Giorgio Mendoza

RBE595-S24-S01

Dr. Brodovsky

Kalman Filter Report

Task 1: Kalman Filter Equations

Using the provided information, determine the matrices used in the process and measurement models. Note, you do not need to include the noise matrices. For the process model, be sure to start from a continuous time model, extract the time step from the data, and use that to create a discrete time model. Note that in doing this you may treat the drone as a point mass and the input has no constraints on it. This is not realistic but should still yield good results.

Given variables:

m = 27 grams

covariance matrix = $\Sigma = \sigma^2 I$, where I is identity matrix and σ^2 is the covariance

σ = 0.05m (low-noise)

σ = 0.20m (high-noise)

σ = 0.05m/s (velocity)

standard deviation $\sigma = 10^{-6}$(too low); $\sigma = 10^{3}$(too large)

State Transition Matrix (F):

This is a 6x6 matrix that transitions the state from one time step to the next. Since our state includes position and velocity for three dimensions, and we're assuming no acceleration (constant velocity model).

State Variables: The state vector x is [position, velocity] or [p, ṕ]

Kinematic Equations: The next position p_next is the current position p plus the current velocity ṕΔt. The velocity ṕ_next is assumed to be constant over the small time interval Δt. So the final matrix form of the transition matrix F for a constant velocity model is:

$$F = \begin{pmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Control Input Matrix (B):

This matrix accounts for the effect of control input (thrust) on the drone's acceleration, in other words, the control input matrix relates the control input to the state vector.

Control inputs: The control input ‘u’ is proportional to the force applied to the drone.

Newton’s 2nd Law: Acceleration is force over mass, so $\boldsymbol{u}=m\ddot{\boldsymbol{p}}$

Matrix Form: The control input matrix ‘B’ assuming direct control of acceleration would be:

$$B = \begin{pmatrix} \frac{(\Delta t^2)}{2m} & 0 & 0 \\ 0 & \frac{(\Delta t^2)}{2m} & 0 \\ 0 & 0 & \frac{(\Delta t^2)}{2m} \\ \frac{\Delta t}{m} & 0 & 0 \\ 0 & \frac{\Delta t}{m} & 0 \\ 0 & 0 & \frac{\Delta t}{m} \end{pmatrix}$$

Observation Matrix (H):

This is the matrix that maps the true state space into the observed space. For position measurements, it looks like:

$$H_{position} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

And for velocity measurements:

$$H_{velocity} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

However, if we are measuring position and velocity directly, then ‘H’ would be:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Process Noise Covariance Matrix (Q):

This matrix represents the process noise. It's usually a diagonal matrix where the diagonal elements are the variances of the process noise for each state variable.

Noise Characteristics: It can be assumed that each state variable has noise with a standard deviation σ_process. Then, the process noise covariance matrix ‘Q’ is:

$$Q = \begin{pmatrix} \sigma^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2 \end{pmatrix}$$

Measurement Noise Covariance Matrix (R):

Similar to Q, but for measurement noise.

Measurement Noise: It can be assumed that each measured variable has noise with a standard deviation σ. If the standard deviation for measurement noise is σ, then the measurement noise covariance matrix ‘R’ matrix is:

$$R = \begin{pmatrix} \sigma^2 & 0 & 0 \\ 0 & \sigma^2 & 0 \\ 0 & 0 & \sigma^2 \end{pmatrix}$$

However, if we are measuring six variables such as x, y, z positions and their respective velocities, then R would be a 6x6 matrix.

$$R = \begin{pmatrix} \sigma^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2 \end{pmatrix}$$

Initial Estimate and Covariance:

Initially, if no better estimate is available, the state vector can be set to zero, and the covariance matrix can be a scaled identity matrix representing initial uncertainty, such as:

$$x_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Where λ is a large number signifying initial uncertainty.

$$P_0 = \begin{pmatrix} \lambda & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & \lambda & 0 \\ 0 & 0 & 0 & 0 & 0 & \lambda \end{pmatrix}$$

Task 2: Kalman Filter Implementation

The next task is to implement the Kalman filter to track the motion of the drone using the matrices derived. The .txt files provided can be used to produce the state vector at each time step as well as a three-dimensional plot of the three-dimensional position at minimum.

The initial covariance is a design parameter that can be tuned.

Code Explanation:

Google Drive Mounting: The code starts by mounting the Google Drive to access data files stored there. This is required since Google Colab is used to develop the assignment.

Data Loading: The given data files are read into Pandas DataFrames. Each file corresponds to a set of motion capture data under different conditions, for example, the actual motion capture data, velocity data, and data with different levels of noise (low and high). The columns for time, control inputs (u1, u2, u3), and actual measurements (z1_actual, z2_actual, z3_actual) are named accordingly.

3D Trajectory Plotting: The plot of the actual path of the drone in 3D space is plotted first to get an idea how the desired output of the Kalman filter should look. Similarly, the velocity and high/low noise data is also plotted to get an idea how this data looks in 3D space.

Kalman Filter Implementation:

- Initial State and Covariance: The Kalman Filter is initialized with the assumption that the drone starts from rest at the origin, and with high initial uncertainty in the state estimation.
- Time Step Calculation: The code calculates the average time step (Delta_t) between measurements, which is used in the state transition matrix to model time-dependent changes.
- Matrix Definitions: The state transition matrix (F), control input matrix (B), process noise covariance (Q), and measurement noise covariance (R) matrices are defined based on the system's dynamics and the characteristics of the noise affecting the process and measurements.
- State Estimation Loop: A loop iterates through the synchronized position and velocity data, applying the Kalman Filter predict and update steps to estimate the state at each time step. The predicted state and updated state are stored at each iteration.
- Observation Matrix (H): An observation matrix suitable for direct measurement of both position and velocity is used in the update step.

State Storage and Plotting: The estimated states are collected into a DataFrame for easy manipulation and visualization. At the end the final plots are generated to display the estimated 3D path of the drone based on the Kalman Filter's output, providing a visual comparison of the drone's estimated trajectory against its

actual path. A higher sigma_process was needed for high-noise data to achieve accurate tracking while over-reducing sigma_process resulted in distorted estimations.

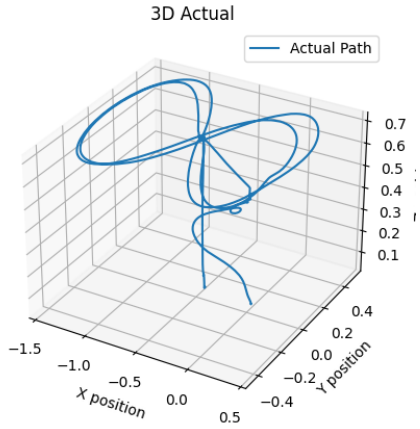

Figure 1. Actual Trajectory Path in 3D

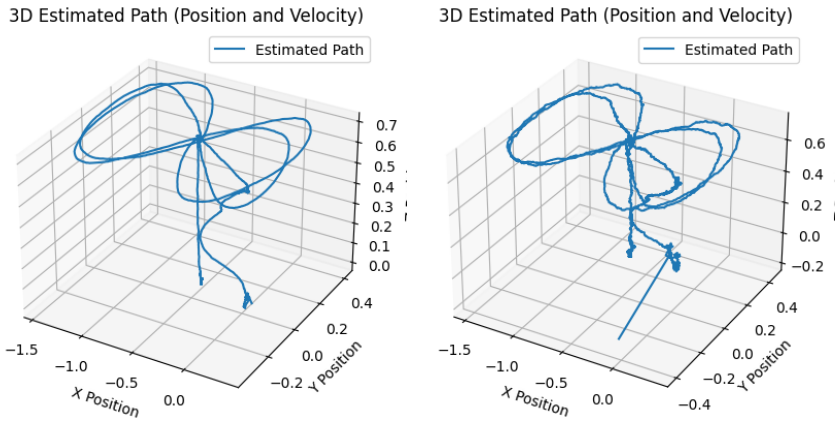

Figure 2. Estimated Trajectory Path w/ Low-Noise and w/ High- Noise

Figure 1 shows the drone's actual flight path as tracked by the motion capture system. It serves as the ground truth against which the Kalman Filter's performance can be evaluated. The trajectory appears to be smooth which is typical for motion capture data with high precision and low noise.

On the other hand, the plot on the lower left represents the estimated trajectory under low-noise conditions while the one on the lower right illustrates the estimated trajectory under high-noise conditions.

In both scenarios, the Kalman Filter has produced a trajectory that closely follows the complex path of the actual flight. The estimated path in under low noise aligns more closely with the actual path, indicating the filter's capability to effectively track the drone's position and velocity when the measurements are relatively clean. The path estimated with high noise shows more deviation from the actual path but still captures the general behavior and shape of the flight trajectory.

Overall, the estimated trajectories can be used to verify that the Kalman Filter is a capable tool for real-time state estimation in systems with noisy measurements.