深蓝学院
shenlanxueyuan.com

Local Planning for Mobile Robots

Lecture 8

主讲人 Qianhao Wang

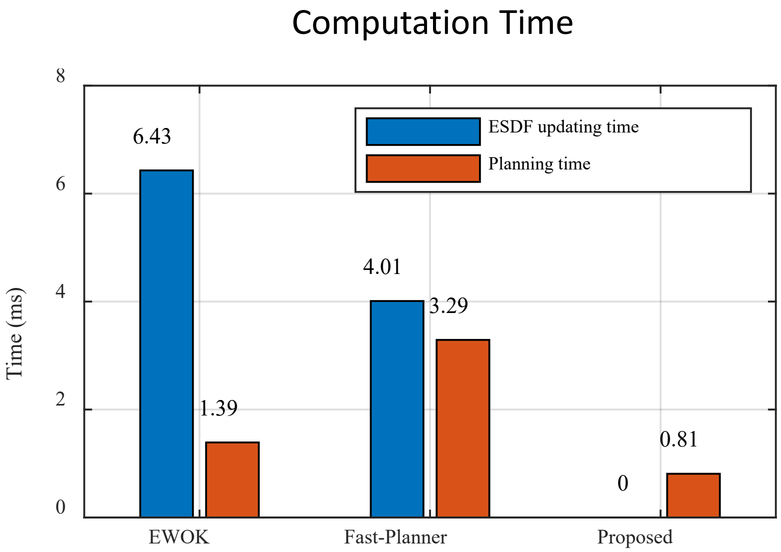
Ph.D. Candidate in Robotics
Zhejiang University

Ego-Planner[1]

[1] Zhou, Xin, et al. "Ego-planner: An esdf-free gradient-based local planner for quadrotors." *IEEE Robotics and Automation Letters* 6.2 (2020): 478-485.

Ego-Planner

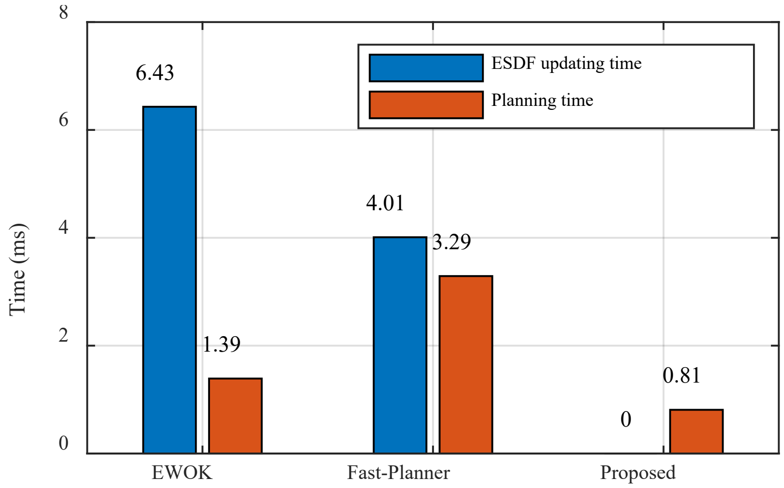
- An **ESDF-free** gradient-based local planner for autonomous flight is proposed.
- It significantly reduces the computation time while achieving impressive flight performance.



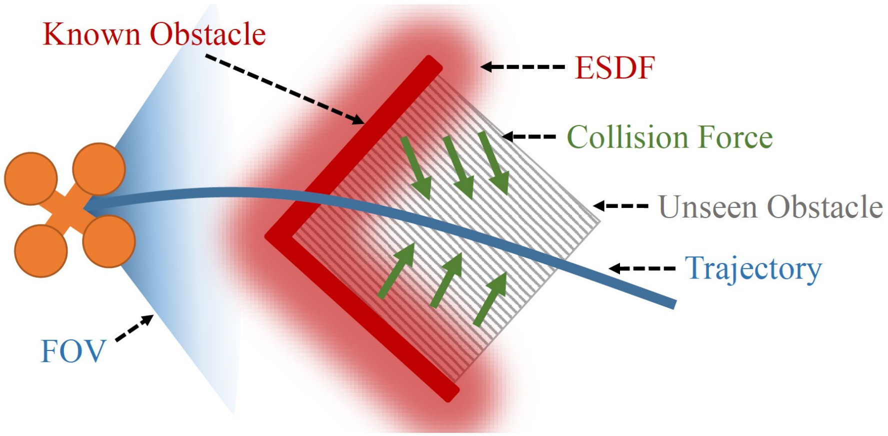
Ego-Planner

- Why not **ESDF**?

Computation Time

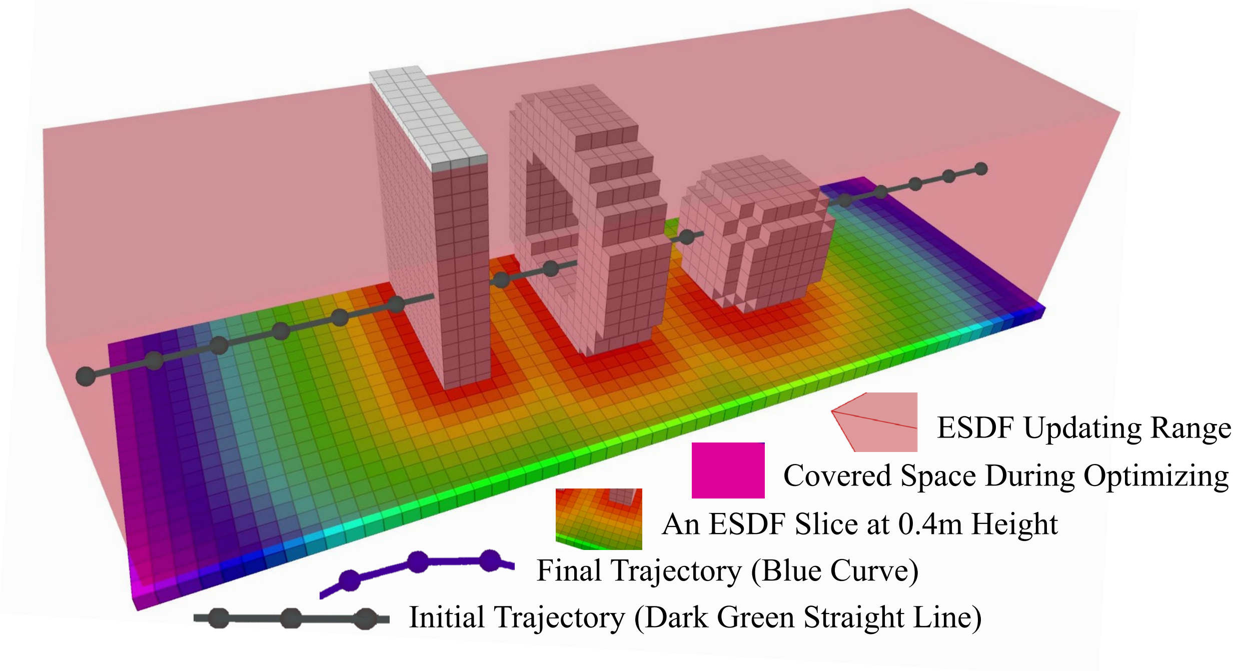


The trajectory gets stuck into a local minimum, which is very common since the camera has no vision of the back of the obstacle.



Ego-Planner

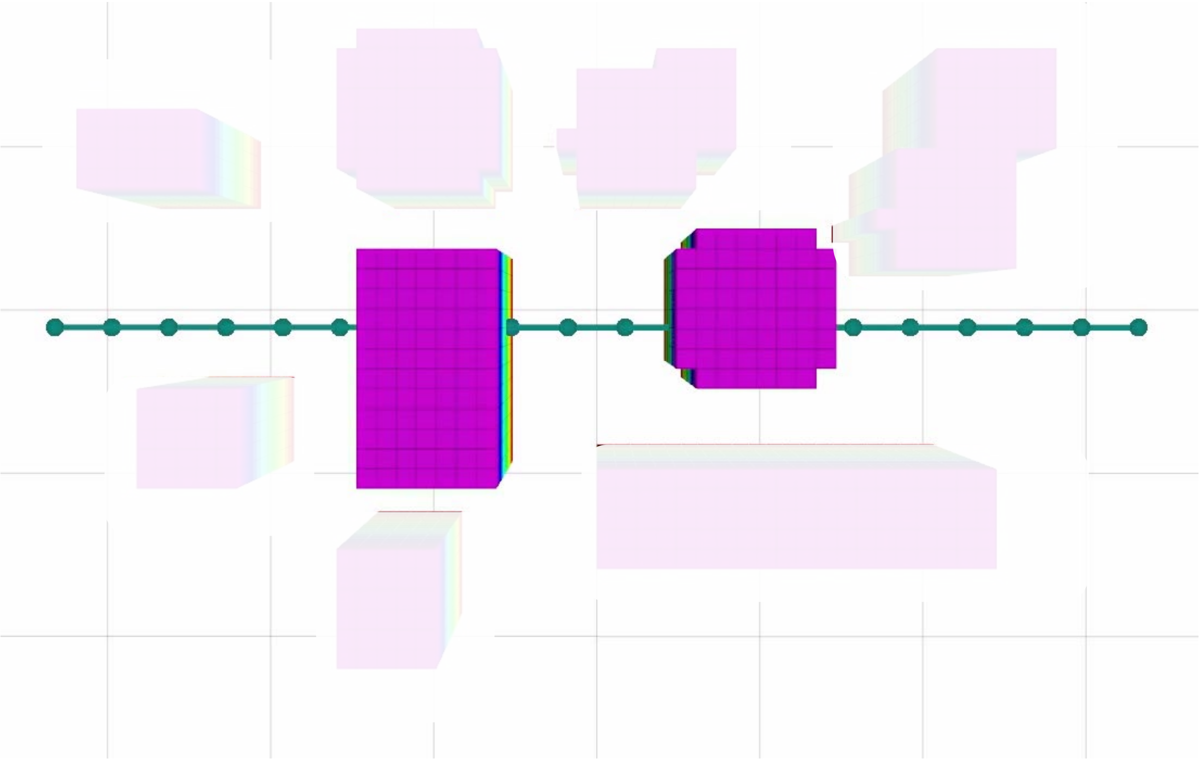
- ESDF contains significant redundancy as illustrated below.
- ESDF computation takes up about 70% of total computation time stated in [1].



[1]. Usenko, Vladyslav, et al. "Real-time trajectory replanning for MAVs using uniform B-splines and a 3D circular buffer." *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017.

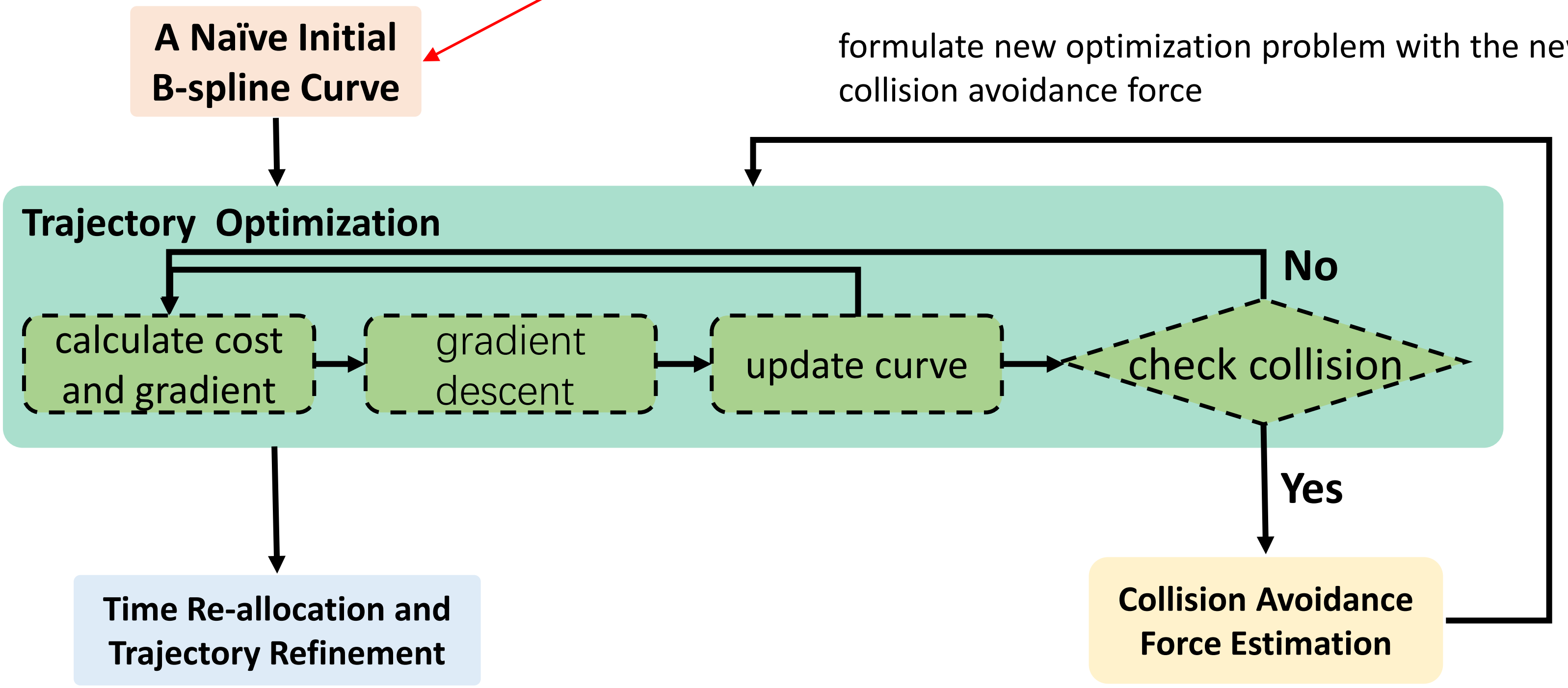
Ego-Planner

1. Trajectory moves away from obstacles without ESDF.
2. Computation time is reduced by only operating the necessary obstacles.



Ego-Planner

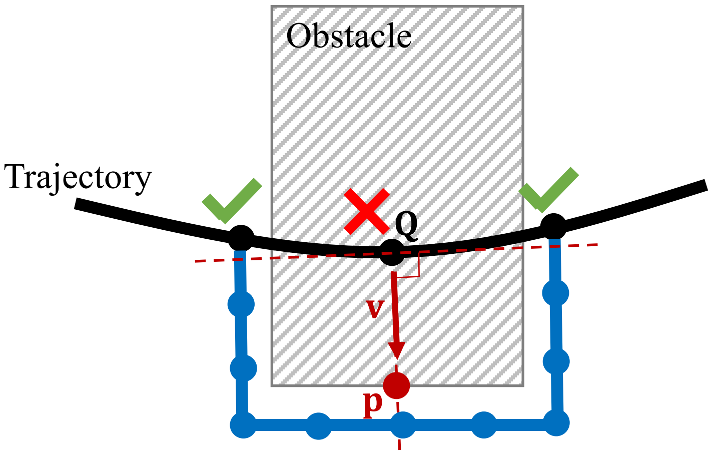
- A naive B-spline curve satisfying terminal constraints is given, regardless of collision.



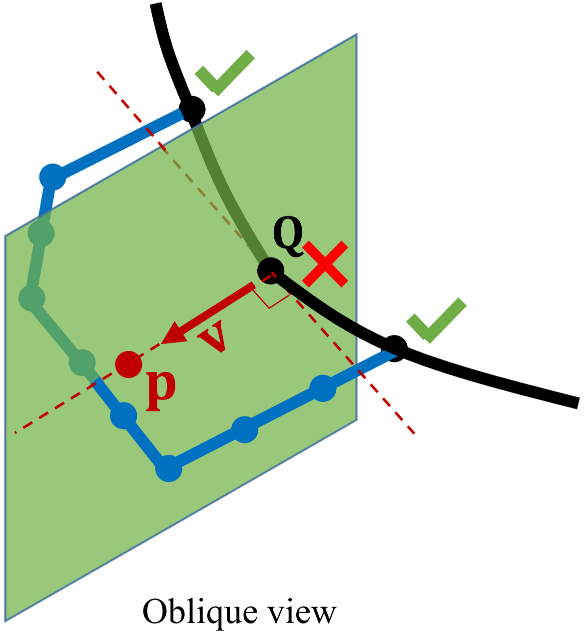
Ego-Planner

Collision Avoidance Force Estimation

- Extracting collision avoidance information denoted by $\{\mathbf{p}, \mathbf{v}\}$ pair.
- Obstacle distance is defined as $d = (\mathbf{Q} - \mathbf{p})^T \mathbf{v}$.



Top view

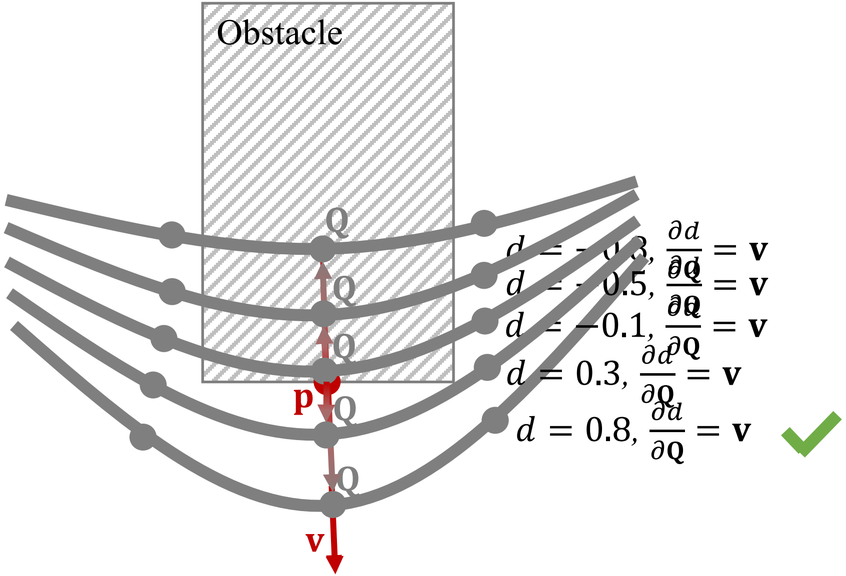


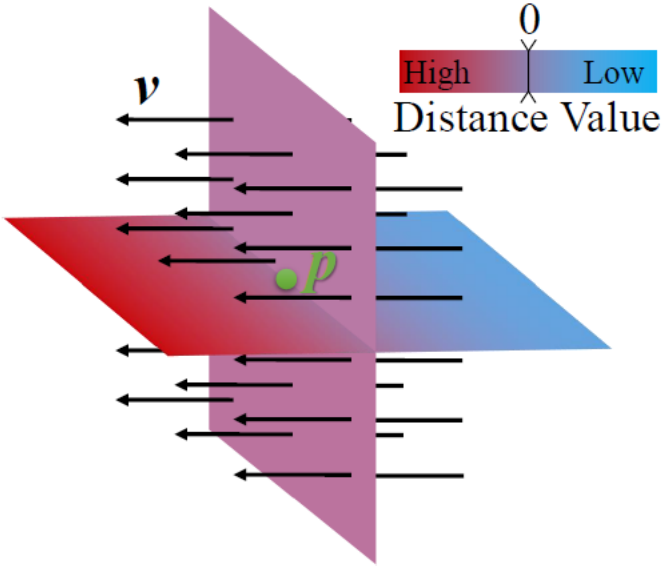
Oblique view
(The obstacle is concealed)

Ego-Planner

◆ Collision Avoidance Force Estimation

- The trajectory then utilizes the distance and gradient information to escape collision.
- For example, require the obstacle distance $d = (\mathbf{Q} - \mathbf{p})^T \mathbf{v}$ larger than 0.6 .





Back-End

slightly different from the paper of Fast-Planner

◆ Gradient-based Trajectory Optimization

- the trajectory is parameterized by a uniform B-spline curve , which is uniquely determined by its degree p_b, N_c control points $\boxed{\{\mathbf{Q}_1, \mathbf{Q}_2, \dots, \mathbf{Q}_N\}}$, and a knot vector $\boxed{[t_1, t_2, \dots, t_M]}$.
- each knot is separated by the same time interval $\Delta t_m = t_{m+1} - t_m$.
- the k^{th} derivative of a B-spline is still a B-spline with order $p_{b,k} = p_b - k$.
- Since Δt is identical alone , the control points of the velocity $\mathbf{V}_i$, acceleration $\mathbf{A}_i$, and jerk $\mathbf{J}_i$ curves are obtained by

$$\mathbf{V}_i = \frac{\mathbf{Q}_{\mathrm{i+1}} - \mathbf{Q}_{\mathrm{i}}}{\Delta t}, \qquad \mathbf{A}_i = \frac{\mathbf{V}_{\mathrm{i+1}} - \mathbf{V}_{\mathrm{i}}}{\Delta t}, \qquad \mathbf{J}_i = \frac{\mathbf{A}_{\mathrm{i+1}} - \mathbf{A}_{\mathrm{i}}}{\Delta t}$$

- The optimization problem is then formulated as follows:

$$\min_{\mathbf{Q}} \mathcal{J} = \lambda_s \boxed{\mathcal{J}_s} + \lambda_c \boxed{\mathcal{J}_c} + \lambda_d \boxed{\mathcal{J}_d}$$

smoothness collision dynamic feasibility

Back-End

◆ Smoothness penalty $\mathcal{J}_s$

- In [1], the smoothness penalty is formulized as the time integral over square derivatives of the trajectory (acceleration, jerk, etc.).
- In Fast-Planner , only geometric information of the trajectory is taken regardless of time allocation.

$$\Big\downarrow \text{ combine}$$

- Penalize squared acceleration and jerk without time integration.
- Benefiting from the convex hull property, minimizing the control points of second and third order derivatives of the B-spline trajectory is sufficient to reduce these derivatives along the whole curve.

$$\mathcal{J}_s = \sum_{i=1}^{N_c-2} \|\mathbf{A}_i\|_2^2 + \sum_{i=1}^{N_c-3} \|\mathbf{J}_\mathrm{i}\|_2^2$$

[1] V. Usenko, L. von Stumberg, A. Pangercic, and D. Cremers, "Realtime trajectory replanning for mavs using uniform b-splines and a 3d circular buffer," in Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.(IROS). IEEE, 2017, pp. 215–222.

Back-End

◆ Feasibility penalty $\mathcal{J}_d$

- Feasibility is ensured by restricting the higher order derivatives of the trajectory on every single dimension.
- Thanks to the convex hull property, constraining derivatives of the control points is sufficient for constraining the whole B-spline.

$$\mathcal{J}_d = \sum_{i=1}^{N_c-1} \omega_v\, F(\mathbf{V}_i) + \sum_{i=1}^{N_c-2} \omega_a\, F(\mathbf{A}_i) + \sum_{i=1}^{N_c-3} \omega_j\, F(\mathbf{J}_i)$$

- $F(\cdot)$ is a twice continuously differentiable metric function of higher order derivatives of control points.

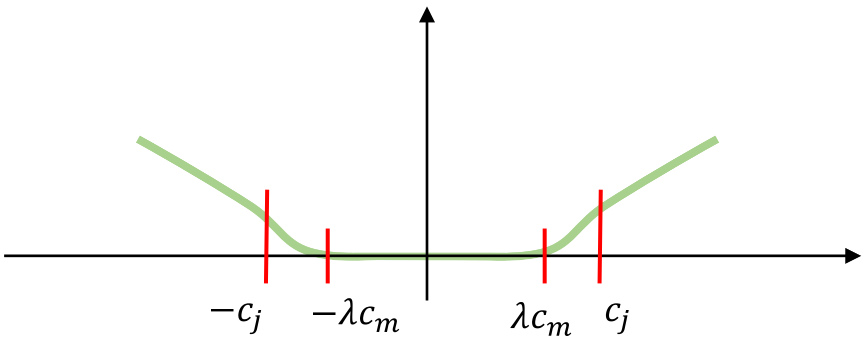
$$F(\mathbf{C}) = \sum_{r=x,y,z} f(c_r)$$

$$f(c_r) = \begin{cases} a_1 c_r^2 + b_1 c_r + c_1 & (c_r \le -c_j) \\ (-\lambda c_m - c_r)^3 & (-c_j < c_r \le -\lambda c_m) \\ 0 & (-\lambda c_m < c_r \le \lambda c_m) \\ (-\lambda c_m + c_r)^3 & (\lambda c_m < c_r \le c_j) \\ a_2 c_r^2 + b_2 c_r + c_2 & (c_j \le c_r) \end{cases}$$

Back-End

◆ **Feasibility penalty $\mathcal{J}_d$**

$$f(c_r) = \begin{cases} a_1 c_r^2 + b_1 c_r + c_1 & (c_r \le -c_j) \\ (-\lambda c_m - c_r)^3 & (-c_j < c_r \le -\lambda c_m) \\ 0 & (-\lambda c_m < c_r \le \lambda c_m) \\ (-\lambda c_m + c_r)^3 & (\lambda c_m < c_r \le c_j) \\ a_2 c_r^2 + b_2 c_r + c_2 & (c_j \le c_r) \end{cases}$$



- $a_1, b_1, c_1, a_2, b_2, c_2$ are chosen to meet the second-order continuity.
- c_m is the derivative limit, such as v_{max}.
- c_j is the splitting points of the quadratic interval and the cubic interval.
- $\lambda < 1 - \epsilon$ is an elastic coefficient with $\epsilon \ll 1$ to make the final results meet the constraints, since the cost function is a tradeoff of all weighted terms.

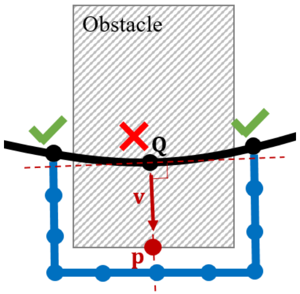
Back-End

◆ Collision penalty $\mathcal{J}_c$

- Collision penalty pushes control points away from obstacles. This is achieved by adopting a safety clearance s_f and punishing control points until $d_{ij} > s_f$.
- The cost on each $\mathbf{Q}_i$ is evaluated independently and accumulated from all corresponding $\{\mathbf{p},\mathbf{v}\}_j$ pairs. The cost value produced by $\{\mathbf{p},\mathbf{v}\}_j$ pairs on $\mathbf{Q}_i$ is:

$$d_{ij} = (\mathbf{Q}_i - \mathbf{p}_{ij}) \cdot \mathbf{v}_{ij}$$

$$c_{ij} = s_f - d_{ij}$$



$$j_c(i,j) = \begin{cases} 0 & (c_{ij} \le 0) \\ c_{ij}^3 & (0 < c_{ij} \le s_f) \\ 3s_f c_{ij}^2 - 3s_f^2 c_{ij} + s_f^3 & (s_f < c_{ij}) \end{cases}$$

a twice continuously differentiable penalty function

- N_p is the number of $\{\mathbf{p},\mathbf{v}\}_j$ pairs belonging to $\mathbf{Q}_i$. The cost value added to $\mathbf{Q}_i$:

$$j_c(\mathbf{Q}_i) = \sum_{j=1}^{N_p} j_c(i,j)$$

- Combining costs on all $\mathbf{Q}_i$ yields the total collision cost is

$$\mathcal{J}_c = \sum_{i=1}^{N_c} j_c(\mathbf{Q}_i) = \sum_{i=1}^{N_c}\sum_{j=1}^{N_p} j_c(i,j)$$

Back-End

◆ Collision penalty $\mathcal{T}_c$

- Unlike traditional ESDF-based methods, which compute gradient by trilinear interpolation, we obtain gradient by directly computing the derivative of $\mathcal{T}_c$ with respect to $\mathbf{Q}_i$.

discontinuous

$$\frac{\partial \mathcal{T}_c}{\partial \mathbf{Q}_i} = \sum_{i=1}^{N_c}\sum_{j=1}^{N_p} \frac{\partial j_c(i,j)}{\partial \mathbf{Q}_i}$$

$$\frac{\partial j_c(i,j)}{\partial \mathbf{Q}_i} = \frac{\partial j_c(i,j)}{\partial c_{ij}}\frac{\partial c_{ij}}{\partial \mathbf{Q}_i} = \frac{\partial j_c(i,j)}{\partial c_{ij}}\frac{\partial c_{ij}}{\partial d_{ij}}\frac{\partial d_{ij}}{\partial \mathbf{Q}_i} = -\frac{\partial j_c(i,j)}{\partial c_{ij}}\mathbf{v}_{ij}$$

$$j_c(i,j) = \begin{cases} 0 & (c_{ij} \le 0) \\ c_{ij}^3 & (0 < c_{ij} \le s_f) \\ 3s_f c_{ij}^2 - 3s_f^2 c_{ij} + s_f^3 & (s_f < c_{ij}) \end{cases}$$

$$c_{ij} = s_f - d_{ij}$$

$$d_{ij} = (\mathbf{Q}_i - \mathbf{p}_{ij}) \cdot \mathbf{v}_{ij}$$

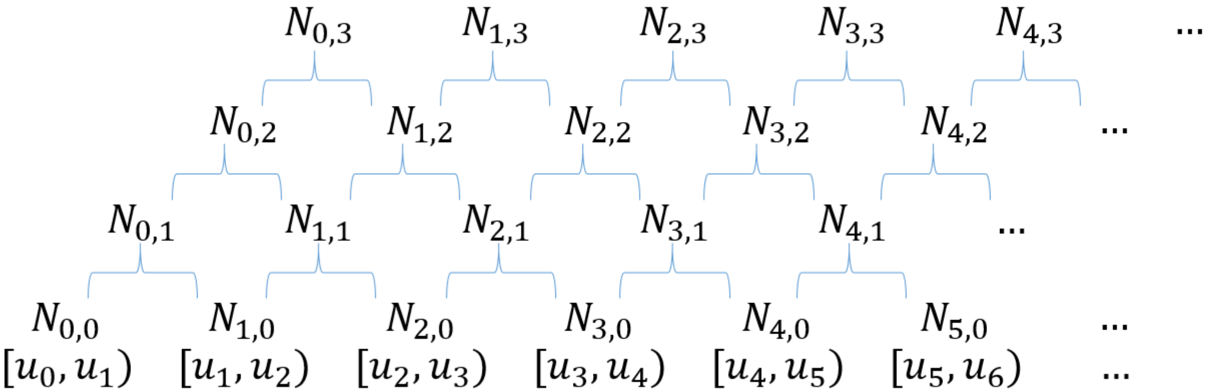
continuous

$$\boxed{\frac{\partial \mathcal{T}_c}{\partial \mathbf{Q}_i} = \sum_{i=1}^{N_c}\sum_{j=1}^{N_p} \mathbf{v}_{ij} \begin{cases} 0 & (c_{ij} \le 0) \\ -3c_{ij}^2 & (0 < c_{ij} \le s_f) \\ -6s_f c_{ij} + 3s_f^2 & (s_f < c_{ij}) \end{cases}}$$

Back-End

◆ Time Re-allocation and Trajectory Refinement

- **Fast-Planner** parameterizes the trajectory as a **nonuniform B-spline** and iteratively lengthen a subset of knot spans when some segments exceed derivative limits.
- One knot span Δt_m influences multiple control points and vice versa, leading to high-order discontinuity to the previous trajectory when adjusting knot spans near the start state.



- a p degree B-Spline $\boldsymbol{C}(u)$

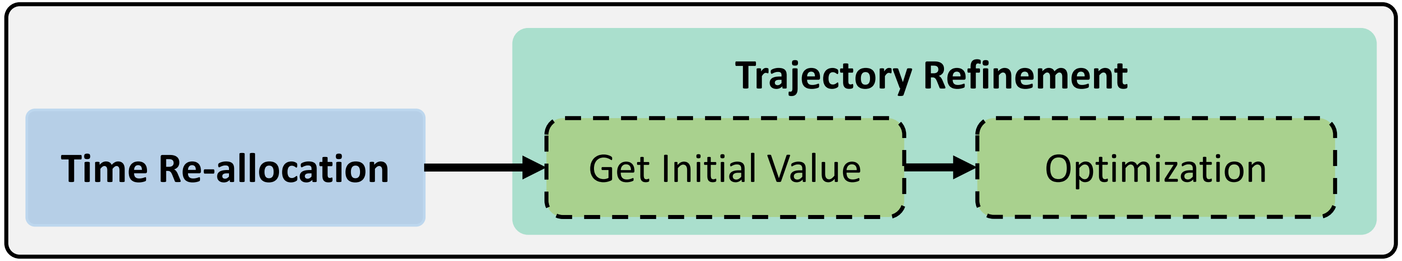
$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{Q}_i$$

Back-End

obtained from Gradient-based Trajectory Optimization

Time Re-allocation and Trajectory Refinement

- In Ego-Planner, a **uniform B-spline** trajectory $\boldsymbol{\Phi}_f$ is re-generated with reasonable time re-allocation according to the safe trajectory $\boldsymbol{\Phi}_s$.
- Then, an anisotropic curve fitting method is proposed to make $\boldsymbol{\Phi}_f$ freely optimize its control points to meet higher order derivative constraints while maintaining a nearly identical shape to $\boldsymbol{\Phi}_s$.



Back-End

detailed in ppt of Fast-Planner

◆ Time Re-allocation

- In as Fast-Planner does, we compute the limits exceeding ratio.

$$r_e = max\left\{\left|\mathbf{V}_{i,r}/v_m\right|, \sqrt{\left|\mathbf{A}_{j,r}/a_m\right|}, \sqrt[3]{\left|\mathbf{J}_{k,r}/j_m\right|}, 1\right\}$$

$$i \in \{1, \dots, N_c - 1\}, j \in \{1, \dots, N_c - 2\}, k \in \{1, \dots, N_c - 3\}, r \in \{x, y, z\}$$

- A notion with subscript m represents the limitation of a derivative (v_m is v_{max}).
- r_e indicates how much we should lengthen the time allocation for $\mathbf{\Phi}_f$ relative to $\mathbf{\Phi}_s$.

- Then we obtain the new time span of $\mathbf{\Phi}_f$:

$$\Delta t' = r_e \Delta t$$

$$[t_1, t_2, \dots, t_M] \longrightarrow [t_1, t_2', \dots, t_M']$$

Back-End

◆ Trajectory Refinement: Initial Value

- the new time span of $\mathbf{\Phi}_f$, $\Delta t'$, is initially generated under boundary constraints while maintaining the identical shape and control points number to $\mathbf{\Phi}_s$, by solving a closed-form min-least square problem.

$$[t_1, t_2, \dots, t_M] \rightarrow [t_1, t_2', \dots, t_M']$$

$$\mathbf{\Phi}_s(t) = \mathbf{s}(t)^{\mathrm{T}}\mathbf{M}_{p_b+1}\mathbf{q}_m$$

$$\mathbf{s}(t) = [1 \quad s(t) \quad s^2(t) \quad \dots \quad s^{p_b}(t)]^{\mathrm{T}}$$

$$\mathbf{q}_m = \left[\mathbf{Q}_{\mathrm{m}-p_b} \quad \mathbf{Q}_{\mathrm{m}-p_b+1} \quad \dots \quad \mathbf{Q}_{\mathrm{m}}\right]^{\mathrm{T}}$$

$$s(t) = (t - t_m)/\Delta t$$

$$\mathbf{\Phi}_f(t) = \mathbf{s}'(t)^{\mathrm{T}}\mathbf{M}_{p_b+1}\mathbf{q}_m{}'$$

$$\mathbf{s}'(t) = \left[1 \quad s'(t) \quad s'^2(t) \quad \dots \quad s'^{p_b}(t)\right]^{\mathrm{T}}$$

$$\mathbf{q}_m{}' = \left[\mathbf{Q}_{\mathrm{m}-p_b}{}' \quad \mathbf{Q}_{\mathrm{m}-p_b+1}{}' \quad \dots \quad \mathbf{Q}_{\mathrm{m}}{}'\right]^{\mathrm{T}}$$

$$s'(t) = (t - t_m{}')/\Delta t'$$

assumption: $t_1 = 0, \quad t_m' = r_e t_m$

$$\mathbf{\Phi}_f(r_e t) = \mathbf{\Phi}_s(t), t \in trajectory\ duration\ of\ \mathbf{\Phi}_s$$

Back-End

◆ Trajectory Refinement: Initial Value

- For knot span $[t_1, t_2, \ldots, t_M]$, A B-spline trajectory is parameterized by time t, where $t \in [t_{p_b+1}, t_{M-p_b}]$.

  And $M = N + p_b + 1$.

$$\mathbf{\Phi}_f(t) = \mathbf{s}'(t)^{\mathrm{T}}\mathbf{M}_{p_b+1}\mathbf{q}_m{}'$$

$$\mathbf{s}'(t) = \begin{bmatrix}1 & s'(t) & s'^2(t) & \ldots & s'^{p_b}(t)\end{bmatrix}^{\mathrm{T}}$$

$$\mathbf{q}_m{}' = \begin{bmatrix}\mathbf{Q}_{\mathrm{m}-p_b}{}' & \mathbf{Q}_{\mathrm{m}-p_b+1}{}' & \ldots & \mathbf{Q}_{\mathrm{m}}{}'\end{bmatrix}^{\mathrm{T}}$$

$$s'(t) = (t - t_m{}')/\Delta t'$$

assumption: $t_1 = 0$

$t_m{}' = r_e t_m$

$$\mathbf{\Phi}_f(r_e t) = \mathbf{\Phi}_s(t), t \in trajectory\ duration\ of\ \mathbf{\Phi}_s$$

$$\mathbf{s}'(t_{p_b+1}{}')^{\mathrm{T}}\mathbf{M}_{p_b+1}\begin{bmatrix}\mathbf{Q}_1{}' & \mathbf{Q}_2{}' & \ldots & \mathbf{Q}_{p_b+1}{}'\end{bmatrix}^{\mathrm{T}} = \mathbf{\Phi}_f(r_e t_{p_b+1}) = \mathbf{\Phi}_s(t_{p_b+1})$$

$$\mathbf{s}'(t_{p_b+2}{}')^{\mathrm{T}}\mathbf{M}_{p_b+1}\begin{bmatrix}\mathbf{Q}_2{}' & \mathbf{Q}_3{}' & \ldots & \mathbf{Q}_{p_b+2}{}'\end{bmatrix}^{\mathrm{T}} = \mathbf{\Phi}_f(r_e t_{p_b+2}) = \mathbf{\Phi}_s(t_{p_b+2})$$

$$\vdots$$

$$\mathbf{s}'(t_N{}')^{\mathrm{T}}\mathbf{M}_{p_b+1}\begin{bmatrix}\mathbf{Q}_{N-p_b}{}' & \mathbf{Q}_{N-p_b+1}{}' & \ldots & \mathbf{Q}_N{}'\end{bmatrix}^{\mathrm{T}} = \mathbf{\Phi}_f(r_e t_N) = \mathbf{\Phi}_s(t_N)$$

+

At the start and end of the trajectory

$$\dot{\mathbf{\Phi}}_f(r_e t) = \dot{\mathbf{\Phi}}_s(t)$$

$$\ddot{\mathbf{\Phi}}_f(r_e t) = \ddot{\mathbf{\Phi}}_f(t)$$

a closed-form min-least square problem: $\mathbf{AQ}' = \mathbf{b}$ $\quad \mathbf{Q}' = \begin{bmatrix}\mathbf{Q}_1{}' & \mathbf{Q}_2{}' & \ldots & \mathbf{Q}_N{}'\end{bmatrix}^{\mathrm{T}}$

Back-End

◆ Trajectory Refinement: Optimization

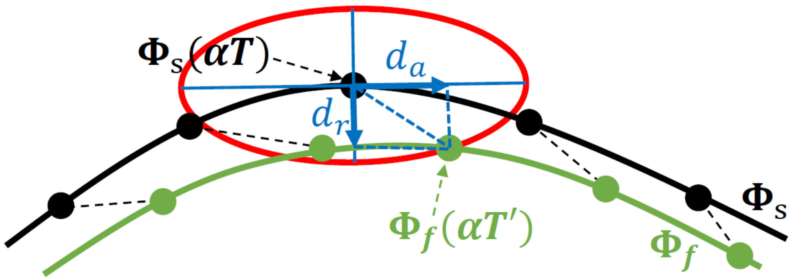
- After obtaining the initial value of $\mathbf{\Phi}_f$, the smoothness and feasibility are then refined by optimization.

$$\min_{\mathbf{Q}} \mathcal{J}' = \lambda_s \underbrace{\boxed{\mathcal{J}_s}}_{\text{smoothness}} + \lambda_f \underbrace{\boxed{\mathcal{J}_f}}_{\boxed{\textbf{curve fitting}}} + \lambda_d \underbrace{\boxed{\mathcal{J}_d}}_{\text{dynamic feasibility}}$$

Back-End

Trajectory Refinement: Fitting penalty $\mathcal{J}_f$

- $\mathcal{J}_f$ is formulated as the integral of anisotropic displacements from points $\mathbf{\Phi}_f(\alpha T')$ to the corresponding $\mathbf{\Phi}_s(\alpha T)$, where T' and T are the trajectory duration of $\mathbf{\Phi}_f$ and $\mathbf{\Phi}_s$, $\alpha \in [0,1]$
- Since the fitted curve $\mathbf{\Phi}_s$ is already collision-free, we assign the axial displacement of two curves with low penalty weight to relax smoothness adjustment restriction, and radial displacement with high penalty weight to avoid collision.
- To achieve this, we use the spheroidal metric.



$$d_a(\alpha) = \left(\mathbf{\Phi}_f(\alpha T') - \mathbf{\Phi}_s(\alpha T)\right) \cdot \frac{\dot{\mathbf{\Phi}}_s(\alpha T)}{\left\|\dot{\mathbf{\Phi}}_s(\alpha T)\right\|}$$

$$d_r(\alpha) = \left\|\left(\mathbf{\Phi}_f(\alpha T') - \mathbf{\Phi}_s(\alpha T)\right) \times \frac{\dot{\mathbf{\Phi}}_s(\alpha T)}{\left\|\dot{\mathbf{\Phi}}_s(\alpha T)\right\|}\right\|$$

$$\mathcal{J}_f = \int_0^1 \left(\frac{d_a(\alpha)^2}{a^2} + \frac{d_r(\alpha)^2}{b^2}\right) \mathrm{d}\alpha$$

where a and b are semi-major and semi-minor axis of the ellipse.

Thanks for Listening!