SuperLine3D: Self-supervised Line Segmentation and Description for LiDAR Point Cloud

Xiangrui Zhao[1,2], Sheng Yang[2], Tianxin Huang[1], Jun Chen[1], Teng Ma[2], Mingyang Li[2], and Yong Liu[1,†]

[1] APRIL Lab, Zhejiang University, China
{xiangruizhao, 21725129, junc}@zju.edu.cn, yongliu@iipc.zju.edu.cn
[2] Autonomous Driving Lab, DAMO Academy, China
{shengyang93fs, mingyangli009}@gmail.com, damon.mt@alibaba-inc.com

Abstract. Poles and building edges are frequently observable objects on urban roads, conveying reliable hints for various computer vision tasks. To repetitively extract them as features and perform association between discrete LiDAR frames for registration, we propose the first learning-based feature segmentation and description model for 3D lines in LiDAR point cloud. To train our model without the time consuming and tedious data labeling process, we first generate synthetic primitives for the basic appearance of target lines, and build an iterative line auto-labeling process to gradually refine line labels on real LiDAR scans. Our segmentation model can extract lines under arbitrary scale perturbations, and we use shared EdgeConv encoder layers to train the two segmentation and descriptor heads jointly. Base on the model, we can build a highly-available global registration module for point cloud registration, in conditions without initial transformation hints. Experiments have demonstrated that our line-based registration method is highly competitive to state-of-the-art point-based approaches. Our code is available at `https://github.com/zxrzju/SuperLine3D.git`.

Keywords: 3D Line Feature, Point Cloud Registration

1 Introduction

Point cloud registration is an essential technique for LiDAR-based vehicle localization on urban road scenes [28]. Considering recent researches [18,15], the SLAM community [19] divides these algorithms into two categories regarding their purpose, as *local* and *global* search methods, respectively. The *local* search category [6,7] typically constructs a non-convex optimization problem by greedily associating nearest entities to align. This often relies on a good initial guess, and thus mostly used for incremental positioning modules such as the LiDAR odometry [41] and map-based localization [32]. The *global* search category is used for less informative conditions, i.e., relocalization and map initialization

† indicates the corresponding author.

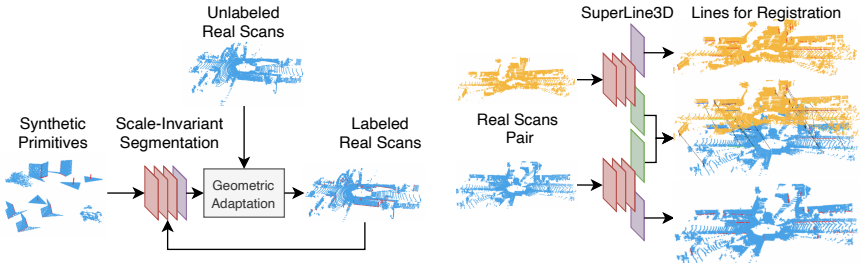
problems when the initial guess is not reliable and large positional and rotational change exists. Since nearest neighbor search methods cannot find correct matching pairs in the Euclidean space, *global* search algorithms choose to extract distinct entities and construct feature descriptors [45], to establish matches in the description space.

There exists a variety of classical hand-crafted features (e.g., FPFH [33]) for global search and registration, and recent learning-based methods [43] have improved the registration accuracy and success rate. However, the performance of some methods [25,27] severely drops when adapting to real LiDAR scans, because the density of scanned points is inversely proportional to the scanning distance, and thus influences the coherence of point description. Considering such limitation of a single point, we propose an idea of using structural lines, analogously as previous approaches proposed for images [16,40], to see whether a relatively stable descriptor can be concluded through a semantically meaningful group of scattered points.

In typical LiDAR point cloud scanned from urban road scenes, there are three categories of lines. 1) Intersection of planes, e.g., edge of two building facades and curbs. 2) Standalone pole objects, e.g., street lamps and road signs alongside the road. 3) Virtual line consists of edge points across multiple scan rings, generated by ray-obstacle occlusions. While the last category is not repeatable and thus inappropriate for localization, the first two types are practical landmarks suitable to be extracted and described. Since these line segments are larger targets compared to point features, they have a higher chance to be repeatably observed. Moreover, the concluded position of each line is more precise to a single corresponding point between frames due to the limited scanning resolution, which causes sampling issues.

In this paper, we propose a self-supervised learning method for line segmentation and description on LiDAR scans (Fig. 1). Following the training procedure of SuperPoint [11] to solve the lack of publicly available training data, we choose to train our line extraction model, by first construct limited synthetic data and then perform auto labeling on real scans. By sharing point cloud encoding layers and use two separate branches for decoding and application headers, we are able to jointly train two tasks on those generated data. We view such a pipeline to train and use line features for the scan registration purpose as the key contribution of our work, which includes:

- From the best of our knowledge, we propose the first learning-based line segmentation and description for LiDAR scans, bringing up an applicable feature category for global registration.
- We propose a line segment labeling method for point clouds, which can migrate the model learned from synthetic data to real LiDAR scans for automatic labeling.
- We explore the scale invariance of point cloud features, and provide a feasible idea for improving the generalization of learning-based tasks on the point cloud under scale perturbations by eliminating the scale factor in Sim(3) transformation.



(a) Automatic Line Segments Labeling (b) Line Segmentation and Description

Fig. 1. Pipeline overview. a): We train a scale-invariant segmentation on the synthetic data and get the precise line segment labels after multiple geometric adaptation iterations. b): We simultaneously train segmentation and description on labeled LiDAR scans, where red, purple, and green layers stand for encoders, segmentation header, and description header, respectively.

Extensive experimental results have shown that our line-based registration can maintain high success rate and accuracy under large-angle perturbations, and the trained model on one real scans dataset is highly adaptable to other urban scene datasets.

2 Related Work

Learning-based Point Cloud Registration. In recent researches, there are a variety of learning-based approaches proposed for registering point clouds, and we can divide them into two groups considering whether explicit features have been extracted. End-to-end approaches use ground-truth transformation in loss calculation, and predict the transformation directly through the network: FMR [17] registers point clouds by minimizing feature-metric loss, and PCR-Net [34] evaluate the similarity of PointNet [30] features and regresses poses through fully connected layers directly. These trained end-to-end models work well on tested sequences, but they are facing a practical problem on how to perform a joint state estimation in a multi-sensor fusion system [12]. Nevertheless, knowledge of these models are hardly adaptable to different motion scheme and other datasets. Therefore, methods with explicit feature extraction and description are still an active branch in the SLAM community.

Registration with Explicit Features. Start with hand-crafted features (e.g., FPFH [33] and ISS [44]) concluding local patch appearances of point clouds, methods of extracting and describing explicit features mainly aim at the saliency of entities and coherency of description. While hand-crafted features are mostly designed for evenly sampled clouds, learning-based features [21,10,25,9,22,4] have better robustness and generalization, once trained on the target LiDAR scan datasets. D3Feat [5] uses kernel-based convolution [36] to learn feature detection

and description. SpinNet [3] builds a rotation-invariant local surface descriptor through novel spatial point transformer and 3D cylindrical convolutional layer. Both D3Feat [5] and SpinNet [3] are state-of-the-art learning-based point features, but they still suffer from the inherent problem of point features, and thus requires sample consensus as a post pruning procedure to filter correct feature associations.

Line Features for SLAM. Image based line-aware approaches for detection (e.g., LSD [37], EDLines [2], and TP-LSD [16]), description (e.g., LBD [42] and SOLD2 [26]), and systematical SLAM designs (e.g., PL-SLAM [29]) have been well studied in recent years, whereas LiDAR scan based extraction and description methods, although heavily used in modern LiDAR SLAM approaches (e.g., LOAM [41] and LeGO-LOAM [35]), are under explored. To the best of our knowledge, we found Lu et al. [24] have proposed a 3D line detection method through projecting LiDAR points onto image, and thus convert the task into a 2D detection problem. Chen et al. [8] based on this work [24] to carry out a line-based registration approach for structural scenes. However, their limitations are two folds: 1) only work on organized point clouds, and 2) have not addressed line description and thus not suitable for global search registration problems. In contrast, we follow the idea of descriptor conclusion from SOLD2 [26], which has been proven to be useful in our paper for the coherency of describing a group of points.

3 Method

Considering the lack of available labeled line datasets of LiDAR scans, we follow the self-supervised idea of SuperPoint [11], to train our line segmentation model, by first constructing a simple synthetic data to initialize a base model, and then refining the model iteratively with auto-labeled real LiDAR scans from geometric adaptation (Sec. 3.1). After that, we gather line correspondences between different LiDAR scans, and jointly train the line segmentation and description in an end-to-end approach (Sec. 3.2).

3.1 Line Segmentation Model

Synthetic Data Generation. As discussed above in Sec. 1, there are two types of reliable line segments to detect: 1) intersection between planes, and 2) poles. Hence, we choose to use the following two mesh primitives shown in Fig. 2(a) for simulating their local appearances, respectively. These two mesh models are first uniformly sampled into 4,000 points as Fig. 2(b), with 5% relative 3-DOF positional perturbance added for each point. Then, to simulate possible background points nearby, we randomly cropped 40 basic primitives with each containing 1,000 points from real scans [14], and put them together to compose the final synthetic data. In total, we generated 5,000 synthetic point clouds with 5,000 points per each cloud.

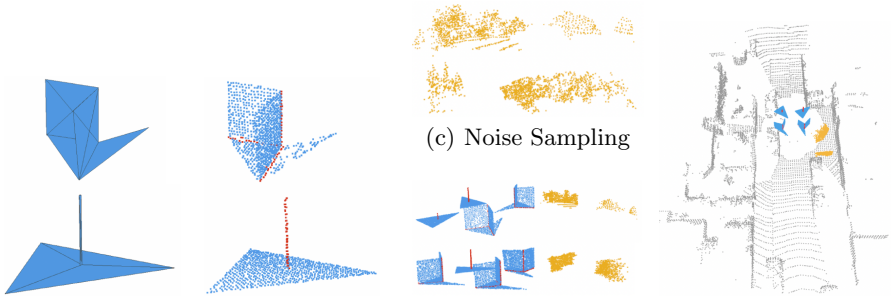


Fig. 2. Synthetic data generation steps. We generate synthetic data through sampling primitive mesh models and augmenting real scan scattered points as noises.

Scale-Invariant Line Segmentation. We treat line detection as a point cloud segmentation problem, and the main challenge is the primitive scaling issue: In a real LiDAR frame, the density of points decreases with the scanning distance, and the voxel grid downsampling cannot fully normalize the density when the target feature is far away from the sensor. Moreover, our synthetic data generation also did not consider the scale of line segments (as visualized in Fig. 2(e) when put together). If such an issue is not handled, the model will not produce reasonable predictions when the training and test data are on different scales.

To address this issue, our network obtains scale invariance by eliminating the scale factor s of the Sim(3) transformation and using relative distances, as:

$$\begin{aligned} p' &= s \cdot \mathbf{R}p + t, \\ f &= \frac{\sum_i (p' - p'_i)}{\sum_i \|p' - p'_i\|} = \frac{s \cdot \sum_i \mathbf{R}(p - p_i)}{s \cdot \sum_i \|p - p_i\|}. \end{aligned} \tag{1}$$

In Eq. 1, we search $k = 20$ nearest points $\{p_1, p_2, ..., p_k\}$ of a point p, and calculate the scale-invariant local feature f as the ratio of the Manhattan distance to the Euclidean distance between p and its neighbors. The trade-off of such a feature definition is that f cannot reflect the position of the original point in the Euclidean space, so the transformation has information loss. Such an influence are further evaluated in Sec. 4.3.

Model architecture. We choose DGCNN [39] as our backbone, since it directly encodes points and their nearest neighbors without complicated operations. Eq. 2 shows its local feature encoding function called *EdgeConv* [39], where $\mathbf{x}_j$ is the j-th feature, ${}^S\mathbf{x}_i$ is the neighbor of the $\mathbf{x}_j$ in the feature space S, and h is the learnable model.

$$h\left(\mathbf{x}_j, {}^S\mathbf{x}_i\right) = \bar{h}\left(\mathbf{x}_j, {}^S\mathbf{x}_i - \mathbf{x}_j\right). \tag{2}$$

In the first *EdgeConv* layer, x represents the point coordinates in Euclidean space. In our implementation, we gather $k = 20$ nearest neighbors of each points

and calculate scale-invariant feature f. Then we turn the first *EdgeConv* layer into:

$$h\left(f_j, {}^{E}f_i\right) = \bar{h}\left(f_j, {}^{E}f_i - f_j\right). \tag{3}$$

It replaces the coordinates in the Euclidean space with scale-invariant feature f, but ${}^{E}f_i$ is still the feature of i-th neighbor of point p_j in Euclidean space, not the neighbor of f_j in feature space. Since part of the information in the original Euclidean space has been lost when generating scale-invariant features, preserving the neighbor relationship in the original Euclidean space can reduce further information loss.

Automatic Line Segment Labeling. There is no available labeled line dataset of LiDAR scans, and performing manual labeling on point clouds is difficult. Hence, we build an automatic line labeling pipeline (Fig. 3). Inspired by homographic adaptation in SuperPoint [11], we perform geometric adaptation on LiDAR scans. First, we train a scale-invariant segmentation model purely on the synthetic data, and apply 2D transformations with a uniform distribution of $20m$ in XOY and 360° in yaw to the LiDAR scans. Then, we use the trained model to predict labels on the perturbed data, aggregate the scan labels from all the perturbations and take the points that are predicted more than 80% belonging to lines as candidate points. To cluster binary points into lines, we use the region-growth algorithm. The connectivity between points is defined through a $0.5m$ KD-Tree radius search. We use the labeled points as seeds, grow to nearby labeled points, and fit lines. Once such line segments are extracted, we continue to refine the segmentation model on the obtained labeled LiDAR scans. We repeat the geometric adaptation 3 times to generate 12,989 automatically labeled LiDAR frames on the KITTI odometry sequences [14].

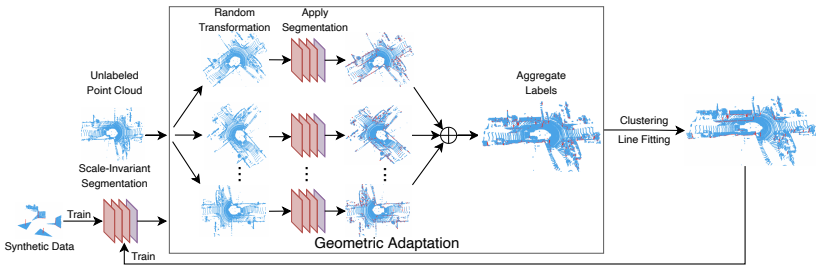


Fig. 3. Automatic line labeling pipeline. We use geometric adaptation and line fitting to reduce the network prediction noise and improve model accuracy on real LiDAR scans through iterative training.

3.2 Joint Training of Line Segmentation and Description

Definition of Line Descriptors. Different from the geometry definition which only requires two endpoints of a line segment. A descriptor for each line should

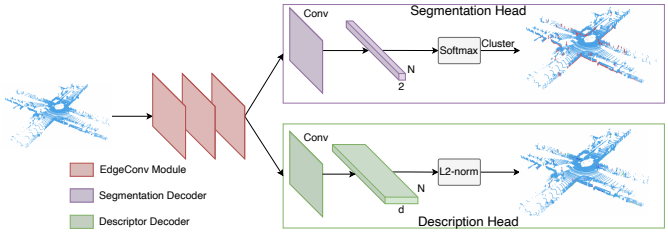


Fig. 4. Network architecture. The network uses the EdgeConv [39] module to extract features. The segmentation head and the description head predict the label and descriptor for each point, respectively.

convey local appearances through its all belonged points, since observed end points may be varied between frames due to possible occlusions. Therefore, we define the descriptor as an average of its all belonged points.

Network Architecture. Our network structure (Fig. 4) consists of a stacked three *EdgeConv* [39] layers for feature encoding, and two decoders for line segmentation and description, respectively. Each *EdgeConv* layer outputs a $N \times 64$ tensor used for 3-layer segmentation and description after a $MaxPooling$ layer. We use *ReLU* for activation. The segmentation head turns the feature vector to a tensor sized $N \times 2$ after convolution (N for the number of input points), and then obtains a boolean label per each point through a *Softmax* layer, to predict whether it belongs to a line. The descriptor head outputs a tensor sized $N \times d$, and then performs *L2-Norm* to get a d-dimensional descriptor.

Loss Functions. Our segmentation loss $\mathbf{L}_{seg}$ is a standard cross-entropy loss, and we follow [38] and [5] to build a discrimitive loss for the descriptor. In detail, we first use the line segment label to get the mean descriptor μ of each line segment, and then use the $\mathbf{L}_{same}$ for each line to pull point descriptors towards μ. The $\mathbf{L}_{diff}$ is proposed to make the descriptors of different lines repel each other. In addition for a point cloud pair, we calculate the matched loss $\mathbf{L}_{match}$ and the loss between the non-matched lines $\mathbf{L}_{mismatch}$. Each term can be written as follows:

$$
\begin{aligned}
\mathbf{L}_{same} &= \frac{1}{\mathbf{N}} \cdot \sum_{i}^{\mathbf{N}} \left(\frac{1}{|\mathbf{K}_i|} \cdot \sum_{j}^{\mathbf{K_i}} \left[\|\mu_i - d_j\|_1 - \delta_{\mathrm{s}} \right]_+^2 \right), \\
\mathbf{L}_{diff} &= \frac{1}{|\mathbf{C}_\mathbf{N}^2|} \cdot \sum_{\langle i_a, i_b \rangle}^{\mathbf{C}_\mathbf{N}^2} \left[2\delta_{\mathrm{d}} - \|\mu_{i_A} - \mu_{i_B}\|_1 \right]_+^2, \\
\mathbf{L}_{match} &= \frac{1}{\mathbf{N}} \cdot \sum_{i}^{\mathbf{N}} \left[\|\mu_i - \mu_i'\|_1 - \delta_{\mathrm{s}} \right]_+^2, \\
\mathbf{L}_{mismatch} &= \frac{1}{|\mathbf{C}_\mathbf{N}^2|} \cdot \sum_{\langle i_a, i_b \rangle}^{\mathbf{C}_\mathbf{N}^2} \left[2\delta_{\mathrm{d}} - \left\|\mu_{i_A} - \mu_{i_B}'\right\|_1 \right]_+^2,
\end{aligned}
\tag{4}
$$

where $\mathbf{N}$ is the number of detected lines and $\mathbf{C}_{\mathbf{N}}^{2}$ stands for all pairs of two lines. i and j are two iterators, for lines and points on a line, respectively. μ_i is the aforementioned mean descriptor of a line, and d_j is the descriptor of its related point descriptor j. μ_i' and μ_{i_B}' are mean descriptors in another associated point cloud, and δ_{s} and δ_{d} are the positive and negative margins. $[x]_+ = \max(0, x)$, and $\|\cdot\|_1$ for the L1-distance. Finally, we use $\omega = 2$ to balance the final loss $\mathbf{L}$ as:

$$\mathbf{L} = \omega \cdot \mathbf{L}_{seg} + \mathbf{L}_{same} + \mathbf{L}_{diff} + \mathbf{L}_{match} + \mathbf{L}_{mismatch}. \tag{5}$$

Line-based Registration Our network outputs labels and descriptors for each point. We first extract lines using steps in Section 3.1. Then we perform descriptor matching to get line correspondences. The threshold of the matched descriptor is set to 0.1. The transformation $\mathbf{T}$ for registering the source cloud $\mathbb{S}$ to the target cloud $\mathbb{T}$ is optimized by minimizing point-to-line distances of all line matching cost $\xi_i, i \in \mathbf{N}$:

$$\xi_i = \sum_{j}^{\mathbf{N}_i} \frac{\left| \left(\mathbf{T} \cdot p_j^{\mathbb{S}} - p_{i_{e_0}}^{\mathbb{T}} \right) \times \left(\mathbf{T} \cdot p_j^{\mathbb{S}} - p_{i_{e_1}}^{\mathbb{T}} \right) \right|}{\left| p_{i_{e_0}}^{\mathbb{T}} - p_{i_{e_1}}^{\mathbb{T}} \right|} \tag{6}$$

where $p_j^{\mathbb{S}}$ is the line points in the source frame, $p_{i_{e_0}}^{\mathbb{T}}$ and $p_{i_{e_1}}^{\mathbb{T}}$ are endpoints of the matched line $\langle i_{e_0}, i_{e_1} \rangle$ of line i.

4 Experiments

4.1 Network Training

To begin with our generated synthetic data, we first train our line segmentation network using those synthetic point clouds with 50 epochs to converge. Then, to use the auto labeling method for generating sufficient and qualified real-world labeled scans, we obtain 12,989 LiDAR frames and iteratively train 100 epochs to refine these auto labeling results. Finally, we train our whole line segmentation and description network with 120 epochs to obtain the final applicable model for real-world scans.

We use scans including sequences 00-07 from the KITTI odometry dataset [14], with the last two sequences 06-07 for the validation set, and the rest 00-05 for the training set, to train our network. For each LiDAR frame, we voxelize the points cloud with $0.25m$ voxel size. We sample 20,000 points for evaluation and 15,000 points for training, since the kNN in *EdgeConv* is $O(N^2)$ space complexity and consumes large memory in the training process. We calculate point-to-line distances following Eq. 6 on the line segments in Sec. 3.1. The line pair whose mean distance is within $0.2m$ will be selected as a line correspondence to calculate descriptor loss. We implement our network in Tensorflow [1] with Adam [20] optimizer. The learning rate is set to 0.001 and decreases by 50% for every 15 epochs. The whole network is trained on 8 NVIDIA RTX 3090 GPUs.

4.2 Point Cloud Registration Test

Benchmarking. We use sequences 08-10 from the KITTI odometry dataset [14] to test the ability of our network on extracting line features and using them for point cloud registration. The preprocessing steps remain the same with our data preparation, and we choose to compare with traditional and learning-based methods for the global search registration. These traditional methods include ICP [6], RANSAC [13] and Fast Global Registration(FGR) [45], are all implemented by Open3D [46]. Specifically, The RANSAC and FGR use the FPFH [33] feature extracted from $0.25m$ voxel grid downsampled point clouds, and the max iteration is set to $4e^6$. Two learning-based methods include HRegNet [22] and Deep Global Registration (DGR) [9], and they use ground-truth pose to calculate loss and predict the transformation directly through the network. PointDSC [4] learns to prune outlier correspondences. D3Feat [5] and SpinNet [3] extract salient features from point clouds. Our line-based registration extracts 18 line segments with 350 points per frame on average. For fair comparisons, the number of keypoints in learning-feature-based methods is also set to 350, while other parameters remain unchanged.

Metrics. We use both the Relative Translation Error (RTE) and Relative Rotation Error (RRE) [22] to measure the registration accuracy. Additionally, as a special reference for evaluating the success rate for global search registration methods, we treat those calculated transformations with relative error w.r.t. the ground truth smaller than $2m$ and $5°$, as a successful attempt of registration.

Table 1. Registration performance on KITTI dataset. Our line segmentation and description method is highly competitive to the SOTA point-based approaches on the success rate, and both RTE and RRE can be refined with a subsequent coarse-to-fine ICP strategy.

	RTE (m)		RRE (deg)		Recall
	Mean	Std	Mean	Std	
ICP [6]	0.417	0.462	0.707	0.741	11.30%
FGR [45]	0.685	0.514	1.080	0.921	81.17%
RANSAC [13]	0.214	0.193	0.924	0.907	52.45%
HRegNet [22]	0.299	0.380	0.712	0.643	75.93%
DGR [9]	0.164	0.385	0.226	0.569	41.41%
PointDSC [4]	0.187	0.225	0.306	0.297	44.98%
SpinNet [3]	0.183	0.142	1.267	0.761	93.98%
D3Feat [5]	0.088	**0.043**	**0.343**	**0.242**	**98.90%**
SuperLine3D	**0.087**	0.104	0.591	0.444	97.68%

Results and Discussions. Table 1 shows the registration performances. Under random rotation perturbation, the recall of ICP is only 11.3%. The FGR and RANSAC methods based on FPFH features have higher recall but larger

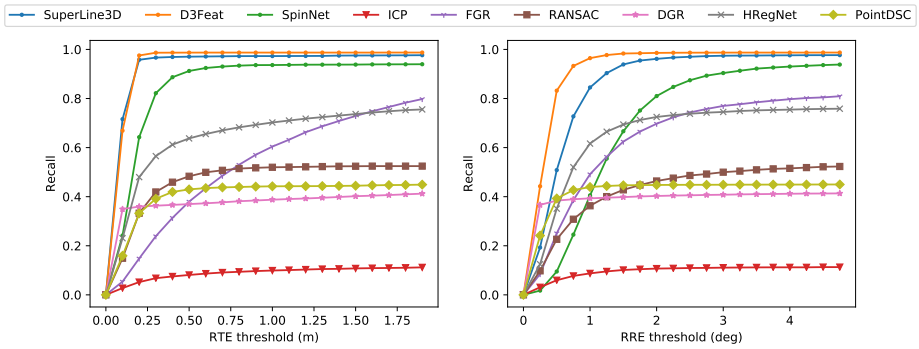


Fig. 5. Registration recall with different RRE and RTE thresholds on the KITTI dataset. The registration success rate of our line-based approach (blue) is close to the SOTA point-based approach D3Feat (orange) under different criteria.

errors. The learning-based end-to-end methods HRegNet and DGR also drop in recall and accuracy when dealing with large perturbed scenarios. PointDSC relies on the feature model, and the features do not have full rotation invariance, so its performance also deteriorates. Fig. 5 shows the registration recall with different error thresholds. SpinNet and D3Feat have better performances, with recall of over 90%. Our line-based registration achieves comparable performance to point features, with a similar mean translation error and 1.22% lower recall than D3Feat. Fig. 7 shows the visualization results on KITTI test sequence. Our method successfully registers point clouds under arbitrary rotation perturbations. We will give more results in supplementary materials.

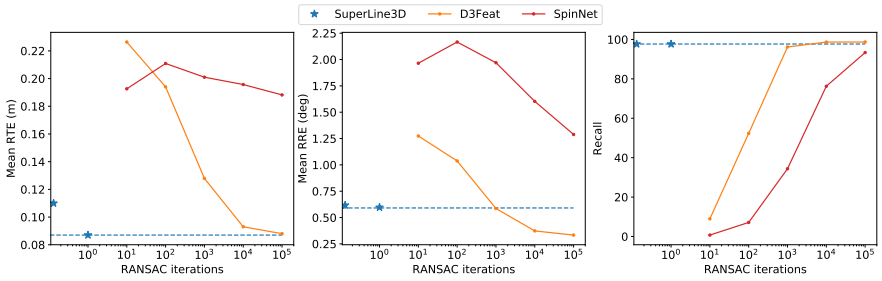


Fig. 6. Registration performance with different RANSAC iterations. There are many mismatches in point feature correspondences, which leads to unstable results when the number of iterations is small.

Ablation on RANSAC iterations. Point feature-based registration requires RANSAC to remove outliers and calculate the correct transformation. In the Table 1 and Fig. 5, the max iteration of RANSAC in the D3Feat and SpinNet

operations are set to $5e^4$. In contrast, our line-based registration does not rely on the RANSAC to filter erroneous matches: To perform outlier removal during transformation estimation, we calculate the line-to-line distances of line correspondences after the initial alignment, to remove the line correspondences with the mean distance greater than $1m$ and recalculate.

Fig. 6 shows the performance of point cloud registration under different RANSAC iterations. The x-coordinates in the figure are logarithmic coordinates. Our method does not use RANSAC for outlier rejection, and we use a dashed line in blue as a reference when comparing with other methods requiring RANSAC post processing. The star near the y coordinates represents the original result, and the star with an x-coordinate of 1 is the result after outlier removal. Both D3Feat and SpinNet can not get accurate transformation without RANSAC until the max iteration exceeds 1,000.

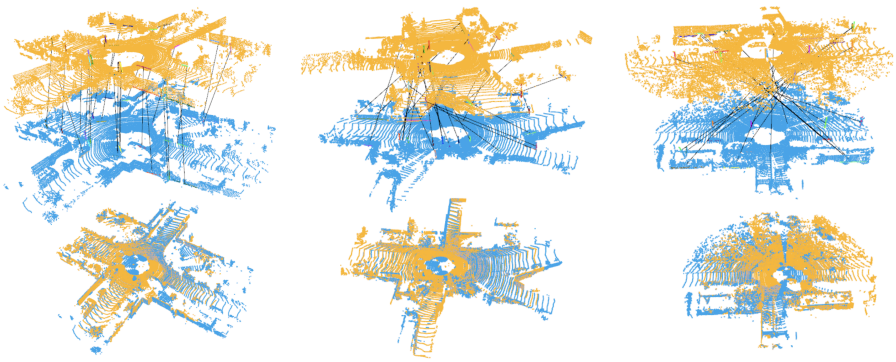


Fig. 7. Qualitative visualization on KITTI test sequence. Top: line associations between two LiDAR frames, Bottom: registration results of two frames.

4.3 Line Segmentation Evaluation

To evaluate the scale-invariance of our base segmentation model, we train PointNet [30], PointNet++ [31] and vanilla DGCNN [39] on the synthetic dataset. The training set includes 4,000 synthetic point clouds normalized within $[0, 1]$. We test the trained model with point clouds scaled from 0.1 to 3.0.

Fig. 8 shows the accuracy and mIOU of network predictions. Methods without scale adaptation suffer from performance decrease when the scale changes. The vanilla DGCNN gets best accuracy and mIOU in small scale disturbance (0.8 to 1.6), while our scale-invariant approach is stable under arbitrary scales. We can find that when the scale is determined, using the scale-invariant approach will decrease the accuracy, so we only use it in synthetic data training. In the joint training of segmentation and description, we utilize the vanilla DGCNN instead.

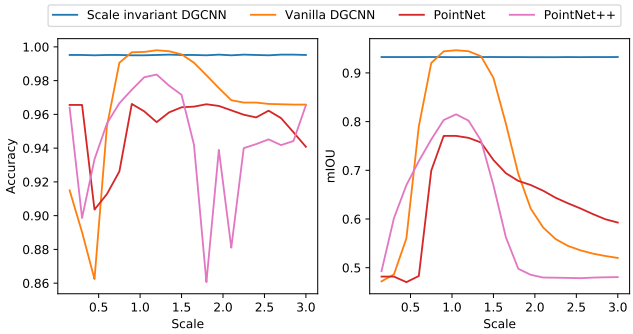


Fig. 8. Accuracy and mIOU of network predictions under different scale disturbances. Our scale-invariant approach is stable under arbitrary scales, but is a little worse than the vanilla DGCNN in the original scale.

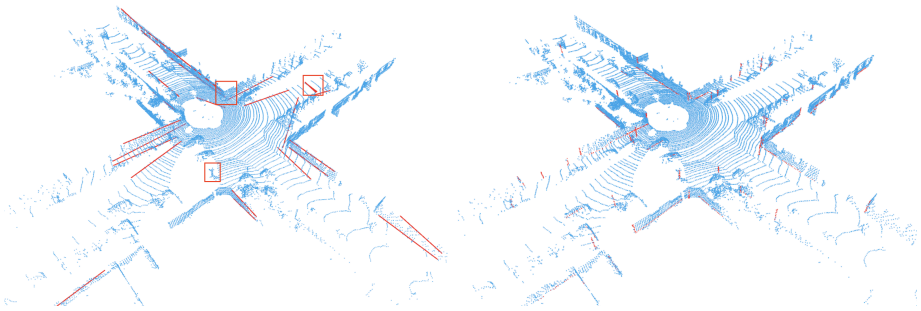


Fig. 9. Qualitative visualization of line segmentation between Lu et al. [24] (left) and ours (right). Our method segments most of the poles and building edges.

Fig. 9 shows the qualitative visualization of our line segmentation compared with the only open-source 3D line detection method [24] we found. Our method segments most of the lines, while the open-source one extracts LiDAR scan lines on the ground and cannot detect the poles.

4.4 Generalization on Unseen Dataset

To compare the generalization of learning feature-based models, we test our method against state-of-the-art point feature methods on the unseen Apollo Sourthbay dataset [23] using the models trained on the KITTI dataset. We uniformly choose half of the point clouds from the SanJoseDownTown sequence as the source frames, select target frames every 5 frames, and add random yaw-axis rotation perturbances on the source frames. We get 8,296 point cloud pairs for evaluation. The data preprocessing of the point cloud is the same as the KITTI dataset.

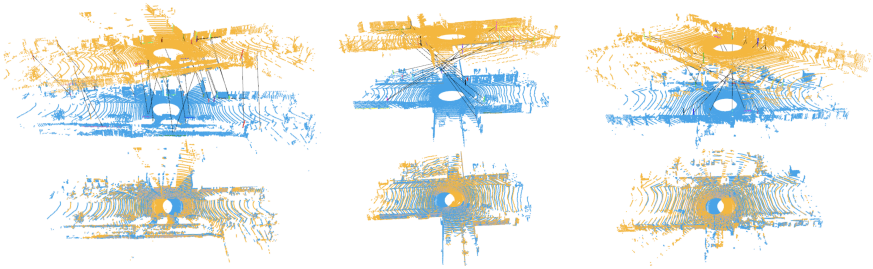


Fig. 10. Qualitative visualization on Apollo SourthBay Dataset, SanJoseDowntown sequence. The majority of the line correspondences are stable poles, which helps reduce the translation error by a large margin.

Table 2 shows the point cloud registration results. On unseen datasets, all methods show a drop in recall. D3feat has the best performance, while the mean translation error of our method is the smallest one. Fig 10 shows qualitative visualization on the test data. There are more poles in this sequence, which is beneficial to our line-based registration.

Table 2. Test on unseen Apollo SourthBay Dataset, SanJoseDowntown sequence.

	RTE (m)		RRE (deg)		Recall
	Mean	Std	Mean	Std	
SpinNet [3]	0.199	0.203	1.207	0.874	75.66%
D3Feat [5]	0.079	**0.046**	**0.206**	**0.144**	**95.94%**
SuperLine3D	**0.045**	0.107	0.262	0.402	93.84%

4.5 Ablation Study

Skip Encoding The receptive field is directly related to the number of encoded features in the EdgeConv module. When k is greater than 20, we can only set the batch size to 1 due to the enormous space complexity of EdgeConv. Its receptive field cannot be increased by increasing k. To this end, we utilize skip encoding. We gather $S \times k$ nearest neighbor features each time and select k features with stride size S for encoding. In this way, the receptive field increases S times without consuming too much memory (gathering $S \times k$ nearest-neighbor features will also increase a little memory usage). In the experiments, we test the cases with stride 1 (nearest neighbor encoding), 2, 4, and 6. As shown in the Table 3, adjusting the stride to 4 reaches the best performance, since the local features cannot be well encoded when the stride is too large.

Descriptor Dimension The descriptor dimension is one of the key factors for the feature matching performance, and the matching performance is poor when

the dimension is low. Our network extracts dense descriptors. Each point has a descriptor of d float numbers. It will take up a lot of storage space when its dimension is too large. Compared with the 16-dimension descriptor, the 32-dimension one has a more obvious improvement on the recall, while the 64-dimension descriptor has a small improvement. And increasing dimension to 128 only brings a smaller rotation error variance. Considering the average performances, we choose the 64-dimension implementation.

Table 3. Ablation study on stride and descriptor dimension.

		RTE (m)		RRE (m)		Recall
		Mean	Std	Mean	Std	
Stride	1	0.092	0.134	0.594	0.449	96.51%
	2	0.088	0.116	0.595	0.465	96.70%
	4	**0.087**	**0.104**	**0.591**	**0.444**	**97.68%**
	6	0.134	0.216	0.783	0.757	64.23%
Descriptor dimension	16	0.115	0.175	0.627	0.510	87.56%
	32	0.095	0.132	0.597	0.462	95.28%
	64	**0.087**	**0.104**	**0.591**	0.444	**97.68%**
	128	0.090	0.120	0.593	**0.441**	96.70%

5 Conclusions

This paper proposes the first learning-based 3D line feature segmentation and description method for LiDAR scans, which achieves highly-competitive performance to the point-feature-based methods in the point cloud registration. In the future, we will explore the usage of our deep learning line features on SLAM problems such as mapping, map compression, and relocalization. We will also optimize the network structure and reduce training resource consumption.

Acknowledgments This work is supported by the National Key R&D Program of China (Grant No: 2018AAA0101503) and Alibaba-Zhejiang University Joint Institute of Frontier Technologies.

References

1. Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al.: {TensorFlow}: A system for {Large-Scale} machine learning. In: 12th USENIX symposium on operating systems design and implementation (OSDI 16). pp. 265–283 (2016)
2. Akinlar, C., Topal, C.: Edlines: A real-time line segment detector with a false detection control. Pattern Recognition Letters **32**(13), 1633–1642 (2011)
3. Ao, S., Hu, Q., Yang, B., Markham, A., Guo, Y.: Spinnet: Learning a general surface descriptor for 3d point cloud registration. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11753–11762 (2021)
4. Bai, X., Luo, Z., Zhou, L., Chen, H., Li, L., Hu, Z., Fu, H., Tai, C.L.: Pointdsc: Robust point cloud registration using deep spatial consistency. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15859–15869 (2021)
5. Bai, X., Luo, Z., Zhou, L., Fu, H., Quan, L., Tai, C.L.: D3feat: Joint learning of dense detection and description of 3d local features. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6359–6367 (2020)
6. Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor fusion IV: control paradigms and data structures. vol. 1611, pp. 586–606. Spie (1992)
7. Biber, P., Straßer, W.: The normal distributions transform: A new approach to laser scan matching. In: Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453). vol. 3, pp. 2743–2748. IEEE (2003)
8. Chen, G., Liu, Y., Dong, J., Zhang, L., Liu, H., Zhang, B., Knoll, A.: Efficient and robust line-based registration algorithm for robot perception under large-scale structural scenes. In: 2021 6th IEEE International Conference on Advanced Robotics and Mechatronics (ICARM). pp. 54–62. IEEE (2021)
9. Choy, C., Dong, W., Koltun, V.: Deep global registration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2514–2523 (2020)
10. Choy, C., Park, J., Koltun, V.: Fully convolutional geometric features. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8958–8966 (2019)
11. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 224–236 (2018)
12. Fang, F., Ma, X., Dai, X.: A multi-sensor fusion slam approach for mobile robots. In: IEEE International Conference Mechatronics and Automation, 2005. vol. 4, pp. 1837–1841. IEEE (2005)
13. Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)
14. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition. pp. 3354–3361. IEEE (2012)
15. Gu, X., Wang, X., Guo, Y.: A review of research on point cloud registration methods. IOP Conference Series: Materials Science and Engineering **782**(2), 022070 (2020)

16. Huang, S., Qin, F., Xiong, P., Ding, N., He, Y., Liu, X.: Tp-lsd: Tri-points based line segment detector. In: European Conference on Computer Vision. pp. 770–785. Springer (2020)
17. Huang, X., Mei, G., Zhang, J.: Feature-metric registration: A fast semi-supervised approach for robust point cloud registration without correspondences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11366–11374 (2020)
18. Huang, X., Mei, G., Zhang, J., Abbas, R.: A comprehensive survey on point cloud registration. arXiv preprint arXiv:2103.02690 (2021)
19. Khan, M.U., Zaidi, S.A.A., Ishtiaq, A., Bukhari, S.U.R., Samer, S., Farman, A.: A comparative survey of lidar-slam and lidar based sensor technologies. In: 2021 Mohammad Ali Jinnah University International Conference on Computing (MAJICC). pp. 1–8. IEEE (2021)
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
21. Li, J., Lee, G.H.: Usip: Unsupervised stable interest point detection from 3d point clouds. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 361–370 (2019)
22. Lu, F., Chen, G., Liu, Y., Zhang, L., Qu, S., Liu, S., Gu, R.: Hregnet: A hierarchical network for large-scale outdoor lidar point cloud registration. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16014–16023 (2021)
23. Lu, W., Zhou, Y., Wan, G., Hou, S., Song, S.: L3-net: Towards learning based lidar localization for autonomous driving. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6389–6398 (2019)
24. Lu, X., Liu, Y., Li, K.: Fast 3d line segment detection from unorganized point cloud. arXiv preprint arXiv:1901.02532 (2019)
25. Pais, G.D., Ramalingam, S., Govindu, V.M., Nascimento, J.C., Chellappa, R., Miraldo, P.: 3dregnet: A deep neural network for 3d point registration. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 7193–7203 (2020)
26. Pautrat, R., Lin, J.T., Larsson, V., Oswald, M.R., Pollefeys, M.: Sold2: Self-supervised occlusion-aware line description and detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11368–11378 (2021)
27. Perez-Gonzalez, J., Luna-Madrigal, F., Piña-Ramirez, O.: Deep learning point cloud registration based on distance features. IEEE Latin America Transactions **17**(12), 2053–2060 (2019)
28. Pomerleau, F., Colas, F., Siegwart, R.: A review of point cloud registration algorithms for mobile robotics. Foundations and Trends in Robotics **4**(1), 1–104 (2015)
29. Pumarola, A., Vakhitov, A., Agudo, A., Sanfeliu, A., Moreno-Noguer, F.: Pl-slam: Real-time monocular visual slam with points and lines. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 4503–4508. IEEE (2017)
30. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017)
31. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)

32. Rozenberszki, D., Majdik, A.L.: Lol: Lidar-only odometry and localization in 3d point cloud maps. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 4379–4385. IEEE (2020)
33. Rusu, R.B., Blodow, N., Beetz, M.: Fast point feature histograms (fpfh) for 3d registration. In: 2009 IEEE international conference on robotics and automation. pp. 3212–3217. IEEE (2009)
34. Sarode, V., Li, X., Goforth, H., Aoki, Y., Srivatsan, R.A., Lucey, S., Choset, H.: Pcrnet: Point cloud registration network using pointnet encoding. arXiv preprint arXiv:1908.07906 (2019)
35. Shan, T., Englot, B.: Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4758–4765. IEEE (2018)
36. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6411–6420 (2019)
37. Von Gioi, R.G., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: a line segment detector. Image Processing On Line **2**, 35–55 (2012)
38. Wang, X., Liu, S., Shen, X., Shen, C., Jia, J.: Associatively segmenting instances and semantics in point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4096–4105 (2019)
39. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. Acm Transactions On Graphics (tog) **38**(5), 1–12 (2019)
40. Zhang, H., Luo, Y., Qin, F., He, Y., Liu, X.: Elsd: Efficient line segment detector and descriptor. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2969–2978 (2021)
41. Zhang, J., Singh, S.: Loam: Lidar odometry and mapping in real-time. In: Robotics: Science and Systems. Berkeley, CA (2014)
42. Zhang, L., Koch, R.: An efficient and robust line segment matching approach based on lbd descriptor and pairwise geometric consistency. Journal of Visual Communication and Image Representation **24**(7), 794–805 (2013)
43. Zhang, Z., Dai, Y., Sun, J.: Deep learning based point cloud registration: an overview. Virtual Reality & Intelligent Hardware **2**(3), 222–246 (2020)
44. Zhong, Y.: Intrinsic shape signatures: A shape descriptor for 3d object recognition. In: 2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops. pp. 689–696. IEEE (2009)
45. Zhou, Q.Y., Park, J., Koltun, V.: Fast global registration. In: European conference on computer vision. pp. 766–782. Springer (2016)
46. Zhou, Q.Y., Park, J., Koltun, V.: Open3d: A modern library for 3d data processing. arXiv preprint arXiv:1801.09847 (2018)