算法流程

*初始化 openlist（采用优先队列提高效率）

*将起点加入 openlist，（id=1）

* while(openlist 非空)

*取出 openlist 中 f 值最小的节点

*将该节点从 openlist 中删除，并加入 closedlist，（id=-1）

*If(当前节点是终点)

*算法结束，break，从终点递归找到起点。

*for (扩展当前节点的所有邻接节点, 2D (8 个), 3D(26 个)):

*if(当前邻点没有被探索过):

*更新 f 值等，并加入 openlist

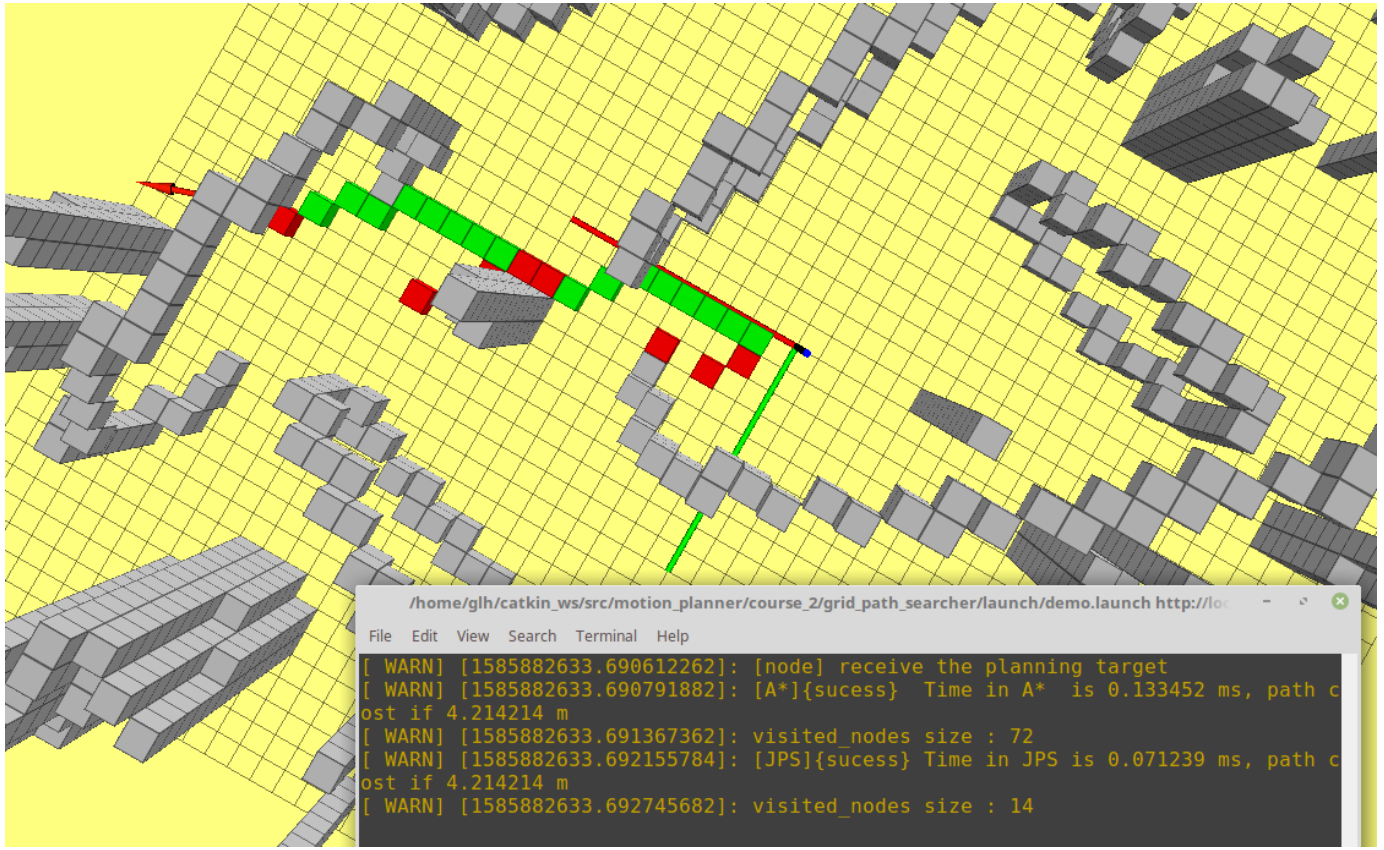
*elseif(当前邻点通过当前节点到达的代价更小):

*更新邻点的 g，f。

*对 openlist 重新排序一次。

Astar与JPS运行结果

- 绿色为Astar运行结果
- 红色为JPS运行结果

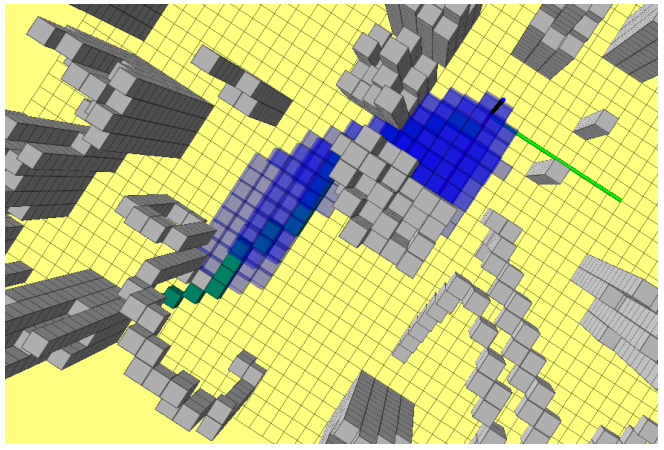


运行结果对比分析：

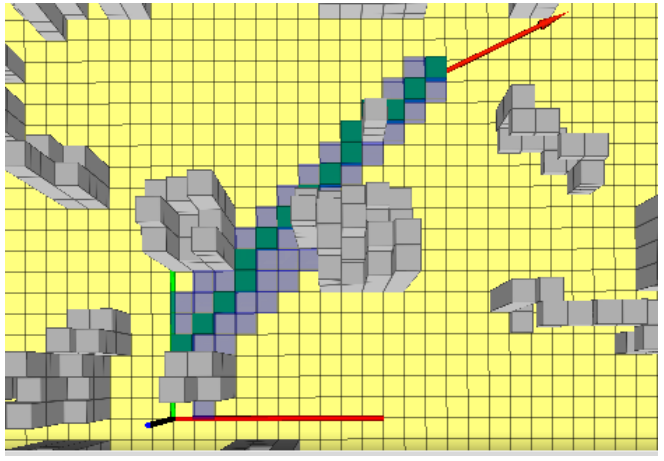
- 明显看出openlist节点数量A*(72个)明显多于JPS(14)
- JPS相对于A*算法虽然openlist的操作大大减少，但是在扩展邻接节点上要花费更多的时间，所以如果在相对空旷的区域，JPS花费的时间有可能比A*更多，比如室外运行的机器人。
- 另外JPS只能用于此类规则的栅格地图，而A*没有此限制。

不同启发函数对算法效果影响

- 启发函数如果满足h(n) <= h*(n)，也就是预估的h值小于等于真实的h*，那么A*结果就是最优的。
- 栅格地图中我们定义机器人可以斜对角运动，右图图中上边欧拉距离，下边对角距离，能够明显看出采用欧拉距离遍历的节点数量要比对角距离多，因为采用对角距离更贴近我们实际的h*。
- 同样，因为曼哈顿距离不一定满足h(n) <= h*(n)，所以得到的路径不能保证最优。



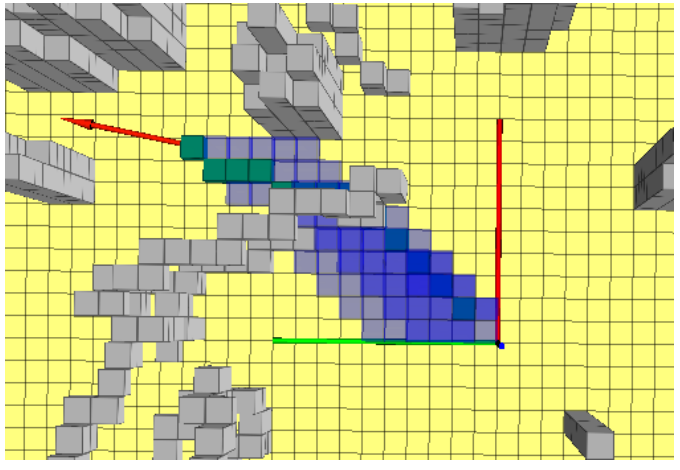
欧拉距离



对角距离

关于tie breaker

- Tie breaker 目的是为了打破路径的对称性，因为在路径中很有可能存在f值相同的路径，这样就会有效减少openlist数量，提高算法效率。
- 右图中采用欧拉距离对h函数+1/1000。
- 实际使用中，我们有时不能单单只考虑算法速度，还需要考虑生成的路径是否便于追踪，如果找到的路径过于贴近障碍物可能也不好。



Tie breaker