*National University of Singapore*

*College of Design and Engineering*

# ME5413 Autonomous Mobile Robotics: Homework 2

*Group 9:*

Chen Yihui A0263115N

Wang Renjie A0263387U

*Supervisor:*

Prof. Marcelo H Ang Jr

*Email Address: e1010473@u.nus.edu*

*e1010745@u.nus.edu*

March 3, 2023

# 1 Task 1: ICP Algorithm for Matched Point Cloud

In Task 1, we assume the correspondence is known for the two point clouds and perform ICP algorithm to do point cloud registration. Firstly, the SVD-based ICP algorithm is reviewed.

Given the source point cloud $X = \{x_1, x_2, ..., x_m\}$ and target point cloud $Y = \{y_1, y_2, ..., y_m\}$, the task for point cloud registration is to find a rotation matrix $R$ and a translation matrix $t$ that can transform $X$ to $Y$. In other words, we need to find $R$ and $t$ to minimize the sum of the squared distance between the corresponding points in the two point clouds:

$$\begin{aligned} E(R,t) &= \frac{1}{m}\sum_{i=1}^{m} \|y_i - Rx_i - t\|^2 \\ &= \frac{1}{m}\sum_{i=1}^{m}(\|y_i - u_y - R(x_i - u_x)\|^2 + \|u_y - Ru_x - t\|^2) \end{aligned} \tag{1}$$

where $u_x$ and $u_y$ are the centroid of the point set $X$ and $Y$, $u_x = \frac{1}{m}\sum_{i=1}^{m} x_i$ and $u_y = \frac{1}{m}\sum_{i=1}^{m} y_i$. To simplify the problem, we divide Eq. 1 to:

$$E_1(R,t) = \frac{1}{m}\sum_{i=1}^{m} \|y_i - u_y - R(x_i - u_x)\|^2, \quad E_2(R,t) = \|u_y - Ru_x - t\|^2 \tag{2}$$

Therefore, we can first find the $R$ to minimize $E_1(R,t)$ and then calculate $t$ from $E_2(R,t)$. Suppose $x_i^{'} = x_i - u_x$, $y_i^{'} = y_i - u_y$, the problem turns to:

$$min\{E_1(R,t)\} = min\{\frac{1}{m}\sum_{i=1}^{m}(y_i^{'T} y_i^{'} + x_i^{'T} R^T R x_i^{'} - 2y_i^{'T} R x_i^{'})\} = max\{\sum_{i=1}^{m} y_i^{'T} R x_i^{'}\} \tag{3}$$

Using the properties of trace, we derive:

$$\sum_{i=1}^{m} y_i^{'T} R x_i^{'} = \sum_{i=1}^{m} Trace(y_i^{'T} R x_i^{'}) = Trace(\sum_{i=1}^{m} R x_i^{'} y_i^{'T}) = Trace(RH) \tag{4}$$

where $H = \sum_{i=1}^{m} x_i^{'} y_i^{'T}$. According to Schwarz inequality, we can maximum Eq. 4 by finding a $R$ that can convert $RH$ to the form of $AA^T$. Therefore, we apply Singular Value Decomposition (SVD) on $H$ and get $H = U\Sigma V^T$. Then we choose the rotation matrix:

$$R = VU^T \tag{5}$$

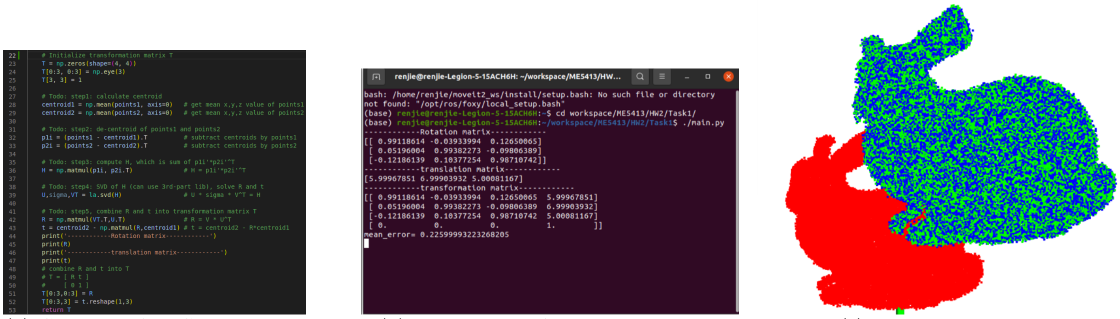
Then, $RH = VU^T U\Sigma V^T = V\Sigma V^T = V\Sigma^{1/2}(V\Sigma^{1/2})^T$, which is in the form of $AA^T$ and we finally minimize $E_1(R,t)$. After obtaining $R$, we can solve the translation matrix $t$ to minimize $E_2(R,t)$:

$$t = u_y - Ru_x \tag{6}$$

Since we have known the corresponding point pairs between source point cloud $X$ and target $Y$, we can easily get the homogeneous transformation matrix $T \in SE(3)$ by combing the rotation matrix $R \in SO(3)$ and translation matrix $t \in \mathbb{R}^3$ computed in Eq. 5 and 6:

$$T = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} \tag{7}$$

The core code of ICP algorithm with comment is shown in Fig. 1a and derived transformation matrix $T$ is recorded in Fig. 1b. As is shown in Fig. 1c, the source point cloud (red) is transformed to the blue one, which is overlapped with the target point cloud (green). The final mean error between the blue and green point clouds is about 0.226.



(a) Core code for ICP algorithm (b) Derived transformation matrix $T$ (c) Visualization result

Figure 1: Implementation of ICP algorithm for matched point cloud

## 2 Task 2: ICP Algorithm for Unmatched Point Cloud

In real cases, the correspondence between the source and target point cloud is usually unknown. A common method is to regard the nearest neighbor point as corresponding point and solve the problem iteratively. The ICP algorithm for unmatched point cloud is given in Alg. 1.

**Algorithm 1** SVD-based ICP algorithm for unmatched point cloud

Initialize the accumulated transformation matrix $T\_accumulated$.
**while** $error > threshold$ **do**
  Step 1: Find the corresponding points from two point clouds.
  Step 2: Compute the transformation matrix $T$ following the steps in Task 1.
  Step 3: Update accmulated matrix. $(T\_accumulated = T * T\_accumulated)$
  Step 4: Apply alignment. $(x_i^{'} = T * x_i)$
  Step 5: Update error.
**end while**

The *threshold* is set to be 0.1 and core code for the iterative ICP algorithm is shown in Fig. 2. To show the more and more overlapped rabits, we record the transformation matrix $T$ and $T\_accumulated$ in iteration 7, 15 and 23, as well as visualizing the two point clouds and mean error between them (Fig. 6).

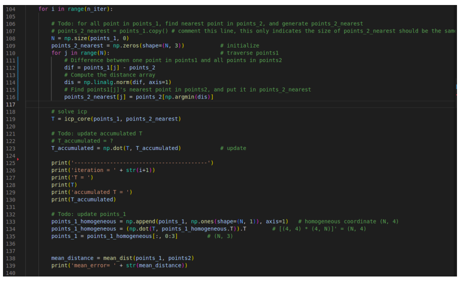
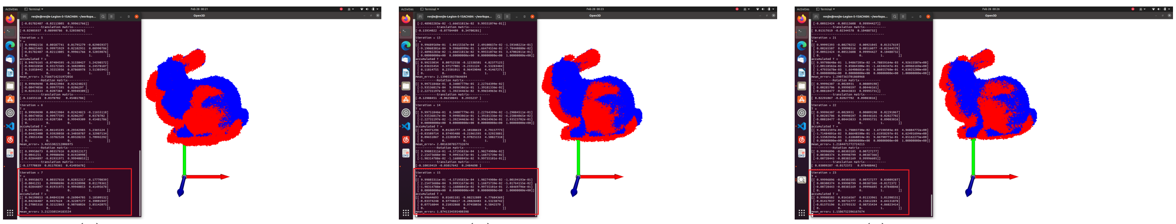


Figure 2: Code for ICP algorithm with unmatched point cloud



(a) Results at iteration 7 (b) Results at iteration 15 (c) Results at iteration 23

Figure 3: Implementation of ICP algorithm for unmatched point cloud

We can see the mean error is reduced after each iteration (3.212 to 1.874 to 1.151), and it declines more and more slowly as iteration number increases. The final result at 30th iteration is shown in Fig. 4, where the mean error is about 0.797 and still higher than the mannually set threshold. We increase the maximum iteration number to 50 and the result is shown in Fig. **??**. The mean error is only ? now and the accumulated transformation matrix $T\_accumulated$ is very close to that we derived in Task1. However, the cost time is rather high (650s) because of the time-consuming corresponding points search procedure. In practice, kd-tree can be used to search for the nearest points more efficiently. Moreover, some learning-based methods have been popular for solving the point cloud registration problem.

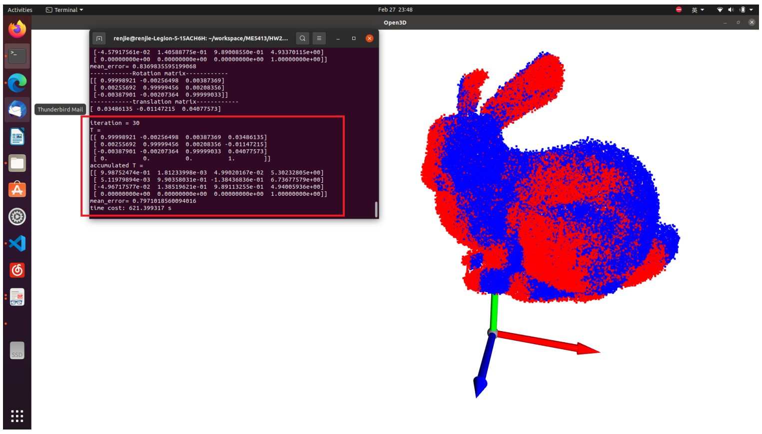


Figure 4: Code for ICP algorithm with unmatched point cloud

# 3 Task 3: Running SLAM Algorithms

## 3.1 Cartographer with 2D LiDAR

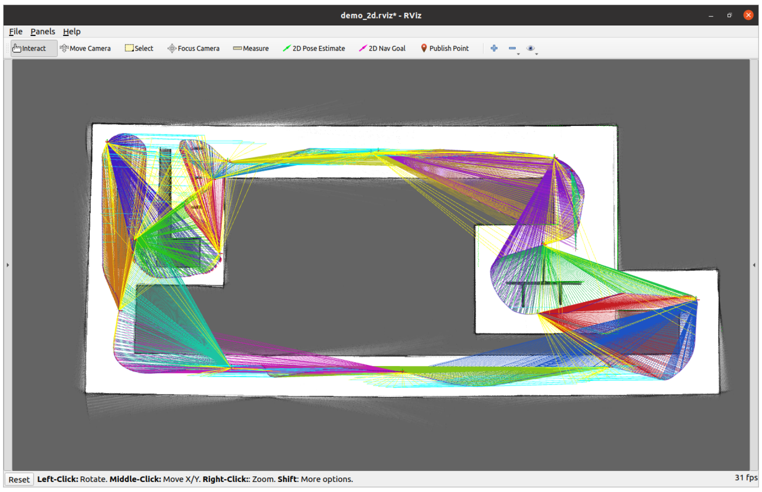
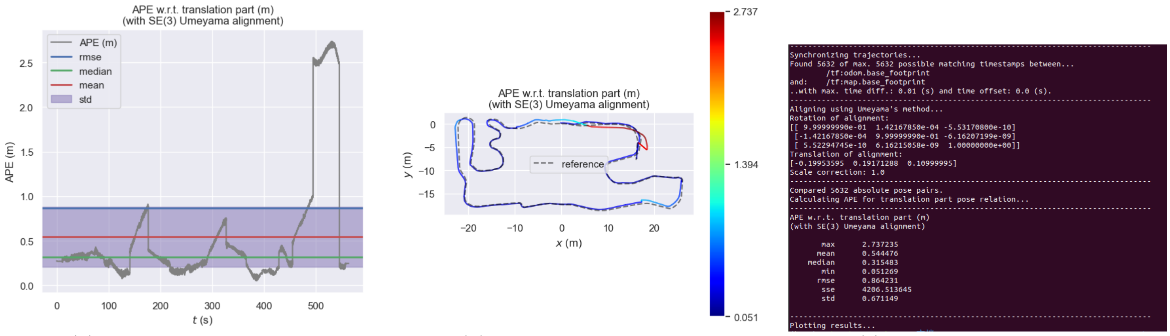



Figure 5: Code for ICP algorithm with unmatched point cloud




(a) Results at iteration 7 (b) Results at iteration 15 (c) Results at iteration 23

Figure 6: Implementation of ICP algorithm for unmatched point cloud

In your report:

- Briefly describe the SLAM1 algorithm you choose to use
- Describe the detailed process of running the SLAM algorithms, give a screenshot of your algorithm running in rviz
- Highlight your modification/tuning done on the algorithms to achieve better results
- Show the Absolute RMSE, as well as the plots generated by the EVO tool
- Discuss the drawbacks/failures in your tests, provide some analysis and propose possible solutions (illustrate with figures and provide your hypostasis)