udacity-nanodegree-estimation

Sumedh V Joshi

August 2020

1 Introduction

This report discusses the various methods implemented in the udacity drone estimator project. In the controller project, it was assumed that all the sensors are perfect (without any noise or measurement error) and hence the estimated positions, velocities were always exact. This is very rarely the case with real sensors. Hence, in the following project, the following workflow has been adapted;

1. Initially, the measurements of the GPS and accelerometer in X direction are analyzed to fully understand the concept of noise in a sensor.
2. The EKF process begins from here. Initially, a better attitude estimator is implemented which accounts for non-linearity's in measurements.
3. Next, the predict step for the estimator is implemented. This starts with defining the new covariance value for the state of the drone. The state mean and covariance from the previous time step and the current thrust vector.
4. Then the individual sensor measurements are read (GPS, magnetometer and so on).
5. Finally, these new sensor measurments, state covariance and mean values from predict step are used to update the final state mean and covariance values.

2 Analysing sensor data for X position and velocity

After running scenario 6 in C++, the generated txt. data is imported into an excel file. The Standard deviation function in excel was directly used to calculate the respective values.

3 Predict state

For the predict state, the classical EKF equation mentioned in the reference pdf, estimation for quadrotors has been implemented.

4 GPS and magnetometer update

The update function is also implemented in exactly the similar way as described in the estimation for quadrotors pdf.

5 Estimator and controller

In this section, I have used the controller implemented in the previous project in conjunction with the estimator. The sensors are modelled with noise, hence it is difficult to exactly estimate the position of the drone. This in turn leads to small changes in the thrust values obtained from the controller. Hence, the drone does not follow an exact square path and instead treads along a slightly different path but within the allowed deviation range. The results from scenario 11 are shown below.

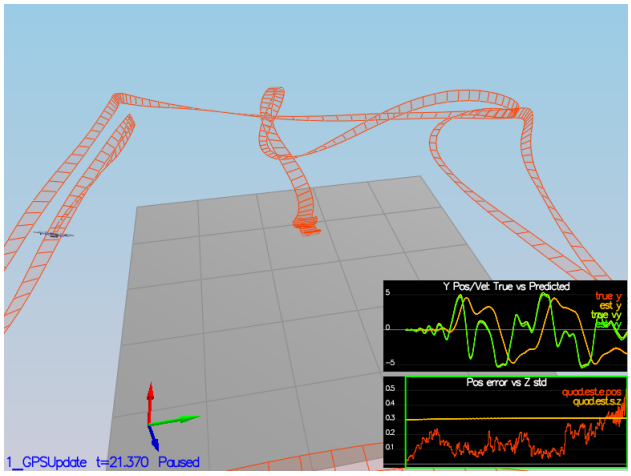

Figure 1: Scenario 11

```
Simulation #8 (../config/11_GPSUpdate.txt)
Simulation #9 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000 seconds
Simulation #10 (../config/01_Intro.txt)
Simulation #11 (../config/11_GPSUpdate.txt)
Simulation #12 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000 seconds
```

Figure 2: Scenario 11-B