A*

source devel/setup.bash
roslaunch grid_path_searcher demo.launch
结果见视频
这是使用欧式距离，并使用了 tie breaker 的结果

```
[ WARN] [1616037115.304149472]: 3D Goal Set
[ INFO] [1616037115.308785683]: [node] receive the planning target
[ WARN] [1616037115.313473867]: [A*]{sucess}  Time in A*  is 4.573504 ms, path c
ost if 1.176539 m
[ WARN] [1616037115.314690425]: visited_nodes size : 1007
```

这是改为对角距离和 tie breaker

```
[ WARN] [1616039121.468967172]: 3D Goal Set
[ INFO] [1616039121.476961474]: [node] receive the planning target
[ WARN] [1616039121.480101482]: [A*]{sucess}  Time in A*  is 0.184490 ms, path c
ost if 1.024249 m
[ WARN] [1616039121.481616465]: visited_nodes size : 21
```

可见时间和扩展的节点都缩小很多，我还是选择了差不多的一个目标点
再改为曼哈顿距离和 tie breaker

```
[ WARN] [1616039856.990286349]: 3D Goal Set
[ INFO] [1616039857.004233113]: [node] receive the planning target
[ WARN] [1616039857.004585860]: [A*]{sucess}  Time in A*  is 0.231602 ms, path c
ost if 1.131536 m
[ WARN] [1616039857.005673560]: visited_nodes size : 24
```

感觉看不太出来，没有显著的变化
最后看 dijkstra 算法

```
[ WARN] [1616040066.731155323]: 3D Goal Set
[ INFO] [1616040066.736834978]: [node] receive the planning target
[ WARN] [1616040066.798518257]: [A*]{sucess}  Time in A*  is 58.710723 ms, path
cost if 1.147257 m
[ WARN] [1616040066.799862514]: visited nodes size : 42426
```

这时间和拓展的节点也太多了
总结下，在有 tie breaker 的情况下，欧式距离用的时间最长，对角距离和曼哈顿距离差不多，但是有贪心项比没贪心项的 dijkstra 强多了。

再看看 tie breaker 的情况
曼哈顿距离和没有 tie breaker：

```
[ WARN] [1616040508.328141297]: 3D Goal Set
[ INFO] [1616040508.336653333]: [node] receive the planning target
[ WARN] [1616040508.339749168]: [A*]{sucess}  Time in A*  is 0.266271 ms, path c
ost if 1.196963 m
[ WARN] [1616040508.340920409]: visited_nodes size : 39
```

感觉并没有太大区别
用欧式距离似乎比之前用了 tie breaker 多用了点时间，但是并不是每次都会多

```
[ WARN] [1616040660.585024231]: 3D Goal Set
[ INFO] [1616040660.594767778]: [node] receive the planning target
[ WARN] [1616040660.604916249]: [A*]{sucess}  Time in A*  is 6.777027 ms, path c
ost if 1.140394 m
[ WARN] [1616040660.606028262]: visited nodes size : 1179
```

用对角线距离

```
[ WARN] [1616040850.925781444]: 3D Goal Set
[ INFO] [1616040850.928491477]: [node] receive the planning target
[ WARN] [1616040850.930691084]: [A*]{sucess}  Time in A*  is 0.612911 ms, path c
ost if 1.309088 m
[ WARN] [1616040850.932141088]: visited_nodes size : 279
```

可以看到拓展的节点数明显多了很多，这大概是因为没用 tie breaker 造成的?

再用 JPS，把 demo_node.cpp 中的_use_jps 置 1，再把 A*中的复制一下，就行了。
前两次取的目标点在空间比较大的地方，后一次取在几根柱子夹缝中：

```
[ WARN] [1616046491.337667128]: [A*]{sucess}  Time in A*  is 0.295806 ms, path c
ost if 1.071112 m
[ WARN] [1616046491.338266075]: visited_nodes size : 122
[ WARN] [1616046491.371843911]: [JPS]{sucess} Time in JPS is 32.695268 ms, path
cost if 5.355562 m
[ WARN] [1616046491.372602269]: visited_nodes size : 8700
[ WARN] [1616046502.215665029]: 3D Goal Set
[ INFO] [1616046502.223544028]: [node] receive the planning target
[ WARN] [1616046502.223771916]: [A*]{sucess}  Time in A*  is 0.177711 ms, path c
ost if 0.986274 m
[ WARN] [1616046502.224465654]: visited_nodes size : 91
[ WARN] [1616046502.243780513]: [JPS]{sucess} Time in JPS is 17.851592 ms, path
cost if 4.931371 m
[ WARN] [1616046502.245099780]: visited_nodes size : 6989
[ WARN] [1616046529.885851355]: 3D Goal Set
[ INFO] [1616046529.896415017]: [node] receive the planning target
[ WARN] [1616046529.901985267]: [A*]{sucess}  Time in A*  is 5.433131 ms, path c
ost if 1.266245 m
[ WARN] [1616046529.903191717]: visited_nodes size : 980
[ WARN] [1616046529.972165908]: [JPS]{sucess} Time in JPS is 67.476022 ms, path
cost if 7.023365 m
[ WARN] [1616046529.972861671]: visited_nodes size : 17053
```

JPS 的效率比 A*差太多了，我最后一次取点在几个柱子之间狭窄的缝隙中，还以为 JPS 会好一点，没想到依然这么差劲。