Self-Driving Engineer

Sensor Fusion

Introduction to Sensors

Aim:

- Fuse LIDAR and RADAR data using Kalman filters as they both produce very different data
- Sensor Fusion with unscented Kalman Filters (involves technique for non-linear motion)

Sensor fusion needs to happen quickly, so need a high performance language like C++

Radar (RAdio Detection and Ranging):

Unlike other vehicles which measure speed by seeing difference between two consecutive readings, Radar measures speed by Doppler effect (change in frequency of radar waves depending upon whether object is moving close to you or away from you)

- Can be used for localization by generating radar maps of the environment, as they bounce off hard surfaces
- RADAR can see underneath vehicles and spot buildings and objects that would be obscured otherwise
- Low Resolution as compared to LIDAR and cameras, vertical resolution negligible
- Better to disregard static objects (radar clutter)
- Least affected by rain or fog
- Wide range of view ~ 150 degrees
- Long range ~ 200+ metres

LIDAR (Light Detection and Ranging):

Infrared beam (900nm range or some use even longer wavelength so that better performance in rain and fog). Rotating swivels scan the laser beam across the field of view. Lasers are pulsed and the pulses are reflected by objects. These reflections return a point cloud that represents these objects.

- Higher spatial resolution because of focussed laser beam, vertical direction also large number of LIDAR points per layers, and higher number of scan points per layer.
- Estimate velocity by differing position between two or more scans, cannot estimate directly
- Affected by weather conditions, and dirt on sensor
- Bulkier, so difficult to integrate

Gaussian distribution Formula

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/(2\sigma^2)}$$

Monte Carlo Localization

Estimate discreet state of object

Gives rise to multi-modal distribution

Kalman Filters (1-D)

Kalman filters estimate continuous state of object (Gaussian distribution)

Gives rise to uni-modal distribution.

Kalman filters represent Gaussian distribution and iterates on two main cycles

First cycle is measurement update

Requires a product

Uses Bayes Rule

```
def update(mean1,var1,mean2,var2):
    """
    Given a mean and variance of belief, and the mean and variance of measurement,
    this function updates the parameters of the belief function
    """
    new_mean = (mean1*var2+mean2*var1)/(var1+var2)
    new_var = 1/(1/var1+1/var2)
    return [new_mean,new_var]
```

Second cycle is motion update (Prediction Cycle)

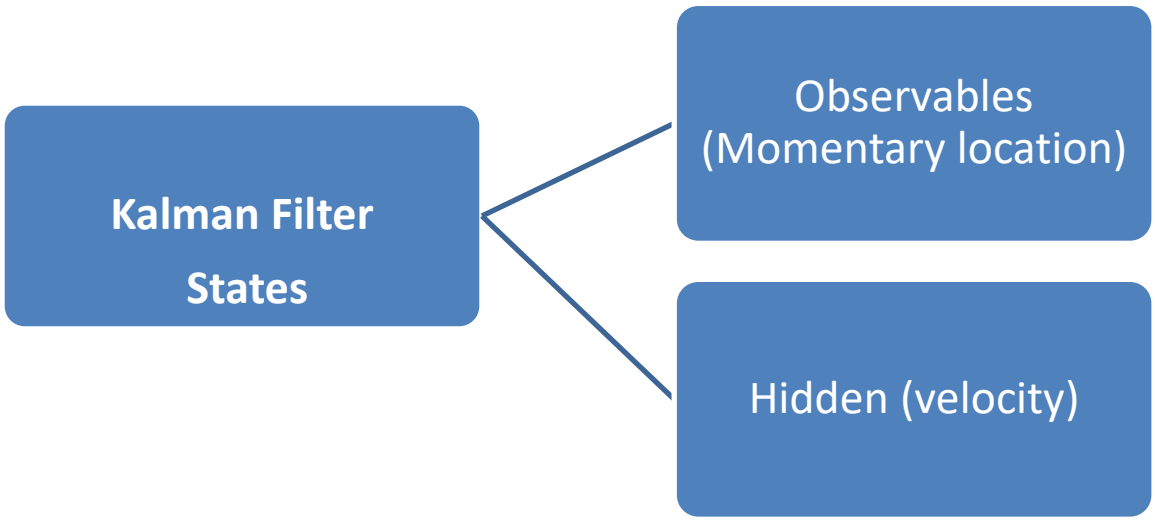
Involves convolution (or simply addition)

Uses total probability

```
def predict(mean1, var1, mean2, var2):
    """
    Motion Update :
    Given a mean and variance of belief, and the mean and variance of measurement,
    this function updates the parameters of the belief function
    mean2 var2 , uncertainty of our motion
    mean1 var 1 , prior uncertainty
    """
    new_mean = mean1 + mean2
    new_var = var1 + var2
    return [new_mean, new_var]
```

Kalman Filter Land

High Dimensional Gaussians / Multivariate Gaussians



Observables and Hidden interact; thereby subsequent values of observables give us information about these hidden variables.

Design Kalman Filter

Update Equations

X = Estimate

F = State Transition Matrix

U = Motion Vector

P = Uncertainty Covariance

Z = Measurement

H = Measurement Function

R = Measurement Noise

I = Identity Matrix

Measurement Step

$y = Z - H.X$ (Error)

$S = H.P.H' + R$

K (Kalman Gain) = $P.H'.in(S)$

$X' = X + (K.y)$

$P' = (I - K.H) . P$

Prediciton Step

$X' = FX + U$

$P' = F.P.F'$