

Motion Planning HW2 report

Xie Fujing

2023.01.15

1. Work flow:

- Complete heuristic function, tried three functions, 0 stands for Euclidean distance, 1 stands for Manhattan distance, 2 stands for Diagonal Heuristic, which can be changed by "heu_type" in launch file.
- Set startPtr's id as 1, means it is in the open set, and insert it to "openSet".
- Set start position in GridNodeMap to 1, means it is in openSet.
- In the main loop, get the lowest cost node from openSet, erase it from openSet, and set the position in GridNodeMap to -1.
- Finish "AstarGetSucc" function, traverse all nodes next to current node, if the node is free and not in closedSet, add it to neighborPtrSets and calculate distance between this node and current node.
- Transverse all neighbors of current node, if it is a new node, add it to openSet, record its parent, and calculate its gScore and fScore. If it is already in the openSet, check if this road has a lower cost, if so update its score and parent.
- From the terminatePtr's cameFrom to find its parent until find startPtr.

2. Running Result

Running result as image below.

```
[ WARN ] [1673779667.260923817]: 3D Goal Set
[ INFO ] [1673779667.272258383]: [node] receive the planning target
[ WARN ] [1673779667.276375530]: [A*](success),Heu=Euclidean, Time In A* is 1.324
130 ms, path cost tf 6.823220 n
[ WARN ] [1673779667.277552212]: visited_nodes size : 306
[ WARN ] [1673779670.090508322]: 3D Goal Set
[ INFO ] [1673779670.102128805]: [node] receive the planning target
[ WARN ] [1673779670.104588078]: [A*](success),Heu=Euclidean, Time In A* is 2.320
739 ms, path cost tf 5.204916 n
[ WARN ] [1673779670.105656240]: visited_nodes size : 859
[ WARN ] [1673779671.722852172]: 3D Goal Set
[ INFO ] [1673779671.732181472]: [node] receive the planning target
[ WARN ] [1673779671.735025228]: [A*](success),Heu=Euclidean, Time In A* is 2.723
972 ms, path cost tf 6.491849 n
[ WARN ] [1673779671.736306081]: visited_nodes size : 810
[ WARN ] [1673779674.046906055]: 3D Goal Set
[ INFO ] [1673779674.052161671]: [node] receive the planning target
[ WARN ] [1673779674.054944772]: [A*](success),Heu=Euclidean, Time In A* is 2.660
888 ms, path cost tf 6.491849 n
[ WARN ] [1673779674.056109572]: visited_nodes size : 743
[ WARN ] [1673779676.808657070]: 3D Goal Set
[ INFO ] [1673779676.812116211]: [node] receive the planning target
[ WARN ] [1673779676.815248075]: [A*](success),Heu=Euclidean, Time In A* is 2.991
987 ms, path cost tf 6.774691 n
[ WARN ] [1673779676.816525384]: visited_nodes size : 813
[ WARN ] [1673779679.376643614]: 3D Goal Set
[ INFO ] [1673779679.382121777]: [node] receive the planning target
[ WARN ] [1673779679.385528165]: [A*](success),Heu=Euclidean, Time In A* is 3.304
145 ms, path cost tf 5.697736 n
[ WARN ] [1673779679.386628347]: visited_nodes size : 1225
```

Figure 1:

3. Compare different heuristic function

Use Diagonal Heuristic as heuristic function can decrease the number of visited node as can be seen from image below: 32 if use Diagonal distance as heuristic cost, 339 if use Euclidean distance as heuristic cost.

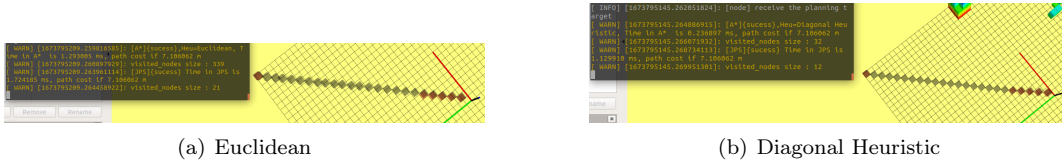


Figure 2:

4. Tie break function influence

The tie breaker simply return slightly different hScore, and in simple environment, it has less visted nodes.

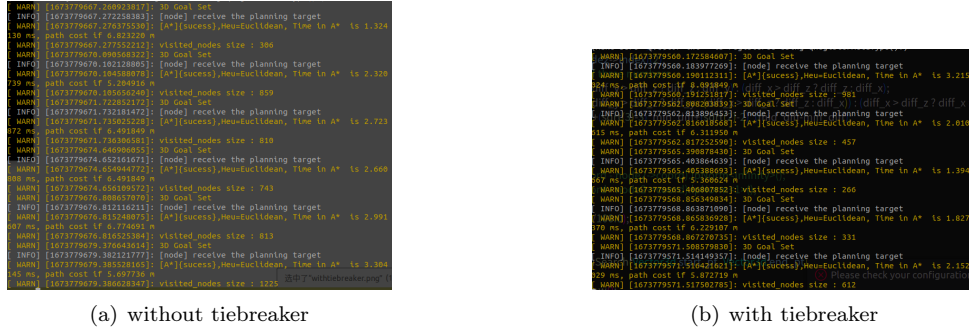


Figure 3:

5. Compare A^* and JPS In simple environment like below, the JPS takes less time to find path.

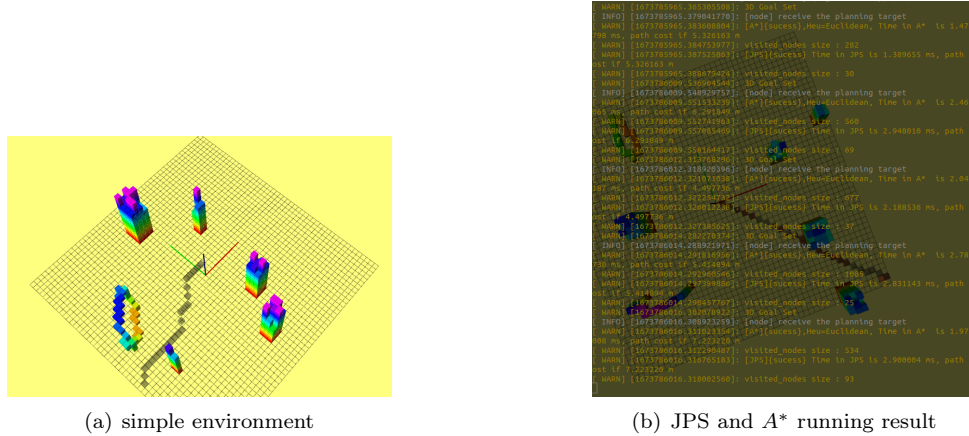
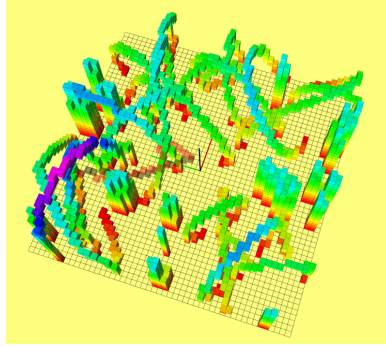
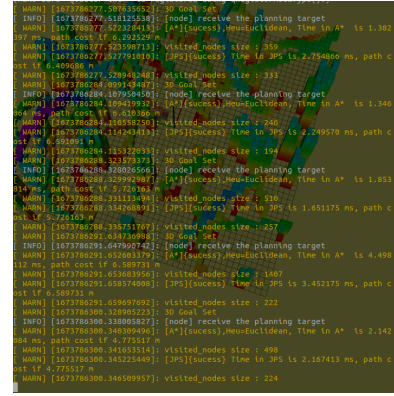


Figure 4:

But in complex environment, sometimes JPS takes more time than A^* .



(a) complex environment



(b) JPS and A* running result

Figure 5: