Local-Global based Deep Registration Neural Network for Rigid Alignment

Victor Villena-Martinez[1] Marcelo Saval-Calvo[1] Jorge Azorin-Lopez[1]
Andres Fuster-Guillo[1] Robert B. Fisher[2]

[1]University of Alicante
{vvillena, msaval, jazorin, fuster}@dtic.ua.es
[2]University of Edinburgh
rbf@inf.ed.ac.uk

Abstract

Three-dimensional registration is a well-known topic in computer vision that aims to align two datasets (e.g. point clouds). Recent approaches to this problem are based on learning techniques. In this paper, we present an improved solution to the problem of registration with a novel architecture that, given two 3D point clouds as input, estimates the rotation to map one into the other. The network architecture is conceptually divided into two parts, the first part is a feature selection based on PointNet and PointNet++. The second part estimates the rotation with Euler angles by calculating the correspondences with a FlowNet-based network, and finally the rotations in yaw, pitch, and roll. The generalization capability of the proposal allows mapping two point clouds in a wide range of angles with a stable error over the whole range. Experiments have been carried out using the ModelNet10 objects dataset, varying the axis and the angle of rotation to provide a sufficiently complete evaluation of the architecture. The results show an average distance Mean Square Error of 4.94E-05 within a unit sphere and a rotation error of 1.18 degrees. The results with noisy point clouds are sufficiently accurate providing the network was trained only with noise-free data, demonstrating good generalization of the approach.
Keywords: deep registration network, 3D registration network, deep learning, point cloud registration

1 Introduction

There is a wide range of applications and topics in which finding the transformation between two views of the same scene is necessary. For instance, when a scene needs to be transformed to a canonical position or a camera position change needs to be recovered. Frequently, the main issue is to find a 3D coordinate transformation where no deformation of the scene is considered. One of the most well-known ways to determine the transformation between two instances of a scene is using a registration technique. Registration methods receive two sets of data, either point clouds, images, or meshes, and minimize the distance between them by calculating a transformation (rotation and/or translation) that maps one onto the other. There are a large number of 2D and 3D registration methods, either traditional approaches or machine learning ones, where the former uses pre-defined methods and the latter learns from examples.

The traditional approaches follow a series of phases [16] where the choice of the matching features are highly relevant on the final result quality. On the other hand, the machine learning techniques, and more recently the deep learning, can either replace some of the phases in traditional registration or perform the complete pipeline from raw data to transformation estimation [19]. They cope with the matching issue by inferring their own features that best fit the purpose or characteristics of the scenes to be aligned.

Given the advantage of deep learning, many different solutions have been proposed. Generic reviews about applying deep learning to 3D point clouds can be found, such as [5]. There are other more specifically focused on registration, such as the recent review by Villena-Martinez et al. [19]. It presents an in-depth study of the different approaches to classifying registration algorithms depending on several aspects, like the step of the traditional pipeline on which they are focused, input data type, or deformation level. Zhang et al. [25] reviewed registration methods for rigid transformations. Various approaches use

pre-trained networks that have shown good performance in some tasks and then transfer learning to adapt to other applications.

Purpose. We propose a new method to learn to register rigidly two point clouds.

Our work. We propose an architecture for rigid registration that combines local and global features extracted only from the geometric information of the point cloud, without any other input channel like color information or normals. Our proposed architecture is based on a combination of PointNet [13], PointNet++ [14] and FlowNet3D [9].

Contributions. We show how the combination of global and local features in a registration network leads to better registration with unseen categories.

2 Related Work

The registration pipeline can be divided into three core components that define the sequence of registration algorithms [18]: target selection, correspondences and constraints, and optimization.

- Target selection. The data that will be used as a reference for the registration process is selected here. Also, the functional model indicating the type of transformation that will be applied is defined (e.g. rigid, Euclidean, affine, etc.). Typically, in a registration problem, it is possible to differentiate between the data that will remain fixed in terms of coordinate reference, and the data that will be moved according to the transformation that aligns both sets.
- Correspondences and constraints. This stage refers to the process of finding the features in the input sets and the matching between them. The pairs of matching features are called correspondences. Usually, they are intended to be invariant to rotations and translations, so features are recognizable from different points of view.
- Optimization. The process of finding the transformation that minimizes a function distance between the correspondences and aligning them into a common reference space.

Some papers use neural networks to address only part of the process (often feature extraction) while others use the networks for the whole rigid registration process.

2.1 Feature extraction

Learning approaches have demonstrated successful results in performing feature extraction and matching for registration purposes. Liu et al. [9] stated that CNNs do not provide good results when working with point clouds due to their irregular structure. In their workflow, they propose a PointNet++ based network architecture to learn hierarchical features of a point cloud. Zeng et al. [24] proposed a convolutional neural network to extract a 512-dimensional representation of a point cloud, which is input into the network as a set of voxels of TDF (Truncated Distance Function). Elbaz et al. [4] extract low-dimensional descriptors to capture significant geometric properties of the input point clouds using a Deep Auto Encoder (DAE). Wang and Solomon [20] employ a DGCNN [21] to extract features of the point clouds and embed them into high-dimensional space.

2.2 Rigid registration

In the literature, several advancements on rigid registration using neural networks could be found. Ding and Feng [3] propose a network which is capable of locating input point clouds in a global space to solve Simultaneous Localization and Mapping (SLAM) problems, where multiple point clouds have to be registered rigidly. Their proposal includes two networks, first a localization network to estimate the sensor pose for a given point cloud. Second, an occupancy map network to retrieve a discrete occupancy map in global space. Aoki et al. [1] modifies the Lucas and Kanade (LK) [10] algorithm with PointNet to extract global feature vectors and compute the rigid transformation that aligns both inputs. To improve the generalization ability, Li et al. [8] modifies PointNetLK [1], including analytical gradients. Kurobe et al. [7] proposed CorsNet, a rigid registration architecture using PointNet to estimate feature vectors from both data sets and SVD for the transformation calculation.

It is noticeable how some authors address the problem of rigid registration by proposing staged-architectures for distributing the different parts of the registration pipeline. Chang and Pham [2] cover

the problem of rigid registration with a two stage CNN framework to perform a coarse registration followed by a fine alignment. Pais et al. [11] presented 3DRegNet, a 2-step network architecture divided into classification and registration blocks. A set of corresponding 3D points are input to the classification block which is based on ResNet [6] to output the corresponding features. Then, these features are used by the registration block, defined by a set of convolutional layers, to output the transformation that aligns both input sets. Yuan et al. [23] propose a method that learns latent correspondences between points and Gaussian Mixture Model (GMM) components to be pose-invariant. They employ a neural network that estimates point-to-component correspondences, following by two compute blocks to obtain the transformation.

3 Problem Formulation

As stated before, registration computes the alignment between datasets. Given two inputs $P = \{_{1},_{2},_{3}, ...,_{n}\}$ and $Q = \{_{1},_{2},_{3}, ...,_{m}\}$, a registration process finds a transformation function χ that minimizes the alignment error between P and Q, by evaluating the distance error (E_P) between pairs of correspondences ($_{i},_{j}$) of each input, as shown in Equation 1. There are different error and distance functions ($dist$), for instance perpendicular distance, Euclidean distance or Huber distance, L_1 error, etc.

$$E_P = \sum_{i,j}^{n,m} gate(_{i},_{j}) * dist(_{i}, \chi(a_{,j})) \tag{1}$$

χ transforms each element of P, according to the transformation parameters a, intending to minimize the error E_P between P and Q. The function $gate(_{i},_{j})$ weights the correspondences between both P and Q datasets, which might include zero weights for features without correspondences. Traditional algorithms often use binary correspondences, such Iterative Closest Point (ICP) [15] with closest point matching, or SIFT+RANSAC [12] that use feature-based correspondences. CPD and variants [17], however, use Gaussian soft assignment to match the data.

4 Method

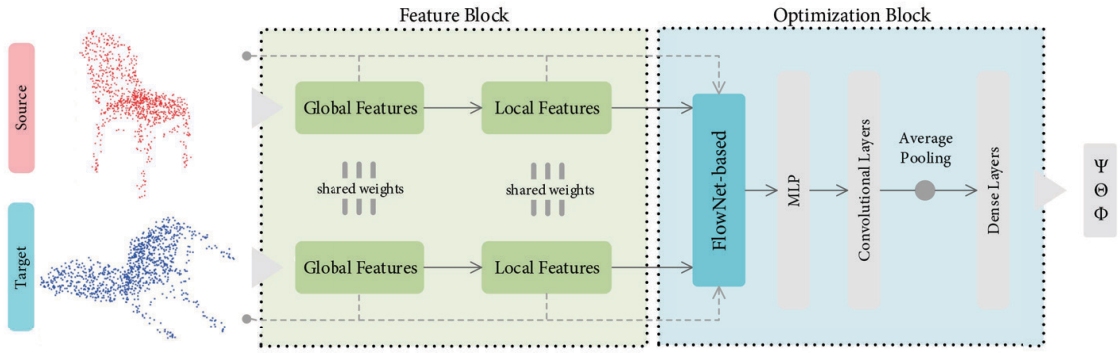

Figure 1: Proposed architecture for two 3D point clouds rigid registration. In the feature block, global features are obtained from the raw point cloud with a modified version of PointNet. Next, a set of local features are obtained from the point cloud concatenated with the global features. In this case we employ the *set abstraction* modules of PointNet++. In the second block, optimization is carried out using FlowNet and adding a Multi-Layer Perceptron (MLP), a convolutional layer with an average pooling, and finally a dense or fully-connected layer. The result is a rotation in the three axis yaw , pitch , and roll scheme.

In this section, we introduce our proposed architecture for rigid registration, illustrated in Figure 1. The proposed network aligns two 3D point clouds finding the rotation in Euler angles (yaw , pitch , and roll). The point clouds cover the same space and they are of fixed size, so both are expected to be the same size, otherwise, the larger one is downsampled to be the same size as the smaller one. The architecture is divided into two main blocks, the former associated with the feature extraction where

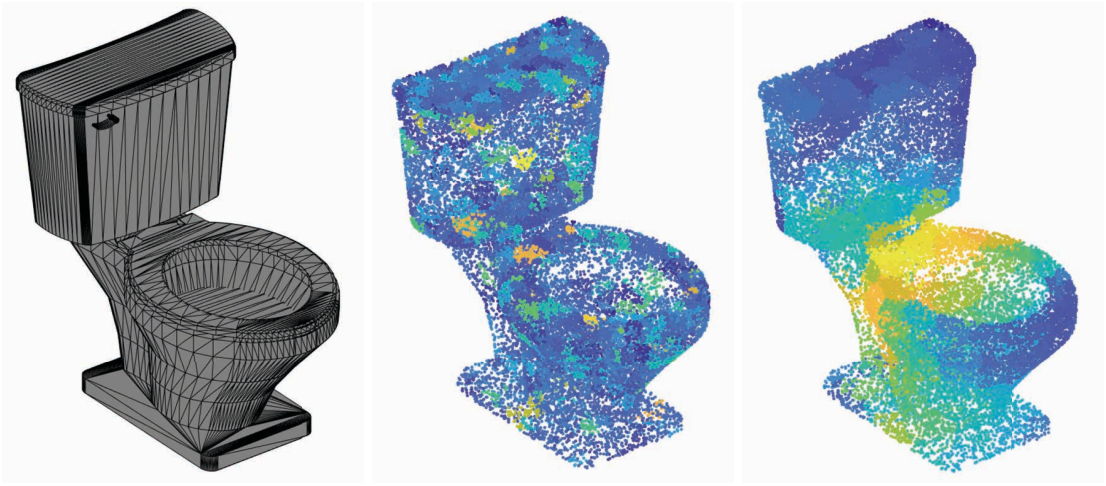

Figure 2: Graphical representation of the features extracted by our method. From left to right: input mesh, global features, and local features, with both sets of features plotted over the up-sampled point cloud. The color indicates the magnitude of the feature. Note that for the global features the representative points are isolated, while the local features are more continuous due to they are extracted hierarchically over sampled regions of the data.

global and local features are inferred from both input datasets; the latter calculates the transformation to best map one set onto the other. It is important to clarify that the divisions are conceptual as the whole architecture is implemented as a set of concatenated layers. The proposed method expects two point clouds as an input, but it is extendable to other 3D inputs.

4.1 Features

In the first block of the pipeline we extract global and local features from each input point cloud. Given two input point clouds of the same size n, $\mathcal{X}$ (source) and $\mathcal{Y}$ (target) defined by points x and y and being $x, y \in R^3$, $\mathcal{X} = \{x_1, x_2, x_3, ..., x_n\}$, $\mathcal{Y} = \{y_1, y_2, y_3, ..., y_n\}$, the first stage of the architecture extracts a pair of vectors of global features $\mathcal{F}_{G\mathcal{X}}$ and $\mathcal{F}_{G\mathcal{Y}}$. Then, a set of local features, $\mathcal{F}_{L\mathcal{X}}$ and $\mathcal{F}_{L\mathcal{Y}}$, are extracted from the previous global ones and the point cloud.

Global features. To extract the global features we use a network based on PointNet [13] which outputs a feature vector describing the point cloud. This network has been used to provide classification and segmentation of point clouds. From the original network, we use all the layers that compute the global features, excluding the last fully connected since we do not need a classification/segmentation (see Figure 3). This network applies to each input a non-linear function to transform them from R^3 to high-dimensional space and outputs a global feature vector which is used in the next stage. As PointNet [13] learns the features from each point independently, structural information between points cannot be captured [5].

Local features. Once the global features have been computed, we employ two *set abstraction* modules from PointNet++ [14] to extract local features of the point cloud. Due to the unstructured nature of a point cloud, understanding the relationship between a point and its neighborhood is crucial because it represents the geometrical properties of an object. The *set abstraction* module combines features from multiple scales by applying PointNet recursively on groups of the point cloud. As a result, a set of local features are extracted. This module is represented in Figure 4 and is composed of three layers. The sampling layer selects points from the point cloud; the grouping layer finds the region of neighborhoods for the previous sampled points, and the PointNet layer extracts features of that region. This module is applied recursively in the point cloud. The result is a matrix of size 128×256 of local features. Then, both local and global features are concatenated with the point cloud information.

A graphical representation of the features extracted by our method is shown in Figure 2. The global features are extracted in a vector of size 1.024 from a point cloud of 1.024 points. For the local features a point cloud of 256 points is employed and we extract 128 x 256 features, which means that each point has a set of 128 features. In this figure for representation purposes, the features of each point have been grouped computing its mean. Also, the feature value has been propagated to an up-sampled point cloud

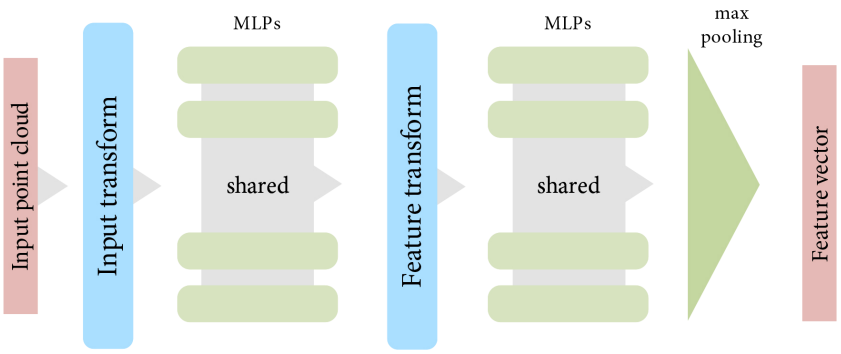

Figure 3: Lightweight representation of PointNet network architecture. We use PointNet to extract a global feature vector of size 1,024 from a point cloud.

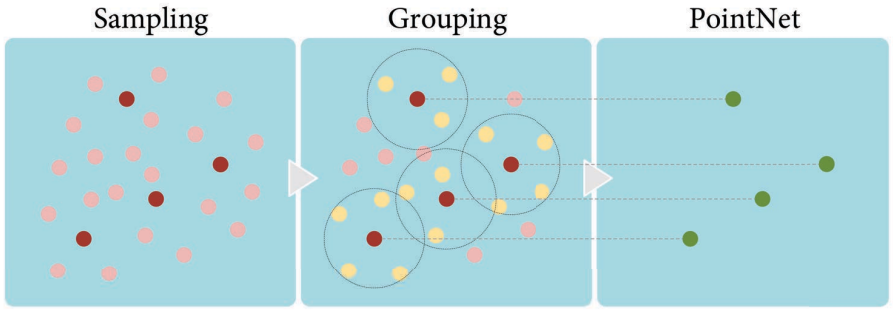

Figure 4: A schematic representation of the *set abstraction* module from PointNet++ [14]. It is composed by three layers: sampling, grouping and PointNet. This module is applied recursively in the point cloud to extract global features.

using K-Nearest Neighbors (KNN) for a better visualization. It is noticeable that the representative points of the global features are isolated while in the local features are continuous.

4.2 Optimization

This block of the architecture optimizes the function in Equation 2, that aims to reduce the Mean Square Error (MSE) between Q and the transformation of P for the given Euler angles e. Formally, the 3D point clouds and the feature vectors are concatenated ($+\!\!+$) to feed the *optimization block*, $P = \mathcal{X} +\!\!+ (\mathcal{F}_{G\mathcal{X}}, \mathcal{F}_{L\mathcal{X}})$ and $Q = \mathcal{Y} +\!\!+ (\mathcal{F}_{G\mathcal{Y}}, \mathcal{F}_{G\mathcal{Y}})$. The network outputs the Euler angles $e = [,,]$ needed to rotate $\mathcal{X}$ so it aligns with $\mathcal{Y}$.

$$E(P,Q) = \arg\min_{e} \frac{1}{n} \sum_{i}^{n} (_i - \chi(e,_i))^2 \tag{2}$$

The first part of the optimization block is based on FlowNet3D [9]. This network provides, given two inputs, the displacement that occurred between them. In our pipeline, the purpose of including part of this network is to estimate the displacement between two feature vectors, or more precisely, to establish the correspondences between features. We employ a Flow Embedding Layer followed by two *set abstraction* modules. In this layer, both point clouds are mixed for the first time in the architecture. The Flow Embedding Layer learns to aggregate the feature similarities of the points to establish motion relationships between them. Then the output of this layer is mixed by the two *set abstraction* modules and up-sampled with a set of *up convs* layers. The next step is a Multi-Layer Perceptron (MLP) followed by a convolutional layer, an average pooling, and four dense layers, which outputs the Euler angles to align both inputs.

5 Implementation Details

The implementation has been carried out using PyTorch 1.7. For extracting the global features we employ the following layers: *input transform*, MLP of dimensions (2,64), *feature transform*, MLP (64,128,1024) and max pool layer. These layers outputs a 1,024 feature vector. For the local features we use two *set abstraction* (SA) modules from PointNet++ [14]. Following the notation of the authors, $SA(K, r, [l_1, ...l_d])$, where K is the number of local regions defined by a ball of radius r, and l_i indicates the fully connected layers that will be used by the PoinNet network of this module. We employ: $SA(16, 0.5, [32, 32, 64])$, $SA(16, 1, [64, 64, 128])$.

The optimization block is composed by: *flow embedding* layer $FE(64, 10, [128, 128, 128])$, $SA(8, 2, [128, 128, 256])$, $SA(8, 4, [256, 256, 512])$ followed by three *set upconv* layers, that perform the opposite operation to *set abstraction*. They include a fourth parameter indicating 2D convolutions. The *set upconv* layers employed are: $SU(8, 2.4, [], [256, 256])$, $SU(8, 1.2, [128, 128, 256], [256])$ and $SU(8, 0.6, [128, 128, 256], [256])$. Then a *feature propagation* (FP) is performed, $FP(256, 256)$. Next we use an MLP of size (256, 128), a convolutional layer of *kernel size = 1* with 3 output channels. Finally, we use an average pooling with dense layers of dimensions (256, 128, 64, 3).

6 Results

In this section, we evaluate the performance of the proposed architecture with different experiments. We evaluate the results of using different training data to clarify whether training with small rotations can improve the result of a more general training; the generalization ability of the method is also tested with input data of unknown objects, and finally, the robustness to noisy data.

Dataset. For training we use ModelNet10 [22] dataset, which is composed of 4,899 CAD models in mesh format from 10 different classes. The dataset is standardly divided into two sets, training and test. It contains 3,991 models for training and 908 for testing. Besides, ModelNet10 is a subset of ModelNet40, which includes 40 classes in total, thirty more than ModelNet10. For testing the generalization ability of the network we employ the objects in test subsets of the classes not present in ModelNet10 (1,560 meshes in total) to evaluate the behavior of the method with unknown categories.

Pre-processing. Every object is uniformly sampled to have 1,024 points coming from the surface of the mesh. Then, the point cloud is introduced into a unit sphere, which means that is centered and normalized between the range $[-1, 1]$. Finally, a random rotation around the Z-axis is applied in order not to condition the point of view. After these pre-processing steps, we proceed to set the pair source/target that will be passed to the network. For that purpose, we randomly select the axis of rotation and the angle to apply the transformation to the target point cloud.

Loss. The loss function minimizes the difference between the source $\mathcal{X}$ and the target $\mathcal{Y}$ point clouds. We measure this difference in terms of distance. The output of the network is the Euler angles, which are applied to the source input give a measure of the error. This is expressed as the Mean Square Error (MSE) of the point-to-point distance between the target and the rotated source point clouds (Equation 3).

$$\frac{1}{n}\sum_{i}^{n}[y_i - \chi(\boxed{\ }, x_i)]^2 \tag{3}$$

Metrics. We employ the distance error from point-to-point correspondences to evaluate the results. To calculate the distance error we use the Mean Square Error (MSE) of the distance between each point in the target set $\mathcal{Y}$ and its correspondence in transformed $\mathcal{X}$ set. The error in terms of angle of rotation is also evaluated. To this end, we compute the Principal Component Analysis (PCA) of the transformed source and the target. Then, the angle difference between each component is computed and we selected the maximum one to get a measure of the rotation difference in degrees between both sets. All of these metrics are obtained from the data normalized into a unit sphere, in order to detach the evaluation of the method from the dimensions of the objects.

6.1 Groundtruth training

We first train the network using ModelNet10 with a random rotation for each object with angles between 0 and 60 degrees. After that, we evaluate the performance of the network using the test subset of the above-mentioned dataset, obtaining the results in Figure 5. It is easy to observe how the error increases when

the network receives rotations not seen during training. The validation of this training with rotations between 0 and 60 degrees with the test classes reaches an average rotation error of 1.18 degrees. The error by category are shown in Table 1, with the highest error of 1.49 degrees in category *table*.

1.1

Table 1: Distance MSE and rotation error for each category of ModelNet10. Observe that the average error in the distance is under 5E-05, and the error in orientation is 1.18 degrees for a range of experiments with random rotations between 0 and 60 degrees.

	category										
	bathtub	*bed*	*chair*	*desk*	*dresser*	*monitor*	*night-stand*	*sofa*	*table*	*toilet*	**Average**
Distance MSE	4.92E-05	4.41E-05	5.03E-05	5.26E-05	5.18E-05	3.83E-05	5.04E-05	3.99E-05	6.36E-05	5.56E-05	**4.94E-05**
Rotation Error (deg.)	1.16	1.15	1.29	1.31	1.04	1.03	1.08	1.10	1.49	1.15	**1.18**

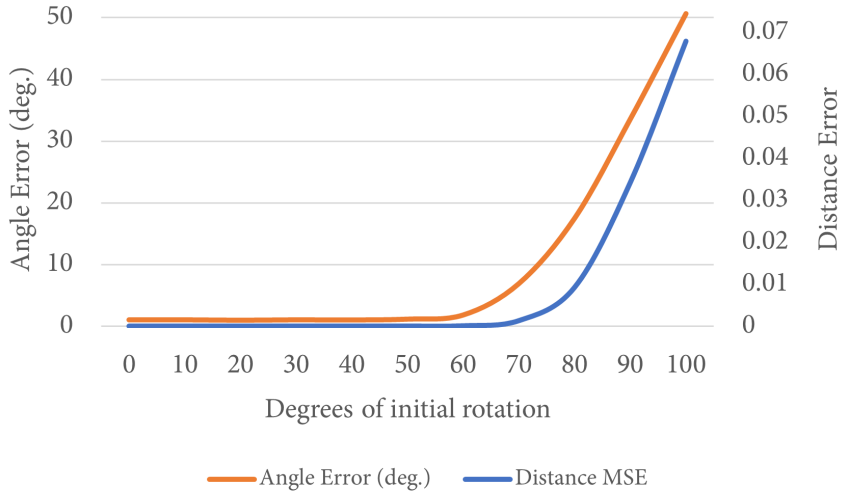

Figure 5: Distance and angle error of the architecture trained with random rotations between 0 and 60 degrees. Notice how the error increases as non-seen angles are introduced.

6.2 Train on rotation ranges

We compare the results of the previous section with different trainings on specific rotation ranges to evaluate if a training with small angle ranges outperforms (in terms of error) a general one for that specific rotations. Figure 6 depicts three cases with different initial positions for visual evaluation of the performance. Observe the accurate alignment for the different rotations.

Figure 7 presents 7 different trainings and their average distance MSE for 6 tests on different ranges for the initial rotation. For instance, the error obtained using the *Training 0-10* means that the training of the network has been done using only cases of rotations in the range of 0 to 10 degrees. It produces acceptable results in the test cases for initial rotations similar to the ones used for training (0-10 degrees). However, for the same training, the error grows as the angle deviates from the training range, but always having a maximum average error below 0.1 units. For the rest of the trainings, except the *Training 0-60*, each performs best in their same initial rotation range and always achieves results under 0.1 units in MSE. For the most general case, i.e. *Training 0-60* which means that the network has been trained using angles between 0 and 60, the performance is always under or equal to 1E-4. This MSE error is better than all the other cases, proving that the network generalizes properly, even better than when it was trained for specific rotation ranges.

Figure 8 depicts the average rotation error in degrees for the seven different training ranges. The error has been calculated as the maximum difference in degrees of the principal components of the source and target sets. The results show that, when the network is trained with a range in the extremes, i.e. 0-10 degrees or 50-60 degrees, the average registration performance is the lowest. Nevertheless, for the most general case where the training includes all angles between 0 and 60 degrees, the average error is 1.18, the lowest of all the cases, proving again the general capabilities of the proposal.

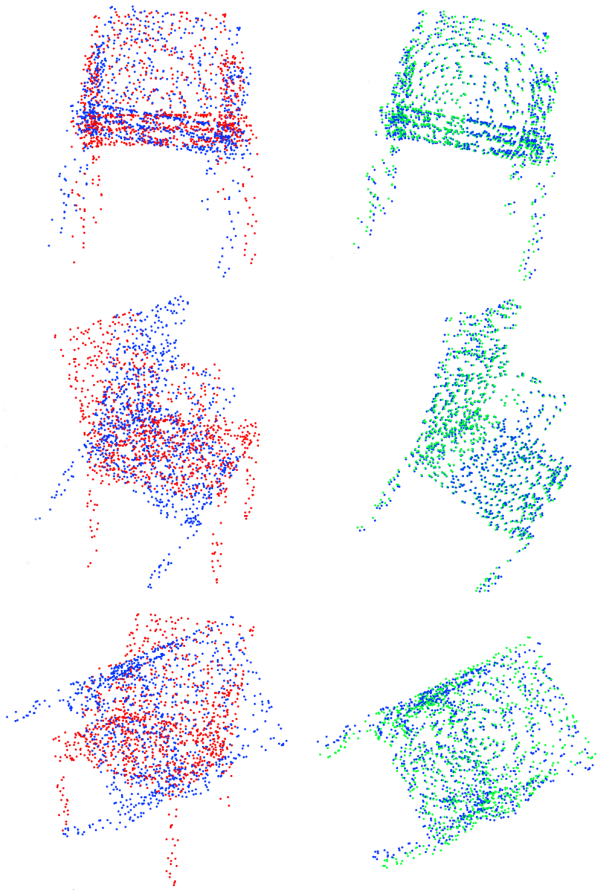

Figure 6: Visual evaluation of the registration result. The first column shows the initial position of source (red) and target (blue) point clouds, the second shows the registration result (green). The rows show the registration cases for 10, 30, 60 degrees of initial rotation.

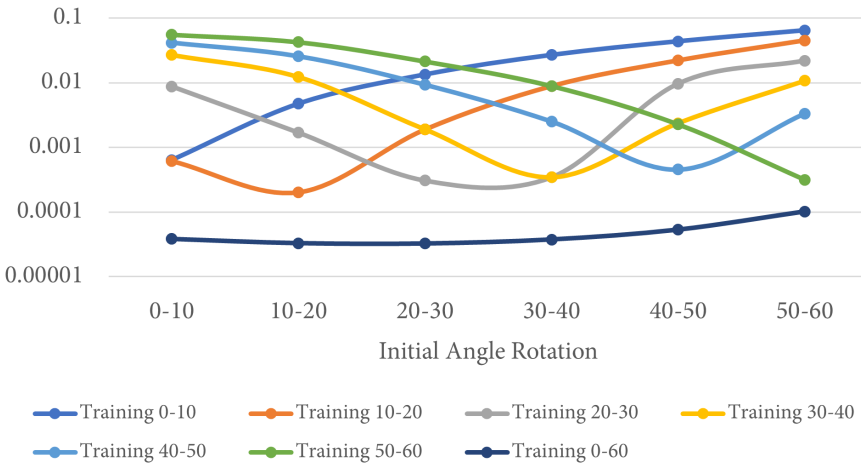

Figure 7: (Logarithmic scale). The figure depicts the different average distance MSE errors for seven trainings on different angle ranges (six specific ranges and one full range) and evaluated in the six ranges. Notice that the best case is achieved by the network trained with the complete range of angles, outperforming even the specific cases.

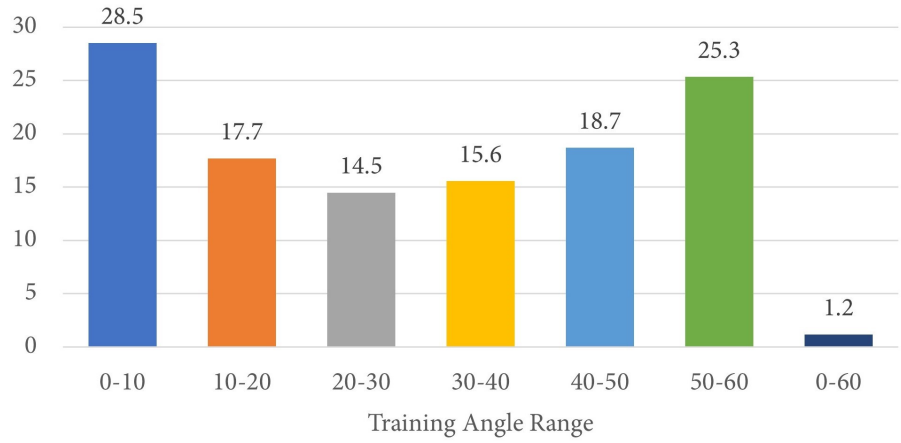

Figure 8: The figure shows the average rotation error in degrees. The error is calculated as the difference between the principal components orientation. Each bar represents the total average error for a network trained in a specific range and evaluated for the whole range of angles.

6.3 Robustness to noise

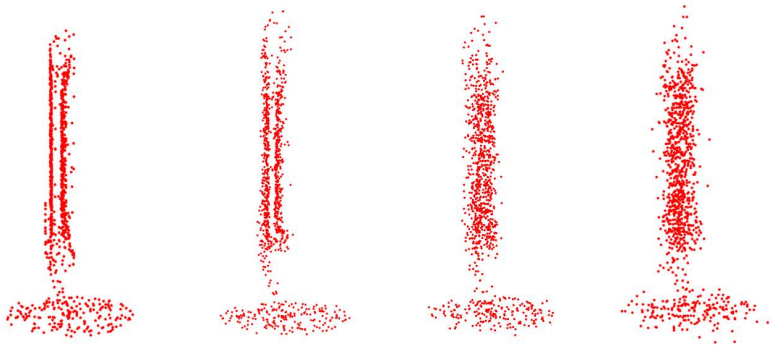

Figure 9: Four side views of an object of class *monitor* with noise distributions computed with different λ values $(0, 0.02, 0.06, 0.1)$.

One evaluation criterion of computer vision algorithms is the robustness to noise (see Figure 9). Acquired data almost always contain noise, so the algorithms need to handle this distortion of the data. We have tested our method with an incremental level of noise to evaluate its robustness. In order to generate the 3D Gaussian noise added to each point of the point cloud, we calculate the standard deviation of the noise relative to the size of the object. This is, the standard deviation of the noise $STD = \lambda\sigma$, depends on the σ (standard deviation of the point cloud) and the scale factor λ. Figure 10 shows how the angle and distance error increases as the noise gets higher. It is noticeable that for low values of noise, the error remains low. The noise is only applied to one of the input point clouds.

6.4 Unseen classes

The ability of a network to manage with good results objects not seen during training is called generalization. The network is able to generalize to solve registration problems in object categories that are unknown. As ModelNet10 is a subset of ModelNet40, we tested registration performance with thirty classes not included in the training subset to test the generalization ability of our method. Figure 11 overlays the error obtained with unseen categories (continuous line) and the error obtained with the categories of ModelNet10 used for training (dotted line). It is possible to observe how the error on unseen objects is only slightly larger than that on the classes from the original test set. The specific information of these errors broken down by category is shown in Table 2. It is noticeable that it reaches the highest error in the category *person*, probably because it is the most different category of the whole set.

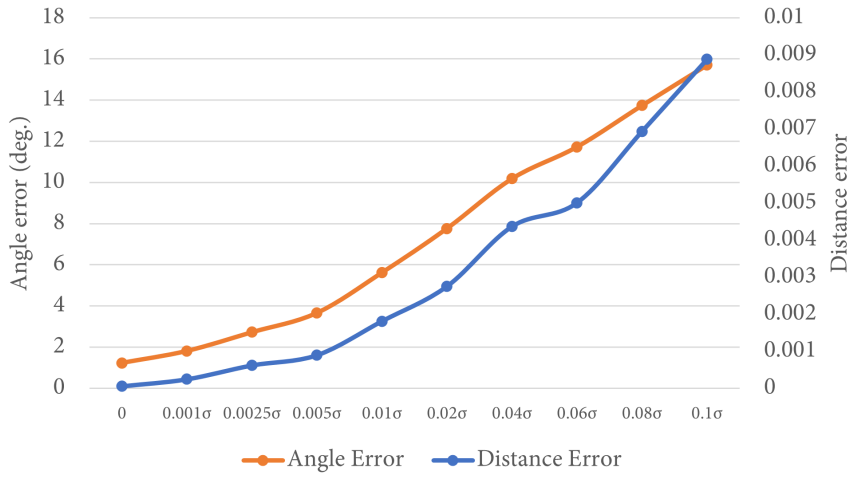

Figure 10: Angle error in degrees and distance MSE error for different levels of noise applied to the source input point cloud.

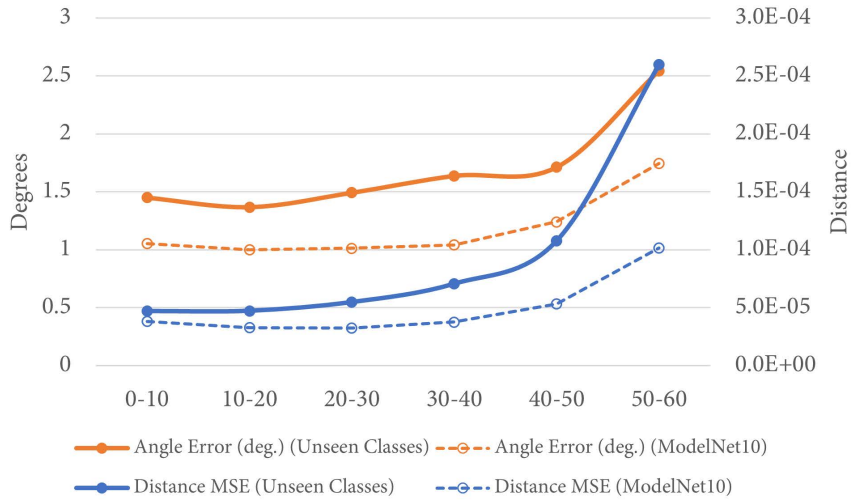

Figure 11: This figure overlays the error obtained with unseen categories (continuous line) and the error of the same categories used for training (dotted line). The distance MSE error is shown in blue, and the angle error in degrees in orange. This data has obtained through different tests with each angle range. Notice how the angle error difference remains almost constant, while the distance error is close to the groundtruth until the last range of angles.

1.1

Table 2: Distance and angle errors for unknown categories of ModelNet40. Notice the average distance and rotation errors are low for almost all cases.

					category					
(lr)2-11	*airplane*	*bench*	*bookshelf*	*bottle*	*bowl*	*car*	*cone*	*cup*	*curtain*	*door*
(lr)1-11 Distance MSE	5.81E-05	8.57E-05	8.14E-05	1.89E-04	4.08E-04	7.48E-05	2.54E-04	6.06E-05	6.20E-05	1.02E-04
Rotation Error (deg.)	1.60	1.52	1.44	3.11	1.88	1.50	2.58	1.12	1.30	1.72
(lr)1-11										

(lr)1-11					**category**					
(lr)2-11	*flower-pot*	*glass-box*	*guitar*	*keyboard*	*lamp*	*laptop*	*mantel*	*person*	*piano*	*plant*
(lr)1-11 Distance MSE	1.16E-04	5.09E-05	5.95E-05	9.14E-05	3.52E-04	5.30E-05	7.04E-05	1.38E-04	5.20E-05	2.36E-04
Rotation Error (deg.)	1.91	1.02	2.27	1.96	3.16	1.18	1.26	8.32	1.22	5.47
(lr)1-11										

(lr)1-11						**category**					
(lr)2-11	*radio*	*range-hood*	*sink*	*stairs*	*stool*	*tent*	*tv-stand*	*vase*	*wardrobe*	*xbox*	**Average**
(lr)1-12 Distance MSE	6.02E-05	5.90E-05	4.19E-05	2.03E-04	1.44E-04	6.16E-05	5.85E-05	5.07E-04	4.31E-05	5.22E-05	**9.79E-05**
Rotation Error (deg.)	1.59	1.30	1.02	2.41	2.14	1.41	1.17	2.31	1.03	1.13	**1.70**

7 Conclusion

In this work, we present a novel deep registration network based on two main stages, feature estimation and transformation optimization. The first extracts global features from the input, which are later concatenated with the point cloud for local feature extraction. The global and local features are concatenated to the original input to feed the optimization stage. In the optimization stage, we employ a network based on FlowNet to capture the matches between the two inputs and their calculated features. This is followed by an MLP, convolution layer, and dense layer to, finally, output a rotation in Euler angles yaw, pitch, and roll. We evaluate the proposal with ModelNet10 and ModelNet40 datasets and various levels of rotation achieving average distance MSE under 1E-4 units of the normalized data into a unit sphere. In angle difference, the proposed network is able to align the input data with an average rotation error of 1.18 degrees. Also, we have carried out some experiments including noise to the 3D data in order to prove the robustness. The results show a good accuracy even with a significant amount of noise and with unknown objects.

As future work, in short term we plan to include more data from other datasets, such as ShapeNet, and address the problem of partial data registration. We consider interesting an ablation study to evaluate how the performance changes leaving out some parts of the architecture. Moreover, we are going to compare our proposal with state-of-the-art networks and traditional methods. Also, we plan to extend our method to be robust with larger angles of initial rotation. For this purpose, we plan to study variations in the architecture of our method to improve the generalization ability in large transformations.

Acknowledgements

This work was supported by a Valencian Regional project (GV/2020/056), a Valencian Grant for Ph.D. studies (ACIF/2017/223) and a Valencian Grant for predoctoral internships (BEFPI/2020/001). This work was also supported by the Spanish State Research Agency (AEI) and the European Regional Development Fund (FEDER) under project TIN2017-89069-R. The DGX-A100 cluster at the University of Alicante has been used for the research, funded by the "Generalitat Valenciana" and the "European Union" through the IDIFEDER/2020/003 project.

References

[1] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. Pointnetlk: Robust & efficient point cloud registration using pointnet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7163–7172, 2019.

[2] Wen-Chung Chang and Van-Toan Pham. 3-d point cloud registration using convolutional neural networks. *Applied Sciences*, 9(16):3273, 2019.

[3] Li Ding and Chen Feng. Deepmapping: Unsupervised map estimation from multiple point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8650–8659, 2019.

[4] Gil Elbaz, Tamar Avraham, and Anath Fischer. 3d point cloud registration for localization using a deep neural network auto-encoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4631–4640, 2017.

[5] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[7] Akiyoshi Kurobe, Yusuke Sekikawa, Kohta Ishikawa, and Hideo Saito. Corsnet: 3d point cloud registration by deep neural network. *IEEE Robotics and Automation Letters*, 5(3):3960–3966, 2020.

[8] Xueqian Li, Jhony Kaesemodel Pontes, and Simon Lucey. Deterministic pointnetlk for generalized registration. *arXiv preprint arXiv:2008.09527*, 2020.

[9] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019.

[10] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. Vancouver, British Columbia, 1981.

[11] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7193–7203, 2020.

[12] Dhanya S Pankaj and Rama Rao Nidamanuri. A robust estimation technique for 3d point cloud registration. *Image Analysis & Stereology*, 35(1):15–28, 2016.

[13] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[14] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++ deep hierarchical feature learning on point sets in a metric space. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5105–5114, 2017.

[15] S. Rusinkiewicz and M. Levoy. Efficient variants of the icp algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling*, pages 145–152, 2001. doi:10.1109/IM.2001.924423.

[16] Marcelo Saval-Calvo, Sergio Orts-Escolano, Jorge Azorin-Lopez, Jose Garcia-Rodriguez, Andres Fuster-Guillo, Vicente Morell-Gimenez, and Miguel Cazorla. Non-rigid point set registration using color and data downsampling. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[17] Marcelo Saval-Calvo, Jorge Azorin-Lopez, Andres Fuster-Guillo, Victor Villena-Martinez, and Robert B Fisher. 3d non-rigid registration using color: color coherent point drift. *Computer Vision and Image Understanding*, 169:119–135, 2018.

[18] Gary KL Tam, Zhi-Quan Cheng, Yu-Kun Lai, Frank C Langbein, Yonghuai Liu, David Marshall, Ralph R Martin, Xian-Fang Sun, and Paul L Rosin. Registration of 3d point clouds and meshes: a survey from rigid to nonrigid. *IEEE transactions on visualization and computer graphics*, 19(7):1199–1217, 2012.

[19] Victor Villena-Martinez, Sergiu Oprea, Marcelo Saval-Calvo, Jorge Azorin-Lopez, Andres Fuster-Guillo, and Robert B Fisher. When deep learning meets data alignment: A review on deep registration networks (drns). *Applied Sciences*, 10(21):7524, 2020.

[20] Yue Wang and Justin M Solomon. Deep closest point: Learning representations for point cloud registration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3523–3532, 2019.

[21] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019.

[22] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[23] Wentao Yuan, Benjamin Eckart, Kihwan Kim, Varun Jampani, Dieter Fox, and Jan Kautz. Deepgmr: Learning latent gaussian mixture models for registration. In *European Conference on Computer Vision*, pages 733–750. Springer, 2020.

[24] Andy Zeng, Shuran Song, Matthias Nießner, Matthew Fisher, Jianxiong Xiao, and Thomas Funkhouser. 3dmatch: Learning local geometric descriptors from rgb-d reconstructions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1802–1811, 2017.

[25] Zhiyuan Zhang, Yuchao Dai, and Jiadai Sun. Deep learning based point cloud registration: an overview. *Virtual Reality & Intelligent Hardware*, 2(3):222–246, 2020. ISSN 2096-5796. doi:https://doi.org/10.1016/j.vrih.2020.05.002. URL https://www.sciencedirect.com/science/article/pii/S2096579620300383. 3D Visual Processing and Reconstruction Special Issue.