Building an Estimator

1. **Writeup**
2. **Implement Estimator**
    - Determine Standard deviation of the measurement
    - Implement a better rate gyro attitude integration scheme
    - Implement all of the elements of the prediction step for the estimator
    - Implement magnetometer
    - Implement GPS

3. **Flight Evaluation**
    - Meet the performance criteria of each step
    - Successfully fly the final desired box trajectory with estimator and realistic sensors

Intro

Project's code is mainly a drone simulator provided by Udacity (see sources and detailed explanation provided in Udacity's readme.

Import the code into your IDE, cloning `git clone https://github.com/udacity/FCND-Estimation-CPP.git`

Tasks

Assessed tasks are given as a result of modifying 4 files mainly:

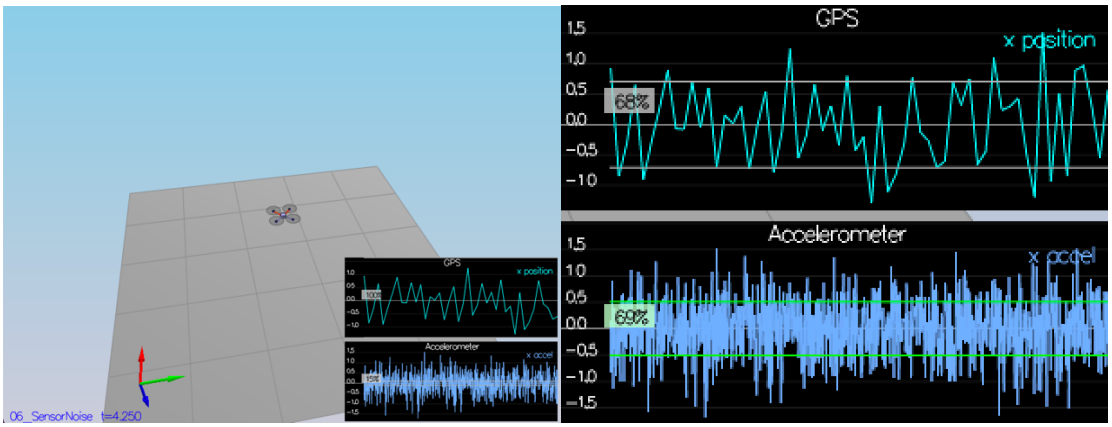
- `QuadEstimatorEKF.cpp`
- `QuadControl.cpp`
- `QuadEstimatorEKF.txt`
- `QuadControlParams.txt`

The tasks are detailed as follows:

Sensor Noise

Add noise to the quad's sensors to approximate control in a more realistic conditions environment. Steps to reproduce:

1. Run scenario 06_NoisySensors and collected some simulated noise sensors data
2. Calculate standard deviation (data registered in log folder, used `np.std` from Numpy)
3. Update `config/6_Sensornoise.txt1` with generated output (see full calculation steps in `sensorNoise.ipynb` available in the repo)
4. Re-run simulator calculated standard deviation correctly capture approximately 68% of the respective measurements as expected.

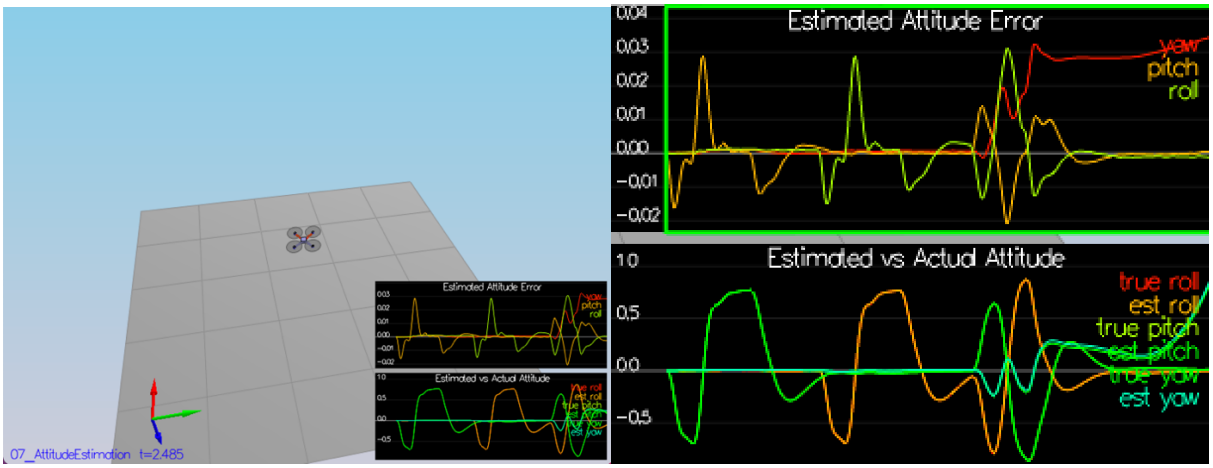



```
Simulation #64 (../config/06_SensorNoise.txt)
PASS: ABS(Quad.GPS.X-Quad.Pos.X) was less than MeasuredStdDev_GPSPosXY for 68% of the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for 69% of the time
```

Attitude Estimation

Steps to reproduce:

1. Run scenario 07_AttitudeEstimation. Sensors used: IMU with noise level set to 0 (`config/07_AttitudeEstimation.txt`)
2. Observe graph with errors in each of the estimated Euler angles (top graph)
3. Observe true Euler Angles and the estimates and observe error in attitude estimation

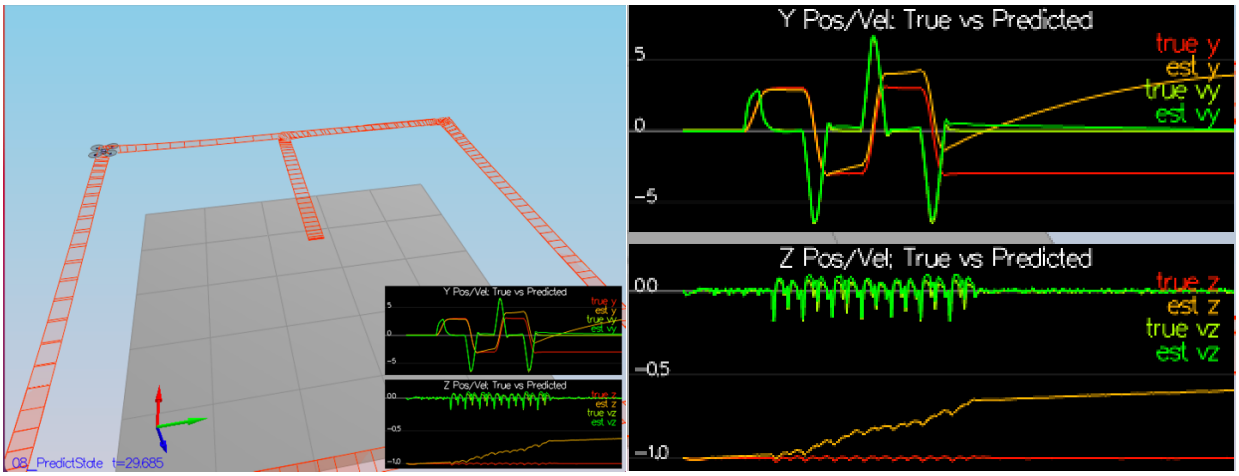



```
Simulation #13 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
```

Prediction State

Steps to reproduce:

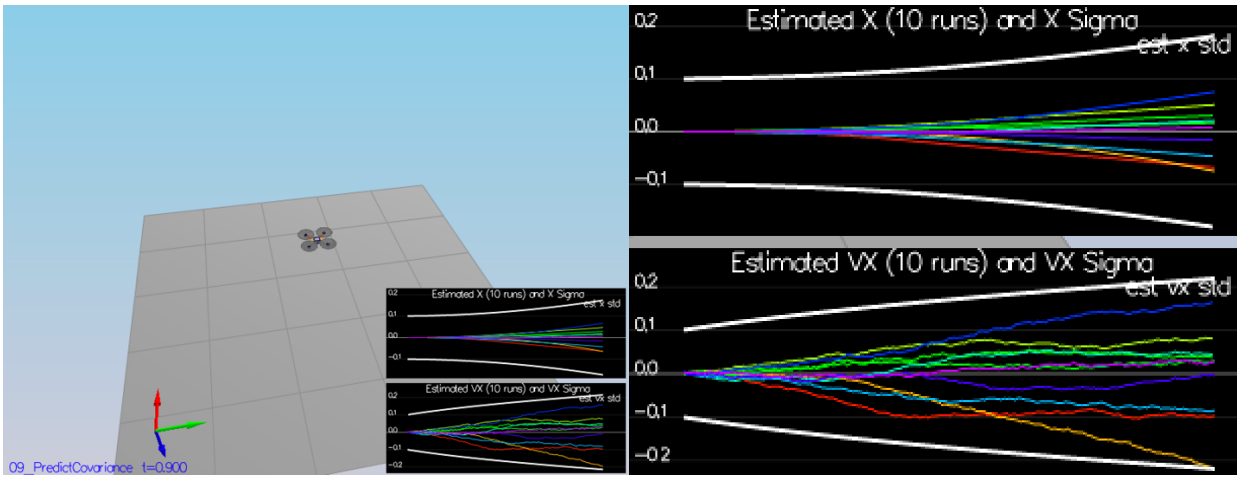
1. Run scenario 08_PredictState. (IMU sensor only with a `QuadEstimatorEKF.attitudeTau=100`)
2. Observe graph with elements of estimated state and true state
3. Implement state prediction in `PredictState()` function, where the transition function needs the rotation matrix Rbq which rotates from the body frame to the global frame.
4. Convert true acceleration from body to global frame (`attitude.Rotate_BtoI()`)



Predict Covariance

Steps to reproduce:

1. Run scenario 09_PredictionCov
2. Observe Position X estimates (Top Graph)
3. Observe velocity Vx estimates (prediction only at the Bottom Graph)
4. In `QuadEstimatorEKF.cpp` can be found the calculation of the partial derivative of the body to global rotation matrix in the function `GetRbgPrime()`

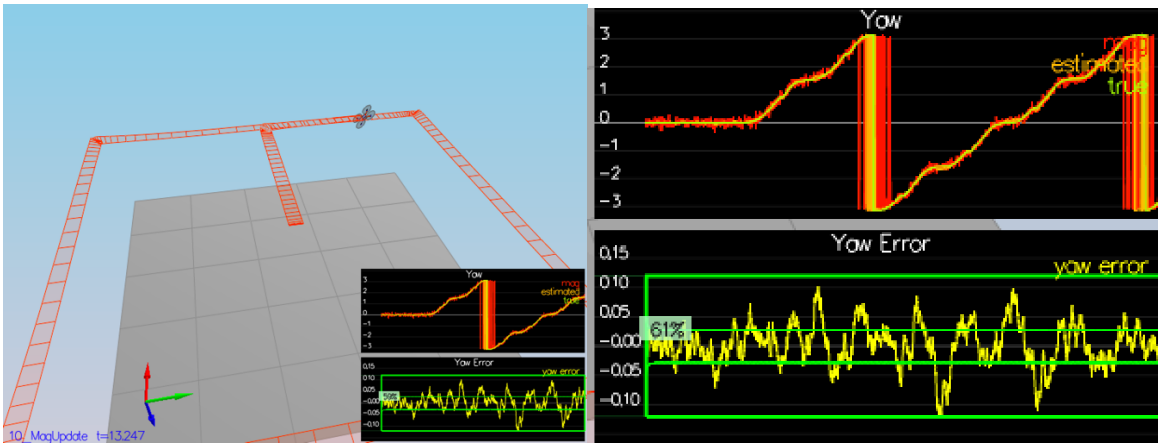


Magnetometer Update

The State of Estimation only has been using accelerometer and gyro until now that we are going to incorporate magnetometer in order to improve performance.

Steps to reproduce:

1. Run scenario 10_MagUpdate
2. Observe parameters tuned `QYawStd` (on `QuadEstimatorEKF.txt`) it will capture approximately the magnitude of the drift.
3. As a result of the implementation, yaw error decrease and sigma remain stable.



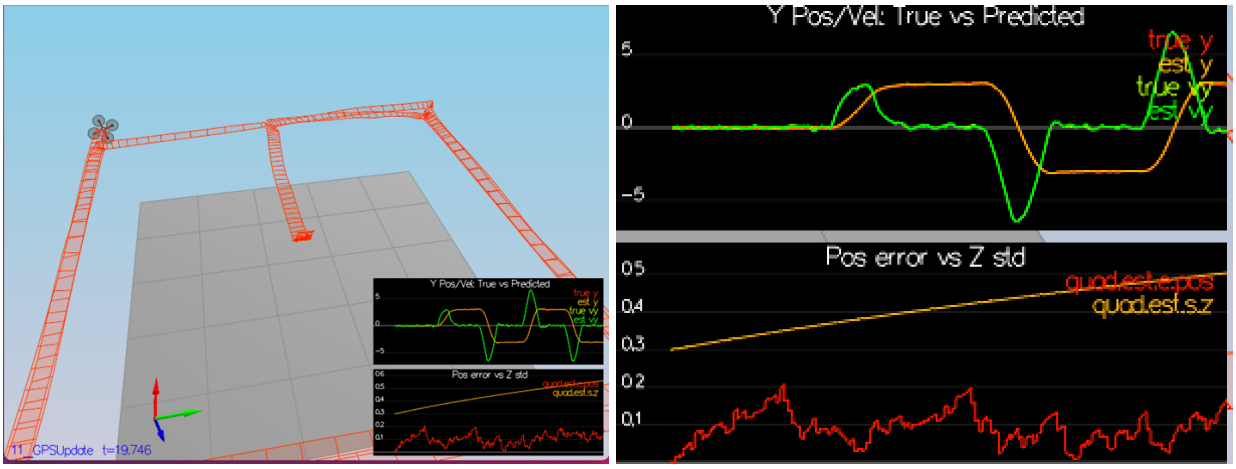

```
Simulation #78 (../config/10_MagUpdate.txt)
PASS: ABS(Quad.Est.E.Yaw) was less than 0.120000 for at least 10.000000 seconds
PASS: ABS(Quad.Est.E.Yaw-0.000000) was less than Quad.Est.S.Yaw for 66% of the time
```

GPS Update

Implements a GPS update step.

Steps to reproduce:

1. Run scenario 11_GPSUpdate
2. Change estimator stting `Quad.UseIdealEstimator` to 0 in `config/11_GPSUpdate.txt`
3. Implement a EKF GPS update `UpdateFromGPS()`
4. Observe successfully simulation with new conditions, flying fine in general terms without the ideal elements.




```
Simulation #82 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000 seconds
```

Flight Evaluation

Flying simulator succeed! You can change and play with the adjusted elements to understand how it behaves in different conditions!

Acknowledgment

The C++ simulator and project scenarios were largely designed and built by Fotokite. Big thanks to the Fotokite team (and Sergei Lupashin in particular) for their great work!