Answers to COMP0119 (Course Work 1)

Name:Ruizhe Zheng **Student Id No.:** :15007260

1 Introduction

I have solved all 6 tasks for this coursework.

2 Task 1

This part is performed by function point_to_point() in my code. The function first use my kdtree_search() function which written by python built in KDtree search in spicy. This function build a KDtree use destination points and takes an array of source points as input and output an array of index of the closest point for each source point in the destination points array and an array of distance between 2 points.

Then we simply apply the result from ICP proof from the lecture to find out the optimal transformation. This is done by my find_optimal_transform() function.

The error I used is the sum of squared distance between the 2 closest points and whenever the error does not change much we stop the loop and output the mesh.

I achieved the tolerance of 0.0000001(the difference between previous error and error) in 14 iterations use bun000_v2.ply and bun045_v2.ply and the result is in the following figures.

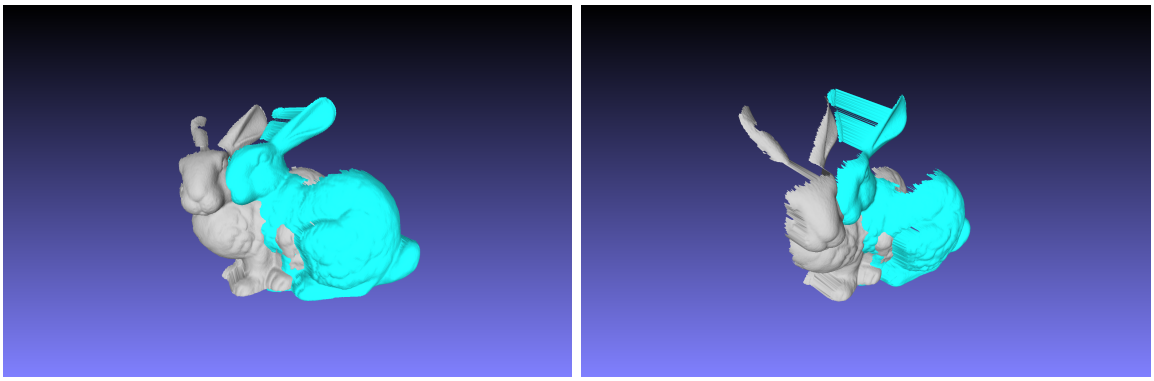

Figure 1: Before point to point ICP

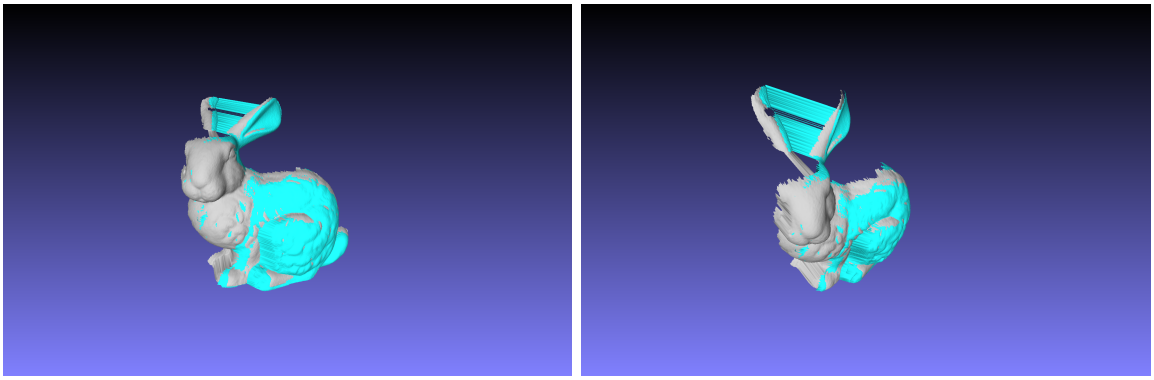

Figure 2: After point to point ICP

Like the above figures showed. The white one is bun000_v2.ply and it is used as a destination and the black one is bun045_v2.ply and it is used as a source where we use point to point ICP to move bun045_v2.ply to bun000_v2.ply.

Consider

$$E(R,t) = \sum_{i=1}^{n} w_i ||Rp_i + t - q_i||^2$$

Similar to the prove from the lecture

$$\begin{aligned}
\frac{\partial E(R,t)}{\partial t} &= 2\sum_{i=1}^{n} w_i ||Rp_i + t - q_i|| = 0 \\
&= \sum_{i=1}^{n} w_i R p_i + \sum_{i=1}^{n} w_i t - \sum_{i=1}^{n} w_i q_i = 0 \\
&= R\frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i} + t - \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}
\end{aligned}$$

Which we can see from there we get

$$\bar{q} = R\bar{p} + t$$

Where $\bar{q}, \bar{p}$ changes to $\frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}$, $\frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}$.
Similarly for the next part of the proof when we substitute t we get from rearranging the above equation, we get

$$\begin{aligned}
E(R,t) &= \sum_{i=1}^{n} w_i ||Rp_i + t - q_i||^2 \\
&= \sum_{i=1}^{n} w_i ||\tilde{p}_i||^2 - \sum_{i=1}^{n} w_i R\tilde{p}_i\tilde{q}_i + \sum_{i=1}^{n} w_i ||\tilde{q}_i||^2
\end{aligned}$$

Where $\tilde{p}_i = p_i - \bar{p}, \tilde{q}_i = q_i - \bar{q}$ From here we know that to minimise $E(R,t)$ we only need to maximise the middle part. And because of the weight is assigned to each point we only need to change the $nx3$ matrix $(\tilde{p}_1^T, \cdots, \tilde{p}_n^T)$ to $(w_1\tilde{q}_1^T, \cdots, w_n\tilde{q}_n^T)$ and the rest part of SVD decomposition is the same as the proof we discussed during the lecture and we still get

$$R = VU^T$$

in the end as the rotation matrix.

3 Task 2

For this task I used bun000 and the rotated bun000 and destination and source. The rotation is done by function rotatez(). The results shows that we need a good ish pre alignment for point to point ICP to preform correctly. we can see once the mesh it flipped by rotation, the result of ICP is just not great since the point to point ICP cannot flip the image back. From my test results i found out that my point to point ICP preforms well in the range of +60 degrees and -60 degrees. In this range I get a good alignment but out of this range I start to get wrong alignment because the initial position of the mesh is flipped in some degree that the ICP just cannot correct it anymore. The following figure shows that the comparison between +60 degrees and +70 degrees. All the other results are in stored which I can show them if needed during the demo session.

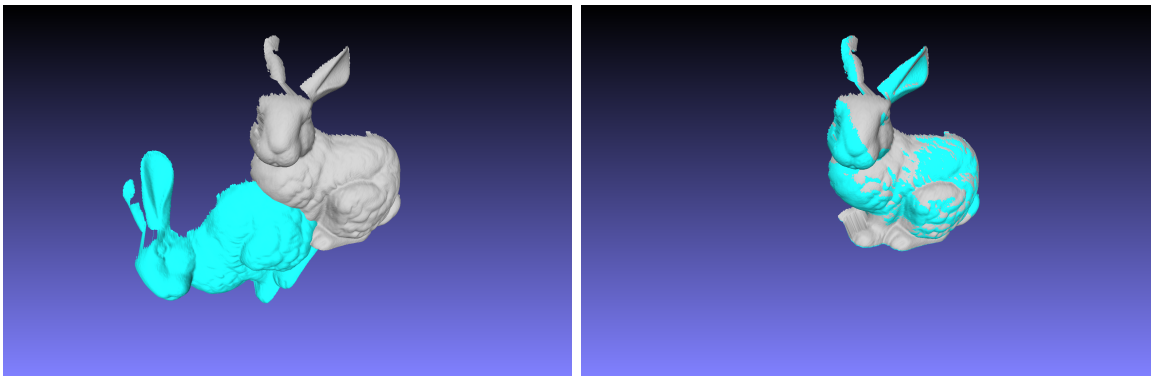

Figure 3: Before vs After point to point ICP for +60 degrees rotation

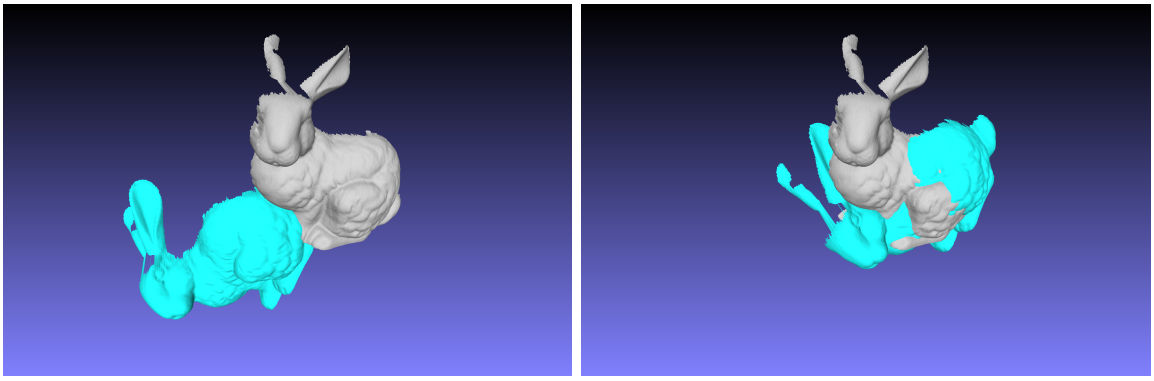

Figure 4: Before vs After point to point ICP for +70 degrees rotation

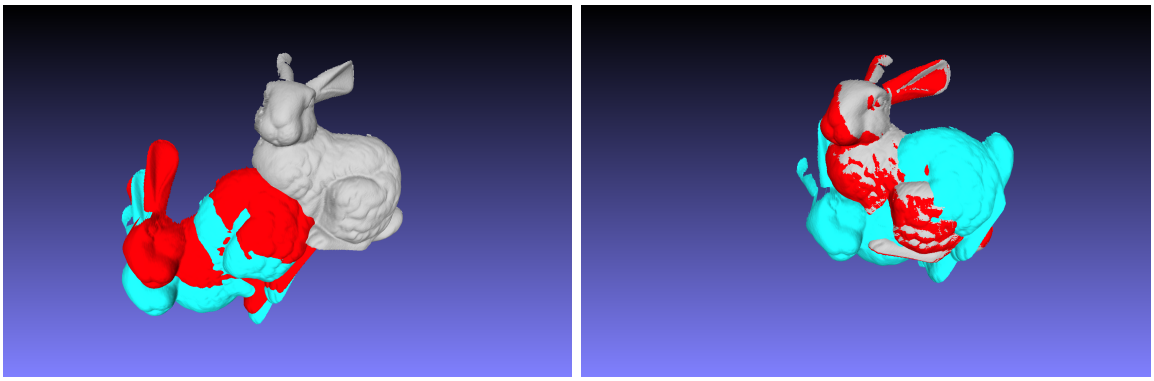

Figure 5: Before vs After point to point ICP for +60 and +70 degrees rotation

4 Task 3

For this task I used bun000 and bun045 for destination and source. The noise is added via function addnoise(). The p2p error increases as the noise level increases, but for all noise level I have used (0.01 to 0.15) the alignment preform fairly good and the iteration seems do not depend on noise level since they are all roughly the same. Here is the graph of how noise level effects iteration numbers and p2p error.

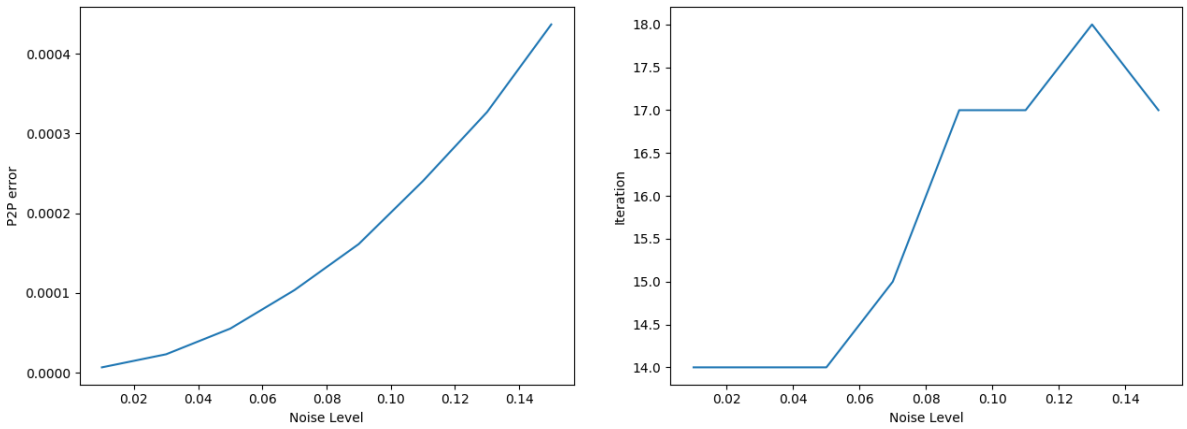

Figure 6: Noise vs P2P error graph and Noise vs Iteration graph

And the following figures show a example of the noised mesh and alignment.(0.03) Since the noise level after this it is really hard to recognise the bunny shape.

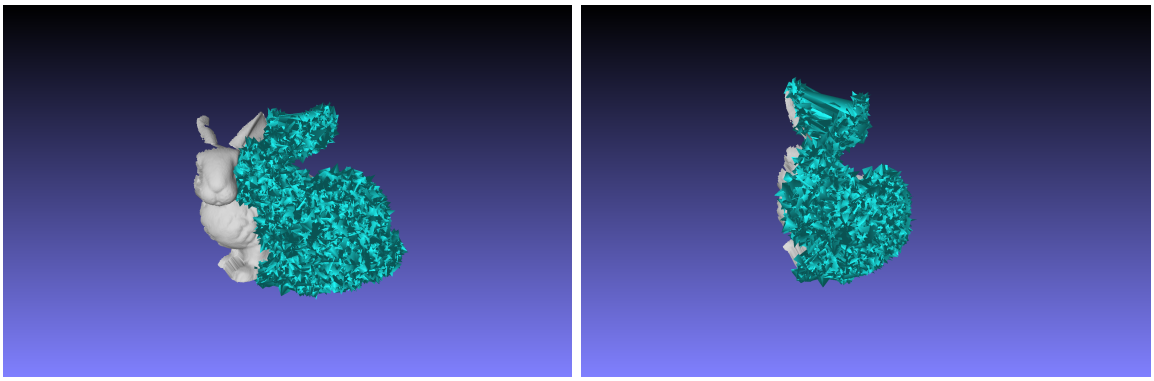

Figure 7: Before vs After alignment for noise level 0.03

5 Task 4

The sampling method I used is uniform_sampling and I used bun000 as destination and bun045 as source points. The run time decrease significantly as the sample size goes down, but the total error did not vary much for my case. The performance is still good even with 10%(0.1 sample size) of the points from source being used and for all test it took 14 iterations. Here are graphs show the effect of sample size on time and total error.

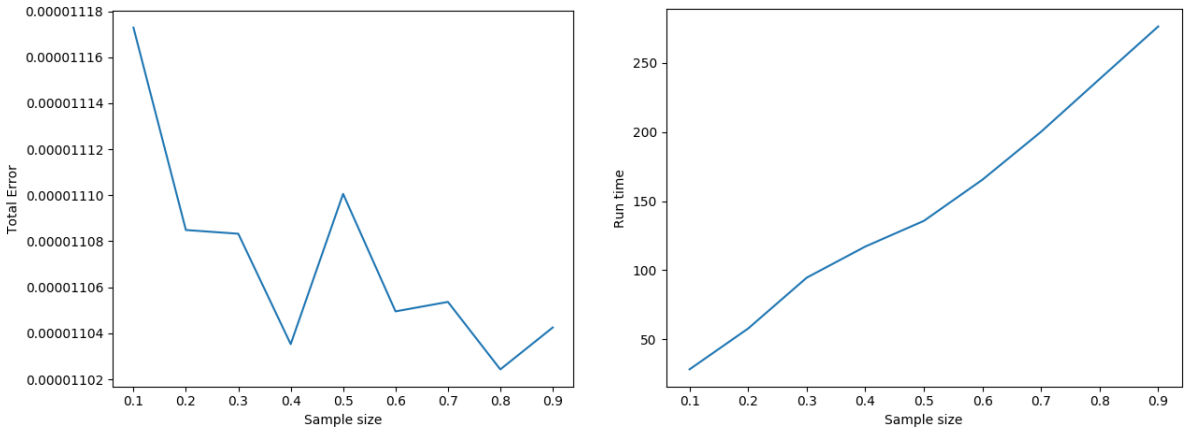

Figure 8: Noise vs P2P error graph and Noise vs Iteration graph

And here is an example of the performance (0.1 sample size)

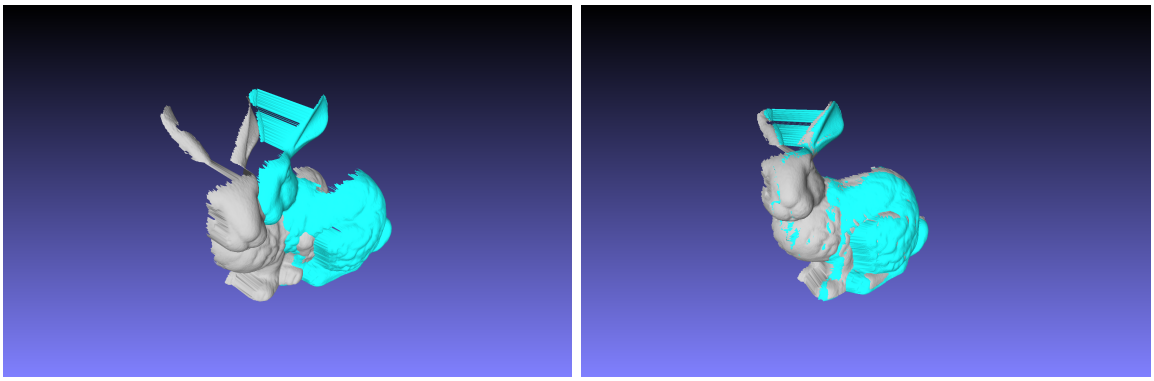

Figure 9: P2P ICP under 0.1 sample size

From this I think if the pre alignment is good enough we can safely use uniform sample to reduce the run time without decrease the quality.

6 Task 5

For this task I used bun000, bun045, bun180, bun270, bun315. For task 5 the method I used is to first align 2 of the mesh then save that as an input of the next destination. The loop it until all 5 meshes are aligned. The downside of the method is the transformation error adds up as you put more meshes into it. The upside of this method is we used all meshes throughout the alignment. Which can be beneficial depends on how pre alignment quality is.

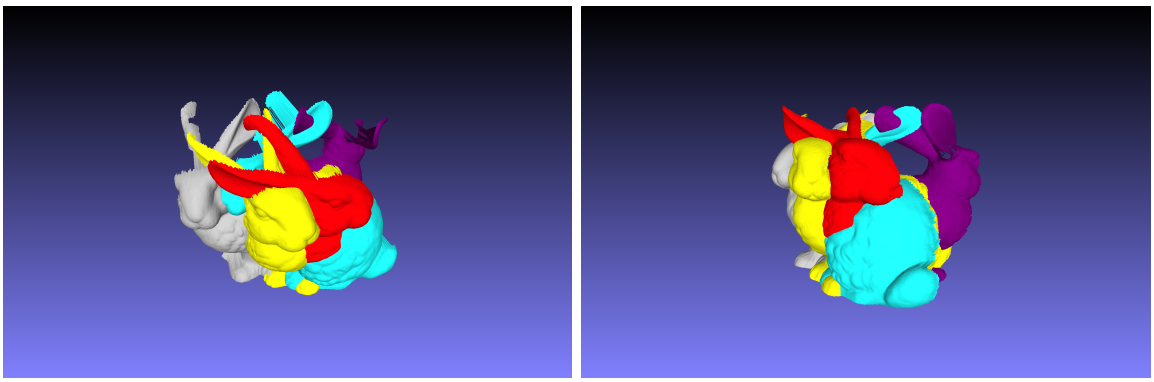

Figure 10: 5 meshes before alignment

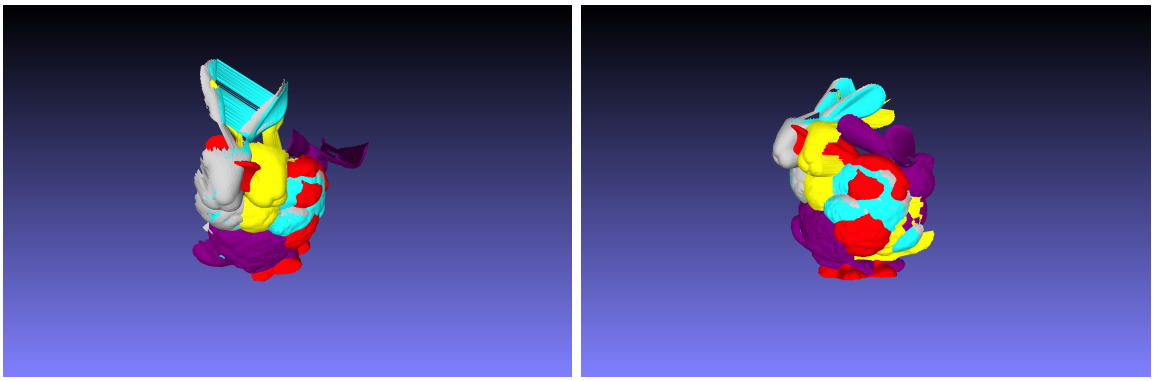

Figure 11: 5 meshes after alignment

From the above figure we can see that red(bun270) and purple(bun180) did not aligned well where white(bun000), blue(bun045) and yellow(bun315) are aligned decently. The reason for this can be the poor pre alignment of bun270 and bun180 and the cumulative error during the process. We may reduce this by align everything to one mesh in a single loop but this rise the problem that not all meshes are considered while preforming the ICP.

7 Task 6

For is task I used bun000 and bun045 again as demonstration. I first estimate the normal for all points in the destination use python open3d built in point could normal estimation.(Since the question states that we can assume we are given the normal I do not think we have to implement it by ourselves) Then I follow the procedure from the paper 'Linear Least-Squares Optimisation for Point-to-Plane ICP Surface Registration' published by Kok-Lim Low from University of North Carolina at Chapel Hill. The detailed thought process will be showed during demo session. This algorithm preforms good for my case. It converges in less iterations.(8 in my case) and the quality is still good for these 2 meshes. This supports the result from the above paper where if the pre alignment is good enough point to plane is more efficient. The result is below and detailed explanation will be made during demo session. And all of the test results such as for angles and for samples are available if asked.

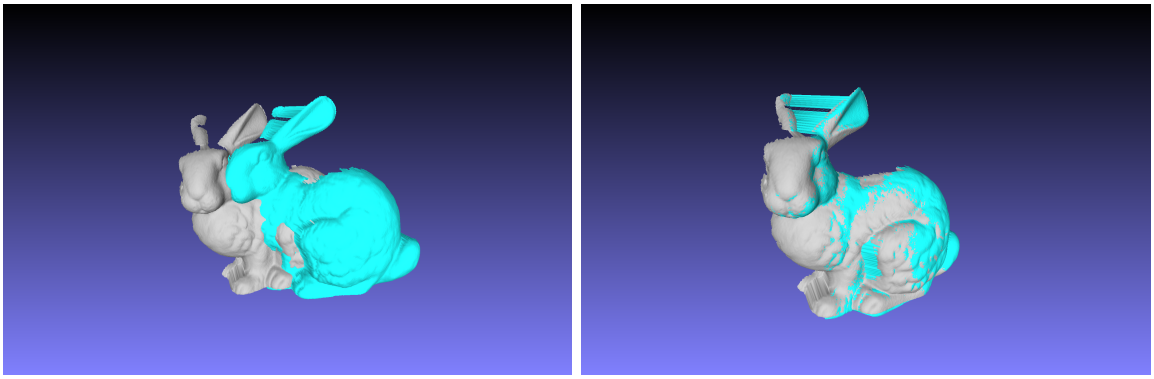

Figure 12: Before point to plane ICP and after point to plane ICP

References

[1] Kok-Lim Low *Linear Least-Squares Optimisation for Point-to-Plane ICP Surface Registration.* Technical Report TR04-004, Department of Computer Science, University of North Carolina at Chapel Hill, February 2004.