The first project of this course is to create a Extended Kalman Filter. This filter, is a 2d dimensional filter which incorporates in itself the two measurements taken by the car, the LIDAR measurement and the RADAR measurement. For me the most complex code which i will comment is in the file fusionEKF.cpp:

if is initialized<< we update our measurements:

```
if (measurement_pack.sensor_type_ == MeasurementPackage::RADAR) {
/**
Convert radar from polar to cartesian coordinates and initialize state.
*/
float rho = measurement_pack.raw_measurements_(0);
float phi = measurement_pack.raw_measurements_(1);
ekf_.x_ << rho * cos(phi), rho * sin(phi), 0, 0;
```

Now if its lidar or laser we initialize as follows

```
else if (measurement_pack.sensor_type_ == MeasurementPackage::LASER) {
/**
Initialize state.
/
ekf_.x_ << measurement_pack.raw_measurements_(0), measurement_pack.raw_measurements_(1), 0, 0;
}
```

If we continue we see that one of the key phases is to update the state matrix-as follows-

```
float dt = (measurement_pack.timestamp_ - previous_timestamp_) / 1000000.0;
ekf_.F_(0, 2) = dt;
ekf_.F_(1, 3) = dt;
```

We also update the Q matrix, with the noise values set beforehand-

```
ekf_.Q_ <<
dt_4/4noise_ax_, 0, dt_3/2noise_ax_, 0,
0, dt_4/4noise_ay_, 0, dt_3/2noise_ay_,
dt_3/2noise_ax_, 0, dt_2noise_ax_, 0,
0, dt_3/2noise_ay_, 0, dt_2*noise_ay_;
```

Finally we perform the update state, fundamental as well, considering both RADAR and LIDAR measurements:

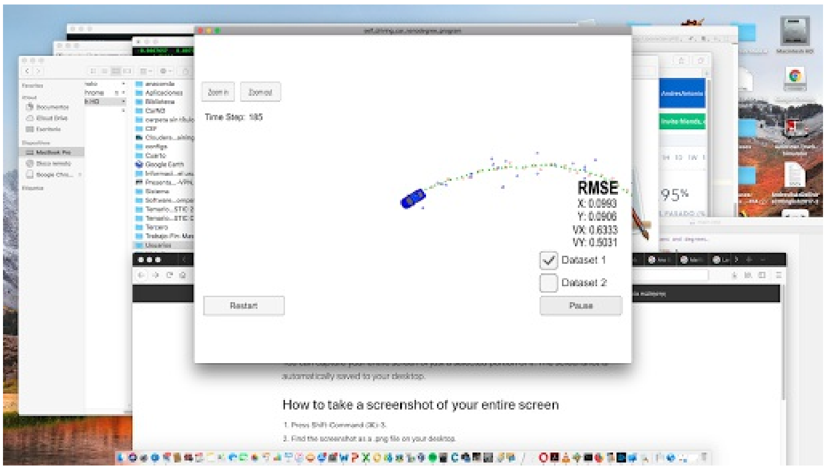
```
if (measurement_pack.sensor_type_ == MeasurementPackage::RADAR) {
// Radar updates
Hj_ = tools.CalculateJacobian(ekf_.x_);
ekf_.H_ = Hj_;
ekf_.R_=R_radar_;
ekf_.UpdateEKF(measurement_pack.raw_measurements_);
```

We will go through the values: Hj is an matrix of Eigen::MatrixXd Hj_; we set with the jacobian of the extended kalman filter. Ekf_r will hold the R measurement of the RADAR. With else we update H and R values of the extended kalman filter, relating to the lidar component of the filter.

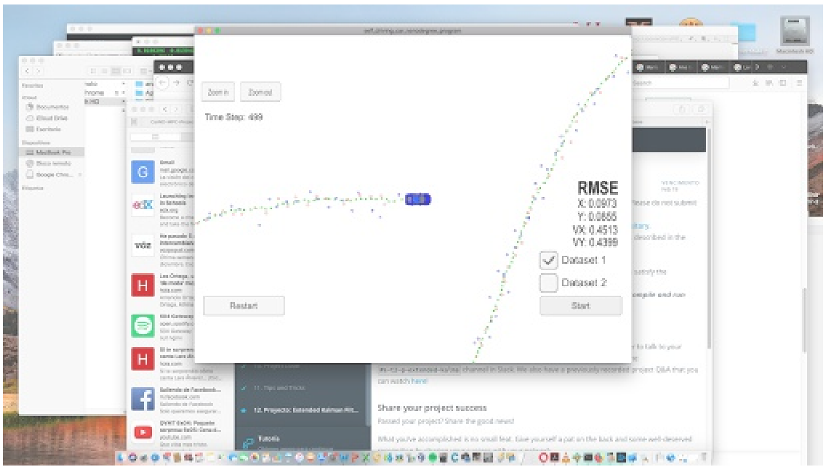
We finally print the output.

We then capture the movement:

P_ = 0.00815384 0.00333016 0.0210784 0.0105279
0.00333016 0.00617554 0.0123927 0.0132026
0.0210784 0.0123927 0.121759 0.0561987
0.0105279 0.0132026 0.0561987 0.0847315
x_ = -8.41345
10.8207
4.72935
-0.26252
P_ = 0.00686288 0.00238874 0.0176571 0.00735842
0.00238874 0.00538388 0.00862104 0.0122781
0.0176571 0.00862104 0.116674 0.0388342
0.00735842 0.0122781 0.0388342 0.093046
x_ = -8.16108
10.8292
4.76408
-0.189205
P_ = 0.00827041 0.00326529 0.0213945 0.0104569
0.00326529 0.00601796 0.0121341 0.012823
0.0213945 0.0121341 0.123154 0.0556293
0.0104569 0.012823 0.0556293 0.0831752
x_ = -7.77955
10.9042
5.15254
0.0386998
P_ = 0.00694562 0.002349 0.0178786 0.00732015
0.002349 0.00528277 0.00847839 0.012043
0.0178786 0.00847839 0.117694 0.038499
0.00732015 0.012043 0.038499 0.0919778
x_ = -7.51129
10.9031
5.14932
0.0779048
P_ = 0.00841311 0.00315017 0.0220687 0.00988118
0.00315017 0.00579187 0.0118113 0.0119264
0.0220687 0.0118113 0.127099 0.0534813
0.00988118 0.0119264 0.0534813 0.0785507
x_ = -7.23246
10.8959
5.19534
0.0613838
P_ = 0.00706985 0.00226997 0.0184169 0.00695398
0.00226997 0.00511771 0.00826626 0.0114158
0.0184169 0.00826626 0.120551 0.0371732
0.00695398 0.0114158 0.0371732 0.0888411
And



Now we see as we progress the following:



Final results are within the rubric.