Unscented Kalman Filter

Objective

Use sensor data from LIDAR and RADAR for object sensor fusion and tracking using the UKF algorithm. Each estimation, displayed with green triangle is compared to the ground truth trajectory and the error is displayed using RMSE.

Project Dependencies and Environment

1. cmake >= 3.5
2. make >= 4.1
    - Linux: make is installed by default on most Linux distros
    - Mac: install Xcode command line tools to get make
    - Windows: Click here for installation instructions

3. gcc/g++ >= 5.4
    - Linux: gcc / g++ is installed by default on most Linux distros
    - Mac: same deal as make - [install Xcode command line tools]((https://developer.apple.com/xcode/features/)
    - Windows: recommend using MinGW
4. Eigen library

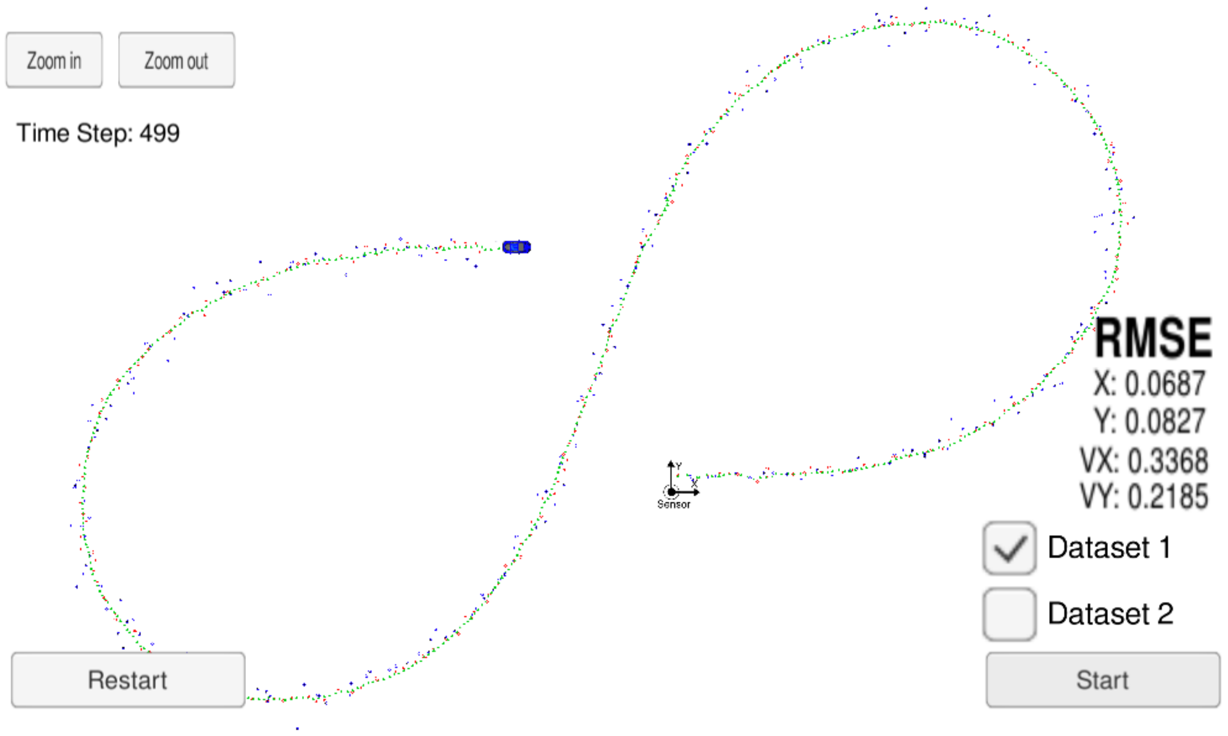
Content of folder

- Src – the folder where the sources are stored.
- CMakeLists.txt the cmake list file
- UKF_readme.pdf – short project documentation
- Install-ubuntu.sh – installation script with the necessary libraries

How to run the code (on Linux)

1. Make a build directory: mkdir build && cd build
2. Compile: cmake .. && make
3. Run it by entering ./UnscentedKF
4. Run the simulator ./term2_sim.x86_64
5. Select desired Graphics
6. Select EKF and UKF project
7. Press Start

Results



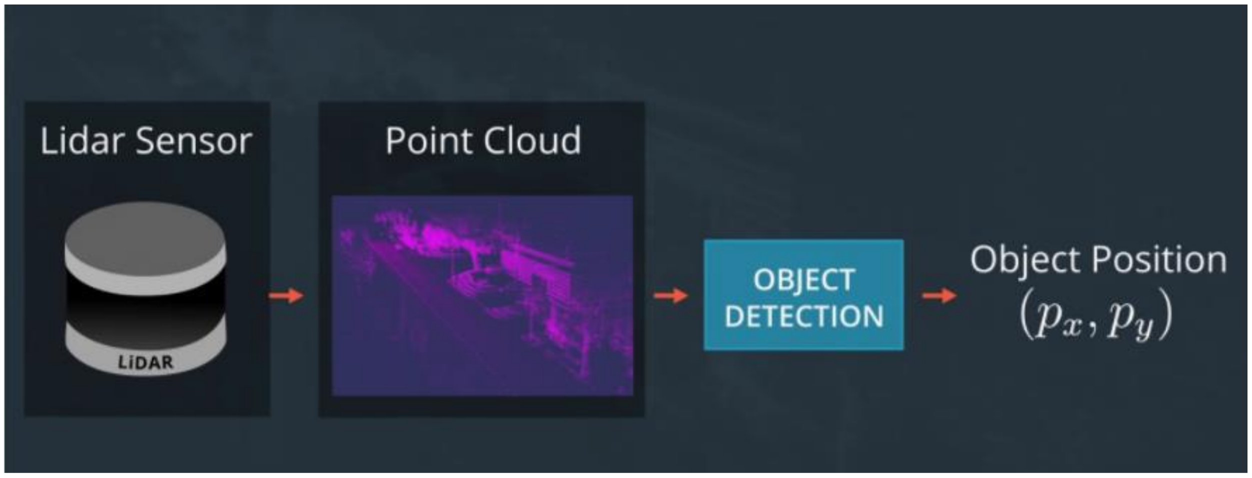
RMSE
X: 0.0687
Y: 0.0827
VX: 0.3368
VY: 0.2185
[✓ Dataset 1]
Dataset 2

	UKF Fused	UKF Radar	UKF Lidar	EKF fused	Minim. Req.
X	0.06	0.24	0.17	0.09	0.09
Y	0.08	0.30	0.14	0.08	0.10
VX	0.33	0.68	0.61	0.45	0.40
VY	0.21	0.59	0.26	0.44	0.30

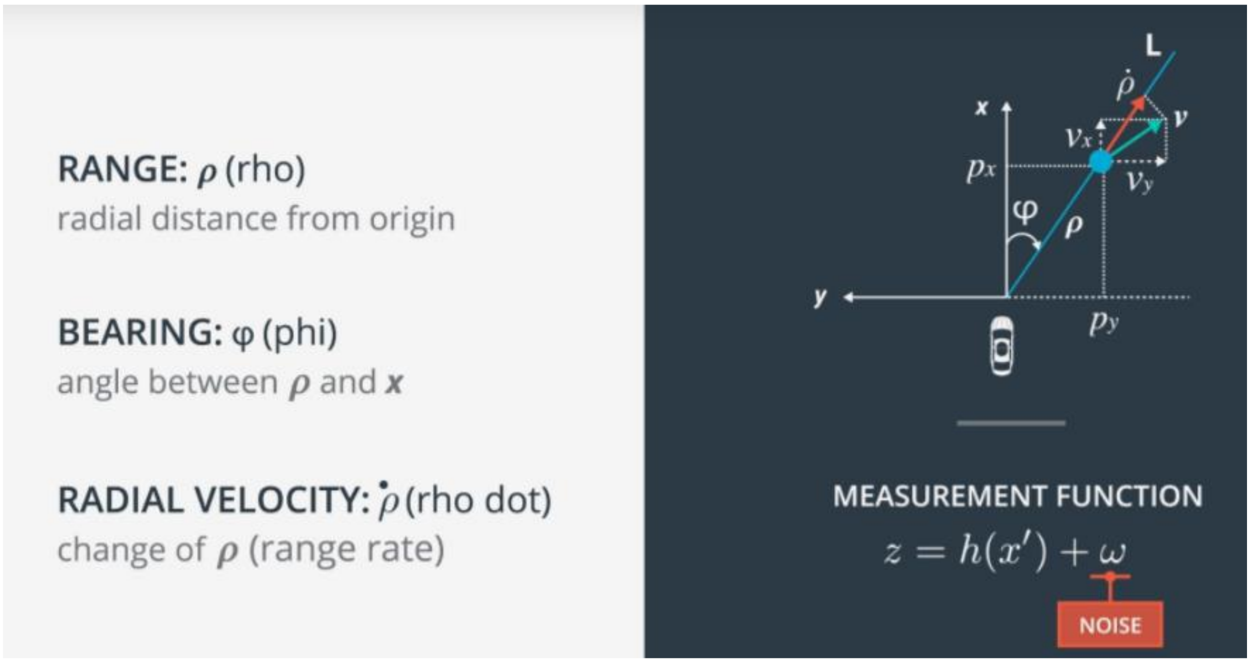
From the obtained RMSE data we observe that the radar measurements are noisier. The UKF is capable of utilizing both RADAR and LIDAR measurements to provide a more robust estimate and overall, better results than each individual sensor. We can also see that due to the fact that the unscented Kaman filer is using sigma points to approximate the result of the nonlinear function (and a non –linear motion model), the results are better than what the EKF can offer.

About the sensors

LIDAR sensors can detect static objects and are less sensitive to weather and illumination conditions, however they have a very high purchasing cost. The prices of LIDARs rises with the number of scanning layers. Some LIDAR sensors can offer as output point clouds or objects (depending on how they are set to give the output). Generally we get x,y,z information for each scanned point. For the case of the project the x and y coordinates are enough.



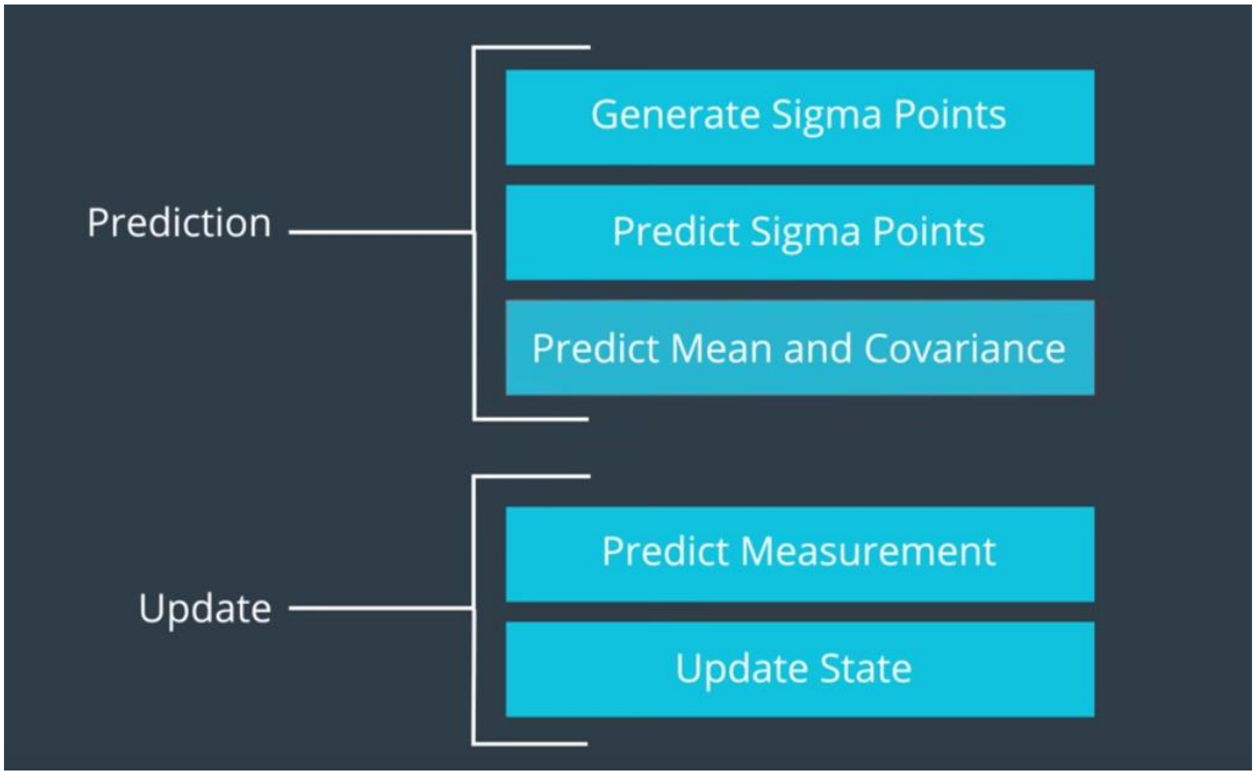
RADAR sensors are great at detecting moving objects made out of metal material, however they fail to detect items made of porous plastic or wood. Even more, RADARs usually have a narrow field of view, and to compensate for this issue they are usually used in arrays that slightly overlap in order to obtain larger fields of view. One of the biggest disadvantages of RADARs is that they omit objects in order, not to over-report. With the raw radar information we can determine the distance and speed of the road object.



The Unscented Kalman Filter

The unscented Kalman filter is an implementation of the recursive Bayesian filter, which uses sigma points to approximate data from a Gaussian distribution. The UKF avoids the computational expensive linearization that the EKF has by generating the Jacobian. The UKF propagates the generated sigma points through the non-linear function (in our case the motion model function), and the Gaussian then can be recovered from the newly transformed points.

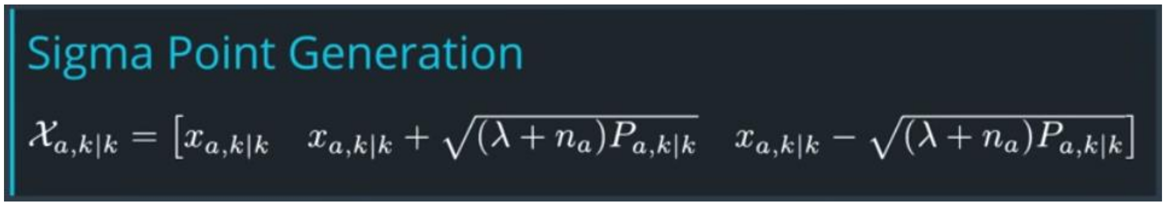
The resulting probability density function is only an approximation of the initial Gaussian distribution, so it is not an optimal filter. From the results section above we can see, that the UKF is shown to have a more accurate estimate than the EKF in the presence of non-linearity.
The general UKF pipeline is depicted in the figure bellow.



The figures below will illustrate the main steps necessary for implementing the predict and update steps of the UKF.

Prediction

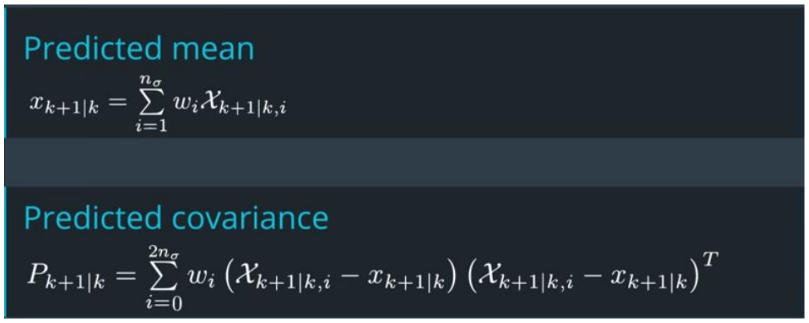
The sigma point are generated according to the following formula.



Sigma Point Generation

$$\mathcal{X}_{a,k|k} = \left[x_{a,k|k} \quad x_{a,k|k} + \sqrt{(\lambda + n_a)P_{a,k|k}} \quad x_{a,k|k} - \sqrt{(\lambda + n_a)P_{a,k|k}}\right]$$

The computed mean and covariance of a number of state samples is given by the equations below.



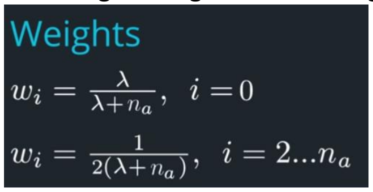
Predicted mean

$$x_{k+1|k} = \sum_{i=1}^{n_\sigma} w_i \mathcal{X}_{k+1|k,i}$$

Predicted covariance

$$P_{k+1|k} = \sum_{i=0}^{2n_\sigma} w_i \left(\mathcal{X}_{k+1|k,i} - x_{k+1|k}\right)\left(\mathcal{X}_{k+1|k,i} - x_{k+1|k}\right)^T$$

The weight are generated using the formulas bellow.

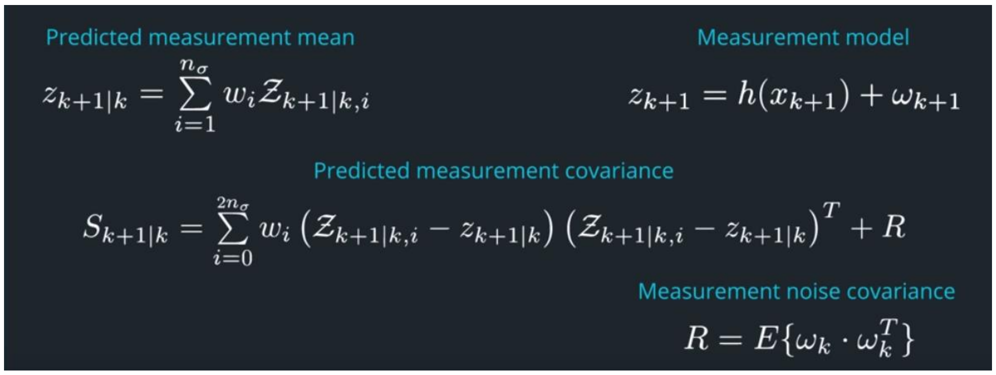


Weights

$$w_i = \frac{\lambda}{\lambda + n_a}, \quad i = 0$$

$$w_i = \frac{1}{2(\lambda + n_a)}, \quad i = 2 \ldots n_a$$

The Update Step



Predicted measurement mean

$$z_{k+1|k} = \sum_{i=1}^{n_\sigma} w_i \mathcal{Z}_{k+1|k,i}$$

Measurement model

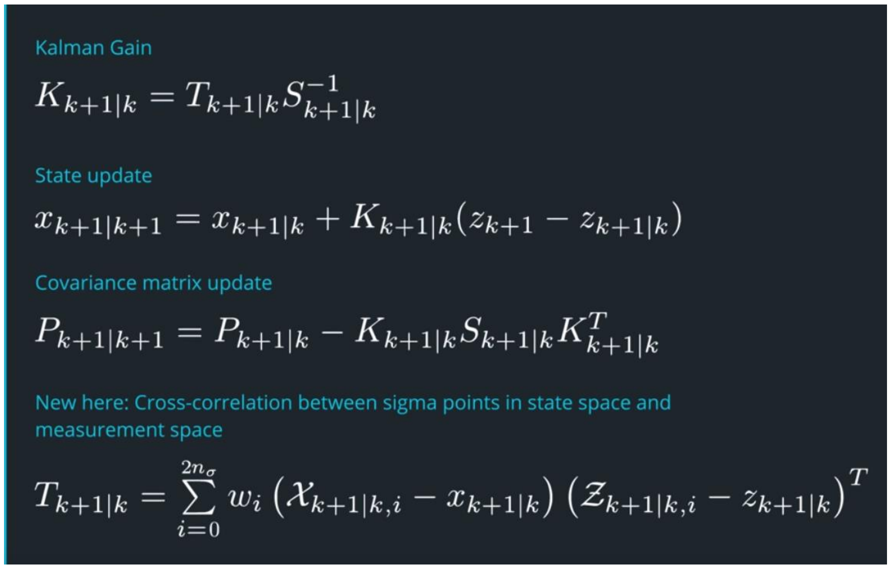
$$z_{k+1} = h(x_{k+1}) + \omega_{k+1}$$

Predicted measurement covariance

$$S_{k+1|k} = \sum_{i=0}^{2n_\sigma} w_i \left(\mathcal{Z}_{k+1|k,i} - z_{k+1|k}\right)\left(\mathcal{Z}_{k+1|k,i} - z_{k+1|k}\right)^T + R$$

Measurement noise covariance

$$R = E\{\omega_k \cdot \omega_k^T\}$$



Kalman Gain

$$K_{k+1|k} = T_{k+1|k} S_{k+1|k}^{-1}$$

State update

$$x_{k+1|k+1} = x_{k+1|k} + K_{k+1|k}\left(z_{k+1} - z_{k+1|k}\right)$$

Covariance matrix update

$$P_{k+1|k+1} = P_{k+1|k} - K_{k+1|k} S_{k+1|k} K_{k+1|k}^T$$

New here: Cross-correlation between sigma points in state space and measurement space

$$T_{k+1|k} = \sum_{i=0}^{2n_\sigma} w_i \left(\mathcal{X}_{k+1|k,i} - x_{k+1|k}\right)\left(\mathcal{Z}_{k+1|k,i} - z_{k+1|k}\right)^T$$