Acquisition and Processing of 3D Geometry
Coursework 1

Jiacheng Liu

Due date: 18th Feb. 2018

Introduction

In this coursework, 6 questions were attempted to be completed totally and some results were obtained as well. The algorithm was implemented in C++ programming language by my own self. This report will give a description about used approaches in each task and show related screenshots of results. In the first task, the point-to-point iterative closest point (ICP) algorithm was programmed and is able to align two given meshes successfully. Next, the ICP algorithm was tested based on two meshes which have the same structure but rotation angles. In the part of task 3, I added zero-mean Gaussian noise with different noise level on source mesh (M2) and tried to use ICP algorithm to align noised source mesh (M2') to target mesh (M1). Then in the task 4, in order to reduce computational cost, subsampled version of the original source mesh (M1) were used in ICP algorithm and an observation about experiment accuracy was recorded. For the task 5, the ICP algorithm was applied for the alignment of five different meshes. Lastly, the vertex normal was considered into the implementation and point-to-plane ICP algorithm was completed.

1. Task 1: point-to-point ICP algorithm

All mesh models had been converted from PLY to OFF format by using meshLab software. This task aims to align M2 to M1 by using point-to-point ICP algorithm. After loading both two meshes, KNN search approach is applied to find nearest neighbor points in M1 (p_i) according to query points in M2 (q_i). Noted that there is only one neighbor point in M1 will be matched for each point in M2. After obtaining matched points set (p_i, q_i), the rigid transformation is calculated based on corresponded matched point pairs with SVD method. The specific calculation could be seen as below,

1) Compute the mean value of vertexes: $\bar{p} = \frac{\sum_{i=1}^{n} p_i}{n}$, $\bar{q} = \frac{\sum_{i=1}^{n} q_i}{n}$
2) Compute the covariance matrix: $\mathrm{A} = \sum_i (p_i - \bar{p})(q_i - \bar{q})^T$
3) Because $\mathrm{A} = U\Sigma V^t$, then we could get $\boldsymbol{R} = VU^T$ and $\boldsymbol{t} = q - \boldsymbol{R}\bar{p}$, where $\boldsymbol{R}$ is the rotation matrix and $\boldsymbol{t}$ is the translation vector.

The rigid transform is applied on M2 and its position updated as well. The above whole process will be repeated for several times to make M2 closer to M1.

In order to observe the alignment effect with intuitive version, two meshes are set with different colors. In additionally, a simple function menu, which could be seen as Figure 1, was built to simplify the test process, including adding input filed and multiple button. The first task could be tested by clicking the button named *"Point-to-point ICP (two meshes alignment)"*. The corresponded iteration times could be modified by inputting different integer in the test field after *"Iteration"*. The *"Initialization"* button is used to restore the default meshes with default setting in the viewer. Noted that the default M1 and M2 are set as "bun000.off" and "bun045.off" respectively and M2 have been translated in x axis direction.

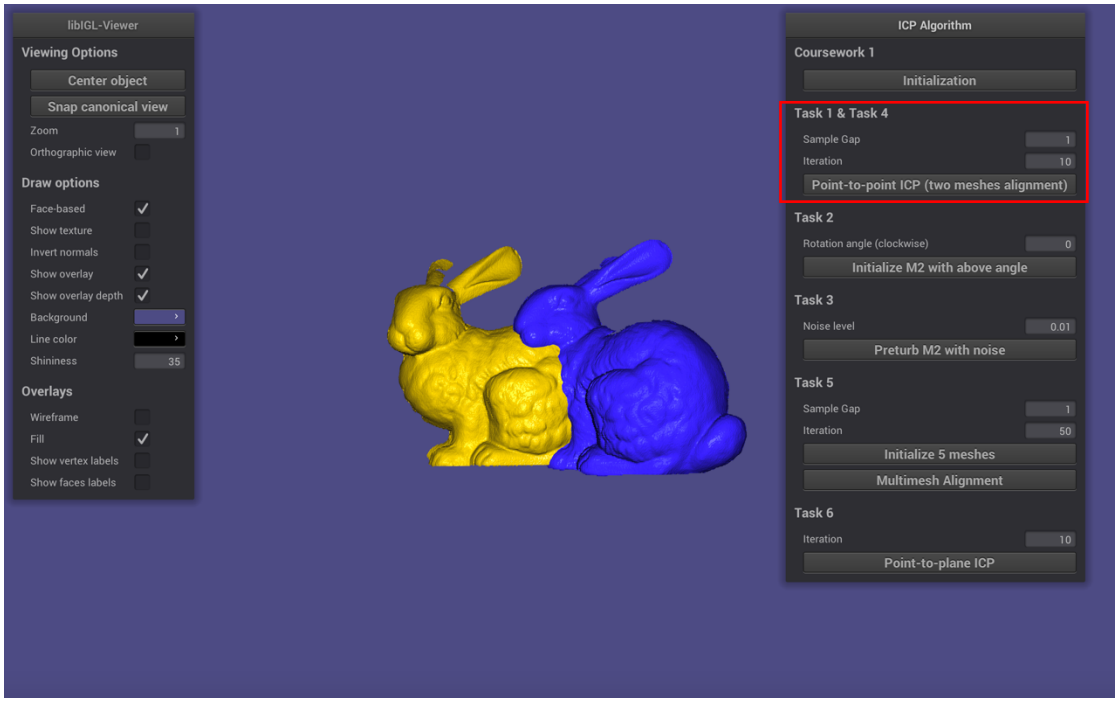
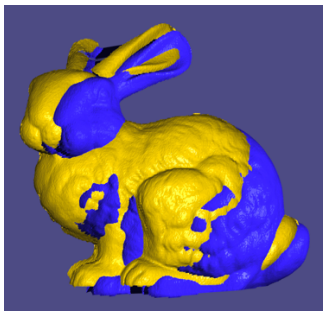
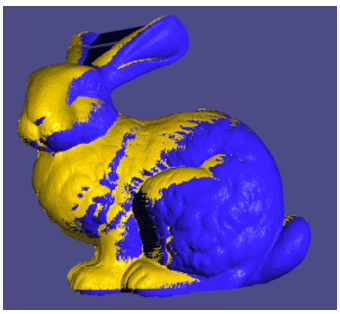


Figure 1: The developed interface

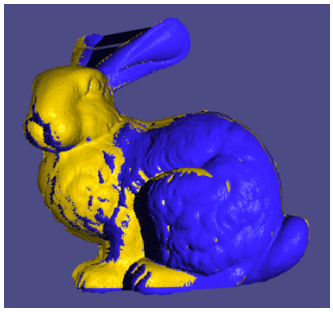
According to different iteration times, the alignment results are different. Figure 2 shows the comparison between alignment results between M1 and M2 with various iteration times.

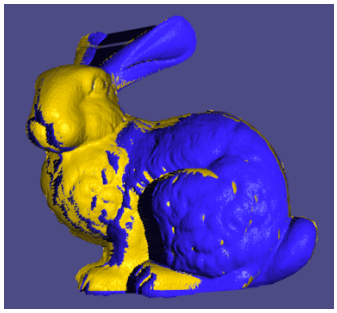


(a) 10 times



(b) 20 times





(d) 30 times (e) 40 times

Figure 2: alignment results with different iteration times

In the aspect of weighted point-to-point ICP algorithm, we may want to assign a weight $w_i \in [0,1]$ to neighbors p_i in target model to make reliable points become more important. The alignment error function could be expressed as $E(\mathbf{R},\mathbf{t}) = \sum_{i=1}^{n} w_i \left|\left|\mathbf{R}p_i + \mathbf{t} - q_i\right|\right|^2$. By using the same derivations discussed in class, we can get

$$0 = \frac{\partial E}{\partial \mathbf{t}} = \sum_{i=1}^{n} 2\, w_i(\mathbf{R}p_i + \mathbf{t} - q_i) = 2\mathbf{t}\left(\sum_{i=1}^{n} w_i\right) + 2\mathbf{R}\left(\sum_{i=1}^{n} w_i p_i\right) - 2\sum_{i=1}^{n} w_i q_i$$

Then we get

$$\mathbf{t} = \bar{q} - \mathbf{R}\bar{p}$$

where

$$\bar{p} = \frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i}, \qquad \bar{q} = \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}$$

Compute the centered vectors

$$x_i \coloneqq p_i - \bar{p}, \quad y_i \coloneqq q_i - \bar{q} \qquad i = 1,2,\dots,n.$$

Get the $d \times d$ covariance matrix

$$\mathrm{A} = XWY^T$$

Where X and Y are matrices with the dimension of $d \times n$ and uses x_i and y_i as their columns respectively. W represents a $n \times n$ matrix whose diagonal is $(w_1, w_2, \dots, w_n)$. Compute the singular value deposition $\mathrm{A} = U\Sigma V^T$, then the rotation matrix could be solved as

$$\mathbf{R} = UV^T$$

In conclusion, the solution for rigid transformation could be expressed as below.

$$\mathbf{R} = VU^T, \mathbf{t} = \bar{q} - \mathbf{R}\bar{p}$$

2. Task 2: source mesh with rotation

In this task, it is required to rotate a mesh (here is M1) with any angles and set this rotation version as the source mesh (M1'). The ICP algorithm implemented in task 1 is used to align rotated mesh to original mesh. The code to rotate loaded mesh could be seen as Figure 3. In this task, full mesh "bun000.off" was used as the default mesh.

```
// rotation matrix
Eigen::Matrix3d R;
R << AngleAxisd(rotation_angle * M_PI/180,
    Eigen::Vector3d(0,0,1)).toRotationMatrix();

// get new source mesh 2
igl::readOFF(TUTORIAL_SHARED_PATH "/bun000.off", V1, F1);      //m1
V2 = V1 * R;
F2 = F1;
```

Figure 3: Code for rotating given mesh with certain angle

To obtain the alignment results with different rotation angles, a text field is set specially to modify the rotation angle at any time (Figure 4). The rotation is applied along z axis in clockwise direction and the default rotation angle is 0 degree. After initialize source mesh M2 by clicking button named *"Initialize M2 with above angle"*, the button named *"point-to-point ICP"* located in *"Task 1 & Task 4"* region should be pressed to align meshes.

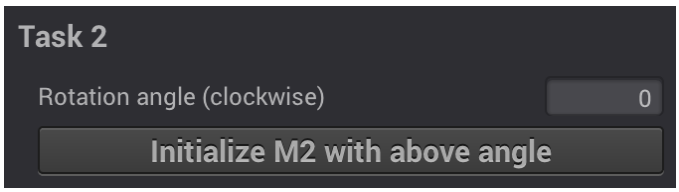


Figure 4: Interface for task 2

Table 1 shows the results for source mesh with different rotation angels.

Table 1: Alignment results for source mesh with different angles

Rotation angle (clockwise)	Success alignment?	Iteration
5	Yes	28
10	Yes	38
20	Yes	40
30	Yes	45
40	Yes	50
50	Yes	51
60	Yes	52
70	Yes	53
80	No	/
-60	Yes	43
-70	Yes	/

From above table, it could be found that the required iteration times for success alignment increases with the rotation angle increase. In addition, we can find that there is a limitation for success alignment for both clockwise and anti-clockwise rotation, which means that the ICP algorithm cannot work very well when two meshes have very large angle difference. Additionally, this algorithm was also applied on the mesh called *"bunny.off"*(Figure 5) which is a complete rabbit model. It could be found that the angle limitation of success alignment of the complete model is higher than that of incomplete model, which are 33 iterations for 70 degrees in clockwise and 153 iterations for 75 degrees in anti-clockwise direction. This data could be explained as complete mesh has more reference vertexes hence the algorithm could realize more accurate alignment with less iterations.

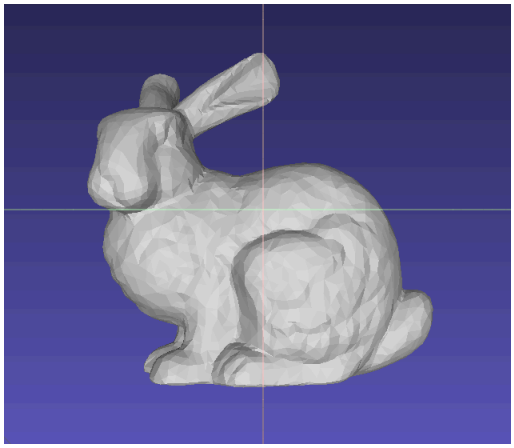


Figure 5: "bunny.off"

3. Task 3: noisy source mesh

According to the material requirement, zero-mean Gaussian noise are added to M2 with different amount and the noisy model is set as source mesh. ICP algorithm developed in task 1 is applied to align this noisy model to target mesh M1. The function for adding noise with certain amount could be seen as Figure 6.

```
////////////////// used to add gaussian noise to mesh ///////////////////////
MatrixXd icp::addNoise(MatrixXd source, double noiseDiv){
    default_random_engine randGen;
    normal_distribution<double> gaussian_dis(0.0, noiseDiv);    // gaussian_dis(dis_mean, dis_div)
    Eigen::Vector3d noise;
    for (int i=0; i<source.rows(); i++){
        noise[0]=gaussian_dis(randGen)/100;
        noise[1]=gaussian_dis(randGen)/100;
        noise[2]=gaussian_dis(randGen)/100;
        source.row(i) = source.row(i) + noise.transpose();
    }
    return source;
```

Figure 6: Code for adding noise to source mesh with certain amount

In the viewer interface for this task, a text field named *"Noise level"* is displayed to simplify the experiment procedure (Figure 7). The default noise level is 0.01. After initializing the source mesh with noise, clicking the button *"point-to-point ICP"* in *"Task 1 & Task 4"* region to align meshes.

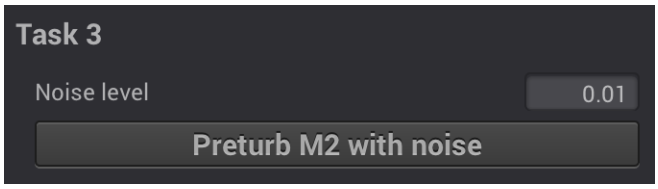
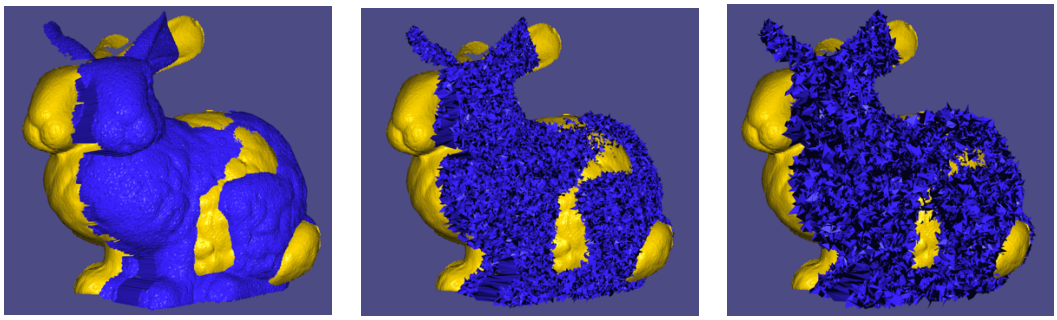


Figure 7: Interface for task 3

In this task, default mesh is set as the complete mesh *"bun000.off"*. The mesh with different noise level could be seen as Figure 8.



(a) 0.01 (b) 0.1 (c) 0.2

Figure 8: Model M2 with different noise amount

The success alignment result could be seen as Figure 8.

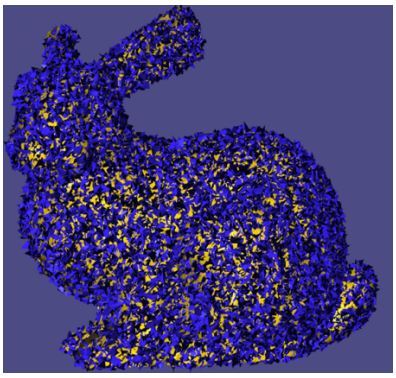


Figure 9: Success alignment result (noise level =0.1)

Table 2 shows the test alignment result for source mesh with different noise amount, the recorded items include noise level, if the algorithm could achieve the success alignment and the iteration times to get success alignment.

Table 2: Alignment results for task 3

Noise level	Success alignment?	Iteration
0.1	Yes	37
0.2	Yes	37
0.3	Yes	39
0.4	Yes	40
0.5	Yes	40
0.6	Yes	40
1.0	Yes	43
2.0	Yes	45

From above table, it could be found that the algorithm is able to realize success alignment even for source mesh with quite large amount of noise and noise level has slight influence on algorithm accuracy. The iteration times increase with noise amount rises.

4. Task 4: subsample version ICP algorithm

In this task, the subsampled source mesh is used to speed up the alignment process. The sampling code could be seen as Figure 10.

```
////////////////// used to get sampled point set from source mesh ////////////////////
MatrixXd icp::getSampleSource(MatrixXd source, int sample_gap){
    int pts_num = round(source.rows()/sample_gap);

    Eigen::MatrixXd sample = Eigen::MatrixXd(pts_num,3);
    for(int idx=0;idx<pts_num;idx++){
        sample.row(idx) = source.row(idx*sample_gap);
    }

    return sample;
}
```

Figure 10: Function to get subsampled source mesh

This function could be accessed by input wanted sample gap in the text field named *"sample gap"* in the region of *"Task 1 and 4"*, then click the button "point-to-point ICP" to start alignment.

The alignment accuracy between subsampled ICP with different number of samples could be seen as Table 3. The source and target mesh are set as *"bun000.off"* and *"bun045.off"* respectively.

Table 3: Aliment accuracy for ICP with different sample gap

Sample gap	Success alignment?	Iteration times
1	Yes	31
5	Yes	31
10	Yes	31
20	Yes	31
30	Yes	31
40	Yes	31
100	Yes	33
200	Yes	35
400	Yes	43
500	Yes	48

From the above table, it could be found that the alignment accuracy decreases with the increase of the sample gap though the decrease is small. We can notice that the less sampled points cause more errors in the estimation of rigid transformation, hence the iteration times are required to rise as well to improve the alignment results.

5. Task 5: multi-mesh alignment

In the part of task, it is required to align 5 meshes (M1, M2, M3, M4 and M5) by applying ICP algorithm. From task 2, it could be learned that the alignment might be failed if two meshes have quite large angle difference. Therefore, the meshes I selected for testing in this part is *"bun270.off"*(M1), *"bun315.off"*(M2), *"bun000.off"*(M3), *"bun045.off"*(M4) and *"bun090.off"* (M5). The angle difference of each adjacent pairs is less than 60 degrees, which is able to increase the success possibility for mesh alignment. The approach that I used to align these 5 meshes is aligning adjacent pairs step by step, i.e. cumulative alignment. The specific expression for this approach is written below.

$$M_{12} = M_2 + align(M_2 \leftarrow M_1)$$
$$M_{123} = M_3 + align(M_3 \leftarrow M_{12})$$
$$M_{1234} = M_4 + align(M_4 \leftarrow M_{123})$$
$$M_{12345} = M_5 + align(M_5 \leftarrow M_{1234})$$

The interface for this task contains one text field and two buttons (Figure 10). The text field allows user to input certain iteration times, the default times is 50. The first button is used to display the initial five meshes without alignment, while the second button is used to implementing the ICP algorithm onto these meshes and show the final alignment results.

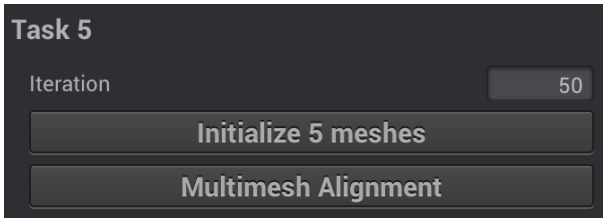
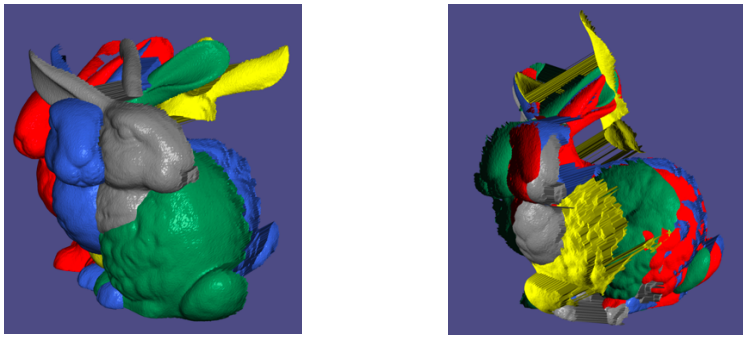


Figure 10: Interface for task 5

The comparison between views of initial multi-mesh and alignment result after 50 iteration times can be seen as Figure 11.



(a) Initial display (b) 50 iterations

Figure 12: Comparison between initial display and alignment results of multi-mesh

From above results, it could be found that there still are some gaps between meshes, which means that the alignment is not perfect enough. Additionally, the yellow model *("bun090.off")* is not aligned successfully with other models.

The meshes used in this task is incomplete mostly, which means it is possible to find no suitable neighbors in KNN tree for current query point. The characteristics of cumulative alignment ease up this limitation, because it allows increased points to come into neighbor search and rises the possibility for query point to find matched neighbors in KNN library. However, cumulative alignment is influenced significantly by previous alignment results, as each alignment implementation is based on the previous combined meshes. If the previous alignment is not success, the following alignment will have much less possibility to apply correct rigid transformation. What is more, the computational cost for this approach is quite expensive and the running time for this task is a little long.

6. Task 6: point-to-plane ICP

The idea of point-to-plane ICP is that using the normal information to express the error equation,

$$E(R,t) = \sum_i \left|\left|(Rp_i + t - q_i) \cdot n_i\right|\right|^2$$

Where n_i represents the normal of plane that conformed by neighbor points in target model. The overall process of point-to-plane ICP could be summarized as below.

1) According to q_i in source mesh, search neighbors p_i in target mesh and get normal n_i.
2) Solve $Ax = b$, where $A = \begin{bmatrix} \leftarrow p_1 \times n_1 \rightarrow & \leftarrow n_1 \rightarrow \\ \leftarrow p_2 \times n_2 \rightarrow & \leftarrow n_2 \rightarrow \\ \vdots & \vdots \end{bmatrix}$ and $b = \begin{bmatrix} -(p_1 - q_1) \cdot n_1 \\ -(p_2 - q_2) \cdot n_2 \\ \vdots \end{bmatrix}$.
3) The solution of x could be expressed as $x = [r_x \quad r_y \quad r_z \quad t_x \quad t_y \quad t_z]^T$, where the first three elements are rotation angels along three directions and last three elements are translation distance in three directions. Use the first three elements to form the 3×3 rotation matrix $\boldsymbol{R}$ and last three elements to get translation vector $\boldsymbol{t}$.
4) Apply rigid transformation matrix on source model.

Unlike point-to-point ICP, which have a closed-from solution, point-to-plane ICP should be solved with a nonlinear least squares approach [1]. This algorithm could be accessed by pressing the button *"Point-to-plane ICP"* in *"Task 6"* region (Figure 13). The specific iteration number could be modified by inputting integer in the text field after *"Iteration"*. Noted that the point-to-plane ICP only can be applied on the alignment of two meshes in this assignment.

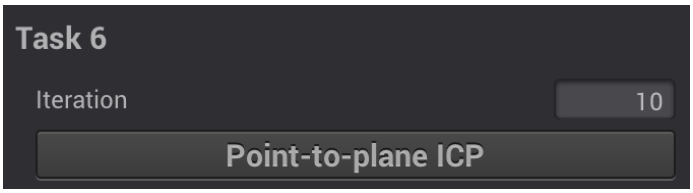
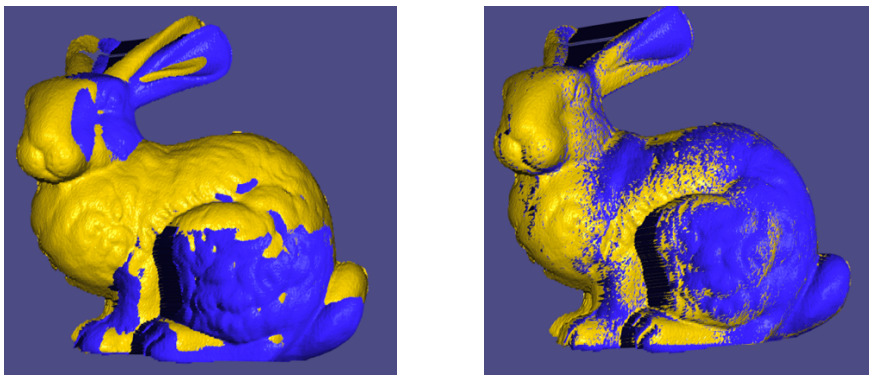


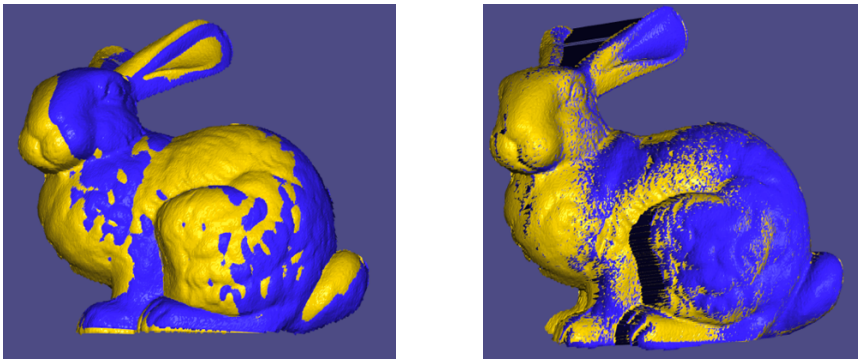
Figure 13: Interface for task 6

The alignment comparison between point-to-point and point-to-plane ICP could be seen as below. It could be found that, with the same number of iteration, point-to-plane ICP is able to realize better alignment results than point-to-point ICP. It means that point-to-plane ICP has higher alignment accuracy than point-to-point ICP. However, point-to-plane ICP is more expensive and is generally slower than point-to-point ICP.



(a) point-to-point ICP (b) point-to-plane ICP

Figure 14: M1="bun000.off", M2="bunn045.off", iteration = 5



(a) point-to-point ICP (b) point-to-plane ICP

Figure 15: M1="bun000.off", rotation angle=10°, iteration = 5

Reference

[1] Kok-Lim Low. February 2004. "Linear Least-Squares Optimization for Point-to-Plane ICP Surface Regestration". Technical Report TR04-004.