

# 3DRegNet: A Deep Neural Network for 3D Point Registration

G. Dias Pais<sup>1</sup>, Srikumar Ramalingam<sup>2</sup>, Venu Madhav Govindu<sup>3</sup>,  
Jacinto C. Nascimento<sup>1</sup>, Rama Chellappa<sup>4</sup>, and Pedro Miraldo<sup>1</sup>

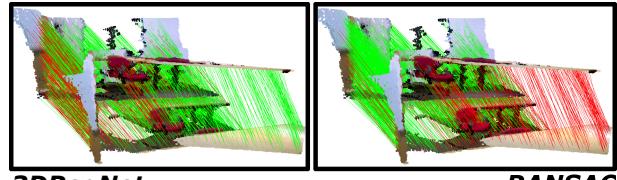
<sup>1</sup>Instituto Superior Técnico, Lisboa    <sup>2</sup>Google Research, NY  
<sup>3</sup>Indian Institute of Science, Bengaluru    <sup>4</sup>University of Maryland, College Park

## Abstract

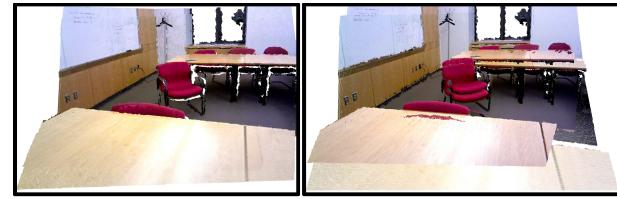
We present 3DRegNet, a novel deep learning architecture for the registration of 3D scans. Given a set of 3D point correspondences, we build a deep neural network to address the following two challenges: (i) classification of the point correspondences into inliers/outliers, and (ii) regression of the motion parameters that align the scans into a common reference frame. With regard to regression, we present two alternative approaches: (i) a Deep Neural Network (DNN) registration and (ii) a Procrustes approach using SVD to estimate the transformation. Our correspondence-based approach achieves a higher speedup compared to competing baselines. We further propose the use of a refinement network, which consists of a smaller 3DRegNet as a refinement to improve the accuracy of the registration. Extensive experiments on two challenging datasets demonstrate that we outperform other methods and achieve state-of-the-art results. The code is available at <https://github.com/3DVisionISR/3DRegNet>.

## 1. Introduction

We address the problem of 3D registration, which is one of the classical and fundamental problems in geometrical computer vision due to its wide variety of vision, robotics, and medical applications. In 3D registration, the 6 Degrees of Freedom (DoF) motion parameters between two scans are computed given noisy (outliers) point correspondences. The standard approach is to use minimal solvers that employ three-point correspondences (see [48, 39]) in a RANSAC [17] framework, followed by refinement techniques such as the Iterative Closest Point (ICP) [6]. In this paper, we investigate if the registration problem can be solved using a deep neural methodology. Specifically, we study if deep learning methods can bring any complementary advantages over classical registration methods. In particular, we wish to achieve speedup without compromis-



(a) Inliers/outliers classification using the proposed 3DRegNet vs. a RANSAC approach. Green and red colors indicate the inliers and outliers, respectively.



(b) Results of the estimation of the transformation that aligns two point clouds, 3DRegNet vs. the current state-of-the-art Fast Global Registration method (FGR) [65].

Figure 1: Given a set of 3D point correspondences from two scans with outliers, our proposed network 3DRegNet simultaneously classifies the point correspondences into inliers and outliers (see (a)), and also computes the transformation (rotation, translation) for the alignment of the scans (see (b)). 3DRegNet is significantly faster and outperforms other standard geometric methods.

ing the registration accuracy in the presence of outliers. In other words, the challenge is not in pose given point correspondences, but how can efficiently handle the outliers. Figure 1 illustrates the main goals of this paper. Figure 1(a) depicts the classification of noisy point correspondences into inliers and outliers using 3DRegNet (left) and RANSAC (right) for aligning two scans. Figure 1(b) shows the estimation of the transformation that aligns two point clouds using the proposed 3DRegNet (left) and current state-of-the-art FGR [65] (right).

In Fig. 2(a), we show our proposed architecture with two sub-blocks: classification and registration. The for-

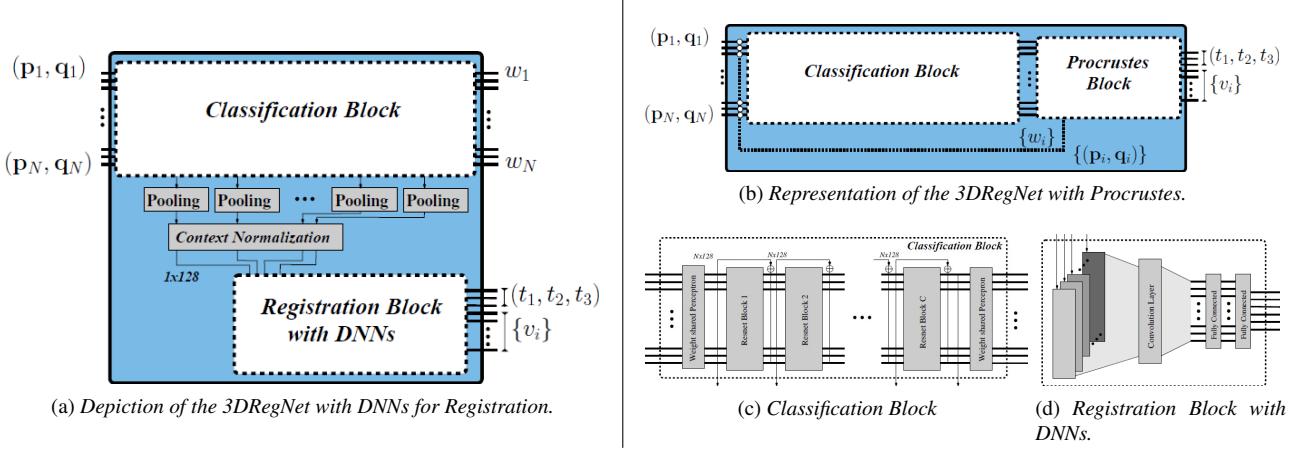


Figure 2: Two proposed architectures. (a) shows our first proposal with the classification and the registration blocks. (b) shows our second proposal with the same classification block as in the first one, but with a different registration block based on the differential Procrustes method. (c) classification block using C ResNets, which receives a set of point correspondences as input and outputs weights classifying them as inliers/outliers. (d) registration block (used in the architecture shown in (a)) that is obtained from the features of classification block and where its parameters are obtained through a DNN.

mer takes a set of noisy point correspondences between two scans and produces weight (confidence) parameters that indicate whether a given point correspondence is an inlier or an outlier. The latter directly produces the 6 DoF motion parameters for the alignment of two 3D scans. Our main contributions are as follows. We present a novel deep neural network architecture for solving the problem of 3D scan registration, with the possibility of a refinement network that can fine-tune the results. While achieving a significant speedup, our method achieves state-of-the-art registration performance.

## 2. Related Work

The ICP is widely considered as the gold standard approach to solve point cloud registration [6, 44]. However since ICP often gets stuck in local minima, other approaches have proposed extensions or generalizations that achieve both efficiency and robustness, e.g., [49, 40, 41, 58, 20, 31, 43, 29]. The 3D registration can also be viewed as a non-rigid problem motivating several works [67, 5, 51, 34]. A survey of rigid and non-rigid registration of 3D point clouds is available in [52]. An optimal least-squares solution can be obtained using methods such as [53, 49, 40, 38, 24, 57, 65, 7, 36]. Many of these methods require either a good initialization or identification of inliers using RANSAC. Subsequently, the optimal pose is estimated using only the selected inliers. In contrast to the above strategies, we focus on jointly solving (i) the inlier correspondences and (ii) the estimation of the transformation parameters without requiring an initialization. We propose a unified deep learning framework to address both challenges mentioned above.

Deep learning has been used to solve 3D registration problems in diverse contexts [14, 15, 23]. PointNet is a Deep Neural Network (DNN) that produces classification and segmentation results for unordered point clouds [46]. It strives to achieve results that are invariant to the order of points, rotations, and translations. To achieve invariance, PointNet uses several Multi-Layer Perceptrons (MLP) individually on different points, and then use a symmetric function on top of the outputs from the MLPs. PointNetLK builds on PointNet and proposes a DNN loop scheme to compute the 3D point cloud alignment [2]. In [54], authors derive an alternative approach to ICP, i.e., alternating between finding the closest points and computing the 3D registration. The proposed method focuses on finding the closest points at each step; the registration is computed with Procrustes. [32] proposes a network that initially generates correspondences based on learned matched probabilities and then creates an aligned point cloud. In [56, 50, 25, 55], other methods are proposed for object detection and pose estimation on point clouds with 3D bounding boxes. In contrast to these methods, our registration is obtained from pre-computed 3D point matches, such as [47, 61], instead of using the original point clouds and thereby achieving considerable speedup.

A well-known approach is to use point feature histograms as features for describing a 3D point [47]. The matching of 3D points can also be achieved by extracting features using convolutional neural networks [61, 12, 59, 15, 13, 19]. Some methods directly extract 3D features from the point clouds that are invariant to the 3D environment (spherical CNNs) [10, 16]. A deep network has been designed recently for computing the pose for direct image

to image registration [21]. Using graph convolutional networks and cycle consistency losses, one can train an image matching algorithm in an unsupervised manner [45].

In [60], a deep learning method for classifying 2D point correspondences into inliers/outliers is proposed. The regression of the Essential Matrix is computed separately using eigendecomposition and the inlier correspondences. The input of the network is only pixel coordinates instead of original images allowing for faster inference. The method was improved in [62], by proposing hierarchically extracted and aggregated local correspondences. The method is also insensitive to the order of correspondences. In [11], an eigendecomposition-free approach was introduced to train a deep network whose loss depends on the eigenvector corresponding to a zero eigenvalue of a matrix predicted by the network. This was also applied to 2D outlier removal. In [33], a DNN classifier was trained on a general match representation based on putative match through exploiting the consensus of local neighborhood structures and a nearest neighbor strategy. In contrast with the methods mentioned above, our technique aims at getting an end-to-end solution to the registration and outlier/inlier classification from matches of 3D point correspondences.

For 3D reconstruction using a large collection of scans, rotation averaging can be used to improve the pairwise relative pose estimates using robust methods [8]. Recently, it was shown that it would be possible to utilize deep neural networks to compute the weights for different pairwise relative pose estimates [26]. The work in [64] focuses on learning 3D match of features in three views. Our paper focuses on the problem of pairwise registration of 3D scans.

### 3. Problem Statement

Given a set of  $N$  3D point correspondences  $\{(\mathbf{p}_i, \mathbf{q}_i)\}_{i=1}^N$ , where  $\mathbf{p}_i \in \mathbb{R}^3$ ,  $\mathbf{q}_i \in \mathbb{R}^3$  are the 3D points in the first and second scan respectively, our goal is to compute the transformation parameters (rotation matrix  $\mathbf{R} \in \mathcal{SO}(3)$  and translation vector  $\mathbf{t} \in \mathbb{R}^3$ ) as follows

$$\mathbf{R}^*, \mathbf{t}^* = \underset{\mathbf{R} \in \mathcal{SO}(3), \mathbf{t} \in \mathbb{R}^3}{\operatorname{argmin}} \sum_{n=1}^N \rho(\mathbf{q}_n, \mathbf{R}\mathbf{p}_n + \mathbf{t}), \quad (1)$$

where  $\rho(\mathbf{a}, \mathbf{b})$  is some distance metric. The problem addressed in this work is shown in Fig. 1. The input consists of  $N$  point correspondences, and the output consists of  $N + M + 3$  variables. Specifically, the first  $N$  output variables form a weight vector  $W := \{w_i\}_{i=1}^N$ , where  $w_i \in [0, 1]$  represents the confidence that the  $i$ -th correspondence pair  $(\mathbf{p}_i, \mathbf{q}_i)$  is an inlier. By comparing  $w_i$  with a threshold  $\mathcal{T}$ , i.e.,  $w_i \geq \mathcal{T}$  we can classify all the input correspondences into inliers/outliers. The next  $M$  output variables represent the rotation parameters, i.e.,  $(v_1, \dots, v_M)$ . The remaining three parameters  $(t_1, t_2, t_3)$

denote the translation. Although a 3D rotation has exactly 3 degrees of freedom, there are different possible parameterizations. As shown in [66], choosing the correct parameterization for the rotation is essential for the overall performance of these approaches. Previous methods use over-parameterization for the rotation (e.g., PoseNet [27] uses four parameter-quaternions for representing the rotation, while deep PnP [11] uses nine parameters). We study the different parameterizations of the rotation and evaluate their performance.

### 4. 3DRegNet

The proposed 3DRegNet architecture is shown in Fig. 2 with two blocks for classification and registration. We have two possible approaches for the registration block, either using DNNs or differentiable Procrustes. This choice does not affect the loss functions presented in Sec. 4.1.

**Classification:** The classification block (see the respective block in Fig. 2(c)) follows the ideas of previous works [46, 60, 11, 62]. The input is a 6-tuples set of 3D point correspondences given by  $\{(\mathbf{p}_i, \mathbf{q}_i)\}_{i=1}^N$  between the two scans.

Each 3D point correspondence is processed by a fully connected layer with 128 ReLU activation functions. There is a weight sharing for each of the individual  $N$  point correspondences, and the output is of dimension  $N \times 128$ , where we generate 128 dimensional features from every point correspondence. The  $N \times 128$  output is then passed through  $C$  deep ResNets [22], with weight-shared fully connected layers instead of convolutional layers. At the end, we use another fully connected layer with ReLU ( $\text{ReLU}(x) = \max(0, x)$ ) followed by tanh ( $\text{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \in (-1, 1)$ ) units to produce the weights in the range  $w_i \in [0, 1]$ . The number  $C$  of deep ResNets depends on the complexity of the transformation to be estimated as is discussed in Sec. 5.

**Registration with DNNs:** The input to this block are the features extracted from the point correspondences. As shown in Fig. 2(d), we use pooling to extract meaningful features of dimensions  $128 \times 1$  from each layer of the classification block. We extract features at  $C + 1$  stages of the classification, i.e., the first one is extracted before the first ResNet and the last one is extracted after the  $C$ -th ResNet. Based on our experiments, max-pooling performed the best in comparison with other choices such as average pooling. After the pooling is completed, we apply context normalization, as introduced in [60], and concatenate the  $C + 1$  feature maps (see Figs. 2(a) and 2(d)). This process normalizes the features and it helps to extract the necessary and fixed number of features to obtain the transformation at the end of the registration block (that should be independent of  $N$ ). The features from the context normalization is of size  $(C + 1) \times 128$ , which is then passed on to a con-

volutional layer, with 8 channels. Each filter passes a 3-by-3 patch with a stride of 2 for the column and of 1 for the row. The output of the convolution is then injected in two fully connected layers with 256 filters each, with ReLU between the layers, that generate the output of  $M+3$  variables:  $\mathbf{v} = (v_1, \dots, v_M)$  and  $\mathbf{t} = (t_1, t_2, t_3)$ .

**Registration with Differentiable Procrustes:** In contrast to the previous block, we present another alternative to perform the registration. Now, we obtain the desired transformation through the point correspondences (see Fig. 2(b)). We filter out the outliers and compute the centroid of the inliers, using this as the origin. Since the centroids of the point clouds are now at the origin, we only need to obtain the rotation between them. Note that the outlier filtering and the shift in the centroids can be seen as intermediate layers, thereby allowing end-to-end training for both classification and pose computation. This rotation is computed from the SVD of the matrix  $\mathbf{M} = \mathbf{U}\Sigma\mathbf{V}^T$  [3], where  $\mathbf{M} \in \mathbb{R}^{3 \times 3}$  is as follows:

$$\mathbf{M} = \sum_{i \in \mathcal{I}} w_i \mathbf{p}_i \mathbf{q}_i^T, \quad (2)$$

where  $\mathcal{I}$  represents the set of inliers obtained from the classification block. The rotation is obtained by

$$\mathbf{R} = \mathbf{U} \operatorname{diag}(1, 1, \det(\mathbf{U}\mathbf{V}^T)) \mathbf{V}^T. \quad (3)$$

The translation parameters are given by

$$\mathbf{t} = \frac{1}{N_{\mathcal{I}}} \left( \sum_{i \in \mathcal{I}} \mathbf{p}_i - \mathbf{R} \sum_{i \in \mathcal{I}} \mathbf{q}_i \right), \quad (4)$$

where  $N_{\mathcal{I}}$  and  $\mathcal{I}$  are the number of inliers and the inlier set, respectively.

#### 4.1. Loss Functions

Our overall loss function has two individual loss terms, namely classification and registration losses from the two blocks of the network.

**Classification Loss:** The classification loss penalizes incorrect correspondences using cross-entropy:

$$\mathcal{L}_c^k = \frac{1}{N} \sum_{i=1}^N \gamma_i^k H(y_i^k, \sigma(o_i^k)), \quad (5)$$

where  $o_i^k$  are the network outputs before passing them through ReLU and tanh for computing the weights  $w_i$ .  $\sigma$  denotes the sigmoid activation function. Note that the motion between pairs of scans are different, and the index  $k$  is used to denote the associated training pair of scans.  $H(\cdot, \cdot)$  is the cross-entropy function, and  $y_i^k$  (equals to one or zero) is the ground-truth, which indicates whether the  $i$ -th point correspondence is an inlier or outlier. The term  $\mathcal{L}_c^k$  is the classification loss for the 3D point correspondences of a particular scan-pair with an index  $k$ . The  $\gamma_i^k$  balances the

classification loss by the number of examples for each class in the associated scan pair  $k$ .

**Registration Loss:** The registration loss penalizes misaligned points in the point cloud using the distance between the 3D points in the second scan  $\mathbf{q}_i$  and the transformed points from the first 3D scan  $\mathbf{p}_i$ , for  $i = \{1, \dots, N\}$ . The loss function becomes

$$\mathcal{L}_r^k = \frac{1}{N} \sum_{i=1}^N \rho \left( \mathbf{q}_i^k, \mathbf{R}^k \mathbf{p}_i^k + \mathbf{t}^k \right), \quad (6)$$

where  $\rho(\cdot, \cdot)$  is the distance metric function. For a given scan pair  $k$ , the relative motion parameters obtained from the registration block are given by  $\mathbf{R}^k$  and  $\mathbf{t}^k$ . We considered and evaluated distance metrics:  $L_1$ , weighted least squares,  $L_2$ , and Geman-McClure [18] in Sec. 7.

**Total Loss:** The individual loss functions are given below:

$$\mathcal{L}_c = \frac{1}{K} \sum_{k=1}^K \mathcal{L}_c^k \quad \text{and} \quad \mathcal{L}_r = \frac{1}{K} \sum_{k=1}^K \mathcal{L}_r^k, \quad (7)$$

where  $K$  is the total number of scan pairs in the training set. The total training loss is the sum of both the classification and the registration loss terms:

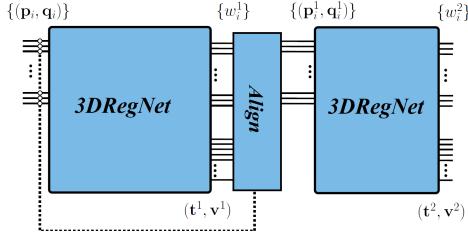
$$\mathcal{L} = \alpha \mathcal{L}_c + \beta \mathcal{L}_r, \quad (8)$$

where the coefficients  $\alpha$  and  $\beta$  are hyperparameters that are manually set for classification and registration terms in the loss function.

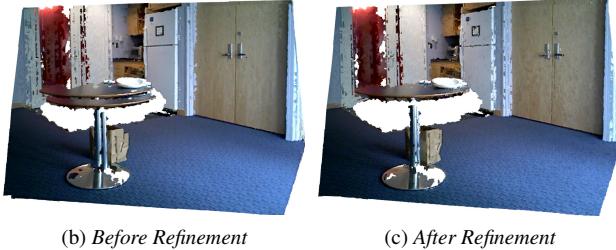
#### 5. 3DRegNet Refinement

We describe our architecture consisting of two 3DRegNet where the second network provides a regression refinement (see Fig. 3(a)). A commonly adopted approach for 3D registration is to first consider a rough estimate for the transformation followed by a refinement strategy. Following this reasoning, we consider the possibility of using an additional 3DRegNet. The first 3DRegNet provides a rough estimate trained for larger rotation and translation parameters values. Subsequently, the second smaller network is used for refinement, estimating smaller transformations. This can also be seen as deep-supervision that is shown to be useful in many applications [30]. Figure 3(a) illustrates the proposed architecture.

**Architecture:** As shown in Fig. 3(a), we use two 3DRegNets, where the first one is used to obtain the coarse registration followed by the second one doing the refinement. Each 3DRegNet is characterized by the regression parameters  $\{\mathbf{R}^r, \mathbf{t}^r\}$  and the classification weights  $\{w_i^r\}_{i=1}^N$ , with  $r = \{1, 2\}$ . We note that the loss on the second network has to consider the cumulative regression of both



(a) Scheme for refinement using 3DRegNet.



(b) Before Refinement

(c) After Refinement

Figure 3: (a) shows the proposed architecture with two 3DRegNet blocks in sequence. (b),(c) show an improvement upon using an additional 3DRegNet to fine-tune or refine the registration from the first 3DRegNet.

3DRegNets. Hence, the original set of point correspondences  $\{\{p_i, q_i\}\}_{1=1}^N$  are transformed by the following cumulative translation and rotation

$$\mathbf{R} = \mathbf{R}^2 \mathbf{R}^1 \text{ and } \mathbf{t} = \mathbf{R}^2 \mathbf{t}^1 + \mathbf{t}^2. \quad (9)$$

Notice that, in (9), the update of the transformation parameters  $\mathbf{R}$  and  $\mathbf{t}$ , depends on the estimates of both 3DRegNets. The point correspondence update at the refinement network becomes

$$\{(p_i^1, q_i^1)\} = \{(w_i^1 (\mathbf{R}^1 p_i + \mathbf{t}^1), w_i^1 q_i)\}, \quad (10)$$

forcing the second network to obtain smaller transformations that corrects for any residual transformation following the first 3DRegNet block.

**Loss Functions:** The classification and registration losses are computed as in (5) and (6) at each step, then averaged by the total loss:

$$\mathcal{L}_c = \frac{1}{K} \sum_{k=1}^K \frac{1}{2} \sum_{r=1}^2 \mathcal{L}_c^{k,r} \text{ and } \mathcal{L}_r = \frac{1}{K} \sum_{k=1}^K \frac{1}{2} \sum_{r=1}^2 \mathcal{L}_r^{k,r}. \quad (11)$$

We then apply (8) as before.

## 6. Datasets and 3DRegNet Training

**Datasets:** We use two datasets, the synthetic augmented ICL-NUIM Dataset [9] and the SUN3D [63] consisting of real images. The former consists of 4 scenes with a total of about 25000 different pairs of connected point clouds. The latter is composed of 13 randomly selected scenes,

with a total of around 3700 different connected pairs. Using FPFH [47], we extract about 3000 3D point correspondences for each pair of scans in both datasets. Based on the ground-truth transformations and the 3D distances between the transformed 3D points, correspondences are labeled as inliers/outliers using a predefined threshold (set  $y_n^k$  to one or zero). The threshold is set such that the number of outliers is about 50% of the total matches. We select 70% of the pairs for training and 30% for testing for the ICL-NUIM Dataset. With respect to the SUN3D Dataset, we select 10 scenes, for training and 3 scenes, completely unseen with respect to the training set, for testing.

**Training:** The proposed architecture is implemented in Tensorflow [1]. We used  $C = 8$  for the first 3DRegNet and  $C = 4$  for the refinement 3DRegNet<sup>1</sup>. The other values for the registration blocks are detailed in Sec. 4. The network was trained for 1000 epochs with 1092 steps for the ICL-NUIM dataset and for 1000 epochs with 200 steps for the SUN3D dataset. The learning rate was  $10^{-4}$ , while using the Adam Optimizer [28]. A cross-validation strategy is used during training. We used a batch size of 16. The coefficients of the classification and registration terms are given by  $\alpha = 0.5$  and  $\beta = 10^{-3}$ . The network was trained using an INTEL i7-7600 and a NVIDIA GEFORCE GTX 1070. For a fair comparison to the classical methods, all run times were obtained using CPU, only.

**Data Augmentation:** To generalize for unseen rotations, we augment the training dataset by applying random rotations. Taking inspiration from [4, 37, 42], we propose the use of Curriculum Learning (CL) data augmentation. The idea is to start small [4], (i.e., easier tasks containing small values of rotation) and having the tasks ordered by increasing difficulty. The training only proceeds to harder tasks after the easier ones are completed. However, an interesting alternative of traditional CL was adopted. Let the magnitude of the augmented rotation to be applied in the training be denoted as  $\theta$ , and an epoch such that  $\tau \in [0, 1]$  (normalized training steps). In CL, we should start small at the beginning of each epoch. However, this breaks the smoothness of  $\theta$  values (since the maximum value for  $\theta$ , i.e.,  $\theta_{\text{Max}}$  has been reached at the end of the previous epoch). This can easily be tackled if we progressively increase the  $\theta$  up to  $\theta_{\text{Max}}$  at  $\tau = 0.5$ , decreasing  $\theta$  afterwards.

## 7. Experimental Results

In this section, we start by defining the evaluation metrics used throughout the experiments. Then, we present some ablation studies considering: 1) the use of different distance metrics; 2) different parameterizations for the rotation; 3) the use of Procrustes vs. DNN for estimat-

<sup>1</sup> $C$  was chosen empirically by training and testing.

Distance Function	Rotation [deg]		Translation [m]		Time [s]	Classification Accuracy
	Mean	Median	Mean	Median		
$L_2$ -norm	2.44	1.64	0.087	0.067	0.0295	0.95
$L_1$ -norm	<b>1.37</b>	<b>0.90</b>	<b>0.054</b>	<b>0.042</b>	0.0281	<b>0.96</b>
Weighted $L_2$ -norm	1.89	1.33	0.070	0.056	0.0294	0.95
Geman-McClure	2.45	1.59	0.089	0.068	0.0300	0.95

Table 1: Evaluation of the different distance functions on the training of the proposed architecture.

ing the transformation parameters; 4) the sensitivity to the number of point correspondences; 5) the use of Data-Augmentation in the training; and 6) the use of the refinement network. The ablation studies are performed on the ICL-NUIM dataset. We conclude the experiments with some comparison with previous methods and the application of our method in unseen scenes.

**Evaluation Metrics:** We defined the following metrics for accuracy. For rotation, we use

$$\delta(\mathbf{R}, \mathbf{R}_{GT}) = \text{acos} \left( \frac{\text{trace}(\mathbf{R}^{-1} \mathbf{R}_{GT}) - 1}{2} \right), \quad (12)$$

where  $\mathbf{R}$  and  $\mathbf{R}_{GT}$  are the estimated and ground-truth rotation matrices, respectively. We refer to [35] for more details. For measuring the accuracy of translation, we use

$$\delta(\mathbf{t}, \mathbf{t}_{GT}) = \|\mathbf{t} - \mathbf{t}_{GT}\|. \quad (13)$$

For the classification accuracy, we used the standard classification error. The computed weights  $w_i \in [0, 1]$  will be rounded to 0 or 1 based on a threshold ( $\mathcal{T} = 0.5$ ) before measuring the classification error.

## 7.1. Ablation Studies

**Distance Metrics:** We start these experiments by evaluating the 3DRegNet training using different types of distance metrics in the regression loss function. Namely, we use: 1) the  $L_2$ -norm; 2)  $L_1$ -norm; 3) Weighted  $L_2$ -norm with the weights obtained from the classification block; and 4) German-McClure distances. For all the pairwise correspondences in the testing phase, we compute the rotation and translation errors obtained by the 3DRegNet. The results of the classification are reported in Tab. 1, in which we use the minimal Lie algebra representation for the rotation.

As it can be seen from these results (see Tab 1), the  $L_1$ -norm gives the best results in all the evaluation criteria. It is interesting to note that weighted  $L_2$ -norm, despite using the weights from the classification block, did not perform as good as the  $L_1$ -norm. This is possible since the registration block also utilizes the outputs from some of the intermediate layers of the classification block. Based on these results, the remaining evaluations are conducted using the  $L_1$ -norm.

**Parameterization of R:** We study the following three parameterizations for the rotation: 1) minimal Lie algebra

Representation	Rotation [deg]		Translation [m]		Time [s]	Classification Accuracy
	Mean	Median	Mean	Median		
Lie Algebra	<b>1.37</b>	<b>0.90</b>	<b>0.054</b>	<b>0.042</b>	0.0281	<b>0.96</b>
Quaternions	1.55	1.11	0.067	0.054	0.0284	0.95
Linear	5.78	4.78	0.059	0.042	0.0275	0.95
Procrustes	1.65	1.52	0.235	0.233	0.0243	0.52

Table 2: Evaluation of different representations for the rotations.

Matches	Rotation [deg]		Translation [m]		Time [s]	Classification Accuracy
	Mean	Median	Mean	Median		
10%	2.40	1.76	0.089	0.073	0.0106	0.94
25%	1.76	1.22	0.068	0.054	0.0149	0.95
50%	1.51	1.01	0.060	0.047	0.0188	0.95
75%	1.41	0.92	0.056	0.044	0.0241	0.96
90%	1.38	0.90	0.055	0.043	0.0267	0.96
100%	<b>1.37</b>	<b>0.90</b>	<b>0.054</b>	<b>0.042</b>	0.0281	<b>0.96</b>

Table 3: Evaluation of different number of correspondences.

(three parameters); 2) quaternions (four parameters); and 3) linear matrix form (nine parameters). The results are shown in Tab. 2. We observe that the minimal parameterization using Lie algebra provides the best results. In the experimental results that follows, we use the three parameters Lie algebra representation. While Lie algebra performs better for the problem on hand, we cannot generalize this conclusion to other problems like human pose estimation, as shown in [66].

**Regression with DNNs vs. Procrustes:** We aim at evaluating the merits of using DNNs vs. Procrustes to get the 3D registration, as shown in Fig. 2(a) and Fig. 2(b). From Tab. 2, we conclude that the differentiable Procrustes method does not solve the problem as accurately as DNNs. The run time is lower than the DNNs with the Lie Algebra, but the difference is small and can be neglected. On the other hand, the classification accuracy degrades significantly. From now on, we use the DNNs for the regression.

**Sensitivity to the number of correspondences:** Instead of considering all the correspondences in each of the pairwise scans of the testing examples, we select a percentage of the total number of matches ranging from 10% to 100% (recall that the total number of correspondences per pair is around 3000). The results are shown in Tab. 3.

As expected, the accuracy of the regression degrades as the number of input correspondences decreases. The classification, however, is not affected. The inlier/outlier classifications should not depend on the number of input correspondences, while the increase of the number of inliers should lead to a better estimate.

**Data Augmentation:** Using the 3DRegNet trained in the previous sections, we select a pair of 3D scans from the training data and rotate the original point-clouds to increase the rotation angles between them. We vary the magnitude

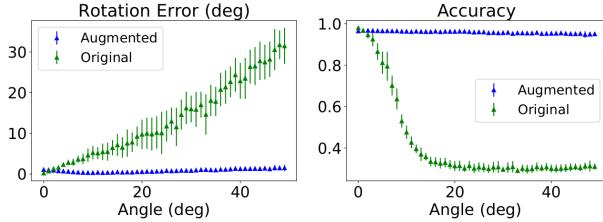


Figure 4: *Training with and without data augmentation.* It is observed an improvement on the test results when perturbances are applied. The data augmentation regularizes the network for other rotations that were not included in the original dataset.

Refinement	Rotation [deg]		Translation [m]		Time [s]	Classification Accuracy
	Mean	Median	Mean	Median		
without	1.37	0.90	0.054	0.042	0.0281	0.96
with	1.19	0.89	0.053	0.044	0.0327	0.94

Table 4: *Evaluation of the use of 3DRegNet refinement.*

of this rotation ( $\theta$ ) from 0 to 50 degrees, and the results for the rotation error and accuracy in the testing are shown in Fig. 4 (green curve). Afterward, we train the network a second time, using the data augmentation strategy proposed in Sec. 6. At each step, the pair of examples is perturbed by a rotation with increasing steps of  $2^\circ$ , setting the maximum value of  $\theta = 50^\circ$ . We run the test as before, and the results are shown in Fig. 4 (blue curve).

From this experiment we can conclude that, by only training with the original dataset, we constrained to the rotations contained in the dataset. On the other hand, by performing a smooth regularization (CL data augmentation), we can overcome this drawback. Since the datasets at hand are sequences of small motions, there is no benefit on generalizing the results for the rotation parameters. If all the involved transformations are small, the network should be trained as such. We do not carry out data augmentation in the following experiments.

**3DRegNet refinement:** We consider the use of the extra 3DRegNet presented in Sec. 5 for regression refinement. This composition of two similar networks was developed to improve the accuracy of the results. From Tab. 4, we observe an overall improvement on the transformation estimation, without compromising the run time significantly. The classification accuracy decreases by 2%, but does not influence the final regression. This improvement on the estimation can also be seen in Fig. 3, where the estimation using only one 3DRegNet (Fig. 3(b)) is still a bit far from the true alignment, in comparison to using the 3DRegNet with refinement, shown in Fig. 3(c), which is closer to the correct alignment. For the remainder of the paper, when we refer to 3DRegNet, we are using the refinement network.

Method	Rotation [deg]		Translation [m]		Time [s]
	Mean	Median	Mean	Median	
FGR	1.39	0.53	<b>0.045</b>	0.024	0.2669
ICP	3.78	<b>0.43</b>	0.121	<b>0.023</b>	0.1938
RANSAC	1.89	1.45	0.063	0.051	0.8441
<b>3DRegNet</b>	<b>1.19</b>	0.89	0.053	0.044	<b>0.0327</b>
FGR + ICP	1.01	0.38	0.038	0.021	0.3422
RANSAC + U	1.42	1.02	0.050	0.042	0.8441
<b>3DRegNet + ICP</b>	0.55	0.34	0.030	0.021	0.0691
<b>3DRegNet + U</b>	<b>0.28</b>	<b>0.22</b>	<b>0.014</b>	<b>0.011</b>	<b>0.0327</b>

(a) *Baselines results on the ICL-NUIM Dataset.*

Method	Rotation [deg]		Translation [m]		Time [s]
	Mean	Median	Mean	Median	
FGR	2.57	1.92	0.121	<b>0.067</b>	0.1623
ICP	3.18	<b>1.50</b>	0.146	0.079	0.0596
RANSAC	3.00	1.73	0.148	0.074	2.6156
<b>3DRegNet</b>	<b>1.84</b>	1.69	<b>0.087</b>	0.078	<b>0.0398</b>
FGR + ICP	1.49	<b>1.10</b>	0.070	<b>0.046</b>	0.1948
RANSAC + U	2.74	1.48	0.134	0.061	2.6157
<b>3DRegNet + ICP</b>	1.26	1.14	0.066	0.048	0.0852
<b>3DRegNet + U</b>	<b>1.16</b>	<b>1.10</b>	<b>0.053</b>	0.050	<b>0.0398</b>

(b) *Results on unseen sequences (SUN3D Dataset).*

Table 5: *Comparison with the baselines: FGR [65]; RANSAC-based approaches [17, 48]; and ICP [6].*

## 7.2. Baselines

We use three baselines. The Fast Global Registration [65] (FGR) geometric method, that aims to provide a global solution for some set of 3D correspondences. The second baseline is the classical RANSAC method [17]. The third baseline is ICP [6]. Note that we are comparing our technique against both correspondence-free (ICP) and correspondence-based methods (FGR, RANSAC). For this test, we use the ICL-NUIM dataset. In the attempt to ascertain what is the strategy that provides the best registration prior for the ICP, we applied two methods termed as FGR + ICP and 3DRegNet + ICP, where the initialization for ICP is done using the estimated transformations given by the FGR and the 3DRegNet, respectively. Also, for evaluating the quality of the classification, we take the inliers given by the 3DRegNet and RANSAC, and input these in a least square non-linear Umeyama refinement technique presented in [53]. These methods are denoted as 3DRegNet + U and RANSAC + U, respectively. The results are shown in Tab. 5(a).

Cumulative distribution function (i.e., like a precision-recall curve) is shown in Fig. 6(a) to better illustrate the performance of both 3DRegNet and FGR. In this figure, part of the tests are shown where the rotation error is less than a given error angle. It can be seen that FGR performs better than 3DRegNet (until  $2^\circ$  error). Afterward, 3DRegNet starts to provide better results. This implies that FGR does better for easier problems but for a larger number of cases it

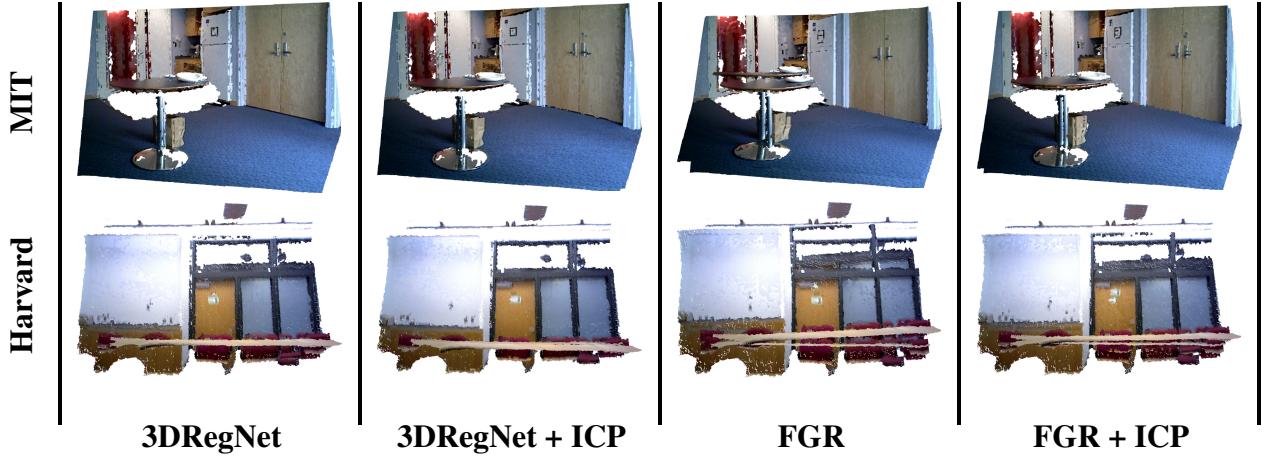


Figure 5: Two examples of 3D point-cloud alignment using the 3DRegNet, 3DRegNet + ICP, FGR, and FGR + ICP methods. A pair of 3D scans were chosen from three scenes in the SUN3D data-set: MIT and Harvard sequences. These sequences were not used in the training of the network.

has high error (also higher than that of 3DRegNet). In other words, FGR has a heavier tail, hence lower median error and higher mean error compared to 3DRegNet as evident from Tab. 5. As the complexity of the problem increases, 3DRegNet becomes a better algorithm. This is further illustrated when we compare their performance in combination with ICP. Here, we can see that the initial estimates provided by 3DRegNet (3DRegNet + ICP) outperform to those of FGR + ICP. It is particularly noteworthy that even though ICP is local, 3DRegNet + ICP converges to a better minimum than FGR + ICP. This means that a deep learning approach allows us to perform better when the pairwise correspondences are of lower quality, which makes the problem harder. In terms of computation time, we are at least 8x faster than FGR, and 25x faster than RANSAC. To do a fair comparison for all the methods, all computation timings are obtained using CPU.

When considering the use of ICP and Umeyama refinement techniques, in terms of accuracy, we see that both the 3DRegNet + ICP and the 3DRegNet + U beat any other methods. With results from 3DRegNet + ICP, we conclude that the solution to the transformation provided by our network leads ICP to a lower minimum than FGR + ICP. From 3DRegNet + U, we get that our classification selects better the inliers. In terms of computation time, we can draw the same conclusions as before.

### 7.3. Results in Unseen Sequences

For this test, we use the SUN3D dataset. We run the same tests as in the previous section. However, while in Sec. 7.2 we used all the pairs from the sequences and split them into training and testing, here, we run our tests in hold-out training sequences. The results are shown in Tab. 5(b) and Fig. 6(b). The conclusions are similar as in the previous

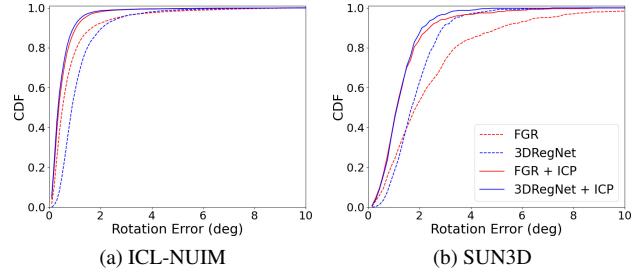


Figure 6: Cumulative distribution function of the rotation errors of 3DRegNet vs. FGR.

section. We observe that the results from 3DRegNet do not degrade significantly, which means that the network is able to generalize the classification and registration to unseen sequences. Some snapshots are shown in Fig. 5.

## 8. Discussion

We propose 3DRegNet, a deep neural network that can solve the scan registration problem by jointly solving the outlier rejection given 3D point correspondences and computing the pose for alignment of the scans. We show that our approach is extremely efficient. It performs as well as the current baselines, while still being significantly faster. We show additional tests and visualizations of 3D registrations in the Supplementary Materials.

## Acknowledgements

This work was supported by the Portuguese National Funding Agency for Science, Research and Technology project PTDC/EEI-SII/4698/2014, and the LARSyS - FCT Plurianual funding 2020-2023.

## References

- [1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [2] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivastan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 7163–7172, 2019.
- [3] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 9(5):698–700, 1987.
- [4] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Int'l Conf. Machine learning (ICML)*, pages 41–48, 2009.
- [5] Florian Bernard, Frank R. Schmidt, Johan Thunberg, and Daniel Cremers. A combinatorial solution to non-rigid 3d shape-to-image matching. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 1436–1445, 2017.
- [6] Paul J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 14(2):239–256, 1992.
- [7] Alvaro Parra Bustos and Tat-Jun Chin. Guaranteed outlier removal for point cloud registration with correspondences. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 40(12):2868–2882, 2018.
- [8] Avishek Chatterjee and Venu Madhav Govindu. Robust relative rotation averaging. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 40(4):958–972, 2018.
- [9] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5556–5565, 2015.
- [10] Taco S. Cohen, Mario Geiger, Jonas Koehler, and Max Welling. Spherical cnns. In *Int'l Conf. Learning Representations (ICLR)*, 2018.
- [11] Zheng Dang, Kwang Moo Yi, Yinlin Hu, Fei Wang, Pascal Fua, and Mathieu Salzmann. Eigendecomposition-free training of deep networks with zero eigenvalue-based losses. In *European Conf. Computer Vision (ECCV)*, pages 792–807, 2018.
- [12] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 195–205, 2018.
- [13] Haowen Deng, Tolga Birdal, and Slobodan Ilic. 3d local features for direct pairwise registration. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 3239–3248, 2019.
- [14] Li Ding and Chen Feng. Deepmapping: Unsupervised map estimation from multiple point clouds. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 8650–8659, 2019.
- [15] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2472 – 2481, 2017.
- [16] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning  $so(3)$  equivariant representations with spherical cnns. In *European Conf. Computer Vision (ECCV)*, pages 52–68, 2018.
- [17] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.
- [18] Stuart Geman and Donald E. McClure. Bayesian image analysis: An application to single photon emission tomography. In *Proc. American Statistical Association*, pages 12–18, 1985.
- [19] Zan Gojcic, Caifa Zhou, Jan D. Wegner, and Andreas Wieser. The perfect match: 3d point cloud matching with smoothed densities. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5545–5554, 2019.
- [20] Venu Madhav Govindu and A. Pooja. On averaging multi-view relations for 3d scan registration. *IEEE Trans. Image Processing (T-IP)*, 23(3):1289–1302, 2014.
- [21] Lei Han, Mengqi Ji, Lu Fang, and Matthias Niessner. Regnet: Learning the optimization of direct image-to-image pose registration. *arXiv:1812.10212*, 2018.
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.
- [23] Joao F. Henriques and Andrea Vedaldi. Mapnet: An allocentric spatial memory for mapping environments. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 8476–8484, 2018.
- [24] Dirk Holz, Alexandru E. Ichim, Federico Tombari, Radu B. Rusu, and Sven Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine (RA-M)*, 22(4):110–124, 2015.
- [25] Ji Hou, Angela Dai, and Matthias Niessner. 3d-sis: 3d semantic instance segmentation of rgb-d scans. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 4416–4425, 2019.
- [26] Xiangru Huang, Zhenxiao Liang, Xiaowei Zhou, Yao Xie, Leonidas Guibas, and Qixing Huang. Learning transformation synchronization. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 8082–8091, 2019.
- [27] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Poseonet: A convolutional network for real-time 6-dof camera

- relocalization. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 2938–2946, 2015.
- [28] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *Int'l Conf. Learning Representations (ICLR)*, 2015.
- [29] Huu M. Le, Thanh-Toan Do, Tuan Hoang, and Ngai-Man Cheung. Sdrsac: Semidefinite-based randomized approach for robust point cloud registration without correspondences. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 124–133, 2019.
- [30] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets, 2014.
- [31] Hongdong Li and Richard Hartley. The 3d-3d registration problem revisited. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1–8, 2017.
- [32] Weixin Lu, Guowei Wan, Yao Zhou, Xiangyu Fu, Pengfei Yuan, and Shiyu Song. Deepvcp: An end-to-end deep neural network for point cloud registration. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 3523–3532, 2019.
- [33] Jiayi Ma, Xingyu Jiang, Junjun Jiang, Ji Zhao, and Xiaojie Guo. Lmr: Learning a two-class classifier for mismatch removal. *IEEE Trans. Image Processing (T-IP)*, 28(8):4045–4059, 2019.
- [34] Lingni Ma, Jorg Stuckler, Christian Kerl, and Daniel Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 598–605, 2017.
- [35] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision*. Springer-Verlag New York, 2004.
- [36] Andre Mateus, Srikumar Ramalingam, and Pedro Miraldo. Minimal solvers for 3d scan alignment with pairs of intersecting lines. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [37] Tamara Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *IEEE Trans. Neural Networks and Learning Systems (T-NNLS)*, 2019.
- [38] Nicolas Mellado, Niloy Mitra, and Dror Aiger. Super 4pcs: Fast global pointcloud registration via smart indexing. *Computer Graphics Forum (Proc. EUROGRAPHICS)*, 33(5):205–215, 2014.
- [39] Pedro Miraldo, Surojit Saha, and Srikumar Ramalingam. Minimal solvers for mini-loop closures in 3d multi-scan alignment. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 9699–9708, 2019.
- [40] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 32(12):2262–2275, 2010.
- [41] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE Int'l Symposium on Mixed and Augmented Reality (ISMAR)*, pages 127–136, 2011.
- [42] Ilkay Oksuz, Bram Ruijsink, Esther Puyol-Antn, James R. Clough, Gastao Cruz, Aurelien Bustin, Claudia Prieto, Rene Botnar, Daniel Rueckert, Julia A. Schnabel, and Andrew P. King. Automatic cnn-based detection of cardiac mr motion artefacts using k-space data augmentation and curriculum learning. *Medical Image Analysis*, 55:136–147, 2019.
- [43] Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Colored point cloud registration revisited. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 143–152, 2017.
- [44] Graeme P. Penney, Philip J. Edwards, Andrew P. King, Jane M. Blackall, Philipp G. Batchelor, and David J. Hawkes. A stochastic iterative closest point algorithm (stochasticicp). In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 762–769, 2001.
- [45] Stephen Phillips and Kostas Daniilidis. All graphs lead to rome: Learning geometric and cycle-consistent representations with graph convolutional networks. *arXiv:1901.02078*, 2019.
- [46] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017.
- [47] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *IEEE Int'l Conf. Robotics and Automation (ICRA)*, pages 3212–3217, 2009.
- [48] Peter H. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.
- [49] Aleksandr V. Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-icp. In *Robotics: Science and Systems (RSS)*, 2009.
- [50] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019.
- [51] Miroslava Slavcheva, Maximilian Baust, Daniel Cremers, and Slobodan Ilic. Killingfusion: Non-rigid 3d reconstruction without correspondences. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5474–5483, 2017.
- [52] Gary K.L. Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C. Langbein, Yonghuai Liu, David Marshall, Ralf R. Martin, Xian-Fang Sun, and Paul L. Rosin. Registration of 3d point clouds and meshes: A survey from rigid to nonrigid. *IEEE Trans. Visualization and Computer Graphics (T-VCG)*, 19(7):1199–1217, 2013.
- [53] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 13(4):376–380, 1991.
- [54] Yue Wang and Justin Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 3522–3531, 2019.
- [55] Xinshuo Weng and Kris Kitani. Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud. In *ICCV Workshops*, 2019.
- [56] Jay M. Wong, Vincent Kee, Tiffany Le, Syler Wagner, Gian-Luca Mariottini, Abraham Schneider, Lei Hamilton,

- Rahul Chipalkatty, Mitchell Hebert, David M.S. Johnson, Jimmy Wu, Bolei Zhou, and Antonio Torralba. Segicp: Integrated deep semantic segmentation and pose estimation. In *IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS)*, pages 5784–5789, 2017.
- [57] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 38(11):2241–2254, 2016.
- [58] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *IEEE Int'l Conf. Computer Vision (ICCV)*, pages 1457–1464, 2013.
- [59] Zi Jian Yew and Gim Hee Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conf. Computer Vision (ECCV)*, pages 630–646, 2018.
- [60] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2674, 2018.
- [61] Andy Zeng, Shuran Song, Matthias Niessner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgbd reconstructions. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 199–208, 2017.
- [62] Chen Zhao, Zhiguo Cao, Chi Li, Xin Li, and Jiaqi Yang. Nm-net: Mining reliable neighbors for robust feature correspondences. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 215–224, 2019.
- [63] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Advances in Neural Information Processing Systems (NIPS)*, pages 487–495, 2014.
- [64] Lei Zhou, Siyu Zhu, Zixin Luo, Tianwei Shen, Runze Zhang, Mingmin Zhen, Tian Fang, and Long Quan. Learning and matching multi-view descriptors for registration of point clouds. In *European Conf. Computer Vision (ECCV)*, pages 527–544, 2018.
- [65] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *European Conf. Computer Vision (ECCV)*, pages 766–782, 2016.
- [66] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, pages 5745–5753, 2019.
- [67] Michael Zollhofer, Matthias Niessner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, and Marc Stamminger. Real-time non-rigid reconstruction using an rgbd camera. *ACM Trans. Graphics*, 33(4), 2014.

# 3DRegNet: A Deep Neural Network for 3D Point Registration

## (SUPPLEMENTARY MATERIALS)

G. Dias Pais<sup>1</sup>, Sri Kumar Ramalingam<sup>2</sup>, Venu Madhav Govindu<sup>3</sup>,  
Jacinto C. Nascimento<sup>1</sup>, Rama Chellappa<sup>4</sup>, and Pedro Miraldo<sup>1</sup>

<sup>1</sup>Instituto Superior Técnico, Lisboa    <sup>2</sup>Google Research, NY  
<sup>3</sup>Indian Institute of Science, Bengaluru    <sup>4</sup>University of Maryland, College Park

*In these supplementary materials, we start by showing additional figures illustrating the 3DRegNet vs. FGR, with and without ICP for refinement, (see Sec. A). In Sec. B, we discriminate the results obtained in Tab. 5 of the paper.*

## A. Additional Results

We show some new figures to better illustrate the advantages of the 3DRegNet against previous methods (i.e., Tab. 5 of the main document).

We start by showing additional experimental results on the 3D scan alignment to complement the results shown in Fig. 5 of the paper. Two sequences were used, MIT and BROWN, from the SUN3D dataset. **Please note that the 3DRegNet was not trained using these sequences; these are used for testing only.** These experiments are similar to the ones in Fig. 5 of the paper. However, instead of only showing a pair of 3D scans (required by each of the methods), we show the registration of 10 3D scans. We compute the 3D alignment in a pairwise manner, i.e., we compute the transformation from Scan 1 to Scan2, from Scan 2 to Scan 3, ..., and Scan 9 to Scan 10. Then, we apply transformations to move all the 3D Scans 2, 3, ..., 10 into the first one, which we selected for the reference frame. We consider the cumulative transformation from the first to  $i^{\text{th}}$  3D scan, i.e., we pre-multiplied all the transformations from 1 to  $i$  to move all the point clouds into the first (common) reference frame. We used the methods: (i) 3DRegNet, (ii) 3DRegNet + ICP, (iii) FGR, and (iv) FGR + ICP. These results are shown in Fig. A.7. We show an additional column with the ground-truth transformation for comparison. We use the network trained for the results in Tab. 5(b) of the paper.

As we can see from Fig. A.7, for both the Brown and the MIT sequences, the registration results for the 10 scans given by the 3DRegNet method are much closer to the ground-truth than the FGR. When running the ICP after the 3DRegNet, while for the Brown, we see some improvements (compare the door in 3DRegNet vs. 3DRegNet +

ICP), for the MIT we see some degradation on the results. When comparing FGR with 3DRegNet, for the Brown sequence, we see that the 3DRegNet is performing better than the FGR, even for the case in which we use ICP for the FGR refinement. For the MIT sequence, we see that, while the 3DRegNet is performing better than the FGR, the ICP for refinement after both is leading to the same final 3D registration. However, we can also observe that the 3DRegNet is giving better results than 3DRegNet + ICP and FGR + ICP (see the cabinets in the environment).

We further evaluate the use of 3dRegNet against the current state-of-the-art FGR method by showing the trajectories obtained from each of the methods. The results for 20 frames in two sequences are shown in Fig. A.8. The point clouds shown in this figure are registered using the ground-truth transformations, and the paths shown are computed directly from 3DRegNet + ICP and FGR + ICP. From the top of the Fig. A.8 (Harvard sequence), it can be seen that we are performing better than the FGR + ICP, i.e., 3DRegNet + ICP provides a trajectory estimate that is closer to the ground-truth. For the Brown dataset (bottom of Fig. A.8), we see that both trajectories perform similarly. However, we stress that the 3DRegNet is faster than the competing methods, as shown in the Tab. 5(b) of the paper.

## B. Discriminate Results for SUN3D

Although the main paper presents the overall mean and median for all the pairs in the three sequences of the SUN3D data set, the individual errors for each of the sequence vary significantly. This is because each sequence has its own characteristics. Here we show the discriminate results for each sequence of the SUN3D sequences (see Tab. 6).

From the results, we see that while ICP is performing better than 3DRegNet for the MIT sequence, 3DRegNet is superior in Harvard (both with and without ICP or Umeyama). In the Brown sequence, we see that while we are beating the current state-of-the-art in the mean, without refinement, we are loosing for RANSAC and FGR in the

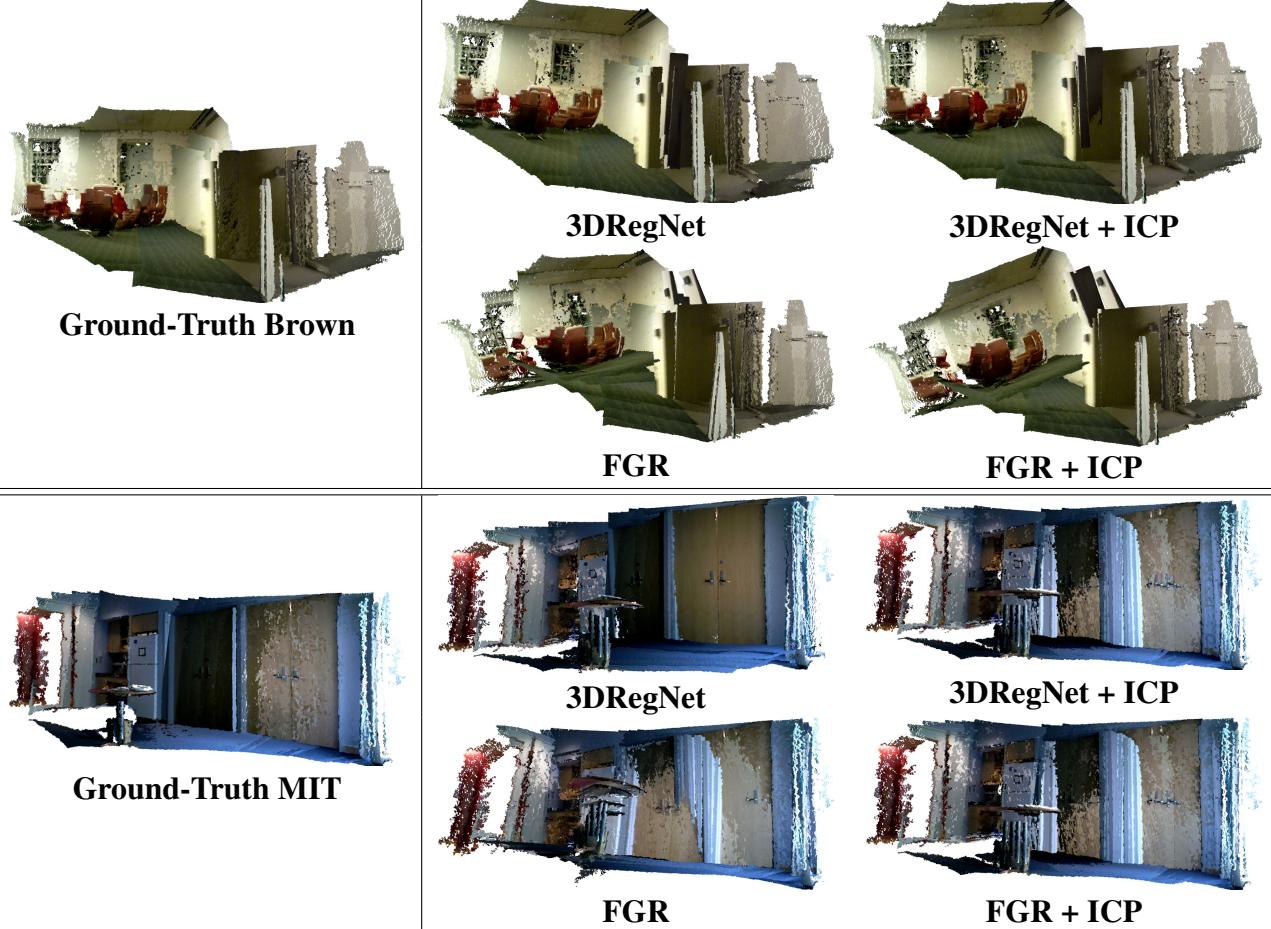


Figure A.7: Results for the alignment of 20 3D scans using the 3DRegNet, 3DRegNet + ICP, FGR, and FGR + ICP. We consider just the transformations computed using the respective methods, i.e., we are not removing the drift from the estimation. No transformation averaging for final refinement was used.

median (though the differences are minor). When considering refinement (i.e. with Umeyama or ICP), in general, our proposal is the best method. Exception is the slightly better performance in the FGR + ICP where the estimated median and the translation are superior by a small margin. Overall, when we see these results, we can draw the same conclusions as the ones addressed in the paper. While both ICP and FGR perform well for less challenging scenarios (small transformations), our method is superior for larger transformations. In addition to these conclusions, we can easily see that the 3DRegNet is significantly faster than any other method, with and without refinement<sup>1</sup>.

<sup>1</sup>We stress that all the methods are being run the same conditions, only using CPU.

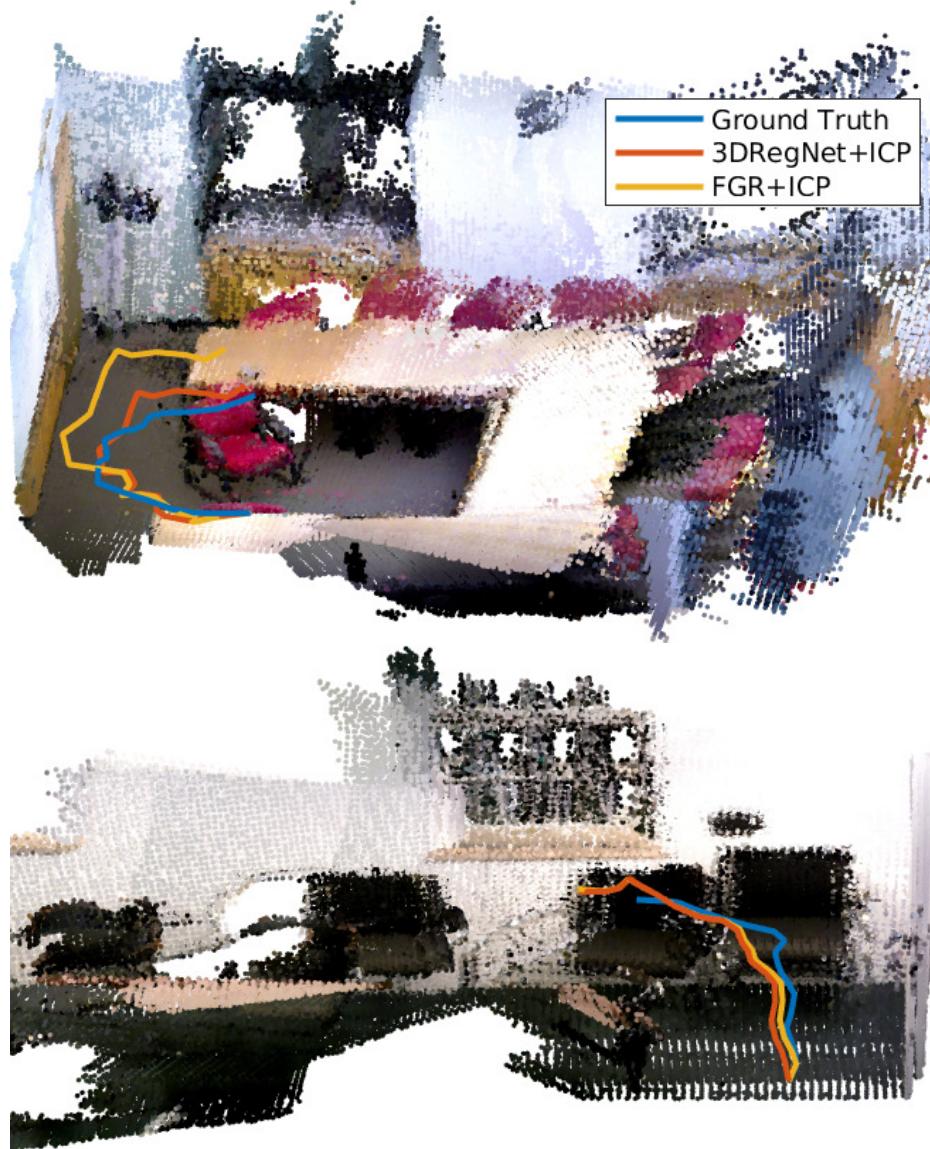


Figure A.8: Two examples of trajectories obtained using the 3DRegNet + ICP vs. FGR + ICP against the Ground-Truth.

Method	Rotation [deg]		Translation [m]		Time [s]
	Mean	Median	Mean	Median	
FGR	1.96	1.58	0.083	0.055	0.16
ICP	<b>1.53</b>	<b>1.14</b>	<b>0.071</b>	<b>0.045</b>	0.086
RANSAC	1.90	1.64	0.080	0.065	2.28
<b>3DRegNet</b>	1.77	1.62	0.080	0.070	<b>0.023</b>
FGR + ICP	<b>1.01</b>	<b>0.38</b>	<b>0.038</b>	<b>0.021</b>	0.19
RANSAC + U	1.58	1.35	0.065	0.053	2.28
<b>3DRegNet + ICP</b>	1.10	1.04	0.047	0.039	0.062
<b>3DRegNet + U</b>	1.15	1.10	0.048	0.047	<b>0.023</b>

(a) MIT

Method	Rotation [deg]		Translation [m]		Time [s]
	Mean	Median	Mean	Median	
FGR	3.25	2.63	0.169	0.117	0.14
ICP	4.94	3.11	0.275	0.221	0.082
RANSAC	2.87	2.28	0.166	0.113	3.49
<b>3DRegNet</b>	<b>1.75</b>	<b>1.60</b>	<b>0.095</b>	<b>0.078</b>	<b>0.023</b>
FGR + ICP	1.59	1.30	0.112	0.067	0.18
RANSAC + U	2.54	1.82	0.149	0.092	3.49
<b>3DRegNet + ICP</b>	1.38	1.28	0.098	0.075	0.085
<b>3DRegNet + U</b>	<b>1.20</b>	<b>1.13</b>	<b>0.069</b>	<b>0.059</b>	<b>0.023</b>

(b) Harvard

Method	Rotation [deg]		Translation [m]		Time [s]
	Mean	Median	Mean	Median	
FGR	2.72	1.77	0.12	<b>0.060</b>	0.15
ICP	3.74	1.69	0.16	0.11	0.080
RANSAC	3.99	<b>1.66</b>	0.20	0.071	2.55
<b>3DRegNet</b>	<b>1.92</b>	1.78	<b>0.089</b>	0.082	<b>0.020</b>
FGR + ICP	1.64	1.14	0.079	<b>0.046</b>	0.19
RANSAC + U	3.77	1.48	0.182	0.059	2.55
<b>3DRegNet + ICP</b>	1.33	1.18	0.067	0.047	0.085
<b>3DRegNet + U</b>	<b>1.13</b>	<b>1.06</b>	<b>0.051</b>	0.048	<b>0.020</b>

(c) Brown

Table 6: Comparison with the baselines: FGR [65]; and RANSAC-based approaches [17, 48].