SE 423 Mechatronics Homework Assignment #6
Due 4:00pm Friday May 3rd.

1. The goal of this problem is to get you to read through given code implementing the A* method of path planning. Most of the code has been given to you. Your job is either to fill in the blank with some code or add some comments indicating that you understand the algorithm. I suggest you perform this homework exercise in the lab. The lab PCs have the correct version of Microsoft Visual Studio 2022 installed. All the code and project files you need for this problem is posted at the course website http://coecsl.ece.illinois.edu/se423/hw/astarhw6.zip. Download this zip file and unzip it into a folder.
   Perform the following:
   i. In this project's directory you will find the executable file "astarDemo.exe." This is given to you to see how the program should run when you fill in the appropriate code asked for. Run "astarDemo.exe" and play around with the different maps that are created for you to test the algorithm. Use astarDemo.exe in the below items to double check that your code is running properly.
   j. Load the Visual Studio 2022 project by double clicking on the astar.sln file. If Windows ask select Visual Studio 2022. This will load Visual Studio 2022. It may take a bit longer to load the project the first time the project is opened. Under "source files" find and open astar.cpp. Take some time to run through the code in this file. There are many helpful comments in the file that will help you with this exercise. Also the pseudocode for the A* algorithm we went through in class and posted here could be helpful for this exercise. There are 12 items that are numbered in the A* function of astar.cpp. Some of the items have you add a line or two of source code. Other items have you simply write comments explaining what certain lines of code are accomplishing. Complete these 12 items. Then compile and run your project by pressing the arrow/play button. Once you program is compiled it should run identical to the astarDemo.exe file. For your homework submission, print out the changes and comments you made for these twelve items.
   k. Change the location of at least three obstacles in the course map defined by the array "char mapCourseStart[176]". This is also the map that is used when item 7 is selected when running the application. Then for your homework submission, print the output of your compiled program when you have selected a start and finish point that has a number of obstacles in between them.
   l. As a final exercise, add enough obstacles so that there is no path A* can find from the starting point to the ending point. Verify that A* returns an error when it cannot find a path.
2. The goal of this problem is to give you experience with the Kalman filter and understanding how it is helping with the robot's navigation. Given a set of actual data stored during a run of the robot car, your goal is to find a Q and a R matrix that makes the Kalman filter merge the dead reckoning data from the wheel encoders and rate gyro with the optitrack data. The "ideal" merge of this data is not to track exactly the optitrack data. It should definitely be close to the optitrack data but filter the optitrack data so large errors are not followed exactly. The optitrack's resolution is +\- 2cm and it is possible (but not very often) for the optitrack to send incorrect measurements. I have purposely added a 1 tile error in one of the optitrack data points so you can see how your Kalman filter reacts to that kind of large error.
   What are you given:
   A. The data file data_forKalman_Filter_HWProblem.mat found at http://coecsl.ece.illinois.edu/se423/hw/HW6.htm. It has 6 columns: index, average wheel velocity, gyro reading in rad/s, x optitrack position, y optitrack position, theta optitrack. After you load the file you will see a lot of zeros in the optitrack columns. The optitrack only samples at 100 Hz where the dead reckoning data (wheel velocity, gyro) is updated at 1000 Hz. Your code will not use the optitrack data when it is 0.0. Load the "data" variable by loading the data_forKalman_Filter_HWProblem.mat file.

This can be done using the "load" command or double clicking the file in the "current directory" window of the Matlab window.

B. A Matlab script file that gives you most of what you need for the code for this problem, KalmanFilterHWProblem.m located http://coecsl.ece.illinois.edu/se423/hw/HW6.htm. You will need to make two modifications to this file: (1) adding the Kalman equations to the large for loop, and (2) assigning values to the Q and R matrices used in the filter.

In the m-file's for loop, there are comments to guide you through adding the code. Within the for loop you need to implement the prediction and correction steps. Referring to the given equations at the end of this homework might be useful during this coding.

After adding the Kalman filter code to the for loop, you need to choose values for the Q and R matrices. The following form works well for the robot car.

$$U_P = Process\ (Dead\ Reckoning)\ Uncertainity, Q = \begin{bmatrix} U_p & 0 & \frac{U_p}{10} \\ 0 & U_p & \frac{U_p}{10} \\ \frac{U_p}{10} & \frac{U_p}{10} & U_p \end{bmatrix}$$

$$U_M = Measurement\ (Optitrack)\ Uncertainity, R = \begin{bmatrix} U_M & 0 & \frac{U_M}{10} \\ 0 & U_p & \frac{U_M}{10} \\ \frac{U_M}{10} & \frac{U_M}{10} & U_M \end{bmatrix}$$

Start out with U_P *and* U_M at 1.0 and run the M-file. (U_P and U_m are named "MeasUncert" and "ProcUncert" in the code.) Zoom in on the third plot that plots both data sets on top of each other. Does it look like the Kalman filtered data is following the Optitrack data too closely and jumping around? Optitrack data is the dots. Also make sure to see what happens at the one error point that is off by one tile in the y direction. Then tune U_P *and* U_M values by changing them by a positive or negative power of 10 relative to each other. Come up with 3 Q and R matrices pairs that show three cases:

1. Kalman output data follows the Optitrack data too closely.
2. Kalman output is filtered too much and converges slowly to the Optitrack data.
3. Kalman output is just right, converges pretty quickly but does not jump imidiately to every Optitrack data point.

Make sure to produce a plot for each of these cases showing how the Kalman filter is working. In your plots, be sure to zoom in on the behavior of your fused data around the discontinuity in the optitrack data. Be sure to show how quickly the fused data converges to the optitrack data.

What to turn in:

1. A print out of your edited M-file
2. Three plots for the three Q and R pairs that you chose and somewhere by the plot the value of the Q and R matrices.

Kalman Equations

For the robot car

$$F = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \cos(\theta_k)\Delta t & 0 \\ \sin(\theta_k)\Delta t & 0 \\ 0 & \Delta t \end{bmatrix}, \text{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{and start P as identity P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

K is 3 x 3 and S is 3 x 3

$$\begin{bmatrix} \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} = \begin{bmatrix} x\ position\ of\ robot \\ y\ position\ of\ robot \\ \theta\ of\ robot \end{bmatrix}, \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} average\ velocity\ of\ wheels \\ rate\ gyro\ measurement \end{bmatrix}, \begin{bmatrix} \text{z}_1 \\ z_2 \\ \text{z}_3 \end{bmatrix} = \begin{bmatrix} optitrack\ x\ position\ of\ robot \\ optitrack\ y\ position\ of\ robot \\ optitrack\ \theta\ of\ robot \end{bmatrix}$$

Q and R are 3 x 3 matrixes that you need to tune.

Prediction Steps

$$\hat{x}_{k+1|k} = F\hat{x}_{k|k} + Bu_k$$

$$P_{k+1|k} = FP_{k|k}F^T + Q_k$$

Correction Steps

$$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K\tilde{y} \qquad \text{where} \quad \tilde{y} = z_{k+1} - H\hat{x}_{k+1|k}$$

$$P_{k+1|k+1} = (I - KH)P_{k+1|k} \qquad S = HP_{k+1|k}H^T + R_{k+1}$$

$$K = P_{k+1|k}H^T(S)^{-1}$$