Compact EKF Formulation for RoboMaster S1 State Estimation (8-DOF)

Project Guide

1 State, System, and Measurement Models

1.1 State Vector

The state vector is designed for planar (2D) ground robot motion. It includes 2D position, 2D velocity, yaw angle, and biases for the accelerometer and gyroscope, resulting in an 8-DOF state.

$$\mathbf{x}_k = \begin{bmatrix} p_x \\ p_y \\ \theta \\ v_x \\ v_y \\ b_{ax} \\ b_{ay} \\ b_\omega \end{bmatrix} \in \mathbb{R}^8$$

where:

- (p_x, p_y): Position in the world frame.
- θ: Yaw (heading) angle.
- (v_x, v_y): Velocity in the world frame.
- (b_{ax}, b_{ay}): Accelerometer biases in the body frame.
- b_ω: Gyro (yaw-rate) bias in the body frame.

1.2 Process Model

The process model describes the rover's kinematics, driven by IMU inputs. The state propagates forward based on the previous state and the current IMU readings.

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{q}_{k-1}, \quad \mathbf{u}_{k-1} = \begin{bmatrix} \mathbf{a}_{\text{meas}} \\ \omega_{\text{meas}} \end{bmatrix}_{k-1} \tag{1}$$

where $\mathbf{a}_{\text{meas}} = [a_x, a_y]^\top$ is the 2D accelerometer reading and ω_{meas} is the gyroscope yaw-rate reading. The nonlinear state transition function f is defined by integrating the continuous-time dynamics over the interval Δt:

$$f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} p_x + v_x \Delta t \\ p_y + v_y \Delta t \\ \theta + (\omega_{\text{meas}} - b_\omega)\Delta t \\ v_x + \big((a_{x,\text{meas}} - b_{ax})\cos\theta - (a_{y,\text{meas}} - b_{ay})\sin\theta\big)\Delta t \\ v_y + \big((a_{x,\text{meas}} - b_{ax})\sin\theta + (a_{y,\text{meas}} - b_{ay})\cos\theta\big)\Delta t \\ b_{ax} \\ b_{ay} \\ b_\omega \end{bmatrix}$$

The process noise $\mathbf{q}_{k-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ models the uncertainty in the dynamics, particularly how the biases drift over time. It is typically defined as a diagonal matrix:

$$\mathbf{Q} = \text{diag}(\sigma_{p_x}^2, \sigma_{p_y}^2, \sigma_\theta^2, \sigma_{v_x}^2, \sigma_{v_y}^2, \sigma_{b_{ax}}^2, \sigma_{b_{ay}}^2, \sigma_{b\omega}^2) \tag{2}$$

1.3 Measurement Models

We have two primary sensor models for the update step: wheel encoders and magnetometer.

Wheel Odometry

The wheel encoders provide a measurement of the rover's body-frame velocity, $\mathbf{v}^{\text{body}}$. We use this to correct our world-frame velocity estimate. The measurement model is non-linear due to the rotation from world to body frame.

$$\mathbf{y}_k^{\text{odom}} = h_{\text{odom}}(\mathbf{x}_k) + \mathbf{r}_k^{\text{odom}} = \mathbf{R}(\theta_k)^\top \begin{bmatrix} v_x \\ v_y \end{bmatrix}_k + \mathbf{r}_k^{\text{odom}} \tag{3}$$

with the 2D rotation matrix $\mathbf{R}(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$. The measurement noise is $\mathbf{r}_k^{\text{odom}} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}^{\text{odom}})$.

Magnetometer

The magnetometer provides a direct, but noisy, measurement of the yaw angle (heading).

$$y_k^{\text{mag}} = h_{\text{mag}}(\mathbf{x}_k) + r_k^{\text{mag}} = \theta_k + r_k^{\text{mag}} \tag{4}$$

The measurement model is linear. The measurement noise is $r_k^{\text{mag}} \sim \mathcal{N}(0, R^{\text{mag}})$.

2 EKF Recursion

2.1 Prediction (IMU-driven at each time step)

The prediction computes a priori estimates based on the previous posterior estimate $\mathbf{m}_{k-1}$ and the IMU inputs $\mathbf{u}_{k-1}$.

$$\mathbf{m}_k^- = f(\mathbf{m}_{k-1}, \mathbf{u}_{k-1}) \tag{5}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}\mathbf{F}_{k-1}^\top + \mathbf{Q} \tag{6}$$

The process Jacobian $\mathbf{F}_{k-1} = \frac{\partial f}{\partial \mathbf{x}}\big|_{\mathbf{m}_{k-1}, \mathbf{u}_{k-1}}$ is an 8×8 matrix. Let $c\theta = \cos\theta$, $s\theta = \sin\theta$, $a'_x = a_{x,\text{meas}} - b_{ax}$, and $a'_y = a_{y,\text{meas}} - b_{ay}$. The non-zero elements are:

$$\mathbf{F}_{k-1} = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -\Delta t \\ 0 & 0 & (-a'_x s\theta - a'_y c\theta)\Delta t & 1 & 0 & -c\theta\Delta t & s\theta\Delta t & 0 \\ 0 & 0 & (a'_x c\theta - a'_y s\theta)\Delta t & 0 & 1 & -s\theta\Delta t & -c\theta\Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

2.2 Update (when external sensor data arrives)

When a measurement is available, we correct the predicted state.

$$\mathbf{v}_k = \mathbf{y}_k - h(\mathbf{m}_k^-) \tag{7}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^\top + \mathbf{R}_k \tag{8}$$

$$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^\top \mathbf{S}_k^{-1} \tag{9}$$

$$\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k \mathbf{v}_k \tag{10}$$

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k)\mathbf{P}_k^- \tag{11}$$

The specific measurement matrix $\mathbf{H}_k$ and noise covariance $\mathbf{R}_k$ depend on the sensor.

For Wheel Odometry Update:

The Jacobian $\mathbf{H}_k^{\text{odom}} = \frac{\partial h_{\text{odom}}}{\partial \mathbf{x}}\big|_{\mathbf{m}_k^-}$ is a 2×8 matrix. Let v_x, v_y, θ be the components of $\mathbf{m}_k^-$.

$$\mathbf{H}_k^{\text{odom}} = \begin{bmatrix} 0 & 0 & -v_x s\theta + v_y c\theta & c\theta & s\theta & 0 & 0 & 0 \\ 0 & 0 & -v_x c\theta - v_y s\theta & -s\theta & c\theta & 0 & 0 & 0 \end{bmatrix}$$

For Magnetometer Update:

The Jacobian is a constant 1×8 matrix:

$$\mathbf{H}_k^{\text{mag}} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3 Real-Time Implementation Logic

For a successful real-time sensor fusion operation, we must handle multiple sensors that arrive at different rates. The RoboMaster S1's IMU provides data at a high frequency, while wheel encoders and magnetometers provide updates at a potentially different (though also high) frequency.

3.1 Main Filter Loop

The implementation is built around a main loop that runs at a fixed, high frequency, driven by the IMU data stream.

Step 1: Initialization

- **Initialize State Mean ($\mathbf{m}_0$):** Start with the best available information. Typically, at rest on a flat surface:

$$\mathbf{m}_0 = [0, 0, \theta_{\text{init}}, 0, 0, b_{ax,\text{init}}, b_{ay,\text{init}}, b_{\omega,\text{init}}]^\top$$

  where initial yaw θ_{init} might come from a magnetometer or be set to zero, and initial biases can be set to zero or pre-calibrated values.

- **Initialize State Covariance ($\mathbf{P}_0$):** Set diagonal elements to represent the initial uncertainty. Position and velocity uncertainty might be small if starting from a known point, while bias uncertainty should be non-zero.

- **Define Noise Covariances ($\mathbf{Q}, \mathbf{R}$):** These are crucial tuning parameters.
  - $\mathbf{Q}$: The process noise covariance, representing the uncertainty of the motion model. The diagonal elements corresponding to the biases control how quickly the filter allows the bias estimates to change.
  - $\mathbf{R}_{\text{odom}}$ and $\mathbf{R}_{\text{mag}}$: The measurement noise covariances for wheel odometry and the magnetometer, respectively. These values reflect the trustworthiness of each sensor.

Step 2: Start the Real-Time Loop

The loop is asynchronous and event-driven, responding to incoming sensor data.

1. **On Receipt of a new IMU packet ($\mathbf{a}_{\text{meas}}, \omega_{\text{meas}}$):**
   - Calculate the time delta Δt from the previous IMU packet.
   - Perform the EKF Prediction Step using the new IMU data as the input $\mathbf{u}_{k-1}$.

$$(\mathbf{m}_k^-, \mathbf{P}_k^-) = \text{Predict}(\mathbf{m}_{k-1}, \mathbf{P}_{k-1}, \mathbf{u}_{k-1}, \Delta t)$$

     This step happens continuously at the IMU's high frequency, propagating the state estimate forward in time.

2. **On Receipt of a new Wheel Encoder packet ($\mathbf{y}_k^{\text{odom}}$):**
   - Perform an EKF Update Step.
   - Use the predicted state $(\mathbf{m}_k^-, \mathbf{P}_k^-)$ from the latest prediction.

- Use the wheel odometry measurement model: $h = h_{\text{odom}}$, $\mathbf{H}_k = \mathbf{H}_k^{\text{odom}}$, $\mathbf{R}_k = \mathbf{R}^{\text{odom}}$.

$$(\mathbf{m}_k, \mathbf{P}_k) = \text{Update}(\mathbf{m}_k^-, \mathbf{P}_k^-, \mathbf{y}_k^{\text{odom}}, h, \mathbf{H}_k, \mathbf{R}_k)$$

  The result $(\mathbf{m}_k, \mathbf{P}_k)$ becomes the new official state estimate, which will be used in the next prediction step.

3. **On Receipt of a new Magnetometer packet (y_k^{mag}):**

   - Perform another EKF Update Step.
   - Use the most recently calculated state $(\mathbf{m}_k^-, \mathbf{P}_k^-)$ (this could be the output of a prediction or a previous update).
   - Use the magnetometer measurement model: $h = h_{\text{mag}}$, $\mathbf{H}_k = \mathbf{H}_k^{\text{mag}}$, $\mathbf{R}_k = \mathbf{R}^{\text{mag}}$.

$$(\mathbf{m}_k, \mathbf{P}_k) = \text{Update}(\mathbf{m}_k^-, \mathbf{P}_k^-, y_k^{\text{mag}}, h, \mathbf{H}_k, \mathbf{R}_k)$$

Outcome and Efficiency

This structure ensures that the high-frequency IMU data continuously drives the state estimate forward, providing a smooth but potentially drifting trajectory. The lower-frequency, absolute measurements from the wheel encoders (for velocity) and magnetometer (for heading) serve to correct this drift whenever they arrive. This asynchronous fusion is the standard and most efficient way to implement sensor fusion filters for real-time robotics.