Rheinische Friedrich-Wilhelms-Universität Bonn
Prof. Dr. Maren Bennewitz

Institut für Informatik
Abteilung VI
Humanoid Robots Lab
Adresse:
Friedrich-Hirzebruch-Allee 8
53115 Bonn

Humanoid Robotics

Assignment 9

Due Thursday, July 3rd, before class.

Legged Robots Locomotion:

1. **Robot Kinematics** **(Total: 5 points)**
   In the lecture, we discussed the limitations of Euler angles and how quaternions help overcome them.

   a. What is gimbal lock, and how does it arise in Euler angle parameterization? Can changing the rotation order avoid this singularity? (1 point)

   b. A quaternion is a 4D vector representing rotation. Why does a quaternion need to be normalized to be used for rotation, and what does the quaternion conjugate represent physically? (1 point)

   c. **Rotation with Quaternion** (3 points)
   In this task, you will implement quaternion-based rotation using concepts from the lecture in Python. You will need to:
   - Convert an **axis-angle representation** into a **quaternion**.
   - Implement **quaternion multiplications** to apply rotation.
   - Use quaternion algebra to rotate a 3D vector in four different test cases.
   - **Verify your implementation** by comparing the result of the quaternion-based rotation with the result from a corresponding **rotation matrix** (use `axis_angle_to_rot_matrix()` function) for each test case.

2. **Robot Control** **(Total: 2 points)**
   In class, we discussed the trade-offs between **position control** and **force control**. Below are three robot control scenarios. For each one, answer which control strategy, force or position control, is more suitable? Justify your choice clearly, referring to task properties (e.g., contact, compliance, sensing, dynamics).

   a. Peg in the hole assembly: a 6-DoF robotic manipulator must insert a cylindrical peg into a tight-fitting hole during an automated assembly process. The hole may slightly misalign due to manufacturing tolerances.

   b. Legged robot traversing uneven terrain: a quadruped robot walks over rocky terrain outdoors. The robot must adapt its motion to variable surface contact, like slipping rocks or yielding grass, while maintaining balance.

   c. High-speed pick-and-place in a factory line: a robotic arm repeatedly moves items from one conveyor belt to another with high precision. Timing and accuracy are critical.

3. Base State Estimation with Linear Kalman Filter (Total: 8 points)

In the lecture, we discussed how **nonlinear motion and measurement models** can be used with an **Extended Kalman Filter (EKF)** to jointly estimate the base position, velocity, and orientation of a legged robot. However, EKFs have some drawbacks; therefore, as a simpler alternative, this exercise implements a **linear Kalman Filter (KF)** for estimating **base position and linear velocity** based on the approach in **(Section III)** of the paper below:

Flayols et al., **Experimental evaluation of simple estimators for humanoid robots**, Humanoids 2017.

Task 3.a) (1 point)

1. What are the **drawbacks of using an Extended Kalman Filter (EKF)** for base state estimation in legged robots?

2. Under what **assumptions** can we design a **linear Kalman Filter (KF)** to estimate the robot's base position and linear velocity?

3. In this linear KF setting, what are the **inputs to the filter**? Specify which sensors provide the prediction (process model) input and which measurements are used in the correction (measurement model).

Kalman Filter

In this section we develop a KF class in Python, and we will apply the method to a **quadruped robot**, which means we have **four-foot contact points**.

Goal

Implement and run a linear KF to estimate the robot's **base position and linear velocity** from known:

- IMU acceleration data
- Robot base orientation
- Foot positions relative to the base (calculated from forward kinematics)
- Contact array (whether each foot is in contact with the ground)

Task 3.b) Compute KF Matrices (2 points)

Since the process model is linear time invariant in the Kalman Filter, the corresponding matrix remains constant over time. **Precompute** the system matrix **F** in the constructor of the Kalman Filter class (e.g., `__init__()` method). The measurement matrix **H** is depending on which feet are in contact and it should be computed in the `update_step()` function.

Task 3.c) Implement the Prediction Step (1 point)

Complete the `prediction_step()` function in your Kalman Filter class. Use the motion model described in **Equation (6)** of the paper to integrate the motion model over time.
Note: The IMU acceleration (`imu_acc`) is provided in the base frame.

Task 3.d) Implement the Measurement model in the Update Step (1 point)

Complete the `update_step()` function in your Kalman Filter class.
When a foot is in contact with the ground, it provides a measurement of the base position relative to that foot. Focus on implementing the measurement model described in **Equations (7), (8)** of the paper.

Task 3.e) Apply the Kalman Filter Update (2 points)

In the `update_step()` function, compute the **Kalman gain** and apply the update equations for:

- The **state estimate**
- The **error covariance matrix**

Use the standard Kalman Filter update equations.

Task 3.f) Visualize the Results (1 point)

Plot the **estimated base position** and **velocity** obtained from the Kalman Filter compared to the **ground truth**.

- Use different colors for estimated and ground truth values.
- Ensure to label the axes and add a legend for clarity.

Rheinische Friedrich-Wilhelms-Universität Bonn

Institut für Informatik
Abteilung VI
Humanoid Robots Lab

Additional Material (Kalman Filter)

Kalman filtering [Kalman60] is based on linear dynamical systems discretized in the time domain. We consider the linear discrete dynamic system, where both prediction model F_k and measurement model H_k are corrupted by zero-mean Gaussian process noise $\boldsymbol{w}_k \sim \mathcal{N}(0, \boldsymbol{Q}_k)$, and measurement noise $\boldsymbol{v}_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$ Respectively.

$$\begin{aligned}\boldsymbol{x}_k &= \boldsymbol{F}_k\boldsymbol{x}_{k-1} + \boldsymbol{B}_k\boldsymbol{u}_k + \boldsymbol{w}_k\\ \boldsymbol{y}_k &= \boldsymbol{H}_k\boldsymbol{x}_k + \boldsymbol{v}_k.\end{aligned}$$

The goal of the Kalman Filter is to estimate the mean μ and error covariance matrix P (which specifies the uncertainty of the estimate) of the Gaussian distribution over the state x.

1. Prediction Step

First we calculate the predicted “a priori” state estimate and estimate error covariance matrix

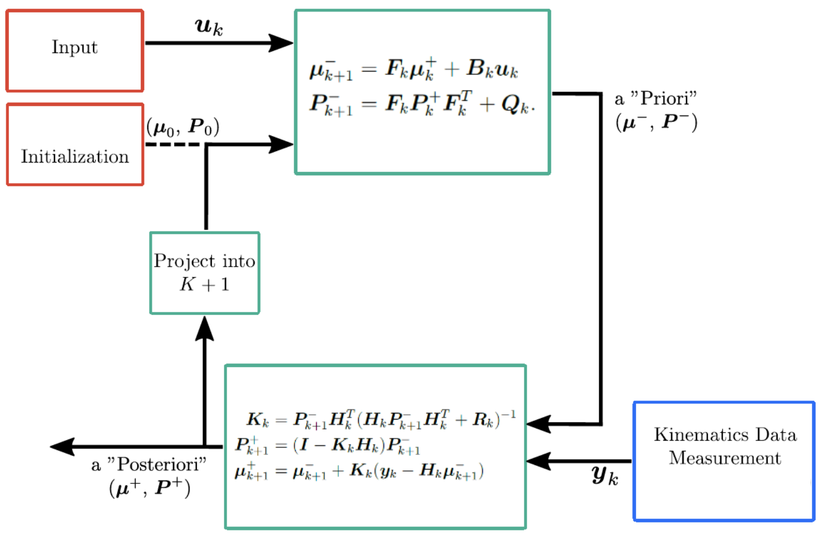
$$\begin{aligned}\boldsymbol{\mu}_{k+1}^- &= \boldsymbol{F}_k\boldsymbol{\mu}_k^+ + \boldsymbol{B}_k\boldsymbol{u}_k\\ \boldsymbol{P}_{k+1}^- &= \boldsymbol{F}_k\boldsymbol{P}_k^+\boldsymbol{F}_k^T + \boldsymbol{Q}_k.\end{aligned}$$

2. Update Step

Then the updated “a posteriori” state estimate and estimate error covariance matrix are computed as follows

$$\begin{aligned}\boldsymbol{K}_k &= \boldsymbol{P}_{k+1}^-\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_{k+1}^-\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}\\ \boldsymbol{P}_{k+1}^+ &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k+1}^-\\ \boldsymbol{\mu}_{k+1}^+ &= \boldsymbol{\mu}_{k+1}^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k\boldsymbol{\mu}_{k+1}^-)\end{aligned}$$

where K_k is the optimal Kalman gain.


Input
$\boldsymbol{u}_k$
$\boldsymbol{\mu}_{k+1}^- = \boldsymbol{F}_k\boldsymbol{\mu}_k^+ + \boldsymbol{B}_k\boldsymbol{u}_k$
$\boldsymbol{P}_{k+1}^- = \boldsymbol{F}_k\boldsymbol{P}_k^+\boldsymbol{F}_k^T + \boldsymbol{Q}_k.$
Initialization
$(\boldsymbol{\mu}_0, \boldsymbol{P}_0)$
a ”Priori”
$(\boldsymbol{\mu}^-, \boldsymbol{P}^-)$
Project into $K+1$
$\boldsymbol{K}_k = \boldsymbol{P}_{k+1}^-\boldsymbol{H}_k^T(\boldsymbol{H}_k\boldsymbol{P}_{k+1}^-\boldsymbol{H}_k^T + \boldsymbol{R}_k)^{-1}$
$\boldsymbol{P}_{k+1}^+ = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k+1}^-$
$\boldsymbol{\mu}_{k+1}^+ = \boldsymbol{\mu}_{k+1}^- + \boldsymbol{K}_k(\boldsymbol{y}_k - \boldsymbol{H}_k\boldsymbol{\mu}_{k+1}^-)$
a ”Posteriori”
$(\boldsymbol{\mu}^+, \boldsymbol{P}^+)$
$\boldsymbol{y}_k$
Kinematics Data Measurement

Reference

Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institut für Informatik
Abteilung VI
Humanoid Robots Lab

[Kalman60] Kalman, R.E.: “A new approach to linear filtering and prediction problems. Journal of basic Engineering,” Vol. 82, 1960.