Acquisition and Processing of 3D
Assignment 1

Name: **Yinji Zhu** ID: **18062537** Email: **ucaby22@ucl.ac.uk**

February 21, 2019

Introduction

In the assignment, different aspects of the ICP algorithm were explored by the sub-datasets of bunny as the models, which uses the library from Stanford 3D Scanning Repository[1]. Six tasks exist in this assignment. All these tasks were achieved in the experiment and combined to one GUI.

Question 1

The main task of this question question requires to transform mesh M_2 to similarly mesh M_1 and combine the overlap position together. *bun000.ply* and *bun045.ply* has sufficient overlap, so they are chosen as the meshes M_1 and M_2 respectively.

To start with, the algorithm uses the whole points in M_1 and M_2 to match. Using the library FLANN[2], the nearest point between M_1 and M_2 can be obtained. Noting that the M_1 has several points matching to the same points in M_2, the matrix of M_2-M_1 needs to be rearranged.

Next, the barycenters for each point set needs to be calculated, the equations can be seen below:

$$\overline{\mathbf{p}} := \frac{1}{m}\sum_{i=1}^{m}\mathbf{p}_i \text{ and } \overline{\mathbf{q}} := \frac{1}{m}\sum_{i=1}^{m}\mathbf{q}_i$$

Then, recenter point sets:

$$\hat{\mathbf{p}}_i := \mathbf{p}_i - \overline{\mathbf{p}} \text{ and } \hat{\mathbf{q}}_i := \mathbf{q}_i - \overline{\mathbf{q}}$$

To minimize the function: $\min_{\mathbf{R},\mathbf{t}}\sum_i \|\mathbf{p}_i - \mathbf{R}\mathbf{q}_i - \mathbf{t}\|^2$, the least-squares solution reduces to:

$$\mathbf{A} = \sum_{i=1}^{m}(\mathbf{q}_i - \overline{\mathbf{q}})(\mathbf{p}_i - \overline{\mathbf{p}})^T = \sum_{i=1}^{m}\hat{\mathbf{q}}_i\hat{\mathbf{p}}_i^T$$

Using singular value decomposition to extract rotation form A, the decomposed matrix can be seen below:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$$

where all of A, U, Σ, and V are 3 by 3 matrix.

Consequently, the rotation $\mathbf{R}$ and transformation $\mathbf{t}$ can be calculated by equations:

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T \text{ and } \mathbf{t} = \overline{\mathbf{p}} - \mathbf{R}\overline{\mathbf{q}}$$

Using the values of R and t, the bunny set of M_2 can be moved closer to M_1. In this case, doing the iteration several times, the two parts of bunny will be aligned.

The original model of M_1 and M_2 can be seen in fig.1,where the green model is the bun045.ply without rotation.

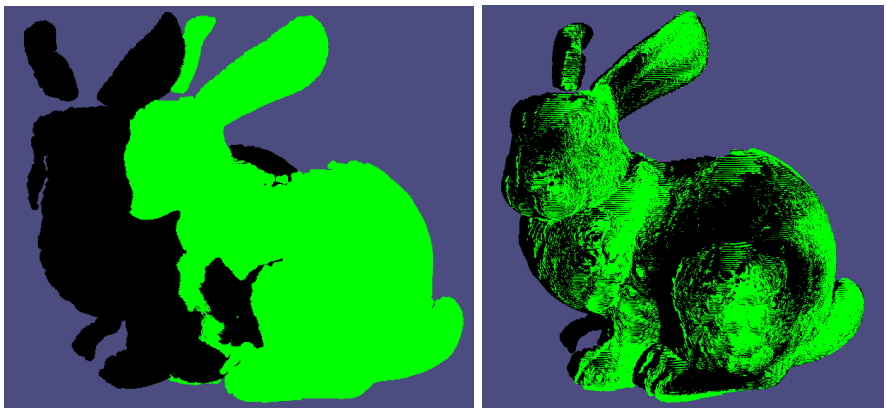

Figure 1: The original models of M_1 and M_2

Figure 2: The models after 70 times iterations

Iterating 70 times, the combined bunny can be seen in fig.2, which seems to have a nice fit.

Also, the judgement of bad pairs has been added to the code. Since several points in M_1 correspond to the the same point in M_2, only the closest point is chosen as the right pair and others are deleted. Consequently, after the first FLANN has done, using **set** to store the point in M_1 only once. Then, using points in M_2 to find the closest points in M_1, the closest corresponding points can be obtained. In this case, the optimization: $E(\mathbf{R}, \mathbf{t}) = \sum_i w_i \left\|(\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i)\right\|^2$ can be regarded as: If the points are the closest points between M_1 and M_2, ω_i is set to 1. If not, ω_i is set to 0.

Since the weight is added to the equation $E(\mathbf{R}, \mathbf{t})$, the derivative of E can be written as:

$$\frac{d(E(R,T))}{dt} = \sum_i 2w_i \left(Rp_i + t - q_i\right) = 0$$

The equation can be transformed to:

$$\frac{d(E(R,T))}{dt} = R\sum_i w_i p_i + nt - \sum_i w_i q_i = 0$$

Knowing that $\overline{p} = \frac{\sum_i w_i p_i}{n}$ and $\overline{q} = \frac{\sum_i w_i q_i}{n}$, get $t = \overline{q} - R\overline{p}$.

Substitute T, the equation can be written as:

$$E(R) = \sum_i \left\| w_i \left(R \left(p_i - \overline{p} \right) - \left(q_i - \overline{q} \right) \right)^2 \right\|$$

Set $\tilde{p}_i = p_i - \overline{p}$ and $\tilde{q}_i = q_i - \overline{q}$, the equation change to:

$$E(R) = \sum_i \left\| w_i \left(R\tilde{p}_i - \tilde{q}_i \right)^2 \right\|$$

To minimize it, the expression can be written as:

$$\min(E(R)) = \max\left(\mathrm{tr}\left(RPQ^T\right)\right) = \max(\mathrm{tr}(RA))$$

where A equals to : $A = \sum_i w_i \left(p_i - \overline{p} \right) \left(q_i - \overline{q} \right)^T$

Finally, $R = VU^T$.

Question 2

In this task, the rotation is done in y-axis. Set the model M_1 to the origin, which uses barycenters to recenter the points. The rotation matrix can be seen as:

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{bmatrix}$$

Since the two models are the same, bad pairs extracting algorithm is not required. Increasing the angles from 10 to 90 degrees, the matched bunny and the mismatched bunny can be seen in fig.3 and fig.4 respectively, where the black model is the original rabbit, the yellow model is the rotated rabbit after iteration and the green model is the rotated rabbit before iteration.

Figure 3: The matched bunny

Figure 4: The mismatched bunny

If the rotation gradually increases, the intersecting area gradually decreases. As what can be seen in fig.3, when the rotation is less than 75 degrees, the yellow model can completely fit the black model. However, when the intersecting area is small enough, the yellow model cannot fit the original model anymore. Consequently, it is important to start the algorithm with two nearly aligned models.

Question 3

The objective of this question is to evaluate the performance of the ICP under the noisy condition. Noise are added on the basis of the second task. Rotating M_1 45 degrees in y axis, the model M_2 can be obtained. Since the noise has zero mean, the noise is Gaussian white noise. The standard deviations is set to 0.001, 0.003, 0.005 and 0.008. Comparing different standard deviations, the results after 100 iterations can be seen in fig.5 and fig.6:

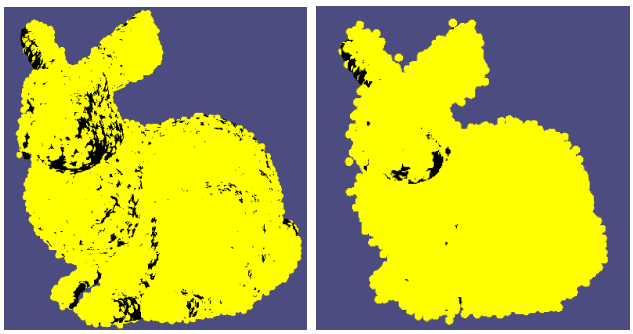

Figure 5: The results with standard deviation 0.001 and 0.003

As what can be seen in the graph, when the standard deviation equals to 0.001 and 0.003, the rabbit can fit well. In the current, errors mainly come from the noise. When the deviation gradually increases to 0.008, the final results have much larger deviation. In this time, much larger errors are obtained. According to the fig.6, errors might mainly come from the matching.

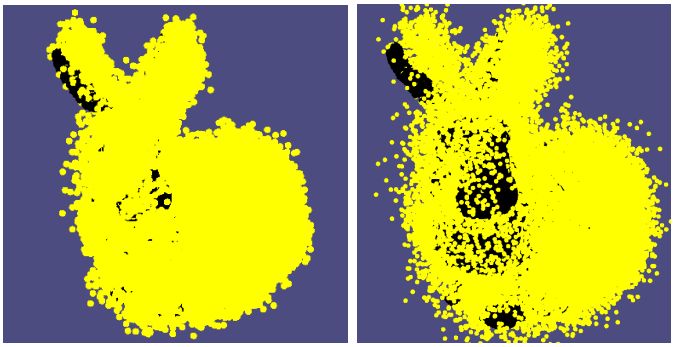

Figure 6: The results with standard deviation 0.005 and 0.008

Question 4

The aim of this question is to use subsampling algorithm to enhance the ICP algorithm. In this stage, the sub-sampled algorithm is based on uniform sub-sampling. The points are extracted every 4 points in both of M_1 and M_2. The results can be seen in fig.7:

When the interval changes from 4 to 8, 16, and 32, the algorithm is much faster but still works well. It seems that changing sampling rate may not have a obvious impact to final results. If the sampling rate keep increasing, it seems that it will make a few influence to the

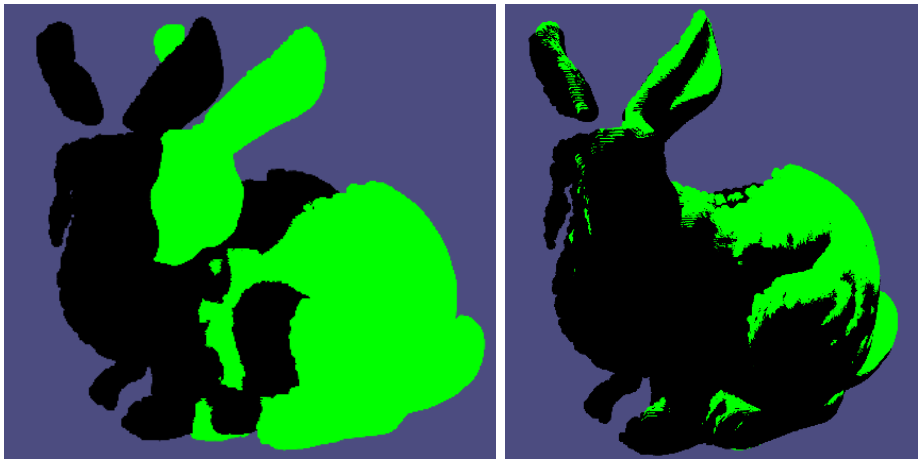

Figure 7: Sub-sampled versions

value of error, which is caused by losing features from the uniform sampling. However, smaller dataset does largely increase the speed of ICP programming.

Question 5

To achieve this task, the five bunnies has been roughly adjusted the rotation first to make the bunny easy to match. Firstly, the model *bun000.ply* is used as the base. Comparing the *bun045* to the *bun000* first, the rotated bunny model *bun045* can be obtained. Combine the base *bun000* and the rotated *bun045* together as a new matrix called tem_1. Using the same algorithm, matching the *bun090* with tem_1, then the bun090 will be rotated to the aligned position. Combing the bun090 with tem_1 as a new matrix called tem_2, the *bun180* can be matched. The following models are matched using the same way.

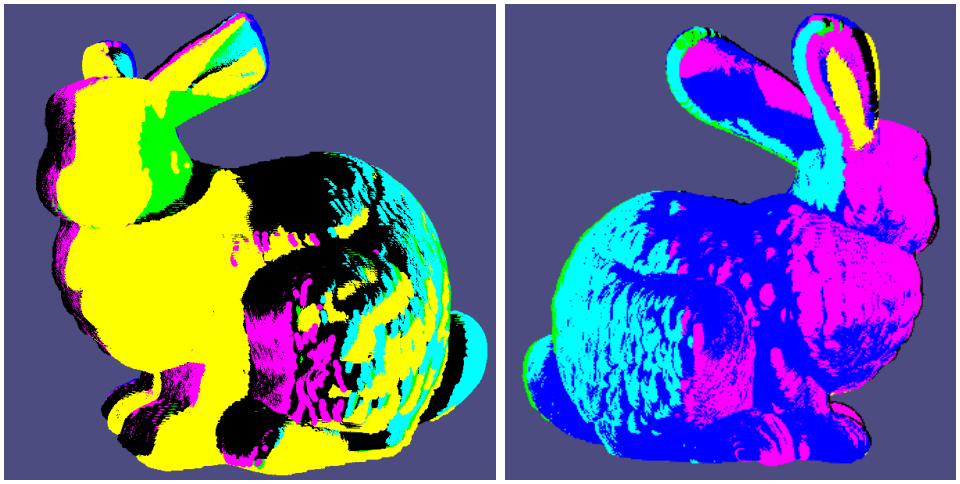

Figure 8: The multiply scans results

The algorithm without bad pairs has been tried firstly. However, it has a bad effect. In this case, the aligned algorithm uses bad pairs, since the different models always have points in different areas, these points need to be abandoned and just keep the matching points. After

combining the whole six models,the results can be seen in fig.8:

Comparing to the algorithm without using tem_1, this algorithm is more accurate. However, the algorithm can be used when rotation angles are given. In this case, the algorithm cannot be commonly used.

Knowing the limitation, an algorithm is developed based on it. The model rotates every 5 degrees, comparing the rotated M_2 with the base M_1 and using the bad pair algorithm above, the different matching points can be obtained. Then, comparing the different number of matching points, the closest rotation angle can be regarded as the rotation which has the largest number of matching points. However, it will largely increase the computational expense and it can just apply to two axes. Trying it to *bun090* and *bun180*, it has the same results comparing fig.8. Consequently, the algorithm works well. Also, using subsampling might be a good choice to reduce the computational expense of the judgement statement.

Question 6

The objective of this question is to use vertex normal to modify ICP algorithm. Consequently, the vertex normal is required to be calculated. The calculation of vertex requires values of face normals. After the face normals are calculated, the neighbor face of each vertex needs to be obtained. The value of vertex normals are the average neighbor values of face normals. However, some vertex does not have neighbors. If a value divide 0, it will obtain NaN, which will influence the calculation of rotation and transformation. Comparing the normals which has been calculated with the library **per_vertex_normals.h**, the results seem to be very close. In this case, the normals seem to have right answers.

The point-to-plane ICP references to the thesis from NUS [3]. To find the rotation and translation, six unknown elements are required, they are: (t_x,t_y,t_z) and (α,β,γ).

In this case, the rotation matrix can be written as:

$$R \approx \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix}$$

The unknown vector x can be written as:

$$\mathbf{x} = \begin{pmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{pmatrix}^{\mathrm{T}}$$

Knowing that the vector b:

$$\mathbf{b} = \begin{pmatrix} n_{1x}d_{1x} + n_{1y}d_{1y} + n_{1z}d_{1z} - n_{1x}s_{1x} - n_{1y}s_{1y} - n_{1z}s_{1z} \\ n_{2x}d_{2x} + n_{2y}d_{2y} + n_{2z}d_{2z} - n_{2x}s_{2x} - n_{2y}s_{2y} - n_{2z}s_{2z} \\ \vdots \\ n_{Nx}d_{Nx} + n_{Ny}d_{Ny} + n_{Nz}d_{Nz} - n_{Nx}s_{Nx} - n_{Ny}S_{Ny} - n_{Nz}s_{Nz} \end{pmatrix}$$

Also, the expression of matrix A :

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & n_{1x} & n_{1y} & n_{1z} \\ a_{21} & a_{22} & a_{23} & n_{2x} & n_{2y} & n_{2z} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & n_{Nx} & n_{Ny} & n_{Nz} \end{pmatrix}$$

Let $\mathbf{A} = \mathbf{U\Sigma V}^{\mathrm{T}}$ be the SVD of A. In this case, the pseudo-inverse of A is defined as the matrix: $\mathbf{A}^{+} = \mathbf{V\Sigma}^{+}\mathbf{U}^{\mathbf{T}}$. Consequently, the least squares solution is: $\mathbf{x} = \mathbf{A}^{+}\mathbf{b}$.

As what has been shown above T = [x_4,x_5,x_6] and $\mathbf{R} = \begin{pmatrix} 1 & -x_1 & x_2 \\ x_1 & 1 & -x_0 \\ -x_2 & x_0 & 1 \end{pmatrix}$.

In this case, the rotation and transformation can be obtained. After around 15 times interactions, the models converge, which is much less than the iterations in the task 1. The result model can be seen in fig.9. As what can be seen in the graph, the two models seems to fit well, where the white lines are normals. Consequently, the result seems to be reasonable.

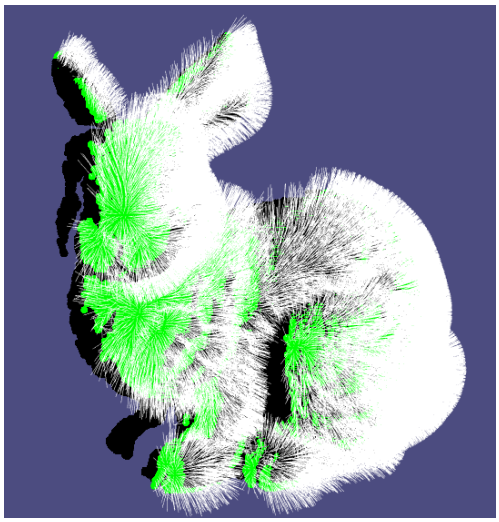

Figure 9: The result model

References

[1] Graphics.stanford.edu. (2019) The stanford 3d scanning repository. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/

[2] Jlblancoc.github.io. (2019) nanoflann: nanoflann c++ api documentation. [Online]. Available: https://jlblancoc.github.io/nanoflann/

[3] K.-L. Low. (2019) Linear least-squares optimization for point-to-plane icp surface registration. [Online]. Available: https://www.comp.nus.edu.sg/~lowkl/publications/lowk_point-to-plane_icp_techrep.pdf