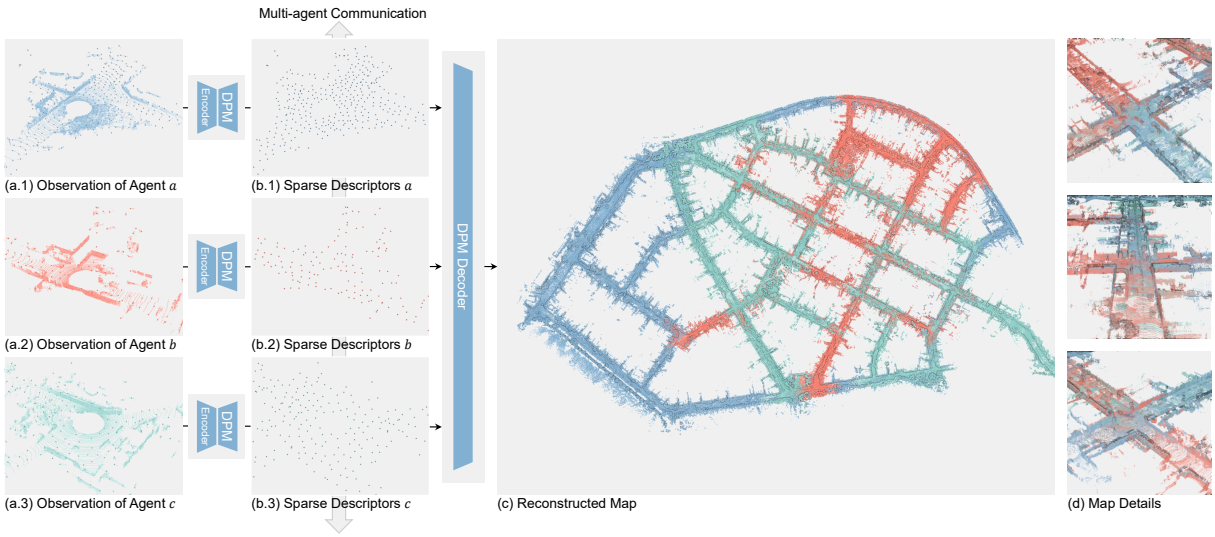

Figure 1: An Illustration of *DeepPointMap*. The agents collect point cloud (a) and extract it to sparse descriptors (b) locally. These descriptors are then gathered to reconstruct the complete map by multi-agent cooperative SLAM (c).

- We propose *DeepPointMap*[1], a novel unified learning-based framework for LiDAR SLAM by conducting multi-scale matching and registration based on the aforementioned neural descriptors. The proposed framework achieves SOTA performance in localization accuracy and memory consumption, with real-time inference speed.
- To highlight the advantages of our framework, we extend *DeepPointMap* to multi-agent cooperative SLAM task. Experimental results show that our approach demonstrates potential improvements in localization accuracy and mapping fidelity with constrained communication overhead.

Related Work

Map Reconstruction and Representation. Accurately and efficiently reconstructing the map of the environment is a crucial objective of LiDAR SLAM. Two of the most natural ways to represent the map are: (1) simply aligning the point clouds in global coordinates to construct a huge map-scale point cloud, or (2) storing the points explicitly into voxel-grid (Dellenbach et al. 2022; Vizzo et al. 2023). However, The first method requires a considerable number of points to adequately describe the map, leading to significant memory overhead. Conversely, the latter method is challenging to modify once the mapping process is completed. Some methods use hand-crafted descriptors such as curvature, density and normal to represent the environment. FLIRT (Tipaldi and Arras 2010) and FALKO (Kallasi, Rizzini, and Caselli 2016) introduced hand-based methods for detecting keypoints and calculating features in 2D range data. IMLS (Deschaud 2018) used *implicit moving least squares surfaces*, while SuMa (Behley and Stachniss 2018) and SuMa++ (Chen et al. 2019) represented the point cloud into surfels. PUMA (Vizzo et al. 2021) introduced a surface mesh representation that better captured the geometric appearance of objects in the scene. However, these hand-crafted features are weakly representable, and thus require complex registration algorithms to achieve accurate SLAM tasks. Recently, some methods use neural networks to extract implicit features from point clouds. For instance, SegMatch (Dubé et al. 2017) and its subsequent work SegMap (Dube et al. 2020) extracted condensed segment-level features as descriptors for localization and map representation. Other approaches, such as Liu et al. (2019); Cattaneo, Vaghi, and Valada (2022); Xiang et al. (2022); Arce et al. (2023), extract global features of point clouds and predict the transformation based on these features. However, these features tend to be either overly coarse or only applicable to specific pairs, limiting their universality for high-precision odometry. Moreover, NeRF (Xu et al. 2022) also shows credible ability to represent point cloud. However, it usually has difficulties in online updating and accuracy localization, which is essential to autonomous driving-related tasks. In contrast, our *DeepPointMap* utilizes sparse efficient neural descriptors to uniformly represent all point cloud scenarios with low memory overhead. These descriptors can be employed for various tasks in SLAM, including odometry and loop-closure, without any re-extraction.

Point Cloud Localization. Localization is a fundamental task in LiDAR SLAM, encompassing scan-level localization (*i.e.*, odometry) and map-level localization (*i.e.*, loop-closure). Achieving accurate localization heavily relies on estimating the transformation between two point clouds. Most traditional methods are based on the Iterative Closest Point (ICP) method (Besl and McKay 1992) and its variants (Low 2004; Censi 2008; Ramalingam and Taguchi 2013), which is a classic local registration algorithm with high computational complexity. To improve accuracy and efficiency, LOAM (Zhang and Singh 2014) and its derived algorithms (Shan and Englot 2018; Wang et al. 2021) detected edge and planar points as key points, which were used for more efficient numerical

[1]Source code: https://github.com/ZhangXiaze/DeepPointMap

optimization in alignment. Additionally, MULLS (Pan et al. 2021) classified key points into more specific classes to establish more accurate correspondences. However, these local registration methods are primarily limited to initial transformation guess (*e.g.*, from kinematic priors), which constrains their application to loop-closure and multi-agent SLAM without prior information. With the introduction of PointNet (Qi et al. 2017a), numerous deep learning methods for point cloud registration have emerged. Most of these methods (Yuan et al. 2020; Huang et al. 2021; Cao et al. 2021; Yew and Lee 2022; Qin et al. 2022, 2023) estimated correspondence in a global scope by training models to minimize feature distances between corresponding points. They then solved the transformation matrix using SVD (Arun, Huang, and Blostein 1987). DCP (Wang and Solomon 2019) applied this strategy to loop-closure, introducing learning-based methods to the SLAM field. However, these approaches are impractical for tackling both localization tasks in a unified manner as they can only handle single-scale inputs. In contrast, our approach, *DeepPointMap*, based on unified descriptors, excels at global registration tasks involving multi-scale inputs, such as maps composed of multiple independent scans. This enables direct handling of various localization tasks, including scan-to-map odometry and loop-closure without any priors or iterations.

Multi-agent Collaborative SLAM. Multi-agent SLAM requires collaboration among agents to improve mapping completeness and accuracy. DDF-SAM (Cunningham, Paluri, and Dellaert 2010) introduced an extended smoothing and mapping method for decentralized data fusion for multi-agents. Lazaro et al. (2013) proposed a method based on condensed measurements (Grisetti, Kümmerle, and Ni 2012) to reduce the information exchanged between agents. However, these approaches primarily focus on optimizing communication rules rather than reducing the size of descriptors. In contrast, our approach focuses on fundamentally reducing communication overhead through lightweight and precise descriptors, making it widely applicable to both single- and multi-agent SLAM scenarios.

Network Architecture

The key step of *DeepPointMap* is to extract the unified descriptors and utilize them in localization tasks. As described in Fig. 2, *DeepPointMap* consists of two parts: (a) **DPM Encoder**: a point cloud backbone such as PointNeXt (Qian et al. 2022), and (b) **DPM Decoder**: a Point-wise Attention Block and three heads as shown in Fig. 2 (c), (d) and (e).

DPM Encoder

DeepPointMap utilizes sparse descriptors that contain compressed semantic information to represent the complex point cloud for LiDAR SLAM. A point cloud $\widetilde{\mathcal{P}}$, which can be represented as a set of 3D points $\widetilde{\mathcal{P}} = \left\{\widetilde{\mathbf{p}}_1, \cdots, \widetilde{\mathbf{p}}_{\widetilde{N}}\right\} \in \mathbb{R}^{\widetilde{N}\times 3}$, is feeded into DPM Encoder to extract sparse descriptors. The DPM Encoder is essentially a PointNeXt (Qian et al. 2022) backbone, which is one of the most famous neural architectures for point cloud understanding improved from PointNet (Qi et al. 2017a) and PointNet++ (Qi et al. 2017b). The backbone samples sparse keypoints $\mathbf{p}_i^{\mathrm{xyz}} = \{x_i, y_i, z_i\} \in \mathbb{R}^3$ from dense point clouds $\widetilde{\mathcal{P}}$ and extract their neighboring geometric features $\mathbf{p}_i^{\mathrm{feat}} = \{f_i^1, \cdots, f_i^C\} \in \mathbb{R}^C$, where C is the dimension of feature vector. The final output of the backbone, denoted by $\mathcal{P} = \{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_M\} \in \mathbb{R}^{M\times(3+C)}$, is named as *unified descriptor cloud* to represent the dense point cloud.

It is easy to merge multiple descriptor clouds by a union operation $\bigcup_i \{\mathcal{P}_i\}$. In localization, we utilize this method to merge multiple scan-level descriptor clouds, thus forming the map-level descriptor clouds without any re-extraction.

DPM Decoder

DPM Decoder predicts the transformation between two aforementioned descriptor clouds, which consists of four parts: (1) Descriptor-wise Transformer Block, which is used for fusing deep correlation features between two input descriptor clouds, (2) Similarity Head, which aims to calculate the correspondence of descriptors between two clouds, (3) Offset Head, which predicts the relative positional offset between corresponding descriptors, enabling a more precise transformation estimation, and (4) Overlap Head, which predicts the probability of loop-closure occurred between two descriptor clouds, identifying potential loop closures in SLAM system.

Descriptor-wise Transformer Block. As shown in Fig. 2 (b), DPM Decoder takes a descriptor cloud pair $(\mathcal{P}, \mathcal{Q})$ as input and feeds them into the Descriptor-wise Transformer Block. Inspired by (Yew and Lee 2022), this block consists of three layers, each of them comprises three sub-layers in sequence: (1) Multi-Head Self-Attention layer that operates independently on $\mathcal{P}$ and $\mathcal{Q}$, (2) Multi-Head Cross-Attention layer that exchanges contextual information between $\mathcal{P}$ and $\mathcal{Q}$, and (3) MLP that enhances the fitting capability. Residual connections and normalization are applied on each layer to expedite convergence and stabilize the gradient. As the output of Descriptor-wise Transformer Block, the descriptor cloud pair is transformed into *correlated descriptors*, denoted as $\bar{\mathcal{P}} = \{\bar{\mathbf{p}}_1, \cdots, \bar{\mathbf{p}}_M\} \in \mathbb{R}^{M\times(3+C')}$ (*resp.*, $\bar{\mathcal{Q}}$). The position of the correlated descriptor is unchanged in this process.

Similarity Head. After an MLP (denoted as $\mathrm{Head}_{\mathrm{sim}}(\cdot)$), the correspondence matrix S is then calculated by descriptor-wise cosine-similarity (denoted as $\odot$) $S_{i,j} = \mathrm{Head}_{\mathrm{sim}}(\mathbf{p}_i^{\mathrm{feat}}) \odot \mathrm{Head}_{\mathrm{sim}}(\mathbf{q}_j^{\mathrm{feat}})$. The higher similarity indicates that descriptor pair $\mathbf{p}_i^{\mathrm{xyz}}$ and $\mathbf{q}_j^{\mathrm{xyz}}$ are closer in global coordinate. During inference, we select the top-k descriptor pairs with the highest value $S_{i,j}$ to form the correspondence $\sigma = \arg\mathrm{top}_k S$.

Offset Head. Due to the sparsity of descriptors, the descriptors are not likely to be perfectly aligned in global coordinates, which will introduce error to the transformation estimated by SVD even if the correspondence prediction were perfect. To address this issue, we use Offset Head to predict the relative positional offset $\delta_i \in \mathbb{R}^3$ between descriptor pairs $(\bar{\mathbf{p}}_i, \bar{\mathbf{q}}_i) \in \sigma$, enabling us to achieve more accurate alignment. This prediction is directional: for the direction of $\bar{\mathcal{P}} \rightarrow \bar{\mathcal{Q}}$, we aim to find the offset $\delta_i^{\rightarrow}$ (*resp.*, $\delta_i^{\leftarrow}$) that aligns $\bar{\mathbf{q}}_i$ to $\bar{\mathbf{p}}_i$ in $\bar{\mathcal{P}}$'s coordinate and vice versa. We concatenate (denoted as $\oplus$) the features of the paired descriptors and use an MLP to predict the precise offset $\delta_i = \mathrm{Head}_{\mathrm{offset}}(\bar{\mathbf{p}}_i^{\mathrm{feat}} \oplus \bar{\mathbf{q}}_i^{\mathrm{feat}})$. After that, a weighted-SVD (Arun, Huang, and Blostein 1987)

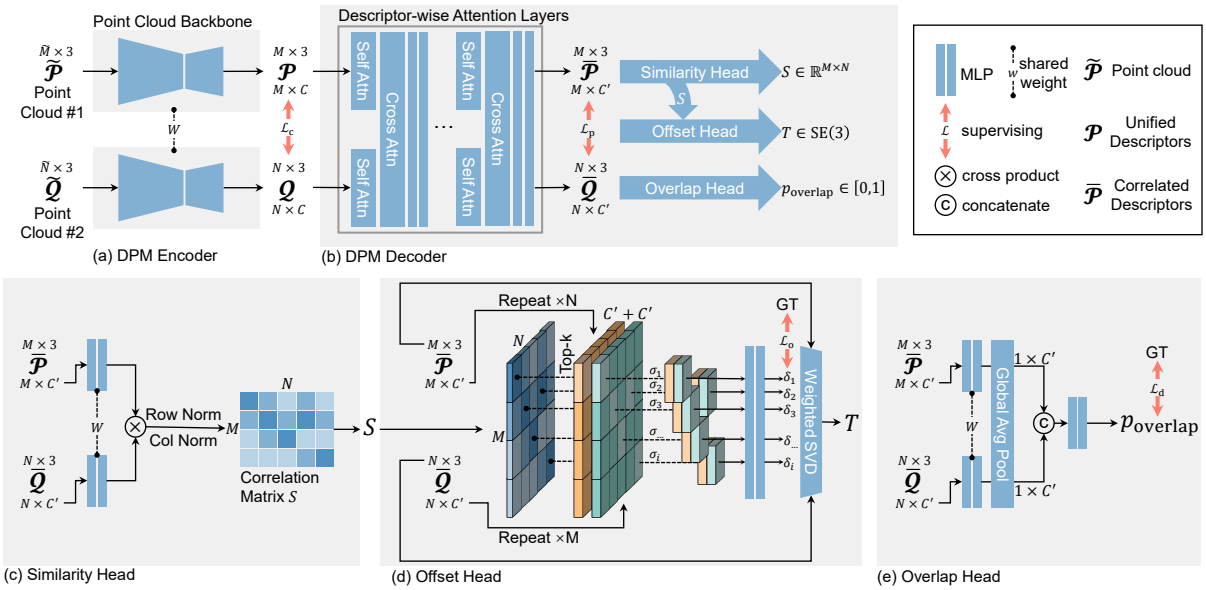

Figure 2: Overview of Network Architecture.

is applied to solve the transformation T:

$$\overrightarrow{\epsilon} = \sum_i \left\| T\left(\bar{\mathbf{p}}_i^{\mathrm{xyz}} + \overrightarrow{\delta}_i\right) - \bar{\mathbf{q}}_i^{\mathrm{xyz}} \right\|_2^2 \tag{1}$$

$$\overleftarrow{\epsilon} = \sum_j \left\| T\bar{\mathbf{p}}_j^{\mathrm{xyz}} - \left(\bar{\mathbf{q}}_j^{\mathrm{xyz}} + \overleftarrow{\delta}_j\right) \right\|_2^2 \tag{2}$$

$$T = \arg\min_T \left(\overrightarrow{\epsilon} + \overleftarrow{\epsilon}\right) \tag{3}$$

Overlap Head. The Overlap Head aims to predict whether the input descriptor cloud pair are overlapped, *i.e.*, their distance is shorter than a threshold $\varepsilon_{\mathrm{overlap}} = 20\mathrm{m}$. This head has a concise structure as shown in Fig. 2 (e). The first share-weighted MLP applies nonlinear transformations to correlated descriptors, followed by an average pooling to gather global features of each cloud. The other MLP predicts the probability p_{overlap} of overlap based on the concatenated global features.

Training

We jointly train the DPM Encoder and Decoder end-to-end with the following losses and strategies.

Pairing Loss. We adopt InfoNCE loss (Oord, Li, and Vinyals 2018) as Pairing Loss $\mathcal{L}_{\mathrm{p}}$ on the correlated descriptors. For each $\bar{\mathbf{p}}_i \in \bar{\mathcal{P}}$, the descriptor $\mathbf{q}_j$ is assigned as either (1) *positive* pair iff $j = \arg\min_j \left\|\mathbf{p}_i^{\mathrm{xyz}} - \mathbf{q}_j^{\mathrm{xyz}}\right\|_2^2$ and $\left\|\mathbf{p}_i^{\mathrm{xyz}} - \mathbf{q}_j^{\mathrm{xyz}}\right\|_2 \le \varepsilon_{\mathrm{positive}}$, or (2) *negative* pair. We denote these two pair sets as $\bar{\mathcal{Q}}_+$ and $\bar{\mathcal{Q}}_-$, respectively. The pairing loss for the correlated descriptors is:

$$\mathcal{L}_{\mathrm{p}} = \mathbb{E}_{\bar{\mathbf{p}}_i}\left[-\log\left(\frac{\sum_{\bar{\mathbf{q}}_j^{\mathrm{feat}} \in \bar{\mathcal{Q}}_+} \exp\left(\bar{\mathbf{p}}_i^{\mathrm{feat}} \odot \bar{\mathbf{q}}_j^{\mathrm{feat}}/\tau\right)}{\sum_{\bar{\mathbf{q}}_j^{\mathrm{feat}} \in \bar{\mathcal{Q}}} \exp\left(\bar{\mathbf{p}}_i^{\mathrm{feat}} \odot \bar{\mathbf{q}}_j^{\mathrm{feat}}/\tau\right)}\right)\right] \tag{4}$$

Coarse Pairing Loss. To ensure the unified descriptor clouds extracted by DPM Encoder can preliminary distinguish the correspondence of descriptors, we adopt the Coarse Pairing Loss $\mathcal{L}_{\mathrm{c}}$ similar to $\mathcal{L}_{\mathrm{p}}$ on unified descriptors $\mathcal{P}$ and $\mathcal{Q}$. We keep the definition of *positive* pair as above, but split the *negative* pair $\mathbf{q}_j$ as: (1) *negative* pair iff $\|\mathbf{p}_i - \mathbf{q}_j\|_2 > \varepsilon_{\mathrm{positive}}$, otherwise (2) *neutral* pair, which is not used in the calculation of loss. This strategy provides more inclusiveness for those pairs of descriptors that are close but not closest, because of the probability that these pairs will act as *positive* and *negative* pairs respectively at different times, thus causing ambiguity. We denote *positive*, *negative* and *neutral* pairs as $\mathcal{Q}_+$, $\mathcal{Q}_-$ and $\mathcal{Q}_\circ$. The Coarse Pairing Loss is defined as:

$$\mathcal{L}_{\mathrm{c}} = \mathbb{E}_{\mathbf{p}_i}\left[-\log\left(\frac{\sum_{\mathbf{q}_j \in \mathcal{Q}_+} \exp\left(\mathbf{p}_i^{\mathrm{feat}} \odot \mathbf{q}_j^{\mathrm{feat}}/\tau\right)}{\sum_{\mathbf{q}_j \in (\mathcal{Q}_+ \cup \mathcal{Q}_-)} \exp\left(\mathbf{p}_i^{\mathrm{feat}} \odot \mathbf{q}_j^{\mathrm{feat}}/\tau\right)}\right)\right] \tag{5}$$

Offset Loss. We adapt the Offset Loss to train Offset Head. Following the definition of three pair types above, we use both *positive* and *nurture* pairs to train the Offset Head to predict the offsets.

$$\mathcal{L}_{\mathrm{o}} = \mathbb{E}_{\mathbf{p}_i}\left[\frac{1}{\left|\bar{\mathcal{Q}}_+ \cup \bar{\mathcal{Q}}_\circ\right|} \sum_{\bar{\mathbf{q}}_j \in \bar{\mathcal{Q}}_+ \cup \bar{\mathcal{Q}}_\circ} \left\|\delta_{i,j} - \delta_{i,j}^*\right\|_\Sigma\right] \tag{6}$$

where $\delta_{i,j}$ represents the predicted offset between $(\bar{\mathbf{p}}_i, \bar{\mathbf{q}}_j)$, δ_i^* is the ground-truth and $\|\cdot\|_\Sigma$ is the Mahalanobis distance between them. We utilize both *positive* and *neutral* pairs during training to accelerate the convergence and improve the robustness. We use a different distance threshold $\varepsilon_{\mathrm{offset}}$ here to define *positive* and *negative* pairs.

Overlap Loss. We use Binary Cross Entropy (BCE) loss $\mathcal{L}_{\mathrm{d}}$ to train the Overlap Head. However, it will not be applied together with other losses.

Overall Loss. The directional (*i.e.*, $\mathcal{P} \to \mathcal{Q}$) loss of *DeepPointMap* is defined in Eq. (7). We use the average of loss in both directions, with the hyper-parameters of $\varepsilon_{\mathrm{positive}} = 1\mathrm{m}$, $\varepsilon_{\mathrm{offset}} = 2\mathrm{m}$, $\tau = 0.1$, $\lambda_1, \lambda_2, \lambda_3 = (1.0, 0.1, 1.0)$.

$$\mathcal{L}_{\mathrm{r}} = \lambda_1\mathcal{L}_{\mathrm{p}} + \lambda_2\mathcal{L}_{\mathrm{c}} + \lambda_3\mathcal{L}_{\mathrm{o}} \tag{7}$$

Data Augmentation. In some cases, objects nearby may obstruct the LiDAR's scan, which results in large point vacuum areas and may reduce localization accuracy. Laser2Vec (Nashed 2020) introduced an augmentation that restricts the FoV of 2D LiDAR sensor. However, this approach significantly alters the distribution of point cloud. Instead, we propose a novel data augmentation technique named *Random Occlusion* to simulate this situation. We first randomly generate some virtual boxes with random size and position. Each box introduces an occlusion effect by removing all points that pass through it. This augmentation can significantly improve the performance on occlusion cases.

Curriculum Learning. In our SLAM task, both scan- and map-level registrations are involved. However, regular training strategies (Yew and Lee 2022; Qin et al. 2023) only focus on scan-to-scan samples and neglect multi-scale registration. To address this, we adapt the curriculum learning strategy to progressively train *DeepPointMap* from simple to complex scenarios. In our training procedure, we gradually increase the scale of the descriptor clouds, which leads the model to gradually learn the capability of large-scale registration tasks. After that, we freeze all modules except Overlap Head and perform an individual training procedure to train Overlap Head. The scan pair is sampled with an equal probability of positive and negative samples.

DeepPointMap Framework

During inference, we compose the network into a full SLAM system, named *DeepPointMap*. We introduce it in two aspects: (1) Mapping, including map reconstruction and optimization, and (2) Localization, containing odometer and loop-closure. In addition, we further briefly introduce a multi-agent expansion of *DeepPointMap*.

Pose-Graph based Map. We adopt *Pose-Graph* to represent our reconstructed map, denoted as $\mathcal{M} = (V, E)$. Each observation $v_t \in V$ at time-step t contains the estimated pose T_t of the agent and the corresponding descriptor cloud $\mathcal{P}_t$. Meanwhile, the edge $e(v_i, v_j) \in E$ represents the positional relation between observations v_i, v_j. To ensure global consistency in the reconstructed map, we utilize pose graph optimization to globally optimize the pose estimates T_t of all observations v_t once the loop-closure (see description below) edge is inserted.

Keyframe Selection. The main limitation of pose-graph based map representation is that the agent seems to continuously insert new observations into the map, regardless of whether the location has been visited and already well-represented. To address this issue, we adopt a simple keyframe selection algorithm. For a new observation v_x, after estimating its pose T_x (described below), we accept v_x and insert it into $\mathcal{M}$ iff the distance of the nearest keyframe is greater than a dynamic threshold $\varepsilon_{\text{keyframe}}$, otherwise v_x is discarded. Threshold $\varepsilon_{\text{keyframe}}$ will increase if the current registration confidence is low and vice versa.

Odometer. The odometer estimates the current pose of the agent by continuously predicting the transformation $T_{x,t}$ between the current observation v_t and a previous one v_i. Unlike the temporal-based method, we first search the *K*-nearest-neighbors $N(v_{t-1})$ of v_{t-1} and find the closest keyframe v_i respect to the last known position. Then, T_t is estimated using DPM Decoder by registering the descriptor clouds $\mathcal{P}_t$ and $\mathcal{P}_i$. To obtain a more accurate pose estimation $T_{i,t}$, we apply a scan-to-map registration between $\mathcal{P}_t$ and $\bigcup N(v_i)$ as soon as the observation v_t is accepted (as described above). Finally, we insert an *odometer edge* $e(v_i, v_t)$ to the pose-graph.

Loop-Closure. The loop-closure procedure is composed of three stages: detection, registration, and verification. Given an observation v_t, we apply a spatial search to find the candidate observation list $V_c \subset V$ within 100m with respect to v_t. For each candidate $v_i \in V_c$, Overlap Head predicts their loop probability p_{overlap}. If the probability reaches a threshold $\varepsilon_{\text{loop}}$, the loop proposal is accepted. Then, we adapt a map-to-map registration using DPM Decoder on descriptor clouds $\bigcup N(v_t)$ and $\bigcup N(v_i)$, to estimate an accurate pose $T_{t,i}$. Finally, a *loop-edge* $e(v_i, v_t)$ is inserted into the pose-graph.

Multi-Agent Cooperative SLAM. In our multi-agent *DeepPointMap* framework, each agent maintains its own SLAM system and performs odometry and loop-closure locally. To ensure global consistency of trajectories and to merge the observations from each agent, we extend the *DeepPointMap* framework as follows: (1) Map Representation: Unless the agents' trajectories are spatially intersected, the map $\mathcal{M}$ is separated into multiple components $\mathcal{M}^1, \cdots \mathcal{M}^m$, where each component $\mathcal{M}^a = (V^a, E^a)$ represents a sub-map reconstructed by agent a. (2) Trajectory Merging: To detect trajectory intersections, we use the same strategy as single-agent loop detection, except assigning all observations from other agents as the candidate list (if no additional global position *e.g.*, GNSS is provided). When a trajectory intersection between observations v_i^a and v_j^b is verified, a *cross edge* $e(v_i^a, v_j^b)$ is inserted into the pose-graph that connects two components. Subsequently, a pose-graph optimization is applied to align trajectories.

Experimental Analysis

We conduct extensive experiments to evaluate our proposed *DeepPointMap* and current *state-of-the-art* algorithms on benchmark datasets.

Experimental Settings. We train the network following the strategy described above, with AdamW optimizer (Reddi, Kale, and Kumar 2019), initial learning rate $lr = 10^{-3}$, weight decay $wd = 10^{-4}$, cosine lr scheduler, on 6× RTX 3090 GPU. For all tasks, the network is trained for 12 epochs. When training Overlap Head, we decay lr and wd with a rate of 0.1. The model of last epoch is used for evaluation. We use the standard Pose-Graph Optimization provided in Open3D (Zhou, Park, and Koltun 2018) library without any modification.

Benchmarks. We conduct the experiments on the following four autonomous driving-oriented datasets: (1) SemanticKITTI (Behley et al. 2019, 2021) is a widely used benchmark dataset based on KITTI Vision Benchmark (Geiger, Lenz, and Urtasun 2012), consisting of 11 sequences (`00-10`) of LiDAR scans collected in various driving scenarios. We use the first 6 sequences as training-set. (2) KITTI-360 (Liao, Xie, and Geiger 2022) includes 11 large-

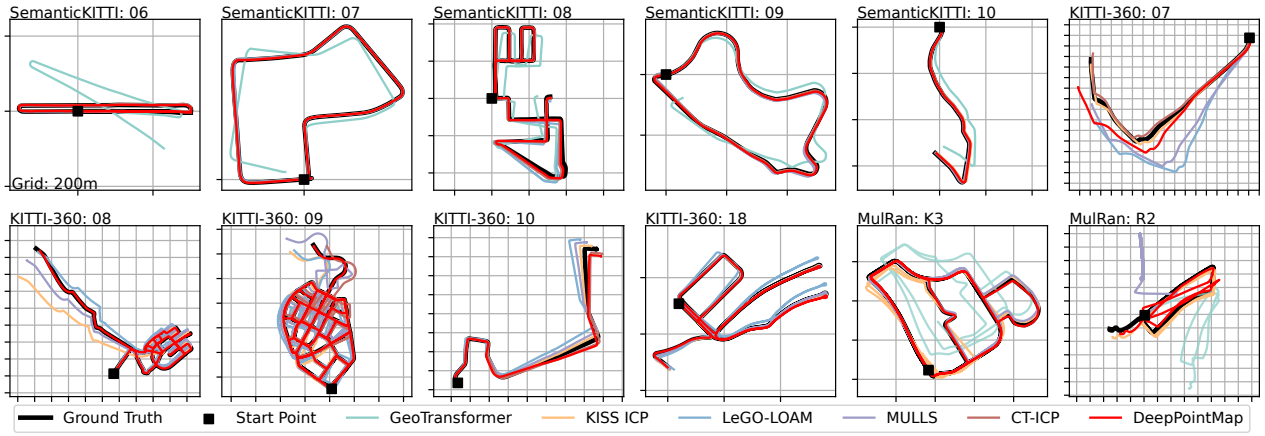

Figure 3: Trajectories Estimated with *DeepPointMap* and Comparison Methods on Testing Sequences.

	SemanticKITTI					KITTI-360					MulRan	
Method	06	07	08	09	10	07	08	09	10	18	K3	R2
LeGO-LOAM	0.88	0.67	8.84	1.95	1.37	82.84	32.79	7.36	26.56	2.80	×	×
SC-LeGO-LOAM	1.02	1.46	6.23	8.31	1.69	47.78	8.47	22.38	9.57	6.27	3.85	28.16
MULLS	**0.48**	0.38	4.16	1.99	0.97	47.25	8.24	93.88	11.99	1.52	6.63	×
CT-ICP	0.56	0.43	4.07	1.29	0.94	**14.43**	-	11.41	**7.11**	-	-	-
GeoTransformer[1]	24.38	8.71	22.68	22.14	16.36	×	×	75.93	×	34.79	71.95	×
KISS-ICP	0.61	0.35	3.58	1.32	**0.94**	20.51	26.24	24.00	8.75	2.22	12.85	27.16
DeepPointMap (ours)	0.92	**0.26**	**3.49**	**1.27**	1.28	93.77	**5.21**	**1.02**	10.46	**1.28**	**1.73**	**15.67**
DeepPointMap (ours)[2]	0.98	0.51	6.06	1.38	1.73	-	-	-	-	-	-	-

[1] GeoTransformer is a point cloud registration method. We estimate trajectory using scan-to-scan registration.
[2] Cross-dataset transfer experiment of *DeepPointMap*, trained on KITTI-360 and KITTI-Carla, tested on unseen SemanticKITTI.

Table 1: Localization Accuracy of *DeepPointMap* and Other SOTA Methods (APE↓).

scale sequences (`00`,`01`,`03-10` and `18`). We use the first 6 sequences as training-set. (3) MulRan (Kim et al. 2020) is a range dataset for urban place recognition and SLAM studies, which contains 12 sequences collected from 4 different environments. Following Kim, Choi, and Kim (2021), we use `KAIST03` (`K3`) and `Riverside02` (`R2`) sequences as testing-set, and use all 6 sequences collected in `Sejong` and `DCC` as training-set. (4) KITTI-Carla (Deschaud 2021) is a synthetic dataset with 7 sequences (`Town01-Town07`) generated on the CARLA simulation platform (Dosovitskiy et al. 2017), which provides noise-free data. We use all sequences in KITTI-Carla for training. Unless specified, all other sequences are used as testing sets. The split of training and testing set is based on the frame amount of *approx.* 6:4, without any manual picking. For quantitative evaluations, we adapted the stranded Absolute Pose Error (APE) to evaluate the global accuracy of the predicted trajectory, which is also used in KITTI-360 benchmark.

Localization Accuracy

This experiment is designed to show the localization accuracy of our method. To this end, we compare our method with 6 recent *state-of-the-art* Odometer and SLAM methods: KISS-ICP (Vizzo et al. 2023), LeGO-LOAM (Shan and Englot 2018) and its following work SC-LeGO-LOAM (Kim, Choi, and Kim 2021), MULLS (Pan et al. 2021), CT-ICP (Dellenbach et al. 2022), as well as a *state-of-the-art* point cloud registration method GeoTransformer (Qin et al. 2023). To make a fair comparison, we execute their open-source code on our platform and evaluate them using the same settings.

Fig. 3 presents the trajectories estimated by *DeepPointMap* *vs.* comparison methods on SemanticKITTI, KITTI-360 and MulRan. We observe that most methods achieve similar localization accuracy on SemanticKITTI, even for those approaches without global optimization. The performance difference between methods gradually emerges as the map scales up. KITTI-360 `09` is one of the longest (exceeding $10,000\text{m}$) sequences in our experiment, where most of the comparison methods fail to reconstruct a consistent global map. However, *DeepPointMap* successfully managed to reconstruct it with very slight distortion. For sequence `Riverside02`, some traditional methods struggled to produce accurate maps due to the scarcity of distinct reference objects in the scans. However, thanks to its superior feature extraction and representation capabilities, *DeepPointMap* successfully reconstructed these challenging scenes.

Table 1 reports the quantitative results, where *DeepPointMap* competes very favorably with its 6 peers on 12 sequences, by attaining 8 best results. Some results of CT-ICP are missing since CT-ICP can only run on its own provided data, while these missing sequences are not available to us. We also conduct additional experiments, as shown in the last line, to demonstrate the generalization ability on unseen data. We train *DeepPointMap* on KITTI-360 and KITTI-Carla, then directly evaluate its performance on SemanticKITTI, which still shows favorable localization accuracy.

Memory Efficiency

Efficient map representation is crucial for reconstructing large-scale maps. To evaluate the memory efficiency of *DeepPointMap*, we compared the reconstructed map size with several widely used methods: Original Point Cloud, Voxel Hashmap (Dellenbach et al. 2022) and Mesh (Vizzo et al. 2021). To achieve a fair comparison, we only count the size of data that is necessary for further localization. All data is stored in `float32` dtype. Fig. 4 shows the memory consumption for different map representations in two sequences.

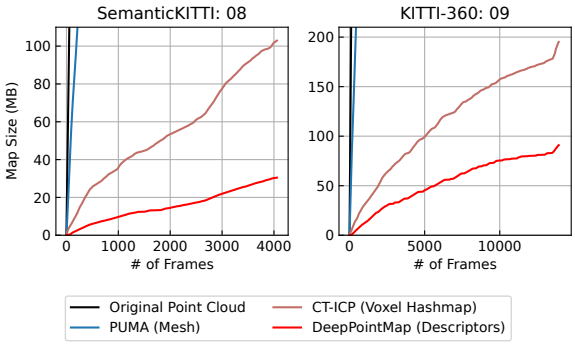

Figure 4: Map Size for Different Representations.

As shown in Fig. 4, our *DeepPointMap* manages to save up to 70% memory in SemanticKITTI `07` and about 50% memory in KITTI-360 `08`, compared to other methods.

Multi-agent Cooperative SLAM

To demonstrate the superiority of *DeepPointMap*, we extend our method to the multi-agent cooperative SLAM task and conduct an additional experiment. We select 3 sequences from SemanticKITTI and split them into 3 subsequences. The subsequences are then assigned to 3 individual agents to simulate the real-world multi-agent cooperative SLAM scenario. The resulting reconstructed point cloud is shown in Fig. 5, where the observed point clouds of each agent are marked with different colors. Agents successfully recognize the trajectory-intersect and perform multi-agent loop-closure, thus successfully reconstructing the environment.

Ablation Study

To gain a deeper understanding of *DeepPointMap*, we conduct several ablation studies to investigate the role of various components. All variants, as well as the baseline, are trained

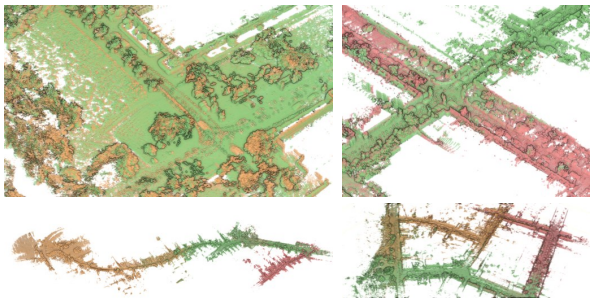

Figure 5: Multi-agent Cooperative Mapping and Localization.

Method	`08`	`09`	`10`
DeepPointMap (baseline)	4.7	2.53	1.35
w/o Offset Head	3.69	2.84	63.26
w/o Coarse Pairing Loss	3.92	2.58	2.27
w/o Curriculum & Augmentation	174.57	92.61	162.83

Table 2: Ablation Study on SemanticKITTI.

on SemanticKITTI, using the training set of the first 8 sequences and the testing set of the last 3 sequences.

We remove *Offset Head*, *Coarse Pairing Loss* and *Curriculum Learning & Random Occlusion* respectively to evaluate their importance. Our training strategy plays a critical role as it enables the model to learn the accurate and multi-scale representation of the environment. Offset Head was found to be necessary for achieving higher precision registration. It effectively mitigated the negative impact of descriptor sparsity and reduced the occurrence of trajectory prediction failures. Coarse Pairing Loss helps the model extract features at an earlier stage. This facilitated the DPM Decoder in capturing their fine-grained relationships and accelerated convergence, leading to improved performance of *DeepPointMap*.

Conclusions

We present *DeepPointMap*, a novel deep learning algorithm for LiDAR SLAM. This approach achieves accurate localization and reconstructs lightweight maps using uniform and efficient descriptors. Additionally, it demonstrates adaptability to multi-agent cooperative SLAM scenarios. Our method outperforms previous approaches on challenging benchmarks, particularly in large and complex urban scenes.

Limitation. Compared to traditional methods, neural network-based approaches require more precise labels *i.e.*, the pose of agent. However, in the field of autonomous driving, SLAM label quality often falls short, potentially impacting experimental results. Notably, our method showed weaker performance in certain distant countryside sequences (like KITTI-360 `07`, `10`, etc.), due to sparse reference objects and limited geometric information in the point cloud. This could explain why our model lagged behind those utilizing kinematic estimation (*e.g.*, Vizzo et al. (2023)).

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 62172101), the Science and Technology Commission of Shanghai Municipality (No.22DZ1100101 and No. 21511100500) and Shanghai Pujiang Program (23PJ1400900).

References

Arce, J.; Vödisch, N.; Cattaneo, D.; Burgard, W.; and Valada, A. 2023. PADLoC: LiDAR-Based Deep Loop Closure Detection and Registration Using Panoptic Attention. *IEEE Robotics and Automation Letters*, 8(3): 1319–1326.

Arun, K. S.; Huang, T. S.; and Blostein, S. D. 1987. Least-squares fitting of two 3-D point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5): 698–700.

Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Gall, J.; and Stachniss, C. 2021. Towards 3D LiDAR-based semantic scene understanding of 3D point cloud sequences: The SemanticKITTI Dataset. *The International Journal of Robotics Research*, 40(8-9): 959–967.

Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; and Gall, J. 2019. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9297–9307.

Behley, J.; and Stachniss, C. 2018. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Robotics: Science and Systems*, volume 2018, 59.

Besl, P. J.; and McKay, N. D. 1992. Method for registration of 3-D shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, 586–606. Spie.

Cao, A.-Q.; Puy, G.; Boulch, A.; and Marlet, R. 2021. PCAM: Product of cross-attention matrices for rigid registration of point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13229–13238.

Cattaneo, D.; Vaghi, M.; and Valada, A. 2022. Lcdnet: Deep loop closure detection and point cloud registration for lidar slam. *IEEE Transactions on Robotics*, 38(4): 2074–2093.

Censi, A. 2008. An ICP variant using a point-to-line metric. In *2008 IEEE International Conference on Robotics and Automation*, 19–25. Ieee.

Chen, X.; Milioto, A.; Palazzolo, E.; Giguere, P.; Behley, J.; and Stachniss, C. 2019. Suma++: Efficient lidar-based semantic slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4530–4537. IEEE.

Cunningham, A.; Paluri, M.; and Dellaert, F. 2010. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 3025–3030. IEEE.

Dellenbach, P.; Deschaud, J.-E.; Jacquet, B.; and Goulette, F. 2022. CT-ICP: Real-time elastic LiDAR odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, 5580–5586. IEEE.

Deschaud, J.-E. 2018. IMLS-SLAM: Scan-to-model matching based on 3D data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2480–2485. IEEE.

Deschaud, J.-E. 2021. KITTI-CARLA: a KITTI-like dataset generated by CARLA Simulator. *arXiv preprint arXiv:2109.00892*.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, 1–16.

Dube, R.; Cramariuc, A.; Dugas, D.; Sommer, H.; Dymczyk, M.; Nieto, J.; Siegwart, R.; and Cadena, C. 2020. SegMap: Segment-based mapping and localization using data-driven descriptors. *The International Journal of Robotics Research*, 39(2-3): 339–355.

Dubé, R.; Dugas, D.; Stumm, E.; Nieto, J.; Siegwart, R.; and Cadena, C. 2017. Segmatch: Segment based place recognition in 3d point clouds. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 5266–5272. IEEE.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, 3354–3361. IEEE.

Grisetti, G.; Kümmerle, R.; and Ni, K. 2012. Robust optimization of factor graphs by using condensed measurements. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 581–588. IEEE.

Huang, S.; Gojcic, Z.; Usvyatsov, M.; Wieser, A.; and Schindler, K. 2021. Predator: Registration of 3d point clouds with low overlap. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, 4267–4276.

Kallasi, F.; Rizzini, D. L.; and Caselli, S. 2016. Fast keypoint features from laser scanner for robot localization and mapping. *IEEE Robotics and Automation Letters*, 1(1): 176–183.

Kim, G.; Choi, S.; and Kim, A. 2021. Scan context++: Structural place recognition robust to rotation and lateral variations in urban environments. *IEEE Transactions on Robotics*, 38(3): 1856–1874.

Kim, G.; Park, Y. S.; Cho, Y.; Jeong, J.; and Kim, A. 2020. MulRan: Multimodal Range Dataset for Urban Place Recognition. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. Paris.

Lazaro, M. T.; Paz, L. M.; Pinies, P.; Castellanos, J. A.; and Grisetti, G. 2013. Multi-robot SLAM using condensed measurements. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1069–1076. IEEE.

Liao, Y.; Xie, J.; and Geiger, A. 2022. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Liu, Z.; Suo, C.; Zhou, S.; Xu, F.; Wei, H.; Chen, W.; Wang, H.; Liang, X.; and Liu, Y.-H. 2019. Seqlpd: Sequence matching enhanced loop-closure detection based on large-scale point cloud description for self-driving vehicles. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1218–1223. IEEE.

Low, K.-L. 2004. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10): 1–3.

Nashed, S. B. 2020. Laser2Vec: Similarity-based Retrieval for Robotic Perception Data. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 10657–10662. IEEE.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Pan, Y.; Xiao, P.; He, Y.; Shao, Z.; and Li, Z. 2021. MULLS: Versatile LiDAR SLAM via multi-metric linear least square. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 11633–11640. IEEE.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 652–660.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30.

Qian, G.; Li, Y.; Peng, H.; Mai, J.; Hammoud, H.; Elhoseiny, M.; and Ghanem, B. 2022. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35: 23192–23204.

Qin, Z.; Yu, H.; Wang, C.; Guo, Y.; Peng, Y.; Ilic, S.; Hu, D.; and Xu, K. 2023. GeoTransformer: Fast and Robust Point Cloud Registration With Geometric Transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Qin, Z.; Yu, H.; Wang, C.; Guo, Y.; Peng, Y.; and Xu, K. 2022. Geometric transformer for fast and robust point cloud registration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11143–11152.

Ramalingam, S.; and Taguchi, Y. 2013. A theory of minimal 3D point to 3D plane registration and its generalization. *International journal of computer vision*, 102(1): 73–90.

Reddi, S. J.; Kale, S.; and Kumar, S. 2019. On the Convergence of Adam and Beyond.

Shan, T.; and Englot, B. 2018. Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4758–4765. IEEE.

Tipaldi, G. D.; and Arras, K. O. 2010. Flirt-interest regions for 2d range data. In *2010 IEEE International Conference on Robotics and Automation*, 3616–3622. IEEE.

Vizzo, I.; Chen, X.; Chebrolu, N.; Behley, J.; and Stachniss, C. 2021. Poisson surface reconstruction for LiDAR odometry and mapping. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 5624–5630. IEEE.

Vizzo, I.; Guadagnino, T.; Mersch, B.; Wiesmann, L.; Behley, J.; and Stachniss, C. 2023. KISS-ICP: In Defense of Point-to-Point ICP Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE Robotics and Automation Letters*.

Wang, H.; Wang, C.; Chen, C.-L.; and Xie, L. 2021. F-loam: Fast lidar odometry and mapping. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4390–4396. IEEE.

Wang, Y.; and Solomon, J. M. 2019. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF international conference on computer vision*, 3523–3532.

Xiang, H.; Zhu, X.; Shi, W.; Fan, W.; Chen, P.; and Bao, S. 2022. DeLightLCD: A Deep and Lightweight Network for Loop Closure Detection in LiDAR SLAM. *IEEE Sensors Journal*, 22(21): 20761–20772.

Xu, Q.; Xu, Z.; Philip, J.; Bi, S.; Shu, Z.; Sunkavalli, K.; and Neumann, U. 2022. Point-nerf: Point-based neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 5438–5448.

Yew, Z. J.; and Lee, G. H. 2022. Regtr: End-to-end point cloud correspondences with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 6677–6686.

Yuan, W.; Eckart, B.; Kim, K.; Jampani, V.; Fox, D.; and Kautz, J. 2020. Deepgmr: Learning latent gaussian mixture models for registration. In *European conference on computer vision*, 733–750. Springer.

Zhang, J.; and Singh, S. 2014. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 1–9. Berkeley, CA.

Zhou, Q.-Y.; Park, J.; and Koltun, V. 2018. Open3D: A modern library for 3D data processing. *arXiv preprint arXiv:1801.09847*.