FastSLAM Implementation on Pioneer 3-DX Robot

1st Ricardo Francisco
IST Number 103093
ricardo.n.francisco@tecnico.ulisboa.pt

2nd Afonso Duarte
IST Number 103913
afonso.g.duarte@tecnico.ulisboa.pt

3rd Carolina Carvalho
IST Number 99523
maria.carolina.carvalho.silva@tecnico.ulisboa.pt

4th David Ribeiro
IST 112202
david.ribeiro.sanjorge@tecnico.ulisboa.pt

***Abstract*—In this project, a SLAM technique known as FastSLAM was developed and implemented. The algorithm was programmed in Python and tested using a microsimulator to verify its correct implementation. Following successful validation, the algorithm was applied to real-world data. For this purpose, ROS was used to record a data set in a ROS bag file, which included the robot's motion data and landmark detections, represented by ArUco markers.**

The ROS data was then integrated to python to enable offline map estimation using the FastSLAM algorithm. After the original version was implemented, additional variations of FastSLAM were developed and tested, including a range-only version and a bearing-only version. The parameters of these variants were tuned to optimize performance, and the results were analyzed to draw comparative conclusions.

***Index Terms*—FastSLAM, Simultaneous Localization and Mapping (SLAM), Particle Filter, ArUco Markers, Range-Only SLAM, Bearing-Only SLAM, Mobile Robotics**

GitHub repository: Githuh source code
Bag files: Google Drive link

I. Introduction and Motivation

Simultaneous Localization and Mapping (SLAM) is a fundamental and challenging problem in the field of robotics. It involves enabling a robot to construct a map of an unknown environment while simultaneously estimating its own position within that environment. This dual process relies on sensor data and detected landmarks, and must account for uncertainties in both the robot's motion and its observations.

This project focuses on the development of a more scalable and computationally efficient SLAM approach — FastSLAM — implemented in Python. The algorithm is tested in a real-world environment, and the resulting performance is evaluated through both quantitative and qualitative analysis. Based on these evaluations, conclusions are drawn regarding the effectiveness and limitations of the algorithm under practical conditions.

II. Methods and Algorithms

A. SLAM Problem

In a static and unknown environment, SLAM aims to estimate the map of the environment as well as the robot's path. The path consists of every pose taken to that moment and the map l is composed of the location of all landmarks.

A pose x_t represents the xy coordinates and orientation of the robot taken in an instance t. According to the motion model, each pose is estimated based on the previous pose and the received control input u_t as seen in (1).

$$p(x_t|x_{t-1}, u_t) \tag{1}$$

Since the landmarks are estimated relatively to the robot, knowing the previous pose, input given to the robot, and its sensor observations at each instance z_t, the SLAM problem can be formulated as follows:

$$p(x_{1:t}, l|z_{1:t}, u_{1:t}) \tag{2}$$

Where the subscript $_{1:t}$ refers to a sequence of values from time step 1 to t.

Thus, (2) represents the probability of a given path and map given all the previous z_t and u_t measurements. Because the pose can be estimated from the robot actuation and sensor observations and the map can be estimated from the sensor position and observations, (2) can be writen as follows:

$$p(x_{1:t}, l|z_{1:t}, u_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}) \cdot p(l|x_{1:t}, z_{1:t}) \tag{3}$$

B. Factored Representation

When it's possible to establish the correspondence of which landmark is being observed (i.e., landmarks are distinguishable) and keep track of the number of observed landmarks K, each landmark estimation problem becomes independent. This is the case for the project since ArUco markers are being used for each landmark. Consequently, the second term of (3) can be decomposed into the product of all independent landmark estimations:

$$p(x_{1:t}|z_{1:t}, u_{1:t}) \cdot p(l|x_{1:t}, z_{1:t}) = p(x_{1:t}|z_{1:t}, u_{1:t}) \cdot \prod_{i=1}^{K} p(l|x_{1:t}, z_{1:t}) \tag{4}$$

At this stage, the estimation process first addresses the robot's trajectory, followed by the conditionally independent estimation of each landmark's position. As a result, the problem is decomposed into K independent landmark estimation tasks in addition to one trajectory estimation task.

C. Particle Filter for Path Estimation

A set of N particles is used, where each particle (denoted by its superscript index $^{[n]}$) maintains an estimate of the posterior distribution $p(x_{1:t}|z_{1:t}, u_{1:t})$. Each particle represents a possible trajectory of the robot, and all particles are stored in the set Y_t. At each time step, Y_t is updated based on the previous set Y_{t-1}, the control input u_t, and the sensor observations z_t. Each particle samples a new pose from the motion model. Let n denote the index of a particle:

$$x_t^{[n]} \sim p(x_t|x_{t-1}^{[n]}, u_t) \tag{5}$$

The new pose is added to the previous path and all the N particles go through this process. Then, the updated set Y_t is obtained by sampling these particles according to the weight factor of each one. Therefore, particles with more weight will be sampled more often as their path guess is more probable. These weights will be discussed later on.

D. Landmarks Estimation

Having addressed the path estimation, the focus shifts to the problem of landmark localization. As previously discussed, each particle maintains its own set of K landmark estimations. Kalman filters are used to represent the probability distributions in the second term of Equation (3). Accordingly, each particle contains its estimated path $x_{1:t}^{[n]}$, along with the mean $\mu_k^{[n]}$ and the covariance matrix $\Sigma_k^{[n]}$ for each landmark k.

Unlike traditional SLAM, which requires updating a $(2K+3) \times (2K+3)$ covariance matrix—where each landmark is represented by its (x, y) coordinates and the robot pose by (x, y) and orientation—FastSLAM updates only K individual 2×2 Gaussian matrices per particle.

This significantly reduces computational complexity from $O(K^2)$ to $O(N \log K)$, making FastSLAM more suitable for environments with a large number of landmarks.

E. Importance Weights and Resampling

Finally, the role of importance weights and their influence on particle resampling are addressed. Each particle n in the set Y_t is sampled with a probability proportional to its associated weight. As a result, particles with higher importance weights are more likely to appear multiple times in the subsequent particle set. The importance weights are computed as follows:

$$w_t^{[n]} = \frac{p(x_{1:t}^n|z_{1:t}, u_{1:t}, i_{1:t})}{p(x_{1:t}^n|z_{1:t-1}, u_{1:t}, i_{1:t-1})} \tag{6}$$

Where $i \in \{1, ..., K\}$ is the index of the landmark being observed. Then, considering that $p(i_t|l, x_t^{[n]})$ constitutes a uniform distribution, (6) can be approximated as:

$$\approx \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(z_t - \hat{z}_t^{[n]})^\top \Sigma^{-1}(z_t - \hat{z}_t^{[n]})\right) \tag{7}$$

Here, $\hat{z}_t^{[n]}$ represents the expected observation given the landmark and particle pose. This expression is a multivariate Gaussian density capturing the measurement likelihood with covariance Σ, and is computationally efficient to implement.

F. Algorithm Block Diagram

Finally, after going through all the steps and significant equations, it is presented a visual representation of how each part fits into the other and the general loop. For a given time instant t, Figure (1) represents the algorithm that updates the particle set.

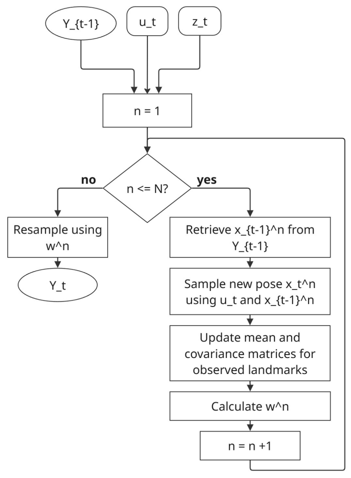

Y_{t-1}
u_t
z_t
n = 1
n <= N?
no
yes
Resample using w^n
Y_t
Retrieve x_{t-1}^n from Y_{t-1}
Sample new pose x_t^n using u_t and x_{t-1}^n
Update mean and covariance matrices for observed landmarks
Calculate w^n
n = n +1

Fig. 1: Simplified FastSLAM Block Diagram.

III. Implementation

A. Data Acquisition

To collect the necessary data in a ROS bag, the `/pose` topic was used to obtain the robot's motion, which is published by the robot's onboard firmware. For landmark observations, a webcam connected to a student's laptop was utilized. The `cv_camera` package [1] was employed to publish the webcam feed to ROS. In order to estimate the pose of the landmarks relative to the robot, the camera first had to be calibrated. This was achieved using the `camera_calibration` package [2] along with a black-and-white checkerboard of known dimensions.

Once calibration was complete, the `aruco_detection` package [3] was used to detect ArUco markers and publish their corresponding IDs and poses, based on a predefined marker dictionary. With all necessary nodes running, a ROS bag file was recorded while navigating through the environment, capturing the data required for subsequent analysis.

B. Microsimulator

The microsimulator was designed with the objective of seamlessly switching between simulated and real-world data. It provides the algorithm with inputs in the same format and structure as those obtained from actual sensor data, thereby increasing the modularity and flexibility of the system. The simulation includes the option to add noise to measurements to further test the capabilities of the algorithm beyond perfect conditions.

C. Algorithm

To implement the algorithm, an object-oriented programming approach was adopted. The `FastSLAM` algorithm was encapsulated in a class structure, where the main object maintains methods for updating the system at each time step. This object contains N particle objects, each of which include K Kalman filter objects for estimating the positions of the landmarks.

At each time step, the algorithm proceeds as follows. First, all particles update their pose using the robot's measured linear and angular velocities, applying first-order Euler integration to update x, y, and θ. If the measured velocities are close to zero, no update is performed, and the particles remain stationary, reflecting the robot's lack of motion.

Next, when observed, from the range, bearing and de id of the landmark, each particle either initializes the corresponding Kalman filter—if the landmark has not been previously detected—or updates its state. Standard Kalman filter logic is applied, with two key exceptions: the prediction step is omitted, since the landmarks are assumed to be static, and Cholesky decomposition is used during matrix inversion for improved numerical stability.

After updating the Kalman filters, each particle's importance weight is computed using a maximum likelihood function based on the measured observations.

After computing the importance weights for all particles, the algorithm performs a resampling step to focus on the most likely hypotheses. In this step, particles with higher weights are more likely to be selected, while those with low weights are likely to be discarded.

A low-variance resampling algorithm is used to efficiently sample particles according to their weights. This helps reduce particle depletion and maintain diversity in the particle set. The selected particles are then deep-copied to form a new particle set, and their weights are reset to a uniform value. This updated set becomes the basis for the next iteration of the algorithm.

D. Parameter Tuning

To ensure the FastSLAM algorithm performed reliably in both simulation and real-world conditions, several key parameters were carefully tuned through iterative testing and evaluation.

- **Number of Particles:** The number of particles was selected to provide a balance between computational efficiency and estimation accuracy. A higher number of particles improves robustness to noise but increases computational cost.
- **Odometry Noise:** The uncertainty in the robot's motion was modeled to reflect the noise in both linear and angular velocity measurements. These values were tuned to capture realistic motion error while avoiding overestimation that could degrade performance.
- **Initial Landmark Uncertainty:** When a new landmark is detected, it is initialized with a moderate level of uncertainty. This allows the filter to remain flexible and adapt quickly as more observations become available.
- **Measurement Noise:** The observation noise, including both range and bearing components, was calibrated based on the characteristics of the camera and marker detection system. These parameters are critical in determining how much trust is placed in each measurement during the update step of the kalman filter.

Fine-tuning these parameters was essential to achieving consistent and accurate mapping results, especially when working with real-world sensor data.

E. Range Only algorithm

After successfully implementing FastSLAM, the challenge of reproducing the same results while having access to less information about the environment was taken. Now, each observation z_t only describes the range to the observed landmark, so there is no information on its bearing relative to the robot.

The way to tackle this obstacle is by introducing the concept of trilateration. This method of localization uses the distances obtained in sequential instances to obtain the coordinates of an observed stationary point of interest. By overlapping the circles with the same radius as the range observed at each moment, the intersection provides the landmark's position.

Nonetheless, the problem was simplified by making an initial guess for the landmark's position: a random point on the circle segment within the camera's field of view. Then, with the consecutive observations, the position was updated.

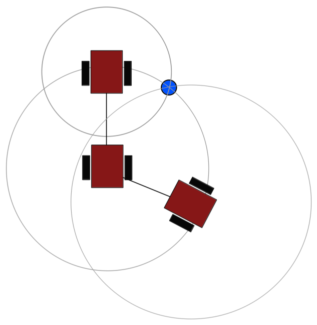


Fig. 2: Trilateration.

F. Bearing Only Algorithm

Having implemented the Range Only algorithm, the logical follow-up action was to experiment with the observation z_t being exclusively the bearing.

In this case, the landmark should be at some point on the vector whose direction was the bearing angle.

Similarly to what was done before, an initial guess for the landmark's position was made: a random distance value within the camera's operational range, using the bearing as direction. Again, with the consecutive observations, that initial guess was updated.

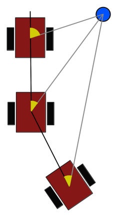


Fig. 3: Bearing Only.

IV. Experimental Results

A. Camera Measurement Characterization

To better assess the reliability of the camera-based measurements, a series of controlled tests were conducted. These tests aimed to estimate the measurement covariance and to determine the operational limits of the sensor in terms of range and field of view.

First, to estimate the covariance of range and bearing measurements, static tests were performed at various known distances. In each test, the camera remained stationary while repeatedly detecting a fixed landmark. By collecting multiple measurements from each position, the empirical covariance for both range and bearing was calculated, providing valuable data for tuning the observation noise parameters in the filter.

In addition to covariance estimation, the effective range and field of view of the camera were also evaluated. To determine the maximum detection range, the camera was initially placed very close to a landmark and then gradually moved away until the marker was no longer detected. For assessing the angular field of view, the camera was slowly rotated until the landmark exited the visible area. These limits are particularly relevant for the range-only and bearing-only versions of the FastSLAM algorithm, which rely solely on partial observation information.

The results obtained were the following:

- **Range Measurement Covariance:** 5.656×10^{-7}
- **Bearing Measurement Covariance:** 1.472×10^{-10}
- **Operational Range:** 0.336 m to 5.36 m
- **Camera Field of View:** 49.56°

The covariance values were applied to the tuning parameters of the observation model; however, the results were poor. This can be explained by the fact that the covariance values were very low, indicating a high level of trust in the observations. While this assumption is valid for a stationary camera, it does not hold true for a moving robot—especially due to the vibrations introduced by the camera's movement. In such cases, the actual covariance values shift, increasing the uncertainty in the measurements. Because the model placed too much trust in the observations under these new conditions, the results drifted significantly from the ground truth.

B. Qualitative Analysis

1) Standard FastSLAM: In the evaluation of the standard FastSLAM algorithm, which uses both *range* and *bearing* measurements, distinct behaviors were observed in its two core components: mapping and localization.

a) Mapping Accuracy: FastSLAM consistently demonstrated high accuracy in mapping the environment. The estimated landmark positions were well aligned with the ground truth, and the actual landmark positions typically fell within the *covariance ellipses* of the estimates, as seen in Figure 4. This suggests that the algorithm's uncertainty modeling was effective, and that the particle filter combined with local EKFs was able to converge reliably on the true map.

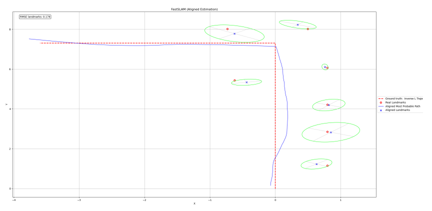


Fig. 4: SLAM estimation of an L-shaped trajectory

b) Localization Performance: In contrast, the localization (trajectory estimation) was slightly less precise. The estimated trajectory exhibited a small but consistent offset from the ground truth. While this does not constitute a complete failure in localization, it was less accurate than expected. Nonetheless, FastSLAM showed an ability to correct its drift over time. This is a typical feature of particle-filter-based SLAM methods: while motion noise can cause drift, landmark observations enable periodic re-anchoring of the trajectory estimate.

Interestingly, it was observed that the algorithm performed worse during straight-line motion, a somewhat counterintuitive finding. Generally, turning maneuvers are expected to introduce more uncertainty due to greater motion complexity. However, in our case, the decline in localization performance during straight-line segments can be attributed to poor landmark observation quality. When the robot moves straight, it often faces away from nearby landmarks or observes them at steep angles. Coupled with the significant camera vibration in our setup, this reduces the reliability of landmark detection. Consequently, the algorithm receives weaker corrections during these intervals, allowing the trajectory to drift more than during turns, where landmark observations are often more robust. This highlights the importance of map quality: the less time the robot moves without observing landmarks, the better. In straight paths, the robot often moved without seeing any, which reduced accuracy. More landmarks and longer observations lead to better performance.

c) Impact of Post-Processing: A contributing factor to the observed localization inaccuracy is the post-processing step applied. To compare results with ground truth, it was performed a *Procrustes transformation* to align the estimated landmarks to their true positions, then applied the same transformation to the estimated trajectory. While this ensures alignment of the map, it does not guarantee accurate alignment of the trajectory, for two main reasons:

- The transformation minimizes error over landmark positions, not the trajectory.
- The true trajectory may not be perfectly preserved under the transformation, especially in the presence of unmodeled drift or noise.

One practical side effect of this is that the estimated trajectory often fails to correctly match the robot's starting position. For instance, in runs with square trajectories, FastSLAM recovered the shape of the path well but failed to start at the origin or close the loop, as seen in Figure 5. This behavior was consistent across trials and likely results in part from the misalignment introduced by the transformation.

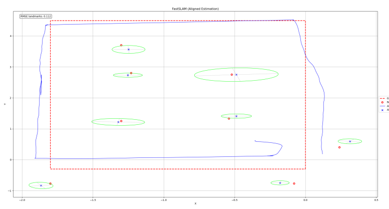


Fig. 5: SLAM estimation of a square trajectory

d) Sensor Setup Limitations: An important source of noise not accounted for in the system model is the physical vibration of the camera. In the setup, the camera was attached to the robot using tape, which led to significant vibration during motion. This caused the camera to frequently lose focus while the robot was moving, resulting in blurred frames and degraded observation quality. This was one of the main contributors to localization error. For better performance, the camera should be mounted more securely using a rigid, vibration-damping solution to ensure stable, clear visual input during operation.

2) Bearing-Only and Range-Only FastSLAM: Similar qualitative observations can be made for the alternative FastSLAM implementations that use only a single type of sensor measurement. While these versions inherit many of the characteristics of the standard method, they also introduce unique challenges and limitations due to their reduced observation models.

a) Range-Only FastSLAM: Interestingly, the range-only implementation performed surprisingly well in terms of localization. The estimated trajectories were often comparable to those produced by the standard FastSLAM, suggesting that even without bearing information, the algorithm was still able to maintain accurate pose estimates. This is likely due to the fact that the relative distances to landmarks provide sufficient constraint on the robot's position, particularly when multiple range observations are available from distinct angles or at different time steps.

However, the mapping performance was noticeably weaker. Without bearing information, the algorithm struggles to resolve the angular positioning of landmarks, leading to consistent but systematically misaligned landmark estimates. The landmarks tend to cluster in geometrically plausible patterns relative to one another, but their absolute placement in the environment is often incorrect. As a result, when the Procrustes transformation is applied, the alignment of the estimated trajectory to the ground truth appears accurate, even though the map itself is offset. This reflects a decoupling between relative geometry and absolute accuracy, which is a known limitation of range-only SLAM.

b) Bearing-Only FastSLAM: The bearing-only implementation, in contrast, demonstrated the weakest performance in both localization and mapping. Without range information, the algorithm relies solely on angular measurements to infer both pose and landmark positions. This leads to severe ambiguities, especially when landmarks are distant or clustered closely in angle, making it difficult for the algorithm to resolve their relative distances or distinguish between them.

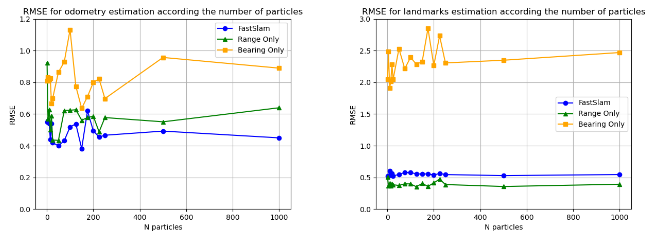
Consequently, the estimated maps showed significant distortion, and the trajectories exhibited greater drift and instability. Because the algorithm cannot triangulate landmark positions accurately, errors in the map propagate directly into localization estimates. The system's ability to recover from such errors is also reduced, as bearing-only observations provide limited corrective power, especially when the robot is not actively turning or when landmarks are observed at shallow angles.

C. Quantitative Analysis

1) RMSE: The root-mean-squared error (RMSE) is a standard metric used in model evaluation. It assess the error between observations and model predictions.

This indicator was used to evaluate the accuracy of the algorithm in terms of both path and landmarks position estimation. Therefore, applied to the problem under study, it evaluates the difference between the ground truth and their associated predictions.

The purpose of using this metrics was to try to find the optimal number of particles as well as the best technique between the three implemented.



(a) RMSE for robot's path estimation.

(b) RMSE for landmarks position estimation.

Fig. 6: Comparison of RMSE for robot path and landmark position estimation.

The original FastSLAM algorithm demonstrated the best performance for localization, while the Range-Only method excelled in mapping. In contrast, the Bearing-Only approach consistently performed the worst across all scenarios. This outcome is likely due to the nature of the dataset, which was collected along a mostly straight-line path with some minor turns and sufficient observation time—conditions that favor the original FastSLAM's strengths. Additionally, no performance divergence was observed with varying numbers of particles, possibly because the measurements contained little uncertainty, leading to consistent results regardless of particle count.

Another contributing factor may be the deterministic nature of the motion model used in the experiments. Minimal or underestimated odometry noise causes limited particle dispersion during prediction, reducing the benefit of using larger particle sets. This likely explains why increasing particle numbers did not improve performance as expected. Interestingly, in the controlled conditions of our micro-simulator environment, an exponential decline in RMSE with more particles was observed, reinforcing the notion that the limitations seen in real-world tests stem from dataset characteristics rather than algorithmic shortcomings.

2) Computational time: An important aspect in robotics is the time the processor is used. Low times lead to faster responses and less resources consumption.

From this perspective, the average computational time was measured as a function of the number of particles for each method. This time includes all the calculus performed between two consecutive poses of the robot including the landmarks estimation.

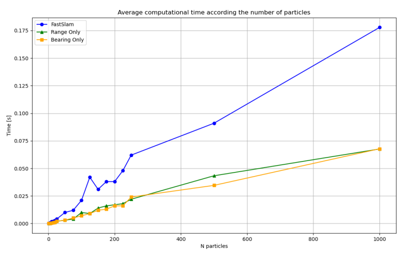


Fig. 7: Average computational time.

It is natural that the primary algorithm needs more time since it measures and processes both, bearing and range, while the other just one of them. However, from this plot and the image 6a, it may be affirmed that 150 particles is optimal. Moreover, the total computation time was evaluated and the tendency was exactly the same.

D. Frobenius Norm

The matrix norm provides a real value from a matrix offering an idea about the magnitude of its elements. For a 2x2 matrix, it is given by: $\sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2}$. where a_{ij} are the elements of the matrix. So, applying this to the covariance matrix of the landmarks presents an indication of the certainty of the estimations.

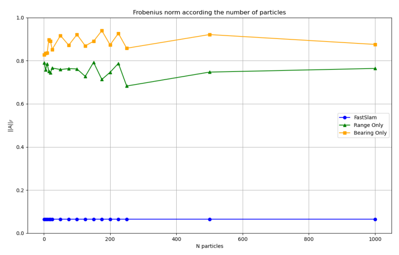


Fig. 8: Frobenius norm of the average of landmarks covariance matrices.

From this results, it can be said that the estimations of the landmarks position from the root algorithm are much more certain than those from the derived methods. Therefore, the result observed in picture 6b where Range Only showed better performance could not be truth since there is a lot of uncertainty in those estimations.

$$||A||_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n}|a_{ij}|^2} \tag{8}$$

V. Conclusions

Throughout the project, multiple versions of FastSLAM were developed and implemented in simulation. These were iteratively tested under ideal conditions and with added noise in both odometry and observations. The algorithms were subsequently tested using real data collected in the laboratory. They produced acceptable results, although some limitations were observed.

While the algorithms performed well in simulation, this highlighted discrepancies between the simulated environment and real-world conditions. One notable issue was camera vibration, which led to poor landmark identification. Additionally, the layout of the environment often caused the robot to view landmarks from a side angle, reducing the detection rate.

The ground truth of the robot's path in the laboratory was not entirely reliable. Two improvements are suggested: enhancing the landmark detection method and minimizing camera vibrations. Furthermore, implementing optical lab triangulation could provide more accurate ground truth data for comparison with the estimated trajectory.

Overall, the proposed method was successfully developed, tested, and implemented, though several areas for improvement have been identified.

References

[1] T. Yasuda, "cv_camera," [Online]. Available: https://wiki.ros.org/cv_camera. [Accessed: Jun. 12, 2025].

[2] J. Bowman and P. Mihelich, "camera_calibration," [Online]. Available: http://wiki.ros.org/camera_calibration. [Accessed: Jun. 12, 2025].

[3] J. Vaughan, "aruco_detect," [Online]. Available: http://wiki.ros.org/aruco_detect. [Accessed: Jun. 12, 2025].

[4] Montemerlo, Thrun, Koller, Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem" (2002)

[5] Timothy O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE):when to use them or no" (2022)