Estimation Final Project

[1. Introduction](https://classroom.udacity.com/nanodegrees/nd787/parts/a1505b23-c1aa-4bc6-a94c-d44d062d0209/modules/19b5af05-2ec7-491a-94db-1befc15d07c0/lessons/8c02e235-6019-40a2-b1b8-85d6dfe9a978/concepts/4c69a6e2-2c7e-49ba-be28-5007cc30671d)

Welcome to the estimation project. In this project, you will be developing the estimation portion of the controller used in the CPP simulator. By the end of the project, your simulated quad will be flying with your estimator and your custom controller (from the previous project)!

## **Setup**

This project will continue to use the C++ development environment you set up in the Controls C++ project.

1. Clone the repository
2. git **clone** https:*//github.com/udacity/FCND-Estimation-CPP.git*
3. Import the code into your IDE like done in the [Controls C++ project](https://github.com/udacity/FCND-Controls-CPP#development-environment-setup)
4. You should now be able to compile and run the estimation simulator just as you did in the controls project

## **Project Structure**

For this project, you will be interacting with a few more files than before.

* The EKF is already partially implemented for you in QuadEstimatorEKF.cpp
* Parameters for tuning the EKF are in the parameter file QuadEstimatorEKF.txt
* When you turn on various sensors (the scenarios configure them, e.g. Quad.Sensors += SimIMU, SimMag, SimGPS), additional sensor plots will become available to see what the simulated sensors measure.
* The EKF implementation exposes both the estimated state and a number of additional variables. In particular:
  + Quad.Est.E.X is the error in estimated X position from true value. More generally, the variables in <vehicle>.Est.E.\* are relative errors, though some are combined errors (e.g. MaxEuler).
  + Quad.Est.S.X is the estimated standard deviation of the X state (that is, the square root of the appropriate diagonal variable in the covariance matrix). More generally, the variables in <vehicle>.Est.S.\* are standard deviations calculated from the estimator state covariance matrix.
  + Quad.Est.D contains miscellaneous additional debug variables useful in diagnosing the filter. You may or might not find these useful but they were helpful to us in verifying the filter and may give you some ideas if you hit a block.

### **config Directory**

In the config directory, in addition to finding the configuration files for your controller and your estimator, you will also see configuration files for each of the simulations. For this project, you will be working with simulations 06 through 11 and you may find it insightful to take a look at the configuration for the simulation.

As an example, if we look through the configuration file for scenario 07, we see the following parameters controlling the sensor:

**# Sensors**  
Quad.Sensors = SimIMU  
**# use a perfect IMU**  
SimIMU.AccelStd = 0,0,0  
SimIMU.GyroStd = 0,0,0

This configuration tells us that the simulator is only using an IMU and the sensor data will have no noise. You will notice that for each simulator these parameters will change slightly as additional sensors are being used and the noise behavior of the sensors change.

Now that you have everything set up, you'll be able to dive right into the project!

The math that you'll need to implement is explained in the [Estimation for Quadrotors](https://www.overleaf.com/read/vymfngphcccj) document that you worked with in the lessons on Kalman Filters. The **Three Dimensional Quad** section should be most helpful.

### **Acknowledgment**

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

[2. Project: Building an Estimator](https://classroom.udacity.com/nanodegrees/nd787/parts/a1505b23-c1aa-4bc6-a94c-d44d062d0209/modules/undefined/lessons/8c02e235-6019-40a2-b1b8-85d6dfe9a978/project)

## Project Submission

## **The Estimation Project**

The final instruction for the project.

## **The Goal of this Project**

In this project, you will be developing an estimator to be used by your controller to successfully fly a desired flight path using realistic sensors. This project is built on the same simulator you should now be familiar with from the Controls C++ project.

## **Project Steps**

1. Make sure you have cloned the repository and gotten familiar with the additional project repository files **Introduction**.
2. Implement all the necessary update and prediction steps required for your estimator to meet all the performance criteria of each step, outlined in detail in the [project README](https://github.com/udacity/FCND-Estimation-CPP)
3. Tune your estimator, and re-tune your controller from your controls C++ project to successfully fly the desired trajectory with realistic sensors.

For more detailed instructions on the individual steps, make sure to read through the [project readme](https://github.com/udacity/FCND-Estimation-CPP), specifically the [section outlining the tasks for the project](https://github.com/udacity/FCND-Estimation-CPP#the-tasks).

## **Evaluation**

For the submission, you will be evaluated on your completion of each of the components of your estimator, and final performance of your estimator and previously made controller. Performance metrics are provided for each of the different scenarios, and your controller will need to meet these minimum performance metrics for each scenario. These performance metrics are outlined in each of the steps in the [task outline portion of the project readme](https://github.com/udacity/FCND-Estimation-CPP#the-tasks).

## **Submission**

For this project you will need to submit:

1. QuadEstimatorEKF.cpp and QuadEstimatorEKF.txt containing your estimator and associated estimator parameters that successfully meets all the performance criteria.
2. QuadController.cpp and QuadControlParams.txt containing your re-tuned controller needed to work successfully with your estimator.
3. a writeup addressing all the points of the [rubric](https://review.udacity.com/#!/rubrics/1807/view)

## **Feedback**

Please fill out the [Building an Estimator Project Feedback Form](https://goo.gl/forms/8Vm4IyEUVHadL2i52) after you have completed the project.

## **Share Your Project**

Have something fun to share? Controller flying in the desired box trajectory? Or maybe it wanted to do circles instead? We love to see any and all of your creations!

What you’ve accomplished is no small feat. Give yourself a pat on the back and some well-deserved recognition by sharing your amazing project with your network.

Create a gif or youtube video of your project implementation in the simulator and share with us in the [Flying Car Slack Community](https://udacity-flying-car.slack.com/messages/C77GCKZ8U) and on social media using the links below!

Make sure to use **@Udacity**, **#flyingcarnd**, **#quadestimator**, and **#madewithudacity** in your posts!