Quadcopters – Basics, Simulator and Contest

Danping Zou

Assistant Professor

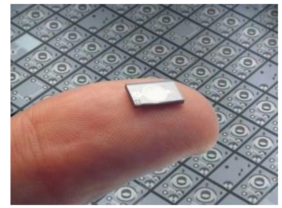
dpzou@sjtu.edu.cn

School of Electronic Information and Electrical Engineering

Shanghai Jiao Tong University

Era of small drones

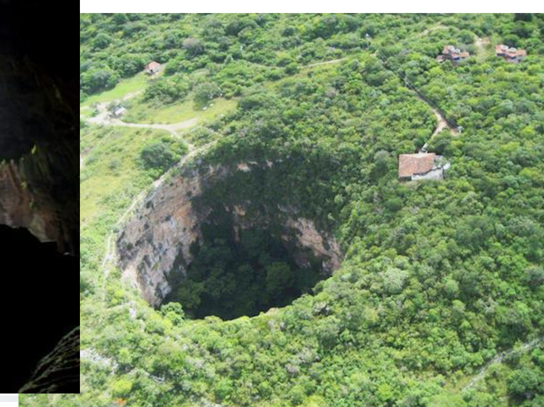
Smart phone



Sensors
- gyroscope
- accelerometer
- video camera
- barometer
- Compass

Flying cameras

Potential applications



Outline

Platform

- Physics
- Control
- Choose a platform for development

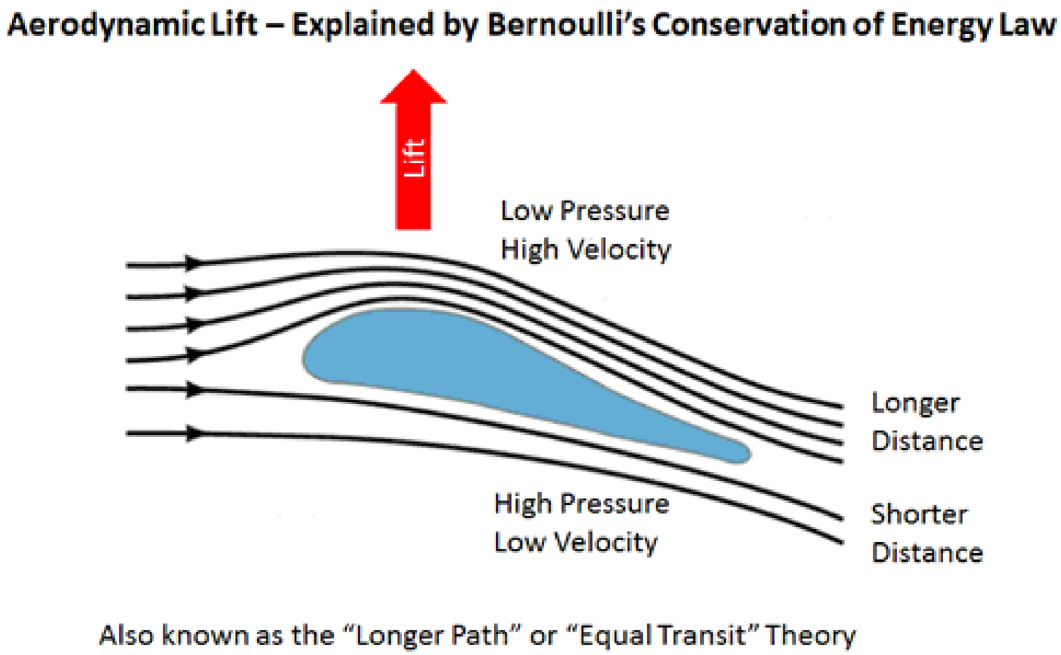
Gazebo+ROS quadcoptor simulator

UAV contest

- Target tracking
- Path following
- Crazy landing
- Voice control
- Kick the ‘ball’

Platform - Physics

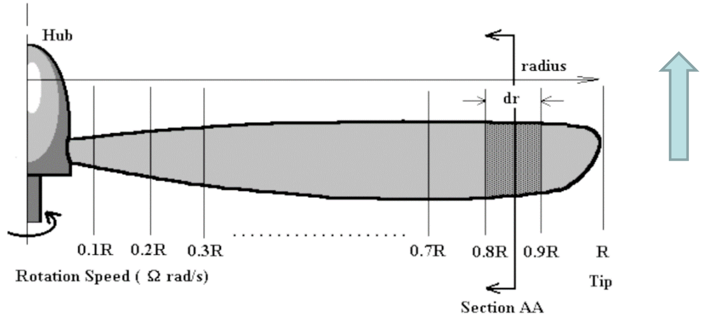
- Where does lift come from?

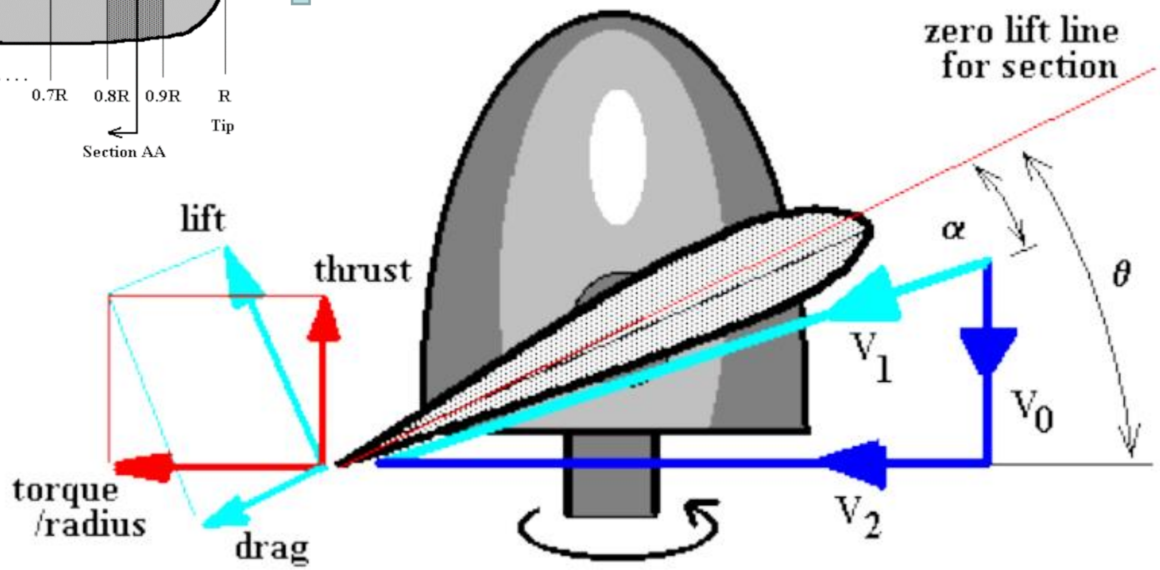


Picture from https://www.mpoweruk.com/figs/flight_theory.htm

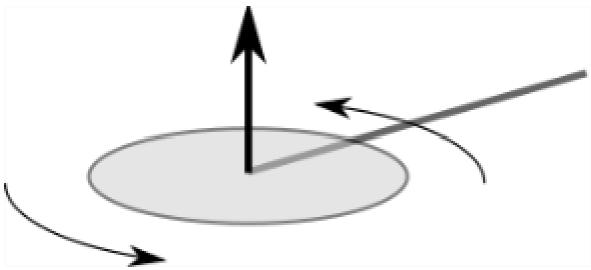
When the velocity of a fluid increases, its pressure decreases by an equivalent amount to maintain the overall energy. This is known as **Bernoulli's Principle**

Rotor dynamics (Blade Element Theory)





- Thrust and torque by a single rotor



$$F_i = C\rho A r^2 \omega_i^2$$

C :Trust coefficient

ρ :Density of air

A :Rotor disk area

r :Rotor disk radius

ω_i :Angular velocity

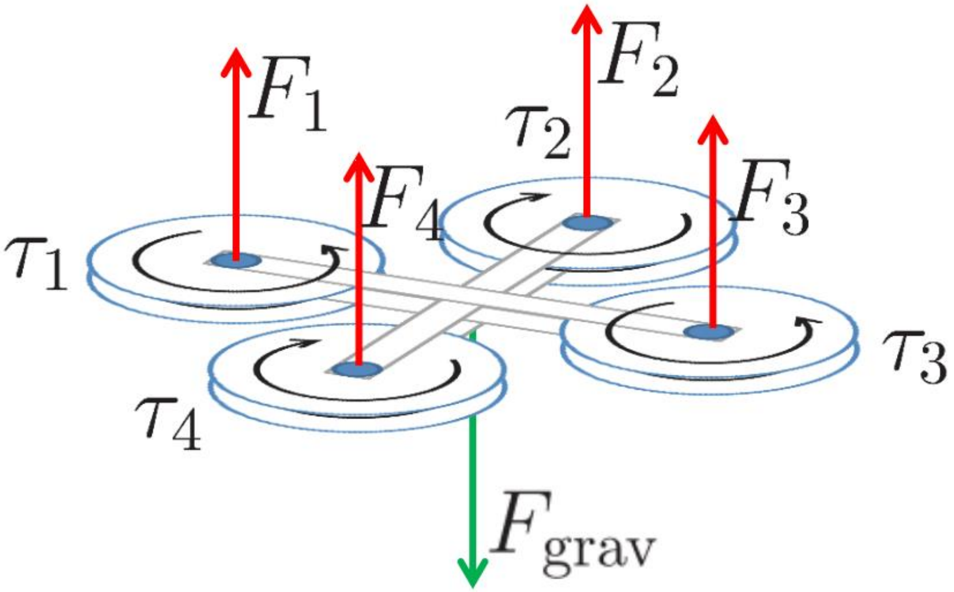
Thrust : $F_i = C_T \omega_i^2$

Torque : $\tau_i = C_Q \omega_i^2$

Aerodynamics

Quadcopter

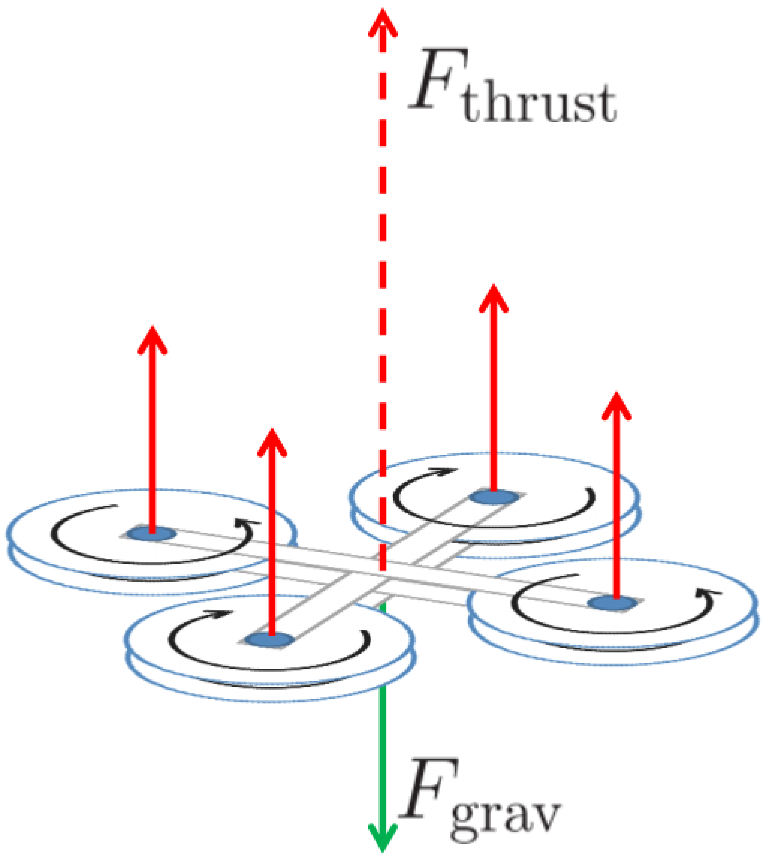
- Thrust
- Torque
- Gravity



*From Jurgen Strum's lecture slides

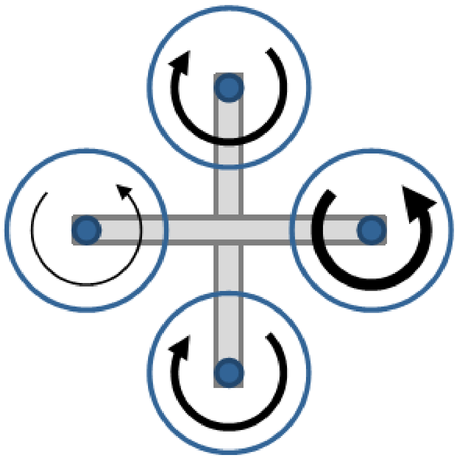
Aerodynamics

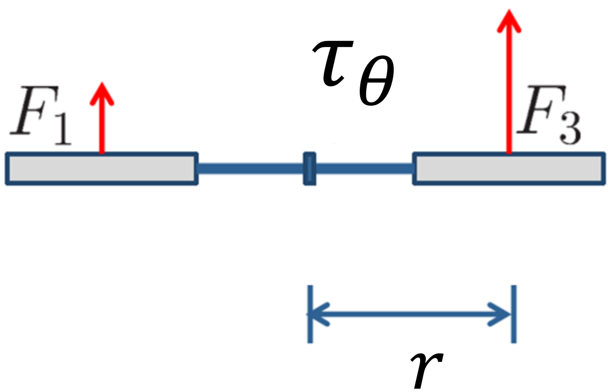
- Thrust $F_{\text{thrust}} = F_1 + F_2 + F_3 + F_4$



Pitch (and roll)

- Unbalanced thrusts make the quadcopter tilt





Side view of quadrocopter

Induced torque:

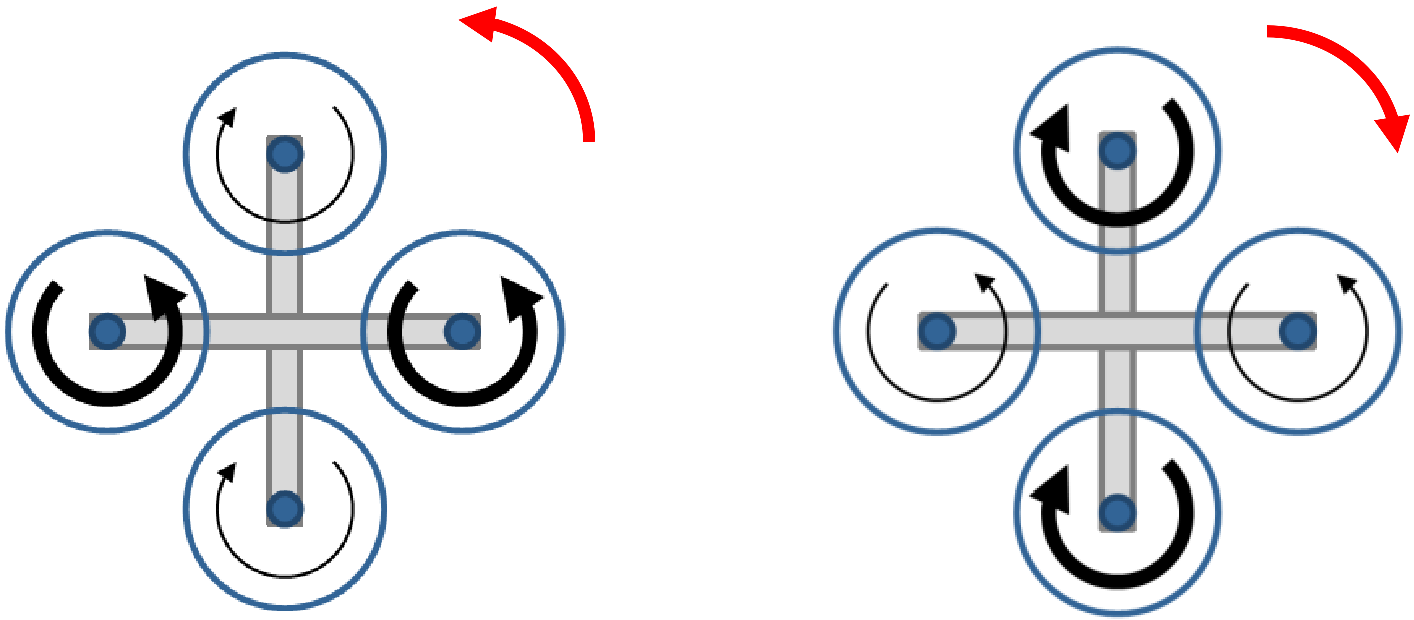
Pitch: $\tau_x = (F_1 - F_3)r$

Roll: $\tau_y = (F_2 - F_4)r$

Aerodynamics

Yaw

- Unbalanced torques make the quadcopter spinning



Induced torque: $\tau_z = \tau_1 - \tau_2 + \tau_3 - \tau_4$

Aerodynamics

- Putting all together

Thrust : $F_i = C_T \omega_i^2$

Torque : $\tau_i = C_Q \omega_i^2$

$$\begin{bmatrix} F \\ \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} = \begin{bmatrix} C_T & C_T & C_T & C_T \\ 0 & C_T \cdot d & 0 & -C_T \cdot d \\ C_T \cdot d & 0 & -C_T \cdot d & 0 \\ -C_Q & C_Q & -C_Q & C_Q \end{bmatrix} \begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \omega_4 \end{bmatrix}$$

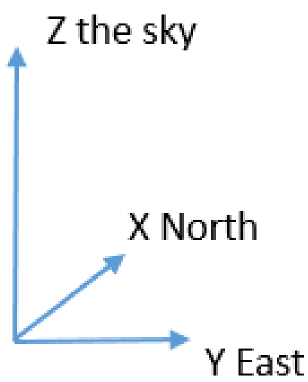
Rigid body dynamics

- We have the force and torque. How do we relate them to the motion of a quadcopter?
  - Newton-Euler equation
- Newton equation
  - **Total force = variation of linear momentum**

$$F = \frac{d(mv)}{dt} = m\dot{v}$$

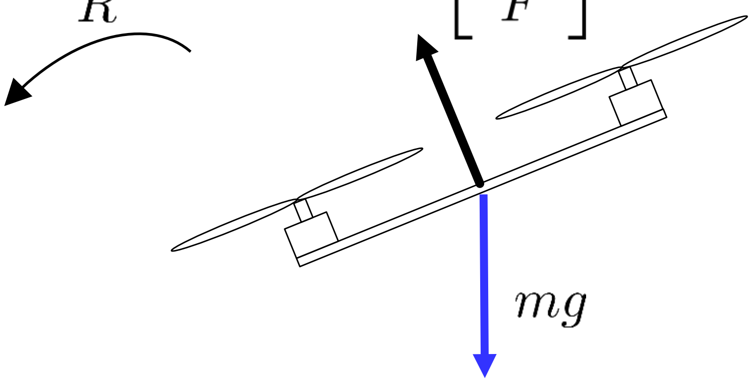
Rigid body dynamics

- Newton equation



World coordinate system

$$T_b = \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix}$$



$$F = RT_b - mg$$

$$m\dot{v} = RT_b - mg$$

Rigid dynamics

Euler equation

- **Total torque = variance of angular momentum**

$$\tau = \frac{d(I(R)\omega)}{dt}$$

$I(R)$ is a inertial tensor expressed in the world coordinate system. It can be transformed from inertial tensor expressed in the body frame by

$$I(R) = RI_bR^T$$

Euler equation

$$\tau = \frac{d(I(R)\omega)}{dt} = \frac{RI_b R^T \omega}{dt}$$

$$= \dot{R}I_b R^T \omega + RI_b \dot{R}^T \omega + I(R)\dot{\omega}$$

$$= \omega \times I(R)\omega + I(R)\omega \times \omega + I(R)\dot{\omega}$$

$$\tau = \omega \times I(R)\omega + I(R)\dot{\omega}$$

World frame

$$\omega_b = R^T \omega$$

$$\tau_b = R^T \tau$$

$$\tau_b = \omega_b \times I_b \omega_b + I_b \dot{\omega}_b$$

Body frame

Drag force

$$m\dot{v} = mg\vec{\mathbf{a}}_3 + RT_b + \boxed{Dv}$$

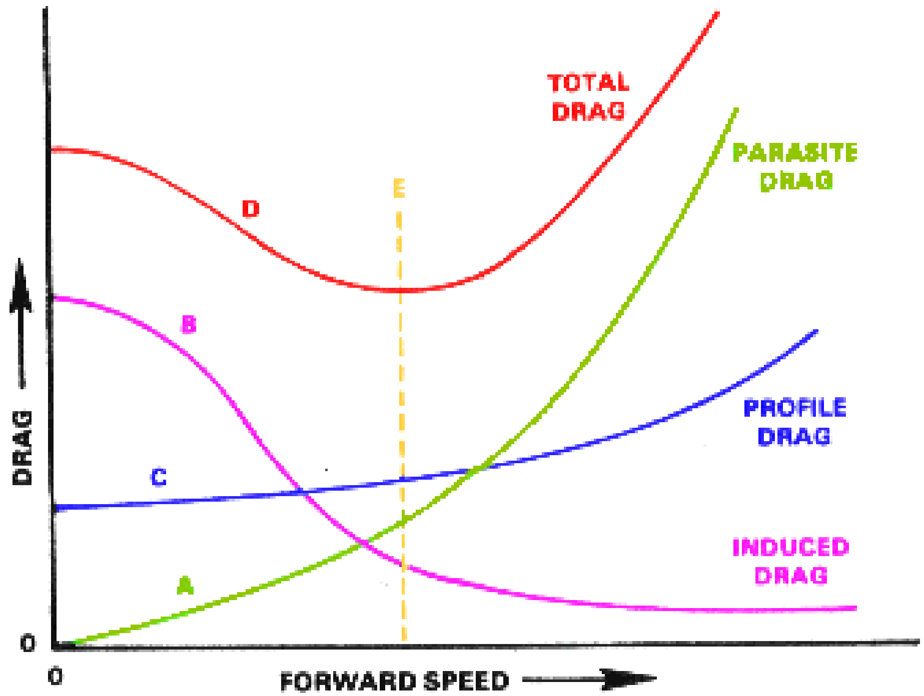


FIGURE 2-23. DRAG/AIRSPEED RELATIONSHIP.

A Profile drag is the drag incurred from frictional resistance of the blades passing through the air.

B Induced drag is the drag incurred as a result of production of lift

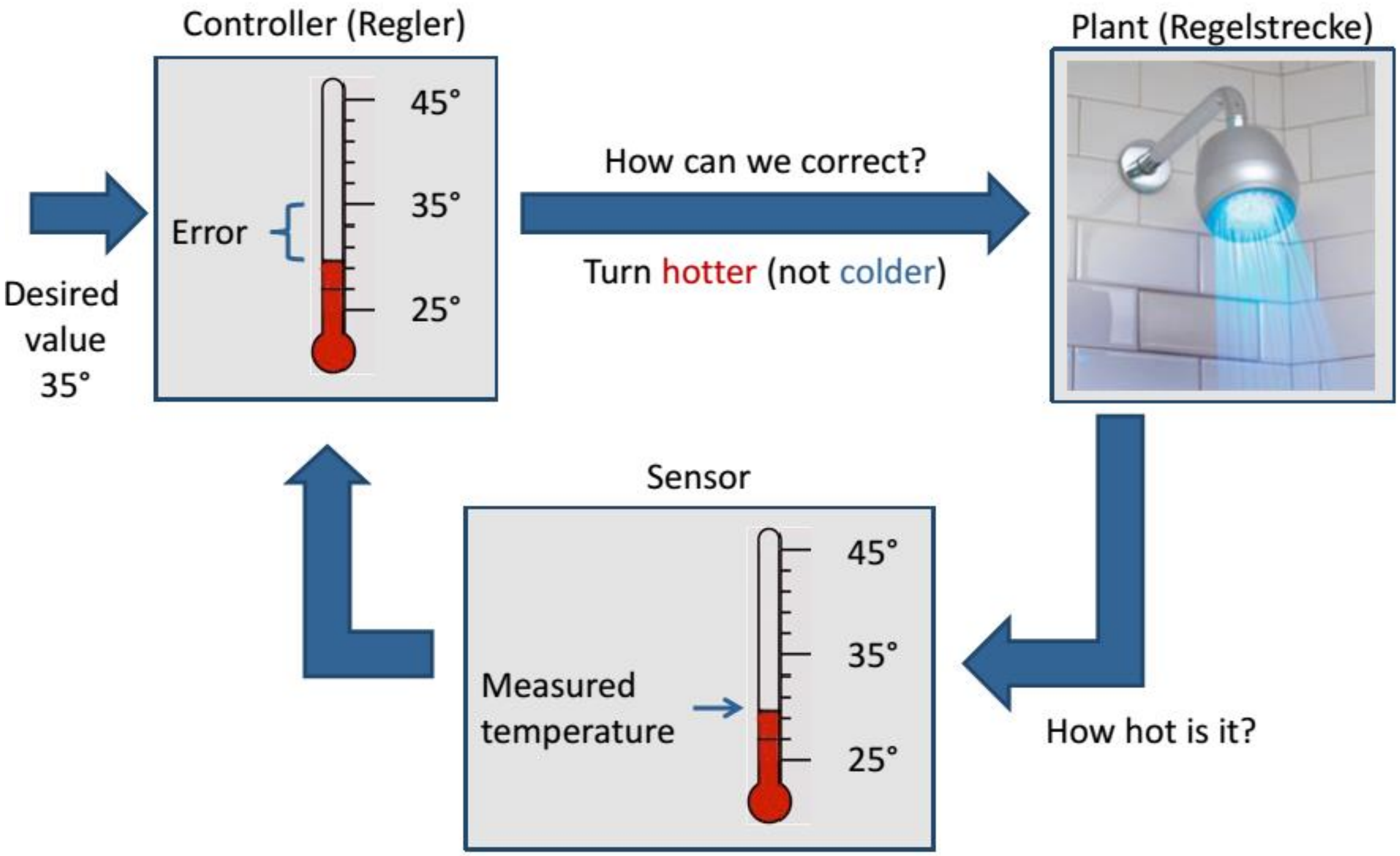
C Parasite drag is the drag incurred from the nonlifting portions of the aircraft

Summary

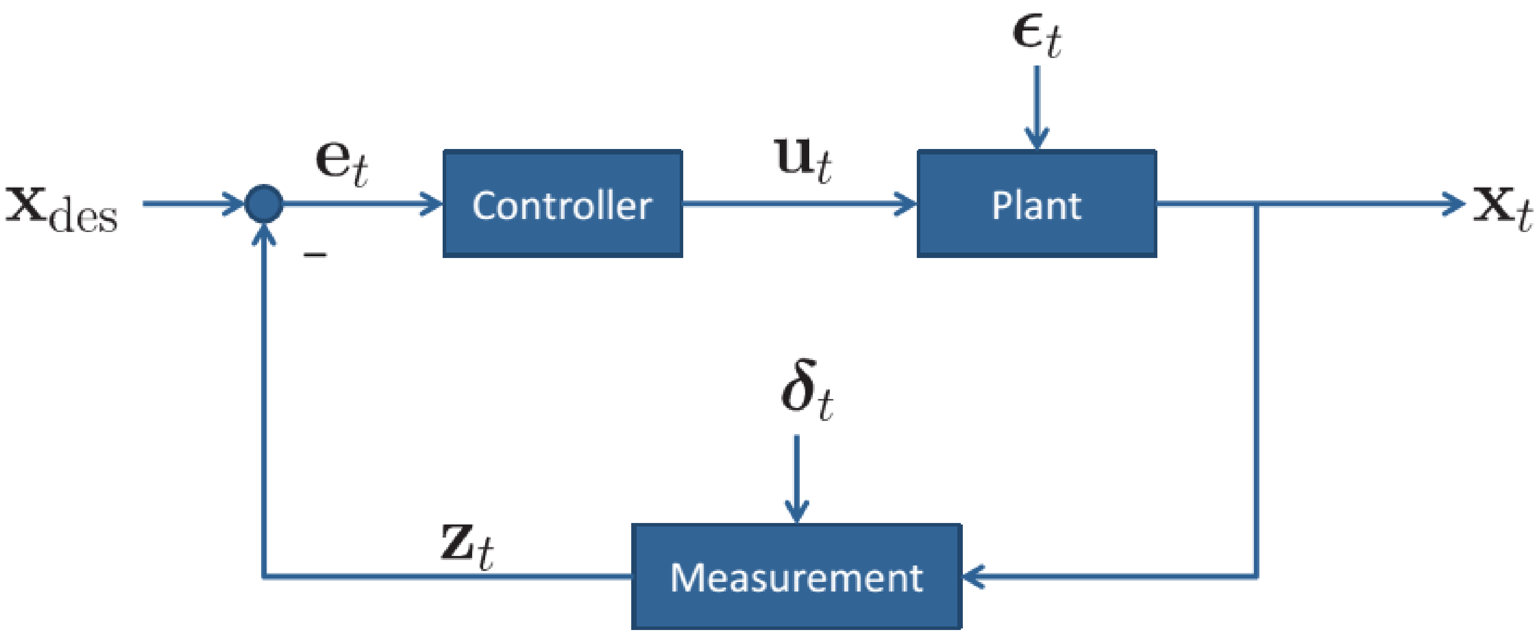
- Each rotor produces thrust and torque
- Pitch, Roll are controlled by unbalanced thrusts
- Yaw is controlled by unbalanced torques
- Thrust and torque can be converted in to the rotation speed of the motors linearly.
- Drag mainly consists of induced drag in low-speed movements

Platform – Control

Feedback control



Control Block diagram



PID control

- Proportional-Integration-Derivative controller

$$u_t = K_P e_t + K_I \int e_t dt + K_D \dot{e}_t$$
$$e_t = (x_t^* - x_t)$$

K_P Proportional gain

K_I Integral gain

K_D Derivative gain

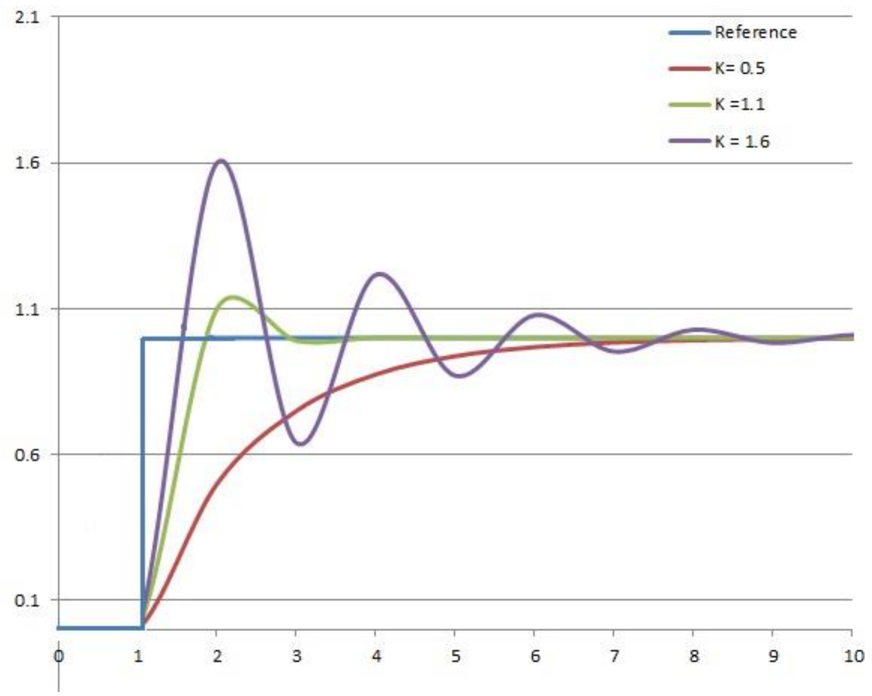
PID controllers are used in more than 95% of closed-loop industrial processes.[1]

[1] *Astrom K. J. and Hagglund T. H., "New tuning methods for PID controllers", Proceedings of the 3rd European Control Conference, 1995*

Proportional term

- A high proportional term *(Kp)* will have the effect of reducing the rise time but will cause oscillation.

$$u_t = K_P e_t$$

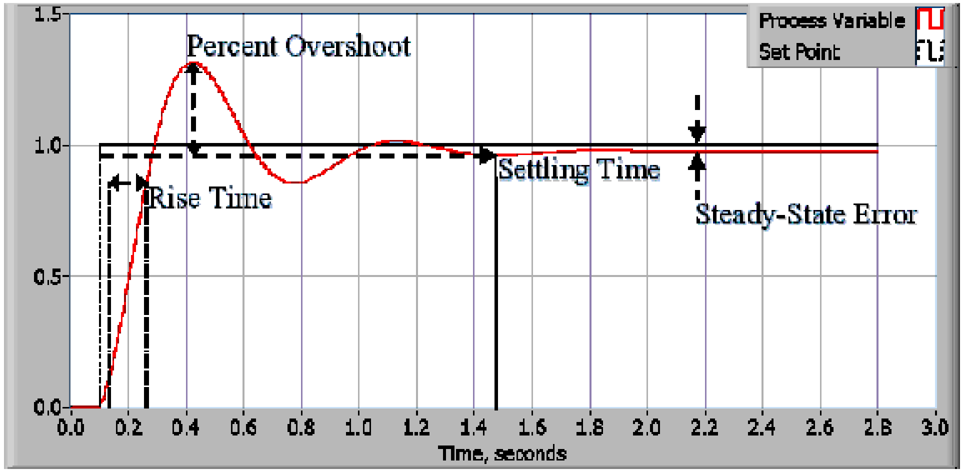


Integral term

- The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error.

$$u_t = K_I \int e_t dt$$

- It aims to reduce the steady-state error.
- Steady-state error can be caused by non-zero noises in feed-back measurements.

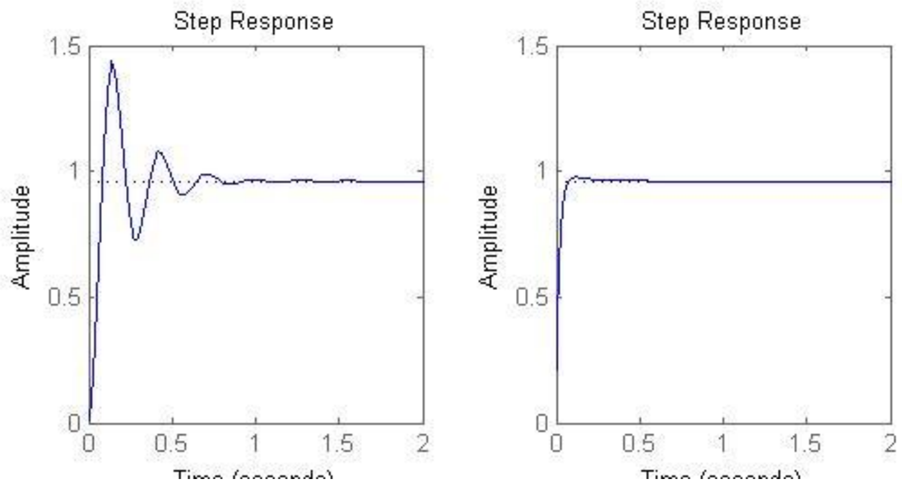


Derivative term

- The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain *Kd*

$$u_t = K_D \dot{e}_t$$

- Derivative term is used *to* decrease oscillations, since it yields large influence when the error changes rapidly.



PID controller

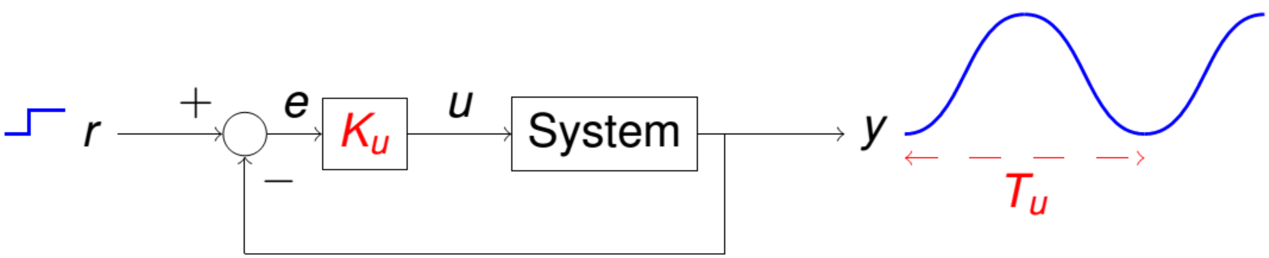
- The effect of increasing the PIC control parameters K_p, K_I, K_D are summarized as

Response	Rise Time	Overshoot	Settling Time	S-S Error
K_P	Decrease	Increase	NT	Decrease
K_I	Decrease	Increase	Increase	Eliminate
K_D	NT	Decrease	Decrease	NT

- Use K_P to decrease the rise time.
- Use K_D to reduce the overshoot and settling time.
- Use K_I to eliminate the steady-state error.

Find PID parameters efficiently

- Ziegler–Nichols method is a heuristic method of tuning a PID controller.
- It performs by setting the I and D gains to zero and increase P gain so as to reach the ultimate gain K_u at which the control loop has stable and consistent oscillations.



Controller	K	T_i	T_d	T_p
P	$0.5K_u$			T_u
PI	$0.4K_u$	$0.8T_u$		$1.4T_u$
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$	$0.85T_u$

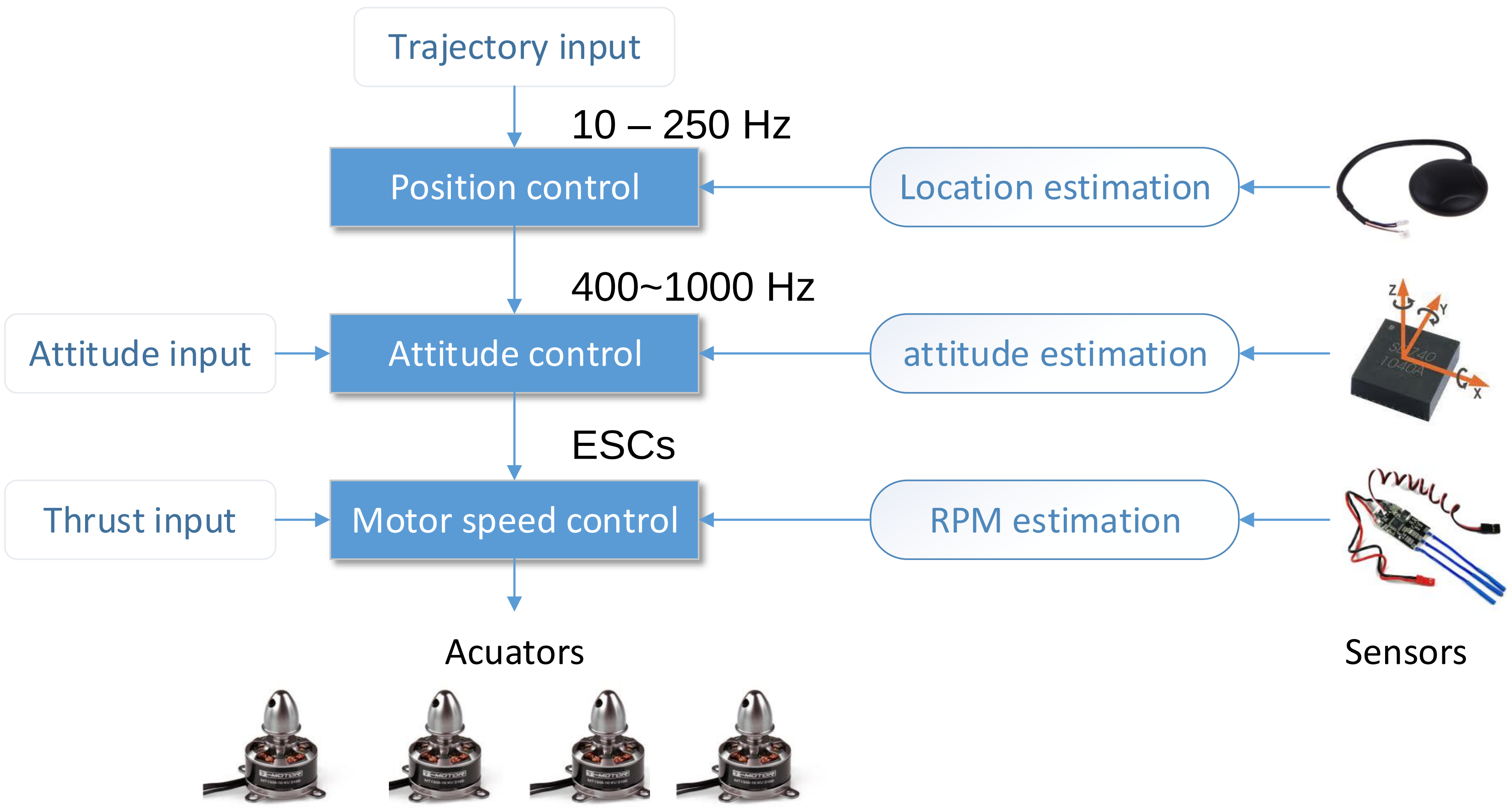
Practical consideration

- To design a controller in practice, we need to also take care of the following issues:
  - Dynamic of the plant
  - Stability analysis by Laplace Transform
  - Measurement noise
  - Measurement delay

Summary

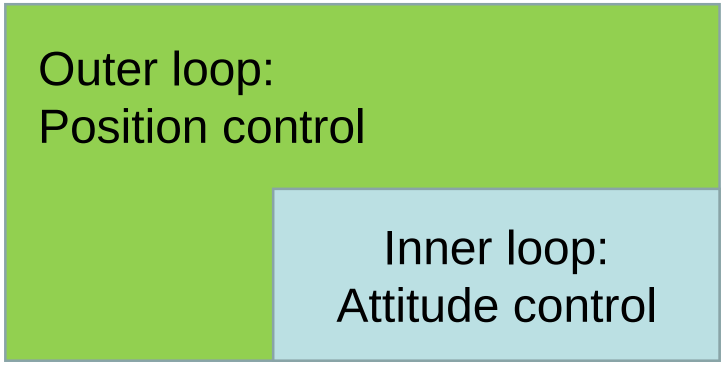
- Feedback is generally required for designing a stable controller.
- PID controller is simple but work very well for most applications.
- P is used control the rise time
- I is used to reduce the steady-state error
- D is used to reduce the overshoots and the settling time

Pipeline of controlling a quadcopter



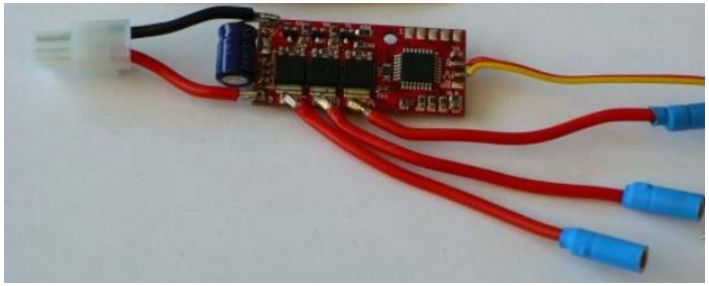
Control frequency

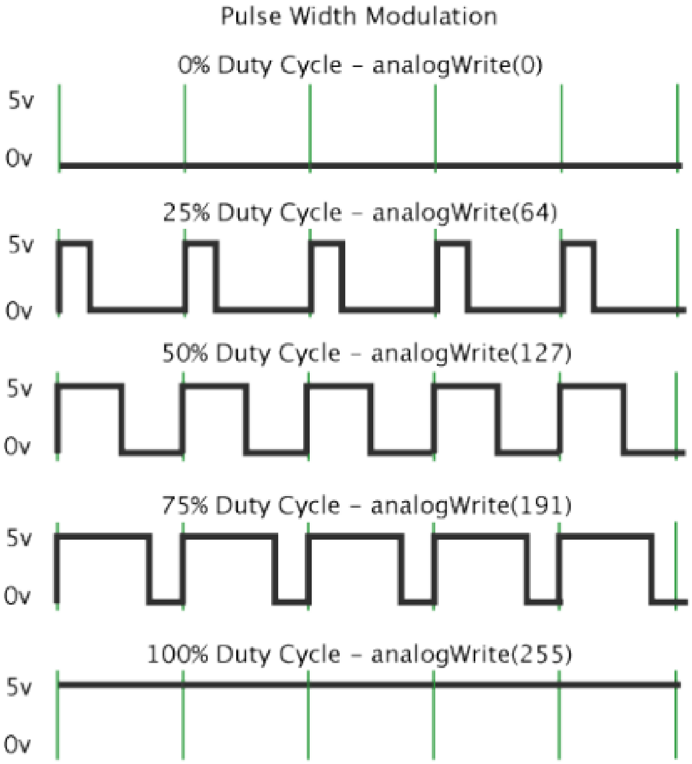
- Motor control happens on motor boards (ESCs) (controls every motor tick)
- Attitude control implemented on microcontroller with hard real-time (at 400~1000 Hz)
- Position control (at 10 – 250 Hz)
- Trajectory (waypoint) control (at 0.1 – 1 Hz)



Motor speed control

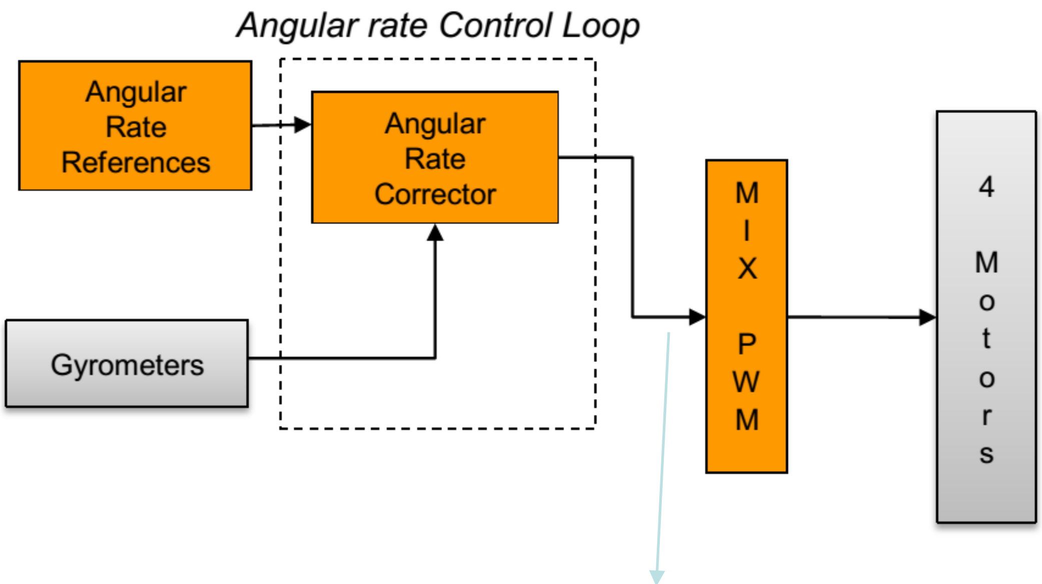
- Done by hardware: Brushless motor + Motor controller





Attitude control

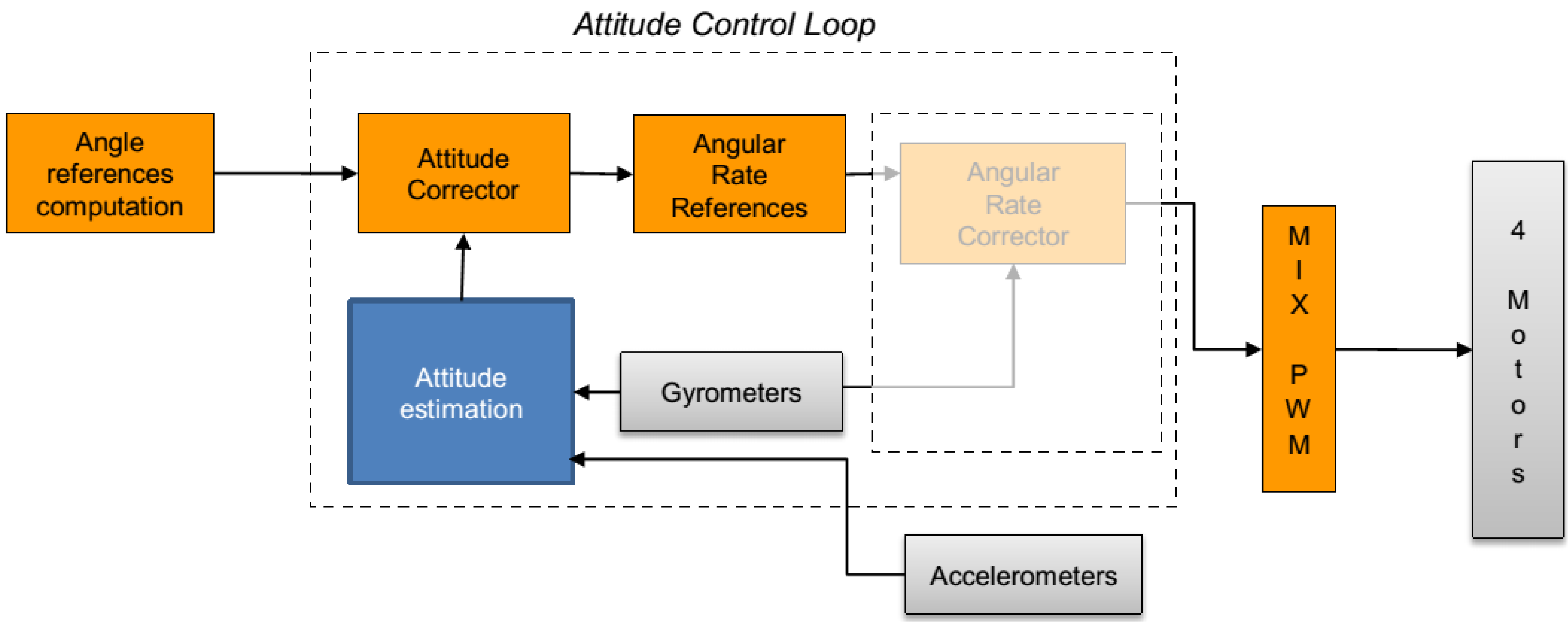
- Agular rate controller



$$\tau_b = \omega_b \times I_b \omega_b + I_b \dot{\omega}_b$$

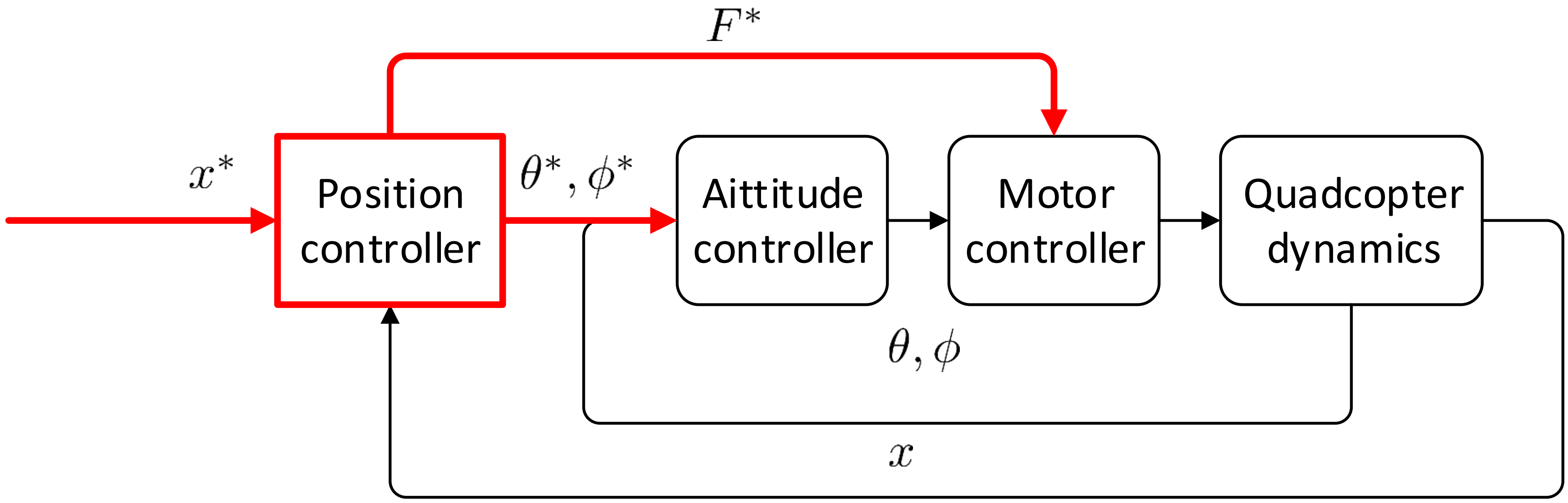
Attitude estimation

Attitude control



Position/Trajectory control

- Given a specified trajectory $x^*(t)$, we want to control the quadcopter flying along it.



Position/Trajectory control

- Step I: get the command acceleration vector use PID controller

$$a_t^* = K_P e_t + K_I \int e_t dt + K_D \dot{e}_t$$
$$e_t = (x_t^* - x_t)$$

- Step II: convert the command acceleration to thrust, pitch and roll set points

$$ma = RT_b - mg$$

Positon/Trajectory control

- Get the control outputs

$$ma = RT_b - mg$$

$$R = R_z(\psi)R_y(\phi)R_x(\theta)$$

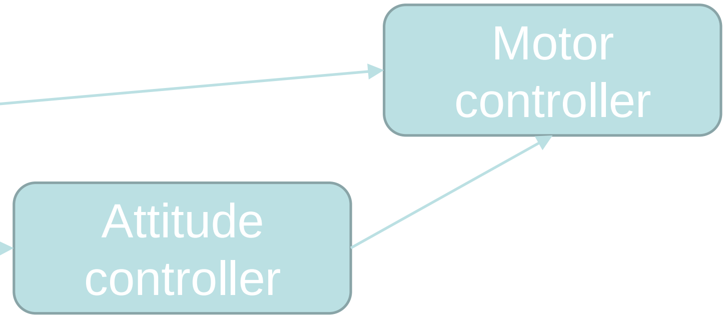
$$T_b = [0, 0, F]^T$$

$$a = f(\theta, \phi, F)$$

$$F^* = f^{-1}(a^*)$$

$$\theta^* = f_2^{-1}(a^*)$$

$$\phi^* = f_3^{-1}(a^*)$$



Summary

- Cascaded control scheme is used for controlling Quadcopter:
  - Position/Trajectory controller (10 – 250 Hz)
  - Attitude controller (400~1000 Hz)
    - Attitude rate controller
  - Motor speed controller

Choose a development platform

Commercial solutions

- DJI's phantom, or inspire series

DJI's mobile SDK:

Control of
- Roll, pitch, Yaw and throttle
- Waypoint
- Gimbal

Access of
- Real-time video sequence
- System status and flight data

But only supports development of ios and android applications.

Choose a development platform

Commercial solution

- DJI's M100

DJI's onboard SDK:

+More control (such as directly control the motor speed)
+ Real-time High frequency flight data
+ Can mount customized hardware

- Need onboard computer run with Linux + ROS
- For highly professional developers
- Too big for indoor applications
- Too expensive

Choose a development platform

Commercial solution

- Parrot ardrone & Bebop drone

ARDrone 2.0

- Roll, pitch, yaw and vertical speed control
- Real-time video sequence 1280x720p@30hz
- Real-time flight data @200hz
- Well supported by the community
- Safe and strong

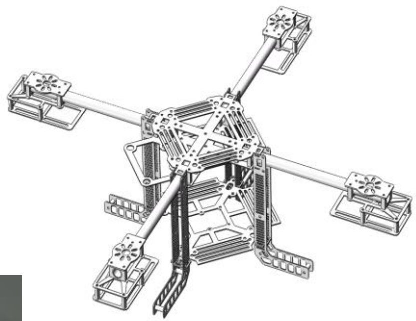
Bebop

+ way points control
+ GPS supported
- 640x368 @30hz
- Flight data @5hz

Open-source solution:

- Pixhawk flight control unit + Odroid xu3/4 (onboard computer) + DIY quadcopter





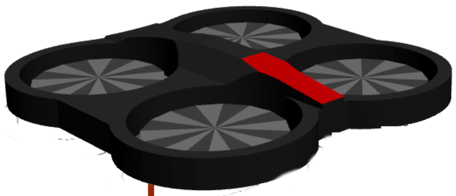
Use open source drone API to control the drone.
+ Highly customized
+ Source codes are available
+ Mount all kinds of sensors
- To write sensor data acquisition programs
- Need to be well tested



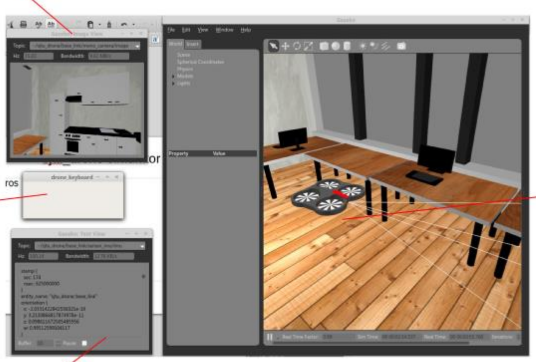

Sjtu-drone : ROS+Gazebo Qaudcoptor simulator

sjtu_drone simulator

- https://bitbucket.org/dannis/sjtu_drone
- Motivated by tum_simulator
- New features:
  - Support the newest version of ROS and Gazebo
  - Keyboard controller
  - Bug fix
  - Remove the dependence on gazebo-ros package
  - Many new sensors
  - New scenes

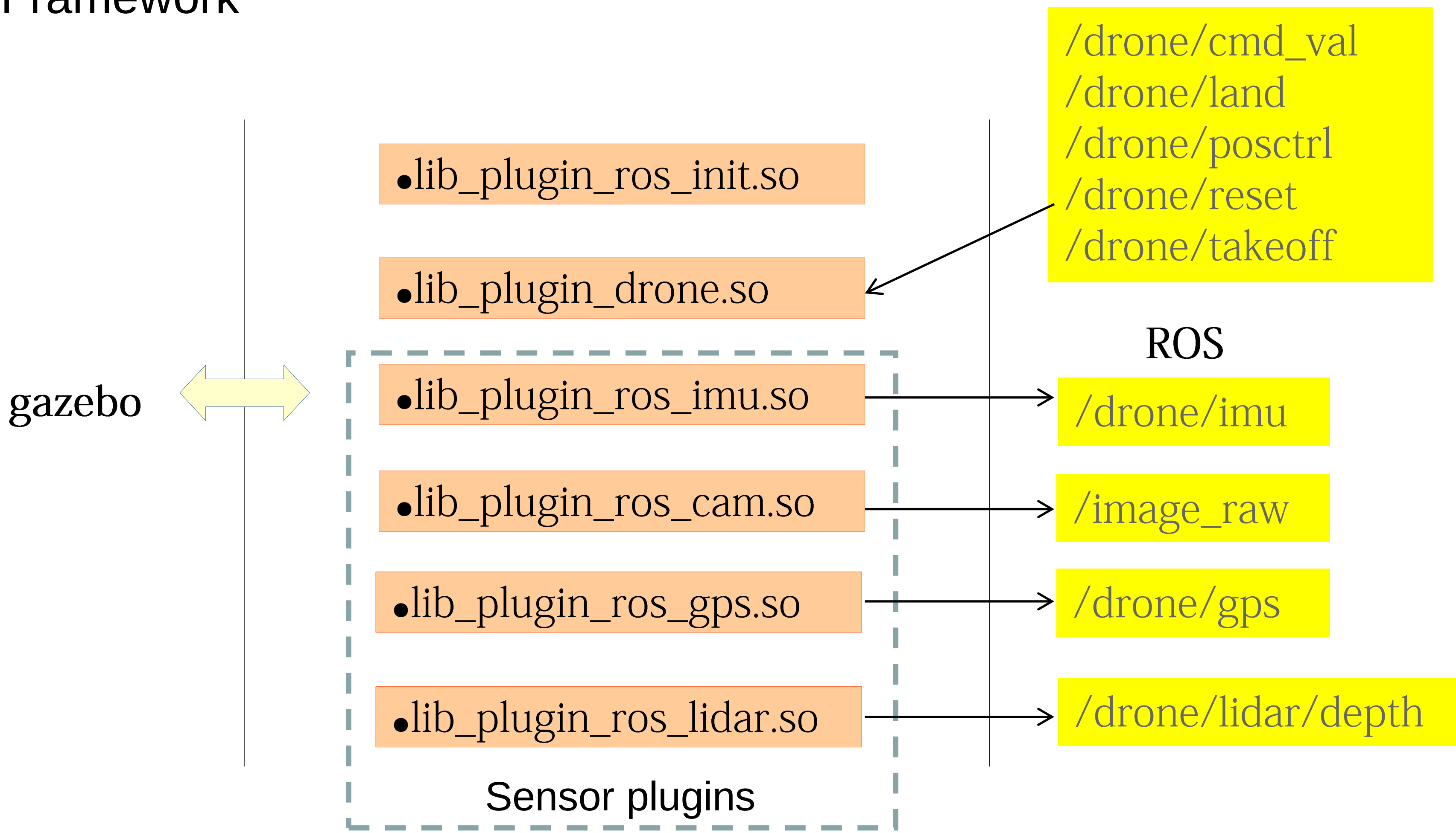


IMU
Compass
Camera
Barometer
Sonar
Lidar
GNSS receiver
……



sjtu_drone simulator

Framework



Directory structure

- bin (store binary executables)
- plugins (store Gazebo plugins)
- build (automatically generated files by ROS)
- include (header files)
- src (source files)
- launch (ROS launch files)
- scripts (script executables)
- meshes (*.dae files)
- model (drone model files)
- worlds (world files)

Code structure

Plugins:

- lib_plugin_ros_init.so (for initialize the ROS)
- lib_plugin_drone.so (PID controller for ardrone)
- lib_plugin_ros_imu.so (to publish the imu information on ROS topics)
- lib_plugin_ros_cam.so (to publish the image information on ROS topics)
- lib_plugin_xx.so (other sensors)

Program:

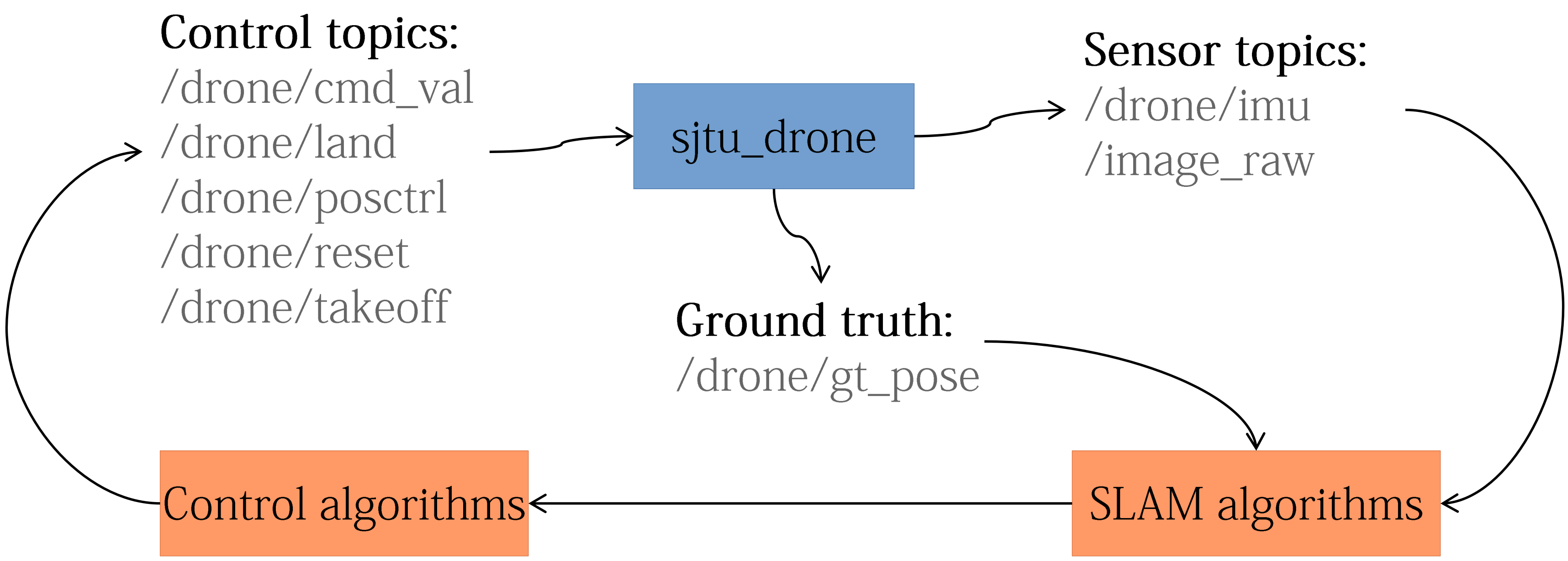
- drone_keyboard (send commands to the drone)
- spawn_drone (spawn a drone model in Gazebo)

Scripts:

- start_gzserver (set the environment variables, start Gazebo server)
- start_gui (start the Gazebo client)
- spawn_model (spawn a drone model in Gazebo)
- nogui.launch (launch file for no gui)
- start.launch (launch file for calling all scripts)

sjtu_drone as a testbed：

- SLAM algorithm
- Control algorithm



sjtu_drone simulator

- An example - Attitude estimation from IMU data

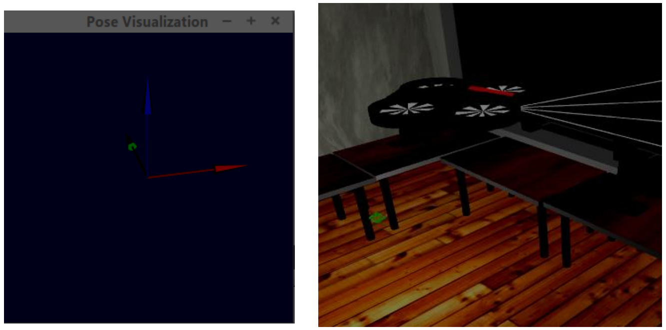
sjtu_drone → Sensor topics: /drone/imu

sjtu_drone → Ground truth: /drone/gt_pose

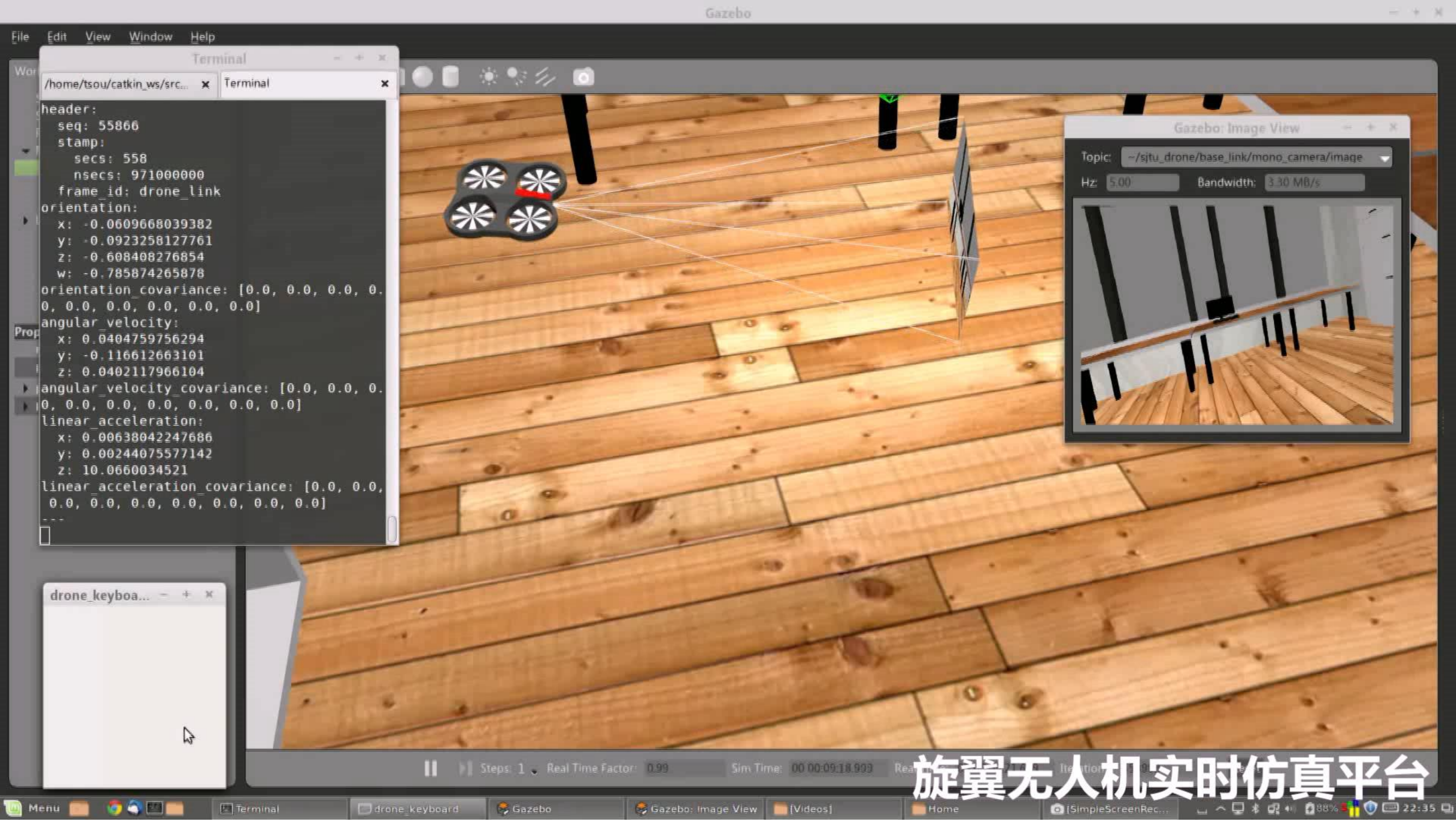
Sensor topics: /drone/imu → Attitude estimation

Ground truth: /drone/gt_pose → Attitude estimation

Attitude estimation → Result: /pose → Visualization



sjtu_drone


Gazebo
File Edit View Window Help
Terminal
/home/tsou/catkin_ws/src...
Terminal
header:
seq: 55866
stamp:
secs: 558
nsecs: 971000000
frame_id: drone_link
orientation:
x: -0.0609668039382
y: -0.0923258127761
z: -0.608408276854
w: -0.785874265878
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
x: 0.0404759756294
y: -0.116612663101
z: 0.0402117966104
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
x: 0.00638042247686
y: 0.00244075577142
z: 10.0660034521
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
drone_keyboa...
Gazebo: Image View
Topic: ~/sjtu_drone/base_link/mono_camera/image
Hz: 5.00
Bandwidth: 3.30 MB/s
Steps: 1
Real Time Factor: 0.99
Sim Time:
旋翼无人机实时仿真平台
Menu
Terminal
drone_keyboard
Gazebo
Gazebo: Image View
[Videos]
Home
[SimpleScreenRec...
22:35

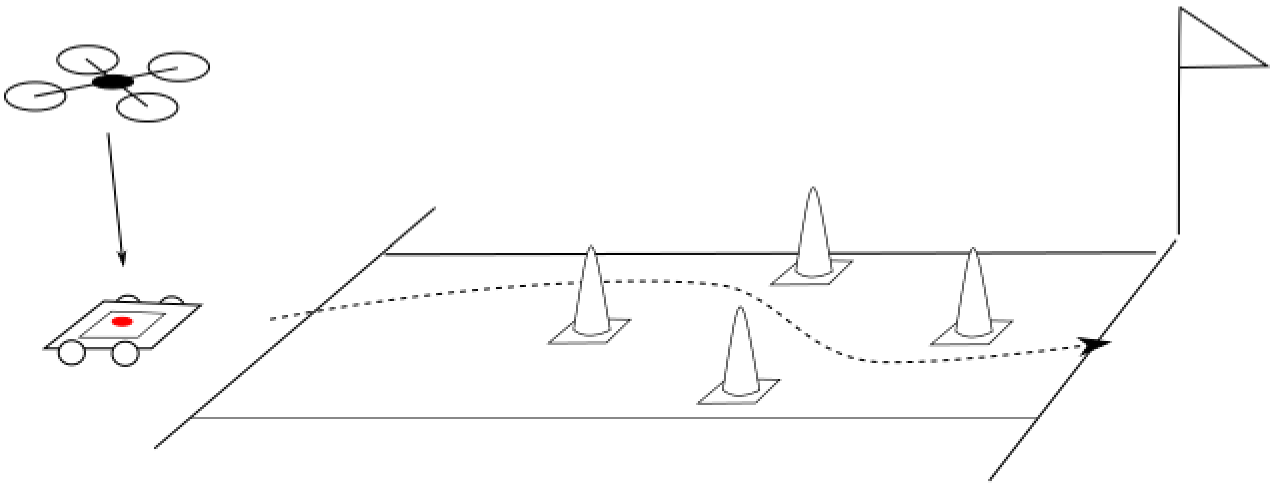
Contests

UAV contest in SJTU

- About Competition on Micro-UAV Related Technologies in Shanghai Jiao Tong University
  - Started in Oct. 2013 and have hold twice.
  - The price for first place winner is 20,000 Yuan

Projects

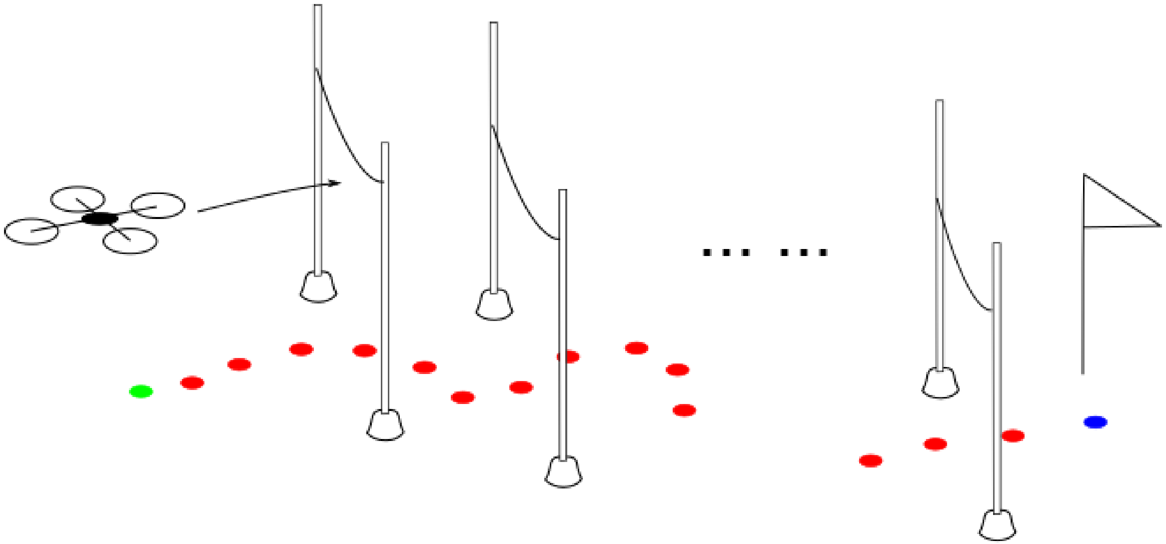
Ground vehicle tracking (2014)



- Use self-designed pattern attached on the ground vehicle
- Track the moving vehicle on the ground to reach the destination.
- The vehicle is manually controlled by remote controller
- The faster more scores are gained.

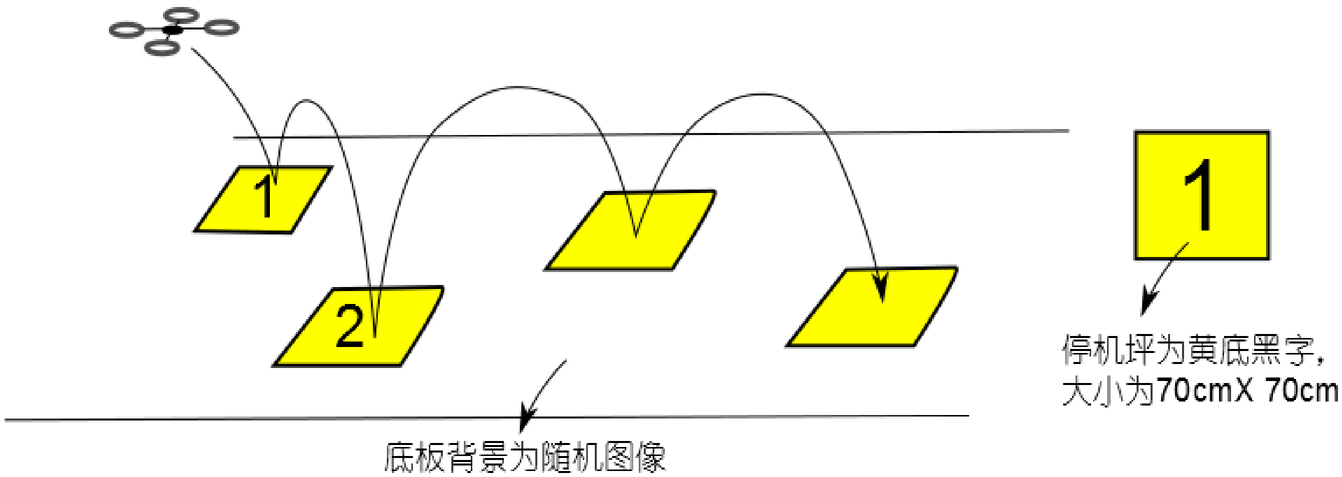
Projects

Path following (2014)



- Recognize the colored dots on the ground
- Fly through all the obstacle rods
- Travel as fast as possible
- Automatically take off and land

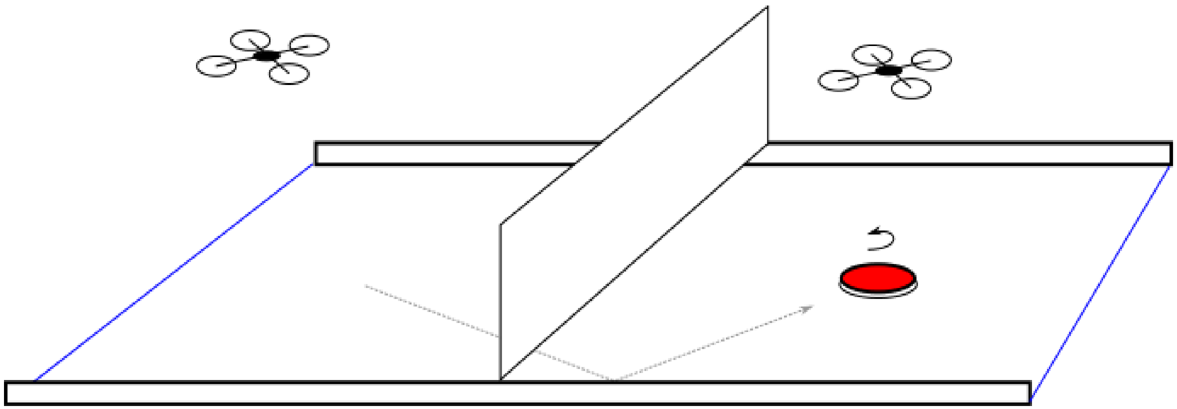
Crazy landing (2015-2016 preliminary round)

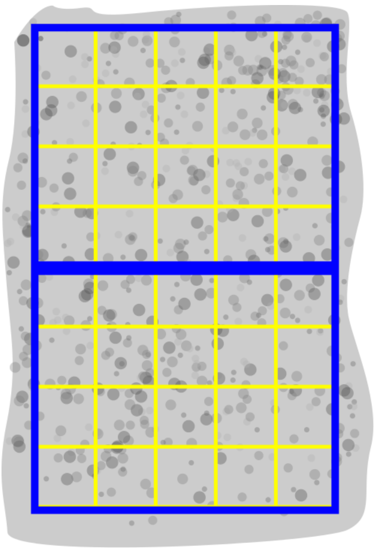


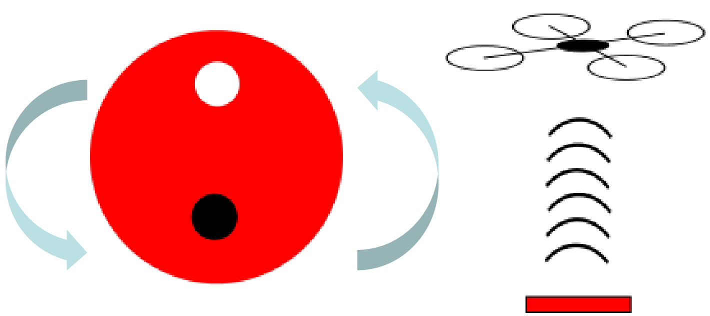
- Recognize the numbered squares and land on them according to the order of numbers.
- It is recognized as failure if the drone lands on the wrong square or in the non-square area.

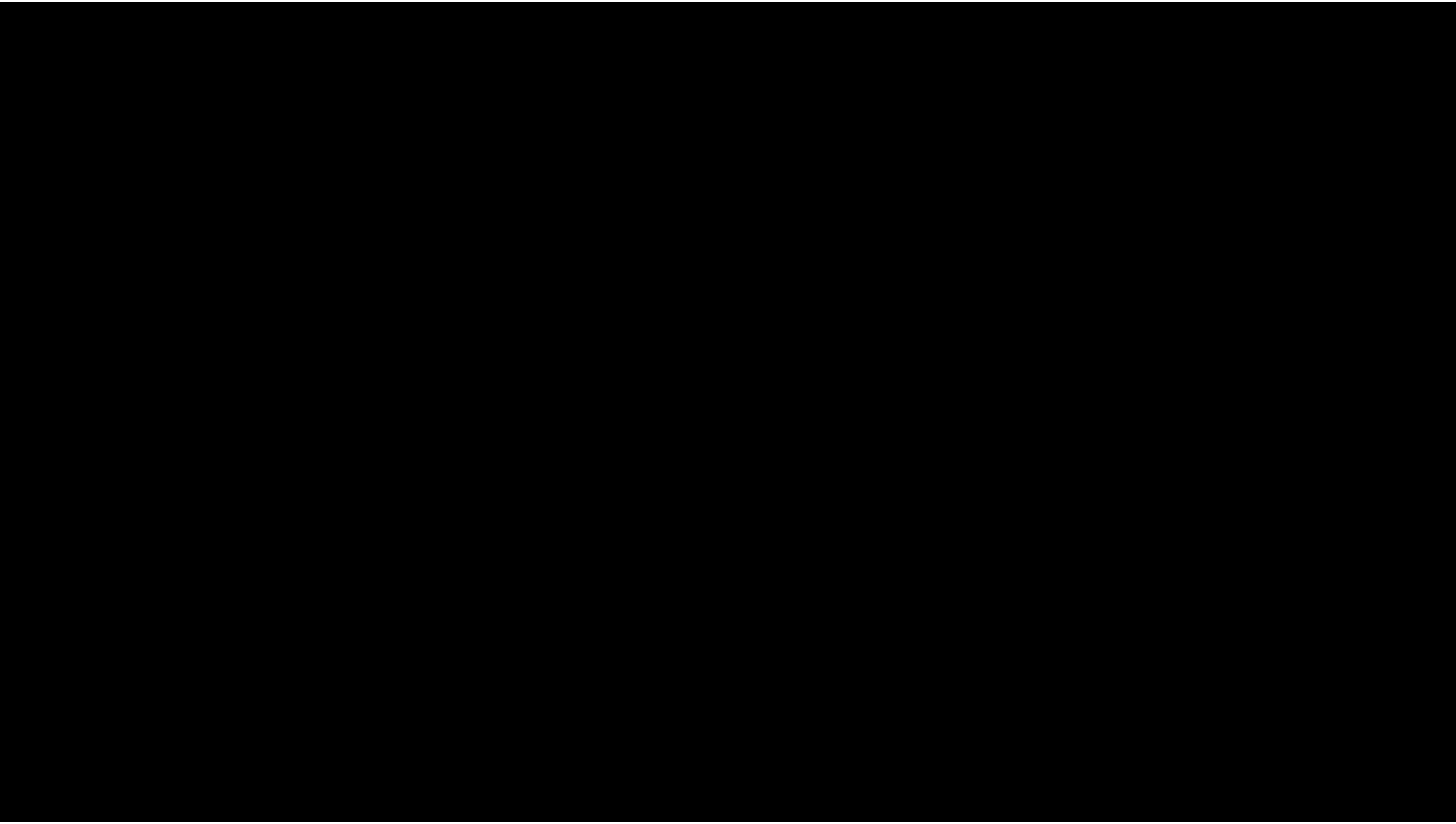
Projects

- Kick the ‘ball’ (2016 final match)










优酷


东方财经·浦东
直播
浦东新闻谱
新闻制高点
APEX NEWS
役部队士兵，曾在阿富汗服役。
(资讯):韩国

- Contest about AI technology (computer vision, learning, autonomous exploration) on micro drones

http://drone.sjtu.edu.cn