EKF-SLAM Indoor Simulation

Instituto Superior Técnico, Autonomous Systems, 2021/2022, 2nd Semester

Ana Catarina L. Pereira
Number 92654
Instituto Superior Técnico
Lisbon, Portugal

André M. Ribeiro
Number 92662
Instituto Superior Técnico
Lisbon, Portugal

Diogo C. Ferreira
Number 92674
Instituto Superior Técnico
Lisbon, Portugal

João Pedro L. Albuquerque
Number 92698
Instituto Superior Técnico
Lisbon, Portugal

***Abstract*—This paper investigates the convergence properties and consistency of an Extended Kalman Filter (EKF) based Simultaneous Localization and Mapping (SLAM) algorithm on a simulated robot and environment. Proofs of convergence are provided for the nonlinear two-dimensional SLAM problem with ArUco landmarks observed using a range-bearing sensor.**

***Index Terms*—EKF, SLAM, Indoor, ArUco**

I. Introduction

This paper describes the implementation of an Extended Kalman filter based Simultaneous Localization and Mapping algorithm applied to a simulation of a mobile robot in an indoor environment. The robot must be able to move autonomously and with sufficient accuracy in this situation to maintain a safety distance from the nearest hindrance at all times. As a result, the exact position of the vehicle and surroundings location must be known at all times.

Regarding the algorithm, Simultaneous Localization and Mapping, or SLAM, is the process of building the map of an environment while concurrently generating an estimate for the pose of the robot. The most common non-linear SLAM implementation is an Extended Kalman filter, or EKF, where non-linear models are linearized to fit the Kalman filter method and system noise is assumed to be Gaussian.

II. Methods and Algorithms

In this section, the EKF SLAM algorithm is presented. There are several classifications of SLAM, two categories of which are Full SLAM and Online SLAM. While Full SLAM derives the entire series of robot poses, Online SLAM just derives the most recent robot position. Given that only the current position and map are required, the scenario described in this paper corresponds to an Online SLAM problem.

A. SLAM

There are several steps in the SLAM method. It aims to use the environment to update the position of the robot, which cannot rely solely on the odometry of the robot due to its erratic behaviour. In order to adjust the position of the robot, features from the environment are extracted, and then the robot is moved around as new observations are made. The core of the SLAM procedure is an EKF (Extended Kalman Filter), which is responsible for updating the robot's perception of its location, based on these observations, also known as landmarks. The EKF keeps track of an estimate of the level of uncertainty associated with both the position of the robot and the location of the landmarks it has observed in the area surrounding.

B. Extended Kalman Filter

Odometry data and landmark observations are utilized to determine the state (position) of the robot and the map using the Extended Kalman Filter over two main steps, the prediction step and the correction step. In the prediction step, based on the previous robot pose and the robot odometry, the future robot state is predicted and in the correction step, the Kalman Gain is used to update the state (mean and covariance matrix) by computing the difference between the measured value and the predicted value.

1) State Representation: The state vector represents an estimation, or a belief, of the robot's pose and the location of n landmarks, or map, and is denoted as

$$x_t = (x, y, \theta, m_{1,x}, m_{1,y}, \ldots, m_{n,x}, m_{n,y})^T \tag{1}$$

where $(x, y, \theta)^T$ is the robot position and orientation and $(m_{1,x}, m_{1,y})^T, \ldots, (m_{n,x}, m_{n,y})^T$ are the positions of the landmarks. Assuming a Gaussian distribution for both the robot pose and the landmark positions, the brief probabilistic representation of the state is the mean and variance:

$$\mu = \begin{bmatrix} x \\ m \end{bmatrix} \quad \Sigma = \begin{bmatrix} \Sigma_{xx} & \Sigma_{xm} \\ \Sigma_{mx} & \Sigma_{mm} \end{bmatrix} \tag{2}$$

where Σ_{xx} is the co-variance matrix of the robot pose, Σ_{mm} is the co-variance matrix of each landmark and Σ_{xm} and Σ_{mx} represent the cross-variance between the robot pose and the landmarks.

2) Prediction Step: In this step, given the prior robot stance and robot odometry, the expected robot state can be predicted. The robot is only supposed to adjust its own position, and not the placement of any landmarks, affecting only x, Σ_{xx}, Σ_{xm}, and Σ_{mx}.

a) Robot Motion Model: An internal kinematic model of a mobile robot provides its odometry. Since odometry also accounts for friction and the internal dynamics of the motor, using its odometry values is preferable to building a kinematic model from scratch.

The robot odometry readings at each time step are given by $u_t = (\delta_{rot_1}, \delta_{trans}, \delta_{rot_2})$, where δ_{rot_1} is the discrepancy of the robot in orientation in the instant before the displacement, δ_{trans} the total distance covered throughout that time step, and δ_{rot_2}, the discrepancy in orientation in the instant after the displacement. Thus, the motion of the robot update considered is given by:

$$\begin{bmatrix} x_t \\ y_t \\ \theta_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \delta_{\text{trans}} \cos(\theta_{t-1} + \delta_{rot_1}) \\ \delta_{\text{trans}} \sin(\theta_{t-1} + \delta_{rot_1}) \\ \delta_{rot_1} + \delta_{rot_2} \end{bmatrix} \quad (3)$$

The robot motion model, as well as the Jacobian and motion noise, are used, respectively, to generate the estimated mean and covariance, given by:

$$\bar{\mu}_t = g(u_t, \mu_{t-1}) \quad \bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t \quad (4)$$

3) Correction Step: The observations used for correction are landmarks identifications by the camera, returning the range and bearing of the landmarks. These are then used by the Extended Kalman Filter to adjust the results.

a) Range-bearing Observation Model: Observing landmarks is essential for SLAM because the entire structure of the SLAM problem critically depends on maintaining complete knowledge of the cross correlation between landmark estimates. In fact, in the limit, the errors in the estimates of any pair of landmarks becomes fully correlated. [1]

The observed location of landmark i with the estimated location of the robot and the relative measurement is given by the model h:

$$\begin{bmatrix} \bar{\mu}_{i,x} \\ \bar{\mu}_{i,y} \end{bmatrix} = \begin{bmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{bmatrix} + \begin{bmatrix} r_t \cos(\phi_t + \bar{\mu}_{t,\theta}) \\ r_t \sin(\phi_t + \bar{\mu}_{t,\theta}) \end{bmatrix} \quad (5)$$

Some observations are not very reliable, particularly the bearing, due to certain movements done by the robot, thus an effort is taken to keep the robot as stable as possible.

Following data association, the Kalman gain is computed as follows:

$$K_t = \bar{\Sigma}_t H_t^T \left(H_t \bar{\Sigma}_t H_t^T + Q \right)^{-1} \quad (6)$$

where H is the Jacobian of the observation model and Q the measurement noise. The state mean and covariance are, then, given by

$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t)) \quad \Sigma_t = (I - K_t H_t) \bar{\Sigma}_t \quad (7)$$

Little is known about the optimal density of landmarks. One should choose landmarks sufficiently far away from each other so that confusing one with another is negligible. However, too few landmarks makes it more difficult to localize the robot, resulting in an increase of the chances of confusing landmarks. [2] The estimated errors in landmark location should decrease monotonically, reducing the overall map error with observations. The landmarks estimation error lower bound corresponds to the initial uncertainty in vehicle location. [1]

The correction step may differ slightly depending on the chosen implementation.

C. Landmarks with IDs

For each seen ID, the correction step converts the sensor data range and bearing into the estimated xy position of the landmark, given the estimated position of the robot. The previous estimated position of the landmark is also required. Then the Kalman Filter equations are used to update the state and the covariance matrices.

D. Landmarks without IDs

The underlying theory is the same, but implementation is slightly different given that landmark IDs are not known. Thus, the algorithm must also identify the correct landmark it is seeing. This feature is implemented with a Maximum Likelihood algorithm. For each observation, the Mahalanobis distance is computed for all the landmarks already seen and the smallest one is chosen. However, if the Mahalanobis distance is larger than a certain threshold for all known landmarks, then the seen landmark is a new one. The threshold choice is manual and done empirically. One possible improvement to this algorithm is presented in [1] and [2] and consists in maintaining a provisional list of landmarks, where a new landmark is firstly added to a provisional list and isn't used to update the pose of the robot. Once a landmark has consistently been observed it is transitioned into the regular map, or, if not, is deleted. [1]

III. Implementation

A. Data collection

To implement the EKF-SLAM algorithm using the experimental data from the Gazebo Simulator, the rosbag function of ROS was used. This function not only allows for the collection of all relevant data from the Gazebo Experiments, but also allows for the replicability of the experiments without the need to run them all over again. In order to read the rosbag, the ROS toolbox from MATLAB® was used and the ground truth and odometry data were extrated and saved to the workspace, in order to use it as input for the EKF-SLAM algorithm. The observation data and landmark detection data was processed in a python script, explained further in the following sections.

B. Landmark Extraction

The ArUco library, an OpenSource toolkit for camera position estimation, provides the visual landmarks that were used throughout this project for landmark pose estimation. The range-bearing measurement method used in this project is vision, more specifically a simulated ASUS Xtion Pro with a 60º horizontal field of view.

Firstly, given an image containing ArUco markers, the detection process returns a list of detected markers, containing the position of its four corners in the image and its identification, or ID.

Secondly, the camera pose with respect to a marker is estimated through a transformation from the marker coordinate system to the camera coordinate system, using the calibration parameters of the camera obtained from the corresponding

ROS topic. These are the camera matrix and distortion coefficients.

Consequently, the rotation and translation vectors for each of the markers are obtained. Therefore, their range, r, and bearing, ϕ, can be extrapolated by

$$t_{vec} = \begin{bmatrix} x & y & z \end{bmatrix} \quad r = \sqrt{x^2 + z^2} \quad \phi = \arctan\left(\frac{x}{z}\right) \quad (8)$$

where t_{vec} is the translation vector. The height, y, of the marker is disregarded while performing the computations because the motion of the robot is only in two dimensions, leaving the angle and distance as merely an horizontal change.

C. Outlier Removal

Despite being a rare occurrence, some measurements done with the ArUco library do not correctly identify the marker. Since this can hinder the convergence of the algorithms, an outlier removal algorithm is employed in order to prevent such misidentifications from corrupting the data as this outliers that might appear in some observations are meant to be disregarded.

The algorithm simply determines if a measurement is credible or not by comparing the ID provided by the observation at a particular instant with the ID provided by the observation an instant prior and posterior. It also confirms whether the landmark is among those linked to the simulation. If it determines that the measurement is an outlier, it discards it and does not update the map or co-variances matrix, otherwise, using the Kalman gain, this algorithm may continue to update the map and co-variances matrix.

D. Micro-simulator

1) Purpose and how it works: In order to validate the EKF algorithms a micro-simulator was developed in which it is possible to obtain positional and observational data in a simple, controlled and replicable environment. This has allowed for extensive tests of the algorithms in order to tune them for the experimental data.

The micro simulator receives as inputs an ordered list of waypoints (position x and y on the map) and a list of landmarks (ID and position x and y on the map). It then returns the real trajectory, an artificial odometry trajectory and the observations for each time frame. Both the odometry and the observations have added noise (white Gaussian noise) in order to better test the algorithms.

2) Results obtained: Using the micro simulator several trajectories were tested and validated. Fig. 1 shows one interesting example of such trajectory and Fig. 2 the error results and number of landmarks seen for that trajectory.

It is clear there was an improvement in position tracking using the EKF algorithm, compared to the odometry. The data of this simulation can be split into 3 phases:

1) Phase 1: Initial algorithm prediction
2) Phase 2: Dark region (reduced or no landmarks sights)
3) Phase 3: Loop closure (observes the first landmark a second time)

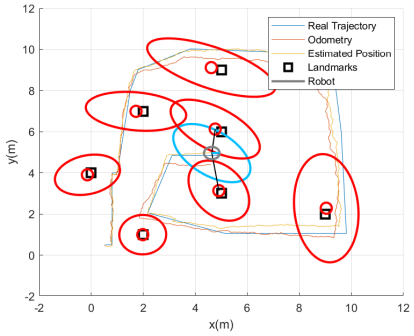

Fig. 1. Micro-simulator trajectory

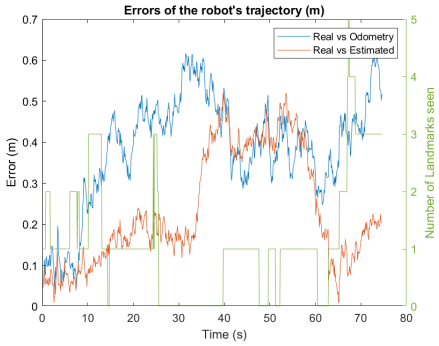

Fig. 2. Estimation error $[m]$ vs Odometry error $[m]$ and Landmarks seen

The average error was computed for each phase and is displayed in Table I.

TABLE I
EVOLUTION OF THE ERROR DURING THE SIMULATION

Phase	Time (s)	Odometry error (m)	Prediction error (m)	Average number of landmarks seen
1	1 - 30	0.3145	0.1336	1.1833
2	30 - 62	0.4288	0.3648	0.5514
3	62 - 75	0.4436	0.1437	2.444

The data shows that the prediction error rises and falls depending on which phase it is in. At the start of the simulation it starts to slowly improve its position estimation relative to the odometry as it sees new landmarks with an average error of $0.1336m$ for an average of 1.1833 landmarks seen. However when in phase 2 the average number of landmarks seen decreases to 0.5514, the error begins to increase reaching $0.3648m$, approximating the odometry error at $0.4288m$. In phase 3, when the loop closure occurs, the error decreases rapidly in the robot position, achieving an average of $0.1437m$ deviation from the real trajectory in the last seconds of the simulation.

3) Conclusions: From the previous results it can be deduced that in the experiment it is optimal to avoid phase 2 (region with few observations of the landmarks) and try to achieve loop closure like in phase 3 to reduce the estimation error. With this in mind, 2 more landmarks were implemented

in order to obtain at least one landmark observation in each frame. The results achieved for the error were lower than the previous results throughout the whole simulation which indicates a good performance for the EKF-SLAM algorithm (see Fig. 3 and Fig. 4). In Fig. 3 the variance of the robot (blue ellipses) is also represented in several instances along the trajectory. It is noted that the variance increases slowly throughout the trajectory, reducing significantly when loop closure happens (robot sees the 1st landmark for a 2nd time). Loop closure is represented in Fig. 5, where four different time-steps are overlaid, allowing to check the evolution of the covariance of the landmarks. Since the micro-simulator allows for the robot to check several landmarks at a time (broad field of view) and to do so in almost every iteration, each covariance of the landmarks is deeply related to others. On loop-closure, viewing the first landmark, with the least covariance, reduces the covariance of the others.

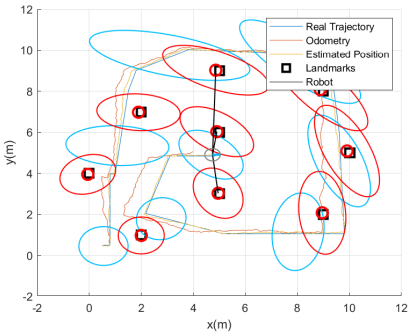

Fig. 3. Micro-simulator trajectory with added landmarks

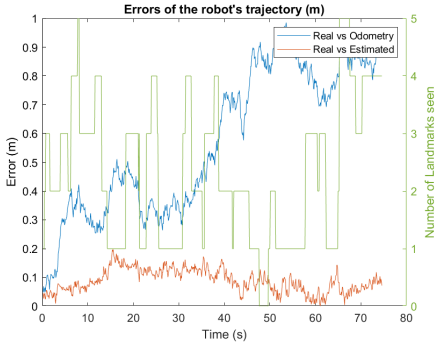

Fig. 4. Micro-simulator error with added landmarks

IV. Experimental Results

The algorithms were ran with experimental data, although still obtained from simulation. However, even after outliers removal to attempt data integrity, the results obtained for all the simulations performed indicated that the camera measurements were highly inaccurate. In order to test the different algorithms, several simulations with experimental data were ran and two of them are presented in this section to enlighten the problems faced. The noise matrices were empirically

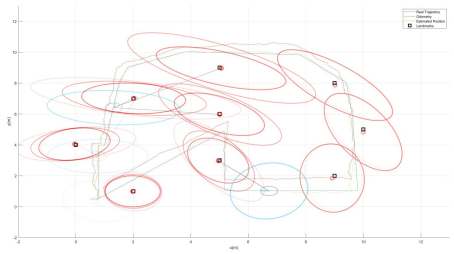

Fig. 5. Evolution of a micro-simulation trajectory with overlaid time-steps

calculated for each simulation, given its dependence on the robot swinging and bad camera data. Odometry data is very reliable while sensor data is not. Thus, the EKF won't have a very strong correction step.

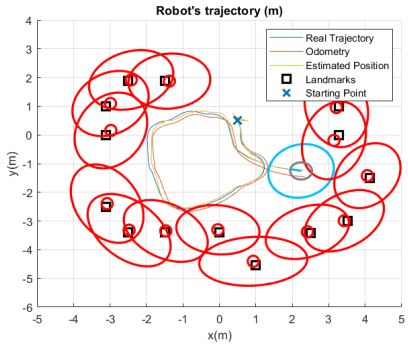

Fig. 6. Real data trajectory with 14 landmarks - Simulation 1

A. Landmarks with IDs

As mentioned before, each ArUco marker has its own ID. Therefore, the programme can easily and correctly identify each landmark. In Fig. 7 (Simulation 1), it is seen that the estimated position error is slightly superior than the odometry error, but after the loop closure, the estimated value tends to the real one, as seen in Fig. 6 as the estimated error decreases. In Fig. 8 the position error of the landmarks is shown over time and it is seen that the error overall decreases, however not always monotonically. Increasing the covariance matrix resulted in worse errors and didn't solve the monotony problem. Thus one can conclude that there must be sensor incoherent data for some iterations resulting in an unexpected error increase.

As for Simulation 2, in Fig. 9, it is noticeable that the estimation error is lower than the odometry one, which is in accordance to the fact that it relies on odometry and only makes smaller corrections using the positions of the landmarks due to its somewhat inconsistent sensor data. It is clear that it is correcting the odometry position, even though the error does not converge to zero, in part due to the higher position of the landmarks error, which also varies more (Fig, 10), an indication higher sensor imprecision.

Loop closure is also harder with experimental data because the sampling frequency of the camera is lower than the

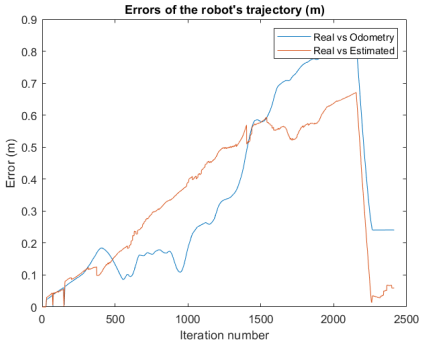

Fig. 7. Real data trajectory error - Simulation 1

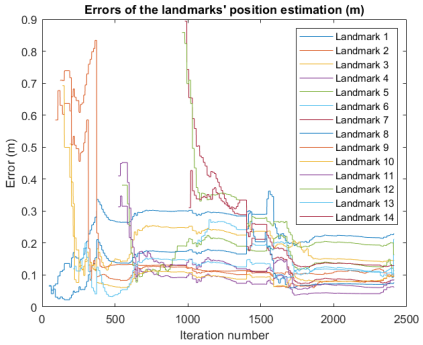

Fig. 8. Real data position of the landmarks error - Simulation 1

odometry's and the field of view may sometimes be obstructed, resulting in very few observations compared to the micro-simulator and less inter-landmarks correlations.

Fig. 9. Real data trajectory error - Simulation 2

B. *Landmarks without IDs*

Not having an ID for each landmark raises the data association problem, possibly resulting in wrong landmark associations. The threshold must be very carefully chosen to prevent the creation of new landmarks when an old landmark is being seen, thus depending on the variance of the observations. For an equivalent number of landmarks to be considered, increasing the covariance matrix implies a decrease in the threshold. The number of landmarks also has to be carefully chosen in order to prevent miss-sightings, being a compromise between sufficient landmark observations and correct identification.

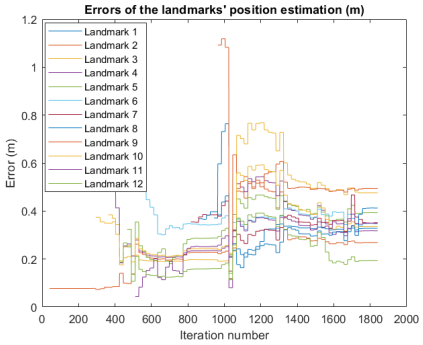

Fig. 10. Real data position of the landmarks error - Simulation 2

As expected, the experimental results are worse than the micro-simulation ones due to random sensor corrupted data and that the maximum-likelihood algorithm performs worse, since some landmarks are often mistaken by others.

In Fig. 11, the estimation error is smaller than the odometry error after loop closure and the image is quite similar to the one obtained in the same simulation with known IDs (Fig. 7). This indicated that on this case, little to no landmarks were miss-identified.

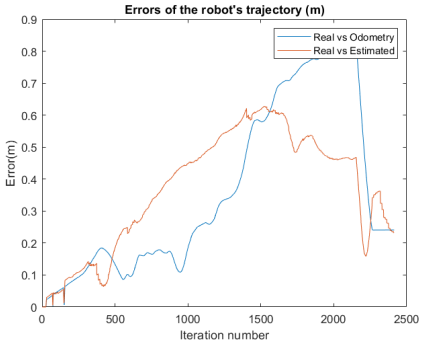

Fig. 11. Real data trajectory error - unknown landmarks - Simulation 1

As for Simulation 2, the results obtained with this algorithm are much worse than when using the ID of the landmarks, as can be seen in Fig. 12. Since Fig. 13 showed that the position of the landmarks estimation was much worse, the conclusion that there was an error in data association was drawn. To verify this, a study on the impact of the number and position of landmarks was conducted (Fig. 14) which indicated that the estimation with 10 landmarks is actually better than the estimation with 11 because of wrong data association. Although it does not seem intuitive, this once again goes to prove that the errors inherited from the sensors and the pose of the robot are often high enough so that even landmarks quite distanced are sometimes mistaken. As for the results with 5, 7, or 9 landmarks, as expected, the amount of observations is not enough to provide a good correction of the odometry data.

Note that even if the sensors provide wrongful measurements in the case of known IDs, the measurements will be associated to the correct ID, thus yielding better results. On the other hand, when the ID are unknown, a miss-identification on the landmark results on a wrong update on the covariance matrix as well as a inadequate update on the position of the robot and map. To try to overcome this issue, a provisional landmark list was attempt, but no better results were obtained.

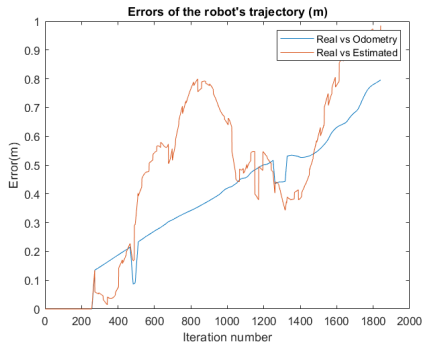

Fig. 12. Real data trajectory error - unknown landmarks - Simulation 2

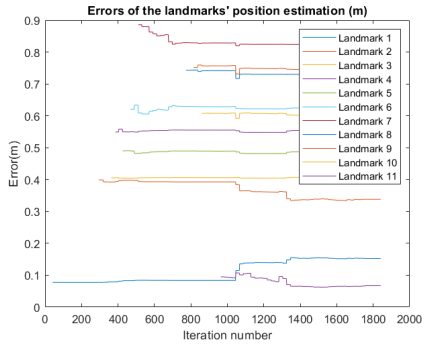

Fig. 13. Real data position of the landmarks error - unknown landmarks - Simulation 2

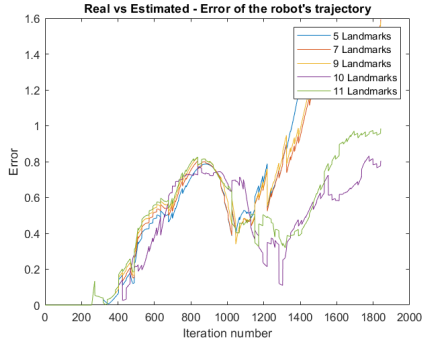

Fig. 14. Real data trajectory error for different numbers of unknown landmarks - Simulation 2

V. Conclusions

In conclusion, the EKF-SLAM algorithm used for this paper was validated to its full extent in a micro-simulator, both using landmarks with IDs and landmarks without IDs enabled by a Maximum Likelihood method (EKF-SLAM with ML) also explained thoroughly in this paper.

The micro-simulator additionally enabled testing of potential limiting situations in both algorithms. It was a valuable tool to test the algorithms' dependence on landmarks and loop closure, confirming its fitness and the presence of those features.

The localisation was found to be according to what is expected since the estimation error converges to zero. Regarding mapping, it was apparent in the micro-simulation that mapping of both algorithms was valid since both accurately detected all of the landmarks, which is directly correlated to good mapping precision, as well as providing a good estimation for the positions.

As for the experimental data from Gazebo, there were some noticeable adjustments that needed to be made, mostly due to irregularities in the data acquisition process such as errors resulting from the robot's physics (shaking, tilting, etc.), differences in the sampling frequency of the camera, and odometry and observation outliers. Due to all of these errors, in order to be able to run the already validated algorithms in this scenario, a lot of data treatment was needed. In this way, a filter function for outliers was developed, the data from the observations and the odometry was matched and the Q and R values for the EKF algorithm were properly calibrated.

Furthermore, due to the anomalies in the data, the threshold for the EKF-SLAM with ML also required a lot of adjusting. All of this data treatment was fruitful, and good results were achieved for the EKF-SLAM with IDs algorithm, that were considerably worse than the micro-simulation results, which is to be expected given the relatively weak data samples obtained from the Gazebo experiments. Considering this major handicap it can be said that, for a better set of observational data, using a better camera with a much higher sampling frequency, the results would be closer to those obtained in the micro-simulator. Despite this, the algorithm still presents an error correction from the odometry during most of the trajectory.

Regarding the EKF-SLAM with ML, the results were worse than with ID, as expected. Landmark association was more challenging which in turn worsened the results. In order to solve these issues in the maximum-likelihood algorithm, a provisional list of landmarks was attempted but did not return worth-mentioning results. If this method can be improved in the future, it might produce better results; if not, it might be worth looking into other landmark association techniques that do not rely on maximum likelihood.

References

[1] M. W. M. Gamini Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem" in IEEE Transactions on Robotics and Automation, vol. 17, no. 3, pp. 229-241, June 2001, doi: 10.1109/70.938381.

[2] S. Thrun, W. Burgard, D. Fox, "Probabilistic Robotics" in IEEE Transactions on Robotics and Automation, The MIT Press, 2005, ISBN 9780262201629