Introduction to Aerial Robotics
Lecture 1

Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

7 February 2023

Logistics

About this Course…

- This course is about:
  - Autonomy for aerial robots
  - Modeling and control of multi-rotor aerial robots
  - Path and trajectory planning for multi-rotor aerial robots
  - Algorithms for vision-based state estimation
  - Algorithms for multi-sensor fusion
  - Real-time software implementations for autonomous aerial robots
  - Lots of fun with robots ☺
- This course is NOT about:
  - Aerodynamics ☹
  - Aircraft design ☹
  - Electromechanical systems of aerial robots ☹

Course structure

- Weekly lectures + lots of projects + an in-class midterm exam

- Grading Scheme:
  - Midterm exam: 20%
  - Project 1: 30%
  - Project 2: 20%
  - Project 3: 30%

- More details can be found on the Canvas homepage.

Teaching Team

- **Instructor:**
  - Shaojie Shen (eeshaojie@ust.hk)
    Office Hour: by appointment

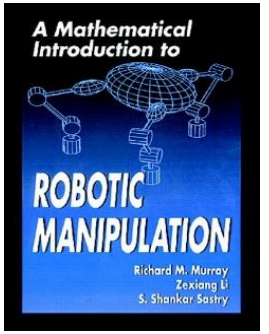
- **Teaching Assistants:**
  - Haokun Wang (hwangeh@connect.ust.hk)
    Office Hour: by appointment
  - Peize Liu (pliuan@connect.ust.hk)
    Office Hour: by appointment

Administrative Stuff

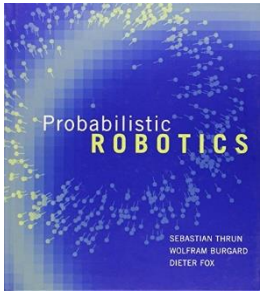
- **Lecture:**
  - Tuesday 1:30 pm - 4:20 pm
    - Rm 5560
    - Students who are not registered are welcome to sit in the lectures, but their assignments will not be graded
- **Labs:**
  - LA1: We 06:00PM - 08:50PM, LA2: Th 01:30PM - 04:20PM
    - Robotics Institute Flight Area
    - We will not have lab sessions every week, refer to Canvas site for details
- **Course Website:**
  - https://canvas.ust.hk/courses/48637

(Non-Compulsory) Reference Books

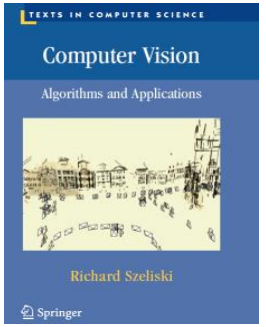
- A Mathematical Introduction to Robotic Manipulation
  - Richard M. Murray, Zexiang Li, S. Shankar Sastry;
  - http://www.cds.caltech.edu/~murray/books/MLS/pdf/mls94-complete.pdf



- Probabilistic Robotics;
  - Sebastian Thrun, Wolfram Burgard, Dieter Fox;



- Computer Vision: Algorithms and Applications
  - Richard Szeliski
  - http://szeliski.org/Book/



Workload & Expectation

- **Expected Student Background:**
  - Linear algebra
  - Probability
  - MATLAB programming skills
  - C++ programming skills (VERY IMPORTANT)
  - Linux
  - Love robots ☺

- **Workload:**
  - Attend lectures
  - Lots of project work
  - Have fun with robots ☺

Programming!

- MATLAB
- C++
- Robot Operating System (ROS) [http://www.ros.org/]
- Requires coding on both desktop PC and embedded platforms

Overview of Aerial Robotics

Aerial Robots = Military Drones?

- Mostly Big
- 10+ Hours of Flight Time
- Remote Control
  - Waypoint planning
  - Some joystick
  - Flight crew 4-10

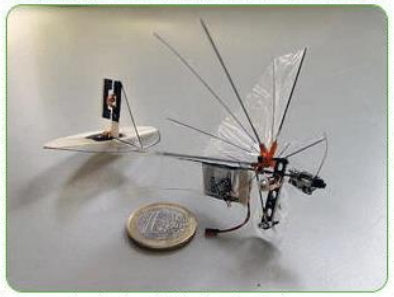
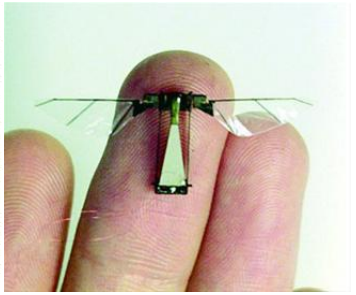
"Drones mischaracterize what these things are. They're not dumb. Nor are they unmanned, actually. They're remotely piloted aircraft."

-- Gen. Norton Schwarz, August 10, 2012

Unmanned Aerial Vehicles



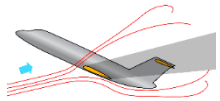


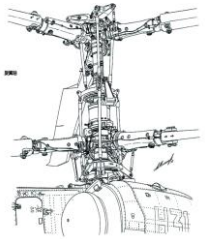




Unmanned Aerial Vehicles

- Types of UAVs
  - Fixed wing
    - Long flight time, large payload, self-stabilized system ☺
    - Requires runway, may stall, needs aerodynamic design ☹
  - Helicopter
    - Vertical takeoff and landing ☺
    - OK flight time and payload
    - Complex mechanical system, unstable systems ☹
  - Multi-rotor
    - VTOL, simple mechanical design ☺
    - Hard to scale up
    - Short flight time, small payload, unstable system ☹

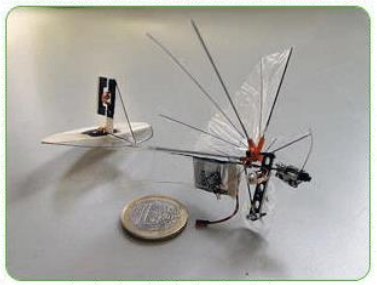


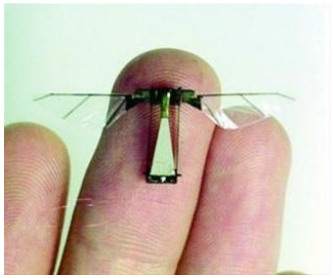




Unmanned Aerial Vehicles

- Anything even better?
  - Vertical Take-Off and Landing (VTOL) UAV
    - VTOL + long flight time + long range ☺
    - Technology not well developed ☹
  - Flapping Wing / Bio-Inspired UAVs
    - Suitable for small platforms ☺
    - Technology not well developed ☹

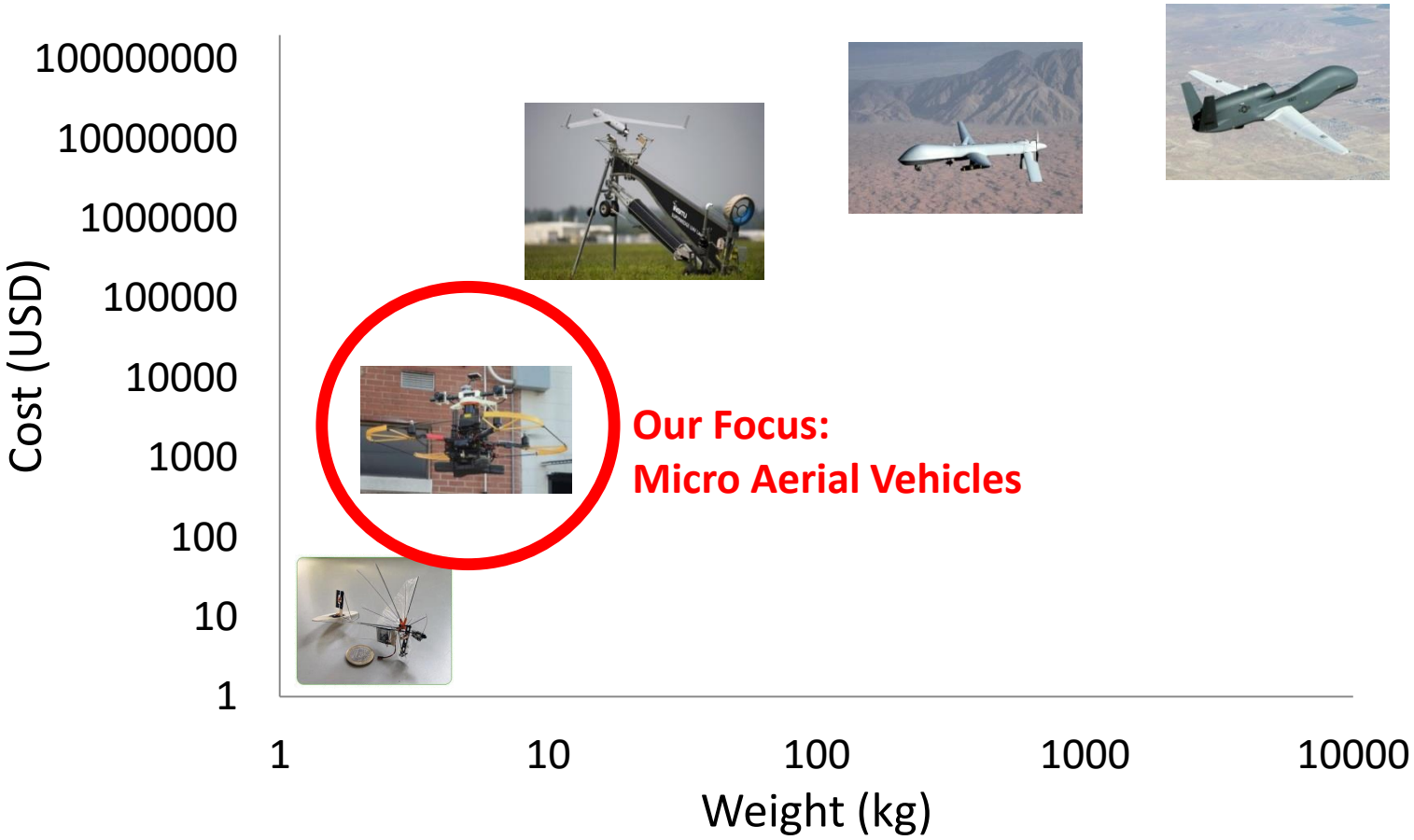




Contents [hide]

Unmanned Aerial Vehicles

- Projected civilian market of $400 billion USD by 2020
- Regulations are coming in place



Multi-Rotor Micro Aerial Vehicles

- Small size (<1m)
- Adequate payload (1-5kg)
- Low cost (< 10k USD)
- Safe
- Superior mobility


Inspection


Search and Rescue


Transportation

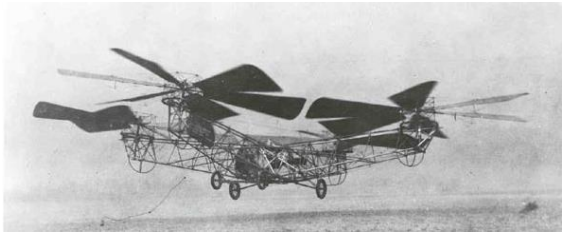

Aerial Photography

Law enforcement


Agriculture

Development of Multi-Rotors

- de Bothezat helicopter (1922)
  - Flight altitude 5m during demonstration
  - Fully manual controlled unstable system ☹



- Nothing happened in the next few decades
  - Almost impossible for manual control ☹
  - Multi-rotors cannot be scaled up ☹
  - No suitable sensors (tradition IMUs are big and expensive) ☹

- STARMAC @ Stanford University (2004)
  - Thanks to the development of smart mobile devices
  - MEMS sensors + embedded computers

Development of Multi-Rotors

- The Role of Academia
  - University of Pennsylvania
    - Vijay Kumar (Planning, control, swarm)
  - Massachusetts Institute of Technology
    - Jonathan How (modelling, control)
    - Nicolas Roy (perception)
  - University of California, Berkeley
    - Claire Tomlin (control, policies)
  - ETH Zurich
    - Roland Siegwart (perception, control)
    - Raffaello D'Andrea (control, swarm)
    - Marc Pollefeys (perception, Pixhawk)
  - Hong Kong University of Science Technology
    - Zexiang Li (founded DJI with Frank Wang)
    - I actually want to put myself here, but…
  - Also involves other areas such as communication, material science, mechanical engineering, electronics, aerospace engineering, system engineering…

Commercialization of Aerial Robots

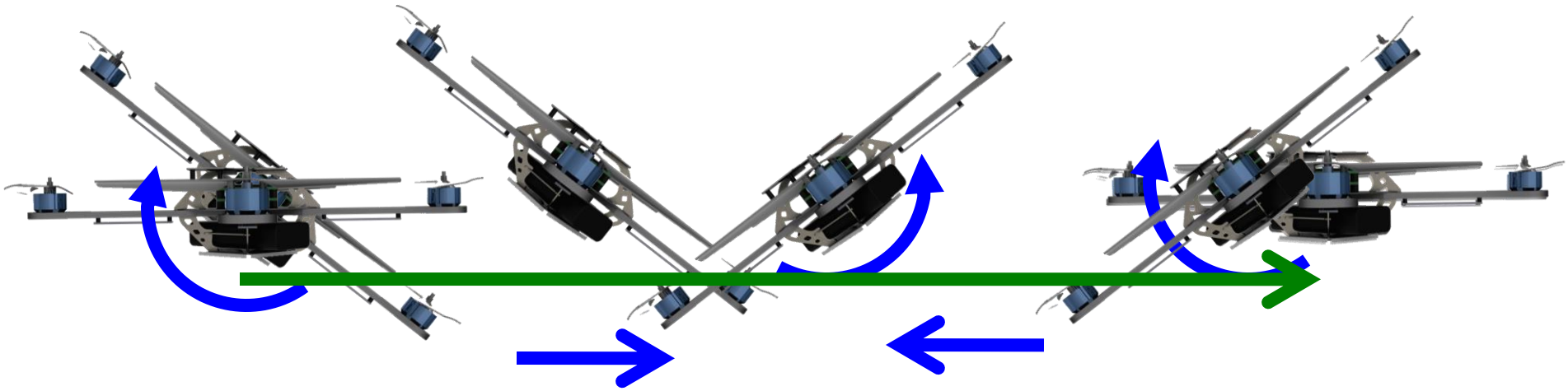
Quadrotors

CAD Models from V. Kumar

Quadrotors - Rotation

CAD Models from V. Kumar

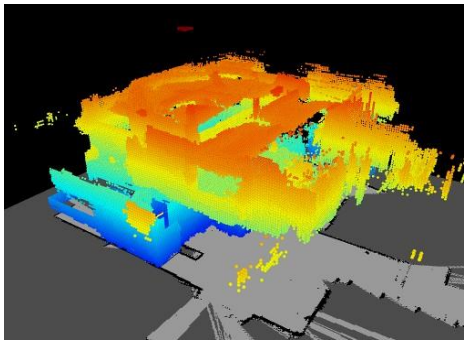
Quadrotors - Translation

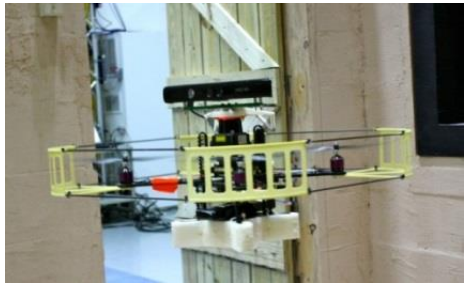


CAD Models from V. Kumar

Towards Autonomous Flight…

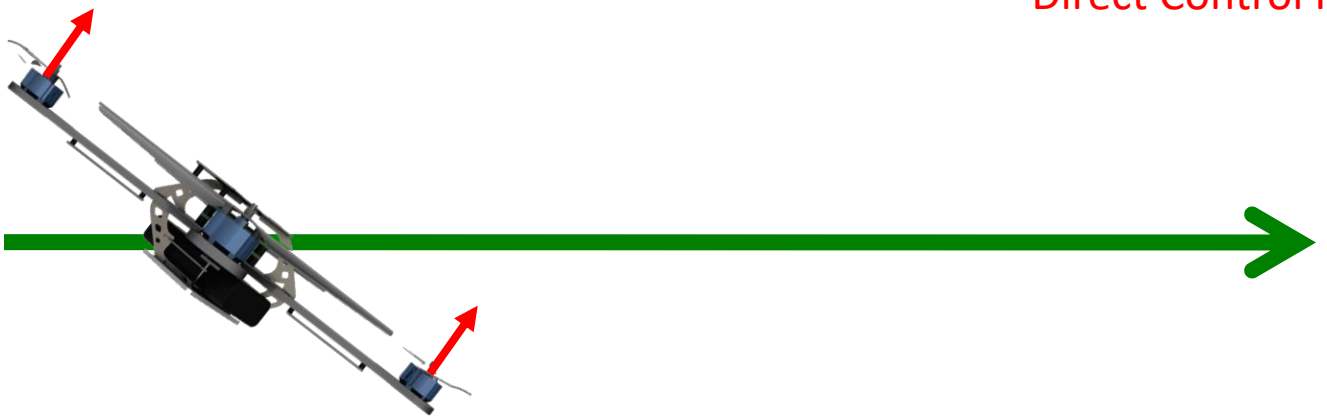
- Sensing and perception — Part 2
- State estimation — Part 3
- Path planning
- Trajectory generation — Part 1
- Control
- System integration — Projects!





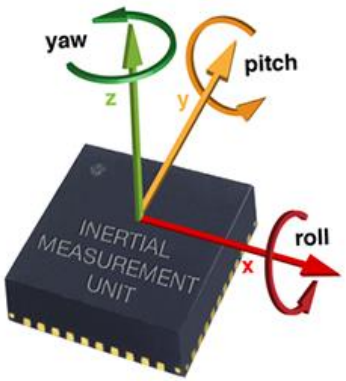
How to fly? – Dynamics & Control

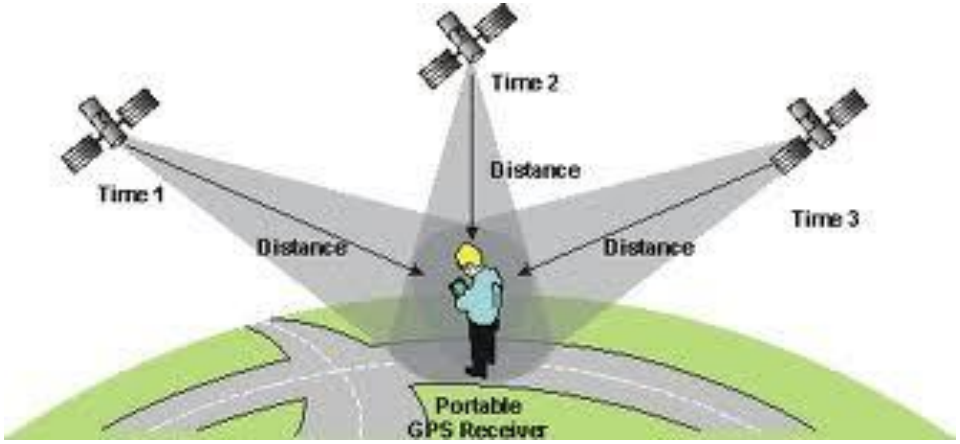
Derivative	Translation	Rotation	Thrust
0	Position ⇨ Control Target		
1	Velocity		
2	Acceleration	Rotation	
3	Jerk	Angular Velocity	
4	Snap	Angular Acceleration	Differential Thrust
5	Crackle	Angular Jerk	Change in Thrust ⇩ Direct Control Input



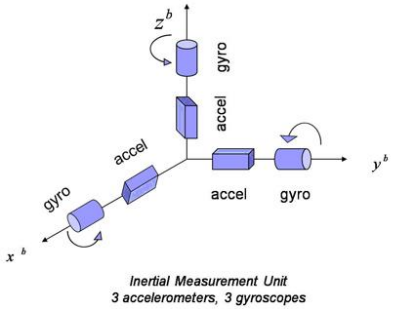
How to Fly? – Sensing & Estimation

- Answer the key question: Where am I?
- Proprioceptive sensors
  - Low cost Inertial Measurement Unit
- Exteroceptive sensors
  - GPS
  - Magnetometer
  - Barometer
  - Cameras
  - Laser range finders
  - etc.
- Processors
- Algorithms

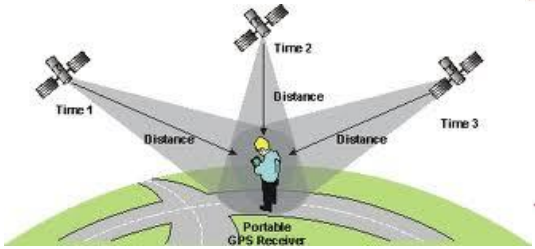




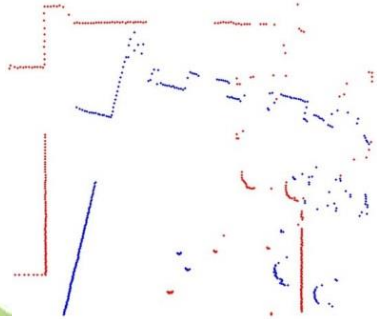
How to Fly? – Sensing & Estimation



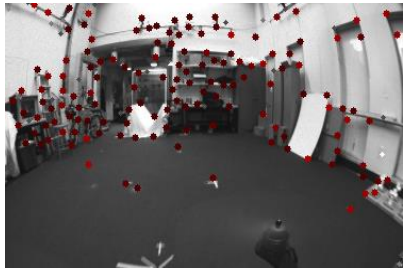
Inertial Measurement Unit



GPS



Laser Range Finder



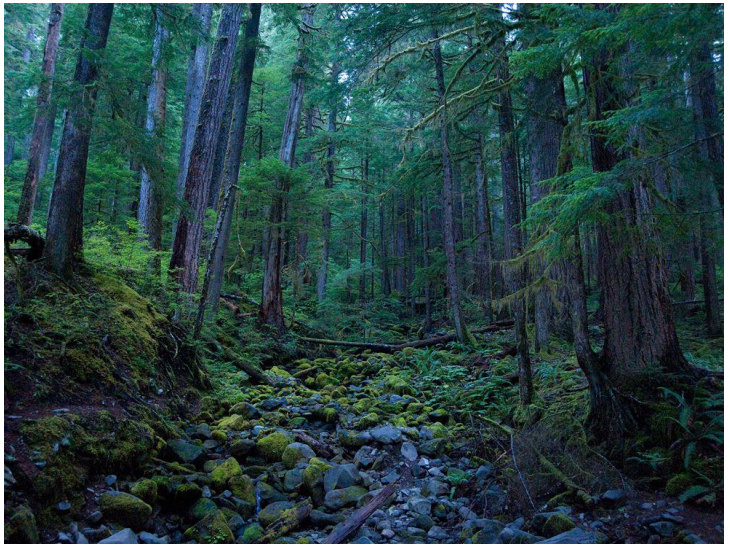
Camera

How to Fly? – Navigation

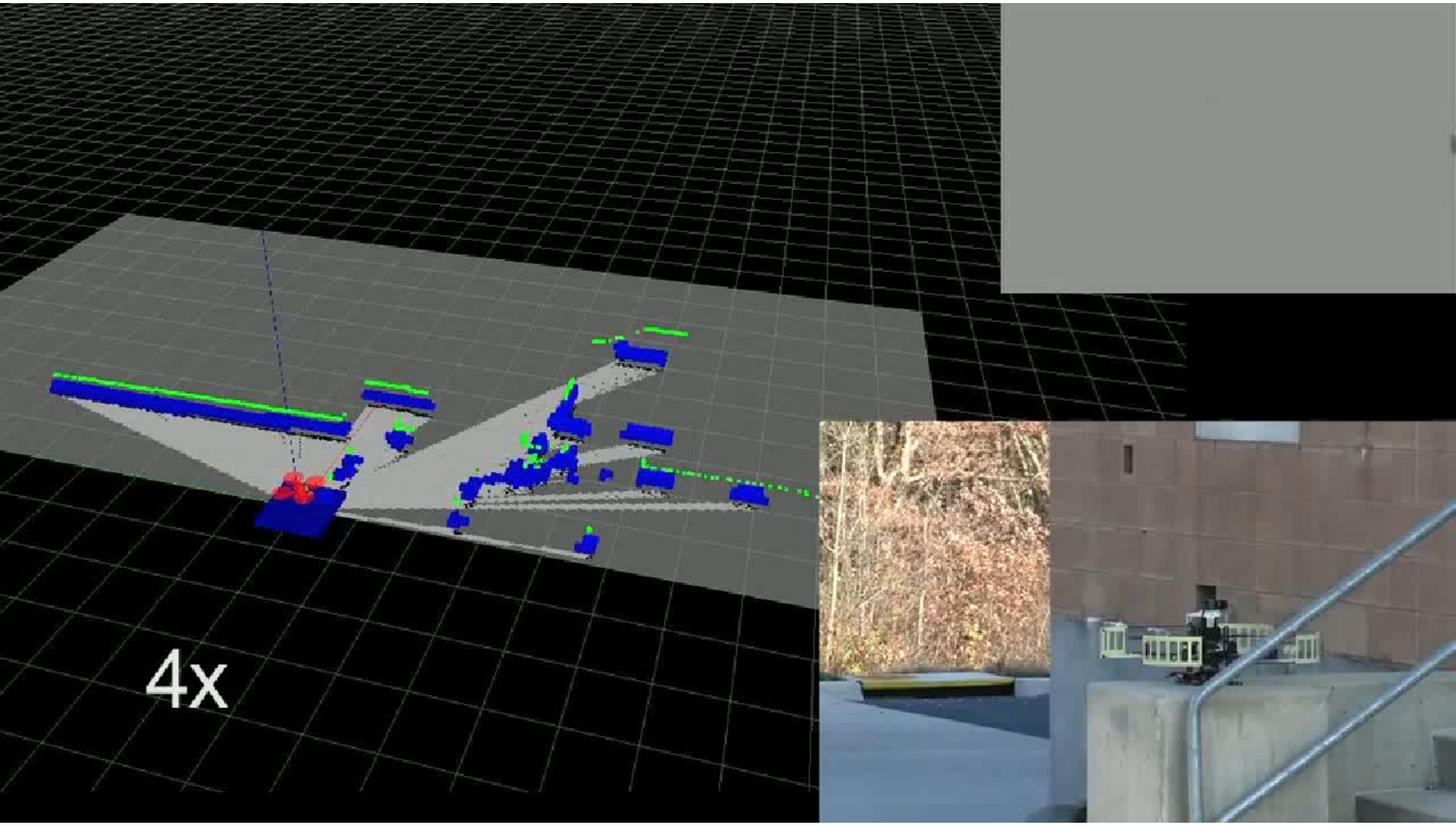
- Remote control
  - Requires line of sight
  - Requires communication link
  - Requires skilled pilots
- Inertial navigation
  - Requires aviation grade IMU
  - Heavy and expensive
- GPS-based navigation
  - Waypoint following
  - No obstacle avoidance
  - GPS can be unreliable

How to Fly? – GPS-based Navigation

How to Fly? – GPS-denied Navigation



How to Fly? – Laser-based Navigation



How to Fly? – Laser-based Navigation

Online Quadrotor Trajectory Generation and Autonomous Navigation on Point Clouds

Fei Gao and Shaojie Shen

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

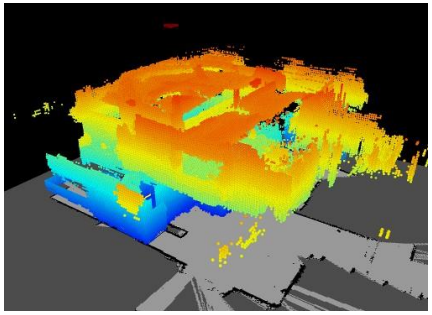
High resolution video available at
http://www.ece.ust.hk/~eeshaojie/ssrr2016fei.mp4

Gao F, Shen S. SSRR, 2016

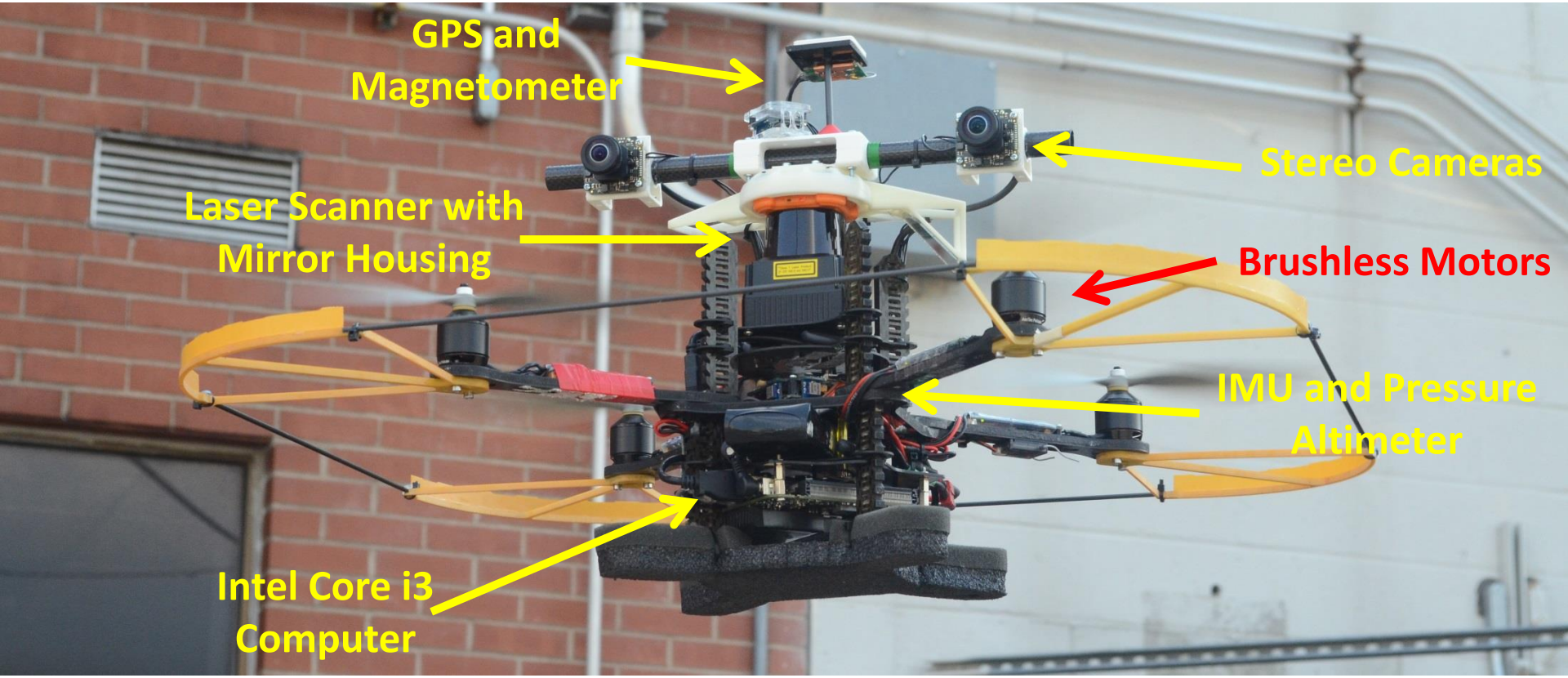
How to Fly? – Vision-based Navigation

Challenges

- Sensing & Perception
  - 3D sensing & mapping
- State Estimation & Localization
  - Low latency & high accuracy
- Obstacle avoidance
  - Complex & unknown environments
- Trajectory Control
  - Aggressive maneuvers
  - Smooth trajectory tracking
- System integration
  - Limited sensing & computation
  - Autonomous operations



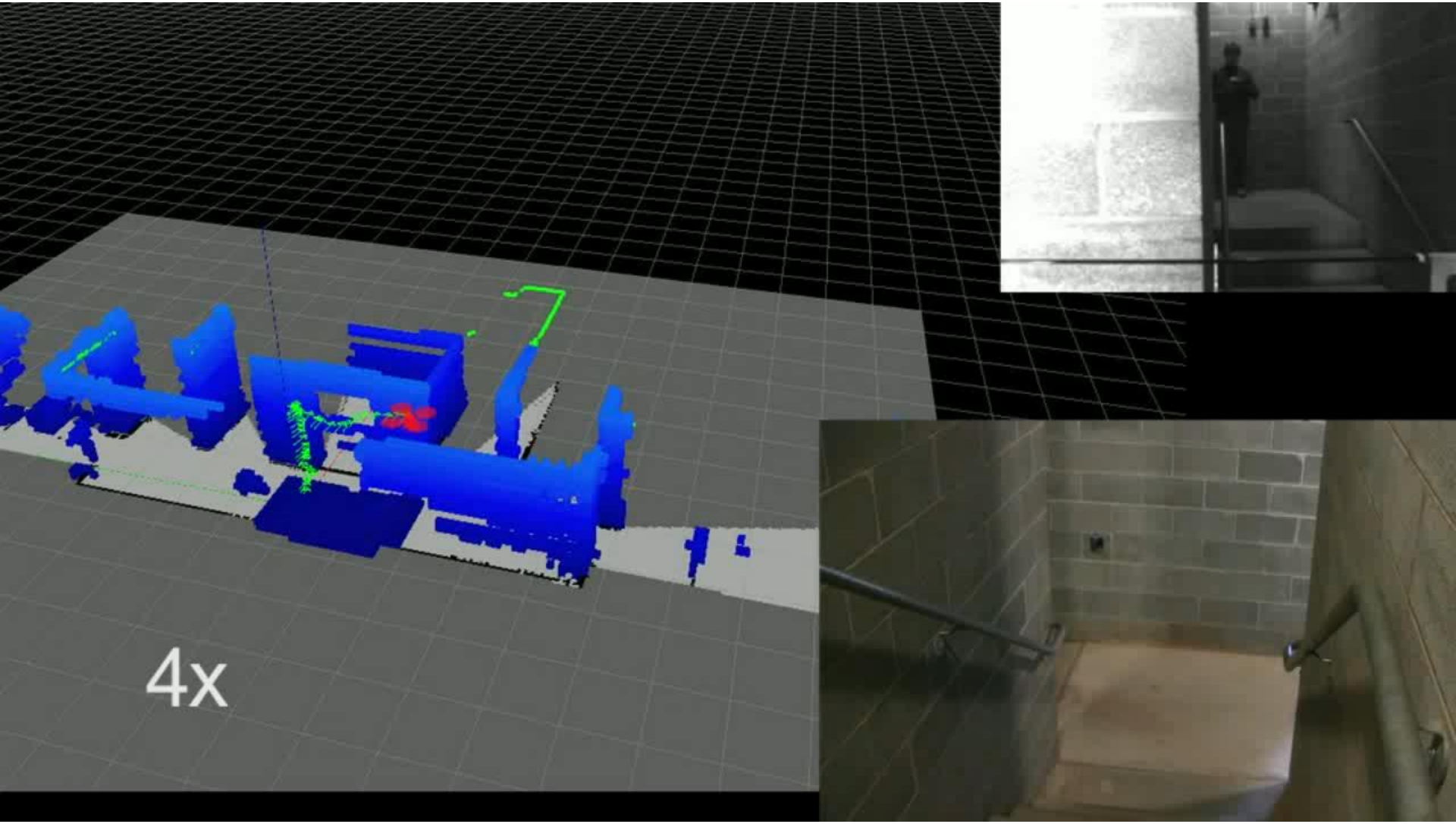
A Flying Robot



Goal

"Prometheus"

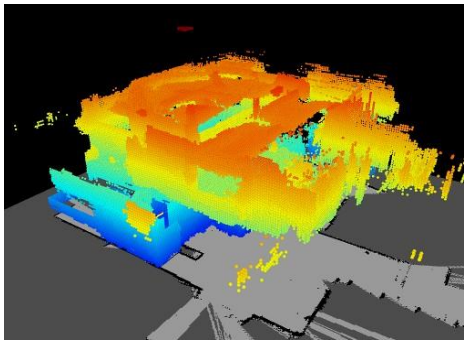
Fly over Multiple Floors

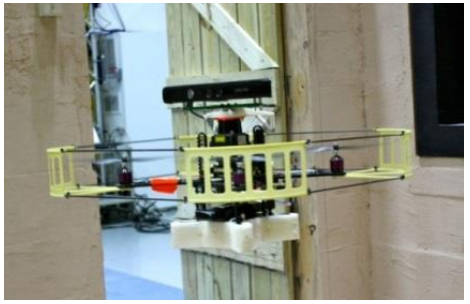


Shen, et al, 2013

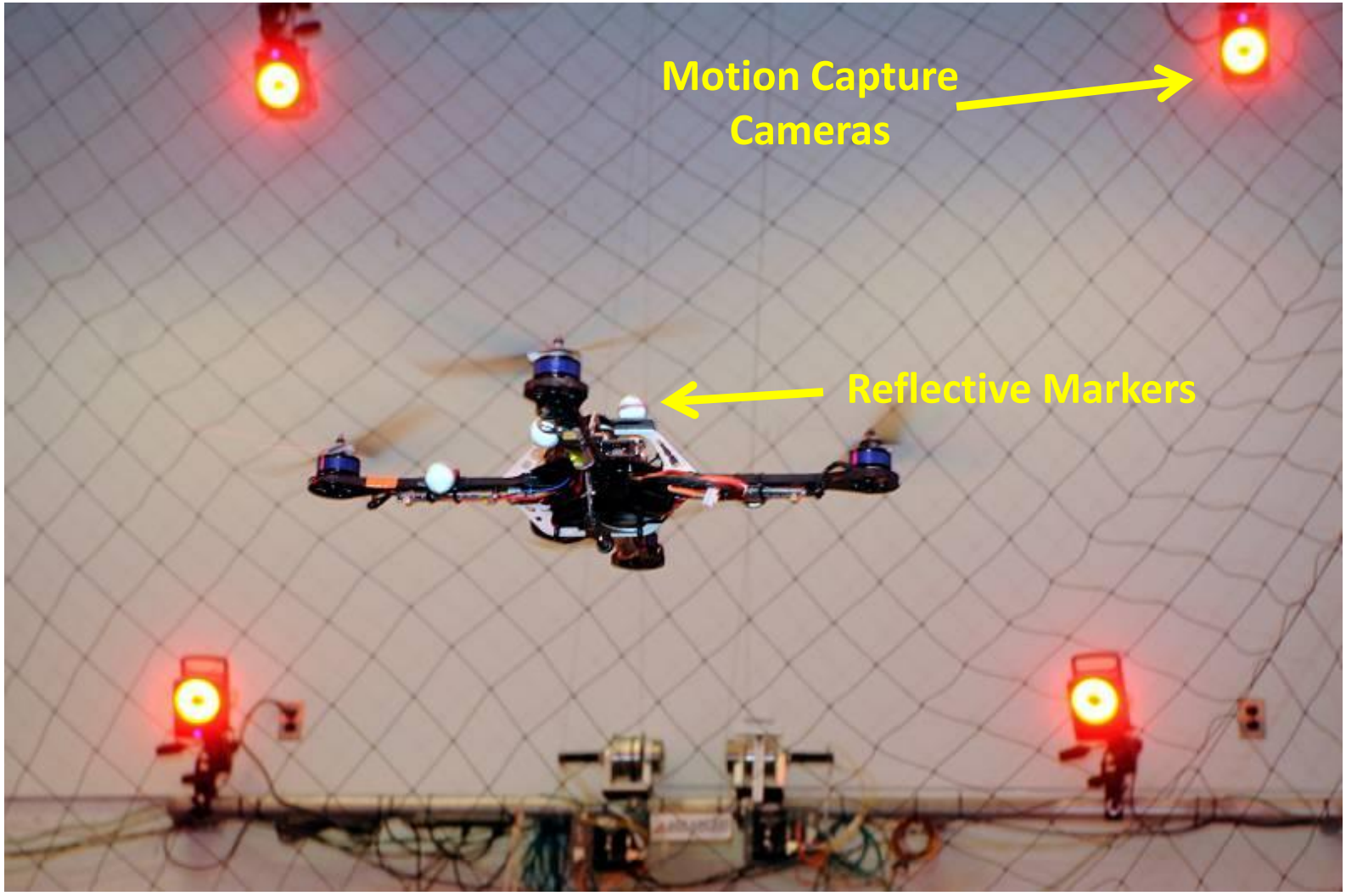
Towards Autonomous Flight…

- Sensing and perception
- State estimation
- Path planning
- Trajectory generation
- Control
- System integration





Bypass the Sensing Problem



Robust Control

Mellinger, 2011

Flying Inverted Pendulum

A Flying Inverted Pendulum

Cooperative Ball Throwing

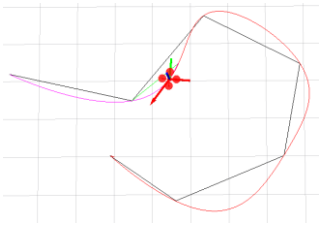
The Flying Machine Arena

Cooperative Quadrocopter Ball Throwing and Catching

IDSC

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

The Complete System



IMU

GPS Velocity

GPS Position

Pressure Altimeter

Cameras

Laser Scanner

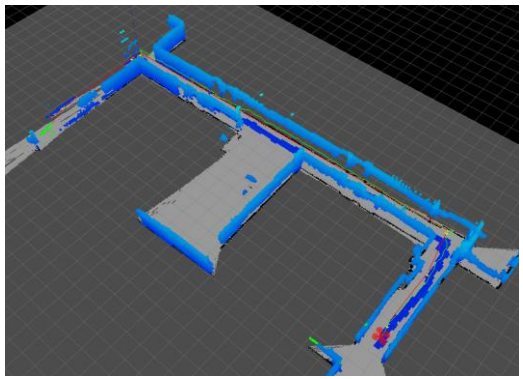
Laser Altimeter

State Estimation & Multi-Sensor Fusion

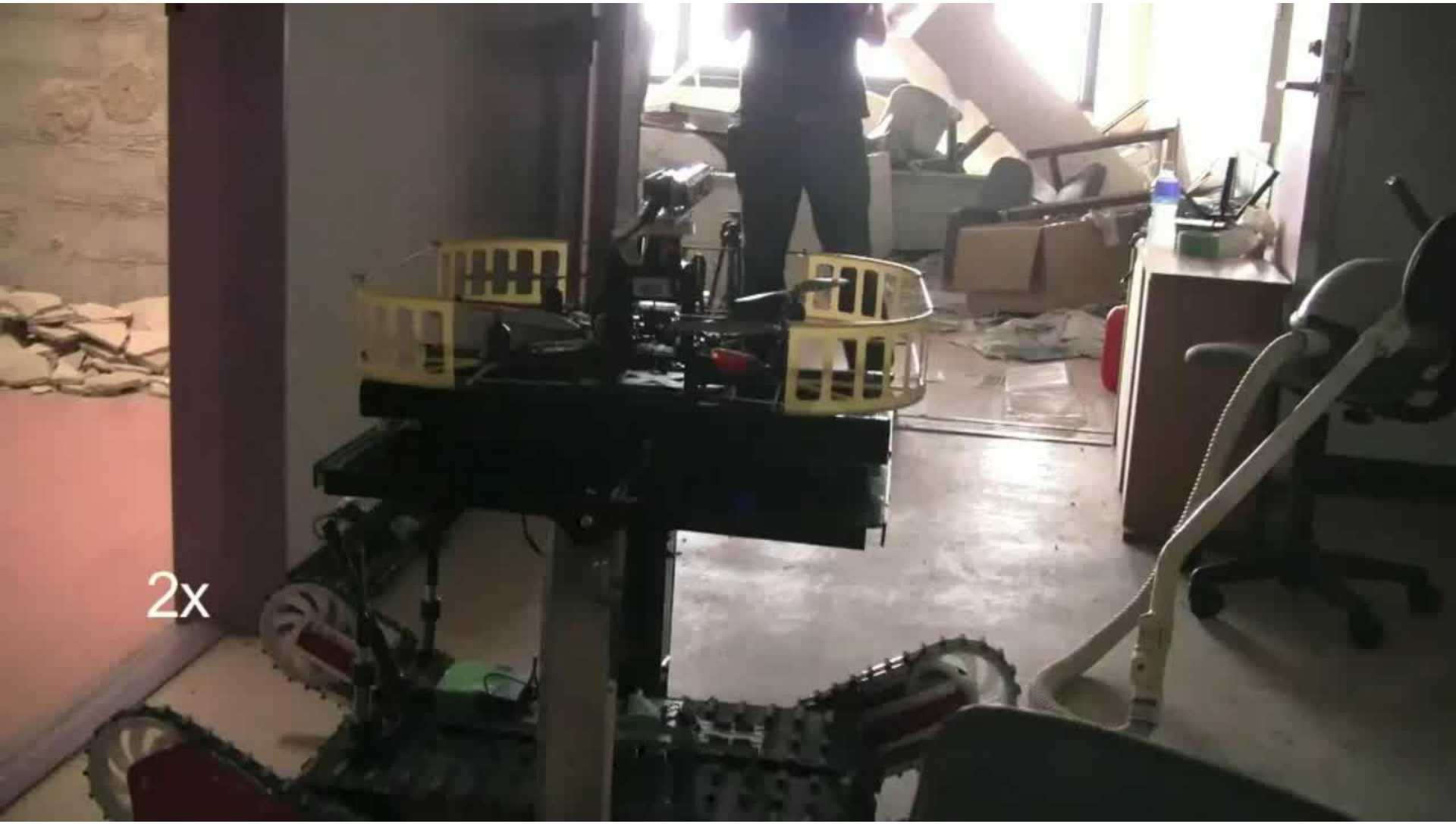
3D Mapping

Trajectory Generation & Control

Path Planning & Obstacle Avoidance



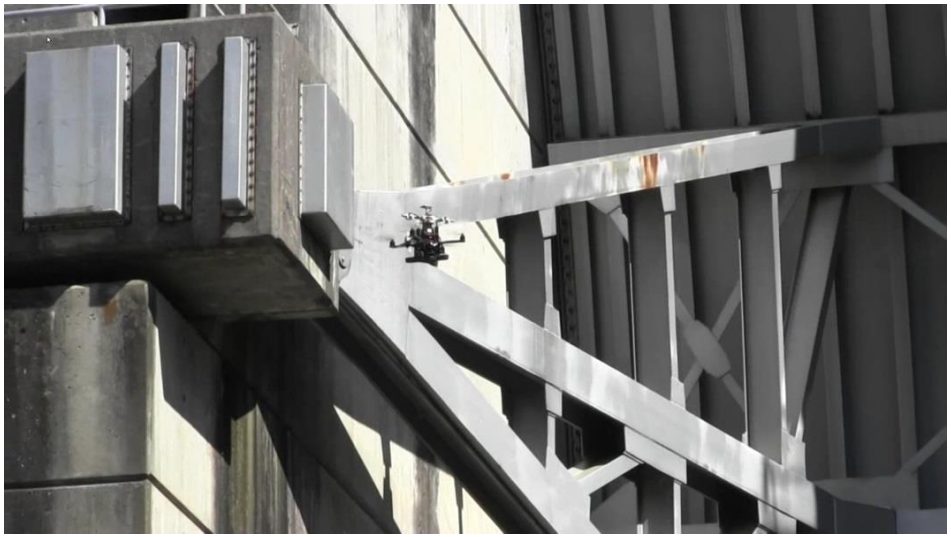
Search and Rescue



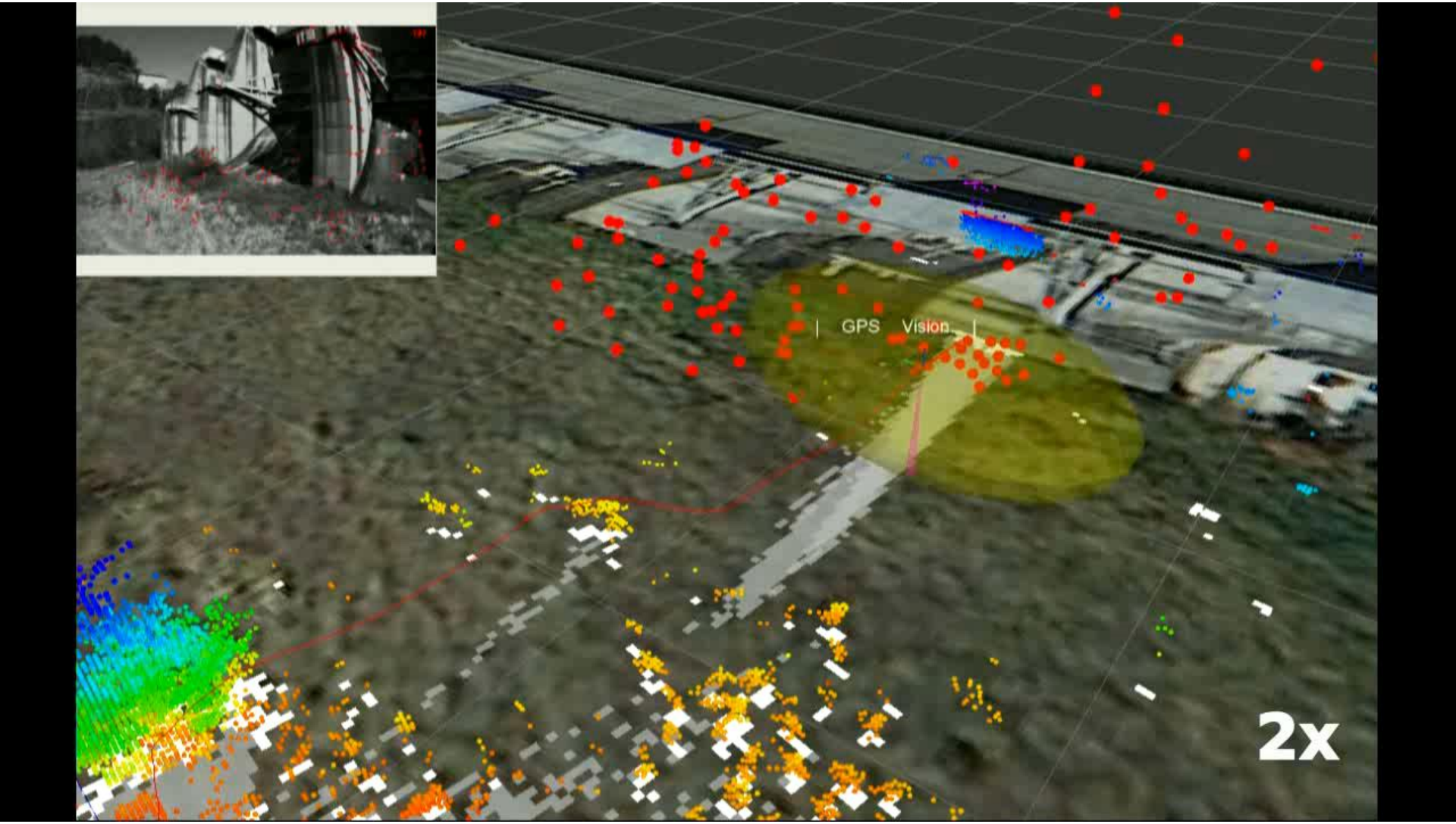
Michael, et al, 2012

Dam Inspection

Emergency flooding gates at the Carter's Dam, GA, USA



Dam Inspection



Shen, et al, 2014

Dam Inspection

Penstock at the Carter's Dam, GA, USA

Dam Inspection

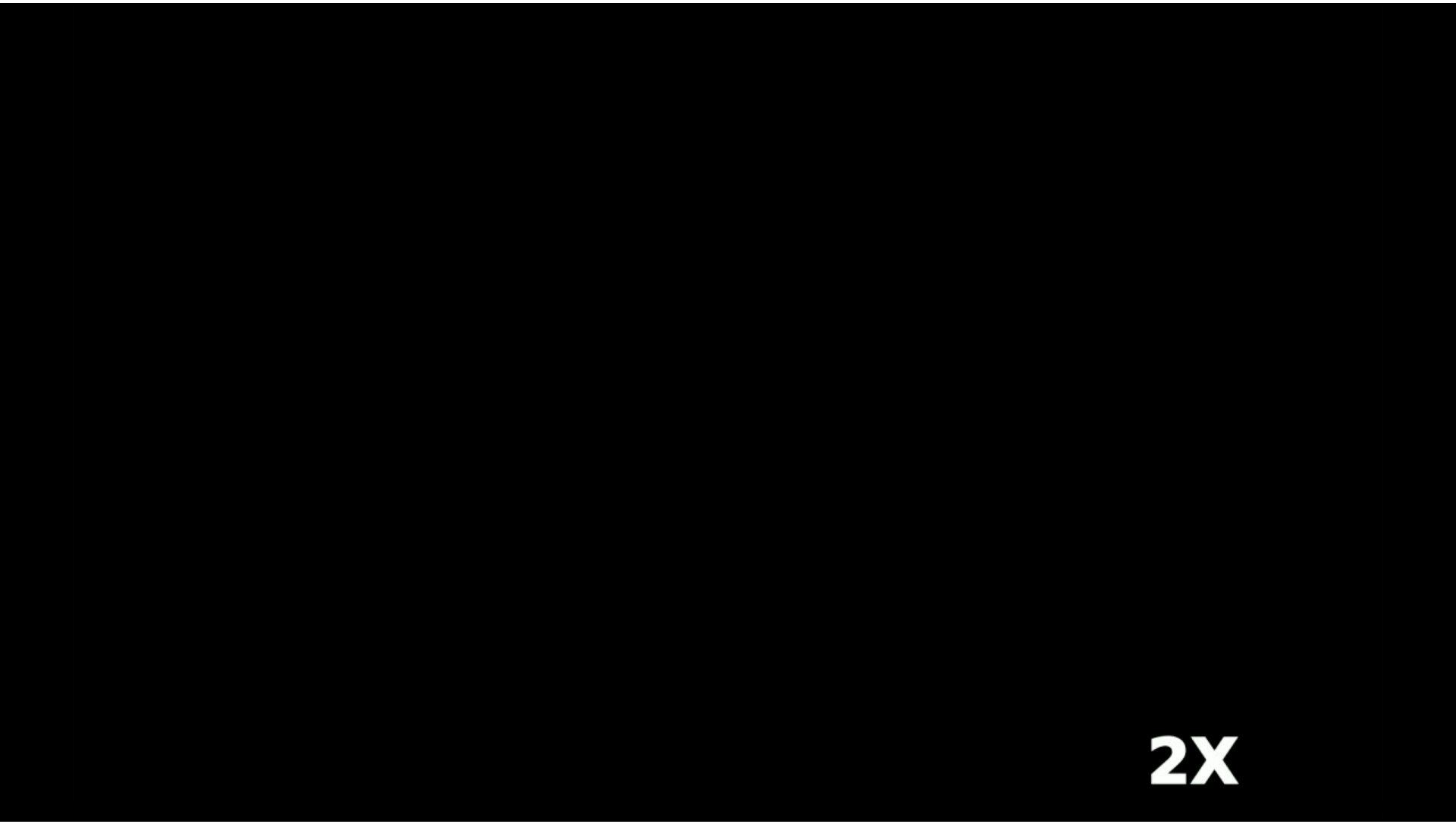
Ozaslan, et al, 2014

Cellular Tower Inspection

Precision Farming

E.&J. Gallo Winery, CA, USA

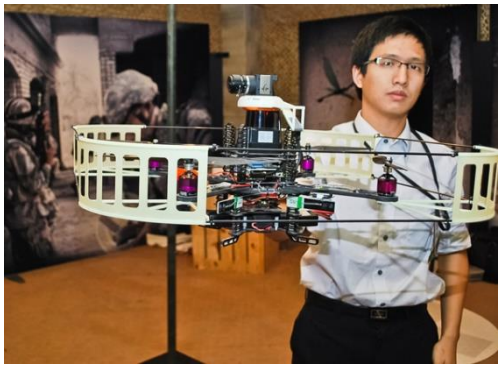
Precision Farming



Shen, et al, 2014

Delivery

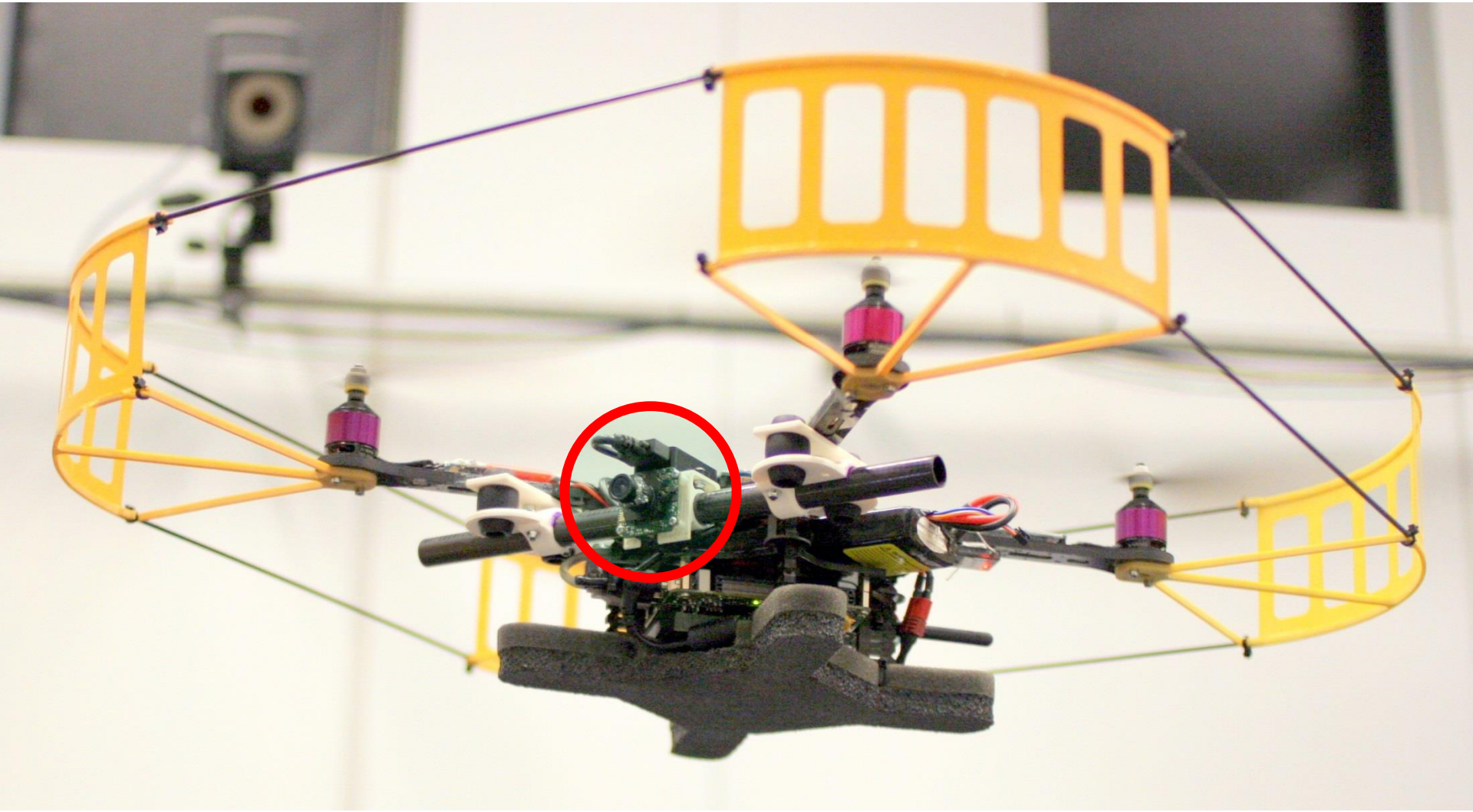
Small and Autonomous



Processing: Not quite there yet, but remember the **Moore's Law!**

Sensing: **YES!**

Minimum Sensing: 1 Camera + 1 IMU



Minimum Sensing: 1 Camera + 1 IMU

Autonomous Aerial Navigation Using Monocular Visual-Inertial Fusion

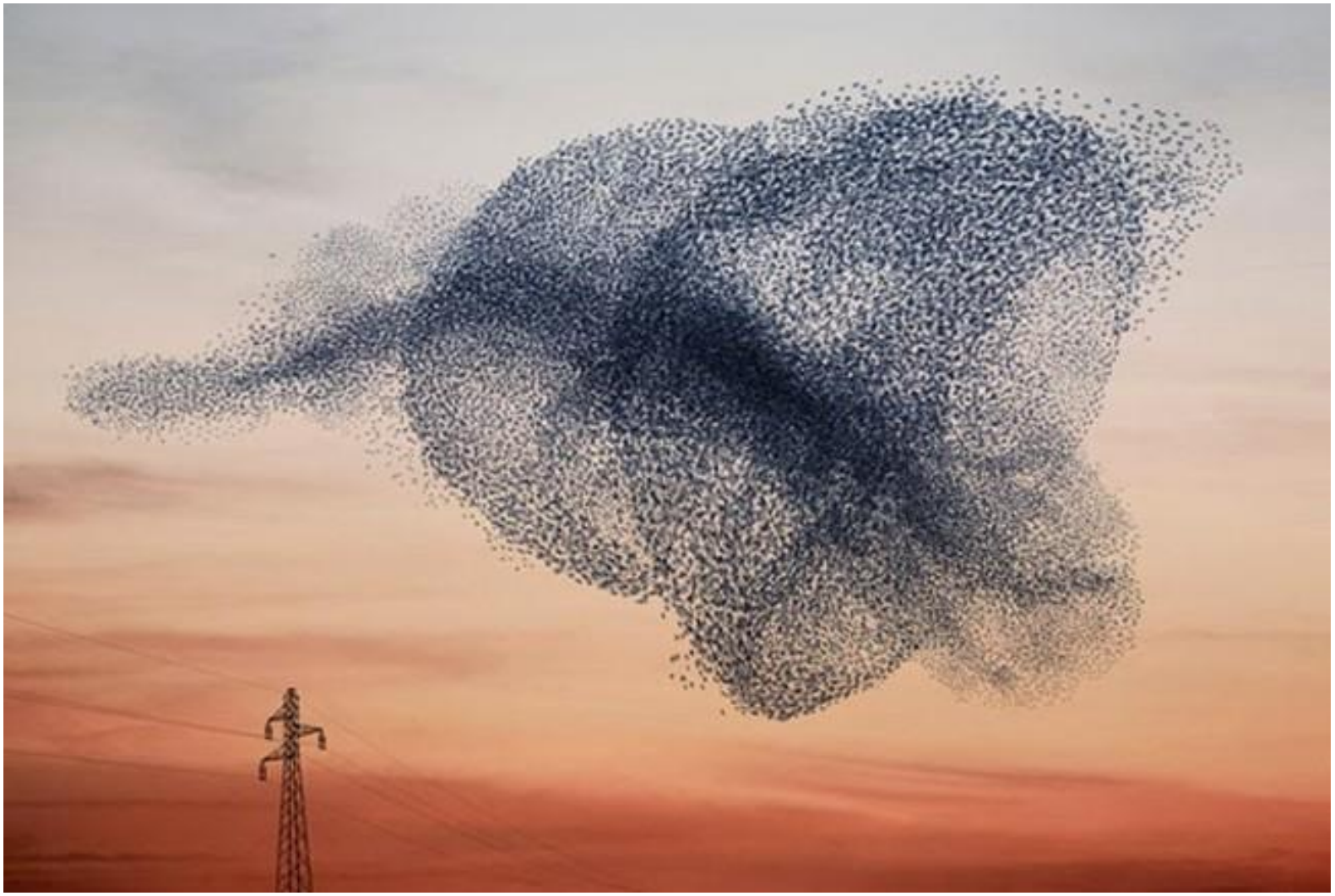
Yi Lin, Fei Gao, Tong Qin, Wenliang Gao,
Tianbo Liu, William Wu, Zhenfei Yang and Shaojie Shen

香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

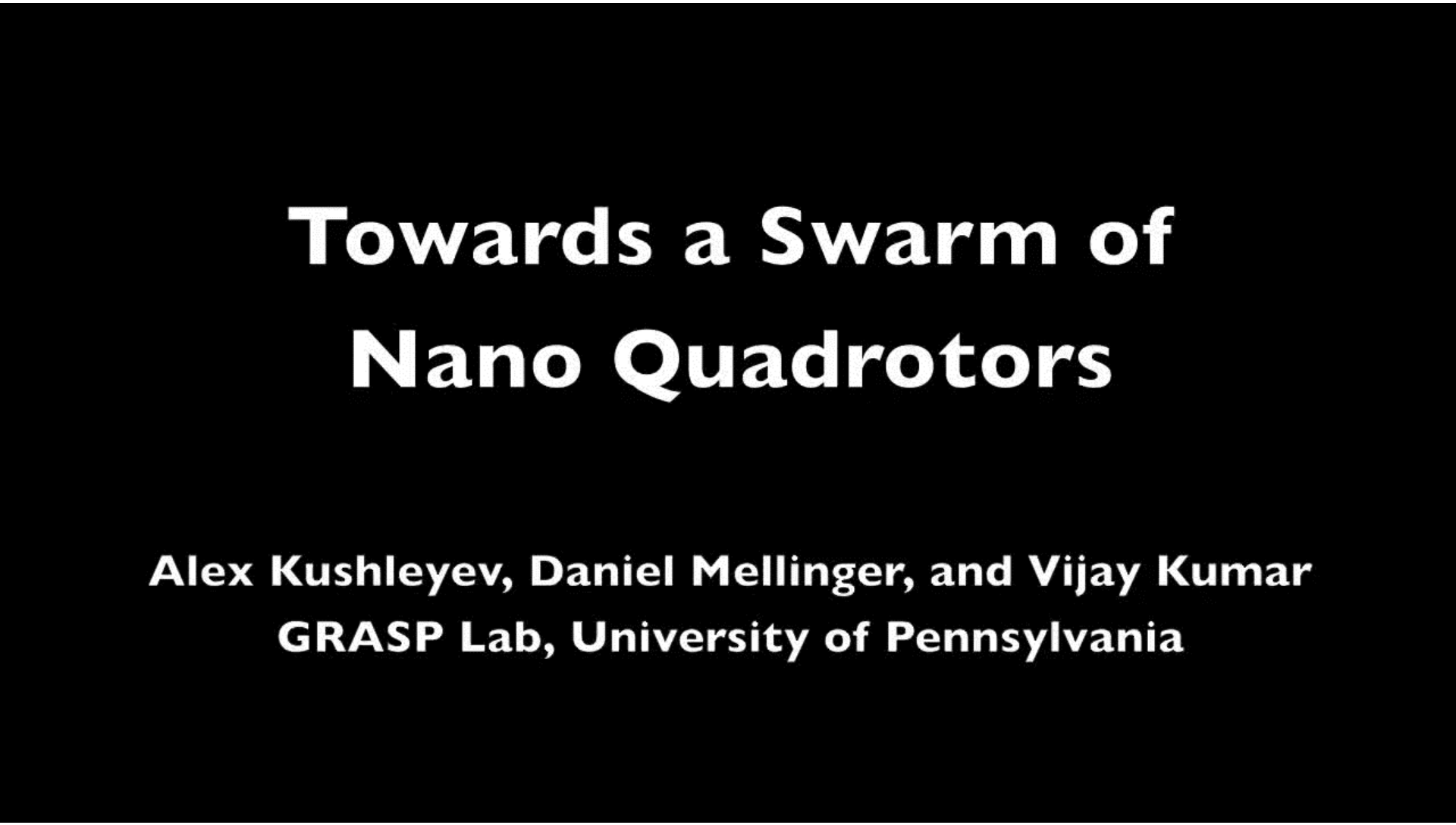
High resolution video available at:
http://ece.ust.hk/~eeshaojie/jfr2017yi.mp4

Lin, et al, 2017

Swarm



Swarm



Mellinger, et al, 2013

Amazing In Motion

Kmel Robotics, 2014

Course Outline

Course Outline

- Dynamics, Planning & Control
- Vision
- Estimation

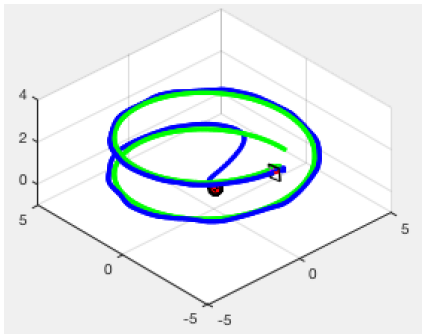
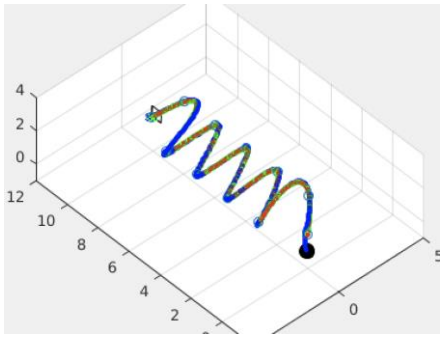
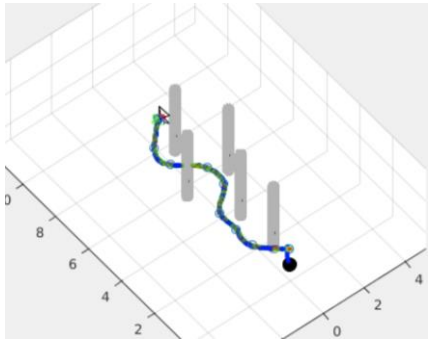
Week	Lecture Date Tus 13:30-16:20	Topic	Assignment (due 11:59 PM on Friday of the corresponding week)	Lab Wed 18:00-20:50 Thu 13:30-16:20
1	2/7	Introduction		No Lab
2	2/14	Rigid Body Transformation Quadrotor Modeling		No Lab
3	2/21	Control Basics Quadrotor Control Trajectory Generation	Project 1 Phase 1 Out	No Lab
4	2/28	Trajectory Generation Path Planning	Project 1 Phase 1 Due Project 1 Phase 2 Out	No Lab
5	3/7	Camera Modeling & Calibration Feature Detection & Matching	Project 1 Phase 2 Due Project 1 Phase 3 Out	Lab Tutorial 1: Robot Assembly
6	3/14	Midterm Exam	Project 1 Phase 3 Due Project 1 Phase 4 Out	Lab Tutorial 2: Prepare P1P4
7	3/21	Multi-View Geometry Pose Estimation	Project 2 Phase 1 Out	Free Lab Time
8	3/28	Optical Flow Dense Stereo	Project 1 Phase 4 Due Project 2 Phase 1 Due Project 2 Phase 2 Out	Free Lab Time
9	4/4	Probability Basics Bayesian Inferencing Kalman Filter	Project 2 Phase 2 Due Project 3 Phase 1 Out	No Lab
10	4/11	Midterm Break		No Lab
11	4/18	Extended Kalman Filter Augmented State EKF Particle Filter	Project 3 Phase 1 Due Project 3 Phase 2 Out	No Lab
12	4/25	SLAM	Project 3 Phase 2 Due Project 3 Phase 3 Out	Lab Tutorial 3: Prepare P3P3
13	5/2	x		Free Lab Time
14	5/9	x	Project 3 Phase 3 Due	Free Lab Time

Course Outline

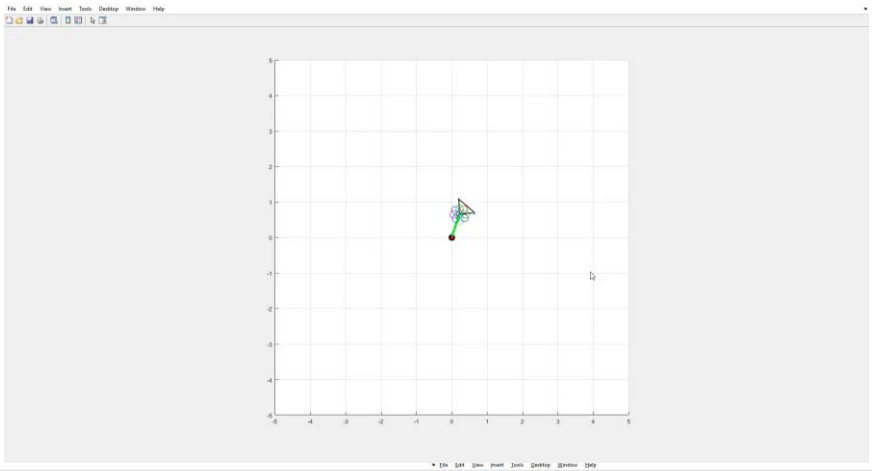
- Dynamics, Control, and Planning
  - Understand 3D rigid body transformation
  - Derive the model of quadrotor aerial robots
  - Able to hover the quadrotor given perfect localization
  - Able to generate and track smooth trajectories given user-defined waypoints
  - Able to generate smooth and safe paths and trajectories given obstacle locations
  - Project 1
- Vision
  - Understand how cameras capture the world
  - Understand feature-based and optical flow-based image processing pipeline
  - Able to use only images to compute camera position, orientation, and velocity
  - Project 2
- Estimation
  - Understand how to incorporate probability principles into robotics
  - Able to integrate heterogeneous noisy measurements to get robust estimates of the platform motion
  - A light touch of simultaneous localization and mapping (SLAM)
  - Project 3

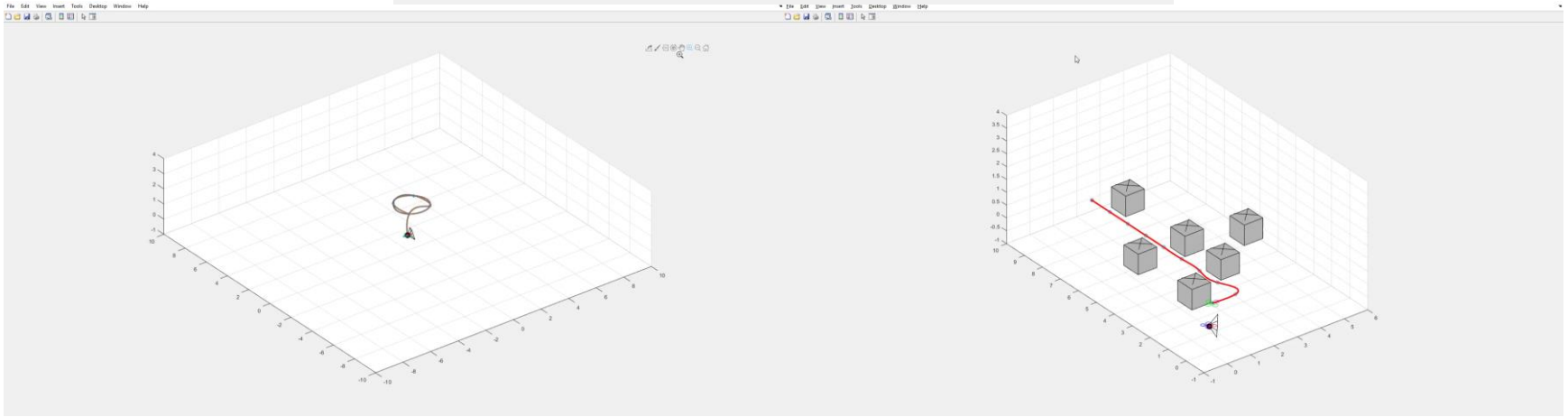
Project 1

- Phase 1: Hovering and trajectory tracking of a simulated quadrotor
  - MATLAB, offline
- Phase 2: Trajectory generation and tracking of a simulated quadrotor
  - MATLAB, offline
- Phase 3: Obstacle avoidance using A* path planning and smooth trajectory generation
  - MATLAB, offline
- Phase 4: Fly the robot!

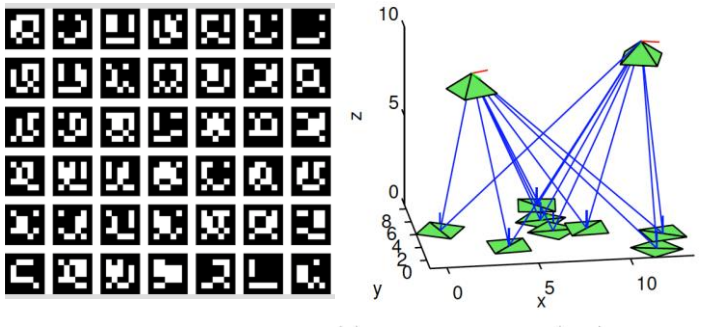

Project 1 in Past Years





Project 2

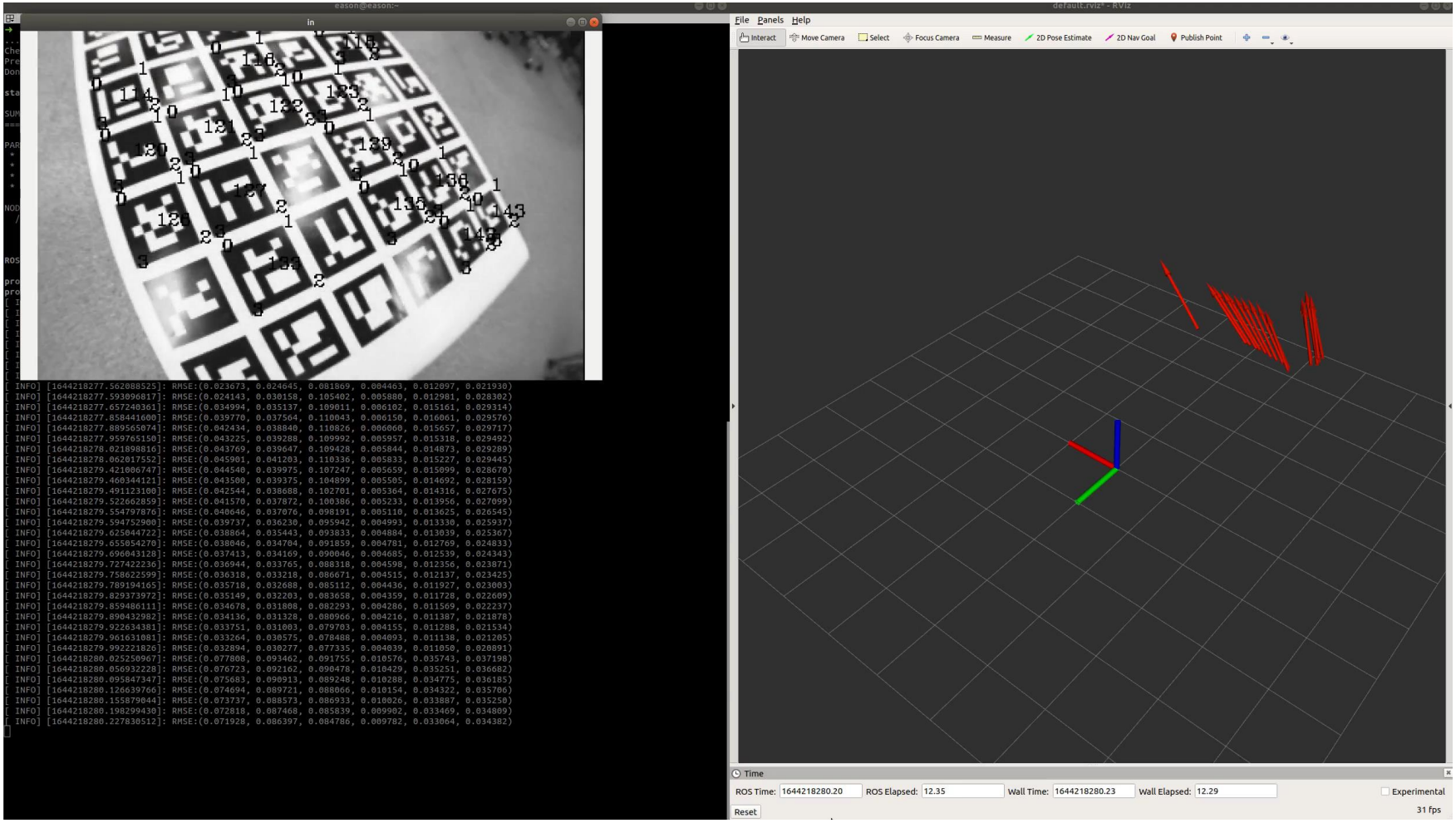
- Phase 1: PnP-based localization on marker map
  - ROS & C++, Desktop PC, offline



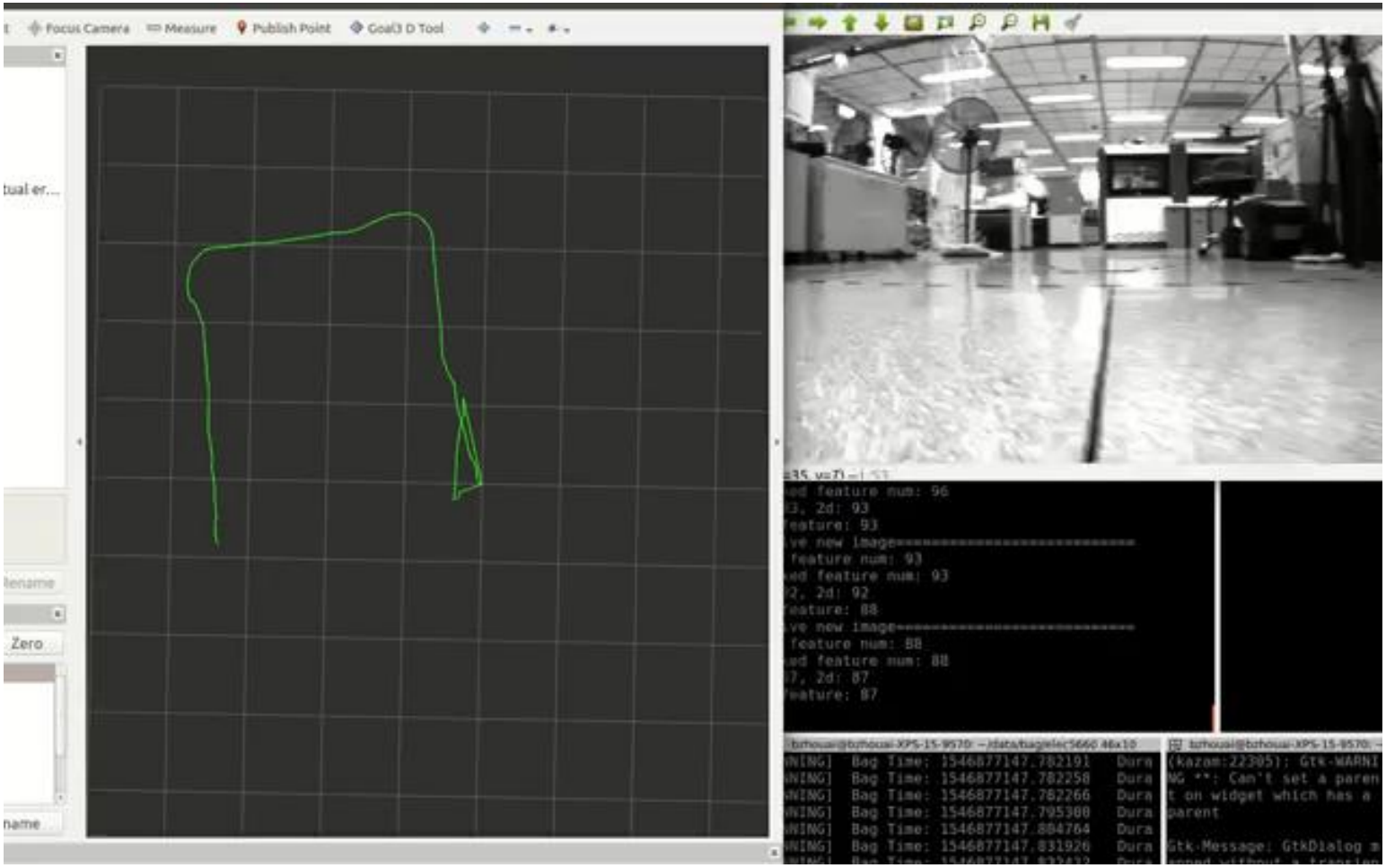
- Phase 2: Visual odometry in markerless environment
  - ROS & C++, Desktop PC, offline

(b)

Project 2 Phase 1 in Past Years

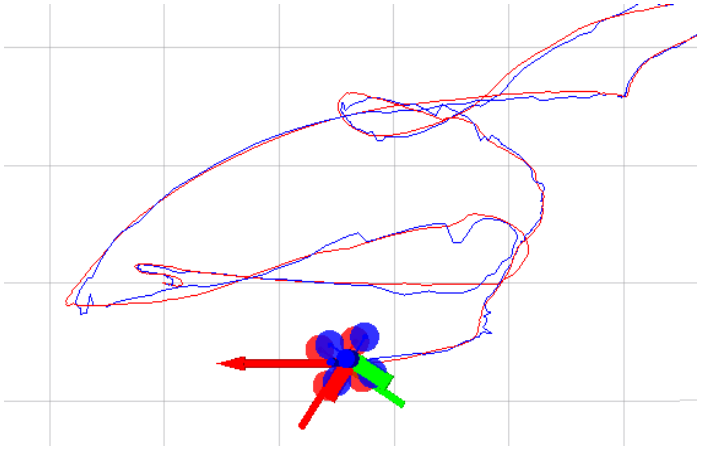


Project 2 Phase 2 in Past Years

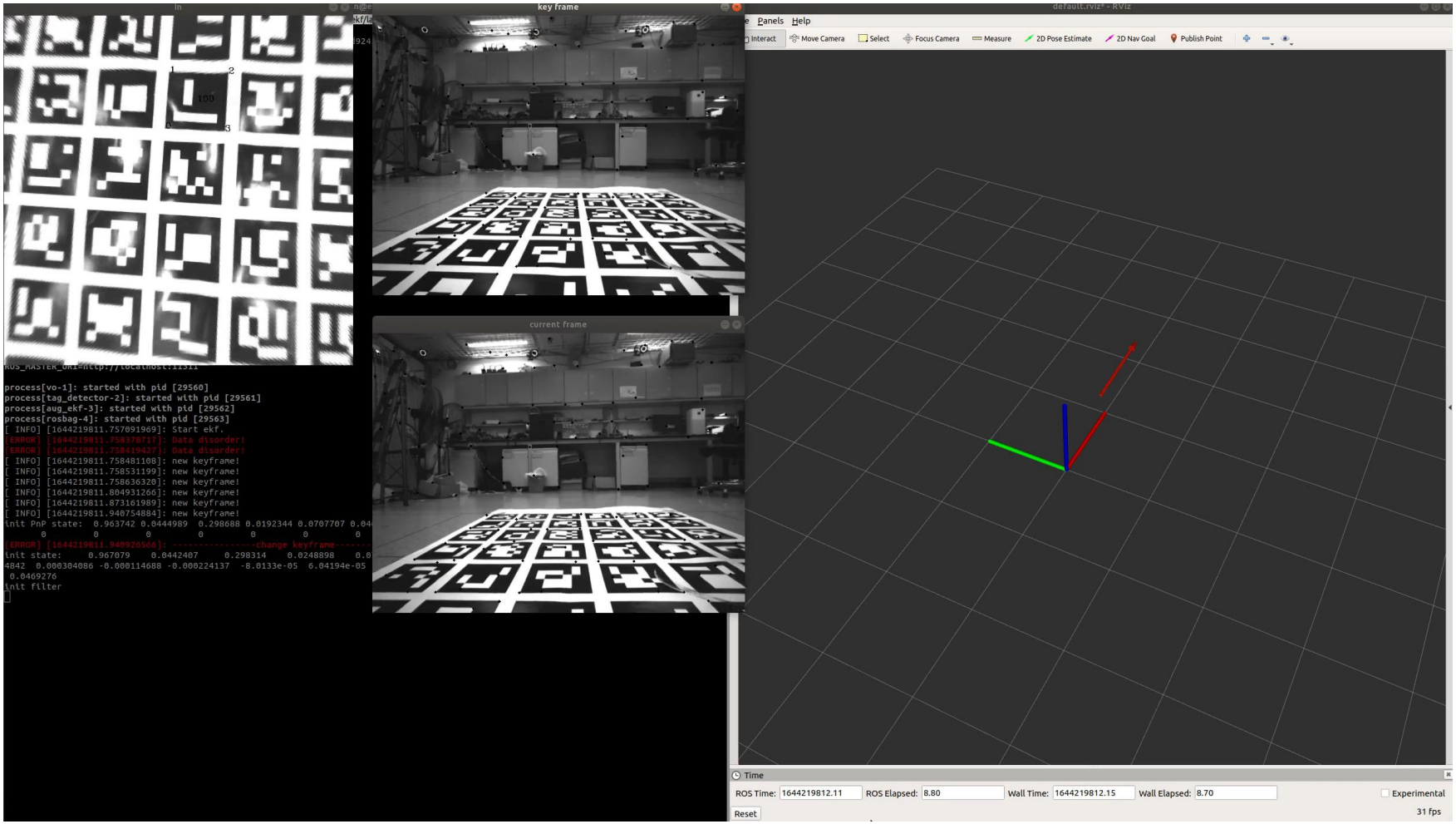


Project 3

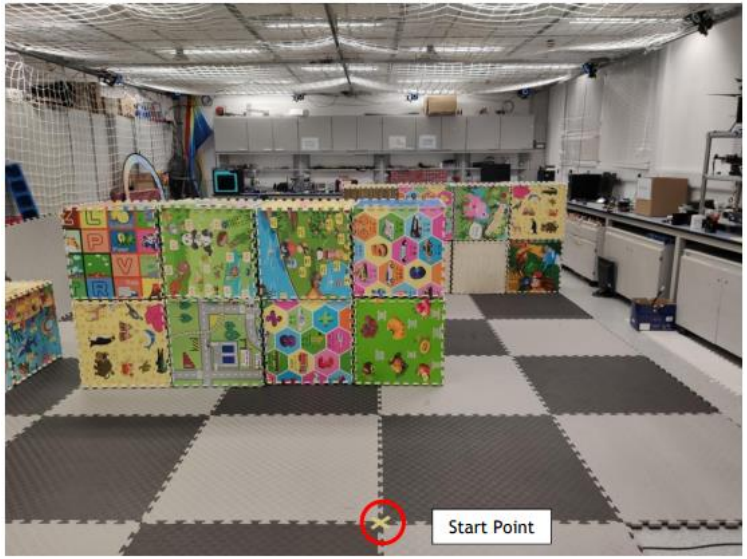
- Phase 1: EKF-based sensor fusion of IMU and tag-based position estimation
  - ROS & C++, Desktop PC, offline
- Phase 2: Augmented state EKF for fusion of IMU and keyframe-based visual odometry together with the PnP localization on markers
  - ROS & C++, Desktop PC, offline
- Phase 3: Fly the robot!

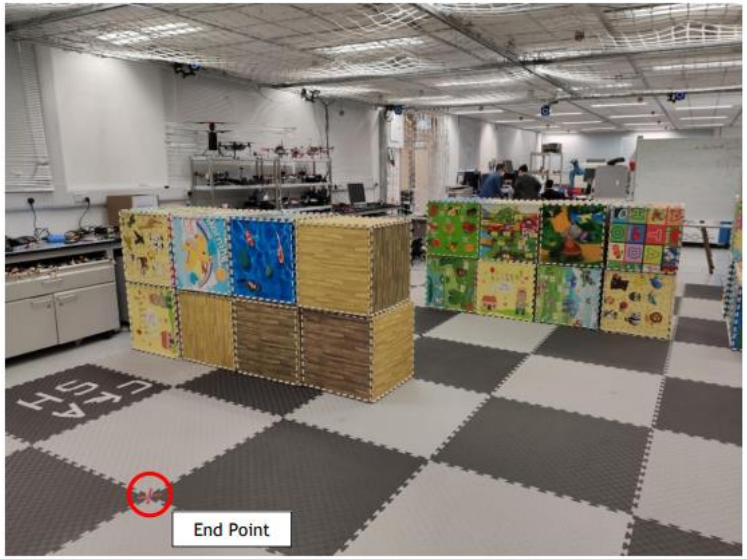


Project 3 in Past Years

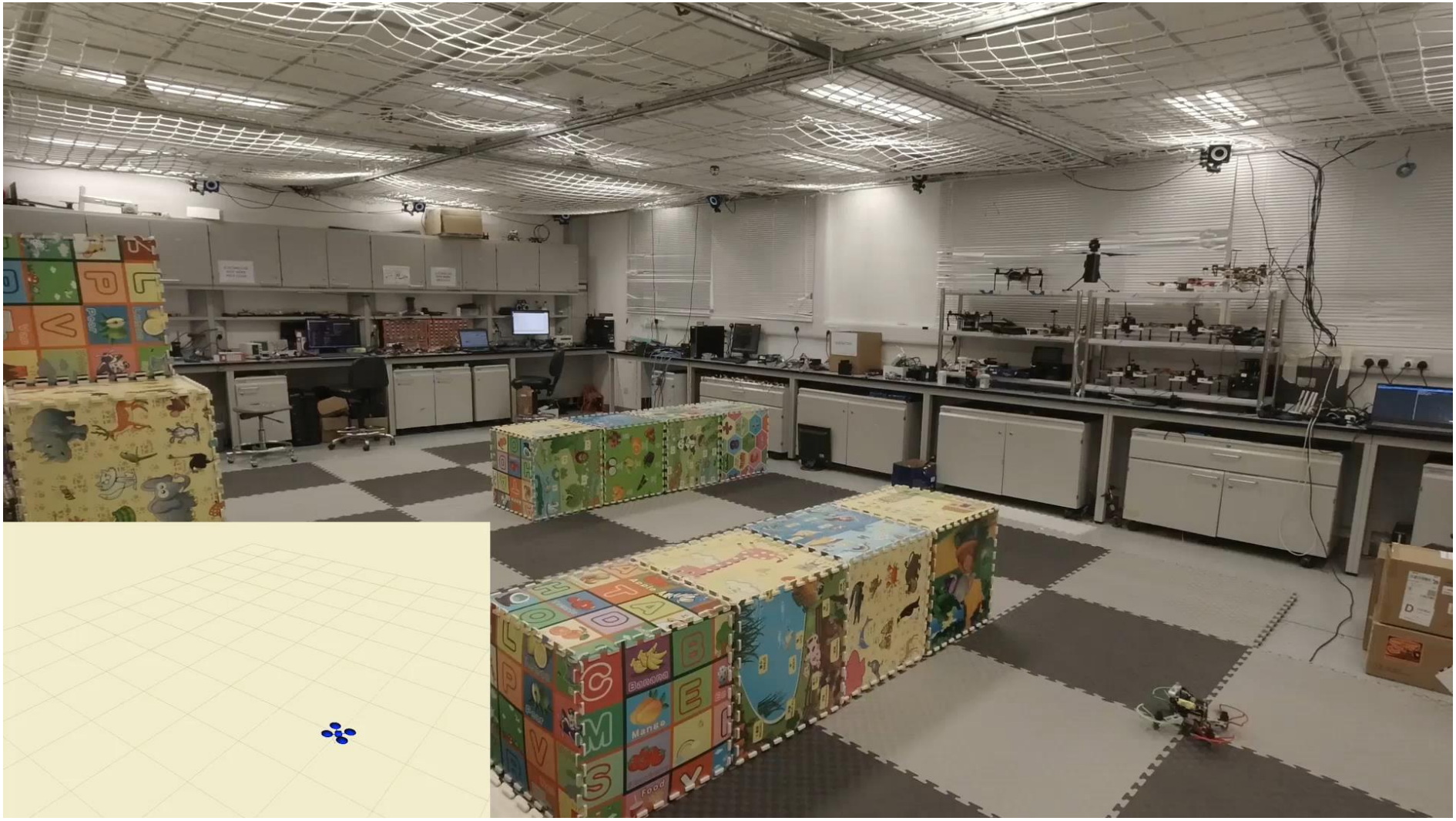


Project 3 in Past Years





Project 3 in Past Years



Next Lecture…

- Rigid Body Transformations
- Rotational Motions
- Rotation Representations
- Rigid Body Motions
- Rigid Body Velocities
- Quadrotor Dynamics

Introduction to Aerial Robotics
Lecture 2

Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

14 February 2023

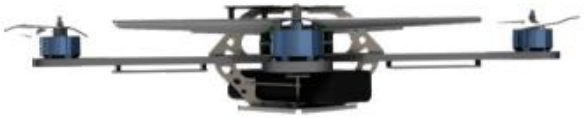
Outline

- Rigid Body Transformations
- Rotational Motions
- Rotation Representations
- Rigid Body Motions
- Rigid Body Velocities
- Quadrotor Dynamics

Rigid Body Transformations

Rigid Body

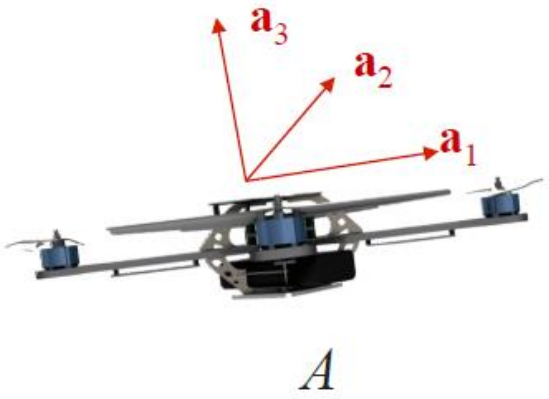
- Two distinct positions and orientations of the same rigid body
  - Let $\mathbf{p}$ and $\mathbf{q}$ be two points on a rigid body
  - $\|\boldsymbol{p}(t) - \boldsymbol{q}(t)\| = \|\boldsymbol{p}(0) - \boldsymbol{q}(0)\| = \text{constant}$

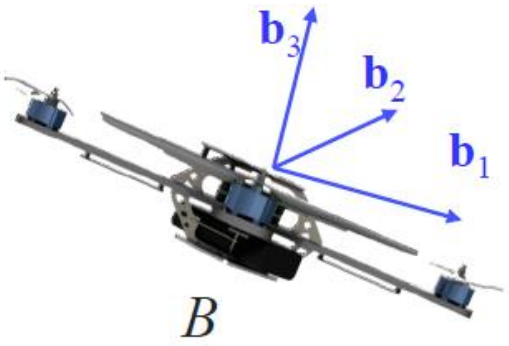


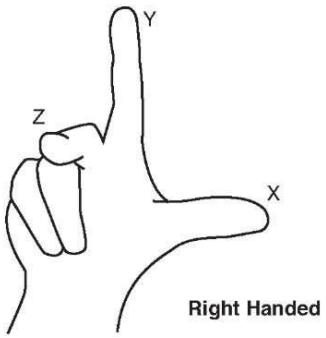
Courtesy Vijay Kumar

Reference Frames

- We associate any position and orientation with a reference frame
  - We use **right-handed** coordinate frames
  - We can find three linearly independent vectors $\mathbf{a}_1$, $\mathbf{a}_3$, $\mathbf{a}_3$ that are basis vectors for reference frame A
  - We can write any vector as a linear combination of basis vectors in either frame $\boldsymbol{v} = \boldsymbol{v_1}\mathbf{a_1} + v_2\mathbf{a}_2 + v_3\mathbf{a}_3$







Notation

- Vectors
  - $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{a}$, …
- Matrices
  - $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, …
- Reference frames
  - A , B, C, …
  - a , b , c, …
- Transformations
  - ${}^{A}\mathbf{A}_{B}$, ${}^{A}\mathbf{R}_{B}$ …
  - $\mathbf{A}_{ab}$, $\mathbf{R}_{ab}$ …
  - $g_{ab}(.)$, $h_{ab}(.)$ …

Be Aware of **Potential Confusion!!!**

Rigid Body Displacement

- Object:

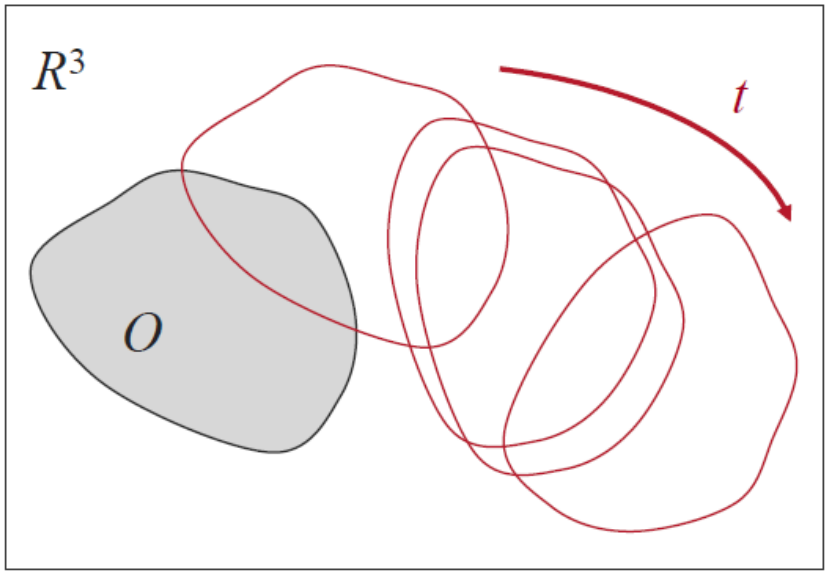
$$O \in \mathbb{R}^3$$

- Rigid body displacement
  - Map

$$g: O \longrightarrow \mathbb{R}^3$$

- Rigid body motion
  - Continuous family of maps

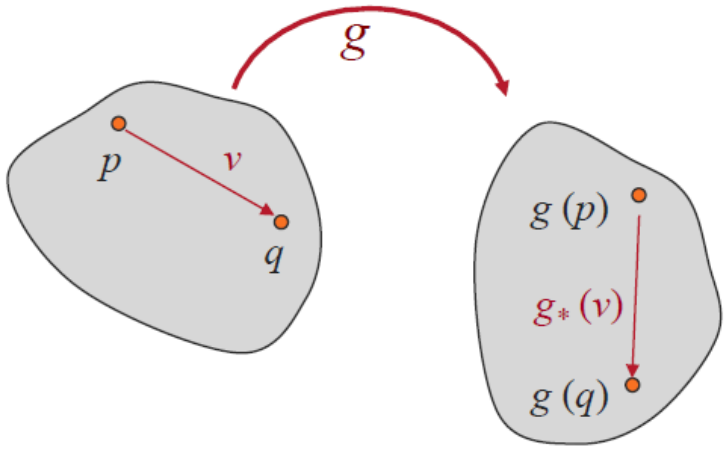
$$g(t): O \longrightarrow \mathbb{R}^3$$



Rigid Body Displacement

- A displacement of a transformation of points
  - Transformation (g) of points induces an action (g*) on vectors

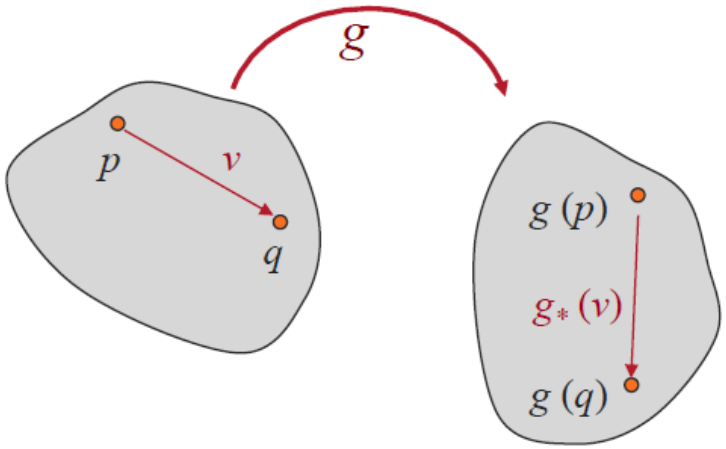
$$g_*(\mathbf{v}) = g(\mathbf{q}) - g(\mathbf{p})$$



Definition of Rigid Body Displacement

- Lengths are preserved

$$\|g(\mathbf{q}) - g(\mathbf{p})\| = \|\mathbf{q} - \mathbf{p}\|$$



- Cross products are preserved

$$g_*(\mathbf{v}) \times g_*(\mathbf{w}) = g_*(\mathbf{v} \times \mathbf{w})$$

Why?

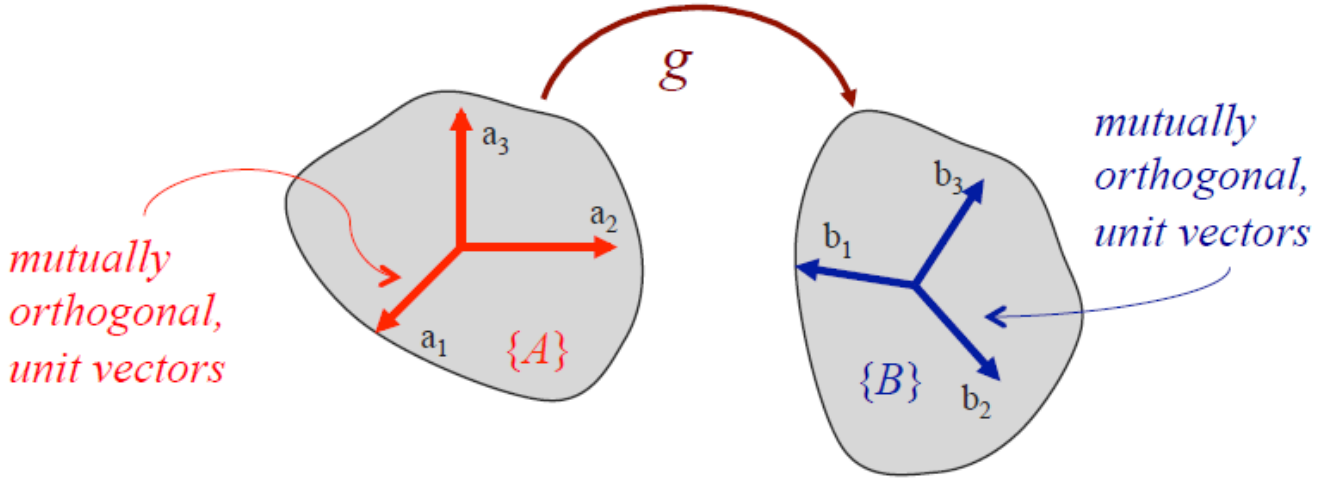
Eliminate internal reflection: $(x, y, z) \rightarrow (x, y, -z)$

Properties of Rigid Body Displacement

- Inner products are also preserved

$$g_*(\mathbf{v}) \cdot g_*(\mathbf{w}) = \mathbf{v} \cdot \mathbf{w}$$

- Orthogonal vectors are mapped to orthogonal vectors



Rigid Body Displacement

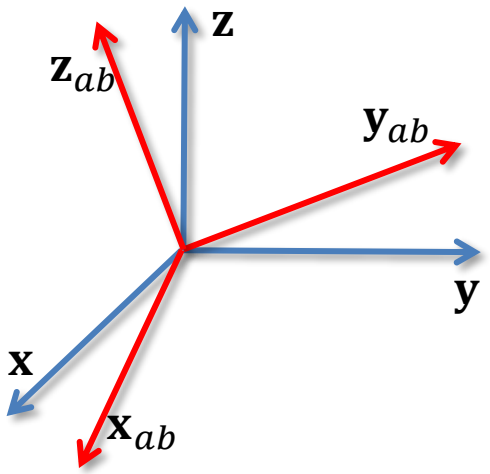
- Rigid body displacements are transformations that satisfy two important properties:
  1. Lengths are preserved
  2. Cross products are preserved

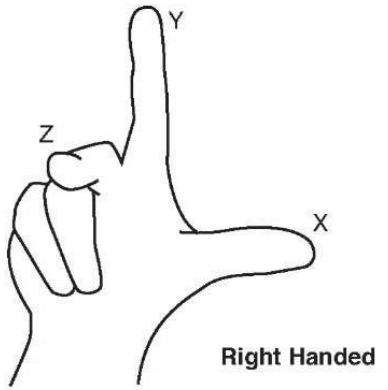
- Rigid body transformations and rigid body displacements are often used interchangeably
  - Transformations generally used to describe relationship between reference frames attached to different rigid bodies.
  - Displacements describe relationships between two positions and orientation of a frame attached to a displaced rigid body

Rotational Motions

Rotation

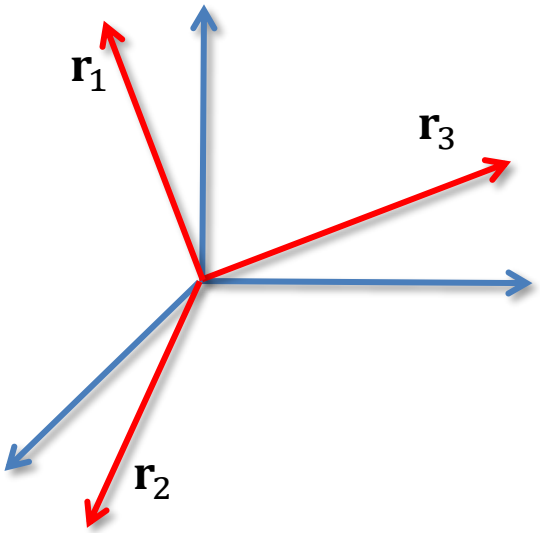
- Coordinate frames are right-handed
- Principle axes of frame *A*:
  - $\mathbf{x} = [1\ 0\ 0]^T$
  - $\mathbf{y} = [0\ 1\ 0]^T$
  - $\mathbf{z} = [0\ 0\ 1]^T$
- Principle axes of frame *B*:
  - $\mathbf{x}_{ab}, \mathbf{y}_{ab}, \mathbf{z}_{ab} \subset \mathbb{R}^3$
- The Rotation Matrix:
  - $\mathbf{R}_{ab} = [\mathbf{x}_{ab}, \mathbf{y}_{ab}, \mathbf{z}_{ab}]$
  - Coordinates of principle axes of *B* related to *A*





Properties of a Rotation Matrix

- Let $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ be a rotation matrix
- Orthogonal:
  - $\mathbf{r}_i{}^T \cdot \mathbf{r}_j = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases}$
  - $\mathbf{R} \cdot \mathbf{R}^T = \mathbf{I}$
- Special orthogonal:
  - $\det\mathbf{R} = \mathbf{r}_1{}^T \cdot (\mathbf{r}_2 \times \mathbf{r}_3) = \mathbf{r}_1{}^T \cdot \mathbf{r}_1 = 1$



- The set of all rotations forms the Special Orthogonal Group
  - Special orthogonal group
  - 3D rotations: $SO(3)$
  - 2D rotations: $SO(2)$
  - $SO(n) = \{\mathbf{R} \in \mathbb{R}^{n \times n} | \mathbf{R} \cdot \mathbf{R}^T = \mathbf{I}, \det\mathbf{R} = 1\}$

Properties of a Rotation Matrix

- $SO(3) = \{\mathbf{R} \in \mathbb{R}^{3\times3} | \mathbf{R} \cdot \mathbf{R}^T = \mathbf{I}, \det\mathbf{R} = 1\}$
- $SO(3)$ is a group under the operation of matrix multiplication
  1. Closure: If $\mathbf{R}_1, \mathbf{R}_2 \in SO(3)$, then $\mathbf{R}_1 \cdot \mathbf{R}_2 \in SO(3)$
  2. Identity: The identity matrix is the identity element
  3. Inverse: If $\mathbf{R} \in SO(3)$, then $\mathbf{R}^{-1} \in SO(3)$
  4. Associativity: $\mathbf{R}_1 \cdot (\mathbf{R}_2 \cdot \mathbf{R}_3) = (\mathbf{R}_1 \cdot \mathbf{R}_2) \cdot \mathbf{R}_3$

$(G,\cdot)$ is a group if:

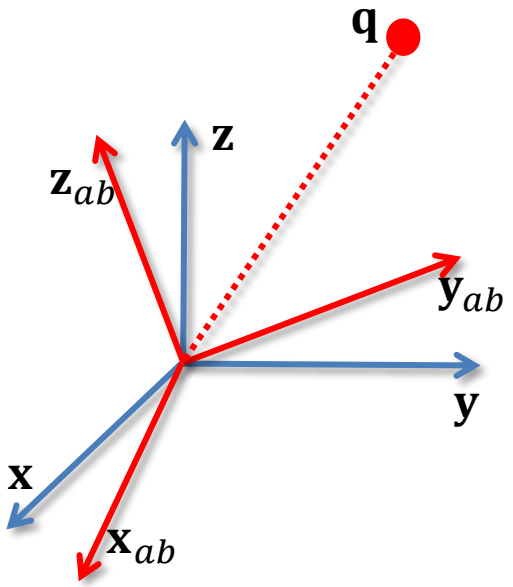
1) $g_1, g_2 \in G \Rightarrow g_1 \cdot g_2 \in G$
2) $\exists!\, e \in G,\ s.t.\ g \cdot e = e \cdot g = g,\ \forall g \in G$
3) $\forall g \in G,\ \exists!\ g^{-1} \in G,\ s.t.\ g \cdot g^{-1} = g^{-1} \cdot g = e$
4) $g_1 \cdot (g_2 \cdot g_3) = (g_1 \cdot g_2) \cdot g_3$

Group examples:

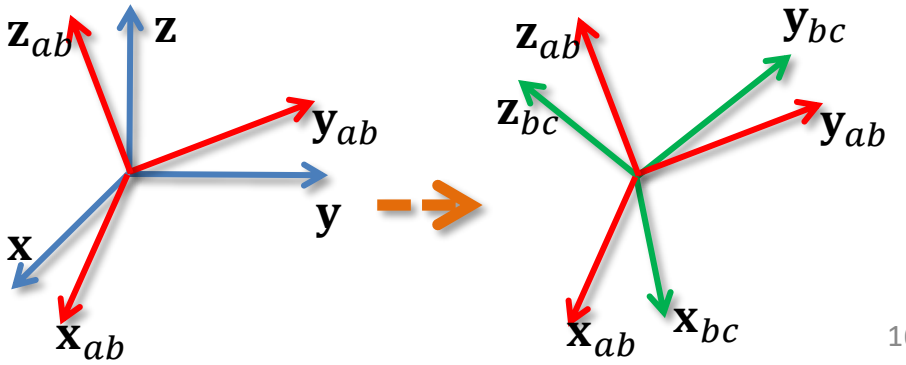
1. The set of all integers with addition operation
2. The set of all real numbers with arithmetic operations

Properties of a Rotation Matrix

- A transformation that rotates the coordinates of a point from frame *B* to frame *A*
  - Let $\mathbf{q}_b = [x_b, y_b, z_b]^T \in \mathbb{R}^3$ be coordinate of point $\mathbf{q}$ in frame *B*
  - Let $\mathbf{q}_a$ be coordinate of point $\mathbf{q}$ in frame *A*
  - $\mathbf{q}_a = x_b \cdot \mathbf{x}_{ab} + y_b \cdot \mathbf{y}_{ab} + z_b \cdot \mathbf{z}_{ab} =$ $[\mathbf{x}_{ab}, \mathbf{y}_{ab}, \mathbf{z}_{ab}] \begin{bmatrix} x_b \\ y_b \\ z_b \end{bmatrix} = \mathbf{R}_{ab} \cdot \mathbf{q}_b$



- Composition Rule
  - $\mathbf{R}_{ac} = \mathbf{R}_{ab} \cdot \mathbf{R}_{bc}$

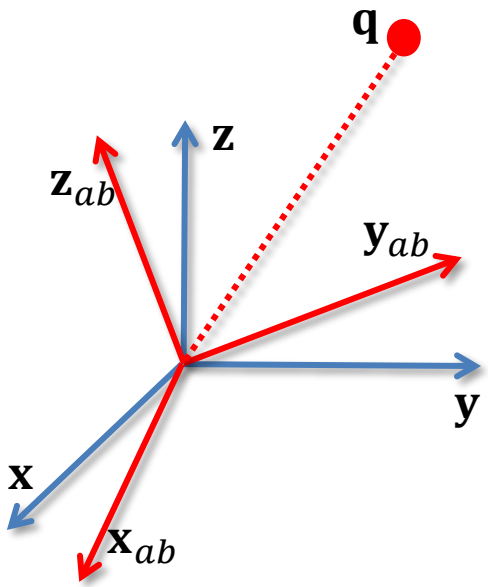


Rotation is Rigid Body Transformation

$\mathbf{R}_{ab} = [\mathbf{x}_{ab}, \mathbf{y}_{ab}, \mathbf{z}_{ab}]$ preserves:

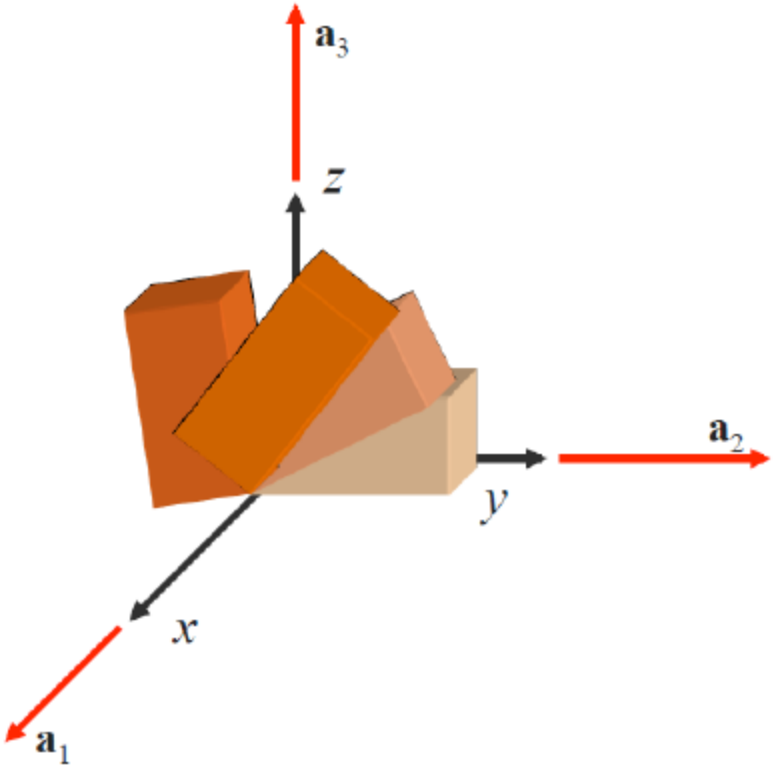
- Length:
  - $\|\mathbf{R}_{ab}(\mathbf{p}_b - \mathbf{q}_b)\| = \|\mathbf{p}_b - \mathbf{q}_b\|$

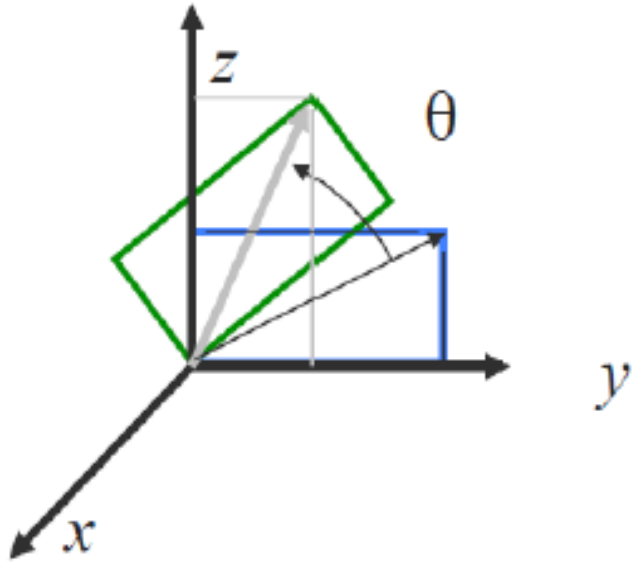
- Cross product:
  - $\mathbf{R}_{ab}(\mathbf{v} \times \boldsymbol{w}) = (\mathbf{R}_{ab}\mathbf{v}) \times (\mathbf{R}_{ab}\mathbf{w})$
  - Use the fact $\mathbf{R}(\mathbf{v})^{\wedge}\mathbf{R}^T = (\mathbf{R}\mathbf{v})^{\wedge}$ to prove, where $(\mathbf{a})^{\wedge} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$ is the skew-symmetric matrix, and $\mathbf{a} \times \mathbf{b} = (\mathbf{a})^{\wedge}\mathbf{b}$



Example - Rotation

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{bmatrix}$$

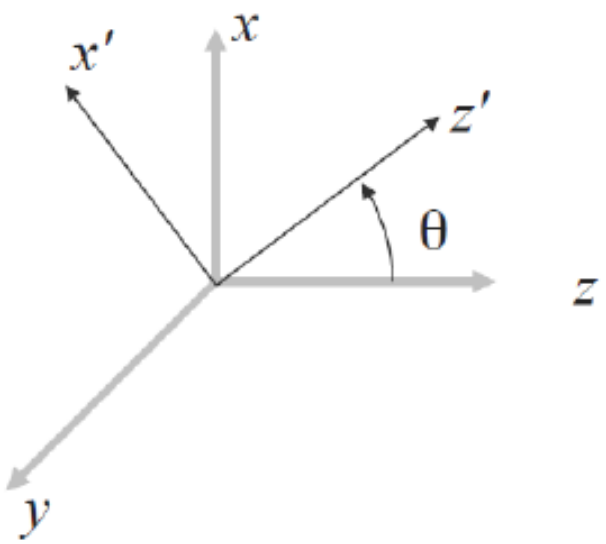


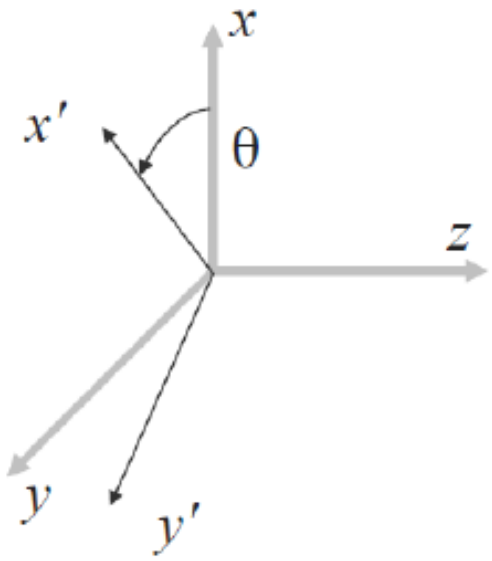


Example - Rotation

$$\mathbf{R}_y(\theta) = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix}$$

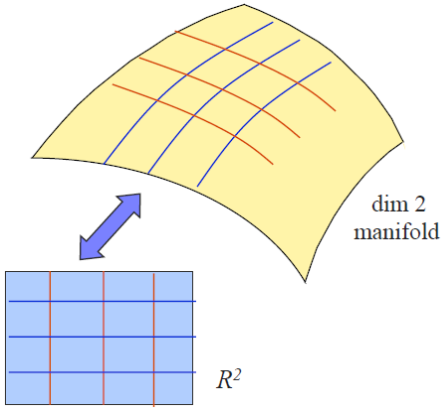
$$\mathbf{R}_z(\theta) = \begin{bmatrix} cos\theta & -sin\theta & 0 \\ sin\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

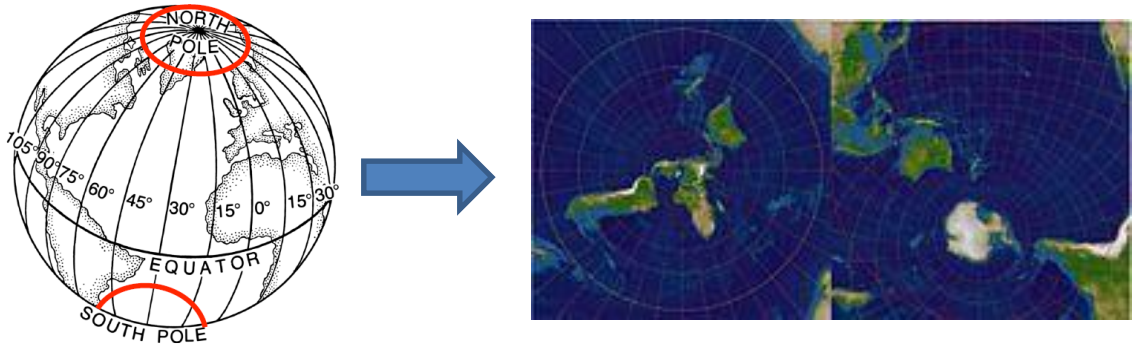


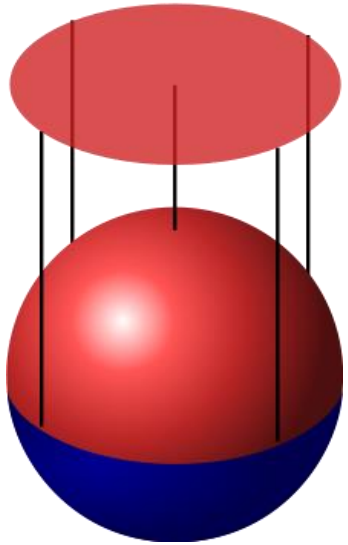


Properties of Rotation

- $SO(3)$ is a continuous group
  - The multiplication operation is a continuous operation
  - The inverse is a continuous function
- $SO(3)$ is a smooth manifold
  - A manifold of dimension n is a set M which is locally resembled to Euclidean space $\mathbb{R}^n$ near each point
  - Example: sphere is a differentiable manifold that is locally resembled to $\mathbb{R}^2$







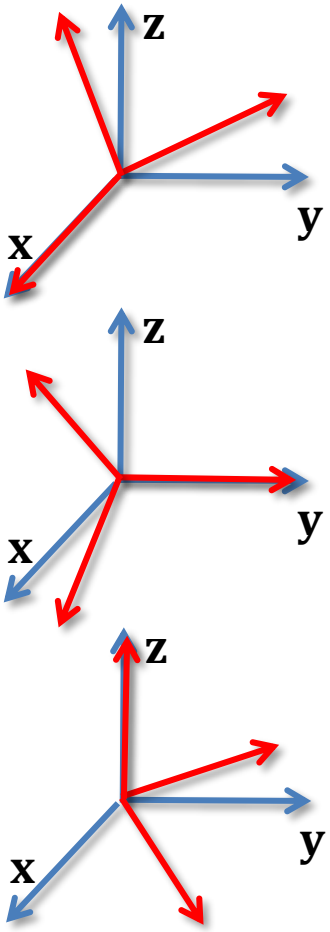
Rotation Representations

Rotation Representations

- Rotation matrices
- Euler angles
- Exponential coordinates
- Quaternions
  - Will be discussed later in the semester
  - Slides are provided as appendix for this lecture

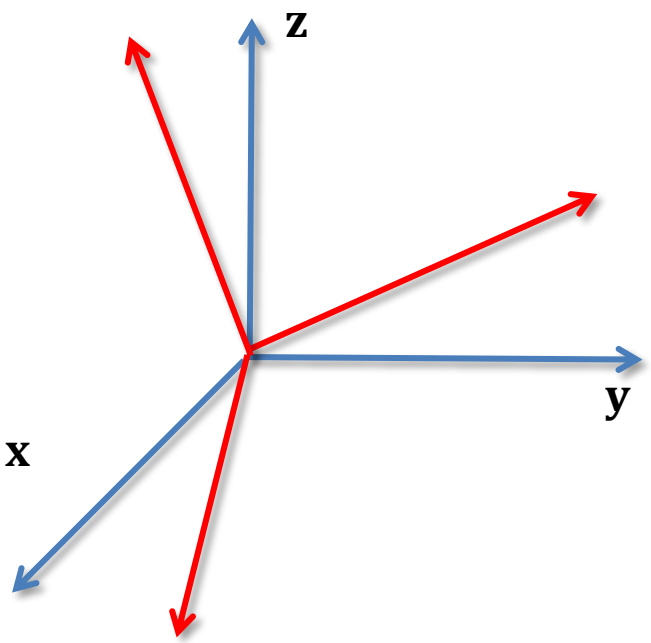
Euler Angles

- Elementary rotations:
  - $\boldsymbol{R}_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$
  - $\boldsymbol{R}_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$
  - $\boldsymbol{R}_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$



Euler Angles

- Any rotation can be described by three successive rotations about linear independent axes
- However, this is an almost 1-1 transform with singularities:
  - $\boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_x(\phi) \cdot \boldsymbol{R}_y(\theta) \Rightarrow \boldsymbol{R}$
  - $\boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_x(\phi) \cdot \boldsymbol{R}_y(\theta) \nLeftarrow \boldsymbol{R}$



Euler Angles

- Different Euler angle conversions:

Proper Euler angles		Tait-Bryan angles	
$X_1Z_2X_3 = \begin{bmatrix} c_2 & -c_3s_2 & s_2s_3 \\ c_1s_2 & c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 \\ s_1s_2 & c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$		$X_1Z_2Y_3 = \begin{bmatrix} c_2c_3 & -s_2 & c_2s_3 \\ s_1s_3 + c_1c_3s_2 & c_1c_2 & c_1s_2s_3 - c_3s_1 \\ c_3s_1s_2 - c_1s_3 & c_2s_1 & c_1c_3 + s_1s_2s_3 \end{bmatrix}$	
$X_1Y_2X_3 = \begin{bmatrix} c_2 & s_2s_3 & c_3s_2 \\ s_1s_2 & c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 \\ -c_1s_2 & c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$		$X_1Y_2Z_3 = \begin{bmatrix} c_2c_3 & -c_2s_3 & s_2 \\ c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 & -c_2s_1 \\ s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 & c_1c_2 \end{bmatrix}$	
$Y_1X_2Y_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & s_1s_2 & c_1s_3 + c_2c_3s_1 \\ s_2s_3 & c_2 & -c_3s_2 \\ -c_3s_1 - c_1c_2s_3 & c_1s_2 & c_1c_2c_3 - s_1s_3 \end{bmatrix}$		$Y_1X_2Z_3 = \begin{bmatrix} c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 & c_2s_1 \\ c_2s_3 & c_2c_3 & -s_2 \\ c_1s_2s_3 - c_3s_1 & c_1c_3s_2 + s_1s_3 & c_1c_2 \end{bmatrix}$	
$Y_1Z_2Y_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_1s_2 & c_3s_1 + c_1c_2s_3 \\ c_3s_2 & c_2 & s_2s_3 \\ -c_1s_3 - c_2c_3s_1 & s_1s_2 & c_1c_3 - c_2s_1s_3 \end{bmatrix}$		$Y_1Z_2X_3 = \begin{bmatrix} c_1c_2 & s_1s_3 - c_1c_3s_2 & c_3s_1 + c_1s_2s_3 \\ s_2 & c_2c_3 & -c_2s_3 \\ -c_2s_1 & c_1s_3 + c_3s_1s_2 & c_1c_3 - s_1s_2s_3 \end{bmatrix}$	
$Z_1Y_2Z_3 = \begin{bmatrix} c_1c_2c_3 - s_1s_3 & -c_3s_1 - c_1c_2s_3 & c_1s_2 \\ c_1s_3 + c_2c_3s_1 & c_1c_3 - c_2s_1s_3 & s_1s_2 \\ -c_3s_2 & s_2s_3 & c_2 \end{bmatrix}$		$Z_1Y_2X_3 = \begin{bmatrix} c_1c_2 & c_1s_2s_3 - c_3s_1 & s_1s_3 + c_1c_3s_2 \\ c_2s_1 & c_1c_3 + s_1s_2s_3 & c_3s_1s_2 - c_1s_3 \\ -s_2 & c_2s_3 & c_2c_3 \end{bmatrix}$	
$Z_1X_2Z_3 = \begin{bmatrix} c_1c_3 - c_2s_1s_3 & -c_1s_3 - c_2c_3s_1 & s_1s_2 \\ c_3s_1 + c_1c_2s_3 & c_1c_2c_3 - s_1s_3 & -c_1s_2 \\ s_2s_3 & c_3s_2 & c_2 \end{bmatrix}$		$Z_1X_2Y_3 = \begin{bmatrix} c_1c_3 - s_1s_2s_3 & -c_2s_1 & c_1s_3 + c_3s_1s_2 \\ c_3s_1 + c_1s_2s_3 & c_1c_2 & s_1s_3 - c_1c_3s_2 \\ -c_2s_3 & s_2 & c_2c_3 \end{bmatrix}$	

Euler Angles

- Example: Z-Y-Z Euler angles:
  - Sequence of three rotations about body-fixed axes
  - $\boldsymbol{R} = \boldsymbol{R}_z(\phi) \cdot \boldsymbol{R}_y(\theta) \cdot \boldsymbol{R}_z(\psi)$
  - $\boldsymbol{R} = \begin{bmatrix} c\phi c\theta c\psi - s\phi s\psi & -c\phi c\theta s\psi - s\phi c\psi & c\phi s\theta \\ s\phi c\theta c\psi + c\phi s\psi & -s\phi c\theta s\psi + c\phi c\psi & s\phi c\theta \\ -s\theta c\psi & s\theta s\psi & c\theta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$
- If $\sin\theta \neq 0$:
  - $\theta = \text{acos}(r_{33})$
  - $\psi = \text{atan}\,2(\frac{r_{32}}{\sin\theta}, -\frac{r_{31}}{\sin\theta})$
  - $\phi = \text{atan}\,2(\frac{r_{23}}{\sin\theta}, \frac{r_{13}}{\sin\theta})$

Euler Angles

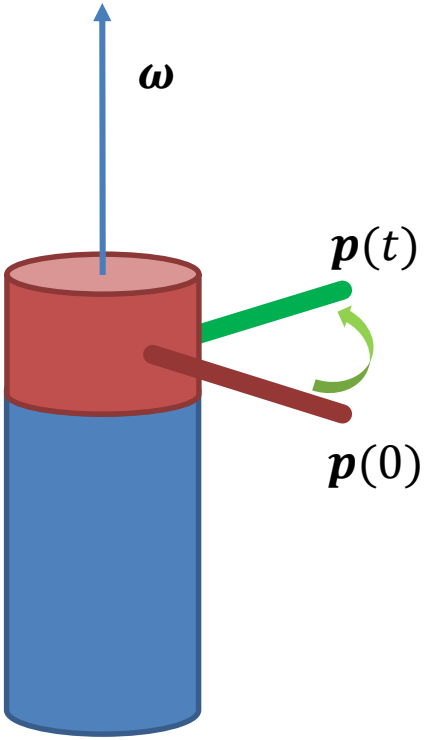
- Example: Z-Y-Z Euler angles:
  - Sequence of three rotations about body-fixed axes
  - $\boldsymbol{R} = \boldsymbol{R}_z(\phi) \cdot \boldsymbol{R}_y(\theta) \cdot \boldsymbol{R}_z(\psi)$
  - $\boldsymbol{R} = \begin{bmatrix} c\phi c\theta c\psi - s\phi s\psi & -c\phi c\theta s\psi - s\phi c\psi & c\phi s\theta \\ s\phi c\theta c\psi + c\phi s\psi & -s\phi c\theta s\psi + c\phi c\psi & s\phi c\theta \\ -s\theta c\psi & s\theta s\psi & c\theta \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$
- If $\sin\theta = 0$:
  - $\boldsymbol{R} = \begin{bmatrix} c\phi c\psi - s\phi s\psi & -c\phi s\psi - s\phi c\psi & 0 \\ c\phi s\psi + s\phi c\psi & -s\phi s\psi + c\phi c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} = \boldsymbol{R}_z(\phi + \psi)$
  - As long as $\phi + \psi$ is preserved, we have infinite set of Euler angles!

Exponential Coordinates

- Scalar differential equation:
  - $\begin{cases} \dot{x}(t) = ax(t) \\ x(0) = x_0 \end{cases} \Longrightarrow x(t) = e^{at}x_0$
- Matrix differential equation:
  - $\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) \\ \boldsymbol{x}(0) = \boldsymbol{x}_0 \end{cases} \Longrightarrow \boldsymbol{x}(t) = e^{\boldsymbol{A}t}\boldsymbol{x}_0$
  - $e^{\boldsymbol{A}} = \boldsymbol{I} + \boldsymbol{A} + \frac{1}{2}\boldsymbol{A}^2 + \frac{1}{3!}\boldsymbol{A}^3 + \cdots + \frac{1}{n!}\boldsymbol{A}^n + \cdots$

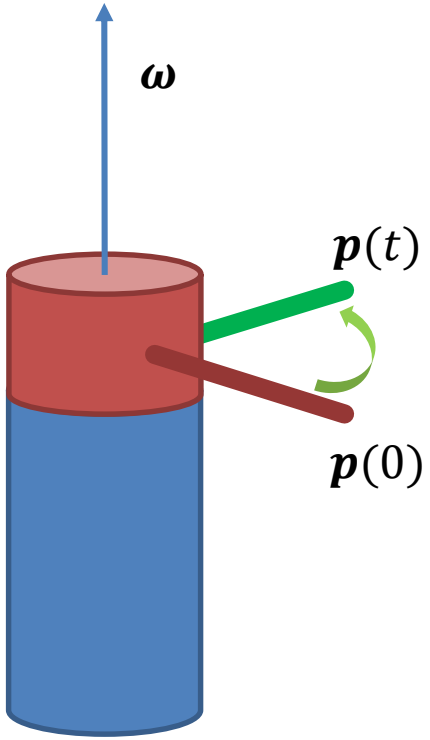
Exponential Coordinates

- Degree-of-freedom of $SO(3)$:
  - $\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$
  - $\boldsymbol{r}_i{}^T \cdot \boldsymbol{r}_j = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases} \Longrightarrow$ 6 constraints
  - $\boldsymbol{R}$ has only 3 independent parameters
- Consider the motion of a point about a unit length rotating link $\boldsymbol{\omega}$ at constant unit velocity:
  - $\begin{cases} \dot{\boldsymbol{p}}(t) = \boldsymbol{\omega} \times \boldsymbol{p}(t) = \hat{\boldsymbol{\omega}} \cdot \boldsymbol{p}(t) \\ \boldsymbol{p}(0) = \boldsymbol{p}_0 \end{cases} \Longrightarrow \boldsymbol{p}(t) = e^{\hat{\boldsymbol{\omega}} t} \boldsymbol{p}_0$



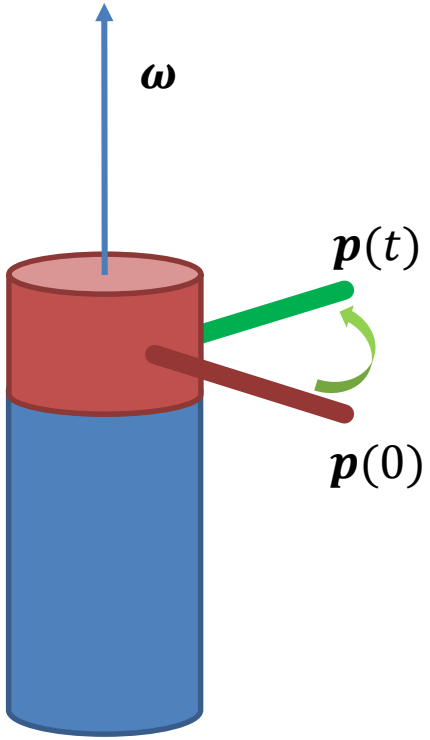
Exponential Coordinates

- Consider the motion of a point about a unit length rotating link $\boldsymbol{\omega}$ at constant unit velocity:
  - $\begin{cases} \dot{\boldsymbol{p}}(t) = \boldsymbol{\omega} \times \boldsymbol{p}(t) = \hat{\boldsymbol{\omega}} \cdot \boldsymbol{p}(t) \\ \boldsymbol{p}(0) = \boldsymbol{p}_0 \end{cases} \Longrightarrow \boldsymbol{p}(t) = e^{\hat{\boldsymbol{\omega}} t} \boldsymbol{p}_0$
  - $\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$
- Rotating about ω at unit velocity for θ units:
  - $\boldsymbol{R}(\boldsymbol{\omega}, \theta) = e^{\hat{\boldsymbol{\omega}} \theta}$



Exponential Coordinates

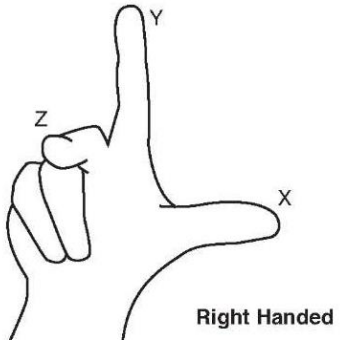
- The vector space of all 3×3 skew-symmetric matrices is denoted as $so(3)$:
  - $so(3) = \{\boldsymbol{S} \in \mathbb{R}^{3\times 3} : \boldsymbol{S}^T = -\boldsymbol{S}\}$
- The exponential map:
  - $\boldsymbol{R}(\boldsymbol{\omega}, \theta) = e^{\hat{\boldsymbol{\omega}}\theta} = \boldsymbol{I} + \hat{\boldsymbol{\omega}} \sin\theta + \hat{\boldsymbol{\omega}}^2(1 - \cos\theta)$
- $e^{\hat{\boldsymbol{\omega}}\theta} \in SO(3)$
  - $\left[e^{\hat{\boldsymbol{\omega}}\theta}\right]^{-1} = e^{-\hat{\boldsymbol{\omega}}\theta} = e^{\hat{\boldsymbol{\omega}}^T\theta} = \left[e^{\hat{\boldsymbol{\omega}}\theta}\right]^T$
  - Since $\det e^{\mathbf{0}} = 1$, and both determinant and exponential map are continuous functions, we know $\det e^{\hat{\boldsymbol{\omega}}\theta} = 1$
- The exponential map is onto (many to one)
  - $\theta = 0 \Rightarrow \boldsymbol{\omega}$ can be chosen arbitrary

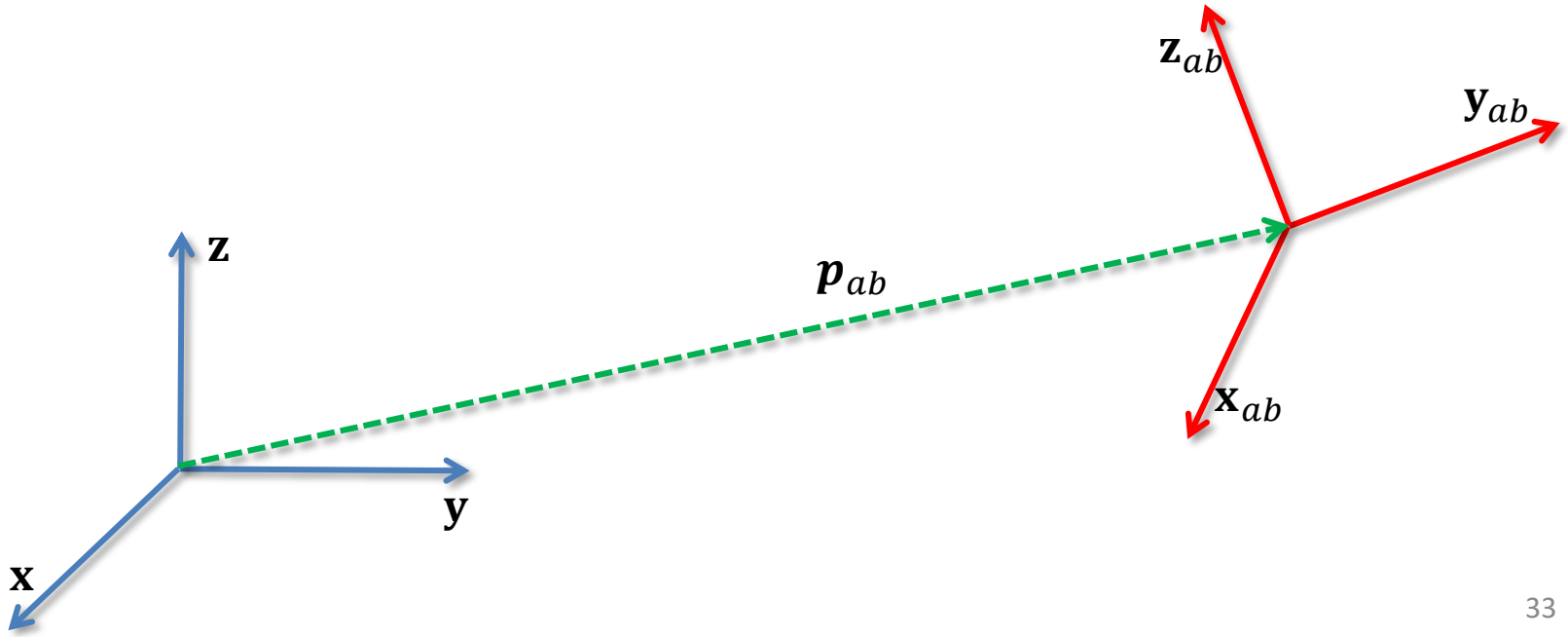


Rigid Body Motions

Rigid Body Motion

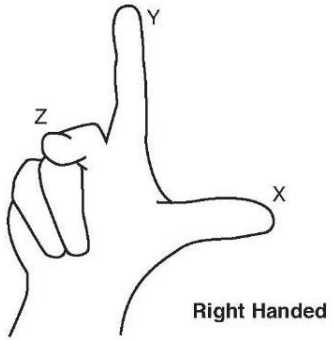
- General rigid body motions that includes both translation and rotation forms the product space of $\mathbb{R}^3$ and $SO(3)$. Denoted as $SE(3)$ – Special Euclidean group.
  - $SE(3) = \{(\boldsymbol{p}, \boldsymbol{R}) : \boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{R} \in SO(3)\} = \mathbb{R}^3 \times SO(3)$

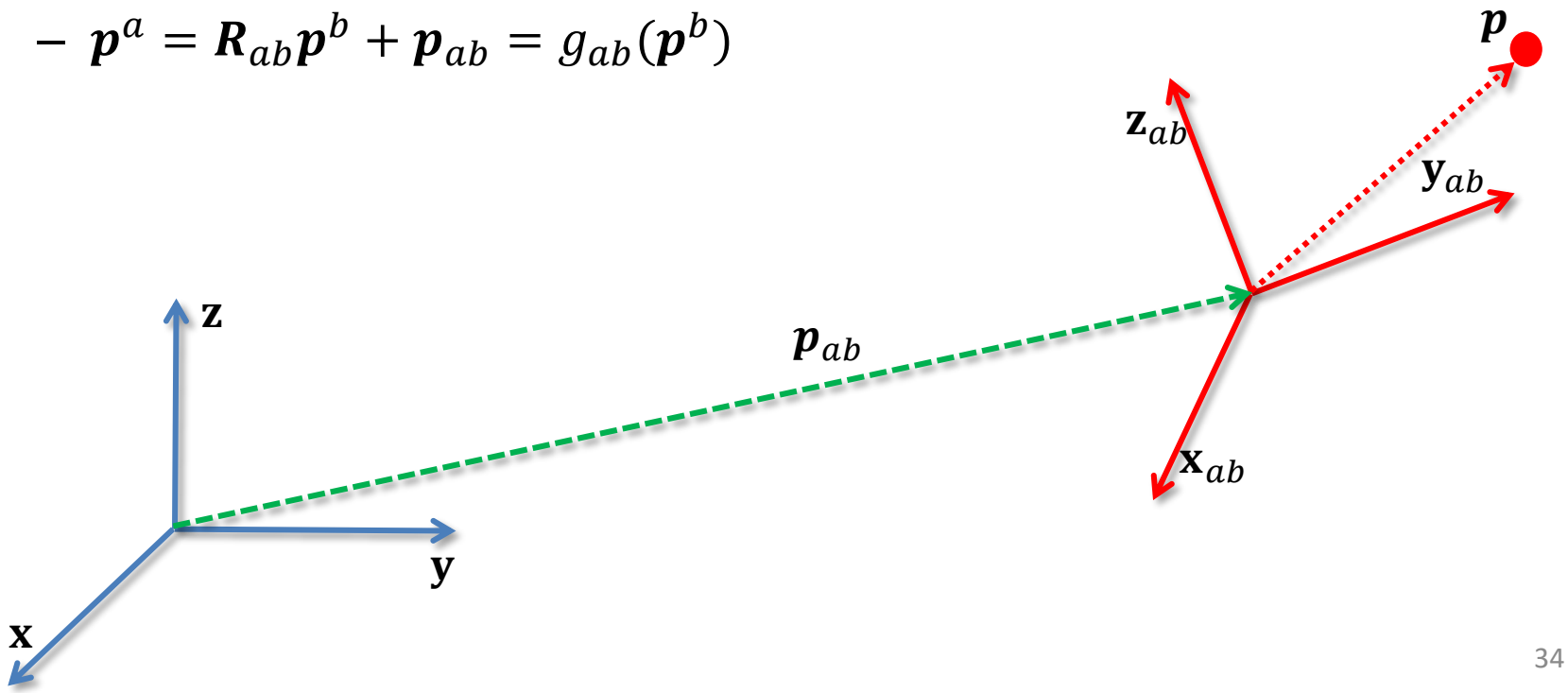




Rigid Body Motion

- Special Euclidean group:
  - $SE(3) = \{(\boldsymbol{p}, \boldsymbol{R}) : \boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{R} \in SO(3)\} = \mathbb{R}^3 \times SO(3)$
- Transformation of a point between different coordinate frames:
  - $\boldsymbol{p}^a = \boldsymbol{R}_{ab}\boldsymbol{p}^b + \boldsymbol{p}_{ab} = g_{ab}(\boldsymbol{p}^b)$



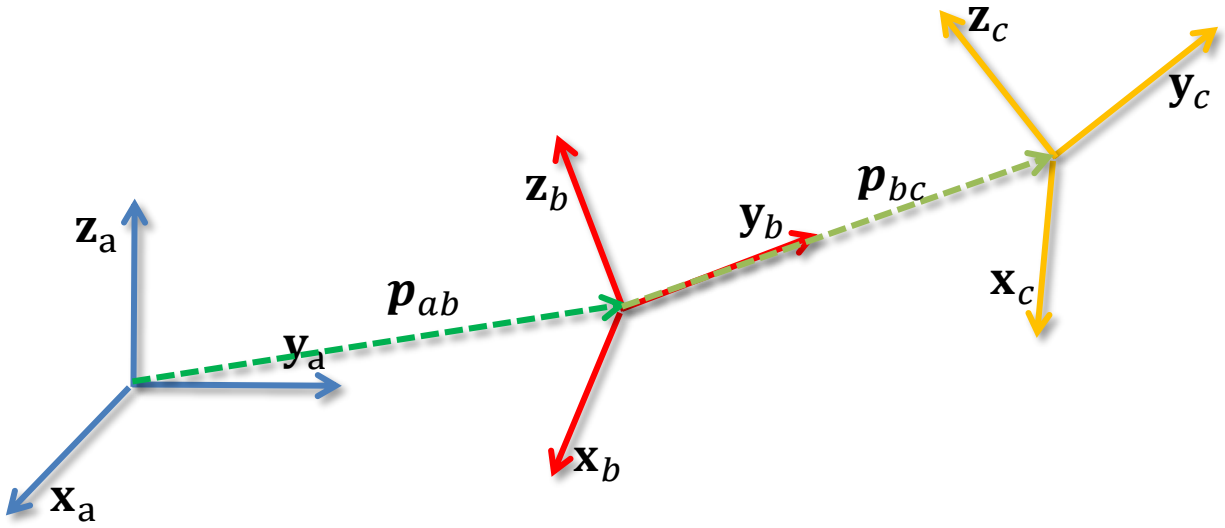


Rigid Body Motion

- Homogeneous coordinates of a point:
  - $\bar{\boldsymbol{p}} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$
- Homogeneous coordinates of a vector:
  - $\bar{\boldsymbol{v}} = \begin{bmatrix} v_x \\ v_y \\ v_z \\ 0 \end{bmatrix}$
- Homogeneous representation of rigid body motion:
  - $\bar{\boldsymbol{p}}^a = \begin{bmatrix} \boldsymbol{p}^a \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{ab} & \boldsymbol{p}_{ab} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}^b \\ 1 \end{bmatrix} = \bar{g}_{ab} \bar{\boldsymbol{p}}^b$

Rigid Body Motion

- Homogeneous representation of rigid body motion:
  - $\bar{g}_{ab} = \begin{bmatrix} \boldsymbol{R}_{ab} & \boldsymbol{p}_{ab} \\ 0 & 1 \end{bmatrix}$
- Composition rule for rigid body motions:
  - $\bar{g}_{ac} = \bar{g}_{ab} \cdot \bar{g}_{bc} = \begin{bmatrix} \boldsymbol{R}_{ab}\boldsymbol{R}_{bc} & \boldsymbol{R}_{ab}\boldsymbol{p}_{bc} + \boldsymbol{p}_{ab} \\ 0 & 1 \end{bmatrix}$
  - Compare with composition of rotational motion: $\boldsymbol{R}_{ac} = \boldsymbol{R}_{ab} \cdot \boldsymbol{R}_{bc}$



Properties of Rigid Body Motion

- $SE(3) = \{(\boldsymbol{p}, \boldsymbol{R}) : \boldsymbol{p} \in \mathbb{R}^3, \boldsymbol{R} \in SO(3)\} = \mathbb{R}^3 \times SO(3)$

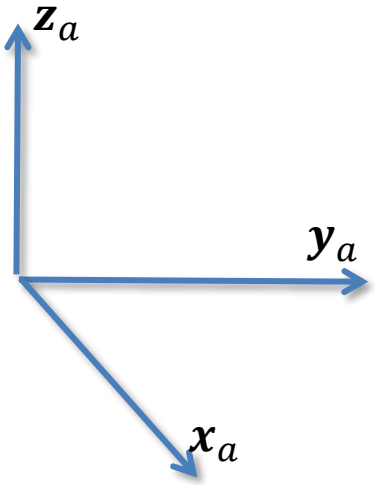
- $SE(3)$ is a group under the operation of matrix multiplication
  - Closure
  - Identity
  - Inverse
  - Associativity
- $g \in SE(3)$ is a rigid body transformation
  - Lengths are preserved
  - Cross products are preserved

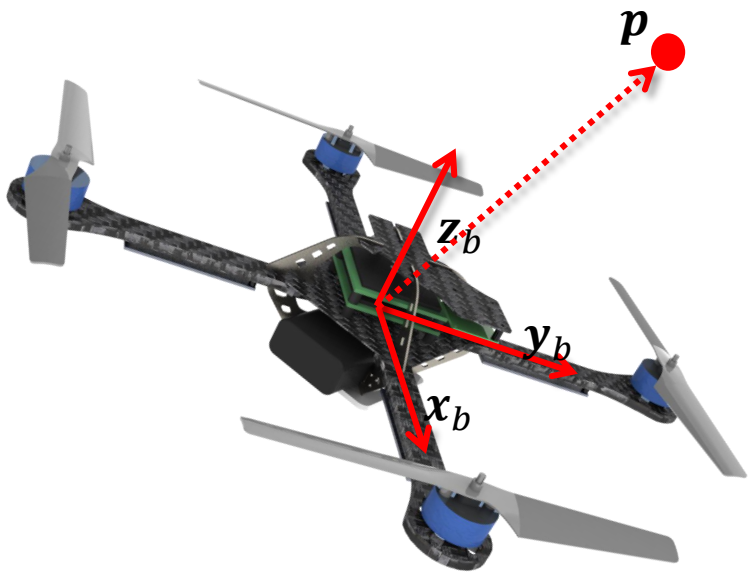
Prove it yourself!

Rigid Body Velocities

Angular Velocity

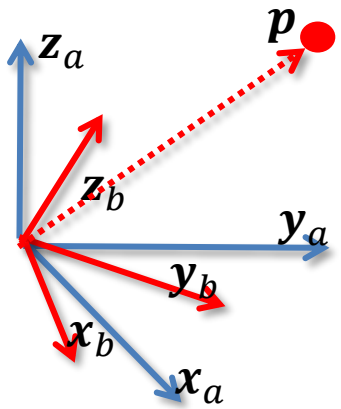
- Coordinate frames:
  - Frame A: spatial frame
  - Frame B: body frame
- A point attached to the body follows a rotational path in spatial frame:
  - $\boldsymbol{p}^a(t) = \boldsymbol{R}_{ab}(t)\boldsymbol{p}^b$





Angular Velocity

- Coordinate frames:
  - Frame A: spatial frame
  - Frame B: body frame
- A point attached to the body follows a rotational path in spatial frame:
  - $\boldsymbol{p}^a(t) = \boldsymbol{R}_{ab}(t)\boldsymbol{p}^b$
- The velocity of the point in spatial frame:
  - $\boldsymbol{v}_p^a(t) = \frac{d}{dt}\boldsymbol{p}^a(t) = \dot{\boldsymbol{R}}_{ab}(t)\boldsymbol{p}^b$
- This can be rewritten as:
  - $\boldsymbol{v}_p^a(t) = \dot{\boldsymbol{R}}_{ab}(t)\boldsymbol{R}_{ab}^{-1}(t)\boldsymbol{R}_{ab}(t)\boldsymbol{p}^b$

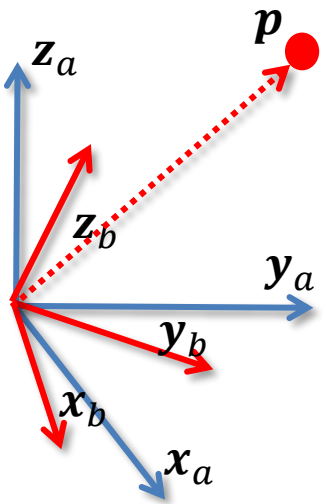


Skew-symmetric matrix.
Why?

Prove $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I} \implies \dot{\boldsymbol{R}}\boldsymbol{R}^T + \boldsymbol{R}\dot{\boldsymbol{R}}^T = 0$

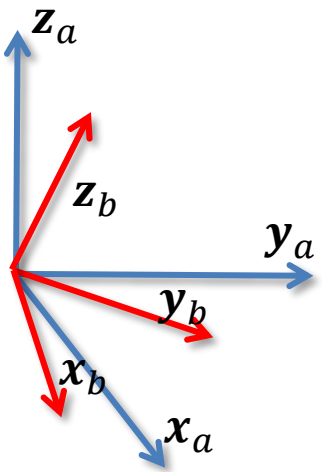
Angular Velocity

- The instantaneous spatial angular velocity $\boldsymbol{\omega}_{ab}^{a}$
  - $\hat{\boldsymbol{\omega}}_{ab}^{a} = \dot{\boldsymbol{R}}_{ab} \cdot \boldsymbol{R}_{ab}^{-1}$
- The instantaneous body angular velocity $\boldsymbol{\omega}_{ab}^{b}$
  - $\hat{\boldsymbol{\omega}}_{ab}^{b} = \boldsymbol{R}_{ab}^{-1} \cdot \dot{\boldsymbol{R}}_{ab}$
- Conversion:
  - $\hat{\boldsymbol{\omega}}_{ab}^{b} = \boldsymbol{R}_{ab}^{-1} \cdot \hat{\boldsymbol{\omega}}_{ab}^{a} \cdot \boldsymbol{R}_{ab}$
  - $\boldsymbol{\omega}_{ab}^{b} = \boldsymbol{R}_{ab}^{-1} \cdot \boldsymbol{\omega}_{ab}^{a}$
- Velocity induced by rotational motion:
  - $\boldsymbol{v}_{p}^{a} = \hat{\boldsymbol{\omega}}_{ab}^{a} \cdot \boldsymbol{R}_{ab} \cdot \boldsymbol{p}^{b} = \boldsymbol{\omega}_{ab}^{a} \times \boldsymbol{p}^{a}$
  - $\boldsymbol{v}_{p}^{b} = \boldsymbol{R}_{ab}^{T} \cdot \boldsymbol{v}_{p}^{a} = \boldsymbol{\omega}_{ab}^{b} \times \boldsymbol{p}^{b}$



Angular Velocity

- Numerical Integration
  - $\dot{\boldsymbol{R}} = \boldsymbol{R}\hat{\boldsymbol{\omega}}^b \Rightarrow \boldsymbol{R}(t+\Delta t) \sim \boldsymbol{R}(t) + \Delta t \cdot \boldsymbol{R}(t)\hat{\boldsymbol{\omega}}^b$
  - $\dot{\boldsymbol{R}} = \hat{\boldsymbol{\omega}}^a\boldsymbol{R} \Rightarrow \boldsymbol{R}(t+\Delta t) \sim \boldsymbol{R}(t) + \Delta t \cdot \hat{\boldsymbol{\omega}}^a\boldsymbol{R}(t)$
- Constant speed rotation
  - $\boldsymbol{R}(t) = \boldsymbol{R}_0 \cdot \exp(\hat{\boldsymbol{\omega}}_0^b \cdot t)$
  - $\boldsymbol{R}(t) = \exp(\hat{\boldsymbol{\omega}}_0^a \cdot t) \cdot \boldsymbol{R}_0$



Angular Velocity

- Simple example

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{R^T} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \dot{\boldsymbol{R}} = \begin{bmatrix} -\sin\theta & -\cos\theta & 0 \\ cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta}$$

Angular Velocity

- Simple example

$$\boldsymbol{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\widehat{\boldsymbol{\omega}}_{ab}^{b} = \boldsymbol{R}^{\boldsymbol{T}}\dot{\boldsymbol{R}} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\sin\theta & -\cos\theta & 0 \\ cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta} = \widehat{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}} \dot{\theta}$$

$$\widehat{\boldsymbol{\omega}}_{ab}^{a} = \dot{\boldsymbol{R}}\boldsymbol{R}^{\boldsymbol{T}} = \dot{\theta} \begin{bmatrix} -\sin\theta & -\cos\theta & 0 \\ cos\theta & -\sin\theta & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \dot{\theta} = \widehat{\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}} \dot{\theta}$$

Angular Velocity

- Two rotations

$$\boldsymbol{R} = \boldsymbol{R}_z(\theta)\,\boldsymbol{R}_x(\varphi)$$

$$\widehat{\boldsymbol{\omega}}^b = \boldsymbol{R}^{\boldsymbol{T}}\dot{\boldsymbol{R}} = (\boldsymbol{R}_z\,\boldsymbol{R}_x)^T(\dot{\boldsymbol{R}}_z\boldsymbol{R}_x + \boldsymbol{R}_z\dot{\boldsymbol{R}}_x)$$

$$= \boldsymbol{R}_x^T\boldsymbol{R}_z^T\dot{\boldsymbol{R}}_z\boldsymbol{R}_x + \boldsymbol{R}_x^T\dot{\boldsymbol{R}}_x$$

$$\widehat{\boldsymbol{\omega}}^a = \dot{\boldsymbol{R}}\boldsymbol{R}^{\boldsymbol{T}} = (\dot{\boldsymbol{R}}_z\boldsymbol{R}_x + \boldsymbol{R}_z\dot{\boldsymbol{R}}_x)(\boldsymbol{R}_z\,\boldsymbol{R}_x)^T$$

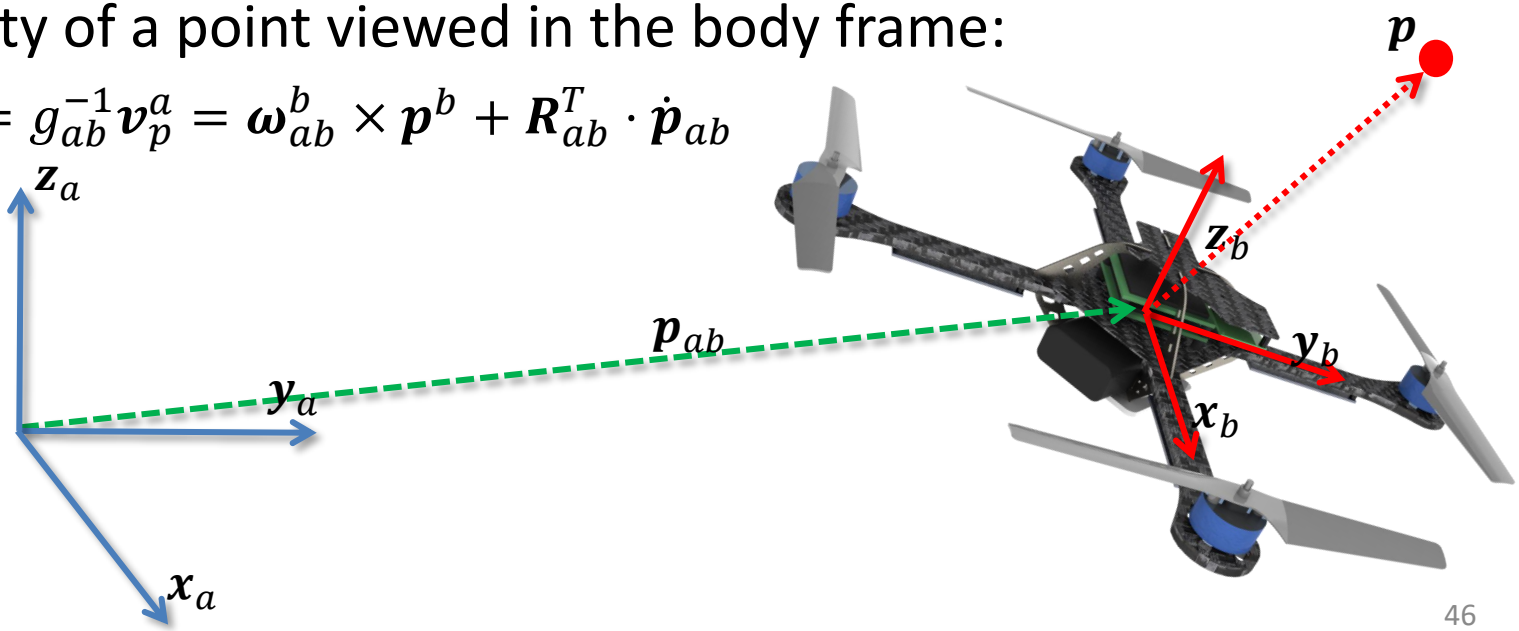
$$= \dot{\boldsymbol{R}}_z\boldsymbol{R}_z^T + \boldsymbol{R}_z\dot{\boldsymbol{R}}_x\boldsymbol{R}_x^T\boldsymbol{R}_z^T$$

Rigid Body Velocity

- General rigid body transformation:
  - $g_{ab} = \begin{bmatrix} \boldsymbol{R}_{ab} & \boldsymbol{p}_{ab} \\ 0 & 1 \end{bmatrix}$

For detailed interpretation, refer to Chapter 2.4 of "A Mathematical Introduction to Robotic Manipulation"

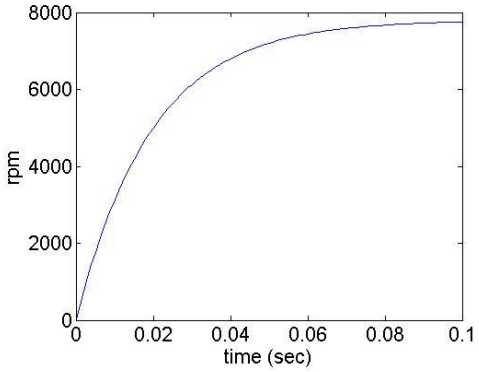
- Velocity of a point viewed in the spatial frame:
  - $\boldsymbol{v}_p^a = \dot{g}_{ab} g_{ab}^{-1} \boldsymbol{p}^a = \boldsymbol{\omega}_{ab}^a \times \boldsymbol{p}^a - \boldsymbol{\omega}_{ab}^a \times \boldsymbol{p}_{ab} + \dot{\boldsymbol{p}}_{ab}$
- Velocity of a point viewed in the body frame:
  - $\boldsymbol{v}_p^b = g_{ab}^{-1} \boldsymbol{v}_p^a = \boldsymbol{\omega}_{ab}^b \times \boldsymbol{p}^b + \boldsymbol{R}_{ab}^T \cdot \dot{\boldsymbol{p}}_{ab}$

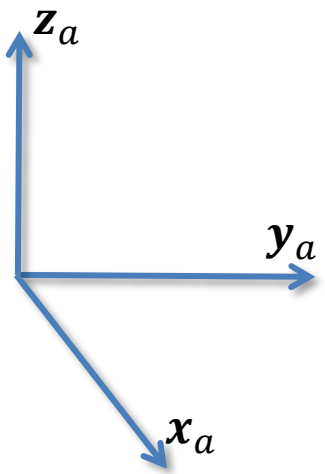


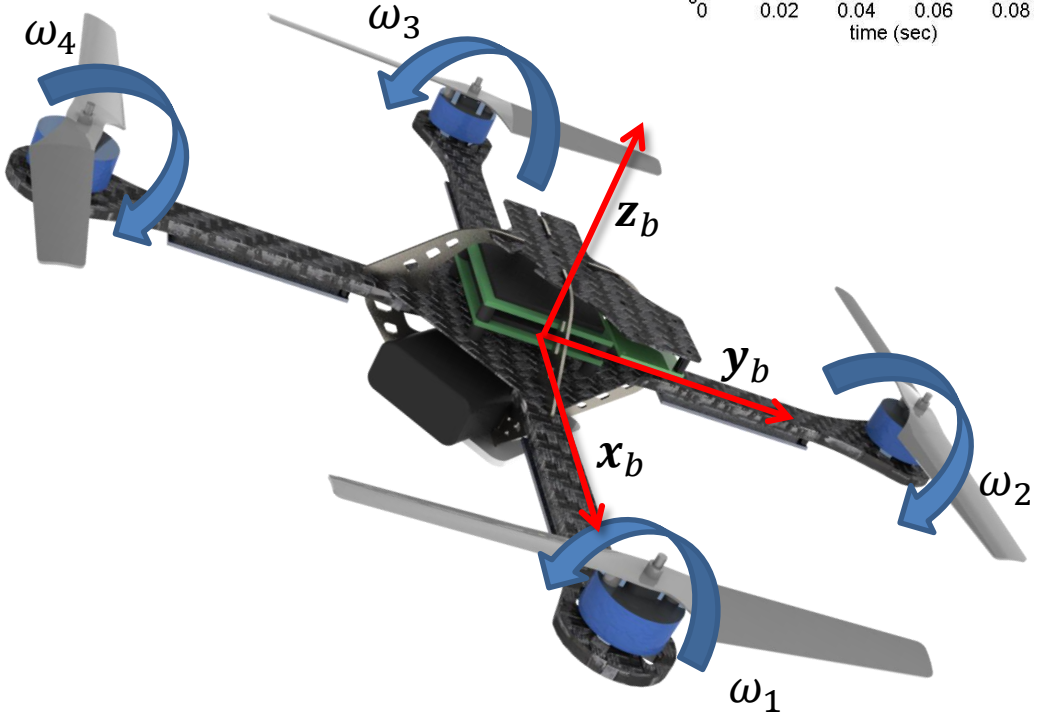
Quadrotor Dynamics

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: $M_i = k_M\omega_i^2$

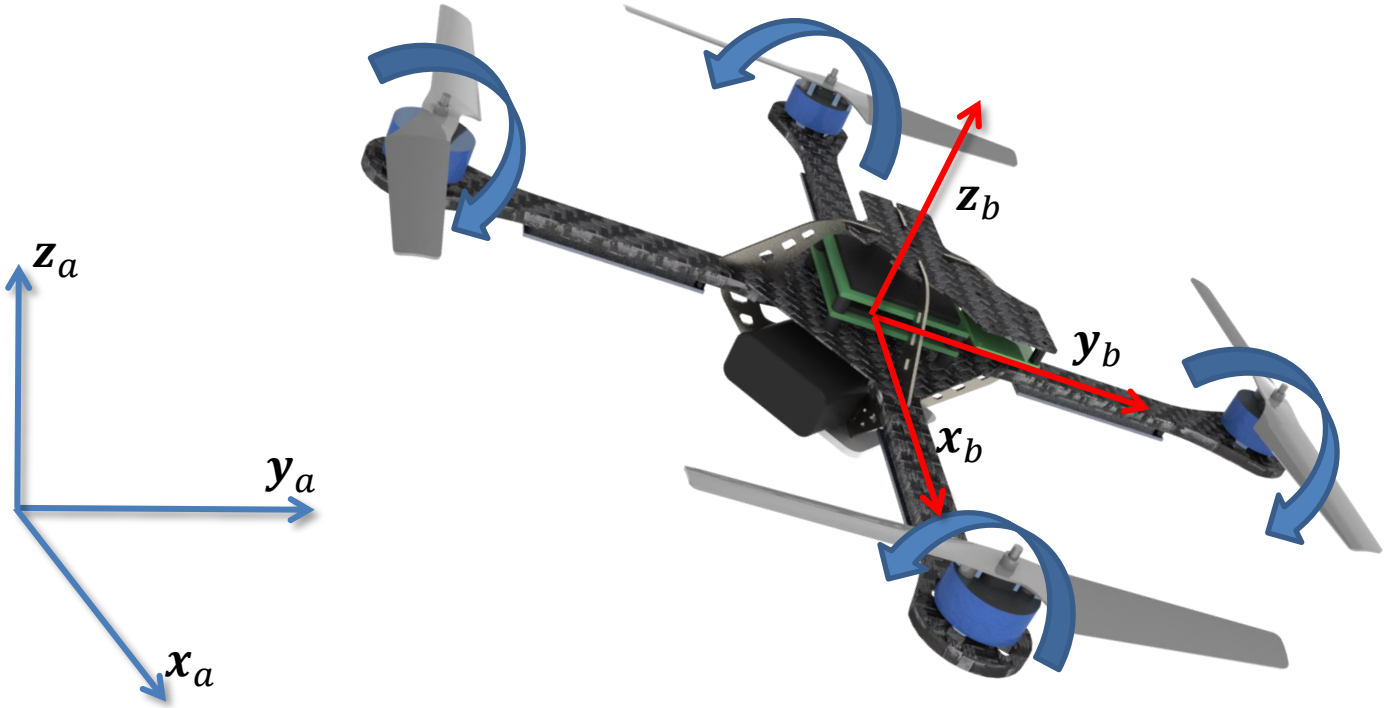






Quadrotor Dynamics

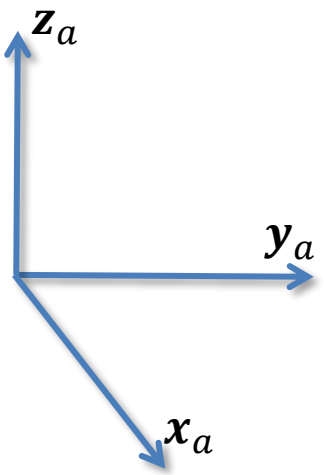
- Z-X-Y Euler Angles: $\boldsymbol{R}_{ab} = \boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_x(\phi) \cdot \boldsymbol{R}_y(\theta)$
- Sequence of three rotations about body-fixed axes
- What are the singularities?

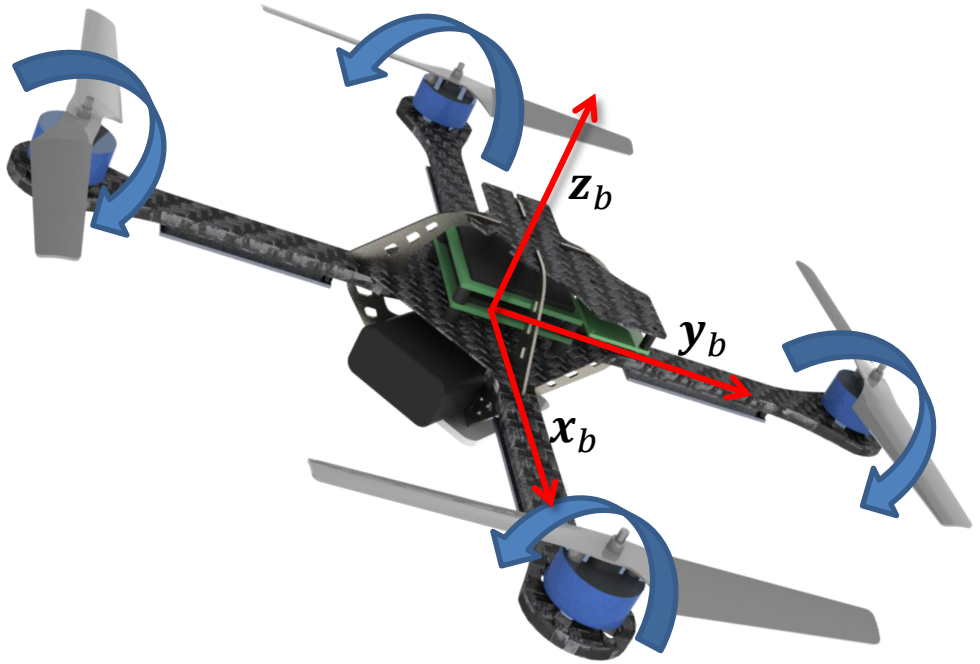


Quadrotor Dynamics

- $\boldsymbol{R}_{ab} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$
- $\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$

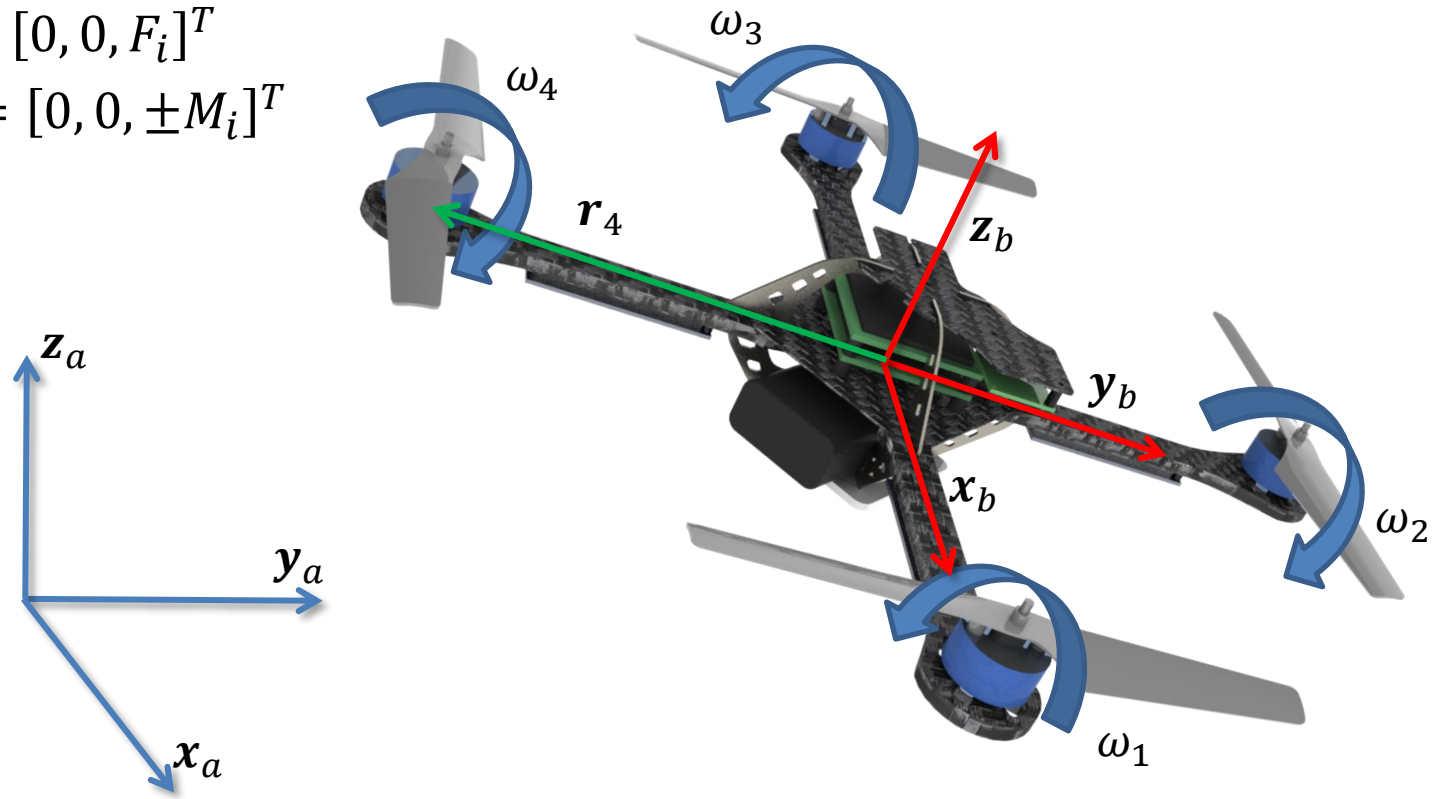
Instantaneous body angular velocity.





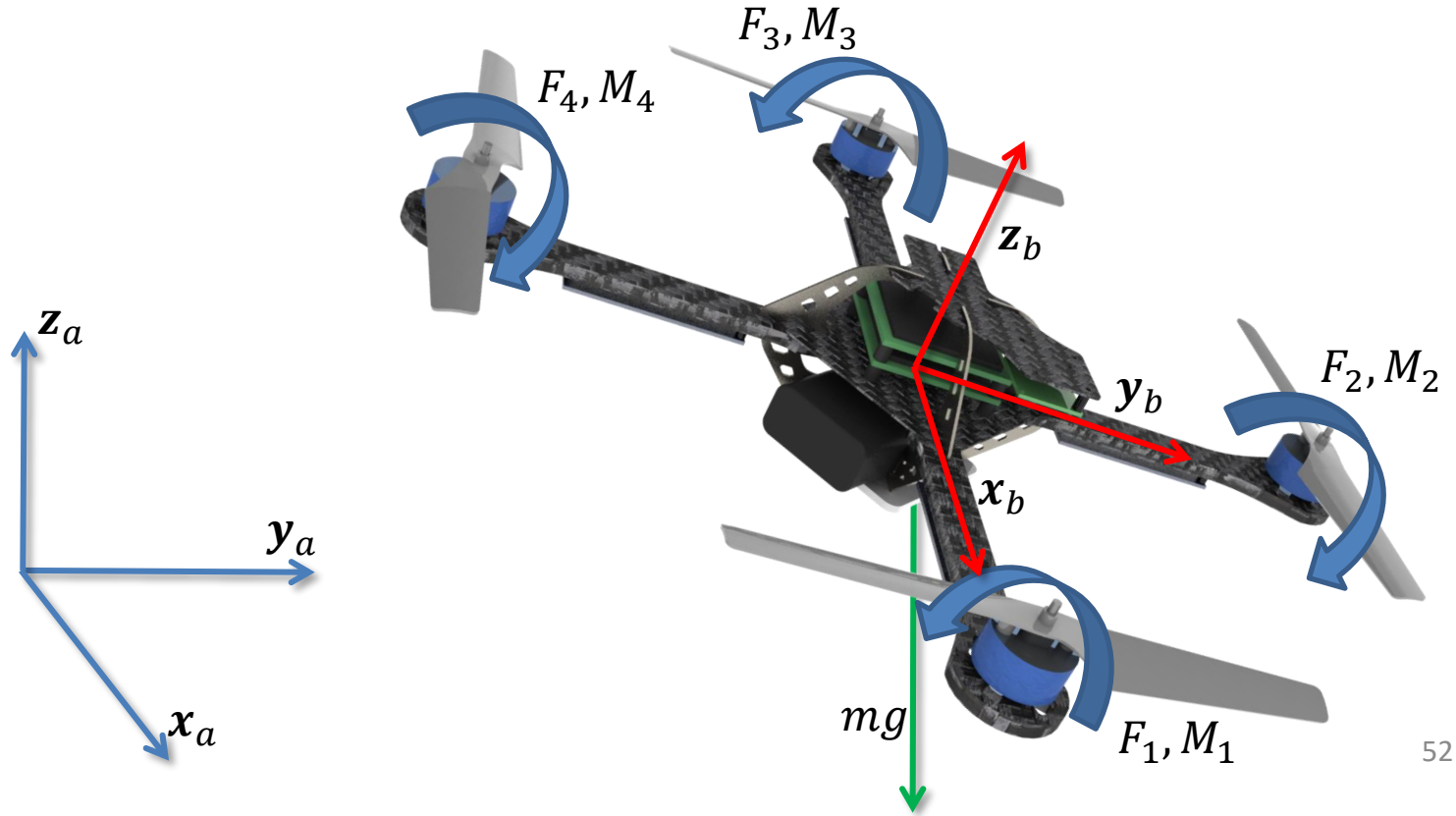
Quadrotor Dynamics

- Consider body frame
  - $\boldsymbol{F} = \boldsymbol{F}_1 + \boldsymbol{F}_2 + \boldsymbol{F}_3 + \boldsymbol{F}_4 - \boldsymbol{R}_{ab}^T[0, 0, mg]^T$
  - $\boldsymbol{M} = \boldsymbol{r}_1 \times \boldsymbol{F}_1 + \boldsymbol{r}_2 \times \boldsymbol{F}_2 + \boldsymbol{r}_3 \times \boldsymbol{F}_3 + \boldsymbol{r}_4 \times \boldsymbol{F}_4 + \boldsymbol{M}_1 + \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_4$
  - $\boldsymbol{F}_i = [0, 0, F_i]^T$
  - $\boldsymbol{M}_i = [0, 0, \pm M_i]^T$



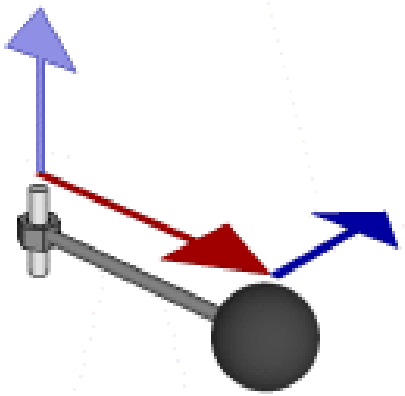
Newton-Euler Equations

- Newton Equation: $m\ddot{\boldsymbol{p}}^a = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}_{ab} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$



Newton-Euler Equations

$$\tau = \mathbf{r} \times \mathbf{F}$$
$$\mathbf{L} = \mathbf{r} \times \mathbf{p}$$

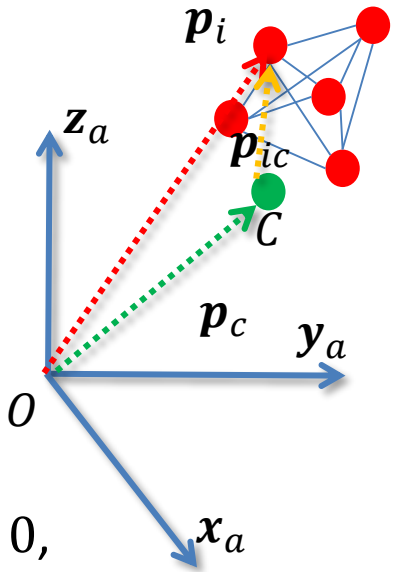


Relationship between force (F), torque/moment of force(τ), momentum (p), and angular momentum (L) vectors in a rotating system. r is the position vector.

Newton-Euler Equations

- The rigid body as a collection of particles
  - Center of mass (CoM): $\boldsymbol{p}_c$
  - Position of the i-th particle to CoM: $\boldsymbol{p}_{ic} = \boldsymbol{p}_i - \boldsymbol{p}_c$
  - Velocity of the i-th particle to CoM: $\boldsymbol{v}_{ic} = \dot{\boldsymbol{p}}_i - \dot{\boldsymbol{p}}_c = \boldsymbol{v}_i - \boldsymbol{v}_c$
  - Angular momentum of the i-th particle:

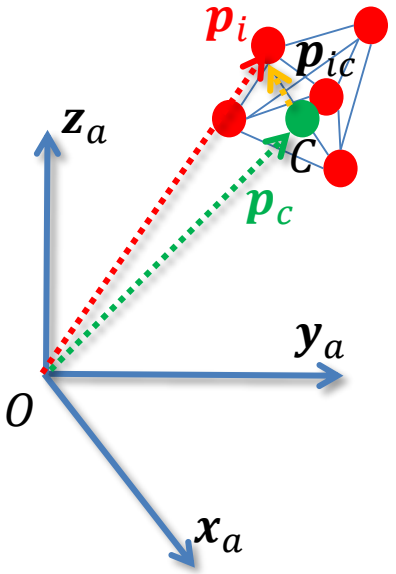
$$\boldsymbol{H}_i = \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$$



- Angular momentum of the rigid body:
  - $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$
  - Since: $\sum m_i \boldsymbol{p}_{ic} = \sum m_i (\boldsymbol{p}_i - \boldsymbol{p}_c) = \sum m_i \boldsymbol{p}_i - \boldsymbol{p}_c \sum m_i = 0,$
  - We have: $\sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_c = (\sum m_i \boldsymbol{p}_{ic}) \times \boldsymbol{v}_c = 0$
  - Therefore: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i - \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_c = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_{ic}$
  - Since: $\boldsymbol{v}_{ic} = \boldsymbol{\omega} \times \boldsymbol{p}_{ic},$
  - We have: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times (\boldsymbol{\omega} \times m_i \boldsymbol{p}_{ic}) = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$

Newton-Euler Equations

- Rotational dynamics
  - Angular momentum: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$
  - Take the derivative: $\dot{\boldsymbol{H}} = \sum \dot{\boldsymbol{p}}_{ic} \times m_i \boldsymbol{v}_i + \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i$
  - Since $\sum \dot{\boldsymbol{p}}_{ic} \times m_i \boldsymbol{v}_i = \sum \boldsymbol{v}_i \times m_i \boldsymbol{v}_i - \boldsymbol{v}_c \times m_i \boldsymbol{v}_i = \sum -\boldsymbol{v}_c \times m_i \boldsymbol{v}_i = -\boldsymbol{v}_c \times \frac{d}{dt} \sum m_i \boldsymbol{p}_i = -\boldsymbol{v}_c \times \frac{d}{dt} \boldsymbol{p}_c \sum m_i = 0$
  - We have $\dot{\boldsymbol{H}} = \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i$
  - Referring to Newton's second law: $\boldsymbol{F}_i + \sum_{i \neq j} \boldsymbol{F}_{ij} = m_i \dot{\boldsymbol{v}}_i$
  - $\dot{\boldsymbol{H}} = \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i = \sum \boldsymbol{p}_{ic} \times (\boldsymbol{F}_i + \sum_{i \neq j} \boldsymbol{F}_{ij}) = \sum \boldsymbol{p}_{ic} \times \boldsymbol{F}_i$
  - We also know that the external moment: $\boldsymbol{M} = \sum \boldsymbol{p}_{ic} \times \boldsymbol{F}_i$
  - We have the rotational dynamics: $\boldsymbol{M} = \dot{\boldsymbol{H}}$



Newton-Euler Equations

- Finishing the work on rotational dynamics
  - Given: $\boldsymbol{H} = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$
  - And using the fact: $(\boldsymbol{R}\boldsymbol{a}) \times (\boldsymbol{R}\boldsymbol{b}) = \boldsymbol{R}(\boldsymbol{a} \times \boldsymbol{b})$
    - $\boldsymbol{R}$: rotation matrix
    - $\boldsymbol{a}, \boldsymbol{b}$: vectors
  - We can transform the representation of the angular momentum to the body frame with constant inertia matrix:

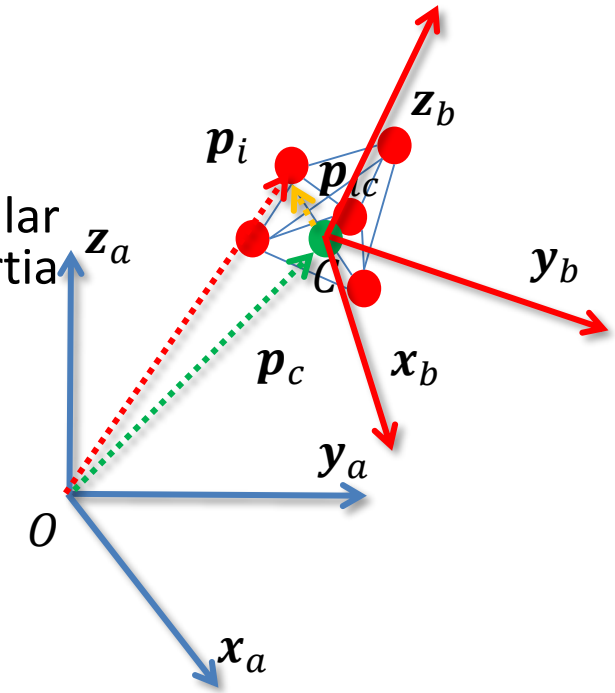
$$\boldsymbol{H} = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$$

$$= -\sum \boldsymbol{R}_{ab} \boldsymbol{p}_{ic}^b \times \left(m_i \boldsymbol{R}_{ab} \boldsymbol{p}_{ic}^b \times \boldsymbol{R}_{ab} \boldsymbol{\omega}^b\right)$$

$$= -\boldsymbol{R}_{ab} \sum \boldsymbol{p}_{ic}^b \times \left(m_i \boldsymbol{p}_{ic}^b \times \boldsymbol{\omega}^b\right)$$

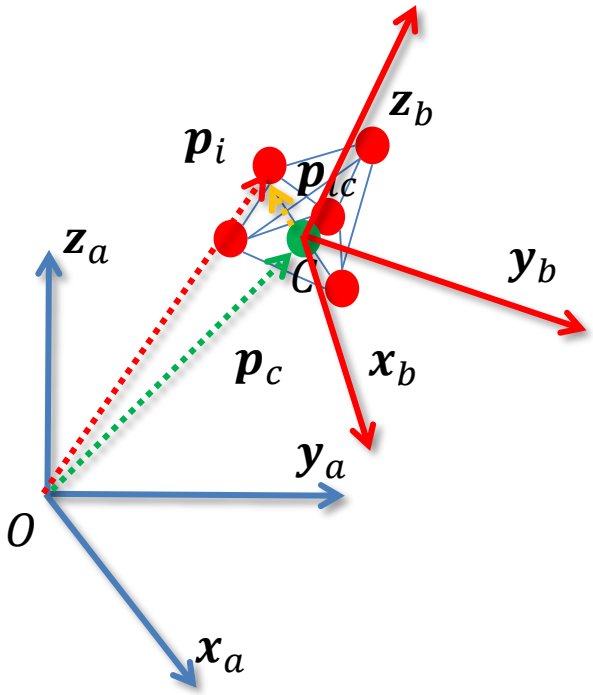
$$= -\boldsymbol{R}_{ab} \sum m_i \cdot \boldsymbol{p}_{ic}^b \times \left(\widehat{\boldsymbol{p}}_{ic}^b \cdot \boldsymbol{\omega}^b\right)$$

$$= \boldsymbol{R}_{ab}\left(-\sum m_i \cdot \widehat{\boldsymbol{p}}_{ic}^b \cdot \widehat{\boldsymbol{p}}_{ic}^b\right) \cdot \boldsymbol{\omega}^b = \boldsymbol{R}_{ab}(\boldsymbol{I}^b \boldsymbol{\omega}^b)$$



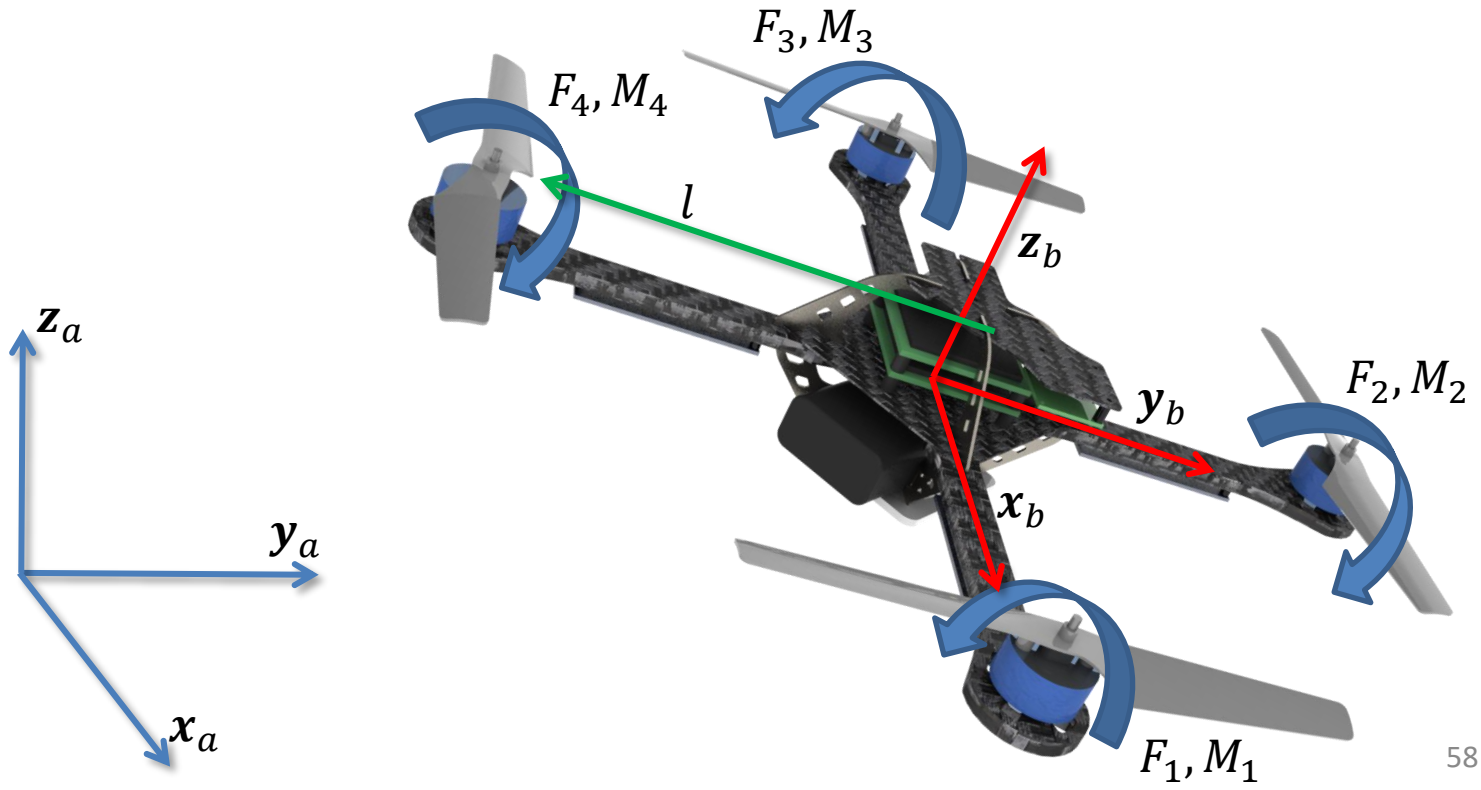
Newton-Euler Equations

- Finishing the work on rotational dynamics
  - Given $\boldsymbol{H} = \boldsymbol{R}_{ab}(\boldsymbol{I}^b\boldsymbol{\omega}^b)$
  - Take the derivative: $\dot{\boldsymbol{H}} = \dot{\boldsymbol{R}}_{ab}\boldsymbol{I}^b\boldsymbol{\omega}^b + \boldsymbol{R}_{ab}\boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b = \boldsymbol{R}_{ab}\hat{\boldsymbol{\omega}}^b\boldsymbol{I}^b\boldsymbol{\omega}^b + \boldsymbol{R}_{ab}\boldsymbol{I}^b\dot{\omega}^b = \boldsymbol{R}_{ab}(\boldsymbol{\omega}^b \times (\boldsymbol{I}^b\boldsymbol{\omega}^b) + \boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b)$
  - Also transform the moment into body frame: $\boldsymbol{M} = \boldsymbol{R}_{ab}\boldsymbol{M}^b$
  - Finally: $\boldsymbol{M}^b = \boldsymbol{\omega}^b \times (\boldsymbol{I}^b\boldsymbol{\omega}^b) + \boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b$



Newton-Euler Equations

- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$

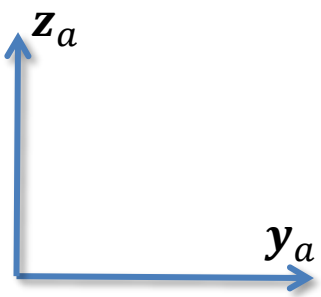


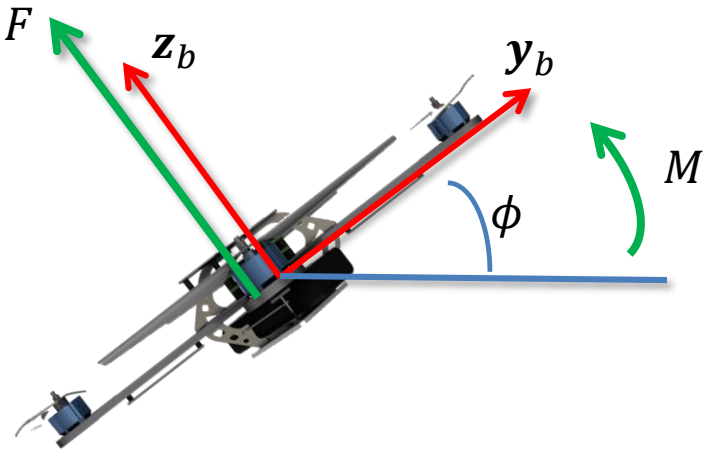
Quadrotor Dynamics

- 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: $m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$
- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$

A Planar Quadrotor

- $$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} F \\ M \end{bmatrix}$$





ω_1

Assignment

- Chapter 2.1-2.4 of “A Mathematical Introduction to Robotic Manipulation”

Next Lecture…

- Control basics
- Quadrotor control
- Trajectory generation

Introducing Quaternion

- Rotation matrix $\boldsymbol{R} \in SO(3)$
  - No singularity
  - Redundant parameters
  - Kinematics: $\dot{\boldsymbol{R}} = \boldsymbol{R}\hat{\boldsymbol{\omega}}$, $\boldsymbol{\omega}$ is the body angular rate
- ZYX Euler angle
  - Singular at roll angle of 90 degrees
  - Minimum number of parameters
  - Kinematics: $\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} \frac{\cos(\psi)}{\cos(\theta)} & \frac{\sin(\psi)}{\cos(\theta)} & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ \cos(\psi)\tan(\theta) & \sin(\psi)\tan(\theta) & 1 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$
- Is there a *singularity free* parameterization that with *reduced* parameters? YES ➔ quaternion

Introducing Quaternion

- Recall a rotation matrix $\boldsymbol{R}$ can be decomposed as a rotation axis $\boldsymbol{u}$ of angle θ by $\boldsymbol{R} = e^{\hat{\boldsymbol{u}}\theta} = \boldsymbol{I} + \hat{\boldsymbol{u}}\sin\theta + \hat{\boldsymbol{u}}^2(1-\cos\theta)$
- Any vector $\boldsymbol{p}$, after rotation $\boldsymbol{R}$, is $\boldsymbol{p}' = \boldsymbol{R}\boldsymbol{p}$

$$\Rightarrow \begin{bmatrix} 0 \\ \boldsymbol{p}' \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & \boldsymbol{R} \end{bmatrix} \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix}$$

A representation of rotation

$$\Rightarrow \begin{bmatrix} 0 \\ \boldsymbol{p}' \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2} u \end{bmatrix} \circ \begin{bmatrix} 0 \\ \boldsymbol{p} \end{bmatrix} \circ \begin{bmatrix} \cos\frac{\theta}{2} \\ -\sin\frac{\theta}{2}\boldsymbol{u} \end{bmatrix}; \forall \boldsymbol{p}, \boldsymbol{u}, \theta$$

A new binary operation

where

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} \circ \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \stackrel{\text{def}}{=} \begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & b_3 & -b_2 \\ b_2 & -b_3 & b_0 & b_1 \\ b_3 & b_2 & -b_1 & b_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Definitions of Quaternion

- Quaternion

$$\boldsymbol{Q} \overset{\text{def}}{=} q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} \overset{\text{def}}{=} q_0 + \boldsymbol{q} \overset{\text{def}}{=} \begin{bmatrix} q_0 \\ \boldsymbol{q} \end{bmatrix}$$

where $\boldsymbol{i} \circ \boldsymbol{i} = \boldsymbol{j} \circ \boldsymbol{j} = \boldsymbol{k} \circ \boldsymbol{k} \overset{\text{def}}{=} -1; \boldsymbol{i} \circ \boldsymbol{j} \overset{\text{def}}{=} \boldsymbol{k}; \boldsymbol{j} \circ \boldsymbol{k} \overset{\text{def}}{=} \boldsymbol{i}; \boldsymbol{k} \circ \boldsymbol{i} \overset{\text{def}}{=} \boldsymbol{j}$

- Conjugate

$$\overline{\boldsymbol{Q}} \overset{\text{def}}{=} q_0 - q_1\boldsymbol{i} - q_2\boldsymbol{j} - q_3\boldsymbol{k} \overset{\text{def}}{=} q_0 - \boldsymbol{q} \overset{\text{def}}{=} \begin{bmatrix} q_0 \\ -\boldsymbol{q} \end{bmatrix}$$

- Norm $\|\boldsymbol{Q}\| \overset{\text{def}}{=} \boldsymbol{Q} \circ \overline{\boldsymbol{Q}}$

- Inverse $\boldsymbol{Q}^{-1} \overset{\text{def}}{=} \dfrac{\overline{\boldsymbol{Q}}}{\|\boldsymbol{Q}\|}$

Properties of Quaternion

- Property 1

$$\boldsymbol{A} = a_0 + \boldsymbol{a}, \quad \boldsymbol{B} = b_0 + \boldsymbol{b} \qquad \text{particularly}$$

$$\boldsymbol{A} \circ \boldsymbol{B} = (a_0 b_0 - \boldsymbol{a} \cdot \boldsymbol{b}) + (a_0 \boldsymbol{b} + \boldsymbol{a} b_0 + \boldsymbol{a} \times \boldsymbol{b})$$

$$(0 + \boldsymbol{a}) \circ (0 + \boldsymbol{b}) = -\boldsymbol{a} \cdot \boldsymbol{b} + \boldsymbol{a} \times \boldsymbol{b}$$

- Property 2 $(\boldsymbol{A} \circ \boldsymbol{B}) \circ \boldsymbol{C} = \boldsymbol{A} \circ (\boldsymbol{B} \circ \boldsymbol{C})$
- Property 3 $\overline{\boldsymbol{A} \circ \boldsymbol{B}} = \overline{\boldsymbol{B}} \circ \overline{\boldsymbol{A}}$
- Property 4 $\boldsymbol{A} = a_0 + a_1 \boldsymbol{i} + a_2 \boldsymbol{j} + a_3 \boldsymbol{k}$
  $\boldsymbol{B} = b_0 + b_1 \boldsymbol{i} + b_2 \boldsymbol{j} + b_3 \boldsymbol{k}$

Exactly what we need for representing rotations!!

$$\boldsymbol{A} \circ \boldsymbol{B} = \underbrace{\begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix}}_{\mathcal{L}(\boldsymbol{A})} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \underbrace{\begin{bmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & b_3 & -b_2 \\ b_2 & -b_3 & b_0 & b_1 \\ b_3 & b_2 & -b_1 & b_0 \end{bmatrix}}_{\mathcal{R}(\boldsymbol{B})} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

Properties of Quaternion

- Product of Quaternions $\boldsymbol{A} \circ \boldsymbol{B} = \mathcal{L}(\boldsymbol{A}) \begin{bmatrix} b_0 \\ \boldsymbol{b} \end{bmatrix} = \mathcal{R}(\boldsymbol{B}) \begin{bmatrix} a_0 \\ \boldsymbol{a} \end{bmatrix}$

$$\mathcal{L}(\boldsymbol{A}) = \begin{bmatrix} a_0 & -a_1 & -a_2 & -a_3 \\ a_1 & a_0 & -a_3 & a_2 \\ a_2 & a_3 & a_0 & -a_1 \\ a_3 & -a_2 & a_1 & a_0 \end{bmatrix} = \begin{bmatrix} a_0 & -\boldsymbol{a}^T \\ \boldsymbol{a} & a_0 \boldsymbol{I}_3 + \widehat{\boldsymbol{a}} \end{bmatrix}$$

$$\mathcal{R}(\boldsymbol{B}) = \begin{bmatrix} b_0 & -b_1 & -b_2 & -b_3 \\ b_1 & b_0 & b_3 & -b_2 \\ b_2 & -b_3 & b_0 & b_1 \\ b_3 & b_2 & -b_1 & b_0 \end{bmatrix} = \begin{bmatrix} b_0 & -\boldsymbol{b}^T \\ \boldsymbol{b} & b_0 \boldsymbol{I}_3 - \widehat{\boldsymbol{a}} \end{bmatrix}$$

Unit Quaternion and Rotation

Recall

$$\underbrace{\begin{bmatrix}0\\ \boldsymbol{p}'\end{bmatrix}}_{\boldsymbol{P}'} = \underbrace{\begin{bmatrix}\cos\frac{\theta}{2}\\ \sin\frac{\theta}{2}\boldsymbol{u}\end{bmatrix}}_{\boldsymbol{Q}} \circ \underbrace{\begin{bmatrix}0\\ \boldsymbol{p}\end{bmatrix}}_{\boldsymbol{P}} \circ \underbrace{\begin{bmatrix}\cos\frac{\theta}{2}\\ -\sin\frac{\theta}{2}\boldsymbol{u}\end{bmatrix}}_{\overline{\boldsymbol{Q}}}$$

$$\boldsymbol{P}' \overset{\text{def}}{=} 0 + \boldsymbol{p}'; \boldsymbol{P} \overset{\text{def}}{=} 0 + \boldsymbol{p}; \qquad \boxed{\boldsymbol{Q} \overset{\text{def}}{=} \cos\frac{\theta}{2} + \sin\frac{\theta}{2}\boldsymbol{u}}$$

If using quaternion Q to represent a rotation:

Basic principle: $\boldsymbol{P}' = \boldsymbol{Q} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}$

- $\boldsymbol{P}', \boldsymbol{P}$ are respectively the coordinates of the post-rotation vector and prior-rotation vector, both in the same frame.
- $\boldsymbol{Q}$ is the rotation quaternion associated with the rotation

Unit Quaternion and Rotation

Basic principle: $\boldsymbol{P'} = \boldsymbol{Q} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}$

Property 1 Constant vector: $\boldsymbol{P}^b = \overline{\boldsymbol{Q}} \circ \boldsymbol{P} \circ \boldsymbol{R}$

- $\boldsymbol{P}^b, \boldsymbol{P}$ are respectively the coordinates in the body frame and the world frame, of the same vector
- Can be interpreted as that the vector is rotating in the opposite direction, then call the first result

Unit Quaternion and Rotation

Basic principle： $\boldsymbol{P'} = \boldsymbol{Q} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}$

Property 2 Two sequent rotations

- Case 1

  $\boldsymbol{Q}_1, \boldsymbol{Q}_2$ are the two rotation quaternions where the rotation axes are both represented in the initial frame

$$\left.\begin{aligned} \boldsymbol{P'} &= \boldsymbol{Q}_1 \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}_1 \\ \boldsymbol{P''} &= \boldsymbol{Q}_2 \circ \boldsymbol{P'} \circ \overline{\boldsymbol{Q}}_2 = \underbrace{\boldsymbol{Q}_2 \circ \boldsymbol{Q}_1}_{\boldsymbol{Q}} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}_1 \circ \overline{\boldsymbol{Q}}_2 \end{aligned}\right\} \Rightarrow \boldsymbol{Q} = \boldsymbol{Q}_2 \circ \boldsymbol{Q}_1$$

- Case 2

  $\boldsymbol{Q}_1, \boldsymbol{Q}_2$ are the two rotation quaternions where the rotation axis of $\boldsymbol{Q}_2$ is represented in the frame obtained by performing $\boldsymbol{Q}_1$.

$$\boldsymbol{P'} = \boldsymbol{Q}_1 \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}_1$$

$$\boldsymbol{P''} = (\boldsymbol{Q}_1 \circ \boldsymbol{Q}_2 \circ \overline{\boldsymbol{Q}}_1) \circ \boldsymbol{P'} \circ \overline{\boldsymbol{Q}_1 \circ \boldsymbol{Q}_2 \circ \overline{\boldsymbol{Q}}_1} = (\boldsymbol{Q}_1 \circ \boldsymbol{Q}_2 \circ \overline{\boldsymbol{Q}}_1) \circ \boldsymbol{Q}_1 \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}_1 \circ \overline{\boldsymbol{Q}_1 \circ \boldsymbol{Q}_2 \circ \overline{\boldsymbol{Q}}_1} = \underbrace{\boldsymbol{Q}_1 \circ \boldsymbol{Q}_2}_{\boldsymbol{Q}} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}_1 \circ \boldsymbol{Q}_2}$$

$$\Rightarrow \boldsymbol{Q} = \boldsymbol{Q}_1 \circ \boldsymbol{Q}_2$$

Quaternion Kinematics

Recall

$$\boldsymbol{P'} = \boldsymbol{Q} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}$$

Taking derivative yields

$$\dot{\boldsymbol{P}}' = \dot{\boldsymbol{Q}} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}} + \boldsymbol{Q} \circ \boldsymbol{P} \circ \dot{\overline{\boldsymbol{Q}}}$$

$$= \dot{\boldsymbol{Q}} \circ \underbrace{\overline{\boldsymbol{Q}} \circ \boldsymbol{P'} \circ \boldsymbol{Q}}_{\boldsymbol{P}} \circ \overline{\boldsymbol{Q}} + \boldsymbol{Q} \circ \underbrace{\overline{\boldsymbol{Q}} \circ \boldsymbol{P'} \circ \boldsymbol{Q}}_{\boldsymbol{P}} \circ \dot{\overline{\boldsymbol{Q}}}$$

$$= \dot{\boldsymbol{Q}} \circ \overline{\boldsymbol{Q}} \circ \boldsymbol{P'} + \boldsymbol{P'} \circ \boldsymbol{Q} \circ \dot{\overline{\boldsymbol{Q}}}$$

As $\boldsymbol{Q} \circ \overline{\boldsymbol{Q}} = 1$

Taking derivative yields

$$\dot{\boldsymbol{Q}} \circ \overline{\boldsymbol{Q}} + \boldsymbol{Q} \circ \dot{\overline{\boldsymbol{Q}}} = 0 \Rightarrow \left.\begin{array}{l} \boldsymbol{Q} \circ \dot{\overline{\boldsymbol{Q}}} = -\dot{\boldsymbol{Q}} \circ \overline{\boldsymbol{Q}} \\ \boldsymbol{Q} \circ \dot{\overline{\boldsymbol{Q}}} = \overline{\dot{\boldsymbol{Q}} \circ \overline{\boldsymbol{Q}}} \end{array}\right\} \dot{\boldsymbol{Q}} \circ \overline{\boldsymbol{Q}} = 0 + \boldsymbol{a}; \boldsymbol{Q} \circ \dot{\overline{\boldsymbol{Q}}} = 0 - \boldsymbol{a}$$

Notice

$$\dot{\boldsymbol{P}}' = (0 + \boldsymbol{a}) \circ \boldsymbol{P'} + \boldsymbol{P'} \circ (0 - \boldsymbol{a})$$

Quaternion Kinematics

$$\dot{\boldsymbol{P}}' = (0 + \boldsymbol{a}) \circ \boldsymbol{P}' + \boldsymbol{P}' \circ (0 - \boldsymbol{a})$$

$$= \begin{bmatrix} 0 \\ \boldsymbol{a} \end{bmatrix} \circ \begin{bmatrix} 0 \\ \boldsymbol{p}' \end{bmatrix} + \begin{bmatrix} 0 \\ \boldsymbol{p}' \end{bmatrix} \circ \begin{bmatrix} 0 \\ -\boldsymbol{a} \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ \boldsymbol{a} \times \boldsymbol{p}' \end{bmatrix} - \boldsymbol{a} \cdot \boldsymbol{p}' + \begin{bmatrix} 0 \\ \boldsymbol{p}' \times (-\boldsymbol{a}) \end{bmatrix} - \boldsymbol{p}' \cdot (-\boldsymbol{a})$$

$$\begin{bmatrix} 0 \\ \dot{\boldsymbol{p}}' \end{bmatrix} = \begin{bmatrix} 0 \\ 2\boldsymbol{a} \times \boldsymbol{p}' \end{bmatrix}$$

Recall $\dot{\boldsymbol{p}}' = \boldsymbol{\omega} \times \boldsymbol{p}'$, $\boldsymbol{\omega}$ is angular velocity vector represented in the world frame

Then $2\boldsymbol{a} = \boldsymbol{\omega} \Rightarrow \boldsymbol{a} = \frac{1}{2}\boldsymbol{\omega}$

$$\Rightarrow \dot{\boldsymbol{Q}} \circ \overline{\boldsymbol{Q}} = \begin{bmatrix} 0 \\ \boldsymbol{a} \end{bmatrix} = \frac{1}{2} \underbrace{\begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}}_{\boldsymbol{\Omega}} \quad \Rightarrow \dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{\Omega} \circ \boldsymbol{Q}$$

$$\boldsymbol{Q} = \begin{bmatrix} q_0 \\ \boldsymbol{q} \end{bmatrix} \Rightarrow \dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{\Omega} \circ \boldsymbol{Q} = \mathcal{L}(\boldsymbol{\Omega})\boldsymbol{Q} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & \widehat{\boldsymbol{\omega}} \end{bmatrix}\boldsymbol{Q}$$

Quaternion Kinematics

$$\dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{\Omega} \circ \boldsymbol{Q}$$

$$\boldsymbol{\Omega} = \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix}$$

$\boldsymbol{\omega}$ is angular velocity represented in the world frame

- What if $\boldsymbol{\omega}$ is represented in the body frame (i.e. $\boldsymbol{\omega}^b$)?

$$\boldsymbol{\Omega}^b = \begin{bmatrix} 0 \\ \boldsymbol{\omega}^b \end{bmatrix}$$ is represented in body frame

$$\Rightarrow \quad \boldsymbol{\Omega}^b = \overline{\boldsymbol{Q}} \circ \boldsymbol{\Omega} \circ \boldsymbol{Q}$$

$$\Rightarrow \quad \boxed{\dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{Q} \circ \boldsymbol{\Omega}^b}$$

$$\boldsymbol{Q} = \begin{bmatrix} q_0 \\ \boldsymbol{q} \end{bmatrix}$$

$$\Rightarrow \dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{Q} \circ \boldsymbol{\Omega}^b = \mathcal{R}(\boldsymbol{\Omega}^b)\boldsymbol{Q} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^{b\,T} \\ \boldsymbol{\omega}^b & -\widehat{\boldsymbol{\omega}^b} \end{bmatrix}\boldsymbol{Q}$$

Unit Quaternion and Rotation

- Rotation to quaternion: $\boldsymbol{Q} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}\boldsymbol{u} \end{bmatrix}$, where $\boldsymbol{R} = e^{\hat{\boldsymbol{u}}\theta}$
- Quaternion to rotation
- $\boldsymbol{R} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_0q_2 + q_1q_3) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(-q_0q_1 + q_2q_3) \\ 2(-q_0q_2 + q_1q_3) & 2(q_0q_1 + q_2q_3) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}$
- Rotating a vector: $\boldsymbol{P'} = \boldsymbol{Q} \circ \boldsymbol{P} \circ \overline{\boldsymbol{Q}}$
- Rotating a frame: $\boldsymbol{P}^b = \overline{\boldsymbol{Q}} \circ \boldsymbol{P} \circ \boldsymbol{Q}$
- Sequential rotation (extrinsic): $\boldsymbol{Q} = \boldsymbol{Q}_n \ldots \circ \boldsymbol{Q}_2 \circ \boldsymbol{Q}_1$
- Sequential rotation (intrinsic): $\boldsymbol{Q} = \boldsymbol{Q}_1 \circ \boldsymbol{Q}_2 \ldots \circ \boldsymbol{Q}_n$
- Kinematics under spatial frame: $\dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{\Omega} \circ \boldsymbol{Q} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & \hat{\boldsymbol{\omega}} \end{bmatrix}\boldsymbol{Q}$
- Kinematics under body frame: $\dot{\boldsymbol{Q}} = \frac{1}{2}\boldsymbol{Q} \circ \boldsymbol{\Omega} = \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -\hat{\boldsymbol{\omega}} \end{bmatrix}\boldsymbol{Q}$

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 3

Shaojie Shen
Associate Professor
Dept. of ECE, HKUST

21 February 2023

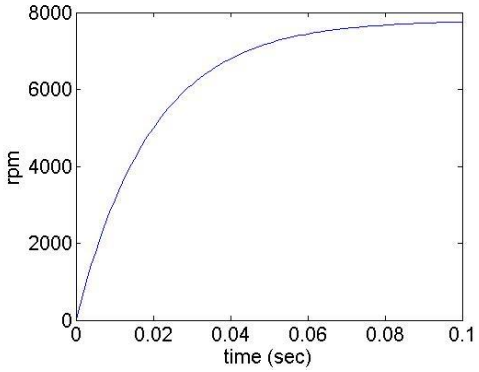
Outline

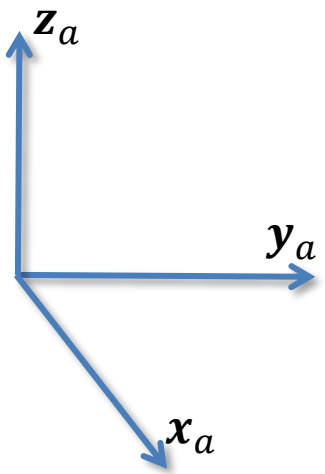
- Quadrotor Dynamics
- Control Basics
- Quadrotor Control
- Trajectory Generation

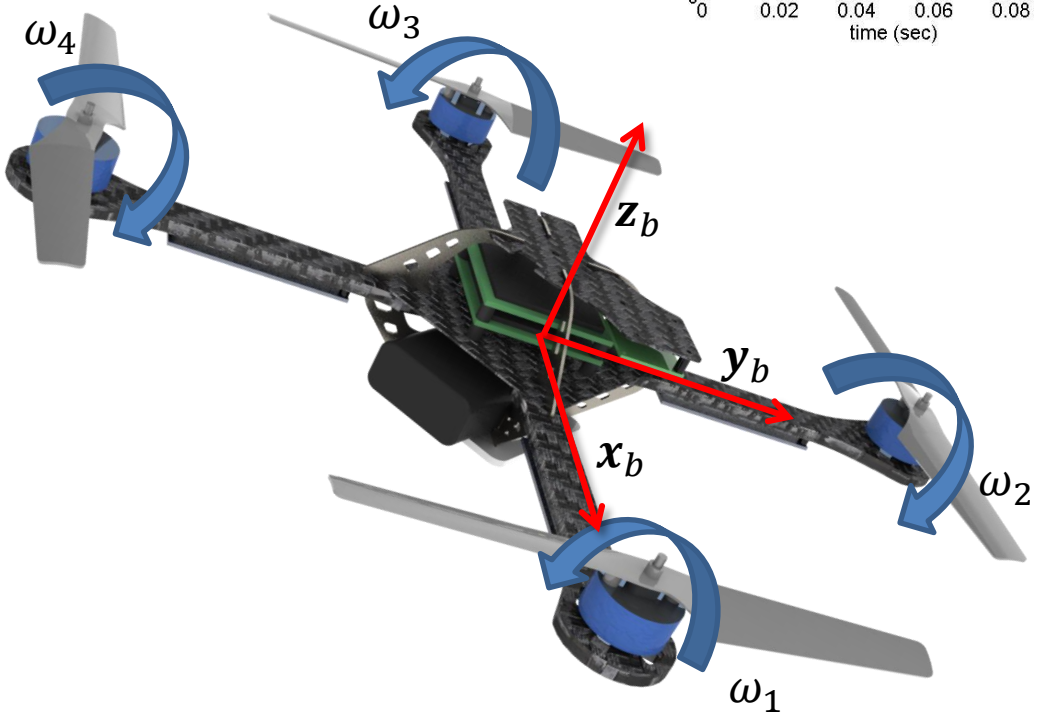
Quadrotor Dynamics

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: $M_i = k_M\omega_i^2$

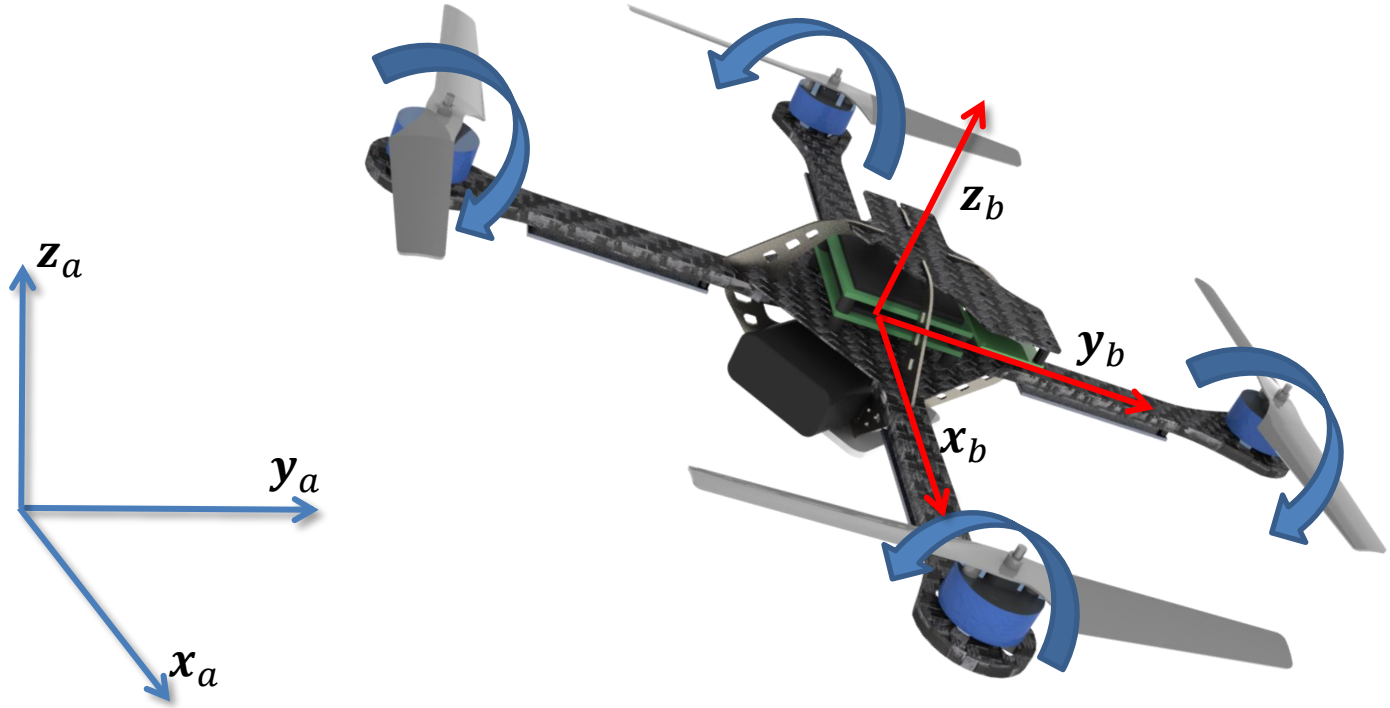






Quadrotor Dynamics

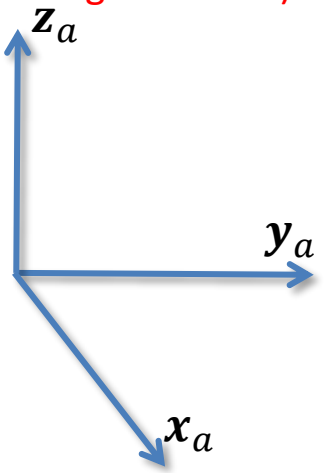
- Z-X-Y Euler Angles: $\boldsymbol{R}_{ab} = \boldsymbol{R}_z(\psi) \cdot \boldsymbol{R}_x(\phi) \cdot \boldsymbol{R}_y(\theta)$
- Sequence of three rotations about body-fixed axes
- What are the singularities?

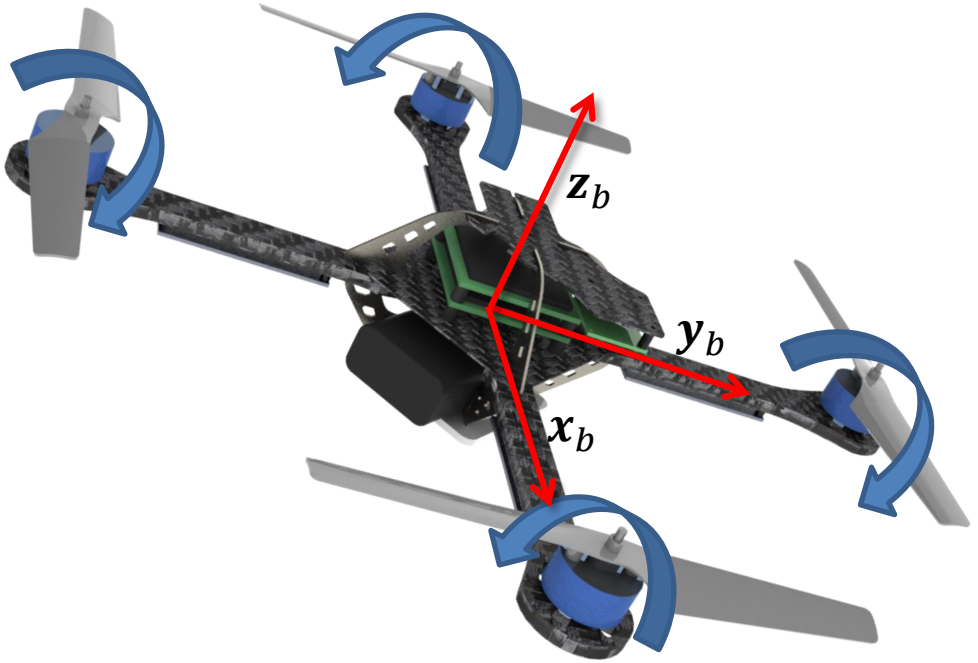


Quadrotor Dynamics

- $\boldsymbol{R}_{ab} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$
- $\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$

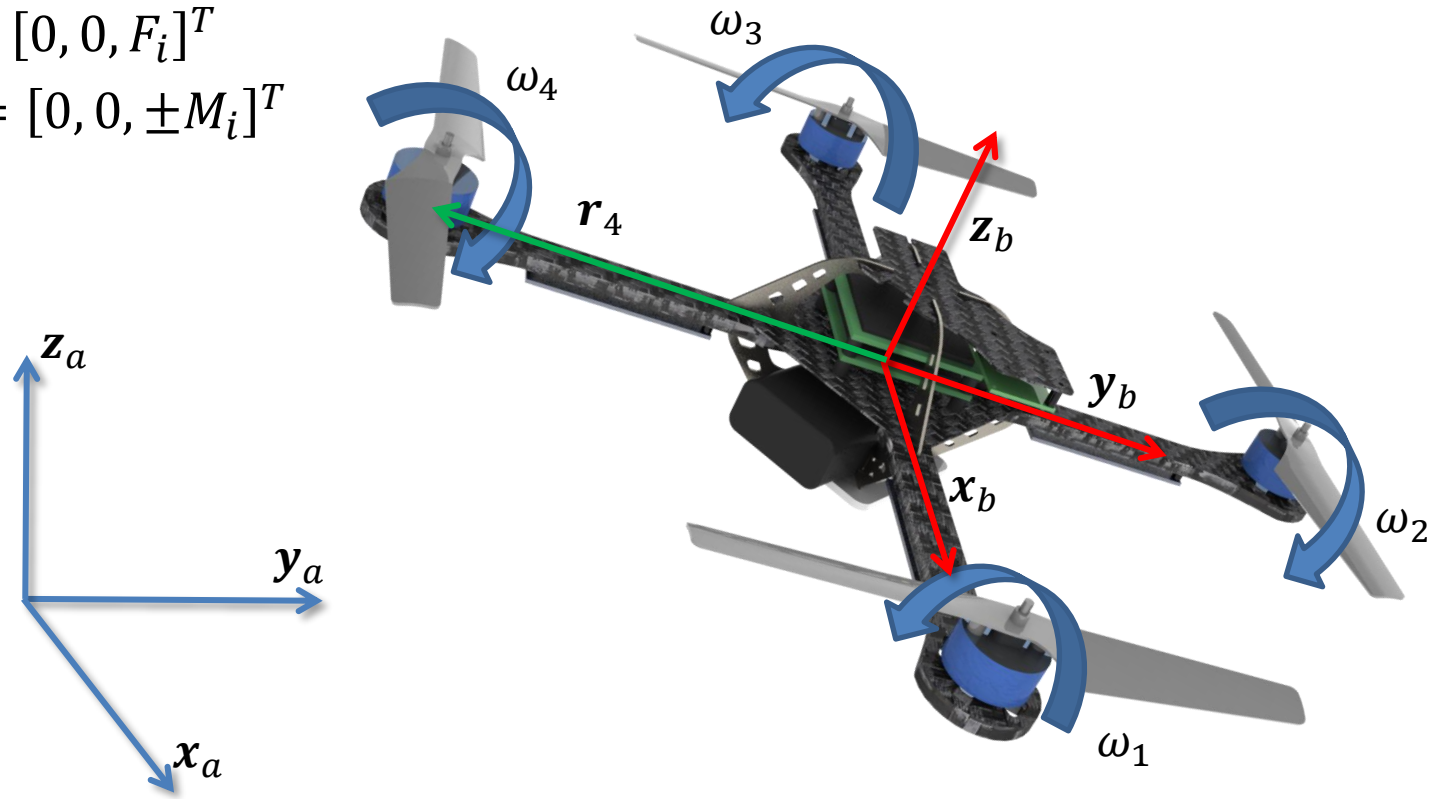
Instantaneous body angular velocity viewed in the body frame
(sorry for abusing notations)





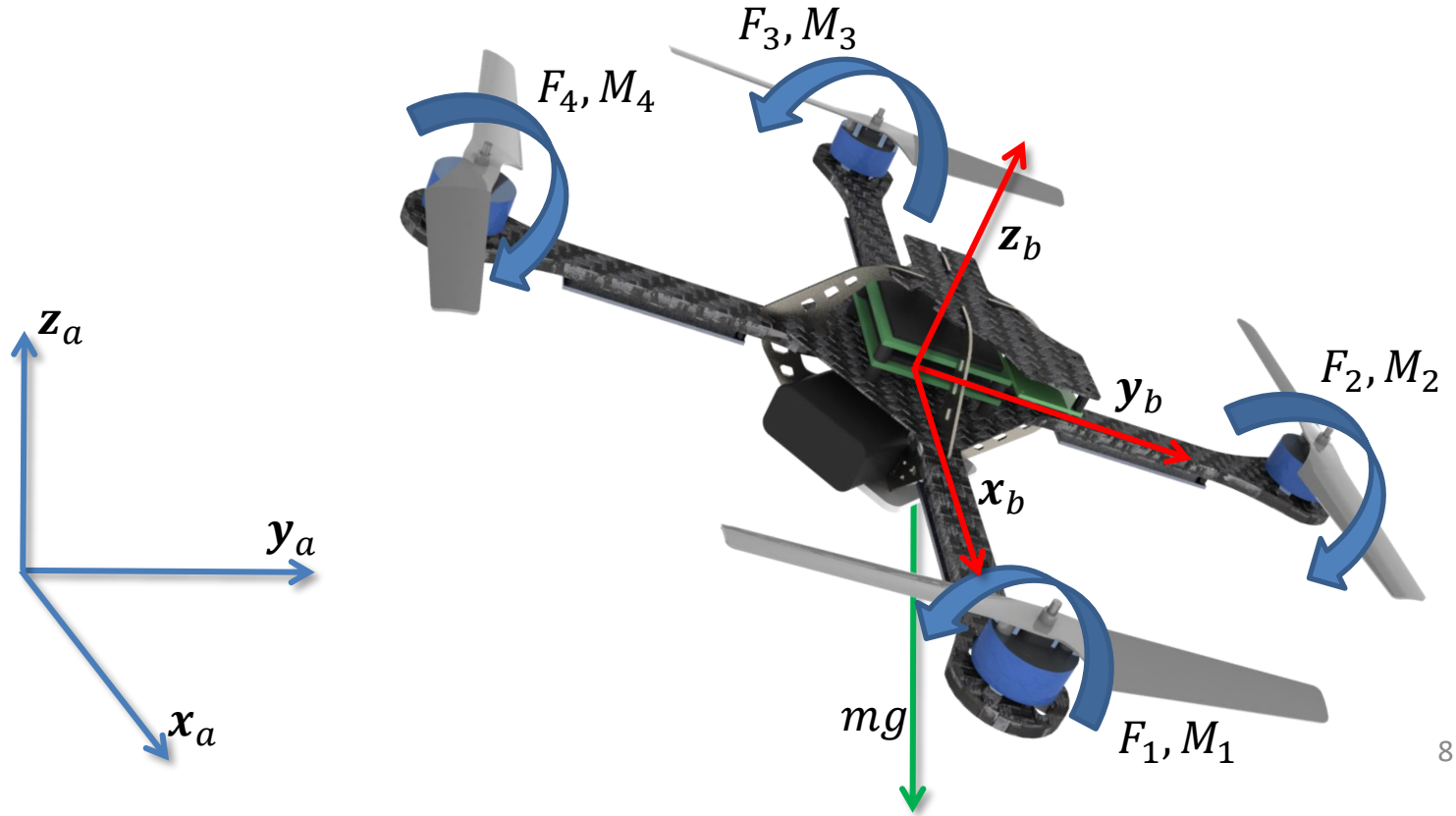
Quadrotor Dynamics

- Consider body frame
  - $\boldsymbol{F} = \boldsymbol{F}_1 + \boldsymbol{F}_2 + \boldsymbol{F}_3 + \boldsymbol{F}_4 - \boldsymbol{R}_{ab}^T[0, 0, mg]^T$
  - $\boldsymbol{M} = \boldsymbol{r}_1 \times \boldsymbol{F}_1 + \boldsymbol{r}_2 \times \boldsymbol{F}_2 + \boldsymbol{r}_3 \times \boldsymbol{F}_3 + \boldsymbol{r}_4 \times \boldsymbol{F}_4 + \boldsymbol{M}_1 + \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_4$
  - $\boldsymbol{F}_i = [0, 0, F_i]^T$
  - $\boldsymbol{M}_i = [0, 0, \pm M_i]^T$



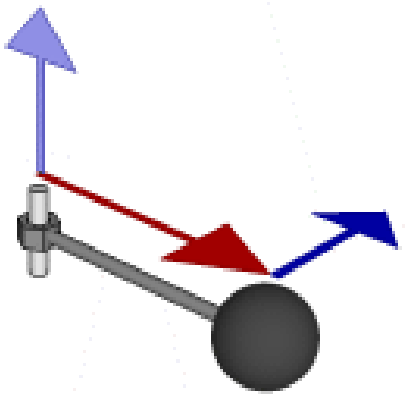
Newton-Euler Equations

- Newton Equation: $m\ddot{\boldsymbol{p}}^a = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R}_{ab} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$



Newton-Euler Equations

$$\tau = \mathbf{r} \times \mathbf{F}$$
$$\mathbf{L} = \mathbf{r} \times \mathbf{p}$$

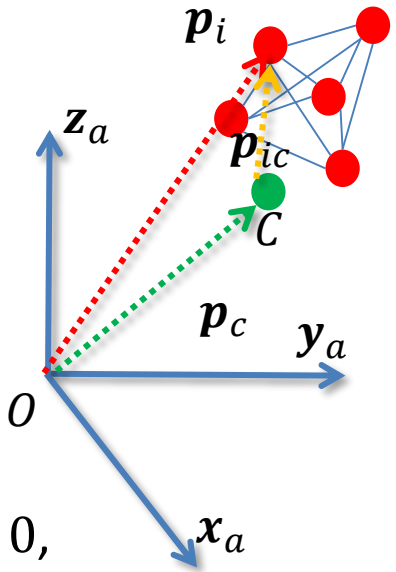


Relationship between force (F), torque/moment of force(τ), momentum (p), and angular momentum (L) vectors in a rotating system. r is the position vector.

Newton-Euler Equations

- The rigid body as a collection of particles
  - Center of mass (CoM): $\boldsymbol{p}_c$
  - Position of the i-th particle to CoM: $\boldsymbol{p}_{ic} = \boldsymbol{p}_i - \boldsymbol{p}_c$
  - Velocity of the i-th particle to CoM: $\boldsymbol{v}_{ic} = \dot{\boldsymbol{p}}_i - \dot{\boldsymbol{p}}_c = \boldsymbol{v}_i - \boldsymbol{v}_c$
  - Angular momentum of the i-th particle:

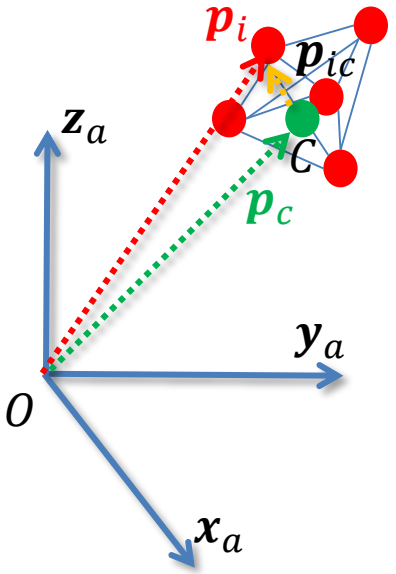
$$\boldsymbol{H}_i = \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$$



- Angular momentum of the rigid body:
  - $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$
  - Since: $\sum m_i \boldsymbol{p}_{ic} = \sum m_i (\boldsymbol{p}_i - \boldsymbol{p}_c) = \sum m_i \boldsymbol{p}_i - \boldsymbol{p}_c \sum m_i = 0,$
  - We have: $\sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_c = (\sum m_i \boldsymbol{p}_{ic}) \times \boldsymbol{v}_c = 0$
  - Therefore: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i - \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_c = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_{ic}$
  - Since: $\boldsymbol{v}_{ic} = \boldsymbol{\omega} \times \boldsymbol{p}_{ic},$
  - We have: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times (\boldsymbol{\omega} \times m_i \boldsymbol{p}_{ic}) = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$

Newton-Euler Equations

- Rotational dynamics
  - Angular momentum: $\boldsymbol{H} = \sum \boldsymbol{p}_{ic} \times m_i \boldsymbol{v}_i$
  - Take the derivative: $\dot{\boldsymbol{H}} = \sum \dot{\boldsymbol{p}}_{ic} \times m_i \boldsymbol{v}_i + \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i$
  - Since $\sum \dot{\boldsymbol{p}}_{ic} \times m_i \boldsymbol{v}_i = \sum \boldsymbol{v}_i \times m_i \boldsymbol{v}_i - \boldsymbol{v}_c \times m_i \boldsymbol{v}_i = \sum -\boldsymbol{v}_c \times m_i \boldsymbol{v}_i = -\boldsymbol{v}_c \times \frac{d}{dt} \sum m_i \boldsymbol{p}_i = -\boldsymbol{v}_c \times \frac{d}{dt} \boldsymbol{p}_c \sum m_i = 0$
  - We have $\dot{\boldsymbol{H}} = \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i$
  - Referring to Newton's second law: $\boldsymbol{F}_i + \sum_{i \neq j} \boldsymbol{F}_{ij} = m_i \dot{\boldsymbol{v}}_i$
  - $\dot{\boldsymbol{H}} = \sum \boldsymbol{p}_{ic} \times m_i \dot{\boldsymbol{v}}_i = \sum \boldsymbol{p}_{ic} \times (\boldsymbol{F}_i + \sum_{i \neq j} \boldsymbol{F}_{ij}) = \sum \boldsymbol{p}_{ic} \times \boldsymbol{F}_i$
  - We also know that the external moment: $\boldsymbol{M} = \sum \boldsymbol{p}_{ic} \times \boldsymbol{F}_i$
  - We have the rotational dynamics: $\boldsymbol{M} = \dot{\boldsymbol{H}}$



Newton-Euler Equations

- Finishing the work on rotational dynamics
  - Given: $\boldsymbol{H} = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$
  - And using the fact: $(\boldsymbol{R}\boldsymbol{a}) \times (\boldsymbol{R}\boldsymbol{b}) = \boldsymbol{R}(\boldsymbol{a} \times \boldsymbol{b})$
    - $\boldsymbol{R}$: rotation matrix
    - $\boldsymbol{a}, \boldsymbol{b}$: vectors
  - We can transform the representation of the angular momentum to the body frame with constant inertia matrix:

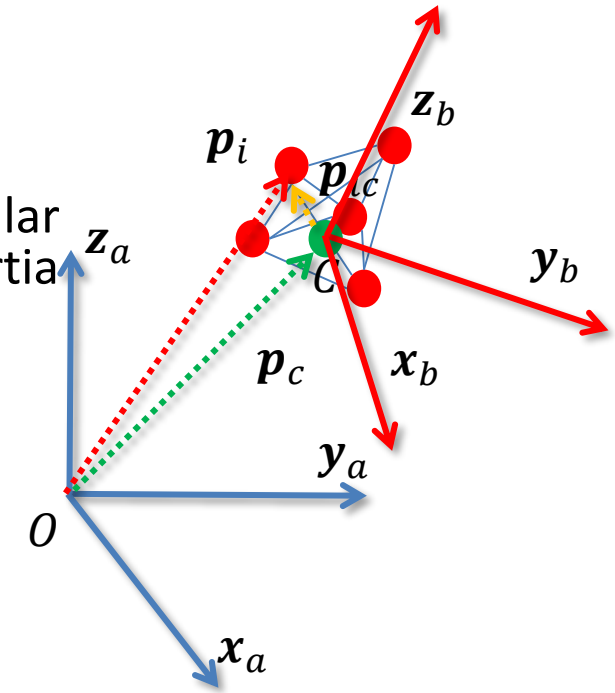
$$\boldsymbol{H} = -\sum \boldsymbol{p}_{ic} \times (m_i \boldsymbol{p}_{ic} \times \boldsymbol{\omega})$$

$$= -\sum \boldsymbol{R}_{ab} \boldsymbol{p}_{ic}^b \times \left(m_i \boldsymbol{R}_{ab} \boldsymbol{p}_{ic}^b \times \boldsymbol{R}_{ab} \boldsymbol{\omega}^b\right)$$

$$= -\boldsymbol{R}_{ab} \sum \boldsymbol{p}_{ic}^b \times \left(m_i \boldsymbol{p}_{ic}^b \times \boldsymbol{\omega}^b\right)$$

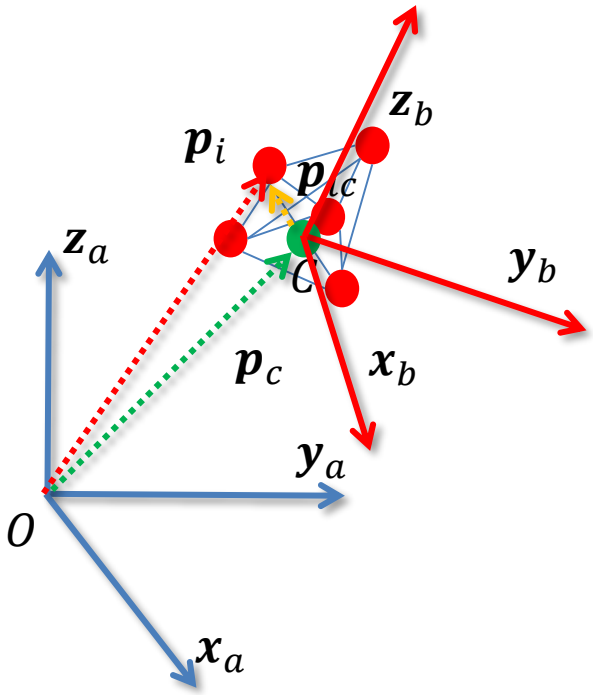
$$= -\boldsymbol{R}_{ab} \sum m_i \cdot \boldsymbol{p}_{ic}^b \times \left(\widehat{\boldsymbol{p}}_{ic}^b \cdot \boldsymbol{\omega}^b\right)$$

$$= \boldsymbol{R}_{ab}\left(-\sum m_i \cdot \widehat{\boldsymbol{p}}_{ic}^b \cdot \widehat{\boldsymbol{p}}_{ic}^b\right) \cdot \boldsymbol{\omega}^b = \boldsymbol{R}_{ab}(\boldsymbol{I}^b \boldsymbol{\omega}^b)$$



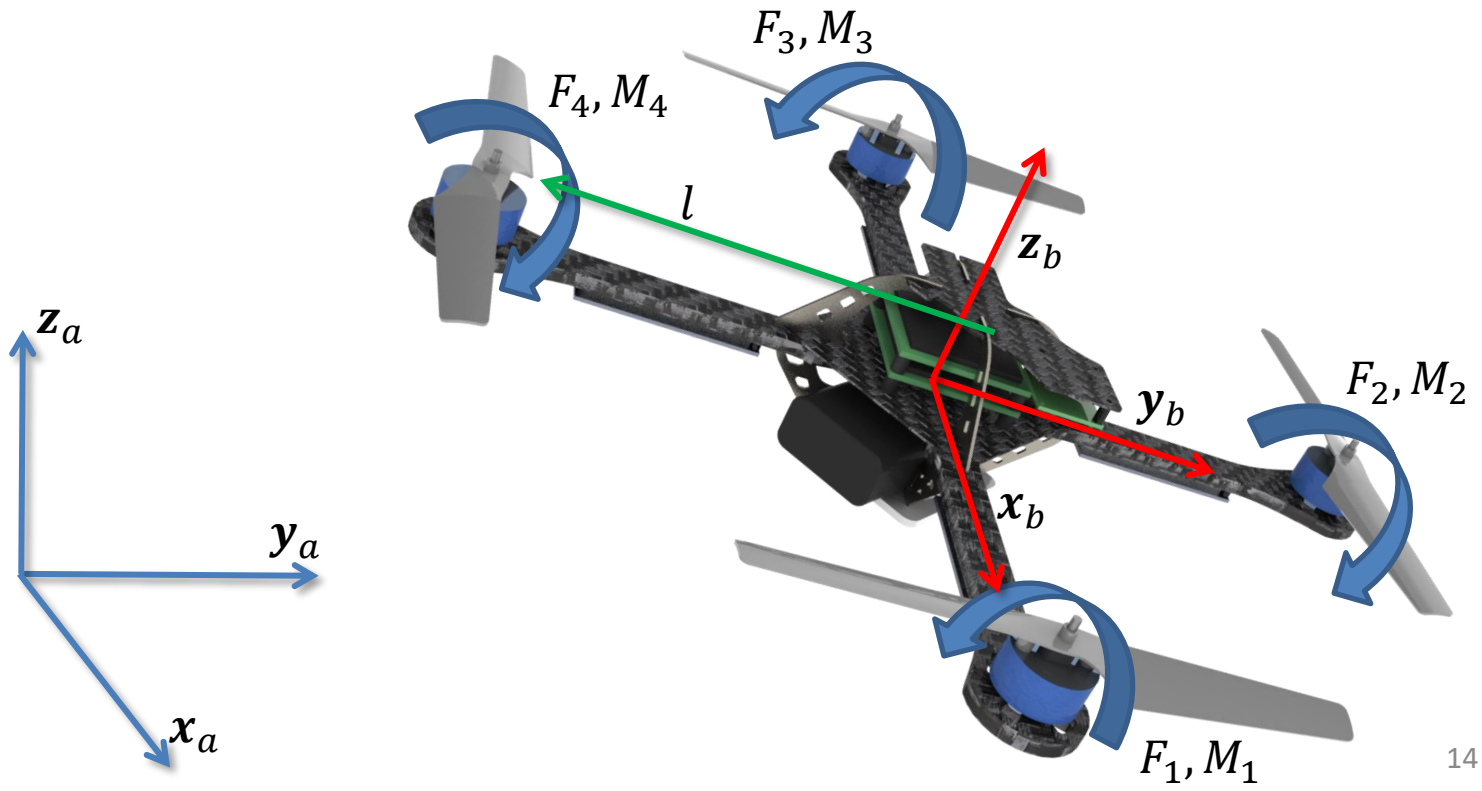
Newton-Euler Equations

- Finishing the work on rotational dynamics
  - Given $\boldsymbol{H} = \boldsymbol{R}_{ab}(\boldsymbol{I}^b\boldsymbol{\omega}^b)$
  - Take the derivative: $\dot{\boldsymbol{H}} = \dot{\boldsymbol{R}}_{ab}\boldsymbol{I}^b\boldsymbol{\omega}^b + \boldsymbol{R}_{ab}\boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b = \boldsymbol{R}_{ab}\hat{\boldsymbol{\omega}}^b\boldsymbol{I}^b\boldsymbol{\omega}^b + \boldsymbol{R}_{ab}\boldsymbol{I}^b\dot{\omega}^b = \boldsymbol{R}_{ab}(\boldsymbol{\omega}^b \times (\boldsymbol{I}^b\boldsymbol{\omega}^b) + \boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b)$
  - Also transform the moment into body frame: $\boldsymbol{M} = \boldsymbol{R}_{ab}\boldsymbol{M}^b$
  - Finally: $\boldsymbol{M}^b = \boldsymbol{\omega}^b \times (\boldsymbol{I}^b\boldsymbol{\omega}^b) + \boldsymbol{I}^b\dot{\boldsymbol{\omega}}^b$



Newton-Euler Equations

- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$

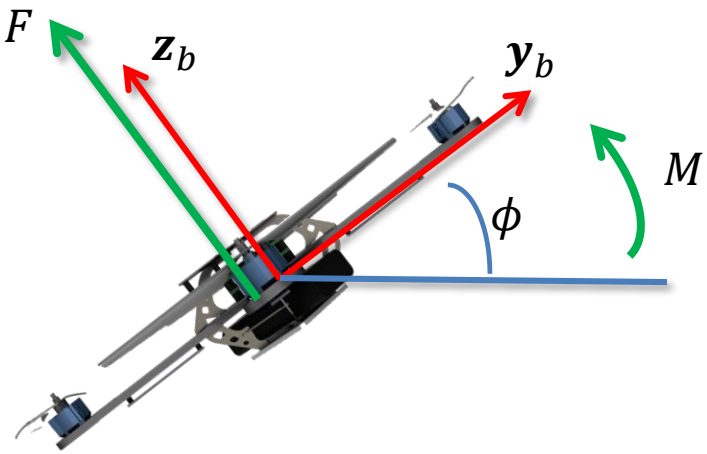


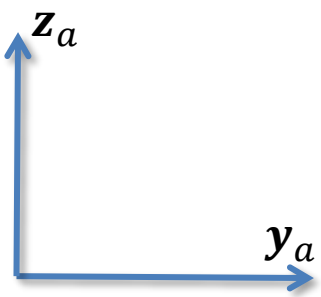
Quadrotor Dynamics

- 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: $m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$
- Euler Equation: $\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$

A Planar Quadrotor

- $$\begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} F \\ M \end{bmatrix}$$





ω_1

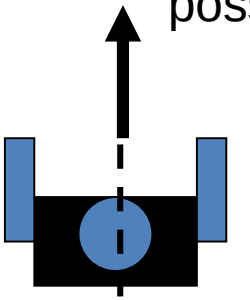
Control System Design

Control System Design

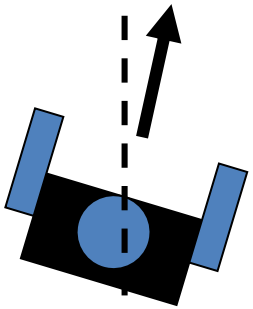
- Introduction of control systems
- Linear Time Invariant (LTI) systems
  - Simple first-order system
  - Simple second-order system
- Controller design
  - Gain tuning
  - Model-based control

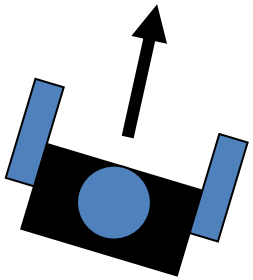
Introduction of control system

A robot needs to move in a straight line or close to it as much as possible



If the movement of the robot gets disturbed,





the robot will deviate from its destination

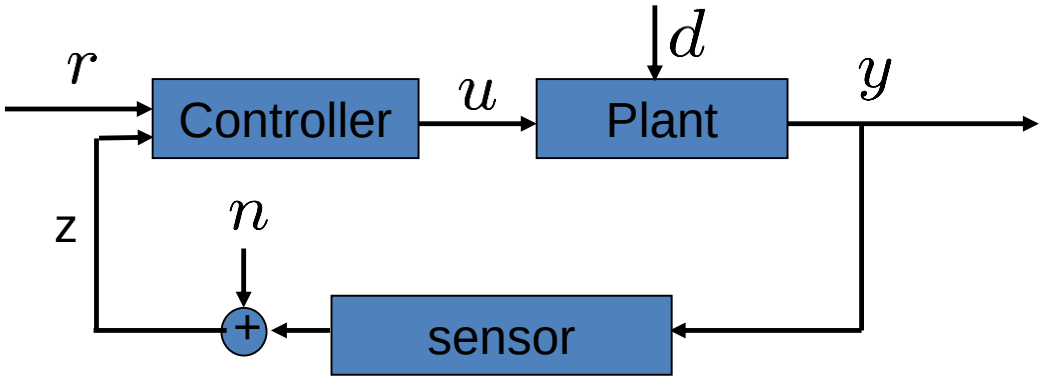
- Therefore we need to have a controller to control its movement in real time based on its movement and the destination

Introduction of control system

- Open loop control
  - Move the robot in a pre-determined way
  - Example: walking with your eyes closed
- Closed loop (feedback) control
  - Use the output (i.e. the location of the robot) to adjust the input (i.e. the direction and may be speed) to the movement of the robot
  - We also call it feedback control, since we make the control decision based on the output feedback
  - Example: walking with your eyes open
- We want to stabilize a system with closed loop control

Introduction of control system

- One objective of control is to make the plant stable and track a given reference signal as precise and swift as possible



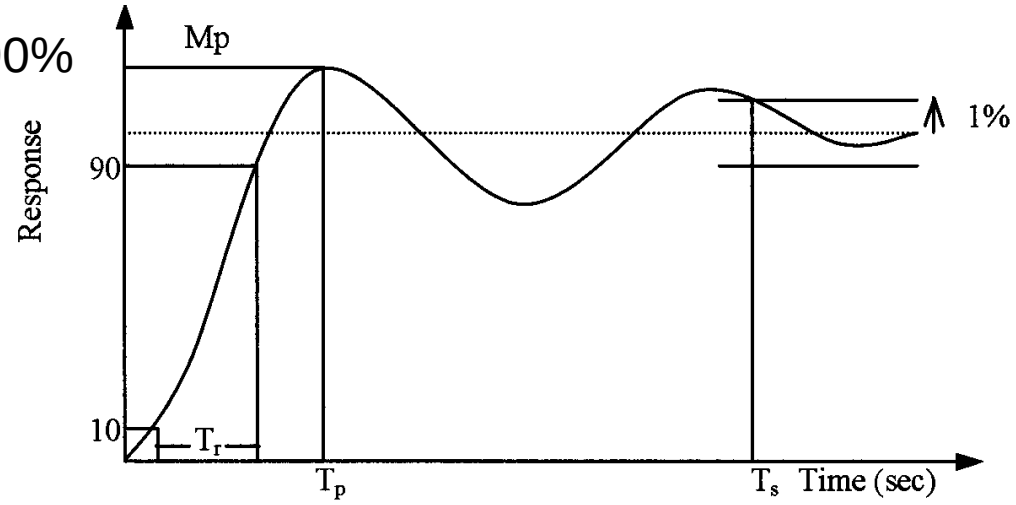
r: reference input
z: state feedback
u: controller output
d: plant disturbance
y: output
n: communication noise

- A controller is simply a computation unit that computes the "optimal" or "desired" input to the plant

"Feedback is a method of controlling a system by inserting into it the result of its past performance"

Introduction of control system

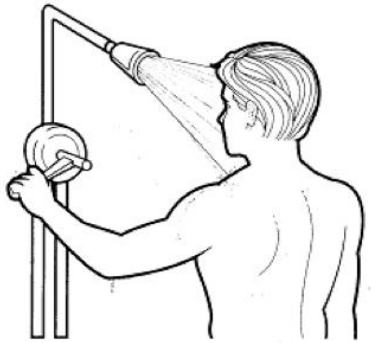
- Rise time:
  - Time it takes from 10% to 90%
- Steady-state error
- Overshoot
  - Percentage by which peak exceeds final value

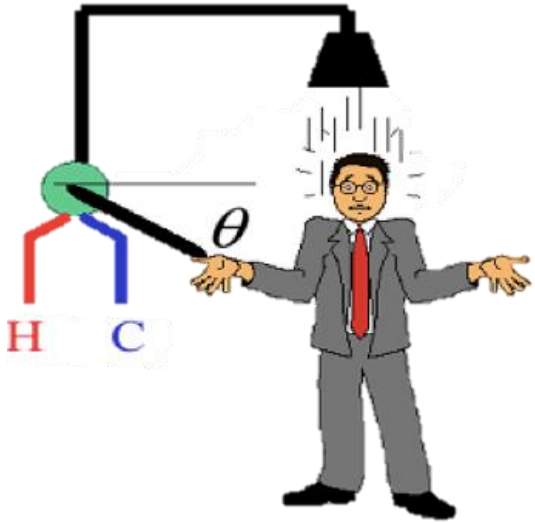


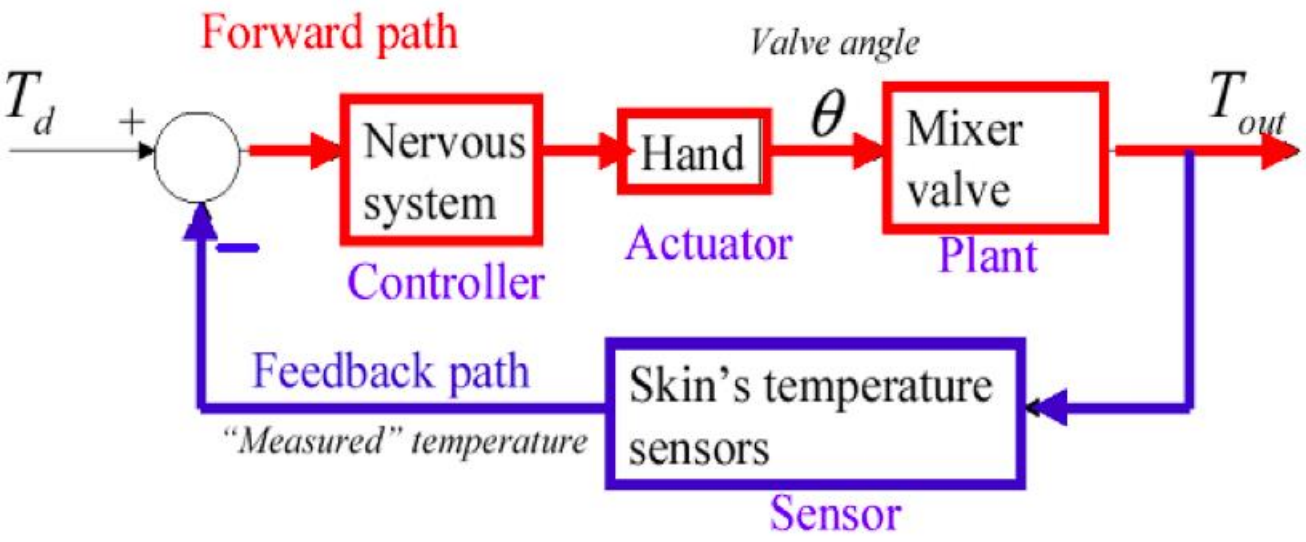
- Settling time
  - Time it takes to reach 1% of final value
- A good control system has small rise time, overshoot, settling time and steady-state error

Introduction of control system

When taking a shower



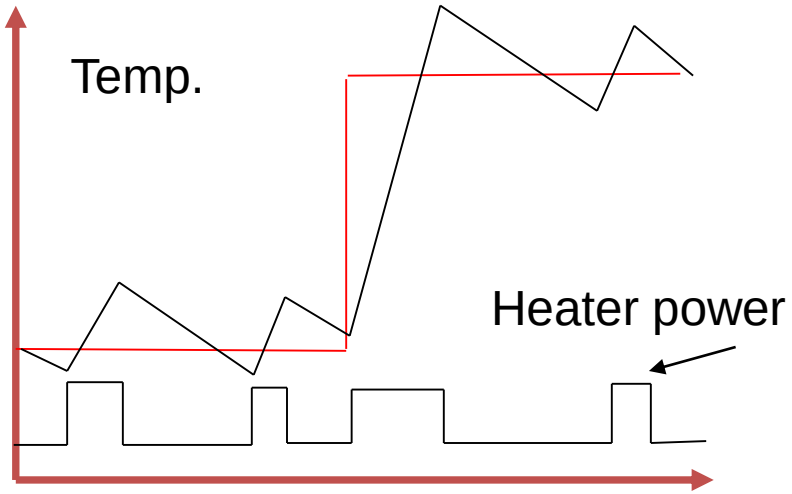




Introduction of control system

- Example: shower water temperature control
  - Turn the heater on if T_{water} is below certain value
  - Turn the heater off if T_{water} is above certain value

- Simple
- Transition is not smooth



Control of a simple first-order system

- Problem

  State, input

  $$x, u \in \boldsymbol{R}$$

  Kinematic plant model

  $$\dot{x} = u$$

  Want x to follow trajectory $x^{des}(t)$

- General Approach

  Define error, $e(t) = x^{des}(t) - x(t)$

  Want $e(t)$ to converge exponentially to zero

- Strategy

  Find u such that

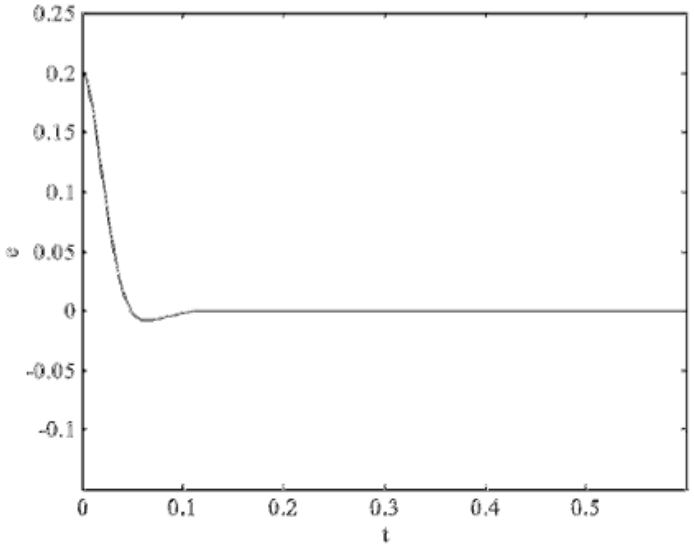
  $$\dot{e} + K_p e = 0 \qquad K_p > 0$$

  $$u(t) = \dot{x}^{des}(t) + K_p e(t)$$

  Feed forward Proportional

Control of a simple second-order system

- Problem
  - State, input
    $$x, u \in \boldsymbol{R}$$
  - Kinematic plant model
    $$\ddot{x} = u$$
  - Want x to follow trajectory $x^{des}(t)$



- General Approach
  - Define error, $e(t) = x^{des}(t) - x(t)$
  - Want $e(t)$ to converge exponentially to zero
- Strategy
  - Find u such that

$$\ddot{e} + K_d\dot{e} + K_p e = 0 \qquad K_d, K_p > 0$$

$$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)$$

Feed forward Derivative Proportional

PD control and PID control

- PD control

$$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)$$

Proportional control acts like a spring (capacitance) response

Derivative control is a viscous dashpot (resistance) response

Large derivative gain makes the system overdamped and the system converges slowly

- PID control

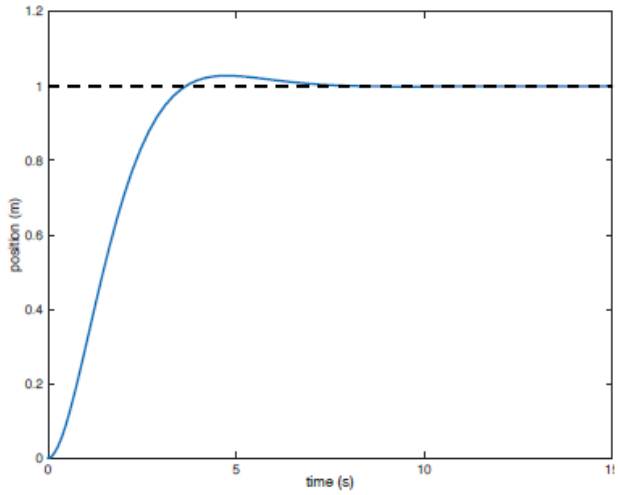
$$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t) + K_I \int_0^t e(\tau)d\tau$$

Integral

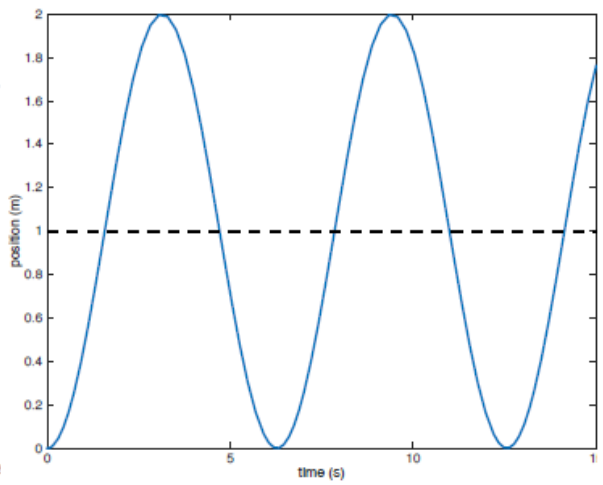
PID control generates a third-order closed-loop system

Integral control makes the steady-state error go to zero

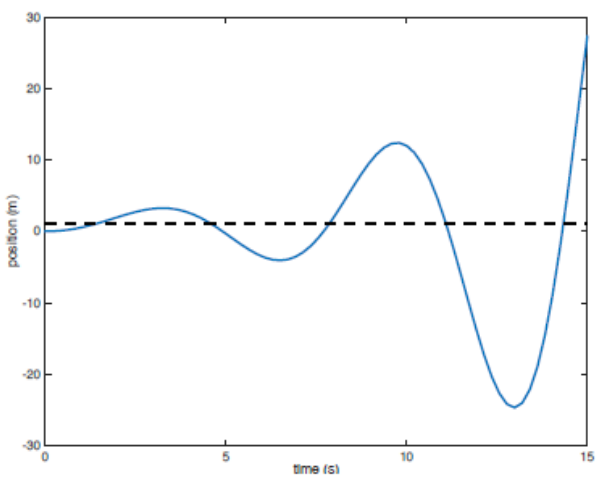
Gain Tuning



Stable
(converge)

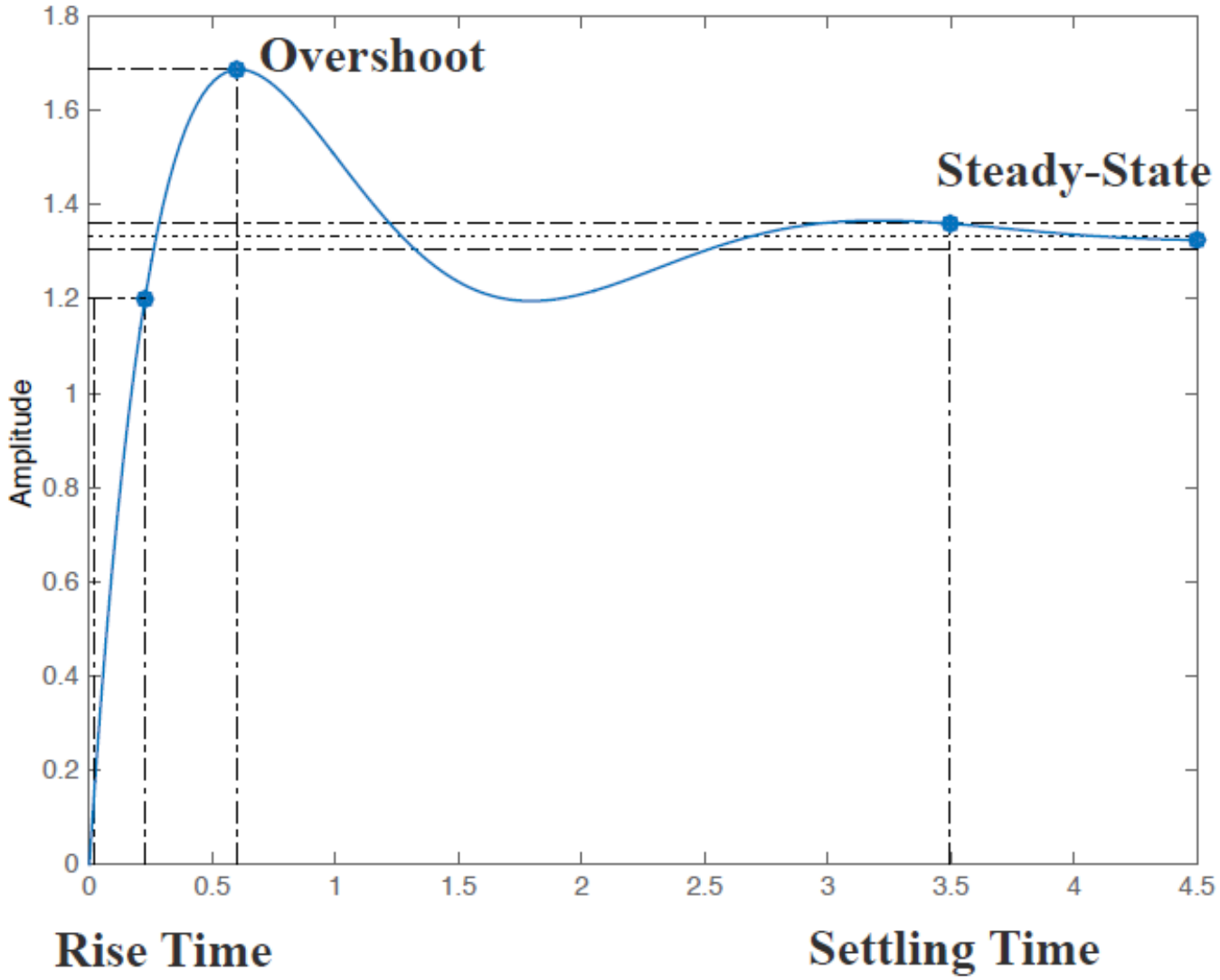


Marginally Stable
(oscillate)



Unstable
(diverge)

Manual Tuning

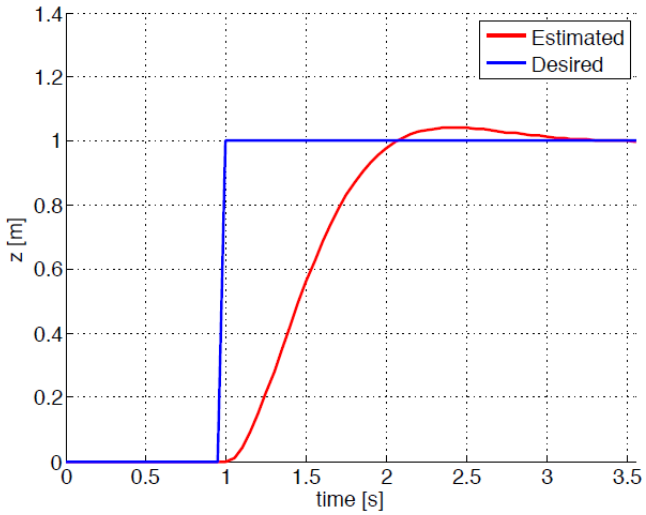


Manual Tuning

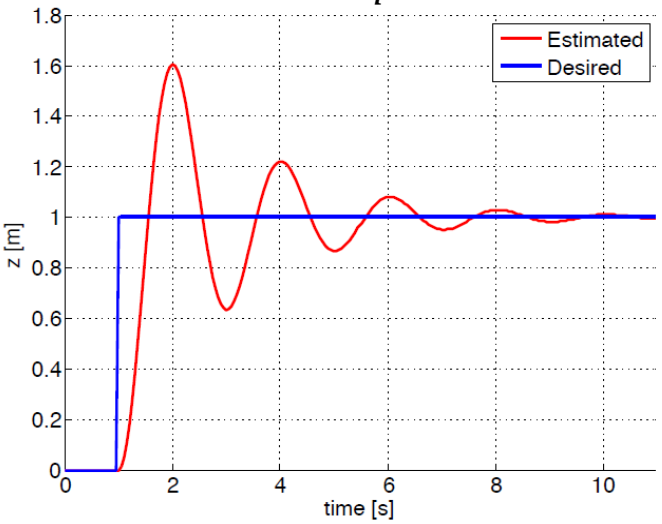
Parameter Increased	$K_p \uparrow$	$K_d \uparrow$	$K_i \uparrow$
Rise Time	Decrease	-	Decrease
Overshoot	Increase	Decrease	Increase
Settling Time	-	Decrease	Increase
Steady-State Error	Decrease	-	Eliminate

Manual Tuning

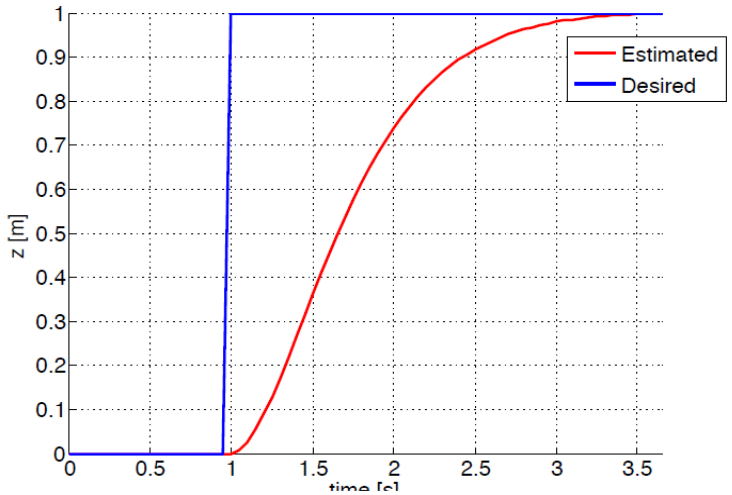
PD controller



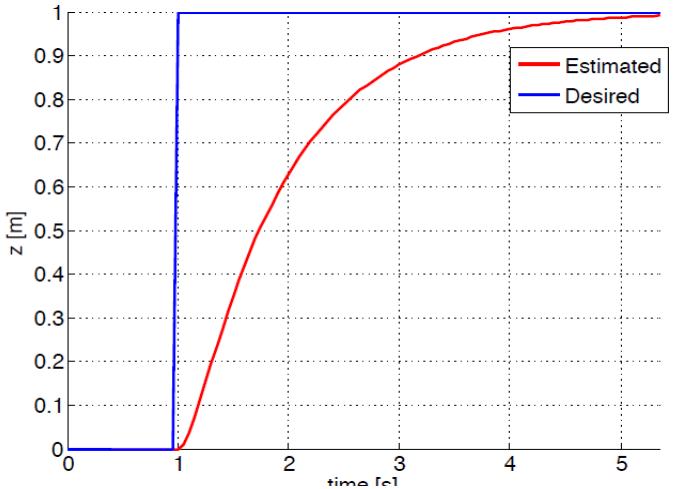
High K_p(overshoot)



Low K_p(soft response)



High K_d(overdamped)



Ziegler-Nichols Method

- Heuristic for tuning gains
  1. Set $K_i = K_d = 0$
  2. Increase K_p until ultimate gain, K_u, when output starts to oscillate
  3. Find the oscillation period T_u at K_u
  4. Set gains according to:

Controller	K_p	K_d	K_i
P	$0.5K_u$	-	-
PD	$0.8K_u$	$K_pT_u/8$	-
PID	$0.6K_u$	$K_pT_u/8$	$2K_p/T_u$

Model-based control

- Consider a general second-order model

$$m\ddot{x}(t) + b\dot{x}(t) + kx(t) = u(\mathrm{t})$$

- Disadvantages of PID or PD control schemes

$$u(t) = \ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)$$

  - Performance will depend on the model
  - Need to tune gains to maximize performance

- Model based control law

Model based

$$u(t) = \hat{m}(\ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)) + \hat{b}\dot{x}(t) + \hat{k}x(t)$$

Servo: feedforward + PD feedback

  - Model based part
    - Cancel the dynamics of the system
    - Specific to the model
  - Servo based part
    - Use PID or PD with feedforward to drive errors to zero
    - Independent of the model of the system

Model-based control

- Model based control law

Model based (estimates)

$$u(t) = \hat{m}(\ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)) + \hat{b}\dot{x}(t) + \hat{k}x(t)$$

Servo: feedforward + PD feedback

- Advantage
  - Decomposes the control law into
    - Model-dependent part (depends on the knowledge of the model)
    - Model-independent part (servo control, gains are independent of the model)
- Disadvantage
  - Based on estimates of model parameters
    - Ideal performance

$$\ddot{e} + K_d\dot{e} + K_p e = 0$$

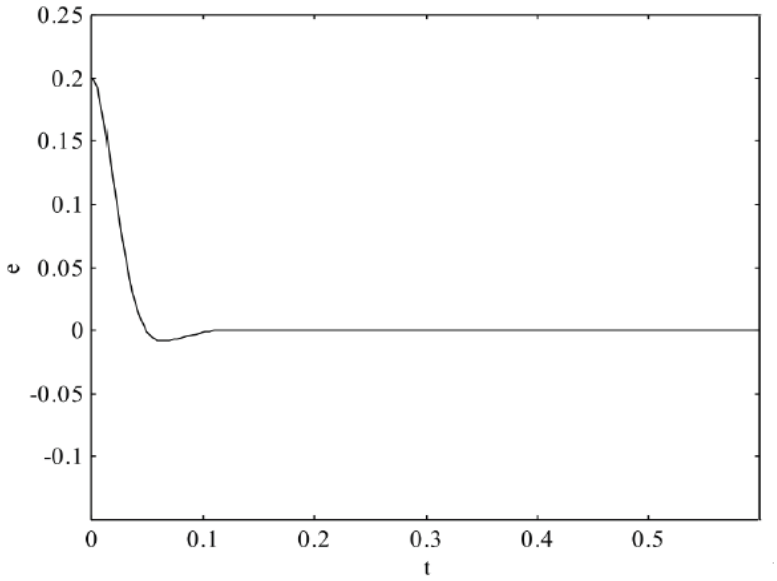
    - Actual performance

$$\ddot{e} + K_d\dot{e} + K_p e = \left(\frac{m}{\hat{m}} - 1\right)\ddot{x} + \frac{(b-\hat{b})}{\hat{m}}\dot{x} + \frac{(k-\hat{k})}{\hat{m}}x$$

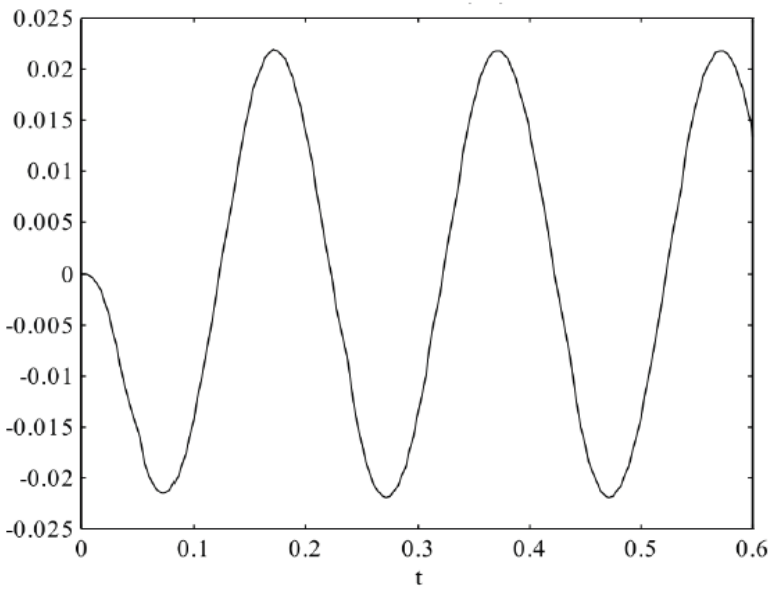
Model-based control

- Performance

$$u(t) = \hat{m}(\ddot{x}^{des}(t) + K_d\dot{e}(t) + K_p e(t)) + \hat{b}\dot{x}(t) + \hat{k}x(t)$$



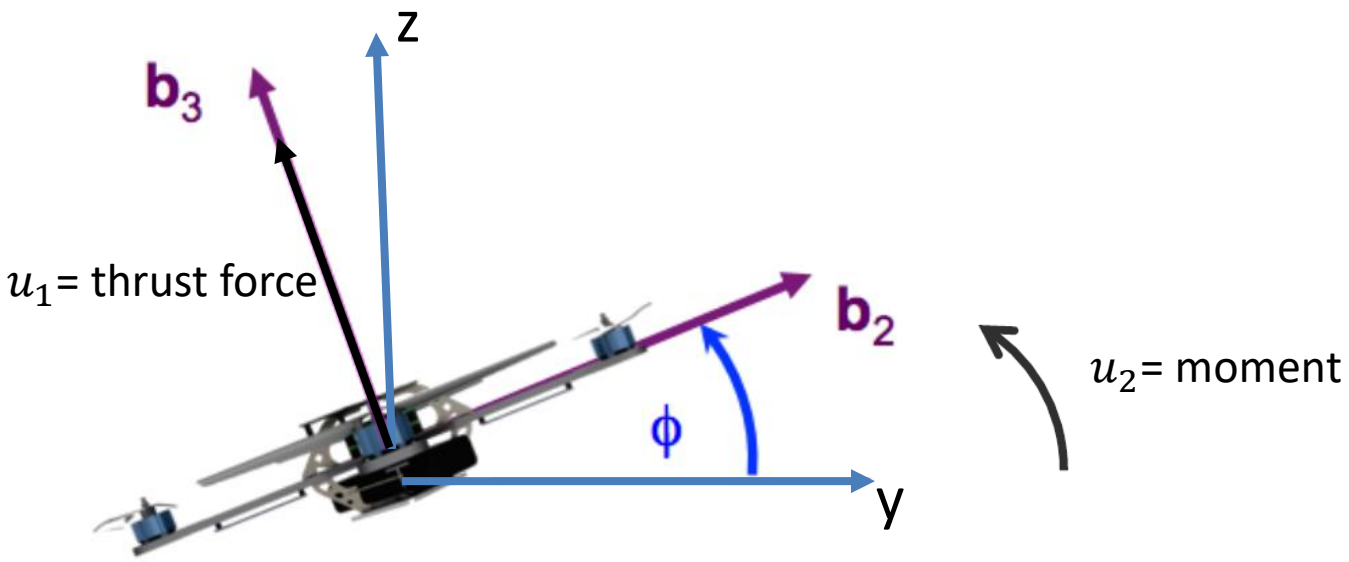
Perfect model



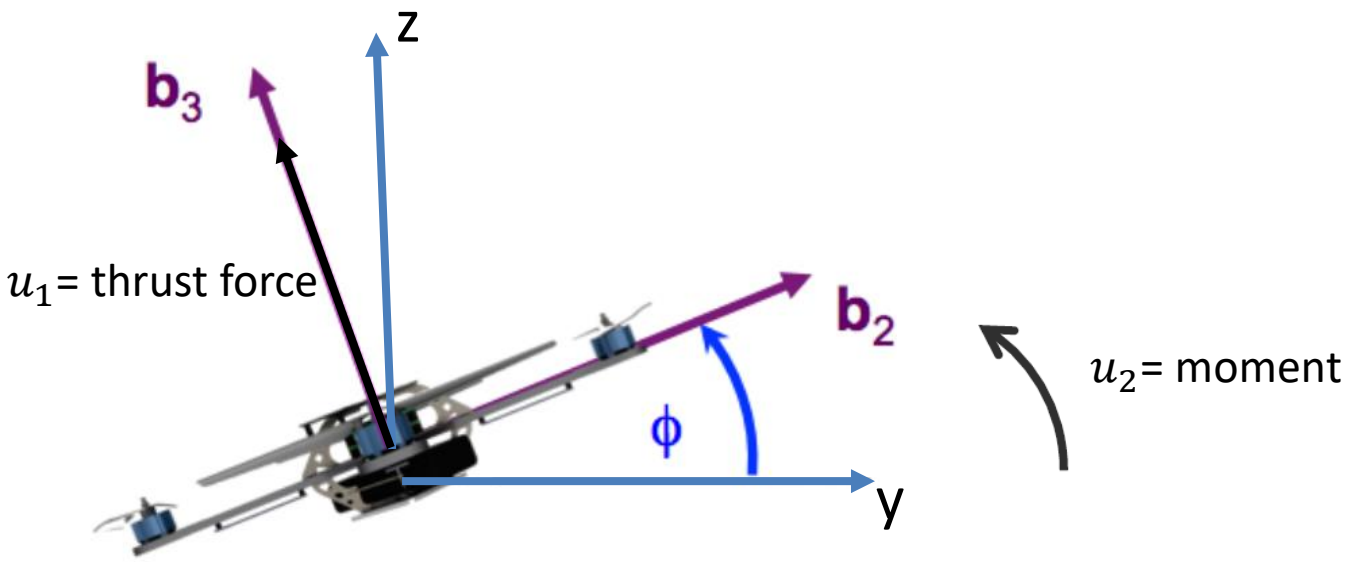
Imperfect model, 10% errors in parameters

Quadrotor Control

Application to Quadrotors



Planar Quadrotor Model



$$\sum F_y = -u_1 \sin(\phi) = m\ddot{y}$$

$$\sum F_z = -mg + u_1 \cos(\phi) = m\ddot{z}$$

$$M = u_2 = I_{xx}\ddot{\phi}$$

$$\Rightarrow \begin{bmatrix} \ddot{y} \\ \ddot{z} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \\ 0 \end{bmatrix} + \begin{bmatrix} -\frac{1}{m}\sin\phi & 0 \\ \frac{1}{m}\cos\phi & 0 \\ 0 & \frac{1}{I_{xx}} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

Linearized Dynamic Model

- Nonlinear dynamics

$$\ddot{y} = -\frac{u_1}{m} sin(\phi)$$
$$\ddot{z} = -g + \frac{u_1}{m} cos(\phi)$$
$$\ddot{\phi} = \frac{u_2}{I_{xx}}$$

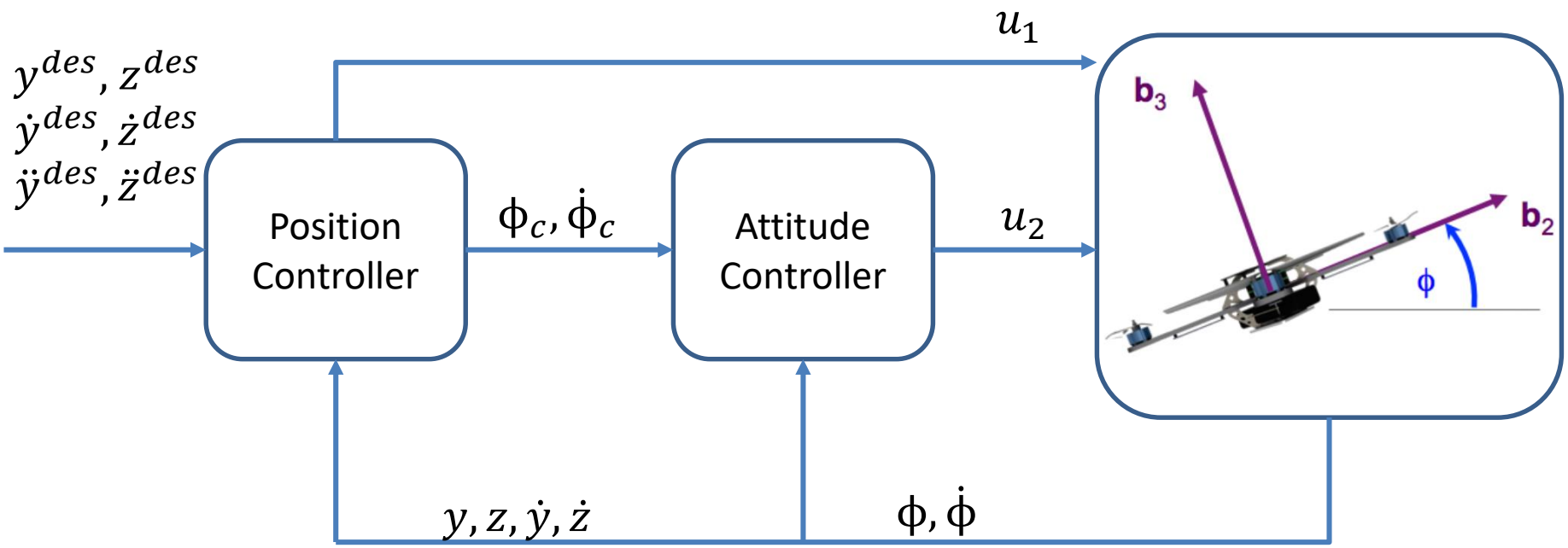
- Equilibrium hover configuration

$$y_0, z_0, \phi_0 = 0, u_{1,0} = mg, u_{2,0} = 0$$

- Linearized dynamics

$$\ddot{y} = -g\phi$$ Cascaded second order system
$$\ddot{z} = -g + \frac{u_1}{m}$$
$$\ddot{\phi} = \frac{u_2}{I_{xx}}$$ A simple second order system

Control Structure



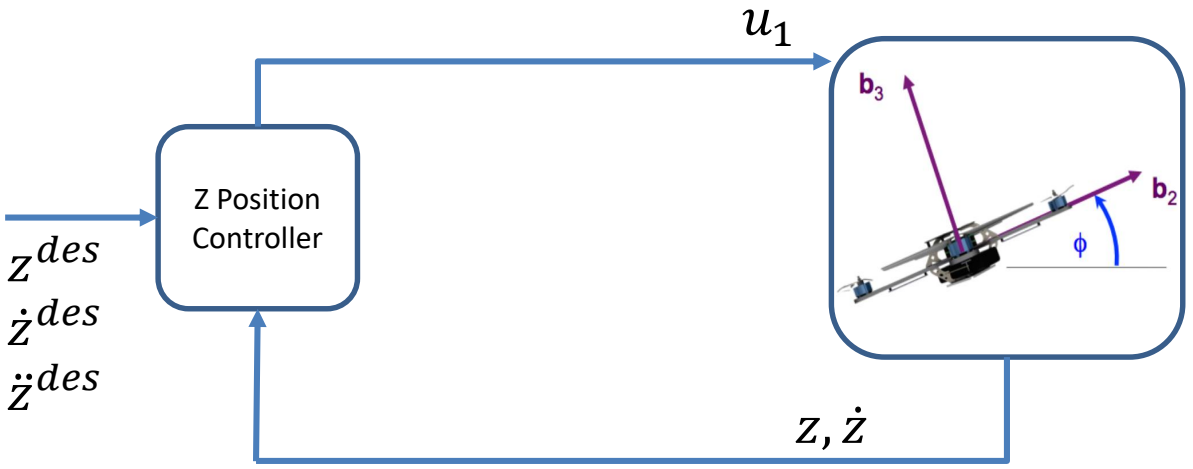
Control Equations

- Z-position control

PD: $\ddot{z}_c = \ddot{z}^{des} + K_{d,z}(\dot{z}^{des} - \dot{z}) + K_{p,z}(z^{des} - z)$

Model: $\ddot{z} = -g + \frac{u_1}{m}$

$$u_1 = m(g + \ddot{z}^{des} + K_{d,z}(\dot{z}^{des} - \dot{z}) + K_{p,z}(z^{des} - z))$$



Linearized Dynamic Model

- Y-position control

PD: $\ddot{y}_c = \ddot{y}^{des} + K_{d,y}(\dot{y}^{des} - \dot{y}) + K_{p,y}(y^{des} - y)$

Model: $\ddot{y} = -g\phi$

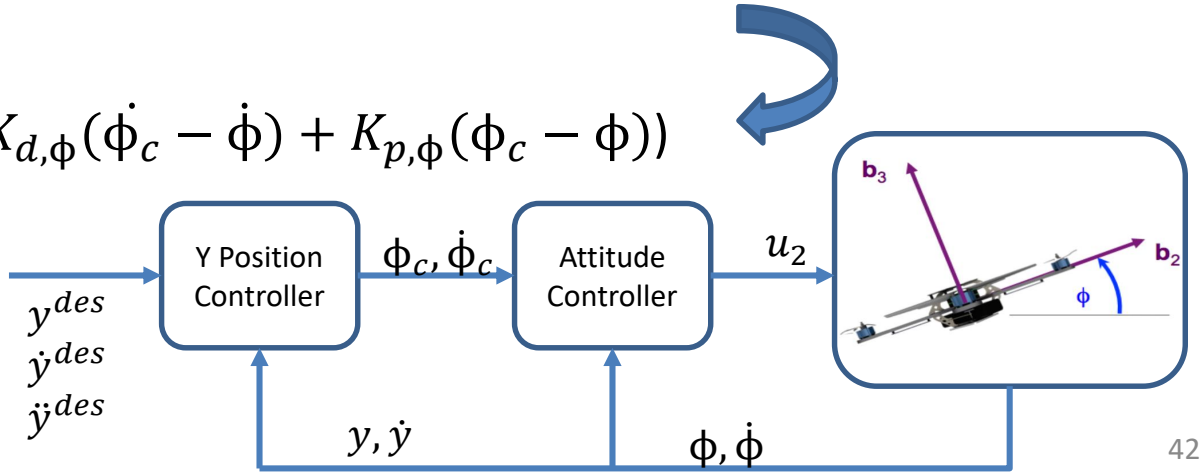
$$\phi_c = -\frac{1}{g}(\ddot{y}^{des} + K_{d,y}(\dot{y}^{des} - \dot{y}) + K_{p,y}(y^{des} - y))$$

- Attitude control

PD: $\ddot{\phi}_c = K_{d,\phi}(\dot{\phi}_c - \dot{\phi}) + K_{p,\phi}(\phi_c - \phi)$

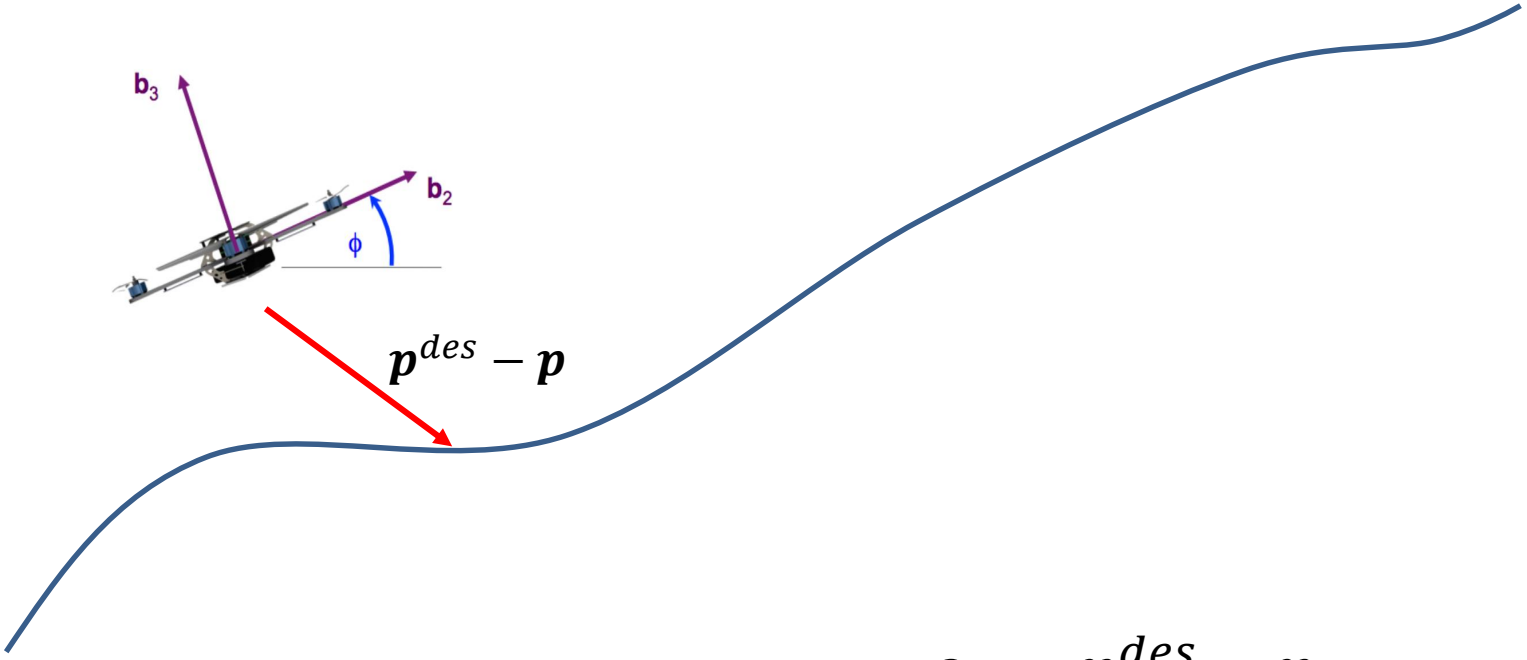
Model: $\ddot{\phi} = \frac{u_2}{I_{xx}}$

$$u_2 = I_{xx}(K_{d,\phi}(\dot{\phi}_c - \dot{\phi}) + K_{p,\phi}(\phi_c - \phi))$$



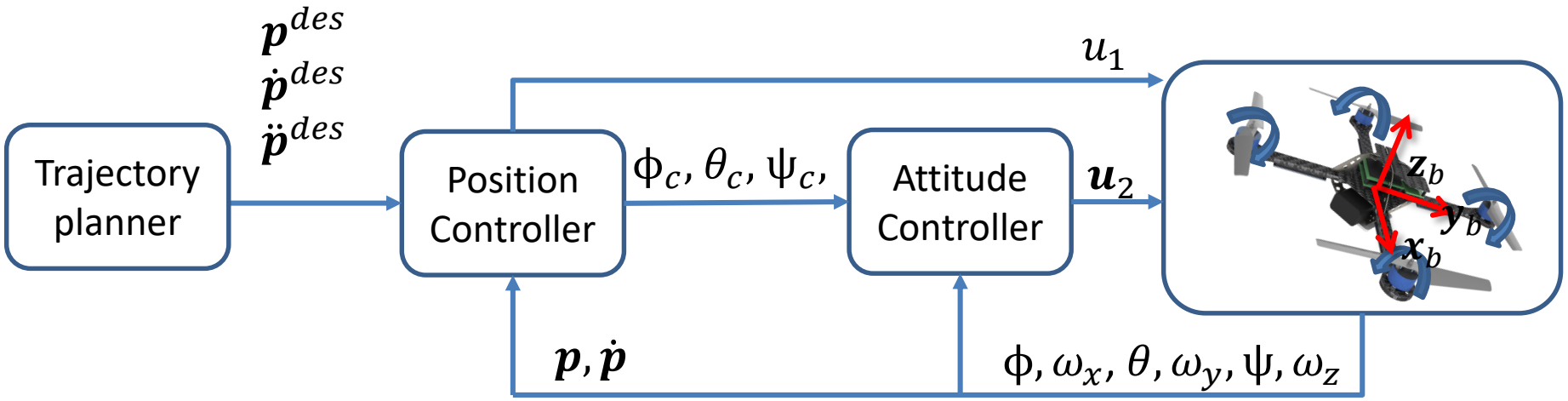
Trajectory Tracking

Given $\boldsymbol{p}^{des}$,$\dot{\boldsymbol{p}}^{des}$,$\ddot{\boldsymbol{p}}^{des}$



$$\boldsymbol{e_p} = \boldsymbol{p}^{des} - \boldsymbol{p}$$
$$\boldsymbol{e_v} = \dot{\boldsymbol{p}}^{des} - \dot{\boldsymbol{p}}$$
$$\ddot{\boldsymbol{p}}_C = \ddot{\boldsymbol{p}}^{des} + \boldsymbol{K_d e_v} + \boldsymbol{K_p e_p}$$

3-D Quadrotor



- Nonlinear dynamics

Newton Equation: $$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$ u_1

Euler Equation: $$\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$$ $\boldsymbol{u}_2$

3-D Quadrotor

- Linearization

Equilibrium hover $(\phi_0 \sim 0, \theta_0 \sim 0, u_{1,0} \sim mg)$

Newton equation

$$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$

$$\boldsymbol{R} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix}$$

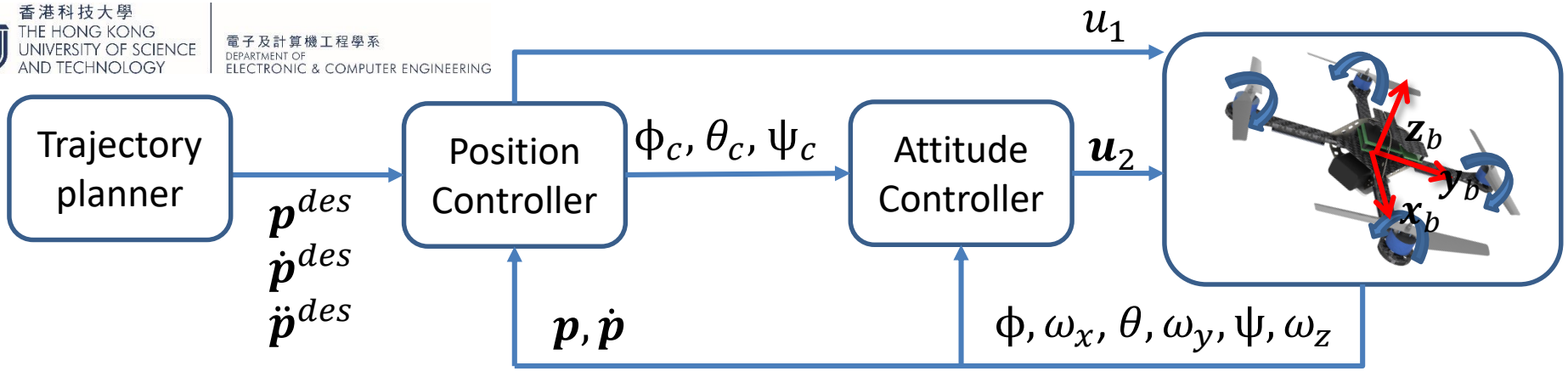
$$\ddot{\boldsymbol{p}}_1 = \ddot{x} = g(\theta cos\psi + \phi sin\psi)$$
$$\ddot{\boldsymbol{p}}_2 = \ddot{y} = g(\theta sin\psi - \phi cos\psi)$$
$$\ddot{\boldsymbol{p}}_3 = \ddot{z} = -g + \frac{u_1}{m}$$

Euler angle derivative

$$\begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix}$$

Euler Equation:

$$\boldsymbol{I} \cdot \begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$$



- Position control

PID: $\ddot{\boldsymbol{p}}_{i,c} = \ddot{\boldsymbol{p}}_i^{des} + K_{d,i}\left(\dot{\boldsymbol{p}}_i^{des} - \dot{\boldsymbol{p}}_i\right) + K_{p,i}\left(\boldsymbol{p}_i^{des} - \boldsymbol{p}_i\right)$

Model: $u_1 = m(g + \ddot{\boldsymbol{p}}_{3,c})$ (Newton Equation)

$$\phi_c = \frac{1}{g}(\ddot{\boldsymbol{p}}_{1,c} sin\psi - \ddot{\boldsymbol{p}}_{2,c} cos\psi) \quad \theta_c = \frac{1}{g}(\ddot{\boldsymbol{p}}_{1,c} cos\psi + \ddot{\boldsymbol{p}}_{2,c} sin\psi)$$

These are current yaw, not the commanded yaw

- Attitude control

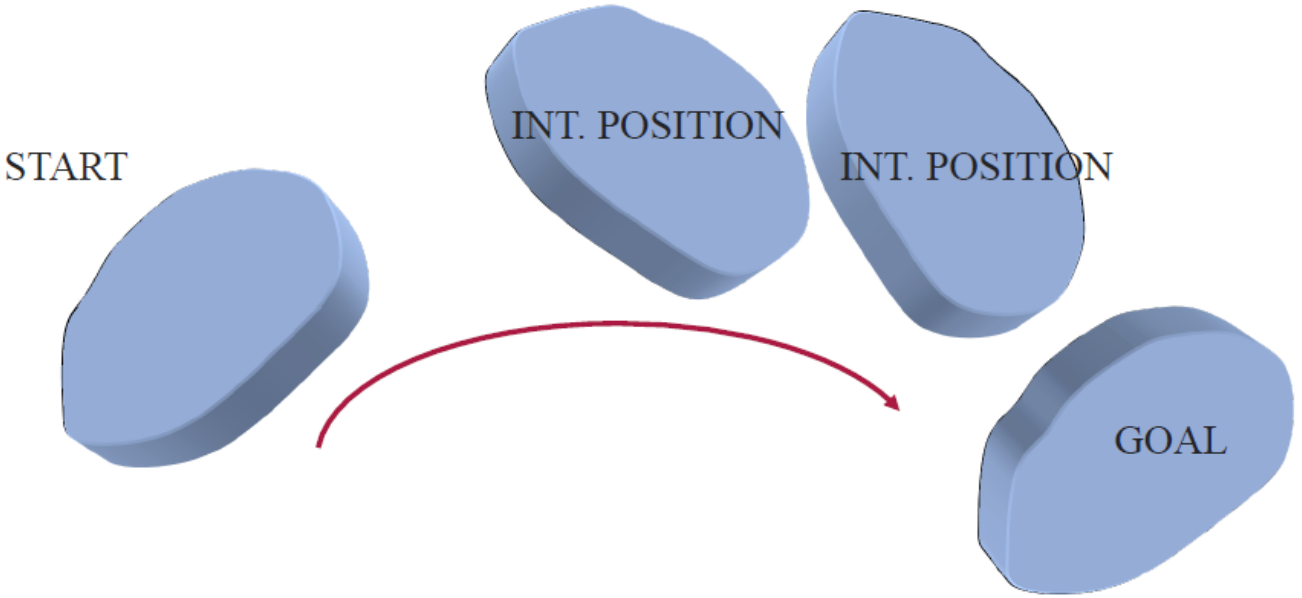
PID: $$\begin{bmatrix} \ddot{\phi}_c \\ \ddot{\theta}_c \\ \ddot{\psi}_c \end{bmatrix} = \begin{bmatrix} K_{p,\phi}(\phi_c - \phi) + K_{d,\phi}(\dot{\phi}_c - \dot{\phi}) \\ K_{p,\theta}(\theta_c - \theta) + K_{d,\phi}(\dot{\theta}_c - \dot{\theta}) \\ K_{p,\psi}(\psi_c - \psi) + K_{d,\psi}(\dot{\psi}_c - \dot{\psi}) \end{bmatrix}$$

Model: $$\boldsymbol{u}_2 = \boldsymbol{I} \cdot \begin{bmatrix} \ddot{\phi}_c \\ \ddot{\theta}_c \\ \ddot{\psi}_c \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}$$ (Euler Equation)

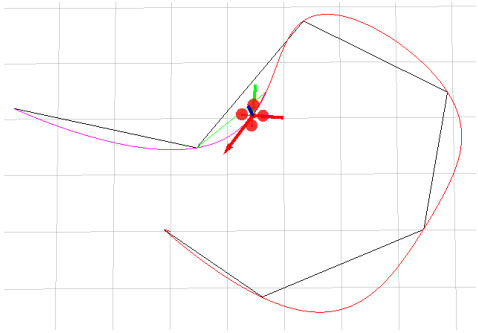
Trajectory Generation

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)
    - To be covered in the next lecture
  - Smoothness criterion
    - Generally translates into minimizing rate of change of “input”
- Question: How to make sure that a trajectory can be tracked by the quadrotor?

Differential Flatness

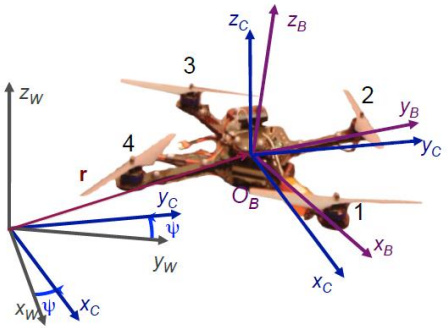
- The states and the inputs of a quadrotor can be written as algebraic functions of four carefully selected flat outputs and their derivatives
  - Enables automated generation of trajectories
  - Any smooth trajectory in the space of flat outputs (with reasonably bounded derivatives) can be followed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

Differential Flatness

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

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

Body angular velocity viewed in the body frame



  - Equation of motions:

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

$$\boldsymbol{\omega}_B = \begin{bmatrix}\omega_x \\ \omega_y \\ \omega_z\end{bmatrix}, \quad \dot{\boldsymbol{\omega}}_B = \boldsymbol{I}^{-1}\left[-\boldsymbol{\omega}_B \times \boldsymbol{I} \cdot \boldsymbol{\omega}_B + \begin{bmatrix}u_2 \\ u_3 \\ u_4\end{bmatrix}\right]$$

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

Differential Flatness

- Orientation
  - Quadrotor state:

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

  - From the equation of motion:

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

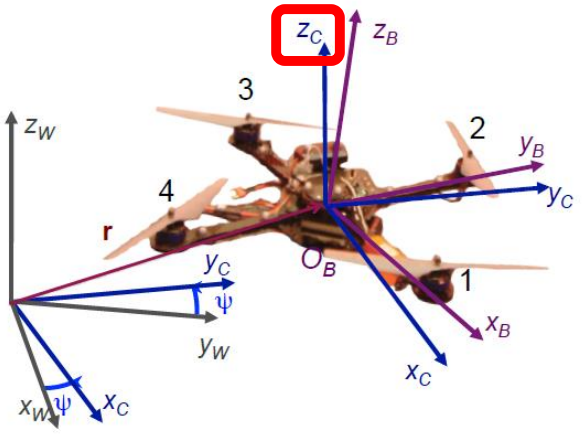
  - Define the yaw vector (Z-X-Y Euler):

$$\mathbf{x}_C = [cos\boldsymbol{\sigma}_4, sin\boldsymbol{\sigma}_4, 0]^T$$

  - Orientation can be expressed in terms of flat outputs

$$\mathbf{y}_B = \frac{\mathbf{z}_B \times \mathbf{x}_C}{\|\mathbf{z}_B \times \mathbf{x}_C\|}, \quad \mathbf{x}_B = \mathbf{y}_B \times \mathbf{z}_B \quad \boldsymbol{R}_B = [\mathbf{x}_B \ \mathbf{y}_B \ \mathbf{z}_B]$$

A “no tilt” frame



Differential Flatness

- Angular velocity
  - Quadrotor state:

    $\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_z, \omega_y, \omega_z]^T$

  - Take the derivative of the equation of motion

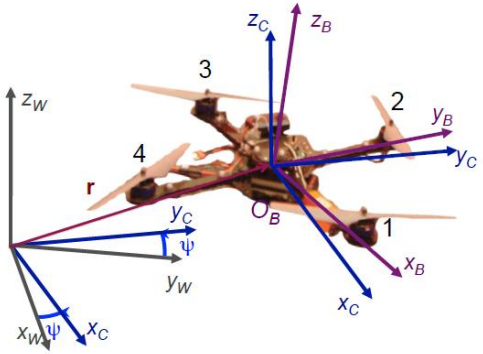
    $m\ddot{\boldsymbol{p}} = -mg\mathbf{z}_W + u_1\mathbf{z}_B.$ ⟶ $m\dot{\boldsymbol{a}} = \dot{u}_1\mathbf{z}_B + \boldsymbol{\omega}_{BW} \times u_1\,\mathbf{z}_B$

  - Quadrotors only have vertical thrust:

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

  - We have:

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



Body angular velocity viewed in the world frame

Differential Flatness

- Angular velocity
  - Quadrotor state:

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

  - We have:

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

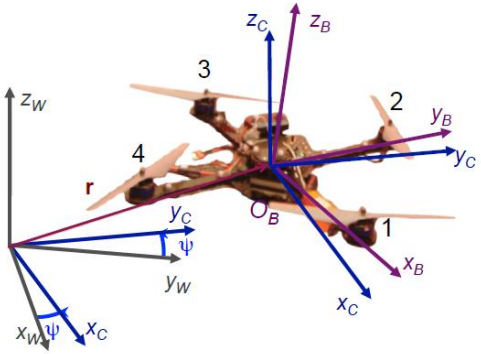
    - This is the projection of $\frac{m}{u_1}\dot{\boldsymbol{a}}$ onto the $x_B - y_B$ plane

  - We know that:

$$\boldsymbol{\omega}_{BW} = \omega_x \mathbf{x}_B + \omega_y \mathbf{y}_B + \omega_z \mathbf{z}_B$$

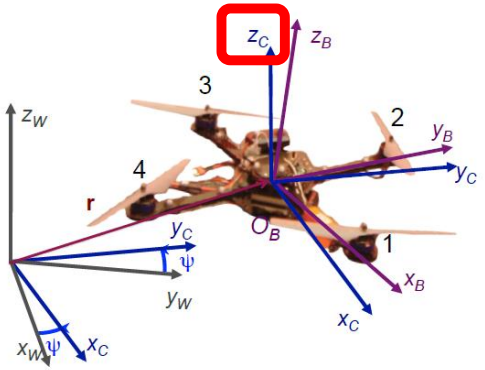
  - Angular velocities along x_B and y_B directions can be found as:

$$\omega_x = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \quad \omega_y = \mathbf{h}_\omega \cdot \mathbf{x}_B$$



Differential Flatness

A "no tilt" frame



- Angular velocity
  - Quadrotor state:

$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

  - We have:

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

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

$$\omega_z = \boldsymbol{\omega}_{BW} \cdot \mathbf{z}_B = \boldsymbol{\omega}_{CW} \cdot \mathbf{z}_B = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B.$$

Differential Flatness

- Summary
  - Quadrotor state:

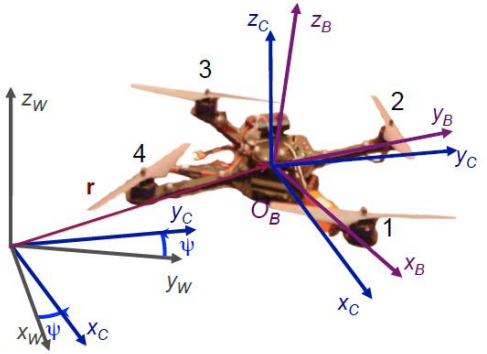
$$\mathbf{X} = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \omega_x, \omega_y, \omega_z]^T$$

  - Flat outputs:
    - $\boldsymbol{\sigma} = [x, y, z, \psi]^T$
  - Position, velocity, acceleration
    - Derivatives of flat outputs
  - Orientation

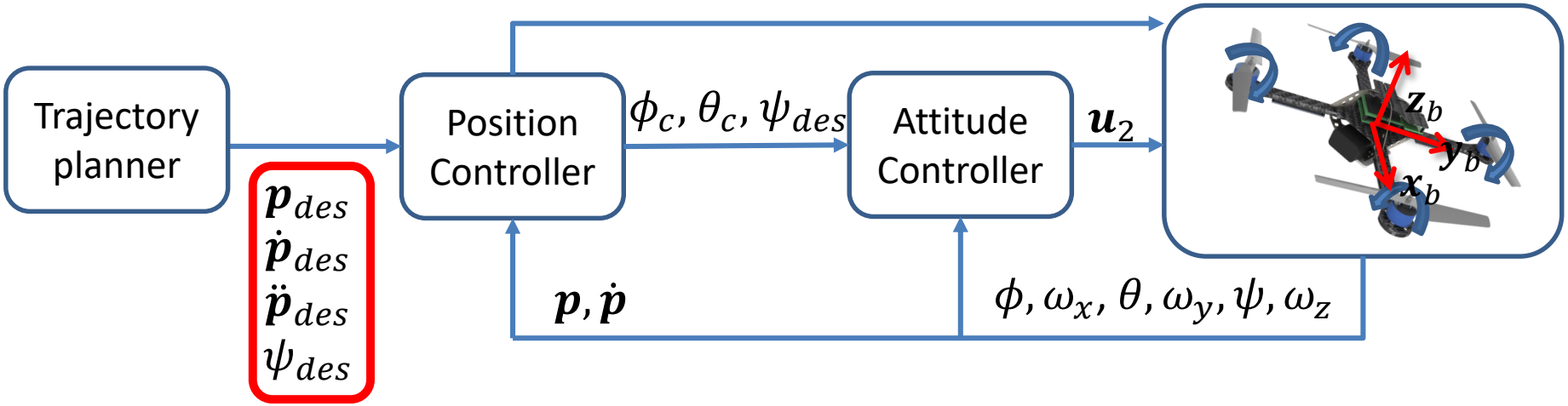
$$\mathbf{x}_C = [cos\sigma_4, sin\sigma_4, 0]^T \longrightarrow \boldsymbol{R}_B = [\mathbf{x}_B \ \ \mathbf{y}_B \ \ \mathbf{z}_B]$$

  - Angular velocity

$$\omega_x = -\mathbf{h}_\omega \cdot \mathbf{y}_B, \quad \omega_y = \mathbf{h}_\omega \cdot \mathbf{x}_B, \quad \omega_z = \dot{\psi}\mathbf{z}_W \cdot \mathbf{z}_B$$



How about Force and Moment Input (u_1, u_2)?



Nonlinear dynamics

Newton Equation: $$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + \boldsymbol{R} \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix}$$

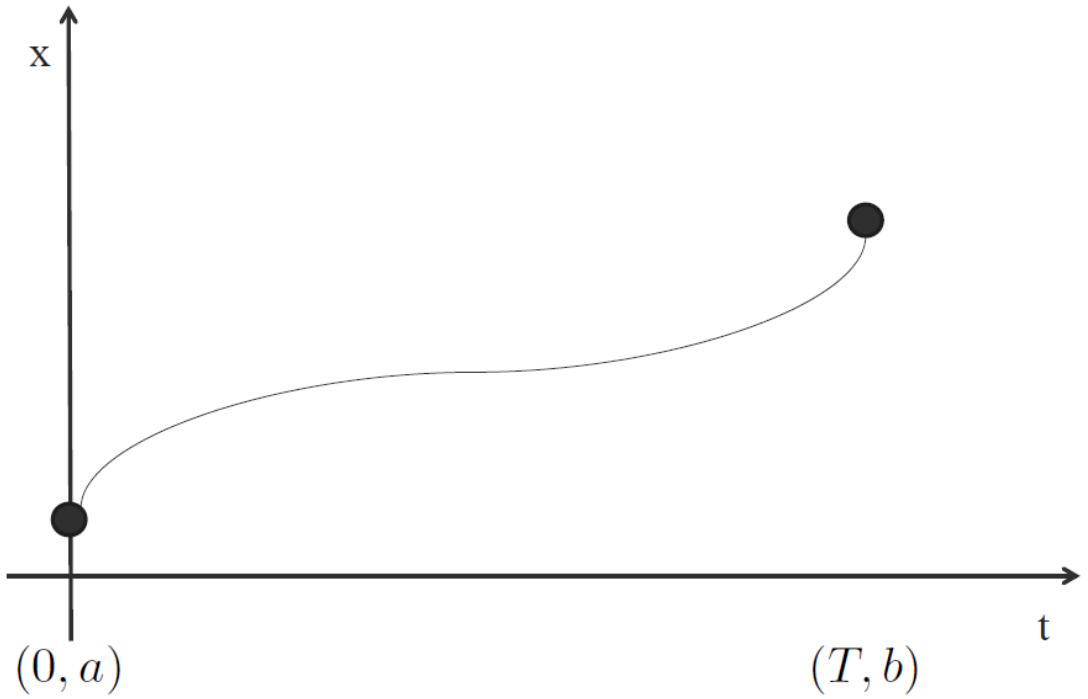
Euler Equation: $$\boldsymbol{I} \cdot \begin{bmatrix} \dot{\omega}_x \\ \dot{\omega}_y \\ \dot{\omega}_z \end{bmatrix} + \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \times \boldsymbol{I} \cdot \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \begin{bmatrix} l(F_2 - F_4) \\ l(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix}$$

Polynomial Trajectories

- Flat outputs:
  - $\boldsymbol{\sigma} = [x, y, z, \psi]^T$
- Trajectory in the space of flat outputs:
  - $\boldsymbol{\sigma}(t) = [T_0, T_M] \rightarrow \mathbb{R}^3 \times SO(2)$
- 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

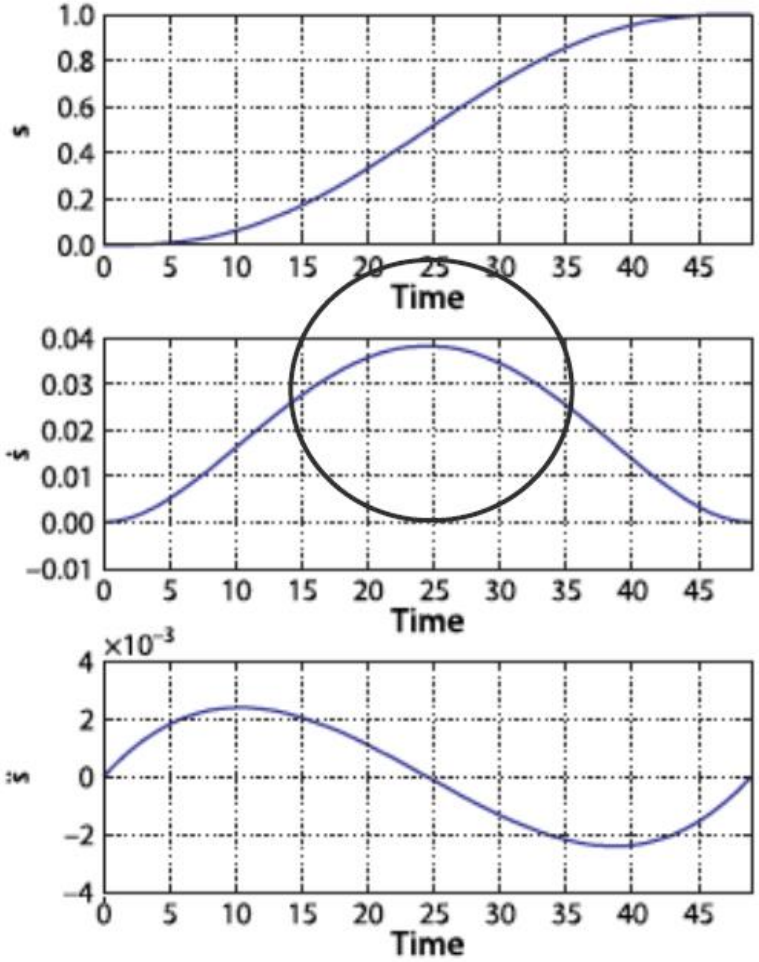
- 5^{th} order polynomial trajectory:
  - $x(t) = c_5t^5 + c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0$
- Boundary conditions

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

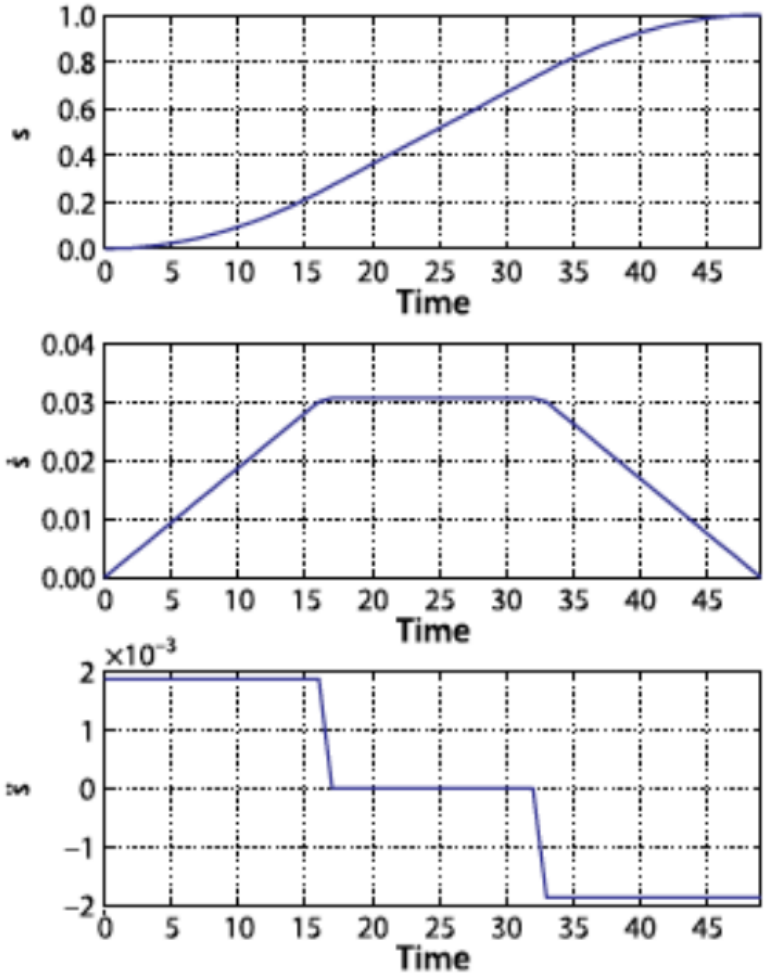
- Solve:

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

Smooth 1D Trajectory

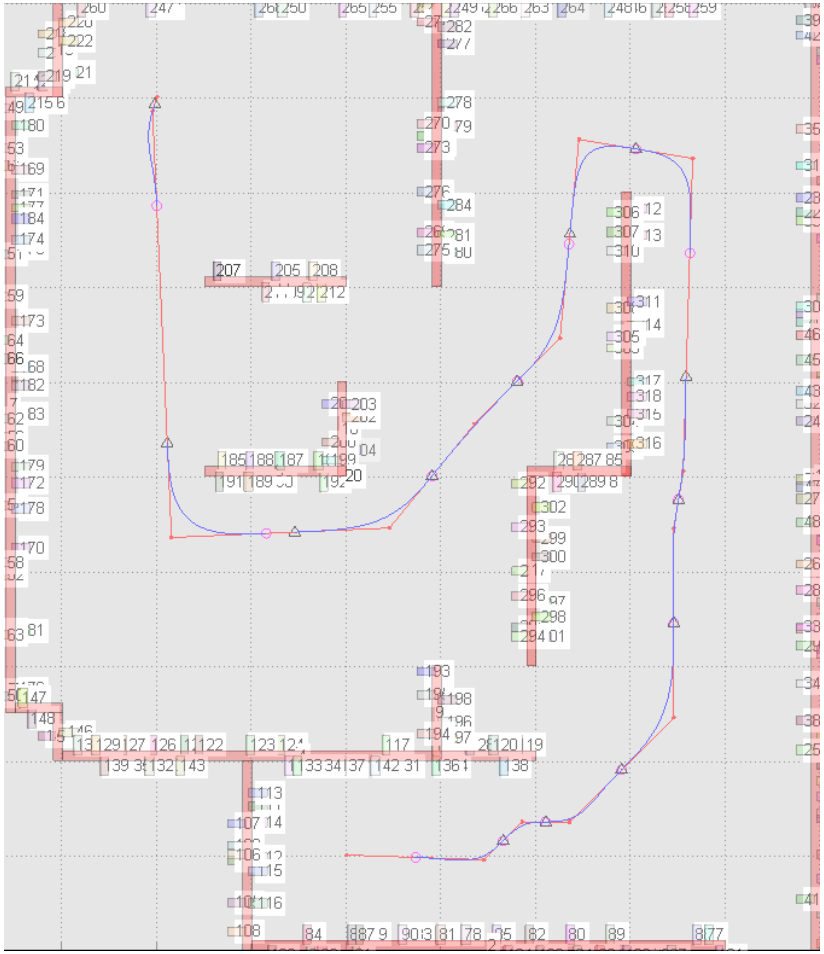


Bang-Bang Trajectory



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 independently:
  - $x(t) = c_5t^5 + c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0$
- Boundary conditions

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

- Solve:

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

Next Lecture…

- Continuation on trajectory generation
- Path planning

Logistics

- Project 1, phase 1 is released (02/21)
  - Due on 3/3. Early submission is encouraged.

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
Associate Professor
Dept. of ECE, HKUST

28 February 2023

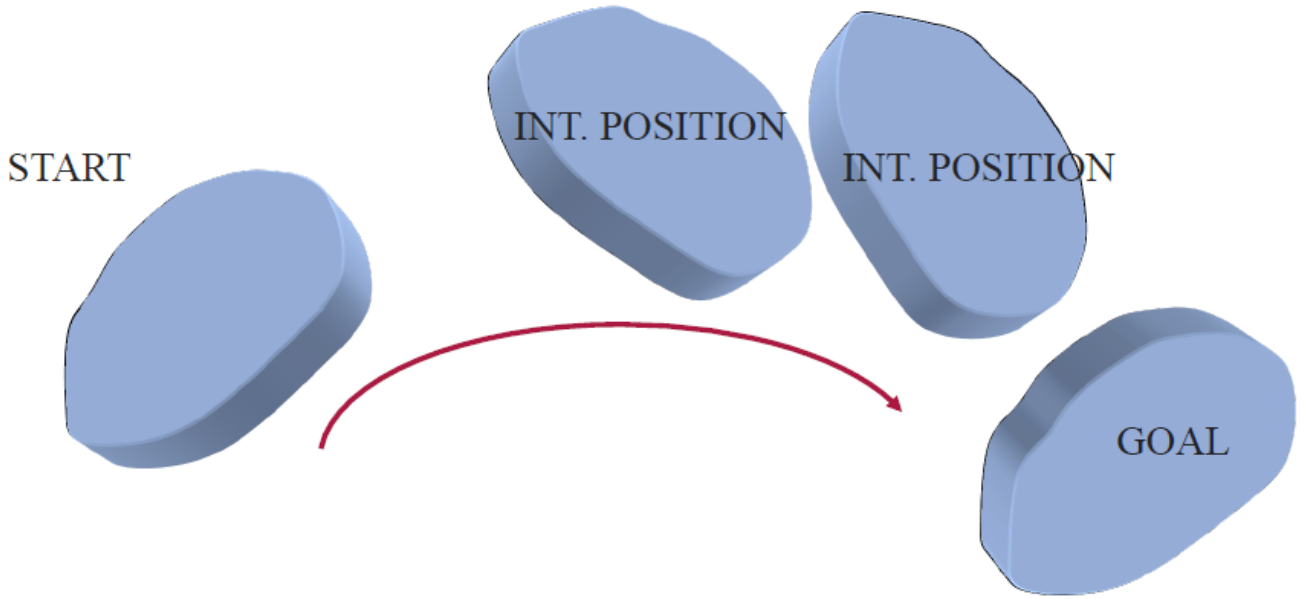
Outline

- Continuation on Trajectory Generation
- Path Planning

Trajectory Generation

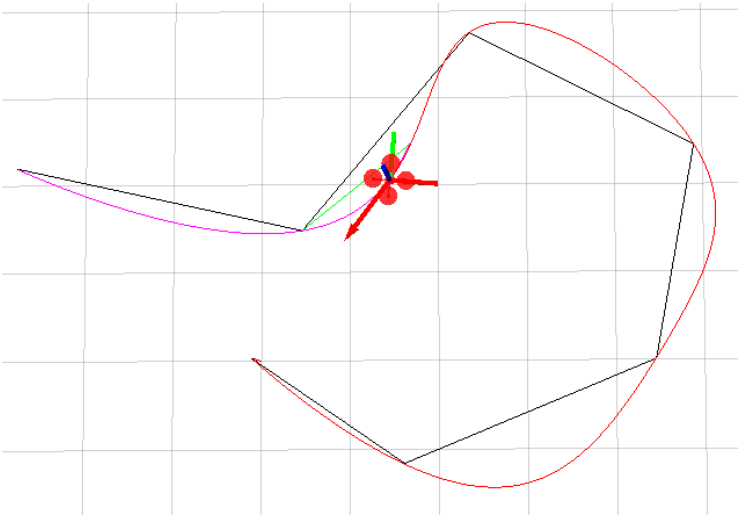
Smooth 3D Trajectories

- 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

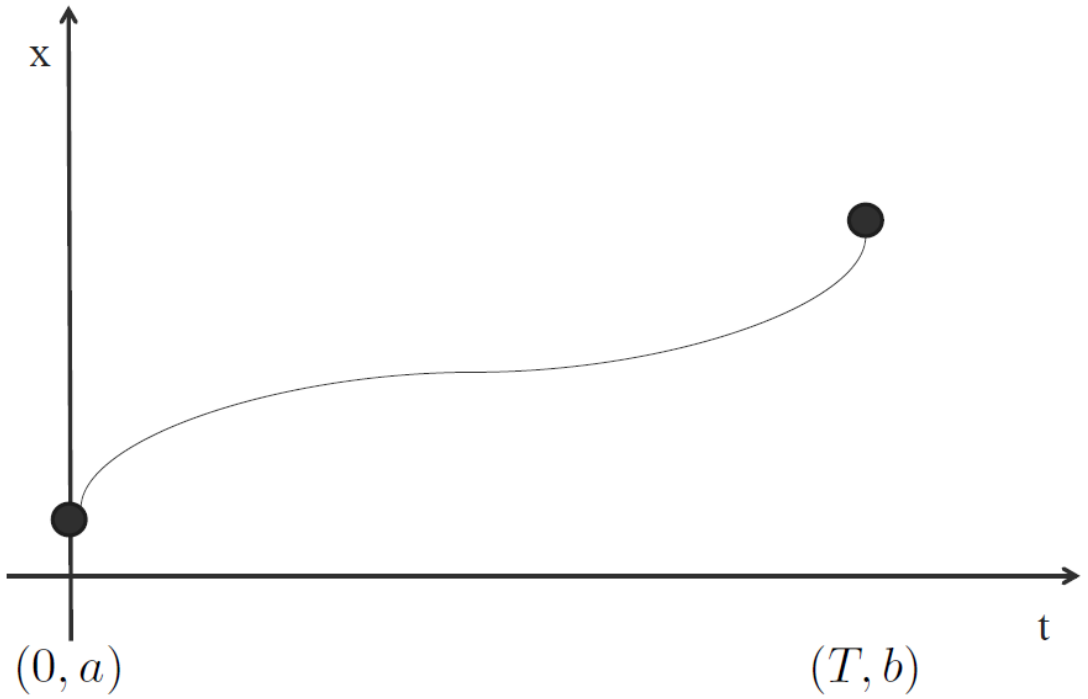
- The states and the inputs of a quadrotor can be written as algebraic functions of four carefully selected flat outputs and their derivatives
  - Enables automated generation of trajectories
  - Any smooth trajectory in the space of flat outputs (with reasonably bounded derivatives) can be followed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

Polynomial Trajectories

- Flat outputs:
  - $\boldsymbol{\sigma} = [x, y, z, \psi]^T$
- Trajectory in the space of flat outputs:
  - $\boldsymbol{\sigma}(t) = [T_0, T_M] \rightarrow \mathbb{R}^3 \times SO(2)$
- 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

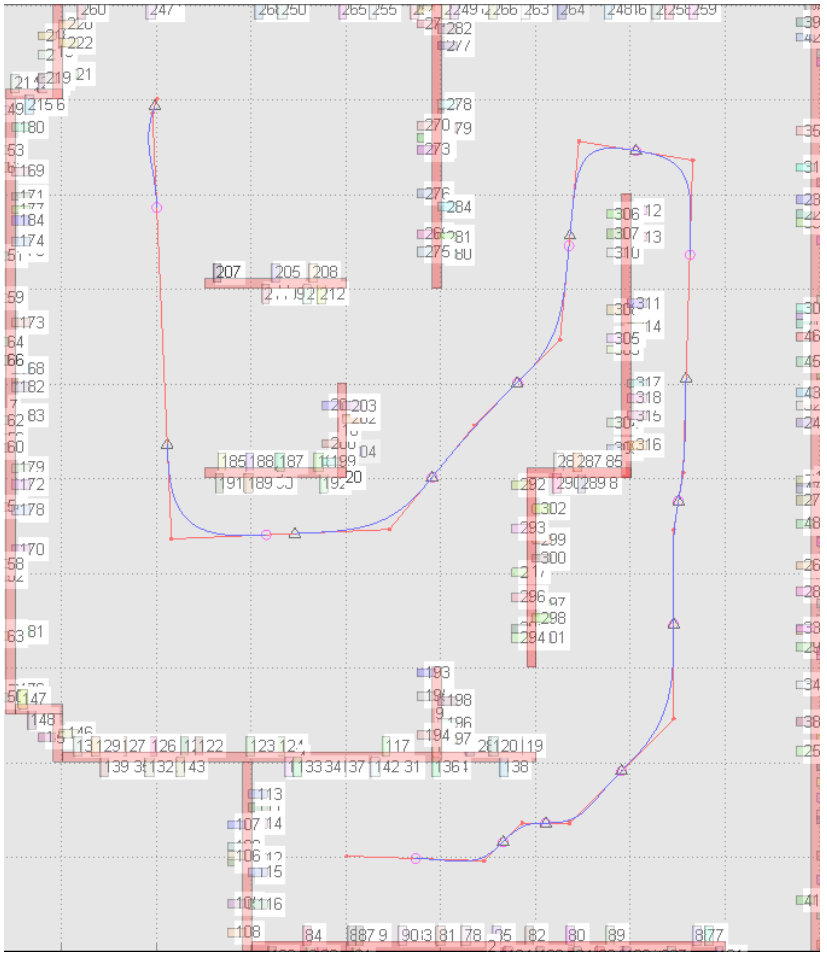
	Position	Velocity	Acceleration
t = 0	a	0	0
t = T	b	0	0

- 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 Multi-Segment Trajectory

- Given waypoints to a desired goal
- 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 independently:
  - $x(t) = c_5t^5 + c_4t^4 + c_3t^3 + c_2t^2 + c_1t + c_0$
- 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

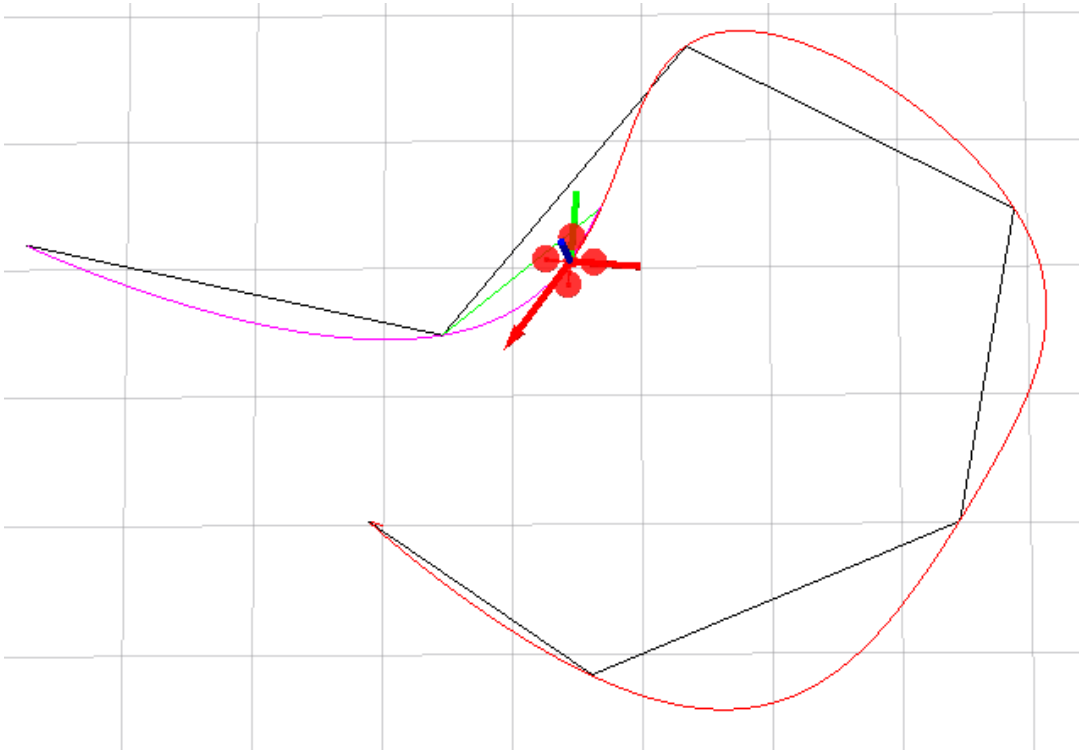
Optimization-based Trajectory Generation

- 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

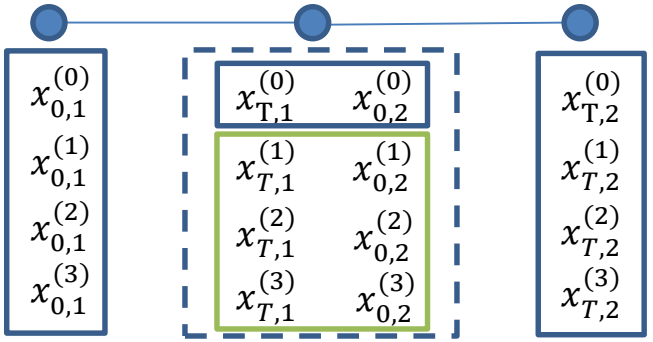
- Formulation – segment durations must be known!

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

Subject to:

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



- Minimum degree polynomial to ensure smoothness for one-segment trajectory:
  - Minimum jerk: N = 2 x 3 - 1 = 5
  - Minimum snap: N = 2 x 4 - 1 = 7

Minimum Snap Trajectory Generation

- Cost function for one polynomial segment:

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

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

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

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

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

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

Minimum Snap Trajectory Generation

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

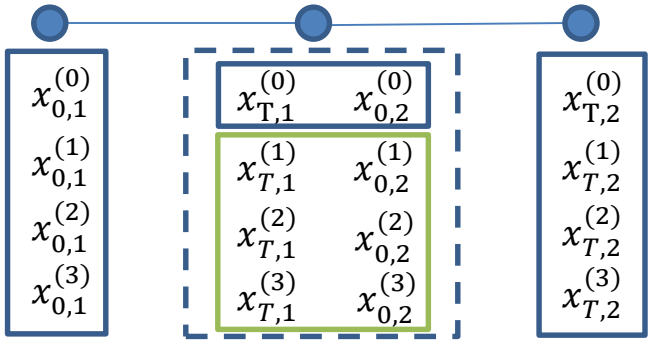
$$f_j^{(k)}(T_j) = x_j^{(k)}$$

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

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

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

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



Minimum Snap Trajectory Generation

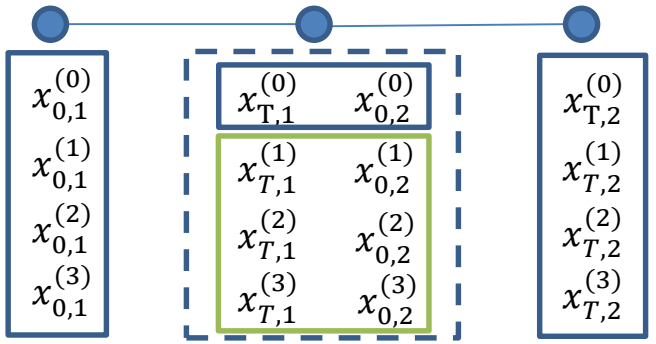
- Continuity constraint between two segments:
  - Ensures continuity between trajectory segments when no specific derivatives are given

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

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

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

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



Minimum Snap Trajectory Generation

- Constrained quadratic programming (QP) formulation:

$$\min \quad \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix}^T \begin{bmatrix} \mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_M \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix}$$

$$\text{s.t.} \quad \mathbf{A}_{\text{eq}} \begin{bmatrix} \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_M \end{bmatrix} = \mathbf{d}_{eq}$$

Minimum Snap Trajectory Generation

- Direct optimization of polynomial trajectories is numerically unstable
- A change of variable that instead optimizes segment endpoint derivatives is preferred
- We have $\boldsymbol{M}_{\mathrm{j}}\mathbf{p}_j = \mathbf{d}_j$, where $\boldsymbol{M}_{\mathrm{j}}$ is a mapping matrix that maps polynomial coefficients to derivatives

$$J = \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}^T \begin{bmatrix} \boldsymbol{M}_1 & & \\ & \ddots & \\ & & \boldsymbol{M}_M \end{bmatrix}^{-T} \begin{bmatrix} \mathbf{Q}_1 & & \\ & \ddots & \\ & & \mathbf{Q}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{M}_1 & & \\ & \ddots & \\ & & \boldsymbol{M}_M \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_M \end{bmatrix}$$

Minimum Snap Trajectory Generation

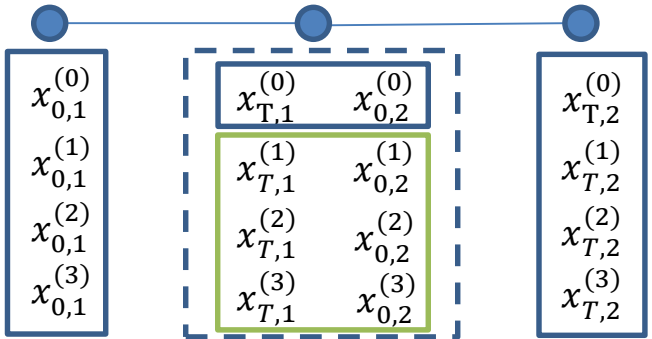
- Use a selection matrix $\mathbf{C}$ to separate fixed / constrained ($\mathbf{d}_C$) and free / unknown ($\mathbf{d}_U$) variables
  - Free variables : derivatives unspecified, only enforced by continuity constraints

$$J = \begin{bmatrix}\mathbf{d}_C \\ \mathbf{d}_U\end{bmatrix}^T \underbrace{\mathbf{C}\boldsymbol{M}^{-T}\mathbf{Q}\boldsymbol{M}^{-1}\mathbf{C}^T}_{\mathbf{R}} \begin{bmatrix}\mathbf{d}_C \\ \mathbf{d}_U\end{bmatrix} = \begin{bmatrix}\mathbf{d}_C \\ \mathbf{d}_U\end{bmatrix}^T \begin{bmatrix}\mathbf{R}_{CC} & \mathbf{R}_{CU} \\ \mathbf{R}_{UC} & \mathbf{R}_{UU}\end{bmatrix} \begin{bmatrix}\mathbf{d}_C \\ \mathbf{d}_U\end{bmatrix}$$

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

$$J = \mathbf{d}_C^T\mathbf{R}_{CC}\mathbf{d}_C + \mathbf{d}_C^T\mathbf{R}_{CU}\mathbf{d}_U + \mathbf{d}_U^T\mathbf{R}_{UC}\mathbf{d}_C + \mathbf{d}_U^T\mathbf{R}_{UU}\mathbf{d}_U$$

$$\mathbf{d}_U^* = -\mathbf{R}_{UU}^{-1}\mathbf{R}_{CU}^T\mathbf{d}_C$$

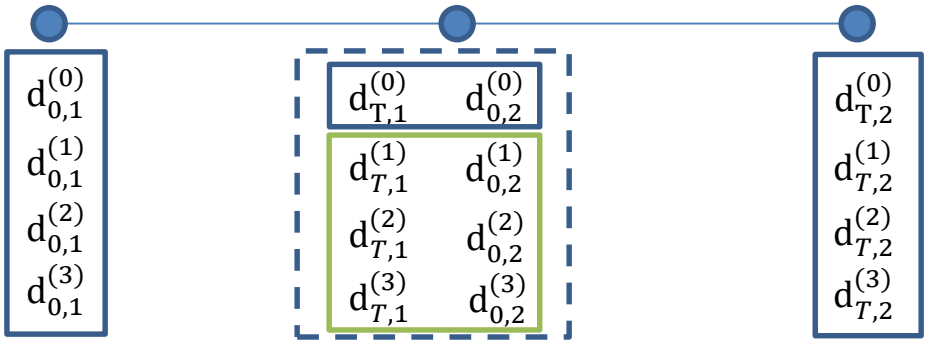


Minimum Snap Trajectory Generation

$\mathrm{d}_{0,j}^{(k)}$ ← Index of derivative
Index of segment
Time index

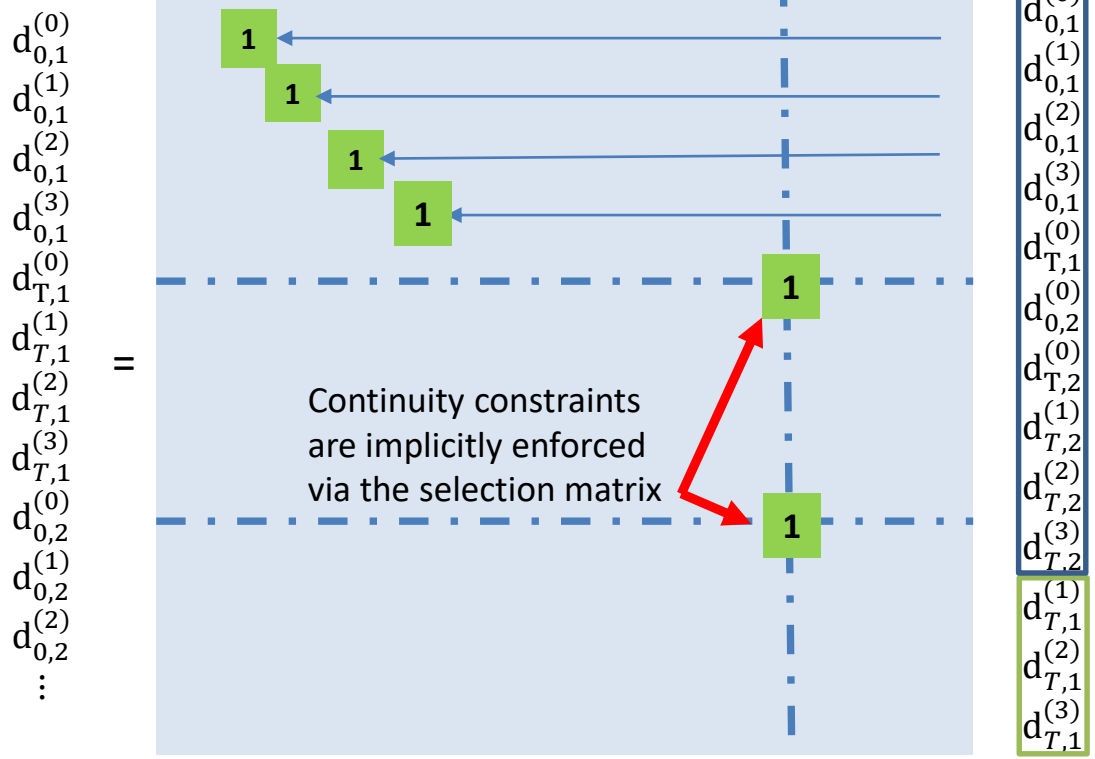
Fixed / constrained derivatives: fixed start, goal state, and intermediate positions

Free / unknown derivatives: all derivatives at intermediate connections.



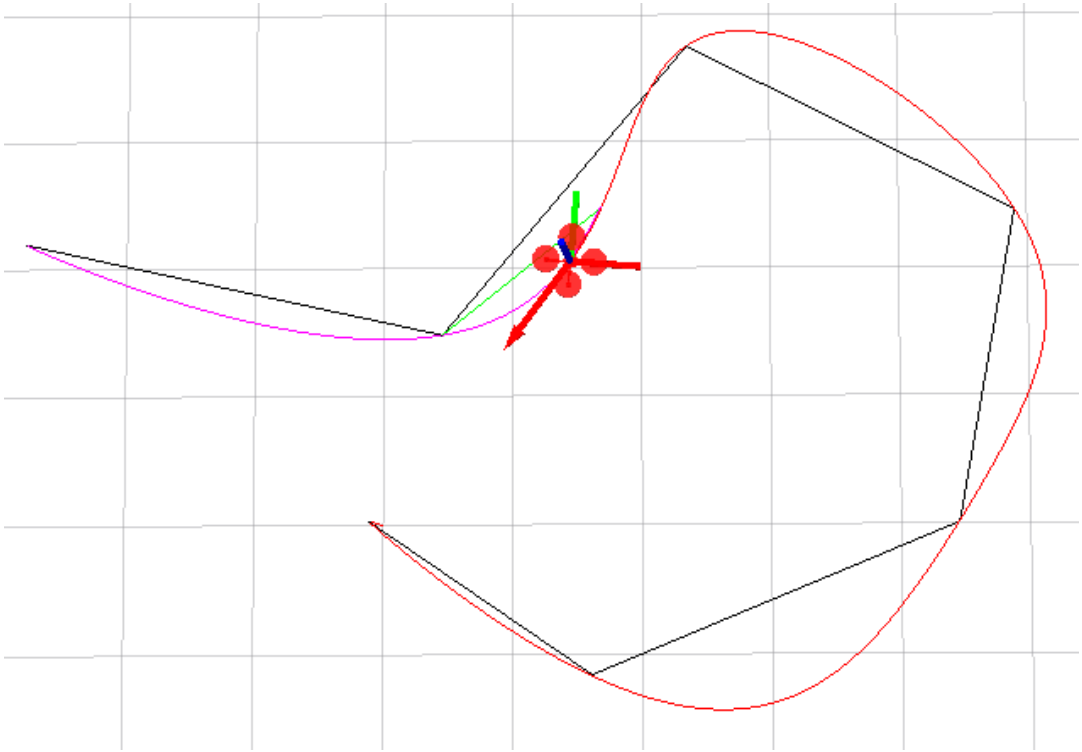
$$\begin{bmatrix}\mathbf{d}_1\\ \vdots \\ \mathbf{d}_M\end{bmatrix} = \mathbf{C}^T \begin{bmatrix}\mathbf{d}_C\\ \mathbf{d}_U\end{bmatrix}$$

16x1 = $\mathbf{C}^T$:16x13 × 13x1



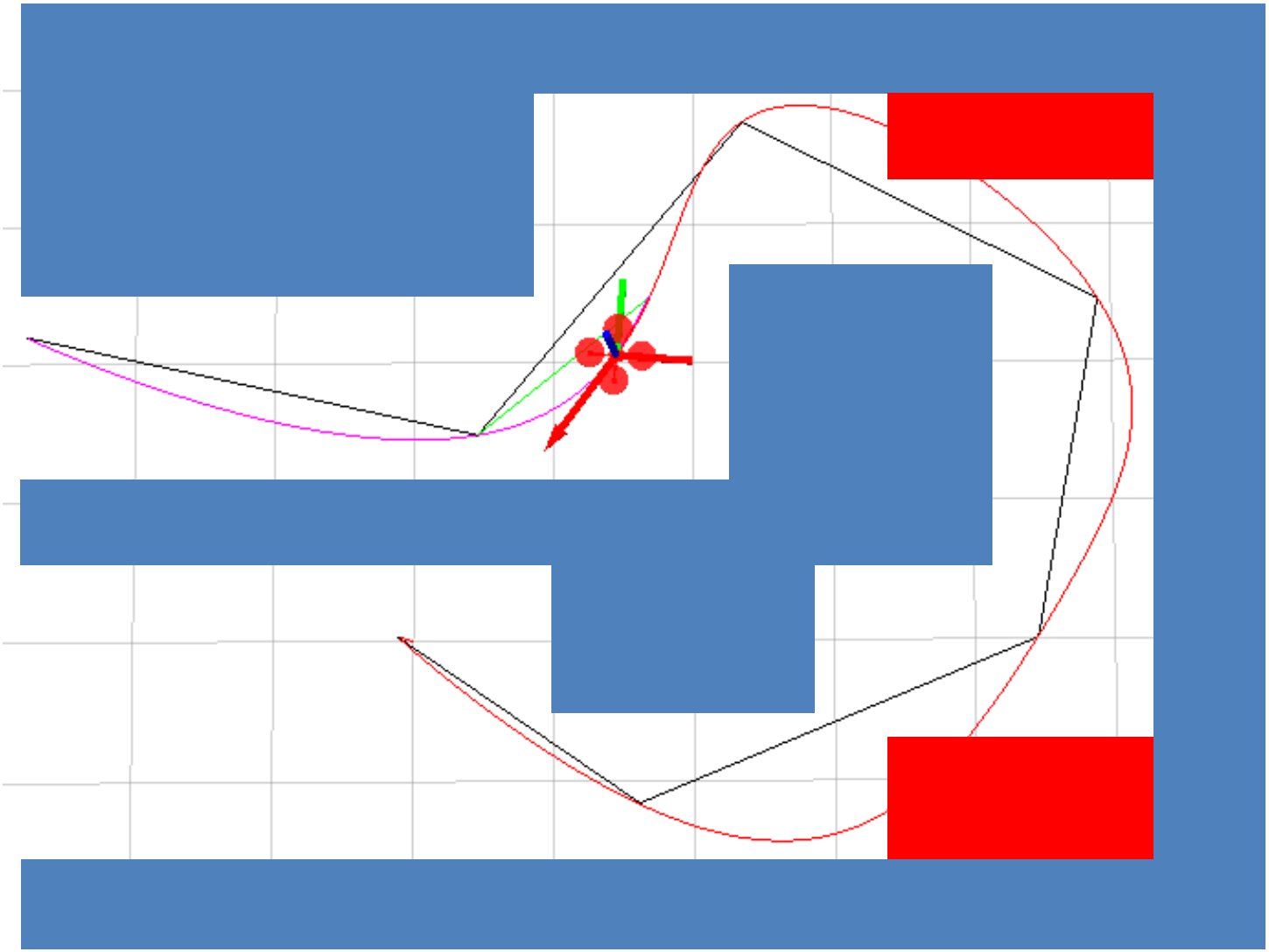
Minimum Snap Trajectory Generation

- Final trajectory



Minimum Snap Trajectory Generation

How to Ensure Collision-Free Trajectories?



Extension: Kinodynamic Search, Fast Collision Avoidance

Extension: Temporal Optimization, Teach and Repeat

Optimal Trajectory Generation for Quadrotor Teach-and-Repeat

Fei Gao, Luqi Wang, Kaixuan Wang, William Wu, Boyu Zhou, Luxin Han and Shaojie Shen

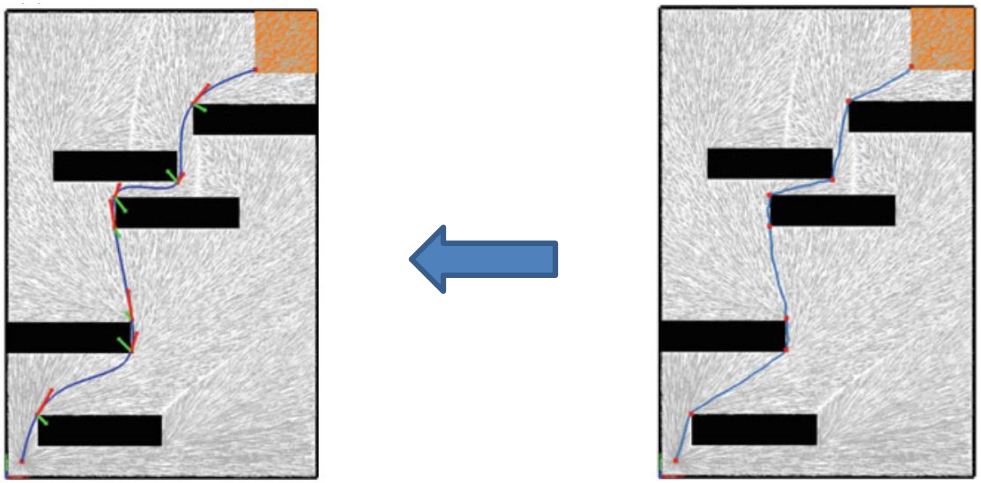
香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

香港科技大學-
大疆創新科技聯合實驗室
HKUST-DJI JOINT INNOVATION LABORATORY

Path Planning

Motivation

- Why we need path planning?
  - Fundamental problem in robotics - finding collision-free route from A to B
- Hierarchical approach (path planning + trajectory generation)
  - Low complexity solution
    - Path planning can be more efficient since it's in a much lower dimension state space.



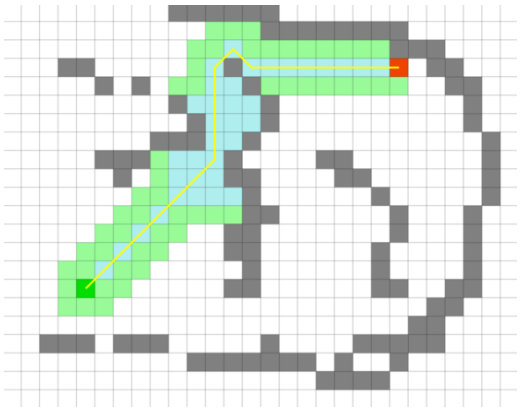
We already know how to fit the polynomial for given waypoints

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

Outline

- Configuration space obstacle
- Search-based methods
  - General graph search: DFS, BFS
  - A* search
- Sampling-based methods
  - Probabilistic roadmap (PRM)
  - Rapidly exploring random tree (RRT)

Grid-based graph:
use grid as vertices and grid connections as edges

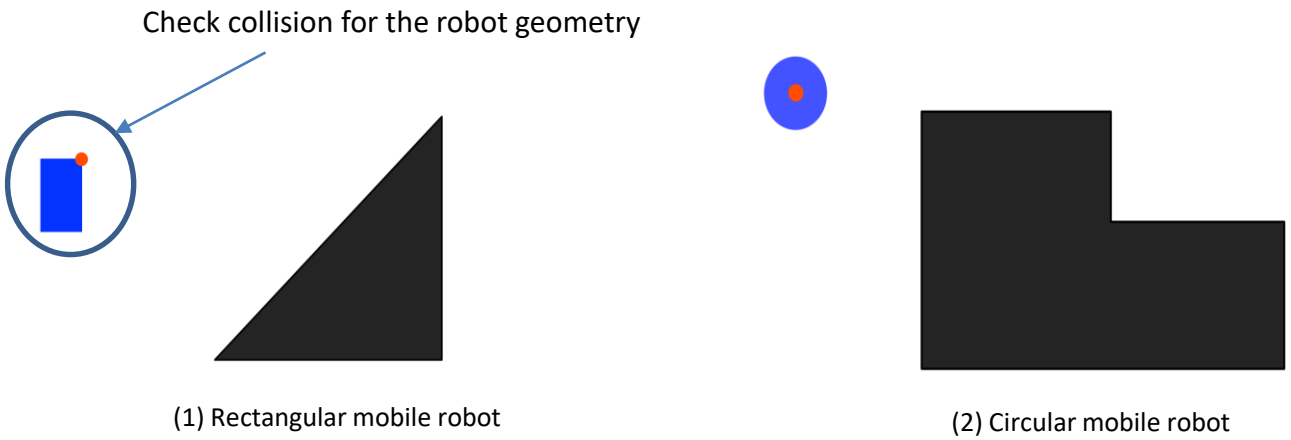


RRT example
See more examples at http://ompl.kavrakilab.org/gallery.html

A* search example
Try more search-based method at http://qiao.github.io/PathFinding.js/visual/

Workspace Space Obstacle

- Planning in *workspace*
  - Robot has different shape and size
  - Collision detection requires knowing the robot geometry - time consuming and hard



(1) Rectangular mobile robot

(2) Circular mobile robot

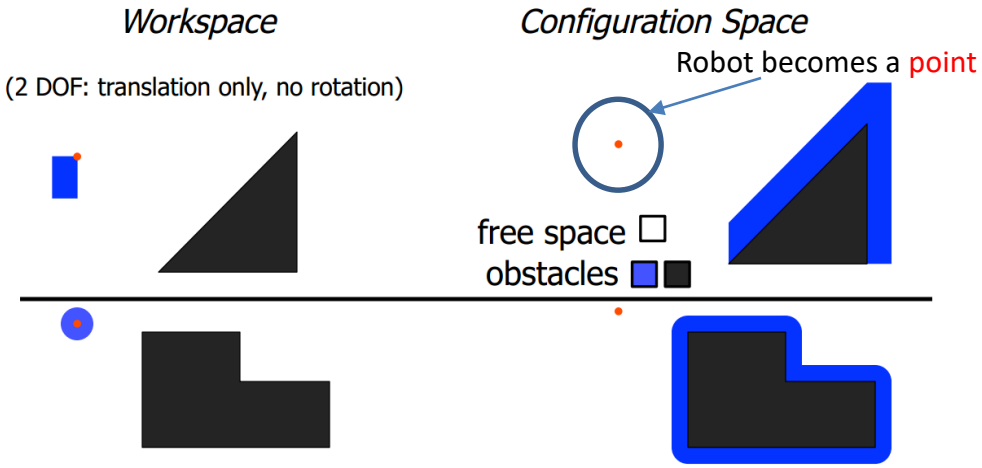
Configuration Space

- **Robot configuration**: a specification of the positions of all points of the robot
- **Robot degree of freedom (DOF):** The minimum number n of real-valued coordinates needed to represent the robot **configuration**
- **Robot configuration space**: a n-dim space containing all possible robot configurations, denoted as C-space
- Each robot pose is a point in the C-space
- Examples

	Configuration	C-space	DOF
Rigid rotation	$\boldsymbol{R}$	$SO(3)$	3
Rigid motion	$g = (\boldsymbol{R}, \boldsymbol{p})$	$SE(3)$	6
Flat outputs	$\boldsymbol{\sigma} = (\psi, \boldsymbol{p})$	$SO(2) \times \mathbb{R}^3$	4

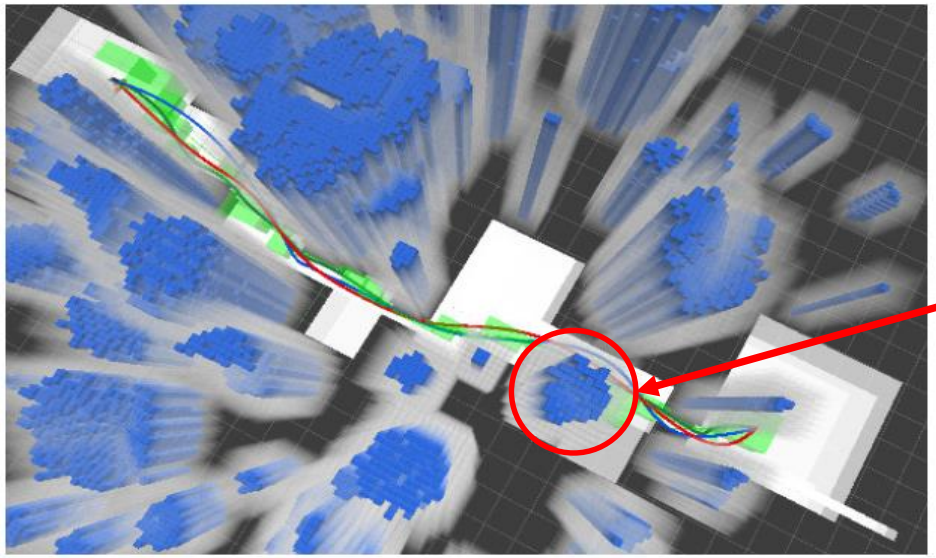
Configuration Space Obstacle

- Planning in *configuration space*: C-space
  - Robot is represented by a point in C-space, e.g. position (a point in $\mathbb{R}^3$), pose (a point in $SE(3)$) , etc.
  - Obstacles need to be represented in configuration space (one-time work prior to motion planning), called configuration space obstacle, or C-obstacle
  - C-space = (C-obstacle) ∪ (C-free)
  - The path planning is finding a path between start point q_{start} and goal point q_{goal} within C-free



Workspace and Configuration Space Obstacles

- In *workspace*
  - Robot has shape and size (i.e. hard for motion planning)
- In *configuration space*: C-space
  - Robot is a point (i.e. easy for motion planning)
  - Obstacle are represented in C-space prior to motion planning
- Representing an obstacle in C-space can be extremely complicated. So approximated (but more conservative) representations are used in practice.

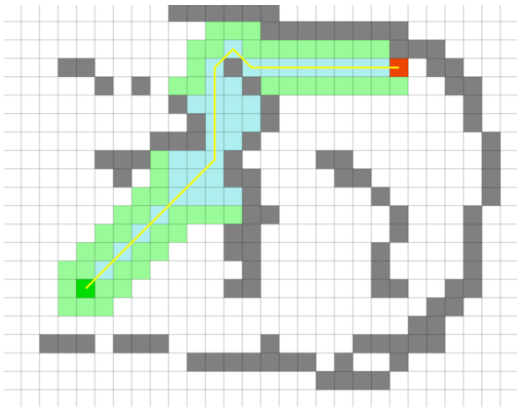


If we model the robot conservatively as a ball with radius δ_r,
then the C-space can be constructed by inflating obstacle at all directions by δ_r.

Outline

- Configuration space obstacle
- Search-based methods
  - General graph search: DFS, BFS
  - A* search
- Sampling-based methods
  - Probabilistic roadmap (PRM)
  - Rapidly exploring random tree (RRT)

Grid-based graph:
use grid as vertices and grid connections as edges

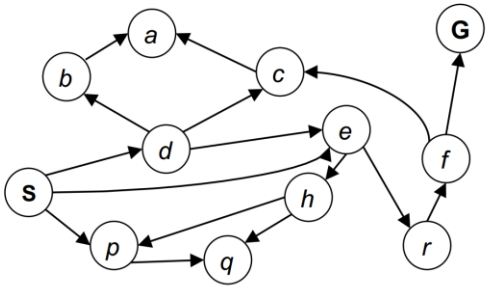


RRT example
See more examples at http://ompl.kavrakilab.org/gallery.html

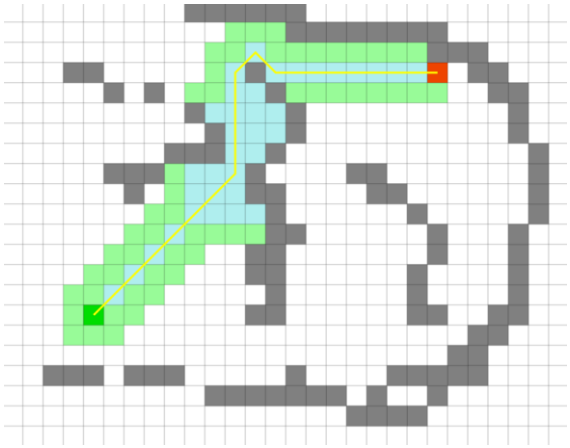
A* search example
Try more search-based method at http://qiao.github.io/PathFinding.js/visual/

Search-based Method

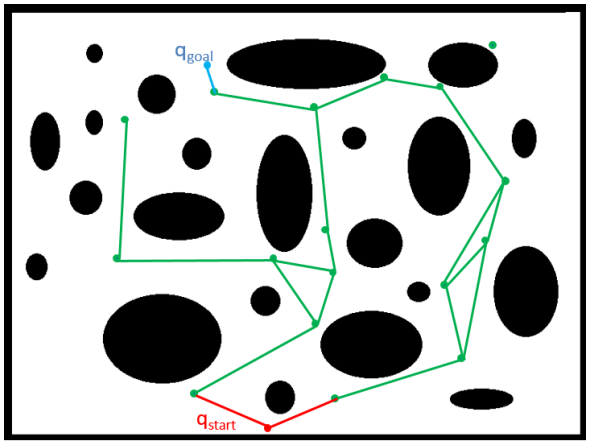
- State space graph: a mathematical representation of a search algorithm
  - For every search problem, there's a corresponding state space graph
  - Connectivity between nodes in the graph is represented by (directed or undirected) edges



Ridiculously tiny search graph for a tiny search problem



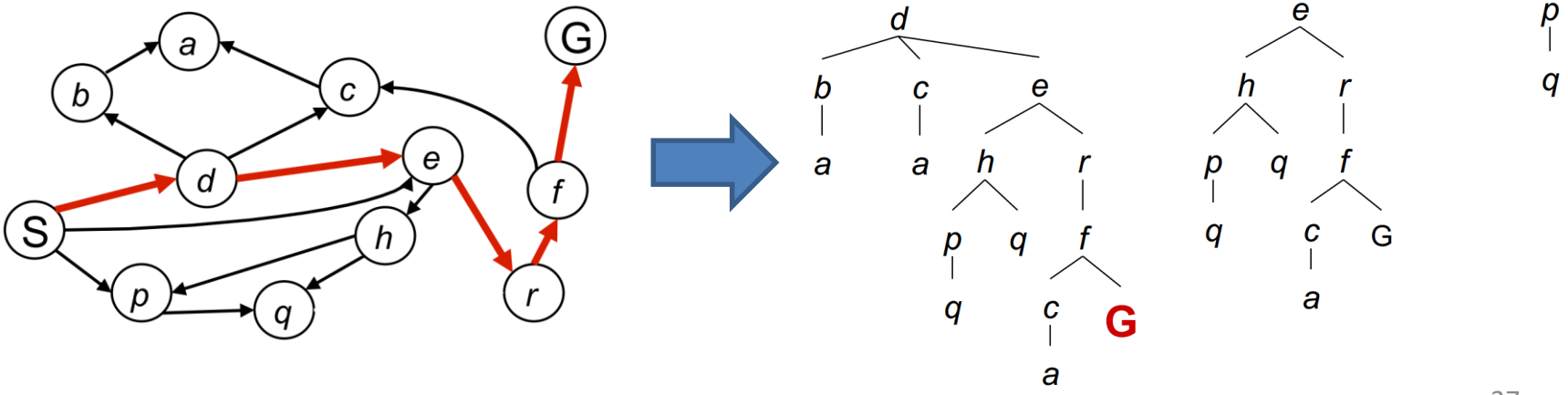
Grid-based graph: use grid as vertices and grid connections as edges



The graph generated by probabilistic roadmap (PRM)

From Graph to Search Tree

- The search always start from start state X_S
  - Searching the graph produces a search tree, this is a "what if" tree of plans and outcomes
  - Back-tracing a node in the search tree gives us a path from the start state to that node
  - For many problems we can never actually build the whole tree, too large or inefficient – we only want to reach the goal node asap.

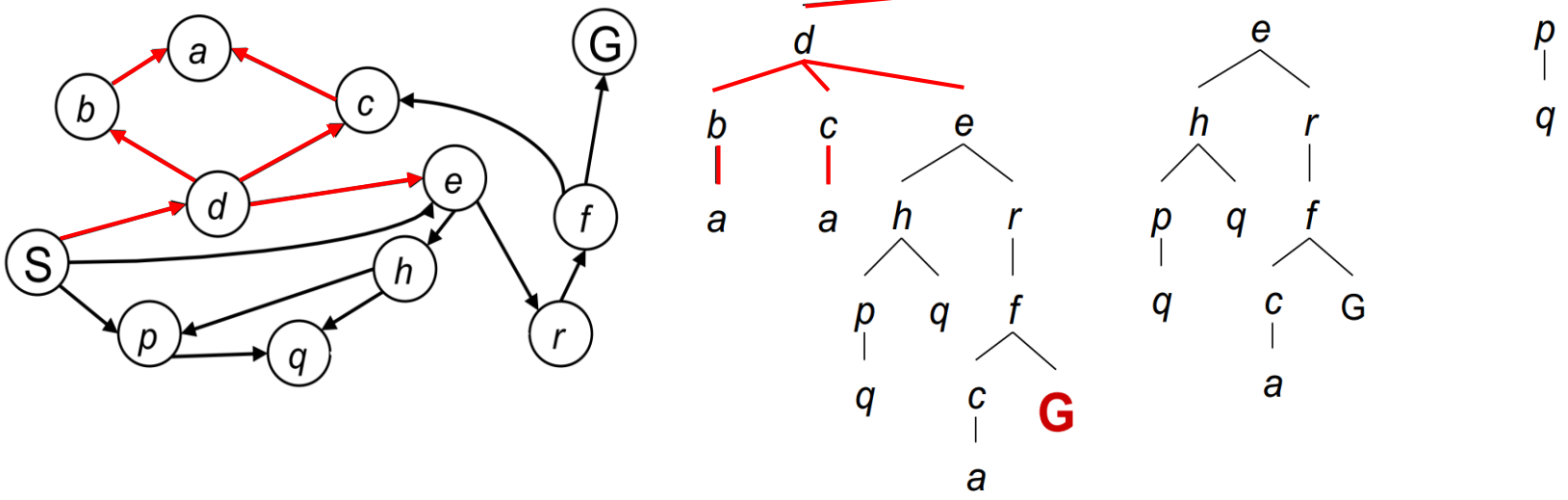


How to Construct a Search Tree?

- Maintain a container to store all the nodes to be visited
  - Intuition: When we "discover" a node, we store it in our "memory". We can only visit one node at a time, but we can teleport to any node that we discover before.
- The container is initialized with the start state X_S
- Loop
  - Remove a node from the container according to some pre-defined score function
    - Visit a node
  - Expansion: Obtain all neighbors of the node, and push them into the container
    - Discover all its neighbors
- End Loop
- Question 1: When to end the loop?
  - Possible option: End the loop when the container is empty
- Question 2: What if the graph is cyclic?
  - When a node is removed (visited) from the container, it should never be added back to the container again
- Question 3: In what way to remove the right node such that the goal state can be reached as soon as possible, which results in less expansion of the graph node.

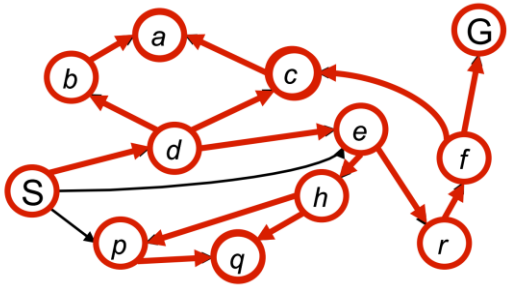
Depth First Search (DFS)

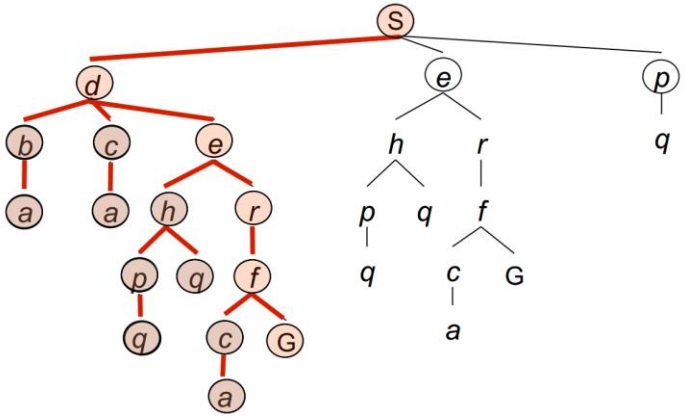
- Strategy: remove (visit) the deepest node in the container

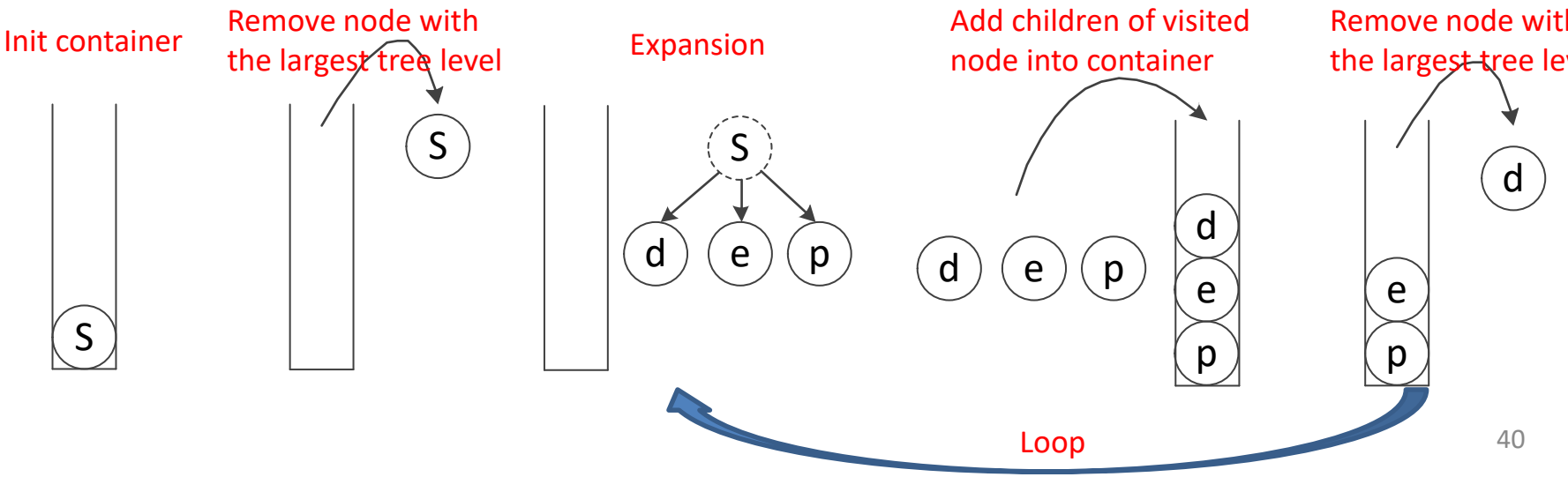


Depth First Search (DFS)

- Implementation: maintain a last in first out (LIFO) container (i.e. stack)

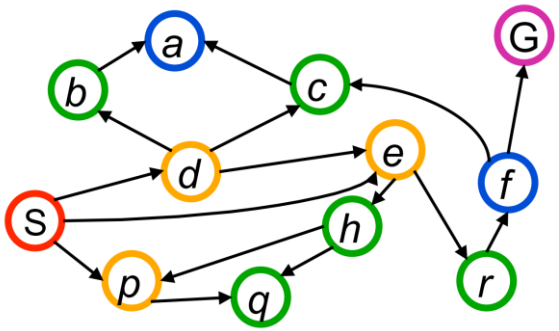


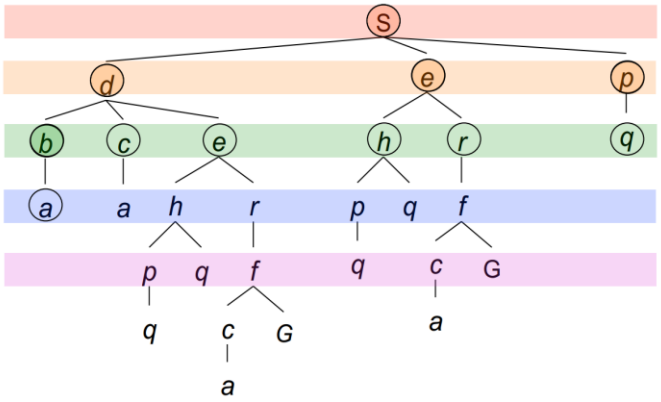




Breadth First Search (BFS)

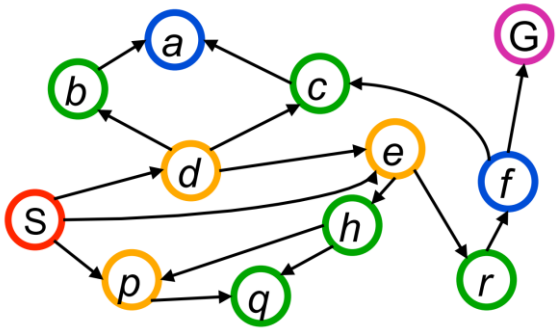
- Strategy: remove (visit) the shallowest node in the container

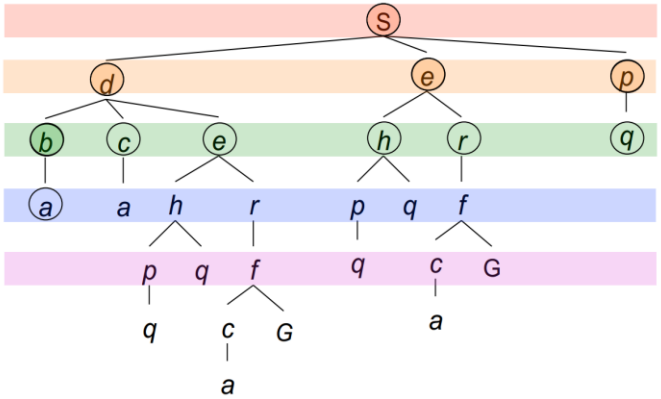


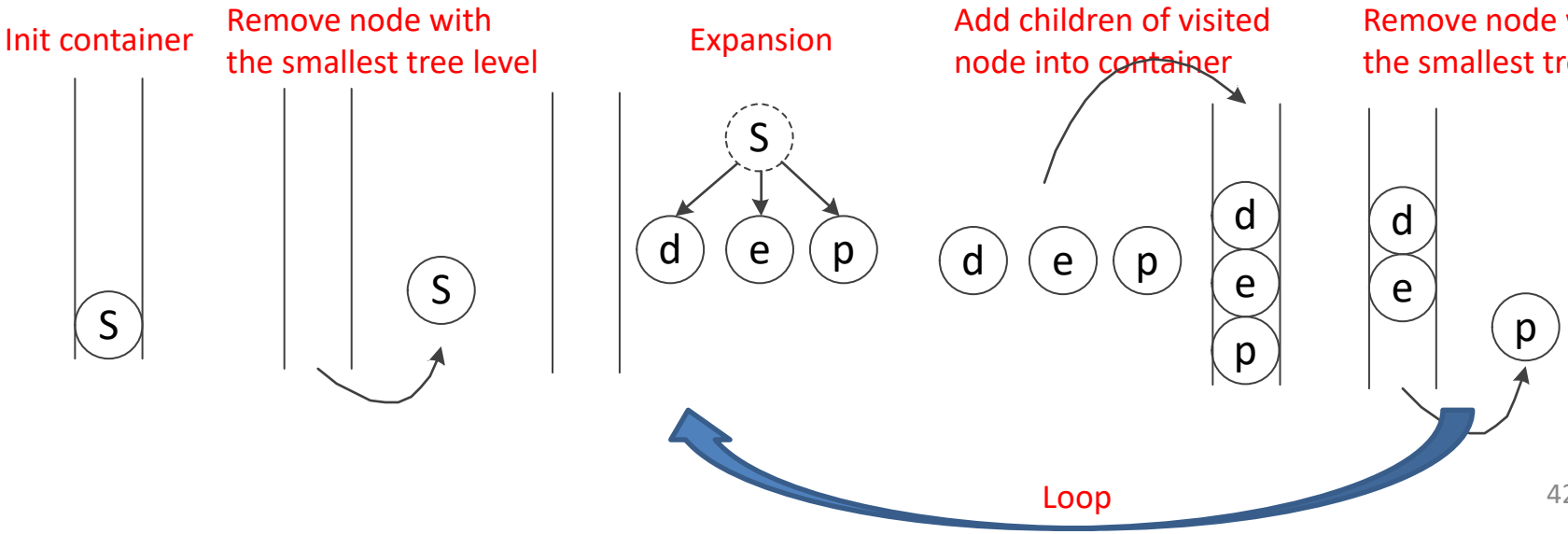


Breadth First Search (BFS)

- Implementation: maintain a first in first out (FIFO) container (i.e. queue)

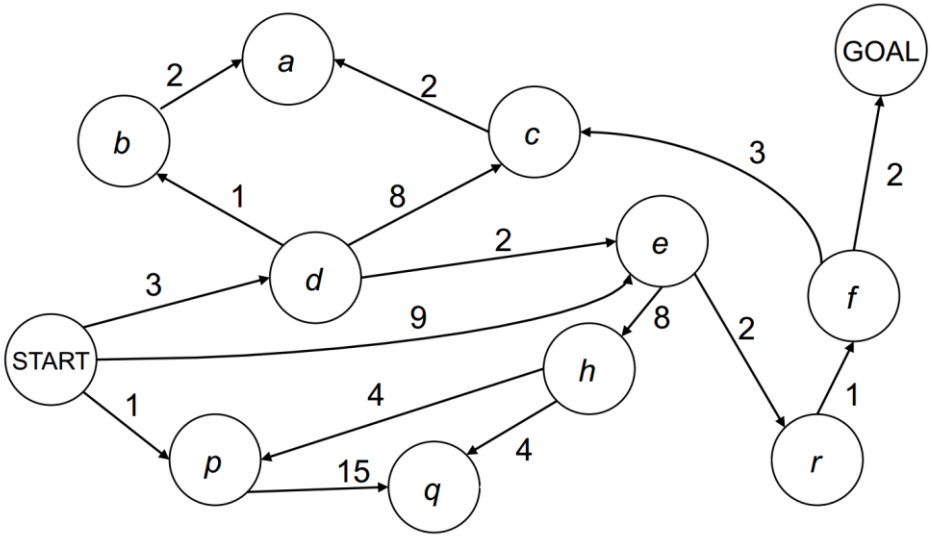






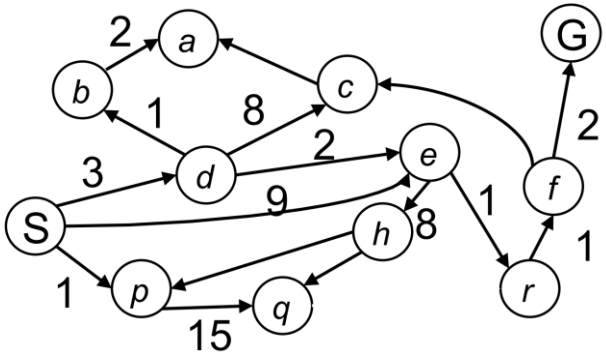
Costs on Actions

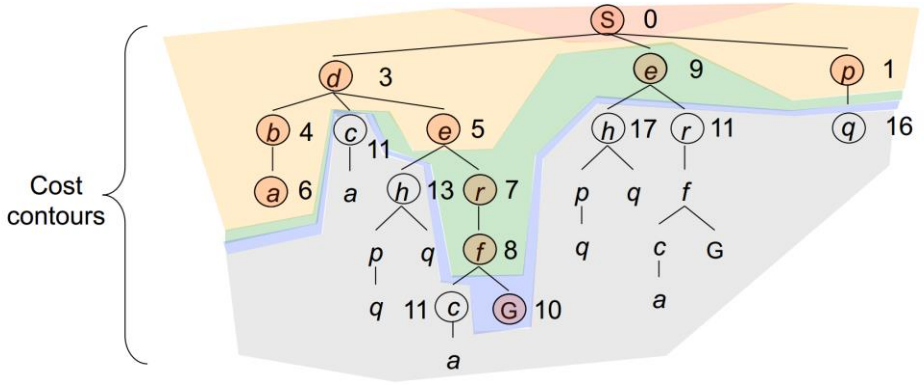
- A practical search problem has a cost "C" from a node to its neighbor
  - Length, time, energy, etc.
- When all weight are 1, BFS finds the least-cost path with minimal steps
- For general cases, how to find the least-cost path as soon as possible?



Dijkstra's Algorithm

- Strategy: remove (visit) the node with cheapest accumulated cost g(n)
  - g(n): The current best estimates of the accumulated cost from the start state to node "n"
  - Update the accumulated costs g(m) for all unvisited neighbors "m" of node "n"
  - A node that has been visited is guaranteed to have the smallest cost from the start state

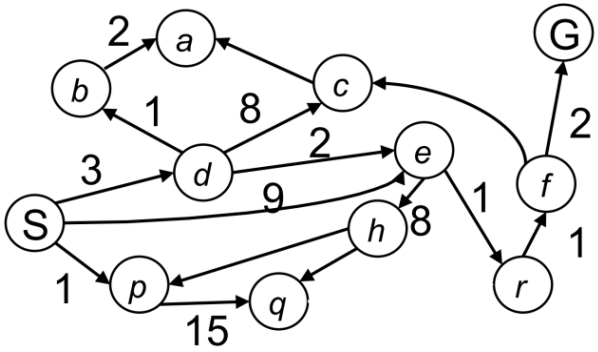


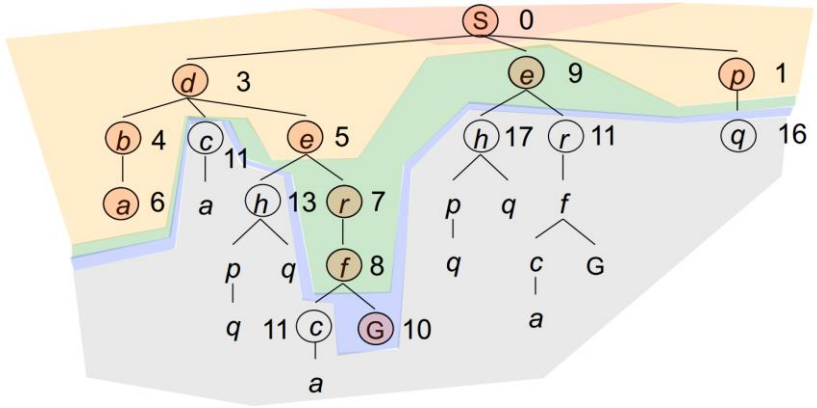


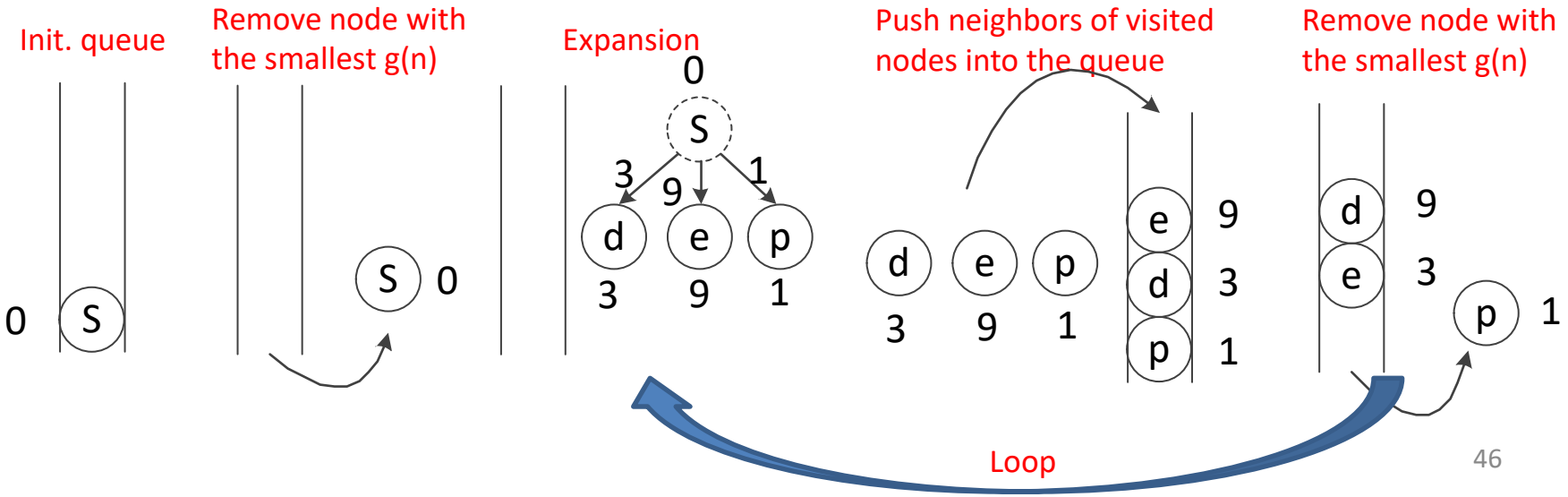
Dijkstra's Algorithm

- Maintain a priority queue to store all the nodes to be visited
- The priority queue is initialized with the start state X_S
- Assign $g(X_S)=0$, and g(n)=infinite for all other nodes in the graph
- Loop
  - If the queue is empty, return FALSE; break;
  - Remove the node "n" with the lowest g(n) from the priority queue
  - Mark node "n" as visited
  - If the node "n" is the goal state, return TRUE; break;
  - For all unvisited neighbors "m" of node "n"
    - If g(m) = infinite
      - Push node "m" into the queue
    - If $g(m) > g(n) + C_{nm}$
      - $g(m)= g(n) + C_{nm}$
  - end
- End Loop

Dijkstra's Algorithm

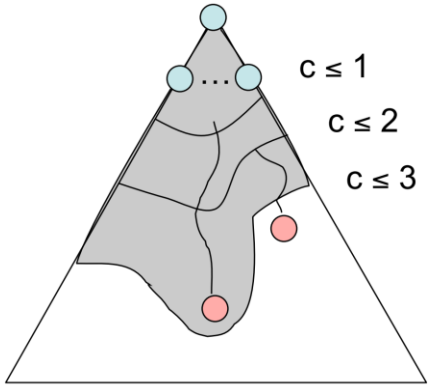


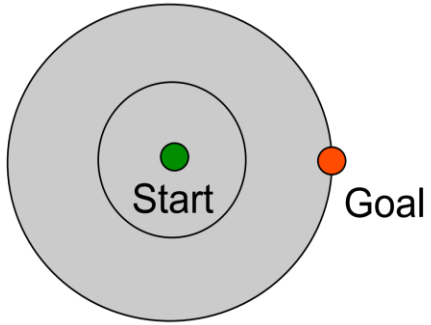




Issues of Dijkstra's Algorithm

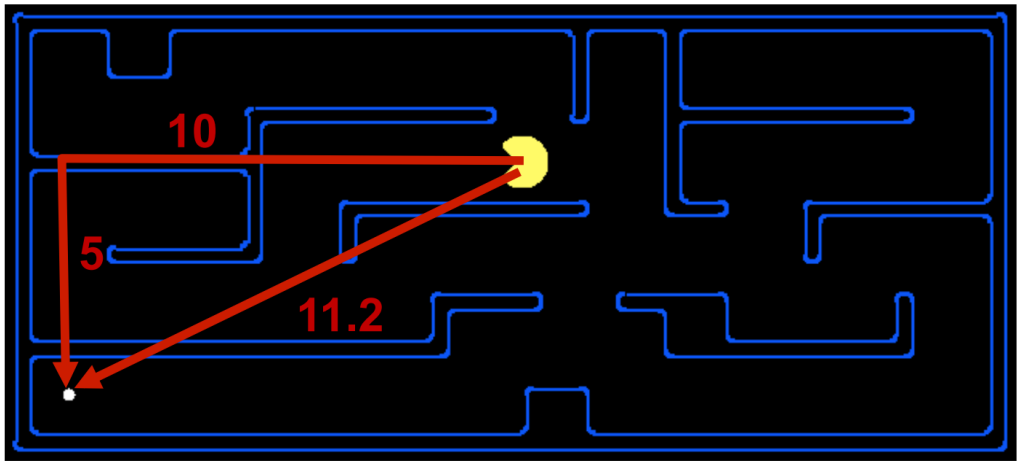
- The good:
  - Complete and optimal
- The bad:
  - Can only see the cost *accumulated so far* (i.e. the uniform cost), thus exploring next state in every "direction"
  - No information about goal location





Search Heuristics

- Overcome the shortcomings of uniform cost search by inferring the least cost to goal (i.e. goal cost)
- Designed for particular search problem
- Examples: Manhattan distance VS. Euclidean distance



A* Search: Combining Dijkstra's and a Heuristic

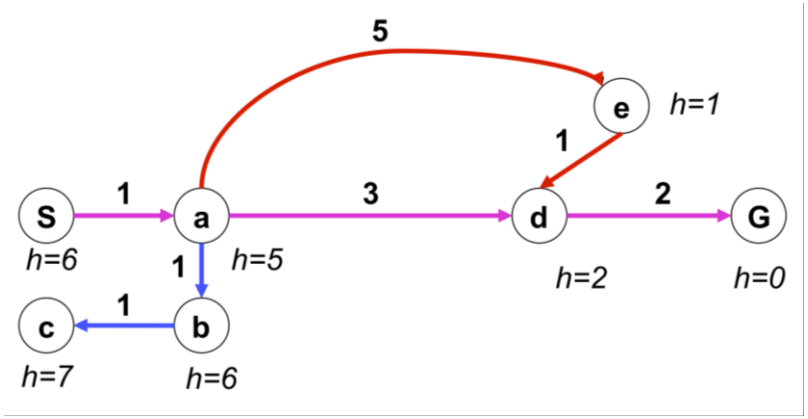
- Accumulated cost
  - g(n): The current best estimates of the accumulated cost from the start state to node "n"
- Heuristic
  - h(n): The estimated least cost from node n to goal state (i.e. goal cost)
- The least estimated cost from start state to goal state passing through node "n" is f(n) = g(n) + h(n)
- Strategy: remove (visit) the node with cheapest f(n) = g(n) + h(n)
  - Update the accumulated costs g(m) for all unvisited neighbors "m" of node "n"
  - A node that has been visited is guaranteed to have the smallest cost from the start state

A* Algorithm

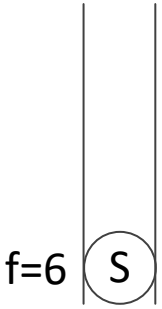
- Maintain a priority queue to store all the nodes to be visited
- The heuristic function h(n) for all nodes are pre-defined
- The priority queue is initialized with the start state X_S
- Assign $g(X_S)=0$, and g(n)=infinite for all other nodes in the graph
- Loop
  - If the queue is empty, return FALSE; break;
  - Remove the node "n" with the lowest f(n)=g(n)+h(n) from the priority queue
  - Mark node "n" as visited
  - If the node "n" is the goal state, return TRUE; break;
  - For all unvisited neighbors "m" of node "n"
    - If g(m) = infinite
      - Push node "m" into the queue
    - If $g(m) > g(n) + C_{nm}$
      - $g(m)= g(n) + C_{nm}$
  - end
- End Loop

Only difference comparing to Dijkstra's algorithm

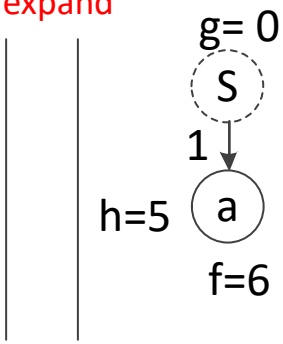
A* Search Example



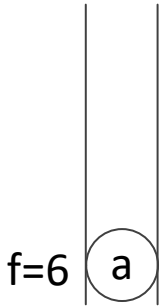
Initial queue



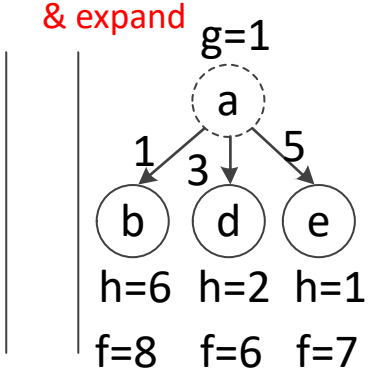
1st dequeue & expand



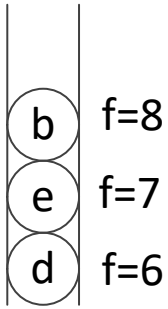
1st enqueue



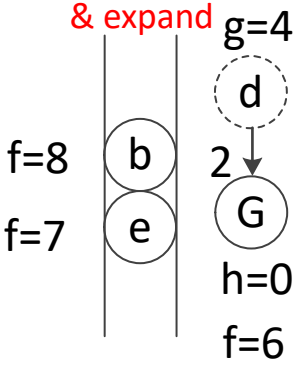
2nd dequeue & expand



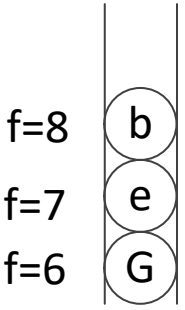
2nd enqueue



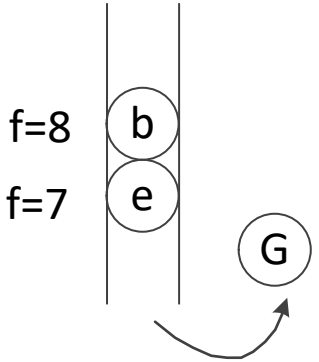
3rd dequeue & expand



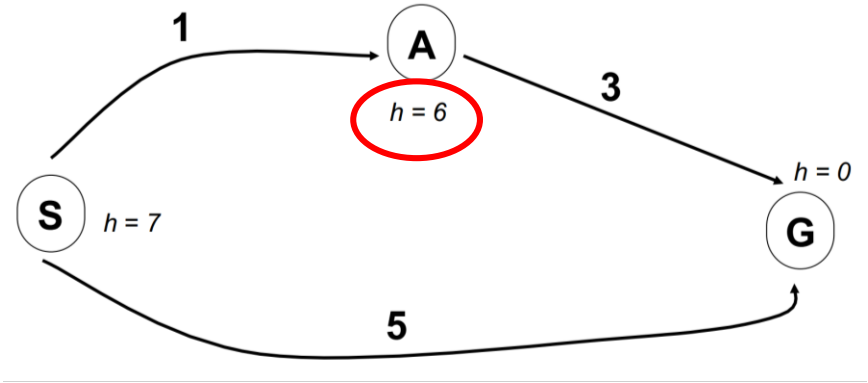
3rd enqueue



4th dequeue



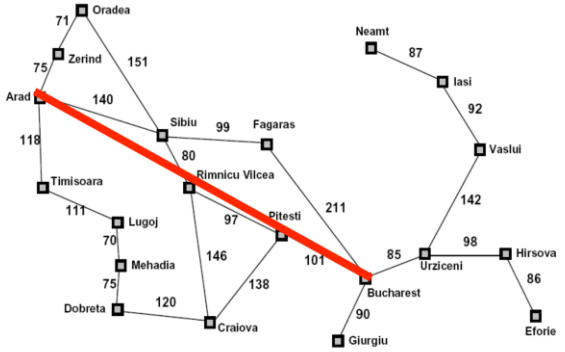
Is A* Optimal?

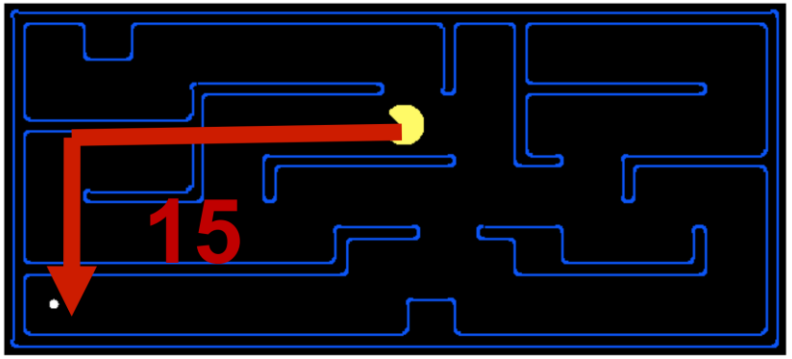


- What went wrong?
- For node A: actual least cost to goal (i.e. goal cost) < estimated least cost to goal (i.e. heuristic)
- We need the estimate to be less than actual least cost to goal (i.e. goal cost) for all nodes!

Admissible Heuristics

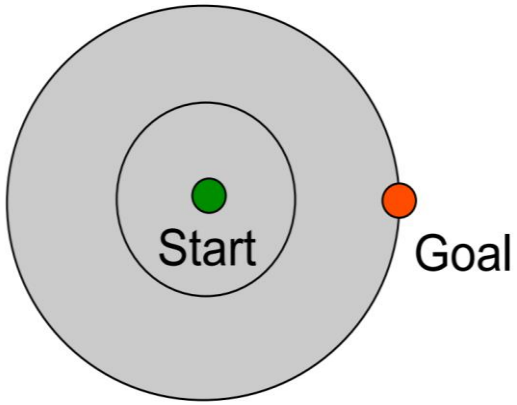
- A Heuristic h is admissible (optimistic) if:
  - $h(n) < h^*(n)$ for all node "n", where $h^*(n)$ is the true least cost to goal from node "n"
- If the heuristic is admissible, the A* search is optimal
- Coming up with admissible heuristics is most of what's involved in using A* in practice.
- Example:

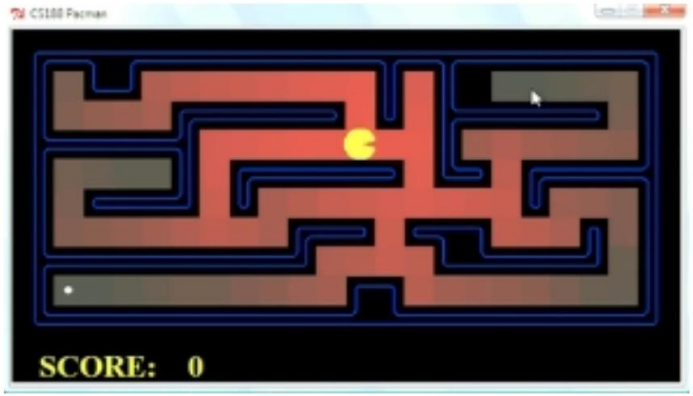




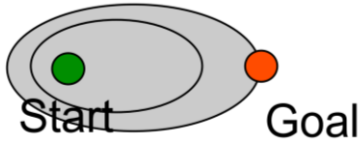
Dijkstra's VS A*

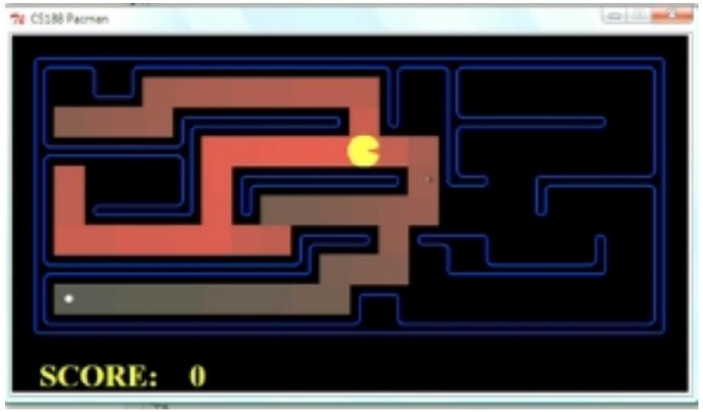
- Dijkstra's algorithm visits in all directions





- A* visits mainly towards the goal, but does not hedge its bets to ensure optimality

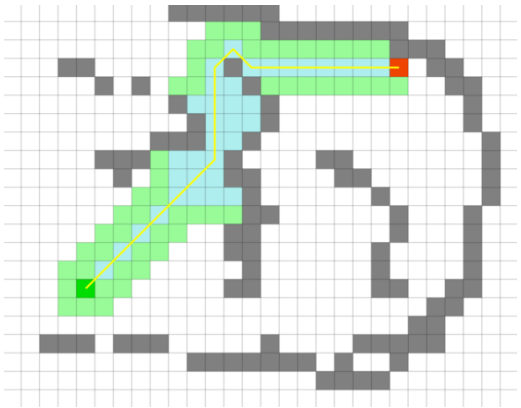




Outline

- Configuration space obstacle
- Search-based methods
  - General graph search: DFS, BFS
  - A* search
- Sampling-based methods
  - Probabilistic roadmap (PRM)
  - Rapidly exploring random tree (RRT)

Grid-based graph:
use grid as vertices and grid connections as edges



RRT example
See more examples at http://ompl.kavrakilab.org/gallery.html

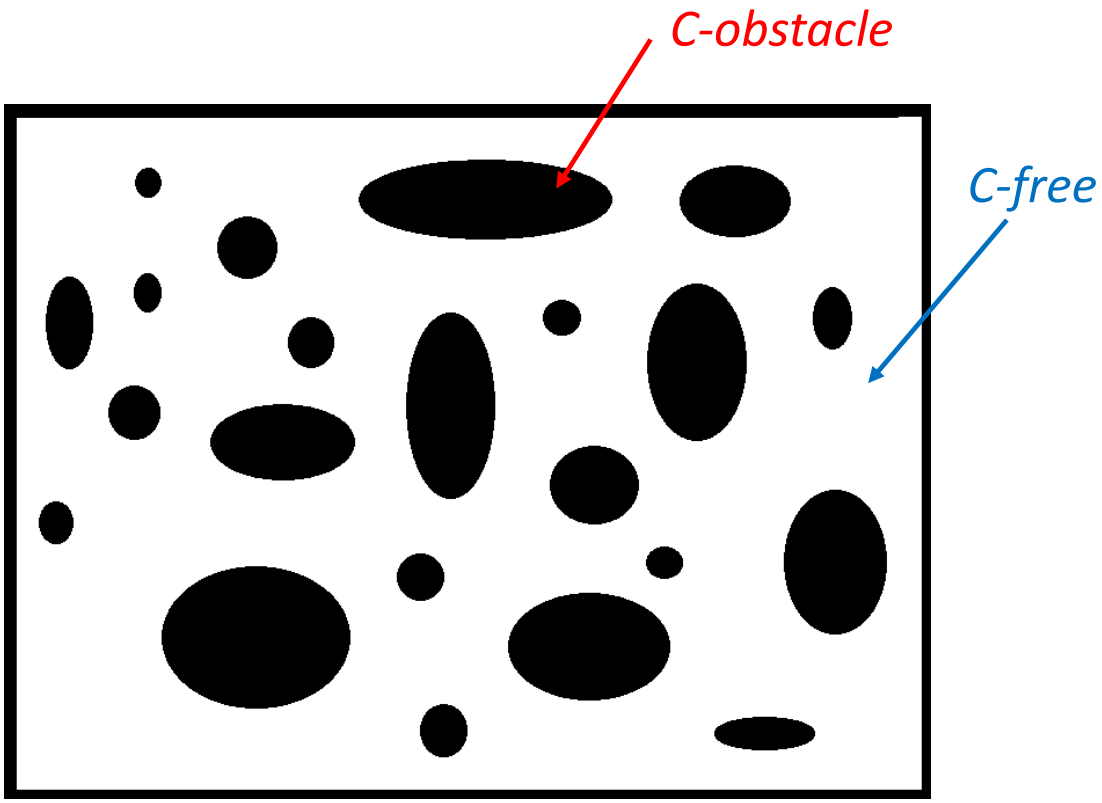
A* search example
Try more search-based method at http://qiao.github.io/PathFinding.js/visual/

Probabilistic Roadmap (PRM)

- Basic Idea
  - Build a **graph** to characterizes the free configuration space in probabilistic manner, and then use graph search algorithm to find a path
- Algorithm
  - Initialize set of points with q_{start} and q_{goal}
  - Randomly sample points in configuration space
  - Connect nearby points if they can be reached from each other
  - Find path from q_{start} to q_{goal} in the graph
- Step by step illustration as follows

Probabilistic Roadmap (PRM)

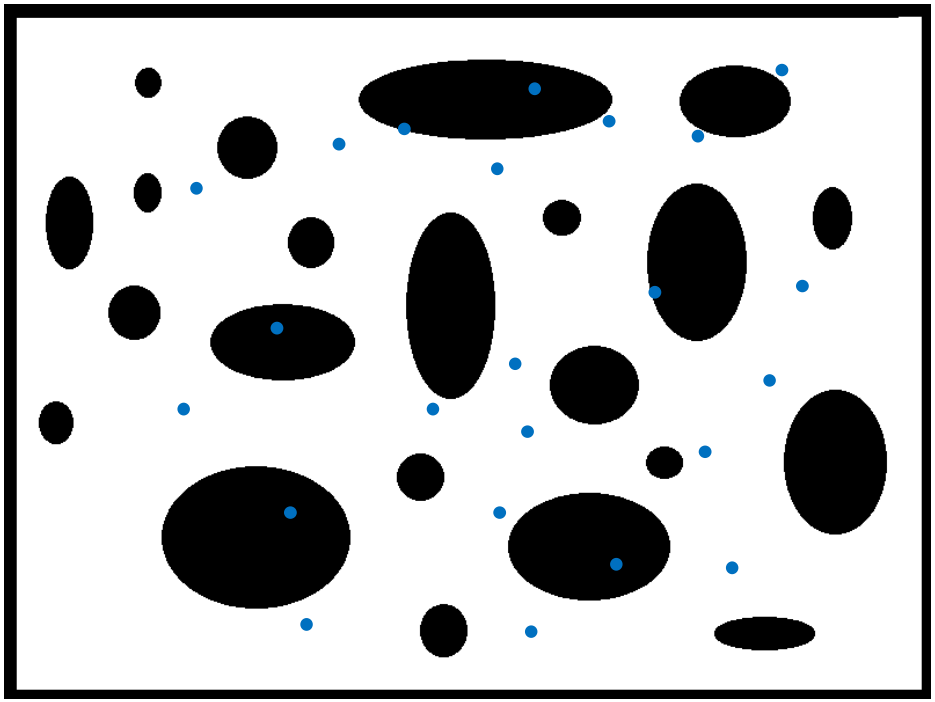
- Free space and obstacle space



C-space

Probabilistic Roadmap (PRM)

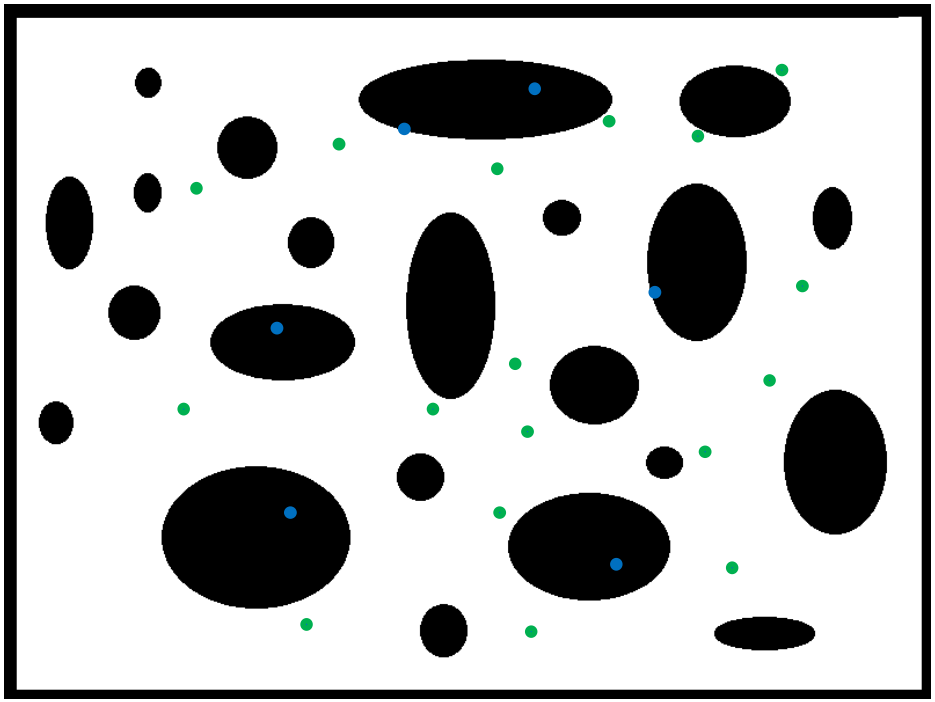
- Configurations are sampled by picking each coordinate at random.



C-space

Probabilistic Roadmap (PRM)

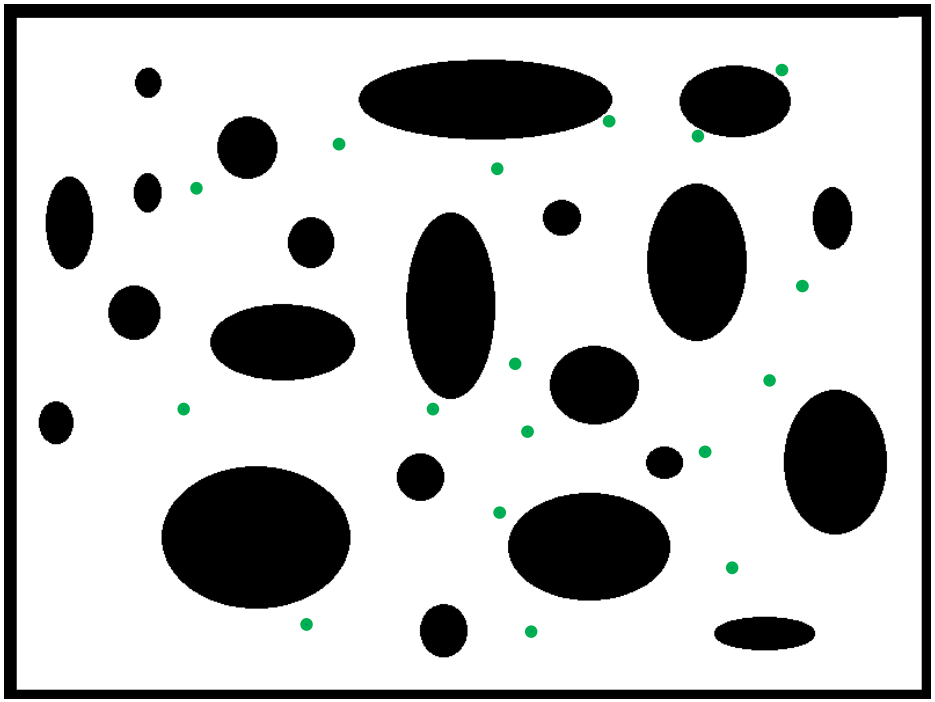
- Sampled configurations are tested for collision.



C-space

Probabilistic Roadmap (PRM)

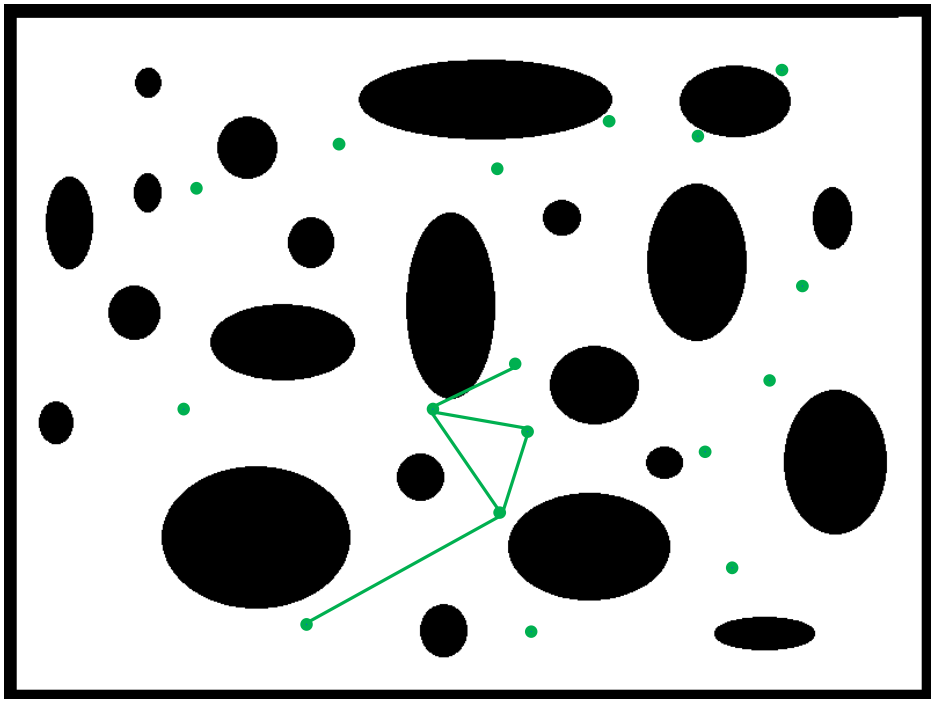
- The collision-free configurations are retained as milestones.



C-space

Probabilistic Roadmap (PRM)

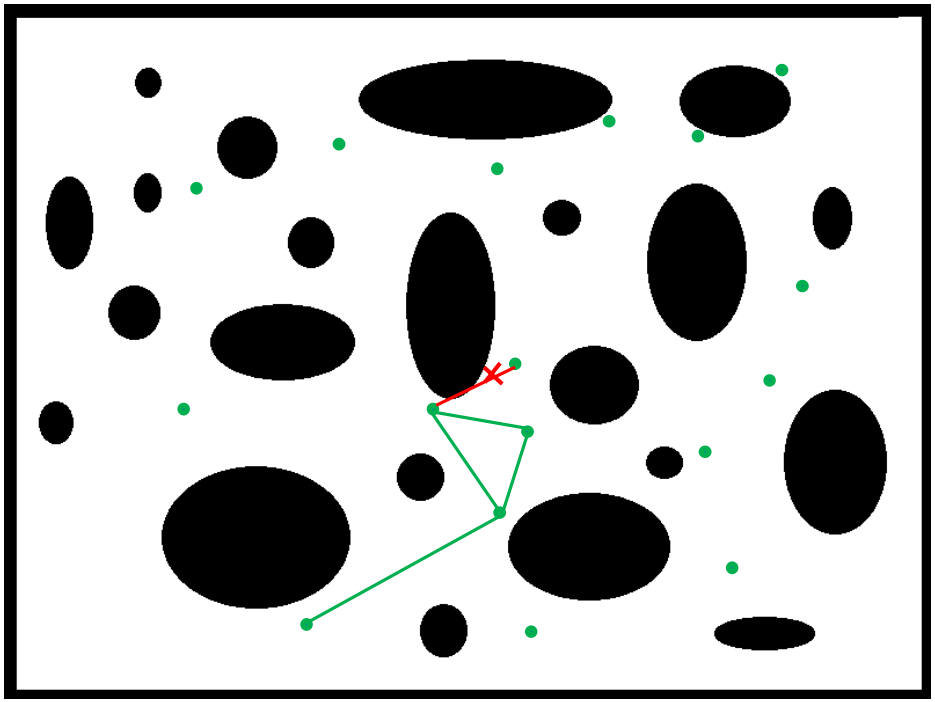
- Each milestone is linked by straight paths to its nearest neighbors.



C-space

Probabilistic Roadmap (PRM)

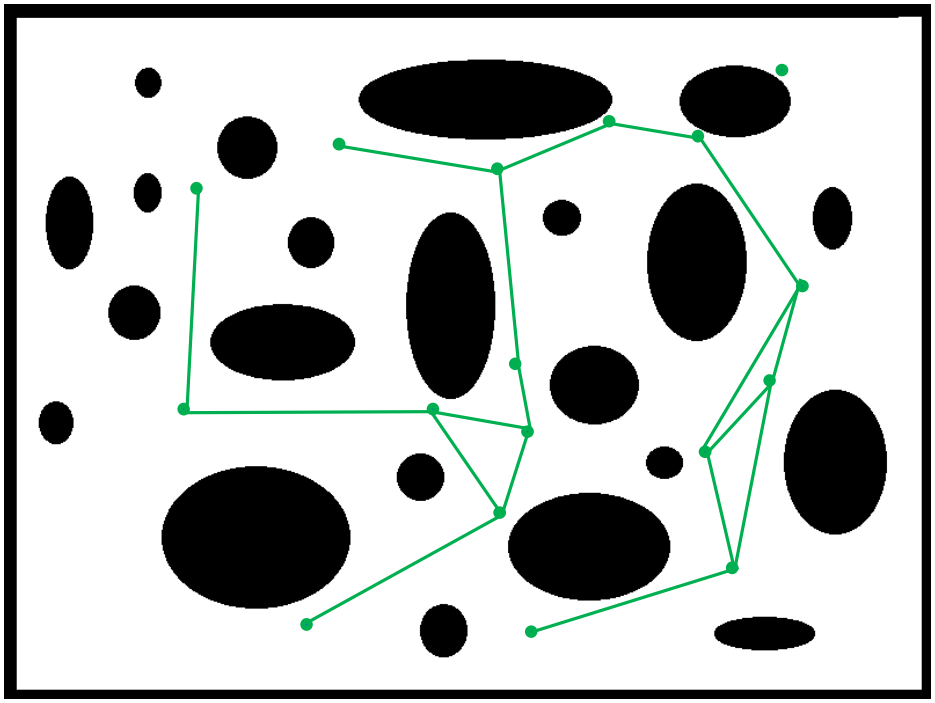
- Eliminate collision links.



C-space

Probabilistic Roadmap (PRM)

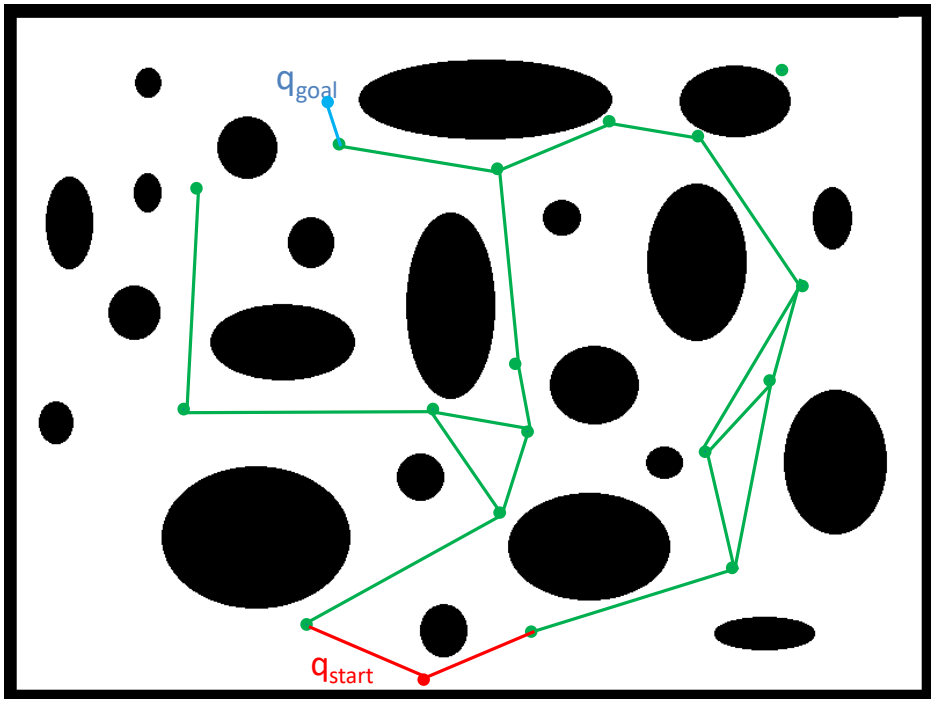
- The collision-free links are retained as local paths to form the PRM.



C-space

Probabilistic Roadmap (PRM)

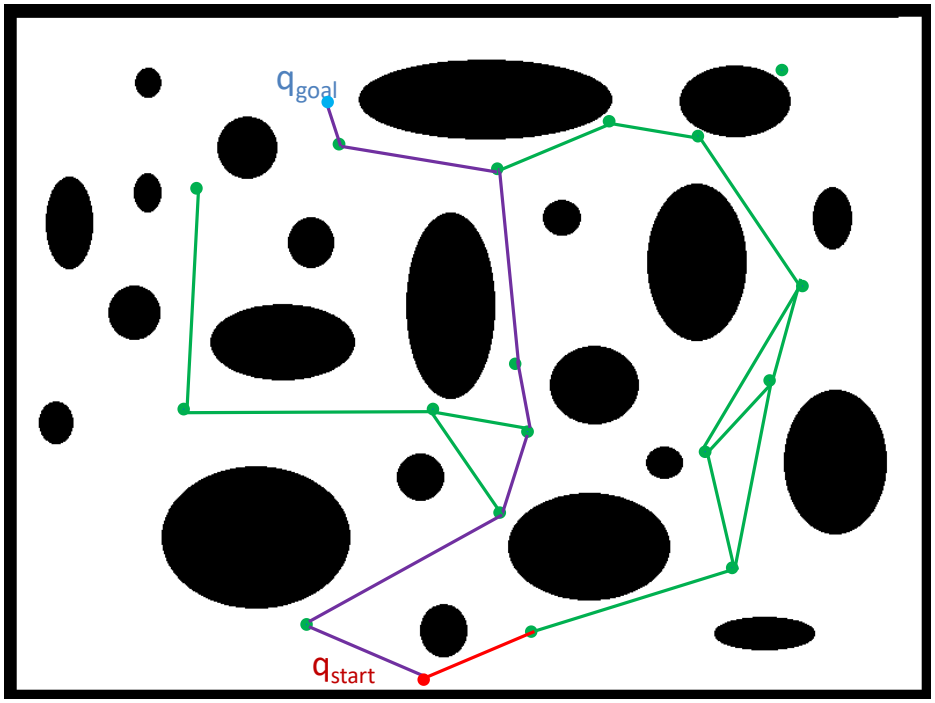
- Connect the start and goal point to the roadmap.



C-space

Probabilistic Roadmap (PRM)

- Search the roadmap for a path from start to goal point (e.g. A* algorithm).



C-space

PRM's Pros and Cons

- Pros:
  - Probabilistically complete: i.e., with probability one, if run for long enough the graph will contain a solution path if one exists.
  - Can cope with high-dimensional system
- Cons:
  - Collision detection takes majority of time
  - Suboptimal solution if only limited samples are given
  - Build graph over C-space but no particular focus on generating a path

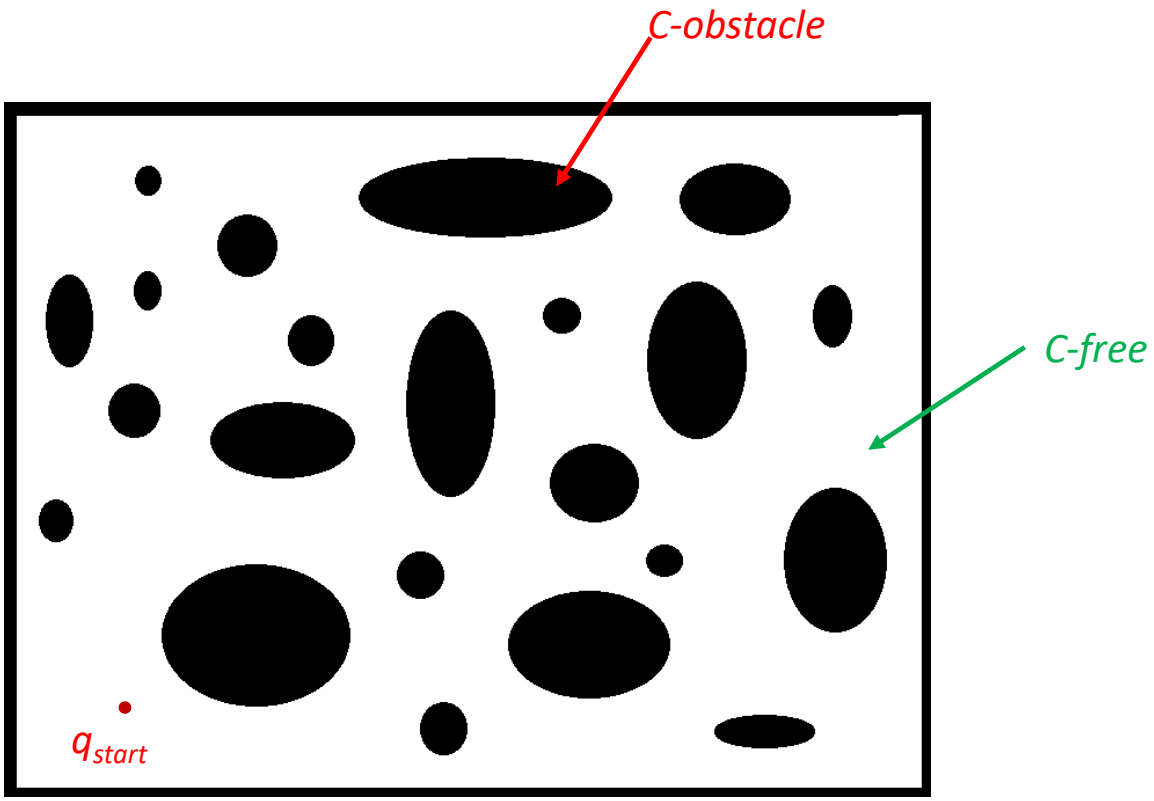
Rapidly exploring Random Trees (RRT)

- Basic Idea
  - Starting from the start configuration q_{start}, build up a **tree** through generating "next configuration"
- Algorithm

Algorithm BuildRRT
Input: Start configuration q_{start}, number of vertices in RRT K
Output: RRT T
L1: G.init(q_{start})
L2: **for** k = 1 **to** K
L3: $q_{rand} \leftarrow$ RAND_CONF();
L4: $q_{near} \leftarrow$ NEAREST_VERTEX(q_{rand}, T);
L5: $q_{new} \leftarrow$ NEW_CONF(q_{near}, q_{rand});
L6: T.add_vertex(q_{new}); T.add_edge(q_{near}, q_{new})
L7: **return** G

Rapidly exploring Random Trees (RRT)

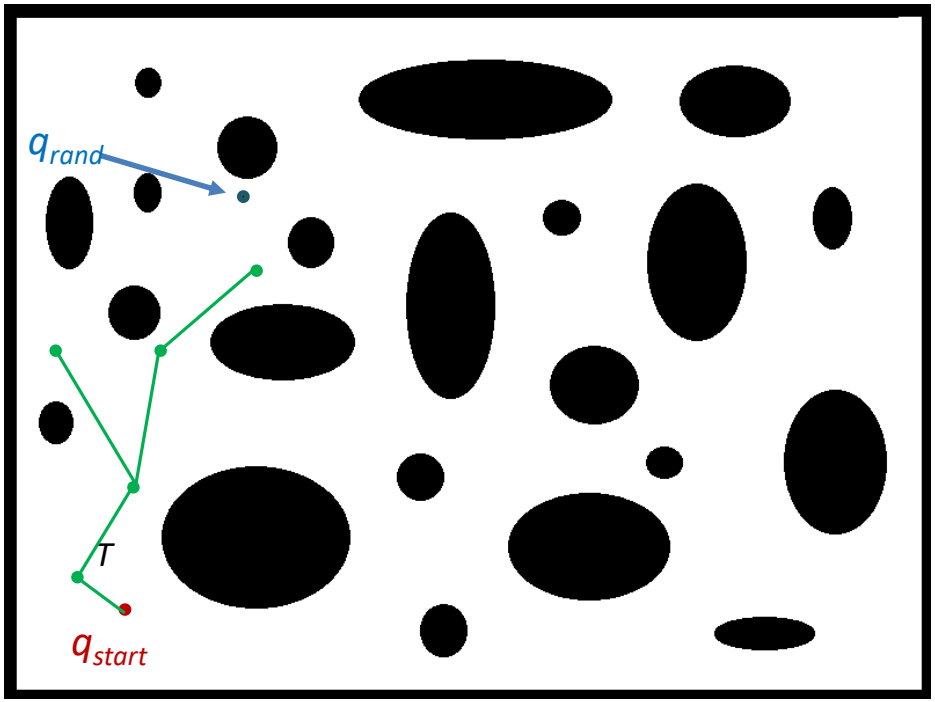
- L1: T.init(q_{start}); Initialize



C-space

Rapidly exploring Random Trees (RRT)

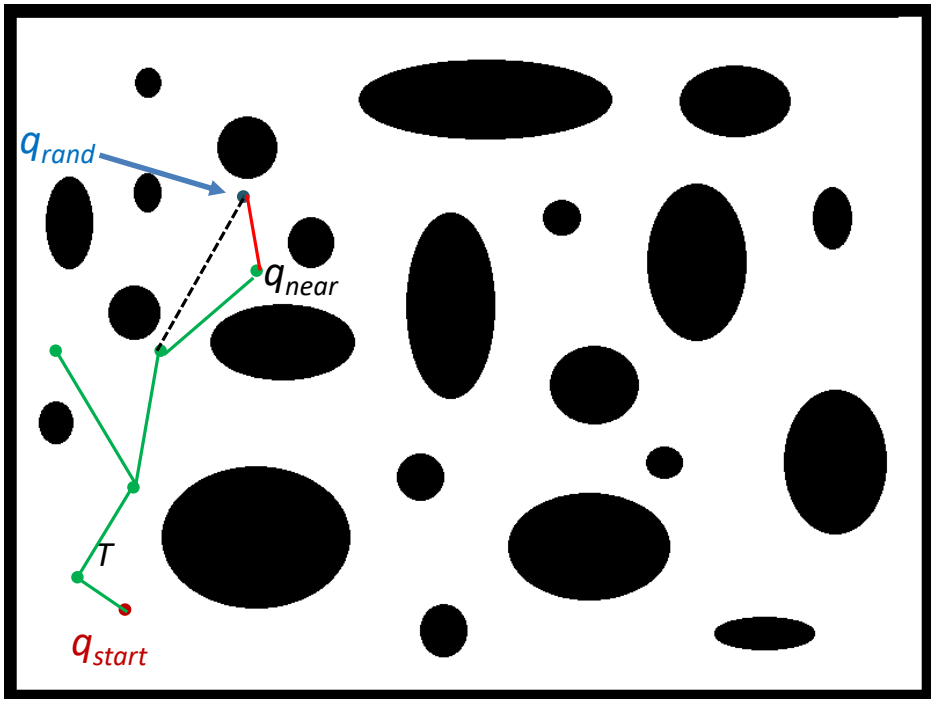
- L3: $q_{rand} \leftarrow$ RAND_CONF(); generate a random configuration
  - q_{rand} is sampled from a uniform distribution on C-space



C-space

Rapidly exploring Random Trees (RRT)

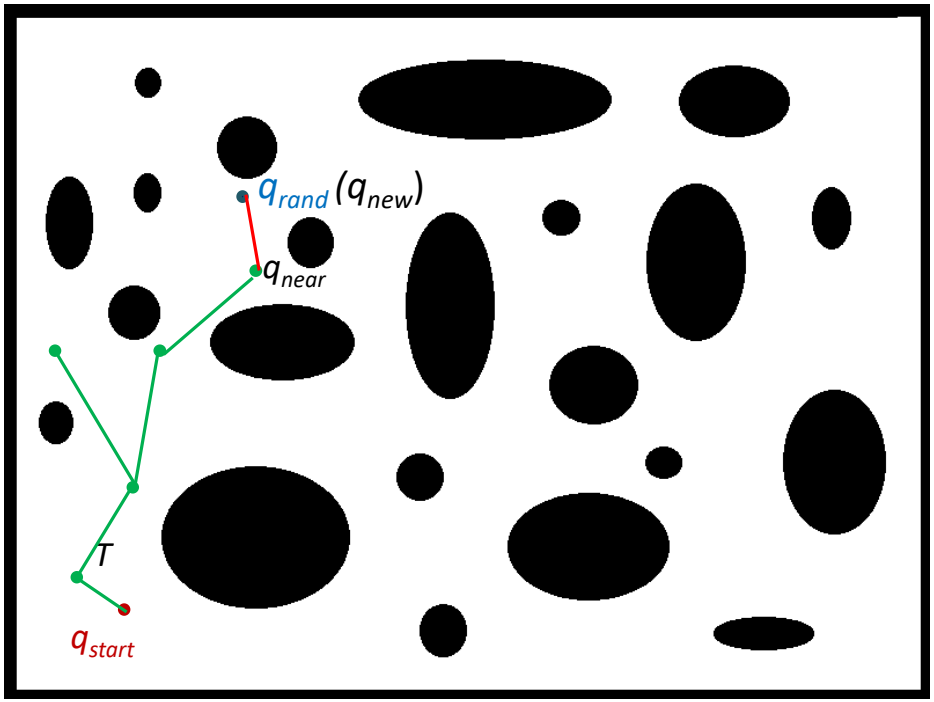
- L4: $q_{near} \leftarrow$ NEAREST_VERTEX(q_{rand}, T); find the nearest configuration
  - Define the proper distance



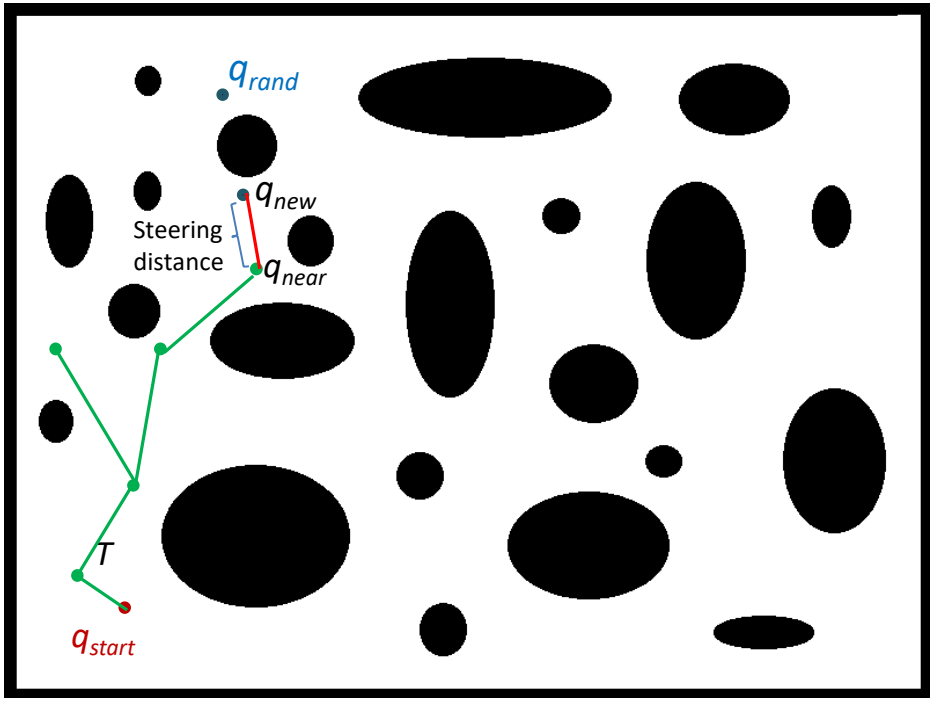
C-space

Rapidly exploring Random Trees (RRT)

- L5: $q_{new} \leftarrow$ NEW_CONF(q_{near}, q_{rand}); generate a new configuration
- L6: T.add_vertex(q_{new}); T.add_edge(q_{near}, q_{new}); add the new configuration



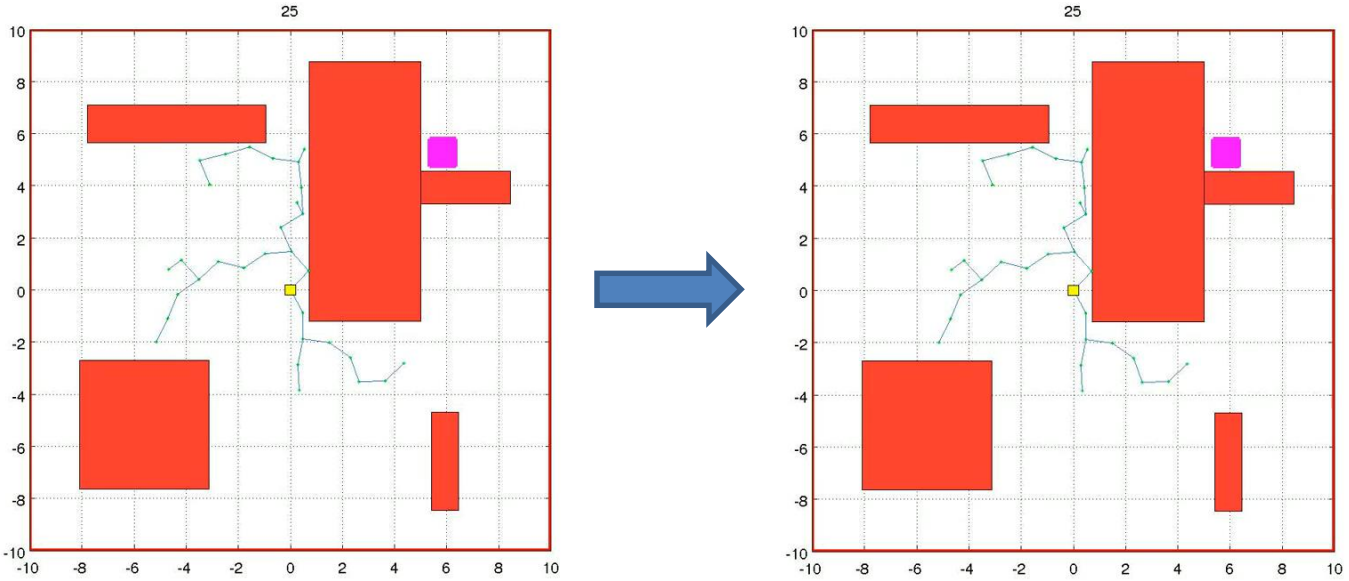
C-space



C-space

RRT*

- Basic Idea
  - RRT is simple, but is prone to be probabilistic incomplete.
  - Add **rewire** function: swap new point in as parent for nearby vertices who can be reached along shorter path through new point than through their original (current) path
  - RRT* is asymptotically optimal.



[Karaman, Sertac, and Emilio Frazzoli. "Sampling-based algorithms for optimal motion planning." *The international journal of robotics research* 30.7 (2011): 846-894.]

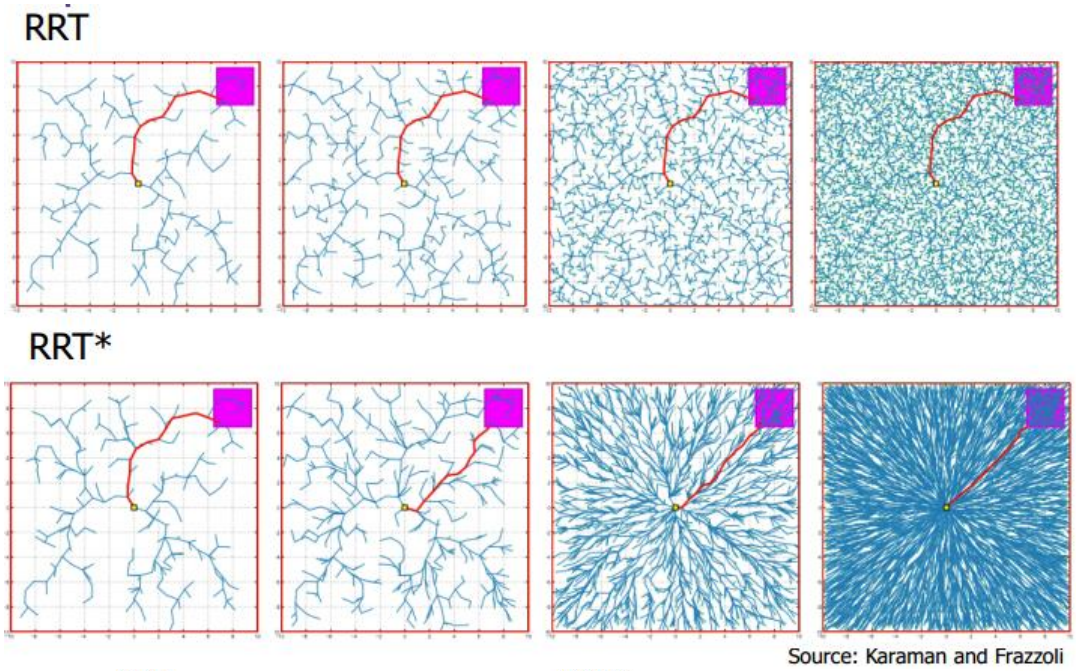
RRT*

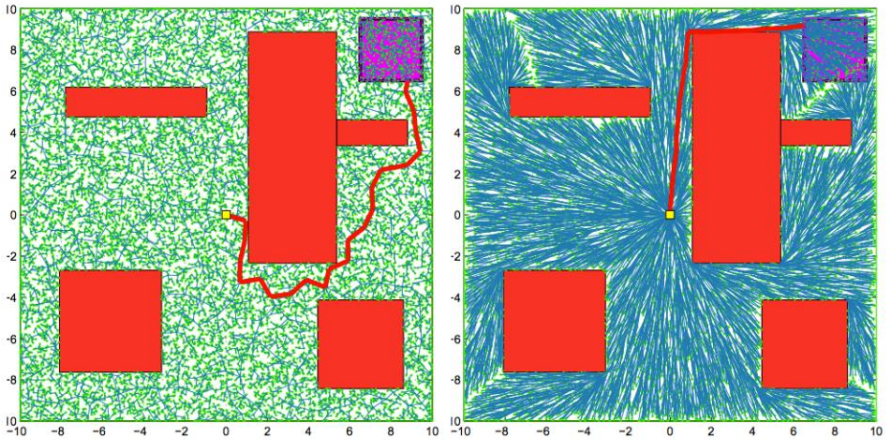
- Algorithm

Algorithm 6: RRT*

$V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
for $i = 1, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow \texttt{SampleFree}_i$;
  $x_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
  $x_{\text{new}} \leftarrow \texttt{Steer}(x_{\text{nearest}}, x_{\text{rand}})$;
  if $\texttt{ObtacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
    $X_{\text{near}} \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $x_{\min} \leftarrow x_{\text{nearest}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{nearest}}) + c(\texttt{Line}(x_{\text{nearest}}, x_{\text{new}}))$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Connect along a minimum-cost path
      if $\texttt{CollisionFree}(x_{\text{near}}, x_{\text{new}}) \wedge \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}})) < c_{\min}$ **then**
        $x_{\min} \leftarrow x_{\text{near}}$; $c_{\min} \leftarrow \texttt{Cost}(x_{\text{near}}) + c(\texttt{Line}(x_{\text{near}}, x_{\text{new}}))$
    $E \leftarrow E \cup \{(x_{\min}, x_{\text{new}})\}$;
    foreach $x_{\text{near}} \in X_{\text{near}}$ **do** // Rewire the tree
      if $\texttt{CollisionFree}(x_{\text{new}}, x_{\text{near}}) \wedge \texttt{Cost}(x_{\text{new}}) + c(\texttt{Line}(x_{\text{new}}, x_{\text{near}})) < \texttt{Cost}(x_{\text{near}})$ **then** $x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
      $E \leftarrow (E \setminus \{(x_{\text{parent}}, x_{\text{near}})\}) \cup \{(x_{\text{new}}, x_{\text{near}})\}$
return $G = (V, E)$;

RRT* vs RRT





Logistics

- Project 1, phase 1 this Friday (03/03)
- Project 1, phase 2 is released (02/28)
  - Due next Friday: 03/10