Numerical Optimization in Robotics
Final Project Hints

Task1 Collision distance computation

Efficiently computing collision distance is a key function in robot navigation. To generate a smooth path that avoids those convex polytopes/obstacles, we need compute collision vectors for necessary obstacles then choise the shortest one first.

This course/Dr. Wang shows three methods for this (See Lecture 3 Slides 75-79). Here we use the following method to model the collision vector computation.

Collision vector from a robot to a polytope obstacle: H-rep cases

The polytope is characterized by finite half-spaces intersection, namely,

$$P = \left\{x \in R^d \mid A_p x \le b_p\right\}$$

These half-spaces are subsets of . In practical, we consider two-dimension cases. After those polygons are characterized, we next model the computation of the collision vector.

$$\min_{x \in R^d} \left\| x - x_{robot} \right\|^2, s.t. A_p x \le b_p$$

This model can be solved using low-dimension QP which our course has covered. Infeasibility implies collision occurs or the robot falls into a obstacle. Such cases shall be avoided by imposing strong penalties. After obtaining the collision vector, we next generate a smooth path that avoids those polygons. Recalling the homework for chapter 2, we assume the smooth path is modeled by a cubic spline curve.

$$p_i(s) = a_i + b_i s + c_i s^2 + d_i s^3, s \in [0,1]$$

We aim to generate such smooth path by the following unconstrained minimization problem

$$\min_{x_1, x_2, \ldots x_{N-1}} Energy(x_1, x_2, \ldots x_{N-1}) + Potential(x_1, x_2, \ldots x_{N-1})$$

The energy function of the optimization problem is

$$Energy(x_1, x_2, \ldots, x_{N-1}) = \sum_{i=0}^{N} \int_0^1 \| p_i^{(2)}(s) \|^2 \, ds$$

Let $f_i = \int_0^1 \left\| p_i^2(s) \right\|^2 ds$, we then obtain the following results by some simple calculation

$$\sum_{i=0}^{N} f_i = \sum_{i=0}^{N} 12 d_i^2 + 12 c_i d_i + 4 c_i^2$$

Furthermore, the derivative of f_i with respect to implicit variable x (vector) can be derived by

$$\frac{df_i}{dx}=24\frac{dd_i}{dx}d_i+12\frac{dc_i}{dx}d_i+12\frac{dd_i}{dx}c_i+8\frac{dc_i}{dx}c_i$$

In order to obtain the expression of $\frac{df_i}{dx}$, we have to know two quantities $\frac{dc_i}{dx}$ and $\frac{dd_i}{dx}$.

A cubic curve sequentially crossing $x_0, x_1, \ldots, x_N$ is given by

$$\begin{aligned} a_i &= x_i \\ b_i &= D_i \\ c_i &= 3(x_{i+1}-x_i)-2D_i-D_{i+1} \\ d_i &= 2(x_i-x_{i+1})+D_i+D_{i+1} \end{aligned}$$

where

$$\begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_{n-2} \\ D_{n-1} \end{bmatrix} = \begin{bmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 3(x_2-x_0) \\ 3(x_3-x_1) \\ 3(x_4-x_2) \\ 3(x_5-x_3) \\ \vdots \\ 3(x_{n-1}-x_{n-3}) \\ 3(x_n-x_{n-2}) \end{bmatrix}, \text{ and } D_0 = D_N = 0$$

According to the chain rule of derivation

$$\frac{dd_i}{dx}=2\frac{d(x_i-x_{i+1})}{dx}+\frac{dD_i}{dx}+\frac{dD_{i+1}}{dx}$$
$$\frac{dc_i}{dx}=3\frac{d(x_{i+1}-x_i)}{dx}-2\frac{dD_i}{dx}-\frac{dD_{i+1}}{dx}$$

In which,

$$\left[\frac{d(x_i-x_{i+1})}{dx}\right]_{i=1,2,3,\cdots,N-1} = \begin{pmatrix} 1 & -1 & & & & & \\ & 1 & -1 & & & & \\ & & 1 & -1 & & & \\ & & & 1 & -1 & & \\ & & & & \ddots & \ddots & \\ & & & & & 1 & -1 \\ & & & & & & 1 \end{pmatrix}$$

Let’s simplify the symbols as follow

$$\underset{(N-1)*1}{B} = \begin{bmatrix} 3(x_2-x_0) \\ 3(x_3-x_1) \\ 3(x_4-x_2) \\ 3(x_5-x_3) \\ \vdots \\ 3(x_{n-1}-x_{n-3}) \\ 3(x_n-x_{n-2}) \end{bmatrix} \quad \underset{(N-1)*(N-1)}{A} = \begin{pmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & 1 & 4 & 1 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{pmatrix} \quad \underset{(N-1)*1}{D} = \begin{bmatrix} D_1 \\ D_2 \\ D_3 \\ D_4 \\ \vdots \\ D_{n-2} \\ D_{n-1} \end{bmatrix}$$

Then we can get $D = A^{-1}B$, since A is a constant matrix

$$\frac{dB}{dx} = \begin{pmatrix} 0 & 3 & & & & & \\ -3 & 0 & 3 & & & & \\ & -3 & 0 & 3 & & & \\ & & -3 & 0 & 3 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -3 & 0 & 3 \\ & & & & & -3 & 0 \end{pmatrix}$$

So the derivative of D_i with respect to x(vector) can be represented as

$$\frac{dD}{dx} = A^{-1}\frac{dB}{dx} = A^{-1}\begin{pmatrix} 0 & 3 & & & & & \\ -3 & 0 & 3 & & & & \\ & -3 & 0 & 3 & & & \\ & & -3 & 0 & 3 & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & -3 & 0 & 3 \\ & & & & & -3 & 0 \end{pmatrix}$$

Then we can get the grad of the energy function.

Next, we take the potential function into consideration.

$$\text{Potential}(x_1, x_2 \ldots, x_{N-1}) = 1000\sum_{i=1}^{N-1}\sum_{j=1}^{M}\max(r_j - \|x_i - o_j\|,\ 0)$$

This function is non-zero only when the path collides with the obstacle, so if the front end uses a collision-free method such as A* this function will not work.

When the path is not collision-free,(we can use Euclidean norm)

$$P = 1000\sum_{i=1}^{N-1}\sum_{j=1}^{M} r_j - \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2}$$

where x and y are the two dimensions of and the above equation is derived for the variables as follows

$$\frac{dP}{dx_i} = -1000\sum_{j=1}^{M}\frac{x_i - a_j}{\sqrt{(x_i - a_j)^2 + (y_i - b_j)^2}}$$

$$\frac{dP}{dy_i} = -1000\sum_{j=1}^{M}\frac{y_i - a_j}{\sqrt{(x_i - a_j)^2 + (y_i - b_j)^2}}$$

Then we can get the grad of the potential function.

Task2 Time-Optimal Path Parameterization (TOPP)

Assume the path $q(s)$ is parameterized by the arc-length variable s. Two auxiliary variables $a(s)$ and $b(s)$ are introduced. The variable $a(s)$ represents the acceleration on arc-length , while $b(s)$ represents the square of the speed. That is,

$$a(s)=\frac{d^2s}{dt^2},b(s)=\left(\frac{ds}{dt}\right)^2$$

We have

$$b'(s)=\frac{d}{ds}b(s)=\frac{d}{dt}b(s)\cdot\frac{dt}{ds}=2\frac{ds}{dt}\cdot\frac{d^2s}{dt^2}\cdot\frac{dt}{ds}=2a(s)$$

The purpose of this task (TOPP) is to minimize the total time

$$T=\int_0^T 1dt=\int_{s(0)}^{s(T)}\frac{1}{ds/dt}ds=\int_0^L\frac{1}{ds/dt}ds=\int_0^L\frac{1}{\sqrt{b(s)}}ds$$

True velocity:

$$\frac{dq}{dt}=q'(s)\frac{ds}{dt}=q'(s)\sqrt{b(s)}$$

True acceleration:

$$\frac{d^2q}{dt^2}=q''(s)\left(\frac{ds}{dt}\right)^2+q'(s)\frac{d^2s}{dt^2}=q''(s)b(s)+q'(s)a(s)$$

More importantly, we only consider the forward moving cases, thus

$$b(s)\geq 0$$

We can thus formulate this simple TOPP problem as

$$\begin{aligned}
\min_{a(s),b(s)}\ & \int_0^L\frac{1}{\sqrt{b(s)}}\mathrm{d}s \\
\text{s.t.}\ & b(s)\geq 0, && \forall s\in[0,L],\\
& b'(s)=2a(s), && \forall s\in[0,L],\\
& \left\|q'(s)\sqrt{b(s)}\right\|_\infty\leq v_{max}, && \forall s\in[0,L],\\
& \left\|q''(s)b(s)+q'(s)a(s)\right\|_\infty\leq a_{max}, && \forall s\in[0,L],\\
& b(0)=b_0,\ b(L)=b_L.
\end{aligned}$$

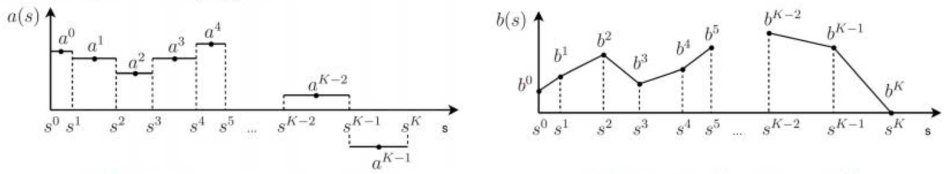
Blue parts are all known constants.

To solve this continuous model, we need discretize it and transfer the discretization model to a conic programming problem.

The discretizing steps:

Time Optimal Path Parameterization

Discretize the convex program:



$a(s)$ is piecewise constant for $s \in [0, L]$ $\qquad$ $b(s)$ is piecewise linear for $s \in [0, L]$

Objective:	$\int_0^L \frac{1}{\sqrt{b(s)}}\mathrm{d}s \Leftrightarrow \sum_{k=0}^{K-1} \frac{2(s^{k+1}-s^k)}{\sqrt{b^{k+1}}+\sqrt{b^k}}$
Positiveness:	$b(s) \geq 0,\ \forall s \in [0, L] \Leftrightarrow b^k \geq 0,\ 0 \leq k \leq K$
Physical Law:	$b'(s) = 2a(s),\ \forall s \in [0, L] \Leftrightarrow \frac{b^{k+1}-b^k}{s^{k+1}-s^k} = 2a^k,\ 0 \leq k \leq K$
Vel. Bound:	$\left\lVert q'(s)\sqrt{b(s)}\right\rVert_\infty \leq v_{max},\ \forall s \in [0, L] \Leftrightarrow \left\lVert q'(s^k)\sqrt{b^k}\right\rVert_\infty \leq v_{max},\ 0 \leq k \leq K$
Acc. Bound:	$\left\lVert q''(s)b(s) + q'(s)a(s)\right\rVert_\infty \leq a_{max},\ \forall s \in [0, L] \Leftrightarrow \left\lVert q''(s^k)b^k + q'(s^k)a^k\right\rVert_\infty \leq a_{max},\ 0 \leq k \leq K$

The original TOPP is discretized into the program below:

$$
\begin{array}{llll}
\min\limits_{a^k, b^k} & \boxed{\sum_{k=0}^{K-1} \frac{2(s^{k+1}-s^k)}{\sqrt{b^{k+1}}+\sqrt{b^k}}} & & \\
\text{s.t.} & b^k \geq 0, & 0 \leq k \leq K, & \text{Linear Inequalities} \\
& b^{k+1} - b^k = 2(s^{k+1}-s^k)a^k, & 0 \leq k \leq K & \text{Linear Equalities} \\
& \left\lVert q'(s^k)\sqrt{b^k}\right\rVert_\infty \leq v_{max}, & 0 \leq k \leq K & \text{Linear Inequalities} \\
& \left\lVert q''(s^k)b^k + q'(s^k)a^k\right\rVert_\infty \leq a_{max}, & 0 \leq k \leq K & \text{Linear Inequalities} \\
& b(0) = b_0,\ b(L) = b_L. & & \text{Linear Equalities}
\end{array}
$$

We note that those constraints above are equality or inequality in essence.

The optimization objective can be reformulated as follow

$$
\min_{a^k, b^k} \sum_{k=0}^{K-1} \frac{2(s^{k+1}-s^k)}{\sqrt{b^{k+1}}+\sqrt{b^k}} \iff
\begin{array}{lll}
\min\limits_{a^k, b^k, c^k, d^k} & \sum_{k=0}^{K-1} 2(s^{k+1}-s^k)d^k & \\
\text{s.t.} & \left\lVert \begin{matrix} 2 \\ c^{k+1}+c^k-d^k \end{matrix} \right\rVert_2 \leq c^{k+1}+c^k+d^k, & 0 \leq k \leq K-1, \\
& \left\lVert \begin{matrix} 2c^k \\ b^k-1 \end{matrix} \right\rVert_2 \leq b^k+1, & 0 \leq k \leq K.
\end{array}
$$

Therefore, the TOPP model reads

$$
\begin{aligned}
&\min_{a,b,c,d} \sum_{k=0}^{K-1} 2(s^{k+1}-s^k)d^k \\
&s.t. \left\| \begin{matrix} 2 \\ c^{k+1}+c^k-d^k \end{matrix} \right\|_2 \le c^{k+1}+c^k+d^k, 0 \le k \le K-1 \\
&\left\| \begin{matrix} 2c^k \\ b^k-1 \end{matrix} \right\|_2 \le b^k+1, 0 \le k \le K \\
&b^k \ge 0, 0 \le k \le K \\
&b^{k+1}-b^k = 2(s^{k+1}-s^k)a^k, 0 \le k \le K \\
&-v_{\max} \le q'(s^k)\sqrt{b^k} \le v_{\max}, 0 \le k \le K \\
&-a_{\max} \le q''(s^k)b^k + q'(s^k)a^k \le a_{\max}, 0 \le k \le K \\
&b(0)=b_0, b(L)=b_L.
\end{aligned}
$$

One can try Mosek tool to solve it or use Conic-AML method. The details are left to the reader.