Programming Assignments 3 and 4 – 601.455/655 Fall 2021

Score Sheet (hand in with report) Also, PLEASE INDICATE WHETHER YOU ARE IN 601.455 or 601.655

(one in each section is OK)

Name 1	Shuojue Yang
Email	syang131@jh.edu
Other contact information (optional)	
Name 2	Zijian Wu
Email	zwu52@jhu.edu
Other contact information (optional)	
Signature (required)	I (we) have followed the rules in completing this assignment Shuojue Yang Zijian Wu

Grade Factor		
Program (40)		
Design and overall program structure	20	
Reusability and modularity	10	
Clarity of documentation and programming	10	
Results (20)		
Correctness and completeness	20	
Report (40)		
Description of formulation and algorithmic approach	15	
Overview of program	10	
Discussion of validation approach	5	
Discussion of results	10	
TOTAL	100	

CIS I - Programming Assignment 4

Zijian Wu and Shuojue Yang

1 Method Overview

In this programming assignment, our task is to compute the registration transformation F_{reg} using Iterative Closest Point (ICP) algorithm. During the step of finding the closest triangle in ICP algorithm, We respectively implemented Brute-Force Search, Simple Bounding Sphere Search, and Octree Search. We also compared their performance in boosting the search efficiency.

In Programming Assignment 3, we already realize the critical subroutine $FindClosestPoint()$ based on Brute-Force Search, Simple Bounding Sphere Search, and Octree Search. In this assignment, we complete the ICP algorithm [1] to implement registration on the basis of PA3. The workflow is as following steps:

1. Matching

   For every $F_{reg}\vec{q}_k$, where $\vec{q}_k$ is the point in the point cloud, use $FindClosestPoint()$ to find the matching triangles and the closest distance. Note that we assume the rotation and translation part of F_{reg} are I and $[0\ 0\ 0]^T$, respectively, in the first iteration. In addition, we conduct robust pose estimation in this step to filter out the outliers.

2. Transformation update

   After getting the point pairs in the last step, we use the 3D to 3D point cloud registration package developed previously to update the F_{reg}. Meanwhile, we compute three residual-error-related parameters, which are used to evaluate whether terminate this loop in the future step.

3. Adjustment

   Update threshold η_n. This parameter is used in matching step (robust pose estimation) to restrict the influence of clearly wrong matches on the computation of F_{reg}.

4. Iteration

   If the current F_{reg} satisfies the termination condition, this F_{reg} is our result. If not, go to next iteration.

1.1 Matching

In the matching step, the algorithm identifies the closest pair between the points cloud and surface model. As introduced in PA3, we implement three searching methods to solve this task, namely Octree, Bouding Sphere Search and Brute Force Search methods. With the algorithm proceeds, the 'bound' is iteratively reduced as the followed formulation:

$$bnd_k = ||F_n \cdot \vec{q}_k - \vec{c}_k||$$

where F_n denotes the F_{reg} updated in the previous iteration, $\vec{q}_k$ is the k-th point in point cloud and $\vec{c}_k$ is the corresponding closest needlepoint on surface model in previous match. With such a tighter bound, the speed of ICP algorithm can be significantly improved.

Then we use the search method to search the closest point on surface model for each element in the point cloud. In this assignment, as a default option, we adopt the Octree searching method taught in the class. In addition to the closest point coordinates, the algorithm also saves the distance of each point pairs.

Another difference with PA3 is the robust estimation part. With the iteration proceeding, we can leverage the distance values of each point pairs and their residual errors serving as thresholds to filter out some clearly wrong matches.

1.2 Transformation update

With the matched point pairs, we adopted the 3D point-cloud-to-point-cloud registration method introduced in PA1 and PA2 to compute the best transformation. After updating the registration transformation, we calculate residuals for each transformed point with the corresponding closest needle point. Then we can compute the residual error based values shown as following:

$$\sigma = \frac{\sqrt{\sum_k \vec{e} \cdot \vec{e}}}{k}$$

$$\varepsilon_{max} = \max_k \sqrt{\vec{e}_k \cdot \vec{e}_k}$$

$$\overline{\varepsilon} = \frac{\sum_k \sqrt{\vec{e}_k \cdot \vec{e}_k}}{k}$$

where $\vec{e}_k$ denote residual vector for the k-th point pair.

1.3 Adjustment

In order to realize robust pose estimation in the matching step, we need a threshold η_n to filter out the wrong pairs. Here we adopt 3 $\overline{\varepsilon}_n$ as this threshold. In practice, the setting of η_n is very tricky. On the on one hand, too tight threshold might delete some matching pairs that are actual correct. On the other hand, too loose threshold might not filter out the wrong pairs.

1.4 Iteration

The criterion of termination we adopted is to stop the iteration when $\sigma_n,\ \overline{\varepsilon}_n,\ (\varepsilon_{max})_n$ are less than desired thresholds (all take 0.01) and $\gamma \leq \frac{\overline{\varepsilon}_n}{\overline{\varepsilon}_{n-1}} \leq 1$. Here we assume that $\gamma = 0.95$.

2 Result and Discussion

2.1 Evaluation Metrics

The implementation of the ICP algorithm underlines the performance of both the accuracy and speed, which are respectively measured with Euclidean distance with ground-truth coordinates and the running time.

2.2 Unit test for robust pose estimation

We wrote a test program, $unit\_test.py$, for the most important subroutine $RobustPoseEstimation()$, which is used to filter out outliers (incorrect matching).

We artificially corrupt the given 'clean' surface points to simulate the noises in real applications. Specifically, we set noisy ratio as the ratio of correct points and noisy points which are corrupted by randomly shifting from the original coordinates. Based on these simulated datasets, we can investigate the effectiveness of the robust pose estimation module in alleviating the negative effects brought by mismatching. With various noisy ratios, we compared the performance between algorithms with and without using robust pose estimation in terms of the average error with ground-truth. As shown in Fig. 1, the robust estimation demonstrates a strong capability of improving the robustness under various noises ratios.

Furthermore, we also adopted the robust pose estimation to facilitate the registration of the given Debug

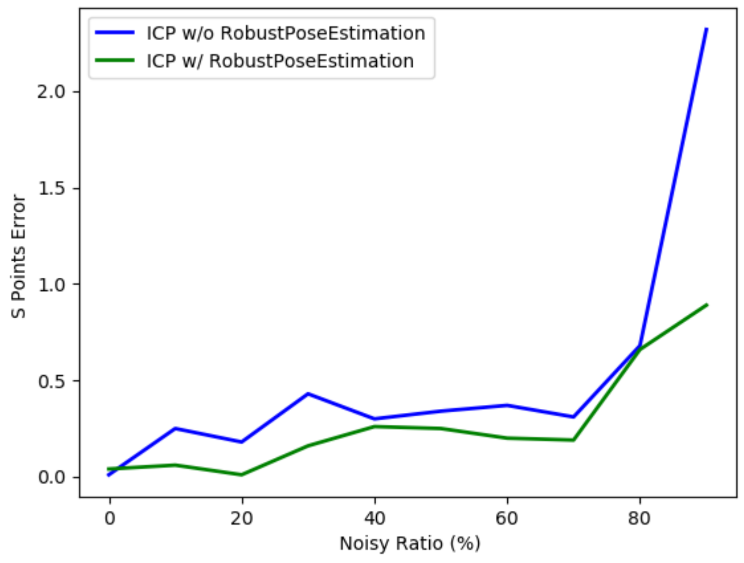

Figure 1: Quantitative comparison on simulated noisy dataset with different noisy ratios.

datasets. However, only Case E exhibits a reduction in error from 0.04 to 0.02 with robust pose estimation. Considering the discrepancy between the given output file and the answer file, we conjecture there could be some outliers leading to incorrect matches in this dataset.

2.3 Result of ICP

In this section, we first give a quantitative evaluation of the implemented ICP algorithm on various cases shown in Table 1. In addition, for both the Debug and Unknown cases, we compared the running time of the three made searching methods, which are Brute-Force Search, Simple Bounding Sphere Search, and Octree Search, to demonstrate the efficiency improvement.

2.3.1 Debug case

According to Table 1, the error less than 0.01 means that the predicted number is exactly the same as the given reference ground-truth, which verifies the correctness of our implemented module in finding the best rigid body transformation from point cloud to surface model by utilizing ICP algorithm. For cases D, E, and F, the errors are more than 0.01 regardless of whether implementing robust pose estimation.

Table 1: Quantitative evaluation of the implemented methods' accuracy. Due to the same closest point on mesh w.r.t. point, all the four methods achieve the same accuracy. The error is computed in terms of L2 Norm.

Case	s **error**	c **error**	mag **error**
A	0.0071	0.0068	0.0021
B	0.0082	0.0081	0.0023
C	0.0092	0.0093	0.0032
D	0.0109	0.0104	0.0044
E*	0.0252	0.0239	0.0119
F	0.0458	0.0452	0.0197

* : For case E, we use RobustPoseEstimation to filter out the outliers.

We guess that this phenomenon is because the current robust pose estimation method is not effective enough to filter out the outliers in the dataset. Another explanation may be that the ICP algorithm suffers from local optimal solution. Its performance critically relies on the quality of initialization, and only local optimality is guaranteed [2].

Table 2: Comparison of various methods in terms of running time for Debug case A-F.

Method	Running Time (s)					
	A	**B**	**C**	**D**	**E**	**F**
Brutal	760.11	1815.87	1632.02	1535.52	722.29	721.99
BoundSp*	64.89	154.46	132.93	194.63	59.84	78.42
Octree	**12.72**	**18.35**	**17.35**	**22.07**	**8.56**	**10.20**

* 'BoundSp' represents the Bounding Sphere Search method.

As shown in Table 2, 'Octree' denoting the Octree Search demonstrates the highest speed in implementing ICP in every case. Compared to naive Brutal Search, all the two smarter methods significantly improved the computation efficiency, especially Octree Search. Table 2 demonstrates that the clever search algorithm saves much time.

2.3.2 Unknown case

For Unknown cases, we save the results in $./OUTPUT$. As shown in Table 3, we simply give the time comparison in this section. For these three cases, the Octree based Search method significantly outperforms the Bounding Sphere Search method. Both of these two clever algorithms are much faster than Brute-Force Search method.

Table 3: Comparison of various methods in terms of running time for Unknown case G,H,J,K.

Method	Running Time (s)			
	G	**H**	**J**	**K**
Brutal	1181.00	1641.08	568.27	622.80
BoundSp*	107.56	47.59	47.59	49.24
Octree	**14.24**	**17.65**	**7.23**	**7.59**

* 'BoundSp' represents the Bounding Sphere Search method.

3 Contribution

Shuojue Yang is responsible for Octree Based ICP algorithm and Zijian Wu is resopnsibel for other search methods based ICP algorithm.

Bibliography

[1] P. Besl and N. D. McKay, “A method for registration of 3-d shapes,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.

[2] J. Yang, H. Li, and Y. Jia, “Go-icp: Solving 3d registration efficiently and globally optimally,” in *2013 IEEE International Conference on Computer Vision*, 2013, pp. 1457–1464.