**移动机器人规划与控制期末报告**

学号：22354119 姓名： 唐嘉隆 提交日期：2025.1.23

提交的Github仓库链接：

<https://github.com/Tang0705ui/Mobile-Robot-Planning-Assignment-and-control>

**实验一：**

电脑萤幕画面

描述已自动生成

**1.1 v\_dot 的计算与分析**

v\_dot=(1/mass\_)\*(thrust\*cur\_state.R.col(2)-g\_\*mass\_\*Eigen::Vector3d::UnitZ() + external\_force\_ - resistance \* vnorm);

v\_dot 是飞行器的线加速度，受以下因素影响：

（1）推力：thrust \* cur\_state.R.col(2)：总推力在飞行器机体坐标系下的方向是 R.col(2)（即 z 轴方向），乘以总推力得到推力向量。cur\_state.R 是飞行器的姿态矩阵，将推力从机体坐标系转换到世界坐标系。thrust 是总推力，Eigen::Vector3d::UnitZ() 是单位向量 [0, 0, 1]，表示推力方向。

（2）重力：g\_ \* mass\_ \* Eigen::Vector3d::UnitZ()：重力作用在地面坐标系下的方向是 -z 轴方向，乘以质量和重力加速度得到重力向量。g\_ 是重力加速度，mass\_ 是飞行器的质量，Eigen::Vector3d::UnitZ() 是单位向量 [0, 0, 1]，表示重力方向。

（3）外力：external\_force\_：外部环境对飞行器施加的力。

（4）空气阻力：方向与速度方向相反。resistance \* vnorm：空气阻力，resistance 是阻力系数，vnorm 是速度的单位向量。

（5）(1/mass\_) 将所有力除以质量，得到加速度。

**1.2omega\_dot 的的计算与分析**

omega\_dot=J\_.inverse()\*(moments-cur\_state.omega.cross(J\_\*cur\_state.omega) + external\_moment\_);

omega\_dot 是飞行器的角加速度，受以下因素影响：

（1）力矩：moments：由四个旋翼产生的力矩。

（2）科里奥利力：cur\_state.omega.cross(J\_ \* cur\_state.omega)：科里奥利力，表示角速度与惯性矩阵乘以角速度的叉积。

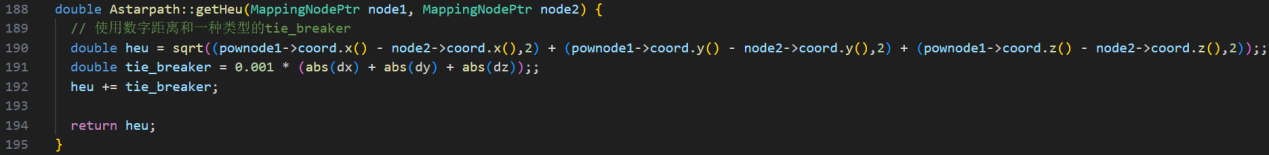
（3）外部力矩：external\_moment\_：外部环境对飞行器施加的力矩。

J\_.inverse()：惯性矩阵的逆矩阵，用于将力矩转换为角加速度。

**1.3总结**

这两行代码分别计算了飞行器的线加速度 v\_dot 和角加速度 omega\_dot，考虑了推力、重力、外力、空气阻力、力矩、科里奥利力和外部力矩等因素。这些计算确保了飞行器的动力学模型能够准确反映其运动特性。

**实验二：**



**2.1 getHeu的的计算与分析**

这段代码实现了 A\* 算法中的启发式函数 getHeu，用于估计从当前节点目标节点的代价。启发式函数在 A\* 算法中起着关键作用，影响路径搜索的效率和效果。

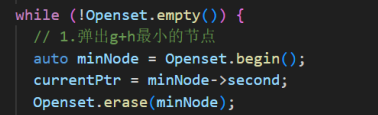
欧几里得距离：欧几里得距离是两点之间的直线距离，通常用于三维空间中的路径规划。它能够提供较为准确的距离估计，有助于 A\* 算法找到最优路径。

tie\_breaker：在路径搜索过程中，可能会遇到多个节点具有相同的启发式估计值。tie\_breaker 用于打破这种平局，防止算法在代价相同的情况下出现路径选择上的平局。通过增加一个小的偏移量，使得启发式估计值略微偏大，有助于算法更快地收敛到最优路径。

**2.2Astarpath&&Astarsearch**

这段代码实现了 A\* 算法的主循环部分，用于在搜索空间中找到从起点到终点的最优路径。A\* 算法是一种启发式搜索算法，通过结合实际代价和启发式估计来找到从起点到终点的最优路径。主循环的作用是不断从开放列表（OpenSet）中取出代价最小的节点，扩展其邻居节点，并更新邻居节点的信息，直到找到终点或开放列表为空。

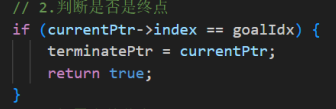
1. 弹出 g+h 最小的节点：



Openset 是一个 std::multimap<double, MappingNodePtr>，按 f\_score（即 g\_score + h\_score）排序。minNode 是 Openset 中 f\_score 最小的节点对。

currentPtr 是当前扩展的节点。从 Openset 中移除 minNode，以保证每次扩展的都是当前代价最小的节点，确保 A\* 算法的最优性。

1. 判断是否是终点：



如果当前节点是终点节点，则设置 terminatePtr 为当前节点，并返回 true，表示找到路径。提前终止搜索，节省计算。

1. 拓展当前节点：

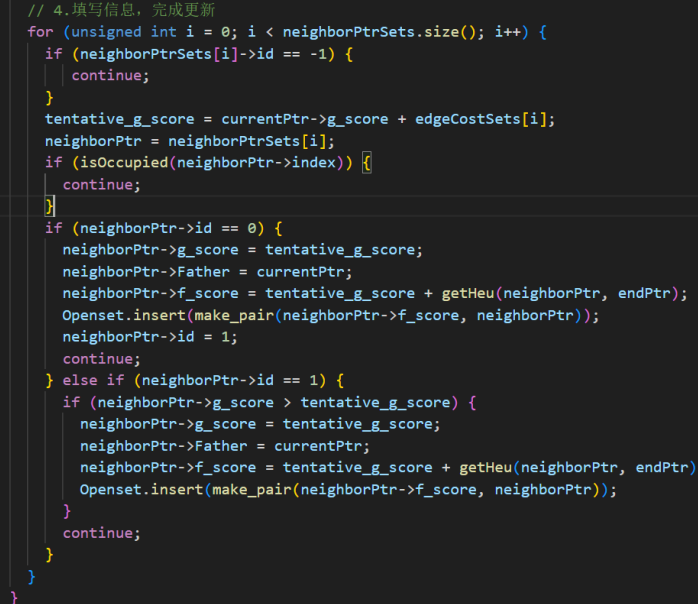


调用 AstarGetSucc 函数，获取当前节点的所有邻居节点和对应的边代价。

neighborPtrSets 存储邻居节点，edgeCostSets 存储边代价。

原因：获取当前节点的所有邻居节点，进行下一步扩展。

1. 填写信息，完成更新：

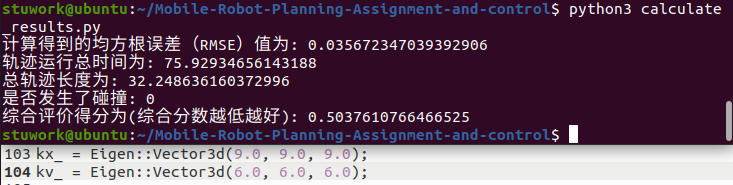


遍历所有邻居节点，更新每个邻居节点的信息。（1）如果邻居节点的 id 为 -1，表示该节点不可用，跳过。计算从当前节点到邻居节点的临时 g\_score。（2）如果邻居节点被占用，跳过。（3）如果邻居节点的 id 为 0，表示该节点未被访问过，更新邻居节点的 g\_score 和 f\_score。设置邻居节点的父节点为当前节点。将邻居节点插入 Openset。将邻居节点的 id 设置为 1，表示已在 Openset 中。（4）如果邻居节点的 id 为 1，表示该节点已在 Openset 中。（5）如果新的 g\_score 小于当前 g\_score，更新邻居节点的信息，确保每个节点的 g\_score 和 f\_score 是最优的。使用 id 标记节点的状态，避免重复访问。

**2.3总结**

这段代码实现了 A\* 算法的主循环部分，通过不断从开放列表中取出代价最小的节点，扩展其邻居节点，并更新邻居节点的信息，直到找到终点或开放列表为空。每一步都确保扩展的是当前代价最小的节点，并且更新邻居节点的信息，保证算法的最优性和效率。

**实验三：**



均方误差为：0.035672347039392906

运行时间为：75.92934656143188

轨迹长度为：32.248636160372996

是否发生了碰撞：0

综合评分：0.5037610766466525

**实验四： 无**