香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY
電子及計算機工程學系 DEPARTMENT OF ELECTRONIC & COMPUTER ENGINEERING

Introduction to Aerial Robotics
Lecture 4

Shaojie Shen

Assistant Professor

Dept. of ECE, HKUST

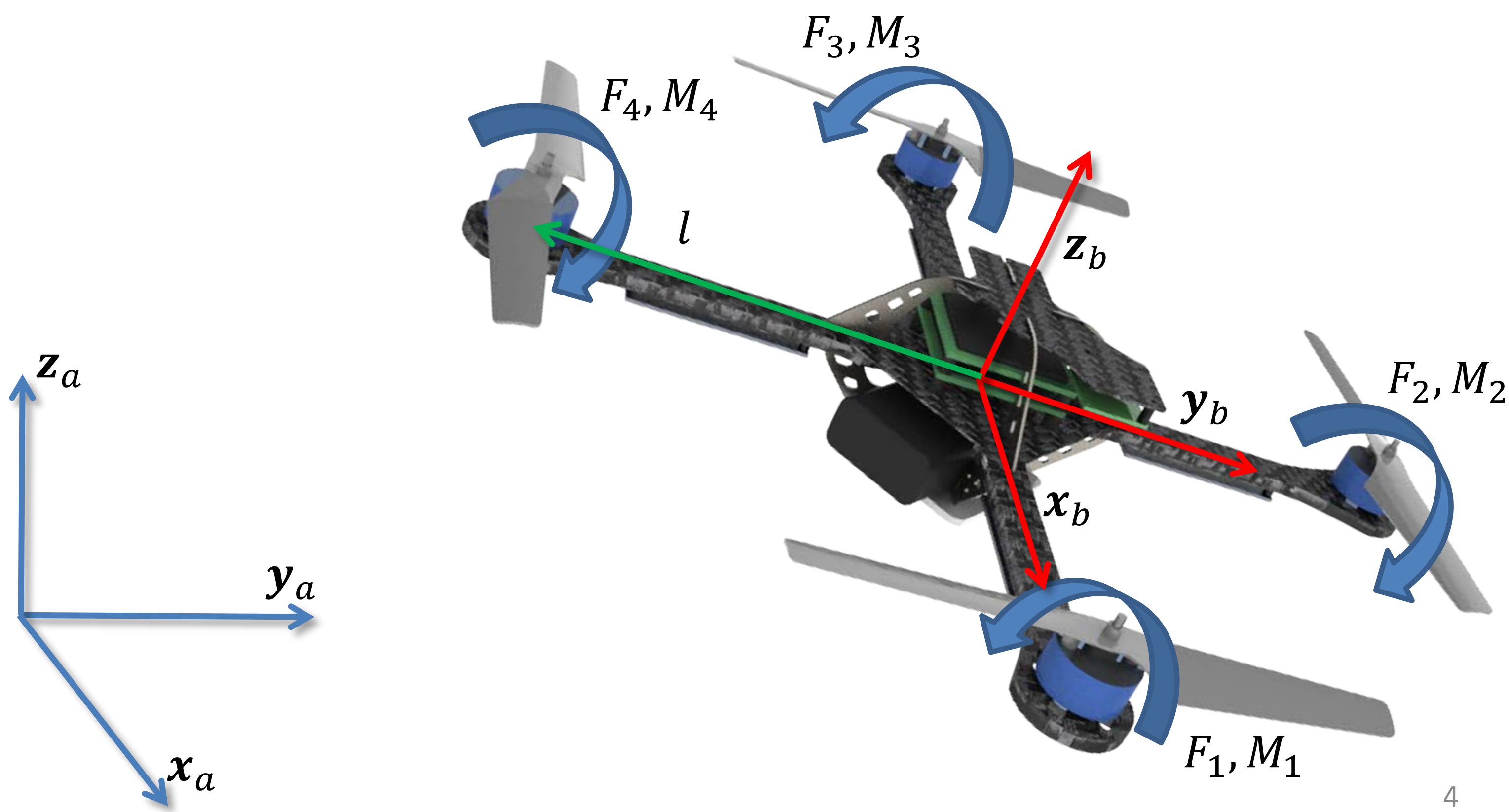
29 September 2015

Outline

- Review: Quadrotor Control
- Trajectory Generation

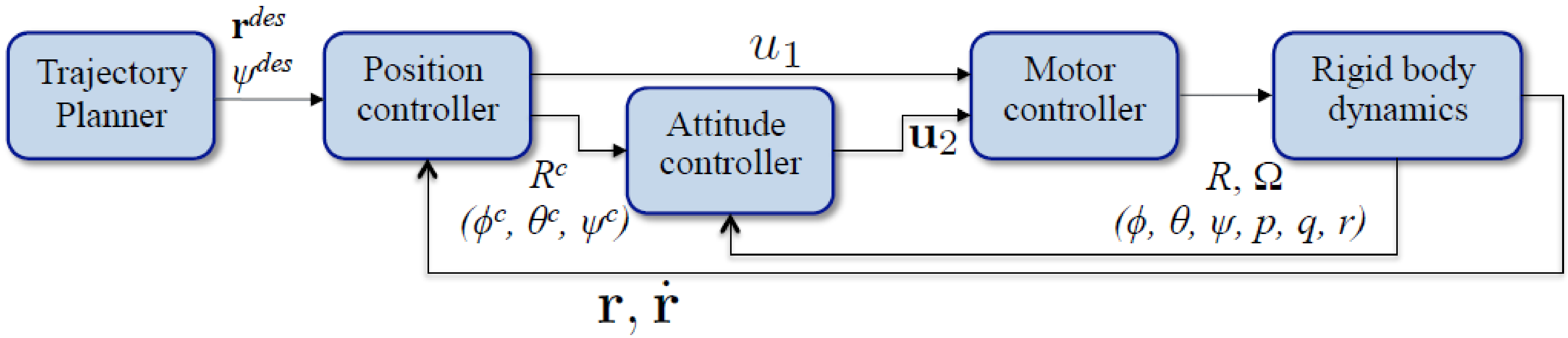
Review: Quadrotor Control

Quadrotor Dynamics



Quadrotor Dynamics

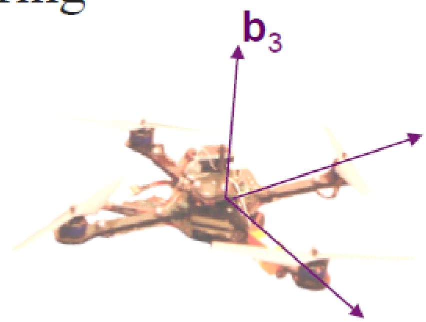
- Motor model: $\dot{\omega}_i = k_m(\omega_i^{des} - \omega_i)$
- Thrust from individual motor: $F_i = k_F \omega_i^2$
- Moment from individual motor: $M_i = k_M \omega_i^2$
- Newton Equation: $m\ddot{\boldsymbol{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$
- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$



$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \underbrace{\begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}}_{u_1}$$

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \underbrace{\begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}}_{\mathbf{u}_2} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix}$$

Control for Hovering



$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \boxed{F_1 + F_2 + F_3 + F_4} \end{bmatrix}$$

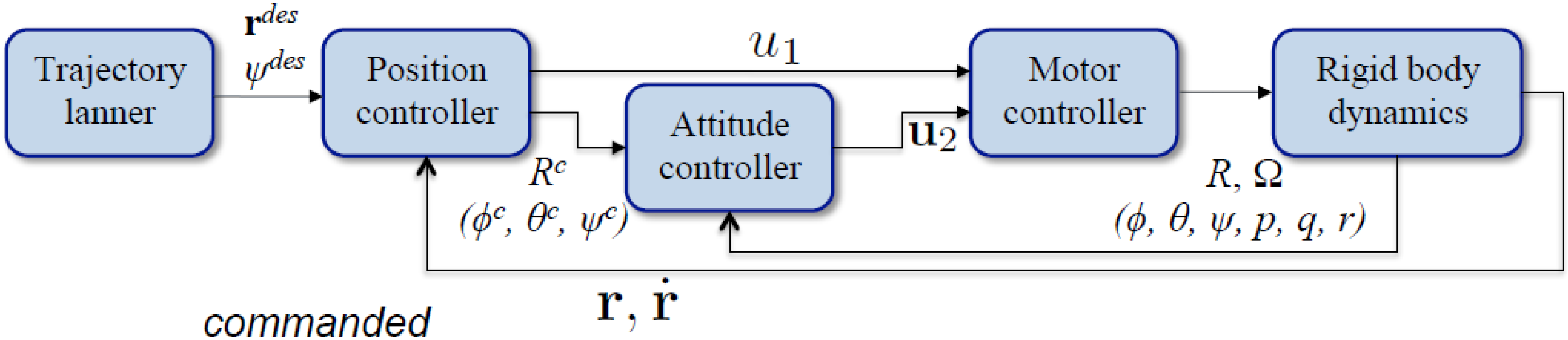
u_1

Linearization

$$(u_1 \sim mg, \theta \sim 0, \phi \sim 0, \psi \sim \psi_0)$$

$$\ddot{r}_1 = \ddot{x} = g(\theta \cos\psi + \phi \sin\psi)$$

$$\ddot{r}_2 = \ddot{y} = g(\theta \sin\psi - \phi \cos\psi)$$



commanded

actual (feedback)

$$(\ddot{r}_{i,des} - \ddot{r}_{i,c}) + k_{d,i}(\dot{r}_{i,des} - \dot{r}_i) + k_{p,i}(r_{i,des} - r_i) = 0$$

specified

$$u_1 = m(g + \ddot{r}_{3,c})$$

$$\phi_c = \frac{1}{g}(\ddot{r}_{1,c} \sin\psi_{des} - \ddot{r}_{2,c}\cos\psi_{des})$$

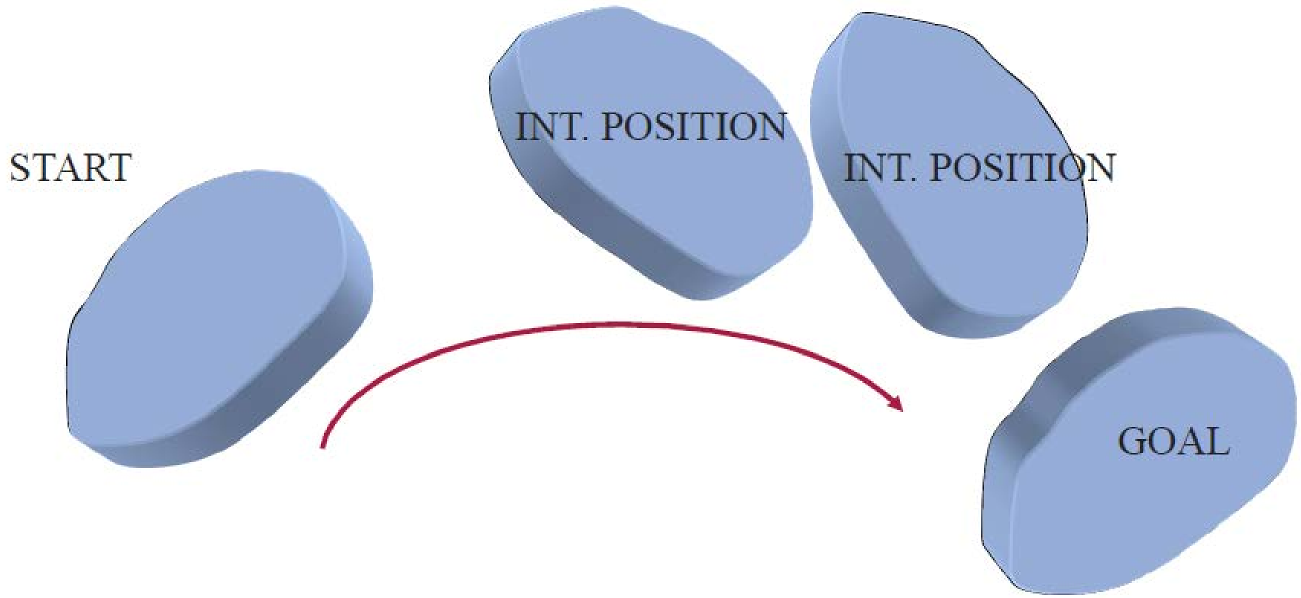
$$\theta_c = \frac{1}{g}(\ddot{r}_{1,c} \cos\psi_{des} + \ddot{r}_{2,c}\sin\psi_{des})$$

$$\mathbf{u}_2 = \begin{bmatrix} k_{p,\phi}(\phi_c - \phi) + k_{d,\phi}(p_c - p) \\ k_{p,\theta}(\theta_c - \theta) + k_{d,\theta}(q_c - q) \\ k_{p,\psi}(\psi_c - \psi) + k_{d,\psi}(r_c - r) \end{bmatrix}$$

Trajectory Generation

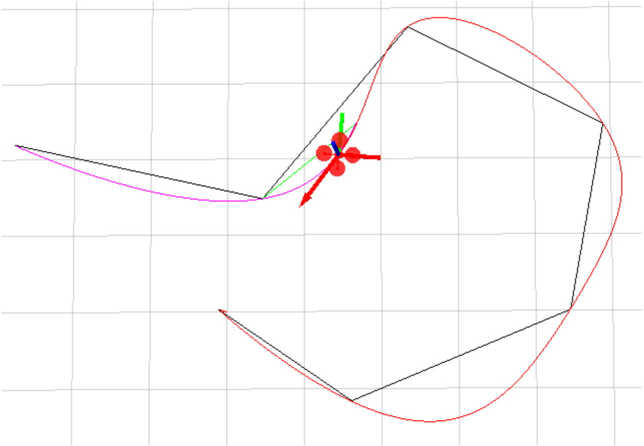
Smooth 3D Trajectories

- Smooth trajectory is beneficial for autonomous flight
  - Smooth trajectories respect the continuous nature of aerial robots
  - The robot should not stop at turns



Smooth 3D Trajectories

- General setup
  - Start, goal positions (orientations)
  - Waypoint positions (orientations)
    - Waypoints can be found by path planning (A*, RRT*, etc)
  - Smoothness criterion
    - Generally translates into minimizing rate of change of “input”



Differential Flatness

- The states and the inputs of a quadrotor can be written as algebraic functions of four carefully selected flat outputs and their derivatives
  - Enables automated generation of trajectories
  - Any smooth trajectory in the space of flat outputs (with reasonably bounded derivatives) can be followed by the under-actuated quadrotor
  - A possible choice:
    - $\sigma = [x, y, z, \psi]^T$
  - Trajectory in the space of flat outputs:
    - $\sigma(t) = [T_0, T_M] \rightarrow \mathbb{R}^3 \times SO(2)$

Differential Flatness

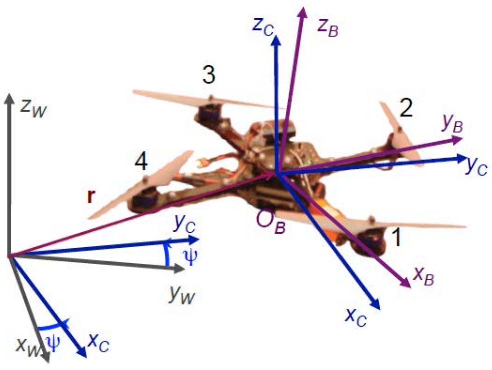
- Quadrotor states
  - Position, orientation, linear velocity, angular velocity

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

  - Equation of motions:

$$m\ddot{\mathbf{r}} = -mg\mathbf{z}_W + u_1\mathbf{z}_B.$$

$$\dot{\omega}_{\mathcal{BW}} = \mathcal{I}^{-1}\left[-\omega_{\mathcal{BW}} \times \mathcal{I}\omega_{\mathcal{BW}} + \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}\right]$$



- Position, velocity, and acceleration are simply derivatives of the flat outputs

Differential Flatness

- Orientation
  - Quadrotor state:

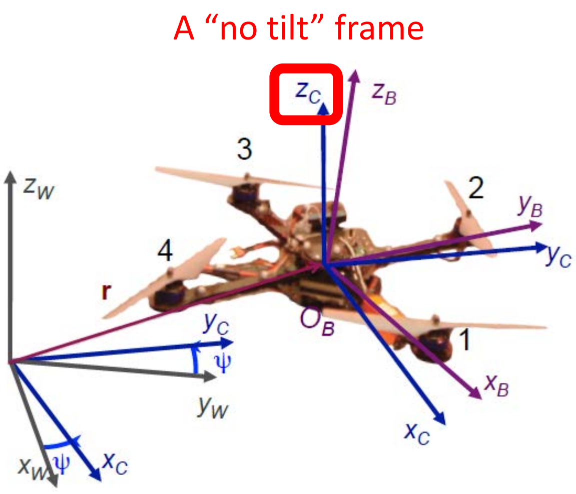
$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

  - From the equation of motion:

$$\mathbf{z}_B = \frac{\mathbf{t}}{\|\mathbf{t}\|}, \ \mathbf{t} = [\ddot{\sigma}_1, \ \ddot{\sigma}_2, \ \ddot{\sigma}_3 + g]^T$$

  - Define the yaw vector (Z-X-Y Euler):

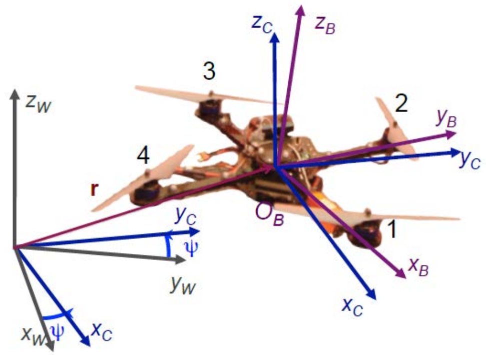
$$\mathbf{x}_C = [\cos\sigma_4, \ \sin\sigma_4, \ 0]^T$$



  - Orientation can be expressed in terms of flat outputs

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{\|\mathbf{z}_B \times \mathbf{x}_C\|}, \ \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \qquad {}^W R_B = [\mathbf{x}_B \ \ \mathbf{y}_B \ \ \mathbf{z}_B]$$

Differential Flatness



- Angular velocity
  - Quadrotor state:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

  - Take the derivative of the equation of motion

$$m\ddot{\mathbf{r}} = -mg\mathbf{z}_W + u_1\mathbf{z}_B. \longrightarrow m\dot{\mathbf{a}} = \dot{u}_1\mathbf{z}_B + \boxed{\omega_{\mathcal{BW}}} \times u_1\mathbf{z}_B$$

Body angular velocity viewed in the world frame

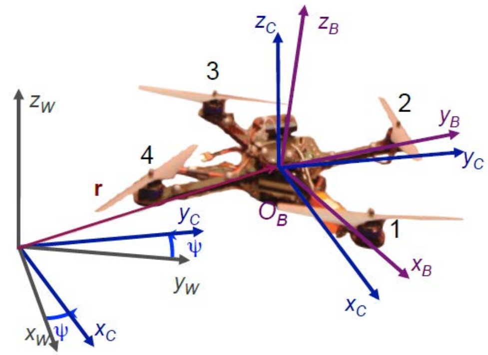
  - Quadrotors only have vertical thrust:

$$\dot{u}_1 = \mathbf{z}_B \cdot m\dot{\mathbf{a}}$$

  - We have:

$$\mathbf{h}_\omega = \omega_{\mathcal{BW}} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\mathbf{a}} - (\mathbf{z}_B \cdot \dot{\mathbf{a}})\mathbf{z}_B).$$

Differential Flatness



- Angular velocity
  - Quadrotor state:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

  - We have:

$$\mathbf{h}_\omega = \omega_{\mathcal{BW}} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\mathbf{a}} - (\mathbf{z}_B \cdot \dot{\mathbf{a}})\mathbf{z}_B).$$

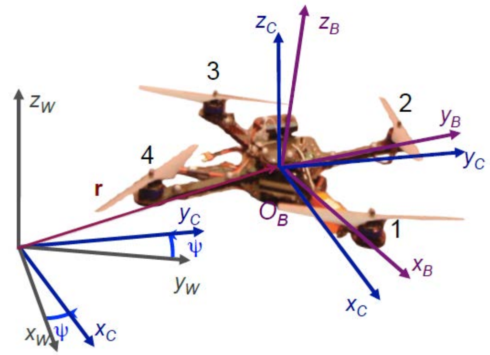
    - This is the projection of $\frac{m}{u_1}\dot{\boldsymbol{a}}$ onto the $x_B - y_B$ plane
  - We know that:

$$\omega_{\mathcal{BW}} = p\mathbf{x}_B + q\mathbf{y}_B + r\mathbf{z}_B.$$

  - Angular velocities along x_B and y_B directions can be found as:

$$p = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \; q = \mathbf{h}_\omega \cdot \mathbf{x}_B$$

Differential Flatness



- Angular velocity
  - Quadrotor state:

$$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$

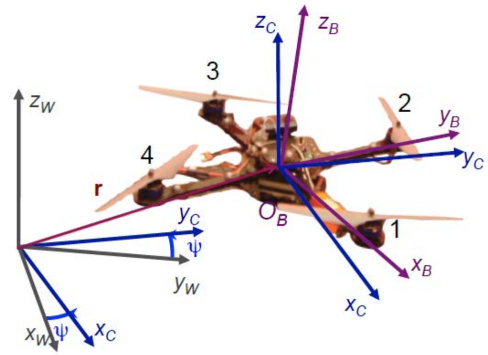
  - We have:

$$\mathbf{h}_\omega = \omega_{\mathcal{BW}} \times \mathbf{z}_B = \frac{m}{u_1}(\dot{\mathbf{a}} - (\mathbf{z}_B \cdot \dot{\mathbf{a}})\mathbf{z}_B).$$

    - This is the projection of $\frac{m}{u_1}\dot{\boldsymbol{a}}$ onto the $x_B - y_B$ plane
  - Since $\omega_{BW} = \omega_{BC} + \omega_{CW}$, where ω_{BC} has no z_B component:

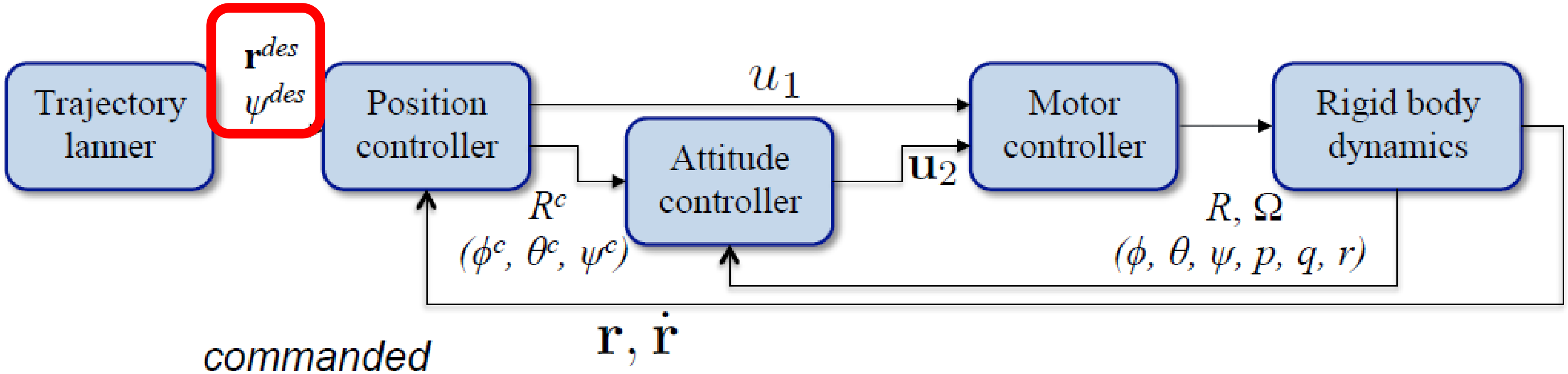
$$r = \omega_{CW} \cdot \mathbf{z}_B = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B.$$

Differential Flatness



- Summary
  - Quadrotor state:
    $$\mathbf{x} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, p, q, r]^T$$
  - Flat outputs:
    - $\sigma = [x, y, z, \psi]^T$
  - Position, velocity, acceleration
    - Derivatives of flat outputs
  - Orientation
    $$\mathbf{x}_C = [\cos\sigma_4,\ \sin\sigma_4,\ 0]^T \longrightarrow {}^W R_B = [\mathbf{x}_B \ \ \mathbf{y}_B \ \ \mathbf{z}_B]$$
  - Angular velocity
    $$p = -\mathbf{h}_\omega \cdot \mathbf{y}_B,\ q = \mathbf{h}_\omega \cdot \mathbf{x}_B \qquad r = \omega_{CW} \cdot \mathbf{z}_B = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B.$$

How about Inputs (u_1, u_2)??



commanded

actual (feedback)

$$(\ddot{r}_{i,des} - \ddot{r}_{i,c}) + k_{d,i}(\dot{r}_{i,des} - \dot{r}_i) + k_{p,i}(r_{i,des} - r_i) = 0$$

specified

$$u_1 = m(g + \ddot{r}_{3,c})$$

$$\phi_c = \frac{1}{g}(\ddot{r}_{1,c}\sin\psi_{des} - \ddot{r}_{2,c}\cos\psi_{des})$$

$$\theta_c = \frac{1}{g}(\ddot{r}_{1,c}\cos\psi_{des} + \ddot{r}_{2,c}\sin\psi_{des})$$

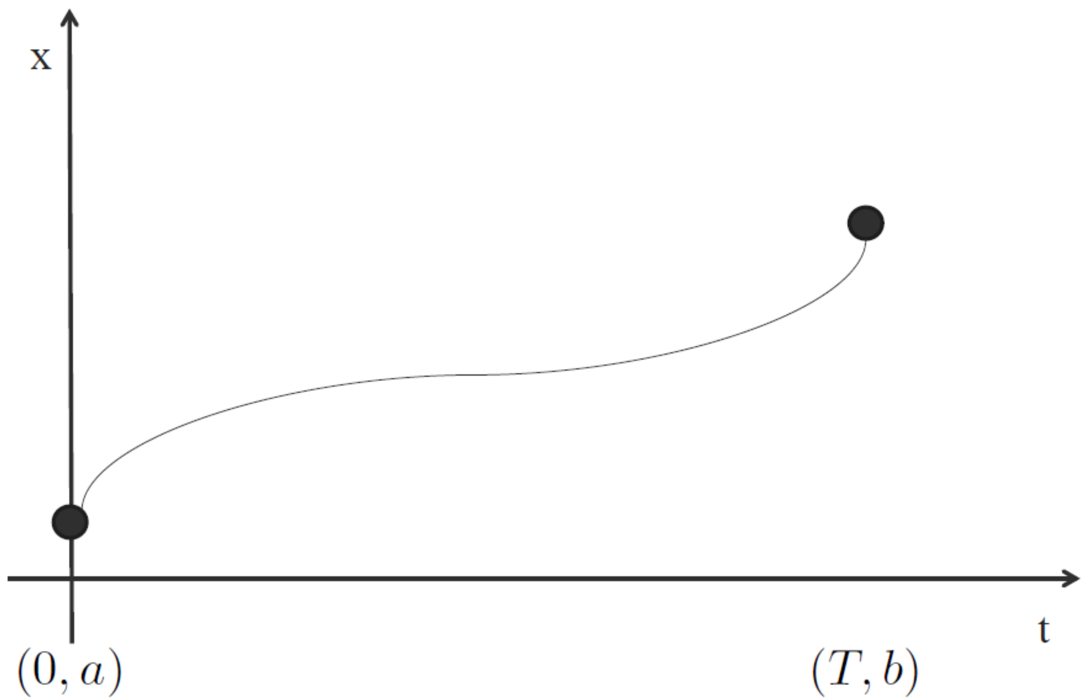
$$\mathbf{u}_2 = \begin{bmatrix} k_{p,\phi}(\phi_c - \phi) + k_{d,\phi}(p_c - p) \\ k_{p,\theta}(\theta_c - \theta) + k_{d,\theta}(q_c - q) \\ k_{p,\psi}(\psi_c - \psi) + k_{d,\psi}(r_c - r) \end{bmatrix}$$

Polynomial Trajectories

- Flat outputs:
  - $\sigma = [x, y, z, \psi]^T$
- Trajectory in the space of flat outputs:
  - $\sigma(t) = [T_0, T_M] \to \mathbb{R}^3 \times SO(2)$
- Polynomial functions can be used to specify trajectories in the space of flat outputs
  - Easy determination of smoothness criterion with polynomial orders
  - Easy and closed form calculation of derivatives
  - Decoupled trajectory generation in three dimensions

Smooth 1D Trajectory

- Design a trajectory $x(t)$ such that:
  - $x(0) = a$
  - $x(T) = b$



Smooth 1D Trajectory

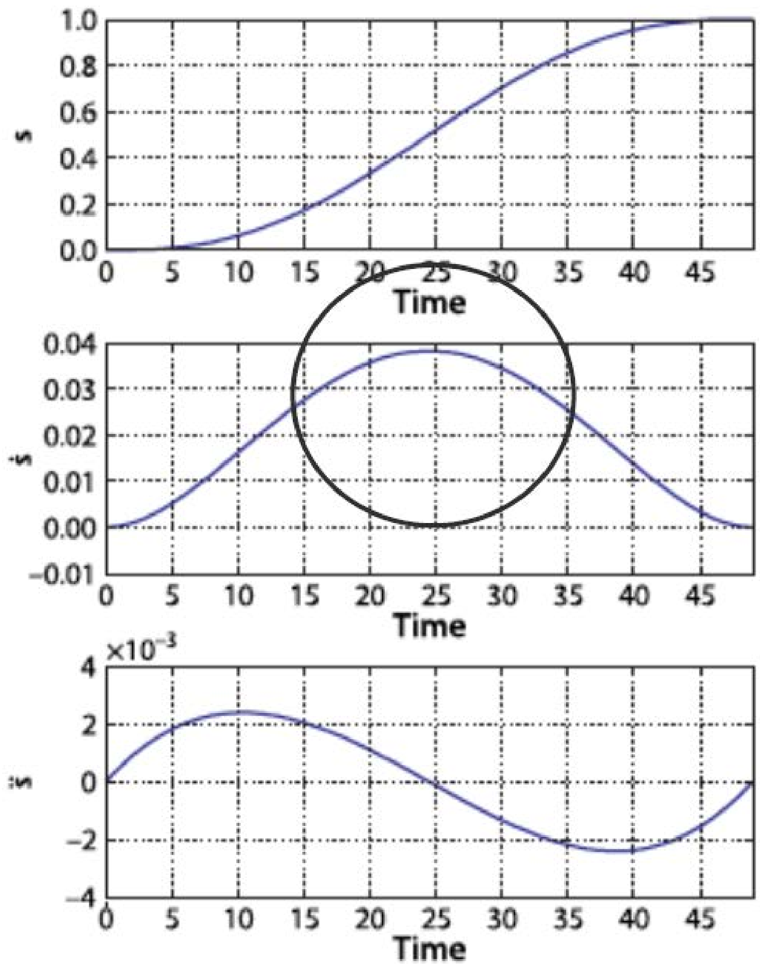
- 5^{th} order polynomial trajectory:
  - $x(t) = c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0$
- Boundary conditions

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

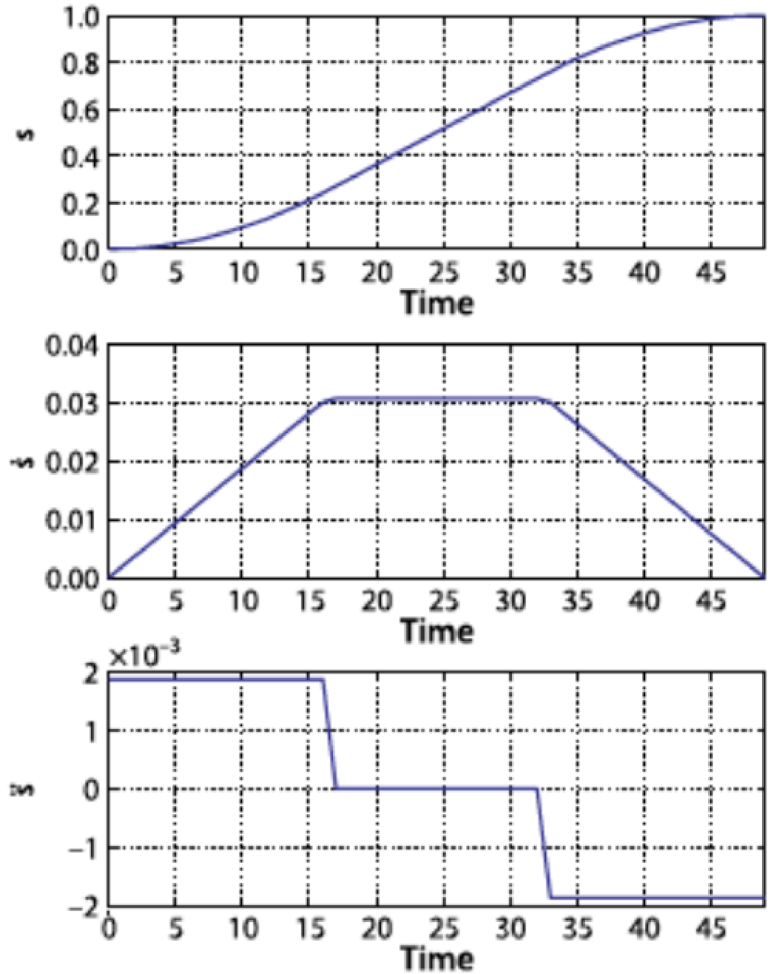
- Solve:

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

Smooth 1D Trajectory

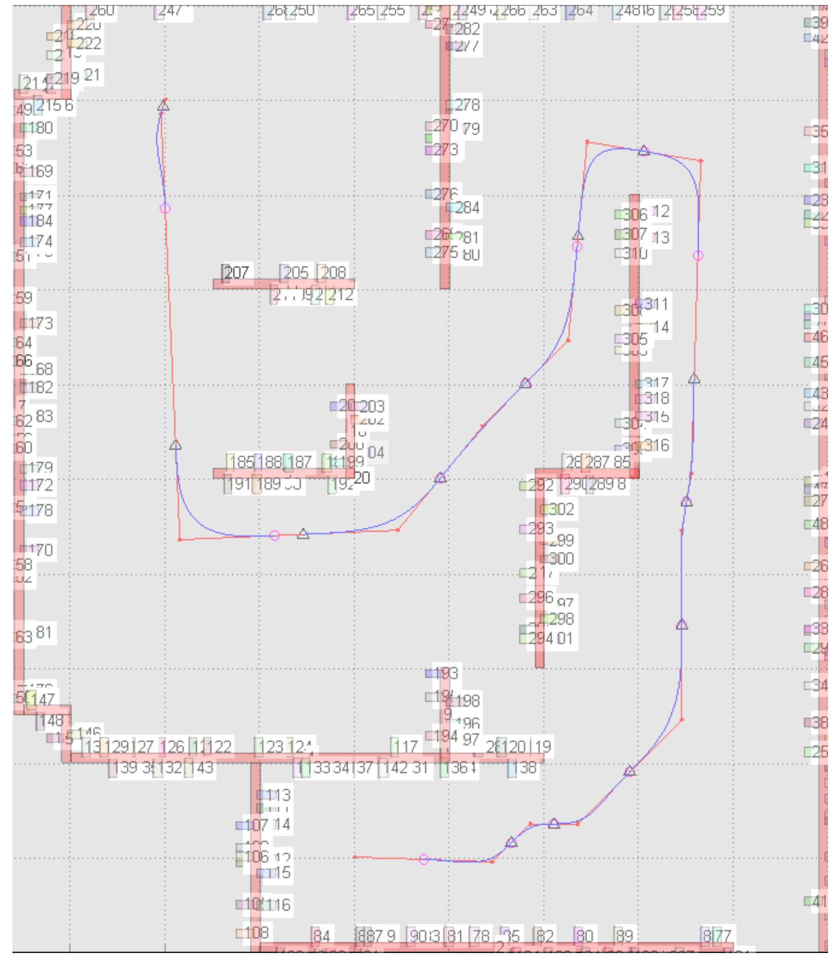


Bang-Bang Trajectory



Smooth Multi-Segment Trajectory

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



Smooth 1D Trajectory

- Generate each 5^{th} order polynomial indepently:
  - $x(t) = c_5 t^5 + c_4 t^4 + c_3 t^3 + c_2 t^2 + c_1 t + c_0$
- Boundary conditions

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

- Solve:

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

Optimization-based Trajectory Generation

- Explicitly minimize certain derivatives in the space of flat outputs
- Quadrotor dynamics

Derivative	Translation	Rotation	Thrust
0	Position		
1	Velocity		
2	Acceleration	Rotation	
3	Jerk	Angular Velocity	
4	Snap	Angular Acceleration	Differential Thrust
5	Crackle	Angular Jerk	Change in Thrust

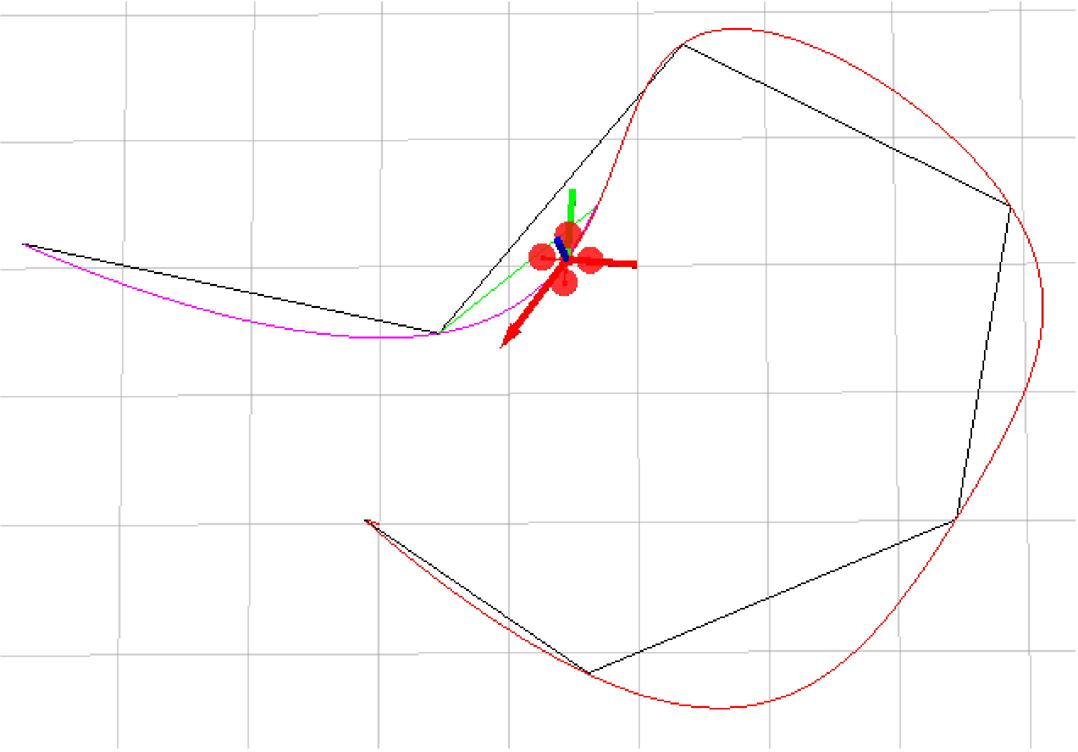
Optimization-based Trajectory Generation

- Explicitly minimize certain derivatives in the space of flat outputs
  - Minimum jerk: minimize angular velocity, good for visual tracking
  - Minimum snap: minimize differential thrust, saves energy

Derivative	Translation	Rotation	Thrust
0	Position		
1	Velocity		
2	Acceleration	Rotation	
3	Jerk	Angular Velocity	
4	Snap	Angular Acceleration	Differential Thrust
5	Crackle	Angular Jerk	Change in Thrust

Minimum Snap Trajectory Generation

- Multi-segment minimum snap trajectory



Minimum Snap Trajectory Generation

- Formulation – segment durations must be known!

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

Subject to:

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

$$\text{Continuity constraints: } f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j)$$

- Minimum degree polynomial:
  - Minimum jerk: N = 2 * 3(jerk) - 1 = 5
  - Minimum snap: N = 2 * 4(snap) - 1 = 7

Minimum Snap Trajectory Generation

- Cost function for one polynomial segment:

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

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

$$\Rightarrow (f^{(4)}(t))^2 = \sum_{i \geq 4, j \geq 4} i(i-1)(i-2)(i-3)j(j-1)(j-2)(j-3)t^{i+j-8} p_i p_j$$

$$\Rightarrow J(T) = \int_0^T (f^{(4)}(t))^2 dt = \sum_{i \geq 4, j \geq 4} \frac{i(i-1)(i-2)(i-3)j(j-1)(j-2)(j-3)}{i+j-7} T^{i+j-7} p_i p_j$$

$$\Rightarrow J(T) = \int_0^T (f^{(4)}(t))^2 dt = \begin{bmatrix} \vdots \\ p_i \\ \vdots \end{bmatrix}^T \begin{bmatrix} & & \vdots & & \\ \dots & & \frac{i(i-1)(i-2)(i-3)j(j-1)(j-2)(j-3)}{i+j-7} T^{i+j-7} & & \dots \\ & & \vdots & & \end{bmatrix} \begin{bmatrix} \vdots \\ p_j \\ \vdots \end{bmatrix}$$

$$\Rightarrow \boxed{J_k(T) = \mathbf{p}_k^T \mathbf{Q}_k \mathbf{p}_k}$$

Minimize this!

Minimum Snap Trajectory Generation

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

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

$$\Rightarrow \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} = x_{T,j}^{(k)}$$

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

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

$$\Rightarrow \mathbf{A}_j \mathbf{p}_j = \mathbf{d}_j$$

Minimum Snap Trajectory Generation

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

$$
\begin{aligned}
& f_j^{(k)}(T_j) = f_{j+1}^{(k)}(T_j) \\
\Rightarrow & \sum_{i \geq k} \frac{i!}{(i-k)!} T_j^{i-k} p_{j,i} - \sum_{l \geq k} \frac{l!}{(l-k)!} T_j^{l-k} p_{j+1,l} = 0 \\
\Rightarrow & \begin{bmatrix} \cdots & \frac{i!}{(i-k)!} T_j^{i-k} & \cdots & -\frac{l!}{(l-k)!} T_j^{l-k} \cdots \end{bmatrix} \begin{bmatrix} \vdots \\ p_{j,i} \\ \vdots \\ p_{j+1,l} \\ \vdots \end{bmatrix} = 0 \\
\Rightarrow & \begin{bmatrix} \mathbf{A}_j & -\mathbf{A}_{j+1} \end{bmatrix} \begin{bmatrix} \mathbf{p}_j \\ \mathbf{p}_{j+1} \end{bmatrix} = 0
\end{aligned}
$$

Minimum Snap Trajectory Generation

- Constrained quadratic programming (QP) formulation:

$$
\begin{aligned}
\min \quad & \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix}^T \begin{bmatrix} \mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_M \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix} \\
\text{s.t.} \quad & \mathbf{A} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix} = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}
\end{aligned}
$$

Minimum Snap Trajectory Generation

- Direct optimization of polynomial trajectories is numerically unstable
- A change of variable that instead optimizes segment endpoint derivatives is preferred

$$J = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}^T \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_M \end{bmatrix}^{-T} \begin{bmatrix} Q_1 & & \\ & \ddots & \\ & & Q_M \end{bmatrix} \begin{bmatrix} A_1 & & \\ & \ddots & \\ & & A_M \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}$$

Minimum Snap Trajectory Generation

- Use a selection matrix to separate free and constrained variables
  - Free variables: derivatives unspecified, only enforced by continuity constraints

$$J = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \underbrace{CA^{-T}QA^{-1}C^T}_{R} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix} = \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}^T \begin{bmatrix} R_{FF} & R_{FP} \\ R_{PF} & R_{PP} \end{bmatrix} \begin{bmatrix} \mathbf{d}_F \\ \mathbf{d}_P \end{bmatrix}$$

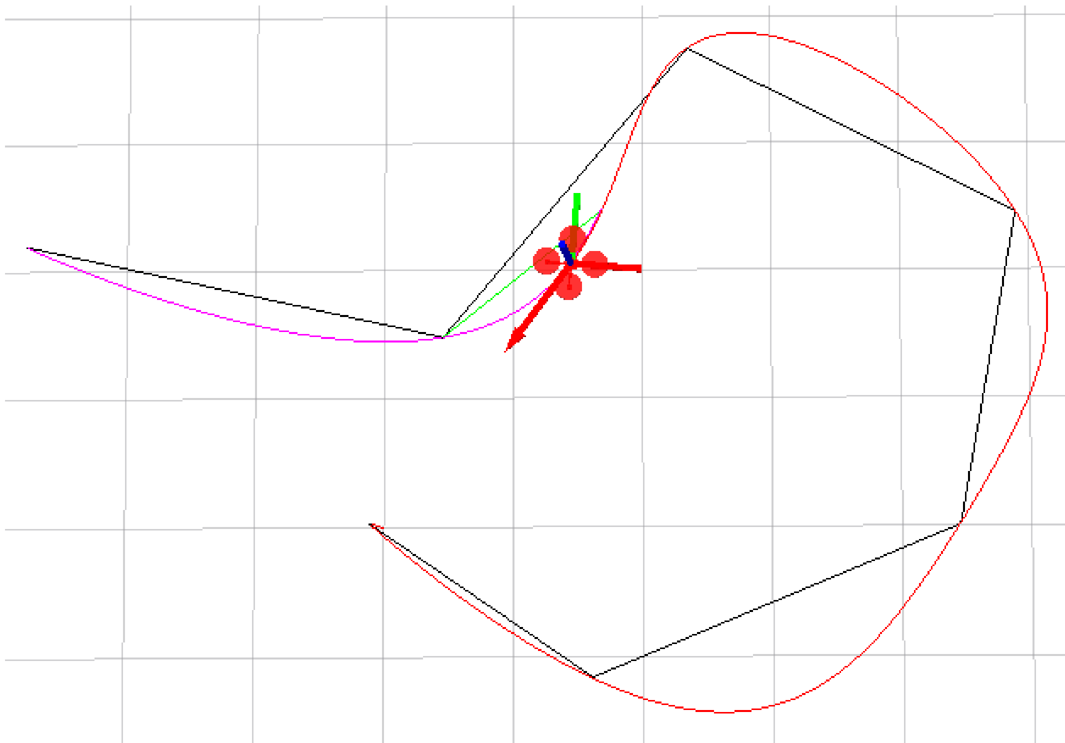
- Turned into an unconstrained quadratic programming that can be solved in closed form:

$$J = \mathbf{d}_F^T R_{FF} \mathbf{d}_F + \mathbf{d}_F^T R_{FP} \mathbf{d}_P + \mathbf{d}_P^T R_{PF} \mathbf{d}_F + \mathbf{d}_P^T R_{PP} \mathbf{d}_P$$

$$\mathbf{d}_P^* = -R_{PP}^{-1} R_{FP}^T \mathbf{d}_F$$

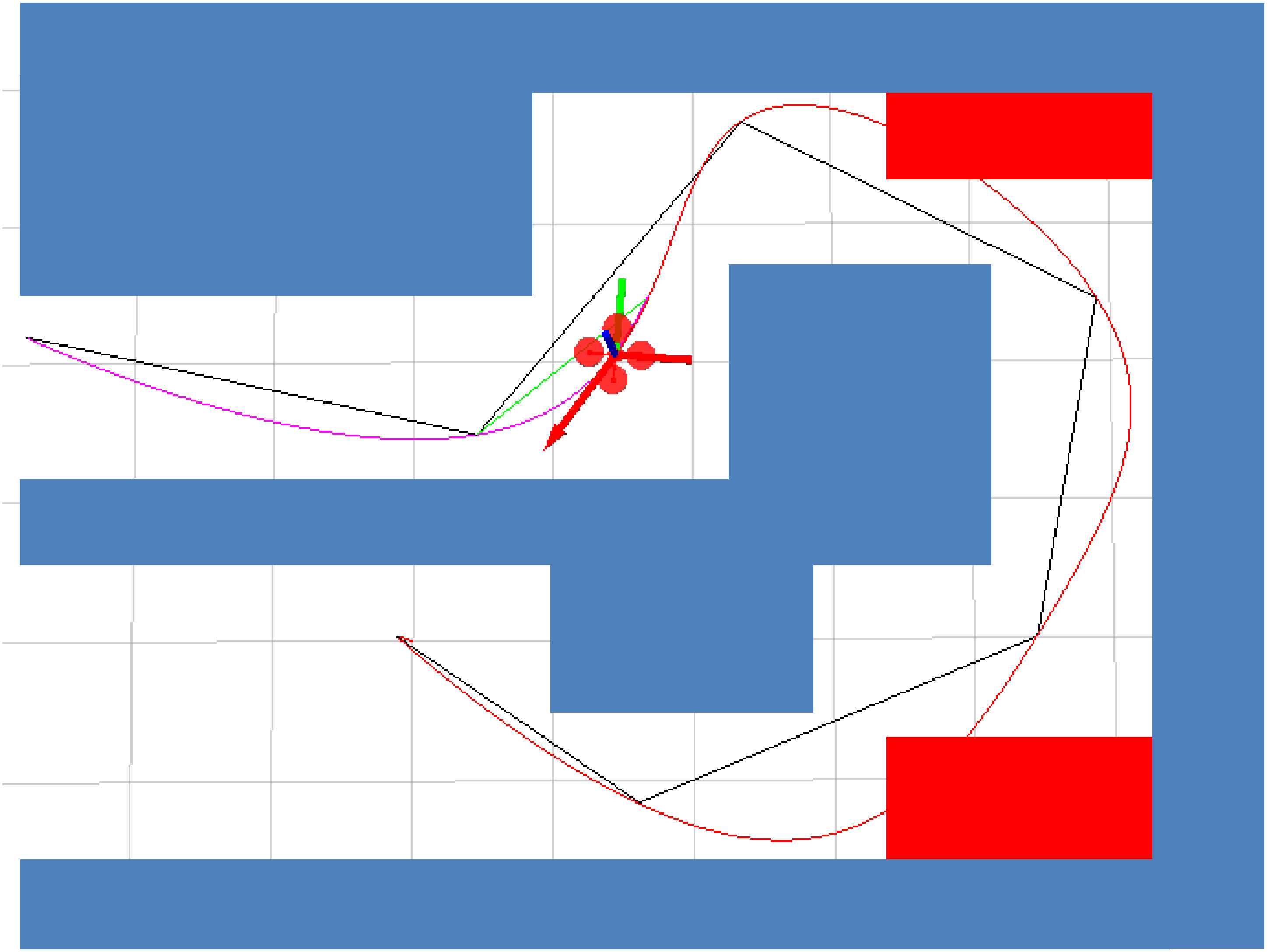
Minimum Snap Trajectory Generation

- Final trajectory

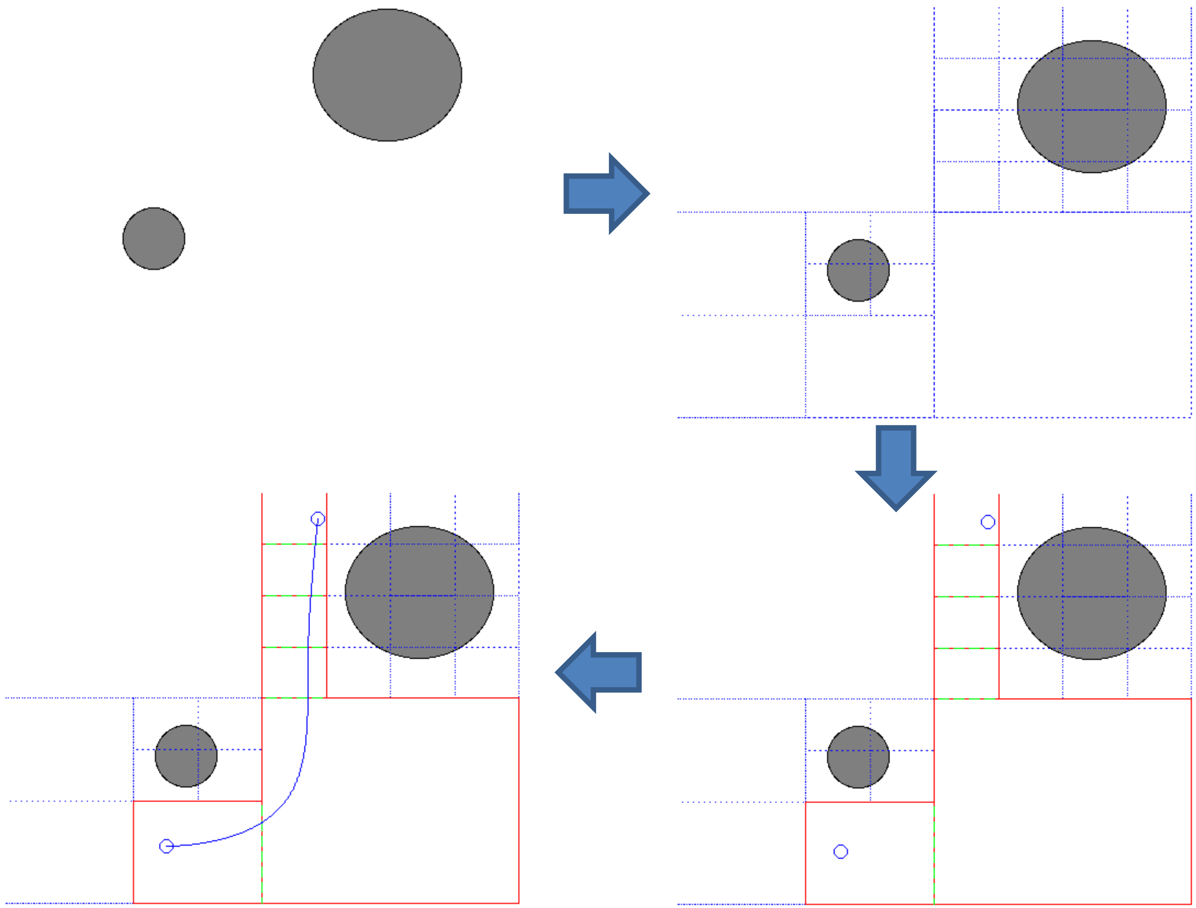


Minimum Snap Trajectory Generation

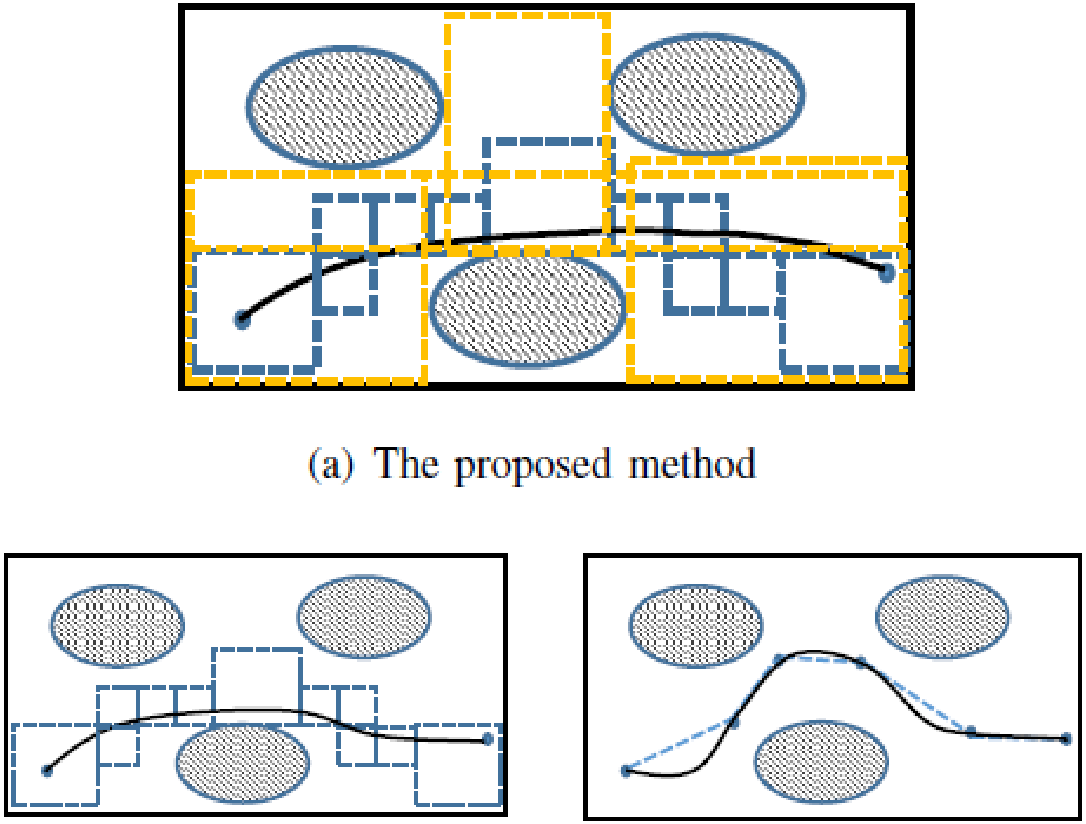
Minimum Snap Safe Trajectory Generation



Smooth Trajectory Generation with Guaranteed Obstacle Avoidance



Smooth Trajectory Generation with Guaranteed Obstacle Avoidance



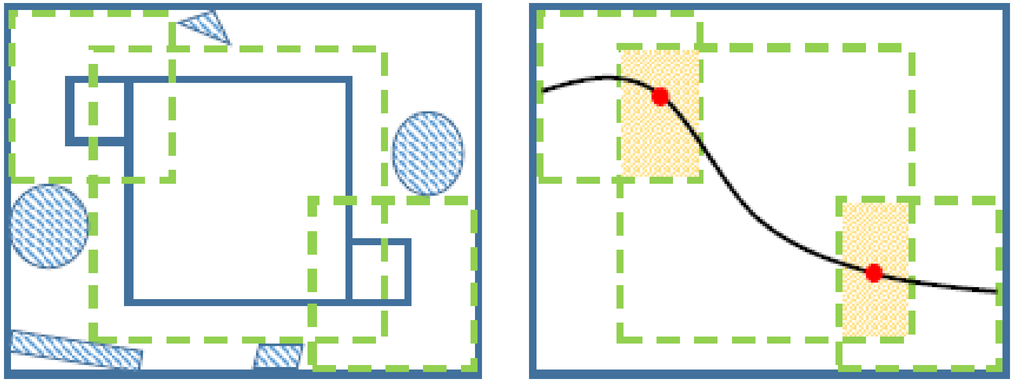
(a) The proposed method

(b) Our previous method [17]   (c) Waypoint-based method [2]

Smooth Trajectory Generation with Guaranteed Obstacle Avoidance

Quadratic programming formulation

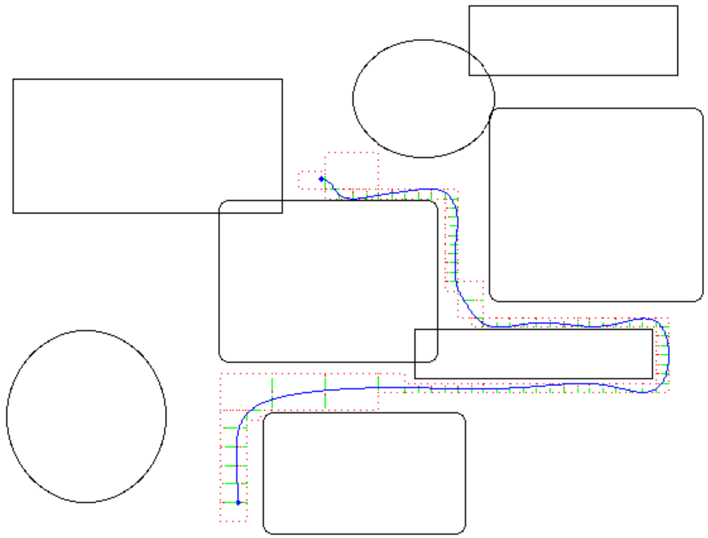
$$\begin{aligned} \min \quad & \mathrm{p}^{\mathrm{T}}\mathrm{H}\mathrm{p} \\ \text{s.t.} \quad & \mathrm{A}_{eq}\mathrm{p} = \mathrm{b}_{eq} \\ & \mathrm{A}_{lq}\mathrm{p} \leq \mathrm{b}_{lq} \end{aligned}$$

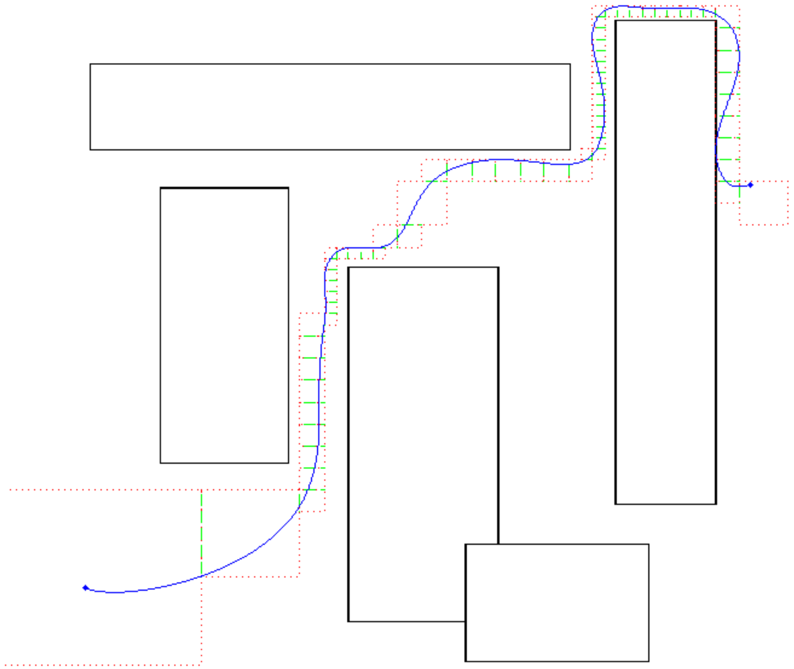


(a) Maximal volume flight corridor shown as green dashed boxes

(b) Adjustment of trajectory transition points within overlapping regions

Results - Trajectory Generation with Obstacle Avoidance





Results - Trajectory Generation with Obstacle Avoidance

Reading

- Paper Reading: “The GRASP Multiple Micro-UAV Test Bed”, Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar.
- Paper Reading: “Minimum Snap Trajectory Generation and Control for Quadrotors”, Daniel Mellinger and Vijay Kumar.

Logistics

- Project 1, phase 2 will be released tomorrow (30/9)
  - Tentative due: 7/10