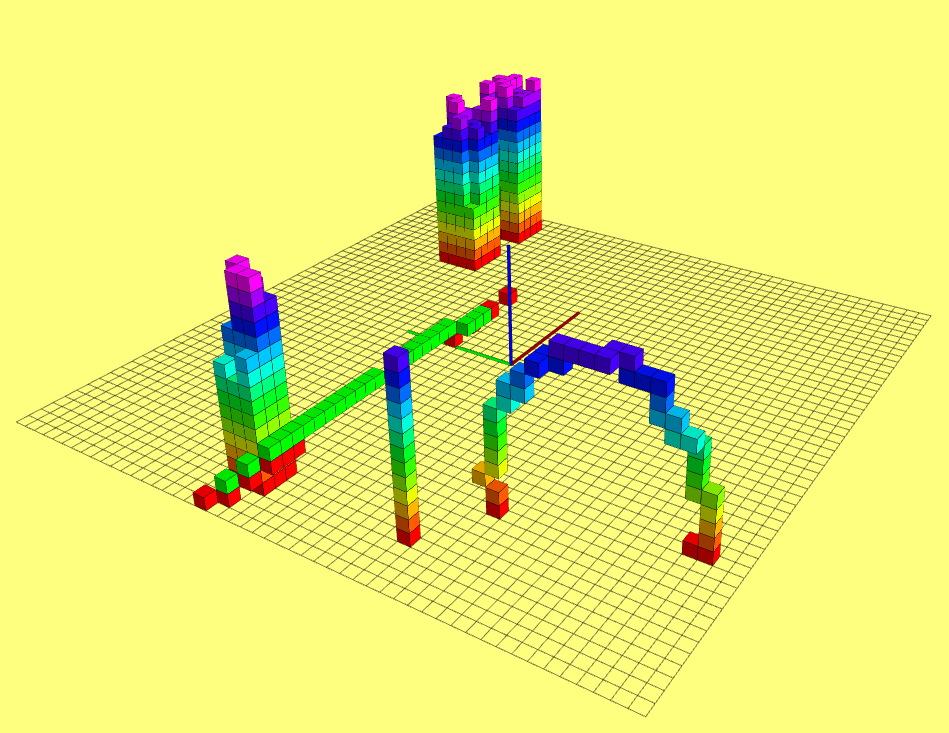
## Optimal A\* VS JPS

算法流程

日程表

中度可信度描述已自动生成

实现结果



### 采用的Heuristic Function

三维栅格地图下，最短路径问题的最优解：Diagnal heuristic 3d + simple tie breaker

 double hn = (sqrt(3)-sqrt(2))\*dmin + (sqrt(2)-1)\*dmid + dmax;

  //simple tie breaker

hn = hn\*(1+1/1000);

## 两种路径搜索方法的对比

### 数据背景：

两个场景

Sparse：circle\_num: 1; obs\_num: 5

Dense：circle\_num: 50; obs\_num: 1000

数据：对于四个目标点（尽量在顶点附近（-5,-5,0), (5,5,0),(-5, 5, 5), (5,-5,5)），生成路径所遍历的节点的个数，及所用时间的平均值。

两组对比：

对比1：A\* VS JPS without tie breaker

表格

描述已自动生成

对比2：A\* VS JPS with tie breaker

表格

描述已自动生成

### 数据分析

A\*和JPS区别：

A\*在稀疏环境下，表现要好于JPS，这也符合理论，因为JPS花费大量时间在搜索边界上了，而不是专注向目标前进。

相反，在复杂场景下，JPS表现更优，而且在实际中遇到过40ms VS 4ms的情况，当时的目标点与起始点之间有很多障碍物，可以表明越是在复杂的场景下，JPS的效果越好。理论上解释，就是得益于它跳跃的特点从而节省了遍历中间无用节点的时间，而且复杂的场景的边界一般很容易触及，这也弥补了JPS相对于A\*的不足。

Tie Breaker的作用

就结果来说，tie breaker在复杂的场景下发挥着比较好的作用，但在环境简单情况下并没有那么好。这有可能是数据的原因：对比的时候地图有差异(因为需要重新编译运行)，或者是tie breaker设置的过于简单。

### 遇到的问题和解决方法

Multimap的遍历 🡪 运用同类型的iterator遍历map，并于first, second的方式指向元素。

Index和coord两种坐标表示方法之间的区别，计算edgeCost和heuristic cost时使用的坐标不一致导致的各种问题。🡪 在loop中统一使用index，显示在地图上时用coord

三维中对角线启发函数的计算 🡪 先假设起始点沿3d对角线方向移动直到和终点在同一平面上，将三维问题简化为二维问题。