Kalman Filter Project

Objective

The project implemented Kalman Filter fusion sensor data from Laser and Radar sensors with C++ implementation.

Algorithm & Implementation

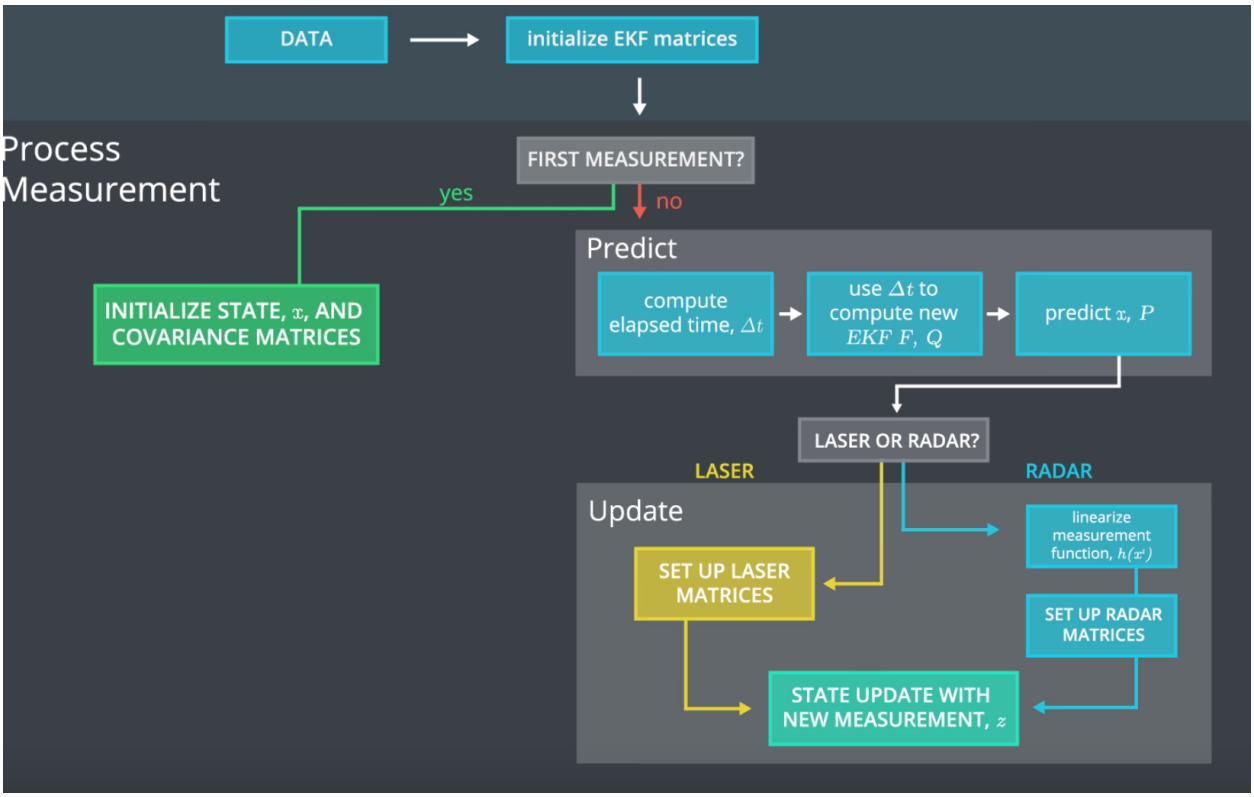

Fig. 1 Sensor Fusion General Flow

Fig. 1 shows sensor fusion general flow.
In main.cpp, it parses command arguments for input file, output file, optional sensor (both, laser, or radar), and optional process ax/ay variances.
Then from input data file, we get sensor data and group truth values.

An instance of `FusionEKF` is created to process measurement for each sensor data. Radar data are in polar coordination. After all data are processed, I use `CalculateRMSE()` function from Tools to calculate RMSE between estimation and the ground truth.

FusionEKF is the main class to handle Extended Kalman Filter. It has a Kalman Filter sub-component, Covariance Matrix, initialization flag, timestamp, and process noise variances. ProcessMeasurement function is the primary function to run the flow as shown in Fig. 1. For efficiency and accuracy, I choose to pass both raw measurement Radar data in polar coordinate and converted Cartesian coordinate data to Extended Kalman Filter update.

KalmanFilter class implements Kalman Filter with constructor, Predict(), Update() for Laser, and Extended Kalman Filter update UpdateEKF() for Radar. Constructor simply initializes each matrix with corresponding dimensions. Predict() and Update() are straight-forward implementation from math formula. UpdateEKF() needs to handle divide-by-zero cases. I use 0.0001 for calculation in divide-by-zero for Jacobian Matrix. For efficiency, I calculate Jacobian Matrix in UpdateEKF() instead of Tools. I use std::atan2 to calculate radian which limits it to pi and –pi. For other calculation, I normalize radian value to the range between pi and –pi.

Results

> ./ExtendedKF ../data/obj_pose-laser-radar-synthetic-input.txt ../data/out1.txt
The above command reads data from ../data/obj_pose-laser-radar-synthetic-input.txt and output data in ../data/out1.txt.

./ExtendedKF ../data/obj_pose-laser-radar-synthetic-input.txt ../data/laser.txt l
With an additional parameter “l” at the end, it processes only Laser data.

./ExtendedKF ../data/obj_pose-laser-radar-synthetic-input.txt ../data/radar.txt r
With an additional parameter “r” at the end, it processes only Laser data.

Table 1 shows RMSE accuracy when run with fusion of both sensors, Laser only, and Radar only. Laser has better accuracy than Radar in general.

RMSE	x	y	Vx	Vy
Both	**0.096198**	**0.0852897**	**0.413292**	**0.480286**
Laser Only	**0.122191**	**0.0983799**	**0.582513**	**0.456699**
Radar Only	**0.197623**	**0.264278**	**0.456697**	**0.679961**

Table 1. RMSE Accuracy from different sensors

Float or Double
When we use double instead float for variables, it only improves marginally. For example, y RMSE accuracy is **0.0983798** instead of **0.0983799.** For this project, double has the same result as float in calculation.

Variances Ax/Ay on RMSE

Process variances Ax and Ay impact RMSE of x, y, Vx, and Vy.
We were given 9.0 for both Ax and Ay, which are good values.
Fig. 2 shows the chart with Ax/Ay from 1.0 to 15.0.
Table 1 shows it in a table format.

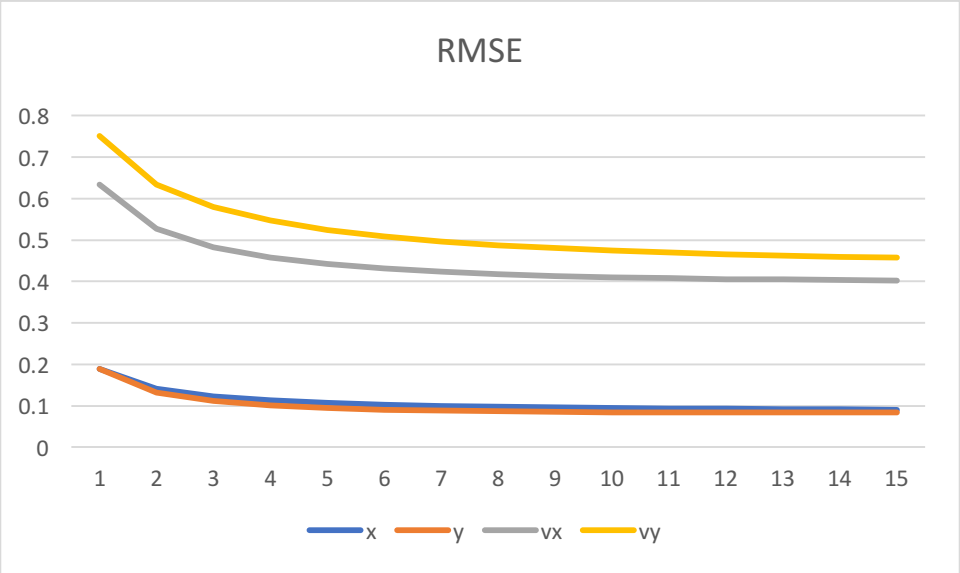

Fig. 2 RMSE of x, y, Vx, Vy for Process Variance from 1.0 to 15.0

Ax/Ay	x	y	vx	vy
1	0.18885	0.189258	0.634104	0.751234
2	0.140932	0.132366	0.527852	0.633679
3	0.122774	0.111117	0.482639	0.579296
4	0.113137	0.100427	0.457432	0.546703
5	0.107157	0.0942843	0.441519	0.524723
6	0.103088	0.0904832	0.430729	0.508864
7	0.100148	0.0880232	0.423071	0.496914
8	0.0979283	0.0863887	0.417471	0.487639
9	0.096198	0.0852897	0.413292	0.480286
10	0.0948148	0.0845521	0.410134	0.474367
11	0.0936866	0.0840658	0.407731	0.469552
12	0.0927514	0.0837589	0.405903	0.465604
13	0.0919656	0.0835823	0.404518	0.462352
14	0.0912977	0.0835021	0.403484	0.459667
15	0.0907246	0.0834939	0.402729	0.457451

Table 2 RMSE of x, y, Vx, Vy for Process Variance from 1.0 to 15.0

Future

Linear approximation with Jacobian Matrix in Kalman Filter might not be good for some cases. Unscented Kalman Filter (UKF) might be better choice.

For some Asian cities with lots of motorcycles, whether UKF can handle multiple targets in both speed and accuracy can be one of the deciding factors for self-driving car in those regions.