Report on Gyro Sensor Drift Compensation by Kalman Filter to Control a Mobile Inverted Pendulum Robot System

Hyung-Jik Lee; Seul Jung,
Department of Mechatronics Engineering, Chungnam National University, Deajoen, Korea

REPORT BY : MUNYARADZI P ANTONY (UID: 120482731)
AND VARAD NERLEKAR (UID: 120501135)

***Abstract*—The project focuses on designing and implementing an optimized sensor fusion system that integrates complementary and Kalman filtering techniques to enhance performance in sensor data processing and fusion tasks. Sensor fusion is critical in modern engineering applications, particularly in robotics, autonomous systems, and IoT devices, as it combines data from multiple sensors to improve accuracy and reliability. This research aims to address the limitations of single-filter approaches by leveraging the strengths of both complementary and Kalman filters. Specifically, it seeks to quantify performance improvements in both static and dynamic scenarios, providing a comprehensive evaluation of the dual-filter system's effectiveness. The project will also focus on developing practical guidelines for optimizing filter parameters to ensure robust performance across diverse use cases.**

Key research questions guide this investigation: How does the dual-filter approach affect drift compensation compared to traditional single-filter solutions? What are the optimal filter parameters for different operational scenarios, and how can these be determined systematically? Additionally, the research seeks to understand the system's performance under various environmental disturbances, such as noise, vibration, and changing conditions. By simulating a wide range of operational scenarios, the project will validate the dual-filter system's performance, ensuring its adaptability and reliability.

This work has significant implications for the development of advanced sensor fusion systems. It aims to provide actionable insights into the design and tuning of dual-filter solutions, enabling practitioners to achieve superior performance in real-world applications. The findings from this research are expected to contribute to the broader field of sensor data fusion, offering a robust framework for future advancements. Through rigorous validation and simulation, the project will establish a strong foundation for the practical application of optimized sensor fusion systems in diverse engineering domains.

I. INTRODUCTION

Balancing a mobile inverted pendulum robot requires accurate angle estimation, a challenging task due to sensor limitations. This study addresses the gyro sensor's drift problem by combining complementary and Kalman filters, providing a cost-effective and precise solution. Gyro sensors, though fast, suffer from drift caused by accumulated integration errors, while tilt sensors are accurate at steady states but prone to high-frequency noise. To merge their strengths, a complementary filter was designed: a high-pass filter for the gyro sensor and a low-pass filter for the tilt sensor. This provided a preliminary fused signal compensating for each sensor's weaknesses. To refine the angle estimation further, a Kalman filter was implemented, utilizing a state-space model that accounted for process and measurement noise.

Experimental tests involved balancing and trajectory-tracking tasks. The complementary filter successfully reduced noise and drift, while the Kalman filter enhanced accuracy by dynamically correcting errors. Results demonstrated that the robot achieved stable balancing and precise trajectory tracking, even during sinusoidal motion. This study highlights the synergy of complementary and Kalman filters for robust angle estimation, enabling effective control of mobile inverted pendulum robots. The approach offers practical implications for robotics applications requiring precise and reliable state estimation.

II. METHODOLOGY

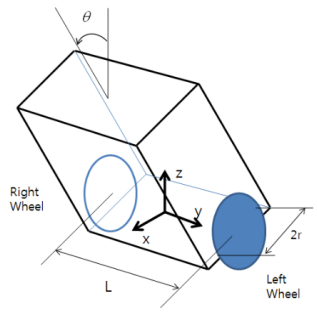

Fig. 1. Mobile pendulum robot system

A. Kinematics Of Mobile Inverted Pendulum Robot

The velocity v_m and the angular velocity ω_m of the robot are defined as:

$$v_m = \frac{v_R + v_L}{2} \tag{1}$$

This is the average of the velocities of the right and left wheels. The robot's linear velocity is simply the mean of the two wheel velocities, v_R and v_L.

$$\omega_m = \frac{v_R - v_L}{L} \tag{2}$$

The angular velocity ω_m is the difference between the velocities of the right and left wheels divided by the wheelbase L. This determines how much the robot rotates.

The angular velocity is also the time derivative of the orientation angle:

$$\omega_m = \dot{\phi}_m \tag{3}$$

ω_m is the rate of change of the orientation angle ϕ_m, where $\dot{\phi}_m$ represents the derivative of the angle with respect to time.

The velocities in the x- and y-directions are given by:

$$\dot{x} = v_m \cos(\phi) \tag{4}$$

The velocity in the x-direction depends on the linear velocity v_m and the orientation angle ϕ.

$$\dot{y} = v_m \sin(\phi) \tag{5}$$

Similarly, the velocity in the y-direction is determined by the same linear velocity v_m and the orientation angle ϕ.

Combining equations (3), (4), and (5), we obtain the following matrix form for the system:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos(\phi) & 0 \\ \sin(\phi) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_m \\ \omega_m \end{bmatrix} \tag{6}$$

This equation combines the linear and angular velocities into a single matrix form. It expresses the rate of change of the robot's position (x, y) and orientation (ϕ) in terms of the robot's linear velocity v_m and angular velocity ω_m.

To relate the velocities of the robot to the angular velocities of the wheels, we use the relationships $v_R = \omega_R r$ and $v_L = \omega_L r$, where r is the radius of the wheels:

$$\begin{bmatrix} v_m \\ \omega_m \end{bmatrix} = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{7}$$

The wheel velocities v_R and v_L are related to the wheel angular velocities ω_R and ω_L via the wheel radius r. This equation expresses the robot's linear and angular velocities as a function of the wheel angular velocities.

Combining equations (6) and (7), we get the kinematic equation of the mobile inverted pendulum system:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \frac{r}{2}\cos(\phi) & \frac{r}{2}\cos(\phi) \\ \frac{r}{2}\sin(\phi) & \frac{r}{2}\sin(\phi) \\ \frac{r}{L} & -\frac{r}{L} \end{bmatrix} \begin{bmatrix} \omega_R \\ \omega_L \end{bmatrix} \tag{8}$$

This is the final kinematic equation of the system. It describes how the robot's position and orientation change over time based on the wheel velocities ω_R and ω_L.

III. Control Scheme

A. Composite Control Law Derivation

The objective of the control system is to stabilize the pendulum while simultaneously driving the robot to a desired position and orientation. Specifically, the goal is to minimize the error between the actual and desired states of the system. We aim to control the following variables:

- Pendulum angle θ and angular velocity $\dot{\theta}$,
- Robot position (x_m, y_m) and velocity v_m,
- Robot orientation ϕ_m and angular velocity ω_m.

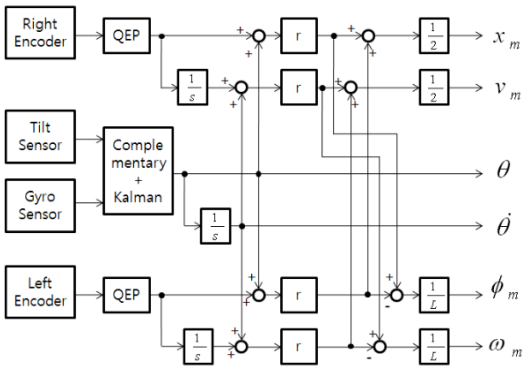

Fig. 2. Variables for control

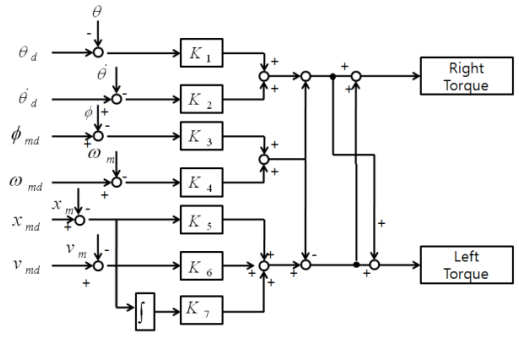

Fig. 3. Controller block diagram

B. Define the Error Variables

We first define the error terms between the desired and actual states of the robot and pendulum:

- Pendulum angle error: $\epsilon_\theta = \theta_d - \theta$,
- Pendulum angular velocity error: $\epsilon_{\dot{\theta}} = \dot{\theta}_d - \dot{\theta}$,
- Robot position error in the x-direction: $\epsilon_x = x_{md} - x_m$,
- Robot position error in the y-direction: $\epsilon_y = y_{md} - y_m$,
- Robot velocity error: $\epsilon_v = v_{md} - v_m$,
- Robot orientation error: $\epsilon_\phi = \phi_{md} - \phi_m$,
- Robot angular velocity error: $\epsilon_\omega = \omega_{md} - \omega_m$.

C. Control Law Design Using PID Control

We will use a **PID control law** to correct the errors in each of the states. A general PID controller for a variable $e(t)$ (in our case, any of the state errors) is of the form:

$$u(t) = K_p e(t) + K_d \dot{e}(t) + K_i \int_0^t e(\tau)\, d\tau \tag{9}$$

where:

- K_p is the proportional gain,
- K_d is the derivative gain,
- K_i is the integral gain.

D. Define the Control Law for the Right Wheel

The control law for the right wheel torque τ_R is derived by applying the PID control law to each of the error terms.

1) Pendulum Angle Error (ϵ_θ): To keep the pendulum upright, we apply a torque proportional to the **pendulum angle error**. This is a **proportional control** term:

$$K_1\epsilon_\theta = K_1(\theta_d - \theta). \tag{10}$$

2) Pendulum Angular Velocity Error ($\epsilon_{\dot\theta}$): To prevent the pendulum from oscillating too much or moving too quickly, we apply **derivative control** to the angular velocity error:

$$K_2\epsilon_{\dot\theta} = K_2(\dot\theta_d - \dot\theta). \tag{11}$$

3) Robot Orientation Error (ϵ_ϕ): To keep the robot's body upright, we apply a torque proportional to the **robot's orientation error**. This is another **proportional control** term:

$$K_3\epsilon_\phi = K_3(\phi_{md} - \phi_m). \tag{12}$$

4) Robot Angular Velocity Error (ϵ_ω): To prevent fast angular movements (which may lead to instability), we apply **derivative control** to the **angular velocity error**:

$$K_4\epsilon_\omega = K_4(\omega_{md} - \omega_m). \tag{13}$$

5) Robot Position Error (ϵ_x, ϵ_y): To correct for the robot's position, we apply **proportional control** for both the x- and y-position errors:

$$K_5\epsilon_x = K_5(x_{md} - x_m), \quad K_6\epsilon_y = K_6(y_{md} - y_m). \tag{14}$$

6) Robot Velocity Error (ϵ_v): To ensure that the robot moves at the desired speed, we apply **proportional control** to the velocity error:

$$K_7\epsilon_v = K_7(v_{md} - v_m). \tag{15}$$

7) Integral Control for Position: Even with proportional control, the robot may experience steady-state errors due to external disturbances or biases. We use **integral control** to account for accumulated position errors over time, especially in the x-direction:

$$K_8 \int_0^t \epsilon_x \, dt. \tag{16}$$

This term corrects for any persistent drift in position.

Combine the Terms for the Right Wheel Torque

Now that we have derived the individual terms for each error, we combine them to form the total control law for the right wheel torque τ_R. This is a combination of the proportional, derivative, and integral terms for each state:

$$\begin{aligned}\tau_R = {} & K_1(\theta_d - \theta) + K_2(\dot\theta_d - \dot\theta) + \\ & K_3(\phi_{md} - \phi_m) + K_4(\omega_{md} - \omega_m) + \\ & K_5(x_{md} - x_m) + K_6(v_{md} - v_m) + \\ & K_7 \int_0^t (x_{md} - x_m)\, dt.\end{aligned} \tag{17}$$

E. Step 6: Control Law for the Left Wheel

The control law for the left wheel torque τ_L is very similar to the right wheel torque, except for the sign changes to account for the opposite direction of the left wheel. Specifically, the orientation and angular velocity terms for the left wheel have opposite signs:

$$\begin{aligned}\tau_L = {} & K_1(\theta_d - \theta) + K_2(\dot\theta_d - \dot\theta) - \\ & K_3(\phi_{md} - \phi_m) - K_4(\omega_{md} - \omega_m) + \\ & K_5(x_{md} - x_m) + K_6(v_{md} - v_m) + \\ & K_7 \int_0^t (x_{md} - x_m)\, dt.\end{aligned} \tag{18}$$

The derived control law combines **Proportional**, **Derivative**, and **Integral** (PID) terms for the various state errors. The proportional terms provide immediate corrective action, the derivative terms ensure smooth motion and prevent overshooting, and the integral term eliminates steady-state errors. By carefully tuning the gains $K_1, K_2, \ldots, K_8$, we can achieve desired robot performance, stabilizing the pendulum while tracking the desired trajectory.

IV. Filter Transfer Functions

The transfer functions for the filter used in the control system are given as:

$$F_r(s) = \frac{1}{\tau s + 1} \tag{19}$$

The filter $F_r(s)$ is a first-order low-pass filter, where τ is the time constant that determines the cutoff frequency.

A. First-Order Low-Pass Filter:

The low-pass filter is designed to allow low-frequency signals to pass through while attenuating high-frequency signals. The transfer function of the low-pass filter is derived from the time-domain differential equation relating the input and output signals.

The general form of a first-order linear time-invariant system in the Laplace domain is:

$$Y(s) = \frac{1}{\tau s + 1} X(s) \tag{20}$$

Where:

- $X(s)$ is the Laplace transform of the input signal.
- $Y(s)$ is the Laplace transform of the output signal.
- τ is the **time constant** that determines the system's cutoff frequency.

To derive this, we start with the standard first-order system equation in the time domain:

$$\tau \frac{d}{dt} y(t) + y(t) = x(t) \tag{21}$$

Taking the Laplace transform of both sides, and assuming zero initial conditions, we get:

$$\tau s Y(s) + Y(s) = X(s) \tag{22}$$

Solving for $Y(s)$, we get:

$$Y(s) = \frac{1}{\tau s + 1} X(s) \tag{23}$$

Thus, the transfer function of the first-order low-pass filter is:

$$F_r(s) = \frac{1}{\tau s + 1} \tag{24}$$

This filter will **attenuate high-frequency components** (with frequencies higher than the cutoff $\omega_c = \frac{1}{\tau}$) and pass low-frequency components through without significant distortion.

B. First-Order High-Pass Filter:

The second filter is a **high-pass filter**, represented by:

$$F_g(s) = \frac{\tau s}{\tau s + 1} \tag{25}$$

A **high-pass filter** allows high-frequency components to pass through and attenuates low-frequency components. The transfer function for a first-order high-pass filter can be derived from the time-domain differential equation for the high-pass system.

The general form of the first-order high-pass filter in the Laplace domain is:

$$Y(s) = \frac{\tau s}{\tau s + 1} X(s) \tag{26}$$

To derive this, we start with the standard time-domain equation for a high-pass filter:

$$\tau \frac{d}{dt} y(t) + y(t) = \frac{d}{dt} x(t) \tag{27}$$

Taking the Laplace transform of both sides:

$$\tau s Y(s) + Y(s) = s X(s) \tag{28}$$

Solving for $Y(s)$, we get:

$$Y(s) = \frac{s}{\tau s + 1} X(s) \tag{29}$$

Thus, the transfer function of the first-order high-pass filter is:

$$F_g(s) = \frac{\tau s}{\tau s + 1} \tag{30}$$

This filter **attenuates low-frequency components** (with frequencies below the cutoff $\omega_c = \frac{1}{\tau}$) and **passes high-frequency components**.

C. Combining the Filters

To combine the filters with the sensor characteristics, we use the equation:

$$H_t(s) F_r(s) + H_g(s) F_g(s) = 1 \tag{31}$$

This equation represents the combined effect of the sensor filters and the control system filters. If the sensor characteristics are unknown, we assume $H_t(s) = 1$ and $H_g(s) = 1$, leading to:

$$F_r(s) + F_g(s) = 1 \tag{32}$$

This ensures that the overall system maintains the required dynamics without additional attenuation or amplification in the frequency range of interest. The combination of these filters allows the system to pass desired frequencies and attenuate unwanted noise.

V. Kalman Filter

The Kalman filter is a recursive estimator that uses measurements over time to estimate the state of a dynamic system. The Kalman filter assumes that both the system and the measurements are subject to noise. The objective of the Kalman filter is to combine the model of the system with the noisy measurements to provide an optimal estimate of the system's state.

VI. Kalman Filter Derivations

A. State Equation Derivation

The system evolves in a linear fashion from one time step to the next. At time step k, the state of the system is described by the vector x_k. The evolution of the system is subject to a process noise w_k, which accounts for uncertainties in the model (e.g., unmodeled dynamics, external disturbances).

Let's begin with the general form of a state-space model, which describes the dynamics of the system over time:

$$x_{k+1} = f(x_k, u_k, w_k) \tag{33}$$

Where:

- x_k is the state at time k,
- u_k is the control input at time k,
- w_k is the process noise.

Assume the system is linear with respect to the state and control input. This means we can express the next state as a linear combination of the current state and the control input, plus some noise term. Therefore, we can write:

$$x_{k+1} = A x_k + B u_k + w_k \tag{34}$$

Where:

- A is the state transition matrix that describes how the state evolves over time,
- B is the control matrix that represents how the control input u_k influences the state,
- w_k is the process noise that accounts for random disturbances in the system.

In practice, the control input u_k may be absent (i.e., $B = 0$), in which case the equation simplifies to:

$$x_{k+1} = Ax_k + w_k \tag{35}$$

This is the **state equation**. The noise term w_k is typically assumed to be zero-mean Gaussian noise with covariance Q, i.e.,

$$w_k \sim \mathcal{N}(0, Q) \tag{36}$$

Thus, the state equation accounts for both the deterministic evolution of the system (through Ax_k) and the random disturbances (through w_k).

B. Predicted State Equation Derivation

The predicted state $\hat{x}_k^-$ at time k is computed from the previous predicted state $\hat{x}_{k-1}^-$ and the control input u_{k-1}.

To predict the state at time k, we assume the same linear evolution model as in the state equation. The predicted state $\hat{x}_k^-$ is based on the previous state $\hat{x}_{k-1}^-$, which is an estimate of the state at time $k-1$. So the prediction equation can be written as:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{37}$$

Where:

- $A\hat{x}_{k-1}^-$ propagates the previous estimate of the state $\hat{x}_{k-1}^-$ forward in time using the system dynamics,
- Bu_{k-1} accounts for any control input u_{k-1} that might influence the state transition.

This prediction step assumes that, given the state at time $k-1$, the state at time k will evolve in a deterministic way, according to the matrix A, and may be influenced by the control input u_{k-1} through the matrix B.

If there is no control input, the equation simplifies to:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- \tag{38}$$

But if the control input is present, Bu_{k-1} accounts for the influence of the control on the state.

Thus, the predicted state equation is:

$$\hat{x}_k^- = A\hat{x}_{k-1}^- + Bu_{k-1} \tag{39}$$

Where:

- $\hat{x}_{k-1}^-$ is the predicted state at time $k-1$,
- A is the state transition matrix,
- B is the control matrix,
- u_{k-1} is the control input at time $k-1$.

C. Measurement Equation Derivation

The measurements are related to the system state through the following linear equation:

$$z_k = Hx_k + v_k \tag{40}$$

Where:

- z_k is the measurement at time k,
- H is the measurement matrix, which maps the state vector x_k to the measurement space,
- v_k is the measurement noise, assumed to be zero-mean Gaussian noise with covariance R, i.e., $v_k \sim \mathcal{N}(0, R)$.

The measurement noise v_k represents errors in the measurement process, such as sensor noise or inaccuracies. This equation describes how the true state x_k is mapped to the measurement space via the matrix H, with an added noise term v_k.

D. Prediction of the Error Covariance Derivation

The error covariance P_k represents the uncertainty associated with the state estimate $\hat{x}_k$. It is updated recursively and provides a measure of how much the true state is expected to deviate from the estimated state. To predict the error covariance at time k, we use the following equation:

$$P_k^- = AP_{k-1}^-A^T + Q \tag{41}$$

Where:

- P_k^- is the predicted error covariance at time k,
- P_{k-1}^- is the error covariance at the previous time step $k-1$,
- $AP_{k-1}^-A^T$ is the propagated error covariance, which reflects how uncertainty in the previous state estimate evolves according to the system dynamics,
- Q is the process noise covariance, representing the uncertainty added by the process noise at each time step.

To derive this equation: 1. **State Evolution**: The system state evolves according to the linear system model:

$$x_{k+1} = Ax_k + Bu_k + w_k \tag{42}$$

Where:

- A is the state transition matrix,
- Bu_k is the control input,
- $w_k \sim \mathcal{N}(0, Q)$ is the process noise.

2. **Error Covariance Propagation**: The error covariance P_k at time k is the expected value of the error in the state estimate:

$$P_k = E\left[(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right] \tag{43}$$

To propagate the error covariance forward in time, we use the state transition matrix A. The predicted error covariance at time k is:

$$P_k^- = AP_{k-1}^-A^T + Q \tag{44}$$

Where:

- P_{k-1}^- is the error covariance at time $k-1$,
- $AP_{k-1}^-A^T$ propagates the error covariance according to the system dynamics,
- Q is the process noise covariance, representing the uncertainty added by the system noise.

This equation describes how the error covariance evolves over time, taking into account the system dynamics (through A) and the process noise (through Q).

Derivation of the Kalman Gain

We are trying to compute the Kalman gain K_k, which balances the predicted state estimate and the measurement update in a way that minimizes the uncertainty in the updated state estimate.

State Update Formula

Given the predicted state estimate $\hat{x}_k^-$, the updated state estimate $\hat{x}_k$ is computed as:

$$\hat{x}_k = \hat{x}_k^- + K_k \left(z_k - H\hat{x}_k^-\right)$$

Where:

- $\hat{x}_k^-$ is the predicted state estimate,
- K_k is the Kalman gain,
- z_k is the measurement at time k,
- $H\hat{x}_k^-$ is the predicted measurement,
- $\left(z_k - H\hat{x}_k^-\right)$ is the innovation (or residual), the difference between the measurement and the predicted measurement.

Innovation Covariance

The term $z_k - H\hat{x}_k^-$ is the innovation (or residual). Its covariance represents the uncertainty of the innovation and is given by:

$$S_k = E\left[\left(z_k - H\hat{x}_k^-\right)\left(z_k - H\hat{x}_k^-\right)^T\right] \tag{45}$$

Using the measurement model $z_k = Hx_k + v_k$, where v_k is the measurement noise (assumed to be zero-mean Gaussian noise with covariance R), we can write:

$$S_k = E\left[\left(Hx_k + v_k - H\hat{x}_k^-\right)\left(Hx_k + v_k - H\hat{x}_k^-\right)^T\right] \tag{46}$$

Since $\hat{x}_k^-$ is the predicted state, the error term $x_k - \hat{x}_k^-$ is the true error in the prediction. Expanding this expression:

$$S_k = HE\left[\left(x_k - \hat{x}_k^-\right)\left(x_k - \hat{x}_k^-\right)^T\right]H^T + E\left[v_k v_k^T\right] \tag{47}$$

The term $E\left[\left(x_k - \hat{x}_k^-\right)\left(x_k - \hat{x}_k^-\right)^T\right]$ is the predicted error covariance, denoted as P_k^-, and the term $E\left[v_k v_k^T\right]$ is the measurement noise covariance R. Thus, the innovation covariance is:

$$S_k = HP_k^-H^T + R \tag{48}$$

State Covariance Update

The error covariance of the updated state estimate P_k is given by:

$$P_k = E\left[\left(x_k - \hat{x}_k\right)\left(x_k - \hat{x}_k\right)^T\right] \tag{49}$$

Substituting $\hat{x}_k = \hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right)$ into this expression for P_k:

$$P_k = E\left[\left(x_k - \left(\hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right)\right)\right) \times \left(x_k - \left(\hat{x}_k^- + K_k\left(z_k - H\hat{x}_k^-\right)\right)\right)^T\right] \tag{50}$$

Now expand this expression:

$$P_k = P_k^- - K_k E\left[\left(z_k - H\hat{x}_k^-\right)\left(z_k - H\hat{x}_k^-\right)^T\right]K_k^T \tag{51}$$

Using the innovation covariance S_k, we get:

$$P_k = P_k^- - K_k S_k K_k^T \tag{52}$$

Optimal Kalman Gain

To minimize the error covariance P_k, we need to find the optimal K_k. The optimal K_k is the one that minimizes the trace of P_k, or equivalently minimizes the error covariance.

The expression we need to minimize is:

$$\text{tr}\left(P_k\right) = \text{tr}\left(P_k^-\right) - \text{tr}\left(K_k S_k K_k^T\right) \tag{53}$$

Since the trace is invariant to cyclic permutations, we can write:

$$\text{tr}\left(P_k\right) = \text{tr}\left(P_k^-\right) - \text{tr}\left(S_k K_k^T K_k\right) \tag{54}$$

The term $\text{tr}\left(S_k K_k^T K_k\right)$ is a quadratic form in K_k. To minimize this, we take the derivative of the trace with respect to K_k and set it equal to zero. This gives:

$$S_k K_k^T + K_k S_k = P_k^- K_k^T \tag{55}$$

After solving for K_k, we obtain the final equation for the Kalman gain:

$$K_k = P_k^- H^T\left(HP_k^-H^T + R\right)^{-1} \tag{56}$$

This is the optimal Kalman gain formula. It provides a way to optimally combine the predicted state and the measurement to minimize the uncertainty in the updated state estimate.

Intuition

The Kalman gain K_k balances the predicted state estimate $\hat{x}_k^-$ and the new measurement z_k.

- If the predicted error covariance P_k^- is large, meaning the prediction is uncertain, the Kalman filter gives more weight to the measurement.
- If the measurement noise R is large, meaning the measurement is uncertain, the Kalman filter gives more weight to the prediction.

Thus, K_k optimally combines the predicted state and the measurement in such a way that the updated state estimate has the minimum possible uncertainty.

Final Formula

The final formula for the Kalman gain is:

$$K_k = P_k^- H^T\left(HP_k^-H^T + R\right)^{-1} \tag{57}$$

VII. Experiment

In this experiment, we designed and built a robot using the **Raspberry Pi 4B** as the main processing unit, an **MPU6050** for motion sensing, and **TT motors** controlled by an **L298N motor driver**. The goal of the experiment was to estimate the robot's orientation (angle) in real-time using sensor data from the MPU6050, and to apply different filtering techniques (including a Kalman filter) to improve the angle estimation accuracy.

A. Hardware Setup

The robot's hardware components consisted of:

- **Raspberry Pi 4B**: The main computing unit, running the control software, interfacing with the sensors and motors.
- **MPU6050**: A six-axis sensor providing both *gyroscope* and *accelerometer* data. The gyroscope measures the angular velocity (in degrees per second), while the accelerometer measures the linear acceleration and can be used to estimate the tilt angle based on gravity.
- **TT motors**: Two DC motors used for controlling the movement of the robot.
- **L298N motor driver**: A motor driver board used to interface the Raspberry Pi with the TT motors, providing the necessary current to control the motors' speed and direction.

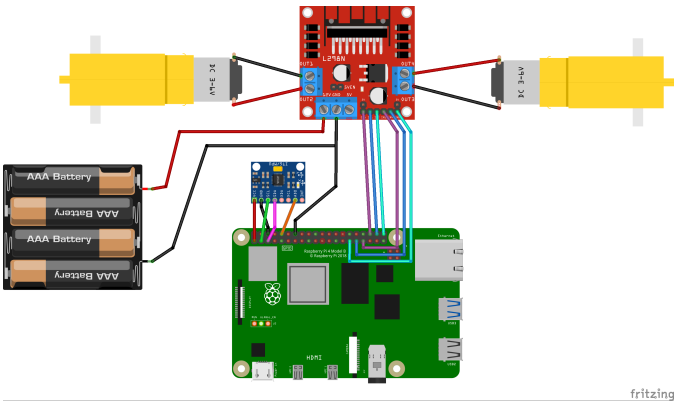

Fig. 4. Circuit Diagram

B. Software Implementation

The software was designed to read data from the MPU6050 sensor and process it in real-time to estimate the robot's orientation. We used Python for the implementation, leveraging libraries like `smbus2` to interface with the MPU6050, and custom code to process the sensor data. The following processing steps were applied:

1) Sensor Data Acquisition: The MPU6050 sensor was configured to provide both gyroscope and accelerometer data. The accelerometer readings were used to calculate the angle based on the pitch and roll, while the gyroscope readings were used to measure the rate of change of the angle over time.

The data acquisition process involved:

Fig. 5. Robot Setup

- **Gyroscope Data**: Provides angular velocity in three axes (X, Y, Z).
- **Accelerometer Data**: Provides acceleration in three axes (X, Y, Z), which can be used to infer the robot's tilt angle relative to the ground (via pitch and roll).

2) Low Pass and High Pass Filters: To reduce noise and improve the quality of the sensor data, we applied a combination of **low-pass** and **high-pass** filters:

- **Low-Pass Filter**: Used to remove high-frequency noise from the accelerometer and gyroscope data, which was critical for ensuring stable angle measurements.
- **High-Pass Filter**: Applied to the accelerometer data to remove the low-frequency drift caused by the gravity component in the accelerometer measurements.

These filters allowed us to separate the noise from the actual signal and obtain cleaner data for further processing.

3) Complementary Filter for Angle Estimation: We first implemented a **complementary filter** to combine the accelerometer and gyroscope data for angle estimation. The complementary filter is a simple, efficient way of fusing data from both sensors to obtain an accurate estimate of the robot's orientation:

$$\theta_{\text{combined}} = \alpha(\theta_{\text{gyro}}) + (1 - \alpha)(\theta_{\text{accel}}) \tag{58}$$

Where:

- θ_{gyro} is the angle calculated from the gyroscope data by integrating the angular velocity over time.
- θ_{accel} is the angle calculated from the accelerometer data using the arctangent of the acceleration components.
- α is a constant that determines the relative weight of the gyroscope and accelerometer in the final estimate (typically set between 0.98 and 0.99).

While the complementary filter provided a reasonable angle estimate, it was **shaky** and exhibited **slight deviations** over time, likely due to the inherent noise in the sensor readings and the limitations of the filter itself.

4) Kalman Filter for Improved Angle Estimation: To improve the angle estimation and reduce the shakiness and drift, we applied a **Kalman filter**. The Kalman filter is an optimal estimator that minimizes the uncertainty in the state estimate by taking into account both the sensor measurements and the model of the system dynamics.

We modeled the robot's angle using the following assumptions:

- The robot's motion is approximated by a simple constant velocity model, where the angle at the next time step is predicted based on the current angle and angular velocity (from the gyroscope).
- The measurement update uses the accelerometer data to correct the predicted angle, with the Kalman gain balancing the trust between the prediction (from the gyroscope) and the measurement (from the accelerometer).

The Kalman filter equations are as follows:

- **Prediction Step**:
  - Predict the next state (angle and angular velocity) based on the previous state and gyroscope data.
  - Predict the error covariance (uncertainty in the prediction).
- **Update Step**:
  - Correct the predicted angle using the accelerometer measurement, with the Kalman gain determining how much weight should be given to the new measurement versus the predicted value.
  - Update the error covariance based on the Kalman gain.

This filter provided much more accurate and stable angle estimates compared to the complementary filter, significantly reducing the shakiness and deviation in the orientation over time.

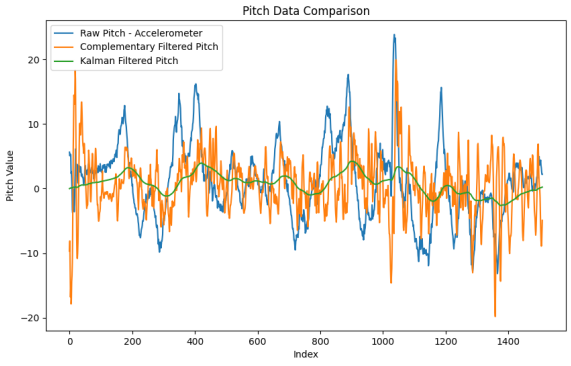

Fig. 6. Comparison of Complementary Filter and Kalman Filter results

VIII. Results

The implementation of the Kalman filter significantly enhanced the accuracy and stability of the robot's angle estimation when compared to the complementary filter. The Kalman filter was particularly effective in addressing the noise and drift that are common challenges in sensor-based angle estimation, leading to smoother and more reliable outputs.

We observed the following key improvements:

- **More Consistent and Accurate Angle Estimates:** The Kalman filter demonstrated a marked improvement in the consistency and accuracy of the angle estimates. While the complementary filter struggled with fluctuations and inaccuracies due to sensor noise, the Kalman filter was able to effectively combine the accelerometer and gyroscope data in a way that minimized these errors. The result was a much more precise estimation of the robot's orientation, even in the presence of noisy or unreliable sensor readings. This improvement is particularly important in dynamic environments, where rapid changes in orientation occur frequently.
- **Reduced Shakiness in the Output:** One of the most noticeable improvements with the Kalman filter was the reduction in the "shaky" behavior of the angle estimates. The complementary filter, though effective to a degree, still exhibited visible oscillations and inconsistencies, especially when the robot was at rest or experiencing low-frequency movement. These fluctuations, which are typically a result of the accelerometer's sensitivity to noise and small changes in tilt, were significantly smoothed out by the Kalman filter. This reduction in shakiness resulted in a much more stable and natural response from the robot, making its movements appear more fluid and predictable.
- **Correction of Deviations in Angle Estimation:** The complementary filter, while useful, was prone to slight deviations from the true angle over time, especially when the gyroscope experienced drift or when the accelerometer measurements were noisy. These small but cumulative errors were corrected by the Kalman filter, which dynamically adjusted the weighting of the sensor inputs based on their respective uncertainties. As a result, the angle estimates produced by the Kalman filter remained more accurate over extended periods, reducing long-term drift and providing a more reliable and stable state estimation. This correction of deviations was particularly evident in situations where the robot had been operating for a longer period, where the complementary filter's estimates tended to degrade due to the gyroscope's drift.

Overall, the Kalman filter's superior ability to fuse sensor data and account for the inherent noise and drift in the gyroscope and accelerometer led to a significant improvement in the robot's performance. The angle estimates were not only more accurate but also much smoother and stable, enabling the robot to maintain a more precise and consistent orientation, which is critical for tasks requiring high levels of control and stability.

IX. Conclusion

This experiment demonstrated the effective application of filtering techniques for improving sensor-based angle estimation in a robot, specifically using data from an accelerometer and gyroscope. The complementary filter and Kalman filter were applied to fuse these sensor readings and achieve more accurate and stable estimates of the robot's orientation.

The accelerometer, while providing reliable measurements of tilt, is prone to noise and a shaky response, particularly in dynamic environments or at low frequencies. On the other hand, the gyroscope provides smooth and continuous measurements of angular velocity but suffers from drift over time, leading to cumulative errors in angle estimation. By combining both sensor types through filtering techniques, we were able to mitigate the individual weaknesses of each sensor.

Among the two filtering methods, the Kalman filter emerged as the optimal solution. Its ability to dynamically weigh the contributions of each sensor based on their respective uncertainties allowed for a more robust estimate of orientation. The Kalman filter was particularly effective in reducing the noise and drift in the gyroscope data, as well as the shakiness and inconsistencies inherent in the accelerometer measurements. As a result, the angle estimates were significantly more accurate, stable, and reliable, making the Kalman filter an invaluable tool for sensor fusion in robotics.

In conclusion, the integration of filtering techniques, especially the Kalman filter, proves to be a powerful approach for addressing the limitations of individual sensors and improving the performance of angle estimation in robotic systems.

ACKNOWLEDGMENT

We would like to extend our deep gratitude to **Dr. Waseem Malik**, whose guidance and support have been invaluable throughout the course. We would also like to express our sincere appreciation to our grader, **Fazil Mammadli**, for their prompt feedback, constructive comments, and support throughout the course.

Their efforts have been essential in refining our understanding and ensuring the successful completion of our work.

X. REFERENCES

REFERENCES

1) Lee, Hyung-Jik, and Seul Jung. “Gyro sensor drift compensation by Kalman filter to control a mobile inverted pendulum robot system.” *2009 IEEE International Conference on Industrial Technology*, Feb. 2009, https://doi.org/10.1109/icit.2009.4939502.
2) Negenborn, Rudy. “Robot Localization and Kalman Filters: On Finding Your Position in a Noisy World.” Master’s thesis, Institute of Information and Computing Sciences, Utrecht University, 2003. http://www.negenborn.net/kalman/.
3) Larsen, Thomas Dall, Karsten Lentfer Hansen, Nils A. Andersen, and Ole Ravn. “Design of Kalman Filters for Mobile Robots: Evaluation of the Kinematic and Odometric Approach.” Proceedings of the Department of Automation, Technical University of Denmark, 1998. Email: tdl@iau.dtu.dk, Fax: +45 45 88 12 95.