Lidar-inertial sensor fusion

RPCN Groupwork B
Prawin Kumar (s2731991) Vigneshwar Karunanithi (s3207595)
Matthijs Reus(s2519402) Yaswanth Sathiyan (s3259358) &
Tongli Zhu (s3159396)
January 2024

I. Introduction

Dead reckoning using an IMU sensor system is often susceptible to drift. Small errors in the measurements result in big errors in position over time. Using additional sensors to provide location is essential for improved accuracy. This paper explores capabilities of LiDAR sensors for positioning. The sensor system will be compared to the IMU dead reckoning and to a sensor fusion approach, where IMU and LiDAR data are combined.

II. Materials

The materials used in the project is detailed in Table I.

Table I: Material list

Items	Contents
Software	RVIZ, PyCharm
Hardware	ITC backpack
Dataset	Group 9 bag 1301

Note: We are a newly merged group, so our group number is group 6, but the data we use is group 9.

III. Questions and answers

A. What are the relations between keyframes (lidar lecture 1), poses seen in rviz, and submaps (cartographer documentation)?

Pose in RVIZ refers to the current position and orientation of the robot, which can be represented by an arrow or a coordinate system. At each timestamp, the sensor can obtain a Frame through the acquired data. At two time points, we use ICP to calculate the rotation and translation between frames. However, LiDAR errors may cause drift, especially when the difference between the two frames is not very large, you will find a lot of noise. So we use a keyframe, that is, when the difference between the next frame and keyframe exceeds a certain threshold, we will update the keyframe. How to evaluate the ”difference” is done through pose. For example, the difference between the current pose and the key frame pose is 10cm, 10degree, etc. Submapping is a map-building strategy in which a map of an environment is divided into smaller, more manageable parts. After a certain number of keyframes are collected, SLAM will start building a new submap.

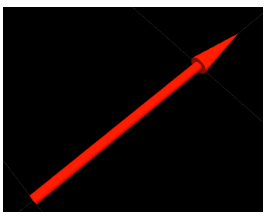

Figure 1: Pose in RVIZ

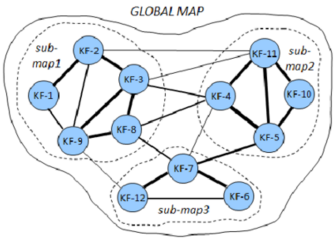

Figure 2: Keyframe relational graph

B. What are the conditions for creating submaps in cartographer?

A submap is a local representation of the environment, typically consisting of a collection of sensor data or scans. The creation of submaps starts with an initial guess from the pose extrapolator, which uses sensor data from various sensors besides the range finder. The scans are inserted into the current submap using scan matching algorithms such as Ceres. Scan matching involves aligning the new scan with the existing submap to determine the relative pose (position and orientation) of the new scan. To avoid inserting too many scans per submap, a motion filter is applied which considers the significance of the motion between two scans before inserting a new scan into the submap. This is considered complete when a certain amount of range data has been received. This ensures that the submaps are sufficiently informative for the SLAM algorithm. Parameters such as maximum time, distance, and angle thresholds in the motion filter determine whether a scan is significant enough to be inserted into the submap.

C. How are the submaps used?

Submaps are a crucial technique used to stitch together the global map using the local lidar and IMU readings. Any new information encountered by these sensors are made into submaps which are then added to the global map. This is a continuous process that happens during the motion of the robot and as it observes new environments. The cartographer also uses the submaps for the purpose of loop closure by checking the similarity between submaps.

D. Compared to the 1st loop, is the number of submaps created higher, lower, or the same during the 2nd and the 3rd loop? Why/ Why not?

The number of submaps created during the 2nd and 3rd loops are lower compared to the first loop. Submaps are only created when the sensors detect new information in the surroundings that are not already registered into the global map. Since after the first loop closure we are following a very similar path to the first loop, the chances of the sensors encountering new information during the 2nd and 3rd loop are much lower leading to lower number of submap creations.

E. Think and play with the parameters: What would be the smallest number of submaps for running the part B exercise successfully? Hint: there needs to be enough overlap between them for the SLAM algorithm.

We control the size of the submap using the command `TRAJECTORY_BUILDER_2D.submaps.num_range_data` and conducted experiments with different parameters. The screenshots of the experiments are in the appendix. Based on the results in Table II, we believe that a reasonable submap should be around 20. If it's smaller, the trajectory will deviate.

Num_range_data	1st	2nd	3rd	Total	Plot evaluation
50	16	14	12	42	Good
90	9	8	7	24	Good
100	8	7	6	21	Good
120	8	6	5	19	Slight deviation on the third lap
150	7	5	3	14	Deviation on the third lap
200	5	3	3	11	Serious deviation on the third lap

Table II: Experimental Results of Submap Sizes

F. How is the loop closing done?

The loop closure is done by comparing the current environment of the robot detected by the sensors with all the previous keyframes that the robot has passed through to verify if it has arrived at any of the previously visited locations. If it detects a very similar environment to any of the previous keyframes, it adjusts the poses of the plotted trajectories and keyframes to make sure the current location aligns with the detected old location. The submaps can also be utilised to find the similarity between the environments.

IV. Discussion of Results

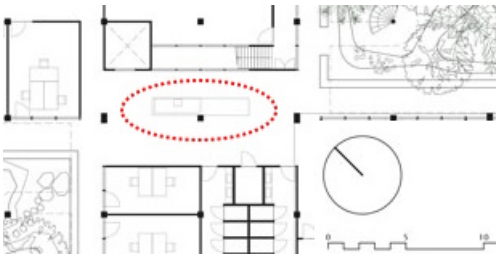

Figure 3: Expected path [1]

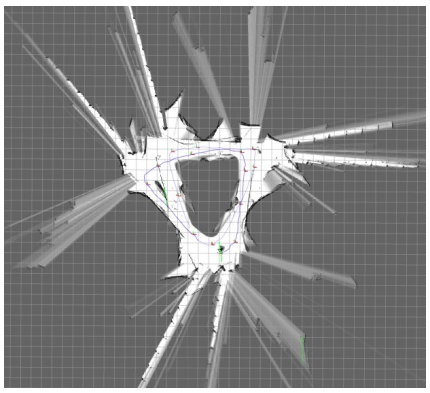

Figure 4: LiDAR map

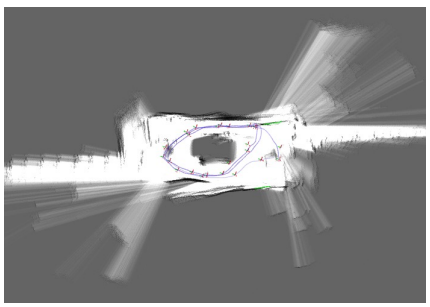

Figure 5: LiDAR + IMU map

In the experiment, we circled the water bar three times. In fig.3 the expected path can be seen. A somewhat circular path around the water bar with a width of 8-10m and height of 4-5m. The floorplan is taken from [1].

The results obtained by looking at only IMU data accurately display three circular loops (see fig.6). The trajectory looses a sense of scale[2]. The LiDAR data confuses each 180° turn with a 120° turn. This is then reinforced when after three turns the bag reaches its starting point and the loop is closed. The global map it constructs can be seen in fig. 4. These results show that the IMU has good pose awareness and can accurately determine the orientation of the sensor system and its sudden velocity changes. In contrast the LiDAR system can accurately determine the scale

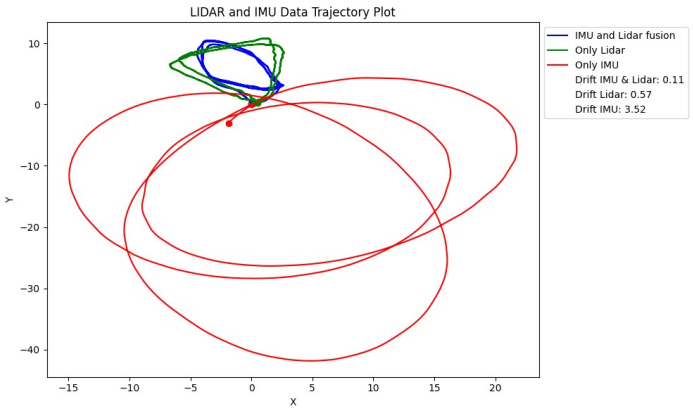

Figure 6: Comparison of 3 methods

of the environment and keep track of constant environmental features. Figure 4 & 5 are made using the ROS Cartographer [3].

The resulting map is the most accurate when using both LiDAR and IMU data. The IMU provides additional information about movement and direction, helping to pose keyframes in the LiDAR data, especially when turning and accelerating or decelerating (see fig.5). This way, the start and end points of each lap can be identified more accurately, allowing three separate laps to be drawn. Moreover the constructed map accurately portrays the space as well as the openings to the corridors.

Based on the experiments performed with the lua files for ROS Cartographer for the obtained bag file, we see that modifying the parameters into Table III. We see the best results for lower latency by modifying the values of the configuration values as described in [3]. The default configuration values were identified from [4]. The findings of our experiment on the submaps with the parameter TRAJECTORY_BUILDER_2D.submaps.num_range_data is as follows:

1) For the default value of 100 we identified that 21 submaps were produced by the cartographer. This is close to our most accurate representation which was obtained with a value of 90 that yielded 23 submaps.
2) For lower values of range data we see that the the number of submaps produced increased and for higher values of range data, we observe that the number of submaps decreased which also results in a considerable drop in accuracy.

This aligns with our theoretical framework, wherein the quantity of range data received by the local SLAM dictates the completion criteria for a submap. Consequently, a reduced amount of range data results in the generation of a higher number of submaps. Our experiments were designed around submaps that strike a balance, being sufficiently large for effective loop closure yet compact enough to minimize drift. Through experimentation, we determined that a representable trajectory, while minimizing the number of submaps, was achieved with a total of 11 submaps .

V. Conclusion

This paper describes some common terminology used in LiDAR mapping and functionalities of the cartographer tool. The strengths and weaknesses of the LiDAR and IMU positioning systems are compared. The IMU's strength to determine the pose and the LiDAR's strength to determine scale and sense environmental features create a powerful sensor system when fused together. Our results show that with the fused sensor system indoor localization can be achieved.

REFERENCES

[1] Civic public architecture. Itc universiteit twente enschede. https://www.civicarchitects.eu/nl/projects/itc-twente-2. Accessed: 2022-01-15.

[2] P. Kumar, N. Meijer, and M. Reus, "Pedestrian dead reckoning system - rpcn groupwork a."

[3] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[4] Open Robotics. Cartographer trajctory builder 2d configuration. https://github.com/cartographer-project/cartographer/blob/master/configuration_files/trajectory_builder_2d.lua#L68. Accessed: 2022-01-15.

VI. Appendix A

Parameter	**Value**
MAP_BUILDER.use_trajectory_builder_2d	true
TRAJECTORY_BUILDER_2D.min_range	0.06
TRAJECTORY_BUILDER_2D.max_range	30.
TRAJECTORY_BUILDER_2D.missing_data_ray_length	30
TRAJECTORY_BUILDER_2D.use_imu_data	true
TRAJECTORY_BUILDER_2D.use_online_correlative_scan_matching	true
TRAJECTORY_BUILDER_2D.real_time_correlative_scan_matcher.linear_search_window	0.25
TRAJECTORY_BUILDER_2D.real_time_correlative_scan_matcher.translation_delta_cost_weight	12.
TRAJECTORY_BUILDER_2D.real_time_correlative_scan_matcher.rotation_delta_cost_weight	4e2
TRAJECTORY_BUILDER_2D.motion_filter.max_angle_radians	math.rad(0.2)
POSE_GRAPH.optimize_every_n_nodes	10
TRAJECTORY_BUILDER_2D.ceres_scan_matcher.occupied_space_weight	0.35
POSE_GRAPH.optimization_problem.acceleration_weight	0.015
POSE_GRAPH.optimization_problem.rotation_weight	0.015
TRAJECTORY_BUILDER_2D.submaps.num_range_data	90

Table III: My_Robot.lua Configuration Table

VII. Appendix B

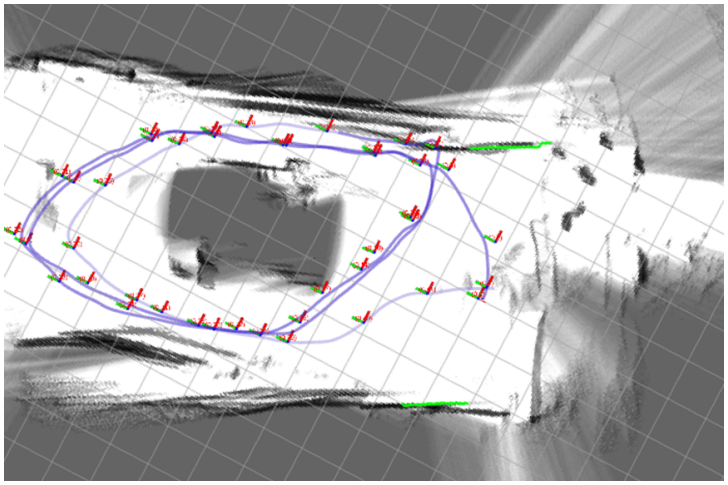

Figure 7: Num range data-50

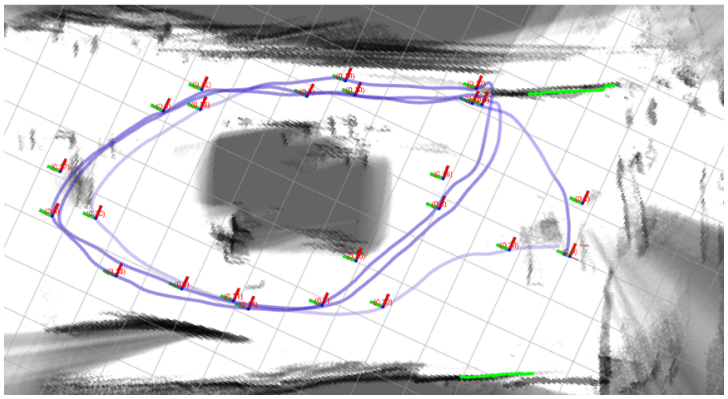

Figure 8: Num range data-90

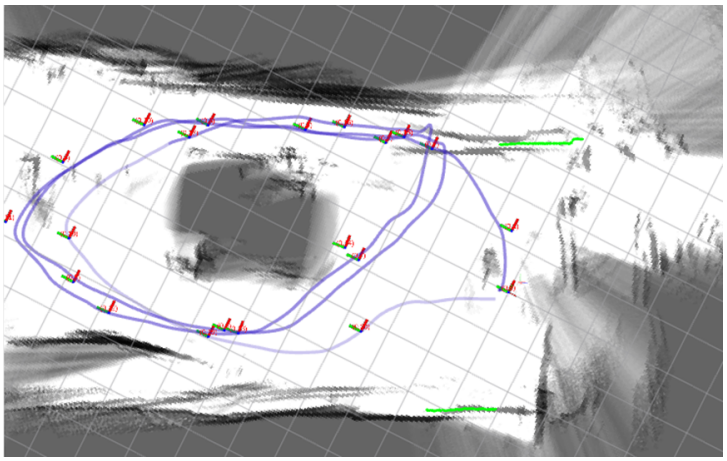

Figure 9: Num range data-100

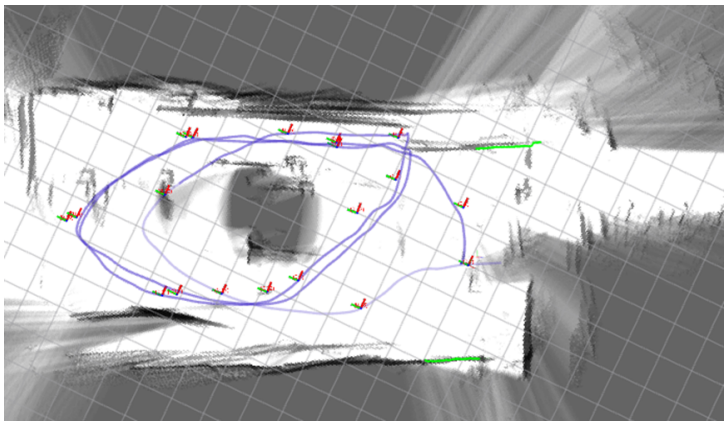

Figure 10: Num range data-120

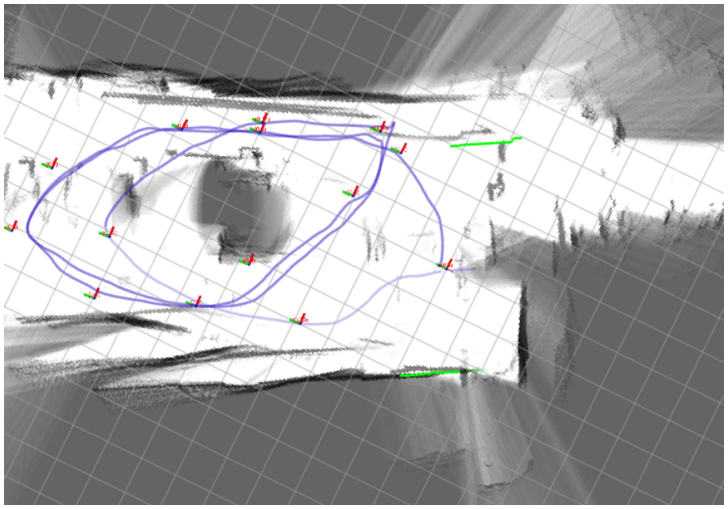

Figure 11: Num range data-150

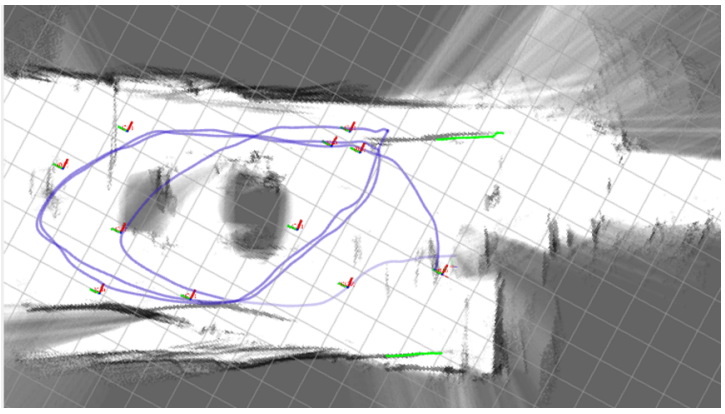

Figure 12: Num range data-200