# Self-supervised Point Set Local Descriptors for Point Cloud Registration

Yijun Yuan<sup>1</sup>, Jiawei Hou<sup>1</sup>, Andreas Nüchter<sup>2</sup> and Sören Schwertfeger<sup>1</sup>

Abstract—In this work, we propose to learn local descriptors for point clouds in a self-supervised manner. In each iteration of the training, the input of the network is merely one unlabeled point cloud. On top of our previous work, that directly solves the transformation between two point sets in one step without correspondences, the proposed method is able to train from one point cloud, by supervising its self-rotation, that we randomly generate. The whole training requires no manual annotation. In several experiments we evaluate the performance of our method on various datasets and compare to other state of the art algorithms. The results show, that our self-supervised learned descriptor achieves equivalent or even better performance than the supervised learned model, while being easier to train and not requiring labeled data.

#### I. Introduction

Point cloud registration (PCR) is an essential task in various applications, such as 3D reconstruction and simultaneous localization and mapping (SLAM). Usually, the accuracy of the calculated transformation will dominate the performance of higher level tasks. Thus, researchers either make backend optimization on the high level task, such as SLAM [1] or work on the PCR side. In PCR, rigid-transformation is mostly considered. The Iterative Closest Point (ICP) algorithm, which iteratively solves the point correspondences and transformation, is the most famous algorithm in this field and has been widely used. Using correspondence criterion other than closeness promises improvements in point cloud registration, especially if no good initial guess is available. Descriptors on feature points are used for that. However, the correspondence computing with features requires a good distinctiveness of the descriptor, but the performance of different descriptors usually varies on various point sets. The popular hand-crafted detector ISS [2] and descriptor FPFH [3] are widely used.

Aside from the handcrafted descriptors, in recent years, with the fast development of image recognition, deep learning comes into the view. Both point-wise supervised models [4], [5], [6] and weakly supervised methods [7] are proposed to improve the matching performance. However, those supervising requires large amounts of labor to label the data. Those algorithms are either getting the correspondence from the matched point clouds [4], [5], [6], which is costly,

This work was supported by a German Academic Exchange Service (DAAD) scholarship granted to Yijun Yuan.

<sup>1</sup>The authors are with the School of Information Science and Technology, ShanghaiTech University, China. [yuanwj, houjw, soerensch]@shanghaitech.edu.cn

<sup>2</sup>The author is with the Department of Informatics VII – Robotics and Telematics, Julius-Maximilians-University Würzburg, Germany. [andreas.nuechter]@uni-wuerzburg.de

or they are labeling the inter-point cloud relation [7], which is is inefficient to train. In addition, the existing supervised models usually come with a triplet siamese structure or other loss functions, that are not directly related to the registration.

Therefore, we wonder if there is some method that does not require any labeling for training. Which means, that the supervised information comes from the same point cloud itself. Thus we call our method, self-supervised learning of point set local descriptors for point cloud registration.

Self-supervised learning is proposed for utilizing unlabeled data with the success of supervised learning. Producing a dataset with good labels is expensive, while unlabeled data is being generated all the time. The motivation of selfsupervised learning is to make use of the large amount of unlabeled data. The main idea of self-supervised learning is to generate the labels from unlabeled data, according to the structure or characteristics of the data itself, and then train on this unsupervised data in a supervised manner. Selfsupervised learning is wildly used in representation learning to make a model learn the latent features of the data. This technique is often employed in computer vision [8], [9], [10], [11], [12], video processing [13], [14] and robot control [15], [16], [17]. Similarly, we also find a work on self-supervised learning of 3D local features (MortonNet) [18]. Given a sequence of n points in Morton order, the MortonNet learns to predict the last point from the first n-1 points. However, the paper only discusses the application on segmentation.

In this paper, we propose a self-supervised learning model to learn the point cloud local descriptor for registration. The input of the network is merely one raw point cloud for each iteration of training. In our previous work we proposed the Full Connection Form Solution (CF) model [19] to solve the PCR problem non-iteratively in one-step without correspondences. There, handcrafted descriptors, such as FPFH [3], are used, which work well when point clouds are sampled from the same distribution.

The fully connected graph in CF considers the true corresponding point pairs especially important for the registration. So each descriptor should be similar to its possible corresponded points while different to others. Since the feeds are from one single point cloud, our self-supervise setting fits well to the CF module.

The idea of the network is that we, given a random rotation, feed the original point cloud PC1 and its rotated point cloud PC2 into the neural net to learn the feature, then solve the transformation with CF. Since the whole model is differentiable, with a loss function on the predicted transformation, we propagate the gradient back to the descriptor

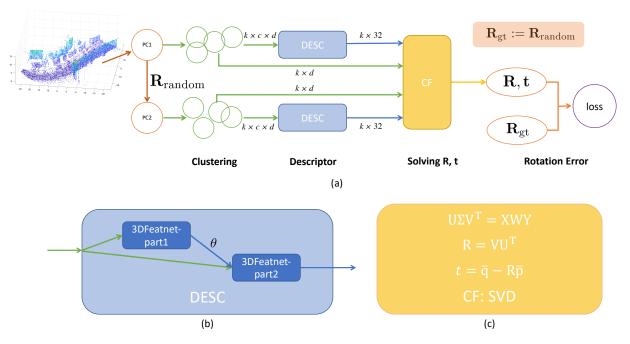


Fig. 1: (a) Pipeline of training: Single input point cloud; branching with random rotation; clustering; descriptor; CF algorithm to estimate R, t and rotation error as loss function. (b) Detail of of the descriptor. (c) SVD part of CF.

network. The whole pipeline diagram is shown in Fig. 1.

We build on the 3DFeatNet [7] as the Deep Neural Network body, as it solely feeds in points and does not require additional processing on the data representation.

Experiments on various datasets, i.e., [20], [21], demonstrate the performance of our descriptor.

To summarize the major contributions of this paper are:

- A new self-supervised learning method for descriptors in point clouds, that requires no manual annotation.
- Experiments that show that the self-supervised learned local descriptor has an equivalent performance as the supervised 3DFeatNet.

# II. RELATED WORK

This section reviews the technique advances in 3D local point cloud descriptors for registration. It describes hand-crafted descriptors and learned models. The handcrafted features usually resort on the geometric correlation inside a local cluster. Borrowing the strong representation ability of Deep Neural Networks, the learned models regress a descriptor function from data.

## A. Handcrafted 3D Descriptor

Point Feature Histogram (PFH) is known as the most typical 3D local descriptor [3]. It encodes the neighborhood geometrical properties with a multi-dimensional histogram [22]. For real time application, Fast Point Feature Histogram (FPFH) breaks the full interconnection of neighbors in PFH. Thus it achieves a linear time complexity and gradually becomes the most commonly used handcrafted 3D descriptor [3].

Apart from the descriptors from geometry, spin images (SI) [23] and unique shape context (USC) [24] split the spatial space into bins and count the number of points in each as a histogram.

## B. Learned 3D Local Descriptor

Focusing on the representation of point clouds fed into the deep neural net (DNN) model, we simply divide the learned descriptors into two classes: the voxel based and the point-set based methods.

Voxel based methods represent the local region around a keypoint as a volume. The most representative work is 3DMatch [4]. They use a 3D convolutional network to learn the weight from matching and non-matching. With the voxelized smoothed density value (SDV) representation, 3DSmoothNet [5] performs additional pre-processing, local reference frame (LRF), before DNN, to achieve rotation invariance.

The point set based methods start with the success of PointNet [25], which made it convenient and efficient to learn with DNN. PPFNet [6] and PPF-FoldNet [26] feed an encoding of points into PointNet to learn the features. However, they are not straight forward, because the encoding requires additional processes such as normal and point pair features. Then 3DFeatNet [7], by feeding in pure points set, models a mapping function from points to descriptors directly.

3DFeatNet uses whole point clouds instead of local patches as input, which is already different from most works. Thus it merely needs to annotate the point cloud relation instead of point cluster relation. This step saves lots of labor in labeling and tuning on triplet design. Such an

advantage resorts to its clustering step which is following PointNet++ [27] to separate point clouds into clusters. Then to learn the rotation invariance, it additionally uses a network sub-module to learn the orientations, to rotate clusters afterward. During training, an attention value is also extracted for loss function design. 3DFeatNet also has an extra usage. During training, attention is utilized to weight the pairwise distance between clusters of point clouds. However during inference, attention makes no use of the descriptor. So 3DFeatNet creatively uses the attention value to filter out non-interest points from the randomly sampled points, as the 3DFeatNet keypoints.

## III. METHODOLOGY

## A. Problem Statement

Given two point clouds P and Q, the goal of rigid registration is to solve

$$\min_{\mathbf{R}, \mathbf{t}} \sum_{(i,j) \in \mathcal{C}} ||\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_j||^2$$
 (1)

where  $\mathbf{p}_i \in \mathbf{P}|_{i \in 1,...,N}$ ,  $\mathbf{q}_j \in \mathbf{Q}|_{j \in 1,...,M}$  and  $\mathcal{C}$  is the set of correspondences.

A descriptor for each keypoint  $p_i$  is a vector utilized to compute the correspondences with points in the other point cloud. We denote the mapping from point location to descriptor vector as  $f_{\mathcal{X}}(\mathbf{x})$  with point  $\mathbf{x} \in \mathcal{X}$ . So the goal for this paper is to regress such a mapping f.

## B. Network Architecture

We demonstrate the pipeline of the training process in Fig. 1. The DESC module in between is the f we want to extract.

The whole training process consists of four parts. The first two modules, clustering and descriptor, follow 3DFeat-Net [7]. Next, the CF module [19] solves the transformation of sampled points with descriptors from the two point clouds and the Rotation Matrix Distance Module computes the error between the solved R and  $R_{\rm gt}$ , which is then the loss.

1) Network Body: For each of the point clouds, we sample the points from the cloud and then group points surrounding the sampled points into clusters. Following the example of [7], we are using grouping and sampling layers from PointNet++ [27] in this paper. With k sampled points as the centers, k clusters are extracted. For that, first all points within radius  $r_{\rm cluster}=2m$  around the sampled point are selected. Then c points are randomly sampled from that set. In the cluster, each point is of dimension d, that can be 3 (xyz), 6 (xyzrgb) or others. In this paper we use d=3, so we are only using the xyz location of the point.

Then the points of those clusters are passed into the descriptor network, which processes each cluster with size  $c \times 3$  and outputs their descriptor. The descriptor body consists of two parts, 3DFeatnet-part1 and 3DFeatnet-part2. The 3DFeatnet-part1 learns to generate the orientation to rotate the cluster input of 3DFeatnet-part2.

3DFeatNet only predicts a 1D rotation to avoid unnecessary equivariances. Thus they only rotate around the gravity

axis. Different from 3DFeatNet [7], that uses a Siamese Network with three branches for triplet input (anchor, positive, negative), we only use two branches with feeding PC1, PC2, as shown in Fig. 1. The two branches of network share weights.

2) Solving  $\mathbf{R}$ ,  $\mathbf{t}$ : The CF module, as discussed in [19], solves the registration for point sets with the same distribution in one step. Since PC2 is with a rotation  $R_{\text{random}}$  from PC1 that fits for such a requirement, the transformation is solved

Please note that the location input of CF are the subsampled PC1 and PC2 from clustering in Fig. 1.

Given points  $\mathbf{p} \in g(\mathbf{P})$ ,  $\mathbf{q} \in g(\mathbf{Q})$  and their descriptors  $f_{\mathbf{P}}(\mathbf{p})$  and  $f_{\mathbf{Q}}(\mathbf{q})$ , the optimization program in the CF module is

$$\min_{\mathbf{R}, \mathbf{t}} \sum_{i=1}^{k_{\mathbf{P}}} \sum_{i=1}^{k_{\mathbf{Q}}} w_{i,j} ||\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_j||^2$$
 (2)

where

$$w_{i,j} = e^{-\frac{1}{\alpha}||f_{\mathbf{P}}(\mathbf{p}_i) - f_{\mathbf{Q}}(\mathbf{q}_j)||^2}.$$
 (3)

Let  $\mathcal{X} = \{\mathbf{p}'_1, \dots, \mathbf{p}'_{k_{\mathbf{P}}k_{\mathbf{Q}}}\}$ ,  $\mathcal{Y} = \{\mathbf{q}'_1, \dots, \mathbf{q}'_{k_{\mathbf{P}}k_{\mathbf{Q}}}\}$ .  $\mathbf{p}'$ ,  $\mathbf{q}'$  where the same index indicates the point of  $\mathbf{p}$ ,  $\mathbf{q}$  in one term of the summation in Eq. (2). w is correspondingly set.

Then above problem Eq. (2) is transformed into

$$\min_{\mathbf{R}, \mathbf{t}} \sum_{i=1}^{k_{\mathbf{P}} k_{\mathbf{Q}}} w_i ||(\mathbf{R} \mathbf{p}_i' + \mathbf{t}) - \mathbf{q}_i'||^2$$
 (4)

The transformation  $\mathbf{R}$ ,  $\mathbf{t}$  in Eq. (2) has a closed form solution using the SVD [28], cf. Fig. 1 (c).

3) Loss Function: The loss function is to supervise the one-step solved rotation [19]. Given the ground truth transformation  $\mathbf{R}_{gt}$ ,  $\mathbf{t}_{gt}$ , the loss function is the deviation from the identity matrix [29] as follows

$$loss = ||\mathbf{I} - \mathbf{R}\mathbf{R}_{gt}^T||_F. \tag{5}$$

4) Self-supervised Learning: With the above four parts of network components, it merely requires to feed in one raw point cloud to learn for each iteration. Given a random rotation, we want to minimize its distance from the solved rotation.

Since the whole pipeline is differentiable, the parameters in the descriptor network is updated with gradient backpropagation.

We call our model self-supervised learning model, because we generate labels ( $\mathbf{R}_{random}$ ) from nothing, and train the unlabeled data in a supervised way. The model is learned from a raw point cloud itself.

# IV. EXPERIMENTS AND RESULTS

Our implementation and experiments are based on top of the open source release<sup>1</sup> of 3DFeatNet [7]. The Oxford RobotCar dataset [20] is used for network training and testing, following the settings of [7]. Additionally, the KITTI dataset is also used for testing the model.

<sup>1</sup>https://github.com/yewzijian/3DFeatNet

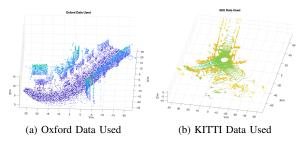


Fig. 2: The processed data utilized in this paper. In Oxford data, the span on X,Y axis is around 50m. For Kitti data, span on one axis may exceed 100m.

## A. Datasets

1) Oxford RobotCar Dataset: The Oxford dataset contains repeat traverses through the Oxford city center from May 2014 to December 2015, that were collected with the Oxford RobotCar platform. We use the pre-processed data from 3DFeatNet [7], that has 35 trajectories for training and another 5 trajectories for testing. The points scanned from 2D LIDAR are accumulated into 3D point clouds, according to the GPS/INS poses. Those poses were refined with ICP. 3DFeatNet then downsampled the training point clouds to about  $50,000 \pm 20,000$  points and the test point clouds to exactly 16,384 points. During training, the training point clouds are further randomly subsampled to 4096 points. This way we obtain 21,875 training and 828 testing point cloud sets.

2) KITTI Dataset: We test our model on the 11 training sequences from KITTI dataset [21] as [7]. The parts of KITTI dataset used in the experiments include Velodyne laser point clouds, GPS/INS as ground truth poses and the calibration files. The point clouds are also downsampled with a grid size of 0.2 m and obtain 2,369 point clouds in the end.

## B. Setting

On training, 3DFeatNet feeds the network a triplet of 3 point clouds (anchor, positive, negative) from the Oxford Dataset. For our training of the model, we simply use the one point cloud (anchor). The clustering and network settings are following 3DFeatNet. The radius of the cluster is  $r_{\rm cluster} = 2m$ 

Since our work only consists of the descriptor, we use the keypoints from other detectors. The detectors used in this evaluation are ISS [2] and 3DFeatNet keypoints (kpt). The experiment consists of descriptor matching joint performance with keypoint and geometric registration. The other baseline descriptors are FPFH [3], SI [23], USC [24], CGF [30], 3DMatch [4] and 3DFeatNet [7]. In the CF module, we set  $\alpha=1$ .

The 3DFeatNet takes 2 epochs to pretrain 3DFeatNet-part2 and trains whole model 70 epochs with lr=1e-5. We use the open released sample Tensorflow [31] checkpoint to achieve the network weight of 3DFeatNet [7].

During the training of our model, we follow the 3DFeatNet to set the network hyperparameters, such as batch size 6,

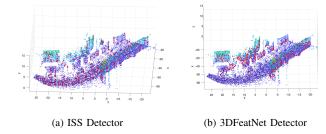


Fig. 3: Keypoint Demo on the same point cloud of Fig. 2a. Key-points are plotted with red dots on point cloud.

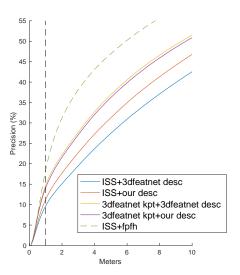


Fig. 4: Precision plot for distance between nearest neighbor point and the ground truth location.

Adam optimizer and 32 dimension descriptor. 3DFeatNet claims that it is hard to train, while our network is easy to train: Without any pre-training, our model is randomly initialized and saved at iteration 72,500 (same as 3DFeatNet's 20 epochs training) with learning rate lr=1e-3. The provided  $R_{\rm random}$  is generated to only have the Z-axis rotate  $\phi \sim \mathcal{N}(0,\sigma_r^2)$ . In the experiment, we set  $\sigma_r=0.6$ . In addition, we apply a 3D jitter with  $\Delta p \sim \mathcal{N}(\mathbf{0},\sigma_p\mathbf{I})$  ( $\sigma_p=0.01$ ) for each point in PC1 and PC2.

On inferencing, the setting of the 3DFeatNet detector such as,  $\beta_{attention}$  and  $r_{nms}$ , follows [7].

## C. Evaluations

To evaluate the performance of our descriptor, we use ISS and 3DFeatNet detector to provide the keypoint, and demonstrate the performance on precision and geometric registration.

An example of keypoint detection of ISS and 3DFeatNet detector is shown in Fig. 3. We observe that the interest points of the ISS are distributed in the whole point cloud while most keypoints of 3DFeatNet detector are on the wall.

1) Precision Test: Using exhaustive search, this test searches for the nearest descriptor neighbor in the paired

TABLE I: Registration Error on the Oxford Dataset. The first 8 rows are collected from [7]. We obtained the last 5 rows using ISS keypoints, FPFH, 3DFeatNet and our approach. The parameters are the same as in [7].

	RTE	RRE	Success Rate	Avg #Iter
ISS+FPFH [3]	0.396	1.60	92.32%	7171
ISS+SI	0.415	1.61	87.45%	9888
ISS+USC	0.324	1.22	94.02%	7084
ISS+CGF	0.431	1.62	87.36%	9628
ISS+3DMatch	0.494	1.78	69.06%	9131
ISS+PN++	0.511	1.88	48.86%	9904
ISS+3DFeatNet Desc	0.314	1.08	97.66%	7127
3DFeatNet Kpt+3DFeatNet Desc	0.300	1.07	98.10%	2940
ISS+FPFH [3]	0.354	1.43	93.37%	3239
ISS+3DFeatNet Desc [7]	0.314	1.08	97.66%	7126
ISS+Our Desc	0.311	1.01	98.10%	5648
3DFeatNet Kpt+3DFeatNet Desc	0.304	1.08	97.66%	3294
3DFeatNet Kpt+Our Desc	0.310	1.08	97.05%	3650

TABLE II: Registration Error on the KITTI Dataset. The first 6 rows are collected from [7]. We obtained the last 4 rows using ISS keypoints, FPFH and our approach.

	RTE	RRE	Success Rate	Avg #Iter
ISS+FPFH [3]	0.325	1.08	58.59%	7462
ISS+SI	0.358	1.17	55.92%	9219
ISS+USC	0.262	0.83	78.24%	7873
ISS+CGF	0.233	0.69	87.81%	7442
ISS+3DMatch	0.283	0.79	89.12%	7292
3DFeatNet Kpt+3DFeatNet Desc	0.258	0.57	95.97%	3798
ISS+3DFeatNet Desc	0.246	0.627	93.50%	8311
ISS+Our Desc	0.215	0.510	93.50%	5960
3DFeatNet Kpt+3DFeatNet Desc	0.264	0.599	95.58%	4394
3DFeatNet Kpt+Our Desc	0.258	0.570	95.44%	3732

model for each keypoint. Then the Euclidean distance between the neighbor and ground truth location is computed. We show the plot in Fig. 4. The x-axis is a threshold to consider a pair as correct and the y-axis is the correct proportion. Comparing to the plot in [7], the tested ISS+FPFH achieved a better result than the plot in [7].

For both 3DfeatNet Descriptor and our descriptor, the test with 3DFeatNet kpt works better than ISS kpt. Our proposed unsupervised model achieved a similar result to 3DFeatNet Desc with 3DfeatNet kpt and a better result on ISS kpt, comparing with 3DFeatNet Desc. We use the x=1m line as a cut. Both 3DFeatNet Desc and our model achieves around 15% precision, which is close to the best score in the record of [7].

2) Geometric Registration: With ISS keypoint and 3DFeatNet keypoint, we evaluate the descriptor algorithm on geometric registration. The registration is with the nearest neighbor match and applying RANSAC for transformation estimation. The RANSAC iteration is limited to 10,000 and adjusted with 99% confidence. Then Relative Rotation Error (RRE), Related Translation Error (RTE) are computed with ground truth to evaluate the accuracy of registration. A success is decided when RTE<2 m, RRE<5 deg.. The speed of converging is reflected by average number of iterations. Since we use the same datasets (Oxford and KITTI) as [7], we compare to the results from their table.

The evaluation on Oxford data is demonstrated in Table I. The first eight rows are fetched from [7] and the last five rows are from our own experiments.

We observe that, firstly, except for PN++, the handcrafted descriptors cannot exceed the learned descriptors. Secondly, our unsupervised learned descriptor achieves the best result on RRE and the success rate with ISS, while it has similar RTE and average iterators as 3DFeatNet on 3DFeatNet keypoint. Thirdly, the 3DFeatNet keypoint provide better accuracy for our descriptor. Regarding the success rate and convergence speed, when 3DFeatNet keypoint is used, they (3DFeatNet Desc and our Desc) are close to our approach. However, when ISS keypoint is utilized, our descriptor works better.

An example of a registration is shown in Fig. 5. We observe that ISS keypoints are distributed over whole point clouds while 3DFeatNet keypoints are mainly on the wall. For both keypoints, our descriptor matches accurately.

Then the algorithms are tested on other outdoor data, the KITTI dataset. The registration results are shown in Table II. The first six rows of the results are fetched from [7] and the last four rows are from our experiments.

From the table, we observe that, firstly, ISS+our descriptor achieves best accuracy. And secondly, with 3DFeatNet keypoint, our model achieves as good results as the 3DFeatNet descriptor. A further example of a registration is in shown in Fig. 6. For both ISS and 3DFeatNet keypoints, our descriptor performs well.

## D. Exploration

As in [7], we also perform a descriptor matching test that doesn't use a keypoint detector.

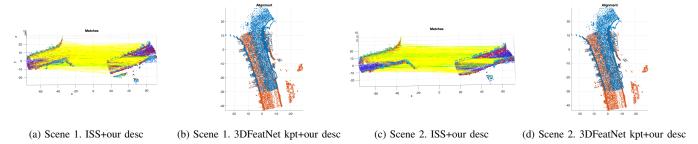


Fig. 5: Oxford data geometric registration. Top view. Yellow dot line shows the matched pairs.

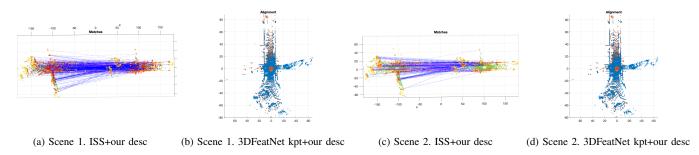


Fig. 6: KITTI data geometric registration. Top view. Blue dot line shows the matched pairs.

1) Random Point Descriptor Matching: The data of descriptor matching is extracted from Oxford Robot Car Dataset at randomly selected locations. It consists of 30,000 pairs of clusters with 4.0m radius. Half of them are matched pairs that are with matched frames and the other half are from point clouds at least 20m away. [7] uses a false-positive rate of 95% recall.

Please note that the clusters extracted are at a random location with the purpose of isolating the effect of keypoint detector. The keypoint locates in center of each cluster.

The result for our model is 66.33%, while the errors for SI, FPFH, USC, 3DMatch, PN++ and 3DFeatNet from [7] are 68.51%, 54.13%, 91.59%, 59.77%, 38.49%, 50.57% and 36.84%, respectively. Other than having equivalent performance as 3DFeatNet in the previous joint performance and geometric registration, our model only has 66.33% recall.

So we consider that our learned model is not distinctive on random points as on keypoints. Thus we make further experiments on the given descriptor matching data to find if it performs on non-random keypoints.

We use ISS to filter out the non-interested clusters pairs. For those 60,000 clusters, we use the ISS keypoint detector with the same parameters in previous evaluation to detect keypoints in each cluster. Then we set a radius  $r_{th}$  to threshold if there exists a keypoint that is in the sphere, which has radius  $r_{th}$  on the center of cluster. If both clusters in a pair have keypoints in the sphere, this pair is kept for test.

The results are demonstrated in Table III.

We find that a large amount of clusters are not good keypoints. With small  $r_{th}=0.1m,\,0.2m,$  our model works better. However, when keypoints are farther to center, with

TABLE III: Descriptor matching Error (%). Lower is better.

$r_{tj}(m)$	0.1	0.2	0.3	0.4
pairs(posi,neg)	58(34,24)	178(103,75)	722(407,315)	1601(850,751)
3DFeatNet	37.5	28	31.43	35.955
Our	29.17	13.33	53.65	56.99

 $r_{th}$  larger, our model again does not works well.

Therefore, the use of our learned descriptor should be embedded with a key-point detector, but not with random points.

## V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a self-supervised learning model to learn a point set local descriptor. Borrowing the high efficiency of the one-step registration model CF, a transformation is predicted with a learned descriptor function. Thus, with a CF module as the last layer, only one point cloud is needed to be fed in the network for training. Using the same network body as 3DFeatNet, our model is much easier to train, because this self-supervised method does not require any manual labor on annotation and, without any pretraining, can converge with a higher learning rate, requiring far fewer iterations. Our experimental evaluation showed that our descriptor achieves equivalent performance on precision and geometric registration as the 3DFeatNet Descriptor.

Needless to say, a lot of work remains to be done. As future work we want to more deeply explore the parameter space of the descriptor model to improve the descriptors of random points. Furthermore, we want to train and test our method with indoor LIDAR and RGBD datasets, that could then also use RGB information in the descriptor.

#### REFERENCES

- [1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.
- [2] Y. Zhong, "A shape descriptor for 3d object recognition," in *Proceedings ICCV 2009 Workshop 3DRR*, vol. 6, 2009.
- [3] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in 2009 IEEE international conference on robotics and automation. IEEE, 2009, pp. 3212–3217.
- [4] A. Zeng, S. Song, M. Nießner, M. Fisher, J. Xiao, and T. Funkhouser, "3dmatch: Learning local geometric descriptors from rgb-d reconstructions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1802–1811.
- [5] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, "The perfect match: 3d point cloud matching with smoothed densities," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5545–5554.
- [6] H. Deng, T. Birdal, and S. Ilic, "Ppfnet: Global context aware local features for robust 3d point matching," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 195–205.
- [7] Z. J. Yew and G. H. Lee, "3dfeat-net: Weakly supervised local 3d features for point cloud registration," in *European Conference on Computer Vision*. Springer, 2018, pp. 630–646.
- [8] A. Dosovitskiy, P. Fischer, J. T. Springenberg, M. Riedmiller, and T. Brox, "Discriminative unsupervised feature learning with exemplar convolutional neural networks," *IEEE transactions on pattern analysis* and machine intelligence, vol. 38, no. 9, pp. 1734–1747, 2015.
- and machine intelligence, vol. 38, no. 9, pp. 1734–1747, 2015.
  [9] S. Gidaris, P. Singh, and N. Komodakis, "Unsupervised representation learning by predicting image rotations," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=S1v4N210-
- [10] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1422–1430.
- [11] R. Zhang, P. Isola, and A. A. Efros, "Colorful image colorization," in *European conference on computer vision*. Springer, 2016, pp. 649–666
- [12] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," arXiv preprint arXiv:1605.09782, 2016.
- [13] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2794–2802.
- [14] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy, "Tracking emerges by colorizing videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 391–408.
- [15] E. Jang, C. Devin, V. Vanhoucke, and S. Levine, "Grasp2vec: Learning object representations from self-supervised grasping," *Conference on Robot Learning*, 2018.

- [16] X. Zhi, X. He, and S. Schwertfeger, "Learning autonomous exploration and mapping with semantic vision," in *International Conference on Image, Video and Signal Processing. IVSP*, ACM. ACM, 02/2019 2019. [Online]. Available: https://arxiv.org/abs/1901.04782
- [17] A. Nair, S. Bahl, A. Khazatsky, V. Pong, G. Berseth, and S. Levine, "Contextual imagined goals for self-supervised robotic learning," *Conference on Robot Learning*, 2019.
- [18] A. Thabet, H. Alwassel, and B. Ghanem, "Mortonnet: Self-supervised learning of local features in 3d point clouds," arXiv preprint arXiv:1904.00230, 2019.
- [19] Y. Yuan, D. Borrmann, A. Nüchter, and S. Schwertfeger, "Non-iterative one-step solution for point set registration problem on pose estimation without correspondence," arXiv preprint arXiv:2003.00457, 2020.
- [20] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.
- [21] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012, pp. 3354– 3361
- [22] R. B. Rusu, Z. C. Marton, N. Blodow, and M. Beetz, "Learning informative point classes for the acquisition of object model maps," in 2008 10th International Conference on Control, Automation, Robotics and Vision. IEEE, 2008, pp. 643–650.
- [23] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Transactions on pattern* analysis and machine intelligence, vol. 21, no. 5, pp. 433–449, 1999.
- [24] F. Tombari, S. Salti, and L. Di Stefano, "Unique shape context for 3d data description," in *Proceedings of the ACM workshop on 3D object retrieval*, 2010, pp. 57–62.
- [25] C. R. Qi, H. Su, K. Mo, and L. 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, 2017, pp. 652–660.
- [26] H. Deng, T. Birdal, and S. Ilic, "Ppf-foldnet: Unsupervised learning of rotation invariant 3d local descriptors," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 602–618.
- [27] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances* in neural information processing systems, 2017, pp. 5099–5108.
- [28] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, no. 5, pp. 698–700, 1987.
- [29] P. M. Larochelle, A. P. Murray, and J. Angeles, "A distance metric for finite sets of rigid-body displacements via the polar decomposition," 2006.
- [30] M. Khoury, Q.-Y. Zhou, and V. Koltun, "Learning compact geometric features," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 153–161.
- [31] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "Tensorflow: A system for large-scale machine learning," in 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16), 2016, pp. 265–283.