Ian Chadwick
11/26/2021

Homework #4 Report

Question 2.1: Graph Representations of SphereWorld for three values of NCells: 6, 50, 100

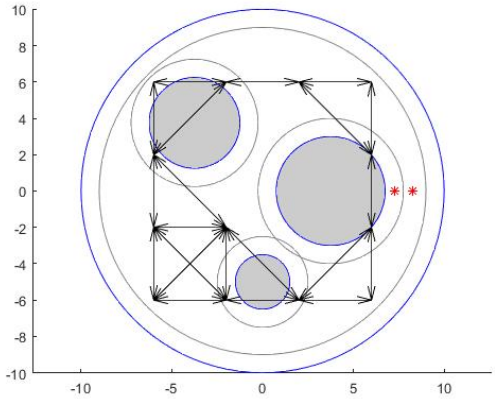

Figure 1: The number of cells, NCell value = 6, in this figure is much too low and the obstacles have fused together making the graph too sparse to be accurate. Also, several edges between nodes are incorrect because they cross through the obstacles.

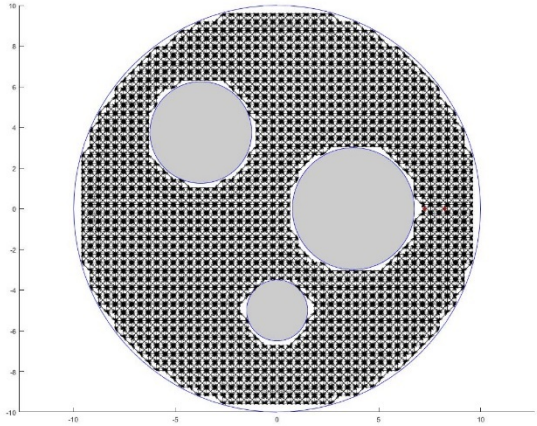

Figure 2: The number of cells NCell value = 50, is an accurate approximation of the topology of the sphere world is well captured without having so many nodes that the performance of the search algorithm will be significantly/adversely impacted.

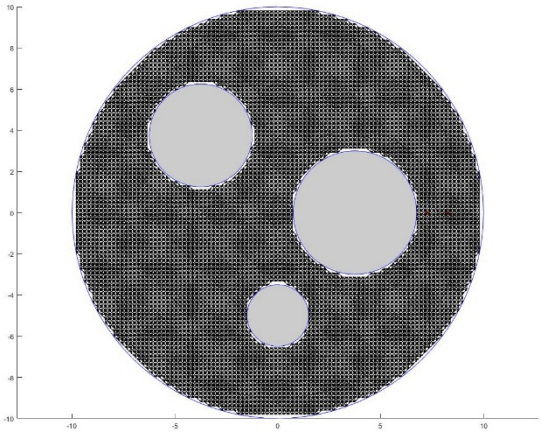

Figure 3: The number of cells NCell value = 100, in this figure shows a case where the graph is much finer than is necessary, and as a result the search algorithm will be slower but the accuracy of the path isn't significantly improved.

Question 2.2: Path trajectories for each start location in SphereWorld for values of NCells: 6, 50, 100

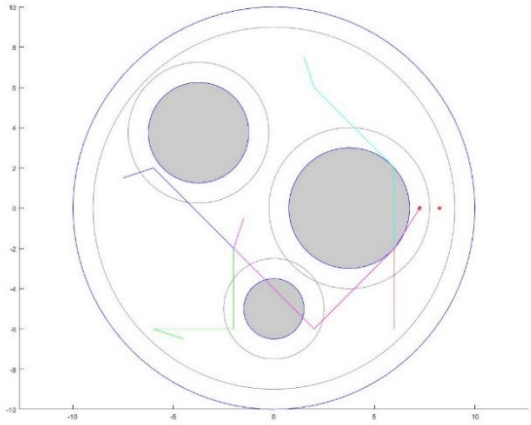

Figure 4: Goal 1 for NCells = 6. The paths are not correct since the graph's node distribution is too coarse and as a result there are edges that pass through the obstacles.

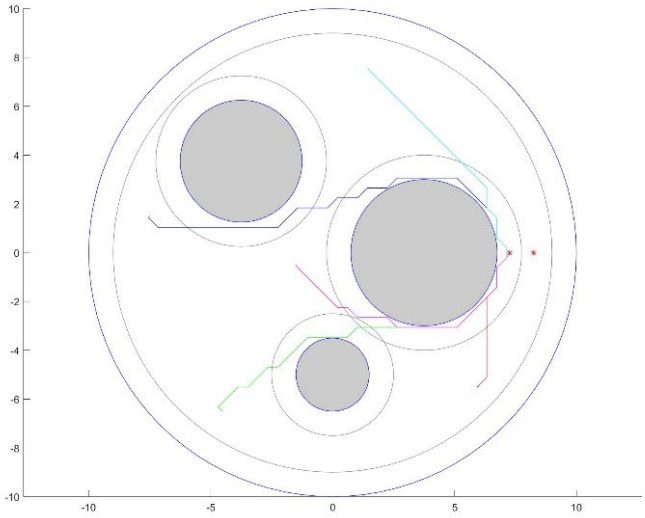

Figure 5: Goal 1 for NCells = 50. The paths chosen appear to be a good approximation of the shortest path and are what's expected. The runtime of the algorithm was relatively quick.

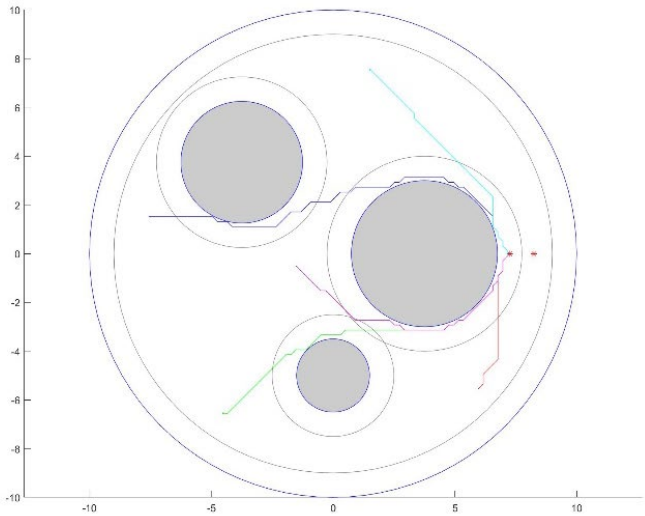

Figure 6: Goal 1 for NCells = 100. The paths found for this implementation are roughly the same as the paths for NCells = 50, but the runtime was longer. The only real difference is that the trajectories are generally closer to straight lines than for NCells = 50.

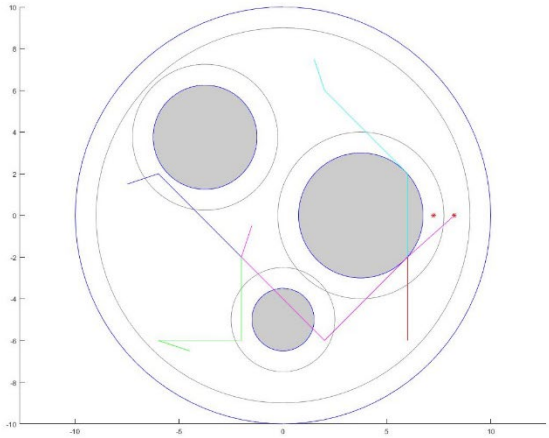

Figure 7: Goal 2 for NCells = 6. Similar performance and issues for these paths as paths for as for Goal 1, in Figure 4.

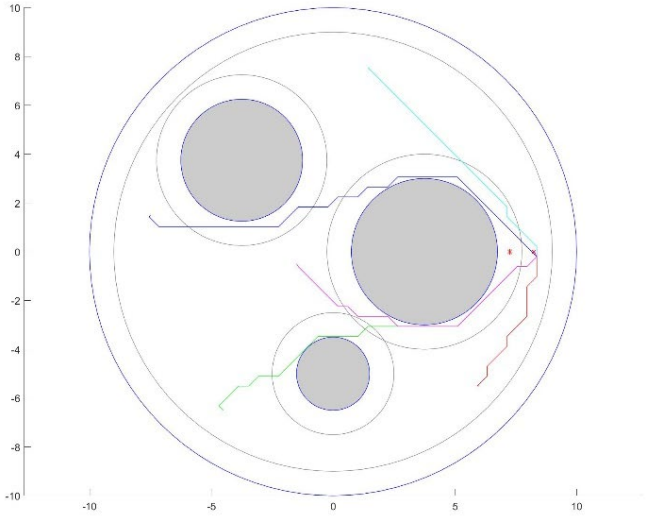

Figure 8: Goal 2 for NCells = 50. Similar performance for these paths as paths in Figure 5.

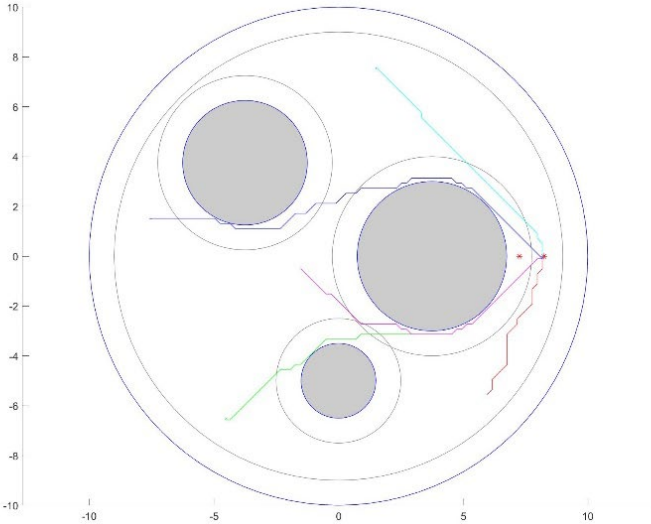

Figure 9: Goal 2 for NCells = 100. Similar performance as the paths generated in Figure 6.

Question 2.3: Performance of A* with respect to values of NCells

Generally, the algorithm performs faster with lower values of NCells, but the accuracy of the path (i.e. if it's actually the shortest) suffers until a certain threshold number for NCells, after which the effect of increasing NCells has a smaller effect on refining the path, but a larger effect on runtime.
This is to be expected, since the higher the value of NCells, the more total nodes that are in the graph and therefore, the more nodes that must be explored before a shortest path is found. However, at low values of NCells, the algorithm performs fast, but the topology of the space was not well captured. As a result, the paths were very coarse and sometimes even incorrect for *very* low values of NCells (see figures 1, 4 and, 7).
The accuracy of the generated path increased until a certain threshold was reached and increasing NCells beyond that did not significantly improve the path, but instead significantly increased the runtime of the algorithm. Through experimentation NCells = 27 appears to be the first correct (albeit rough) version of the shortest path for this SphereWorld configuration space. This was not chosen as the "ideal" NCells representation in the figures above because for this value, there were still some gaps in around the obstacles and toward the outer boundary of the configuration space. Increasing NCells beyond this value generated paths that were technically more accurate, but the paths moved closer the obstacles and were roughly along the same general trajectory. Increasing the number of cells from 27 to 50 did not show any noticeable performance loss but increasing from 50 to 100 was the point where the algorithm started to perform noticeably slower.

Question 2.3: Performance of A* planner with respect to potential based planners

Both the A* algorithm and the potential based planners generated very similar paths. One critical difference was the fact that A* did not require as much fine tuning as the potential planner, which required a fair amount of time spent to determine the optimal values for the input parameters for step size (epsilon), the number of steps and the repulsive weights to ensure that it found the path. Additionally, while the CLF-CBF formulation was the most accurate and required the fewest number of steps to reach the goal, it required significant computation time. Whereas overall, the A* algorithm had comparable results in a fraction of the time that it took for the accurate versions of the potential planners to generate a path, especially when time spent experimenting with different values and repulsive potentials was factored in.
One advantage to the potential planners over this version of the A* planner is that the CLF-CBF formulation was able to keep a certain distance away from the obstacles, which may be an important feature of a planner that had to account for robots that weren't able to be represented as a single point and had a larger mass to account for. There may be a version of the A* algorithm with modified "valid node" creation criteria that could account for maintaining certain distances away from obstacles (i.e., a buffer zone). The implementation of this modified A* algorithm would likely be much faster to compute than the CLF-CBF formulation but also achieve similar results in regard to maintaining a buffer zone between the robot and the obstacles.

Question 3.1: Graph representation of two-link manipulator configuration space

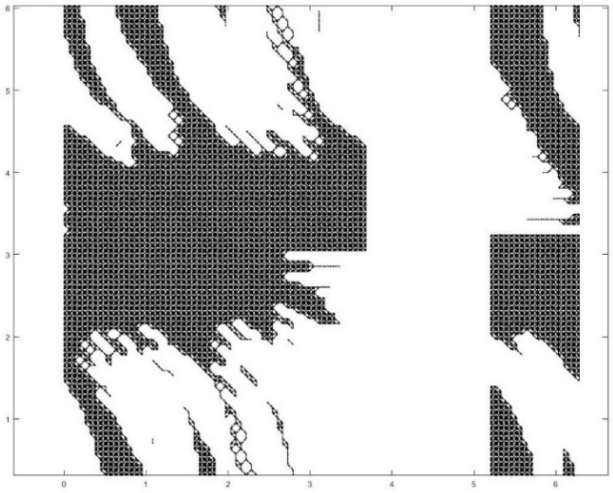

Figure 10: Graph representation of the configuration space for the two-link manipulator with axes $[\begin{matrix}\theta_1\\ \theta_2\end{matrix}]$

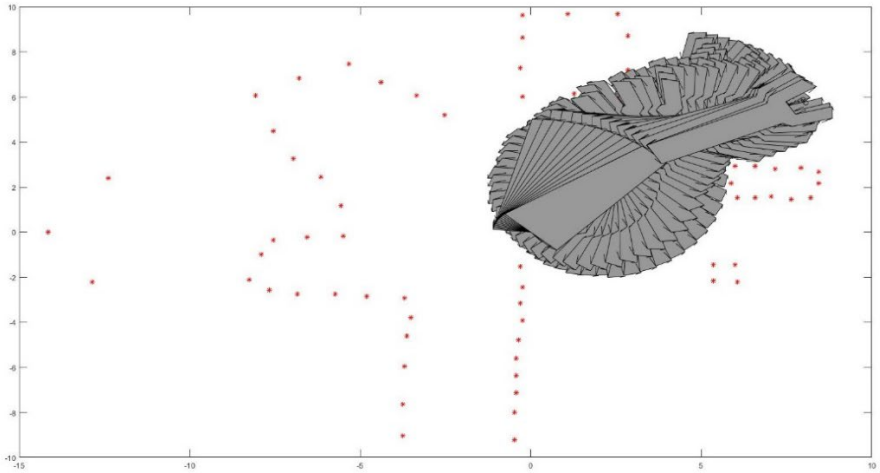

Figure 11: Path for the "Easy" difficulty with thetaStart =$[\begin{matrix}0.76\\ 0.12\end{matrix}]$ *and thetaGoal = =*$[\begin{matrix}0.76\\ 6.00\end{matrix}]$

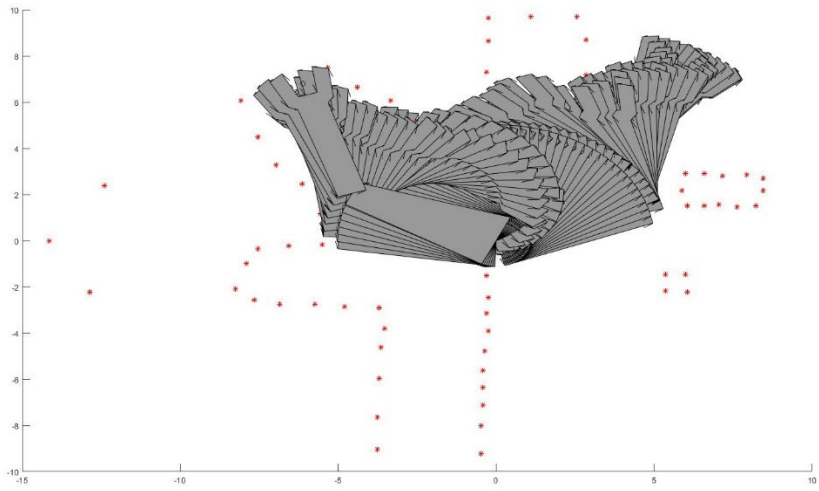

Figure 12: Path for the "Easy" difficulty with thetaStart =$[\begin{matrix}0.76\\ 0.12\end{matrix}]$ *and thetaGoal = =*$[\begin{matrix}2.72\\ 5.45\end{matrix}]$

Question 3.2: Explanation of "Easy" case behavior

The easy case had odd behavior where instead of moving directly toward the goal, it "unwound" because without the torus representation, where the edges of the plane are "glued together" to form a torus (Figure 13 from homework 2, below) it is not an accurate and full topological representation of the configuration space. In the current "non-torus" planar form, some of the valid paths are not included in this representation of the grid that is input into the grid2graph function. The missing neighbor connections are between nodes on the edges of the grid, connecting the nodes on the left and right and top and bottom edges respectively. However, since in its current form, the way that the graph is constructed, by graph2grid, these neighbors are not added to their respective neighbor lists for the nodes, the planner is forced to find sub-optimal paths, since it does not have the option to consider the connected paths that a torus representation would allow.

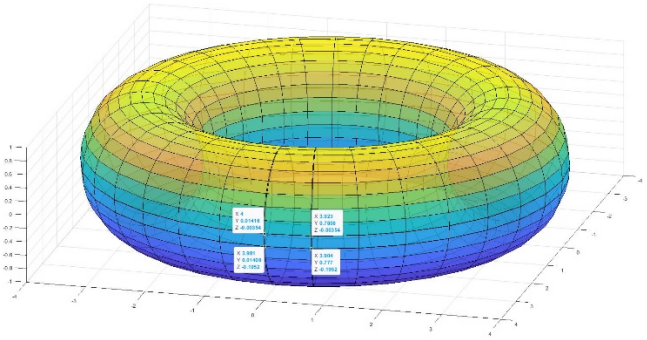

Figure 13: Torus representation of the configuration space for two-link manipulator

Question 3.3: Obstacle avoidance

The issue is occurring because in the graph representation, the collision is only checked at each of the location represented at each of the nodes and the space in between is not considered. This could allow the manipulator to move through space that would normally cause a collision because the nodes on either end of the move are in free space. One solution is to incorporate aspects of the potential function to create a repulsive effect from the obstacles. If the collision detection is modified to instead consider a buffer zone or allowed closest distance to obstacles, it can prevent the manipulator from getting too close to the obstacle.

Another potential solution would be to modify the selection of "valid nodes", to take into consideration the distance from the nearest obstacle and exclude moves to nodes that allowed the manipulator to approach too close to obstacles. Essentially, this would reduce the total free space that was available to the manipulator and may have an effect on the accuracy of the "shortest path" but it would ensure that the manipulator didn't come into contact with obstacles.