**Extended-Kalman-Filter**

**Write up** 1st submit:August 22th, Kenta Kumazaki

**1. Purpose**

In this project you utilized a Kalman Filter to estimate the state of a moving object of interest with noisy lidar and radar measurements. Passing the project requires obtaining RMSE values that are lower than the tolerance outlined in the project rubric as below.

RMSE of px, py, vx, and vy should be less than or equal to the values [0.11, 0.11, 0.52, 0.52]

**2. Goals/Steps**

The goals / steps of this project are the following:

* Complete the codes in FusionEKF.cpp, kalman\_filer.cpp, tools.cpp.
* Run Term2 Simulator and validate RMSE achieves target performance.
* Summarize the results with a written report

**3. Submission**

**(1) GitHub**

<https://github.com/kkumazaki/Self-Drivig-Car_Project5_Extended-Kalman-Filter>

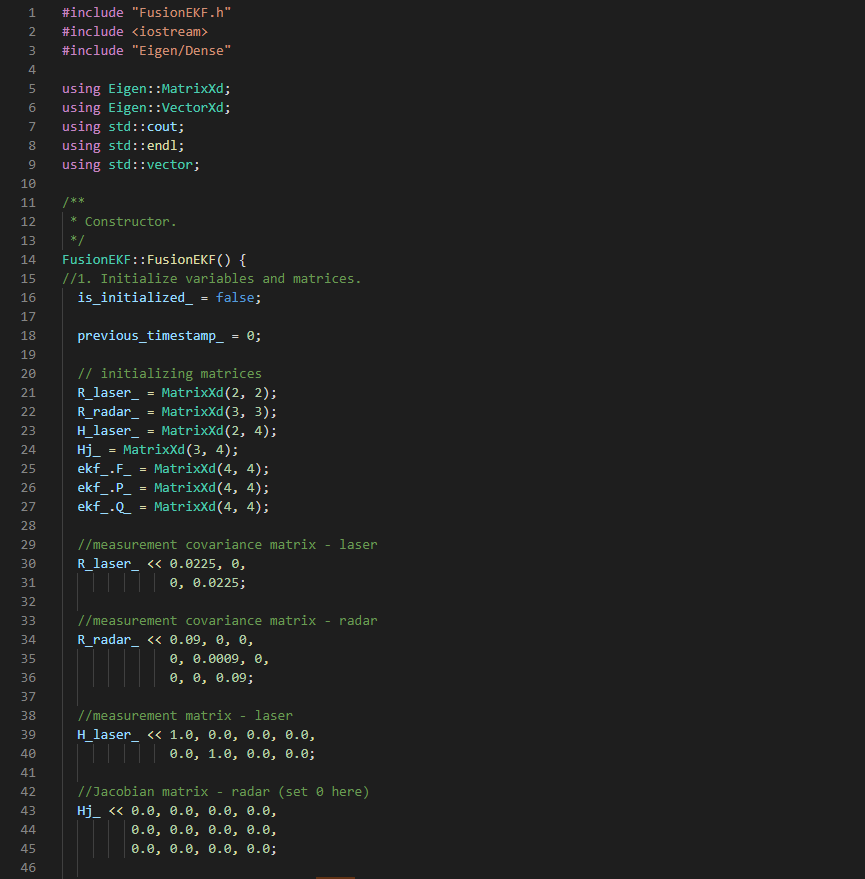
**(2) Directory**

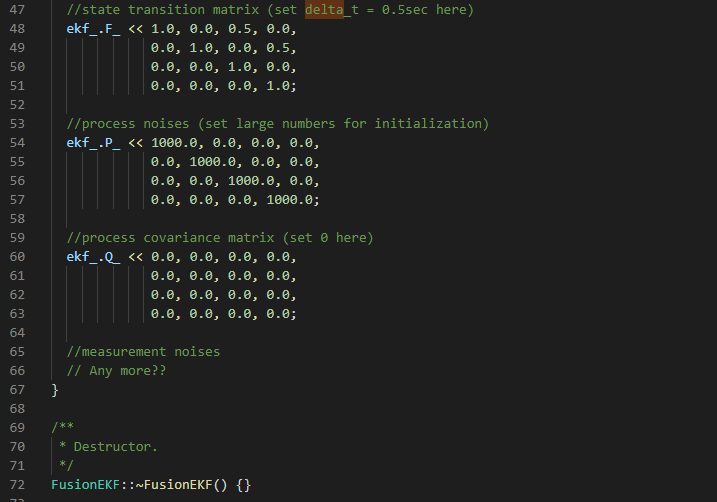
* **Writeup\_of\_Lesson24.pdf**: This file
* **src**
  + **main.cpp**: (didn’t modify this file)
    - Communicates with Term2 Simulator receiving data measurements.
    - Calls the function to run Kalman Filter.
    - Calls the function to calculate RMSE.
  + **FusionEKF.cpp**:
    - Initialize the filter.
    - Calls the prediction function.
    - Calls the update function.
  + **kalman\_filter.cpp**:
    - Define the prediction function.
    - Define the update function for LiDAR and Radar.
  + **tools.cpp**:
    - Calculate RMSE and Jacobian Matrix.

**4. Coding**

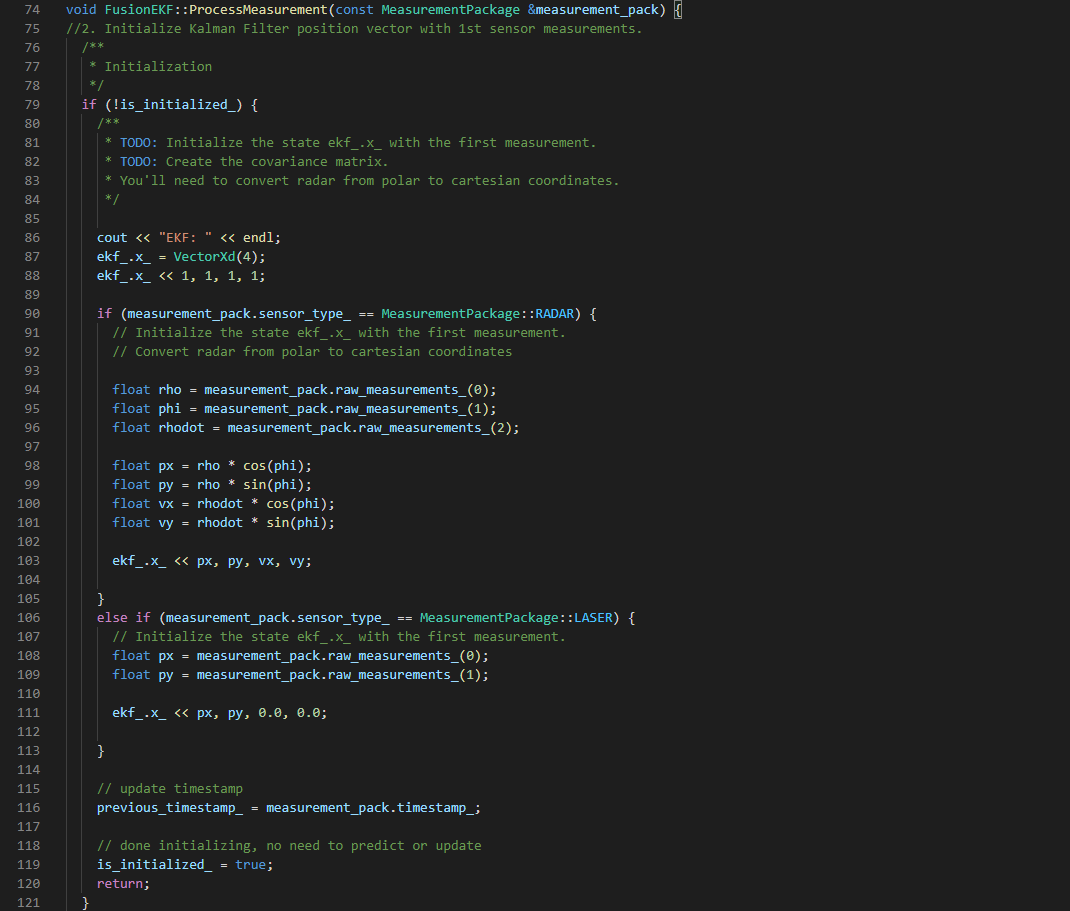
**(1) FusionEKF.cpp**

In Constructor, I initialized variables (previous\_timestamp\_) and Matrices (R, H, F, P, Q, etc).





In the function ProcessMeasurement(), at first I initialize Kalman Filter position vector with 1st sensor measurements.



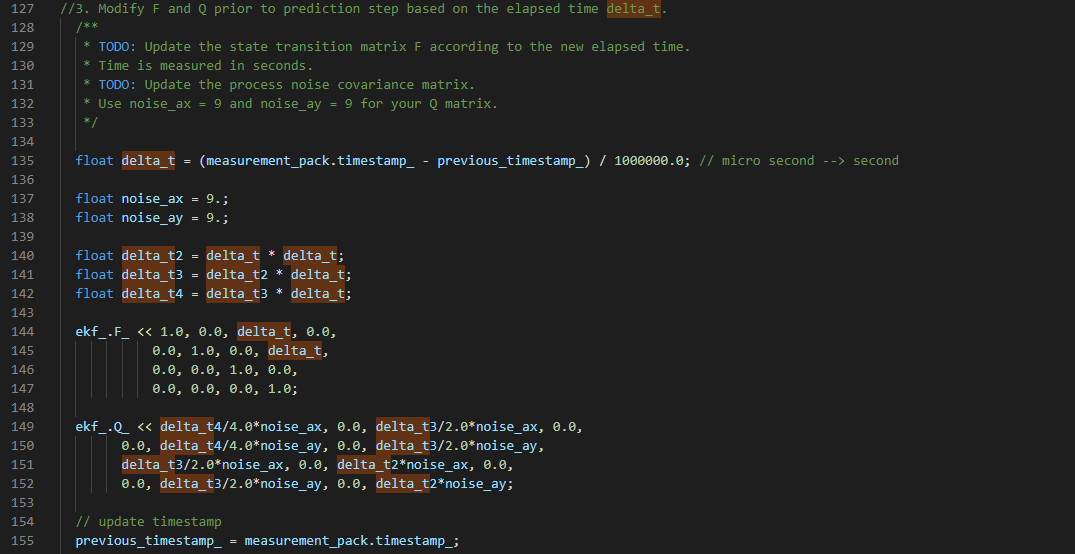
Initialization

(Laser)

Initialization

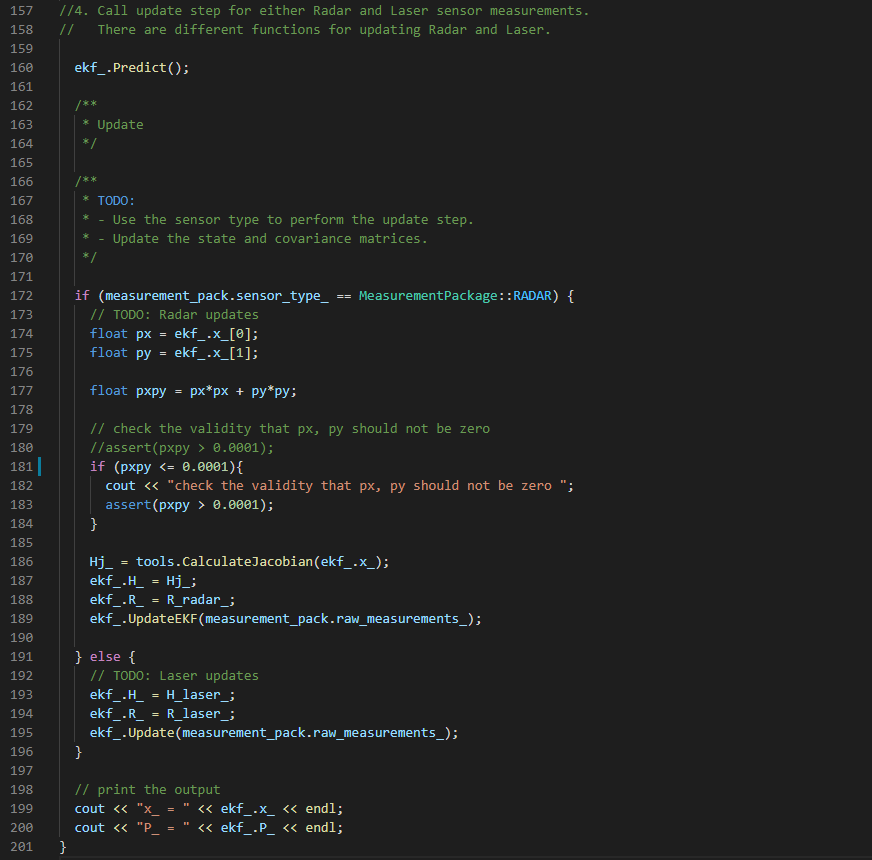
(Radar)

Next, I modify F and Q prior to prediction step based on the elapsed time delta\_t.



Finally, I call update step for either Radar or Laser sensor measurements.

To avoid division by zero when I calculate Jacobian, **I added assertion (A)**.



A

Output

(Same for Radar and Laser)

Measurement

(Laser)

Measurement

(Radar)

Prediction

(Same for Radar and Laser)

**(2) kalman\_filter.cpp**

This file starts with Initialization of x vector and Matrices (P, F, H, R, Q).

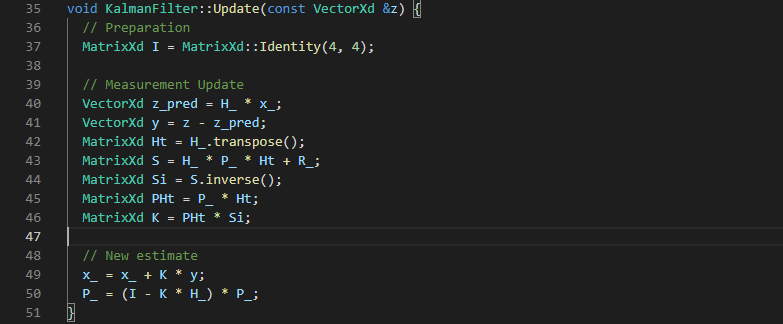
Then, prediction function is described for both Radar and Laser.



Prediction

(Same for Radar and Laser)

Then I create update function for Laser.



Measurement

(Laser)

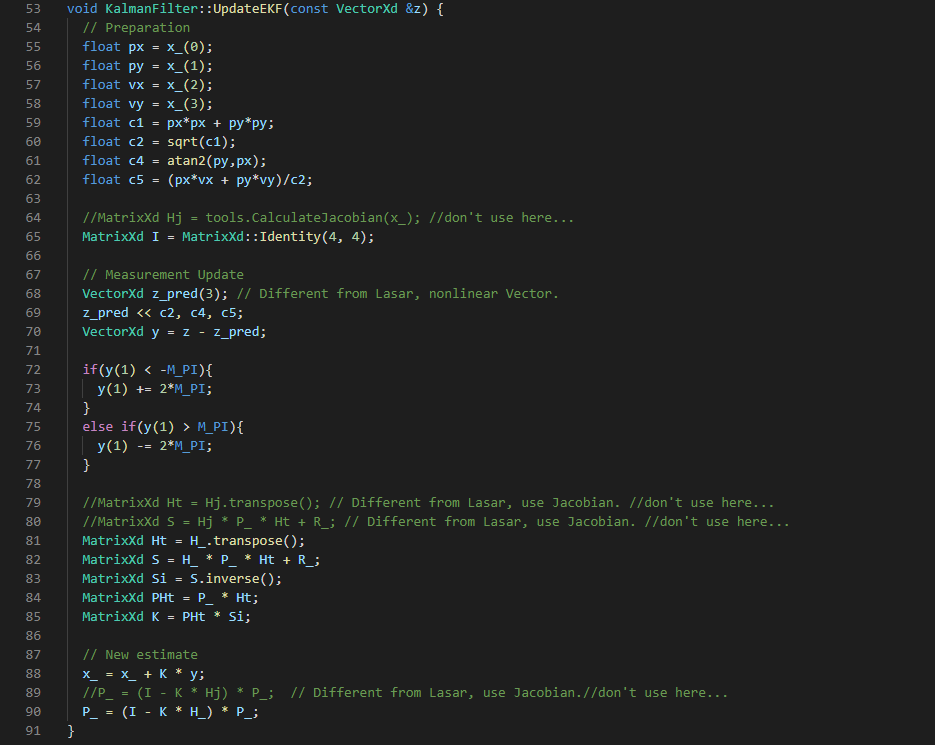
Finally, I create update function for Radar.

Radar measurement model is not linear, so **I calculated z\_pred not using H matrix (A)**.

Angle phi should be between -PI and +PI, so **I added calculation (B)**.

Measurement

(Radar)



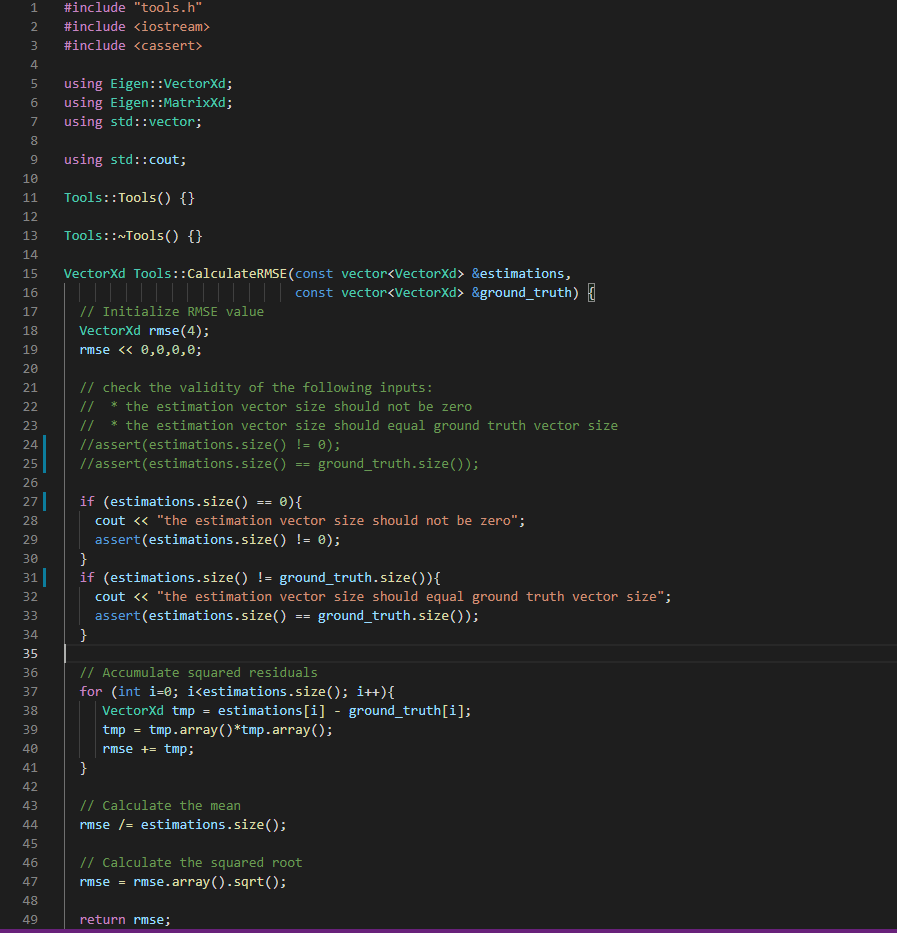
B

A

**(3) tools.cpp**

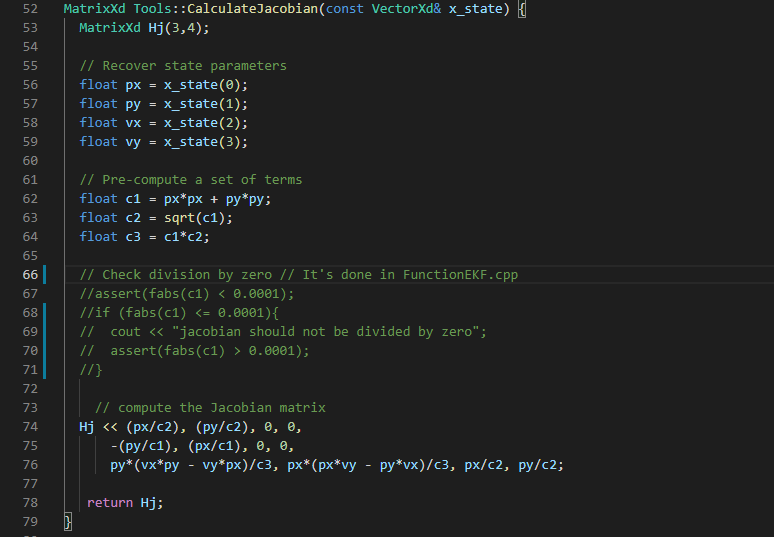
The following function calculates RMSE, which is used in main.cpp.

To check the validity of the inputs, **I added assertion (A)**.



A

The following function calculates Jacobian, which is used in FunctionEKF.cpp.



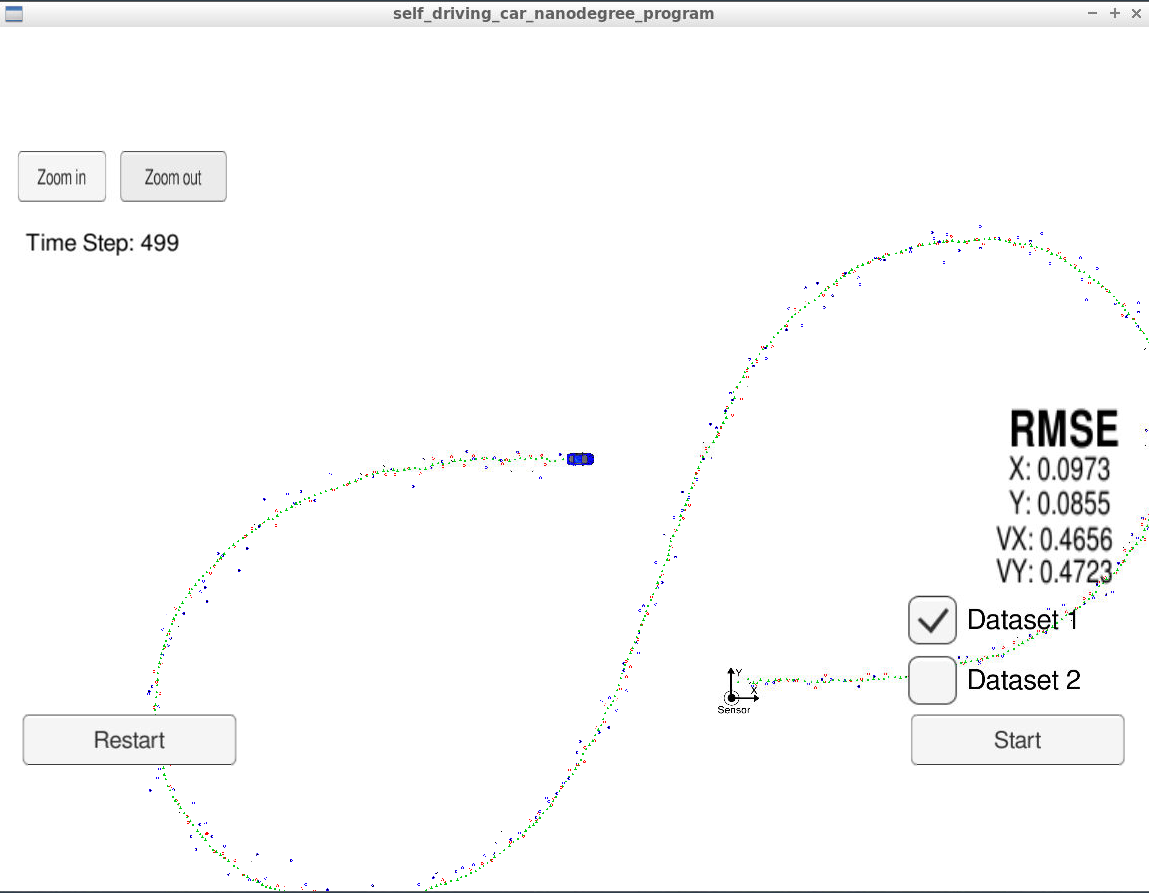
**2. Result**

I show the result below.

It achieved the target performance of RMSE in Project Rubric.

RMSE\_x = 0.0973 < 0.11, RMSE\_y = 0.0855 < 0.11

RMSE\_vx = 0.4656 < 0.52, RMSE\_vy = 0.4723 < 0.52



However, the **measurement positions of Radar** are less accurate than **Laser**, so sometimes **estimated positions become unstable**. I assume that the accuracy will be improved if I neglect the measurement positions that deviate a lot from the estimation positions.

