空中机器人

末茶 Mo_Tu

课程简介

【主要内容】介绍空中机器人的基本概念和理论，培养空中机器人设计、实践能力，产生空中机器人研究兴趣。

【教师】任沁源，高飞。

【成绩考核】

1. 实验课程报告：40%
2. 期末：30%，开卷
3. MOOC：20%
4. 平时点到：10%

【答疑】 课间提问、预约答疑、电子邮件等均可。

> Tips
>
> 1. 任沁源老师和高飞老师上本科生课都不点名。
> 2. 高飞老师上的部分会比较难，偏数理。

> 提示
>
> 致歉！由于鄙人周末有两门期末，时间仓促，实在无法将所有资料都理到一份文件当中，不过已经尽力将大部分考试所需都整理完了，还请大家谅解！！如有错误，欢迎加我的微信：DaaiDashaTaran，来私戳我！
>
> ——末茶 Mo_Tu，写于 2025 年 11 月 8 日

§1 绪论（主讲：任沁源）

§2 动态模型（主讲：任沁源）

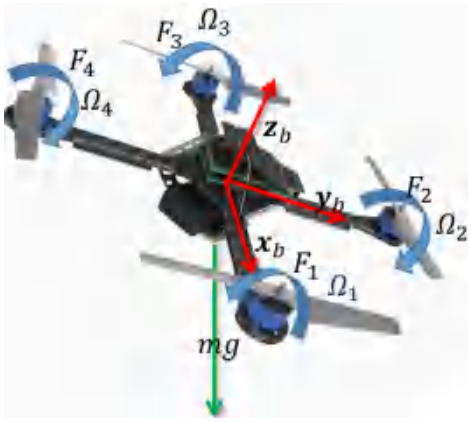
§3 控制基础（主讲：任沁源）

§4 导航基础（主讲：任沁源）

这几部分见另一部分 PPT（“空中机器人 rqy 部分.pdf”）

§5 模型预测控制(MPC)(主讲:高飞)

5.1 四旋翼动力学模型



1. 欧拉角

$$\boldsymbol{R}_x(\varphi)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \sin\varphi & \cos\varphi\end{bmatrix}$$

$$\boldsymbol{R}_y(\theta)=\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}$$

$$\boldsymbol{R}_z(\psi)=\begin{bmatrix}\cos\psi & -\sin\psi & 0\\ \sin\psi & \cos\psi & 0\\ 0 & 0 & 1\end{bmatrix}$$

$$\boldsymbol{R}_z(\psi)\cdot\boldsymbol{R}_x(\varphi)\cdot\boldsymbol{R}_y(\theta)\Rightarrow\boldsymbol{R} \tag{1}$$

(其中 R 为 body to world (ZXY), φ,θ,ψ 分别对应于 X-Y-Z 欧拉角。)

2. 系统状态

$$\boldsymbol{x}=\begin{pmatrix}x & \dot{x} & y & \dot{y} & z & \dot{z} & \varphi & \dot{\varphi} & \theta & \dot{\theta} & \psi & \dot{\psi}\end{pmatrix}^T \tag{2}$$

3. 系统输入

$$u=\begin{pmatrix}U_1\\U_2\\U_3\\U_4\end{pmatrix}=\underbrace{\begin{pmatrix}b & b & b & b\\0 & bl & 0 & -bl\\-bl & 0 & bl & 0\\d & -d & d & -d\end{pmatrix}}_{\text{混控矩阵}}\begin{pmatrix}\Omega_1^2\\\Omega_2^2\\\Omega_3^2\\\Omega_4^2\end{pmatrix} \tag{3}$$

(1) l 是四轴飞行器中心与螺旋桨中心之间的距离，b 和 d 分别是推力和反扭矩系数。

(2) $\Omega_1,\Omega_2,\Omega_3,\Omega_4$ 是螺旋桨的速度。

(3) $F_i=b\Omega_i^2$ 是单桨推力，$U_1=F_1+F_2+F_3+F_4$ 是合推力。

(4) U_2 是绕 x_b 轴旋转的力矩，U_3 是绕 y_b 轴旋转的力矩，U_4是绕 z_b 轴旋转的力矩。

(5) J_r 是总转动惯量，I_x,I_y,I_z 是三轴转动惯量。

5.1.1 完整模型

$$\begin{aligned}
\Omega_r&=-\Omega_1+\Omega_2-\Omega_3+\Omega_4\\
\ddot{x}&=(\sin\psi\sin\varphi+\cos\varphi\sin\theta\cos\psi)\frac{U_1}{m}\\
\ddot{y}&=(-\cos\psi\sin\varphi+\sin\psi\sin\theta\cos\varphi)\frac{U_1}{m}\\
\ddot{z}&=-g+(\cos\theta\cos\varphi)\frac{U_1}{m}\\
\ddot{\varphi}&=\frac{I_y-I_z}{I_x}\dot{\theta}\dot{\psi}-\frac{J_r}{I_x}\dot{\theta}\Omega_r+\frac{U_2}{I_x}\\
\ddot{\theta}&=\frac{I_z-I_x}{I_y}\dot{\varphi}\dot{\psi}+\frac{J_r}{I_y}\dot{\varphi}\Omega_r+\frac{U_3}{I_y}\\
\ddot{\psi}&=\frac{I_x-I_y}{I_z}\dot{\varphi}\dot{\theta}+\frac{U_4}{I_z}
\end{aligned} \tag{4}$$

5.1.2 线性简化模型

平衡悬停态 $(\varphi_0\sim 0,\theta_0\sim 0,u_{1,0}\sim mg)$

1. 牛顿方程

$$m\ddot{\boldsymbol{p}}=\begin{pmatrix}0\\0\\-mg\end{pmatrix}+\boldsymbol{R}\begin{pmatrix}0\\0\\F_1+F_2+F_3+F_4\end{pmatrix}\Rightarrow\begin{cases}\ddot{\boldsymbol{p}}_1=\ddot{x}=g(\theta\cos\psi+\varphi\sin\psi)\\\ddot{\boldsymbol{p}}_2=\ddot{y}=g(\theta\sin\psi-\varphi\cos\psi)\\\ddot{\boldsymbol{p}}_3=\ddot{z}=-g+\dfrac{u_1}{m}\end{cases} \tag{5}$$

其中 $\boldsymbol{R}=\begin{pmatrix}c\psi c\theta-s\varphi s\psi s\theta & -c\varphi s\psi & c\psi s\theta+c\theta s\varphi s\psi\\c\theta s\psi+c\psi s\varphi s\theta & c\varphi c\psi & s\psi s\theta-c\psi c\theta s\varphi\\-c\varphi s\theta & s\varphi & c\varphi c\theta\end{pmatrix}$。

2. 欧拉角微分

$$\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}=\begin{pmatrix}c\theta & 0 & -c\varphi s\theta\\ 0 & 1 & s\varphi\\ s\theta & 0 & c\varphi c\theta\end{pmatrix}\begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}\xrightarrow{\text{平衡悬停态}}\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}=\begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}\tag{6}$$

3. 欧拉方程

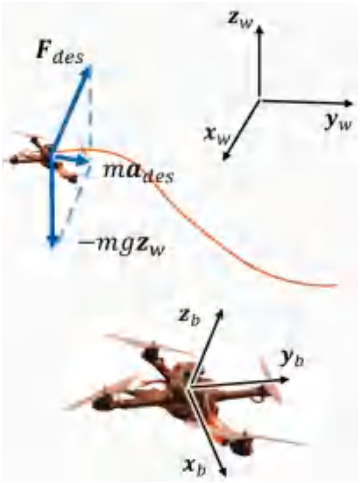
$$\boldsymbol{I}\cdot\begin{pmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{pmatrix}+\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}\times\boldsymbol{I}\cdot\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}=\begin{pmatrix}l(F_2-F_4)\\ l(F_3-F_1)\\ M_1-M_2+M_3-M_4\end{pmatrix}=\begin{pmatrix}u_2\\ u_3\\ u_4\end{pmatrix}\tag{7}$$

5.2 反应式控制

1. 控制目标

(1) 理想情况

$$ma=F-mgz_w,\quad \boldsymbol{F}_{des}=mgz_W+m\boldsymbol{a}_{des},\quad z_{b,des}=\frac{\boldsymbol{F}_{des}}{\|\boldsymbol{F}_{des}\|}\tag{8}$$



(2) 现实情况：用好前馈（预先建立的数学模型，model_base），减小反馈控制调参的压力。

① 期望推力与姿态 -> 自驾仪 Autopilot

$$\boldsymbol{q}_{des}=\boldsymbol{q}_z\otimes\boldsymbol{q}_\psi\tag{9}$$

② 位置与速度误差

$$\boldsymbol{e}_p=\boldsymbol{p}-\boldsymbol{p}_{des},\boldsymbol{e}_v=\boldsymbol{v}-\boldsymbol{v}_{des}\tag{10}$$

③ 反应式控制

$$\boldsymbol{F}_{des}=-K_p\boldsymbol{e}_p-K_v\boldsymbol{e}_v+\underbrace{mgz_W+m\boldsymbol{a}_{des}}_{\text{前馈}}\tag{11}$$

其中 K_p 和 K_v 是增益。

2. 优势

(1) 易于实现

(2) 考虑误差

3. 缺陷

(1) 对于复杂系统实现 Non-trivial

(2) 增益项手动调整

(3) 对于耦合系统与约束无处理

(4) 忽略未来的决定

4. 举例

(1) 腿式机器人：太过复杂

(2) 无人机，车：分解为姿态位置或横向纵向控制器，忽略状态耦合

5.3 最优控制

1. 系统模型

$$\begin{aligned}\dot{\boldsymbol{x}} &= f_c(\boldsymbol{x}, \boldsymbol{u}) \\ \boldsymbol{x}_{k+1} &= f_d(\boldsymbol{x}_k, \boldsymbol{u}_k)\end{aligned} \tag{12}$$

其中 x_0 为初始条件，x_k 为状态，u_k 为输入。

2. 动力学模型（系统状态见式 2，系统输入见式 3，完整模型见式 4）

$$\dot{\boldsymbol{x}} = f_c(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \dot{x} \\ u_x \frac{1}{m} U_1 \\ \dot{y} \\ u_y \frac{1}{m} U_1 \\ \dot{z} \\ -g + (\cos\theta\cos\varphi)\frac{1}{m}U_1 \\ \dot{\varphi} \\ a_1\dot{\theta}\dot{\psi} - a_2\dot{\theta}\Omega_r + b_1 U_2 \\ \dot{\theta} \\ a_3\dot{\varphi}\dot{\psi} + a_4\dot{\varphi}\dot{\Omega}_r + b_2 U_3 \\ \dot{\psi} \\ a_5\dot{\varphi}\dot{\theta} + b_3 U_4 \end{pmatrix} \tag{13}$$

(1) $a_1 = \frac{I_y - I_z}{I_x}, a_2 = \frac{J_r}{I_x}, a_3 = \frac{I_z - I_x}{I_y}, a_4 = \frac{J_r}{I_y}, a_5 = \frac{I_x - I_y}{I_z}$

(2) $b_1 = \frac{1}{I_x}, b_2 = \frac{1}{I_y}, b_3 = \frac{1}{I_z}$

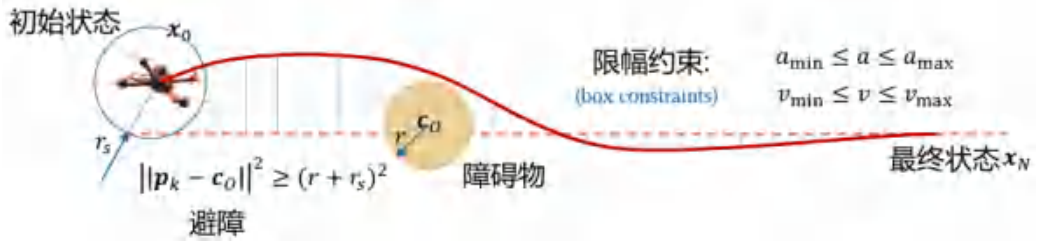
(3) $u_x = (\sin\psi\sin\varphi + \cos\psi\sin\theta\cos\varphi), \quad u_y = (-\cos\psi\sin\varphi + \sin\psi\sin\theta\cos\varphi)$

3. 目标最小化函数

$$\min_{u_{0:N-1}} \sum_{k=0}^{N-1} \underbrace{q(\boldsymbol{x}_k, \boldsymbol{u}_k)}_{\text{阶段代价 (stage cost)}} + \underbrace{p(\boldsymbol{x}_N)}_{\text{终端代价 (terminal cost)。}} \tag{14}$$

【约束项】

$$\begin{aligned}\boldsymbol{x}_{k+1} &= f_d(\boldsymbol{x}_k, \boldsymbol{u}_k) \\ h(\boldsymbol{x}_k, \boldsymbol{u}_k) &= \boldsymbol{0} \quad \text{等式约束} \\ g(\boldsymbol{x}_k, \boldsymbol{u}_k) &\le \boldsymbol{0} \quad \text{不等式约束}\end{aligned} \tag{15}$$

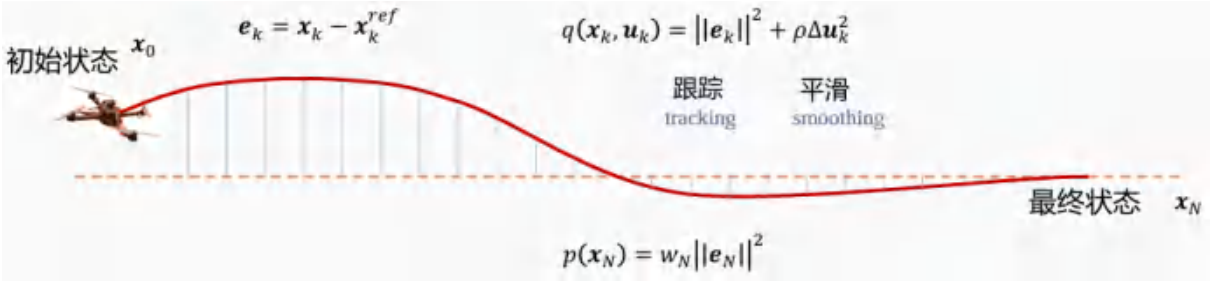


【最优解】

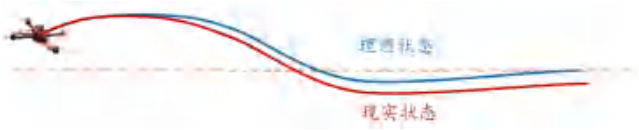
$$\boldsymbol{z}^* = \begin{pmatrix} \boldsymbol{u}_0^T & \cdots & \boldsymbol{u}_{N-1}^T \end{pmatrix}^T \tag{16}$$

理想情况下用最优解 z^* 作为系统的控制输入!

4. 跟踪任务（tracking）举例



5. 开环最优控制难点



- 系统模型不准确，时间积累更多累计误差
- 最优解 z^* 无法被准确执行
- 较长的预测时域使得问题难以求解
- 系统可能受到外界扰动

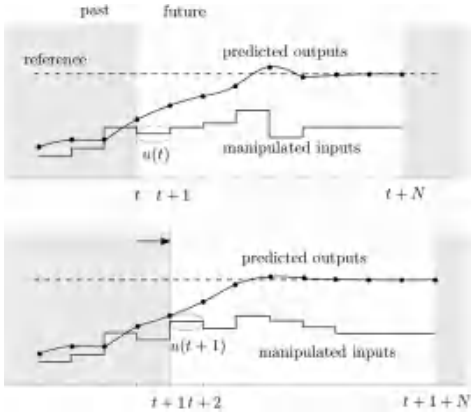
5.4 模型预测控制（MPC）

5.4.1 概念

1. 使用动力学模型预测未来走向（有限的预测时域）选择最佳的控制输入
   (1) 测量信息的反馈 → 从估计的当前状态开始
   (2) 优化最佳的控制序列 → 在有限的预测时域内找到最佳的控制输入序列
   (3) 滚动预测框架 → 执行第一个最佳的控制输入，更新状态重规划
2. 跟踪目标为例

$$\min_{u_0, u_1, \ldots, u_N} \sum_{k}^{N} \|p_k - r(t)\|^2 + \rho \Delta u_k^2$$

$$\text{s.t.} \begin{cases} x_{k+1} = f(x_k, u_k) & \text{预测模型} \\ y_k = g(x_k) & \text{预测模型} \\ x_{min} \le x_k \le x_{max} & \text{约束} \\ u_{min} \le u_k \le u_{max} & \text{约束} \\ x_0 = x(t) & \text{状态反馈} \end{cases}$$



3. 别名
   - 开环最优反馈 (Open Loop Optimal Feedback)
   - 反应式规划 (Reactive Scheduling)
   - 滚动优化控制 (Receding Horizon Control)
4. 优势
   - 考虑未来时域（尽管有限）

- 考虑误差
- 减小问题规模（求解器一般有 warm-start，由上次最优解开始作为初值）

5.4.2 案例

1. 非线性 MPC 用于容错控制
2. Whole-body MPC 用于轮腿式机器人

5.4.3 参数选择

1. 模型的选择 - 效率与准确性之间权衡
   (1) 二维简化模型（简易性）

$$\dot{\boldsymbol{x}} = f_c(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} \dot{z} \\ -g + \frac{1}{m} U_1 \cos\varphi \\ \dot{y} \\ -\frac{1}{m} U_1 \sin\varphi \\ \dot{\varphi} \\ \frac{1}{I_x} U_2 \end{pmatrix} \tag{17}$$

   ① 平衡悬停态 $$\theta = \dot{\theta} = \psi = \dot{\psi} \sim 0 \tag{18}$$

   ② 系统状态 $$\boldsymbol{x} = [z, \dot{z}, y, \dot{y}, \varphi, \dot{\varphi}] \tag{19}$$

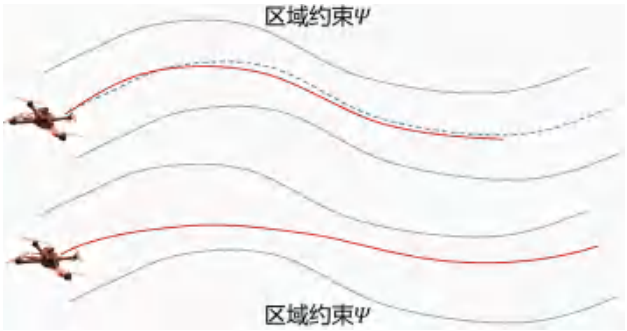
   ③ 系统输入 $$\boldsymbol{u} = [U_1, U_2] \tag{20}$$

   (2) 完整状态模型 式 4（准确性）
2. 代价函数的选择
   (1) 最小化跟踪误差（控制器）

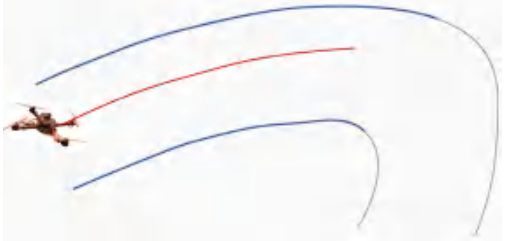
$$\min_{u_0, u_1, \ldots, u_N} \sum_k^N \|\boldsymbol{p}_k - \boldsymbol{r}(t)\|^2 + \rho \Delta \boldsymbol{u}_k^2 \tag{21}$$

   (2) 最小化时间（规划器）：时间参数化控制指令

$$\min_{\boldsymbol{u}(t)} \sum_k^N \rho \Delta \boldsymbol{u}_k^2 + \gamma T \quad \text{st. } \boldsymbol{p}(t) \in \Psi \tag{22}$$



3. 预测时域：计算量与解的可行性的权衡
   (1) 较短的预测时域
      ① 较小的计算量
      ② 次优的解（可能不安全）
   (2) 较长的预测时域
      ① 较大的计算量
      ② 更优更安全的解



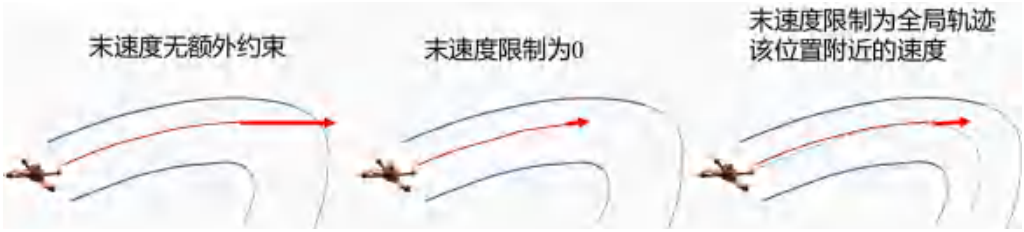
4. 预测时长 (Prediction horizon) 的长短？→ 权衡计算时间 (computation overload) 和递归可行性 (recursive feasibility)

   (1) 较短的预测时长

      ① 减少计算时间

      ② 短视性（不安全）

   (2) 末状态增加某些额外的约束可以提高递归可行性。



5.5 离散化求解

1. 线性时变模型 (Linear Time-Varying (LTV) model)

$$x_{k+1} = Ax_k + Bu_k \rightarrow x_{k+1} = A_k(t)x_k + B_k(t)u_k \tag{23}$$

2. 非线性模型

$$\dot{x} = f(x, u) \tag{24}$$

3. 通过线性化非线性模型得到 LTV。

$$\begin{aligned} \dot{x} &\approx f(\overline{x}, \overline{u}) + \frac{\partial f}{\partial x}\,|_{\overline{x},\overline{u}}\,(x - \overline{x}) + \frac{\partial f}{\partial u}\,|_{\overline{x},\overline{u}}\,(u - \overline{u}) \\ \dot{x} &= A_c x + B_c u + g_c \end{aligned} \tag{25}$$

4. 用前向欧拉法把线性模型转化为离散形式。

$$\begin{aligned} &\dot{x} = A_c x + B_c u + g_c \\ \Rightarrow\ &\frac{x_{k+1} - x_k}{T_s} = A_c x_k + B_c u_k + g_c \\ \Rightarrow\ &x_{k+1} = (I + T_s A_c)x_k + T_s B_c u_k + T_s g_c \\ \Rightarrow\ &x_{k+1} = A_k x_k + B_k u_k + g_k \end{aligned} \tag{26}$$

5. 此时可以在线求解线性 MPC。

5.6 离散线性模型计算

1. 线性预测模型

$$\begin{cases} x_{k+1} = Ax_k + Bu_k \\ y_k = Cx_k \end{cases} \tag{27}$$

2. 状态量与输入的关系

$$x_k = A^k x_0 + \sum_{j=0}^{k-1} A^j B u_{k-1-j} \tag{28}$$

3. 状态传播与计算

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \\ x_N \end{pmatrix} = \begin{pmatrix} B & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{pmatrix} \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} + \begin{pmatrix} A \\ A^2 \\ \vdots \\ A^N \end{pmatrix} x_0 \tag{29}$$

4. 线性 MPC

(1) 二次型 cost

$$J = \boldsymbol{x}_N^T \boldsymbol{P} \boldsymbol{x}_N + \sum_{k=0}^{N-1} \left(\boldsymbol{x}_k^T \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k \right) \tag{30}$$

其中（半正定条件）$\boldsymbol{P} = \boldsymbol{P}^T \succeq 0, \boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0, \boldsymbol{R} = \boldsymbol{R}^T \succ 0$。

(2) 目标：找到最优的控制序列 $u_{0:N-1}^*$ 最小化 cost 函数 J。

$$J = \boldsymbol{x}_0^T \boldsymbol{Q} \boldsymbol{x}_0 + \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{N-1} \\ \boldsymbol{x}_N \end{pmatrix}^T \begin{pmatrix} \boldsymbol{Q} & 0 & 0 & \dots & 0 \\ 0 & \boldsymbol{Q} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \boldsymbol{Q} & 0 \\ 0 & 0 & \dots & 0 & \boldsymbol{P} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{N-1} \\ \boldsymbol{x}_N \end{pmatrix} + \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{R} & 0 & \dots & 0 \\ 0 & \boldsymbol{R} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \boldsymbol{R} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{pmatrix} \tag{31}$$

① 定义紧凑形式的矩阵

$$\begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{N-1} \\ \boldsymbol{x}_N \end{pmatrix} = \underbrace{\begin{pmatrix} \boldsymbol{B} & 0 & \dots & 0 \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \dots & \boldsymbol{B} \end{pmatrix}}_{\overline{\boldsymbol{S}}} \underbrace{\begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{pmatrix}}_{\boldsymbol{z}} + \underbrace{\begin{pmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{pmatrix}}_{\overline{\boldsymbol{T}}} \boldsymbol{x}_0$$

$$J = \boldsymbol{x}_0^T \boldsymbol{Q} \boldsymbol{x}_0 + \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{N-1} \\ \boldsymbol{x}_N \end{pmatrix}^T \underbrace{\begin{pmatrix} \boldsymbol{Q} & 0 & 0 & \dots & 0 \\ 0 & \boldsymbol{Q} & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & 0 & \boldsymbol{Q} & 0 \\ 0 & 0 & \dots & 0 & \boldsymbol{P} \end{pmatrix}}_{\overline{\boldsymbol{Q}}} \begin{pmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_{N-1} \\ \boldsymbol{x}_N \end{pmatrix} + \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{pmatrix}^T \underbrace{\begin{pmatrix} \boldsymbol{R} & 0 & \dots & 0 \\ 0 & \boldsymbol{R} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \boldsymbol{R} \end{pmatrix}}_{\overline{\boldsymbol{R}}} \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{pmatrix} \tag{32}$$

② 简化后的成本函数

$$\begin{aligned} J(\boldsymbol{z}, \boldsymbol{x}_0) &= \left(\overline{\boldsymbol{S}} \boldsymbol{z} + \overline{\boldsymbol{T}} \boldsymbol{x}_0 \right)^T \overline{\boldsymbol{Q}} \left(\overline{\boldsymbol{S}} \boldsymbol{z} + \overline{\boldsymbol{T}} \boldsymbol{x}_0 \right) + \boldsymbol{z}^T \overline{\boldsymbol{R}} \boldsymbol{z} + \boldsymbol{x}_0^T \boldsymbol{Q} \boldsymbol{x}_0 \\ &= \frac{1}{2} \boldsymbol{z}^T \underbrace{2\left(\overline{\boldsymbol{R}} + \overline{\boldsymbol{S}}^T \overline{\boldsymbol{Q}} \overline{\boldsymbol{S}} \right)}_{\boldsymbol{H}} \boldsymbol{z} + \boldsymbol{x}_0^T \underbrace{2\overline{\boldsymbol{T}}^T \overline{\boldsymbol{Q}} \overline{\boldsymbol{S}}}_{\boldsymbol{F}} \boldsymbol{z} + \frac{1}{2} \boldsymbol{x}_0^T \underbrace{2\left(\boldsymbol{Q} + \overline{\boldsymbol{T}}^T \overline{\boldsymbol{Q}} \overline{\boldsymbol{T}} \right)}_{\boldsymbol{Y}} \boldsymbol{x}_0 \end{aligned} \tag{33}$$

(3) MPC 的紧凑形式

$$J(\boldsymbol{z}, \boldsymbol{x}_0) = \frac{1}{2} \boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} + \boldsymbol{x}_0^T \boldsymbol{F} \boldsymbol{z} + \frac{1}{2} \boldsymbol{x}_0^T \boldsymbol{Y} \boldsymbol{x}_0, \quad \boldsymbol{z} = \begin{pmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{pmatrix} \tag{34}$$

目标即是找到梯度为零的解。

$$\nabla_{\boldsymbol{z}} J(\boldsymbol{z}, \boldsymbol{x}_0) = \boldsymbol{H} \boldsymbol{z} + \boldsymbol{F}^T \boldsymbol{x}_0 = 0 \rightarrow \boldsymbol{z}^* = -\boldsymbol{H}^{-1} \boldsymbol{F}^T \boldsymbol{x}_0 \quad \text{（解序列“batch” solution）} \tag{35}$$

(4) 无约束 MPC = 线性反馈控制 (linear state-feedback)

5.7 线性模型计算实例

1. 例子：选择最佳系统输入，即总推力 U_1 与 x 轴力矩 U_2。
2. 这里采用二维简化模型 式 17
3. 线性模型离散化

(1) 前向欧拉法

$$\dot{\boldsymbol{x}}=f_c(\boldsymbol{x},\boldsymbol{u})=\begin{pmatrix}\dot{z}\\ -g+\frac{1}{m}U_1\cos\varphi\\ \dot{y}\\ -\frac{1}{m}U_1\sin\varphi\\ \dot{\varphi}\\ \frac{1}{I_x}U_2\end{pmatrix}\xrightarrow{\text{前向欧拉法}}\boldsymbol{x}_{t+1}=f(\boldsymbol{x}_t,\boldsymbol{u}_t)=\begin{pmatrix}\boldsymbol{x}_t^1+h\boldsymbol{x}_t^2\\ \boldsymbol{x}_t^2+h\left(-g+\frac{1}{m}\boldsymbol{u}_t^1\cos\boldsymbol{x}_t^5\right)\\ \boldsymbol{x}_t^3+h\boldsymbol{x}_t^4\\ \boldsymbol{x}_t^4-\frac{h}{m}\boldsymbol{u}_t^1\sin\boldsymbol{x}_t^5\\ \boldsymbol{x}_t^5+h\boldsymbol{x}_t^6\\ \boldsymbol{x}_t^6+\frac{h}{I_x}\boldsymbol{u}_t^2\end{pmatrix}\tag{36}$$

> 提示
>
> 做法参考 式 25 和 式 26，h 是步长。

(2) 在工作点 $\left(\boldsymbol{x}_t^{ref},\boldsymbol{u}_t^{ref}\right)$ 处线性展开

$$\boldsymbol{x}_{t+1}=f(\boldsymbol{x}_t,\boldsymbol{u}_t)\approx f\left(\boldsymbol{x}_t^{ref},\boldsymbol{u}_t^{ref}\right)+\tilde{\boldsymbol{A}}_t\left(\boldsymbol{x}_t-\boldsymbol{x}_t^{ref}\right)+\tilde{\boldsymbol{B}}_t\left(\boldsymbol{u}_t-\boldsymbol{u}_t^{ref}\right)\tag{37}$$

$$\tilde{\boldsymbol{A}}_t=\frac{\partial f(\boldsymbol{x},\boldsymbol{u})}{\partial\boldsymbol{x}}\Big|_{\left(\boldsymbol{x}_t^{ref},\boldsymbol{u}_t^{ref}\right)}\longrightarrow\tilde{\boldsymbol{A}}_t=\begin{pmatrix}1&h&0&0&0&0\\0&1&0&0&-\frac{h}{m}\boldsymbol{u}_t^{ref,1}\sin\boldsymbol{x}_t^{ref,5}&0\\0&0&1&h&0&0\\0&0&0&1&-\frac{h}{m}\boldsymbol{u}_t^{ref,1}\cos\boldsymbol{x}_t^{ref,5}&0\\0&0&0&0&1&h\\0&0&0&0&0&1\end{pmatrix}\tag{38}$$

$$\tilde{\boldsymbol{B}}_t=\frac{\partial f(\boldsymbol{x},\boldsymbol{u})}{\partial\boldsymbol{u}}\Big|_{\left(\boldsymbol{x}_t^{ref},\boldsymbol{u}_t^{ref}\right)}\longrightarrow\tilde{\boldsymbol{B}}_t=\begin{pmatrix}0&0\\\frac{h}{m}\cos\boldsymbol{x}_t^{ref,5}&0\\0&0\\-\frac{h}{m}\sin\boldsymbol{x}_t^{ref,5}&0\\0&0\\0&\frac{h}{I_x}\end{pmatrix}$$

> 提示
>
> 求 $\tilde{A}_t$（$\tilde{B}_t$），就是用分别用 x_t^1~x_t^6（u_t^1和u_t^2）对 式 36 分别求导，然后将得到的 6（2）列合并，即得到 $\tilde{A}_t$（$\tilde{B}_t$）。

(3) 结果

$$\boldsymbol{A}_t=\begin{pmatrix}1&h&0&0&0&0\\0&1&0&0&-\frac{h}{m}\overline{\boldsymbol{u}}_t^1\sin\overline{\boldsymbol{x}}_t^5&0\\0&0&1&h&0&0\\0&0&0&1&-\frac{h}{m}\overline{\boldsymbol{u}}_t^1\cos\overline{\boldsymbol{x}}_t^5&0\\0&0&0&0&1&h\\0&0&0&0&0&1\end{pmatrix}\quad\boldsymbol{B}_t=\begin{pmatrix}0&0\\\frac{h}{m}\cos\overline{\boldsymbol{x}}_t^5&0\\0&0\\-\frac{h}{m}\sin\overline{\boldsymbol{x}}_t^5&0\\0&0\\0&\frac{h}{I_x}\end{pmatrix}\tag{39}$$

$$\boldsymbol{g}_t=f\left(\boldsymbol{x}_t^{ref},\boldsymbol{u}_t^{ref}\right)-\boldsymbol{A}_t\boldsymbol{x}_t^{ref}-\boldsymbol{B}_t\boldsymbol{u}_t^{ref}$$

$$\boldsymbol{x}_{t+1}=\boldsymbol{A}_t\boldsymbol{x}_t+\boldsymbol{B}_t\boldsymbol{u}_t+\boldsymbol{g}_t$$

4. 最小化的累加和

$$\left(p_x-x_{ref}\right)^2+\left(p_y-y_{ref}\right)^2+w_p\Delta a^2+w_s\Delta\delta^2\rightarrow\text{二次型}\tag{40}$$

5. 增广模型

$$\begin{pmatrix} x \\ u \end{pmatrix}_{k+1} = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix} \begin{pmatrix} x \\ u \end{pmatrix}_k + \begin{pmatrix} B \\ I \end{pmatrix} \Delta u_k \quad z = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N-1} \end{pmatrix} \text{ or } z = \begin{pmatrix} \Delta u_0 \\ \Delta u_1 \\ \vdots \\ \Delta u_{N-1} \end{pmatrix} \tag{41}$$

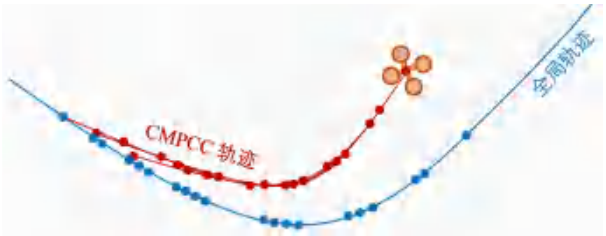
6. 约束形式

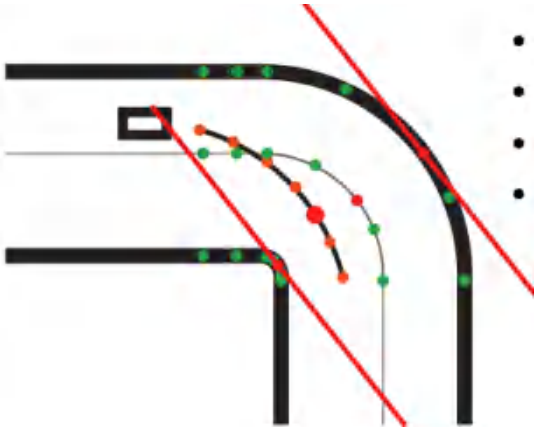
$$\begin{cases} a_{\min} \le a \le a_{\max} \\ v_{\min} \le v \le v_{\max} \end{cases} \rightarrow \text{线性约束二次型 (Linear constrained QP formulation)} \tag{42}$$

> 一些工程技巧
>
> 1. 根据参考轨迹人为给定。
> 2. Warm-start：在上一次求解处对模型线性化 u^*_{k-1}, x^*_{k-1}。

5.8 CMPCC (Corridor-based MPC Contouring Control)

1. 平衡跟踪进度和执行进度。
2. 走廊约束保证安全。
3. 终端速度约束保证可行。
4. QP 方法让问题在机载电脑上求解时间少于 5ms。





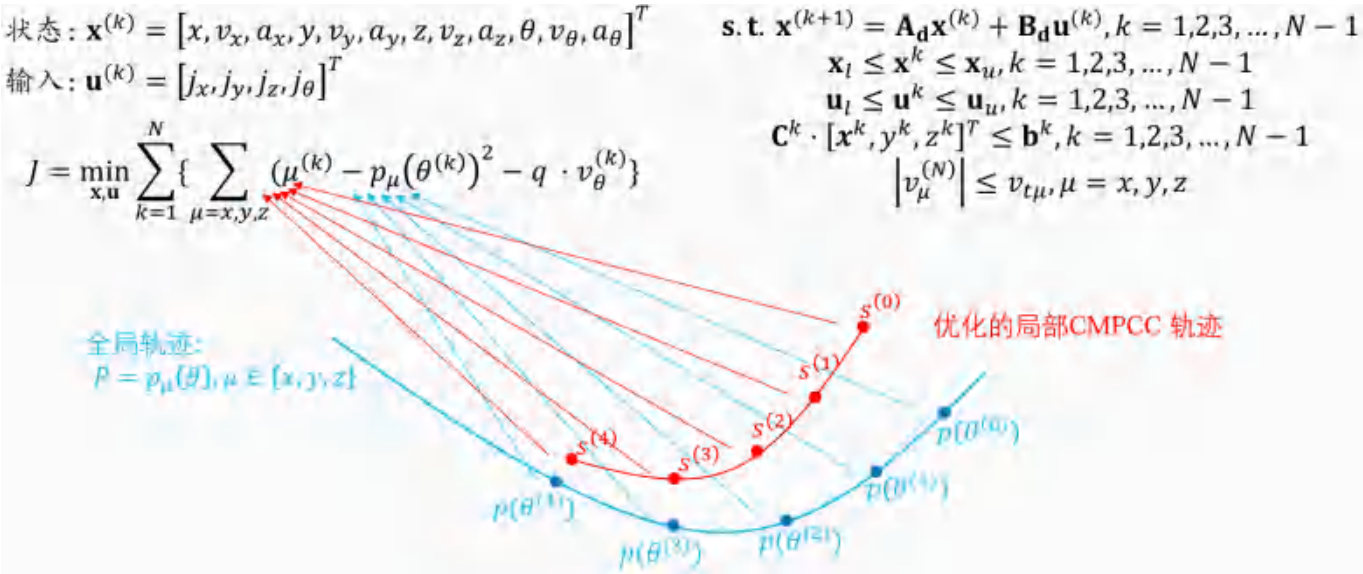
5.8.1 目标函数

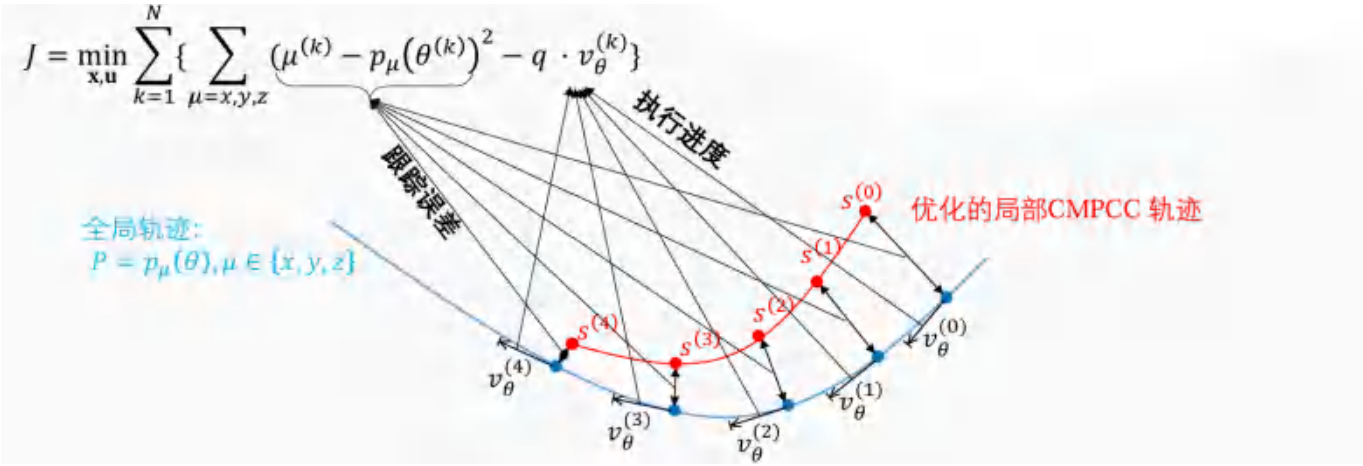
状态：$\mathbf{x}^{(k)} = [x, v_x, a_x, y, v_y, a_y, z, v_z, a_z, \theta, v_\theta, a_\theta]^T$

输入：$\mathbf{u}^{(k)} = [j_x, j_y, j_z, j_\theta]^T$

$$J = \min_{\mathbf{x},\mathbf{u}} \sum_{k=1}^{N} \{ \sum_{\mu=x,y,z} (\mu^{(k)} - p_\mu(\theta^{(k)}))^2 - q \cdot v_\theta^{(k)} \}$$

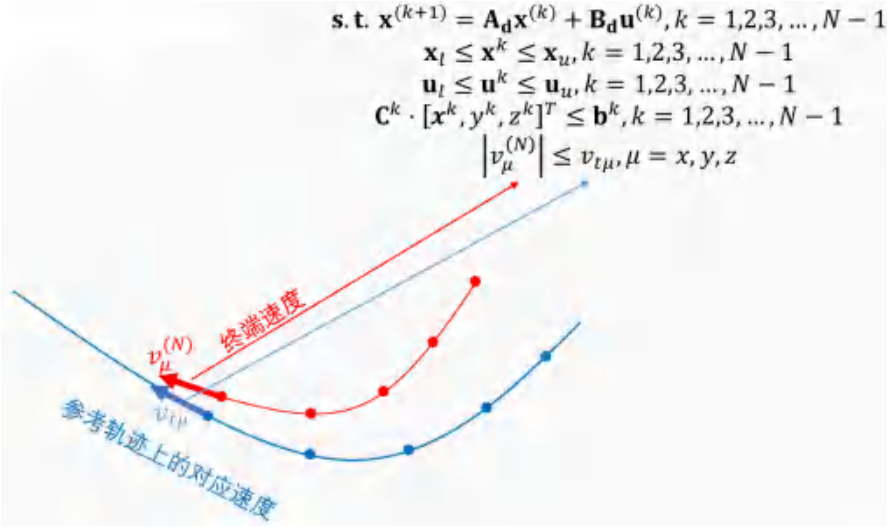
$$\text{s.t. } \mathbf{x}^{(k+1)} = \mathbf{A_d}\mathbf{x}^{(k)} + \mathbf{B_d}\mathbf{u}^{(k)}, k = 1,2,3,\dots,N-1$$
$$\mathbf{x}_l \le \mathbf{x}^k \le \mathbf{x}_u, k = 1,2,3,\dots,N-1$$
$$\mathbf{u}_l \le \mathbf{u}^k \le \mathbf{u}_u, k = 1,2,3,\dots,N-1$$
$$\mathbf{C}^k \cdot [\mathbf{x}^k, y^k, z^k]^T \le \mathbf{b}^k, k = 1,2,3,\dots,N-1$$
$$\left| v_\mu^{(N)} \right| \le v_{t\mu}, \mu = x, y, z$$



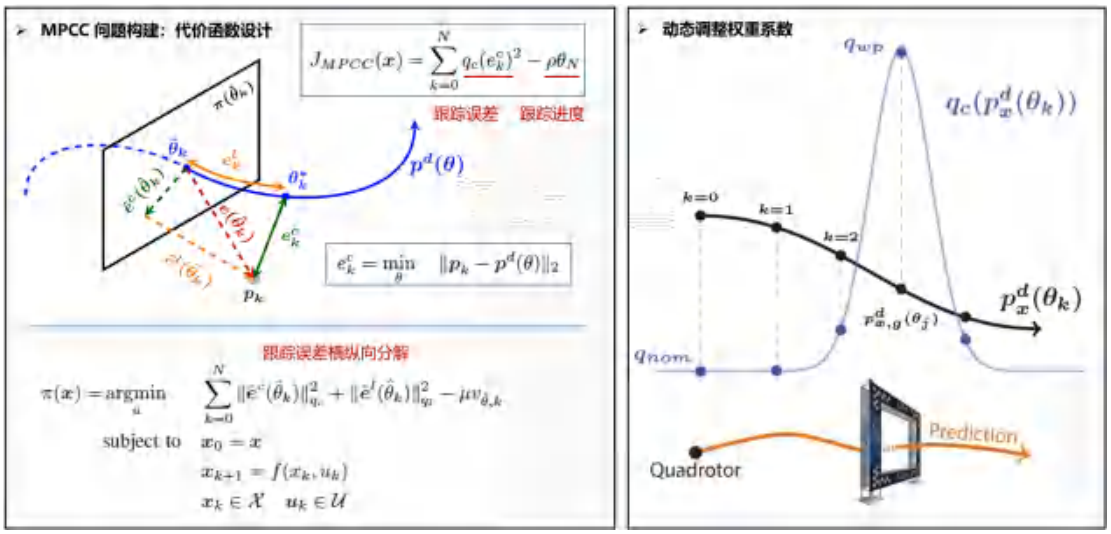


5.8.2 终端速度约束

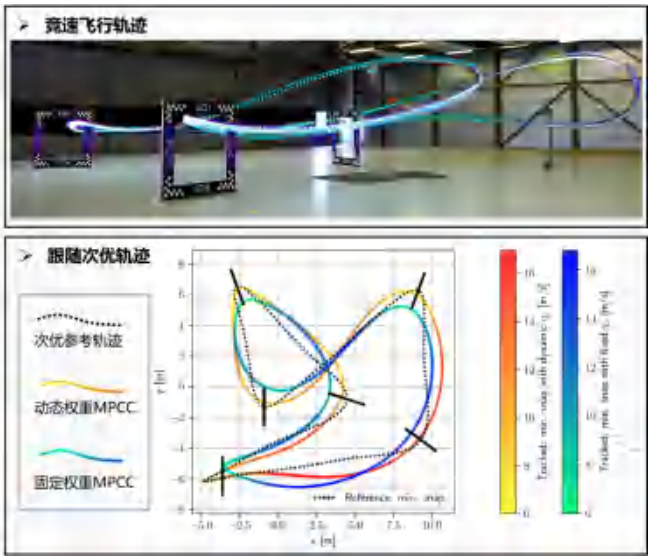
限制终端速度可确保轨迹可行性，同时大大缩短规划的时间。



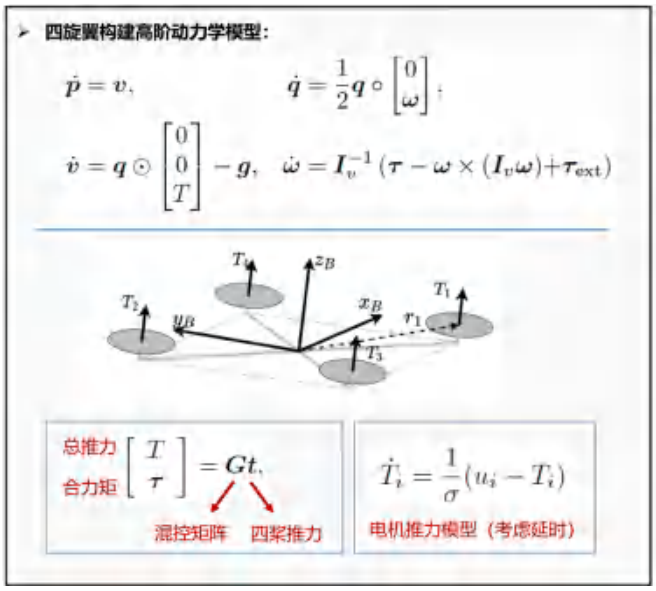
5.9 无人机竞速

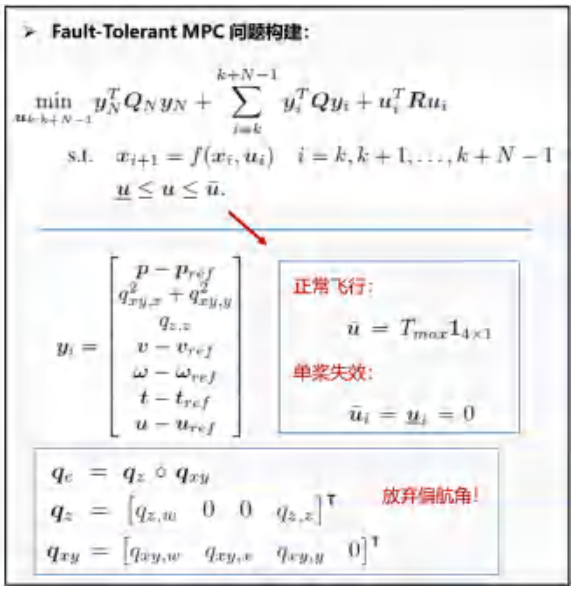


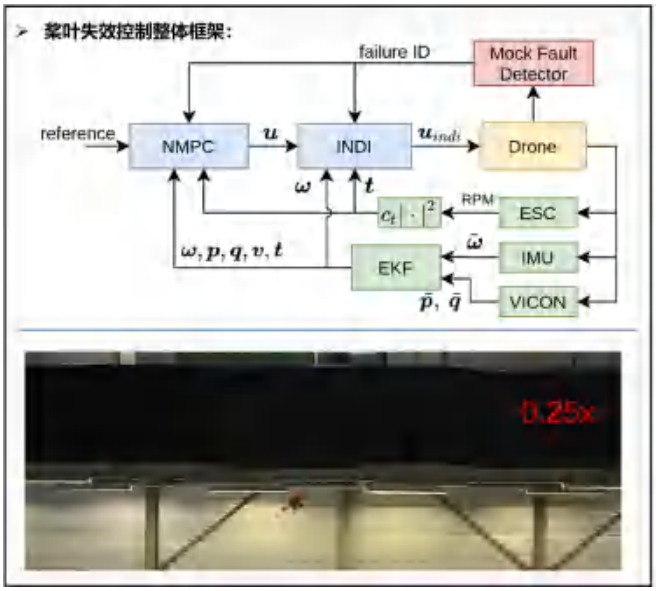

MPCC 问题构建：完整优化问题
$\dot{p} = v$
$\dot{q} = \frac{1}{2} q \odot \begin{bmatrix} 0 \\ \omega \end{bmatrix}$
$\dot{v} = g + \frac{1}{m} \mathrm{R}(q) f_T$
$\dot{\omega} = \mathbf{J}^{-1} (\tau - \omega \times \mathbf{J}\omega)$
$x = [p \quad q \quad v \quad w \quad f \quad \theta \quad v_\theta]^T$,
$u = [\Delta v_\theta \quad \Delta f]^T$
$\pi(x) = \underset{u}{\operatorname{argmin}} \sum_{k=0}^{N} \|e^l(\theta_k)\|^2_{q_l} + \|e^v(\theta_k)\|^2_{q_v} + \|\omega_k\|^2_{Q_\omega} + \|\Delta v_{\theta_k}\|^2_{r_{\Delta v}} + \|\Delta f_k\|^2_{R_{\Delta f}} - \mu v_{\theta,k}$
subject to $x_0 = x$
$x_{k+1} = f(x_k, u_k)$
$\underline{\omega} \le \omega \le \overline{\omega}$
$\underline{f} \le f \le \overline{f}$
$0 \le v_\theta \le \overline{v_\theta}$
$\underline{\Delta v_\theta} \le \Delta v_\theta \le \overline{\Delta v_\theta}$
$\underline{\Delta f} \le \Delta f \le \overline{\Delta f}$

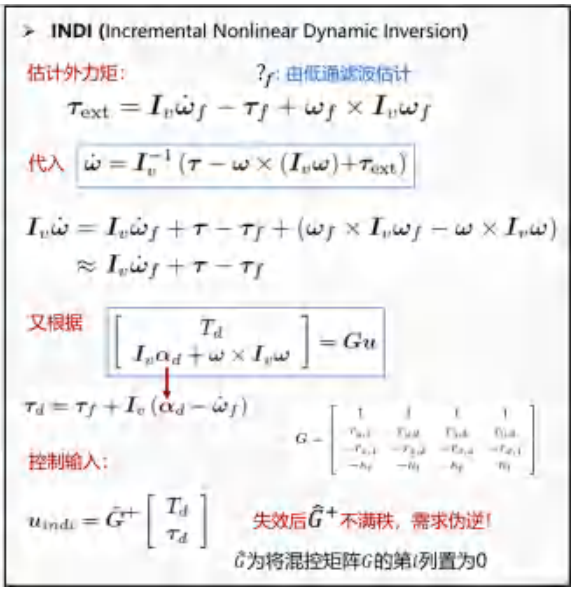

竞速飞行轨迹
跟随次优轨迹
次优参考轨迹
动态权重MPCC
固定权重MPCC

5.10 桨叶失效控制


四旋翼构建高阶动力学模型：
$\dot{p} = v, \quad \dot{q} = \frac{1}{2} q \circ \begin{bmatrix} 0 \\ \omega \end{bmatrix};$
$\dot{v} = q \odot \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} - g, \quad \dot{\omega} = I_v^{-1} (\tau - \omega \times (I_v \omega) + \tau_{\text{ext}})$
T_0
z_B
T_1
T_2
y_B
x_B
T_3
总推力
合力矩 $\begin{bmatrix} T \\ \tau \end{bmatrix} = Gt,$
混控矩阵
四桨推力
$\dot{T}_i = \frac{1}{\sigma}(u_i - T_i)$
电机推力模型（考虑延时）


Fault-Tolerant MPC 问题构建：
$\min_{u_{k:k+N-1}} y_N^T Q_N y_N + \sum_{i=k}^{k+N-1} y_i^T Q y_i + u_i^T R u_i$
s.t. $x_{i+1} = f(x_i, u_i) \quad i = k, k+1, \ldots, k+N-1$
$\underline{u} \le u \le \bar{u}.$
$y_i = \begin{bmatrix} p - p_{ref} \\ q_{xy,x}^2 + q_{xy,y}^2 \\ q_{z,z} \\ v - v_{ref} \\ \omega - \omega_{ref} \\ t - t_{ref} \\ u - u_{ref} \end{bmatrix}$
正常飞行：
$\bar{u} = T_{max} 1_{4\times 1}$
单桨失效：
$\bar{u}_i = \underline{u}_i = 0$
$q_e = q_z \diamond q_{xy}$
$q_z = [q_{z,w} \quad 0 \quad 0 \quad q_{z,z}]^\intercal$
$q_{xy} = [q_{xy,w} \quad q_{xy,x} \quad q_{xy,y} \quad 0]^\intercal$
放弃偏航角！


桨叶失效控制整体框架：
failure ID
Mock Fault Detector
reference
NMPC
u
INDI
u_{indi}
Drone
ω
t
RPM
$c_t \cdot \cdot^2$
ESC
ω, p, q, v, t
EKF
$\tilde{\omega}$
IMU
$\tilde{p}, \tilde{q}$
VICON
0.25x


INDI (Incremental Nonlinear Dynamic Inversion)
估计外力矩：
$?_f$: 由低通滤波估计
$\tau_{\text{ext}} = I_v \dot{\omega}_f - \tau_f + \omega_f \times I_v \omega_f$
代入 $\dot{\omega} = I_v^{-1} (\tau - \omega \times (I_v \omega) + \tau_{\text{ext}})$
$I_v \dot{\omega} = I_v \dot{\omega}_f + \tau - \tau_f + (\omega_f \times I_v \omega_f - \omega \times I_v \omega)$
$\approx I_v \dot{\omega}_f + \tau - \tau_f$
又根据 $\begin{bmatrix} T_d \\ I_v \alpha_d + \omega \times I_v \omega \end{bmatrix} = Gu$
$\tau_d = \tau_f + I_v (\dot{\alpha}_d - \dot{\omega}_f)$
$G = \begin{bmatrix} 1 & 1 & 1 & 1 \\ r_{x,1} & r_{x,2} & r_{x,3} & r_{x,4} \\ -r_{y,1} & -r_{y,2} & -r_{y,3} & -r_{y,4} \\ -k_f & -k_f & k_f & k_f \end{bmatrix}$
控制输入：
$u_{indi} = \tilde{G}^+ \begin{bmatrix} T_d \\ \tau_d \end{bmatrix}$
失效后$\tilde{G}^+$不满秩，需求伪逆！
$\tilde{G}$为将混控矩阵G的第i列置为0

§6 路径规划（主讲：高飞）

6.1 常用地图形式

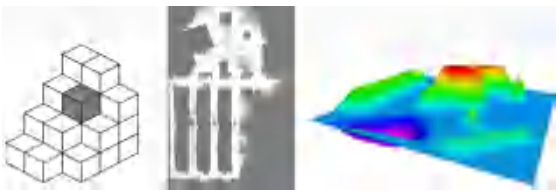
1. 栅格地图 (Grid Map)

(1) 占据栅格地图 (Occupancy Grid Map)

① 最为稠密

② 有结构的

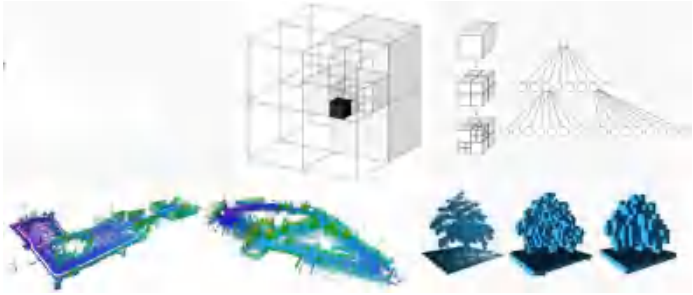
③ 直接索引查询



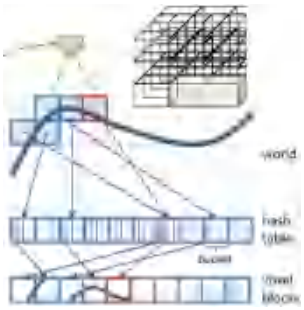
(2) 高程图 (Elevation Map)

(3) 拓展高程图 (Extended Elevation Map)

2. 八叉树地图 (Octomap)
   (1) 稀疏
   (2) 有结构的
   (3) 直接索引查询



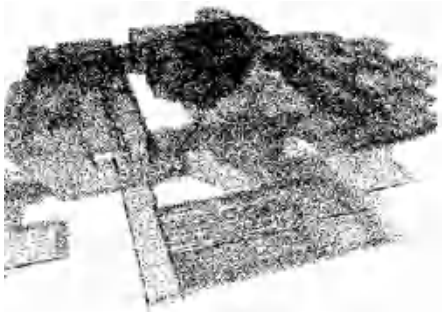
3. 体素哈希表 (Voxel Hashing)
   (1) 更稀疏
   (2) 有结构的
   (3) 直接索引查询

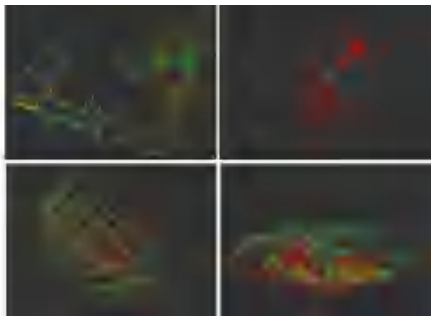


Voxel Hashing

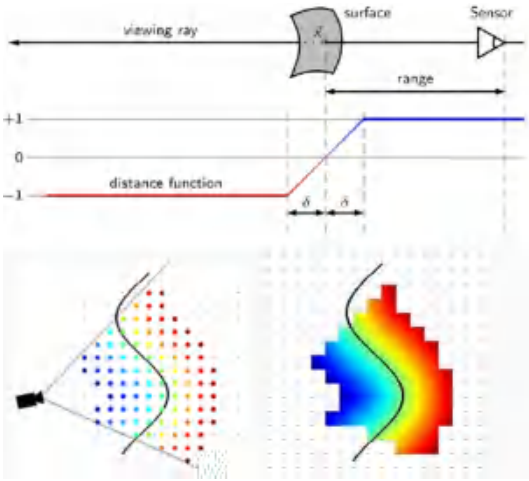
InfiniTAM

4. 点云地图 (Point Cloud)
   (1) 无顺序
   (2) 无法索引查询



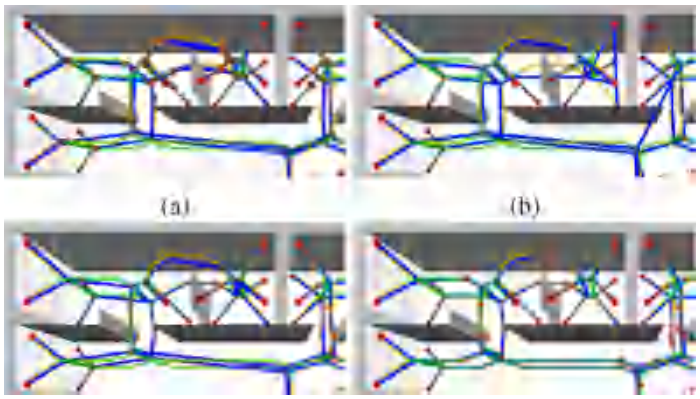


5. TSDF 地图 - 截断符号距离函数 TSDF (Truncated Signed Distance Functions)

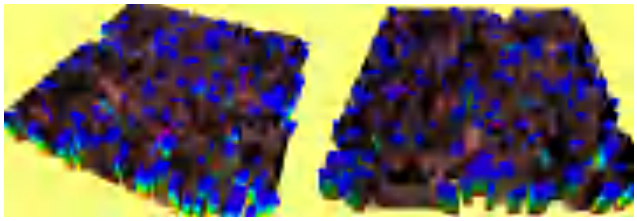


6. ESDF 地图 - 欧式符号距离函数 ESDF (Euclidean Signed Distance Functions)
   (1) 增量式更新，全局地图：*VoxBlox* 和 *FIESTA*
   (2) 分批更新，局部地图：*TRR's Local Map*
7. 特征地图
   (1) 以几何特征描述环境
   (2) 不利于导航规划

8. 沃罗诺伊图 (Voronoi Diagram)

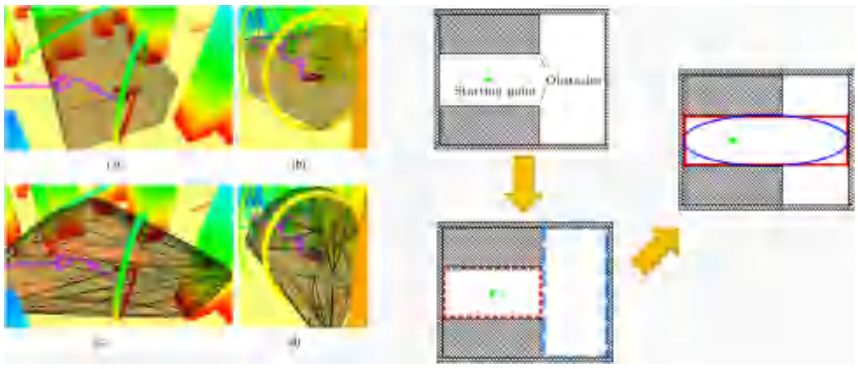


9. 自由空间路线图 (Freespace Roadmap)

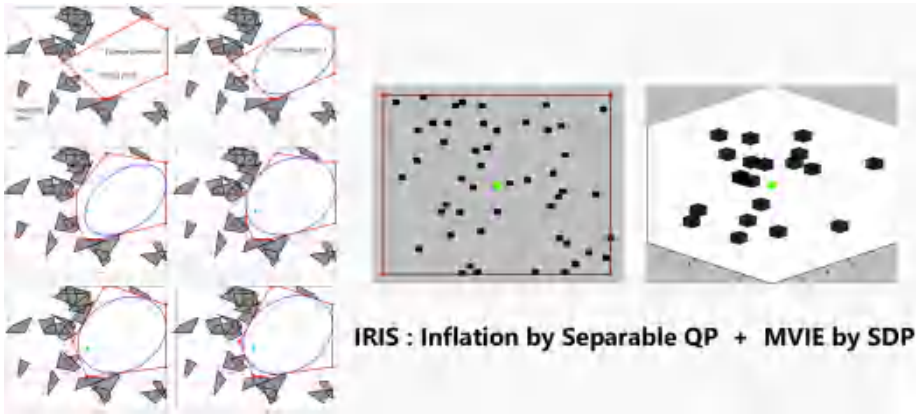


10. 飞行走廊 (Flight Corridor)

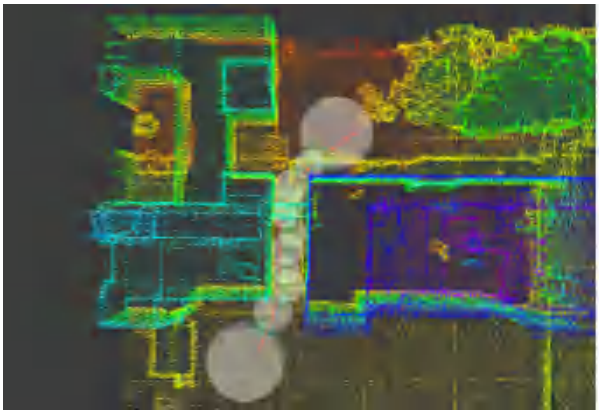
(1) 多面体膨胀方法



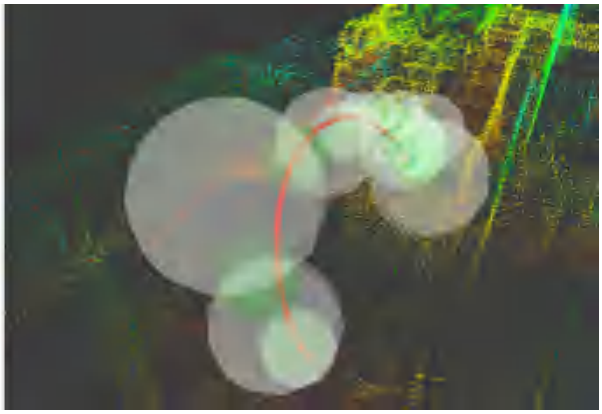
(2) 凸多面体迭代膨胀方法



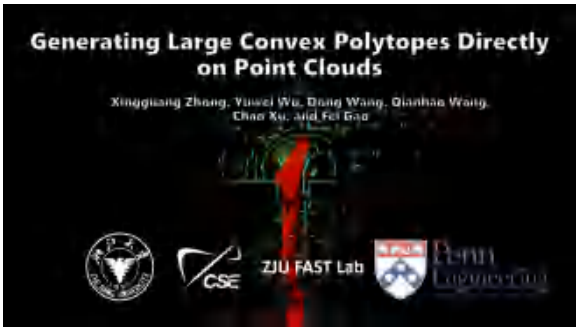
(3) 球形 KD-Tree 方法

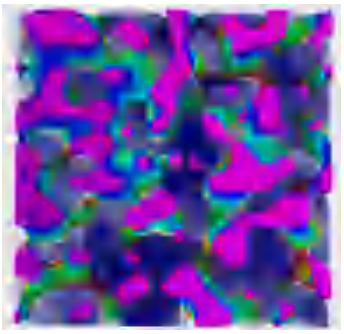


(4) 基于星凸多面体快速获取凸包



(5) 动态自由空间路线图



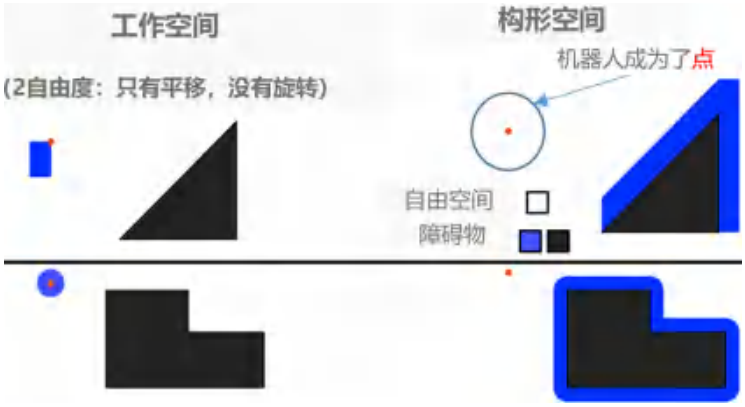


6.2 构形空间 (Configuration Space)

1. 基本概念
   (1) 机器人构形: 机器人所有的位置点。
   (2) 机器人自由度 (DOF): 用来表示机器人构形的最小实数坐标数。
   (3) 机器人构形空间 (C-space): 包含所有可能的机器人构形的 n 维空间。
   (4) 每个机器人姿态都是 C-Space 中的一个点。
2. 构形空间障碍物
   (1) 在工作空间中规划

   ① 机器人具有不同的形状和大小
   ② 碰撞检测需要知道机器人的几何信息——费时，难度大

(2) 在构形空间中规划



① 机器人表示为 C-space 中的一个点，如位（$\mathbb{R}^3$ 中的点），姿态（$SO(3)$ 中的点）。

② 障碍物需要在构形空间中表示（先于运动规划完成），称为构形空间障碍（C-obstacle）。

③ C-space = C-obstacle ∪ C-free

④ 路径规划就是在 C-free 中寻找起点 q_{start} 至终点 q_{goal} 的路径。

(3) 构形空间障碍物表示

① 在 C-space 中表示障碍物非常复杂。

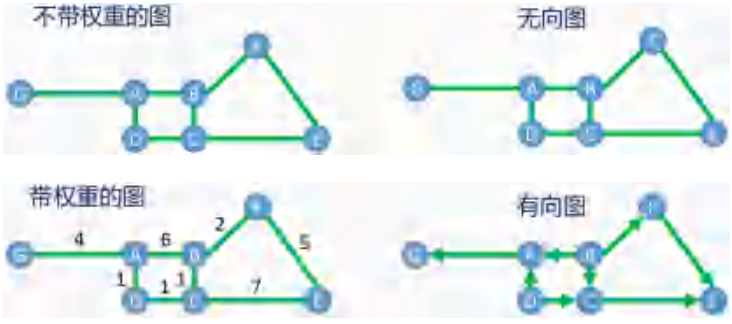
② 在实际中使用近似（但是更保守）的表示方法。

③ 如果保守地把机器人建模成一个半径为 δ_r 的球体，构建 C-space 可以把所有障碍物向各个方向膨胀 δ_r

6.3 基于搜索的路径规划

6.3.1 图的定义

1. 图 (Graphs) 由节点和边构成。
2. 类型



6.3.2 图搜索算法概述

1. 搜索过程开始于初始点 S

(1) 搜索过程会生成一个搜索树。

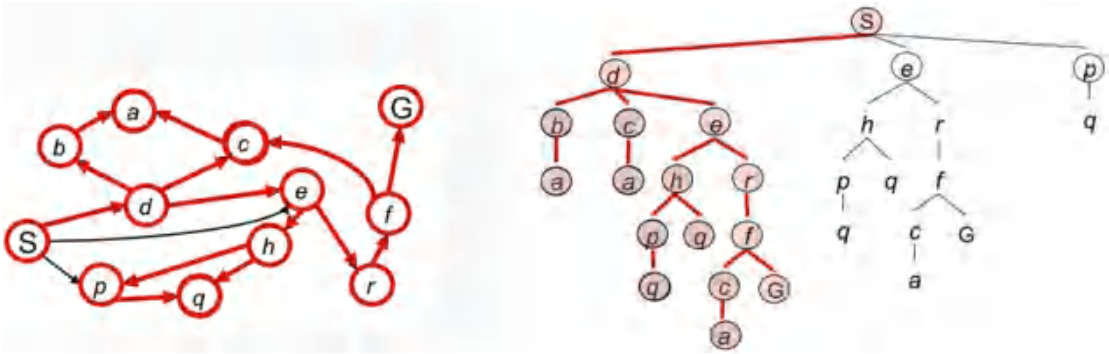
(2) 对树中节点逆向搜索可以得到一条路径。

(3) 很多情况下，构建图的整棵搜索树是不明智的（工作量大且低效）——尽快到达目标点才是我们的目的。

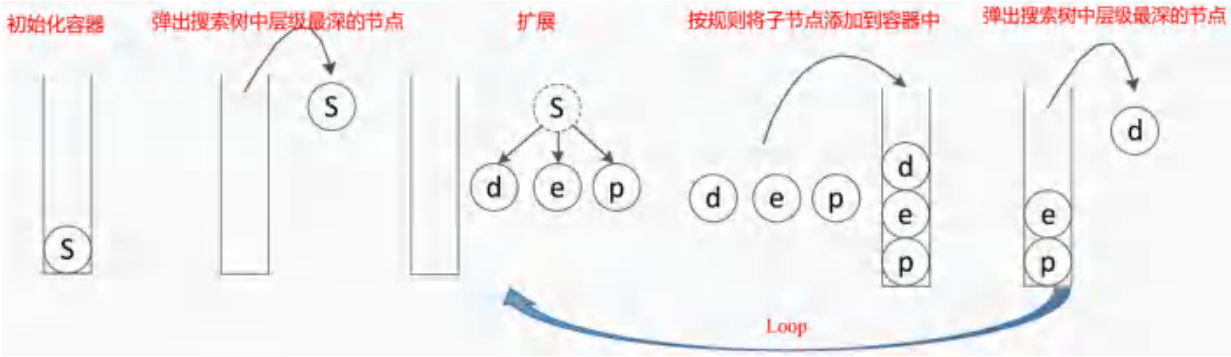
2. 基本步骤（general 的）
   (1) 维护一个容器来存储所有待访问的节点。
   (2) 容器初始化时只存在初始状态 X_s。
   (3) 开始循环
      - 删除 (Remove)：根据指定的规则从容器中移出一个节点。
      - 扩展 (Expansion)：得到该节点的所有邻节点。
      - 存入 (Push)：将这些邻节点存入容器。
   (4) 结束循环
3. 关键问题
   - Question 1：如何退出循环？
     - 例：容器空掉时退出循环。
   - Question 2：如果图中存在回环？
     - 禁止从容器中移出的节点再次进入容器。
   - Question 3：如何制定取出节点的规则使目标尽快被达到，且尽可能少的扩展结点？

6.3.3 图的遍历

1. 深度优先搜索 (Depth First Search, DFS)
   (1) 策略：移除/扩展搜索树中层级最深的节点。

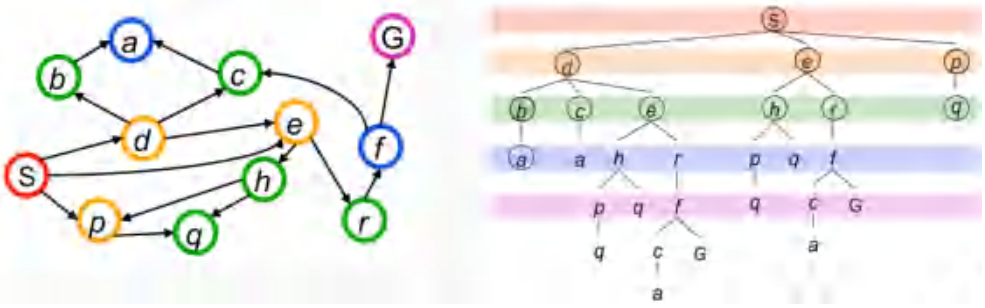


   (2) 实现：维护后进先出（LIFO）容器（即堆栈 Stack）。

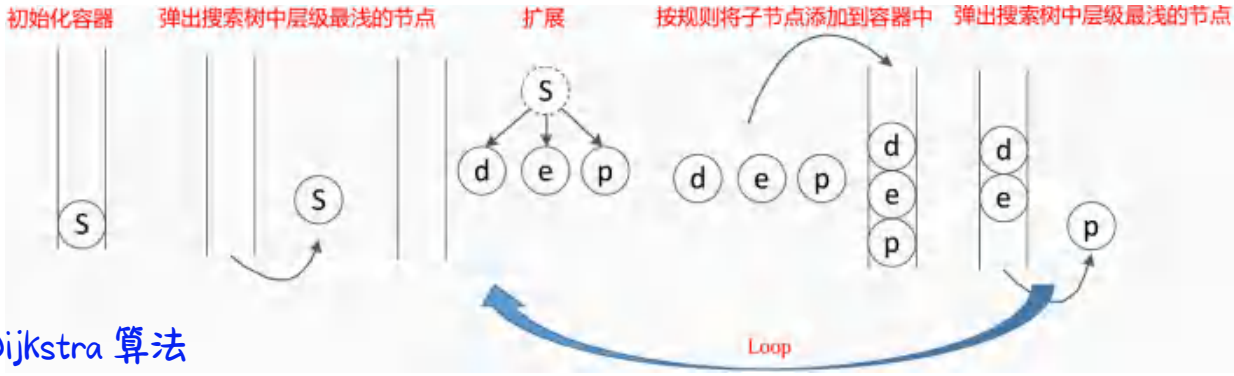


2. 广度优先搜索 (Breadth First Search, BFS)

(1) 策略：移除/扩展搜索树中层级最浅的节点

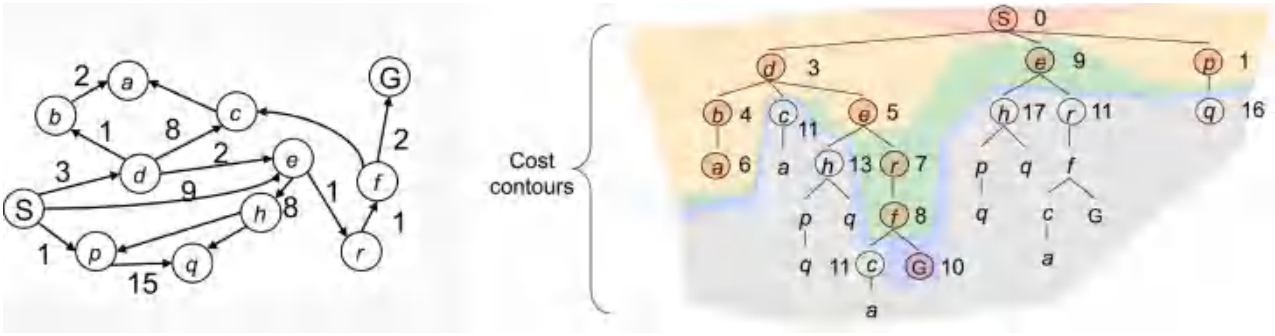


(2) 实现：维护先进先出（FIFO）容器（即队列 Queue）



6.3.4 Dijkstra 算法

1. 策略：每次取出容器中累计代价 $g(n)$ 最小的节点。

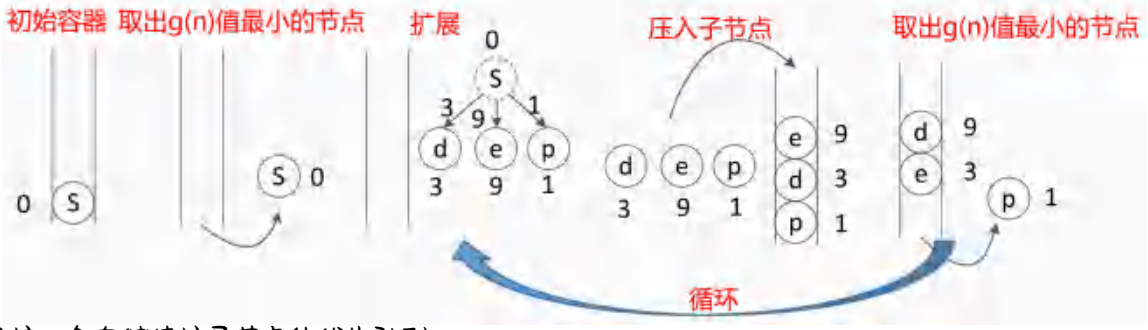


(1) $g(n)$：从初始点到点 n 的累计代价。

(2) 更新节点 n 的所有未拓展邻居 m 的累计成本 $g(m)$。

(3) 已扩展节点的累计代价应为到起始点的最短路径代价。

2. 算法流程



(1) 维护一个存储待扩展节点的优先队列。

(2) 根据初始状态 X_S 初始队列。

(3) 赋值 $g(X_S)=0$，且对于图中其他节点 $g(n)=\infty$。

(4) 循环

① 如果队列为空，返回 FALSE；退出循环。

② 从优先队列中移出最小 $g(n)$ 的节点 n。

③ 将节点 n 记作已扩展的节点。

④ 如果节点 n 是终点，返回 TRUE；退出循环。

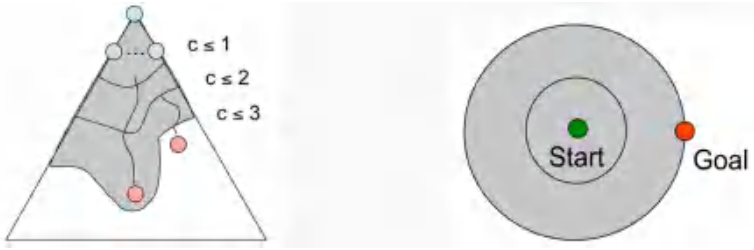
⑤ 对于所有未扩展的节点 n 的邻居节点 m。

a. 如果 $g(m)=\infty$，$g(m)=g(n)+C_{nm}$，将节点 “m” 压入队列。

b. 如果 $g(m)>g(n)+C_{nm}$，$g(m)=g(n)+C_{nm}$。

(5) 结束循环

3. 算法优缺点



(1) 优点：具有完备性和最优性。

(2) 缺点：扩展不具有方向性，没有利用到目标点的信息。

6.3.5 A* 算法：Dijkstra + 启发式函数

1. 核心思想

(1) 从起始状态到目标状态，经过节点 n 的最小估计代价为

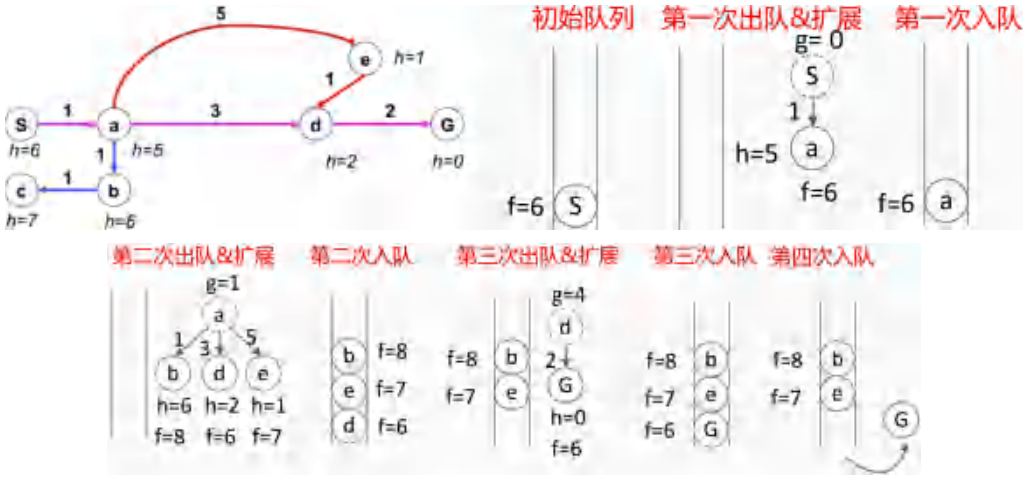
$$f(n)=g(n)+h(n) \tag{43}$$

① 累计代价 $g(n)$：从起始状态到节点 n 的最小估计代价。

② 启发式函数 $h(n)$：从节点到目标点的最小估计代价。

(2) 策略：取出具有最小 $f(n)$ 的节点。

2. 算法流程



(1) 维护一个存储待扩展节点的优先队列。

(2) 预先定义所有节点的启发式函数 $h(n)$。

(3) 根据初始状态 X_S 初始队列。

(4) 赋值 $g(X_S)=0$，且对于图中其他节点 $g(n)=\infty$。

(5) 循环

① 如果队列为空，返回 FALSE；退出循环。

② 从优先队列中移出最小 $f(n)=g(n)+h(n)$ 的节点 “n”。(和 Dijkstra 算法的唯一区别)

③ 将节点 “n” 记作已扩展的节点。

④ 如果节点 “n” 是终点，返回 TRUE；退出循环。

⑤ 对于所有未扩展的节点 n 的邻居节点 m。

a. 如果 $g(m)=\infty$，$g(m)=g(n)+C_{nm}$，将节点 “m” 压入队列。

b. 如果 $g(m)>g(n)+C_{nm}$，$g(m)=g(n)+C_{nm}$。

(6) 结束循环

3. 对比

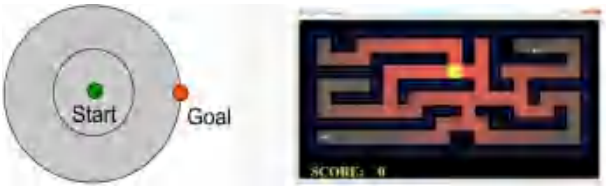


图 31 Dijkstra 算法朝各个方向探索

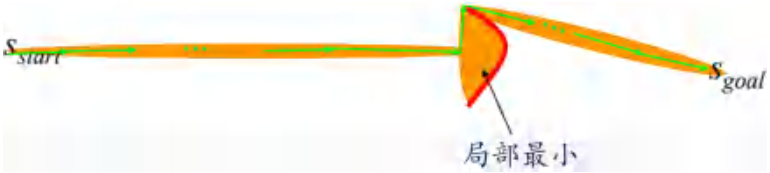
图 32 A* 算法主要朝着目标点方向探索

6.3.6 可采用的启发式函数

1. 启发式函数 h 是可采用的 (admissible)，如果对所有的节点 $h(n)\le h^*(n)$，其中 $h^*(n)$ 是从节点 n 到终点的真实最小距离。
2. 如果启发式函数可采用，那么 A* 搜索是最优的。
3. 实际中想出可采用的启发式函数是使用 A* 中的主要环节。

6.3.7 次优解

1. 使用过分估计的启发式函数会怎样？—— 次优路径，但更快。



2. 加权 A* (Weighted A*) 搜索：根据 $f=g+\varepsilon h, \varepsilon>1$ 拓展搜索，更倾向于靠近终点的状态。

(1) ε-suboptimal（次最优的）。

(2) 比 A* 快几个数量级。

3. 比较（贪婪优先搜索 vs. 加权 A* vs. A*）动画演示

$$f=a\cdot g+b\cdot h \tag{44}$$

(1) 最贪婪的情况：$a=0, b=1$。

(2) 可调整的贪婪 (Weighted A*)：$a=1, b=\varepsilon>1$。

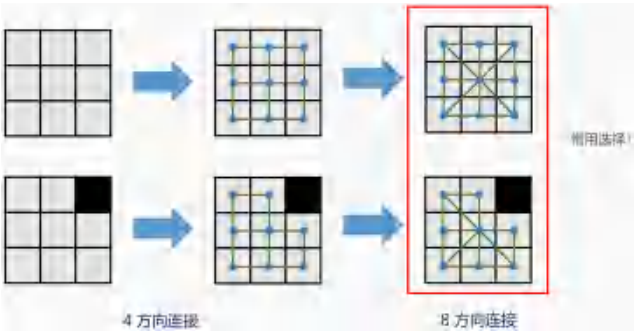
(3) 最优 (A*)：$a=1, b=1$。

(4) Dijkstra：$a=1, b=0$。

6.4 工程技巧

6.4.1 基于栅格的路径搜索

1. 怎么把栅格表示为图？—— 每个单元是一个节点，边连接相邻的单元。



2. 实现要点

(1) 创建一个稠密的栅格地图

(2) 连接存储在栅格地图之间的状态

(3) 通过栅格索引发现邻居

(4) 执行 A* 搜索

3. 优先级队列实现 (C++)

- `std::priority_queue`
- `std::make_heap`
- `std::multimap`

6.4.2 最优启发式函数

1. 常用启发式函数

(1) 欧氏距离 (Euclidean Distance) ☑

(2) 曼哈顿距离 (Manhattan Distance) ✗

(3) L_∞ 范数 ☑

(4) 零启发：$h=0$ ☑

2. 问题

(1) 它们是有用的，但没有一个是最好的选择，为什么？

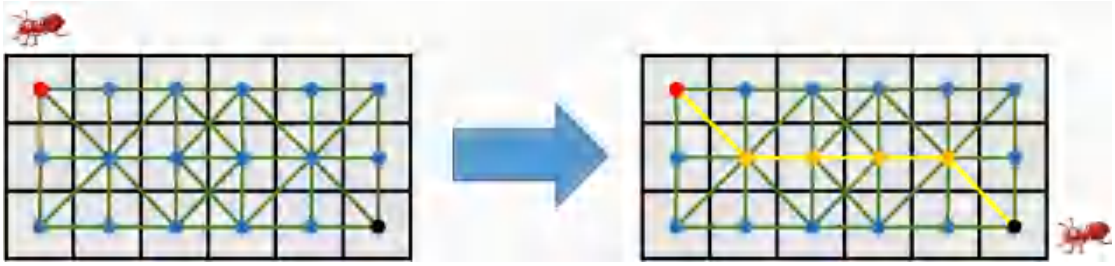
- 因为没有一个是紧的 (tight)，紧是指测量的真正的最短距离。

(2) 为什么欧氏距离在搜索时扩展了这么多节点？

- 因为欧氏距离远不是真正的理论最优解。

3. 对角启发式函数 (Diagonal Heuristic) —— 网格地图的理论最优解

(1) 幸运的是，网格地图是高度结构化的。



- 不需要搜索路径
- 具备理论上的闭式解

$$
\begin{aligned}
dx &= |\text{node.x} - \text{goal.x}| \\
dy &= |\text{node.y} - \text{goal.y}| \\
h &= (dx + dy) + \left(\sqrt{2} - 2\right) \cdot \min(dx, dy)
\end{aligned} \tag{45}
$$

(2) 对比

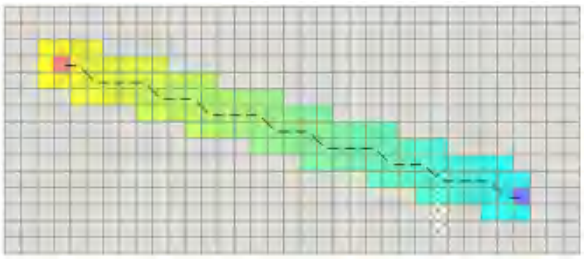


图 36 对角启发式函数

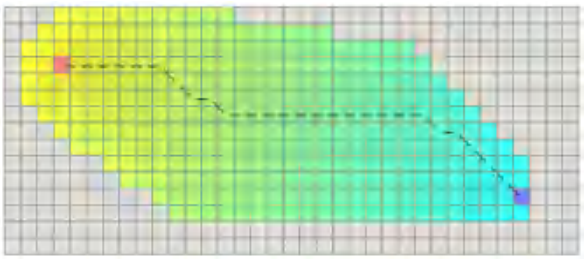


图 37 欧氏距离

6.4.3 打破对称性: Tie Breaker

1. 最优启发式函数仍存在的问题

   (1) 许多节点具有相同的 f 值。

   (2) 它们之间没有差异，这使得 A* 平等地扩展它们。

2. 方法: 修改 f 打破对称性，使相同 f 值不同 —— 轻微地放大 h。

$$
h = h \times (1.0 + p) \tag{46}
$$

   其中 $p < \dfrac{\text{minimum cost of one step}}{\text{expected maximum path cost}}$ 稍微打破了最优性。

3. 核心思想: 在相同 f 的节点中找到倾向性。

$$
\begin{aligned}
dx_1 &= \text{abs}(\text{node.x} - \text{goal.x}) \\
dy_1 &= \text{abs}(\text{node.y} - \text{goal.y}) \\
dx_2 &= \text{abs}(\text{start.x} - \text{goal.x}) \\
dy_2 &= \text{abs}(\text{start.y} - \text{goal.y}) \\
\text{cross} &= \text{abs}(dx_1 \times dy_2 - dx_2 \times dy_1) \\
h &= h + \text{cross} \times 0.001
\end{aligned} \tag{47}
$$

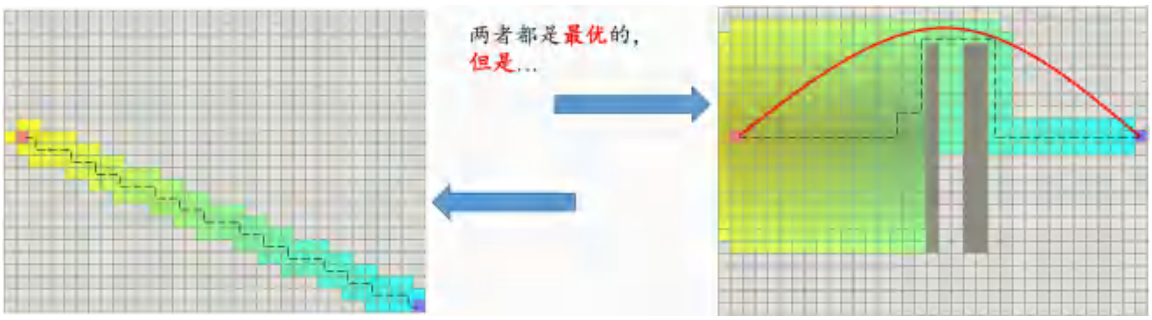
   (1) 当节点具有相同 f，比较他们的 h。

   (2) 将确定性随机数添加到启发式中。

   (3) 倾向从起点到目标的直线路径。

4. 回顾

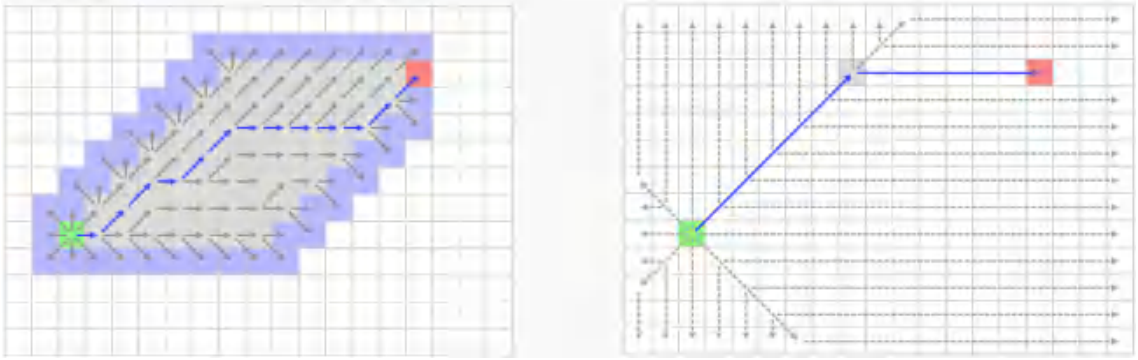
   - 如右下图中，当存在障碍物时，虚线所示路径是最短的路径，但不利于轨迹优化。



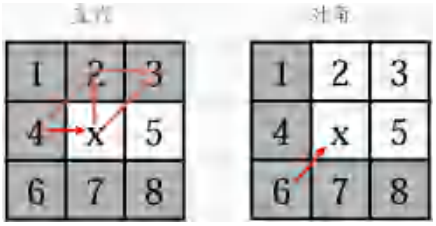
6.4.4 Jump Point Search (JPS) —— 一种系统性实现打破对称性的方法

> 提示
>
> 该章节高飞老师上课直接跳过了，略作了解即可，应该不会作为考察重点。详见论文。

1. 核心思想：找到对称性并打破它们。
   - A* 探索所有对称路径，JPS 选择一条路径。



2. Look Ahead 规则
   (1) 考虑：当前节点 x 和 x 的扩展方向。
   (2) 直线修剪 & 对角修剪

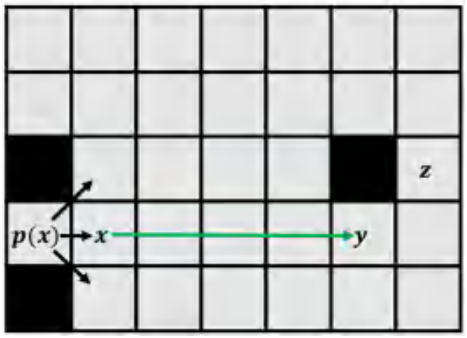


   - 灰色节点：较差的邻居，到它的路径没 x 更短，丢弃。
   - 白色节点：自然节点。
   - 在扩展搜索时，我们只需考虑自然节点和强制节点。

   (3) 强制节点：强制前往的节点。

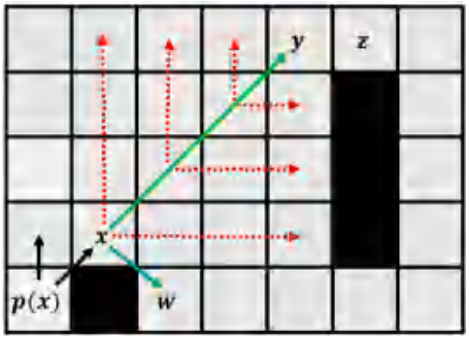
   - x 附近有障碍物。
   - 红色节点：强制节点（从 x 的父母到强制节点的路径被障碍物挡住了）。

3. Jumping 规则
   (1) 直线跳跃



   ① 递归地应用直线修剪规则，将 y 识别为 x 的跳点后继。

   (2) 对角跳跃



   ① 递归地应用对角修剪规则，将 y 识别为 x 的跳点后继。
   ② 在每个对角线步骤前，我们都先直线跳跃。

② 这个节点很有趣，因为它有一个邻居 z，除了访问 x 然后访问 y 的路径之外，无法以最佳方式到达。

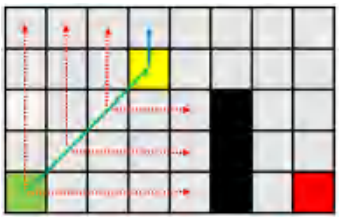
③ 只有当两个直线递归都不能识别跳跃点时，才能再次对角跨步。

④ 节点 w：x 的强制邻居，被正常扩展。

4. 算法流程：和A*完全相同。
5. 案例（A Visual Explanation of Jump Point Search）

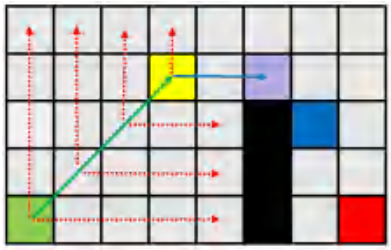
(1)

(2)



- 扩展一次对角移动
- 最后找到一个关键节点，将其添加到open list
- 从open list中弹出它（唯一一个）。
- 无需扩展，在障碍物处终点。

(3)

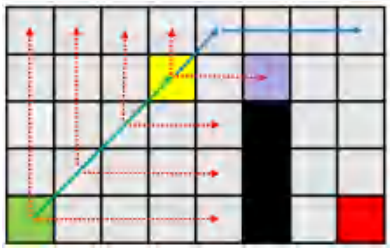


- 水平扩展，遇到具有强制邻居的节点。
- 将其添加到open list

Recall the rule

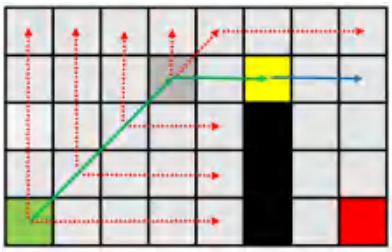
So we have

(4)



- 对角扩展，find nothing
- 完成当前节点的扩展。

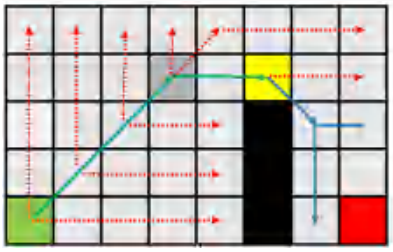
(5)



- 检查打开列表中的“新最佳”节点
- 水平扩展
- Finds nothing。

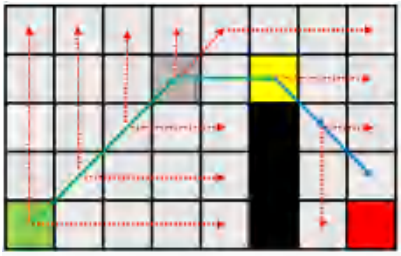
Remember the rule

(6)



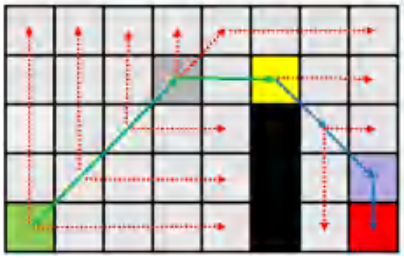
- 沿对角线移动
- 首先沿垂直和水平方向扩展

(7)



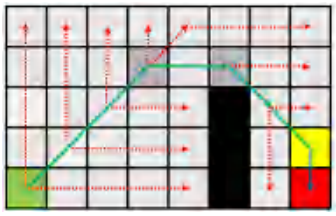
- Finds neighbors.
- 沿射线移动，

(8)



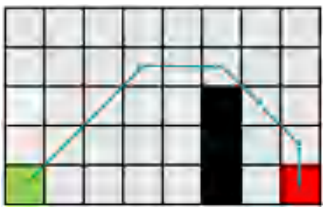
- 水平和垂直扩展
- 查找目标。与查找具有强制邻居的节点同样权重
- 将此节点添加到open list
- 完成当前节点的扩展（没有自然邻居）
- 将其从open list 中弹出

(9)



- 在open list 中的“new best”节点，
- 水平扩展（无位置），垂直扩展（找到目标），
- 结束。

(10)



6. 3D JPS

7. JPS 总是更好?

   (1) 大多数时候，尤其是在复杂的环境中，JPS 更好，但远不是“总是”。为什么?

   - JPS 减少了 Open List 中的节点数量，但增加了状态查询的数量。

   (2) JPS 的限制：仅适用于统一网格地图。

8. 实例介绍

   (1) Fast Marching in Distance Field

   □ 前端路径搜索：获得时间索引最小路径

   ● 模拟波传播

   $$|\nabla T(x)| = \frac{1}{f(x)}$$

   T 是到达时间的函数，x 是位置，$f(x)$ 表示不同位置 x的速度

   ● 速度场

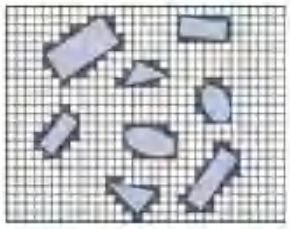
   $$f(d) = \begin{cases} v_m \cdot (\tanh(d-e)+1)/2, & 0 \le d \\ 0, & d < 0 \end{cases}$$

   v_m 为最大速度，d 为 ESDF 中位置 x 处的距离值

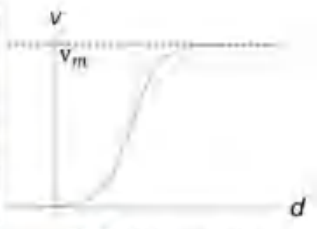
   ● 启发函数：到达目标的最短时间

   $$h(x) = d^*(x)/v_m$$

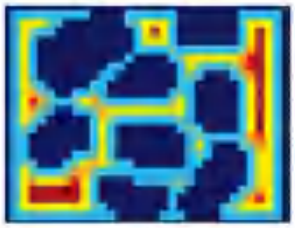
   $d^*(x)$ 表示 x 到目标点的欧氏距离



占据栅格地图

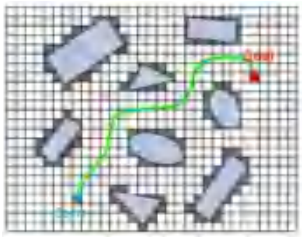


速度函数 $v=f(d)$

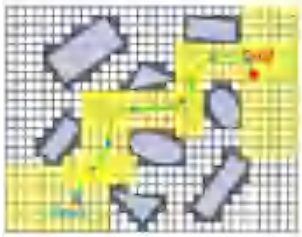


速度场

每个点到达的时间



前端搜索的路径　　飞行走廊初始化



飞行走廊修剪　　优化后的轨迹

□ 后端轨迹优化

轨迹表示：贝塞尔曲线

$$B_j(t) = c_j^0 b_n^0(t) + c_j^1 b_n^1(t) + \cdots + c_j^n b_n^n(t) = \sum_{i=0}^{n} c_j^i b_n^i(t),$$

路点约束　连续性约束　安全性约束　运动可行性约束

Online Safe Trajectory Generation For Quadrotors Using Fast Marching Method and Bernstein Basis Polynomial

Fei Gao, William Wu, Yi Lin, and Shaojie Shen

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

HKUST-DJI JOINT INNOVATION LABORATORY

(2) Elastic Tracker

□ 前端路径搜索：

- 对目标未来 T_p 时间内的轨迹 z_k 做出预测

$$\mathcal{T} = \{t_k \in [0, T_p] | t_k \to z_k, 0 < k \leq M_T\}$$

M_T 为预测的被跟踪目标的未来位置的个数，

$\mathcal{T}$ 为与目标未来每个位置相对应的时刻

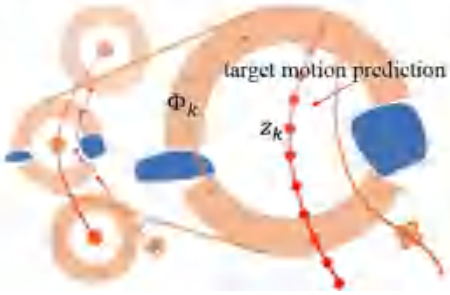
- 为每一个目标预测位置 z_k 定义一个无遮挡观察区域 Φ_k；
- 使用A*找到一条路径依次经过 $\Phi_1, \Phi_2, \ldots \Phi_k$ 区域

$$f^k(n) = g^k(n) + h^k(n)$$

$$h^k(n) = \sqrt{\{d_{xy}^k(n) - d_d\}^2 + \{d_z^k(n)\}^2}$$

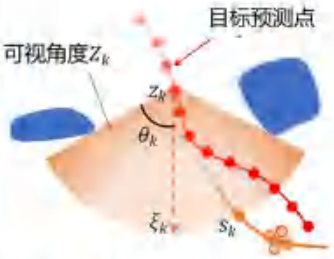
d_{xy}^k 和 d_z^k 是与目标预测位置点 z_k 的距离的水平和竖直方向分量；

d_d 是无人机和目标的期望距离；



□ 后端轨迹优化：

- 可视性 Visibility 约束：

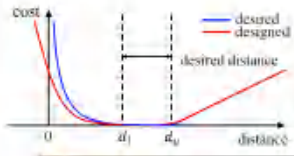


$$\mathcal{V}_k = \{x \in \mathbb{R}^3 \mid (x - z_k, \xi_k) \leq \theta_k\}$$

$\mathcal{V}_k$：扇形可视区域；z_k：目标预测位置

s_k：可视点；ξ_k：扇形的角平分线向量

- 距离约束：



$$\min_{p(t),T} \quad J_o = \rho T + \int_0^T \| p^{(3)}(t) \|^2 \, dt$$

$$\text{s.t.} \quad p^{[s-1]}(0) = \overline{p}_o, p^{[s-1]}(T) = \overline{p}_f$$

$$\| p^{(1)}(t) \| \leq v_m, \| p^{(2)}(t) \| \leq a_m, \forall t \in [0, T]$$

$$p(t) \in \mathcal{P}, \forall t \in [0, T]$$

$$p(t_k) \in \mathcal{V}_k, \forall t \in [0, T]$$

$$d_l \leq \| p(t_k) - z_k \| \leq d_u, \forall t_k \in \mathcal{T}$$

$$\mathcal{T} \geq T_p$$

6.5 基于采样的路径规划

【采样类最短路径算法】

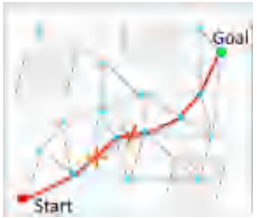
1. 高效地探索环境的拓扑结构，即可行区域的连接情况。
2. 不显示地构建构型空间及其边界。
3. 一般具有概率完备性。
4. 一般具有次优性或渐进最优性。
5. 可分为单查询（single-query）和多查询（multi-query）。

6.5.1 Probabilistic Road Map (PRM)

1. PRM 是一种多查询（multi-query）算法。
   (1) 将规划分为两个阶段: 构建阶段 和 搜索阶段。
   (2) 用相对少量的路标点和局部路径来构建一个连接图以得到可行区域的连接情况。
2. 构建阶段：构建一个表征环境连通情况的路标图。
   (1) 在工作空间采样 N 个点。
   (2) 删除和环境碰撞的点。
   (3) 连接近邻的节点。
   (4) 删除和环境碰撞的路径段。

3. 搜索阶段
   (1) 在构建出的路标连接图中搜索一条起点到终点的路径（使用 Dijkstra 或者 A* 算法）。
   (2) 路标图可近似看作栅格图。
   (3) 可利用路标图进行多次搜索（multi-query）。

4. 算法优劣
   (1) 优势：概率完备性。
   (2) 劣势
      ① 构建连接图没有专注于产生路径。
      ② 产生大量低效无用的采样和路径连接。
      ③ 低效。
5. 效率提升方法 - 懒惰 (Lazy) 的碰撞检查
   (1) 碰撞检查过程非常耗时，尤其是在复杂或高维环境中。
   (2) 采样点并生成线段，不考虑碰撞（Lazy）
   (3) 碰撞检查（如有必要）：
      - 在未进行碰撞检查的情况下生成的路线图上查找路径。
      - 如果路径是碰撞的，则删除相应的边和节点。



- 重新开始路径查找。

6.5.2 Rapidly-exploring Random Tree (RRT)

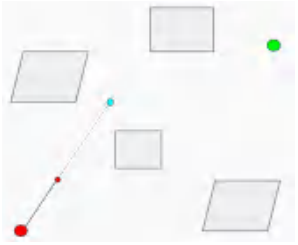
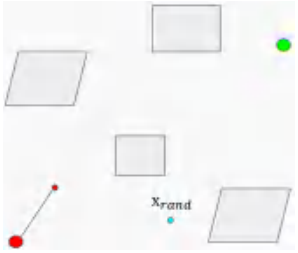
1. RRT 是一种单查询（single-query）算法。
   - 通过在工作空间采样节点来构建一棵从起点到终点的树，随着采样增加，树从起点向终点生长。
2. 算法流程

Algorithm 1: RRT Algorithm

Input: $\mathcal{M}, x_{init}, x_{goal}$
Result: A path Γ from x_{init} to x_{goal}
$\mathcal{T}$.init();
for $i = 1$ *to* n **do**
  $x_{rand} \leftarrow Sample(\mathcal{M})$;
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  $E_i \leftarrow Edge(x_{new}, x_{near})$;
  if *CollisionFree*$(\mathcal{M}, E_i)$ **then**
    $\mathcal{T}.addNode(x_{new})$;
    $\mathcal{T}.addEdge(E_i)$;
  if $x_{new} = x_{goal}$ **then**
    Success();

(1) 在可行空间随机采样。

$x_{rand} \leftarrow Sample(\mathcal{M})$;

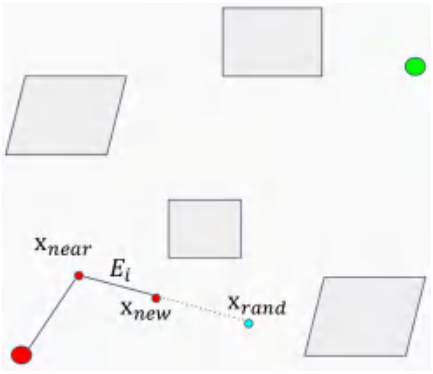



(2) 找到当前树中离采样点最近的树节点。

$x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;

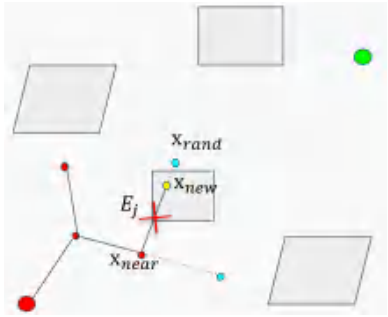
(3) 从最近的树节点“生长”出新的节点和树枝（路径）。

$x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
$E_i \leftarrow Edge(x_{new}, x_{near})$;



(4) 如果此路径没有和环境发生碰撞，则将此节点和路径加到树中。

$\mathcal{T}.addNode(x_{new})$;
$\mathcal{T}.addEdge(E_i)$;



(5) 重复 n 次采样，直到树生长到终点区域。

Success();

3. 算法优劣
   (1) 优势
      ① 致力于找到从起点到终点的路径

② 相比 PRM 更具目标导向性

(2) 劣势

① 产生的路径非最优

② 不够高效

4. 效率提升方法

(1) Kd-tree：快速最近邻搜索

(2) 双向 RRT

① 从起点和目标点分别搜索树

② 查找连接两棵树的路径

(3) Other variants: Spatial grid, hill climbing, etc.

6.5.3 Rapidly-exploring Random Tree* (RRT*)

1. RRT* 是什么？

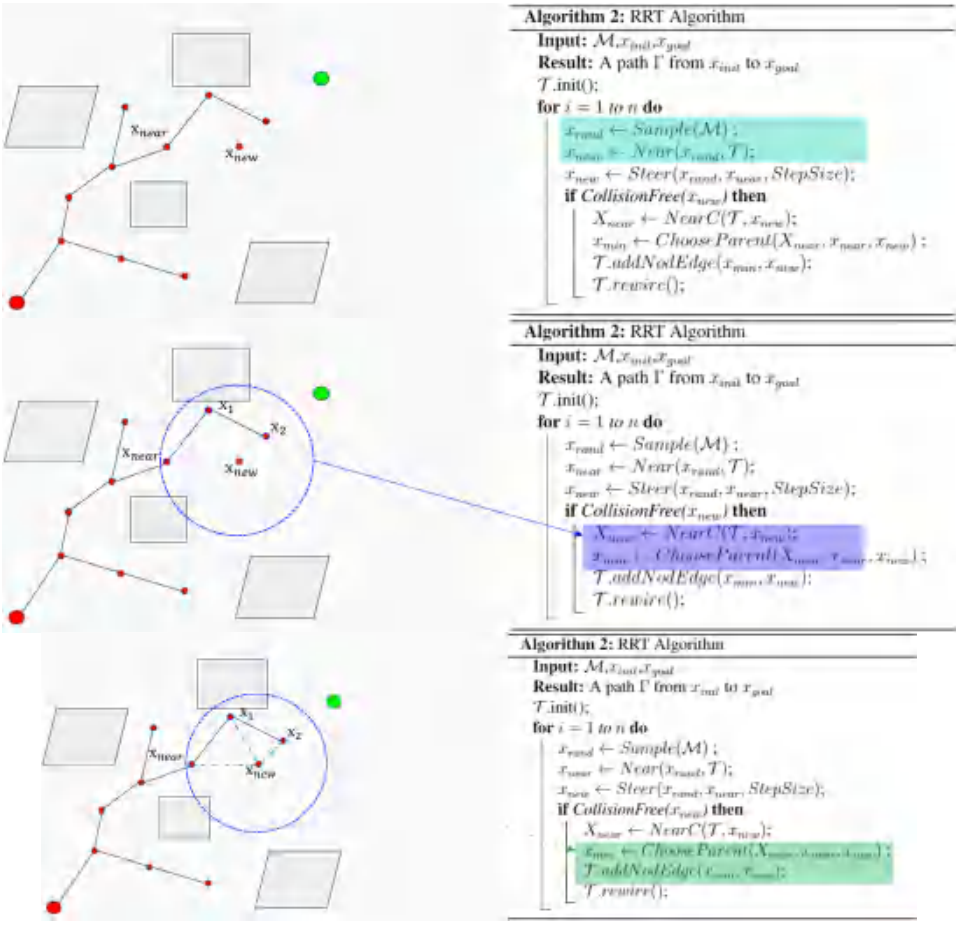
(1) 对 RRT 的改进。

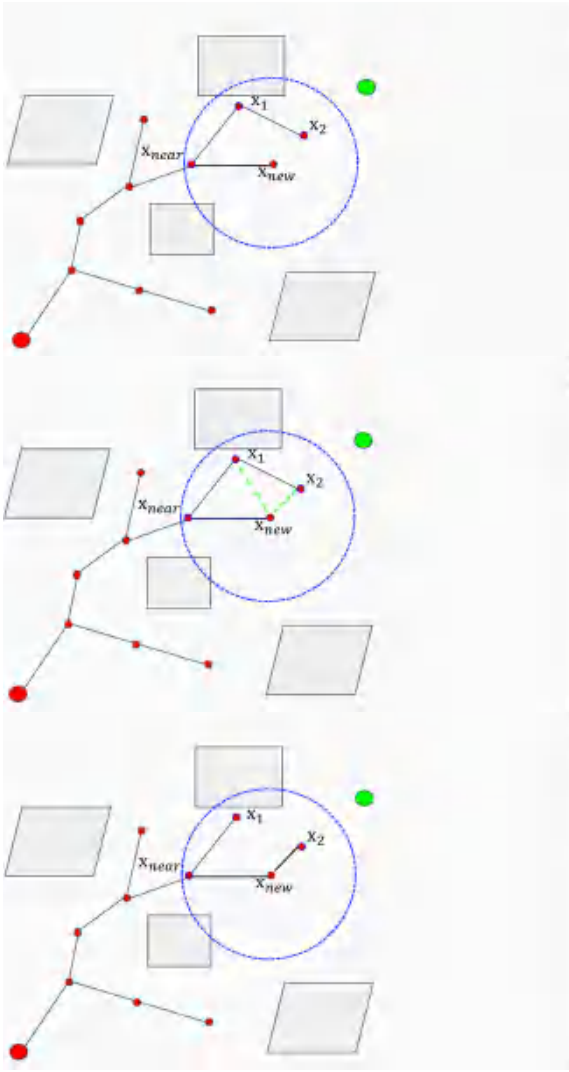
(2) 具备概率完备性和渐进最优性。

(3) 通过在工作空间采样节点来构建一棵从起点到终点的树。

(4) 随着采样增加，树从起点向终点生长，并且改进已有路径。

2. 算法流程





Algorithm 2: RRT Algorithm

Input: $\mathcal{M}, x_{init}, x_{goal}$
Result: A path Γ from x_{init} to x_{goal}
$\mathcal{T}$.init();
for $i = 1$ *to* n **do**
  $x_{rand} \leftarrow Sample(\mathcal{M})$;
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  if *CollisionFree*(x_{new}) **then**
    $X_{near} \leftarrow NearC(\mathcal{T}, x_{new})$;
    $x_{min} \leftarrow ChooseParent(X_{near}, x_{near}, x_{new})$;
    $\mathcal{T}.addNodEdge(x_{min}, x_{new})$;
    $\mathcal{T}.rewire()$;

Algorithm 2: RRT Algorithm

Input: $\mathcal{M}, x_{init}, x_{goal}$
Result: A path Γ from x_{init} to x_{goal}
$\mathcal{T}$.init();
for $i = 1$ *to* n **do**
  $x_{rand} \leftarrow Sample(\mathcal{M})$;
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  if *CollisionFree*(x_{new}) **then**
    $X_{near} \leftarrow NearC(\mathcal{T}, x_{new})$;
    $x_{min} \leftarrow ChooseParent(X_{near}, x_{near}, x_{new})$;
    $\mathcal{T}.addNodEdge(x_{min}, x_{new})$;
    $\mathcal{T}.rewire()$;

Algorithm 2: RRT Algorithm

Input: $\mathcal{M}, x_{init}, x_{goal}$
Result: A path Γ from x_{init} to x_{goal}
$\mathcal{T}$.init();
for $i = 1$ *to* n **do**
  $x_{rand} \leftarrow Sample(\mathcal{M})$;
  $x_{near} \leftarrow Near(x_{rand}, \mathcal{T})$;
  $x_{new} \leftarrow Steer(x_{rand}, x_{near}, StepSize)$;
  if *CollisionFree*(x_{new}) **then**
    $X_{near} \leftarrow NearC(\mathcal{T}, x_{new})$;
    $x_{min} \leftarrow ChooseParent(X_{near}, x_{near}, x_{new})$;
    $\mathcal{T}.addNodEdge(x_{min}, x_{new})$;
    $\mathcal{T}.rewire()$;

3. 改进之处

(1) 考虑邻近节点的历史路径长度，而非只是局部路径长度。

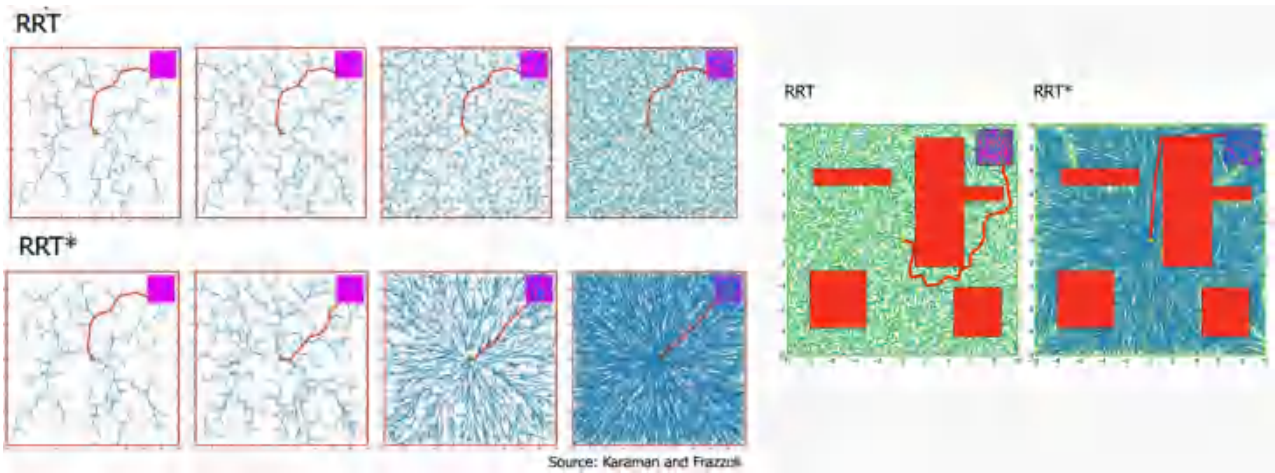
(2) 选择父节点时考虑多个邻近节点。

- 找到 x_{new} 的邻居节点集合，选择使得从起点到 x_{new} 路径代价最小的父节点。

(3) 重连接操作，改进局部最优解。

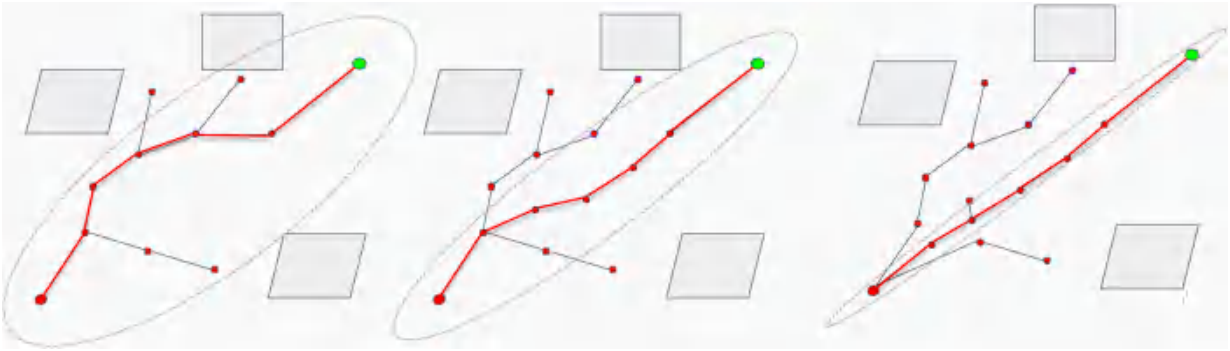
- 对于 x_{new} 的邻居节点，如果通过 x_{new} 到达该邻居的代价更小，则重新连接。

4. RRT vs. RRT*



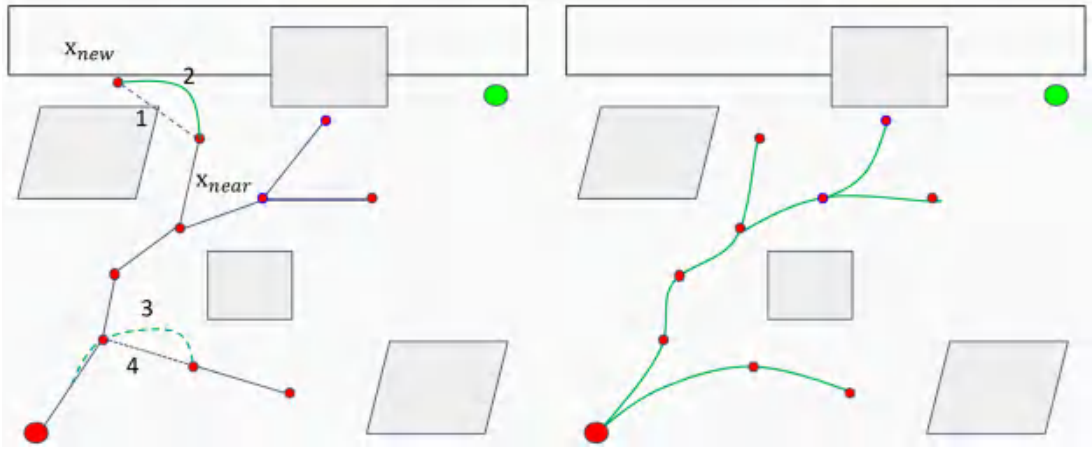
6.6 高级采样方法 (Advanced Sampling-based Methods)

6.6.1 Informed RRT*



1. 核心思想
   (1) 一旦找到初始解，将采样限制在一个椭圆区域内。
   (2) 椭圆的两个焦点分别是起点和终点。
   (3) 椭圆的长轴长度等于当前最优路径长度。
2. 优势：更快地收敛到最优解，减少不必要的采样。

6.6.2 Kinodynamic-RRT*



1. 核心思想
   (1) 更改函数以适应机器人导航中的运动或其他约束。
   (2) 考虑系统的动力学约束。
2. 关键改进
   - 使用动力学模型进行状态扩展。
   - 边不再是简单的直线段，而是满足动力学约束的轨迹。
3. 变种
   - Smart Kinodynamic RRT*: 引导采样 (Guided Sample)。
   - Bidirectional sampling: 双向采样。
   - Efficient Sampling-based Kinodynamic Planning。

6.6.3 Forward Spanning Tree (FST)

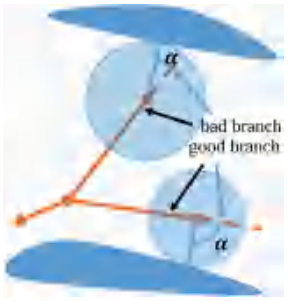
1. 采样

(1) 采样。

(2) 碰撞检测。

(3) 从近到远连接节点 -> 无需重连接。

2. 打分



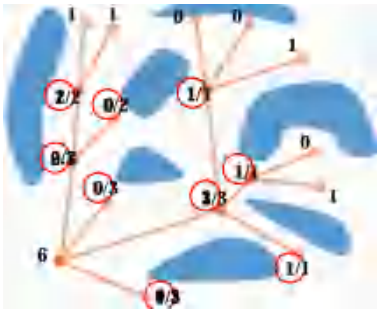
(1) 对子节点依据角度 α 进行 0-1 打分。

① 1 表示延伸到自由空间。

② 0 表示将会产生碰撞。

(2) 计算父节点得分。

3. 修剪

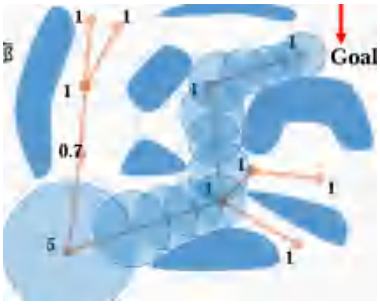


(1) 修剪权重 = 其得分 ÷ 其兄弟节点的最大得分。

(2) 广度优先遍历。

(3) 提取无碰撞空间框架。

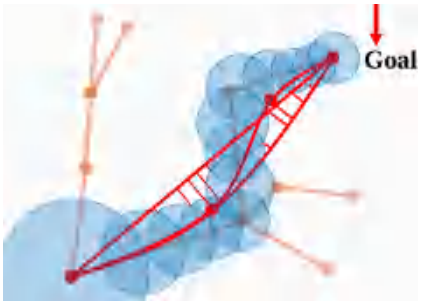
4. 生成安全飞行走廊



(1) 找到最短的分支。

(2) 生成球形安全飞行走廊。

5. 轨迹优化



(1) 迭代路径点插入。

(2) 迭代优化。

§7 运动动力学规划 (Kinodynamic Planning)（主讲：高飞）

7.1 什么是 Kinodynamic?

Kinodynamic = Kinematic + Dynamic

动力学规划问题是在同时受到运动学约束（如避开障碍物）和动力学约束（如速度、加速度和力的界限）的情况下综合机器人运动。运动动力学的解是从时间到广义力或加速度的映射。

—— *Kinodynamic Motion Planning, Bruce Donald, Patrick Xavier, John Canny, John Reif*

受微分约束（运动学模型） 和 上至力层面（动力学模型）

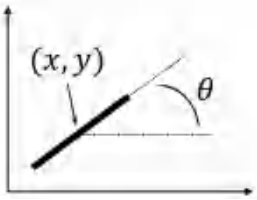
7.1.1 为什么需要 Kinodynamic Planning?

由于系统的微分约束，状态对之间的直线连接通常不是有效的轨迹。

1. 问题：为什么还需要动力学规划？
   (1) 回顾传统的规划流程：$\xrightarrow{\text{任务}}$ 路径规划 $\rightarrow$ 轨迹优化 $\xrightarrow{\text{轨迹}}$
   (2) 传统方法的局限性：
      ① 路径规划忽略动力学约束
      ② 轨迹优化可能无解或效率低
      ③ 分离的两步法可能导致不可行路径
2. 典型例子
   (1) 无人车：转弯半径约束、速度限制
   (2) 四旋翼：姿态约束、加速度限制

7.1.2 简单的例子

1. 独轮车模型与差速模型
   (1) 独轮车 (Unicycle)



① 系统模型：

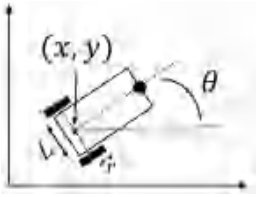
$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}\cos\theta\\ \sin\theta\\ 0\end{pmatrix}v+\begin{pmatrix}0\\ 0\\ 1\end{pmatrix}\omega \tag{48}$$

② 约束：$|v|\le v_{\max},\quad |\omega|\le \omega_{\max}$ (49)

③ 状态：(x,y,θ) (50)

④ 控制输入：(v,ω) (51)

   (2) 差速驱动 (Differential Drive)



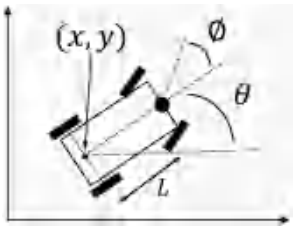
① 系统模型：

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}\frac{r}{2}(u_l+u_r)\cos\theta\\ \frac{r}{2}(u_l+u_r)\sin\theta\\ \frac{r}{L(u_r-u_l)}\end{pmatrix} \tag{52}$$

② 约束：$|u_l|\le u_{l,\max},|u_r|\le u_{r,\max}$ (53)

其中 $v=\frac{r}{2}(u_l+u_r),\quad \omega=\frac{r}{L(u_r-u_l)}$，$u_l$ 为左轮速度，u_r 为右轮速度。

2. 简化小车模型



(1) 系统模型：

$$\begin{pmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{pmatrix}=\begin{pmatrix}v\cos\theta\\ v\sin\theta\\ \frac{r}{L}\tan\varphi\end{pmatrix} \tag{54}$$

(2) 状态：(x,y,θ) (55)

(3) 控制输入：(v,φ) (56)

   (4) 三种车辆模型：
      ① Simple car model：$|v|\le v_{\max},\quad |\varphi|\le\varphi_{\max}<\frac{\pi}{2}$。
      ② Reeds & Shepp's car (bang-bang 控制)：$v\in\{-v_{\max},v_{\max}\},\quad |\varphi|\le\varphi_{\max}<\frac{\pi}{2}$。
      ③ Dubin's car：$v=v_{\max},\quad |\varphi|\le\varphi_{\max}<\frac{\pi}{2}$。

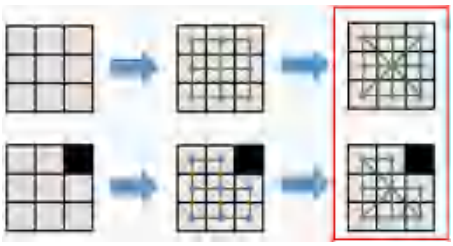
7.2 状态栅格规划 (State Lattice Planning)

7.2.1 基本思路

1. 基于搜索的路径生成。
2. 假设机器人是一个质点已经不再适用，我们需要一个具有可行运动连接 (feasible motion connections) 的图。
3. 我们手动构建一个所有边都可由机器人执行的图。

(1) 正向：在控制空间采样。

【举例】



① 实际上，这是在控制空间进行离散化采样。

② 我们假设机器人可以向 4/8 个方向运动。

(2) 反向：在状态空间采样。

【举例】

① 实际上，这是在状态空间进行离散化采样。

② 这里的状态在 $\mathbb{R}^2$ 上，只有位置 (x, y) 被考虑。

这是所有运动动力学规划的基本动机，State lattice planning 是最直接的一种。

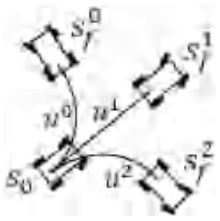
7.2.2 算法流程

1. 维护一个优先队列来存储所有要扩展的节点。
2. 对所有节点的启发式函数 $h(n)$ 都是提前定义好的。
3. 优先队列被起始状态 X_S 初始化。
4. 赋值 $g(X_S) = 0$，且对于图中其他节点 $g(n) = \infty$。
5. 循环
   (1) 如果队列为空，返回 FALSE；退出循环。
   (2) 从优先队列中移除 $f(n) = g(n) + h(n)$ 值最低的节点 n。
   (3) 将节点 n 记作被拓展过的。
   (4) 如果节点 n 是目标状态，返回 TRUE；退出循环。
   (5) 对于所有未被拓展过的节点 n 的邻居节点 m。
      ① 如果 $g(m) = \infty$，$g(m) = g(n) + C_{nm}$，将节点 “m” 压入队列。
      ② 如果 $g(m) > g(n) + C_{nm}$，$g(m) = g(n) + C_{nm}$。
6. 结束循环

7.2.3 控制空间采样 vs. 状态空间采样

对于一个机器人模型：$\dot{s} = f(s, u)$，机器人可以被微分驱动。我们有一个机器人的初始状态 s_0，通过下面的方法生成可行的局部运动。

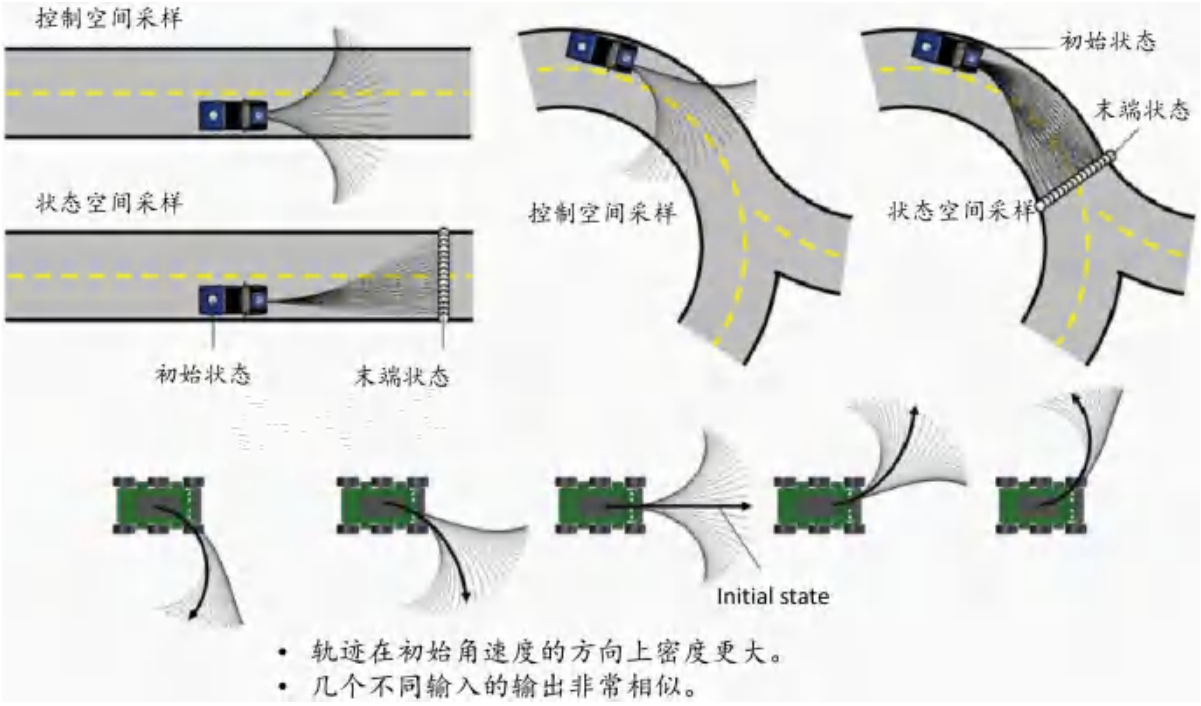
1. 选择一个输入 u，固定一个持续时间 T，前向模拟系统（数值积分）。



(1) 前向模拟
(2) 固定 u, T
(3) 没有任务引导
(4) 易于实现
(5) 低规划效率

2. 选择一个状态 s_f，找到两个状态 s_0 和 s_f 之间的连接（一段轨迹）。

(1) 反向计算
(2) 需要计算 u, T
(3) 较好的任务引导
(4) 难以实现
(5) 高规划效率



7.2.4 控制空间采样

1. 状态：$\boldsymbol{s} = (x \ y \ z \ \dot{x} \ \dot{y} \ \dot{z})^T$。
2. 输入：$\boldsymbol{u} = (\ddot{x} \ \ddot{y} \ \ddot{z})$。

> **提示**
>
> 这里的输入也可以放到状态中，然后输入变为三阶导。以此类推 …

3. 系统方程

$$\dot{\boldsymbol{s}} = \boldsymbol{A} \cdot \boldsymbol{s} + \boldsymbol{B} \cdot \boldsymbol{u} \tag{57}$$

其中 $\boldsymbol{A} = \begin{pmatrix} 0 & I_3 & 0 & \cdots & 0 \\ 0 & 0 & I_3 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & I_3 \\ 0 & \cdot & \cdots & 0 & 0 \end{pmatrix}$，$\boldsymbol{B} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I_3 \end{pmatrix}$。

(1) 高阶积分器（幂零系统）

$$s(t) = \underbrace{e^{\boldsymbol{A}t}}_{F(t)} s_0 + \underbrace{\left(\int_0^t e^{\boldsymbol{A}(t-\sigma)} \boldsymbol{B} \, \mathrm{d}\sigma \right)}_{G(t)} u_m \tag{58}$$

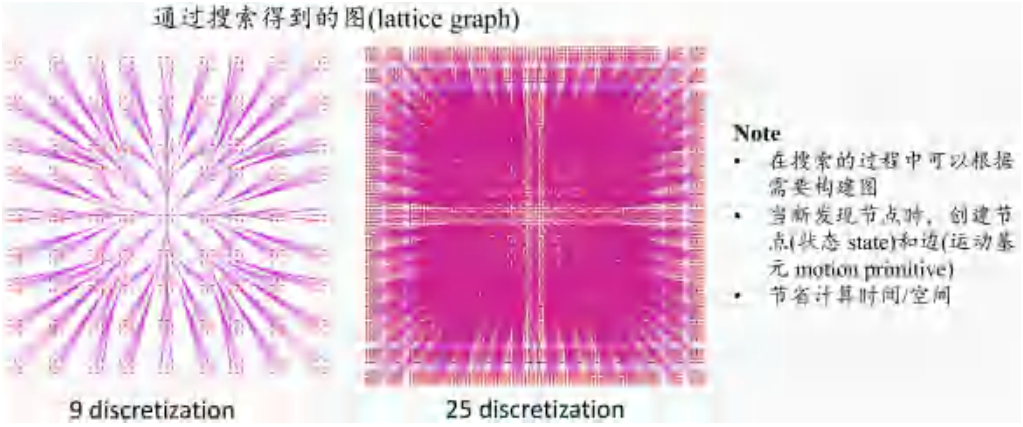
(2) 状态转移矩阵（幂零性质）

如果矩阵 $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ 是幂零的（如 $\boldsymbol{A}^n = 0$），那么 $e^{\boldsymbol{A}t}$ 有闭式解：

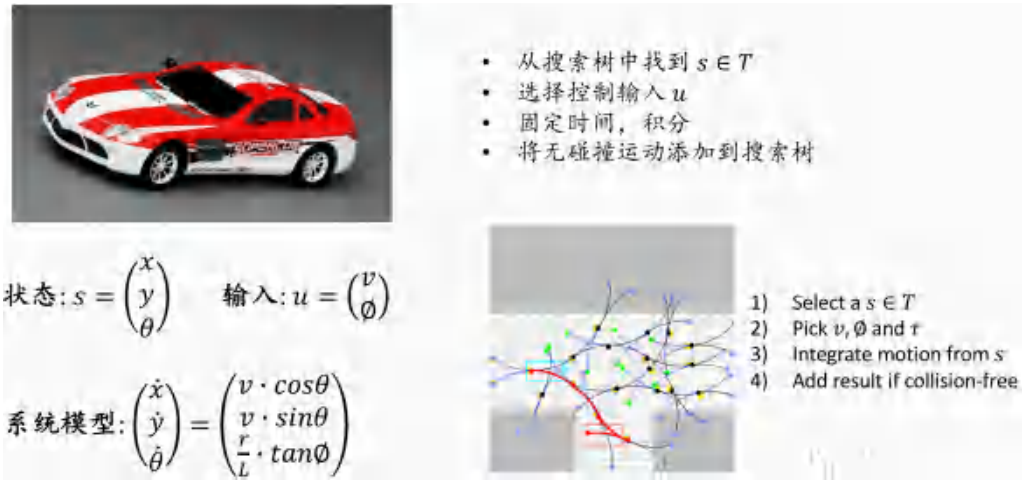
$$e^{\boldsymbol{A}t} = \boldsymbol{I} + \frac{\boldsymbol{A}t}{1!} + \frac{(\boldsymbol{A}t)^2}{2!} + \frac{(\boldsymbol{A}t)^3}{3!} + \cdots + \frac{(\boldsymbol{A}t)^k}{k!} \tag{59}$$

4. 举例

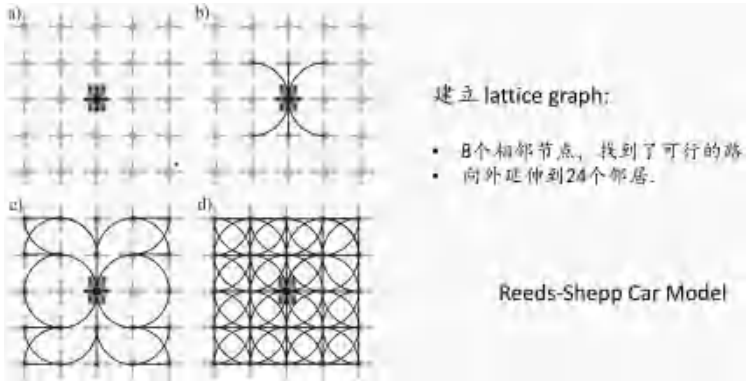
(1)

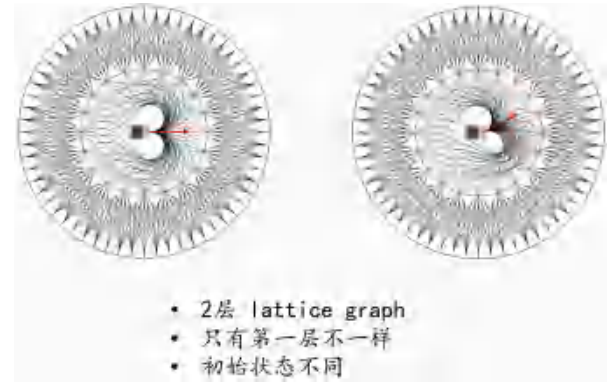


(2)



7.2.5 状态空间采样

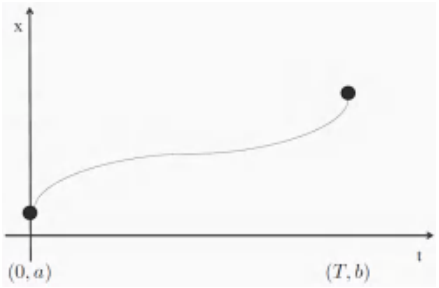




7.3 边值问题 (Boundary Value Problem, BVP)

7.3.1 问题定义

1. 目标：设计一条轨迹 $x(t)$ 使得：$x(0)=a, x(T)=b$。
2. 特点
   (1) BVP 是状态空间采样规划的基础。
   (2) 没有通用的解决方案。
   (3) 通常需要复杂的数值优化。



7.3.2 变分法求解最优控制问题（上课跳了，应该不作考察重点）

1. 系统状态方程：
$$\dot{\boldsymbol{s}}(t)=f(\boldsymbol{s},\boldsymbol{u},t),\quad \boldsymbol{s}(t_0)=\boldsymbol{s}_0 \tag{60}$$
2. 性能指标泛函：
$$J=\varphi\left[\boldsymbol{x}(\boldsymbol{s}_f),t_f\right]+\int_{t_0}^{t_f}L(\boldsymbol{s},\boldsymbol{u},t)\,\mathrm{d}t \tag{61}$$
3. 末端约束：
$$\psi\left(\boldsymbol{s}(t_f),t_f\right)=0 \tag{62}$$
4. 求解：最优控制 $\boldsymbol{u}^*(t)$ 与对应的最优轨迹 $\boldsymbol{s}^*(t)$。
5. 末端时刻 t_f 固定的情形
   (1) 引入拉格朗日乘子，将有约束问题转换为无约束问题，构造广义泛函

$$\begin{aligned}J_a&=\varphi\left[\boldsymbol{s}(t_f)\right]+\underbrace{\gamma^T\psi\left[\boldsymbol{s}(t_f)\right]}_{\text{末态约束}}+\int_{t_0}^{t_f}\left\{L(\boldsymbol{s},\boldsymbol{u},t)+\underbrace{\lambda^T(t)[f(\boldsymbol{s},\boldsymbol{u},t)-\dot{\boldsymbol{s}}(t)]}_{\text{系统状态方程}}\right\}\mathrm{d}t\\&=\varphi\left[\boldsymbol{s}(t_f)\right]+\gamma^T\psi\left[\boldsymbol{s}(t_f)\right]+\int_{t_0}^{t_f}\left[\underbrace{H(\boldsymbol{s},\boldsymbol{u},\lambda,t)}_{\text{Hamilton 函数}}-\lambda^T(t)\dot{\boldsymbol{s}}(t)\right]\mathrm{d}t\end{aligned} \tag{63}$$

   (2) 定义 Hamilton 函数

$$H(\boldsymbol{s},\boldsymbol{u},\lambda,t)=L(\boldsymbol{s},\boldsymbol{u},t)+\lambda^T(t)f(\boldsymbol{s},\boldsymbol{u},t) \tag{64}$$

   (3) 泛函极值的必要条件：泛函一次变分为零。

> $J[x]$ 是在线性赋范空间 $\mathbb{R}^n$ 上某个开子集 D 定义的可微泛函，在 $x=x^*$ 处达到极值。那么泛函 $J[x]$ 在 $x=x^*$ 处必有
> $$\delta J[\boldsymbol{x}^*,\delta\boldsymbol{x}]=0 \tag{65}$$

$$\begin{aligned}J_a&=\varphi\left[\boldsymbol{s}(t_f)\right]+\gamma^T\psi\left[\boldsymbol{s}(t_f)\right]+\int_{t_0}^{t_f}\left[H(\boldsymbol{s},\boldsymbol{u},\lambda,t)-\lambda^T(t)\dot{\boldsymbol{s}}(t)\right]\mathrm{d}t\\&=\varphi\left[\boldsymbol{s}(t_f)\right]+\gamma^T\psi\left[\boldsymbol{s}(t_f)\right]+\int_{t_0}^{t_f}H(\boldsymbol{s},\boldsymbol{u},\lambda,t)\,\mathrm{d}t-\int_{t_0}^{t_f}\lambda^T(t)\dot{\boldsymbol{s}}(t)\,\mathrm{d}t\\&=\varphi\left[\boldsymbol{s}(t_f)\right]+\gamma^T\psi\left[\boldsymbol{s}(t_f)\right]+\int_{t_0}^{t_f}H(\boldsymbol{s},\boldsymbol{u},\lambda,t)\,\mathrm{d}t+\int_{t_0}^{t_f}\dot{\lambda}^T(t)\boldsymbol{s}(t)\,\mathrm{d}t-\lambda^T(t)\boldsymbol{s}(t)\,|_{t_0}^{t_f}\\&=\varphi\left[\boldsymbol{s}(t_f)\right]+\gamma^T\psi\left[\boldsymbol{s}(t_f)\right]+\int_{t_0}^{t_f}\left[H(\boldsymbol{s},\boldsymbol{u},\lambda,t)+\dot{\lambda}^T(t)\boldsymbol{s}(t)\right]\mathrm{d}t-\lambda^T\left(t_f\right)\boldsymbol{s}\left(t_f\right)+\lambda^T(t_0)\boldsymbol{s}(t_0)\end{aligned} \tag{66}$$

分部积分：$\int_{t_0}^{t_f} \lambda^T(t)\dot{s}(t)\,\mathrm{d}t = \int_{t_0}^{t_f} \dot{\lambda}^T(t)s(t)\,\mathrm{d}t - \lambda^T(t)s(t)\,|_{t_0}^{t_f}$

(4) 现在取变分 δJ_a，注意：① **变分是一种线性算子**。② **重要公式**：$\delta\frac{\mathrm{d}J}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\delta J$。

$$J_a = \varphi\left[s(t_f)\right] + \gamma^T\psi\left[s(t_f)\right] - \lambda^T(t_f)s(t_f) + \lambda^T(t_0)\underbrace{s(t_0)}_{\text{常数变分为 }0} + \int_{t_0}^{t_f}\left[H(s,u,\lambda,t) + \dot{\lambda}^T(t)s(t)\right]\mathrm{d}t$$

$$0 = \delta J_a = \delta s^T(t_f)\left[\frac{\partial\varphi}{\partial s(t_f)} + \frac{\partial\psi^T}{\partial s(t_f)} - \lambda(t_f)\right] + \int_{t_0}^{t_f}\left[\left(\frac{\partial H}{\partial s} + \dot{\lambda}\right)^T\delta s + \left(\frac{\partial H}{\partial u}\right)^T\delta u\right]\mathrm{d}t \tag{67}$$

(5) 拉格朗日乘子向量不变分

由 式 67 和 式 64 可得

① 横截条件：
$$\lambda(t_f) = \frac{\partial\varphi}{\partial s(t_f)} + \frac{\partial\psi^T}{\partial s(t_f)} \tag{68}$$

② 正则方程：
$$\dot{\lambda}(t) = -\frac{\partial H}{\partial s} \tag{69}$$

③ 极值条件：
$$\frac{\partial H}{\partial u} = 0 \tag{70}$$

④ 正则方程：
$$\frac{\partial H}{\partial\lambda} = f(s,u,t) = \dot{s}(t) \tag{71}$$

求解最优控制问题的步骤

(1) 根据问题建立 Hamilton 函数。

$$H(s,u,\lambda,t) = L(s,u,t) + \lambda^T(t)f(s,u,t)$$

(2) 利用极值条件求得最优控制 $u^*(t)$。

$$\frac{\partial H}{\partial u} = 0 \longrightarrow u^*(t) = u^*(s,\lambda,t)$$

(3) 将 $u^*(t)$ 代入正则方程，得到正则方程组（两点边值问题）。

$$\begin{cases}\dot{s}(t) = \dfrac{\partial H}{\partial\lambda} = f(s,u^*,t)\\ \dot{\lambda}(t) = -\dfrac{\partial H}{\partial s}\end{cases}\quad\text{with}\quad\begin{cases}s(t_0) = s_0\\ \lambda(t_f) = \dfrac{\partial\varphi}{\partial s(t_f)} + \dfrac{\partial\psi^T}{\partial s(t_f)}\end{cases}$$

(4) 求解正则方程组，得到最优轨迹 $s^*(t)$ 和协态变量 $\lambda^*(t)$。

(5) 将 $s^*(t)$ 和 $\lambda^*(t)$ 代入第 2 步的表达式，得到最优控制 $u^*(t)$。

6. **控制也存在约束**的情形

$\frac{\partial H}{\partial u} = 0$ 不成立，此时 $u^* \in \Omega$。横截条件与边界条件、正则方程不变，但是极值条件变为

$$H(x^*,u^*,\lambda) = \min_{u\in\Omega} H(x^*,u,\lambda) \tag{72}$$

7.3.3 Pontryagin 最小值原理（上课跳了，应该不作考察重点）

1. 优化目标：

$$J = \underbrace{h(\boldsymbol{s}(T))}_{\text{最终状态}} + \underbrace{\int_0^T g(\boldsymbol{s}(t), \boldsymbol{u}(t)) \cdot \mathrm{d}t}_{\text{转移代价}} \tag{73}$$

2. 哈密顿量和状态：$$H(\boldsymbol{s}, \boldsymbol{u}, \lambda) = g(\boldsymbol{s}, \boldsymbol{u}) + \lambda^T f(\boldsymbol{s}, \boldsymbol{u}), \quad \lambda = (\lambda_1, \lambda_2, \lambda_3) \tag{74}$$

3. 定义：s^* 为最优状态，u^*为最优输入。

4. 最小值原理

$$\dot{\boldsymbol{s}}^*(t) = f(\boldsymbol{s}^*(t), \boldsymbol{u}^*(t)), \quad \text{given: } \boldsymbol{s}^*(0) = \boldsymbol{s}(0) \tag{75}$$

$\lambda(t)$ 是如下公式的解：$$\dot{\lambda}(t) = -\nabla_{\boldsymbol{s}} H(\boldsymbol{s}^*(t), \boldsymbol{u}^*(t), \lambda(t)) \tag{76}$$

边值条件为 $$\lambda(T) = \nabla h(\boldsymbol{s}^*(T)) \tag{77}$$

最优控制输入为 $$\boldsymbol{u}^*(t) = \arg\min_{\boldsymbol{u}(t)} H(\boldsymbol{s}^*(t), \boldsymbol{u}(t), \lambda(t)) \tag{78}$$

7.4 最优边值问题 (Optimal Boundary Value Problem, OBVP)

7.4.1 问题建模（以四旋翼为例）

1. 目标：最小化 jerk 平方的积分。

$$J_\Sigma = \sum_{k=1}^{3} J_k, \quad J_k = \frac{1}{T} \int_0^T j_{k(t)}^2 \, \mathrm{d}t \tag{79}$$

2. 状态与输入：$$\boldsymbol{s}_k = (p_k, v_k, a_k), \quad \boldsymbol{u}_k = j_k \tag{80}$$

3. 系统模型：$$\dot{\boldsymbol{s}}_k = f_s(\boldsymbol{s}_k, \boldsymbol{u}_k) = (v_k, a_k, j_k) \tag{81}$$

7.4.2 求解

1. 通过 Pontryagin 最小值原理（式 75 ~ 式 78），引入协态

$$\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3) \tag{82}$$

2. 定义 Hamilton 函数：$$H(\boldsymbol{s}, \boldsymbol{u}, \boldsymbol{\lambda}) = \frac{1}{T} j^2 + \boldsymbol{\lambda}^T f_s(\boldsymbol{s}, \boldsymbol{u}) = \frac{1}{T} j^2 + \lambda_1 v + \lambda_2 a + \lambda_3 j \tag{83}$$

$$\dot{\lambda}(t) = -\nabla_s H(s^*(t), u^*(t), \lambda(t)) = (0, -\lambda_1, -\lambda_2)$$

3. 协态方程与解：

$$\lambda(t) = \frac{1}{T} \begin{pmatrix} -2\alpha \\ 2\alpha t + 2\beta \\ -\alpha t^2 - 2\beta t - 2\gamma \end{pmatrix} \tag{84}$$

4. 最优控制输入：$$u^*(t) = j^*(t) = \arg\min_{j(t)} H(s^*(t), j(t), \lambda(t)) = \frac{1}{2}\alpha t^2 + \beta t + \gamma \tag{85}$$

5. 最优状态轨迹：

$$s^*(t) = \begin{pmatrix} \frac{\alpha}{120} t^5 + \frac{\beta}{24} t^4 + \frac{\gamma}{6} t^3 + \frac{a_0}{2} t^2 + v_0 t + p_0 \\ \frac{\alpha}{24} t^4 + \frac{\beta}{6} t^3 + \frac{\gamma}{2} t^2 + a_0 t + v_0 \\ \frac{\alpha}{6} t^3 + \frac{\beta}{2} t^2 + \gamma t + a_0 \end{pmatrix} \tag{86}$$

   - 初始条件：$s(0) = (p_0, v_0, a_0)$。

6. 将最优输入代入目标函数，可得代价函数

$$J=\gamma^2+\beta\gamma T+\frac{1}{3}\beta^2T^2+\frac{1}{3}\alpha\gamma T^2+\frac{1}{4}\alpha\beta T^3+\frac{1}{20}\alpha^2T^4 \tag{87}$$

7. α,β,γ 被如下公式求解：

$$\begin{pmatrix}\frac{1}{120}T^5 & \frac{1}{24}T^4 & \frac{1}{6}T^3\\ \frac{1}{24}T^4 & \frac{1}{6}T^3 & \frac{1}{2}T^2\\ \frac{1}{6}T^3 & \frac{1}{2}T^2 & T\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\\ \gamma\end{pmatrix}=\begin{pmatrix}\Delta p\\ \Delta v\\ \Delta a\end{pmatrix}=\begin{pmatrix}p_f-p_0-v_0T-\frac{1}{2}a_0T^2\\ v_f-v_0-a_0T\\ a_f-a_0\end{pmatrix} \tag{88}$$

$$\Rightarrow\begin{pmatrix}\alpha\\ \beta\\ \gamma\end{pmatrix}=\frac{1}{T^5}\begin{pmatrix}720 & -360T & 60T^2\\ -360T & 168T^2 & -24T^3\\ 60T^2 & -24T^3 & 3T^4\end{pmatrix}\begin{pmatrix}\Delta p\\ \Delta v\\ \Delta a\end{pmatrix}$$

> 📌 注意
>
> (1) 这个推导适用于固定的最终状态：$s(T)=(p_f,v_f,a_f)$。
>
> - 对于最终状态固定的问题，$h(s(T))=\begin{cases}0, & \text{if } s=s(T)\\ \infty, & \text{otherwise}\end{cases}$ 不可微，故放弃这个条件，用已知的 $x(T)$ 来直接求解未知变量。
>
> (2) 对于（部分）自由最终条件的问题，given $s_i(T),i\in I$，我们对其他协态有边界条件
>
> $$\lambda_j(T)=\frac{\partial h(s^*(T))}{\partial s_j},\text{for } j\neq i \tag{89}$$
>
> 然后求解即可。

7.4.3 最优边值-中值问题 (Boundary-intermediate Value Problem, BIVP)

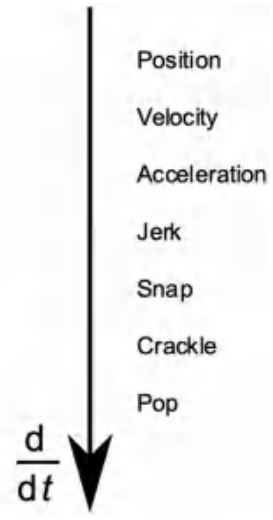
1. 边值问题 (BVP)

(1) BVP 经常被应用于基于运动基元的轨迹规划方法。

(2) 我们的最优条件显示，解为

① 一个 5 阶多项式，当 $s=3$ (minimum jerk)。

② 一个 7 阶多项式，当 $s=4$ (minimum snap)。



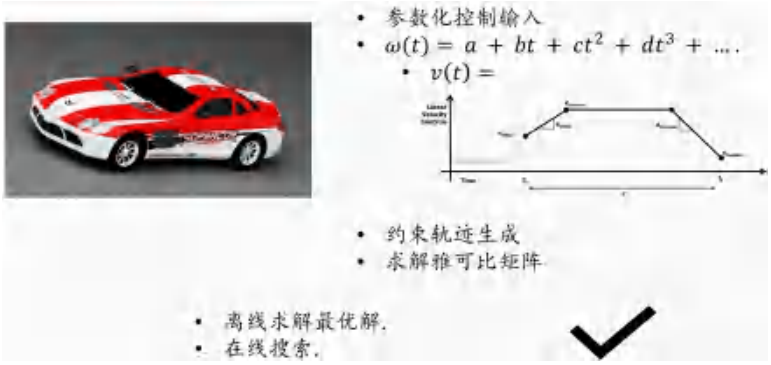
2. 最优边值-中值问题 (BIVP)

我们的最优条件表明，只有中间路径点条件情况下

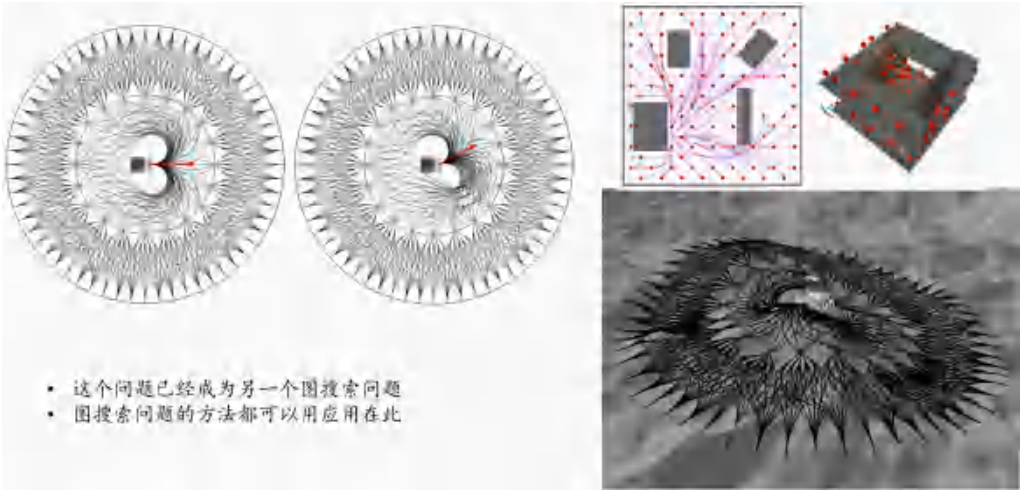
(1) 分段 5 阶多项式轨迹有着连续的 snap，当 $s=3$ (minimum jerk trajectory)。

(2) 分段 7 阶多项式轨迹有着连续的 pop，当 $s=4$ (minimum snap trajectory)。

7.5 实例分析



1. 图搜索问题

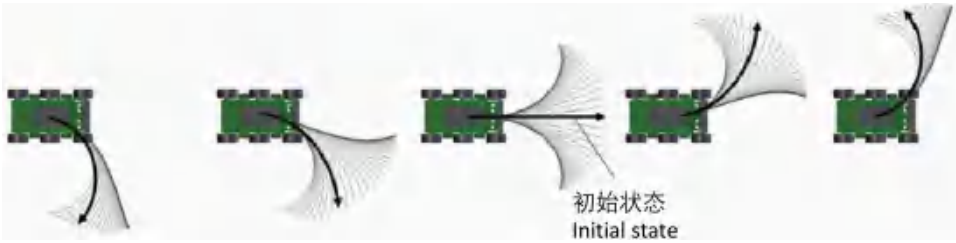


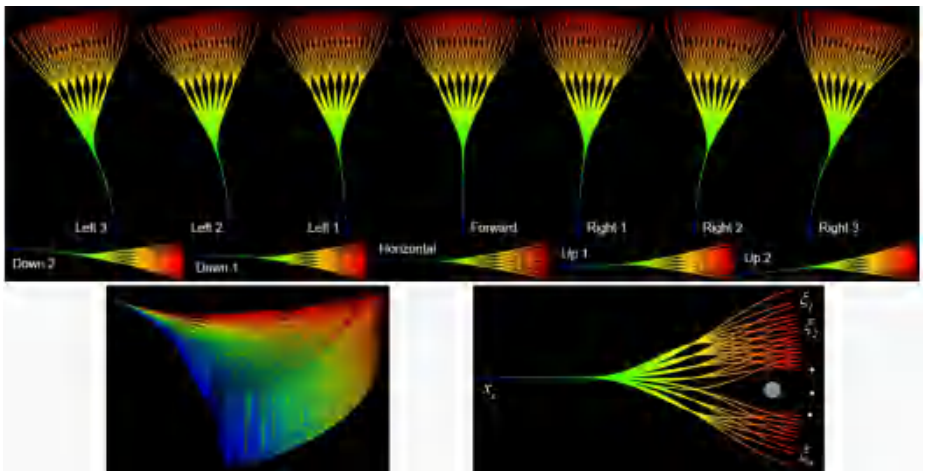
2. 轨迹库 (Trajectory library)

(1) 单层 lattice planning 是一种常见的用于局部避障的选择。

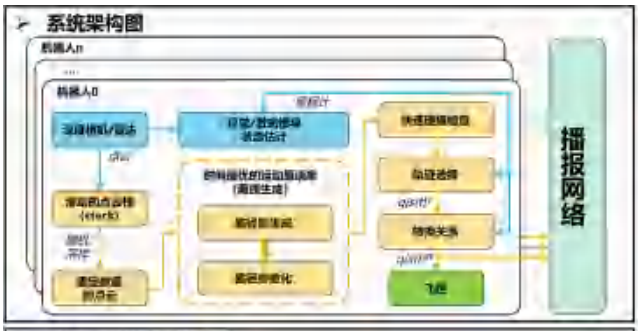
(2) 不需要图搜索，只需要轨迹选择。

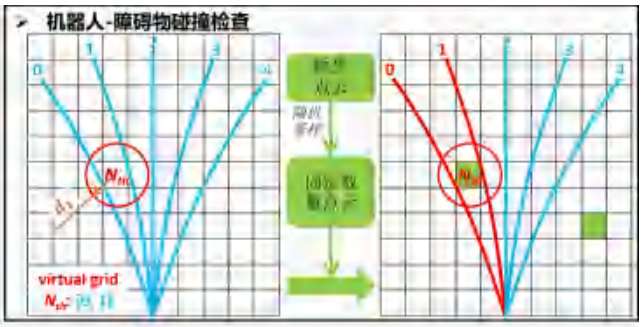
(3) 基于多项成本函数（碰撞风险、信息获取、舒适度、能量等）对每个轨迹进行评级。

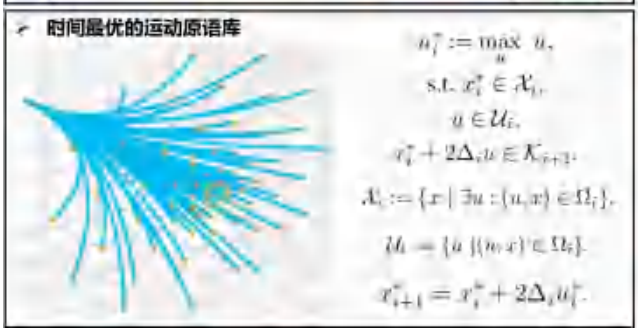


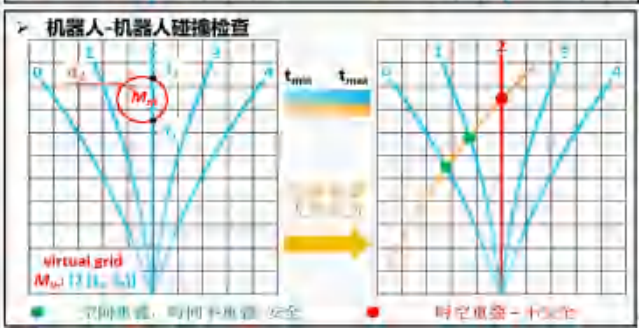


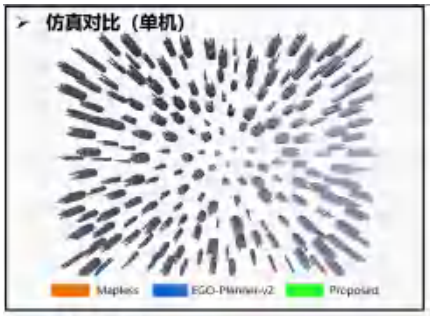
3. 超轻量级规划无人机规划

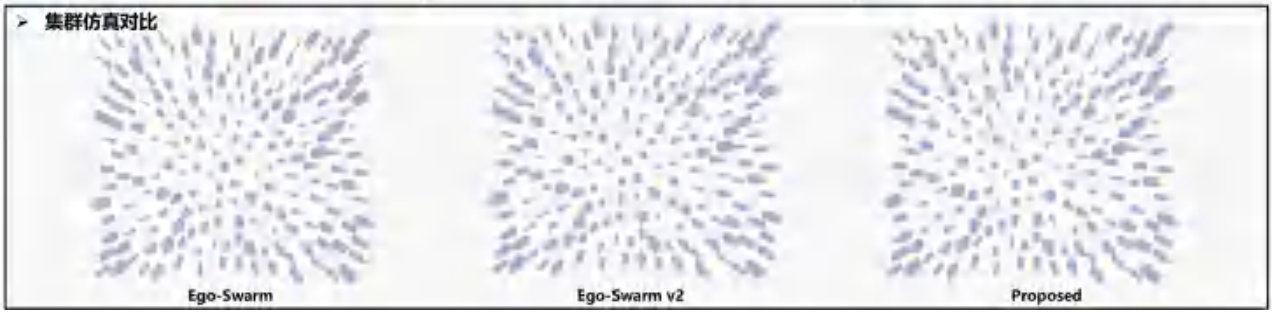


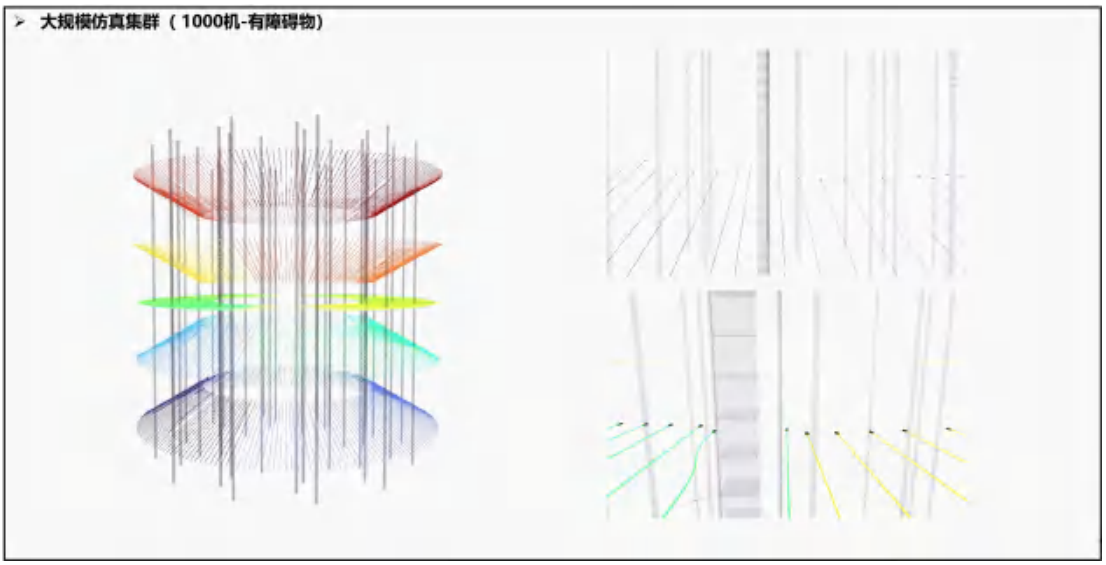




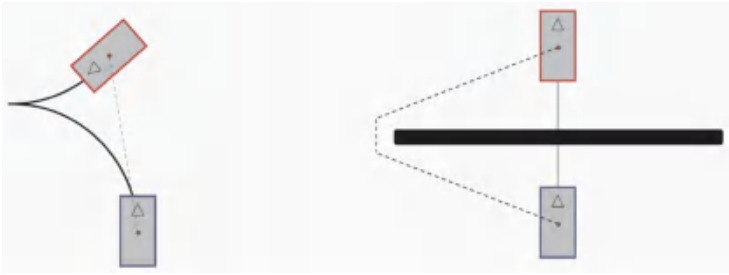








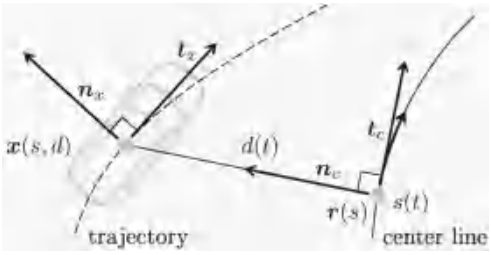
7.6 启发式函数设计



1. 简化原则
   (1) 不考虑动力学约束
   (2) 不考虑障碍物
2. 启发式函数计算
   (1) 对于每个节点（状态），不考虑动力学模型，搜索其最短路径。
   (2) 对于每个节点（状态），使用最优边值问题 (OBVP) 作为启发式函数，求解到规划目标状态 h。

7.7 基于 Frenet 坐标系的规划

7.7.1 基本思想

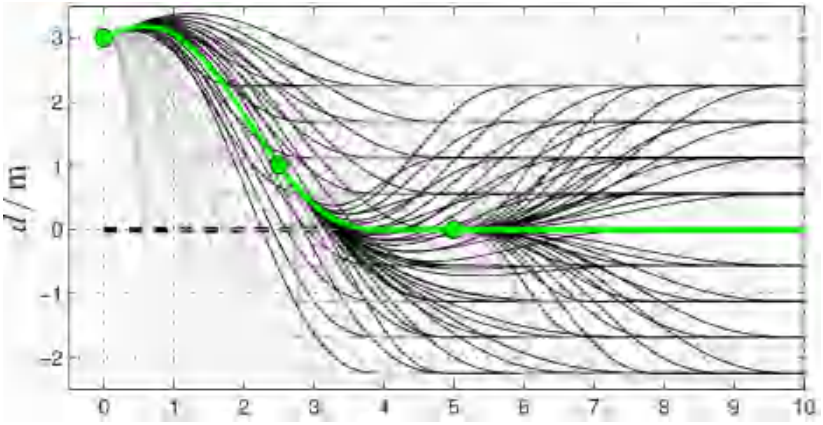


1. Frenet 坐标系特点
   (1) 动态坐标系。
   (2) 切向和径向独立。
   (3) 对于车道跟随问题是解耦的。
2. 参数化方式

$$\begin{aligned} d(t) &= a_{d0} + a_{d1}t + a_{d2}t^2 + a_{d3}t^3 + a_{d4}t^4 + a_{d5}t^5 \\ s(t) &= a_{s0} + a_{s1}t + a_{s2}t^2 + a_{s3}t^3 + a_{s4}t^4 + a_{s5}t^5 \end{aligned} \tag{90}$$

其中 $d(t)$ 为横向位置，$s(t)$ 为径向位置，采用五次多项式参数化。

7.7.2 横向轨迹生成（这里只讨论横向规划）



1. 初始条件 $$D(0) = \left(d_0, \dot{d}_0, \ddot{d}_0\right), \quad d(0) = d_0, \quad \dot{d}(0) = \dot{d}_0, \quad \ddot{d}(0) = \ddot{d}_0 \tag{91}$$
2. 终止条件 $$D(T) = \left(d_f, \dot{d}_f, \ddot{d}_f\right) \tag{92}$$
3. 车道跟随情况 $$D(T) = \left(d_f, 0, 0\right) \tag{93}$$
4. 系数求解

$$\begin{pmatrix} T^3 & T^4 & T^5 \\ 3T^2 & 4T^3 & 5T^4 \\ 6T & 12T^2 & 20T^3 \end{pmatrix} \begin{pmatrix} a_{d3} \\ a_{d4} \\ a_{d5} \end{pmatrix} = \begin{pmatrix} \Delta p \\ \Delta v \\ \Delta a \end{pmatrix} = \begin{pmatrix} d_f - \left(d_0 + \dot{d}_0 T + \frac{1}{2}\ddot{d}_0 T^2\right) \\ \dot{d}_f - \left(\dot{d}_0 + \ddot{d}_0 T\right) \\ \ddot{d}_f - \ddot{d}_0 \end{pmatrix} \tag{94}$$

7.8 Hybrid A*

7.8.1 基本思路

1. 核心思想

   (1) 在线生成密集栅格需要花费太多时间。

   (2) 通过聚集和修剪减少计算量。

   (3) 定义剪枝规则，利用栅格地图。

2. 算法流程

   (1) 维护一个优先队列来存储所有要扩展的节点。

   (2) 对所有节点的启发式函数 $h(n)$ 都是提前定义好的。

   (3) 优先队列被起始状态 X_S 初始化。

   (4) 赋值 $g(X_S)=0$，且对于图中其他节点 $g(n)=\infty$。

   (5) 循环

   ① 如果队列为空，返回 FALSE；退出循环。

   ② 从优先队列中移除 $f(n)=g(n)+h(n)$ 值最低的节点 n。

   ③ 将节点 n 记作被拓展过的。

   ④ 如果节点 n 是目标状态，返回 TRUE；退出循环。

   ⑤ 对于所有未被拓展过的节点 n 的邻居节点 m。（通过对节点中的状态进行前向积分来查找邻居）

   a. 如果 $g(m)=\infty$，$g(m)=g(n)+C_{nm}$，将节点 m 压入队列。（记录节点 m 内部的状态）

   b. 如果 $g(m)>g(n)+C_{nm}$，$g(m)=g(n)+C_{nm}$。（更新节点 m 内部的状态）

   (6) 结束循环

7.8.2 启发式函数设计

图 89 2D 欧式距离

图 90 不考虑障碍物

图 91 不考虑障碍的非完整性，在死胡同中表现不佳

图 92 不考虑障碍 + （2D 最短路径）

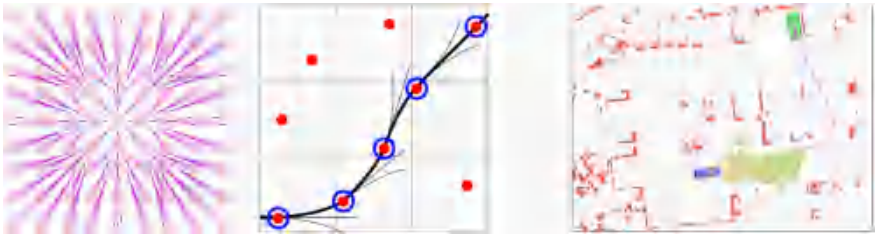
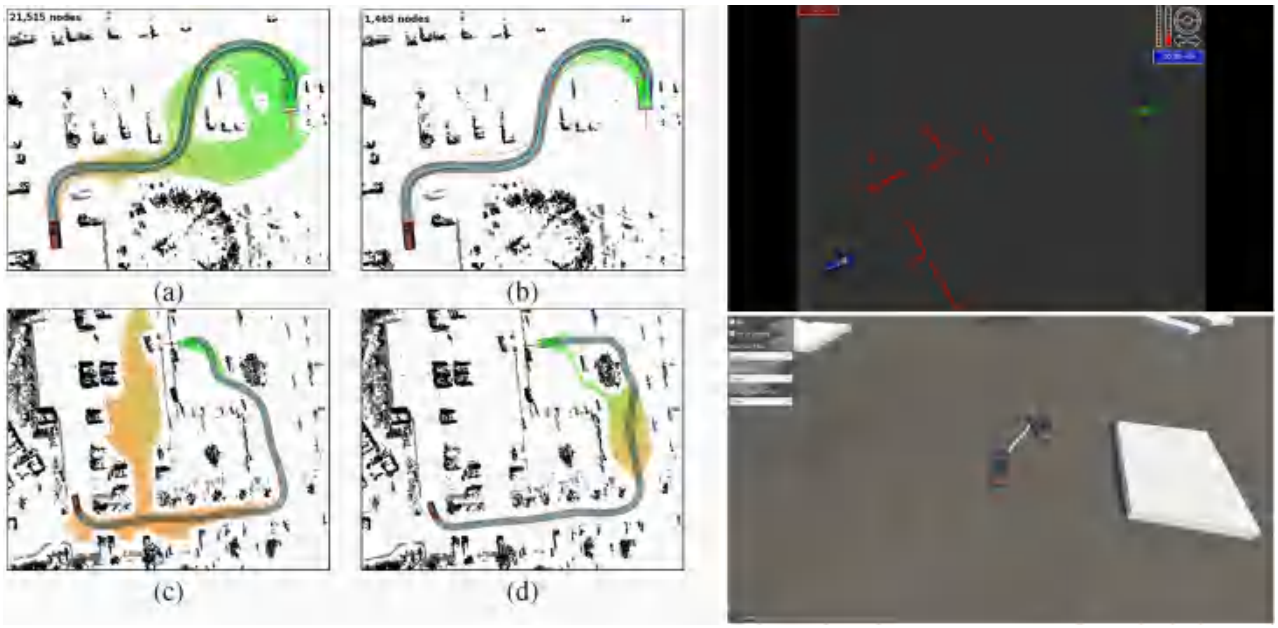


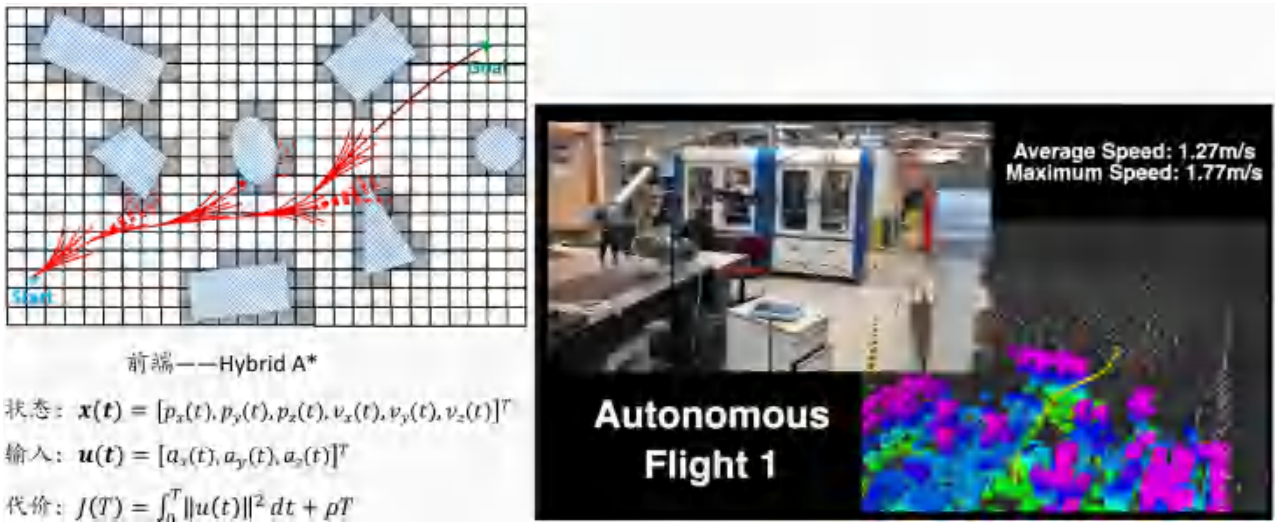
图 93　一次性启发 One shot

7.8.3　应用

1. 无人车



2. 无人机



7.9　Kinodynamic RRT*

7.9.1　算法流程 (Workflow)

```
Kinodynamic RRT*
Input: E, x_init, x_goal
Output: A trajectory T from x_init to x_goal
T.init();
for i = 1 to n do
    x_rand ← Sample(E);
    X_near ← Near(T, x_rand);
    x_min ← ChooseParent(X_near, x_rand);
    T.addNode(x_rand);
    T.rewire();
```

【RRT*回顾】

1. Sample：在配置空间中随机采样。
2. Near：找到树中最近的节点。
3. Steer：从最近节点向采样点延伸。
4. CollisionFree：检查新节点是否无碰撞。
5. NearC：在半径内找到邻近节点集合。
6. ChooseParent：选择代价最小的父节点。
7. Rewire：重新连接树以优化路径。

1. 如何采样 (How to “Sample”)

(1) LTI 系统状态空间方程

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{B}\boldsymbol{u}(t)+\boldsymbol{c} \tag{95}$$

(2) 双积分系统示例

$$\boldsymbol{x}=\begin{pmatrix}p\\v\end{pmatrix},\quad \boldsymbol{A}=\begin{pmatrix}0&I\\0&0\end{pmatrix},\quad \boldsymbol{B}=\begin{pmatrix}0\\I\end{pmatrix} \tag{96}$$

(3) 关键：不像 RRT 只在位置空间采样，还需要在完整状态空间中采样，包括采样速度。

2. 如何定义邻点 (How to define “Near”)

(1) 如果没有运动约束，可以使用欧氏距离或曼哈顿距离。

(2) 在具有运动约束的状态空间中，引入最优控制。

① 定义状态转移的 cost functions。

$$c[\pi]=\int_0^{\tau}(1+\boldsymbol{u}(t)^T\boldsymbol{R}\boldsymbol{u}(t))dt \tag{97}$$

通常采用时间-能量最优的二次形式。

② 如果从一个状态转移到另一个状态的 cost 很低，则两个状态接近。

- 注意：状态反向转换的 cost 可能不同。

③ 给定终点状态 x_1 和到达时间 τ 的情况：使用经典的最优控制解决方案 (OBVP)。

a. 最优控制输入

$$\boldsymbol{u}^*(t)=\boldsymbol{R}^{-1}\boldsymbol{B}^T\exp[\boldsymbol{A}^T(\tau-t)]\boldsymbol{G}(\tau)^{-1}[\boldsymbol{x}_1-\overline{\boldsymbol{x}}(\tau)] \tag{98}$$

其中权重矩阵 $\boldsymbol{G}(t)$ 满足

$$\boldsymbol{G}(t)=\int_0^t\exp[\boldsymbol{A}(t-t')]\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\exp[\boldsymbol{A}^T(t-t')]dt' \tag{99}$$

b. 李雅普诺夫 (Lyapunov) 方程的解

$$\dot{\boldsymbol{G}}(t)=\boldsymbol{A}\boldsymbol{G}(t)+\boldsymbol{G}(t)\boldsymbol{A}^T+\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T,\quad \boldsymbol{G}(0)=\boldsymbol{0} \tag{100}$$

c. 如果没有应用控制输入，x 在 t 时的状态

$$\overline{\boldsymbol{x}}(t)=\exp(\boldsymbol{A}t)\boldsymbol{x}_0+\int_0^t\exp[\boldsymbol{A}(t-t')]\boldsymbol{c}dt' \tag{101}$$

满足微分方程

$$\dot{\overline{\boldsymbol{x}}}(t)=\boldsymbol{A}\overline{\boldsymbol{x}}(t)+\boldsymbol{c},\quad \overline{\boldsymbol{x}}(0)=\boldsymbol{x}_0 \tag{102}$$

④ 给定终点状态 x_1，改变到达时间 τ 的情况：如果要找到最佳到达时间 τ，可以通过改变控制输入 $u^*(t)$，评估代价函数 $c[\tau]$ 来实现。

a. 代价函数

$$c[\tau]=\tau+[\boldsymbol{x}_1-\overline{\boldsymbol{x}}(\tau)]^T\boldsymbol{G}(t)^{-1}[\boldsymbol{x}_1-\overline{\boldsymbol{x}}(\tau)] \tag{103}$$

b. 最优到达时间 τ^* 满足

$$\dot{c}[\tau] = 1 - 2(\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{c})^T \boldsymbol{d}(\tau) - \boldsymbol{d}(\tau)^T \boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T \boldsymbol{d}(\tau) = 0 \tag{104}$$

其中

$$\boldsymbol{d}(\tau) = \boldsymbol{G}(t)^{-1}[\boldsymbol{x}_1 - \overline{\boldsymbol{x}}(\tau)] \tag{105}$$

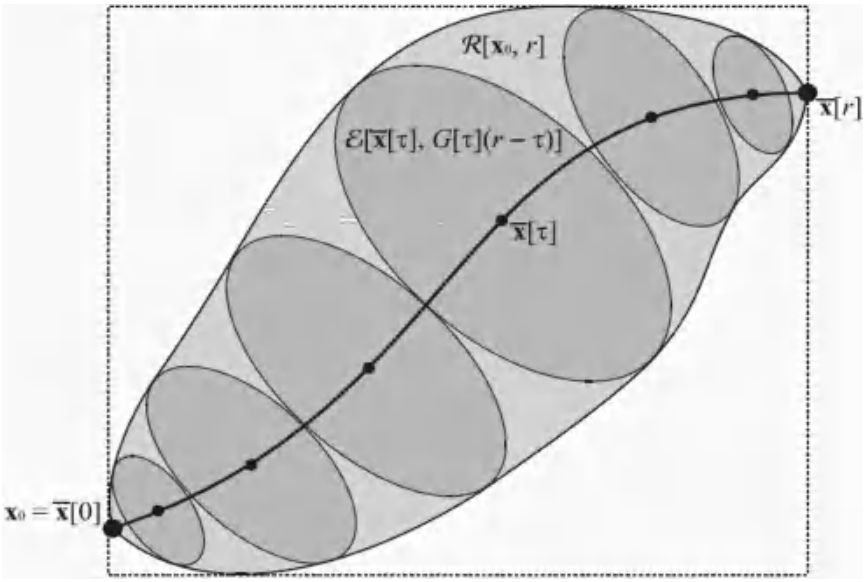
c. 对于双积分系统，$c[\tau]$ 为 4 阶多项式，可能有多个局部最小值。

d. 若给定最优到达时间 τ^*，则问题转变为“给定终点状态 x_1 和到达时间 τ”的问题。

3. 高效查找近邻节点 (How to find near nodes efficiently)

如果设置代价容限 (cost tolerance) r，可以计算 当前状态的前向可达集 和 终止状态的向后可达集。如果以 kd 树的形式存储节点，就可以在树中进行范围查询。

(1) 前向可达集：所有可能到达时间 τ 的高维椭球体的并集。



$$\left\{\boldsymbol{x}_1 \mid \tau + [\boldsymbol{x}_1 - \overline{\boldsymbol{x}}(\tau)]^T \boldsymbol{G}(t)^{-1}[\boldsymbol{x}_1 - \overline{\boldsymbol{x}}(\tau)] < r\right\} = \mathcal{E}[\overline{\boldsymbol{x}}(\tau), \boldsymbol{G}(t)(r-\tau)] \tag{106}$$

其中 $\mathcal{E}[x, M]$ 是一个具有中心 x 和正定权重矩阵 M 的椭球体：

$$\mathcal{E}[\boldsymbol{x}, \boldsymbol{M}] = \left\{\boldsymbol{x}' \mid (\boldsymbol{x}' - \boldsymbol{x})^T \boldsymbol{M}^{-1}(\boldsymbol{x}' - \boldsymbol{x}) < 1\right\} \tag{107}$$

(2) 向后可达集与向前可达集的计算类似。

- 当进行邻居节点查询和选择父节点时，可以从 x_{rand} 的后向可达集中找到 X_{near}。

(3)【问题简化】对几个 τ 进行采样，为每个 τ 计算椭球的轴对齐边界框，更新每个维度中的最大值和最小值：

$$\prod_{k=1}^{n}\Big[\min\{0 < \tau < r\}\Big(\bar{\boldsymbol{x}}(\tau)_k - \sqrt{\boldsymbol{G}[\tau]_{(k,k)}(r-\tau)}\Big), \max\{0 < \tau < r\}\Big(\bar{\boldsymbol{x}}(\tau)_k + \sqrt{\boldsymbol{G}[\tau]_{(k,k)}(r-\tau)}\Big)\Big] \tag{108}$$

4. 如何选择父节点 (How to “ChooseParent”)

(1) 在高维空间对随机状态进行采样，计算从树中当前节点到采样状态的控制输入和代价。

(2) 选择代价最低的节点，并检查 $x(t)$ 和 $u(t)$ 在边界内。

(3) 如果找不到合格的父节点，对另一个状态进行采样。

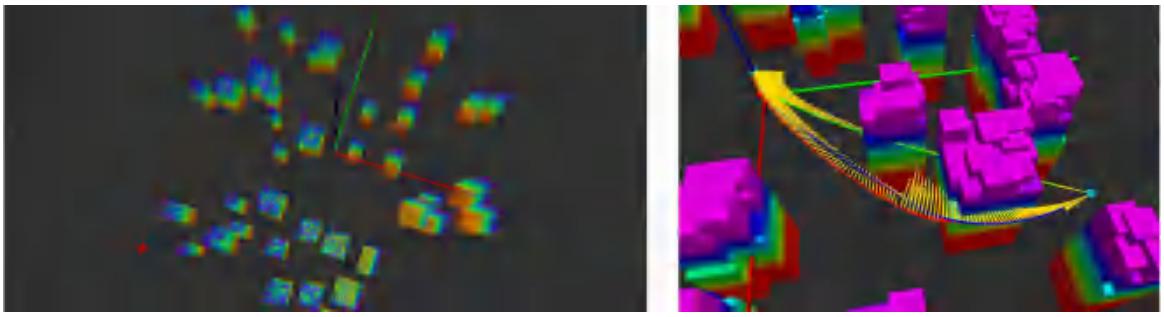
5. 如何重新连线 (How to “Rewire”)

(1) 计算 x_{rand} 的前向可达集。

(2) 求解到前向可达集内节点的 OBVP。

(3) 如果通过 x_{rand} 可以降低代价，则更新连接关系。

7.9.2 Demos



1. 绿色曲线没有考虑障碍物。
2. 黄线是每个控制点的控制输入。
3. 红色曲线是 Kinodynamic 轨迹规划器的结果。
4. 蓝色曲线是 Kinodynamic 轨迹规划器发现的第一条可行轨迹。

§8 轨迹优化（主讲：高飞）

8.1 轨迹优化的必要性

1. 为什么需要轨迹优化？

(1) 机器人的速度和动力学高阶量无法突变。

(2) 节约机器人运动过程的能量消耗。

(3) 保证轨迹执行时间的合理性。

(4) 确保机器人移动过程中的安全。

2. 核心问题

(1) Q: 我们有前端路径搜索，为什么需要后端轨迹优化？

(2) Q: 前端 Kinodynamic Path Finding 是可行的，为什么后端轨迹优化是必要的？

3. 答案：路径搜索给出的是离散的路径点，但机器人需要连续、光滑、满足动力学约束的轨迹。

8.2 微分平坦 (Differential Flatness)

8.2.1 基本概念

1. 定义

(1) 四旋翼的状态和输入可以写成四个挑选出来的输出变量及其导数的代数函数。

(2) 能够自动生成轨迹。

(3) 在空间中任何具有平坦输出（输出具有合理的有界导数）的任意平滑轨迹都可以被旋翼跟踪。

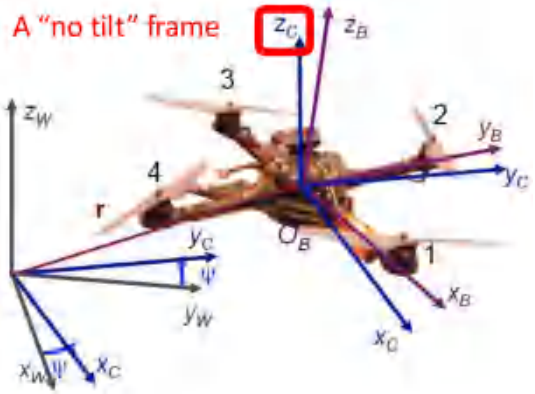
2. 一种平坦输出选择

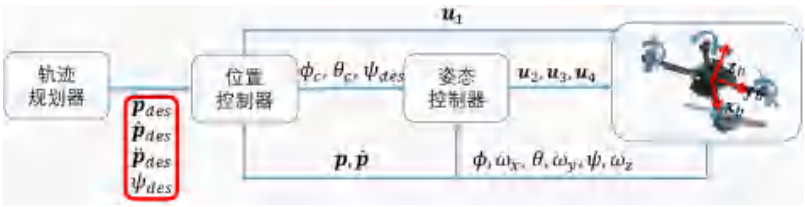
$$\boldsymbol{\sigma} = [x, y, z, \psi]^T \tag{109}$$

其中 (x, y, z) 是位置，ψ 是偏航角 (yaw)。

3. 在空间中有平坦输出轨迹 $\boldsymbol{\sigma}(t) : [T_0, T_M] \to \mathbb{R}^3 \times SO(2)$ (110)

8.2.2 四旋翼动力学





1. 旋翼状态向量（位置、姿态、线速度、角速度）

$$\boldsymbol{X} = \left[x, y, z, \varphi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z\right]^T \tag{111}$$

其中，$\omega_x, \omega_y, \omega_z$ 是机体系下，机体旋转的即时角速度。

2. 非线性动力学方程

在之前的章节中，我们有牛顿方程 $m\ddot{\boldsymbol{p}} = \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix} + \boldsymbol{R} \begin{pmatrix} 0 \\ 0 \\ F_1+F_2+F_3+F_4 \end{pmatrix}$ 和欧拉方程 $\boldsymbol{I} \begin{pmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{pmatrix} + \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \times \boldsymbol{I} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} l(F_2-F_4) \\ l(F_3-F_1) \\ M_1-M_2+M_3-M_4 \end{pmatrix}$，将其简化我们可以得到

$$m\ddot{\boldsymbol{p}} = -mg\boldsymbol{z}_W + u_1\boldsymbol{z}_B, \quad \boldsymbol{\omega}_B = \begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}, \quad \dot{\boldsymbol{\omega}}_B = \boldsymbol{I}^{-1}\left[-\boldsymbol{\omega}_B \times \boldsymbol{I}\dot{\boldsymbol{\omega}}_B + \begin{pmatrix}u_2\\ u_3\\ u_4\end{pmatrix}\right] \tag{112}$$

3. 姿态角的推导

(1) 从运动方程可得机体 z 轴方向：$\boldsymbol{z}_B = \dfrac{\boldsymbol{t}}{\|\boldsymbol{t}\|}, \quad \boldsymbol{t} = [\ddot{\sigma}_1, \ddot{\sigma}_2, \ddot{\sigma}_3 + g]^T$ (113)

(2) 定义 yaw 的向量（Z-X-Y 欧拉角）：$\boldsymbol{x}_C = [\cos(\sigma_4), \sin(\sigma_4), 0]^T$ (114)

(3) 姿态角可以按如下公式获取：

$$\boldsymbol{y}_B = \frac{\boldsymbol{z}_B \times \boldsymbol{x}_C}{\|\boldsymbol{z}_B \times \boldsymbol{x}_C\|}, \quad \boldsymbol{x}_B = \boldsymbol{y}_B \times \boldsymbol{z}_B \Longrightarrow \boldsymbol{R}_B = [\boldsymbol{x}_B \quad \boldsymbol{y}_B \quad \boldsymbol{z}_B] \tag{115}$$

4. 角速度的推导

(1) 对运动方程求导可得 $m\dot{\boldsymbol{a}} = \dot{u}_1\boldsymbol{z}_B + \boldsymbol{\omega}_{BW} \times u_1\boldsymbol{z}_B$ (116)

- ω_{BW} 为世界系下，机体旋转的即时角速度

(2) 旋翼只有机体 z 轴方向有推力：$\dot{u}_1 = \boldsymbol{z}_B \cdot m\dot{\boldsymbol{a}}$ (117)

(3) 将 $\frac{m}{u_1}\dot{\boldsymbol{a}}$ 投影到 $x_B - y_B$：$\boldsymbol{h}_\omega = \boldsymbol{\omega}_{BW} \times \boldsymbol{z}_B = \dfrac{m}{u_1}(\dot{\boldsymbol{a}} - (\boldsymbol{z}_B \cdot \dot{\boldsymbol{a}})\boldsymbol{z}_B)$ (118)

(4) 由于 $\omega_{BW} = \omega_x\boldsymbol{x}_B + \omega_y\boldsymbol{y}_B + \omega_z\boldsymbol{Z}_B$，故沿着 $\boldsymbol{x}_B$ 和 $\boldsymbol{y}_B$ 方向的角速度为

$$\omega_x = -\boldsymbol{h}_\omega\dot{\boldsymbol{y}}_B, \quad \omega_y = \boldsymbol{h}_\omega\dot{\boldsymbol{x}}_B \tag{119}$$

(5) 由于 $\omega_{BW} = \omega_{BC} + \omega_{CW}$，$\boldsymbol{\omega_{BC}}$ 没有 z_B 分量，故沿着 z_B 方向的角速度为

$$\omega_z = \boldsymbol{\omega}_{BW}\dot{\boldsymbol{z}}_B = \boldsymbol{\omega}_{CW}\dot{\boldsymbol{z}}_B = \dot{\psi}\boldsymbol{z}_W\dot{\boldsymbol{z}}_B \tag{120}$$

8.3 SE(3)控制

8.3.1 问题描述

要求跟踪一条期望轨迹：$\boldsymbol{\sigma}_T(t) = [\boldsymbol{r}_T(t)^T, \psi_T(t)]^T$，其中，$\boldsymbol{r}$ 是位置，ψ 是 yaw 角。

8.3.2 控制律设计

1. 位置控制

(1) 定义位置和速度误差：$\boldsymbol{e}_p = \boldsymbol{r} - \boldsymbol{r_T}, \quad \boldsymbol{e}_v = \dot{\boldsymbol{r}} - \dot{\boldsymbol{r}}_{\boldsymbol{T}}$ (121)

(2) 计算期望的推力：$\boldsymbol{F}_{\text{des}} = -K_p\boldsymbol{e}_p - K_v\boldsymbol{e}_v + mg\boldsymbol{z_W} + m\ddot{\boldsymbol{r}}_{\boldsymbol{T}}$ (122)

(3) 将计算得到的推力投影到当前实际的机体坐标系的 z 轴作为输入：

$$u_1 = \boldsymbol{F}_{\text{des}} \cdot \dot{\boldsymbol{z}}_B \tag{123}$$

2. 姿态控制 - 为了得到剩下的三个输入，需要计算旋转误差。

(1) 计算期望的 z_B：$\boldsymbol{z}_{B,\,\text{des}} = \dfrac{\boldsymbol{F}_{\text{des}}}{\|\boldsymbol{F}_{\text{des}}\|}$ (124)

(2) 根据微分平坦计算 $\boldsymbol{x}_{B,\,\text{des}}$ 和 $\boldsymbol{y}_{B,\,\text{des}}$：

$$\boldsymbol{x}_{C,\,\text{des}} = [\cos(\psi_{\boldsymbol{T}}), \sin(\psi_{\boldsymbol{T}}), 0]^T \tag{125}$$

$$\begin{aligned}\boldsymbol{y}_{B,\,\mathrm{des}} &= \frac{\boldsymbol{z}_{B,\,\mathrm{des}} \times \boldsymbol{x}_{C,\,\mathrm{des}}}{\left\|\boldsymbol{z}_{B,\,\mathrm{des}} \times \boldsymbol{x}_{C,\,\mathrm{des}}\right\|} \\ \boldsymbol{x}_{B,\,\mathrm{des}} &= \boldsymbol{y}_{B,\,\mathrm{des}} \times \boldsymbol{z}_{B,\,\mathrm{des}} \\ \boldsymbol{R}_{B,\,\mathrm{des}} &= \begin{pmatrix}\boldsymbol{x}_{B,\,\mathrm{des}} & \boldsymbol{y}_{B,\,\mathrm{des}} & \boldsymbol{z}_{B,\,\mathrm{des}}\end{pmatrix}\end{aligned} \tag{125}$$

(3) 定义姿态误差：

$$\boldsymbol{e}_R = \frac{1}{2}\left(\boldsymbol{R}_{\mathrm{des}}^T W \boldsymbol{R}_B - W \boldsymbol{R}_B^T \boldsymbol{R}_{\mathrm{des}}\right)^{\vee} \tag{126}$$

其中 $\vee$ 表示从 $SO(3)$ 到 $\mathbb{R}^3$ 的 vee 映射。

(4) 定义角速度误差：

$$\boldsymbol{e}_\omega = {}^B\boldsymbol{\omega}_{BW} - {}^B\boldsymbol{\omega}_{BW,T} \tag{127}$$

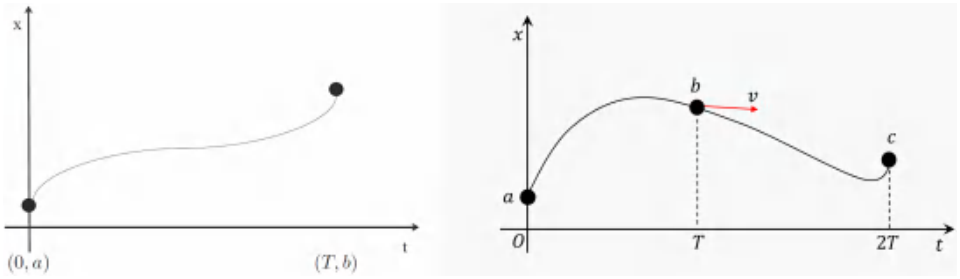
(5) 剩下的三个输入：

$$[u_2, u_3, u_4]^T = -K_R \boldsymbol{e}_R - K_\omega \boldsymbol{e}_\omega \tag{128}$$

8.4 光滑轨迹生成

1. 边界条件：起始、目标位置（朝向）。
2. 中间条件：路径点位置（朝向）。
   - 路径点由路径规划得到（A*，RRT*等）。
3. 光滑性的标准 - 通常通过最小化“输入”的变化率。

8.4.1 光滑一维轨迹



1. 问题描述

   (1) 设计一条轨迹 $x(t)$ 使得：$x(0) = a$ 和 $x(T) = b$。

   (2) 轨迹光滑性通过轨迹参数化形式保证。

2. 5 次多项式轨迹：

$$x(t) = p_5 t^5 + p_4 t^4 + p_3 t^3 + p_2 t^2 + p_1 t + p_0 \tag{129}$$

   (1) 边界条件：

	位置	速度	加速度
$t = 0$	a	0	0
$t = T$	b	0	0

   (2) 求解线性方程组：

$$\begin{pmatrix} a \\ b \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{pmatrix} \tag{130}$$

8.4.2 光滑多段一维轨迹

1. 中间条件

   (1) 光滑直线段的转角处

   ① 倾向于以 v 作匀速运动。

   ② 倾向于零加速度。

   (2) 短的直线段需要特殊处理。

2. 边界条件（含速度）

	位置	速度	加速度
$t=0$	a	v_0	0
$t=T$	b	v_T	0

3. 求解线性方程组：

$$\begin{pmatrix} a \\ b \\ v_0 \\ v_T \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{pmatrix} \begin{pmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{pmatrix} \tag{131}$$

8.5 光滑 3 维轨迹 - 最小化 Snap 的轨迹生成（课上讲了很久）

8.5.1 问题定义

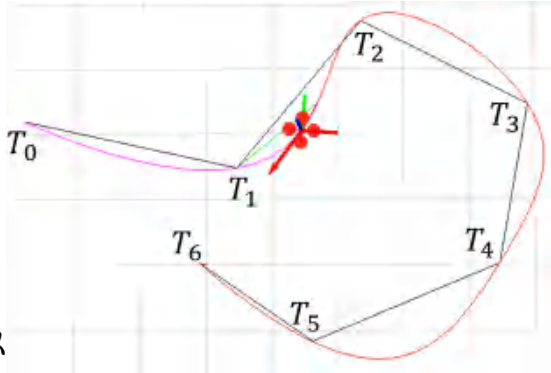
1. 目标

   (1) 轨迹的初始（或初末）状态给定。

   (2) 轨迹在预定时刻需经过预设航点。

   (3) 轨迹的 4 阶导数处处存在。

   (4) 最小化轨迹 4 阶导数平方幅值积分。



2. 重要性质

   (1) 最优轨迹的参数化形式为分段多项式。

   (2) 每段多项式的次数不超过 $2\times 4-1=7$，即每段均可以被 7 次多项式表征。

3. 轨迹表示

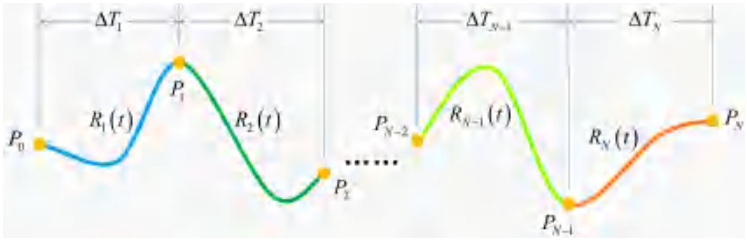
$$f(t)=\begin{cases} f_1(t)=\sum_{i=0}^{N} p_{1,i}t^i & T_0\le t\le T_1 \\ f_2(t)=\sum_{i=0}^{N} p_{2,i}t^i & T_1\le t\le T_2 \\ \vdots & \\ f_{M(t)}=\sum_{i=0}^{N} p_{M,i}t^i & T_{M-1}\le t\le T_M \end{cases} \tag{132}$$

4. 如何合理确定轨迹的阶数

(1) 确保单段平滑多项式轨迹的最小次数

① Minimum jerk：$N = 2 \times 3 - 1 = 5$

② Minimum snap：$N = 2 \times 4 - 1 = 7$



(2) 确保多段平滑多项式轨迹的最小次数

【Minimum jerk】

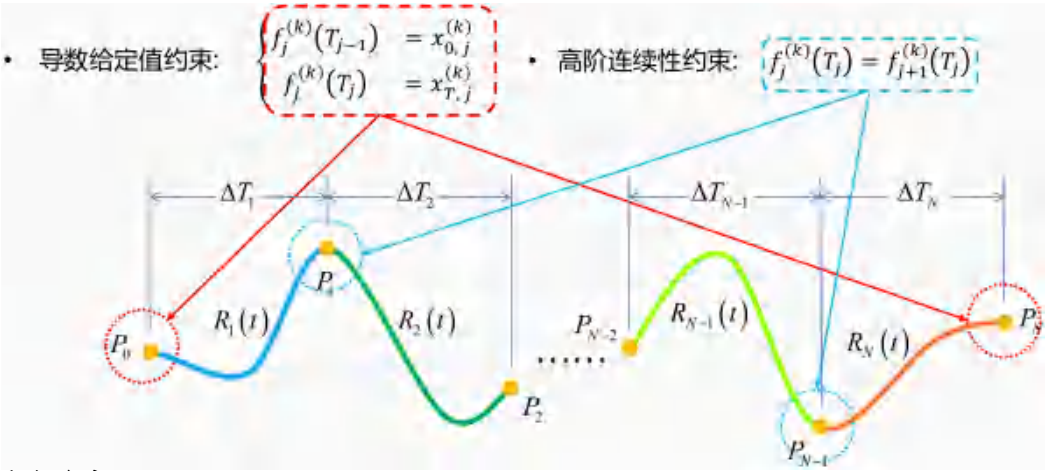
① 约束数量：$3 + 3 + (k-1) = k + 5$

② 未知数数量：$(N+1) \times k$

③ $(N+1) \times k = k + 5 \Rightarrow N = \frac{5}{k}$

> 提示
>
> 光滑性意味着导数是连续的，施加的约束独立！

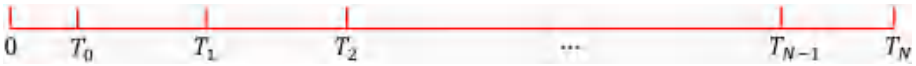
5. 约束



6. 两种时间参数方案

(1) 方案 1：相对每一段初始的时间。优点：数值稳定性高。

(2) 方案 2：相对第一段初始的时间。优点：数学表达清楚。



7. 目标函数的解析表达

(1) 推导过程

$$f(t) = \sum_i p_i t^i \Rightarrow f^{(4)}(t) = \sum_{i \geq 4} i(i-1)(i-2)(i-3) t^{i-4} p_i$$

$$\left(f^{(4)}(t)\right)^2 = \sum_{i \geq 4, l \geq 4} i(i-1)(i-2)(i-3) l(l-1)(l-2)(l-3) t^{i+l-8} p_i p_l$$

$$\Rightarrow J(T) = \int_{T_{j-1}}^{T_j} \left(f^{(4)}(t)\right)^2 \mathrm{d}t = \sum_{i \geq 4, l \geq 4} \frac{i(i-1)(i-2)(i-3) l(l-1)(l-2)(l-3)}{i+l-7} \left(T_j^{i+l-7} - T_{j-1}^{i+l-7}\right) p_i p_l$$

$$\Rightarrow J(T)=\begin{pmatrix}\vdots\\ p_i\\ \vdots\end{pmatrix}^T\begin{pmatrix}\vdots\\ \cdots\ \dfrac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7}T^{i+l-7}\ \cdots\\ \vdots\end{pmatrix}\begin{pmatrix}\vdots\\ p_l\\ \vdots\end{pmatrix}$$

(2) 矩阵形式

$$J_j(T)=\boldsymbol{p}_j^T\boldsymbol{Q}_j\boldsymbol{p}_j \tag{133}$$

其中 $\boldsymbol{Q}_j$ 是半正定矩阵。

8.5.2 约束条件

1. 导数给定值约束

$$f_j^{(k)}(T_j)=x_j^{(k)}$$

可表示为：

$$\boldsymbol{A}_j\boldsymbol{p}_j=\boldsymbol{d}_j$$

2. 高阶连续性约束

$$f_j^{(k)}(T_j)=f_{j+1}^{(k)}(T_j)$$

可表示为：

$$\begin{bmatrix}\boldsymbol{A}_j & -\boldsymbol{A}_{j+1}\end{bmatrix}\begin{pmatrix}\boldsymbol{p}_j\\ \boldsymbol{p}_{j+1}\end{pmatrix}=\boldsymbol{0}$$

8.5.3 线性等式约束的二次规划

1. 标准形式

$$\min_{\boldsymbol{p}_1,\cdots,\boldsymbol{p}_M}\ \begin{pmatrix}\boldsymbol{p}_1\\ \vdots\\ \boldsymbol{p}_M\end{pmatrix}^T\begin{pmatrix}\boldsymbol{Q}_1 & \boldsymbol{0} & \boldsymbol{0}\\ \boldsymbol{0} & \ddots & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}_M\end{pmatrix}\begin{pmatrix}\boldsymbol{p}_1\\ \vdots\\ \boldsymbol{p}_M\end{pmatrix}$$

$$\text{s.t.}\quad \boldsymbol{A}_{\text{eq}}\begin{pmatrix}\boldsymbol{p}_1\\ \vdots\\ \boldsymbol{p}_M\end{pmatrix}=\boldsymbol{d}_{\text{eq}}$$

2. 最小化 Snap 的轨迹生成是一个典型的凸优化问题

8.6 凸优化基础

8.6.1 凸函数和凸集

1. 凸函数定义

(1) 一个函数 $f:\mathbb{R}^n\to\mathbb{R}$ 被称作凸函数，如果域 $\text{dom}\ f$ 是凸，并且对于任意 $x,y\in\text{dom}\ f$ 和 $0\le\theta\le1$：

$$f(\theta \boldsymbol{x}+(1-\theta) \boldsymbol{y}) \leq \theta f(\boldsymbol{x})+(1-\theta) f(\boldsymbol{y})$$

(2) 如果不等关系对 $0<\theta<1$ 严格成立，那么 f 是严格凸的

(3) 如果 $-f$ 是凸，那么 f 是凹的

2. 凸集定义

(1) 一个集合 $C \in \mathbb{R}^n$ 被称作凸集，如果任意两点间的线段在集合中：

$$\theta \boldsymbol{x}+(1-\theta) \boldsymbol{y} \in C, \quad \forall \boldsymbol{x}, \boldsymbol{y} \in C, \quad 0 \leq \theta \leq 1$$

8.6.2 凸优化问题

1. 标准形式

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_{i(\boldsymbol{x})} \leq 0, \quad i=1, ..., m \\ & \boldsymbol{A} \boldsymbol{x}=\boldsymbol{b} \end{aligned}$$

其中 $f_0, f_1, ..., f_m$ 是凸函数，等式约束是仿射变换

2. 重要性质

(1) 局部和全局最优：任何凸问题的局部最优点也是全局最优

(2) 大多数问题在被表达时是非凸的

(3) 将问题以凸的形式表达是一门艺术，没有系统的方法

8.6.3 几类规范的凸优化问题

1. 线性规划 (LP)

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^T \boldsymbol{x}+d \\ \text{s.t.} \quad & \boldsymbol{G} \boldsymbol{x} \leq \boldsymbol{h} \\ & \boldsymbol{A} \boldsymbol{x}=\boldsymbol{b} \end{aligned}$$

2. 二次规划 (QP)

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2} \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x}+\boldsymbol{q}^T \boldsymbol{x}+r \\ \text{s.t.} \quad & \boldsymbol{G} \boldsymbol{x} \leq \boldsymbol{h} \\ & \boldsymbol{A} \boldsymbol{x}=\boldsymbol{b} \end{aligned}$$

其中 $\boldsymbol{P} \in S_{\geq 0}^n$ (半正定)

3. 二次约束的二次规划 (QCQP)

$$\min_{\boldsymbol{x}} \quad \frac{1}{2} \boldsymbol{x}^T \boldsymbol{P}_0 \boldsymbol{x}+\boldsymbol{q}_0^T \boldsymbol{x}+r_0$$

$$\text{s.t.} \quad \frac{1}{2}\boldsymbol{x}^T\boldsymbol{P}_i\boldsymbol{x}+\boldsymbol{q}_i^T\boldsymbol{x}+r_i\leq 0, \quad i=1,...,m$$
$$\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$$

4. 二次锥规划 (SOCP)

$$\min_{\boldsymbol{x}} \quad \boldsymbol{f}^T\boldsymbol{x}$$
$$\text{s.t.} \quad \|\boldsymbol{A}_i\boldsymbol{x}+\boldsymbol{b}_i\|\leq \boldsymbol{c}_i^T\boldsymbol{x}+d_i, \quad i=1,...,m$$
$$\boldsymbol{F}\boldsymbol{x}=\boldsymbol{g}$$

8.7 决策变量映射

1. 问题

(1) 直接优化多项式轨迹在数值上不稳定

(2) 更倾向于替代变量，使得优化每段端点的微分

2. 映射矩阵

(1) 我们有 $M_j\boldsymbol{p}_j=\boldsymbol{d}_j$

(2) M_j 是一个映射矩阵，将多项式系数映射至微分

(3) 例如，对于 5 次多项式：

$$\boldsymbol{M}=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & 0 & 0 \\ T^5 & T^4 & T^3 & T^2 & T & 1 \\ 5T^4 & 4T^3 & 3T^2 & 2T & 1 & 0 \\ 20T^3 & 12T^2 & 6T & 2 & 0 & 0 \end{pmatrix}$$

3. 变换后的目标函数

$$J=\begin{pmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_M \end{pmatrix}^T \begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M}_M \end{pmatrix}^{-T} \begin{pmatrix} \boldsymbol{Q}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{Q}_M \end{pmatrix} \begin{pmatrix} \boldsymbol{M}_1 & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \ddots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{M}_M \end{pmatrix}^{-1} \begin{pmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_M \end{pmatrix}$$

8.8 分离固定变量和自由变量

1. 关键思想

(1) 使用选择矩阵 C 来分离自由变量 (d_P) 和受约束变量 (d_F)

(2) 自由变量：微分不受指定，只受连续性约束

2. 选择矩阵

$$\boldsymbol{C}^T\begin{pmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{pmatrix}=\begin{pmatrix} \boldsymbol{d}_1 \\ \vdots \\ \boldsymbol{d}_M \end{pmatrix}$$

3. 转变后的目标函数

$$J = \begin{pmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{pmatrix}^T \begin{pmatrix} \boldsymbol{R}_{FF} & \boldsymbol{R}_{FP} \\ \boldsymbol{R}_{PF} & \boldsymbol{R}_{PP} \end{pmatrix} \begin{pmatrix} \boldsymbol{d}_F \\ \boldsymbol{d}_P \end{pmatrix} = \boldsymbol{d}_F^T \boldsymbol{R}_{FF} \boldsymbol{d}_F + \boldsymbol{d}_F^T \boldsymbol{R}_{FP} \boldsymbol{d}_P + \boldsymbol{d}_P^T \boldsymbol{R}_{PF} \boldsymbol{d}_F + \boldsymbol{d}_P^T \boldsymbol{R}_{PP} \boldsymbol{d}_P$$

4. 闭式解

(1) 转变为一个不受约束的二次规划问题，可直接闭式求解：

$$\boldsymbol{d}_P^* = -\boldsymbol{R}_{PP}^{-1} \boldsymbol{R}_{FP}^T \boldsymbol{d}_F$$

5. 连续性约束通过选择矩阵隐性地实现

8.9 保证避障的平滑轨迹生成

8.9.1 飞行走廊 (Flight Corridor)

1. 核心思想

(1) Step 1: 障碍物检测

(2) Step 2: 寻找飞行走廊

(3) Step 3: 膨胀飞行走廊

(4) Step 4: 在走廊内生成轨迹

2. 约束类型

(1) 点线性约束：

① 起点终点的状态约束 $(\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b})$

② 中间轨迹点状态约束 $(\boldsymbol{A}\boldsymbol{p} = \boldsymbol{b}, \boldsymbol{A}\boldsymbol{p} \leq \boldsymbol{b})$

③ 分段连续性约束 $(\boldsymbol{A}\boldsymbol{p}_i = \boldsymbol{A}\boldsymbol{p}_{i+1})$

(2) 区间线性约束：

① 飞行走廊边界约束 $(\boldsymbol{A}(t)\boldsymbol{p} \leq \boldsymbol{b}, \forall t \in [t_l, t_r])$

② 动力学约束 $(\boldsymbol{A}(t)\boldsymbol{p} \leq \boldsymbol{b}, \forall t \in [t_l, t_r])$

- 速度限制
- 加速度限制

8.9.2 如何施加全局约束

1. 方案 1：反复检查极值并添加额外的约束

(1) 迭代求解非常耗时

(2) 严格来说，没有可行的解决方案满足所有约束

(3) 需运行 10 次迭代来确定解决方案的状态

2. 方案 2：在离散时间点添加大量约束

(1) 生成轨迹过于保守

(2) 约束太多，计算负担高

3. 更好的方案：使用凸包、多面体等几何方法构建飞行走廊

8.10 时间分配 (Time Allocation)

8.10.1 问题定义

1. 核心问题
   (1) 分段轨迹取决于分段时间分配
   (2) 时间分配显著影响最终轨迹
   (3) 如何合理分配时间？

8.10.2 简单的解决方法

1. “T 形速度时间”曲线
   (1) 使用预期平均速度来获得每段轨迹的持续时间
   (2) 假设每一段轨迹：匀加速 -> 最大速度 -> 匀减速 -> 速度至 0
   (3) 缺点：
      ① 远非最优时间分配
      ② 只能获得保守的时间分配
      ③ 对环境无法变通

2. 轨迹点重合区域
   (1) 允许轨迹点在飞行走廊的重合区域内进行调节
   (2) 提供一定解空间的自由度
   (3) 只给定了段与段之间的时间 T
   (4) 一定程度上缓解了时间分配不好带来的问题

8.10.3 迭代数值求解

1. 惩罚 Cost 中的时间

$$J_T = \begin{pmatrix} p_1 \\ \vdots \\ p_M \end{pmatrix}^T \begin{pmatrix} Q_1(T_1) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_M(T_M) \end{pmatrix} \begin{pmatrix} p_1 \\ \vdots \\ p_M \end{pmatrix} + k_T \sum_{i=1}^{M} T_i$$

2. 优化策略
   (1) 最小化目标函数 J
   (2) 从数值上得到 T 的梯度
   (3) 根据 k_T 的大小：

① k_T 小：调整轨迹段间的时间比例，调整总时间

② k_T 大：调整轨迹段间的时间比例，固定总时间

3. 两阶段优化

(1) 在没有约束的情况下，寻找最佳的时间分配，固定比例

(2) 通过保持最佳比例来调整总轨迹时间，直到出现约束为止

8.11 总结

最小 Snap 轨迹生成方法对比

方法	优势	劣势
基于闭式解的层次化规划	• 初始路径无碰撞 • 利用闭式解生成轨迹 • 迭代加入中间航点	• 轨迹可能碰撞 • 需要迭代修正 • 依赖路径质量
基于飞行走廊的轨迹生成	• 保证无碰撞 • 考虑动力学约束 • 可处理复杂环境	• 需要构建走廊 • 计算复杂度较高 • 可能过于保守
带时间优化的轨迹生成	• 自动优化时间分配 • 考虑动力学约束 • 轨迹更优	• 需要迭代优化 • 计算量大 • 收敛速度慢

关键要点

1. 微分平坦是连接轨迹规划和控制的桥梁
2. 凸优化是求解轨迹生成问题的有力工具
3. 决策变量映射可以提高数值稳定性
4. 分离固定变量和自由变量可以降低计算复杂度
5. 飞行走廊是保证避障的有效方法
6. 时间分配对轨迹质量有重要影响

§9 RL 在无人机中的应用（主讲：高飞）

这几部分见另一部分 PPT（“L4-RL 在无人机中的应用.pdf”）

§10 L4-Multi-agent 算法在无人机中应用（主讲：高飞）

这几部分见另一部分 PPT（“L4-Multi-agent 算法在无人机中应用.pdf”）

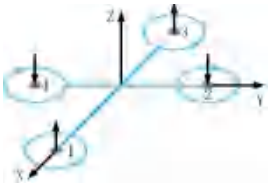
§11 期末复习

11.1 历年卷（部分重复的题就删去了，答案仅供参考）

11.1.1 2024-2025 秋冬

11.1.1.1 选择（10 道 × 2 分）

1. 悬停的无人机沿着 x 轴正方向运动需要让哪个电机加速，哪个电机减速。



   - 电机 1（前方）需要减速，电机 3（后方）需要加速

2. 3.6N 四旋翼无人机悬停，转速 3000RPM，求拉力系数。
   - $F_i = b\Omega_i^2 \Rightarrow b = F_i/\Omega_i^2 = 3.6/(4 * (3000 * 2\pi)^2)$

3. 哪个算法不基于采样？PRM/RRT/RRT*/A*
   - A*（基于搜索）

4. 空中机器人轨迹生成的基本要求不包括：安全/可拓展/光滑/动力学可行
   - 可扩展（？存疑）

5. NED 中文名是
   - 北东地坐标系
6. 哪个是本课任课老师
   - 任沁源、高飞
7. 实验课的地点
   - 教九-404
8. 树莓派的密码
   - pi
9. 系统 UBUNTU 的版本
   - 20.04
10. 实验中期望的 z 轴加速度是多少
    - 0 或 g（？这题有点不知道在问什么）

11.1.1.2 判断（10 道 × 2 分）

1. 四旋翼无人机对角桨叶转向相同？
   - 对
2. 四元数可以准确表示物体三维空间姿态？
   - 对
3. 四旋翼无人机四个桨叶相同，可以互换？
   - 错，对角的相同，相邻的相反
4. 反应式控制无法有效处理状态耦合？
   - 对
5. MPC 参数选择时，较小的预测时域会带来较小的计算量？
   - 对
6. BFS 用栈，DFS 用队列？
   - 错，反了
7. Dijkstra 不具备完备性与最优性？
   - 错，具备的
8. minimum snap 是凸优化问题？
   - 对
9. 四元数乘法满足结合和分配律，不满足交换律？
   - 对
10. RRT* 是基于搜索的？
    - 错，采样

11.1.1.3 设计（12 分）

设计具有自主导航功能的四旋翼空中机器人，画系统流程图，介绍用到的传感器及其作用。

- 可以参考“期末复习”的硬件框图（小节 11.2.2）。

11.1.1.4 简答（6 道 × 8 分）

1. 实验中，无人机的主要部件与对应的功能连接，并写出无人机用到的另一种部件。
   - （给框连线）
2. 文字描述 A*算法较 Dijkstra 算法的改进。对于下面的图，补全 h，f 并写出第三次出队的点。
   - （做不了）
3. 如下
   (1) 横滚、俯仰、偏摆角分别对应的旋转轴?
   - x, y, z
   (2) 横滚 φ，俯仰 θ，偏航角 ψ，推导 ZXY 欧拉公式。
   - $R_z(\psi)R_x(\varphi)R_y(\theta)$
   (3) 某无人机位姿如上一问且没有平移，某点在机体坐标系下为 $(x,0,0)^T$ ，求世界坐标系下的坐标。
   - 乘旋转矩阵（得结合具体题目）
4. 写出四旋翼动力学下的牛顿欧拉方程，解释变量含义；在平衡悬停条件下进行线性化简化。
   (1) 四旋翼动力学下的
   ① 牛顿方程：$m\ddot{\boldsymbol{p}} = -mg\boldsymbol{z}_W + u_1\boldsymbol{z}_B$
   - m：四旋翼质量
   - $\ddot{\boldsymbol{p}}$：位置的二阶导数（加速度矢量）
   - g：重力加速度
   - $\boldsymbol{z}_W$：世界坐标系（惯性系）的 z 轴单位向量，指向下
   - $u_1 = \sum_{i=1}^{4} f_i$：总推力（四个螺旋桨推力之和）
   - $\boldsymbol{z}_B$：机体坐标系的 z 轴单位向量，指向下

   ② 欧拉方程：$\boldsymbol{\omega}_B = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$，$\dot{\boldsymbol{\omega}}_B = \boldsymbol{I}^{-1}\left[-\boldsymbol{\omega}_B \times \boldsymbol{I}\dot{\boldsymbol{\omega}}_B + \begin{pmatrix} u_2 \\ u_3 \\ u_4 \end{pmatrix}\right]$
   - $\boldsymbol{\omega}_B = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} p \\ q \\ r \end{pmatrix}$：机体坐标系下的角速度矢量
     - p：滚转角速度
     - q：俯仰角速度
     - r：偏航角速度
   - $\boldsymbol{I} = \mathrm{diag}(I_x, I_y, I_z)$：转动惯量矩阵（对角阵）
     - I_x, I_y, I_z：绕机体 x, y, z 轴的转动惯量
   - u_2, u_3, u_4：控制力矩
     - u_2：滚转力矩
     - u_3：俯仰力矩

(2) 线性化简化后的

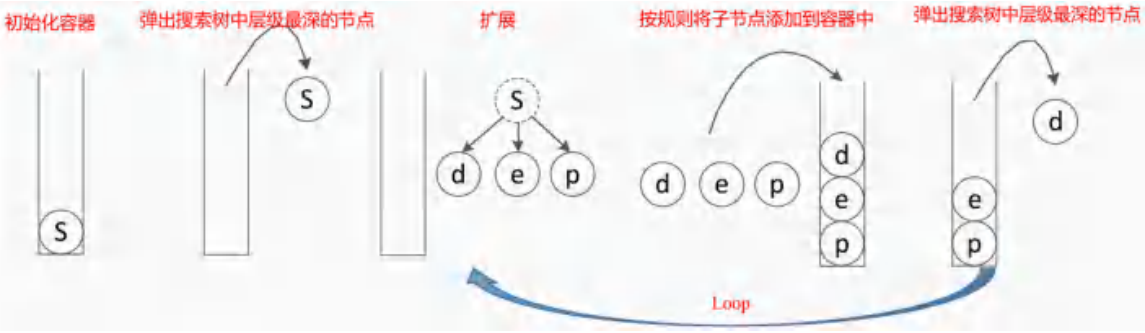
- 牛顿方程：$\begin{cases} \ddot{p}_1 = \ddot{x} = g(\theta\cos\psi + \varphi\sin\psi) \\ \ddot{p}_2 = \ddot{y} = g(\theta\sin\psi - \varphi\cos\psi) \\ \ddot{p}_3 = \ddot{z} = -g + \dfrac{u_1}{m} \end{cases}$
- 欧拉方程（疑?）：$\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} = \begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}, \boldsymbol{I}\cdot\begin{pmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{pmatrix} + \begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} \times \boldsymbol{I}\cdot\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} = \begin{pmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{pmatrix}$

5. 文字描述 DFS 和 BFS 的流程及二者区别。

- (1) 深度优先搜索 (Depth First Search, DFS)

① 策略：移除/扩展搜索树中层级最深的节点。

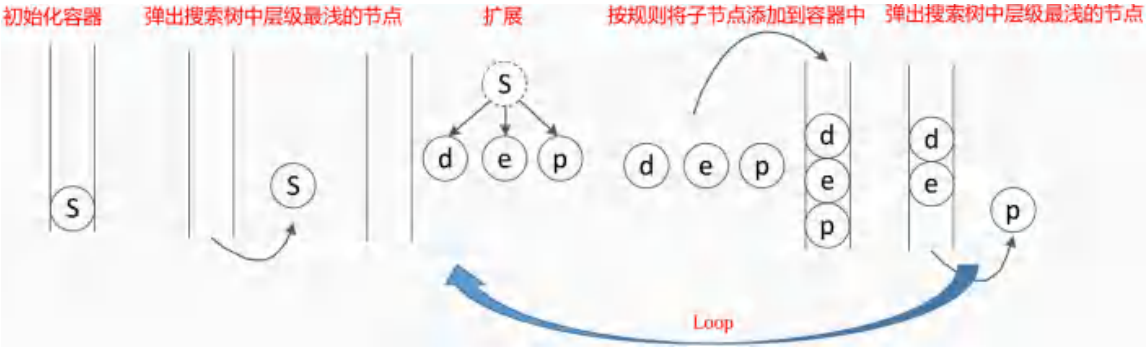
② 实现：维护后进先出（LIFO）容器（即堆栈 Stack）。



(2) 广度优先搜索 (Breadth First Search, BFS)

① 策略：移除/扩展搜索树中层级最浅的节点

② 实现：维护先进先出（FIFO）容器（即队列 Queue）



6. 用分段轨迹 $[0,T]$ 和 $[T,2T]$，描述某机器人在一维上的状态，状态包括位置，速度和加速度 $[p,v,a]$。假设 $t=0$ 时状态为 $[x,0,0]$，$t=2T$ 时状态为 $[z,0,0]$。

- 使用五阶多项式描述轨迹：$p(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$

(1) 以状态 $[y,v,a]$ 在 T 时刻通过中间点，请写出线性方程组。

① 轨迹分段：将轨迹分为两段。

- 第一段：$[0,T]$：$p_1(t) = c_{1,0} + c_{1,1} t + c_{1,2} t^2 + c_{1,3} t^3 + c_{1,4} t^4 + c_{1,5} t^5$
- 第二段：$[T,2T]$：$p_2(t) = c_{2,0} + c_{2,1} t + c_{2,2} t^2 + c_{2,3} t^3 + c_{2,4} t^4 + c_{2,5} t^5$

② 约束条件

第一段 $[0,T]$ 的边界条件：　　第二段 $[T,2T]$ 的边界条件：

$$\begin{cases} p_1(0)=x \\ \dot{p}_1(0)=0 \\ \ddot{p}_1(0)=0 \\ p_1(T)=y \\ \dot{p}_1(T)=v \\ \ddot{p}_1(T)=a \end{cases} \qquad \begin{cases} p_2(T)=y \\ \dot{p}_2(T)=v \\ \ddot{p}_2(T)=a \\ p_2(2T)=z \\ \dot{p}_2(2T)=0 \\ \ddot{p}_2(2T)=0 \end{cases}$$

③ 线性方程组：将约束条件展开为线性方程组。

对于第一段：

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^5 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{pmatrix} \begin{pmatrix} c_{1,0} \\ c_{1,1} \\ c_{1,2} \\ c_{1,3} \\ c_{1,4} \\ c_{1,5} \end{pmatrix} = \begin{pmatrix} x \\ 0 \\ 0 \\ y \\ v \\ a \end{pmatrix}$$

对于第二段（设 $\tau = t - T$）：

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^5 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{pmatrix} \begin{pmatrix} c_{2,0} \\ c_{2,1} \\ c_{2,2} \\ c_{2,3} \\ c_{2,4} \\ c_{2,5} \end{pmatrix} = \begin{pmatrix} y \\ v \\ a \\ z \\ 0 \\ 0 \end{pmatrix}$$

(2) 以状态 $[y, v]$ 在 T 时刻通过中间点，则轨迹需要几阶连续才能唯一确定两个五阶多项式的轨迹系数。

① 约束条件数量：

- 两个五阶多项式共有 12 个系数。
- 当前约束条件（共 10 个）：
  - 起点：$p_1(0)=x, \dot{p}_1(0)=0, \ddot{p}_1(0)=0$ （3 个）
  - 中间点：$p_1(T)=y, \dot{p}_1(T)=v, p_2(T)=y, \dot{p}_2(T)=v$ （4 个，连续性已满足）
  - 终点：$p_2(2T)=z, \dot{p}_2(2T)=0, \ddot{p}_2(2T)=0$ （3 个）

② 所需额外约束：需要 $12-10=2$ 个额外约束条件。

③（连续性要求）在 $t=T$ 处，需要保证：

- 加速度连续：$\ddot{p}_1(T)=\ddot{p}_2(T)$ （1 个）
- 加加速度连续：$\ddot{p}_1(T)=\ddot{p}_2(T)$ （1 个）

④【结论】需要轨迹在 $t=T$ 处保持 3 阶连续（即从位置到 3 阶导数全部连续），才能唯一确定两个五阶多项式的轨迹系数。

11.1.2 2023-2024 春夏

11.1.2.1 选择（10 道 × 2 分）

1. 欧拉角转四元数

【相关概念】

(1) 四元数定义

$$\boldsymbol{q} = [q_0, q_1, q_2, q_3]^T,$$

$$\|\boldsymbol{q}\| = \sqrt{q_0^2+q_1^2+q_2^2+q_3^2} = 1$$

- q_0：标量部分（实部）
- $[q_1, q_2, q_3]^T$：矢量部分（虚部）

(2) 欧拉角定义（ZYX 顺序）

- ψ：偏航角（绕 z 轴）
- θ：俯仰角（绕 y 轴）
- ϕ：滚转角（绕 x 轴）
- 旋转顺序：先 $\psi \rightarrow$ 再 $\theta \rightarrow$ 最后 ϕ

(3) ZYX 欧拉角转四元数

(4) 四元数转 ZYX 欧拉角

$$\begin{cases} c_\phi = \cos\left(\frac{\phi}{2}\right), & s_\phi = \sin\left(\frac{\phi}{2}\right) \\ c_\theta = \cos\left(\frac{\theta}{2}\right), & s_\theta = \sin\left(\frac{\theta}{2}\right) \\ c_\psi = \cos\left(\frac{\psi}{2}\right), & s_\psi = \sin\left(\frac{\psi}{2}\right) \end{cases} \Rightarrow \begin{cases} q_0 = c_\psi c_\theta c_\phi + s_\psi s_\theta s_\phi \\ q_1 = c_\psi c_\theta s_\phi - s_\psi s_\theta c_\phi \\ q_2 = c_\psi s_\theta c_\phi + s_\psi c_\theta s_\phi \\ q_3 = s_\psi c_\theta c_\phi - c_\psi s_\theta s_\phi \end{cases} \quad \begin{cases} \phi = \operatorname{atan2}(2(q_0q_1 + q_2q_3), 1 - 2(q_1^2 + q_2^2)) \\ \theta = \operatorname{asin}(2(q_0q_2 - q_3q_1)) \\ \psi = \operatorname{atan2}(2(q_0q_3 + q_1q_2), 1 - 2(q_2^2 + q_3^2)) \end{cases}$$

2. 搜索方法基于采样 or 搜索
   - （不知道在问啥）
3. 哪种形式不是常见的 3d 场景表征方式（高飞老师 ppt 里有）
   - （看具体选项）
4. snap 是位置的几阶导数
   - 4 阶
5. 发送给电调的协议是
   - DSHOT 协议
6. 哪种控制方法不可以做到轨迹跟踪
   - 回忆的历年卷中好像没给出正确选项
7. 下面哪项是飞控的合理输入
   - 合推力与三轴角加速度（？存疑，aby 说是这个）

11.1.2.2 判断（10 道 × 2 分）

1. RRT 是否概率完备且渐进最优
   - 错，是 RRT*
2. MPC 的缺点

   （随便抄几点，判断就凭直觉应该够了）

   (1) 计算复杂度高：在线优化求解耗时，尤其在大型系统中效率较低。

   (2) 依赖动态模型：需精确的系统模型，模型不匹配易导致控制效果下降。

   (3) 参数调优困难：不同任务需大量试验调整参数，且参数变化对控制性能影响显著。

   (4) 约束处理挑战：确保硬约束满足存在技术难点，尤其在采样周期内。

   (5) 初期应用限制：新系统启动时模型不完善，通常需先用简单控制策略。

   (6) 维护成本高：系统标准化和持续管理难度较大。
3. PID 的微分项类似于阻尼
   - 对
4. 实数的四元数是虚部为单位向量的四元数
   - （不知道在问啥）
5. MPC 预测窗口小，预测结果更可靠
   - 错

11.1.2.3 设计（12 分）

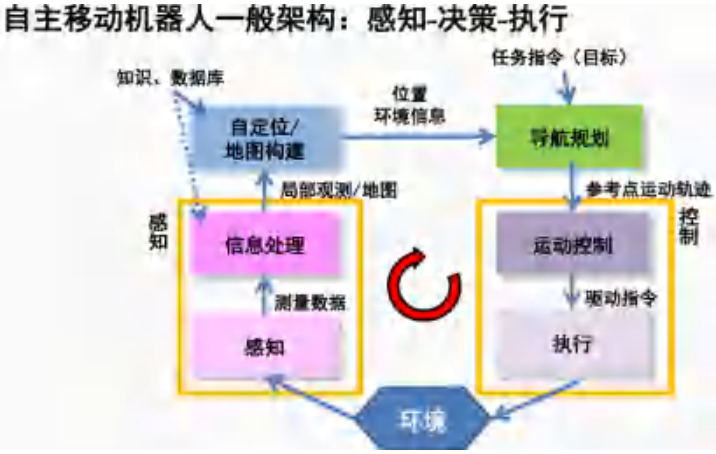
一个特技飞行无人机在户外，面临多种多样的难题，请从导航规划的角度讲讲可能面临的问题以及如何解决。

1. 动态轨迹规划
   (1) 问题
   - 特技动作需要高动态、大角度机动
   - 轨迹优化需要同时满足动力学约束和任务要求
   - 实时性要求高
   (2) 解决方案
   - 使用微分平坦特性进行快速轨迹生成
   - 采用多项式轨迹（5 阶或 7 阶）保证平滑性
   - 最小化时间/能量/Jerk 的优化目标
3. 定位与状态估计
   (1) 问题
   - GPS 在高速机动时精度下降
   - IMU 存在漂移和噪声
   - 环境遮挡导致定位失效
   (2) 解决方案
   - 多传感器融合（GPS + IMU + 视觉）
   - 扩展卡尔曼滤波（EKF）或无迹卡尔曼滤波（UKF）
   - 视觉惯性里程计（VIO）作为备份
   - SLAM 技术用于未知环境
5. 环境感知
   (1) 问题
   - 快速运动导致运动模糊
   - 光照变化影响视觉
   - 传感器量程限制
   (2) 解决方案
   - 多模态感知（激光雷达 + 相机 + 毫米波雷达）
   - 高帧率相机减少运动模糊
   - 深度学习进行障碍物检测与分类
   - 预测性感知提前规划
2. 动态避障
   (1) 问题
   - 环境中存在移动障碍物
   - 避障需要在线重规划
   - 感知延迟导致碰撞风险
   (2) 解决方案
   - 采用 RRT*、PRM 等快速采样算法
   - 使用人工势场法进行局部避障
   - 预测障碍物运动轨迹
   - 设置安全裕度和碰撞检测
4. 风扰动补偿
   (1) 问题
   - 户外风场不确定
   - 阵风影响轨迹跟踪精度
   - 特技动作对姿态控制要求极高
   (2) 解决方案
   - 风场估计与前馈补偿
   - 鲁棒控制器设计（H∞、滑模控制）
   - 自适应控制补偿未知扰动
   - 模型预测控制（MPC）处理约束
6. 多约束优化
   (1) 问题
   - 推力、角速度、加速度有物理限制
   - 避障与动作执行存在冲突
   - 电池续航时间有限
   (2) 解决方案
   - 带约束的非线性优化
   - 分层规划：全局路径 + 局部轨迹
   - 凸优化技术加速求解
   - 能量优化的轨迹设计

11.1.2.4 简答（6 道 × 8 分）

1. 简述自主移动机器人的定义，画出自主导航框架图，并解释各部分作用

   自主移动机器人（AMR）是一种具备自主移动和环境适应能力的智能设备，能够通过传感器和控制器独立完成任务，无需人工干预。



2. 八联通栅格地图 A* （1）给出最优的 h （2）写出从起点到终点每一个点的 openlist 和 cost

(1) 最优启发式函数 h

① 代价定义：八联通栅格中

- 直线移动代价：$d = 1$
- 对角线移动代价：$d_{\text{diag}} = \sqrt{2} \approx 1.414$

② 最优 h 函数：Octile 距离（对角启发式函数）

设当前点 (x, y)，目标点 (x_g, y_g)。

$$\begin{aligned} h(x,y) &= d \cdot \max(|\Delta x|, |\Delta y|) + \left(d_{\text{diag}} - d\right) \cdot \min(|\Delta x|, |\Delta y|) \\ &= \max(|\Delta x|, |\Delta y|) + \left(\sqrt{2} - 1\right) \cdot \min(|\Delta x|, |\Delta y|) \end{aligned} \tag{134}$$

其中 $\Delta x = x_g - x, \Delta y = y_g - y$。

(2) A* 核心：$f(n) = g(n) + h(n)$

① 使用 Octile 距离作为 h 时，A* 找到最优解

② OpenList 按 f 值排序，每次扩展 f 最小的节点

③ ClosedList 存储已扩展节点，避免重复搜索

3. 写出线性 MPC 控制器的一般函数形式，解释各个变量含义，解释滚动优化

(1) 线性 MPC 控制器的一般函数形式（实在没找到对应的，就用了二次型 cost）

在时刻 k，求解以下优化问题：

$$\begin{aligned} \min_{\boldsymbol{u}_k, \ldots, \boldsymbol{u}_{k+N-1}} \quad & J = \boldsymbol{x}_N^T \boldsymbol{P} \boldsymbol{x}_N + \sum_{i=0}^{N-1} \left(\boldsymbol{x}_{k+i}^T \boldsymbol{Q} \boldsymbol{x}_{k+i} + \boldsymbol{u}_{k+i}^T \boldsymbol{R} \boldsymbol{u}_{k+i} \right) \\ s.t. \quad & \boldsymbol{x}_{k+i+1} = \boldsymbol{A}\boldsymbol{x}_{k+i} + \boldsymbol{B}\boldsymbol{u}_{k+i} \quad i = 1, \ldots, N-1 \\ & \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k \quad i = 1, \ldots, N-1 \\ & \boldsymbol{x}_{\min} \le \boldsymbol{x}_{k+i} \le \boldsymbol{x}_{\max} \quad i = 1, \ldots, N-1 \\ & \boldsymbol{u}_{\min} \le \boldsymbol{u}_{k+i} \le \boldsymbol{u}_{\max} \quad i = 1, \ldots, N-1 \\ & \boldsymbol{x}_{k+0} = \boldsymbol{x}(k) \quad (\text{初始条件}) \end{aligned} \tag{135}$$

其中（半正定条件）$\boldsymbol{P} = \boldsymbol{P}^T \succeq 0, \boldsymbol{Q} = \boldsymbol{Q}^T \succeq 0, \boldsymbol{R} = \boldsymbol{R}^T \succ 0$。

(2) 变量含义

① 状态与输入

- x_k：当前时刻的状态
- u_k：当前时刻的控制输入
- k：当前时刻索引

② 预测时域

- N：预测时域长度（prediction horizon）
- x_{k+i}：从当前时刻 k 开始预测的第 i 步状态
- u_{k+i}：从当前时刻 k 开始的第 i 步控制输入

③ 代价函数权重

- $Q \in \mathbb{R}^{n\times n}$：状态偏差权重矩阵（半正定）
- $R \in \mathbb{R}^{m\times m}$：控制输入权重矩阵（正定）
- $P \in \mathbb{R}^{n\times n}$：终端状态权重矩阵（半正定）

④ 权重作用：

- Q 大：更注重状态跟踪精度
- R 大：更注重控制能量节省
- P：保证闭环稳定性

⑤ 约束

- $x_{\min}, x_{\max}$：状态约束
- $u_{\min}, u_{\max}$：输入约束

(3) 滚动优化

① 在线优化：在每个时刻 k，求解 N 步预测的优化问题，得到最优控制序列 $u_k^*, u_{k+1}^*, ..., u_{k+N-1}^*$。

② 执行首步：只应用第一个控制输入 $u(k) = u_k^*$。

③ 状态更新：系统执行控制后，状态更新为 $x(k+1)$。

④ 时域推进：时刻推进到 $k+1$，重复步骤 1-3

4. 推导无人机动力模型（1）混控矩阵（2）角速度与欧拉角微分的关系（3）小角度下用欧拉方程求出三轴力矩

(1) 混控矩阵

① 四旋翼配置

表 1　四旋翼“X”型配置，电机 1、3 逆时针旋转，电机 2、4 顺时针旋转

	电机 2	
电机 4	×	电机 1
	电机 3	

② 控制输入定义

- f_i：第 i 个电机产生的推力
- $M_i = k_M f_i$：第 i 个电机产生的反扭矩（k_M 为反扭矩系数）

③ 混控关系 - 总推力和三轴力矩：

$$\begin{cases} u_1 = f_1 + f_2 + f_3 + f_4 & \text{总推力} \\ u_2 = l(f_2 - f_4) & \text{滚转力矩（绕 x 轴）} \\ u_3 = l(f_3 - f_1) & \text{俯仰力矩（绕 y 轴）} \\ u_4 = M_1 - M_2 + M_3 - M_4 & \text{偏航力矩（绕 z 轴）} \end{cases}$$

其中 l 为电机到机体中心的距离。

④ 混控矩阵

$$\begin{pmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F l & 0 & -k_F l \\ -k_F l & 0 & k_F l & 0 \\ k_M & -k_M & k_M & -k_M \end{pmatrix}$$

(2) 角速度与欧拉角微分的关系

① 【角速度到欧拉角微分】

基本思想：将欧拉角的每次旋转分别对角速度的贡献叠加。

a. 航空常用的 ZYX 外旋顺序：

i. 先绕惯性系 Z 轴旋转 ψ（偏航，yaw）

ii. 再绕旋转后的 Y' 轴旋转 θ（俯仰，pitch）

iii. 最后绕机体 X 轴旋转 ϕ（滚转，roll）

b. 各旋转的角速度贡献

i. $\dot{\phi}$ 绕机体 x 轴旋转，直接在机体系中表示为 $\boldsymbol{\omega}_\phi = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix}_B$

ii. $\dot{\theta}$ 绕中间 y' 轴旋转。中间 y' 轴相对机体系只差一个滚转旋转 ϕ。将 y' 轴上的角速度变换到机体系：

$$\boldsymbol{\omega}_\theta = \boldsymbol{R}_{x(\phi)} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix}_{y'} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ \cos\phi\dot{\theta} \\ -\sin\phi\dot{\theta} \end{pmatrix}_B$$

iii. $\dot{\psi}$ 绕惯性系 Z 轴旋转。惯性系 Z 轴单位矢量为 $[0,0,1]_I^T$。

1. 先看 Z 轴经过俯仰旋转 $\boldsymbol{R}_{y(\theta)}$ 后的表示：

$$\boldsymbol{R}_{y(\theta)} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} -\sin\theta \\ 0 \\ \cos\theta \end{pmatrix}_{y'}$$

2. 所以 $\dot{\psi}$ 在 y' 坐标系中的角速度矢量为：$\begin{pmatrix} -\sin\theta\dot{\psi} \\ 0 \\ \cos\theta\dot{\psi} \end{pmatrix}_{y'}$

3. 再通过滚转旋转 $\boldsymbol{R}_{x(\phi)}$ 变换到机体系：

$$\boldsymbol{\omega}_\psi = \boldsymbol{R}_{x(\phi)} \begin{pmatrix} -\sin\theta\dot{\psi} \\ 0 \\ \cos\theta\dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} -\sin\theta\dot{\psi} \\ 0 \\ \cos\theta\dot{\psi} \end{pmatrix} = \begin{pmatrix} -\sin\theta\dot{\psi} \\ \sin\phi\cos\theta\dot{\psi} \\ \cos\phi\cos\theta\dot{\psi} \end{pmatrix}_B$$

c. 叠加所有贡献

i. 机体角速度是三者的矢量和：

$$\boldsymbol{\omega} = \boldsymbol{\omega}_\phi + \boldsymbol{\omega}_\theta + \boldsymbol{\omega}_\psi$$

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ \cos\phi\dot{\theta} \\ -\sin\phi\dot{\theta} \end{pmatrix} + \begin{pmatrix} -\sin\theta\dot{\psi} \\ \sin\phi\cos\theta\dot{\psi} \\ \cos\phi\cos\theta\dot{\psi} \end{pmatrix}$$

ii. 按分量整理：

$$\begin{cases} p = \dot{\phi} - \sin\theta\dot{\psi} \\ q = \cos\phi\dot{\theta} + \sin\phi\cos\theta\dot{\psi} \\ r = -\sin\phi\dot{\theta} + \cos\phi\cos\theta\dot{\psi} \end{cases}$$

iii. 写成矩阵形式：

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}$$

②【欧拉角微分到角速度】

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \dfrac{\sin\phi}{\cos\theta} & \dfrac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

5. 给了一段实验课的任务代码，改正其中不对的部分

（98 上给的当时的答案，仅供参考）

(1) des_acc 没有加 g

(2) 计算使用的 yaw 角是实际的 yaw 角而不是期望的

(3) 公式里正负号写反了

(4) pitch 和 roll 对应 y 和 x 对应错了

(5) u.q 没有经过姿态转换

11.1.3 2022-2023 春夏

11.1.3.1 选择题（10 道 × 2 分）+ 判断题 （10 道 × 2 分）（记不太清了所以写在一起）

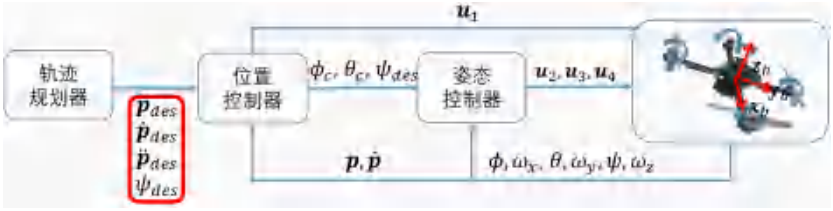
1. 本课程实验中发送给飞控的数据格式
   - （不知道）
2. 本课程实验树莓派的密码是
   - pi
3. 本课程实验中无人机的坐标系为（北东地/ECEF/…）
   - （不知道）
4. 本课程实验中电调的发送协议为
   - DSHOT 协议
5. 超高音速是大于多少马赫
   - 5
6. （判断）GPS 总共有 12 颗卫星
   - 错，应该是 24 个

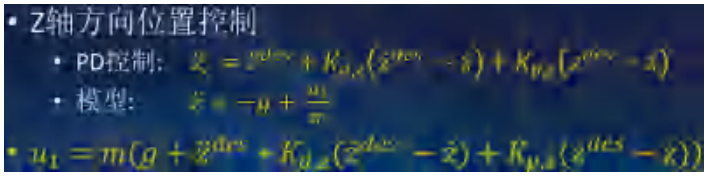
11.1.3.2 设计（12 分） - （可以参考后面的“各种框图”）

1. 需要一款能够在室内外环境下都可以实现定位与建图功能的无人机，请设计多传感器融合方案，说明各传感器的功能，绘制完整程序的流程图。
2. 在本课程实验制作无人机的基础上进行必要的改进，使得无人机可以在室外实现对于特定（如指定二维码）目标的跟踪，说明各模块的功能，绘制硬件结构图并注明数据流传输方式。

11.1.3.3 简答（6 道 × 8 分）

1. 画四旋翼无人机控制框图 推导 z 轴方向上的线性化模型以及 PD 控制算法





2. 有 A B 两个坐标系，先 A 坐标系绕 B 坐标系的 Z 轴旋转 30 度，沿 B 坐标系 Z 轴正方向移动 10 个单位，然后绕 A 坐标系的 Y 轴旋转 60 度，沿 A 坐标系的 X 轴正方向移动 5 个单位，已知 P 点在 A 坐标系中的坐标为[6 0 4]^T,求 P 点在 B 坐标系中的坐标（具体数据记不太清应该有区别 大概这个意思）

（可以模仿《机器人学 I》的这道例题的步骤）

例 3

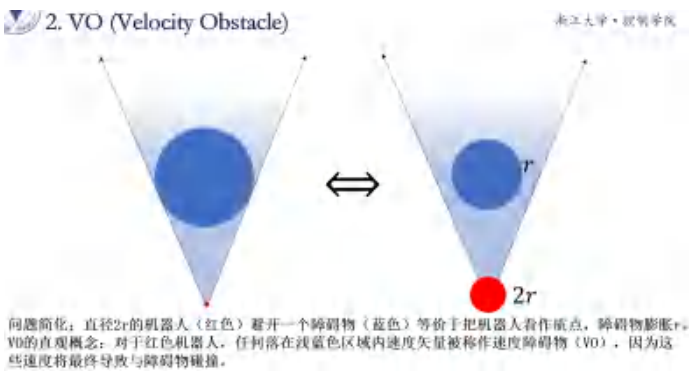
坐标系 $\{B\}$ 相对坐标系 $\{A\}$ 绕 $\hat{Z}_A$ 轴旋转 30 度，沿 $\hat{X}_A$ 轴平移 10 个单位，沿 $\hat{Y}_A$ 轴平移 5 个单位. 已知 ${}^BP = (3.0 \;\; 7.0 \;\; 0.0)^T$，求 ${}^AP$.

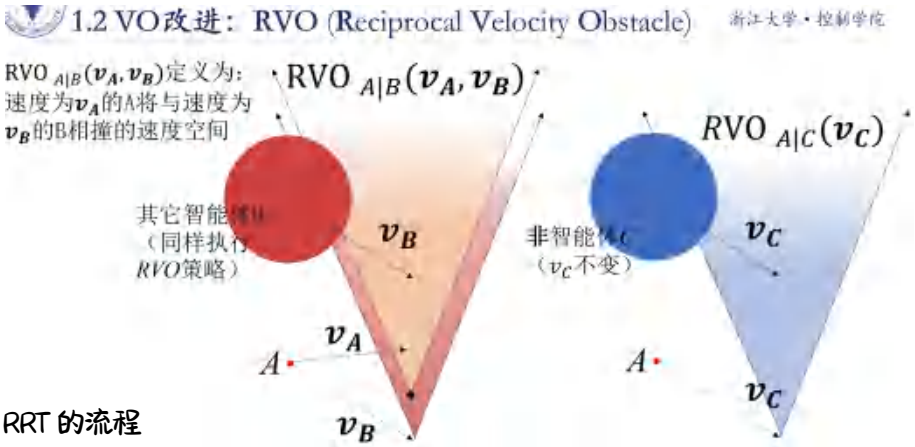
例 3 的解答

$$ {}^A_BR = \begin{pmatrix} \cos 30° & -\sin 30° & 0 \\ \sin 30° & \cos 30° & 0 \\ 0 & 0 & 1 \end{pmatrix}, {}^AO_B = \begin{pmatrix} 10 \\ 5 \\ 0 \end{pmatrix} \Rightarrow {}^A_BT = \left(\begin{array}{ccc|c} {}^A_BR & & & {}^AO_B \\ \hline 0 & 0 & 0 & 1 \end{array}\right) = \begin{pmatrix} 0.866 & -0.500 & 0.000 & 10.0 \\ 0.500 & 0.866 & 0.000 & 5.0 \\ 0.000 & 0.000 & 1.000 & 0.0 \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

$$ \Rightarrow \begin{pmatrix} {}^AP \\ 1 \end{pmatrix} = {}^A_BT \begin{pmatrix} {}^BP \\ 1 \end{pmatrix} = \begin{pmatrix} 9.098 \\ 12.562 \\ 0.000 \\ 1 \end{pmatrix} \Rightarrow {}^BP = (9.098 \;\; 12.562 \;\; 0.000)^T $$

3. VO RVO 画图
   - （详见高飞老师的“L4-Multi-agent”PPT，这里只随便截两张图举例）





1. 文字阐述 RRT 的流程

   (1) 在可行空间随机采样。

   (2) 找到当前树中离采样点最近的树节点。

   (3) 从最近的树节点“生长”出新的节点和树枝（路径）。

   (4) 如果此路径没有和环境发生碰撞，则将此节点和路径加到树中。

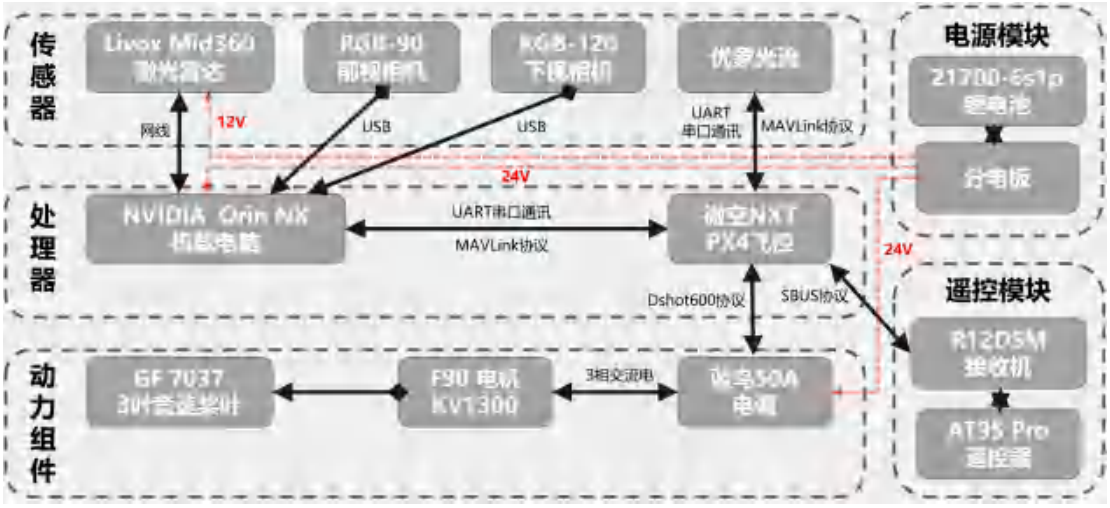
   (5) 重复 n 次采样，直到树生长到终点区域。

11.2 各种框图（仅供参考）

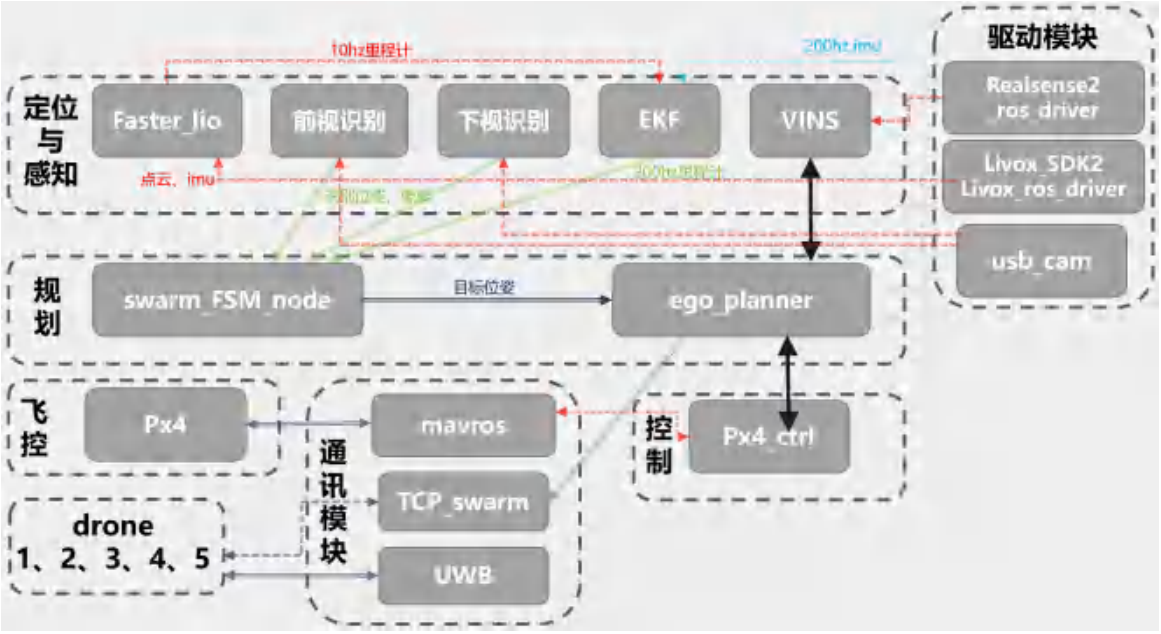
11.2.1 Lidar300-蜂群无人机 - 系统选型

类型	品牌型号	数量	备注
双目深度相机	Intel RealSense D430 module	1	选装
USB相机	90度FOV	1	下视相机
	120度FOV	1	前视相机
雷达	livox mid360	1	倾斜安装
机载电脑	nvidia orin nx 16g 核心板	1	
	达妙载板	1	
动力类	tmotor f90 1300kv	4	
	PC桨叶 乾丰7037	4	
飞塔	nxtpx4 +蓝鸟50A飞塔+优象光流	1	
遥控器	乐迪at9s+R12DSM+控电	1	
电池	定制21700-6S1P	1	
碳纤维板	机架上下中心板，机臂，雷达安装板	1套	
3D打印件	雷达倾斜安装板，相机安装板，电池舱起落架	1套	

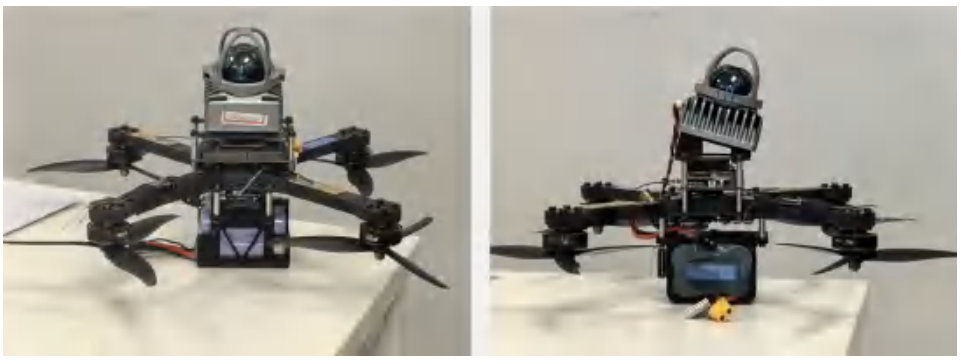
11.2.2 Lidar300-蜂群无人机 - 硬件框图



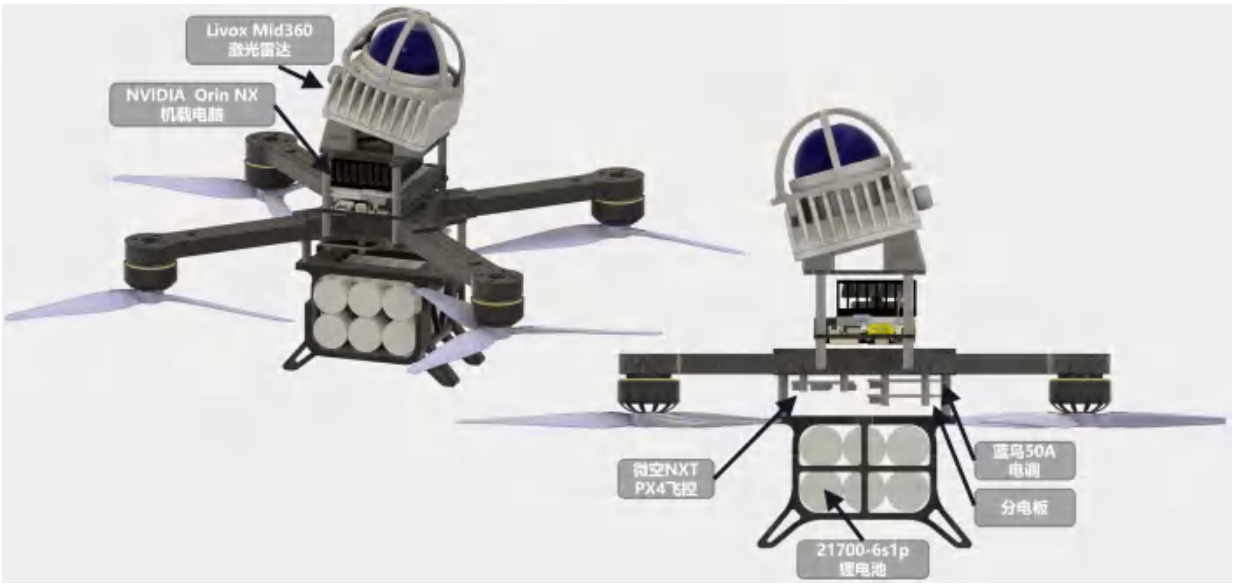
11.2.3 Lidar300-蜂群无人机 - 软件框图



11.2.4 Lidar300-蜂群无人机 - 实物图



11.2.5 Lidar300-蜂群无人机 - 设计图



11.3 实验相关

11.3.1 项目内容

1. Project 1：配置系统环境
   (1) 机载电脑 - 网络连接与调试使用
   (2) Linux
      ① 基本命令行操作
      ② 熟悉文件系统
      ③ 常用的软件安装
   (3) Robot Operating System (ROS)
      ① 安装与配置
      ② 完成 Tutorial，熟悉基本概念和操作
      ③ 参考网站：http://wiki.ros.org/
2. Project 2：组装四旋翼无人机
   (1) 材料清单
      ① **动力系统**：电机x4、电调x4、电池x1
      ② **控制系统**：飞控x1、遥控器x1、接收机x1
      ③ **机械结构**：机架x1、打印件、桨叶保护罩x4
   (2) 组装流程
      ① 固定、布线、焊接、测试……
      ② 实现手动遥控飞行
3. Project 3：自主悬停
   (1) 利用动作捕捉系统反馈的位置信息，基于 ROS 编写控制程序，实现无人机悬停。
   (2) 系统架构
      ① Motion Capture System：提供位置信息（Odometry）
      ② Workstation：运行控制程序
      ③ CONTROLLER：基于 ROS 的控制算法
      ④ 通过 Ethernet 连接各个组件

11.3.2 Project 1：配置系统环境

1. 系统配置
   (1) **硬件**：树莓派 4B (4GB RAM + 16GB SD 卡)
   (2) **系统**：预装 ubuntu mate 20.04.1 64 位系统
   (3) **下载地址**：https://ubuntu-mate.org/download/arm64
   (4) **用户名**：fast+队伍编号（如 fast0）

(5) 初始密码：pi

(6) 注意：若重装系统，仍保持此命名规则，避免冲突

2. 网络连接配置

(1) 网络连接方式

① 本节课使用校园网或手机热点

② 后面实验课时会使用实验室路由器

(2) 远程桌面

① 建议使用 NoMachine

② 安装命令：

```shell
sudo dpkg -i 包名.deb
```

③ 下载地址：https://www.nomachine.com/download

④ 树莓派版本：https://downloads.nomachine.com/linux/?id=29&distro=Raspberry

3. 软件源更换（参考网址：http://mirrors.zju.edu.cn/docs/ubuntu/）

(1) 打开配置文件

```shell
sudo nano /etc/apt/sources.list
```

- 其他常用编辑器如 gedit 需要下载

(2) 修改内容为

```shell
deb https://mirrors.zju.edu.cn/ubuntu/ focal main restricted universe multiverse
deb https://mirrors.zju.edu.cn/ubuntu/ focal-updates main restricted universe multiverse
deb https://mirrors.zju.edu.cn/ubuntu/ focal-backports main restricted universe multiverse
deb https://mirrors.zju.edu.cn/ubuntu/ focal-security main restricted universe multiverse
```

(3) 更新软件源

```shell
sudo apt update
```

4. 网络工具安装

(1) 下载 net-tools

```shell
sudo apt install net-tools
```

(2) NoMachine 连接：使用 `ifconfig` 命令获取树莓派 IP，在笔记本中打开 NoMachine，搜索 IP（或根据用户名）进行连接。

(3) SSH 连接（实验时必需）- 在笔记本终端使用命令：

```shell
ssh XXX(host name)@xxx.xxx.xxx.xxx(host IP)
```

5. ROS 安装

(1) 安装步骤

① 进入官网：http://wiki.ros.org/noetic/Installation/Ubuntu

② 复制其中的命令走一遍流程

③ 注意：步骤 1.4 中，安装 Desktop-Full install

④ 注意：步骤 1.6 中，1.6.1 步骤不需要做

(2) 测试 ROS 安装

① 终端 1：

```shell
roscore
```

② 终端 2：

```shell
rosrun turtlesim turtlesim_node
```

③ 终端 3：

```shell
rosrun turtlesim turtle_teleop_key
```

(3) 配置 VSCode 远程 SSH 连接

(4) 使用鱼香 ROS 一键安装脚本：

```shell
wget http://fishros.com/install -O fishros && . fishros
```

6. 实用软件推荐

(1) Terminator（可分屏终端，课后自己学习安装，本课不做要求）

```shell
sudo apt-get install terminator
```

(2) PlotJuggler（时序数据可视化，用于调试）

① 参考链接：https://blog.csdn.net/qq_39779233/article/details/106478608

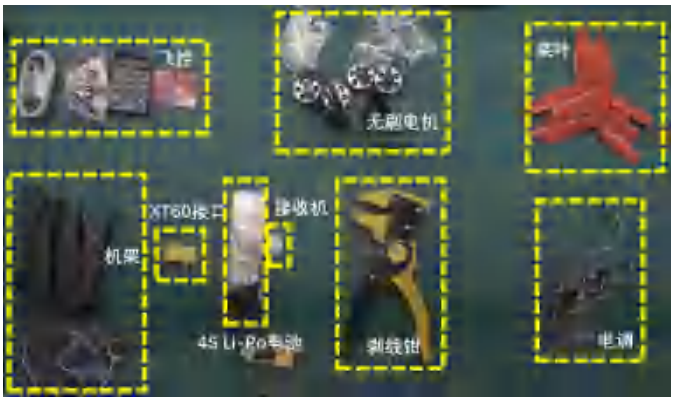
(3) htop（系统状态监测与进程管理，课后自己学习安装，本课不做要求）

```shell
sudo apt-get install htop
```

(4) SSH, NoMachine/VNC viewer

① 配置 SSH 教程：https://www.cnblogs.com/livelab/p/13033175.html

11.3.3 Project 2: 组装四旋翼无人机



1. 材料清单

(1) 动力系统：电机 x 4、电调 x 4、电池 x 1

(2) 控制系统：飞控 × 1、遥控器 × 1、接收机 × 1

(3) 机械结构：机架 × 1、打印件、桨叶保护罩 × 4

2. 组件认识

(1) 飞控（Flight Controller）

① 定义：飞行控制器，用于根据输入指令解算电机推力

② 内置传感器：IMU（惯性测量单元）、气压计、磁罗盘

③ 接口说明

- 遥控器输入：接收遥控信号
- 电机/舵机输出：控制电机转速
- 电源输入：供电接口

④ 型号：V5 nano / V5+

(2) 电调（ESC - Electronic Speed Control）

① 定义：电子调速器，相当于电机驱动器

② 关键参数：最大通过电流、DSHOT 协议支持

③ 接口说明

- 电机输出：连接无刷电机
- 电源输入：接收主电源
- 飞控信号输入：接收 PWM/DSHOT 信号

(3) 无刷电机

① 特点：无人机上常用无刷电机

② 关键参数

- KV 值：电机转速常数（RPM/V）
- 能效比：效率指标

(4) 桨叶：叶片数、尺寸（直径×螺距）

(5) 电池

① 类型：4S Li-Po 电池

② 关键参数

- 电压：电池芯数（4S = 14.8V 标称电压）
- 容量：mAh（毫安时）

3. 组装步骤

(1) Step 1：焊接 XT60 接头与硅胶线 - 硅胶线大约留 3cm，焊接时插上一个母头，锡要多给，避免虚焊

(2) Step 2：为主电源焊盘上锡，贴 3M 胶 - 焊盘锡尽量多给

(3) Step 3：粘贴插头，拔掉线壳，焊接插头 - 焊接前先给导线上尽量多的锡，待导线与焊盘焊牢后，在焊盘处继续补锡，直至焊锡充分与导线接触，注意正负极

(4) Step 4：为电调焊盘上锡 - 尽量多给锡

(5) Step 5：焊接电调电源线 - 电源线朝内，不要碰到螺丝连接处，注意正负极

(6) Step 6：连接机臂与下板 - 使用螺丝固定机臂到下板。

(7) Step 7：剪短电机电源线，上锡 - 线留大约一半，锡多上

(8) Step 8：电调电源处上锡 - 先把电调在机臂里走走线，正反触点都可以用，锡主要不要跟电调芯片接触

(9) Step 9：焊接电调与电机 - 将电调的三根输出线焊接到电机上。

(10) Step 10：完成所有机臂 - 重复以上步骤，完成四个机臂的组装。

(11) Step 11：粘贴飞控，插上电调信号线与接收机 - 飞控四角的海绵胶不要太大，注意飞控方向，注意电机顺序，注意正负极

【电机顺序与转向】

- 机头朝向为前方
- 1 号电机：前右，逆时针（CCW）
- 2 号电机：后左，逆时针（CCW）
- 3 号电机：前左，顺时针（CW）
- 4 号电机：后右，顺时针（CW）

【接线顺序】

- 电调信号线按 1-4 顺序插入飞控
- 接收机连接到飞控 SBUS/PPM 接口

(12) Step 12：焊接飞控电源模块，连接飞控 - 焊接前先给电源线上锡，注意正负极

(13) Step 13：检查短路!!! - 检查主电源是否短路，检查主电源正负是否与所有电调正负相连通，使用万用表测量，确保无短路

(14) Step 14：配对遥控器 - 将遥控器开机，飞机通电（电池或 Type-C），长按接收机的 ID SET 键，等待指示灯常亮，表示配对成功

(15) Step 15：安装起落架 - 把起落架柱子插进去，操作简单。

(16) Step 16：理线 - 装上桨后桨不打到线，善待强迫症 😄

(17) Step 17：设置飞控参数

① 下载 QGroundControl（win10, ubuntu）：在自己电脑上安装，不要在树莓派上

② Type-C 连接飞控与电脑，更新飞控固件 - 点击固件选项，按要求更新

③ 设置机架 - 选择“Generic 250 Racer”或相应机架类型

④ 校准传感器 - 按照界面提示依次校准

⑤ 校准遥控器 - 按照界面右侧图片指示操作

⑥ 设置飞行模式

- Mode Channel: Channel 5
- 飞行模式 1-6: 全部设置为 Stabilized

⑦ 设置电源 - 填入 4S 后点计算

⑧ 进入参数设置界面

⑨ 搜索并设置以下参数：

```shell
CBRK_IO_SAFETY = 22027
CBRK_USB_CHK = 197848
MAV_1_CONFIG = TELEM 2
```

```
SYS_USE_IO = 0
DSHOT_CONFIG = DShot600
```

(18) Step 18：调整电机转向

① 断开 Type-C 连接

② 用电池给飞机上电

③ 解锁飞机（不要带桨！！！！！！）

④ 用手确认哪些电机转向反了

⑤ 断电，连接电脑

⑥ 进入 Mavlink Console

⑦ 根据反转电机编号，依次输入指令，以 1 号电机反转了为例：

```shell
dshot reverse -m 1
dshot save -m 1
```

(19) Step 19：起飞！

4. 常见问题与解决方案

(1) 电机不转：检查电调连接、检查电池电量、检查飞控参数设置

(2) 飞机倾斜：检查飞控安装方向、重新校准传感器、检查电机转向

(3) 遥控器失联：重新配对接收机、检查天线连接、检查遥控器电量

(4) 电机抖动：检查桨叶安装、检查电调参数、检查电机螺丝

5. 维护与保养

(1) 定期检查：螺丝是否松动、桨叶是否损坏、电池电量、焊点是否牢固

(2) 飞行后：检查机架是否有裂纹、清洁机身、充电保养电池

(3) 存储：电池半电存储、避免潮湿环境、定期通电检查

11.3.4 Project 3: 动捕下实现自主悬停

1. Part 1: 控制器算法及仿真

(1) 控制器概述

① 控制器的作用是根据当前状态和期望状态，计算出控制指令：待跟踪的状态 + 当前的位姿、速度 -> 控制器 -> 控制指令（电机转速、推力）

② 本节介绍两种控制器：

a. 线性控制器 (Linear Controller)

b. SE(3)控制器 (Geometric SE(3) Controller)

c. 两者都是串级控制，外环控制位置，内环控制姿态。

d. 相比于 Linear Controller，SE(3) Controller 的特点：

i. 根据误差计算期望的推力向量，用其根据微分平坦计算姿态误差

ii. 推力控制输入是期望的推力向量投影到当前测量的姿态 Z 轴上的分量

(2) 线性控制器

① 动力学模型

a. 合力与力矩：

$$\begin{aligned}\boldsymbol{F} &= \boldsymbol{F}_1+\boldsymbol{F}_2+\boldsymbol{F}_3+\boldsymbol{F}_4-mg\boldsymbol{z}_w\\ \boldsymbol{M} &= \boldsymbol{r}_1\times\boldsymbol{F}_1+\boldsymbol{r}_2\times\boldsymbol{F}_2+\boldsymbol{r}_3\times\boldsymbol{F}_3+\boldsymbol{r}_4\times\boldsymbol{F}_4+\boldsymbol{M}_1+\boldsymbol{M}_2+\boldsymbol{M}_3+\boldsymbol{M}_4\end{aligned} \tag{136}$$

b. 单桨推力与力矩：

$$\begin{aligned}\boldsymbol{F}_i &= \left[0,0,k_F\omega_i^2\right]^T\\ \boldsymbol{M}_i &= \left[0,0,\pm k_M\omega_i^2\right]^T\end{aligned} \tag{137}$$

c. 混控矩阵：

$$\boldsymbol{u}=\begin{pmatrix}k_F & k_F & k_F & k_F\\ 0 & k_FL & 0 & -k_FL\\ -k_FL & 0 & k_FL & 0\\ k_M & -k_M & k_M & -k_M\end{pmatrix}\begin{pmatrix}\omega_1^2\\ \omega_2^2\\ \omega_3^2\\ \omega_4^2\end{pmatrix} \tag{138}$$

② 线性化（平衡悬停态 $\varphi_0\sim 0,\theta_0\sim 0,u_{1,0}\sim mg$）

a. 牛顿方程：

$$m\ddot{\boldsymbol{p}}=\begin{pmatrix}0\\0\\-mg\end{pmatrix}+\boldsymbol{R}\begin{pmatrix}0\\0\\F_1+F_2+F_3+F_4\end{pmatrix}\Rightarrow\begin{cases}\ddot{\boldsymbol{p}}_1=\ddot{x}=g(\theta\cos\psi+\varphi\sin\psi)\\ \ddot{\boldsymbol{p}}_2=\ddot{y}=g(\theta\sin\psi-\varphi\cos\psi)\\ \ddot{\boldsymbol{p}}_3=\ddot{z}=-g+\dfrac{u_1}{m}\end{cases} \tag{139}$$

b. 欧拉角微分：

$$\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}=\begin{pmatrix}c\theta & 0 & -c\varphi s\theta\\ 0 & 1 & s\varphi\\ s\theta & 0 & c\varphi c\theta\end{pmatrix}\begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix}\xrightarrow{\text{平衡悬停态}}\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}=\begin{pmatrix}\dot{\varphi}\\ \dot{\theta}\\ \dot{\psi}\end{pmatrix} \tag{140}$$

c. 欧拉方程：

$$\boldsymbol{I}\cdot\begin{pmatrix}\ddot{\varphi}\\ \ddot{\theta}\\ \ddot{\psi}\end{pmatrix}+\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}\times\boldsymbol{I}\cdot\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}=\begin{pmatrix}l(F_2-F_4)\\ l(F_3-F_1)\\ M_1-M_2+M_3-M_4\end{pmatrix}=\boldsymbol{u}_2 \tag{141}$$

③ 控制律设计

a. 位置控制器

i. PID 控制：
$$\ddot{\boldsymbol{p}}_{i,c}=\ddot{\boldsymbol{p}}_i^{des}+K_{d,i}(\dot{\boldsymbol{p}}_i^{des}-\dot{\boldsymbol{p}}_i)+K_{p,i}(\boldsymbol{p}_i^{des}-\boldsymbol{p}_i) \tag{142}$$

ii. 根据模型计算控制量：

$$u_1=m(g+\ddot{\boldsymbol{p}}_{3,c}),\quad \varphi_c=\frac{1}{g}(\ddot{\boldsymbol{p}}_{1,c}\sin\psi-\ddot{\boldsymbol{p}}_{2,c}\cos\psi),\quad \theta_c=\frac{1}{g}(\ddot{\boldsymbol{p}}_{1,c}\cos\psi+\ddot{\boldsymbol{p}}_{2,c}\sin\psi) \tag{143}$$

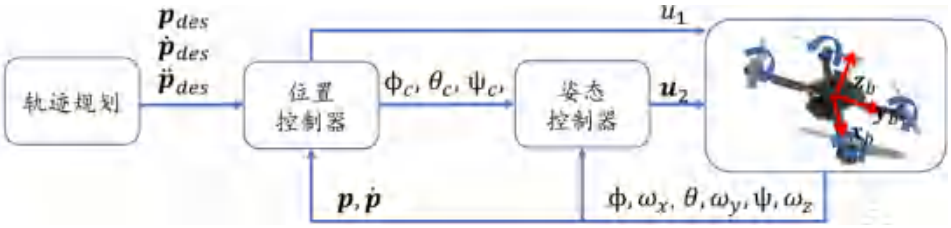
iii. 注意：这些 ψ 是当前测量的 yaw，不是期望的 yaw。

b. 姿态控制器

i. PID 控制：
$$\begin{pmatrix}\ddot{\varphi}_c\\ \ddot{\theta}_c\\ \ddot{\psi}_c\end{pmatrix}=\begin{pmatrix}K_{p,\varphi}(\varphi_c-\varphi)+K_{d,\varphi}(\dot{\varphi}_c-\dot{\varphi})\\ K_{p,\theta}(\theta_c-\theta)+K_{d,\theta}\left(\dot{\theta}_c-\dot{\theta}\right)\\ K_{p,\psi}(\psi_c-\psi)+K_{d,\psi}\left(\dot{\psi}_c-\dot{\psi}\right)\end{pmatrix} \tag{144}$$

ii. 根据欧拉方程计算力矩：
$$\boldsymbol{u}_2=\boldsymbol{I}\cdot\begin{pmatrix}\ddot{\varphi}_c\\ \ddot{\theta}_c\\ \ddot{\psi}_c\end{pmatrix}+\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix}\times\boldsymbol{I}\cdot\begin{pmatrix}\omega_x\\ \omega_y\\ \omega_z\end{pmatrix} \tag{145}$$

④ 控制框图



(3) SE(3)控制器 - 基于微分平坦特性：四旋翼的状态和输入可以写成四个挑选出来的输出变量及其有限阶导数的代数函数。

① 理想情况

$$m\boldsymbol{a} = \boldsymbol{F} - mg\boldsymbol{z}_w, \quad \boldsymbol{F}_{des} = mg\boldsymbol{z}_W + m\boldsymbol{a}_{des}, \quad \boldsymbol{z}_{b,des} = \frac{\boldsymbol{F}_{des}}{\|\boldsymbol{F}_{des}\|} \tag{146}$$

② 现实情况

a. 定义位置和速度误差：

$$\boldsymbol{e}_p = \boldsymbol{p} - \boldsymbol{p}_{des}, \quad \boldsymbol{e}_v = \boldsymbol{v} - \boldsymbol{v}_{des} \tag{147}$$

b. 计算期望推力：

$$\boldsymbol{F}_{des} = -K_p\boldsymbol{e}_p - K_v\boldsymbol{e}_v + \underbrace{mg\boldsymbol{z}_w + m\boldsymbol{a}_{des}}_{\text{前馈}} \tag{148}$$

c. 投影到当前姿态的 z 轴：

$$u_1 = \boldsymbol{F}_{des} \cdot \boldsymbol{z}_B \tag{149}$$

③ 姿态控制

a. 计算期望的 z_B：

$$\boldsymbol{z}_{B,des} = \frac{\boldsymbol{F}_{des}}{\|\boldsymbol{F}_{des}\|} \tag{150}$$

b. 根据微分平坦计算 x_B, y_B：

$$\begin{aligned} \boldsymbol{x}_{C,des} &= [\cos(\psi_d), \sin(\psi_d), 0]^T \\ \boldsymbol{y}_{B,des} &= \frac{\boldsymbol{z}_{B,des} \times \boldsymbol{x}_{C,des}}{\|\boldsymbol{z}_{B,des} \times \boldsymbol{x}_{C,des}\|} \\ \boldsymbol{x}_{B,des} &= \boldsymbol{y}_{B,des} \times \boldsymbol{z}_{B,des} \\ \boldsymbol{R}_{B,des} &= [\boldsymbol{x}_{B,des} \quad \boldsymbol{y}_{B,des} \quad \boldsymbol{z}_{B,des}] \end{aligned} \tag{151}$$

c. 定义姿态误差（$\vee$ 表示 vee 映射：从 so(3)到 $\mathbb{R}^3$）：

$$\boldsymbol{e}_R = \frac{1}{2}(\boldsymbol{R}_{des}^T\,{}^W\boldsymbol{R}_B - {}^W\boldsymbol{R}_B\boldsymbol{R}_{des}^T)^\vee \tag{152}$$

d. 定义角速度误差：

$$\boldsymbol{e}_\omega = {}^B[\omega_{BW}] - {}^B[\omega_{BW,T}] \tag{153}$$

e. 计算力矩：

$$[u_2, u_3, u_4]^T = -K_R\boldsymbol{e}_R - K_\omega\boldsymbol{e}_\omega \tag{154}$$

(4) MATLAB 仿真

① 启动程序 `test_trajectory.m`，得到位置、速度、角度的实际量和控制量曲线。

② 算法实现：`controller.m`

- 输入：当前状态 `s`，期望状态 `s_des`
- 已知量：质量 `m`，重力加速度 `g`，转动惯量 `I`
- 输出：力 `F`、力矩 `M`

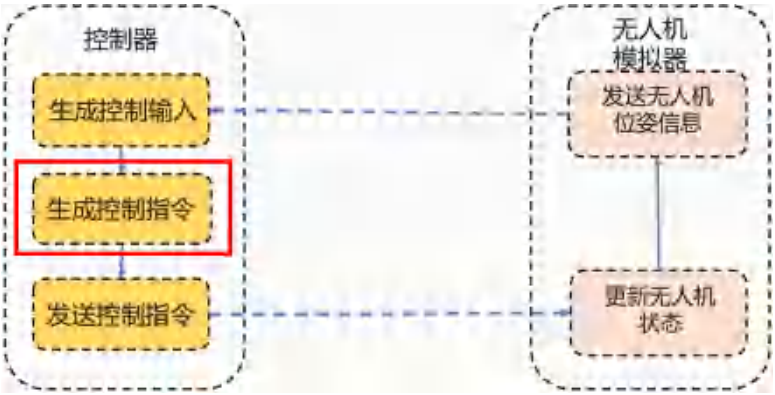
(5) ROS 仿真

① 工作空间设置

```shell
# 解压后进入工作空间
cd controller_sim
# 编译
```

```
catkin_make
# 配置工作空间
source devel/setup.bash
# 运行程序
roslaunch so3_quadrotor_simulator simulator_example.launch
```

② 系统架构



③ 性能分析

a. 录制 rosbag：

```
rosbag record /sim/odom
```

b. 使用 PlotJuggler 数据分析软件：https://github.com/facontidavide/PlotJuggler

2. Part 2: 完善飞机硬件

(1) 树莓派供电系统

① Step 1: 为 5V 电源模块上锡 - 烙铁头别碰到电阻电容等元件

② Step 2: 裁剪适当长度的硅胶线，正负各两条 - 要点：用细一点的线，长度大约在 10cm

③ Step 3: 为长线套上壳子，为 TYPE-C 套上壳子

④ Step 4: 为硅胶线上锡

⑤ Step 5: 为 TYPE-C 头上锡，焊接 - 要点：注意正负，只需要焊接两个触点，不要碰到其余触点

⑥ Step 6: 套上壳子，缠上绝缘胶带

⑦ Step 7: 焊接 5V 模块 - 要点：注意输入输出与正负

⑧ Step 8: 缠绝缘胶带

⑨ Step 9: 焊接 5V 模块到主电源处 - 要点：可以适当提高烙铁温度到 400 度左右来加快融化电源处的大块锡

⑩ Step 10: 确认树莓派可以正常开机后，固定 5V 模块

⑪ Step 11: 在电源处重新缠上绝缘胶带

(2) 树莓派固定

① Step 12: 将树莓派和连接件用扎带固定

② Step 13: 连接件和机架用螺丝固定 - 要点：注意树莓派的安装方向，确保能插上供电线

③ Step 14: 借助连接件固定一下供电线/接收机等器件

(3) 粘贴动捕 marker

① **Step 15**: 粘贴动捕球 marker - 要点：动捕球间距离尽可能大，保持一定非对称性

② **Step 16**: 用数据线连接树莓派的 USB 口和飞控的 TYPE-C 口

3. Part 3: 动捕测试

(1) WiFi 信息 - **SSID**: CSC301，**Password**: 11223344

(2) 动捕系统设置

① 相机标定

a. 打开 Camera Calibration 界面

b. 选择 Start Wanding

c. 使用相机标定杆在场地中跳舞（Just Dance）

d. 采样数过 3000 即可

e. 点击 Apply 应用标定结果

② 地面识别

a. 切换到 Ground Plane 界面

b. 将地面标定杆放置在场地中心

c. 点击 Set Ground Plane

d. 设置 Vertical Offset 为 45mm（标定杆高度）

③ 创建刚体

a. 进入 Layout 界面

b. 点击 Create 按钮

c. 在 3D 视图中选中 marker

d. 给刚体命名（注意小组间区分）

(3) VRPN 客户端配置

① 下载并安装 VRPN

```shell
cd ~/catkin_ws/src
git clone https://github.com/clearpathrobotics/vrpn_client_ros.git
sudo apt-get install ros-noetic-vrpn
cd ~/catkin_ws
```

```
5 catkin_make
```

② 查看电脑 IP 地址：在 Motive 软件中查看 VRPN Broadcast Port（默认 3883）

③ 启动 VRPN 客户端 - 注意：替换为实际的电脑 IP 地址

```
1 roslaunch vrpn_client_ros sample.launch server:=192.168.43.195:3883
```
Shell

④ 读取位姿数据

```
1 rostopic echo /vrpn_client_node/刚体名称
```
Shell

4. 测试检查

(1) 检查动捕球是否都被相机识别

(2) 检查所有动捕相机是否正常工作

(3) 检查刚体识别是否稳定

(4) 注意小组间区分刚体名称

(5) 移动飞机，查看获得的位姿是否连续光滑

(6) 确认位姿数据更新频率正常（通常 100-200Hz）

11.3.5 坐标系的变换

不同于仿真所得与所发出的四旋翼角度状态(角位移、角速度、角加速度)均处于世界坐标系下,实机实验中存在着三个坐标系,即“机体坐标系”、“世界坐标系”与“IMU 北东地坐标系”。因此变带来了有些复杂的坐标变换,包括:

1. 课程描述中,期望设定值 s_{des} 中 $x_{des}, y_{des}, z_{des}, \varphi_{des}, \theta_{des}, \psi_{des}, x, y, z$ 处在世界坐标系下,w_x, w_y, w_z 为机体系下机体旋转的即时角速度;
2. 实机内环接口为 $[u_1, \varphi_{des}, \theta_{des}, \psi_{des}]$,其中,$u_1$ 需处于动捕世界坐标系下,而 $[\varphi_{des}, \theta_{des}, \psi_{des}]$ 需处在 IMU 北东地坐标系下。
3. 实机中可以获得的姿态有里程计与 IMU 进行 EKF 融合后的里程计姿态表示 ${}^{W}R_{B}$、从 IMU 直接获得的北东地坐标系下的 IMU 姿态表示 ${}^{E}R_{B}$,与我们可以经过线性化公式或微分平坦变换得出的世界坐标系下的期望姿态表示 ${}^{W}R_{B'}$。

结合以上,最终求得姿态转换公式为:

$$ {}^{E}R_{B'} = {}^{E}R_{B}({}^{W}R_{B})^{-1}{}^{W}R_{B'} \tag{155} $$

事实上,北东地坐标系与世界坐标系之间的转换应为定值,但由于每次上电之后 IMU 都会初始化,使得这个固定值无法在多次测试中始终保持恒定。可以采取了如同以上公式一样在线求取北东地坐标系与世界坐标系旋转矩阵的方法,并选择将此旋转矩阵打印,待到矩阵稳定时放飞四旋翼,以避免初始状态 IMU 与动捕在进行 EKF 时可能带来的一些滞后反应。

11.3.6 代码

实机需要编写的示例代码（real_ws/src/px4ctrl/src/linear_control.cpp）

C++

```cpp
#include "linear_control.h"
#include <iostream>
#include <ros/ros.h>
LinearControl::LinearControl(Parameter_t &param) : param_(param)
{
  resetThrustMapping();
}
/*
  compute u.thrust and u.q, controller gains and other parameters are in param_
*/
quadrotor_msgs::Px4ctrlDebug
LinearControl::calculateControl(const Desired_State_t &des,
    const Odom_Data_t &odom,
    const Imu_Data_t &imu,
    Controller_Output_t &u)
{
  /* WRITE YOUR CODE HERE */
      //compute disired acceleration
      Eigen::Vector3d des_acc(0.0, 0.0, 0.0);

      //supposed to be readonly, compute thrust by acc
      u.thrust = computeDesiredCollectiveThrustSignal(des_acc);

      //compute control attitude in the BODY frame
      u.q = Eigen::Quaterniond(1.0,0.0,0.0,0.0);
  /* WRITE YOUR CODE HERE */

  //used for debug
  debug_msg_.des_p_x = des.p(0);
  debug_msg_.des_p_y = des.p(1);
  debug_msg_.des_p_z = des.p(2);

  debug_msg_.des_v_x = des.v(0);
  debug_msg_.des_v_y = des.v(1);
  debug_msg_.des_v_z = des.v(2);

  debug_msg_.des_a_x = des_acc(0);
  debug_msg_.des_a_y = des_acc(1);
  debug_msg_.des_a_z = des_acc(2);

  debug_msg_.des_q_x = u.q.x();
  debug_msg_.des_q_y = u.q.y();
  debug_msg_.des_q_z = u.q.z();
  debug_msg_.des_q_w = u.q.w();

  debug_msg_.des_thr = u.thrust;

  // Used for thrust-accel mapping estimation
```

```
49    timed_thrust_.push(std::pair<ros::Time, double>(ros::Time::now(), u.thrust));
50    while (timed_thrust_.size() > 100)
51    {
52      timed_thrust_.pop();
53    }
54    return debug_msg_;
55  }
56  /*
57    compute throttle percentage
58  */
59  double
60  LinearControl::computeDesiredCollectiveThrustSignal(
61      const Eigen::Vector3d &des_acc)
62  {
63    double throttle_percentage(0.0);
64
65    /* compute throttle, thr2acc has been estimated before */
66    throttle_percentage = des_acc(2) / thr2acc_;
67
68    return throttle_percentage;
69  }
70  bool
71  LinearControl::estimateThrustModel(
72      const Eigen::Vector3d &est_a,
73      const Parameter_t &param)
74  {
75    ros::Time t_now = ros::Time::now();
76    while (timed_thrust_.size() >= 1)
77    {
78      // Choose data before 35~45ms ago
79      std::pair<ros::Time, double> t_t = timed_thrust_.front();
80      double time_passed = (t_now - t_t.first).toSec();
81      if (time_passed > 0.045) // 45ms
82      {
83        // printf("continue, time_passed=%f\n", time_passed);
84        timed_thrust_.pop();
85        continue;
86      }
87      if (time_passed < 0.035) // 35ms
88      {
89        // printf("skip, time_passed=%f\n", time_passed);
90        return false;
91      }
92
93      /***********************************************************/
94      /* Recursive least squares algorithm with vanishing memory */
95      /***********************************************************/
96      double thr = t_t.second;
```

```
97      timed_thrust_.pop();
98
99      /*********************************/
100     /* Model: est_a(2) = thr1acc_ * thr */
101     /*********************************/
102     double gamma = 1 / (rho2_ + thr * P_ * thr);
103     double K = gamma * P_ * thr;
104     thr2acc_ = thr2acc_ + K * (est_a(2) - thr * thr2acc_);
105     P_ = (1 - K * thr) * P_ / rho2_;
106     //printf("%6.3f,%6.3f,%6.3f,%6.3f\n", thr2acc_, gamma, K, P_);
107     //fflush(stdout);
108
109     debug_msg_.thr2acc = thr2acc_;
110     return true;
111   }
112   return false;
113 }
114 void
115 LinearControl::resetThrustMapping(void)
116 {
117   thr2acc_ = param_.gra / param_.thr_map.hover_percentage;
118   P_ = 1e6;
119 }
```

11.4 概念要点总结

11.4.1 空中机器人的定义与形态

11.4.1.1 固定翼飞行器

1. 工作原理：靠螺旋桨或者涡轮发动机产生的推力作用飞机向前飞行，主要升力来自机翼与空气的相对运动。
2. 特点：飞行高度高，速度快，载重大。
3. 应用领域：运输、测绘等。
4. 组成结构（5 个部分）：机体结构、航电系统、动力系统、起降系统、地面控制站

11.4.1.2 旋翼飞行器

1. 工作原理：旋翼升力平衡重力；改变旋翼的转速控制飞行器的平稳和姿态；可以悬停，或在一定速度范围内以任意方式飞行；可视为一个空中飞行平台。
2. 特点：操控简单、可靠性高、模块易替换。
3. 应用领域：智能交通、航拍等。
4. 结构组成：
   - 旋翼：对称分布在机体的前后、左右四个方向，大小、高度等相同
   - 支架：安装旋翼与电机，构成机体

- 电机：四个电机对称安装在飞行器的支架端
- 飞控：支架中间空间安放飞行控制计算机和外部设备

11.4.1.3 其他飞行器类型

1. 扑翼型
2. 气囊型

11.4.2 多旋翼空中机器人动态模型

11.4.2.1 核心模型

1. 螺旋桨拉力模型
2. 多旋翼的拉力与力矩模型
3. 多旋翼空中机器人运动模型

11.4.2.2 飞控方式（3 种）

1. 遥控飞行（Remote Control Flying）：航向、速度、高度等参数由地面操作人员通过通信数据链路遥控控制
2. 程控飞行（Autopilot Flying）：航向、速度、高度等参数根据预先制定的规划航线由机载计算机控制系统计算获得并实施控制
3. 自主飞行（Autonomous Flying）：机载计算机控制系统能够感知环境的变化自动进行实时路径规划，并进行相应的姿态、航向、速度、高度等参数控制
4. 基本原理：反馈控制
5. 要求：稳、准、快

11.4.2.3 导航系统

1. 定义：将运载体从起始点引导到目的地的技术或方法。
2. 导航的功能：
   (1) 测量定位：感知运载体的姿态、位置、速度、方向等信息
   (2) 航行决策：决定运载体运动的方向、速度
   (3) 路径规划：确定运载体从当前位置到达目的地的可行路径或最优路径
3. 比喻：导航是“眼睛”和“地图”
4. 常用方法：SLAM（同步定位与地图构建）

11.4.3 名词解释

11.4.3.1 速度相关概念

1. 真空速（TAS）：飞机相对于空气的运动速度，是考虑了空气密度影响的速度。

2. 指示空速（IAS）：折算到海平面高度的真空速，忽略了空气密度的变化，又称表速，是空速管测出的速度，也是表征飞机升力的速度。
3. 地速：飞机相对于地面运动速度的水平分量，是真空速与风速水平分量的矢量和。
4. 垂直速度：飞机相对于地面运动速度的垂直分量，即升降速度。
5. 马赫：
   - 马赫是表示速度的量词。一马赫即一倍音速。
   - 音波可以在固体、液体或是气体介质中传播，介质密度愈大，则音速愈快，所以马赫的大小不是固定的。
   - 马赫数小于 1 者为亚音速，马赫数大于 5 左右为超高音速。

11.4.3.2 激波

1. 飞机飞行对空气产生扰动，扰动（以扰动波的形式）以音速传播并积聚。
2. 超音速飞行时激波后的空气压力和温度急剧下降，导致水汽冷凝，形成雾化现象。

11.4.3.3 飞行包线

1. 以速度作为横坐标，以高度作为纵坐标，把各个高度下的速度上限和下限画出来，这样就构成了一条边界线，称为飞行包线。
2. 注意：飞机不是只能在这个线确定的范围内飞行。

11.4.3.4 气动布局

1. 常规布局
   (1) 有主机翼和水平尾翼，大的主机翼在前，小机翼也就是水平尾翼在后。
   (2) 有一个或者两个垂直尾翼。
2. 变后掠翼布局
   (1) 主翼的后掠角度可以改变。
   (2) 高速飞行可以加大后掠角，相当于飞鸟收起翅膀。
   (3) 低速飞行时减小后掠角，展开翅膀。
3. 无尾布局
   (1) 优点：没有水平尾翼，大大减少了空气阻力
   (2) 缺点
      - 低速性能不好，影响飞机的低速机动性能和起降能力
      - 只能依靠主翼控制飞行，稳定性不理想
4. 鸭式布局
   (1) 无尾布局加个鸭翼。
   (2) 有了这个鸭翼，无尾布局的缺点得到明显改善：
      - 高速飞行时更加稳定
      - 起降距离明显缩短

- 机动性能比常规布局更加出色

5. 飞翼布局

(1) 特征：只有飞机翅膀的布局，看上去只有机翼，没有机身，机身和机翼融为一体

(2) 优点：

- 空气动力效率最高，所有机身结构都是机翼，用于产生升力
- 最大程度降低了阻力
- 空气阻力最小所以雷达波反射自然也是最小，隐身性能最好

(3) 缺点：操控性能极差，完全依赖电子传感控制机翼和发动机的矢量推力

6. 前掠翼布局 - 主翼前掠而不是后掠，有巨大缺陷。

11.4.3.5 主动控制（ACT）

在飞行器设计的初始阶段，考虑控制系统对飞行器总体设计的影响，充分发挥飞行控制的潜力的设计技术。

11.4.4 飞行器运动描述

11.4.4.1 刚体运动模型

1. 飞行器可以抽象为一个刚体。
2. 描述任意时刻的空间运动需要六个自由度：三个质心运动、三个角运动
3. z 指向地心

11.4.4.2 坐标系定义

坐标系采用右手定则。地面坐标系常用于指示飞机的方位，用于近距离导航和航迹控制。

1. 地球中心坐标系（ECEF）

(1) ECEF 坐标系与地球固联，且随着地球转动。

(2) 原点 O 位置在地球质心。

(3) X 轴通过格林尼治线和赤道线的交点，正方向为原点指向交点方向。

(4) Z 轴通过原点指向北极。

(5) Y 轴与 X、Z 轴构成右手坐标系。

2. WGS-84 坐标系

(1) X 轴指向 BIH（国际时间服务机构）1984.0 定义的零子午面（Greenwich）和协议地球极（CTP）赤道的交点。

(2) Z 轴指向 CTP 方向。

(3) Y 轴与 X、Z 轴构成右手坐标系。

(4) 注意：尽管有 GPS，但仍需要气压计辅助定高。

3. NED 坐标系

(1) 在导航计算时使用的坐标系。

(2) 向量分别指向北、东、地，因此 NED 坐标系也经常称为“北东地坐标系”。

4. 机体坐标系

   (1) 与飞行器固联，坐标系符合右手法则。

   (2) 原点在飞行器重心处。

   (3) X 轴指向飞行器机头前进方向。

   (4) Y 轴由原点指向飞行器右侧。

   (5) Z 轴方向根据 X、Y 轴由右手法则确定。

11.4.4.3 控制系统

1. PID 整定

参数变化	K_p ↑	K_d ↑	K_i ↑
Rise Time（上升时间）	减少	-	减少
Overshoot（超调量）	增加	减少	增加
Setting Time（校正时间）	-	减少	增加
Steady-State Error（稳态误差）	减少	-	消除

2. Ziegler-Nichols 整定法

   (1) 设置 $K_i = K_d = 0$

   (2) 增大 K_p 直到输出发生震荡，此时 K_p 记为最大增益 K_u

   (3) 记录增益为 K_u 时的振荡周期 T_u

   (4) 根据下表来设置增益参数

控制器	K_p	K_d	K_i
P	$0.5K_u$	-	-
PD	$0.8K_u$	$K_p\frac{T_u}{8}$	-
PID	$0.6K_u$	$K_p\frac{T_u}{8}$	$2\frac{K_p}{T_u}$

11.4.5 导航系统详解

11.4.5.1 导航方式分类

1. 单一导航：卫星导航系统、多普勒导航、惯性导航系统（INS）、图形匹配导航系统、无线电跟踪系统、地磁导航、天文导航
2. 组合导航：INS/GPS 组合导航系统、惯导/多普勒组合导航系统、惯导/地磁组合导航系统、惯导/地形匹配组合导航系统、GPS/航迹推算组合导航系统

11.4.5.2 著名导航系统

1. GPS 全球卫星定位系统

   (1) 组成：三部分

- 空间部分：GPS 星座（24 个卫星）
- 地面控制部分：地面监控系统
- 用户设备部分：GPS 信号接收机

(2) 工作原理：每次用四个卫星定位

(3) 功能：导航、定位、授时

2. 惯性导航系统（INS）

(1) 分类：平台式惯导系统、捷联惯导系统

(2) 包含模块：计算机、加速度计、陀螺仪或其他运动传感器的平台

(3) 工作原理：加速度 + 角速度 $\rightarrow$ 位姿、速度变化 $\xrightarrow{\text{结合初始条件}}$ 定位导航

(4) 优点：不依赖外界任何信息实现完全自主的导航、隐蔽性好、不受外界干扰、不受地形影响、能够全天候工作

(5) 缺点：系统精度取决于单个传感器精度、实际空间位置的漂移是不可避免的、误差随时间累积

3. 多普勒导航系统

(1) 原理：利用多普勒效应实现

(2) 组成：磁罗盘或陀螺仪表、多普勒雷达、导航计算机

(3) 工作方式：不断向地面发射无线电波，通过回波和发射波的频率变化推算速度大小和方向

(4) 优点：自主性好、反应快、抗干扰性强、测速精度高、能用于各种气候条件

(5) 缺点：隐蔽性不好、系统工作受地形影响、测量有积累误差

4. 图形匹配导航系统（地形辅助导航）

(1) 原理：预先植入数字化地图，届时实时测量并比较原图、推算偏差。

(2) 要求：必须和其他导航方式进行组合，更多的是和图形/惯性组合。

(3) 分类：地形匹配导航、景像匹配导航

(4) 优点：没有累积误差、隐蔽性好、抗干扰性能较强

(5) 缺点：实时性受到制约、工作性能受地形影响、受天气影响、受飞行器的机动性影响

5. 地磁导航

(1) 原理：用地磁场定位

(2) 分类：地磁匹配（研究中更为广泛）、地磁滤波

(3) 优点：无源、无辐射，隐蔽性强，不受敌方干扰，全天时、全天候、全地域，能耗低，导航不存在误差积累，在跨海制导方面有一定的优势

(4) 缺点：地磁匹配需要存储大量的地磁数据，实时性与计算机处理数据的能力有关

6. 组合导航

(1) 一般以 INS 为主，误差不随时间累计的导航方式为辅。

(2) 黄金组合：GPS 的低动态、窄带宽、高精度和 INS 的高动态、宽频带、误差慢漂移特性的结合。

11.4.6 路径搜索算法（Path Finding）

11.4.6.1 基础搜索算法

1. 深度优先搜索（DFS） - 使用栈数据结构
2. 广度优先搜索（BFS） - 使用队列数据结构

11.4.6.2 A*搜索算法

1. 启发函数的可采用性
   (1) 要求：所有的节点 $h(n) \leq h^*(n)$，其中 $h^*(n)$ 是从节点 n 到终点的真实最小距离。
   (2) 性质：启发式函数可采用，那么 A*搜索是最优的。
2. 加权 A*算法
   (1) 公式：根据 $f = g + \varepsilon h, \varepsilon > 1$ 拓展搜索
   (2) 特点：更倾向于靠近终点的状态，比 A*快几个数量级。
   (3) 速度比较：加权 $\mathrm{A}^* \rightarrow$ Anytime $\mathrm{A}^* \rightarrow \mathrm{ARA}^* \rightarrow \mathrm{D}^*$
3. 不同参数组合 对于 $f = a \cdot g + b \cdot h$：
   (1) 贪婪：$a = 0, b = 1$
   (2) 可调整的贪婪：$a = 1, b > 1$
   (3) 最优：$a = b = 1$
   (4) **Dijkstra** 算法：$a = 1, b = 0$
4. 启发式函数选择
   (1) 最优启发式函数：
      - 欧氏距离
      - L_∞ 范数
      - 0
   (2) 非最优：曼哈顿距离（但有时需要打破对称性）
   (3) 打破对称性
      ① 核心思想：在相同 f 的节点中找到倾向性。
      ② 例如遇到斜线路径，但实际算法搜索了很大的区域，就是因为启发函数非最优，f 值相同使得被平等地拓展。
5. 跳跃点搜索（JPS）
   (1) 特点：JPS 减少了 Open List 中的节点数量，但增加了状态查询的数量
   (2) 限制：仅适用于统一网格地图

11.4.6.3 采样类路径搜索算法

1. 算法特点
   (1) 高效地探索环境的拓扑结构，即可行区域的连接情况
   (2) 不显示地构建构型空间及其边界

(3) 一般具有概率完备性

(4) 一般具有次优性或渐进最优性

(5) 可分为：单查询（single-query）、多查询（multi-query）

2. PRM（概率路径图）

(1) 类型：多查询算法

(2) 阶段：构建阶段、搜索阶段

(3) PRM 优化方法 - 懒惰的碰撞检查（Lazy Collision Checking）

- 建图不进行碰撞检查
- 生成路径后再判断
- 若碰撞则删除对应点和边
- 重新查找

3. RRT（快速随机树）

(1) 类型：单查询算法

(2) 原理：通过在工作空间采样节点来构建一棵从起点到终点的树

(3) 特点：随着采样增加，树从起点向终点生长

(4) RRT 效率提升方法：KD 树、双向 RRT、RRT*

4. RRT*算法

(1) 性质：具备概率完备性和渐进最优性

(2) 相比于 RRT 的改进：

- 考虑邻近节点的历史路径长度，而非只是局部路径长度
- 选择父节点时考虑多个邻近节点
- 考虑邻近节点的历史路径长度，而非只是局部路径长度
- 重连接操作，改进局部最优解

5. Informed RRT*

(1) 划定椭圆范围，路径长度为 $2a$

(2) 在找到路径后，后续的采样在椭圆内部进行并修正

6. Forward Spanning Tree

(1) **Step 1**：采样，碰撞检测，由近到远连接

(2) **Step 2**：对子节点依据角度进行 0-1 打分

- 1 表示延伸到自由空间
- 0 表示将会产生碰撞
- 计算父节点累计得分

(3) **Step 3**：修剪

- 修剪权重 = 其得分 ÷ 其兄弟节点的最大值
- 广度优先遍历，提取无碰撞框架

(4) **Step 4**：找到最短分支，生成无碰撞安全球形走廊，迭代优化

7. 算法比较

方法	优势	劣势	适用场景
BFS/DFS	简单直观	无启发，效率低	简单图搜索
Dijkstra	完备性、最优性	无方向性，效率低	权重图搜索
A*	启发式、高效	需要好的启发函数	栅格地图规划
JPS	打破对称、高效	仅适用于规则网格	大规模栅格地图
PRM	多查询、概率完备	构建图效率低	多次查询场景
RRT	单查询、快速探索	路径非最优	高维空间
RRT*	渐进最优	收敛慢	需要最优解
Informed RRT*	更快收敛	需初始解	已知初始路径
Kinodynamic-RRT*	考虑动力学	计算复杂	考虑动力学约束

祝考试顺利！