COMP0119 CW1

Feb 2024

1 Half edge mesh data structure:

1.1 Cube:

Following figure 1, the cube can be generated, shown in figure 2.

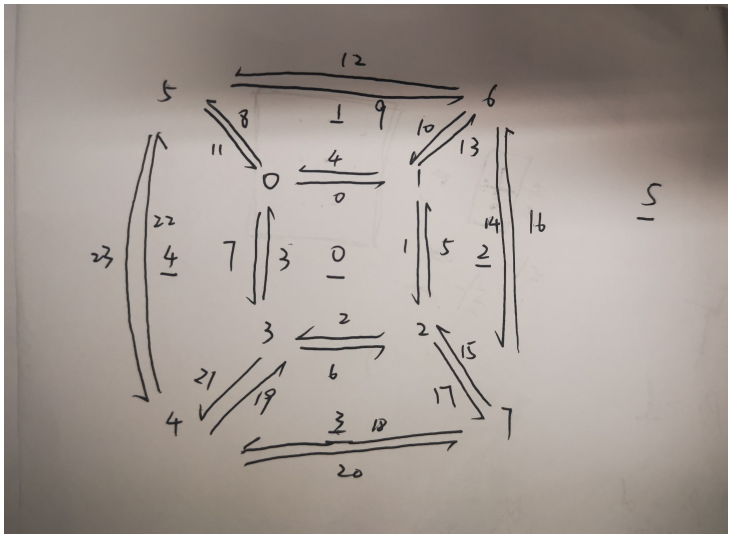


Figure 1: half edge map for cube

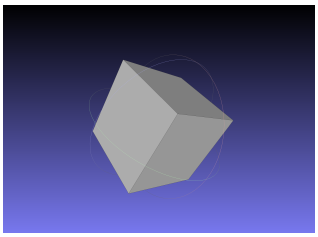


Figure 2: generated cube in meshlab

1.2 Centroid:

The centroid position can be calculated by the average of the points around the facet. To get the points around a facet, take a facet pointer to get a half-edge on the facet. Then follow the pointer to the next half-edge until back at the beginning.

1.3 Dual graph:

The dual graph method takes in the half-edge mesh for the original polyhedron and outputs a dual graph half-edge mesh. The number of facets and vertices are flipped for the dual graph. And the number of edges are the same. We can first connect each vertex with edges by traversing the facets in the original polyhedron. This is done by going to the opposite half-edge of the current half-edge and get the facet(which would be the vertex of the dual graph). After the vertex connection is established, we can convert it in to half-edge mesh structure using the pseudo algorithm notes from the provided half-edge package:

```
for each face F
        {
            for each edge (u,v) of F
            {
                Edges[ pair(u,v) ] = new HalfEdge();
                Edges[ pair(u,v) ]->face = F;
            }
            for each edge (u,v) of F
            {
                set Edges[ pair(u,v) ]->nextHalfEdge to next half-edge in F
                if ( Edges.find( pair(v,u) ) != Edges.end() )
                {
                    Edges[ pair(u,v) ]->oppositeHalfEdge = Edges[ pair(v,u) ];
                    Edges[ pair(v,u) ]->oppositeHalfEdge = Edges[ pair(u,v) ];
            }
        }
```

The result of the dual graph of the cube is shown in figure 3.

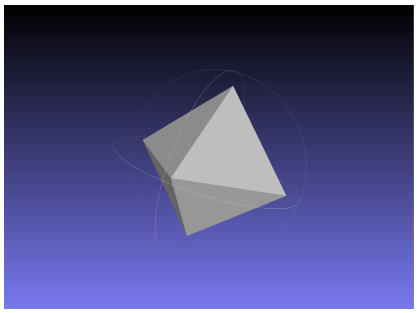


Figure 3: dual f the cube

2 Core section:

2.1 Simple ICP:

The meshes *bun000_v2.ply* and *bun045_v2.ply* are chosen for the first task. The original relative position is shown in figure 4. The four steps of the basic ICP algorithm is followed according to the

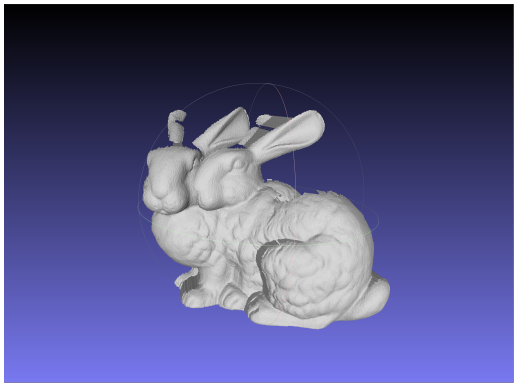


Figure 4: Original two meshes

lecture, namely: 1. matching closest point pairs, 2. rejecting bad pairs, this is done by a combination of distance and curvature elimination. 3. Compute the rotation and transformation according to the derivation 1, and 4. update the new mesh by the transformation and iterate.

$$R = V \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0s & \det\left(VU^T\right) \end{bmatrix} U^T \tag{1}$$
$$t = \bar{p} - R\bar{q}$$

The resulted mesh is shown in figure 5. The meshes are colored in Meshlab, The left and right parts are almost all red or white because the other mesh does not contain those information.

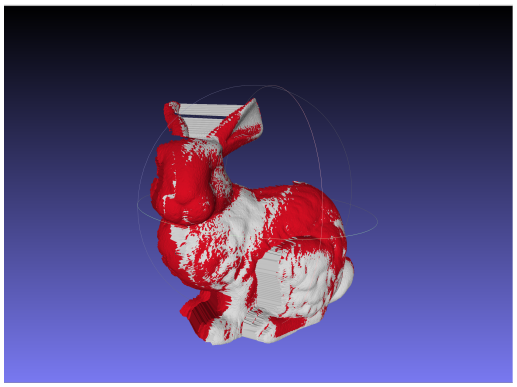


Figure 5: colored aligned meshes

For weighted ICP points, we optimize the objective function:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i}^{n} w_i \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i \right\|^2$$

, where w_i are the weights for each point and R is rigid transformation where $RR^T = I$ and $det(R) = 1$. We can get the minimized solution by setting the derivatives for both variables to zero. Thus, equation 5 in the derivation notes can be rewritten as:

$$\frac{\sum_i^n w_i \mathbf{R}\mathbf{p}_i}{n} + \frac{\sum_i^n w_i \mathbf{t}}{n} - \frac{\sum_i^n w_i \mathbf{q}_i}{n} = 0$$

So, by making $\bar{p}$ and $\bar{q}$ the weighted mean of ps and qs, equation 5 still holds. E(R) becomes:

$$E(\mathbf{R}, \mathbf{t}) = \sum_{i}^{n} w_i \left\| \mathbf{R}\tilde{\mathbf{p}}_i - \tilde{\mathbf{q}}_i \right\|^2$$

Bringing t back to the objective: we can get:

$$R* = \max_R tr(\mathbf{W}\tilde{\mathbf{P}}^{\mathbf{T}}\mathbf{R}\tilde{\mathbf{Q}}) = \max_R tr(\mathbf{R}\tilde{\mathbf{Q}}\mathbf{W}\tilde{\mathbf{P}}^{\mathbf{T}}) = \max_R tr(\mathbf{R}\mathbf{S})$$

, where W is diagonal matrix of weights. Then, the SVD decomposition is performed to $\mathbf{S}$ and we arrive at

$$\mathbf{S} = \mathbf{U}\Sigma\mathbf{V}^\top$$

By substitution:

$$tr(\mathbf{R}\tilde{\mathbf{Q}}\mathbf{W}\tilde{\mathbf{P}}^{\mathbf{T}}) = tr(\mathbf{R}\mathbf{S}) = \mathrm{tr}\left(\Sigma\mathbf{V}^\top\mathbf{R}\mathbf{U}\right)$$

By definition, $\mathbf{R}, \mathbf{U}$ and $\mathbf{V}$ are all orthogonal matrices, so the matrix $\mathbf{M} = \mathbf{V}^T\mathbf{R}\mathbf{U}$ is orthogonal. Then $\mathbf{M}$ satisfies that the rows and columns of the matrix are all orthogonal vectors, then $|m_{ij}| \leq 1$. Denoting diagonal elements of the singular value matrix as $\sigma_1, \sigma_2, \ldots, \sigma_d \geq 0$, we can derive

$$\mathrm{tr}\left(\Sigma\mathbf{V}^\top\mathbf{R}\mathbf{U}\right) = \mathrm{tr}(\Sigma\mathbf{M}) = \sum_{i=1}^{d} \sigma_i m_{ii} \leq \sum_{i=1}^{d} \sigma_i$$

When $m_{ii} = 1$ we have the optimal solution:

$$\mathbf{M} = \mathbf{I} = \mathbf{V}^\top\mathbf{R}\mathbf{U} \Rightarrow \mathbf{R} = \mathbf{V}\mathbf{U}^\top.$$

$\mathbf{R}$ is rigid transform matrix with det $= \pm 1$, We can distinguish it by the following conditions,

$$\begin{cases} \det\left(\mathbf{V}\mathbf{U}^\top\right) = -1, \mathbf{R} \text{ is the reflection matrix} \\ \det\left(\mathbf{V}\mathbf{U}^\top\right) = +1, \mathbf{R} \text{ is the rotation matrix.} \end{cases}$$

Thus, the rest follows the derivation notes with $\bar{p}$ and $\bar{q}$ being the weighted mean, and the SVD is operated on $\mathbf{Q}\mathbf{W}\mathbf{P}^T$

$$\begin{cases} \overline{\mathbf{p}} = \frac{\sum_i^n w_i \mathbf{p}_i}{n} \\ \overline{\mathbf{q}} = \frac{\sum_i^n w_i \mathbf{q}_i}{n} \end{cases}$$

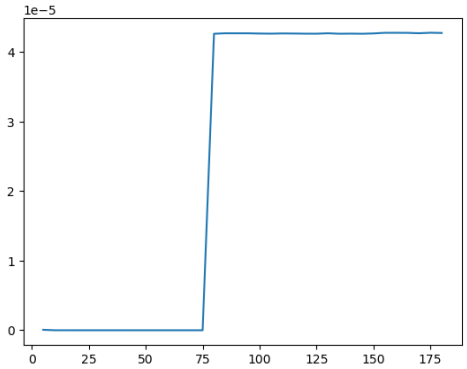


Figure 6: The error by perturbing angle around z axis, where the x axis in the plot is the change of angle

2.2 Perturbing angle and noise:

- The change of angle is from 0 to 180 degrees with 5 degree increments. The results is shown in figure 6. From the figure, it shows that from the angles above 75 degrees, the algorithm starts to match significantly bad point pairs to have misalignment.
- The noise is a zero mean Gaussian additive noise. The standard deviation of the distribution is a ratio of the range that measures the scale of the mesh. The range is calculated by the difference between maximum and minimum positions for the three dimension. The result is shown in 7. We can see that the error increase linearly with noise for small amount of perturbation. But for larger noise added, the mesh deforms greatly, meaning that there is no alignment and the error gets very large, this is not shown in the figure because the smaller noise increment for small noise will become a horizontal line.

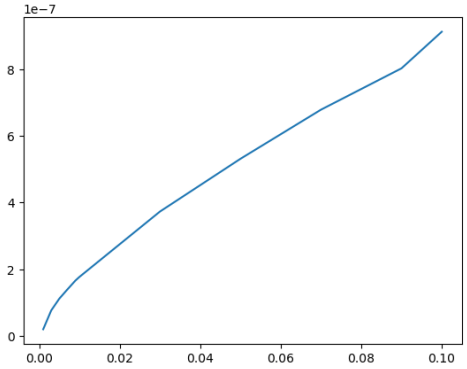


Figure 7: The error by perturbing noise, where the x axis is the ratio for σ = boundingbox $\times$ ratio

2.3 Sampling:

Sampling is implemented by taking randomly with no replacement from the points in the mesh to be fitted to. The speed is faster with the cost of accuracy. The relationship is shown in 8. It shows that

although taking the average of multiple ICP runs, the error for smaller sampling ratio is unstable.

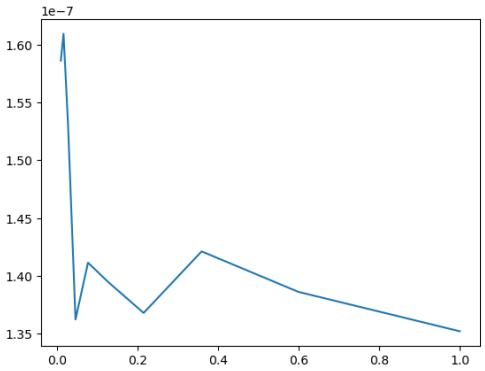
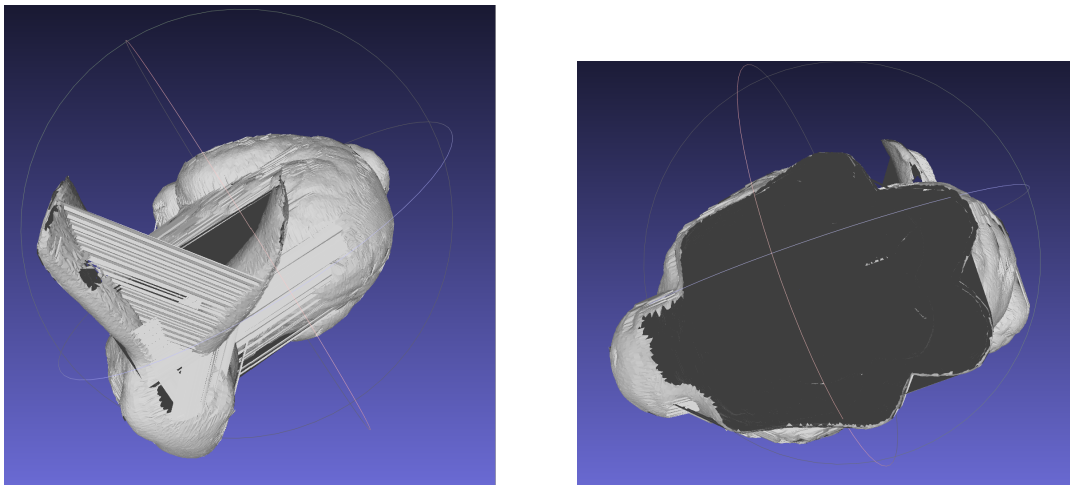


Figure 8: The error by increasing sampling, where the x axis is the sampling ratio

2.4 Global:

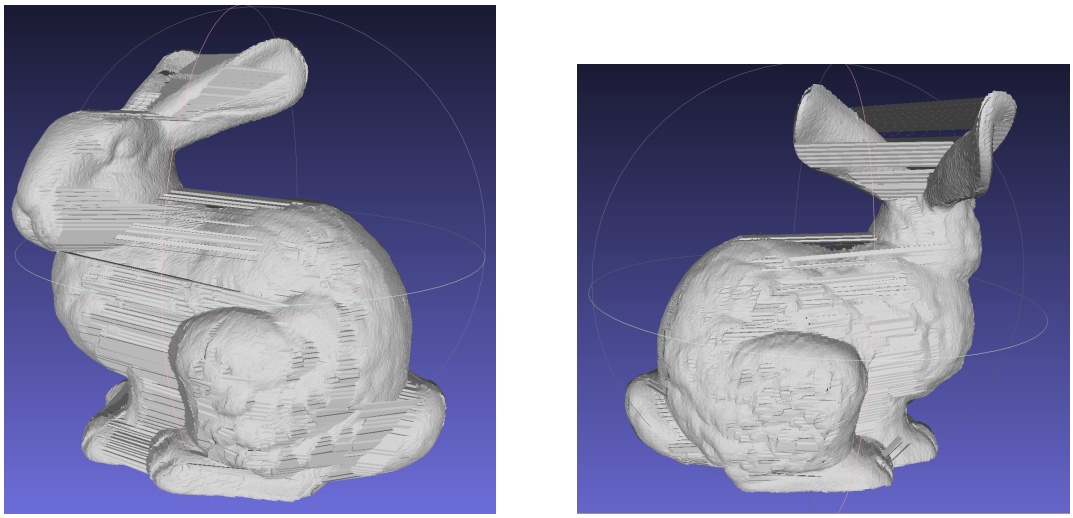
First, the meshes are checked and manually roughly aligned to a similar global position. Then, to avoid misalignment, the ICP algorithm is run for all meshes pairs once. Then according to the lectures brute force method 2. For each mesh vertices, all the other vertices are fixed as one single mesh and the ICP is performed. This is easier to implement, but distribute errors slower. The results are shown in figure 9 to 11.



(a) Top

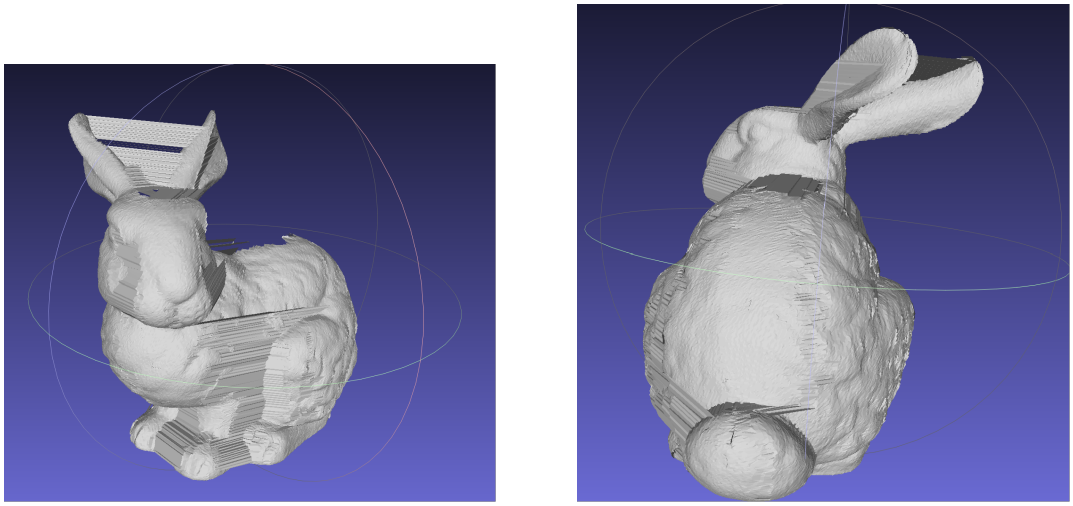
(b) Bottom

Figure 9: Global



(a) Left (b) Right

Figure 10: Global



(a) Front (b) Back

Figure 11: Global

2.5 Point to normal:

The derivation is followed from a Linear Least-Squares Optimization for Point-to-Plane ICP Surface Registration[1]. The normal map is shown in figure 12.

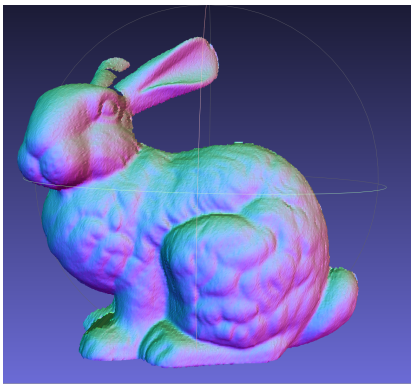


Figure 12: Normal colored mesh

References

[1] Kok-Lim Low. "Linear least-squares optimization for point-to-plane icp surface registration". In: *Chapel Hill, University of North Carolina* 4.10 (2004), pp. 1–3.