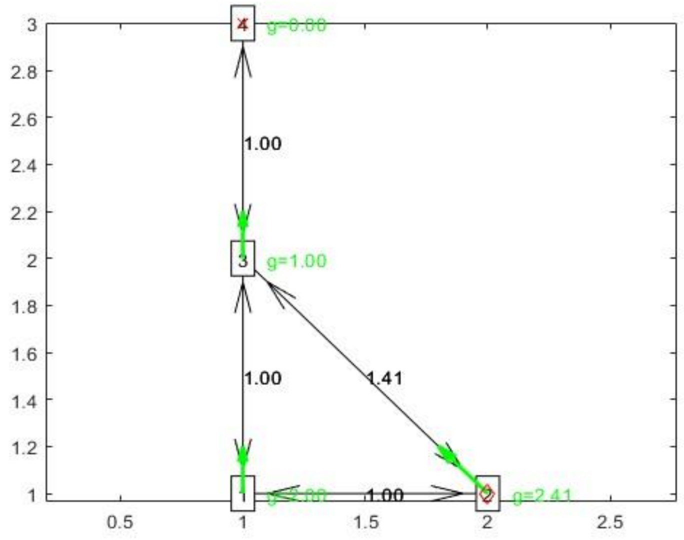
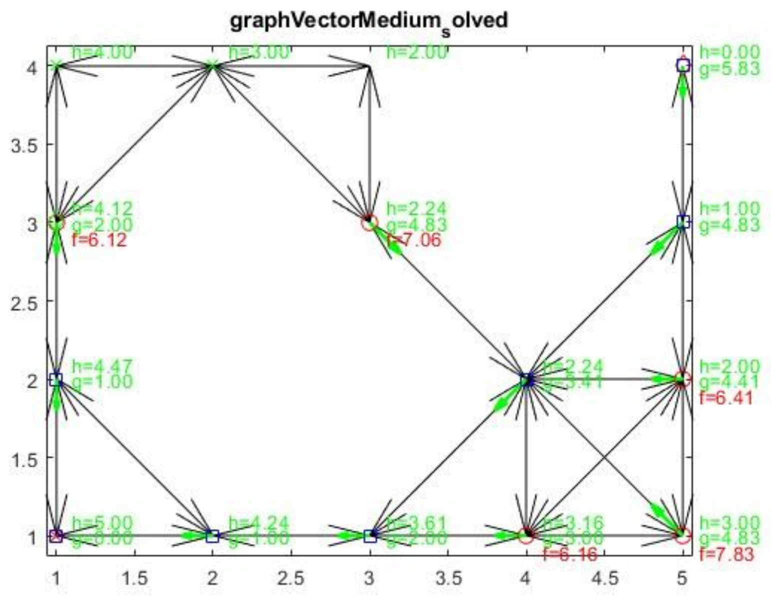
HomeWork 4
Report
By(U25712111)

(I have **2** Homework Credits for this semester and I am using **1** Homework Credit on this report. After this I will be only left with **1** more Homework credit)

Question report 1.1: Run the provided function graph_testData_plot ( ), and, in your report:
1) include the two figures with the two solved examples, and
2) describe the meaning of the arrows, and the numbers labeled with g , h and f .

Solution :





Arrows denote the direction and magnitude of the vector field at various points.
G denotes the scalar field at the starting point of the vector.
F denotes the change in the scalar field value along the direction of the arrow.

Question report 1.2: Make a function graph_search_test ( ) that test your graph search function on one of the provided graphs.
Solution :

Question Report 2.1 : Pick three values of NCells such that, after discretization:
1) Some or all of the obstacles fuse together (NCells is too low);
2) The topology of the Sphere World is well captured (NCells is "just right");
3) The graph is much finer than necessary (NCells is too high).

Include the three values in your report, together with a visualization of the corresponding graphs (using graph_plot ( )).

Solution :

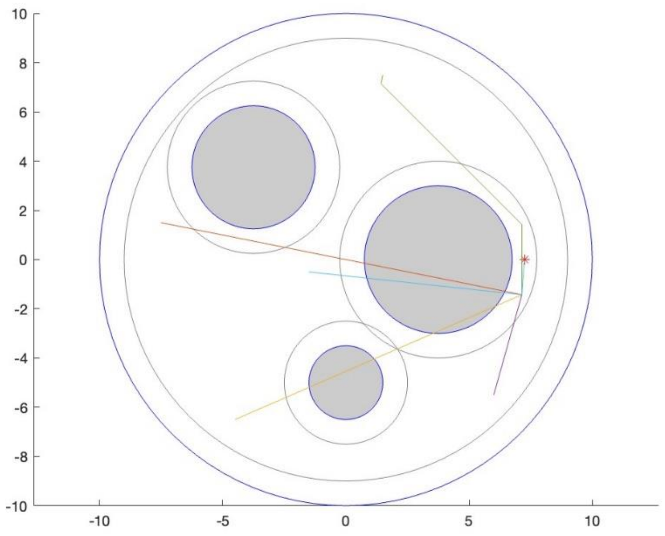
Selecting Values for NCells:

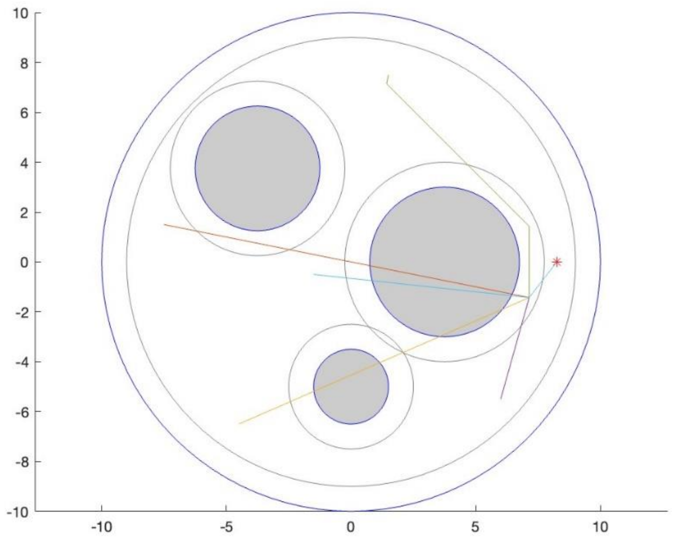
For Some or all of the obstacles fuse together (NCells is too low) = Low NCells = 5 or 10.
For The topology of the Sphere World is well captured (NCells is "just right") = Medium NCells = 20 to 50.
For The graph is much finer than necessary (NCells is too high) = High NCells = 100 or more.

Question report 2.2: Create the following function: sphereworld_search (NCells)
Input arguments
• NCells : Size of the discretization grid to use (as number of cells on one side).
Description:

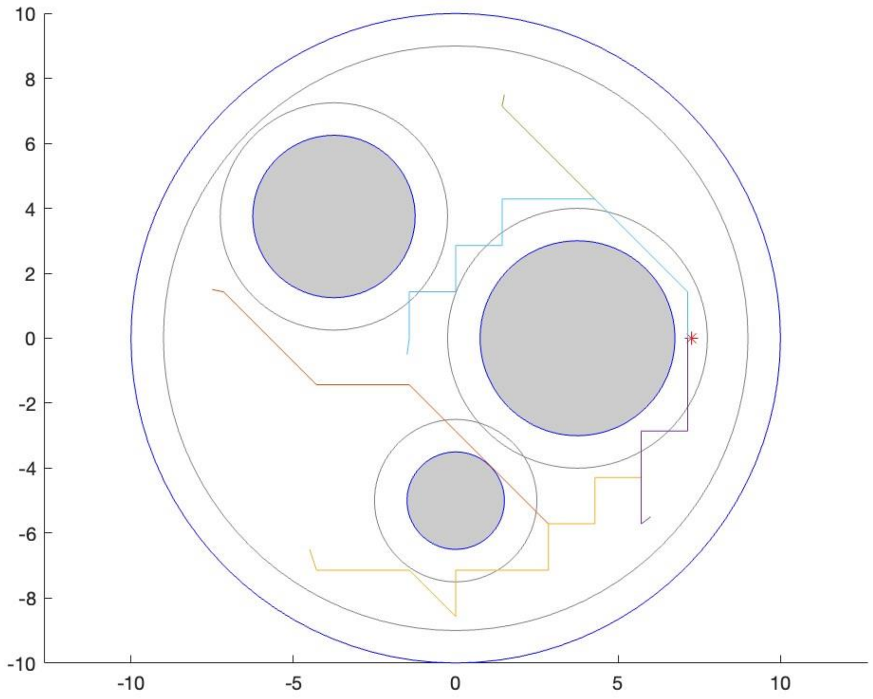
- Load the variables xStart , xGoal from sphereworld.mat .
- For each of the three values for NCells :
    - Run the function sphereworld_freeSpace_graph ( ) for the given value of NCell .
    - For each goal in xGoal :

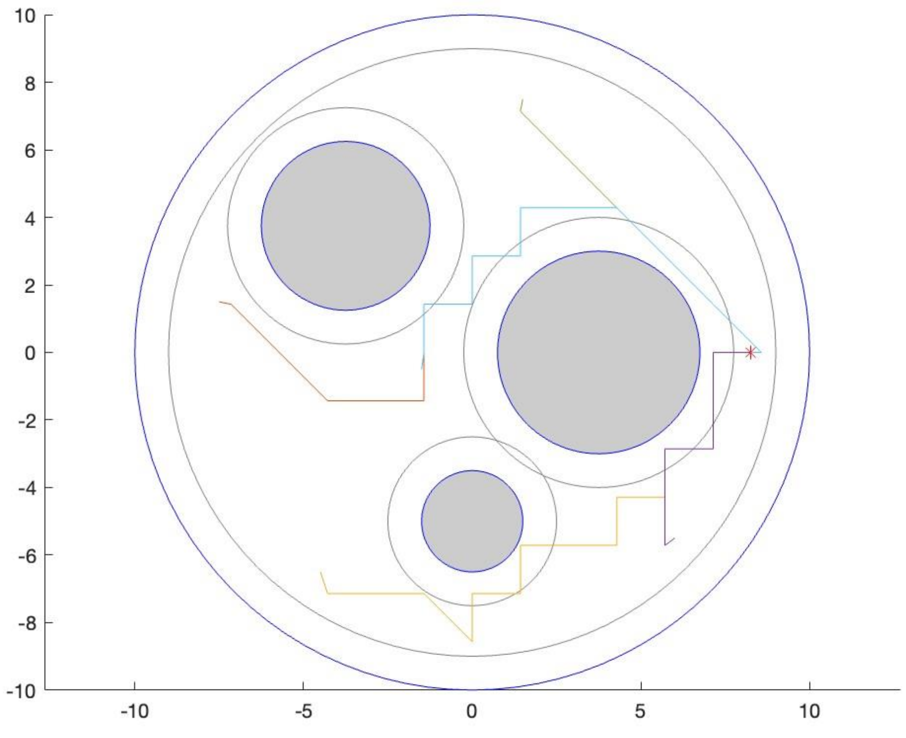
i. Run graph_search_startGoal ( ) from every starting location in xStart to that goal.
ii. Plot the world using sphereworld_plot ( ), together with the resulting trajectories.
In total, you should produce six different images (three choices for NCell times two goals, five trajectories in each plot). Include all the images in the report. Please make sure that images from different choices of NCell but the same goal appear together in the same page (to help comparisons).
Solution :



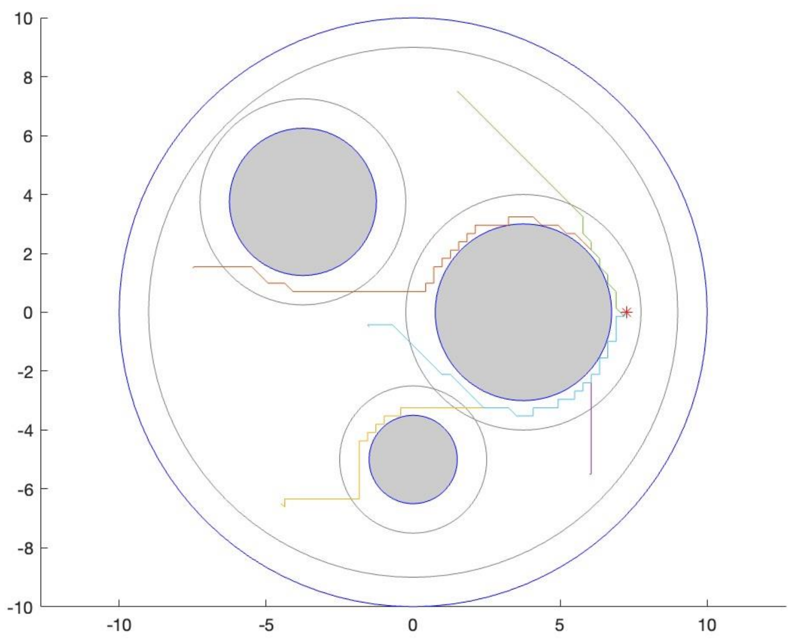


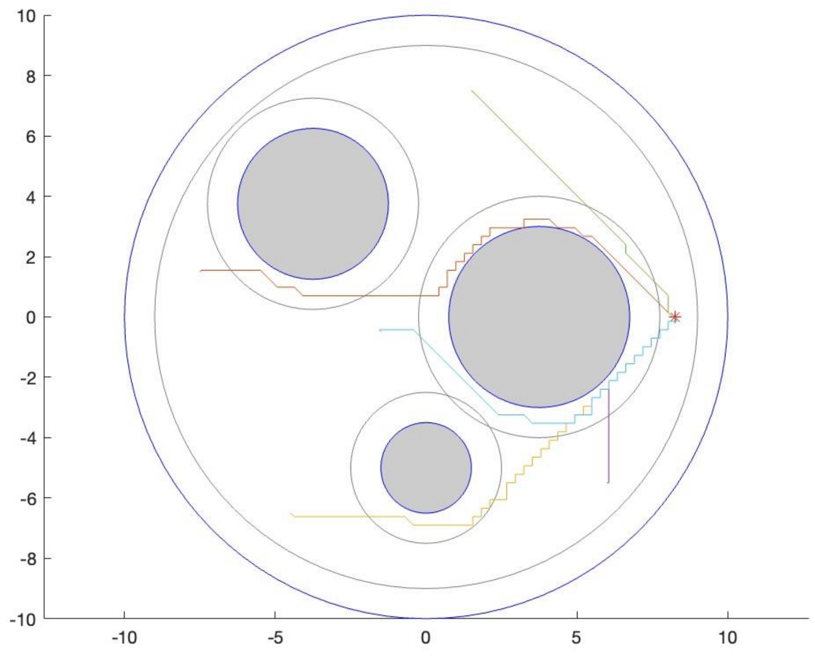
NCell = 8





NCell = 15





NCell = 72

Question report 2.3: Comment on the behavior of the A∗ planner with respect to the choice of NCell .

Solution :

Ncell=8: With this setting, the spatial discretization is pretty coarse. Each grid cell is large, which can lead to a few issues:

- The planner might not accurately interpret the space around obstacles, leading to paths that aren't optimal or might even be unfeasible in a more detailed space.
- While the A* algorithm can find a route quickly due to fewer nodes to check, the path might not be the best in a continuous space.
- If the robot's size isn't well-represented by these big grid cells, we might not calculate obstacle clearance well, increasing the risk of collisions.

Ncell=15: This is a finer discretization than Ncell=8, and it helps the A* planner better represent the continuous space and its obstacles.

- The paths found are likely to be closer to the optimal route and avoid obstacles more effectively.
- With a higher resolution, the robot's size and obstacles are better represented, making the paths safer for navigation.

Ncell=72: Here, the discretization is very fine.

- The paths generated are likely to be very accurate in avoiding obstacles and close to the optimal path in continuous space.

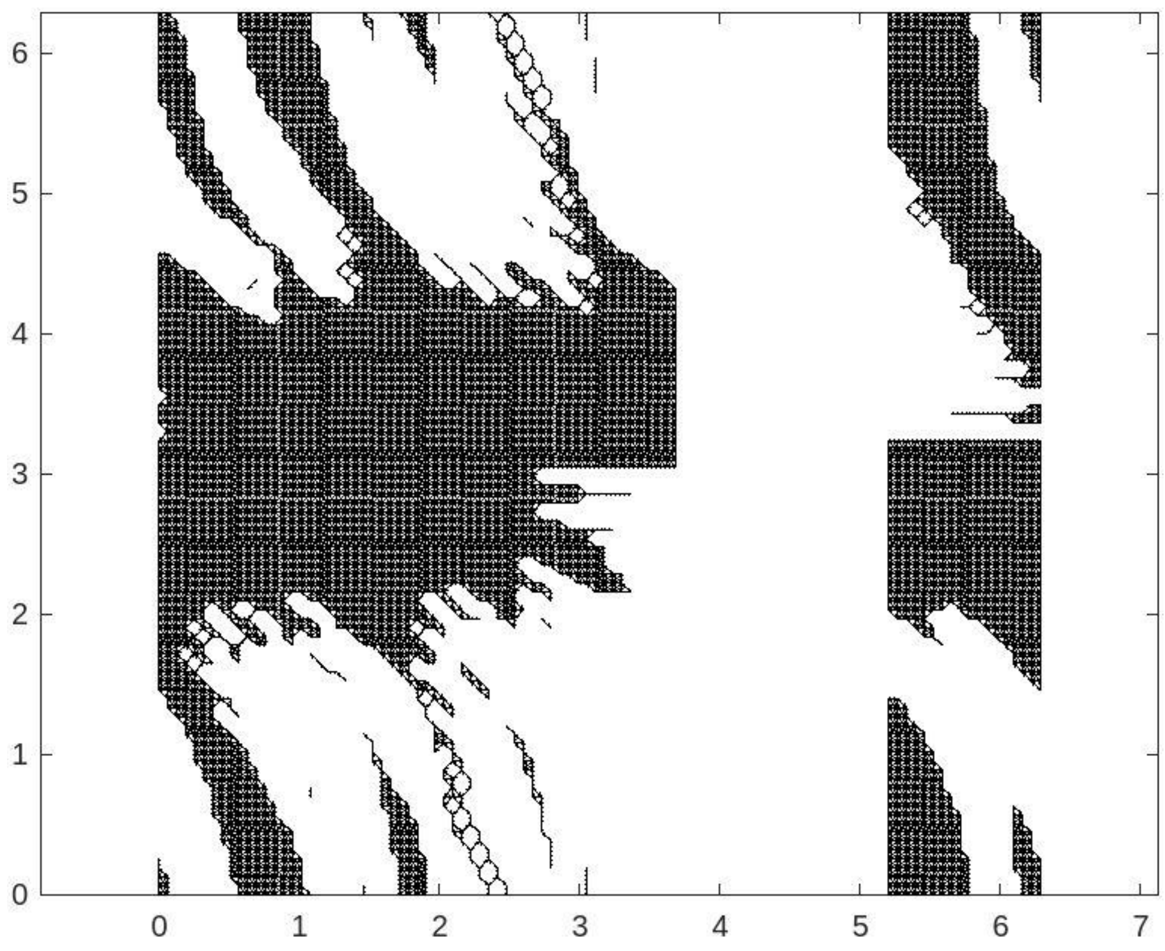
Question report 2.4: Comment on the behavior of the A∗ planner with respect to the potential planner from Homework 3.

Solution :

Question report 3.1: For this question, you need to implement the following functions:
twolink_freeSpaceGraph ( )
Description: The function performs the following steps

- Loads the contents of twolink_freeSpace_data.mat .
- Calls grid2graph ( ).
- Stores the resulting vectorGraph struct array in the file twolink_freeSpace_graph.mat .

Use the function graph_plot ( ) to visualize the contents of the file twolink_freeSpace_graph.mat .
Include the figure in your report.
Solution :

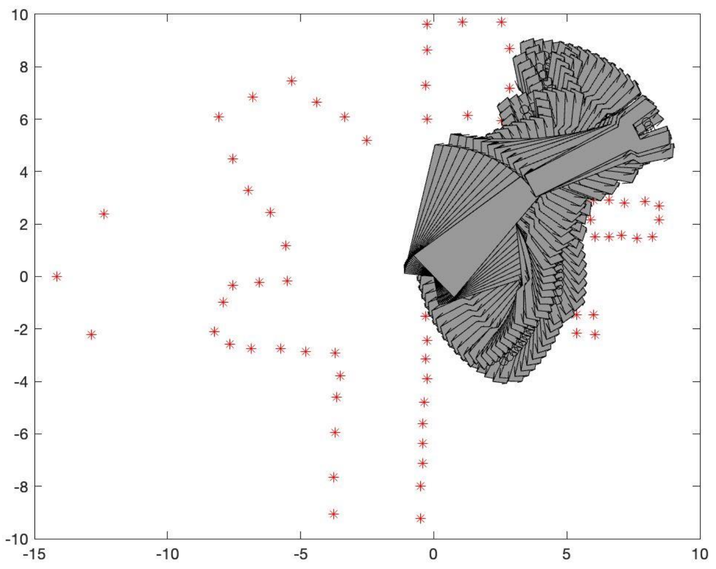


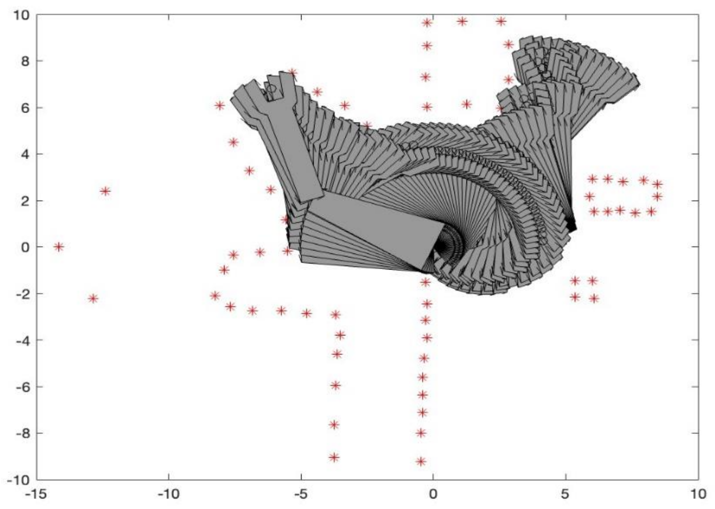
Question report 3.2: The following function visualizes a sample path, both in the graph,
and in the workspace. twolink_testPath ( )
Description: Visualize, both in the graph, and in the workspace, a sample path where the second link rotates and then the first link rotates (both with constant speeds). You should obtain figures similar to those shown in Figure 2. In your report, explain why in the area delimited the dashed rectangle and marked with the symbol there are no nodes.
Solution :

Question report 3.3: Plot the points obstaclePoints in twolink_testData.mat , call the function graph_search_startGoal ( ) (or twolink_search_startGoal ( ), if you completed the previous optional question), and then twolink_animatePath ( ), for the following start/goal configurations:

thetaStart=[0.76;0.12] thetaGoal=[0.76;6.00]
thetaStart=[0.76;0.12] thetaGoal=[2.72;5.45]

Solution :





Question report 3.4: For the Easy case in the question above, comment on the unwinding phenomenon that appears if you do not use the torus option (that is, why the planner does not find the straightforward path that keeps the first link fixed). To obtain full marks, make sure to include the relation between your answer and the visualization of the configuration space from Homework 2.

Solution :
In the context of the robotic arms motion planning, not using the torus option can lead to some inefficiencies. This option is important because it recognizes that certain angles are essentially the same point, like how 0 degrees and 360 degrees are identical in terms of the arms position. Without this understanding, the motion planner might end up rotating a link more than necessary.

For instance, imagine a situation where the planner needs to move a link to what is practically the same position but sees it as 0 degrees and then 360 degrees. Without the torus option, it does not realize these points are the same and might rotate the link all the way around unnecessarily.

Now, when we don't use the torus option, the motion planner doesn't fully grasp the continuity of the robotic arm's rotation. It misses the fact that after a full 360-degree turn, the link is back where it started. As a result, it might come up with complex routes, like spinning the first link several times to reach a position that could have been achieved without any rotation.

On the other hand, when the torus option is active, it joins points like 0 degrees and 360 degrees together, accurately reflecting the arm's movement. This way, the planner can recognize the most efficient route might be to keep the first link stationary and just move the second one. This approach aligns better with how the joints of the robotic arm actually function, leading to simpler and more efficient motion paths.