IMPORTANT

A PDF version of this file is provided (`README.pdf`) which shows the math equations/formulas (Github does not parse them).

CS791X Project: Sensor Fusion with Kalman Filter

This is the first project for the CS791X (Robotics+Mobile SensorNetworks (MSNs)) (Fall 2015, University of Nevada, Reno) class. In this project we design a Kalman filter to fuse sensor data from GPS, IMU, and the wheel encoder of a mobile robot. The result is compared to the raw data from the sensors. The comparison shows a smoother trajectory when the filter is used.

Introduction

In this project we design a Kalman filter to fuse sensor data from GPS, IMU, and the wheel encoder of a mobile robot. The result is compared to the raw data from the sensors. The comparison shows a smoother trajectory when the filter is used.

Kalman Filter Design

The filter is a common linear Kalman filter. First, the raw data from the sensors have to be calibrated before any manipulation is executed. To calibrate the IMU heading (θ_i), the data gathered is first reduced by the first measurement of the wheel encoder heading (which has less accumulated error), resulting in the corrected value of θ_{ic}.

Since the GPS data sometimes present a null covariance (missing data, it is not updated), the system utilizes the odometry as a substitute for the missind information. Another preparation step is to extract the angular velocity from the wheel encoder (odometer) dataset, which consisting of the position (x_o, y_o) and heading (θ_o), though only the heading (and a fixed translational velocity V) is used to obtain the angular velocity ω. This is simply done by the following equation, where the translational velocity is assumed to be $V = 0.14m/s$, and the distance between the robot's wheels is fixed at $L = 1$ meter:

$$\omega(k) = \frac{V * tan(\theta_{odo}(k))}{L}$$

Where k denotes the iteration step. Also the time interval between iterations is fixed in $dt = .001s$.

Kalman Filter class

Refer to the `KF.py` file to check the filter implementation.

Initialization

The Kalman filter was implemented using the *python* programming language. The definition of the matrices can be found in the lecture materials of the class, and for the interest of lenght they will not be discussed here. The initial/default values of the system are:

$$A = \begin{bmatrix} 1 & 0 & V\Delta t\theta_{k-1} & 0 & 0 \\ 0 & 1 & V\Delta t\theta_{k-1} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

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

$$Q = \begin{bmatrix} .00004 & 0 & 0 & 0 & 0 \\ 0 & .00004 & 0 & 0 & 0 \\ 0 & 0 & .00001 & 0 & 0 \\ 0 & 0 & 0 & .00001 & 0 \\ 0 & 0 & 0 & 0 & .00001 \end{bmatrix}$$

$$R = \begin{bmatrix} .04 & 0 & 0 & 0 & 0 \\ 0 & .04 & 0 & 0 & 0 \\ 0 & 0 & .01 & 0 & 0 \\ 0 & 0 & 0 & .01 & 0 \\ 0 & 0 & 0 & 0 & .01 \end{bmatrix}$$

$$P = \begin{bmatrix} .01 & 0 & 0 & 0 & 0 \\ 0 & .01 & 0 & 0 & 0 \\ 0 & 0 & .01 & 0 & 0 \\ 0 & 0 & 0 & .01 & 0 \\ 0 & 0 & 0 & 0 & .01 \end{bmatrix}$$

$$X = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & V & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Where the velocity $V = 0.14$ is constant, and Δt is the time step between measurements, which is also constant 0.001.

Results

Refer to the `SensorFusion.ipynb` notebook to visualize the `KalmanFiltert` class, as well as the experiment results.