Abstract—Existing learning-based point feature descriptors are usually task-agnostic, which pursue describing the individual 3D point clouds as accurate as possible. However, the matching task aims at describing the corresponding points consistently across different 3D point clouds. Therefore these too accurate features may play a counterproductive role due to the inconsistent point feature representations of correspondences caused by the unpredictable noise, partiality, deformation, etc., in the local geometry. In this paper, we propose to learn a robust task-specific feature descriptor to consistently describe the correct point correspondence under interference. Born with an Encoder and a Dynamic Fusion module, our method EDFNet develops from two aspects. First, we augment the matchability of correspondences by utilizing their repetitive local structure. To this end, a special encoder is designed to exploits two input point clouds jointly for each point descriptor. It not only captures the local geometry of each point in the current point cloud by convolution, but also exploits the repetitive structure from paired point cloud by Transformer. Second, we propose a dynamical fusion module to jointly use different scale features. There is an inevitable struggle between robustness and discriminativeness of the single scale feature. Specifically, the small scale feature is robust since little interference exists in this small receptive field. But it is not sufficiently discriminative as there are many repetitive local structures within a point cloud. Thus the resultant descriptors will lead to many incorrect matches. In contrast, the large scale feature is more discriminative by integrating more neighborhood information. But it is easier to be disturbed since there is much more interference in the large receptive field. Compared with the conventional fusion strategy that handles multiple scale features equally, we analyze the consistency of them to judge the clean ones and perform larger aggregation weights on them during fusion. Then, a robust and discriminative feature descriptor is achieved by focusing on multiple clean scale features. Extensive evaluations validate that EDFNet learns a task-specific descriptor, which achieves state-of-the-art or comparable performance for robust matching of 3D point clouds.

Index Terms—Point cloud, task-specific descriptor, convolution and transformer encoder, dynamic fusion module.

I. INTRODUCTION

WITH the vigorous development of 3D measurement technology, point cloud has received widespread research attention [5]–[7]. Point matching plays fundamental roles in various applications, e.g. 3D reconstruction [8], [9], object pose estimation [10], autonomous driving [11], [12], etc. To achieve accurate matching, using the point feature descriptors to associate the source and target point clouds has been a widely adopted strategy. Many point feature descriptors have been proposed recently.

Before the era of deep learning, handcrafted point feature descriptors are always custom-tailored by investigating the neighborhood distribution of each 3D point (e.g. [13]–[16]). Recently, with the flourishing of deep learning, the exploration of feature descriptors has shifted to learning-based approaches, and superior performance has been achieved. Many methods use multi-layer perceptron (MLP) to handle the irregular point clouds to learn point feature descriptors [2], [17], [18]. Furthermore, convolution is applied [3], [19] to incorporate local information more effectively. Besides, customized rotation-equivariant/-invariant descriptors [20]–[25] have been developed to handle the pose change. However, in these existing methods, the obtained local features are always task-agnostic, where feature extraction and matching are totally separated. They are committed to describing each 3D point as accurate as possible. Note that the matching task resorts to describing the corresponding points consistently. Then, these accurate feature descriptors may backfire since the local geometry is usually disturbed by noise, partiality, deformation, etc. Then, the descriptors of corresponding points are thus inconsistent.

To resolve this problem, we propose a method termed EDFNet (Encoder-Dynamic Fusion structure) to learn a task-specific descriptor in this paper. Our descriptor is devoted to consistently describing the correspondence under irregular and unpredictable interference. EDFNet is developed from the following two aspects.

First, since the matching task solves for the point correspondences from the source and target point clouds, we advocate facilitating the matchability of correspondence based on their repetitive local structure. To this end, a convolution and Transformer encoder (notated as CT-encoder) is designed. It not only learns the local geometry of each point in the current point cloud by convolution but also exploits the repetitive structure in the paired point cloud by using the Transformer [26]–[28]. Different from [29]–[31] that use
Transformer to sense the paired point cloud once, our CT-encoder strengthens the matchability of correspondence more sufficiently. Specifically, we apply multiple encoder layers to integrate the local and paired geometries from multiple scales.

Second, we observe that there exists an inevitable struggle between robustness and discriminativeness of single scale feature. Small scale features are not sufficiently discriminative because their small receptive fields cannot capture enough local geometry information. Then, many repetitive minor structures in point clouds will cause incorrect matches. Nevertheless, this small scale feature is robust, which describes correspondence consistently due to little interference in such small receptive field. By contrast, the large receptive field improves the discriminativeness of large scale feature by encoding more neighborhood information. However, it is easy to be affected by embedding significantly increased disturbers as shown in Fig. 1. To achieve better matching, we advocate learning a robust and discriminative descriptor. To this end, we propose to jointly use different scale features. However, we notice that the existing multiple scale feature fusion methods cannot handle the disturbers. These methods usually take addition [3], [4] or concatenation [1], [2], [32] strategies, which essentially treat different scale features equally. Thus the contaminated scale features also contribute to the final fused descriptor. Inspired by the fusion on 2D image investigation [33], [34], we propose a dynamic fusion module. Specifically, we observe an intuitive rule, i.e., the clean scale features are usually consistent to describe the similar local geometries. And the contaminated ones are usually diverse due to the irregular disturbers. Thus, we propose to guide the obtained descriptor to approach to the consistent and clean scale features and keep away from the contaminated ones. To this end, we analyze the distribution of them and focus on the clean ones more during fusion. Through our dynamic fusion, the final obtained feature descriptor is robust and discriminative via utilizing multiple clean scale features.

With the above two strategies, our EDFNet achieves a robust matching task-specific point feature descriptor. Extensive evaluations on several benchmark datasets have validated our method, which achieves a state-of-the-art (SOTA) or comparable matching performance. Besides, the strong robustness and generalization ability of EDFNet have also been verified on rigid and non-rigid point cloud data.

Our contributions can be summarized as follows:

- We propose EDFNet to learn matching task-specific feature descriptors to consistently describe the correspondences under unpredictable interference.
- We propose an encoder to extract multiple scale features by sensing both two input point clouds simultaneously, which consists of a convolution branch and a Transformer branch. Besides, we design a dynamic fusion module to guide the final fused feature to approach to the consistent and clean scale features and far away from the contaminated ones to reply to disturbers. This dynamic fusion module achieves a good balance between robustness and discriminativeness.
- Extensive experiments on several benchmark datasets show that our EDFNet achieves comparable if not SOTA performance for robust 3D point cloud matching.

II. RELATED WORK

In this section, we briefly review the representative feature descriptors of 3D point clouds, including the handcrafted ones and the learning-based ones.

A. Handcrafted Point Feature Descriptors

To match 3D points in the source and target point clouds, many customized pose-invariant feature descriptors have been designed following two representative strategies, i.e., local reference frame (LRF)-based approaches and LRF-free approaches. For the LRF-based approaches, the unique LRF is explored to transform the local neighborhood of 3D point to a canonical representation [16], [35]–[37]. On the other hand, the LRF-free approaches are another hot spot, and many methods have emerged, e.g., [15], [38], [39]. These LRF-free methods usually focus on the intrinsically invariant geometry peculiarities. Although significant progress has been achieved, the performance of these handcrafted descriptors based methods is limited. Furthermore, most existing handcrafted features cannot handle the noise, occlusions, deformation, etc., [40].

B. Learning-Based Point Feature Descriptors

Deep learning technology has inspired various 3D point cloud descriptor learning methods. The early works apply deep learning after converting the raw irregular 3D point cloud data into regular data, such as 2D image [41], [42], 3D voxel [19], [43]–[46], etc. Recently, using deep learning to directly handle 3D unstructured point cloud data has become popular. To this end, PointNet [17] introduces a symmetric function, max-pooling for permutation invariance. Then, to achieve a more
To achieve a robust and discriminative descriptor, EDFNet explores the consistency of these different scale features themselves and guides the final descriptor to approach to consistent and clean ones and dodge the contaminated ones.

A. CT-Encoder

In this section, we use an encoder network to extract different scale features. Here, we exploit the repetitive structure in the input two point clouds to facilitate the matchability of correspondences. We learn the point features by capturing the local geometry information of the current point cloud and the relevant co-contextual information from the paired point cloud. Note that convolution [4] has been proved to be effective for local feature extraction and the Transformer [26] has been proved to be suitable for global search. We propose a Convolution and Transformer encoder (CT-encoder), where each encoder layer consists of a convolution branch and a Transformer branch. By applying our CT-encoder, different scale point features are learned, where the receptive field of each point feature is enlarged by summarizing the output of the previous layer. The size of the input point set of each encoder layer is reduced by down-sampling the input point set of the previous encoder layer. This operation is memory efficient in implementation in response to the huge memory consumption of the Transformer. Here, we take one encoder layer as an example for a clear explanation.

1) Convolution Branch: The convolution-based 3D point cloud processing [3], [4] has shown advanced performance in capturing the local geometry. It not only defines the convolution area according to the kernel points, but also achieves density invariance by considering all supporting neighbors. We build our convolution branch by adopting this convolution operation to capture local geometry.

Formally, the input of the \( \ell \)-th encoder layer includes the point set \( \mathbf{X}^\ell = \{ \mathbf{x}_{i}^\ell \} \in \mathbb{R}^{N^\ell \times 3} \), the corresponding point feature set \( \mathbf{F}^\ell = \{ \mathbf{f}_{x}^\ell \} \in \mathbb{R}^{N^\ell \times d^f} \), and the point set of next layer \( \mathbf{X}^{\ell+1} = \{ \mathbf{x}_{i}^{\ell+1} \} \in \mathbb{R}^{N^{\ell+1} \times 3} \), where \( \mathbf{X}^{\ell+1} \subset \mathbf{X}^\ell \), \( N^\ell \) and \( N^{\ell+1} \) indicate the sizes of the point sets, \( d^f \) is the feature dimension of the \( \ell \)-th encoder layer. Note that the initial point feature is the 3D coordinate. Meanwhile, a density normalization term considering the whole supporting neighbor set is applied to ensure sparsity invariance. Then, the convolution feature of the next layer \( \psi_{x_{i}^{\ell+1}} \) at the point \( \mathbf{x}_{i}^{\ell+1} \in \mathbf{X}^{\ell+1} \) is obtained from the convolution branch using the kernel \( g \),

\[
\psi_{x_{i}^{\ell+1}} = \frac{1}{|\mathcal{N}_{x_{i}^{\ell+1}}^{\ell+1}|} \sum_{\mathbf{x}_{j}^{\ell+1} \in \mathcal{N}_{x_{i}^{\ell+1}}^{\ell+1}} g(\mathbf{x}_{i}^{\ell+1} - \mathbf{x}_{j}^{\ell+1}) \mathbf{f}_{x_{j}^{\ell+1}},
\]

where \( \mathcal{N}_{x_{i}^{\ell+1}}^{\ell+1} \) is the radius neighborhood of point \( \mathbf{x}_{i}^{\ell+1} \) in the current point set, \( i.e. \mathcal{N}_{x_{i}^{\ell+1}}^{\ell+1} = \{ \mathbf{x}_{j}^{\ell+1} \in \mathbf{X}^{\ell+1} \mid \|\mathbf{x}_{i}^{\ell+1} - \mathbf{x}_{j}^{\ell+1}\| \leq r^{\ell+1} \} \), where \( r^{\ell+1} \) is a pre-defined radius. The kernel function \( g(\cdot) \) is defined as \( g(\mathbf{x}_{i}^{\ell+1} - \mathbf{x}_{j}^{\ell+1}) = \sum_{i=1}^{S} h(\mathbf{x}_{i}^{\ell+1} - \mathbf{x}_{j}^{\ell+1}) \mathbf{W}_{i} \), where \( h(\mathbf{x}_{i}^{\ell+1} - \mathbf{x}_{j}^{\ell+1}, \hat{x}_{j}^{\ell+1}) = \max(0,1 - \|\mathbf{x}_{i}^{\ell+1} - \mathbf{x}_{j}^{\ell+1} - \hat{x}_{j}^{\ell+1}\|/\sigma) \) is the correlation function between the kernel point \( \hat{x}_{j}^{\ell+1} \) and the supporting point \( \mathbf{x}_{j}^{\ell+1} \), \( \sigma \) is a hyperparameter. \( \mathbf{W}_{i} \) is the weight matrix of the kernel point \( \hat{x}_{j}^{\ell+1} \), and \( S \) is the number of kernel
points. More details can be seen in [3], [4]. We denote the set of the obtained feature as $\Psi_{X,t+1} = [\psi_{X,t+1}] \in \mathbb{R}^{N_{X,t+1} \times D_{t+1}}$, and $D_{t+1}$ is the feature dimension of the next encoder layer.

2) Transformer Branch: Besides learning the point feature by focusing on the source or target point cloud individually, we utilize Transformer [26] to construct our Transformer branch in each encoder layer to explore the repetitive structure from the paired point clouds. The architecture of this branch is shown in Fig. 3, and refer to [26] for more details.

As mentioned above, the features of the source and target point clouds at the current encoder layer are notated as $\mathbf{F}_X \in \mathbb{R}^{N_X \times D^t}$ and $\mathbf{F}_Y \in \mathbb{R}^{N_Y \times D^t}$ respectively, $N_X$ and $N_Y$ are the sizes of point sets, where we add the subscripts $X$, $Y$ to distinguish the source and target. By analyzing $\mathbf{F}_X$ and $\mathbf{F}_Y$ together using the Transformers $\eta_1^t$, $\eta_2^t$, the co-contextual information is collected as,

$$
\begin{align}
\Phi_X &= \eta_1^t(\mathbf{F}_X, \mathbf{F}_Y) \\
\Phi_Y &= \eta_2^t(\mathbf{F}_Y, \mathbf{F}_X)
\end{align}
$$

(2)

where $\eta_1^t : \mathbb{R}^{N_X \times D^t} \times \mathbb{R}^{N_Y \times D^t} \rightarrow \mathbb{R}^{N_X \times D^{t+1}}$, and $\eta_2^t : \mathbb{R}^{N_Y \times D^t} \times \mathbb{R}^{N_X \times D^t} \rightarrow \mathbb{R}^{N_Y \times D^{t+1}}$. Then we select the feature subsets corresponding to the $\mathbf{X}^{t+1}$, $\mathbf{Y}^{t+1}$, notated as $\Phi_{X,t+1} \in \mathbb{R}^{N_X^{t+1} \times D^{t+1}}$, $\Phi_{Y,t+1} \in \mathbb{R}^{N_Y^{t+1} \times D^{t+1}}$, $N_X^{t+1}$ and $N_Y^{t+1}$ are point sizes of the next encoder layer.

3) Different Scale Features Extraction: In the $t$-th encoder layer, after capturing the results of both convolution branch and Transformer branch, we integrate them for the features of the next encoder layer,

$$
\begin{align}
\mathbf{F}_{X,t+1} &= \psi_{X,t+1} + \Phi_{X,t+1} \\
\mathbf{F}_{Y,t+1} &= \psi_{Y,t+1} + \Phi_{Y,t+1}
\end{align}
$$

(3)

Then, by several encoder layers, different scale features are achieved. Notably, in each encoder layer, the information from both the current point cloud and the paired point cloud are mixed together. Compared with DCP [29] and PRNet [30], our method facilitates the matchability of correspondences further by exploiting and capturing co-contextual information from multiple scale features.

Fig. 2. Illustration of our proposed EDFNet architecture. Given two point clouds, the CT-encoder network is used to extract different scale features, where convolution and Transformer are employed to extract local information of the current point cloud and the repetitive structure information of paired point clouds, respectively. Then, we design a dynamic fusion module to fuse these different scale features to the final feature descriptor. The entire network is trained with a metric learning loss. Note that the robot indicates the Transformer module.

Fig. 3. The architecture of the Transformer branch. Two inputs correspond to the feature maps of source and target respectively. At first, the input feature is embedded by MLP to the query, key and value respectively. The skip connection structure is applied by adding the processed feature map to the original feature map, and the results are normalized further. The feed forward module is implemented by the MLP. By the interaction between the source and target, the matchability is augmented by the searched repetitive local structures.

B. Dynamic Fusion

In the encoder network mentioned above, the original point set has been down-sampled layer-by-layer and the learned features become sparse corresponding to those down-sampled points with enlarged receptive fields. However, these different scale features always face a struggle between robustness and discriminativeness due to the unpredictable interference. We observe an intuitive rule, i.e. the clean scale features are usually consistent in describing similar local geometries, while the contaminated ones are usually diverse due to the irregular disturbers. Then, we dynamically and adaptively fuse these obtained different scale features depending on the consistency of their distributions rather than a fixed trained
decoder network. Essentially, these clean scale features cluster naturally, and we concentrate on them more and perform larger aggregation weights on them during our dynamic fusion process. In this way, the final obtained descriptor is guided to approach to the consistent and clean scale features while keeping away from the disturbed ones.

Specifically, we firstly propagate the obtained sparse point features to the dense point features corresponding to the original input point cloud, and then fuse these dense different scale features through our designed dynamic fusion module. Note that the source and target point clouds are handled in the same manner in this module, we take the source point cloud as an example for a clear explanation.

1) Up-Sampling for Dense Point Features: To achieve dense point features, we adopt a propagation strategy with distance-based interpolation here. Specifically, given the dense point features $F^\ell \in \mathbb{R}^{N^\ell \times D^\ell}$ learned in the encoder network as mentioned above, we interpolate them to the coordinates of the original input point cloud $X \in \mathbb{R}^{N \times 3}$, where $N$ is the size of the original input point cloud. Notably, among the many choices for interpolation, we use the inverse distance weighted average based on K-nearest neighbors in our implementation, which is given as,

$$f_{x_i}^{\ell-d} = \frac{\sum_{k=1}^{K} w_k f_{x_k}^{\ell}}{\sum_{k=1}^{K} w_k}, \quad f_{x_k}^{\ell} \in F^\ell,$$  \hspace{1cm} (4)$$

where $w_k = 1/d(x_i, x_k)^p$, $d(\cdot, \cdot)$ indicates the Euclidean distance, $p$ is the exponent, $x_i \in X$ is the current point and the $x_k \in X^\ell$ indicates the $k$-th neighbor point in the K-nearest neighbors of $x_i$. Then, the dense feature of layer $\ell$ is returned as $F^\ell_{d} = [f_{x_i}^{\ell-d}] \in \mathbb{R}^{N \times D^\ell}$. By the above up-sampling operation, all these different scale features have been unified to the same density. Notably, during our dynamic fusion process, we use the inverse distance weighted average based on K-nearest neighbors in our implementation, which is given as,

$$f_{x_i}^{\ell-d} = \frac{\sum_{k=1}^{K} w_k f_{x_k}^{\ell}}{\sum_{k=1}^{K} w_k}, \quad f_{x_k}^{\ell} \in F^\ell,$$  \hspace{1cm} (4)$$

Next, we update the parameters according to the similarity between $s_r$ and each scale feature, i.e.

$$b_{\ell+1} = b_{\ell} + s_r y^{T}, \quad \ell = [1, 2, \ldots, L].$$  \hspace{1cm} (6)$$

Note that the coefficients of consistent scale features will increase as the iteration progresses due to the large similarity. After $T$ iterations, we get the final point feature descriptor as $y^* = \sum_{\ell=1}^{L} c_{\ell} y^{\ell}$. The whole iteration likes a clustering process to recognize the consistent ones with large coefficients, which will contribute more during the fusion. The complete fusion algorithm can be seen in Alg. 1.

Due to our dynamic fusion, the final obtained features approach to the consistent and clean scale features and keep away from the contaminated ones. Specifically, for the point far away from the disturbers, all scale features are consistent in describing a similar geometry structure. For the point near the disturbers, all scale features are disturbed. In these two rare cases, our dynamic fusion degenerates to equally processing like addition or concatenation, which is the bottleneck of our method for future study. However, in most cases, where disturbers exist in some larger scale features, the final fused feature is determined by the clean and consistent scale features more. Thus, under the premise of dodging the interference, we achieve the discriminative descriptor by jointly as many consistent and clean scale features as possible. Note that our dynamic fusion module is deep learning-free without any learning parameters. Thus, it determines the final fused feature descriptor totally depending on the different scale features themselves rather than a customized trained decoder network. This operation reiterates the generalization ability further when it is transferred to different datasets in practical applications.

C. Loss Function

In our method, the metric learning with the contrastive loss is used to train our EDFNet. Formally, given the source and target point clouds $X$ and $Y$, and a correspondence set $\Omega = [(x_i, y_i) | i = 1, \ldots, n]$, where the size of this set is $n$ and $(x_i, y_i)$ indicates the $i$-th correspondence, we note the final learned descriptors of this $i$-th pair of points as
Algorithm 1 Dynamic Fusion Algorithm

Input: Number of scales \( L \), features at different scales \( y^t, t = [1, 2, \ldots, L] \), iteration number \( T \).

Output: Fused point feature descriptor \( y^* \).

Initialize parameters \( b_0^L, b_2^L, \ldots, b_L^L = 0 \).

while \( r \leq T - 1 \) do
  \[
  \begin{align*}
  c^t_r, & = \text{softmax}(b^t_r, b^t_{2r}, \ldots, b^t_{L}); \\
  s_r & = \sum_{l=1}^{L} c^t_r \cdot \gamma_r; \\
  b^t_{r+1} & = b^t_r + s_r \cdot y^T. \\
  end while
  \]

where \( \gamma_r \) is the dynamic fusion coefficient.

\[
\gamma^*_r \text{ and } \gamma^*_y \text{ respectively. We use the Euclidean distance to measure the disparity between two descriptors. Then, the distance between the descriptors of a positive pair of points is defined as } d^*_i = ||y^*_x - y^*_y||. \text{ The distance between the descriptors of a negative pair of points is defined as } d^*_i = \min(||y^*_x - y^*_y||) \text{ s.t. } ||y - y|| > \epsilon, \text{ where } \epsilon \text{ is the safe radius, } y^*_y \text{ is the hardest negative sample that lies outside the safe radius of the correct corresponding point } y. \text{ Then, our final loss function is defined as,}
\]

\[
L = \frac{1}{n} \sum_i \left[ \max(0, d^*_i - M^*_i) + \max(0, M^*_i - d^*_i) \right], \quad (7)
\]

where \( M^*_i \) indicates the margin for positive pairs and \( M^*_i \) indicates the margin for negative pairs.

D. Implementation Details

During the training, we apply data augmentation to each point cloud including the Gaussian jitter with mean 0, standard deviation 0.005, the random scaling \( [0.9, 1.1] \), and the random rotation \( [0^\circ, 360^\circ] \) along an arbitrary axis.

We set the number of scales \( L = 5 \). Then, we use grid down-sampling to reduce the size of the input point cloud layer-by-layer. The numbers of feature channels are set to \( 128, 256, 512, 1024 \) in five encoder layers, respectively. In up-sampling procedure, the parameter \( K \) in K-nearest neighbor is set to 24. And all point features are finally unified to 32 dimension. The positive margin \( M^*_p = 0.1 \) and negative margin \( M^*_n = 1.4 \) in our contrastive loss. The size of the correspondence set \( n = 64 \). The iteration number \( T = 5 \). Our EDFNet is implemented in Pytorch 1.7.0 and trained on RTX 3090. We use the SGD optimizer with momentum parameter 0.98 and weight decay parameter 1e\(^{-6}\) to optimize the network with the initial learning rate of 0.1 for 150 epochs. The learning rate scheduler adopts exponentially decayed with gamma parameter 0.97.

IV. Experiments and Evaluation

In this section, we report the accuracy and robustness of our proposed EDFNet regarding 3D point cloud matching. To this end, we evaluate our method on several benchmark datasets including the typical indoor dataset, 3DMatch [46], and the outdoor dataset, KITTI [57]. Besides, to verify the robustness and generalization ability of our method, we perform the test on both rigid and non-rigid point cloud data further.

A. Indoor Dataset: 3DMatch

The training and testing sets of the 3DMatch dataset, and the ground truth transformation have been provided officially [46]. The ground truth correspondences are constructed according to the nearest neighbor principle. Specifically, if the distance between the transformed point, achieved by the ground truth transformation, and the nearest point is less than a pre-defined threshold (5cm), this correspondence is confirmed as a ground truth correspondence. Following [4], during the training, all the point cloud fragment pairs with at least 30% overlap are selected from the official training set. The test set contains 8 scenes with partially overlapped point cloud fragments. All fragments have been down-sampled with 2.5cm voxel size.

1) Evaluation Metrics: Following [4], [19], three evaluation metrics are used including Inlier Ratio, Feature Matching Recall, and Registration Recall. Specifically, given two point clouds \( X \) and \( Y \), a feature learning network maps the input points to feature descriptors. Then the index set of predicted correspondences is notated as \( \pi = (i, j) \), where \( x_i \in X \) and \( y_j \in Y \) are predicted correspondences obtained by nearest neighbor searching in the feature space.

- **Inlier Ratio.** This metric reports the mean percentage of successfully identified inliers. Given a set of point cloud pairs \( M \), the inlier ratio \( \mathcal{I} \) of one pair \( m \) is defined as,

\[
\mathcal{I} = \frac{1}{|\pi|} \sum_{(i,j) \in \pi} \mathbb{I}(\|T_m(x_i) - y_j\| < \tau_1), \quad (8)
\]

where \( T_m(\cdot) \) is the ground truth transformation of the point cloud pair \( m \). \( \mathbb{I}(\cdot) \) is a binary function. It equals to 1 if the input is true, otherwise it equals to 0. \( \tau_1 \) is the inlier distance threshold, which is set to \( \tau_1 = 10cm \) following [46]. Finally, we report the mean of inlier ratios of all pairs of point clouds as the evaluation result.

- **Feature Matching Recall.** This metric is defined as the percentage of successful alignment whose inlier ratio is above a threshold to measures the matching quality of pairwise registration. Then, the feature matching recall is defined as,

\[
\eta_{\text{match}} = \frac{1}{|M|} \sum_{M} \mathbb{I}((\mathcal{I} > \tau_2), \quad (9)
\]

where \( \tau_2 \) is the inlier ratio threshold of the point cloud pair. And the point cloud pairs with more than \( \tau_2 = 5\% \) inliers will be counted as one successfully matched pair.

- **Registration Recall.** This metric is defined as the percentage of successful alignment whose transformation estimation error is below a threshold. At first, we use a robust local registration algorithm, RANSAC with 50000 max iterations, to estimate the rigid transformation between the two input point clouds. Then, the RMSE distance of the ground truth correspondences under the estimated transformation matrix is calculated. The ground truth correspondence set for the point cloud pair \( m \) is given as \( \Omega^m = \{x^*_i \in X, y^*_j \in Y \} \). Then, the
RMSE distance notated as $\text{RMSE}_{\text{dis}}$ of this point cloud pair $m$ is defined as,

$$
\text{RMSE}_{\text{dis}} = \sqrt{\frac{1}{|\Omega|^2} \sum_{(x^*, y^*) \in \Omega} \| \hat{T}_m(x^*) - y^* \|^2},
$$

where $\hat{T}_m(\cdot)$ is the estimated transformation of the point cloud pair $m$ returned by RANSAC. Finally, the registration recall is defined as,

$$
\eta_{rr} = \frac{1}{|M|} \sum_{m} 1(\text{RMSE}_{\text{dis}} < \tau_3),
$$

where $\tau_3$ is set to 0.2 following [4].

2) Evaluation Points Sampling: Following [4], during the test, we select 5000 points to conduct the evaluation rather than taking all points. Here, two strategies of points selection are employed, i.e., randomly sampled points ($\text{rand}$) and keypoints predicted by a detector ($\text{pred}$). For the first one, we randomly select a specified number of points from the pairs of the point cloud fragments for evaluation. For the second one, we use the detector proposed in [4], which solves a score to indicate the matchability of each point. We select a specified number of points with higher scores for our evaluation. Notably, in our implementation of the $\text{pred}$ setting, we add the detector after our original EDFNet, and train the network using the equal weighted loss functions of ours in (7) and the detector loss function provided in [4]. Please refer to [4] for more details about the detector.

3) Evaluation: In this part, we compare our EDFNet with the SOTA methods on the 3DMatch dataset. In Table I, we report the average feature matching recall result and its standard deviation indicated by $\text{AVG}$ and $\text{STD}$ respectively. The results of the handcrafted features, such as Spin [13], SHOT [16], usually fail in such a challenging dataset. The learning-based methods perform better than these handcrafted methods. Our proposed EDFNet performs the second-top performance among all methods, whose results are the same as LMVD [61]. SpinNet [62] achieves a slightly better result than ours (0.1%) due to the customized rotation invariant operation based on the rigid transformation assumption. This makes its obtained descriptor more specific. And SpinNet achieves more accurate results in this evaluation on the 3DMatch dataset because a rigid transformation exists in the input point cloud pair. By contrast, ours is devoted to learning a matching task-specific descriptor based on the local geometry structure of the input point clouds. Thus, our EDFNet can be extended to handle the non-rigid point cloud data as presented in Sec. IV-C, which brings more general applications.

Besides, we also provide the time consumption per feature. D3Feat presents the best time-efficiency, and ours is slightly slower than D3Feat and FCGF due to the multiple Transformer modules and the iteration in the dynamic fusion module. Note that DGR [63] performs a further optimization on the initial matches, which are determined based on the FCGF descriptor. DGR learns a weight to indicate the reliability of each match. Here, after normalizing the weight coefficients, we select the matches whose weight $\eta > 0.5$ to conduct the evaluation. In Table I, we can see that DGR presents a better feature matching recall result than FCGF after this further optimization at the cost of discarding many matches whose weight $\eta \leq 0.5$. Nevertheless, our proposed EDFNet still outperforms DGR.

According to the matching results based on the learned point feature descriptor, we accurately register the two input point clouds. To validate our method intuitively, we visualize some registration results in Fig. 4, from which we can find that our method achieves accurate registration. Besides, we also present the qualitative results of the distribution of the learned task-specific descriptors. Specifically, we reduce the dimension of the descriptors to 3-dimension by T-SNE algorithm to represent the RGB in color space. Then, we show the pair of point clouds where each point is marked with the corresponding color. From Fig. 5, we find that the point correspondences tend to be marked with the same color under the interference, which validates the effectiveness of our learned descriptor for the matching task.

### B. Outdoor Dataset: KITTI

In this section, we evaluate our method on the outdoor dataset, KITTI, which consists of 11 sequences of outdoor driving scenarios. The sequences 0-5 are used for training, 6-7 for validation, and 8-10 for testing following [19]. Since the GPS ground truth is noisy, we firstly use the standard ICP [64] to refine the ground truth transformation. The ground truth correspondences are determined as the 3DMatch, where the threshold is set to 5cm. We select LiDAR scan pairs with at least 10m intervals to construct our input point cloud pairs. All point clouds are voxel down-sampled with 30cm voxel size.

1) Evaluation Metrics: Following DGR [63], we evaluate the estimated transformation matrices by Translation

| Methods          | AVG(%) | STD | Dim | Times(ms) |
|------------------|--------|-----|-----|-----------|
| Spin [13]        | 22.7   | 11.4| 153 | 0.093     |
| FPFH [15]        | 35.9   | 13.4| 32  | 0.022     |
| SHOT [16]        | 23.8   | 10.9| 332 | 0.201     |
| USC [36]         | 40.0   | 12.5| 1980| 2.483     |
| PointNet [17]    | 47.1   | 12.7| 256 | 0.117     |
| 3DMatch [46]     | 59.6   | 8.8 | 512 | 2.151     |
| CGF [58]         | 58.2   | 14.2| 32  | 0.981     |
| PPFNet [24]      | 62.3   | 10.8| 64  | 1.510     |
| PPF-FoldNet [25] | 71.8   | 10.5| 512 | 0.531     |
| DirectReg [59]   | 74.6   | 9.4 | 512 | 0.531     |
| CapsuleNet [60]  | 80.7   | 6.2 | 512 | 0.808     |
| PerfertMatch [23]| 94.7   | 2.7 | 32  | 3.678     |
| FCGF [19]        | 95.2   | 2.9 | 32  | 0.013     |
| D3Feat(rand) [4] | 95.3   | 2.7 | 32  | 0.005     |
| D3Feat(pred) [4] | 95.8   | 2.9 | 32  | 0.005     |
| LMVD [61]        | 97.5   | 2.8 | 32  | -         |
| SpinNet [62]     | 97.6   | 1.9 | 32  | 4.216     |
| DGR [63]         | 97.1   | 2.7 | 32  | 0.842     |
| EDFNet(rand)     | 95.8   | 2.7 | 32  | 0.015     |
| EDFNet(pred)     | 97.5   | 2.7 | 32  | 0.016     |
Fig. 4. Visualization of registration results on the 3DMatch dataset. The left column indicates two partial input point clouds with significant rigid transformation. The right column indicates the accurate registration result achieved by performing the RANSAC on the point correspondences constructed based on our learned descriptors.

Fig. 5. Visualization of the distribution of our learned feature descriptor. Left is the source point cloud and right is the target point cloud. Each high-dimensional feature is reduced to 3-dimension by the T-SNE algorithm. Then, we normalize them to the color space to represent the color of each point. We can find that the correspondences tend to be marked with the same color. In other words, consistent feature descriptors are achieved for correspondences, which validates the effectiveness of our EDFNet in the point cloud matching task.

Error (TE) and Rotation Error (RE). The rotation error is calculated as,

$$ RE = \arccos \left( \frac{\text{Tr}(R_{\text{pre}}^TR_{\text{gt}}) - 1}{2} \right), $$

where $R_{\text{gt}}$ and $R_{\text{pre}}$ indicate the ground truth and predicted rotation matrix. $\text{Tr}(\cdot)$ solves the trace of the input matrix. The translation error is calculated by,

$$ TE = \|t_{\text{pre}} - t_{\text{gt}}\|_2^2, $$

where $t_{\text{pre}}$ and $t_{\text{gt}}$ indicate the predicted and ground truth translation vector, respectively. Besides RE and TE, we also report the recall results, which indicate the percentage of successful pairwise registration. The successful registration is confirmed if $TE \leq 0.6m$ and $RE \leq 5^\circ$ following [63]. Average TE and RE reported in Table II are computed only on these successfully registered pairs since the poses returned by failed registrations can be drastically different from the ground truth, making the error metrics unreliable.

2) Evaluation: We compare our EDFNet with FCGF, 3DFeat-Net, and D3Feat, which are committed to extracting advanced point feature descriptors. Besides, we also compare our method with SOTA learning-based point cloud registration methods, including DGR [63] and PointDSC [66], which focus on selecting reliable correspondences from the initially constructed correspondences. Notably, DCP [29] and PRNet [30] are divergent in this setting. We evaluate two versions of EDFNet. For the first version, the final transformation is obtained by performing RANSAC with 50,000 max iterations after initial nearest neighbor feature matching. The registration results of FGR, 3DFeat-Net, FCGF, SpinNet and D3Feat are also achieved by this operation. For the second version, we use the proposed EDFNet to extract point features instead of the original FCGF in DGR framework to estimate the final transformation. The registration results are provided in Table II, where EDFNet outperforms the SOTA point feature descriptor learning methods in all metrics. But it underperforms the learning-based registration methods, DGR [63] and PointDSC [66] since they optimize the matching result further after point feature learning. In addition, the final registration performance is further improved when our method is integrated into the advanced learning-based point cloud registration framework DGR. Moreover, to illustrate our method intuitively, we present some typical registration results of EDFNet + DGR in Fig. 6, where ours achieves successful registration.

C. Generalization Evaluation

1) Evaluation on Rigid Point Cloud Data: To evaluate the robustness and generalization ability of our EDFNet, we use the model trained on the 3DMatch dataset to conduct the test on four outdoor laser scan datasets (Gazebo-Summer, Gazebo-Winter, Wood-Summer and Wood-Autumn) from the ETH dataset [67] following [23]. The evaluation metric is feature matching recall, and the experiments adopt the same settings as the previous evaluation on 3DMatch. The baselines involve two settings for the voxelsize, i.e. 5cm and

| Methods          | TE(cm) | RE(°) | Recall(%) |
|------------------|--------|-------|-----------|
|                  | AVG    | STD   | AVG       | STD       |
| FGR [65]         | 42.8   | 19.7  | 1.89      | 1.11      | 2.74      |
| 3DFeat-Net [18]  | 21.4   | 17.2  | 0.44      | 0.36      | 45.72     |
| FCGF [19]        | 10.5   | 1.37  | 0.30      | 0.29      | 97.34     |
| D3Feat [4]       | 6.54   | 1.06  | 0.33      | 0.23      | 97.81     |
| SpinNet [62]     | 9.13   | 1.42  | 0.73      | 0.41      | 97.50     |
| EDFNet           | 5.12   | 1.01  | 0.31      | 0.20      | 98.14     |
| DGR [63]         | 4.01   | 0.82  | 0.23      | 0.14      | 97.94     |
| PointDSC [66]    | 3.84   | 0.83  | 0.22      | 0.12      | 98.70     |
| EDFNet+DGR       | 3.82   | 0.77  | 0.17      | 0.14      | 98.92     |

TABLE II
REGISTRATION ON THE KITTI DATASET. AVG INDICATES THE AVERAGE RESULT AND STD INDICATES THE STANDARD DEVIATION.
Fig. 6. Registration results on the KITTI dataset. The left is the input point cloud pair while the right is the registration result. The proposed descriptor used for the matching results in accurate registration performance.

6.25cm. For a fair and comprehensive comparison, we evaluate ours in these two settings. The results of PerfectMatch with 6.25cm voxel size and FCGF with 5cm voxel size are provided. For D3Feat, we report the results on both 6.25cm and 5cm voxel sizes. As shown in Table III, under the 5cm voxel size setting, our EDFNet presents better generalization ability than other baselines. Under the 6.25cm voxel size setting, PerfectMatch achieves better results than our EDFNet on the Gazebo-Summer and Gazebo-Winter. However, for Wood-Summer and Wood-Autumn, ours outperforms PerfectMatch. And our EDFNet is better than PerfectMatch in terms of the final average results. We suspect that the reason is that there are more noise and outliers in the Wood sub-dataset than the Gazebo sub-dataset, which decreases the performance of PerfectMatch. And our EDFNet exhibits more robustness and presents more stable matching results. LMVD [61] and SpinNet [62] do not report their voxel sizes, while SpinNet obtains the most accurate results among all methods whether 5cm voxel size or 6.25cm voxel size. We suspect the reason is that SpinNet designs a learning-free proactive operation for rotation invariance, which helps improve the generalization ability significantly. Our learning-free module, dynamic fusion, is post-processing after different scale features learning, which is affected more heavily than SpinNet. Nevertheless, ours achieves the second-top performance among all methods.

We report the average Inlier Ratio here to evaluate the matching performance. Different from the version used before, where the threshold to confirm the correspondence inlier is set as a fixed distance value, we adopt a self-adaptive threshold as follows. To avoid confusion caused by the fixed distance based threshold \( r_1 \), we notate this self-adaptive threshold as \( \kappa \), which is formulated based on the K-nearest neighbor principle. Specifically, \( \kappa_i = \frac{1}{K} \sum_{j=1}^{K} l_{x_i, x_j} \), where \( l_{x_i, x_j} \) is the Euclidean distance between the points \( x_i \) and \( x_j \), \( KNN(\cdot) \) represents the K-nearest neighbor. In other words, for the \( i \)-th point \( x_i \), \( \kappa_i \) is computed as the mean distances between the point \( x_i \) and its \( K \) nearest neighbor points.

As shown in Fig. 7, we report the results according to different \( K \) value. During the evaluation, the nearest neighbor principle in feature space is applied to determine the matching after point feature learning. For PointNet [17], DGNN [1], the metric learning with contrastive loss function is used to
train the network to learn the point features. Besides the point feature learning based methods, two versions of CorrNet3D [68], i.e., supervised version (notated as S-CorrNet3D) and unsupervised version (U-CorrNet3D) are also evaluated. U-F3D and S-F3D indicate unsupervised FlowNet3D [71] and supervised FlowNet3D respectively. Among all point feature learning based methods, ours achieves the best matching results. Notably, SpinNet is designed for the rigid point cloud, which underperforms ours significantly. Comparing to the SOTA point cloud matching method, SpinNet, besides the time efficiency as introduced in Sec. IV-A, our EDFNet presents another advantage here. Since SpinNet is designed based on the assumption of rigid transformation, it fails and performs decreased performance when handling the non-rigid point cloud data. By contrast, our EDFNet also achieves reliable results because it is built by focusing on local geometry structure learning, which enables more broad applications. Meanwhile, compared with the SOTA method, S-CorrNet3D, ours also achieves comparable or even better performance, which validates the generalization ability of our method sufficiently. Meanwhile, for an intuitive understanding of our method, we provide some matching results on these non-rigid point cloud pairs in Fig. 8. Obviously, our method achieves advanced matching performance.

3) Evaluation on Non-Rigid Point Cloud Sequences Data: To demonstrate the matching effectiveness and temporal consistency of our proposed method, we test our EDFNet on a dynamic point cloud sequences dataset. We adopt the 8iVFB dataset [72] here, which is a real scanned dataset consisting of four point cloud sequences named Long Dress, Loot, Red and Black, and Solder respectively. Each sequence is generated by scanning a moving human body. We uniformly sample 5000 points for each point cloud. Then, we construct the input point cloud pairs in turn based on these processed point cloud frames.

However, to the best of our knowledge, the real scanned dynamic point cloud sequence dataset usually lacks ground truth information about point matches. Thus, we adopt another evaluation metric here based on the Chamfer distance. Specifically, we estimate the corresponding point for each point of the source point cloud from the target point cloud. To quantitatively compare ours with baselines, we calculate the Chamfer distance between the target point cloud and the corresponding points of the source point cloud. Formally, we notate the source and target point clouds as \( \mathbf{X} \) and \( \mathbf{Y} \), and the corresponding point set of \( \mathbf{X} \) as \( \hat{\mathbf{X}} \). Then, the Chamfer distance \( \text{CD} \) between \( \hat{\mathbf{X}} \) and \( \mathbf{Y} \) is calculated by

\[
\text{CD}(\hat{\mathbf{X}}, \mathbf{Y}) = \frac{1}{N_{\hat{\mathbf{X}}}} \sum_{\hat{x} \in \hat{\mathbf{X}}} \min_{y \in \mathbf{Y}} \|\hat{x} - y\|_2^2 + \frac{1}{N_{\mathbf{Y}}} \sum_{y \in \mathbf{Y}} \min_{\hat{x} \in \hat{\mathbf{X}}} \|\hat{x} - y\|_2^2,
\]

where \( N_{\hat{\mathbf{X}}} \) and \( N_{\mathbf{Y}} \) indicate the numbers of points in the \( \hat{\mathbf{X}} \) and \( \mathbf{Y} \) respectively. Finally, we report the average Chamfer distance.

As reported in Table IV, we can find that our EDFNet achieves the best results among all baselines, which validates the matching effectiveness of our method. To further present the temporal consistency of our method for handling the point cloud sequence, we visualize the matching results of some consecutive point cloud frames of 8iVFB dataset [72] in Fig. 9. To clearly observe the temporal consistency of results, we just show the matching of a local part, (e.g. the hands part of the human in Loot sequence and the foot part of the human in Long Dress sequence). Here, we take the Loot as an example for a clear explanation. Specifically, for the first frame, we select some points of hand, notated as \( \mathbf{X}_1 \). Then, we solve the corresponding points of \( \mathbf{X}_1 \) in the second point cloud frame, notated as \( \hat{\mathbf{X}}_2 \). Next, we take this second frame as the source point cloud and the third frame as the target point cloud. And according to these corresponding points \( \hat{\mathbf{X}}_2 \) in the second frame, we estimate the corresponding points of \( \hat{\mathbf{X}}_2 \) in the third frame, notated as \( \hat{\mathbf{X}}_3 \). This process continues in these consecutive frames. Finally, we find that all the corresponding points in these consecutive frames, i.e. \( \mathbf{X}_1, \hat{\mathbf{X}}_2, \hat{\mathbf{X}}_3, \ldots \), keep stability in the hand part. This result validates the temporal consistency of ours for handling the point cloud sequence.

### TABLE IV

| Methods          | Long Dress | Loot  | Red and Black | Soldier |
|------------------|------------|-------|---------------|---------|
| D3Feat [4]       | 0.041      | 0.033 | 0.037         | 0.055   |
| PointNet [17]    | 0.064      | 0.057 | 0.061         | 0.072   |
| S-CorrNet3D [68] | 0.014      | 0.012 | 0.012         | 0.023   |
| U-CorrNet3D [68] | 0.057      | 0.051 | 0.055         | 0.063   |
| S-F3D [71]       | 0.019      | 0.017 | 0.018         | 0.025   |
| U-F3D [71]       | 0.061      | 0.052 | 0.054         | 0.066   |
| SpinNet [62]     | 0.022      | 0.019 | 0.021         | 0.029   |
| XPCorr [3]       | 0.034      | 0.027 | 0.031         | 0.047   |
| DGCNN [11]       | 0.047      | 0.042 | 0.045         | 0.059   |
| EDFNet           | **0.012**  | **0.010** | **0.010**   | **0.019** |

D. Ablation Study

Here, we provide several important ablation studies to analyze our proposed method. The experiments are conducted on the 3DMatch dataset unless otherwise explained.

---

Fig. 8. Matching results of our method on non-rigid point clouds. Green indicates the source point cloud and blue indicates the target. Lines present the matching results, where our method achieves advanced performance.
Fig. 9. Matching results for handing point cloud sequences in Loot and Long Dress from 8iVFB dataset [72]. Here, we take the Loot as an example for a clear explanation. 1) indicates the frame index of the first frame. For the first frame, we select some points in the hand part, notated as $X_1$. Then, we solve the corresponding points of $X_1$ in the second point cloud frame, notated as $\hat{X}_2$. Next, we take this second frame as the source point cloud and the third frame as the target point cloud. And according to these corresponding points $\hat{X}_2$ in the second frame, we estimate the corresponding points of $\hat{X}_2$ in the third frame, notated as $\hat{X}_3$. This process continues in these consecutive frames. Finally, we find that all the corresponding points in consecutive frames, i.e., $X_1, \hat{X}_2, \hat{X}_3, \ldots$, keep stability in the hand part.

1) Performance Under Different Numbers of Selected Points: As mentioned above, the results of our evaluations are counted in some selected 3D points following D3Feat [4]. Here, the performance under different numbers of selected points captured by two different strategies (i.e., rand and pred) is presented, where the number of selected points is reduced from 5000 to 2500, 1000, 500, and even 250. The final results are presented in Table V. We can find that, when the points number gets smaller, the performance of PerfectMatch, FCGF, and D3Feat(rand) drops at a similar magnitude while EDFNet(rand) keeps stability and is better than others. When the detector is equipped, EDFNet(pred) achieves the best matching quality results under all different numbers of selected points on both feature matching recall and registration recall metrics. As for the metric of inlier ratio, our proposed EDFNet also achieves better performance in most settings compared with the SOTA methods. These results validate that our method makes corresponding points more matchable by learning a matching task-specific feature descriptor.

2) Effectiveness of Each Component: Here, we conduct several experiments on the 3DMatch dataset to analyze the effectiveness of each component of our EDFNet. Specifically, to verify the Transformer branch in our encoder, we compare our entire framework with the results returned by learning the source and target independently without Transformer. Besides, to validate the importance of the proposed dynamic fusion module, we compare ours with the pipeline without dynamic fusion including two versions. 1) The decoder proposed in [4] is used here, which adds the different scale features together for the final feature descriptor. 2) The decoder proposed in [1] is tested, where these different scale features are concatenated for the final feature descriptor. The results are reported in Table VI, where other settings are totally the same for a fair comparison. Obviously, in both rand and pred settings with different numbers of selected points, the results are consistent. In particular, EDFNet with the entire pipeline outperforms the pipeline of “EDFNet w/o dynamic fusion but with addition”, the pipeline of “EDFNet w/o dynamic fusion but with concatenation”, and the pipeline of “EDFNet w/o Transformer” in all evaluation metrics. These results have

| Methods          | 5000 | 2500 | 1000 | 500  | 250  |
|------------------|------|------|------|------|------|
| Feature Matching Recall (%) |
| PerfectMatch [23] | 94.7 | 94.2 | 92.6 | 90.1 | 82.9 |
| FCGF [19]        | 95.2 | 95.5 | 94.6 | 93.0 | 89.9 |
| D3Feat(rand) [4]  | 95.3 | 95.1 | 94.2 | 93.6 | 90.8 |
| D3Feat(pred) [4]  | 95.8 | 95.6 | 94.6 | 94.3 | 93.3 |
| EDFNet(rand)     | 95.3 | 95.6 | 95.3 | 96.2 | 93.3 |
| EDFNet(pred)     | 97.5 | 96.5 | 95.7 | 96.2 | 95.5 |
| Registration Recall (%) |
| PerfectMatch [23] | 80.3 | 77.5 | 73.4 | 64.8 | 50.9 |
| FCGF [19]        | 87.3 | 85.8 | 85.8 | 81.0 | 73.0 |
| D3Feat(rand) [4]  | 83.5 | 82.1 | 81.7 | 77.6 | 68.8 |
| D3Feat(pred) [4]  | 82.2 | 84.4 | 84.9 | 82.5 | 79.3 |
| EDFNet(rand)     | 87.5 | 86.2 | 87.9 | 87.9 | 86.8 |
| EDFNet(pred)     | 87.5 | 86.2 | 87.9 | 87.9 | 86.8 |
| Inlier Ratio (%) |
| PerfectMatch [23] | 37.7 | 34.5 | 28.3 | 23.0 | 19.1 |
| FCGF [19]        | 56.9 | 54.5 | 49.1 | 43.3 | 34.7 |
| D3Feat(rand) [4]  | 40.6 | 38.3 | 33.3 | 28.6 | 23.5 |
| D3Feat(pred) [4]  | 40.7 | 40.6 | 42.7 | 44.1 | 45.0 |
| EDFNet(rand)     | 52.5 | 51.0 | 45.3 | 38.1 | 39.7 |
| EDFNet(pred)     | 55.2 | 54.5 | 51.7 | 49.3 | 48.8 |
validated the improvement caused by each component of our EDFNet sufficiently and clearly.

3) Single Scale Feature vs. Multiple Scale Fused Feature:
As reported before, our dynamic fusion module brings better performance than handling different scale features indiscriminately and equally, i.e. addition and concatenation. However, if we only focus on the robustness and dodging disturbers, we can use the minimal scale feature as the final descriptor rather than the fused one. Nevertheless, the discriminativeness is limited in this way. To further validate the effectiveness of the designed dynamic fusion module, we provide the experimental results of using each single scale feature as the final feature descriptor. The experiments are conducted on 3DMatch following the previous settings of Sec. IV-A. Here, according to Table VII, we can find that the dynamic fusion of different scale features generates better performance than using any single scale feature. These results further validate our fused descriptor, which takes both robustness and discriminativeness into account to achieve better 3D point cloud matching.

4) Effectiveness of the Data Augmentation:
As mentioned above, during the training in our experiments, we apply data augmentation, including Gaussian jitter, random scaling, and random rotation. To demonstrate the effectiveness of these data augmentation strategies, we provide a comparison here. Specifically, we train a model without augmentation and evaluate it on the 3DMatch dataset. Other settings are same as Sec. IV-A. The results are presented in Table VIII. We find that the results are consistent under different numbers of the selected points, and the performance of EDFNet is improved with a large margin by the augmentation operation.

5) Contrastive Loss vs. Triplet Loss:
In our method, we use metric learning with contrastive loss to train our EDFNet. Besides the contrastive loss, another typical metric learning loss function has also received widespread attention, i.e. triplet loss function [19]. Here, we provide a comparison between these two loss functions. Other settings are same as Sec. IV-A. The results are given in Table IX, where we provide the approximate epoch when the network converges besides the feature match recall results. We can find that the two metric learning loss functions lead to similar feature match recall

---

**Table VI**

| Methods                      | 5000 | 2500 | 1000 | 500  | 250  |
|------------------------------|------|------|------|------|------|
|                              | Recall (%) |     |      |      |      |
| EDFNet w/o DF w/ Add         | 93.2 | 93.2 | 92.6 | 92.1 | 89.9 |
| EDFNet w/o DF w/ Cat         | 92.9 | 93.0 | 92.7 | 91.7 | 89.4 |
| EDFNet w/ Transformer        | 93.4 | 93.3 | 92.9 | 93.0 | 90.3 |
| **EDFNet**                   | 95.8 | 95.6 | 95.3 | 96.2 | 93.3 |
| EDFNet w/o DF w/ Add & DF    | 94.7 | 94.2 | 93.0 | 93.4 | 92.9 |
| EDFNet w/o DF w/ Cat & DF    | 94.1 | 93.2 | 92.2 | 91.8 | 90.9 |
| EDFNet w/ Transformer & DF   | 94.7 | 94.2 | 93.0 | 93.4 | 92.9 |

**Table VII**

| Methods                      | 5000 | 2500 | 1000 | 500  | 250  |
|------------------------------|------|------|------|------|------|
|                              | Recall (%) |     |      |      |      |
| EDFNet w/o DF w/ Add         | 83.7 | 82.4 | 81.4 | 78.7 | 69.4 |
| EDFNet w/o DF w/ Cat         | 83.2 | 83.1 | 81.6 | 79.1 | 70.3 |
| EDFNet w/ Transformer        | 83.9 | 81.2 | 82.6 | 78.2 | 70.9 |
| **EDFNet**                   | 87.2 | 85.9 | 85.5 | 81.3 | 73.3 |
| EDFNet w/o DF w/ Add & DF    | 83.5 | 83.2 | 83.3 | 81.7 | 82.1 |
| EDFNet w/o DF w/ Cat & DF    | 82.7 | 83.1 | 82.6 | 81.3 | 81.2 |
| EDFNet w/ Transformer & DF   | 84.1 | 82.9 | 84.2 | 82.8 | 83.2 |
| **EDFNet**                   | 87.5 | 86.2 | 87.9 | 87.9 | 86.3 |

**Table VIII**

| Methods                      | 5000 | 2500 | 1000 | 500  | 250  |
|------------------------------|------|------|------|------|------|
|                              | Recall (%) |     |      |      |      |
| EDFNet w/ 1# feature         | 81.9 | 80.8 | 78.1 | 74.3 | 66.2 |
| EDFNet w/ 2# feature         | 82.3 | 81.2 | 81.1 | 77.3 | 70.1 |
| EDFNet w/ 3# feature         | 82.4 | 81.7 | 80.1 | 75.6 | 69.3 |
| EDFNet w/ 4# feature         | 80.4 | 81.1 | 79.5 | 71.8 | 68.4 |
| EDFNet w/ 5# feature         | 80.2 | 78.5 | 72.7 | 70.3 | 67.0 |
| **EDFNet**                   | 87.2 | 85.9 | 85.5 | 81.3 | 73.3 |
| EDFNet w/ 1# feature & DF    | 82.3 | 80.9 | 80.4 | 79.3 | 71.4 |
| EDFNet w/ 2# feature & DF    | 82.1 | 81.3 | 81.1 | 80.1 | 78.4 |
| EDFNet w/ 3# feature & DF    | 83.7 | 81.2 | 81.0 | 81.3 | 79.9 |
| EDFNet w/ 4# feature & DF    | 81.8 | 79.7 | 80.5 | 80.7 | 78.2 |
| EDFNet w/ 5# feature & DF    | 80.3 | 78.4 | 78.3 | 75.2 | 75.7 |
| **EDFNet**                   | 87.5 | 86.2 | 87.9 | 87.9 | 86.3 |

**Table IX**

| Methods                      | 5000 | 2500 | 1000 | 500  | 250  |
|------------------------------|------|------|------|------|------|
|                              | Recall (%) |     |      |      |      |
| EDFNet w/ 1# feature         | 44.2 | 43.0 | 36.3 | 34.2 | 31.7 |
| EDFNet w/ 2# feature         | 44.9 | 44.2 | 38.4 | 35.7 | 33.4 |
| EDFNet w/ 3# feature         | 47.0 | 46.3 | 33.3 | 37.7 | 35.7 |
| EDFNet w/ 4# feature         | 45.1 | 45.6 | 40.0 | 36.2 | 32.7 |
| EDFNet w/ 5# feature         | 42.5 | 42.0 | 38.1 | 35.4 | 30.9 |
| **EDFNet**                   | 52.8 | 51.0 | 45.3 | 43.1 | 39.7 |
| EDFNet w/ 1# feature & DF    | 44.7 | 45.3 | 43.2 | 42.5 | 40.3 |
| EDFNet w/ 2# feature & DF    | 46.6 | 47.0 | 44.5 | 43.8 | 41.8 |
| EDFNet w/ 3# feature & DF    | 49.3 | 47.3 | 44.1 | 45.8 | 43.9 |
| EDFNet w/ 4# feature & DF    | 46.1 | 47.4 | 43.2 | 44.0 | 42.4 |
| EDFNet w/ 5# feature & DF    | 42.7 | 43.2 | 43.4 | 41.9 | 40.7 |
| **EDFNet**                   | 55.2 | 54.5 | 51.7 | 49.3 | 48.8 |

---
results. Nevertheless, the contrastive loss helps EDFNet to converge much faster.

V. DISCUSSION AND CONCLUSION

In this paper, we have proposed a novel task-specific point cloud feature descriptor learning method to tackle the 3D point cloud matching problem. We have presented EDFNet to learn point cloud descriptor, i.e., pruning the descriptor according to the specific task. Specifically, considering that the matching task aims at discovering point correspondences, we proposed to make the correspondence more matchable by exploiting the repetitive structure in two input point clouds. A CT-encoder is applied to extract different scale features. It captures the local geometry from the current point cloud by convolution and parallel searches the co-contextual information from paired point cloud by Transformer. Besides, a dynamic fusion module is designed to fuse these different scale features adaptively. It guides the final descriptor approach to the consistent and clean ones to dodge the disturbers to achieve a robust and discriminative feature descriptor. Extensive experiments on different datasets have validated the superiority of our proposed EDFNet. In the future, this task-specific learning method can be extended to other matching tasks, such as 2D-2D matching and 2D-3D matching.

However, the limitations of our method are also obvious. Since EDFNet is customized for the matching task, it is limited in other applications, such as classification, segmentation, etc. Besides, although we combine the descriptor learning with the matching task, our method still cannot handle the similar or symmetrical structures well. This is a stubborn problem in the point cloud matching community without effective solutions [4], [62], [73].

REFERENCES

[1] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, “Dynamic graph CNN for learning on point clouds,” ACM Trans. Graph., vol. 38, no. 5, pp. 1–12, 2019.
[2] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5099–5108.
[3] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, “KPCConv: Flexible and deformable convolution for point clouds,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 6410–6419.
[4] X. Bai, Z. Luo, L. Zhou, H. Fu, L. Quan, and C.-L. Tai, “D3Feat: Joint learning of dense detection and description of 3D local features,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 6358–6366.
[5] X. Wu, Y. Zhang, C. Fan, J. Hou, and S. Kwong, “Subjective quality database and objective study of compressed point clouds with 6DoF head-mounted display,” IEEE Trans. Circuits Syst. Video Technol., vol. 31, no. 12, pp. 4630–4644, Dec. 2021.
[6] Y. Qian, J. Hou, S. Kwong, and Y. He, “Deep magnification-flexible upsampling over 3D point clouds,” IEEE Trans. Image Process., vol. 30, pp. 8354–8367, 2021.
[7] H. Liu, H. Yuan, Q. Liu, J. Hou, H. Zeng, and S. Kwong, “A hybrid compression framework for color attributes of static 3D point clouds,” IEEE Trans. Circuits Syst. Video Technol., vol. 32, no. 3, pp. 1564–1577, Mar. 2022.
[8] J.-E. Deschaud, “IMLS-SLAM: Scan-to-model matching based on 3D data,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2018, pp. 2480–2485.
[9] S. Agarwal et al., “Building Rome in a day,” Commun. ACM, vol. 54, no. 10, pp. 105–112, Oct. 2011.
[10] J. M. Wong et al., “SegICP: Integrated deep semantic segmentation and pose estimation,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Sep. 2017, pp. 5784–5789.
[11] X. Huang, J. Zhang, Q. Wu, L. Fan, and C. Yuan, “A coarse-to-fine algorithm for matching and registration in 3D cross-source point clouds,” IEEE Trans. Circuits Syst. Video Technol., vol. 28, no. 10, pp. 2965–2977, Oct. 2018.
[12] L. Ding and C. Feng, “DeepMapping: Unsupervised map estimation from multiple point clouds,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 8650–8659.
[13] A. E. Johnson and M. Hebert, “Using spin images for efficient object recognition in cluttered 3D scenes,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 5, pp. 433–449, May 1999.
[14] A. Flint, A. Dick, and A. V. D. Hengel, “Thrift: Local 3D structure recognition,” in Proc. 9th Biennial Conf. Austral. Pattern Recognit. Soc. Digit. Image Comput. Techn. Appl. (DICTA), Dec. 2007, pp. 182–188.
[15] R. B. Rusu, N. Blidow, and M. Beetz, “Fast point feature histograms (FPFH) for 3D registration,” in Proc. IEEE Int. Conf. Robot. Autom., May 2009, pp. 3212–3217.
[16] F. Tombari, S. Salti, and L. D. Stefano, “Unique signatures of histograms for local surface description,” in Proc. Eur. Conf. Comput. Vis., 2010, pp. 356–369.
[17] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jul. 2017, pp. 652–660.
[18] Z. J. Yew and G. H. Lee, “3DFeat-Net: Weakly supervised local 3D features for point cloud registration,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 630–646.
[19] C. Choy, J. Park, and V. Koltun, “Fully convolutional geometric features,” in Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV), Oct. 2019, pp. 8958–8966.
[20] C. Esteves, C. Allen-Blanchette, A. Makaadia, and K. Daniilidis, “Learning SO(3) equivariant representations with spherical CNNs,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 54–70.
[21] Y. Rao, J. Lu, and J. Zhou, “Spherical fractal convolutional neural networks for point cloud recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 452–460.
[22] J. Fang, D. Zhou, X. Song, S. Jin, R. Yang, and L. Zhang, “RotPredictor: Unsupervised canonical viewpoint learning for point cloud classification,” in Proc. Int. Conf. 3D Vis. (DV), Nov. 2020, pp. 987–996.
[23] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser, “The perfect match: 3D point cloud matching with smoothed densities,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 5545–5554.
[24] H. Deng, T. Birdal, and S. Ilic, “PFNet: Global context aware local features for robust 3D point matching,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., Jun. 2018, pp. 195–205.
[25] H. Deng, T. Birdal, and S. Ilic, “PF-FFoldNet: Unsupervised learning of rotation invariant 3D local descriptors,” in Proc. Eur. Conf. Comput. Vis., 2018, pp. 620–638.
[26] A. Vaswani et al., “Attention is all you need,” in Proc. Adv. Neural Inf. Process. Syst., 2017, pp. 5998–6008.
[27] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, “Super- Glue: Learning feature matching with graph neural networks,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2020, pp. 4937–4946.
[28] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou, “LoFTR: Detector-free local feature matching with transformers,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2021, pp. 8922–8931.
