Name: Ashwin Biju Nair | **Andrew ID:** abijunai

16-833: Robot Localization and Mapping

Homework 4 - Dense SLAM with Point-based Fusion

2. Iterative Closest Point (ICP)

2.1. Projective Data Association

When a point p is projected to a vertex map, the (u, v) coordinate is obtained with a depth d. Given are the vertex maps height (H) and width (W). The coordinates u, v & depth d must satisfy the following conditions:

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

In python the indexes start from 0, therefore the '$\leq$' condition for (u, v) is not used. After obtaining the correspondences, an additional filter on distance and angle thresholds are applied to ensure that the correspondences between the source point cloud and the target vertex map are geometrically valid. In the distance filter, depth values of the transformed source points (after projecting them into the target frame) are compared with the corresponding depth values in the `target_vertex_map`. The filter ensures that only points with a depth difference below a specified threshold (dist_diff = 0.07) are considered valid correspondences.

Similarly, the angle filter ensures that only the transformed normals that make an angle of < 20 with the corresponding normals in the `target_normal_map` are selected as valid points.

This filtering is done primarily to handle occlusions and ensure geometric consistency. Occlusions occur when parts of the scene that are visible in one frame but hidden in another, leading to some points in the source point cloud to not have a valid correspondences in the target frame. Without these filters, these occluded points could incorrectly match to unrelated points in the target frame, introducing noise into the alignment process.

2.2 Linearization

KinectFusion seeks to minimize the point-to-plane error between associated points (p, q) where p is from the source and q is from the target. The error function can be written by:

$$\sum_{i \in \Omega} r_i^2(R, t) = \left\| n_{q_i}^T (R p_i + t - q_i) \right\|^2, \tag{1}$$

where (p_i, q_i) is the i-th associated point pair in the assocation set $\Omega = \{(p_n, q_n)\}$, and n_{q_i} is the estimated normal at q_i (we have covered the data association step, i.e., how to associate p_i and q_i).

By using small-angle assumption and parameterizing δR with

$$\delta R = \begin{bmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{bmatrix}$$

$$\delta t = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}$$

With δR and δt, we can write down the incremental update version as

$$\sum_{i \in \Omega} r_i^2(\delta R, \delta t) = \left\| n_{q_i}^T((\delta R)p_i^{'} + \delta t - q_i) \right\|^2, \tag{2}$$

where $p_i' = R^0 p_i + t^0$ aiming at solving $\alpha, \beta, \gamma, t_x, t_y, t_z$ with the given initial R^0, t^0.

Rearranging the system such that,

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

where A_i is a 1×6 matrix and b_i is a scalar.

Writing $r_i(\delta R, \delta t)$ as

$$r_i(\delta R, \delta t) = n_{q_i}^T[(\delta R)p_i^{'} + \delta t - q_i], \tag{3}$$

We can define δR as follows

$$\delta R = I + [w]_\times \tag{4}$$

where

$$[w]_\times = \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix}, w = [\alpha, \beta, \gamma]^\top \in R^3.$$

Therefore, $(\delta R)p_i^{'}$ can be written as

$$(\delta R)p_i^{'} = p_i^{'} + [w]_\times p_i^{'} \tag{5}$$

Substituting back in equation (3) we get,

$$r_i(\delta R, \delta t) = n_{q_i}^T(p_i^{'} + [w]_\times p_i^{'} + \delta t - q_i), \tag{6}$$

On simplifying,

$$r_i(\delta R, \delta t) = n_{q_i}^T \left([w]_\times p_i^{'} + \delta t \right) - n_{q_i}^T \left(q_i - p \right), \tag{7}$$

Here we notice that the second term $-n_{q_i}^T(q_i - p)$ is a scalar and corresponds to b_i. Further simplifying the first term we get,

$$n_{q_i}^T([w]_\times p_i^{'} + \delta t) = n_{q_i}^T \left(\begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix} \begin{bmatrix} x & y & z \end{bmatrix} + \begin{bmatrix} t_x & t_y & t_z \end{bmatrix} \right) \tag{8}$$

$$n_{q_i}^T([w]_\times p_i^{'} + \delta t) = n_{q_i}^T \left(\begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \begin{bmatrix} \alpha & \beta & \gamma \end{bmatrix} + \begin{bmatrix} t_x & t_y & t_z \end{bmatrix} \right) \tag{9}$$

$$n_{q_i}^T([w]_\times p_i^{'} + \delta t) = \begin{bmatrix} n_{q_i}^T[p_i]_\times, n_{q_i}^T \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} \tag{10}$$

where $[p_i]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}$

Therefore,

$$A_i = \begin{bmatrix} n_{q_i}^T[p_i]_\times, n_{q_i}^T \end{bmatrix}, b_i = -n_{q_i}^T(q_i - p) \tag{11}$$

2.3. Optimization

The equation to be optimized is

$$\sum_{i=1^n} r_i^2(\alpha, \beta, \gamma, t_x, t_y, t_z) = \sum_{i=1}^{n} ||A_i \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} + b_i||^2. \tag{12}$$

We can write the residual $r_i = n_q^T(Rp + t - q)$, which takes the form $r_i = A_i \begin{bmatrix} \alpha & \beta & \gamma & t_x & t_y & t_z \end{bmatrix}^T + b_i$ where A_i and b_i are defined as shown in equation (11). The residual can be written in the form $r_i = A_i\delta + b_i$. Therefore the close form solution can be written as $A_i\delta = -b_i$.

The above equation is solved using QR factorization in `solve` function.

$$A = QR \tag{13}$$

where Q is orthogonal $(QQ^T = I)$ and R is an upper triangular matrix. Now, Multiply the transpose of Q with b to simplify the system:

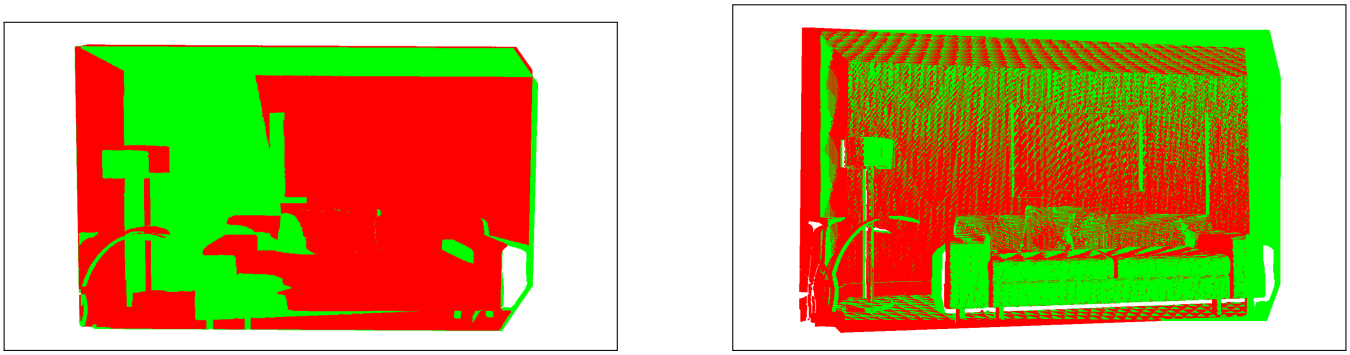
$$Q^T b = c \tag{14}$$

After substitution, the following simplified system is solved:

$$Rx = c \tag{15}$$

Since R is upper triangular, it is solved efficiently using back substitution.

The below figure shows the visualization of the map before and after ICP registration.

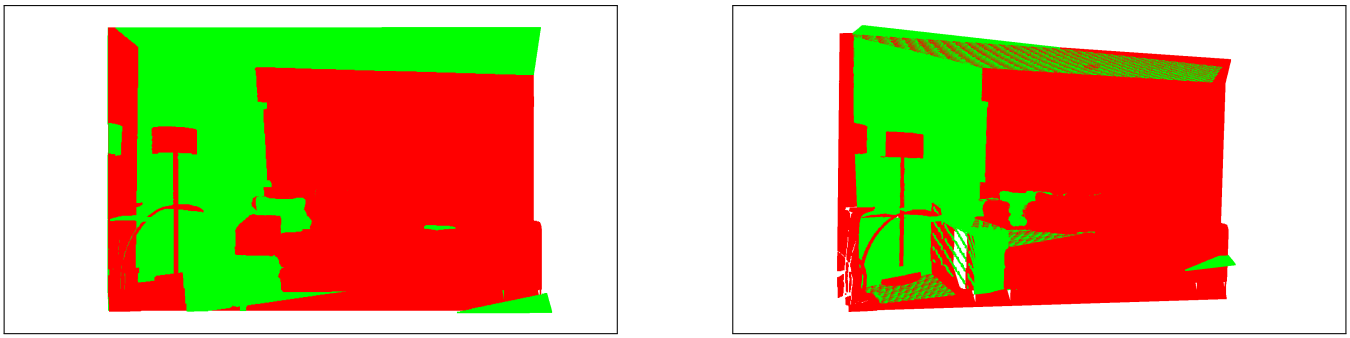


(a) Visualization before ICP

(b) Visualization after ICP

Figure 1: Point clouds before and after registration (source frame- 10, target frame- 50)

The target frame was changed form 50 to 100 and the results are shown in Figure 2. The 100th frame being farther away in the sequence from the reference frame introduce more significant transformations, making it harder for ICP to converge correctly. In Figure 2. (a), there is a noticeable angular disparity between the reference and target frames (large angle), which might cause the small angle assumption that was made during linearization to fail, leading to poor registration.



(a) Visualization before ICP

(b) Visualization after ICP

Figure 2: Point clouds before and after registration (source frame- 10, target frame- 100)

3. Point-based Fusion

3.1. Filter

`filter_pass_1` creates a mask similar to the ICP model with the following constraints:

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

`filter_pass_2` creates a mask based on a stricter constraint as follows:

$$\begin{aligned} distance < 0.03; \\ angle < 5^{\circ} \end{aligned}$$

3.2 Merge

Weighted Average of Positions:

- p: Existing map point in the world coordinate system.
- q: Input point in the camera coordinate system.
- ${R_c}^w$: Rotation matrix from the camera frame to the world frame.
- ${t_c}^w$: Translation vector from the camera frame to the world frame.
- w: Weight of the existing map point (p)

The input point if first transformed into the world coordinate system using:

$$q_w = {R_c}^w q + {t_c}^w \tag{16}$$

The weighted average for this point is:

$$p_{new} = \frac{wp + q_w}{w + 1} \tag{17}$$

$$p_{new} = \frac{wp + {R_c}^w q + {t_c}^w}{w + 1} \tag{18}$$

Weighted average of normals:

- n_p: Existing normal in the world coordinate system.
- n_q: Input normal in the camera coordinate system.
- ${R_c}^w$: Rotation matrix from the camera frame to the world frame.

- w: Weight of the existing map normal (n_p)

The input normal if first transformed into the world coordinate system using:

$$n_{q,w} = R_c{}^w n_q \tag{19}$$

The weighted average for normals is:

$$n_{new} = w n_p + n_{q,w} \tag{20}$$

After this is computed, normalization is done to ensure n_{new} has unit length:

$$n_{new} = \frac{w n_p + n_{q,w}}{||w n_p + n_{q,w}||} \tag{21}$$

$$n_{new} = \frac{w n_p + R_c{}^w n_q}{||w n_p + R_c{}^w n_q||} \tag{22}$$

The above method is implemented in `merge` function.

3.3. Add

The `add` function integrates new, unassociated points from the input frame into the global map. It transforms the input points and normals from the camera coordinate system to the world coordinate system using a rotation matrix R and translation vector t. The transformed points, normalized normals, and corresponding colors are then concatenated to the existing map properties (`self.points`, `self.normals`, `self.colors`). Each new point is initialized with a weight of 1. This process ensures that previously unobserved regions in the scene are incorporated into the map for complete reconstruction.

3.4. Results

The following figure showcases the results of running `fusion.py`.

(a) Visualization of fusion.py

(b) Visualization of Normal Map

Figure 3: Fusion with ground truth poses with a normal map

- Final number of points in the map, $N_{fused} = 1,362,157$

- Downsampled Height, $H_{down} = 480/2 = 240$
- Downsampled Width, $W_{down} = 680/2 = 340$
- Number of frames $= 200$

Finding the number of points if the output is naively concatenated:

$$N_{naive} = H_{down} * W_{down} * (Numberofframes) \tag{23}$$

$$N_{naive} = 240 * 340 * 200 = 16,320,000 \tag{24}$$

The compression ratio is given by:

$$CompressionRatio = \frac{N_{fused}}{N_{naive}} \tag{25}$$

Compression Ratio

$$\text{Compression Ratio} = \frac{1,362,157}{16,320,000} = 0.08346 \approx 8.35\%$$

4. The dense SLAM system

The source is the input RGB-D frame, while the target is the map. If the roles are swapped, ICP would compute a transformation from the map to the camera frame instead of from the camera frame to the map. While mathematically valid, this would require reversing or reinterpreting transformations during integration into SLAM's global coordinate system. Additonally, it would introduce inefficiencies since you'd need to reprocess large amounts of data for every frame because the map is typically more stable and larger than individual frames. The input RGB-D frame is smaller and more dynamic, making it better suited as a source for incremental updates.

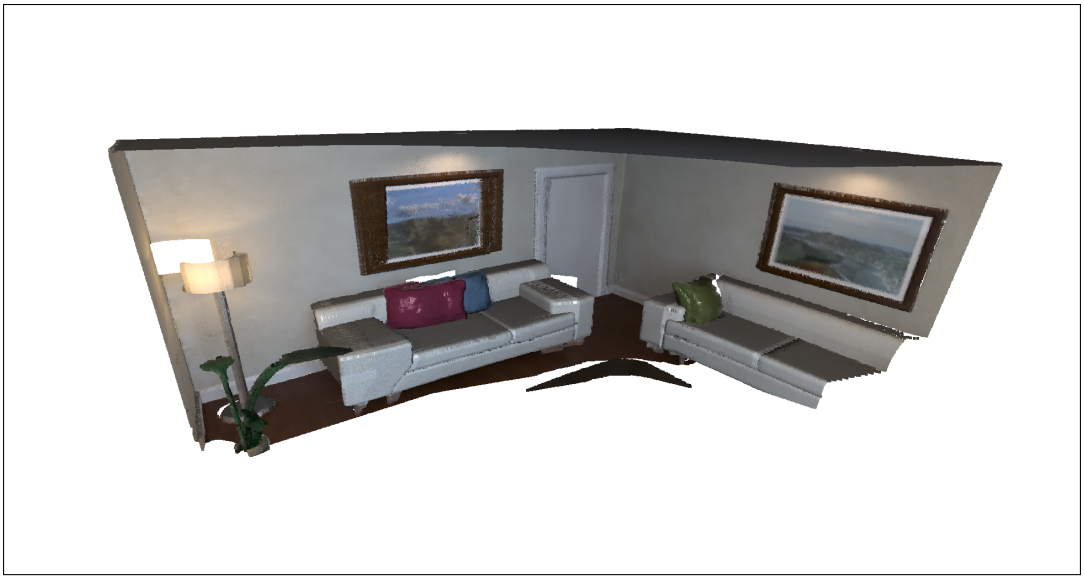


Figure 4: Fusion with poses estimated from frame-to-model ICP

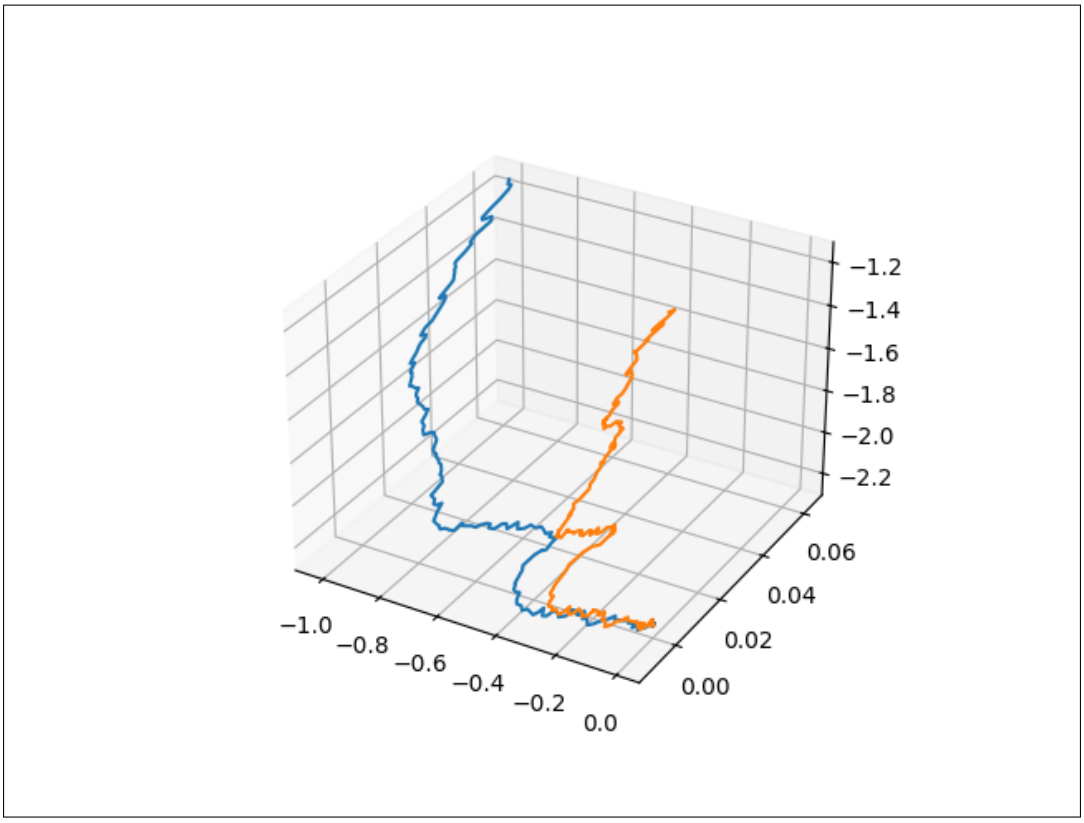


Figure 5: Estimated trajectory and ground truth