DCPCR: Deep Compressed Point Cloud Registration in Large-Scale Outdoor Environments

Ghailen Ben Achour

January 2023

Contents

List of Figures

List of Tables

1 Introduction

Pointcloud registration is a common technique that aims to match multiple pointclouds captured at different time steps in order to perform map matching. However, pointcloud registration mainly causes two problems:

- A downgrade in term of matching performance especially when
  1. Pointcloud is noisy (presence of noisy objects in the scene, Sensor with high precision error ...)
  2. Huge gap between the initial guessed position of the pointclouds
- A huge computation cost due to the high number of data points per scan (thousands of points per scan).

1.1 Background

For pointcloud registration, two main methods are used:

1.1.1 Local Registration

The most common approach for aligning two point clouds is ICP [1] with its variants (Point-to-Point, Point-to-Plane...) (Figure 1). Here, the main challenge is to find the correct correspondences. Looking only at spatially close points often fails when the initial guess is too far from the correct transformation. Moreover, outdoor environments often change (moving objects, ...), and therefore assuming to have a lot of one-to-one correspondences does not necessarily hold.

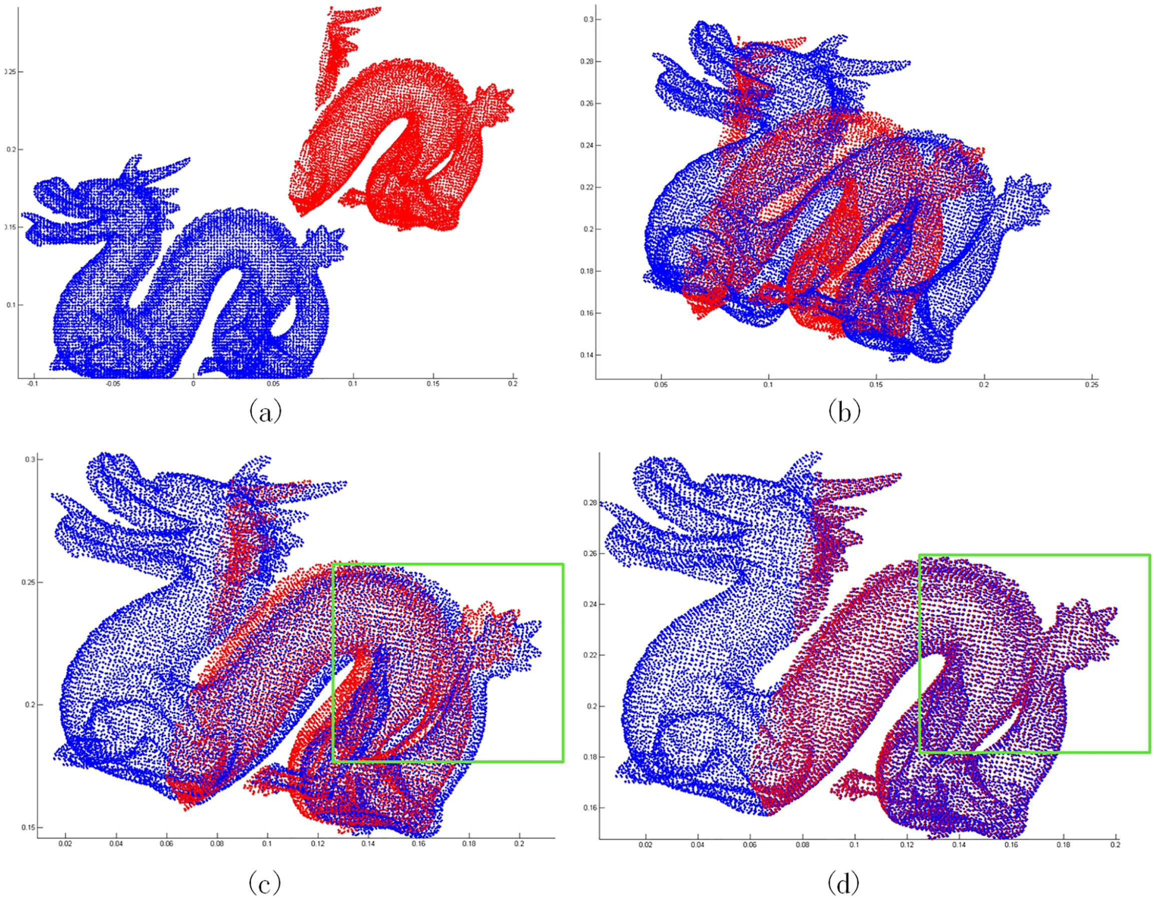

Figure 1: Registration results on a point cloud for different ICP variants

1.1.2 Global Registration

Global registration methods try to estimate the pose between the point clouds even when the initial guess is quite far away from the correct transformation. RANSAC (RANdom SAmple Consensus) [2] is one of these methods that provides the opportunity to deal with large transformations as well as outliers by sampling correspondences combined with a large number of repetitions. The biggest disadvantage of those methods is typically the high computation time.

2 DCPCR

2.1 Feature-Based Point Cloud Registration

For the alignment of two point clouds, DCPCR [3] follows the classical paradigm of first finding point correspondences, which are then used to estimate the relative transformation between a source and a target point cloud. While in the classical ICP, correspondences are determined via geometric neighborhood, DCPCR follows a feature-based approach. The transformation consisting of rotation and translation is estimated using a neural network described in figure 2.

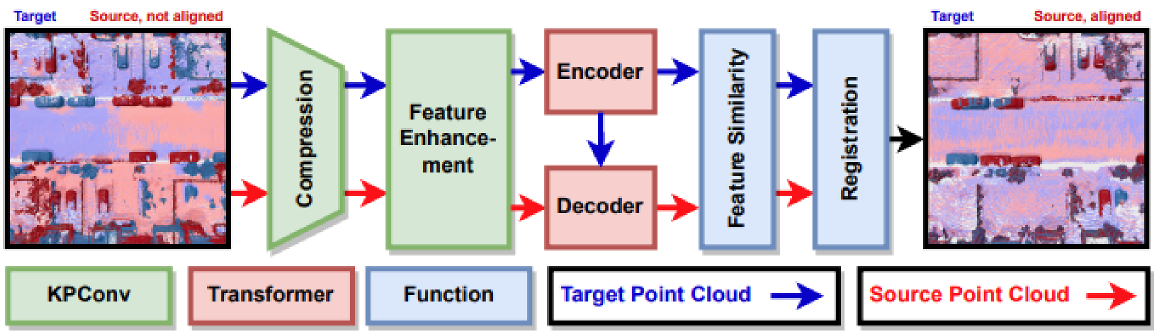

Figure 2: Network architecture

This network architecture consists of three parts, namely a compression encoder for memory and compute efficiency, a convolutional block to increase the receptive field, and a transformer head for global feature aggregation. The network can work on raw or compressed pointcloud. In DCPCR work, the compression of the pointcloud was done using a neural network [4] which substantially reduces the number of points and provides locally-aware features.

1. **Compression Network** PointNet [5] is used to transform the compressed point representation to a localization-specific feature space, which is better suited for matching than for reconstruction.

2. **Feature Enhancement Network** Since the features from the compression network contain only local information, KPConv , which is a sparse convolution, was used to aggregate the information from the points within a radius r.

3. **Transformer** Transformers are used for the multihead self-attention mechanism for global feature aggregation.

2.2 Results

We train DCPCR network on Apollo Dataset for 30 epochs on RTX 3060 GPU. We test the performance of the neural network on the test set and we compare it a to a pretrained model trained for 1000 epochs.

Test metric (mean)	dr_degrees	dt_meters
Our model	2.2	0.38
Pretrained model	**0.36**	**0.12**

Table 1: Metric comparison: **dr_degrees** is the angle between ground truth and prediction in degrees. **dt_meters** is the euclidean distance between the ground truth and the estimated translation vectors.

Since we would like to test the model's performance on newly seen dataset, we also report inference results on our local data. We split inference into 3 main categories.

1. **Easy tests**: During which we test our model's performance on the most non complex cases. One of these tests is introducing the same source and target input pointcloud to the model. We do not apply any rotation, translation or noise. For these experiments, the models performs perfectly by predicting the identity matrix as the estimated pose. The figure 3 below shows how the predicted position (blue) and the target/ground truth position (in green) are perfectly aligned.

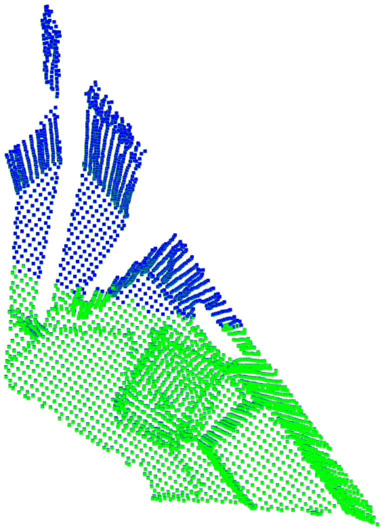

Figure 3: Registration: Identity matrix

2. **Medium tests** During which we test our model's performance on slightly challenging cases. We introduce some modifications to the source pointcloud including rotation, translation, random noise, interpolation of points, subsampling, ... We than try to match it with the unmodified target pointcloud. For these kind of experiments, we notice that, for small errors: translation offset $\simeq$[-1, 1], rotation $\simeq$[-35°, 35°], and few hundreds of interpolated or jittered points, the performance of the model is still high and precise enough. The figure 4 below shows how for slightly modified source pointcloud (green), the predicted position (blue) and the target/ground truth position (in red) are well aligned.

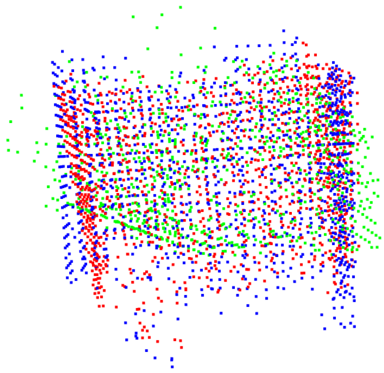

Figure 4: Registration: Low offset between source and target

3. **Hard tests**: During which we test our model's performance on challenging cases. We introduce many modifications to the source pointcloud at the same time including rotation, translation, random noise, interpolation of points, subsampling, ... We than try to match it with the unmodified target pointcloud. For these kind of experiments, we notice that, for big errors: translation offset $\|dt\| \geq 1, rotation\|rt\| \geq 35°$, and few hundreds of interpolated or jittered points, the performance of the model is very low. The figure 5 below shows how for highly modified source pointcloud (red), the predicted position (blue) and the target/ground truth position (green) are not well aligned.

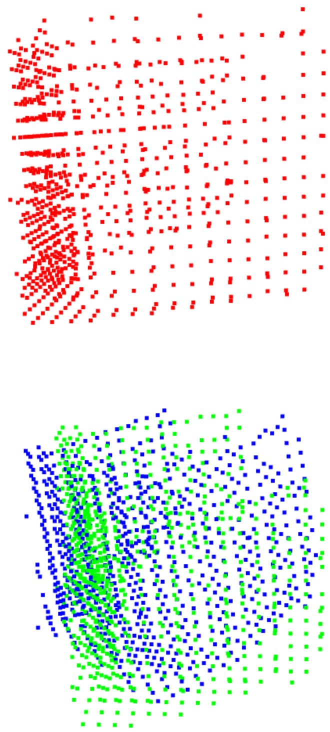

Figure 5: Registration: High offset between source and target

2.3 Generalized-ICP

Since the model's performance is highly effected by how challenging the data is, we follow DCPCR authors and we introduce Generalized-ICP (GICP) [6]. GICP is used on top of our network. It takes as input the estimated pose predicted by DCPCR. This input will be the initial guess for GICP algorithm making it more accurate. We confirm that the best results can only be achieved when enabling all parts of the network and finetuning the results using GICP. The figure below shows the estimated position with and without enabling GICP.

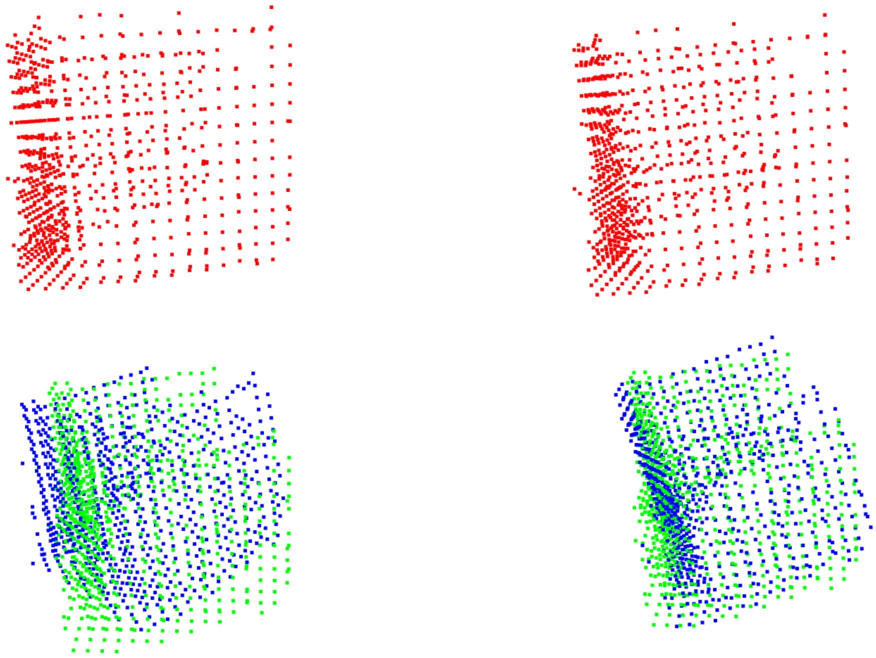

Figure 6: An example of the network estimated pose with (right) and without (left) GICP

3 Conclusion

The performance of the model depends on how challenging the data is. It also depends on how different the test cases are from the dataset it was trained on. With the current pretrained model on Apollo-Southbay and finetuned with GICP, we still can not achieve good results on our local data.

References

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

[2] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24:381–395, 1981.

[3] Ignacio Vizzo Giorgio Grisetti Jens Behley Louis Wiesmann, Tiziano Guadagnino and Cyrill Stachniss. Dcpcr: Deep compressed point cloud registration in large-scale outdoor environments. *IEEE ROBOTICS AND AUTOMATION LETTERS*, 2022.

[4] Louis Wiesmann, Andres Milioto, Xieyuanli Chen, Cyrill Stachniss, and Jens Behley. Deep compression for dense point cloud maps. *IEEE Robotics and Automation Letters*, 6(2):2060–2067, 2021.

[5] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[6] Aleksandr V. Segal, Dirk Hähnel, and Sebastian Thrun. Generalized-icp. 2009.