Mobile Robotics LAB 03

CoppeliaSimulator and ROS2 and Plotjuggler

Theoretical Background

1. Pose representation: position of a robot according to a frame
   a. World frame: inertial frame, where the robot operates
   b. Robot frame: local reference frame, origin at the robot center

The position is represented in the world frame by x (horizontal pose), y (vertical position), theta the orientation with respect to the global ref frame

2. Kinematic Model: describes relationship between robot global velocity (world) and control inputs (local velocity, in robot frame), robot vel in robot frame is expressed in linear velocity and angular velocity, multiplying by R(theta) to express in world frame, and that is what we call a kinematic model for the rel between control input and global motion

3. Pose calculation using kinematic model
   a. x_dot = linear_vel * cos (theta)
   b. y_dot = linear_vel * sin (theta)
   c. theta_dot = angular_vel
4. Pose measurement: 2 sensors (encoders, inertial measurement units)
   a. Incremental encoders: spin speed of robot wheel/motor, integrating these measures through time => robot trajectory can be estimated in a process called odometry
   b. Inertial Measurement Units: angular velocity and linear acceleration, can estimate robot orientation and velocity => used to calculate the pose
5. Kalman Filter: recursive algo used for state estimation in dynamic sys
   a. Estimates the system state by combining predictive model with noisy measurements, assuming:
      - System dynamics and measurements are linear
      - The process and measurement noise are Gaussian and uncorrelated

   It works in 2 phases
      - Prediction: future state estimation using motion model
      - Correction: prediction refining using sensor measurements

b. Application in Robotics:
   - Localization: estimate robot position and orientation (odometry ..)
   - Sensor fusion: combining info from (IMU, GPS, encoders ..)
   - Tracking: following dynamic objects in robot environment

c. Mathematical formulation: state of system represented by vector `xt`
   - State transition model:

$$\mathbf{x}_t = \mathbf{F_t}\ \mathbf{x}_{t-1} + \mathbf{B}\ \mathbf{u}_t + \mathbf{w}_t,$$

F => state transition matrix

B => control input matrix, map control input to the state

u => the control input

w => follow normal distribution, process noise, 0 mean, and Q covariance

   - Measurement model:

$$\mathbf{z}_t = \mathbf{H}\ \mathbf{x}_t + \mathbf{v}_t,$$

H => the observation matrix, relates state to measurements

v => follow normal distribution, measurement noise, 0 mean, and R covariance

d. Algorithm:
   - Prediction

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t\ \hat{\mathbf{x}}_{t-1} + \mathbf{B}\ \mathbf{u}_t,$$

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t\ \mathbf{P}_{t-1}\ \mathbf{F}_t^T + \mathbf{Q}$$

   - Update (Correction): uses actual sensor measurements, the gain K is computed to weigh the predicted state and measurements optimally

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\ \mathbf{H}^T \left(\mathbf{H}\ \mathbf{P}_{t|t-1}\ \mathbf{H}^T + \mathbf{R}\right)^{-1}$$

State is updated:

$$\mathbf{x}_t = \mathbf{x}_{t|t-1} + \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{H}\ \mathbf{x}_{t|t-1}\right)$$

Uncertainty (covariance) is updated:

$$\mathbf{P}_{t|t} = \left(\mathbf{I} - \mathbf{K}_t\ \mathbf{H}\right)\ \mathbf{P}_{t|t-1}$$

Since Kalman filter assumes linearity, so for nonlinear system extensions like Extended Kalman filter is used

6. Extended Kalman Filter for Pose estimation:
   a. State vector (robot pose)
   b. State transition Model

$$\mathbf{x}_{t|t-1} = f(\mathbf{x}_{t-1}, \mathbf{u}, \Delta t)$$

State transition function:

$$f(\mathbf{x}_{t-1}, \mathbf{u}, \Delta t) = \begin{bmatrix} x_{t-1} + v\ \Delta t\ \cos(\theta_{t-1}) \\ y_{t-1} + v\ \Delta t\ \sin(\theta_{t-1}) \\ \theta_{t-1} + \omega\ \Delta t \end{bmatrix}$$

State transition matrix:

$$\mathbf{F}_t = \frac{\partial f}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial x_t}{\partial x_{t-1}} & \frac{\partial x_t}{\partial y_{t-1}} & \frac{\partial x_t}{\partial \theta_{t-1}} \\ \frac{\partial y_t}{\partial x_{t-1}} & \frac{\partial y_t}{\partial y_{t-1}} & \frac{\partial y_t}{\partial \theta_{t-1}} \\ \frac{\partial \theta_t}{\partial x_{t-1}} & \frac{\partial \theta_t}{\partial y_{t-1}} & \frac{\partial \theta_t}{\partial \theta_{t-1}} \end{bmatrix}$$

=>

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & -v\ \Delta t\ \sin(\theta_{t-1}) \\ 0 & 1 & v\ \Delta t\ \cos(\theta_{t-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

   c. Measurement model:
      zt = h(xt)= H xt + v (measurements directly observe the pose => H= I)

7. Sensor fusion for robust pose estimation: fusethe data of encoders and IMUs, combining the reliability of encoder-based odometry and the orientation robustness of IMUs, the approach:
   - Encoder data => short-term accuracy and linear displacement

        - IMU data => robot orientation and angular velocity, correct odometry drift (wheel slippage)

Odometry Calculation

- Odometry: method used to estimate robot pose over time based on motion data (proprioceptive sensors)
- Algorithm:
    a. Inputs:
        - Left and right angular velocities
        - Time step
        - Radius of wheel
        - Distance between the left and right wheels
    b. Steps:
        - Compute linear and angular velocity

$$v = \frac{r}{2}(\dot{\phi}_L + \dot{\phi}_R), \quad \omega = \frac{r}{l}(\dot{\phi}_R - \dot{\phi}_L).$$

        - Update x, y, theta

$$\begin{aligned} x_{t+\Delta t} &= x_t + \Delta t \cdot v \cos\theta_t, \\ y_{t+\Delta t} &= y_t + \Delta t \cdot v \sin\theta_t, \\ \theta_{t+\Delta t} &= \theta_t + \Delta t \cdot \omega. \end{aligned}$$

    c. Loop, integrate over successive time steps
    d. Calculating error

$$\text{Position\_Error} = \sqrt{(x_{\text{actual}} - x_{\text{odom}})^2 + (y_{\text{actual}} - y_{\text{odom}})^2}$$

$$\text{Orientation\_Error} = \sqrt{(\theta_{\text{actual}} - \theta_{\text{odom}})^2}$$

Pose Estimation Using Extended Kalman Filter

Estimates state of system

- State vector => position and angle
- Control inputs =>linear and angular vel
- Measurements => odometry position

- Algorithm:
    - initialization
        - Define state vec
        - Covariance matrix (initial uncertainties of x, y, theta on diagonal elements)
        - Process noise definition(small expected variations in process)
        - Measurement covariance matrix (noise of odom)
        - State transition matrix = identity
    - Prediction:
    - Time step dt
    - Update state vec

$$\mathbf{x}_{t|t-1} = f(\mathbf{x}_{t-1}, \mathbf{u_t}, \Delta t) = \begin{bmatrix} x_{t-1} + v_t \ \Delta t \ \cos(\theta_{t-1}) \\ y_{t-1} + v_t \ \Delta t \ \sin(\theta_{t-1}) \\ \theta_{t-1} + \omega_t \ \Delta t \end{bmatrix}$$

    - Update transition matrix

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & -v_t \ \Delta t \ \sin(\theta_{t-1}) \\ 0 & 1 & v_t \ \Delta t \ \cos(\theta_{t-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

    - Predict covariance

$$\mathbf{P}_{t|t-1} = \mathbf{F}_t \ \mathbf{P}_{t-1} \ \mathbf{F}_t^T + \mathbf{Q}$$

    - Updating measurements:
        - Kalman gain

$$\mathbf{K}_t = \mathbf{P}_{t|t-1} \ \mathbf{H}^T \left(\mathbf{H} \ \mathbf{P}_{t|t-1} \ \mathbf{H}^T + \mathbf{R}\right)^{-1}$$

- Update state estimate and covariance

$$\mathbf{x}_t = \mathbf{x}_{t|t-1} + \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{H}\ \mathbf{x}_{t|t-1} \right)$$

$$\mathbf{P}_t = (\mathbf{I} - \mathbf{K}_t\ \mathbf{H})\, \mathbf{P}_{t|t-1}$$

-

- PUBLISH RESULTS

After modifying the script, and trying to do the tasks, the robot is not moving, even though the odom pose value changes but the velocity is stuck on null, which makes robot not move, unless it is published manually
I tried to fix this problem but no solution was found