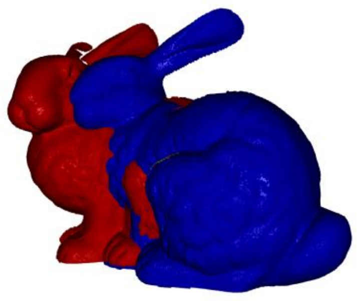
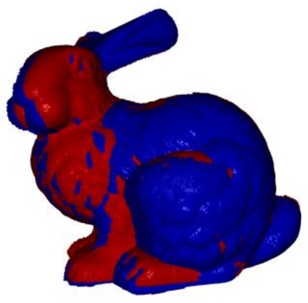
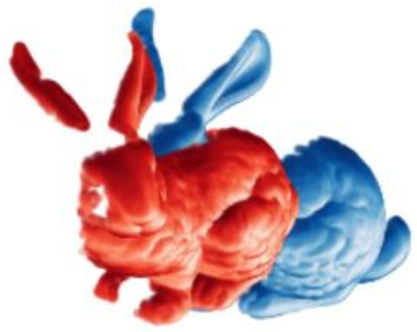
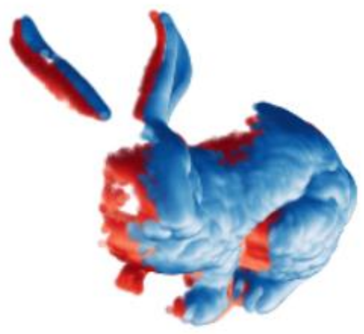
# COMP119 - Assignment 1

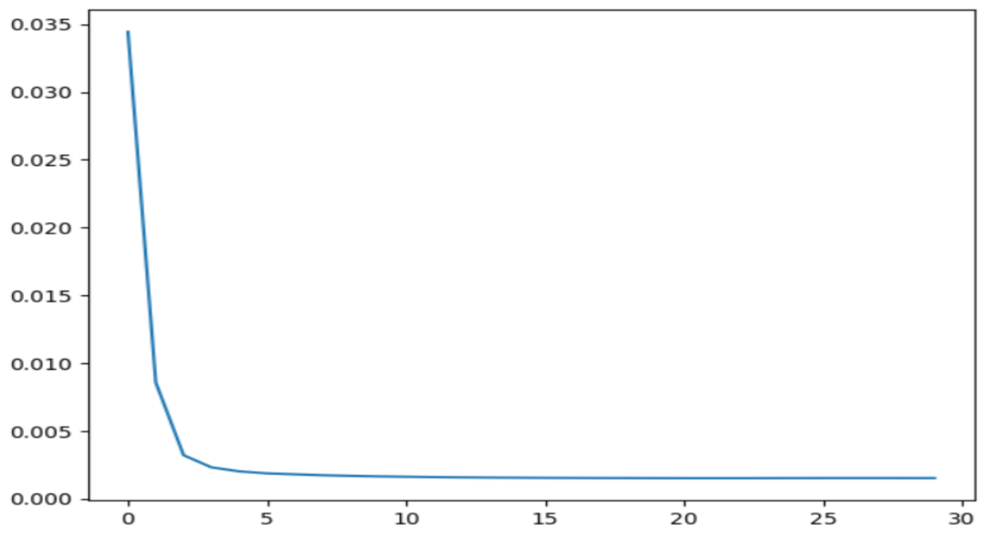
## QUESTION 1:

### a) Implement point-to-point ICP.

We select *bun000_v2.ply* and *bun045_v2.ply* to test our ICP algorithm.
For implementing ICP, I use The least-squares method to solve the g objective function. I use kd-tree to find the nearest neighbor vertices, and for easily I use sklearn library here. As for rejection part of ICP, I set a normal vectors' angle threshold and reject all of the corresponding vertices whose angle residual is more than 20°.

In this part, I show my result by using open3d and matplotlib respectively. The result as following.



The line chart show the convergence process by Distance Mean Error.

For testing my code:

```
# see the result by open3d
# remember to press ‘q’ in keyboard to see the next plot
python part1.py

# see the result by matplotlib
python part1.py -plt
```

**b) solve the optimization $E(\boldsymbol{R},\boldsymbol{t}) = \sum_i w_i \|\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}_i\|^2$ over unknown rotation R and translation t.**

Firstly, we try to get optimized **t**,

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{t}} = \sum_{i=1}^{n} 2w_i(\boldsymbol{R}\boldsymbol{p}_i + \boldsymbol{t} - \boldsymbol{q}_i)$$

$$= 2\mathbf{t}\sum w_i + 2\boldsymbol{R}\sum w_i\boldsymbol{p}_i - 2\sum w_i\boldsymbol{q}_i$$

$$= 0$$

so we get:

$$\mathbf{t} = \frac{\sum w_i\boldsymbol{q}_i - \boldsymbol{R}\sum w_i\boldsymbol{p}_i}{\sum w_i}$$

and thus, we can get **t** by **R**. So we can try to get **R** without considering translation. For doing this, we translate $\boldsymbol{q}_i$and $\boldsymbol{p}_i$ to their centroids coordinates, $\widehat{\boldsymbol{p}}_i = \boldsymbol{p}_i - \overline{\boldsymbol{p}}$ and $\widehat{\boldsymbol{q}}_i = \boldsymbol{q}_i - \overline{\boldsymbol{q}}$. And then the optimization function transform to:

$$\boldsymbol{E}(\boldsymbol{R}) = \sum_i w_i \|\boldsymbol{R}\widehat{\boldsymbol{p}}_i - \widehat{\boldsymbol{q}}_i\|^2 \qquad \text{s.t.}\ \boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}$$

And we have,

$$
\begin{aligned}
\left|\left|\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} - \widehat{\boldsymbol{q}_\iota}\right|\right|^2 &= (\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} - \widehat{\boldsymbol{q}_\iota})^T(\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} - \widehat{\boldsymbol{q}_\iota}) \\
&= \widehat{\boldsymbol{p}_\iota}^T\boldsymbol{R}^T\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} - \widehat{\boldsymbol{q}_\iota}^T\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} - \widehat{\boldsymbol{p}_\iota}^T\boldsymbol{R}^T\widehat{\boldsymbol{q}_\iota} + \widehat{\boldsymbol{q}_\iota}^T\widehat{\boldsymbol{q}_\iota} \\
&= \|\widehat{\boldsymbol{p}_\iota}\|^2 + \|\widehat{\boldsymbol{q}_\iota}\|^2 - 2\widehat{\boldsymbol{q}_\iota}^T\boldsymbol{R}\widehat{\boldsymbol{p}_\iota}
\end{aligned}
$$

So,

$$
\begin{aligned}
\mathbf{R} &= \mathrm{argmin}_{\mathbf{R}}\boldsymbol{E}(\boldsymbol{R}) \\
&= \mathrm{argmin}_{\mathbf{R}}\sum w_i\,(\|\widehat{\boldsymbol{p}_\iota}\|^2 + \|\widehat{\boldsymbol{q}_\iota}\|^2) - 2\sum w_i\widehat{\boldsymbol{q}_\iota}^T\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} \\
&= \mathrm{argmax}_{\mathbf{R}}\sum w_i\widehat{\boldsymbol{q}_\iota}^T\boldsymbol{R}\widehat{\boldsymbol{p}_\iota} \\
&= \mathrm{argmax}_{\mathbf{R}}\,trace(\boldsymbol{Q}^T\boldsymbol{RPW}) \\
&= \mathrm{argmax}_{\mathbf{R}}\,trace(\boldsymbol{RPWQ}^T) \\
&= \mathrm{argmax}_{\mathbf{R}}\,trance(\boldsymbol{RA})
\end{aligned}
$$

where, $\widehat{\boldsymbol{q}_\iota} = \left(\widehat{q_{\iota x}}, \widehat{q_{\iota y}}, \widehat{q_{\iota z}}\right)^T$, $\widehat{\boldsymbol{p}_\iota} = \left(\widehat{p_{\iota z}}, \widehat{p_{\iota y}}, \widehat{p_{\iota z}}\right)^T$ and

$$\boldsymbol{Q} = (\widehat{\boldsymbol{q}_1} \quad \widehat{\boldsymbol{q}_2} \quad \cdots \quad \widehat{q_n}),\ \ \boldsymbol{P} = (\widehat{\boldsymbol{p}_1} \quad \widehat{\boldsymbol{p}_2} \quad \cdots \quad \widehat{\boldsymbol{p}_n})\ \ and\ \boldsymbol{W} = diag(w_1, w_2, \dots, w_n)$$

$$\mathbf{A} = \boldsymbol{PWQ^T}$$

use SVD method to solve it,

$$
\begin{aligned}
trace(\boldsymbol{RA}) &= trace(RU\Sigma V^T) \\
&= trace(\boldsymbol{\Sigma V^T RU})
\end{aligned}
$$

and basing on the property of SVD, when $\boldsymbol{V^TRU} = \boldsymbol{I}$ , $trace(\boldsymbol{RA})$ can reach maximum.

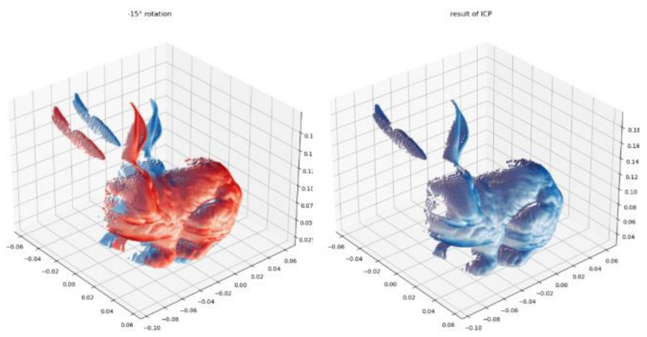
so,

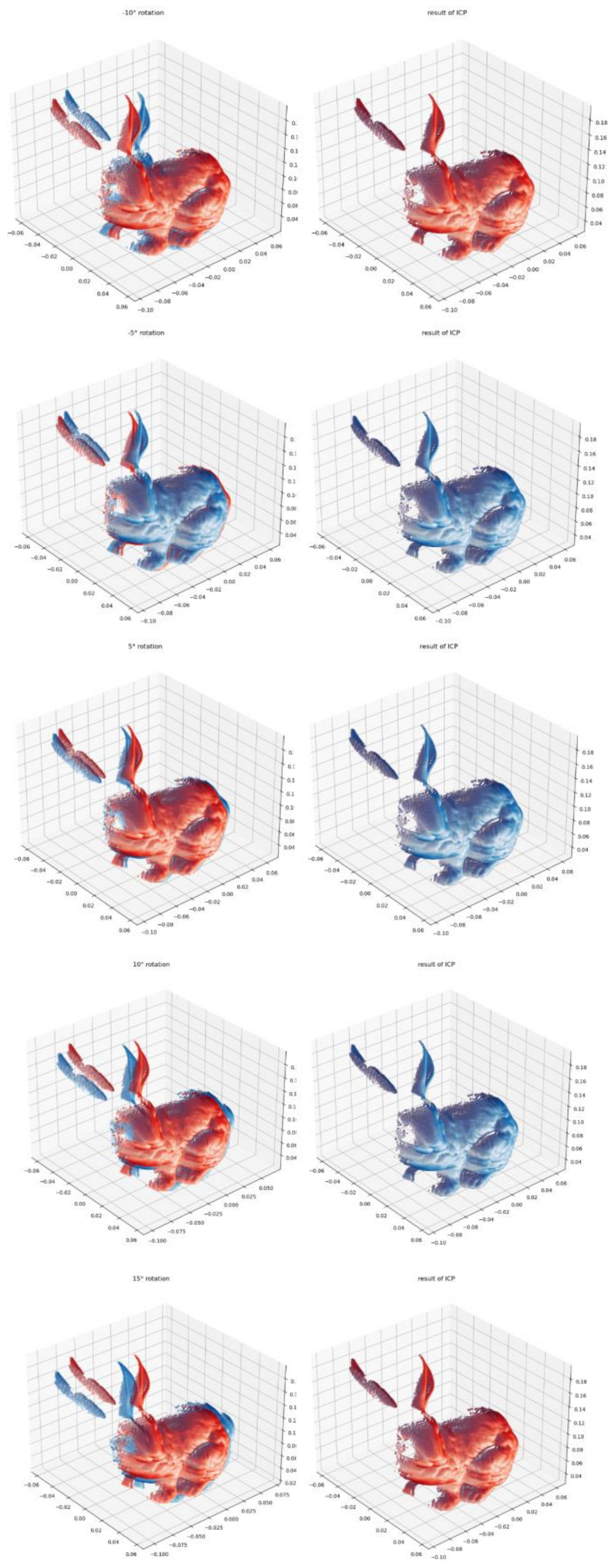
$$\boldsymbol{R} = \boldsymbol{VU^T} \qquad , where\ \boldsymbol{A} = \boldsymbol{PWQ^T}$$

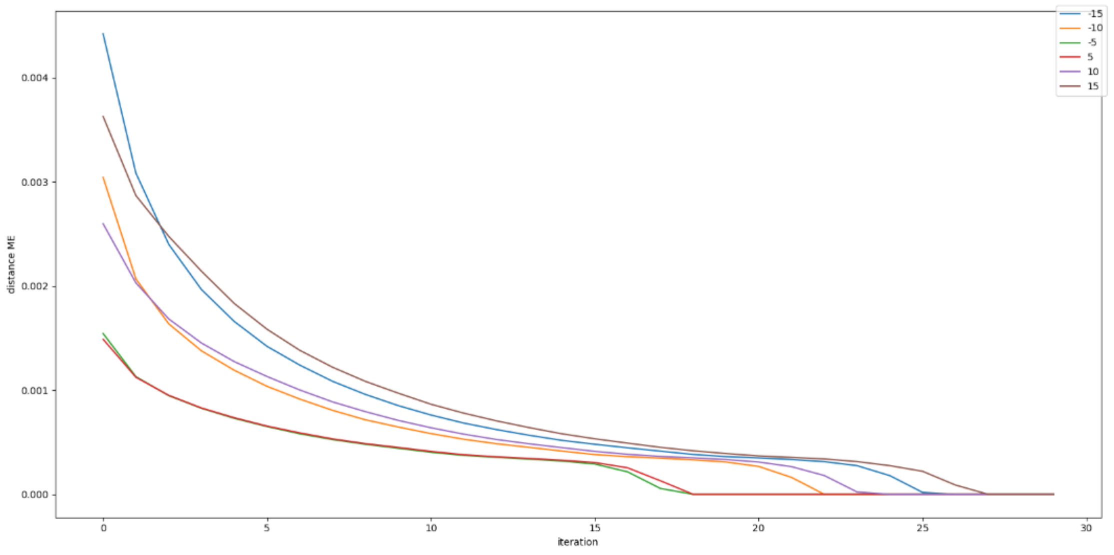
**QUESTION 2:**

**Rotate the object about z-axis over increasing rotation angles (say ±5, ±10, ±15,.. degrees) to simulate the effect of increasing misalignment.**

My result is shown as following:




-10° rotation
result of ICP
-5° rotation
result of ICP
5° rotation
result of ICP
10° rotation
result of ICP
15° rotation
result of ICP



From the line chart, we can see that with the increasing of rotation misalignment, the convergence will be lower, but finally reach the same ME error.

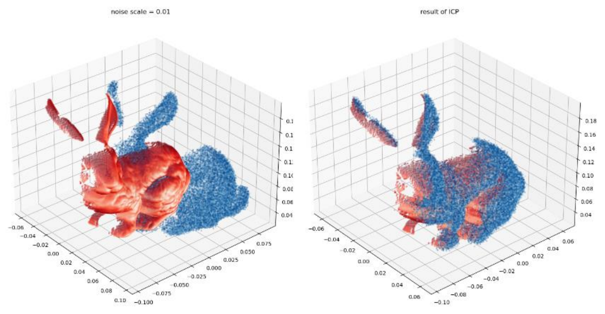
For testing my code:

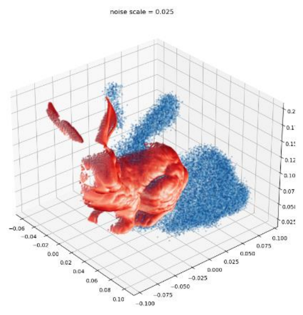
```
python part2.py
```

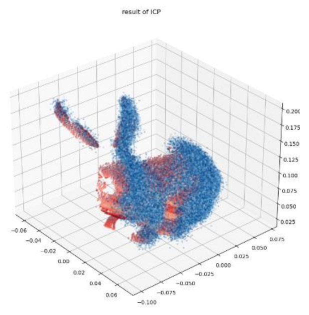
**QEUSTION 3:**

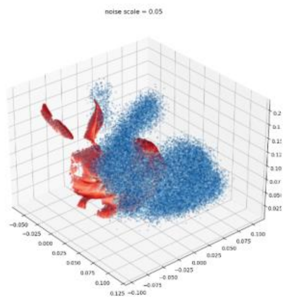
**Evaluate how well ICP performs as you continue to add more noise. As for noise, add zero-mean Gaussian noise – you can simply perturb each vertex of the mesh M2 under this Gaussian model. Adjust the amount of noise based on the bounding box dimensions of M2.**

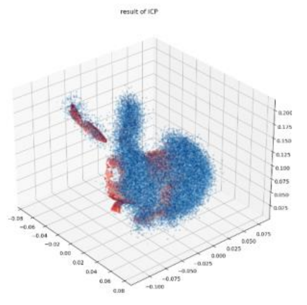
My result is shown as following:

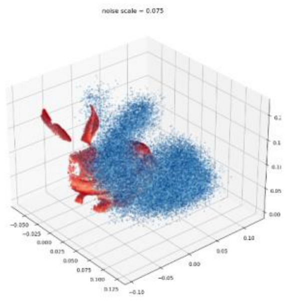


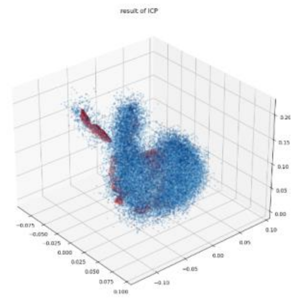

noise scale = 0.025

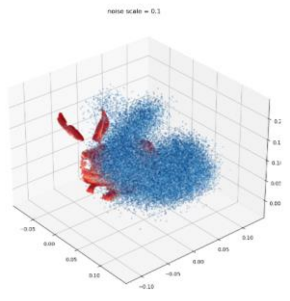

result of ICP

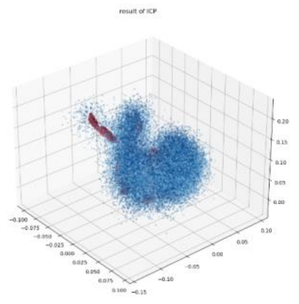

noise scale = 0.05

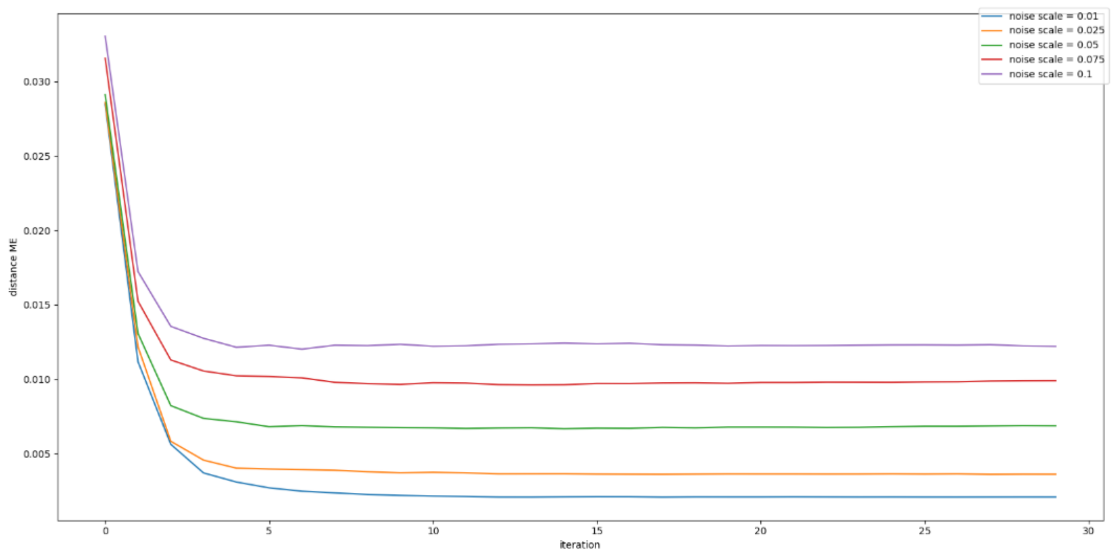

result of ICP


noise scale = 0.075


result of ICP


noise scale = 0.1


result of ICP



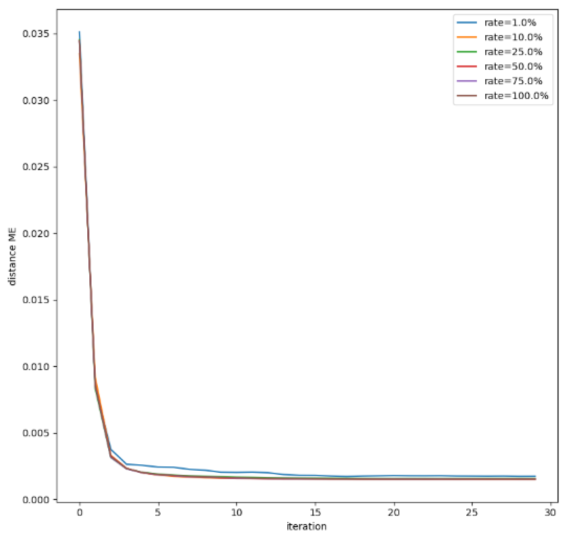
From the line chart, we can see that with the increasing of noisy, it seems that there is no change for convergence rate, but the final MSE error is going to be bigger.

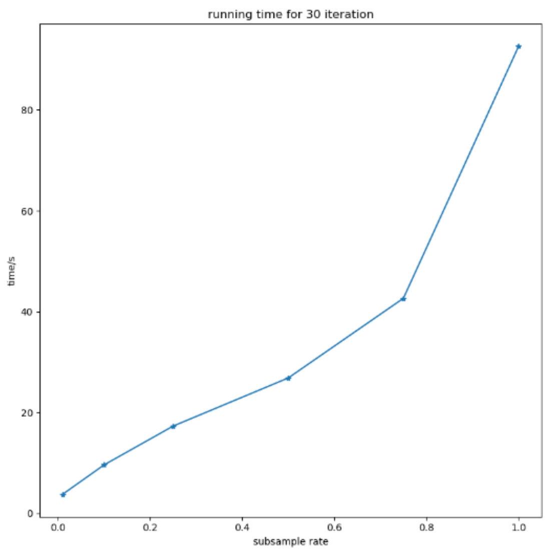
For testing my code:

```
python part3.py
```

**QUESTION 4:**
**Instead of directly aligning M2 to M1, speed up your computation by estimating the aligning transform using subsampled versions of M1 and/or M2 as appropriate. Report accuracy with increasing subsampling rates.**

In this part, I use np.random.uniform to subsample the vertices of M2, and for a higher performance I do not subsample M1. The final result is shown:





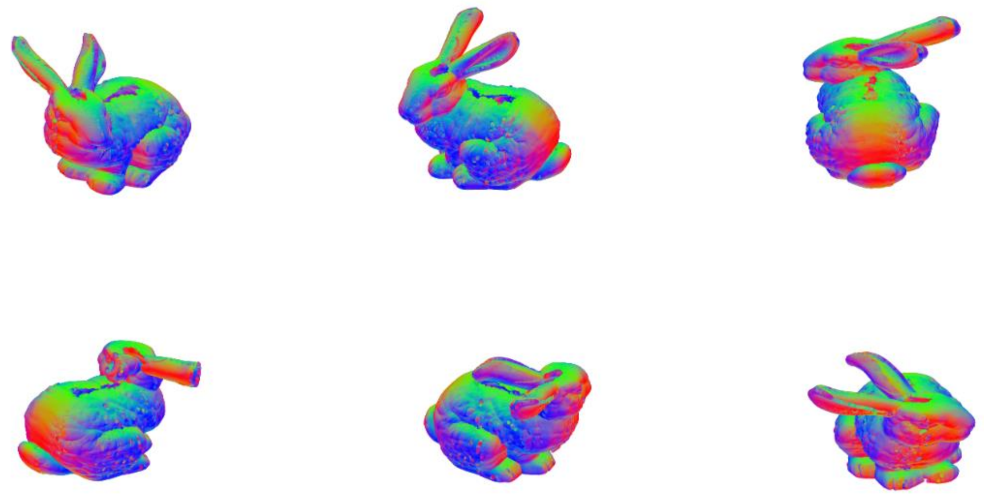
When we change the sample rate from 1% to 100%, there is almost no difference for accuracy defined by distance mean error here. The reason is the vertex number of one scan mode is big, and even the sampling rate reduces to 1%, the number of points used for computing transformation is still sufficient. But for the run time, it will cost more time with the increasing of sample rate.

For testing my code:

```
python part4.py
```

**QUESTION 5:**
**Now given multiple scans M1,M2,...M5 align all of them to a common global coordinate frame. Make sure to think about the best strategy to do this. Describe the method you propose and discuss its pros/cons.**

The result of reconstruction is shown in the picture below.



The step for doing this is following:

a) STEP 1, select scans M1 and M2,
b) STEP 2, align M2 to M1 by using ICP algorithm
c) STEP 2, combine M1 and aligned M2 as new trimesh object, and mark it as a new M1
d) STEP 3, select a new M2, and then go to STEP 2

And also there are some tips for doing it. Firstly, we can know the scanning direction from the file's name, so we can using a rotation matrix by this angle to preprocess M2 for a better aligning. Secondly, we can start reconstructing from *bun270_v2.ply* to *bun315_v2.ply* and then *bun000_v2.ply, bun045_v2.ply, bun090_v2.ply,* and finally *bun180_v2.ply.* Because the difference of scan angle between 090, 180, 270 is much bigger than previous models.

For the pros, the accuracy is better.
And for the cons, the search space will be bigger with the iteration, because M1's vertices number is growing. So the running time will be logger, and waste much more computable resources.

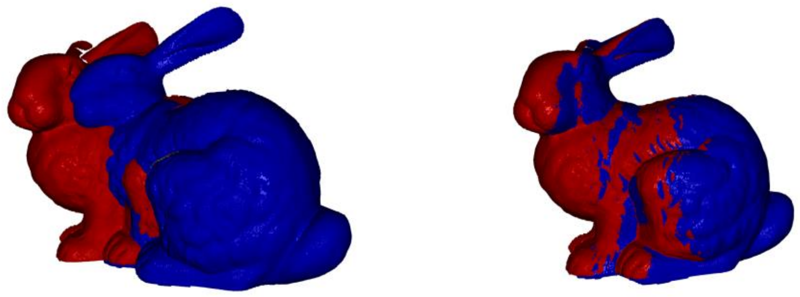
For testing code:

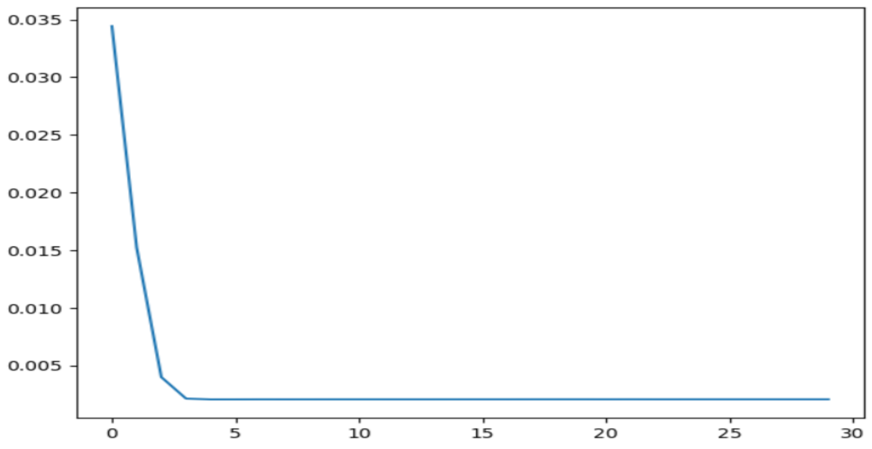
```
python part5.py
```

**QUESTION 6:**

**Use the normal information to improve the ICP performance. and you are attempting to solve minimize the following objective function $\mathbf{E}(\mathbf{R},\mathbf{t}) = \sum_i \left\|(\mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i) \cdot n_i^q\right\|^2$. This method is commonly referred to point-to-plane ICP.**

My reference for this part to finish the algorithm is (https://www.comp.nus.edu.sg/~lowkl/publications/lowk_point-to-plane_icp_techrep.pdf)





The line chart show the convergence process by Distance Mean Error.

For testing code:

```
# see the result by open3d
python part6.py

# see the result by matplotlib
python part6.py -plt
```

**APPENDIX:**

The library I used is shown in this list:

a) open3d==0.12.0

b) scikit-learn @ file:///D:/bld/scikit-learn_1611079982022/work

c) transformations==2020.1.1

d) trimesh==3.9.1

e) numpy @ file:///C:/ci/numpy_and_numpy_base_1603480701039/work

f) matplotlib @ file:///C:/ci/matplotlib-base_1603357981283/work

g) python==3.6.12

h) argparse==1.4.0