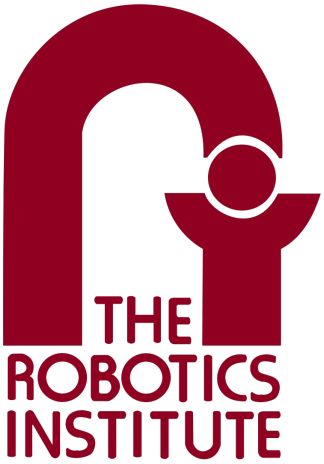
Homework 4



Dense SLAM with Point-based Fusion

16-833A Spring 2025

Author: **Boxiang (William) Fu**
Andrew ID: boxiangf
E-mail: *boxiangf@andrew.cmu.edu*

April 3, 2025

Contents

1 Overview

No questions from this section.

2 Iterative Closest Point (ICP)

2.1 Projective data association

A valid correspondence must satisfy the following conditions:

$$\begin{aligned} 0 \leq & u < W \\ 0 \leq & v < H \\ d \geq & 0 \end{aligned}$$

In other words, the pixel must be within the boundaries of the vertex map and the depth must be non-negative (so it is not inside the camera and/or the data entry is not missing).

For the second filter, it is necessary to impose the restriction $|p - q| < d_{thr}$ because we should reject incorrect correspondences arising from 2D projections. This reduces the likelihood of false associations in image space but are actually far apart in 3D space (this could arise from noise, projection errors, and/or occlusions etc).

Additionally, since ICP is an iterative optimization procedure, we restrict associations to prevent pose updates heading in the wrong direction. This will lead to the optimizer potentially diverging and giving poor results. Therefore, the restriction $|p - q| < d_{thr}$ ensures that only geometrically consistent pairs influence the optimization.

Finally, the restriction ensures that only the most informative and trustworthy matches are left after filtering. This reduces the number of correspondences and helps convergence in fewer iterations.

2.2 Linearization

Rewriting $r_i(\delta R, \delta t)$ in the form of:

$$r_i(\alpha, \beta, \gamma, t_x, t_y, t_z) = A_i \boldsymbol{\xi} + b_i = A_i \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} + b_i$$

We use the definition $r_i(\delta R, \delta \mathbf{t}) = \mathbf{n}_i^\top (\delta R \mathbf{p}_i' + \delta \mathbf{t} - \mathbf{q}_i)$ and the small-angle approximation $\delta R \approx I + [\boldsymbol{\omega}]_\times$, where $\boldsymbol{\omega} = [\alpha, \beta, \gamma]^T$. Substituting in, we obtain:

$$\begin{aligned} r_i(\delta R, \delta \mathbf{t}) &= \mathbf{n}_i^\top (\delta R \mathbf{p}_i' + \delta \mathbf{t} - \mathbf{q}_i) \\ &\approx \mathbf{n}_i^\top \left(\mathbf{p}_i' + [\boldsymbol{\omega}]_\times \mathbf{p}_i' + \delta \mathbf{t} - \mathbf{q}_i\right) \\ &= \mathbf{n}_i^\top \left([\boldsymbol{\omega}]_\times \mathbf{p}_i' + \delta \mathbf{t} + (\mathbf{p}_i' - \mathbf{q}_i)\right) \end{aligned}$$

Using the identity $[\boldsymbol{\omega}]_\times \mathbf{p}_i' = -[\mathbf{p}_i']_\times \boldsymbol{\omega}$, we have

$$r_i = \mathbf{n}_i^\top \left(-[\mathbf{p}_i']_\times \boldsymbol{\omega} + \delta \mathbf{t} + \mathbf{p}_i' - \mathbf{q}_i\right)$$

Rewriting as a linear system:

$$r_i(\alpha, \beta, \gamma, t_x, t_y, t_z) = \underbrace{\mathbf{n}_i^\top \begin{bmatrix} -[\mathbf{p}_i']_\times & I \end{bmatrix}}_{A_i} \underbrace{\begin{bmatrix} \boldsymbol{\omega} \\ \delta\mathbf{t} \end{bmatrix}}_{\boldsymbol{\xi}} + \underbrace{\mathbf{n}_i^\top (\mathbf{p}_i' - \mathbf{q}_i)}_{b_i}$$

Thus, we get:

$$\begin{aligned} A_i &= \mathbf{n}_i^\top \begin{bmatrix} -[\mathbf{p}_i']_\times & I \end{bmatrix}, \\ b_i &= \mathbf{n}_i^\top (\mathbf{p}_i' - \mathbf{q}_i) \end{aligned}$$

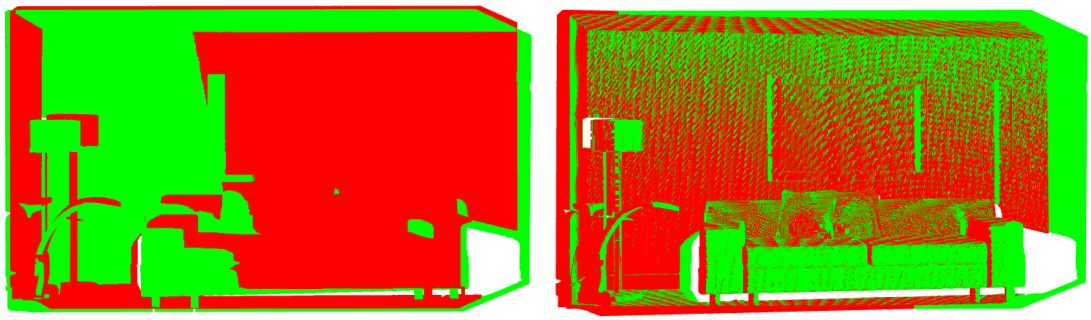
Note that the A_i is a 1×6 matrix and b_i is a scalar.

2.3 Optimization

The linear system that provides the closed form solution of $\boldsymbol{\xi}$ can be solved through the normal equation $A^T A\boldsymbol{\xi} = -A^T b$ derived in the lectures. Note that a negative sign appears before $A^T b$ as the minimization equation is of the form $Mx - z$ while our characterization is $r_i = A_i\boldsymbol{\xi} + b_i$, so we introduce a negative sign for the b vector.

Exploiting sparsity, we chose to use the LU formulation. This refactors into $A^T A = R^T R$, where R is an upper-triangular matrix. We are able to efficiently solve by forward substitution for $R^T y = -A^T b$ (since R^T is lower-triangular), and then back substituting for $R\boldsymbol{\xi} = y$ to obtain $\boldsymbol{\xi}$. This is implemented in the `solve` function.

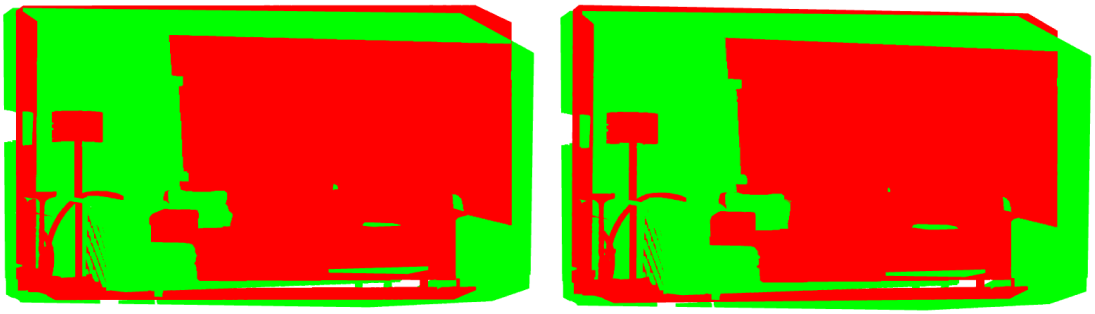
The visualization before and after ICP for frame 10 to frame 50 is shown in Figure 1. The number of inliers after 9 iterations is 281,370 with an average loss of 8.8886e-06.



(a) Before ICP **(b)** After ICP

Figure 1: Point clouds before and after registration from frame 10 to frame 50

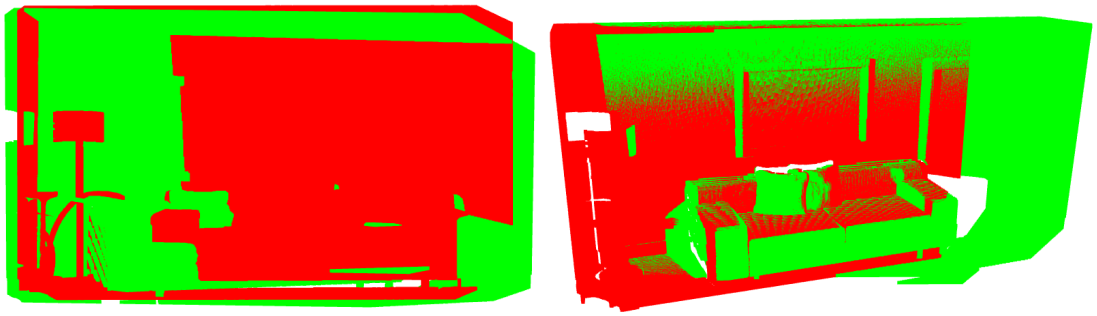
The visualization before and after ICP for frame 10 to frame 100 is shown in Figure 2. Using the default distance threshold of `dist_diff` $= 0.07$, the number of inliers after 9 iterations is 28,525 with an average loss of 1.2803e-03.



(a) Before ICP **(b)** After ICP

Figure 2: Point clouds before and after registration from frame 10 to frame 100 with `dist_diff` = 0.07

The visualization before and after ICP for frame 10 to frame 100 is shown in Figure 3. Using an increased distance threshold of `dist_diff` = 0.17, the number of inliers after 9 iterations is 236,786 with an average loss of 1.5993e-04.



(a) Before ICP **(b)** After ICP

Figure 3: Point clouds before and after registration from frame 10 to frame 100 with `dist_diff` = 0.17

Referring to Figure 1, the visualization before ICP is misaligned due to the camera shifting between the two frames. We can visually see that the red point cloud does not align with the green point cloud. After ICP, the point clouds are visibly much more aligned as we associate the data points between the two frames using ICP. The reason that the ICP is successful is because the source and target frames are not too far apart. This meant that data association is effective at reducing the average loss and converging to the accurate homogeneous transform. Additionally, the small variation between the source and target frame also meant that our rotation linearization is appropriate.

Referring to Figure 2 and Figure 3, the visualization before ICP is very misaligned since the camera has traveled quite significantly between the two frames. We can visually see that the red point cloud does not align with the green point cloud. Using the default distance threshold of `dist_diff` = 0.07, the ICP fails to give the correct association between the two frames. The alignment remains poor, and the average loss remains high. The reason for the failure is primarily because the distance threshold of `dist_diff` = 0.07 is too tight. The camera has moved significantly, and hence the transformation is above the threshold. Limiting the transformation distance to `dist_diff` = 0.07 means that the data points are not being correctly associated (as the association exceeds the threshold).

When we increase the threshold to `dist_diff` = 0.17, we can visibly see that the association is much better. We achieve more success in associating the two point clouds in the two frames. This confirms the previous hypothesis that the distance threshold is too

tight. Therefore, increasing the distance threshold allows the ICP to correctly associate data points between the two frames. However, the average loss is still higher compared to the loss from frame 10 to frame 50. This is because the homogeneous transform between frame 10 and frame 100 is much more pronounced than frame 10 to frame 50. The linearization of rotation is not as accurate since the angle is no longer small, thus leading to a greater average loss.

3 Point-based Fusion

3.1 Filter

Implemented filter in `filter_pass1` and `filter_pass2` functions.

3.2 Merge

The weighted average of the positions (p_{new}) is given by:

$$p_{new} = \frac{w \cdot p + 1 \cdot (R_c^w \cdot q + t_c^w)}{w + 1}$$

Where $R_c^w \cdot q + t_c^w$ is the transformation of point q from the frame coordinate system to the map coordinate system. The weighted average of the unnormalized normals (n_{new}) is given by:

$$n_{new}^{unnormalized} = \frac{w \cdot n_p + 1 \cdot (R_c^w \cdot n_q)}{w + 1}$$

Normalizations can be done by dividing the new normal by its norm:

$$n_{new}^{normalized} = \frac{n_{new}^{unnormalized}}{\|n_{new}^{unnormalized}\|}$$

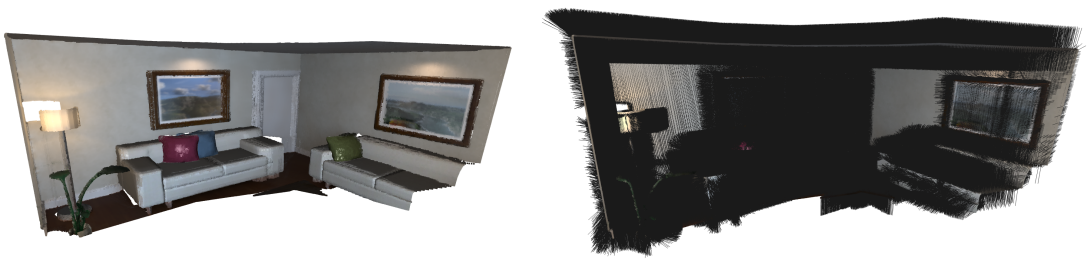
The implementation is done in `merge`.

3.3 Addition

Implemented in `add` function.

3.4 Results

The visualization with a normal map is shown in Figure 4. After fusing 200 frames, the final number of points in the map is 1,362,157.



(a) Visualization map **(b)** Normal map

Figure 4: Fusion with ground truth poses with a normal map

Noting a downsample factor of 2 for all the inputs, we obtain a compression ratio of 8.87%. This is obtained by dividing the final number of points in the map $(1,362,157)$ by the number of frames times the number of pixels per frame after downsampling $(200 \times \frac{640}{2} \times \frac{480}{2} = 15,360,000)$.

4 The dense SLAM system

The **source frame is the input RGBD-frame** and the **target frame is the map frame**. In general, we cannot swap their roles. This is because the ICP procedure transforms the 3D point from the input frame into the target frame. We then project it to 2D in the target frame and index it into the target's vertex map. This only works when we have a vertex map in a fixed target frame that does not change between different instances of the input frame. If the RGBD-frame is used as the target frame instead, at every frame update, we would need to project the entire vertex map into the new RGBD-frame. This would be very noisy and computationally expensive. Therefore, it is infeasible to switch their roles.

The visualization with a normal map is shown in Figure 5. The estimated trajectory against the ground truth is shown in Figure 6.

(a) Visualization map **(b)** Normal map

Figure 5: Fusion with poses estimated from ICP

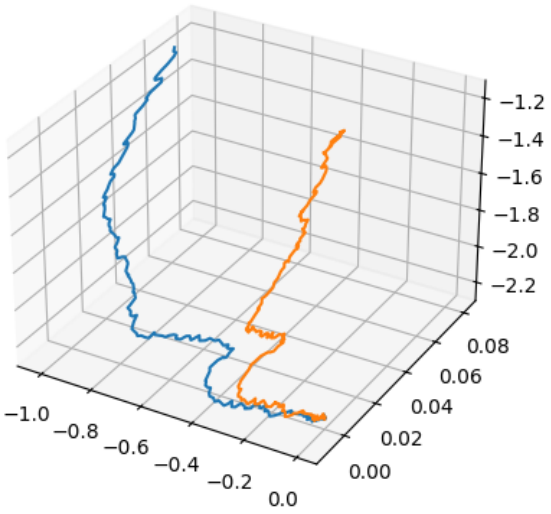


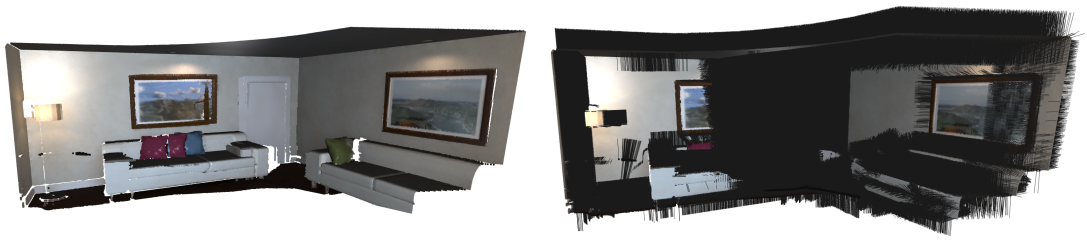
Figure 6: Estimated trajectory against ground truth

Referring to Figure 5, the visualization is similar to the one observed in Figure 4. However, a noticeable difference is that the vertex map now only has 807,812 points as compared to the 1,362,157 previously. This is because we are now estimating the homogeneous transformations back to the map frame and are no longer using the ground-truth. This causes noise in data association, thus leading to less points being identified as belonging to the same vertex point during ICP. This is most pronouncedly manifested when

looking at the picture frames on the wall. The relatively texture-less oak-colored frames tends to be blurry, indicating that the ICP data association for these points tends to be poor.

Referring to Figure 6, the drift appears to be quite significant. For the bonus question, I attempted to reduce this drift through two methods. The first is that I tuned the rejection filter parameters from `dist_diff` $= 0.03$ to `dist_diff` $= 0.01$ and `angle_diff` $= 5$ degrees to `angle_diff` $= 1$ degree. This is because from the dataset, I noticed that the camera moves relatively slowly between sequential frames. Therefore, the translational and rotational separation between the frames should be small. Setting a tighter bound filters out the outliers.

Additionally, I implemented a filter to eliminate dormant points. Points that survive for more than 20 frames but have points associated with it for less than 15 frames are filtered out and removed. This eliminates any outliers that may be present for only a few frames from the vertex map. The drift reduction methods are implemented in the `fusion.py` file. The results are depicted in Figure 7 and Figure 8. Note that the oak-colored frames on the wall are less blurry, indicating that the drift reduction measures are working as intended. The trajectories displayed in Figure 8 also visually appear tighter. However, this is at the expense of some missing point renderings (e.g. the bottom side of the door).



(a) Visualization map **(b)** Normal map

Figure 7: Fusion with poses estimated from ICP and dormant point elimination

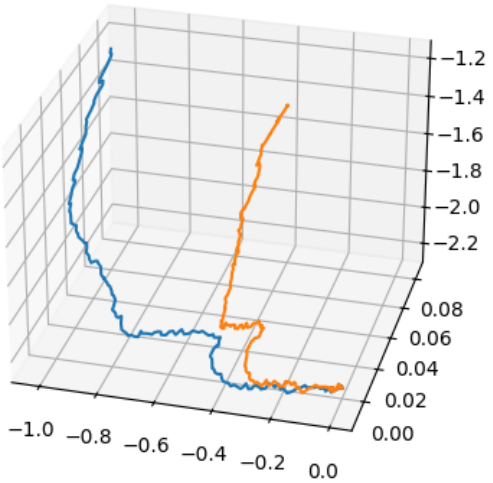


Figure 8: Estimated trajectory against ground truth with dormant point elimination