Overview of Autonomous Quadrotor UAV Research

AI Enabled Control Engineering

2026 Summer

Lecture roadmap

This lecture gives a system-level overview of autonomous quadrotor UAV research.

1. Broad UAV applications.
2. Basic quadrotor dynamics and differential flatness.
3. What autonomy means for UAV systems.
4. External localization and onboard perception.
5. Path planning, trajectory generation, high-level control, and PX4.
6. End-to-end autonomous flight with AI: potential and limitations.

Low-altitude economy and UAV applications

The low-altitude economy refers to economic activities that use low-altitude airspace for transportation, inspection, emergency response, public services, and other aerial operations. It is supported by UAVs, eVTOL aircraft, general aviation platforms, ground infrastructure, communication systems, and airspace management.

Main components include:

- low-altitude airspace resources;
- aerial vehicles such as UAVs and eVTOLs;
- takeoff, landing, charging, and communication facilities;
- operation, supervision, and air-traffic management systems.

eVTOL

Application scenario I: UAV logistics

UAV logistics is one of the most visible application scenarios.

- Urban last-mile delivery.
- Medical supply transportation.
- Mountain, island, and disaster-area delivery.
- Industrial park.

Compared with ground vehicles, UAVs can use three-dimensional space and may bypass some limitations of road traffic.

UAV logisticsMeituan

Application scenario II: infrastructure inspection

Inspection is another important UAV application.

- Powerline inspection.
- Bridge and tower inspection.
- Tunnel inspection.
- Factory, warehouse inspection.

UAVs can provide flexible viewpoints and reduce the need for human workers to enter dangerous or difficult areas.

Tunnel inspection in Hongkong

Application scenario III: home interaction

Indoor and home scenarios show another possible direction of UAV applications.

- Object search.
- Human-UAV interaction.
- Lightweight aerial manipulation.
- Robotic assistance in daily environments.

Recent aerial manipulation research suggests that UAVs may become active robotic assistants rather than only flying cameras.

Human-UAV interaction
Zhejiang University Fastlab

Many UAV applications use quadrotors

In many of the above scenarios, the most common platform is the quadrotor.

We first need a simple dynamic model. A dynamic model describes how motor commands change the motion of the vehicle.

For this introductory lecture, we only want to understand three basic ideas:

- how the propellers generate thrust;
- how thrust and gravity determine translational motion;
- how different motor speeds produce rotation.

A simple model is enough to understand why quadrotors can hover, move horizontally, and follow trajectories.

Basic quadrotor structure

A typical quadrotor includes:

- airframe;
- four motors;
- four propellers;
- electronic speed controllers;
- flight controller;
- battery;
- optional onboard computer and sensors.

The flight controller changes the speeds of the four motors to control the motion of the vehicle.

DJI UAV

Position, attitude, and motion

To describe a quadrotor, we usually care about two types of quantities.

- **Position:** where the UAV is in space.
- **Attitude:** how the UAV body is oriented.

Position can be written as

$$p = \begin{bmatrix} x & y & z \end{bmatrix}^\top.$$

Attitude is often described by roll, pitch, and yaw:

$$\phi \quad \theta \quad \psi.$$

- Roll ϕ: left-right tilting.
- Pitch θ: forward-backward tilting.
- Yaw ψ: heading direction.

Propellers generate thrust

Each propeller pushes air downward and generates an upward thrust force.

A simple approximation is

$$f_i = k_f \Omega_i^2,$$

where

- f_i is the thrust generated by rotor i;
- Ω_i is the rotor speed;
- k_f is a thrust coefficient.

The total thrust is

$$T = f_1 + f_2 + f_3 + f_4.$$

In words: faster propellers generate larger thrust.

Vertical motion and hover

The simplest motion is vertical motion.

- If total thrust is smaller than weight, the UAV goes down.
- If total thrust is larger than weight, the UAV goes up.
- If total thrust equals weight, the UAV can hover.

The hover condition is

$$T = mg.$$

Here, m is the mass of the UAV and g is gravitational acceleration.

Key point

Hovering is the most basic equilibrium of a quadrotor.

Tilting produces horizontal motion

A quadrotor does not have a separate horizontal engine. Its propellers mainly generate thrust along the body vertical direction.

When the vehicle tilts, the thrust vector is no longer purely vertical. It can be decomposed into a vertical component and a horizontal component:

$$T = T_{\text{vertical}} + T_{\text{horizontal}}.$$

- the vertical component mainly balances gravity;
- the horizontal component produces horizontal acceleration;
- forward tilt produces forward acceleration;
- sideways tilt produces lateral acceleration;
- larger tilt usually means larger horizontal acceleration.

A simple translational model

A compact way to write the translational dynamics is

$$m\ddot{\mathbf{p}} = T\mathbf{b}_3 - mg\mathbf{e}_3.$$

Here:

- $m\ddot{\mathbf{p}}$ means mass times acceleration.
- T is the total thrust.
- $\mathbf{b}_3$ is the direction of the UAV body vertical axis.
- $mg\mathbf{e}_3$ is gravity.

This equation simply says:

$$\text{mass} \times \text{acceleration vector} = \text{thrust vector} - \text{gravity vector}.$$

This is Newton's second law applied to a quadrotor.

How motor differences create rotation

The four motors do not only control vertical thrust. Their differences also create roll, pitch, and yaw motion.

- Increasing thrust on the left side can roll the UAV to the right.
- Increasing thrust at the back can pitch the UAV forward.
- Different propeller rotation directions create yaw torque.

Motor-speed pattern	Main effect
All motors increase together	Move upward
Left-right imbalance	Roll motion
Front-back imbalance	Pitch motion
Clockwise-counterclockwise imbalance	Yaw motion

Abstract control inputs

At the high-level modeling level, we often do not directly discuss the four motor speeds.

Instead, we use four abstract inputs:

$$u = \begin{bmatrix} T & \tau_x & \tau_y & \tau_z \end{bmatrix}^\top.$$

They represent:

- T: total thrust;
- τ_x: roll torque;
- τ_y: pitch torque;
- τ_z: yaw torque.

This abstraction is useful because it separates high-level control from low-level motor allocation.

Quadrotor is underactuated

A quadrotor moves in three-dimensional space. Its full pose includes position and attitude:

$$x,\ y,\ z,\ \phi,\ \theta,\ \psi.$$

That is six quantities.

However, the quadrotor has only four main control inputs:

$$T,\ \tau_x,\ \tau_y,\ \tau_z.$$

Therefore, it cannot independently choose all positions and all angles at the same time.

Key point

The quadrotor must use attitude changes to generate horizontal motion. This is why it is called an underactuated system.

Differential flatness: what it means

Quadrotors have an important property called differential flatness.

For an introductory understanding, we can state it as follows:

- If we design a smooth trajectory for position x, y, z and yaw angle ψ,
- then the desired velocity, acceleration, attitude, thrust, and other control-related quantities can be computed from this trajectory and its derivatives.

The commonly used flat outputs are

$$x, \quad y, \quad z, \quad \psi.$$

Why differential flatness matters

Differential flatness is important because it makes trajectory planning much easier.

Without this property, we might need to directly plan all position, velocity, attitude, and angular velocity variables.

With differential flatness, we can first plan a smooth curve in 3D space:

$$x(t), \quad y(t), \quad z(t),$$

and then compute the corresponding desired attitude and thrust.

Key point

This is why many quadrotor trajectory methods focus on generating smooth position curves.

Beyond quadrotors: heterogeneous aerial robots

Quadrotors are common, but they are not the only aerial robot design.

- Single-rotor vehicles.
- Tilt-rotor vehicles.
- Fixed-wing multirotor hybrid vehicles.
- Morphing aerial robots.
- Aerial robots with manipulators.

These platforms are usually designed for specific goals, such as longer endurance, higher speed, narrow-space traversal, or aerial manipulation.

ETH Zurich Monospinner

A short note

The previous slides only give a first overview of quadrotor UAVs. They are enough for building intuition, but not enough for a complete theoretical understanding.

To fully understand quadrotor modeling, trajectory generation, and control, further study is needed in:

- theoretical mechanics and rigid-body dynamics;
- coordinate transformations and attitude representation;
- rotation matrices, Euler angles, and quaternions;
- Lie groups and Lie algebras, especially $SO(3)$ and $SE(3)$;

For the differential flatness property of quadrotors, useful references include:

- Mellinger and Kumar, *Minimum Snap Trajectory Generation and Control for Quadrotors*: Minimum snap trajectory generation and control for quadrotors
- Faessler, Franchi, and Scaramuzza, *Differential Flatness of Quadrotor Dynamics Subject to Rotor Drag*

What does autonomy mean for UAVs?

An autonomous UAV is not simply a remotely piloted aircraft. It is a robotic system that can estimate its state, perceive the environment, plan motion, and execute control commands.

A minimal autonomous UAV system must answer four questions:

- **Localization:** Where am I?
- **Perception:** What is around me?
- **Planning:** How should I move to the target?
- **Control:** How can I track the planned motion safely?

Key point

Dynamics is the foundation, but autonomy requires perception, planning, control, and execution.

Localization: knowing the UAV's own state

The first sensing problem is localization. Localization means estimating the UAV's own position and attitude.

A basic state description includes position

$$\mathbf{p} = \begin{bmatrix} x & y & z \end{bmatrix}^\top,$$

velocity

$$\mathbf{v} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^\top,$$

and attitude, often described by roll, pitch, and yaw:

$$\phi, \quad \theta, \quad \psi.$$

In simple terms, localization answers two questions:

- **Position:** where is the UAV?
- **Attitude:** how is the UAV oriented?

Perception: understanding the surrounding environment

Localization tells the UAV its own state. But autonomous flight also requires information about the environment.

Perception is used to obtain information such as:

- obstacles around the UAV;
- free space for safe flight;
- local or global maps of the environment.

Different sensors can be used for localization and perception.

This leads to the next part: sensors and perception systems for UAVs.

Two classes of UAV localization and perception

UAV sensors can be roughly divided into two classes.

Class	Typical devices	Main function
External localization	GPS RTK, motion capture	Provide external position or pose measurements for the UAV.
Onboard perception	IMU, camera, RGB-D camera, LiDAR	Support self-localization, mapping, obstacle detection, and safe flight.

External localization is convenient for experiments and open outdoor environments. Onboard perception is closer to real autonomous deployment.

Outdoor localization: GPS and RTK

GPS provides global positioning outdoors. RTK improves accuracy by using differential correction between a base station and a rover.

Advantages

- Large outdoor coverage.
- Useful for open-area flight.

Limitations

- Not available indoors.
- Degraded by urban area and occlusion.

GPS module

Indoor localization: motion capture system

A motion capture system is a common high-precision indoor localization tool.

- Multiple infrared cameras are installed around the lab.
- Reflective markers are attached to the UAV body.
- Multi-view geometry is used to recover rigid-body pose.
- The output frequency is high and latency is low.

In experiments, motion capture is often used as the ground-truth pose source.

Motion capture system

Motion capture workflow

A typical motion capture workflow includes:

1. Place multiple infrared cameras around the flight area.
2. Calibrate the motion capture volume.
3. Attach reflective markers to the UAV.
4. Define the UAV as a rigid body in the software.
5. Stream real-time position and attitude.
6. Send pose data to the control program through network or ROS.

In experiments, coordinate-frame conventions must be checked carefully.

Advantages and limitations of motion capture

Advantages:

- High accuracy.
- Stable and repeatable measurements.
- Convenient for controller validation.
- No need for onboard SLAM.
- Useful for evaluating tracking error.

Limitations:

- Works only in instrumented laboratories.
- Marker occlusion may cause pose loss.
- Does not represent real unknown-environment autonomy.

Key point

Motion capture is excellent for experiments, but it is not a replacement for onboard autonomy.

Onboard sensor: IMU

IMU stands for **Inertial Measurement Unit**. It is one of the most fundamental onboard sensors for UAVs.

IMU usually contains:

- **Gyroscope:** measures angular velocity.
- **Accelerometer:** measures specific force.

Key properties:

- High sampling frequency.
- Smooth short-term motion estimation.
- Long-term integration drift.
- Usually fused with other sensors.

NxtPX4 V2 Flight Controller

IMUs are commonly integrated into flight controller boards.

Onboard sensor: RGB camera

RGB camera provides rich visual information.

- Visual odometry.
- Object detection.
- Semantic understanding.
- Low cost and lightweight.
- High information density.

Limitations:

- Sensitive to illumination.
- Degenerates in low-texture scenes.

USB Camera

Onboard sensor: RGB-D camera

RGB-D camera provides both color images and depth images.

- Suitable for indoor obstacle avoidance.
- Provides local 3D structure directly.
- Useful for mapping and local planning.

Limitations:

- Limited sensing range.
- Outdoor sunlight can degrade performance.
- Transparent, reflective, or dark objects can be difficult.

Realsense D435

Onboard sensor: LiDAR

LiDAR directly measures 3D point clouds.

- High geometric accuracy.
- Less sensitive to illumination.
- Long sensing range.
- Useful for outdoor, indoor,underground, and industrial scenes.

Limitations:

- Transparent surfaces can cause missed detections.
- Dust, rain may scatter laser beams.
- Degeneracy may occur in long corridors or repetitive structures.

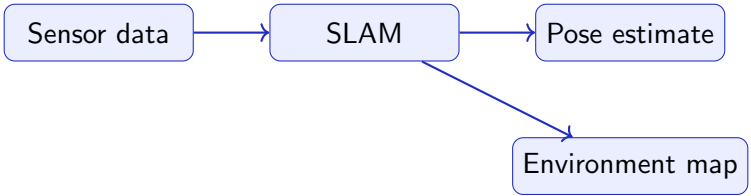
Livox MID360

What is SLAM?

SLAM means Simultaneous Localization and Mapping.

It answers two questions at the same time:

- **Localization:** What is the UAV pose in the map?
- **Mapping:** What is the structure of the surrounding environment?



SLAM methods by sensor type

- **Visual SLAM:** monocular, stereo, or RGB-D camera.
- **Visual-inertial SLAM:** camera and IMU fusion, often called VIO or VI-SLAM.
- **LiDAR SLAM:** 2D or 3D LiDAR.
- **LiDAR-inertial SLAM:** LiDAR and IMU fusion, often called LIO.

Representative SLAM algorithms

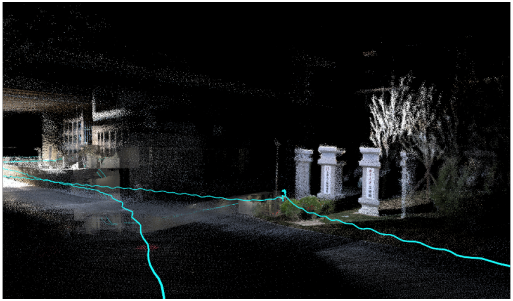
In practice, UAV localization and mapping systems are often implemented using existing open-source algorithms. Different algorithms are designed for different sensor combinations.

Algorithm type	**Representative algorithms**	**Typical sensor setup**
Visual-inertial odometry	VINS-Mono, VINS-Fusion, ORB-SLAM3	Camera + IMU. Commonly used for lightweight UAV localization.
LiDAR-inertial odometry	LOAM, LIO-SAM, FAST-LIO2	LiDAR + IMU. Commonly used for robust 3D localization and mapping.
LiDAR-visual-inertial odometry	R3LIVE, FAST-LIVO, FAST-LIVO2	LiDAR + camera + IMU. Combines geometric and visual information.

For UAV experiments, the choice of algorithm usually depends on the available sensors, onboard computing power, flight environment, and required localization accuracy.

Example SLAM visualization results

Fast-LIVO2 result1



Fast-LIVO2 result2

SLAM as an important research topic

SLAM is not only a tool for UAV localization. It is also one of the central research topics in robotics.

It is difficult because a robot must estimate its own motion and build a map at the same time, usually with noisy sensors, limited computation, and uncertain environments.

Knowledge needed for deeper study: rigid-body motion, coordinate transformations, probability and Bayesian estimation, Kalman filtering, nonlinear optimization, graph optimization, computer vision, point-cloud registration, and Lie groups such as $SO(3)$ and $SE(3)$.

Representative papers and systems

- **VINS-Mono**: T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, 2018.
- **FAST-LIO2**: W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast Direct LiDAR-Inertial Odometry," *IEEE Transactions on Robotics*, 2022.
- **FAST-LIVO2**: C. Zheng, W. Xu, Z. Zou, T. Hua, C. Yuan, D. He, B. Zhou, Z. Liu, J. Lin, F. Zhu, Y. Ren, R. Wang, F. Meng, and F. Zhang, "FAST-LIVO2: Fast, Direct LiDAR-Inertial-Visual Odometry," *IEEE Transactions on Robotics*, 2024.

From perception to planning

The perception module tells us where the UAV is and what the environment looks like.

The planning module asks:

- How should the UAV move from start to goal?
- How should it avoid obstacles?
- How can it maintain a safe distance?
- How can it reduce flight time?
- How can it reduce energy consumption, noise, or risk?

Key point

Planning is not simply drawing a line. It searches for executable motion under environmental and vehicle constraints.

Map representations for UAV navigation

After localization and mapping, the UAV usually does not use the raw SLAM result directly for planning. Instead, SLAM-based mapping results are often converted into navigation maps that are easier for planners to use.

Common map representations used in autonomous UAV navigation include:

- **Occupancy grid / voxel map:** divides the environment into cells or voxels and marks them as free, occupied, or unknown. It is commonly used for collision checking.
- **ESDF:** Euclidean signed distance field. It stores the distance from each position to the nearest obstacle and is widely used for safe trajectory optimization.
- **Topological map:** represents the environment as connected regions, nodes, and edges. It is useful for high-level route planning in large environments.

Path search in UAV planning

After obtaining a navigation map, the next step is to search for a feasible path from the start position to the goal position.

A path search algorithm usually needs:

- a start point and a goal point;
- a map representation, such as an occupancy grid, voxel map, ESDF, or topological graph;
- obstacle information for collision checking;
- a cost function that describes what kind of path is preferred.

What is a cost function?

In path planning, there are usually many possible paths from the start point to the goal point. A cost function is used to evaluate these candidate paths.

A simple interpretation is:

$$\text{cost function} = \text{a score assigned to each candidate path.}$$

The planner then searches for the path with the smallest cost:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} J(\mathbf{p}).$$

Here:

- $\mathbf{p}$ represents a candidate path;
- $J(\mathbf{p})$ is the cost of this path;
- a smaller cost means the path is more preferred.

For example, if the cost only measures path length, the planner will prefer the shortest path.

Cost functions are task-dependent

The simplest planning objective is shortest distance or shortest time. However, in real UAV tasks, the "best" path is not always the shortest one.

A path may be evaluated by several factors:

- **Distance cost:** fly a shorter path.
- **Safety cost:** stay away from obstacles, people, and no-fly zones.
- **Energy cost:** reduce battery consumption.
- **Passage-volume cost:** prefer wider and safer aerial corridors.
- **Noise cost:** reduce noise exposure over residential areas.

A general cost function can be written as a weighted combination:

$$J = w_1 J_{\text{distance}} + w_2 J_{\text{safety}} + w_3 J_{\text{energy}} + \cdots$$

Changing the weights changes the planner's preference.

Example: planning with clearance cost

If the objective only considers path length, the planner may fly too close to obstacles.

A simple safety-aware cost can be written as

$$J = \int_0^T \left(w_l \|\dot{p}(t)\| + w_s \frac{1}{d(p(t)) + \epsilon} \right) dt.$$

Here:

- $d(p(t))$: distance from current position to the nearest obstacle.
- w_l: length or time weight.
- w_s: safety weight.
- ϵ: small positive constant.

Different weights lead to different planned paths.

Representative path search methods

There are many path search methods in robotics and UAV navigation. For an introductory understanding, two representative families are enough.

Graph-search methods

- Examples: Dijkstra, A*.
- Usually work on grids, graphs, or roadmap structures.
- Suitable when the environment has been discretized.
- The search result is often a sequence of connected nodes or waypoints.

Sampling-based methods

- Examples: PRM, RRT, RRT*.
- Explore continuous space by sampling candidate states.
- Useful in high-dimensional or complex environments.
- The result usually still needs smoothing before flight.

For UAVs, path search gives a collision-free geometric route, while later trajectory generation makes it smooth and dynamically trackable.

Path smoothing and curve fitting

The output of path search is usually a geometric path, often represented by a sequence of waypoints.

However, this path is usually not suitable for direct flight.
If the UAV directly follows such a path, it may require large acceleration, angular velocity, or aggressive attitude changes.

Therefore, the path is usually smoothed before flight.

Common smoothing and curve-fitting methods include:

- **Bezier curves:** use control points to generate smooth geometric curves.
- **B-splines:** provide smooth curves with good local control.
- **Polynomial fitting:** fits a smooth curve through or near selected waypoints.

In simple terms, path smoothing converts a rough geometric route into a smoother curve that is easier for the UAV to follow.

Trajectory generation

A smoothed path only describes where the UAV should pass through. For real flight, the UAV also needs time, velocity, and acceleration information.

Trajectory generation converts a smooth geometric path into a time-parameterized motion plan:

$$\mathbf{p}(t), \quad t \in [0, T].$$

It usually provides desired quantities such as:

$$\mathbf{p}_d(t), \quad \mathbf{v}_d(t), \quad \mathbf{a}_d(t), \quad \psi_d(t).$$

The main goals are:

- assign timing to the path;
- satisfy velocity and acceleration limits;
- make the trajectory smooth and dynamically feasible.

Representative trajectory generation methods

- **Polynomial trajectory generation:** represents each trajectory segment as a polynomial:

$$\mathbf{p}(t) = \sum_{i=0}^{n} \mathbf{c}_i t^i.$$

- **Minimum-snap trajectory generation:** minimizes the fourth derivative of position:

$$\min_{\mathbf{p}(t)} \int_0^T \left\| \frac{d^4\mathbf{p}(t)}{dt^4} \right\|^2 dt.$$

- **MINCO-based trajectory optimization:** uses a compact minimum-control trajectory parameterization. It optimizes intermediate waypoints and segment times instead of all polynomial coefficients.

  Original paper: Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically Constrained Trajectory Optimization for Multicopters," IEEE Transactions on Robotics, vol. 38, no. 5, pp. 3259–3278, 2022.

The goal is to generate a trajectory that the UAV can actually track.

Path versus trajectory

These two concepts should not be confused.

- **Path:** a geometric curve in space.
- **Trajectory:** a time-parameterized curve.

Path:

$$p(s), \quad s \in [0, 1].$$

Trajectory:

$$p(t), \quad t \in [0, T].$$

A UAV controller ultimately needs a trajectory, not only a path.

Output of trajectory generation

A trajectory generation module usually outputs desired states:

$$p_d(t), \quad v_d(t), \quad a_d(t), \quad j_d(t), \quad \psi_d(t).$$

Where:

- p_d: desired position.
- v_d: desired velocity.
- a_d: desired acceleration.
- j_d: desired jerk.
- ψ_d: desired yaw angle.

These values are then converted by the high-level controller into commands that the flight controller can execute.

Task of the high-level controller

The high-level controller does not usually control the motors directly. It computes how the UAV should move.

Typical inputs:

$$p_d,\ v_d,\ a_d,\ \psi_d$$

and current estimated state:

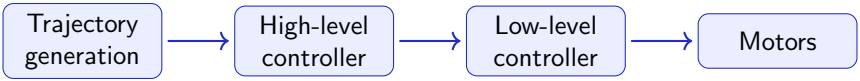
$$p,\ v,\ \phi,\ \theta,\ \psi.$$

Possible outputs include:

- Desired position or velocity setpoint.
- Desired acceleration.
- Desired attitude.
- Desired thrust and angular velocity.
- Desired thrust and torque.

From high-level commands to low-level control

After trajectory generation and high-level control, the UAV has a set of desired commands.
The low-level controller reads these high-level outputs and converts them into commands that can be executed by the motors.



In many UAV systems, the low-level controller is implemented using an autopilot software stack such as PX4.

PX4 and low-level flight control

PX4 is an open-source autopilot software stack widely used for UAV research and development.

In a typical autonomous UAV system, PX4 receives setpoints from the onboard computer and performs low-level control.

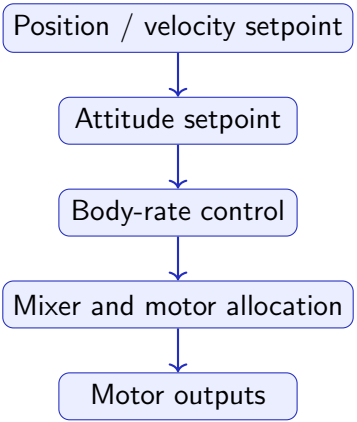
PX4 is mainly responsible for:

- reading low-level sensors such as IMU and barometer;
- estimating attitude and basic flight states;
- tracking position, velocity, attitude, or body-rate setpoints;
- running failsafe and flight-mode logic;
- converting control outputs into motor commands.

Flight controller hardware

Basic control structure inside PX4

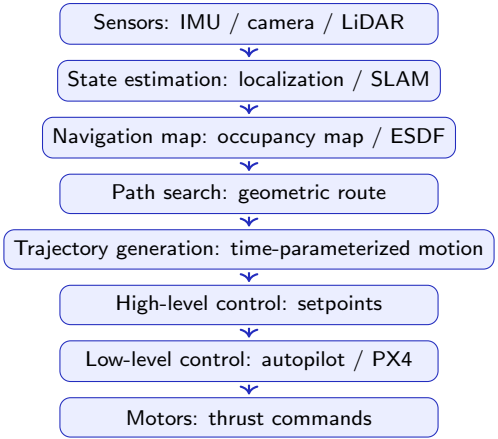
For multirotor UAVs, PX4 commonly uses a cascaded control structure.



The higher-level algorithm gives setpoints; PX4 handles stabilization, attitude/rate control, and motor mixing.

Classical modular autonomy architecture

A classical autonomous UAV system is usually divided into several modules. Each module solves one subproblem and passes its output to the next module.



Classical modular autonomy architecture

Main functional layers

- **Perception and mapping:** estimate the UAV state and build a navigation map.
- **Planning and trajectory generation:** find a safe route and convert it into a trackable trajectory.
- **Control and execution:** track the desired motion and generate motor commands.

Advantages

- Each module has a clear function and input-output interface.
- The system is easier to debug, test, and replace module by module.
- This architecture is widely used in UAV research and experiments.

The modular structure is still the most common engineering framework for autonomous UAV systems.

Limitations of modular autonomy

Modular autonomy also has limitations.

- Errors can propagate from one module to the next.
- Each module may be locally optimized, but the whole system may not be globally optimal.
- At high speed, perception-planning-control latency can be a bottleneck.

These limitations motivate learning-based and end-to-end approaches.

Basic idea of end-to-end autonomous flight

End-to-end methods try to reduce hand-designed intermediate modules.

Traditional modular pipeline:

$$\text{image} \rightarrow \text{depth / semantics / map} \rightarrow \text{planning} \rightarrow \text{control}.$$

End-to-end pipeline:

$$\text{image / point cloud / state} \rightarrow \text{neural network} \rightarrow \text{control command}.$$

For example, a network may take a front-view image or depth image and output a velocity command or a local trajectory.

Learning-based and end-to-end UAV autonomy

With the development of AI, learning-based methods have become an active research direction in UAV autonomy.

Instead of designing every module manually, these methods try to learn part of the perception, planning, or control process from data or interaction.

Common approaches include:

- **Imitation learning:** learns control or planning policies from expert demonstrations.
- **Reinforcement learning:** trains a policy through trial-and-error interaction with an environment and reward function.
- **Learning-based perception-to-action:** maps images, depth maps, or point clouds to velocity commands or control commands.
- **Hybrid learning-based autonomy:** combines neural networks with classical mapping, planning, optimization, or control modules.

Advantages and limitations of end-to-end methods

Advantages

- Can learn complex behaviors from data.
- May react quickly in local decision-making.
- Can jointly optimize perception and action.

Limitations

- Requires large and diverse training data.
- Generalization to unseen environments is difficult.
- Interpretability is usually limited.
- Safety constraints are hard to guarantee formally.
- Real-world deployment and certification remain challenging.

Therefore, current UAV systems often use learning-based methods together with classical modular autonomy, rather than relying on fully end-to-end control alone.

Summary

This lecture gives an overview of autonomous quadrotor UAV research from applications to autonomy.

- Basic dynamics explains thrust, hover, attitude-based motion, and underactuation.
- Autonomous UAVs rely on localization and perception, using sensors such as GPS RTK, motion capture, cameras, RGB-D cameras, LiDAR, and IMU.
- SLAM provides state estimation and mapping, which are essential for autonomous navigation.
- Planning, path smoothing, and trajectory generation convert map information into smooth and trackable motion.
- Low-level control and autopilot systems such as PX4 convert high-level commands into motor outputs.
- AI-based methods are promising, but safety and real-world reliability are still open problems.