

# RoCNet: 3D Robust Registration of Point-Clouds using Deep Learning

Karim Slimani<sup>1</sup>, Brahim Tamadazte<sup>1</sup>, and Catherine Achard<sup>1</sup>

**Abstract**—This paper introduces a new method for 3D point cloud registration based on deep learning. The architecture is composed of three distinct blocs: (i) an encoder composed of a convolutional graph-based descriptor that encodes the immediate neighbourhood of each point and an attention mechanism that encodes the variations of the surface normals. Such descriptors are refined by highlighting attention between the points of the same set and then between the points of the two sets. (ii) a matching process that estimates a matrix of correspondences using the Sinkhorn algorithm. (iii) Finally, the rigid transformation between the two point clouds is calculated by RANSAC using the  $K^c$  best scores from the correspondence matrix. We conduct experiments on the ModelNet40 dataset, and our proposed architecture shows very promising results, outperforming state-of-the-art methods in most of the simulated configurations, including partial overlap and data augmentation with Gaussian noise.

**Index Terms**—Point clouds, Registration, Deep Learning, Attention Mechanisms, Pose Estimation

## I. INTRODUCTION

Point cloud registration is a widespread problem and a key task in 3D pose estimation in robotics and computer vision, with applications in autonomous driving [1], simultaneous localization and mapping (SLAM) [2], etc. The registration process involves matching points between the input and target point clouds, eliminating outliers, and estimating the rigid transformation parameters that align one point cloud to the other. Traditional algorithms, such as Iterative Closest Point (ICP) [3], alternate between matching and aligning iterations. Recently, neural network-based techniques, such as [4; 5; 6], are increasingly used to address these problems. These techniques typically encode each point and its neighbourhood through a learned descriptor, such as [7; 8], and use a transformer module [2; 5] to propagate local and global information between the two points sets. Point matching is often performed based on similarities in the descriptor space, and the transformation matrix can be estimated by integrating differentiable Singular Value Decomposition (SVD) into the network. Alternatively, WsDesc [9] suggested tackling the matching problem by looking, for each pair point, their nearest neighbour in the target point cloud during the training phase. Furthermore, recent work [2] suggested using an optimization procedure on the scoring matrix using the iterative Sinkhorn algorithm [10]. Therefore, the estimation of the rigid transformation can be achieved in two different ways, either end-to-

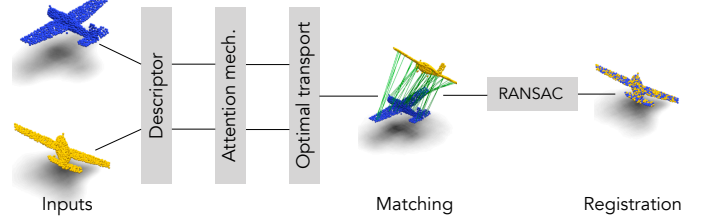


Fig. 1: Overall concept of the proposed RoCNet method.

end learning by integrating differentiable SVD [11] into the network as proposed in [5], or by applying a simple SVD or a RANSAC based on feature matching.

Certainly, the learning-based approaches have made significant progress and have led to overcome the numerous limitations of iterative methods, such as converging to a local minimum, conditioned by a correct initialization, and may find it difficult to estimate large transformations. Additionally, most of these methods extract point-wise features using the neighbourhood of each point by applying learning-based descriptors only. This makes these methods more sensitive to noise and outliers. The methods that apply RANSAC to overcome this problem may need a large number of iterations (e.g., 50, 000 iterations) to reach a correct estimation of the transformation.

In this paper, we proposed a new architecture (Fig. 1), called RoCNet, which includes three main blocks: 1) a descriptor that is composed of a convolutional graph-based network that encodes the immediate neighbourhood of each point and an attention mechanism that encodes the variations of the surface normals, 2) a matching module that estimates a matrix of correspondences using the Sinkhorn algorithm, 3) a RANSAC module which computes the rigid transformation using the  $K^c$  (e.g., 256) best matches with a limited number of iterations (e.g., 500 iterations). The proposed architecture was assessed using ModelNet40 dataset [4] in different favourable and unfavourable conditions. It has been demonstrated that our method outperformed the related state-of-the-art algorithms, especially in unfavourable conditions, e.g., with noisy data and partial occlusions.

## II. RELATED WORK

This section provides a review of the state-of-the-art regarding the main methods for 3D pose estimation and point cloud registration. We begin by introducing some interesting descriptors for 3D points cloud. Then, the main registration approaches are presented, which can be classified into three

This work was supported by the French ANR program MARSurg (ANR-21-CE19-0026).

<sup>1</sup> authors are with Sorbonne Université, CNRS UMR 7222, INSERM U1150, ISIR, F-75005, Paris, France. [karim.slimani@isir.upmc.fr](mailto:karim.slimani@isir.upmc.fr)

main categories: 1) iterative methods, 2) matching learning 3) transformation learning. For each category, descriptors can be handcrafted or learned.

#### A. 3D Point Cloud Descriptors

The first iterative methods presented below, use the 3D coordinates points directly as input to their system or handcrafted features like Fast Point Feature Histograms (FPFH) [12]. Since the rise of deep learning (DL), several methods have been developed to learn 3D point cloud descriptors. For instance, Qi *et al.* [13] proposed a neural architecture named PointNet describing unordered 3D points for tasks such as classification, segmentation, and registration. An extension named PointNet++ [8], which exploits metric space distances, has then been proposed by the same authors. Wang *et al.* [7] proposed the Dynamic Graph CNN (DGCNN) descriptor based on a module called EdgeConv that acts on graphs computed in each layer of the network. It captures semantic characteristics over potentially long distances as shown in segmentation and classification tasks. This descriptor is also used for registration tasks as reported in [4]. It is important to note that both previous descriptors are learnt on segmentation or classification tasks before being used, without new learning, on registration. In [2], the authors build a feature with two main components, a descriptor encoder that highlights handcrafted features obtained with FPFH and a positional encoder that highlights spatial properties of the point cloud. A multiplex dynamic graph attention network is added to reinforce the matching power of the descriptor. This descriptor is learnt conjointly with the matching process, in an end-to-end way.

#### B. Iterative Methods

The ICP [3] is probably the most popular method to address a point cloud registration problem. Given two sets of 3D points, the purpose of the algorithm is to minimise the Euclidean distance between the points. At each iteration, a mapping of the two sets of points and the computation of the 3D rigid transformation using an SVD are performed. This procedure is repeated until convergence. In RANSAC [14], the two-point clouds are randomly split into subsets on which a transformation is estimated. The final transformation is chosen among them using a criterion such as the weighted error on the set of points [15] or by selecting the transformation generating the largest number of inliers. Both methods have been associated with neural network based learned descriptors like in [5; 16] for ICP and in [9] for RANSAC. A most recent approach called Fast Global Registration (FGR) [17] operates on candidate matches that cover the surfaces. The surface alignment is defined with an objective optimized thanks to an iterative procedure.

#### C. Methods based on Matching Learning

Recent registration methods investigated DL architecture to match the points. Later, standard methods like RANSAC or SVD can be used to estimate the rigid transformation. For instance, Predator [18] is trained with three different

weighted matching losses to be more robust to low partial overlap between the input point clouds. Alternatively, 3DFeat-Net [19] train their architecture to detect key points and predict discriminative descriptors in a weakly supervised manner using the triplet loss [20], while D3Feat [21] combined two losses, one for the descriptor and the other for the detector. All these methods use a RANSAC module on feature matching to estimate the transformation parameters. In the MDGAT method [2], proposed to learn the matching using a new loss inspired by the triplet function. Roufousse *et al.* [22] proposed an unsupervised matching approach by optimizing the global structural properties of functional map [23], such as their bijectivity or approximate isometry. Such properties allow the creation of a loss function that does not require the knowledge of the ground truth during the learning process.

#### D. Methods based on Transformation Learning

In the Deep Closest Point (DCP) architecture [5], the descriptor is first performed using DGCNN and an attention-based module is introduced into a transformer. Then a soft SVD, where soft matching is used, allows computing the rigid transformation between both sets of points clouds. The training loss function is defined from this estimated rigid transformation that is compared to the ground truth one. In GeoTransformer [24], the input point clouds are down-sampled in several super-points (a subset of points) that are described using the Geometric Transformer which encodes intra-point-cloud and inter-point-cloud geometric structures. A matching module extracts super-point correspondences, each one being used to compute a soft SVD (from the subset of points corresponding to the super point) and estimate the transformation. The global transformation is the one that admits the most inlier matches over the transformation obtained for each super point. The loss function is composed of two terms: one measuring the alignment quality of super points and the other one measuring the alignment quality of the whole set of points. Wang *et al.* proposes PRNET [4] that, similarly to ICP, is designed to be applied iteratively to estimate the transformation between points clouds. The matching is performed using an approximately differentiable version of Gumbel-Softmax and the transformation is obtained using an SVD. Another approach was proposed in [9] which uses a differentiable nearest neighbour search algorithm in the descriptor space to match the points, and then proposes to relax the registration problem and seeks to estimate an affine transformation matrix computed by a least squares optimisation. An end-to-end architecture [6] includes a descriptor based on PointNet [13] and a neural network version of the Lucas & Kanade algorithm that allows incrementally updating the transformation.

### III. METHOD

#### A. Problem Statement

Let us start by defining a common problem of 3D point cloud registration. Considering two-point clouds  $\mathbf{X}$  and  $\mathbf{Y}$  such that:  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_i, \dots, \mathbf{x}_M\} \subset \mathbb{R}^{3 \times M}$  and  $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_j, \dots, \mathbf{y}_N\} \subset \mathbb{R}^{3 \times N}$ . It is assumed that the two sets at least partially overlap, so that there are  $K$  pairs of

matches between  $\mathbf{X}$  and  $\mathbf{Y}$ , with  $K \leq \min(M, N)$ . The two subsets containing the matching points in the first and second point clouds are defined by:  $\tilde{\mathbf{X}} \subset \mathbb{R}^{3 \times K}$  and  $\tilde{\mathbf{Y}} \subset \mathbb{R}^{3 \times K}$ , respectively. Note that the set  $\tilde{\mathbf{Y}}$  is obtained by applying a rotation  $\mathbf{R} \in SO(3)$  and a translation  $\mathbf{t} \in \mathbb{R}^3$  of the set  $\tilde{\mathbf{X}}$ . Both the rotation matrix  $\mathbf{R}$  and translation vector  $\mathbf{t}$  define the  $4 \times 4$  rigid transformation we are looking for, noticed  $\mathbf{T}$ .

### B. Descriptor

One of the most fundamental components in a point cloud registration problem is the relevance and quality of the descriptor used to encode the points. Therefore, we proposed a new descriptor by projecting the initial sets of points  $\mathbf{X}$  and  $\mathbf{Y}$  in a new base of higher dimension, de facto, more discriminating than the initial spatial representation, and as invariant as possible to rotations and translations. It combines a geometrical-based descriptor and a normal-based one, followed by an attention mechanism.

1) *Geometrical-based descriptor*: Different types of descriptors, that learn local geometrical properties around each point, were reported in the literature such as PointNet [13], PointNet++ [8] or DGCNN descriptor [7]. We integrated DGCNN as a part of our descriptor because it better captures local geometric features of point clouds while still maintaining permutation invariance. It consists of mainly *EdgeConv* convolution layers where the points represent nodes connected by arcs to their  $k$  nearest neighbours in the encoding space to build graphs that express the local geometric structure surrounding each point and then spread dynamically the information at a higher level (global encoding). Let us denote  $\mathbf{f}_i^X$  the extracted feature vector of dimension  $d$  for point  $\mathbf{x}_i$ .

2) *Normal based descriptor*: The main idea of this descriptor is to better encode the surface around each point using the variation of the normals of points in the neighbourhood: in a flat surface, there is no variation of the normals, along a ridge, the normals vary only in one direction, whereas on a summit, the normals vary in all directions. Thus, the variation of the angle of the normals in a neighbourhood is informative about the type of surface.

The normals are estimated using Principal Component Analysis (PCA). Indeed, for each point  $\mathbf{x}_i \in \mathbf{X}$ , a local neighbourhood subset of points  $S_i = \{\mathbf{x}_j / \|\mathbf{x}_j - \mathbf{x}_i\|^2 \leq r\}$  is defined while delimiting the size of the set points with  $|S_i| < K_{nn}$ .  $r$  is the radius of a sphere centred in  $\mathbf{x}_i$  and  $K_{nn}$  the maximum number of points included in the set  $S_i$ . The eigenvalue decomposition of the covariance matrix  $Cov(S_i)$  allows defining the normal  $\mathbf{n}_i$  as the vector associated with the smallest eigenvalue. The covariance matrix  $Cov(S_i)$  is expressed as follows:

$$Cov(S_i) = \frac{1}{|S_i|} \sum_{\mathbf{x}_j \in S_i} (\mathbf{x}_j - \mathbf{x}_i)(\mathbf{x}_j - \mathbf{x}_i)^T \quad (1)$$

where  $|S_i|$  represents the number of points in  $S_i$ .

Since the PCA does not inherently determine the direction of the normal vector which can point in either direction, we propose to address the ambiguity of the sign by using a new vector  $\mathbf{z}_i$ . It is colinear to  $\mathbf{n}_i$  and is defined by ensuring

that it points towards the side of the surface with a higher point density. This means that the normal vector should be pointing away from sparse areas and towards denser surface areas. Similar to [9], we solve this ambiguity thanks to the:

$$\mathbf{z}_i = \begin{cases} \mathbf{n}_i, & \text{if } \sum_{\mathbf{x}_j \in S_i} \mathbf{n}_i^T (\mathbf{x}_i - \mathbf{x}_j) \geq 0 \\ -\mathbf{n}_i, & \text{otherwise} \end{cases} \quad (2)$$

Finally, we build the final encoding based on [24] and [25] using sinusoidal functions of different frequencies. Knowing the angle between the normals of two points  $\mathbf{x}_i$  and  $\mathbf{x}_j$  noted  $\angle(\mathbf{z}_i, \mathbf{z}_j)$ , the vector  $\mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j}$  encoding the normals is given by:

$$\begin{cases} \mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j}^{2ind} = \sin\left(\frac{\angle(\mathbf{z}_i, \mathbf{z}_j)}{\tau \times 10000^{2ind/d}}\right) \\ \mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j}^{2ind+1} = \cos\left(\frac{\angle(\mathbf{z}_i, \mathbf{z}_j)}{\tau \times 10000^{2ind/d}}\right) \end{cases} \quad (3)$$

where  $ind$  is the current value index of  $\mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j}$ ,  $\tau$  a normalisation coefficient and  $d$  the dimension of the descriptor  $\mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j}$  fixed to the same size as the geometrically based descriptor DGCNN. A fully connected layer is then applied to  $\mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j}$  to obtain the final embedding

$$\mathbf{e}_{i,j}^X = \mathbf{g}_{\mathbf{x}_i, \mathbf{x}_j} \mathbf{W}_E^s \quad (4)$$

where  $\mathbf{W}_E^s \in \mathbb{R}^{d \times d}$  is a learned projection matrix.

3) *Attention mechanism*: A key point of recent descriptors of points cloud is the introduction of an attention mechanism that highlights some features dynamically. SuperGlue [26] uses a module based on attention graphs that alternately stacks 'self-attention' and 'cross-attention' layers. The former links all the nodes of a point cloud to each other, while the latter links each point of set  $\mathbf{X}$  to all points of set  $\mathbf{Y}$ . Contrary to SuperGlue [26] or MDGAT [2], which compute the attention weights on the encoding vectors, some methods propose adding information on local inter-points geometry at the entry of the mechanism. For instance, [27] associates the 3D coordinates of each point with the descriptor, while GeoTransformer [24] proposes to use the distances and angles between each point and its  $k$  nearest neighbours. Alternatively, in our approach, we propose the use of four attention heads with geometric self-attention inside each set  $\mathbf{X}$  and  $\mathbf{Y}$  integrating the associated normals embeddings  $\mathbf{e}^X$  and  $\mathbf{e}^Y$ , respectively, followed by cross-attention between the two sets of points and then alternate between them for  $L$  times.

4) *Self-attention*: This type of layer predicts an attention-based feature  $\tilde{\mathbf{f}}_i$  for each point of a point cloud ( $\mathbf{X}$  or  $\mathbf{Y}$ ), paying attention to all the other points of the same cloud. In the following, the algorithm is detailed for a point  $\mathbf{x}_i \in \mathbf{X}$ , the same is used for all the points in  $\mathbf{X}$  and  $\mathbf{Y}$ . Thus, an attention weight is then obtained for each query/key pair:

$$\alpha_{ij}^X = \underset{j}{softmax} \left( \frac{(\mathbf{f}_i^X \mathbf{W}_Q^s)(\mathbf{f}_j^X \mathbf{W}_K^s + \mathbf{e}_{i,j}^X \mathbf{W}_R^s)^T}{\sqrt{d}} \right) \quad (5)$$

where  $\mathbf{W}_Q^s$ ,  $\mathbf{W}_K^s$  and  $\mathbf{W}_R^s \in \mathbb{R}^{d \times d}$  are the learnt projection matrices for queries, keys and normal-based embeddings,  $d$  is the dimension of the features  $\mathbf{f}_i^X$  and  $\mathbf{e}_{i,j}^X$ . These weights are used to rate which elements we have to pay attention to, and to obtain the final self-attention-based feature  $\tilde{\mathbf{f}}_i^X$ :

$$\tilde{\mathbf{f}}_i^X = \sum_{j=1} \alpha_{ij}^X \mathbf{v}_j \quad (6)$$

with,

$$\mathbf{v}_j = \mathbf{f}_j^X \mathbf{W}_V^s \quad (7)$$

where  $\mathbf{W}_V^s \in \mathbb{R}^{d \times d}$  is the learnt projection matrix for values.

5) *Cross-Attention*: A *cross-attention* layer is used to propagate the local information between the two previously obtained representations  $\bar{\mathbf{f}}_i^X$  and  $\bar{\mathbf{f}}_j^Y$  of  $\mathbf{x}_i$  and  $\mathbf{y}_j$  belonging respectively to point clouds  $\mathbf{X}$  and  $\mathbf{Y}$ . Formally, it works similarly as for the *self-attention* layer, except for the estimation of the attention key, which now uses a point in the second point cloud. The final encoding for any point  $\mathbf{x}_i$  (or  $\mathbf{y}_j$ ) is given by:

$$\mathbf{h}_i^X = \sum_{j=1}^{|\mathbf{Y}|} \left( \text{softmax}_j \left( \frac{(\bar{\mathbf{f}}_i^X \mathbf{W}_Q^c)(\bar{\mathbf{f}}_j^Y \mathbf{W}_K^c)^T}{\sqrt{d}} \right) \right) \left( \bar{\mathbf{f}}_j^Y \mathbf{W}_V^c \right) \quad (8)$$

where  $\mathbf{W}_Q^c$ ,  $\mathbf{W}_K^c$  and  $\mathbf{W}_V^c \in \mathbb{R}^{d \times d}$  are the learnt projection matrices for queries, keys and values in the cross-attention layers.

### C. Point Matching

The second step of the proposed algorithm is the matching procedure. We first estimate a score matrix  $\mathbf{C} \in \mathbb{R}^{M \times N}$  between each point  $\mathbf{x}_i \in \mathbf{X}$  and  $\mathbf{y}_j \in \mathbf{Y}$ :

$$\mathbf{C}_{i,j} = \mathbf{h}_i^X{}^\top \mathbf{h}_j^Y \quad (9)$$

where  $\mathbf{h}_i^X$  and  $\mathbf{h}_j^Y$  are the final encoding of the points  $\mathbf{x}_i$  and  $\mathbf{y}_j$  defined previously. To build a matrix of correspondence probabilities  $\bar{\mathbf{C}}$ , we first augment the dimensions of  $\mathbf{C}$  to  $M+1$  and  $N+1$  respectively, such that the non-matched points will explicitly be assigned to the last dimensions. We then use the differentiable *Sinkhorn Algorithm* [10] which is widely used in optimal transport and graph-matching problems.

As all the previous steps are differentiable, the weights of the networks can be learnt by introducing a loss function. To do so, we follow [2] and adopt the gap loss function (10) which allows enlarging the assignment scores difference between the true matches and the wrong matches. It is expressed as follows:

$$L_{Gap} = \sum_{i=1}^M \log \left( \sum_{n=1}^{N+1} [\max((- \log \bar{\mathbf{C}}_{i,\bar{i}} + \log \bar{\mathbf{C}}_{i,n} + \alpha), 0)] + 1 \right) + \sum_{j=1}^N \log \left( \sum_{n=1}^{M+1} [\max((- \log \bar{\mathbf{C}}_{j,\bar{j}} + \log \bar{\mathbf{C}}_{n,j} + \alpha), 0)] + 1 \right) \quad (10)$$

where  $\alpha$  is a positive scalar having a value of 0.5,  $\bar{\mathbf{C}}_{i,\bar{i}}$  and  $\bar{\mathbf{C}}_{j,\bar{j}}$  are the scores for the ground truth true matches of the points  $\mathbf{x}_i$  and  $\mathbf{y}_j$ , respectively.

### D. Pose Estimation

In the evaluation phase, we build a hard assignment binary matrix  $\mathbf{A}$  thanks to the following algorithm:

$$\mathbf{A}^1_{i,j} = \begin{cases} 1 & \text{if } \bar{\mathbf{C}}_{i,j} = \max_n(\bar{\mathbf{C}}_{i,n}) \\ 0 & \text{otherwise,} \end{cases} \quad (11)$$

$$\mathbf{A}^2_{i,j} = \begin{cases} 1 & \text{if } \bar{\mathbf{C}}_{j,i} = \max_n(\bar{\mathbf{C}}_{j,n}) \\ 0 & \text{otherwise,} \end{cases} \quad (12)$$

$$\mathbf{A}_{i,j} = \mathbf{A}^1_{i,j} \times \mathbf{A}^2_{i,j} \quad (13)$$

The matrix  $\mathbf{A}$  gives us the two final sets of matched points  $\bar{\mathbf{X}} \in \mathbb{R}^K$  and  $\bar{\mathbf{Y}} \in \mathbb{R}^K$  by re-indexing the original point clouds  $\mathbf{X} \in \mathbb{R}^M$  and  $\mathbf{Y} \in \mathbb{R}^N$  with the row and column indices of the non-zero values of the matrix  $\mathbf{A}$  respectively. An example of a performed matching is depicted in Fig. 3. Once the sets of matched points are built, different techniques can be used to determine the rigid transformation. A classical SVD to the cross-covariance matrix between the centred subsets  $\bar{\mathbf{X}}$  and  $\bar{\mathbf{Y}}$  is used in MDGAT [2], while DCP [5] suggested a differentiable and soft SVD where the weights of each point are determined by applying a *Softmax* to the score matrix  $\mathbf{C}$ . An alternative method is to apply a RANSAC technique based on feature matching as reported in [18; 9]. In our method, we propose to use RANSAC based on our predicted correspondences to reduce the computational cost. Moreover, instead of considering all the  $K$  matched points, we only use the  $K^c$  most relevant ones allowing us to filter the outliers before the first iteration such that the transformation is performed in 500 iterations maximum.

## IV. EXPERIMENTS

### A. Dataset and Parametrisation

To assess the RoCNet, we opted for the ModelNet40 dataset [4]. It is a synthetic database representing 40 classes of objects designed using Computer-Aided Design (CAD) software. The database consists of 12,311 3D point clouds divided into 9,843 sets for training and 2,468 for testing. Each of these point clouds is scaled to fit inside a sphere of radius  $r = 1$  m. For each initial point cloud called  $\mathbf{X}$ , a new point cloud  $\mathbf{Y}$  is created by applying a random rigid transformation with a rotation ranging from  $0^\circ$  to  $45^\circ$  around each axis and a translation ranging from 0 cm to 50 cm in each direction, and finally, a random permutation of the points is performed. In the end, a point cloud of 1024 points is generated for each object. In addition, to be able to evaluate the robustness of registration or pose estimation methods, the initial points clouds are reduced by a range of points emulating partial occlusions.

The following lists the different configurations used to assess our method in terms of accuracy, robustness and generalization:

- *Partial overlap*: To simulate partial occlusions, 768 points are subsampled from the 1024 components of both the first and the second points clouds  $\mathbf{X}$  and  $\mathbf{Y}$ , respectively. Specifically, between 512 and 768 points from  $\mathbf{X}$  have a match in  $\mathbf{Y}$ .
- *Noisy data*: Clipped Gaussian noise with a range of  $[-0.05, 0.05]$ , a mean of  $\mu = 0$ , and a variance of 0.01 is added to each point.
- *Partial overlap and noisy data*: The same type of noise is also added to the subsampled point clouds.
- *Unseen objects*: To test the generalization of the registration and pose estimation methods, the model is trained



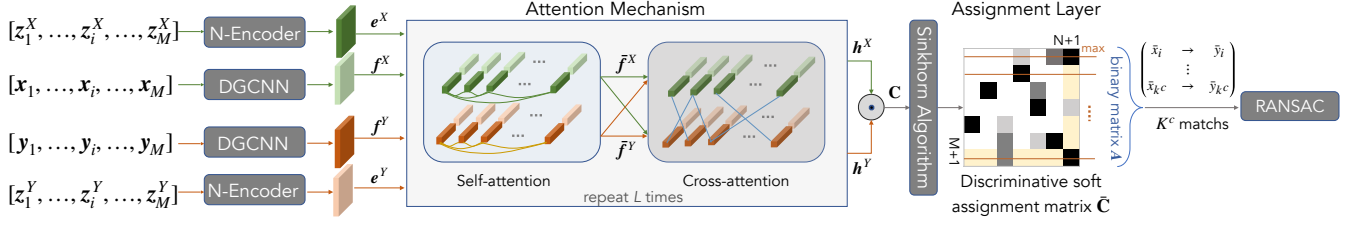


Fig. 2: Overview of the proposed RoCNet architecture.

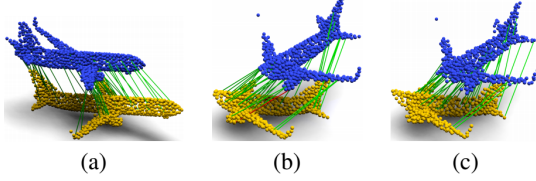


Fig. 3: Example of a performed 3D matching of point clouds in different configurations: (a) clean data, (b) partial overlap, and (c) noisy data and partial overlap. The green lines show the correct matches and the red lines show the wrong ones.

only on the first 20 classes of objects and tested on the remaining 20 categories.

RoCNet is trained for 30 epochs on the clean data and for 80 epochs on the noisy data with a learning rate of  $10^{-4}$  in both cases. The number of predicted correspondences used as input in RANSAC is  $K^c = 256$ . The parameters  $d$  and  $L$  are set to 96 and 6 respectively.

### B. Metrics

To benchmark our RoCNet architecture against state-of-the-art methods, we opted for two metrics widely used in the literature which consist of the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) used to compute the difference in rotation and translation between the estimated transformation and the provided ground truth. The RMSE and MAE are respectively defined as follows:

$$\text{RMSE}(\mathbf{p}) = \left( \frac{1}{M} \sum_{i=1}^M \|\mathbf{p}_i - \mathbf{p}_i^{GT}\|^2 \right)^{\frac{1}{2}} \quad (14)$$

$$\text{MAE}(\mathbf{p}) = \frac{1}{M} \sum_{i=1}^M |\mathbf{p}_i - \mathbf{p}_i^{GT}| \quad (15)$$

with,  $M$  is the number of pairs of point clouds in the test set,  $\mathbf{p}_i$  and  $\mathbf{p}_i^{GT}$  are the estimated rotation (Euler angles) or translation and the ground truth ones, respectively.

### C. Results

RoCNet is assessed in different configurations (i.e., favourable and unfavourable) and compared to the main DL-based methods of the state-of-the-art as well as with the traditional ICP approach. The recent VRNet [28] has summarised nicely the performance of most of the related methods reported in the literature. We use this performance as a basis for comparison to which we have added those of recently

published methods such as WsDesc [9] and R-PointHop [16]. Note that in all the tables,  $\mathbf{R}$  is given in degrees, and  $\mathbf{t}$  is in meters, while the best results are highlighted in bold and the second ones are underlined.

1) *Model trained on all classes with clean data:* The first configuration consists of the evaluation of RoCNet when the model is trained across all the provided 40 classes of objects. All the data are clean and do not involve occlusions.

TABLE I: Performances of RoCNet using all classes without noise and occlusions.

Method	RMSE( $\mathbf{R}$ )	MAE( $\mathbf{R}$ )	RMSE( $\mathbf{t}$ )	MAE( $\mathbf{t}$ )
ICP'92 [3]	12.28	4.613	0.04774	0.00228
PTLK'19 [6]	13.75	3.893	0.01990	0.00445
DCP-V2'19 [5]	1.090	0.752	0.00172	0.00117
PRNET'19 [4]	1.722	0.665	0.00637	0.00465
R-PointHop'22 [16]	0.340	0.240	<u>0.00037</u>	0.00029
VRNet'22 [28]	<u>0.091</u>	<u>0.012</u>	<b>0.00029</b>	<b>0.00005</b>
<b>Ours</b>	<b>0.082</b>	<b>0.011</b>	0.00047	<u>0.00008</u>

Table I provides the results. It can be highlighted that our method outperforms the other methods in rotation with an improvement of a dozen per cent for the RMSE and MAE *versus* the second best method VRNet [28]. However, VRNet remains the best in translation although the difference is slight compared to RoCNet, specifically for MAE( $\mathbf{t}$ ) where RoCNet is second. From a purely numerical point of view, RoCNet provides an RMSE( $\mathbf{R}$ ) of  $0.082^\circ$  (respectively, an MAE( $\mathbf{R}$ ) of  $0.011^\circ$ ) (mean of the three Euler angle rotations) and an RMSE( $\mathbf{t}$ ) of  $47 \times 10^{-5}$  m (respectively, an MAE( $\mathbf{t}$ ) of  $8 \times 10^{-5}$  m) (mean of the three translations along  $x$ ,  $y$ , and  $z$  axes).

2) *Model trained on all classes with noisy data:* Table II depicts the results of RoCNet under the same conditions as the first assessment but with adding Gaussian noise to the initial data. RoCNet achieves the best performance on three of the four metrics with an improvement of 25%, 45% and 56% for RMSE( $\mathbf{R}$ ), MAE( $\mathbf{R}$ ) and RMSE( $\mathbf{t}$ ) respectively, and ranked second in MAE( $\mathbf{t}$ ) behind R-PointHop. From a numerical point of view, RoCNet provides an RMSE( $\mathbf{R}$ ) of  $1.92^\circ$  (respectively, an MAE( $\mathbf{R}$ ) of  $0.55^\circ$ ) and an RMSE( $\mathbf{t}$ ) of  $2.6 \times 10^{-3}$  m (respectively, an MAE( $\mathbf{t}$ ) of  $1.8 \times 10^{-3}$  m).

3) *Model trained on half classes with clean data:* The third configuration consists of the evaluation of the generalisation capacity of RoCNet and the related methods when the models are trained on only half the data (i.e., 20 classes), and tested on the 20 remaining classes. The obtained performances are reported in Table III. RoCNet outperforms the state-of-the-art



Fig. 4: Illustration of some examples of performed registrations using RoCNet in case of clean data and without occlusions *versus* the ground truth registrations.

TABLE II: Performances of RoCNet when the model is trained in all the classes with noisy data and without occlusions.

Method	RMSE( <b>R</b> )	MAE( <b>R</b> )	RMSE( <b>t</b> )	MAE( <b>t</b> )
ICP'92 [3]	11.971	4.497	0.04832	0.00433
PTLK'19 [6]	15.692	3.992	0.02395	0.00563
DCP-V2'19 [5]	08.417	5.685	0.03183	0.02337
PRNET'19 [4]	03.218	1.446	0.11178	0.00837
R-PointHop'22 [16]	02.780	0.980	0.01400	<b>0.00080</b>
VRNet'22 [28]	02.558	1.016	0.00570	0.00289
<b>Ours</b>	<b>01.920</b>	<b>0.555</b>	<b>0.00260</b>	0.00180

method in one metric which is the MAE(**R**) and ranked second for both RMSE(**R**) and MAE(**t**). Overall, the performance of our method is similar to that of VRNet and R-PointHop.

TABLE III: Performances of RoCNet when the model is trained in half the classes with clean data and without occlusions.

Method	RMSE( <b>R</b> )	MAE( <b>R</b> )	RMSE( <b>t</b> )	MAE( <b>t</b> )
ICP'92 [3]	12.707	5.075	0.04853	0.00235
PTLK'19 [6]	15.901	4.032	0.02611	0.00621
DCP-V2'19 [5]	3.256	2.102	0.00631	0.00462
PRNET'19 [4]	3.060	1.326	0.01009	0.00759
R-PointHop'22 [16]	0.340	0.240	<b>0.00040</b>	0.00030
VRNet'22 [28]	<b>0.209</b>	0.028	0.00078	<b>0.00009</b>
<b>Ours</b>	0.235	<b>0.026</b>	0.00180	0.00020

4) *Model trained on all the classes with clean data and under partial occlusions:* In this assessment, we evaluate the behaviour of our method and the other methods when only a part is shared by the two point clouds to be aligned. This simulates, for example, partial occlusions. From Table IV, it can be highlighted that RoCNet outperforms significantly the others methods in all the metrics. Overall, our method reduces the registration error by roughly half in comparison with the second-ranked methods, i.e., VRNet and R-PointHop.

5) *Model trained on all the classes with noisy data and under partial occlusions:* The last configuration concerns the evaluation of the proposed method under partial occlusions (partial overlap) using noisy data. As can be seen in Table V, RoCNet outperforms the other methods on all metrics, both in rotation and translation. RoCNet allows significant enhance-

TABLE IV: Performances of RoCNet when the model is trained in all the classes with clean data and partial occlusions.

Method	RMSE( <b>R</b> )	MAE( <b>R</b> )	RMSE( <b>t</b> )	MAE( <b>t</b> )
ICP'92 [3]	33.683	25.045	0.293	0.2500
PTLK'19 [6]	16.735	07.550	0.045	0.0250
DCP-V2'19 [5]	06.709	04.448	0.027	0.0200
PRNET'19 [4]	03.199	01.454	0.016	0.0100
R-PointHop'22 [16]	01.660	00.350	0.014	0.0008
VRNet'22 [28]	00.982	00.496	0.006	0.0039
WsDesc'22 [9]	01.187	00.975	0.008	0.0070
<b>Ours</b>	<b>00.412</b>	<b>00.133</b>	<b>0.002</b>	<b>0.0002</b>

ment of the registration error, ranging from two-thirds to one-quarter compared to the method ranked second, i.e., WsDesc and even more in comparison to VRNet. This can be explained by the robustness of RoCNet to partial occlusions or noise or both at the same time.

TABLE V: Performances of RoCNet when the model is trained in all the classes with noisy data and partial occlusions.

Method	RMSE( <b>R</b> )	MAE( <b>R</b> )	RMSE( <b>t</b> )	MAE( <b>t</b> )
ICP'92 [3]	33.067	25.564	0.294	0.250
PTLK'19 [6]	19.939	9.076	0.057	0.032
DCP-V2'19 [5]	06.883	4.534	0.028	0.021
PRNET'19 [4]	04.323	2.051	0.017	0.012
VRNet'22 [28]	03.615	1.637	0.010	0.006
WsDesc'22 [9]	03.500	0.759	0.006	0.004
<b>Ours</b>	<b>01.810</b>	<b>0.620</b>	<b>0.004</b>	<b>0.003</b>

Finally, the performance of RoCNet and its ranking in the context of wider state-of-the-art, including the eleven best methods, can be seen in Fig. 5. It can be highlighted that our method outperforms all the methods when considering simultaneously performances in rotation and translation, this both on clean data (Fig. 5(a)) and on noisy data (Fig. 5(b)).

Figure 4 depicts some examples of aligned points clouds (objects) available in the ModelNet40 dataset. The first row shows the initial positions of the points cloud  $X$  and  $Y$  to be aligned, the second row shows the performed registrations and the third shows the ground truth ones.

Furthermore, to be able to assess visually the robustness ability of the proposed method, we performed different regis-

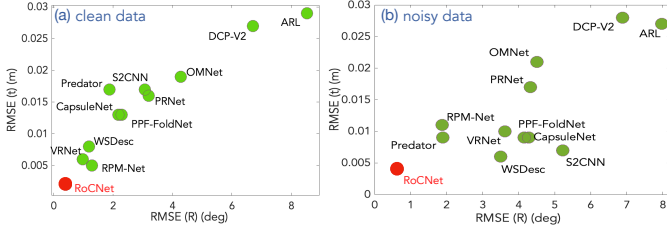


Fig. 5: Comparison of RoCNet performances against the eleven most relevant state-of-the-art methods. (a) in the case of clean data and (b) in the case of data with Gaussian noise, both with partial overlap.

trations by progressively decreasing (from 95% to 50%) the rate of shared points between  $X$  and  $Y$ . Figure 6 depicts the obtained results of one object. As can be seen, RoCNet can register point clouds even with only 50% of the data without much difficulty. On the other hand, the method shows its limits for objects with perfect symmetry when the overlap is low.

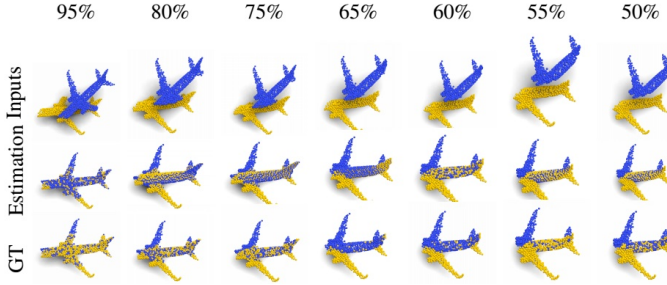


Fig. 6: Illustration of the robustness of RoCNet against partial occlusions (i.e., partial overall between the point clouds to be aligned).

## V. ABLATION STUDY

We conduct ablation studies on the three main blocks of the proposed architecture, i.e., the descriptor, the transformer, and the RANSAC-based estimation of the transformation.

### A. Ablation of Descriptor

To study the impact of our descriptor which uses both normals and DGCNN as inputs on the transformer, we compared the performance of our architecture to that of MDGAT [2] in case of point matching problem. For a proper comparison, both methods are trained in the same configuration and with the same number of epochs. Two types of data are used: 1) clean data and 2) noisy data, both with partial overlap. To achieve the comparison, we use the following metrics: Precision (**P**), Accuracy (**A**), Recall (**R**) and F1-score (**F1**). Table VI gives an insight into the ablation study of the descriptor. It can be underlined that our descriptor outperforms MDGAT one except for the Precision (**P**) in the case of clean data with partial overlap. The difference is substantial, in favour of our descriptor when it concerns noisy data with partial overlap.

TABLE VI: Performances assessment in a matching challenge with partial overlap matching (**P**: Precision, **A**: Accuracy, **R**: Recall, and **F1**: F1-score).

Method	clean data				noisy data			
	<b>P</b>	<b>A</b>	<b>R</b>	<b>F1</b>	<b>P</b>	<b>A</b>	<b>R</b>	<b>F1</b>
MDGAT	<b>98.1</b>	93.7	93.4	95.7	85.7	68.5	75.3	80.2
Ours	98.0	<b>97.2</b>	<b>97.6</b>	<b>97.8</b>	<b>85.8</b>	<b>85.9</b>	<b>85.5</b>	<b>85.7</b>

### B. Ablation of DGCNN and Transformer

RoCNet is compared to other alternatives in which our descriptor and attention mechanism are changed to those proposed in [2]. In view of evaluating the contribution of the proposed attention mechanism, another architecture of the DGCNN is associated with a classical mechanism without the normals gradient embedding [26]. Table VIII reports the obtained results showing that RoCNet architecture is more relevant on three of the four used metrics emphasizing a significant contribution (about 10%) of the association of DGCNN and normals compared to a classical attention mechanism.

### C. Ablation of RANSAC

The last ablation study consists of the comparison of the contribution of an SVD *versus* RANSAC to estimate the rigid transformation when the matching is performed. Table VII reports the performances of each alternative using: 1) clean data with full overlap (full), 2) noisy data with full overlap (noisy) and 3) clean data with partial overlap (partial). It can be seen that RANSAC approach outperforms slightly the SVD one.

## VI. CONCLUSION

This paper presented a new 3D point cloud registration and pose estimation using a deep learning architecture. The proposed architecture is composed of three main blocks: 1) the newly designed descriptor which encodes the neighbourhood of each point and an attention mechanism that encodes the variations of the surface normals, 2) the matching method that estimates a matrix of correspondences using the Sinkhorn algorithm, and 3) the estimation of the rigid transformation using a RANSAC applied to the  $K^c$  best matches from the correspondence matrix. The proposed architecture was evaluated using the ModelNet40 dataset in different favourable and unfavourable configurations. It has been demonstrated that our method outperformed the related state-of-the-art algorithms, especially in unfavourable conditions, e.g., with noisy data and partial occlusions.

In the future, we intend to extend this work to a new approach where the descriptor will be expressed in the frequency range. This will certainly improve the accuracy of our architecture, but also its robustness to noise and partial occlusions.

## REFERENCES

- [1] S. Chen, B. Liu, C. Feng, *et al.*, “3d point cloud processing and learning for autonomous driving: Impacting

TABLE VII: SVD *versus* RANSAC for transformation estimation in case of full overlap, noisy data and partial overlap.

Method	RMSE(R)			MAE(R)			RMSE(t)			MAE(t)		
	full	noisy	partial	full overlap	noisy	partial	full	noisy	partial	full	noisy	partial
SVD	3.94	1.94	4.21	0.124	<b>0.484</b>	0.168	<b>0.0005</b>	<b>0.0017</b>	<b>0.001</b>	<b>0.0001</b>	<b>0.0018</b>	<b>0.0001</b>
RANSAC	<b>0.06</b>	<b>1.92</b>	<b>0.40</b>	<b>0.010</b>	0.555	<b>0.133</b>	<b>0.0005</b>	0.0026	0.002	<b>0.0001</b>	<b>0.0018</b>	0.0002

TABLE VIII: Performances assessment in a matching challenge with clean data and partial overlap.

Descriptor	Attention	P	A	R	F1
MDGAT's descriptor	<b>Ours</b>	92.1	84.6	84.8	88.3
<b>Ours</b>	MDGAT	96.6	96.3	96.3	95.4
<b>Ours</b>	<b>Ours</b> w/o normals	79.8	80.2	80.5	80.1
MDGAT's descriptor	MDGAT	<b>98.1</b>	93.7	93.4	<b>95.7</b>
<b>Ours</b>	<b>Ours</b>	<u>98.0</u>	<b>97.2</b>	<b>97.6</b>	<b>97.8</b>

map creation, localization, and perception,” *IEEE Sig. Process. Mag.*, vol. 38, pp. 68–86, 2020.

- [2] C. Shi, X. Chen, K. Huang, *et al.*, “Keypoint matching for point cloud registration using multiplex dynamic graph attention networks,” *IEEE Rob. and Auto. Let.*, vol. 6, pp. 8221–8228, 2021.
- [3] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611, pp. 586–606, 1992.
- [4] Y. Wang and J. M. Solomon, “Prnet: Self-supervised learning for partial-to-partial registration,” *Adv. Neural. Inf. Process. Syst.*, vol. 32, 2019.
- [5] Y. Wang and J. M. Solomon, “Deep closest point: Learning representations for point cloud registration,” in *IEEE Int. Conf. on Comput. Vision*, 2019.
- [6] Y. Aoki, H. Goforth, R. A. Srivatsan, *et al.*, “Pointnetlk: Robust & efficient point cloud registration using pointnet,” in *IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pp. 7163–7172, 2019.
- [7] Y. Wang, Y. Sun, *et al.*, “Dynamic graph cnn for learning on point clouds,” *ACM Trans. on Grap.*, 2019.
- [8] C. R. Qi, L. Yi, H. Su, *et al.*, “Pointnet++: Deep hierarchical feature learning on point sets in a metric space,” *Adv. Neural. Inf. Process. Syst.*, vol. 30, 2017.
- [9] L. Li, H. Fu, and M. Ovsjanikov, “Wsdsc: Weakly supervised 3d local descriptor learning for point cloud registration,” *IEEE Trans. Vis. Comput. Graph.*, 2022.
- [10] R. Sinkhorn and P. Knopp, “Concerning nonnegative matrices and doubly stochastic matrices,” *Pacific J. of Math.*, vol. 21, pp. 343–348, 1967.
- [11] T. Papadopoulos and M. L. Lourakis, *Estimating the jacobian of the singular value decomposition: Theory and applications*. PhD thesis, Inria, 2000.
- [12] R. B. Rusu, N. Blodow, *et al.*, “Fast point feature histograms (fpfh) for 3d registration,” in *IEEE Int. Conf. on Rob. and Auto.*, pp. 3212–3217, 2009.
- [13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “Pointnet: Deep learning on point sets for 3d classification and segmentation,” *arXiv preprint arXiv:1612.00593*, 2016.
- [14] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, vol. 24, pp. 381–395, 1981.
- [15] M. El Banani, L. Gao, and J. Johnson, “Unsupervisedr&r: Unsupervised point cloud registration via differentiable rendering,” in *IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pp. 7129–7139, 2021.
- [16] P. Kadam, M. Zhang, S. Liu, *et al.*, “R-pointnet: A green, accurate, and unsupervised point cloud registration method,” *IEEE Trans. on Im. Process.*, vol. 31, pp. 2710–2725, 2022.
- [17] Q. Zhou, J. Park, *et al.*, “Fast global registration,” in *Eur. Conf. Comput. Vis.*, vol. 9906, pp. 694–711, 2016.
- [18] S. Huang, Z. Gojcic, M. Usvyatsov, *et al.*, “Predator: Registration of 3d point clouds with low overlap,” in *IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pp. 4267–4276, 2021.
- [19] Z. J. Yew and G. H. Lee, “3dfeat-net: Weakly supervised local 3d features for point cloud registration,” in *Euro. Conf. on Comput. Vision*, 2018.
- [20] M. Khoury, Q.-Y. Zhou, *et al.*, “Learning compact geometric features,” in *IEEE Int. Conf. Comput. Vision*, pp. 153–161, 2017.
- [21] X. Bai, Z. Luo, L. Zhou, *et al.*, “D3feat: Joint learning of dense detection and description of 3d local features,” in *IEEE/CVF Conf. Comput. Vision Pattern Recognit.*, pp. 6359–6367, 2020.
- [22] J.-M. Roufousse, Sharma, *et al.*, “Unsupervised deep learning for structured shape matching,” in *IEEE/CVF Int. Conf. Comput. Vision*, pp. 1617–1627, 2019.
- [23] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas, “Functional maps: a flexible representation of maps between shapes,” *ACM Trans. on Grap.*, vol. 31, pp. 1–11, 2012.
- [24] Z. Qin, H. Yu, C. Wang, *et al.*, “Geometric transformer for fast and robust point cloud registration,” in *IEEE/CV Conf. Comput. Vision Pattern Recognit.*, pp. 11143–11152, 2022.
- [25] A. Vaswani, N. Shazeer, *et al.*, “Attention is all you need,” *Adv. Neural. Inf. Process. Syst.*, vol. 30, 2017.
- [26] P.-E. Sarlin, D. DeTone, *et al.*, “SuperGlue: Learning feature matching with graph neural networks,” in *Conf. Comput. Vision Pattern Recognit.*, 2020.
- [27] H. Zhao, L. Jiang, *et al.*, “Point transformer,” in *IEEE/CVF Int. Conf. Comput. Vision*, pp. 259–268, 2021.
- [28] Z. Zhang, J. Sun, Y. Dai, *et al.*, “Vrnet: Learning the rectified virtual corresponding points for 3d point cloud registration,” *IEEE Trans. on Cir. and Sys. for Video Tech.*, vol. 32, pp. 4997–5010, 2022.