PCRNet: Point Cloud Registration Network using PointNet Encoding

Vinit Sarode[1*] Xueqian Li[1*] Hunter Goforth[3] Yasuhiro Aoki[2]
Rangaprasad Arun Srivatsan[4] Simon Lucey[1,3] Howie Choset[1]
[1]Carnegie Mellon University [2]Fujitsu Laboratories Ltd. [3]Argo AI. [4]Apple.
{vsarode, xueqianl}@andrew.cmu.edu aoki-yasuhiro@fujitsu.com
{hgoforth, slucey, choset}@cs.cmu.edu aruns@apple.com

Abstract

PointNet has recently emerged as a popular representation for unstructured point cloud data, allowing application of deep learning to tasks such as object detection, segmentation and shape completion. However, recent works in literature have shown the sensitivity of the PointNet representation to pose misalignment. This paper presents a novel framework that uses the PointNet representation to align point clouds and perform registration for applications such as tracking, 3D reconstruction and pose estimation. We develop a framework that compares PointNet features of template and source point clouds to find the transformation that aligns them accurately. Depending on the prior information about the shape of the object formed by the point clouds, our framework can produce approaches that are shape specific or general to unseen shapes. The shape specific approach uses a Siamese architecture with fully connected (FC) layers and is robust to noise and initial misalignment in data. We perform extensive simulation and real-world experiments to validate the efficacy of our approach and compare the performance with state-of-art approaches. Code is available at https://github.com/vinits5/pcrnet

1. Introduction

3D point clouds are ubiquitous today, thanks to the development of lidar, stereo cameras and structured light sensors. As a result there has been a growing interest in developing algorithms for performing classification, segmentation, tracking, mapping, etc. directly using point clouds. However, the inherent lack of structure presents difficulties in using point clouds directly in deep learning architectures. Recent developments such as PointNet [28] and its variants [29] have been instrumental in overcoming some of these difficulties, resulting in state-of-the-art methods for object detection and segmentation tasks [27, 39].

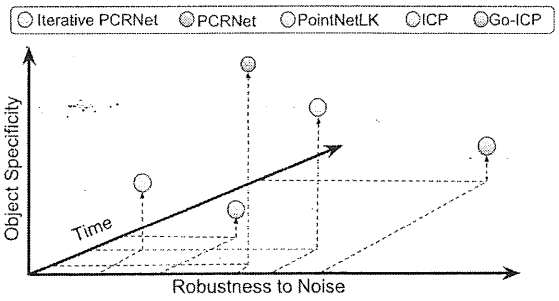


Figure 1: Comparison of different registration methods based on their robustness to noise and computation time with respect to object specificity. The iterative version of PCRNet exploits object specificity to produce accurate results. The PCRNet without iterations is computationally faster but compromises a little on accuracy. PointNetLK [2] exhibits good generalizability, but is not robust to noise. ICP [7] is object-shape agnostic and slow for large point clouds, while Go-ICP [37] is computationally expensive.

Prior works [39, 2] have observed that robust performance of PointNet requires minimal misalignment of the point clouds with respect to a canonical coordinate frame. While this is present in synthetic datasets such as *ModelNet40* [35], real world data is seldom aligned to some canonical coordinate frame. Inspired by recent works on iterative transformer network (IT-Net) [39] and PointNetLK [2], this work introduces point cloud registration network (PCRNet), a framework for estimating the misalignment between two point clouds using PointNet as an encoding function. It is worth noting that our approach can directly process point clouds for the task of registration, without the need for hand crafted features [31, 13], voxelization [25, 22] or mesh generation [34]. Depending on the prior knowledge of the shape formed by the point

*equal contribution

clouds, presence of noise, and computational requirements, our framework provides a well-suited approach for each scenario. Our framework also provides additional context for PointNetLK (see Fig. 1) within a family of PointNet-based registration algorithms.

Our approach uses PointNet in a Siamese architecture to encode the shape information of a template and a source point cloud as feature vectors, and estimates the pose that aligns these two features using data driven techniques. Using shape-specific prior information in the training phase allows us to be robust to noise in the data, compared to shape agnostic methods such as iterative closest point (ICP) [7] and its variants [30]. Furthermore, we find that the PointNetLK approach, which uses classical alignment techniques such as Lucas-Kanade (LK) algorithm [21, 5] for aligning the PointNet features, produces good generalizability to shapes unseen in training but is not robust to noise. Unlike conventional registration approaches such as ICP, our approach does not require costly closest point correspondence computations, resulting in improved computational efficiency and robustness to noise. Further, the approach is fully differentiable which allows for easy integration with other deep networks and can be run directly on GPU without need for any CPU computations.

In summary, our contributions are (1) presenting two novel point cloud alignment algorithms which utilize a PointNet representation for effective registration and (2) a thorough experimental validation of these two approaches including comparison against PointNetLK, ICP, and Go-ICP, on both simulated and real-world data

2. Related Work

Classical registration. Iterative Closest Point (ICP) [7] remains one of the most popular techniques for point cloud registration, as it is straightforward to implement and produces adequate results in many scenarios. Extensions of ICP have added features such as increased computational efficiency [20, 9, 4] or improved accuracy [37]. However, nearly all ICP variants rely on explicit computation of closest points correspondences, a process which scales poorly with the number of points. Additionally, ICP is not differentiable (due to the requirement to find discrete point correspondences) and thus cannot be integrated into end-to-end deep learning pipelines, inhibiting the ability to apply learned descriptors for alignment.

Interest point methods compute and compare local descriptors to estimate alignment [13, 15, 16]. Interest point methods have the advantage of being computationally favorable, however, their use is often limited to point cloud data having identifiable and unique features which are persistent between point clouds that are being registered [23, 26, 31].

Globally optimal methods seek to find optimal solutions which cannot reliably be found with iterative techniques such as ICP [18, 19, 24]. A representative example which we use as a baseline is Go-ICP [37], a technique using branch-and-bound optimization. These techniques are characterized by extended computation times, which largely precludes their use in applications requiring real-time speed.

PointNet. PointNet [28] is the first deep neural network which processes point clouds directly, as opposed to alternative representations such as 2D image projections of objects [36, 8, 14], voxel representations [25, 35, 41] or graph representations [34]. Within larger network architectures, PointNet has proven to be useful for tasks including classification, semantic segmentation, object detection [27], and completion of partial point clouds [40]. An extension to PointNet for estimating local feature descriptors is described in [29]. Wentao *et al.* introduced iterative transformer network (IT-Net) [39] which uses PointNet to estimate a canonical orientation of point clouds to increase classification and segmentation accuracy. Global descriptors from PointNet are used in [1] for place recognition from 3D data. The loss function used in deep networks for point cloud processing is an important consideration, which we discuss more in Section 3. Earth Mover Distance (EMD) and Chamfer Distance (CD) are introduced in [12], while in [2] a Frobenius norm of a difference between estimated and ground truth transformation matrices is used.

Learned registration. Discriminative optimization [32] and the recent inverse composition discriminative optimization [33] combine hand-crafted feature vectors and learned map sets for the task of point cloud registration. The shortcoming of these approaches is a quadratic complexity in the number of points, and a lack of generalization due to the feature vector and registration maps both being learned. Deep auto-encoders are used to extract local descriptors for registration of large outdoor point clouds in [10]. In [38], a network is designed which learns both interest point detection and descriptor computation, for a descriptor-matching registration approach. Wang *et al.* perform convolution operations on the edges that connect neighboring point pairs, by using a local neighborhood graph [34]. PointNetLK [2], which performs registration of arbitrary point clouds by minimizing the distance between the fixed-length, global descriptors produced by PointNet, is the most closely related to our work and serves as a baseline.

3. Method

Point clouds are highly unstructured with ambiguities in the order permutations. While performing classification using PointNet, a symmetric pooling function such as max

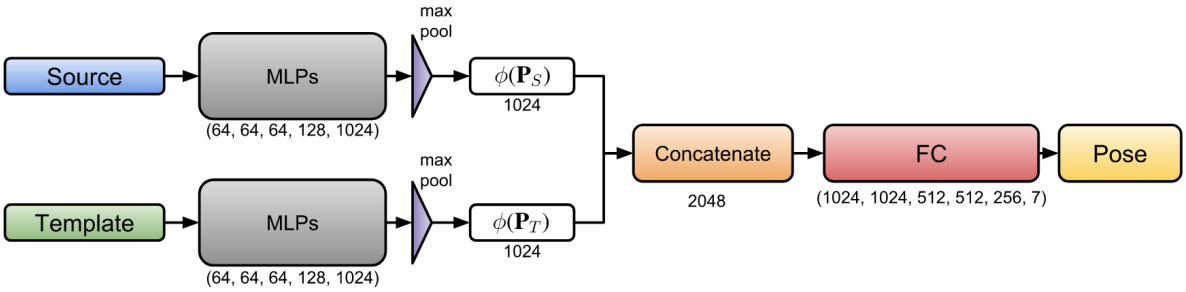


Figure 2: PCRNet Architecture: The model consists of five MLPs having size (64, 64, 64, 128, 1024). The source and template point clouds are sent as input through a twin set of MLPs, arranged in a Siamese architecture. Using a max-pooling function, we obtain global features. Weights are shared between MLPs. These features are concatenated and provided as an input to five fully connected layers 1024, 1024, 512, 512, 256, and an output layer of size 7. The first three output values represent the translation and the last four after normalization represent the rotation quaternion.

pool is used to afford invariance to input permutation. The output vector of the symmetry function is referred to as a global feature vector. We will denote the template point cloud $\mathbf{P}_T$ and source $\mathbf{P}_S$, and the PointNet function ϕ. Since the global feature vectors contain the information about the geometry as well as the orientation of the point clouds, the transformation between two point clouds can be obtained by comparing the feature vectors. In other words, we calculate the rigid-body transformation $\mathbf{T} \in SE(3)$, that minimizes the difference between $\phi(\mathbf{P}_S)$ and $\phi(\mathbf{P}_T)$.

3.1. PCRNet

This section introduces the PCRNet architecture. A block diagram of the architecture is shown in Fig. 2. The point cloud data obtained from a sensor is referred to as the *source* and the point cloud corresponding to the known model of the object to be registered is referred to as the *template*. The model consists of five multi-layered perceptrons (MLPs) similar to the PointNet architecture having size 64, 64, 64, 128, 1024. The MLPs are arranged similar to a Siamese architecture [17]. Both source $\mathbf{P}_S$ and template $\mathbf{P}_T$ are given as input to the MLPs which are arranged in Siamese architecture and symmetric max-pooling function is used to find the global feature vectors $\phi(\mathbf{P}_S)$ and $\phi(\mathbf{P}_T)$. Weights are shared between MLPs used for source and template.

The global features are concatenated and given as an input to a number of fully connected layers. In this work, we choose five fully connected layers, as they seemed to be sufficient enough for robust performance. We tried using lesser number of FC layers, but the performance of the network was poor.

The FC layers shown by the red block in Fig. 2 has five hidden layers, 1024, 1024, 512, 512, 256, and an output layer of size 7 whose parameters will represent the estimated transformation $\mathbf{T}$. The first three of the output values we use to represent the translation vector $\mathbf{t} \in \mathbb{R}^3$ and last four represents the rotation quaternion $\mathbf{q} \in \mathbb{R}^4$, $\mathbf{q}^T\mathbf{q} = 1$. In this way, the transformation $\mathbf{T}$ which aligns $\phi(\mathbf{P}_S)$ and $\phi(\mathbf{P}_T)$ is estimated with a single forward pass, or single-shot, through the network. The single-shot design lends itself particularly well to high-speed applications, which will be discussed further in Section 4.

Note that if we were to replace the FC layers in the network with a traditional alignment algorithm such as the Lucas-Kanade [21, 5], the resulting implementation would be similar to the PointNetLK [2].

3.2. Iterative PCRNet

In this section, we present a network with an iterative scheme similar to ICP and Lucas-Kanade for image alignment as shown in Fig. 3. We retain the structure but modify the number of layers from the single-shot PCRNet. For the iterative implementation, the fully connected layers have three hidden layers with size 1024, 512, 256, and an output layer of size seven. Also, there is an additional dropout layer before the output layer, to avoid overfitting. We empirically observe that introducing iterations, allows us to use lesser number of hidden layers compared to PCRNet, and yet obtain robust performance.

In the first iteration, original source and template point clouds are given to PCRNet which predicts an initial misalignment $\mathbf{T}(1)$ between them. For the next iteration, $\mathbf{T}(1)$ is applied to the source point cloud and then the transformed source and the original template point clouds are given as input to the PCRNet. After performing n iterations, we find the overall transformation between the original source and template point clouds by combining all the poses in each iteration:

$$\mathbf{T} = \mathbf{T}(n) \times \mathbf{T}(n-1) \times \cdots \times \mathbf{T}(1). \tag{1}$$

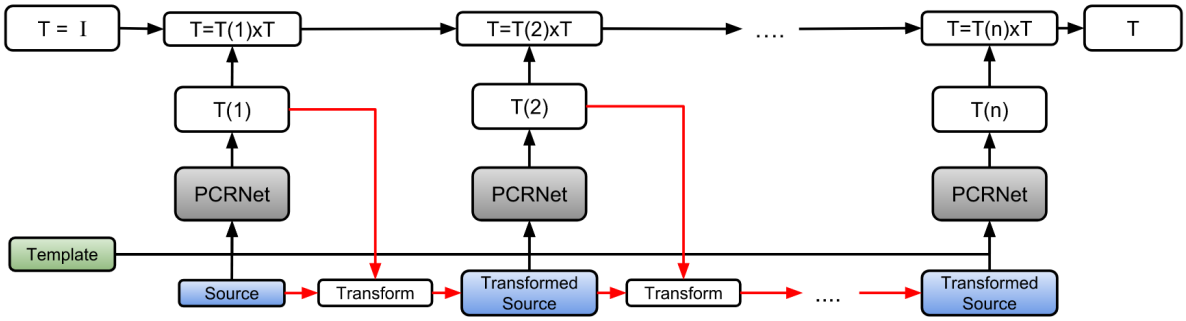


Figure 3: Iterative PCRNet Architecture: The iterative PCRNet uses a modified form of PCRNet described in Fig. 2 and iteratively improves the estimate of PCRNet. In the first iteration, the source and template point clouds are given to PCRNet which predicts an initial misalignment $\mathbf{T}(1)$. The source point cloud is transformed using $\mathbf{T}(1)$ and the original template are given as input to the PCRNet, in the next iteration. After performing n iterations, we combined the poses from each iteration to find the overall transformation between the original source and template.

3.3. Loss Function

The aim of the loss function used to train registration networks should be minimization of distance between the corresponding points in source and template point cloud. This distance can be computed using Earth Mover Distance (EMD) function,

$$\text{EMD}(\mathbf{P}_S^{\text{est}}, \mathbf{P}_T) = \min_{\psi:\mathbf{P}_S^{\text{est}} \to \mathbf{P}_T} \frac{1}{|\mathbf{P}_S^{\text{est}}|} \sum_{x \in \mathbf{P}_S^{\text{est}}} \|x - \psi(x)\|_2, \quad (2)$$

where $\mathbf{P}_T$ is the template point cloud and $\mathbf{P}_S^{\text{est}}$ is the source point cloud $\mathbf{P}_S$, transformed by the estimated transformation $\mathbf{T}$ from Eq. 1. This function finds a bijection ψ and minimizes the distance between corresponding points based on ψ. While there are many other choices for loss function including Frobenius norm [2], and PoseLoss [36], we find EMD loss is most effective for learning on the training data described in Section 4 for both iterative and single-shot PCRNet.

3.4. Training

In this work, we use *ModelNet40* dataset [35] to train the network. This dataset contains CAD models of 40 different object categories. We uniformly sample points based on face area and then used farthest point algorithm [11] to get a complete point cloud. We train the networks with three different types of datasets as following – (1) Multiple categories of objects and multiple models from each category, (2) Multiple models of a specific category, (3) A single model from a specific category. We choose these 3 cases to showcase the performance of the PointNet-based approaches on data with differing levels of object-specificity.

We train the iterative PCRNet with 8 iterations during training, observing that more than 8 produced little improvement to results. In some experiments the training data was corrupted with Gaussian noise, which will be discussed in detail in Sec. 4.2. The networks are trained for 300 epochs, using a learning rate of 10^{-3} with an exponential decay rate of 0.7 after every 3×10^6 steps and batch size 32. The network parameters are updated with Adam Optimizer on a single NVIDIA GeForce GTX 1070 GPU and a Intel Core i7 CPU at 4.0GHz.

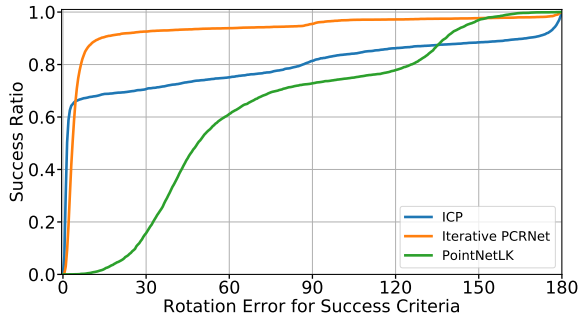
4. Results

In this section, we compare performance of our networks on test data with multiple object categories, a specific object category, a specific object from training dataset and objects unseen in training. We use models from *ModelNet40* dataset [35] for the following experiments. Template point clouds are normalized into a unit box and then their mean is shifted to origin. We randomly choose 5070 transformations with Euler angles in the range of $[-45°, 45°]$ and translation values in the range of [-1, 1] units. We apply these rigid transformations on the template point clouds to generate the source point clouds. We allow a maximum of 20 iterations for both iterative PCRNet and PointNetLK while performing tests, while the maximum iterations for ICP was chosen as 100. In addition to maximum iterations, we also use the convergence criteria

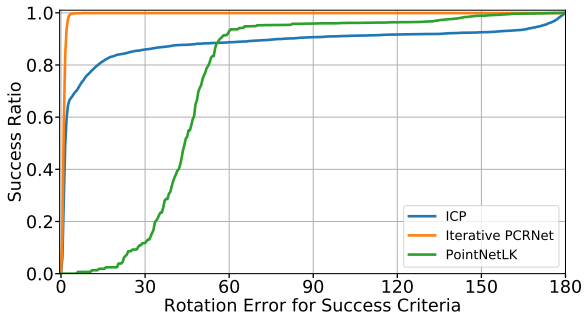
$$\left\|\mathbf{T}_i \mathbf{T}_{i-1}^{-1} - \mathbf{I}\right\|_F < \epsilon,$$

where $\mathbf{T}_i, \mathbf{T}_{i-1} \in SE(3)$ are the transformations predicted in current and previous iterations, and the value of ϵ is chosen to be 10^{-7}.

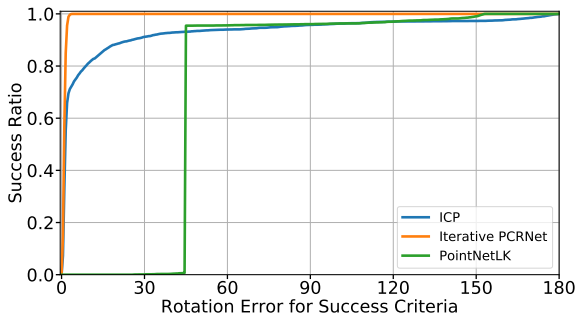
In order to evaluate the performance of the registration algorithms, a metric we use is area under the curve (AUC). Plots showing success ratio versus success criteria on rotation error (in degrees) are generated for ICP, iterative PCRNet and PointNetLK. Fig. 4 shows examples of these curves. The area below the curves in these plots, divided



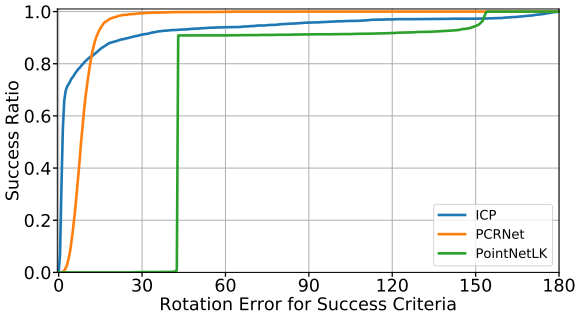
(a) Training and testing: Multiple object categories with noise.



(b) Training and testing: Multiple models of a category with noise



(c) Training and testing: Only one model with noise



(d) Trained on one model without noise and tested on data with noise

Figure 4: Results for Section 4.2. The y-axis is the ratio of experiments that are successful and the x-axis shows value of the maximum rotation error that qualifies the estimation to be a success. (a), (b) and (c) shows results for comparisons of iterative PCRNet with ICP and PointNetLK using three different types of datasets. We observe superior performance of iterative PCRNet as our network has more model/category specific information. (d) PCRNet which has not seen noise during training but tested with noisy data also shows good performance and is faster than ICP and PointNetLK. Speed considerations are discussed in Sec. 4.3.

by 180 to normalize between 0 and 1, is defined as AUC [1]. AUC expresses a measure of success of registration and so the higher the value of AUC, the better the performance of the network. We measure the misalignment between predicted transformation and ground truth transformation and express it in axis-angle representation and we report the angle as rotation error. As for the translation error, we report the L2 norm of the difference between ground truth and estimated translation vectors.

4.1. Generalizability versus specificity

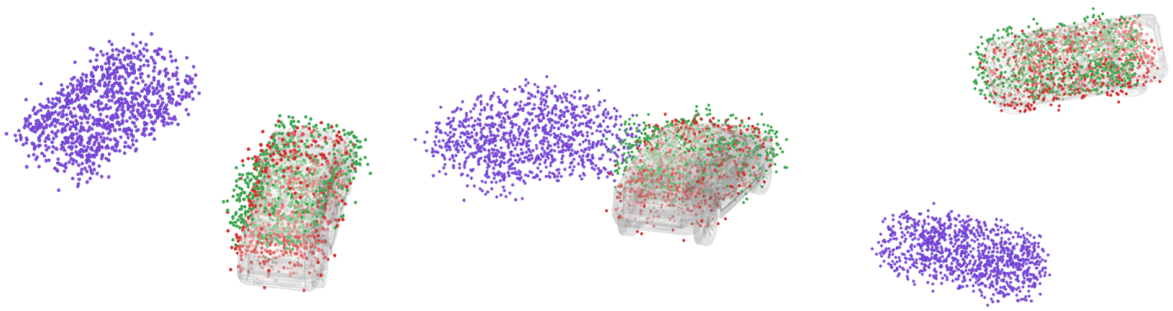
In the first experiment, iterative PCRNet and PointNetLK are trained on 20 different object categories from *ModelNet40* with total of 5070 models. We perform tests using 100 models chosen from 5 object categories which are not in training data (referred as unseen categories) with no noise in point clouds. We ensure that same pair of source and template point clouds are used to test all algorithms, for a fair comparison.

[1]We define success ratio as the number of test cases having rotation error less than success criteria.

We trained iterative PCRNet and PointNetLK using multiple object categories and tested them using object categories which are not in training data. There was no noise in source data during training and testing for this experiment. With these tests, we found that AUC for ICP is 0.802, for our iterative PCRNet is 0.682 and for PointNetLK it is 0.998.

Upon repeating the experiments by training the networks with objects from the same category as the data being tested on, we observe a massive improvement in the AUC for iterative PCRNet, going from 0.682 to 0.972. The AUC for ICP and PointNetLK were similar to earlier at 0.862 and 0.998 respectively, and the AUC of PCRNet was 0.998.

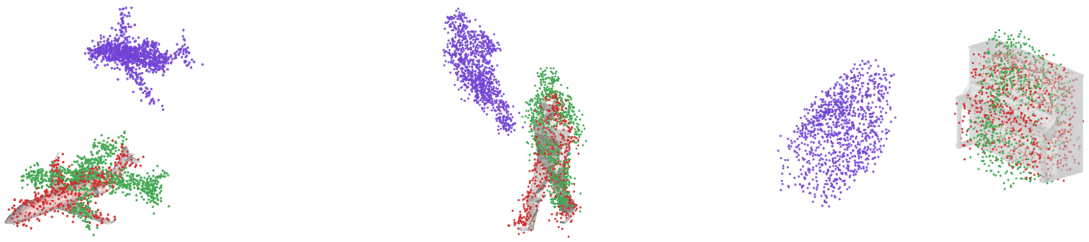
These results emphasize that the iterative PCRNet and PCRNet, when retrained with object specific information, provide improved registration results compared to ICP as well as the version trained with multiple categories. Their performance is comparable to PointNetLK when trained with object specific information. However, PointNetLK shows better generalization than iterative PCRNet across various object categories and has better performance compared to ICP (as also observed by [2]). We attribute this



(a) Trained on one car
Iterative PCRNet: Rot error = 2.14°, Trans error = 0.0056 units.

(b) Trained on multiple cars
Iterative PCRNet: Rot error = 2.14°, Trans error = 0.0056 units.

(c) Trained on multiple categories
Iterative PCRNet: Rot error = 3.07°, Trans error = 0.0107 units.



(d) Trained on multiple categories
Iterative PCRNet: Rot error = 0.34°, Trans error = 0.0048 units.
ICP: Rot error = 43.62°, Trans error = 0.2564 units.

(e) Trained on multiple categories
Iterative PCRNet: Rot error = 5.55°, Trans error = 0.0042 units.
ICP: Rot error = 45.15°, Trans error = 0.1767 units.

(f) Trained on multiple categories
Iterative PCRNet: Rot error = 5.96°, Trans error = 0.0035 units.
ICP: Rot error = 75.02°, Trans error = 0.0683 units.

Figure 5: Qualitative results for Section 4.2. For each example, template is shown by a grey rendered CAD model, purple points show initial position of source and red points show converged results of iterative PCRNet trained on data with noise and green points show results of ICP. (a), (b), (c), (d) and (e) show the results for objects from seen categories, while (f) shows results of unseen category.

to the inherent limitation of the learning capacity of PCRNet to large shape variations, while PointNetLK only has to learn the PointNet representation rather than the task of alignment. However, in the next set of experiments, we demonstrate the definite advantages of PCRNet over PointNetLK and other baselines, especially in the presence of noisy data.

4.2. Gaussian noise

In order to evaluate robustness of our networks to noise, we perform experiments with Gaussian noise in the source points. For our first test, we use dataset as described in Sec. 4.1. We sample noise from Gaussian distribution for each point in source point cloud with 0 mean and a standard deviation varying in the range of 0 to 0.04 units. For these results, we trained an iterative PCRNet and a PointNetLK which trained with noisy source point clouds using 20 different object categories and a total of 5070 models.

During testing, we compare ICP, PointNetLK and iterative PCRNet with noise in source data for each algorithm. We ensured that the dataset has the same pairs of source and template point clouds for a fair comparison. Fig. 4a shows the result. We observe that our iterative PCRNet has higher

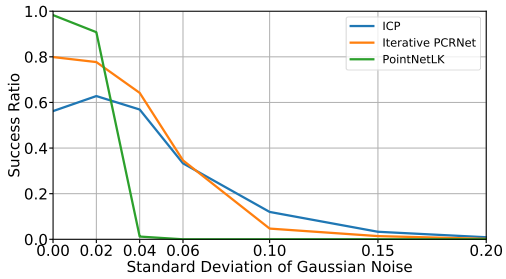


Figure 6: Results for Sec. 4.2. Iterative PCRNet and PointNetLK are trained on multiple object categories with Gaussian noise, having maximum value of std. dev. equal to 0.04. The x-axis shows different values of standard deviation in noise used in testing. PointNetLK is most accurate in the absence of noise, while iterative PCRNet is robust to noise around the levels that it has observed during training (0.02-0.06).

Table 1: Results from Section 4.3. Accuracy and computation time comparisons for registering noisy data. Notice that both PCRNet models achieve nearly the same AUC as Go-ICP while being orders of magnitude faster.

	Rot. Error (deg)		Trans. Error		Time (ms)		AUC
Algorithm	Mean	Std. Dev.	Mean	Std. Dev.	Mean	Std. Dev.	
PCRNet	8.82	4.82	0.0077	0.0008	1.89	0.398	0.9544
Iterative PCRNet	1.03	2.56	0.0085	0.0024	146	30.40	0.9943
PointNetLK [2]	51.80	29.63	0.8783	0.0054	234	41.60	0.7059
ICP [6]	11.87	31.87	0.0282	0.0392	407	128.00	0.9321
Go-ICP [37]	0.45	0.19	0.0016	0.0007	2.7×10^5	1.5×10^5	1.0000

number of successful test cases with smaller rotation error as compared to ICP and PointNetLK, which shows that our iterative PCRNet is robust to Gaussian noise. It is worth noting that PointNetLK performs the worst and is very sensitive to noisy data. The above test results emphasize that iterative PCRNet works quite well in the presence of noise in the source data, with performance beating all other methods if the object category is known.

For the second test, we used the dataset as described in Sec. 4.1 and added Gaussian noise in source point clouds as described above. We train the networks on a specific object category and test them on the same category using 150 models of cars. Gaussian noise was present during training and testing in source point clouds. The result in Fig. 4b shows that iterative PCRNet performs the best and has higher number of successful test cases.

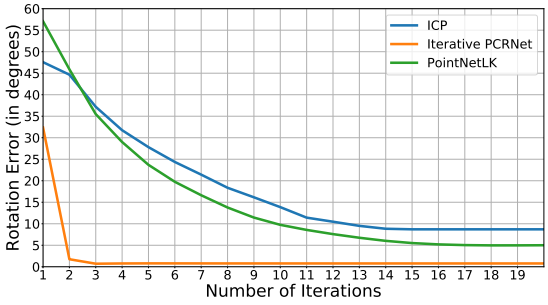


Figure 7: The y-axis is rotation error between the predicted and ground truth transformation, and x-axis shows the number of iterations performed to find the transformation. Iterative PCRNet shows the ability to align source and template point clouds in fewer iterations.

We compare the success ratio of networks when training and testing on only one noisy model (see Fig. 4c). Iterative PCRNet once again exhibits a high success ratio, which is better than ICP and PointNetLK. Finally, we compare PCRNet that is trained without noise and tested on noisy data, with ICP and PointNetLK. While not being as good as ICP, our result is still competitive, and performs much better than PointNetLK (See Fig. 4d).

We present qualitative results in Fig. 5 using iterative PCRNet trained on multiple datasets and testing with noisy data. As expected, the accuracy of iterative PCRNet is highest when trained on the same model that it is being tested on. However, the accuracy drops only a little when trained on multiple models and multiple categories, showing a good generalization as long as there is some representation of the test data in the training. Further the results are accurate also for some unseen categories as shown in Fig. 5(c,f), which shows the generalizability of iterative PCRNet.

Fig. 6 shows success ratio versus change in the amount of noise added to source point clouds during testing. Both iterative PCRNet and PointNetLK are trained on multiple object categories with Gaussian noise having a maximum standard deviation of 0.04. We observe a sudden drop in the PointNetLK performance as the standard deviation for noise increases above 0.02. On the other hand, iterative PCRNet performs best in the neighbourhood of the noise range that it was trained on (0.02-0.06), and produces results comparable to ICP beyond that noise level. This shows that our network is more robust to noise as compared to PointNetLK.

Fig. 7 shows the rotation error versus number of iterations in for the different methods. Notice that the iterative PCRNet takes only 3 iterations to get close to convergence, compared to the other methods that take upwards of 15 iterations.

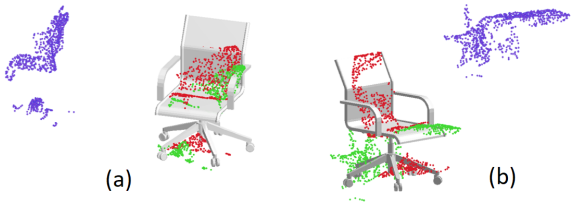


Figure 8: Registration of chair point cloud taken from Stanford *S3DIS* indoor dataset [3]. CAD model shows the template data from *ModelNet40*, purple points is from *S3DIS* dataset, red points represent iterative PCRNet estimates, while the green ones represent ICP estimates.

Figure 9: Qualitative results for Section 5. Replacement of chairs in office scene from Stanford *S3DIS* indoor dataset[3]. Red leather chairs shows the replaced chair from *ModelNet40* [35] (a) Original scene. Red leather chair replaced by using registration from (b) ICP [7], (c) mixed integer programming [19], and (d) iterative PCRNet.

4.3. Computation speed comparisons

In this experiment, we use a testing dataset with only one model of car from *ModelNet40* dataset, with Gaussian noise in the source data. We apply 100 randomly chosen transformations with Euler angles in range of $[-45°, 45°]$ and translation values in range of [-1, 1] units. The networks are all trained using multiple models of same category (i.e. car). We compared the performance of iterative PCRNet, PCRNet, PointNetLK, ICP and Go-ICP, as shown in Table 1. We report the rotation and translation error after registration, computation time, and the AUC.

The results demonstrate that Go-ICP converges to a globally optimal solution with a very small rotation error and translation error, but the time taken is three orders of magnitude more than iterative PCRNet and five orders of magnitude more than PCRNet. The AUC value of Go-ICP is 1, meaning that it has converged in all test cases, while our network has the second best AUC value. This experiment shows how the iterative PCRNet is similar to Go-ICP in terms of accuracy, but computationally much faster, allowing for use in many practical applications. Further, while the PCRNet is less accurate than iterative PCRNet and Go-ICP, the accuracy may be good enough as a pre-aligning step in applications such as object detection and segmentation [39].

5. Model replacement using segmentation

To show qualitative performance on real-world data, we demonstrate the use of our approach to find the pose and modify the models in an indoor point cloud dataset [3]. we perform model replacement in use the semantic segmentation network introduced in PointNet [28] to predict labels for each object in the Stanford *S3DIS* indoor dataset [3]. Point cloud corresponding to a chair are selected from the scene and registered to a chair model from *ModelNet40* dataset using iterative PCRNet, which was trained on multiple object categories with noise (see Fig. 8).

The transformation predicted by iterative PCRNet is then applied to the model chosen from *ModelNet40* and replaced at the place of the original chair as shown in Fig. 9. The original scene is shown in Fig. 9(a). The blue chair is replaced with a red chair of a different model from *ModelNet40* dataset. Fig. 9(b) shows the result from using ICP. Notice how ICP fails to register the chair to the right pose. While we observed in Sec. 4.3 that Go-ICP produces the most accurate results, in this example Go-ICP did not improve upon the result of ICP. Thus we report the result of another global registration method that uses mixed integer programming (MIP) [19] (see Fig. 9(c)). Note that neither ICP nor MIP produce results that are as accurate as that produced by the iterative PCRNet in Fig. 9(d). This is because ICP, Go-ICP and MIP require the template and the source to be of the same object and any variation between objects of the same category can result in poor registration. Our approach however, is robust to changes in shapes within the same category and produces a better result.

6. Discussions and future work

This work presents a novel data-driven framework for performing registration of point clouds using the PointNet representation. The framework provides an approach to use data specific information for producing highly accurate registration that is robust to noise and initial misalignment, while being computationally faster than existing methods. The framework can be implemented in an iterative manner to obtain highly accurate estimates comparable to global registration methods. The framework could also be implemented without the iterations, but with deeper layers to produce two to five orders of magnitude speed improvement compared to popular registration methods. The framework illustrates how data-driven techniques may be used to learn a distribution over appearance variation in point cloud data, including noisy data or category-specificity, and perform better at test time using such a learned prior. Finally, this framework also puts into context other recent PointNet-based registration methods in literature such as the PointNetLK.

Future work would involve modifying the network to handle partial and occluded point clouds, as well as inte-

gration into larger deep neural network systems, for tasks such as multi-object tracking, style transfer, mapping, etc. Future work may explore the limitations of the learning capacity of the fully-connected registration layers to the size of data distribution.

References

[1] M. Angelina Uy and G. Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4470–4479, 2018. 2

[2] Y. Aoki, H. Goforth, R. A. Srivatsan, and S. Lucey. PointNetLK: Robust & Efficient Point Cloud Registration using PointNet. *arXiv preprint arXiv:1903.05711*, 2019. 1, 2, 3, 4, 5, 7

[3] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 7, 8

[4] R. Arun Srivatsan, M. Xu, N. Zevallos, and H. Choset. Probabilistic pose estimation using a Bingham distribution-based linear filter. *The International Journal of Robotics Research*, 37(13-14):1610–1631, 2018. 2

[5] S. Baker and I. Matthews. Lucas-Kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004. 2, 3

[6] P. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb 1992. 7

[7] P. J. Besl and N. D. McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992. 1, 2, 8

[8] H. Bristow, J. Valmadre, and S. Lucey. Dense semantic correspondence where every pixel is a classifier. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4024–4031, 2015. 2

[9] B. Eckart, K. Kim, and J. Kautz. Fast and accurate point cloud registration using trees of gaussian mixtures. *arXiv preprint arXiv:1807.02587*, 2018. 2

[10] G. Elbaz, T. Avraham, and A. 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. 2

[11] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 1997. 4

[12] H. Fan, H. Su, and L. J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[13] N. Gelfand, N. J. Mitra, L. J. Guibas, and H. Pottmann. Robust global registration. In *Symposium on geometry processing*, volume 2, page 5, 2005. 1, 2

[14] G. Georgakis, S. Karanam, Z. Wu, and J. Kosecka. Matching RGB Images to CAD Models for Object Pose Estimation. *arXiv preprint arXiv:1811.07249*, 2018. 2

[15] J. Glover, G. Bradski, and R. B. Rusu. Monte carlo pose estimation with quaternion kernels and the distribution. In *Robotics: Science and Systems*, volume 7, page 97, 2012. 2

[16] Y. Guo, M. Bennamoun, F. Sohel, M. Lu, and J. Wan. 3D object recognition in cluttered scenes with local surface features: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2270–2287, 2014. 2

[17] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016. 3

[18] M. B. Horowitz, N. Matni, and J. W. Burdick. Convex relaxations of SE(2) and SE(3) for visual pose estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1148–1154. IEEE, 2014. 2

[19] G. Izatt, H. Dai, and R. Tedrake. Globally Optimal Object Pose Estimation in Point Clouds with Mixed-Integer Programming. In *International Symposium on Robotics Research*, 12 2017. 2, 8

[20] T. Jost and H. Hugli. A multi-resolution scheme ICP algorithm for fast shape registration. In *Proceedings. First International Symposium on 3D Data Processing Visualization and Transmission*, pages 540–543. IEEE, 2002. 2

[21] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. *Proceedings of 7th IJCAI*, 1981. 2, 3

[22] Z. Ma, B. Liu, F. Zhou, and J. Chen. Point reg net: Invariant features for point cloud registration using in image-guided radiation therapy. *Journal of Computer and Communications*, 06:116–125, 01 2018. 1

[23] A. Makadia, A. Patterson, and K. Daniilidis. Fully automatic registration of 3D point clouds. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 1297–1304. IEEE, 2006. 2

[24] H. Maron, N. Dym, I. Kezurer, S. Kovalsky, and Y. Lipman. Point registration via efficient convex relaxation. *ACM Transactions on Graphics (TOG)*, 35(4):73, 2016. 2

[25] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015. 1, 2

[26] M. Ovsjanikov, Q. Mérigot, F. Mémoli, and L. Guibas. One point isometric matching with the heat kernel. In *Computer Graphics Forum*, volume 29, pages 1555–1564. Wiley Online Library, 2010. 2

[27] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. Frustum pointnets for 3d object detection from RGB-D data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018. 1, 2

[28] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017. 1, 2, 8

[29] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In

Advances in Neural Information Processing Systems, pages 5099–5108, 2017. 1, 2

[30] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3dim*, volume 1, pages 145–152, 2001. 2

[31] R. B. Rusu, N. Blodow, and M. Beetz. Fast point feature histograms (FPFH) for 3D registration. In *IEEE International Conference on Robotics and Automation*, pages 3212–3217. IEEE, 2009. 1, 2

[32] J. Vongkulbhisal, F. De la Torre, and J. P. Costeira. Discriminative optimization: Theory and applications to point cloud registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4112, 2017. 2

[33] J. Vongkulbhisal, B. Irastorza Ugalde, F. De la Torre, and J. P. Costeira. Inverse composition discriminative optimization for point cloud registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2993–3001, 2018. 2

[34] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon. Dynamic graph CNN for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018. 1, 2

[35] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. 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. 1, 2, 4, 8

[36] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. PoseCNN: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017. 2, 4

[37] J. Yang, H. Li, D. Campbell, and Y. Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE transactions on pattern analysis and machine intelligence*, 38(11):2241–2254, 2016. 1, 2, 7

[38] Z. J. Yew and G. H. Lee. 3dfeat-net: Weakly supervised local 3d features for point cloud registration. In *European Conference on Computer Vision*, pages 630–646. Springer, 2018. 2

[39] W. Yuan, D. Held, C. Mertz, and M. Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018. 1, 2, 8

[40] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert. PCN: Point Completion Network. In *3D Vision (3DV), 2018 International Conference on*, 2018. 2

[41] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4490–4499, 2018. 2