

MICRO-502_Aerial_Robotics_Notes

Intro (week1)

Overview

Fixed wing Staying in the Air

Maintaining constant speed V

Major Application Fields

For Agriculture

For Energy

For Public safety & security

For Delivery

Category

«Can Drones Deliver?»

Last-cm delivery - Dronistics

Checkpoints

Multicopters (week1)

Introduction

Rotorcrafts (helicopters vs multicopters)

Pros and Cons

Pros-Easy to build and maintain

Cons

Structure and Physics

Main components

Configuration

Rotation speeds / Forces / Moments

Hover conditions

Flight mechanics

Moving Up and Down

Rotating in Yaw

Rotation in Roll/Pitch

Summary of equations

Example-Translated flight

Types of Multicopters

Configuration

Fully Actuated Multicopters

Features

Pros

Cons

Energetics

- Energy in hovering
- Energy in forward flight of quadcopter
- Increase flight time

Checkpoints

Attitude representations (week2)

- 3D Attitude representation- Euler Angles

- Quaternion

- Conversion

- Complex examples

Checkpoints

Control (week2&3)

- Cascaded control Architecture

- Control allocation

- Method

- Remark

- Rate control

- Attitude control

- Full quaternion based attitude control for a quadrotor

- Control strategies for multicopters

- Linear

- Examples

- Features

- PID controller

- LQR/LQG

- Nonlinear

- Examples

- Features

- An introduction to fully actuated multirotor UAVs

Checkpoints

State Estimation (week3&4)

- Introduction to State Estimation

- Why State Estimation in Robotics?

- Estimation for deterministic systems

- State Observer (or Luenberger Observer)

- Dynamical Systems

- State Estimation for stochastic systems

- Recap of fundamental concepts

- Kalman filter (KF)

- Intro to EKF, UKF and particle filters

- State Estimation in aerial robotics

Checkpoints

 Exercise summary

Navigation Methods (week5)

Perception (week5)

Fixed-wing drones (week6)

Aerial Swarms (week7)

Intro

Reynolds flocking algorithm (Reynolds, 1987)

 Reynolds flocking: model

 Reynolds flocking with migration

Case: Aerial swarms for disaster mitigation

 Communication radius and turning angle

 Virtual agents for flocking with fixed-wing drones

Reynolds flocking with obstacles (Virtual agents)

Other models

 Vicsek model: particles in confined environments (密闭环境)

 Olfati-Saber model

Drone Swarms

Visual information in flocking

 Soria2019IRC-influence of limited visual sensing using Reynolds

 Schilling2019RAL-Learning to flock in simulation with vision

 Schilling2021RAL-Learning to flock outdoor with vision

Check points

Flapping-Wing (week8)

Drone Regulations (week8)

Checkpoints

UAS Hardware (week9)

Introduction

Frame and materials

 materials comparison

 metric when considering materials

Energy sources

 Category

 Energy and power density

 Li-Po batteries

 Discharge Curves of Li-Po battery

 Energy Curve of Li-Po battery

Actuators

 Actuators for propulsion

 Electric motor example-Brushless DC electric motors

Actuators for control/maneuvering

Servomotors

Examples of Servomotors

Propellers

Characteristics

Pitch and efficiency at different cruise speed

Choose the right combination actuator and propeller

Sensors

Gyroscopes

Accelerometers

Magnetometers

Pressure / Altitude sensors

Airspeed sensors

Global positioning system (GPS)

Power sensors

Optic flow cameras

Autopilots

Communication protocols

Checkpoints

Insect-inspired vision (week10)

Optical flow

For pure translational motion

Sensors used for flight control

Architecture of insect eyes and brains

Elementary Motion Detector 初级运动检测器

Experiment - Optomotor Response 视运动反应

Wide-field, motion-specific neurons

Optic Flow Computation

Gradient Descent Methods

Image Interpolation Algorithm –I2A

Obstacle avoidance with I2A

Checkpoints

Adaptive Morphology in Flying Animals and Drones (week10)

Bioinspired Mechanical Resilience

How do insects cope with collisions?

The Size Problem

Self-deployable origami drone

Origami Drone Wing

Adaptive Morphology

Checkpoints

Agile Flight (week11)

Autonomous drone racing

Drone acrobatics

Low-latency sensing

Learning of flight controllers (week11)

challenge1: Architecture/Input and Output representation

challenge2: Data collection

Limitations of Imitation Learning -> RL

RL limitations

challenge3: Guarantee the platform's safety during training and testing

Takeaways

MICRO-502_Aerial_Robotics_Notes

Lecture notes by Yujie He

Last updated on 2021/05/04

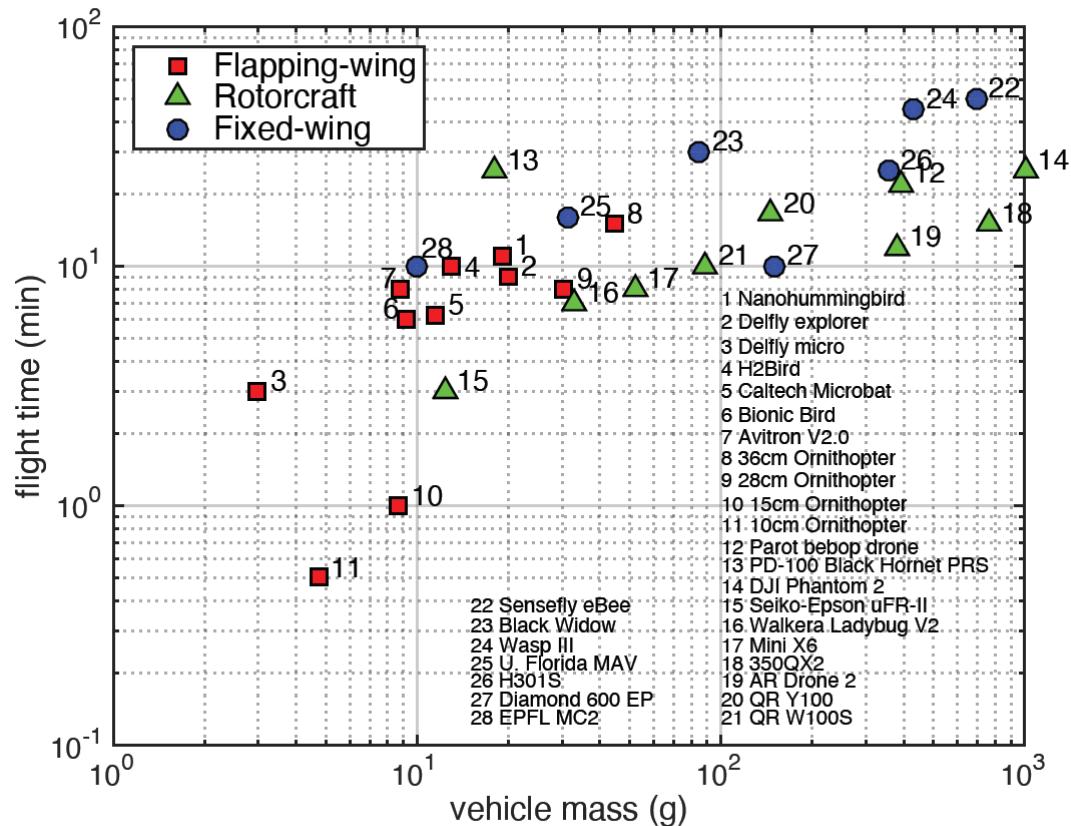
Intro (week1)

Overview

- Most Civilian Drones are Small
 - flapping wings: operates in small scale; short flight time
 - rotorcraft/multicopters: in medium; could hover in place
 - fixed wings: fixed has longest flight time; cannot hover in place
- but endurance is a challenge

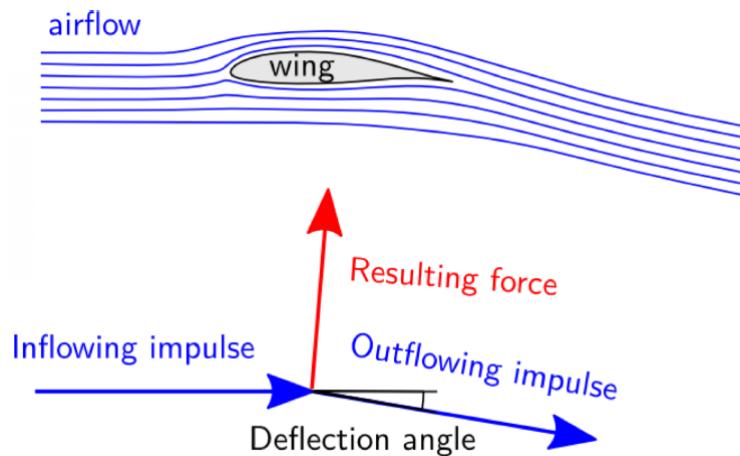
Flight endurance is a challenge

Floreano & Wood,
Nature, 2015



Fixed wing Staying in the Air

generate a force **Lift** L equal and **opposite** to its own **weight** W



fixed-wing generates lift by airflow

- computing lift: $W = \frac{1}{2} C_l * \rho * V^2 * S \approx 0.3 * 1.25 * V^2 * S$ (sea level)

- lift coefficient C^l : proportional to angle of attack
at cruise speed 6° , assuming $\frac{1}{2}C^l = 0.2$
- Air density: decreases with the altitude
 1.25 kg/m^3 at sea level
- Air speed V: fly 2x as fast $\rightarrow 4x$ lift
- Wing area S
- compute the required **velocity** $V = \sqrt{(W/0.38 * S)}$
- Speed is proportional only to **wing loading** W/S $W/S = 0.38V^2$

Maintaining constant speed V

- generate a **thrust** force T equal and opposite to **Drag** force D
- Drag $D = \frac{1}{2}C^d * \rho * V^2 * S = \text{Lift}/r$
lift-drag ratio $r = \text{Lift}/\text{Drag}$

Major Application Fields

- Agriculture
- Energy
- Public safety & security
- Delivery

For Agriculture

- fixed-wing
inspection: large fields; few lights/value
- quadcopter
spraying: small fields; high-value crops; difficult terrain

For Energy

- stationary inspection
fire, related to human safety

- long-range inspection
frequent, faster inspection to powerline, gasoline
- Power generation
strong & steady winds

For Public safety & security

- in-vehicle
policeman
- long-range
Border patrol, fast intervention

For Delivery

Category

- Long-range: fixed wing; combination of wing and rotors to cover long distance
- Short-range: multicopter

Forecast: parcel delivery > air freight in near future

«Can Drones Deliver?»

- Power requirement: 0.59kW
deliver 2 kg payload at cruising speed of 45 km/h
- Energy requirement: about 0.39kWh
deliver 2 kg payload within 10 km radius with 30 km/h headwind
 - power (kW) x distance to speed ratio (d/v) to get energy requirement (kWh)
- Battery & Platform: choose considerations -> cost; weight; energy density; lifetime
- Economics: Electricity/Battery cost per km

Last-cm delivery - Dronistics

- Packdrone + SimplyFly
Protective foldable cage; Redundant GPS
- Temperature-control box

Checkpoints

- For a given total mass, what type of small drones (multi-copter, fixed-wing, flapping wing) displays the longest endurance?

fixed-wing

fixed-wing drones generate lift with their wings. This means that, unlike a multirotor drone, they don't expend large amounts of energy just to stay in the air and fly more efficiently as a result

- How much faster must an intercontinental airplane fly at cruising altitude compared to sea level?

$$W = \frac{1}{2} C^l * \rho * V^2 * S \rightarrow V = \sqrt{\frac{2W}{C^l \rho S}}$$

- sea level: $\rho = 1.25 \text{ kg/m}^3$
- intercontinental airplane fly $\sim 10000 \text{ m}$: $\rho = 0.4135 \text{ kg/m}^3$

- What structural factor (i.e. not the engine) affects the cruising speed of fixed-wing drones?

Speed is only proportional to wing loading (W/S) = Weight/Wing area

- What are the two major drone applications in agriculture?

Inspection and spraying

- What are the drone applications in the energy sector?

Stationary inspection; long-range inspection; and power generation

- What are the factors to consider for calculating the cost/km of drone delivery?

Electricity cost and battery cost

Multicopters (week1)

	Fixed wing	Flapping wing	rotating wing
Examples	airplanes, gliders	new robots	helicopters, multicopters
Pros	Fast; Efficient	Efficient	Can hover; Highly maneuverable
Cons	Cannot Hover	Hard to build and control	Less efficient

	Fixed wing	Flapping wing	rotating wing
Features		Scale down in size	Vertical take-off and landing (VTOL)

Introduction

Rotorcrafts (helicopters vs multicopters)

generates lift using high speed rotary blades called rotors

- Features
 - Vertical take-off and landing (VTOL)
 - Very maneuverable
 - Less efficient than fixed wing vehicle
- helicopters-Need complex variable pitch rotors
 - the tail produce a moment to counteract the force generated by main propeller blade when generating main lift
 - change the pitch of the blades (force vector) to produce translation-add complexity
- multicopters-Use multiple fixed-pitch blades
 - each spin in different directions; don't need tails due to balanced moment
 - fix pitch propellers

overtaken by helicopters due to heavy workload of the pilot

Pros and Cons

Pros-Easy to build and maintain

- Mechanically simple
- Does not require any complex mechanical parts
- Can move around by changing motor speed
- Can hover, takeoff, and land vertically

Cons

- Required energy constantly to hover
- less efficient than helicopters of the same size because the **thrust is generated by smaller propellers.**

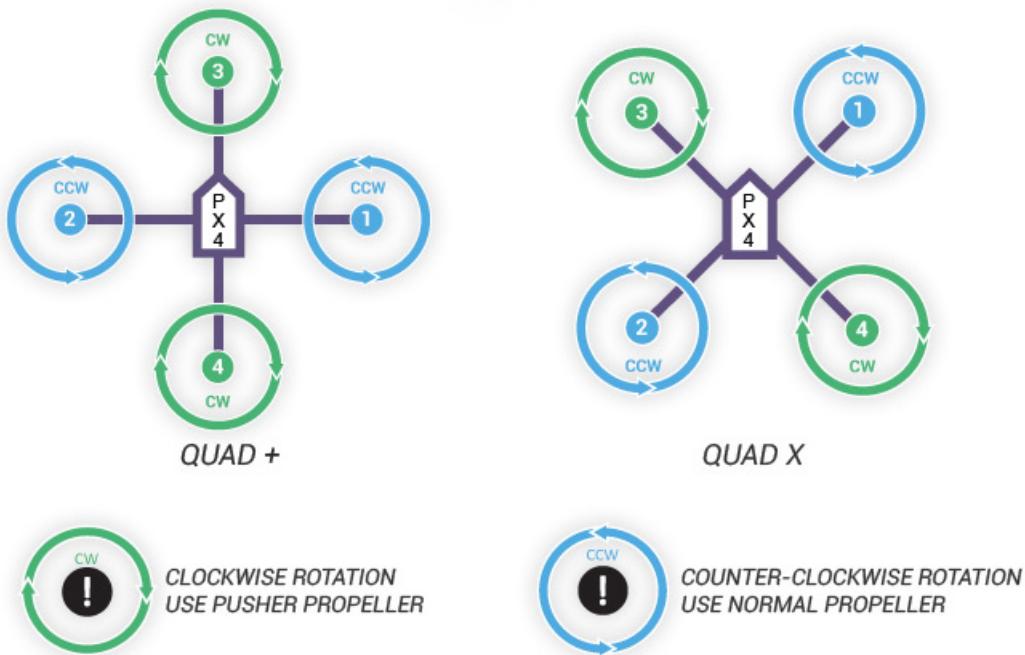
Structure and Physics

Main components

frame; control board; Motors and motor drivers (ESC, electronic speed controller); Propellers; Battery; Receiver

Configuration

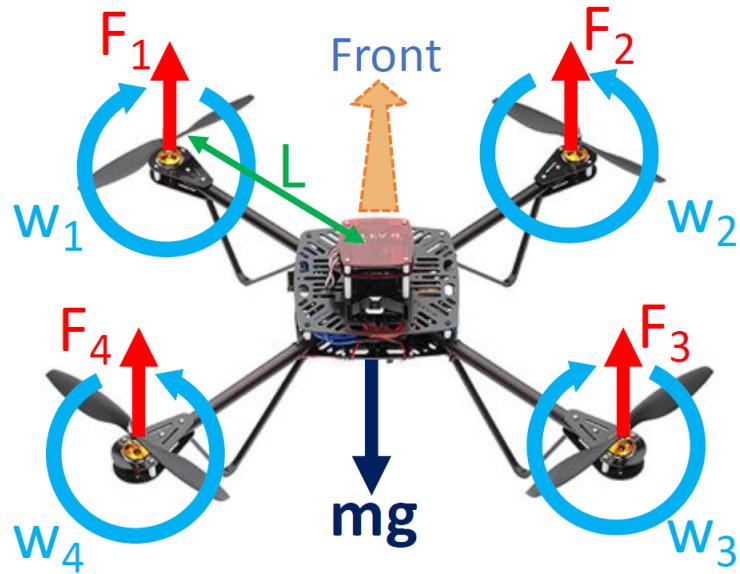
- Four propellers generate four lift forces
- Propellers 1& 2 (CCW) have opposite pitch compared to propellers 3 & 4 (CW)



Rotation speeds / Forces / Moments

movement are controlled by changing the rotation speed of the propellers

- Force F is **proportional to square of propeller speed** $F_i \propto w_i^2$
- mg is the **weight** of the quadrotor
- **Moments** generated by the forces are $M_i = L \propto F_i$



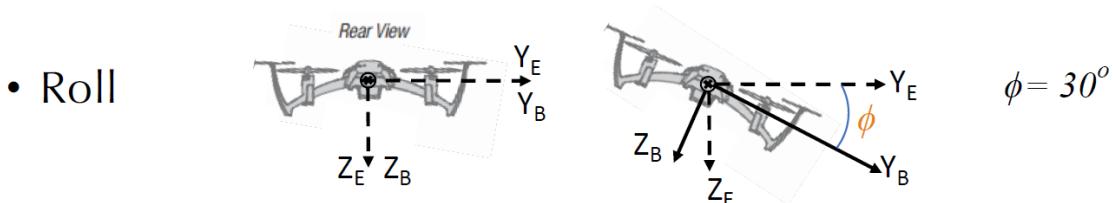
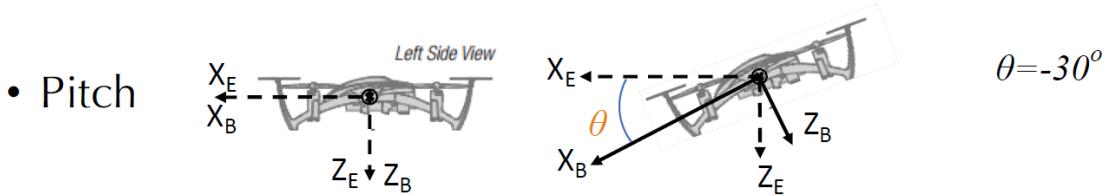
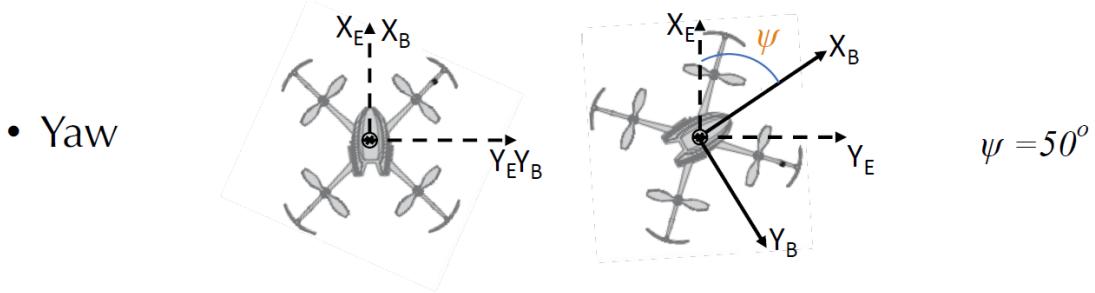
Hover conditions

1. All forces must be balanced $F_1 + F_2 + F_3 + F_4 + mg = 0$
move up and down
2. Lift forces must be parallel to gravity $F_i \parallel g$
3. All moments must be balanced $M_1 + M_2 + M_3 + M_4 = 0$
pitch and roll
4. Rotor speeds must be balanced (torque balanced) $(w_1 + w_3) - (w_2 + w_4) = 0$
yaw

Flight mechanics

| *How to move a quadrotor around?*

- Orientation



Violating one or more of these conditions implies that the quadcopter starts to move

Moving Up and Down

- condition1: Forces Not balanced $F_1 + F_2 + F_3 + F_4 + mg \neq 0$

Rotating in Yaw



- Rotor speeds not balanced (torque balanced) $(w_1 + w_3) - (w_2 + w_4) \neq 0$
- $$\dot{\psi} = k_\psi ((w_2 + w_4) - (w_1 + w_3))$$
- Note: opposite motor pair should increase/decrease motor speeds to keep hovering, or it will keep flying up!

Rotation in Roll/Pitch

- Forces not parallel to gravity $F_i \nparallel mg$

- moments not balanced $M_1 + M_2 + M_3 + M_4 \neq 0$

$$\dot{\phi} = k_\phi ((w_1 + w_4) - (w_2 + w_3))$$

$$\dot{\theta} = k_\theta ((w_1 + w_2) - (w_3 + w_4))$$

Summary of equations

- Assumption: $k = k_\phi = k_\theta = k_\psi$

$$\dot{\phi} = k ((w_1 + w_4) - (w_2 + w_3)) = k w_1 - k w_2 - k w_3 + k w_4$$

$$\dot{\theta} = k ((w_1 + w_2) - (w_3 + w_4)) = k w_1 + k w_2 - k w_3 - k w_4$$

$$\dot{\psi} = k ((w_2 + w_4) - (w_1 + w_3)) = -k w_1 + k w_2 - k w_3 + k w_4$$

$$T = k ((w_1 + w_3 + w_2 + w_4)) = k w_1 + k w_2 + k w_3 + k w_4$$

Can also be represented by matrices:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \\ T \end{pmatrix} = \begin{pmatrix} k & -k & -k & k \\ k & k & -k & -k \\ -k & k & -k & k \\ k & k & k & k \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix} = \mathbf{K} \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{pmatrix}$$

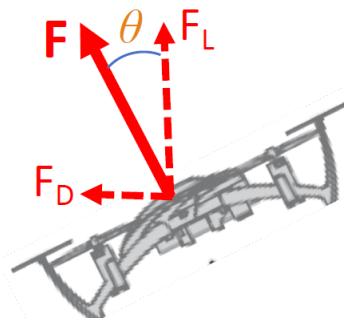
- These equations give the angular rates/thrust given the motor speeds.
- to control the quadrotor state -> setting the rotor speeds for obtaining a desired angular rotation
- **inverse** operation

Example-Translated flight

Moving Forward

Translated flight requires more thrust than hovering, but not always more power (see section 6)!

- pitch down: decrease F_1 and F_2
- hover: increase F_1 and F_2 to stop rotation
- translate: keep balance between mg and thrust $\mathbf{F} \cos \theta = -mg$

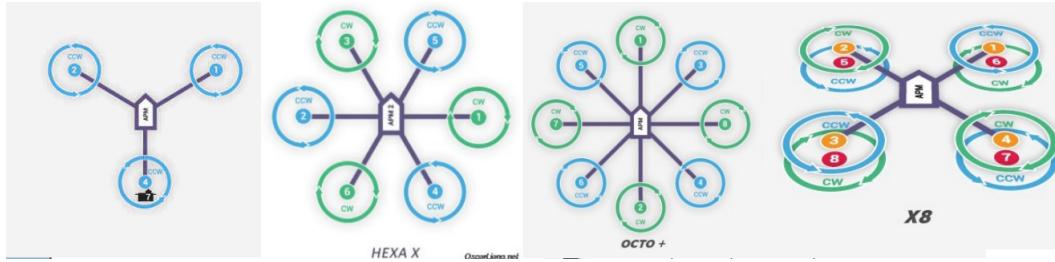


- pitch back

Types of Multicopters

main feature; fully actuated multicopters

Configuration



- Tricopter
 - More yaw authority compared to quadcopters
 - direct control of yaw
 - More complex mechanical design due to the **servo** in tail
 - need additional servo to roll/pitch the back motor
- Hexacopter/Octacopter
 - More lifting capacity -> more payload
 - **Redundancy**, so robust to failure
 - Larger size; expensive
- X8 configuration
 - More lifting capacity
 - **More efficiency** thanks to the **coaxial configurations**

Fully Actuated Multicopters

underactuated multicopters -> all propellers are rotated in the same plane

Features

- Rotors disks are in different planes
- Fully Actuated to control 6 DoF (heave, roll, pitch and yaw, x and y translation) by using 6 motors

Pros

- Translational and rotational **dynamics** are **decoupled**
- **improved robustness to disturbances** and to perform complex manipulation tasks
- Possibility to plan **more complex trajectories**

Cons

- Decreased energetic efficiency

Energetics

What is the power consumption of a multicopter during flight?

How to extend multicopters flight time?

- Multicopters have high power requirements
inefficient compared to fixed-wing or flapping-wing aircraft
 - consume about 200 W/kg on average
 - Centimeter scale quad with LiPo -> 5-7min
 - Decimeter scale quad with LiPo ->20-30min
- changes corresponding to **weather conditions** and **aggressiveness of flight**

Energy in hovering

$$P_r = \frac{(T_r)^{3/2}}{\eta_r r_p \sqrt{2\rho_a \pi}}$$

$P_r \rightarrow$ Rotor power
 $T_r \rightarrow$ Rotor Thrust
 $\eta_r \rightarrow$ Figure of Merit (propeller efficiency)
 $r_p \rightarrow$ Propeller radius
 $\rho_a \rightarrow$ Air density

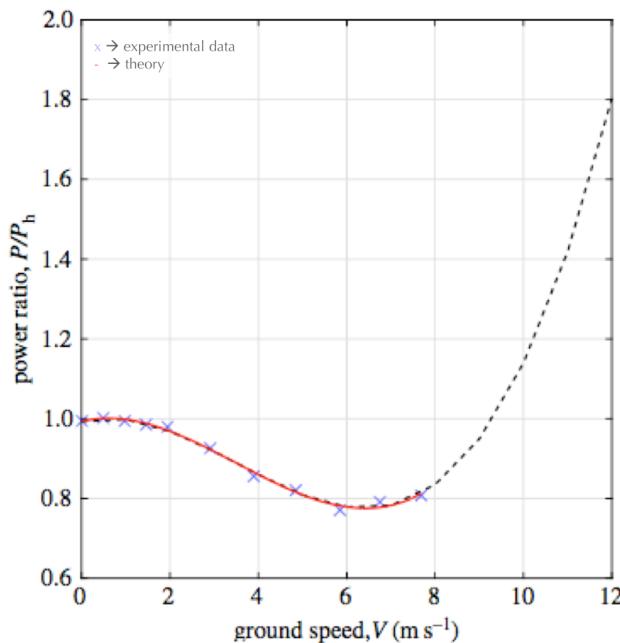
$$t_e = \frac{E_b m_b}{n_p P_r + P_i}$$

$t_e \rightarrow$ Time of flight
 $E_b \rightarrow$ Battery specific density (Wh/Kg)
 $m_b \rightarrow$ Mass of the battery
 $n_p \rightarrow$ Number of rotors
 $P_i \rightarrow$ Power required by other components (avionics and sensors)

- Power calculation of single motor when drone in hovering
 - Propeller efficiency ranges from 0.85 to about 0.9
 - $r_p \sqrt{\pi}$ is the square root of the disk area
- Flight time
 - P_i has high variability

Energy in forward flight of quadcopter

- Power consumption during forward flight normalized by the power consumption at hover



- lower power need but higher thrust -> aerodynamics

tilting the quad to the side -> artificially increase the angle of attack -> generate the increase in thrust in the given power

- high power ratio again
drag increases quickly again!

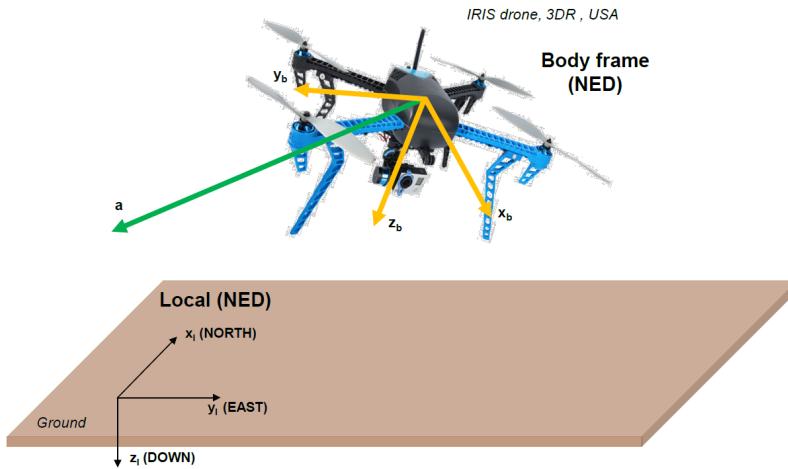
Increase flight time

1. Weight and drag reduction
2. Increase the specific power of the energy source
 - switch from LiPo batteries to gasoline (far more energy stored)
3. Docking station for charging/battery swapping
4. Tether for power supply
 - reduce battery
5. Improving efficiency via mechanical
6. Energy aware motion planning
 - reduce acceleration (aggressive flight)
7. Multi-modal operation
 - perching; walking and rolling

Checkpoints

- What set of conditions corresponds to hovering in a quadcopter?
Four conditions: balanced weight; parallel to weight; balanced moment (thrust x half frame); balanced rotation speed
- What set of conditions corresponds to a rotation around the yaw axis in a quadcopter?
imbalanced rotation speed
- How the time of flight of multicopters can be increased?
- How the power consumption and total thrust evolve at different forward speeds in multicopters?
first goes down as thus increases when angle of attack increases
then goes down because the drag force increases again

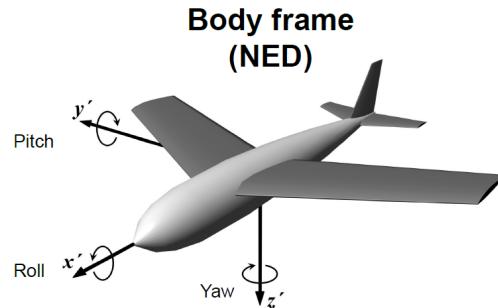
Attitude representations (week2)



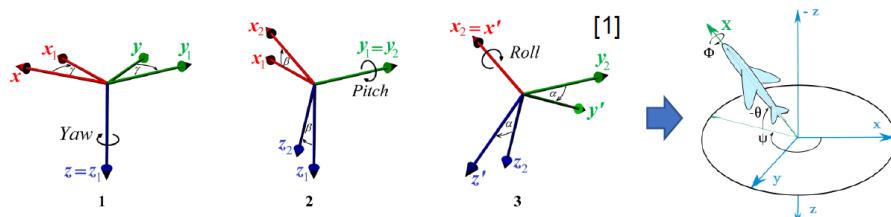
- how to convert state variables from body frame to earth frame

3D Attitude representation- Euler Angles

- Rotation matrices can be parametrized by Euler Angles
 - Roll ϕ : rotation around x
 - Pitch θ : rotation around y
 - Yaw ψ : rotation around z



- Yaw-Pitch-Roll rotation matrix (Z,Y,X) or Heading-Pitch-Roll (h-p-r)



$$\begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{R}_\psi \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} \cos \theta & 0 & -\sin \theta \\ 0 & 1 & 0 \\ \sin \theta & 0 & \cos \theta \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} = \mathbf{R}_\theta \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & \sin \phi \\ 0 & -\sin \phi & \cos \phi \end{pmatrix} \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \mathbf{R}_\phi \begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix}$$

can be summarized as $R = R_\phi R_\theta R_\psi$ (左乘)

- Claw Example

rotation + translation

补充示例！！！

- Issues: gimbal lock problem

- sensitive to singularities-when pitch angle is 90 degrees, **roll and yaw rotations give the same sensor readings**

lost 1 DoF

$$R = R_\phi R_{\frac{\pi}{2}} R_\psi = \begin{bmatrix} 0 & 0 & -1 \\ \sin \phi \cos \psi - \cos \phi \sin \psi & \sin \phi \sin \psi + \cos \phi \cos \psi & 0 \\ \cos \phi \cos \psi + \sin \phi \sin \psi & \cos \phi \sin \psi - \sin \phi \cos \psi & 0 \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 0 & -1 \\ \sin(\phi - \psi) & \cos(\phi - \psi) & 0 \\ \cos(\phi - \psi) & -\sin(\phi - \psi) & 0 \end{bmatrix}$$

- **fail to produce reliable estimates** when the pitch angle approaches 90 degrees.
 - can only be solved by **switching to a different representation** methods, for example **quaternions**

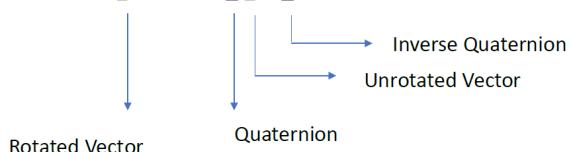
Quaternion

the union of scalar and vector

The diagram illustrates the components of a vector $q = [s, \mathbf{v}]$. The scalar component s is represented by a blue horizontal bar with red endpoints, labeled "Scalar". The vector component \mathbf{v} is represented by a red horizontal bar with blue endpoints, labeled "Vector".

- rotation vector = $\text{quat} * \text{vector} * \text{quat}^{-1}$

$$p' = qpq^{-1}$$



- $p = [0, \mathbf{p}]$ vector in 3-space
- $q = [s, \lambda\hat{\mathbf{v}}]$ Must meet these requirements

$$|\hat{\mathbf{v}}| = 1; s^2 + \lambda^2 = 1$$

- multiplication

Rotated vector

$$p' = qpq^{-1} = [0, (1 - \cos(\theta)(\hat{\mathbf{v}} \cdot \mathbf{p})\hat{\mathbf{v}} + \cos(\theta)\mathbf{p} + \sin(\theta)\hat{\mathbf{v}} \times \mathbf{p})]$$
 ↓
 Note that the scalar here is 0
 The angle to be rotated

$$q = [\cos \frac{1}{2}\theta, \sin \frac{1}{2}\theta\hat{\mathbf{v}}] \text{ works for relative to x axis } (\theta)$$

week2_quaternion_mat.png

$$qpq^{-1} = [0, (1 - \cos \theta)(\hat{\mathbf{v}} \cdot \mathbf{p})\hat{\mathbf{v}} + \cos \theta \mathbf{p} + \sin \theta \hat{\mathbf{v}} \times \mathbf{p}]$$

$$\mathbf{p}' = \begin{bmatrix} 2(s^2 + x^2) - 1 & 2(xy - sz) & 2(xz + sy) \\ 2(xy + sz) & 2(s^2 + y^2) - 1 & 2(yz - sx) \\ 2(xz - sy) & 2(yz + sx) & 2(s^2 + z^2) - 1 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix}$$

$$\begin{aligned} q &= [s, \mathbf{v}] & p &= [0, \mathbf{p}] \\ &= [s, x\mathbf{i} + y\mathbf{j} + z\mathbf{k}] & &= [0, x_p\mathbf{i} + y_p\mathbf{j} + z_p\mathbf{k}] \end{aligned}$$

$$q = \left[\cos \frac{1}{2}\theta, \sin \frac{1}{2}\theta\hat{\mathbf{v}} \right]$$

$$\text{For multiple rotations: } q_2(q_1pq_1^{-1})q_2^{-1} = (q_2q_1)p(q_2q_1)^{-1}$$

From the book "Quaternions for Computer Graphics" by John Vince.

Claw Example

rotation + translation using quaternions

Conversion

- Euler angles to quaternion
- Quaternion to Euler angles (arctan2 is recommended)

Complex examples

frame: image/centered -> camera -> gimbal -> body -> local

$$v_l = p + R_p^l v_p = p + R_b^l R_g^b R_c^g R_i^c R_p^i v_p$$

注意谁相对谁！！！

Checkpoints

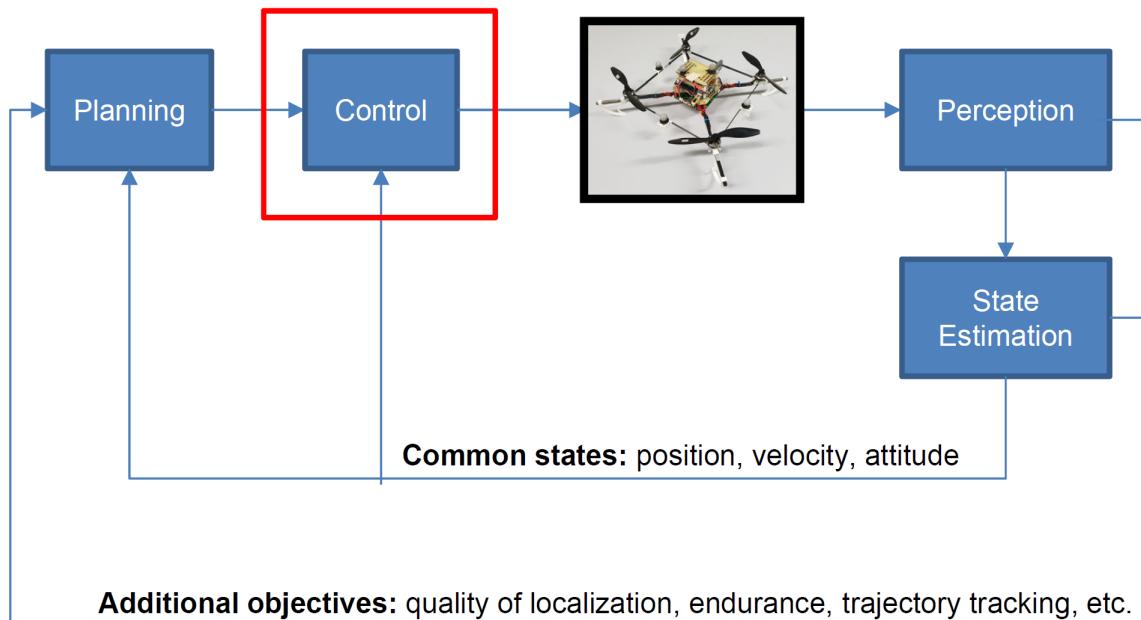
- nothing left in this course

Control (week2&3)

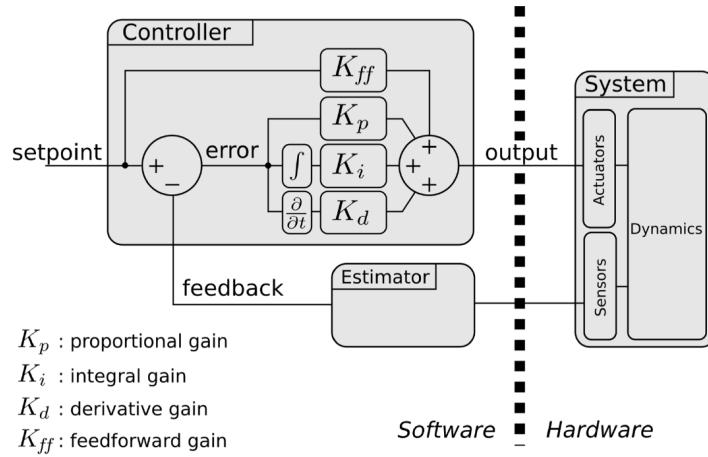
结合exercise进行再次查看

Cascaded control Architecture

- drone architecture



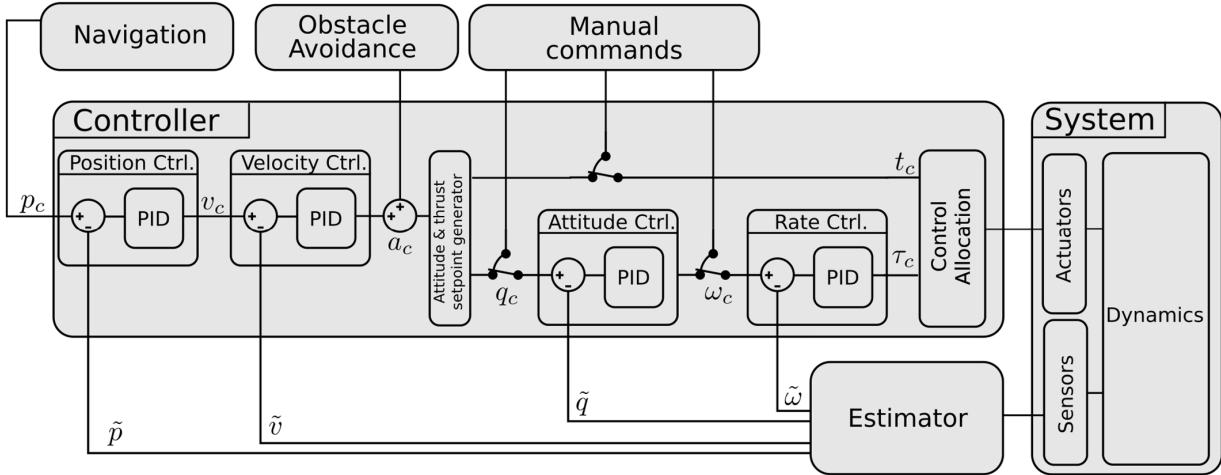
- communication: glue to connect different parts (FCU to onboard computer to ground station)
- PID controller



- Open-loop
- Feedforward control
 - responds to its control signal in a **pre-defined way**
 - based on a **previous knowledge** of the system
- Feedback control
- control example

Examples	Free-flight glider	Passively stable helicopter	Racing drone	Autonomous Delivery drone
Sensors	None	None	IMU	IMU + GPS + Vision + Mag
Controllers	None	None	(Attitude), Rate	Position, Velocity, Attitude, Rate
Setpoint	None	Manual actuator setpoint	Manual rate setpoint	Autonomous navigation/Manual

- challenges
 - underactuated system for quadcopter
 - approximate aerodynamic model
 - control inputs are idealized (the **real model of the motors** can be quite hard to model)

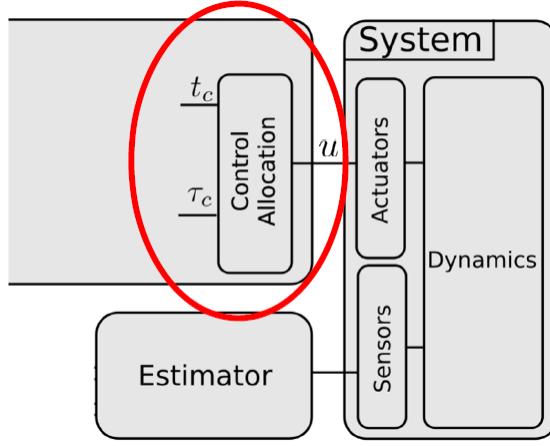


- features
 - insert the obstacle avoidance together with velocity setpoint
 - manual commands send thrust and attitude rate commands
 - **the lowest level of the controller, the higher bandwidth it needs**
- Pros
 - Decouple translational and rotational dynamics
 - Facilitate implementation of the controller
implement one by one
 - Better failure diagnosis capability
modular control architecture for debugging them one by one
- Question: why a more advanced controller?
 - linear assumption vs nonlinear system
 - more advanced system can bring optimal performance

Control allocation

*convert **thrust & torque** setpoint into **actuator commands***

account for different geometries



- from **thrust & torque setpoint** into **actuator**

$$\begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} = B \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ t_z \end{bmatrix} \quad u : \text{actuator commands (Nx1)}$$

\mathcal{T} : moment setpoints (3x1)

t : thrust setpoint (1×1)

Method

1. Actuator torque and force:

$$\vec{t}_i = C_t \rho D^2 \omega^2 \vec{\nu}_i \quad \vec{\tau}_i = \vec{\tau}_i^{\text{thrust}} \pm \vec{\tau}_i^{\text{drag}}$$

$$\vec{t}_i \approx (k_t \vec{\nu}_i) u_i \quad \vec{\tau}_i \approx (\vec{p}_i - \vec{p}_{cg}) \times \vec{t}_i \pm (k_\tau \vec{\nu}_i) u_i$$

2. Actuator effectiveness matrix:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ t_z \end{bmatrix} = A \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \quad A = \begin{bmatrix} \begin{bmatrix} \vec{\tau}_0 \end{bmatrix} & \dots & \begin{bmatrix} \vec{\tau}_i \end{bmatrix} & \dots & \begin{bmatrix} \vec{\tau}_{N-1} \end{bmatrix} \\ \begin{bmatrix} t_0 \end{bmatrix} & & \begin{bmatrix} t_i \end{bmatrix} & & \begin{bmatrix} t_{N-1} \end{bmatrix} \end{bmatrix}$$

3. Control allocation matrix:

$$\begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} = B \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ t_z \end{bmatrix} \quad B = \begin{bmatrix} B_0^{\tau_x} & B_0^{\tau_y} & B_0^{\tau_z} & B_0^{t_z} \\ \vdots & \vdots & \vdots & \vdots \\ B_{N-1}^{\tau_x} & B_{N-1}^{\tau_y} & B_{N-1}^{\tau_z} & B_{N-1}^{t_z} \end{bmatrix}$$

- Compute generated force and torque of each actuator
- Build matrix "Actuator Effectiveness"
- Compute **allocation matrix B** as the **pseudo-inverse** of A
not always square, so using pseudo-inverse ($B = A^+$)

Remark

- less effective at producing yaw torque
- yaw torque requires more actuator effort than roll and pitch torque

$$A = \begin{bmatrix} \text{Rotor 1} & \text{Rotor 2} & \text{Rotor 3} & \text{Rotor 4} \\ -0.71 & 0.71 & 0.71 & -0.71 \\ 0.71 & -0.71 & 0.71 & -0.71 \\ 0.05 & 0.05 & -0.05 & -0.05 \\ -1. & -1. & -1. & -1. \end{bmatrix} \begin{array}{l} \text{Roll moment} \\ \text{Pitch moment} \\ \text{Yaw moment} \\ \text{Thrust} \end{array}$$

$$B = \begin{bmatrix} -0.35 & 0.35 & 5. & -0.25 \\ 0.35 & -0.35 & 5. & -0.25 \\ 0.35 & 0.35 & -5. & -0.25 \\ -0.35 & -0.35 & -5. & -0.25 \end{bmatrix} \begin{array}{l} \text{Rotor 1} \\ \text{Rotor 2} \\ \text{Rotor 3} \\ \text{Rotor 4} \end{array}$$

Rate control

Input body rate setpoint -> **Output** torque to each angle rate

Attitude control

Input quaternions setpoint -> **Output** torque to each angle rate

- How to compute the quaternion error

$$\mathbf{q}_{err} = \mathbf{q}_{ref} \otimes \mathbf{q}_m^*$$

- axis error

$$\begin{bmatrix} \Phi_e \\ \Theta_e \\ \Psi_e \end{bmatrix} \approx \text{sgn}(q_e^0) \begin{bmatrix} 2q_e^1 \\ 2q_e^2 \\ 2q_e^3 \end{bmatrix}$$

Full quaternion based attitude control for a quadrotor

Fresk, E. and Nikolakopoulos, G., 2013, July. Full quaternion based attitude control for a quadrotor. In 2013 European control conference (ECC) (pp. 3864- 3869). IEEE.

- without any transformations and calculations in the Euler's angle space
- quaternions

- scalar + vector part

$$\mathbf{q} = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k} = [q_0, q_1, q_2, q_3]^T$$

- Multiplication of two quaternions p, q is being performed by the **Kronecker product**

- $p \otimes q$ represents the combined rotation
- quaternion multiplication is non-commutative 不遵循交换律
- All quaternions are assumed to be of **unitary length** 单位长度
- The **complex conjugate** of a quaternion has the same definition as normal complex numbers 复共轭

$$\text{Conj}(\mathbf{q}) = \mathbf{q}^* = [q_0 \quad -q_1 \quad -q_2 \quad -q_3]^T$$

- if the length of the quaternion is unitary then the **inverse is the same as its conjugate** 逆即是复共轭

$$\text{Inv}(\mathbf{q}) = \mathbf{q}^{-1} = \frac{\mathbf{q}^*}{\|\mathbf{q}\|^2} = \mathbf{q}^*$$

- **Rotation transformation** is built by two quaternion multiplications-the normal and its conjugate

- rotates the **vector v from the fixed frame to the body frame** represented by \mathbf{q}

$$\mathbf{w} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v} \end{bmatrix} \otimes \mathbf{q}^*$$

where \mathbf{v} can be rewritten as location in x, y and z axis. For example,

$$R_x(\mathbf{q}) = \mathbf{q} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \mathbf{q}^* = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 \\ 2(q_1 q_2 + q_0 q_3) \\ 2(q_1 q_3 - q_0 q_2) \end{bmatrix}$$

- u is the rotation axis (unit vector) and α is the angle of rotation

$$\mathbf{q} = \cos\left(\frac{\alpha}{2}\right) + \mathbf{u} \sin\left(\frac{\alpha}{2}\right)$$

- Calculating quaternions error dynamics

- desired \mathbf{q}_{ref} , measured \mathbf{q}_m , error \mathbf{q}_{err} --multiplying the reference with the conjugate of the estimated quaternion in Kronecker

$$\mathbf{q}_{err} = \mathbf{q}_{ref} \otimes \mathbf{q}_m^*$$

- Angular errors: the reference is **demanding a rotation more than π radians**, the closest rotation is the **inverted direction** and this is found by examining q_0 . If $q_0 < 0$ then the desired orientation is more than π radians away

$$\begin{bmatrix} \Phi_e \\ \Theta_e \\ \Psi_e \end{bmatrix} \approx \text{sgn}(q_e^0) \begin{bmatrix} 2q_e^1 \\ 2q_e^2 \\ 2q_e^3 \end{bmatrix}$$

其中 sgn 函数就对应着计算 q_0 的正负

Control strategies for multicopters

- Inner loops for attitude dynamics **stabilization**
- Outer loops for **translational dynamics**

Linear

Examples

- Proportional Integral Derivative (PID)
- Linear Quadratic Regulator (LQR)
- LQG

Features

- a linear **approximation** of the system dynamics
- used when the attitude of the multicopter involves **angles that are close to zero**

PID controller

- Pros: Easy to design; Intuitive to tune manually (not necessarily easy on a real drone)
- Cons: Imprecise nature of the model due to inaccurate modelling; Limits the performance
- Effects of increasing a parameter independently

Parameter Increased	K_P	K_I	K_D
Rise time	↓	↓	-
Peak overshoot	↑	↑	↓
Settling time	-	↑	↓
Steady-state error	↓	Eliminate	-
Stability	↓	↓	Improve if K_D is small

LQR/LQG

- Features
 - a multirotor (linearized around an equilibrium point) by **minimizing a suitable cost function**
 - Pros: optimal control method and better than PID (LQR); not need complete knowledge of the state and can rely on estimator/observer (LQG)
 - Cons: requires tuning of Q and R Matrixes
- Practical example-Foldable Drone
 - Attitude control (providing **torques**) requires adaptation since morphology has an impact on the rotation dynamics
- Effect of Q and R matrixes
 - design parameters to penalize (1) the **state** variables and (2) the **control** signals
 - Large value: stabilize the system with less change in **State (Q) and Control input (R)**
 - Small value: stabilize the system without “caring too much” about ...

Nonlinear

Examples

- Model Predictive Control (MPC)
- Artificial Neural Networks

Features

- The **stability domain** of a drone controlled with a nonlinear controller can be **bigger**
- **a better model** of the drone can lead to better performances
- **allows to take explicitly into account aerodynamic effects and forces** acting on the drones

An introduction to fully actuated multirotor UAVs

- conventional ones have coupling between the horizontal translational and rotational dynamics

- fully actuated can help decouple translation and rotation; have direct control to corresponding DoF

key: **allocation matrix changes**

- Fixed-tilt
don't need additional motors
- Variable-tilt: change tilt of propellers to achieve desired motion
can achieve optimal configuration but with **complex mechatronics and control**

Checkpoints

- What is the size of the control allocation matrix of an hexacopter with **4 controlled degrees of freedom**?

- hexacopter 六轴 , 6motors
- Effectiveness matrix: 4x6
- Controller allocation: 6x4
Output: 6 motors; input $B \times (3\text{torque} + 1\text{thrust})$

- transformation relationship to torque and thrust for each rotor

- Why do we use the **pseudo inverse** instead of standard matrix inverse to compute the control allocation matrix from the actuator effectiveness matrix?

there existing other layout of multicopters except for quadcopters with 4 rotors

- What happens when a rate setpoint $[0 \ 0 \ 1]$ is applied?

rotate counterclockwise with 1 rad/s in yaw angle

- What are the input and output of the rate controller?

Input: rate setpoint

Output: generated angle torque

- What happens when an attitude setpoint $[0 \ 0 \ \pi/2]$ is applied?

rotate counterclockwise with $\pi/2$ in yaw angle

- What are the input and output of the attitude controller in a cascaded architecture?

Input: quaternion setpoint

Output: generated angle rotation rate

- What is the output of the attitude controller in a cascaded architecture?

 Output: generated angle rotation rate?

- How to compute attitude error quaternion from the estimated attitude quaternion and the attitude setpoint quaternion?

State Estimation (week3&4)

Introduction to State Estimation

- **STATE:** variables used to describe the mathematical state of a dynamical system
- **ESTIMATION:** finding an approximation of state variables even if the data is incomplete, uncertain, or unstable
- **estimate** something which you **can not see or measure directly**

Why State Estimation in Robotics?

- Perfect model & sensor are not available; **Errors** exist in **both sensors & models**
 - State of the robot (proprioception)
 - State of the environment (exteroception)
- **State Estimation on Aerial Robots**

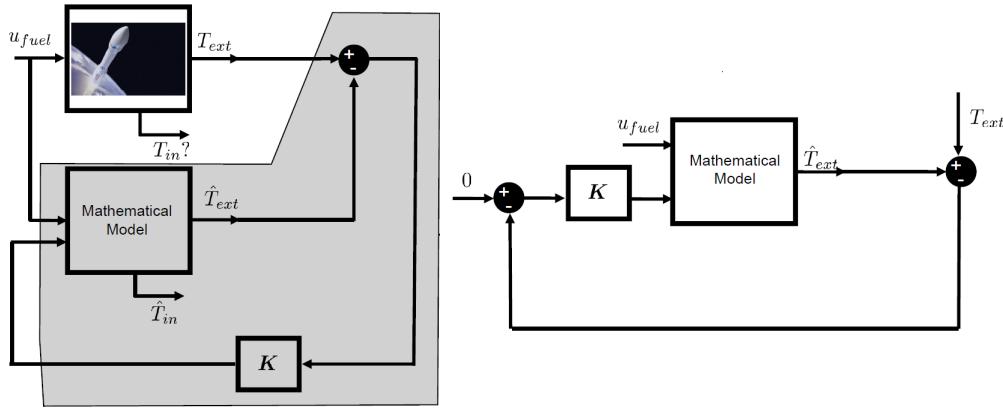
Height; **Attitude; Angular velocity** (Most important as they are the **primary variables used for the stabilization** of the UAV); linear velocity

 - main sensors
 - **Inertial Measurement Unit (IMU)**
 - **Height Measurement** (Acoustic, infrared, barometric, laser based)
 - Others: Global Navigation Satellite System (GNSS, outdoor) or VICON (indoor); Camera, Kinect, LIDAR

Estimation for deterministic systems

State Observer (or Luenberger Observer)

- minimize the error between measured and estimated values



- how do we choose the “feedback loop” gain K ?

Dynamical Systems

Linear systems

Continuous-time

$$\dot{x}(t) = Ax(t) + Bu(t)$$

$$y(t) = Cx(t) + Du(t)$$

Discrete-time

$$x(k+1) = Ax(k) + Bu(k)$$

$$y(k) = Cx(k) + Du(k)$$

where:

x : State vector

u : Input vector

y : Output vector

A : State (or system) matrix

B : Input matrix

C : Output matrix

D : Feedforward matrix

$f(\cdot), g(\cdot)$: nonlinear functions

State Estimation for stochastic systems

Recap of fundamental concepts

Kalman filter (KF)

use Gaussians to implement a Bayesian filter. That's all the **Kalman filter** is - a Bayesian filter that uses Gaussians

Intro to EKF, UKF and particle filters

State Estimation in aerial robotics

Checkpoints

- What structure does the Luenberger Observer remind you?
 control loop?
- What does the Kalman Filter (KF) gain do?
- For which kind of dynamical systems the KF is useful?
- What do we need to know a-priori if we want to apply the KF?
- What are the advantages/disadvantages of the **UKF** w.r.t an **EKF**?
- What are the advantages/disadvantages of a **particle filter** w.r.t an **UKF**?

Exercise summary

Navigation Methods (week5)

Perception (week5)

Fixed-wing drones (week6)

Aerial Swarms (week7)

Intro

- Drone light shows

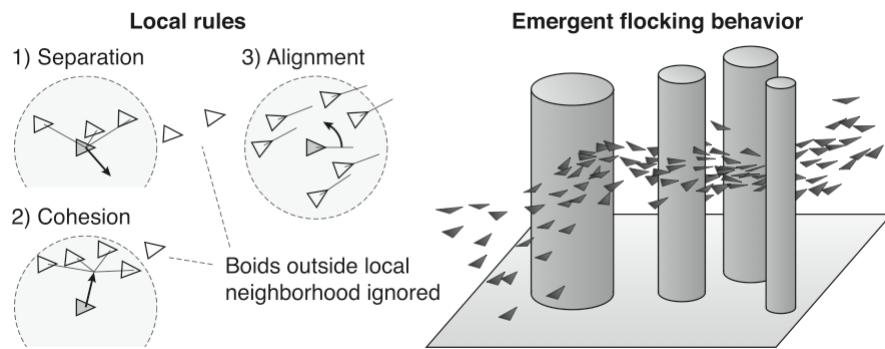
Centralized = agents transmit individual position to ground computer and receive next location

- Collective Motion in nature

Decentralized = agents rely on **local information and computation**

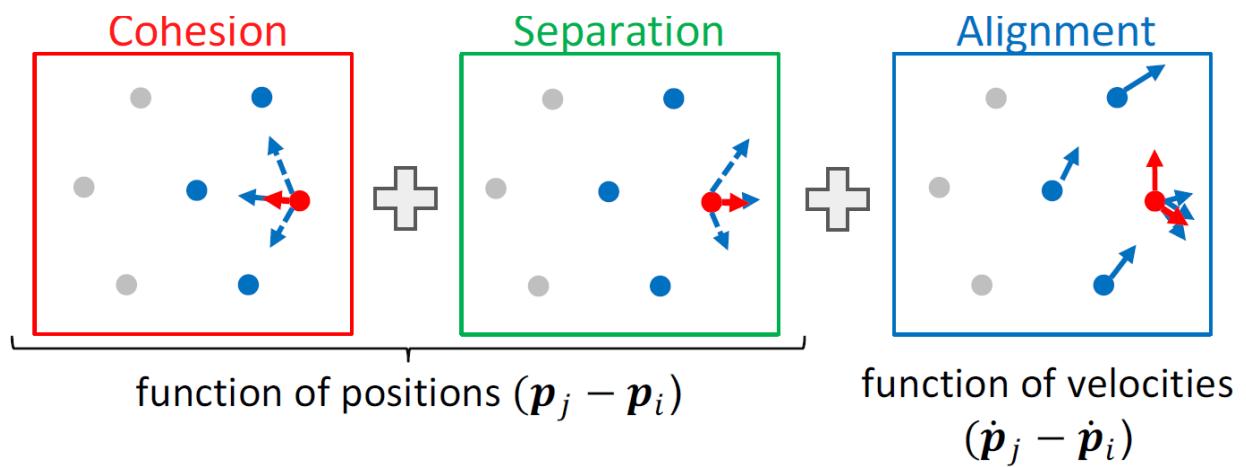
Reynolds flocking algorithm (Reynolds, 1987)

- radius of communication or neighborhood R



- **Separation**: avoid collision
- **Cohesion**: attempt to keep close
- **Alignment**: attempt to match velocity

Reynolds flocking: model



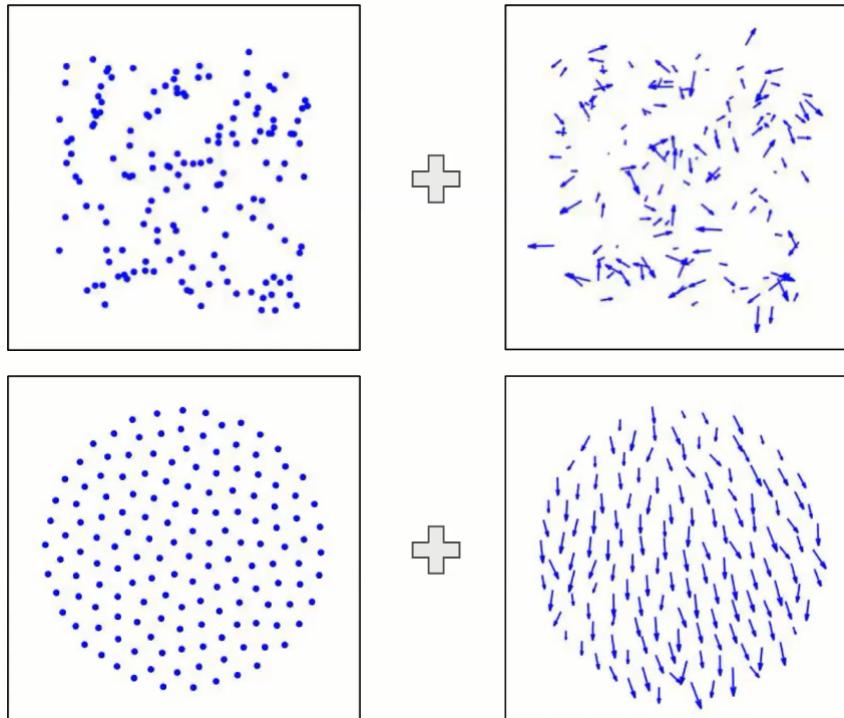
- Equations

$$\begin{aligned} \forall i \in \{1, 2, \dots, N\} \\ \ddot{\mathbf{p}}_i = \mathbf{u}_i \\ \mathbf{u}_i = c_c \frac{\sum_{j \in N_i} (\mathbf{p}_j - \mathbf{p}_i)}{|N_i|} - c_s \frac{\sum_{j \in N_i} \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}}{|N_i|} + c_a \frac{\sum_{j \in N_i} (\dot{\mathbf{p}}_j - \dot{\mathbf{p}}_i)}{|N_i|} \end{aligned}$$

constant gains

$a_{coh,i}$ $a_{sep,i}$ $a_{align,i}$

- Set of agents in neighborhood N
- identity of i -th agent
- position \mathbf{p}_i
- velocity $\dot{\mathbf{p}}_i$
- acceleration $\ddot{\mathbf{p}}_i$ = control command
- acceleration term due to the cohesion/separation/alignment $\mathbf{a}_{coh,i}$, $\mathbf{a}_{sep,i}$, and $\mathbf{a}_{align,i}$
- constant gains corresponding to the cohesion/separation/alignment C_c , C_s , and C_a
- Equilibrium



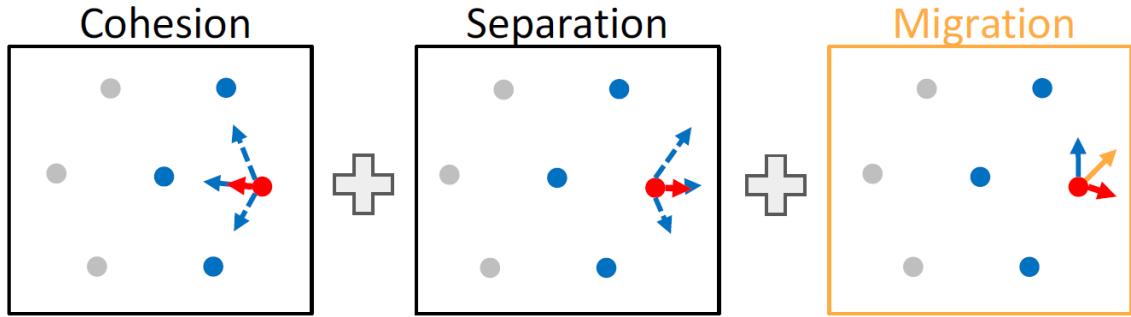
- Positions **converge** to a lattice formation (晶格式)
- Velocities **converge** to the average of initial velocities

$$\lim_{t \rightarrow \infty} \dot{\mathbf{p}}_i = \frac{\sum_{i \in \{1, 2, \dots, N\}} \dot{\mathbf{p}}_i(0)}{N}$$

Reynolds flocking with migration

- new **migration rule** steers the swarm towards a desired direction

- replaces the alignment rule
- cohesion and separation rules are kept to regulate the agents distances



- Equation

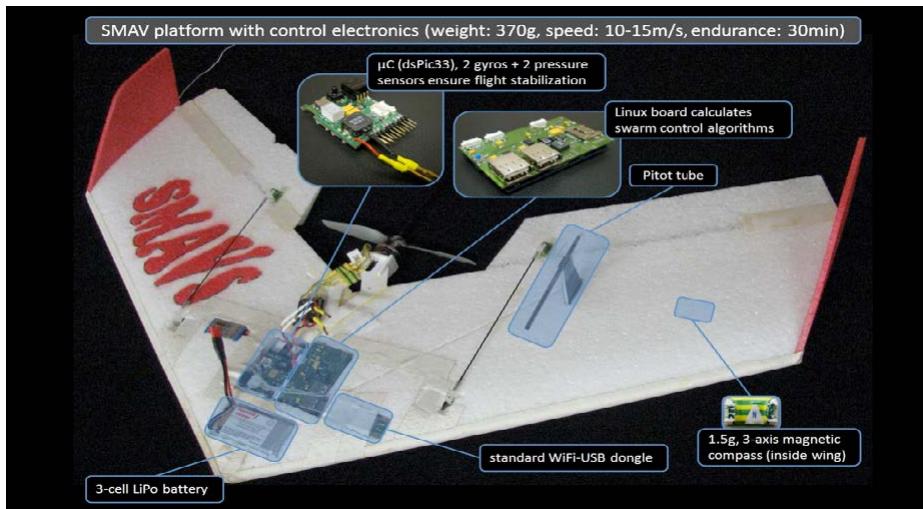
$$\ddot{\mathbf{p}}_i = \mathbf{u}_i, \mathbf{u}_i = c_c \frac{\sum_{j \in N_i} (\mathbf{p}_j - \mathbf{p}_i)}{|N_i|} - c_s \frac{\sum_{j \in N_i} \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}}{|N_i|} + c_m \frac{\mathbf{v}_{mig} - \dot{\mathbf{p}}_i}{1}$$

$$\forall i \in \{1, 2, \dots, N\}$$

- parameters

- migration velocity \mathbf{v}_{mig}
- Denominator = 1 since **neighbors are not relevant for migration**

Case: Aerial swarms for disaster mitigation



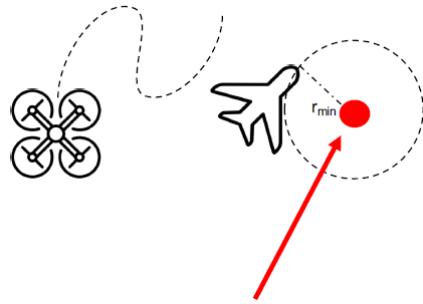
SMAV platform with control electronics

Communication radius and turning angle

- large communication radius -> can make sharp turn together because of knowing the position of other robots
- smaller communication radius -> may separate and gather into a flocking often

Virtual agents for flocking with fixed-wing drones

- Winged drone flies around Virtual Agent which moves according to Reynolds rules

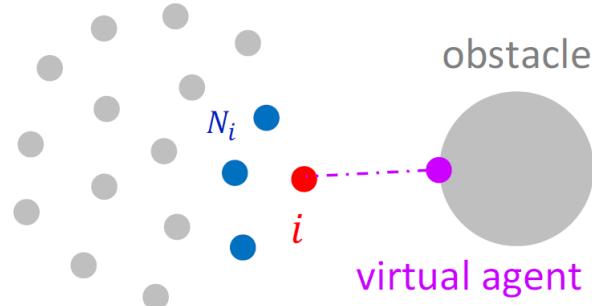


- Varga et al., Distributed Formation Control of Fixed Wing Micro Aerial Vehicles for Uniform Area Coverage, IROS 2015

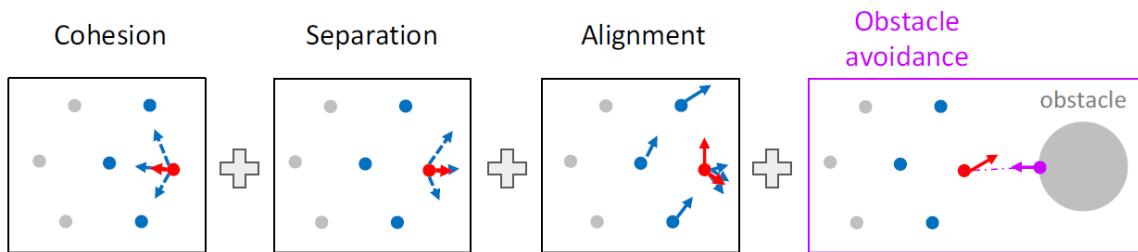
video: <https://youtu.be/FYsd2VckGA0>

Reynolds flocking with obstacles (Virtual agents)

- Obstacles are modelled as **virtual agents**
 - Its **position** is the obstacle's **closest point** to the agent
 - Its **velocity** is **perpendicular to the tangent** to the obstacle
- ($\mathbf{p}_k, \dot{\mathbf{p}}_k$) position and velocity of the virtual agent
- Virtual agents exert **separation** and **alignment** effects, but not **cohesion** (not collide with the agent)



- Visualization



- Equation (two extra separation and alignment term regarding obstacles)

$$\ddot{\mathbf{p}}_i = \mathbf{u}_i$$

$$\mathbf{u}_i = c_c \frac{\sum_{j \in N_i} (\mathbf{p}_j - \mathbf{p}_i)}{|N_i|} - c_s \frac{\sum_{j \in N_i} \frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|^2}}{|N_i|} + c_a \frac{\mathbf{v}_{mig} - \dot{\mathbf{p}}_i}{1} - \left[c_s \frac{\mathbf{p}_k - \mathbf{p}_i}{\|\mathbf{p}_k - \mathbf{p}_i\|^2} + c_a (\dot{\mathbf{p}}_k - \dot{\mathbf{p}}_i) \right]$$

$$\forall i \in \{1, 2, \dots, N\}$$

Other models

Vicsek model: particles in confined environments (密闭环境)

Vasarhelyi et al., Optimized flocking of autonomous drones in confined environments, Science Robotics, 2019

DOI: <http://doi.org/10.1126/scirobotics.aat3536>

Video: <https://youtu.be/E4XpyG4eMKE>

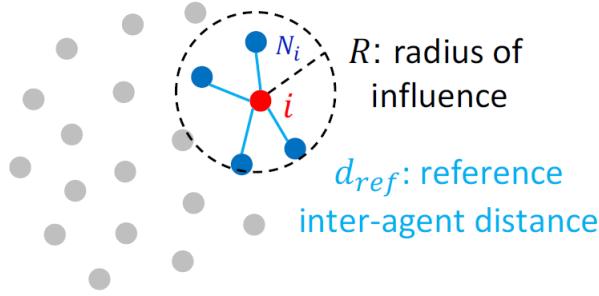
Project web: <http://hal.elte.hu/drones/scirob2018.html>

- Rules
 - Separation
 - Self propulsion: Makes the agent match a preferred speed
 - Friction: Viscosity (internal friction) for alignment and oscillation damping
- Equation

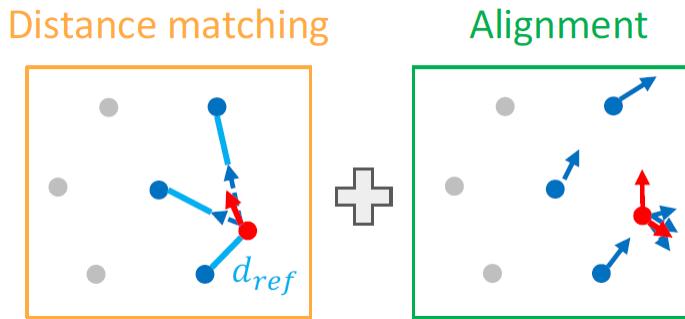
$$\begin{cases} \dot{\mathbf{p}}_i = \mathbf{u}_i \\ \mathbf{u}_i = \mathbf{v}_{sep,i} + \mathbf{v}_{spp,i} + \mathbf{v}_{fric,i} \end{cases}$$
- The full equation contains 12 parameters and requires heuristic methods for optimization

Olfati-Saber model

R. Olfati-Saber, Flocking for multi-agent dynamic systems: algorithms and theory, IEEE Transactions on Automatic Control, 2006



- Rules
 - Distance matching
 - Makes the agents **match a desired inter-agent distance**
 - **Replaces cohesion and separation rules** of Reynolds model
 - Mathematically defined as a potential function
 - Alignment: attempt to match the velocity and direction



- Equation

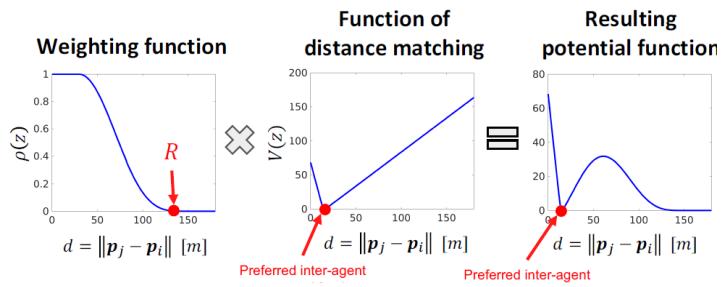
$$\ddot{\mathbf{p}}_i = \mathbf{u}_i$$

$$\mathbf{u}_i = c_d \frac{\sum_{j \in N_i} \nabla(\rho(\mathbf{p}_j - \mathbf{p}_i) V(\|\mathbf{p}_j - \mathbf{p}_i\|))}{|N_i|} - c_a \frac{\sum_{j \in N_i} (\dot{\mathbf{p}}_j - \dot{\mathbf{p}}_i)}{|N_i|}$$

$$\forall i \in \{1, 2, \dots, N\}$$
 - radius of influence R
 - desired inter-agent distance d_{ref}
 - weighting function ρ
 - distance matching function V
 - gradient, derivative in three dimensions $\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right)$

- **distance matching example**
 - Components
 - **weighting** function 越近影响越大
 - **distance matching** function 越靠近 d_{ref} 越小，线性
 - Result: **potential function**

R : radius of influence = (150m)
 d_{ref} : target inter-agent distance = (20m)
 ρ : weighting function
 V : function of distance matching

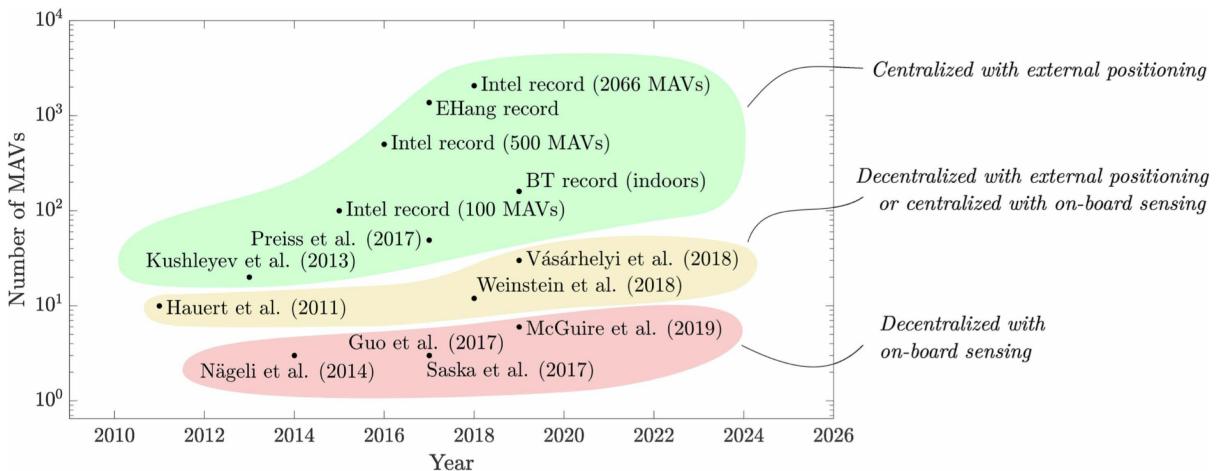


- o Note

- Principle of minimum potential: **minimum** defines the stable equilibrium of the system
- d_{ref} is a stable equilibrium
- The force acting on an agent is zero in the minimum of the potential. For $d = d_{ref}$, it holds $\nabla(\rho V) = \mathbf{0}$

Drone Swarms

Coppola et al., A Survey on Swarming With Micro Air Vehicles: Fundamental Challenges and Constraints, Front. Robot. AI, '20



The combination of centralized planning/control with external positioning has **allowed to fly significantly larger swarms**. The **numbers are lower** for the **works featuring decentralized control with external positioning**, or centralized control with local sensing

Three categories

1. Centralized with external positioning

latest: September 20 2020

3,051 drones

News: <https://www.guinnessworldrecords.com/news/2020/10/3051-drones-create-spectacular-record-breaking-light-show-in-china> (Company: <https://www.dmdav.com/>)

YouTube: <https://youtu.be/44KvHwRHb3A>

Bilibili: <https://www.bilibili.com/video/BV1jt4y1q762>

2. Decentralized with external positioning or centralized with on-board sensing

Vasarhelyi et al. (2019)

3. Decentralized with on-board sensing

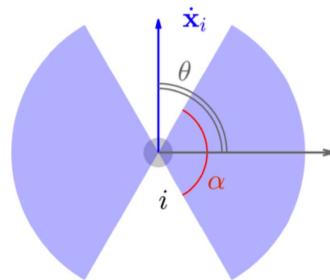
Saska et al. (2017)

Visual information in flocking

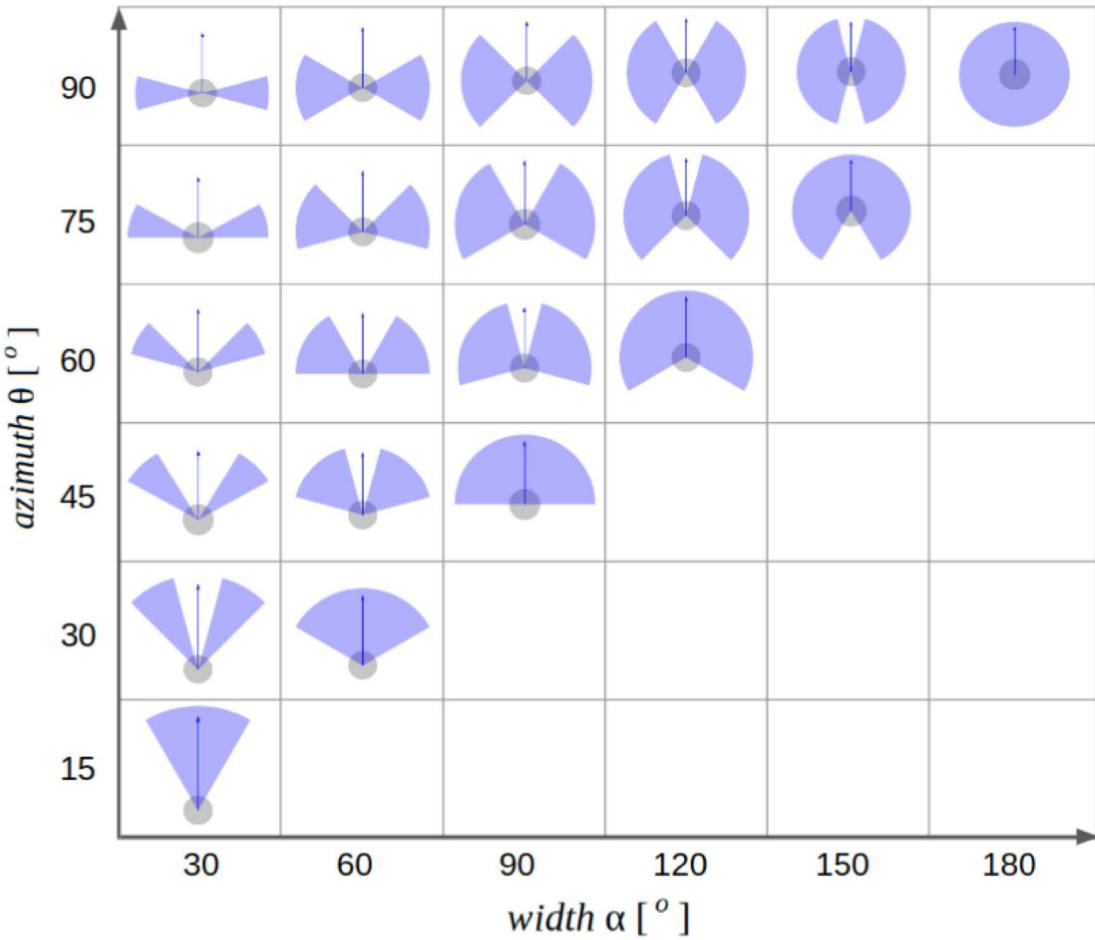
Soria2019IRC-influence of limited visual sensing using Reynolds

Soria et al., *The influence of limited visual sensing on the Reynolds flocking algorithm*, 2019

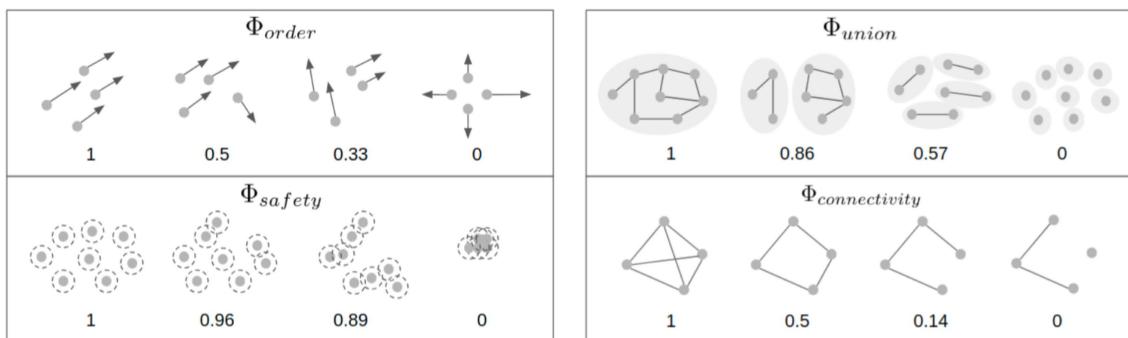
- generate flocks with different fields of view



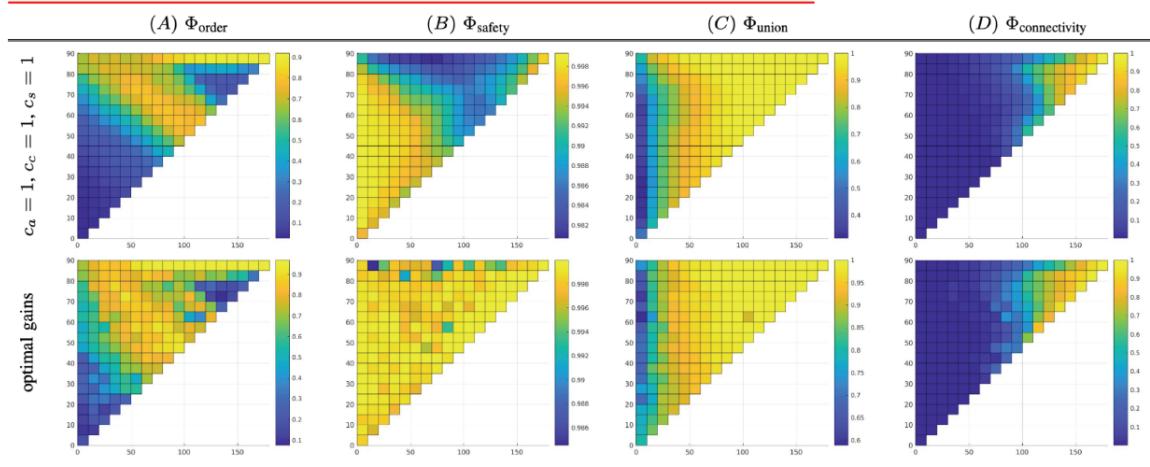
- azimuth/方位角 $\theta[^{\circ}]$
- width $\alpha[^{\circ}]$



- measure flocking performance (all individuals in the flock have the same visual configuration)
 - **Order:** measure of alignment
 - **Safety:** ability to avoid collisions
 - **Union:** ability to stay informed on neighbors
 - **Connectivity:** ability to broadcast messages among drones



- results

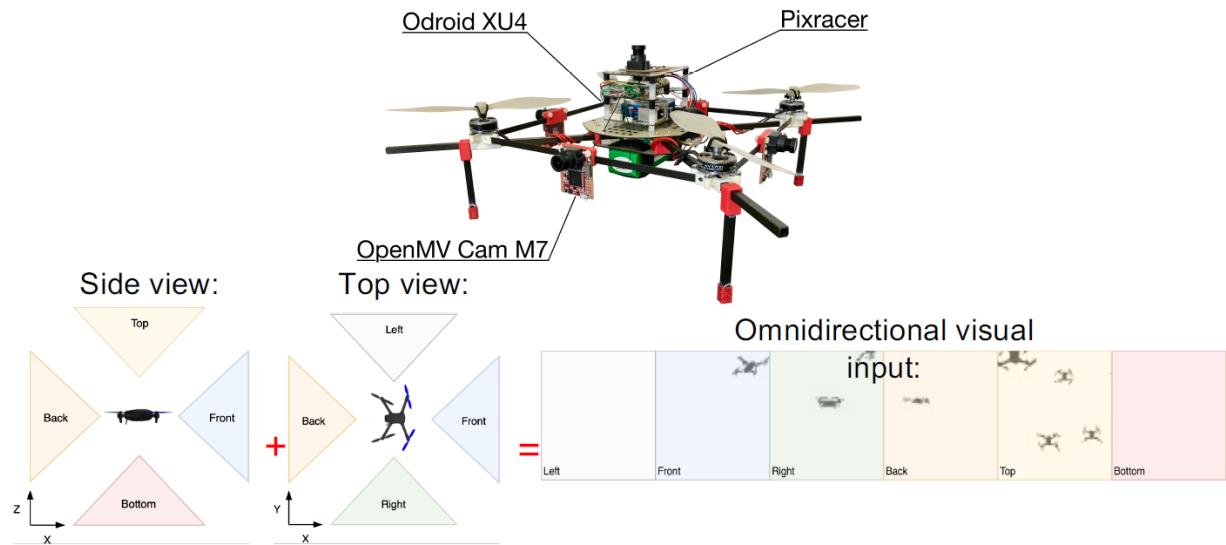


- o focus on order and safety (alignment and collision prevention capability)
- o largest azimuth and FoV has best performance
- o increase in either azimuth or FoV only will degrade the performance
- o safety can be achieved even with lower FoV

Schilling2019RAL-Learning to flock in simulation with vision

Schilling et al., Learning Vision-Based Flight in Drone Swarms by Imitation, RAL2019

- use 6 cameras in each side
- training on a dataset to generate the velocity vector for the drone

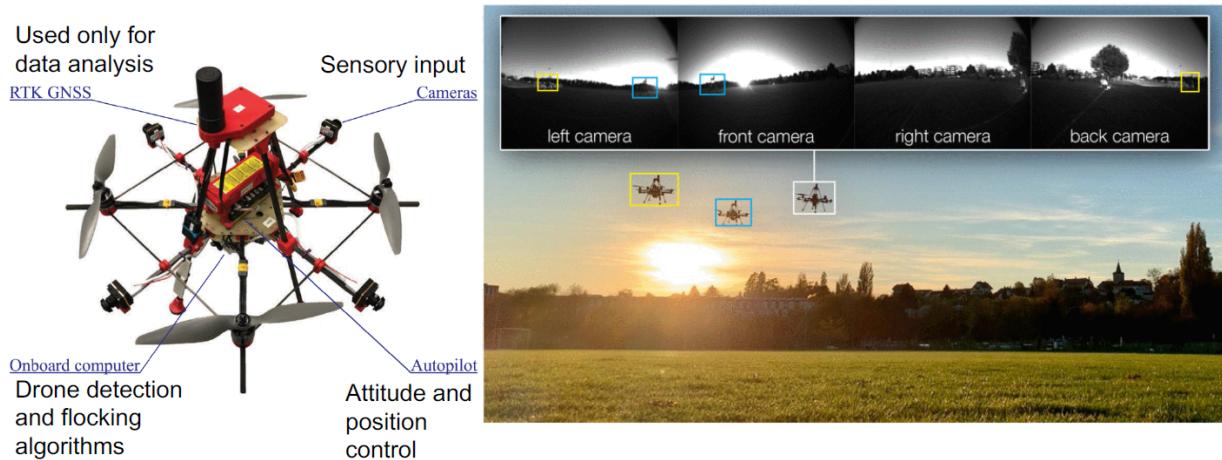


- Stages
 - o Dataset generation: Flocking algorithm as ground truth
 - o Training phase: Learn **mapping between vision and control output**
 - o Vision-based control: **Neural controller for collision-free and cohesive flight**
- Note
 - o work well in simulation indoor environment

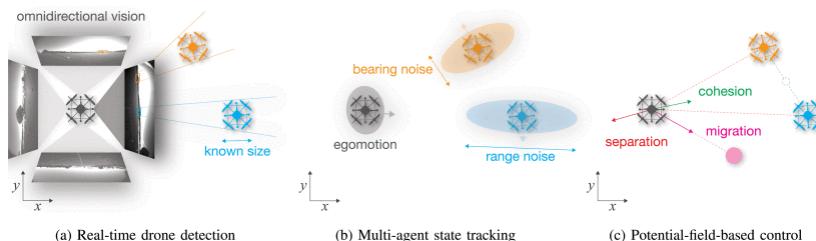
- it can be robust when individuals has different migration points
- cannot generalize well in background clutter and different lighting condition

Schilling2021RAL-Learning to flock outdoor with vision

Schilling et al., Vision-Based Drone Flocking in Outdoor Environments, RAL2021



- Setup
 - Drone with only with 4 cameras in four side
 - RTK GNSS is used to compute performance
 - train YOLOv3 tiny to recognize other drones using YOLO
- Control method



1. Real-time drone detection

- Input: images from 4 cameras
- Output: x,y coordinates of perceived drones in image frame coordinates
known size to compute corresponding distance

2. Multi-agent state tracking

- Input: Locations of drones & noise models
- Output: **Range and bearing** of all perceived drones with **noise**

3. Potential-field-based control

- Input: Range and bearing of all perceived drones
- Output: **velocity vector** resulting from Reynolds algorithm

Check points

- What information does each agent receive in the Reynolds flocking algorithm?
 position and velocity of self and neighbor agents
- How are obstacles modeled in Reynold's flocking
visual agent; integrate into equations with alignment and separation term
- How is a migration point incorporated in flocking algorithms
 add a velocity term?
- What does the Olfati-Saber algorithm ensure?
distance matching with potent ion function
- What are the three steps of vision-based drone flocking algorithm?
 1. Real-time drone detection
 2. Multi-agent state tracking
 3. Potential-field-based control

images from 4 cameras -> x,y coordinates of perceived drones in images -> Range and bearing of all perceived drones -> velocity vector

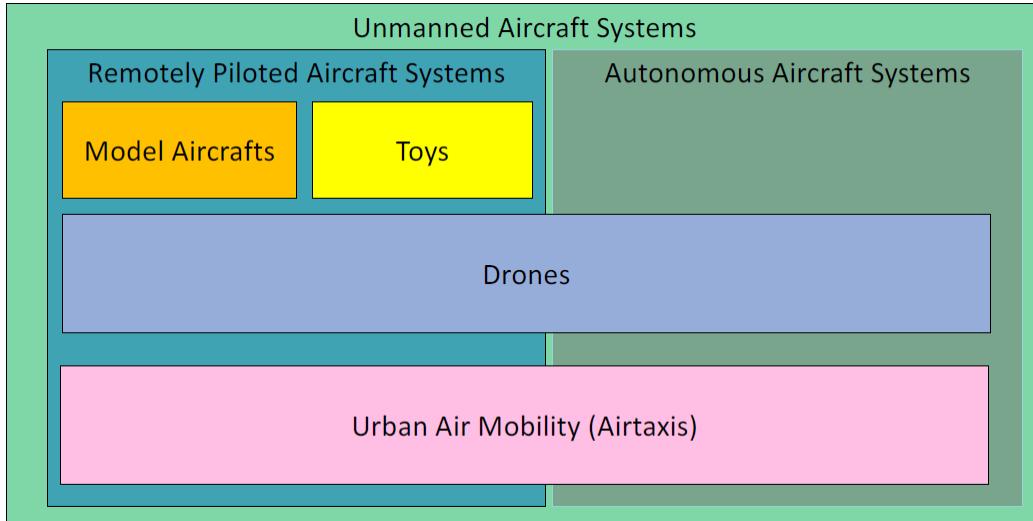
Flapping-Wing (week8)

Drone Regulations (week8)

Author: Markus Farner

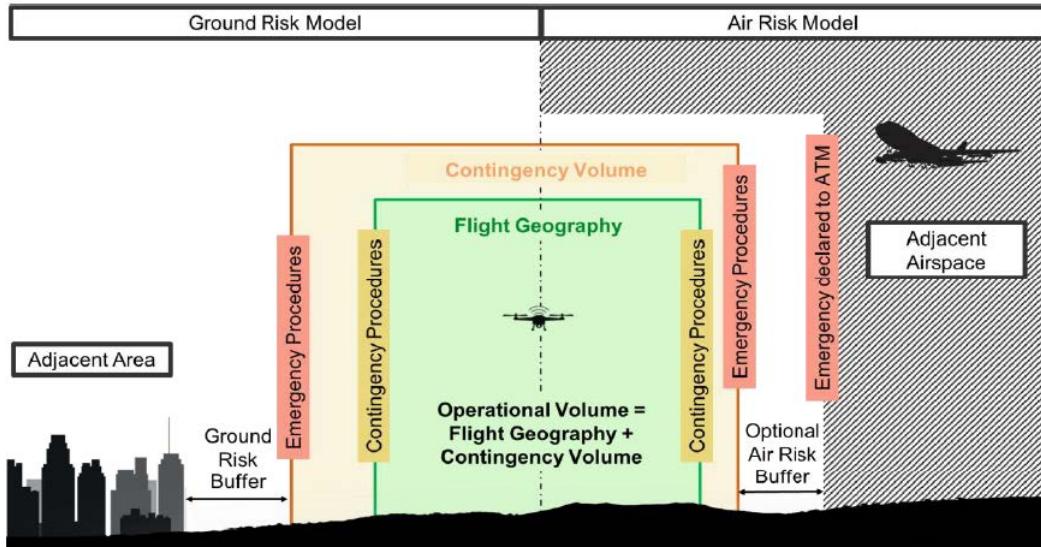
<https://www.bazl.admin.ch/bazl/en/home/good-to-know/drohnen.html>

- Unmanned Aircraft Systems (UAS) >= Drones; UAS = Remotely piloted aircraft systems / autonomous aircraft systems



- Rules in Aviation: Federal Office of Civil Aviation Switzerland
- Everything which is not forbidden is allowed -> Switzerland
Trust, less difficult for innovation
- 3 Pillar Concept / Drone Categories
 1. Open-Within the legal framework (No Authorization required)
 2. Specific-Not sufficiently safe (Authorization required)
 3. Certified-Approved to accepted standards
- Act
 - **Ordinance on Special Category Aircraft**
 - No authorization required for commercial flights
 - No distinction between Unmanned Aircraft and Model Aircraft
 - **DETEC Ordinance on Special Category Aircraft**
 - No authorization below **30kg**
 - Within direct visual contact (VLOS)
 - Not within a distance $\leq 100\text{m}$ around crowds
 - **ANSP (Skyguide) or Airport responsibility**
 - $> 5\text{km}$ Distance to civil & military airports/aerodromes
 - $< 150\text{m}$ AGL (Above Ground Level) within a CTR
- Act in EU
 - Open/Specific/Certified
 - Difference
 - restrictions: MTOM **25kg**
 - maximum flying altitude: **120m**
- Specific Category

Operational Volume = Flight Geography + Contingency Volume



- Robustness Levels: Integrity + Assurance
- U-Space

The U-space is a collection of decentralized services that collectively aim to safely and efficiently integrate drones into the airspace and enable drone operations alongside manned flight.

<https://www.bazl.admin.ch/bazl/en/home/good-to-know/drohnen/wichtigsten-regeln/uspace.html.html>

<https://www.skyguide.ch/en/events-media-board/u-space-live-demonstration/>

airspace in block to avoid collision and report the location for further path calculation

Checkpoints

UAS Hardware (week9)

Introduction

main component required

1. The aerial vehicle

- Air frame
- Actuators for propulsion and control
- Energy source
- Autopilot
 - Sensors for attitude estimation
 - Electronics for regulation, control and communication
 - Sensor and avoid system

2. Payload

- Cameras
- Environmental sensors (wind, temperature, humidity)
- Robotic arms for manipulation

3. Ground Control Station

- Communication systems
- Interface to monitor internal parameters and to send commands to the vehicle

Frame and materials

materials comparison

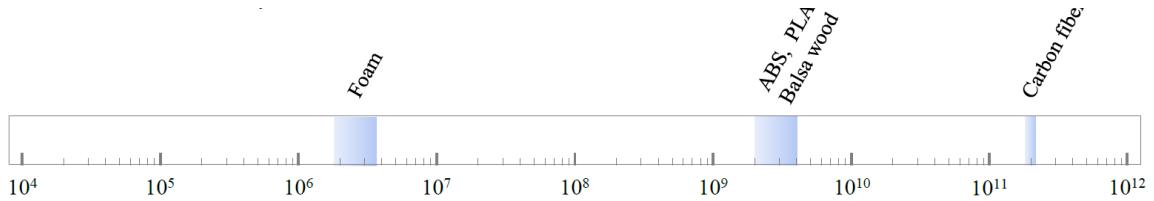
Material	Composite	ABS/PLA	Wood	Foam
Pros	Stiff, lightweight	Easy to manufacture by 3D printing or injection molding	Lightweight and cheap	Lightweight and soft, resistance to collision
Cons	Expensive, complex to manufacture	Heavier, less stiff	complex to work with	limited load
Comment	-	useful for prototyping	-	absorb energy, less prone to damage

metric when considering materials

- Young's modulus [[wiki](#)]

弹性模量，正向应力与正向应变的比值

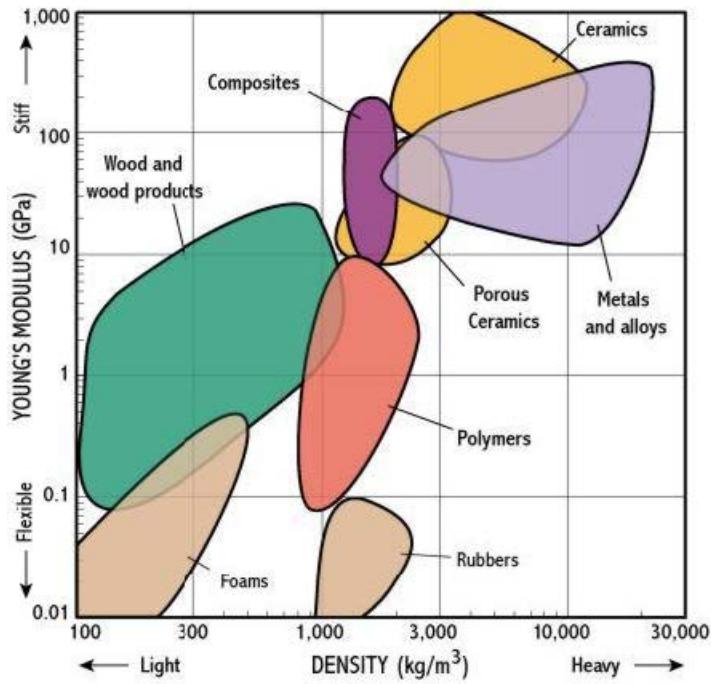
- measure of **stiffness**
- defines the relationship between stress and strain
- Foam < ABS/PLA/Wood < Carbon fiber



- Specific modulus [[wiki](#)]

比模量，单位密度的弹性模量，劲度 - 质量比，在航天工业中有广泛应用。

- elastic modulus per mass density of a material
- stiffness to weight ratio
- **High specific modulus materials** find wide application in UAVs where **minimum structural weight** is required.



Energy sources

Goal: power the robots to fly

Metric: energy density, power density, charging time and so on

Category

- Nickel-Cadmium (NiCd) | 镍镉
 - Mature and cheap
 - Low energy and power density -> short flight time
- Nickel-Metal Hydride (NiMh) | 镍氢电池

由**镍镉电池** (*NiCd battery*) 改良而来的，其以能吸收氢的金属代替**镉** (*Cd*)。它以相同的价格提供比镍镉电池更高的电容量、较不明显的**记忆效应**、以及较低的环境污染 (不含有毒的镉)

[wiki-zh]

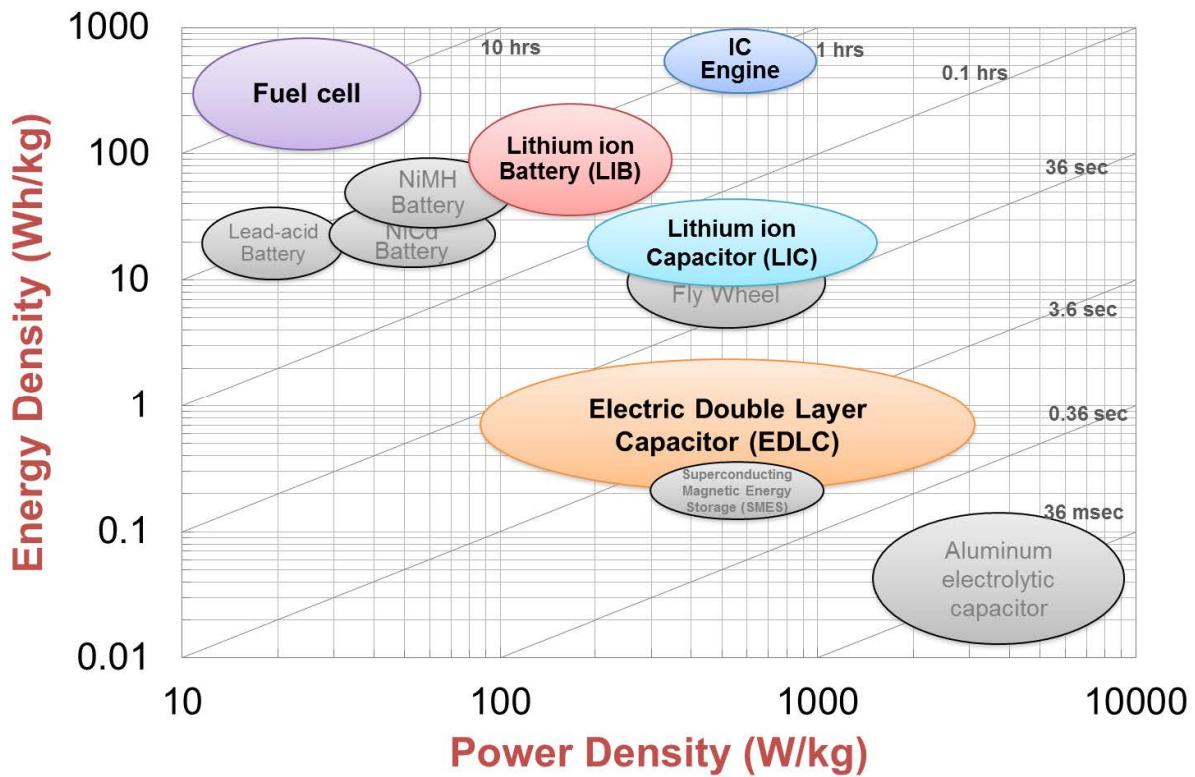
- Higher energy density than NiCd
- Lithium-Polymer (Li-Po) | 锂离子聚合物电池
 - rapidly growing market and performance
 - Higher energy and power density compared to NiCd
 - Regular geometry for easy integration, e.g., cuboid or cuboid
- Fuel
 - **Highest energy and power density**
 - **complex and higher weight**-requires tank, distribution system and maintenance
- Fuel cell
 - Electrochemical reaction of hydrogen fuel with oxygen

Energy and power density

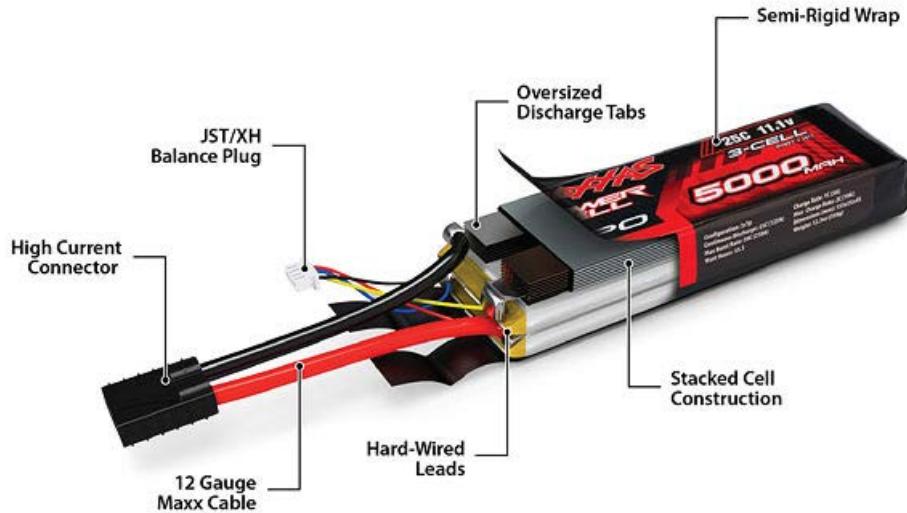
- energy density
amount of energy stored per unit volume or mass
- power density

how fast or quickly to discharge into mechanics

- amount of power (time rate of energy) per unit volume or mass
- Conclusion
 - Fuel has **highest energy and power density**
 - Fuel cell has highest energy but lower power density
 - LiB has higher energy and power density than NiMH and NiCd



Li-Po batteries



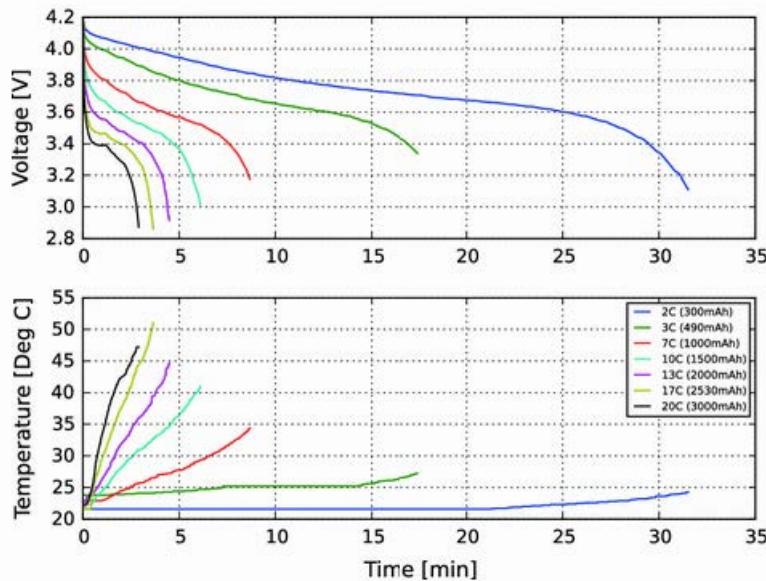
- most commonly-used UAV energy source
- Each **battery** composed of one or more **cells** connected in series

S=series, P=Parallel

- Each cell has
 - nominal voltage of 3.7 V
 - a maximum voltage of 4.2 V

- a capacity (mAh)
 - e.g., 1000 mAh
- a specific discharge and charge rate (C)
 - e.g., Discharge rate with $25-50C = 25-50$ A of max continuous discharge current;
Charge rate $2C = 2$ A

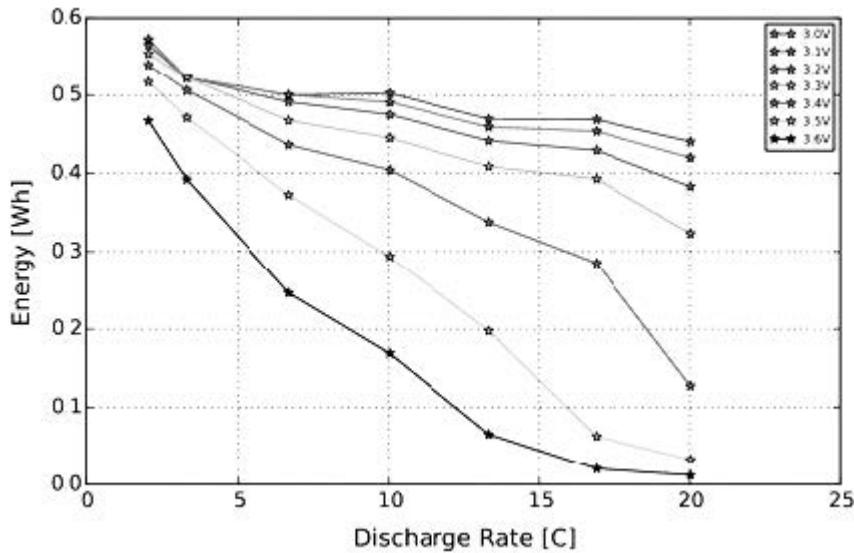
Discharge Curves of Li-Po battery



- not linear of time
- the discharge curve is determined by **the amount of current (expressed in “C”)** drawn from the battery.
- higher discharge rates -> faster rising temperature -> poses overheating risks.

Book: G. C. H. E. Decroon, M. Perçin, B. D. W. Remes, R. Ruijsink, and C. De Wagter, *The delfly: Design, aerodynamics, and artificial intelligence of a flapping wing robot*. 2015.

Energy Curve of Li-Po battery



- How much energy the same LiPo battery can provide until its voltage drops below a certain voltage
- 10 times higher battery load (discharge rate) -> 17 times shorter flight time
nonlinear relationship

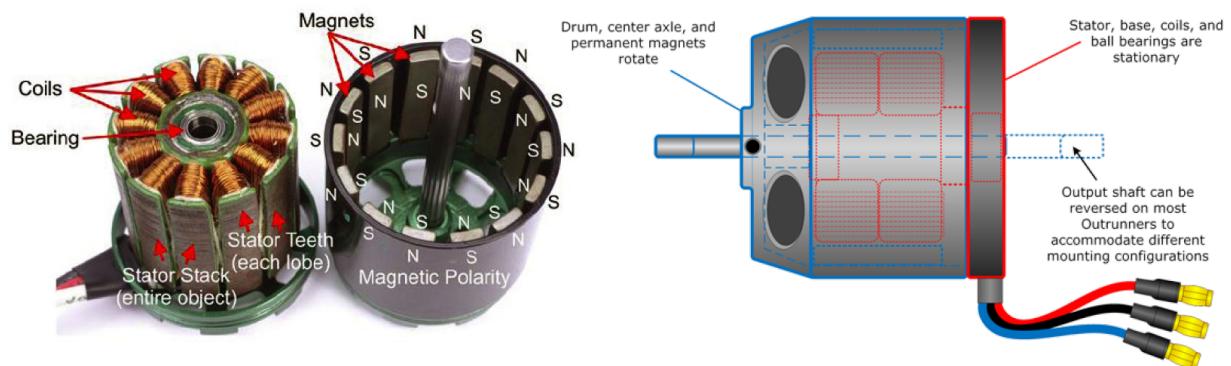
Actuators

Actuators for propulsion

	Electric motors	Combustion engine	Hybrid ²
Pros	clean and quiet; Reliable and easy to maintain; Fast to change operational state (accelerate and decelerate)	High weight to power ratio using fuel	Long endurance; Suited for fast change of speed
Cons	Limited weight to power ratio due to battery	Vibration, dirt, and noise; Requires tuning; Not suited for fast change of speed ¹	Complex and expensive

1. Combustion engine is not suited for fast change of speed (problem in controlling quadcopters)
2. Hybrid systems (fuel generator coupled with electric motor)
e.g. [skyfront](#) drone with 4.5 hour endurance (demonstrated) and 3 kg payload capacity

Electric motor example-Brushless DC electric motors



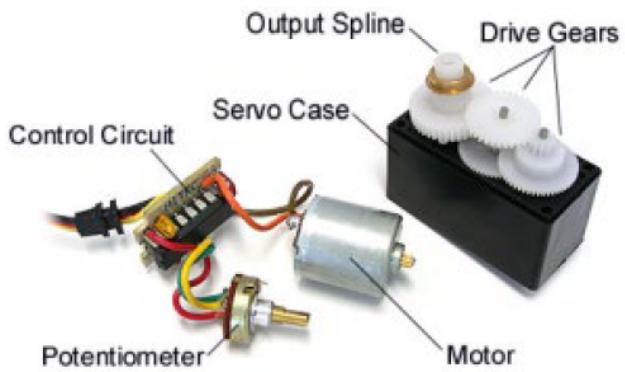
- Brushless: no electrical physical connection
- Pros
 - High efficiency and high torque/power density
 - High speed range
 - Large range of thrust (from 10^{-2} to 10^2 N)
- Cons
 - manufacturing complexity -> expensive
 - Control is complex and expensive requiring an **electronic speed controller (ESC, 电控)**
- Main motor data
 - 3 primary data:
 - Size
 - **Nominal voltage** (number of battery cells, e.g., 3S)
 - Speed constant KV (No load rpm/Volt)
 - High KV -> high speed and low torque
 - Low KV -> low speed and high torque

Actuators for control/maneuvering

Servomotors

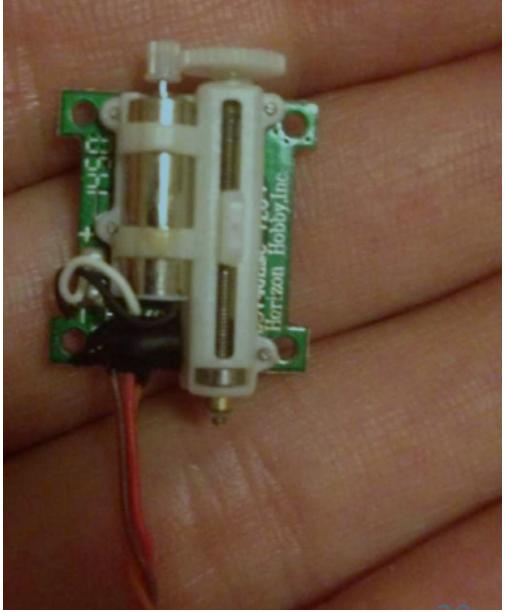
need to deflect the control surfaces

- rotary or linear actuators



- 3 wires (B-Ground, R-Voltage, Y-Signal) - send power and signal to control circuit
 - brush motor in small scale and connected to gear drive (set correct speed and torque, connect to potentiometer)
- potentiometer (电位器) sensor for angular position control

Examples of Servomotors

Rotary servos with push rod	Linear servos
	
	
Weight: 1 to 500 g	Weight: 1 to 5 g
-	to control elevators, flaps and ailerons

Propellers

to convert power (delivered by a rotating shaft) into thrust

Characteristics

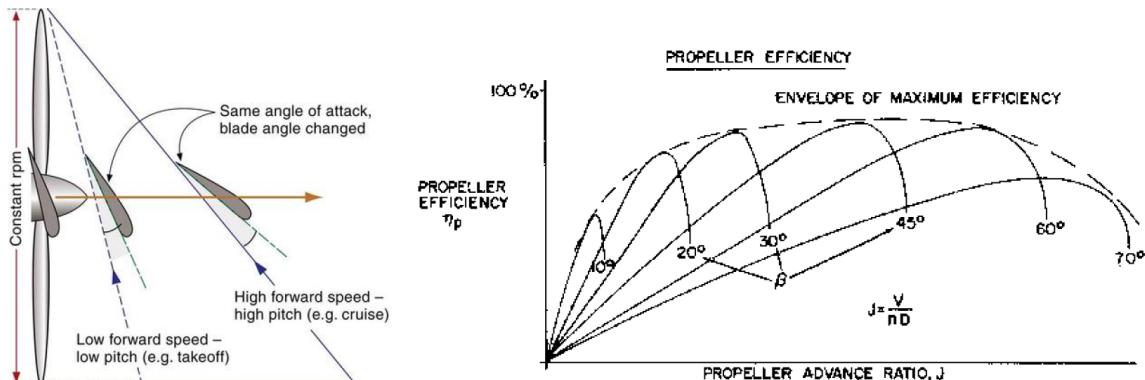
- Diameter
 - the length of prop from tip to tip
 - larger diameter are more efficient
- Pitch
 - the angle of attack in the propellers
 - higher at the root (center) and lower at the tip
- Number of blades
 - majority of propellers used in UAVs have two blades because of efficiency
 - 3 or 4 blades are more compact for a given thrust

Pitch and efficiency at different cruise speed

- the blade pitch could be varied in flight
- propeller advance ratio J VS Propeller efficiency η_p

$J = V/nD$, **flight** speed V , **angular** speed n , and Diameter $D \rightarrow$ tip speed

choose the suitable propeller according to the **diameter and pitch** to achieve better **efficiency curve**



- **Variable pitch propeller with servo** \rightarrow achieve best efficiency all the time



Choose the right combination actuator and propeller

match the propeller and the motor to maximize propulsive efficiency

- modelling (<http://web.mit.edu/drela/Public/web/qprop/motorprop.pdf>)
- calculation software (<http://ecalc.ch/>)
- testing

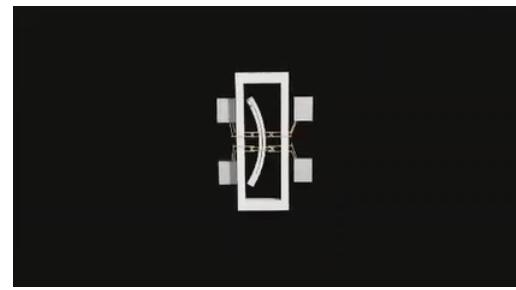
Sensors

- Proprioceptive sensors: measure the internal state of the UAV, mainly for control
 - IMU: accelerometer, gyroscope and magnetometer
 - Pressure / altitude sensors
 - GPS
 - Velocity (Airspeed sensors)
 - Power sensor
- Exteroceptive sensors: provide information about the UAS environment and are usually carried as a payload
 - Camera and sonar for obstacle detection and avoidance
 - Environmental sensors
 - Camera for video streaming, thermal or hyperspectral imaging

Gyroscopes

measure changes in vehicle orientation

- Type: Mechanical; Optical; Micro-electromechanical systems (MEMS)
- Categories
 - Orientation -> directly measure angles (very rare in robotics!)
 - Rate gyros -> measure rotation velocity, which can be integrated
- Cons
 - all gyroscopes are prone to drift unless the error is corrected through reference to some alternate measurement
(not relative to absolute reference but past state)
 - the drift will eventually exceed the required accuracy
- MEMS rate gyros

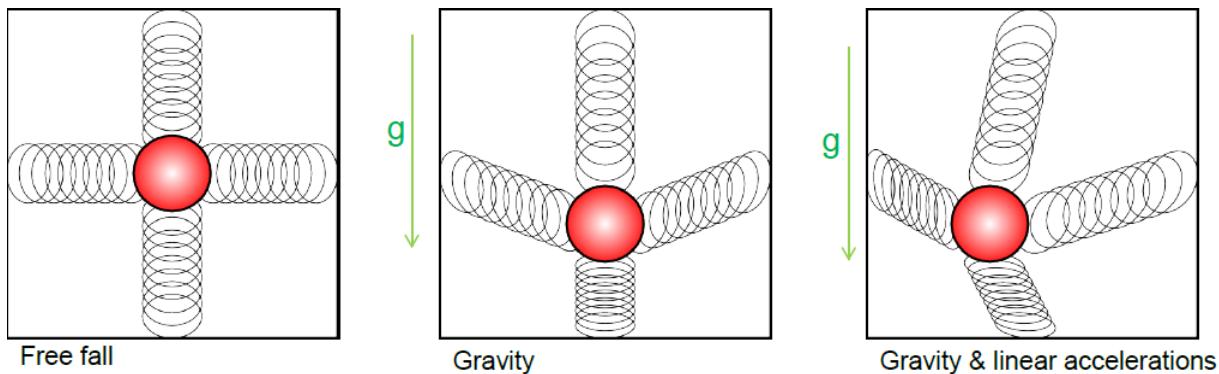


- vibrating mechanical elements to sense **Coriolis acceleration** (振动机械元件以感应科里奥利加速度)
 - induce an vibration outside the plane and measure the out-of-plane motion
- Pros -> replacing mechanical and optical gyros
 - have no rotating parts
 - have low-power consumption requirements
 - small

Accelerometers

measure acceleration to get the inertial information

- behaves as a damped mass on a spring
- MEMS use cantilever beams (悬臂梁) and a proof mass.
- The way of measuring the beam deflection is often capacitive or piezoresistive (电容性或压阻性的)
- have three axes => inclinometers (inclinometers)



Magnetometers

Exteroceptive

- electronically compass 电子罗盘
- direct measure of the magnetic field

- Hall-effect (霍尔效应)
- Flux Gate (磁通罗盘) [[wiki](#)]
 - two perpendicular circuits to get the force

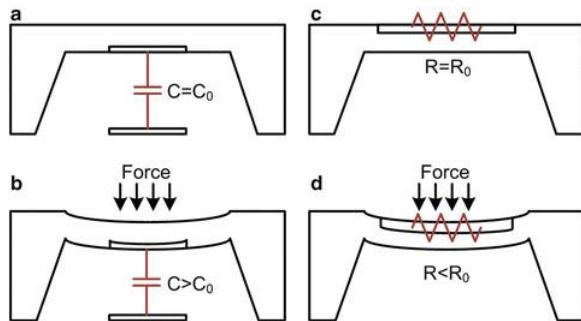
- Pros

- weakness of the Earth magnetic field
- easily disturbed by magnetic objects or other sources
- not working in indoor environments
because of wires or other electronic device

Pressure / Altitude sensors

to measure the altitude according the atmosphere pressure

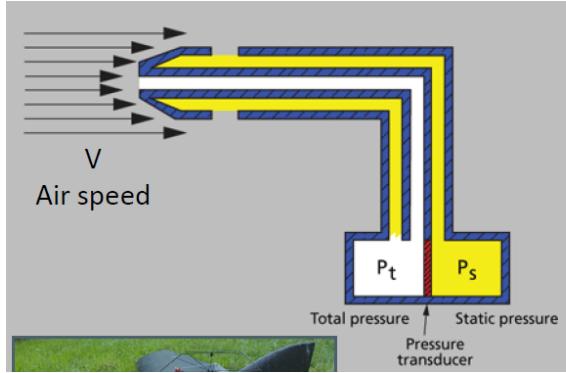
- measure the changing distance of the deforming membranes: piezoresistive (压阻式), capacitive, optical, electromagnetic, etc



- has a vacuum inside the housing to get an absolute pressure

Airspeed sensors

- measured using a pitot tube (皮托管)
- directed into the direction of motion
- the difference between the stagnation pressure (static + dynamic pressure) -> the airspeed



$$\left(p_s + \rho \frac{V^2}{2} \right) = p_t$$

$$V = \sqrt{\frac{2(p_t - p_s)}{\rho}}$$

Measured by a pressure transducer

- measures the **speed** of a UAV with **respect to the air** (airspeed) -> used for fixed-wing UAV
not the absolute speed of the UAV



Global positioning system (GPS)

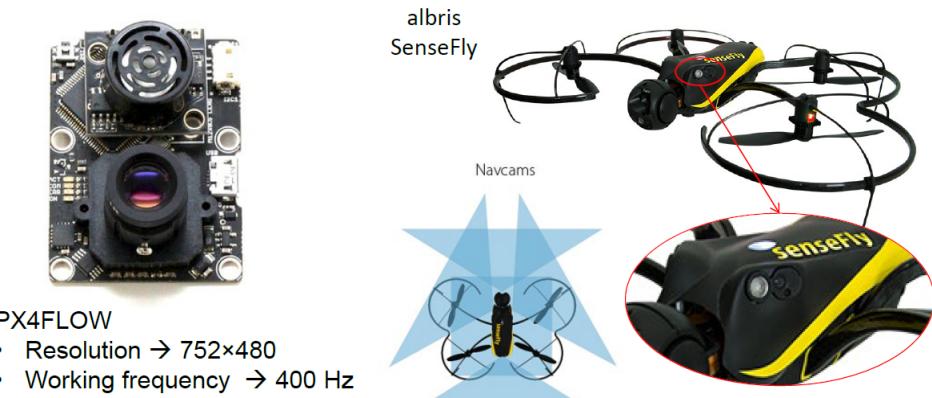
- Global Navigation Satellite System (GNSS): This term includes
 - e.g. the GPS, GLONASS, Galileo, Beidou and other regional systems.
 - Pros: multiple satellites is accuracy, redundancy and availability at all times.**
- Relatively lower accuracy: have a position accuracy within 20 m in the horizontal plane and 45 m in the vertical plane
- enhancement techniques**
 - WAAS or other ground tower-based services: static
get close to **1-2 m accuracy**
 - Real time Kinematic (RTK)** positioning: Base Station receiver and a receiver on the vehicle
close to **1 cm accuracy**

Power sensors

- measure the **battery voltage/current**
- -> trigger safety procedures (return to home on low battery.)

Optic flow cameras

- used to improve **state estimation** for accurate positioning and **height estimation** also in **GPS denied environments**
- measure the movements along x, y and z direction by tracking the features
- used for obstacle avoidance, position holding, and precise landing

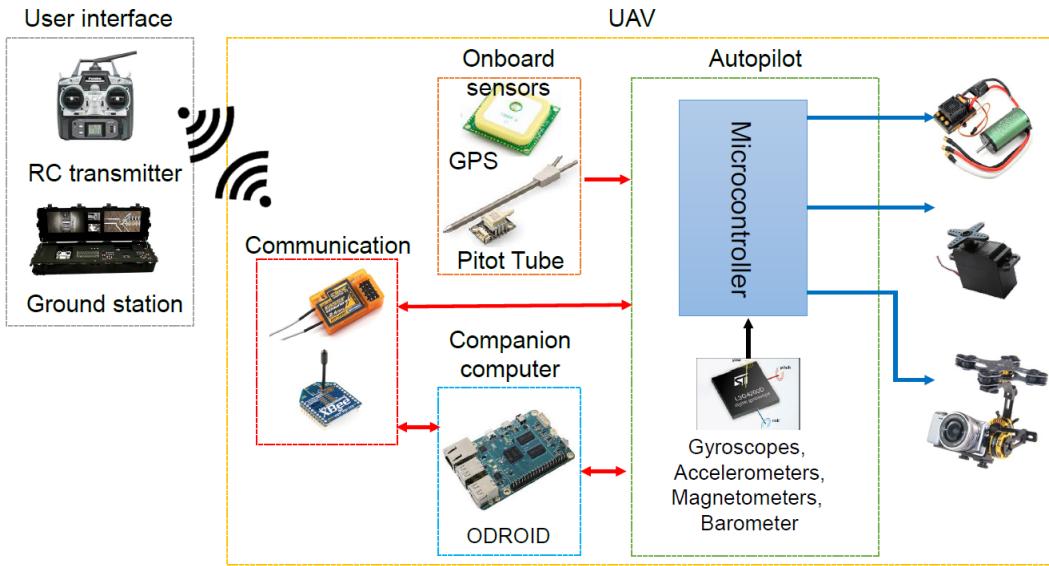


Autopilots

system used to **stabilize** (e.g. attitude stabilization of a multicopter) or to **control the trajectory**

- Microcontroller
- Attitude sensors
- I/O interfaces

receive the input information -> process information -> send actuator commands



- A **companion computer** is used to perform high level computation tasks **that can't be directly performed by the autopilot**

Communication protocols

- **RC transmitter** to communicate between RC and flight controller
- **Telemetry** to communicate between PC and flight controller

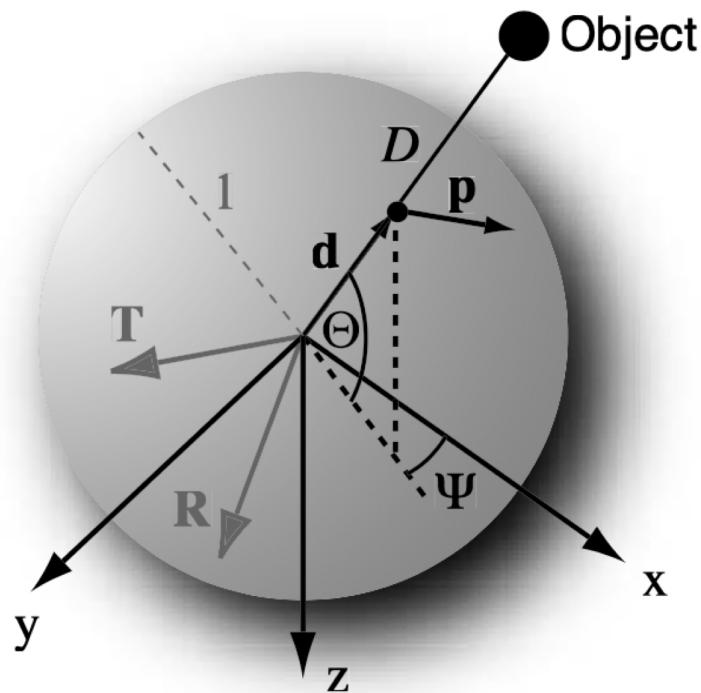
✗ Checkpoints

- nothing left in this course

Insect-inspired vision (week10)

- Insects rely on vision for several flight behaviors
- attitude stabilization; collision avoidance; altitude regulation ..

Optical flow



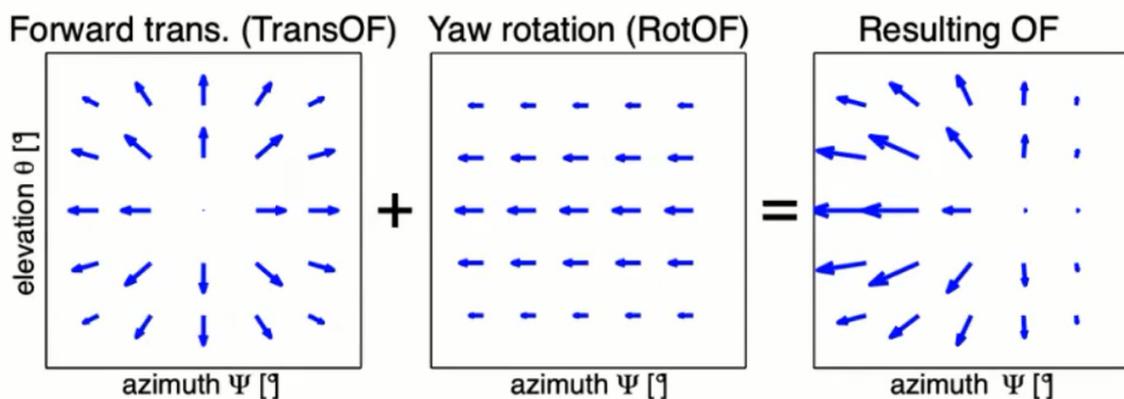
$$\mathbf{p}(\Psi, \Theta) = \left[-\frac{\mathbf{T} - (\mathbf{T} \cdot \mathbf{d}(\Psi, \Theta)) \mathbf{d}(\Psi, \Theta)}{D(\Psi, \Theta)} \right] + [-\mathbf{R} \times \mathbf{d}(\Psi, \Theta)]$$

- Ψ azimuth 方位角 ; Θ elevation 升角
- T/R Translation/Rotation vector
- d/D viewing direction/distance to object
- p result optical flow (tangential to viewing direction)

$$\mathbf{p}(\Psi, \Theta) = \left[-\frac{\mathbf{T} - (\mathbf{T} \cdot \mathbf{d}(\Psi, \Theta)) \mathbf{d}(\Psi, \Theta)}{D(\Psi, \Theta)} \right] + [-\mathbf{R} \times \mathbf{d}(\Psi, \Theta)]$$

The magnitude of this component is inversely proportional to the Distance from the object

The magnitude of this component depends only on the Rotation vector



- translation + rotation component of the optical flow

For pure translational motion

$$p(\Psi) = \frac{\|\mathbf{T}\|}{D(\Psi)} \sin \Psi, \text{ where } p = \|\mathbf{p}\| \quad D(\Psi) = \frac{\|\mathbf{T}\|}{p(\Psi)} \sin \Psi$$

- motion parallax, OF is
 1. directly proportional to the forward speed T
 2. Inversely proportional to Distance D
- Experiments1
 1. landing: the flying speed of bees is decreasing as the height decreases
 2. crossing speed: the flying speed of bees is decreasing as the corridors narrow (distance)
- Experiments2
 1. both vertical stripes: try to balance the OF magnitude in both sides to fly in the center
 2. horizontal vs vertical stripe: low optical filter on the horizontal side because of it it in the same direction as bird moves while more OF near the vertical side (fly to H side to balance)

Sensors used for flight control

1. Compound eyes: a large set of eyes to detect in different directions, no color capability -> used to detect optical flow
 - small viewing angle
 - several small eyes
2. Ocelli: a small set of eyes are sensitive to luminosity to detect contrast; on the head and point upward -> used for stabilization, orientation, and attitude
3. Halteres (like accelerators) -> used to measure rotational speed and stabilize than visual information

Architecture of insect eyes and brains

Optic flow is detected by neurons in the lamina (椎板), whose response is aggregated and transformed by neurons in the medulla (髓质) and in the lobula plate (小叶平台) of brain regions

Elementary Motion Detector 初级运动检测器

- Correlation between two adjacent, time-delayed contrast detectors | 两个相邻的延时对比检测器（小眼）之间的相关性
 - photo receptor -> temporal delay -> correlation -> subtraction-
 - the speed of motion can be detected as the peak the motion
 - **not a reliable** velocity estimator -> depends on temporal and spatial frequency -> cannot measure velocity objectively

Experiment - Optomotor Response 视运动反应

torque response is not related to optical flow speed

Wide-field, motion-specific neurons

- specialized **neurons** that integrate EMD signals from **different regions** and respond only to **specific OF patterns**. 有些特定的神经元只对特定区域特定方向的光流起作用

Optic Flow Computation

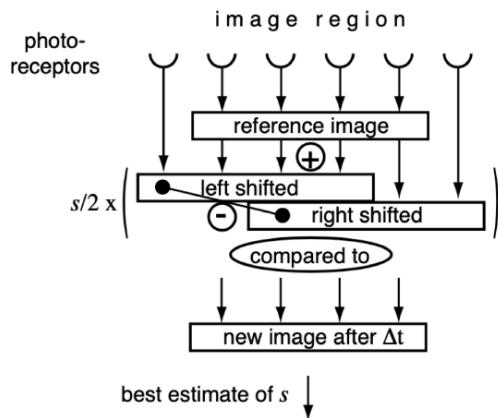
Gradient Descent Methods

$$\frac{dI(n,m,t)}{dt} = 0$$

- assumption: brightness I **does not change across the image** (n,m) as the agent moves **over time t**
- Handcrafted Example: **Lucas-Kanade method**
 1. image smoothing (low-pass filter)
 2. Compute spatiotemporal derivative
 3. Integration of derivatives to produce optic flow vector
- often **iterative** and **requires significant computing power**

Image Interpolation Algorithm –I2A

computed as the image shift s that generates the smallest error between artificially shifted versions of the image at time t and the image at time t + Δt



1. Estimate image at $t + \Delta t$ as combination of shifted versions of image at time t
 $\hat{I}(n, t + \Delta t) = I(n, t) + s \frac{I(n - k, t) - I(n + k, t)}{2k}$

2. Error between estimated and new image

$$E = \sum_n [I(n, t + \Delta t) - \hat{I}(n, t + \Delta t)]^2$$

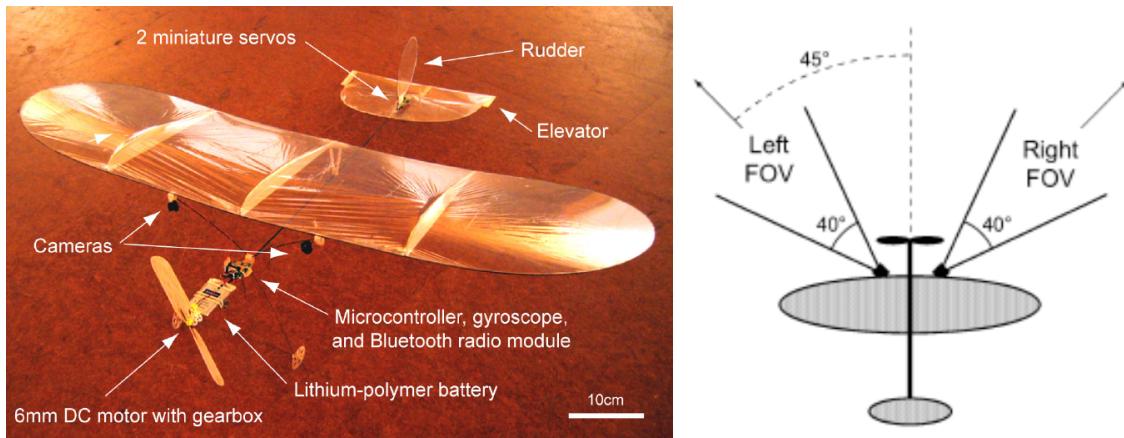
3. Compute image shift s that minimizes error E

$$\frac{dE}{ds} = 0 \Leftrightarrow s = 2k \frac{\sum_n [I(n, t + \Delta t) - I(n, t)][I(n - k, t) - I(n + k, t)]}{\sum_n [I(n - k, t) - I(n + k, t)]^2}.$$

- used in Crazyflie drone for optical flow calculation
 1. moves k pixels in opposite directions to generate different images
 2. calculate errors between real images and generated ones
 3. compute the shift s to minimize error

Obstacle avoidance with I2A

- two cameras to look to left and right with DoF of 40 degrees



- $OFDiv = OFRight - OFLeft$ initiate rotation if $OFDiv > threshold$
- $OFDiff = abs[OFRight] - abs[OFLef]$ rotate towards $OFDiff$
- use cameras (optical flow) and gyro with mechanisms (rudder 方向舵/elevator 升降舵) to control more DoF systems

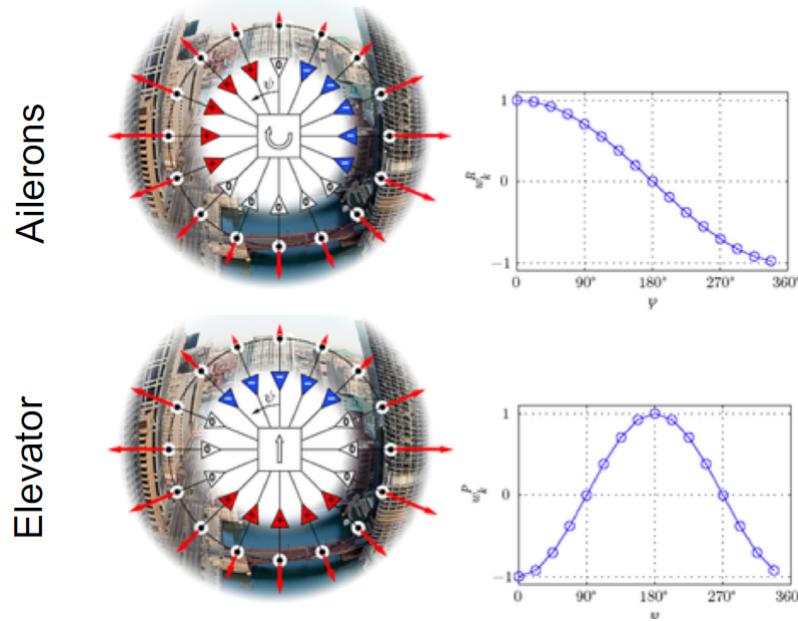
Zufferey, Klaptocz, Beyeler, Nicoud and Floreano (2007), Advanced Robotics

- control pitch and roll of fixed-wing drones by building relation **between optical and control planes directly**

A. Beyeler, J.-C. Zufferey and D. Floreano (2009) Autonomous Robots, 27(3), 201-219

- Ailerons to control roll
- Elevator to control roll

Roll and Pitch weight functions



- use the information from optical flow estimator when the drone closes to the ground
- Making an Artificial Compound Eye

Floreano, Pericet-Camara, Viollet et al, PNAS, 2013

Checkpoints

- Influence of agent's rotation and translation on optic flow and distance estimation
 - rotation flow gives no information about distance; only proportional to the angular velocity of the agent
 - cannot calculate absolute distance if do not know speed
- Influence of angular velocity, spatial frequency, and temporal frequency on EMD (elementry motion decoder)
 -
 - angular velocity
 - spatial frequency
 - temporal frequency
- Functioning of Image Interpolation Algorithm

generate optical flow by computing **image shift s** to minimize the overlap error

- Methods for discounting rotational optic flow
remove the rotational optical flow by using IMU to measure yaw angle

Adaptive Morphology in Flying Animals and Drones (week10)

Bioinspired Mechanical Resilience

How do insects cope with collisions?

- drones hit to obstacle and fall into the ground will lead to damage
- inspired from insects
 1. Sturdy yet **flexible** exoskeleton | 坚固而灵活的外骨骼
 2. Dual stiffness wing | 双刚度翼
can bend with surface

Mintchev, de Rivaz, Floreano, Insect-Inspired Mechanical Resilience for Multicopters, IEEE Robotics & Automation Letters, 2017

- Frames could
 - transit from stiff to soft state
 - use Energy absorbing material

An active uprighting mechanism for flying robots, Klaptocz et al., IEEE Transactions on Robotics, 2012

- Morphology simplifies control - frame could fly against obstacle

Briod, Kornatowski, Zufferey, Floreano, A collision-resilient flying robot, Journal of Field Robotics, 2014

flyability

- separate from inner and outer frame while keeping collision prevention indoors

The Size Problem

Self-deployable origami drone

Mintchev, Daler, L'Eplattanier, Saint_Raymond, Floreano, Foldable and self-deployable pocket sized quadrotor, ICRA, 2015

- more durable when collides with obstacle

Origami Drone Wing

Dufour, Owen, Mintchev, Floreano, A drone with insect-inspired folding wings, IROS 2016

Adaptive Morphology

- fixed-wing: only fly upper in the sky cannot stop
- quadrotor: less efficient and short range

Daler, Lecoeur, Hählen, Floreano, A flying robot with adaptive morphology for multi-modal locomotion, IROS Proceedings, 2013

A bioinspired multi-modal flying and walking robot, Bioinspiration & Biomimetics, 2015

- can use wing to fly and rotate wing to walk on the ground

Air and Ground Locomotion

Ajanic, Feroshkan, Mintchev, Noca, & Floreano (2020) Science Robotics

- Morphing Wings to adapt different winds
- LIS hawk inspired from northern goshawk

A

	aggressive flight	cruise flight
n. goshawk		
LisHawk		
	northern goshawk	LisHawk
wing span (<i>a</i>) (m)	0.89 - 1.22	1.05
tail length (<i>b</i>) (m)	0.20 - 0.27	0.24
wing chord (<i>c</i>) (m)	0.28 - 0.35	0.32

- Maneuverability

able to change the velocity vector by changing the wing status

higher linear accelerations on body frame/higher lift/drag coefficient

- Agility

able to change angular rate, e.g. pitch and roll angles

Higher angular rates are produced by larger pitch and Agility roll coefficients by **extending the wing** or not

- Power requirement and speed range

Extending wing and tail increases lift and allows lower speed range

Checkpoints

- Strategies for collision resilience in flying animals and robots

- inserts

- Sturdy yet **flexible** exoskeleton | 坚固而灵活的外骨骼
 - Dual stiffness wing | 双刚度翼

- Robots

- dual-stiffness frame and energy-absorbing material
 - Use morphology when against obstacle

- Design flexible protective frame
- Trade-offs between aerial and ground locomotion that require adaptation
 can use wing to fly and rotate wing to walk on the ground
- Effects of wing and tail span on lift and drag coefficients, and on required power
 - | *Extending for aggressive flight while Tucking wing (折翼) for cruise flight*
 - **Tucking** -> reduces power requirement at high speeds
 - **Extending** -> increases lift;

Agile Flight (week11)

- topics: Perception, Learning, and Control
- Why agile? flying robots to search & rescue

Pfeiffer, Scaramuzza (2021) Human-piloted drone racing: Perception and control, RAL'21. PDF. Dataset.

- Humans focus visual attention on **future waypoints**: receding planning horizon
- 220 ms **perception-control latency** → humans **can definitely be beat** by a machine in a speed race
- try to plot literature in different axis (External sensors & agile)
- Research challenges & Opportunities
 - Autonomous drone racing
 - Drone acrobatics
 - Low-latency sensing

Autonomous drone racing

P. Foehn et al., AlphaPilot: Autonomous Drone Racing, RSS 2020, Best System Paper Award. PDF YouTube

- pass a sequence of gates as soon as possible
 - time-optimized trajectory
 - execute the trajectory while being robust to disturbances

[1] Foehn, Scaramuzza, CPC: Complementary Progress Constraints for Time-Optimal Quadrotor Trajectories, arXiv preprint, 2020. PDF. Video.

- tight coupling of perception and action necessary?
- plan **perception-aware** planning and control

Falanga, Foehn, Peng, Scaramuzza, PAMPC: Perception-Aware Model Predictive Control, IROS18. PDF. Video. Open Source: https://github.com/uzh-rpg/rpg_quadrotor_mpc

- action objectives + perception objectives => optimization problem
 - perception: maximize visibility of POI (minimize the deviation) + minimize the blur (minimize rotation speed)

Kaufmann et al., Beauty and the Beast: Optimal Methods Meet Learning for Drone Racing, ICRA'19. PDF. Video Deployed to win the IROS Autonomous Drone Racing Competition, 2018. Video.

- NN to detect the gate (robust to motion blur and illumination variation)

Loquercio, et al., Deep Drone Racing: From Simulation to Reality with Domain Randomization

- predict gate pose and covariance
 - trained in simulation only
 - sim-to-real world transfer via domain randomization!!!
 - can be used to drone approach
 - gazebo simulator was used
- ACADO (<http://acado.github.io/>). ACADO's C++ interface is used to describe the quadrotor model and parameters for transcription into a quadratic program, which is then solved with qpOASES (<https://projects.coin-or.org/qpOASES>).

Drone acrobatics

Kaufmann, Loquercio, Ranftl, Mueller, Koltun, Scaramuzza, Deep Drone Acrobatics, RSS 2020. PDF. Video. Code Best Paper Award Honorable Mention

- traditional drone control architecture
 - Image/IMU -> State estimation (VIO) -> Planning -> Control -> Collective thrust/Bodyrates -> Low-level controllers

- End-to-end sensorymotor control (in a concurrent design fashion)

Image/IMU -> NN -> Collective thrust/Bodyrates

- Zero-shot Sim2Real Transfer
- related literature

| Does computer vision matter for action? Science Robotics

Controller experiments indicate that explicit intermediate representations help action

- use **Feature tracks** into the NN tracking
- Simulation results: lower tracking errors and 0 errors in acrobatics
- Cons: does not generalize to different tasks

Learning-based controller

- Robust to imperfect perception
- Low-Latency
- Does not generalize to different tasks
- Difficult to interpret

Traditional controller

- Sensitive to imperfect perception
- High-Latency
- High generalization: same controller for all maneuvers
- Transparent and easy to interpret

Low-latency sensing

- event cameras do not suffer from latency/motion blur or illumination variation.
- Application
 - VIO in high-speed and HDR environment
 - Dynamic Obstacle Avoidance

Learning of flight controllers (week11)

- tightly coupled vision and control

challenge1: Architecture/Input and Output representation

- processing **asynchronous** data (different frequencies): sampler of IMU info and trajectory, feature tracks -> temporal convolutions -> multilayer perceptron

challenge2: Data collection

1. imitate and use data from cars and bicycles
 - already integrated into city environments
 - large publicly available datasets
 - no need for an expert drone pilot

DroNet: Learning to Fly by Driving, Loquercio et al., Robotics and Automation Letters 2018, PDF, Video, Code.

- fair performance outdoors; Generalization in Indoor corridors and parking lots
- Analysis on the DroNet -> Line-like feature are a strong cue of direction; Cannot work in forest-like environment/water (state estimation)

2. Simulation to Reality Transfer -> Sim2Real: Domain Randomization for Drone Racing

Deep Drone Racing with Domain Randomization

Loquercio et al., Agile Autonomy: Learning High-Speed Flight in the Wild, Under Review

- Transfer learning: Abstraction
 - use abstraction function to make Sim2Real more similar

Transfer Learning: Abstraction

- The performance gap for a finite-horizon controller directly depends on the covariate shift introduced by the perception/action models of the two domains.

$$J(\pi_r) - J(\pi_s) \leq C_{\pi_s} K \mathbb{E}_{\rho(\pi_r)} [DW(M, L)]$$

- We propose to use an abstraction function to reduce the gap between models.
This function is task and sensor dependent.

$$DW(f(M), f(L)) \leq DW(M, L)$$



- How to find the abstraction function



- Learn them end-to-end together with the downstream task (as in drone racing).
➤ They are automatically optimal for the task.
➤ They require strong engineering of the simulators for effective domain randomization.
- They can be pre-defined using domain knowledge (as in drone acrobatics).
➤ They strongly favor the training generalization.
➤ They introduce human biases and could potentially be non-optimal for the task.

- Imitation learning by using current SOTA method (RRT, A*, MPC, LQR) as GT

Limitations of Imitation Learning -> RL

- We don't yet know how to solve tasks like interaction with other agents, dynamic obstacle avoidance, or complex object manipulation.
- The trained policy does not improve in time.

RL limitations

- Design a reward function, then learn via interaction
difficult to design the reward function
- state space
vision-based state space is huge -> hard to guarantee enough coverage of the state space

challenge3: Guarantee the platform's safety during training and testing

- difficult to interpret -> **measuring uncertainty**

Loquercio et al., A General Framework for Uncertainty Estimation in Deep Learning, RA-L 2020

- uncertainty: data uncertainty & model uncertainty
- Demonstrator: future motion prediction; closed loop control of a Quadrotor

Takeaways

1. Car-Driving Datasets or Simulation can be used to train deep navigation systems.
2. Transfer knowledge via input and output abstractions.
3. Measuring the networks' uncertainty is necessary to guarantee safety.