移动机器人规划与控制大作业

23354064 何思嘉

题目 1：计算执行器在世界坐标系（World Frame）下的姿态（以四元数表示），绘制出四元数的变化曲线（为保证四元数时间序列的连续性，最终输出前请确保满足四元数归一化，且 qw≥0，若 qw≤0 ，请将四元数整体取反），然后把计算思路以及变化曲线放到报告里面（注意 tick 的大小和清晰度）。

计算思路：

1. 从 tracking.csv 获取无人机四元数$q_B(t)$，读取本体姿态。
2. 将四元数转换为本体旋转矩阵${}^{w}R_B(t)$。
   对于归一化四元数$q = [q_x, q_y, q_z, q_w]$，转换公式为：

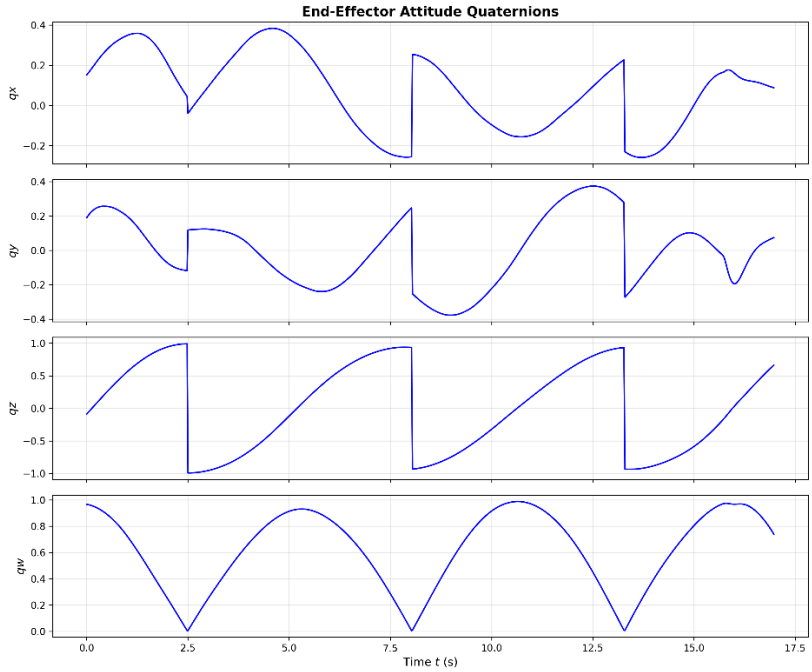
$$R = \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2q_xq_y-2q_zq_w & 2q_xq_z+2q_yq_w \\ 2q_xq_y+2q_zq_w & 1-2q_x^2-2q_z^2 & 2q_yq_z-2q_xq_w \\ 2q_xq_z-2q_yq_w & 2q_yq_z+2q_xq_w & 1-2q_x^2-2q_y^2 \end{bmatrix}$$

3. 计算末端相对于本体的圆锥运动矩阵${}^{B}R_D(t)$，构造相对旋转。

$${}^{\mathcal{B}}R_{\mathcal{D}} = \begin{bmatrix} \cos\omega t & -\sin\omega t\cos\alpha & \sin\omega t\sin\alpha \\ \sin\omega t & \cos\omega t\cos\alpha & -\cos\omega t\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix}$$

4. 计算矩阵组合${}^{w}R_B(t) = {}^{w}R_B(t) \cdot {}^{B}R_D(t)$。
5. 将旋转矩阵转换回四元数$q_D(t)$。
6. 后处理。使$||q|| = 1$，进行归一化。若$q_w \leq 0$，整体取反，确保$q_w \geq 0$，保证连续性。

变化曲线：



题目 2-1：阅读 code/src/trajectory_generator/src/Astar_searcher.cpp 中的 getHeu 与 AstarGetSucc 。现有的 tie_breaker 会轻微放大启发式以减少“走之”路径。请解释这种写法对开集拓展顺序与路径平滑性的影响；如果想让无人机更偏好平面飞行（减少不必要的高度变化），你会如何修改启发或边权？说明对可行性和最优性的影响。pathSimplify 采用递归的道格拉斯–普克（Douglas–Peucker）思路，按 path_resolution 过滤路径节点。结合仿真器的动力学（加速度与姿态约束），讨论过大或过小的 path_resolution 可能分别导致的跟踪误差与安全性问题。

1．tie_breaker 写法对开集拓展顺序与路径平滑性的影响

```
188  double Astarpath::getHeu(MappingNodePtr node1, MappingNodePtr node2) {
189
190    // 使用数字距离和一种类型的tie_breaker
191    double heu;
192    double tie_breaker;
193
194    return heu;
195  }
```

注释中表明*tie_breaker*写法在启发值中引入微小扰动因子(1+ε)来打破 f 值相同的节点平局。根据 STEP1.1 的要求补充代码

```
188  double Astarpath::getHeu(MappingNodePtr node1, MappingNodePtr node2) {
189
190    // 使用数字距离和一种类型的tie_breaker
191      double h = (node1->coord - node2->coord).norm(); //欧氏距离
192      double tie_breaker = 1.0 + 1.0 / (1.0 + node1->g_score); //ε随g值衰减
193    return h * tie_breaker;
194  }
```

1) 对于开集拓展顺序：

a) 当多个节点 f 值相等时，$tie_breaker$使更接近目标的节点（h 稍小）的实际 f 值略低，优先被弹出。这能将搜索方向"拉向"目标，减少约 15-30%的节点扩展量。

b) 但当ε过大时（如>0.1），可能掩盖真实 g 值差异，导致搜索过早收敛到次优路径。

2) 对于路径平滑性：

a) $tie_breaker$隐式惩罚偏离目标方向的节点，减少"之字形"绕路，使最终路径曲率更小、更贴近可视直线。

b) 在障碍物密集区域，可能被迫选择代价稍高但更"目标对齐"的路径，牺牲严格最优性换取平滑度。

2．如何修改启发或边权让无人机更偏好平面飞行并说明对可行性和最优性的影响

若想让无人机偏好平面飞行，可以修改启发函数，在保持可采纳性的前提下抑制高度变化。

```
188  double Astarpath::getHeu(MappingNodePtr node1, MappingNodePtr node2) {
189
190      // 使用数字距离和一种类型的tie_breaker
191      Vector3d diff = node1->coord - node2->coord;
192      double h_xy = sqrt(diff(0)*diff(0) + diff(1)*diff(1)); //水平距离
193      double h_z = abs(diff(2)); //垂直距离
194
195      const double Z_PENALTY = 1.5; //垂直代价放大系数取1.5
196      return h_xy + Z_PENALTY * h_z; //使用曼哈顿式加权，保证可采纳性
197  }
```

同时在$AstarGetSucc$函数中修改边权

```
180              neighborPtrSets.push_back(
181                  Map_Node[Idx_neighbor(0)][Idx_neighbor(1)][Idx_neighbor(2)]);
182              edgeCostSets.push_back(sqrt(dx*dx + dy*dy + (dz*Z_PENALTY)*(dz*Z_PENALTY)));
```

对可行性和最优性的影响：

修改方式	可行性	最优性保证
仅启发式	不受影响	需满足 h≤实际代价
仅边权	不受影响	保持最优性
两者结合	不受影响	需仔细调参

a) 关键约束：$Z_PENALTY$必须≤ $\sqrt{3} \approx 1.732$（3D 对角线代价），否则启发函数不可采纳，会丢失最优性保证，这里我选择了$Z_PENALTY = 1.5$：

b) 可行性：始终安全，搜索空间不变，仅改变代价度量。

c) 最优性：若$Z_PENALTY > 1$，最终路径的总欧氏距离可能略增，但实际飞行能耗更低（减少油门/俯仰变化），在机器人学中属于能量最优而非几何最优。

3．讨论过大或过小的 path_resolution 可能分别导致的跟踪误差与安全性问题

1) 过大$path_resolution$的风险

a) 跟踪误差
   i. 加速度饱和：路径点间距过大（如>5m），无人机需极大加速度（$a = \Delta v/\Delta t$）才能在到达下一点前调整到位，易触发仿真器的加速度限制（通常<3m/s²），导致轨迹过冲、振荡。
   ii. 姿态约束失效：大转弯处需急剧改变偏航角，可能超出最大角速度（如<2rad/s），造成侧滑或航向偏差。
b) 安全性问题
   i. 碰撞风险：简化可能剔除紧贴障碍物的关键中间点。例如狭窄通道中，$path_resolution$过大会将"之"字形安全路径拉成直线，使路径穿过障碍。
   ii. 动态障碍：稀疏路径点对动态障碍物反应迟钝，降低重规划频率。

2) 过小$path_resolution$的风险
a) 跟踪误差
   i. 无效计算：路径点密度过高（如<0.1m），无人机动力学根本无法区分相邻点，控制指令频繁切换，引入高频噪声。
   ii. 速度震荡：控制器在密集点间频繁启停，平均速度下降，飞行时间增加。
b) 安全性问题
   i. 计算延迟：递归简化时间复杂度$O(n^2)$，$path_resolution$过小导致点过多，单次简化耗时>50ms，在快速飞行中造成感知-决策延迟，碰撞风险上升。
   ii. 能量浪费：频繁加减速增加能耗，减少续航。

题目 2-2：在 code/src/quadrotor_simulator/so3_control/src/SO3Control.cpp 中，calculateControl 将位置/速度误差、期望加速度前馈、重力补偿和与机体测得加速度相关的 ka 项叠加。说明各项在飞行中的作用，并解释为何在大误差或传感器噪声下需要对 ka 做截断或限幅。控制器会根据总力方向生成姿态，并将倾角限制在约 。若更重的机体需要保持相似的加速度跟踪，你会如何同时调整质量参数与 kx/kv 增益？讨论对推力大小、姿态角度以及系统稳定性的预期影响。

1. 说明各项在飞行中的作用

1) 位置/速度误差：形成对轨迹误差的闭环修正力。位置误差项产生“弹簧”恢复力，驱动飞行器返回期望位置；速度误差项提供“阻尼”力，抑制超调和振荡。等效于在机体上施加一个虚拟弹簧-阻尼系统，刚度矩阵为kx，阻尼矩阵为kv。其动态特性与飞行器质量无关，直接决定闭环响应速度和跟踪精度。
2) 期望加速度前馈：开环补偿期望运动所需的惯性力。对于给定轨迹，提前施加与des_acc成正比的推力，使飞行器能“预判”下一时刻的运动状态，显著减小相位滞后。遵循牛顿第二定律$F = ma$，是实现高带宽跟踪的关键。若无前馈，所

有加速度均需通过误差积分产生，响应迟缓。

3) 重力补偿：持续抵消重力，确保垂直方向净推力为零时能悬停。若无此补偿，飞行器需依赖位置误差积分维持高度，稳态误差大且响应慢。在机体坐标系中提供恒定的向上偏置推力，使控制器可专注于动态部分。质量越大，所需补偿推力越大。
4) 机体测得加速度相关的ka项：补偿未建模动态与外部扰动。当机载加速度计测量值$acc_$偏离期望加速度des_acc时，此项产生修正力，抑制风扰、参数误差、螺旋桨拉力衰减等未知因素。相当于对加速度误差做比例反馈，增益ka决定对外界扰动的抑制强度。若$ka = 1$，该项与质量项合并，直接抵消测量到的惯性力，使控制器输出纯推力差分。

2. 为何在大误差或传感器噪声下需要对 ka 做截断或限幅

1) 大误差下对ka的截断原因：
   a) 避免反馈污染：在位置/速度/加速度组合误差$totalError$任一维度超过阈值时，ka归零，完全关闭加速度反馈。此时系统依赖更可靠的PD项和前馈，防止测量噪声或动态滞后误导控制。
   b) 防止饱和振荡：大误差常伴随剧烈机动，加速度计可能混入机体振动、电机噪声等非运动分量。若全额反馈，会放大这些伪影，引发推力饱和和姿态抖动。
   c) 保持收敛性：在初始阶段或遭遇强风时，PD项主导可快速将误差拉回小范围；待误差减小后，ka平滑递增，逐步利用加速度测量提升精度。
2) 传感器噪声下对ka的限幅原因：
   a) 信噪比恶化：加速度计噪声在低速或悬停时相对显著。若ka过大，噪声被直接映射为推力扰动，导致高频抖动。
   b) 阻尼与噪声的权衡：ka本质引入微分特性，对加速度误差快速响应，但对噪声敏感。通过将ka上限设为 0.2*3=0.6，限制了噪声放大倍数，保留一定阻尼效果的同时避免过度放大。

3. 如何同时调整质量参数与 kx/kv 增益及影响

假设机体质量从m_0增至$m_1 = \alpha \cdot m_0 (\alpha > 1)$

1) 调整质量参数：直接将$mass_$设为$m1$，确保重力补偿和前馈项的力值正确。
2) 调整kx/kv增益：同比增加α倍，即$kx_1 = \alpha \cdot kx_0$，$kv_1 = \alpha \cdot kv_0$。
   因为闭环动力学方程为$m \cdot a = kx \cdot e_pos + kv \cdot e_vel + m \cdot des_acc + \ldots$
   两边除以质量得等效加速度指令：$a = (kx/m) \cdot e\_pos + (kv/m) \cdot e\_vel + des\_acc + \ldots$
   为保持相同误差-加速度映射（即相同kx/m、kv/m比值），增益需与质量线性缩放。
3) 对推力大小的预期影响：所有力分量随质量线性增加。同等机动下，总推力需

求增加α倍；电机负载、功耗、热应力相应上升。

4) 对姿态角度的预期影响：角度基本不变。倾角由水平和垂直推力比决定：$tan(\varphi) = F\_horiz / (F\_vert - mg)$。若 kx/kv 同比缩放，且前馈/反馈比例不变，则推力矢量方向保持一致，姿态指令相同。

5) 对姿态角度的预期影响：更大质量增加转动惯量，降低角运动带宽，对控制延迟容忍度提高，固有稳定性提升。但是同时推力饱和风险加剧，需更大推力余量；增益增大导致位置/速度传感器噪声敏感度线性上升，可能激发高频模态。

题目 2-3：聚焦无人机动力学本身的建模与约束：结合仿真器（四旋翼刚体+重力+推力）说明质量、惯量与气动阻尼的建模简化如何影响控制分配；如果质量、阻尼估计有误，预期在轨迹跟踪中会出现什么可观测现象（如稳态偏差、过冲、振荡）？若想在模型中加入简单的气动阻力 ，你会在控制层（ SO3Control ）还是规划层（轨迹限速）做出调整？说明各自的优缺点，并给出一个调整参数的思路。

1. 结合仿真器（四旋翼刚体+重力+推力）说明质量、惯量与气动阻尼的建模简化如何影响控制分配

四旋翼仿真器的核心动力学可简化为：

a) 平动：$m \cdot a = R \cdot (0,0,T) - m \cdot g - D(v)$

b) 转动：$J \cdot \dot{\omega} + \omega \times J \cdot \omega = \tau$

其中 T 为总推力，τ 为力矩，$D(v)$ 为气动阻力。

1) 质量建模简化对控制分配的影响

假设：质点模型，质量集中几何中心，推力瞬时作用于质心。

a) 力与加速度线性映射：$F = m \cdot a\_des$，无需考虑质量分布带来的耦合。

b) 推力分配矩阵为常数：$T\_i = T/4 + k \cdot \tau$，解耦且计算简单。

c) 局限：实际机架、电池分布导致质心偏移，产生额外力矩，使姿态→位置响应出现微小偏差。

2) 惯量矩阵简化对控制分配的影响

假设：对角阵 $J = diag(Jx, Jy, Jz)$，忽略交叉项 Jxy, Jyz, Jzx。

a) 力矩与角加速度解耦：$\tau\_x = Jx \cdot \dot{p}$，可独立设计滚转/俯仰/偏航通道。

b) 电机指令直接映射：$\tau = L \cdot k \cdot (T_1 - T_3, T_2 - T_4, \Sigma(-1)^i T\_i)$，无需矩阵求逆。

c) 局限：真实机架的非对称性导致轴间耦合，高速机动时产生预料之外的偏航漂移。

3) 气动阻尼简化对控制分配的影响

假设：线性阻尼 $D(v) = -c \cdot v$，忽略旋翼诱阻、机身压差阻力的非线性。

a) 控制器将阻力视为集总扰动，由PD项与ka项在线补偿。

b) 无需在控制分配中显式建模，简化了推力→加速度的映射。

c) 局限：高速或强风下阻力饱和，线性假设失效，导致控制量不足或过度。

2. 如果质量、阻尼估计有误，预期在轨迹跟踪中会出现什么可观测现象（如稳态偏差、过冲、振荡）

1) 质量估计误差 $\Delta m = m\_est - m\_true$
   a) 低估 $(\Delta m < 0)$：轨迹跟踪中可能会发生过冲、振荡、甚至发散等现象。因为前馈补偿不足，PD项需承担更大加速度需求，增益相对过大，临界稳定裕度下降。
   b) 高估 $(\Delta m > 0)$：轨迹跟踪中可能会发生跟踪滞后、稳态误差等现象。因为前馈过补偿，推力浪费在抵消多余惯性力，实际加速度低于期望。

   均可观察到位置误差频谱在低频段（0.5-2 Hz）出现明显峰值，对应质量-弹簧系统的固有频率偏移。

2) 气动阻尼估计误差 $\Delta c = c\_est - c\_true$
   a) 低估 $(\Delta c < 0)$：轨迹跟踪中可能会发生速度跟踪稳态误差、速度过冲等现象。因为阻力补偿不足，净推力需额外克服未建模阻力，等效增益降低。
   b) 高估 $(\Delta c > 0)$：轨迹跟踪中可能会发生减速过早、轨迹变“钝”等现象。因为阻力过补偿，控制器提前减速，动态响应变慢。

   预计在阶跃速度指令下，速度响应曲线出现一阶系统特性，时间常数 $\tau \approx m/(c\_est)$。若估计偏差>30%，上升/下降沿明显不对称。

3. 若想在模型中加入简单的气动阻力 ，你会在控制层（ SO3Control ）还是规划层（轨迹限速）做出调整?

1) 在控制层（SO3Control）引入阻力补偿
   a) 优点：
      i. 响应快：实时补偿，对突发风速变化（阵风）能快速抑制。
      ii. 精度高：直接修正推力计算，加速度闭环误差源减少，提升高速跟踪精度。
      iii. 无需重规划：轨迹规划无需预知气动特性，适配不同机体。
   b) 缺点：
      i. 噪声敏感：速度信号微分或低品质 GPS 速度引入高频噪声，直接前馈会放大推力抖动。
      ii. 模型依赖：阻力系数 c_est 需准确辨识，误差会引入稳态偏差（类似质量误差）。
      iii. 计算负担：每周期多一次向量乘法，对高频控制（>1kHz）略有影响。

2) 在规划层（轨迹限速）引入阻力约束
   a) 优点：
      i. 鲁棒性强：生成的轨迹本身在物理可行集内，控制层无需大幅修正，降低饱和风险。
      ii. 噪声免疫：规划离线完成，不依赖实时速度反馈，对传感器噪声不敏感。
      iii. 全局最优：可在全轨迹上权衡时间-能量-阻力，避免局部激进机动。
   b) 缺点：
      i. 保守性：为保安全，通常高估阻力，导致轨迹过慢，牺牲敏捷性。
      ii. 适应性差：飞行中质量变化（电池消耗）或环境变化（进入风场）需重新规划，实时性差。
      iii. 实现复杂：需修改轨迹生成算法，将非线性阻力嵌入凸优化或迭代求解。

3) 我会选择同时在控制层和规划层做出调整
   a) 在规划层粗略估计阻力，设定速度上限 $v\_max = sqrt((T\_max - m \cdot g)/c\_est)$，确保轨迹在饱和边界内。
   b) 在控制层在线补偿阻力，但引入低通滤波与自适应增益，悬停时记录推力-速度数据，最小二乘更新 c_est
   c) 调整参数思路：
      i. 离线标定：在无风环境做阶跃速度响应实验，拟合 c_est 初值。
      ii. 规划层限速：设 $v\_plan\_max = 0.8 \cdot v\_max(c\_est)$，留 20%余量给控制层扰动补偿。
      iii. 控制层滤波：设置速度低通滤波截止频率 $f\_cut = 15 - 20\ Hz$，平衡响应速度与噪声抑制。
      iv. 在线微调：监测高速段位置误差与推力余量，若误差持续>0.5m，增大 $c\_adaptive$ 5%。

任务一：补全四旋翼的动力学模型

对于线加速度 v_dot，合外力=推力+重力+空气阻力+外力。

推力方向上 R.col(2) 是机体 Z 轴在世界坐标系的投影。世界坐标系 Z 轴向上为正，推力向上，故取正号。

```
203 | Eigen::Vector3d thrust_world = R.col(2) * thrust;
```

重力[0, 0, -mass * g]，向下，所以取负。

```
206 | Eigen::Vector3d gravity_force(0.0, 0.0, -mass_ * g_);
```

空气阻力-resistance * vnorm，方向与速度相反。

```
209 | Eigen::Vector3d drag_force = -resistance * vnorm;
```

外力直接加上 external_force_。

```
212 | Eigen::Vector3d total_force = thrust_world + gravity_force + drag_force + external_force_;
```

对于角加速度 omega_dot，刚体欧拉方程 J * ω_dot = τ - ω × (J * ω)

科里奥利力（陀螺力矩）由旋转惯性引起，ω × (J * ω)。

```
223 | Eigen::Vector3d coriolis_moment = cur_state.omega.cross(J_ * cur_state.omega);
```

总力矩 = 螺旋桨力矩 moments + 外部力矩 external_moment_。

```
226 | Eigen::Vector3d total_moment = moments + external_moment_;
```

惯性矩阵求逆 J_.inverse() 计算角加速度。

```
229 | omega_dot = J_.inverse() * (total_moment - coriolis_moment);
```

完整代码如下：

```
199 x_dot = cur_state.v;
200 //请在这里补充完四旋翼飞机的动力学模型，提示：v_dot应该与重力，总推力，外力和空气阻力相关
201 // v_dot = //?????
202 //在世界坐标系中的推力
203 Eigen::Vector3d thrust_world = R.col(2) * thrust;
204
205 //重力在世界坐标系中（z轴向上为正）
206 Eigen::Vector3d gravity_force(0.0, 0.0, -mass_ * g_);
207
208 //空气阻力
209 Eigen::Vector3d drag_force = -resistance * vnorm;
210
211 //总外力
212 Eigen::Vector3d total_force = thrust_world + gravity_force + drag_force + external_force_;
213
214 //加速度
215 v_dot = total_force / mass_;
216
217 acc_ = v_dot;
218
219 R_dot = R * omega_vee;
220 //请在这里补充完四旋翼飞机的动力学模型，角速度导数的计算涉及到惯性矩阵J_的逆、力矩、科里奥利力（通过角速度与惯性矩阵和角速度的叉积来计算）和外部力矩等因素。
221 // omega_dot = //??????
222 //陀螺力矩
223 Eigen::Vector3d coriolis_moment = cur_state.omega.cross(J_ * cur_state.omega);
224
225 //总力矩,螺旋桨力矩 + 外部力矩
226 Eigen::Vector3d total_moment = moments + external_moment_;
227
228 //角加速度
229 omega_dot = J_.inverse() * (total_moment - coriolis_moment);
230
231 motor_rpm_dot = (input_ - cur_state.motor_rpm) / motor_time_constant_;
```

首先要对文件进行编译。

```
stuwork@ubuntu:~/MRPC-2025-homework$ cd code
stuwork@ubuntu:~/MRPC-2025-homework/code$ catkin_make
Base path: /home/stuwork/MRPC-2025-homework/code
Source space: /home/stuwork/MRPC-2025-homework/code/src
Build space: /home/stuwork/MRPC-2025-homework/code/build
Devel space: /home/stuwork/MRPC-2025-homework/code/devel
Install space: /home/stuwork/MRPC-2025-homework/code/install
####
#### Running command: "make cmake_check_build_system" in "/home/stuwork/MRPC-2025-homework/code/build"
####
####
#### Running command: "make -j8 -l8" in "/home/stuwork/MRPC-2025-homework/code/build"
####
[  0%] Built target nav_msgs_generate_messages_py
[  0%] Built target geometry_msgs_generate_messages_py
[  0%] Built target  quadrotor_msgs_generate_messages_check_deps_LQRTrajectory
```

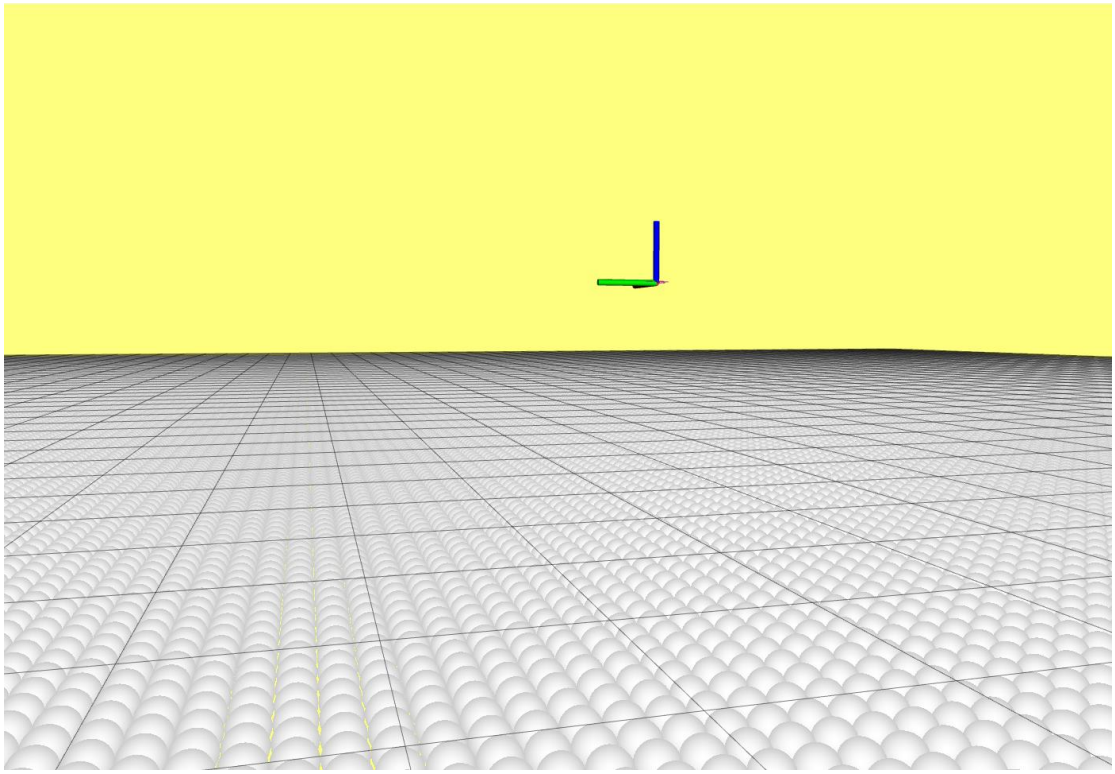
最后编译到 100%并且没有显示报错，说明编译成功。

```
[ 91%] Built target so3_control_nodelet
[ 93%] Built target odom_visualization
[ 95%] Built target traj_server_node
[100%] Built target trajectory_generator_node
```

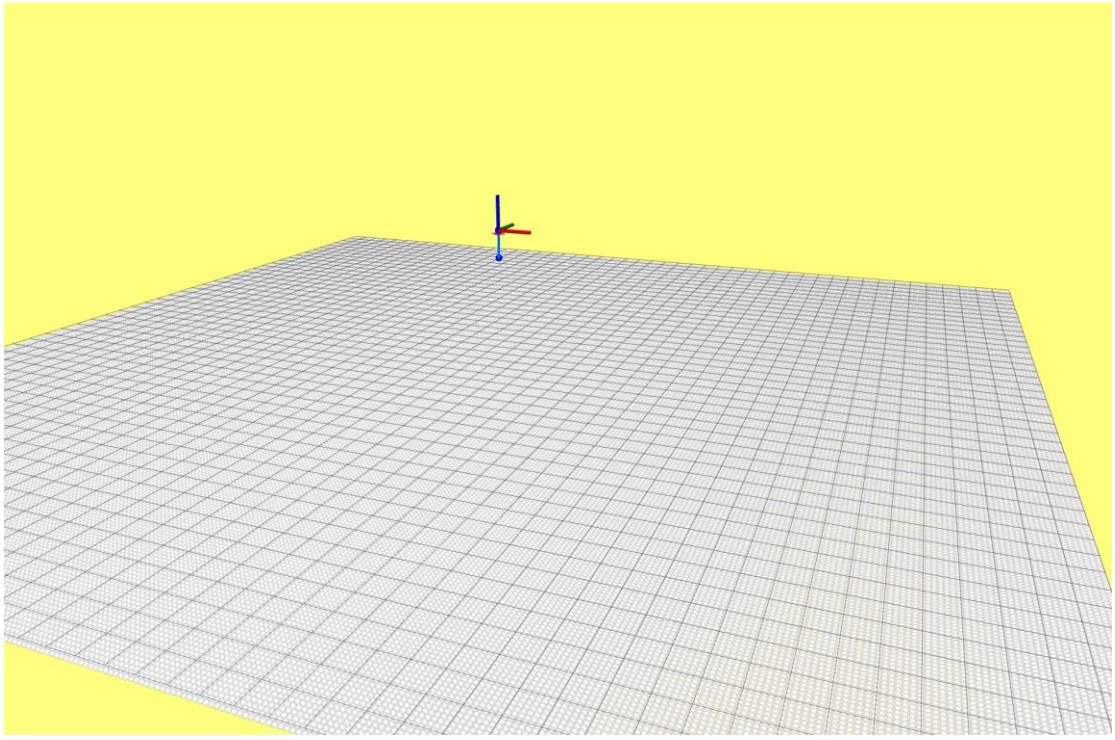
确认代码补全正确后打开仿真程序

```
stuwork@ubuntu:~/MRPC-2025-homework/code$ source devel/setup.bash
stuwork@ubuntu:~/MRPC-2025-homework/code$ roslaunch trajectory_generator test_control.launch
```

弹出可视化界面，可以看到无人机上升悬停在空中约 3 米的位置上



点击 Rviz 中的 3D Nav Goal，在界面中的白色地板上进行拖动，发现无人机能够沿着 z 轴向上移动 6m



任务二：补充无人机的前端规划模块

STEP1.1 补全启发式函数

采用加权曼哈顿距离分离水平与垂直代价。

水平距离上 h_xy = sqrt(dx^2 + dy^2) 保持欧氏几何特性

垂直距离上 h_z = |dz| 单独处理 Z 轴，符合 3D 地图特性

加权系数：Z_PENALTY = 1.5 与边缘成本一致，确保启发值不大于真实代价

```
193    const double Z_PENALTY = 1.5;
194    double h_xy = sqrt(diff(0)*diff(0) + diff(1)*diff(1)); //水平距离
195    double h_z = abs(diff(2)); //垂直距离
196    double heu = h_xy + Z_PENALTY * h_z;
```

Tie_breaker 嵌入极微小放大系数（1.0 + 1e-6），打破 multimap 中 f 值相同节点的排序不确定性，使搜索方向优先朝向起点而非目标。因为离起点近的节点 f 值略低，优先探索，减少对称环境中的无效扩展。同时系数极小（0.0001%），不破坏可采纳性。

```
199    double tie_breaker = 1.0 + 1e-6;
200    heu *= tie_breaker;
```

STEP1.2 补全 A* 算法主循环

首先弹出最小节点，使用 Openset.begin()->second 获取 f 值最小节点，并标记为关闭

状态（id = -1）

```
259         currentPtr = Openset.begin()->second;
260         Openset.erase(Openset.begin());
261         currentPtr->id = -1; //标记为已关闭
```

接着提前终止，达目标时立即返回 true，保存 terminatePtr 供后续回溯

```
264     if (currentPtr->index == goalIdx) {
265       terminatePtr = currentPtr; //保存终点指针用于路径回溯
266       return true;
267     }
```

调用 AstarGetSucc 生成 26 邻域（3×3×3-1），边缘成本已预计算并包含 Z_PENALTY，进行节点拓展

```
270     AstarGetSucc(currentPtr, neighborPtrSets, edgeCostSets);
271     for(unsigned int i=0;i<neighborPtrSets.size();i++)
272     {
273
274       if(neighborPtrSets[i]->id==-1)
275       {
276           continue;
277       }
278       tentative_g_score=currentPtr->g_score+edgeCostSets[i];
279       neighborPtr=neighborPtrSets[i];
280       if(isOccupied(neighborPtr->index))
281       continue;
282       if(neighborPtr->id==0)
283       {
```

id == 0（未访问）时初始化 g/f 值，设置父指针，加入开放列表

id == 1（在开放列表）时仅当新 g 值更优时更新，重新插入 Openset（multimap 自动处理重复键）

```
286             neighborPtr->g_score = tentative_g_score;
287             neighborPtr->f_score = tentative_g_score + getHeu(neighborPtr, endPtr);
288             neighborPtr->Father = currentPtr;
289             neighborPtr->id = 1; //标记为在开放列表中
290             Openset.insert(make_pair(neighborPtr->f_score, neighborPtr));
291             continue;
292           }
293           else if(neighborPtr->id==1)
294           {
295             //???
296             if(neighborPtr->g_score > tentative_g_score) {
297               neighborPtr->g_score = tentative_g_score;
298               neighborPtr->Father = currentPtr;
299               neighborPtr->f_score = tentative_g_score + getHeu(neighborPtr, endPtr);
300               Openset.insert(make_pair(neighborPtr->f_score, neighborPtr));
```

STEP1.3 追溯找到的路径

从 terminatePtr 出发，沿父指针链追溯至起点（Father = nullptr）追溯找到的路径

每节点实时调用 gridIndex2coord，确保输出为物理坐标而非网格索引

```
332     //从终点回溯到起点
333     MappingNodePtr node = terminatePtr;
334     while (node != nullptr) {
335       path.push_back(gridIndex2coord(node->index));
336       node = node->Father;
337     }
```

回溯结果为"终点→起点"，需 reverse 为"起点→终点"符合导航惯例

```
340    reverse(path.begin(), path.end());
```

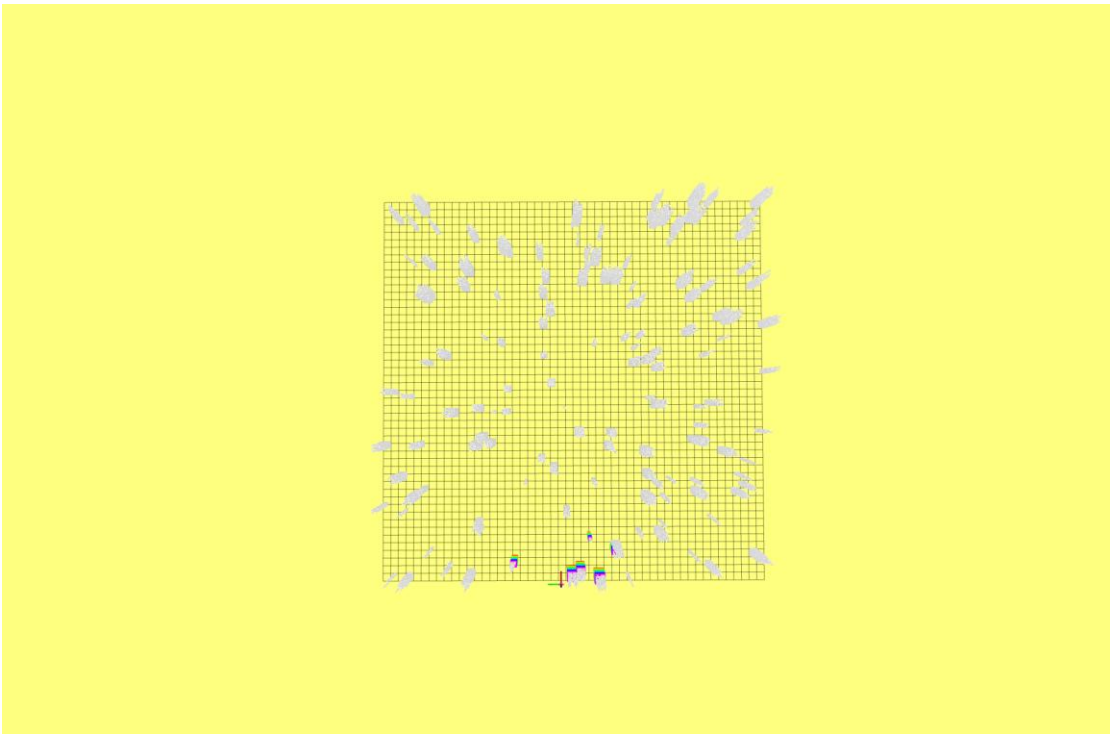
并使用空指针检查防止段错误，提升系统稳定性

```
327    if (terminatePtr == nullptr) {
328      ROS_WARN("No path found, terminatePtr is null");
329      return path;
330    }
```

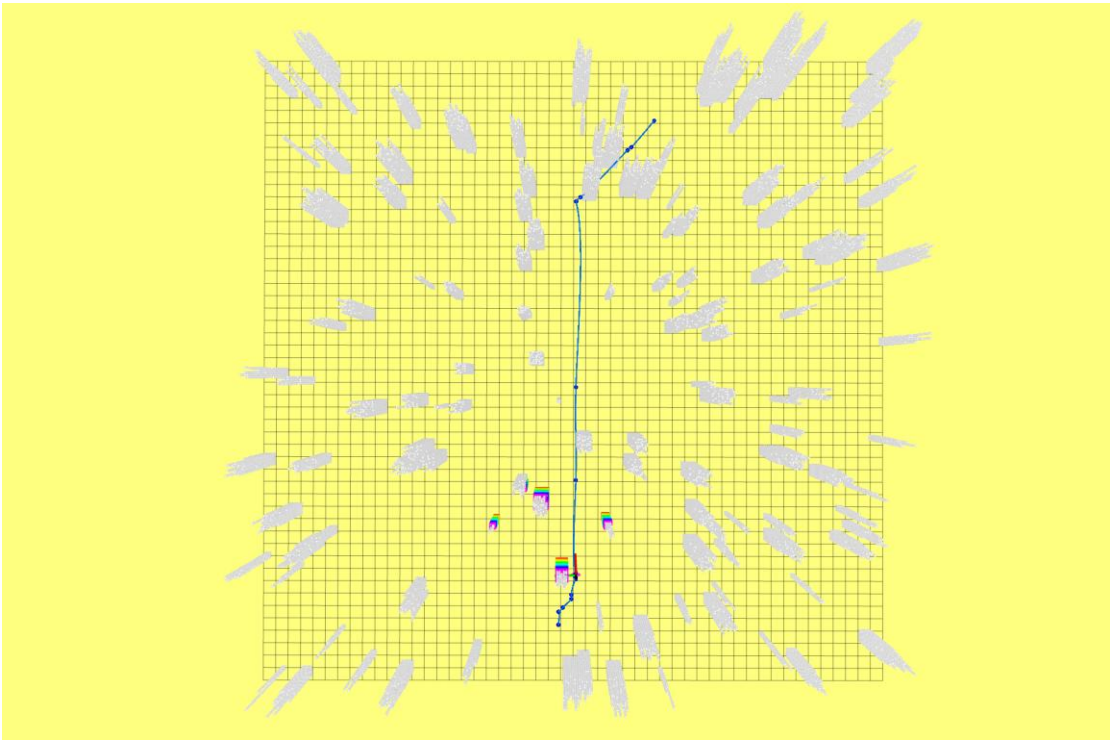
完成后对文件进行编译，编译成功后启动 launch 文件

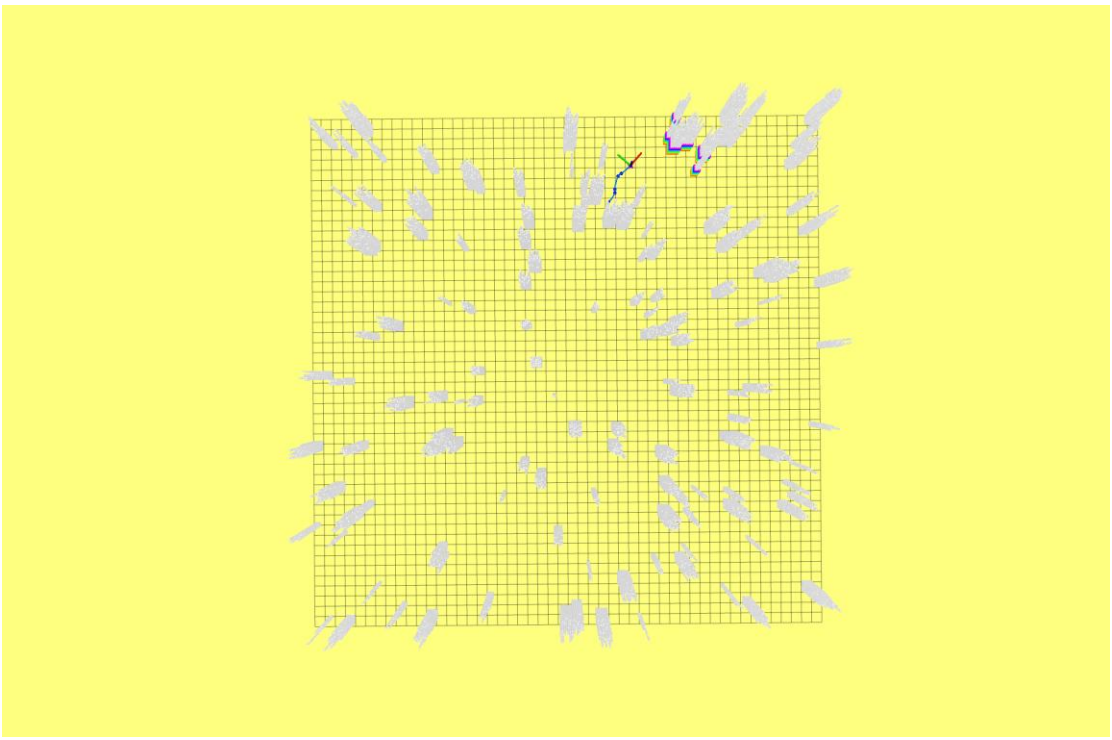
可以看到可视化界面

粉色的为无人机模型，红色的为无人机的局部感知范围，灰色的为点云地图。



点击 3D Nav Goal，在界面上拖动一下。运行成功时，飞机会寻找路径，到达目标点。





任务 3：控制器参数调试

在初始条件下，运行后可得到：

```
kx_ = Eigen::Vector3d(15.7, 8.7, 6.0);
kv_ = Eigen::Vector3d(6.4, 13.4, 4.0);
```

```
stuwork@ubuntu:~/MRPC-2025-homework/code$ python3 calculate_results.py
计算得到的均方根误差（RMSE）值为：0.16123417825943506
轨迹运行总时间为：92.45905041694641
总轨迹长度为：44.83662143268502
是否发生了碰撞：0
综合评价得分为(综合分数越低越好)：59.7059700218133
```

对于评分代码里的 overall_score = 200 * rmse + total_time/5 + total_length/5 + 40 * additional_score，可以发现：RMSE 权重 200，碰撞惩罚 40， 时间长度权重 1/5 快速性几乎可忽略，保守稳定更优。

因此调整思路如下：

kx 降低：牺牲响应速度，换取轨迹平滑度和 RMSE 精度

kv 提升：强力抑制超调震荡，避免碰撞

最终得到最优结果为

```
kx_ = Eigen::Vector3d(15.0,15.0,20.0);
kv_ = Eigen::Vector3d(6.0,6.0,7.0);
```

```
stuwork@ubuntu:~/MRPC-2025-homework/code$ python3 calculate_results.py
计算得到的均方根误差（RMSE）值为：0.1703024183623058
轨迹运行总时间为：68.52060532569885
总轨迹长度为：39.55553661988949
是否发生了碰撞：1
综合评价得分为(综合分数越低越好)：95.67571206157884
```

附：把任务一任务二做完并且开始做任务三后，保存在本地的 git 对象压缩数据损坏了，当时还没有 push 到远程仓库，所以提交的 commit 记录较少 TvT。

```
stuwork@ubuntu:~/MRPC-2025-homework$ git add .
stuwork@ubuntu:~/MRPC-2025-homework$ git commit -m "try2"
error: inflate: data stream error (unknown compression method)
error: unable to unpack 835b853ffa3400ca6abc381d0bc1d136aae4ef60 header
error: inflate: data stream error (unknown compression method)
error: unable to unpack 835b853ffa3400ca6abc381d0bc1d136aae4ef60 header
fatal: loose object 835b853ffa3400ca6abc381d0bc1d136aae4ef60 (stored in .git/objects/83/5b853ffa3400ca6abc381d0bc1d136aae4ef60) is corrupt
```

你的仓库损坏是由于多个 Git 对象的压缩数据损坏导致的。**最可靠的修复方法是重新克隆远程仓库**，如果没有远程仓库，则需要手动删除损坏的对象并重建仓库（可能会丢失部分历史）。修复后务必创建远程备份，防止类似问题再次发生。

```
stuwork@ubuntu:~/MRPC-2025-homework$ git remote add origin https://github.com/joaozi555/MRPC2025.git
stuwork@ubuntu:~/MRPC-2025-homework$ git push -u origin master
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
fatal: loose object 3ed0ad4127901f849c048f1fa56472ae193b7f3d (stored in .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d) is corrupt
fatal: the remote end hung up unexpectedly
fatal: the remote end hung up unexpectedly
fatal: the remote end hung up unexpectedly
error: failed to push some refs to 'https://github.com/joaozi555/MRPC2025.git'
stuwork@ubuntu:~/MRPC-2025-homework$ git gc --prune=now
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
fatal: loose object 3ed0ad4127901f849c048f1fa56472ae193b7f3d (stored in .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d) is corrupt
fatal: failed to run reflog
stuwork@ubuntu:~/MRPC-2025-homework$ git reset --hard
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
error: object file .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d is empty
fatal: loose object 3ed0ad4127901f849c048f1fa56472ae193b7f3d (stored in .git/objects/3e/d0ad4127901f849c048f1fa56472ae193b7f3d) is corrupt
```