Path Finding

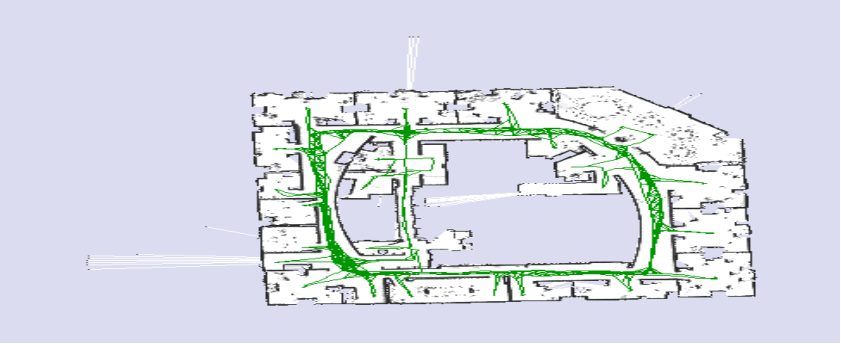
Overview

Since the start of my computer science education I was always fascinated with algorithms and making programs faster and more efficient. When I got to study fastest path algorithms like Dejikstra, BFS, DFS, etc.. I was captivated by how a couple lines of code could find the fastest and more efficient path from node A to node B. After learning and understanding these algorithms I always wanted to see how pathfinding and fastest path algorithms are used in the modern world. That's where the family of SLAM algorithms come into play. Simultaneous Localization and Mapping better known as SLAM is a family of algorithms in the robotics sphere that focuses on constructing and updating a map of an environment while also keeping track of the robots location. There are many types of SLAM algorithms. In my project I have concentrated on **Graph Slam**. Graph SLAM abstracts sensor readings into a graph structure, in the graph each node represents a specific robot pose namely position and orientation and the edges between these nodes represent special constraints. The constraints come from motion which comes from the odometer data and other robots sensors like lazars. "*In graph-based SLAM, the poses of the robot are modeled by nodes in a graph and labeled with their position in the environment. Spatial constraints between poses that result from observations or from odometry measurements are encoded in the edges between the nodes.*"[1]. In this project I will be implementing Graph SLAM algorithm with an already gathered data set. The gathered data set comes from "SLAM benchmarking"[2] website. This website has many datasets for the robot community to use for projects. The potential applications of Graph SLAM in the industry are vast. The applications range from Enhanced GPS integration in Urban mapping to Autonomous Vehicles. I find autonomous vehicles very interesting and a field that I would like to work in. Graph SLAM enhances both navigation and safety of autonomous vehicles. Moreover, the precision that Graph SLAM provides to autonomous vehicles is crucial and vital for cars to traverse complex environments. "*Graph-based SLAM can be crucial in the development of autonomous vehicle systems, providing the backbone for precise and robust navigation capabilities in complex environments. The implementation can handle dynamic updates to the map, which is essential in urban settings where the environment changes frequently*"[1].

Project Results

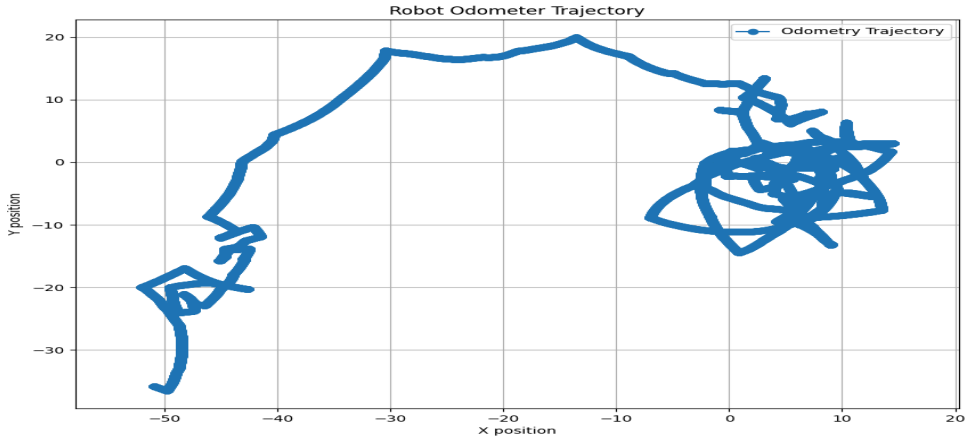
Data overview

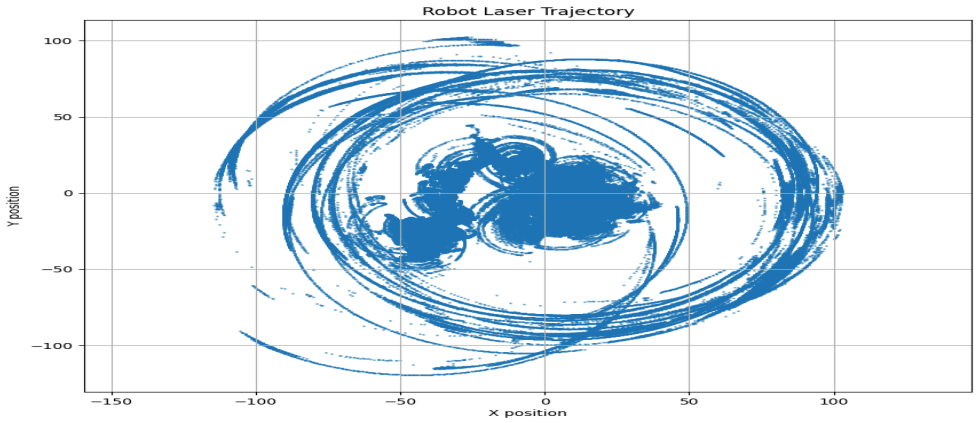
As I have stated in the **Overview** section of the report I have used an existing data set from "SLAM benchmarking"[2] website. The data set is called "intel.clf". "intel.clf" is a collection of odometer and laser readings of an Intel research lab in Seattle that was collected by Dirk Hähnel (below shown how it looks in the web)



It is not described how the data was collected however upon research into data collection for navigation datasets I found out that a lot of data sets are gathered by sensors that are mounted on a robot and the robot traversing the

area[3, site]. Before I used the “intel.clf” dataset I wanted to make sure that the dataset is well structured and not corrupt. Namely, I wanted to see if the sensor data from odometer readings and laser reading were correct. For that I have decided to construct and see the odometer and laser scans before using the dataset with my algorithm.

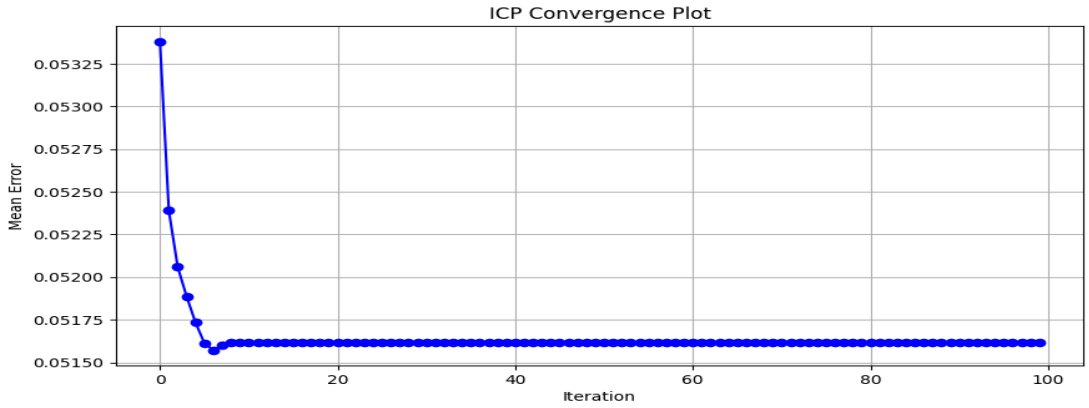




From the above graphs the following can be deduced. Based on the odometer plot, the beginning of the robot's path was somewhat structured but as the robot moves further away from the starting position the path becomes more hectic. Moreover, the hectic lines and loops that the odometer graph shows towards the end are an accumulation of errors over time. In addition to that, apart from the errors over time it can also suggest that those loops and out of order lines are the areas where the robot corrected itself and did complex maneuvers. Based on the laser plot, we can see that the robot traversed the same area more than once. This is done to get a more detailed scan. Furthermore, there is a high concentration of points in the center indicating the robot spent a lot of time there. This means the space is more complex. The spreading lines in the outer loops indicate some drift in the lazer reading or errors in the position estimation over time. I hypothesize that these drifts and errors in the position estimation over time are guilty of tricking my graph SLAM implementation by making it traverse unnecessary areas. I will get more in depth to the previous statement further in the report. Namely in the **Analysis and Major Results** part. However, for the sake of the data itself it's satisfactory and I will use it in my graph SLAM implementation.

Approach

To implement Graph SLAM I have decided to use python3. In the beginning stages I wanted to use ROS to implement the algorithm however I found that implementing the algorithm just by using python3 was more efficient (in the sense of it satisfying the target of the project) and is more convenient. In my approach and implementation I have decided to use the "g2o" framework. The "g2o" framework is known for its optimization capabilities and is standardly used in the robotics community for SLAM implementations. *"The G2O framework is formulated as a nonlinear least-squares optimization. Given the robot measurements, the optimizer is able to formulate the solution with the least total of the square of the residual error between the expected and actual states"[4].* In addition to using the "g2o" framework I have used an Iterative Closest Point better known as ICP script from an author called Clay Flannigan [5] who is a researcher at Southwest Institute. ICP is an algorithm which minimizes the difference between two clumps of points. When I decided to use the ICP implementation from Mr. Flannigan I wanted to see the ICP convergence with my chosen data. The plot helped me to understand and assess the performance and stability of the ICP algorithm.

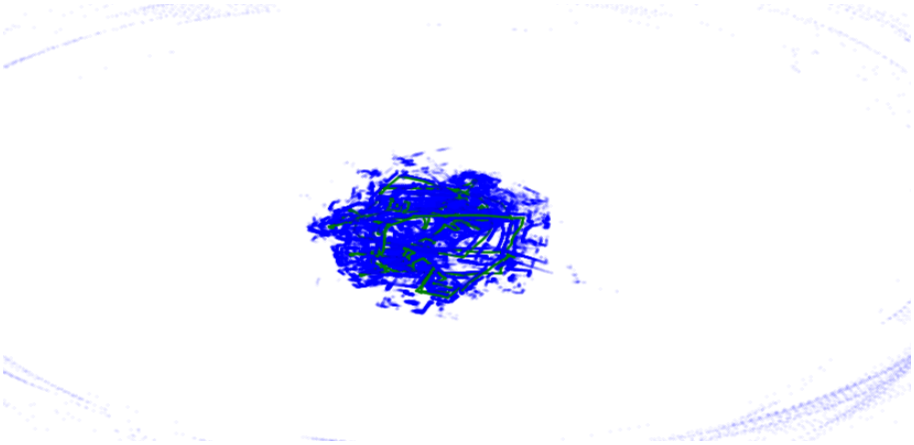


From the plot we can see that the mean error drops significantly within the first few iterations. This shows that the ICP algorithm found a good fit between the point sets. Furthermore, the graph shows that between iteration 15 to 100 it had reached a plateau. This suggests that the ICP algorithm has reached the convergence criteria and/or is close to the local minimum of the error function. To sum up about the ICP algorithm the low error rate after the drop is a very good sign and the ICP algorithm works very well with my chosen data. In my code the class called **PoseGraphOptimization** is the core of the SLAM algorithm. This class is responsible for adjusting the robot's trajectory to best fit measurements. Moreover, **BlockSolverSE2** a which is part of the "g2o" framework works with matrices that represent the relationships between different robot poses and controints (in graph terms: vertices and nodes). The **Optimization AlgorithmLevenberg** which also is part of the "g2o" framework is an algorithm for nonlinear least squares problems which optimizes the graph to find the best possible set of node positions. Now let's look at the vertex addition. The vertices in the pose graph represent the robots position and orientation at different points in time. Adding the vertices means creating a vertex for each pose and adding it to the graph. In my code the pose is represented as an "SE2" object which is part of the "g2o" framework which includes 2D positions and orientation. Now let's look at edge additions. Edges represent the spatial relationship between poses which were taken from the sensors. In graph SLAM they represent the robot's motion or the observed relative position of landmarks. Now let's look at how I used the ICP algorithm for scan matching. As I have stated previously the ICP algorithm is used to align clumps of points which is commonly used when matching laser scans to build a continuous map. The ICP in my code iteratively revises the transformation needed to make the distance smaller between corresponding points in the dataset. After we have the vertices and edges all added up now it's time for optimizing the pose graph. The optimization adjusts the vertices position to best satisfy the constraints represented by the edges. After the optimization, the graphs vertices represent the estimated best fit poses of the robot at different points in time. To visualize this requires extraction of these poses and plotting them to provide a graphical

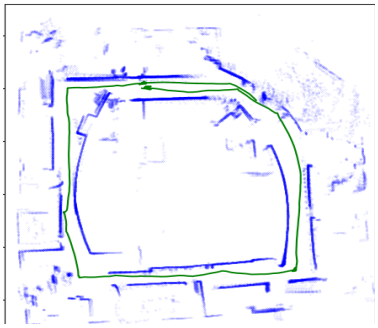
representation of the robot's trajectory and the environment it navigated. Graph SLAM has several advantages over other SLAM algorithms which make it more suitable and favorable over other SLAM algorithms. Graph SLAM performs global optimization meaning it performs optimization on the entire graph (as discussed above). This graph optimization leads to an accurate map due to consideration of all constraints. In addition to that Graph SLAM is very flexible and can handle many types of sensor data ranging from odometer readings to lidar. One of the main advantages of Graph SLAM is its efficiency in large scale environments due to its ability to optimize only the most recent part of the graph. Despite the advantages of Graph SLAM the choice of the SLAM algorithm for a project depends on real time constraints, computational resources, sensor availability and the environment upon which the robot operates.

Analysis and Major Results

I have done the data analysis in the **Data Overview** section and I have done the algorithm analysis (Graph SLAM, ICP and Levenberg) in the **Approach** section. The only part left to analyze is the results from running my implementation of the Graph SLAM algorithm.



With the given results it's obvious that a combination of laser and odometer sensor data was used. The reason being is the large ellipse circles around the graph (see figures in **Data Overview**). Moreover, my Graph SLAM implementation decided to traverse the area where it was not supposed to traverse. My hypothesis is that the algorithm decided to traverse the rooms rather than sticking to the main path. As I have talked about in the **Data Overview** due to the sensors picking up data of the places that the robot did not traverse, it still has enough data to create a path in these spots. However, in the **Data Overview** part I hypothesized that the data drifts in the laser readings tricked my algorithm implementation. In my opinion this hypothesis stays true because as you can see in the output of my algorithm it's evident that the algorithm mapped out paths that are not even possible to access. Compared to how the data looks from the source which was provided in the **Data Overview** section, the mapping and triversion that my implementation uses seems a bit dirty in simple words. My implementation does not deal well with noise and is very sensitive to the data that is given. There needs to be some refining done to my implementation. However, my algorithm implementation is very consistent in the estimation of the trajectory over time. Moreover, the map does not show significant distortion which says that the pose optimization has worked effectively to correct cumulative errors. Even though my implementation is not perfect it outlines the importance and the effectiveness of Graph SLAM. There is not much material and data based around Graph SLAM as it is a very advanced and complicated algorithm in the robotics space. One of the main reasons that I enjoyed implementing it is due to its advancement and complexity. In reference below is how a very well implemented Graph SLAM algorithm is from a software developer called Göktuğ Karakaşlı[6]



About The Project

As I'm the only person working on this project, my role was to do everything. The analysis of the data, algorithm and results were done in the **Project Results** section of the report. In this section I would like to focus on the hardships and obstacles that I have faced in this project. Since I was the only person on this project I had to do a substantial amount of research. There is not a lot of information in regard to Graph SLAM, as I have mentioned earlier in the report. Before I got into understanding Graph SLAM I decided to understand what SLAM is. A book that came in handy was "*Springer Handbook of Robotics"[7]*, this book namely chapter 37 helped me alot in understanding about SLAM. In addition to that, the data set that I have used ("intel.clf") was recommended by the book to use in a SLAM implementation (p.884). After understanding SLAM I decided to see easier implementation of SLAM algorithms. I saw tutorials of people implementing SLAM with the "slam_toolbox" in ROS however it did not fit my criteria for the project. After further research I have stumbled upon Graph SLAM research papers. In those papers the Graph SLAM algorithm was outlined. Namely, in the research paper called "*A Tutorial on Graph-Based SLAM"[1]* it explained the algorithm. It was very difficult for me to understand it due to some mathematical terms and the complexity itself. As I have stated many times before Graph SLAM is not an easy algorithm to grasp let alone implement to a novice robotics student. At first I decided to implement the algorithm using only research papers however it was quite difficult and I had a lot of knowledge gaps. This task was not impossible but would take much more time than was permitted by the course. What I decided to do next is to see other peoples implementations of Graph SLAM. There were not many implementations of this algorithm, however the implementation of Göktuğ Karakaşlı, a software developer in the robotics field stood out to me. I dissected his code, understood his way of thinking and got a pretty good idea how to implement the algorithm. The basic structure of Graph SLAM was as follows: Optimizer setup -> read data -> add vertex -> add edge -> optimize graph -> visualize optimized graph. With the inspiration from Göktuğ Karakaşlı implementation I have implemented the Graph SLAM algorithm. It was a lot of trial and error, most of the time was spent debugging and re-running the code over and over again. And even still my implementation is not fully refined and is flawed. Compared to the results of Göktuğ Karakaşlı my algorithm is not even comparable, the skill gap is very noticeable. With some more time to work on it and further refinement I believe I could make my algorithm perform similarly to Göktuğ Karakaşlı. Nevertheless, I'm very happy with the results and with my implementation. As I have mentioned before this whole project was written by me therefore all the analysis, results, observations, work, code, research, data set selection, explanation, etc.. was done by me, meaning team results are my individual results.

conclusion

In conclusion, this report has shown a comprehensive analysis and implementation of Graph SLAM to effectively map and estimate the trajectory of a robot. My presentation detailed the integration of sensor data, the optimization of a pose graph and the effective use of scan matching techniques to refine the results. The analysis highlighted the robustness of Graph SLAM in managing noise and loop closures, as well as its scalability and efficiency for large scale environments. In short, it was not an easy task but a very interesting one. In the future I want to refine my implementation and optimize my algorithm.

Reference

1. “A Tutorial on Graph-Based SLAM” by:Giorgio Grisetti, Rainer Kummerl,e Cyrill Stachniss, Wolfram Burgard [1]
2. slam benchmarking website: http://ais.informatik.uni-freiburg.de/slamevaluation/datasets.php# [2]
3. Data set collection methods: https://www.polymathrobotics.com/blog/recording-robot-data [3]
4. “Analysis for Graph-Based SLAM Algorithms under g 2 o Framework” by:Tianxiang Lin, Jiayin Xia, Qishun Yu, Kerou Zhang and Ben Zhou [4]
5. Clay Flannigan github: https://github.com/ClayFlannigan [5]
6. Göktuğ Karakaşlı github: https://github.com/goktug97 [6]
7. “springer handbook of robotics” by:Oussama Khatib [7]
8. “The GraphSLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures” by: Sebastian Thrun, Michael Montemerlo