Paradigms for 6D object pose estimations: ICP-based, (PointNet-like) learning-based, as well as the combinations of them

1st Nan Xiao
University of California, San Diego
Jacobs School of Engineering
La Jolla, USA
naxiao@ucsd.edu

This report acts as the solution report to HW2 of CSE 275 3D Deep Learning, at University of California, San Diego. Codes are on the link: https://drive.google.com/file/d/1g37KO-Sn-qMAvbaufsz1uhkUaLtmooDu/view?usp=sharing

I. Our methods

A. ICP-based registration

I implemented the Iterative Closest Point (ICP) algorithm to estimate the object poses by iteratively re-assigning the closest points between the source and the target point clouds, as well as re-fitting the rotation matrices and translation vectors by optimizing the point-wise Mean Square Error (MSE). The results of ICP are stored in a JSON file according to the submission format and submitted.

To fine-tune the results, I adopted the following optimization strategies:

1) Voxel down-sampling on parts: Since the source (model) point cloud typically has a larger size than the target (scene) point cloud, I explored different ways for down-sampling the source point cloud to balance the size as well as reduce the time and memory costs, and finally decided the voxel down-sampling strategy with parts. Specifically, rasterized from a certain viewpoint in the world frame, the scene (target) point cloud typically only contains information on one side of the object. Hence, a down-sampling strategy that keeps half of the shape of the object while leaving out the other half can help align the source and target's geometric features.

I first divided the scene point cloud into two halves uniformly. After that, I adopt a heavy voxel down-sampling strategy by applying a larger voxel size on voxel down-sampling, and used a mild voxel down-sampling strategy on the other one with a smaller voxel size, so that the resulting point cloud has a shape similar to the target point cloud. Such method helps better align the geometric features and improved the performance in reality.

2) RANSAC-based global initialization: I used the RANSAC-based global initialization method to generate the initial transformation matrix to supervise the ICP more appropriately, by iteratively selecting the optimal subset of data, fitting the subset on a model, as well as removing the model's outliers. To better supervise the RANSAC procedures, I computed the FPFH features of the source and target point clouds and feed them into the RANSAC module for iterations. Such procedures are implemented with the help of the Open3D package.

3) Random initializations: To stabilize the results, I implemented random initializations and optimal result selections. I used the chamfer distance loss and the point-wise mean square error (MSE) as criteria to choose the optimal transformation. Specifically, for each estimation, I run RANSAC-based global initializations and ICP together 100 times. After getting the transformation, I use it to transform the source point cloud and compute the two losses specified above between the transformed source point cloud and the target point cloud. I chose the one with the minimal weighted sum of the two losses (with weights 0.1 for chamfer loss, and 0.9 for point-wise MSE loss) as the optimal registration result.

Due to the lack of rotation and transformation supervision, ICP is not capable of optimizing the shape-agnostic losses, and hence can hardly give accurate rotation predictions. To improve this, I tried a PointNet-like network (PointNetDense) instead.

B. Learning-based registration

I implemented a PointNet [1]-like network (PointNetDense) added with more convolution and linear layers to make it dense enough to learn the point cloud geometric features. The architecture is very similar to PointNet [1], which is given in the following figure 1. I simply replace the global feature extraction method: By directly convolving on the concatenations of the object id and the scene ids' one-hot vectors with 2 convolution layers of size 128 and 1024 respectively. Then, I concatenated the global features with the convolved point features, same as PointNet's concatenation, to merge them. Finally, I use two separate dense modules (each containing (1024, 512, 256, 128, 64, 3 for translation/6 for rotation) dense layers) to replace the segmentation network for the estimations of rotation and translation, respectively. I use shape-agnostic losses, including the rotation matrix's MSE and the translation's L1-norm error losses, to back-propagate the network.

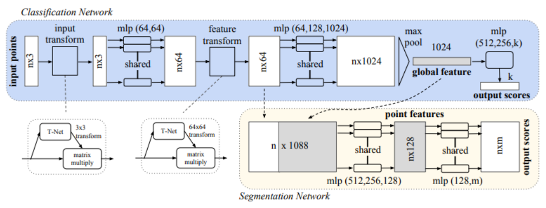

Fig. 1: PointNet figure

1) Feature Alignment network: To align the features of the input point cloud under different geometric transformations, I use a feature alignment network composed of 3 convolution layers (Conv(3, 64, 256, 512)) and 3 dense layers (Dense(512, 256, 128, 9)), as shown in the figure, and obtain the aligned local features finally. It helps make the rotation and translation predictions transformations-tolerant.

2) Local and global feature combinations: Poses for different objects and scenes can be different for given point clouds. To incorporate such global features, I transformed the object ID and scene ID information into one-hot vectors and fed them into Conv(81, 64, 256, 1024) to extract the global features. After that, I concatenate the aligned local features with the global ones and infer the rotation and translations with dense layers.

3) Separate predictions of rotation and translation: Rotations and translations are predicted in two separate dense networks instead of one since they have quite different features in predictions. Inspired by DenseFusion [2], I predicted pixel-wise rotations and translations and aggregated based on the confidences to find the most representative rotation for the point cloud.

C. Combining Neural Network with ICP

Optimizing only on shape-agnostic loss, PointNetDense performs poorly on translation while very brilliantly in rotation. Instead, ICP does well in translation while poor in rotations since it has no rotation and translation supervisions. Hence, I combined PointNetDense with ICP: I use PointNetDense to predict the rotation, and then feed the result to ICP to predict the translation.

II. Experiments and Results

A. ICP

The results of ICP on the benchmark test dataset are shown in the table below (where "Mine" refers to ICP+init+voxel-down-sampling). We can see that adding RANSAC-based global initializations and optimal initialization selections helps advance the accuracy above the baseline. In addition, the method with voxel down-sampling outperforms the other results in all accuracy metrics, demonstrating the effectiveness of the heavy-mild voxel down-sampling strategy in predicting both translations and rotations.

Pose accuracy				
	5deg1cm	10deg1cm	10deg2cm	15deg2cm
Mine	0.17	0.28	0.29	0.33
ICP+init	0.14	0.25	0.27	0.31
Baseline	0.13	0.24	0.27	0.31
ICP	0.02	0.25	0.33	0.61

B. PointNetDense

The results for pure PointNetDense with and without alignment are shown in table below. We can see that the alignment network significantly improved the rotation and translation results. However, due to their poor performance on translation, they all have very low pose accuracy (around 2%). This inspired me to improve by combining PointNetDense with ICP.

RRE(degree), RTE(cm), PtsError(cm)			
	RRE	RTE	PtsError
Mine	4.2415	0.2064	0.0038
No alignment	17.5680	0.6634	0.0138

C. PointNetDense+ICP

The results of combining PointNetDense with ICP are shown in the table figure. We can see that the pure ICP-based algorithm gives better predictions on translation because of the shape-aware supervision while giving poor predictions for rotation. The pure PointNetDense-based algorithm gives better ones for rotation while having poor translation predictions. After combining them, I refine the output by taking the PointNetDense's output rotation and the ICP's output (initialized by PointNetDense) translation as predictions. We see that the refinement gives much better predictions in rotation, and hence makes a significant improvement on pose accuracy scores.

RRE(degree), RTE(cm), Pose accuracy			
	RRE	RTE	5deg1cm
Mine	4.3216	0.0075	0.3693
ICP+NN, no refinement	10.8427	0.0075	0.0407
ICP optimal	39.4175	0.0232	0.1673
NN optimal	4.2415	0.2064	0.0007

Visualization of poses on test scenes including 1-10-1, and 2-14-1 are shown in figure 2.

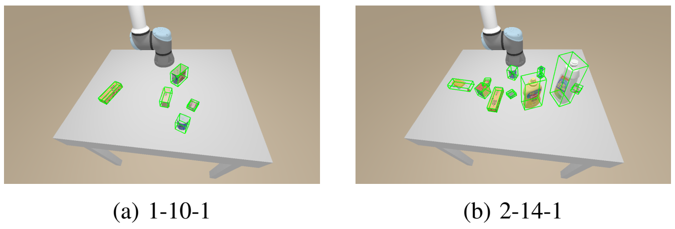
(a) 1-10-1 (b) 2-14-1

Fig. 2: Visualizations on poses

References

[1] Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3D classification and segmentation. In: Proceedings CVPR, pp. 652–660 (2017)

[2] Wang C, Xu D, Zhu Y, Martín-Martín R, Lu C, Fei-Fei L, Savarese S (2019) Densefusion: 6d object pose estimation by iterative dense fusion. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3343–3352