Homework 2 – Motion Planning – Hari Sudan Sundar

Points to Note:

1. This executable was built with Visual Studio 2013 and Multi-Threaded (/MT).
2. The version of Mujoco.exe (server) is probably the old version, since this version was downloaded on 05/18. The server used is also attached in the zip for reference.
3. The program execution pauses before starting to animate the shortest path for each of the three sources. It waits for a user input “ENTER KEY” to start each animation. And sleep is also introduced for each interpolated step to help you closely track the trajectory.
   1. For example: Before starting animation for each source, it would prompt with the below message and wait for you to press enter key
      1. “Press enter key to do shortest path animation for source: 0”
      2. “Press enter key to do shortest path animation for source: 1”
      3. “Press enter key to do shortest path animation for source: 2”
4. The default values for N, K in the program are N = 1000, K = 30. However, you can try other combinations by giving command line parameters, like the following.
   1. For example, to start it with N = 2000, K = 50, Results file = results.txt , you would execute
      1. **MujocoClient.exe 2000 50 results.txt**
   2. If you execute just “**MujocoClient.exe**”, then default values of N = 1000, K = 30 are used and the results (shortest path distances) are not saved on disk anywhere, nor shown on command line.
   3. **So if you prefer to see the shortest path distance results and with faster completion time, I would encourage you to execute with the following parameters:**
      1. **MujocoClient.exe 1000 30 results.txt** – This would save the shortest path distance results in results.txt for each of the three sources.
5. Dijkstra’s algorithm was used to find shortest paths.

Best Shortest Path distance achieved during experimentation with various values of N, K is the below

For N = 10000, K = 200:

Shortest Path Distance for Qinit0 = 2.04932

Shortest Path Distance for Qinit1 = 2.22834

Shortest Path Distance for Qinit2 = 2.39185

**The next page gives the complete table of shortest path distances for different values of N, K for each source.**

Analysis of shortest path distances based on the values observed in the next page:

As is expected, as we increase N and K, since the number of sampled points and number of possible neighbor transitions increase, graph becomes more dense and hence, the shortest path distances decrease, since we find more alternative routes with lesser cost to reach the goal. But as seen from the table next page, if we keep the number of samples N as constant and keep increasing only the K, though the shortest path distances reduces for the first few K increments, it tops out and doesn’t reduce further even if we keep increasing the value of K for the same N. For example, for N = 1000, shortest path distance does reduce when we increase K from 10 to 80 but however for K = 80, 100, 150, the program gives the same shortest path distances and does not get better. However, if the number of samples N is increased and as well as K is increased at-least logarithmically compared to N, we do notice good decrease in shortest path distances. For example with increase of N to 5000 and K = 80 and N to 10000 and K = 200, we do notice significant reduction in the shortest path distances. This can be explained with the idea that even though we are ready to accept more neighbor transitions, there aren’t enough valid samples to take advantage of the increase in K, for constant N.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Shortest Path for Source QINIT 0** | | | | **Shortest Path for Source QINIT 1** | | | | **Shortest Path for Source QINIT 2** | | |
|  |  |  |  |  |  |  |  |  |  |  |
| **N** | **K** | **ShortestPath** |  | **N** | **K** | **ShortestPath** |  | **N** | **K** | **ShortestPath** |
|  |  |  |  |  |  |  |  |  |  |  |
| 500 | 10 | 3.57594 |  | 500 | 10 | 3.33553 |  | 500 | 10 | 4.91619 |
| 500 | 20 | 3.22691 |  | 500 | 20 | 2.89125 |  | 500 | 20 | 3.41754 |
| 500 | 30 | 3.17514 |  | 500 | 30 | 2.56383 |  | 500 | 30 | 3.1986 |
| 500 | 50 | 2.78686 |  | 500 | 50 | 2.56383 |  | 500 | 50 | 3.1986 |
| 500 | 65 | 2.60875 |  | 500 | 65 | 2.56383 |  | 500 | 65 | 3.1986 |
| 500 | 80 | 2.60875 |  | 500 | 80 | 2.47142 |  | 500 | 80 | 3.1986 |
| 500 | 100 | 2.60875 |  | 500 | 100 | 2.32563 |  | 500 | 100 | 3.1986 |
| 500 | 150 | 2.58725 |  | 500 | 150 | 2.32563 |  | 500 | 150 | 3.1986 |
|  |  |  |  |  |  |  |  |  |  |  |
| 1000 | 10 | 3.48935 |  | 1000 | 10 | 3.13794 |  | 1000 | 10 | 4.54298 |
| 1000 | 20 | 3.14074 |  | 1000 | 20 | 2.80039 |  | 1000 | 20 | 4.07567 |
| 1000 | 30 | 2.65053 |  | 1000 | 30 | 2.73594 |  | 1000 | 30 | 3.89869 |
| 1000 | 50 | 2.65053 |  | 1000 | 50 | 2.57774 |  | 1000 | 50 | 3.1986 |
| 1000 | 65 | 2.65053 |  | 1000 | 65 | 2.57774 |  | 1000 | 65 | 3.1986 |
| 1000 | 80 | 2.34539 |  | 1000 | 80 | 2.44528 |  | 1000 | 80 | 3.1986 |
| 1000 | 100 | 2.34539 |  | 1000 | 100 | 2.44528 |  | 1000 | 100 | 3.1986 |
| 1000 | 150 | 2.34539 |  | 1000 | 150 | 2.44528 |  | 1000 | 150 | 3.1986 |
|  |  |  |  |  |  |  |  |  |  |  |
| 2000 | 20 | 2.2379 |  | 2000 | 20 | 2.54061 |  | 2000 | 20 | 3.65201 |
| 2000 | 30 | 2.2379 |  | 2000 | 30 | 2.54061 |  | 2000 | 30 | 3.65201 |
| 2000 | 50 | 2.04932 |  | 2000 | 50 | 2.54061 |  | 2000 | 50 | 3.49323 |
| 2000 | 65 | 2.04932 |  | 2000 | 65 | 2.54061 |  | 2000 | 65 | 3.49323 |
| 2000 | 80 | 2.04932 |  | 2000 | 80 | 2.5066 |  | 2000 | 80 | 3.37786 |
| 2000 | 100 | 2.04932 |  | 2000 | 100 | 2.47877 |  | 2000 | 100 | 3.1986 |
| 2000 | 150 | 2.04932 |  | 2000 | 150 | 2.29094 |  | 2000 | 150 | 3.1986 |
|  |  |  |  |  |  |  |  |  |  |  |
| 3000 | 60 | 2.04932 |  | 3000 | 60 | 2.51677 |  | 3000 | 60 | 3.2218 |
| 5000 | 80 | 2.16726 |  | 5000 | 80 | 2.42809 |  | 5000 | 80 | 2.75961 |
| 10000 | 200 | 2.04932 |  | 10000 | 200 | 2.22834 |  | 10000 | 200 | 2.39185 |