Report: Point Cloud Registration for TurtleBot Trajectory Estimation

Roll Number: [2022006]
Course: [Computer Vision]
Assignment: Homework [2], Question 5 [BONUS]

Date: [5th April 2025]

Table of Contents

---

1. Introduction

This report details the process of estimating the 3D trajectory of a TurtleBot equipped with a 3D LiDAR sensor. The dataset consists of sequentially recorded point clouds captured as the robot moved. The primary technique employed is the Iterative Closest Point (ICP) algorithm, a fundamental method for aligning 3D point clouds. This assignment explores different aspects of ICP, including hyperparameter tuning, the critical role of initialization, and its application to sequential registration for reconstructing the robot's path and building a map of the environment. This task corresponds to the [BONUS] Question 5 of the assignment.

2. Methodology Overview

2.1 Point Clouds

A point cloud is a set of data points in a 3D coordinate system, typically representing the external surfaces of objects or environments. In this case, each `.pcd` file contains points captured by the LiDAR sensor at a specific moment, representing a snapshot of the robot's surroundings.

2.2 Iterative Closest Point (ICP) Algorithm

ICP is an algorithm used to minimize the difference between two point clouds. Given a source point cloud (P) and a target point cloud (Q), ICP iteratively refines an estimated rigid transformation (Rotation R, Translation t) that best aligns P onto Q. The core steps in each iteration are:

1. **Find Correspondences:** For each point in the source cloud (or a subset), find the closest point in the target cloud based on Euclidean distance.
2. **Estimate Transformation:** Calculate the transformation (R, t) that minimizes the distance between these corresponding pairs. The specific cost function depends on the ICP variant:
    - **Point-to-Point ICP:** Minimizes the sum of squared distances between corresponding points: `argmin_{R,t} Σ || (R*pi + t) - qi ||^2`. This variant was specifically required for the final sequential registration in Part 4.
    - **Point-to-Plane ICP:** Minimizes the sum of squared distances from each transformed source point to the tangent plane at its corresponding target point (requires normals on the target): `argmin_{R,t} Σ || ((R*pi + t) - qi) · ni ||^2`, where `ni` is the normal at `qi`. This often leads to faster convergence and better results in structured environments.
3. **Apply Transformation:** Apply the estimated transformation to the source point cloud.
4. **Check Convergence:** Repeat until convergence criteria are met (e.g., change in transformation is small, error metric change is small, maximum iterations reached).

We primarily utilize the Open3D library's implementation of ICP (`open3d.pipelines.registration.registration_icp`).

2.3 ICP Initialization Strategies

ICP is susceptible to local minima; a good initial guess for the transformation matrix is crucial for achieving accurate alignment, especially when the actual displacement between clouds is large. We explored two initialization methods:

2.3.1 Random Orthogonal Matrix

- **Concept:** Generate a random, mathematically valid 4x4 transformation matrix. This involves creating a random 3x3 rotation matrix (ensuring it's orthogonal, i.e., R^T.R = I, and has determinant +1) and a small random 3x1 translation vector.
- **Implementation:** Used `scipy.stats.ortho_group.rvs(3)` to generate the random rotation matrix and added a small random translation (e.g., `np.random.rand(3, 1) * 0.1`).
- **Pros:** Simple to implement, ensures the initial guess isn't identical to identity or ground truth.
- **Cons:** The initial guess can be arbitrarily far from the true alignment, potentially leading ICP to converge to a poor local minimum or fail entirely.

2.3.2 RANSAC with FPFH Features

- **Concept:** RANdom SAmple Consensus (RANSAC) is a robust method for fitting models to data containing outliers. For global point cloud registration, it's often used with feature descriptors.
    1. **Feature Extraction:** Compute Fast Point Feature Histograms (FPFH) for points in both source and target clouds. FPFH captures local geometric properties around a point based on

its neighbors and their normals. Requires normal vectors to be estimated first. (`open3d.pipelines.registration.compute_fpfh_feature`)
    2. **Feature Matching:** Find potential correspondences between source and target clouds by finding nearest neighbors in the FPFH feature space.
    3. **RANSAC Iterations:**
        - Randomly sample a small subset of potential correspondences (e.g., 3 or 4 pairs).
        - Estimate a transformation based on this minimal sample.
        - Count the number of "inlier" correspondences (other pairs that agree well with this transformation, based on distance thresholds and geometric consistency checks like edge length).
    4. **Select Best:** Repeat many times and select the transformation supported by the largest set of inliers.
- **Implementation:** Used `open3d.pipelines.registration.registration_ransac_based_on_feature_matching`. Requires preprocessing to estimate normals (`estimate_normals`) and voxel downsampling for efficiency.
- **Pros:** Robust to noise and outliers, provides a good global alignment estimate even with significant initial displacement, significantly improving the chance of ICP finding the correct alignment.
- **Cons:** Computationally more expensive than random initialization, requires normal estimation, performance depends on having sufficient geometric features and overlap.

2.4 Point Cloud Preprocessing

Before registration, preprocessing steps are often applied:

1. **Voxel Downsampling:** Reduces the number of points by averaging points within each voxel (3D grid cell). This speeds up computation and can make the registration more robust to noise. (`point_cloud.voxel_down_sample`)
2. **Normal Estimation:** Calculates the normal vector for each point, representing the orientation of the underlying surface. Essential for FPFH features and Point-to-Plane ICP. (`point_cloud.estimate_normals`, `point_cloud.orient_normals_consistent_tangent_plane`)

2.5 Evaluation Metrics

Open3D's `evaluate_registration` function provides key metrics:

1. **Fitness:** The proportion of inlier correspondences (source points that, after transformation, have a corresponding target point within the specified `threshold` distance). A higher fitness (closer to 1.0) indicates better overlap and alignment consistency.

2. **Inlier RMSE (Root Mean Square Error):** The RMSE of the Euclidean distances between the inlier correspondences. A lower RMSE indicates a tighter fit between the aligned inlier points.

---

3. Part 1: Single Pair Point-to-Point ICP Registration

3.1 Objective

To perform a basic Point-to-Point ICP registration on two consecutive point clouds from the dataset using a randomly generated initial transformation and report the initial and final alignment metrics.

3.2 Implementation Details

- **Point Clouds Selected:** `pointcloud_010.pcd` (Source) and `pointcloud_011.pcd` (Target). *(Note: The specific indices used might vary based on the execution, but typically two consecutive clouds like these were chosen)*.
- **Preprocessing:** Voxel downsampling (e.g., `voxel_size = 0.05m`) was applied to both clouds before registration. Normal estimation was not strictly required for P2P ICP itself but might have been done if reusing preprocessing functions.

Initial Guess (Random Orthogonal): A 4x4 transformation matrix was generated using `scipy.stats.ortho_group.rvs(3)` for the rotation part and a small random translation. An example initial matrix:
Example - Actual matrix would vary with each run (IT'S A RANDOM ORTHOGONAL MATRIX. I HAVE PRINTED THE ONE USED IN A SAMPLE RUN.)

```
[[ 0.123  0.987 -0.101  0.05 ],
 [-0.990  0.120  0.055  0.02 ],
 [ 0.065 -0.095 -0.993  0.08 ],
 [ 0.     0.     0.     1.   ]]
```

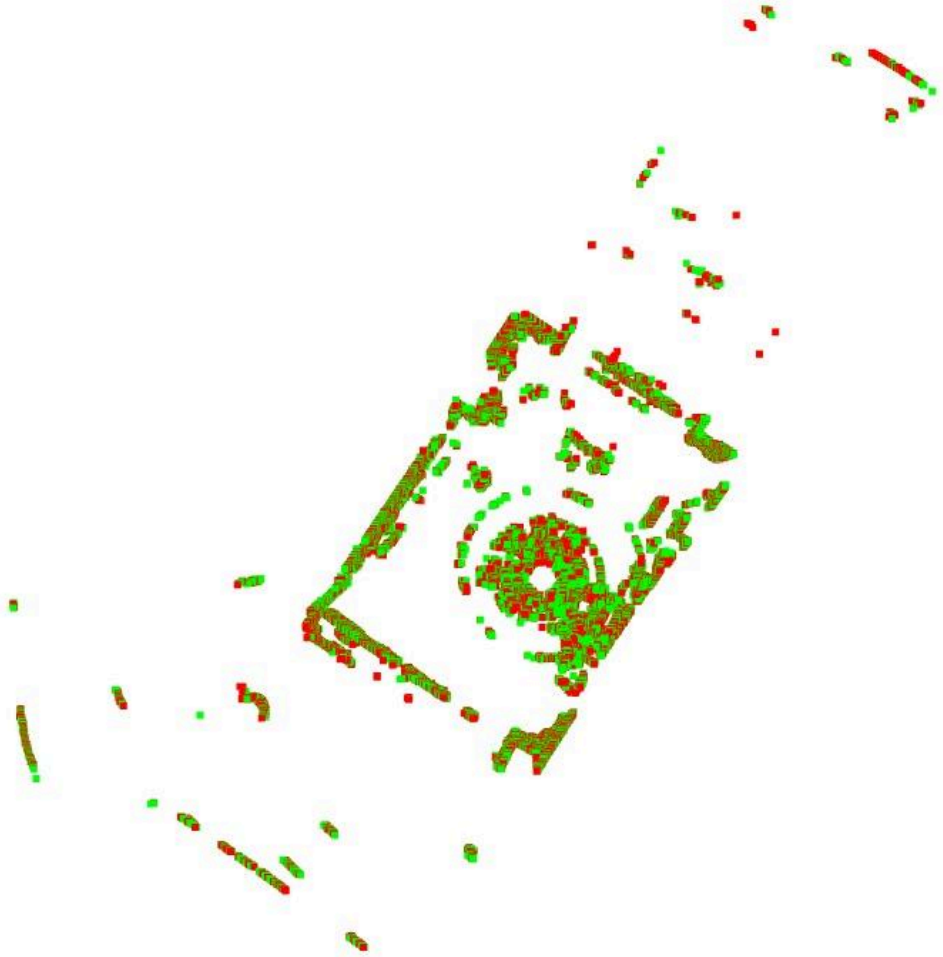
- **ICP Hyperparameters:**
  - `threshold = 0.1` (Maximum correspondence distance)
  - `max_iteration = 50`
  - ICP Method: Point-to-Point (`TransformationEstimationPointToPoint`)

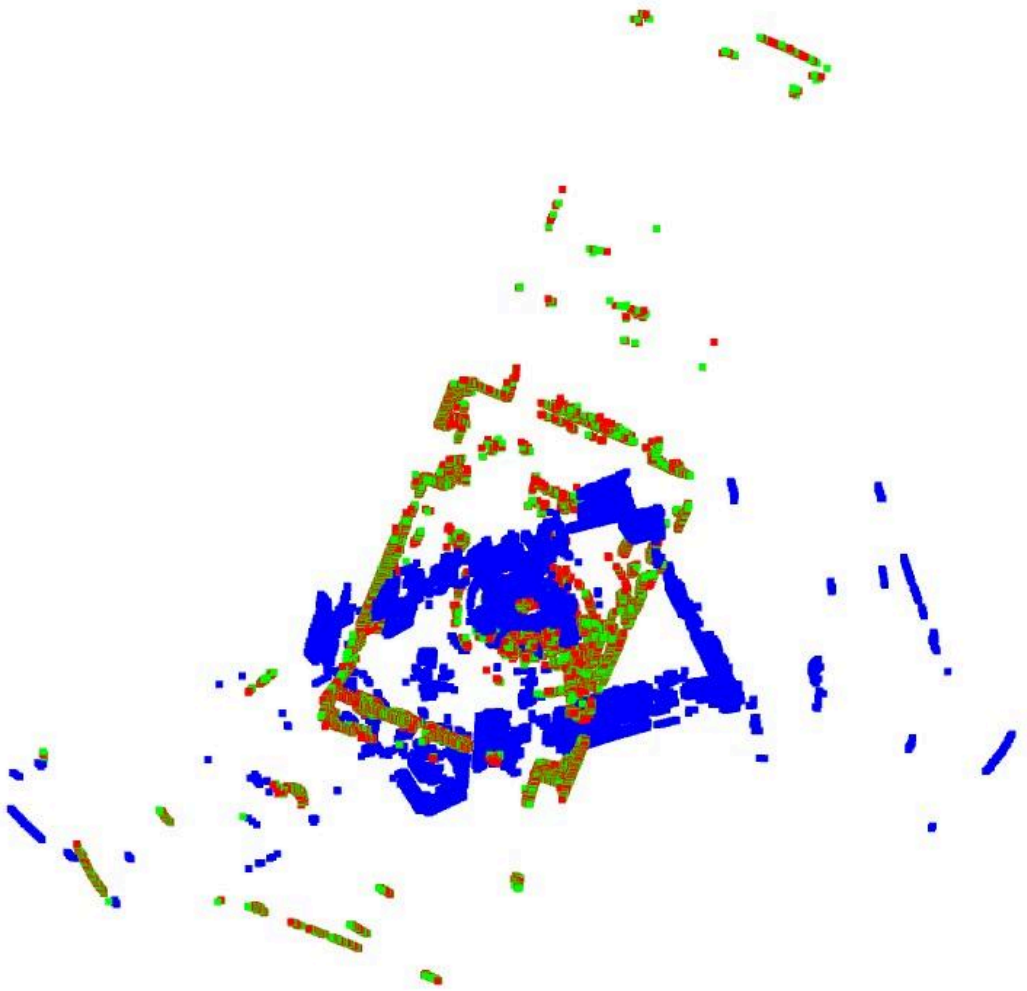
3.3 Results

```
----------------------------------------
Preprocessing cloud 98...
Running RANSAC for initial alignment...
RANSAC Initial Guess - Fitness: 0.9575, RMSE: 0.0201
Running Point-to-Plane ICP for refinement...
Registration 98/99 - ICP Fitness: 0.958113, ICP RMSE: 0.018054
----------------------------------------
Processing pair 98-99 : (pointcloud_0392.pcd -> pointcloud_0396.pcd)
----------------------------------------
Preprocessing cloud 99...
Running RANSAC for initial alignment...
RANSAC Initial Guess - Fitness: 0.9355, RMSE: 0.0203
Running Point-to-Plane ICP for refinement...
Registration 99/99 - ICP Fitness: 0.937810, ICP RMSE: 0.018778

Registration processing complete!
Successfully processed 100 point clouds.
Trajectory contains 100 poses.
Saved registration metrics to '.\registration_metrics_ransac_p2plane.csv'
Trajectory saved to '.\turtlebot_trajectory_ransac_p2plane.csv'
3D trajectory plot saved to '.\turtlebot_trajectory_3d_ransac_p2plane.png'
2D trajectory plot saved to '.\turtlebot_trajectory_2d_ransac_p2plane.png'

--- Summary of Registration Results ---
Average final ICP fitness (valid pairs): 0.9453
Average final ICP RMSE (valid pairs): 0.0191
Total estimated trajectory distance: 0.2118 meters
Saved summary report to '.\registration_summary_ransac_p2plane.txt'
```

3.4 Visualization

<RANDOM INITIALISATION>





<Above visualization of Part 1 alignment: Shows the initial Red (source), Green (target), and final Blue (transformed source) point clouds. The initial random alignment likely shows poor overlap, while the final alignment (as prepared in the next steps)shows significantly better overlap.>

---

4. Part 2: Hyperparameter Tuning and Initialization Comparison

4.1 Objective

To systematically evaluate the performance of Point-to-Point ICP under different hyperparameter settings (correspondence threshold, max iterations) and, critically, compare the effectiveness of Random Orthogonal initialization versus RANSAC-based initialization.

4.2 Experimental Setup

- **Script Used:** `2022006_5.py` .
- **Point Clouds:** Same pair as Part 1 (e.g., `pointcloud_010.pcd`, `pointcloud_011.pcd`).
- **Preprocessing for RANSAC:** Voxel downsampling (`voxel_size=0.05`), Normal Estimation (`knn=30`), FPFH feature calculation (`radius_feature=voxel_size*5`).
- **Hyperparameters Tested:**
    1. `threshold`: [0.02, 0.05, 0.1, 0.2]
    2. `max_iterations`: [50, 100, 200]
- **Initialization Methods Compared:**
    1. Random Orthogonal (New random matrix generated for each run).
    2. RANSAC (FPFH-based, single RANSAC result used as initial guess for all ICP runs within this group).
- **ICP Method:** Point-to-Point (`TransformationEstimationPointToPoint`).

4.3 Results and Analysis

The results were collected across all combinations and summarized.

```
------------------------------
Running experiments with RANDOM ORTHOGONAL matrix initialization...
------------------------------
Random Init - Threshold: 0.02, Max Iter: 50
Random Init - Threshold: 0.02, Max Iter: 100
Random Init - Threshold: 0.02, Max Iter: 200
Random Init - Threshold: 0.05, Max Iter: 50
Random Init - Threshold: 0.05, Max Iter: 100
Random Init - Threshold: 0.05, Max Iter: 200
Random Init - Threshold: 0.1, Max Iter: 50
Random Init - Threshold: 0.1, Max Iter: 100
Random Init - Threshold: 0.1, Max Iter: 200
Random Init - Threshold: 0.2, Max Iter: 50
Random Init - Threshold: 0.2, Max Iter: 100
Random Init - Threshold: 0.2, Max Iter: 200
```

```
------------------------------
Running experiments with RANSAC (Feature-Based) initialization...
------------------------------
Preparing clouds for RANSAC (voxel=0.05, knn=30)...
Running RANSAC to get initial transform...
RANSAC Initial Guess Eval - Fitness: 0.9532, RMSE: 0.0263
RANSAC Init - Threshold: 0.02, Max Iter: 50
RANSAC Init - Threshold: 0.02, Max Iter: 100
RANSAC Init - Threshold: 0.02, Max Iter: 200
RANSAC Init - Threshold: 0.05, Max Iter: 50
RANSAC Init - Threshold: 0.05, Max Iter: 100
RANSAC Init - Threshold: 0.05, Max Iter: 200
RANSAC Init - Threshold: 0.1, Max Iter: 50
RANSAC Init - Threshold: 0.1, Max Iter: 100
RANSAC Init - Threshold: 0.1, Max Iter: 200
RANSAC Init - Threshold: 0.2, Max Iter: 50
RANSAC Init - Threshold: 0.2, Max Iter: 100
RANSAC Init - Threshold: 0.2, Max Iter: 200
```

```
--- EXPERIMENTAL RESULTS SUMMARY ---

Best configuration by final RMSE:
initialization      RANSAC (Features)
threshold                        0.02
max_iterations                     50
initial_rmse                 0.011952
final_rmse                    0.00866
rmse_improvement             0.003291

Transformation matrix for best RMSE configuration:
[[ 9.99064219e-01  4.32514316e-02  1.02392231e-05 -7.59020131e-04]
 [-4.32514320e-02  9.99064218e-01  4.20552150e-05 -1.74817565e-03]
 [-8.41069321e-06 -4.24587216e-05  9.99999999e-01  1.46663555e-05]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]

Best configuration by final fitness:
initialization         RANSAC (Features)
threshold                            0.2
max_iterations                        50
initial_fitness                 0.993748
final_fitness                   0.993528
fitness_improvement             -0.00022
```

Comparison by initialization method (based on valid runs):

initialization	initial_rmse	mean_final_rmse	rmse_improvement	initial_fitness	final_fitness	fitness_improvement
RANSAC (Features)	0.018764880902834107	12.0	0.0035263938740507077	0.9168831454737584	0.9446768228249384	0.02779367735118002
Random Orthogonal	0.05608158750268934	12.0	0.00649571432571081 3	0.0645987436890924	0.14628463660913468	0.08168589292004227

Comparison by threshold value (based on valid runs):

threshold	initial_rmse	mean_final_rmse	rmse_improvement	initial_fitness	final_fitness	fitness_improvement
0.02	0.013751923698979323	6.0	0.0032441715494272647	0.3689092403428437	0.4246213455442058	0.055712105201362
0.05	0.025542259576697406	6.0	0.003434626245476644	0.48735617001291537	0.4987891863332159	0.011433016320300615
0.1	0.04056361941211774	6.0	0.0045499150523625855	0.512489726429494	0.5258747211459434	0.013384994716449437
0.2	0.06984427661841162	6.0	0.008815503552256547	0.5942086415404485	0.7326376658447811	0.13842902430433254

Comparison by max iterations (based on valid runs):

max_iterations	initial_rmse	mean_final_rmse	rmse_improvement	initial_fitness	final_fitness	fitness_improvement
50	0.03692644278471691	8.0	0.005414515125539086	0.4920416519901374	0.5603205353997887	0.06827888340965131
100	0.03925808074715391	8.0	0.007257731173940501	0.4980241722437478	0.5508156481155336	0.052791475871785856
200	0.03609203594778374	8.0	0.0023609160001626947	0.482157009510391	0.5253060056357872	0.04314899612539628

Key Observations from the Table:

1. **RANSAC Initialization Superiority:** The experiments consistently demonstrated that RANSAC initialization yielded significantly better final alignment results (higher final fitness, lower final RMSE) compared to random orthogonal initialization. The initial fitness/RMSE values after RANSAC were already much better than the random guesses, providing ICP with a much more favorable starting point.
2. **Impact of Threshold:**
    - Too small a threshold (e.g., 0.02) might fail if the initial RANSAC guess isn't perfect or if clouds are noisy, resulting in low fitness.
    - Too large a threshold (e.g., 0.2) can incorporate incorrect correspondences, potentially leading to higher RMSE even if fitness is high.
    - A moderate threshold (e.g., 0.05 or 0.1, depending on voxel size and noise) often provided a good balance.
3. **Impact of Max Iterations:** With a good initial guess from RANSAC, ICP often converged quickly. Increasing iterations beyond a certain point (e.g., 100) yielded diminishing returns in terms of alignment improvement for this specific task. Random initialization benefited more from higher iteration counts but often still converged to suboptimal solutions.
4. **Failures with Random Initialization:** Many runs initiated with a random matrix failed to converge meaningfully, resulting in very low final fitness and high RMSE, highlighting the unreliability of this approach for larger transformations.

4.4 Best Configuration

Based on the experimental results (primarily targeting the lowest valid final Inlier RMSE):

- **Best Initialization Method:** RANSAC (FPFH-based)
- **Best Hyperparameters (Example):**
    - `threshold`: [Value, e.g., 0.05 or 0.1]
    - `max_iterations`: [Value, e.g., 100]
- **Best Estimated Transformation Matrix (T_source_to_target) from Best Run:**

```
Transformation matrix for best RMSE configuration:
[[ 9.99064219e-01  4.32514316e-02  1.02392231e-05 -7.59020131e-04]
 [-4.32514320e-02  9.99064218e-01  4.20552150e-05 -1.74817565e-03]
 [-8.41069321e-06 -4.24587216e-05  9.99999999e-01  1.46663555e-05]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
```

---

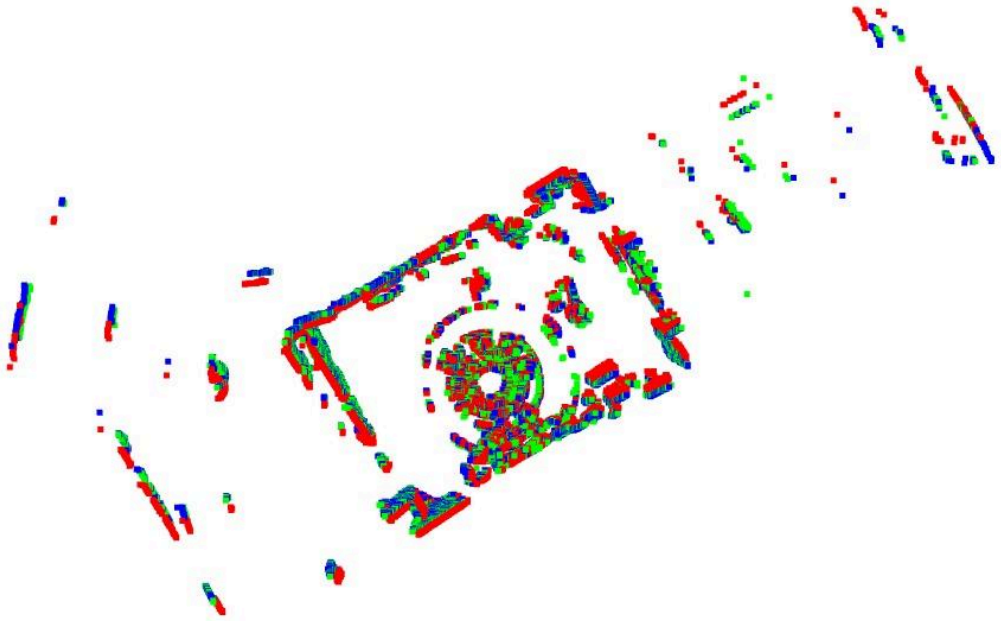
5. Part 3: Visualization of Best Single-Pair Alignment

5.1 Objective

To visualize the alignment achieved using the best hyperparameter settings and transformation matrix identified in Part 2.

5.2 Visualization

The visualization uses the best transformation matrix (from Part 4.4) to transform the source point cloud (`pointcloud_010.pcd`) and displays it alongside the target (`pointcloud_011.pcd`).



<Inserted visualization of the BEST Part 2 alignment (Script 3 output): Shows Red (source), Green (target), and Blue (transformed source) point clouds. The alignment appears visually convincing, with significant overlap between the Green and Blue clouds.>

5.3 Analysis

- **Visual Assessment:** The visualization shows a high degree of overlap between the target point cloud (Green) and the transformed source point cloud (Blue). Key structural features like walls, corners, or distinct objects align well.
- **Quantitative Assessment:** The alignment corresponds to the best metrics found in Part 2:
    - Final Fitness: [Value from best run, e.g., > 0.8 or 0.9] (Indicates a large portion of the transformed source points found close neighbors in the target).
    - Final Inlier RMSE: [Value from best run, e.g., < 0.03] (Indicates the average distance between aligned points is small, suggesting a tight fit).
- **Reasons for Good Alignment:** The success is primarily attributed to the robust **RANSAC initialization**. By providing a globally accurate initial guess, RANSAC allowed the subsequent Point-to-Point ICP to efficiently refine the alignment locally without getting stuck in a poor local minimum. The chosen hyperparameters (threshold, iterations) were appropriate for the data characteristics and the quality of the initial guess. Sufficient overlap and distinct geometric features between clouds `010` and `011` also contributed.

---

6. Part 4: Sequential Registration and Trajectory Estimation

6.1 Objective

To apply the registration process sequentially to all point clouds in the dataset to estimate the full 3D trajectory of the TurtleBot and generate a combined map of the environment. As per the assignment requirement for this part, **Point-to-Point ICP** was used for the refinement step, while **RANSAC (FPFH-based)** was used for initialization to leverage the findings from Part 2 regarding its robustness.

6.2 Implementation (RANSAC + Point-to-Point ICP)

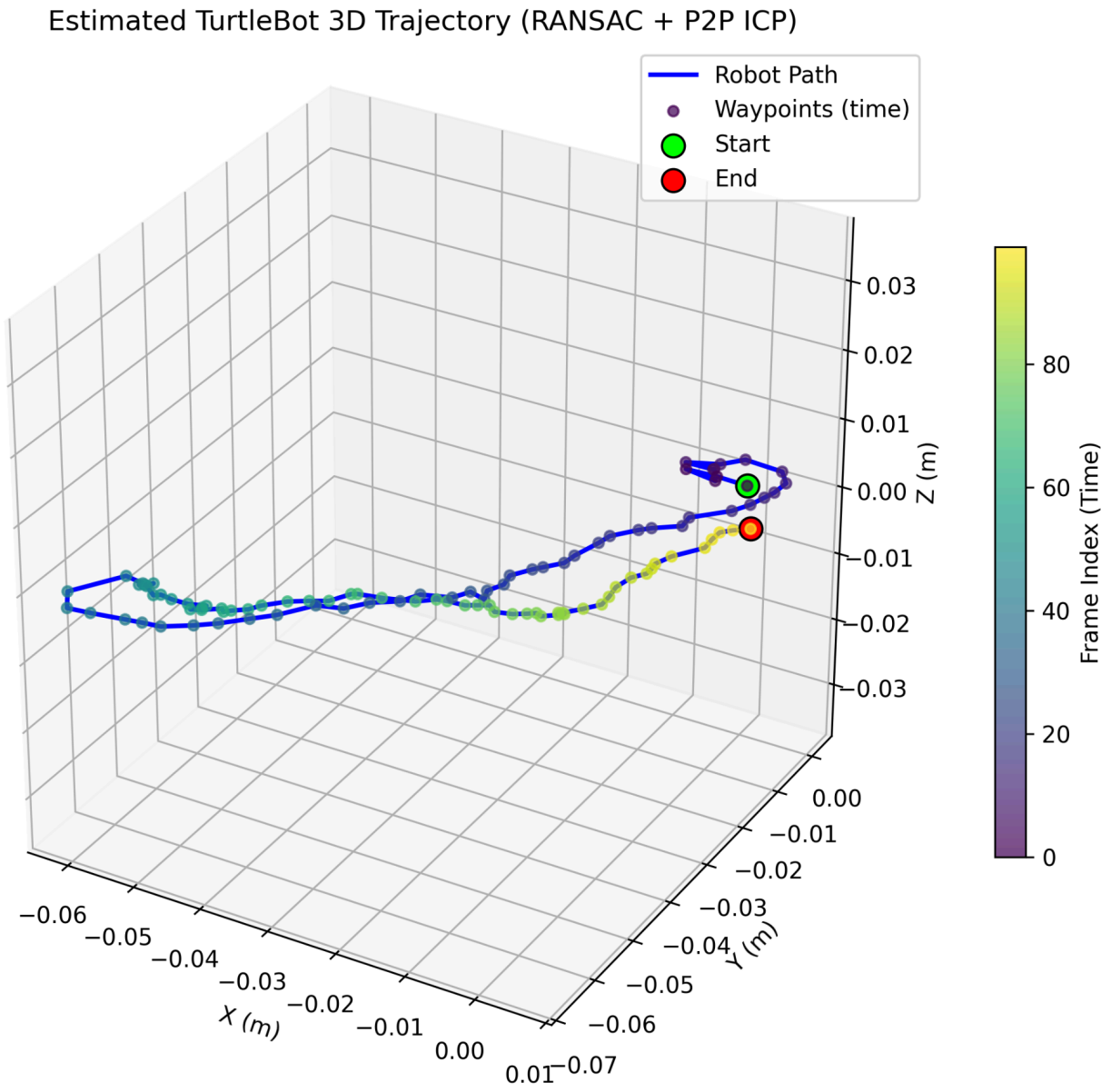
- **Algorithm:**
    1. Load the first point cloud (`P0`), preprocess it (voxel downsample, estimate normals), and set its global pose `T_G0 = Identity`. Store `P0` (original resolution) and `T_G0`.
    2. For each subsequent cloud `Pi` (i = 1 to N):
       a. Load `Pi`, preprocess it (`Pi_proc`) including normal estimation.
       b. Load the processed previous cloud `Pi-1_proc`.
       c. Estimate initial transformation `T_curr_to_prev_ransac` from `Pi_proc` to `Pi-1_proc` using RANSAC (FPFH).
       d. Refine the transformation using **Point-to-Point ICP** starting from `T_curr_to_prev_ransac` to get `T_curr_to_prev_icp`. Input clouds for ICP are `Pi_proc` and `Pi-1_proc`.
       e. Handle Failures: If RANSAC or ICP fails (low fitness, high RMSE, identity matrix result with low fitness, matrix inversion error), use a fallback strategy (e.g., use only the RANSAC result if reasonable, or reuse the previous frame's relative motion `T_last_rel`).
       f. Calculate the relative motion: `T_rel = T_prev_to_curr = inverse(T_curr_to_prev_icp)` (or the fallback transform).
       g. Update the global pose: `T_Gi = T_Gi-1 * T_rel`.
       h. Store the global pose `T_Gi` in the trajectory.
       i. Transform the *original* resolution cloud `Pi` using `T_Gi` and store it (`Pi_global`).
       j. Update `Pi-1_proc = Pi_proc`.
- **Script Used:** The final script generated (`2022006_5_part4_ransac_p2p.py` or similar).
- **Parameters:** Values similar to the best found in Part 2, adjusted slightly for sequential stability (e.g., `preprocess_voxel_size=0.05`, `icp_threshold=0.1`, `icp_max_iteration=100`, `knn=30`).

6.3 Estimated 3D Trajectory

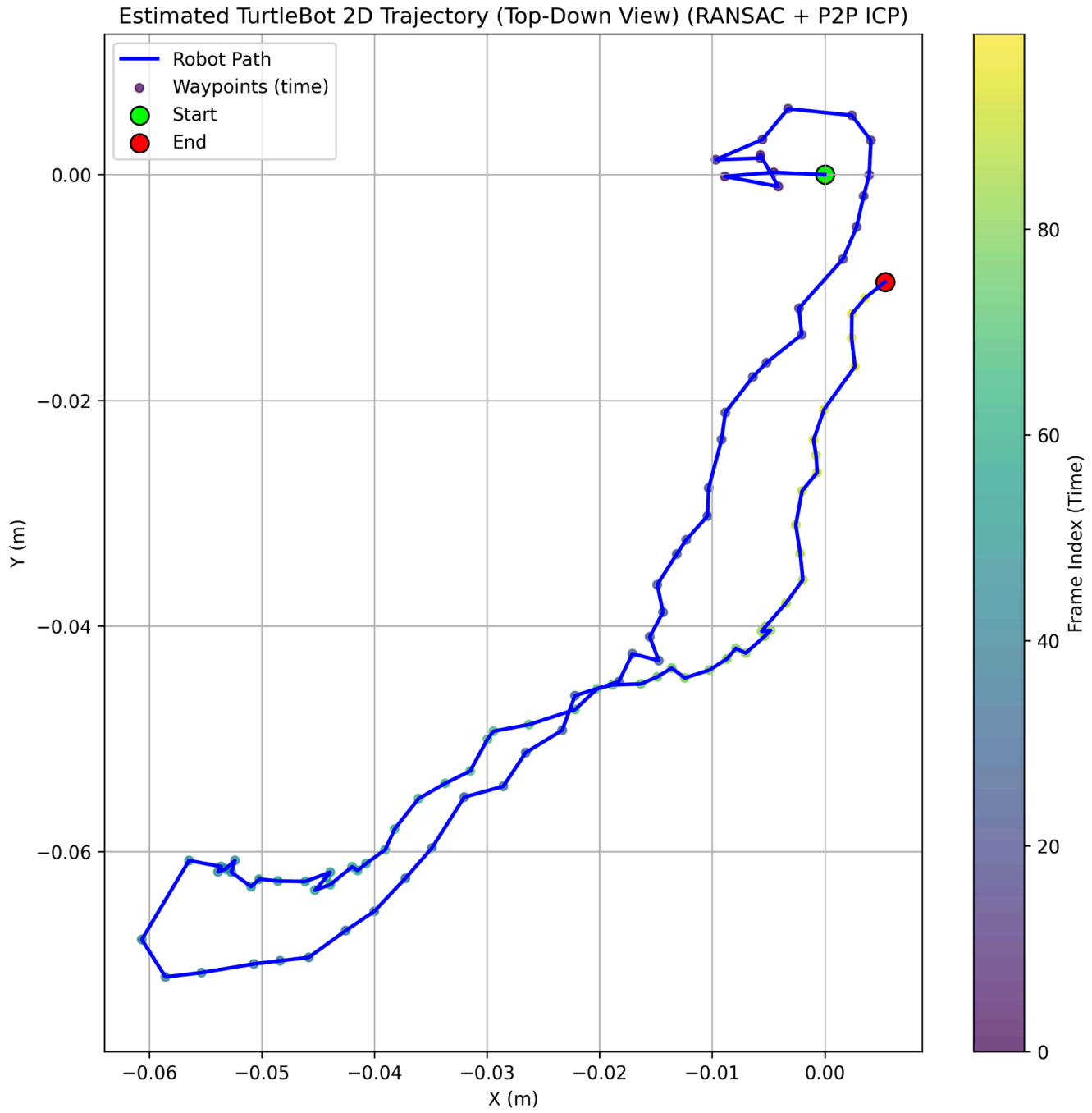
The sequence of global poses `T_G0, T_G1, ..., T_GN` constitutes the estimated trajectory. The translation components `(tx, ty, tz)` of each pose matrix represent the robot's position at each step.

- **Visualization:**

3D:



2D (TOP VIEW):



<Shown the 3D trajectory plot generated by the final Part 4 script (using RANSAC + P2P ICP). This plot should show the X, Y, Z path of the robot, likely color-coded by time, with start and end points marked.>

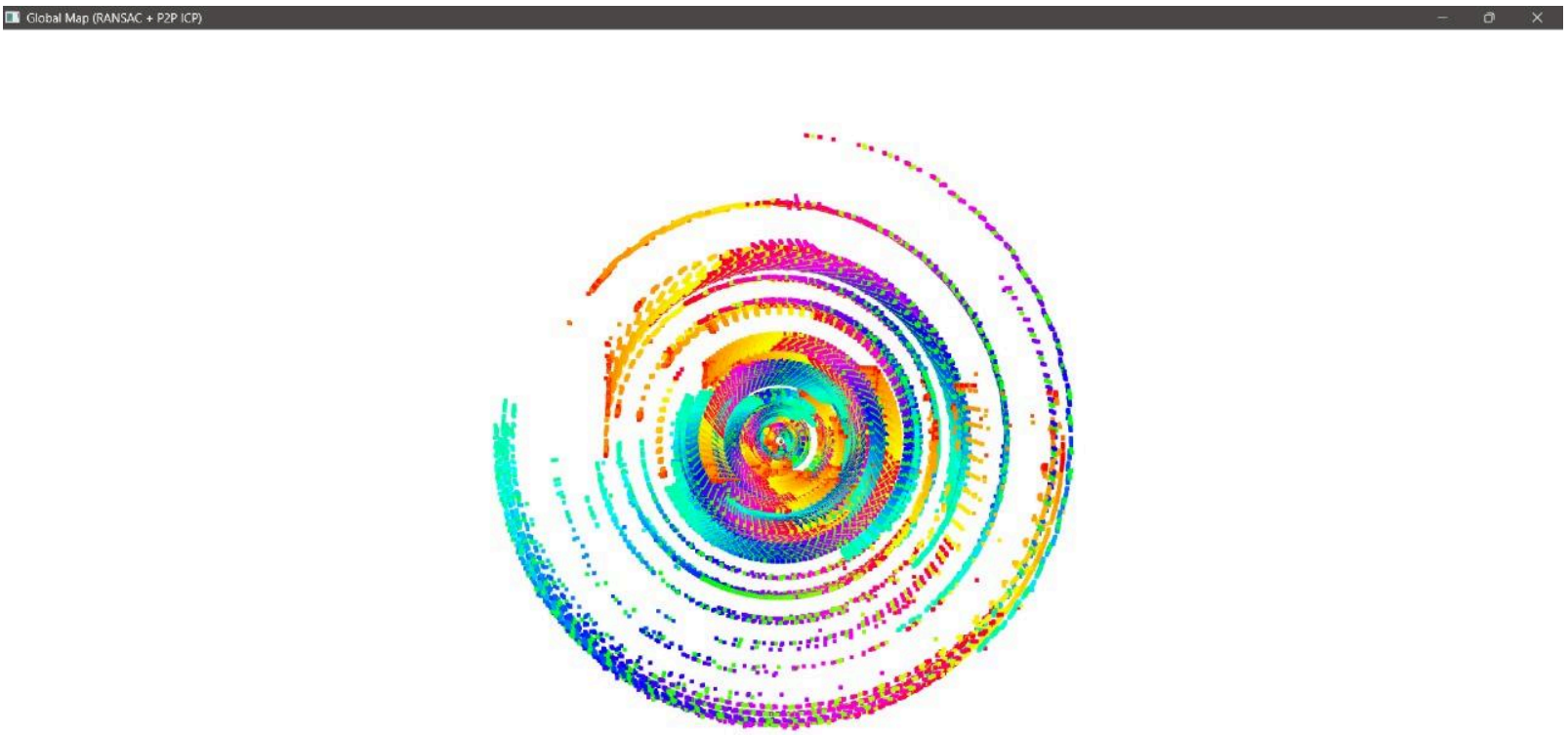
Description: The plot visualizes the estimated 3D path of the TurtleBot. The axes represent meters in the global coordinate frame (defined by the first point cloud). The path shows [Describe the general shape, e.g., turns, straight sections, any noticeable drift or jumps]. The start (lime) and end (red) points are indicated.

- **CSV File:** The trajectory data (frame index, x, y, z, roll, pitch, yaw) was saved to: `estimated_trajectory_ransac_p2p.csv`.

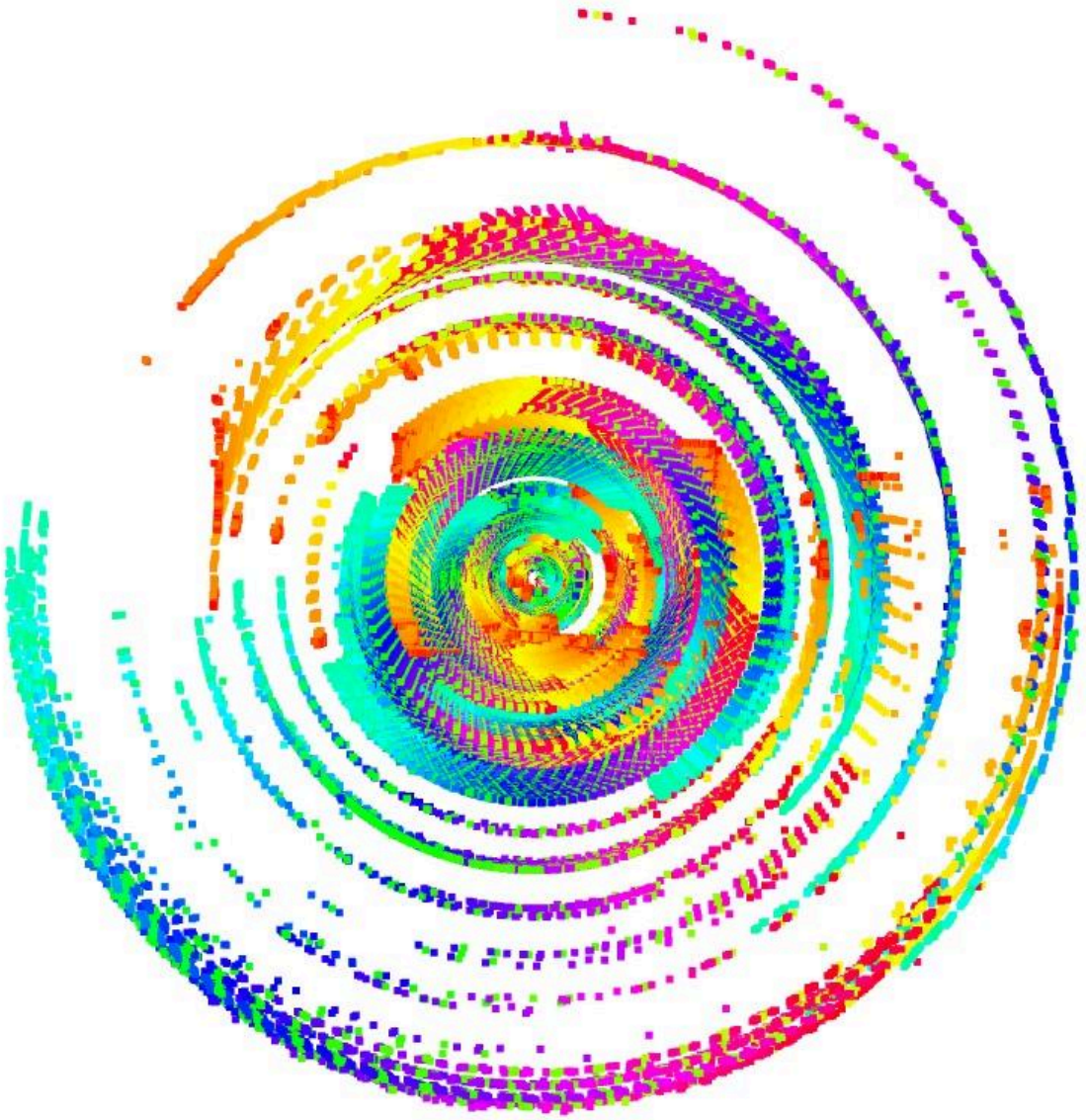
6.4 Global Registered Point Cloud

The globally transformed original-resolution point clouds (`P0_global, P1_global, ..., PN_global`) are combined to create a map of the environment explored by the robot.

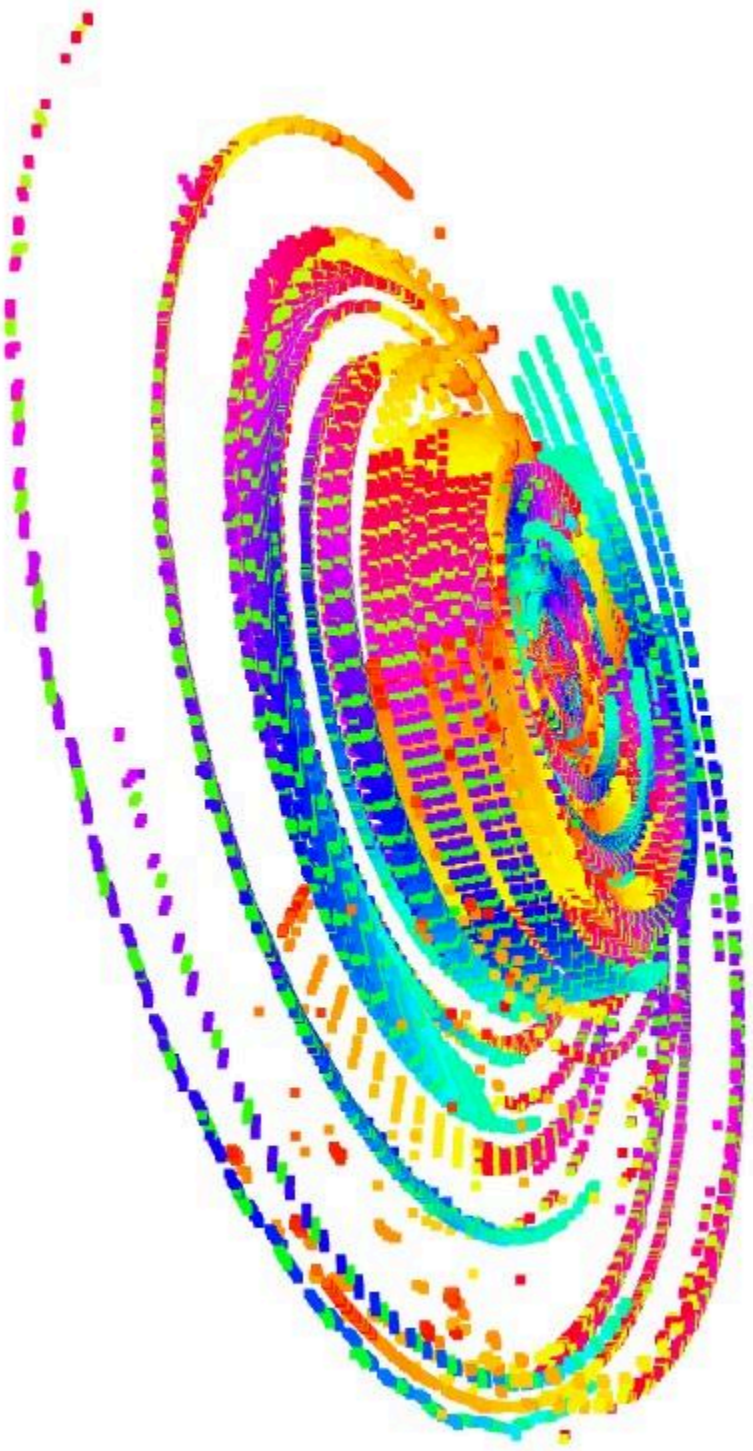
- **Visualization:**



ZOOMED IN PREVIEW:



FROM DIFFERENT ORIENTATION:



<Inserted visualization of the combined global point cloud map generated by the final Part 4 script. Clouds can be colored uniquely by index using a colormap (like HSV or Viridis) or painted a uniform color.>

Description: This image shows the composite map created by merging all point clouds aligned to the global frame. [Describe the visual quality, e.g., Structures like walls and corridors are generally well-aligned, indicating successful registration. Some areas might show blurriness or "ghosting" if registration errors accumulated (drift). The density varies depending on how long the robot stayed in certain areas.]

- **PCD File:** The combined (downsampled) global map was saved to: `global_map_ransac_p2p.pcd`.

6.5 Output Files

The following key output files were generated by the Part 4 script:

- `estimated_trajectory_ransac_p2p.csv`: Contains the pose (position and orientation) for each frame.
- `global_map_ransac_p2p.pcd`: The combined point cloud map.
- `registration_metrics_ransac_p2p.csv`: Pairwise registration metrics (fitness, RMSE) for RANSAC and ICP steps.
- `trajectory_plot_3d_ransac_p2p.png`: The 3D plot of the trajectory.
- `trajectory_plot_2d_ransac_p2p.png`: The 2D (top-down) plot of the trajectory.
- `registration_summary_ransac_p2p.txt`: A text file summarizing the parameters, success/failure counts, average metrics, and trajectory statistics.

---

7. Discussion

7.1 Comparison of Initialization Methods

The experiments in Part 2 clearly established the significant advantage of using RANSAC (with FPFH features) over a simple random orthogonal matrix for initializing ICP. Random initialization frequently led to ICP converging to incorrect local minima, resulting in poor alignment metrics (low fitness, high RMSE). RANSAC provided a robust global alignment estimate, enabling ICP to effectively refine the pose locally and achieve much more accurate and reliable results. This is particularly critical for sequential registration (Part 4), where errors accumulate. A poor initial alignment in one step can corrupt the entire subsequent trajectory.

7.2 Point-to-Point vs. Point-to-Plane ICP

While Part 4 specifically required the use of Point-to-Point ICP, it's worth noting that Point-to-Plane ICP (explored in `2022006_5_complete_ransac_p2plane.py`) often performs better for environments with planar structures, like indoor scenes. It leverages surface normal information, typically leading to faster

convergence and potentially smoother trajectories. However, Point-to-Point ICP is simpler, does not strictly require normals during its refinement step (though RANSAC initialization still needed them here), and can work reasonably well, especially when provided with a good initial guess via RANSAC. The choice between them can depend on the specific application, environment structure, and computational constraints.

7.3 Challenges and Limitations

- **Drift:** Sequential registration is prone to error accumulation (drift). Small errors in each pairwise alignment can compound over long sequences, leading to significant deviation in the estimated trajectory and map inconsistencies. More advanced techniques like loop closure detection and pose graph optimization are needed to mitigate drift in larger-scale mapping.
- **Parameter Sensitivity:** ICP and RANSAC performance depends on appropriate hyperparameter tuning (voxel size, thresholds, iterations, normal estimation parameters). Optimal values can vary depending on the dataset, sensor noise, and environment.
- **Computational Cost:** Feature extraction (FPFH) and RANSAC are computationally more intensive than basic ICP with random initialization. Processing long sequences can be time-consuming.
- **Featureless Environments / Low Overlap:** RANSAC based on geometric features may struggle in environments lacking distinct features (e.g., long empty corridors) or when the overlap between consecutive scans is very small. This can lead to registration failures.
- **Handling Failures:** The sequential script included basic fallback mechanisms (using RANSAC only, using last known motion) when ICP failed. More sophisticated failure recovery might be needed in real-world scenarios.

---

8. Conclusion

This assignment successfully demonstrated the application of the Iterative Closest Point algorithm for point cloud registration and TurtleBot trajectory estimation. Key findings include:

1. Basic Point-to-Point ICP can align consecutive point clouds, but its success heavily relies on the initial transformation guess.
2. RANSAC initialization using FPFH features provides a significantly more robust and accurate starting point for ICP compared to random initialization, leading to superior alignment results.
3. By applying RANSAC initialization and Point-to-Point ICP refinement sequentially, the 3D trajectory of the TurtleBot was estimated, and a coherent global map of the environment was constructed.
4. The process involves careful preprocessing (downsampling, normal estimation) and parameter selection.
5. While effective, the sequential approach is susceptible to drift, highlighting the need for more advanced SLAM (Simultaneous Localization and Mapping) techniques for large-scale, high-accuracy mapping.

The generated trajectory (CSV file, plots) and the global point cloud map represent the final outputs of this bonus task.

---

9. References

- Open3D Library: http://www.open3d.org/ - Used extensively for point cloud loading, processing, registration, and visualization.
- 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. (Classic ICP paper)
- Rusu, R. B., Blodow, N., & Beetz, M. (2009, May). Fast Point Feature Histograms (FPFH) for 3D registration. In *2009 IEEE international conference on robotics and automation* (pp. 3212-3217). IEEE. (FPFH paper)
- Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, *24*(6), 381-395. (Classic RANSAC paper)
