**Tracking**

**Simple Motion Model**

State vector:

Linear Motion Model with constant acceleration.

Write the motion model in state space representation

Output …..

In this project, is set to be 1 second for simplicity.

**Sensor Fusion**

Problem: If have a noisy laser and noisy radar sensor at the origin, can we track a moving car accurately by combining laser and radar measurements. This topic is important because in order to drive a car safely, we must be able to know the positions and velocities of other cars.

**Sensors**

1. Laser

Laser uses an infrared laser beam to determine the distance between the sensor and a nearby object. Laser cannot measure the velocity of objects directly and has to rely on two or more position measurements to estimate the velocity.

In practice, the measurement of a laser sensor is point clouds. For simplicity, here assume that the measurement from a laser sensor is already computed to the 2D location.

The state vector is

The measurement matrix of a laser sensor is as below.

Based on measurement matrix above, it can be found that laser sensors can only measure x position and y position. The x and y velocity can be calculated by state transition matrix.

The laser measurement noise covariance matrix is as below.

Here assume that x direction noise and y direction noise are uncorrelated.

The matrix R represents the uncertainty in the position measurements receive from the laser sensor.

2. Radar

Radar uses radio waves and Doppler effect to measure speed directly, thus it is important for sensor fusion because it helps fusion algorithms converge much faster.

Different from measurements from laser sensors which are in cartesian plane, measurements from radar sensors are in polar plane. The measurement vector is as below.

Where is the distance between a radar sensor and the observed object, is the angle between the ray and x axis, and is Doppler velocity which represents the velocity of the observed object.

The radar measurement noise covariance matrix is as below.

The measurement function:

Here assume that three measurement noises are uncorrelated.

**Kalman Filter**

Kalman Filter is the most popular technique for solving tracking problems because the it’s very efficient. In Kalman Filter, the distribution is given by a unimodal Gaussian. The task of Kalman Filter is to maintain mean(Mu) and variance(Sigma square) as the best estimate of the location.

Kalman Filter algorithm iterates on two cycles. The first one is the measurement update which is based on Bayes Rule. Taking a product of prior distribution and measurement probability and

then normalize it to get posterior distribution. The second is the predict process applying total probability. The pseudo code of time-varying-gain Kalman Filter is as below.

Prediction:

Measurement update:

H is measurement matrix

R is measurement noise

P is uncertainty covariance matrix

K is Kalman gain matrix

I is identity matrix

F is state transition matrix

Q is noise covariance

u is input

z is measurement

**Extended Kalman Filter**

The Kalman Filter algorithm works well for laser sensors because of the linearity of measurements from laser sensors. However, the measurement function of radar sensors is not a linear function, thus, during the measurement update process, the distribution becomes non-Gaussian. Therefore, Extended Kalman Filter is necessary to predict states from non-linear measurements.

Extended Kalman Filter can deal with non-linear model (non-linear state transition matrix) or non-linear measurement functions, the ideal of it is linearizing these matrices by first order Taylor expansion.

where is Jacobian matrix: Hj

Given that the motion model is a linear model, there is no need to linearize state transition matrix. The pseudo code of time-varying-gain Extended Kalman Filter is as below.

Prediction:

Measurement update:

**Localization**

After implementing Kalman Filter and Extended Kalman Filter, the location of the observed object can be known. Another important issue is that how to know our present location relative to fixed coordinate when we are tracking the target or a given trajectory.

**Particle Filter**

The first step of implementing a particle filter would be the initialization step, that is, create particles based on the location of the car. Each particle contains three piece of information which are x position, y position, and heading direction, respectively. Although GPS is usually not precise enough, it is the only information which can be get at the beginning.

The second step would be the prediction step, that is, predict the new location of the car. Each particle will move to its new location based on the velocity and yaw rate measurement. Also, to account for the uncertainty in the control input, Gaussian noise is added to yaw rate and velocity during the prediction step.

The third step is update step. This step takes measurements of landmarks into account. The measurements will inform the weight of each particle, instead of affecting the states of the car directly. Here, the weight of each particle is updated by using multivariate Gaussian probability density function for each measurement. The assumptions here are the car uses its laser sensors the measure the distance between landmarks and itself, laser sensors have Gaussian noise and the x axis and y axis measurement are uncorrelated, and each measurement is independent. After this step, particles which are closer to the car will get higher weights. The weight of each particle represents probability, and thus the sum of weight of each particle would be 1.

The last step is resample step which can be said is the most important part of the particle filter. The essence of this step is to create a new set of particles based on the weight calculated in update step. Particles which have higher weight are closer to the car and have higher probability to be selected during the resample step. The means of each particle’s x and y position represent the guess of the car true position.

**Code**

**Sensor**

**Laser Sensor Class**

In this project, the assumption of laser sensor is that it already analysis the point could and return x and y position output.

In this class, two fields are defined, which are standard deviation of x and y directions. The default setting of these two standard deviations are both 0.15. There are three methods defined in this class. The first method is simply measuring the x and y position of the target with respect to the fixed frame. This method is used to test the performance of the sensors and filters. Other two methods are used to simulate a moving car sensing a moving object and the landmarks.

**Radar Sensor Class**

In this class, seven fields are defined. The first three are the standard deviations of distance, angle and velocity measurement and the default settings are 0.5, 0.05, and 0.5, respectively. Another four are data used to store and calculate distance, angle and velocity.

Two methods are defined in this class. The first method is simply measuring the distance, angle and velocity of the target with respect to the fixed frame. The second method is used to simulate a moving car sensing a moving object.

**Kalman Filter Class**

In this class, nine fields are defined. Users need to provide F, state transition matrix, H, measurement matrix, P, uncertainty covariance matrix, R, measurement noise, and dt to instantiate a Kalman Filter.

Q is noise covariance, here, it is used to represent the effect of the acceleration of the motion model. Also, four lists are defined to store x, y position data of predict step and update step.

There are three methods defined in this class. The first one is to filter through a set of measurement data in Cartesian coordinate, and store x, y position data during predict and update step. This method is used to evaluate the performance of a Kalman Filter. The second method is to estimate the next states based on the model and present states, and this method is used in sensor fusion. The third method is to set Q, noise covariance matrix.

**Extended Kalman Filter Class**

In this class, eight fields are defined. Users need to provide F, state transition matrix, P, uncertainty covariance matrix, R, measurement noise, and dt to instantiate an Extended Kalman Filter. Also, four lists are defined to store x, y position data of predict step and update step. There are three methods defined in this class. The first one is to filter through a set of measurement data in Polar coordinate, and store x, y position data during predict and update step. This method is used to evaluate the performance of an Extended Kalman Filter. The second method is to estimate the next states based on the model and present states, and this method is used in sensor fusion. The third method is to set Q, noise covariance matrix.

**Particle Filter**

**Particle Class**

In this class, nine fields are defined. Users need to provide the target, which contains x, y position information to instantiate a particle. A particle contains x, y, and orientation information. Also, three fields are defined to store noise information. There are six methods defined in this class.

The first method is a setter to set x, y, and orientation of a particle.

The second method is to set the noise level of a particle.

The third method is for prediction step. A particle will move based on velocity and yaw rate measurement. Also, Gaussian noise is added during this process.

The fourth method is used to calculate multivariate Gaussian probability density function.

The fifth method is to calculate the probability of a particle based on multivariate Gaussian probability density function.

The sixth method is to define the representation of an instance of Particle Class.

**Particle Filter Class**

In this class, eight fields are defined. Users need to provide (1) a target, which contains x, y, and orientation information, (2) particle numbers, (3) landmarks’ positions in matrix form, and (4) GPS error. Two lists are defined to store the all particles information and residual of each estimation. There are two methods defined in this class. The first method is to implement the process for localization, including predict, update, resample steps. After finishing these steps, the estimated x and y position are obtained by taking average of all x, y position of all particles. The last step is to calculate the residual of each estimation.

The second method is to calculate the residual of an estimation and is used in the first method.

**Car Class**

In this class, there are fifteen fields are defined. Users need to provide landmarks’ positions in matrix form to create an instance of Car Class. The default setting of x, y, and orientation of a car are 0. Also, once create an instance of Car Class, one laser sensor, one radar sensor, and a localizer are created by default. Five methods are defined in this class. The first is a setter to set the x, y position, and orientation of a car. The second method is to set steering drift parameter of a car. The third method is to move a car based on given steering angle and distance and update the new x, y position and orientation of a car.

The fourth method is to measure the distance between a car and each landmark by using a laser sensor. This method is used in the third method. The fifth method is to define the representation of an instance of Car Class.