Homework 0: Prerequisites Check-in / Warm-up

CSCI 3302: Introduction to Robotics

Report due 1/27/23 @ 11:59pm

Total points: 25

Pro tip: Use Piazza or Office Hours if you are stuck

Submission Instructions:

- Write your code for Q3-4 in the provided Python files (please do not change the python file names and the function signature).
- Create a PDF with detailed, step-by-step solutions to problems 1-2.
- Zip your 3 files (*shortest_path_without_obstacles_Q3.py, shortest_path_with_obstacles_Q4.py*, and *HW0.pdf*) and submit on Canvas. Your zip folder should be titled HW0_lastname.zip. **If you don't follow these naming conventions for the zip folder and .py + .pdf files, the grading script will most likely erroneously give you a zero, so please adhere to these requirements.**
- **You should complete these tasks on your own and without external help. *We will check for plagiarism.***
- ***Please attend office hours if you need support. We are here to help!***

Algebra

We need to see your steps and/or justifications for this section. Simply providing an answer won't earn you full credit.

1) **[5 pts]** Find the *inverse* of the following orthonormal matrix:
[$\frac{2}{3}$, $\frac{1}{3}$, $\frac{2}{3}$]
[-$\frac{2}{3}$, $\frac{2}{3}$, $\frac{1}{3}$]
[$\frac{1}{3}$, $\frac{2}{3}$, -$\frac{2}{3}$]

To find the inverse of the above matrix we must find the Adjuct Matrix and the determinate of the matrix.
We know that A^-1 = (adjA)/(detA)
And to find the AdjA we know that AdjA is equivalent to the transpose of the cofactor matrix
The cofactor matrix is as follows:
[-$\frac{2}{3}$, -$\frac{1}{3}$, -$\frac{2}{3}$]
[$\frac{2}{3}$, -$\frac{2}{3}$, -$\frac{1}{3}$]
[-$\frac{1}{3}$, -$\frac{2}{3}$, $\frac{2}{3}$]
And the transpose of this is:
[-$\frac{2}{3}$, $\frac{2}{3}$, -$\frac{1}{3}$]
[-$\frac{1}{3}$, -$\frac{2}{3}$, -$\frac{2}{3}$]
[-$\frac{2}{3}$, -$\frac{1}{3}$, $\frac{2}{3}$]

Thus we know the AdjA is the above matrix.

Now all we have to do is find the determinant of the matrix. This is done by multiplying each node of the top row with their corresponding cofactor and summing the result.

I found the determinant to be 1.

Thus, to find the inverse we know that A^-1 = (adjA)/(detA), and since the detA is 1. We can conclude that the inverse is equivalent to the adjunct Matix, which brings the final answer to be:

[-⅔, ⅔ , -⅓]
[-⅓ , -⅔, -⅔]
[-⅔ , -⅓ , ⅔]

2) **[5 pts]** Given vector **v** = < 3, 3, -1 >, find vector **w that is orthogonal to v.**

To find vector w which is orthogonal to v, we know that the dot product of
w*v=0
Thus we can build the equation,
3a+3b-c=0 given that w = (a,b,c) and v = (3,3,-1)
From there we can set a=1 and b=1 and solve for c.
We get 3*1+3*1-c = 0 which gives c=6.
Thus, we have found that a vector orthogonal to vector v is **w = < 1, 1, 6 >**

Shortest Path in a grid (coding)

- **[5 pts] (Coding)** NASA's Perseverance is on Mars and has mapped out an **obstacle free** square sector with grids. The free space is represented with 0s. Help It save precious energy by figuring out the shortest path from a given start position on this grid to a goal position. It cannot travel diagonally because of some issues. Assume that the grid is square and **the number of cells on the path is the length of the path including the start and the goal cells**. Complete the provided python function. Do not forget to add comments to your code. Here is an example -

  grid = [[0,0,0],
  [0,0,0],
  [0,0,0]
  start, goal = (0,0), (2,2)
  Solution = 5

  Note: Do not return the shortest path but just a single integer representing the length of the shortest path.

- **[10 pts] (Coding)** Perseverance is able to perform some science thanks to you. It now spots a sector that has **obstacles represented by 1s.** Keeping the assumptions same as Q3, return the length of the shortest path avoiding obstacles. **Return -1 if no such path exists**.

  Hints:

- Don't explore neighboring cells that don't exist (are beyond the grid's boundary)
- Don't explore neighboring cells that are obstacles.
- Don't revisit a cell.
- Your search ends when there is nothing to explore.

- You will have to propagate the information of the explored length so far.
- CSCI 2270

Here are some examples.

grid = [[0,0,0],
[1,1,0],
[1,1,0]]
start, goal = (1,2), (0,0)
Solution = 4

grid = [[0,1],
[1,0]]
start, goal = (0, 0), (1,1)
Solution = -1 ---> Perseverance can't travel diagonally and thus there doesn't exist any feasible path.