2.UNSCENTED KALMAN FILTER.

In the second exercise of the second term of udacity, we are proposed a modification on the exercise seen before, the unscented kalman filters. Unscented kalman filters are quite more powerful than 2D kalman filters. In this exercise we have filled in the code so that it works correctly. We will go here through each of the blocks of code.

In the constructor phase we initialize values. We have modified std_a, which was set initially to 30, to 0.5. Equally we have changed std_yawdd to 0.5. The next values (n_x, n_aug, lambda) are set following the values recommended in the lessons. We have updated the weights vector including weights_(0).

In ProcessMeasurement, we input RADAR or LIDAR measurements, update rho, phi, and rho_dot, create px and py, and create vx, vy, and v. We also push this parameters to x, use P and setIdentity and initialize time_us. We do this when initialization is false. If however we have initialized, the we compute delta, update values using Prediction and launch UpdateLidar and UpdateRadar. Finally we add a validity check with infinite. We check if x_sum()or P *sum add to infinite. If this is the case we consider initialization has failed and x* and P will hold the values set before. If this isnt the case, fallback contains the values obtained before.

In prediction, we estimate the object location, sigma points, the state and the state covariance matrix.

In UpdateLidar we use lidar data to update the belief about's the object position. We need to calculate T and K matrixs, plus z_diff. The final formula for the state vector x_, and convariance matrix P, is as follows

```
x_ = x_ + K * z_diff;
P_ = P_ - K * S_ * K.transpose();
```

Next function is similar to the one seen before except we use Radar instead of LIDAR. In this case, the function is slightly more complex. We have to compute px, py, v, and psi as well as the S_matrix. We also have to compute the yaw angle, as seen in the mid level loop.

Finally two functions that helped be bring together the filter. Both were heavily inputted from suggestions by tutors and fellow members of my cohort.

Finally we use the helper function GenerateAugmentedSigmaPoints. We generate sigma points adding in the top left corner the values of P_ as in P_aug.topLeftCorner and Q in the bottomLeftCorner, Q being the noise matrix. We also add sqrt_P_aug which is the 11t() of P_aug. Finally we creates sqrt_term which contains the sqrt of lamba +n_aug by the sqrt of P_aug.

Finally in the last loop we return the values of sigmaPoints using XSig_aug_col, as follows :

```
Xsig_aug.col(i + 1) = x_aug + sqrt_term.col(i);
Xsig_aug.col(i + n_aug_ + 1) = x_aug - sqrt_term.col(i);
```

Another important helper function we use , is PredictSigmaPoints.

We use a for to encapsulate all the logic of this function. First we take all 6 first values from Xsig_aug transforming them to a vector. Auto creates automatically the vector that will contain sigma points.

Sigma points contain px,py, v,psi, psi_dot, and nu_a and nu_psi_dot.

If angle psi_dot is more than 0 set a set of equations as follows

```
x(0) = px + v / psi_dot * (sin(psi + psi_dot * delta_t) - sin(psi)) + 0.5 * delta_t * delta_t * cos(psi) * nu_a;
x(1) = py + v / psi_dot * (-cos(psi + psi_dot * delta_t) + cos(psi)) + 0.5 * delta_t * delta_t * sin(psi) * nu_a;
```

If not

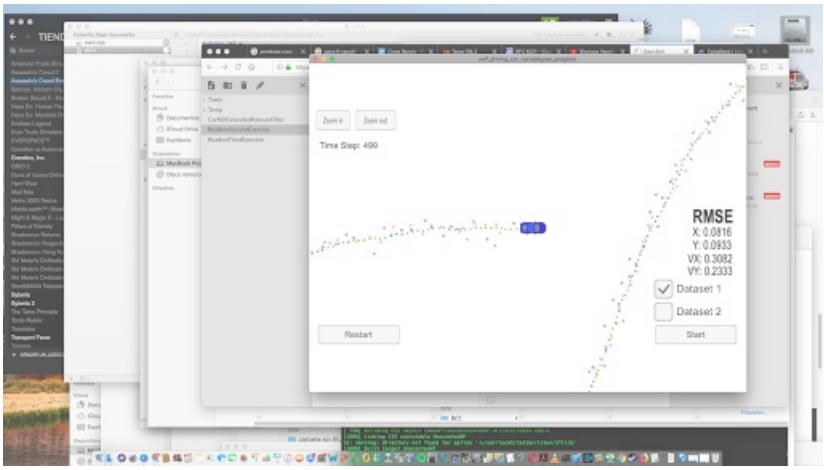
```
x(0) = px + v * (v * cos(psi) * delta_t) + 0.5 * delta_t * delta_t * cos(psi) * nu_a;
x(1) = py + v * (v * sin(psi) * delta_t) + 0.5 * delta_t * delta_t * sin(psi) * nu_a;
```

Both formulas come from the lesson on unscented filters. The next two equations also correspond to the equations of the class.

x(2) = v + 0 + delta_t * nu_a;
x(3) = psi + psi_dot * delta_t + 0.5 * delta_t * delta_t * nu_psi_dot;
x(4) = psi_dot + 0 + delta_t * nu_psi_dot;
We then add x values to xsig_pred and return it.
Finally NormalizeAngle function normalizes the angle.

RESULTS,

We launch and see if the values are within the rubric and less than the values for the unscented filter.



We now go to the rubric page:
px, py, vx, vy output coordinates must have an RMSE <= [.09, .10, .40, .30]
We can see that

1. the unscented filter shows a substantial gain over values in the scented filter
2. Those values are within the rubric
   However as disadvantages we can see that:
   It is fairly complex to program, much more than the first filter.

> Written with StackEdit.