Exercise 07 for MA-INF 2201 Computer Vision WS24/25
Submission on 15.12.2024

Overview

This exercise focuses on state estimation and tracking, combining theoretical foundations with practical implementation. You will implement a Kalman filter and a fixed-lag smoother to track a moving object in 2D space. Use the numpy and matplotlib libraries for implementation and visualization. Provide detailed comments for your implementation.

1. **Kalman Filter Implementation:** Consider a 2D signal with acceleration. The state vector includes position, velocity, and acceleration components: $\mathbf{x} = [x, y, v_x, v_y, a_x, a_y]^T$.

   Data: The observation data is provided in 'data/observations.npy' file which contains noisy 2D position measurements file which can be loaded using:

   ```
   observations = np.load('observations.npy')  # Shape: (N, 2)
   ```

   System Model:

   - Time step: $\Delta t = 0.1$ seconds
   - State transition matrix Ψ incorporating acceleration terms:

   $$\Psi = \begin{bmatrix} 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

   - Process Noise Covariance (Σ_p): Diagonal matrix with process noise parameter $sp = 0.001$
   - Measurement Matrix (Φ) observing only positions:

   $$\Phi = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

   - Measurement Noise Covariance (Σ_m): Diagonal matrix with measurement noise parameter $sm = 0.05$

   1.1. What should be the time evolution equation? Explain each term. *(1 point)*

   1.2. What should be the measurement equation? Explain each term.*(1 point)*

   1.3. Implement a Kalman filter to track the signal: Initialize the state vector with $\mathbf{x}_0 = [-10, -150, 1, -2, 0, 0]^T$. Implement the prediction step (time update). and the correction step (measurement update). *(6 points)*

   1.4. Visualize the observations vs. filtered estimates. *(2 points)*

2. **Fixed-Lag Smoothing:** Extend your Kalman filter implementation to perform fixed-lag smoothing for improved state estimation.

   2.1. Implement the smoother with a lag of 5 time steps. *(6 points)*

   2.2. Analysis of lag size effects. *(4 points)*

Submission Guidelines

- Submit your implementation in a Python file named solution.py, which should include two classes: KalmanFilter and FixedLagSmoothing.
- Include clear documentation and comments in your code.
- Provide theoretical answers and analysis (1.1, 1.2, explanation of 2.2) in `answers.pdf`.(Please limit your answers to no more than 15 lines.)