Unscented Kalman Filters project report

Introduction

The project was set using Eclipse CDT IDE. The following paragraphs comment on the various aspects of the project rubric.

Compiling

Your code should compile

The cmake and make commands were executed in windows according to the project instructions and the code is compiling successfully. No changes were made in the CMakeLists.txt file.

Accuracy

For the new data set, your algorithm will be run against "obj_pose-laser-radar-synthetic-input.txt". We'll collect the positions that your algorithm outputs and compare them to ground truth data. Your px, py, vx, and vy RMSE should be less than or equal to the values [0.09, 0.10, 0.40, 0.30].

In this project, the **obj_pose-laser-radar-synthetic-input** measurement file is processed. The UKF code is implemented in the **UKF.cpp** file.

The achieved RMSE using both sensors is **[0.076, 0.086, 0.370, 0.290]**, which lies within the required limit. Some of the parameters have been modified to more realistic values, such as for example the standard deviation of the laser. Also, covariance matrix P was initialized to a unity matrix, apart from one parameter which was changed randomly to achieve the desired RMSE.

Figure 1 is showing the path under investigation. The estimated red line is achieving fairly accurate detection of the vehicle, despite the fact that the measurement is noisy at many points. This proves that the applied CTRV model is suitable when the path has turns.

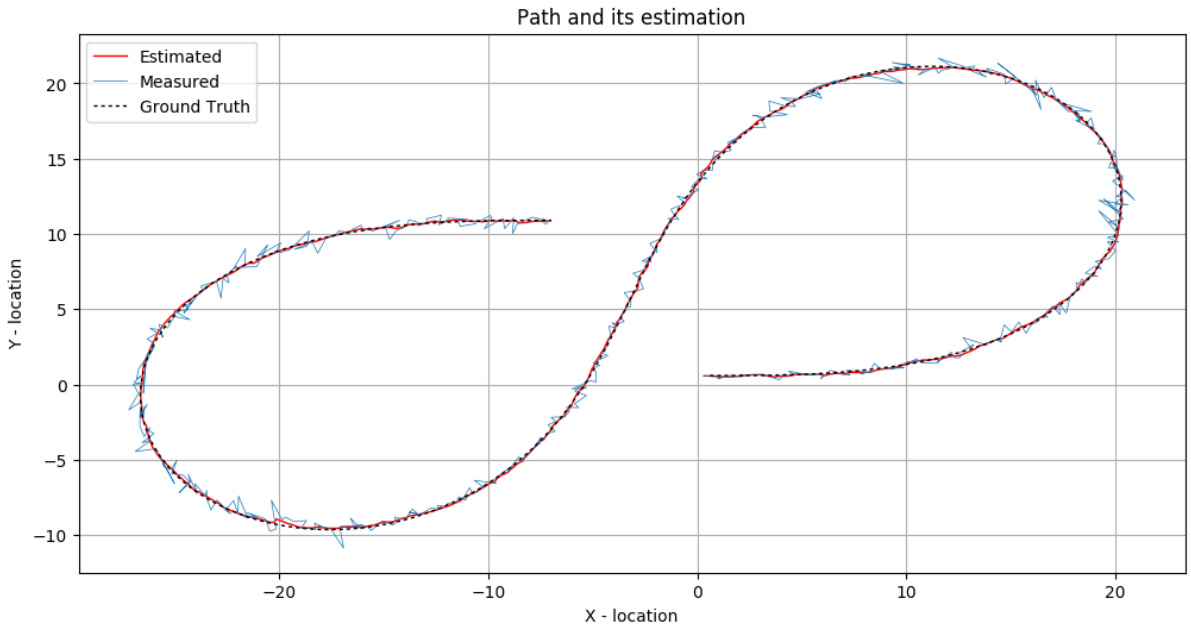

Figure 1: Path under investigation

Figure 2 depicts the estimated velocity of the model vs the ground truth one. Despite the spiky behaviour due to the uncertainty in the measurement, the trend is predicted correctly. Also, the yaw angle plot in Figure 3 and the yaw rate in Figure 4, are successfully estimating the ground truth state.

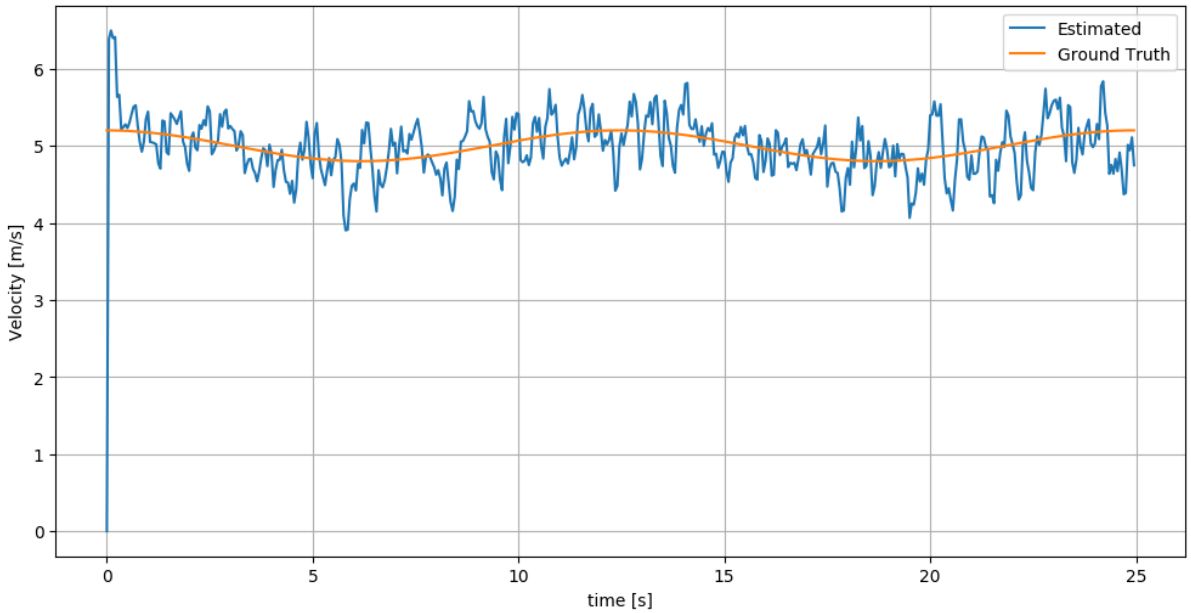

Figure 2: Estimated vs ground truth velocity

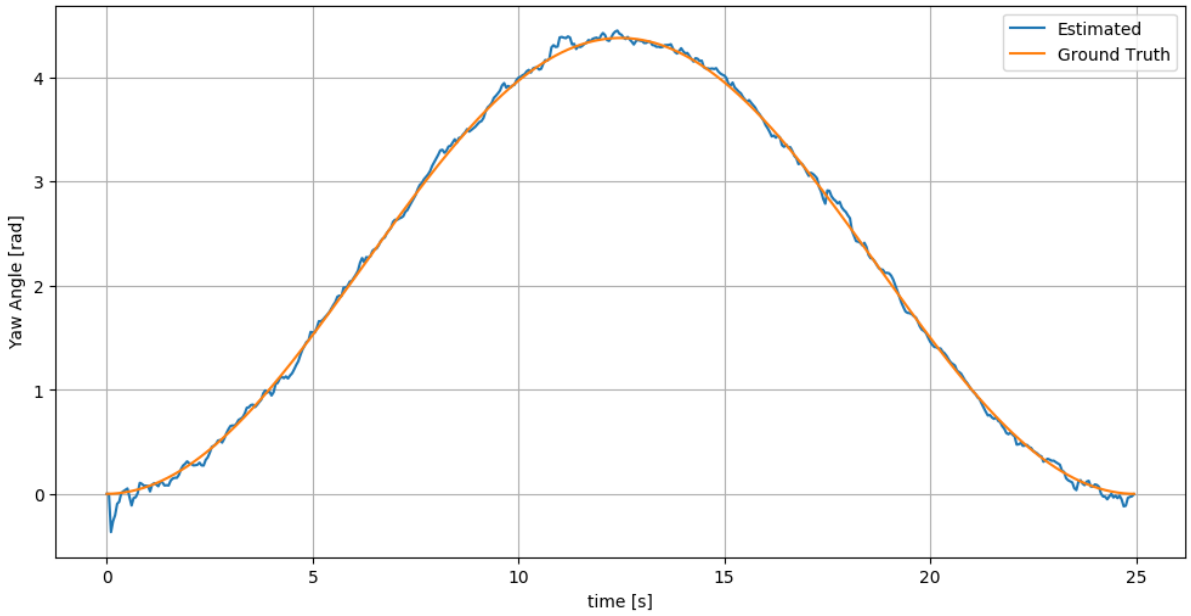

Figure 3: Estimated vs ground truth yaw angle

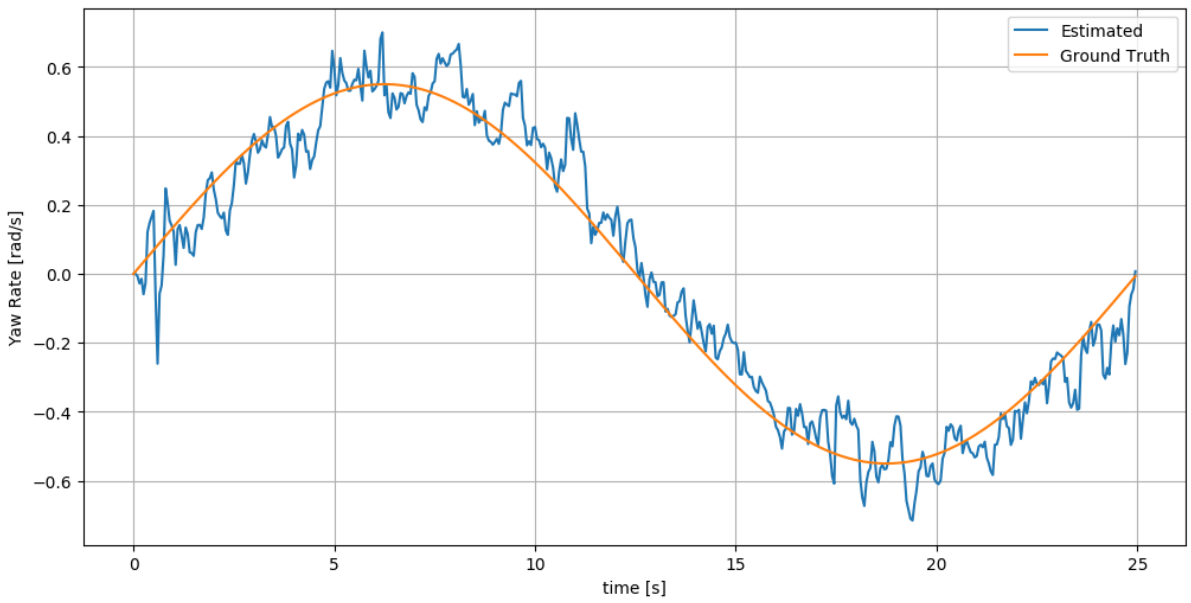

Figure 4: Estimated vs ground truth Yaw rate

In Figure 5 the RMSE performance is shown, when Laser and Radar sensors operate individually or combined. The Laser is more accurate for position detection while for velocity the two sensors provide similar values. When combined though, the improvement is remarkable and within the limits.

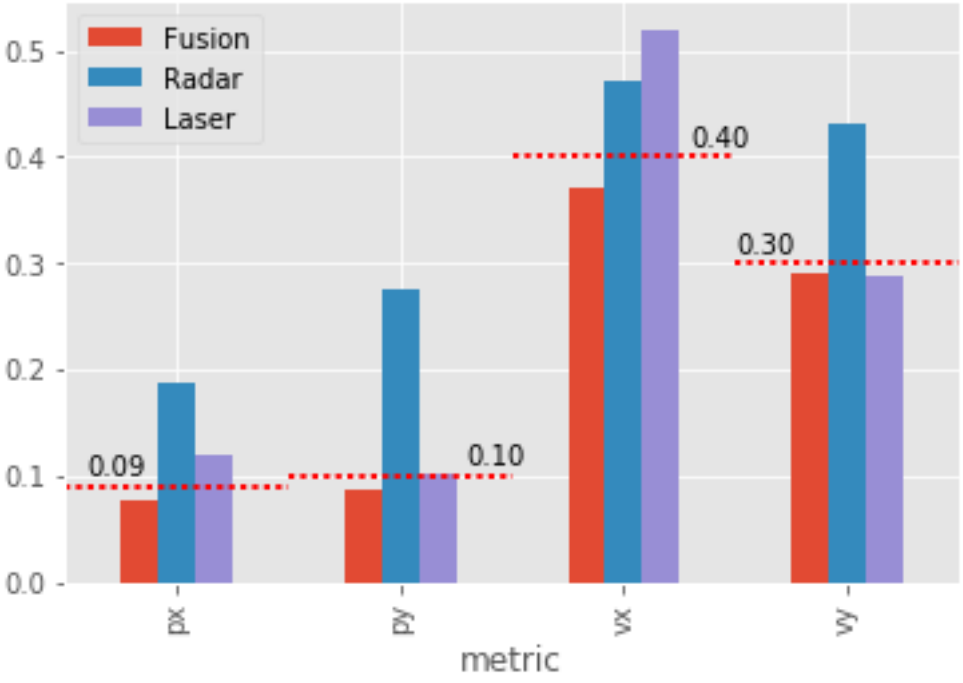

Figure 5: RMSE performance for individual sensor and for sensor fusion. Limits are shown as red dashed lines.

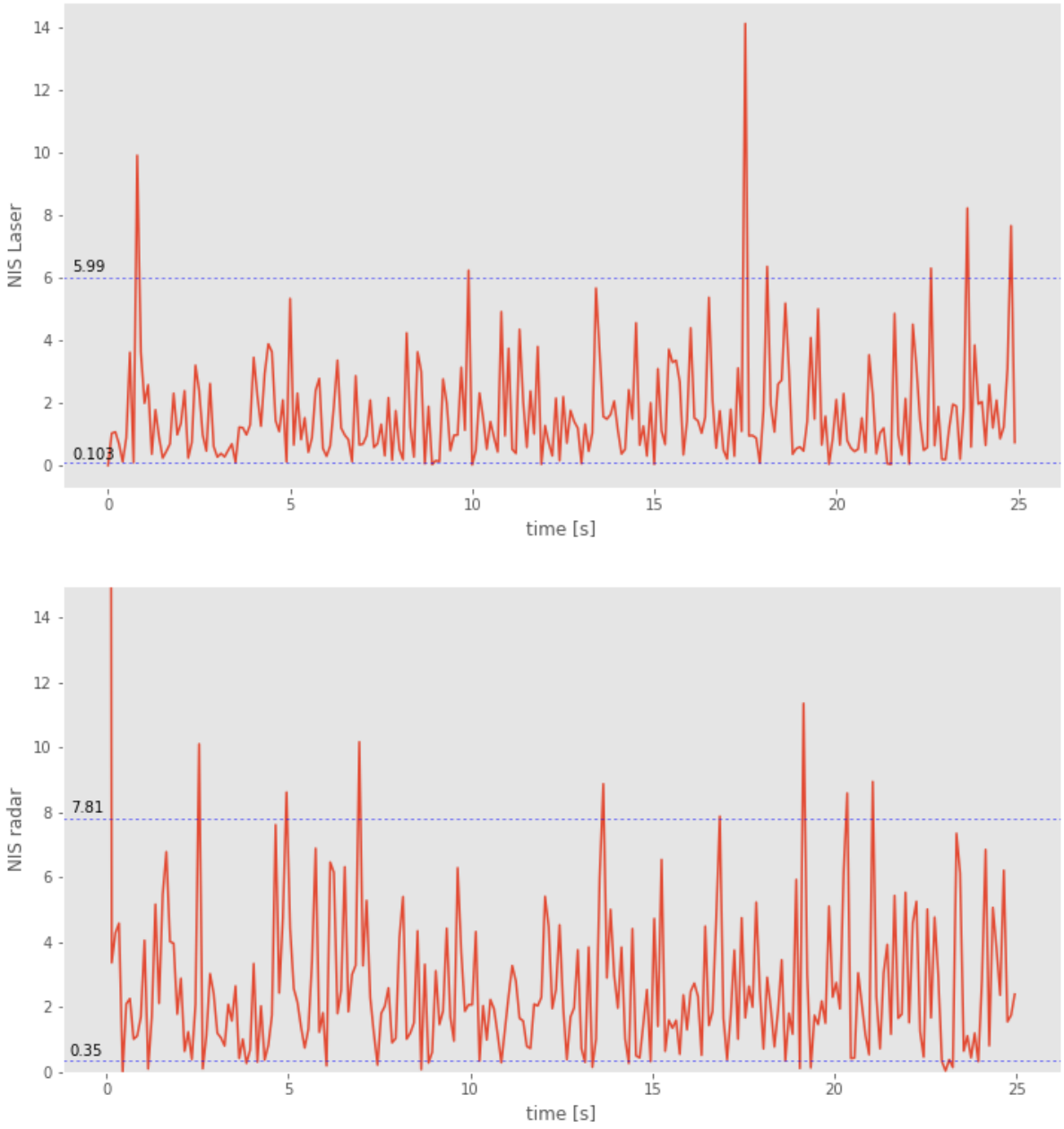

Figure 6: NIS for Laser and Radar

Finally, the NIS of the sensors, showing whether the model overestimates or underestimates the position are shown in Figure 6. The limits are different because the laser has 2 degrees of freedom and the radar 3. It can be visually verified that the model is balanced as the NIS values lie within the limits almost the entire time and there is no tendency towards the one or the other limit.

Follows the Correct Algorithm

Your Sensor Fusion algorithm follows the general processing flow as taught in the preceding lessons

Your Kalman Filter algorithm handles the first measurements appropriately

Your Kalman Filter algorithm first predicts then updates

Your Kalman Filter can handle radar and lidar measurements

The implementation shown in the **ukf.cpp** file is following the flow taught in the lessons and satisfies all the above requirements.

Code Efficiency

Your algorithm should avoid unnecessary calculations

In the implementation, unnecessary calculations were avoided.

Further investigation

The UKF implementation is compared to the EKF one, which was the subject of the previous project. The RMSEs are shown in the following table for the latest measurement data. The superiority of the UKF approach is quite emphatic.

RMSE	EKF	UKF
$\mathbf{p_x}$	0.170	0.076
$\mathbf{p_y}$	0.665	0.086
$\mathbf{u_x}$	0.625	0.370
$\mathbf{u_y}$	1.610	0.290