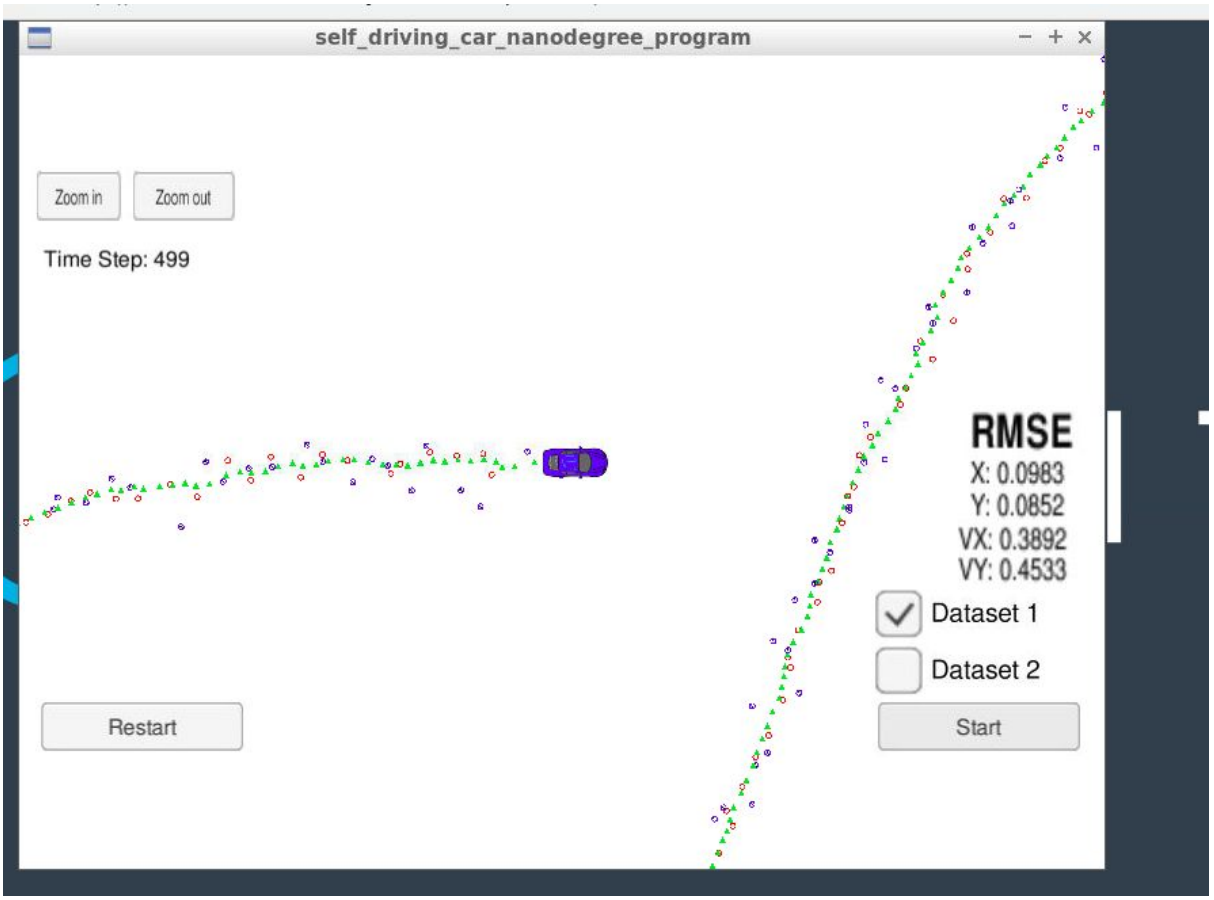
Write Up for P5: Extended Kalman Filter

Compiling

This document is a write up for the 5th project. My submission is included in the git repository: https://github.com/mmarouen/CarND-Behavioral-Cloning-P3
The code was run using the workspace settings and is compiled using cmake and make

Accuracy

RMSE=[***0.0983***,***0.0852***,***0.3892***,***0.4533***]



Following correct algorithm

Code follows general flow

1. Initialization

```
if (!is_initialized_) {

  // first measurement
  cout << "EKF: " << endl;
  VectorXd x_(4);
  x_ << 1, 1, 1, 1;

  if (measurement_pack.sensor_type_ == MeasurementPackage::RADAR) {
      double rho = measurement_pack.raw_measurements_[0]; //radius
      double theta = measurement_pack.raw_measurements_[1]; //angle
      x_ << rho*cos(theta), rho*sin(theta),0.0,0.0;
      ekf_.Init(x_,P_,F_, Hj_,R_radar_,Q_);
  }
  else if (measurement_pack.sensor_type_ == MeasurementPackage::LASER) {
    x_<< measurement_pack.raw_measurements_[0], measurement_pack.raw_measurements_[1],0.0,0.0;
    ekf_.Init(x_,P_,F_, H_laser_,R_laser_,Q_);
  }
  previous_timestamp_ = measurement_pack.timestamp_;

  // done initializing, no need to predict or update
  is_initialized_ = true;
  return;
}
```

2. Prediction

```
float dt = (measurement_pack.timestamp_ - previous_timestamp_) / 1000000.0; //dt - expressed in seconds
previous_timestamp_ = measurement_pack.timestamp_;

/*****************************************************************************
 *  Prediction
 ****************************************************************************/

// State transition matrix update
ekf_.F_(0,2) = dt;
ekf_.F_(1,3) = dt;

float dt_2 = dt * dt;
float dt_3 = dt_2 * dt;
float dt_4 = dt_3 * dt;
//set the process covariance matrix Q
ekf_.Q_ = MatrixXd(4, 4);
ekf_.Q_ <<  dt_4*noise_ax/4, 0, dt_3*noise_ax/2, 0,
0, dt_4*noise_ay/4, 0, dt_3*noise_ay/2,
dt_3*noise_ax/2, 0, dt_2*noise_ax, 0,
0, dt_3*noise_ay/2, 0, dt_2*noise_ay;

ekf_.Predict();
```

3. Update

```
/*****************************************************************************
 *  Update
 ****************************************************************************/

/**
 TODO:
   * Use the sensor type to perform the update step.
   * Update the state and covariance matrices.
 */

if (measurement_pack.sensor_type_ == MeasurementPackage::RADAR) {
    //return;
    ekf_.H_ = tools.CalculateJacobian(ekf_.x_);
    ekf_.R_=R_radar_;
    ekf_.UpdateEKF(measurement_pack.raw_measurements_);
} else {
    //return;
    ekf_.H_ = H_laser_;
    ekf_.R_=R_laser_;
    ekf_.Update(measurement_pack.raw_measurements_);
}
```

Dealing with first frame

The parameters that can be tuned for the first frame are:
Initial state **x_0** + initial covariance **P_0** (other parameters were chosen adequately).

- Initial state x_0:
  I use the first measurement as initialization for **px** & **py** and set **vx=vy=0**.

```
x_ << rho*cos(theta), rho*sin(theta),0.0,0.0;
```

```
x_<< measurement_pack.raw_measurements_[0], measurement_pack.raw_measurements_[1],0.0,0.0;
```

- Initial covariance P_0:
  For me this is the trickiest part in all the project!
  After several trials, I decided to use the following covariance matrix as initialization:

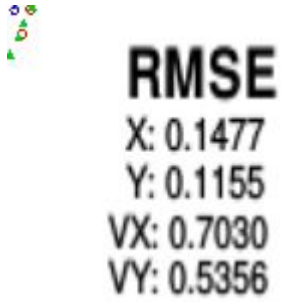
```
P_ << 200.0, 50.0, 30.0, 0.0,
      50.0, 100.0, 0.0, 15.0,
      30.0, 0.0, 20.0, 5.0,
      0.0, 15.0, 5.0, 10.0;
```

  Here are the main reasons for this:
  -High covariance for px compared to py because movement is mostly along x-axis
  -Same applied for vx and vy
  -An increase in px will affect positively py

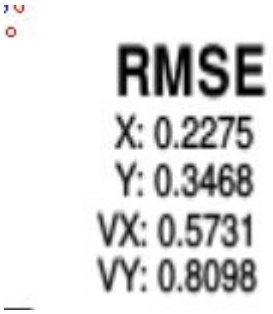
Project discussion

Remove radar/laser

Laser only



Radar only



- Laser is better at detecting `px` & `py`
- Radar is better at detecting `vx`
- Laser is better at detecting `vy`

These results are not surprising because radar measurements are more noisy.
On the other hand, radar observes speed, which may explain why radar detect vx better.

Going further

This project was beneficial to get started with c++ and apply my knowledge on Kalman filters. I think the RMSE can be improved by:

- Better initialization of `P_0`
- Better initialization of `x_0`