**张 文泰**

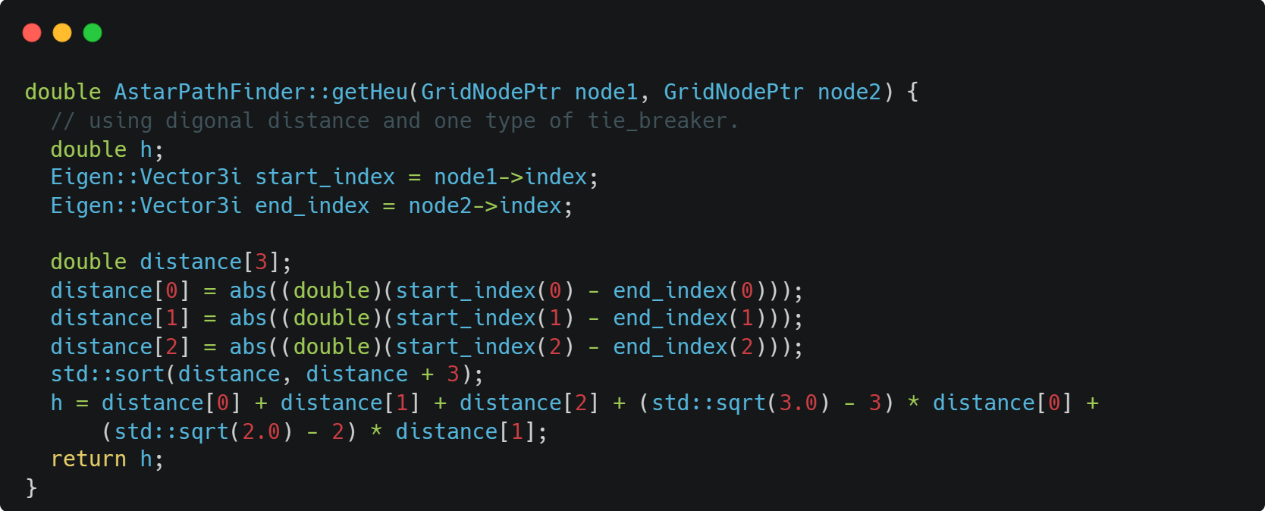
**学号：21009101463**

**机器人路径规划**

大作业

1. 程序分析

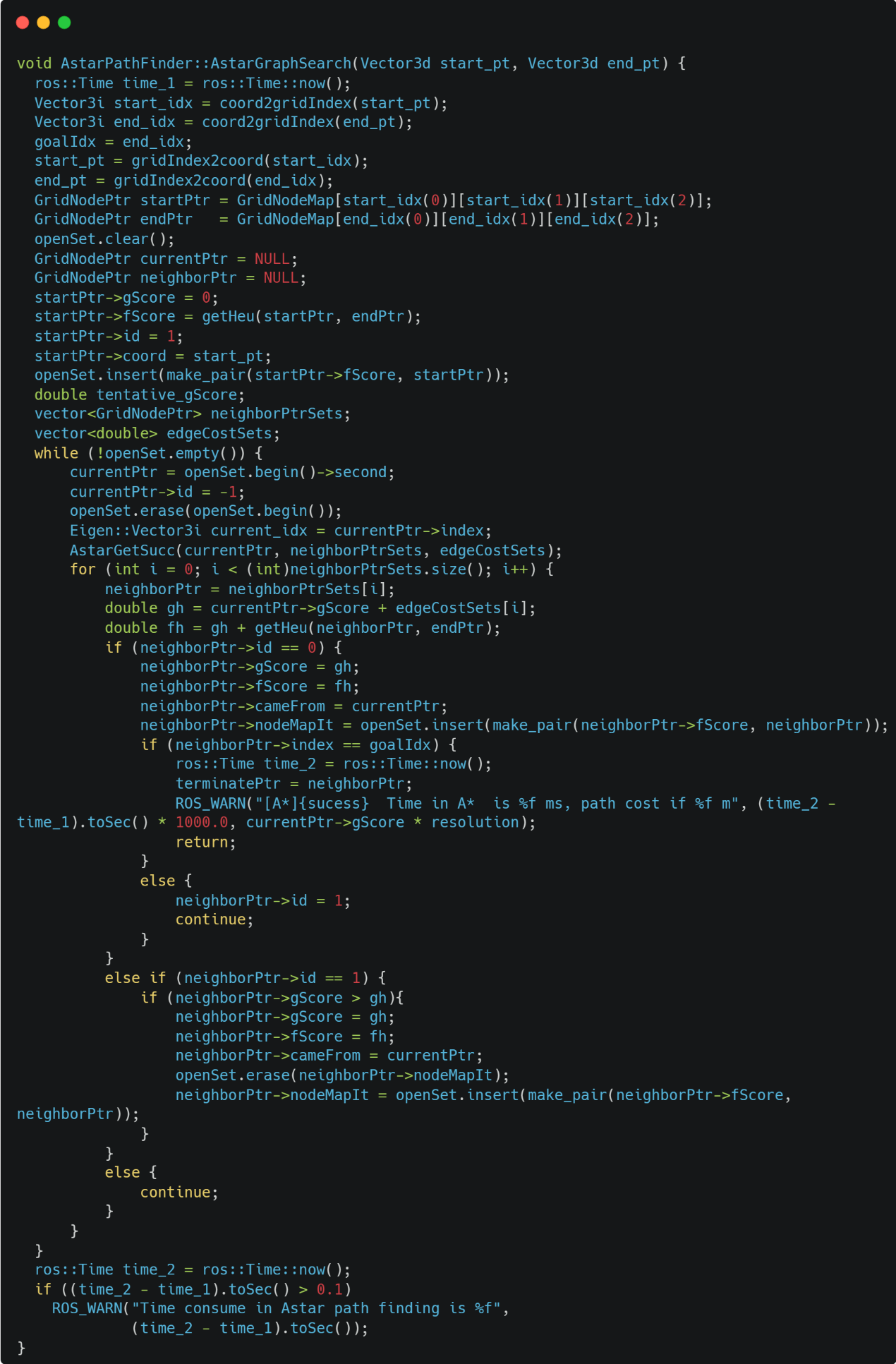
算法



getHeu函数

该函数在 算法中用于计算启发函数——用于估计从当前节点到目标节点的最短路径代价，它接收两个指向 类型对象的指针参数 和 ，下面是该函数的大致思路：

1. 获取当前节点 和目标节点 的索引，这些索引表示节点在三维空间中的位置。
2. 计算节点 和 在三个维度上的距离。这里使用了绝对差值来计算距离。
3. 对计算得到的距离进行排序，以确保最大的距离值在数组的最后。这里采用的排序方式是标准库中的 函数。
4. 将三个维度上的距离合并为一个启发式估计值 。实现方式是将三个维度上的距离相加，并且加入了一个 。 的作用是在启发式估计值相等的情况下，优先选择距离更远但是在对角线上的节点。
5. 返回计算得到的启发式估计值 。



AstarGraphSearch函数

这段代码实现了A\*算法的核心逻辑，用于在给定起点和终点的情况下，在地图上搜索最短路径。代码大致逻辑如下：

1. 记录起点和终点对应的栅格坐标，并将目标索引设置为终点的索引。然后，将起点和终点的位置重新计算（似乎是为了确保它们在栅格中的位置是准确的）。
2. 初始化起点和终点节点，这里假设它们已经在之前的地图初始化中被创建。
3. 清空待弹出点集，并定义要弹出的当前节点和邻居节点。
4. 计算起点的启发式函数值，并将其加入开集。
   1. 进入主循环，直到开集为空：
   2. 弹出开集中最大 f 值的节点，并标记为闭集。
   3. 获取当前节点的邻居节点集合。
   4. 遍历邻居节点集合：
      1. 如果邻居节点是自由节点：
      2. 计算邻居节点的 g 和 f 值，并加入开集。
      3. 如果该邻居节点是目标节点，则搜索成功，记录终点并返回。
      4. 否则，将邻居节点标记为开集。
   5. 如果邻居节点已在开集中：
      1. 如果通过当前节点到达邻居节点的路径更短，则更新邻居节点的 g 和 f 值，并重新排序开集。
      2. 如果邻居节点已在闭集中，则不做处理。
5. 如果搜索失败，记录时间消耗。

简单地说，这段代码主要实现的是A\*算法的搜索过程，其中通过开集和闭集来管理节点的状态，并在搜索过程中不断更新节点的代价值以寻找最短路径。

RDP算法简化轨迹



pathSimplify函数

这段代码实现了RDP 算法，用于简化路径。其主要逻辑是通过找到距离最大点来决定路径的主要形状，并不断递归分割和简化路径，以达到减少路径点数的目的。其主要逻辑如下：

1. 检查：如果路径中的点数少于或等于2，则路径不需要简化，直接返回原路径。
2. 提取首尾点：取出路径第一个点和最后一个点，作为简化路径的首尾点。
3. 寻找最大偏离点：遍历路径中的所有中间点，计算每个点到首尾点连线的距离，找出距离最大的点并记录其下标和距离。
4. 判断是否需要进一步简化：如果最大距离小于给定的阈值，认为路径在此范围内可以简化为一条直线，只保留首尾点作为简化路径。如果最大距离大于阈值，说明路径不能简化为一条直线，需要进一步处理。
5. 递归简化路径：将路径在最大偏离点处分成两段，对每段路径分别递归调用简化函数，简化每段子路径，将两段简化后的路径合并为一条完整的简化路径。
6. 合并结果：合并时，避免重复添加交界点，确保简化路径中的点是唯一且按顺序排列的。
7. 返回简化路径：最终返回简化后的路径，包含保留下来的关键点，达到减少路径点数的目的。

RDP算法通过递归分割路径，仅保留对整体形状影响最大的关键点，从而高效简化路径，同时保持路径的主要形状特征，具有高效、灵活、易实现和广泛适用的优点。

Minisnap轨迹优化



trajOptimization函数

这段代码实现了一个轨迹优化函数，用于确保生成的轨迹在空间中是安全的。该函数接受一个路径矩阵path，通过多项式拟合生成轨迹，并检查其安全性。如果发现不安全段，则通过细分路径并重新生成轨迹，直到所有段都安全。



timeAllocation函数

该函数用于为路径中的各段分配时间，它根据路径段长度、最大速度和最大加速度，为路径的每一段分配合适的时间。其主要优点在于能合理考虑加速、匀速和减速三个阶段，使得时间分配更为精准和高效。



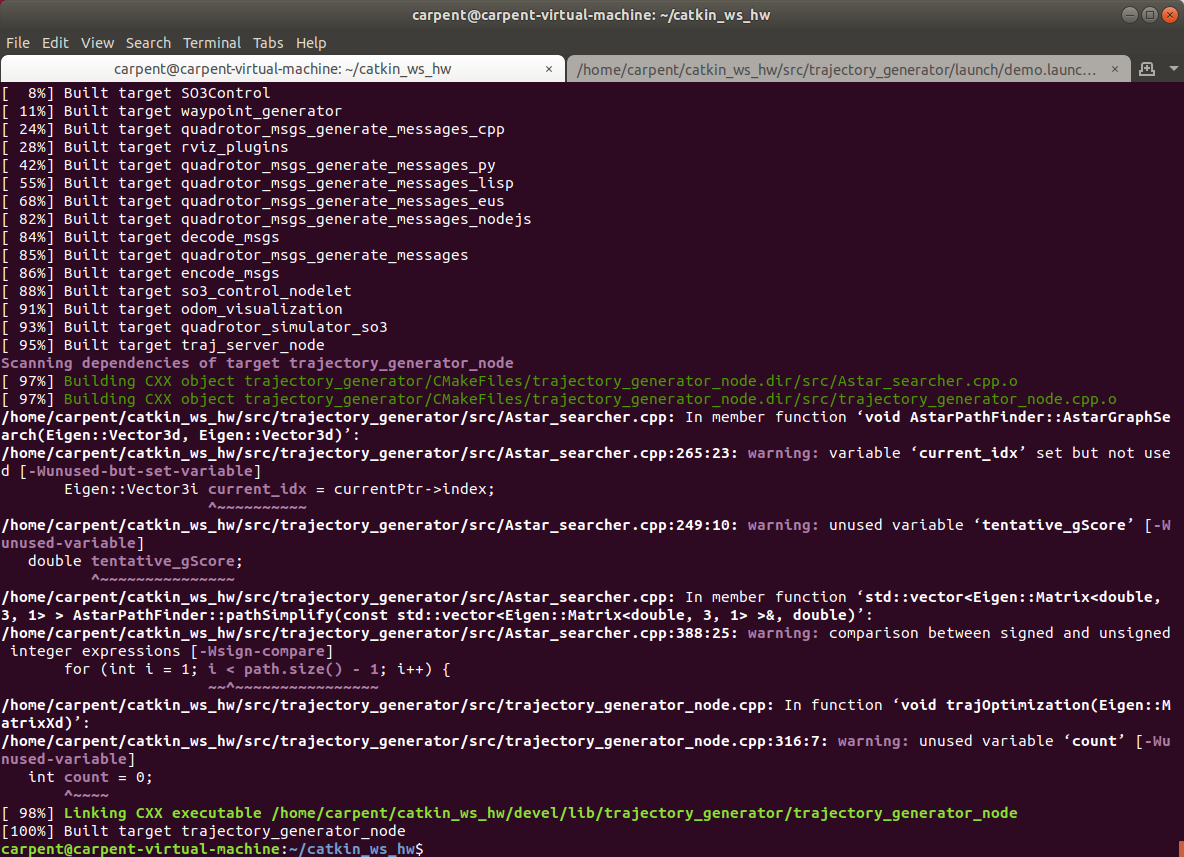
safeCheck函数

safeCheck 函数通过遍历轨迹的每一段和每段内的多个时间点，检查轨迹是否安全。如果任何位置位于障碍物中，函数会返回该段的索引，否则返回 -1 表示整个轨迹安全。这种方法保证了生成的轨迹能够避开障碍物，确保路径的安全性。函数关键点在于：

1. 时间步长：时间步长 0.01 是检查轨迹的精细程度。
2. 轨迹位置计算：getPosPoly 函数通过多项式系数计算出当前时间的轨迹位置。
3. 位置检查：isOccupied 函数结合 coord2gridIndex 转换，将轨迹位置映射到网格上，并检查该位置是否有障碍物。

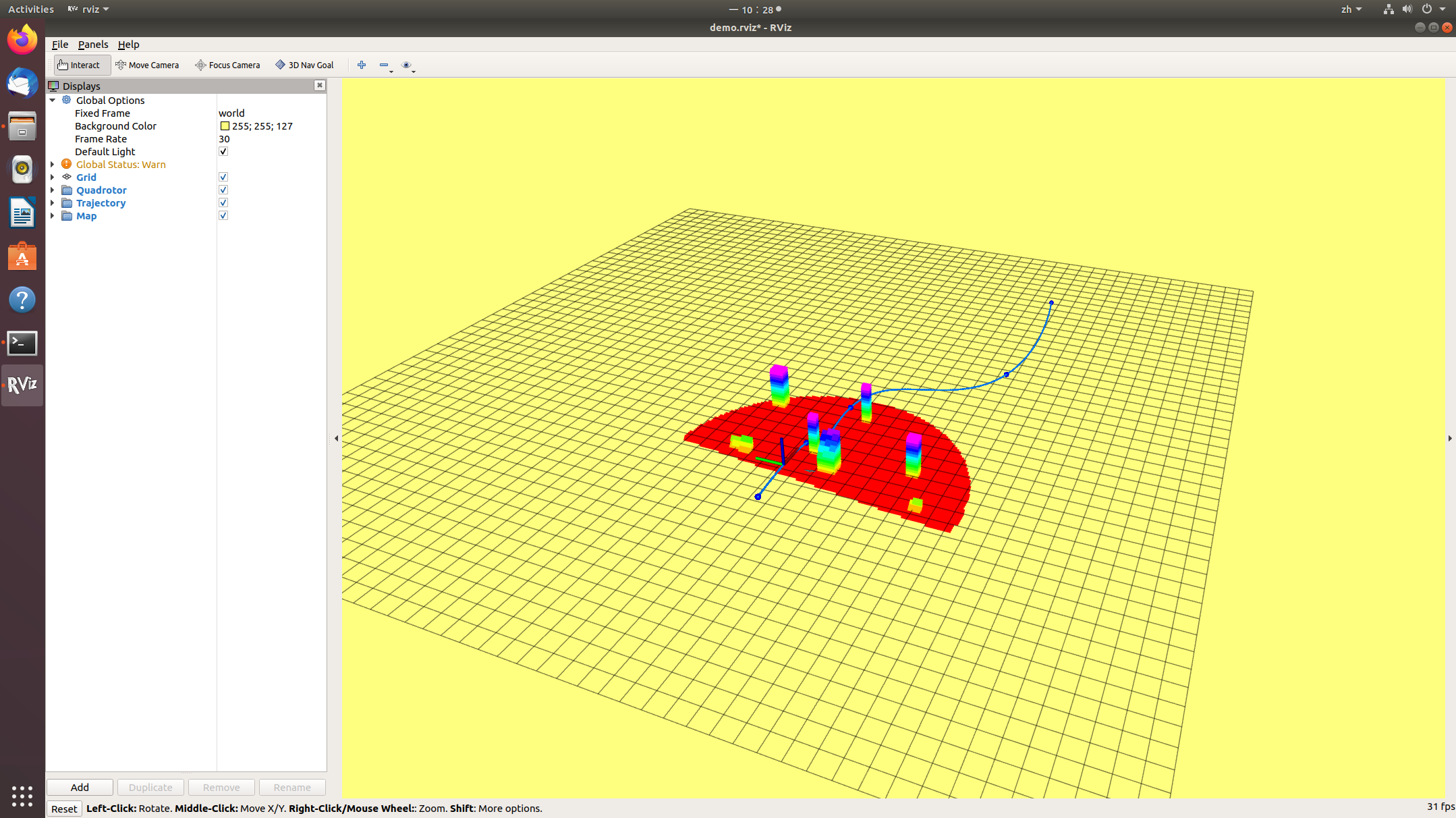
Minisnap轨迹优化通过最小化加速度的高次导数来实现路径的平滑化和优化，确保机器人的运动平稳且安全，其优点包括生成的路径平滑度高、加速度变化小、适应性强等。

1. 实验结果

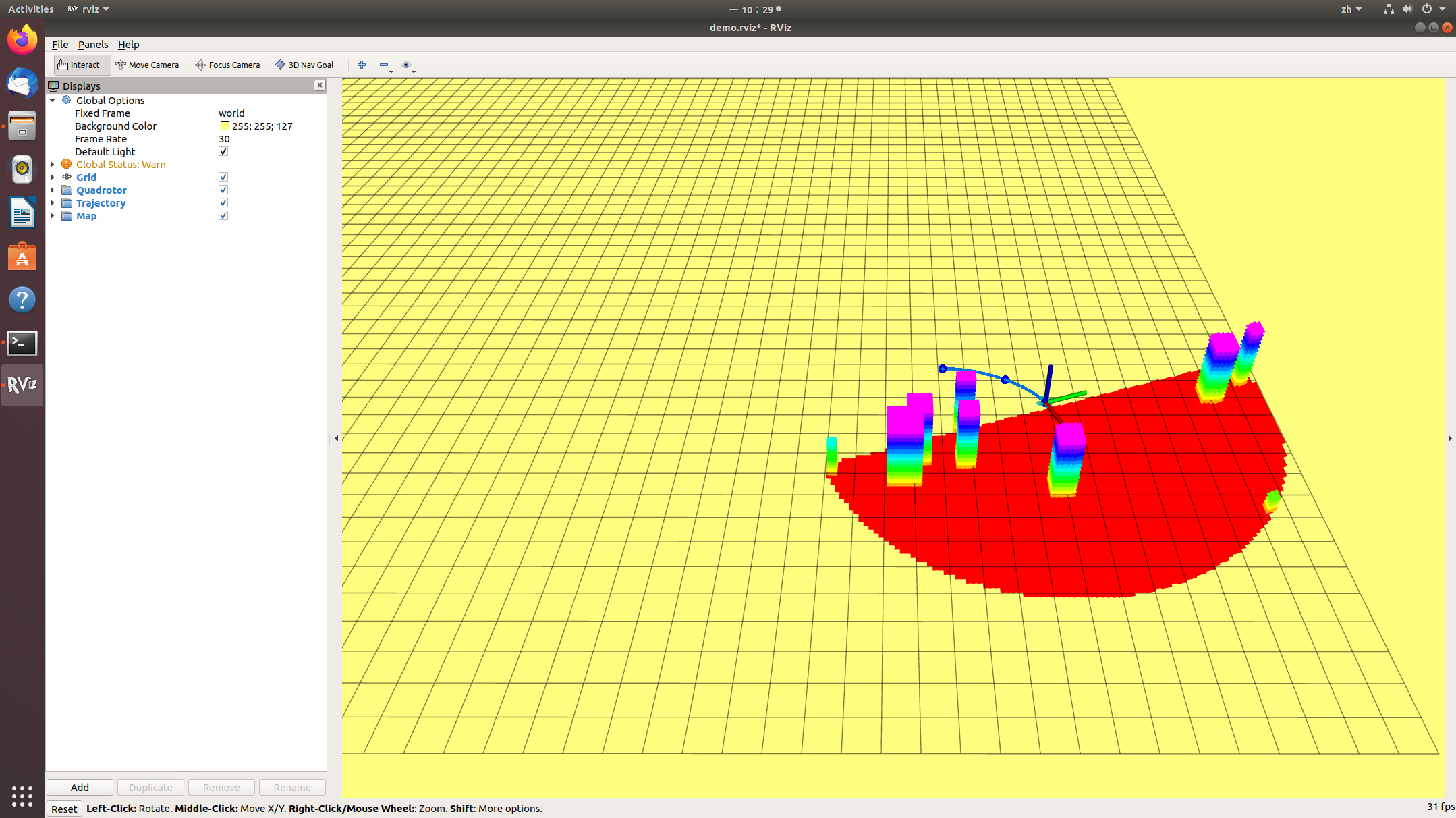


程序运行成功

如图所示，程序成功运行，地图显示在RViz中。接下来使用3 nav goal工具在图中选取终点，运行结果如图所示：

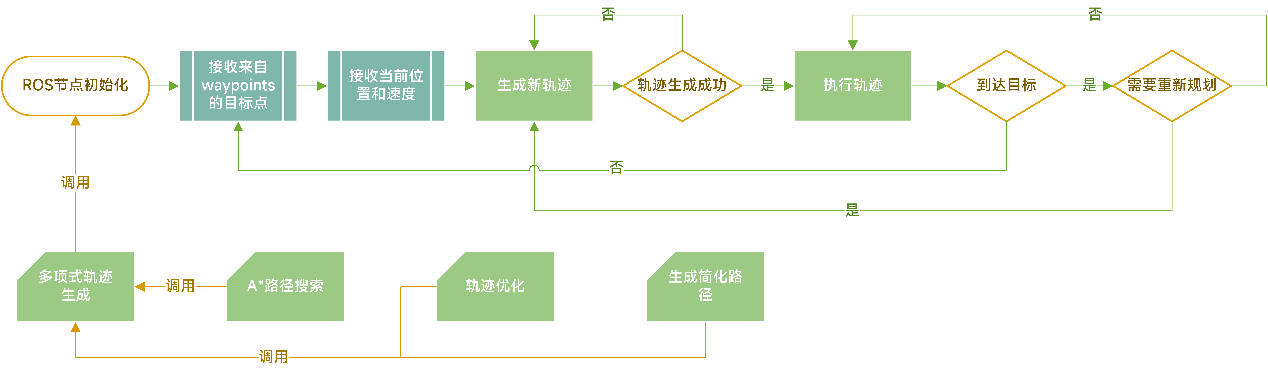


程序开始运行



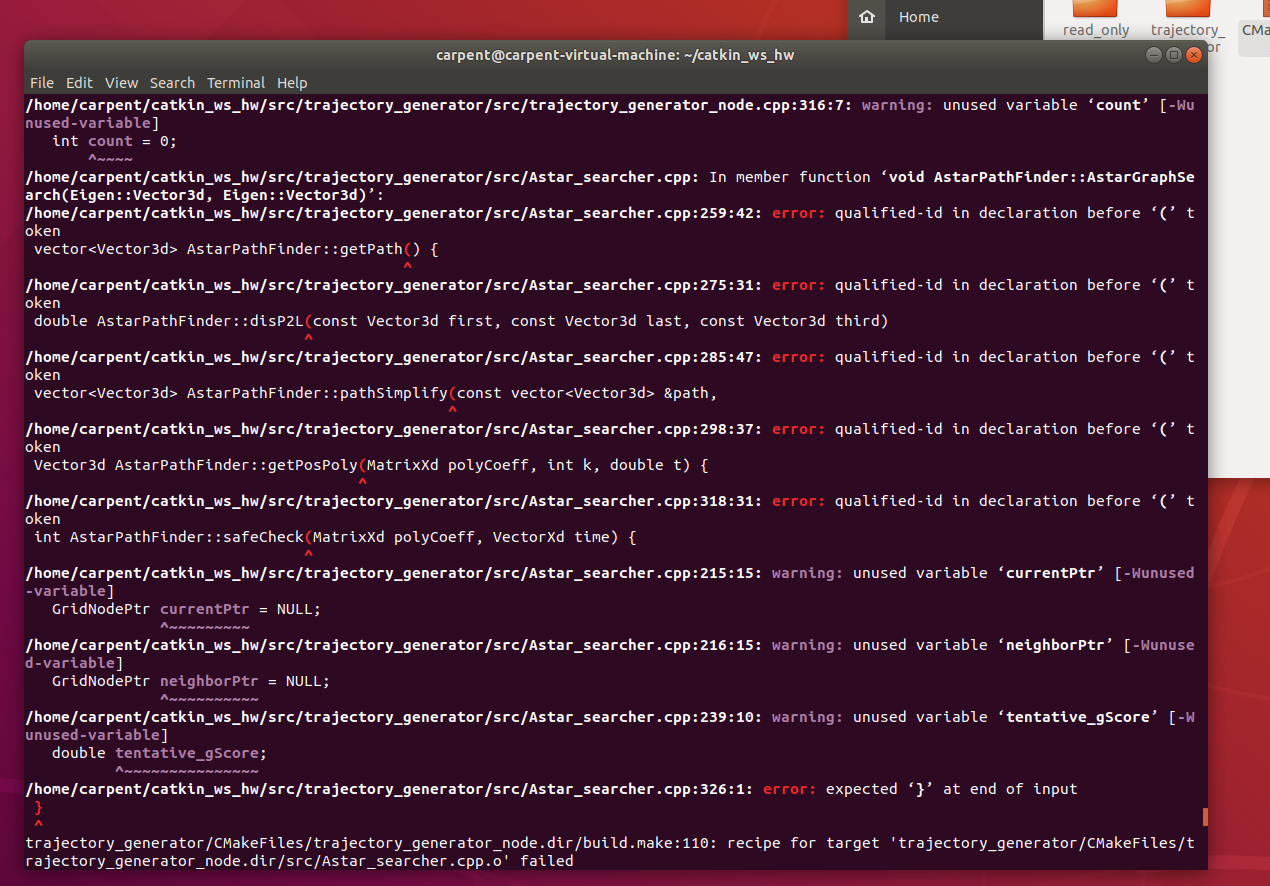
轨迹运行结果

下面是本实验中的程序运行流程图：



流程图

1. **实验中遇到的问题及解决方案**



报错现象

如图所示，我在第一次编译工作空间时出现了报错，阅读报错信息之后意识到，是由未补全代码导致，在完成代码补全之后解决了问题。

1. **实验感悟与课程总结**

在本学期的课程学习以及完成大作业的过程中，我收获了许多宝贵的经验和感悟，不仅学习了课内专业知识，还深刻理解了路径规划在机器人、无人机等领域的重要性。通过实践，我掌握了A\*算法、Minisnap轨迹优化等技术，加深了对这些技术原理和实现方法的理解。

不仅如此，几次课程作业的学习与锻炼，加深了我对C++、MATLAB编程语言的掌握程度，锻炼了自己的问题解决能力和编程技能，并可以使用它们实现路径规划算法，将其应用于实际场景中。除此之外，还包括对于虚拟机的使用以及Ubuntu系统的认识，我之前只用VMware来搭建饥荒联机版的云服务器，这次课程作业让我对虚拟机的认识更深一层，同时还让我认识了此前从未接触过的Ubuntu操作系统，极大地拓宽了我的视野。

最重要的是，通过这次大作业，我意识到路径规划不仅是一项技术，更是一种思维方式，需要综合考虑问题的复杂性和多样性，不断探索和创新。因此，我将继续学习和探索路径规划领域的知识，努力提升自己的能力，为未来的科技创新和应用做出更多贡献。

最后要说的是，非常感谢常晶老师与智斌亮老师在本学期的课程学习过程中对我的指导与帮助，我的成长得益于两位老师的悉心付出，对两位老师致以衷心的感谢。