Answers to COMP0119 (Course Work 1)

Chadjiminas Chrysostomos **Student Id No.: 19109700** :

1 Tasks attempted

Task 1

ICP algorithm implemented and derivation refined using weights

Task 2

For a fixed source bunny mesh added pertubations on rotation along an axis. For each pertubation run icp to match a fixed destination mesh and compared the results.

Task 3

For a fixed source bunny mesh added gaussian noise pertubations on each vertex. For each noisy mesh run icp to match a fixed destination mesh and compared the results.

Task 4

Added the subsampling part to my ICP algorithm.

Task 5

Implemented an algorithm that makes some assumptions on distance between meshes.

Task 6

Implemented point to plane method based on https://www.comp.nus.edu.sg/ lowkl/publications/lowk_poin to-plane_icp_techrep.pdf.

Task 7

Implemented a demo.

2 Results and conclusions

Task 1

Derivation:

$$E(R,T) = \sum_i w_i ||Rp_i + t - q_i||^2 \textbf{ s.t } RR^T = I \textbf{ and } \det R = 0 \tag{1}$$

We must minimise so we calculate the gradients:

$$\frac{\partial E}{\partial t} = 0 \tag{2}$$

$$\frac{\partial E}{\partial R} = 0 \tag{3}$$

So for 2:

$$\frac{\partial E(R,t)}{\partial t} = 2\sum_i w_i(Rp_i + t - q_i) = 0$$

Equivalent to

$$\sum_i w_i(Rp_i + t - q_i) = 0$$

$$\sum_i w_i Rp_i + \sum_i t - \sum_i wiq_i = 0$$

We define weighed mean as:

$$\bar{p} = \frac{\sum_i w_i p_i}{\sum_i w_i}$$
$$\bar{q} = \frac{\sum_i w_i q_i}{\sum_i w_i}$$

And we rewrite the equation as

$$R\bar{p} + t - \bar{q} = 0$$

By changing the order we have

$$t = \bar{q} - R\bar{p} = 0 \tag{4}$$

Using 4 on 1

$$E(R,t) = \sum_i w_i ||Rp_i + \bar{q} - R\bar{p} - q_i||^2$$

$$E(R,t) = \sum_i w_i ||R(p_i - \bar{p}) - (q_i - \bar{q})||^2$$

We define:

$$\tilde{p_i} = \sum_i p_i - \bar{p}$$

$$\tilde{q_i} = \sum_i q_i - \bar{q}$$

And so the energy becomes:

$$E(R) = \sum_i w_i ||R\tilde{p_i} - \tilde{q_i}||^2$$

$$E(R) = \sum_i (w_i (R\tilde{p_i} - \tilde{q_i})^T (R\tilde{p_i} - \tilde{q_i}))$$

$$\begin{aligned} E(R) = \sum_i w_i (\sum_i & \tilde{p_i}^T R^T R \tilde{R} \tilde{p_i} \\ & - \sum_i \tilde{q_i}^T R\tilde{p_i} \\ & - \sum_i (R\tilde{p_i})^T * \tilde{q_i} \\ & + \sum_i \tilde{q_i}\tilde{q_i}) \end{aligned}$$

Since $RR^T = I$ and $(R\tilde{p_i})^T * \tilde{q_i} = \tilde{q_i}^T R\tilde{p_i}$

$$E(R) = \sum_i w_i (\sum \tilde{p_i}^T \tilde{p_i} + \sum \tilde{q_i}^T \tilde{q_i}) - 2 \sum_i w_i \tilde{q_i}^T R\tilde{p_i}$$

Because the first part is not dependent on R and the second part is negative then minimising this function is equal to maximising:

$$\arg\max_R(\sum_i \tilde{q_i}^T w_i R \tilde{p_i}) \tag{5}$$

So this reduces to:

$$\sum_i \tilde{q_i}^T w_i R \tilde{p_i} = trace(Q^T W R P)$$

Where

$$Q^T = \begin{bmatrix} q_1^T \\ q_2^T \\ \vdots \\ q_n^T \end{bmatrix}$$

$$W = \begin{bmatrix} w_1 & 0 & 0 & \dots & 0 \\ 0 & w_2 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \dots & w_n \end{bmatrix}$$

$$P = \begin{bmatrix} p_1^T & p_2^T \dots p_n^T \end{bmatrix}$$

Since

$$trace(AB) = trace(BA) \tag{6}$$

$$trace(Q^T W R P) = trace(R P Q^T W)$$

Using SVD we can decompose PQ^TW

$$PQ^TW = U\Sigma V^T \tag{7}$$

And so we plug in in the result and use the trace property 6

$$\begin{aligned} trace(Q^T W R P) &= trace(R U \Sigma V^T) \\ &= trace(\Sigma V^T R U) \end{aligned}$$

Lets define

$$M = V^T R U = \begin{bmatrix} | & | & | \\ m_1 & m_2 & m_3 \\ | & | & | \end{bmatrix}$$

Since $M^T M = I$ (because U and V are orthonormal)

$$1 = m_i^T m = \sum_i^3 = m_{ij}^2$$

Therefore

$$|m_{ij}| \leq 1 \tag{8}$$

The energy function maximisation problem is reduced to (using 8

$$\arg\max_R trace(\Sigma M) = \sum_i \sigma_i m_{ii} \leq \sum_i \sigma_i$$

This function can be maximised if $m_{ii} = 1$ for i=[1,3].

And so we can say that

$$\begin{aligned} M = I &=> V^T R U = I \\ &=> V V^T R U U^T = V U^T \\ &=> R = V U^T \end{aligned}$$

And thus we reach to the derivation (using 4):

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{bmatrix} U^T, t = \tilde{p} - R\tilde{q} \tag{9}$$

Where

$$\tilde{p} = \frac{\sum w_i p_i}{\sum_i w_i}$$

$$\tilde{q} = \frac{\sum w_i q_i}{\sum_i w_i}$$

Task 2

Task 2 icp was run with the same destination bunny (bun000_v2) and the same source bunny (bun045_v2) for all perturbations. The destination bunny mesh was each time transformed by a pure rotation transform so that it changes by an angle around one of the global axis.

For each axis, we transformed the destination mesh by the angles

$$\{\theta_i = c_i * \frac{\pi}{16}, c_i \in [1..16]\}$$

We perform the icp algorithm for a maximum of 40 iterations or until error stops improving.

From figures 1, 2, and 3 we can see that usually algorithm breaks at a certain point when the angle is too large. This happens because we have a very wrong correspondence since points that are very close together are not corresponding points but are close because of the angle.

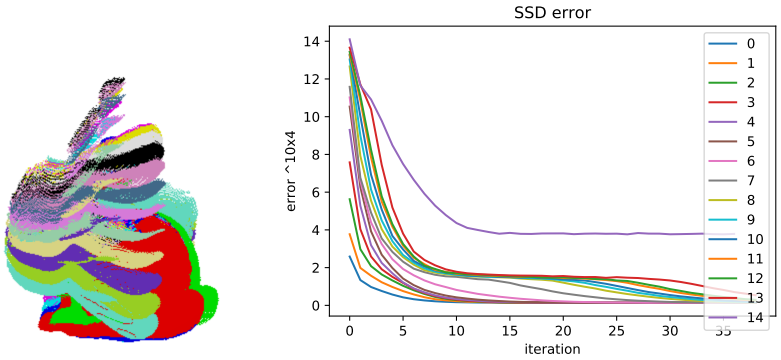

Figure 1: "Perturbations around axis x"

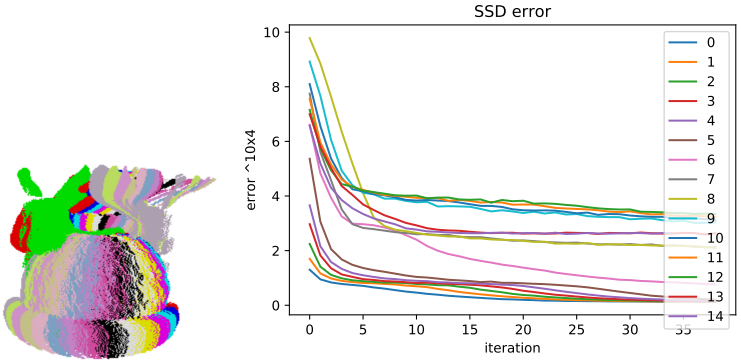

Figure 2: "Perturbations around axis y. Green bunny is the destination mesh."

Task 3

Similarly to task 2 we have a fixed destination mesh and a source mesh we perturb, this time by applying noise to each vertex.

We apply a zero mean Gaussian noise to each vertex around it's initial position for increasing amount of noise. In particular, we use noise values that are factors of the scale

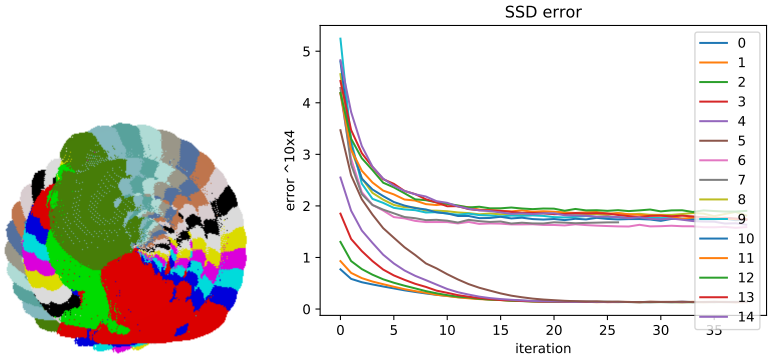

Figure 3: "Perturbations around axis z. Green bunny is the destination mesh."

s of the bounding box that surrounds the destination mesh (length of diagonal). The factors are generated as:

$$f_i \in \{[0.01, 0.1] \text{ with step } =0.01 \} \cup \{0.1, 0.15, 0.2\}$$

And the resultant noise is:

$$\sigma_i = f_i s$$

The error naturally is increased as noise is increased because, even when the meshes are matched as best as they could, the distances from corresponding points will be bigger because of the noise.

On the other hand, he visual results are very good and the meshes are matched as good as they can. The error results can be seen at figure 5.

The split between the errors can be explained by figure 4 where we see where the split happens. Two bunnies with similar rotation end up with very different results and very different errors. This could be due to sub par rejection.

Task 4

We implement the sub-sampling part of the algorithm

Task 5

For this task part of the algorithm was implemented. The idea is to sort meshes based on some metric so that meshes that are close to each other to be matched first rather than matching all the meshes to a single mesh from the get go.

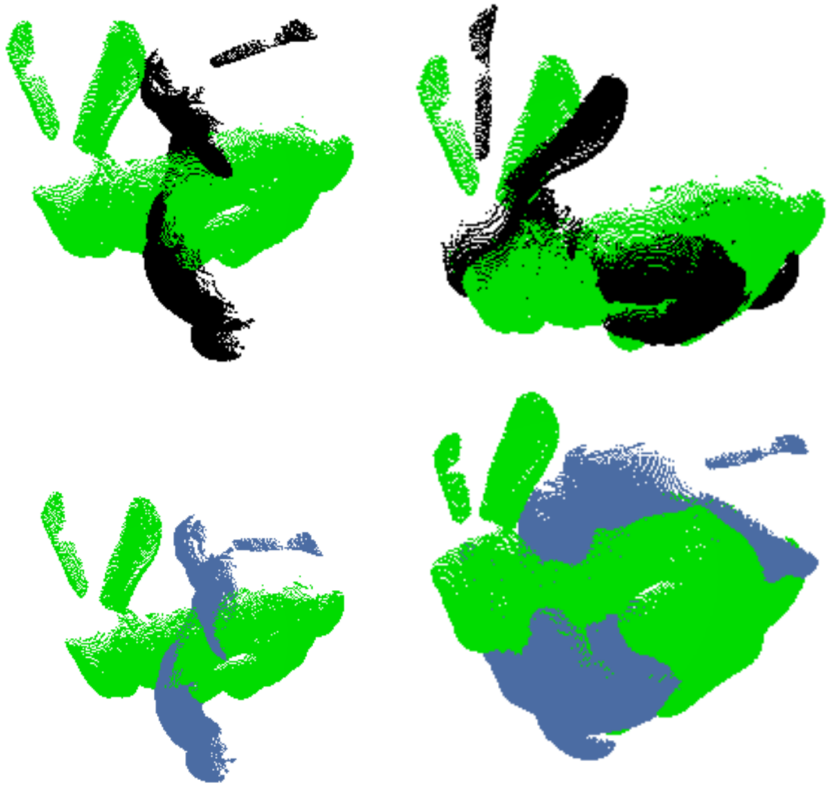

Figure 4: Initial and best mesh matching transformation of bunnies with similar initial rotation.

In my code I just made the assumption that I know which meshes are closer together and I sort them manually. The meshes are then put into a queue. Each iteration we deque a mesh and match it with the one that is at the head of the queue. When we reach to the destination mesh we start all over again

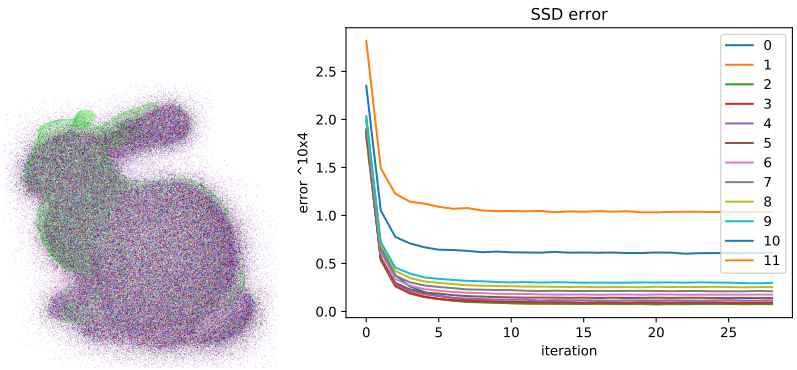

Figure 5: Noisy versions of the source bunny superimposed. Green bunny is the destination mesh.