16-833: Robot Localization and Mapping
Homework 4: Written Report

Shahram Najam Syed
Andrew ID: snsyed

26th November, 2024

2. Iterative Closest Point (ICP)

2.1 Projective Data Association

Question (5 points)

Suppose you have projected a point p to a vertex map and obtained the u, v coordinates with a depth d. The vertex map's height and width are H and W. Write down the conditions u, v, d must satisfy to set up a valid correspondence. Also implement the `TODO: first filter` section in the function `find_projective_correspondence`.

Answer:

To establish a valid correspondence between the source point $\mathbf{p}$ and a point in the target vertex map:

1. **Pixel Coordinates Within Image Bounds:**
   The projected pixel coordinates (u, v) must lie within the bounds of the vertex map dimensions. This ensures that we are referencing valid pixels in the vertex map.

   Mathematically:

   $$0 \leq u < W$$

   $$0 \leq v < H$$

   This condition checks that the pixel coordinates are non-negative and less than the width W and height H of the vertex map.

2. **Positive Depth Value:**
   The depth d at the projected pixel (u, v) must be positive. A non-positive depth indicates invalid or missing data (e.g., no point exists at that pixel in the target vertex map).

   Mathematically:

   $$d > 0$$

   A positive depth ensures that there is a valid 3D point $\mathbf{q}$ corresponding to the pixel (u, v) in the target vertex map.

These conditions are critical to ensure that the projected point $\mathbf{p}$ corresponds to a valid point $\mathbf{q}$ in the target vertex map, which is necessary for accurate data association in the ICP algorithm.

Question (5 points):

After obtaining the correspondences q from the vertex map and the corresponding normal $\mathbf{n_q}$ in the normal map, you will need to additionally filter them by distance thresholds and angle thresholds, so that $\|\mathbf{p} - \mathbf{q}\| < d_{\mathbf{thr}}$. Why is this step necessary? Implement this filter in the `TODO: second filter` section in the function `find_projective_correspondence`.

Answer:

Filtering correspondences using both distance thresholds $\|\mathbf{p}-\mathbf{q}\| < d_{\text{thr}}$ and angle thresholds between normals is necessary for the following reasons:

1. **Ensuring Geometric Proximity:**
   - Projective data association may assign correspondences based on pixel projections, which does not guarantee that the points are close in 3D space.
   - Points $\mathbf{p}$ and $\mathbf{q}$ might project to the same pixel (u, v) but be physically distant due to depth discontinuities or occlusions in the scene.
   - By applying a distance threshold, we ensure that only points that are close in 3D space are considered valid correspondences.
2. **Ensuring Surface Compatibility:**
   - The angle between normals of corresponding points should be small if they truly belong to the same surface.
   - Large angular differences between normals often indicate that points belong to different surfaces or are at surface boundaries.
   - Filtering based on normal angles helps preserve surface continuity and prevents matching points from different surface orientations.
3. **Reducing the Impact of Outliers:**

- Depth sensors can produce noisy or erroneous measurements, leading to incorrect correspondences that can adversely affect the ICP algorithm.
- Filtering out correspondences where $\|\mathbf{p} - \mathbf{q}\|$ exceeds a reasonable threshold or where normal angles differ significantly helps eliminate outliers and reduces the influence of noise.

4. **Maintaining Algorithmic Assumptions:**
   - ICP assumes that the initial transformation is a good approximation and that correspondences are locally accurate.
   - Enforcing both distance and angle thresholds helps maintain this assumption by only considering points that are likely to correspond under small transformations and have similar surface orientations.

5. **Improving Convergence and Stability:**
   - Including distant or incorrect correspondences, or points with vastly different surface orientations, can lead to inaccurate estimation of the transformation parameters.
   - Filtering ensures that the optimization process focuses on reliable data, enhancing convergence speed and stability.

2.2 Linearization

Question (15 points):

Now reorganize the parameters and rewrite $r_i(\delta R, \delta t)$ in the form of:

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

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

Answer:

To linearize the error function and express r_i in the desired form, we proceed step by step.

Step 1: Expand the Error Function The point-to-plane error function is given by:

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

where:

- n_{q_i} is the normal at point q_i,
- $p_i' = R_0 p_i + t_0$ is the transformed source point using the initial estimate R_0, t_0,
- δR and δt are the incremental rotation and translation to be estimated.

Step 2: Linearize the Rotation Matrix Using the small-angle approximation, the incremental rotation δR can be approximated as:

$$\delta R \approx I + [\omega]_\times,$$

where $\omega = [\alpha, \beta, \gamma]^\top$ represents the rotation vector, and $[\omega]_\times$ is the skew-symmetric matrix (cross-product operator) corresponding to ω:

$$[\omega]_\times = \begin{bmatrix} 0 & -\gamma & \beta \\ \gamma & 0 & -\alpha \\ -\beta & \alpha & 0 \end{bmatrix}.$$

Step 3: Expand $\delta R\, p_i'$ Using the approximation:

$$\delta R\, p_i' = (I + [\omega]_\times) p_i' = p_i' + [\omega]_\times p_i'.$$

Step 4: Substitute Back into the Error Function Substitute $\delta R\, p_i'$ back into r_i:

$$r_i = [n_{q_i}]^\top (p_i' + [\omega]_\times p_i' + \delta t - q_i),$$

$$r_i = [n_{q_i}]^\top (p_i' - q_i) + [n_{q_i}]^\top [\omega]_\times p_i' + [n_{q_i}]^\top \delta t.$$

Step 5: Simplify the Second Term The term $[n_{q_i}]^\top [\omega]_\times p_i'$ can be rewritten using properties of the cross product:

$$[n_{q_i}]^\top [\omega]_\times p_i' = -(p_i' \times n_{q_i})^\top \omega.$$

Thus:

$$r_i = [n_{q_i}]^\top (p_i' - q_i) - (p_i' \times n_{q_i})^\top \omega + [n_{q_i}]^\top \delta t.$$

Step 6: Rewrite the Error Function Group terms to identify A_i and b_i:

$$r_i = \begin{bmatrix} -(p_i' \times n_{q_i})^\top & [n_{q_i}]^\top \end{bmatrix} \begin{bmatrix} \omega \\ \delta t \end{bmatrix} + [n_{q_i}]^\top (p_i' - q_i).$$

Step 7: Define A_i and b_i We can now identify:

$$A_i = \begin{bmatrix} -(p_i' \times n_{q_i})^\top & [n_{q_i}]^\top \end{bmatrix}, \quad b_i = [n_{q_i}]^\top (p_i' - q_i).$$

Here:

- A_i is a 1×6 matrix encapsulating the relationship between the rotational increments (α, β, γ), the translational increments (t_x, t_y, t_z), and the geometry of the points and normals.
- b_i is a scalar representing the residual error based on the current estimate p_i' and the corresponding point q_i.

Step 8: Final Expression The linearized form of the error function is:

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

where A_i and b_i are computed as shown above.

2.3 Optimization

Question (15 points):

Write down the linear system that provides a closed-form solution of $\alpha, \beta, \gamma, t_x, t_y, t_z$ in terms of A_i and b_i. You may either choose a QR formulation by expanding a matrix and filling in rows (resulting in an $n \times 6$ linear system), or an LU formulation by summing up n matrices (resulting in a 6×6 system). Implement `build_linear_system` and the corresponding `solve` with your derivation.

Answer:

After linearizing the residuals r_i in terms of the incremental parameters $\alpha, \beta, \gamma, t_x, t_y, t_z$, we have:

$$r_i = A_i x + b_i,$$

where:

- A_i is a 1×6 matrix derived previously,
- $x = [\alpha, \beta, \gamma, t_x, t_y, t_z]^\top$,
- b_i is a scalar.

Our objective is to find x that minimizes the sum of squared residuals:

$$\min_x \sum_{i=1}^{n} r_i^2 = \min_x \sum_{i=1}^{n} (A_i x + b_i)^2.$$

This is a standard linear least-squares problem.

Formulating the Linear System

1. **Stack the Equations:**
   We stack all n residual equations into a single matrix equation. Let A be an $n \times 6$ matrix where each row is A_i, and let b be an $n \times 1$ vector where each entry is b_i:

   $$r = Ax + b.$$

2. **Objective Function in Matrix Form:**
   The objective function becomes:

   $$\min_x \|Ax + b\|^2.$$

3. **Rewriting for Least Squares:**
   To align with the standard least-squares form $\min_x \|Ax - y\|^2$, we define:

   $$y = -b.$$

   Then, the problem becomes:

   $$\min_x \|Ax - y\|^2.$$

4. **Deriving the Normal Equations:**
   The solution to the least-squares problem is given by the normal equations:

   $$A^\top A x = A^\top y.$$

   This is a 6×6 linear system that can be solved for x.

Implementation Details

1. **In `build_linear_system`:**
   - For QR Decomposition:
     - Initialize A as an $n \times 6$ matrix.
     - Initialize b as an $n \times 1$ vector.
     - For each correspondence i:
       - Compute A_i and append to A,
       - Compute b_i and append to b.
2. **In `solve`:** Using QR Decomposition:

```
def solve(A, b):
    """
    Solves the linear system A x = -b for x.
    :param A: (N, 6) matrix in the QR formulation
    :param b: (N, 1) vector in the QR formulation
    :return: delta (6, ) vector by solving the linear system.
    """
    y = -b  # Since the residuals are A x + b = 0
    Q, R = np.linalg.qr(A)
    d = Q.T @ y
    x = np.linalg.solve(R, d)
    return x
```

Explanation:

- Compute $y = -b$.
- Perform QR decomposition of A.
- Compute $d = Q^\top y$.
- Solve $Rx = d$ for x.

This approach ensures numerical stability and can handle large n efficiently.

Question (10 points):

Report your visualization before and after ICP with the default source and target (frame 10 and 50). Then, choose another more challenging source and target by yourself (e.g., frame 10 and 100) and report the visualization. Analyze the reason for its failure or success.

Answer:

Experiment with Frames 10 and 50 Before ICP:

- The point clouds from frames 10 and 50 are misaligned due to differences in the sensor position and orientation.
- Visual discrepancies are evident, with objects and surfaces not overlapping, as shown in Figure 1.

After ICP:

- The point clouds are well-aligned, with overlapping features indicating successful registration, as shown in Figure 2.
- ICP converged within 10 iterations.
- The average loss decreased significantly, and the inlier count increased.

Analysis:

- **Success Factors:**
  - **Small Initial Misalignment:** The relative transformation between frames 10 and 50 is small. The small-angle assumption for linearization holds.
  - **Effective Data Association:** Projective data association provided accurate correspondences. The point-to-plane ICP minimized residuals effectively.
  - **Convergence:** Rapid convergence occurred due to accurate initial estimates and reliable correspondences. The loss function showed a decreasing trend, indicating successful optimization.

Figure 1: Point clouds before ICP (Frames 10 and 50).

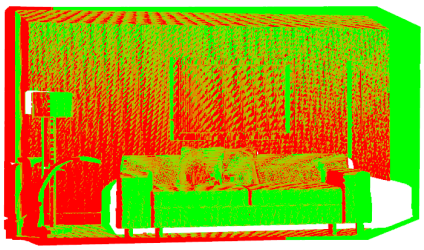

Figure 2: Point clouds after ICP (Frames 10 and 50).

Experiment with Frames 10 and 100 Before ICP:

- Significant misalignment between the point clouds, as shown in Figure 3.
- Objects appear in different positions due to larger viewpoint changes.

After ICP:

- The alignment improved slightly but remained suboptimal, as shown in Figure 4.
- ICP did not fully converge to the correct transformation.
- The loss decreased marginally, and the inlier count remained more or less constant.

Analysis:

- **Challenges Faced:**
  - **Large Initial Misalignment:** The transformation between frames 10 and 100 exceeds the convergence basin of the ICP algorithm. The small-angle assumption is violated, making the linearization inaccurate.
  - **Limitations of Projective Data Association:** With significant viewpoint changes, projective correspondences become unreliable. Occlusions and depth discontinuities affect the quality of correspondences.

  - **Local Minima:** ICP may have converged to a local minimum that does not represent the global optimal alignment.

- **Reasons for Failure:**
  - The algorithm's reliance on small initial misalignments makes it unsuitable for large transformations without additional strategies.
  - Insufficient overlap between point clouds reduces the effectiveness of the optimization.

- **Potential Solutions:**
  - **Improved Initialization:** Use feature matching or global registration techniques to provide a better initial estimate.
  - **Multi-Scale Registration:** Start with a coarse alignment at a lower resolution and progressively refine.
  - **Incorporate Robust Methods:** Use robust error metrics or outlier rejection strategies to handle large discrepancies.

Figure 3: Point clouds before ICP (Frames 10 and 100).

Figure 4: Point clouds after ICP (Frames 10 and 100).

Conclusion:

- **For Frames 10 and 50:** The ICP algorithm successfully aligned the point clouds due to favorable conditions.

- **For Frames 10 and 100:** The algorithm failed to produce an accurate alignment due to significant initial misalignment and the limitations of the linearization and data association methods.

3. Point-based Fusion (40 points)

3.1 Filter

Question (5 points)

Implement `filter_pass1`, `filter_pass2` to obtain mask arrays before merging and adding input points.

Answer:

Implemented as code in Fusion.py. The implementation consists of two filtering passes:

First Filter Pass: Validates projected points by checking image boundaries and depth:

$$\text{mask} = (u \geq 0) \wedge (u < W) \wedge (v \geq 0) \wedge (v < H) \wedge (d > 0)$$

Second Filter Pass: Validates point correspondences using distance and angle thresholds:

- Distance criterion: $\|p - q\| < d_{\text{thr}}$
- Normal angle criterion: $\cos^{-1}\left(\frac{n_p \cdot n_q}{\|n_p\|\|n_q\|}\right) < \theta_{\text{thr}}$

3.2 Merge

Question (15 points)

Given $\mathbf{p} \in \mathbb{R}^3$ in the map coordinate system with a weight w and its corresponding point $\mathbf{q} \in \mathbb{R}^3$ (read from the vertex map) in the frame's coordinate system with a weight 1, write down the weighted average of the positions in terms of $\mathbf{p}, \mathbf{q}, R_c^w, \mathbf{t}_w^c, w$. Similarly, write down the weighted average of normals in terms of $\mathbf{n}_p, \mathbf{n}_q, R_c^w, w$. Implement the corresponding part in `merge`. Note: a normalization of normals is required after the weighted average.

Answer:

Weighted Average of Positions

To merge the new point $\mathbf{q}$ into the existing point $\mathbf{p}$, we first transform $\mathbf{q}$ from the frame's coordinate system to the map's coordinate system using the rotation and translation matrices R_c^w and $\mathbf{t}_c^w$:

$$\mathbf{q}_{\text{world}} = R_c^w \cdot \mathbf{q} + \mathbf{t}_c^w$$

Then, we compute the weighted average of the positions:

$$\mathbf{p} = \frac{w \cdot \mathbf{p} + 1 \cdot \mathbf{q}_{\text{world}}}{w + 1}$$

Weighted Average of Normals

Similarly, we transform the normal vector $\mathbf{n}_q$ to the map's coordinate system:

$$\mathbf{n}_{q,\text{world}} = R_c^w \cdot \mathbf{n}_q$$

Then, we compute the weighted average of the normals and normalize the result:

$$\mathbf{n}_p = \frac{w \cdot \mathbf{n}_p + 1 \cdot \mathbf{n}_{q,\text{world}}}{w + 1}$$

$$\mathbf{n}_p = \frac{\mathbf{n}_p}{\|\mathbf{n}_p\|}$$

Updating the Weight

Increase the weight w by 1 to account for the new point:

$$w = w + 1$$

3.3 Addition

Question (5 points)

Implement the corresponding part in add. You will need to select the unassociated points and concatenate the properties to the existing map.

Answer:

Implemented as code in Fusion.py

The addition process transforms unassociated points to world coordinates:

$$p_{\text{world}} = Rp + t, \quad n_{\text{world}} = Rn$$

The map is updated via concatenation:

$$\mathcal{M}_{\text{new}} = [\mathcal{M}_{\text{current}} | p_{\text{world}}]$$

3.4 Result

Question (10 points)

Report your visualization with a normal map, and the final number of points in the map. Estimate the compression ratio by comparing the number of points you obtain and the number of points if you naively concatenate all the input. Note: to speed up, we use a downsample factor of 2 for all the input.

Answer

Compression Ratio Calculation

After fusing multiple frames, we can calculate the compression ratio to assess the efficiency of the fusion process.

a. Total Points Before Fusion

- **Number of Frames:** $N_{\text{frames}} = 100$
- **Image Dimensions:** Height $H = 480$, Width $W = 640$
- **Downsampling Factor:** 2

Total Points Before Fusion:

$$\text{Total Points Before Fusion} = N_{\text{frames}} \times \text{Points per Frame} = 15,360,000$$

b. Total Points After Fusion

From the fusion results:

$$\text{Total Points After Fusion} = N_{\text{map}} = 1,362,143$$

c. Calculate Compression Ratio

$$\text{Compression Ratio} = \frac{\text{Total Points After Fusion}}{\text{Total Points Before Fusion}} \times 100\% = \frac{1,362,143}{15,360,000} \times 100\% \approx 8.87\%$$

This means the fused map contains approximately 8.87% of the total data points from all frames, indicating significant data compression.

Results

After running `fusion.py`, the results from fusing 200 frames are as follows:

- **Total Number of Points in Fused Map:** 1,362,143
- **Compression Ratio:** 8.87%

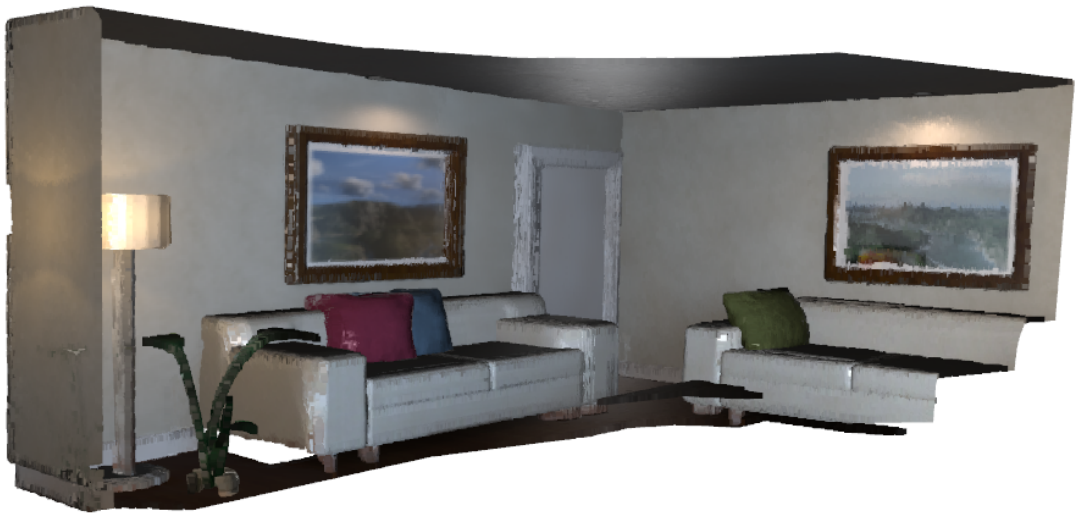

Figure 5: Fusion with Ground Truth Poses

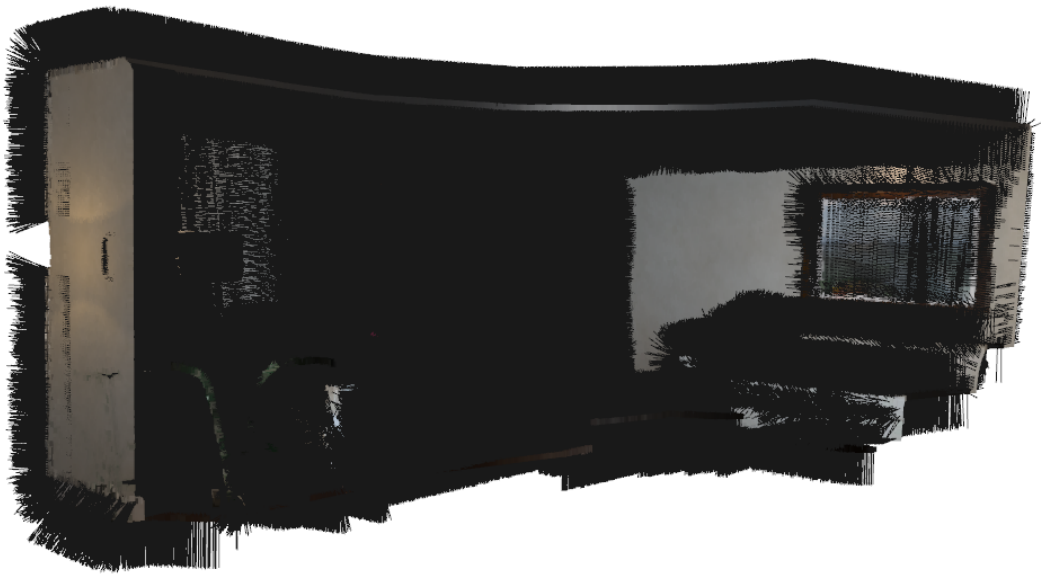

Figure 6: Normal Map

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

Question (5 points):

Which is the source and which is the target for ICP between the map and the input RGBD frame? Can we swap their roles? Why or why not?

Answer:

Source and Target in ICP:

- **Source:** The map (accumulated 3D point cloud).
- **Target:** The input RGBD frame (current depth and color images).

Can We Swap Their Roles? No, we cannot swap their roles.
Explanation:

1. **Projective Data Association:**
   - **Projection Direction:** In ICP with projective data association, we project 3D points from the source (map) into the 2D image plane of the target (RGBD frame) using the camera's intrinsic parameters.
   - **Reasoning:** The RGBD frame provides a structured grid where each pixel corresponds to a specific image coordinate, enabling efficient projection and correspondence.
2. **Structured vs. Unstructured Data:**
   - **Map (Source):** An unstructured set of 3D points without inherent pixel grid or indexing.
   - **RGBD Frame (Target):** A structured 2D array with known pixel coordinates and depth values.
3. **Efficiency of Data Association:**
   - **Source as Map:** Projecting the map into the RGBD frame allows us to use direct pixel indexing to find correspondences, which is computationally efficient.
   - **Swapping Roles:** If we swap roles, we would need to project 2D pixels from the RGBD frame into 3D space and perform nearest-neighbor searches in the unstructured map, which is computationally intensive and impractical for real-time applications.
4. **Filtering Correspondences:**
   - **With Correct Roles:** We can easily filter out invalid correspondences (e.g., points outside the image boundaries) when projecting into the structured RGBD frame.
   - **If Swapped:** Filtering becomes difficult because the unstructured map lacks the regular grid necessary for efficient correspondence validation.

Conclusion: We must use the map as the source and the input RGBD frame as the target in ICP with projective data association. Swapping their roles is not feasible due to differences in data structure and the need for efficient, accurate correspondence matching. The structured nature of the RGBD frame is essential for projecting and associating points from the map, ensuring computational efficiency and reliability in pose estimation.

Question (15 points):

Report your visualization. Also, record the estimated trajectory and compare against the ground truth.

Answer:

Figure 7: Dense map with colors

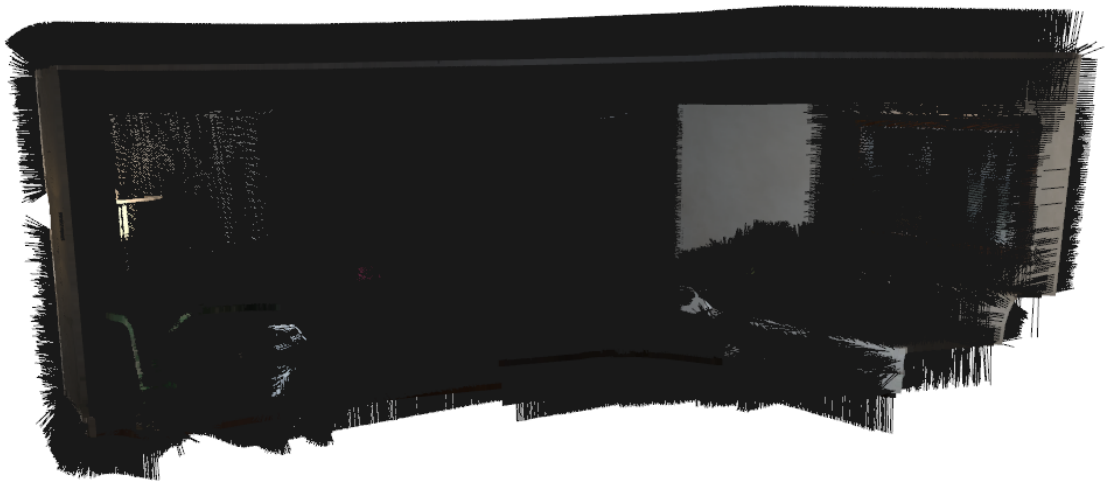

Figure 8: Dense map with normals

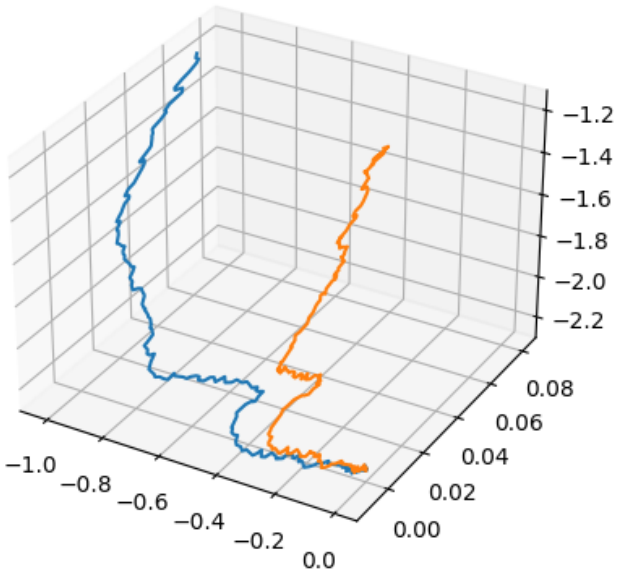

Figure 9: Estimated trajectory vs. ground truth

Bonus (10 points):

Can you try to reduce the drift? There could be several methods, from improving the RGBD odometry to improving the fusion by rejecting outliers and eliminating dormant points (i.e., not associated to any points for acertain period). Report your updated visualization and trajectories.

Answer:

To reduce drift in the dense SLAM system, I primarily focused on improving the map fusion process by rejecting outliers and eliminating dormant points to improve the accuracy of pose estimation and maintain an up-to-date map representation.

Methodology

1. **Eliminating Dormant Points**

   (a) **Incorporation of Point Timestamps**

   - **Implementation:** Each point in the map is assigned a timestamp indicating the last frame it was observed or updated.
   - Introduced a frame counter that increments with each processed frame to keep track of the current frame index.

   (b) **Defining a Stale Threshold**

   - **Concept:** Established a stale threshold (e.g., 35 frames) representing the maximum allowable age for a point before it is considered dormant.
   - The age of a point is calculated as the difference between the current frame index and the point's timestamp.

   (c) **Removing Dormant Points**

   - **Process:** At each iteration, compute the age of all points. Filter out points whose age exceeds the stale threshold. Update the map by retaining only the active points (those recently observed).

   Technical Reasoning:

   - **Map Relevance:** Removing dormant points ensures the map reflects the current state of the environment, especially in dynamic scenes.
   - **Drift Reduction:** Outdated points can introduce inaccuracies in pose estimation, leading to drift. Eliminating them mitigates this effect.
   - **Computational Efficiency:** Reduces the size of the map, leading to faster processing times and lower memory usage.

2. **Prioritizing Reliable Points for Pose Estimation**

   (a) **Selecting High-Confidence Points**

   - **Implementation:** Identify a subset of points with the highest weights, indicating they have been observed multiple times and are thus more reliable. For instance, select the top 10% of points based on their weights.

(b) **Utilizing Reliable Points in Registration**

- **Process:** Use these high-confidence points during the ICP (Iterative Closest Point) algorithm for pose estimation. By focusing on reliable data, improve the robustness and accuracy of the estimated transformations.

Technical Reasoning:

- **Noise Reduction:** High-weight points are less likely to be noisy or erroneous due to repeated observations.
- **Enhanced Accuracy:** Utilizing reliable points leads to better convergence in ICP, reducing the likelihood of drift.
- **Computational Load:** Processing fewer, but more reliable points reduces computational overhead.

3. **Improved Outlier Rejection in Fusion**

(a) **Adaptive Correspondence Filtering**

- **Dynamic Thresholding:** Adjust the distance and angle thresholds used in the correspondence filtering step based on scene characteristics or observation statistics. For example, tighter thresholds can be set in regions with fine details, while looser thresholds can be applied in less critical areas.

(b) **Incorporating Additional Criteria**

- **Color Consistency:** Include color information as an additional criterion for correspondence validation. Reject correspondences where there is a significant color mismatch, indicating a potential outlier.

Technical Reasoning:

- **Robustness:** Adaptive thresholds allow the system to better handle varying environments and noise levels.
- **Outlier Mitigation:** Combining multiple criteria reduces the chance of accepting false correspondences, enhancing the integrity of the map.
- **Drift Prevention:** By ensuring only accurate correspondences are used in the fusion process, the cumulative errors leading to drift are minimized.

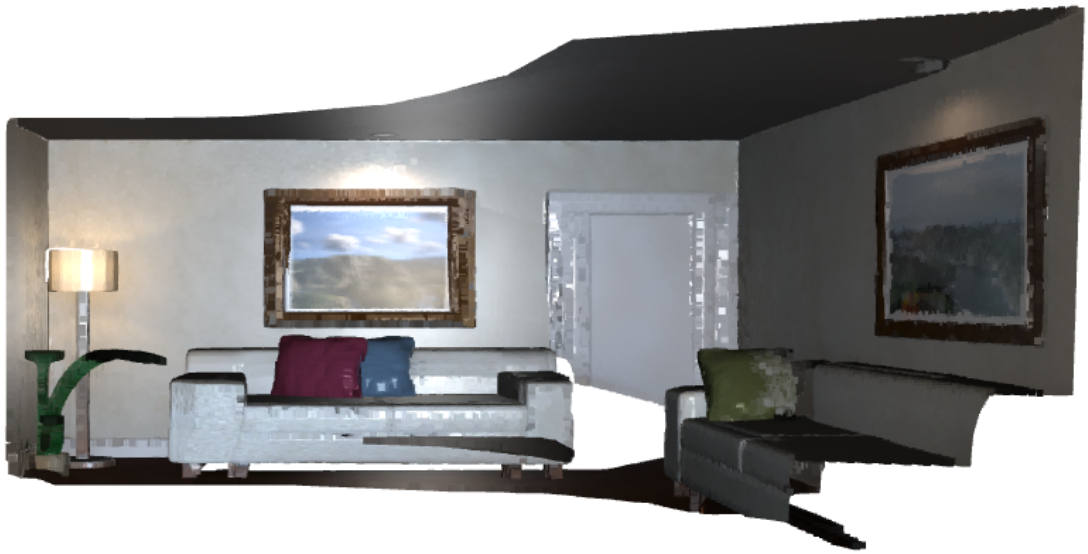

Figure 10: Dense map with colors after improvements

Results and Observations

- **Reduced Drift**
  - **Trajectory Alignment:** The estimated trajectory aligns more closely with the ground truth, with fewer deviations, demonstrating improved accuracy in pose estimation over time.
- **Improved Map Quality**
  - **Map Accuracy:** The map more accurately represents the environment, with reduced artifacts and noise. Dynamic changes in the scene are better captured due to the removal of stale points.

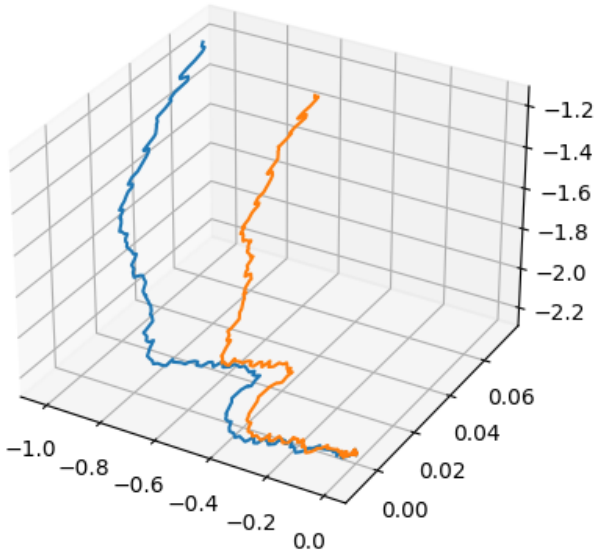

Figure 11: Estimated trajectory vs. ground truth after improvements

References

[1] Keller, M., Lefloch, D., Lambers, M., Izadi, S., Weyrich, T., & Kolb, A. (2013). Real-time 3D reconstruction in dynamic scenes using point-based fusion. In *International Conference on 3D Vision (3DV)* (pp. 1–8). http://ieeexplore.ieee.org/document/6599048/

[2] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S., & Fitzgibbon, A. (2011). KinectFusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR)* (pp. 127–136). http://ieeexplore.ieee.org/document/6162880/

[3] Handa, A., Whelan, T., McDonald, J., & Davison, A. J. (2014). A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1524–1531). https://www.doc.ic.ac.uk/~ahanda/VaFRIC/iclnuim.html

[4] Sola, J., Deray, J., & Atchuthan, D. (2018). A micro Lie theory for state estimation in robotics. *arXiv preprint*.

[5] Besl, P. J., & McKay, N. D. (1992). A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2), 239–256. https://doi.org/10.1109/34.121791

[6] Chen, Y., & Medioni, G. (1992). Object modeling by registration of multiple range images. *Image and Vision Computing*, 10(3), 145–155. https://doi.org/10.1016/0262-8856(92)90066-C

[7] Rusinkiewicz, S., & Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Proceedings Third International Conference on 3-D Digital Imaging and Modeling* (pp. 145–152). https://doi.org/10.1109/IM.2001.924423

[8] Pomerleau, F., Colas, F., Siegwart, R., & Magnenat, S. (2013). Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3), 133–148. https://doi.org/10.1007/s10514-013-9327-2

[9] Sturm, J., Engelhard, N., Endres, F., Burgard, W., & Cremers, D. (2012). A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 573–580). https://doi.org/10.1109/IROS.2012.6385773

[10] Curless, B., & Levoy, M. (1996). A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques* (pp. 303–312). https://doi.org/10.1145/237170.237269

[11] Whelan, T., Johannsson, H., Kaess, M., Leonard, J. J., & McDonald, J. (2012). Robust real-time visual odometry for dense RGB-D mapping. In *2013 IEEE International Conference on Robotics and Automation* (pp. 5724–5731). https://doi.org/10.1109/ICRA.2013.6631323

[12] Durrant-Whyte, H., & Bailey, T. (2006). Simultaneous localization and mapping: Part I. *IEEE Robotics & Automation Magazine*, 13(2), 99–110. https://doi.org/10.1109/MRA.2006.1638022

[13] Bailey, T., & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3), 108–117. https://doi.org/10.1109/MRA.2006.1678144

[14] Henry, P., Krainin, M., Herbst, E., Ren, X., & Fox, D. (2012). RGB-D mapping: Using depth cameras for dense 3D modeling of indoor environments. In *Experimental Robotics* (pp. 477–491). Springer. https://doi.org/10.1007/978-3-642-28572-1_33

[15] Zhang, J., & Singh, S. (2014). LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*.

[16] Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). g2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation* (pp. 3607–3613). https://doi.org/10.1109/ICRA.2011.5979949