DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration

Weixin Lu Guowei Wan Yao Zhou Xiangyu Fu Pengfei Yuan Shiyu Song*
Baidu Autonomous Driving Technology Department (ADT)
{luweixin, wanguowei, zhouyao, fuxiangyu, yuanpengfei, songshiyu}@baidu.com

Abstract

We present DeepVCP - a novel end-to-end learning-based 3D point cloud registration framework that achieves comparable registration accuracy to prior state-of-the-art geometric methods. Different from other keypoint based methods where a RANSAC procedure is usually needed, we implement the use of various deep neural network structures to establish an end-to-end trainable network. Our keypoint detector is trained through this end-to-end structure and enables the system to avoid the inference of dynamic objects, leverages the help of sufficiently salient features on stationary objects, and as a result, achieves high robustness. Rather than searching the corresponding points among existing points, the key contribution is that we innovatively generate them based on learned matching probabilities among a group of candidates, which can boost the registration accuracy. We comprehensively validate the effectiveness of our approach using both the KITTI dataset and the Apollo-SouthBay dataset. Results demonstrate that our method achieves comparable registration accuracy and runtime efficiency to the state-of-the-art geometry-based methods, but with higher robustness to inaccurate initial poses. Detailed ablation and visualization analysis are included to further illustrate the behavior and insights of our network. The low registration error and high robustness of our method make it attractive to the substantial applications relying on the point cloud registration task.

1. Introduction

Recent years has seen a breakthrough in deep learning that has led to compelling advancements in most semantic computer vision tasks, such as classification [22], detection [15, 32] and segmentation [24, 2]. A number of works have highlighted that these empirically defined problems can be solved by using DNNs, yielding remarkable results and good generalization behavior. The geometric problems that are defined theoretically, which is another important category of the problem, has seen many recent developments with emerging results in solving vision problems, including stereo matching [47, 5], depth estimation [36] and SFM [40, 51]. But it has been observed, for tasks using 3D point clouds as input, for example, the 3D point cloud registration task, experiential solutions of most recent attempts [49, 11, 7] have not been adequate, especially in terms of local registration accuracy.

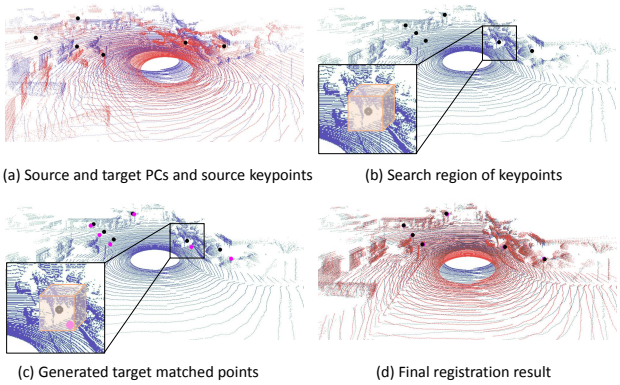

Figure 1. The illustration of the major steps of our proposed end-to-end point cloud registration method: (a) The source (red) and target (blue) point clouds and the keypoints (black) detected by the point weighting layer. (b) A search region is generated for each keypoint and represented by grid voxels. (c) The matched points (magenta) generated by the corresponding point generation layer. (d) The final registration result computed by performing SVD given the matched keypoint pairs.

Point cloud registration is a task that aligns two or more different point clouds collected by LiDAR (Light Detection and Ranging) scanners by estimating the relative transformation between them. It is a well-known problem and plays an essential role in many applications, such as LiDAR SLAM [50, 8, 19, 27], 3D reconstruction and mapping [38, 10, 45, 9], positioning and localization [48, 20, 42, 25], object pose estimation [43] and so on.

LiDAR point clouds have innumerable unique aspects that can enhance the complexity of this particular problem, including the local sparsity, large amount of data generated and the noise caused by dynamic objects. Compared to the image matching problem, the sparsity of the point cloud

*Author to whom correspondence should be addressed

makes finding two exact matching points from the source and target point clouds usually infeasible. It also increases the difficulty of feature extraction due to the large appearance difference of the same object viewed by a laser scanner from different perspectives. The millions of points produced every second requires highly efficient algorithms and powerful computational units. ICP and its variants have relatively good computational efficiency, but are known to be susceptible to local minima, therefore, rely on the quality of the initialization. Finally, appropriate handling of the interference caused by the noisy points of dynamic objects typically is crucial for delivering an ideal estimation, especially when using real LiDAR data.

In this work, titled "DeepVCP" (Virtual Corresponding Points), we propose an end-to-end learning-based method to accurately align two different point clouds. The name DeepVCP accurately captures the importance of the virtual corresponding point generation step which is one of the key innovative designs proposed in our approach. An overview of our framework is shown in Figure 1.

We first extract semantic features of each point both from the source and target point clouds using the latest point cloud feature extraction network, PointNet++ [31]. They are expected to have certain semantic meanings to empower our network to avoid dynamic objects and focus on those stable and unique features that are good for registration. To further achieve this goal, we select the keypoints in the source point cloud that are most significant for the registration task by making use of a point weighting layer to assign matching weights to the extracted features through a learning procedure. To tackle the problem of local sparsity of the point cloud, we propose a novel corresponding point generation method based on a feature descriptor extraction procedure using a mini-PointNet [30] structure. We believe that it is the key contribution to enhance registration accuracy. Finally, besides only using the L1 distance between the source keypoint and the generated corresponding point as the loss, we propose to construct another corresponding point by incorporating the keypoint weights adaptively and executing a single optimization iteration using the newly introduced SVD operator in TensorFlow. The L1 distance between the keypoint and this newly generated corresponding point is again used as another loss. Unlike the first loss using only local similarity, this newly introduced loss builds the unified geometric constraints among local keypoints. The end-to-end closed-loop training allows the DNNs to generalize well and select the best keypoints for registration.

To summarize, our main contributions are:

- To the best of our knowledge, our work is the first end-to-end learning-based point cloud registration framework yielding comparable results to prior state-of-the-art geometric ones.
- Our learning-based keypoint detection, novel corresponding point generation method and the loss function that incorporates both the local similarity and the global geometric constraints to achieve high accuracy in the learning-based registration task.
- Rigorous tests and detailed ablation analysis using the KITTI [13] and Apollo-SouthBay [25] datasets to fully demonstrate the effectiveness of the proposed method.

2. Related Work

The survey work from F. Pomerleau et al. [29] provides a good overview of the development of traditional point cloud registration algorithms. [3, 37, 26, 39, 44] are some representative works among them. A discussion of the full literature of the these methods is beyond the scope of this work.

The attempt of using learning based methods starts by replacing each individual component in the classic point cloud registration pipeline. S. Salti et al. [35] proposes to formulate the problem of 3D keypoint detection as a binary classification problem using a pre-defined descriptor, and attempts to learn a Random Forest [4] classifier that can find the appropriate keypoints that are good for matching. M. Khoury et al. [21] proposes to first parameterize the input unstructured point clouds into spherical histograms, then a deep network is trained to map these high-dimensional spherical histograms to low-dimensional descriptors in Euclidean space. In terms of the method of keypoint detection and descriptor learning, the closest work to our proposal is [46]. Instead of constructing an End-to-End registration framework, it focuses on joint learning of keypoints and descriptors that can maximize local distinctiveness and similarity between point cloud pairs. G. Georgakis et al. [14] solves a similar problem for RGB-D data. Depth images are processed by a modified Faster R-CNN architecture for joint keypiont detection and descriptor estimation. Despite the different approaches, they all focus on the representation of the local distinctiveness and similarity of the keypoints. During keypoint selection, content awareness in real scenes is ignored due to the absence of the global geometric constraints introduced in our end-to-end framework. As a result, keypoints on dynamic objects in the scene cannot be rejected in these approaches.

Some recent works [49, 11, 7, 1] propose to learn 3D descriptors leveraging the DNNs, and attempt to solve the 3D scene recognition and re-localization problem, in which obtaining accurate local matching results is not the goal. In order to achieve that, methods, as ICP, are still necessary for the registration refinement.

M. Velas et al. [41] encodes the 3D LiDAR data into a specific 2D representation designed for multi-beam mechanical LiDARs. CNNs is used to infer the 6 DOF poses as a classification or regression problem. An IMU assisted LiDAR odometry system is built upon it. Our approach processes the original unordered point cloud directly and is de-

signed as a general point cloud registration solution.

3. Method

This section describes the architecture of the proposed network designed in detail as shown in Figure 2.

3.1. Deep Feature Extraction

The input of our network consists of the source and target point cloud, the predicted (prior) transformation, and the ground truth pose required only during the training stage. The first step is extracting feature descriptors from the point cloud. In the proposed method, we extract feature descriptors by applying a deep neural network layer, denoted as the Feature Extraction (FE) Layer. As shown in Figure 2, we feed the source point cloud, represented as an $N_1 \times 4$ tensor, into the FE layer. The output is an $N_1 \times 32$ tensor representing the extracted local feature. The FE layer we used here is PointNet++ [31] which is a poineer work addressing the issue of consuming unordered points in a network architecture. We are also considering to try rotation invariant 3D descriptors [6, 16, 23] in the future.

These local features are expected to have certain semantic meanings. Working together with the weighting layer to be introduced next, we expect our end-to-end network to be capable to avoid the interference from dynamic objects and deliver precise registration estimation. In Section 4.4, we visualize the selected keypoints and demonstrate that the dynamic objects are successfully avoided.

3.2. Point Weighting

Inspired by the attention layer in 3DFeatNet [46], we design a point weighting layer to learn the saliency of each point in an end-to-end framework. Ideally, points with invariant and distinct features on static objects should be assigned higher weights.

As shown in Figure 2, $N_1 \times 32$ local features from the source point cloud are fed into the point weighting layer. The weighting layer consists of a multi-layer perceptron (MLP) of 3 stacking fully connected layers and a top k operation. The first two fully connected layers use the batch normalization and the ReLU activation function, and the last layer omits the normalization and applies the *softplus* activation function. The most significant N points are selected as the keypoints through the top k operator and their learned weights are used in the subsequent processes.

Our approach is different from 3DFeatNet [46] in a few ways. First, the features used in the attention layer are extracted from local patches, while ours are semantic features extracted directly from the point cloud. We have greater receptive fields learned from an encoder-decoder style network (PointNet++ [31]). Moreover, our weighting layer does not output a 1D rotation angle to determine the feature direction, because our design of the feature embedding layer in the next section uses a symmetric and isotropic network architecture.

3.3. Deep Feature Embedding

After extracting N keypoints from the source point cloud, we seek to find the corresponding points in the target point cloud for the final registration. In order to achieve this, we need a more detailed feature descriptor that can better represent their geometric characteristics. Therefore, we apply a deep feature embedding (DFE) layer on their neighborhood points to extract these local features. The DFE layer we used is a mini-PointNet [30, 7, 25] structure.

Specifically, we collect K neighboring points within a certain radius d of each keypoint. In case that there are less than K neighboring points, we simply duplicate them. For all the neighboring points, we use their local coordinates and normalize them by the searching radius d. Then, we concatenate the FE feature extracted in Section 3.1 with the local coordinates and the LiDAR reflectance intensities of the neighboring points as the input to the DFE layer.

The mini-PointNet consists of a multi-layer perceptron (MLP) of 3 stacking fully connected layers and a *max-pooling* layer to aggregate and obtain the feature descriptor. As shown in Figure 2, the input of the DFE layer is an $N \times K \times 36$ vector, which refers to the local coordinate, the intensity, and the 32-dimensional FE feature descriptor of each point in the neighborhood. The output of the DFE layer is again a 32-dimensional vector. In Section 4.3, we show the effectiveness of the DFE layer and how it help improve the registration precision significantly.

3.4. Corresponding Point Generation

Similar to ICP, our approach also seeks to find corresponding points in the target point cloud and estimate the transformation. The ICP algorithm chooses the closest point as the corresponding point. This prohibits backpropagation as it is not differentiable. Furthermore, there are actually no exact corresponding points in the target point cloud to the source due to its sparsity nature. To tackle the above problems, we propose a novel network structure, the corresponding point generation (CPG) layer, to generate corresponding points from the extracted features and the similarity represented by them.

We first transform the keypoints from the source point cloud using the input predicted transformation. Let $\{x_i, x_i'\}, i = 1, \cdots, N$ denote the 3D coordinate of the keypoint from the source point cloud and its transformation in the target point cloud, respectively. In the neighborhood of x_i', we divide its neighboring space into $(\frac{2r}{s} + 1, \frac{2r}{s} + 1, \frac{2r}{s} + 1)$ 3D grid voxels, where r is the searching radius and s is the voxel size. Let us denote the centers of the 3D voxels as $\{y_j'\}, j = 1, \cdots, C$, which are considered as the candidate corresponding points. We also extract their DFE

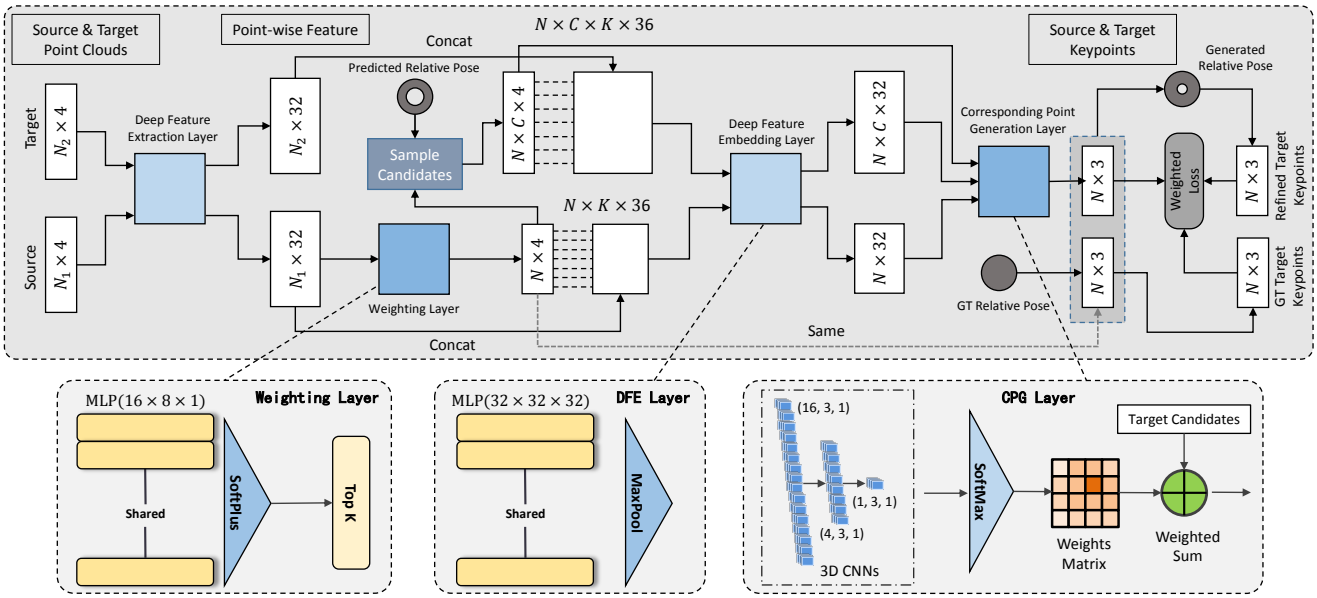

Figure 2. The architecture of the proposed end-to-end learning network for 3D point cloud registration, DeepVCP. The source and target point clouds are fed into the deep feature extraction layer, then N keypoints are extracted from the source point cloud by the weighting layer. $N \times C$ candidate corresponding points are selected from the target point cloud, followed by a deep feature embedding operation. The corresponding keypoints in the target point cloud are generated by the corresponding points generation layer. Finally, we propose to use the combination of two losses those encode both the global geometric constraints and local similarities.

feature descriptors as we did in Section 3.3. The output is an $N \times C \times 32$ tensor. Similar to [25], those tensors representing the extracted DFE features descriptors from the source and target are fed into a three-layer 3D CNNs, followed by a *softmax* operation, as shown in Figure 2. The 3D CNNs can learn a similarity distance metric between the source and target features, and more importantly, it can smooth (regularize) the matching volume and suppress the matching noise. The *softmax* operation is applied to convert the matching costs into probabilities.

Finally, the target corresponding point y_i is calculated through a *weighted-sum* operation as:

$$y_i = \frac{1}{\sum_{j=1}^{C} w_j} \sum_{j=1}^{C} w_j \cdot y'_j, \tag{1}$$

where w_j is the similarity probability of each candidate corresponding point y'_j. The computed target corresponding points are represented by a $N \times 3$ tensor.

Compared to the traditional ICP algorithm that relied on the iterative optimization or the methods [33, 7, 49] which search the corresponding points among existing points from the target point cloud and use RANSAC to reject outliers, our approach utilizes the powerful generalization capability of CNNs in similarity learning, to directly "guess" where the corresponding points are in the target point cloud. This eliminates the use of RANSAC, reduces the iteration times to 1, significantly reduces the running time, and achieves fine registration with high precision.

Another implementation detail worth mentioning is that we conduct a bidirectional matching strategy during inference to improve the registration accuracy. That is, the input point cloud pair is considered as the source and target simultaneously. While we do not do this during training, because this does not improve the overall performance of the model.

3.5. Loss

For each keypoint x_i from the source point cloud, we can calculate its corresponding ground truth $\bar{y}_i$ with the given ground truth transformation $(\bar{R}, \bar{T})$. Using the estimated target corresponding point y_i in Section 3.4, we can directly compute the $L1$ distance in the Euclidean space as a loss:

$$Loss_1 = \frac{1}{N} \sum_{i=1}^{N} |\bar{y}_i - y_i|. \tag{2}$$

If only the $Loss_1$ in Equation 2 is used, the keypoint matching procedure during the registration is independent for each one. Consequently, only the local neighboring context is considered during the matching procedure, while the registration task is obviously constrained with a global geometric transform. Therefore, it is essential to introduce another loss including global geometric constraints.

Inspired by the iterative optimization in the ICP algorithm, we perform a single optimization iteration. That is, we perform a singular value decomposition (SVD) step to estimate the relative transformation given the corresponding keypoint pairs $\{x_i, y_i\}$, $i = 1, \cdots, N$, and the learned

weights from the weighting layer. Following an outlier rejection step, where 20% point pairs are rejected given the estimated transformation, another SVD step is executed to further refine the estimation (R, T). Then the second loss in our network is defined as:

$$Loss_2 = \frac{1}{N}\sum_{i=1}^{N} |\bar{y}_i - (Rx_i + T)|. \quad (3)$$

Thanks to [18], the latest Tensorflow has supported the SVD operator and its backpropagation. This ensures that the proposed network can be trained in an end-to-end pattern. As a result, the combined loss is defined as:

$$Loss = \alpha Loss_1 + (1 - \alpha) Loss_2, \quad (4)$$

where α is the balancing factor. In Section 4.3, we demonstrate the effectiveness of our loss design. It has been tested that the convergence rate is faster and the accuracy is higher when the L_1 loss is applied.

It is worth to note that the estimated corresponding keypoints y_i are actually constantly being updated together as the estimated transformation (R, T) during the training. When the network converges, the estimated corresponding keypoints become unlimitedly close to the ground truth. It is interesting that this training procedure is actually quite similar to the classic ICP algorithm. While the network only needs a single iteration to find the optimal corresponding keypoint and then estimate the transformation during inference, which is very valuable.

3.6. Dataset Specific Refinement

Moreover, we find that there are some characteristics in KITTI and Apollo-SouthBay datasets that can be utilized to further improve the registration accuracy. Experimental results using many different datasets are introduced in the supplemental material. This specific network duplication method is not applied in these datasets.

Because the point clouds from Velodyne HDL64 are distributed within a relatively narrow region in the z-direction, the keypoints constraining the z-direction are usually quite different from the other two, such as the points on the ground plane. This causes the registration precision at the z, $roll$ and $pitch$ directions to decline. To tackle this problem, we actually duplicate the whole network structure as shown in Figure 2, and use two copies of the network in a cascade pattern. The back network uses the estimated transformation from the front network as the input, but replaces the 3D CNNs in the CPG step of the latter with a 1D one sampling in the z direction only. Both the networks share the same FE layer, becasue we do not want to extract FE features twice. This increases the z, $roll$ and $pitch$'s estimation precision.

4. Experiments

4.1. Benchmark Datasets

We evaluate the performance of the proposed network using 11 training sequences of the KITTI odometry dataset [13]. The KITTI dataset contains point clouds captured with a Velodyne HDL64 LiDAR in Karlsruhe, Germany together with the "ground truth" poses provided by a high-end GNSS/INS integrated navigation system. We split the dataset into two groups, the training, and the testing. The training group includes 00-07 sequences, and the testing includes 08 - 10 sequences.

Another dataset that is used for evaluation is the Apollo-SouthBay dataset [25]. It collected point clouds using the same model of LiDAR as the KITTI dataset, but, in the San Francisco Bay area, United States. Similar to KITTI, it covers various scenarios including residential areas, urban downtown areas, and highways. We also find that the "ground truth" poses in Apollo-SouthBay is more accurate than KITTI odometry dataset. Some ground truth poses in KITTI involve larger errors, for example, the first 500 frames in Sequence 08. Moreover, the mounting height of the LiDAR in Apollo-SouthBay is slightly higher than KITTI. This allows the LiDAR to see larger areas in the z direction. We find that the keypoints picked up in these high regions sometimes are very helpful for registration. The setup of the training and test sets is similar to [25] with the mapping portion discarded. There is no overlap between the training and testing data. Refer to the supplemental material for additional experimental results using more challenging datasets.

The initial poses are generated by adding random noises to the ground truth. In KITTI and Apollo-SouthBay, we added a uniformly distributed random error of $[0 \sim 1.0]m$ in x-y-z dimension, and a random error of $[0 \sim 1.0]^\circ$ in $roll$-$pitch$-yaw dimension. The models in different datasets are trained separately. Refer to the supplemental material where we evaluate robustness given inaccurate initial poses using other datasets.

4.2. Performance

Baseline Algorithms We present extensive performance evaluation by comparing with a few point cloud registration algorithms based on geometry. They are: (i) The ICP family, such as ICP [3], G-ICP [37], and AA-ICP [28]; (ii) NDT-P2D [39]; (iii) GMM family, such as CPD [26]; (iv) The learning-based method, 3DFeat-Net [46]. The implementations of ICP, G-ICP, AA-ICP, and NDT-P2D are from the Point Cloud Library (PCL) [34]. Gadomski's implementation [12] of the CPD method is used and the original 3DFeat-Net implementation with RANSAC for the registration task is used.

Evaluation Criteria The evaluation is performed by

calculating the angular and translational error of the estimated relative transformation (R, T) against the ground truth $(\bar{R}, \bar{T})$. The *chordal* distance [17] between R and $\bar{R}$ is calculated via the Frobenius norm of the rotation matrix, denoted as $||R - \bar{R}||_F$. The angular error θ then can be calculated as $\theta = 2\sin^{-1}(\frac{||R-\bar{R}||_F}{\sqrt{8}})$. The translational error is calculated as the Euclidean distance between T and $\bar{T}$.

KITTI Dataset We sample the input source LiDAR scans at 30 frame intervals and enumerate its registration target within $5m$ distance to it. The original point cloud in the dataset includes about $108,000$ points/frame. We use original point clouds for methods such as ICP, G-ICP, AA-ICP, NDT, and 3DFeat-Net. To keep CPD's computing time not intractable, we downsample the point clouds using a voxel size of $0.1m$ leaving about $50,000$ points on average. The statistics of the running time of all the methods are shown in Figure 3. For our proposed method, we evaluate two versions. One is the base version, denoted as "Ours-Base", that infers all the degree of freedoms x, y, z, $roll$, $pitch$, and yaw at once. The other is an improved version with network duplication as we discussed in Section 3.6, denoted as "Ours-Duplication". The angular and translational errors of all the methods are listed in Table 1. As can be seen, for the KITTI dataset, DeepVCP achieves comparable registration accuracy with regards to most geometry-based methods like AA-ICP, NDT-P2D, but performs slightly worse than G-ICP and ICP, especially for the angular error. The lower maximum angular and translational errors show that our method has good robustness and stability, therefore it has good potential in significantly improving the overall system performance for large point cloud registration tasks.

Method	Angular Error(°)		Translation Error(m)	
	Mean	Max	Mean	Max
ICP-Po2Po [3]	0.139	1.176	0.089	2.017
ICP-Po2Pl [3]	0.084	1.693	0.065	2.050
G-ICP [37]	**0.067**	**0.375**	**0.065**	2.045
AA-ICP [28]	0.145	1.406	0.088	2.02
NDT-P2D [39]	0.101	4.369	0.071	2.000
CPD [26]	0.461	5.076	0.804	7.301
3DFeat-Net [46]	0.199	2.428	0.116	4.972
Ours-Base	0.195	1.700	0.073	0.482
Ours-Duplication	0.164	1.212	0.071	**0.482**

Table 1. Comparison using the KITTI dataset. Our performance is comparable against traditional geometry-based methods and better than the learning-based method, 3DFeat-Net. The much lower maximum errors demonstrate good robustness.

Apollo-SouthBay Dataset In Apollo-SouthBay dataset, we sample at 100 frame intervals, and again enumerate the target within $5m$ distance. All other parameter settings for each individual method are the same as the KITTI dataset. The angular and translational errors are listed in Table 2. For the Apollo-SouthBay dataset, most methods including ours have a performance improvement, which might be due to the better ground truth poses provided by the dataset. Our system with the duplication design achieves the second-best mean translational accuracy and comparable angular accuracy with regards to other traditional methods. Additionally, the lowest maximum translational error demonstrates good robustness and stability of our proposed learning-based method.

Method	Angular Error(°)		Translation Error(m)	
	Mean	Max	Mean	Max
ICP-Po2Po [3]	0.051	0.678	0.089	3.298
ICP-Po2Pl [3]	0.026	0.543	0.024	4.448
G-ICP [37]	**0.025**	**0.562**	**0.014**	1.540
AA-ICP [28]	0.054	1.087	0.109	5.243
NDT-P2D [39]	0.045	1.762	0.045	1.778
CPD [26]	0.054	1.177	0.210	5.578
3DFeat-Net [46]	0.076	1.180	0.061	6.492
Ours-Base	0.135	1.882	0.024	**0.875**
Ours-Duplication	0.056	0.875	0.018	0.932

Table 2. Comparison using the Apollo-SouthBay dataset. Our system achieves the second best mean translational error and the lowest maximum translational error. The low maximum errors demonstrate good robustness of our method.

Run-time Analysis We evaluate the runtime performance of our framework with a GTX 1080 Ti GPU, Core i7-9700K CPU, and 16GB Memory as shown in Figure 3. The total end-to-end inference time of our network is about 2 seconds for registering a frame pair with the duplication design in Section 3.6. Note that DeepVCP is significantly faster than the other learning-based approach, 3DFeat-Net [46], because we extract only 64 keypoints instead of 1024, and do not rely on a RANSAC procedure.

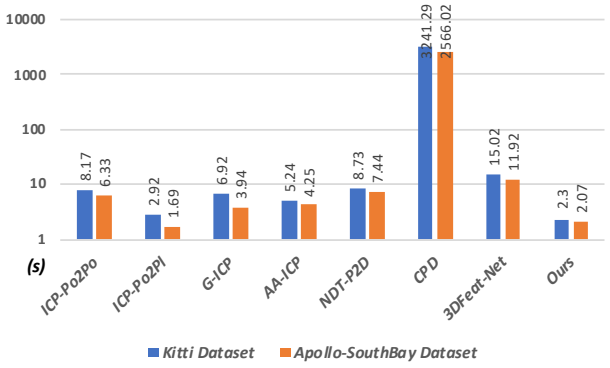

Figure 3. The running time performance analysis of all the methods. The total end-to-end inference time of our network is about 2 seconds for registering a frame pair.

4.3. Ablations

In this section, we use the same training and testing data from the Apollo-SouthBay dataset to further evaluate each component or proposed design in our work.

Deep Feature Embedding In Section 3.3, we propose to construct the network input by concatenating the FE feature together with the local coordinates and the intensities of the neighboring points. Now, we take a deeper look at this design choice by conducting the following experiments: i) LLF-DFE: Only the local coordinates and the intensities are used; ii) FEF-DFE: Only the FE feature is used; iii) FEF: The DFE layer is discarded. The FE feature is directly used as the input to the CPG layer. In the target point cloud, the FE features of the grid voxel centers are interpolated. It is seen that the DFE layer is crucial to this task as there is severe performance degradation without it as shown in Table 3. The LLF-DFE and FEF-DFE give competitive results while our design gives the best performance.

Method	Angular Error($^\circ$)		Translation Error(m)	
	Mean	Max	Mean	Max
LLF-DFE	0.058	0.861	0.024	0.813
FEF-DFE	0.057	**0.790**	0.026	**0.759**
FEF	0.700	2.132	0.954	8.416
Ours	**0.056**	0.875	**0.018**	0.932

Table 3. Comparison w/o the DFE layer. The usage of DFE layer is crucial as there is severe performance degradation as shown in Method FEF. When only partial features are used in DFE layer, it gives competitive results as shown in Method LLF-DFE and FEF-DFE, while ours yields the best performance.

Corresponding Points Generation To demonstrate the effectiveness of the CPG, we directly search the best corresponding point among the existing points in the target point cloud taking the predicted transformation into consideration. Specifically, for each source keypoint, the point with the highest similarity score in the feature space in the target neighboring field is chosen as the corresponding point. It turns out that it is unable to converge using our proposed loss function. The reason might be that the proportion of the positive and negative samples is extremely unbalanced.

Loss In Section 3.5, we propose to use the combination of two losses to incoorporate the global geometric information, and a balancing factor α is introduced. In order to demonstrate the necessity of using both the losses, we sample 11 values of α from 0.0 to 1.0 and observe the registration accuracy. In Figure 4, we find that the balancing factor of 0.0 and 1.0 obviously give larger angular and translational mean errors. This clearly demonstrates the effectiveness of the combined loss function design. It is also quite interesting that it yields similar accuracies for α between 0.1 - 0.9. We conclude that this might be because of the powerful generalization capability of deep neural networks. The parameters in the networks can be well generalized to adopt any α values away from 0.0 or 1.0. Therefore, we use 0.6 in all our experiments.

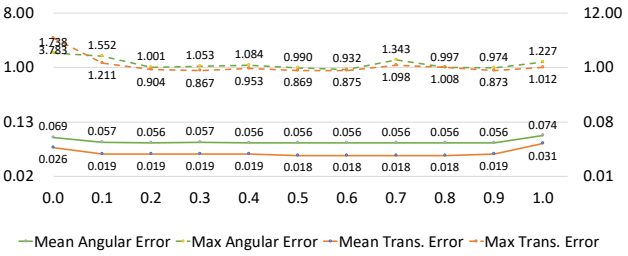

Figure 4. Registration accuracy comparison with different α values in the loss function. Any α values away from 0.0 or 1.0 give similarly good accuracies. This demonstrates the powerful generalization capability of deep neural networks.

4.4. Visualizations

In this section, to offer better insights on the behavior of the network, we visualize the keypoints chosen by the point weighting layer and the similarity probability distribution estimated in the CPG layer.

Visualization of Keypoints In Section 3.1, we propose to extract semantic features using PointNet++ [31], and weigh them using a MLP network structure. We expect that our end-to-end framework can intelligently learn to select keypoints that are unique and stable on stationary objects, such as traffic poles, tree trunks, but avoid the keypoints on dynamic objects, such as pedestrians, cars. In addition to this, we duplicate our network in Section 3.6. The front network with the 3D CNNs CPG layer are expected to find meaningful keypoints those have good constraints in all six degrees of freedom. While the back network with the 1D CNNs are expected to find those are good in z, $roll$ and $pitch$ directions. In Figure 5, the detected keypoints are shown compared with the camera photo and the LiDAR scan in the real scene. The pink and grey keypoints are detected by the front and back network, respectively. We observe that the distribution of keypoints match our expectations as the pink keypoints mostly appear on objects with salient features, such as tree trunks and poles, while the grey ones are mostly on the ground. Even in the scene where there are lots of cars or buses, none of keypoints are detected on them. This demonstrates that our end-to-end framework is capable to detect the keypoints those are good for the point cloud registration task.

Visualization of CPG Distribution The CPG layer in Section 3.4 estimates the matching similarity probability of each keypoint to its candidate corresponding ones. Figure 6 depicts the estimated probabilities by visualizing them in x and y dimensions with 9 fixed z values. On the left and right, the black and pink points are the keypoints from the

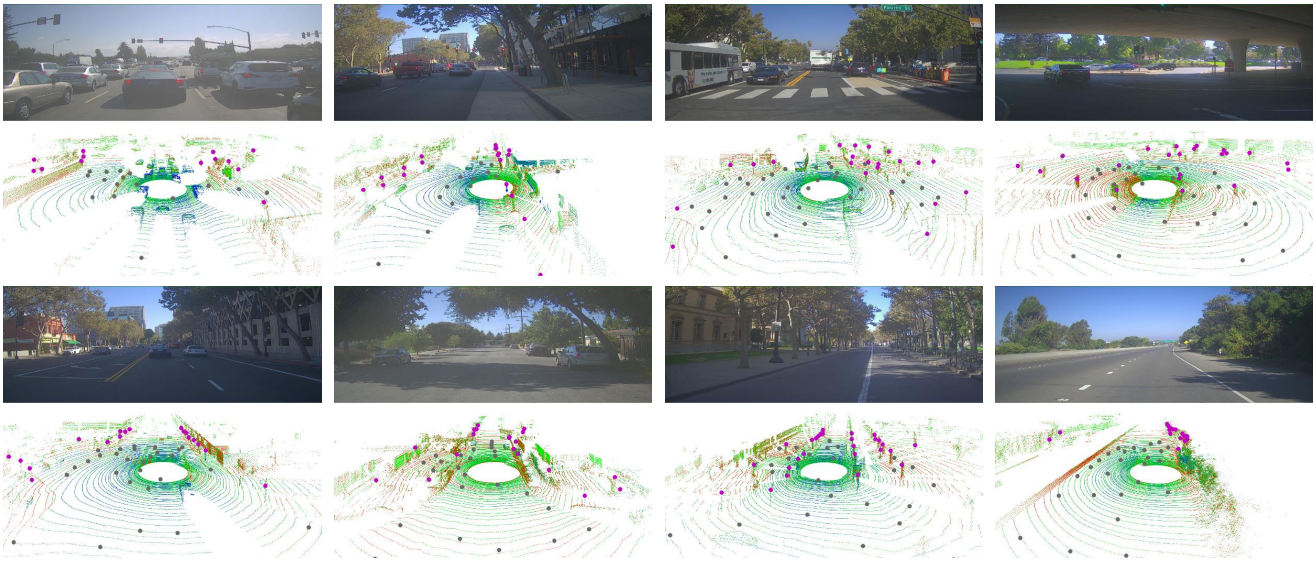

Figure 5. Visualization of the detected keypoints by the point weighting layer. The pink and grey keypoints are detected by the front and back network, respectively. The pink ones appear on stationary objects, such as tree trunks and poles. The grey ones are mostly on the ground, as expected.

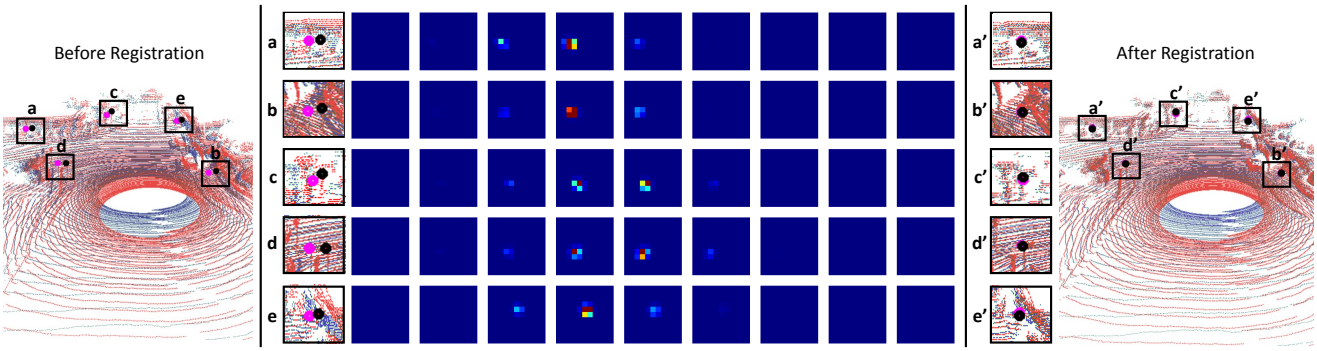

Figure 6. Illustrate the matching similarity probabilities of each keypoint to its matching candidates by visualizing them in x and y dimensions with 9 fixed z values. The black and pink points are the detected keypoints in the source point cloud and the generated ones in the target, respectively. The effectiveness of the registration process is shown on the left (before) and right (after).

source point cloud and the generated ones in the target, respectively. It is seen that the keypoints detected are sufficiently salient that the matching probabilities are concentratedly distributed.

5. Conclusion

We have presented an end-to-end framework for the point cloud registration task. The novel designs in our network make our learning-based system achieve the comparable registration accuracy to the state-of-the-art geometric methods. It has been shown that our network can learn which features are good for the registration task automatically, yielding an outlier rejection capability. Comparing to ICP and its variants, it benefits from deep features and is more robust to inaccurate initial poses. Based on the GPU acceleration in the state-of-the-art deep learning frameworks, it has good runtime efficiency that is no worse than common geometric methods. We believe that our method is attractive and has considerable potential for many applications. In a further extension of this work, we will explore ways to improve the generalization capability of the trained model with more LiDAR models in broader application scenarios.

ACKNOWLEDGMENT

This work is supported by Baidu ADT in conjunction with the Apollo Project (http://apollo.auto/). Natasha Dsouza helped with the text editing and proof reading. Runxin He and Yijun Yuan helped with the DeepVCP's deployment on clusters.

References

[1] Mikaela Angelina Uy and Gim Hee Lee. PointNetVLAD: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4470–4479, 2018. 2

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 39(12):2481–2495, 2017. 1

[3] Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992. 2, 5, 6

[4] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. 2

[5] Xinjing Cheng, Peng Wang, and Ruigang Yang. Learning depth with convolutional spatial propagation network. *arXiv preprint arXiv:1810.02695*, 2018. 1

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPF-FoldNet: Unsupervised learning of rotation invariant 3D local descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 602–618, 2018. 3

[7] Haowen Deng, Tolga Birdal, and Slobodan Ilic. PPFNet: Global context aware local features for robust 3D point matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3, 4

[8] Jean-Emmanuel Deschaud. IMLS-SLAM: scan-to-model matching based on 3D data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485. IEEE, 2018. 1

[9] Li Ding and Chen Feng. DeepMapping: Unsupervised map estimation from multiple point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 1

[10] David Droeschel and Sven Behnke. Efficient continuous-time SLAM for 3D LiDAR-based online mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9. IEEE, 2018. 1

[11] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3D point cloud registration for localization using a deep neural network auto-encoder. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4631–4640, 2017. 1, 2

[12] Pete Gadomski. C++ implementation of the coherent point drift point set registration algorithm. Available at https://github.com/gadomski/cpd, version v0.5.1. 5

[13] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3354–3361. IEEE, 2012. 2, 5

[14] Georgios Georgakis, Srikrishna Karanam, Ziyan Wu, Jan Ernst, and Jana Košecká. End-to-end learning of keypoint detector and descriptor for pose invariant 3D matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1965–1973, 2018. 2

[15] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 580–587, 2014. 1

[16] Zan Gojcic, Caifa Zhou, Jan D Wegner, and Andreas Wieser. The perfect match: 3D point cloud matching with smoothed densities. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5545–5554, 2019. 3

[17] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International Journal of Computer Vision (IJCV)*, 103(3):267–305, 2013. 6

[18] Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Training deep networks with structured layers by matrix backpropagation. *arXiv preprint arXiv:1509.07838*, 2015. 5

[19] Kaijin Ji, Huiyan Chen, Huijun Di, Jianwei Gong, Guangming Xiong, Jianyong Qi, and Tao Yi. CPFG-SLAM: a robust simultaneous localization and mapping based on LiDAR in off-road environment. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 650–655. IEEE, 2018. 1

[20] Shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, Kazuya Takeda, and Tsuyoshi Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, Nov 2015. 1

[21] Marc Khoury, Qian-Yi Zhou, and Vladlen Koltun. Learning compact geometric features. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 153–161, 2017. 2

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 1097–1105, 2012. 1

[23] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8895–8904, 2019. 3

[24] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, 2015. 1

[25] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. L3-Net: Towards learning based LiDAR localization for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2019. 1, 2, 3, 4, 5

[26] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(12):2262–2275, Dec 2010. 2, 5, 6

[27] Frank Neuhaus, Tilman Koß, Robert Kohnen, and Dietrich Paulus. MC2SLAM: Real-time inertial LiDAR odometry using two-scan motion compensation. In *Proceedings of the*

German Conference on Pattern Recognition (GCPR), pages 60–72. Springer, 2018. 1

[28] Artem L Pavlov, Grigory WV Ovchinnikov, Dmitry Yu Derbyshev, Dzmitry Tsetserukou, and Ivan V Oseledets. AA-ICP: Iterative closest point with Anderson acceleration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE, 2018. 5, 6

[29] François Pomerleau, Francis Colas, Roland Siegwart, et al. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends® in Robotics*, 4(1):1–104, 2015. 2

[30] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, July 2017. 2, 3

[31] Charles R. Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, pages 5099–5108, 2017. 2, 3, 7

[32] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. 1

[33] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3-D registration. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3212–3217, May 2009. 4

[34] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011. 5

[35] Samuele Salti, Federico Tombari, Riccardo Spezialetti, and Luigi Di Stefano. Learning a descriptor-specific 3D keypoint detector. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2318–2326, 2015. 2

[36] Ashutosh Saxena, Min Sun, and Andrew Y Ng. Make3d: Learning 3d scene structure from a single still image. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(5):824–840, 2008. 1

[37] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Proceedings of the Robotics: Science and Systems (RSS)*, 06 2009. 2, 5, 6

[38] Takaaki Shiratori, Jérôme Berclaz, Michael Harville, Chintan Shah, Taoyu Li, Yasuyuki Matsushita, and Stephen Shiller. Efficient large-scale point cloud registration using loop closures. In *Proceedings of the International Conference on 3D Vision (3DV)*, pages 232–240. IEEE, 2015. 1

[39] Todor Stoyanov, Martin Magnusson, Henrik Andreasson, and Achim J Lilienthal. Fast and accurate scan registration through minimization of the distance between compact 3D NDT representations. *The International Journal of Robotics Research (IJRR)*, 31(12):1377–1393, 2012. 2, 5, 6

[40] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. DeMoN: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5038–5047, 2017. 1

[41] Martin Velas, Michal Spanel, Michal Hradis, and Adam Herout. CNN for IMU assisted odometry estimation using velodyne LiDAR. In *Proceedings of the IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 71–77. IEEE, 2018. 2

[42] Guowei Wan, Xiaolong Yang, Renlan Cai, Hao Li, Yao Zhou, Hao Wang, and Shiyu Song. Robust and precise vehicle localization based on multi-sensor fusion in diverse city scenes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4670–4677. IEEE, 2018. 1

[43] Jay M Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton, Rahul Chipalkatty, Mitchell Hebert, David MS Johnson, et al. SegICP: Integrated deep semantic segmentation and pose estimation. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 5784–5789. IEEE, 2017. 1

[44] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 38(11):2241–2254, 2015. 2

[45] Sheng Yang, Xiaoling Zhu, Xing Nian, Lu Feng, Xiaozhi Qu, and Teng Mal. A robust pose graph approach for city scale LiDAR mapping. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 1175–1182. IEEE, 2018. 1

[46] Zi Jian Yew and Gim Hee Lee. 3DFeat-Net: Weakly supervised local 3D features for point cloud registration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 630–646. Springer, 2018. 2, 3, 5, 6

[47] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. *arXiv preprint arXiv:1812.06264*, 2018. 1

[48] Keisuke Yoneda, Hossein Tehrani, Takashi Ogawa, Naohisa Hukuyama, and Seiichi Mita. LiDAR scan feature for localization with highly precise 3-D map. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV)*, pages 1345–1350, June 2014. 1

[49] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 4

[50] Ji Zhang and Sanjiv Singh. LOAM: LiDAR odometry and mapping in real-time. In *Proceedings of the Robotics: Science and Systems (RSS)*, volume 2, page 9, 2014. 1

[51] Huizhong Zhou, Benjamin Ummenhofer, and Thomas Brox. DeepTAM: Deep tracking and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 822–838, 2018. 1

DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration

ICCV 2019 Supplementary Material

Weixin Lu Guowei Wan Yao Zhou Xiangyu Fu Pengfei Yuan Shiyu Song*
Baidu Autonomous Driving Technology Department (ADT)
{luweixin, wanguowei, zhouyao, fuxiangyu, yuanpengfei, songshiyu}@baidu.com

1. Implementation Details

We introduce our implementation details in this section.

In the FE layer, a simplified PointNet++ is applied, in which only three *set abstraction* layers with a single *scale grouping* layer are used to sub-sample points into groups with sizes 4096, 1024, 256, and the MLPs of three hierarchical PointNet layer are 32×32, 32×64, 64×64 in the sub-sampling stage, and 64×64, 32×32, $32 \times 32 \times 32$ in the up-sampling stage. This is followed by a fully connected layer with 32 kernels and a dropout layer with the keeping probability as 0.7 to avoid overfitting. The MLP in the point weighting layer is $16 \times 8 \times 1$, and only the top $N = 64$ points are selected in the source point cloud according to their learned weights in the descending order. The searching range d and the number of neighboring points K to be collected in the DFE step are set to be $1m$ and 32, respectively. In the mini-PointNet structure of the DFE layer, the MLP is $32 \times 32 \times 32$. The 3D CNNs settings in the CPG step are Conv3d (16, 3, 1) - Conv3d(4, 3, 1) - Conv3d (1, 3, 1). The grid voxels are set as $(\frac{2\times2.0}{0.4}+1, \frac{2\times2.0}{0.4}+1, \frac{2\times2.0}{0.25}+1)$.

The proposed network is trained with the batch size as 1, the learning rate as 0.01 and the decay rate as 0.7 with the decay step to be 10000. During the training stage, we conduct the data augmentation and supervised training by adding a uniformly distributed random noise of $[0.0 \sim 1.0]m$ in the x, y and z dimensions, and $[0 \sim 1.0]^\circ$ in the $roll$, yaw, and $pitch$ dimensions to the given ground truth. We randomly divide the dataset into the training and validation set, yielding the ratio of training to validation as 4 to 1. We stop at 200 epochs when there is no performance gain.

We list the configuration settings of all the baseline methods in Table 1. We use these parameters in the experiments across all the KITTI, Apollo-SouthBay and 3DMatch datasets.

2. More Implementation Details of the AA-ICP

The Euler angles are used as the rotation representation when solving the optimal weighting terms α based on the history of the latest iterations and residuals in the original implementation of the AA-ICP [2]. It is known that the Euler angles have the singularity problem. In rare cases when we test using the Apollo-SouthBay dataset, we noted that it caused the estimated Euler angles to flip across the axes in our experimental results using the original implementation. Certainly, this results in the algorithm failing to converge. Therefore, we modified the implementation by limiting the valid value range of the Euler angles (α, β, γ) to be [-90, 90]°, [-180, 180]° and [-180, 180]° and the interpolation between the two orientations will always use the shortest path. The comparison of the results using the original and modified version is shown in Table 2.

3. More Results on Other Datasets

We evaluate the performance of the proposed DeepVCP using datasets from the sensors other than the Velodyne HDL64 LiDAR, including the 3DMatch [3] and the TLS [1] dataset.

3.1. The 3DMatch Dataset

The point clouds in the 3DMatch dataset are collected by RGB-D sensors (e.g. Microsoft Kinect, Intel RealSense). Most methods do not converge when we use the original point cloud pairs in the 3DMatch dataset, due to the very small overlap between each pair. Therefore, we synthesize the new data pair by downsampling one of the point clouds to approximately $150,000$ points, duplicating and shifting it with large random initial transformations, and finally adding random errors of less than $0.01m$ to each point in the new point cloud. With the help of the sequential storage in 3DMatch, we take the first 90% and the last 90% points in each point cloud to let the two generated point clouds have an approximate 90% overlap. To conduct a comprehensive evaluation, we gradually increase the initial pose errors during our experiments. The complete results are summarized in in Tab. 3 and Tab. 4. We consider the results with the maximum errors larger than $5.0m$ and 80.0° as non-convergence, marked as "N/A" in the table. The mean value calculation is computed by ignoring the non-convergence cases to show the average performance when they work normally.

The searching range d and the grid voxel size are set as $0.8m$ and $(\frac{2\times4.0}{0.5}+1, \frac{2\times4.0}{0.5}+1, \frac{2\times4.0}{0.5}+1)$, respectively. But to keep CPD not intractable, we once again downsample the point clouds using a voxel size of $0.015m$ leaving about $50,000$ points. All other settings are consistent with the experiments in the main conference proceeding. As can be seen, the performance of the ICP family and the NDT gradually deteriorates as the initial pose errors increase while the CPD and 3DFeat-Net methods are still stable. The CPD and 3DFeat-Net methods match globally so they are expected to be insensitive to the initial errors. By using deep features that are powerful enough to find correct matching keypoints, our DeepVCP achieves good accuracy under large initial errors.

3.2. The TLS dataset

The TLS dataset [1] is from Terrestrial Laser Scanners (TLS), e.g. a Rigel LiDAR. We downsample the original point clouds with a voxel grid of $0.0625m$ and leave about

*Author to whom correspondence should be addressed

	variable	value
Registration (Base Class in PCL)	nr_iterations_	0
	max_iterations_	600
	ransac_iterations_	0
	transformation_epsilon_	1e-6
	transformation_rotation_epsilon_	0.0
	inlier_threshold_	0.05
	min_number_correspondences_	3 (4 for G-ICP)
	euclidean_fitness_epsilon_	1e-4
	corr_dist_threshold_	1.0 (0.5 for ICP-Po2Pl)
IterativeClosestPoint (ICP)	x_idx_offset_	0
	y_idx_offset_	0
	z_idx_offset_	0
	nx_idx_offset_	0
	ny_idx_offset_	0
	nz_idx_offset_	0
GeneralizedIterativeClosestPoint (G-ICP)	k_correspondences_	20
	gicp_epsilon_	0.001
	rotation_epsilon_	2e-3
	mahalanobis_	0
	max_inner_iterations_	20
NormalDistributionsTransform (NDT)	resolution_	1.0 (0.1 for 3DMatch dataset)
	step_size_	0.1 (0.9 for 3DMatch dataset)
	outlier_ratio_	0.55
AndersonIterativeClosestPoint (AA-ICP)	alpha_limit_min_	-10
	alpha_limit_max_	10
	beta_	1.0
	small_step_threshold_	3
	error_overflow_threshold_	0.05
Transform (CPD)	m_correspondence	false
	m_max_iterations	200
	m_normalize	true
	m_outliers	0.2
	m_sigma2	0.0
	m_tolerance	1e-5
Rigid (CPD)	m_reflections	false
	m_scale	false

Table 1. The configuration settings of the baseline methods in the experiments. The ICP family and the NDT all inherit from a common base class in which the parameters are shared across all these methods.

Dataset	Method	Angular Error(°)		Translation Error(m)	
		Mean	Max	Mean	Max
KITTI	Original	0.152	1.406	0.096	1.813
	Modified	0.145	1.406	0.088	2.02
Apollo-SouthBay	Original	0.363	179.9	0.119	5.675
	Modified	0.054	1.087	0.109	5.243

Table 2. Comparison of the original and modified version of the AA-ICP implementation. The issue of irregularly large maximum angle errors in rare cases of Apollo-SouthBay dataset is resolved.

130,000 points as our input. We once again adjust the searching range d and the grid voxels' size accordingly, to $0.5m$ and $(\frac{2\times2.0}{0.4}+1, \frac{2\times2.0}{0.4}+1, \frac{2\times2.0}{0.4}+1)$. Three different scenes in the dataset are evaluated. Please note, that we certainly can not train the network adequately with this little amount of data. It is highly possible that the results are overfitted, and therefore are only visually evaluated. The registration results of a courtyard, an office and a forest are shown in Figure 1, Figure 2 and Figure 3, respectively.

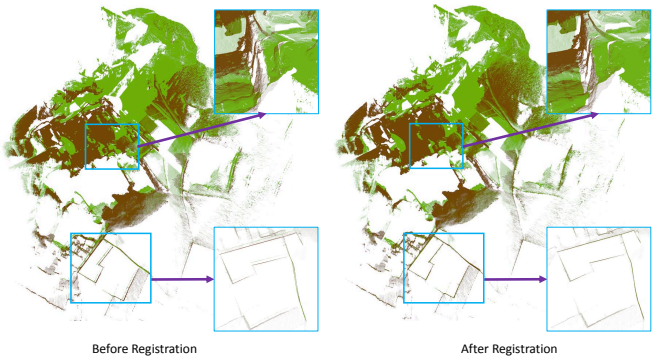

Figure 1. The registration result of a courtyard from the TLS dataset. The point cloud pair is differently colored. From the zoomed figures, we observe that the mountain terrain in the center and the walls of the building at the bottom are aligned well after the registration.

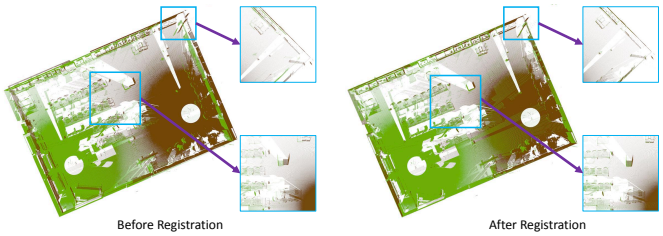

Figure 2. The registration result of an office from the TLS dataset. As shown in the zoomed figures, the desks, chairs, and walls are aligned well after the registration.

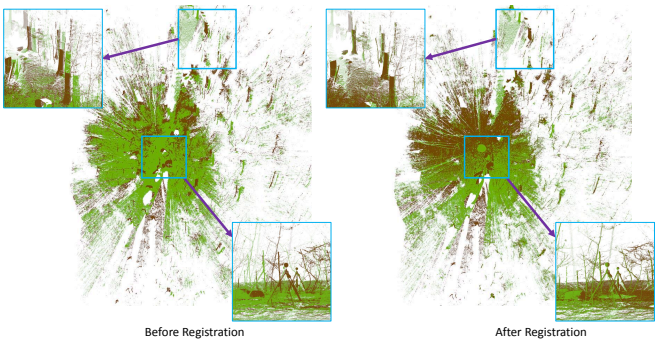

Figure 3. The registration result of a forest from the TLS dataset. From the zoomed figures, the trunks and the tripod are aligned well after the registration.

References

[1] TLS Dataset. http://www.igp.ethz.ch/, 2017.

[2] Artem Leonidovich Pavlov, George Ovchinnikov, D. Yu. Derbyshev, Dzmitry Tsetserukou, and Ivan Oseledets. AA-ICP: Iterative closest point with Anderson acceleration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6, May 2018.

[3] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Initial Error		ICP-Po2Po (m)	ICP-Po2Pl (m)	G-ICP (m)	AA-ICP (m)	NDT-P2D (m)	CPD (m)	3DFeat-Net (m)	Ours (m)
($0.1m$, 1.0°)	Mean	0.023	0.012	0.012	0.018	0.005	0.004	0.072	0.019
	Max	0.092	0.053	0.065	0.097	0.017	0.024	0.257	0.073
($0.2m$, 2.0°)	Mean	0.024	0.012	0.015	0.018	0.009	0.004	0.072	0.021
	Max	0.091	0.054	0.120	0.097	0.206	0.024	0.257	0.065
($0.3m$, 3.0°)	Mean	0.025	0.012	0.013	0.019	0.052	0.004	0.072	0.018
	Max	0.092	0.056	0.044	0.097	0.800	0.024	0.257	0.071
($0.4m$, 4.0°)	Mean	0.026	0.012	0.014	0.019	0.170	0.004	0.072	0.019
	Max	0.093	0.055	0.055	0.097	1.219	0.024	0.257	0.052
($0.5m$, 5.0°)	Mean	0.026	0.012	0.014	0.021	0.282	0.004	0.072	0.019
	Max	0.093	0.055	0.051	0.258	1.535	0.024	0.257	0.060
($0.6m$, 6.0°)	Mean	0.026	0.012	0.015	0.019	0.478	0.004	0.072	0.018
	Max	0.093	0.056	0.062	0.119	4.804	0.024	0.257	0.053
($0.7m$, 7.0°)	Mean	0.026	0.012	0.015	0.019	0.514	0.004	0.072	0.018
	Max	0.094	0.055	0.064	0.098	N/A	0.024	0.257	0.064
($0.8m$, 8.0°)	Mean	0.026	0.012	0.017	0.018	0.709	0.004	0.072	0.017
	Max	0.094	0.055	0.153	0.098	4.889	0.024	0.257	0.057
($0.9m$, 9.0°)	Mean	0.026	0.012	0.020	0.024	0.817	0.004	0.072	0.017
	Max	0.093	0.055	0.522	0.640	4.818	0.024	0.257	0.052
($1.0m$, 10.0°)	Mean	0.026	0.035	0.026	0.024	1.007	0.004	0.072	0.016
	Max	0.094	2.340	0.599	0.585	4.850	0.024	0.257	0.050
($1.1m$, 11.0°)	Mean	0.027	0.075	0.022	0.120	0.949	0.004	0.072	0.019
	Max	0.093	2.502	0.390	5.147	4.120	0.024	0.257	0.077
($1.2m$, 12.0°)	Mean	0.089	0.059	0.057	0.152	1.243	0.004	0.072	0.018
	Max	4.334	2.508	2.312	4.932	4.915	0.024	0.257	0.062
($1.3m$, 13.0°)	Mean	0.114	0.162	0.029	0.118	1.349	0.004	0.072	0.018
	Max	4.336	4.804	0.805	4.715	4.898	0.024	0.257	0.066
($1.4m$, 14.0°)	Mean	0.155	0.192	0.116	0.226	1.332	0.004	0.072	0.018
	Max	4.336	4.822	3.436	4.715	4.890	0.024	0.257	0.077
($1.5m$, 15.0°)	Mean	0.161	0.163	0.232	0.310	1.379	0.004	0.072	0.018
	Max	4.713	4.206	4.669	4.716	N/A	0.024	0.257	0.052
($1.6m$, 16.0°)	Mean	0.197	0.094	0.250	0.342	1.590	0.004	0.072	0.019
	Max	4.712	3.052	4.505	4.715	4.884	0.024	0.257	0.055
($1.7m$, 17.0°)	Mean	0.229	0.286	0.238	0.323	1.678	0.004	0.072	0.017
	Max	N/A	N/A	4.694	N/A	4.952	0.024	0.257	0.063
($1.8m$, 18.0°)	Mean	0.276	0.345	0.314	0.241	1.681	0.004	0.072	0.017
	Max	6.365	N/A	4.704	4.714	4.881	0.024	0.257	0.059
($1.9m$, 19.0°)	Mean	0.244	0.380	0.316	0.376	1.768	0.004	0.072	0.017
	Max	N/A	N/A	4.547	N/A	N/A	0.024	0.257	0.048
($2.0m$, 20.0°)	Mean	0.387	0.343	0.360	0.325	1.826	0.004	0.072	0.017
	Max	N/A	N/A	4.505	N/A	4.885	0.024	0.257	0.046

Table 3. The performance evaluation given different initial errors by illustrating the translational errors. The performance of the ICP family and the NDT gradually deteriorates as the initial pose errors increase. the CPD and 3DFeat-Net methods are not influenced by the initial errors as they are global methods. Our DeepVCP achieves high accuracy consistently demonstrating the matching robustness using deep features.

Initial Error		ICP-Po2Po (deg)	ICP-Po2Pl (deg)	G-ICP (deg)	AA-ICP (deg)	NDT-P2D (deg)	CPD (deg)	3DFeat-Net (deg)	Ours (deg)
($0.1m$, 1.0°)	Mean	0.594	0.285	0.254	0.401	0.104	0.070	1.654	0.465
	Max	2.470	1.264	1.247	2.451	0.346	0.427	6.129	1.538
($0.2m$, 2.0°)	Mean	0.636	0.291	0.291	0.405	0.213	0.070	1.654	0.513
	Max	2.623	1.348	1.558	2.464	7.420	0.427	6.129	1.359
($0.3m$, 3.0°)	Mean	0.660	0.293	0.270	0.404	0.856	0.070	1.654	0.446
	Max	2.658	1.439	1.097	2.457	15.47	0.427	6.129	1.419
($0.4m$, 4.0°)	Mean	0.677	0.294	0.289	0.419	3.238	0.070	1.654	0.448
	Max	2.628	1.453	1.068	2.465	33.00	0.427	6.129	1.198
($0.5m$, 5.0°)	Mean	0.690	0.298	0.288	0.480	5.819	0.070	1.654	0.465
	Max	2.627	1.458	1.071	7.047	50.14	0.427	6.129	1.470
($0.6m$, 6.0°)	Mean	0.701	0.302	0.305	0.440	7.577	0.070	1.654	0.452
	Max	2.613	1.457	1.023	4.547	N/A	0.427	6.129	1.158
($0.7m$, 7.0°)	Mean	0.702	0.310	0.305	0.401	9.042	0.070	1.654	0.456
	Max	2.603	1.457	1.060	2.448	N/A	0.427	6.129	1.416
($0.8m$, 8.0°)	Mean	0.708	0.313	0.354	0.380	9.538	0.070	1.654	0.437
	Max	2.633	1.453	2.806	2.543	N/A	0.427	6.129	1.220
($0.9m$, 9.0°)	Mean	0.706	0.309	0.320	0.698	12.12	0.070	1.654	0.446
	Max	2.659	1.452	1.777	30.93	N/A	0.427	6.129	1.427
($1.0m$, 10.0°)	Mean	0.700	0.729	0.595	0.732	15.76	0.070	1.654	0.406
	Max	2.679	42.69	24.84	33.27	N/A	0.427	6.129	1.386
($1.1m$, 11.0°)	Mean	0.706	1.791	0.623	0.692	14.49	0.070	1.654	0.440
	Max	2.697	63.15	26.29	N/A	N/A	0.427	6.129	1.423
($1.2m$, 12.0°)	Mean	1.077	1.680	1.077	1.166	17.92	0.070	1.654	0.458
	Max	N/A	63.15	N/A	N/A	N/A	0.427	6.129	1.785
($1.3m$, 13.0°)	Mean	1.517	2.147	0.648	1.161	17.68	0.070	1.654	0.391
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.275
($1.4m$, 14.0°)	Mean	1.516	1.935	1.872	1.292	17.48	0.070	1.654	0.445
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.416
($1.5m$, 15.0°)	Mean	1.513	1.875	1.513	0.717	18.43	0.070	1.654	0.440
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.176
($1.6m$, 16.0°)	Mean	1.078	1.763	2.306	1.760	17.27	0.070	1.654	0.481
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.330
($1.7m$, 17.0°)	Mean	1.874	1.770	2.722	1.787	16.85	0.070	1.654	0.446
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.338
($1.8m$, 18.0°)	Mean	2.263	1.836	3.922	0.894	20.44	0.070	1.654	0.419
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.340
($1.9m$, 19.0°)	Mean	2.275	2.226	2.628	3.124	21.49	0.070	1.654	0.423
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.142
($2.0m$, 20.0°)	Mean	2.725	1.665	2.516	1.782	18.702	0.070	1.654	0.423
	Max	N/A	N/A	N/A	N/A	N/A	0.427	6.129	1.140

Table 4. The performance evaluation given different initial errors by illustrating angular errors. The performance of the ICP family and the NDT gradually deteriorates as the initial pose errors increase. The CPD and 3DFeat-Net methods are not influenced by the initial errors as they are global methods. Our DeepVCP achieves high accuracy consistently, demonstrating the matching robustness using deep features.