RoCNet: 3D robust registration of points clouds using deep learning

Karim Slimani, Brahim Tamadazte, Catherine Achard

RoCNet: 3D robust registration of points clouds using deep learning

Karim SLIMANI[1*], Brahim TAMADAZTE[1†] and Catherine ACHARD[1†]

[1*]ISIR, Sorbonne Université, CNRS UMR 7222, INSERM U1150, 4 place Jussieu, Paris, 75005, France.

*Corresponding author(s). E-mail(s): karim.slimani@isir.upmc.fr;
Contributing authors: brahim.tamadazte@cnrs.fr;
catherine.achard@sorbonne-universite.fr;
†These authors contributed equally to this work.

Abstract

This paper introduces a new method for 3D points cloud registration based on deep learning. The architecture is composed of three distinct blocs: (*i*) an encoder with a convolutional graph-based descriptor that encodes the immediate neighborhood 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 (source and target) 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 points clouds is calculated by RANSAC using the best scores of the correspondence matrix. We conduct experiments on the ModelNet40 and real-world Bunny datasets, and our proposed architecture shows promising results, outperforming state-of-the-art methods in most simulated configurations.

Keywords: points cloud Registration, Deep Learning, Attention Mechanism, Pose Estimation

1 Introduction

The development of new generations of 3-dimensional sensors, such as LIDAR and Kinect, has opened up a wide range of applications in computer vision, robotics, 3D printing, graphics, and more. Unlike traditional visual sensors (cameras), these advanced sensors provide a 3D representation of the observed scene as a points cloud, supplying crucial

information for analysing objects and environments. These innovations have resolved the persistent direct scene depth estimation issue that plagues traditional visual sensors. However, the field of view of 3D sensors is still relatively limited in many applications, including robotic navigation, autonomous vehicles, and manipulation tasks. As a result, several scientific questions have arisen, focusing on 3D mapping and reconstruction, object or scene recognition, pose estimation, and medical imaging. As a consequence, the need to register 3-dimensional information has become recurrent in various industrial applications and beyond. The registration process involves matching points between the source and target points clouds, eliminating outliers, and estimating the rigid transformation parameters that align one points cloud to the other. Traditional algorithms, such as Iterative Closest Point (ICP) [1], alternate between matching and aligning iterations. Recently, neural network-based techniques, such as [2–4], are increasingly used to address these problems. These techniques typically encode each point and its neighborhood through a learned descriptor, such as [5, 6]. Point matching is often performed based on similarities in the feature space. The estimation of the rigid transformation can be achieved in two different ways, either using end-to-end learning by integrating differentiable SVD [7] into the network as proposed in [3] or by applying a simple SVD or a RANSAC.

The learning-based approaches have made significant progress and have led to overcoming numerous limitations of iterative methods, such as converging to a local minimum conditioned by a correct initialization. Additionally, most iterative methods extract point-wise features by applying hand-crafted or learned descriptors. This makes these methods more sensitive to noise and outliers. The methods that apply RANSAC to overcome this problem need a huge number of iterations (e.g., 50, 000) to estimate the transformation correctly.

In this paper, we propose a new architecture (Fig. 1) called RoCNet, which includes three main blocks: 1) a descriptor composed of a convolutional graph-based network that encodes the immediate neighborhood 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 first contribution of this work lies mainly in the development of a new transformer module that efficiently encodes the variations of the normals to the surface of the 3D points using sinusoidal functions. Encoding the normals variations allows for improving aggregation and propagation of the local geometry attributes within and between the two points clouds through self-attention and cross-attention, respectively. This makes the 3D points matching problem more trivial and improves the matching performances compared to the traditional transformer module, as detailed in the ablation study (Section 5). The second contribution is the pipeline conception, which associates this new transformer with the DGCNN backbone and the Sinkhorn algorithm that allows extracting accurate hard correspondences. Encoding the spatial coordinates using DGCNN, combined with the advantages of estimation of the variation of normals, enables us to build a descriptor that is sufficiently discriminant and robust in the case of noisy data. This positively impacts the third module of the proposed architecture, which is the estimation of the rigid transformation using RANSAC. Indeed, this

reduces the number of iterations required by the RANSAC module, *i.e.*. 500 iterations for our method *versus* 50,000 iterations for other methods in the literature [8].

The RoCNet architecture was assessed using ModelNet40 dataset [2] in different favorable and unfavorable conditions and the Stanford Bunny dataset [9] on which the proposed method has not been trained. The comparison is done using the metrics commonly employed in the literature. Results show that RoCNet outperforms the best and most recent state-of-the-art methods, especially in unfavorable conditions (*i.e.* partial and noisy data) on both the matching task and the rigid transformation estimation one.

This paper is organised as follows. Section 2 describes the main families of 3D points cloud registration methods studied in the literature. The proposed methodology is detailed in Section 3. The experimental evaluation of our method in the context of the state-of-the-art is reported in Section 4, while Section 5 presents the ablation study of the RoCNet architecture.

2 RELATED WORK

2.1 3D points cloud Descriptors

The first iterative methods presented below directly use the 3D coordinates points as input to their system or handcrafted features like Fast Point Feature Histograms (FPFH) [10]. Since the rise of deep learning (DL), several methods have been developed to learn 3D points cloud descriptors. For instance, PointNet [11] or its extension PointNet++ [6] describes an unordered 3D points set for classification, segmentation, and registration tasks. Wang *et al.* [5] propose the Dynamic Graph CNN (DGCNN) descriptor based on a module that acts on graphs capturing semantic characteristics over potentially long distances. In [12], 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 points cloud.

2.2 Iterative Methods

The ICP [1] is probably the most popular method to address a points cloud registration problem. Given two sets of 3D points, the purpose is to minimize 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, the two-points clouds are randomly split into subsets on which a transformation is estimated. The final transformation is chosen among them using a predefined criterion, such as the number of inliers provided by each transformation.

2.3 Methods based on Matching Learning

Recent registration methods investigate DL architecture to match the points. For instance, Predator [13] is trained with three different weighted matching losses to be more robust to low partial overlap. Alternatively, in 3DFeat-Net [14], the architecture is trained to detect key points and predict discriminative descriptors using the triplet loss [15], while works in [16] combine two losses, one for the descriptor and the other one for the detector. All these methods use RANSAC on matched features to estimate the transformation parameters. MDGAT [12] learn the matching using a new loss inspired by the triplet function. [17]

propose an unsupervised matching approach by optimising the structural properties (*i.e.* bijectivity or approximate isometry) of functional maps [18]. In GeoTransformer [19], super-points are extracted from the source points clouds and described using the Geometric Transformer, which encodes intra-point-cloud and inter-point-cloud geometric structures.

2.4 Methods based on Transformation Learning

These methods are end-to-end and learn directly the transformation. For example, Deep Closest Point (DCP) [3] uses a descriptor based on DGCNN and a soft SVD. In PRNET [2], the matching is performed using an approximately differentiable version of Gumbel-Softmax, and the transformation is also obtained using an SVD. Another approach, proposed in [8], uses a differentiable nearest neighbor 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.

3 METHOD

Let us start by defining a common problem of 3D points cloud registration. Considering two-points clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$ such that: $\boldsymbol{X} = \{\boldsymbol{x}_1, ..., \boldsymbol{x}_i, ..., \boldsymbol{x}_M\} \subset \mathbb{R}^{3\times M}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_1, ..., \boldsymbol{y}_j, ..., \boldsymbol{y}_N\} \subset \mathbb{R}^{3\times N}$. It is assumed that the two sets at least partially overlap so that there are K pairs of matches between $\boldsymbol{X}$ and $\boldsymbol{Y}$, with $K \leq min(M, N)$. The two subsets containing the matching points in the first and second points clouds are defined by: $\bar{\mathbf{X}} \subset \mathbb{R}^{3\times K}$ and $\bar{\mathbf{Y}} \subset \mathbb{R}^{3\times K}$, respectively. Note that the set $\bar{\mathbf{Y}}$ is obtained by applying a rotation $\mathbf{R} \in SO(3)$ and a translation $\mathbf{t} \in \mathbb{R}^3$ of the set $\bar{\mathbf{X}}$. The rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$ define the 4×4 rigid transformation we seek.

3.1 Descriptor

One of the most fundamental components in a points 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 source and target sets of points $\boldsymbol{X}$ and $\boldsymbol{Y}$ 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 and normal-based descriptor, followed by an attention mechanism.

3.1.1 Geometrical-based descriptor

Different types of descriptors that learn local geometrical properties around each point were reported in the literature, such as PointNet [11], PointNet++ [6] or DGCNN descriptor [5]. We integrate DGCNN as a part of our descriptor because it better captures local geometric features of points 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 neighbors in the encoding space to build graphs that express the local geometric structure surrounding each point and then spread the information dynamically at a higher level (global encoding). The original architecture of DGCNN is followed by a two one-dimensional convolutions and a batch normalization. Let us denote $\boldsymbol{f}_i^X$, the extracted feature vector of dimension d for point $\boldsymbol{x}_i$.

3.1.2 Normal-based descriptor

The main idea of this descriptor is to encode the surface around each point better using the variation of the normals of points in the neighborhood: 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 normal angles in a neighborhood is informative about the type of surface.

The normals are estimated using Principal Component Analysis (PCA). Indeed, for each point $\boldsymbol{x}_i \in \boldsymbol{X}$, a local neighborhood subset of points $S_i = \left\{\boldsymbol{x}_j / \left\|\boldsymbol{x}_j - \boldsymbol{x}_i\right\|^2 \leq r\right\}$ is defined while delimiting the size of the set points with $|S_i| \leq K_{nn}$. r is the radius of a sphere centred on $\boldsymbol{x}_i$ 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 $\boldsymbol{n}_i$ as the vector associated with the smallest eigenvalue.

Since the PCA does not inherently determine the direction of the normal vector, we propose to address the sign ambiguity by using a new vector $\boldsymbol{z}_i$. It is colinear to $\boldsymbol{n}_i$ and is defined by ensuring that it points towards the side of the surface with the highest point density. This means the normal vector points away from sparse areas and towards denser surface areas. Similar to [8], we solve this ambiguity thanks to the system:

$$\boldsymbol{z}_i = \begin{cases} \boldsymbol{n}_i, & \text{if } \sum\limits_{\boldsymbol{x}_j \in S_i} \boldsymbol{n}_i^T \left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \geq 0 \\ -\boldsymbol{n}_i, & \text{otherwise} \end{cases} \tag{1}$$

Finally, we build the final encoding based on [19] and [20] using sinusoidal functions of different frequencies. Knowing the angle between the normals of two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ noted $\angle(\boldsymbol{z}_i, \boldsymbol{z}_j)$, the vector $\boldsymbol{g}_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ encoding the normals is given by:

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

where p is the current value index of $\boldsymbol{g}_{\boldsymbol{x}_i,\boldsymbol{x}_j}$, τ a coefficient that control the sensitivity to the variation in the normal orientations and d the dimension of the descriptor $\boldsymbol{g}_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ fixed to the same as the DGCNN output. A fully connected layer is then applied to $\boldsymbol{g}_{\boldsymbol{x}_i,\boldsymbol{x}_j}$ to obtain the final embedding $\boldsymbol{e}_{i,j}^X = \boldsymbol{g}_{\boldsymbol{x}_i,\boldsymbol{x}_j} \mathbf{W}_E^s$ where $\mathbf{W}_E^s \in \mathbb{R}^{d \times d}$ is a learned projection matrix.

3.1.3 Attention mechanism

A central aspect of recent descriptor design is the adoption of an attention mechanism that strategically accentuates certain features. For instance, SuperGlue [21] employs an attention graph-based module, sequentially layering *self-attention* and *cross-attention* mechanisms. The self-attention layers establish connections within a single points cloud, whereas the cross-attention layers form associations between each point in set $\boldsymbol{X}$ and all points in set $\boldsymbol{Y}$. Unlike SuperGlue [21] or MDGAT [12], which derives attention weights from encoding vectors, alternative approaches suggest the inclusion of information about local inter-point geometry at the mechanism's entry. For instance, [22] merges the 3D coordinates with each point's

descriptor, and GeoTransformer [19] utilises distances and angles between each point and its k nearest neighbors.

In our proposed methodology, we use four attention heads [20], each incorporating geometric self-attention within the sets $\boldsymbol{X}$ and $\boldsymbol{Y}$, augmented with their respective normal embedding $\boldsymbol{e}^X$ and $\boldsymbol{e}^Y$. This is followed by cross-attention between these point sets, with this pattern continuing for L iterations.

Self-attention. This layer type generates an attention-based feature $\bar{\mathbf{f}}_i$ for every point within a points cloud (either $\boldsymbol{X}$ or $\boldsymbol{Y}$), focusing on the relationships with all other points in the same cloud. Subsequently, the algorithm is elaborated specifically for a point $\boldsymbol{x}_i$ within $\boldsymbol{X}$; however, the same process applies to every point in both $\boldsymbol{X}$ and $\boldsymbol{Y}$. Consequently, an attention weight is derived for each query($\boldsymbol{x}_i$)/key($\boldsymbol{x}_j$) pair of points:

$$\alpha_{ij}^X = \underset{j}{softmax}\left(\frac{(\boldsymbol{f}_i^X \mathbf{W}_Q^s)(\boldsymbol{f}_j^X \mathbf{W}_K^s + \boldsymbol{e}_{i,j}^X \mathbf{W}_R^s)^T}{\sqrt{d}}\right) \tag{3}$$

where $\mathbf{W}_Q^s$, $\mathbf{W}_K^s$ and $\mathbf{W}_R^s \in \mathbb{R}^{d\times d}$ are the learned projection matrices for queries, keys and normal-based embeddings, d is the dimension of the features $\boldsymbol{f}_i^X$. These weights are used to learn focusing on the specific location of the descriptors and to obtain the final self-attention-based feature $\bar{\mathbf{f}}_i^X$:

$$\bar{\mathbf{f}}_i^X = \sum_{j=1} \alpha_{ij}\boldsymbol{v}_j \text{ with, } \boldsymbol{v}_j = \boldsymbol{f}_j^X \mathbf{W}_V^s \tag{4}$$

where $\mathbf{W}_V^s \in \mathbb{R}^{d\times d}$ is a learned matrix for the values $\boldsymbol{v}_j$.

Cross-attention.

$$\boldsymbol{h}_i^X = \sum_{j=1}^{|\boldsymbol{Y}|}\left(\underset{j}{softmax}\left(\frac{(\bar{\mathbf{f}}_i^X \mathbf{W}_Q^c)(\bar{\mathbf{f}}_j^Y \mathbf{W}_K^c)^T}{\sqrt{d}}\right)\right)\left(\bar{\mathbf{f}}_j^Y \mathbf{W}_V^c\right) \tag{5}$$

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

3.2 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 $x_i \in \boldsymbol{X}$ and $y_j \in \boldsymbol{Y}$:

$$\mathbf{C}_{i,j} = \boldsymbol{h}_i^{X\top} \boldsymbol{h}_j^Y \tag{6}$$

where $\boldsymbol{h}_i^X$ and $\boldsymbol{h}_j^Y$ are the final encoding of the points $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$ 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. The corresponding values are initialized to $z = 1$ and then learned by the network, such that the non-matched points will explicitly be assigned to the last dimension. We then employ the differentiable *Sinkhorn Algorithm* [23], which is widely used

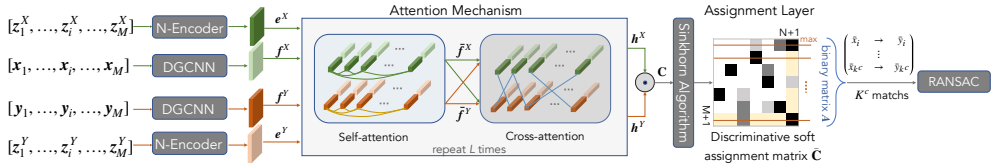


Fig. 1: Overview of the proposed RoCNet method. The details of the architecture are described in Section 4.2

in optimal transport and graph-matching problems. In our case, the latter technique is used to minimize the transport cost of the desired matched points between the source and the target points clouds. The rows and the columns of the scores matrix $\bar{\mathbf{C}}$ are iteratively normalized using equations 8 , leading to a soft assignment matrix [12, 21]that can be turned into a hard binary correspondences matrix in the test phase, using 9.

As all the previous steps are differentiable, the weights of the networks can be learned by introducing a loss function. To do so, we follow [12] and adopt the gap loss function (7), which allows enlarging the assignment scores difference between the true matches and the wrong matches. It is expressed as follows:

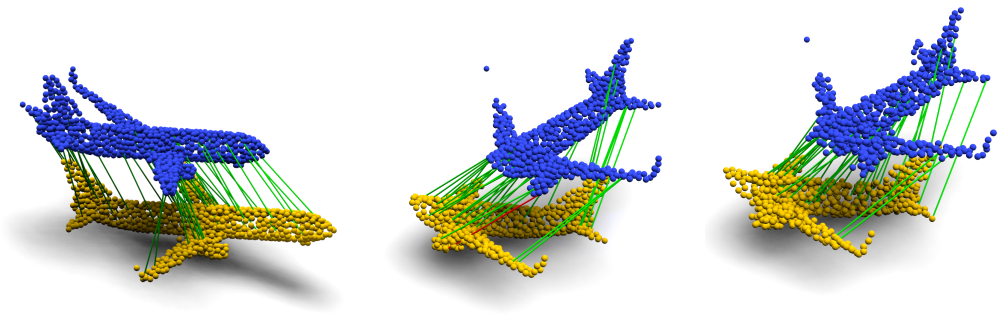
$$L_{Gap} = \sum_{i=1}^{M} \log\left(\sum_{n=1}^{N+1}[\max((-\log \bar{\mathbf{C}}_{i,\bar{i}} + \log \bar{\mathbf{C}}_{i,n} + \alpha), 0)] + 1\right) + \sum_{j=1}^{N} \log\left(\sum_{n=1}^{M+1}[\max((-\log \bar{\mathbf{C}}_{j,\bar{j}} + \log \bar{\mathbf{C}}_{n,j} + \alpha), 0)] + 1\right) \quad (7)$$

where α is a positive scalar set to 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 $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, respectively. The following Pytorch pseudo code 1 provides more details about the matching process. The $Sinkhorn\_iter$ function is defined by the two following operations:

$$\mathbf{C'}_{i,j} = \mathbf{C}_{i,j} - \log \sum_{j} \exp(\mathbf{C}_{i,j})$$
$$\mathbf{C}_{i,j} = \mathbf{C'}_{i,j} - \log \sum_{i} \exp(\mathbf{C'}_{i,j}) \quad (8)$$

Algorithm 1: Pseudo-code of the matching process

Data: N, M: Number of points in the source/target cloud, K: number of predicted correspondences
Data: $\boldsymbol{X}$, $\boldsymbol{Y}$: source and target points clouds
Data: $\boldsymbol{h}^X$, $\boldsymbol{h}^Y$: source and target features
Data: z: bin score
$\mathbf{C}$ = dot_product($\boldsymbol{h}^X$,$\boldsymbol{h}^Y$) ; `# [N, M]`
$\bar{\mathbf{C}}$ = augment_matrix($\mathbf{C}$, z) ; `# [N+1, M+1]`
for $i = 1$ **to** *20* **do**
  $\bar{\mathbf{C}} \leftarrow$ Sinkhorn_iter(i, $\bar{\mathbf{C}}$) ; `# Equation (8)`
end
if ***Training*** **then**
  Return *gap_loss*($\bar{\mathbf{C}}$) ; `# Training loss, equation (7)`
end
if ***Testing*** **then**
  $\mathbf{A}$ = **hard_assignment**($\bar{\mathbf{C}}$) ; `# Equations (9)-(10)`
  $\bar{\mathbf{X}}$, $\bar{\mathbf{Y}}$ = **reindex**($\mathbf{A}$, $\boldsymbol{X}$, $\boldsymbol{Y}$) ; `# re-index points clouds using the binary matrix`
  Return $\bar{\mathbf{X}}$, $\bar{\mathbf{Y}}$; `# Matched K points [K,3],[K,3]`
end



(a) (b) (c)

Fig. 2: Example of a performed 3D matching: (a) clean data, (b) partial overlap, and (c) noisy data and partial overlap. Green and red lines show correct and incorrect predicted matches respectively.

3.3 Pose Estimation

As detailed in the previous pseudocode 1, in the evaluation phase, we build a hard assignment binary matrix $\mathbf{A}$ thanks to the following algorithm:

$$\mathbf{A^1}_{i,j} = \begin{cases} 1 & \text{if } \bar{\mathbf{C}}_{i,j} = \max_n(\bar{\mathbf{C}}_{i,n}) \\ 0 & \text{otherwise,} \end{cases} \qquad \mathbf{A^2}_{i,j} = \begin{cases} 1 & \text{if } \bar{\mathbf{C}}_{j,i} = \max_n(\bar{\mathbf{C}}_{j,n}) \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

$$\mathbf{A} = \mathbf{A^1} \odot \mathbf{A^2} \tag{10}$$

where $\odot$ is the element-wise product. The matrix $\mathbf{A}$ gives 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 points clouds $\boldsymbol{X} \in \mathbb{R}^M$ and $\boldsymbol{Y} \in \mathbb{R}^N$ with the row and column indices of the non-zero values of the matrix $\mathbf{A}$. An example of a performed matching is depicted in Fig. 2. Once the sets of matched points are built, different techniques can be used to determine the rigid transformation. A classical SVD of the cross-covariance matrix between the centred subsets $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ is used in MDGAT [12], while DCP [3] 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 the matched features, as reported in [8, 13]. Our method uses RANSAC based on the 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.

4 EXPERIMENTS

4.1 Datasets and Parametrization

To assess RoCNet, we first use the synthetic ModelNet40 dataset [2] containing 40 CAD models of different objects. The database comprises 12,311 3D points clouds divided into 9,843 examples for training and 2,468 for testing. Each of these points clouds is scaled to fit inside a sphere of radius $r^s = 1$ m. For each source points cloud called $\boldsymbol{X}$, a target points cloud $\boldsymbol{Y}$ is created by applying a random rigid transformation with a rotation ranging from $0°$ to $45°$ around each axis and a translation ranging from 0 cm to 50 cm in each direction. Finally, a random permutation of the points is performed. In the end, a points 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 source 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:

- *Clean data*: synthetic data in ideal conditions.
- *Partial overlap*: To simulate partial occlusions, we sample the 768 nearest neighbors of a random point in both the source and target points clouds.
- *Partial overlap and 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 of the sub-sampled points clouds.

Table 1: Performances on clean and fully overlapping points clouds.

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

- *Real data*: The proposed method has also been evaluated using the real-world Stanford Bunny dataset [9], which consists of 10 points clouds of $\approx$ 100k points obtained by a range scanner.

4.2 Implementation Details and Metrics

RoCNet is implemented with PyTorch on a Nvidia Tesla V100-32G GPU and trained with the Adam optimiser [24] for 30 epochs on clean data and 80 epochs on 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 original architecture of DGCNN is used with a predicted feature size of 128. These features are fed into two one-dimensional convolutions with outputs of 64 and 96, respectively, using a kernel size of 1, followed by batch normalisation and a *ReLU* function. The parameters d, L are set to 96, 6, respectively, while the temperature τ equals to 0.25. Sinkhorn runs for 20 iterations with a dustbin score initialised to $z = 1$. Neighbors are collected under a radius $r = 30$ cm for normal estimation, and a maximum of $K_{nn} = 128$ points are used in the covariance matrix. To benchmark our RoCNet architecture against state-of-the-art methods, we opt for two metrics widely used in the literature: the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE). They are used to compute the difference in rotation and translation between the estimated transformation and the provided ground truth (GT). While the coefficient of determination R^2 is used to evaluate the correlation between these transformations.

4.3 Results

RoCNet is assessed in different configurations (*i.e.* favorable and unfavorable) on ModelNet40 and the Stanford Bunny dataset. In both cases, RoCNet's inference time is around 0.6 second. Our method is compared to the main DL-based methods of the state-of-the-art and with the traditional ICP approach. The recent work VRNet [25] has nicely summarised the performance of most of the related methods reported in the literature. We use these performances as a baseline and add performances of recently published methods such as WsDesc [8] and R-PointHop [26]. Note that in all the tables, $\mathbf{R}$ is given in degrees, and $\mathbf{t}$ in meters, while the best results are highlighted in bold and the second ones are underlined.

Table 2: Performances on clean and partially overlapping points clouds.

Method	RMSE($\mathbf{R}$)	MAE($\mathbf{R}$)	$R^2(\mathbf{R})$	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)	$R^2(\mathbf{t})$
ICP'92	33.683	25.045	-5.696	0.293	0.2500	-0.037
PTLK'19	16.735	07.550	-0.654	0.045	0.0250	0.975
DCP-V2'19	06.709	04.448	0.732	0.027	0.0200	0.991
PRNET'19	03.199	01.454	0.939	0.016	0.0100	0.997
R-PointHop'22	01.660	00.350	-	0.014	0.0008	-
VRNet'22	00.982	00.496	-	0.006	0.0039	-
WsDesc'22	01.187	00.975	0.992	0.008	0.0070	0.999
Ours	**00.412**	**00.133**	**1.000**	**0.002**	**0.0002**	**1.000**

4.3.1 Model trained on clean data

The first configuration consists of the evaluation of RoCNet when the model is trained on clean data and with no occlusions. Table 1 provides the results. RoCNet outperforms the other methods in rotation with an improvement of a dozen percent for the RMSE and MAE *versus* the second best method VRNet [25]. However, VRNet remains the best in translation, although the difference is slight compared to RoCNet, specifically for MAE($\mathbf{t}$) where our method is second.

4.3.2 Model trained on clean and partial overlap data

In this section, we evaluate the behavior of the method when only a part of the points is shared by the two points clouds to be aligned. As in real applications, points cloud registration problems generally aim to align data from different sensors, there is often partial occlusion between the different points clouds. This requires the calculation of non-bijective correspondences. This experiment is, therefore, fundamental to evaluate our method and assess its suitability for real-world applications. Furthermore, this experiment highlights the importance of the matching process, particularly the extension of the similarity matrix fed to the Sinkhorn algorithm to handle the points without matches in the two-points clouds. From Table 2, it can be seen that RoCNet significantly outperforms the other methods in all metrics. Our method reduces the registration error by about half compared to the second-ranked methods, *i.e.* VRNet, and R-PointHop for both MSE and MAE. Note that RoCNet achieves the maximum allowable determination score $R^2 = 1$ for both translation and rotation.

4.3.3 Model trained on noisy and partial overlap data

In line with the previous section, we propose to lead this last configuration on ModelNet40, where a Gaussian noise has been added to the partial points clouds to simulate the presence of noise during the data acquisition for real-world applications. As can be seen in Table 3, RoCNet outperforms the other methods on all metrics in both rotation and translation. RoCNet allows significant enhancement of the registration error, ranging from two-thirds to one-quarter, compared to the method ranked second, *i.e.* WsDesc, and even more so in comparison to VRNet. This can be explained by the robustness of proposed normal encoding module to partial occlusions or noise or both at the same time. These strong performances under unfavorable conditions suggest interesting avenues for future work and the RoCNet use on concrete tasks such as medical imaging and robotics applications.

Table 3: Performances on noisy and partially overlapping points clouds.

Method	RMSE($\mathbf{R}$)	MAE($\mathbf{R}$)	$R^2(\mathbf{R})$	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)	$R^2(\mathbf{t})$
ICP'92	33.067	25.564	-6.252	0.294	0.250	-0.045
PTLK'19	19.939	9.076	-1.343	0.057	0.032	0.960
DCP-V2'19	06.883	4.534	0.718	0.028	0.021	0.991
PRNET'19	04.323	2.051	0.889	0.017	0.012	0.995
VRNet'22	03.615	1.637	-	0.010	0.006	-
WsDesc'22	03.500	0.759	0.928	0.006	0.004	**0.999**
Ours	**01.810**	**0.620**	**0.963**	**0.004**	**0.003**	**0.999**

Table 4: Performances on the Bunny Stanford dataset for the models trained on ModelNet40

Method	RMSE($\mathbf{R}$)	MAE($\mathbf{R}$)	RMSE($\mathbf{t}$)	MAE($\mathbf{t}$)
ICP'92	13.32	10.72	0.0492	0.0242
FGR'16	1.99	1.49	0.1993	0.1658
DCP'19	6.44	4.78	0.0406	0.0374
R-PointHop'22	1.49	1.09	0.0361	**0.0269**
Ours	**0.99**	**0.83**	**0.0338**	0.0288

4.3.4 Model tested on the Stanford Bunny dataset

In this final experiment, we propose to test the generalization ability of RoCNet to real-word scans. Thus, we follow [26] and evaluate the method on the Stanford Bunny dataset by applying a random rigid transformation to the source points clouds and sampling 2048 points out of the initial 100K points to estimate the rigid transformation using the model trained on ModelNET40. The results reported in Table 4 show that RoCNet is also efficient on real-world unseen data, outperforming state-of-the-art methods in three out of four of the used metrics. These performances on unseen points clouds highlight a satisfactory generalization capability of the method. This feature is crucial for industrial applications, as the learning models are frequently used on data that may be different from those used in the learning phase due to the lack of rich datasets. Especially in medical imaging, large training sets are often obtained by applying data augmentation to smaller initial real data. [27]

4.3.5 Additional results

Finally, the performance of RoCNet and its ranking in the context of wider state-of-the-art, including the eleven best methods, is presented in Fig. 3. It can be highlighted that our method outperforms all the methods when considering simultaneously performances in rotation and translation, both on clean data (Fig. 3(a)) and on noisy data (Fig. 3(b)). Furthermore, to be able to assess visually the robustness of the proposed method, we perform different registrations by progressively decreasing (from 95% to 50%) the rate of shared points between $\boldsymbol{X}$ and $\boldsymbol{Y}$. Figure 4 depicts the obtained results for one object. The first row shows the initial positions of source and target points clouds $\boldsymbol{X}$ and $\boldsymbol{Y}$; the second row shows the performed registrations, and the third shows the ground truth ones. As can be seen, RoCNet can register data even with 50% points occluded without much difficulty. On the other hand, the method shows its limits for objects with perfect symmetry in low overlapping cases.

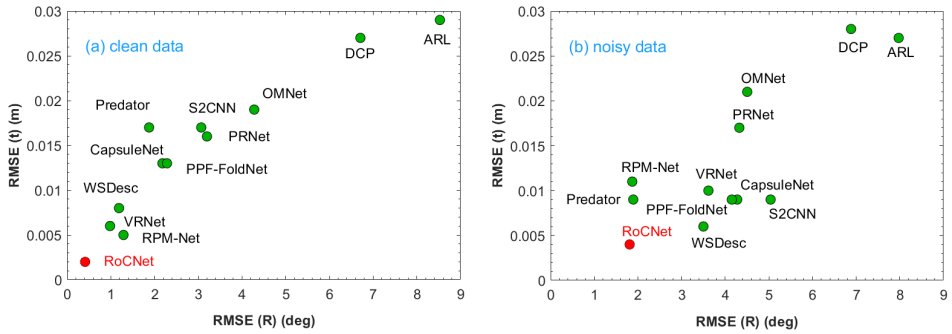


Fig. 3: RoCNet against the most relevant state-of-the-art methods. (a) clean data, and (b) noisy data.

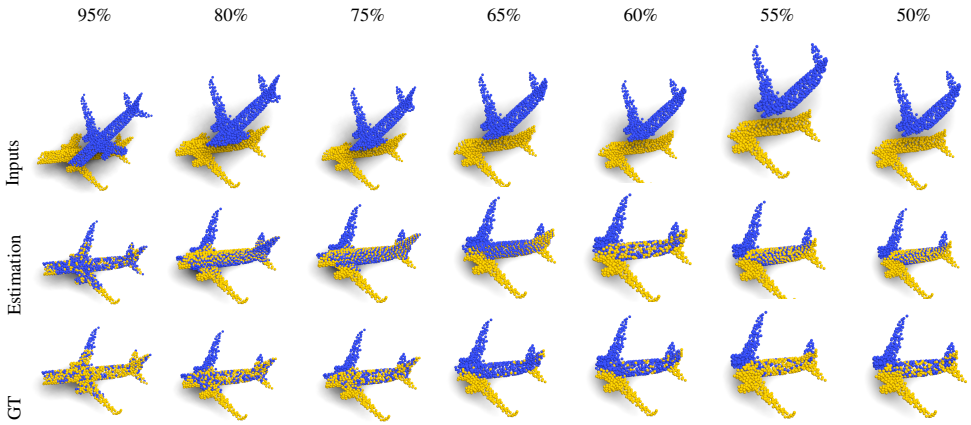


Fig. 4: Robustness of RoCNet against partial occlusions.

5 ABLATION STUDY

We conduct ablation studies on the proposed architecture's three main blocks: the descriptor, the transformer, and the transformation's RANSAC-based estimation.

5.1 Descriptor Ablation

To study the impact of our descriptor combining DGCNN and our normal encoding transformer, we compared the point matching performances of our architecture to MDGAT [12]. For a proper comparison, both methods are trained in the same configurations and with the same number of epochs. Two types of data are used: 1) clean data and 2) noisy data, both with partial overlap. We use the following metrics: Precision (**P**), Accuracy (**A**), Recall (**R**), F1-score (**F1**) False Positive Rate **FPR** and Inlier Ratio **IR**. Table 5 gives an insight into the ablation study of the descriptor. MDGAT and RoCNet use all the points as inputs (*i.e.* 768).

Table 5: Performances assessment in a matching challenge with partial overlap data (**P**: Precision, **A**: Accuracy, **R**: Recall, **F1**: F1-score and **FPR:** False Positive Rate and **IR:** Inliers Ratio.

Method	clean data						noisy data					
	P	**A**	**R**	**F1**	**FPR**	**IR**	**P**	**A**	**R**	**F1**	**FPR**	**IR**
MDGAT	94.6	94.5	94.5	94.6	0.14	92.0	78.4	78.5	77.9	78.1	0.13	93.6
Ours	**98.0**	**97.2**	**97.6**	**97.8**	**0.07**	**94.8**	**85.8**	**85.9**	**85.5**	**85.7**	**0.08**	**96.2**

Table 6: Performances assessment in a matching challenge with clean data and partial overlap. MDGAT* is a version which uses 256 key points instead of the whole points [12]

Descriptor	**Attention**	**P**	**A**	**R**	**F1**
MDGAT*	MDGAT	**98.1**	93.7	93.4	<u>95.7</u>
MDGAT	MDGAT	94.6	94.5	94.5	94.6
MDGAT	**Ours**	92.1	84.6	84.8	88.3
Ours	MDGAT	96.6	<u>96.3</u>	<u>96.3</u>	95.4
Ours	Transformer w/o normals	79.8	80.2	80.5	80.1
Ours	**Ours**	<u>98.0</u>	**97.2**	**97.6**	**97.8**

It can be underlined that our descriptor outperforms MDGAT, and the improvement is significant when it concerns partial noisy data. In terms of computational cost, our method estimates the transformation is 0.6 seconds, similar to WsDesc [8] which requires 0.8 seconds, when it takes only 0.2 seconds for MDGAT [12], which is reasonably acceptable for 3D point registration problem of a thousand points. RoCNet and the two other methods (using the authors' codes) were executed using the same computation support, *i.e.* an RTX 5000 24G GPU.

5.2 DGCNN and Transformer Ablation

To emphasize the significance of integrating DGCNN with our novel normal encoding, we initially suggest substituting the feature extraction performed by DGCNN with the encoding from MDGAT and then merging this with our transformer. Secondly, we keep DGCNN and replace our transformer with MDGAT's attention mechanism. Finally, we propose to associate DGCNN with a classical transformer without our normal encoding, as in [21]. Table 6 reports the obtained results showing that RoCNet architecture is more relevant on three of the four used metrics emphasizing in particular a significant contribution (about 15%) for the proposed transformer compared to a classical attention mechanism.

5.3 RANSAC Ablation

The last ablation study compares the contribution of an SVD *versus* a RANSAC to estimate the rigid transformation when the matching is performed. Table 7 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 the RANSAC approach outperforms slightly the SVD one.

Table 7: SVD *versus* RANSAC for transformation estimation in case of full, partial, and noisy data.

Method	RMSE($\mathbf{R}$)			MAE($\mathbf{R}$)			RMSE($\mathbf{t}$)			MAE($\mathbf{t}$)		
	full	noisy	partial	full	noisy	partial	full	noisy	partial	full	noisy	partial
SVD	3.94	1.94	4.21	0.124	**0.484**	0.168	**0.0005**	**0.0017**	**0.001**	**0.0001**	**0.0018**	**0.0001**
RANSAC	**0.06**	**1.92**	**0.40**	**0.010**	0.555	**0.133**	**0.0005**	0.0026	0.002	**0.0001**	**0.0018**	0.0002

6 Discussion and limitations

The multiple assessments of RoCNet showed very interesting results on fully and partially overlapping points clouds, whether with clean or noisy data. It outperforms the state-of-the-art methods in almost all metrics and scenarios as detailed in Tables 5 and 6. Furthermore, Table 4 showed the ability of RoCNet generalizability for unseen data (*e.g.* Stanford Bunny dataset), as it outperformed the best-reported methods in the literature in three of the four metrics. In return, RoCNet has certain limitations, particularly when it comes to handling larger databases, *i.e.* those containing points clouds of several tens of thousands of points such as KITTI [28]. This is mainly due to our normal encoder transformer, which requires a huge amount of memory, especially in the training phase. This is a relatively common problem with literature methods in the case of the KITTI database. A naive solution is to under-sample the input points clouds, which can significantly reduce the method's performance. Indeed, the normal estimation can be impaired by a lack of local information about the geometry in the neighborhood of each point. A more sophisticated solution has been used in Cofinet [29] and GeoTransformer [19], which extract super-points while simultaneously encoding their local geometry to reduce the size of the input to a few hundred points instead of thousands. In [30], the authors suggest modifying the transformer block by replacing the *softmax* operation with a *cosine* operation, allowing a reduction in computational cost for an input of size N, from $O(N^2)$ to $O(N)$, while maintaining good performance.

7 CONCLUSION

The development of advanced points cloud registration algorithms is crucial in various applications, such as robotics, autonomous driving, and medical imaging. This paper dealt with the investigation of an accurate and robust learning-based registration method. The proposed architecture is composed of three main blocks: 1) the newly designed descriptor, which encodes the neighborhood 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 best matches from the correspondence matrix. The proposed architecture has been evaluated using the ModelNet40 dataset in different favorable and unfavorable configurations. Our method has been demonstrated to outperform the related state-of-the-art algorithms, especially in unfavorable conditions, *e.g.* with noisy data and partial occlusions. Moreover, the new proposed transformer, which encodes normal surface variations, can be extended to other 3D points cloud tasks such as object classification and segmentation and can be easily combined with existing 3D feature extractors. RoCNet has been evaluated on synthetic data augmented with noise and occlusion to simulate real

ones and has proved its potential. One limitation is that DGCNN features combined with the proposed transformer lead to an important memory load for large-size points clouds.

A relevant solution to overcome this limitation seems to be the application of a learning keypoint extractor method to reduce the input size. Otherwise, adopting a multi-scale matching strategy by uniformly down-sampling the input and thus running the transformer at a lower points cloud resolution for super-point matching may also help to address this issue [19]. Further perspectives on this work are to expand our approach to cross-modality (*e.g.* 2D-3D) registration and non-rigid correspondence estimation. We will also examine the evaluation of our descriptor in other contexts, notably object segmentation and classification.

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

References

[1] Besl, P.J., McKay, N.D.: Method for registration of 3-d shapes. In: Sensor Fusion IV: Control Paradigms and Data Structures, vol. 1611, pp. 586–606 (1992)

[2] Wang, Y., Solomon, J.M.: Prnet: Self-supervised learning for partial-to-partial registration. Adv. Neural. Inf. Process. Syst. **32** (2019)

[3] Wang, Y., Solomon, J.M.: Deep closest point: Learning representations for point cloud registration. In: IEEE Int. Conf. on Comput. Vision (2019)

[4] Aoki, Y., Goforth, H., Srivatsan, R.A., *et al.*: Pointnetlk: Robust & efficient point cloud registration using pointnet. In: IEEE/CVF Conf. Comput. Vision Pattern Recognit., pp. 7163–7172 (2019)

[5] Wang, Y., Sun, Y., et al.: Dynamic graph cnn for learning on point clouds. ACM Trans. on Grap. (2019)

[6] Qi, C.R., Yi, L., Su, H., et al.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Adv. Neural. Inf. Process. Syst. **30** (2017)

[7] Papadopoulo, T., Lourakis, M.L.: Estimating the jacobian of the singular value decomposition: Theory and applications. PhD thesis, Inria (2000)

[8] Li, L., Fu, H., Ovsjanikov, M.: Wsdesc: Weakly supervised 3d local descriptor learning for point cloud registration. IEEE Trans. Vis. Comput. Graph. (2022)

[9] Turk, G., Levoy, M.: Zippered polygon meshes from range images. In: Conf. on Comp. Grap. and Inter. Tech., pp. 311–318 (1994)

[10] Rusu, R.B., Blodow, N., *et al.*: Fast point feature histograms (fpfh) for 3d registration. In: IEEE Int. Conf. on Rob. and Auto., pp. 3212–3217 (2009)

[11] Qi, C.R., Su, H., *et al.*: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Conf. Comput. Vision Pattern Recognit., pp. 652–660 (2017)

[12] Shi, C., Chen, X., Huang, K., *et al.*: Keypoint matching for point cloud registration using multiplex dynamic graph attention networks. IEEE Rob. and Auto. Let. **6**, 8221–8228 (2021)

[13] Huang, S., Gojcic, Z., Usvyatsov, M., *et al.*: Predator: Registration of 3d point clouds with low overlap. In: IEEE/CVF Conf. Comput. Vision Pattern Recognit., pp. 4267–4276 (2021)

[14] Yew, Z.J., Lee, G.H.: 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In: Euro. Confe. on Comput. Vision (2018)

[15] Khoury, M. and Zhou, Q.-Y., and others: Learning compact geometric features. In: IEEE Int. Conf. Comput. Vision, pp. 153–161 (2017)

[16] Bai, X., Luo, Z., Zhou, L., *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)

[17] Roufosse, J.-M. and Sharma, and others: Unsupervised deep learning for structured shape matching. In: IEEE/CVF Int. Conf. Comput. Vision, pp. 1617–1627 (2019)

[18] Ovsjanikov, M., Ben-Chen, M., Solomon, J., Butscher, A., Guibas, L.: Functional maps: a flexible representation of maps between shapes. ACM Trans. on Grap. **31**, 1–11 (2012)

[19] Qin, Z., Yu, H., Wang, C., *et al.*: Geometric transformer for fast and robust point cloud registration. In: IEEE/CV Conf. Comput. Vision Pattern Recognit., pp. 11143–11152 (2022)

[20] Vaswani, A., Shazeer, N., et al.: Attention is all you need. Adv. Neural. Inf. Process. Syst. **30** (2017)

[21] Sarlin, P.-E., DeTone, D., *et al.*: SuperGlue: Learning feature matching with graph neural networks. In: Conf. Comput. Vision Pattern Recognit. (2020)

[22] Zhao, H., Jiang, L., *et al.*: Point transformer. In: IEEE/CVF Int. Conf. Comput. Vision, pp. 259–268 (2021)

[23] Sinkhorn, R., Knopp, P.: Concerning nonnegative matrices and doubly stochastic matrices. Pacific J. of Math. **21**, 343–348 (1967)

[24] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[25] Zhang, Z., Sun, J., Dai, Y., *et al.*: Vrnet: Learning the rectified virtual corresponding points for 3d point cloud registration. IEEE Trans. on Cir. and Sys. for Video Tech. **32**, 4997–5010 (2022)

[26] Kadam, P., Zhang, M., Liu, S., *et al.*: R-pointhop: A green, accurate, and unsupervised

point cloud registration method. IEEE Trans. on Ima. Process. **31**, 2710–2725 (2022)

[27] Hu, X., Nguyen, A., Baena, F.R.: Occlusion-robust visual markerless bone tracking for computer-assisted orthopedic surgery. IEEE Transactions on Instrumentation and Measurement **71**, 1–11 (2021)

[28] Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conf. Comput. Vision Pattern Recognit., pp. 3354–3361 (2012)

[29] Yu, H., Li, F., Saleh, M., Busam, B., Ilic, S.: Cofinet: Reliable coarse-to-fine correspondences for robust pointcloud registration. Advances in Neural Information Processing Systems **34**, 23872–23884 (2021)

[30] Qin, Z., Sun, W., Deng, H., Li, D., Wei, Y., Lv, B., Yan, J., Kong, L., Zhong, Y.: cosformer: Rethinking softmax in attention. arXiv preprint arXiv:2202.08791 (2022)