DCPCR: Deep Compressed Point Cloud Registration in Large-Scale Outdoor Environments

Louis Wiesmann Tiziano Guadagnino Ignacio Vizzo Giorgio Grisetti Jens Behley Cyrill Stachniss

***Abstract*—Reliable and accurate registration of point clouds is a challenging problem in robotics as well as in the domain of autonomous driving. In this paper, we address the task of aligning point clouds with low overlap, containing moving objects, and without prior information about the initial guess. We enhance classical ICP-based registration with neural feature-based matching to reliably find point correspondences. Our novel 3D convolutional and attention-based network is trained in an end-to-end fashion to learn features, which are well suited for matching and for rating the quality of the point correspondences. By utilizing a compression encoder, we can directly operate on a compressed map representation, making our approach well suited for operation under memory constraints. We evaluate our approach on point clouds obtained at completely different points in time, showing that our approach is able to register point clouds even under those challenging conditions reliably. The implementation of our approach and the preprocessed data can be accessed at https://github.com/PRBonn/DCPCR.**

***Index Terms*—Deep Learning Methods; Localization; SLAM**

I. Introduction

POINT cloud registration is key for many robotic applications such as map matching, pose tracking, loop closure, or simultaneous localization and mapping (SLAM). Outdoor point clouds, as they are commonly used in the domain of autonomous vehicles, are typically large-scale, contain dynamics, and may vary a lot between different recording times, and therefore require robust and reliable methods. Additionally, the vast amount of points, which can accumulate easily hundred of thousands to millions of points within seconds, pose a particular challenge for processing and storage. In this work, we tackle the problem of registering partially overlapping point clouds acquired at different points in time and with high uncertainty regarding the initial guess. A common approach for point cloud registration is the iterative closest point (ICP) [4] algorithm, which usually only converges to the correct solution with a good initial alignment. Global registration approaches tackle exactly this problem with the goal to increase the robustness when facing the alignment without any prior pose information [36], [37]. Learning-based approaches provide the opportunity to learn distinct features to match more reliably [19], [31], [38], [39], [41] or even to bypass correspondence matching by directly estimating a pose [1], [24].

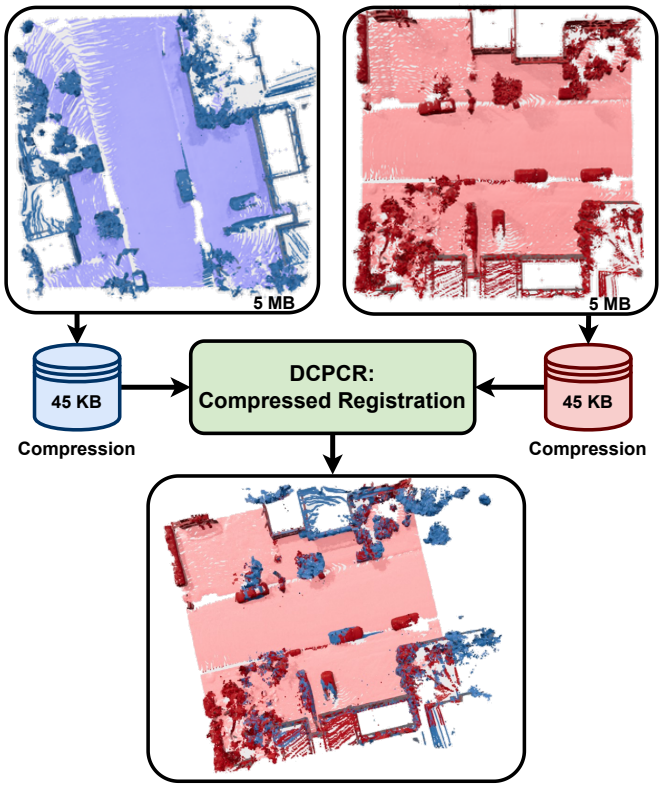


Fig. 1: In this work, we propose a method to directly register point clouds on a compressed representation. For better visualization we show the estimated transformation on the input point clouds, rather than the sparse compressed representation.

A second problem, which especially arises in the automotive domain is the massive memory demand for storing raw point clouds. A single LiDAR sensor acquires millions of points per second, which alone can be challenging for onboard storage. When data for downstream tasks such as map matching or pose tracking have to be stored, a memory-efficient representation is needed. Compression can be one solution to deal with this challenge. It is, however, not entirely clear how the performance of downstream tasks deteriorates when operating on the maps after lossy decompression. Comparably few approaches try to directly operate on compressed maps to alleviate the large memory demands and to avoid decompression [32], [40].

The main contribution of this paper is an approach to perform global point cloud registration in the context of outdoor environments and which operates on a compressed representation, see Fig. 1 for illustration. Specifically, we deal with large non-overlapping areas, moving or moved

Manuscript received: February 23, 2022; Accepted: April 12, 2022. This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments.

This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob and by the European Union's Horizon 2020 research and innovation programme under grant agreement No 101017008 (Harmony).

L. Wiesmann, T. Guadagnino, I. Vizzo, J. Behley, and C. Stachniss are with the University of Bonn, Germany. C. Stachniss is also with the Department of Engineering Science at the University of Oxford, UK. T. Guadagnino and G. Grisetti are with La Sapienza University of Rome, Italy.

Digital Object Identifier (DOI): see top of this page.

objects, and having no prior information about the initial pose. We utilize the compact representation of our prior point cloud compression network [33] to directly operate on the compact representation and therefore bypass decompression and reduce compute when compression is needed. We build upon the classical SVD-based pose estimation using a feature matching with soft assignments. For that, we propose a network architecture that consists of a convolutional backbone with a Transformer [29] head to produce distinct features, which can directly be optimized for point cloud registration. Additionally, the network learns to estimate the quality of the correspondences to focus on points that are well suited for the registration. We investigate the performance of our approach when targeting memory-constrained registration, as well as point cloud registration in the classical setup without compression. In sum, we show that our proposed network is able to reliably register point clouds such as commonly encountered in urban outdoor environments with and without compression.

II. Related Work

Point cloud registration has a long history in robotics, computer graphics, and computer vision [21]. In the following, we will distinguish between local registration, which typically requires a good initial alignment, and global registration where the goal is to register point clouds that are initially poorly aligned.

A. Local Registration

The most common approach for aligning two point clouds is ICP [4] with its variants [5], [23], [25]. Here, the main challenge is to find the correct correspondences. Looking only at spatially close points often fails when the initial guess is too far from the correct transformation. Geometric [9] or photometric [13], [15], [20] features are often used to find suitable correspondences and to resolve ambiguities. Projective data association [3], [23], [26] can be applied to speed up the correspondence search, which usually takes a relatively large proportion of the computation time. Point-to-plane [7] and GICP [25] on the other hand try to relax the assumption of point correspondences for faster convergence and more precise results.

Outdoor environments often change, and therefore assuming to have a lot of one-to-one correspondences does not necessarily hold. Robust optimization [6], [10], [12] or a careful correspondence selection [23] is often needed to overcome this issue.

Deep learning-based approaches have the advantage of computing more distinct and sophisticated features than their handcrafted counterparts. Many learning-based approaches [19], [31], [39] rely on the SVD-based closed-form solution and soft assignments for end-to-end learning. Other approaches [1], [24] try to estimate the transformation directly in the network based on global features and thereby bypass the need to find suitable correspondences. In general, local registration methods require a quite good initial transformation to converge to the desired solution.

B. Global Registration

Global registration methods try to estimate the pose between the point clouds even when the initial guess is quite far away from the correct transformation. RANSAC [8] provides the opportunity to deal with large transformations as well as outliers by sampling correspondences combined with a large number of repetitions. Branch-and-Bound (BB) methods [36], [37] generate multiple hypotheses in an evolutional manner and prune the search space. The biggest disadvantage of those methods is typically the high computation time.

In structural distinct environments, convolutional neural networks can learn feature representations that are well suited for matching [38], [41]. A contrastive loss [41] allows for self-supervised training to enforce similar features for similar areas, while dissimilar features to all other areas at the same time.

When having the information about the transformation between the point clouds at training time, one can directly supervise the pose to get features that are well suited for the matching [19], [39]. Instead of directly estimating the transformation in one shot, or iteratively, recently some approaches [17], [30] propose a coarse to fine registration in a hierarchical manner. DCP [31] is a supervised registration approach, and closely related to our method. It utilizes a graph convolutional neural network and a transformer head to compute features, which are later used to obtain soft correspondences for the registration. DCP was designed and successfully applied on synthetic data with 1,024 points and Gaussian noise, but cannot deal efficiently with a large number of points, non-overlapping areas, or dynamics as they are common in the automotive field. We overcome the scalability problem by integrating a compression network [33], which subsamples the points clouds significantly but preserves the local information in the feature representation. To deal with points that do not have a corresponding point in the other cloud, we propose a way to weigh the correspondences. By this, we also do not rely on RANSAC to evaluate the quality of the correspondences [41].

III. Feature-based Point Cloud Registration

For the alignment of two point clouds, we follow the classical paradigm of first finding point correspondences, which are then used to estimate the relative transformation. While in the classical ICP, correspondences are determined via geometric neighborhood, we follow a feature-based approach. The transformation consisting of rotation and translation is estimated using the closed-form solution [14].

In the following, we will first explain in Sec. III-A how to incorporate the soft-assignments and the feature-based matching of the attention mechanism [29] into the Kabsch algorithm [14]. Afterward, we propose our network architecture to compute the features for the point matching (see Fig. 2). In Sec. III-C, we will address the problem of ambiguities and errors in the matching by introducing a weight for each correspondence.

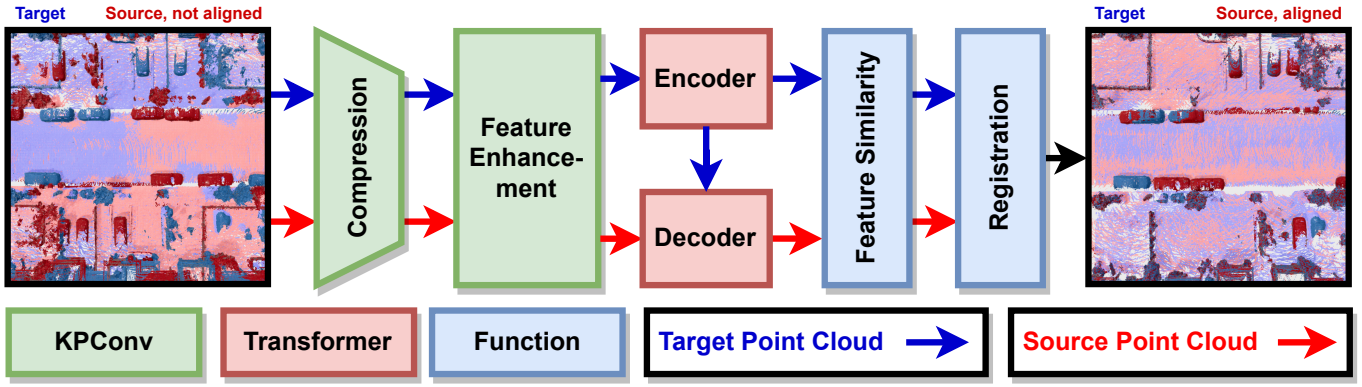


Fig. 2: Given two unaligned point clouds (target: blue, source: red), we first use a compression encoder to compute local features and to subsample the point clouds. A feature enhancement block increases the receptive field to create more distinct features which serve as input to the transformer heads for global aggregation. The resulting features are used for feature-based correspondence matching. Finally, the soft assigned source points are aligned to the target point cloud with a point-to-point registration.

A. Deep Point Cloud Registration

Given are two point clouds P_s and P_z in different coordinate frames, where each point cloud P_i consists of N_i coordinates $\mathsf{X}_i \in \mathbb{R}^{N_i \times 3}$ and their corresponding D_f dimensional features $\mathsf{F}_i \in \mathbb{R}^{N_i \times D_f}$. We want to estimate the transformation T_s^z from the source frame s to the target frame z, which aligns both point clouds. When having a correspondence matrix $\mathsf{W} \in \mathbb{R}^{N_z \times N_s}$, we can compute the corresponding points $\tilde{\mathsf{X}}_z$ of the target points by

$$\tilde{\mathsf{X}}_z = \mathsf{W}\mathsf{X}_s. \tag{1}$$

A row in the W matrix represents the corresponding point in $\tilde{\mathsf{X}}_z$ by a linear combination of the source points. In case of one-to-one correspondences, this is a row with 0 everywhere but a 1 at the index of the corresponding source point. The rotation $\mathsf{R} \in \mathbb{R}^{3\times3}$ and translation $\boldsymbol{t} \in \mathbb{R}^3$ of the transformation T_s^z can then be estimated by the Kabsch algorithm [14] using SVD, as follows:

$$\mathsf{C} = (\mathsf{X}_z^{*T}\tilde{\mathsf{X}}_z^*) \overset{\text{SVD}}{=} \mathsf{U}\mathsf{D}\mathsf{H}^T \tag{2}$$

$$\mathsf{R} = \mathsf{U}\mathsf{H}^T \tag{3}$$

$$\mathbf{t} = \mathbf{x}_z^* - \mathsf{R}\tilde{\mathbf{x}}_z^*, \tag{4}$$

where X_i^* denotes a point cloud shifted by its mean $\mathbf{x}_i^*$ such that the point cloud is centered around the origin. When dealing with uncertainties and partially overlapping point clouds one can incorporate a weight $\mathbf{w} \in \mathbb{R}^{N_z}$ for each correspondence by computing weighted means in Eq. (4) and a weighted cross-covariance C in Eq. (2).

Since the correspondences W are usually unknown, one estimates them by, e.g., nearest neighbors, feature-based matching, etc. as in classical ICP methods [23]. In this work, we use feature-based matching by utilizing the attention mechanism as used in the Transformer [29]. It is defined by three matrices, namely the queries Q, keys K and values V, where the attention A is computed by

$$\mathsf{A} = \text{attn}(\mathsf{Q},\mathsf{K},\mathsf{V}) = \text{softmax}\left(\frac{\mathsf{Q}\mathsf{K}^T}{\sqrt{D_f}}\right)\mathsf{V}. \tag{5}$$

When choosing the target features as queries $\mathsf{Q} := \mathsf{F}_z$, the source features as keys $\mathsf{K} := \mathsf{F}_s$ and the source coordinates as values $\mathsf{V} := \mathsf{X}_s$, we can rewrite Eq. (5) resulting in the desired correspondeces in Eq. (1) with $\mathsf{A} := \tilde{\mathsf{X}}_z$

$$\tilde{\mathsf{X}}_z = \text{softmax}\left(\frac{\mathsf{F}_z\mathsf{F}_s^T}{\sqrt{D_f}}\right)\mathsf{X}_s = \mathsf{W}\mathsf{X}_s. \tag{6}$$

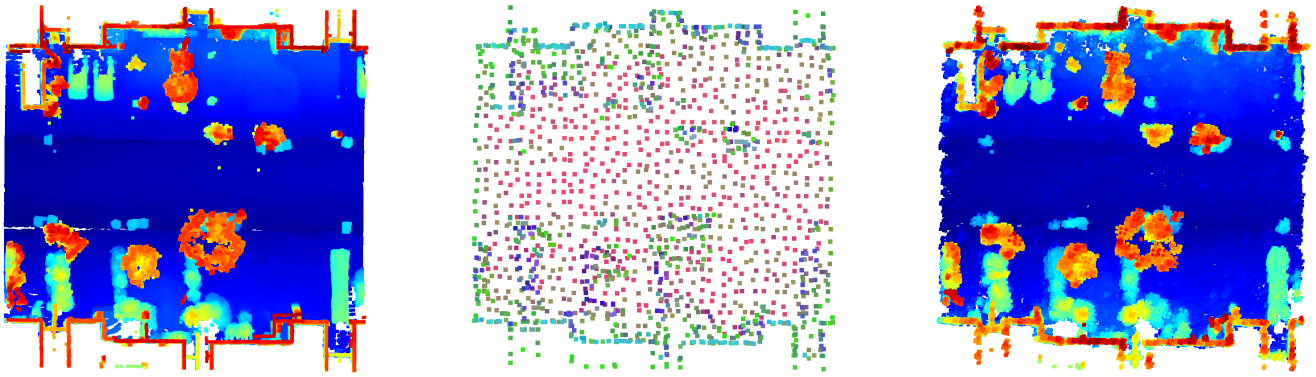
In other words, we are computing a soft assignment between the target points P_z and the source points P_s, based on the cosine similarity between their features F_z and F_s. In the following, we present our network architecture for computing the point features.

B. Feature Generation

Our network architecture consists of three parts, namely a compression encoder for memory and compute efficiency, a convolutional block to increase the receptive field, and a transformer head for global feature aggregation.

1) Compression Network: Point clouds obtained with modern LiDAR scanners easily contain multiple hundred of thousands of points, which is for most networks infeasible to process. Our previously described attention-based registration relies on cross attention between the source and target point cloud and therefore grows quadratically with the number of points. To overcome this issue, we use our previously published compression network [33], which substantially reduces the number of points and provides locally-aware features. We use the memory-efficient representation produced from the convolutional encoder as input for our network, which allows us to run the full registration procedure on the compressed point clouds without the need for decompression. The decoder of Wiesmann et al. [33] can later still be used to recover dense point clouds for other downstream tasks if needed.

Additionally, we use a reduced PointNet [22] to transform the compressed point representation to a localization-specific feature space, which is better suited for matching than for reconstruction. The PointNet consists of a multi-layer perceptron (MLP) with two hidden dimensions plus an initial TNet to create a registration-specific representation.



(a) Input Point Cloud (b) Compressed Point Cloud (c) Decompressed Point Cloud

Fig. 3: Visualization of the different point cloud representation in different stages of the network. (a) represents the original data like it is used as input for our network and the baselines for the registration without memory constraints. In (b) the sparse compressed representation is visualized, colorized by the feature space. In (c) one can see the decompressed point cloud, used for the baselines to register under memory-constrained conditions. The color in (a) and (b) represents the height of the points for visualization purposes.

2) Feature Enhancement Network: The features from the compression network contain local information about the close neighborhood of the points but lack broader context, which might be useful to create more distinct features for better matching and resolving ambiguities. We use additional B_k KPConv [27] blocks in a ResNet-like fashion [11]. KPConv is a sparse convolution that aggregates the information from the points within a radius r. The convolutional weights are defined on kernel points, rather than a grid structure, and therefore avoids voxelization. We are not subsampling further in those blocks to keep the point resolution and thus increase only the receptive field.

3) Transformer: We utilize a small Transformer head [29] consisting of an encoder and decoder. The encoder operates on the target point features F_z and utilizes multihead self-attention for global feature aggregation.

$$\mathsf{F}_z^t = \text{MultiHead}(\mathsf{Q} := \mathsf{F}_z, \mathsf{K} := \mathsf{F}_z, \mathsf{V} := \mathsf{F}_z) \text{ with} \quad (7)$$

$$\text{MultiHead}(\mathsf{Q}, \mathsf{K}, \mathsf{V}) = [\text{attn}(\mathsf{Q}\mathsf{W}_j^Q, \mathsf{K}\mathsf{W}_j^K, \mathsf{V}\mathsf{W}_j^V)]\mathsf{W}^O \quad j \in \{1, N_h\}, \quad (8)$$

where W^Q, W^K, and W^V are projection matrices of the queries, keys, and values, respectively. W^O projects the N_h concatenated heads to the desired feature dimension.

The decoder uses multihead cross-attention between the source and the target, as follows:

$$\mathsf{F}_s^t = \text{MultiHead}(\mathsf{Q} := \mathsf{F}_s, \mathsf{K} := \mathsf{F}_z^t, \mathsf{V} := \mathsf{F}_z^t). \quad (9)$$

This decoder transforms the features of the source points into the feature space of the target to increase the feature similarity and therefore should lead to better matching. We do not explicitly add any positional encoding since our generated features already contain some positional information and we do not want to add any bias to the Transformer, which could increase the influence of the positions. Defined by our task, we know that the point coordinates are in different frames therefore the comparability on those is limited. The final features F_z^t and F_s^t are used for the correspondence matching in Eq. (6).

C. Weighting Scheme

Finding the correct correspondences is a challenging task. Often, the matching is ambiguous in regions that are not very descriptive, e.g., finding point correspondences on large homogenous regions like planes. Additionally, when dealing with point clouds with partial overlap or that contain dynamic objects, some points do not even have a corresponding point in the other point cloud. To deal with these ambiguities and uncertainties, we additionally estimate a weight $\mathbf{w} \in \mathbb{R}_z^N$ for each correspondence. We compute the weights based on the correspondence matrix W and thus based on the feature similarities between the target and the source points. For each correspondence, we feed the top k_w features into an MLP: $\mathbb{R}^{N \times k_w} \mapsto \mathbb{R}^{N \times 1}$ to obtain the respective correspondence weight. The MLP consists of three linear layers with ReLU activation in the first two layers and a sigmoid activation after the last one, to ensure that the weights are between 0 and 1. In Sec. IV-E, we investigate also the performance of different other weighting schemes.

D. Loss Function

The registration procedure (Sec. III-A), the feature generation (Sec. III-B), and the computation of the weights (Sec. III-C) are fully differentiable. This allows us to directly optimize for the correct pose to learn features that are well suited for matching and therefore estimating the transformation. We split the loss $\mathcal{L}_T$ into a rotational part $\mathcal{L}_R$ and translational part $\mathcal{L}_t$. For the rotation, we want to minimize the angle between the ground truth R_{gt} and estimated rotation R_{est}

$$\mathcal{L}_R = \text{trace}(\mathsf{I}_3 - \mathsf{R}_{gt}^T \mathsf{R}_{est}), \quad (10)$$

and for the translational part the Euclidean distance between the ground truth $\mathbf{t}_{gt}$ and estimated $\mathbf{t}_{est}$ translation vectors

$$\mathcal{L}_t = \|\mathbf{t}_{gt} - \mathbf{t}_{est}\|. \quad (11)$$

The overall loss $\mathcal{L}_T$ is the weighted sum of both

$$\mathcal{L}_T = \alpha_1 \mathcal{L}_R + \alpha_2 \mathcal{L}_t, \quad (12)$$

where α_1 and α_2 are the respective weight factors.

TABLE I: Classical Registration

Approach	MAE(R) @0.5°	MAE(R) @1.0°	MAE(R) @5.0°	MAE(t) @0.1m	MAE(t) @0.3m	MAE(t) @0.5m
Teaser [36]	0.254 (62.02%)	0.369 (84.12%)	0.577 (97.80%)	0.060 (39.21%)	0.122 (87.30%)	0.141 (94.61%)
PCRNet [24]	0.366 (3.25%)	0.673 (14.57%)	2.071 (66.17%)	0.072 (1.18%)	0.198 (11.96%)	0.310 (26.62%)
HRegNet [17]	0.243 (74.90%)	0.296 (85.19%)	0.402 (90.86%)	0.060 (66.30%)	0.081 (88.07%)	0.085 (89.37%)
RANSAC + GICP	0.066 (97.52%)	0.067 (97.64%)	0.088 (98.43%)	0.048 (88.28%)	0.056 (97.46%)	0.057 (97.82%)
Ours	**0.045 (98.95%)**	**0.045 (98.95%)**	**0.059 (99.35%)**	**0.047 (96.86%)**	**0.048 (98.69%)**	**0.048 (98.69%)**

Quantitative Results on the Apollo-Southbay dataset. Presented numbers are the mean absolute errors and the success rate in brackets.

TABLE II: Compressed Registration

Approach	MAE(R) @0.5°	MAE(R) @1.0°	MAE(R) @5.0°	MAE(t) @0.1m	MAE(t) @0.3m	MAE(t) @0.5m
Teaser [36]	0.317 (19.90%)	0.545 (45.04%)	1.194 (79.84%)	0.071 (6.63%)	0.179 (46.23%)	0.239 (65.03%)
PCRNet [24]	0.353 (2.91%)	0.672 (13.32%)	2.117 (64.88%)	0.073 (0.87%)	0.200 (9.24%)	0.318 (21.98%)
HRegNet [17]	0.291 (56.99%)	0.390 (77.31%)	0.557 (86.55%)	0.067 (37.87%)	0.115 (79.89%)	0.126 (83.66%)
RANSAC + GICP	0.209 (51.69%)	0.299 (63.44%)	0.691 (78.39%)	**0.062** (26.50%)	0.123 (57.43%)	0.159 (66.58%)
Ours	**0.178 (91.95%)**	**0.207 (98.01%)**	**0.216 (98.55%)**	0.064 **(49.45%)**	**0.109 (96.61%)**	**0.113 (98.26%)**

E. Fine Registration

Performing the aforementioned steps described in Sec. III-A to Sec. III-C iteratively to improve the registration similar to ICP would require a lot of compute. In practice, we have seen that estimating the coarse registration with our network and finetune the slightly misaligned pose with GICP [25] and a Geman-McClure kernel [10] is more accurate and computationally efficient; especially when applied on the compact point cloud representation after the compression in Sec. III-B1. Thus, our approach does not aim at replacing ICP, but rather to support it by providing a good initial guess, such that the ICP converges easily to an accurate alignment. The fine registration can be efficiently done on the compressed point cloud, or slower but more accurate on the input.

IV. Experimental Evaluation

Our goal is to estimate the transformation between point clouds, as they are, for example, used in map matching or loop closure detection. For this, we want to estimate the transformation from a local source point cloud to the target point cloud which was recorded at a different point in time. We evaluate our approach on the Apollo-Southbay-ColumbiaPark dataset [18], which provides multiple runs, namely a mapping, training, and testing run, as well as ground truth poses. By this, we can register point clouds from the training/testing runs with the ones from mapping. Registering point clouds recorded at completely different points in time is especially challenging due to objects which moved slightly or miss completely, e.g., cars on parking lots exchanged or moved which often leads to wrong associations. We generate local map patches as done in other works [28], [33] by aggregating the scans within a 2 s timeframe and a bounding box of size [40 m$\times$ 40 m]. We homogenize the patches using a voxel grid with 10 cm resolution, resulting in point clouds with around 300,000 points, see Fig. 3 on the top left. We consider maps for registration that are within 10 m horizontal range for a sufficient amount of overlap but still have a large non-overlapping area. This can practically be accomplished even with cheap GPS sensors or with place recognition [28]. The initial rotations differ up to 180 degrees due to different driving directions and therefore make it pretty challenging for local registration methods.

We compare our approach with respect to two geometric approaches: RANSAC-based coarse registration with finetuning using GICP [25] as well as Teaser [36]. For the comparison with learning-based baselines we retrained PCRNet [24] and HRegNet [17] on the Apollo data. We exchanged the loss function of the PCRNet from the earth mover distance to our loss function in Eq. (12), which led to better performance for this data.

For the quantitative evaluation, we will evaluate the approaches based on the mean absolute error MAE$(\cdot)$ in terms of the rotation MAE(R) and translation MAE$(\mathbf{t})$, as follows

$$\mathrm{MAE}(\mathsf{R}) = \frac{1}{|I|}\sum_{i\in I}\arccos\left(\frac{\mathrm{trace}(\mathsf{R}_{gt(i)}^{T}\mathsf{R}_{est(i)})-1}{2}\right), \quad (13)$$

$$\mathrm{MAE}(\mathbf{t}) = \frac{1}{|I|}\sum_{i\in I}\|\mathbf{t}_{gt(i)} - \mathbf{t}_{est(i)}\|_2. \quad (14)$$

To measure the success rate of the registration, we always provide additionally for each metric the recall rate of how often it is below a certain threshold, e.g., MAE(R)@5°. We compute the mean absolute errors only for the inlier I, which are below the threshold.

A. Implementation Details

For our network, we utilize the compression encoder [33] with a compression ratio of around 1:100. For the feature enhancement network in Sec. III-B2, we use $B_k = 7$ KPConv blocks with radius $r = 2\,\mathrm{m}$. We use the transformer blocks with pre-layer normalization [35] for faster convergence and more stable training. In our experience, we saw that more than 2 transformer layers require substantially more training time and do not improve the performance significantly. The final feature dimension of the PointNet in Sec. III-B1, the ResNet blocks in Sec. III-B2 and the transformer in Sec. III-B3 is set to 256. The normals for the GICP fine registration are computed on the 25 nearest neighbors. We use Layer normalization [2] in all blocks due to the relatively small batch size of 8. Further, we use gradient accumulation over 4 batches and a learning rate of $5 \cdot 10^{-5}$ with the ADAMW

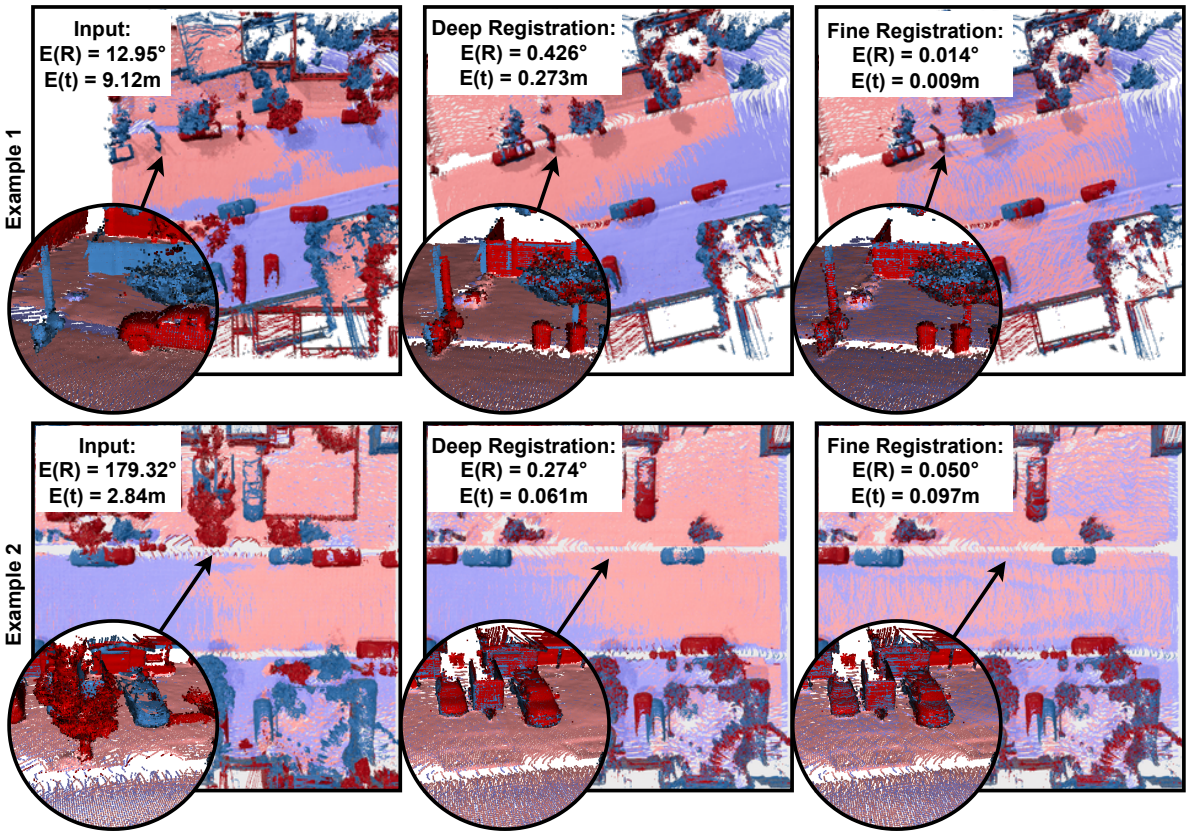


Fig. 4: Qualitative results of our point cloud registration method on two example scenes. In blue, the target point cloud is shown, while in red the source is transformed by either the initial guess (left), the registration solely based on our network (middle), and after the fine registration (right). Even under extreme conditions like 180 degrees wrong rotation, our network is able to align the point clouds properly. The fine registration is there to fix the last slight misalignment. $\mathsf{E}(\mathsf{R})$ and $\mathsf{E}(\mathsf{t})$ denotes the error in rotation and translation, respectively.

TABLE III: Ablation: Architecture

	KPConv	Transf.	GICP	MAE(R) @5.0°	MAE(t) @0.5m
[A]	✗	✗	✓	0.503 (48.89%)	0.120 (13.33%)
[B]	✗	✗	✗	2.990 (40.13%)	0.304 (1.70%)
[C]	✓	✗	✗	0.425 (98.69%)	0.101 (94.38%)
[D]	✗	✓	✗	1.285 (89.80%)	0.252 (54.51%)
[E]	✗	✓	✓	0.253 (94.64%)	0.121 (92.16%)
[F]	✓	✓	✗	0.405 (99.74%)	0.099 (96.21%)
[G]	✓	✓	✓	0.187 (99.87%)	0.115 (99.61%)

optimizer [16]. Registering a compressed point cloud requires around 0.04 s for feature extraction and pose estimation, as well as 0.02 s for the fine registration, i.e., in sum 60 ms for our approach. The compression of a dense point cloud of around 300 thousand points requires additional 0.173 s but can be done in a preprocession step and might be needed anyway for operation or storage. All experiments and the runtime are evaluated on an i7 @ 3.50 GHz with 8 cores and a GeForce RTX 2080 SUPER with 8 GB GPU memory.

B. Point Cloud Registration

In this experiment, we evaluate the accuracy of the point cloud registration without aiming at compression. All methods have access to the original input data (see Fig. 3a). Note that our approach still uses the compression network for feature extraction and subsampling, but uses the original input data (as in Fig. 3a) for a more precise fine registration. The results are depicted in Tab. I. Our approach is able to outperform both, the classical, as well as the other learning-based approaches. The RANSAC-based approach provides quite similar results to ours. In Fig. 4 we visualize some registration results of our approach. Our approach (middle) is able to recover from a bad initial guess (left). The fine-registration (right) allows fixing the small remaining pose error.

C. Compressed Point Cloud Registration

This experiment investigates the registration quality of the aforementioned approaches when only the compressed representation is available, for example, due to memory constraints on the vehicle. For a fair comparison with the baselines, we will either provide them with the compressed (Fig. 3b) or the decompressed point clouds (Fig. 3c), whatever works best for them. We will evaluate our approach directly operating on the compressed point clouds. We want to note that for our approach, we also do the fine registration with the compressed point clouds (see Fig. 3b) and *not* with the denser decompressed ones. In Tab. II the results for our approach and the baselines are shown. Our approach is the only one that is able to consistently (over 95% of the time) estimate the transformation within 1° and 0.3 m. The decompression of the point clouds leads to artifacts or additional noise, which seem to deteriorate the performance of the baselines. Our approach computes the initial guess in both cases on the compressed representation, and therefore the recall does not

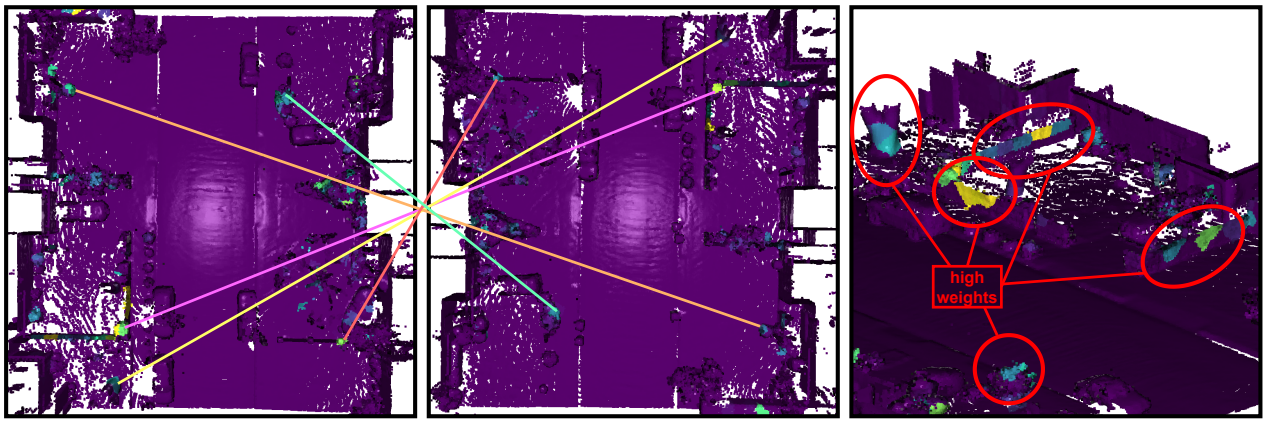


Fig. 5: Point clouds (target: left, source: middle) are visualized by the estimated weights $\mathbf{w}$ to show which points contribute to the estimated transformation. The brighter the color the higher the impact. On the right is a close-up view of the target. Our network only considers static, structured areas for the registration. Weights are upsampled on the input point cloud for better visualization.

drop significantly when compared to the registration without memory constraints (compare Tab. I). The mean absolute errors on the other hand drop by a factor of 2 to 3 when doing the fine registration on the compressed representation. The very low point resolution (around 1.5 m) does not allow for accurate normal estimations, limiting the performance of the GICP in the fine registration.

TABLE IV: Ablation: Weighting schemes

Metric	Constant	Entropy	Max	MLP
MAE(R) @5.0°	1.648 (89.28%)	0.725 (97.78%)	0.620 (98.04%)	0.405 (99.74%)
MAE(t) @0.5m	0.304 (10.85%)	0.186 (82.22%)	0.156 (86.93%)	0.099 (96.21%)

D. Ablation on the Network Architecture

In this section, we will take a look at different parts of the network to provide deeper insights into the approach. All the following results are evaluated on the validation set.

In Tab. III we investigate the performance of the architectural choices by enabling and disabling certain parts of the network. We can see that only using the compressed features [B] is not sufficient to reliably estimate the transformations, leading to the worst performance. Increasing the receptive field by using the proposed feature enhancement increases the performance drastically [C]. An additional transformer head can slightly improve the performance [F], while only using the transformer degrades the performance significantly [D]. These results are in line with results from different domains, showing that the locally inductive bias of convolutions speeds up the training and is especially helpful when having smaller-sized datasets [34]. The best results can be achieved when enabling all parts of the network and finetuning the results using GICP [G]. For completeness, we show the results of only using the GICP without an initial guess from our network [A]. The ICP cannot deal well with the large transformations, which therefore shows the need for a good initial guess.

E. Ablation on the Weighting Schemes

In this ablation study, we will investigate the performance of different weighting schemes. In addition to the MLP-based weighting, proposed in Sec. III-C, we investigate three others schemes. First, we give each correspondence the same constant weight ($\mathbf{w} = \mathbf{1}_N$) as baseline. Second, we compute the relative entropy (Kullback-Leibler divergence) between each row of W and the uniform distribution as a measure of ambiguousness, referred to as entropy. Third, we assign to each correspondence a weight based on its highest similarity, by row-wise max-pooling over the correspondence matrix W to support correspondences that have high similarity (Max).

The results are depicted in Tab. IV. The MLP-based method shows superior performance over the max-pooling and entropy-based versions, which both perform quite similarly. Using for every correspondence a constant weight performs substantially worse (with a 80 percent-point drop in the translation), showing the importance of computing individual weights. In Fig. 5, we additionally visualize the weights for the correspondences to illustrate, which points the network uses for the registration. For each target point, we have exactly one correspondence weight and therefore can directly colorize the points based on the weight magnitude. For the source, we compute the mean activation in W weighted by the correspondence weight $\mathbf{w}$ to colorize the points. Bright colors indicate a high weight, and dark colors indicate a low weight. For better visualization, we show the weights on the input point cloud based on its nearest neighbor in the compressed point cloud. Big areas with a low structure like ground, and huge walls have a very low weight and therefore are not considered for the registration. Even though cars provide a lot of structure, they also have a low weight. The network did maybe learn that cars often move and therefore are not reliable for the pose estimation. Only a few distinct and stable areas like trunks, small walls, or poles have a high weight and therefore are used for the registration. Since our approach relies on point-to-point correspondences, these results are in line with our expectations.

V. Conclusion

In this paper, we presented a novel architecture for point cloud registration under bad initial estimates. We exploit a

compression encoder to directly operate on a compressed representation, making it well suited for the registration when compression is needed. By predicting for each point correspondence a weight, we have shown, that we can improve our registration quality. Our approach is able to reliably align large-scale point clouds, without getting deteriorated by dynamic objects or non-overlapping areas. Additionally, our experiments suggest that it can be beneficial to directly work on the compressed representation, rather than decompressing first and doing the registration on the denser but noisier decompressed point clouds. Working solely on compressed representations, which is two orders of magnitudes smaller than the raw point cloud data has the potential to scale mapping-based systems to substantially environment dimensions, without compromising the systems performance.

REFERENCES

[1] Y. Aoki, H. Goforth, A.S. Rangaprasad, and S. Lucey. PointNetLK: Robust & Efficient Point Cloud Registration Using PointNet. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 7156–7165, 2019.

[2] J.L. Ba, J.R. Kiros, and G.E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] J. Behley and C. Stachniss. Efficient Surfel-Based SLAM using 3D Laser Range Data in Urban Environments. In *Proc. of Robotics: Science and Systems (RSS)*, 2018.

[4] P. Besl and N. McKay. A Method for Registration of 3D Shapes. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 14(2):239–256, 1992.

[5] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse Iterative Closest Point. *Computer Graphics Forum*, 32(5):113–123, 2013.

[6] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, and C. Stachniss. Adaptive Robust Kernels for Non-Linear Least Squares Problems. *IEEE Robotics and Automation Letters (RA-L)*, 6:2240–2247, 2021.

[7] Y. Chen and G. Medioni. Object Modelling by Registration of Multiple Range Images. *Journal on Image and Vision Computing (IVC)*, 10(3):145–155, 1992.

[8] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24(6):381–395, 1981.

[9] N. Gelfand, N.J. Mitra, L.J. Guibas, and H. Pottmann. Robust Global Registration. In *Proc. of the Symp. on Geometry Processing*, volume 2, page 5, 2005.

[10] S. Geman and D. McClure. Bayesian image analysis: An application to single photon emission tomography. *American Statistical Association*, pages 12–18, 1985.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] P.J. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[13] A.E. Johnson and S.B. Kang. Registration and Integration of Textured 3D Data. *Journal on Image and Vision Computing (IVC)*, 17(2):135–147, 1999.

[14] W. Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[15] C. Kerl, J. Sturm, and D. Cremers. Robust Odometry Estimation for RGB-D Cameras. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3748–3754, 2013.

[16] I. Loshchilov and F. Hutter. Decoupled Weight Decay Regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[17] F. Lu, G. Chen, Y. Liu, L. Zhang, S. Qu, S. Liu, and R. Gu. Hregnet: A hierarchical network for large-scale outdoor lidar point cloud registration. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 16014–16023, 2021.

[18] W. Lu, Y. Zhou, G. Wan, S. Hou, and S. Song. L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[19] W. Lu, G. Wan, Y. Zhou, X. Fu, P. Yuan, and S. Song. DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 12–21, 2019.

[20] J. Park, Q. Zhou, and V. Koltun. Colored Point Cloud Registration Revisited. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[21] F. Pomerleau, F. Colas, and R. Siegwart. A Review of Point Cloud Registration Algorithms for Mobile Robotics. *Foundations and Trends in Robotics*, 4:1–104, 05 2015.

[22] C.R. Qi, H. Su, K. Mo, and L.J. Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[23] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Proc. of Int. Conf. on 3-D Digital Imaging and Modeling*, pages 145–152, 2001.

[24] V. Sarode, X. Li, H. Goforth, Y. Aoki, R.A. Srivatsan, S. Lucey, and H. Choset. PCRNet: Point Cloud Registration Network using PointNet Encoding. *arXiv preprint arXiv:1908.07906*, 2019.

[25] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proc. of Robotics: Science and Systems (RSS)*, 2009.

[26] J. Serafin and G. Grisetti. NICP: Dense Normal Based Point Cloud Registration. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 742–749, 2015.

[27] H. Thomas, C. Qi, J. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas. KPConv: Flexible and Deformable Convolution for Point Clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[28] A. Uy and G. Lee. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4470–4479, 2018.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Proc. of the Advances in Neural Information Processing Systems (NIPS)*, 30, 2017.

[30] G. Wang, X. Wu, Z. Liu, and H. Wang. PWCLO-Net: Deep LiDAR Odometry in 3D Point Clouds Using Hierarchical Embedding Mask Optimization. *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 15905–15914, 2021.

[31] Y. Wang and J.M. Solomon. Deep Closest Point: Learning Representations for Point Cloud Registration. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, 2019.

[32] X. Wei, I.A. Bârsan, S. Wang, J. Martinez, and R. Urtasun. Learning to Localize Through Compressed Binary Maps. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10316–10324, 2019.

[33] L. Wiesmann, A. Milioto, X. Chen, C. Stachniss, and J. Behley. Deep Compression for Dense Point Cloud Maps. *IEEE Robotics and Automation Letters (RA-L)*, 6:2060–2067, 2021.

[34] T. Xiao, P. Dollar, M. Singh, E. Mintun, T. Darrell, and R. Girshick. Early Convolutions help Transformers see Better. *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 34, 2021.

[35] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. On Layer Normalization in the Transformer Architecture. In *Proc. of the Int. Conf. on Machine Learning (ICML)*, pages 10524–10533, 2020.

[36] H. Yang, J. Shi, and L. Carlone. Teaser: Fast and Certifiable Point Cloud Registration. *IEEE Trans. on Robotics (TRO)*, 37(2):314–333, 2020.

[37] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration. *IEEE Trans. on Pattern Analalysis and Machine Intelligence (TPAMI)*, 38(11):2241–2254, 2015.

[38] Z.J. Yew and G.H. Lee. 3DFeat-Net: Weakly Supervised Local 3D Features for Point Cloud Registration. In V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, editors, *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 630–646, Cham, 2018. Springer International Publishing.

[39] Z.J. Yew and G.H. Lee. RPM-Net: Robust Point Matching using Learned Features. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] H. Yin, Y. Wang, L. Tang, X. Ding, S. Huang, and R. Xiong. 3D LiDAR Map Compression for Efficient Localization on Resource Constrained Vehicles. *IEEE Trans. on Intelligent Transportation Systems (ITS)*, 22(2):837–852, 2020.

[41] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser. 3DMatch: Learning Local Geometric Descriptors from RGB-D Reconstructions. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.