Study on the Effect of Loop Closures in 2-D SLAM

Vaibhav Shirole
University of Minnesota
Minneapolis, Minnesota
shiro019@umn.edu

Abdullah Alhaidar
University of Minnesota
Minneapolis, Minnesota
alhai006@umn.edu

May 9, 2023

1 Abstract

Autonomous mapping is the process of creating a map of a physical environment with the use of algorithms and sensors. Specifically, Simultaneous Localization and Mapping (SLAM) algorithms along with Light Detection and Ranging (LIDAR) sensors to capture data and optimize it to provide an accurate map of the environment. The paper investigates the impact of loop closures on the accuracy of probabilistic occupancy grids generated through graph-based SLAM algorithms. The study was conducted using a 2-D floor plan of a living room, with a virtual sensor stack collecting LIDAR data. The resulting maps were compared with a ground truth floor plan to evaluate their accuracy. MATLAB was used to simulate a virtual environment for the algorithm to scan. The results proved the impact of loop closures on the accuracy of maps by comparing multiple generated maps with each other, and concluded that the best map is the one generated with the fewest high quality loop closures. This finding can aid in the design of graph-based SLAM algorithms for microcontrollers, which are heavily limited in their memory and processing capabilities.

2 The Problem

The field of robotics is an exciting and rapidly advancing area of research that holds the potential to revolutionize the way we interact with our environment. One of the most fascinating problems in robotics is the task of autonomous mapping, where a robot is required to explore and map an unknown environment without human intervention. To achieve this, Simultaneous Localization and Mapping (SLAM) algorithms are used to process sensor data from the robot's environment and produce an accurate map of the area.

Loop closures are an essential feature of the SLAM algorithm that help to create an accurate map of the environment. In essence, a loop closure is when the robot returns to a previously visited location and recognizes it as such, allowing the SLAM algorithm to correct any errors that may have occurred during the robot's initial traversal of that location. However, errant loop closures can have significant consequences for the accuracy of the resulting map.

In this study, the aim is to investigate the importance of loop closures in SLAM and provide insight into the effects of inaccurate closures on the resulting maps. This is a crucial area of research, as inaccurate maps can lead to significant errors in subsequent robotic operations, such as navigation or object recognition. By better understanding the role of loop closures in SLAM and the consequences of inaccurate closures, more robust and accurate SLAM algorithms can be developed that can help to advance the field of robotics and its real-world applications.

For this problem, a robot is created with the ability to efficiently and effectively map an unknown environment using odometry and light detection and ranging (LIDAR) sensors. LIDAR and odometry are required to work in tandem to keep the robot up to date with its environment and to know its location within the environment during navigation. An important point to note in this experiment is that this experiment is conducted in a virtual environment, but the sensors are virtualized based on properties of the physical devices. As such, they will have variability in their accuracy. For instance, the robot used is the Jackal from Clearpath Robotics, and the LIDAR is a SICK TiM-511, which has a max range of 10 meters.

Figure 1. Jackal from Clearpath Robotics and the SICK TiM-511 mounted on top

The practical implementation of robotic systems is often limited by the capabilities of the hardware that is currently available. These real-world limitations provide a valuable opportunity for researchers to conduct experiments that are applicable to the future applications of Simultaneous Localization and Mapping (SLAM) algorithms in non-virtualized environments. From there, to investigate the role that loop closures have on the accuracy of the resulting map, an occupancy grid map was built and compared to analyze the impact of varying numbers of loop closures on the accuracy of the SLAM algorithm. Occupancy grid maps are a common tool used in robotics to represent the environment in a

discrete and structured manner, where each cell in the grid is either occupied or unoccupied. By comparing occupancy grid maps generated with different numbers of loop closures, impact of loop closures can be quantified, and the accuracy of the resulting map can be determined.

From there, to investigate the role that loop closures have on the accuracy of the resulting map, the occupancy grid maps are built and compared, and the final built map of the environment is overlaid on the actual room to display the accuracy of the SLAM at varying numbers of loop closures.

3 Why the SLAM Problem is Interesting

This problem is interesting because the theoretical solution of SLAM concludes that it is a very effective way to map an environment and navigate through it smoothly.[5] Moreover, in a world full of continuous environments where agents have the ability to add obstacles or move them from their original positions in the environment, solving this problem ensures the robot always has the latest iteration of the environment map which will keep the navigation precise.

Although the simultaneous localization and mapping (SLAM) problem is undeniably fascinating and has attracted significant research efforts, it still poses a major challenge without a definitive resolution. It is true that some implementations in controlled laboratory settings, where the environment is static and predictable, have been largely resolved. However, in complex and dynamic environments where traditional odometry methods are not reliable, mapping remains an elusive problem to solve. This is particularly evident in highly variable environments where unforeseen changes occur frequently.

To illustrate this, one can refer to the research conducted by Google scientists on LIDAR SLAM algorithms. While their study produced promising results, they were unable to devise a solution to the lifelong mapping problem, which is a significant obstacle for long-term autonomy. Their algorithm could not effectively calculate environment changes, limiting its applicability in scenarios where such changes occur frequently[13].

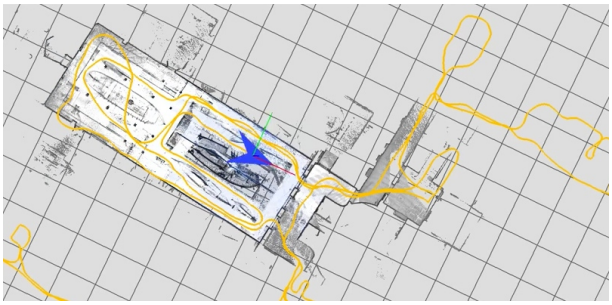

Figure 2. Look at Google's Cartographer autonomously mapping a room

The ultimate objective of lifelong environmental modeling is to equip autonomous devices with the ability to predict the probability of features within an environment remaining constant or undergoing changes over an extended period of time. This capability would represent a significant advancement of simultaneous localization and mapping (SLAM) technology and hold tremendous potential for various industries. With the ability to create and update maps of their environment on-the-fly, autonomous robots would no longer need to constantly re-scan the same area, resulting in more efficient and cost-effective operations.

Lifelong environmental modeling would involve the use of sophisticated algorithms and techniques to detect and track changes in the environment, such as the appearance or disappearance of objects, alterations in lighting conditions, or modifications to the terrain. By continuously analyzing the data gathered by sensors and other sources, the autonomous device could predict whether a particular feature is likely to persist or change in the future, thereby enabling it to update its map accordingly.

The implications of this advancement are vast, as it would revolutionize many industries that rely on autonomous robots, including manufacturing, logistics, transportation, and healthcare, to name just a few. By reducing the need for frequent re-scans and enabling dynamic updates of maps, lifelong environmental modeling would enhance the efficiency and accuracy of autonomous operations, leading to significant cost savings and improved outcomes.

With all that said, SLAM algorithms have a lot of applications, and mapping an unknown environment has yet to have a set solution. By determining how important closures are, it can help guide development for those continuous environments where determining closures remains difficult. The fact that the problem is a mystery makes it compelling to attempt to solve.

4 Introduction to SLAM

The simultaneous localization and mapping (SLAM) problem has been a heavily researched and discussed topic in exploration robotics for decades, and its final solution is something that continues to be elusive. Durrant-Whyte (2006)[5], insists that although SLAM has some solutions, there remain substantial issues in practically realizing more general solutions for the many situations in which rich mapping may be required. Their argument for this centers around the idea that SLAM solutions can be useful in a variety of fields from underwater, land, or airborne. Mapping is necessary for all forms of environments, and the varied environments require an approach that is flexible enough for all of them. SLAM should be general, and not perform significantly worse underwater compared to land. From there, Durrant-Whyte goes on to discuss some of the histories around the SLAM algorithm, from the 1986 IEEE Robotics and Automation Conference

where the beginnings of robotics and AI commenced, to the 1995 International Symposium on Robotics Research where the SLAM acronym was coined. Durrant-Whyte spends the rest of the paper diving into a tutorial on SLAM and how the current implementation works. This includes an explanation of SLAM using an extended Kalman filter, and SLAM using Rao-Blackwellized particle filters. In both cases, probabilistic models are key parts of the math that allows for mapping unknown environments.

In a later paper, Durrant-Whyte and Bailey (2006)[1], discuss the importance of working on the algorithms behind the method, as it will allow for large-scale implementation of SLAM, which is one of the biggest unsolved parts of the problem. They point out that the assumption that the world can be modeled by discrete landmarks doesn't hold in unstructured outdoor environments, and the process to eliminate redundant and moving landmarks is a necessary challenge to overcome to find a real solution to the problem. An approach to this problem, according to Durrant-Whyte, is reducing computational complexity by using a graph-based approach with poses, or states, dictating the environment. As a result, an area can be mapped using partitioned updates rather than the naive implementation where new observations update all vehicle and map states every time a measurement is made. Alongside this advice, the author makes clear that for every problem, there exist at least two unique solutions that have varied advantages and disadvantages to continue to work on.

To further evaluate the accuracy of the SLAM algorithm, final built map of the environment was overlaid onto the actual room. This visualization helps to display the accuracy of the SLAM algorithm and the effectiveness of the loop closure mechanism. By analyzing the differences between the actual room and the built map, one can determine the level of accuracy that has been achieved by the SLAM algorithm with different numbers of loop closures.

5 Applications of SLAM

But to go back in time is a necessary exercise, seeing as exploration robotics can still be taught in the rudimentary form devised by Camillo Taylor and David Kriegman in 1968[14]. In their work, the authors discuss the idea of advanced mobile robotics and the idea of motion planning algorithms for odometric and sensing uncertainty while taking a path. A path, in exploration robotics, refers to the way a robot moves through space in an unknown environment. In their case, as the robot takes a path, it can use ultrasonic sensors, landmarks, and Djikstra's algorithm to explore an environment and create a relational map from a series of local maps created by the robots. However, at this time, no such sensors allowed for the creation of these local maps, leaving the problem to be solved for a future where such sensing and computing power could exist.

An additional foundational piece is the use case of robotic mapping. Thayer (2002)[15], writes that hazardous environments provide a huge opportunity for exploration robotics because humans are unable to map environments that they cannot reach on their own. For that reason, major government agencies have been working on solving the mapping problem, such as the Artisan and Pioneer projects, which use a stochastic element for their exploratory algorithm to determine the fastest way to visit important landmarks, similar to a traveling salesperson problem. One of the areas that they stressed was the fact that there are numerous military scenarios in which mapping and exploration systems are incredibly valuable, as they would help with exploring change on the battlefield and coordinating movements. This would greatly speed up the response time of any military that possesses this technology.

6 Early Approaches

To solve the problem that everyone wanted to solve, Davison's papers on Mono-SLAM[4] were instrumental because the algorithm used cameras, and the image features were taken to represent landmarks on a map. This leveraged the unique areas of spaces to create image patches that can serve as long-term landmarks, and the images were able to be as small as 11 x 11 pixels, saving on limited computational resources. This method also employed Bayesian filtering, which encompasses the extended Kalman Filter, to update beliefs about an unknown state given current observations. At the time, cameras were unused despite providing so much valuable data, and MonoSLAM used the 3D representation provided by a high-frame-rate camera to create a feature-rich map using a Bayesian filtering "top-down" approach for visual SLAM implementations. Towards the end, the authors shifted towards showing the results of the procedure, which was first shown off in interactive augmented reality rather than robotics. In their demonstration, MonoSLAM is used to estimate the hand-held camera's motion in real-time from an image stream, and then feed that information into the rendering engine to convince the user that they are adding virtual pieces to their physical environment. The important part the author stressed is that this could take place at 30Hz, giving a more believable result on the 1.6 GHz processor.

The EKF SLAM techniques that arose as a derivative of the MonoSLAM approach allowed for many advances in autonomous mapping robotics. But, as Tim Bailey (2006)[2], points out, EKF filtering is eager to drift further into error due to mistakes in the covariance matrix it is making. It also has a tendency to need a lot of resources, because it is iteratively making a matrix of landmarks and poses, without discarding bad or irrelevant data. The author concludes that the algorithm is inconsistent under all circumstances, but inconsistency isn't always dire, since if it is small enough it is entirely manageable. The bigger problem, according to

Bailey, is that EKF-SLAM implementations make it difficult to notice an error, and are therefore usually not correctable. It can be prevented if the data is linearized, but heading-uncertainty and error is a major failing of the most popular solution to the problem of mapping the unknown.

7 Improving the Algorithms

Due to the downsides of EKF SLAM, and the need for mapping larger areas, the work by AR researchers Georg Klein and David Murray in 2007[8], adopted parts of previous SLAM techniques to track and map features of an area. The novel idea in their work, however, was to rely on real-time pose estimation using Bundle Adjustment instead of Bayesian filtering, which created a much denser map of lower-quality features. In turn, this provided much better performance and a smoother map. Grisetti's 2011 paper[9] then builds on Bundle Adjustment and provides a framework for optimizing graph-based SLAM. This allows any user to access Graph SLAM, feed in 3D or 2D data with poses and landmarks, and create graphs using ROS. Once again using poses, which are essentially rotations and positions in the unknown environment that the robot is in, and then optimizing the map by optimizing the graph of poses. This framework improved on EKF SLAM, and provided a way for even hobbyists to potentially incorporate Graph SLAM into their own robots.

FastSLAM is an efficient and reliable algorithm for solving the simultaneous localization and mapping (SLAM) problem. The algorithm utilizes a particle filter to sample paths. Each particle uses an N number of extended Kalman filters (EKFs), which indicate the N number of landmarks in the path. Montemerlo explains in his paper[11] that a potential way to improve the fastSLAM algorithm is by having the proposal distribution depend on the most recent sensor data in addition to the motion estimate. Specifically, using linearization on the motion and measurement models in order to calculate the proposal distribution in closed form. This improvement strives to increase the efficiency of the samples, especially in noisy environments, and is referred to as fastSLAM 2.0.

Although simultaneous localization and mapping (SLAM) methods extract maps from a sequence of pictures taken from the agent's sensors/cameras, they are accurate only in the local area and fail to provide clear mapping on the global scale, as mentioned in the paper[16]. Loop closures help eliminate this problem by comparing new keyframes with old keyframes to see if it can find a match or detect an environment that has already been visited, and thus adjust its position on the map along with updating the map's landmarks. This method is called bundle adjustment as it utilizes a First-In-First-Out (FIFO) sliding window where keyframes are stored and compared to each other and after a period of time, outdated keyframes get deleted to create space for the latest keyframes. The size of the sliding window can range between 5 to 8 keyframes and they are used for estimating the pose.

The issue of estimating a collection of poses from pairwise relative measurements is known as pose graph optimization (PGO) (Carlone 2016)[3]. PGO is very common in simultaneous localization and mapping (SLAM) algorithms. PGO works after a loop closure occurs by estimating robot poses by solving nonconvex problems. The algorithm discussed in (Youyang 2020)[17] proposes PGO should estimate relative poses, instead of absolute poses, to acquire the best pose graph possible and calls it the incremental pose graph optimization algorithm. One of the advantages of this algorithm is its ability to deal with outliers using loop closures.

8 Future of SLAM Algorithms

It is clear that although many approaches to autonomous mapping exist, the problem of SLAM remains unsolved because large-scale mapping remains difficult. Yet, it may be the most important application of SLAM, because in environments like Mars or the Moon, which accelerated the development of SLAM, is feasible only by mapping robots. As EKF filtering became more commonly used, the idea of using multiple robots to help with mapping was proposed by Rekelitis and Dudek (2001)[12]. In this approach, localization would happen using positions of other robots, which would help correct the pose estimate during mapping to remove uncertainty of the resulting map. A more recent approach to multi-robot SLAM, came 20 years later, in the form of a Swarm SLAM, which employs multi-robotic mapping to provide a system that is highly scalable, adaptable to unexpected changes, and resilient to failure in hostile environments (Kegelirs, 2021)[7]. This decentalized method is promising because it can create large-scale maps using current approaches of SLAM but combined memory for a large number of poses and loop closures.

With the massive space exploration efforts happening in our present world, multiple space robots have been to our neighboring planet Mars performing extensive surface exploration and uncovering the complex terrain of the red planet. The EKF-SLAM algorithm has been an essential piece in the navigation puzzle for these robots, because it has the ability to accurately highlight landmarks and map out the local area using the sensors and cameras attached to it. In addition, Zheng [18]presents an improved version of the EKF-SLAM algorithm where the resulting map from the algorithm is combined with multiple local submaps to help reduce the computational complexity and refine the computational efficiency. The submaps mentioned are constructed independently for the detected landmarks, where each submap shows a small number of landmarks. The use of submaps works towards reducing the number of feature points used in each

filtering cycle, which in turn, reduces the computational complexity of the EKF-SLAM algorithm.

9 Overview of How SLAM Works

As detailed in the previous sections, there are many existing SLAM algorithms. But, broadly speaking, these algorithms can be classified into two main groups: filtering algorithms and smoothing techniques.

Filtering algorithms model the mapping problem as an online state estimation process, where the robot's state and environment are updated as new measurements are attained. This group of algorithms includes well-known methods such as the Extended Kalman Filter (EKF) and the Particle Filter. These algorithms work by predicting the robot's state and environment using previous measurements and then correcting these predictions as new measurements are acquired.

The second group of algorithms, smoothing techniques, estimate the full robot trajectory from the complete set of measurements, taking into account past measurements. Graph-based SLAM is a well-known example of this group of algorithms. Graph-based SLAM represents the environment as a graph where the nodes represent robot poses and the edges represent the measurements obtained between those poses. The goal of graph-based SLAM is to find the most likely configuration of poses and measurements that explains the sensor data.

For the purposes of this experiment, a graph-based SLAM algorithm was utilized, as it uses smoothing techniques that take into account all available measurements. This has the advantage of being able to integrate information from a wide range of sensors and sources, making it suitable for complex environments. By using a graph-based SLAM algorithm, this experiment can effectively evaluate the impact of loop closures on map accuracy.

In the case of the approach employed here, the SLAM algorithm will be fed position data using the Jackal from Clearpath Robotics' onboard odometry, which is composed of wheel encoders and an IMU for displacement estimation. These encoders will give the exact distance that a wheel has moved, allowing for semi-reliable odometry data. For the environment data, a SICK TiM-511 LIDAR will be used to generate a point cloud that can help with the visualization of an environment that will then be flattened to a 2-D representation. This 2-D map is what the SLAM algorithm is going to be working on, and the noisy LIDAR data will be cleaned by SLAM to return a reliable map of a room that was previously unknown to the robot.

An important note is that although it has been described as a continuous update, reformulation is computationally complex, so sub-mapping is used as described by Bailey and Durrant-Whyte, "Submapping methods exploit the idea that a map can be broken up into regions with local coordinate systems and arranged hierarchically. Updates can occur in a local frame with periodic interframe updates".[1]

Moving to talk more specifically about the algorithm, the smoothing SLAM algorithm using will employ Pose Graph Optimization (PGO), which has become the de facto standard in autonomous mapping robotics. With PGO, a graph is created where nodes represent the robot's position at different points in time, and each node has an edge that connects it to the larger graph. Using the works detailed by Tan and Murphey, our PGO will be accelerated, "our proposed methods update pose estimates by solving optimization sub-problems in closed form. Furthermore, we present methods that significantly accelerate the rates of convergence with no loss of theoretical guarantees".[6]

At the start, the real and estimated robot positions on the pose graph are right on top of one another. As the robot takes measurements and moves, edges are created to connect those nodes, and the edge's "tension" is related to the certainty of the data. These edges appear between two points either based on feedback from the odometry, $x_i^{-1}x_{i+1}$ or from an observation $x_i^{-1}x_j$. Higher confidence leads to a more tense edge. As the robot continues around the room, it adds nodes and their edges to the pose graph, creating a map of constraints that are connecting each of the nodes. As this map is constructed, everything is kept at its nominal length.

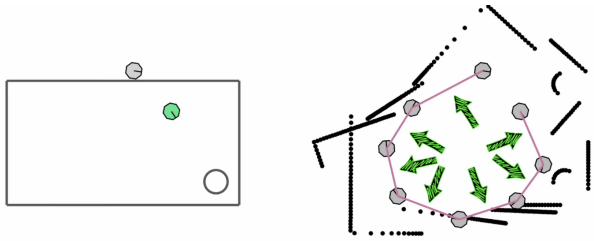

Figure 3. Placing points and defining edges

Eventually, the robot has traveled the entire environment space such that the first and last nodes are seeing the same environment from their LIDAR data. When this happens, an edge is created between the first and last nodes, which creates a loop closure. A loop closure is incredibly important because it injects tension into the system, pulling edges together that are at varying levels of tension. This tension will bring together a more accurate map because the first and last nodes should have ended up being in the same spot on the pose graph if everything was accurate. This loop closure is what has helped create a map that is resilient to noise.

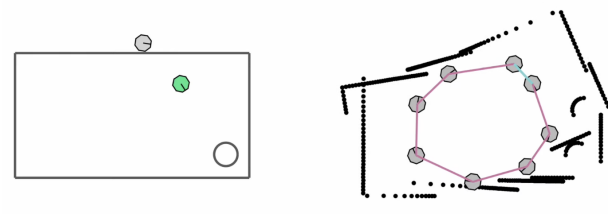

Figure 4. Looped back to original spot, creating first loop closure

Now, the robot can choose to continue driving around to continually improve the map with more loop closures. The math behind these loop closures is part of the "back-end" of the system, where the graph is getting optimized. The optimization math depends on the observation edges, $x_i^{-1}x_j$, with the goal of minimizing the edges by coming up with new configurations for x_i and x_j.

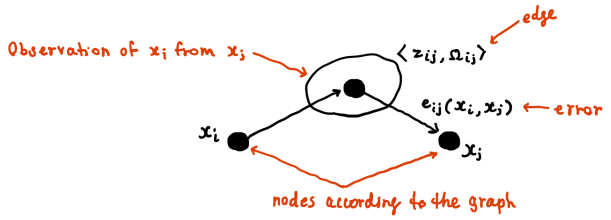

Figure 5. Minimize the arrows between x_i and x_j by coming up with new configurations for them.

This system gets optimized using a least squares error minimization, as shown in the following expression:

$$\hat{x} = \underset{x}{argmin} \sum_{ij} e_{ij}^T(x_i x_j)\Omega_{ij} e_{ij}(x_i x_j) \tag{1}$$

$$= \underset{x}{argmin} \sum_{k} e_k^T(x)\Omega_k e_k(x) \tag{2}$$

With the state vector being given by $x^T = (x_1^T x_2^T ... x_n^T)$, where each component of the vector being a node of the graph. These nodes are used to draw the optimization and create a graph that is corrected for error in odometry or any sort of drift that may be introduced by the system.

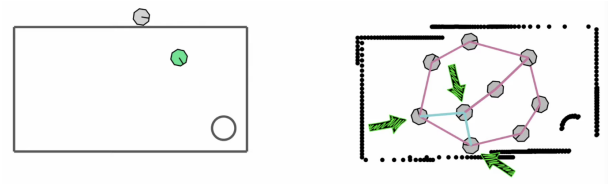

Figure 6. Many loop closures made and making more through observation

Once all of the data has been collected by the SLAM algorithm described above, a Probabilistic Occupancy Grid can be constructed. In a Probabilistic Occupancy Grid, the spatially-related data is broken into a grid, and each grid in the cell is filled in based on whether there is data in that spot. Based on how sure there is to be data in a location, that grid location is shaded in darker.

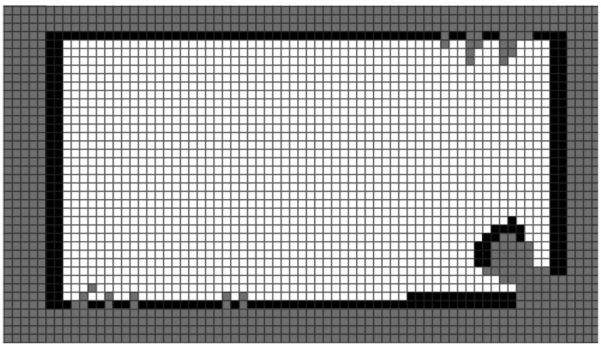

Figure 7. Illustration of a Probabilistic Occupancy Grid

10 The Virtualized Environment

In order to effectively evaluate the importance of loop closures in SLAM algorithms, it was essential to choose a test environment that presented a range of challenges for the robot to navigate. It was important to select a space that would highlight the limitations of the robot's sensors and the potential for drift in its odometry readings, as well as provide ample opportunities for loop closures to occur.

To achieve this, a living room floor plan was chosen as the test environment. This choice was informed by several factors, including the presence of door frames and irregular hallways, which are known to pose challenges for robots navigating in real-world settings. Additionally, the living room floor plan provided a large empty zone in the center of the space, which allowed to test the accuracy of the SLAM algorithm in an area with few obstacles.

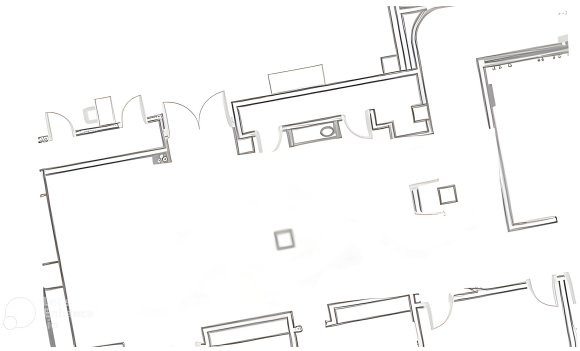

Figure 8. Room layout for the robot to explore

Alongside that, in order to create an accurate map of the living room environment, it was crucial to use sensor

data that was compatible with the chosen SLAM algorithm. The virtual sensor stack used to collect LIDAR data from the simulated robot was specifically designed to produce a 2-D representation of the environment, with each LIDAR scan outputting a collection of points that varied in distance from the measurement point. By choosing a 2-D floor plan for the test environment, it ensured that the LIDAR data collected by the virtual robot would be easily translatable to the mapping algorithm. This meant effective integration of the sensor data into the SLAM to create an accurate map of the living room environment.

11 Experimental Model

To implement the graph-based SLAM to actually map the virtual room detailed in the previous section, MATLAB was used to virtualize the robot and sensing. This was achieved using the Navigation Toolbox[?] add-on for MATLAB R2023a. This add-on allowed for the creation of a lidarSLAM object that could interpret the previously described floor plan as an explorable environment. It also accepted the parameters of the sensor to create a realistic experiment.

```
load('offlineSlamData.mat');
maxLidarRange = 8;
mapResolution = 20;
slamAlg = lidarSLAM(mapResolution,
    maxLidarRange);
```

From there, the modifable values were added that allowed for the experiment to have variable loop closure settings. A higher threshold would reject a greater number of false positives in the closure identification process, and would therefore limit the number of closures. Along with that, a higher closure radius allows the algorithm to look for a wider range of options when looking at the current pose estimate for the closures.

```
slamAlg.LoopClosureThreshold = 210;
slamAlg.LoopClosureSearchRadius = 8;
```

After that was set, the algorithm was implemented in MATLAB that allowed for the robot to dynamically explore the environment based on what it saw with the LIDAR.

```
for i=1:length(scans)
    [isScanAccepted,
        loopClosureInfo,
            optimizationInfo]
                = addScan(slamAlg,
                    scans{i});
    if ~isScanAccepted
        continue;
    end
    show(slamAlg, 'Poses', 'off');
    hold on;
    show(slamAlg.PoseGraph);
    hold off;
    firstTimeLCDetected = true;
    drawnow
end
```

If there was promise in a certain direction, that is the direction that the robot would take. However for the chosen floor plan, the path that the robot takes iteration after iteration should be static, as the optimal path can be determined. It was up to the accuracy of the map being constructed for the robot to discover that path, shown in the figure below.

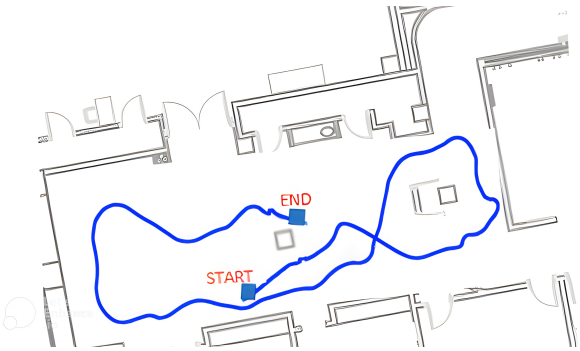

Figure 9. Optimal trajectory for the robot to take around the floor plan

In this study, the goal was to understand the impact of loop closures on the accuracy of the resulting map produced by the SLAM algorithm. To achieve this, three main charts were created to visualize the performance of the algorithm.

The first chart presented in the study depicts the resulting map generated by the exploratory robot as it navigates through the living room environment. This chart also shows the number of loop closures that were required to create the final map. To create the map, the robot uses its LIDAR sensor to collect data on the environment, which is then processed by the SLAM algorithm to generate an occupancy grid map of the environment. The number of closures shown on the chart represents the points at which the robot revisits areas of the environment it has already explored in order to refine its map.

Each red edge on the chart represents a loop closure, which corresponds to a point at which the robot revisited a previously explored area. By visually analyzing the number and location of closures, results can be drawn on the effectiveness of the SLAM algorithm in accurately mapping the environment. In particular, observe the impact of closures on the resulting map and assess whether they improve the accuracy of the algorithm or introduce errors.

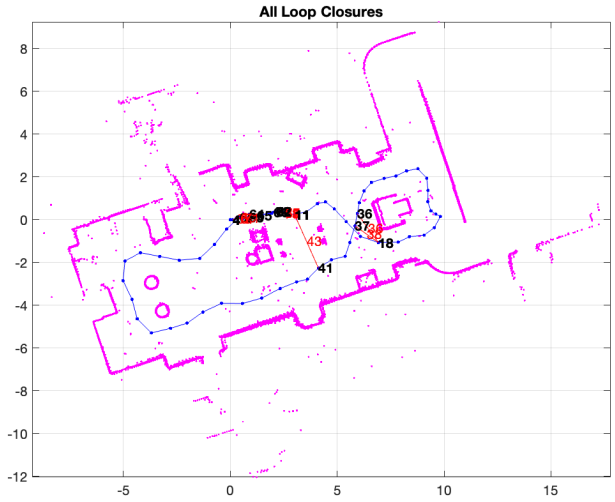

Figure 10. All Loop Closures at LoopClosureThreshold=210

The second chart in the study presents the final map of the living room environment, along with the trajectory taken by the robot. This map was generated using the same SLAM algorithm and LIDAR sensor data as the previous chart, but it abstracts away the number of loop closures required to create the map. This allows to more easily visualize the overall accuracy of the SLAM algorithm in creating a detailed and accurate map of the environment.

Figure 11. Final Built Map of the Environment with Superimposed Trajectory for LoopClosureThreshold=210

The final map produced by the SLAM algorithm is critical in generating the occupancy grid, which is the third chart presented in the study. To create this grid, the optimized scans and poses generated by the SLAM algorithm are used to represent the environment as a probabilistic occupancy grid. This grid provides a visual representation of the likelihood of the robot encountering an obstacle at a given point in the environment.

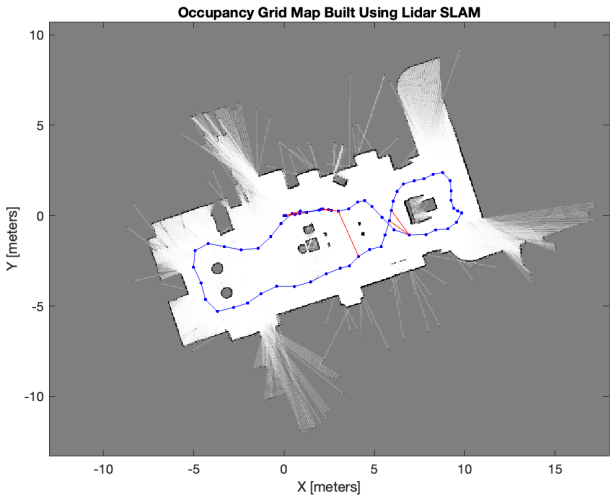

Figure 12. Probabalistic Occupancy Grid for LoopClosureThreshold=210

By varying the number of loop closures used in the SLAM algorithm, one can observe the effect that this has on the resulting probabilistic occupancy grid. This allows for an assessment of the impact loop closures have on the accuracy of the SLAM algorithm in creating an occupancy grid that accurately represents the environment. Ultimately, the goal of the SLAM algorithm is to create a probabilistic occupancy grid that accurately represents the environment, and this study provides a visualization of the effectiveness of the algorithm in achieving this goal, as related to closures.

In this study, the experimental process detailed above was repeated at 5 variations of loop closure sensitivity, modified by:

```
slamAlg.LoopClosureThreshold
    = 310; % 0 loop closures

slamAlg.LoopClosureThreshold
    = 260; % 4 loop closures

slamAlg.LoopClosureThreshold
    = 210; % 8 loop closures

slamAlg.LoopClosureThreshold
    = 190; % 12 loop closures

slamAlg.LoopClosureThreshold
    = 160; % 20 loop closures
```

Varying the number of loop closures in this way allowed the experiment to apply to memory-starved applications such as

SLAM on microcontrollers, where the number of datapoints and possible loop closures are low, and also aggressive graph-based SLAM algorithms where there may be too many.

12 Results

After running the SLAM on MATLAB at the varied thresholds, probabilistic occupancy grids at each number of loop closures were created. These occupancy grids have the path and the number of loop closures, marked in red, layered on the map.

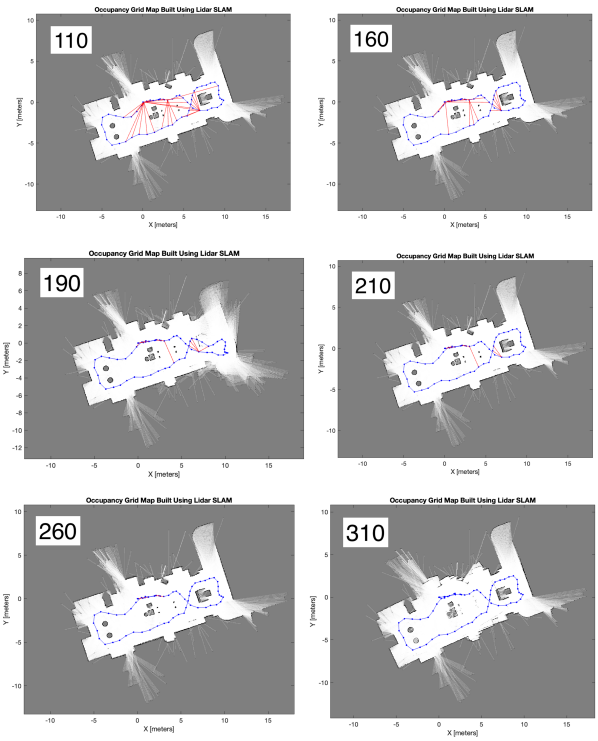

Figure 13. All Probabilistic Occupancy Grids for the tested loop closure thresholds

13 Analysis of Results

Upon analyzing the generated probabilistic occupancy grids at varying levels of loop closures, it becomes apparent that the number of closures had a significant impact on the accuracy of the resulting maps generated by the graph-based SLAM algorithm used in this experiment.

In order to determine the best possible outcome, it is necessary to compare the different maps against each other and to the pre-existing floor plan. The selection criteria for choosing the best map is based on which map most closely resembles the floor plan while using the fewest number of closures. This is an important consideration in SLAM algorithms as minimizing the number of closures leads to less computational effort and a more efficient mapping process.

Based on the above criteria, it appears that the threshold value of 210 provides the best outcome. This map can be visually compared with the others by overlaying it on the actual floor plan and comparing it to the maps generated by the other threshold values.

Figure 14. 210 (Pink) vs. 310 (Blue)

In analyzing the results presented in Figure 14, it is apparent that the inclusion of loop closures greatly impacts the accuracy of the resulting map. Specifically, the map generated with 210 closures is notably superior to the map generated with no closures (310), underscoring the importance of the optimization provided by loop closures in the graph SLAM algorithm. The comparison between these two maps highlights the value of incorporating loop closures to enhance the accuracy of SLAM results, particularly in environments with significant challenges such as door frames, irregular hallways, and large empty zones.

This outcome aligns with the widely recognized understanding that loop closures are a vital component in the SLAM algorithm's ability to accurately estimate a robot's position and the environment. The optimization provided by loop closures enables the SLAM algorithm to correct and update the map in real-time as the robot moves through the environment and collects new data, allowing for a more accurate and comprehensive representation of the environment. Thus, the results presented in Figure 1 serve as a confirmation of the significant impact of loop closures on the accuracy of the SLAM algorithm's map building process.

A key finding of the experiment was the notion that not only the number of closures, but also the quality of the closures can significantly impact the accuracy of the resulting map. This was demonstrated in the comparison of Figure 15, where the occupancy grid generated using threshold 210 with 8 loop closures was compared with the occupancy grid generated using threshold 190 with 12 loop closures.

It was observed that although threshold 190 had more loop closures, the quality of the closures was poor, leading to an increase in errors and inaccuracies in the final map. This was evidenced by the altered path taken by the robot,

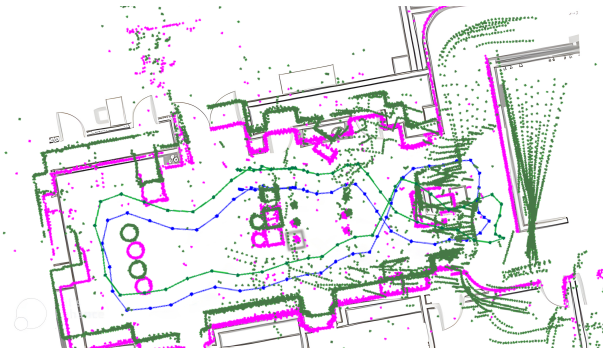

Figure 15. 210 (Pink) vs. 190 (Green)

resulting in a map that was even less accurate than if no closures were made at all. Therefore, it can be inferred that making more closures does not necessarily equate to better accuracy and that it is crucial to ensure that the closures made are of high quality to avoid introducing inaccuracies into the map.

Figure 16. 210 (Pink) vs. 260 (Yellow)

Figure 16 presents a comparison that supports the notion that more loop closures do not necessarily lead to better results than fewer closures, as long as the closures are of high quality. This finding is noteworthy because it suggests that a graph-based SLAM can use fewer closures as long as they are well placed and optimized, and still achieve accurate results. The comparison of the two maps generated using thresholds 210 and 260, respectively, indicates that the two maps are similar in terms of accuracy, with only slight variations around the edges. Threshold 260, which utilized only 4 closures, was able to achieve comparable results to threshold 210, which had twice the number of closures. In fact, in certain areas of the map, the map generated using threshold 260 even surpassed the accuracy of the map generated using threshold 210. For instance, in the challenging hallway area, the map with fewer closures produced better results. This suggests that quality, rather than quantity, is a crucial factor to consider when implementing loop closures in graph-based SLAM algorithms.

The finding that fewer loop closures with better quality can be as effective, if not more effective, than more loop closures has significant implications for the design of graph-based SLAM algorithms, particularly for microcontrollers. Microcontrollers have inherent limitations in terms of memory and processing speed, which can make it challenging to run SLAM algorithms effectively. However, by reducing the number of poses in the optimization process, microcontrollers can be better equipped to perform SLAM with their limited compute resources. This reduction in poses also allows microcontrollers to fit an entire state space onto their on-board flash memory, which enables them to include well-placed loop closures for improved mapping accuracy. This finding can play a crucial role in the design and implementation of SLAM algorithms for microcontrollers, allowing for more efficient and accurate mapping in a range of applications.

14 Conclusion

This paper discussed the problem of autonomous mapping, which is where a robot travels around an unknown environment in order to produce a precise map using the Simultaneous Localization and Mapping (SLAM) algorithm and Light Detection and Ranging (LIDAR) sensors. Concretely, it focused on observing the effectiveness and impact loop closures had in generating an accurate map of an unknown environment for a robot. For the experiment, a graph-based SLAM algorithm was implemented to map the environment, as well as utilizing MATLAB to generate a virtual robot with LIDAR sensors to gather data about the environment.

The results were arrived at by comparing probabilistic occupancy grid maps at different degrees of loop closures. This showed that the quantity of closures had a substantial effect on the precision of the map generated from the graph-based SLAM algorithm. Moreover, the quality of the closures also had a positive impact as the higher quality a closure was, the less number of closures was needed to arrive at the most accurate map. It is safe to say that loop closures provide a significant improvement to the SLAM algorithm for producing the most accurate maps. Finally, the next step for the graph-based SLAM algorithm could be for it to be used in tandem with Extended Kalman Filter (EKF) filtering to further enhance the algorithm's capabilities.

15 Breakdown of Contributions

The numbers in the following list correspond to the numbered sections of this paper:

1. Abdullah
2. Abdullah
3. Vaibhav
4. Vaibhav

5. Vaibhav
6. Vaibhav
7. Vaibhav
8. Abdullah
9. Vaibhav
10. Vaibhav
11. Vaibhav
12. Abdullah
13. Abdullah
14. Abdullah

References

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

[2] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. 2006. Consistency of the EKF-SLAM Algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3562–3568. https://doi.org/10.1109/IROS.2006.281644

[3] Luca Carlone, Giuseppe C. Calafiore, Carlo Tommolillo, and Frank Dellaert. 2016. Planar Pose Graph Optimization: Duality, Optimal Solutions, and Verification. *IEEE Transactions on Robotics* 32, 3 (2016), 545–565. https://doi.org/10.1109/TRO.2016.2544304

[4] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. 2007. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 6 (2007), 1052–1067. https://doi.org/10.1109/TPAMI.2007.1049

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

[6] Taosha Fan and Todd Murphey. 2021. Generalized Proximal Methods for Pose Graph Optimization. arXiv:2012.02709 [math.OC]

[7] Miquel Kegeleirs, Giorgio Grisetti, and Mauro Birattari. 2021. Swarm SLAM: Challenges and Perspectives. *Frontiers in Robotics and AI* 8 (2021). https://doi.org/10.3389/frobt.2021.618268

[8] Georg Klein and David Murray. 2007. Parallel Tracking and Mapping for Small AR Workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. 225–234. https://doi.org/10.1109/ISMAR.2007.4538852

[9] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. 2011. G2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*. 3607–3613. https://doi.org/10.1109/ICRA.2011.5979949

[10]]mathworks MathWorks MathWorks. [n. d.]. Navigation Toolbox. https://www.mathworks.com/help/nav/index.html?s_tid=CRUX_lftnav

[11] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. 2003. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. http://robots.stanford.edu/papers/Montemerlo03a.pdf

[12] Ioannis Rekleitis, Gregory Dudek, and Evangelos Milios. 2001. Multi-robot collaboration for robust exploration. https://link.springer.com/article/10.1023/A:1016636024246

[13] David M. Rosen, Julian Mason, and John J. Leonard. 2016. Towards lifelong feature-based mapping in semi-static environments. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1063–1070. https://doi.org/10.1109/ICRA.2016.7487237

[14] Camillo J. Taylor and David J. Kriegman. 1968. Exploration Strategies for Mobile Robots. https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/t/taylor_exploration.pdf

[15] Scott M. Thayer. 2002. Four Generations of Robotic Mapping and Exploration in Extreme Environments. https://www.ri.cmu.edu/pub_files/pub3/thayer_scott_2002_1/thayer_scott_2002_1.pdf

[16] Yodayoda Yodayoda. 2021. Why loop closure is so important for global mapping. https://medium.com/yodayoda/why-loop-closure-is-so-important-for-global-mapping-34ff136be08f

[17] Feng Youyang, Wang Qing, and Yang Gaochao. 2020. Incremental 3-D pose graph optimization for SLAM algorithm without marginalization. *International Journal of Advanced Robotic Systems* 17, 3 (2020), 1729881420925304. https://doi.org/10.1177/1729881420925304 arXiv:https://doi.org/10.1177/1729881420925304

[18] Bo Zheng and Zexu Zhang. 2019. An improved EKF-Slam for Mars Surface Exploration. https://www.hindawi.com/journals/ijae/2019/7637469/