# Home work of motion planning L2

Name: bruceSz

## matlab version

Main content :

1> matlab code

Main modifications are in A_star_search.m file, also one visualize_path.m file is created to separate visualizing map and path. See the code.

2> Some snapshots of execution of A star.

This are three execution results with randomly initialized map and obstacles.

Blue hollow cycles indicates the path from start to end. Other hollow circles (blue) is the visited nodes(nodes that were added to open list).

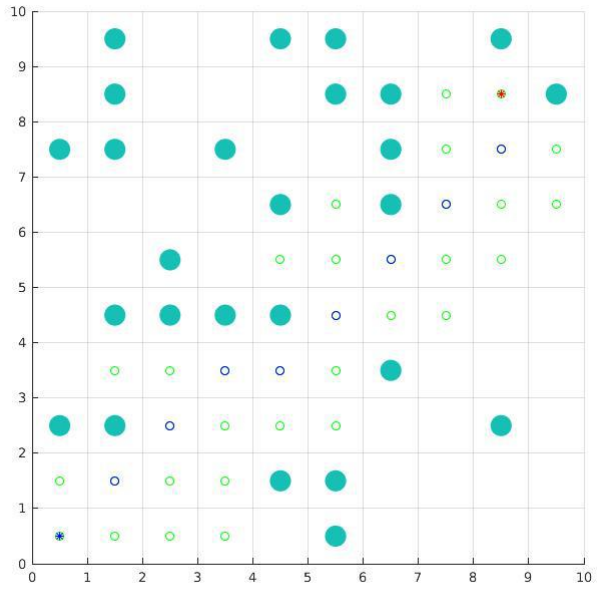


Figure1:

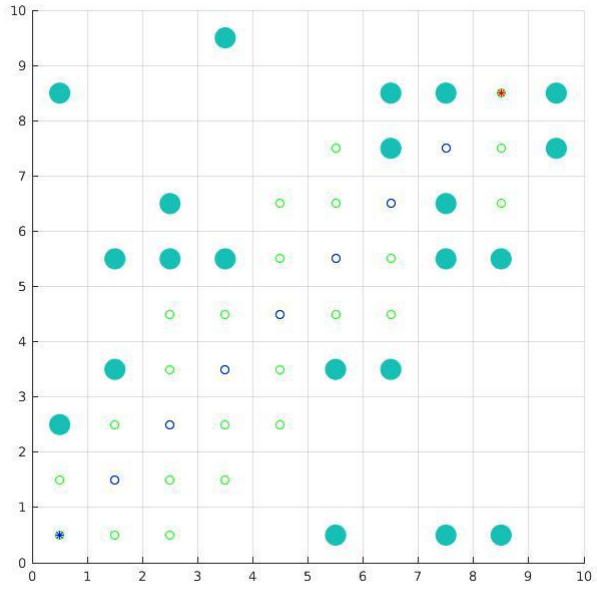


Figure 2

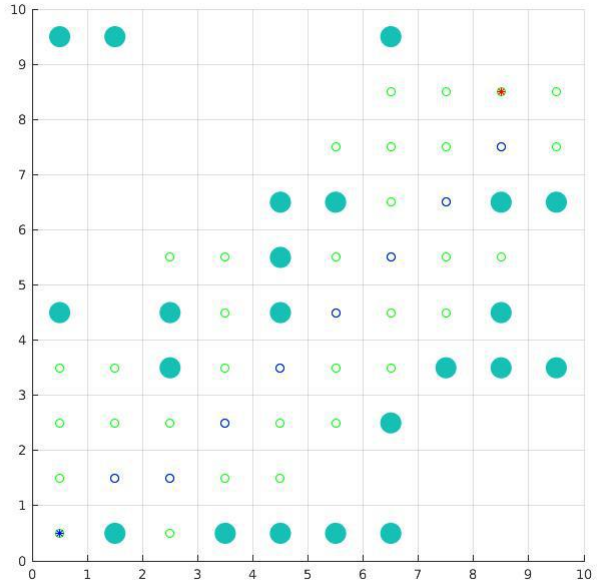


Figure 3.

ros version.

Main content:

1> c++ code and modification description.(including a star and jps)

According to the requirements:

Different GetHeu computation method are implemented.

AtarGetSucc method is implemented.

Getpath is implemented.

AstarGraphSearch is implemented.

JpsGraghSearch is implemented.(most of the code here is same with that of AstarGraphSearch )

2> Execution results and comparison between manhattan, euclidean, diagonal for a star.

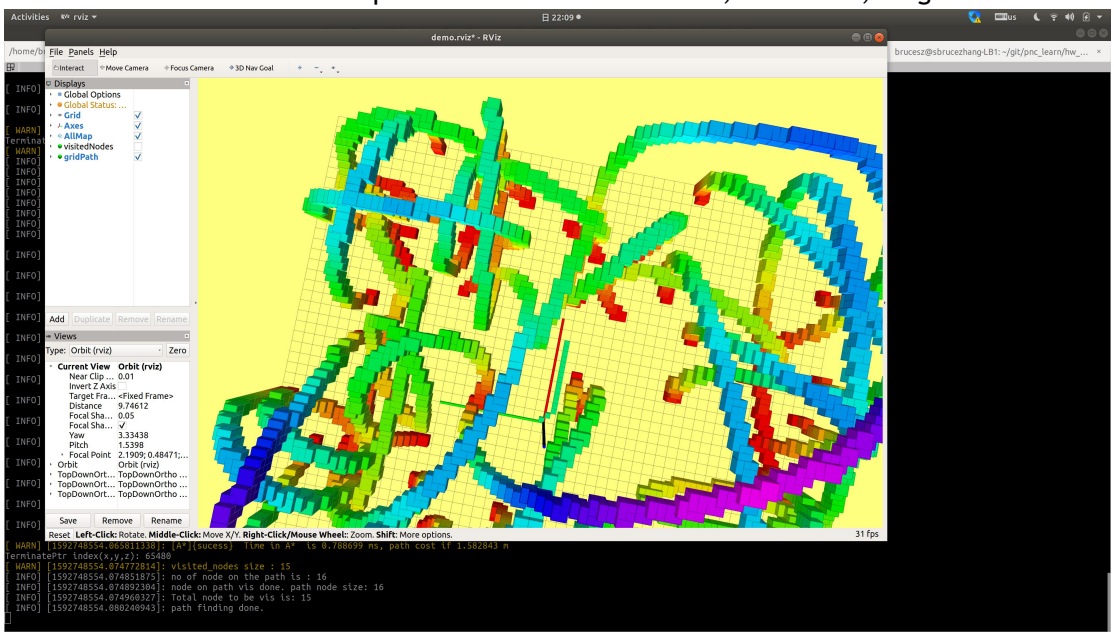


Figure 1: a_star_resul_with_manhattan1

Execution time of A*:

[ INFO] [1592748554.065756364]: ALL neighbor size is: 17

[ WARN] [1592748554.065811338]: [A*]{sucess} Time in A* is 0.788699 ms, path cost if 1.582843 m

TerminatePtr index(x,y,z): 65480

[ WARN] [1592748554.074772814]: visited_nodes size : 15

[ INFO] [1592748554.074851875]: no of node on the path is : 16

[ INFO] [1592748554.074892304]: node on path vis done. path node size: 16

[ INFO] [1592748554.074960327]: Total node to be vis is: 15

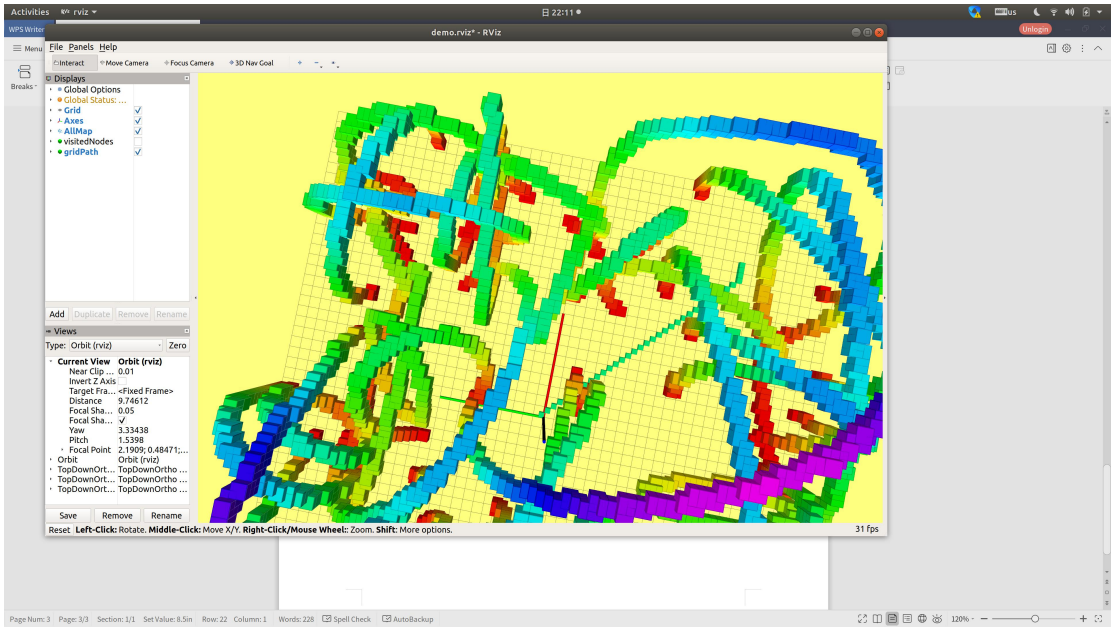


Figure 2 a_star_resul_with_manhattan2

Execution result of A star:

[ WARN] [1592748693.186910870]: [A*]{sucess}  Time in A*  is 1.897427 ms, path cost if 4.825483 m

TerminatePtr index(x,y,z): 85180

[ WARN] [1592748693.192267492]: visited_nodes size : 35

[ INFO] [1592748693.192292815]: no of node on the path is : 36

[ INFO] [1592748693.192305911]: node on path vis done. path node size: 36

[ INFO] [1592748693.192325828]: Total node to be vis is: 35

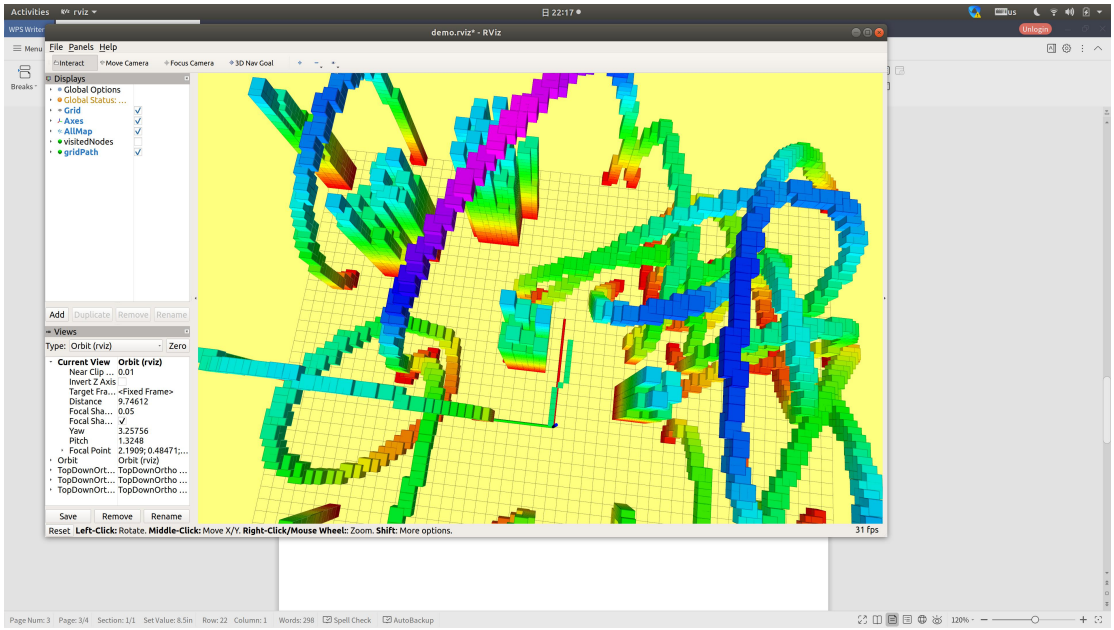


Figure 3:

Execution result:

[ WARN] [1592749023.839943956]: [A*]{sucess}  Time in A*  is 6.720472 ms, path cost if 1.582843 m
TerminatePtr index(x,y,z): 65480
[ WARN] [1592749023.842695967]: visited_nodes size : 57
[ INFO] [1592749023.842731500]: no of node on the path is : 16
[ INFO] [1592749023.842739965]: node on path vis done. path node size: 16
[ INFO] [1592749023.842797845]: Total node to be vis is: 57

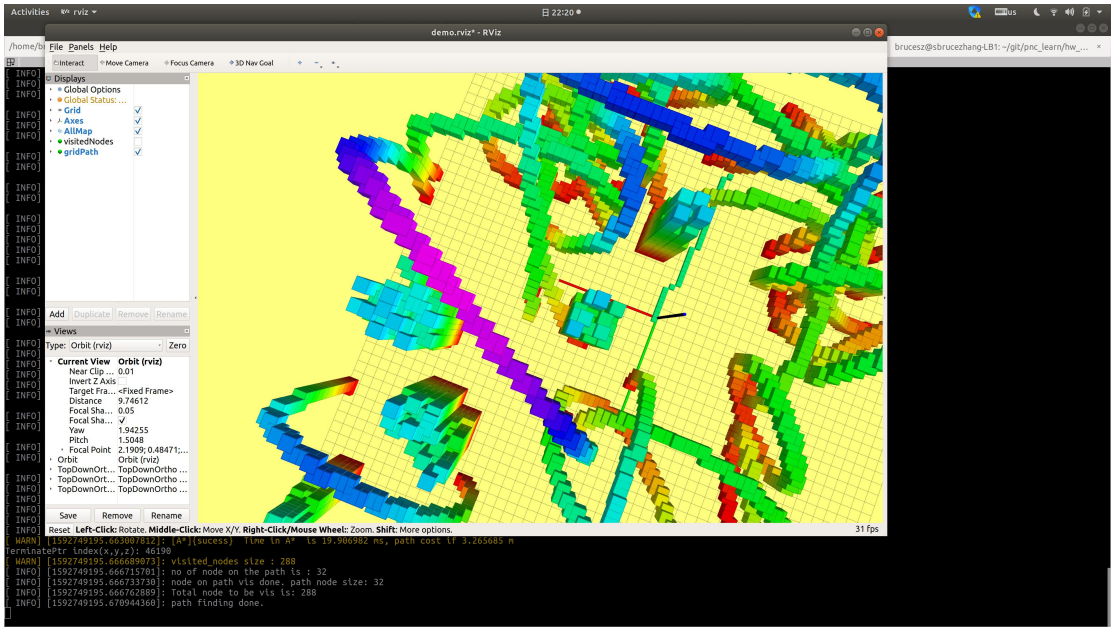


Figure 4:

Execution results:

[ WARN] [1592749195.663007812]: [A*]{sucess}  Time in A*  is 19.906982 ms, path cost if 3.265685 m
TerminatePtr index(x,y,z): 46190
[ WARN] [1592749195.666689073]: visited_nodes size : 288
[ INFO] [1592749195.666715701]: no of node on the path is : 32
[ INFO] [1592749195.666733730]: node on path vis done. path node size: 32
[ INFO] [1592749195.666762889]: Total node to be vis is: 288

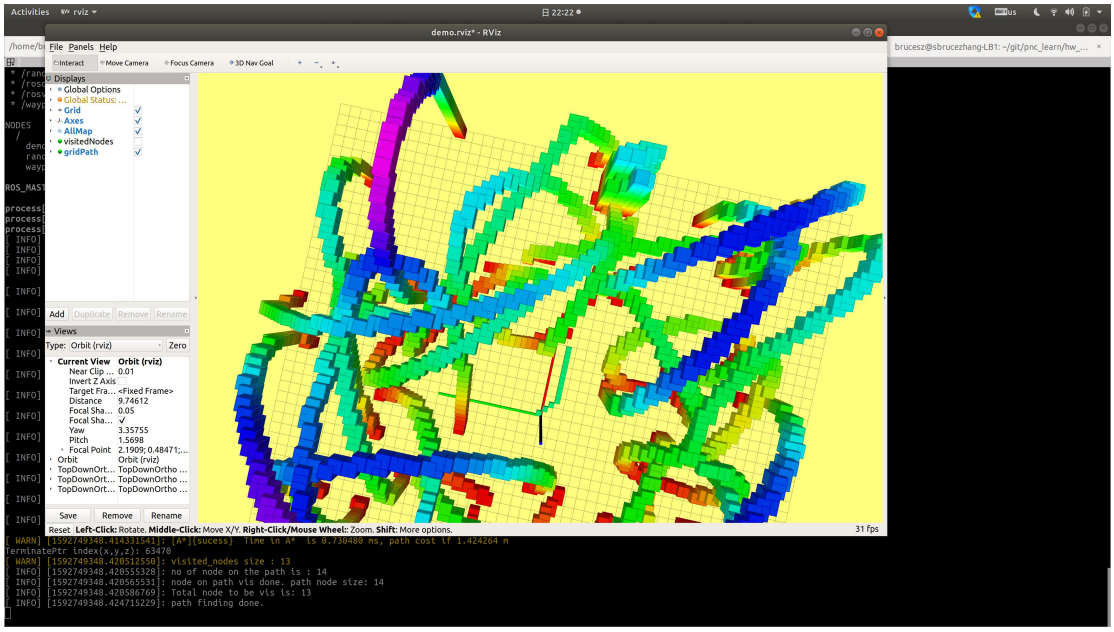


Figure 5

Execution result:

[ WARN] [1592749348.414331541]: [A*]{sucess} Time in A* is 0.730480 ms, path cost if 1.424264 m

TerminatePtr index(x,y,z): 63470

[ WARN] [1592749348.420512550]: visited_nodes size : 13

[ INFO] [1592749348.420555328]: no of node on the path is : 14

[ INFO] [1592749348.420565531]: node on path vis done. path node size: 14

[ INFO] [1592749348.420586769]: Total node to be vis is: 13

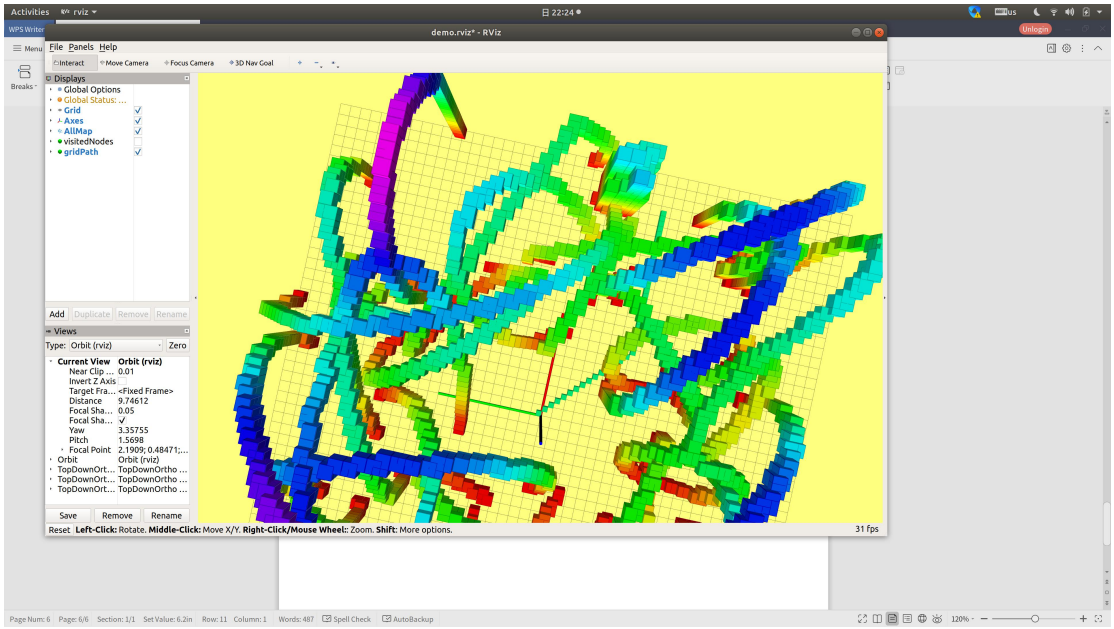


Figure 6

Execution result:

[ WARN] [1592749466.301303236]: [A*]{sucess} Time in A* is 0.525109 ms, path cost if 4.821320 m

TerminatePtr index(x,y,z): 92350
[ WARN] [1592749466.304836610]: visited_nodes size : 42
[ INFO] [1592749466.304852836]: no of node on the path is : 43
[ INFO] [1592749466.304862572]: node on path vis done. path node size: 43
[ INFO] [1592749466.304878636]: Total node to be vis is: 42
[ INFO] [1592749466.309298711]: path finding done.

Conclusion:
Diagonal is the fastest .
Manhattan is faster than eclidean in my scenario.

3> Execution results and comparison between whether using tie breaker.
(same execution condition except tie breaker, comparing with figure 5 and figure 6)

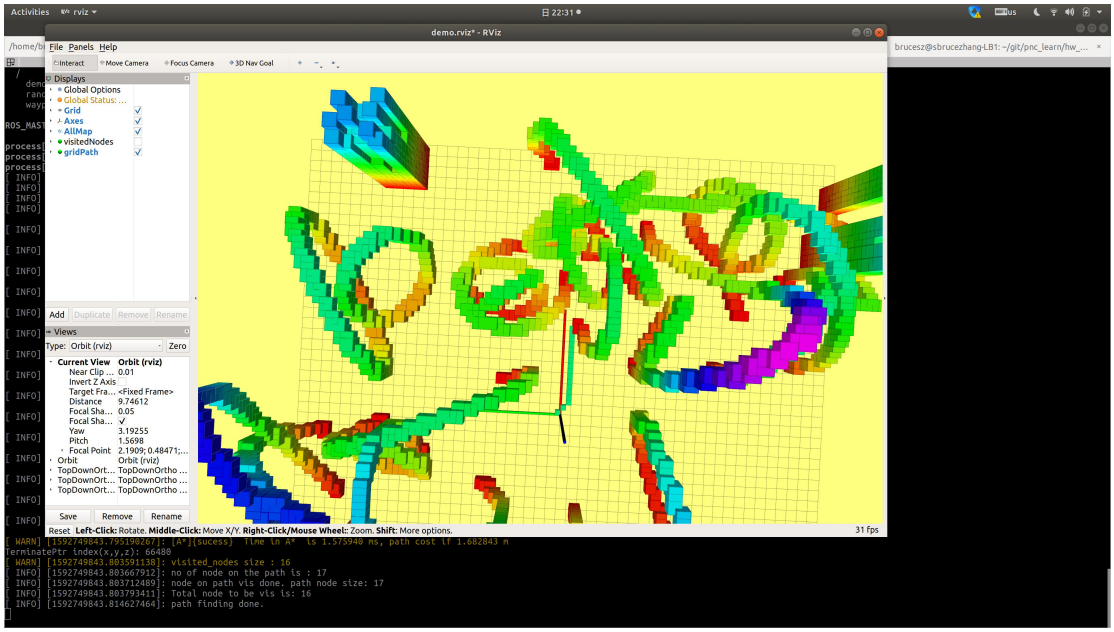


Figure 7 : no_tie_breaker_compare_with_figure5

Execution result:
[ WARN] [1592749843.795190267]: [A*]{sucess} Time in A* is 1.575940 ms, path cost if 1.682843 m
TerminatePtr index(x,y,z): 66480
[ WARN] [1592749843.803591138]: visited_nodes size : 16
[ INFO] [1592749843.803667912]: no of node on the path is : 17
[ INFO] [1592749843.803712489]: node on path vis done. path node size: 17
[ INFO] [1592749843.803793411]: Total node to be vis is: 16

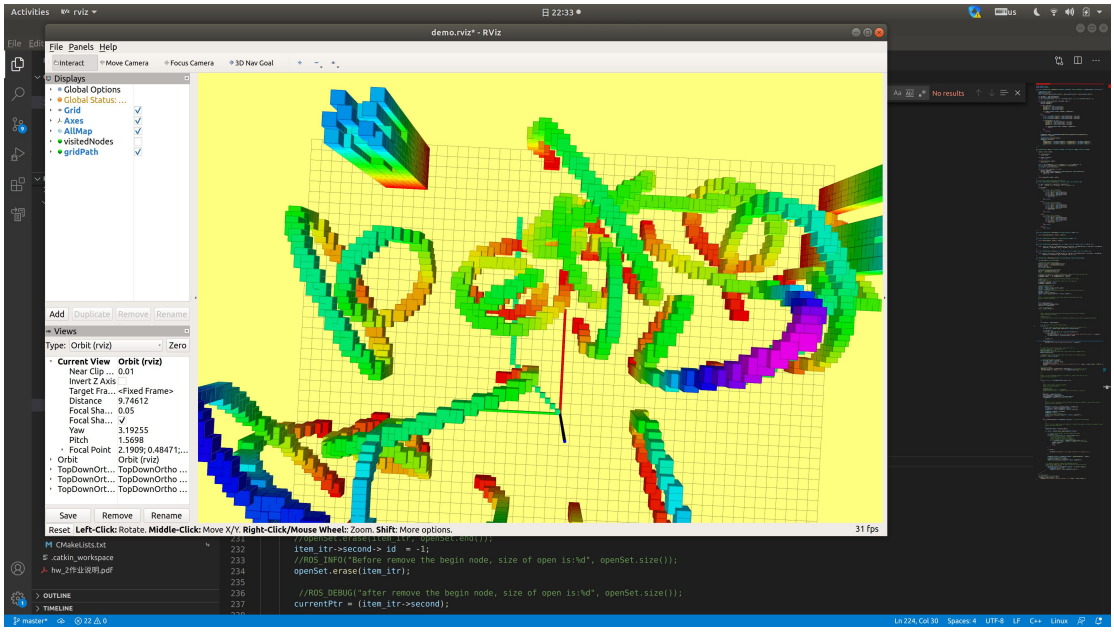


Figure 8: no_tie_breaker_compare_with_figure_7

Execution result:

[ WARN] [1592749994.607148525]: [A*]{sucess}  Time in A*  is 2.086032 ms, path cost if 3.972792 m

TerminatePtr index(x,y,z): 86590

[ WARN] [1592749994.615654085]: visited_nodes size : 36

[ INFO] [1592749994.615725792]: no of node on the path is : 37

[ INFO] [1592749994.615767838]: node on path vis done. path node size: 37

[ INFO] [1592749994.615839312]: Total node to be vis is: 36

[ INFO] [1592749994.627182202]: path finding done.

Conclusion:

With no tie_breaker, the a star can be much slower.

4> Execution results and comparison between jps and a star.

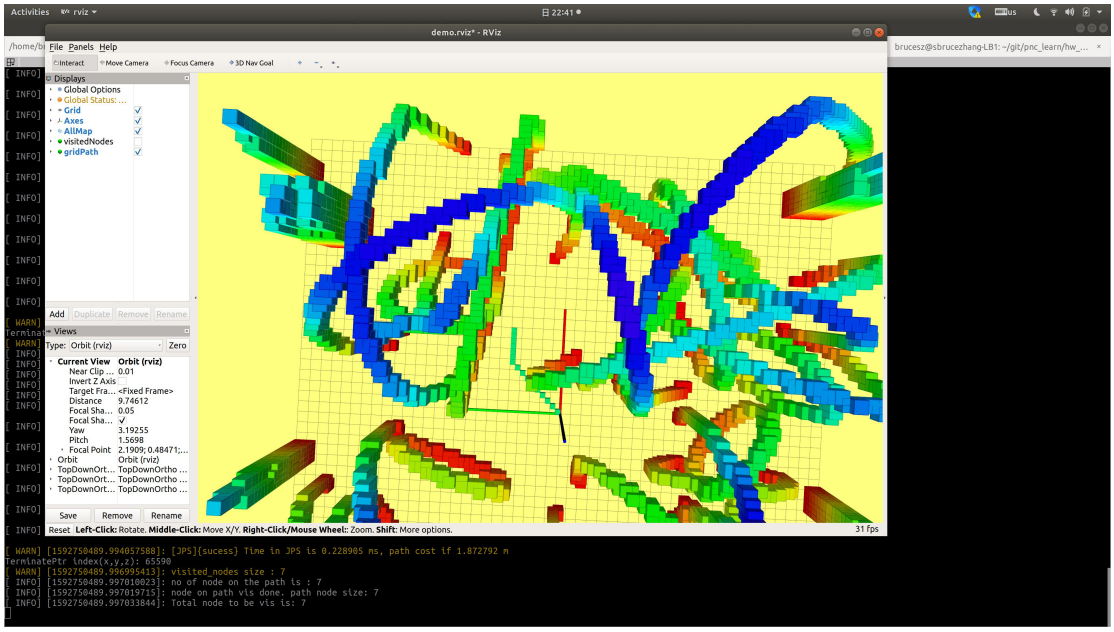


Figure 9 jps

Execution result:

[ WARN] [1592750489.982565561]: [A*]{sucess} Time in A* is 0.897353 ms, path cost if 1.872792 m

TerminatePtr index(x,y,z): 65590

[ WARN] [1592750489.988625835]: visited_nodes size : 15

[ INFO] [1592750489.988668516]: no of node on the path is : 16

[ INFO] [1592750489.988691052]: node on path vis done. path node size: 16

[ INFO] [1592750489.988728007]: Total node to be vis is: 15

[ INFO] [1592750489.993796206]: path finding done.

[ INFO] [1592750489.993817232]: BEGIN jps SEARCH.

[ INFO] [1592750489.993940607]: ALL neighbor size is: 13

[ INFO] [1592750489.993957518]: ALL neighbor size is: 2

[ INFO] [1592750489.993973134]: ALL neighbor size is: 2

[ INFO] [1592750489.993989333]: ALL neighbor size is: 2

[ INFO] [1592750489.994008443]: ALL neighbor size is: 2

[ INFO] [1592750489.994031409]: ALL neighbor size is: 2

[ INFO] [1592750489.994043976]: ALL neighbor size is: 2

[ WARN] [1592750489.994057588]: [JPS]{sucess} Time in JPS is 0.228905 ms, path cost if 1.872792 m

TerminatePtr index(x,y,z): 65590

[ WARN] [1592750489.996995413]: visited_nodes size : 7

[ INFO] [1592750489.997010023]: no of node on the path is : 7
[ INFO] [1592750489.997019715]: node on path vis done. path node size: 7
[ INFO] [1592750489.997033844]: Total node to be vis is: 7

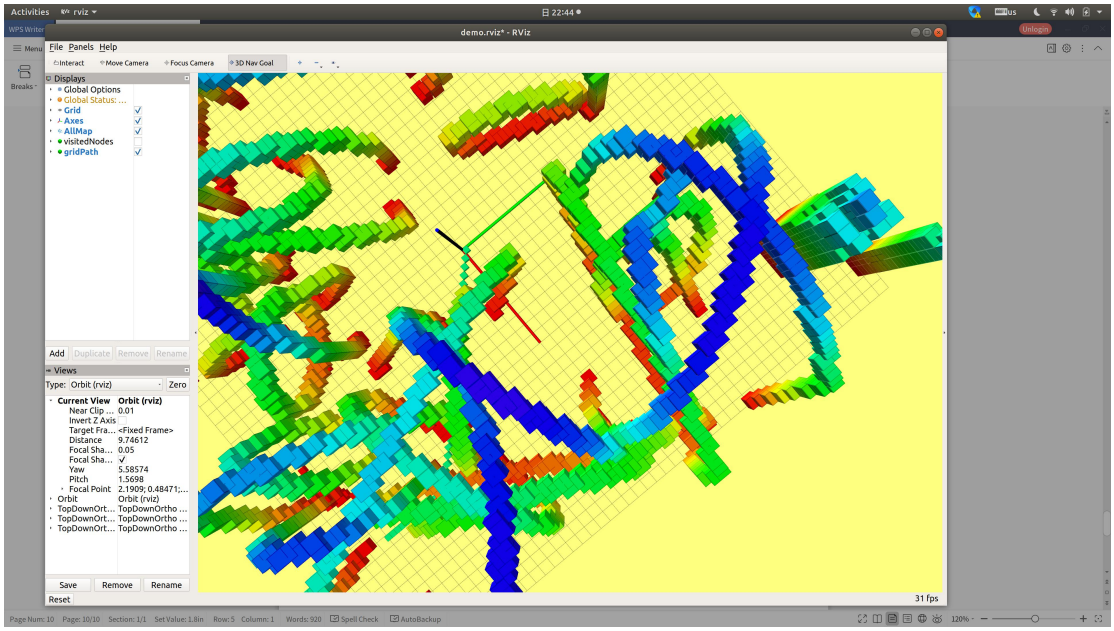


Figure 10: jps2_more_complex env.

Execution res:

[ WARN] [1592750645.655980345]: [A*]{sucess}  Time in A*  is 1.695735 ms, path cost if 5.159798 m
TerminatePtr index(x,y,z): 78100
[ WARN] [1592750645.660856216]: visited_nodes size : 40
[ INFO] [1592750645.660914905]: no of node on the path is : 41
[ INFO] [1592750645.660935571]: node on path vis done. path node size: 41
[ INFO] [1592750645.661001111]: Total node to be vis is: 40
[ INFO] [1592750645.665755131]: path finding done.
[ INFO] [1592750645.665772676]: BEGIN jps SEARCH.
[ INFO] [1592750645.665943426]: ALL neighbor size is: 13

[ INFO] [1592750645.665964786]: ALL neighbor size is: 3

[ INFO] [1592750645.665980700]: ALL neighbor size is: 3

[ INFO] [1592750645.666025444]: ALL neighbor size is: 4

[ INFO] [1592750645.666046312]: ALL neighbor size is: 3

[ INFO] [1592750645.666089795]: ALL neighbor size is: 10

[ INFO] [1592750645.666131713]: ALL neighbor size is: 2

[ INFO] [1592750645.666150231]: ALL neighbor size is: 2

[ INFO] [1592750645.666163639]: ALL neighbor size is: 1

[ INFO] [1592750645.666177897]: ALL neighbor size is: 2

[ INFO] [1592750645.666193008]: ALL neighbor size is: 1

[ INFO] [1592750645.666207188]: ALL neighbor size is: 2

[ INFO] [1592750645.666223469]: ALL neighbor size is: 2

[ INFO] [1592750645.666257378]: ALL neighbor size is: 1

[ INFO] [1592750645.666319116]: ALL neighbor size is: 12

[ INFO] [1592750645.666360206]: ALL neighbor size is: 3

[ INFO] [1592750645.666378985]: ALL neighbor size is: 2

[ INFO] [1592750645.666394647]: ALL neighbor size is: 2

[ INFO] [1592750645.666410627]: ALL neighbor size is: 3

[ INFO] [1592750645.666431311]: ALL neighbor size is: 8

[ INFO] [1592750645.666448336]: ALL neighbor size is: 2

[ INFO] [1592750645.666462284]: ALL neighbor size is: 0

[ INFO] [1592750645.666477629]: ALL neighbor size is: 2

[ INFO] [1592750645.666492819]: ALL neighbor size is: 1

[ INFO] [1592750645.666508283]: ALL neighbor size is: 1

[ INFO] [1592750645.666523848]: ALL neighbor size is: 2

[ INFO] [1592750645.666538881]: ALL neighbor size is: 1

[ INFO] [1592750645.666554146]: ALL neighbor size is: 2

[ INFO] [1592750645.666569513]: ALL neighbor size is: 2

[ INFO] [1592750645.666584357]: ALL neighbor size is: 1

[ INFO] [1592750645.666599228]: ALL neighbor size is: 3

[ INFO] [1592750645.666614488]: ALL neighbor size is: 0

[ INFO] [1592750645.666633107]: ALL neighbor size is: 8

[ INFO] [1592750645.666651523]: ALL neighbor size is: 3

[ INFO] [1592750645.666671831]: ALL neighbor size is: 3

[ WARN] [1592750645.666689457]: [JPS]{sucess} Time in JPS is 0.900314 ms, path cost if 7.223778 m
TerminatePtr index(x,y,z): 78100
[ WARN] [1592750645.669603094]: visited_nodes size : 33
[ INFO] [1592750645.669621218]: no of node on the path is : 19
[ INFO] [1592750645.669636930]: node on path vis done. path node size: 19
[ INFO] [1592750645.669658552]: Total node to be vis is: 33

Conclusion:
1> In simple env,(no obstacles), the jps is way faster than a star,
2> In more complex case, jps is still faster than a_star, (in my experiment), but the speed differences is not that much compared to 1>