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

Ruggero Vasile

July 17, 2017

Abstract

This is a short report for the Uscented 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 the approach.

1 Introduction

In this project I code an unscented Kalman filter in C++. In respect to the previous project on Extended Kalman Filters we deal here also with a different motion model, the CVRT, which is capable of handling more varied object motion cases. This motion model is also non linear and therefore the extended Kalman filter solution may not work properly.

The unscented Kalman filter instead is a very general approach useful when both the motion models and measurement models are non linear. Instead of looking for an exact solution, it achieves the goal using numerical sampling methods, where state estimation is obtained by following trajectories of representative points during the dynamics and the measurement processes. The only relevant approximation of the Unscented Kalman Filter consists in assuming that at any intermediate step the state distribution is Gaussian. When the real distribution does not differ appreciably from the Gaussian distribution, the techniques provides good enough results.

Since I already described the idea behind sensor fusion in the previous report, in this notes I will go through the Code structure only and present the results in a later section.

2 Method and code overview

The Unscented Kalman Filter is a very general techniques built around three main blocks:

- A dynamical model: This model defines our prior belief of tracked object motion, therefore it consists of a state variables vector and information

about noise processes that affect the motion. In the particular case under examination, the CVRT has 5 state variables and two involved process noises, considered Gaussian distributed with zero mean. The variances of these processes are defined as σ_a and σ_ψ. The first process affects the linear motion of the object as a linear acceleration process. The second process affects the rotatory motion as an angular acceleration noice process.

- Measurement models: They define the sort of measurements that we perform on our system. In the particular case we consider Lidar and Radar measurement as in the Extended Kalman Filter cases. Both these measurements are noisy. In the Lidar case we have one single process noise of zero mean and variance $\sigma_{lidar} = 0.0225$. In the Radar case each measurement is affected by a different process noise (or measurement precision). In particular we assume that the variables ρ and $\dot{\rho}$ are affected by a noise of zero mean and variance $\sigma_\rho = 0.09$ and the angle variable θ by a noise process of zero mean and variance $\sigma_\theta = 0.0009$ as in the case of the Extended Kalman Filter. Since these variances depend on the measurement apparatus I will keep their values fixed throughout the presentation.

- Dynamical-Measurement model: this describes the interaction between the measurement models and the dynamical state variables. Specifically they are rules that defines how the measurement devices measure the state variables. For instance the Lidar will measure directly (and linearly) the position variables x and y of the $CVRT$ model. The radar instead has a more involved interaction with the $CVRT$ as explained in the lectures.

Given the previous building blocks the Unscented Kalman Filter consists of a cyclical dynamics of predict-update steps. During the predict step, given an input state distribution (Gaussian), we evaluate the final state distribution according to the $CVRT$ model. This is done using a sampling approach through the sigma points. Given the predicted distribution, we incorporate the measurements by evaluating the residual between the actual measurement variables and the ideal model measurement estimation. The residual will be then employed through the Kalman Filter equations to evaluate the correction to the dynamical system state due to the inconsistence between the actual measurements and the measurement model estimations.

The C++ code produced tries to keep the above conceptual distibutions in code.

- The *DynamicalModel* base class and the derived $CVRT$ class contain all the information about the system dynamics. In principle new derived classes using more complicated dynamical model can be added in order to consider more general cases.

- The *measurement* base class and the derived Lidar and Radar classes contain information and function regarding the measurement processes.

- The *PointGenerator* class is an helper class that deals with the generation and storage of the sigma points throught the predict-update steps. It can

interact with the dynamical model and measurement models, when these require the utilization of the sigma point for state estimation.

- The *BayesianFilter* class and its derived *UKF* define the specific steps to take for the application of the filter. In principle a derived *EFK* classes could be added accordingly to accomodate the Extended Kalman Filter as well.
- The *SensorFusion* class is a template class that depends on a particular dynamical model and filter type. At the moment it is not possible to extend it to measurements different from the Lidar and Radar. This class is instantiated in the main function. When new measurements arrive, the *ProcessMeasurement* function is called to apply the predict-update steps of the current filter type.

Finally I mention the *Tools* class that contain function to evaluate for instance the root mean squared error (RMSE), to perform consistency analysis, to print results on file etc... . Also a *constants.h* file is provided to change some parameters of the filter and the models.

3 Results

As in the case of the Extended Kalman Filter I consider here cases in which only one measurement type is performed (Lidar or Radar) as well as the sensor fusion case. The only parameters to optimize are the dynamical model noise variances σ_a and σ_{psi}. Their values need to be chosen considering the actual motion expected, i.e. since they are acceleration process noises, the expected acceleration values during the system dynamics. After some trial and error I chose $\sigma_a = 1.0$ and $\sigma_\psi = 0.1$. In Fig. 1 I show the result when only Lidar measurement are used, while in Fig. 2 I show those with only radar measurements. In all panels the project rubric threshold are shown. We can see that in both cases not all the RMSE values go below the thresholds, showing that single measurements are not helpful. Better results could be possibly achieved using only Lidar measurements with a higher frequency rate. Instead in Fig. 3 I show how using both Lidar and Radar measurements the situation is much more favourable. Also while precision in the estimation of the position variables x and y does not change appreciably in respect to the Extended Kalman Filter cases, a solid improvement for the velocity variables is instead achieved. This is to be attributed to the choice of the $CVRT$ dynamical model, but also to the use of radar measurement which give some information on the velocity of the system as well.

Finally in Fig. 4 I show a consistency check using the NIS distribution. Since the Lidar and Radar measure a different number of variables I evaluate a normalized NIS using the different number of degrees of freedom (2 for the Lidar and 3 for the Radar). The figure shows that the filter is quite consistence with the dynamics, i.e. the choices of parameters are generating predictive distribution in agreement with the measurement values.

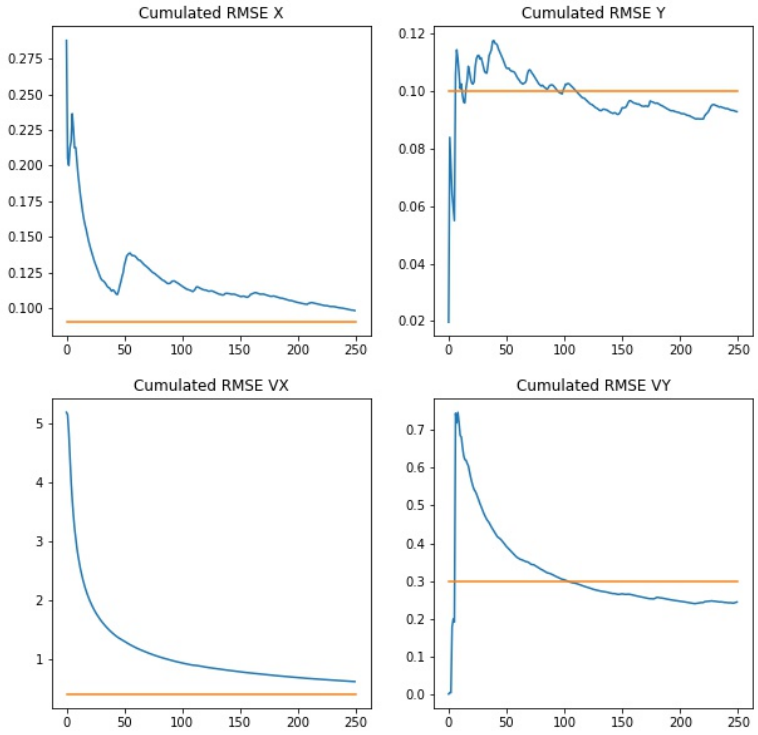

Figure 1: Cumulated RMSE using Lidar measurements for the position and velocity variables.

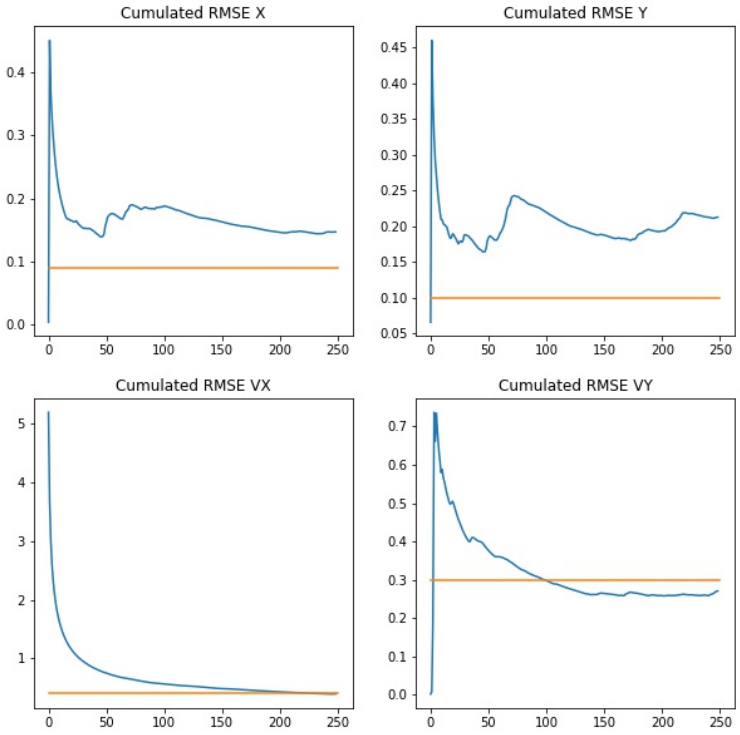

Figure 2: Cumulated RMSE using Radar measurements for the position and velocity variables.

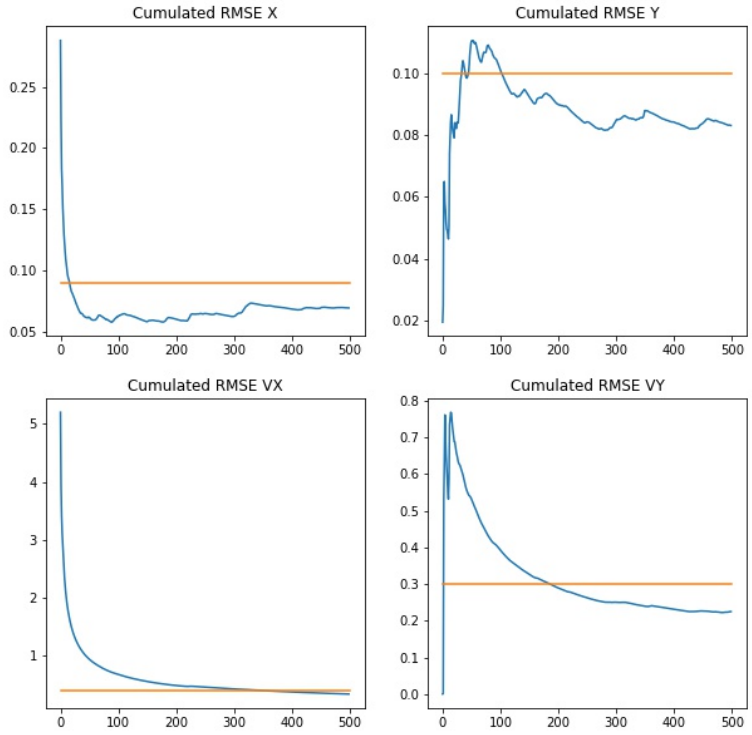

Figure 3: Cumulated RMSE using Lidar and Radar measurements for the position and velocity variables.

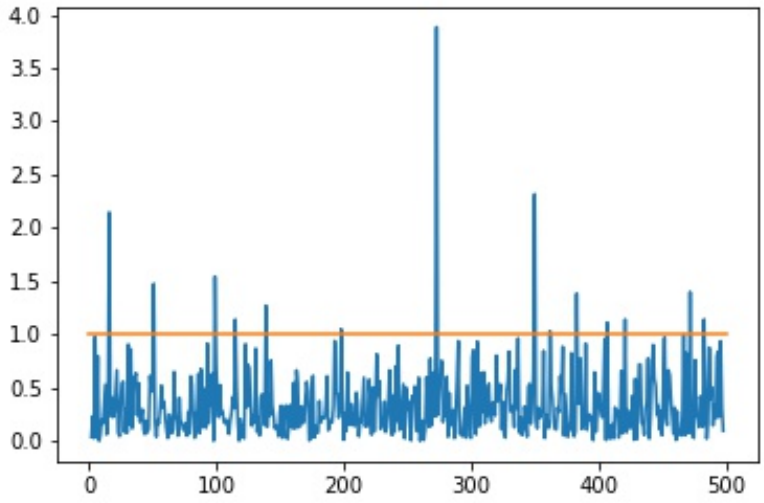

Figure 4: Consistency check using NIS distribution for the sensor fusion state estimation case.