# Pose-Based EKF SLAM Using ICP Scan Registration and Matching

Eliyas Abraha, Mohamed Khaled Othman, Goitom Leaku

**Abstract**—This paper presents a Pose-Based Extended Kalman Filter (EKF) SLAM approach utilizing the Iterative Closest Point (ICP) algorithm for scan registration and matching, implemented on a Kobuki Turtlebot platform. The proposed system integrates data from a 2D LIDAR, IMU, and wheel encoders to perform real-time mapping and localization. Experiments conducted in both simulated environments (RViz and Stonefish) and real-world settings demonstrate the algorithm's high accuracy and effectiveness. Additionally, a pose optimization technique is introduced to manage the state vector size while preserving crucial pose history. The results confirm the proposed method's capability for accurate and reliable autonomous navigation.

**Index Terms**—Pose-Based EKF SLAM, Iterative Closest Point (ICP), Scan matching, real-time mapping, autonomous navigation.

## 1 Introduction

This research paper focuses on the implementation and evaluation of an Extended Kalman Filter (EKF) SLAM algorithm using the Iterative Closest Point (ICP) method for scan matching. The primary objective is to investigate the effectiveness of the proposed EKF SLAM algorithm in both simulated and real-world environments and to address the main challenges associated with its application. By leveraging the ICP method, the algorithm aims to refine pose estimates through accurate scan matching, thereby enhancing the overall SLAM performance. The algorithm is implemented on a Kobuki Turtlebot, a widely used mobile robot platform known for its robustness and versatility. The Turtlebot is equipped with 2D LIDAR, IMU, and wheel encoders, which provide the necessary sensor data for SLAM. The integration of these sensors enables the robot to perform real-time mapping and localization with improved accuracy.

Simulation experiments are conducted in controlled environments using tools such as RViz and Stonefish. These simulations allow for a systematic evaluation of the algorithm's behavior and performance under various conditions. Real-world experiments are also conducted to gain practical insights into the algorithm's applicability and robustness in dynamic settings. These experiments highlight the practical challenges and considerations when deploying the SLAM algorithm in real environments, such as sensor noise, environmental changes, and computational constraints.

A significant aspect of this research is the optimization of the state vector within the EKF framework. This optimization is crucial for maintaining a balance between computational efficiency and the preservation of sufficient pose history information for accurate loop closure. By effectively managing the size of the state vector, the proposed SLAM algorithm ensures efficient processing without compromising on mapping and localization accuracy.

The structure of this paper is as follows: Section 2 provides a brief overview of related work in SLAM, emphasizing the contributions and significance of this research. Section 3 outlines the methodology, detailing the system setup, data acquisition process, and implementation specifics of the EKF SLAM algorithm. Section 4 describes the result and discussion , covering both simulated and real-world experiments.Finally, Section 5 concludes the paper by summarizing the key findings and discussing potential avenues for future research, including possible improvements and applications of the algorithm in more complex scenarios.

## 2 Related Works

Simultaneous Localization and Mapping (SLAM) techniques have undergone extensive exploration, with Extended Kalman Filter (EKF) SLAM emerging as a prominent method for its ability to provide probabilistic estimates of a robot's pose and map of the environment [1]. However, the computational complexity associated with EKF SLAM, particularly as the size of the state vector increases, has spurred research into mitigating this challenge.
One such approach involves leveraging the Iterative Closest Point (ICP) algorithm, renowned for its effectiveness in aligning point clouds and refining pose estimates [2], [3]. This algorithm has been integrated into SLAM systems to enhance pose estimation accuracy by minimizing the error between consecutive scans. Foundational works by [4] and [5] laid the groundwork for ICP's application in SLAM, with subsequent research refining its implementation for various robotic platforms.
This paper contributes to the ongoing efforts in SLAM research by implementing an EKF SLAM algorithm using ICP for scan matching on a Kobuki Turtlebot platform. By conducting experiments in both simulated and real-world environments, this research aims to provide a comprehensive evaluation of the proposed algorithm's effectiveness and address practical deployment challenges faced in real-world robotic applications.

## 3 METHODOLOGY

This section outlines the methodology employed in implementing and evaluating the Extended Kalman Filter (EKF) SLAM algorithm using the Iterative Closest Point (ICP) method for scan matching on the Kobuki Turtlebot platform. The methodology is structured into subsections focusing on the mathematical model, motion model, state vector representation, observation model encompassing sensor fusion, and the scan matching technique utilized for refining pose estimates.

### 3.1 SLAM Mathematical Model

In the context of Pose-Based EKF SLAM, the map is maintained as a scan history denoted by equation 1, which is history of an individual scans:

$$S_H = \begin{bmatrix} {}^{B_1}S_{B_1} & {}^{B_2}S_{B_2} & \cdots & {}^{B_k}S_{B_k} \end{bmatrix} \tag{1}$$

Each scan, ${}^{B_k}S_{B_k}$, is composed of multiple points:

$${}^{B_k}S_{B_k} = \begin{bmatrix} {}^{B_k}S_1 & {}^{B_k}S_2 & \cdots & {}^{B_k}S_N \end{bmatrix}, \quad \forall\, {}^{B_k}S_i \in R^2 \tag{2}$$

These points, ${}^{B_k}S_i$, are linked to the robot's pose at the time of the scan ${}^{B_k}S_{B_k}$, specifying their origin in the environment.

#### 3.1.1 SLAM State Vector

The state vector in our Pose-Based EKF SLAM framework encapsulates the robot's pose and the associated scan poses. This vector, denoted as $Nx_k$, is modeled as a Gaussian Random Vector encompassing the robot's state and the positions from which each scan is acquired. The formulation of the state vector is:

$${}^{N}x_k = \begin{bmatrix} {}^{N}x_{B_1} \\ {}^{N}x_{B_2} \\ \vdots \\ {}^{N}x_{B_k} \end{bmatrix}; {}^{N}x_k = \mathcal{N}\left({}^{N}\hat{x}_k, {}^{N}P_k\right) \tag{3}$$

where the mean and covariance are given by:

$${}^{N}\hat{x}_k = \begin{bmatrix} {}^{N}\hat{x}_{B_1} \\ {}^{N}\hat{x}_{B_2} \\ \vdots \\ {}^{N}\hat{x}_{B_k} \end{bmatrix}; {}^{N}P_k = \begin{bmatrix} {}^{N}P_{B_1} & {}^{N}P_{B_1B_2} & \cdots & {}^{N}P_{B_1B_k} \\ {}^{N}P_{B_2B_1} & {}^{N}P_{B_2} & \cdots & {}^{N}P_{B_2B_k} \\ \vdots & \vdots & \ddots & \vdots \\ {}^{N}P_{B_kB_1} & {}^{N}P_{B_1B_2} & \cdots & {}^{N}P_{B_k} \end{bmatrix} \tag{4}$$

To accurately represent the map, each scan point must be transformed to the global frame using the corresponding pose. This transformation is expressed as:

$$Ns_i = Nx_{B_i} \oplus Bs_i \tag{5}$$

Here, $Ns_i$ denotes the global frame coordinates of the scan point, $Nx_{B_i}$ is the pose from which the scan was taken, and $Bs_i$ represents the scan points in the local frame of the robot. Scans are taken in LiDAR frame, The LiDAR frame is usually mounted on the robot, and the first transformation is from the LiDAR's coordinate system to the robot's coordinate system (footprint frame).

### 3.2 Robot Motion Model

The motion model for the robot can be represented as a differential drive system, in which the encoder measurements provide information about the robot's angular speed in each wheel. This data is used to estimate the robot's linear and angular velocities, which are then used for estimating the robot's pose and orientation.

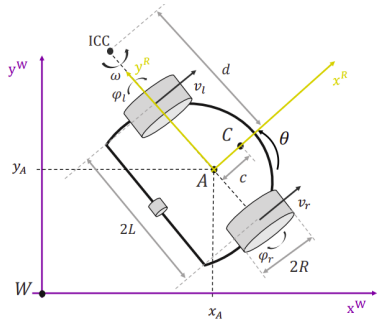



Fig. 1: Differential drive robot

The positioning and yaw orientation of the robot within the world $N-frame$ are encapsulated in its pose. Conventionally, the pose is indicated as follows:

$${}^{N}x_{B_k} = \begin{bmatrix} {}^{N}x_k \\ {}^{N}y_k \\ {}^{N}\theta_k \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} {}^{N}\hat{x}_k \\ {}^{N}\hat{y}_k \\ {}^{N}\hat{\theta}_k \end{bmatrix}, {}^{N}P_k\right) \tag{6}$$

where ${}^{N}x_k$ and ${}^{N}y_k$ represent the robot's position, and ${}^{N}\theta_k$ denotes the orientation. As the robot moves, the encoder provides inputs to predict the position of the robot, as given in Equation 7 and 8.

$${}^{N}x_{B_k} = f\left({}^{N}x_{B_{k-1}}, u_k, w_k\right) \tag{7}$$

$$\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ \theta_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{R\Delta t\cos\theta_{k-1}}{2} & \frac{R\Delta t\cos\theta_{k-1}}{2} \\ \frac{R\Delta t\sin\theta_{k-1}}{2} & \frac{R\Delta t\sin\theta_{k-1}}{2} \\ \frac{r_w\Delta t}{L} & -\frac{r_w\Delta t}{L} \end{bmatrix} \begin{bmatrix} \varphi_l \cdot \\ \varphi_r \cdot \end{bmatrix} \tag{8}$$

Here, ${}^{N}x_{B_{k-1}}$ is the previous pose of the robot, $u_k$ is the control input derived from the left and right wheel velocities, and $w_k$ represents the uncertainties in the wheel velocities. The noise from the left and right wheels, $w_k$ is represented as in eq:9.

$$w_k \sim \mathcal{N}(0, Q_k); \quad Q_k = \begin{bmatrix} \sigma_l^2 & 0 \\ 0 & \sigma_r^2 \end{bmatrix} \tag{9}$$

As the robot moves, the state vector is updated by adding the new robot view pose and also the covariance matrix is updated with a new pose as in equation 4 and in predicted state vector as in eq: 10

$${}^{N}\hat{\hat{x}}_k = f\left({}^{N}\hat{x}_{k-1}, u_k, \hat{w}_k\right) = \begin{bmatrix} {}^{N}\hat{x}_{B_0} \\ \vdots \\ {}^{N}\hat{x}_{B_{k-2}} \\ f\left({}^{N}\hat{x}_{B_{k-1}}, u_k, \hat{w}_k\right) \end{bmatrix} \tag{10}$$

So, the prediction step of the SLAM is detailed in eq: 11,12

$${}^{N}\hat{\hat{x}}_k = f\left({}^{N}\hat{x}_{B_{k-1}}, u_k, 0\right) \tag{11}$$

$$ {}^{N}\bar{P}_k = F_{1_k}\,{}^{N}P_{k-1}F_{1_k}^T + F_{2_k}Q_kF_{2_k}^T \tag{12} $$

where $F_{1_k}$ is the Jacobian of the motion model with respect to the state vector as in 13, and $F_{2_k}$ is the Jacobian of the motion model with respect to the noise as in 14.

$$ F_{1_k} = \frac{\partial f\left({}^{N}\boldsymbol{x}_{k-1}, \boldsymbol{u}_k, \boldsymbol{w}_k\right)}{\partial {}^{N}\boldsymbol{x}_{k-1}} = \begin{bmatrix} \boldsymbol{I} & \dots & \boldsymbol{0} & \boldsymbol{0} \\ \vdots & \ddots & \vdots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \dots & \boldsymbol{0} & \boldsymbol{J}_{\boldsymbol{f}_{\boldsymbol{x}}} \end{bmatrix} \tag{13} $$

$$ \boldsymbol{F}_{2_k} = \frac{\partial f\left({}^{N}\boldsymbol{x}_{\boldsymbol{k}-1}, \boldsymbol{u}_k, \boldsymbol{w}_k\right)}{\partial \boldsymbol{w}_k} = \begin{bmatrix} \frac{\partial {}^{N}\boldsymbol{x}_{B_0}}{\partial w_k} \\ \vdots \\ \frac{\partial {}^{N}\boldsymbol{x}_{B_{k-2}}}{\partial w_{k-}} \\ \frac{\partial f\left({}^{N}\boldsymbol{x}_{B_{k-1}}, \boldsymbol{u}_k, w_k\right)}{\partial w_k} \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \vdots \\ \boldsymbol{0} \\ \boldsymbol{J}_{f_w} \end{bmatrix} \tag{14} $$

### 3.3 Obseravtion Model

Given our focus on pose-based SLAM, with no incorporation of feature observations, we examine two primary observation types: displacement observations derived from lidar scan poses using the ICP registration algorithm, and IMU sensor observations. Displacement measurements, obtained through ICP registration, crucially inform the accurate estimation of the robot's pose and map construction. Meanwhile, IMU sensors act as absolute sensors, aiding in updating the robot's heading.

#### 3.3.1 IMU Sensor Model

The robot's IMU determines the robot absolute heading in relation to the north. The IMU is used to update and correct the heading of the turtlebot, using the Kalman filter as in Equation 15 and 16.

$$ z_k = H_k^N x_k + V_k \tag{15} $$

$$ z_k = \begin{bmatrix} 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} {}^{N}\hat{x}_{B_1} \\ {}^{N}\hat{y}_{B_1} \\ {}^{N}\hat{\theta}_{B_1} \\ \vdots \\ {}^{N}\hat{\theta}_{B_k} \end{bmatrix} + V_k \tag{16} $$

where H_k = [0,0,0.....,1] is the observation matrix, Vk=1, for the Jacobian with respect to observation noise.

#### 3.3.2 Scan Displacement Observation

The scan displacement observation model is designed to estimate the displacement between two robot poses from which scans were taken as shown in fig:2 .To estimate the displacement between the current viewpose ${}^{N}x_{B_1}$ and a previous viewpose ${}^{N}x_{B_0}$, the observation model is given by equ 17.

$$ z_k = h\left({}^{N}x_{B_0}, {}^{N}x_{B_1}, v_1\right) = \left(\left(\ominus {}^{N}x_{B_1}\right) \oplus^{N} x_{B_0}\right) + v_1 \tag{17} $$

Note that, here $\ominus$ is the pose inversion, and $\oplus$ is the pose compounding operation.

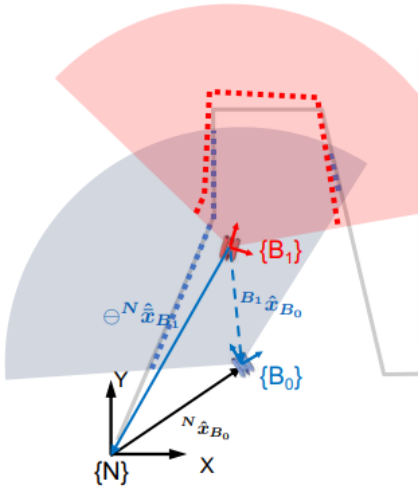



Fig. 2: Two Poses ICP Scan Matching

The observation matrix is obtained as the Jacobian with respect to the state vector as in eq.18 and the jacobian with respect to observation noise is given as in eq.19.

$$ H_k = \left.\frac{\partial \boldsymbol{h}\left({}^{N}\overline{\boldsymbol{x}}_k, v_k\right)}{\partial {}^{N}\overline{\boldsymbol{x}}_k}\right|_{N_{\overline{\boldsymbol{x}}_k = N}\hat{\hat{\boldsymbol{x}}}_k, w_k = 0} \tag{18} $$

$$ V_k = \left.\frac{\partial \boldsymbol{h}\left({}^{N}\overline{\boldsymbol{x}}_k, v_k\right)}{\partial \boldsymbol{v}_k}\right|_{N_{\overline{\boldsymbol{x}}_k = N}\hat{\hat{\boldsymbol{x}}}_k, w_k = 0} \tag{19} $$

### 3.4 SLAM Workflow

This section provides a comprehensive overview of the SLAM implementation, detailing each step of the mathematical models for both the motion and observation components. It begins with a description of the data processing techniques and the algorithms employed, followed by an outline of the specific methods used to address the SLAM problem. The section concludes with the algorithm workflow, visually represented in Figure 3, illustrating the development process.

#### 3.4.1 State Prediction

The EKF prediction step utilizes angular velocity data from the left and right wheel encoders during robot motion. This data is integral to the dead-reckoning motion model, as defined by Equation (10), for estimating the robot's new pose. In this prediction step, adjustments are made not only to the robot's pose but also to the covariance matrix. Specifically, changes occur in the last element of the diagonal and the last row and column of the covariance matrix. These modifications affect the robot's covariance and cross-covariance with the scan viewpoints maintained within the state vector.

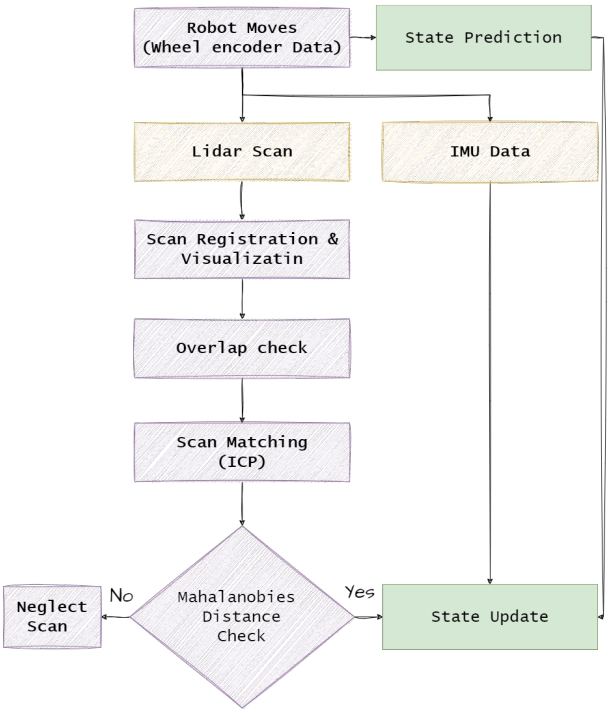



Fig. 3: SLAM Workflow

### 3.4.2 Scan Point Cloud Registration

In this project, Pose-based Graph SLAM, we used the Iterative Closest Point (ICP) algorithm for scan point cloud registration. This process involves aligning two point clouds, source and target, using an initial transformation derived from odometry data. Due to the high frequency of environment scans from the RPLIDAR sensor, not every scan is used; only scans recorded after the robot has traveled a certain distance are registered, in which the library called **Open3D** is used. Each scan is converted into point clouds and added to the scan poses, with the corresponding robot pose cloned. Overlapping scans are checked between the current scan and the scan history to form a hypothesis. To build the map, scans are converted to the world frame using transformations from the LIDAR frame to the robot frame, then to the world frame. The ICP algorithm's primary goal is to align the current scan (source point cloud) with the overlapping reference scan (target point cloud) by finding the transformation matrix that maps the source to the target.

### 3.4.3 Data Association

The data association process is crucial for identifying overlapping scans between the most recent scan and those stored in the scan history. A method was implemented to detect overlaps by applying a distance threshold between the current pose and other poses in the state vector. This threshold is carefully tuned to ensure coverage of the previous pose and several older poses in the case of the loop closure scenario to reduce the computational load of this task, the KDTree structure from the Python library is utilized to perform a k-nearest neighbors search, identifying the closest k poses to the current scan pose. These poses are then filtered using the distance threshold to select only those within the desired range.The identified overlapping scans are then processed using the ICP algorithm to estimate the transformation needed for their alignment.

### 3.4.4 Mahalanobis Distance Check

The EKF-SLAM update step incorporates measurements $z_k$ from ICP and $h_k$ from the observation equation. However, inaccuracies in ICP displacement measurements can compromise the estimated robot pose. To address this, a Mahalanobis distance check was introduced as a quality control measure for the ICP-derived displacement estimates.

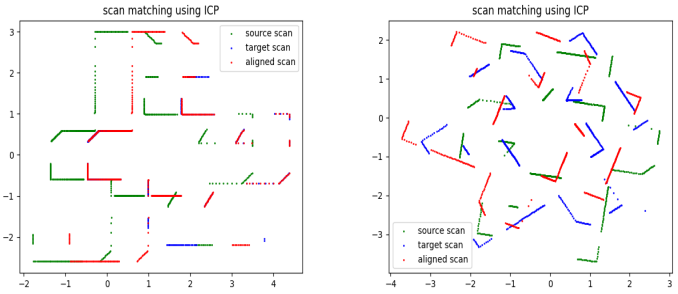
In our approach, the Mahalanobis distance between the ICP displacement measurement and the expected displacement distribution was calculated, considering previous observations and measurement uncertainty. The following equations are used for data association:

$$v = z_k - h\left(x_{k|k-1}\right) \tag{20}$$

$$S = \left(J_{1\oplus} J_{\ominus}\, {}^{N}P_{B_j}\, J_{\ominus} J_{1\oplus}^{T} + J_{2\oplus}\, {}^{N}P_{B_k}\, J_{2\oplus}^{T} + V_k R_k V_k^{T}\right) \tag{21}$$

$$D^2 = v^T S^{-1} v \tag{22}$$

The mahalanobis distance is compared with the chi-square threshold, denoted as $\chi^2_{\rho,\phi}$. If $D^2 < \chi^2_{\rho,\phi}$, the data association is deemed valid as in figure 4a. Conversely, if the Mahalanobis distance exceeds the threshold, as in figure 4b ; it indicates a potential outlier or error in the measurement. In such cases, the measurement is disregarded, and the EKF update step is skipped. This procedure ensures that only valid measurements are used in the SLAM algorithm.




(a) Aligned Scans (b) misaligned scans

Fig. 4: ICP overlapping SCAN matching transformation result

### 3.4.5 State Update

In the iterative process of SLAM, the EKF update step plays a pivotal role in refining the state estimate and minimizing uncertainty by integrating valid measurements. This crucial integration is governed by a set of equations:

$$K_k = {}^{N}\bar{P}_k H_k^T \left(H_k\, {}^{N}\bar{P}_k H_k^T + V_k R_k V_k^T\right)^{-1} \tag{23}$$

$${}^{N}\hat{x}_k = {}^{N}\hat{\bar{x}}_k + K_k \left(z_k - h\left({}^{N}\hat{\bar{x}}_k, z_p, \mathcal{H}_p\right)\right) \tag{24}$$

$${}^{N}P_k = \left(I - K_k H_k\right) {}^{N}\bar{P}_k \left(I - K_k H_k\right)^T \tag{25}$$

Here, $K$ represents the Kalman gain, ${}^{N}x_k$ is the updated state estimate, and ${}^{N}P_k$ is the updated covariance matrix at time $k$, and $Hk$ and $Vk$ are the observation jacobians.

These updates ensure that the state estimate is refined and the associated uncertainty is minimized by incorporating the most recent and accurate measurements into the SLAM process.

#### 3.4.6 Pose Optimization

To enhance the computational efficiency of the SLAM algorithm, a subsampling technique was implemented to reduce the number of scans considered during the mapping process. This technique involved registering a scan into the map only after a certain distance threshold was reached by the robot or rotated by a certain angle. While this condition limited the inclusion of scans, it was not sufficient to bound the size of the state vector, as longer runs or greater distances would significantly increase the state vector dimensionality.

To address this issue and maintain computational efficiency, a strategy was employed to prune the state vector and scan history periodically. Once the state vector reaches a predefined maximum number of scans, the algorithm deletes k even poses and their corresponding scans. The value of k , distance threshold and maximum number of scans can be adjusted based on the specific requirements of the environment and the application needs. This approach helps to bound the state vector's size, ensuring that the algorithm remains efficient even over extended periods of operation.

Additionally, an efficient management approach could be adopted by merging every five consecutive poses' scans into a single pose, thereby preserving the essential scan information while reducing the total number of poses. Both approaches effectively controlled the state vector size and maintained the integrity of the mapping process in real-time.

### 3.5 Experimental Setup

The Kobuki TurtleBot3 is equipped with various components, including a Raspberry Pi micro-controller, a 2D lidar sensor for environment sensing, an IMU for orientation estimation, and wheel encoders for odometry. We used Kobuki TurtleBot3 to test the simulation results that we obtained in the real experiment. Before trying on the real robot, the Stonefish simulator (as in fig:5) was utilized in simulation experiments to replicate the robot's movement, physical environment, and sensor readings. In addition, for visualization, RVIZ was used.

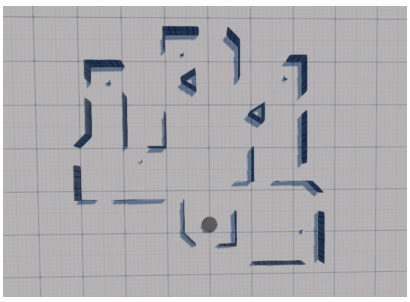
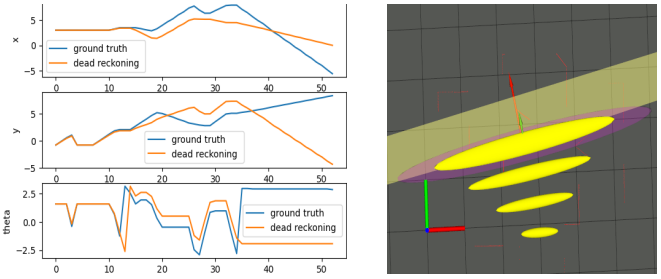


Fig. 5: Stonefish Simulation Environment

## 4 Result and Discussion

### 4.1 Prediction Using Dead reckoning

The consequence of relying solely on dead reckoning for navigation without correction updates is a continuous accumulation of errors in the estimated position of a moving object, as illustrated in Figure 6b. The yellow ellipse in the figure illustrates the robot's uncertainty, which increases over time. This divergence between the estimated and actual paths is depicted in Figure 6a, where the blue line representing the estimated path drifts further away from the orange line representing the ground truth path as time progresses. Thus, the result of dead reckoning without correction updates is an increasingly inaccurate representation of the robot's true position and trajectory.

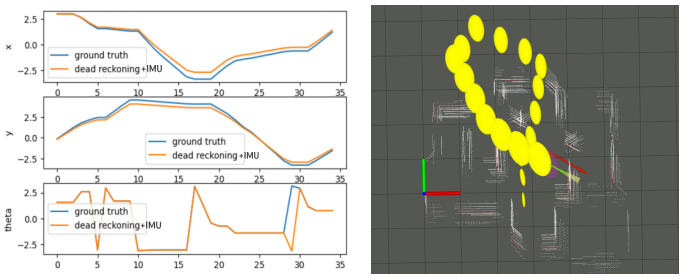



(a) Position (x,y,theta) (b) Uncertainty over time

Fig. 6: Prediction result using Dead reckoning only over time

### 4.2 Incorporate IMU Update(EKF)

In this sub-section, Inertial Measurement Unit (IMU) updates were integrated into the dead reckoning algorithm for robot pose estimation, resulting in enhanced accuracy. A notable consequence was the continuous update of the robot's heading angle and position estimates. Figure 7a illustrates the robot's position with IMU updates (orange) compared to ground truth (blue), showing closer alignment, particularly in heading angle estimation, compared to Figure 6a. Additionally, Figure 7b demonstrates reduced uncertainty using IMU updates compared to Figure 6b, indicating significant improvement.



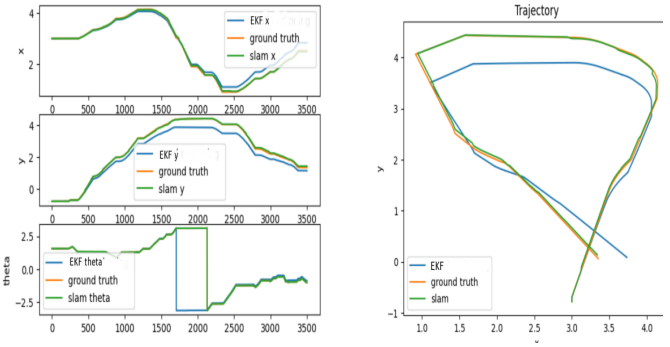

(a) Position (x,y,theta) (b) Uncertainty over time

Fig. 7: result using IMU Update(EKF) over time

### 4.3 SLAM Update

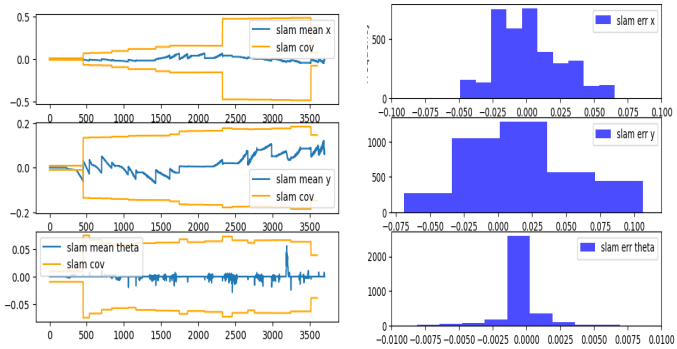
Here, the state update process with scan matching in pose-based SLAM had been augmented. Figure 8a compares the

results of this augmented approach with prediction assisted by IMU updates(EKF). The figure depicts the position (x, y, theta), while Figure 8b showcases the trajectory of the robot over time, indicating prediction (blue) and SLAM updates (green) alongside the ground truth (orange). This comparison underscores the efficiency of incorporating scan matching into the state update process, as evidenced by the closer alignment of estimations with the ground truth, contrasting with the significant deviations observed in the prediction results.




(a) Position (x,y,theta) (b) XY trajectory plot

Fig. 8: Pose Based SLAM position result scan matching update after robot moves 0.8 meters or rotation of pi/4

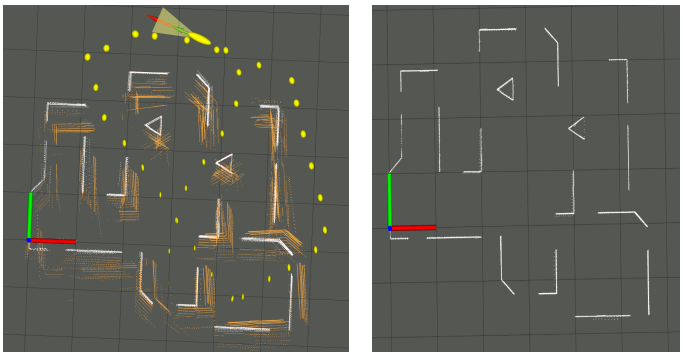



(a) 3-$\sigma$ bound estimated error (b) Histogram of the estimated error.

Fig. 9: pose based slam estimated error and histogram

Figure 9a shows the estimated error, with all errors falling within the 3 $\sigma$ bound, corresponding to a 99.7 percent confidence interval.Additionally, Figure 9b displays a Gaussian-shaped histogram of the estimated error, indicating consistent filter behavior.

Also, figure 10a shows the map generated using dead reckoning (orange color ) and SLAM update (white color ). The map is created by recording the pose of the robot at each scan interval and converting the entire scan into a world-frame map. Compared to the ground truth map Figure 10b, the SLAM map (white) is accurately aligned, while the dead reckoning map is shifted and misaligned.

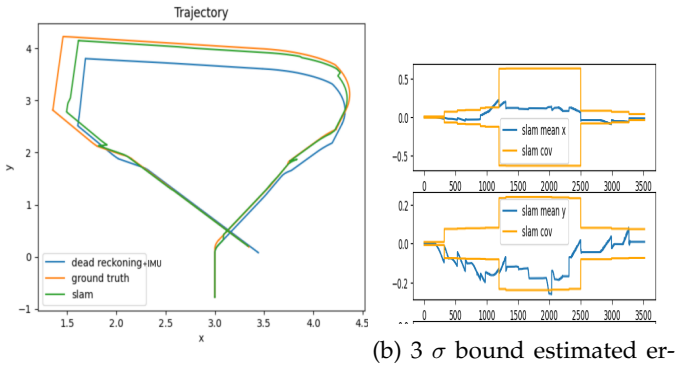


(a) EKF vs SLAM map (b) ground truth map

Fig. 10: comparison of dead reckoning map with IMU update , slam map with ground truth map

### 4.4 Effect of scan taking distance threshold

The effect of increasing the scan interval on pose-based SLAM is depicted in Figure 11a, revealing a trajectory estimation with minor drift when the interval is set to 1.2 meters. Conversely, Figure 8 showcases improved estimation accuracy with a smaller interval of 0.8 units between scans. Notably, frequent scans offer enhanced accuracy at the cost of higher state vector memory usage to store each pose and scan. Conversely, increasing scan distances reduce computation and memory usage but may compromise accuracy. Figure 9a further elucidates these effects, with some errors surpassing the 3$\sigma$ bound due to increased scan matching distances.




(a) XY trajectory (b) 3 $\sigma$ bound estimated error

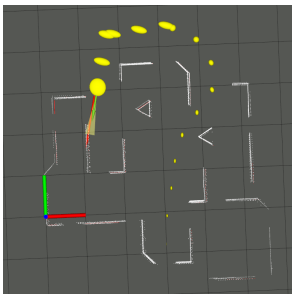
Fig. 11: Pose Based SLAM position result scan matching update after robot moves 1.2 meters or rotation of pi/4

#### 4.4.1 *Loop Closure*

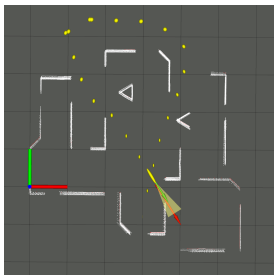
Figure 12 demonstrates the effect of loop closure on uncertainty reduction and map correction. Figure 12a shows the increasing uncertainty over time in pose-based SLAM when the overlapping scan is only from the previous scan. In contrast, Figure 12b illustrates how uncertainty is reduced and the map is corrected when a previously visited area is revisited and the loop is closed.

### 4.5 Real-time Implementation

The methodology outlined in the previous sections was successfully applied to a real-world robotic platform, yielding



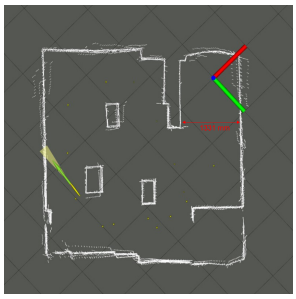
(a) before loop closure



(b) after loop closure

Fig. 12: loop closure: decrease in uncertainty and map correction

(a) Real Distance

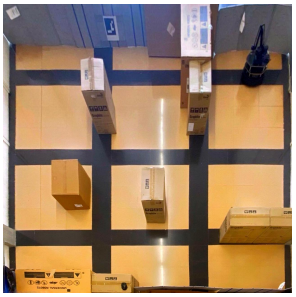


(b) Simulation Distance

Fig. 14: Simulation map accuracy compared to the real map

impressive results. Figure 14 demonstrates the remarkable accuracy of the map generated by the real robot when compared to ground truth data. However, transitioning from simulation to the real robot presented several challenges, including:

- Flipped IMU Axis: During the project's later stages, it was discovered that the IMU sensor's axis was flipped. This misalignment negatively affected the estimation of the robot's orientation, further contributing to inaccuracies in the SLAM results.
- Time Synchronization: Ensuring time synchronization between the laptop roscore and the roscore running on the Raspberry Pi posed a significant challenge. Synchronizing the time stamps between the two systems ensured seamless communication and data consistency.
- Transformation Matrix Discrepancies: The transformation matrices between the robot's subsystems, such as the robot base, lidar, and camera, were not consistent with those in the simulation.



(a) Map top view

(b) Map isometric view

Fig. 13: Real-time implementation environment

## 5 Conclusion

In conclusion, this research demonstrates the successful implementation of a Pose-Based EKF SLAM algorithm utilizing ICP scan matching on a Kobuki Turtlebot. Through the integration of data from a 2D LIDAR, IMU, and wheel encoders, the system achieves real-time mapping and localization with high accuracy in both simulated and real-world environments. The pose optimization technique effectively manages the state vector size, balancing computational efficiency with the preservation of essential pose history for accurate loop closure. The experimental results confirm the robustness and reliability of the proposed approach, indicating its potential for further applications in more complex autonomous navigation scenarios.

## 6 Acknowledgment

The authors would like to thank their supervisors Prof. Ridao Rodriguez, Pedro and Pi Roig, Roger for their invaluable support and guidance throughout this research.

## 7 References

[1] F. Martinelli and F. Romanelli, "A SLAM algorithm based on range and bearing estimation of passive UHF-RFID tags," 2021 IEEE International Conference on RFID Technology and Applications (RFID-TA), Delhi, India, 2021, pp. 20-23, doi: 10.1109/RFID-TA53372.2021.9617420.

[2] A. Mallios, P. Ridao, D. Ribas, F. Maurelli, and Y. Petillot, "EKF-SLAM for AUV navigation under probabilistic sonar scanmatching," in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 4404–4411, 2010, organization: IEEE.

[3] A. Mallios, P. Ridao, D. Ribas, and E. Hern´andez, "Scan matching SLAM in underwater environments," Autonomous Robots, vol. 36, pp. 181–198, 2014, publisher: Springer.

[4] W. Wei, M. Ghafarian, B. Shirinzadeh, A. Al-Jodah, and R. Nowell, "Posture and Map Restoration in SLAM Using Trajectory Information," Processes, vol. 10, no. 8, p. 1433, 2022, publisher: MDPI.

[5] P. Ridao, Probabilistic Robot Localization & Mapping, Unpublished manuscript.

## 8 Appendix

Link to real-time SLAM results and simulation videos