Homework: Week 5

Jasen Carroll

MEM 380 – Wheeled Robotics

2/3/2011

Table of Contents

[Problem 1: 3](#_Toc284694061)

[Problem 2: 4](#_Toc284694062)

[Problem 3: 8](#_Toc284694063)

[Problem 4: 10](#_Toc284694064)

# Problem 1:

function deadReckon1\_jcc329\_AH(filename) %%NOTE HERE

% clear %% NOTE HERE

clc

% initializes the figure window

figure (1)

clf

axis([-5 5 -5 5]);

% prompts the user to select a starting position for the robot

disp('Select an initial start position for your robot.')

[xStart, yStart] = ginput(1); % allows you to select a point in the figure window by clicking on it

% reads in the robot centric accelerations and angular velocities from

% input file

[xRAccel, yRAccel, zROmega] = textread('input\_wk4.txt', '%f%f%f'); %% NOTE HERE

size(zROmega)

% initializes the integration time step and robot velocities and states

dt = 0.02; % integration time step

v = 0; % robots initial speed

% fixes the range of the axes on the figure

axis([-5 xStart+13 -5 yStart+13])

hold on

% the sensors give back the x and y acceleration plus the thetadot info.

% This means that the position data can be updated by integrating the

% accelerations and velocities accordingly

% robot's pose in the world coordinate frame

xW = xStart;

yW = yStart;

thetaW = 0;

rob = makeRobot(xW, yW, thetaW);

for i = 1:length(xRAccel)

thetaW = rob.theta + zROmega(i)\*dt; % updates theta

v = v + xRAccel(i)\*dt; % updates velocity

xW = rob.x + v\*cos(thetaW)\*dt; % updates posistion of x

yW = rob.y + v\*sin(thetaW)\*dt; % updates position of y

% update robot's state and draw robot

rob = moveRobot(rob, xW, yW, thetaW);

pause(.001);

end

drawRobot(rob, 0.5, 'r')



# Problem 2:

function deadReckon2b\_jcc329(~)

clear

clc

% initializes the figure window

figure (1)

clf

axis([-5 5 -5 5]);

% prompts the user to select a starting position for the robot

disp('Select an initial start position for your robot.')

[xStart, yStart] = ginput(1); % allows you to select a point in the figure window by clicking on it

% reads in the robot centric accelerations and angular velocities from

% input file

[xRAccel, yRAccel, zROmega] = textread('input\_2.txt', '%f%f%f');

% initializes the integration time step and robot velocities and states

dt = 0.02; % integration time step

v = 0; % robot's initial speed

% fixes the range of the axes on the figure

axis([-5 xStart+75 -5 yStart+13])

hold on

% the sensors give back the x and y acceleration plus the thetadot info.

% This means that the position data can be updated by integrating the

% accelerations and velocities accordingly

% robot's pose in the world coordinate frame

xW = xStart;

yW = yStart;

thetaW = 0;

rob = makeRobot(xW, yW, thetaW);

for i = 1:length(xRAccel)

thetaW = rob.theta + zROmega(i)\*dt; % updates theta

v = v + 9.807\*.005\*xRAccel(i)\*dt; % updates velocity

xW = rob.x + v\*cos(thetaW)\*dt; % updates posistion of x

yW = rob.y + v\*sin(thetaW)\*dt; % updates position of y

% update robot's state and draw robot

rob = moveRobot(rob, xW, yW, thetaW);

pause(.001);

end

drawRobot(rob, 0.5, 'r')

figure (2)

clf

axis([-5 5 -5 5]);

[xStart, yStart] = ginput(1);

% initializes the integration time step and robot velocities and states

dt = 0.02; % integration time step

v = 0; % robot's initial speed

% fixes the range of the axes on the figure

axis([-5 xStart+75 -5 yStart+13])

hold on

% the sensors give back the x and y acceleration plus the thetadot info.

% This means that the position data can be updated by integrating the

% accelerations and velocities accordingly

% robot's pose in the world coordinate frame

xW = xStart;

yW = yStart;

thetaW = 0;

rob = makeRobot(xW, yW, thetaW);

for i = 1:length(xRAccel)

thetaW = rob.theta + zROmega(i)\*dt; % updates theta

v = v + xRAccel(i)\*dt; % updates velocity

xW = rob.x + v\*cos(thetaW)\*dt; % updates posistion of x

yW = rob.y + 9.807\*.005\*v\*sin(thetaW)\*dt; % updates position of y

% update robot's state and draw robot

rob = moveRobot(rob, xW, yW, thetaW);

pause(.001);

end

drawRobot(rob, 0.5, 'r')

figure (3)

clf

axis([-5 5 -5 5]);

[xStart, yStart] = ginput(1);

% initializes the integration time step and robot velocities and states

dt = 0.02; % integration time step

v = 0; % robot's initial speed

% fixes the range of the axes on the figure

axis([-5 xStart+75 -5 yStart+13])

hold on

% the sensors give back the x and y acceleration plus the thetadot info.

% This means that the position data can be updated by integrating the

% accelerations and velocities accordingly

% robot's pose in the world coordinate frame

xW = xStart;

yW = yStart;

thetaW = 0;

rob = makeRobot(xW, yW, thetaW);

for i = 1:length(xRAccel)

thetaW = rob.theta + zROmega(i)\*dt; % updates theta

v = v + xRAccel(i)\*dt; % updates velocity

xW = rob.x + v\*cos(thetaW)\*dt; % updates posistion of x

yW = rob.y + v\*sin(thetaW)\*dt; % updates position of y

% update robot's state and draw robot

rob = moveRobot(rob, xW, yW, thetaW);

pause(.001);

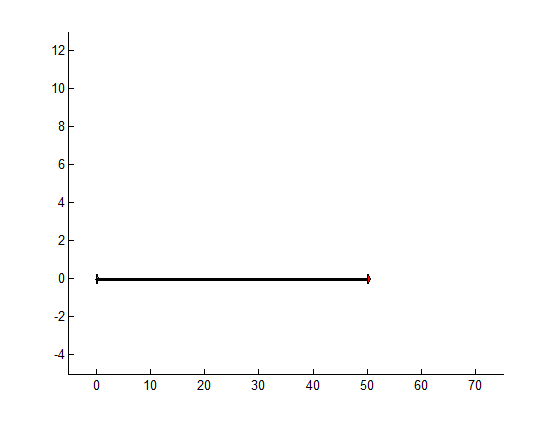
end

drawRobot(rob, 0.5, 'r')

disp('Naturally the only one to affect this robot on this straight path along the x axis would be the bias on the x axis. The y axis bias does nothing and there is no z axis to even implement a bias on')





****

# Problem 3:

clear all

close all

clc

% reads in the laser range finder data

% the first column of the data denotes the angle

% the second column of the data denotes the range or distance in mm

data = load('lineData.txt');

% converts the data in polar coordinates to X and Y coordinates

Xdata = data(:,2).\*cos(data(:,1));

Ydata = data(:,2).\*sin(data(:,1));

% plots the data

plot(Xdata, Ydata, 'r+')

axis([-1000 1500 -1000 1500])

hold on

%% We will first separate the data set into M different clusters. We will

%% then try to fit lines to each individual cluster

tol = 100; % if two consecutive points are more than 100 mm (10 cm) apart, start a new cluster

K = 1; % initialize the number of the clusters

clusterInd = zeros(length(Xdata),2); % initialize the array that will contain the start and end indices for each cluster

% the indices are to index into Xdata and Ydata

% since we don't know how large M will get, we will

% initialize clusterInd to have the same rows as Xdata

clusterInd(1,1) = 1; % setting the start index for the first cluster to be 1

for i = 2:size(data,1)

d = sqrt((Xdata(i)-Xdata(i-1))^2 + (Ydata(i)-Ydata(i-1))^2);

if d > tol

% start new cluster

clusterInd(K,2) = i-1;

K = K + 1;

clusterInd(K,1) = i;

end

end

clusterInd = clusterInd(clusterInd(:,1)>0,:); % this clips off all the excess rows that only contains 0s

clusterInd(end,2) = size(data,1); % this sets the last index to be the last element in data

%% Now that we have the clusters, let's fit the lines

lineCoeff = zeros(K,3); % initialize the vector to store the coefficients for the K line equations

lse = zeros(K,1); % initialize the vector to store the error of the K line fits

% main line fitting code begins here

n=2;

for i = 1:K

q = clusterInd(i,2)-clusterInd(i,1)+1;

M = zeros(q,3);

M(:,1) = Xdata(clusterInd(i,1):clusterInd(i,2));

M(:,2) = Ydata(clusterInd(i,1):clusterInd(i,2));

M(:,3) = 1;

[U,S,V] = svd(M);

lineCoeff(i,:) = V(:,3)';

lse = S(3,3);

n = n+1;

end

% the following plots each of the lines

for i = 1:K

% computes the end points for each line segment

xMax = max(Xdata(clusterInd(i,1):clusterInd(i,2)));

xMin = min(Xdata(clusterInd(i,1):clusterInd(i,2)));

if abs(lineCoeff(i,2)) < 1e-4

xMax = -lineCoeff(i,3)/lineCoeff(i,1);

xMin = xMax;

yMax = max(Ydata(clusterInd(i,1):clusterInd(i,2)));

yMin = min(Ydata(clusterInd(i,1):clusterInd(i,2)));

else

yMax = -lineCoeff(i,1)\*xMax/lineCoeff(i,2) - lineCoeff(i,3)/lineCoeff(i,2);

yMin = -lineCoeff(i,1)\*xMin/lineCoeff(i,2) - lineCoeff(i,3)/lineCoeff(i,2);

end

plot([xMin xMax], [yMin yMax], 'k-')

xMin = 0;

xMax = 0;

yMin = 0;

yMax = 0;

end

hold off



# Problem 4:

clear all

close all

clc

% reads in the laser range finder data

% the first column of the data denotes the angle

% the second column of the data denotes the range or distance in mm

data = load('circData.txt');

% converts the data in polar coordinates to X and Y coordinates

Xdata = data(:,2).\*cos(data(:,1));

Ydata = data(:,2).\*sin(data(:,1));

xm = mean(Xdata);

ym = mean(Ydata);

% plots the data

figure(1)

plot(Xdata, Ydata, 'r+')

axis([-750 1250 -1000 1000])

hold on

%% We will first separate the data set into M different clusters. We will

%% then try to fit circles to each individual cluster

tol = 100; % if two consecutive points are more than 100 mm (10 cm) apart, start a new cluster

K = 1; % initialize the number of the clusters

clusterInd = zeros(length(Xdata),2); % initialize the array that will contain the start and end indices for each cluster

% the indices are to index into Xdata and Ydata

% since we don't know how large M will get, we will

% initialize clusterInd to have the same rows as Xdata

clusterInd(1,1) = 1; % setting the start index for the first cluster to be 1

for i = 2:size(data,1)

d = sqrt((Xdata(i)-Xdata(i-1))^2 + (Ydata(i)-Ydata(i-1))^2);

if d > tol

% start new cluster

clusterInd(K,2) = i-1;

K = K + 1;

clusterInd(K,1) = i;

end

end

clusterInd = clusterInd(clusterInd(:,1)>0,:); % this clips off all the excess rows that only contains 0s

clusterInd(end,2) = size(data,1); % this sets the last index to be the last element in data

%% Now that we have the clusters, let's fit the circles

circCoeff = zeros(K,3); % initialize the vector to store the coefficients for the M line equations

lse = zeros(K,1); % initialize the vector to store the error of the M line fits

% main line fitting code begins here

for i = 1:K

q = clusterInd(i,2)-clusterInd(i,1)+1;

u = Xdata(clusterInd(i,1):clusterInd(i,2));

v = Ydata(clusterInd(i,1):clusterInd(i,2));

ui = u - mean(u);

vi = u - mean(v);

ui2 = sum(ui.^2);

ui3 = sum(ui.^3);

vi2 = sum(vi.^2);

vi3 = sum(vi.^3);

uivi = sum(ui)\*sum(vi);

uivi2 = sum(ui.\*vi.^2);

viui2 = sum(ui.^2)\*sum(vi);

Su = sum(ui);

Suu = sum(ui2);

Suuu = sum(ui3);

Sv = sum(vi);

Svv = sum(vi2);

Svvv = sum(vi3);

Suv = sum(uivi);

Suvv = sum(uivi2);

Svuu = sum(viui2);

uc = (.5\*(Suv\*(Svvv+Svuu)-(Suuu+Suvv)\*Svv))/(Suu\*Svv-Suv^2);

vc = (.5\*(Suvv+Svuu)-uc\*Suv)/(Svv);

alpha = uc^2+vc^2+(Suu+Svv)/q;

xc = uc+xm;

yc = vc+ym;

circCoeff(i,:) = [xc;yc;alpha];

end

% the following plots each of the lines

for i = 1:K

fprintf('I am here %d \n', i);

theta = linspace(0,2\*pi,50);

x = circCoeff(i,1) + circCoeff(i,3)\*cos(theta);

y = circCoeff(i,2) + circCoeff(i,3)\*sin(theta);

plot(x, y, 'k-')

end

hold off

