Contents

MICRO-502 Aerial Robotics Notes

Lecture notes by Yujie He

Last updated on 2021/07/02

All checkpoints summary can be found here!

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

Fixed wing Staying in the Air

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

fixed-wing generates 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

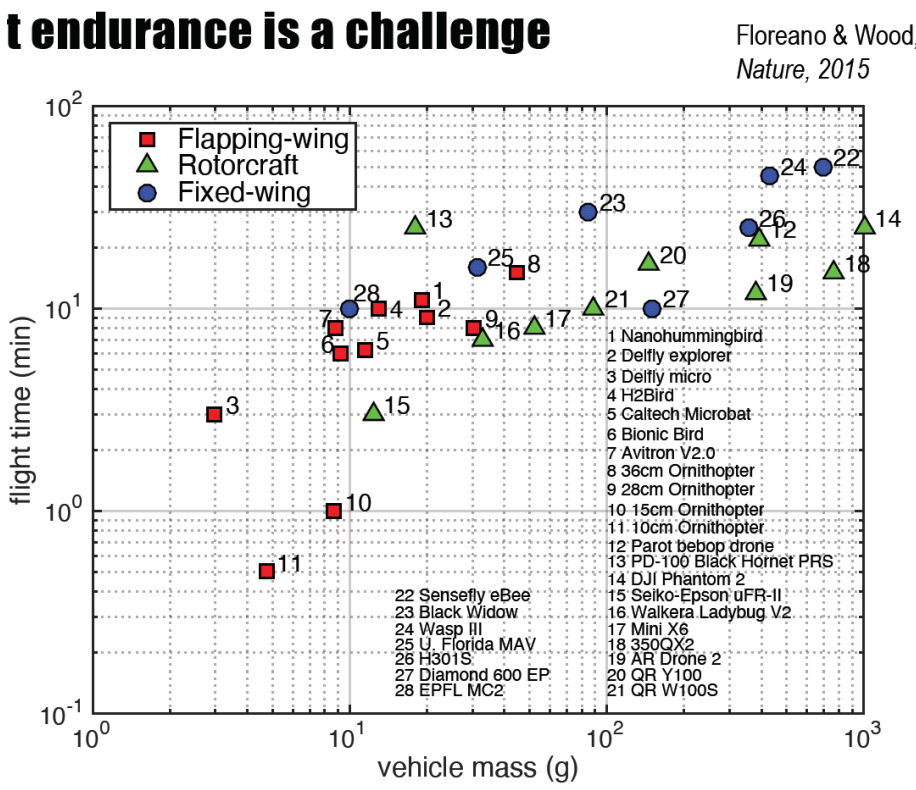


Figure 1: week1_endurance_mass

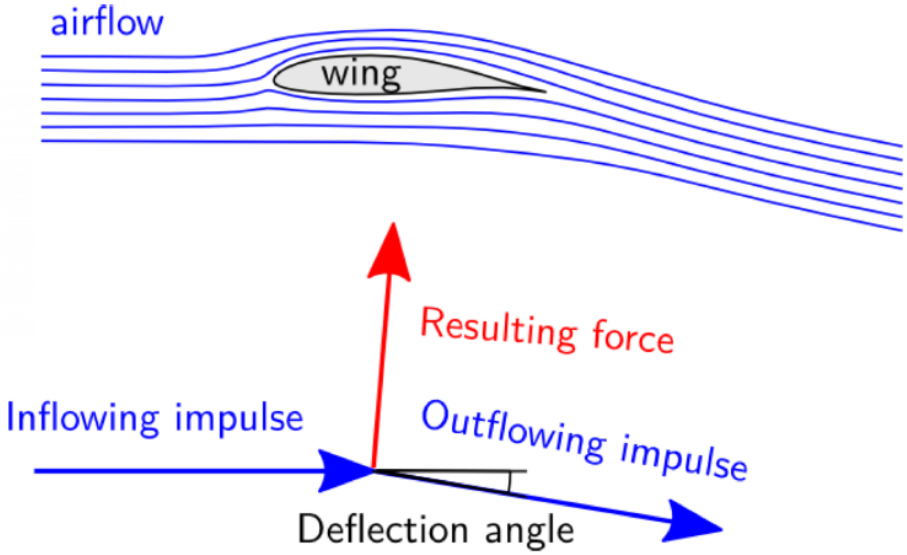


Figure 2: week1_hovering

  - Air speed V: fly 2x as fast -> 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 $r = Lift/Drag$

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 for medicals

⍰ 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: $\rho = 1.25 kg/m^3$
  - intercontinental airplane fly ~10000m: $\rho = 0.4135 kg/m^3$

  $V \propto \sqrt{1/\rho}$

- 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~~

  - wing angle -> lift coefficient
  - 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
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)

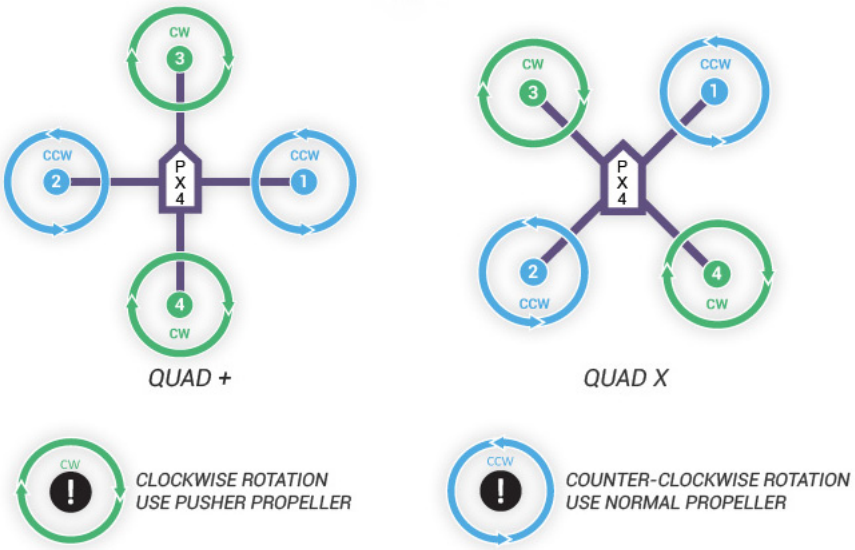


Figure 3: week1_frame

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 $M_i = L \propto F_i$

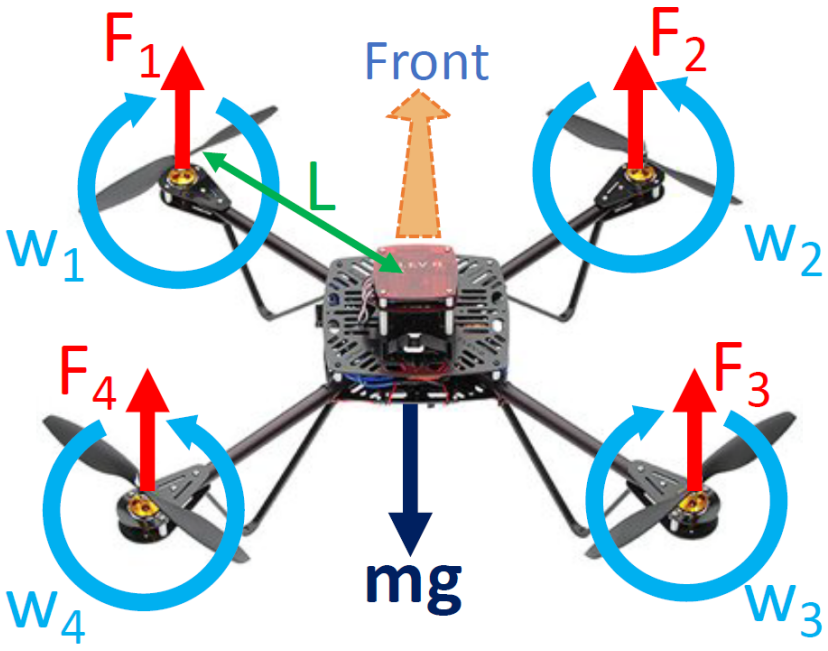


Figure 4: week1_force

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 $F_i \| g$
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) $(w1 + w3) - (w2 + w4) = 0$

   yaw

Flight mechanics

How to move a quadrotor around?

- **Orientation**

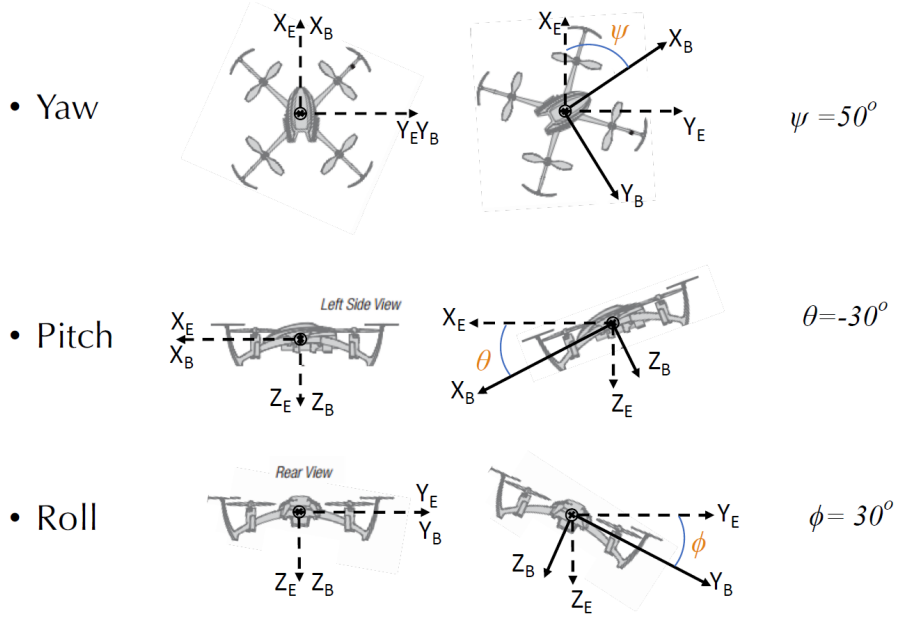


Figure 5: week1_rpy

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

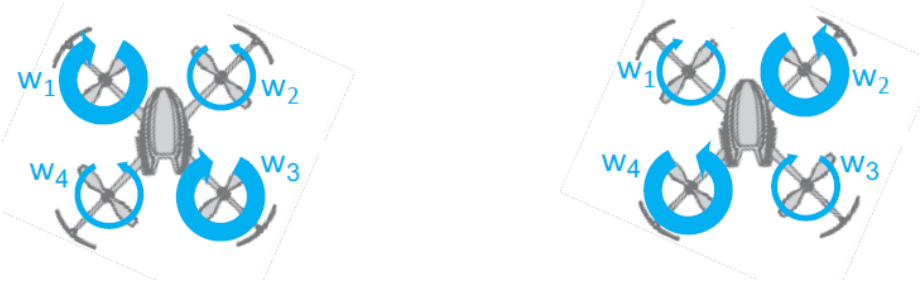


Figure 6: week1_rollpitch

- Rotor speeds not balanced (torque balanced) $(w1 + w3) - (w2 + w4) \neq 0$

  $\dot{\psi} = k_{\psi}\left((w_2 + w_4) - (w_1 + w_3)\right)$

- 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 $M_1 + M_2 + M_3 + M_4 \neq 0$

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

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

Summary of equations

- 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

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

$$\dot{\phi} = k\,((w_1+w_4)-(w_2+w_3)) = k\,w_1-kw_2-kw_3+kw_4$$
$$\dot{\theta} = k\,((w_1+w_2)-(w_3+w_4)) = k\,w_1+kw_2-kw_3-kw_4$$
$$\dot{\psi} = k\,((w_2+w_4)-(w_1+w_3)) = -k\,w_1+kw_2-kw_3+kw_4$$
$$T = k\,((w_1+w_3+w_2+w_4)) = k\,w_1+kw_2+kw_3+kw_4$$

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.

Figure 7: week1_speed2motion

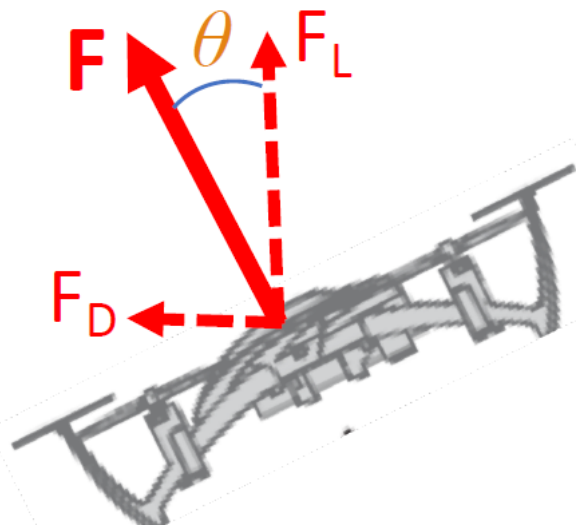


Figure 8: week1_move

Types of Multicopters

main feature; fully actuated multicopters

Configuration

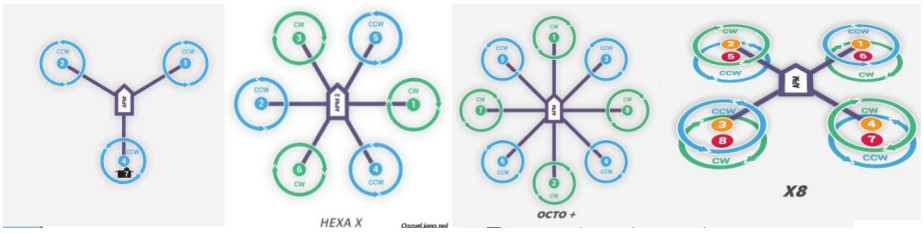


Figure 9: week1_config

- 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/Octocopter
  - 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 → Rotor power
T_r → Rotor Thrust
η_r → Figure of Merit (propeller efficiency)
r_p → Propeller radius
ρ_a → Air density

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

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

Figure 10: week1_energetics

- 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 squire 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

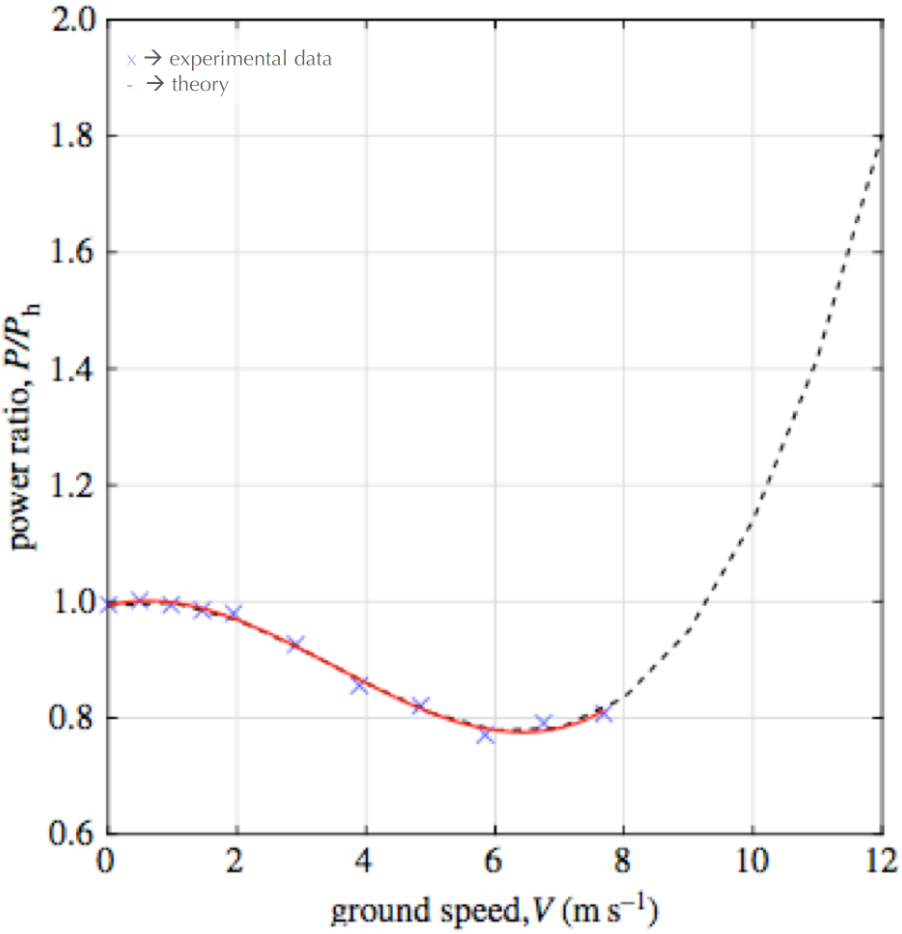


Figure 11: week1_translation_flight

  - 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?
  1. Weight and drag reduction
  2. Increase the specific power of the energy source
  3. Docking station for charging/battery swapping
  4. Tether for power supply
  5. Improving efficiency via mechanical
  6. Energy aware motion planning

7. Multi-modal operation

- 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)

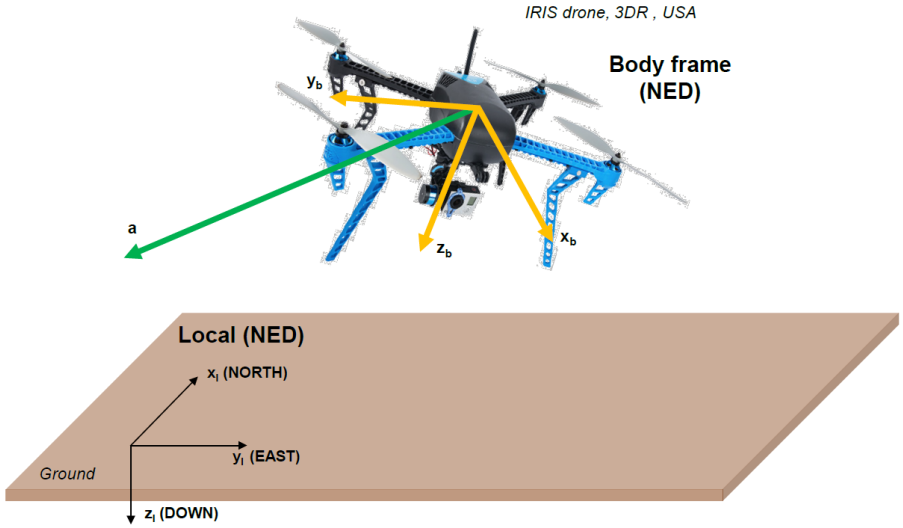


Figure 12: week2_frame

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

3D Attitude representation- Euler Angles

- Rotation matrices can be parametrize 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)

  can be summarized as $R = R_\phi R_\theta R_\psi$ （左乗）
- ? Claw Example

  rotation + translation

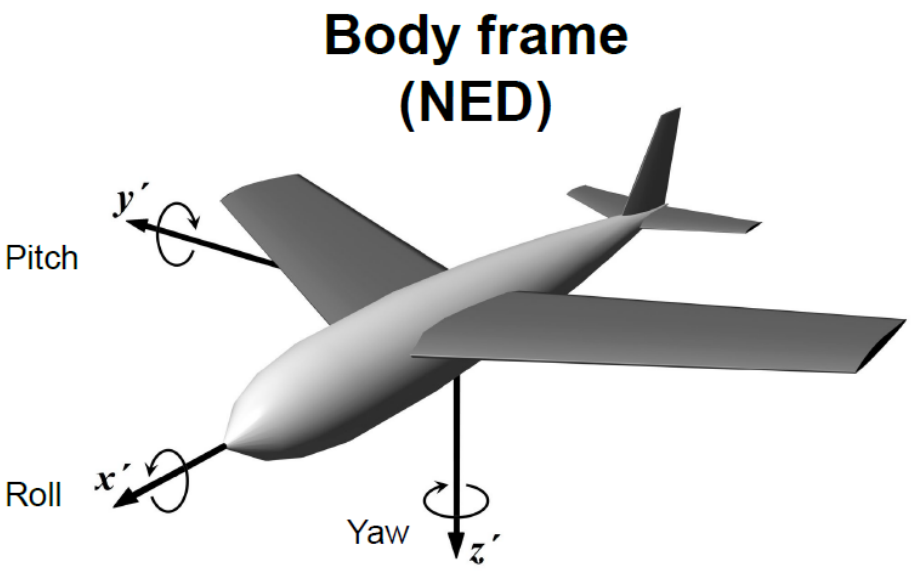
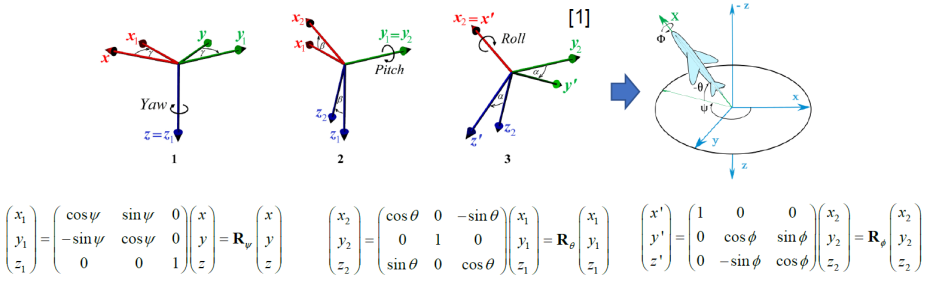


Figure 13: week2_euler



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

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

Figure 14: week2_euler_ypr

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

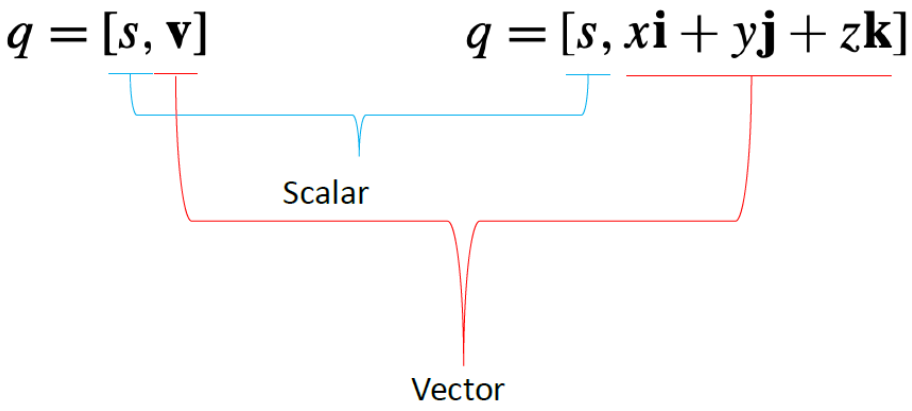


Figure 15: week2_quaternion

- rotation vector = quet * vector * quet^(-1)
  - $p = [0, \mathbf{p}]$ vector in 3-space
  - $q = [s, \lambda\hat{\mathbf{v}}]$ Must meet these requirements

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

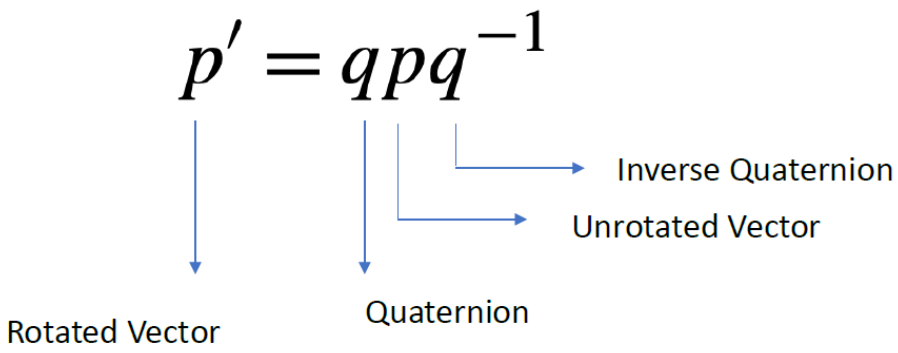


Figure 16: week2_rotata_quet

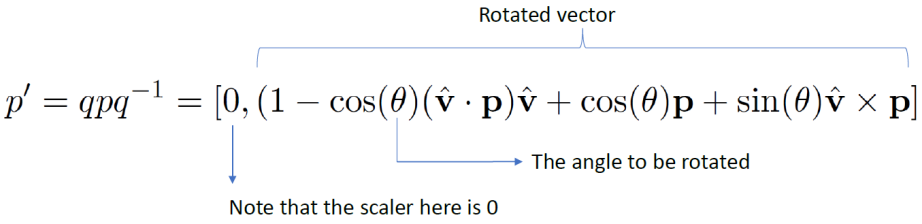


Figure 17: week2_euler_mul

- multiplication

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

  week2*quaternion*mat.png

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

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

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

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

For multiple rotations: $q_2\left(q_1 p q_1^{-1}\right)q_2^{-1} = (q_2 q_1)p(q_2 q_1)^{-1}$

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

Figure 18: week2_quaternion_mat

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

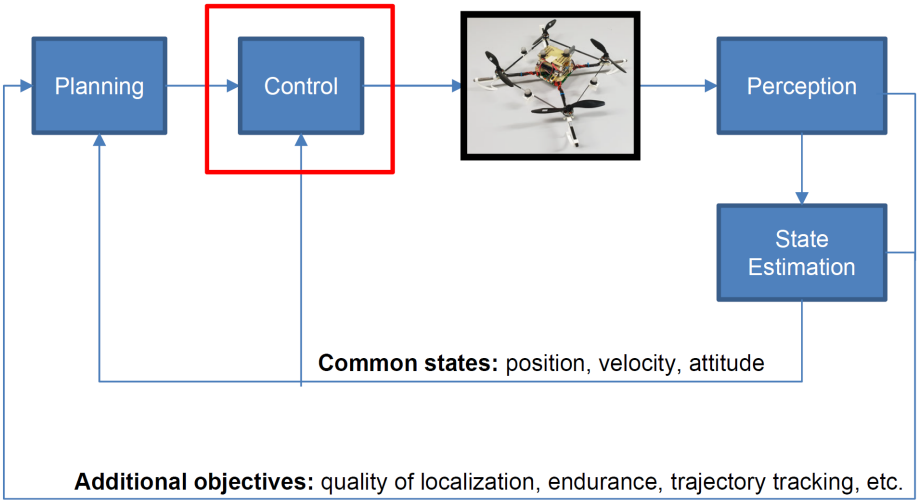


Figure 19: week2_control_diagram

- 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

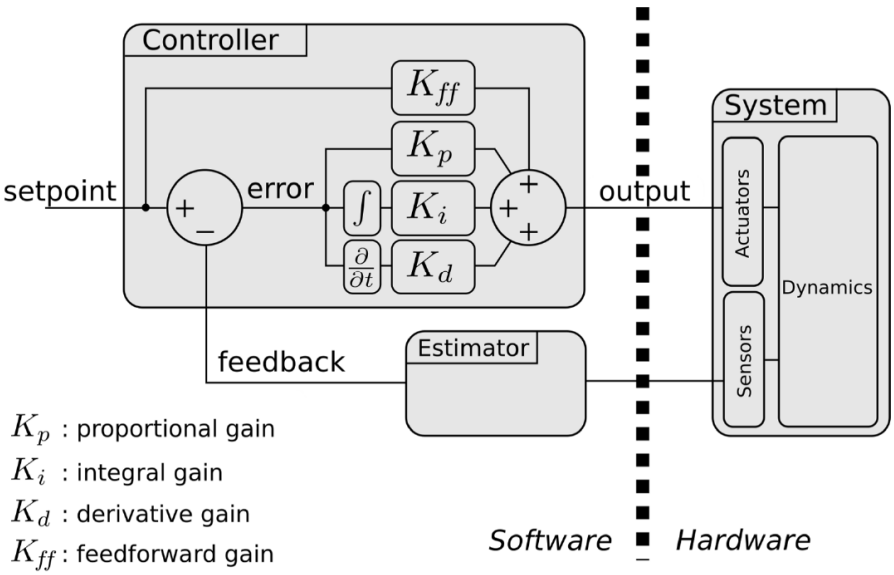


Figure 20: week2_pid

- 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?

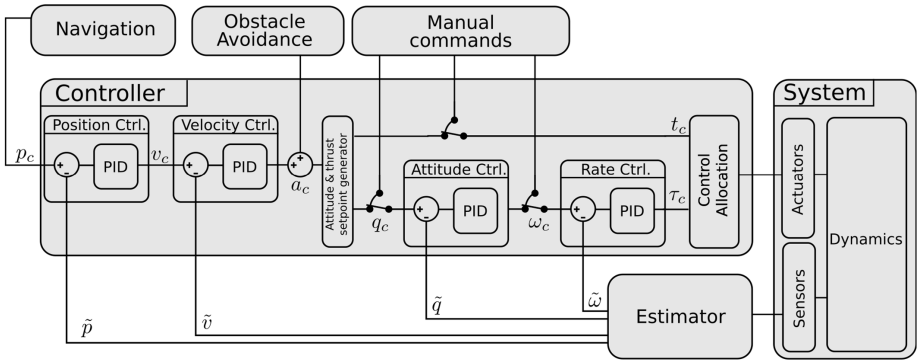


Figure 21: week2_cascaded_control

- 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**

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

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

t : thrust setpoint (1×1)

Method

1. Compute generated force and torque of each actuator
2. Build matrix "Actuator **Effectiveness**"
3. 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

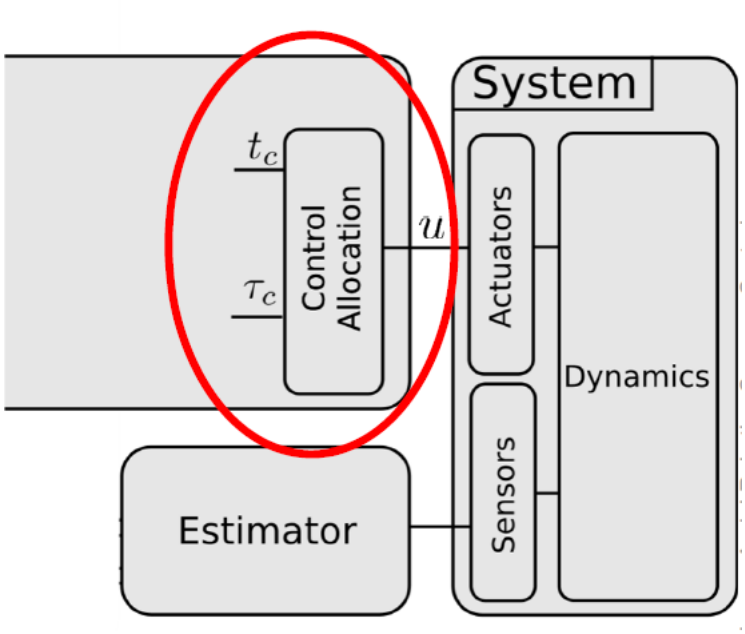


Figure 22: week2_control_allocation

1. Actuator torque and force:

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

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

2. Actuator effectiveness matrix:

$$\begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ t_z \end{bmatrix} = A \begin{bmatrix} u_0 \\ \vdots \\ u_{N-1} \end{bmatrix} \qquad A = \begin{bmatrix} \begin{bmatrix} \vec{\tau_0} \\ t_0 \end{bmatrix} & \cdots & \begin{bmatrix} \vec{\tau_i} \\ t_i \end{bmatrix} & \cdots & \begin{bmatrix} \vec{\tau_{N-1}} \\ t_{N-1} \end{bmatrix} \end{bmatrix}$$

3. Control allocation matrix:

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

Figure 23: week2_allocation_method

- yaw torque requires more actuator effort than roll and pitch torque

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

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

Figure 24: week2_control_allocation_example

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

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

- axis error

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

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

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

    $\text{Conj}(\boldsymbol{q}) = \boldsymbol{q}^* = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T$
  - if the length of the quaternion is unitary then the **inverse is the same as its conjugate** 逆即是复共轭

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

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

    where **v** can be rewritten as location in x, y and z axis. For example,

    $R_x(\boldsymbol{q}) = \boldsymbol{q} \otimes \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \otimes \boldsymbol{q}^* = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 \\ 2\left(q_1 q_2 + q_0 q_3\right) \\ 2\left(q_1 q_3 - q_0 q_2\right) \end{bmatrix}$
  - u is the rotation axis (unit vector) and α is the angle of rotation

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

    $\boldsymbol{q}_{err} = \boldsymbol{q}_{\boldsymbol{ref}} \otimes \boldsymbol{q}_m^*$
- 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 q0. If q0 < 0 then the desired orientation is more than π radians away

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

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

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

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

Figure 25: week3_pid_effect

  - Attitude control (providing **torques**) requires adaptation since morphology has an impact on the rotation dynamics
  - **should update matrix matrix A during flight**
- 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

    easier controller; don't need additional motors; lightweight; cannot achieve optimal performance

  - Variable-tilt: change tilt of propellers to achieve desired motion

    can achieve optimal configuration in given task 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 x (3torque + 1thrust)

    最底层的控制，就是输入是 torque+thrust，输出是 n 个电机的转速
  - Effectiveness：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: angle rate setpoint + estimated angle rate

  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: desired quaternion setpoint **+ estimated** quaternion

  Output: generated angle rotation rate

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

  1. compute quaternion error

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

  2. calculate angular errors

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

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**

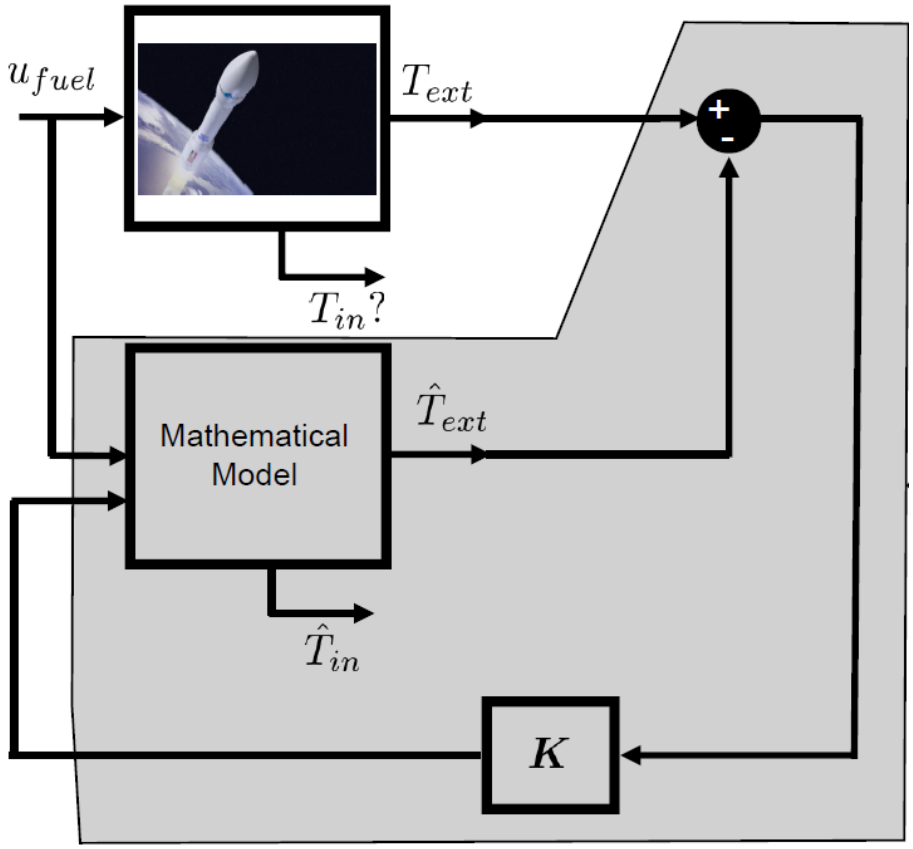
  Height; **Attitude; Angular velocity** (Most important as they are the **primary variables used for the stabilization** of the UAV); linear velocity

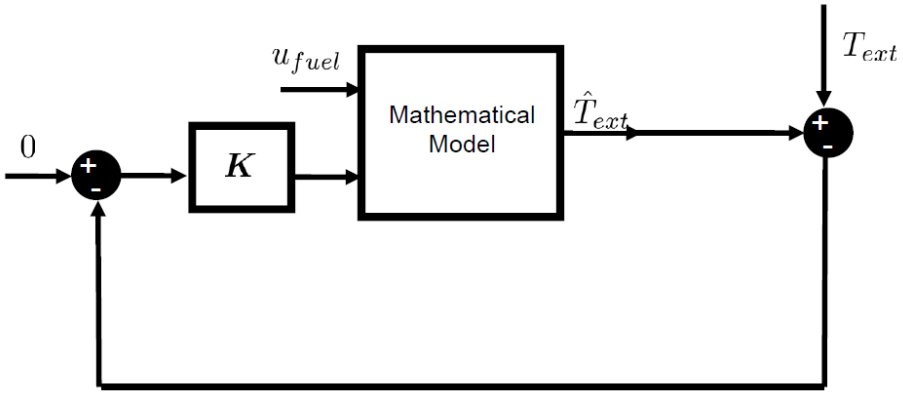
  - main sensors
    - **Inertial** Measurement Unit (IMU)
    - **Height** Measurement (Acoustic, infrared, barometric, laser based)
    - Others: Global Navigation Satellite System (GNSS, outdoor) or VICON (indoor); Camera, Kinect, LIDAR

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

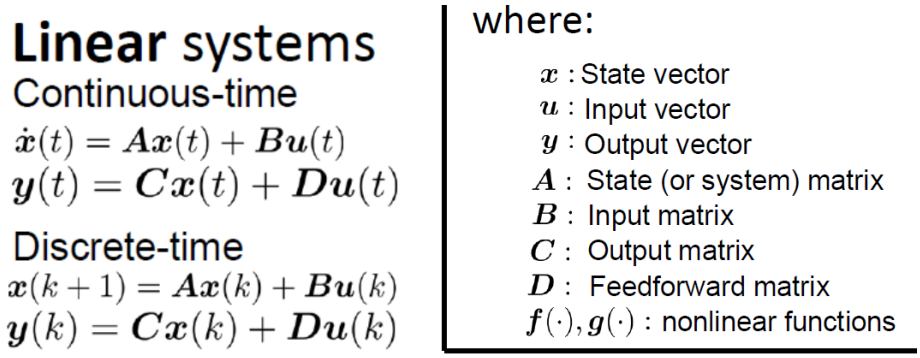


Figure 26: week3_dynamical_system

Estimation error

- observer error

  $\dot{e}_{obs} = (\boldsymbol{A} - \boldsymbol{KC})\boldsymbol{e}_{obs} \rightarrow 0$ => $e_{obs}(t) = e^{(\boldsymbol{A}-\boldsymbol{KC})t} e_{obs}\,(t_0)$

  - with linear model

    $\dot{x} = Ax + Bu$

    $\dot{\hat{x}} = A\hat{x} + Bu + K(y - \hat{y})$

    $e_{obs} = x - \hat{x}$

- if A-KC <0, the error will lead to 0 when t -> infinity
- How to choose K?
  - gives control on the decay rate of the error function
  - similar to P gain in control (if too big, estimation will oscillate)

State Estimation for stochastic systems

Recap of fundamental concepts

- Mean and variance encode the **a-priori knowledge** of the random variable
  - w: Process noise
  - v: Measurement noise

$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{w}\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} + v$

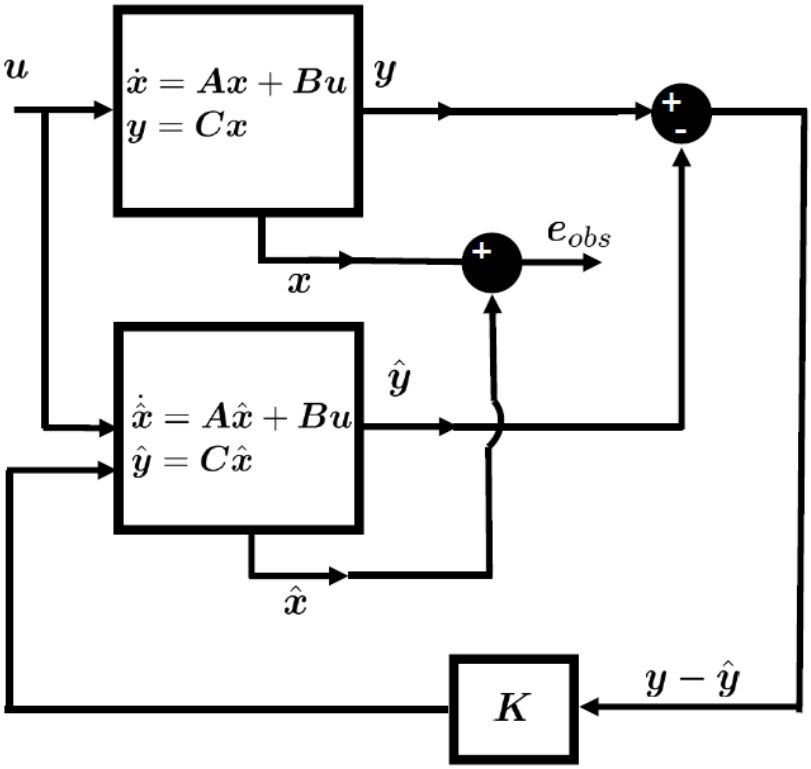


Figure 27: week4_observer

- What is the **a-priori information** about uncertainty in our problems?

  W/V are known but w/v are zero-mean

  - W: **covariance matrix** associated to **process** noise
  - V: c**ovariance matrix** associated to **measurement** noise

- What is the statistical **independence** among the values of a random variable in time?

  $E\left[\boldsymbol{w}(k)\boldsymbol{w}^T(k+n)\right] = 0 \quad \forall n > 0$ $E\left[\boldsymbol{v}(k)\boldsymbol{v}^T(k+n)\right] = 0 \quad \forall n > 0 +$ $E[\boldsymbol{w}(k)] = E[\boldsymbol{v}(k)] = 0$

  white noise stochastic processes -> 期望值都是 0

Kalman filter (KF)

- example: estimate drone's roll angle
  - x: Drone's roll angle
  - u: Roll Body rate
  - y: Drone's roll angle (through IMU)
  - $w \sim N(0, Q)$: process error **(only in the dynamics not in estimator!)**
  - $v \sim N(0, R)$: measurement error **(only in the dynamics not in estimator!)**

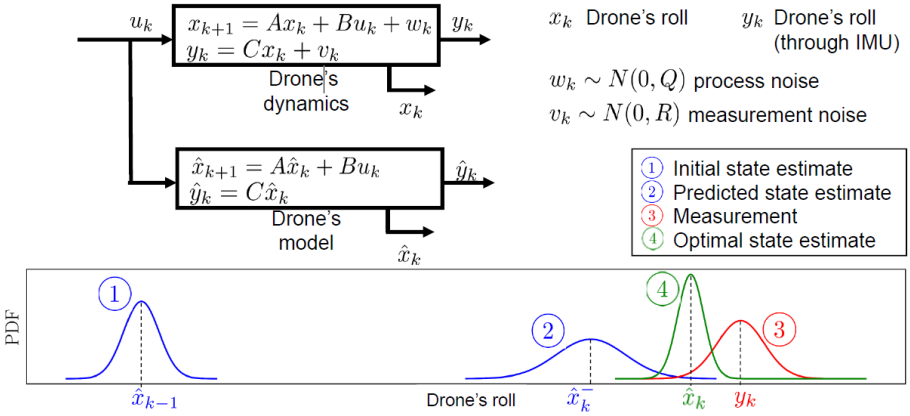


Figure 28: week5_kf

  - initial state estimate
  - -> predicted state estimate
  - + measurement
  - => Optimal state estimate

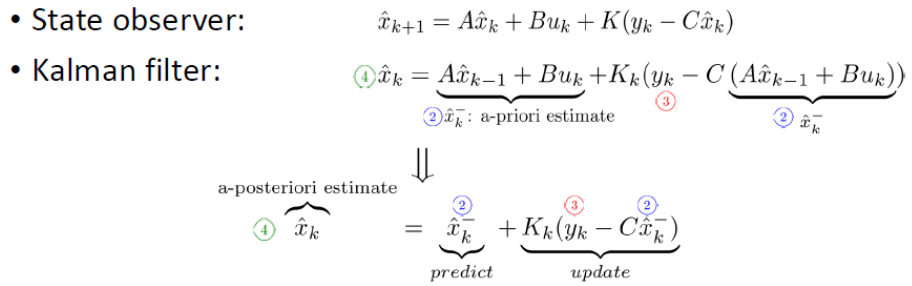


Figure 29: week4_kf_equ

Prediction (A-Priori) + Update

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

PREDICTION

- state $\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
- error covariance $P_k^- = AP_{k-1}A^T + Q$

  keep track not only the state but also the covariance (will keep increase)

UPDATE

- KF Gain $K_k = \frac{P_k^- C^T}{CP_k^- C^T + R}$
  - trust more in model y_k

    R-> 0 => K*k* = *1/C* => *x*k = y_k/C
  - trust more in measurement x_k

    P*k* -> *0* =>*K*k = 0 =>x*k* = *x*k^{_}
- Update state $\hat{x}_k = \hat{x}_k^- + K_k\left(y_k - C\hat{x}_k^-\right)$
- $P_k = (I - K_k C)\, P_K^-$

  minimize the Covariance

sensor fusion on a quadrotor

use multiply KF to achieve better estimation

$\hat{x}_{k_{1\times 1}} = \hat{x}_{k_{1\times 1}}^- + K_{k_{1\times n}}\left(y_{k_{n\times 1}} - C_{n\times 1}\hat{x}_{k_{1\times 1}}^-\right)$

Intro to EKF, UKF and particle filters

EKF

- **linearizes** the nonlinear function **around the mean of the current state estimate**

Overview

- **Nonlinear system:** $\boldsymbol{x}_k = f(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) + \boldsymbol{w}_k \quad \boldsymbol{w}_k \sim \mathcal{N}(0, \mathbf{Q})$
  $\boldsymbol{y}_k = h(\boldsymbol{x}_k) + \boldsymbol{v}_k \quad \boldsymbol{v}_k \sim \mathcal{N}(0, \boldsymbol{R})$
- **Computation of the Jacobian matrices:** $\boldsymbol{F} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k} \quad \boldsymbol{H} = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_k}$
- **Linearized system:** $\boldsymbol{x}_k \approx \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{w}_k$
  $\boldsymbol{y}_k \approx \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k$

Figure 30: week4_ekf_equ

- applied to nonlinear system
- linearize around the state estimate value by computing Jacobian matrices

Equations

- Predication

  $\hat{\boldsymbol{x}}_k^- = f\left(\hat{\boldsymbol{x}}_{k-1}^-, \boldsymbol{u}_k\right)$

  $P_k^- = FP_{k-1}F^T + Q$

  F-Computed Jacobian matrix

  $\boldsymbol{F} = \left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1}, \boldsymbol{u}_k}$

- Update

  Near optimal Kalman Filter gain

  $K_k = \frac{P_k^- H^T}{HP_k^- H^T + R_k}$

UKF

- take several points to compute/approximate probability distribution
- choose sigma points according to algorithms

Particle filters

- Combines the results to estimate the mean and covariance of the state
- !computationally intensive
- Different UKF
  - Not limited to a Gaussian distribution
  - points are chosen randomly

State Estimator	Model	Assumed distribution	Computational cost
Kalman Filter (KF)	Linear	Gaussian	Low
Extended Kalman Filter (EKF)	Locally linear	Gaussian	Low (for analytically-computed Jacobians) Medium (for numerically-computed Jacobians)
Unscented Kalman Filter (UKF)	Nonlinear	Gaussian	Medium
Particle filter	Nonlinear	Non-Gaussian	High

Figure 31: week5_state_estimation

State Estimation in aerial robotics

in lab settings

- use MoCap to estimate position and velocity

Outdoor

- Camera, Depth camera Sensor, ToF Range Sensor to estimate pose and height

⍰ Checkpoints

- What structure does the Luenberger Observer remind you?

  closed-loop estimation for deterministic systems
- What does the Kalman Filter (KF) gain do?

  determine to trust either more on measurement or a-prior/model
- For which kind of dynamical systems the KF is useful?

  linear continuous system

  !cannot work in discontinuous system

- What do we need to know a-priori if we want to apply the KF?

  state mean x_k and error covariance P_k

- What are the advantages/disadvantages of the **UKF** w.r.t an **EKF**?

  EKF

  - **need** linearization
  - Not an optimal estimator (especially nonlinear)
  - may quickly diverge (when incorrect model/wrong covariance init)
  - may difficult to compute Jacobians

- What are the advantages/disadvantages of a **particle filter** w.r.t an **UKF**?

  PF

  - Not limited to a Gaussian distribution
  - need more points → more computational time
  - More accurate as long as having enough particles

□? Navigation (week5)

Velocity control

- □? 如果低速飞行，不去调整 yaw，不然可能会引起震荡!

Waypoint Navigation

Dubins Paths

Vector fields

□? Checkpoints

- Which component of the **acceleration setpoint is converted to roll angle?** To **pitch** angle?
  - roll
  - pitch
- Why is navigation based on fixed position setpoint inappropriate for fixed-wing drones?
  - cannot do sharp turning
- What type of segment compose a Dubins path?
  - arcs of a minimal radius

  - straight lines
- In which case can we construct a RLR Dubins path?

  distance of waypoints is less than the minimal diameter
- What happens when **centrifugal acceleration** is not taken into account in the formulation of a circle following vector field?

  It will not follow the vectors and cross the circle lines

? VIO & SLAM (week5)

VIO

SLAM

? Checkpoints

- What is VIO/SLAM useful for?
  - VIO: Visual inertial odometry

    process of estimating the drone state (position, orientation, and velocity) by using only the inputs from Camera(s) and IMU(s)
  - SLAM: Simultaneous Localization and Mapping
- What are the sensors that can be used to do VIO/SLAM?

  Camera(s) and IMU(s)
- What is loop closure?

  the **recognition** of when the robot has returned to a **previously mapped region** and the use of this information to **reduce the uncertainty in the map estimate.**
- What are the main three VIO paradigms?
  - Filters
    - Restrict inference process to **latest state of the system**
  - Fixed-lag smoothers
    - Restrict inference process **to a given time window**
  - Full smoothers
    - Estimate the **entire history of the states**
    - More accurate
    - Real-time operation can become unfeasible
- Why do we use a camera and an IMU for VIO?

  - **Cheap and complementary to each other**
  - Camera: **Precise** during **slow motion** and provide **rich information**

    Limited output rate; Not robust to scenes with low texture, high speed motions
  - IMU: **Scene independent** and **High** output **rate**

    Poor signal-to-noise ratio at low accelerations and angular velocities; Estimated motion tends to accumulate drift quickly

Fixed-wing drones (week6)

Introduction

Structure

Flight Mechanics

Control surface

- Ailerons
  - attached to the outboard trailing edge of both wings
  - rotate the aircraft around the longitudinal axis (**roll**)
- elevators
  - hinged to the trailing edge of the horizontal stabilizer
  - around the horizontal or lateral axis (pitch)
- rudder
  - hinged to the trailing edge of the vertical stabilizer
  - causes an aircraft to yaw or move about the vertical axis

Stability

longitudinal -> pitching

lateral -> rolling

directional -> yawing

Energetics

Induced power

- **Induced** drag is the drag on the wing that is due to lift

Profile and parasite power

- The **total aerodynamic power** is obtained by multiplying the **drag force** by the **forward velocity**

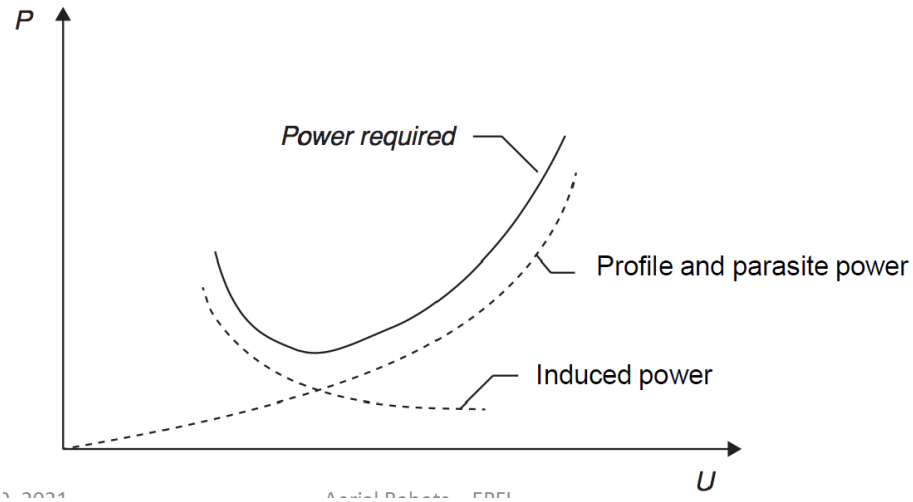


Figure 32: week6_power_fixed

❓ Checkpoints

- What are the main flight control surfaces in fixed-wing aircrafts?
  - Ailerons
    - attached to the outboard trailing edge of both wings
    - rotate the aircraft around the longitudinal axis (**roll**)
  - elevators
    - hinged to the trailing edge of the horizontal stabilizer
    - around the horizontal or lateral axis (pitch)
  - rudder
    - hinged to the trailing edge of the vertical stabilizer
    - causes an aircraft to yaw or move about the vertical axis
- What are the **main stabilization strategies** in fixed-wing aircrafts?
  - Longitudinal stability

    making CoG ahead of the aircraft

  - Lateral stable
    - Dihedral angle 翅膀上扬
    - Keel effect and Weight distribution
  - Vertical stability-pitch

    making the side surface greater than ahead of the center of gravity
- How is a flying wing/Tailless aircraft controlled and stabilized?
  - upward reflex (上扬边缘) or elevons (ailerons+elevator) keeps pitch in equilibrium
- What are the **main contributions to power consumption** in fixed-wing aircrafts?
  - Pressure drag + Friction drag
  - **Induced** drag + Profile drag (wing) + Parasite drag (body)
    - Induced drag(powers proportional to 1/v)

      drag on the wing that is due to lift
    - Profile drag, Parasite drag (powers proportional to v^3)

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
  - Set of agents in neighborhood N

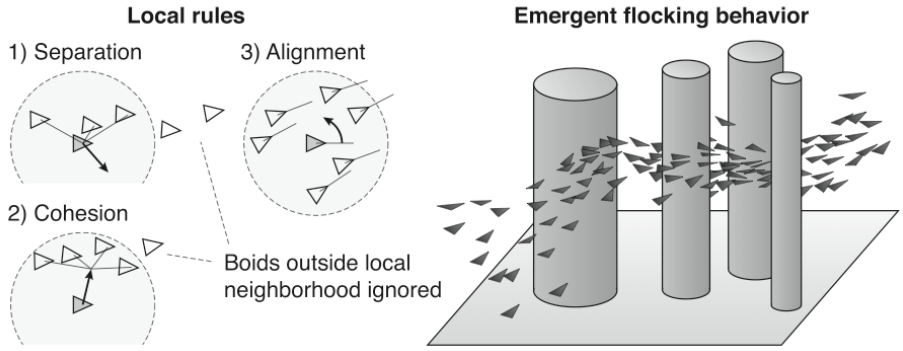


Figure 33: week7_swarm_reynolds

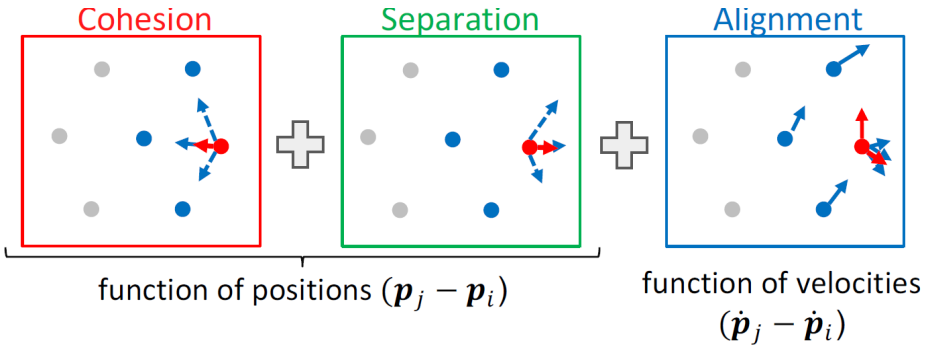


Figure 34: week7_swarm_reynolds_model

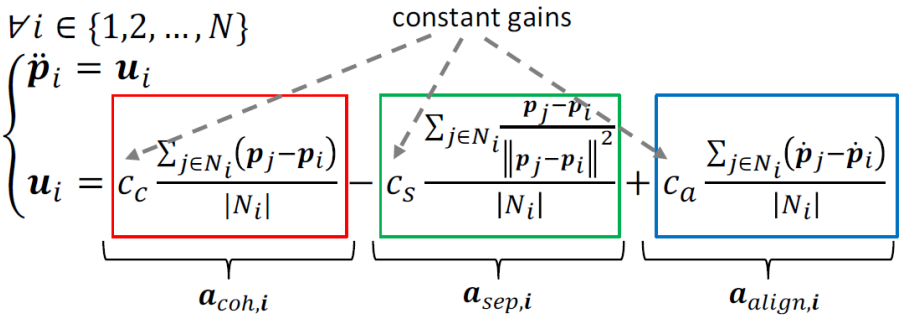


Figure 35: week9_swarm_reynolds_equ

- 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

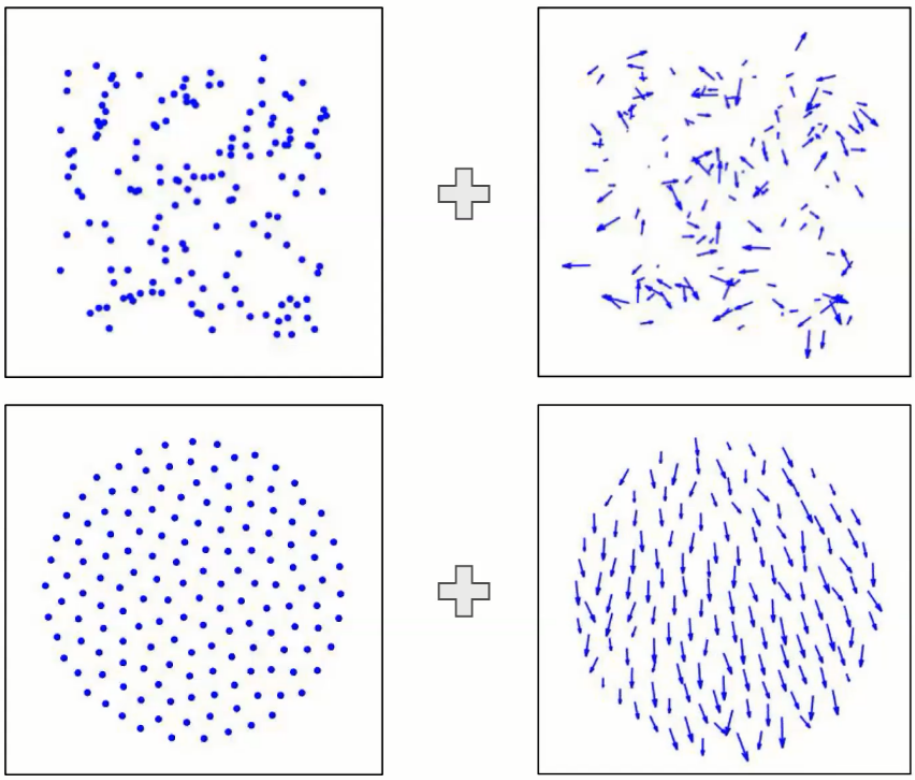


Figure 36: week7_swarm_reynolds_equilibrium

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

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

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

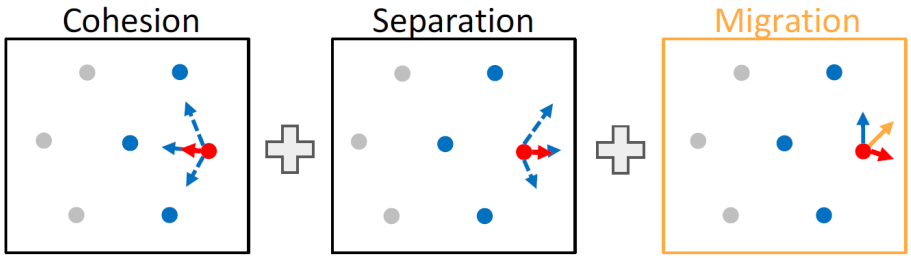


Figure 37: week7_swarm_reynolds_model_migration

- Equation

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

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

  - 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

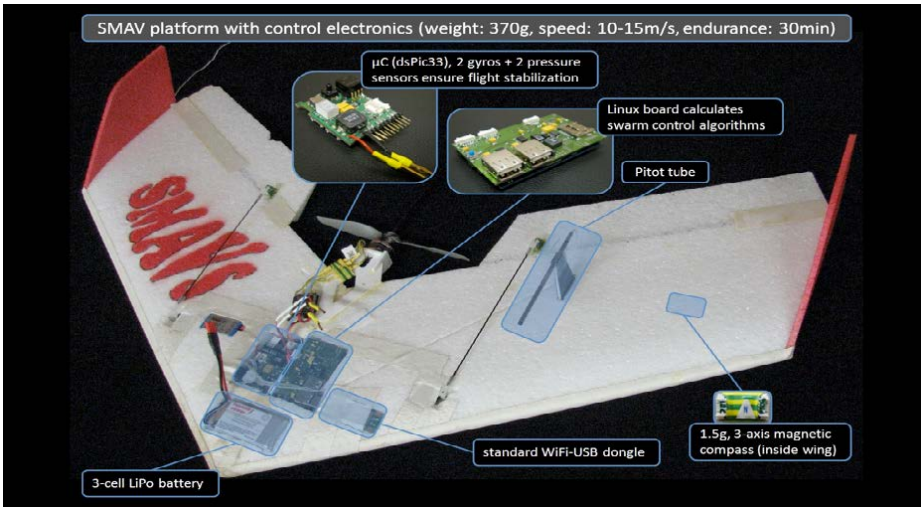


Figure 38: week7_swarm_project_example

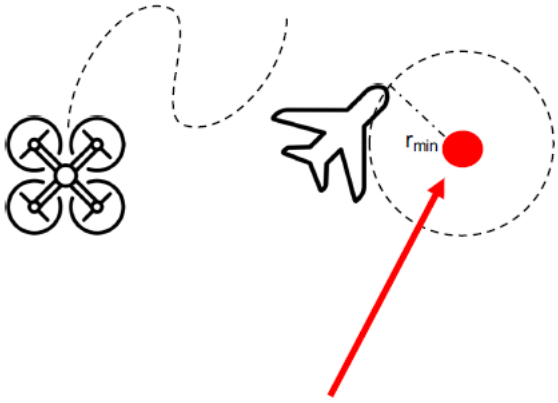


Figure 39: week7_swarm_virtual

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)

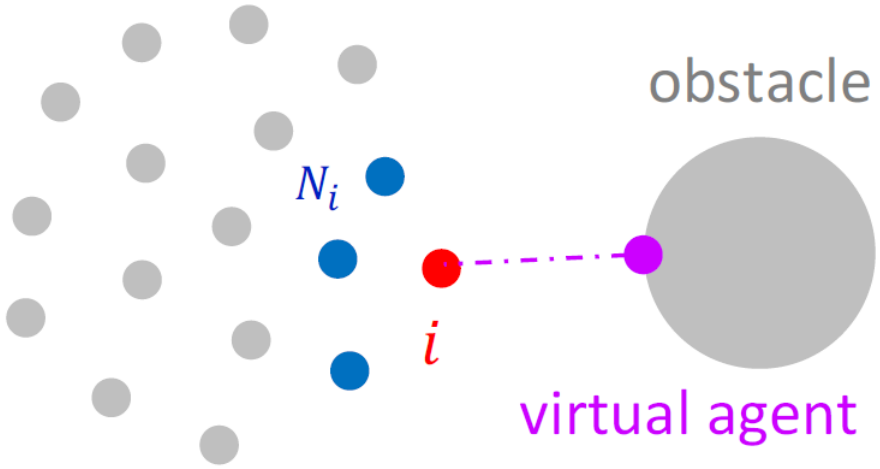


Figure 40: week7_swarm_reynolds_obs

- Visualization

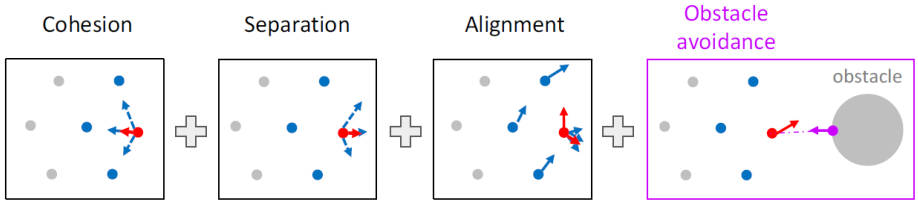


Figure 41: week7_swarm_reynolds_obs_viz

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

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

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

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

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

$$\begin{cases} \dot{\boldsymbol{p}}_i = \boldsymbol{u}_i \\ \boldsymbol{u}_i = \boldsymbol{v}_{sep,i} + \boldsymbol{v}_{spp,i} + \boldsymbol{v}_{\text{fric},i} \end{cases}$$

- 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

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

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

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

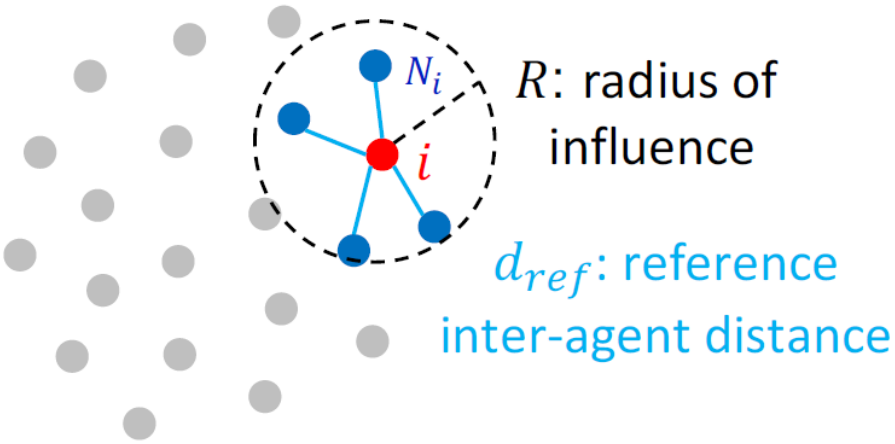


Figure 42: week7_swarm_olfati_model

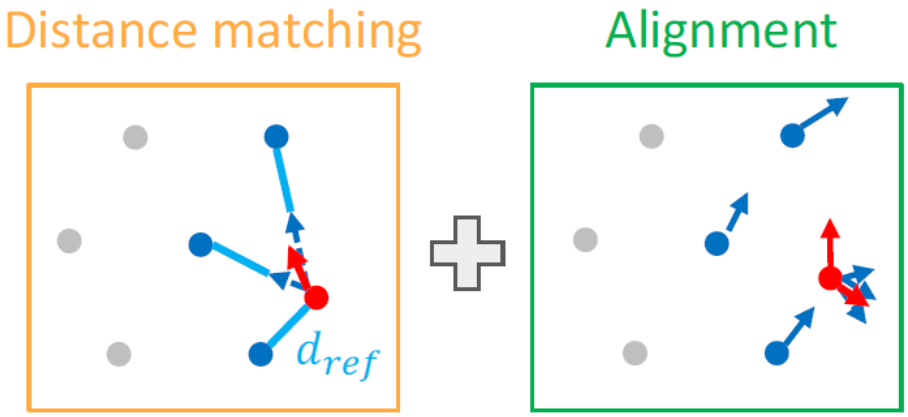


Figure 43: week7_swarm_olfati_model_viz

  - 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**

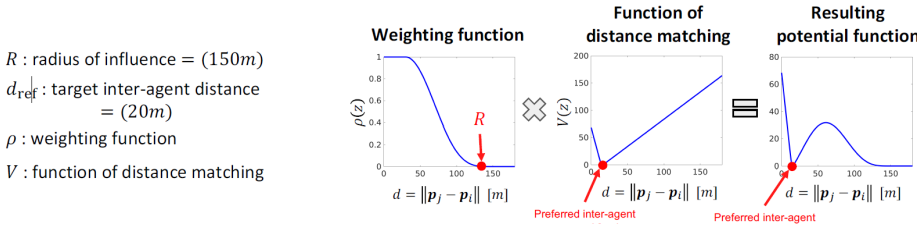


Figure 44: week7_swarm_olfati_model_dist_mat

  - 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

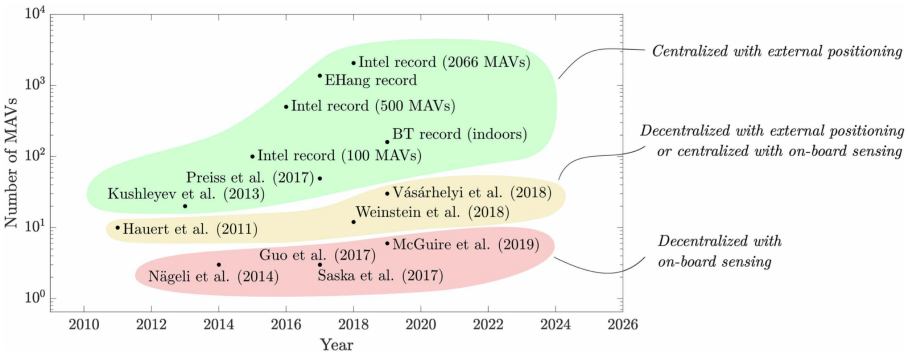


Figure 45: week7_swarm_survey

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.dmduav.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

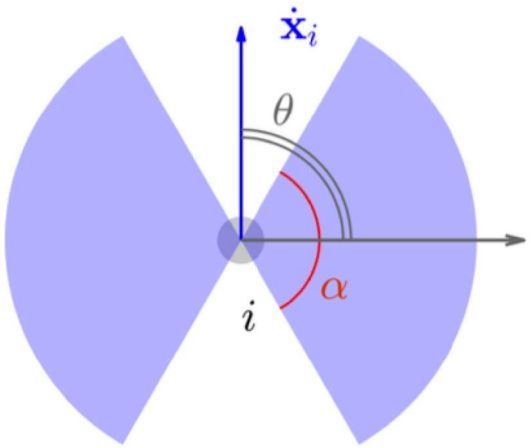


Figure 46: week7_swarm_limited_vision

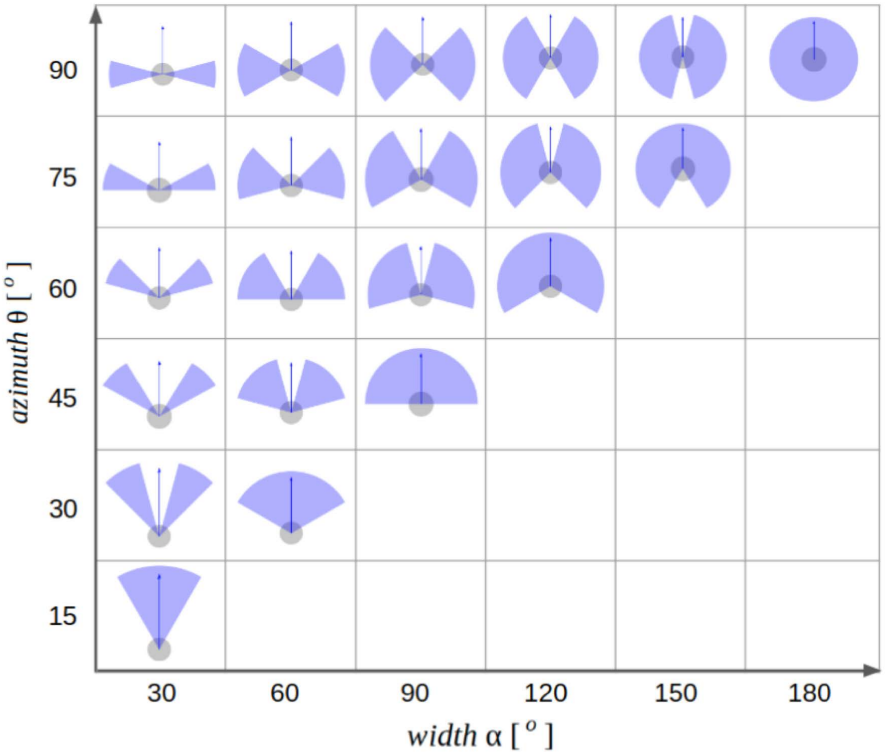


Figure 47: week7_swarm_limited_vision_tests

- **Safety**: ability to avoid collisions
- **Union**: ability to stay informed on neighbors
- **Connectivity**: ability to broadcast messages among drones

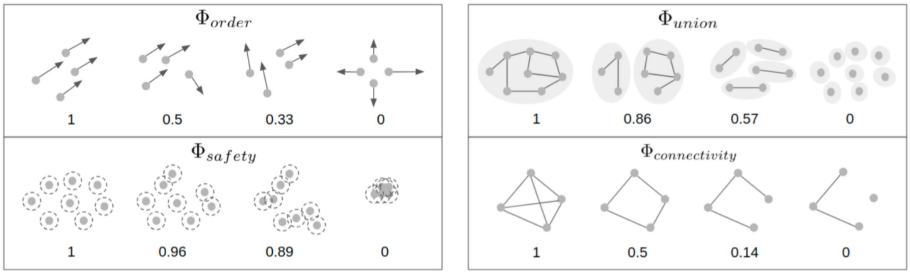


Figure 48: week7_swarm_limited_vision_metrics

- results

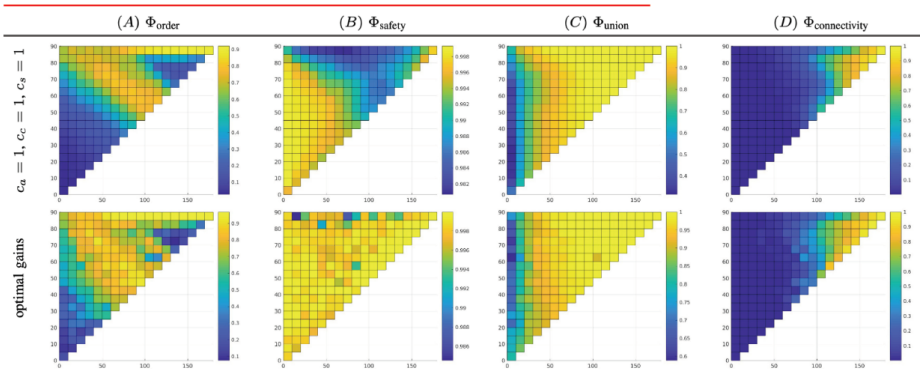


Figure 49: week7_swarm_limited_vision_results

- focus on order and safety (alignment and collision prevention capability)
- largest azimuth and FoV has best performance
- increase in either azimuth or FoV only will degrade the performance
- 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

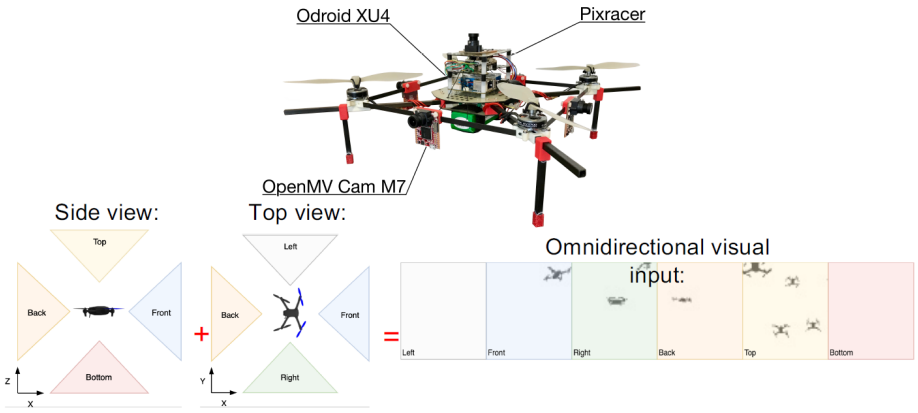


Figure 50: week7_swarm_vision_Schilling19RAL

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

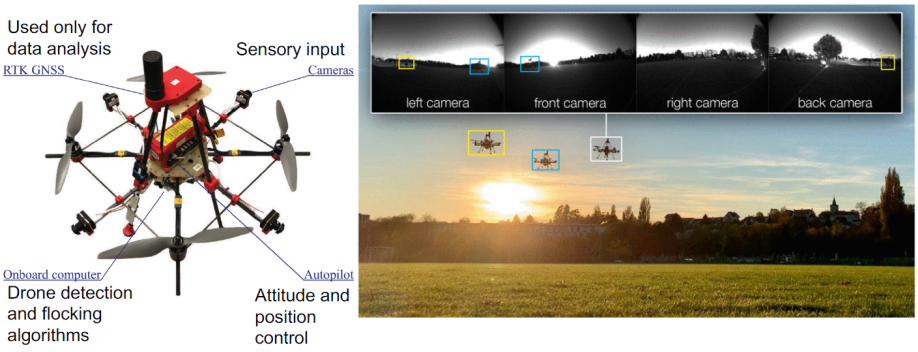
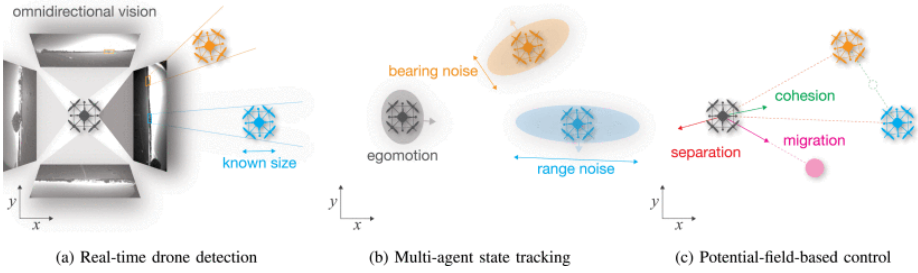


Figure 51: week7_swarm_vision_Schilling2021RAL



(a) Real-time drone detection (b) Multi-agent state tracking (c) Potential-field-based control

Figure 52: week7_swarm_vision_Schilling2021RAL_control

- 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

  virtual agent; integrate into equations with **alignment and separation term** (non cohesion)

- How is a migration point incorporated in flocking algorithms

  add a migration velocity term

- What does the Olfati-Saber algorithm ensure?

  No collision. The **acting force** will be zero when reach the preferred distance d_{ref}

- 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)

Introduction

- **lift** and **thrust** generation, and **maneuvers** mostly obtained by **using the wings**
- imitate the flapping-wing flight of birds, bats, and insects
- **scale down better** then rotary crafts and fixed wing UAVs

- **Challenges**:
  - Increased mechanical and control complexity
  - Complex modelling due to unsteady aerodynamics

Structure

categories were determined to be the tail (1) and wing design (2)

- requires the use of a **tail for stability and/or control** purposes
- g**enerate the lift and thrust forces** necessary for flight using flapping wings

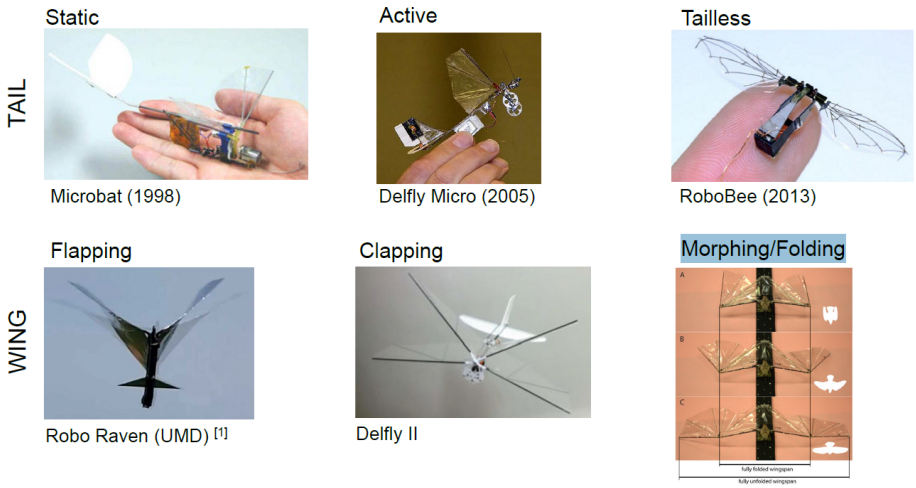


Figure 53: week8_flapping

Flight mechanics - Lift generation

- Flapping Wings

  two wings are flapped to **produce both lift and thrust**, thus overcoming gravity and drag to provide sustained flight

- Clapping

  Increase of lift during the “clapping

  cancels out the vertical oscillations

- Morphing/folding

  flap their wings downward, fold them in toward their body

  minimum wing area during the upward flap -> minimize undesired negative lift

Lift generation in hovering flight

Asymmetric hovering

Symmetric hovering

- produced during the entire wing stroke exploiting different unsteady mechanisms.
  - Leading edge vortex
  - Rotational forces
  - Clap-and-flight motion

Lift generation in forward flight

Downstroke/Upstroke

Flight mechanics - Maneuvering

flapping wing MAVs can use the tail and / or the wings for control.

- ail actuation
  - Aircraft tail
  - Aircraft tail with propeller for yaw and elevator for pitch
  - Inverted V-tail
- Wing actuation
  - Changing the angle of incidence
  - Tensioning the wing (拉紧机翼)

Energetics

? Checkpoints

- What are the main types of **tail** designs found in flapping wing MAVs?
  - Static
  - Active
  - Tailless
- What are the main types of **wings** designs found in flapping wing MAVs?
  - Flapping
  - Clapping
  - Morphing/Folding

- What are the **main mechanisms** for **lift generation** in symmetric hovering flight?
  - Leading edge vortex
  - Rotational forces
  - Clap-and-flight motion
- What are the main **steering strategies** in flapping flight MAVs?
  - Tail actuation
    - Aircraft tail
    - Aircraft tail with propeller for yaw and elevator for pitch
    - Inverted V-tail
  - Wing actuation
    - Changing the angle of incidence
    - Tensioning the wing (拉紧机翼)
    - Controlling the stroke (拍打角度)

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 piolted aircraft systems / autonomous aircraft systems

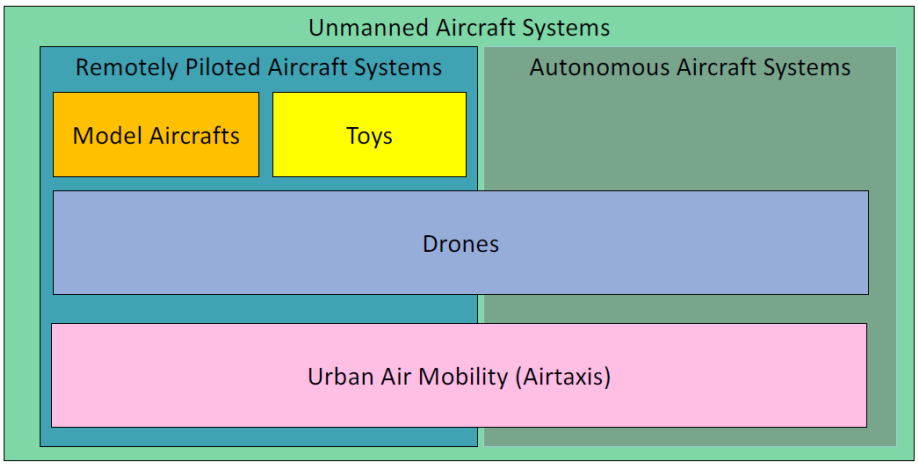


Figure 54: week9_regulation_uas

- 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 <=100m around crowds
- **ANSP (Skyguide) or Airport responsibility**
  - **> 5km** Distance to civil & military airports/aerodromes
  - **< 150m** AGL (Above Ground Level) within a CTR
- Act in EU
  - Open/Specific/Certified
  - Difference
    - restrictions: MTOM **25kg**
    - maximum flying altitude: **120m**

Specific Category

Application for an operating permit on the basis of the **SORA (Specific Operations Risk Assessment)**

Operational Volume = Flight Geography + Contingency Volume

- **?** Robustness Levels: Integrity + Assurance

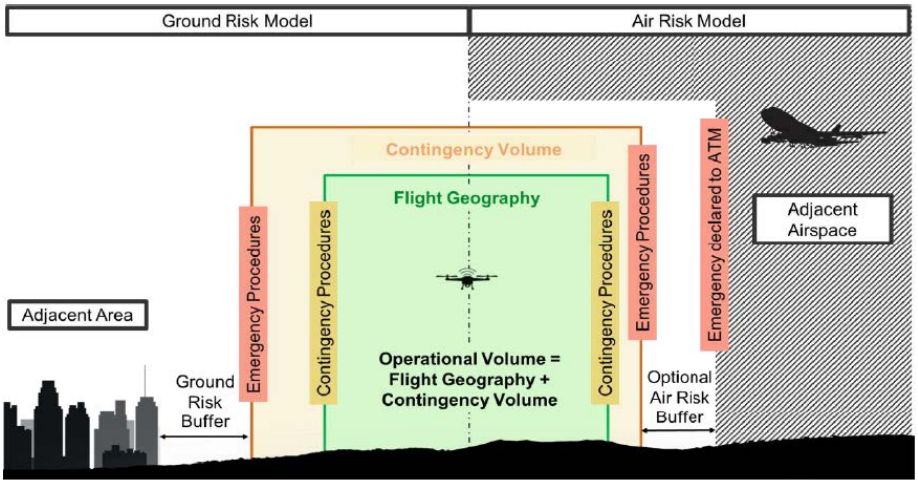


Figure 55: week9_regulation_SORA

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

- U-space is capable of ensuring the **smooth operation of all categories of drones, all types of missions and all drone users** in all operating environment

⍰ Checkpoints

- Federal Office of Civil Aviation FOCA
- What defines the three drone categories?
  1. **Open**-within the legal framework (No Authorization required)

     low risk; maximum flying altitude: 120m
  2. **Specific**-Not sufficiently safe (Authorization required)

     enhanced risk
  3. **Certified**-Approved to accepted standards

     risk comparable to manned aviation

- What is the SORA declaration?
  - stands for Specific Operations Risk Assessment
  - used for Specific Drone Categories
  - assigning to a UAS-operation two classes of risk, i.e., ground risk model and air risk model

  the multi-stage process of risk assessment aiming at risk analysis of certain unmanned aircraft operations, as well as defining necessary mitigations and robustness levels
- Is FOCA authorization sufficient for operating drones in the “Specific” category?

  No. FOCA authorization is required for the ”Specific” category, but it is not sufficient because **other federal, cantonal, and communal authorities may require additional authorizations**
- What is U-space?

  U-Space **provides a framework** to facilitate the implementation of **all** types of **operation** in all **classes** of airspace and all **types** of environment, while ensuring an orderly **coexistence with manned aviation and air traffic control.**

  from U-Space: The airspace of the future

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
Pros	Stiff, lightweight	Easy to manufacture by 3D printing or injection molding	Lightweigh
Cons	Expensive, complex to manufacture	Heavier, less stiff	complex to
Comment	-	useful for prototyping	-

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

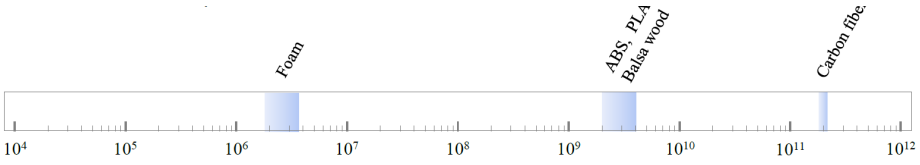


Figure 56: week9_UAS_Hardware_Young_modulus

- 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

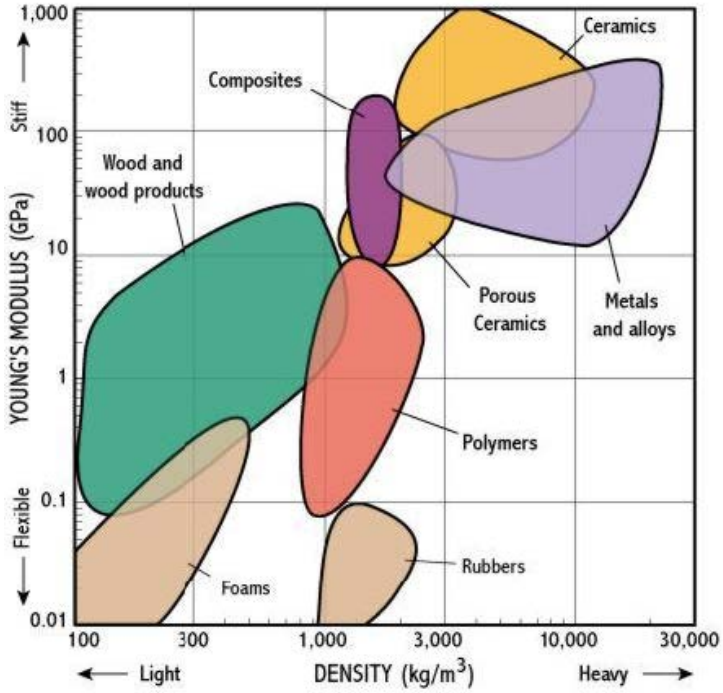


Figure 57: week9_UAS_Hardware_specific_modulus

Category

- Nickel-Cadmium (NiCd) | 镍镉
  - Mature and cheap
  - Low energy and power density -> short flight time
- Nickel-Metal Hydrate (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

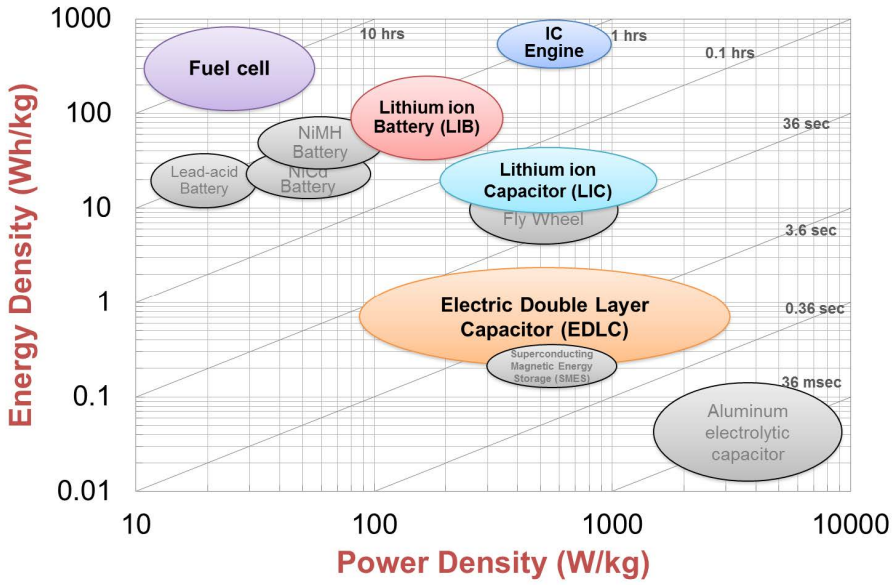


Figure 58: week9_UAS_Hardware_energy_density

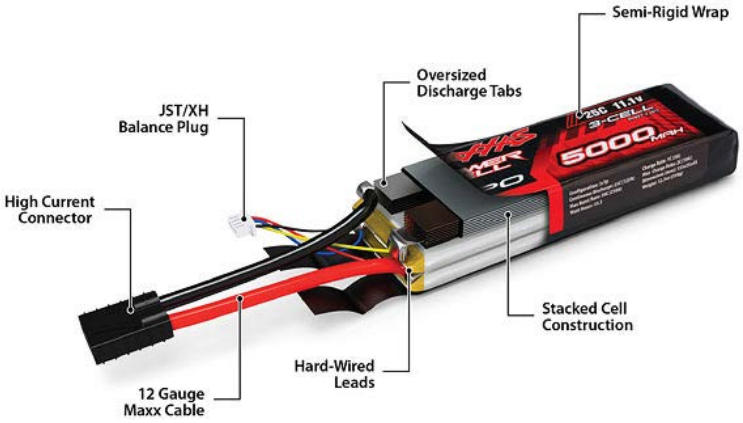


Figure 59: week9_UAS_Hardware_LiPo_battery

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

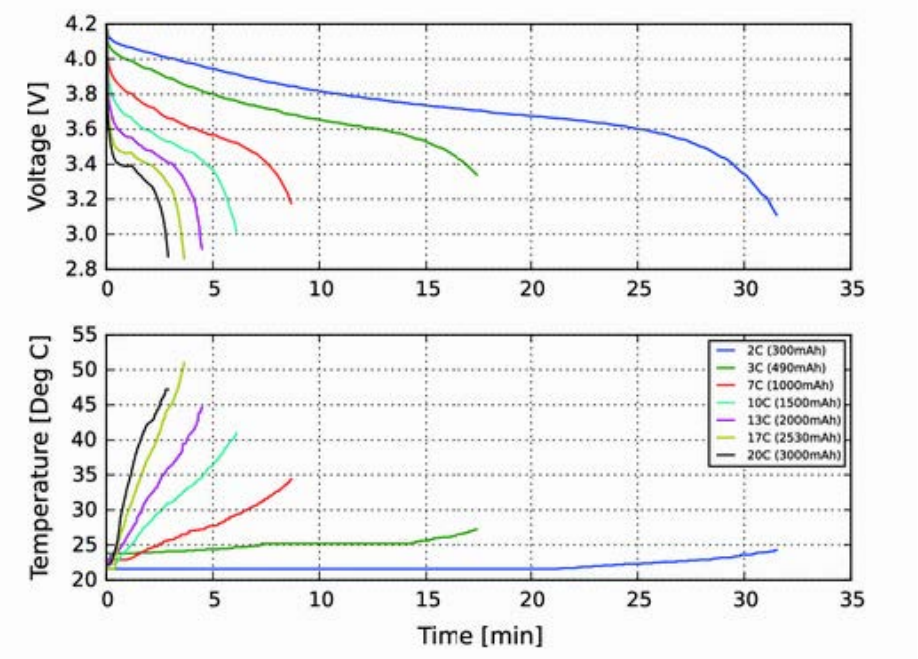


Figure 60: week9_UAS_Hardware_discharge_curve

- 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

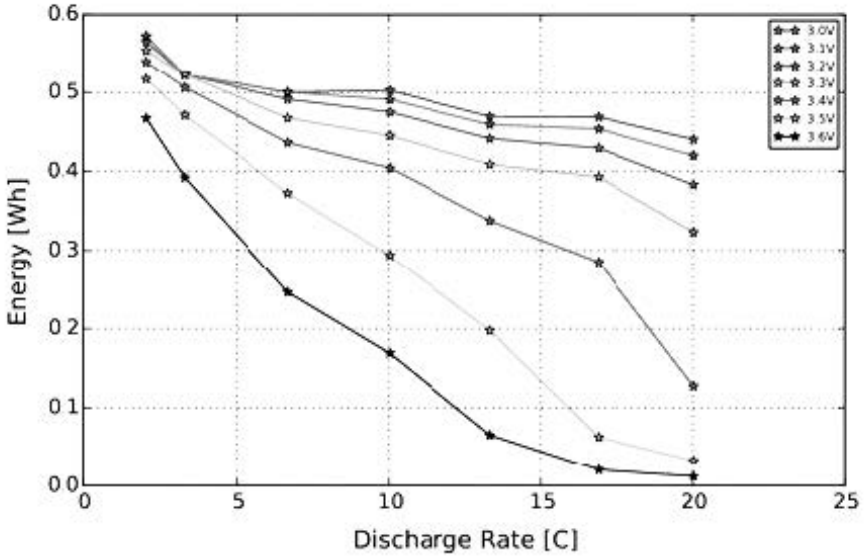


Figure 61: week9_UAS_Hardware_energy_curve

- 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	Comb
Pros	clean and quite; Reliable and easy to maintain; Fast to change operational state (accelerate and decelerate)	High
Cons	Limited weight to power ratio due to battery	Vibra

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

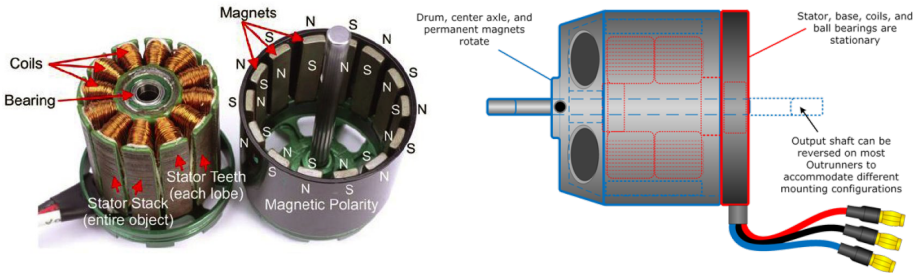


Figure 62: week9_UAS_Hardware_brushless_motor

- 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 and **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

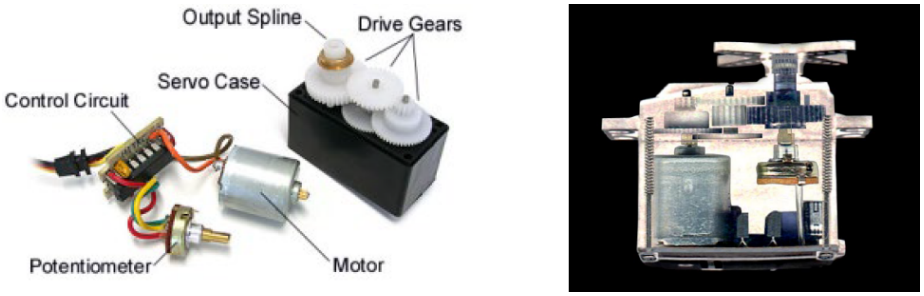
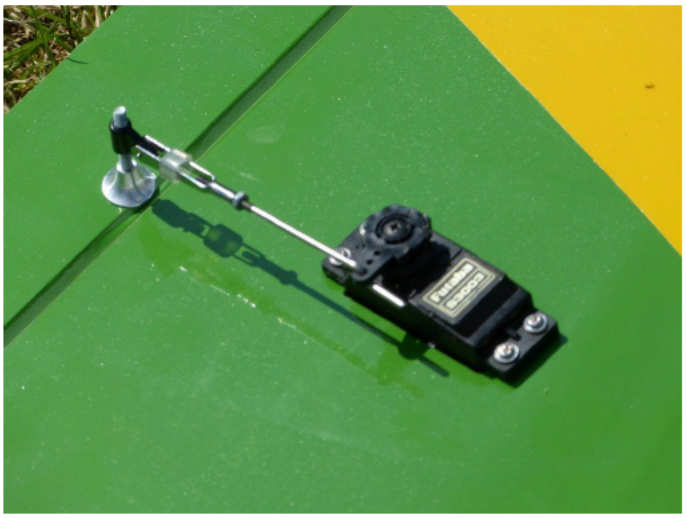
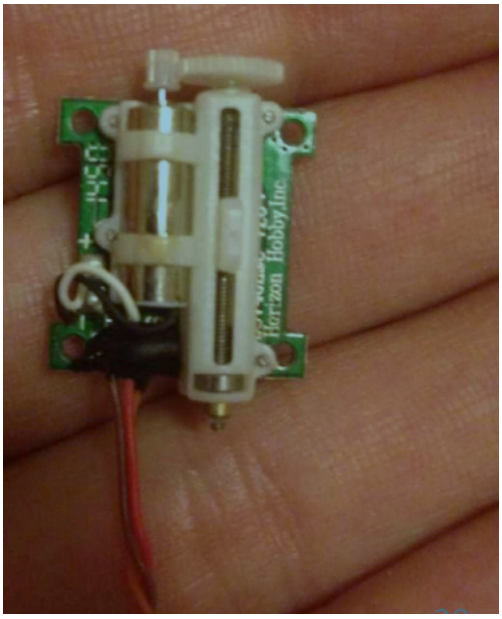


Figure 63: week9_UAS_Hardware_servomotor

- 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**
  - measure how far will fly up
  - 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 -> tip speed

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

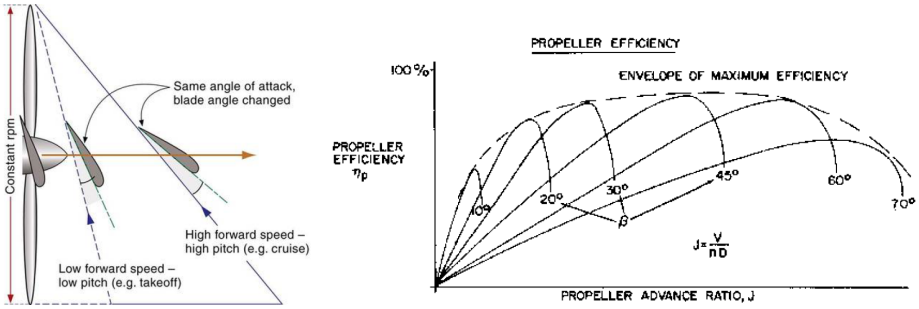


Figure 64: week9_UAS_Hardware_pitch_efficiency

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

Figure 65: week9_UAS_Hardware_variable_propeller

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

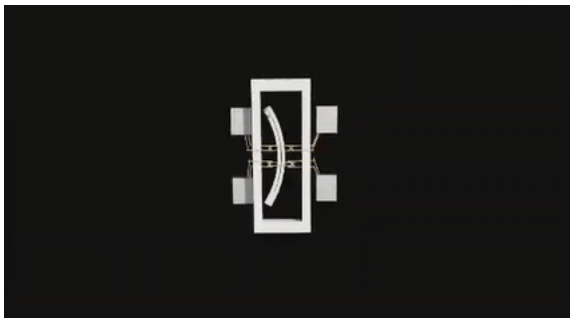


Figure 66: week9_UAS_Hardware_mems_gyros

  - 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)

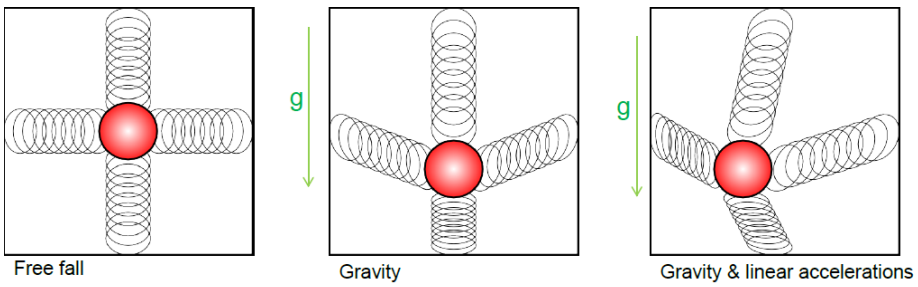


Figure 67: week9_UAS_Hardware_accelerometers

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

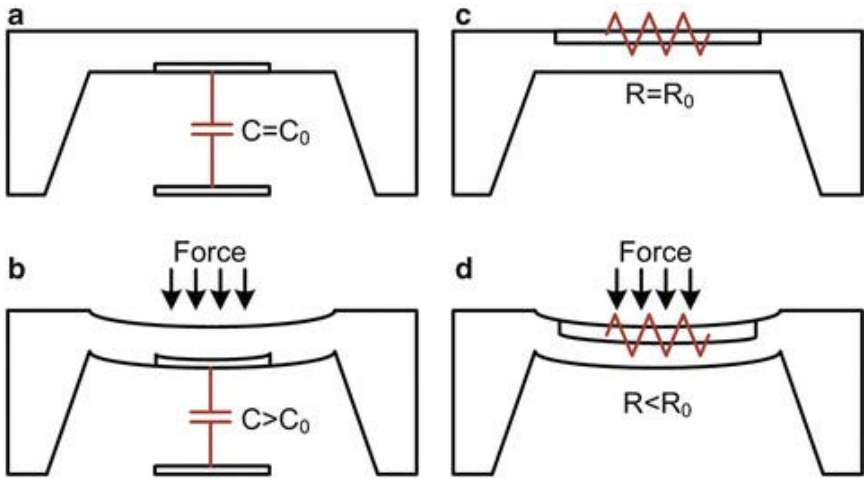


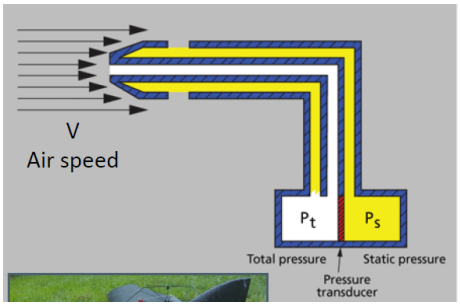
Figure 68: week9_UAS_Hardware_pressure_sensor

- 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

Figure 69: week9_UAS_Hardware_airspeed_sensor

- 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.)

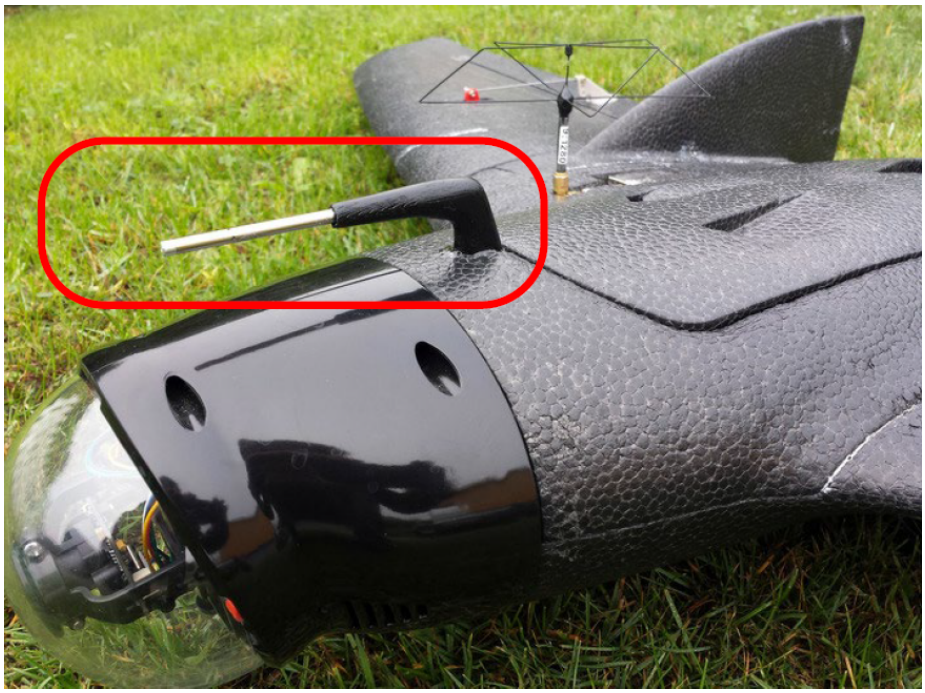


Figure 70: week9_UAS_Hardware_airspeed_sensor_example

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

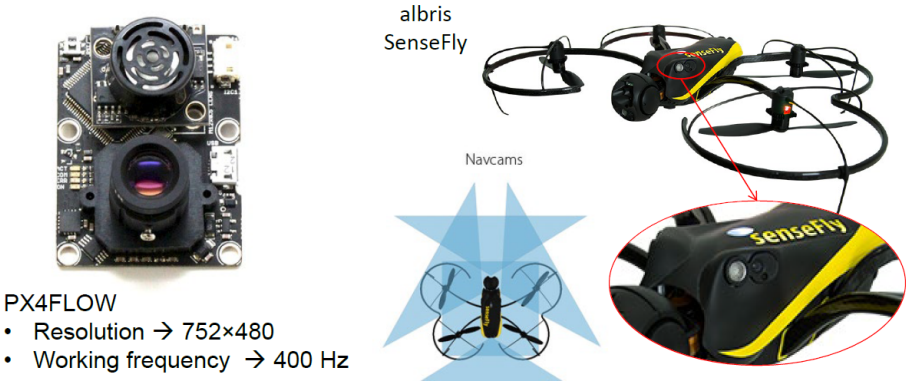


Figure 71: week9_UAS_Hardware_optical_camera

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

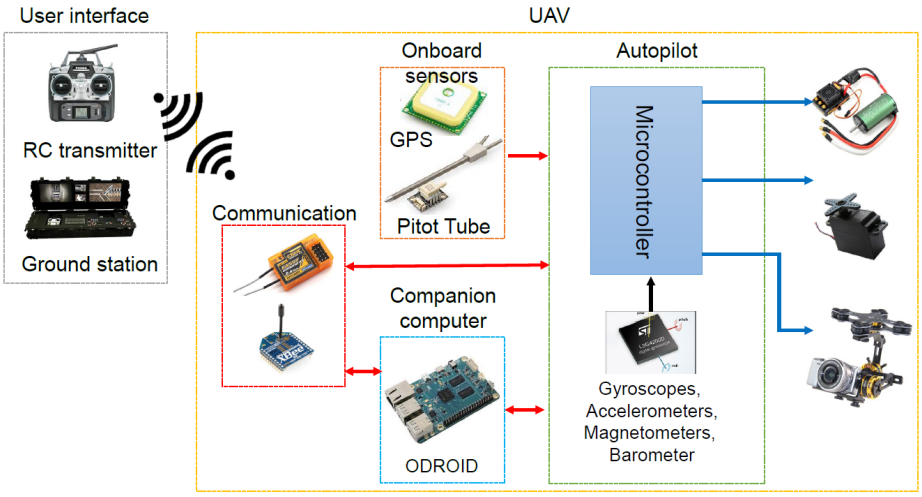


Figure 72: week9_UAS_Hardware_Autopilot_connection

Insect-inspired vision (week10)

- Inserts 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)
- translation + rotation component of the optical flow

For pure translational motion

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

- motion parallax, OF is
  1. directly proportional to the forward speed T
  2. Inversely proportional to Distance D
- Experiments1

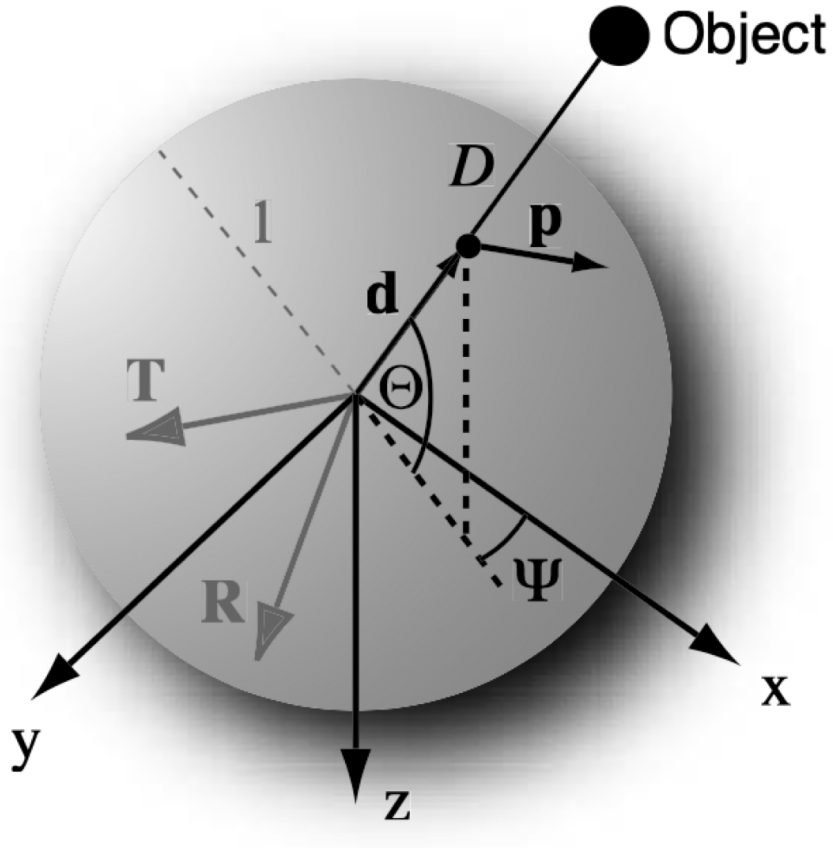


Figure 73: week10_optical_flow

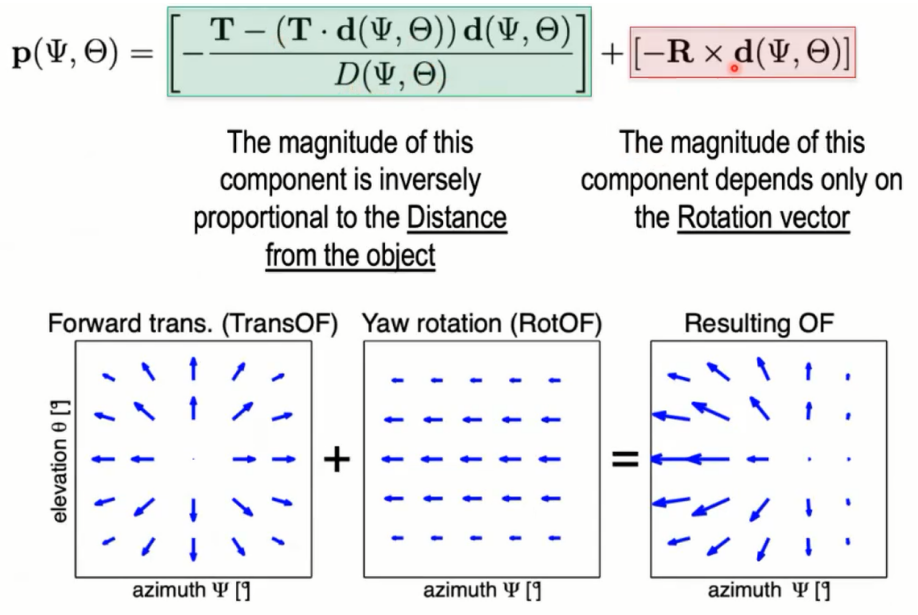


Figure 74: week10_optical_flow_component

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 plata (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

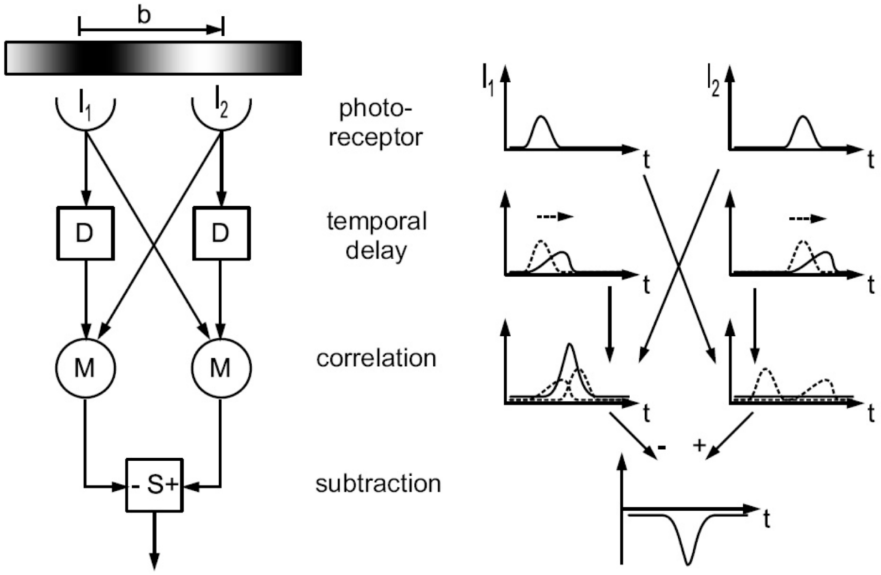


Figure 75: week10_emd

  - 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

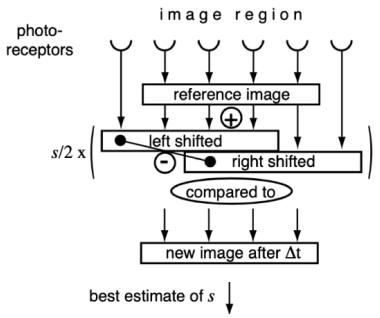
Gradient Descent Methods

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

- assumption: brightness I d**oes 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

$$\hat{I}(n, t + \triangle t) = I(n, t) + s\frac{I(n-k, t) - I(n+k, t)}{2k}$$

2. Error between estimated and new image

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

3. Compute image shift s that minimizes error E

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

Figure 76: week10_optical_flow_i2a

- used in Crazyflie drone for optical flow calculation
  1. moves k pixels in opposite ditections 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

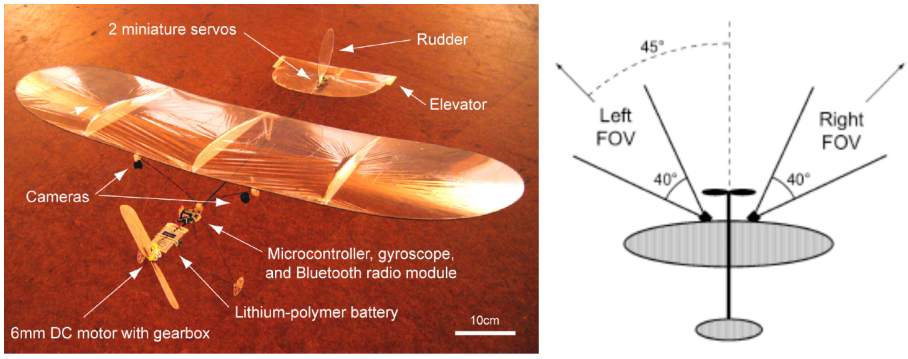


Figure 77: week10_optical_flow_i2a_drone

  - OFDiv = OFRight - OFLeft initiate rotation if OFDiv > threshold
  - OFDiff = abs[OFRight] - abs[OFLeft] 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
  - 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

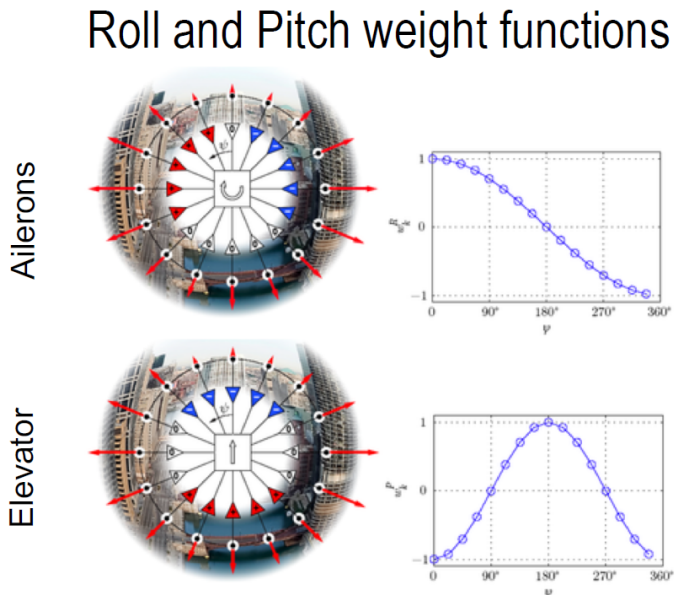


Figure 78: week10_optical_flow_i2a_control

  - **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 and temporal frequency 都是先增后减。

  spatial frequency 越大，角速度的图越向后偏移

- Functioning of Image Interpolation Algorithm

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

  use derivation to minimize the overlap error

- Methods for discounting rotational optic flow

  用 IMU 估计角速度，然后算 rotational optic flow 再减去图片得到的即可

  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
- Maneuverability

  able to change the velocity vector by changing the wing status

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

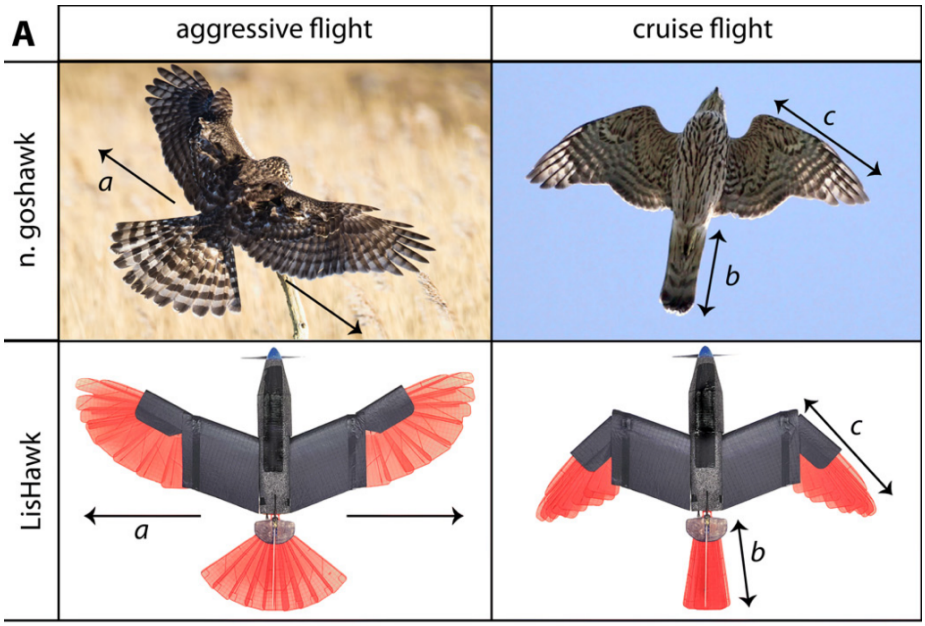
  able to change angular rate, e.g. pitch and roll angles

  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



	northern goshawk	LisHawk
wing span (*a*) (m)	0.89 - 1.22	1.05
tail length (*b*) (m)	0.20 - 0.27	0.24
wing chord (*c*) (m)	0.28 - 0.35	0.32

Figure 79: week10_morphing_wing

  - Actuator torque and speed

    Fly require high speed and ground locomotion requires high torque. Few motor can satisfy both.

  - Mass of center

    Fly: a bit front , Ground walking: in the center

  - Inner ratio

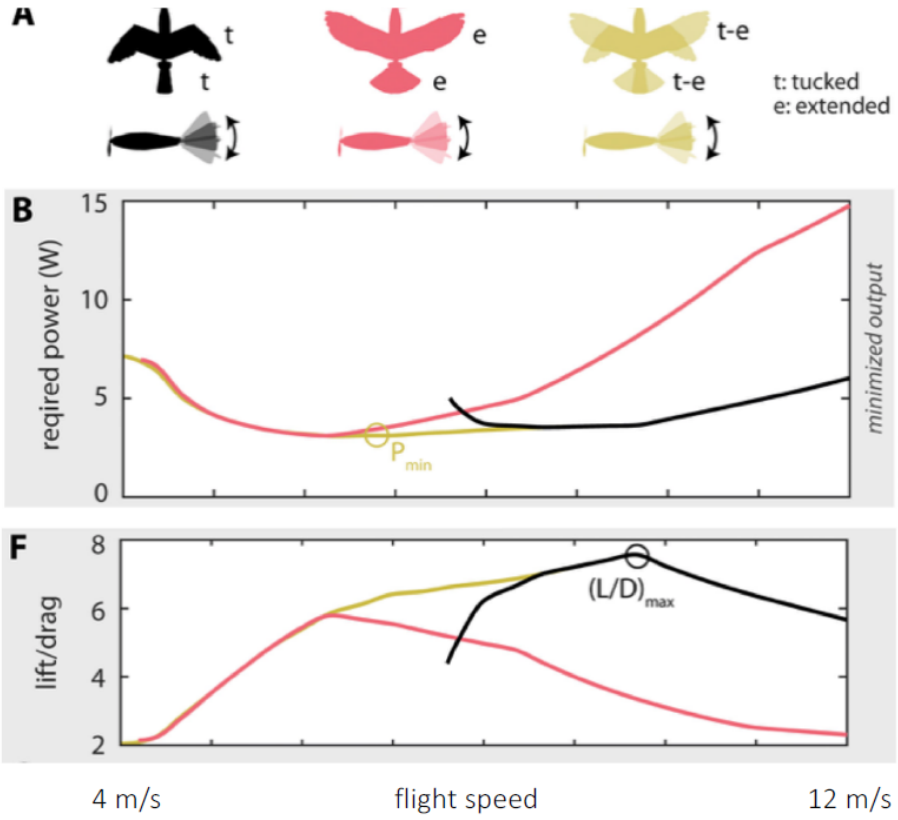
    约等于 wing span，飞的时候展开，地上收起来

- 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 need base speed to activate

- **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 Cotrol, 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 acion? 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

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

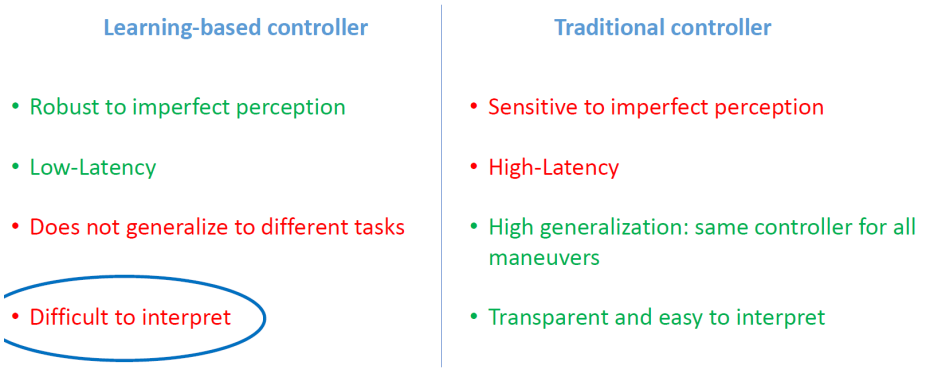


Figure 80: week11_tradition_learning_navigationpng

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)

1. 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

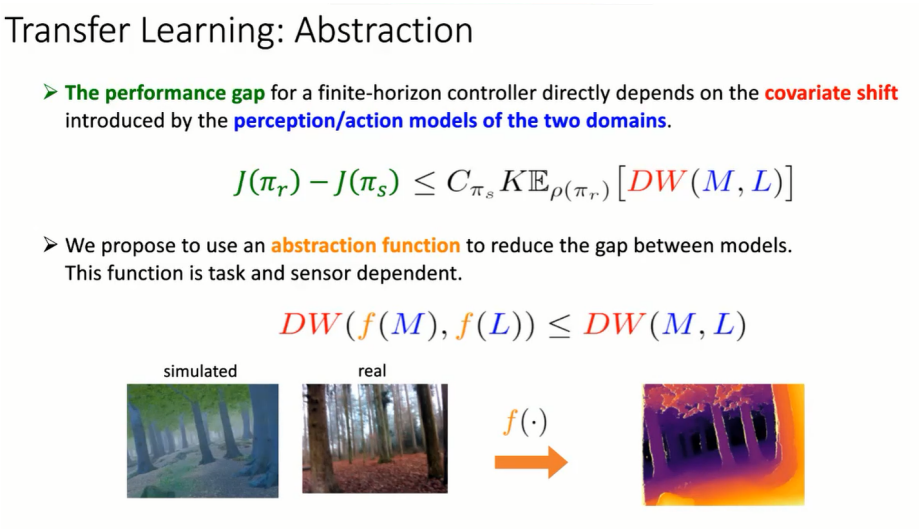


Figure 81: week11_transfer_learning

  - How to find the abstraction function

- 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 solves 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**

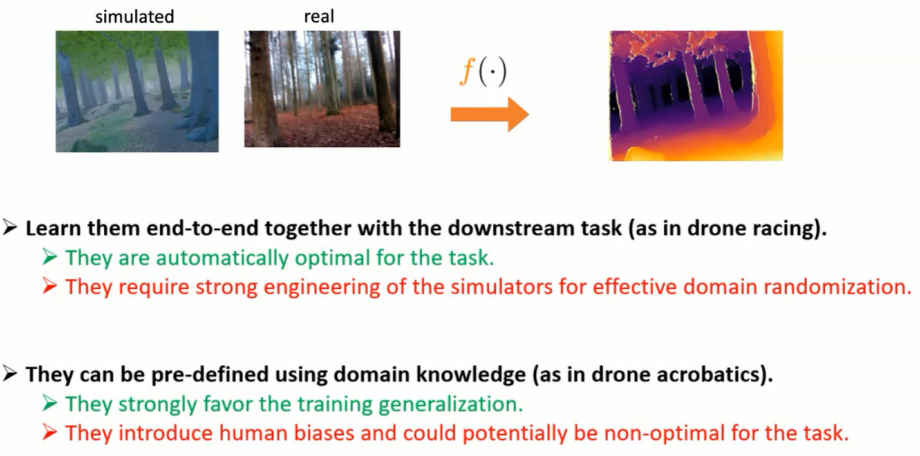


Figure 82: week11_abstraction_function

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.