

轨迹优化

硬约束与软约束

硬约束是指优化过程中的硬性限制条件，包括等式与不等式，比如：

$$\begin{aligned} \min f(x) \\ s.t. g_i(x) = c_i \\ h_j(x) \geq d_j \end{aligned}$$

软约束是指在代价函数中增加带权惩罚项，让求解结果趋于最小化该项，比如：

$$\min f(x) + \lambda_1 \cdot g(x) + \lambda_2 \cdot h(x)$$

如果优化问题同时存在硬约束和软约束，应该可以把硬约束的部分施加一个比较大的惩罚加到代价函数中，以软约束的形式存在。

施加硬约束

如果想要对一段路径施加不等式约束，比如约束曲线的边界或者速度加速度的上下限，使用之前常规的多项式形式是无法办到的。一种可行的方法是将路径离散成很多点，然后对这些点施加约束，另一种更优的方法就是使用贝塞尔曲线作为路径的假设。

贝塞尔曲线

之前使用的常见的多项式成为monomial多项式，现在引入一个称为Bernstein多项式，也就是贝塞尔曲线，它的形式为：

$$\begin{aligned} B_j(t) &= c_j^0 b_n^0(t) + c_j^1 b_n^1(t) + \cdots + c_j^n b_n^n(t) = \sum_{i=0}^n c_j^i b_n^i(t) \\ b_n^i(t) &= \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i} \end{aligned}$$

其中每个 c_j 称为一个控制点，他是具有物理意义的。

值得注意的是，如果将贝塞尔曲线展开，他其实就是monomial多项式的一种特殊形式，他们之间具有映射关系：

$$p = M \cdot c$$

因此之前在monomial多项式中推导的结果也一样可以迁移到贝塞尔曲线中。

性质

1. 贝塞尔曲线只会经过第一个控制点和最后一个控制点，不会经过其他控制点。
2. 凸包：贝塞尔曲线会包含在由所有控制点组成的一个凸多边形称为凸包中
3. 贝塞尔曲线的导数形式跟原形式有非常密切的关系：

$$c'_i = n(c_{i+1} - c_i)$$

那么二阶导的形式就为：

$$c''_i = n(n-1)[(c_{i+2} - c_{i+1}) - (c_{i+1} - c_i)]$$

以此类推。

4. 贝塞尔曲线的时间 t 定义域为 $[0,1]$ 。这意味着实际应用中需要将时间归一化

应用

1. 边界约束

$$\begin{aligned} c_0^0 &= p_0, \\ n(c_0^1 - c_0^0) &= v_0, n(c_0^n - c_0^{n-1}) = c'_{n-1} = v_n \\ n(n-1)[(c_2 - c_1) - (c_1 - c_0)] &= a_0 \end{aligned}$$

2. 连续性约束

$$c_j^n = c_{j+1}^0$$

3. 安全性约束（曲线的限制区域）

$$a \leq c_j^i \leq b$$

4. 动力学约束

$$\begin{aligned} v_m^- &\leq n(c_j^i - c_j^{i-1}) \leq v_m^+ \\ a_m^- &\leq n(n-1)(c_j^i - 2c_j^{i-1} + c_j^{i-2}) \end{aligned}$$

这样整个问题仍是一个典型的QP问题。

软约束

- Differential flatness property
 $\{x, y, z, \dot{x}, \dot{y}, \dot{z}, \theta, \varphi, p, q, r\} \rightarrow \{x, y, z, \varphi\}$
- Piecewise polynomial trajectory

$$f_\mu(t) = \begin{cases} \sum_{j=0}^N p_{1j}(t - T_0)^j & T_0 \leq t \leq T_1 \\ \sum_{j=0}^N p_{2j}(t - T_1)^j & T_0 \leq t \leq T_1 \\ \vdots & \vdots \\ \sum_{j=0}^N p_{Mj}(t - T_{M-1})^j & T_0 \leq t \leq T_1 \end{cases}$$
- Objective function

$$J = J_s + J_c + J_d$$

$\lambda_1 J_1$
Smoothness cost

$\lambda_2 J_2$
Collision cost

$\lambda_3 J_3$
Dynamical cost

- Smoothness cost: minimum snap formulation

$$J_s = \sum_{\mu \in \{x, y, z\}} \int_0^T \left(\frac{d^k f_\mu(t)}{dt^k} \right)^2 dt$$

$$= \begin{bmatrix} d_F \\ d_P \end{bmatrix}^T C^T M^{-T} Q M^{-1} C \begin{bmatrix} d_F \\ d_P \end{bmatrix} = \begin{bmatrix} d_F \\ d_P \end{bmatrix}^T \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix} \begin{bmatrix} d_F \\ d_P \end{bmatrix}$$
- Collision cost: penalize on the distance to nearest obstacle

$$J_c = \int_{T_0}^{T_M} c(p(t)) ds$$

$$= \sum_{k=0}^{T/\delta t} c(p(T_k)) \|v(t)\| \delta t, T_k = T_0 + k \delta t$$
- Dynamical Cost: penalize on the velocity and acceleration where exceeds limits (similar to collision term).

与之前不同的是代价函数中不止有平滑项，还加入了碰撞项和动力学项，其中碰撞项的代价是根据离障碍物的距离构建一个距离场计算的，也就是函数 c 。

碰撞项和动力学项都是通过对 ds 进行积分计算的，原因在于如果对 dt 进行积分，可能会使结果偏向于让速度很大，时间很小来减小cost。这两项无法像平滑项一样可以化成一个很好的形式求解，只能通过离散很多点求和的形式进行数值近似。

- Smoothness cost: minimum snap formulation

$$J_s = \sum_{\mu \in \{x, y, z\}} \int_0^T \left(\frac{d^k f_\mu(t)}{dt^k} \right)^2 dt$$

$$= \begin{bmatrix} \mathbf{d}_p \\ \mathbf{d}_\mu \end{bmatrix}^T \mathbf{C}^T \mathbf{M}^{-1} \mathbf{Q} \mathbf{M}^{-1} \mathbf{C} \begin{bmatrix} \mathbf{d}_p \\ \mathbf{d}_\mu \end{bmatrix} = \begin{bmatrix} \mathbf{d}_p \\ \mathbf{d}_\mu \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_{FF} & \mathbf{R}_{F\mu} \\ \mathbf{R}_{\mu F} & \mathbf{R}_{\mu\mu} \end{bmatrix} \begin{bmatrix} \mathbf{d}_p \\ \mathbf{d}_\mu \end{bmatrix}$$

\mathbf{d}_μ

- Collision cost: penalize on the distance to nearest obstacle

$$J_c = \int_{T_0}^{T_M} c(p(t)) ds$$

Distance penalty at a point along the trajectory

$$= \sum_{k=0}^{T/\delta t} c(p(T_k)) \|v(t)\| \delta t, T_k = T_0 + k \delta t$$

- The Jacobian with respect to free derivatives $\mathbf{d}_{p\mu}$ is:

$$\frac{\alpha J_c}{\alpha \mathbf{d}_{p\mu}} = \sum_{k=0}^{T/\delta t} \left\{ \nabla_\mu c(p(T_k)) \|v\| \mathbf{F} + c(p(T_k)) \frac{\mathbf{v}_\mu}{\|v\|} \mathbf{G} \right\} \delta t, \mu \in \{x, y, z\}$$

- The Jacobian with respect to free derivatives $\mathbf{d}_{p\mu}$ is:

$$\frac{\alpha J_s}{\alpha \mathbf{d}_{p\mu}} = 2 \mathbf{d}_F^T \mathbf{R}_{FP} + 2 \mathbf{d}_P^T \mathbf{R}_{PP}$$

L_{dp} is the right block of matrix $\mathbf{M}^{-1} \mathbf{C}$ which corresponds to the free derivatives on the μ axis $\mathbf{d}_{p\mu}$.

$$\mathbf{F} = \mathbf{T} L_{dp}, \quad \mathbf{G} = \mathbf{T} \mathbf{V}_m L_{dp}$$

$\nabla_\mu c(\cdot)$ is the gradient in μ axis of the collision cost.

\mathbf{V}_m maps the coefficients of the position to the coefficients of the velocity. $\mathbf{T} = [T_k^0, T_k^1, \dots, T_k^n]$

$$\mathbf{H}_0 = \begin{bmatrix} \frac{\partial^2 f_\omega}{\partial \mathbf{d}_{p\mu}^2}, \frac{\partial^2 f_\omega}{\partial \mathbf{d}_{p\mu} \partial \mathbf{d}_{p\mu}}, \frac{\partial^2 f_\omega}{\partial \mathbf{d}_{p\mu}^2} \end{bmatrix},$$

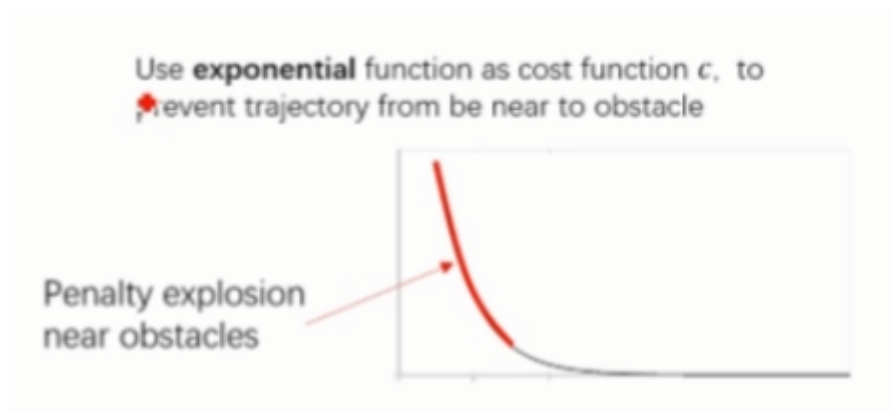
$$\frac{\partial^2 f_\omega}{\partial \mathbf{d}_{p\mu}^2} = \sum_{k=0}^{T/\delta t} \left\{ \mathbf{F}^T \nabla_\mu c(p(T_k)) \frac{\mathbf{v}_\mu}{\|v\|} \mathbf{G} + \mathbf{F}^T \nabla_\mu^2 c(p(T_k)) \|v\| \mathbf{F} \right.$$

$$\left. + \mathbf{G}^T \nabla_\mu c(p(T_k)) \frac{\mathbf{v}_\mu}{\|v\|} \mathbf{F} + \mathbf{G}^T c(p(T_k)) \frac{\mathbf{v}_\mu^2}{\|v\|^3} \mathbf{G} \right\} \delta t,$$

之后计算出各项梯度后，可以使用各种非线性优化的算法进行优化，代码上直接用ceres库就行。

距离场

距离场的构建有很多种办法，比如使用指数函数，当离障碍物很近的时候很大，远的时候很小。

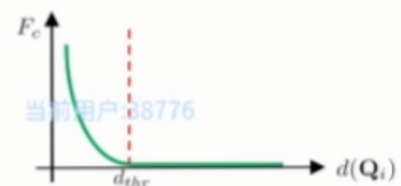


还可以使用二次函数。

- Safety - potential function in EDF

$$f_c = \sum_{i=p_b}^{N-p_b} F_c(d(\mathbf{Q}_i))$$

$$F_c(d(\mathbf{Q}_i)) = \begin{cases} (d(\mathbf{Q}_i) - d_{thr})^2 & d(\mathbf{Q}_i) \leq d_{thr} \\ 0 & d(\mathbf{Q}_i) > d_{thr} \end{cases}$$



对于速度和加速度也可以用同样的方式进行场的构建。