State Estimation

Yuvan Sharma Ziteng (Ender) Ji

***Abstract*—State estimation is a fundamental problem in robotics, enabling robots to estimate their state using control inputs and sensor measurements. This project explores three classical estimation techniques: Dead Reckoning, Kalman Filter (KF), and Extended Kalman Filter (EKF) for robotic localization. We first implement Dead Reckoning, which relies solely on the system dynamics to predict the state, as a baseline, which struggles to track states accurately as error accumulates over time. To improve accuracy, we implement the Kalman Filter, which incorporates sensor measurements for correction in linear systems. For nonlinear systems, we implement the Extended Kalman Filter, which linearizes the system at each timestep to apply the Kalman update. By evaluating these methods under process and measurement noise, we analyze their performance, accuracy, and robustness.**

I. Introduction

State estimation is a critical component of autonomous systems, allowing robots to infer their state using control inputs and sensor measurements. Accurate state estimation is essential for tasks such as localization, navigation, and mapping, where precise knowledge of a robot's position and orientation is required. In this project, we explore three fundamental estimation techniques: Dead Reckoning, Kalman Filter (KF), and Extended Kalman Filter (EKF), each offering different approaches to estimating system states.

Dead Reckoning estimates the state by integrating the system's motion model over time, using only control inputs. While simple and computationally efficient, it suffers from cumulative error and drift due to unmodeled disturbances. The Kalman Filter improves upon this by incorporating sensor measurements to correct state estimates, assuming a linear system with Gaussian noise. For more complex systems with nonlinear dynamics, the Extended Kalman Filter extends the Kalman Filter by linearizing the system at each timestep.

State estimation plays a crucial role in real-world applications such as autonomous vehicles, robotic manipulation, and drone navigation. Self-driving cars rely on state estimation to track their position accurately, even when GPS signals are weak. In aerial robotics, drones use onboard sensors and estimation algorithms to maintain stability and navigate dynamic environments. By studying these estimation methods, we gain a deeper understanding of how robots achieve reliable autonomy in uncertain and noisy conditions.

II. Method

A. Dead Reckoning

Dead reckoning is a classical localization method that estimates the system's state using only its motion model and control inputs, without incorporating sensor measurements. The method originates from solving ordinary differential equations (ODEs) that describe the system's continuous-time dynamics. Given a general system of the form

$$\frac{dx}{dt} = f(x,u),$$

where x represents the system state, u represents the control inputs, and $f(x,u)$ describes the system's evolution over time, we seek to approximate the state at discrete time steps. The exact solution requires integration, but since it is often intractable, we employ Euler's method as a first-order numerical approximation

$$x[t+1] = x[t] + f(x[t], u[t]) \cdot \Delta t.$$

This equation, referred to as Equation (12) in the project document, serves as the foundation for dead reckoning. The theoretical motivation for this equation comes from the Taylor series expansion of $x(t)$, where higher-order terms are neglected

$$x(t+\Delta t) = x(t) + \left.\frac{dx}{dt}\right|_t \cdot \Delta t + \mathcal{O}(\Delta t^2).$$

By dropping the higher-order terms, we approximate the system's evolution using only first-order information. This simplification makes dead reckoning computationally efficient but introduces truncation errors that accumulate over time.

In this project, dead reckoning is applied to a differential-drive unicycle model, where motion is controlled by two independently driven wheels. The system state is defined as

$$x = \begin{bmatrix} \varphi & x & y & \theta_L & \theta_R \end{bmatrix}^T,$$

where φ is the robot's orientation, x and y represent its position in the 2D plane, and θ_L, θ_R are the rolling angles of the left and right wheels. The control input consists of the wheel angular velocities,

$$u = \begin{bmatrix} u_L \\ u_R \end{bmatrix}.$$

Using the project document's provided motion model, the continuous-time system dynamics are given by

$$\dot{\varphi} = -\frac{r}{2d}u_L + \frac{r}{2d}u_R,$$

$$\dot{x} = \frac{r}{2}\cos\varphi(u_L + u_R),$$

$$\dot{y} = \frac{r}{2}\sin\varphi(u_L + u_R),$$

$$\dot{\theta}_L = u_L, \quad \dot{\theta}_R = u_R.$$

Applying Euler's method to discretize the equations, we obtain the update rules for each state variable:

$$\varphi[t+1] = \varphi[t] + \left(-\frac{r}{2d}u_L + \frac{r}{2d}u_R\right)\Delta t,$$

$$x[t+1] = x[t] + \left(\frac{r}{2}\cos\varphi[t](u_L + u_R)\right)\Delta t,$$

$$y[t+1] = y[t] + \left(\frac{r}{2}\sin\varphi[t](u_L + u_R)\right)\Delta t,$$

$$\theta_L[t+1] = \theta_L[t] + u_L\Delta t,$$

$$\theta_R[t+1] = \theta_R[t] + u_R\Delta t.$$

These equations describe how the system state evolves based on the latest control inputs. The dead reckoning method operates iteratively, updating the state at each timestep by propagating the previous state forward using the current control input. A similar model is developed for the planar quadrotor.

Despite its simplicity, dead reckoning suffers from cumulative error accumulation due to numerical integration errors and unmodeled disturbances. Small inaccuracies in the control inputs or initial conditions can lead to significant drift over time, limiting the method's long-term reliability. This limitation motivates the use of the principle of feedback, which leads to the Kalman Filter.

B. Kalman Filter

The Kalman filter provides a method to estimate the state of a system while incorporating sensor measurements to correct for errors. Unlike dead reckoning, which solely relies on integrating the system dynamics forward in time, the Kalman filter updates state estimates using both the motion model and measurement model. This makes it particularly effective for reducing drift and uncertainty in state estimation.

The fundamental assumptions in Kalman filtering are that 1) the system dynamics are linear and 2) that both the process noise and measurement noise are normally distributed with zero mean and covariance matrices Q and R, respectively. The system dynamics in our implementation for the unicycle model thus assume a fixed bearing at $\varphi = \pi/4$, leading to a linearized state transition model. The state evolves according to:

$$x[t+1] = Ax[t] + Bu[t] + w[t],$$

where $w[t] \sim \mathcal{N}(0, Q)$ represents process noise. The measurement model follows:

$$y[t] = Cx[t] + v[t],$$

where $v[t] \sim \mathcal{N}(0, R)$ represents measurement noise. Given these models, the Kalman filter follows a predict-update cycle:

1. State Extrapolation:

$$x_{t+1|t} = Ax_t + Bu_t.$$

2. Covariance Extrapolation:

$$P_{t+1|t} = AP_tA^T + Q.$$

3. Kalman Gain Calculation:

$$K_{t+1} = P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1}.$$

4. State Update:

$$x_{t+1} = x_{t+1|t} + K_{t+1}(y_t - Cx_{t+1|t}).$$

5. Covariance Update:

$$P_{t+1} = (I - K_{t+1}C)P_{t+1|t}.$$

In our implementation, the matrices are defined as:

$$A = I_{4\times 4}$$
$$B = \begin{bmatrix} \frac{r}{2}\cos\varphi & \frac{r}{2}\cos\varphi \\ \frac{r}{2}\sin\varphi & \frac{r}{2}\sin\varphi \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$
$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The matrix A assumes a linear time-invariant model, meaning the state evolves without internal changes aside from control inputs. The control matrix B maps wheel velocities to position updates, while the measurement matrix C extracts the observed positions (x, y).

The covariance matrices Q, R, and the initial state uncertainty P_0 play a crucial role in Kalman filtering. The process noise covariance Q models uncertainty in system dynamics due to unmodeled effects like wheel slippage. A higher Q value assumes that motion model predictions are less reliable, increasing reliance on sensor updates. The measurement noise covariance R reflects sensor accuracy—low values indicate highly precise measurements, while higher values account for noisy sensors. Finally, P_0 represents our initial confidence in the state estimate; large values indicate high initial uncertainty.

The parameters were fine-tuned through empirical testing by evaluating estimation accuracy and convergence behavior. We started with diagonal Q and R matrices with small values and gradually increased them to balance trust in dynamics vs. sensor data. In simulation, we leveraged ground truth data to iteratively adjust these values for optimal tracking. Our final fine-tuned P_0, Q, R matrices were all the identity.

If implementing on a real robot without access to ground truth, an alternative approach would be to estimate Q and R using sensor characterization experiments. For example, running the robot in a controlled environment and analyzing deviations between expected and observed state transitions would allow empirical estimation of process noise. Similarly, repeated sensor readings of a fixed landmark would help estimate measurement noise.

In summary, the Kalman filter significantly improves state estimation over dead reckoning by dynamically weighting the reliability of sensor data against motion model predictions. The fine-tuning of Q, R, and P_0 is critical to achieving accurate and stable estimation.

C. Extended Kalman Filter

The Extended Kalman Filter (EKF) is a nonlinear extension of the Kalman Filter. In contrast to the standard Kalman Filter, which assumes linear system evolution, the EKF approximates the nonlinear dynamics through local linearization. This is achieved by computing the first-order Taylor expansion of the system's state transition function and measurement model around the current state estimate at each timestep.

In this project, the system under consideration is a planar quadrotor, where the state evolves according to nonlinear dynamics. The full state vector consists of the quadrotor's position, velocity, and orientation, given by

$$x = \begin{bmatrix} x & z & \varphi & \dot{x} & \dot{z} & \dot{\varphi} \end{bmatrix}^T .$$

The control inputs are the thrust force and torque applied to the quadrotor, represented as

$$u = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix},$$

where u_1 is the total thrust and u_2 is the net torque. The quadrotor's motion is governed by Newton's second law, leading to the nonlinear equations

$$\ddot{x} = -\frac{u_1}{m}\sin\varphi, \quad \ddot{z} = -g + \frac{u_1}{m}\cos\varphi, \quad \ddot{\varphi} = \frac{u_2}{J}.$$

Using forward Euler discretization, we obtain the discrete-time state update

$$x[t+1] = x[t] + f(x[t], u[t]) \cdot \Delta t.$$

However, due to the presence of trigonometric functions in the state update, this function is nonlinear. The challenge in applying the Kalman Filter is that the standard formulation assumes a linear system of the form

$$x[t+1] = Ax[t] + Bu[t] + w[t],$$

where A and B are constant matrices, and $w[t]$ is process noise. Since our system is nonlinear, we cannot directly apply this formulation.

To address this problem, we approximate the nonlinear state transition function by computing its Jacobian matrix, denoted as $A(x, u)$, which represents the first-order partial derivatives of $f(x, u)$ with respect to the state variables. The relevant nonzero elements in A are:

$$A_{1,3}, A_{2,4}, A_{3,5} = 1, A_{3,2} = -\frac{u_1}{m}\cos\varphi, \quad A_{4,2} = -\frac{u_1}{m}\sin\varphi.$$

Similarly, the measurement model is nonlinear because the quadrotor estimates its position using distance measurements to a fixed landmark at $(0, 5, 5)$. The measurement equations are:

$$y_1 = \sqrt{x^2 + 5^2 + (z-5)^2}, \quad y_2 = \varphi.$$

Since y_1 is a nonlinear function of x and z, we linearize it by computing the Jacobian matrix $C(x)$, which has nonzero entries:

$$C_{0,0} = \frac{x}{\sqrt{x^2 + 5^2 + (z-5)^2}}$$
$$C_{0,1} = \frac{z-5}{\sqrt{x^2 + 5^2 + (z-5)^2}}$$
$$C_{1,2} = 1$$

With these approximations, EKF follows the predict-update steps:

1) Predict the next state using the nonlinear system dynamics.
2) Linearize the system by computing $A(x, u)$ and $C(x)$ at the current state estimate.
3) Propagate the covariance matrix:
$$P_{t+1|t} = AP_tA^T + Q.$$
4) Compute the Kalman Gain:
$$K_t = P_{t+1|t}C^T(CP_{t+1|t}C^T + R)^{-1}.$$
5) Update the state estimate using the measurement:
$$x_{t+1} = x_{t+1|t} + K_t(y_t - h(x_{t+1|t})).$$
6) Update the covariance matrix:
$$P_{t+1} = (I - K_tC)P_{t+1|t}.$$

The technique used to handle nonlinearity is first-order linearization through Jacobian matrices. This approach allows us to locally approximate the nonlinear dynamics, making it possible to apply Kalman filtering. However, this method has limitations. Since the system is only approximated locally, significant errors can arise if the state drifts too far from the linearization point. Additionally, the Gaussian noise assumption in Kalman filtering does not hold perfectly when nonlinear transformations are applied, potentially leading to suboptimal state estimates.

To mitigate these issues, more advanced filtering techniques such as the Unscented Kalman Filter (UKF) or Particle Filters can be used. The UKF propagates a set of carefully chosen sample points through the nonlinear function to better capture higher-order effects, while Particle Filters use a probabilistic approach that does not require linearization. However, these methods are computationally more expensive. Despite its limitations, EKF remains a widely used technique due to its balance between efficiency and improved accuracy compared to dead reckoning. By incorporating sensor measurements, it significantly reduces drift and improves the robustness of state estimation in nonlinear systems.

III. Experiments

A. Performance Table

For better visibility, we put the performance table in the appendix. Here we provide the combined plot, which includes plots of the estimates $\hat{x}$ compared with the ground truth x and plot of the estimated trajectory compare with the ground truth trajectory.

B. Dead Reckoning

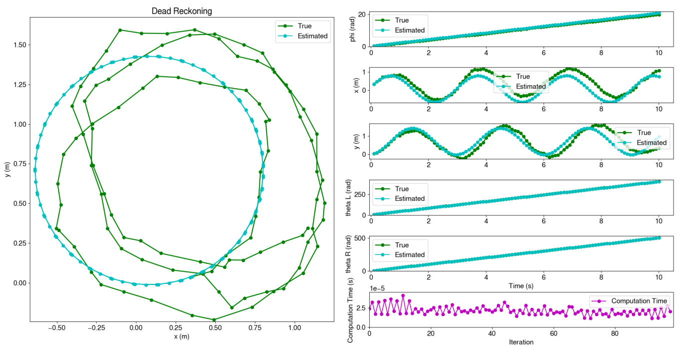

Fig. 1. Performance of Dead Reckoning for the TurtleBot

The performance of the Dead Reckoning estimator for the TurtleBot exhibits significant drift over time, which is evident from the trajectory plot comparing the true and estimated positions. The estimated trajectory (in cyan) deviates from the ground truth (in green), particularly in the xy-plane, where the estimated trajectory forms a larger, misplaced spiral relative to the actual path. This error is a direct consequence of unbounded error accumulation inherent in Dead Reckoning, as it relies purely on integrating velocity inputs without any correction mechanism. The bearing (ϕ) estimation demonstrates a generally increasing trend, matching the expected behavior; however, discrepancies appear as time progresses, contributing to positional errors. The individual x and y displacement plots further reinforce this observation, showing deviations between estimated and true values, especially over longer time horizons. Despite these errors, Dead Reckoning has the advantage of being computationally efficient, as shown in the computation time plot, where per-step calculations remain in the order of 10^{-5} seconds (see Table I in Appendix), making it suitable for real-time applications with limited processing resources. The primary limitation is that even minor inaccuracies in velocity inputs or time integration accumulate rapidly, leading to increasing errors over time. This issue is particularly evident in the TurtleBot's motion, where discrepancies between estimated and actual positions grow as the robot continues its trajectory. While Dead Reckoning may be sufficient for short-term estimation, its long-term accuracy is inadequate without external corrections, such as sensor fusion with GPS or landmark-based localization.

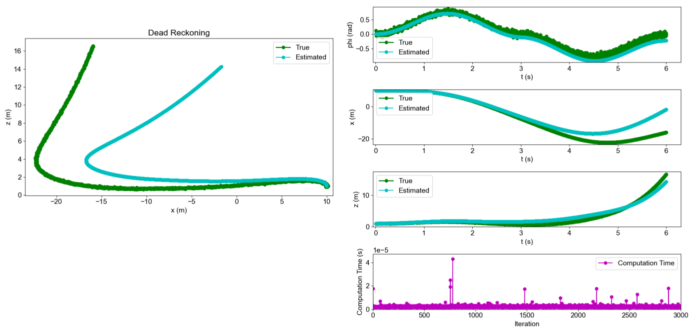

Fig. 2. Performance of Dead Reckoning for the Drone

The performance of the Dead Reckoning estimator for the drone exhibits a noticeable divergence from the true trajectory, as seen in the xz-plane plot, where the estimated trajectory (cyan) veers significantly away from the ground truth (green). This deviation is particularly pronounced over time, demonstrating the inherent limitation of Dead Reckoning—error accumulation due to the absence of corrective feedback. The bearing (ϕ) estimation initially follows the true value but begins to diverge as time progresses, introducing rotational errors that compound positional inaccuracies. The individual x and z displacement plots further illustrate this growing discrepancy, with the estimated trajectory systematically drifting away from the actual path, especially in the later stages. Despite these limitations, the computation time remains extremely low, on the order of 10^{-6} seconds per step, making Dead Reckoning an attractive choice for real-time applications where computational efficiency is critical. However, the observed error growth suggests that Dead Reckoning alone is unsuitable for long-term state estimation, particularly for aerial robots where small deviations in position can lead to large cumulative errors due to dynamic instability.

C. Kalman Filter

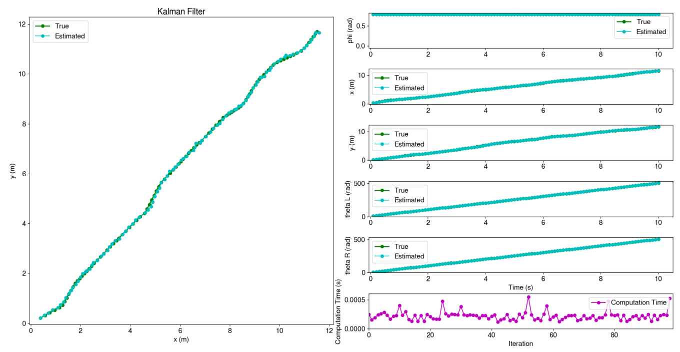

Fig. 3. Performance of the Kalman Filter for TurtleBot

The Kalman Filter estimator for the TurtleBot demonstrates high accuracy and stability in state estimation, as seen from the trajectory and individual state plots. The xy-plane trajectory plot shows that the estimated path (cyan) closely follows the ground truth (green), indicating that the Kalman Filter effectively corrects errors from process noise and sensor inaccuracies. Unlike the Dead Reckoning approach, which accumulates drift over time, the Kalman Filter leverages sensor measurements to update its estimates, resulting in a trajectory that remains well-aligned with the true path. The individual state plots further reinforce the estimator's performance. The ϕ (heading) plot remains stable throughout the execution, suggesting that the filter maintains an accurate estimate of the robot's orientation. The x and y position plots exhibit a nearly perfect overlap between the estimated and true values, which highlights the filter's ability to track position reliably. Additionally, the wheel angle (θ_L and θ_R) plots indicate that the model properly incorporates wheel odometry, keeping the estimated values tightly aligned with the actual readings.

One of the key differences between the performance of the Kalman Filter compared to Dead Reckoning is its computational efficiency. The computation time plot shows that each update step executes in the range of approximately 1.9×10^{-4} seconds, which is around 7 times slower than Dead Reckoning. This suggests that the filter is relatively computationally heavy, which makes real-time implementation on embedded systems a balancing act between accuracy and speed. Overall, the observed performance indicates that the Kalman Filter is highly effective at reducing the error present in Dead Reckoning while maintaining a slightly higher computational overhead. However, its reliance on a linear system model means that it may struggle in scenarios with strong nonlinearities, where the Extended Kalman Filter or other nonlinear estimation techniques may be necessary.

D. Extended Kalman Filter

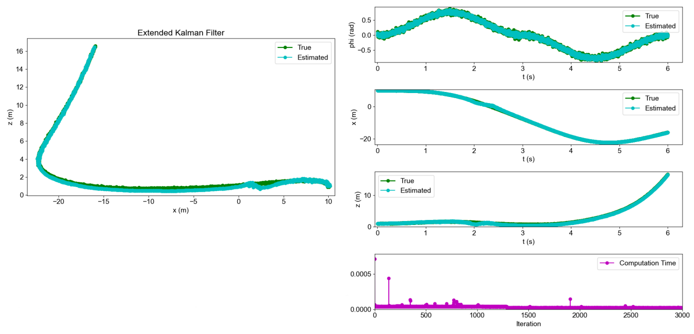

Fig. 4. Performance of Extended Kalman Filter for Drone

The Extended Kalman Filter (EKF) estimator for the drone exhibits a high degree of accuracy in state estimation, as evidenced by the trajectory and state plots. The x-z trajectory plot illustrates that the estimated path (cyan) closely aligns with the ground truth (green), suggesting that the EKF effectively mitigates drift and sensor noise by incorporating nonlinear process and measurement models. Unlike standard Kalman Filters, which assume linear system dynamics, the EKF linearizes the motion model at each timestep using Jacobian matrices, allowing it to handle the nonlinear dynamics of drone motion. This is reflected in the ϕ (orientation) plot, where the estimated values track the true values with minimal deviation, indicating that the filter correctly estimates the drone's angular position over time.

The x and z position plots further demonstrate the EKF's precision, as the estimated trajectory maintains close alignment with the true trajectory, even in dynamic motion. The small deviations observed suggest that measurement noise and model uncertainties are effectively handled, though minor discrepancies indicate that further tuning of the process noise covariance matrix (Q) or measurement noise covariance matrix (R) could further refine accuracy. The computation time plot reveals that the EKF maintains a consistent update time on the order of 4×10^{-5} seconds, which is around 20 times slower than Dead Reckoning. However, this speed is still reasonable to remain within real-time execution constraints. This increase in computational cost is expected due to the Jacobian computations required for state updates. Overall, the EKF demonstrates robust performance in tracking the drone's state while effectively accounting for system nonlinearities, making it well-suited for applications where motion dynamics deviate from strict linearity.

IV. Discussion

The performance of the three estimators varies significantly due to their fundamental assumptions and ability to handle uncertainty. Dead Reckoning relies solely on integrating control inputs over time, making it highly susceptible to cumulative errors due to unmodeled disturbances and sensor noise. This is evident in the Dead Reckoning plots, where the estimated trajectory drifts significantly from the ground truth as time progresses. For both the drone and TurtleBot implementations, the position estimates deviate over time, illustrating the limitations of open-loop estimation in real-world applications. The method is computationally efficient, but the lack of correction from external measurements causes long-term inaccuracies.

In contrast, the Kalman Filter introduces a probabilistic framework that combines noisy sensor measurements with a prediction model to refine state estimates. The performance plots for the Kalman Filter show a significant improvement over Dead Reckoning, with the estimated trajectory closely following the ground truth. This improvement is achieved by incorporating observation updates, which correct the predicted state based on sensor feedback. However, the Kalman Filter assumes linear system dynamics, which may not always hold in real-world scenarios. The estimation accuracy remains high in structured environments where the system dynamics can be well approximated by linear models.

The Extended Kalman Filter (EKF) further extends the Kalman Filter framework to handle nonlinear system dynamics by linearizing the model at each time step. This approach allows the EKF to handle more complex motion models, as seen in the drone experiment where the trajectory estimation remains close to the ground truth despite nonlinear dynamics. The plots indicate that the EKF effectively mitigates drift and maintains higher accuracy compared to Dead Reckoning. However, the need for frequent Jacobian computations increases its computational cost, which is evident from the per-step computation time measurements. Additionally, EKF's reliance on local linearization can lead to errors when the system exhibits highly nonlinear behavior that cannot be well approximated by a first-order Taylor expansion.

Comparing the three estimators, Dead Reckoning is the simplest and fastest but accumulates significant error over time. The Kalman Filter balances computational efficiency with robustness in structured environments but struggles with strong nonlinearity. The Extended Kalman Filter offers the best accuracy for nonlinear systems but at the cost of increased computation time. In real-world applications, Dead Reckoning might be used in scenarios where computational resources are extremely limited, and short-term state estimation is sufficient. The Kalman Filter is well suited for structured environments with reliable sensor measurements, such as mobile robots operating in a mapped indoor space. The EKF is ideal for aerial vehicles or autonomous systems operating in highly dynamic environments, where linear assumptions fail. A quantitative comparison of the three estimators is also provided in Table I, where we present the average error in the predictions for every state variable, as well as the average computation time per step.

To improve these estimators, one possible approach is to introduce adaptive filtering techniques that dynamically adjust process and measurement noise covariance matrices based on real-time observations. Additionally, incorporating sensor fusion, such as combining IMU data with GPS or LiDAR, could further enhance robustness. More advanced approaches like the Unscented Kalman Filter (UKF) or particle filters could also be explored to handle strong nonlinearities more effectively.

Overall, while each estimator has its advantages and drawbacks, the choice depends on the specific application requirements in terms of accuracy, computational efficiency, and robustness against noise and nonlinearity.

V. Bibliography

[1] "Runge-Kutta methods." n.d. https://web.mit.edu/10.001/Web/Course_Notes/Differential_Equations_Notes/node5.html.

VI. Appendix

Our github repository can be found here: https://github.com/ucb-ee106-classrooms/project-3-yuvan.

Algorithm	**Summary of Performance**	**Estimation Accuracy (Deviation from Ground Truth)**	**Per-Step Computational Time (s)**
Dead Reckoning (TurtleBot)	Relies solely on motion integration, accumulates significant drift over time. Poor performance without sensor correction.	$[0.45, 0.25, 0.20, 0.044, 0.027]$	2.7×10^{-5}
Kalman Filter (Turtlebot)	Uses sensor fusion to correct drift from motion integration, significantly improving accuracy.	$[2.16 \times 10^{-8}, 2.79 \times 10^{-2}, 2.85 \times 10^{-2}, 3.55 \times 10^{-2}, 2.55 \times 10^{-1}]$	1.9×10^{-4}
Dead Reckoning (Drone)	Relies solely on motion integration, accumulates significant drift over time. Poor performance without sensor correction.	$[3.10, 0.66, 0.11, 2.38, 0.82, 0.048]$	1.9×10^{-6}
Extended Kalman Filter (Drone)	Handles nonlinearities better, improving accuracy over standard KF.	$[0.11, 0.15, 0.016, 0.05, 0.05, 0.05]$	4×10^{-5}

TABLE I

Comparison of different state estimation algorithms. The Kalman filter outperforms Dead Reckoning for the Turtlebot, achieving much lower errors at the cost of higher computation time. Similarly, the Extended Kalman Filter outperforms Dead Reckoning for the Quadrotor drone, albeit with a slower running time.