Exploring Deep Closest Point: Learning Representations for Point Cloud Registration

Kaan Oguzhan
Technische Universität München
kaan.oguzhan@tum.de

Baris Sen
Technische Universität München
baris.sen@tum.de

Yigit Aras Tunali
Technische Universität München
yigitaras.tunali@tum.de

Kerem Yildirir
Technische Universität München
kerem.yildirir@tum.de

Abstract

Point cloud registration is defined as the task of finding a rigid transformation that aligns two-point cloud sets, such that the resulting point cloud is globally consistent. In recent years, following the increasing availability of 3D scanners and the recent advancements in deep learning in computer vision, the point cloud registration task has gained significant attention. In this project, we investigate the method Deep Closest Point (DCP) [11] in detail. DCP proposes a learning approach to the point cloud registration problem. We evaluate the robustness of the model and propose an extension for improving the model's performance in real-world use cases. We show that DCP suffers when there is no one-to-one matching between the source and target points. Furthermore, we show that using color as an additional input to the model significantly improves the performance of the network, while reducing the required train time. We create a new 3D point cloud dataset from Mixamo [1] character animations and use it for training and testing the model, and compare the models trained with Mixamo on the Mixamo and TUM RGBD datasets [9].

1. Introduction

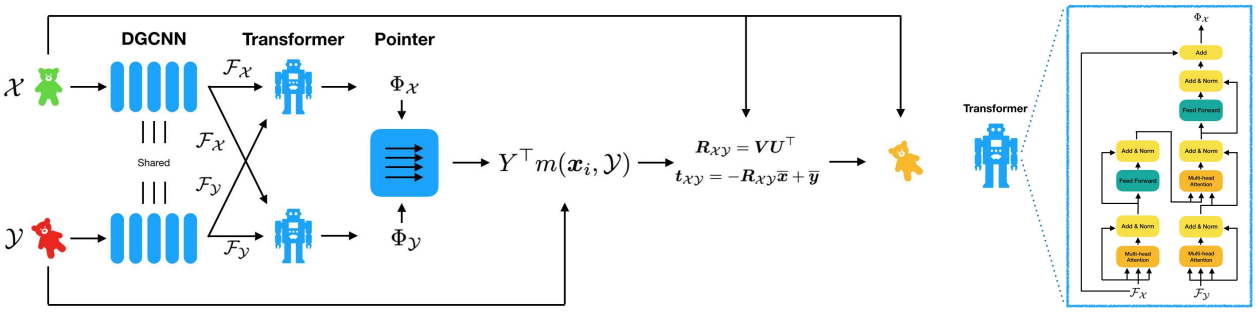
Point cloud registration is one of the fundamental tasks in computer vision. Given two point clouds, we are looking for the rotation and translation that transform one point cloud such that two point clouds together are aligned. One of the most common algorithms used in this task is the Iterative Closest Point (ICP) [3] algorithm. ICP iteratively optimizes for two subtasks of the problem: finding the correspondents in two point clouds and finding the best transformation that makes the resulting point cloud globally consistent. Like many alternating optimization methods, ICP may suffer from bad initialization and converge to a local minimum. Many variants of ICP have been implemented that improve the speed and accuracy of the registration [6].

Deep Closest Point (DCP) is one of the first deep learning approaches on the point cloud registration task. The DCP Network consists of three differentiable modules and therefore can be trained end-to-end. The first module of the network is a point cloud embedding network, that takes the input point cloud and using a Graph Convolutional Neural Network (GCNN) [12] generates embeddings for each point. The second module is a transformer [10] module, that entirely relies on an attention mechanism [2] to find global point correspondences between two point clouds. The final module is a singular value decomposition (SVD) module, that uses the global correspondences outputted by the transformer and calculates the best rigid transformation to align the point clouds. Figure 1 shows the detailed architecture of DCP.

In this project, we investigate DCP in detail, identify possible improvements, and evaluate the identified improvements on a new dataset "Mixamo", which is created using a subset of Mixamo character animations, and the TUM-RGBD dataset [9] for objective evaluation of both models on an unseen data. Specifically, we investigate the following 4 points:

- The performance of DCP when the source and target point clouds do not have a 1-to-1 mapping and ideas to improve its performance
- Color as an additional input on the model performance
- As-Rigid-As-Possible (ARAP) [8] regularization as additional loss signal

In the following, we summarize related work, our method, the experiments we performed, and final conclusions.



(a) Network architecture (b) Transformer module

Figure 1: Detailed Architecture of DCP. DCP receives the source and the target point clouds as input to the point cloud embedding module. The point cloud embedding module is the Dynamic Graph Convolutional Neural Network (DGCNN) [12]. The outputs of DGCNN are fed into the Transformer that helps creating an alignment between the source and target point clouds using attention. Finally, a differentiable SVD module calculates the predicted transformation using the point alignments.

2. Related Work

There are many deep learning approaches to solve the point cloud registration task. Most of the works pursue a similar approach to DCP, where they first find correspondences, and then estimate the target transformation. PRNet [13] performs these steps iteratively to improve performance. It also chooses a subset of points that has correspondences in both point clouds to improve transformation estimation. SuperGlue [7] uses graph attention to find corresponding point pairs. Deep Global Registration [4] identifies that DCP performs poorly on real-world data and excludes the points classified as outliers from transformation estimation. RPM-Net [13] uses soft point assignments and a weighted SVD to predict the transformation. R-PointHop [5] focuses on learning better point features and improve results by using a more complex point feature extraction module. TEASER [14] estimates the translation and rotation matrices separately so that the network can learn different weights for rotation and translation.

3. Method

Our work has two main focus areas. First, we create a new 3D colored point cloud dataset from Mixamo character animations. Second, we implement several improvements to the model and evaluate their effectiveness.

3.1. Datasets

We use 2 different datasets for our experiments. We generate our own 3D point cloud dataset using Mixamo character animations. Moreover, we use the TUM RGBD dataset to test the model on unseen data.

3.1.1 Mixamo

Mixamo is a database of 3D characters with animations of different lengths. Using the database, we have sampled out 10 characters with 10 random animations each. For the final dataset, using the animations of the characters, we have extracted different frames from each animation, resulting in 1031 different colored point clouds.

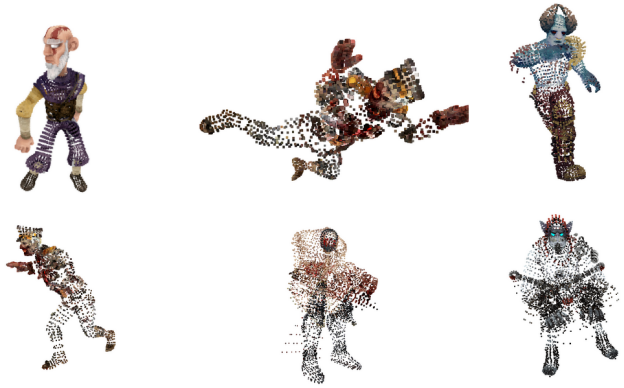


Figure 2: A sample subset of our dataset

For training set and validation set instead of generating the data in a pre-process step and using splits, we followed a different approach. In the two cases, our data loader loads a random point cloud out of the 1031 samples, then applies a random rotation & translation to the point cloud.

The validation set have a random generator with fixed internal seeds, that is independent from the global seed. This approach guarantees that the same point cloud is loaded and

the same rotation and translation is generated for each element index, while also allowing us the change the validation set size dynamically.

For the training set, we use models that aren't used for the training which the model hasn't seen yet.

3.1.2 TUM RGBD

This dataset is only used for the analysis of the trained models and therefore are never seen by the models during any of our trainings. Colored point clouds are taken from different frames of the sequences and used for the analysis.

3.2. Extensions to DCP

As mentioned in Section 1, we have investigated the DCP network in detail. We have concluded that the network can successfully learn to find the rigid motion, that is applied on the source point-cloud. During the training, the network receives two-point clouds, which are transformed from source to target state by a randomly generated rigid motion.

This training method, per se, was not a bad choice, although it comes with a main shortcoming: In real-world applications, the model needs to be able to handle scenarios where there are no perfect correspondences, such as occlusions in the scene or imperfections due to sensory noise. The authors did not provide any robustness evaluation against these types of scenarios and assumed the source and target states are perfect.

In our investigation, we have trained the network on the Mixamo [1] dataset, with and without independently sampled source-target point clouds pairs. Then evaluated all networks on 2 completely different datasets: Mixamo [1] and TUM RGBD [9].

As the commodity 3D sensors become more commercially available, it is easier to get 3D point cloud data which also has color information for each point. We identify color as a strong signal for finding point correspondences and evaluate the effect of this additional signal on the registration task.

3.2.1 Independent Sampling

In the original model, training was done on point clouds by randomly rotating them and using this random rotation between the original and rotated point clouds as the ground truth. Even though this approach works relatively well, it has its weaknesses. It is not robust against occlusions, noise, or partially matching scans which are mostly found in real-life problems. This weakness was more evident when we tried to align two frames from the TUM-RGBD dataset which were more apart from each other. To this problem, we tried the approach of training the model with a independent sampling method. The main idea behind this approach is to first rotate the point cloud but sample both the original and the rotated point clouds at random but separately. Thus we create examples that do not have perfect 1-to-1 correspondences.

3.2.2 Adding color input

In the DCP architecture, the coordinates of points are fed into the DGCNN network as a 3 channel input. Additional to the 3 channel coordinate input, we feed the RGB color of the points as 3 additional channels to DGCNN. The input and output sizes of the layers other than the first layer of DGCNN remains same.

3.2.3 ARAP Regularizer

Original DCP architecture used the DGCNN and Transformer to learn the local features of the mesh points, but in the loss function, there was not any term for protecting local rigidity. This is expected, since we are predicting the correspondences and apply a rigid transformation accordingly. However we wanted to achieve better correspondences by penalizing the wrong correspondences more by adding another loss term. The original ARAP regularizer proposed in [8] enforces local rigidity when computing a (not necessarily rigid) deformation, and it is computed as follows:

$$E\left(\mathcal{C}_i, \mathcal{C}_i'\right) = \sum_{j \in \mathcal{N}(i)} w_{ij} \left\| \left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{R}_i \left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|^2 \tag{1}$$

where $w_i j$ is the weight of the point pair, p_i and p_i' correspond to the same point before and after transformation respectively, p_j and p_j' correspond to the neighbours of p_i and p_i', and R_i is the rotation aligning the local patches around p_i and p_i'. In our case, we are computing a rigid transformation, so local rigidity is preserved and we do not need a R_i for every pair. We also omit the weighting for simplicity. Our loss term is like the following:

$$E\left(\mathcal{C}_i, \mathcal{C}_i'\right) = \sum_{j \in \mathcal{N}(i)} \left\| \left(\mathbf{p}_i' - \mathbf{p}_j'\right) - \mathbf{R} \left(\mathbf{p}_i - \mathbf{p}_j\right) \right\|^2 \tag{2}$$

4. Experiments

All dataset results are generated by models trained on Mixamo dataset, then evaluated on the dataset mentioned in the upper left corner of the corresponding table.

For each comparison we use Mean Square Error (MSE), Root Mean Square Error (RMSE), and Mean Absolute Error (MAE) on both the rotation matrix ($\boldsymbol{R}$) and the translation matrix ($\boldsymbol{t}$).

4.1. Color input

We test the effect of the additional color input on the performance of DCP. For this, we train the baseline architecture with and without the color input. Network trained with additional color input significantly outperforms the model without the color input. Table 1 and 2 show the performance of the networks on the TUM RGBD and Mixamo datasets, respectively.

TUM RGBD	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP	387.982239	19.697266	11.285736	0.029045	0.170427	0.113288
DCP + Color	**0.240215**	**0.490117**	**0.334967**	**0.000094**	**0.009678**	**0.007570**

Table 1: DCP color improvement comparison

Mixamo	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP	51.460548	7.173601	4.399128	0.043476	0.208509	0.075243
DCP + Color	**0.171919**	**0.414631**	**0.106271**	**0.000035**	**0.005910**	**0.001755**

Table 2: DCP color improvement comparison

4.2. ARAP Regularizer

To evaluate whether the additional ARAP regularization loss helps to generalize better, we perform several runs of training with and without ARAP regularization. Table 3 shows the performance of DCP with and without ARAP regularizations. Since the additional color input showed significant improvement, we also evaluated the effect of ARAP regularization on the improved model. Table 4 and 5 show the effect of ARAP regularization on the DCP model with additional color input. Results show that ARAP regularization slightly improves the performance in most of these settings.

Mixamo	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP	51.460548	7.173601	4.399128	**0.043476**	**0.208509**	**0.075243**
DCP + Arap	**48.080051**	**6.933978**	**4.212567**	0.050855	0.225510	0.076915

Table 3: DCP without color using ARAP regularization

TUM RGBD	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP + Color	0.240215	0.490117	0.334967	0.000094	0.009678	0.007570
DCP + Color + ARAP	**0.169369**	**0.411545**	**0.270199**	**0.000075**	**0.008643**	**0.007129**

Table 4: DCP with color using ARAP regularization

Mixamo	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP + Color	0.171919	**0.414631**,	0.106271	**0.000035**	**0.004833**	0.001755
DCP + Color + ARAP	**0.171727**	0.451673	**0.083366**	0.000044	0.006611	**0.001589**

Table 5: DCP with color using ARAP regularization

4.3. Independent Sampling

We evaluate the effect of sampling source and target points independently during training. We compare the baseline model with and without independent sampling on the test data, meaning that source and target point clouds are sampled independently. As expected, we see that model with independent sampling outperforms the baseline in almost all use cases. Table 6 and 7 show the results of the comparison.

Mixamo	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP	53.342667	7.303607	4.438653	0.048586	0.220422	0.077097
DCP + Independent Sampling	**44.15807**	**6.645154**	**4.342775**	**0.030093**	**0.173474**	**0.067707**

Table 6: DCP with independent sampling comparison

TUM RGBD	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
DCP	390.456390	19.759970	11.853582	0.030235	0.173882	**0.119560**
DCP + Independent Sampling	**256.066223**	**16.002069**	**10.963472**	**0.027706**	**0.166452**	0.120926

Table 7: DCP with independent sampling comparison

4.4. Final Comparison

Finally, we compare our best model with the original DCP trained on ModelNet40 dataset. Our best model uses additional color input and ARAP regularization, is trained on our Mixamo dataset. Table 8 shows that our best model significantly outperforms the original model.

TUM RGBD	MSE($\boldsymbol{R}$)	RMSE($\boldsymbol{R}$)	MAE($\boldsymbol{R}$)	MSE($\boldsymbol{t}$)	RMSE($\boldsymbol{t}$)	MAE($\boldsymbol{t}$)
Original DCP	1.573519	1.254400	0.890428	0.000277	0.016652	0.01259
DCP + Color + ARAP	**0.171727**	**0.451673**	**0.083366**	**0.000044**	**0.006611**	**0.001589**

Table 8: Original DCP compared with our best result

5. Conclusion

In this project, we implemented several extensions to the DCP network and showed their effectiveness. Our experiments show that color is a strong signal for point matching and significantly improves the final performance. Another finding we reached is that the originally trained network does not work well with source and target point clouds which do not correspond one-to-one but rather have overlapping points and different points. To perform better in scenarios like this, we train DCP with independently sampled source and target point clouds. We also believe this training will improve the network performance in real-world scenarios, such as aligning point clouds in a frame-to-frame tracking scenario. We also used ARAP regularization as an additional loss in the training and showed its effectiveness on registration performance.

References

[1] Adobe. Mixamo model dataset, 2018. 1, 3

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 1

[3] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data*

structures, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992. 1

[4] Christopher Choy, Wei Dong, and Vladlen Koltun. Deep global registration, 2020. 2

[5] Pranav Kadam, Min Zhang, Shan Liu, and C. C. Jay Kuo. R-pointhop: A green, accurate and unsupervised point cloud registration method, 2021. 2

[6] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001. 1

[7] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks, 2020. 2

[8] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*, SGP '07, page 109–116, Goslar, DEU, 2007. Eurographics Association. 1, 3

[9] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012. 1, 3

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 1

[11] Yue Wang and Justin M. Solomon. Deep closest point: Learning representations for point cloud registration. *CoRR*, abs/1905.03304, 2019. 1

[12] 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. 1, 2

[13] Zihao Yan, Ruizhen Hu, Xingguang Yan, Luanmin Chen, Oliver Van Kaick, Hao Zhang, and Hui Huang. Rpm-net. *ACM Transactions on Graphics*, 38(6):1–15, Nov 2019. 2

[14] Heng Yang, Jingnan Shi, and Luca Carlone. Teaser: Fast and certifiable point cloud registration, 2020. 2