

MTRX5700  
EXPERIMENTAL ROBOTICS

---

**Assignment 3**

---

*Author(s):*

KAUSTHUB KRISHNAMURTHY

JAMES FERRIS

SACHITH GUNAWARDHANA

*SID:*

312086040

311220045

440623630

*Due:* May 6, 2015

# 1 Data Fusion

Our aim was to read the measured observation data from all four sources and fuse them appropriately in order to ascertain a real representation of the robot's path. This section of the report details the methods we employed to conduct this data fusion step. The Velocity observations would provide internal data of the robot's velocities (linear and turn rate) so that its position can be segmentally ascertained using the dead reckon approximation method. However in order to account for issues such as slip (on the wheels or motors) and observational error we need to incorporate compass and GPS readings and use corrective alterations to our position prediction. Using the retro-reflective beacon readings and positions would have added an extra level of position determination by allowing us to use a local triangulation & trilateration method using the angle & distance of the beacons with respect to the robot.

## 1.a Program Flow & Observation Scheduling

The program requires a structure that would allow us to access different courses of action based on the input received at any given time because the order of observations being made is critical to the flow of the program. We can consider this to be four different lists of inputs that need to be scheduled which can be done with a set of flag variables which was stored in an array where if a particular type of observational event was perceived to occur a corresponding array element would be driven high. On testing this element we can allow sections of the code to be run or block them from running so that only the necessary processes occur at any given time. We detect the "next" action by storing all data in a matrix which is ordered by time and an array storing the index (with respect to that matrix) of the next event and these values increment as we access each observation. Using these indices we put the timestamps of each next event into an array, detect the smallest timestamps and test each event time (for coincidental events) for procedures that need to occur.

Iters stores the index of the next even in each observational data set that we are yet to "perceive"

$$iters = [itersVel \quad itersGPS \quad itersComp \quad itersmeasurementn]$$

time stores the timestamps for the next index in each observational data set. This is so that we can use the *min()* function in matlab as opposed to writing the min finding code ourselves (a more elegant solution for adapting to a system with more sensors).

$$time = [timestamp(s) \quad measurement1 \quad measurement2 \quad \dots \quad measurementn]$$

Figure 1: Data I/O Process

### 1.a.i Data I/O

Having to store each set of data separately while still having easily accessible data (for ease of writing the code and minimization of code runtime) we needed to filter the data into a series of structural arrays with the following template:

*ObservationDataStructure* = [*timestampVel(s)* *timestampGPS* *timestampComp* *timestamp...* *measurementn*]

Process Descriptions:

- 1) Parse Input: Takes in the columns of data with space delimiting input saving the first column to "Seconds", second column to "Microseconds", and each subsequent observation column into a Measurements vector. In Figure 2 we see the structural flow for an 2-parameter data set such as Velocity Observations where Velocity would be Measurements 1 and Turn Rate would be Measurements 2.
- 2) Merge Time Stamps: An implementation of the following equation:  $ts1 + ts2^{10^{-6}} - FirstTimeStamp$   
This effectively "zeroes" the events so that the very first event occurs at  $t = 0$

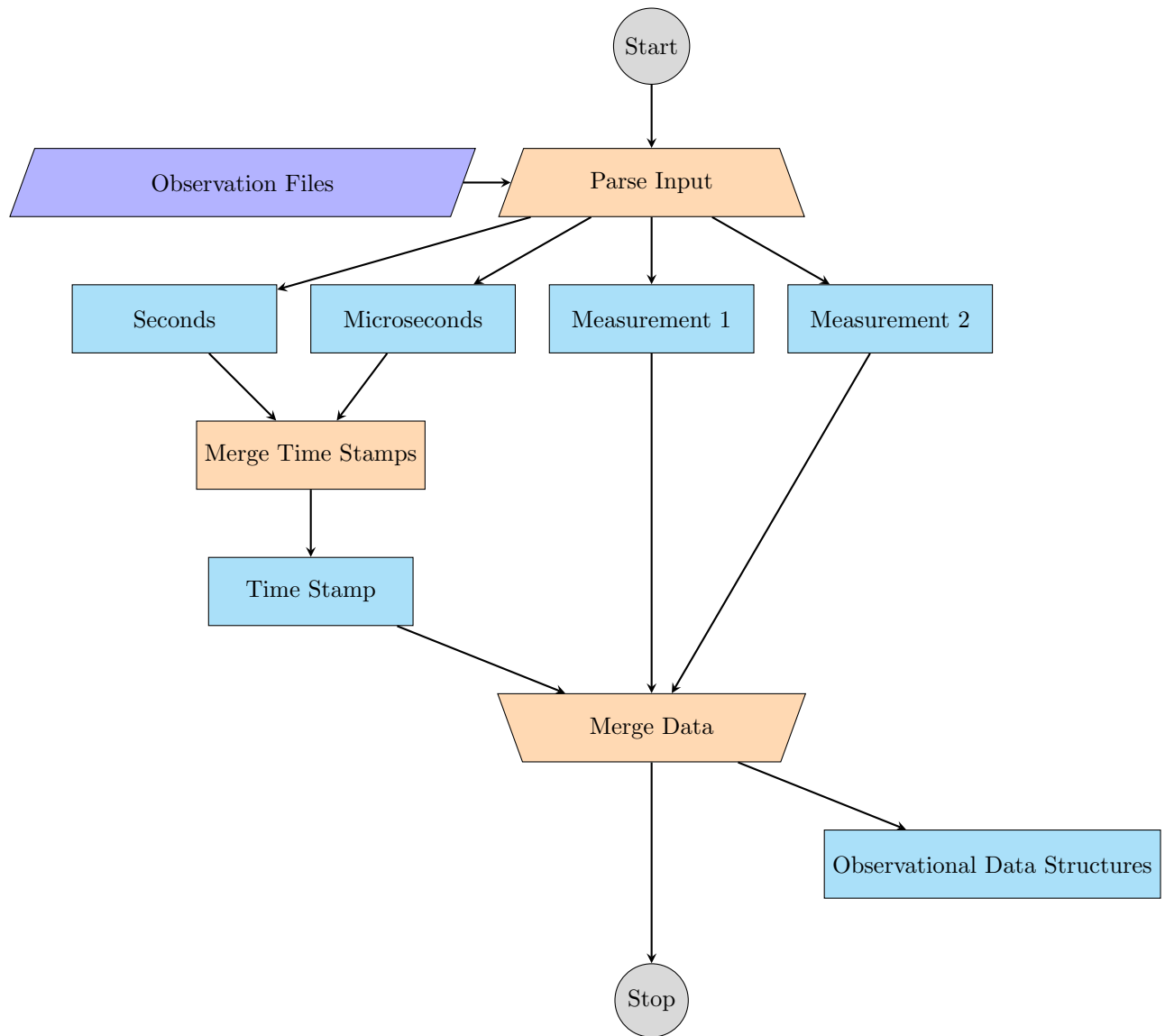


Figure 2: Data I/O Process

## 1.b Prediction Stage Implementation

## 1.c Observation Stage Implementation

## 1.d Update Stage Implementation

## 1.e Code Listing

### 1.e.i Demonstration Code

```
1 clear
2 % close all
3 clc
4
5 DEGREES = 180/pi;
6 RADIANS = pi/180;
7 SUN = 1.496*10^8;
8 % % Prediction Stage
9 % diary './qloutput'
10 %Load observational data
11 velocityObs = load('velocityObs.txt');
12 positionObs = load('positionObs.txt');
13 compassObs = load('compassObs.txt');
14 laserObs = load('laserObs.txt');
15
16 %Get velocity data
17 time1 = velocityObs(:,1) + (velocityObs(:,2)*10^-6) - 1115116000;%get in microseconds
18 velocity = velocityObs(:,3);
19 turnRate = velocityObs(:,4);
20
21 velObs = [time1 velocity turnRate];
22
23 %Get GPS position data
24 time2 = positionObs(:,1) + (positionObs(:,2)*10^-6) - 1115116000;
25 xPos = positionObs(:,3);
26 yPos = positionObs(:,4);
27
28 posObs = [time2 xPos yPos];
29
30 %Get GPS compass data
31 time3 = compassObs(:,1) + (compassObs(:,2)*10^-6) - 1115116000;
32 heading = compassObs(:,3);
33
34 compObs = [time3 heading];
35
36 % % %get laser data
37 time4 = laserObs(:,1) + (laserObs(:,2)*10^-6) - 1115116000;
38 % % i = 1;
39 % % j = 4;
40 % % sizeLas = size(laserObs);
41 % % lasObs = zeros(sizeLas(1, 2));
42 % % while(i <= sizeLas(1))
43 % %     lasObs(i, 1) = [time4(i)];
44 % %     while(j<=sizeLas(2))
45 % %         lasObs(i, j) = laserObs(i, j);
46 % %         lasObs(i, j-1) = laserObs(i, j-1);
47 % %
48 % %         j = j + 2;
49 % %     end
50 % %
51 % %     i = i + 1;
52 % % end
53 % % size(time4)
54
55
56
57 %%postInput parsing
58
59 alphaP = 0.1;
60 alphaTH = 0.1;
61
62 lastTime = 0;
63 deltaT = 0;
64
65 latestVel = 0;
```

```

66 latestTurnRate = 0;
67
68 ourX = 0;
69 ourY = 0;
70 ourHeading = 0;
71
72 indLengths = [length(time1), length(time2), length(time3), length(time4)];
73 maxIters = max(indLengths);
74
75 % output = [lastTime, ourX, ourY, ourHeading];
76 output = zeros(maxIters, 6);
77
78 % deadreckonedpts = zeros(maxIters, 3);
79
80 %iters [velInd, posInd, compInd, lasInd];
81 iters = [2, 2, 2, 2];
82 runFlags = [0, 0, 0, 0];
83 loopFlag = 1;
84 loopCount = 2;
85 %%loop starts
86 while(loopFlag == 1)
87     loopCount = loopCount + 1
88     time = [time1(iters(1)), time2(iters(2)), time3(iters(3)), time4(iters(4)),];
89     nextT = min(time);
90
91
92     for i = 1:4
93         if time(i) == nextT
94             runFlags(i) = 1;
95         else
96             runFlags(i) = 0;
97         end
98     end
99
100     %if velocityobs
101     if(runFlags(1) == 1)
102         deltaT = time1(iters(1)) - lastTime;
103         latestVel = velObs(iters(1),2);%
104         latestTurnRate = velObs(iters(1),3);%
105         pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
106         ourX = pr(1);
107         ourY = pr(2);
108         ourHeading = pr(3);
109
110         deadreckonedpts(1) = ourX;
111         deadreckonedpts(2) = ourY;
112         deadreckonedpts(3) = ourHeading;
113         lastTime = time1(iters(1));%
114         runFlags(1) = 0;
115         if iters(1) == length(time1)
116             time1(iters(1)) = SUN;
117         else
118             iters(1) = iters(1) + 1;
119         end
120     end
121
122     % % if GPS
123     if(runFlags(2) == 1)
124         deltaT = time2(iters(2)) - lastTime;
125         pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
126         gUpd = updateStageGPS(pr(1), pr(2), posObs(iters(2),2), posObs(iters(2),3), alphaP); %xvobs
127         ↪ and yvobs need to come from the file
128         ourX = gUpd(1);
129         ourY = gUpd(2);
130         ourHeading = pr(3);
131         lastTime = time2(iters(2));%
132         runFlags(2) = 0;
133         if time2(iters(2)) == length(time2)
134             time2(iters(2)) = SUN;
135         else
136             iters(2) = iters(2) + 1;

```



```

136         end
137     end
138     %
139     % % if compass
140     if(runFlags(3) == 1)
141         deltaT = time3(iters(3)) - lastTime;
142         pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
143         cUpd = updateStageCompass(pr(3), compObs(iters(3),2), alphaTH);
144         ourX = pr(1);
145         ourY = pr(2);
146         ourHeading = cUpd;
147         lastTime = time3(iters(3));
148         runFlags(3) = 0;
149         if iters(3) == length(time3)
150             time3(iters(3)) = SUN;
151         else
152             iters(3) = iters(3) + 1;
153         end
154     end
155     %
156     % % if laser data
157     if(runFlags(4) == 1)
158         % % fill this in
159         runFlags(4) = 0;
160         time4(iters(4)) = SUN; %remove this line
161     end
162
163     %check loop
164     if(time1(iters(1)) == SUN)
165         if(time2(iters(2)) == SUN)
166             if(time3(iters(3)) == SUN)
167                 if(time4(iters(4)) == SUN)
168                     loopFlag = 0;
169                 end
170             end
171         end
172     end
173
174     %plot stuff
175     hold on
176     title('Robot Path');
177     xlabel('x-axis');
178     ylabel('y-axis');
179     % legend('Dead Reckoning');
180     % drawnow
181
182     plot(ourX, ourY, 'r. ');
183     output(loopCount, 1) = lastTime;
184     output(loopCount, 2) = ourX;
185     output(loopCount, 3) = ourY;
186     output(loopCount, 4) = ourHeading;
187     output(loopCount, 5) = latestVel;
188     output(loopCount, 6) = latestTurnRate;
189     end
190
191     % diary ON
192     % output
193     % diary OFF
194
195     output(:,1) = output(:,1); %+ 1115116000;

```

## 1.e.ii Development Code

This code listing includes a partially complete implementation of fusing the data from the Laser Range Finder detection of retro reflective beacons.

```
1 clear
2 close all
3 clc
4
5 DEGREES = 180/pi;
6 RADIANS = pi/180;
7 SUN = 1.496*10^8;
8
9 %Load observational data
10 velocityObs = load('velocityObs.txt');
11 positionObs = load('positionObs.txt');
12 compassObs = load('compassObs.txt');
13 laserObs = load('laserObs.txt');
14 laserFeat = load('laserFeatures.txt');
15
16 %get laser features
17 lasFeat = [laserFeat(:,1) laserFeat(:,2)];
18
19 %Get velocity data
20 time1 = velocityObs(:,1) + (velocityObs(:,2)*10^-6) - 1115116000;%get in microseconds
21 velocity = velocityObs(:,3);
22 turnRate = velocityObs(:,4);
23
24 velObs = [time1 velocity turnRate];
25
26 %Get GPS position data
27 time2 = positionObs(:,1) + (positionObs(:,2)*10^-6) - 1115116000;
28 xPos = positionObs(:,3);
29 yPos = positionObs(:,4);
30
31 posObs = [time2 xPos yPos];
32
33 %Get GPS compass data
34 time3 = compassObs(:,1) + (compassObs(:,2)*10^-6) - 1115116000;
35 heading = compassObs(:,3);
36
37 compObs = [time3 heading];
38
39 % % %get laser data
40 time4 = laserObs(:,1) + (laserObs(:,2)*10^-6) - 1115116000;
41
42 f1=1;
43
44 range = zeros(length(laserObs), (size(laserObs,2)-2)/2);
45 intensity = zeros(length(laserObs), (size(laserObs,2)-2)/2);
46
47 %Extracting range & intensity data from LaserObs
48 for i=1:length(laserObs)
49     for f2=3:2:size(laserObs,2)
50         range(i,f1)=laserObs(i,f2);
51         intensity(i,f1)=laserObs(i,f2+1);
52         f1=f1+1;
53     end
54     f1=1;
55 end
56
57
58 % postInput parsing
59
60 alphaP = 0.1;
61 alphaTH = 0.1;
62
63 lastTime = 0;
64 deltaT = 0;
65
```

```

66 latestVel = 0;
67 latestTurnRate = 0;
68
69 ourX = 0;
70 ourY = 0;
71 ourHeading = 0;
72
73 indLengths = [length(time1), length(time2), length(time3), length(time4)];
74 maxIters = max(indLengths);
75
76 % output = [lastTime, ourX, ourY, ourHeading];
77 output = zeros(maxIters, 6);
78
79 % deadreckonedpts = zeros(maxIters, 3);
80
81 %iters [velInd, posInd, compInd, lasInd];
82 iters = [2, 2, 2, 2];
83 runFlags = [0, 0, 0, 0];
84 loopFlag = 1;
85 loopCount = 2;
86 %%loop starts
87 while(loopFlag == 1)
88     loopCount = loopCount + 1
89     time = [time1(iters(1)), time2(iters(2)), time3(iters(3)), time4(iters(4)),];
90     nextT = min(time);
91
92
93     for i = 1:4
94         if time(i) == nextT
95             runFlags(i) = 1;
96         else
97             runFlags(i) = 0;
98         end
99     end
100
101     %if velocityobs
102     if(runFlags(1) == 1)
103         deltaT = time1(iters(1)) - lastTime;
104         latestVel = velObs(iters(1),2);%
105         latestTurnRate = velObs(iters(1),3);%
106         pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
107         ourX = pr(1);
108         ourY = pr(2);
109         ourHeading = pr(3);
110
111         %         deadreckonedpts(1) = ourX;
112         %         deadreckonedpts(2) = ourY;
113         %         deadreckonedpts(3) = ourHeading;
114         lastTime = time1(iters(1));%
115         runFlags(1) = 0;
116         if iters(1) == length(time1)
117             time1(iters(1)) = SUN;
118         else
119             iters(1) = iters(1) + 1;
120         end
121     end
122
123     % % if GPS
124     if(runFlags(2) == 1)
125         deltaT = time2(iters(2)) - lastTime;
126         pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
127         gUpd = updateStageGPS(pr(1), pr(2), posObs(iters(2),2), posObs(iters(2),3), alphaP); %xvobs
128         ↪ and yvobs need to come from the file
129         ourX = gUpd(1);
130         ourY = gUpd(2);
131         ourHeading = pr(3);
132         lastTime = time2(iters(2));%
133         runFlags(2) = 0;
134         if iters(2) == length(time2)
135             time2(iters(2)) = SUN;
136         else

```

```

136         iters(2) = iters(2) + 1;
137     end
138 end
139 %
140 % % if compass
141 if(runFlags(3) == 1)
142     deltaT = time3(iters(3)) - lastTime;
143     pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
144     cUpd = updateStageCompass(pr(3), compObs(iters(3),2), alphaTH);
145     ourX = pr(1);
146     ourY = pr(2);
147     ourHeading = cUpd;
148     lastTime = time3(iters(3));
149     runFlags(3) = 0;
150     if iters(3) == length(time3)
151         time3(iters(3)) = SUN;
152     else
153         iters(3) = iters(3) + 1;
154     end
155 end
156 %
157 % % if laser data
158 if(runFlags(4) == 1)
159     % find beacons
160     deltaT = time4(iters(4)) - lastTime;
161     pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
162     ourX = pr(1);
163     ourY = pr(2);
164     ourHeading = pr(3);
165
166     intensityTrav = 1;
167     beaconCentreInds = 0;
168     %find beacon centres
169     while(intensityTrav <= 361)
170         if(intensity(intensityTrav) == 1)
171             iStart = intensityTrav;
172             intensityTrav = intensityTrav + 1;
173             while(intensity(intensityTrav) == 1 && (abs(range(intensityTrav) - range(
174                 ↪ intensityTrav + 1) < 1)))
175                 intensityTrav = intensityTrav + 1;
176             end
177             iEnd = intensityTrav - 1;
178             iMid = (iEnd - iStart)/2;
179             beaconCentreInds = cat(2, beaconCentreInds, iMid);
180         else
181             intensityTrav = intensityTrav + 1;
182         end
183     end
184     % project to real world
185     mb = matchBeacons(ourX, ourY, ourHeading, lasFeat, range(iters(4),:), beaconCentreInds);
186     ourX = mb(1);
187     ourY = mb(2);
188     ourHeading = mb(3);
189
190     lastTime = time4(iters(4));
191     runFlags(4) = 0;
192     if iters(4) == length(time4)
193         time3(iters(4)) = SUN;
194     else
195         iters(4) = iters(4) + 1;
196     end
197 end
198 %check loop
199 if(time1(iters(1)) == SUN)
200     if(time2(iters(2)) == SUN)
201         if(time3(iters(3)) == SUN)
202             if(time4(iters(4)) == SUN)
203                 loopFlag = 0;
204             end
205         end

```

```
206         end
207     end
208
209     %plot stuff
210     hold on
211     title('Robot Path');
212     xlabel('x-axis');
213     ylabel('y-axis');
214     % legend('')
215     % drawnow
216
217     plot(ourX, ourY, 'r.');
218     output(loopCount, 1) = lastTime;
219     output(loopCount, 2) = ourX;
220     output(loopCount, 3) = ourY;
221     output(loopCount, 4) = ourHeading;
222     output(loopCount, 5) = latestVel;
223     output(loopCount, 6) = latestTurnRate;
224 end
```

## 2 Question 2

### 2.a Obtaining Obstacle Location from Laser Data

First we start by taking the data output from question one, which consisted of the robot  $x - y$  coordinates in the world coordinate system, as well as the velocity and turn rate for that particular timestamp, for each timestamp that occurs in the Velocity observation data, the Compass data and the GPS data.

Similar to the way question one works, we combine this data with the Laser observation data by comparing timestamps, using the Prediction Stage equation to estimate the robots  $x - y$  coordinates at the time that the laser data was generated. Combining the robots  $x - y$  world coordinates with the relative position of the obstacles obtained from the Laser data.

```
1 clear
2 clc
3 close all
4
5 DEGREES = 180/pi;
6 RADIANS = pi/180;
7
8
9 positionData = load('qloutput1.txt');
10
11 laserObs = load('laserObs.txt');
12
13 %Get output data
14 time1 = positionData(:,1);
15 Xpos = positionData(:,2);
16 Ypos = positionData(:,3);
17 heading = positionData(:,4);
18 velocity = positionData(:,5);
19 turnRate = positionData(:,6);
20
21 diary './q2Output4'
22
23 %Get laser data
24 time2 = laserObs(:,1) + (laserObs(:,2)*10^-6) - 1115116000;%get in microseconds
25
26
27 %Extracting range & intensity data from LaserObs
28
29 f1=1;
30 range = zeros(length(laserObs), (size(laserObs,2)-2)/2);
31 % intensity = zeros(length(laserObs), (size(laserObs,2)-2)/2);
32
33 %Extracting range & intensity data from LaserObs
34 for i=1:length(laserObs)
35     for f2=3:2:size(laserObs,2)
36         % if(laserObs(i,f2) < 8)
37             range(i,f1)=laserObs(i,f2);
38             % intensity(i,f1)=laserObs(i,f2+1);
39             f1=f1+1;
40         % end
41     end
42     f1=1;
43 end
44
45 lasersX = 0;
46 lasersY = 0;
47
48 % alphaP = 0.5;
49 % alphaTH = 0.5;
50
51 lastTime = 0;
52 deltaT = 0;
53
54 latestVel = 0;
55 latestTurnRate = 0;
```

```

56
57 ourX = 0;
58 ourY = 0;
59 ourHeading = 0;
60
61 indLengths = [length(time1), length(time2)];
62 maxIters = max(indLengths);
63
64 interval = 20;
65
66 %iters [velInd, posInd, compInd, lasInd];
67 iters = [2, 2];
68 runFlags = [0, 0];
69 loopFlag = 1;
70 loopCount = 2;
71 %%loop starts
72
73
74
75 while(loopFlag == 1)
76     loopCount = loopCount + 1;
77     time = [time1(iters(1)), time2(iters(2))];
78     nextT = min(time);
79
80
81     for i = 1:2
82         if time(i) == nextT
83             runFlags(i) = 1;
84         else
85             runFlags(i) = 0;
86         end
87     end
88
89     %if Positiondata
90     if(runFlags(1) == 1)
91         deltaT = time1(iters(1)) - lastTime;
92         latestVel = velocity(iters(1));
93         latestTurnRate = turnRate(iters(1));
94         ourX = Xpos(iters(1));
95         ourY = Ypos(iters(1));
96         ourHeading = heading(iters(1));
97
98         lastTime = time1(iters(1));%
99         runFlags(1) = 0;
100         if iters(1) >= length(time1)-interval
101             time1(iters(1)) = 1.496*10^8;
102         else
103             iters(1) = iters(1) + interval;
104         end
105     end
106
107     %% if laser
108     if(runFlags(2) == 1)
109         deltaT = time2(iters(2)) - lastTime;
110         pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
111         ourX = pr(1);
112         ourY = pr(2);
113         ourHeading = pr(3);
114         lastTime = time2(iters(2));%
115         runFlags(2) = 0;
116
117
118         % Get X,Y coordinates for laser ob data
119
120         for i = 1:size(range,2)
121             if (range(iters(2),i) < 8.0 && range(iters(2),i) > 0.0001)
122                 lasersX = ourX + range(iters(2),i)*cos(((i-1)*0.5)*RADIANS+ourHeading);
123                 lasersY = ourY + range(iters(2),i)*sin(((i-1)*0.5)*RADIANS+ourHeading);
124                 diary ON
125                 fprintf('%d\t%d\n', lasersX, lasersY);
126                 diary OFF

```

```

127         end
128     end
129
130     %increment
131     if iters(2) >= length(time2)-interval
132         time2(iters(2)) = 1.496*10^8;
133     else
134         iters(2) = iters(2) + interval;
135     end
136
137
138 end
139
140 %check loop
141 if (time1(iters(1)) == 1.496*10^8)
142     if (time2(iters(2)) == 1.496*10^8)
143         loopFlag = 0;
144     end
145 end
146
147 %plot stuff
148 % hold on
149 % title('Obstacles');
150 % xlabel('x-axis');
151 % ylabel('y-axis');
152 % legend('')
153 % drawnow
154 end

```

For initial tests to attempt to identify obstacles, any range reading from the Laser data that was less than eight (the maximum range of the sensor) was considered an obstacle. It was intended to apply filters to this data once the Occupancy Grid had been generated. Until then, raw data would be used.

## 2.b Generate Occupancy Grid

To generate the Occupancy Grid, we took the Data set of the X-Y coordinates of all 'Obstacles' detected, and determined the difference between the minimum and maximum  $x$  and  $y$  values detected. Given a user defined size for the occupancy grid, we could then determine the  $x - y$  range that corresponded to a grid location. Then for every Obstacle  $x - y$  coordinate we could determine the grid location it corresponded to, incrementing the grid value to increase the weighting, which would indicate the likelihood of an obstacle being in that region. Some experimenting with the grid size using the data for the Robot path generated in question one indicated that a grid size of  $200 \times 200$  would be best, as this resulted in a grid map that, while not extremely sharp, was also not extremely blurred. Ideally, the grid would be vague enough to generalise a position for the obstacles, but not so clear that it simply resulted in a plot of every possible obstacle coordinates. See the figures below:



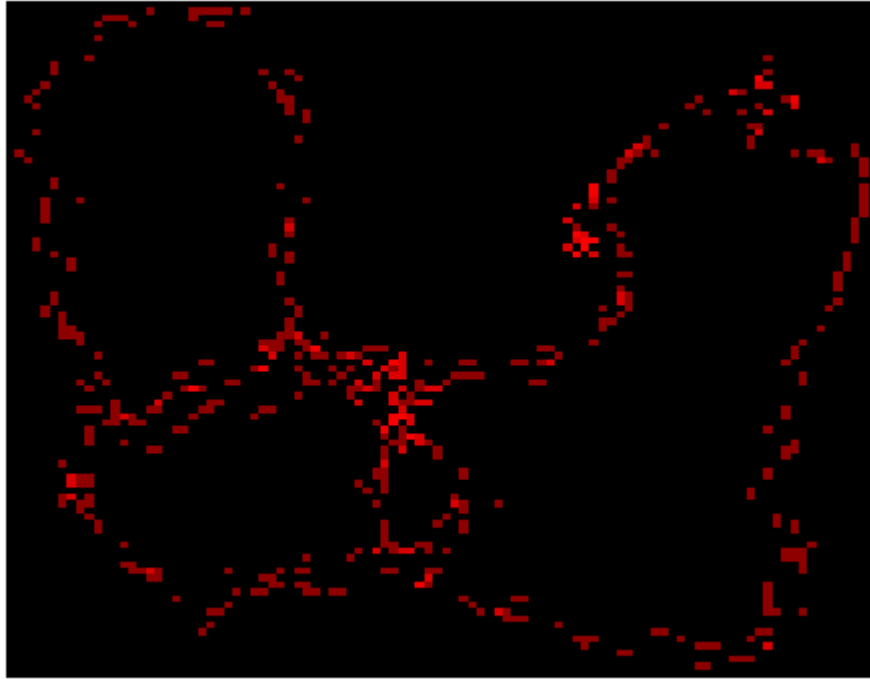


Figure 3: Grid size =  $100 \times 100$

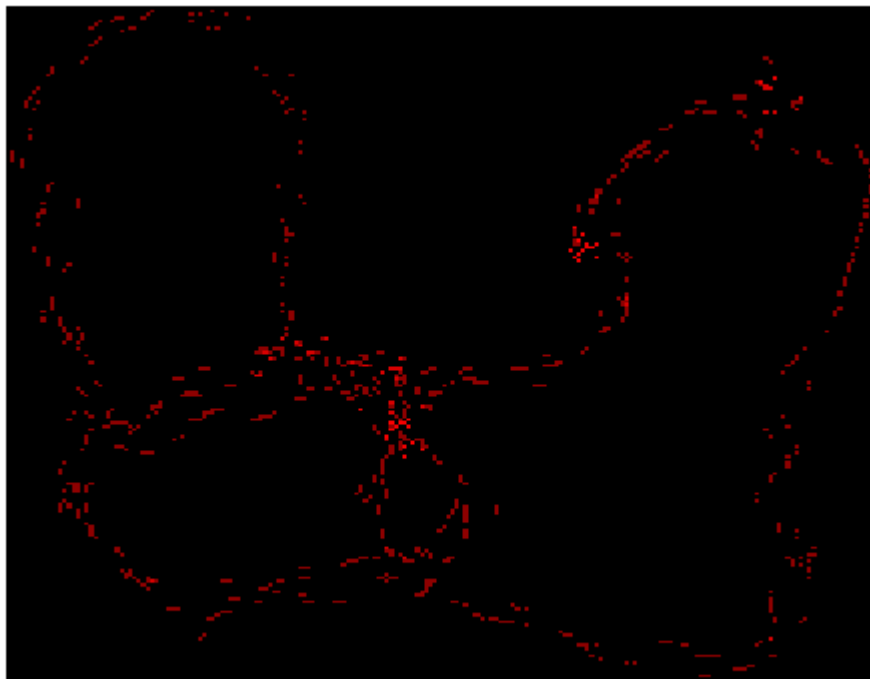


Figure 4: Grid size =  $200 \times 200$

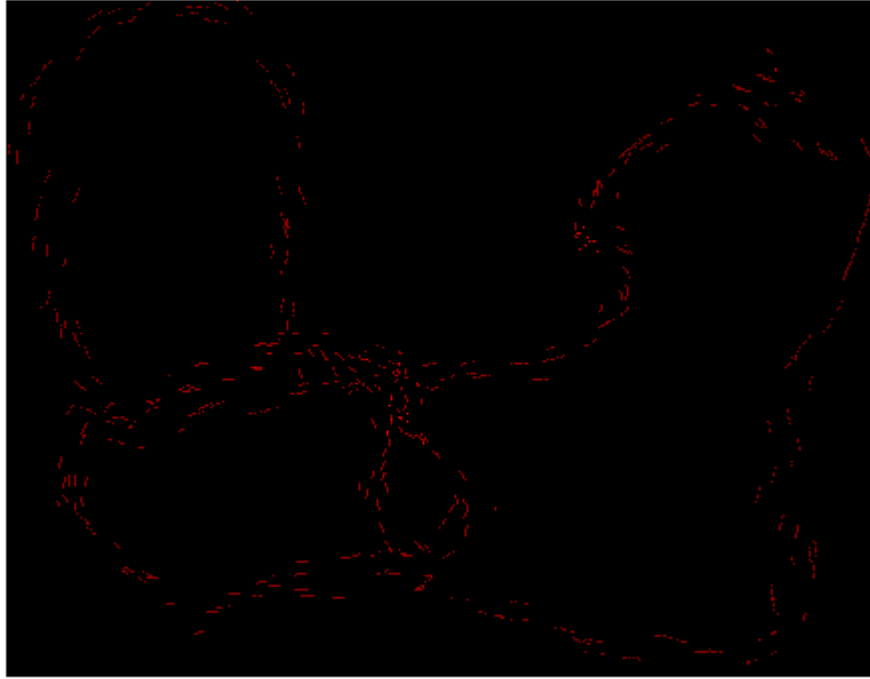


Figure 5: Grid size =  $500 \times 500$

## 2.c Results

### 3 Question 3

## Code Listing

See Appendix A [3] for all code used.

## 4 Appendix A

### 4.1 Question 1

## 4.2 Question 2

### 4.2.i obtainObstacles

### 4.2.ii obtainObstacles<sub>v</sub>2

```
1 clear
2 clc
3 close all
4
5 DEGREES = 180/pi;
6 RADIANS = pi/180;
7
8
9 positionData = load('qloutput1.txt');
10
11 laserObs = load('laserObs.txt');
12
13 %Get output data
14 time1 = positionData(:,1);
15 Xpos = positionData(:,2);
16 Ypos = positionData(:,3);
17 heading = positionData(:,4);
18 velocity = positionData(:,5);
19 turnRate = positionData(:,6);
20
21 % diary './q2.2Output1'
22
23 %Get laser data
24 time2 = laserObs(:,1) + (laserObs(:,2)*10^-6) - 1115116000;%get in microseconds
25
26 lasersX = 0;
27 lasersY = 0;
28
29 % alphaP = 0.5;
30 % alphaTH = 0.5;
31
32 lastTime = 0;
33 deltaT = 0;
34
35 latestVel = 0;
36 latestTurnRate = 0;
37
38 ourX = 0;
39 ourY = 0;
40 ourHeading = 0;
41
42 indLengths = [length(time1), length(time2)];
43 maxIters = max(indLengths);
44
45 interval = 20;
46
47 %iters [velInd, posInd, compInd, lasInd];
48 iters = [2, 2];
49 runFlags = [0, 0];
50 loopFlag = 1;
51 loopCount = 2;
52 %%loop starts
53
54 xPos = zeros(1);
55 yPos = zeros(1);
56
57 while(loopFlag == 1)
58     loopCount = loopCount + 1;
59     time = [time1(iters(1)), time2(iters(2))];
60     nextT = min(time);
61
62
63     for i = 1:2
64         if time(i) == nextT
```

```

65         runFlags(i) = 1;
66     else
67         runFlags(i) = 0;
68     end
69 end
70
71 %if Positiondata
72 if(runFlags(1) == 1)
73     deltaT = time1(iters(1)) - lastTime;
74     latestVel = velocity(iters(1));
75     latestTurnRate = turnRate(iters(1));
76     ourX = Xpos(iters(1));
77     ourY = Ypos(iters(1));
78     ourHeading = heading(iters(1));
79
80     lastTime = time1(iters(1));%
81     runFlags(1) = 0;
82     if iters(1) >= length(time1)-interval
83         time1(iters(1)) = 1.496*10^8;
84     else
85         iters(1) = iters(1) + interval;
86     end
87 end
88
89 % % if laser
90 if(runFlags(2) == 1)
91     deltaT = time2(iters(2)) - lastTime;
92     pr = predictionStage(ourX, ourY, ourHeading, deltaT, latestTurnRate, latestVel);
93     ourX = pr(1);
94     ourY = pr(2);
95     ourHeading = pr(3);
96     lastTime = time2(iters(2));%
97     runFlags(2) = 0;
98
99
100     xpoint = zeros(1);
101     ypoint = zeros(1);
102     for j = 4:2:size(laserObs,2)
103         range = laserObs(iters(2),j-1);
104         bearing = ((j)/2 - 90)*pi/180;
105         if (range < 8.0)
106             xpoint = [xpoint ourX+range*cos(bearing + ourHeading)];
107             ypoint = [ypoint ourY+range*sin(bearing + ourHeading)];
108         end
109
110     end
111
112     xPos = [xPos xpoint];
113     yPos = [yPos ypoint];
114
115     plot(xpoint(:), ypoint(:), '.');
116
117 %increment
118 if iters(2) >= length(time2)-interval
119     time2(iters(2)) = 1.496*10^8;
120 else
121     iters(2) = iters(2) + interval;
122 end
123
124
125 end
126
127 %check loop
128 if(time1(iters(1)) == 1.496*10^8)
129     if(time2(iters(2)) == 1.496*10^8)
130         loopFlag = 0;
131     end
132 end
133
134 %plot stuff
135 % hold on

```

```

136 % title('Obstacles');
137 % xlabel('x-axis');
138 % ylabel('y-axis');
139 % legend('')
140 drawnow
141 % pause
142 end
143
144 xpoint(1) = [];
145 ypoint(1) = [];
146
147 xPos(1) = [];
148 yPos(1) = [];
149
150 xMin = min(xPos);
151 xMax = max(xPos);
152
153 yMin = min(yPos);
154 yMax = max(yPos);
155
156
157 gridSize = 100;
158
159 grid = zeros(gridSize);
160
161
162 yDiff = (yMax - yMin)/(gridSize - 2);
163 xDiff = (xMax - xMin)/(gridSize - 2);
164
165 for i = 1:length(xPos)
166     tmpX = xPos(i);
167     tmpY = yPos(i);
168     j = 1;
169     while (tmpX > xMin)
170         tmpX = tmpX - xDiff;
171         j = j + 1;
172     end
173     k = 1;
174     while (tmpY > yMin)
175         tmpY = tmpY - yDiff;
176         k = k + 1;
177     end
178
179     grid(j,k) = grid(j,k) + 1;
180 end
181
182 HeatMap(grid);

```

#### 4.2.iii generateOccupancyGrid

```

1 clear
2 close all
3 clc
4
5 % positionData = load('qloutput1.txt');
6 positionData = load('q2Output4.txt');
7
8
9 % xPos = positionData(:,2);
10 % yPos = positionData(:,3);
11
12 xPos = positionData(:,1);
13 yPos = positionData(:,2);
14
15 xMin = min(xPos);
16 xMax = max(xPos);
17
18 yMin = min(yPos);

```



```
19 yMax = max(yPos);
20
21
22 gridSize = 100;
23
24 grid = zeros(gridSize);
25
26
27 yDiff = (yMax - yMin)/(gridSize - 2);
28 xDiff = (xMax - xMin)/(gridSize - 2);
29
30 for i = 1:length(xPos)
31     tmpX = xPos(i);
32     tmpY = yPos(i);
33     j = 1;
34     while (tmpX > xMin)
35         tmpX = tmpX - xDiff;
36         j = j + 1;
37     end
38     k = 1;
39     while (tmpY > yMin)
40         tmpY = tmpY - yDiff;
41         k = k + 1;
42     end
43
44     grid(j,k) = grid(j,k) + 1;
45 end
46
47 % imagesc(grid);
48 HeatMap(grid);
```

### 4.3 Question 3

#### Code Listing

See Appendix A [9.1]