1- 22
2- 22
3- 20
4- 20
5- 20
Bonus: 10

Steven Leo Id: 108567094
10/24/16 Mec 560

Mini Project 1: Obstacle Avoidance for Autonomous Robot

A maze was created to for the robot to traverse with a starting position of (1,1) and a final position (9,9). This maze can be seen in figure 1. The robot is defined as a point mass that can move up, down, left, right, and along any of the diagonals. The calculations for the distance between positions can be found below. Straight distance traveled is calculated using a movement to the right. Diagonal movement is calculated using a movement to the upper right.

$$distance = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2}$$

$$Straight\ Movment = \sqrt{(1-0)^2 + (0-0)^2} = 1$$

$$Diagonal\ Movment\ Distance = \sqrt{(1-0)^2 + (1-0)^2} = 1.41$$

For ease of calculation a conversion is used[1]. Both values are multiplied by 10 and rounded to the nearest whole number. Using whole numbers can decrease the computation required for any mathematical operations run on a computer.

$$Stright\ Movment = 10, Diagonal\ Movment\ Distace = 14$$

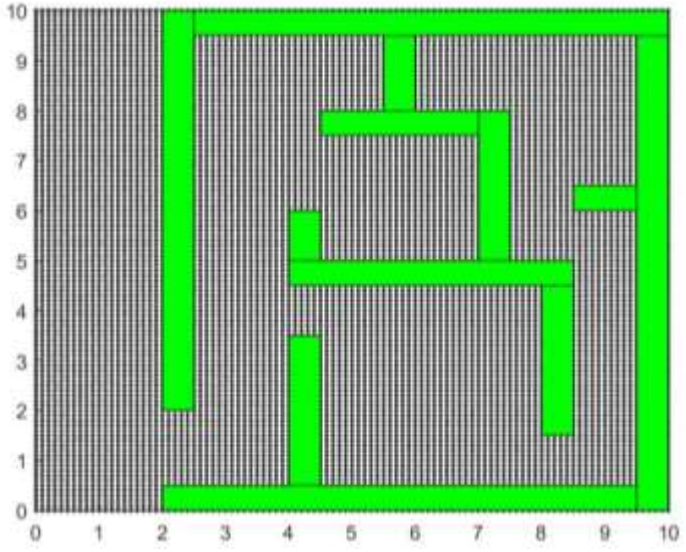


Figure 1: Maze

[1] http://www.policyalmanac.org/games/aStarTutorial.htm

In order to generate a sequence of waypoints an A* algorithm was used to solve the created maze. A* was selected because it does not need to know the maze before attempting a solution. In future projects A* will also prove a useful tool. It should be noted that the A* algorithm used in this project jump to the next available point when arriving at a solution regardless of location. A* can be used in a live path finding scenario if the only points available to be used are points located directly next to the previous position.[2]

With the trajectory found it can be seen that there are sharp turns in the path. In order to follow this path as it is the robot would have to be designed with zero point turning capabilities. This navigation method would mean the robots movement would be a series of straight lines and zero point turns. This is a laborious proses and would restrict the types of robots that can follow this path. In order to solve this issue, the path needs to be smoothed. This can be done by applying gradient decent. The result of the smoothing can be seen in figure 3. With how the system is currently defined, as the trajectory is smoothed using gradient decent, there is a chance of the new smoothed path to be defined in such a way that it crosses through an obstacle. To avoid this a buffer zone can be defined so that the robot does not run through an obstacle. When selecting the buffer zone, an estimate distance of about 0.30 was desired. This value was chosen to be more than half of the thickness of the mazes walls and would allow for sufficient space for when the system is smoothed and later tracked. After testing a number of buffers, a buffer of 0.20 was selected as it had the shortest travel time that satisfied the spacing requirement. The travel time was calculated by taking the distance between each waypoint and dividing by the desired constant speed of 1.5.

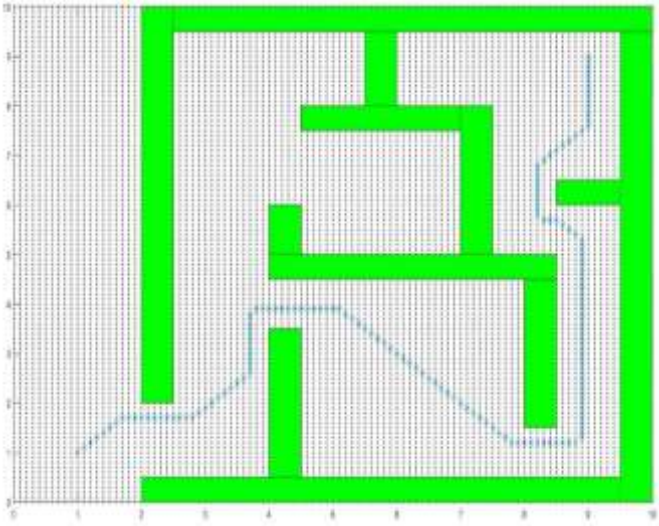
The effect of different buffer values can be seen in table 1. Figure 2 shows both the unsmooth and smooth solution using the A* algorithm and gradient decent method for smoothing.

Effect on Buffer in reference to Obstacle Edge at (8,1.5)			
Buffer	Δy to Obstacle edge	Number of Waypoints	Travel Time
0.10	0.1725	160	11.913
0.15	0.1863	162	12.155
0.20	0.2863	166	12.366
0.25	0.2863	171	12.637
0.30	0.3931	176	12.931

Table 1

[2] https://www.youtube.com/watch?v=KNXfSOx4eEE

Unsmoothed



Smoothed

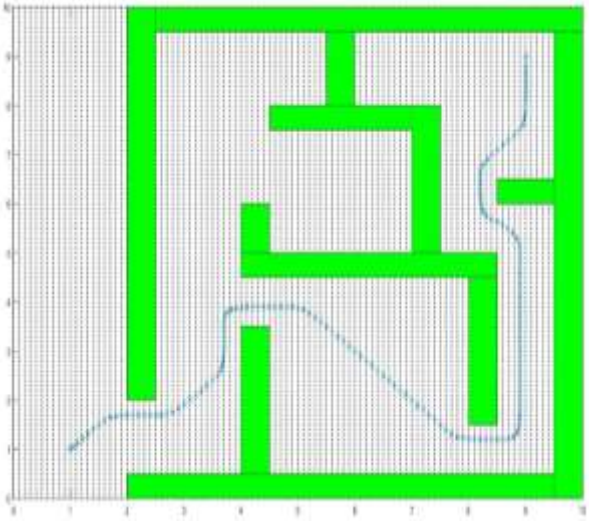


Figure 2

The observer and controller are designed after finding a viable solution to the maze. Analyzing the smoothed path waypoints there are only 176 coordinate pairs to track. In order to get more waypoints along this nonlinear path Bezier curves are used. Bezier curves take control points, 4 for a cubic and 3 for a quadratic, and approximate the coordinates along a path between the first and last control points. These Bezier curves can then be stitched defining the entire path.[3] Using a step size of 0.01 resulting in 4400 waypoints to be used for tracking when designing the observer.

The state space representation for the x position is as follows assuming the mass of the system M =1:

$$\dot{X} = AX + Bu\,, y = CX$$

$$X = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, x_1 = x, x_2 = \dot{x} = v, u = F, M = 1$$

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ \frac{1}{m} = 1 \end{bmatrix}, C = [1 \quad 0]$$

[3] http://devmag.org.za/2011/04/05/bzier-curves-a-tutorial/

Changing the system to also include y position where y_1 = y and $y_2 = \dot{y}$ results in the following:

$$X = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \end{bmatrix}, u = \begin{bmatrix} F_x \\ F_y \end{bmatrix}, A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

In order to design a controller, the true states need to be known. In the present system only the desired states are available. Utilizing the separation principle and tracking control, the controller and observer can be designed simultaneously as the gain matrices K and L do not affect each other in terms of their eigenvalue locations[4]. The tracking control u is given by the following equation where $\hat{x}$, the observer estimate, is used in place of true state values and x_{des} will be the waypoints.[5]

$$u = -K(\hat{x} - x_{des})$$

The desired x and y velocities were found by finding the angle between pairs of coordinates and finding the resulting x and y components of velocity knowing the desired speed is 1.5.

$$Velx = 1.5(\cos\left(tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right)\right), Vely = 1.5(\sin\left(tan^{-1}\left(\frac{y_2 - y_1}{x_2 - x_1}\right)\right)$$

Multiple gain values of K are found using pole placement[6], and LQR[7]. For LQR the Q and R matrices are chosen such that the trajectory error is penalized more heavily than the control cost. For pole placement negative real poles were chosen to arrive at a gain that would drive the system to follow the desired trajectory. The specific values for both LQR and pole placement were selected after a number of test runs were performed.

The observer chosen to estimate the system is a full order observer. This system was chosen due to the ease of implementation even though it has a drawback in accuracy when compared to reduced order observers. The observer was tested by replotting the estimated states to see if the estimated states were to run into any obstacles. The chosen solution was obtained by using LQR control gain matrix K and observer poles such that the observer response will estimate the states faster than the controller. Figure 3 shows an example of a failed solution and the chosen solution. Ares of concern are circled in both the chosen solution and the failed solution. It is clear that the failed solution parameters cause it to run into an obstacle. In the chosen solution and failed solution there is a discrepancy in the first few seconds of

[4] http://ece.gmu.edu/~gbeale/ece_521/xmpl-521-observer-01.pdf
[5] https://mec560sbu.github.io/2016/09/19/Control_synthesis/
[6] https://mec560sbu.github.io/2016/09/19/Control_synthesis/
[7] https://mec560sbu.github.io/2016/09/25/Opt_control/

movement. This is due to using the estimated state locations found by the observer when computing the control u.

Failed Solution

Selected Solution

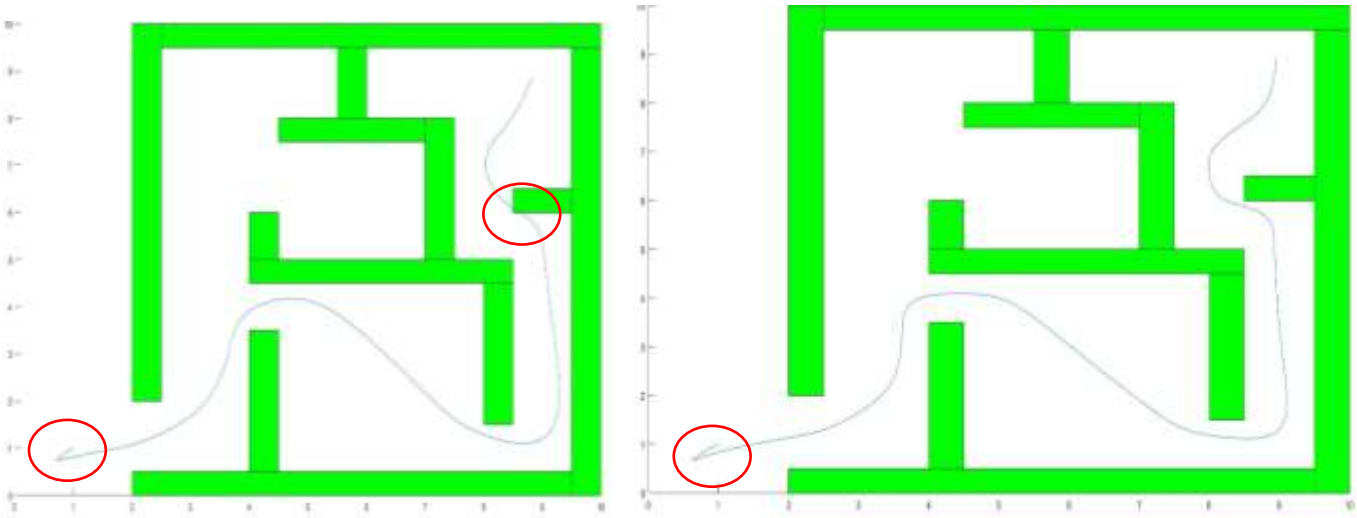


Figure 3

The observer state estimate, the state, and the desired position and velocities can be seen in figure 4. The control is not optimal for this system. There are a number of spikes in control throughout with the maximum and minimum being 7.796 and -14.251 for x and 7.127 and-10.993 for y. After the initial spikes the control for both x and y vary between the values of 5 and -5. This does not meet the problem criteria of control being between -1 and 1. The respective velocities minimum and maximums were -1.399 and 1.8063 for y and -1.502 and 2.082 for x. The maximum x value of 2.082 puts the system values outside the design criteria of having velocities between -2 and 2. To improve the controller design in the future proper initial conditions need to be set to eliminate the initial spikes in control. In order to bring the remaining control values from their range of -5 and 5 to the desired range of -1 and 1, a better selection procedure for selecting the controller poles, the Q and R values, and the observer poles. The final Q,R and observer pole values were selected after a number of tests inputting different values and viewing the results. The values were then adjusted until a solution was found. In future iterations of this project the selection procedure of the observer poles, Q and R is LQR is used, and controller poles if pole placement is used, needs to be reworked. A possible solution is a iterative selection of the desired values enforcing a number of additional constraints such as desired control and speed ranges. The solutions could then be checked to see if they run through any obstacles. The final result of the project resulted in a successful smoothed path for the robot to follow but failed to meet the speed and control criteria.

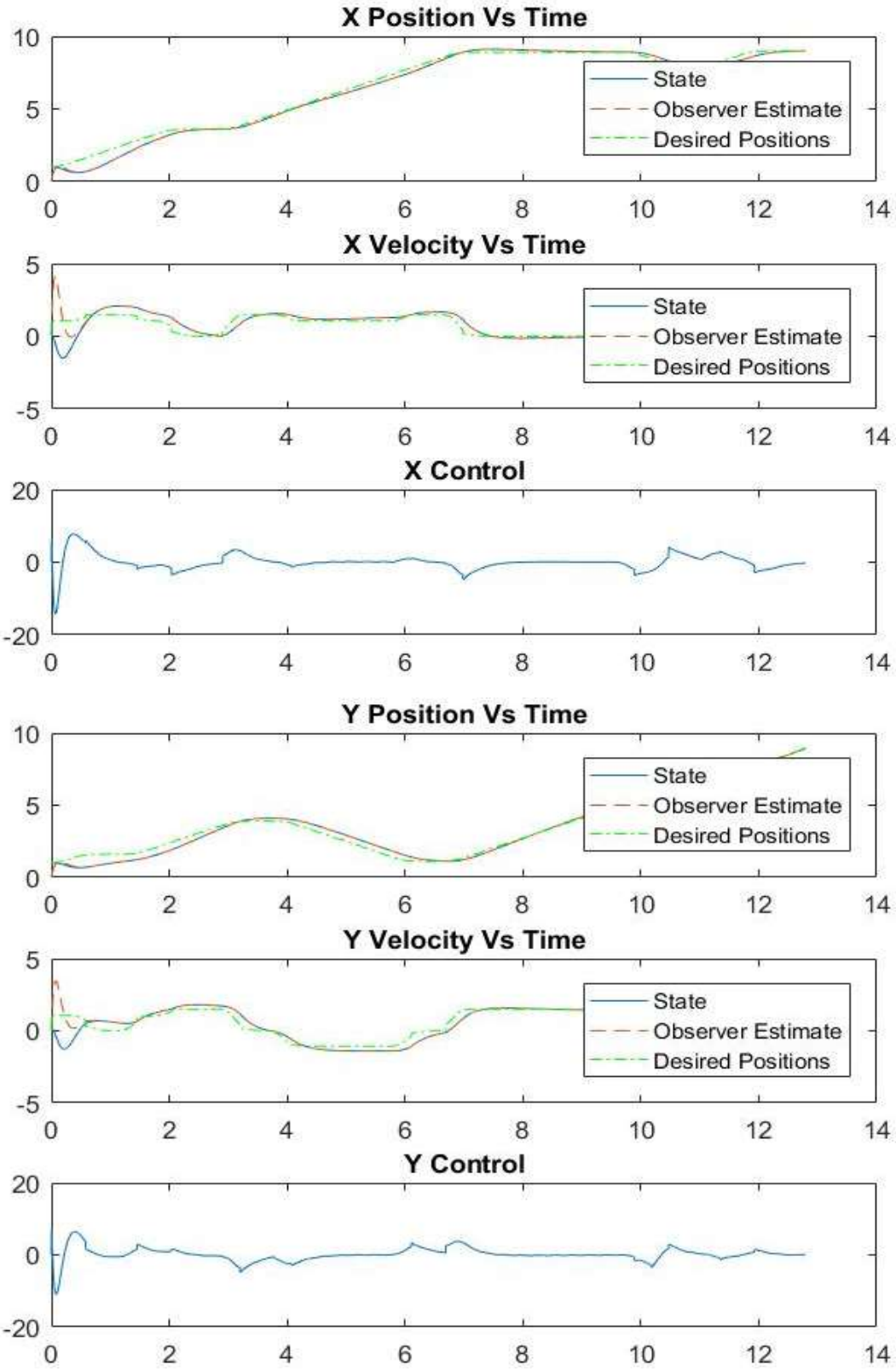

X Position Vs Time
State
Observer Estimate
Desired Positions
X Velocity Vs Time
State
Observer Estimate
Desired Positions
X Control
Y Position Vs Time
State
Observer Estimate
Desired Positions
Y Velocity Vs Time
State
Observer Estimate
Desired Positions
Y Control

Figure 4

References

1. http://www.policyalmanac.org/games/aStarTutorial.htm
2. https://www.youtube.com/watch?v=KNXfSOx4eEE
3. http://devmag.org.za/2011/04/05/bzier-curves-a-tutorial/
4. http://ece.gmu.edu/~gbeale/ece_521/xmpl-521-observer-01.pdf
5. https://mec560sbu.github.io/2016/09/19/Control_synthesis/
6. https://mec560sbu.github.io/2016/09/19/Control_synthesis/
7. https://mec560sbu.github.io/2016/09/25/Opt_control/