16-833: Robot Localization and Mapping, Fall 2024

Solution 4
Dense SLAM with Point-based Fusion

1 Overview

2 Iterative Closest Point (ICP) (50 points)

2.1 Projective data association

Solution: The conditions that u, v, d must satisfy are constraints related to the projective projection and camera pinhole models.

Conditions are:

1. $0 <= u$
2. $u < W$
3. $0 <= v$
4. $v < H$
5. $d > 0$

Where, H, W are the Height and Width of the Target Vertex Map(Image).

Solution: Filtering correspondences based on distance and angle thresholds is necessary to ensure that the identified correspondences are geometrically valid and physically meaningful.

1. **Distance Thresholding**: This ensures that the source point p and it's corresponding point q are spatially close-by.
2. **Angle Thresholding**: This ensures that the surface normals at p and q are aligned, indicating that the source and target points likely belong to the same surface.

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:

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

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).

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

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

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.

Solution: Reorganizing and rewriting the parameters, we get

$$A_i = \begin{bmatrix} n_{q_i z} * p'_{i_y} - n_{q_i y} * p'_{i_z} \\ n_{q_i x} * p'_{i_z} - n_{q_i z} * p'_{i_x} \\ n_{q_i y} * p'_{i_x} - n_{q_i x} * p'_{i_y} \\ n_{q_i x} \\ n_{q_i y} \\ n_{q_i z} \end{bmatrix}$$

$$b_i = \left[n_{q_i}^\top (p_i' - q_i) \right]$$

where $n_{q_i} = [n_{q_i x}, n_{q_i y}, n_{q_i z}]$ & $p_i' = [p'_{i_x}, p'_{i_y}, p'_{i_z}]$

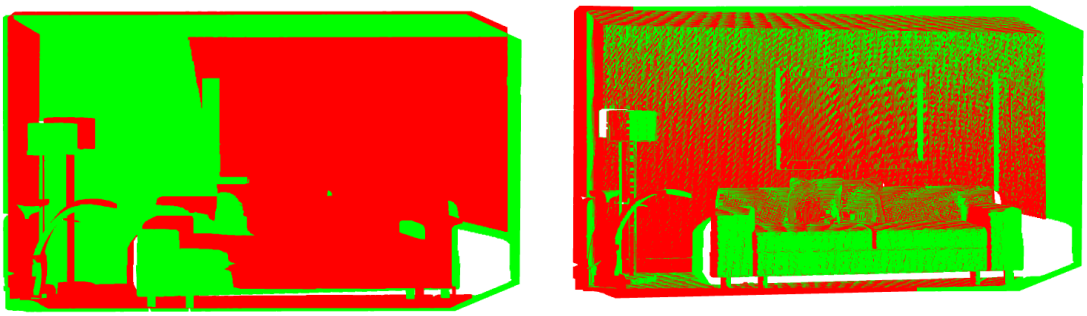

Figure 1: Point clouds before and after registration for source 10 and target 50

```
iter 0: avg loss = 1.3230e-03, inlier count = 71402
iter 1: avg loss = 1.2667e-03, inlier count = 113461
iter 2: avg loss = 3.8088e-04, inlier count = 197850
iter 3: avg loss = 1.5525e-04, inlier count = 247081
iter 4: avg loss = 1.9984e-05, inlier count = 281410
iter 5: avg loss = 8.9510e-06, inlier count = 281404
iter 6: avg loss = 8.8867e-06, inlier count = 281372
iter 7: avg loss = 8.8886e-06, inlier count = 281369
iter 8: avg loss = 8.8885e-06, inlier count = 281370
iter 9: avg loss = 8.8886e-06, inlier count = 281370
```

Figure 2: Average Loss and Inliers for source 10 and target 50 frames

2.3 Optimization

Solution:

We construct the closed-form linear system equation of the form: $Ax = b$ where x is the unknowns $\alpha, \beta, \gamma, t_x, t_y, t_z$.
Finally, we use the equation above using the linear solver LU, in the code.
Note: We change the equation into

$$\sum_{i=1^n} r_i^2(\alpha, \beta, \gamma, t_x, t_y, t_z) = \sum_{i=1}^{n} \left\| A_i \begin{bmatrix} \alpha \\ \beta \\ \gamma \\ t_x \\ t_y \\ t_z \end{bmatrix} - b_i \right\|^2. \tag{3}$$

before using the linear solver LU.

Solution: Figure 1 shows the visualization before and after ICP for source 10 and target 50 frames.

Figure 2 shows the inliers and average loss per iteration (total 10) for source 10 and target 50 frames.

Figure 3 shows the visualization before and after ICP for source 10 and target 100 frames.

Figure 4 shows the inliers and average loss per iteration (total 10) for source 10 and target 100 frames.

Reason for failure with target frame 100

1. **Projection Issues Due to Motion:** When projecting points from the source frame to the target frame's vertex map, the assumption is that the corresponding points lie spatially close in 3D space. However, with significant movement

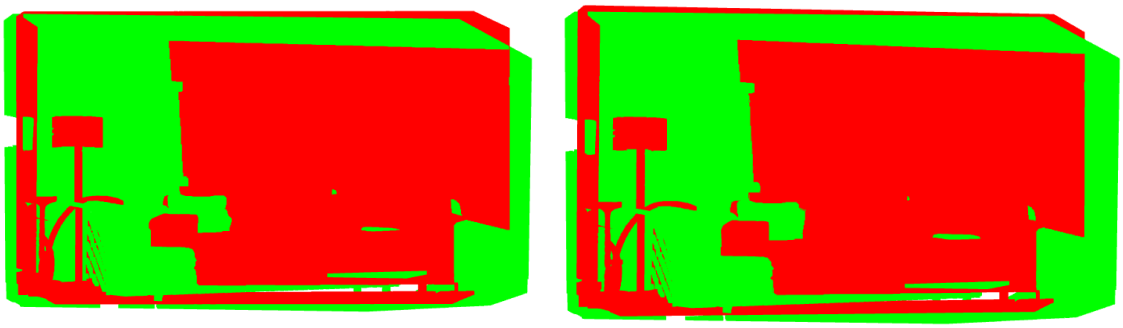

Figure 3: Point clouds before and after registration for source 10 and target 100

```
iter 0: avg loss = 1.2651e-03, inlier count = 26385
iter 1: avg loss = 1.2649e-03, inlier count = 26564
iter 2: avg loss = 1.2661e-03, inlier count = 26770
iter 3: avg loss = 1.2623e-03, inlier count = 26920
iter 4: avg loss = 1.2680e-03, inlier count = 27178
iter 5: avg loss = 1.2550e-03, inlier count = 27300
iter 6: avg loss = 1.2733e-03, inlier count = 27667
iter 7: avg loss = 1.2633e-03, inlier count = 27812
iter 8: avg loss = 1.2854e-03, inlier count = 28255
iter 9: avg loss = 1.2803e-03, inlier count = 28525
```

Figure 4: Average Loss and Inliers for source 10 and target 100 frames

or changes in viewpoint (as seen between frame 10 and frame 100), the projection of points onto the target frame's vertex map may result in points falling outside valid regions. The majority of projected points could be considered invalid (outside bounds or with mismatched depth), leading to their removal during correspondence filtering. This drastically reduces the number of valid inliers for alignment.

2. **Assumptions of Small Transformations:** ICP assumes that the initial transformation brings the source and target frames close enough for correspondences to be reliably established. Large relative motion between frames violates this assumption, leading to poor initialization and misaligned point clouds.

3. **Challenging Geometry:** Frame pairs with significant movement or non-overlapping regions will result in geometric misalignment, as parts of the source frame may project to areas in the target frame that lack corresponding structures.

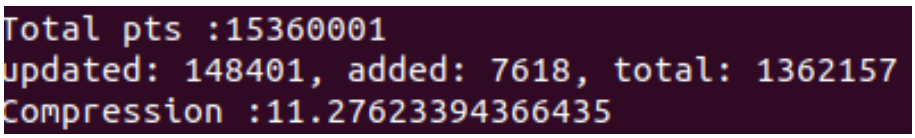

Figure 5: Terminal Screenshot depicting total input points, total points in the map and the compression ratio

3 Point-based Fusion (40 points)

3.1 Filter

Solution: Code implemented in ***fusion.py*** file.

3.2 Merge

Solution: The equation for the weighted average of position is:

$$weighted\_positions = \frac{q * w + p' * 1}{1 + w}$$

where, $p' = (R_c^w)^T * p + t_c^w$

The equation for the weighted average of normals is:

$$weighted\_normals = \frac{n_q * w + n'_p * 1}{1 + w}$$

where, $n'_p = (R_c^w)^T * n_p$

Code implemented in ***fusion.py*** file.
Note:

1. The normal vector n_p represents the plane's orientation, so only the rotational component of the transformation will affect the error when multiplied by it.
2. The weighted normals will be normalized after the above operation.
3. Weights will be updated as $w_{new} = 1 + w$

3.3 Addition

Solution: Code implemented in ***fusion.py*** file.

3.4 Results

Table 1 Shows the 4 different views of the final fusion map after running for 200 frames with a downsampling factor of 2.

Table 2 Shows the 4 different views of the final fusion map after running for 200 frames with a downsampling factor of 2.

Figure 5 Depicts the total **input points: 15360001**,**total points in map: 1362157**, and **compression ratio: 11.276**.

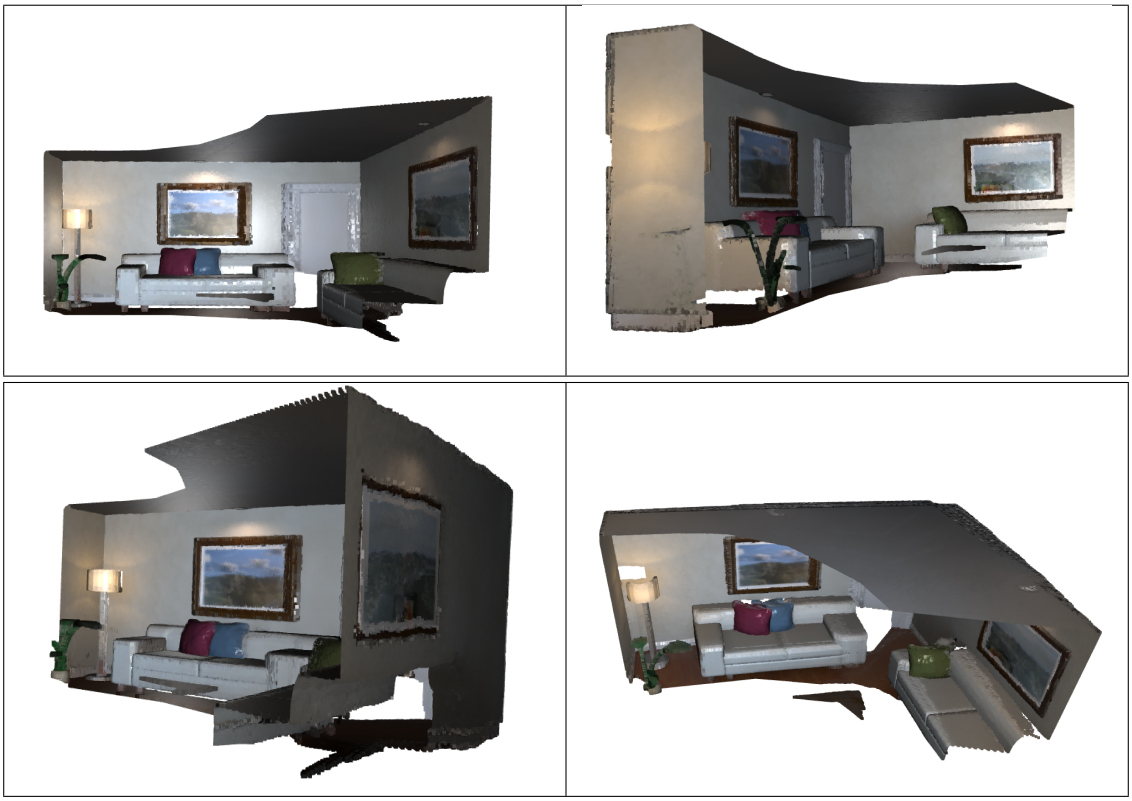

Table 1: Fusion map with 200 frames(down-sample: 2)

Table 2: Normal map with 200 frames(down-sample: 2)

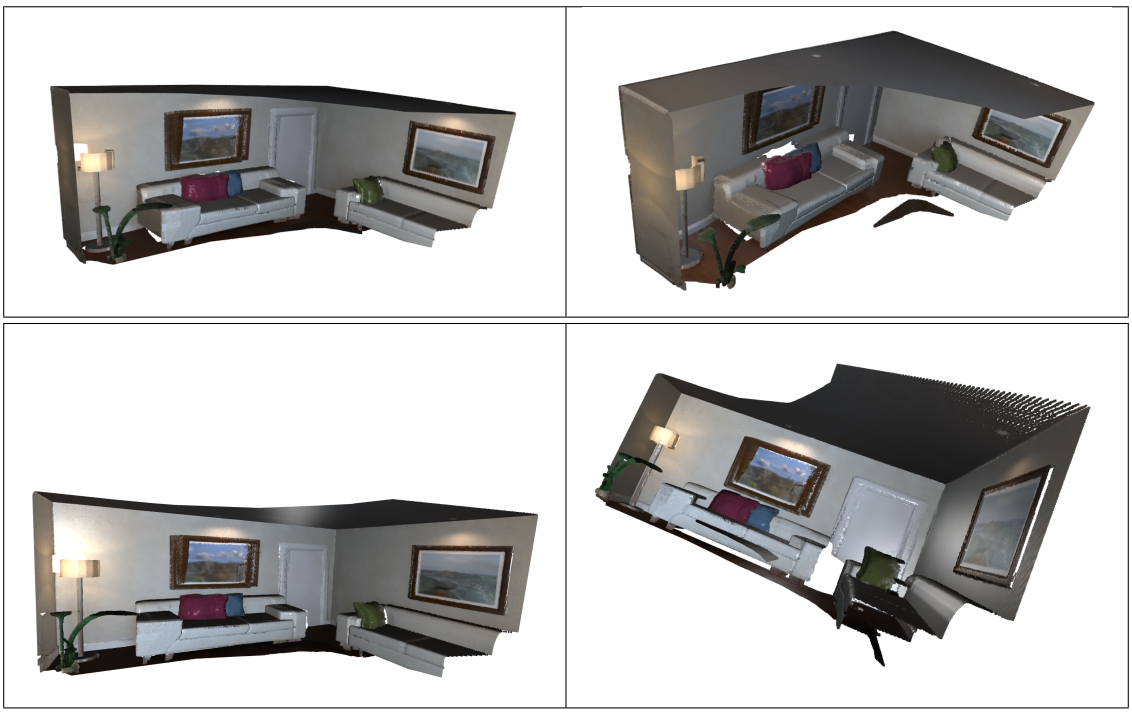

Table 3: Dense SLAM result with 4 different views

4 The dense SLAM system (20 points + 10 points)

Solution:

- **Source:** The input RGBD frame (the current frame you are trying to align with the map).
- **Target:** The map (the reference model or point cloud to which the input frame is being aligned).

No, we cannot simply swap their roles in ICP because:

- The source is the frame we are trying to align to the target (the map). The map is a more stable, fixed reference, while the input frame is dynamic and subject to transformation.
- The source points are transformed to fit the target points, so swapping them would invert the process and lead to incorrect results or non-convergence, as we typically want to align the moving frame to the fixed map.
- Swapping the roles would make the process slower because aligning the large, fixed map to the smaller input frame requires more computations, increasing the number of iterations and making the optimization less efficient.

Solution: ***Table 3*** shows the visualization after running the ensure dense SLAM system for 200 frames.

Figure 6 depicts the plot of the estimated trajectory vs ground truth for the 200 frames.

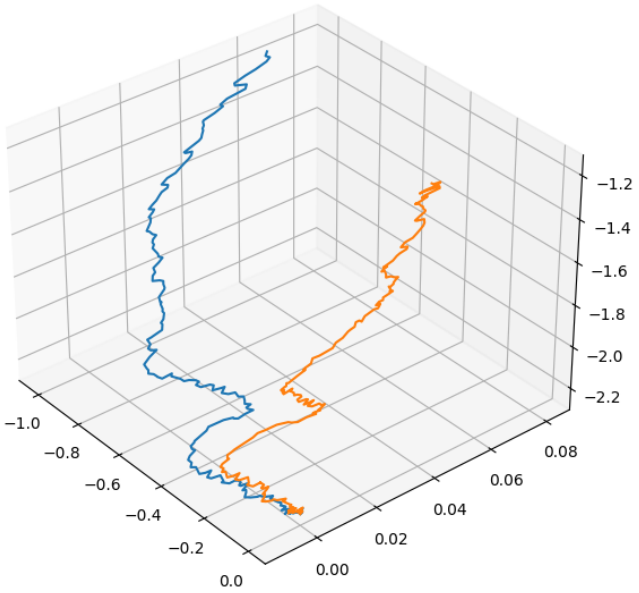

Figure 6: Visualization of the estimated trajectory vs ground truth