Task 4: Tracking Multiple Objects

Ashwani Kumar Kamal

***Abstract*— The purpose of this document is to show the use of Bayes' filter such as Kalman Filter in optimally estimating the trajectory of a drone using noisy and faulty sensors. Kalman filter uses recursive equations to minimise variance by combining two probability density functions (need to be Gaussian distribution).**

I. INTRODUCTION

The problem starts with some csv files populated by some 10,000 rows of data. These values are given after being reported by sensors in ground stations (6 in total). However being noisy in nature, we need to use them to come up with a better estimate of the drone's trajectory (by application of Kalman Filter).

II. PROBLEM STATEMENT

Consider a drone flying in an area. There are 6 ground stations in the area and son seeing the drone they provided you with the coordinates of the drone with respect to them. The exact coordinates of ground stations are also unknown. Your job is to localise the drone and predict its trajectory given that measurements provided by ground stations are noisy due to instruments. Although from experiments we know that noise is distributed according to gaussian.
Problem set 1: This is an easier version of the problem where you are provided with 3 CSV files (CSV1, CSV2, CSV3) with varying magnitude of noise and you have to predict the trajectory of the drone in all the three cases.
Problem set 2: This is a slightly difficult version of the problem where due to server issues some of the values reported by ground stations were modified and were erroneous. Finally you are again supposed to predict the correct trajectory of drone. For this also you are provided with a CSV file (CSV4).
For solving this problem, you need to learn about kalman filters which are used to track an object from noisy data provided by some sensor like ground station in this case

III. RELATED WORK

Read about Kalman Filter and Gaussian distribution function.

IV. INITIAL ATTEMPTS

I was completely clueless to how to solve this task. I had a moderate intuition of Kalman Filter and the fact that we need two sets of data: the observed data and the estimated data, however given only measurement values, I was confused about implementing Kalman. Took me a long while to conclude that we can use the measurement of other ground stations as estimation for the others (perhaps because I had this preconceived notion that we necessarily require a state transition model to list out estimated data).

V. FINAL APPROACH

Since I solved only the first part of the problem, I'll be describing only that in the following lines. Given we have atleast two ground stations I employed the following algorithm to find estimated data set-

- For making the work easier, a class named $Tower()$ has been implemented. It will store three lists x, y, z containing the read values. The method $get\_relative$ will get the relative position of this tower with respect to some other.
- Take the data from first two sensors (6 fields i.e. $\{x_i, y_i, z_i\}_a$ and $\{x_i, y_i, z_i\}_b$ for $i \in [1, 1000]$, a being the first and b being the second ground station)
- Consider the location of first ground station as Origin
- Calculate the relative location of second ground station with respect to the first by simple subtraction (It has been assumed that all ground stations use common axes directions) and then taking an average of those values. Let the this computed location be $(\Delta x, \Delta y, \Delta z)$
- Now take the set $\{x_i, y_i, z_i\}_b$ (second ground station measurement values) and subtract the computed relative location from each row, i.e. compute $\{x_i - \Delta x, y_i - \Delta y, z_i - \Delta z\}_b$
- This new set should be estimated values set for drone after shifting origin from first ground station to second.

The original set $\{x_i, y_i, z_i\}_a$ will be measurement set and newly computed set $\{x_i - \Delta x, y_i - \Delta y, z_i - \Delta z\}_b$ will be the estimated set for first ground station. Then I applied Kalman filter equations on the this data. to obtain the trajectory.
The choice of covariance matrix was made as follows:
Let R and Q be the covariance matrix associated with measurements and estimation model respectively. By definition for a three dimensional set $\{x_i, y_i, z_i\}$ the covariance matrix is defined as follows-

$$\begin{bmatrix} \sigma_x^2 & \sigma_x.\sigma_y & \sigma_x.\sigma_z \\ \sigma_y.\sigma_x & \sigma_y^2 & \sigma_y.\sigma_z \\ \sigma_z.\sigma_x & \sigma_z.\sigma_y & \sigma_z^2 \end{bmatrix}$$

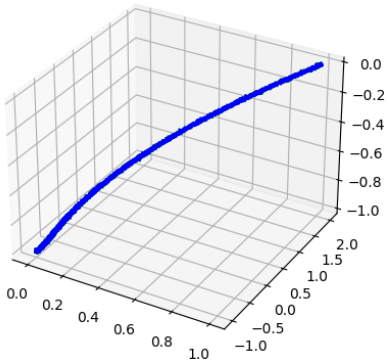
For independent measurements $\sigma_x.\sigma_y$ is zero as change in one quantity does not affect the other but since here all three components in (x, y, z) are taken from single sensor (again an generalised assumption) I have calculated all these values using cov method of $numpy$ module (iterated through all the values and calculated the variance and covariance).
Rest is a simple iteration through the big data pool and application of recursive Kalman equations to evaluate optimal estimated values (posterior estimated set). Then using the almighty $matplotlib$ module, trajectory is plotted and saved

into the output directory.
Then I thought how can I make the trajectory better by using other ground stations. The algorithm that I devised for doing the same has been documented in the following lines-

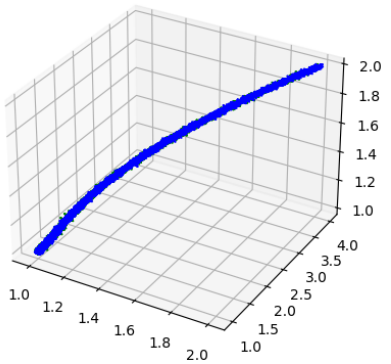
- Take the optimal trajectory produced by first two ground stations as seen in frame of first ground station
- Call this the new measurement data for first ground station, the associated measurement covariance is updated to the value of process noise covariance
- Now repeat the Kalman filter algorithm on first and third ground station, however this time, measurement data for first ground station is previous optimal trajectory (similar analogy for covariance)
- Obtain the newly generated optimal trajectory and call this the new measurement data for first ground station (update covariance again)
- Repeat the process on remaining ground stations
- This should get you 5 trajectories with the last one being most confident and optimal.

VI. Results and Observation

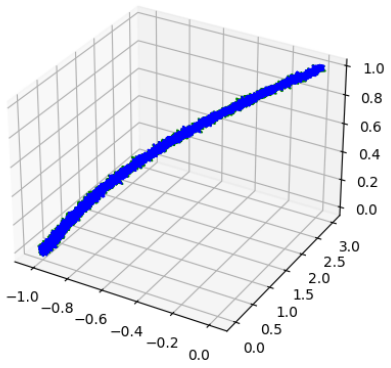
Data3.csv contains the most noisy data, followed by Data2.csv then Data1.csv. The final trajectories for all three cases are given below.



Data1 final trajectory



Data2 final trajectory



Data3 final trajectory

Blue plot is the optimal filtered data and grey one is the measurement data from ground station a.
One thing which can be clearly observed is that since high precision values were given, the trajectory seems very jagged and non smooth. However if we scale our data by some factor (say 100) hence lowering the precision, the data visualised can be a bit smoother.

VII. Future Work

Learning about other techniques of estimation, applying particle particle or unscented Kalman filter.

Conclusion

At first it seems like a very tough task that we can optimally estimate data using faulty and noisy data, but there exist some methods and algorithms through which we can do that. However not accurate but better and close to accurate.

References

[1] Youngjoo Kim, Hyochoong Bang "Introduction to Kalman Filter and Its Applications", IntechOpen, 2018
[2] Tony Lacey "Tutorial: The Kalman Filter", MIT
[3] Matlab "Understanding Kalman Filters", Youtube, 2017