Optimization-Based Trajectory Planning

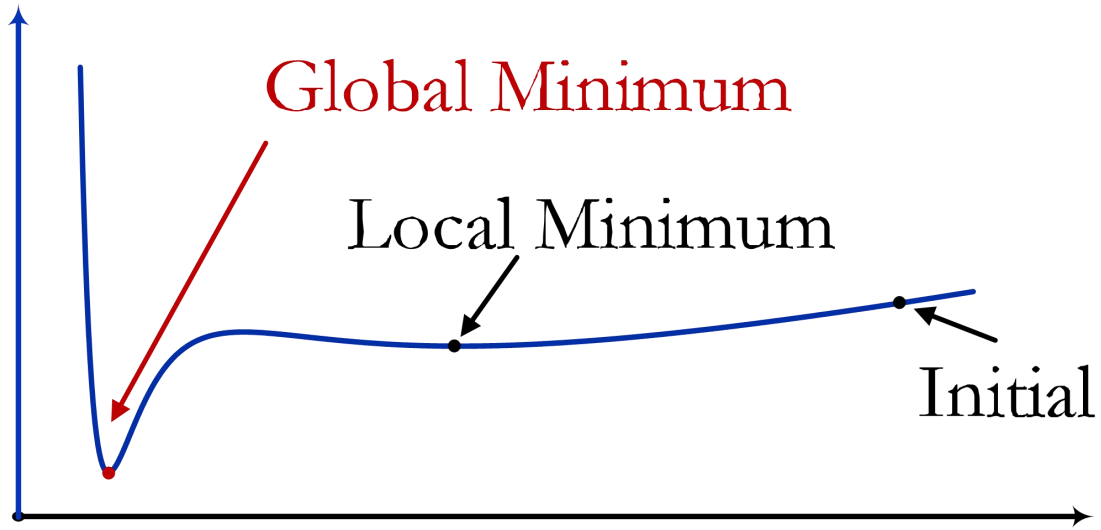
- Lecture 5

Preliminaries

Global and local methods

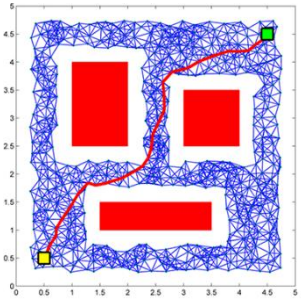
Global Methods: Exploration and Exploitation

Local Methods: Deterministic Optimization

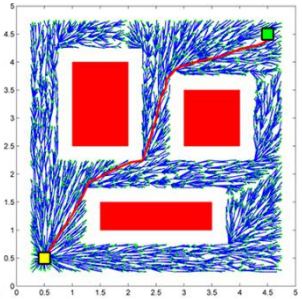


Global and local methods

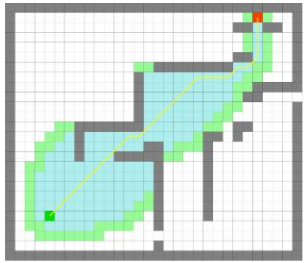
Global Methods: Sampling + Graph Search



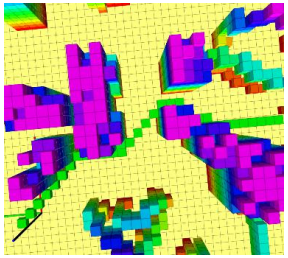
PRM*



RRT*



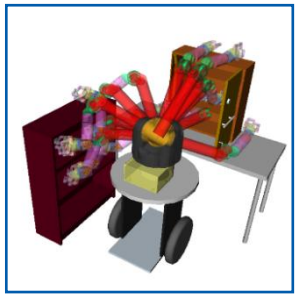
A*



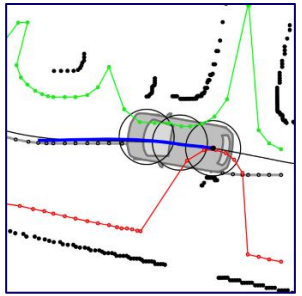
JPS

PRM*/RRT*: https://ompl.kavrakilab.org/
A*: https://github.com/qiao/PathFinding.js/
JPS: https://github.com/KumarRobotics/jps3d

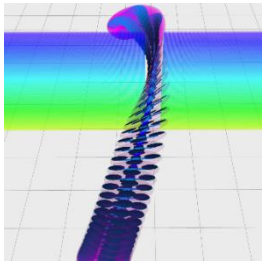
Local Methods: Deterministic Optimization



CHOMP



DDP/iLQR



Flatness

MPC/NMPC

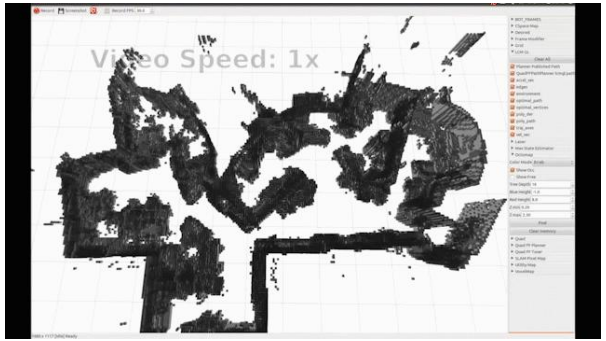
CHOMP: https://github.com/ros-planning/moveit
DDP/iLQR: https://github.com/anassinator/ilqr
MPC/NMPC: https://www.embotech.com/products/forcespro/overview/
Flatness: https://github.com/ZJU-FAST-Lab/GCOPTER

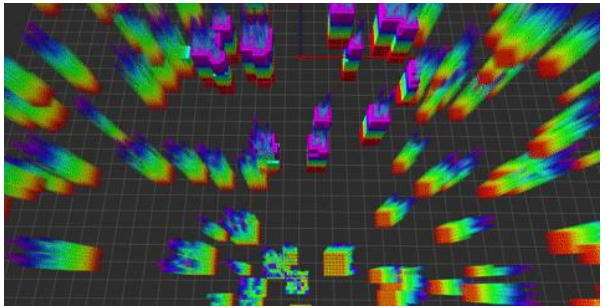
Global and local methods

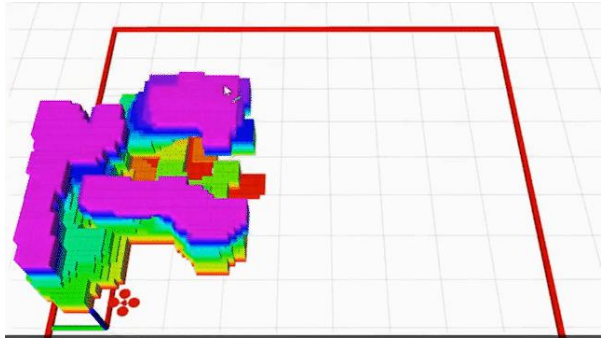
	Global Methods	Local Methods
Pros	• Global optimality • Handling environmental complexity • Portable • Zero-order Information	• Local optimality • Handling dynamics complexity • Fast in high-dimensional space • Promising convergence rate
Cons	• Slow in high-dimensional space • Inconvenient to incorporate dynamics • Poor convergence rate	• More involved • High-order information • Shallow local minima

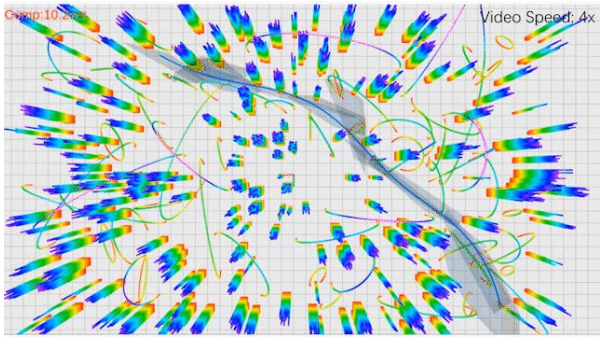
Some works try to combine both, but still inherit most disadvantages.

Global and local methods







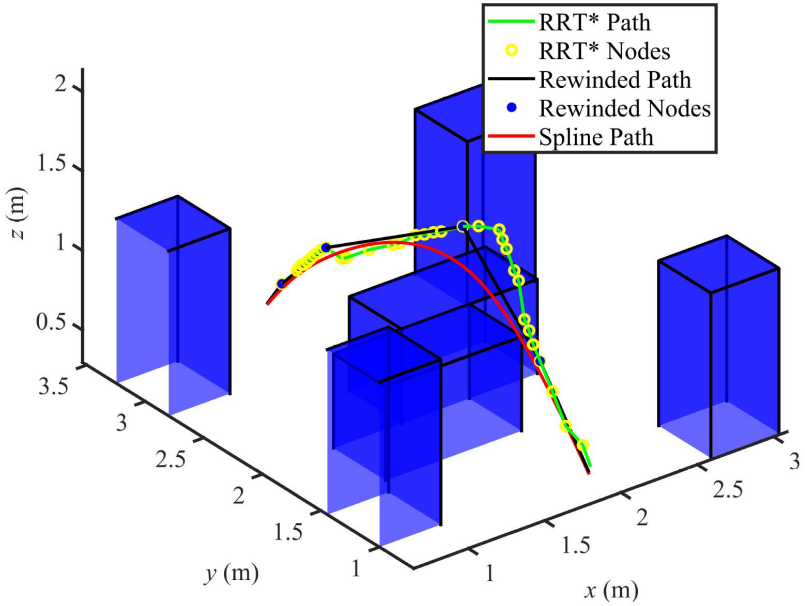


Global and local methods are often deployed in a decoupled way (front end + back end).

Bry et al., Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments, IJRR 2015.
Gao et al., Gradient-based online safe trajectory generation for quadrotor flight in complex environments, IROS 2017.
Zhou et al., Ego-planner: An esdf-free gradient-based local planner for quadrotors. RAL 2020.
Wang et al., Generating Large-Scale Trajectories Efficiently using Double Descriptions of Polynomials, ICRA 2021.

Trajectory Planning

What are trajectories



What are trajectories?

Are smooth curves trajectories?

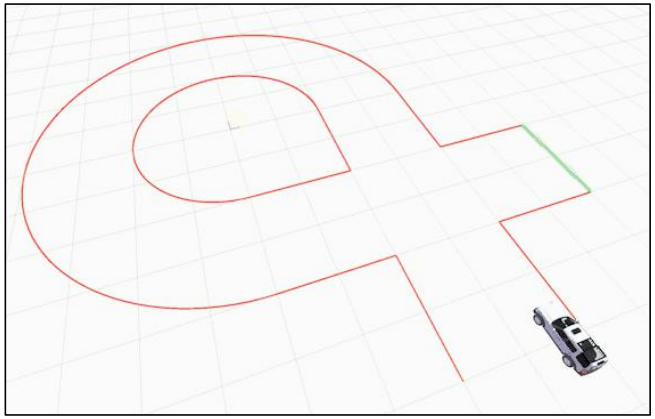
Are trajectories always smooth?

Can nonsmooth paths be trajectories?

Briefly: Trajectories are time-parameterized paths.

Trajectories consist of spatial and temporal profiles.

What are trajectories



Position

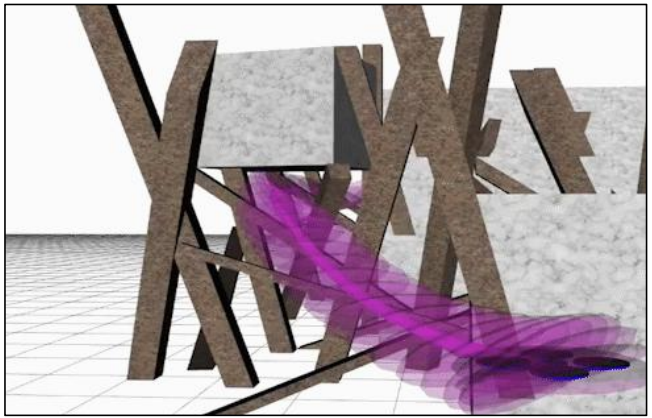
$$[x(t), y(t)] : [0, T] \mapsto \mathbb{R}^2$$

Position
Velocity
Yaw

$$[x(t), y(t), v_x(t), v_y(t), \psi(t)] : [0, T] \mapsto \mathbb{R}^4 \times \mathrm{SO}(2)$$

A trajectory maps time to

configuration space, state-input space, flat space, and so on.



Position
Yaw

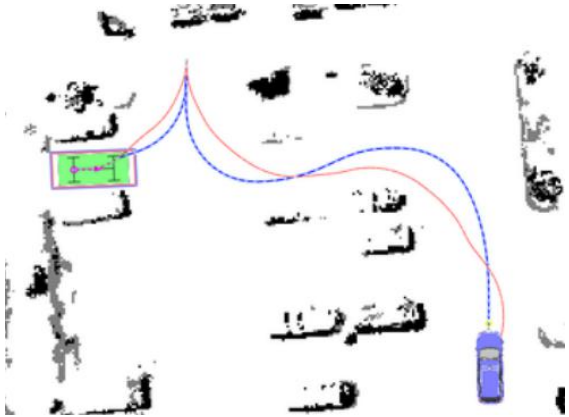
$$[x(t), y(t), z(t), \psi(t)] : [0, T] \mapsto \mathbb{R}^3 \times \mathrm{SO}(2)$$

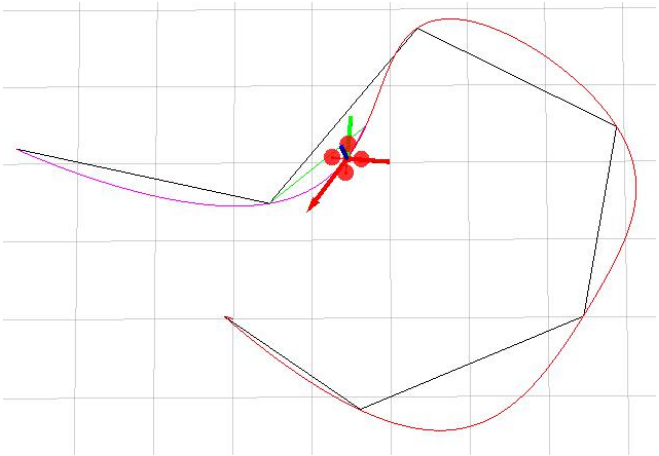
Position
Velocity
Attitude
Angular

$$[\mathbf{r}(t), \mathbf{v}(t), \mathbf{R}(t), \mathbf{w}(t)] : [0, T] \mapsto \mathbb{R}^6 \times \mathrm{SO}(3) \times \mathbb{R}^3$$

What does it mean by smoothness

- Smoothness in trajectory planning is not a geometrical concept.
- A variety of high-quality trajectories have non-smooth shapes.





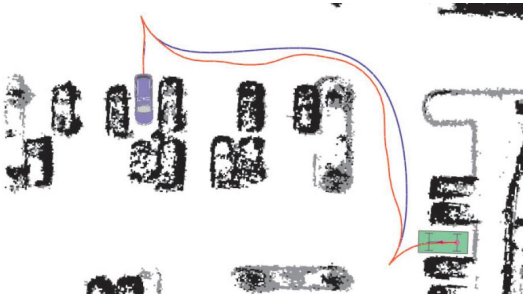
A smooth trajectories should at least:

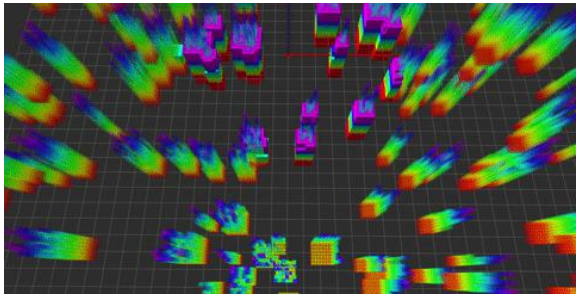
1. satisfy differential constraints of dynamics, $\dot{x} = f(x, u)$

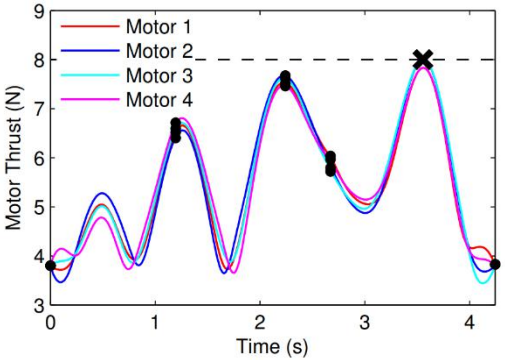
2. minimize an energy functional of state x and input u. $\min \int_0^T \mathcal{L}(x, u)\mathrm{d}t$

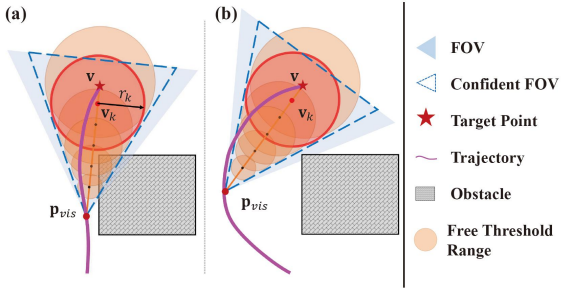
Why trajectory optimization

1. If we have the front-end (path finding?), why back-end optimization necessary?
2. If the front-end is kinodynamic feasible, why back-end optimization necessary?









Energy efficiency,
Time efficiency,
Actuator limits,
Mission requirements,
...

Differential Flatness

Differential flatness

Consider a dynamical system with

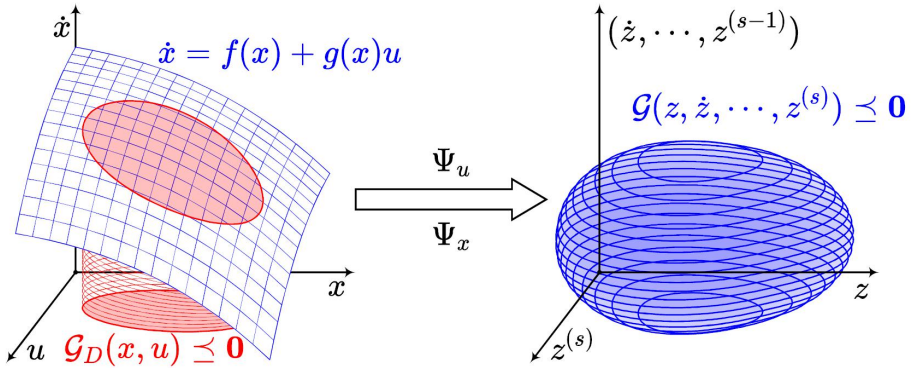
$$\dot{x} = f(x) + g(x)u,$$
$$f\colon \mathbb{R}^n \mapsto \mathbb{R}^n,\ g\colon \mathbb{R}^n \mapsto \mathbb{R}^n,\ x \in \mathbb{R}^n,\ u \in \mathbb{R}^m,\ \mathrm{rank}(g) = m.$$

The system is *differentially flat*, if there exists $z \in \mathbb{R}^m$ which can be determined by the state $x \in \mathbb{R}^n$ and finite derivatives of $u \in \mathbb{R}^m$.

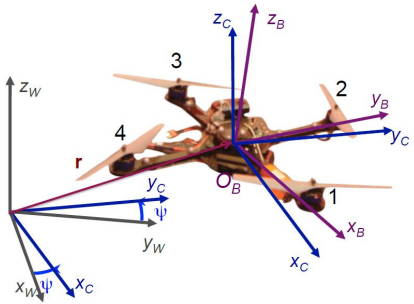
$z \in \mathbb{R}^m$ is called the *flat output*, whose finite derivatives can uniquely determine all state and input:

$$x = \Psi_x(z, \dot{z}, \cdots, z^{(s-1)}),$$
$$u = \Psi_u(z, \dot{z}, \cdots, z^{(s)}).$$

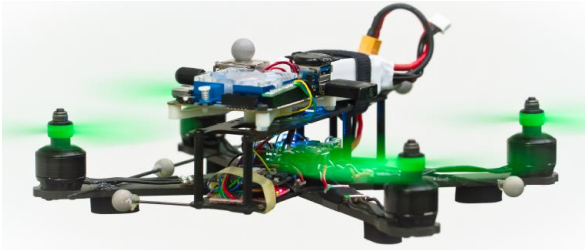
Differential flatness eliminates differential constraints.



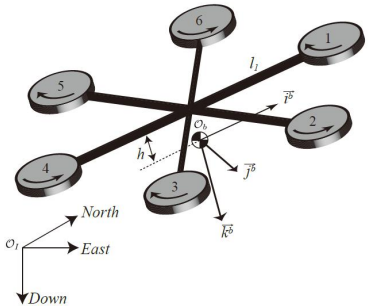
Multicopter dynamics and differential flatness



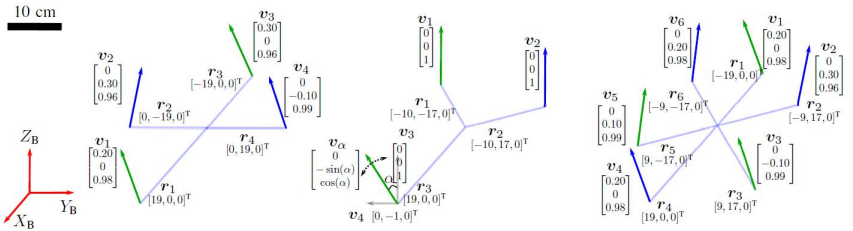
Quadcopters with aligned propellers



Multicopters subject to linear drag



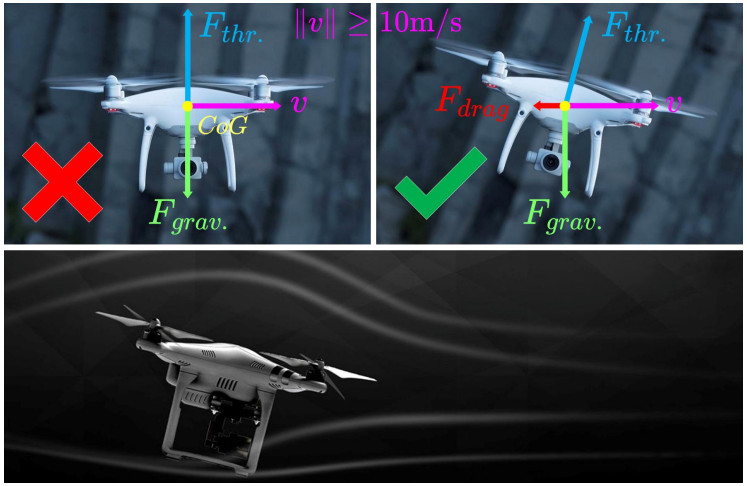
Hexacopters with aligned propellers



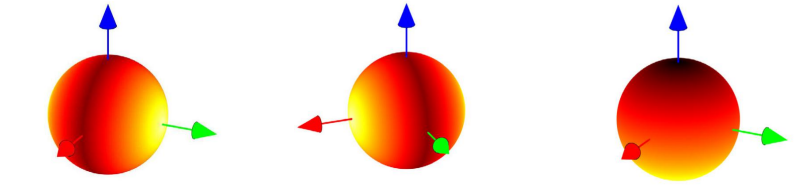
Multicopters with tilted propellers s.t. rank conditions

Mellinger et al., Minimum Snap Trajectory Generation and Control for Quadrotors, ICRA 2011.
Faessler et al., Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag for Accurate Tracking of High-Speed Trajectories, RAL 2017.
Ferrin et al., Differential Flatness Based Control of a Rotorcraft For Aggressive Maneuvers. IROS 2011.
Mu et al., Trajectory Generation for Underactuated Multirotor Vehicles with Tilted Propellers via a Flatness-based Method, AIM 2019.
Watterson et al., Control of Quadrotors Using the Hopf Fibration on SO(3), ISRR 2017.

Multicopter dynamics and differential flatness



Drag-free dynamics is low-fidelity.



Singularity at $\pm 90^\circ$ Roll. Singularity at $\pm 90^\circ$ Pitch. Singularity at $z_b = -e_3$.

1	✓		
2		✓	
3		✓	
4	✓	✓	
5			✓
6			✓

Yaw defined by Euler/Proj. is cursed with singularity.

1. Yaw defined by Z-X-Y Euler angle without drag.
2. Yaw defined by Z-Y-X Euler angle without drag. (Mellinger ICRA 2011, error fixed in Faessler RAL 2017)
3. Yaw defined by body X axis projection without drag.
4. Yaw defined in 3 with linear drag. (Faessler RAL 2017)
5. Yaw defined by Hopf fibration. (Watterson ISRR 2017)
6. SE(3) controller singularity. (Lee CDC 2010)

Differentially flat multicopter dynamics should involve drag but reduce singularity.

Lee et al., Geometric tracking control of a quadrotor UAV on SE(3), CDC 2010

Multicopter dynamics and differential flatness

Multicopter State

$$x = \{r, v, R, \omega\} \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathrm{SO}(3) \times \mathbb{R}^3$$

position, velocity, attitude, body rate

Multicopter Control Input (After Input Mapping)

$$u = \{f, \tau\} \in \mathbb{R}_{\geq 0} \times \mathbb{R}^3$$

collective thrust, torque

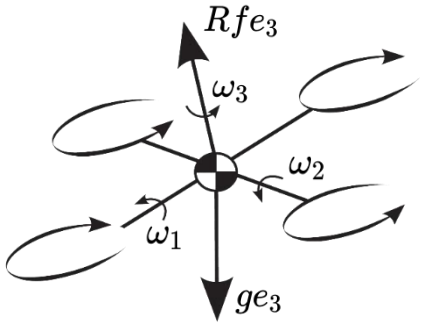
Multicopter Nonlinear Dynamics

$$\begin{cases} \dot{r} = v, \\ m\dot{v} = -mge_3 - RDR^{\mathrm{T}}\sigma(\|v\|)v + Rfe_3, \\ \dot{R} = R\hat{\omega}, \\ M\dot{\omega} = \tau - \omega \times M\omega - A(\omega) - B(R^{\mathrm{T}}v). \end{cases}$$

Multicopter Flat Output

$$z = \{r, \psi\} \in \mathbb{R}^3 \times \mathrm{SO}(2)$$

position, yaw heading



$m \in \mathbb{R}_{\geq 0}$	vehicle mass
$g \in \mathbb{R}_{\geq 0}$	gravitational acceleration
$e_3 = (0, 0, 1)^{\mathrm{T}}$	unit vector
$D = \mathrm{Diag}(d_h, d_h, d_v)$	drag force coefficients
σ: $\mathbb{R}_{\geq 0} \mapsto \mathbb{R}_{\geq 0}$	user-defined nonlinear term
$M \in \mathbb{R}_{\geq 0}^{3\times 3}$	moment of inertia
A: $\mathbb{R}^3 \mapsto \mathbb{R}^3$	user-defined torque induced body rate
B: $\mathbb{R}^3 \mapsto \mathbb{R}^3$	user-defined torque induced velocity

Wang et al., Robust Trajectory Planning for Spatial-Temporal Multi-Drone Coordination in Large Scenes, arXiv 2109.08403

Multicopter dynamics and differential flatness

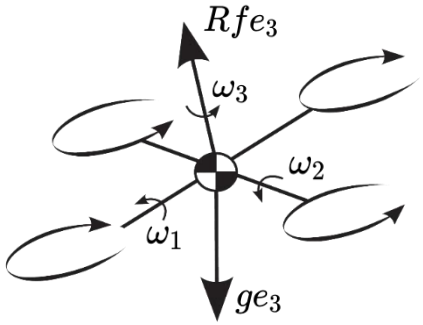
Flatness Transformation?

$$z = \{r, \psi\} \in \mathbb{R}^3 \times \mathrm{SO}(2)$$

$$x = \Psi_x(z, \dot{z}, \cdots, z^{(s-1)}),$$
$$u = \Psi_u(z, \dot{z}, \cdots, z^{(s)}).$$

?

$$x = \{r, v, R, \omega\} \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathrm{SO}(3) \times \mathbb{R}^3$$
$$u = \{f, \tau\} \in \mathbb{R}_{\geq 0} \times \mathbb{R}^3$$



Obviously, we have

$$r = r \qquad v = \dot{r}$$

Dot product the Newton equation by X and Y axes of the body frame:

$$x_b = Re_1,\ y_b = Re_2$$
$$(Re_i)^{\mathrm{T}} m\dot{v} = (Re_i)^{\mathrm{T}} \left(-mge_3 - RDR^{\mathrm{T}}\sigma(\|v\|)v + Rfe_3\right),\ \forall i \in \{1, 2\}$$

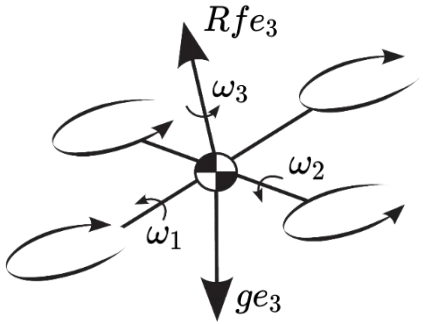
These yield

$$(Re_i)^{\mathrm{T}}(\dot{v} + \frac{d_h}{m}\sigma(\|v\|)v + ge_3) = 0,\ \forall i \in \{1, 2\}.$$

Multicopter dynamics and differential flatness

What does it mean?

$$(Re_i)^{\mathrm{T}}(\dot{v} + \frac{d_h}{m}\sigma(\|v\|)v + ge_3) = 0,\ \forall i \in \{1, 2\}.$$



Geometrically, we have

$$x_b \perp (\dot{v} + \frac{d_h}{m}\sigma(\|v\|)v + ge_3)$$
$$y_b \perp (\dot{v} + \frac{d_h}{m}\sigma(\|v\|)v + ge_3)$$

Since the thrust force and body Z axis share the same direction,

$$z_b = \mathcal{N}(\dot{v} + \frac{d_h}{m}\sigma(\|v\|)v + ge_3),\ \mathcal{N}(x) := x/\|x\|_2$$

Dot product the Newton equation by Z axis of the body frame:

$$(Re_3)^{\mathrm{T}}m\dot{v} = (Re_3)^{\mathrm{T}}\left(-mge_3 - RDR^{\mathrm{T}}\sigma(\|v\|)v + Rfe_3\right)$$

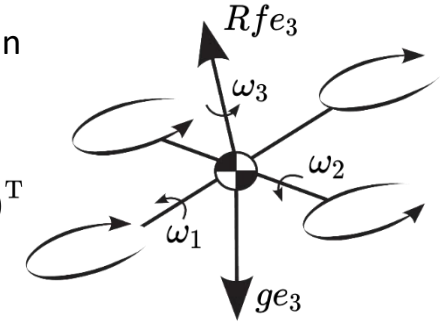
We obtain

$$f = z_b^{\mathrm{T}}(m\dot{v} + d_v\sigma(\|v\|)v + mge_3)$$

Multicopter dynamics and differential flatness

Now that the yaw heading and body Z axis are both known, the yaw rotation quaternion is $q_\psi = (\cos(\psi/2), 0, 0, \sin(\psi/2))^{\mathrm{T}}$

The tilt rotation quaternion is $q_z = \frac{1}{\sqrt{2(1+z_b(3))}}(1+z_b(3), -z_b(2), z_b(1), 0)^{\mathrm{T}}$



The tilt rotation has no component for inertial Z axis , since it is decomposed using Hopf fibration. The attitude quaternion of vehicle is thus

$$q = q_z \otimes q_\psi$$

Expanding it yields

$$q = \frac{1}{\sqrt{2(1+z_b(3))}} \begin{pmatrix} (1+z_b(3))\cos(\psi/2) \\ -z_b(2)\cos(\psi/2) + z_b(1)\sin(\psi/2) \\ z_b(1)\cos(\psi/2) + z_b(2)\sin(\psi/2) \\ (1+z_b(3))\sin(\psi/2), \end{pmatrix}$$

Thus the attitude rotation matrix is uniquely determined, $R = \mathcal{R}_{quat}(q)$

Quaternion product, inverse, and rotation matrix formula are detailed by Vince.

Vince, Quaternions for Computer Graphics, Springer 2011.

Multicopter dynamics and differential flatness

Now the rotation is known.

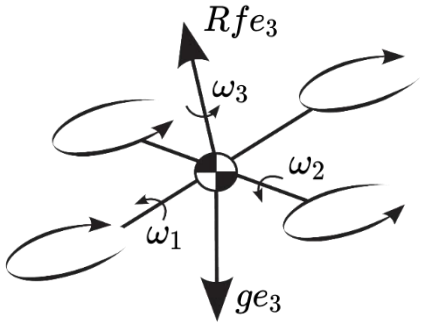
$$\dot{R} = R\hat{\omega}$$

This implies

$$\omega = (R^{\mathrm{T}}\dot{R})^{\vee}$$

Equivalently,

$$\omega = 2(q_z \otimes q_\psi)^{-1} \otimes \left(\dot{q}_z \otimes q_\psi + q_z \otimes \dot{q}_\psi\right)$$



Substituting the tilt rotation quaternion and yaw rotation quaternion gives

$$\omega = \begin{pmatrix} \dot{z}_b(1)\sin(\psi) - \dot{z}_b(2)\cos(\psi) - \dot{z}_b(3)(z_b(1)\sin(\psi) - z_b(2)\cos(\psi))/(1+z_b(3)) \\ \dot{z}_b(1)\cos(\psi) + \dot{z}_b(2)\sin(\psi) - \dot{z}_b(3)(z_b(1)\cos(\psi) + z_b(2)\sin(\psi))/(1+z_b(3)) \\ (z_b(2)\dot{z}_b(1) - z_b(1)\dot{z}_b(2))/(1+z_b(3)) + \dot{\psi}, \end{pmatrix}$$

Differentiate body Z axis gives

$$\dot{z}_b = \frac{d_h}{m}\mathcal{DN}(\dot{v} + \frac{d_h}{m}\sigma(\|v\|)v + ge_3)^{\mathrm{T}}(\frac{m}{d_h}\ddot{v} + \sigma(\|v\|)\dot{v} + \dot{\sigma}(\|v\|)\frac{v^{\mathrm{T}}\dot{v}}{\|v\|}v),$$

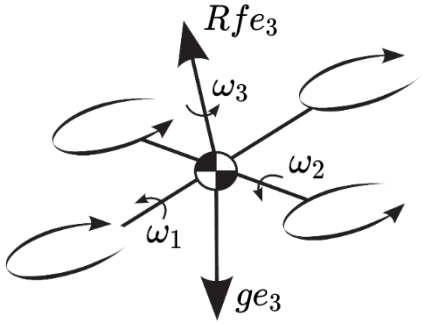
$$\mathcal{DN}(x) := \frac{1}{\|x\|}\left(\mathbf{I} - \frac{xx^{\mathrm{T}}}{x^{\mathrm{T}}x}\right).$$

Now we have expressed the body rate using finite derivatives of the flat output.

Multicopter dynamics and differential flatness

Now all states and inputs are known except the torque.

We trivially differentiate the body rate to get its expression using $r^{(4)},\ \psi^{(2)}$



Consequently,

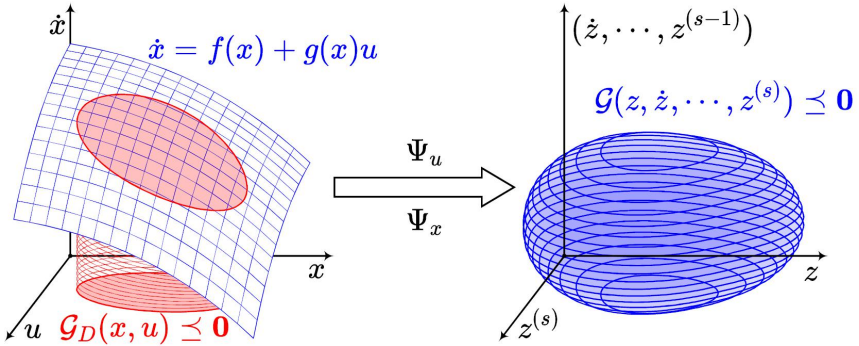
$$\tau = M\dot{\omega} + \omega \times M\omega + A(\omega) + B(R^{\mathrm{T}}v)$$

Finally, we have finished the derivation of **flatness transformation**.

$$z = \{r, \psi\} \in \mathbb{R}^3 \times \mathrm{SO}(2)$$

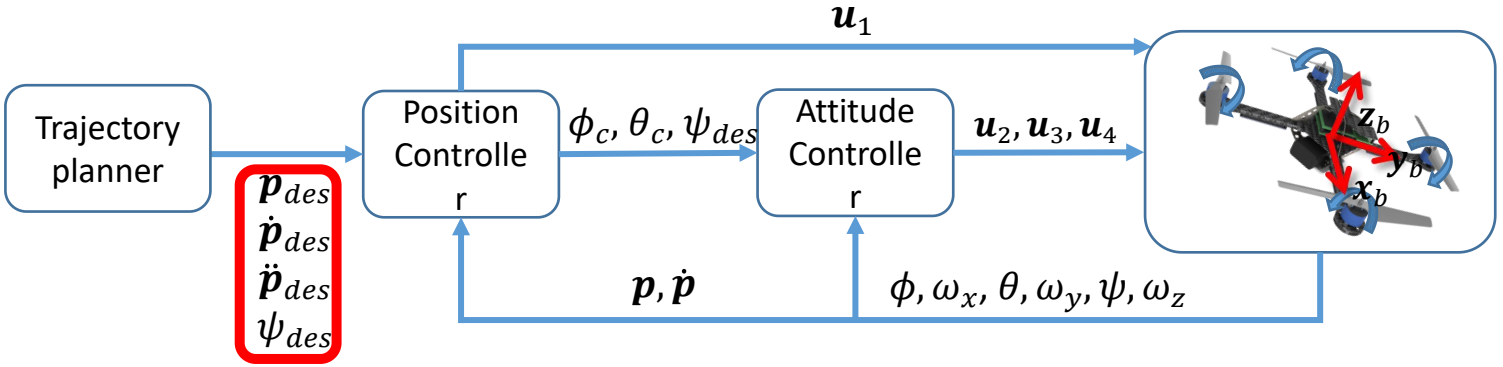
$$x = \Psi_x(z, \dot{z}, \cdots, z^{(s-1)}),$$
$$u = \Psi_u(z, \dot{z}, \cdots, z^{(s)}).$$

$$x = \{r, v, R, \omega\} \in \mathbb{R}^3 \times \mathbb{R}^3 \times \mathrm{SO}(3) \times \mathbb{R}^3$$
$$u = \{f, \tau\} \in \mathbb{R}_{\geq 0} \times \mathbb{R}^3$$



Planning flat-output trajectories with high-order continuity suffices for the dynamics.

Flatness-based planning-control loop



Nonlinear Dynamics and Flatness Transformation in Feedback and Feedforward Control

$$\begin{cases} \dot{r} = v, \\ m\dot{v} = -mge_3 - RDR^{\mathrm{T}}\sigma(\|v\|)v + Rfe_3, \\ \dot{R} = R\hat{\omega}, \\ M\dot{\omega} = \tau - \omega \times M\omega - A(\omega) - B(R^{\mathrm{T}}v). \end{cases}$$

Parameterize flat output via splines

Flat output

$$z = \{r, \psi\} \in \mathbb{R}^3 \times \mathrm{SO}(2)$$

A trajectory in the flat-output space

$$z(t)\colon\ [0, T] \mapsto \mathbb{R}^3 \times \mathrm{SO}(2)$$

Advantages for parameterizing flat-output trajectory via splines

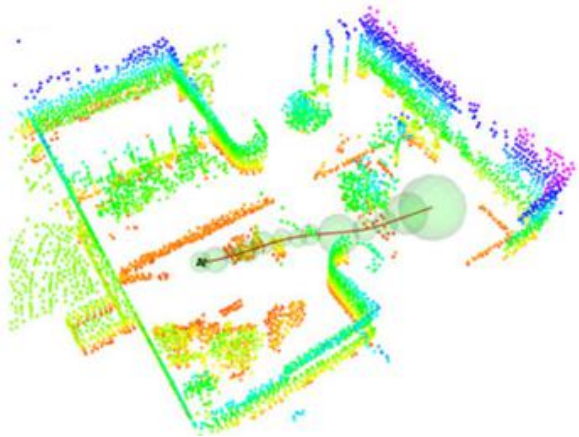
- Easy determination of smoothness criterion
- Easy and closed-form calculation of derivatives
- Decoupled trajectory generation in three dimensions
- High numerical stability

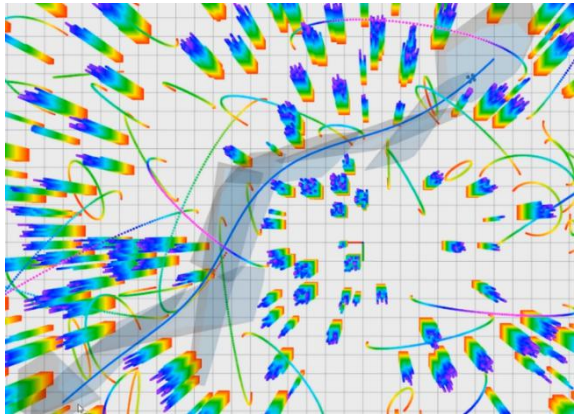
Trajectory Optimization

General problem formulation

$$
\begin{aligned}
&\min_{z(t),T} \int_0^T v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t + \rho(T), \\
&s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [0,T], \\
&\qquad \mathcal{G}(z(t), \cdots, z^{(s)}(t)) \preceq \mathbf{0},\ \forall t \in [0,T], \\
&\qquad z(t) \in \mathcal{F},\ \forall t \in [0,T], \\
&\qquad z^{[s-1]}(0) = \bar{z}_o,\ z^{[s-1]}(T) = \bar{z}_f, \\
&\qquad z^{[s-1]} := (z^{\mathrm{T}}, \dot{z}^{\mathrm{T}}, \cdots, {z^{(s-1)}}^{\mathrm{T}})^{\mathrm{T}}.
\end{aligned}
$$

General but hard





General problem formulation

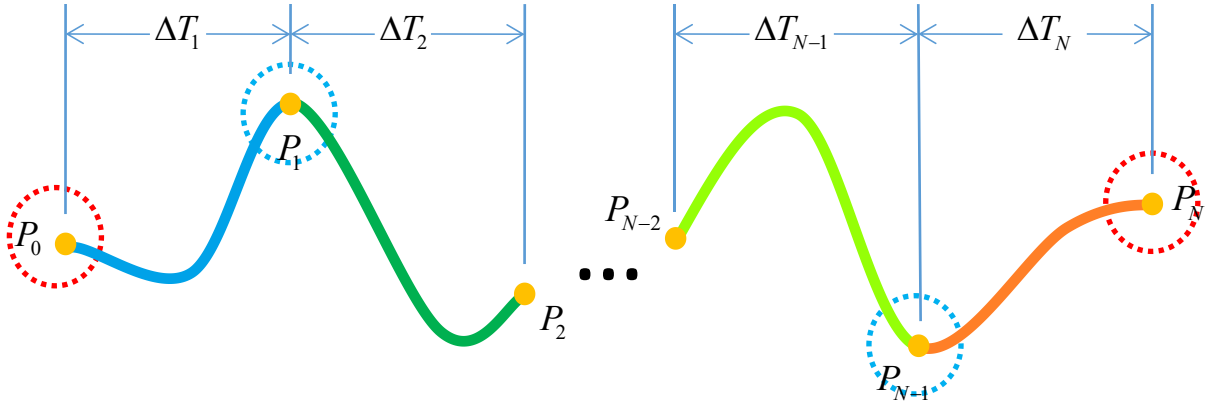
Explicitly minimize certain derivatives in the space of flat outputs:

- Min jerk: minimize angular velocity, good for visual tracking
- Min snap: minimize differential thrust, saves energy

Derivative(s)	Translation	Rotation	Thrust
1	Velocity		
2	Acceleration	Rotation	
3	Jerk	Angular Velocity	Thrust
4	Snap	Angular Acceleration	Differential Thrust

Unconstrained Case

Unconstrained case



$$\min_{z(t)} \int_{t_0}^{t_M} v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t,$$

$$s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [t_0, t_M],$$

Boundary value: $z^{[s-1]}(t_0) = \bar{z}_o,\ z^{[s-1]}(t_M) = \bar{z}_f,$

Intermediate value: $z^{[d_i-1]}(t_i) = \bar{z}_i,\ 1 \le i < M,$

$$t_{i-1} < t_i,\ 1 \le i \le M.$$

Theorem (Optimality Conditions). *A trajectory, denoted by $z^*(t) : [t_0, t_M] \mapsto \mathbb{R}^m$, is optimal, if and only if the following conditions are satisfied:*

- *The map $z^*(t) : [t_{i-1}, t_i] \mapsto \mathbb{R}^m$ is parameterized as a $2s-1$ degree polynomial for any $1 \le i \le M$;*
- *The boundary and intermediate conditions all hold;*
- *$z^*(t)$ is $2s - d_i - 1$ times continuously differentiable at t_i for any $1 \le i < M$.*

Moreover, a unique trajectory exists for these conditions.

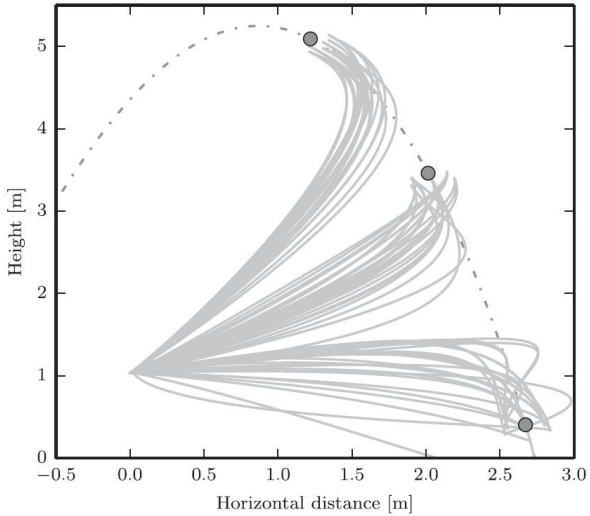
Wang et al., Geometrically Constrained Trajectory Optimization for Multicopters, TRO 2022

Unconstrained Case: BVP

Boundary value problem (BVP)

$$\min_{z(t)} \int_0^T v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t,$$
$$s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [0, T],$$
$$z^{[s-1]}(t_0) = \bar{z}_o,$$
$$z^{[s-1]}(t_M) = \bar{z}_f.$$

Boundary value



Motion primitives by Mueller et al. for ball catching.

Only boundary conditions are considered here.

The boundary value problem (BVP) is often adopted by motion primitive based planners:

Our optimality conditions show the solution is

a 5 degree polynomial when $s = 3$ (minimum jerk)

a 7 degree polynomial when $s = 4$ (minimum snap)

Wang et al., Geometrically Constrained Trajectory Optimization for Multicopters, TRO 2022

Smooth 1D trajectory via BVP

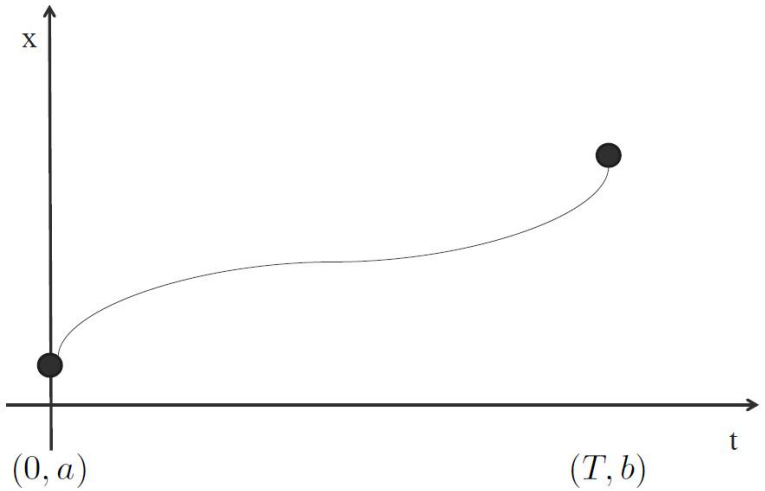
Design a trajectory $x(t)$ such that:

$$x(0) = a$$
$$x(T) = b$$

Boundary condition

Order of continuity ensured by parametrization

Continuity criteria



Smooth 1D trajectory via BVP

5^{th} order polynomial trajectory:

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$ ➡ Trajectory parametrization

Boundary conditions No intermediate condition

	Position	Velocity	Acceleration
$t = 0$	a	0	0
$t = T$	b	0	0

Solve:

$$\begin{matrix} t=0 \\ t=T \end{matrix} \left\{ \begin{matrix} a \\ 0 \\ 0 \\ b \\ 0 \\ 0 \end{matrix} \right. = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$$

$$d = \mathbf{A}_{\mathrm{F}}(T)c$$

Smooth multi-segment trajectory via BVPs

Generate each 5^{th} order polynomial independently:

$$x(t) = c_0 + c_1 t + c_2 t^2 + c_3 t^3 + c_4 t^4 + c_5 t^5$$

Boundary conditions

	Position	Velocity	Acceleration
$t = 0$	a	v_0	0
$t = T$	b	v_T	0

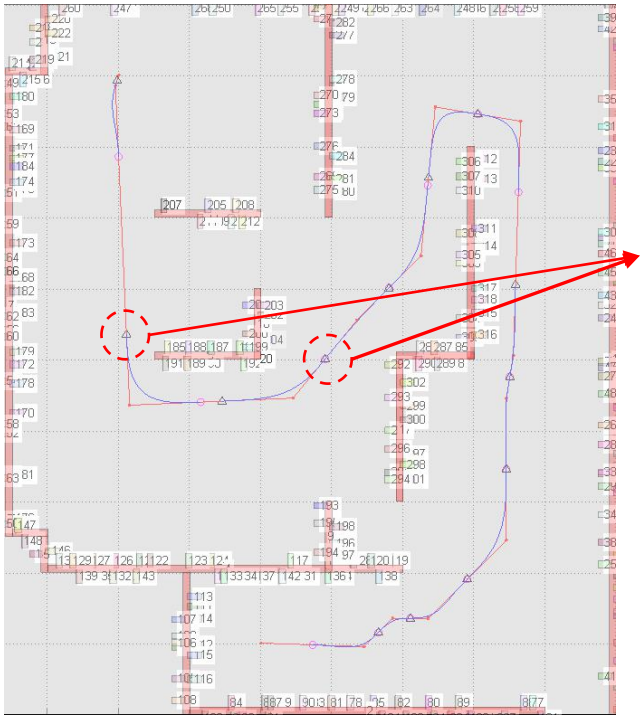
Solve:

$$\begin{bmatrix} a \\ v_0 \\ 0 \\ b \\ v_T \\ 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 & 5T^4 \\ 0 & 0 & 2 & 6T & 12T^2 & 20T^3 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix}$$

$$d = \mathbf{A}_{\mathrm{F}}(T)c$$

Smooth multi-segment trajectory via BVPs

- Smooth the corners of straight line segments.
- Preferred constant velocity motion at v.
- Preferred zero acceleration.
- Requires special handling of short segments.



Explicit solution to the BVP

$$\begin{aligned}
&\min_{z(t)} \int_0^T v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t, \\
&s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [0, T], \\
&\qquad z^{[s-1]}(t_0) = \bar{z}_o, \\
&\qquad z^{[s-1]}(t_M) = \bar{z}_f.
\end{aligned}$$

This BVP has an explicit solution.
Remember the solution is always a 2s-1 degree polynomial.

The coefficients and boundary conditions satisfy

$$d = \mathbf{A}_{\mathrm{F}}(T) c \qquad \mathbf{A}_F(t) = \begin{bmatrix} \mathbf{E} & \mathbf{0} \\ \mathbf{F}(t) & \mathbf{G}(t) \end{bmatrix}$$

in which

$$\mathbf{E}_{ij} = \begin{cases} (i-1)! & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$

$$\mathbf{F}_{ij}(t) = \begin{cases} (j-1)!/(j-i)! \cdot t^{j-i} & \text{if } i \leq j, \\ 0 & \text{if } i > j, \end{cases}$$

$$\mathbf{G}_{ij}(t) = \frac{(s+j-1)!}{(s+j-i)!} \cdot t^{s-i+j}.$$

The explicit solution of coefficients given boundaries are

$$c = \mathbf{A}_{\mathrm{B}}(T) d \qquad \mathbf{A}_B(t) = \begin{bmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{V}(t) & \mathbf{W}(t) \end{bmatrix}$$

in which

$$\mathbf{U}_{ij} = \begin{cases} 1/(i-1)! & \text{if } i = j, \\ 0 & \text{if } i \neq j, \end{cases}$$

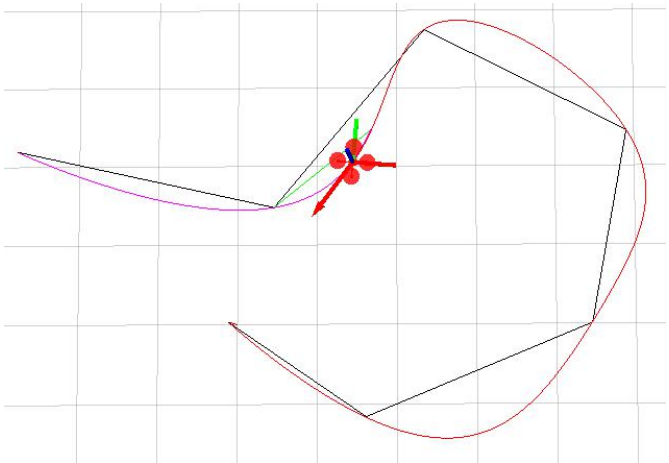
$$\mathbf{V}_{ij}(t) = \frac{\sum_{k=0}^{s-\max(i,j)} (-1)^k \binom{s}{i+k} \binom{2s-j-k-1}{s-1}}{(j-1)!\,(-1)^i \cdot t^{s+i-j}},$$

$$\mathbf{W}_{ij}(t) = \frac{\sum_{k=0}^{s-\max(i,j)} \binom{s-k-1}{i-1} \binom{2s-j-k-1}{s-1}}{(j-1)!\,(-1)^{i+j} \cdot t^{s+i-j}}.$$

Wang et al., Generating Large-Scale Trajectories Efficiently using Double Descriptions of Polynomials, ICRA 2021

Smooth multi-segment trajectory in 3D

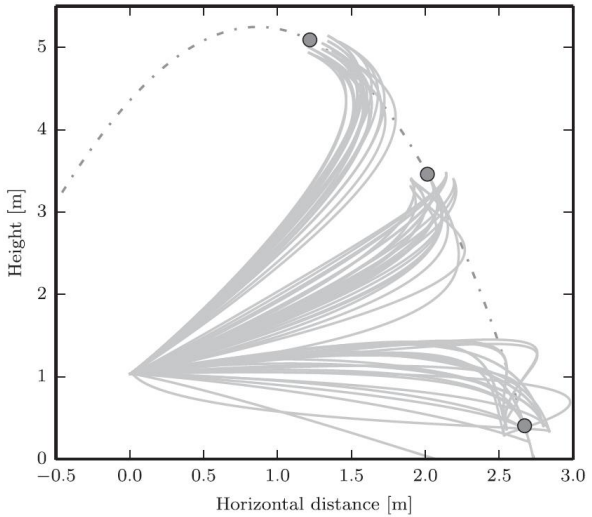
- Boundary condition: start, goal positions (orientations)
- Intermediate condition: waypoint positions (orientations)
  - Waypoints can be found by path planning (A*, RRT*, etc)
  - Introduced in previous 3 lectures
- Smoothness criterion
  - Generally translates into minimizing rate of change of “input”



Motion primitive via BVPs

$$\min_{z(t)} \int_0^T v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t,$$
$$s.t. \;\; z^{(s)}(t) = v(t), \; \forall t \in [0, T],$$
$$z^{[s-1]}(t_0) = \bar{z}_o,$$
$$z^{[s-1]}(t_M) = \bar{z}_f.$$

Boundary value

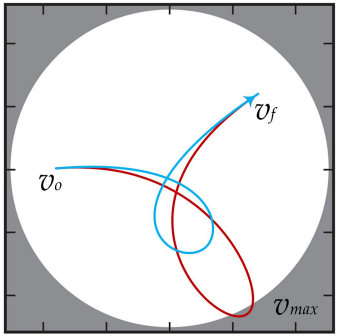


Motion primitives by Mueller et al. for ball catching.

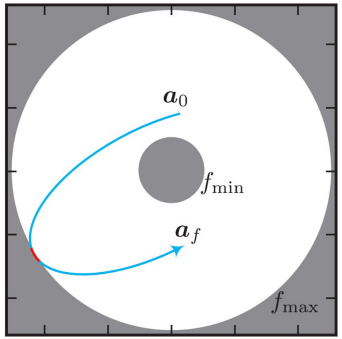
Checking after generating a lot of primitives.

How to check feasibility of primitives

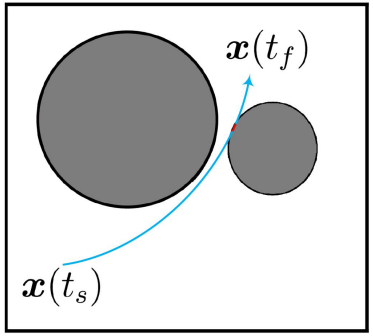
Continuous-time constraint for $p(t) = (p_1(t), p_2(t), p_3(t))^{\mathrm{T}},\ t \in [0, T]$



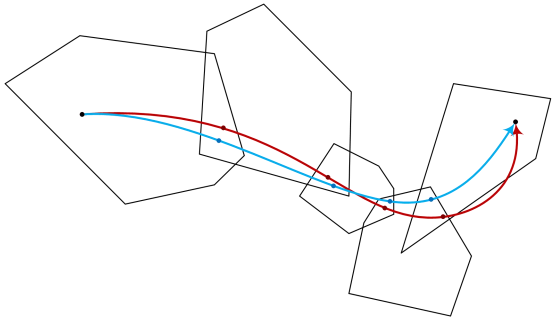
1.Maximum velocity rate.



2.Maximum\ minimum thrust.



3.Geometric obstacles.



4.Safe flight Corridor (Polyhedron/Ball-Shape).

$$G(p_1^{(i)}(t), p_2^{(i)}(t), p_3^{(i)}(t)) < 0, \quad \forall t \in [0, T]$$

$$G(a, b, c) := \sum_{e_1+e_2+e_3 \le d_g}^{d_c \in \mathbb{R}, e_j \in \mathbb{N}} d_c \cdot a^{e_1} b^{e_2} c^{e_3}$$

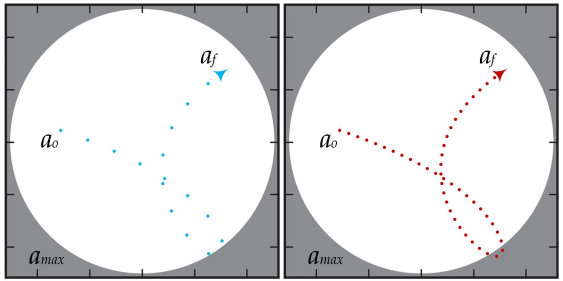
How to know whether the constraint is satisfied all the time?

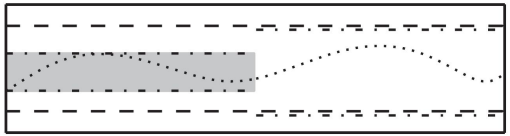
Continuous-time feasibility checkers!

(crucial to sampling based kinodynamic planning)

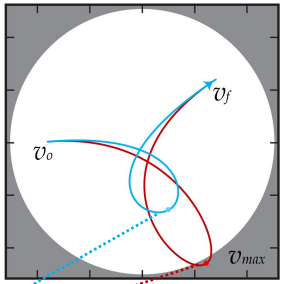
How to check feasibility of primitives

	Discrete time sampling	Recursive Bound Checker [1]	Extreme Value Checker [2]
Pros	• Flexibility	• Efficiency	• Stability
Cons	• False negative under low resolution. • Slow under high resolution.	• Only for thrust/acc/vel of degree 5 • Resolution dependent • Indeterminate cases	• Numerical iterations are slow. • Only for thrust/acc/vel of degree 5





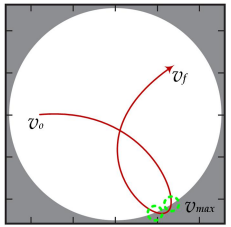
$$\max_{t\in\mathcal{T}} \|a(t)\|^2 \leq \sum_{i=1}^{3} \max_{t\in\mathcal{T}} a_i(t)^2$$

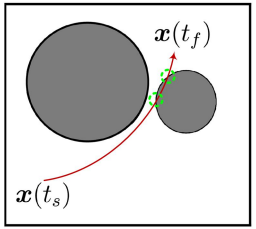


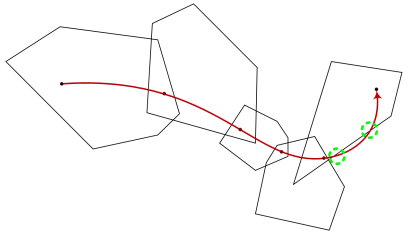
$$\frac{dv_{\text{norm}}(t)}{dt} = \frac{2\left(\dot{p}(t)_x \cdot \ddot{p}(t)_x + \dot{p}(t)_y \cdot \ddot{p}(t)_y + \dot{p}(t)_z \cdot \ddot{p}(t)_z\right)}{-\sqrt{(\dot{p}(t)_x)^2 + (\dot{p}(t)_y)^2 + (\dot{p}(t)_z)^2}}$$

[1] Mueller et al., A Computationally Efficient Motion Primitive for Quadrocopter Trajectory Generation, TRO 2015
[2] Burri et al., Real-Time Visual-Inertial Mapping, Re-localization and Planning Onboard MAVs in Unknown Environments, IROS 2015

How to check feasibility of primitives







$$G(p_1^{(i)}(t), p_2^{(i)}(t), p_3^{(i)}(t)) < 0, \quad \forall t \in [0, T]$$

The trajectory piece is feasible iff no intersection point!

$G(p_1^{(i)}(t), p_2^{(i)}(t), p_3^{(i)}(t))$ is a univariate polynomial of t. Counting root number of a poly? Sturm Theorem!

Sturm Sequence

$$g_0(t) = \mathcal{G}(t),$$
$$g_1(t) = \dot{\mathcal{G}}(t),$$
$$-g_{k+1}(t) = Rem(g_{k-1}(t), g_k(t))$$

Pure algebraic operations, *Rem* means the remainder of *Euclidean division* $r = Rem(a, b),$ $a = bq + r,\ 0 \leq |r| < |b|$

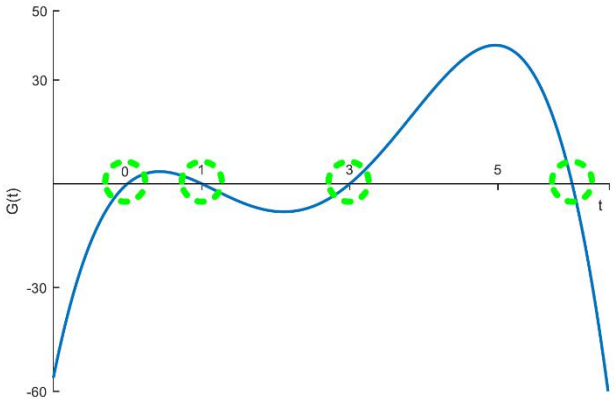
Wang et al., Alternating Minimization Based Trajectory Generation for Quadrotor Aggressive Flight, RAL 2020

How to check feasibility of primitives

Sturm Sequence

$$g_0(t) = \mathcal{G}(t),$$
$$g_1(t) = \dot{\mathcal{G}}(t),$$
$$-g_{k+1}(t) = Rem(g_{k-1}(t), g_k(t))$$

Pure algebraic operations



Example:

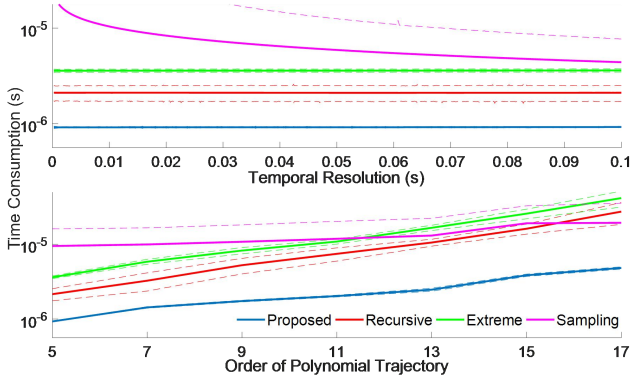
$G(t) = -t(t-1)(t-3)(t-6) = 18t - 27t^2 + 10t^3 - t^4,$
$g_0(t) = G(t),$
$g_1(t) = 18 - 54t + 30t^2 - 4t^3,$
$g_2(t) = -11.25 + 20.25t - 5.25t^2,$
$g_3(t) = 13.2245 - 10.7755t,$
$g_4(t) = -5.69473.$

$t = -1$, the sequence is $-56.00,\ 106.00,\ -36.75,\ 24.00,\ -5.69.$
sign variation number $= 4$
$t = 7$, the sequence is $-168.00\ -262.00,\ -126.75,\ -62.20,\ -5.69.$
sign variation number $= 0$
$4 - 0 = 4$ roots in the interval $(-1,\ 7)$

Wang et al., Alternating Minimization Based Trajectory Generation for Quadrotor Aggressive Flight, RAL 2020

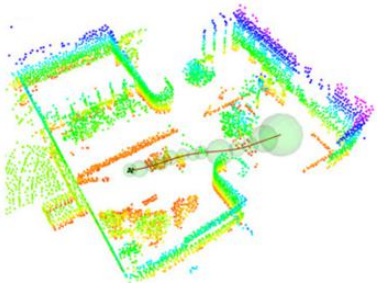
How to check feasibility of primitives

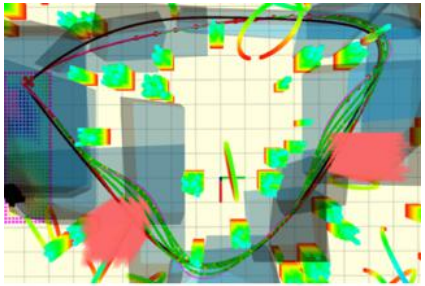
Performance on maximum velocity norm constraint.



The Proposed One: Fastest! Stable! Flexible! Reliable!

Previous methodology is easily extended to various constraints with convex decompositions.

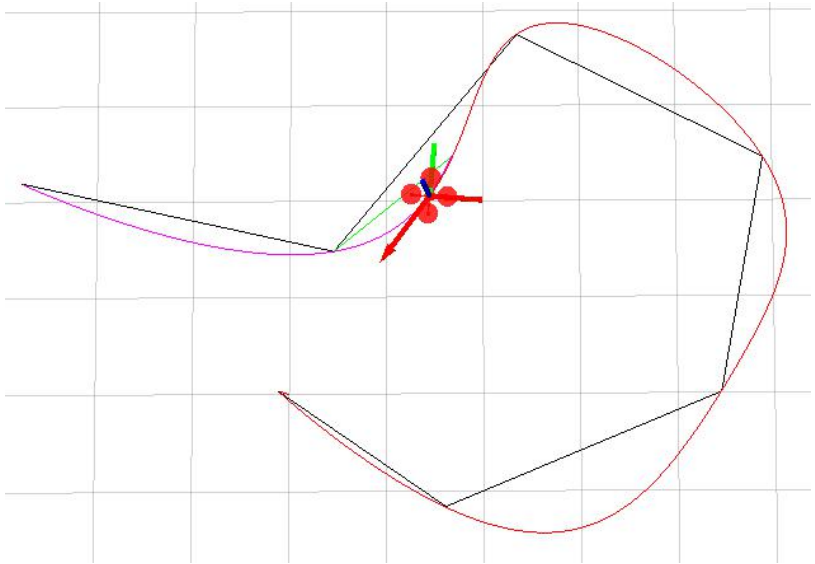




Unconstrained Case: BIVP

Multi-segment with intermediate values

- Multi-segment minimum snap trajectory



Boundary-intermediate value problem (BIVP)

$$\min_{z(t)} \int_{t_0}^{t_M} v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t,$$
$$s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [t_0, t_M],$$
Boundary value: $$z^{[s-1]}(t_0) = \bar{z}_o,\ z^{[s-1]}(t_M) = \bar{z}_f,$$
Intermediate value: $$z^{[d_i-1]}(t_i) = \bar{z}_i,\ 1 \le i < M,$$
$$t_{i-1} < t_i,\ 1 \le i \le M.$$

$\frac{\mathrm{d}}{\mathrm{d}t}$

- Position
- Velocity
- Acceleration
- Jerk
- Snap
- Crackle
- Pop

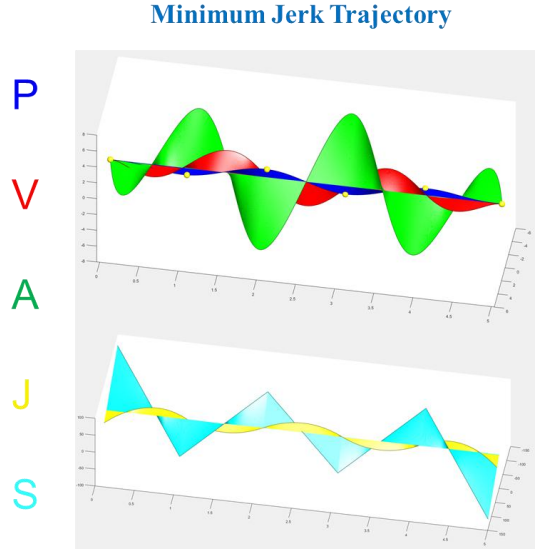
The boundary-intermediate value problem (BIVP) is often adopted by waypoint-based planners:

Our optimality conditions show the solution with **waypoint-only intermediate conditions** is
- a spline with piece-wise 5 degree polynomials
  with continuous snap when s=3 (minimum jerk trajectory)
- a spline with piece-wise 7 degree polynomials
  with continuous pop when s=4 (minimum snap trajectory)

Wang et al., Geometrically Constrained Trajectory Optimization for Multicopters, TRO 2022

Boundary-intermediate value problem (BIVP)

$$
\begin{aligned}
&\min_{z(t)} \int_{t_0}^{t_M} v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t, \\
&s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [t_0, t_M], \\
&\qquad z^{[s-1]}(t_0) = \bar{z}_o,\ z^{[s-1]}(t_M) = \bar{z}_f, \\
&\qquad z^{[d_i-1]}(t_i) = \bar{z}_i,\ 1 \le i < M, \\
&\qquad t_{i-1} < t_i,\ 1 \le i \le M.
\end{aligned}
$$



The solution of BIVP with **waypoint-only intermediate conditions** is

1. a spline with piece-wise 5 degree polynomials with continuous snap when s=3 (minimum jerk trajectory)
2. a spline with piece-wise 7 degree polynomials with continuous pop when s=4 (minimum snap trajectory)

$$\mathbf{Mc} = \mathbf{b}$$

Direct constructing the optimal solution is far easier and more efficient than implicit or explicit optimization.
The result is same as one obtained through QP formulation(Mellinger ICRA 2011 and Bry IJRR 2015).

Mellinger et al., Minimum Snap Trajectory Generation and Control for Quadrotors, ICRA 2011.
Bry et al., Aggressive flight of fixed-wing and quadrotor aircraft in dense indoor environments, IJRR 2015.

Spline trajectory generation via BIVP

Formulation:

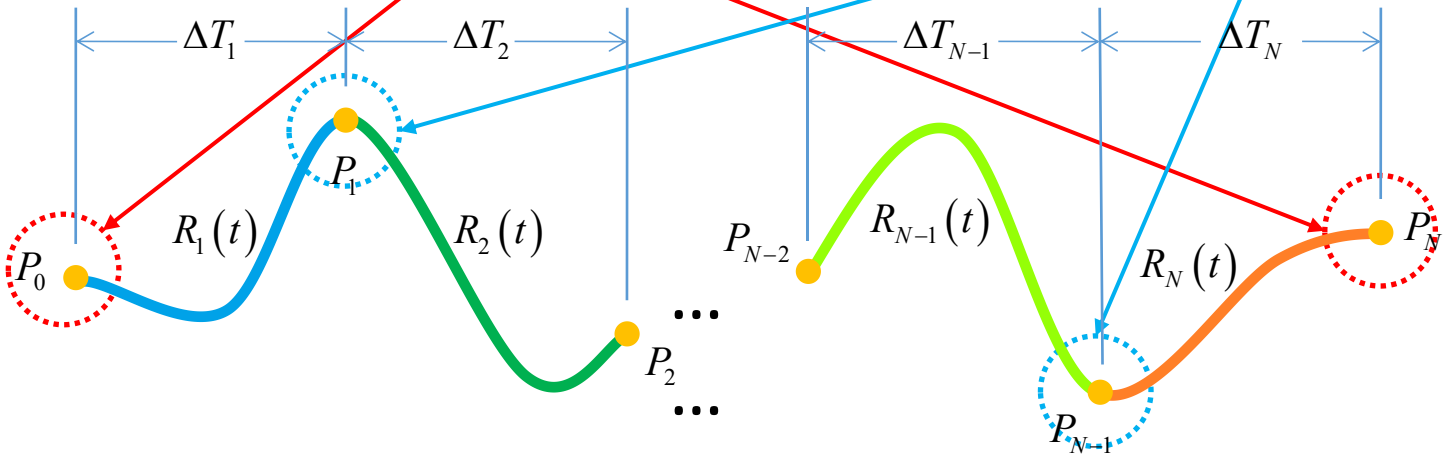
$$f(t) = \begin{cases} f_1(t) \doteq \sum_{i=0}^{N} p_{1,i} t^i & T_0 \le t \le T_1 \\ \vdots & \vdots \\ f_2(t) \doteq \sum_{i=0}^{N} p_{2,i} t^i & T_1 \le t \le T_2 \\ \vdots & \vdots \\ f_M(t) \doteq \sum_{i=0}^{N} p_{M,i} t^i & T_{M-1} \le t \le T_M \end{cases}$$

- Each segment is a polynomial.
- Fix the order as 2s-1.
- **Time durations** for each segment must be known!

Spline trajectory generation via BIVP

Linear conditions:

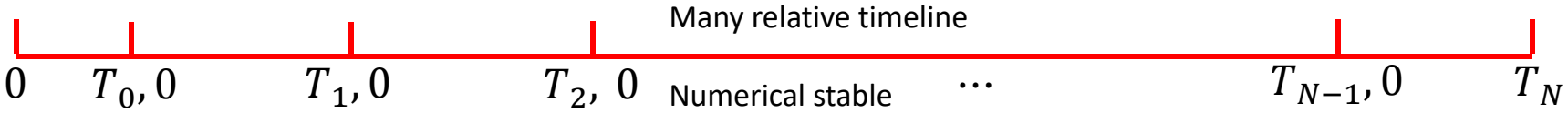
- Derivative constraints: $\begin{cases} f_j^{(k)}(T_{j-1}) = x_{0,\ j}^{(k)} \\ f_j^{(k)}(T_j) = x_{T,\ j}^{(k)} \end{cases}$
- Continuity constraints: $f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$



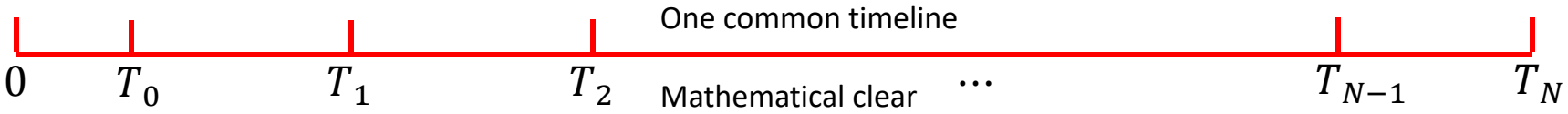
Spline trajectory generation via BIVP

Note different timeline

✓ Implementation, homework



✓ Problem formulation, lecture slides



Spline trajectory generation via BIVP

Finally you will obtain a linear equation system,

$$\begin{aligned}
&\min_{z(t)} \int_{t_0}^{t_M} v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t, \\
&s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [t_0, t_M], \\
&\qquad z^{[s-1]}(t_0) = \bar{z}_o,\ z^{[s-1]}(t_M) = \bar{z}_f, \\
&\qquad z^{[d_i-1]}(t_i) = \bar{z}_i,\ 1 \le i < M, \\
&\qquad t_{i-1} < t_i,\ 1 \le i \le M.
\end{aligned}$$

$$\Rightarrow \quad \mathbf{M}\mathbf{c} = \mathbf{b}$$

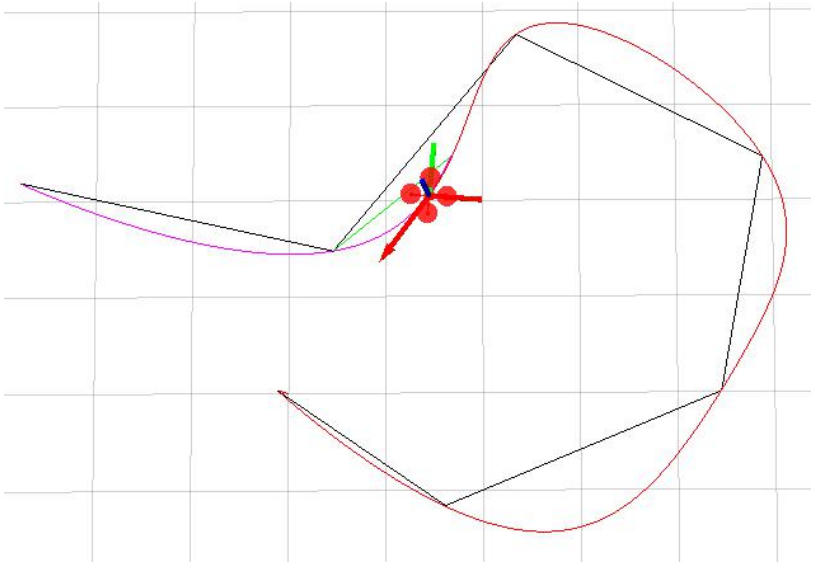
$$\mathbf{M} = \begin{pmatrix}
\mathbf{F}_0 & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{E}_1 & \mathbf{F}_1 & \mathbf{0} & \cdots & \mathbf{0} \\
\mathbf{0} & \mathbf{E}_2 & \mathbf{F}_2 & \cdots & \mathbf{0} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{F}_{M-1} \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{E}_M
\end{pmatrix}$$

$$\mathbf{b} = (\mathbf{D}_0^{\mathrm{T}}, \mathbf{D}_1^{\mathrm{T}}, \mathbf{0}_{m \times \bar{d}_1}, \ldots, \mathbf{D}_{M-1}^{\mathrm{T}}, \mathbf{0}_{m \times \bar{d}_{M-1}}, \mathbf{D}_M^{\mathrm{T}})^{\mathrm{T}}$$

Our optimality conditions guarantee M is always nonsingular as long as all piece durations are positive.
M is a banded matrix, thus solving the relevant system only costs linear time.

Spline trajectory generation via BIVP

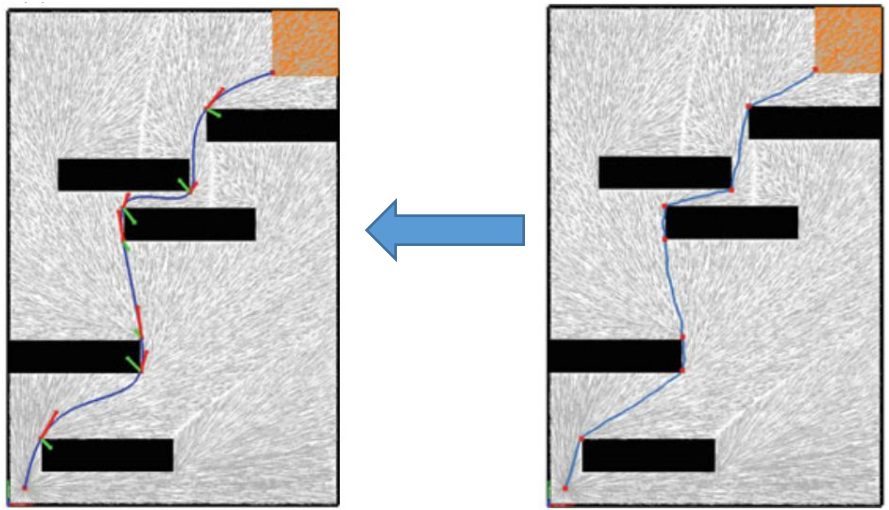
- Final trajectory



Ask: how to get these waypoints?

Hierarchical approach via BIVP

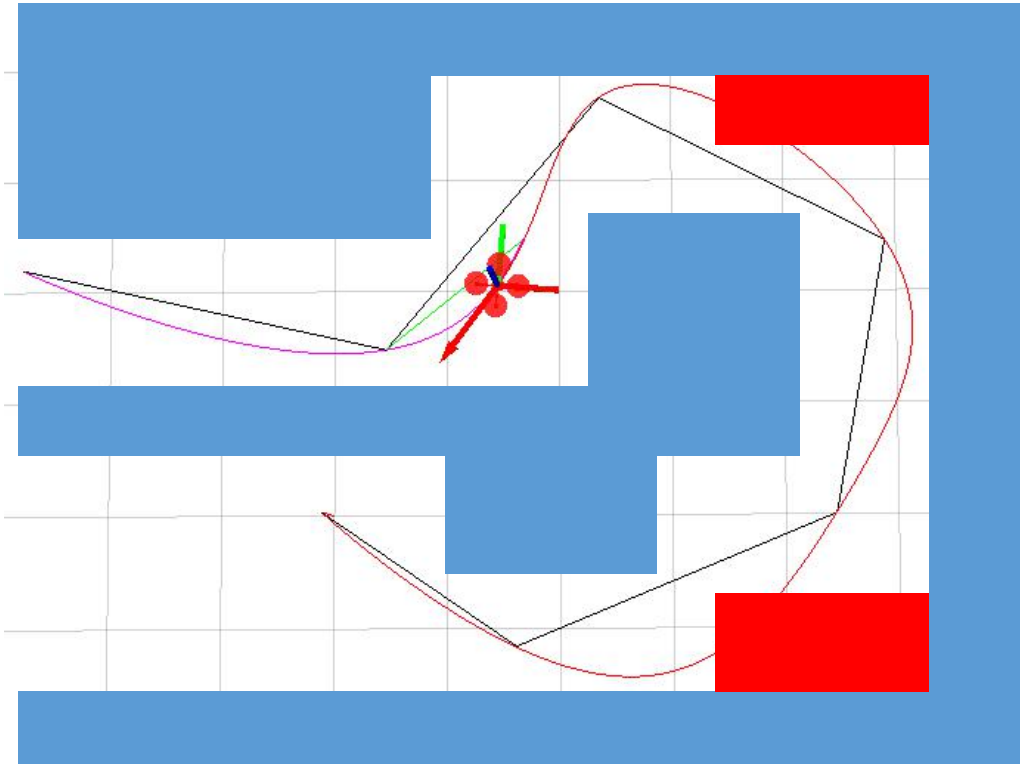
- path planning + trajectory generation
  - Low complexity solution
    - Path planning can be more efficient since it's in a much lower dimension state space.



We already know how to fit the polynomial for given waypoints

Then how to get these collision-free waypoints? → the role of path planning

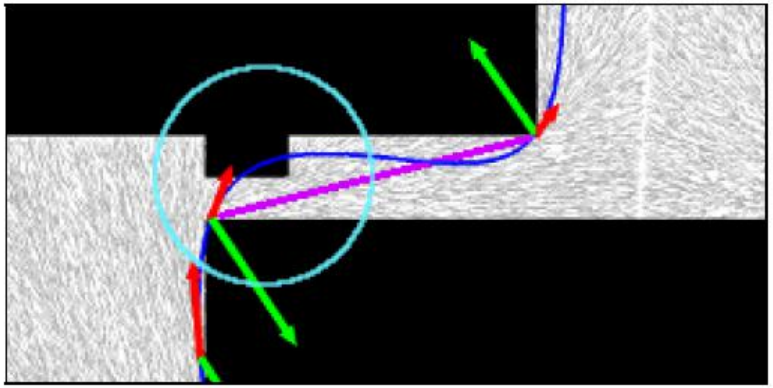
Safety issue in hierarchical approach

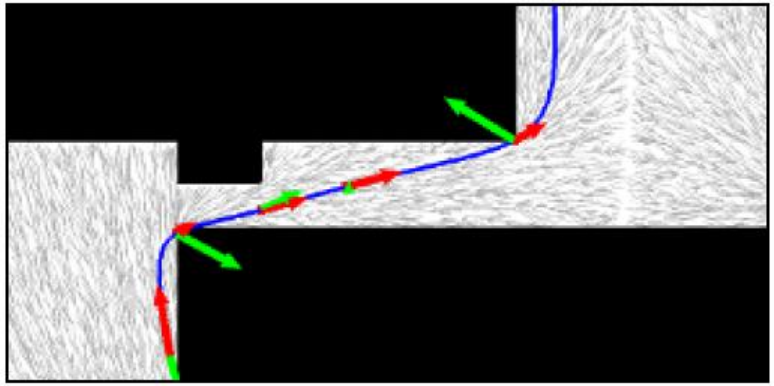


Ask: How to Ensure Collision-Free Trajectories?

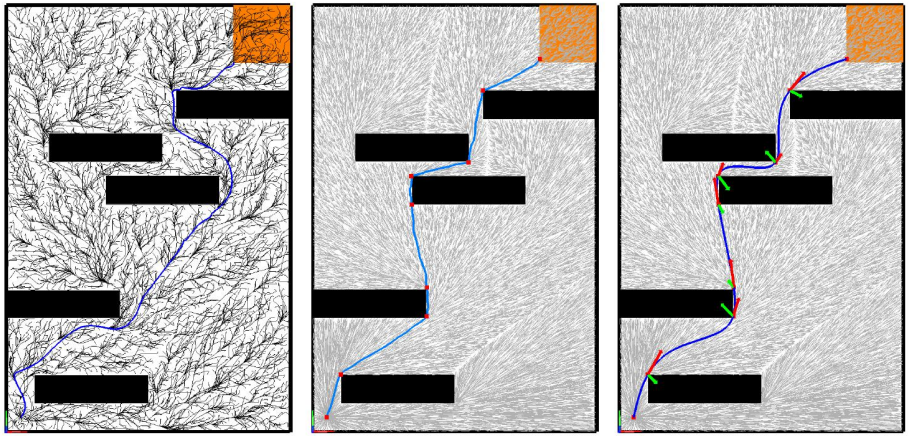
Hierarchical approach via iterative BIVPs

- The initial path is collision-free.
- We can approach the trajectory to the path iteratively.
- This is done by adding intermediate waypoints

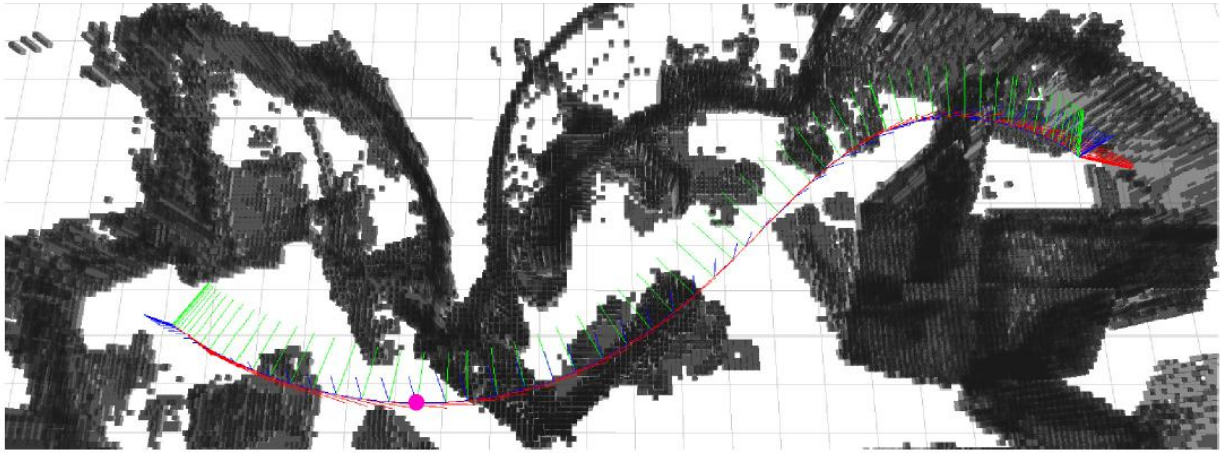




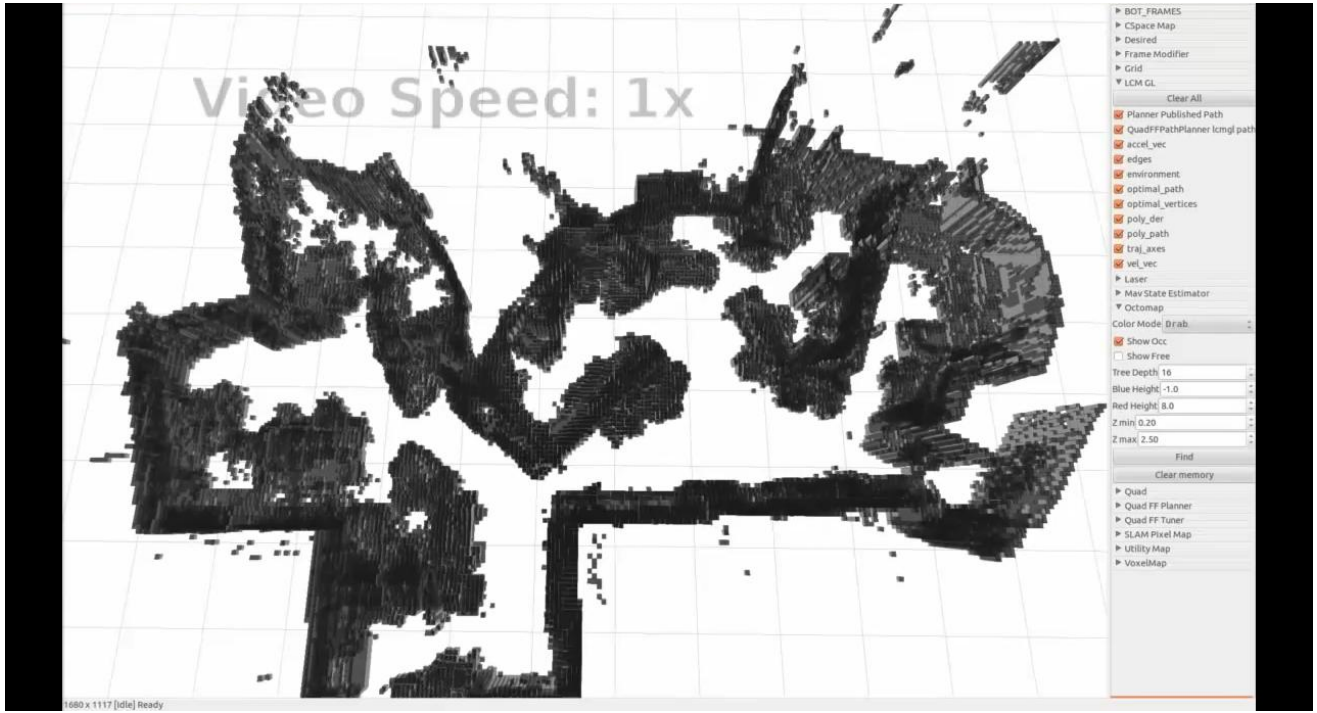
Hierarchical approach: RRT* + BIVPs



Bry IJRR 2015 adopted the unconstrained QP formulation. A much better performance can be achieved if you direct construct the optimal trajectory according to our course.



Hierarchical approach: RRT* + BIVPs



Constrained Case

Constrained Trajectory Optimization

$$
\begin{aligned}
&\min_{z(t),T} \int_0^T v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t + \rho(T), \\
&s.t.\ \ z^{(s)}(t) = v(t),\ \forall t \in [0,T], \\
&\quad \mathcal{G}(z(t), \cdots, z^{(s)}(t)) \preceq \mathbf{0},\ \forall t \in [0,T], \\
&\quad z(t) \in \mathcal{F},\ \forall t \in [0,T], \\
&\quad z^{[s-1]}(0) = \bar{z}_o,\ z^{[s-1]}(T) = \bar{z}_f, \\
&\quad z^{[s-1]} := (z^{\mathrm{T}}, \dot{z}^{\mathrm{T}}, \cdots, {z^{(s-1)}}^{\mathrm{T}})^{\mathrm{T}}.
\end{aligned}
$$

General but hard:

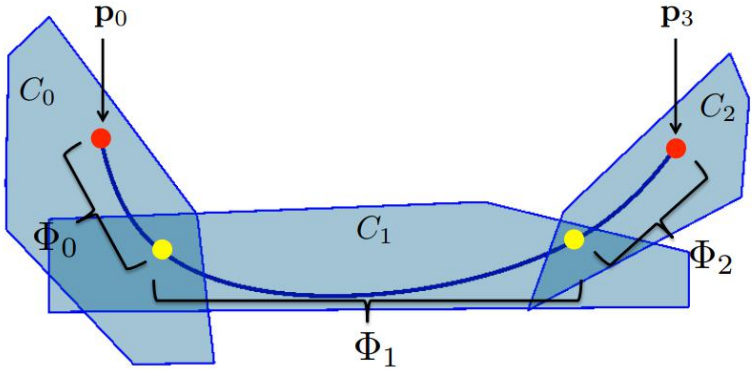
- Optimal parameterization is unknown.
- Space-time profile involves many DoFs.
- Dynamic feasibility can be non-convex.
- Safety region produces high complexity.

Possible simplification:

- Prescribe the parameterization? Polynomial spline.
- Only optimize spatial profile? Fixed time allocation.
- Simplify dynamic feasibility? Norm constraints.
- Convexify safety constraint? Locally convex region.

Constrained Case: Convex Simplification

Convex formulation



Quadratic Objective with Linear Constraints

$$
\begin{aligned}
\arg\min_{\Phi} \; J^q &= \sum_{i=0}^{n-1} \int_0^{\Delta t_i} ||\Phi_i^{(q)}(t)||^2 dt \\
s.t \;\; \Phi_i^{(k)}(\Delta t_i) &= \Phi_{i+1}^{(k)}(0), \quad k = 0 \dots q \text{ and } i = 1 \dots N-2, \\
\Phi_0^{(k)}(0) &= u_s^{(k)}, \; \Phi_{N-1}^{(k)}(\Delta t_{N-1}) = u_g^{(k)}, \quad k = 0 \dots q, \\
\mathbf{A}_i^T \Phi_i(t) &\leq \mathbf{b}_i, \quad i = 0 \dots N-1.
\end{aligned}
$$

Liu et al., Planning Dynamically Feasible Trajectories for Quadrotors Using Safe Flight Corridors in 3-D Complex Environments, RAL 2017

Cost functional

- Cost function for one polynomial segment:

$$f(t) = \sum_i p_i t^i$$

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

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

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

$$\Rightarrow J(T) = \int_{T_{j-1}}^{T_j} \left(f^4(t)\right)^2 dt$$

$$= \begin{bmatrix} \vdots \\ p_i \\ \vdots \end{bmatrix}^T \begin{bmatrix} \cdots & \vdots & \cdots \\ \cdots & \frac{i(i-1)(i-2)(i-3)l(l-1)(l-2)(l-3)}{i+l-7}T^{i+l-7} & \cdots \\ \cdots & \vdots & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_l \\ \vdots \end{bmatrix}$$

$$\Rightarrow J_j(T) = \mathbf{p}_j^T \mathbf{Q}_j \mathbf{p}_j$$

Minimize this!

Equality constraints

- Derivative constraint for one polynomial segment
  - Also models waypoint constraint (0^{th} order derivative)

$$f_j^{(k)}(T_j) = x_j^{(k)}$$
$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} = x_{T,j}^{(k)}$$
$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} = x_{T,j}^{(k)}$$
$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_{j-1}^{i-k} & \cdots \\ \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \end{bmatrix} = \begin{bmatrix} x_{0,j}^{(k)} \\ x_{T,j}^{(k)} \end{bmatrix}$$
$$\Rightarrow \mathbf{A}_j \mathbf{p}_j = \mathbf{d}_j$$

$$x(t) = p_5 t^5 + p_4 t^4 + p_3 t^3 + p_2 t^2 + p_1 t + p_0$$

$$x(0) = \cdots,\ x(T) = \cdots$$
$$\dot{x}(0) = \cdots, \dot{x}(T) = \cdots$$
$$\ddot{x}(0) = \cdots, \ddot{x}(T) = \cdots$$
$$\vdots$$

$$p_0 = \cdots,\ p_5 T^5 + p_4 T^4 + p_3 T^3 + p_2 T^2 + p_1 T + p_0 = \cdots$$

$$[T^5,\ T^4, T^3, T^2, T, 1] \begin{bmatrix} p_5 \\ p_4 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \cdots$$

Equality constraints

- Continuity constraint between two segments:
  - Ensures continuity between trajectory segments when no specific derivatives are given

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

$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} - \sum_{l \geq k} \frac{l!}{(l-k)!} T_j^{l-k} p_{j+1,l} = 0$$

$$\Rightarrow \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots & -\frac{l!}{(l-k)!} T_j^{l-k} & \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \\ p_{j+1,l} \\ \vdots \end{bmatrix} = 0$$

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

Convex function and convex set

Convex function

- A function f: $R^n \longrightarrow R$ is said to be **convex** if the domain, dom f, is convex and for any $x, y \in$ dom f and $0 \leq \theta \leq 1$,

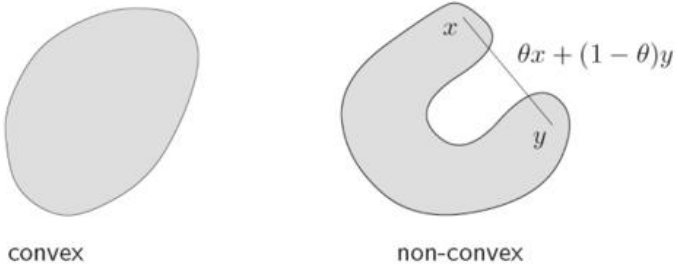
$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta) f(y)$$

- f is strictly convex if the inequality is strict for $0 < \theta < 1$.
- f is concave if $-f$ is convex.

Convex set

- A set $y \in R^n$ is said to be **convex** if the line segment between any two points is in the set: for any $x, y \in C$ and $0 \leq \theta \leq 1$,

$$\theta x + (1-\theta)y \in C.$$



Courtesy: *Convex Optimization*, Daniel Palomar, HKUST

Convex optimization

- Optimization problem in standard form

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & h_i(x) = 0 \qquad i = 1, \ldots, p \end{aligned}$$

$x \in R^n$ is the optimization variable

$f_0 \colon R^n \to R$ is the objective function

$f_i \colon R^n \to R,\ i = 1, \ldots, m$ are inequality constraint functions

$h_i \colon R^n \to R,\ i = 1, \ldots, p$ are equality constraint functions

- Convex optimization problem in standard form:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \leq 0 \qquad i = 1, \ldots, m \\ & Ax = b \end{aligned}$$

where $f_0, f_1, \ldots, f_m$ are convex and equality constraints are affine.

- **Local and goal optima**: any locally optimal point of a convex problem is globally optimal.
- Most problems are not convex when formulated.
- Reformulating a problem in convex form is an art, there is no systematic way.

Disciplined convex optimization programs

Linear Programming (LP)

$$\begin{array}{ll} \underset{x}{\text{minimize}} & c^T x + d \\ \text{subject to} & Gx \leq h \\ & Ax = b \end{array}$$

- Convex problem: affine objective and constraint functions.

Quadratic Programming (QP)

$$\begin{array}{ll} \underset{x}{\text{minimize}} & (1/2)x^T P x + q^T x + r \\ \text{subject to} & Gx \leq h \\ & Ax = b \end{array}$$

- Convex problem (assuming $P \in S^n \geq 0$): convex quadratic objective and affine constraint functions.

Quadratically Constrained QP (QCQP)

$$\begin{array}{ll} \underset{x}{\text{minimize}} & (1/2)x^T P_0 x + q_0^T x + r_0 \\ \text{subject to} & (1/2)x^T P_i x + q_i^T x + r_i \leq 0 \quad i = 1, \ldots, m \\ & Ax = b \end{array}$$

- Convex problem (assuming $P_i \in S^n \geq 0$): convex quadratic objective and constraint functions.

Second-Order Cone Programming (SOCP)

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f^T x \\ \text{subject to} & \|A_i x + b_i\| \leq c_i^T x + d_i \quad i = 1, \ldots, m \\ & Fx = g \end{array}$$

- Convex problem: linear objective and second-order cone constraints
- For A_i row vector, it reduces to an LP.
- For $c_i = 0$, it reduces to a QCQP.
- More general than QCQP and LP.

Solve a convex trajectory generation program

- Your target is to formulate a trajectory generation problem into **Disciplined convex optimization programs** in P.41.
- Many off-the-shelf can help you solve them.
- Choose a proper solver according to your requirement.

CVX

http://cvxr.com/cvx/

Matlab wrapper. Let you write down the convex program like mathematical equations, then call other solvers to solve the problem.

Mosek

https://www.mosek.com/

Very robust convex solvers, can solve almost all kinds of convex programs. Can apply free academic license. Only library available (x86).

OOQP

http://pages.cs.wisc.edu/~swright/ooqp/

Very fast, robust QP solver. Open sourced.

GLPK

https://www.gnu.org/software/glpk/

Very fast, robust LP solver. Open sourced.

Normalization

- Time normalization
  - Some very small time durations may break the generation entirely.
  - Scale short time durations to a normal number (1.0).
  - Or adding scale factor to all piece of the curve.

$$f(t) = \begin{cases} \sum_{i=0}^{N} p_{1,i}\left(\frac{t - T_0}{T_1 - T_0}\right)^i & T_0 \le t \le T_1 \\ \sum_{i=0}^{N} p_{2,i}\left(\frac{t - T_1}{T_2 - T_1}\right)^i & T_1 \le t \le T_2 \\ \vdots & \vdots \\ \sum_{i=0}^{N} p_{M,i}\left(\frac{t - T_{M-1}}{T_M - T_{M-1}}\right)^i & T_{M-1} \le t \le T_M \end{cases}$$

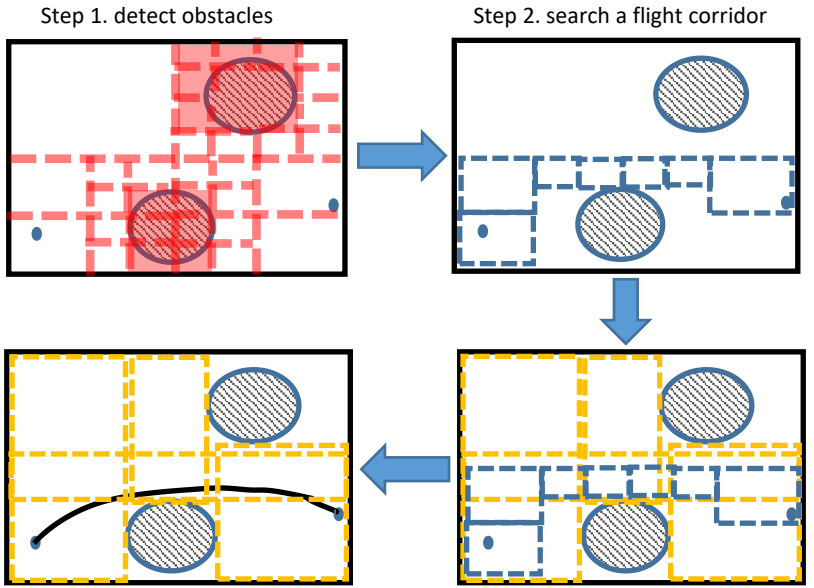
Use relative timeline!

- Problem scale (spatial) normalization
  - If the problem is underlying for large-scale scene.
  - Such as waypoints with $x = 100.0\ m$
  - Consider solve a tiny problem (a sandbox), and re-scale the solution back.

These two operations highly increase the numerical stability in practice.

Experimental validation

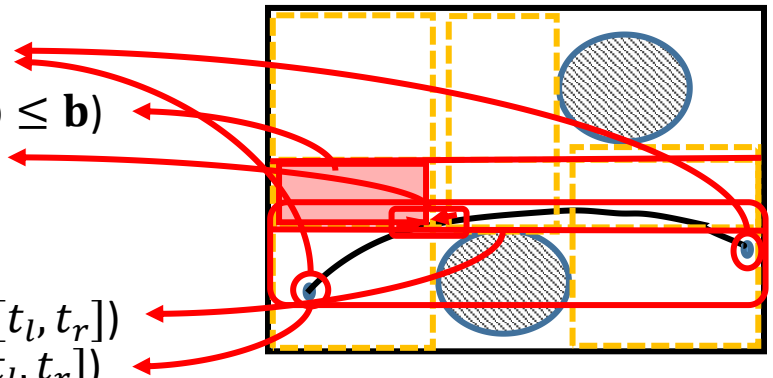
Trajectory with Guaranteed Obstacle Avoidance



Trajectory with Guaranteed Obstacle Avoidance

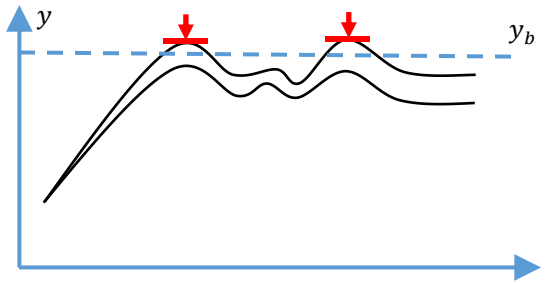
- Instant linear constraints:
  - Start, goal state constraint ($\mathbf{Ap} = \mathbf{b}$)
  - Transition point constraint ($\mathbf{Ap} = \mathbf{b}, \mathbf{Ap} \leq \mathbf{b}$)
  - Continuity constraint($\mathbf{Ap_i} = \mathbf{Ap_{i+1}}$)

- Interval linear constraints:
  - Boundary constraint ($\mathbf{A}(t)\mathbf{p} \leq \mathbf{b}, \forall t \in [t_l, t_r]$)
  - Dynamic constraint ($\mathbf{A}(t)\mathbf{p} \leq \mathbf{b}, \forall t \in [t_l, t_r]$)
    - Velocity constraints
    - Acceleration constraints



How to make constraints globally activated

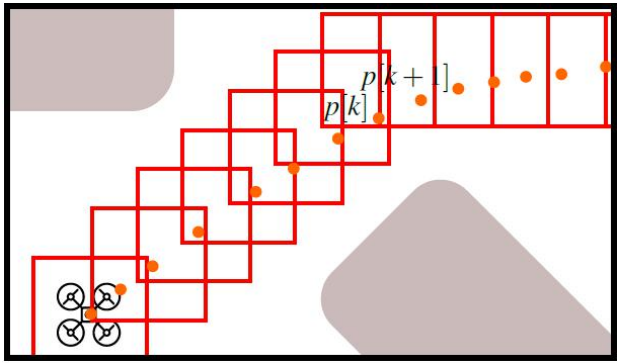
- Iteratively check extremum and add extra constraints.



- Iterative solving is time consuming.
- If strictly no feasible solution meets all constraints. We have to run 10 iterations to determine the status of the solution ?

Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments, J. Chen, **ICRA** 2016

- Adding numerous constraints at discrete time ticks.



- Always generates over-conservative trajectories.
- Too many constraints, the computational burden is high.

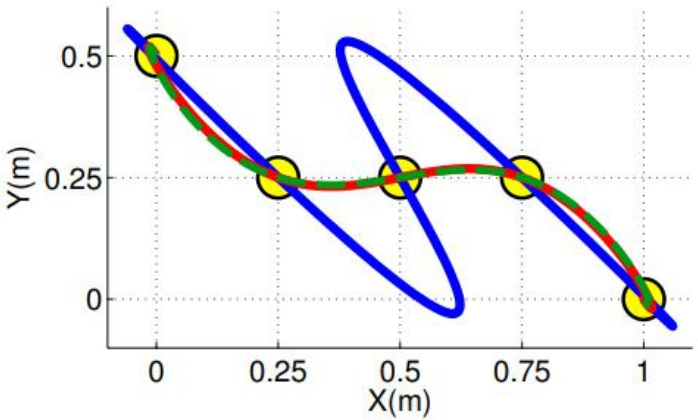
A hybrid method for online trajectory planning of mobile robots in cluttered environments, L Campos-Macías, **RAL** 2017

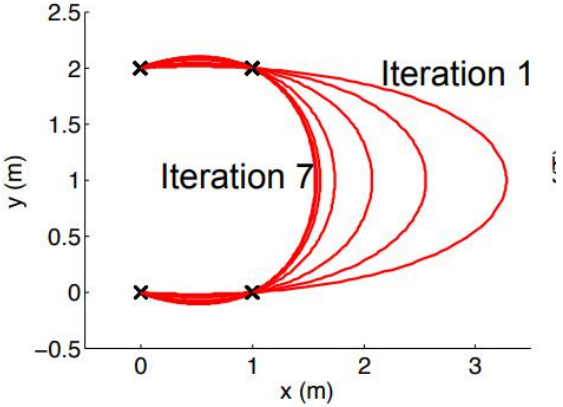
Solution 1

Solution 2

Issues of simplification

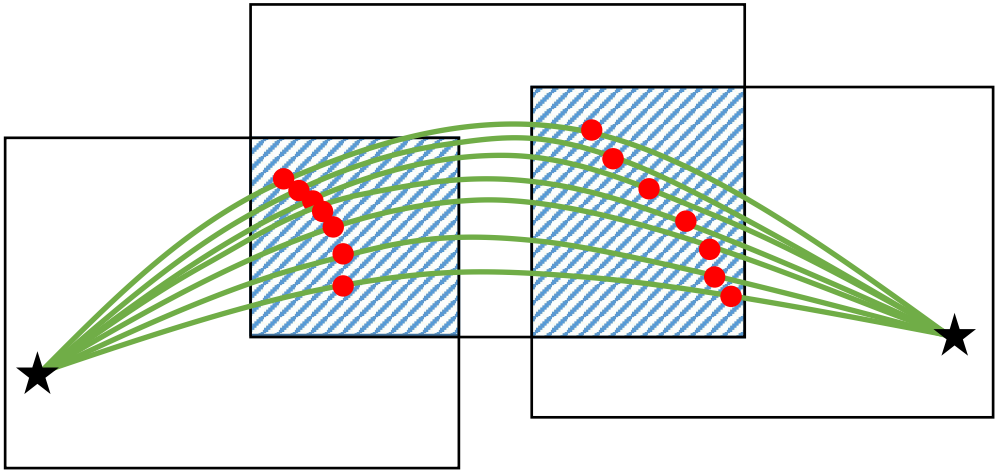
- Piecewise trajectory depends on a piecewise time allocation.
- Time allocation significantly affect the final trajectory.
- How to get a proper time allocation?

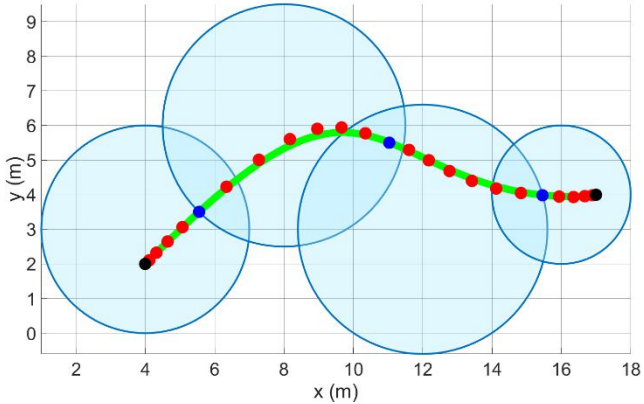




Issues of simplification

- Looks stupid for one piece.
- Works fine in corridor based trajectory generation.
- Overlapping regions in corridor provides large auto-adjustment space.

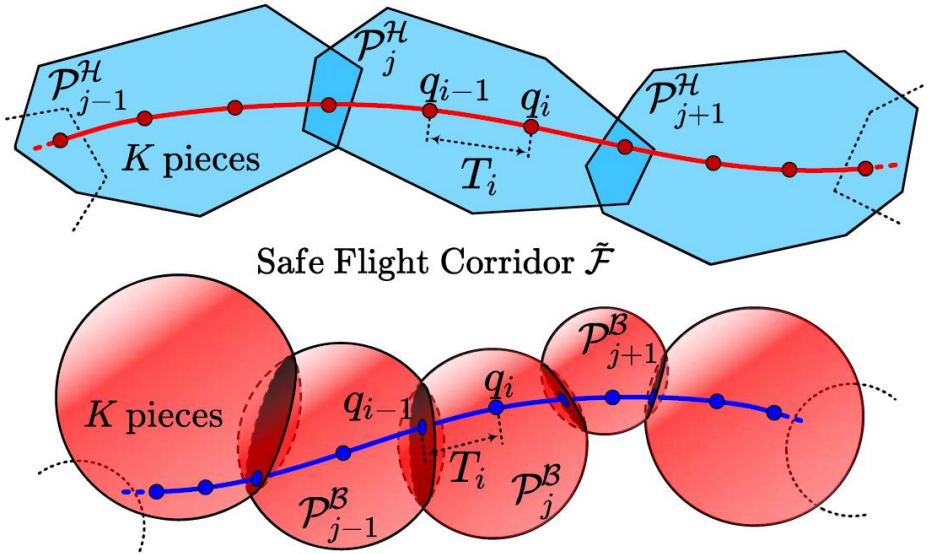




Constrained Case: Spatial-Temporal Deformation (Brief)

Spatial-temporal deformation

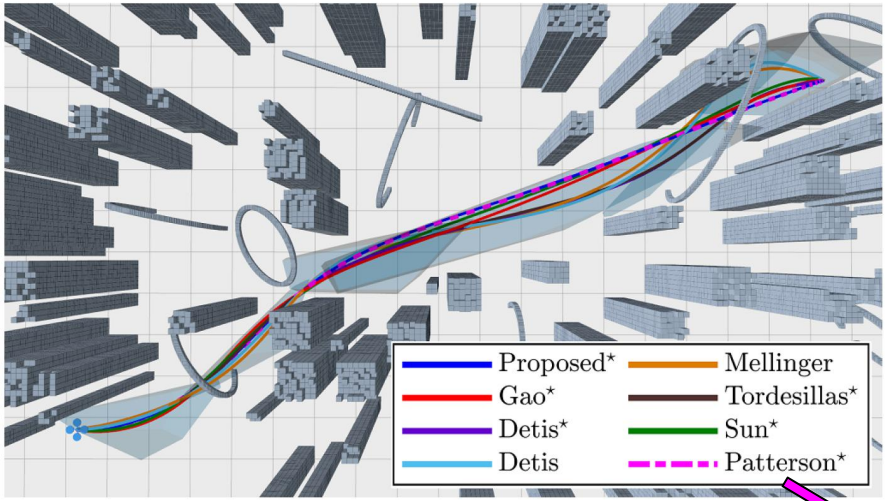
$$
\begin{aligned}
\min_{z(t),T} \quad & \int_0^T v(t)^{\mathrm{T}} \mathbf{W} v(t) \mathrm{d}t + \rho(T), \\
s.t. \quad & v(t) = z^{(s)}(t), \ \forall t \in [0, T], \\
& \mathcal{G}(z(t), \ldots, z^{(s)}(t)) \preceq \mathbf{0}, \ \forall t \in [0, T], \\
& z(t) \in \mathcal{F}, \ \forall t \in [0, T], \\
& z^{[s-1]}(0) = \bar{z}_o, \ z^{[s-1]}(T) = \bar{z}_f, \\
& z^{[s-1]} = (z^{\mathrm{T}}, \dot{z}^{\mathrm{T}}, \ldots, z^{(s-1)^{\mathrm{T}}})^{\mathrm{T}}
\end{aligned}
$$

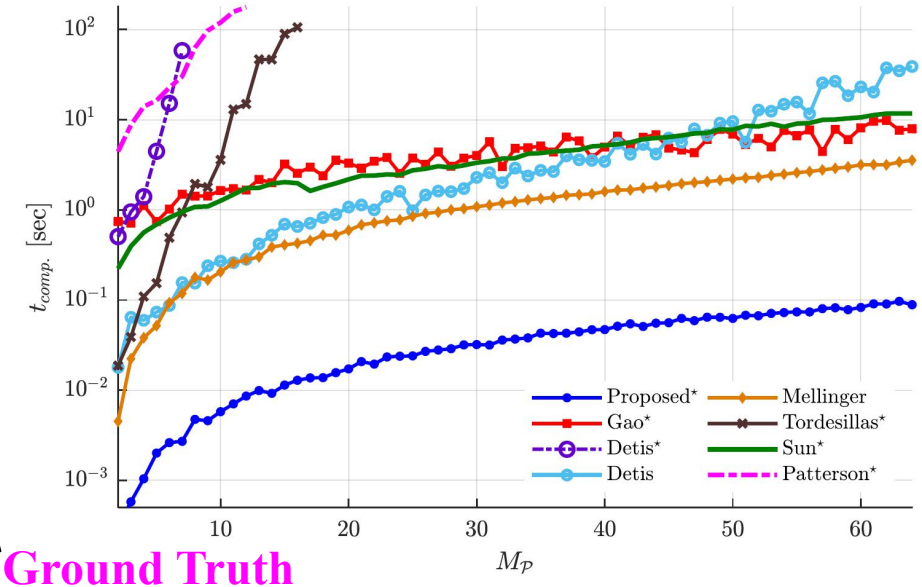


https://github.com/ZJU-FAST-Lab/GCOPTER

Performance

How good is the spatial-temporal deformation?
Benchmarks with 7 classical or SOTA methods!





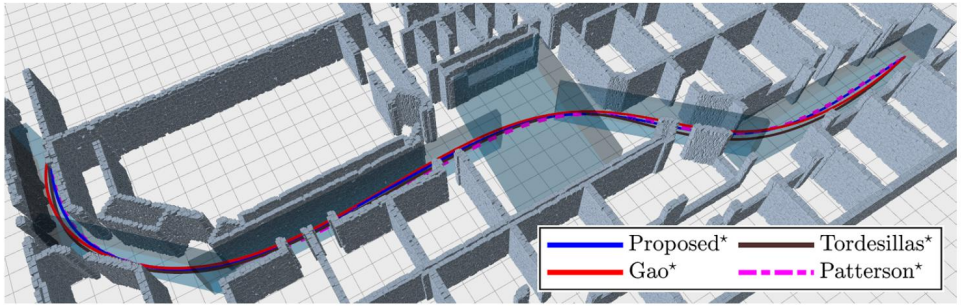
Ground Truth

Gao F, Wang L, Zhou B, Zhou X, Pan J and Shen S (2020) Teach-Repeat-Replan: A complete and robust system for aggressive flight in complex environments. IEEE Transactions on Robotics 36(5): 1526–1545.
Deits R and Tedrake R (2015b) Efficient mixed-integer planning for UAVs in cluttered environments. In: IEEE International Conference on Robotics and Automation. Seattle, USA, pp. 42–49.
Mellinger D and Kumar V (2011) Minimum snap trajectory generation and control for quadrotors. In: IEEE International Conference on Robotics and Automation. Shanghai, China, pp. 2520–2525.
Tordesillas J, Lopez BT and How JP (2019) FASTER: Fast and safe trajectory planner for flights in unknown environments. In: IEEE International Conference on Intelligent Robots and Systems. Macau, China, pp. 1934–1940.
Sun W, Tang G and Hauser K (2020) Fast UAV trajectory optimization using bilevel optimization with analytical gradients. In: American Control Conference. pp. 82–87.
Patterson MA and Rao AV (2014) GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hpadaptive gaussian quadrature collocation methods and sparse nonlinear programming. ACM Transactions on Mathematical Software 41(1): 1–37.

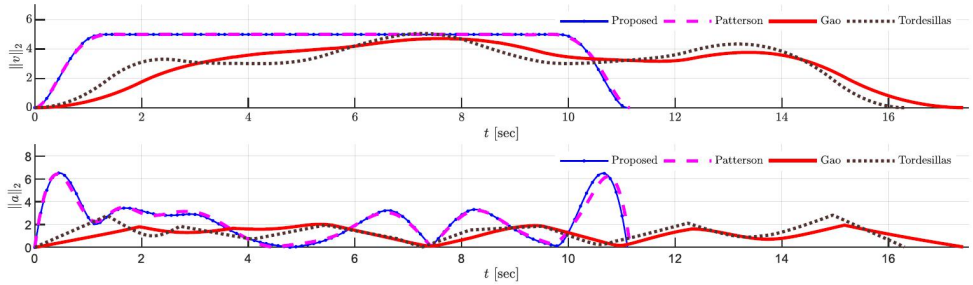
https://github.com/ZJU-FAST-Lab/GCOPTER

Performance

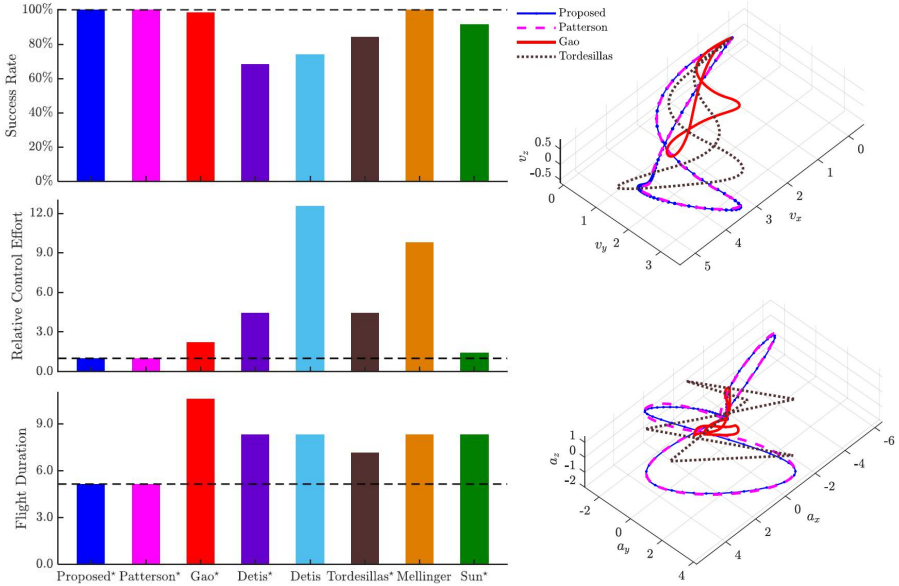
How good is the spatial-temporal deformation?



(a) Trajectories generated by different methods within a long SFC of the office environment



(b) The velocity and acceleration magnitude for different methods.

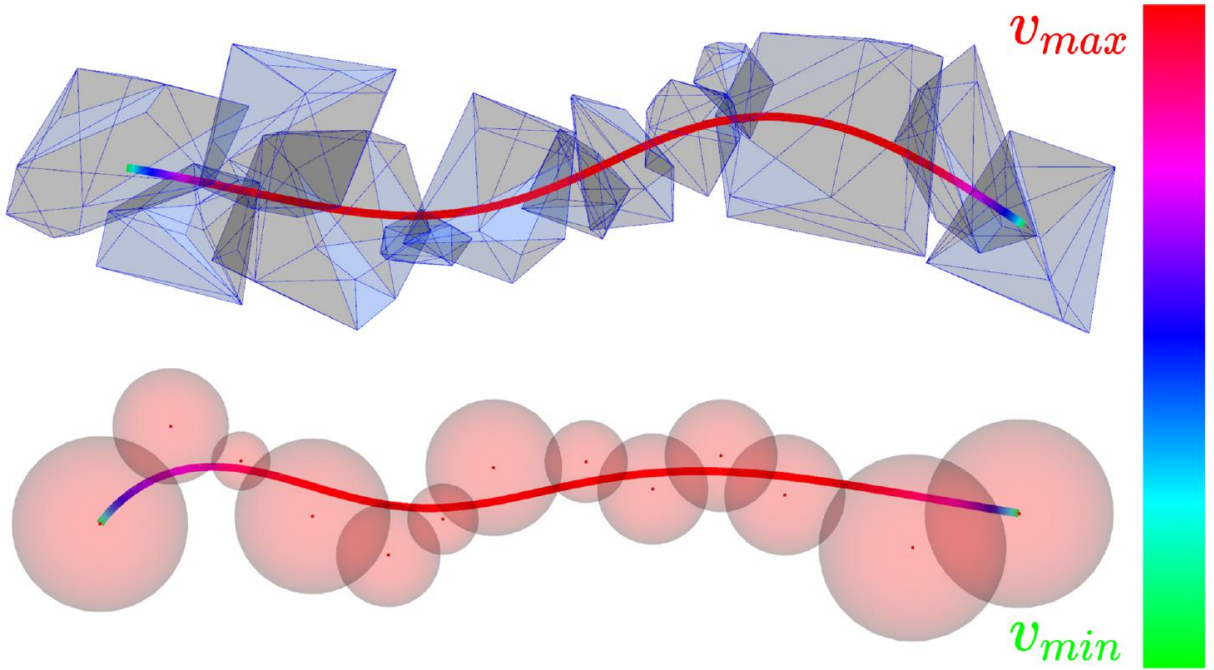


Gao F, Wang L, Zhou B, Zhou X, Pan J and Shen S (2020) Teach-Repeat-Replan: A complete and robust system for aggressive flight in complex environments. IEEE Transactions on Robotics 36(5): 1526–1545.
Deits R and Tedrake R (2015b) Efficient mixed-integer planning for UAVs in cluttered environments. In: IEEE International Conference on Robotics and Automation. Seattle, USA, pp. 42–49.
Mellinger D and Kumar V (2011) Minimum snap trajectory generation and control for quadrotors. In: IEEE International Conference on Robotics and Automation. Shanghai, China, pp. 2520–2525.
Tordesillas J, Lopez BT and How JP (2019) FASTER: Fast and safe trajectory planner for flights in unknown environments. In: IEEE International Conference on Intelligent Robots and Systems. Macau, China, pp. 1934–1940.
Sun W, Tang G and Hauser K (2020) Fast UAV trajectory optimization using bilevel optimization with analytical gradients. In: American Control Conference. pp. 82–87.
Patterson MA and Rao AV (2014) GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hpadaptive gaussian quadrature collocation methods and sparse nonlinear programming. ACM Transactions on Mathematical Software 41(1): 1–37.

https://github.com/ZJU-FAST-Lab/GCOPTER

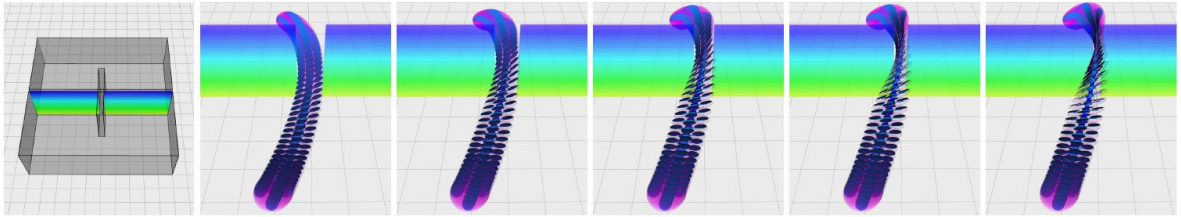
Performance

How good is the spatial-temporal deformation?

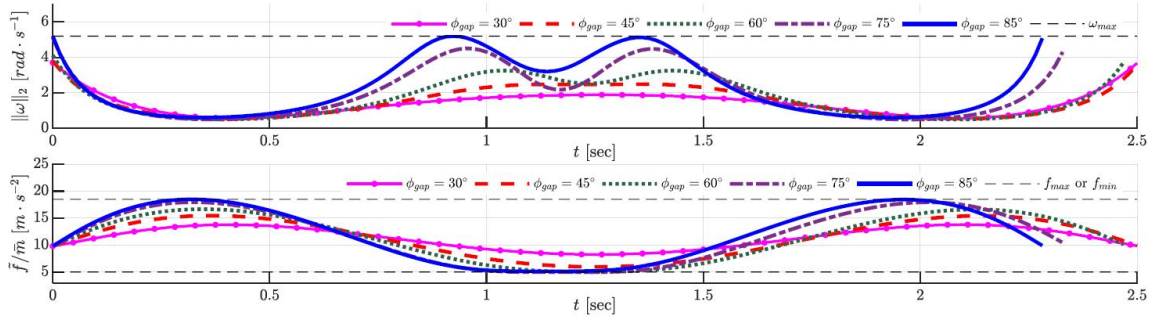


Performance

How good is the spatial-temporal deformation?



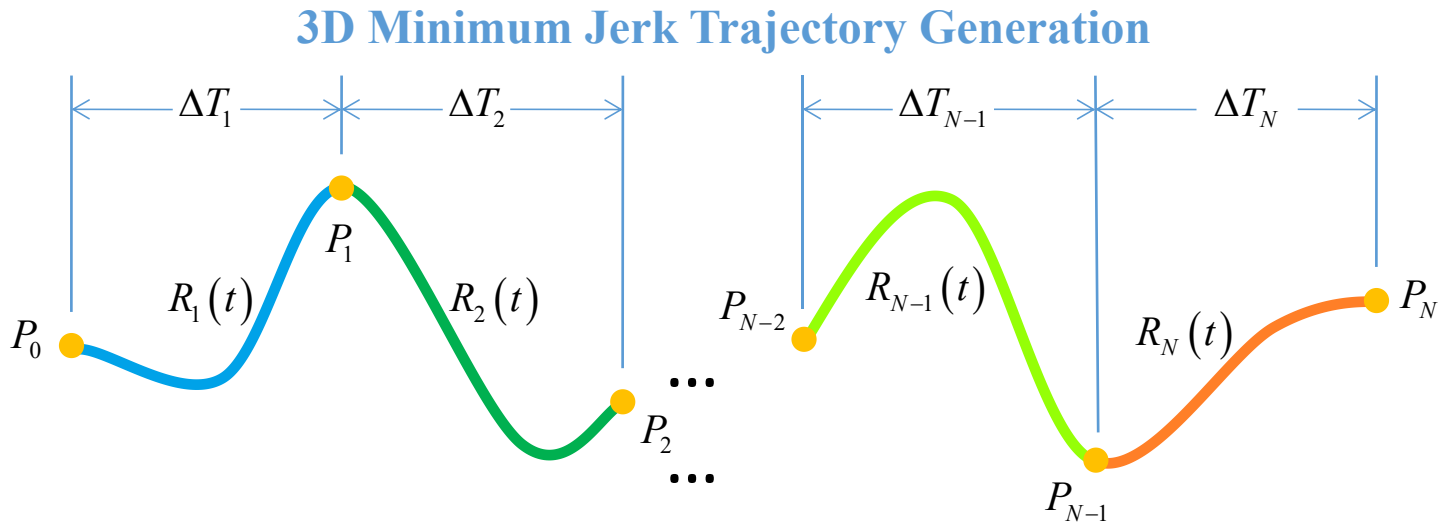
(a) SFC layout (b) $\phi_{gap} = 30°$ (c) $\phi_{gap} = 45°$ (d) $\phi_{gap} = 60°$ (e) $\phi_{gap} = 75°$ (f) $\phi_{gap} = 85°$



(g) The magnitude of angular velocity and the normalized thrust for different SE(3) trajectories.

d_{gap}	$0.88m$	$0.76m$	$0.60m$	$0.40m$	$0.25m$
ϕ_{gap}	$30°$	$45°$	$60°$	$75°$	$85°$
$t_{comp.}$	$4.7ms$	$4.4ms$	$6.0ms$	$6.6ms$	$7.4ms$

Homework



Input:

3-d pos, vel, and acc at start and terminal stamp;
(M-1) 3-d waypoint pos;
M durations for each trajectory pieces.

Output:

Coefficients of 3-d minimum jerk trajectory.

Requirement:

Use optimality conditions.

Theorem (Optimality Conditions). *A trajectory, denoted by* $z^*(t) : [t_0, t_M] \mapsto \mathbb{R}^m$, *is optimal, if and only if the following conditions are satisfied:*

- *The map* $z^*(t) : [t_{i-1}, t_i] \mapsto \mathbb{R}^m$ *is parameterized as a* $2s-1$ *degree polynomial for any* $1 \leq i \leq M$;
- *The boundary and intermediate conditions all hold;*
- $z^*(t)$ *is* $2s - d_i - 1$ *times continuously differentiable at* t_i *for any* $1 \leq i < M$.

Moreover, a unique trajectory exists for these conditions.

Thanks for Listening!