Extended Kalman Filter Navigation System-Slam 2D Problem

Documentation

The following Matlab code implements the extended Kalman filter to be use in a Slam application with encoders and laser sensors.
The data set used was taking in the Car Park, with artificial beacons. This will let us to make a comparison between the real and estimated position of the beacons and analyse the filter's performance.

To run the code:
Run the file "slam". This sets up a graphical window that will let you to see the path estimation and the difference with the "real" path (Gps).
To start the filter, run the script file ***slam***.

Function Definitions:

slam:
Function to run the code

FindT:
Function to find the index of the time sensor vector, for the selected time.

[GPSTIME,LONG,LAT]=ReadGpsData(file):
Function to read the gps data, and transform to a local navigation frame with respect to a reference point.

[Time,STEERING,SPEED1]=ReadUteData(file):
Similar to the *ReadGpsData* function, is to read the encoder data, from a mat file. (Not used in this example since all sensor data is in a single matlab file.

Pred(dt,u):
Function to implement the prediction stage of the EKF. The inputs are the estimated state vector, the estimates covariance matrix and the encoder data (velocity and steering).

[beacon]=getpos:
Function to get the position of the artificial beacons, from a gps data set.

[index,innov,S]=asoc_update(zlaser,pointer,updatee):
Function to make the association of the observation with one beacon, and eventually the update. In the case one beacon was already associated to the observation, or if it is the first time we are seeing it, then we have to incorporate this to the vector state and make an update with this new state.
The flag updatee is used to know if it is an update or if it is only an association. A flag index is return to know the result of the chi square test.

new_state(zlaser):

This function is used when we have a new observation that was not associated with any beacon of the state vector. The Cartesian position of the “new” beacon is evaluated and insert as a new state. A large number for the variance uncertainty is set. This may introduce numerical problems if not properly selected.

[index2]=Zone_Probe(zlaser):
The association task is very expensive. It is important to try association only with the beacons that are in the area. This area can be evaluated with the uncertainty of the vehicle. It returns the indices to all the beacons that are within a certain area.

laserview(RR,a,xp,hhh3,hhh4,hhh2,LASERr,LASERo):
Function to plots the laser scan. Plots the standard reflections in one colour and the high intensity reflection in a different colour.

[LASERr,LASERo,RR,a]=getdata(laser):
Function to get the range and bearing to the beacons. Gets the high intensity point and evaluate the centre of the beacon.

[innov, S]=update_gps(zgps):
Implement the update stage of the EKF with a gps observation. The inputs are the predicted state vector, the predicted covariance matrix, and the gps data (longitude and latitude). The GPS is only used for initialisation purposes.

[meanq,q,chib_up,chib_low,timeinn]=inn_analyse(inn,S):
Function to analyse innovation sequences. Is used only to plot the Normalised Innovations and Innovation Confidence Bounds.

Rxx=auto(x):
Computes autocorrelation of input data set. N is the number of data points, x is a column matrix holding the input data set.
Uses fft method as advertised in Maybeck p193.

Plots:
Script file to run all the plots when the filter finish.

SaveStates(states,diagcov,times,Flag):
Function to save the states and the covariance matrix in arrays. All the beacons in the state vector are saved. Another vector is saved called FlagStates with the number of times the beacon has been seen..