Self-driving car Nanodegree(Term 2). Project 1: Extended Kalman Filter

Ruggero Vasile

July 11, 2017

Abstract

This is a short report for the Extended Kalman Filter project for self-driving car nanodegree program. I will focus on the results of the application, and briefly describe the code overview as well as limitation of the approach.

1 Introduction

The request for this project was to code an Extended Kalman Filter algorithm which would be able to process measurement data from a simulator as well as from input files. We are limited to two kind of measurement apparata, the Lidar and the Radar as well as to a linear motion model. These limitations are imposed to introduce the student to the field of Kalman filtering and will be partially relaxed in the following project.

Another interesting challenge was to code in C++ instead of Python. Apart from the advantage of knowing another programming language, C++ is much more efficient and performant, being as well one of the main languages used by the automotive companies for their devices.

The report is organized as follows. In the first section I will introduce the problem, the limitations involved and possible solutions. In the second section I will briefly discuss about the code and parameters available. In the last section I will present my results.

2 Extended Kalman Filtering with Radar and Lidar

A Kalman filter is an optimal dynamical system state estimator in a purely Gaussian world. This means that theoretically it can be applied consistently when the initial state space distribution is a multivariate Gaussian, the noise processes involved in the dynamics and in the measurements are Gaussian, the dynamical system involved is linear and the function used for performance evaluation is the Root Mean Squared Error (RMSE). Under these assumptions the

Kalman filter equations provide the best possible state estimation under dynamics and repetive measurements. These conditions are satisfied in our case only when we consider measurement coming from the Lidar device. Under the assumptions of the project, in fact, the dynamics is taken to be linear in the state variables, and the Lidar directly measures the cartesian coordinates of the object to be detected. While the assumption on the linear measurements is quite good, the dynamical model used has strong limitations since it assumes that in the interval between two consecutive measurements the object moves with constant velocity. When the measurements are very frequent and precise enough, this model behaves well.

However in this project we also deal with measurements coming from another device: the Radar. The radar has two main disadvantages in respect to the lidar:

- The spatial resolution is much lower and physically limited by the wavelenght of the device.
- It does not measure directly the position in cartesian coordinates, instead it measures coordinates in polar coordinates as well as the radial velocity through the Doppler effect.

The second disadvantage implies that Kalman Filtering is not usable in its current form because the measurement process is not linear in the state variables. The Extended Kalman filter, through a linearization of the measurement operation, comes to our help and allows to solve the problem numerically.

3 C++ code description

The code I used consists of the following parts:

- A main function: This function is used to communicate with a host, in this case the simulator. Its main role is to sequentially ask the host for new data and return the result of the operation. The data consists in real state variables values and measured variables. Every time new data is arrived and processed, new data will be available from the host and the same loop will be ran until no more data is available.
- A SensorFusion class instance: This is where the state estimation problem is tackled. I imagine it as a blackbox which, once initialized with an initial state, receives new data from either Radar or Lidar measurement, and outputs the new estimation of the state.
- A DynamicalModel class which contains the information about the dynamics, i.e. the deterministic part under the form of a matrix and the acceleration noise.
- A MeasurementModel class: it can be both a Radar derived class or a Lidar derived Class and contains information on the measurement noise, and the measurement matrices. The Lidar measures only the cartesian

coordinates, the radar measures the polar coordinates and the radial velocity.

- A EKF class which applies a round of the Extended Kalman Filter algorithms, i.e. propagates the state according to the dynamical model, evaluates the residual error, and updates accordingly the system state mean and covariances.

The algorithm is quite rigid and only few parameters may be tuned:

- Acceleration noise variances.
- Radar and Lidar measurement noise variances. These depend on the manufacturer of the measurement devices therefore I decided not to tweak them and to fix their values to those suggested in the lessons.
- The initial state mean and covariances: the choice of these parameters can be done using the information on the first measurement. However both Radar and Lidar single measurement do not contain provide enough to estimate the velocity of the object. I decide therefore to initialize the position mean to the estimated values while initialize to zero the velocities. The covariance is chosen to be diagoanal with value 1. The value 1 is compatible with the error measurement on the position for both Lidar and Radar.
- Choise of measurement device: Radar and Lidar measurements come one after another. We can decide to use both for state estimation or turning one off and try to estimate the state only with one device.

4 Results

Unless specified otherwise I fix the parameters in this section to: $noise_{ax} = noise_{ay} = 9.0$, $noise_lidar = 0.0225$, $noise_radar_radius = 0.09$, $noise_radar_angle = 0.0009$. These are the values suggested in the lectures.

I first analyzed the error measurement distribution of the two devices, i.e. the distribution of the differences $x_{real} - x_{meas}$ and $y_{real} - y_{meas}$ for Lidar and Radar separately. This gives an idea about their reliability. In Fig. 1 I show the results for Lidar and Radar and we immediately see that in the measurement of the position the Lidar is much more accurate than the Radar. We can expect therefore that estimation of position through Lidar should give better results than that with Radar only. As can be seen in Fig. 2 this is actually the case. Using only Lidar for state estimation we reach asymptotically a root mean squared error of about 0.11 for the coordinate variables, while using the Radar device only we reach about 0.25. The initial transient should not be considered since it depends on the initialization of the state. Its effect on the RMSE will slowly disappear as the dynamics continues. However using only the Lidar is not enough to lower the RMSEs below the limits posed by the project rubric. The

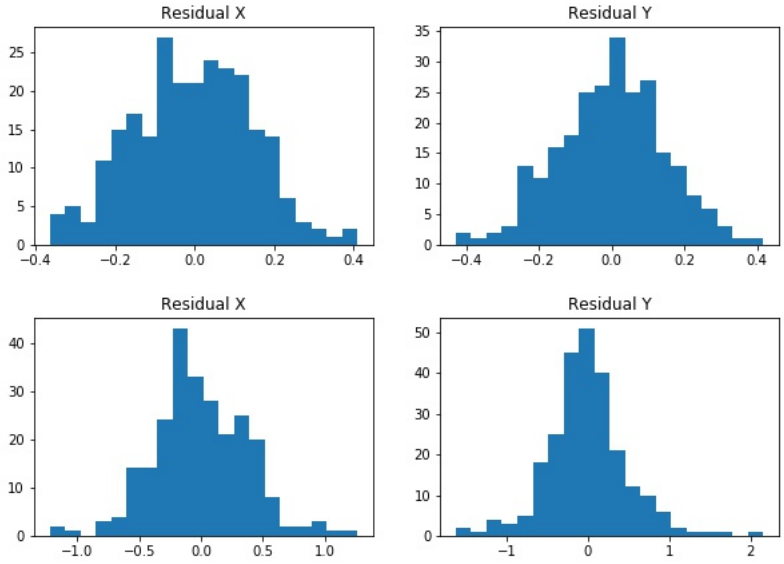

Figure 1: The distributions of the residual: actual position coordinate minus the position measured. (Left) residual for x measurement. (Right) residual for y measurement. (Above) Lidar. (Below) Radar

main reason for that is that Lidar measurements are not frequent enough, and in between the system moves according to the dynamical model which imposes a constant velocity motion. Using the Radar measurements in between should provide an improvement even if it would be better to have a higher number of Lidar measurements. We see the result of the sensor fusion model in 3 where the cumulated RMSE values go below the limit required by the rubric. Despite the Radar is not as precise as the Lidar, its usage gives an improvement in the state estimation simply because more measurements are available and the effect of the not accurate dynamical model is reduced, i.e. we rely more on the measurements than on the dynamics.

5 Conclusion

I conclude the report with some final comments:

- If we increase the error in the measurement we expect the results to get worse since we are less sure of the precision of the devices.
- The same thing cannot be said about the noise acceleration levels. If these increase the relative importance of the noise gets higher than that of the dynamical (not accurate) deterministic part. I observed, in fact, that if the noise is increased the RMSE further decrease especially for the velocity

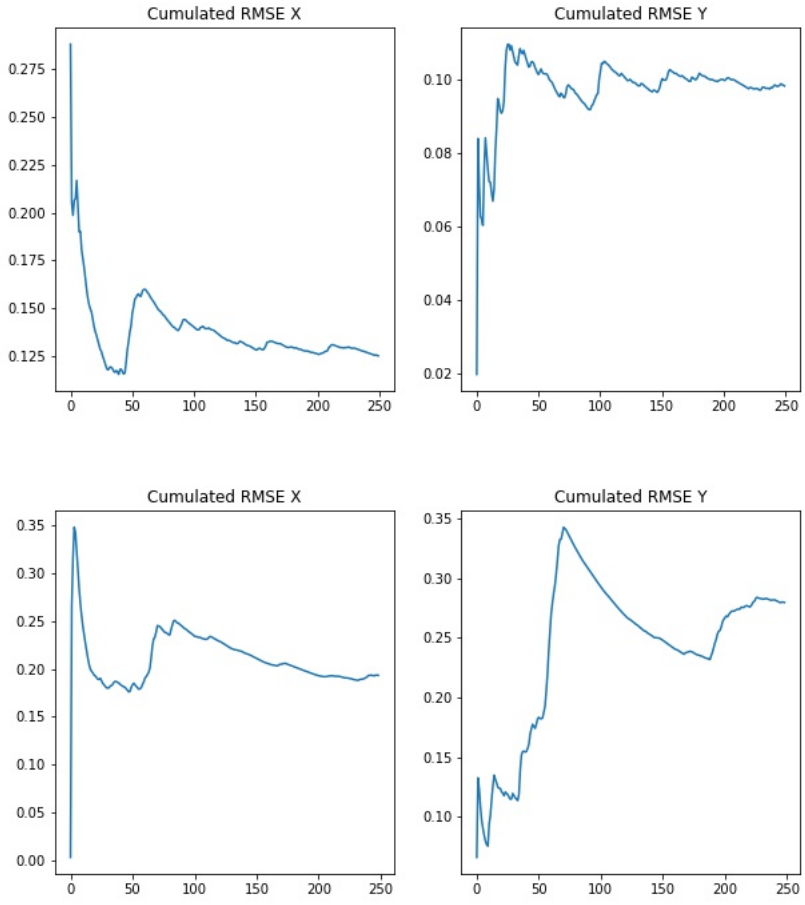

Figure 2: Cumulated RMSE for Lidar (Above) and Radar (below) for coordinates x (left) and y (right) as a function of time (arbitrary units).

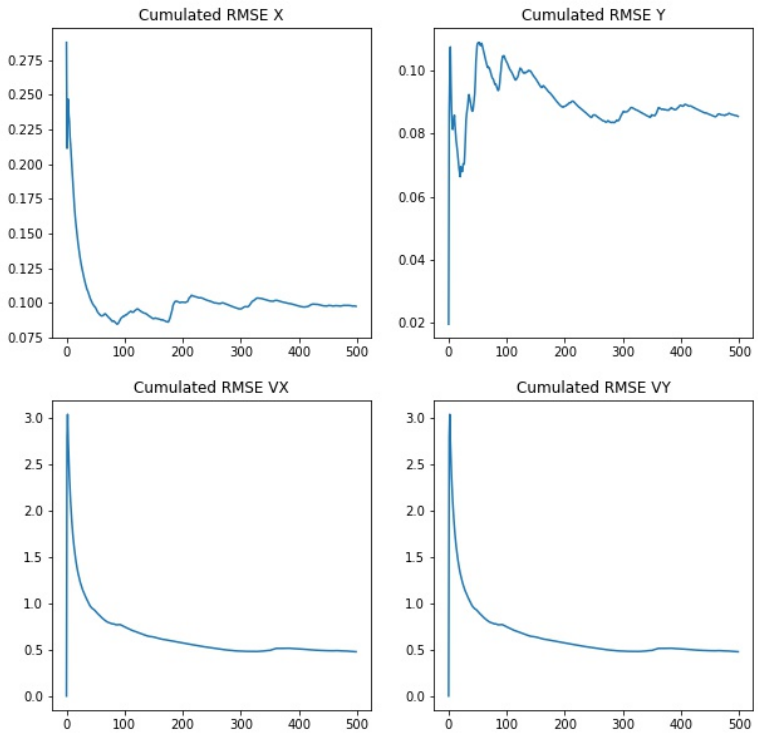

Figure 3: Cumulated RMSE using both Lidar and Radar measurements. (Above Left) RMSE for x position variable. (Above Right) RMSE for y position variable. (Below Left) RMSE for velocity in x direction. (Below Right) RMSE for velocity in y direction.

variables reaching about 0.42 and increases again if the noise is dominant. The position variables RMSE instead decrease just slightly. It seems that a noisy dynamical model works better!!

- I also tried the model on the second set of data observing very similar results. Only in the x position variable the RMSE does not reach the rubric limit of 0.11 while stopping at about 0.113 probably do to an unfortunate initialization of the state.