Landmark Localization Using a Right Invariant Kalman Filter on an MBot

Cameron Harris[1], Kamil Nocon[2], Max Wu[3]

***Abstract*— In this paper, we apply a Right Invariant Kalman Filter on an MBot equipped with standard odometry (IMU and encoders) as well as a camera for AprilTag detection to perform landmark localization. The MBot was commanded to drive in a 340 x 596 cm square using keyboard commands where two AprilTag landmarks are placed along each edge. We demonstrated that applying this form of Kalman Filter has improved the MBot's tracking by 78%.**

I. INTRODUCTION

The Right-Invariant Kalman Filter (RIEKF) is a state estimation technique of systems with nonlinear dynamics and measurements. The difference between an Extended Kalman Filter (EKF) and a regular Kalman Filter (KF) is the fact that EKF linearizes around an operating point while a regular KF can be only applied to linear measurements and dynamics. The concept of right invariance means that the state of the system does not change under a transformation of a Lie group element. In an application of a differential drive mobile robot such as MBot, this is extremely beneficial as its orientation and position can be represented as Lie groups such as SE(2).

The concept of localization is determining a robot's position and orientation in an environment without prior knowledge while leveraging the robot's sensors and dynamics. The use of landmarks allows the robot to detect known features within its environment to localize itself. In the case of MBot, landmarks are detected via AprilTags which is then fed into the RIEKF to correct its state after predicting its position and orientation through its dynamics.

MBot is a differential drive mobile robot shown in Figure 1 that is used in many of University of Michigan's Robotics courses. It is equipped with a Raspberry Pi 5 which allows fast processing of its various onboard sensors such as camera, inertial measurement unit (IMU), encoder. There are two brushed DC motors that drive each of the robot's wheels which is controlled by a Pico Shim, a lower level controller. The onboard sensor suite provides us the capability to gather the data needed to correctly apply the RIEKF to perform a better localization than relying on just the MBot's encoders, which is the control in the experiment.

II. METHODOLOGY

A. Right-Invariant Extended Kalman Filter (RIEKF)

The RIEKF follows the same structure as a standard KF, a prediction step which propagates the state one step in time given a model of the system, and then a correction step which utilizes sensor measurements to correct it depending on the model's accuracy or if there is any process noise.

Fig. 1. The differential drive robot used for this project - MBot

Measurements are also subject to noise, so a gain is applied to the correction which can change depending on how reliable the model or measurement is given their covariances. A detailed step-by-step is explained below, as well as a high level overview shown in Figure 2.

1) Prediction: In the prediction step of the RIEKF, the robot's control input $u = [v, \omega]$, the commanded forward linear velocity and rotational velocity fed to the MBot and then calculated through the encoders, is fed through the system's dynamics to produce the states $states = [x, y, \theta]$ where x, y, and θ is the robot's position and heading. The dynamics of MBot is shown in eq.(1-3) which was taken from Chapter 5 of Probabilistic Robotics [1].

$$x_{k+1} = x_k - \frac{\hat{v}}{\hat{\omega}} sin(\theta_k) + \frac{\hat{v}}{\hat{\omega}} sin(\theta + \hat{\omega}\Delta t) \tag{1}$$

$$y_{k+1} = y_k + \frac{\hat{v}}{\hat{\omega}} cos(\theta_k) - \frac{\hat{v}}{\hat{\omega}} cos(\theta + \hat{\omega}\Delta t) \tag{2}$$

$$\theta_{k+1} = \theta_k + \hat{\omega}\Delta t \tag{3}$$

Given that the robot can be represented in $SE(2)$ group, we can rewrite the state of the robot as:

$$X_k = \begin{bmatrix} R_k & p_k \\ 0 & 1 \end{bmatrix} \in SE(2) \tag{4}$$

where u is the twist, ζ, and can be rewritten as:

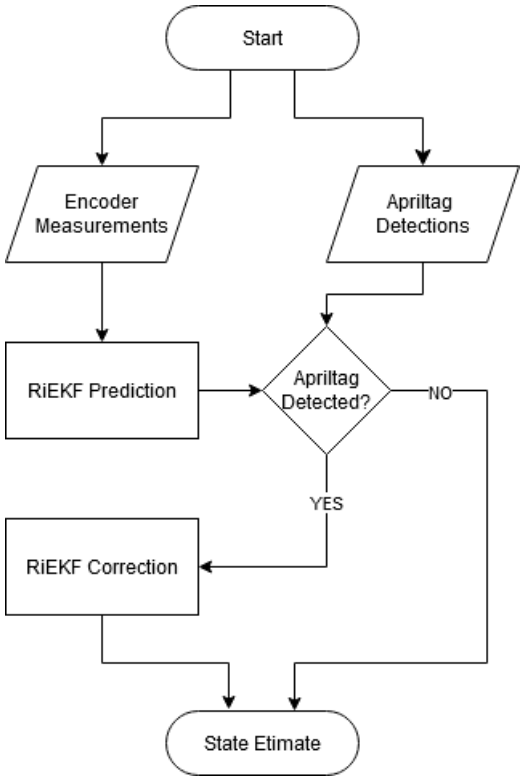

Fig. 2. State estimation through RiEKF flow diagarm

$$\zeta_k^\wedge = \begin{bmatrix} \omega_k^\wedge & v_k \\ 0 & 0 \end{bmatrix} \tag{5}$$

Using the robot dynamics given in Equations 1-3, the next state, x_{k+1}, is predicted with the command velocities. The current pose and the predicted pose are rewritten in $SE(2)$ group and the twist between the two poses is calculated as:

$$\zeta_k^\wedge = \log\left(X_k^{-1} X_{k+1}\right) \tag{6}$$

Using this twist vector, the next state calculated by:

$$X_{k+1} = X_k \exp\left(\zeta_k^\wedge\right) \tag{7}$$

This method of calculating the next state via the motion model, calculating the twist between poses, and then calculating the next state via the twist was used to increase the accuracy of our model by calculating the actual velocities between the two states.

The covariance of the pose estimate is updated via:

$$\Sigma_{k+1} = \Sigma_k + Ad_{X_k} W Ad_{X_k}^T \tag{8}$$

where W is the motion noise, and the Ad_{X_k} is the adjoint calculated by

$$Ad_{X_k} = X_k \zeta_k^\wedge X_k^{-1} \tag{9}$$

This is then called for every time step to get a prediction of the next state which can then be corrected when the MBot detects landmarks in the correction step.

The pseudocode for the prediction step of the RI-EKF algorithm is shown in Algorithm 1.

Algorithm 1: Prediction(X, Σ, v, ω)

$x, y, \theta \leftarrow getState(X)$
$\bar{x}, \bar{y}, \bar{\theta} \leftarrow \mathbf{F}(x, y, \theta, v, \omega)$
$\bar{X} \leftarrow poseMatrix(\bar{x}, \bar{y}, \bar{\theta})$
$u^\wedge = \log\left(X_k^{-1} X_{k+1}\right)$
$X = \exp\left(u^\wedge\right)$
$Ad_X = X u^\wedge X^{-1}$
$\Sigma = \Sigma + Ad_X W Ad_X^T$
Return X, Σ

2) Correction: Due to a variety of factors such as drift and wheel slip, the state that is predicted does not yield very good results and thus needs to be corrected via a landmark in a known location. When the MBot detects a landmark, the correction step is then called. If the ground truth of the landmark is known in the x and y coordinates, we can compare the robot's measured landmark to the ground truth to apply the correction.

$$b = [x_{groundtruth}, y_{groundtruth}, 1] \tag{10}$$

The measurement Jacobian is also necessary to properly correct the MBot's state which can be defined as

$$H = \begin{bmatrix} m^2 & -1 & 0 \\ -m^1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix} \tag{11}$$

However, since this matrix is a 2x3 given the last row of zeros, another landmark needs to be detected at the same time in order to correctly estimate the robot's heading. This can be done using the same exact Jacobian but for a different landmark being detected at the time and just have it be horizontally stacked. This results in the observability Grammiam being full rank which allows the pose to be observable.

$$H = \begin{bmatrix} m_1^2 & -1 & 0 \\ -m_1^1 & 0 & -1 \\ m_2^2 & -1 & 0 \\ -m_2^1 & 0 & -1 \end{bmatrix} \tag{12}$$

$$rank(H) = 3 \tag{13}$$

The Kalman gain K can then be calculated as

$$K = \Sigma H^T S^{-1} \tag{14}$$

where S is the innovation covariance calculated as

$$S = H \Sigma H^T + X V X^T \tag{15}$$

where V is the measurement noise covariance and mu is the mean of the state in the Lie Algebra.

The innovation ν can then be calculated using the distance from the robot to the landmark(s) Y and b, the landmark ground truth.

$$Y = \begin{bmatrix} \Delta x_1 & \Delta_1 y & 1 \\ \Delta x_2 & \Delta_2 y & 1 \end{bmatrix} \tag{16}$$

$$\nu = \begin{bmatrix} X & 0 \\ 0 & X \end{bmatrix} Y - b \tag{17}$$

Since Y has measurements from landmark 1 and landmark 2 horizontally stacked, the block diagonal of μ would need to be used. This innovation is then multiplied with the Kalman gain to get the new corrected mean and covariance.

$$X = exp^{(K\nu)\wedge}\mu \tag{18}$$

$$\Sigma = (I - KH)\Sigma(I - KH)^T + K(XVX^T)K^T \tag{19}$$

The pseudocode for the correction step of the RI-EKF algorithm is shown in Algorithm 2.

Algorithm 2: Correction($X, \Sigma, Y_1, Y_2, id_1, id_2$)

$b_1 = getLandmarkPosition(id_1)$
$b_2 = getLandmarkPosition(id_2)$
$S = H\Sigma H^T + XVX^T$
$K = \Sigma H^T S^{-1}$
$Y = \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix}$
$b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$
$\nu = \begin{bmatrix} X & 0 \\ 0 & X \end{bmatrix} Y - b$
$X = exp^{(K\nu)\wedge}X$
$\Sigma = (I - KH)\Sigma(I - KH)^T + K(XVX^T)K^T$
Return X, Σ

3) RI-EKF: Using the initial state and covariance estimate, along with the motion model noise and measurement model noise, the prediction and correction steps can be performed. The complete RI-EKF is shown in Algorithm 3.

Algorithm 3: RI-EKF Localization

Given: **F**: Dynamics Model;
X_{init}: Initial State (Lie Algebra);
Σ_{init}: Initial State Covariance;
W: Motion noise covariance;
V: Measurement noise covariance;
$X \leftarrow X_{init}$
$\Sigma \leftarrow \Sigma_{init}$
while *task not complete* **do**
  $v, \omega \leftarrow calcVelFromEncoders()$
  $X, \Sigma \leftarrow Prediction(X, \Sigma, v, \omega)$
  if $AprilTagDections()$ **then**
    $Y_1, Y_2, id_1, id_2 \leftarrow AprilTagDections()$
    $X, \Sigma \leftarrow Correction(X, \Sigma, Y_1, Y_2, id_1, id_2)$

B. Apriltag Detection

Apriltags, invented at the University of Michigan, are fiduciary markers which can easily be detected by a camera with both a unique identifier and pose information. The open-source repository provided for Apriltags can be implemented with Python to work out of the box. It is worth noting that the camera used for detection must be calibrated, for which we utilized OpenCV's "Calibrate Camera" functionality.

C. Data Collection

To create the dataset used to process the RIEKF, we commanded an MBot in a square of 340 x 596 cm using a total of 4 landmark pairs (landmark 1 and landmark 2 side-by-side). Placed along each edge of the square as shown in Figure 3. The MBot then starts at one of the corners defined as (0,0) and is commanded to drive on the square by an operator sending keyboard commands manually. The data collected was the time, calculated v_x, v_y, and ωz from the wheel encoders, Apriltag Δx and Δy as well as the calculated odometry x, y, and θ.

$$data = \begin{bmatrix} t \\ v_x \\ v_y \\ \omega_z \\ Apriltag_{ID} \\ \Delta X_{ID_1} \\ \Delta Y_{ID_1} \\ \Delta X_{ID_2} \\ \Delta Y_{ID_2} \\ x \\ y \\ \theta \end{bmatrix} \tag{20}$$

This is all then saved into a single log file using (Lightweight Communications and Messaging) LCM logger and then converted into a $.csv$ for data input to the filter.

III. RESULTS

Upon running the experiment, a 2D representation of the results is shown in Figure 2, where the ground truth of the square is plotted in blue, the estimated state using the RIEKF is in orange, and the odometry is plotted in green. The landmarks are also shown on the plot as pink markers.

To compare the improvement between the RIEKF and odometry, the root-mean-square error (RMSE) was plotted over the course of the experiment and compared on the same axis as shown in Figures 3 and 4.

Implementing the filter resulted in better position tracking of the MBot by 78% (0.16m to 0.04m).

A link to the YouTube of the presentation can be found here: ROB530 Presentation

IV. EVALUATION

A. Discussion

We see that Apriltag landmark localization with a Right-Invariant Extended Kalman Filter is effective in improving pose estimation and reducing odometry drift for even low

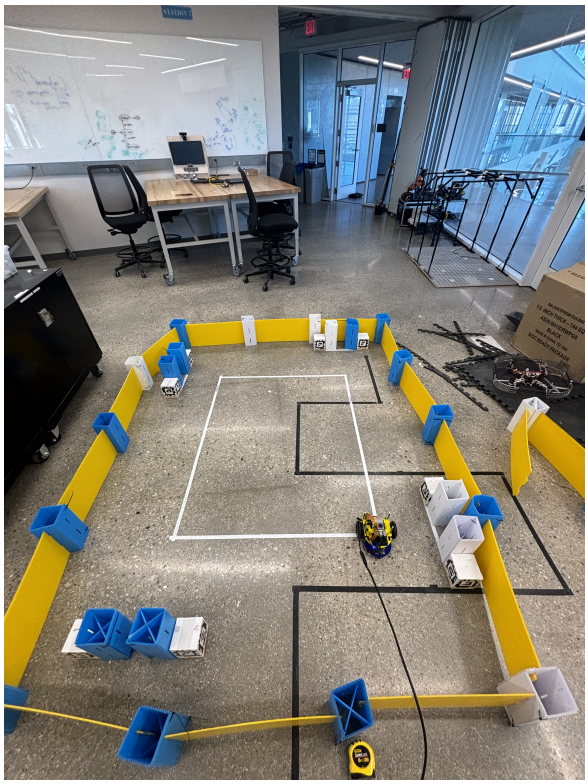

Fig. 3. Experiment set-up for data collection

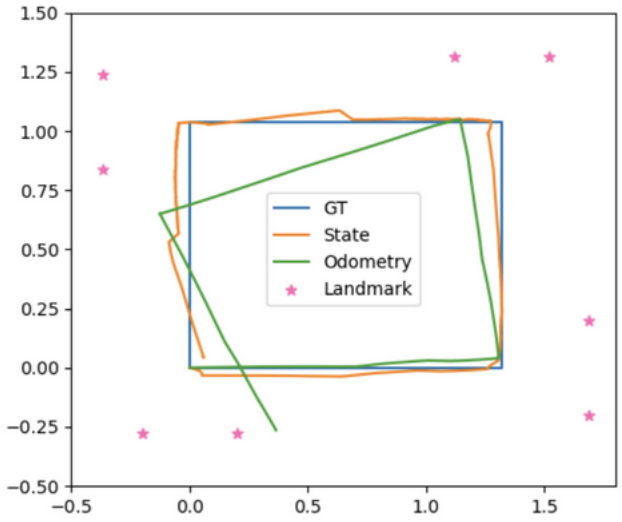

Fig. 4. Results of the experiment

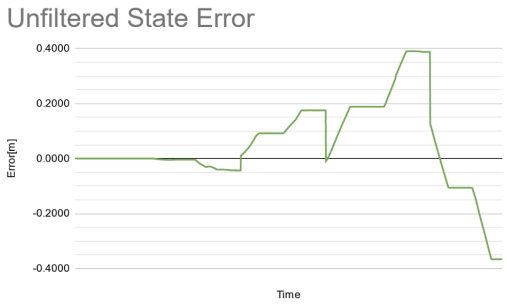

Fig. 5. Unfiltered RMSE Plot

cost robots. With a camera being the only additional sensor needed, the pose estimation of our MBot was improved by over 75% from odometry calculations through encoder measurements alone. With computational and/or cost constraints, this is a viable solution for mitigating odometry drift.

The experiments conducted yield reproducible results with an LCM log file. However, we find that the setup required to create a robust landmark system with Apriltags to be tedious, which leads to a more simplified ”bird’s eye” setup as described in section IV-B.

B. Further Improvements

The obvious improvement to be made on this experimentation is to incorporate the filter to run live on the MBot. While not implemented here, this would be a simple integration into the MBot as an augmentation to the existing LCM ecosystem.

We benefit from the problem simplification into the 2D world, yet are interested in exploring how this system behaves in a 3D, 6-degree of freedom state estimation such as a drone.

Additionally, while the improved error results are compelling, we are limited by the simplicity of the motion model. In order to further increase tracking performance of the MBot, the on-board IMU can be utilized to provide a fused, more accurate motion model that accounts for common issues encountered such as wheel slippage. Beyond this, a simple visual odometry algorithm could be fused as well to provide stronger pose estimation and resulting in an even more accurate final result.

As introduced briefly in section IV-A, the setup process involved with creating an environment of accurately positioned Apriltags in detectable locations proved to be challenging. As a result, we consider a future implementation which employs a ”bird’s eye” approach. Instead of Apriltag landmarks, the robot is tagged with 2 apriltags on the body frame, which are observable from a single bird’s eye camera. Whenever the robot comes within frame of the camera, the same localization algorithm can correct the robot state. This greatly reduces complexity as the observer camera acts as the ”landmark”, effectively shrinking the landmark requirement to one.

V. CONCLUSIONS

Overall, the implementation of the RIEKF resulted in a significant improvement over regular odometry when calculated offline. This means that any form of correction where the localization of the MBot can be compared to some ground truth aside from itself can yield consistently better tracking given noisy measurements from the AprilTags. This filter can also be implemented in real-time to give the MBot better tracking as it makes its way around the square, which can help in developing more advanced localization or

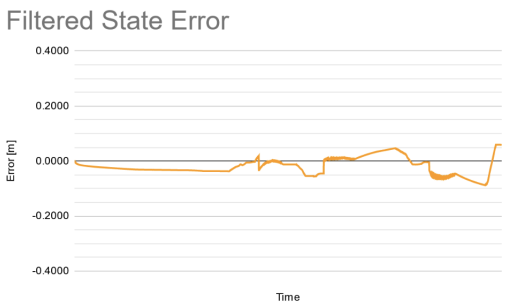

Fig. 6. Filtered RMSE Plot

controls algorithms in the future.

REFERENCES

[1] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.*, ser. Intelligent robotics and autonomous agents. MIT Press, 2005.
[2] M. Ghaffari, "Lecture notes for mobile robotics," University of Michigan, Ann Arbor, 2021.
[3] F. Liu, "Riekf_localization_se2.py," 2020.