EE4308 Advances in Intelligent Systems & Robotics

Part 2: Unmanned Aerial Vehicles

Ben M. Chen

Professor & Provost's Chair
Department of Electrical & Computer Engineering

Office: E4-06-08, Phone: 6516-2289
Email: bmchen@nus.edu.sg http://www.bmchen.net

Course contents & continuous assessments…

This module is to be assessed totally based on projects. For the 2nd part, there will be **introductory lectures** (about 6 hours in the first 2 weeks) on some basic topics related to unmanned aerial vehicles (UAVs), i.e.,

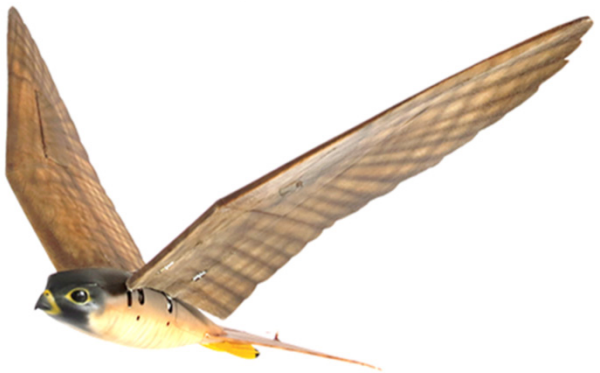
- Introduction to unmanned aerial vehicle systems and applications; UAV hardware and software structures; UAV platform and avionic system design; UAV dynamic modeling and flight control system design, path planning, trajectory generation, etc.

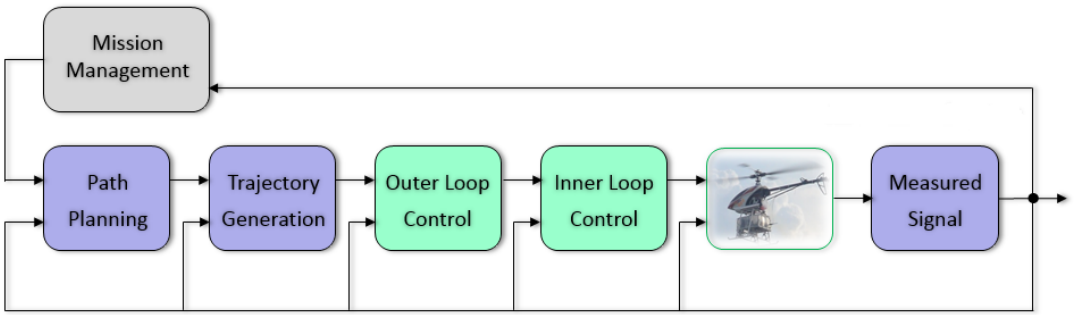
After the introductory part, there will be **tutorial sessions** conducted either in the class room or in the VICON room in T-Lab Building Level 4 (actual location will be announced before hand) for project simulation and actual flight experiments.

The **project** for this part is on how to plan paths for drones to fly in actual complicated environments with obstacles. Simulation in ROS is required together with real experiment to be conducted in the T-Lab VICON room. Project will be assessed based on simulation and experimental result, report and presentation.

Outline of the notes…

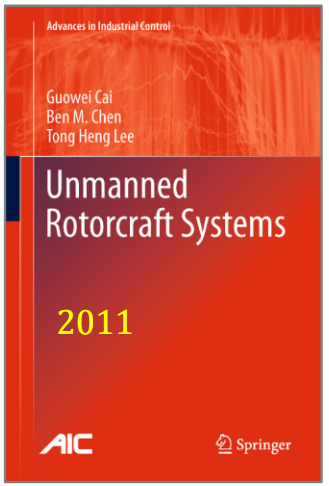
- Introduction to drones and applications
  - Types of drones
  - Applications
  - NUS UAV research highlights
- Overall structure of unmanned systems
  - Communication unit
  - Ground control systems
- Internal structure of unmanned systems
  - Avionic systems
  - Dynamic modeling
  - Flight control systems
  - Measurement systems
  - Path planning
  - Trajectory generation
  - Mission management

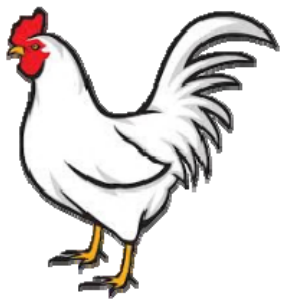




What is a drone or UAV?

An unmanned aerial vehicle (UAV or drone) is an aircraft that is equipped with necessary data processing units, sensors, automatic control, and communications systems and is capable of performing autonomous flight missions without the interference of a human pilot.





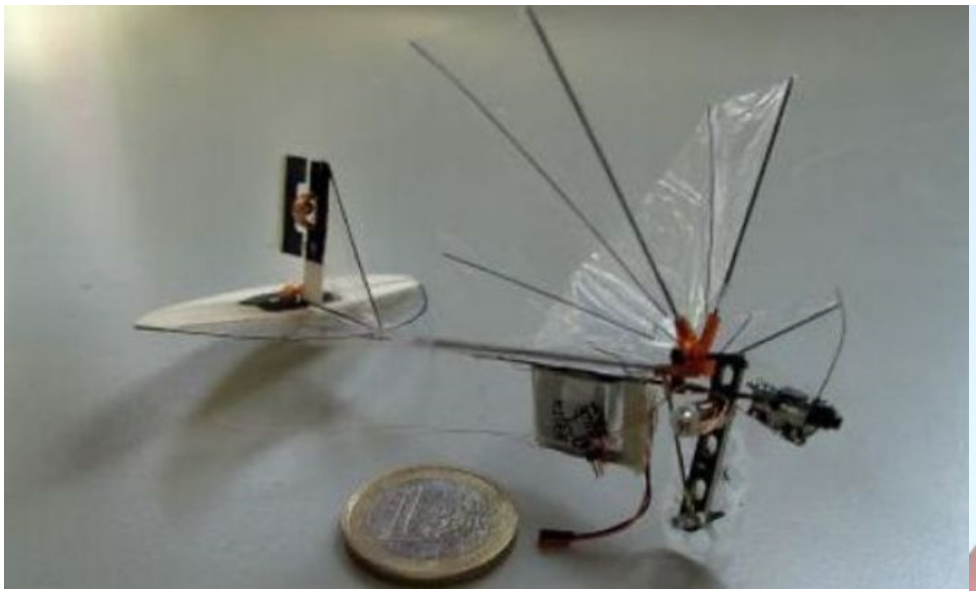
A drone?

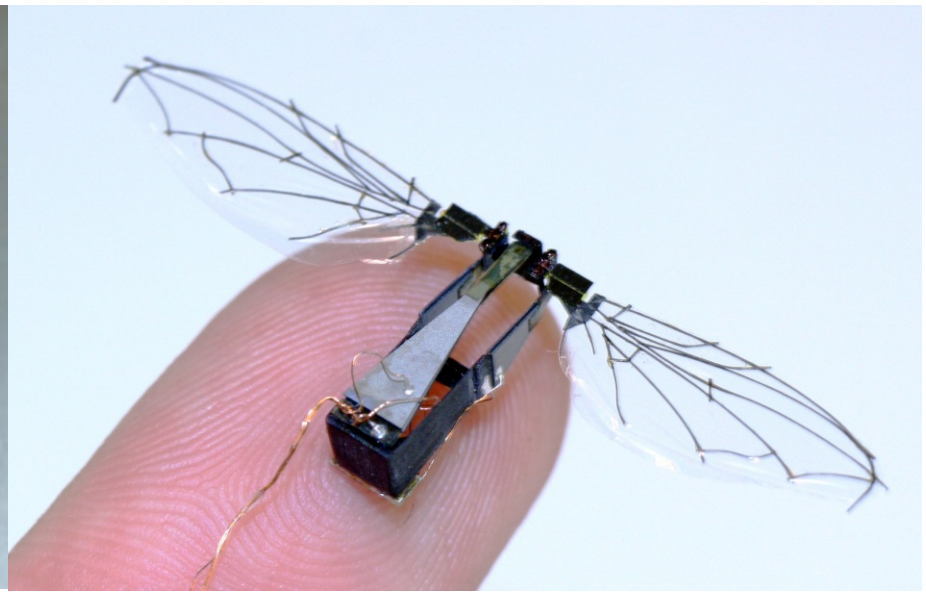
Drone types…

Fixed-wing UAVs…

Drone types…

Flapping-wing UAVs…



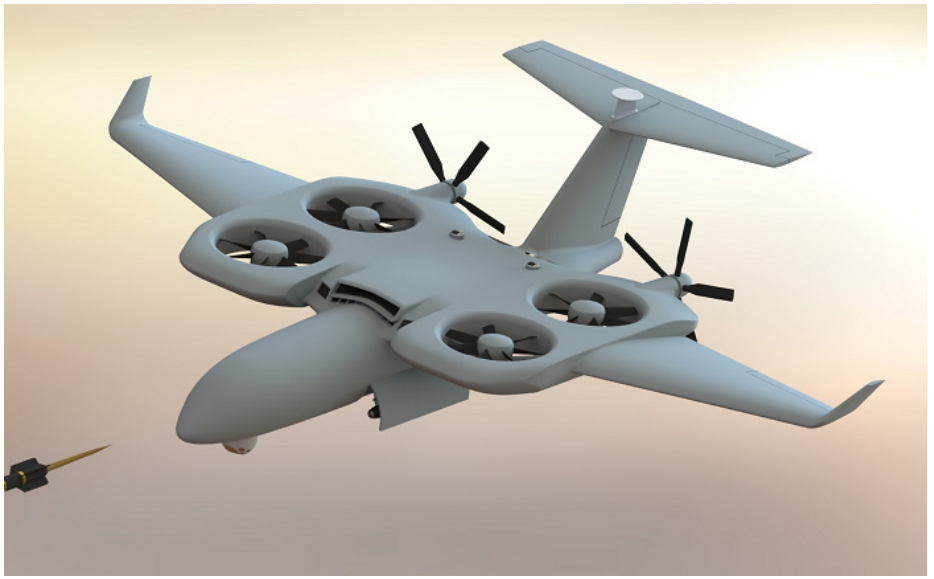


Drone types…

Rotorcraft UAVs…

Drone types…

Hybrid UAVs…



Drone applications…

GPS-based environments…

Search & rescue

Surveillance & patrolling

Forestation & agriculture

Aerial mapping

Drone applications…

GPS-based environments…

Drone applications…

GPS-denied environments…

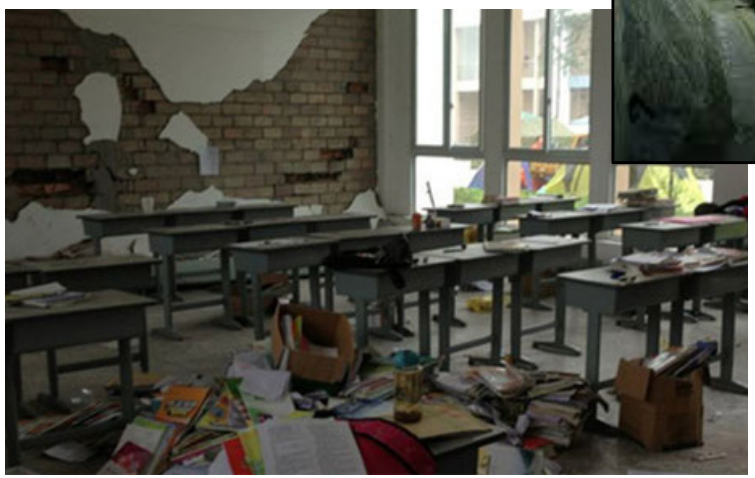
Urban canyon

&

inside forest

Indoor

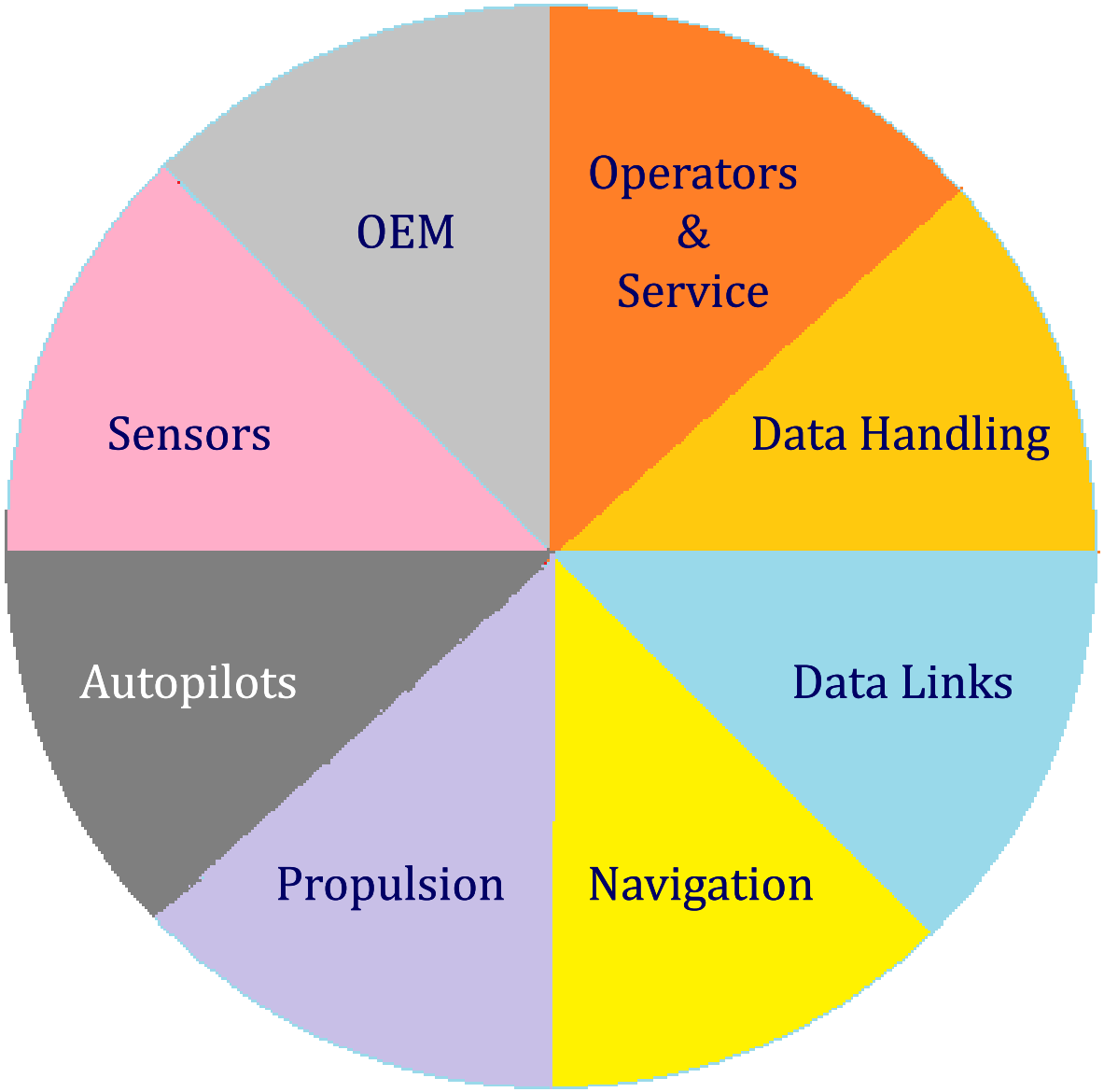
Tunnels



Under bridges

Drone industry…

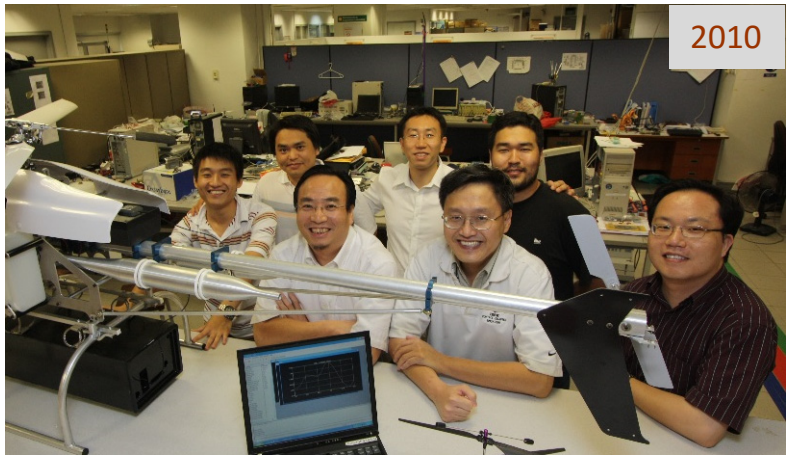
Market sectors…

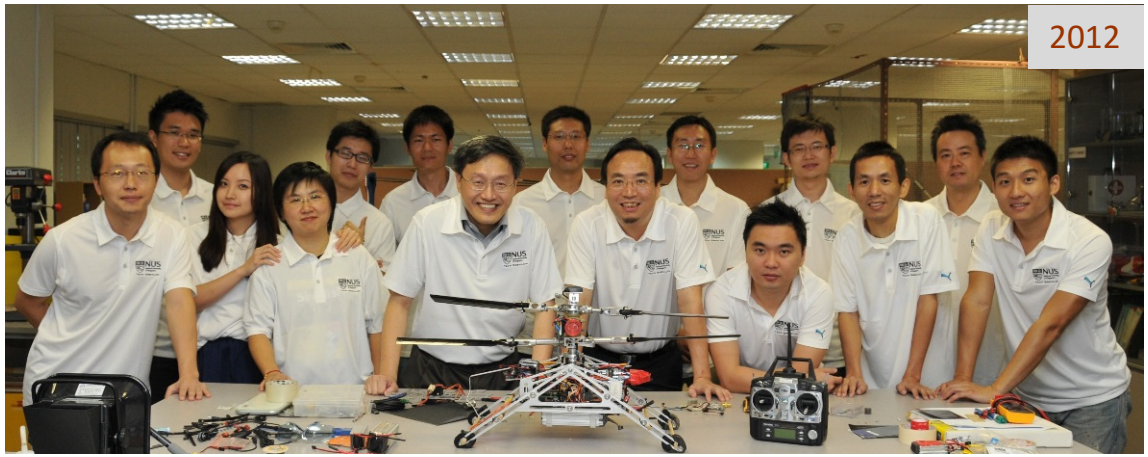


- Military
- Research/education
- Mining
- Forestry
- Power & pipeline
- Media
- Real estate
- Agriculture
- Policing
- Film industry
- Oil & gas industry
- Ports
- Entertainment
- ……

NUS research team & unmanned systems platforms…

NUS National University of Singapore



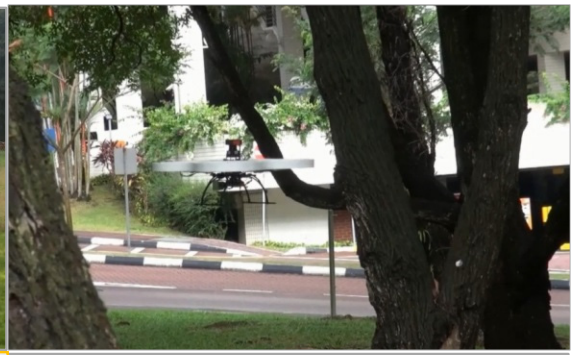


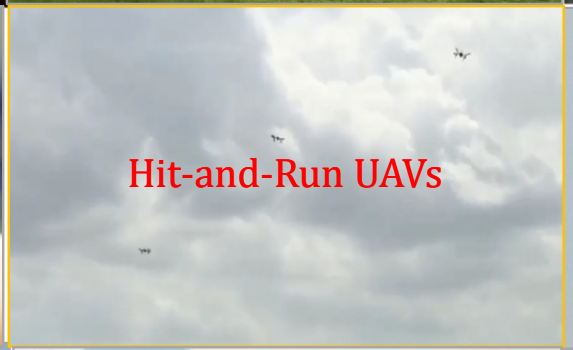
NUS unmanned systems research highlights…

Indoor Navigation & Control

Firefighting

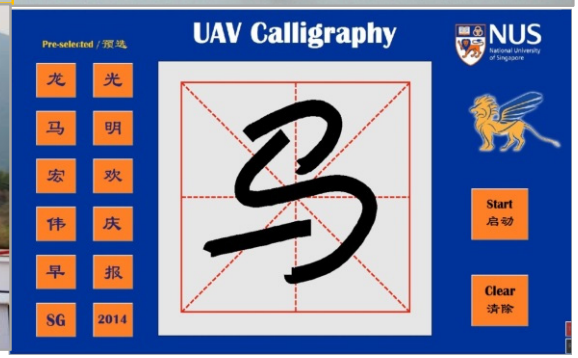
Inside Forest Navigation





Hybrid UAVs

Cargo Transportation



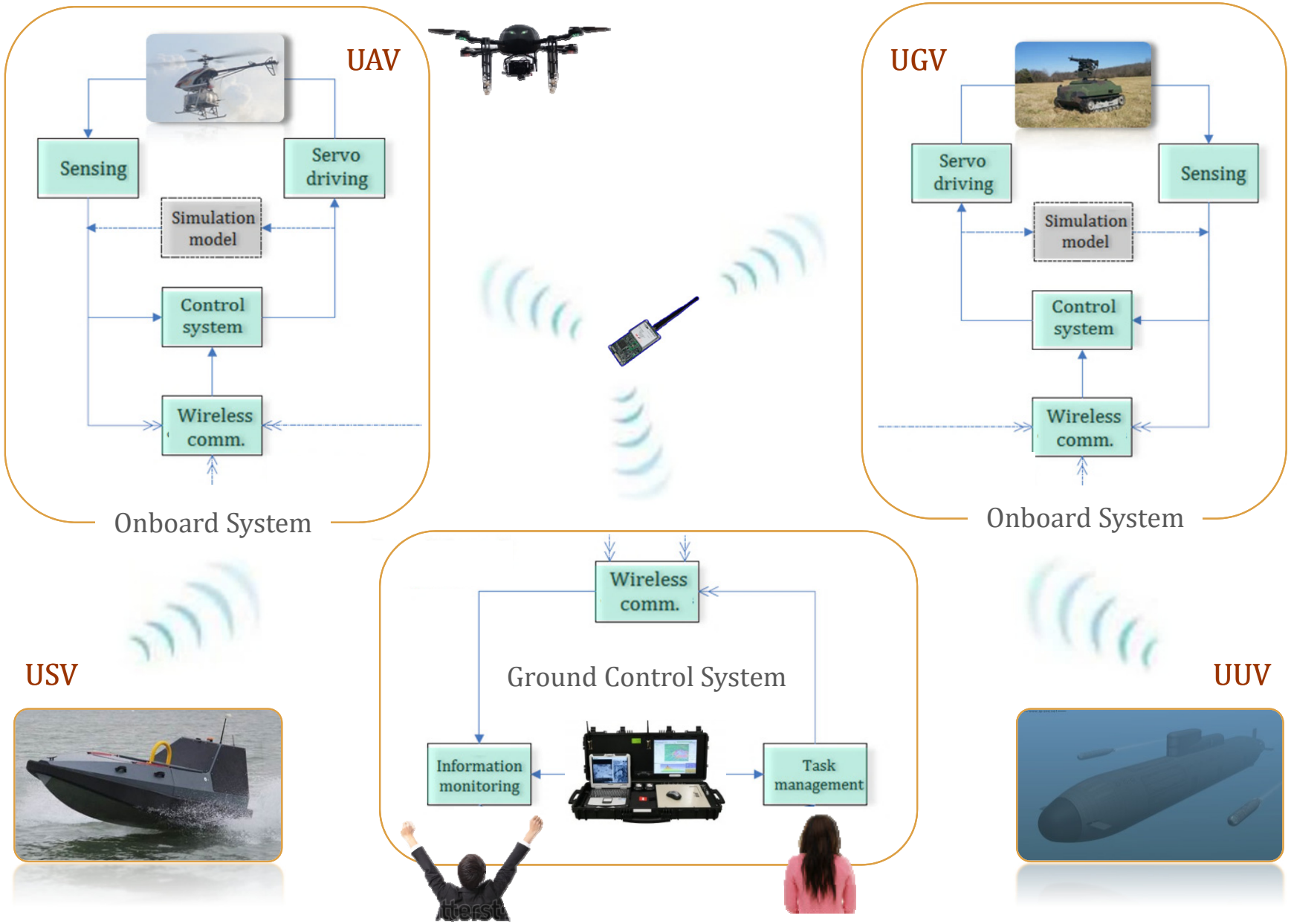
UAV Calligraphy

Awards won by NUS research team…

2014 CCC Guan Zhaozhi Prize

IMAV 2014 Champion

Overall structure of unmanned systems…



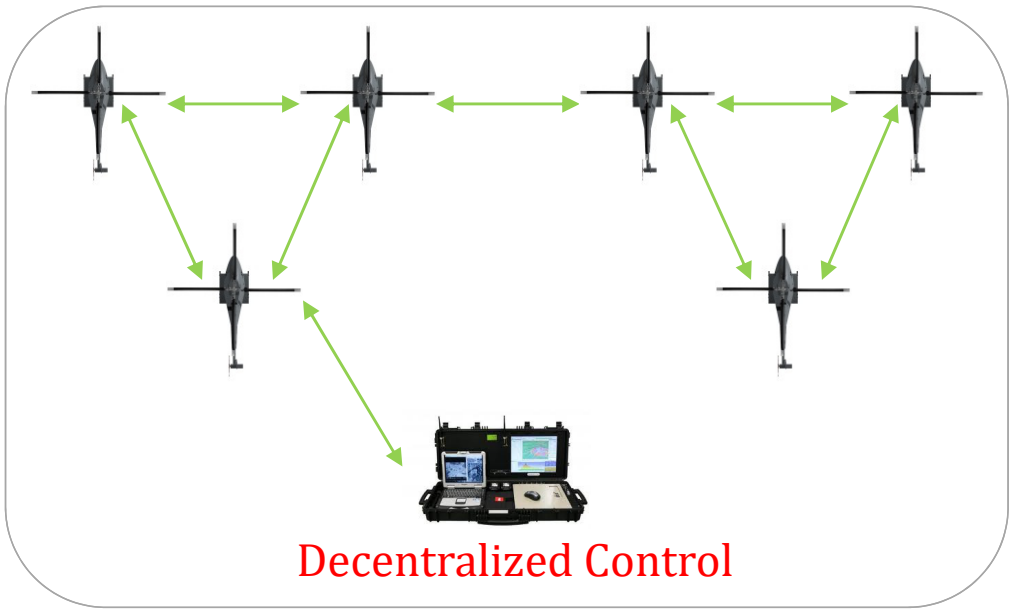
Ground control systems…

Online touch-screen path planning on the map

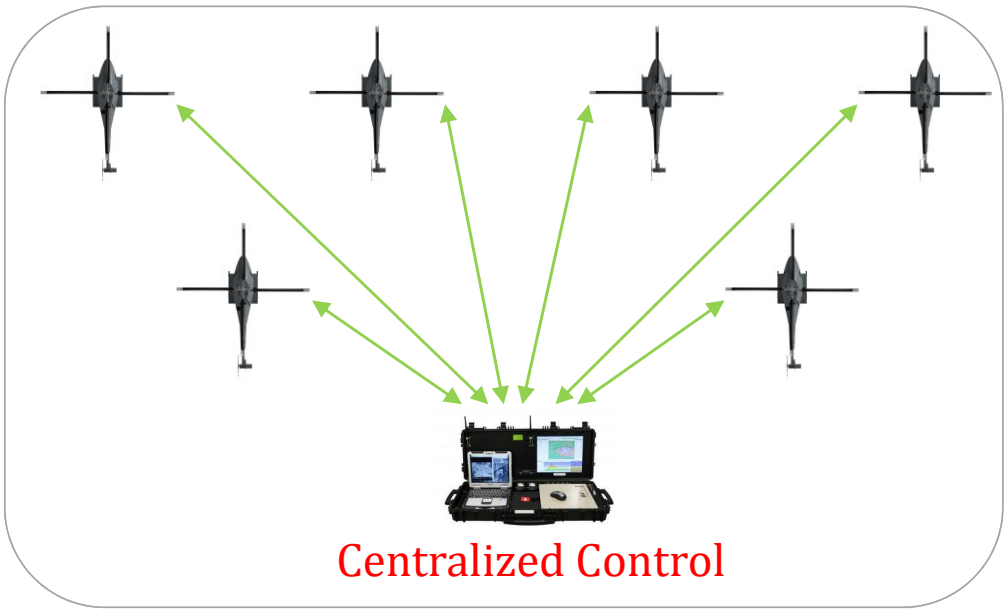
Command window to issue real-time commands

Communication unit…

The communication units in the UAV system framework are deployed as interfaces between the UAV entity itself and external entities. The external entity can be the GCS for the ground operator, or another UAV entity for information exchange. With UAV to GCS communications, the operator can remotely control and monitor UAVs in operation. With inter-UAV communications, the UAV team can multiply their capability and effectiveness in cooperative tasks.

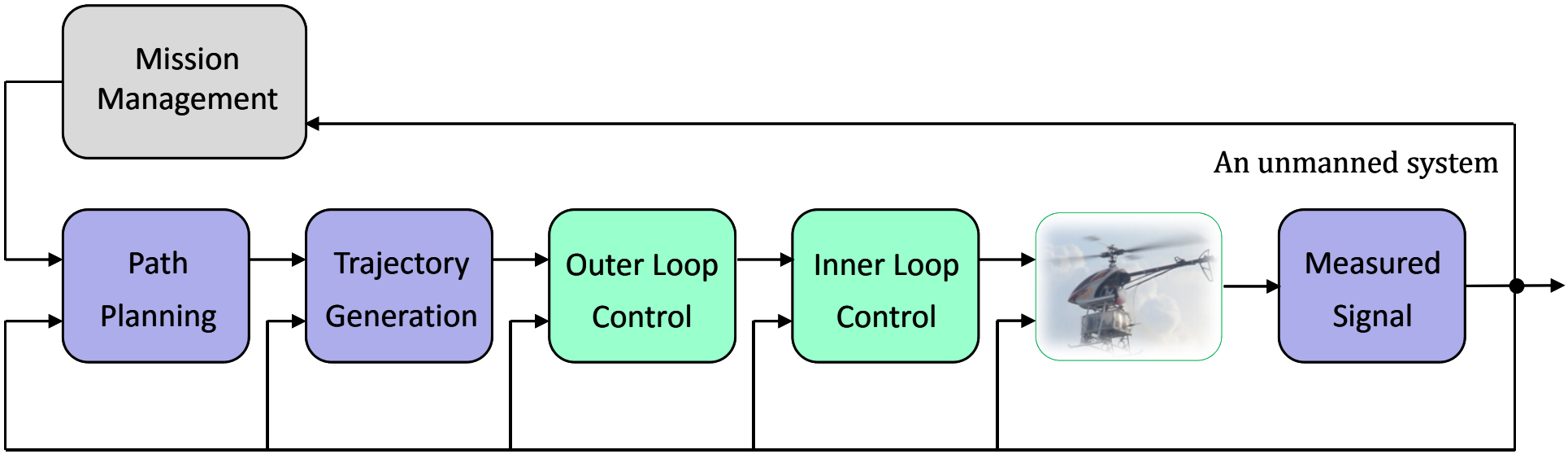


Decentralized Control



Centralized Control

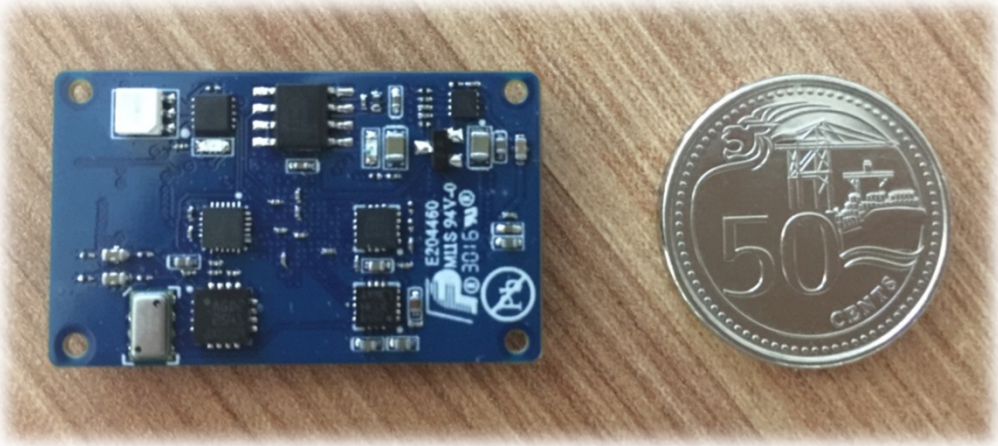
Internal structure of unmanned systems…

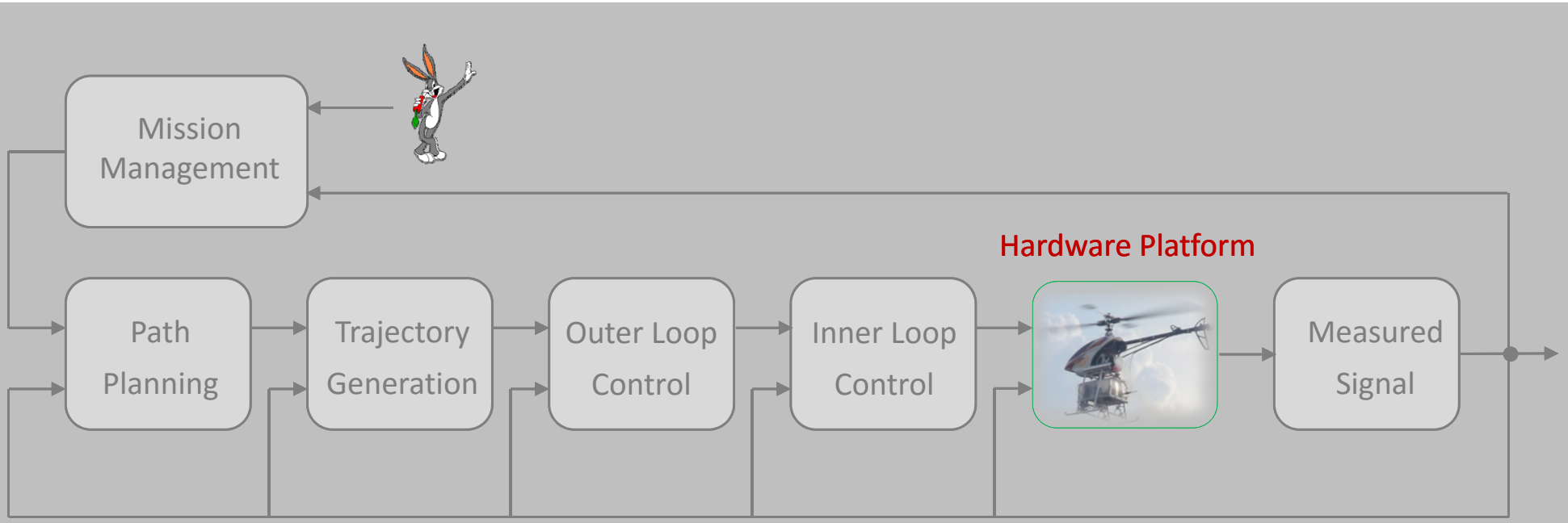


Hardware components & avionic systems…

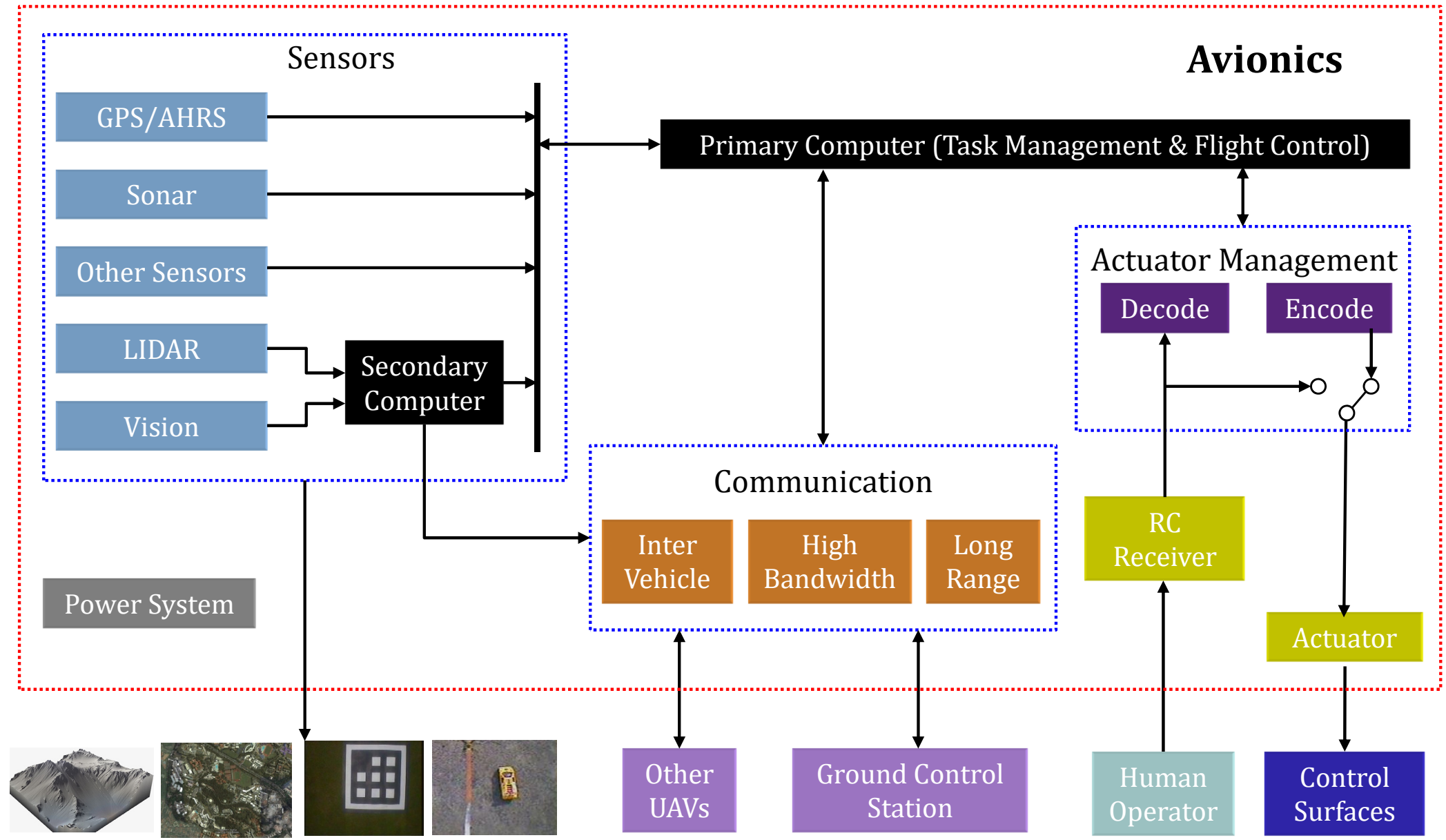
An NUS made micro quadrotor

An NUS made avionic system

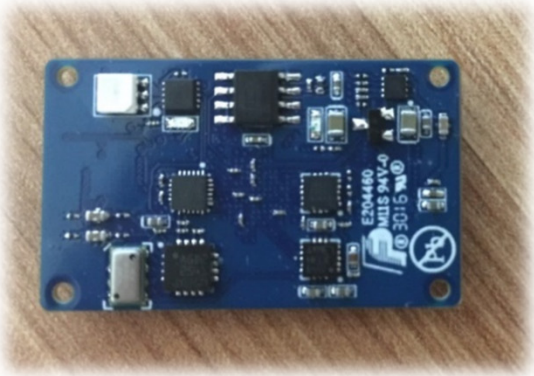




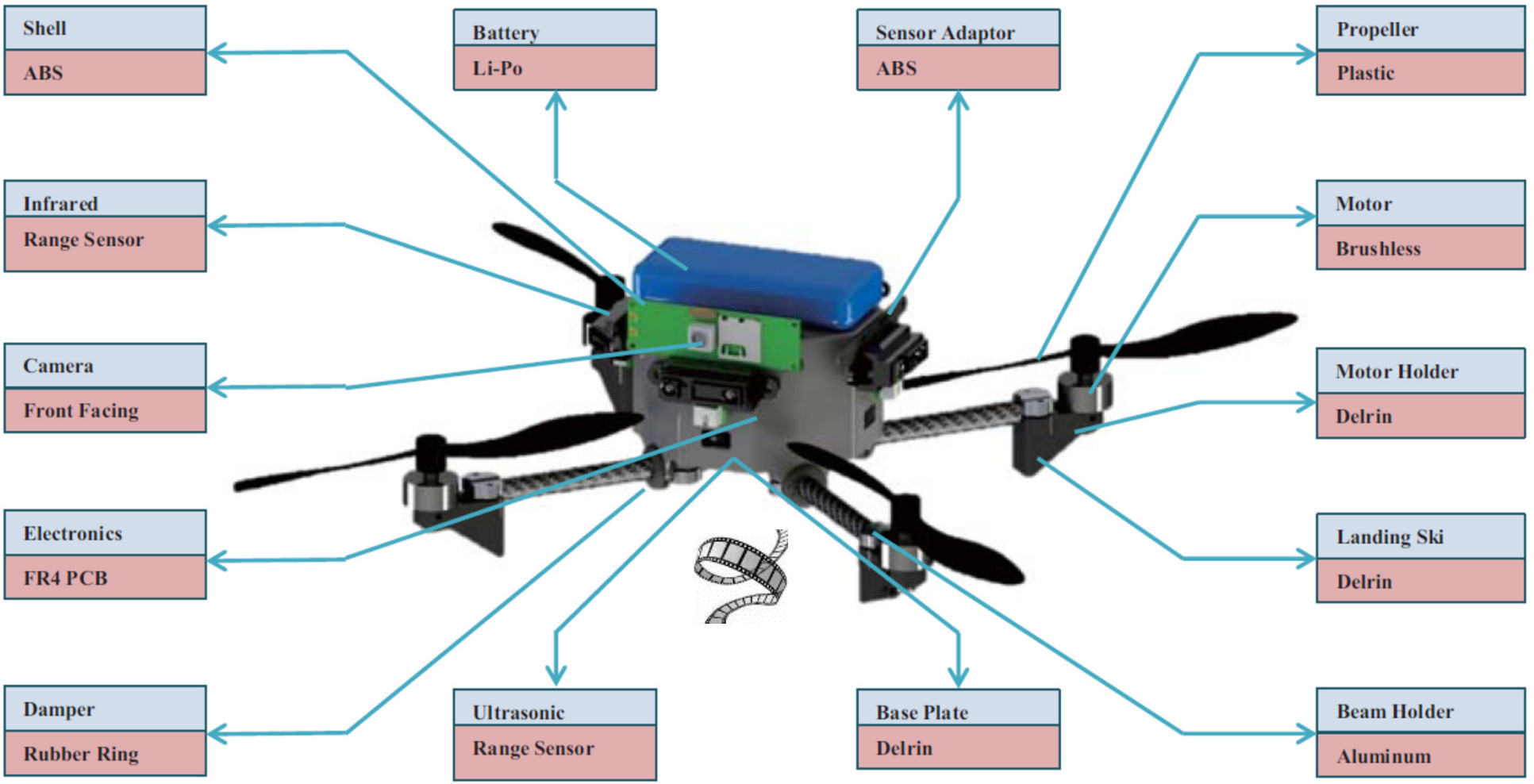
Essential hardware components of unmanned systems…



Hierarchy of an avionic system…



Component breakdown of a typical multi-rotor UAV...



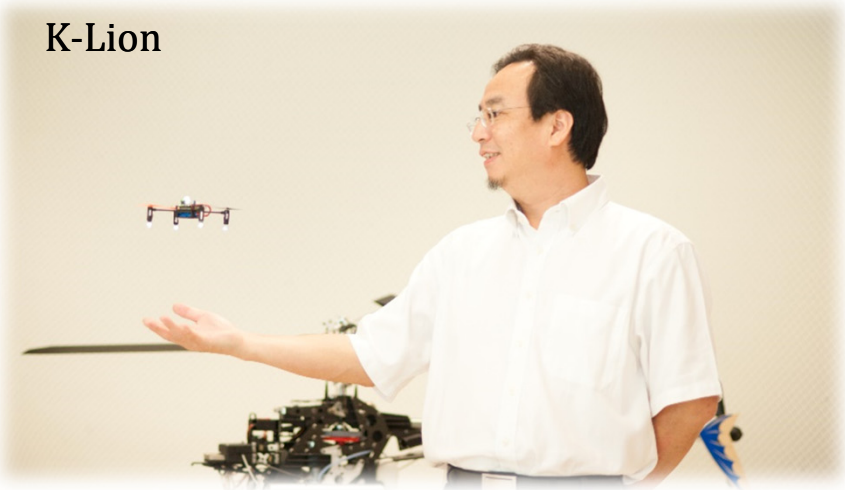
Multi-rotor UAV platforms…

BlackLion 168

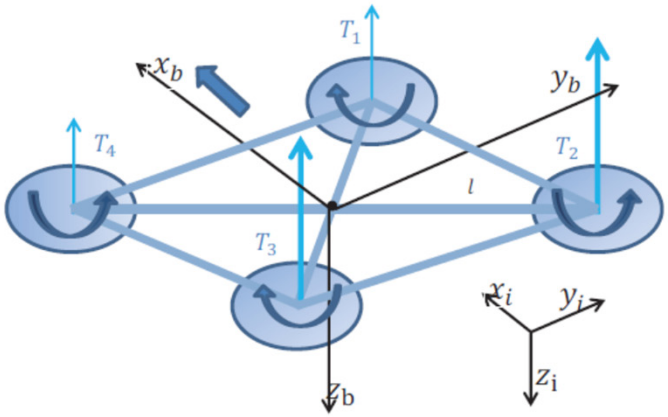
All Made in Singapore

BlackLion 068

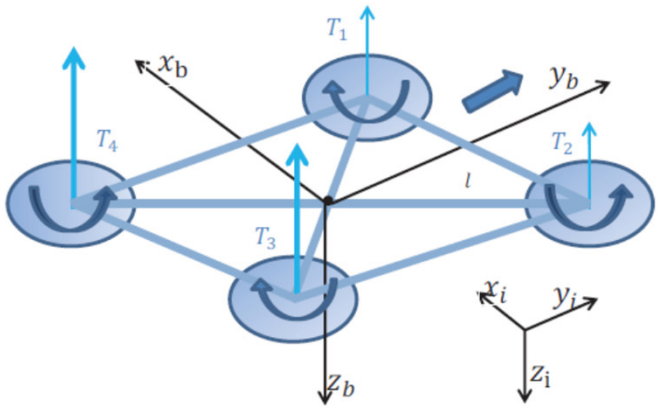
K-Lion



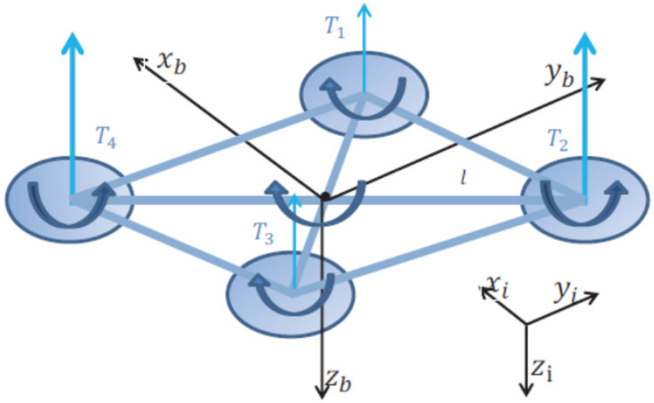
Dynamics modeling of a quadrotor UAV…



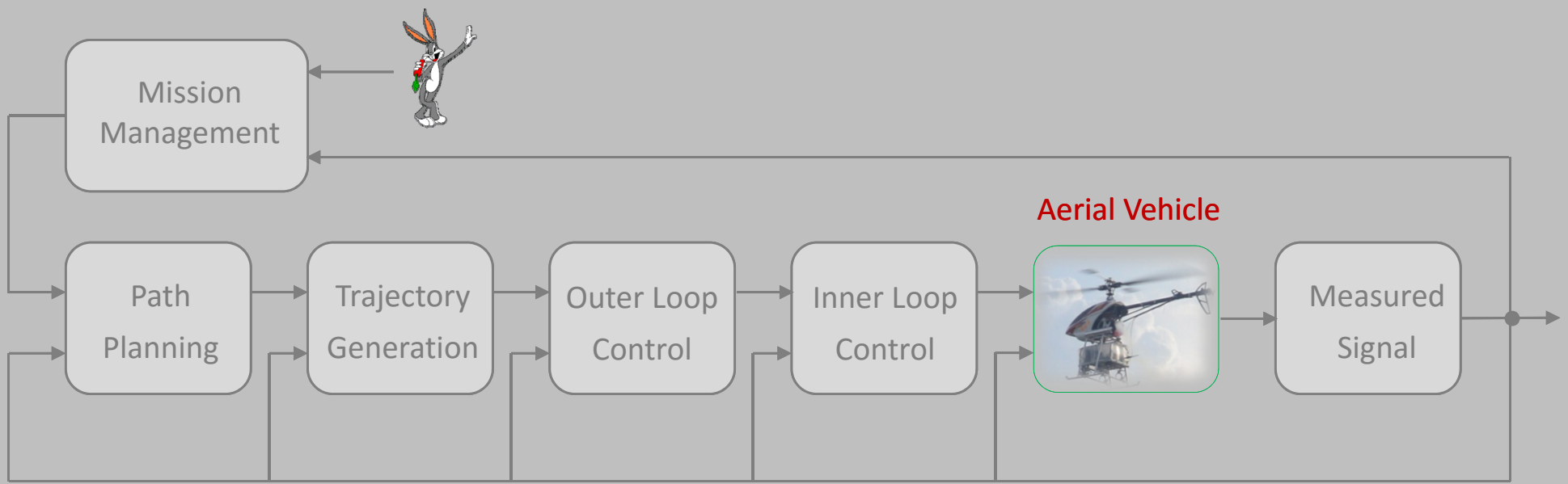
(a) Pitching



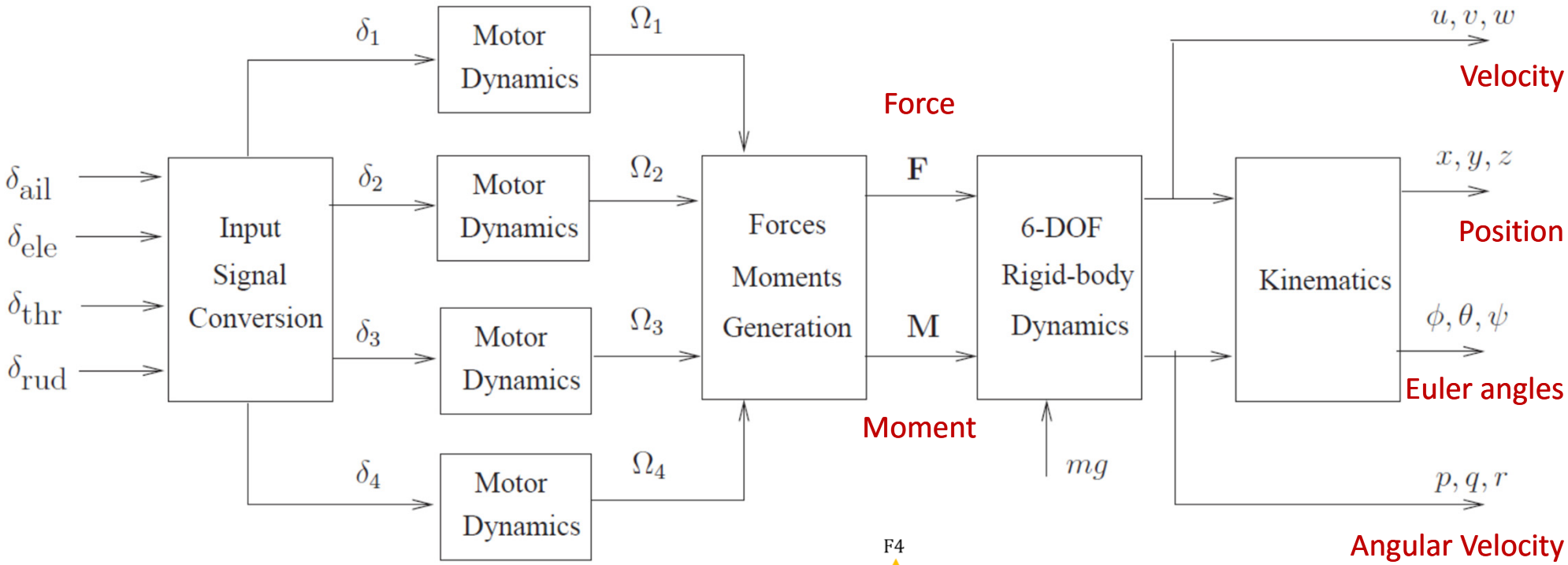
(b) Rolling



(c) Yawing



Dynamics model structure of the aerial vehicle…

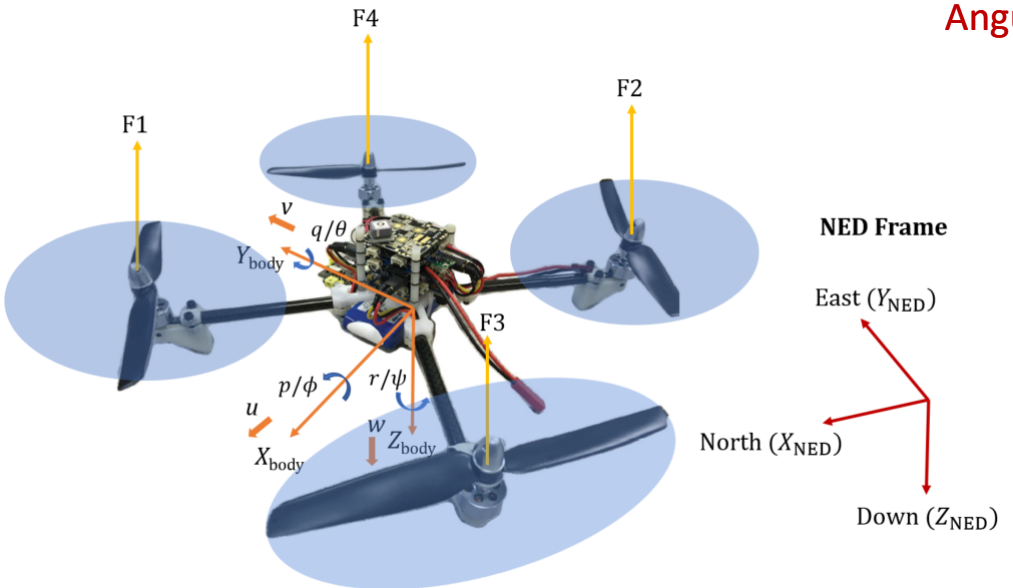


δ_{ail} aileron input

δ_{ele} elevator input

δ_{thr} collective pitch input

δ_{rud} rudder input



Linearized model of the aerial vehicle…

With some appropriate simplifications, we can obtain a simplified linear model for a quadrotor UAV as follows

$$
\begin{aligned}
\dot{u} &= -g\theta \\
\dot{v} &= g\phi \\
\dot{w} &= \tfrac{1}{m}u_1 \\
\dot{p} &= J_{\mathrm{XX}}^{-1}u_2 \\
\dot{q} &= J_{\mathrm{YY}}^{-1}u_3 \\
\dot{r} &= J_{\mathrm{ZZ}}^{-1}u_4 \\
\dot{\phi} &= p \\
\dot{\theta} &= q \\
\dot{\psi} &= r
\end{aligned}
$$

$$
\begin{aligned}
u_1 &= F_\Sigma \\
u_2 &= M_1 \\
u_3 &= M_2 \\
u_4 &= M_3
\end{aligned}
\quad \Leftarrow \quad
\begin{pmatrix} F_\Sigma \\ M_1 \\ M_2 \\ M_3 \end{pmatrix}
= \underbrace{\begin{pmatrix} -k_T & -k_T & -k_T & -k_T \\ dk_T & -dk_T & -dk_T & dk_T \\ dk_T & dk_T & -dk_T & -dk_T \\ k_Q & -k_Q & k_Q & -k_Q \end{pmatrix}}_{\Gamma}
\begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix}
$$

ω_1, ω_2, ω_3, ω_4 are respectively the motor angular speeds and Γ is a constant matrix related to motor properties.

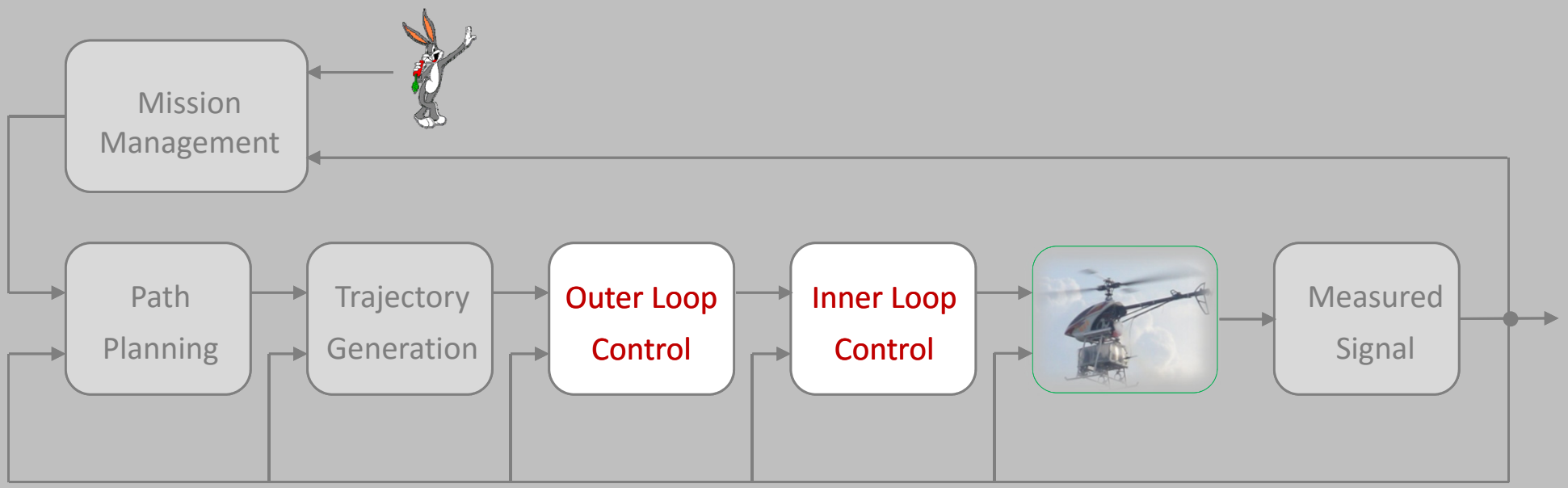
g is the gravity constant, m is the UAV mass.

J_{XX} [kgm^2]	Inertia moment of body about x-axis
J_{YY} [kgm^2]	Inertia moment of body about y-axis
J_{ZZ} [kgm^2]	Inertia moment of body about z-axis

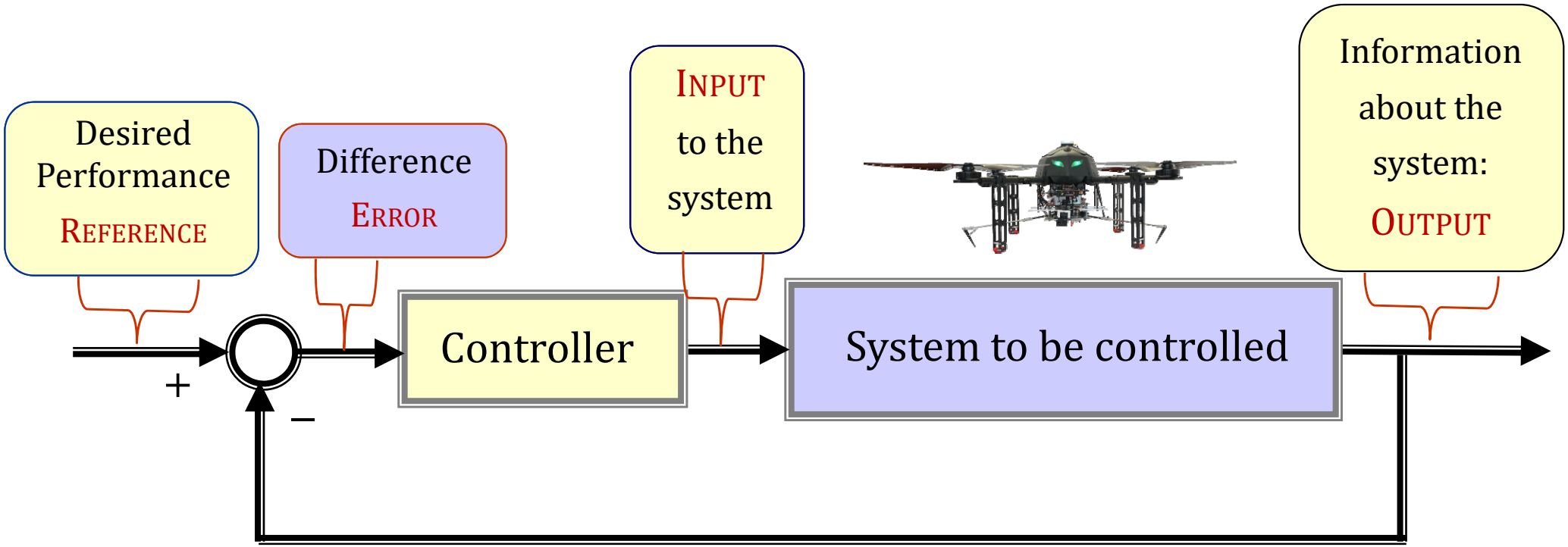
Flight control systems…

Due to the nature of the time scale of the aircraft dynamics, i.e., the attitude response is much faster compared to that of the position and velocity, it is a common practice to separate the flight control system into two parts, i.e., the inner-loop and the outer-loop control. More specifically,

- Inner loop control system is to guarantee the stability of the aircraft attitude; and
- Outer loop control system is to control the aircraft position and velocity.



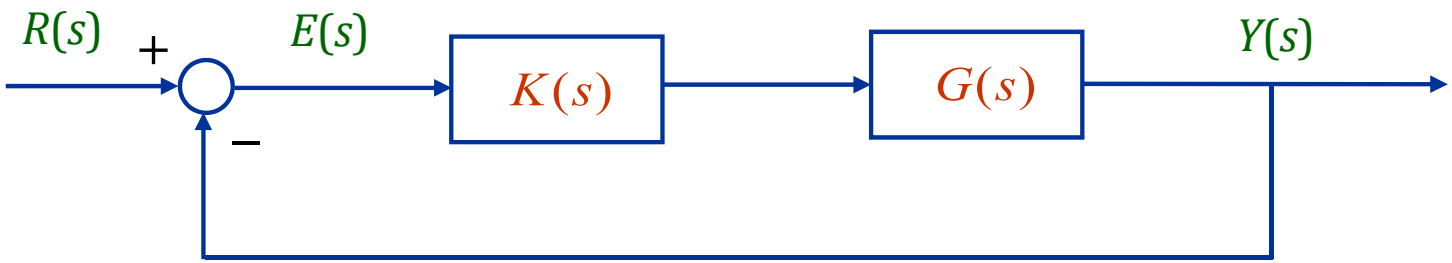
The objectives of a control system…



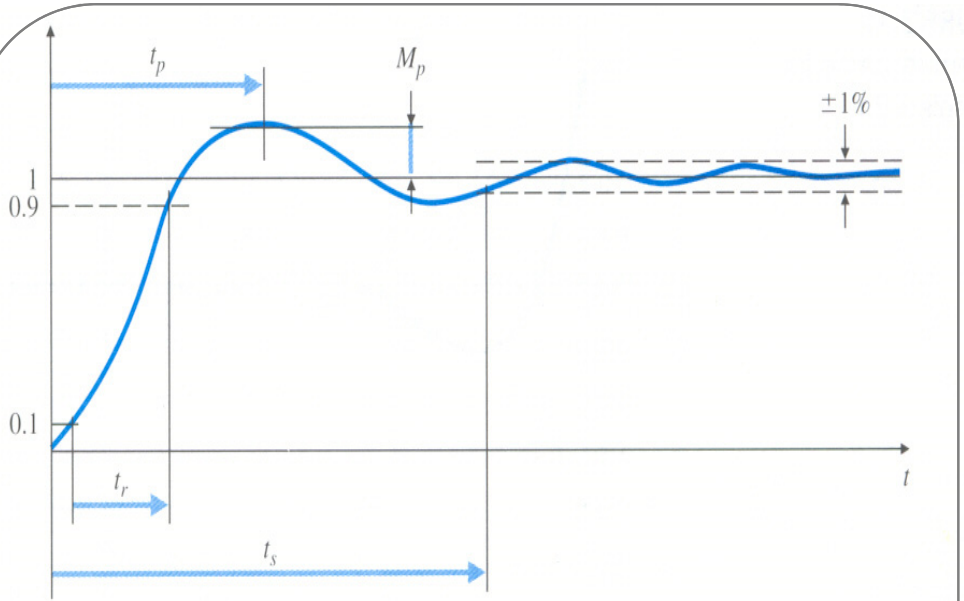
Objectives: To stabilize the overall closed-loop system (Stability); and to make the system **OUTPUT** and the desired **REFERENCE** as close as possible, i.e., to make the **ERROR** as small as possible.

Classical control technique…

Classical control scheme can be depicted in the following diagram:

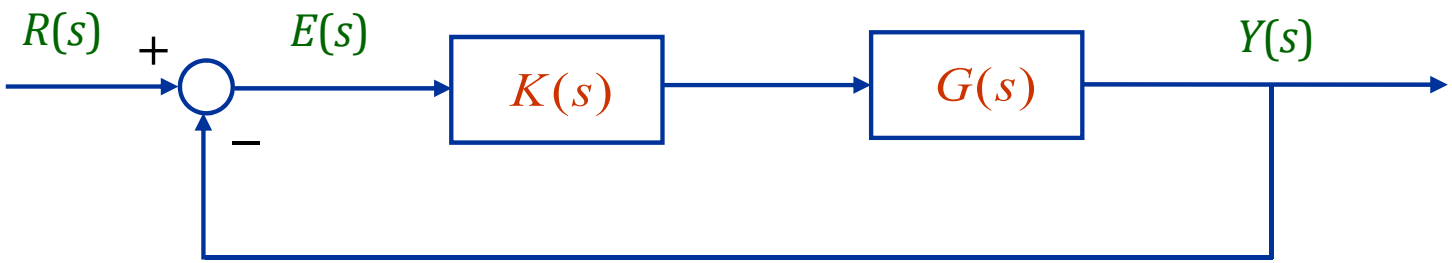


where $G(s)$ is the transfer function of the system to be controlled, $K(s)$ is the controller to be designed, $R(s)$, $Y(s)$ and $E(s)$ are the Laplace transforms of the reference signal r, the measurement y and the tracking error e, respectively. Once again, the objectives are to make the overall closed-loop system stable and y to match r as close as possible.



A typical response of a 2nd order system…

Classical control technique (cont.)



Note that

$$G(s) = \frac{Y(s)}{U(s)} \quad \Rightarrow \quad Y(s) = G(s)U(s)$$

$$U(s) = K(s)E(s) \qquad E(s) = R(s) - Y(s)$$

We have

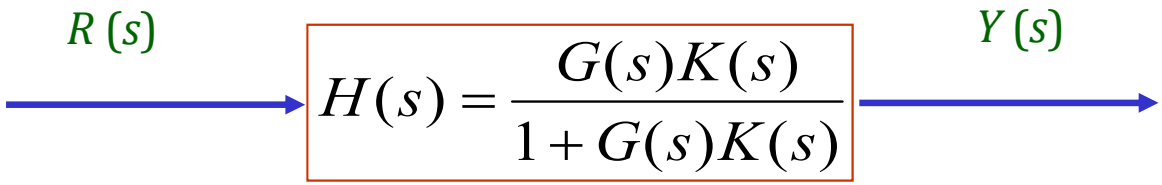
$$Y(s) = G(s)U(s) = G(s)K(s)E(s) = G(s)K(s)\left[R(s) - Y(s)\right]$$

$$Y(s) = G(s)K(s)R(s) - G(s)K(s)Y(s) \Rightarrow \left[1 + G(s)K(s)\right]Y(s) = G(s)K(s)R(s)$$

$$\Rightarrow \quad H(s) = \frac{Y(s)}{R(s)} = \frac{G(s)K(s)}{1 + G(s)K(s)} \Rightarrow$$

Closed-loop transfer function from R to Y.

Thus, the block diagram of the control system can be simplified as



The whole control problem becomes how to choose an appropriate $K(s)$ such that the resulting $H(s)$ would yield desired properties between R and Y.

Let us focus our attention on the control system design of a double integrator with a proportional-deferential (PD) control law, i.e.,

$$G(s) = \frac{1}{s^2} \qquad \text{and} \qquad K(s) = k_p + k_d s$$

This implies

$$H(s) = \frac{G(s)K(s)}{1+G(s)K(s)} = \frac{k_d s + k_p}{s^2 + k_d s + k_p}$$

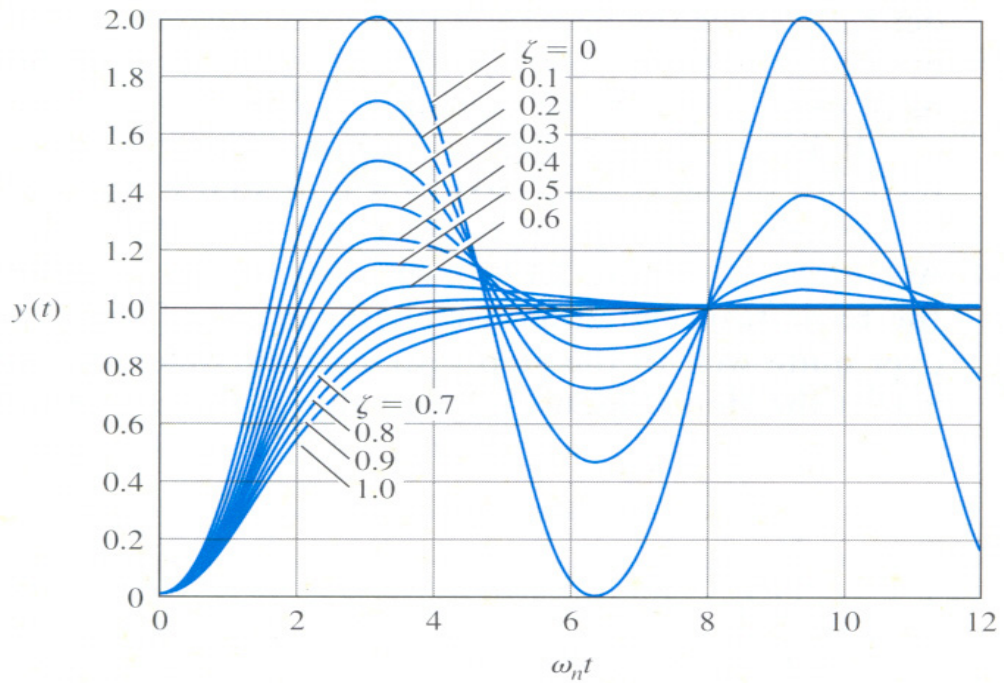
⇒ a 2nd order system

Behavior of the second order systems with a unit step input

Again, consider the following block diagram with a standard 2nd order system

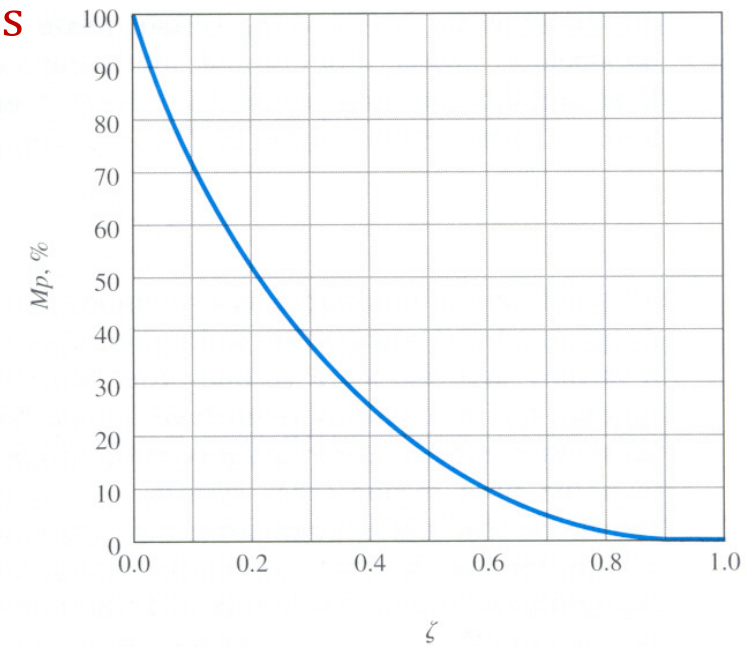
$R(s) = 1/s$ → $r = 1$ → $H(s) = \dfrac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ → $Y(s)$

The behavior of the system is given on the right. It is fully characterized by ζ, which is called the *damping ratio*, and ω_n, which is called the *natural frequency*.

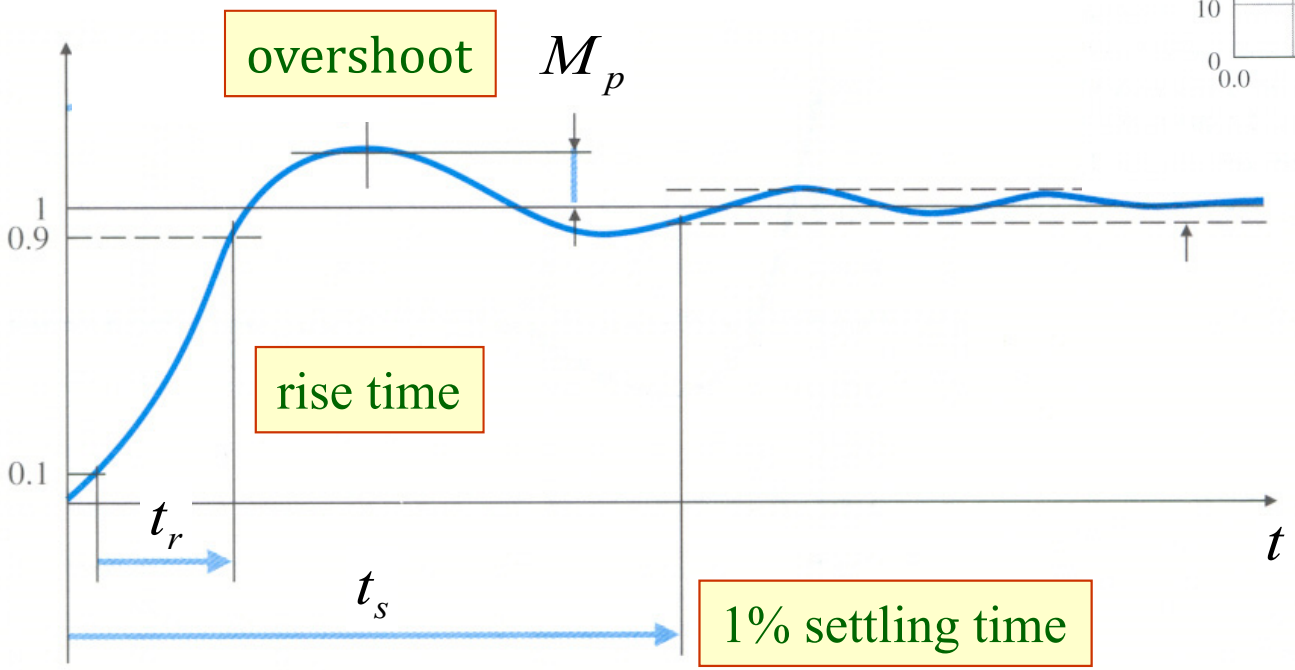


Control system design with time-domain specifications

$R(s)$ → $r = 1$ → $H(s) = \dfrac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2}$ → $Y(s)$



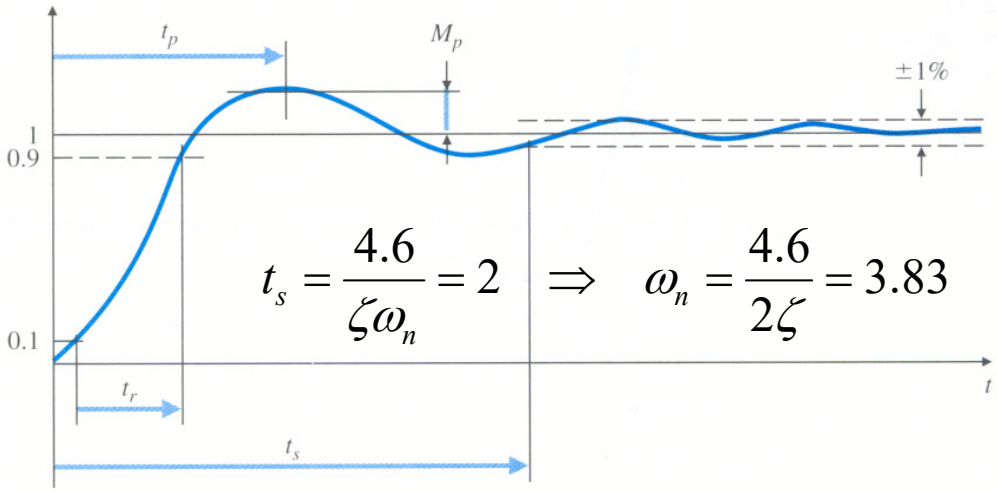
Overshoot vs damping ratio

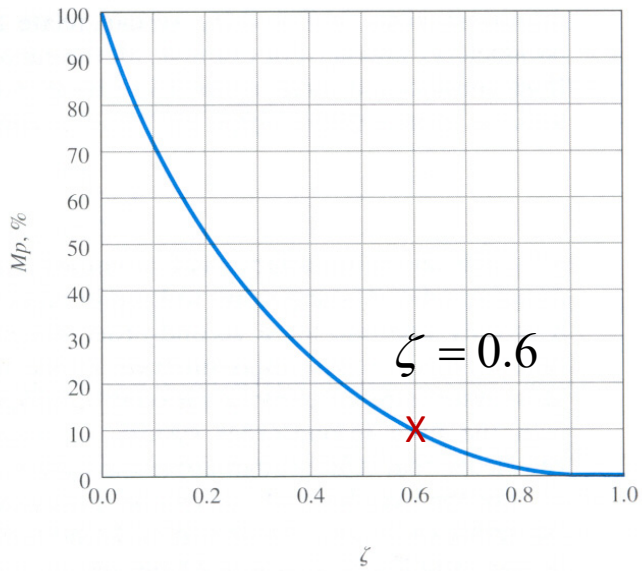


$$t_r \cong \frac{1.8}{\omega_n}$$

$$t_s \cong \frac{4.6}{\zeta\omega_n}$$

Example: Given a double integrator plant, design a PD control law such that the resulting closed-loop system is stable. When it tracks a step reference, it has a settling time less than 2 seconds and overshoot less than 10%.





Recall that

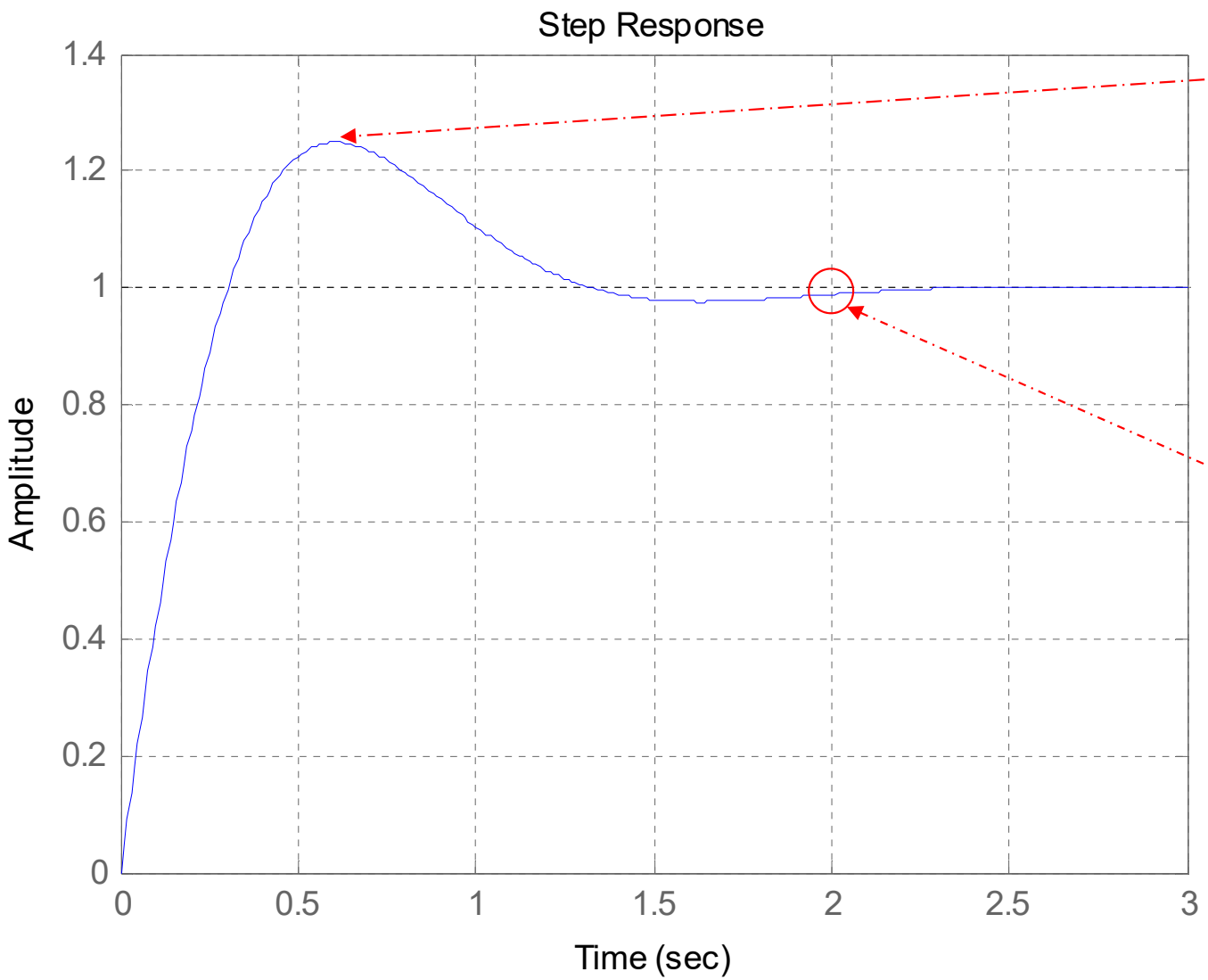
$$H(s) = \frac{G(s)K(s)}{1+G(s)K(s)} = \frac{k_d s + k_p}{s^2 + k_d s + k_p} \Leftrightarrow H(s) = \frac{\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} = \frac{14.7}{s^2 + 4.6s + 14.7}$$

$$\Rightarrow \quad k_d = 4.6, \quad k_p = 14.7$$

This additional term might cause some problems…

Classical control technique (cont.)

Simulation result…



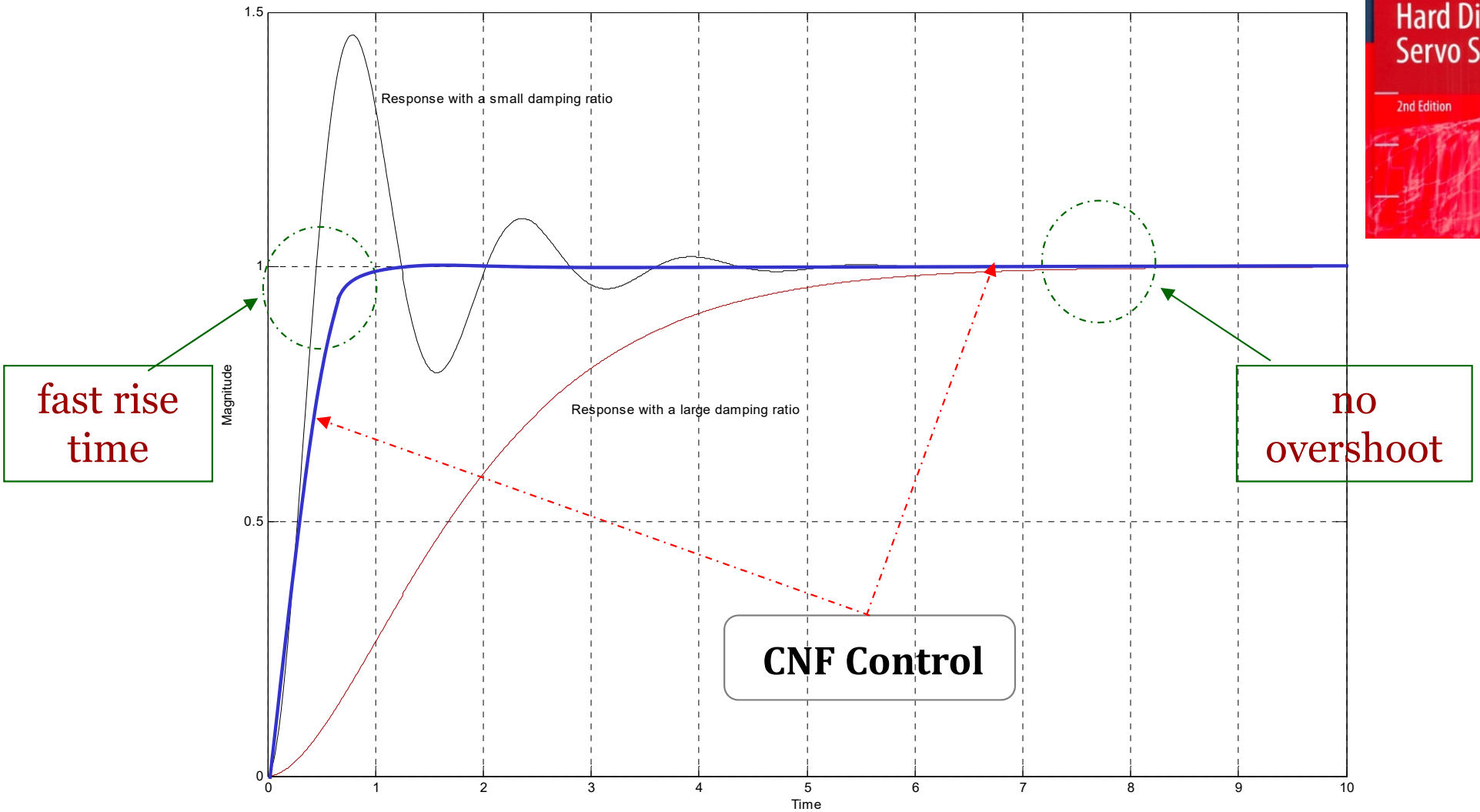
Overshoot is more than 20%

Settling time is right on the target

Need to fine tune it!

NUS National University of Singapore

Motivation…



CNF control (cont.)...

CNF control consists of a linear law and a nonlinear feedback law with any switching element. The linear part is designed to yield a closed-loop system with a small damping ratio for a quick response. On the other hand, the nonlinear part is used to increase the damping ratio of the closed-loop system as the system output approaches the target reference to reduce the overshoot caused by the linear part.

For a double-integrator plant characterized by

$$\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \mathrm{sat}(u), \quad y = x, \quad h = \begin{bmatrix} 1 & 0 \end{bmatrix} x$$

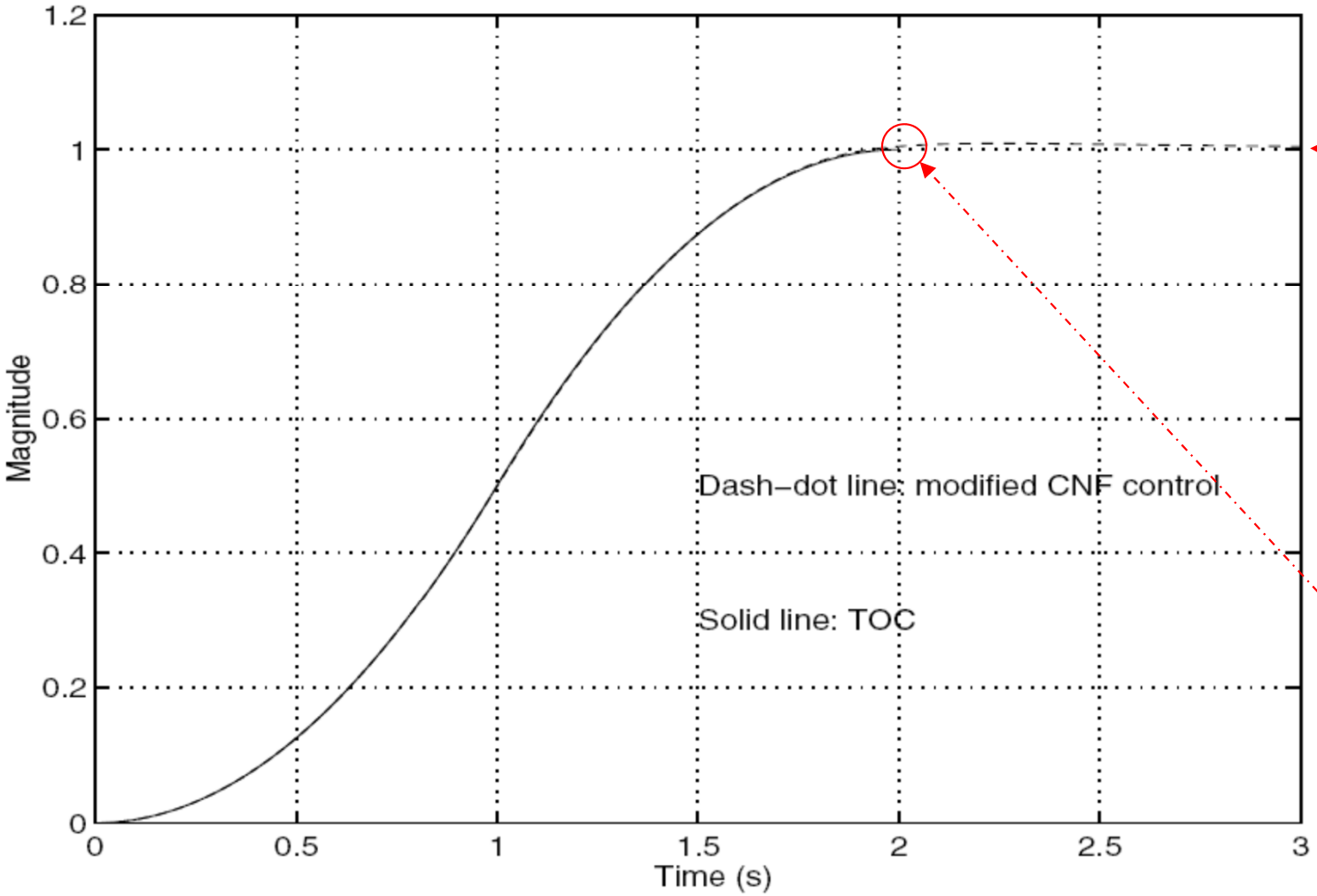
where as usual x is the state, u is the input, and y and h are, respectively, the measurement and controlled outputs.

The following CNF control law is capable of beating even a time-optimal control (TOC) law in settling time with 1% overshoot

$$u = \begin{bmatrix} -6.5 & -1 \end{bmatrix} x + 6.5\, r - \left(\mathrm{e}^{-|1-h|} - 0.36788 \right) \begin{bmatrix} 1.4481 & 10.8609 \end{bmatrix} \left(x - \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right)$$

CNF control (cont.)…

Simulation result…

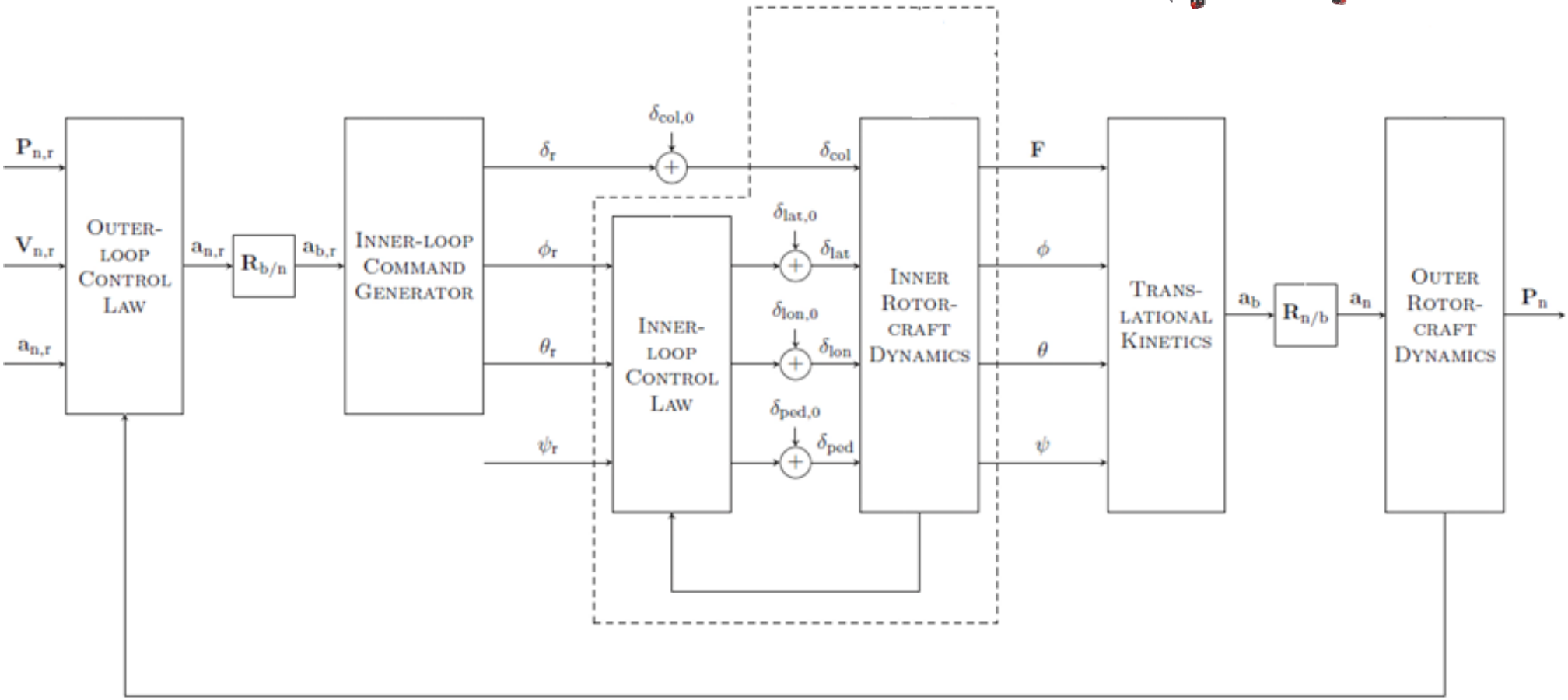


Overshoot is less than 1%

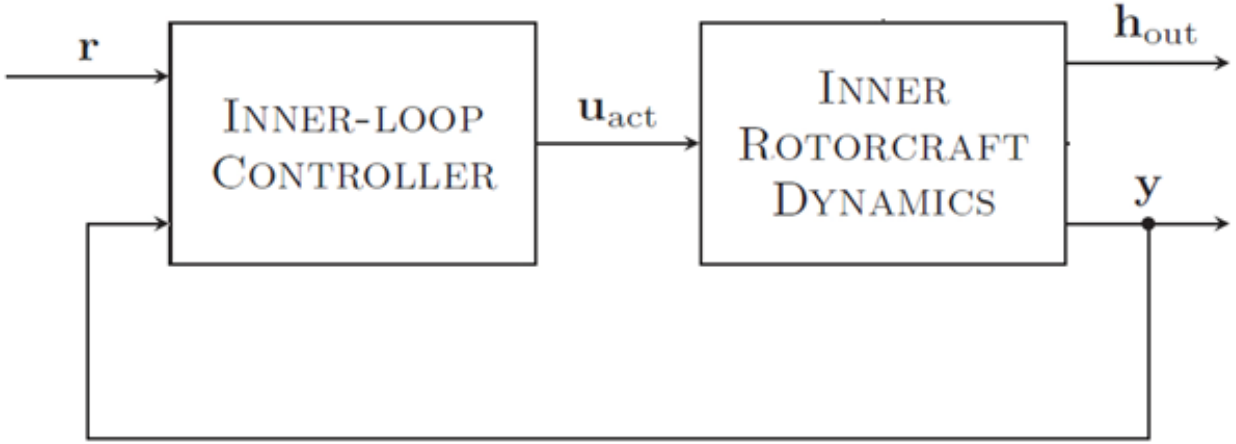
The best possible Settling time

Output responses of the CNF control and the TOC…

Detailed structure of flight control systems…



Inner-loop control system…



$$\mathbf{h}_{\text{out}} := [\phi \quad \theta \quad \psi]^{\mathrm{T}}$$

$$\mathbf{y} = [\phi \quad \theta \quad p \quad q \quad r \quad \psi]^{\mathrm{T}}$$

Controlling a quadrotor type drone is rather simple as mechanically each channel can be regarded as decoupled from one another.

$$\begin{bmatrix} \dot{\phi} \\ \dot{p} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \phi \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ J_{\text{XX}}^{-1} \end{bmatrix} u_2$$

$$\begin{bmatrix} \dot{\theta} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \theta \\ q \end{bmatrix} + \begin{bmatrix} 0 \\ J_{\text{YY}}^{-1} \end{bmatrix} u_3$$

$$\begin{bmatrix} \dot{\psi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \psi \\ r \end{bmatrix} + \begin{bmatrix} 0 \\ J_{\text{ZZ}}^{-1} \end{bmatrix} u_4$$

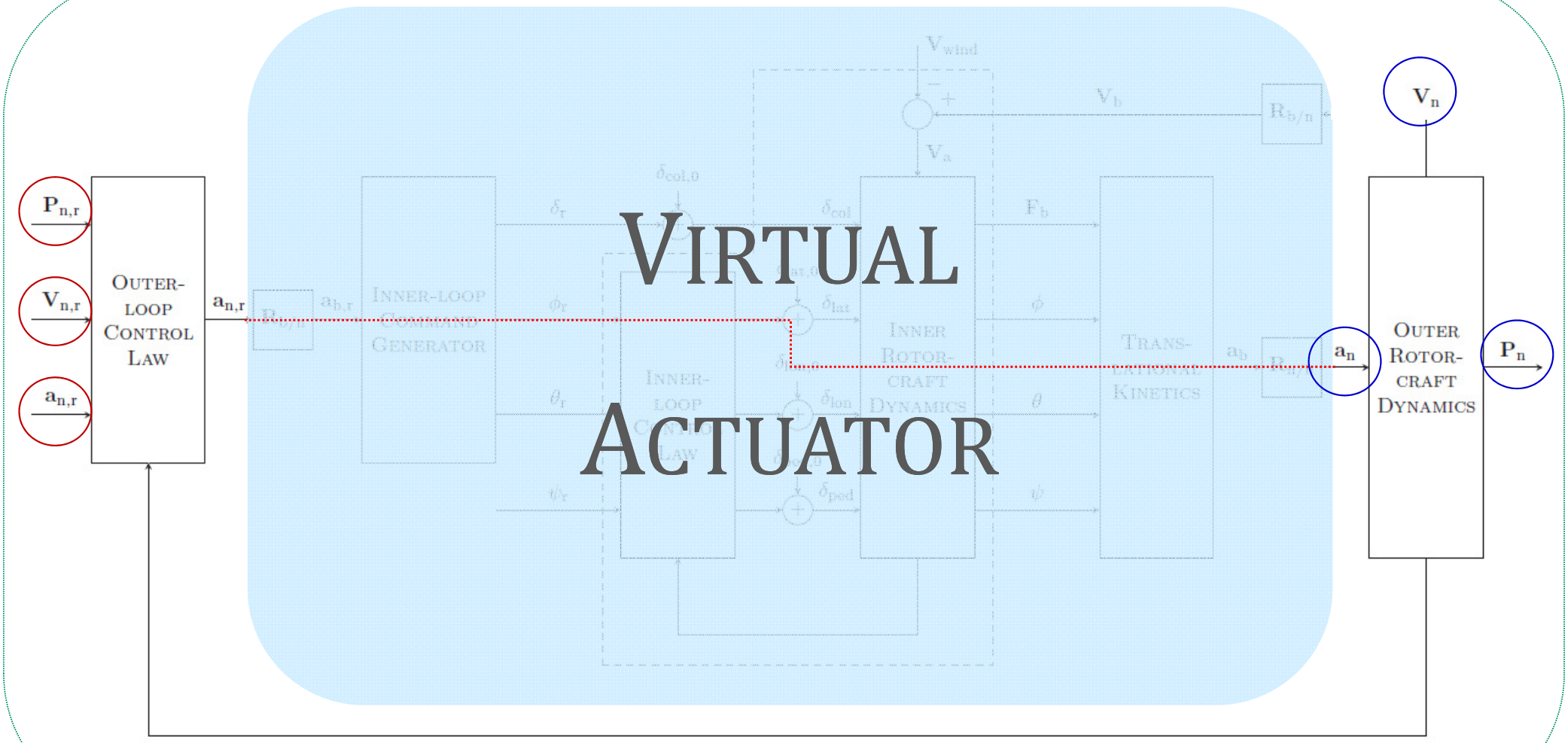
Exercise: Using techniques learnt from EE3331C to design a PD controller for each channel such that

$$\phi \to \phi_{\text{r}}$$

$$\theta \to \theta_{\text{r}}$$

$$\psi \to \psi_{\text{r}}$$

Outer-loop control system design setup…



Properties of the outer-loop dynamics…

It can also be verified that coupling among each channel of the outer loop dynamics is very weak and thus can be ignored. As a result, all the x, y and z channels of the rotorcraft dynamics can be treated as decoupled and each channel can be characterized by

$$\begin{pmatrix} \dot{p}_* \\ \dot{v}_* \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} p_* \\ v_* \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} a_*$$

where p_* is the position, v_* is the velocity and a_* is the acceleration, which is treated a control input in our formulation.

For such a simple system, it can be controlled by almost all the control techniques available in the literature, which include the most popular and the simplest one such as PID control…

Exercise: Design a PD control law for the above system to track a reference position.

RPT outer-loop control law…

RPT control is capable of utilizing all possible information available in its controller structure. Such a feature is highly desirable for flight missions involving complicated maneuvers, in which not only the position reference is useful, but also its velocity and even acceleration information are important or even necessary to be used in order to achieve a good overall performance.

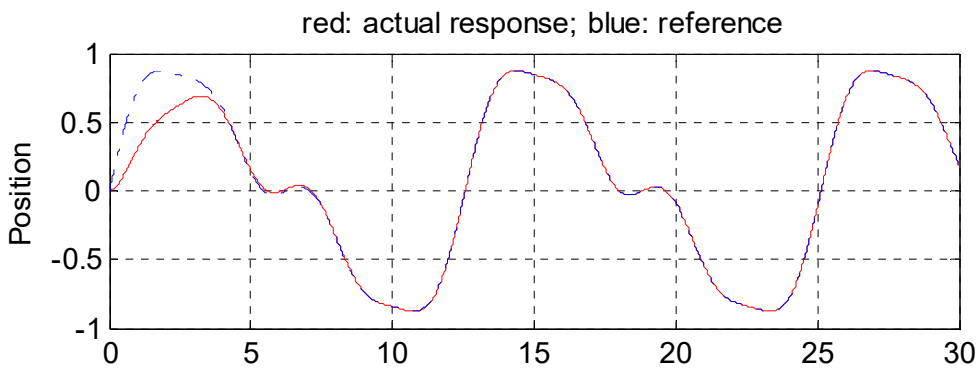
For each x, y, z channel, its RPT control law can be characterized as follows

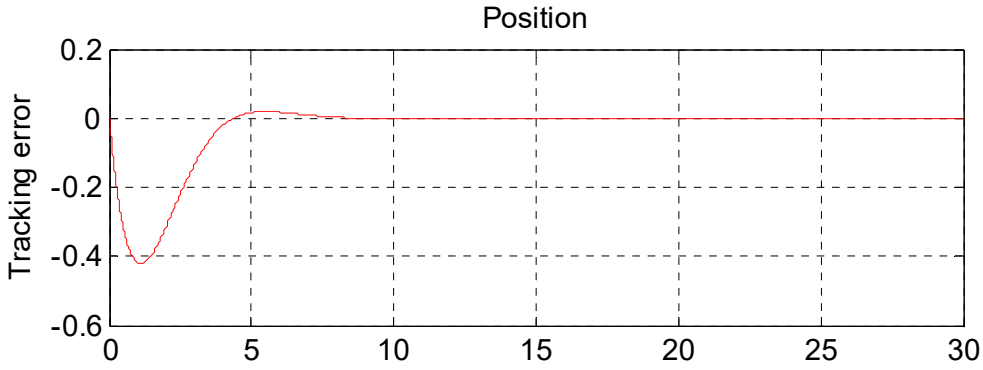
$$a_* = -\begin{bmatrix}\omega_n^2 & 2\varsigma\omega_n\end{bmatrix}\begin{pmatrix}p_* \\ v_*\end{pmatrix} + \left(\omega_n^2\right)p_r + \left(2\varsigma\omega_n\right)v_r + a_r$$

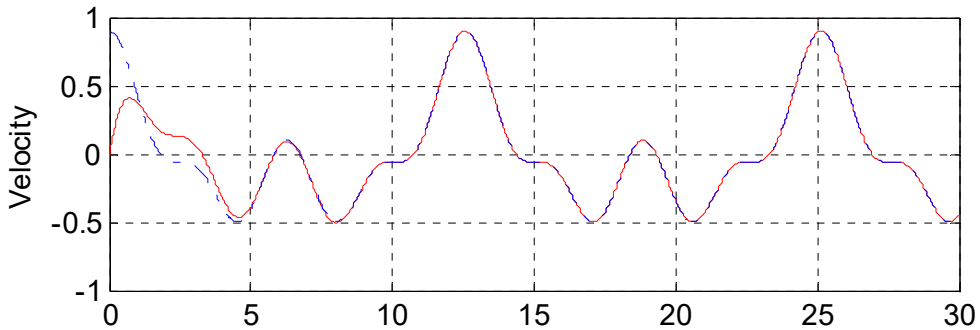
where p_r, v_r and a_r, are respectively the position, velocity and acceleration references; ζ is the damping ratio and ω_n is the natural frequency of the closed-loop system, which can be selected in accordance with the physical properties of the specific channel.

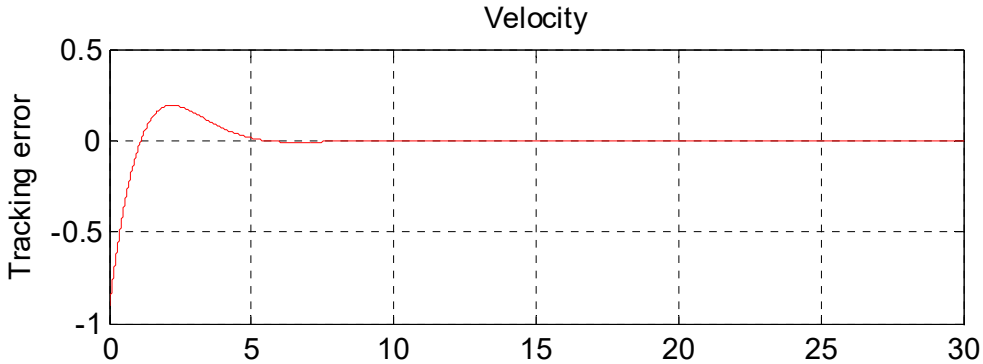
We will notice that such a control structure will link smoothly with the navigation block (i.e., path planning and trajectory generation) and give smoother flight performance.

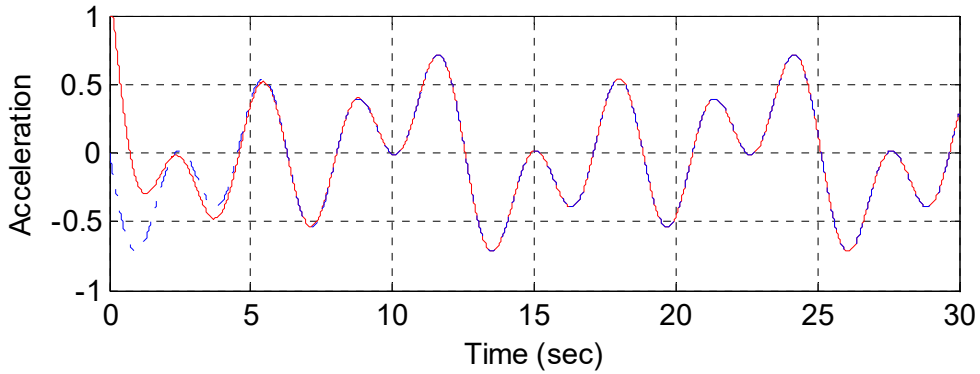
Simulation of RPT control with $\zeta = 0.7$ & $\omega_n = 1$...

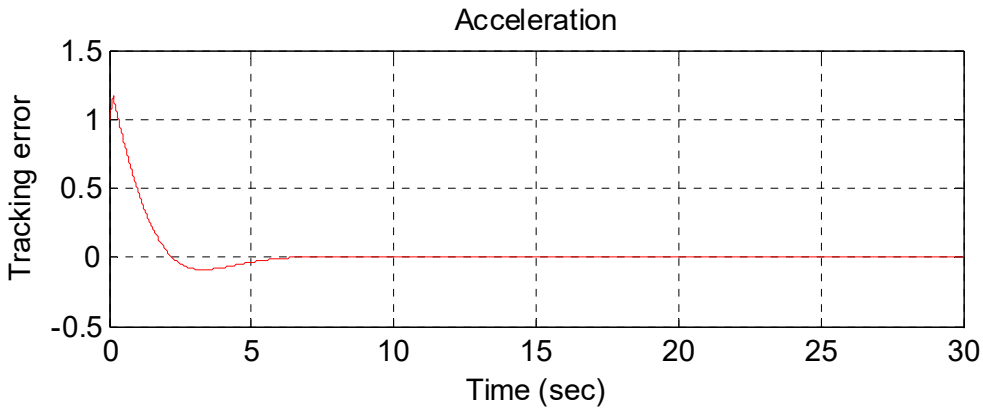












A brief view on the development of control techniques…

- **Classical control**

  PID control, developed in 1940s and used heavily for in industrial processes.

- **Optimal control**

  Linear quadratic regulator control, Kalman filter, H_2 control, developed in 1960s to achieve certain optimal performance.

- **Robust control**

  H_∞ control, developed in 1980s & 90s to handle systems with uncertainties and disturbances and with high performances.

- **Nonlinear control**

  Still on-going research topics, developed to handle nonlinear systems with high performances.

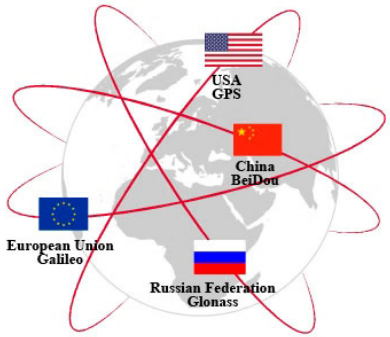
- **Intelligent control**

  Knowledge-based control, adaptive control, neural and fuzzy control, etc., developed to handle systems with unknown models.

Measurement signals…

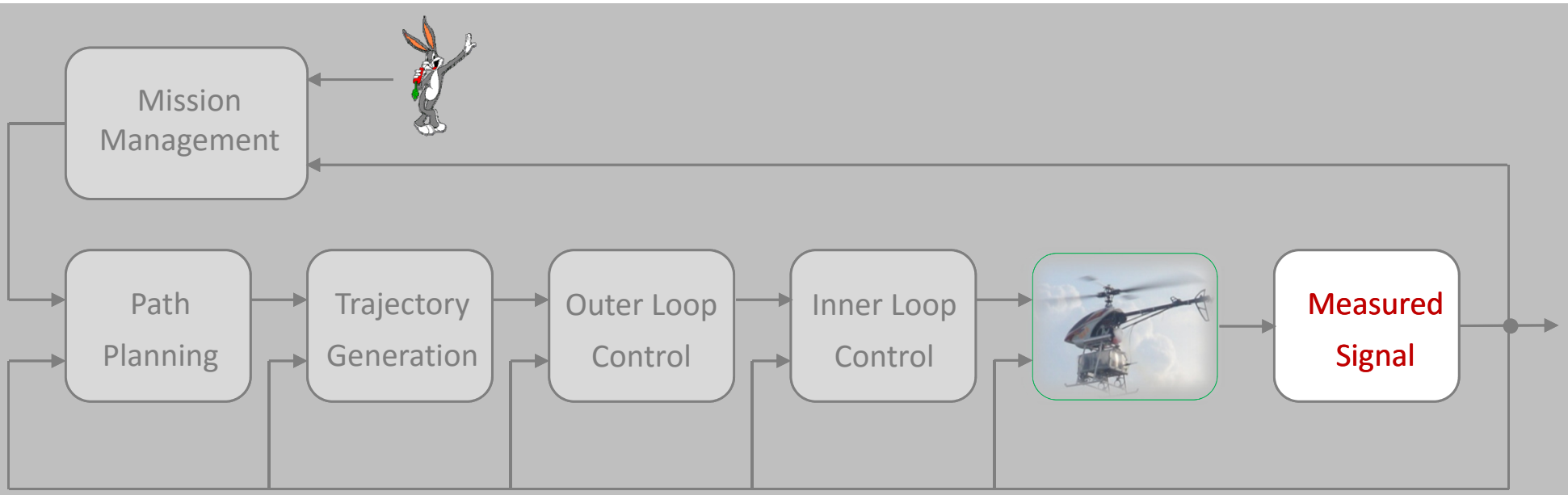
Measurement comes from inertial sensors (gyroscope, accelerometer, magnetometer) and

- External positioning systems
  - GPS
  - Beidou
  - VICON
  - UWB



or

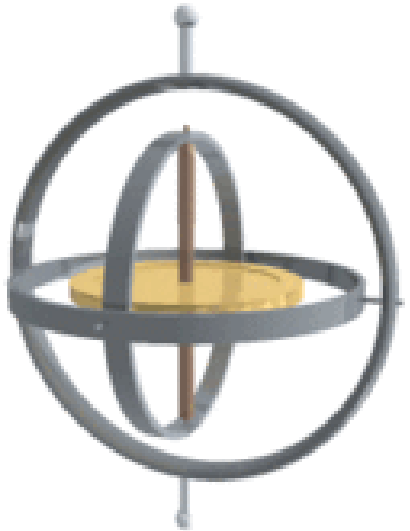
- Internal sensor systems
  - Vision
  - LiDAR
  - Radar
  - Sonar

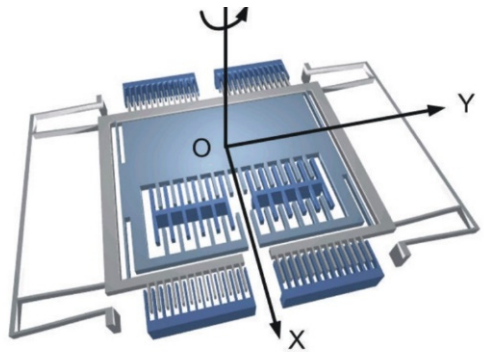


Gyroscope…

A **gyroscope** is a spinning wheel or disc in which the axis of rotation is free to assume any orientation by itself. When rotating, the orientation of this axis is unaffected by tilting or rotation of the mounting, according to the conservation of angular momentum. Because of this, gyroscopes are useful for measuring or maintaining orientation.

Inexpensive vibrating structure gyroscopes manufactured with MEMS technology have become widely available nowadays. These are packaged similarly to other integrated circuits and may provide either analog or digital outputs. In many cases, a single part includes gyroscopic sensors for multiple axes. Some parts incorporate multiple gyroscopes and accelerometers to achieve output that has six full degrees of freedom.

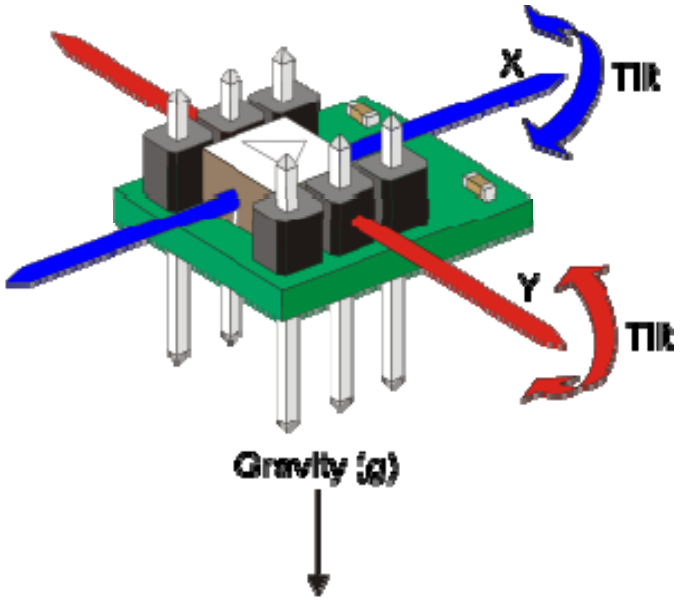




MEMS Gyroscope

Accelerometer & magnetometer…

An **accelerometer** is a device that measures proper acceleration; proper acceleration is not the same as coordinate acceleration (rate of change of velocity). For example, an accelerometer at rest on the surface of the Earth will measure an acceleration due to Earth's gravity, straight upwards (by definition) of $g \approx 9.81\ m/s^2$. By contrast, accelerometers in free fall will measure zero.



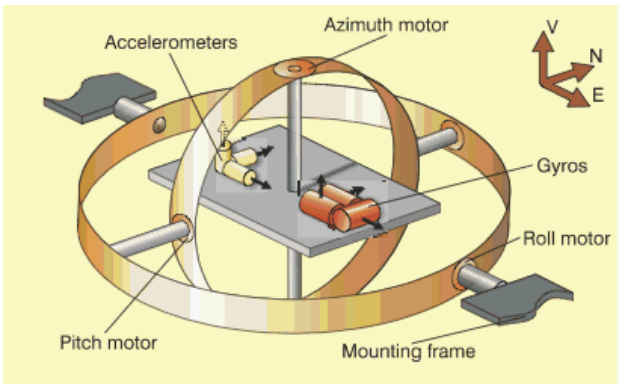
A **magnetometer** is an instrument that measures magnetism—either magnetization of magnetic material like a ferromagnet, or the direction, strength, or the relative change of a magnetic field at a particular location. A **compass** is a simple example of a magnetometer, one that measures the direction of an ambient magnetic field.

Inertial navigation system (INS)

An **inertial navigation system** (INS) is a navigation aid that uses a computer, motion sensors (accelerometers) and rotation sensors (gyroscopes) to continuously calculate via dead reckoning the position, orientation, and velocity (direction and speed of movement) of a moving object without the need for external references. It is used on vehicles such as ships, aircraft, submarines, guided missiles, and spacecraft. Other terms used to refer to inertial navigation systems or closely related devices include inertial guidance system, inertial instrument, **inertial measurement unit** (IMU).

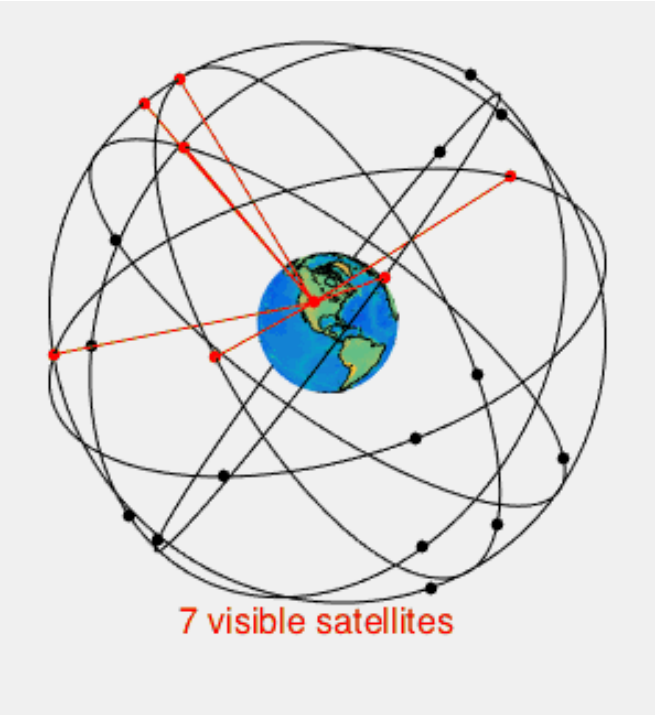
An INS is capable of providing the following information of unmanned vehicles:

- Accelerations
- Velocities
- Rotating angles
- Heading angles



Global positioning system (GPS)

The **Global Positioning System** (**GPS**) is a space-based radio-navigation system owned by the US Government and operated by the US Air Force. It is a global navigation satellite system that provides geolocation and time information to a GPS receiver in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites. The GPS system provides critical positioning capabilities to all users around the world.



GPS is widely used by drones for outdoor applications nowadays. The incorporation of GPS receivers in drones allows for waypoint navigation. It allows a drone to autonomously fly to pre-programmed points, can instruct the drone how fast, how high, and where to fly.

Global positioning system (cont.)…

Space Segment

- NAVSTAR: 30 satellites with atomic clocks
- At least 5 always visible (usually 7-8)
- 20000 km altitude 13000 km/h speed

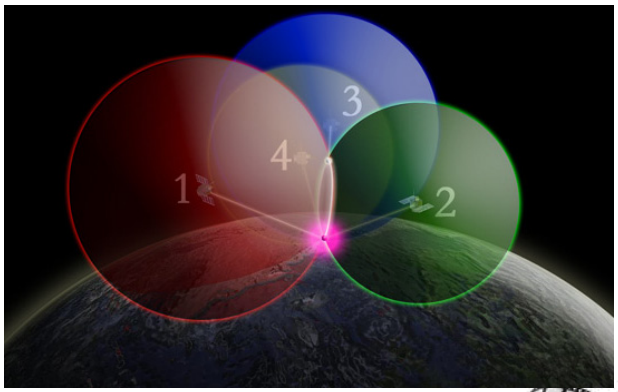
Control Segment

- Master control for satellites
- Update satellite position (Kalman Filter) and atomic clocks

User Segment

- General GPS receiver with antenna and precise clock (crystal)
- Geoid model for height estimation

Navigation Equation



The receiver uses messages received from satellites to determine the satellite positions and time sent.

For n satellites, the equations to satisfy are:

$$(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = \left[(\tilde{t}_i - b - s_i)c\right]^2, \quad i = 1, \cdots, n$$

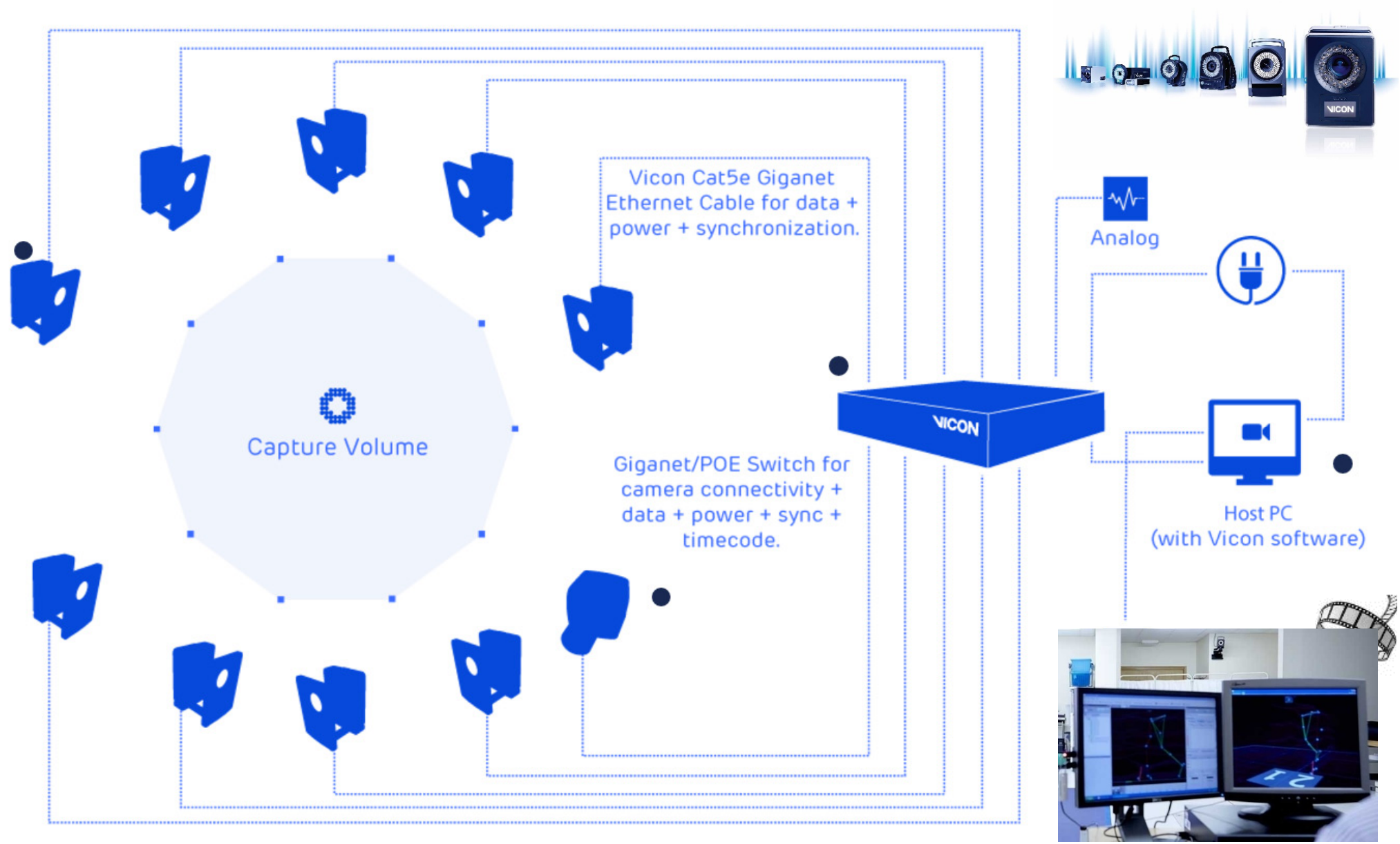
where (x_i, y_i, z_i): position of satellite i; (x, y, z): receiver position; c: speed of light; b: receiver's clock bias; $\tilde{t}_i$: on-board receiver clock; s_i: satellite time.

Least Squares Solution

$$(\hat{x}, \hat{y}, \hat{z}, \hat{b}) = \arg\min_{(x,y,z,b)} \sum_i \left(\sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} + bc - p_i\right)^2$$

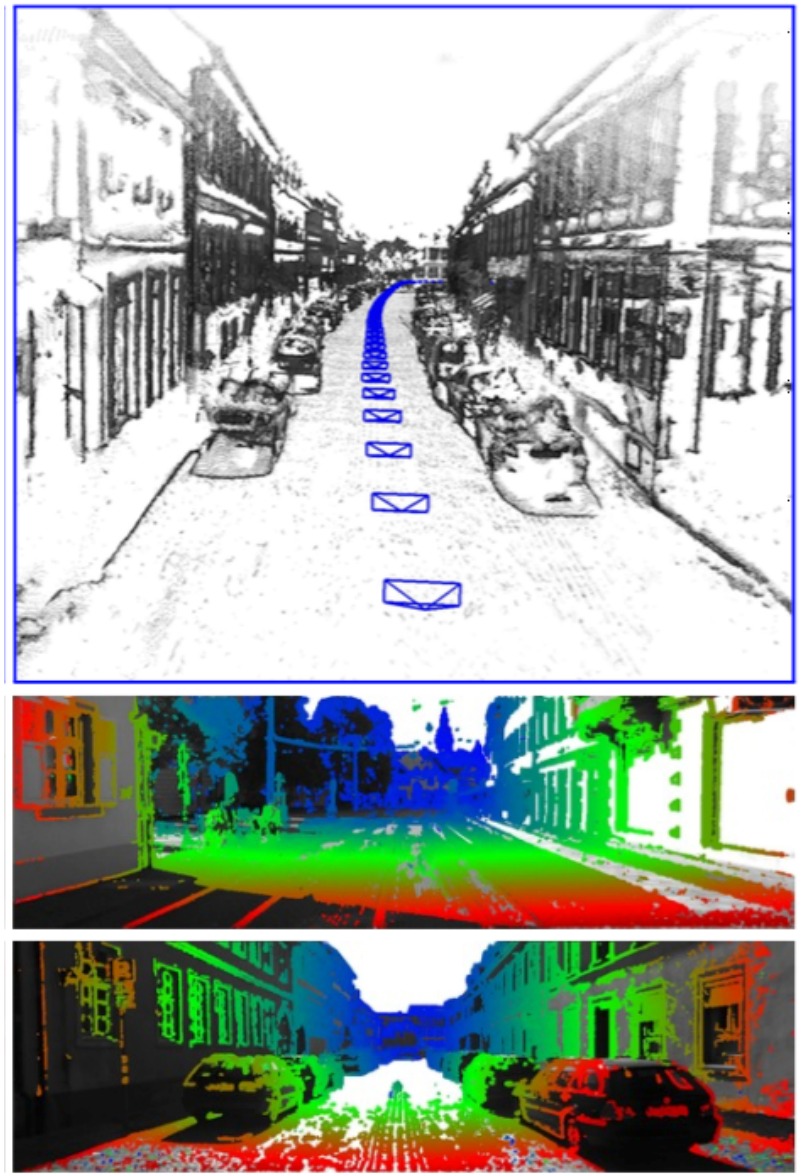
where $p_i = (\tilde{t}_i - s_i)\,c$ is pseudorange. For 3D localization, we need at least 4 satellites!

VICON: a motion capture positioning system…



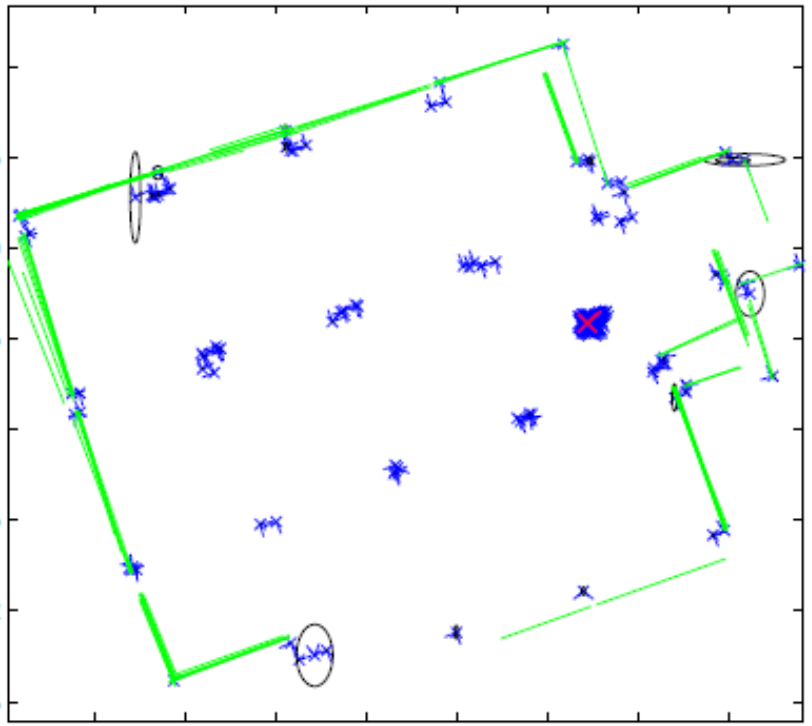
Here is a link to YouTube for a tutorial on VICON…

Sensor-based positioning systems… SLAM



Vision based 3D Dense SLAM

LiDAR based 2D SLAM

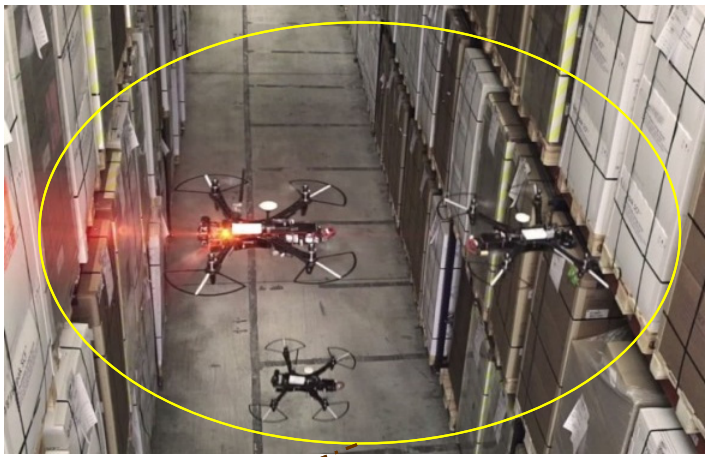


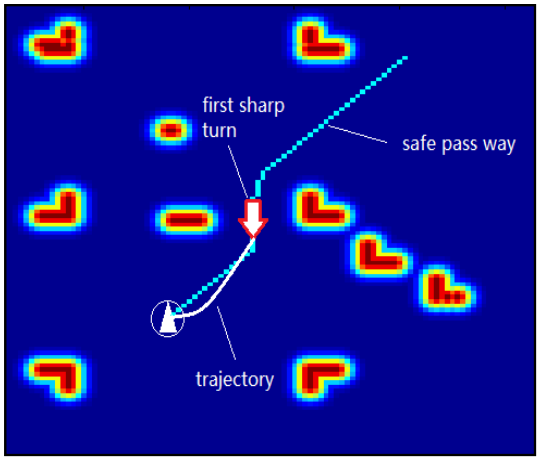
Topological SLAM techniques…

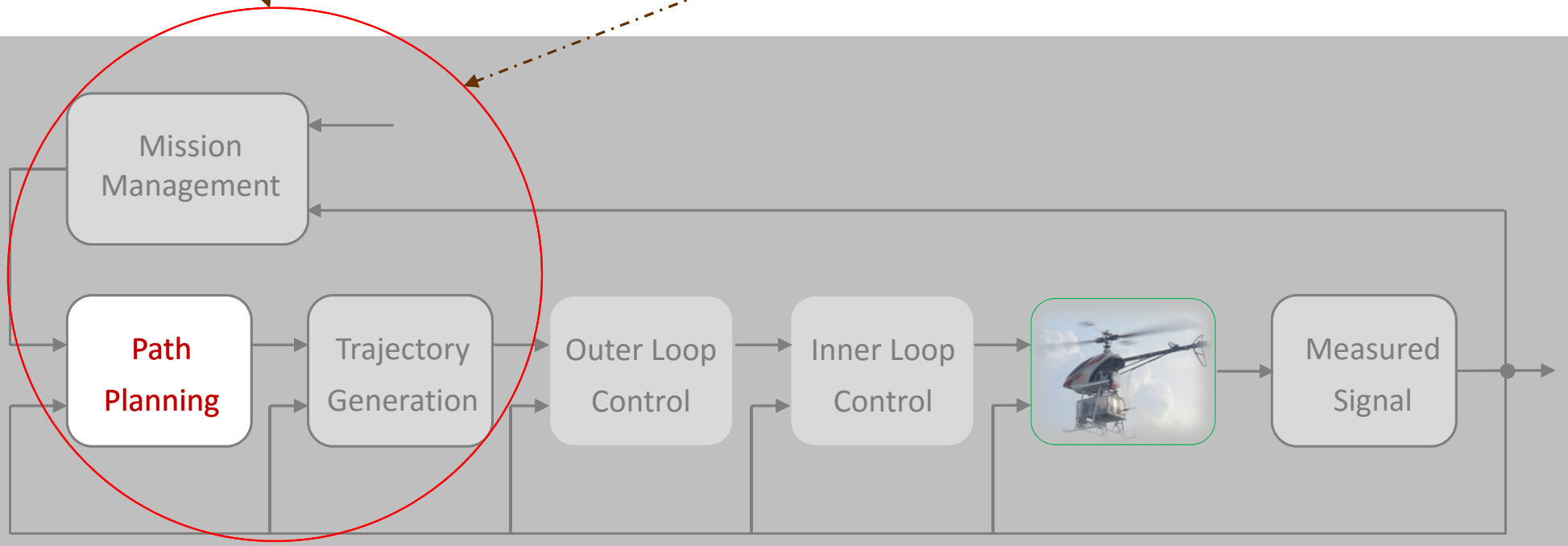
List of topological SLAM methods and their features/associated extraction algorithms…

- Line extractor (LiDAR)
  - Split & merge, expectation maximization, Hough transform, RANdom Sample Consensus
- Haar wavelets (Vision)
  - Fourier transform similar approach
- Edge based detector (Vision)
- Keypoint detectors
  - Blob Detectors: Scale Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Center Surround Extremas (SenSurE)
  - Other Detectors: corner (KLT)
- Fingerprint of places
  - FACT, DP-FACT
- Affine covariant region detector
  - Harris-affine, Maximally Stable Extremal Regions (MSER)
- Bayesian surprise

Path planning…



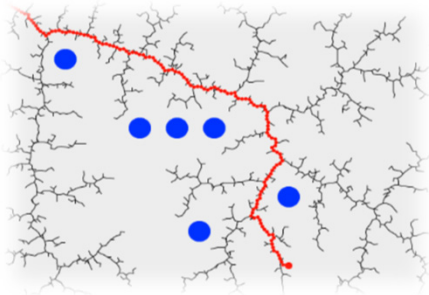


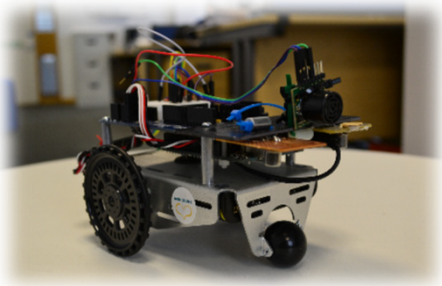


Motion planning…

Motion planning (also known as the **navigation problem**) is a term used in robotics or unmanned systems in general for the process of breaking down a desired movement task into discrete motions that satisfy movement constraints and possibly optimize some aspect of the movement, such as time and/or energy.

For example, consider navigating a UAV inside the T-Lab Level 4 VICON room to a distant waypoint as in the project for the 2nd part of EE4308. It should execute this task while avoiding walls and not hit the poles placed in the room. A motion planning algorithm would take a description of these tasks as input, and produce the speed and turning commands sent to the drone. It might address issues related to multiple UAVs, more complex tasks, different constraints (e.g., velocity, acceleration, heading), and uncertainty (e.g. imperfect models of the environment or UAV). Motion planning algorithms are widely used for ground robotic systems as covered in Part 1.





Motion planning…

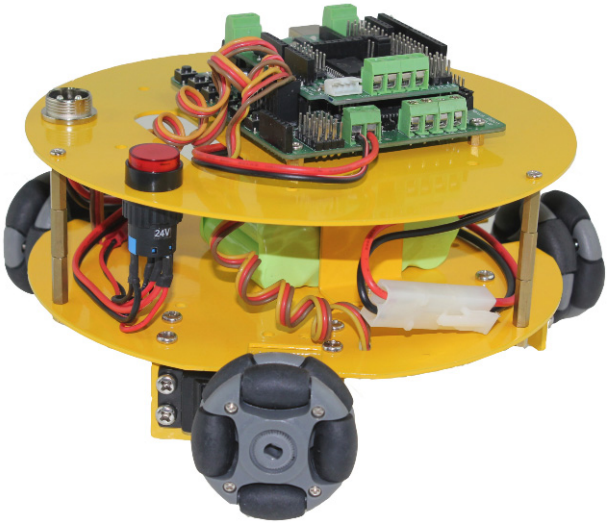
Motion planning is to search an appropriate path for unmanned vehicles that would allow the vehicles to travel safely and efficiently among obstacles. It consists of two parts: the geometrical path planning and the trajectory generation or optimization.

The **path planning** is done in the configuration space with all dynamics of the vehicles being ignored. This covers a lot of classical problems such as the piano mover's problem.

The **trajectory generation** is the process of designing a trajectory that minimizes (or maximizes) some measure of performance while satisfying a set of constraints of the dynamic model of the unmanned vehicles. Generally speaking, trajectory generation or optimization is a technique for computing an open-loop solution to an optimal control problem. It is often used for systems where computing the full closed-loop solution is either impossible or impractical.

Discretized state space

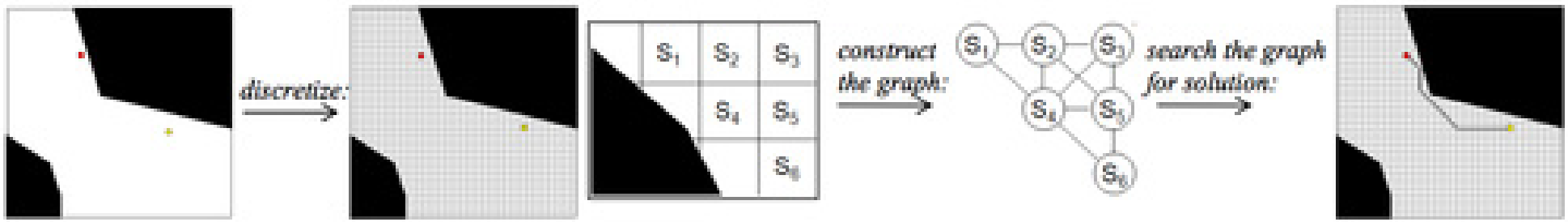
The so-called ‘state’ of a vehicle varies quite a lot. Depending on the vehicles we are talking about, it might includes its position, body angle and their corresponding derivatives. For example, a small omnidirectional robots moving at low speed, one might only need to concern is position on a 2D flat surface, i.e., the work space. Therefore, its state consists of only (x, y).



For quadrotors, the state consists of (x, y, z, roll, pitch, yaw). If the drone is moving in an agile way, its dynamic property might also be considered which enlarge its state space. We call the set of all possible states of the vehicle as the state space.

Search space discretization…

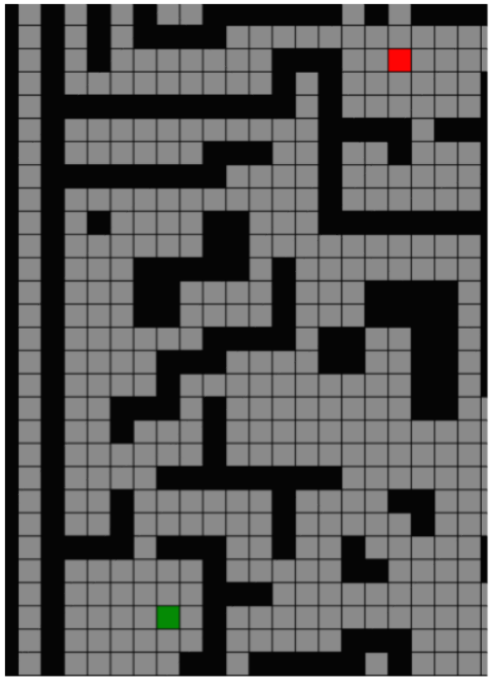
Return to the case of omnidirectional robot, where its state space is (x, y). We cut a part of the state space and discretize it as shown in the following figure. Each grid represents a unique discrete state (a pair of (x, y)) and is denoted as S1, S2, S3…



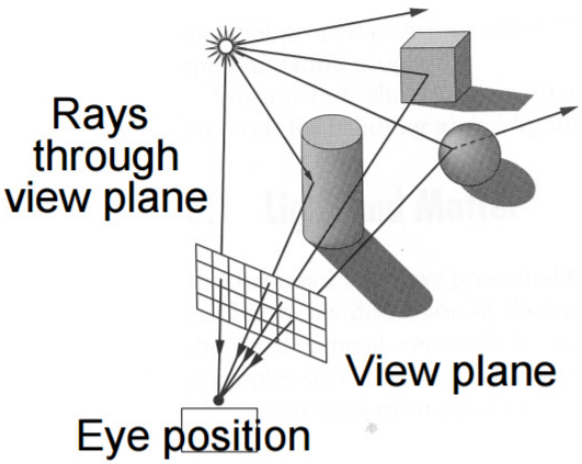
Since it is an omnidirectional vehicle, it can travel towards to its 8 neighbours, i.e., left, right, bottom, top, top left, top right, bottom left, bottom right. By such connection among all the discrete states, a graph emerges with all the discrete states as nodes and the connection to its 8 neighbours as edges. On the same graph, we can set unreachable states/nodes to represent the obstacles. And a path on the graph is a series of inter-connected node/states like S1→ S2 → S5.

Collision checking…

As depicted in the figure, there are states that cannot be reached by the robot (black area), being either walls or other obstacles. Entering those states will cause collision. This is the case of a 2D state space. For other types of vehicles (usually of higher dimensional state space), there might be other constraints other than these physical obstacles. For example, an airplane might have its maximum reachable velocity as well as the minimum velocity to keep it in the air.



One of the most basic method of performing collision checking is the ray casting method. Many other more advanced method are also based on it. The basic idea is to express the vehicle's path as a series of connected line segments. Then a ray is cast from the start point to the end point of each line segment. If the ray passes through any obstacle area, a collision is detected.



A search algorithm…*

A* is a best-first search algorithm, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.), and among these paths it first considers the ones that appear to lead most quickly to the solution.

It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node.

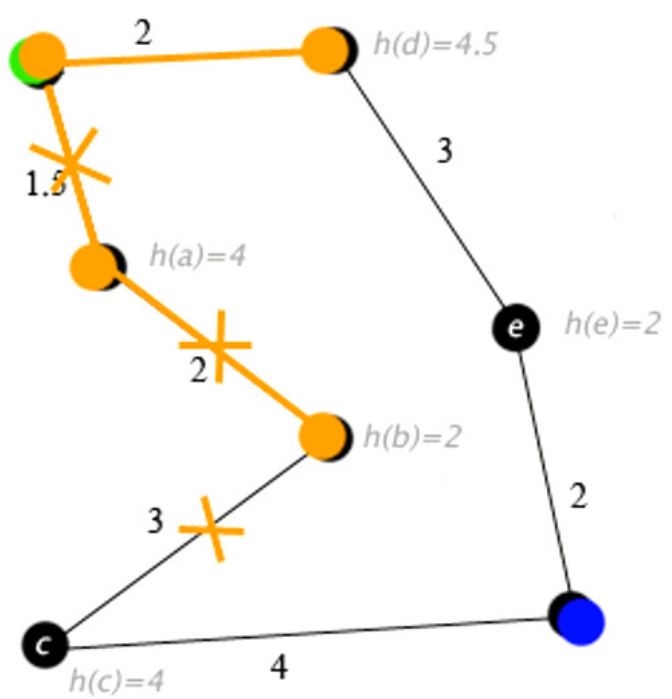
At each iteration of its main loop, A* needs to determine which of its partial paths to expand into one or more longer paths. It does so based on an estimate of the cost (total weight) still to go to the goal node. Specifically, A* selects the path that minimizes

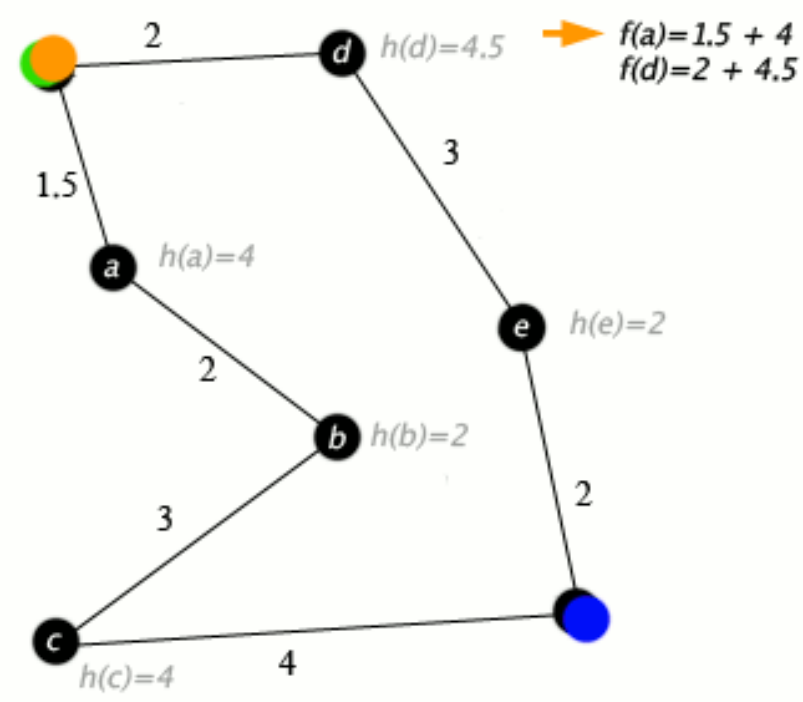
$$f(n) = g(n) + h(n)$$

where n is the last node on the path, $g(n)$ is the cost of the path from the start node to n, and $h(n)$ is a heuristic that estimates the cost of the cheapest path from n to the goal.

A search example…*

The following example is an A* search algorithm in action where nodes are cities connected with roads and $h(x)$ is the straight-line distance to target point:





Key: green: start; blue: goal; orange: visited

A search illustration…*

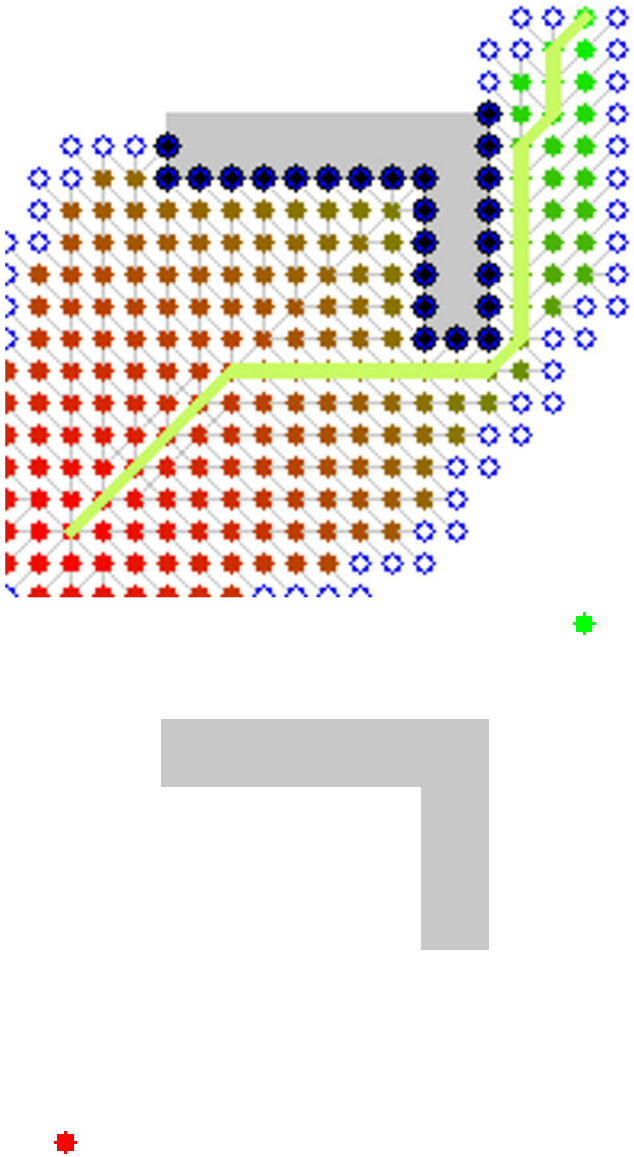
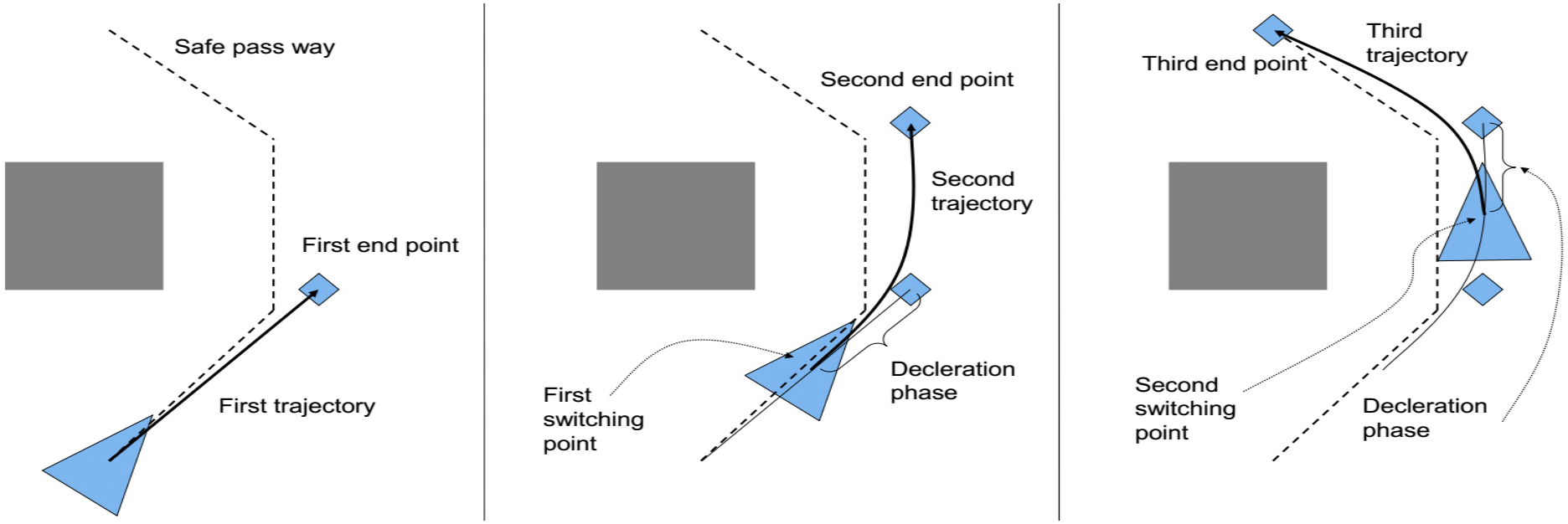


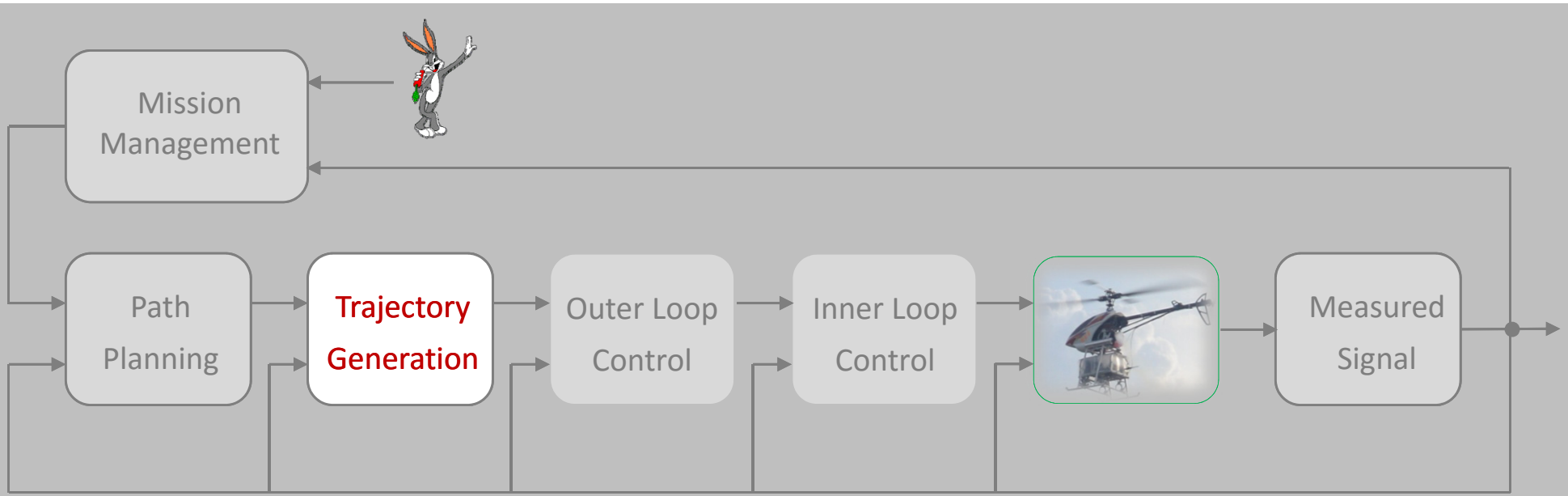
Illustration of A* search for finding path from a start node to a goal node in a robot motion planning problem. The empty circles represent the nodes in the *open set*, i.e., those that remain to be explored, and the filled ones are in the *closed set*. Color on each closed node indicates the distance from the start: the greener, the farther. One can first see the A* moving in a straight line in the direction of the goal, then when hitting the obstacle, it explores alternative routes through the nodes from the open set.

Other path planning search techniques…

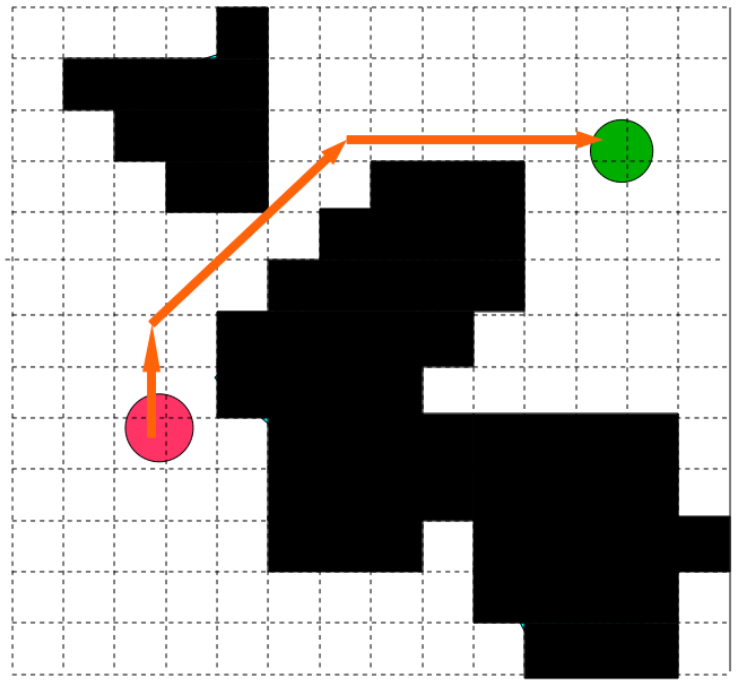
- **Dijkstra's algorithm**: Classic graph searching algorithm originated from dynamic programming
- **R* algorithm**: Optimized based on A* which depends less on the heuristic function
- **D* lite algorithm**: An incremental planning algorithm, build to handle a dynamic changing and unknown environment
- **JPS algorithm**: Jumping point search algorithm - an improved version of A*, created to handle the unnecessary zig-zag created by standard A* algorithm
- **PRM**: Probability roadmap, using random sampling to created connected graph, then use traditional graph search method to generate the map
- **RRT**: Rapidly-exploring random tree, combine the sampling and searching in the same algorithm, however it does not guarantee an optimal solution
- **RRT***: An improved version of RRT, guarantee an optimal solution
- **BIT***: Batch informed trees, a combination of random sampling algorithm with the A* algorithm

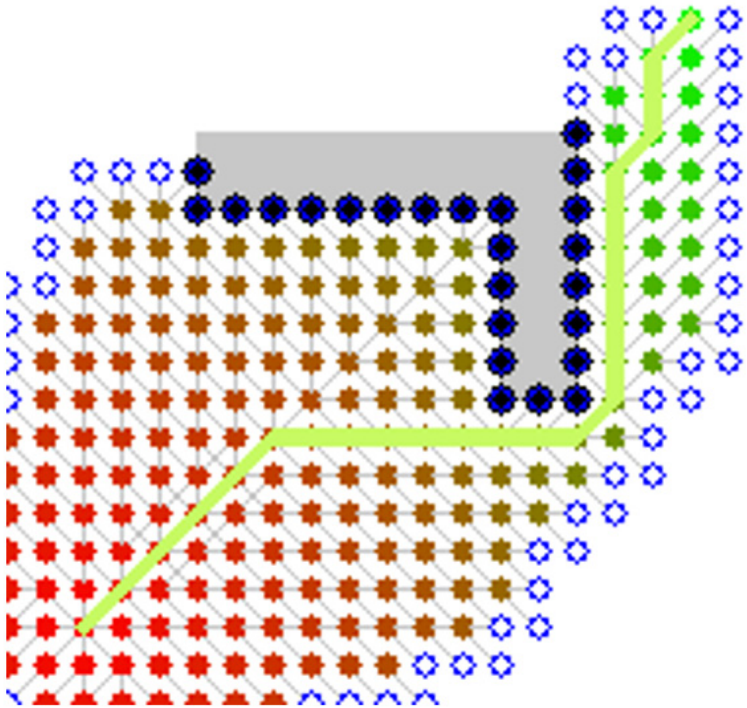
Trajectory generation…





With the A* algorithm, we are able to obtain a series of connected line segments shown as in the figures below





The next question one would ask: How can these line segments be realized (or used to guide) in a quadrotor drone?

Trajectory generation for quadrotor drones…

For a quadrotor drone to follow closely a pre-planned path, we need to specify a set of references to the outer-loop controller as depicted in the general unmanned system framework. For drones, the reference set should consist of the vehicle's

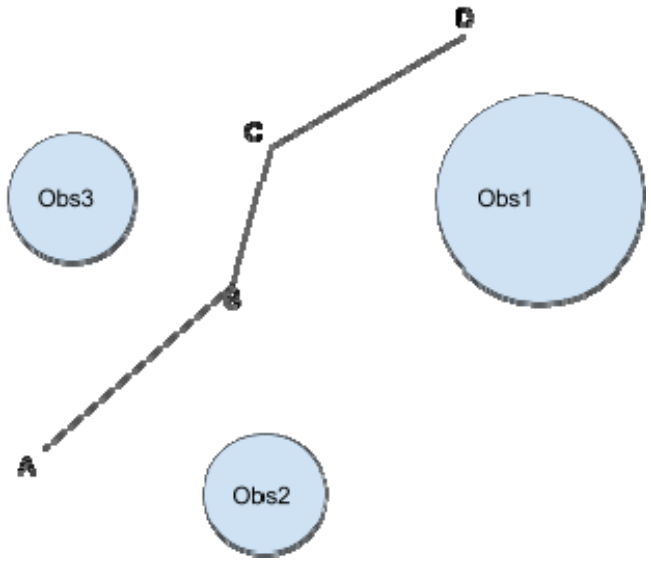
- 3-axis position references, x, y, z
- 3-axis velocity references, v_x, v_y, v_z
- 3-axis acceleration references, a_x, a_y, a_z

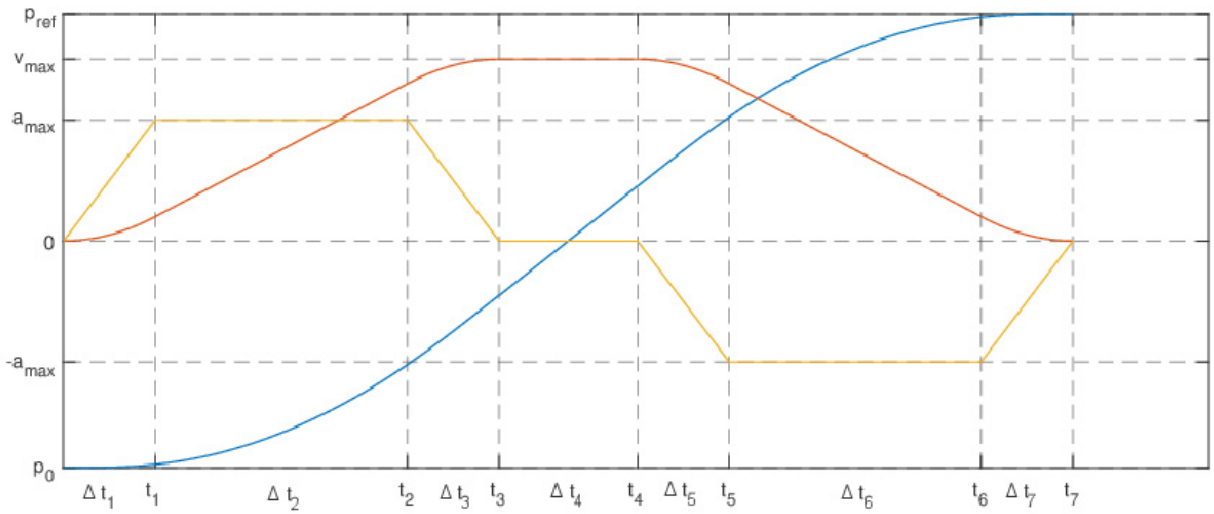
There are extra requirements on these references

- All of the reference signals must be continuous
- The velocity and acceleration must be limited. For the drones used in this module and for safety reason, we limit

$$|v_x|,\ |v_y|,\ |v_z| \le 2 \text{ m/s} \quad \text{and} \quad |a_x|,\ |a_y|,\ |a_z| \le 1.2 \text{ m/s}^2$$

The simplest way for the vehicle to travel on the segment path is to generate velocity profile along that line segment. For example,





red: velocity — yellow: acceleration — blue: position

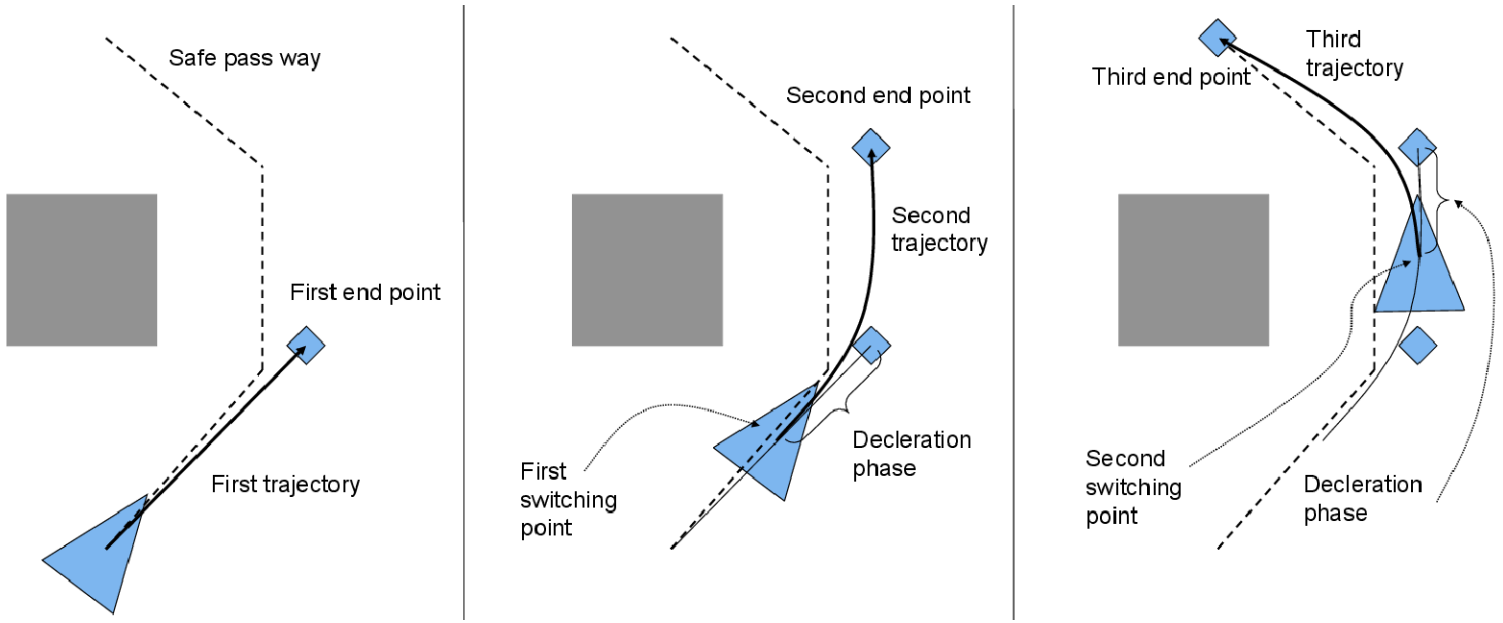
It is to generate trajectories for the vehicle to travel from A to D. For each line segment A–B, B–C, C–D, a velocity and acceleration profile shown above has to be generated.

The problem with this simple approach is that the drone will come to a full stop at the end of each line segment, such as point B and C.

Trajectory generation for quadrotor drones…

How can the drone fly from A→D as fast as possible without stops at the interim notes?

A possible solution is to 'switch' to the next line segment before it goes into full stop.



Step 1: Line segments are generated, vehicle flying towards the end of the 1st line segment.

Step 2: Before reaching the 1st end point to stop, switch to travel on the next line segment.

Step 3: By repeating the process in Step 2, vehicle could reach its target in a smoother way.

The difficulty is on maintaining the continuity of the reference signal for the entire path…

How to generate smooth trajectories?...

Trajectory generation techniques suitable for quadrotors...

- **Acceleration limited time optimal solution**

  Generate time minimal trajectory, under the velocity and acceleration constraint

- **Jerk limited time optimal solution**

  Time minimal trajectory, under the velocity, acceleration and jerk constraint

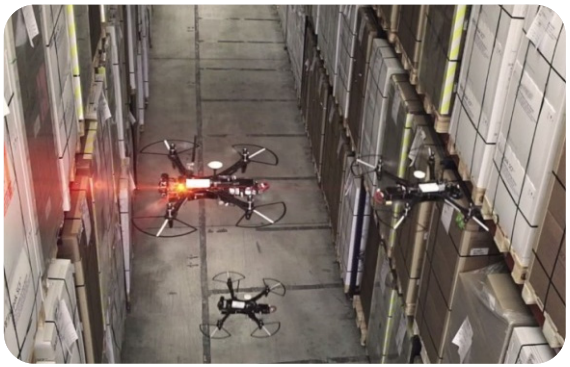
- **Polynomial spline based trajectory**

  Generate high quality, energy optimized trajectory, but requires the involving of numerical optimization

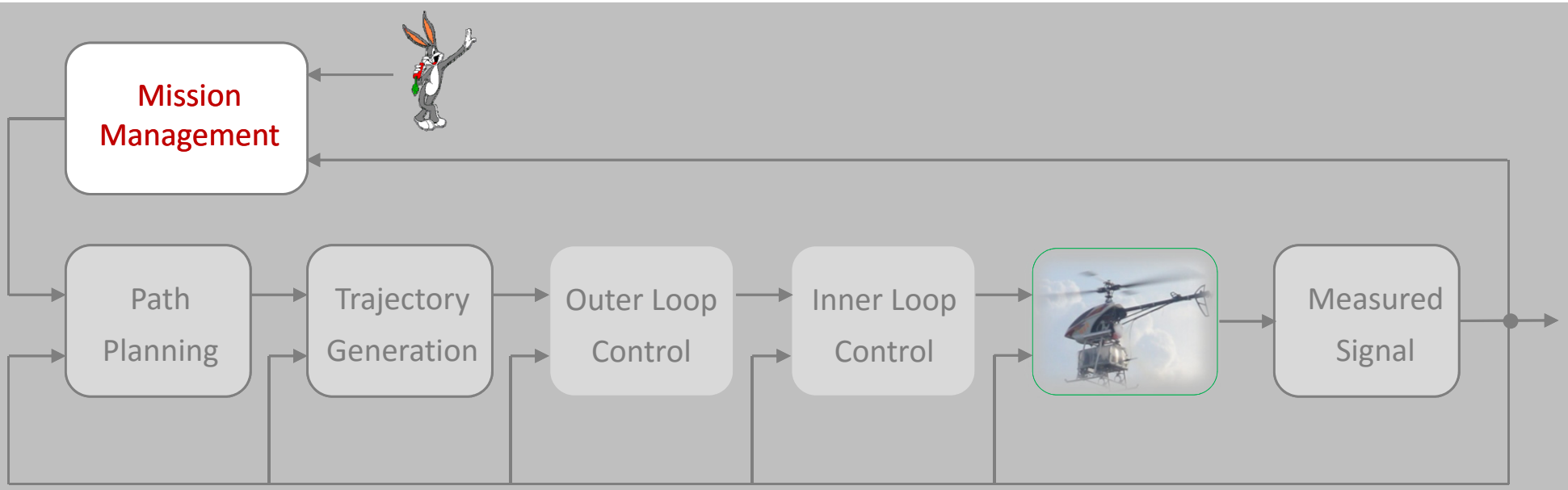
- **Dynamic programming**

  A search based trajectory generation method, very powerful for handling complex dynamics

Mission management…



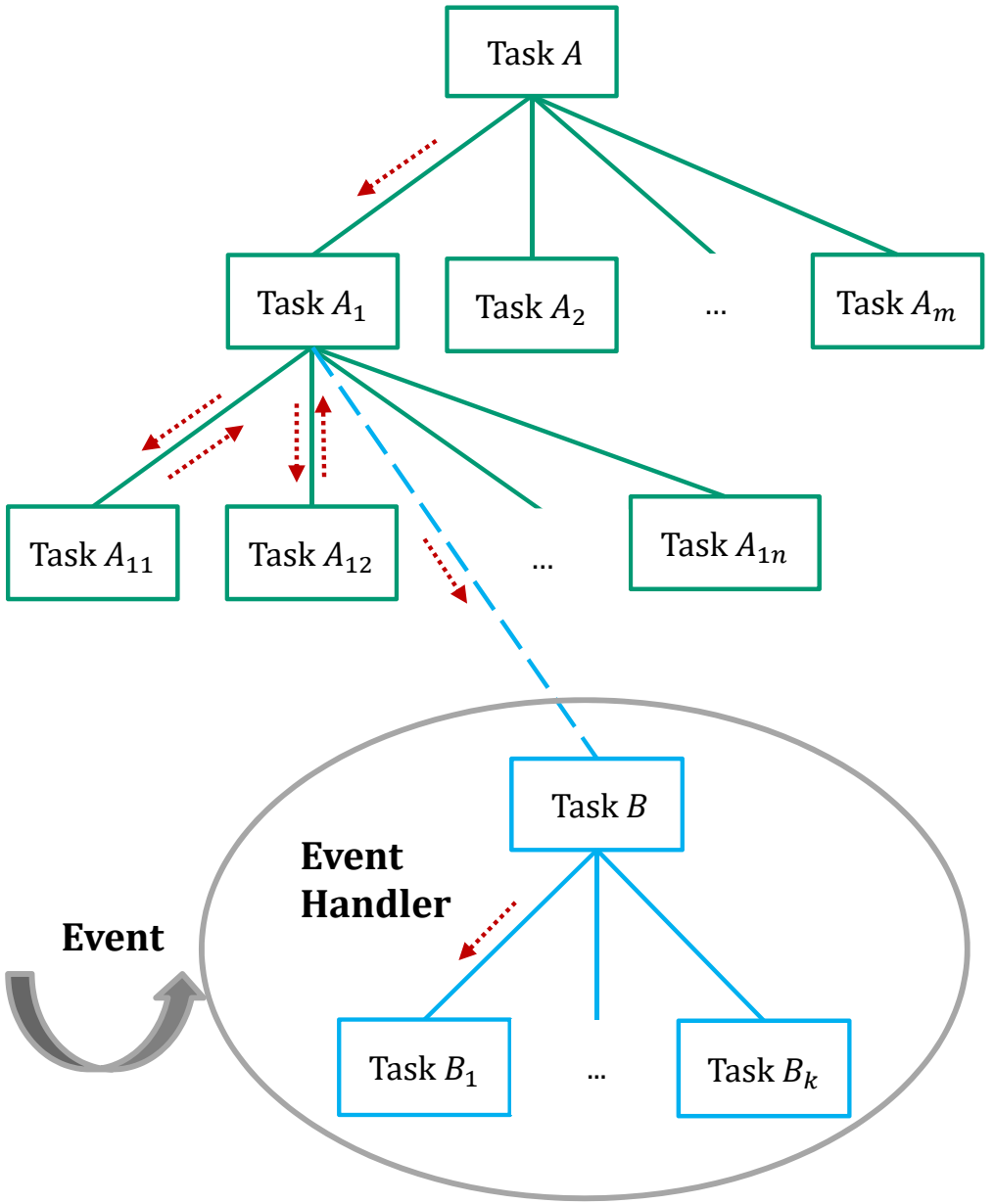
- State Machine
- Automata
- Deep learning? ……



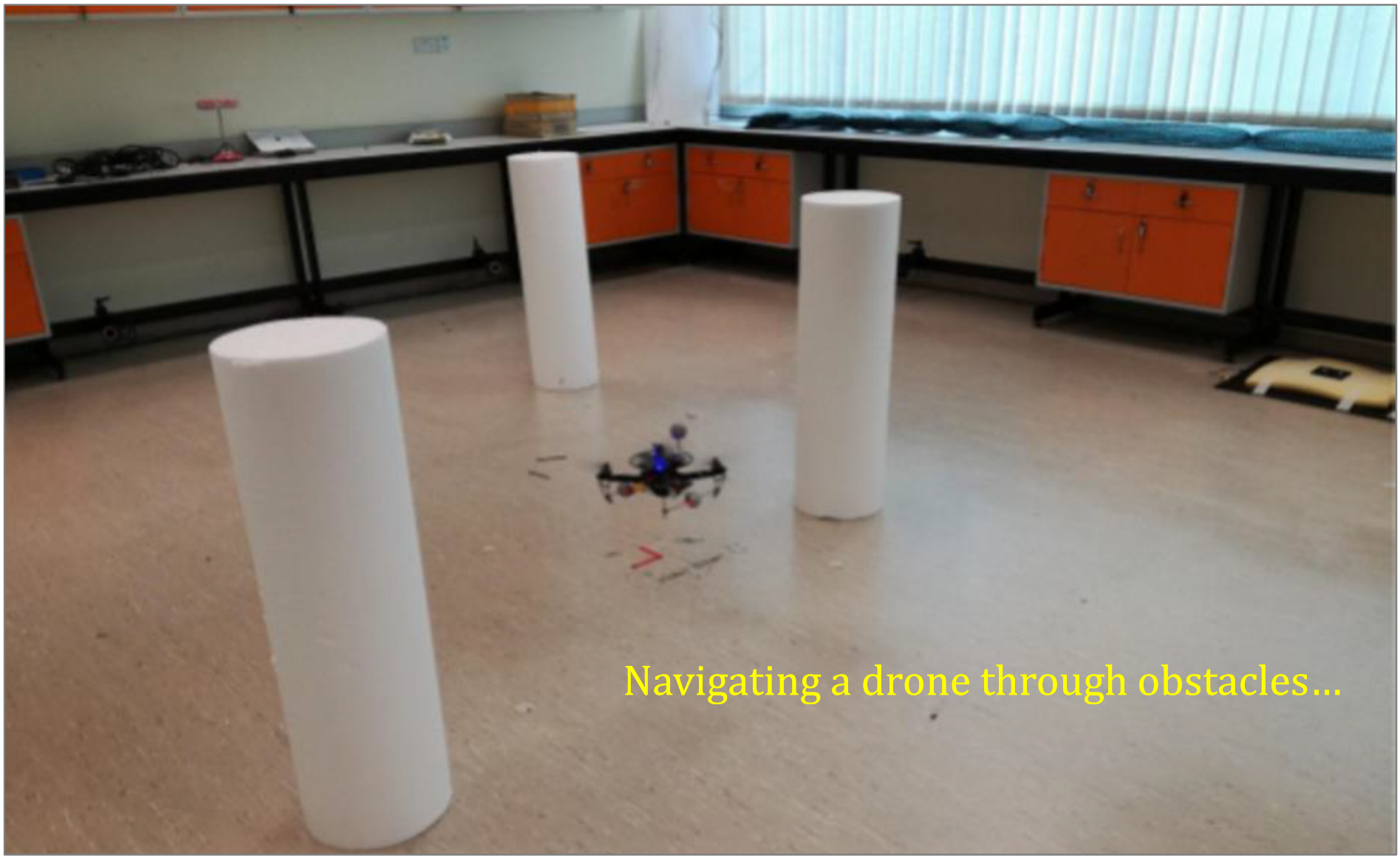
Framework of a mission management…

For a task-based mission, we can use a tree-based framework. The tasks are organized into a **tree** and executed in a manner of **depth-first traversal**. Each leaf node task contains only one single **action**. In other words, executing that leaf node task is equivalent to executing the corresponding action.

When an **event** occurs, a new **task** (event handler) is inserted to the current tree node as a sub-tree. Some events require an immediate termination of mission after the event handler is executed (such as LAND once the battery is low). In that case, the remaining tasks are removed from the tree accordingly.



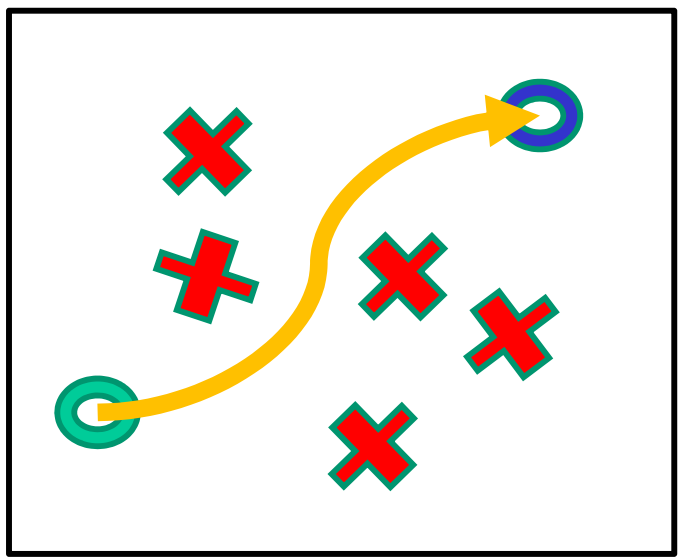
EE4308 Part 2 project…

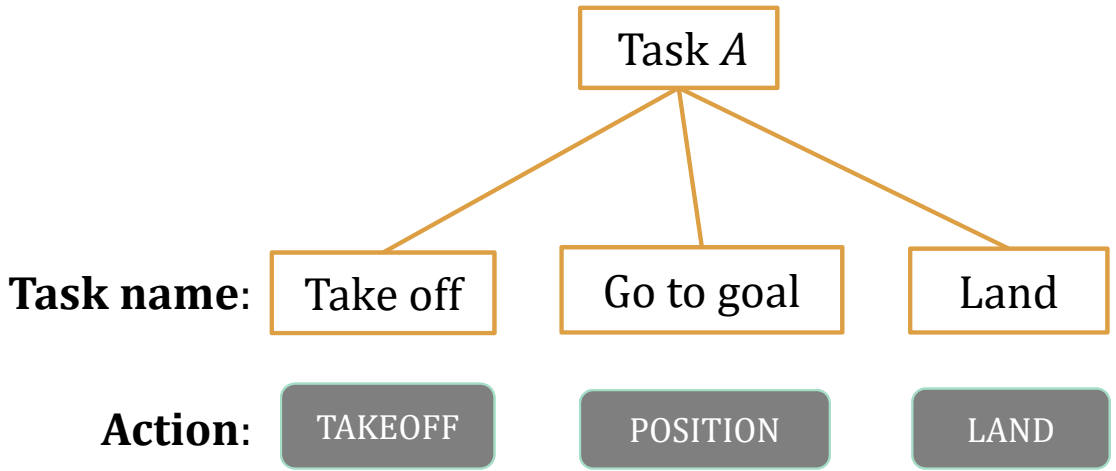


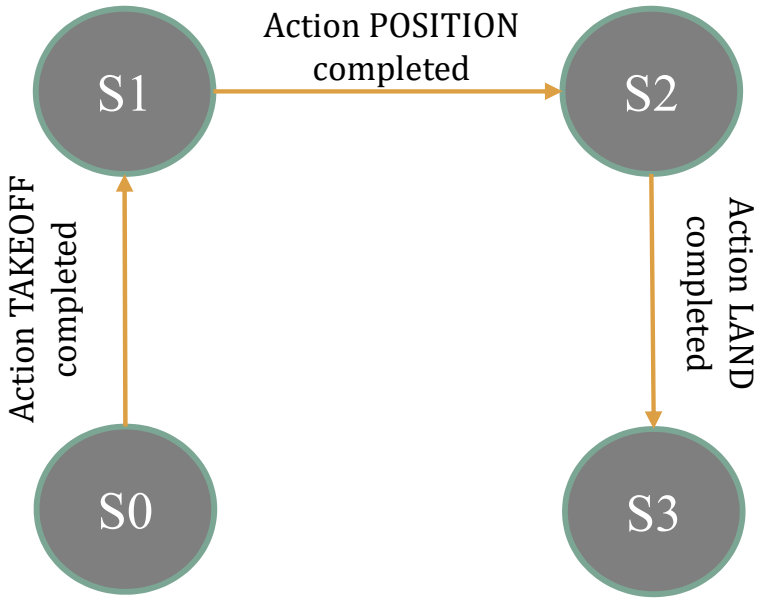
Mission management for EE4308 project…

Mission: Navigate through obstacles and get to the goal position. It can be decomposed into three sub-tasks, where each task will correspond to one action.

- **Task 1**: Take-off (TAKEOFF)
- **Task 2**: Go to goal (POSITION)
- **Task 3:** Land (LAND)







Project overview…

NUS National University of Singapore

Task 1

- Path planning or design of a trajectory for UAV to travel in a fixed confined space
- Make a simulator to show clearly the trajectory of UAV
- UAV must avoid all obstacles and the behavior should be shown in the simulator

15%

Task 2

- With the path designed in Task 1, implement and run it in an actual UAV
- Utilize VICON system to fly UAV autonomously
- Compare performance of simulation and actual flight

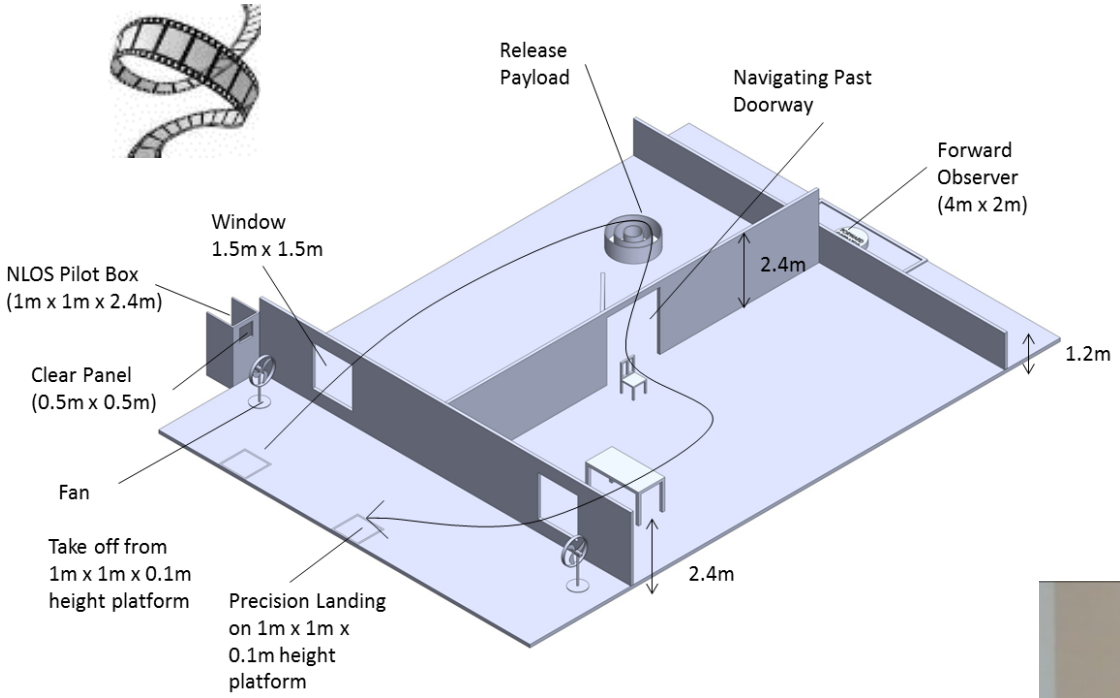
15%

Task 3

- 3 to 5 obstacles will be randomly placed
- Modify simulator of Task 1 to include random obstacles input (if needed)
- Implement and run the design path with the actual UAV in VICON system

10%

Simulation and actual flight experiment…



Simulation for flight missions involved in the 2017 Singapore Amazing Flying Machine Competition

Video taken during an actual flight demonstration to BP

Acknowledgement…

… Special thanks to the whole NUS UAV Research Group

and particularly to

Lai Shupeng, Lan Menglu, Lin Feng, Li Kun, Phang Swee King, Tian Hongyu, Li Jiaxin, Wang Kangli

for their help in preparing lecture materials …

♣

… To YouTube and Wikipedia …

That's all, folks!

Thank You!