Writeup / README

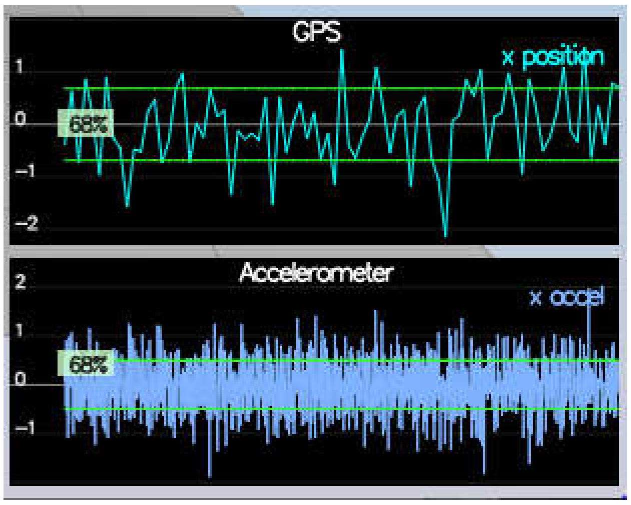
1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf.

You're reading it! Below I describe how I addressed each rubric point and where in my code each point is handled.

Implement Estimator

1. Determine the standard deviation of the measurement noise of both GPS X data and Accelerometer X data.

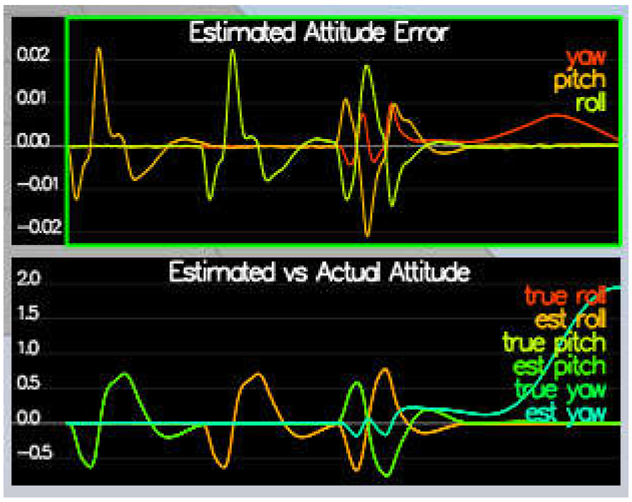
I extracted the data and copied the values into an Excel sheet. Then I used the method for estimating a standard deviation based on a small sample set for the GPS values (Student-t distribution). For the IMU values I used the same method but for a larger sampling set. The resulting values led to the desired 68% interval.

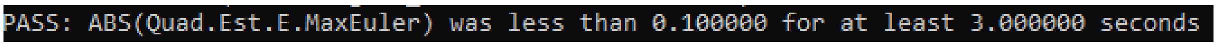


```
PASS: ABS(Quad.GPS.X-Quad.Pos.X) was less than MeasuredStdDev_GPSPosXY for 68% of the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for 68% of the time
```

2. Implement a better rate gyro attitude integration scheme in the UpdateFromIMU() function.

The complementary filter was improved in the method (lines 106 – 115). I used the quaternion approach. The estimates were converted to a quaternion and the measurements were integrated. Additionally, the yaw angle was normalized.

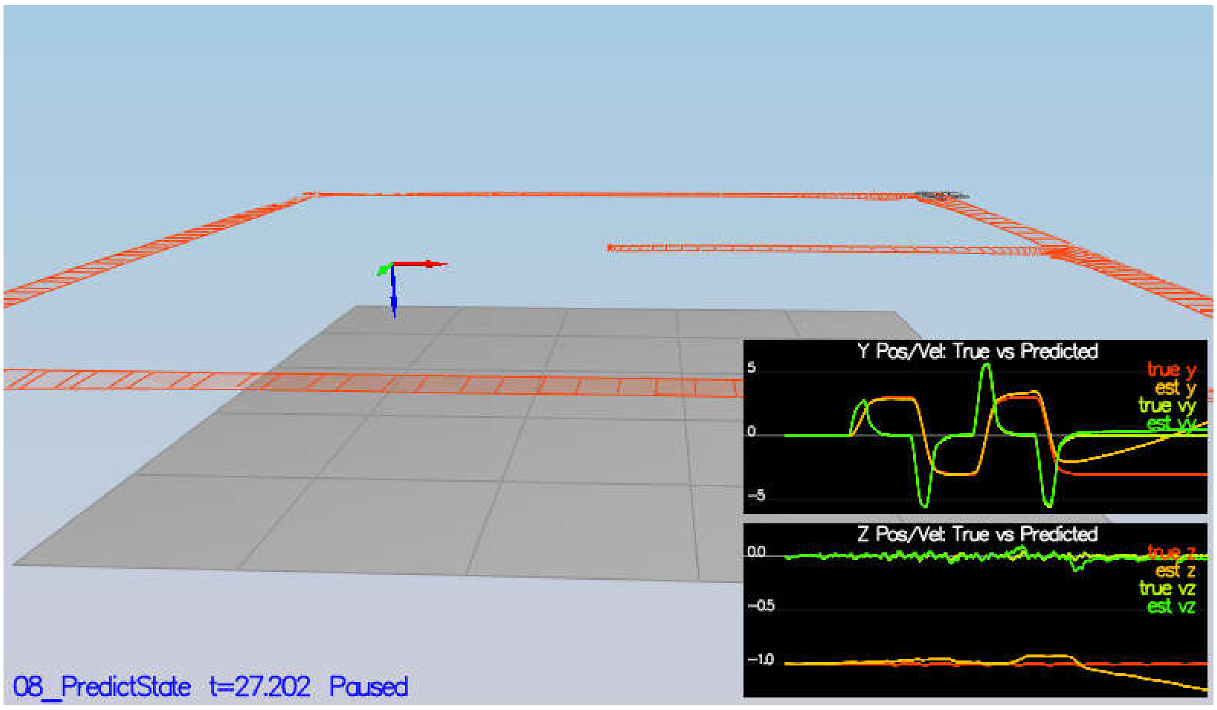




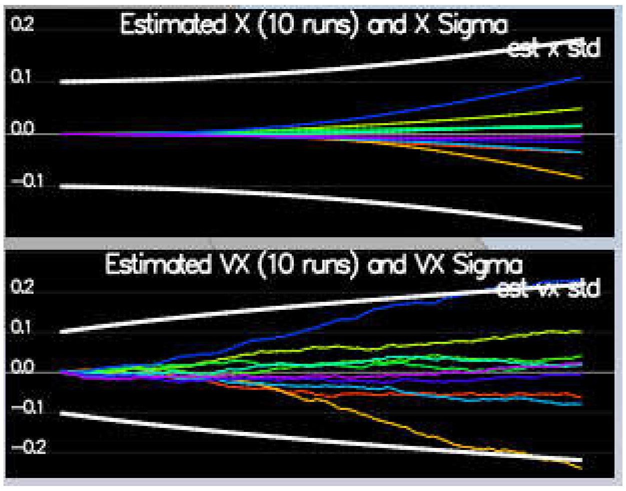
```
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
```

3. Implement all of the elements of the prediction step for the estimator.

The predictState function was implemented (starting line 178). I used a simplistic integration by just multiplying the respective derivate to the current state. The provided quaternion was used to transform the body accelerations into the inertial frame. The estimated position drifts slowly as expected.

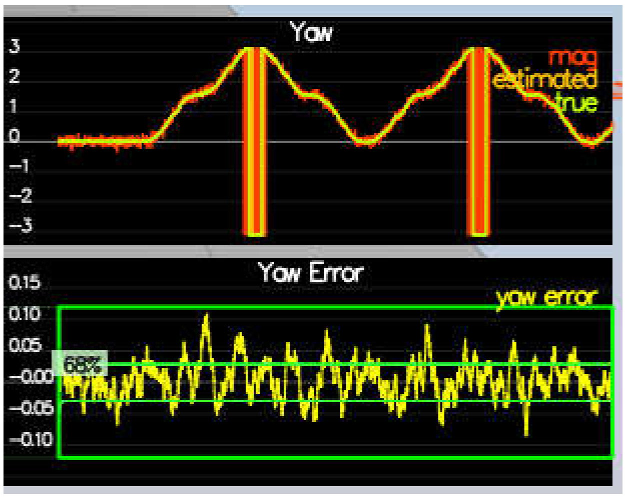

Y Pos/Vel: True vs Predicted
5
0
−5
true y
est y
true vy
est vy
Z Pos/Vel: True vs Predicted
0.0
−0.5
−1.0
true z
est z
true vz
est vz
08_PredictState t=27.202 Paused

The RbgPrime matrix was implemented according to the paper “Estimation for Quadrotors” starting at line 213. Additionally, the predict function was implemented starting at line 268. The gPrime matric was implemented according to the paper “Estimation for Quadrotors”. For this, the covariance was estimated in the scenario.



4. Implement the magnetometer update.

I implemented the magnetometer update starting line 332. The hPrime matrix was implemented according to the paper “Estimation for Quadrotors” and the yaw angle was normalized to get the “short way around”.



```
PASS: ABS(Quad.Est.E.Yaw) was less than 0.120000 for at least 10.000000 seconds
PASS: ABS(Quad.Est.E.Yaw-0.000000) was less than Quad.Est.S.Yaw for 68% of the time
```

5. Implement the GPS update.

I implemented the GNSS update starting line 303. The hPrime matrix was implemented according to the paper “Estimation for Quadrotors”. I switched off the ideal estimator and enabled the error model for the measurements. The scenario was passed for the controller provided.

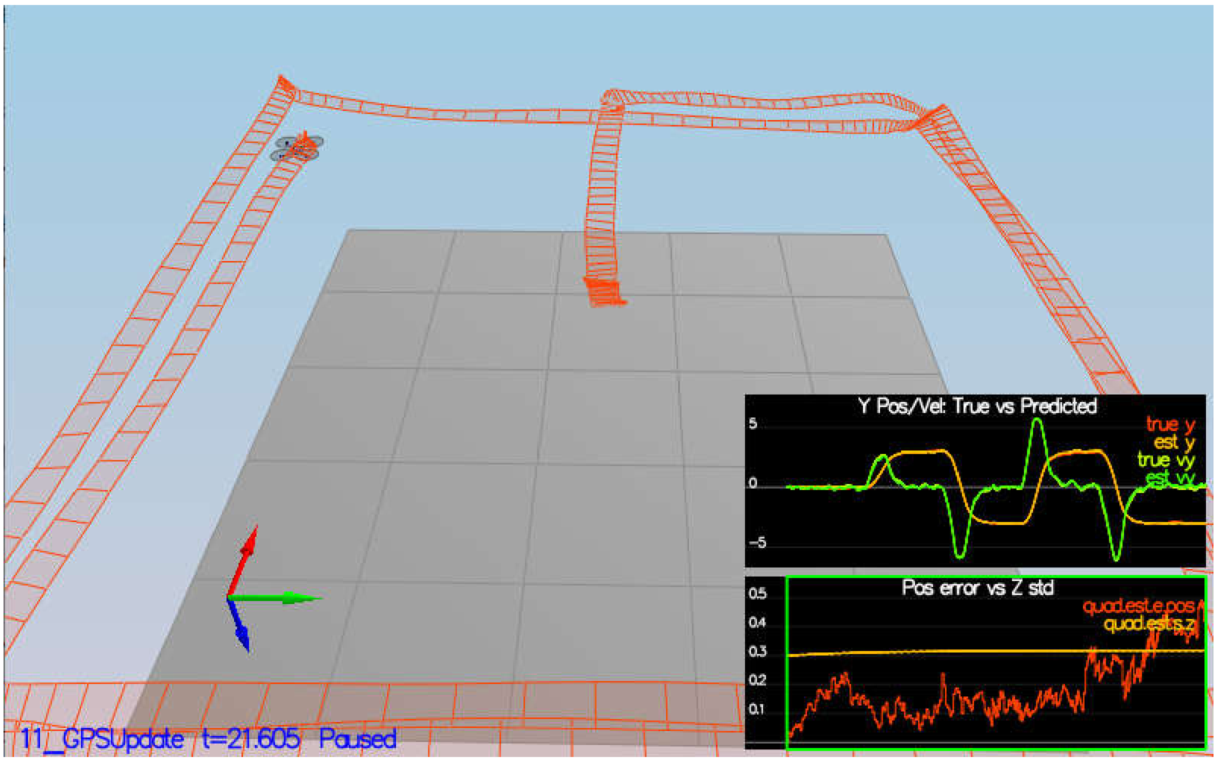
Flight Evaluation

1. Meet the performance criteria of each step.

As shown above, the estimator was able to successfully meet the performance criteria with the controller provided. The estimator's parameters were properly adjusted to satisfy each of the performance criteria elements.

2. De-tune your controller to successfully fly the final desired box trajectory with your estimator and realistic sensors.

Finally, I copied my implemented controller into the estimator project and de-tuned it to fulfil the success criteria as set out.



```
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000 seconds
```