Keypoint Matching for Point Cloud Registration using Multiplex Dynamic Graph Attention Networks

Chenghao Shi[1], Xieyuanli Chen[2], Kaihong Huang[1], Junhao Xiao[1], Huimin Lu[1], and Cyrill Stachniss[2]

***Abstract*—The registration of point clouds is a key ingredient of LiDAR-based SLAM systems and mapping approaches. A challenging task in this context is finding the right data association between 3D points. This paper proposes a novel and flexible graph network architecture to tackle the keypoint matching problem in an end-to-end fashion. Each layer of our multiplex dynamic graph attention network (MDGAT) utilizes an attention mechanism to dynamically construct a multiplex graph and reasons about the contextual information based on the point cloud data. It enriches the feature representation by recovering local information and by aggregating information along with the connections. We also design a scan matcher called MDGAT-matcher, which treats the registration problem as an optimal transport problem and uses the predictions of MDGAT as the cost. It builds upon sparse keypoints extracted from pairs of LiDAR scans. Eventually, MDGAT-matcher finds high-quality correspondences and at the same time handles non-matching points appropriately. We train our matcher using a novel gap loss guiding the network to learn a discriminative cognition about matching and non-matching 3D points. We thoroughly test our approach on the KITTI odometry benchmark. The experiments presented in this paper suggest that our approach outperforms state-of-the-art matching approaches and achieves a substantial improvement.**

***Index Terms*—SLAM; Deep learning metod**

I. Introduction

POINT cloud registration is a fundamental problem in robotics as well as autonomous driving. Many robotic systems use it for mapping, LiDAR-based odometry, for simultaneous localization and mapping, also called SLAM [1]. Given two sets of points, a registration approach aims at finding the relative transformation that best aligns all 3D points in a common coordinate frame. Popular approaches to point cloud registration are the iterative closest point (ICP) algorithm [2] as well as several of its variants [3]–[5]. The key difficulty in registering point clouds is finding the correct data association, i.e., determining which point in point cloud 1 corresponds to which point in cloud 2. Once the data association is known, computing the transformation is straight forward.

A robust approach to finding high-quality correspondences between two point clouds is essential in offline 3D registration and incremental scan matching. While the ICP-alike algorithms are powerful and easy to implement, they often rely on nearest neighbor (NN) assignments or similar distance-based heuristics to find correspondences. This often yields a limited basin of convergence, and as a result of that, the initial transformation between the scans matters enormously. The better the initial guess, the more likely the data association will lead to the correct alignment, and the approach may fail otherwise. In case point cloud-based keypoint descriptors are provided, they can be used for finding likely correspondences even under suboptimal initial configurations. Descriptor-based matching often increases robustness w.r.t. the initial guess. This approach first aims to identify distinctive points in both point clouds using a descriptor and then compute further matches considering an initial relative transformation given the distinct points. To achieve this, feature descriptors for 3D point clouds and matching strategies can be defined manually or learned in an end-to-end fashion.

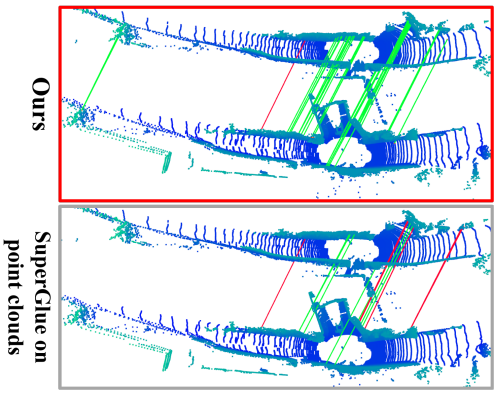

Fig. 1: Keypoint matching results of our method and SuperGlue. Our method better distinguishes nearby keypoints and finds more correct matches (green lines) and less wrong ones (red lines).

The recent work SuperGlue [6] is a seminal paper for learning-based matching of visual features. Given two sets of local descriptors with keypoints, it finds for correspondences. It utilizes a multiplex graph attention network to predict a soft assignment between two sets of features. It selects the matches using a predefined threshold. Inspired by SuperGlue, Simon *et al*. propose StickyPillars [7]. It uses the same graph neural network on point clouds and achieves reasonable results for 3D registration. Although beginning a great step forward, the current SuperGlue architecture offers space for improvements. First, it employs a fully-connected graph construction, which limits the distinctiveness of the learned descriptor. Second, it needs a well-tuned, predefined threshold to determine the final

Manuscript received: February 24, 2021; Revised April 26, 2021; Accepted June, 28, 2021.

This paper was recommended for publication by Editor Editor Javier Civera upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by the National Natural Science Foundation of China [61773393, U1913202, U1813205].

[1] Chenghao Shi, Kaihong Huang, Junhao Xiao and Huimin Lu are with the Robotics Research Center, College of Intelligence Science and Technology, National University of Defense Technology, Changsha, China. [2] Xieyuanli Chen and Cyrill Stachniss are with the University of Bonn, Germany.

Digital Object Identifier (DOI): 10.1109/LRA.2021.3097275.

matches, which is less elegant and can lead to a suboptimal generalization ability. We aim at tackling these challenges in our work.

The main contribution of this paper is a novel approach to feature matching in 3D LiDAR scans using a flexible graph network architecture and allows for end-to-end training. Different from existing work, we propose a novel layer to construct the graph network dynamically. This captures the local information better and improves the distinctiveness of the learned descriptors. Moreover, we propose a gap loss to guide the network to better distinguish between the positive and negative pairs of descriptors. Our method does not require a hand-designed threshold to determine the final matches and achieves feature matching in an end-to-end fashion. In sum, we make two key claims for our approach. It is able to (i) find feature matches end-to-end, (ii) and better distinguishes the features and outperforms the state-of-the-art methods in feature matching. Our code is available at:

https://github.com/chenghao-shi/MDGAT-matcher.

II. Related Work

Point cloud registration refers to finding the spatial transformation that aligns two point clouds. The standard approach for point cloud registration is the iterative closest point (ICP) algorithm [2] and its numerous variants such as [3]–[5]. A large number of existing odometry and SLAM systems rely on ICP-alike algorithms [8]–[10]. The most challenging part in ICP is finding the correct data association and the result of ICP is influenced by the initial guess and it can easily get stuck in a local minimum. To avoid making crisp data association, there are works using soft correspondences [11], robust estimators [12], kernel density estimation [13], or probability density estimation [14] to achieve improved point cloud registration.

Using deep learning for point cloud registration is a comparably novel area. Chen *et al*. [15] exploit a neural network to estimate the similarity and the yaw angle offset between pairs of scans and use it as the initial guess for ICP. Deep closest point proposed by Wang *et al*. [16] and DeepICP by Lu *et al*. [17] are learning-based end-to-end point cloud registration methods. Different to classical ICP that iteratively updates the geometric transform and point correspondences by nearest neighbor searching, they establish the point correspondences based on the similarity of feature space in an end-to-end fashion. PointNetLK proposed by Aoki *et al*. [18] and PCRNet by Sarode *et al*. [19] compute the global descriptors of the point sets and estimate the transform by minimizing the distance of the global descriptors. Recent work called RPM-Net proposed by Yew *et al*. [20] utilizes a different pipeline from ICP. It uses the iterative framework of robust point matching (RPM) [14] with deep learning network-based features, which is less sensitive to initialization and more robust for rigid point cloud registration. To handle registration of partially overlapped point clouds, most recently Xu *et al*. [21] and Sarode *et al*. [22] exploit neural networks to extract both the global feature of a point cloud and point-wise features to predict inlier mask and estimate the transformation in an end-to-end fashion. The common problem with raw point-based methods is that they require the given pair of point sets to have significant overlap or reasonable initial associations.

Feature-based methods form another branch of point cloud registration methods, which are generally composed by keypoints detection, keypoint matching, and the computation of the geometric transformation. For each step, there exist a large number of optimizations. For example, there are methods focusing on keypoint detection, such as hand-crafted keypoints ISS [23], curvature [8], and recently learning-based keypoints 3DFeat-Net [24] and USIP [25]. Feature matching approaches typically exploit carefully designed descriptors. For example, the fast point feature histogram (FPFH) proposed by Rusu *et al*. [26] is a representative, hand-crafted 3D feature descriptor. It provides accurate feature matching with reasonable computational efficiency. Recent works also apply deep neural networks to learn the descriptors and achieve remarkable performance. For example, Zeng *et al*. [27] propose 3DMatch, which is one of the earliest learning-based works that apply deep learning on a voxelized point cloud and computes the descriptors. The work proposed by Gojcic *et al*. [28] is similar to 3DMatch but uses a more efficient data representation called smoothed density value voxelization which reduces the input sparsity and is amenable to fully convolutional layers. Qi *et al*. [29] propose PointNet, which is the first work that directly applies deep learning to the raw point cloud. The main idea behind PointNet is the max pooling operation that aggregates point-wise features into a global feature while being invariant to input permutation. Inspired by PointNet, Deng *et al*. [30] propose PPFNet, which utilizes the PointNet to describe the local feature and further argument the feature using a PointNet-like framework.

Unlike the standard pipeline, graph matching is not limited to rigid transform and is used to model more complex relationships. Recently, Zanfir *et al*. [31] apply deep learning to learn the cost and firstly achieve end-to-end deep graph matching. Based on that, Wang *et al*. [16] propose a graph neural network incorporating node embedding to learn a more flexible cost. Solving a graph matching problem is typically formulated as solving a quadratic assignment problem, which is known to be NP-hard. Thus graph matching problems are usually solved approximately. By simplifying the problem to a linear assignment problem, Hungarian algorithm proposed by Kuhn [32] can be an efficient solution. Sinkhorn algorithm [33] is shown to be a differentiable version of the Hungarian algorithm and is widely used in neural network [6] to enable the network to solve the problem end-to-end.

The most similar works to this paper are SuperGlue [6] and StickyPillars [7]. Sarlin *et al*. propose SuperGlue to use a graph neural network to embed structure information into the descriptor for a more powerful one, which shows promising results on image matching problem. Most recently, Simon *et al*. propose StickyPillars, which firstly applies such a pipeline on 3D feature matching. Different to them, our method uses a novel multiplex dynamic graph attention network with the proposed gap loss to realize 3D point cloud feature matching.

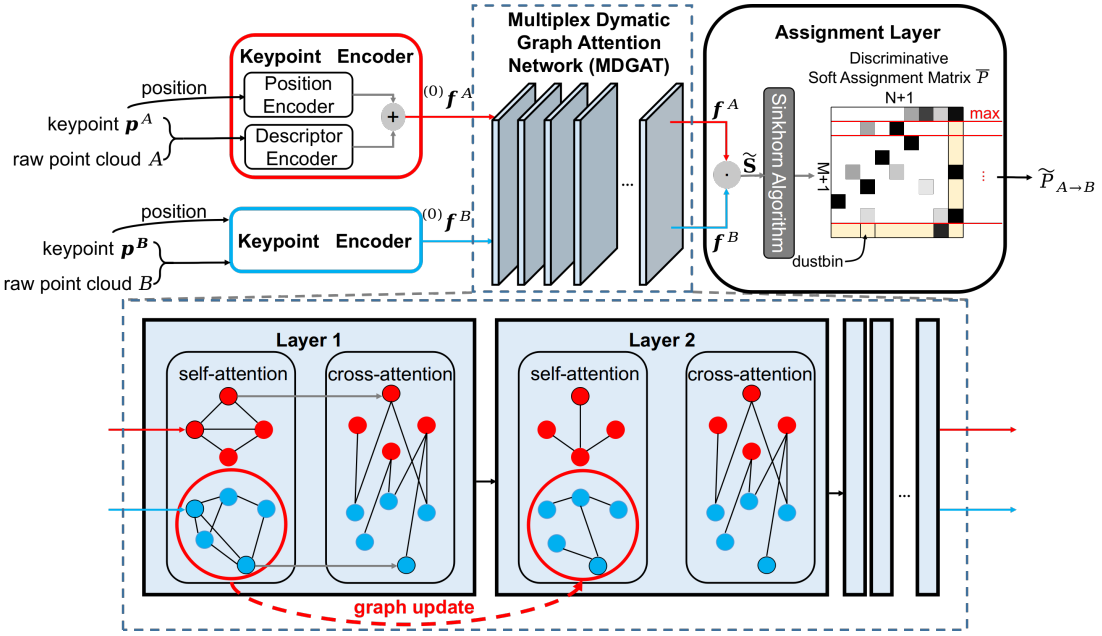

Fig. 2: **Our inference pipeline.** Our network consists of three main parts: keypoint encoder, multiplex dynamic graph attention network, and assignment layer. The keypoint encoder takes the point cloud and keypoint positions $\boldsymbol{p}$ as input and embeds the local keypoint appearance (descriptor encoder) and the spatial clues (position encoder) into a new feature. The multiplex dynamic graph attention network further enhances the feature by aggregating the contextual information from both, keypoint sets based on a dynamically constructed graph. Based on the enhanced descriptions, the assignment layer creates a score matrix, and explicitly incorporates the keypoint dustbin. The soft assignment matrix is then predicted using Sinkhorn algorithm. Performing the max operator to each row except the dustbin, we finally determine the matches.

III. Background – Point Matching and SuperGlue

We denote two point clouds as $\boldsymbol{A}$ and $\boldsymbol{B}$, and the keypoints extracted from them are $\boldsymbol{p}^{\boldsymbol{A}}$ and $\boldsymbol{p}^{\boldsymbol{B}}$, where $\boldsymbol{p}^{\boldsymbol{A}} \subset \boldsymbol{A}$ and $\boldsymbol{p}^{\boldsymbol{B}} \subset \boldsymbol{B}$. Each keypoint $\boldsymbol{p}_i$ has 3D coordinates, $(x, y, z)_i$, and the numbers of the keypoints in these two sets are M and N, respectively. Several learning-based methods [17], [34] assume that all the keypoints have a corresponding partner, and describe the correspondences with a partial assignment matrix $\boldsymbol{P} \in \{0,1\}^{M\times N}$:

$$\boldsymbol{P}\,\mathbf{1}_N = \mathbf{1}_M^{\mathsf{T}}, \tag{1}$$

$$\boldsymbol{P}^{\mathsf{T}}\mathbf{1}_M = \mathbf{1}_N^{\mathsf{T}}. \tag{2}$$

Given the correspondences, the network tries to generate distinctive descriptors by learning a function f, which minimizes the distance between matched keypoint pairs in feature space or maximizes the similarity between them as:

$$f^* = \underset{f}{\operatorname{argmax}} \frac{2}{M} \sum_{i=1}^{M}\sum_{j=1}^{N} P_{ij}\mathbf{S}_{ij}, \tag{3}$$

where the match score matrix $\mathbf{S} = f(\boldsymbol{p}^{\boldsymbol{A}}) \cdot f(\boldsymbol{p}^{\boldsymbol{B}})^{\mathsf{T}}$, and $f(\boldsymbol{p}^{\boldsymbol{A}})$ and $f(\boldsymbol{p}^{\boldsymbol{B}})$ are the descriptors learned by the networks as for the set of keypoints $\boldsymbol{p}^{\boldsymbol{A}}$ and $\boldsymbol{p}^{\boldsymbol{B}}$.

For a generalized formulation and handling non-matched keypoints, we create a so-called "dustbin" $\boldsymbol{b}$ for the non-matched points (we use the name dustbin similar to [6]). Keypoints of $\boldsymbol{A}$ and $\boldsymbol{B}$ can now be written as $\tilde{\boldsymbol{p}}^{\boldsymbol{A}} = [(\boldsymbol{p}^{\boldsymbol{A}})^{\mathsf{T}}, \boldsymbol{b}^A]^{\mathsf{T}}$ and $\tilde{\boldsymbol{p}}^{\boldsymbol{B}} = [(\boldsymbol{p}^{\boldsymbol{B}})^{\mathsf{T}}, \boldsymbol{b}^B]^{\mathsf{T}}$. Consequently, for each keypoint in the point cloud, it is either matched to a keypoint in the other one or to the dustbin of the other one. Assuming there are K matched keypoint pairs, the new partial assignment matrix $\tilde{\boldsymbol{P}} \in \{0,1\}^{(M+1)\times(N+1)}$ is constrained by:

$$\tilde{\boldsymbol{P}}\,\mathbf{1}_{N+1} = [\mathbf{1}_M^{\mathsf{T}},\ N{-}K], \tag{4}$$

$$\tilde{\boldsymbol{P}}^{\mathsf{T}}\mathbf{1}_{M+1} = [\mathbf{1}_N^{\mathsf{T}},\ M{-}K], \tag{5}$$

where $\mathbf{1}_N$ is the all-ones vector of length N, and $N{-}K$ and $M{-}K$ are the numbers of unmatched keypoints.

The match score between any keypoint and the dustbin is set as $z \in \mathbb{R}$, which can be learned by a network. Let $\tilde{\mathbf{S}}$ be the match score matrix of $\tilde{\boldsymbol{p}}^{\boldsymbol{A}}$ and $\tilde{\boldsymbol{p}}^{\boldsymbol{B}}$, where the last row and column are filled with z. We then extend Eq. (3) to:

$$f^* = \underset{f}{\operatorname{argmax}} \frac{2}{M} \sum_{i=1}^{M+1}\sum_{j=1}^{N+1} \tilde{P}_{ij}\tilde{\mathbf{S}}_{ij}. \tag{6}$$

Eq. (6) aims at maximizing the total match scores for all the matched and non-matched keypoint pairs. The keypoint matcher network aims a determining (i) given the ground true matches $\tilde{P}$ during training, how to learn the descriptor function f and (ii) after learning the descriptor function f, during the inferring, how to estimate the final partial assignment $\tilde{P}$ given the estimated $\tilde{S}$ generated from the function f, i.e.

$$\tilde{\boldsymbol{P}}^* = \underset{\tilde{\boldsymbol{P}}}{\operatorname{argmax}} \frac{2}{M} \sum_{i=1}^{M+1}\sum_{j=1}^{N+1} \tilde{P}_{ij}\tilde{\mathbf{S}}_{ij}. \tag{7}$$

Our approach is inspired by SuperGlue [6], which uses a multiplex graph neural network f to generate descriptors for visual features extracted from images. It takes two keypoint sets extracted from two input images $f(\boldsymbol{p}^{\boldsymbol{A}}, \boldsymbol{p}^{\boldsymbol{B}})$ and uses self-

(intra-image) and cross- (inter-image) attention to reason about contextual information in both images. By formulating an optimization function, SuperGlue introduces an optimal transport layer to solve the function and predict a soft assignment matrix $\overline{\boldsymbol{P}} \in [0,1]^{(M+1)\times(N+1)}$. SuperGlue also uses dustbins to handle the unmatched keypoints but with an additional predefined threshold to determine the final assignment matrix $\tilde{\boldsymbol{P}}$.

IV. Our Approach

The overview of our approach is illustrated in Fig. 2 and it consists of three main parts: the *keypoint encoder* introduced in Sec. IV-A, the *multiplex dynamic graph attention network* described in Sec. IV-B, and the *assignment layer*, see Sec. IV-C. We also propose a novel gap loss in Sec. IV-D, which guides our network to better distinguish the feature matches.

A. Keypoint Encoder

Instead of directly using hand-crafted local keypoint descriptors as the input, we use a keypoint encoder to compute descriptors for keypoints. Inspired by SuperGlue [6], we combine existing local keypoint descriptors with the keypoint position information using a neural network. The goal is to provide a learning-supported descriptor, which can better exploit the vicinity information contained in the original hand-crafted descriptor and thus provides a better signature. We design our encoder to mostly keep the local information while leaving the harder part of the global representation of the whole scan to the subsequent dynamic graph network. To this end, we build our encoder with two parts, a descriptor encoder and a positional encoder, as explained below.

The descriptor encoder is designed to embed the local vicinity information into a single, high-dimensional vector $\boldsymbol{d}$. Examples of popular keypoint/descriptors are USIP [25] and FPFH [26]. USIP provides good detection repeatability on the sparse point cloud, while FPFH is an effective local 3D descriptor for 3D registration. In this work, we combine USIP to detect the keypoints and the FPFH descriptors as the starting point for building our descriptor. Each keypoint and FPFH descriptor is encoded into a high-dimensional vector using a multi-layer perceptron (MLP):

$$\boldsymbol{d} = \mathrm{MLP}_{\mathrm{desc}}(\mathrm{FPFH}(\boldsymbol{p}_i, \boldsymbol{C})), \tag{8}$$

where $\boldsymbol{C}$ represents the corresponding raw point cloud.

Following SuperGlue [6], we combine the descriptor $\boldsymbol{d}$ with a position encoder to embed spatial clues into the descriptor by:

$${}^{(0)}\boldsymbol{f}_i = \boldsymbol{d} + \mathrm{MLP}_{\mathrm{pos}}(\boldsymbol{p}_i). \tag{9}$$

Thus, the input of our method are the raw point clouds together with the USIP [25] keypoints and FPFH descriptors [26], and we turn them into a new descriptor. Note, however, our method does not rely on a specific keypoint detector and descriptor such as USIP and FPFH, which can be seen in the ablation study in Sec. VI-C. As we show later in this paper, such combinations are also beneficial for tackling our 3D point matching problem.

B. Multiplex Dynamic Graph Attention Network

After the keypoint encoder, we add a multiplex dynamic graph attention network, short MDGAT, to strengthen the descriptor for matching. Different to a traditional graph neural network, which uses a fixed graph architecture, we propose a novel dynamic graph that updates the architecture dynamically. Furthermore, an attentional aggregation is performed to enable the node to effectively receive the information from neighboring nodes and towards a more global approach.

Multiplex graph. In MDGAT, we firstly construct a multiplex graph [35] taking all the embeddings from the keypoint encoder as nodes. The graph contains two types of edges: self-edges that connect keypoints within the same point cloud and cross-edges that connect a keypoint to the keypoints in the other point cloud. Self-edges are designed to help the network learn the representation of the keypoint based on its vicinity information while cross-edges are designed to help the network examine each validation keypoint in the other point set.

Dynamic graph update. In each iteration, we construct the specific graph structure based on an attention mechanism [36]. For a query node ${}^{(l)}\boldsymbol{f}_i^{\boldsymbol{Q}}$ in point cloud $\boldsymbol{Q}$ (indicating "query") at layer l and all the source nodes in point cloud $\boldsymbol{S}$ (indicating "source") at layer l, the network computes the query $\boldsymbol{q}_i$, key $\boldsymbol{k}_j$, and value $\boldsymbol{v}_j$, which are used for later graph updating and attentional aggregation, by a linear projection:

$$\begin{aligned} \boldsymbol{q}_i &= \boldsymbol{W}_1\, {}^{(l)}\boldsymbol{f}_i^{\boldsymbol{Q}} + \boldsymbol{b}_1, \\ \begin{bmatrix} \boldsymbol{k}_j \\ \boldsymbol{v}_j \end{bmatrix} &= \begin{bmatrix} \boldsymbol{W}_2 \\ \boldsymbol{W}_3 \end{bmatrix} {}^{(l)}\boldsymbol{f}_j^{\boldsymbol{S}} + \begin{bmatrix} \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \end{bmatrix}, \end{aligned} \tag{10}$$

where $\{\boldsymbol{Q}, \boldsymbol{S}\} \in \{\boldsymbol{A}, \boldsymbol{B}\}^2$. If $\boldsymbol{Q} = \boldsymbol{S}$, we refer to Eq. (10) as self-attention, and as cross-attention if $\boldsymbol{Q} \neq \boldsymbol{S}$. The projection parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ are shared for all nodes at each layer l. After that, we compute an attentional weight for the query node with each source node: $\alpha_{ij} = \mathrm{softmax}_j(\boldsymbol{q}_i^{\mathsf{T}}\boldsymbol{k}_j)$. This weight represents how much attention the query node ${}^{(l)}\boldsymbol{f}_i^{\boldsymbol{Q}}$ will pay to this source node ${}^{(l)}\boldsymbol{f}_j^{\boldsymbol{S}}$.

SuperGlue uses a fully-connected graph, which exploits all the connections between all the nodes. However, the connections between the query node and the dissimilar neighbor source nodes may hurt the performance, as recently pointed out by Xie *et al*. [37] (and can also confirm this observation from our experiments). Thus, we propose a dynamic graph that finds the top k edges for each query node based on attentional weight α_{ij} and only uses these top k edges and the corresponding nodes to construct the graph. We apply this to all query nodes and construct a new multiplex graph in each iteration according to the attentional weights.

Instead of using a fixed number of k to choose the edges for the nodes in all the layers, we use a decay strategy, which gradually decreases the connections when the layers go deeper. It is natural to use such a decay strategy as it can be seen as firstly observing the overall scene and then gradually focusing on a specific area of interest. We report the chosen parameters in Sec. VI-A and verify the effectiveness of this design in

Sec. VI-C, showing that the proposed dynamic graph performs better than the fully-connected static graph.

Attentional aggregation. Attentional aggregation [38] is an operation that dynamically assigns weights to the neighbor connections of a node. It is based on an attention mechanism [36] enabling a node to receive the information from neighboring nodes towards a global understanding. After the above-proposed dynamic updating, this attentional aggregation is performed on each node of each newly generated graph and is done via message passing. More details about aggregation could be found in the work by Sarlin [6]. Once the aggregation is done for all the layers, the final descriptors of the nodes are then represented by an additional liner projection layer:

$$\boldsymbol{f}_i = \boldsymbol{W}^{(L)}\boldsymbol{f}_i + \boldsymbol{b}. \tag{11}$$

C. Assignment Layer

The two above-described layers, keypoint encoder, and multiplex dynamic graph attention, together form the feature descriptor $f(\cdot)$. As described in Sec. III, to design the loss function for training our network, we need to solve Eq. (7). The solution can be approximated using the Sinkhorn algorithm [39], which is commonly used for graph matching and is fully differentiable. It iteratively performs normalization along rows and columns. At the t iteration, the score matrix is updated by:

$${}^{(t)}\tilde{\mathbf{S}}'_{ij} = {}^{(t)}\tilde{\mathbf{S}}_{ij} - \log\sum_j e^{{}^{(t)}\tilde{\mathbf{S}}_{ij}}, \tag{12}$$

$${}^{(t+1)}\tilde{\mathbf{S}}_{ij} = {}^{(t)}\tilde{\mathbf{S}}'_{ij} - \log\sum_i e^{{}^{(t)}\tilde{\mathbf{S}}'_{ij}}. \tag{13}$$

After T iterations, we use the solution as the soft assignment matrix: $\overline{\boldsymbol{P}} = {}^{(T)}\tilde{\mathbf{S}}$.

Sarlin *et al*. [6] drop the dustbin and recover the assignment by comparing the soft assignment score $\overline{\boldsymbol{P}}_{1:M,1:N}$ with a hand-tuned threshold. In contrast, we do not manually define a threshold and drop the dustbins. Instead, we directly resolve the $\tilde{\boldsymbol{P}}$ by finding the best match on raw $\overline{\boldsymbol{P}}$.

$$\tilde{\boldsymbol{P}}_{\boldsymbol{A}\to\boldsymbol{B}} = \max_{\text{row}}(\overline{\boldsymbol{P}}), \tag{14}$$

$$\tilde{\boldsymbol{P}}_{\boldsymbol{B}\to\boldsymbol{A}} = \max_{\text{row}}(\overline{\boldsymbol{P}}^{\mathsf{T}}), \tag{15}$$

$$\tilde{\boldsymbol{P}} = \tilde{\boldsymbol{P}}_{\boldsymbol{A}\to\boldsymbol{B}} \odot \tilde{\boldsymbol{P}}^{\mathsf{T}}_{\boldsymbol{B}\to\boldsymbol{A}}, \tag{16}$$

where $\max_{\text{row}}$ represents an operation that sets the maximum value along the rows to 1 and the others to 0, $\tilde{\boldsymbol{P}}_{\boldsymbol{A}\to\boldsymbol{B}}$ represents the predicted matrix assigns the keypoints in $\boldsymbol{A}$ to the keypoints in $\boldsymbol{B}$ or dustbins, and $\odot$ represents the Hadamard product. We use Eq. (16) to perform a mutual check to ensure the match is the best for both keypoints. This does not work well for the original SuperGlue as it requires a discriminative assignment matrix, as can be seen in Sec. VI-B. To this end, we introduce a gap loss that is described in the next section, to guide the network learning such a discriminative assignment matrix.

D. Gap Loss

Our goal is to enlarge the relative difference(i.e., the gap) of the soft assignment values between the true matches and the wrong matches, and not just to maximize the absolute values of the truth matches. In this work, we propose a novel gap loss based on a triplet loss [40]. A triplet $\mathcal{P}_i^{\boldsymbol{A}}$ consists of an anchor keypoint in $\boldsymbol{A}$, a positive correspondence as well as a negative correspondence in $\boldsymbol{B}$, respectively indexed by i, p, n. In our case, the anchor keypoint can be an arbitrary keypoint. A positive match p refers to the truth match for the anchor keypoint, while a negative match n can be any non-matching keypoint for the anchor keypoint.

For training, we can generate such required ground truth correspondences easily. Using ground truth poses (e.g., available in the KITTI dataset), we generate the ground truth correspondences $\boldsymbol{M} \in \{0,1\}^{(M+1)\times(N+1)}$ by projecting the keypoints to the other point cloud. The proposed gap loss is then calculated as:

$$\begin{aligned}\text{Loss}_{\text{gap}} = &\sum_{i=1}^{M}\log(\sum_{n=1}^{N+1}[(-\log r_i + \log\overline{\boldsymbol{P}}_{in} + \eta)_+ - \eta]) \\ &+ \sum_{j=1}^{N}\log(\sum_{n=1}^{M+1}[(-\log c_j + \log\overline{\boldsymbol{P}}_{nj} + \eta)_+ - \eta]),\end{aligned} \tag{17}$$

where $(z)_+ = max(z, 0)$, $r_i = \sum_{n=1}^{N+1}\overline{\boldsymbol{P}}_{in}\boldsymbol{M}_{in}$ refers to the true match assignment value for keypoint $\boldsymbol{p}_i^{\boldsymbol{A}}$, and $c_j = \sum_{n=1}^{M+1}\overline{\boldsymbol{P}}_{nj}\boldsymbol{M}_{nj}$ refers to the true match assignment value for keypoint $\boldsymbol{p}_j^{\boldsymbol{B}}$. Different from triplet loss utilizing only one negative match for each anchor keypoint, the proposed gap loss aims to expand the margin between the positive match with all the other negative matches.

V. Implementation Details

Some parameters related to the architecture are relevant to reproduce our results, and we provide them together with details about the training in this section.

Parameters. We initialize the size of the input keypoints as $N' = M' = 512$ on both point clouds. Based on the uncertainty calculated by USIP for each keypoint, we downsample the keypoints to a size of $N' = M' = 256$. The descriptor encoder uses a nearest neighbor search range of $d = 1\,\text{m}$. The output of the keypoint encoder and all intermediate representations of the MDGAT have the same dimension $D = 128$. For the architecture of MDGAT, we use $L = 9$ layers with 4 attention heads, and each performs self-attention followed by cross-attention. MDGAT reconstructs the graph based on the attention weights and updates dynamically. We use a decay strategy to choose the number of k nearest neighbor when the network goes deeper to force MDGAT to gradually focus more on the distinctive area. More specifically, the first 5 layers are a fully-connected graph, and the number of k of the last 4 layers are $k_6^{\text{self}} = k_7^{\text{self}} = 128$, $k_8^{\text{self}} = k_9^{\text{self}} = 64$, where k_i^{self} refers to using k nearest neighbor for the self connection at i layer. For the cross-attention, we keep it as fully connected, where $k_{1-9}^{\text{cross}} = 256$. We set Sinkhorn iteration to $T = 100$. The

network contains 3 million parameters and takes on average 127 ms per forward pass on a GTX 1660 Ti GPU.

Training. We train our network on the KITTI dataset. Instead of selecting pairs based on fixed frame interval for training as done by Simon *et al*. [7], we randomly select, given one point cloud, another point cloud within a 10 m range in the same sequence to form a pair, which can increase the robustness of our approach. By doing this, we also get more samples, which are 16,820 for training and 1,200 for test pairs. Ground truth correspondences are found by projecting keypoints from a point cloud to the other using the provided ground truth pose. The ground truth matches are either keypoint pairs with residuals smaller than a threshold ϵ or keypoint-dustbin pairs. We set $\epsilon = 0.5$ m, the same as that used in [25] for a fair comparison. For the training, we use the Adam optimizer [41] with a constant learning rate of 10^{-4} and a batch size of 64.

VI. Experimental Evaluation

Our experiments are designed to show the capabilities of our method. They furthermore support our two key claims. These two claims are that our approach is able to (i) find feature matches end-to-end (without requiring a predefined threshold for determining the matches), and (ii) can better distinguish the features and outperforms the state-of-the-art methods in feature matching on the KITTI dataset.

A. Dataset and Evaluation Metrics

We evaluate the proposed matcher on the KITTI odometry dataset [42]. The KITTI dataset contains 3D point cloud data captured by a Velodyne HDL64 LiDAR with the ground truth poses provided by GPS/INS system. Since the ground truth poses are only available for sequences 00-10, we, therefore, use sequences 00-09 for training, except 08 for validation and sequence 10 for testing. To use USIP keypoint detector, we follow the original work [25] to estimate the normal for each point. To speed up the training and increase the robustness of our approach, we randomly downsample the raw point cloud into 16,384 points per frame.

The matching performance is evaluated using the matching precision (P), accuracy (A), recall (R), and also F1-score. Based on predicted matches, we directly estimate the transformation using singular value decomposition in the spirit of the ICP algorithm. For evaluating the registration performance, we follow the metrics used by Li *et al*. [25], which are registering failure rate and inlier ratio. The registration failure rate is calculated by comparing the estimated transform matrix $\boldsymbol{T}$ to the ground truth transform matrix $\boldsymbol{T}_g$. The failure case means relative translational error (RTE) > 2 m, or relative rotation error (RRE) $> 5^\circ$. The inlier ratio is calculated as inliers divided by the total keypoints number.

B. Matching and Registration Performance

We first evaluate the matching performance. We compare our matcher with SuperGlue using multiple threshold settings as well as SuperGlue combined with our assignment algorithm. As shown in Tab. I, our matcher achieves the best matching performance under all criteria. Our matcher achieves a substantial improvement of about 17% on recall (R), 9% improvement on both, precision (P) and accuracy (A), and 22% in F1-score compared to the state-of-the-art SuperGlue, which can also be seen in ROC curve in Fig. 3.

TABLE I: Matching performance.

Local keypoints	Matcher	P	A	R	F1
USIP	Our Desc.+SuperGlue$_{0.1}$	54.3	76.7	57.7	0.559
	Our Desc.+SuperGlue$_{0.2}$	57.6	75.8	49.2	0.531
	Our Desc.+SuperGlue$_{0.3}$	62.2	72.7	39.4	0.482
	Our Desc.+SuperGlue#	57.7	72.2	35.4	0.439
	Our Matcher	**66.9**	**84.1**	**66.2**	**0.665**

Subscript, e.g., $_{0.1}$ is the match threshold used in this estimation.
SuperGlue# uses the threshold-free assignment algorithm described in Sec. IV-C.

TABLE II: Registration performance.

Local keypoints	Matcher	Failure rate	Inlier ratio
USIP	FPFH+RANSAC*	8.37	18.77
	SHOT+RANSAC*	5.40	18.21
	3DFeatNet+RANSAC*	1.55	22.48
	USIP+RANSAC*	1.41	32.20
	USIP+MaskNet [22]+DCP [34]	4.01	35.37
	Our Desc.+SuperGlue#	5.51	23.05
	Our Desc.+SuperGlue	**0.58**	36.19
	Our Matcher	0.67	**42.23**

Superscript * means the results are reported by Li *et al*. [25].

We then evaluate the registration performance. We compare our method against several baselines: the hand-crafted descriptors FPFH [26] and SHOT [43] together with RANSAC, and the learning-based descriptor 3DFeatNet [24] and USIP [25] with RANSAC, as well as SuperGlue [6]. As the pre-trained model of USIP is not available, we directly compare our results to the results provided in the original paper [25], which uses the same setup as we used in this paper. Moreover, we provide the results of the state-of-the-art feature matcher SuperGlue with an extension from our side so that it works with 3D point cloud instead of images, and SuperGlue is retrained with our descriptor encoder on KITTI. We also compare our method with the combination of MaskNet [22] with DCP [34] which also includes filtering, contextual aggregation, and matching. If there is no special statement, the results of SuperGlue are generated using the best configuration with matching threshold of 0.2. As shown in Tab. II, our network achieves the best performance among all the baselines with the highest inlier ratio while having a low registration failure rate, on par with SuperGlue. Note that, SuperGlue has a big performance decline using the threshold-free algorithm both in matching and registration. The reason is that the threshold-free assignment algorithm requires a discriminative assignment matrix that clearly separates the matching and non-matching pairs. The proposed MDGAT trained with our gap loss can provide it, but SuperGlue cannot.

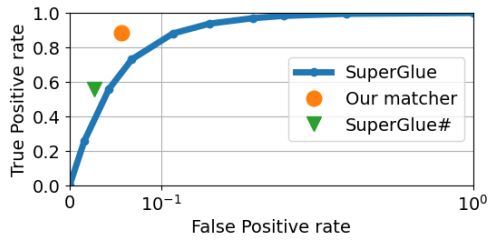

Fig. 3: ROC curve. Our matcher outperforms the state-of-the-art SuperGlue as well as an optimized SuperGlue version called SuperGlue#, which uses our assignment algorithm described in Sec. IV-C. Because no matching-threshold is required in our assignment algorithm, there is only one marker for our matcher and SuperGlue# in the plot.

TABLE III: Ablation of descriptor encoder.

Local keypoints	Descriptor encoder	Matcher	P	A	R	F1
USIP	Our Desc.	SuperGlue	57.6	75.8	49.2	0.531
	PointNet	MDGAT	59.4	80.3	62.3	0.608
	Global Desc.		63.9	82.6	63.2	0.635
	Our Desc.		**66.9**	**84.1**	**66.2**	**0.665**

C. Ablation Study

We conduct ablation studies on our three key contributions: the keypoint encoder, the MDGAT, and the gap loss.

Ablation of keypoints encoder. In this ablation study, we compared three different designs of descriptor encoders. The first descriptor encoder is a vanilla version of PointNet [29], working directly on independent points sampled by the nearest neighbor search. The second descriptor encoder is designed inspired by PPFNet [30] and PointNet [29]. It takes the output of our descriptor encoder ${}^{(0)}\boldsymbol{f}$ and performs a symmetric function, max pooling, to extract global information, $\max({}^{(0)}\boldsymbol{f})$. We then concatenate the global vector with each local descriptor $[{}^{(0)}\boldsymbol{f}_i \| \max({}^{(0)}\boldsymbol{f})]$. An MLP is used to embed it into a global-aware descriptor: ${}^{(0)}\boldsymbol{f}' = \text{MLP}[{}^{(0)}\boldsymbol{f} \| \max({}^{(0)}\boldsymbol{f})]$, fitting the input dimension $D = 128$ of MDGAT. As the descriptor being able to aware of global information, we refer to the first descriptor encoder as *Global Desc*. The third is the one we used in the proposed method (Sec. IV-A), named as *Our Desc*.

All the encoders use the same nearest neighbor search range $r = 1$ m. We retrain our network with the above encoders. This experiment shown in Tab. III demonstrates that our descriptor with the simplest structure achieves the best performance. The possible reason is that the self-attention can be treated as global information aggregating, which is a powerful and flexible mechanism and is further enhanced by dynamic graph updating used in MDGAT. Thus, concatenating global information as *Global Desc.* at the beginning during encoding can be redundant and even cause the decline of discrimination. It is the same to PointNet, which describes the overall appearance using max pooling, and loses the local details.

Ablation of MDGAT. In this ablation study, we evaluate several variants of MDGAT. We refer the one using the first 7 layers $k^{self}_{1-7} = 128, k^{cross}_{1-7} = 256$, the last 2 layers $k^{self}_{8,9} = 64, k^{cross}_{8,9} = 256$ as *MDGAT variant 1* and the one using $6, 7$ layers $k^{self}_{6,7} = k^{cross}_{6,7} = 128$, the last 2 layers $k^{self}_{8,9} = k^{cross}_{8,9} = 64$ as *MDGAT variant 2*.

TABLE IV: Ablation of MDGAT.

Local keypoints	Matcher	P	A	R	F1
USIP	Our Desc.+SuperGlue	57.6	75.8	49.2	0.531
	MDGAT variant 1	62.1	81.6	61.8	0.619
	MDGAT variant 2	64.9	83.3	65.9	0.653
	MDGAT	**66.9**	**84.1**	**66.2**	**0.665**

MDGAT variant 1 uses $k^{self}_{6,7} = k^{cross}_{6,7} = 128, k^{self}_{8,9} = k^{cross}_{8,9} = 64$.
MDGAT variant 2 uses $k^{self}_{1-7} = 128, k^{self}_{8,9} = 64, k^{cross}_{1-9} = 256$.
MDGAT uses $k^{self}_{6,7} = 128, k^{self}_{8,9} = 64, k^{cross}_{1-9} = 256$.

TABLE V: Ablation on loss.

Local keypoints	Matcher	Loss	P	A	R	F1
USIP	Our Desc.+ SuperGlue	raw loss	57.6	75.8	49.2	0.531
		triplet loss (hard)	62.1	82	**62.8**	0.624
		gap loss	64.8	82.9	62.4	0.636
	Our Matcher	raw loss	56.8	76.1	50.4	0.534
		triplet loss (hard)	61.8	81.9	61.6	0.617
		gap loss	**66.9**	**83.4**	62.3	**0.645**

As shown in Tab. IV, comparing to other setups, the gradually decreasing k design performs the best. The possible reason is that it is more natural to firstly observe the overall scene and then gradually focus on specific interests.

Ablation of the loss function. We evaluate our gap loss comparing to the raw loss used in SuperGlue and the triplet loss [40] using the hardest negative match, respectively. We retrain both of our matcher and SuperGlue on three loss functions and consequently get in total six different setups. The evaluation results shown in Tab. V suggest that our gap loss achieves the best performance. The results are not hard to explain that compared to the raw loss, the gap loss is more reasonable for the matching problem, and compared to the triplet loss, the gap loss utilizing all the negative matches is more efficient while being with barely any extra computational burden in our case.

VII. Conclusion

In this paper, we present MDGAT-matcher, a novel neural network-based keypoint matching approach for 3D point cloud data. Our network consists of three components: a keypoint encoder, a novel multiplex dynamic graph attention network, and an assignment layer. It utilizes both, the local vicinity information as well as contextual information to achieve better keypoint matching. We propose a dynamic graph construction with an attention mechanism to boost the network focusing on the key interest points. By introducing a new gap loss, our method can better distinguish true and false matches, and in turn learns a discriminative assignment matrix. We evaluate our approach on the KITTI odometry benchmark and provide

comparisons to the state-of-the-art approaches, including both, learned and hand-crafted ones. The experiments suggest that our approach outperforms existing methods and achieves a substantial improvement. In future work, we want to test our approach in different environments and extend it for tackling localization problem.

REFERENCES

[1] C. Stachniss, J. Leonard, and S. Thrun. *Springer Handbook of Robotics, 2nd edition*, chapter Chapt. 46: Simultaneous Localization and Mapping. Springer Verlag, 2016.

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

[3] Z. Zhang. Iterative point matching for registration of free-form curves and surfaces. *Intl. Journal of Computer Vision*, 13(2):119–152, 1994.

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

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

[6] P.E. Sarlin, D. Detone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[7] M. Simon, K. Fischer, S. Milz, C. Witt, and H. Groß. Stickypillars: Robust feature matching on point clouds using graph neural networks. *arXiv preprint*, 2020.

[8] J. Zhang and S. Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, page 9, 2014.

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

[10] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss. SuMa++: Efficient LiDAR-based Semantic SLAM. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, 2019.

[11] S. Gold, A. Rangarajan, C. P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2d and 3d point matching: pose estimation and correspondence. *Pattern Recognition*, 31(8):1019–1031, 1998.

[12] N. Chebrolu, T. Läbe, O. Vysotska, J. Behley, and C. Stachniss. Adaptive Robust Kernels for Non-Linear Least Squares Problems. *IEEE Robotics and Automation Letters*, 2021.

[13] Y. Tsin and T. Kanade. A correlation-based approach to robust point set registration. In *Proc. of the Europ. Conf. on Computer Vision*, 2004.

[14] A. Myronenko and X.B. Song. Point set registration: Coherent point drift. *IEEE Trans. on Pattern Analalysis and Machine Intelligence*, pages 2262–2275, 2010.

[15] X. Chen, T. Läbe, A. Milioto, T. Röhling, O. Vysotska, A. Haag, J. Behley, and C. Stachniss. OverlapNet: Loop Closing for LiDAR-based SLAM. In *Proc. of Robotics: Science and Systems*, 2020.

[16] R.Z. Wang, J.C. Yan, and X.K. Yang. Learning combinatorial embedding networks for deep graph matching. In *Proc. of the IEEE Intl. Conf. on Computer Vision)*, 2019.

[17] W.X. Lu, G.W. Wan, Y. Zhou, X.Y. Fu, P.F. Yuan, and S.Y. Song. Deepicp: An end-to-end deep neural network for 3d point cloud registration. *arXiv preprint*, 2019.

[18] Y. Aoki, H. Goforth, R.A. Srivatsan, and S. Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

[19] V. Sarode, X.Q. Li, H. Goforth, Y. Aoki, R.S. Srivatsan, S. Lucey, and H. Choset. Pcrnet: Point cloud registration network using pointnet encoding. *arXiv preprint*, 2019.

[20] Z.J. Yew and G.H. Lee. Rpm-net: Robust point matching using learned features. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2020.

[21] H. Xu, S. Liu, G. Wang, G. Liu, and B. Zeng. Omnet: Learning overlapping mask for partial-to-partial point cloud registration. *arXiv preprint*, 2021.

[22] V. Sarode, A. Dhagat, R.A. Srivatsan, N. Zevallos, and H. Choset. Masknet: A fully-convolutional network to estimate inlier points. *arXiv preprint*, 2020.

[23] Y. Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *Proc. of the Int. Conf. on Computer Vision Workshops*, 2009.

[24] Z.J. Yew and G.H. Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *Proc. of the Europ. Conf. on Computer Vision*, 2018.

[25] J.X. Li and G.H. Lee. Usip: Unsupervised stable interest point detection from 3d point clouds. In *Proc. of the IEEE Intl. Conf. on Computer Vision)*, 2019.

[26] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (fpfh) for 3d registration. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation*, pages 3212–3217, 2009.

[27] A. Zeng, S. Song, M. Nießner, M. Fisher, J. X. Xiao, and T. Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[28] Z. Gojcic, C. Zhou, J. D. Wegner, and A. Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2019.

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

[30] H.W. Deng, T. Birdal, and S. Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[31] A. Zanfir and C. Sminchisescu. Deep learning of graph matching. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2018.

[32] H.W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, pages 83–97, 1955.

[33] R. Sinkhorn. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics*, pages 876–879, 1964.

[34] Y. Wang and J.M. Solomon. Deep closest point: Learning representations for point cloud registration. In *Proc. of the IEEE Intl. Conf. on Computer Vision)*, 2019.

[35] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, and J.P. Onnela. Community structure in time-dependent, multiscale, and multiplex networks. *Science*, pages 876–878, 2010.

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

[37] Y.Q. Xie, S. Li, C. Yang, R.C. Wong, and J. Han. When do gnns work: Understanding and improving neighborhood aggregation. In *Proc. of the Intl. Conf. on Artificial Intelligence*, 2020.

[38] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint*, 2017.

[39] R. Sinkhorn and P. Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, pages 343–348, 1967.

[40] M. Khoury, Q.Y. Zhou, and V. Koltun. Learning compact geometric features. *Proc. of the IEEE Intl. Conf. on Computer Vision)*, 2017.

[41] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Computer ence*, 2014.

[42] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2012.

[43] F. Tombari, S. Salti, and L.D. Stefano. Unique signatures of histograms for local surface description. In *Proc. of the Europ. Conf. on Computer Vision*, 2010.