COMPM080 - Report

Iterative Closest Point Alignment Algorithm

Fen Jin

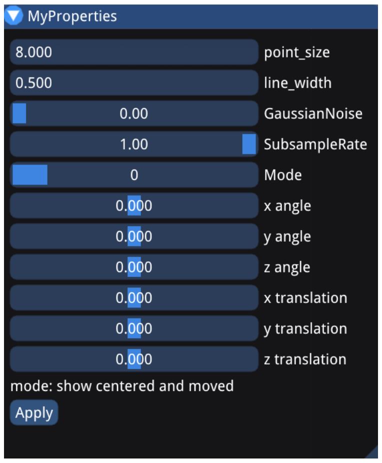
February 2019

1 Introduction

The goal of the coursework is to apply the Iterative Closest Point (ICP) algorithm to the Standford Bunny[1], which is composed with a total of 10 scans in ply format. We are going to read the ply files and work on the point clouds.

On the menu in Fig.1, it is possible to select the options to apply on the source mesh (the one to be moved, which we will call M_2). Among the actions we can select the modes : just show, add noise, subsample, reset, save cloud points on file, ICP point to point and point to plane. It is also possible to choose the amount of noise or percentage of sub-samples to apply. To roughly align the two meshes, we can slide manually the rotation and translation sliders. Once selected the desired options, the button 'apply' will update the changes.

The changes are applied on the existing current M_2, which means that clicking 'Apply' twice will apply two times the options selected.


MyProperties

8.000 point_size
0.500 line_width
0.00 GaussianNoise
1.00 SubsampleRate
0 Mode
0.000 x angle
0.000 y angle
0.000 z angle
0.000 x translation
0.000 y translation
0.000 z translation
mode: show centered and moved

Figure 1: Menu of options.

2 Task 1

The first task is to apply the ICP algorithm to two scans: bun000 and bun045, we will call them M_1 and M_2 respectively. Each row corresponds to a point in 3D. Once loaded, the two cloud points look like Fig.2a.

[1]`http://graphics.stanford.edu/data/3Dscanrep/`

To consider M_2 aligned to M_1, we want iteratively move M_2 with a rigid transformation to minimize, until convergence, the objective function

$$E(R,t) = \sum_{i=1}^{n} \|p_i - Rq_i - t\|^2 \tag{1}$$

R is the rotation matrix, t is the translation vector, q_i and p_i are points in M_2 and M_1 respectively.

To align M_2 to M_1, we first, use ANN to find the nearest neighbor for each $q_i \in M_2$ in M_1. So, we call p_i the corresponding nearest neighbor. We then center all the q_i and p_i in their own barycenters,

$$\tilde{q_i} = q_i - \overline{q}, \qquad \overline{q} = \frac{\sum_{i=1}^{n} q_i}{n} \tag{2a}$$

$$\tilde{p_i} = p_i - \overline{p}, \qquad \overline{p} = \frac{\sum_{i=1}^{n} p_i}{n} \tag{2b}$$

We then construct the 3x3 matrix A:

$$A = \sum_{i=1}^{n} \tilde{q_i}\tilde{p_i}^T \tag{3}$$

By taking the Single Value Decomposition of the matrix A ($A = U\Sigma V^T$), the close form solutions of Eq.1 (details offered later when considering confidence score) are

$$R = VU^T \tag{4a}$$

$$t = \overline{p} - R\overline{q} \tag{4b}$$

We then apply the rigid transformation to all the points in M_2. We can put every point in homogeneous coordinates, by adding a forth column of ones to all the points and apply the transformation matrix H:

$$H = \begin{pmatrix} R_{11} & R_{12} & R_{13} & t_1 \\ R_{21} & R_{22} & R_{23} & t_2 \\ R_{31} & R_{32} & R_{33} & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

After updating the points in M_2, we start the new iteration until convergence.

The convergence conditions used are: mean error compared to a threshold, difference of mean error between iterations and max number of iterations.

The result of the alignment of the meshes bun000 and bun045 can be seen in Fig.2b.

As rejection of bad pairs, I chose to check the normals of the corresponding points. I compared the cosine between the two normals (given by the dot product) with a threshold value; if the cosine is close to 1, it means that the normals have roughly the same direction and the correspondence is good. The subset of points that satisfy this condition is then used to calculate the matrix A, to find the rigid transformation, which is applied still to the whole mesh. This process should make the algorithm more robust. However, the rejection part has been tested and it does not seem to work very well, in particular it slows down the process a lot because it has to calculate the normals and also check the angle of every pair of correspondence. The algorithm without the rejection performs quite well already, for this reason I did not include it in the algorithm, but the implementation can be seen in the code.

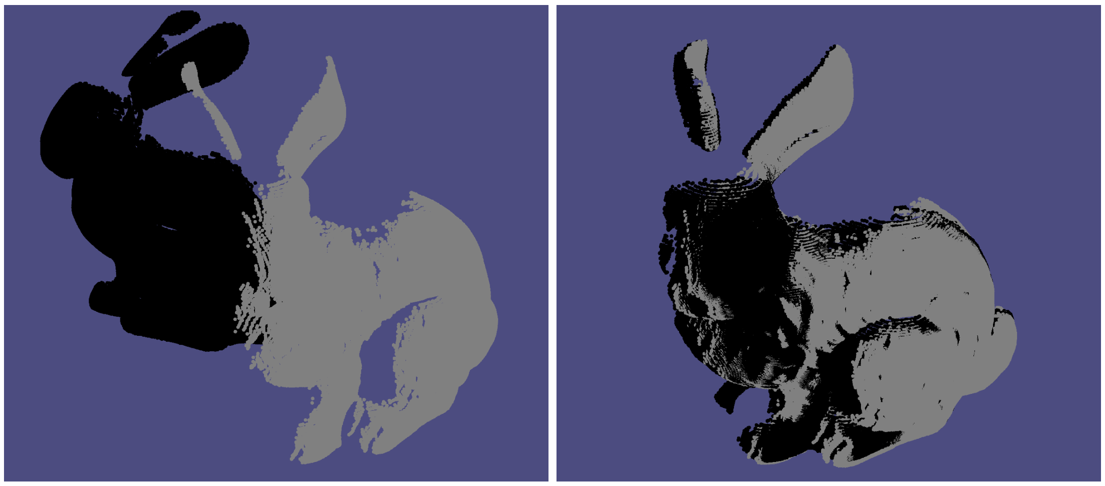

(a) Before ICP (b) After ICP

Figure 2: Cloud points of bun000 (black points) and bun045 (grey points).

Confidence score If we consider a confidence score for each measurement, the new objective function to minimize is

$$E(R,t) = \sum_{i=1}^{n} w_i \|Rp_i + t - q_i\|^2 \tag{5}$$

where we just added a weight $w_i \in [0,1]$ for every correspondence.

We will now show how to get R and t. First, we take the derivative with respect to t and equal it to zero.

$$\frac{\partial E(R,t)}{\partial t} = 2\sum_{i=1}^{n} w_i \|Rp_i + t - q_i\| = 0 \tag{6}$$

$$\sum_{i=1}^{n} w_i R p_i + \sum_{i=1}^{n} w_i t - \sum_{i=1}^{n} w_i q_i = 0$$

$$R\frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i} + t - \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i} = 0$$

$$t = -R\frac{\sum_{i=1}^{n} w_i p_i}{\sum_{i=1}^{n} w_i} + \frac{\sum_{i=1}^{n} w_i q_i}{\sum_{i=1}^{n} w_i}$$

$$t = -R\overline{p} + \overline{q} \tag{7}$$

Now we find R, by substituting t from Eq.7:

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

$$= \sum_{i=1}^{n} w_i \|Rp_i + (-R\overline{p} + \overline{q}) - q_i\|^2 =$$

$$= \sum_{i=1}^{n} w_i \|R(p_i - \overline{p}) - (q_i - \overline{q})\|^2 =$$

$$= \sum_{i=1}^{n} w_i \|R\tilde{p_i} - \tilde{q_i}\|^2 =$$

$$= \sum_{i=1}^{n} w_i \|R\tilde{p_i}\|^2 - \sum_{i=1}^{n} w_i R\tilde{p_i} \cdot \tilde{q_i} + \sum_{i=1}^{n} w_i \|\tilde{q_i}\|^2 =$$
$$= \sum_{i=1}^{n} w_i \|\tilde{p_i}\|^2 - \sum_{i=1}^{n} w_i R\tilde{p_i} \cdot \tilde{q_i} + \sum_{i=1}^{n} w_i \|\tilde{q_i}\|^2$$

where $\|R\tilde{p_i}\| = \|\tilde{p_i}\|$, since rotation do not change length.

Now the objective function depends only on R, and to minimize it we just need to consider the term in the middle:

$$min_R(E(R)) = max_R(\sum_{i=1}^{n} w_i R\tilde{p_i} \cdot \tilde{q_i}) = \tag{8}$$

$$= max_R(\sum_{i=1}^{n} w_i \tilde{q_i}^T R\tilde{p_i}) =$$

$$= max_R(trace \left[\begin{pmatrix} w_1\tilde{q_1} \\ w_2\tilde{q_2} \\ ... \\ w_n\tilde{q_n} \end{pmatrix}_{N\times 3} \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}_{3\times 3} \begin{pmatrix} \tilde{p_1} & \tilde{p_2} & ... & \tilde{p_n} \end{pmatrix}_{3\times N} \right]) =$$

$$= max_R(trace \left[\tilde{Q}^T R\tilde{P}\right]) =$$
$$= max_R(trace \left[R(\tilde{P}\tilde{Q}^T)\right]) =$$
$$= max_R(trace \left[R(U\Sigma V^T)\right]) =$$
$$= max_R(trace \left[\Sigma V^T RU\right])$$

The last steps we applied the SVD decomposition from $\tilde{P}\tilde{Q}^T = U\Sigma V^T$. Indicating $M = V^T RU$, we know that M is orthonormal by construction. So,

$$M^T M = I \Rightarrow m_i^T m_i = 1 \Rightarrow \sum_{j=1}^{3} m_{ij}^2 = 1 \Rightarrow |m_{ij}| \leq 1 \tag{9}$$

$$max_R(trace(\left[\Sigma M\right])) =$$
$$= max_R(trace \left[\begin{pmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{pmatrix} \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \right]) =$$
$$= max_R(\sum_{i=1}^{3} \sigma_i m_{ii})$$

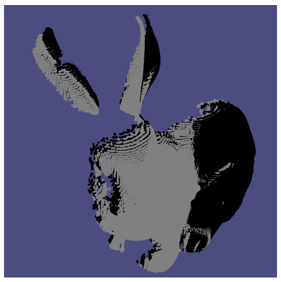
Remembering Eq.9, the values needed to maximize the last step are $m_{ii} = 1$. So, the matrix M must be equal to identity matrix. And since $V^T V = I$, $U^T U = I$:

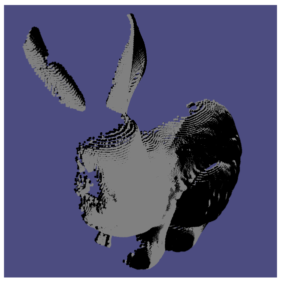
$$M = I \Rightarrow V^T RU = I$$

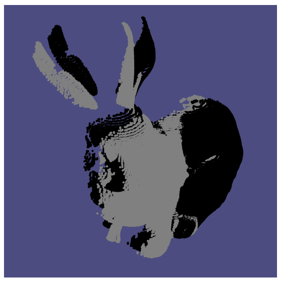
$$R = VU^T \tag{10}$$

3 Task 2

Here we try to align the same mesh (bun000) rotated by some different angles: +5, +15, -20, -40 degrees. We evaluate alignment by the number of iterations till convergence. In all 4 cases, they reach convergence in 1, 6, 9 and 17 iterations, see Fig.3. Of course this demonstrate, the more


(a) +5 degree

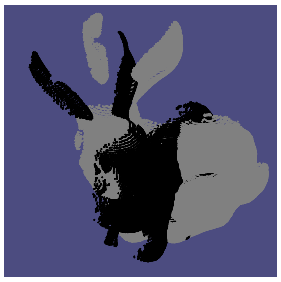

(b) +5 degree, after ICP


(c) +15 degree

(d) +15 degree, after ICP

(e) -20 degree, after ICP

(f) -20 degree, after ICP


(g) -40 degree

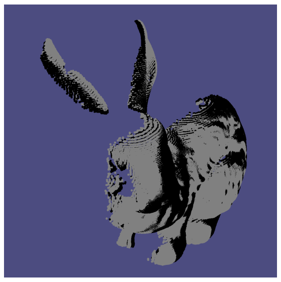

(h) -40 degree, after ICP

Figure 3: Cloud points of bun000 (black points) and bun000 rotated by x degree (grey points). 3b, 3d, 3f and 3h reach convergence with threshold equal to 1e-6, in 1, 6, 9, 17 iterations respectively.

initially aligned the less iterations needed till convergence.

When applying a rotation greater than 75 degrees (or less than -75 degrees) the ICP algorithm does not align very well. This demonstrate that a good initial alignment is needed to have a good alignment performance of the ICP method.

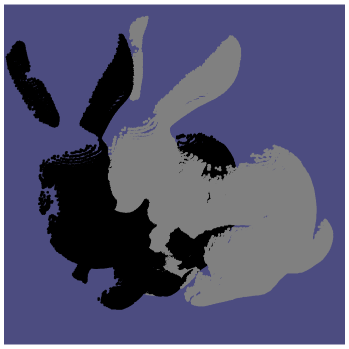
4 Task 3

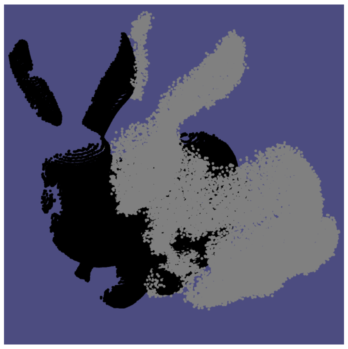
In this task, we will perturb M_2 by adding zero-mean Gaussian noise to each of its points. The standard deviation along each axis dimension will be proportional to the corresponding dimension of the bounding box. This is calculated by taking the maximum and minimum value of M_2 along x,y and z, and subtracting the max with the min. The standard deviation is for the Gaussian noise is then x% of this difference (which can be selected on the menu). In Fig.4, we see the original poses, then the effect of the Gaussian noise on M_2 with 1%, 2% and 5% of perturbation with respect to the bounding box dimensions, and on the right the resulting alignment applying ICP algorithm.

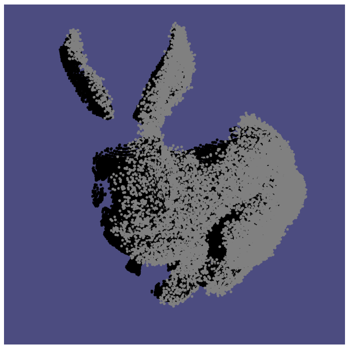
We can notice that, the last case (5% perturbation) already loses a lot of details of the scanned bunny, nevertheless the alignment seems still correct. However, none of the 3 cases would achieve convergence with error threshold 1e-6 in less than 50 iterations, but they achieve pretty good alignment only after a few iterations, in fact the difference of the previous and current error for iterations is becomes small very quickly.

5 Task 4

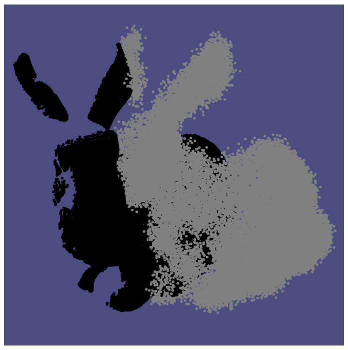
In this task, we will subsample M_2 taking x% of its points (selectable on the menu). We will sample uniformly from M_2. In Fig.5, we see the sampled points at the center and on the right the

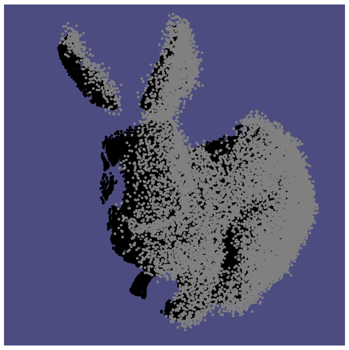

(a) Original position of bun000 and bun045


(b) Bun045 with std equal 1% of the bb

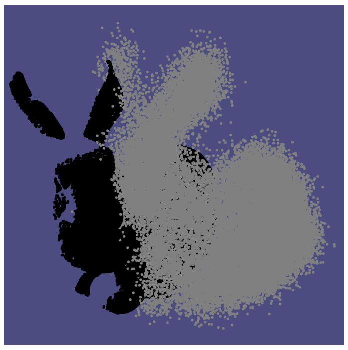

(c) Bun045 with std equal 1% of the bb, after ICP

(d) Original position of bun000 and bun045


(e) Bun045 with std equal 2% of the bb


(f) Bun045 with std equal 2% of the bb, after ICP

(g) Original position of bun000 and bun045


(h) Bun045 with std equal 5% of the bb

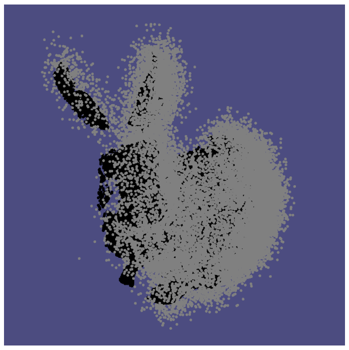

(i) Bun045 with std equal 5% of the bb, after ICP

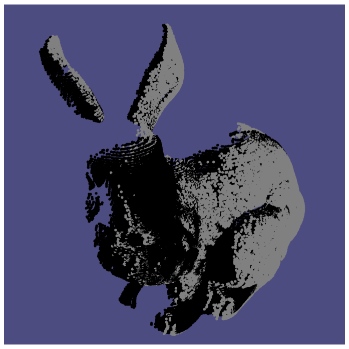
Figure 4: Cloud points of bun000 (black points) and bun045 (grey points). We progressively add zero-mean Gaussian noise with standard deviation (std) proportional to the bounding box (bb) dimensions of bun045.

aligned meshes.

We notice that they all converge, although the number of points is much reduced and the details are lost. The iterations needed for a nice alignment are not many, but the error threshold of 1e-6 might be too restrict. In the examples of Fig.5, they all achieve good result in less than 10 iterations.

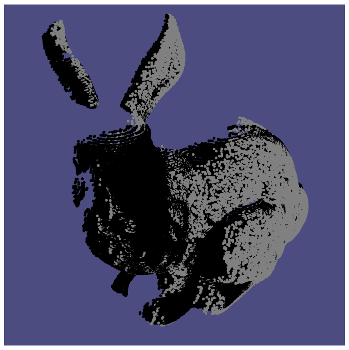
(a) Original position

(b) 80% of its points

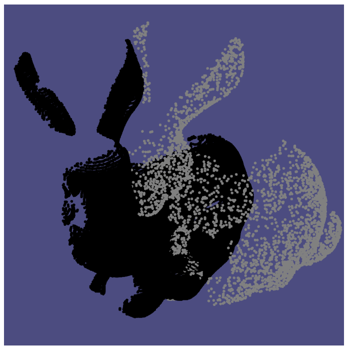

(c) 80% of its points, after ICP

(d) Original position

(e) 50% of its points


(f) 50% of its points, after ICP

(g) Original position


(h) 10% of its points

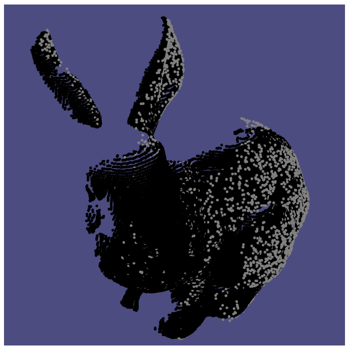

(i) 10% of its points, after ICP

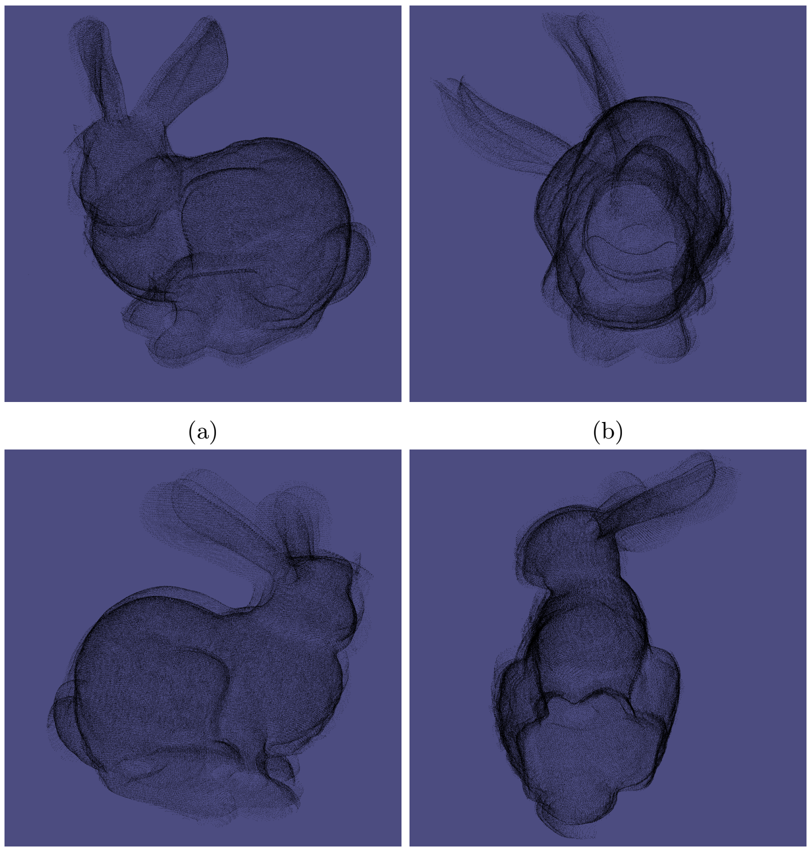
Figure 5: Cloud points of bun000 (black points) and bun045 (grey points). We progressively add zero-mean Gaussian noise with standard deviation (std) proportional to the bounding box (bb) dimensions of bun045.

6 Task 5

To align all the scans of the Standford Bunny, we need to progressively align two scans at the time. It is preferred to align two scans that are close to each other first (like 'bun000' and 'bun045'), i.e. they have most of the point in correct correspondence instead of meshes that are 'opposite' (like 'bun000' and 'bun180'). This is because without a good correspondence of points, a good alignment is hard to obtain. Then, our approach is to start from the first scan (bun000) and align the ones next to it ('bun045' and 'bun315'). Once we align 2 meshes, save the resulting mesh as a single one and load it for the next alignment. We progressively complete the bunny by aligning 'top3', 'bun270', 'bun090', 'top2', 'bun180', 'ear_back' and at last 'chin' because it is the only one having points of the bottom of the bunny.

In Fig.6, we can see the result of the alignments. It is noticeable that the approximate shape is correct but many parts of the bunny are not perfectly aligned, for example the ears. This is probably because, when looking for the corresponding nearest neighbor point, the 'bad' pairs were not discarded. This problem might be solved using the confidence score, described in Task 1.

Even though when aligning two meshes it is visible what the correct alignment is, the current implementation requires a good manual initial alignment for each new scan.



(a) (b) (c) (d)

Figure 6: Multiple views of the complete alignment of the 10 scans of the Stanford Bunny.

It is important to note that, if the scans are very different, the sequential alignment would probably fail, due to absence of good correspondence.

7 Task 6

This task requires to implement the point to plane version of the ICP algorithm. To do this, I followed the procedure in [1]. To find the rigid transformation, i.e. the 6 parameters for movements in 3D space (3 for rotation and 3 for translation), we have to minimize the objective function:

$$E(R,t) = \sum_{i=1}^{n} \|(p_i - Rq_i - t) \cdot n_i^{p_i}\|^2 \tag{11}$$

where $n_i^{p_i}$ is the i-th normal vector of the p_i point.

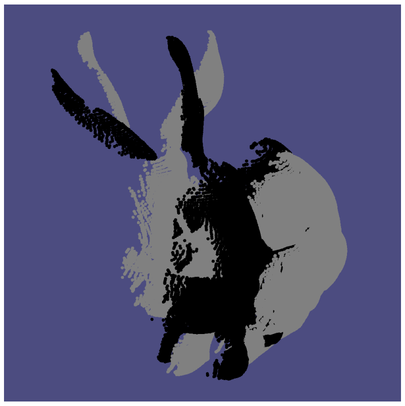
As explained in [1], the problem can be formulated as least square optimization if the rotations are very small, such that we can apply a linear approximation. The problem is reduced to solve a linear system of type $Ax = b$, where x is a 6×1 vector that contains the 6 parameters we need for the rigid transformation. In the implementation we build the matrix A and the vector b, then compute the pseudo-inverse of A (using SVD decomposition) because A has dimensions $N \times 6$. So given:

$$Ax = b$$
$$x = A^{\dagger}b$$

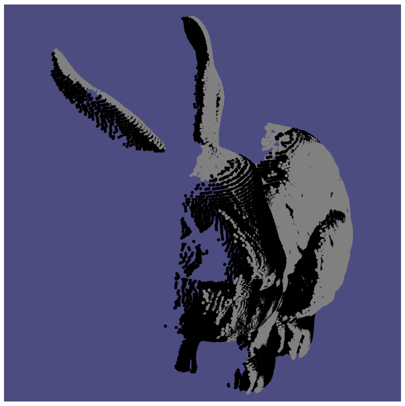
we then construct the rigid transformation matrix. The rest of the procedure is similar to the point to point version.

The point to plane version is much slower than to point to point, because it's not vectorized and the SVD has to solve a much bigger matrix. Unfortunately, it does not converge as quickly as expected; it also delivers worse alignment than point to point version.

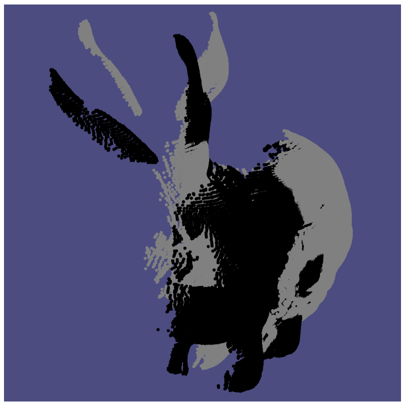
Fig.7 shows some results of the ICP point to point and point to plane versions, under the same conditions. the point to point version converges in 37 iterations, while point to plane does not converge even in 50 iterations (Fig.7c). It is probably caused because the point to plane need a better initial alignment, since the approximation assumes very small movements. In fact, with a better initial alignment the ICP point to plane give a good performance.



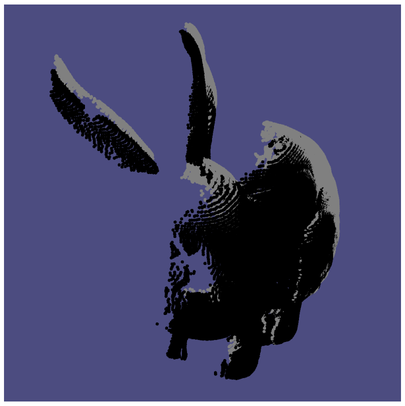
(a) Original position



(b) Point to point alignment



(c) Point to plane before good alignment



(d) Point to plane after better alignment

Figure 7: ICP point to point and point to plane.

References

[1] Kok-Lim Low. Linear least-squares optimization for point-to-plane icp surface registration. *Chapel Hill, University of North Carolina*, 4(10), 2004.