EE5110/6110 Special Topics in Automation and Control

Autonomous Systems: Unmanned Aerial Vehicles

EE5062 Autonomous Systems

Dr. WANG Fei (elewf@nus.edu.sg) & Dr. HUANG Sunan (tslhs@nus.edu.sg)

National University of Singapore

Topic Outline…

1. Introduction to UAV systems
   - What is an unmanned aerial vehicle (UAV)?
   - UAV applications
   - UAV system structures
   - UAV platform design and construction

2. UAV flight dynamics modeling and control

3. UAV navigation and environmental sensing
   - GPS-aided navigation
   - GPS-less navigation and SLAM

Topic Outline…

4. Introduction to motion planning of UAVs
5. Optimal motion planning: AI-based A* search algorithm
6. Trajectory generation

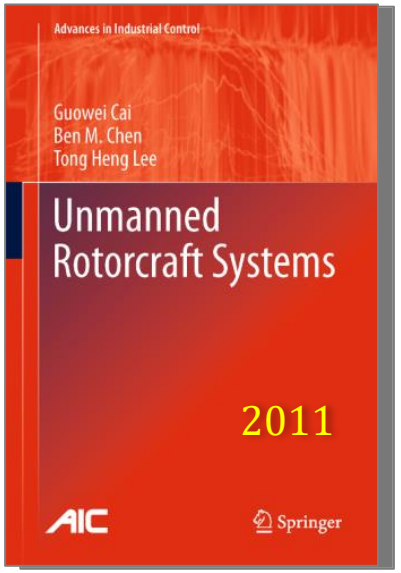
Continuous Assessments…

EE5110/6110 CA4

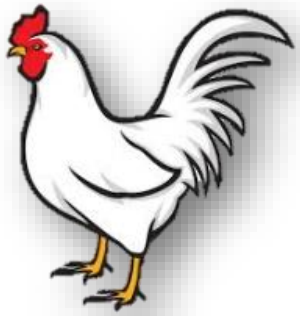
- Students are required to propose the design for a small-size UAV to navigate in a GPS-denied unknown environment, considering its hardware constrains such as footprint, payload, onboard computation resources, power consumption and communication bandwidth. The specification requirements of the UAV and the environment will be provided. A technical report is expected to explain the design decisions among various choices. It should include literature review, conceptual ideas, solution/choice comparisons and discussions.

What is a UAV or Drone?

An unmanned aerial vehicle (UAV), also known as drone, is an aircraft that is equipped with necessary sensors, data processing units, control and navigation software, communications systems and can perform automated flight missions without the intervention of a human pilot.

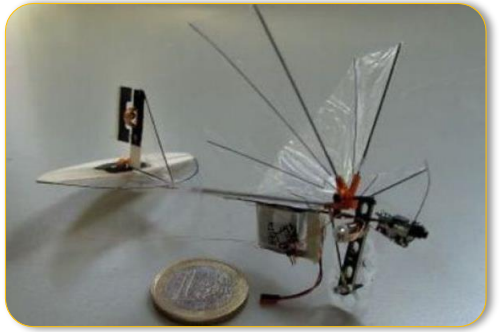


Or simply an aircraft without human on-board.



A drone?

Types of UAVs



and Their Applications…

NUSUAV

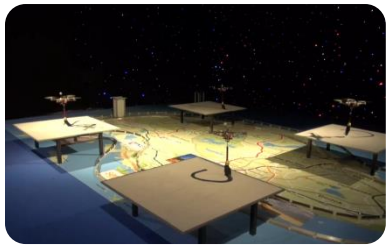
NUS National University of Singapore

UAV
Autonomous Control

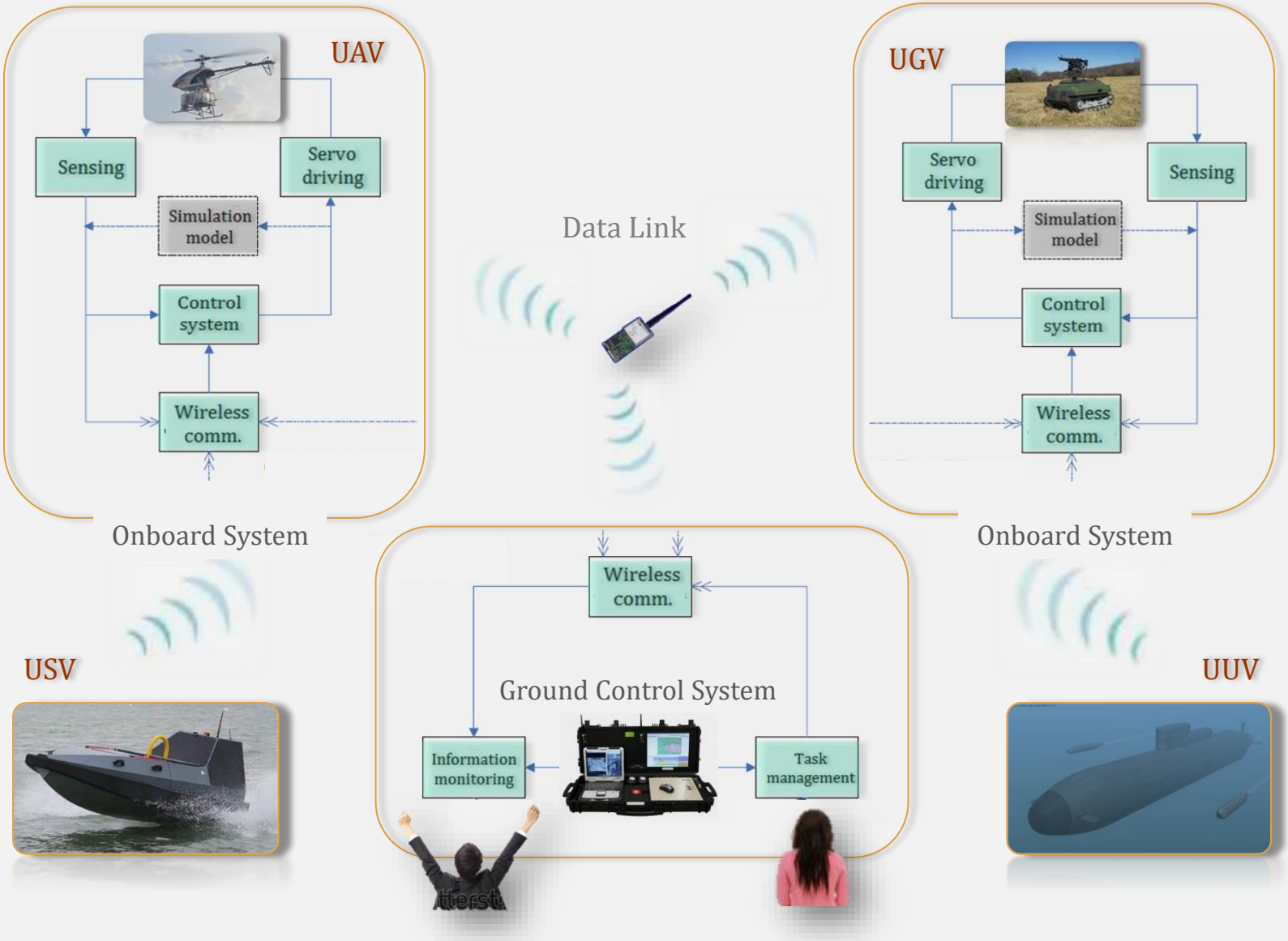
Indoor & Foliage
GPS-less Navigation

Platform
Miniaturization

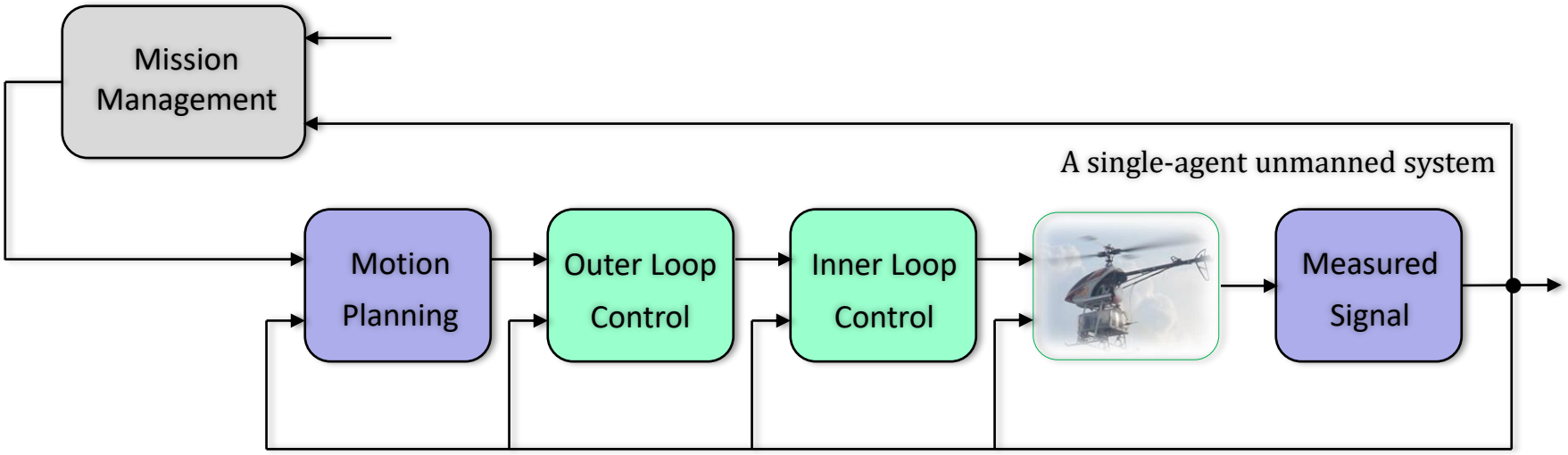
Multi-UAV
Collaborative Control



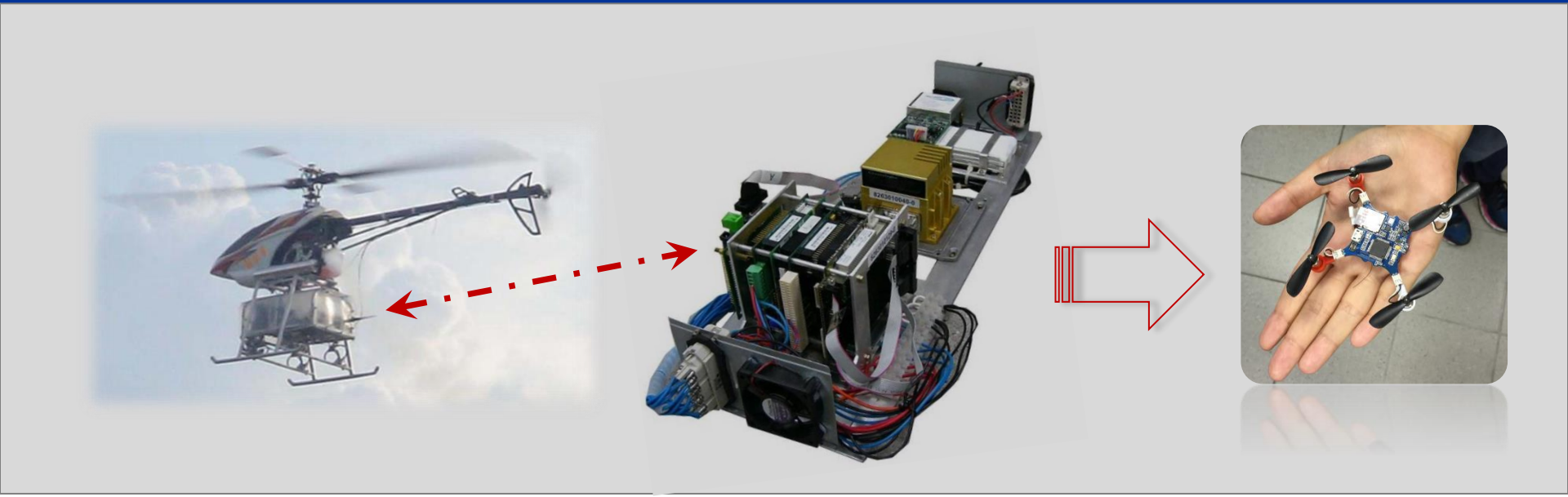
Overall Architecture of Unmanned Systems

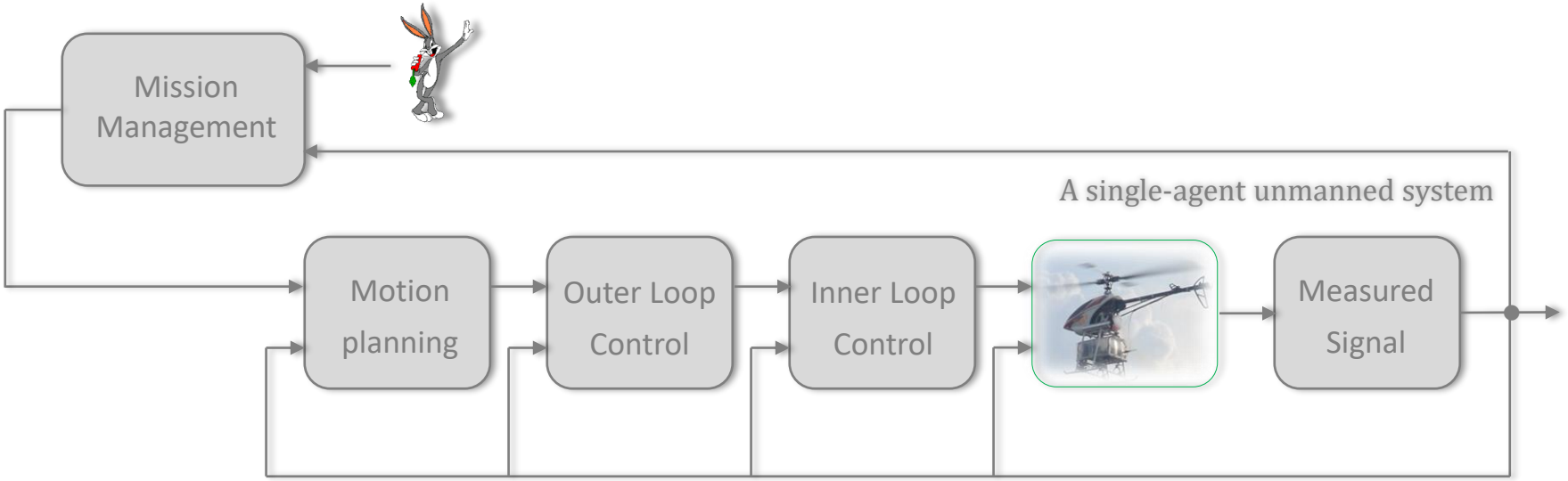


Core Structure of an Unmanned System



Core Subsystem: Avionic Systems

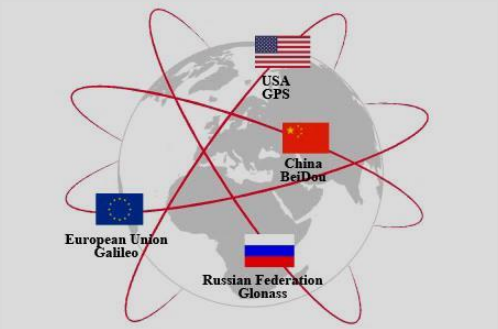




Core Subsystem: Positioning System & Sensors

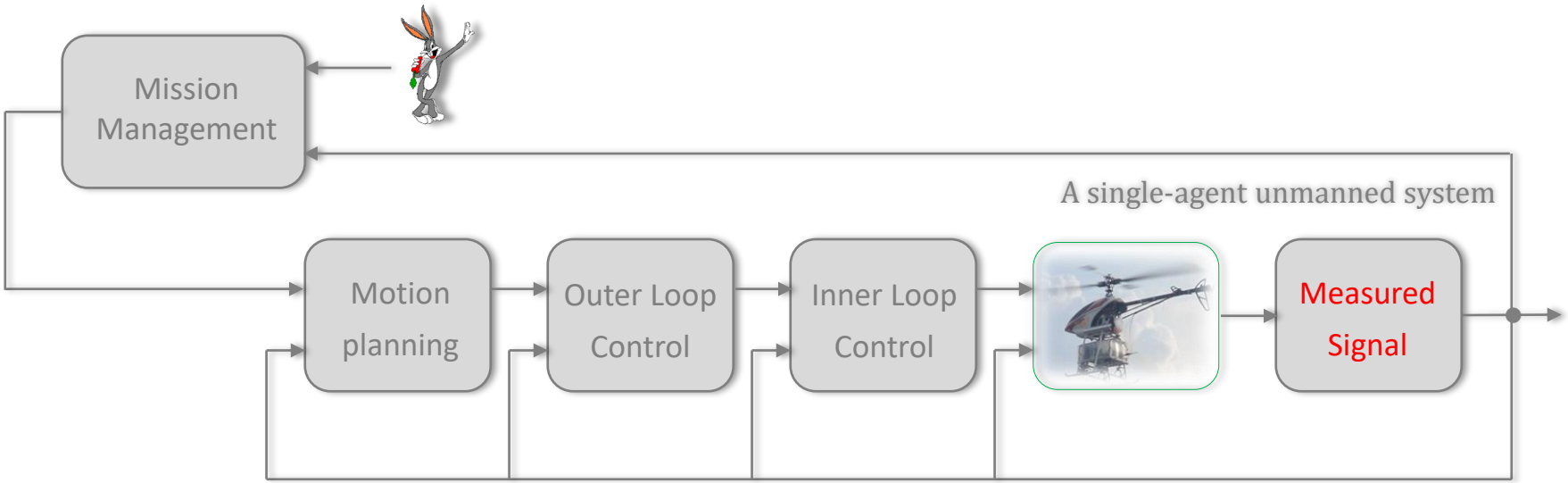
External Systems

- GPS
- Beidou
- VICON
- UWB



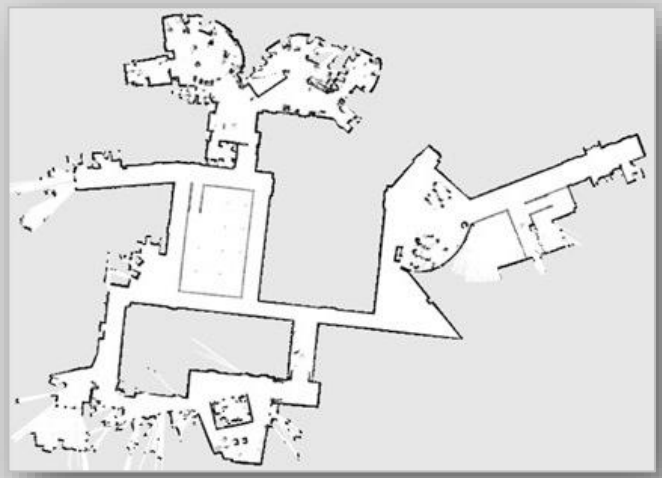
Internal Systems

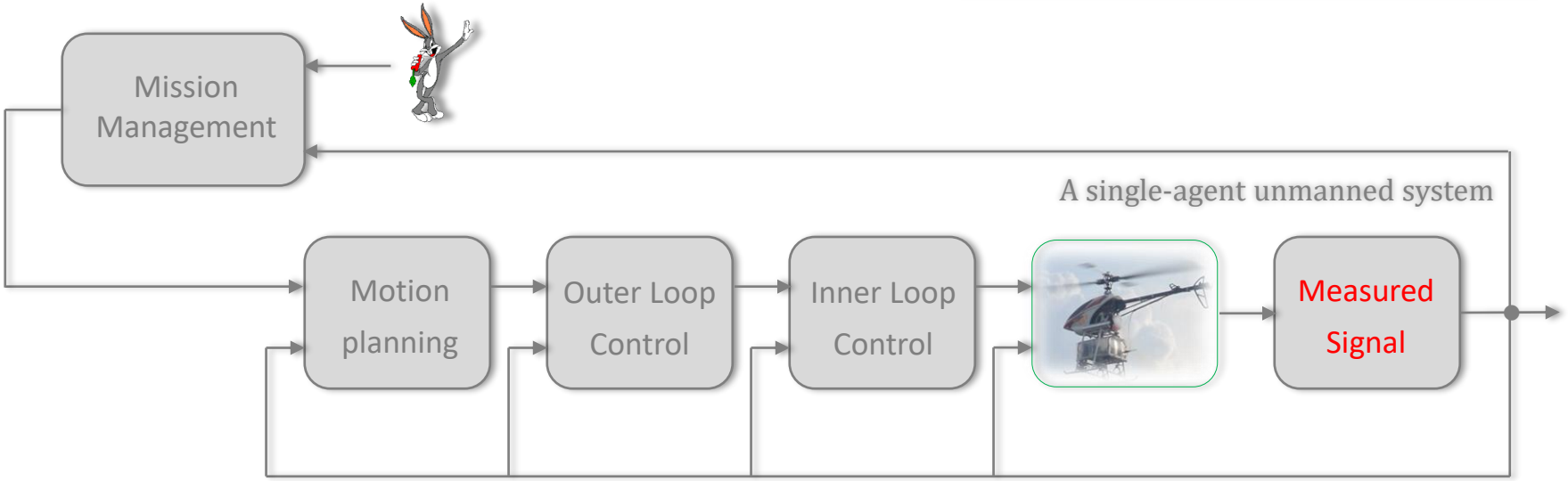
- Vision
- LiDAR
- Radar
- Sonar



Key Technique: Positioning System Algorithms

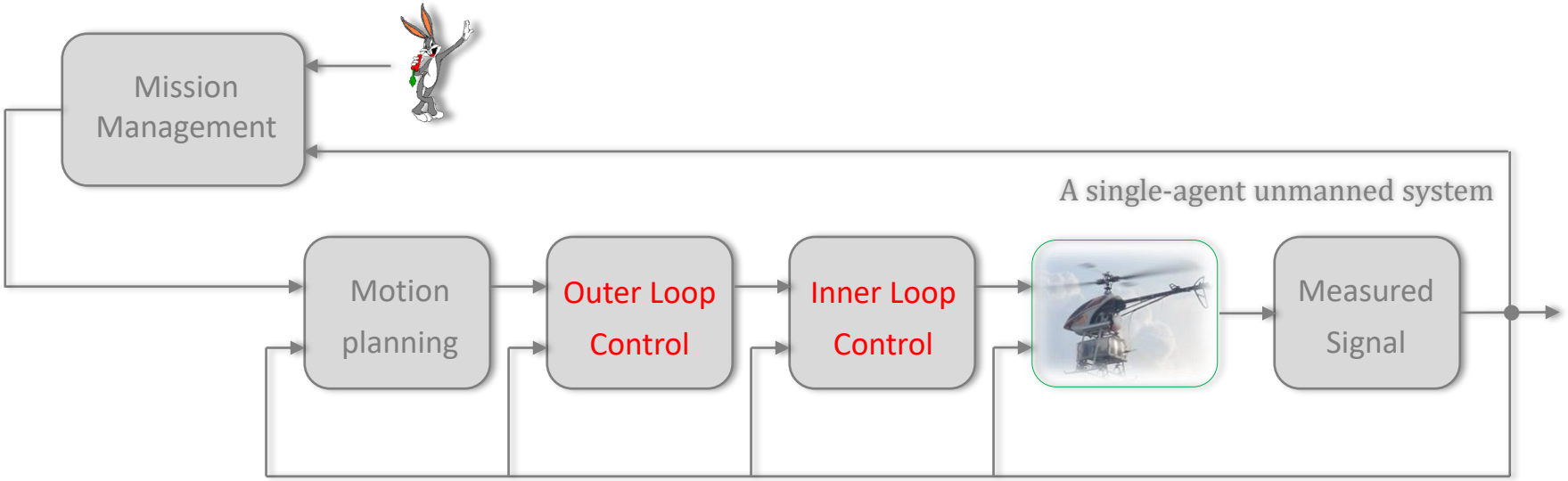
S
L
A
M



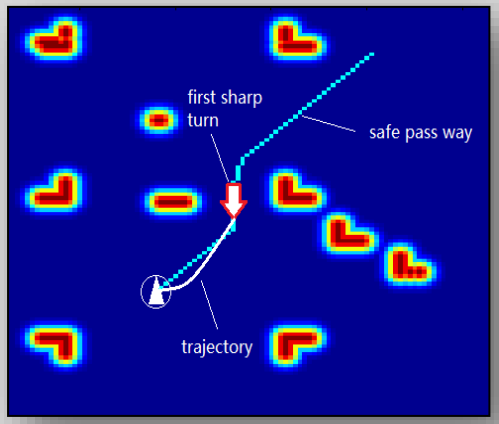


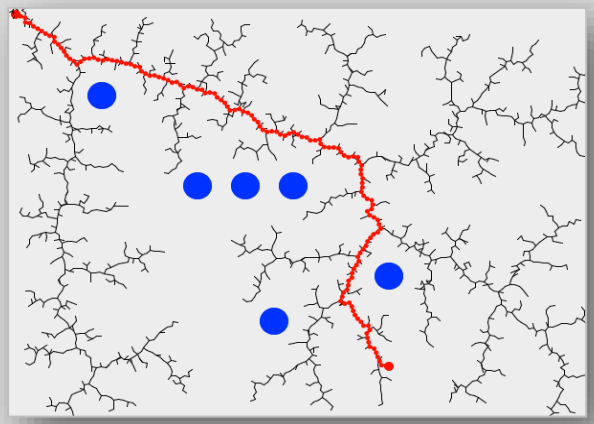
Core Subsystem: Control Systems

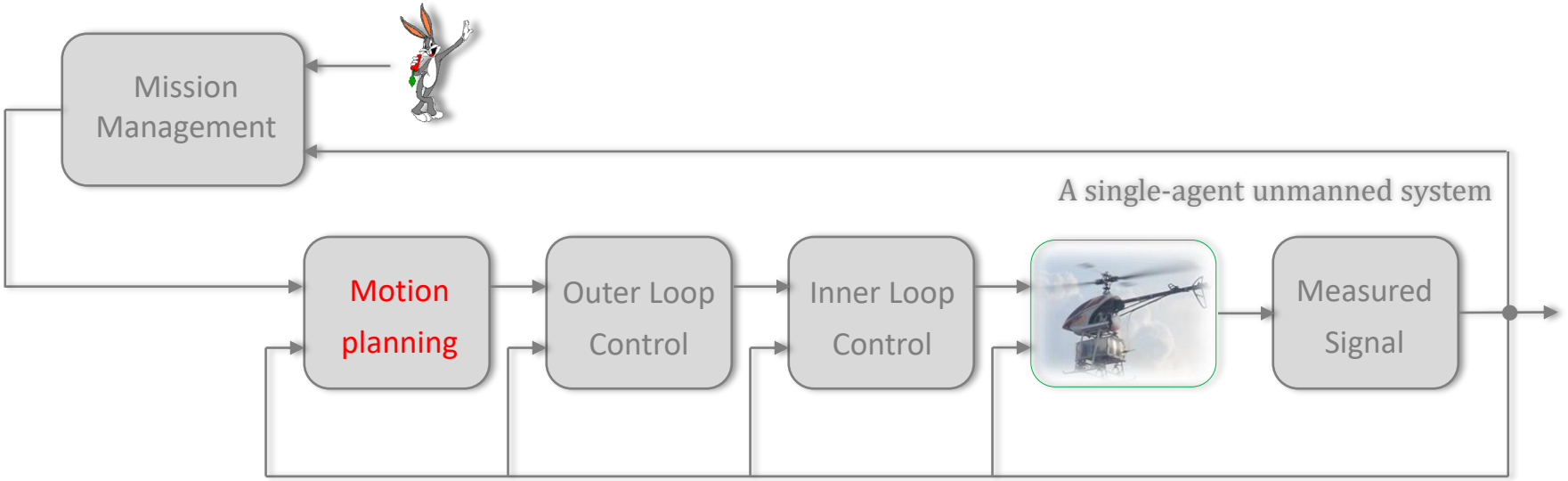
- Inner Loop control
  - PID Control (commonly used)
  - Optimal Control
  - Robust Control
  - Nonlinear Control
- Outer Loop control
  - PID Control (commonly used)
  - Pole placement
  - RPT Control
  - Robust Control



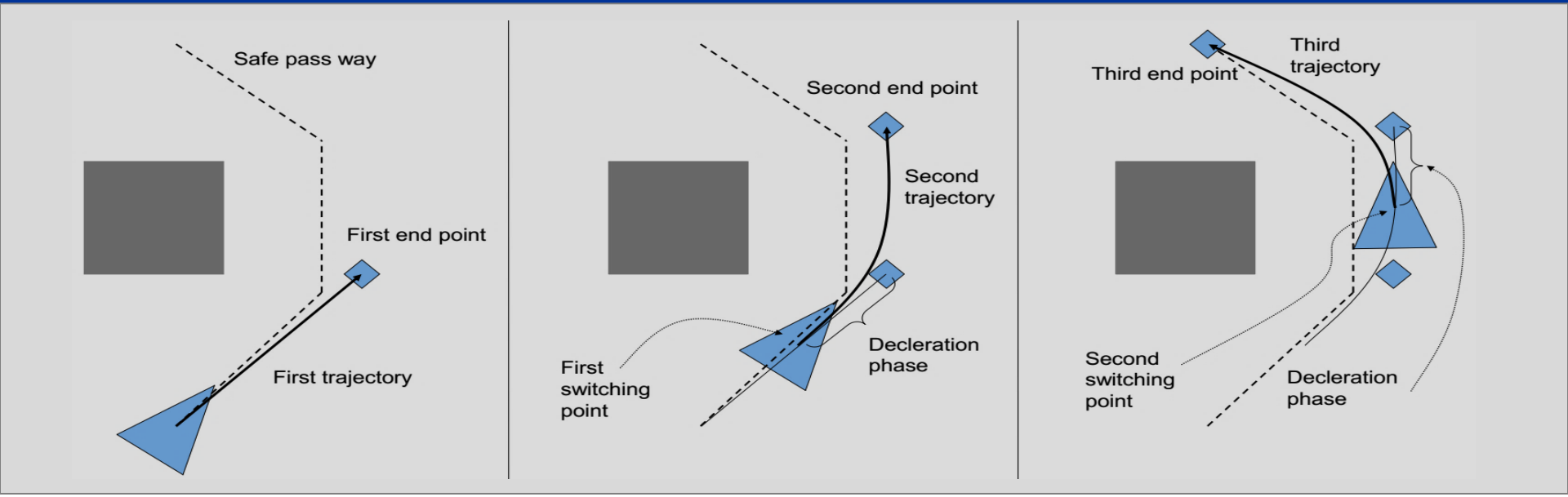
Core Subsystem: Motion Planning

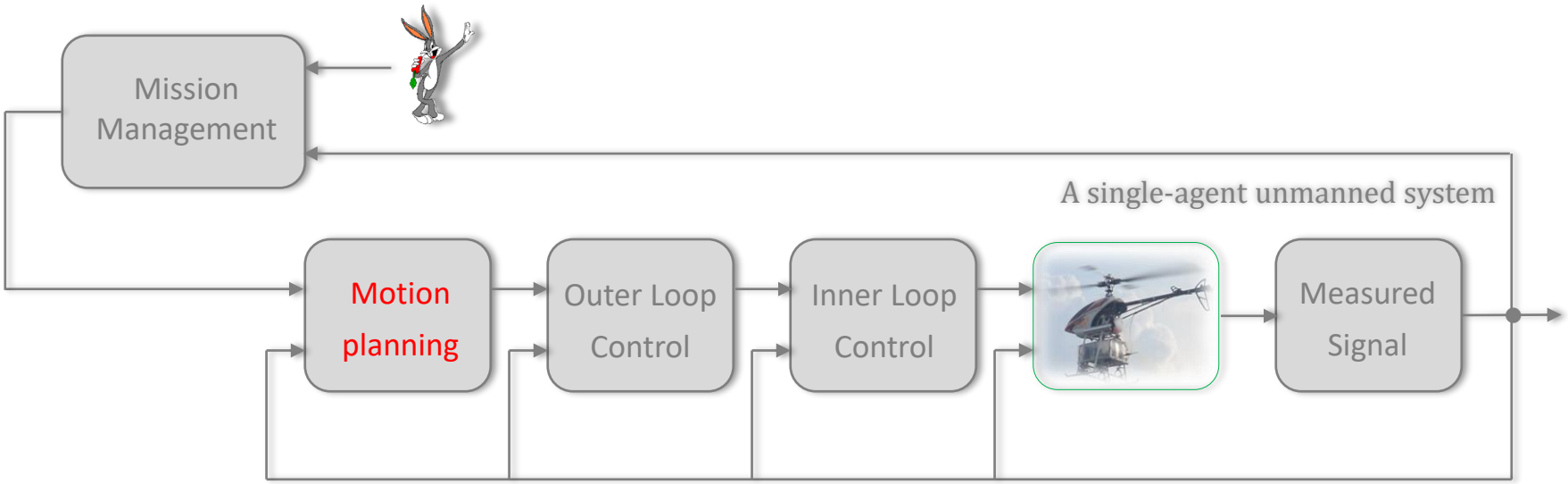






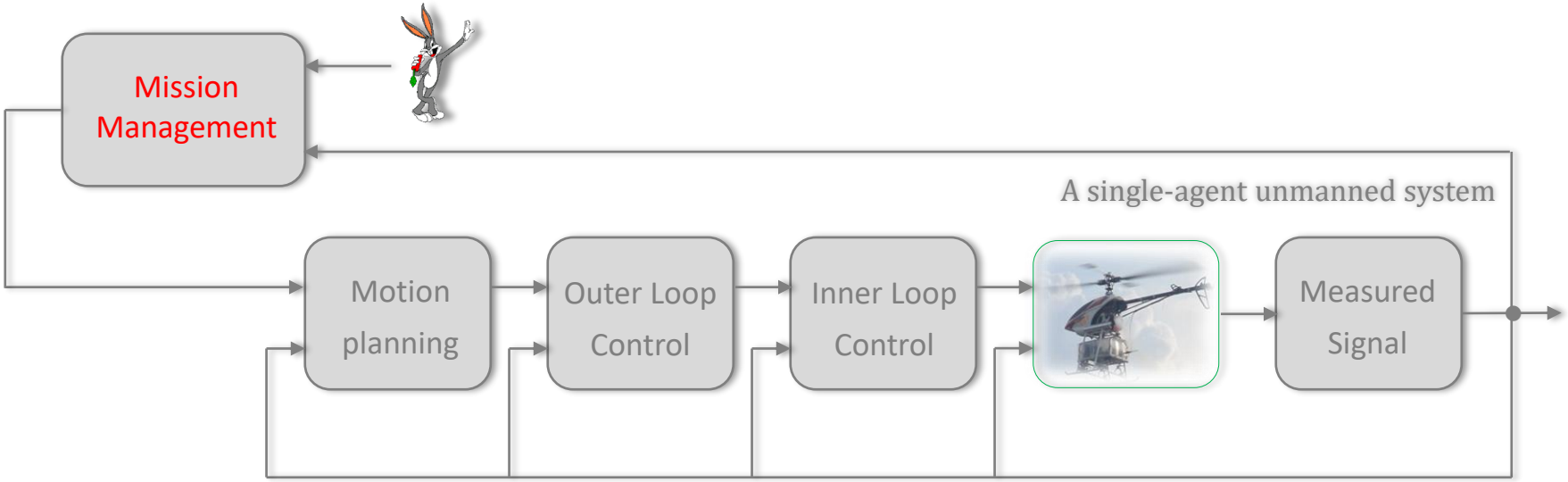
Core Subsystem: Motion Planning



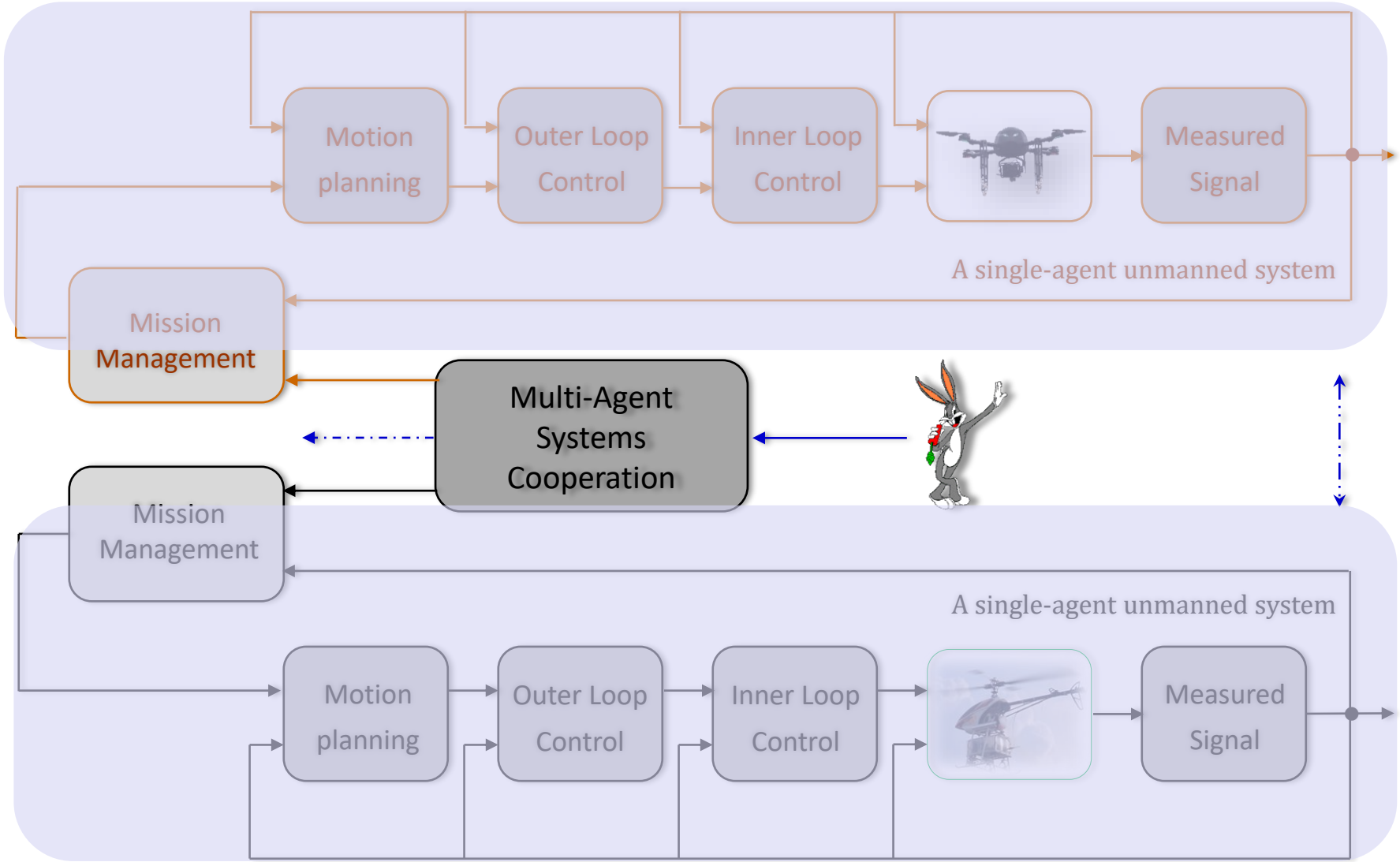


Core Subsystem: Mission Management

- State Machine
- Automata
- Deep learning? ……



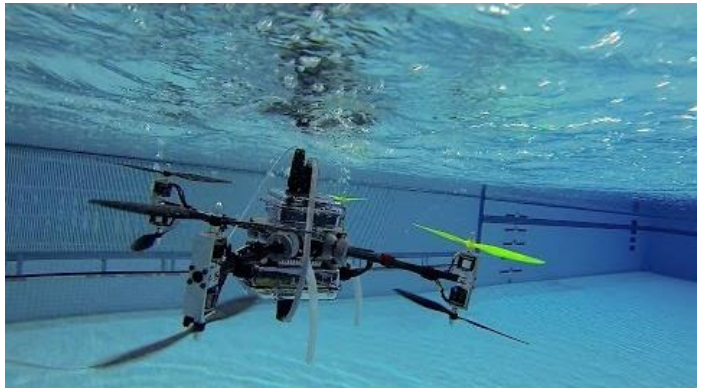
Cooperation of Multi-agent Unmanned Systems



Interesting Research Topics

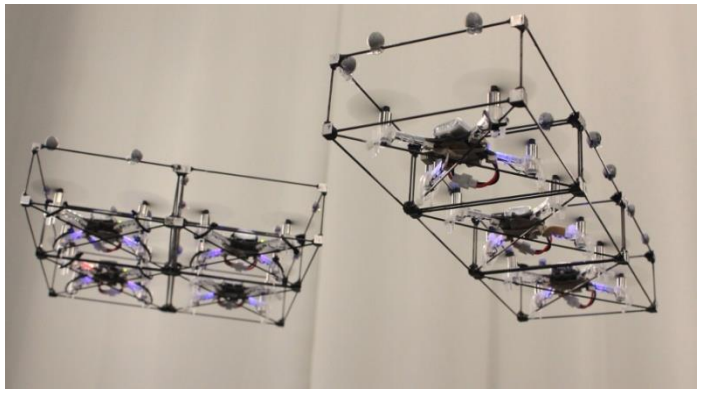
- Unconventional platform design and modeling

Hybrid UAVs



Amphibious UAVs

Bladeless / tough UAVs



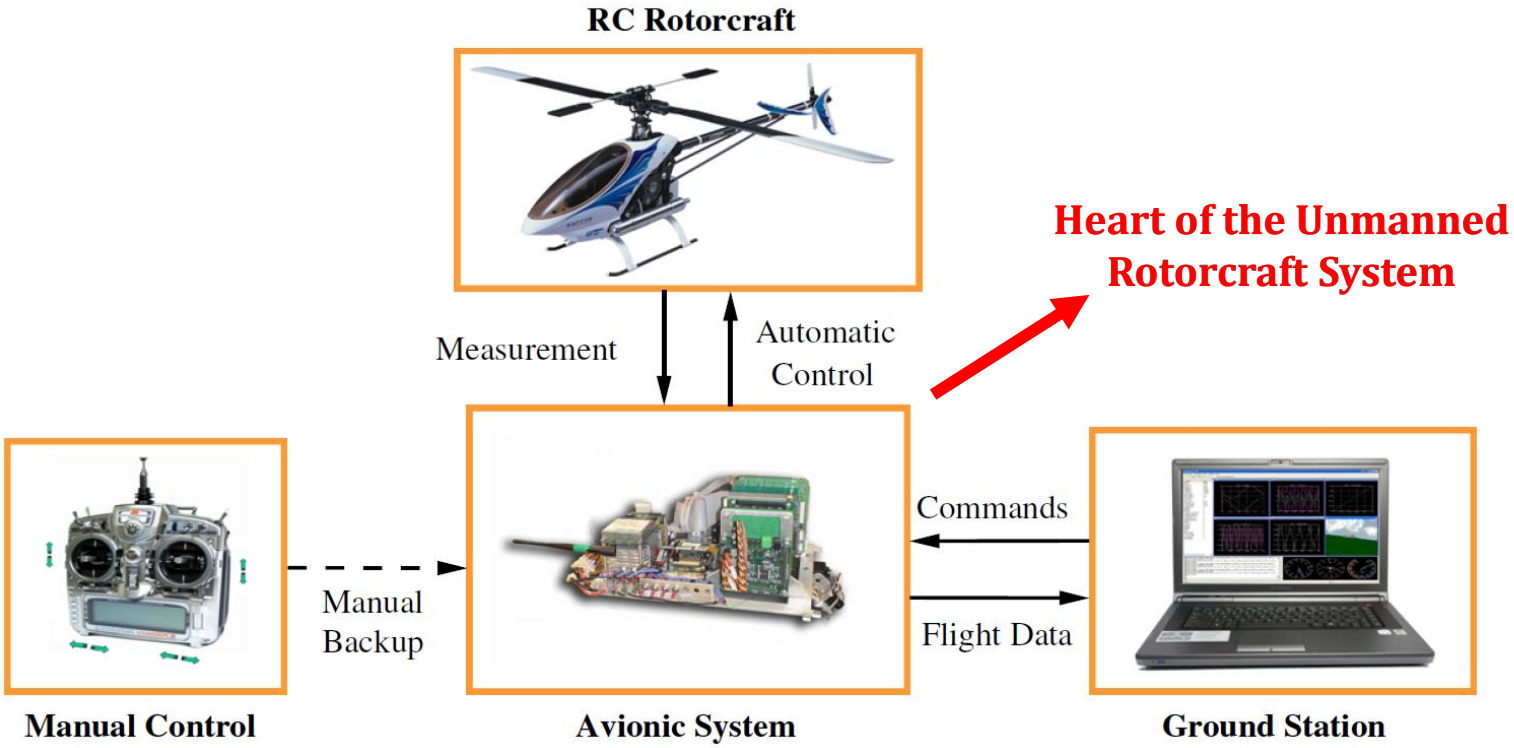
Self-assembled vehicles

Interesting Research Topics

- Control
  - **Model free** control
  - **Model predictive** control with trajectory library
- Task & Motion Planning
  - Planning with **model uncertainty**
  - Sampling based planning in **high dimensional space**
  - Planning in **belief space**
  - **Integrated** task and motion planning

Key Components of Unmanned Rotorcraft System

1. A radio-controlled (RC) rotorcraft
2. An avionic system for collecting inflight data, performing automatic control laws, executing mission-oriented tasks, and communicating with the ground station
3. A manual control system consisting of a pilot and a wireless joystick
4. A ground station system for monitoring the flight states of the UAV and communicating with the avionic system



UAV Platform Design and Construction

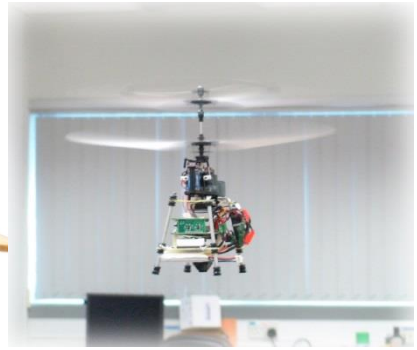
- A reliable unmanned aerial platform is the foundation of the subsequent work (e.g., flight dynamics modeling, control system design, autonomy)

- Essential questions in platform design and construction:
  - What type of aircraft platform to choose?
  - What avionics components should be included in the design?
  - How to achieve a reliable assembly?
  - How to minimize construction time and unnecessary iterations?
  - How to verify the reliability of a constructed UAV platform?

Bare Aircraft Selection

How to choose a platform?

Multi rotor



Coaxial helicopter

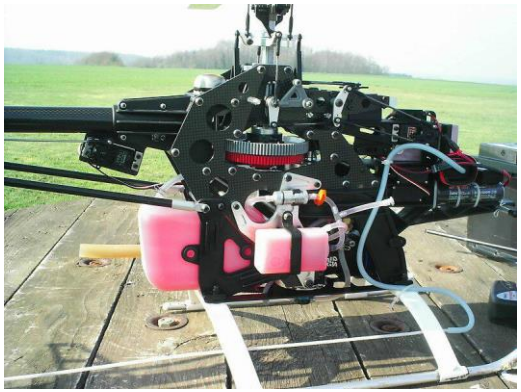
- Size
- Weight
- Max payload
- Max flight time
- Max speed
- Cost
  - Availability
  - Manufacturability
  - Maintainability

Single-rotor helicopter

Bare Aircraft Selection

Bare Helicopter: Raptor 90

Length: 1410 mm
Width: 190 mm
Height: 465 mm
Main rotor diameter: 1580 mm
Tail rotor diameter: 260 mm
Gear ratio: 1 : 8.45 : 4.65
Total weight: 4.8 kg



Bare Aircraft Selection

- UAV Hardware System
  - » Quad-rotor
  - » Avionics
    - IMU + GPS
    - Camera and LDIAR (optional)
    - Dual processors: flight control + navigation and mission

Platform (kg)	Batteries (kg)	Extra Payload (kg)	Total Weight (kg)	Size (m)	Endurance (min)
1.5	2	2	5.5	1.1 (tip-to-tip)	23
1.5	2	1.5	5.0	1.1 (tip-to-tip)	27

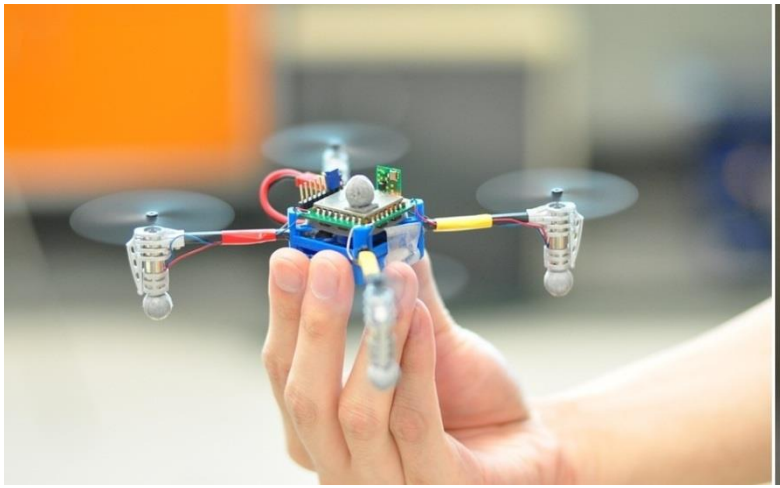
- UAV Software System
  - » Flight control system: ARM processor for real-time sensor reading, flight control, servo driving and wireless communication
  - » Navigation and mission system: Intel x86 processor for image and lidar data capture, localization, mapping and path planning

Bare Aircraft Selection

Micro aerial vehicle — K-Lion…

Specifications

- 40 grams including battery
- 8 minutes flight endurance
- Fully autonomous
- Orientation control 200 Hz
- Position control 50 Hz



Unconventional Aircraft – Hybrid Platforms

Gimballed Vector Thrust

- Roll and pitch controlled by gimbal system
- Yaw controlled by rotational speed difference

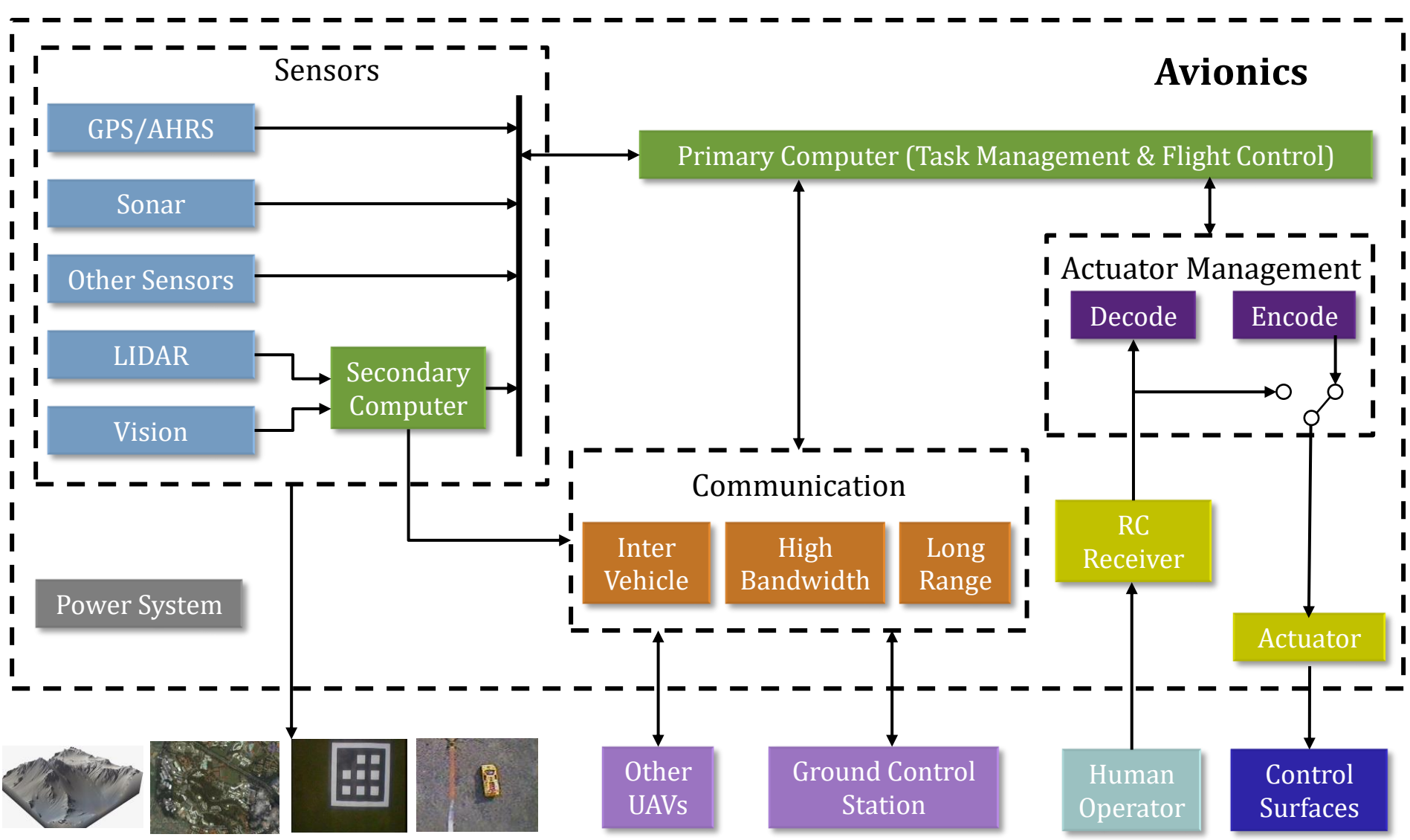
Retractable Wings

- 3 control modes available
- Enable VTOL, hovering, cruise flight

Gyro Stabilizer

- 5 gyros to stabilize 3-axis orientation
- Works on both hovering and cruising modes

Common Avionics Components



Construction of Avionic System… Introduction

- The design process consists of four main steps:

  1) Virtual Design Environment Selection

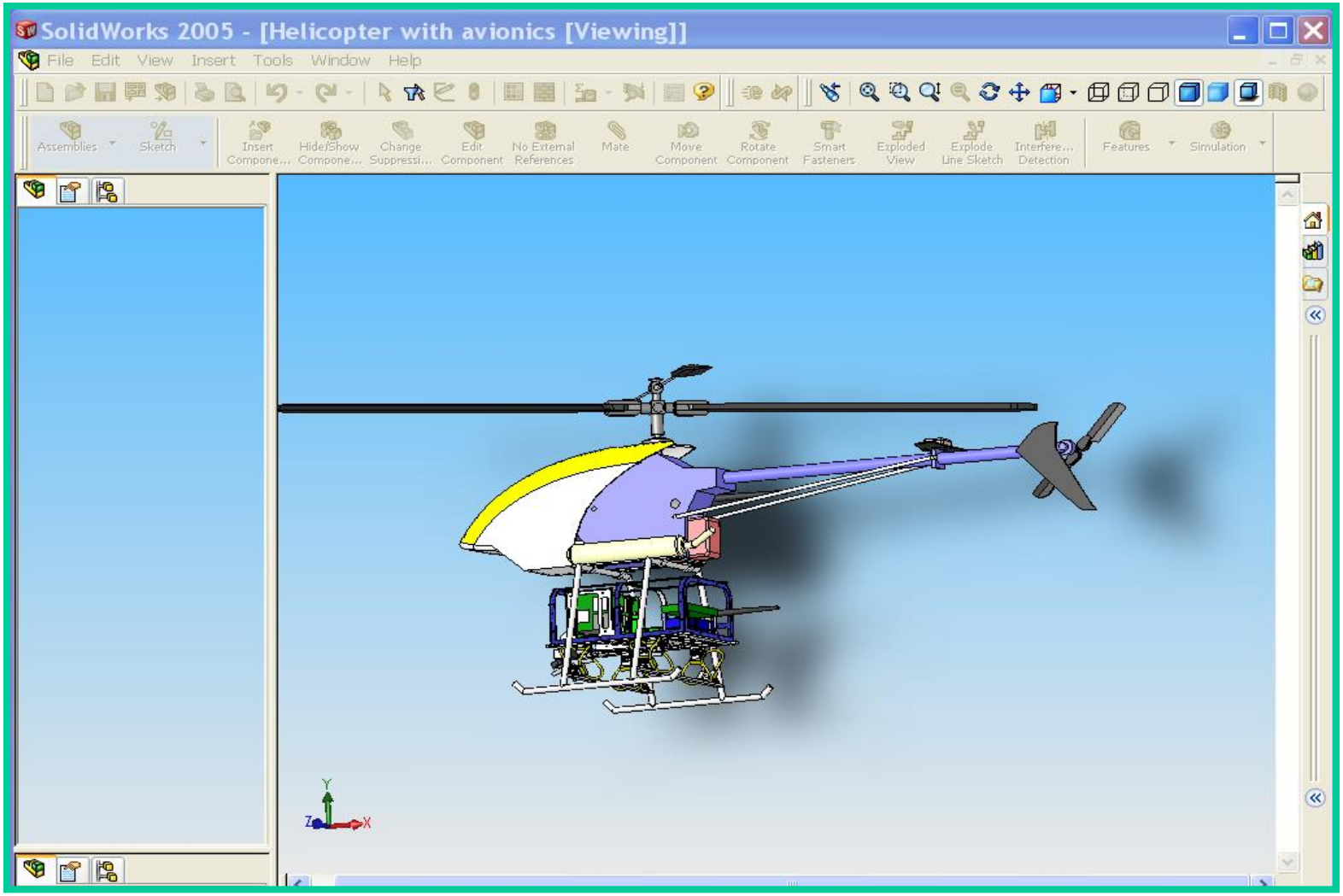
  2) Hardware Components Selection

  3) Layout Design and Integration

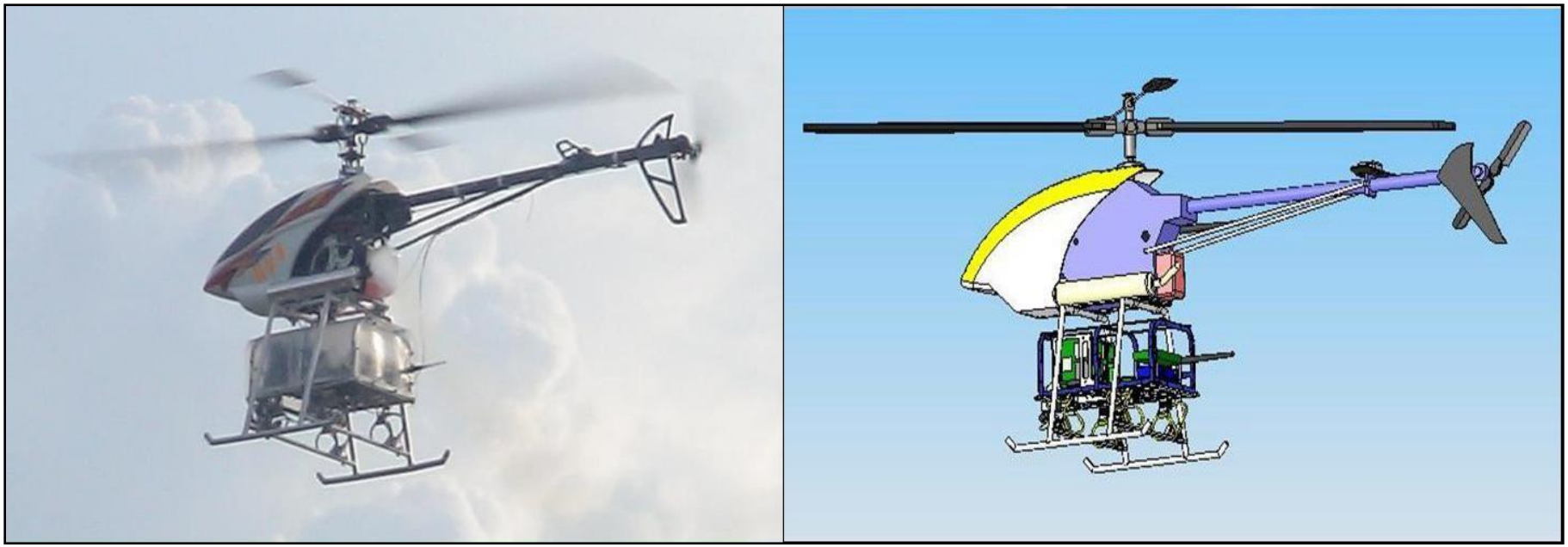
  4) Reliability Evaluation

- We use HeLion, a single-rotor helicopter type of UAV, from NUSUAV to illustrate the concepts

Step 1: Virtual design environment (VDE) Selection: SolidWorks



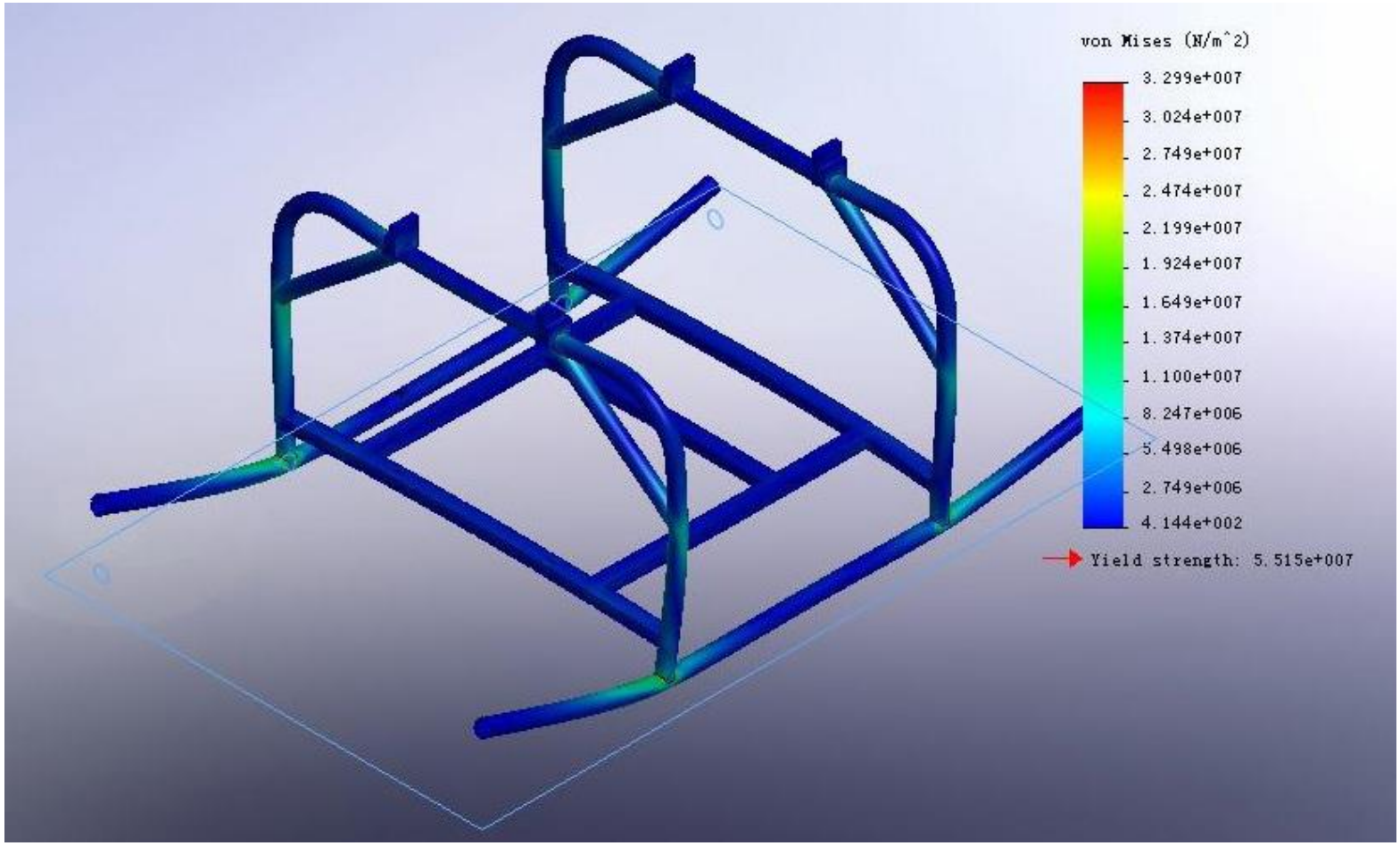
Step 1: Virtual design environment (VDE) Selection: SolidWorks



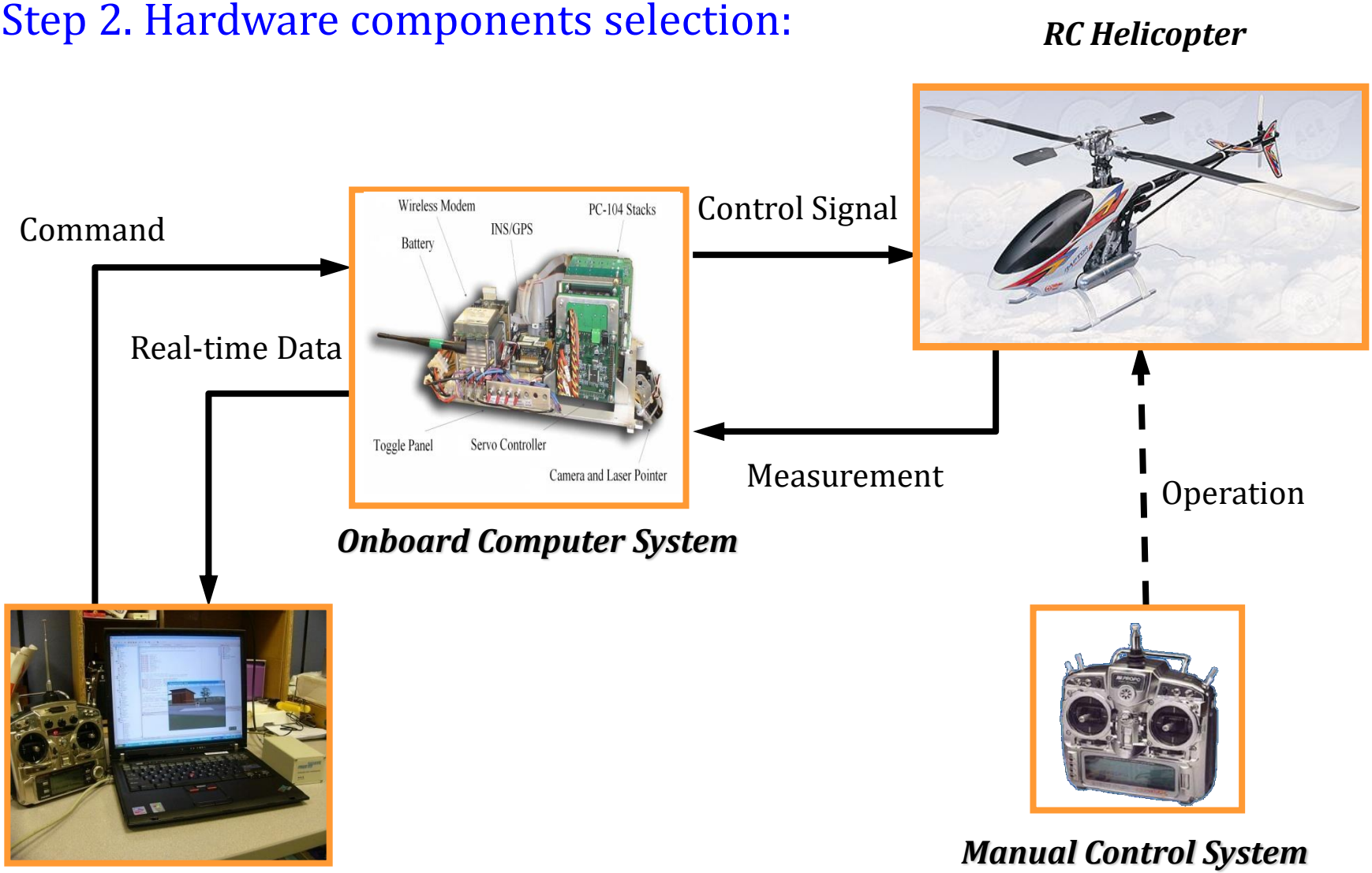
HeLion and its virtual counterpart built in SolidWorks

Construction of Avionic System… VDE Selection

Analysis of structural properties

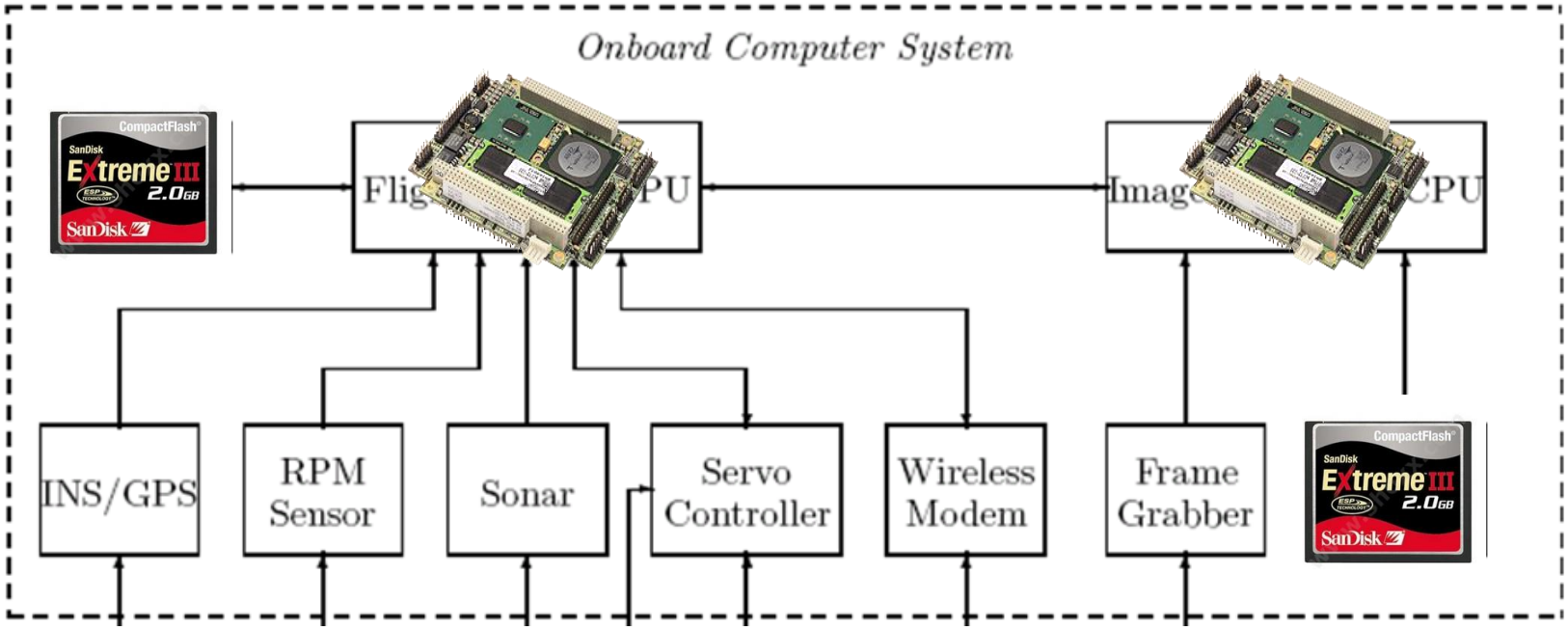


Step 2. Hardware components selection:



Construction of Avionic System… Components Selection

Step 2. Hardware components selection:



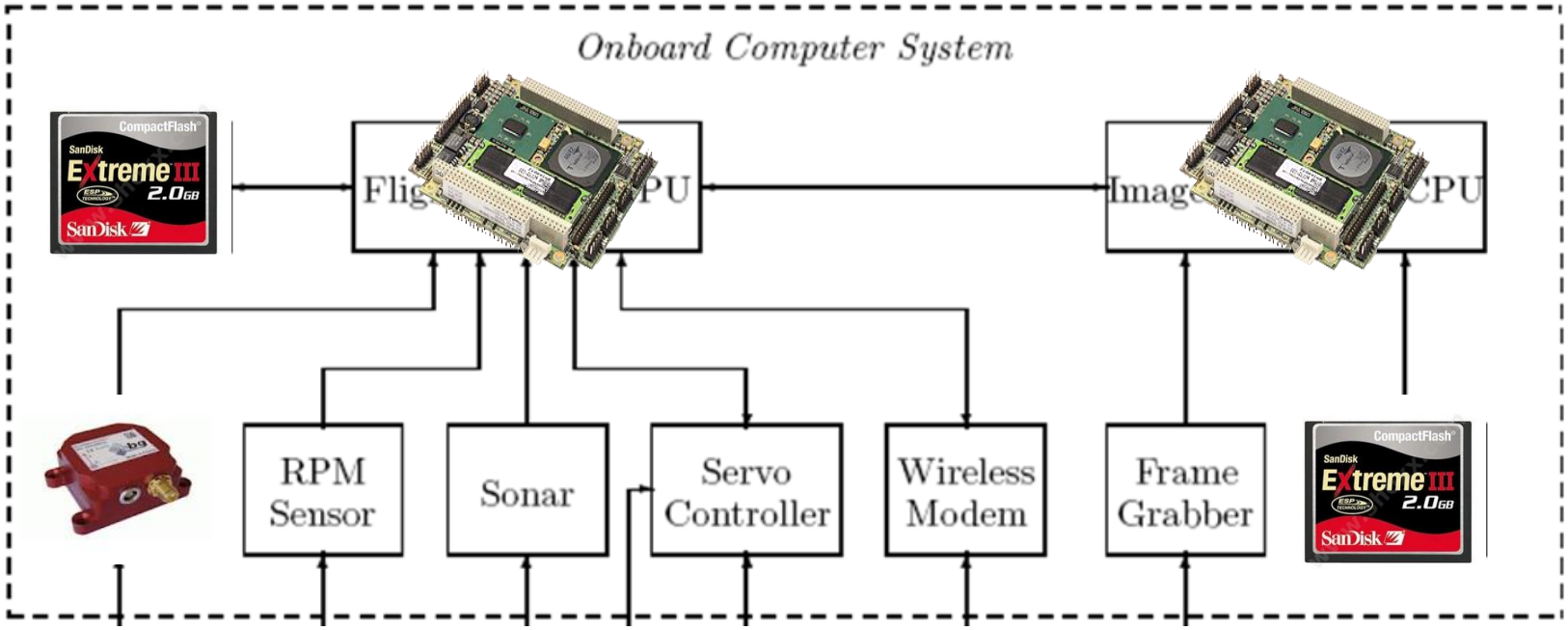
PC/104(-plus) SBC

Reason for selection:

1. Industry/military level reliability
2. Sufficient computational power
3. Small and standard size
4. Good expandability
5. Driver support for real-time OS

Construction of Avionic System…
Components Selection

Step 2. Hardware components selection:



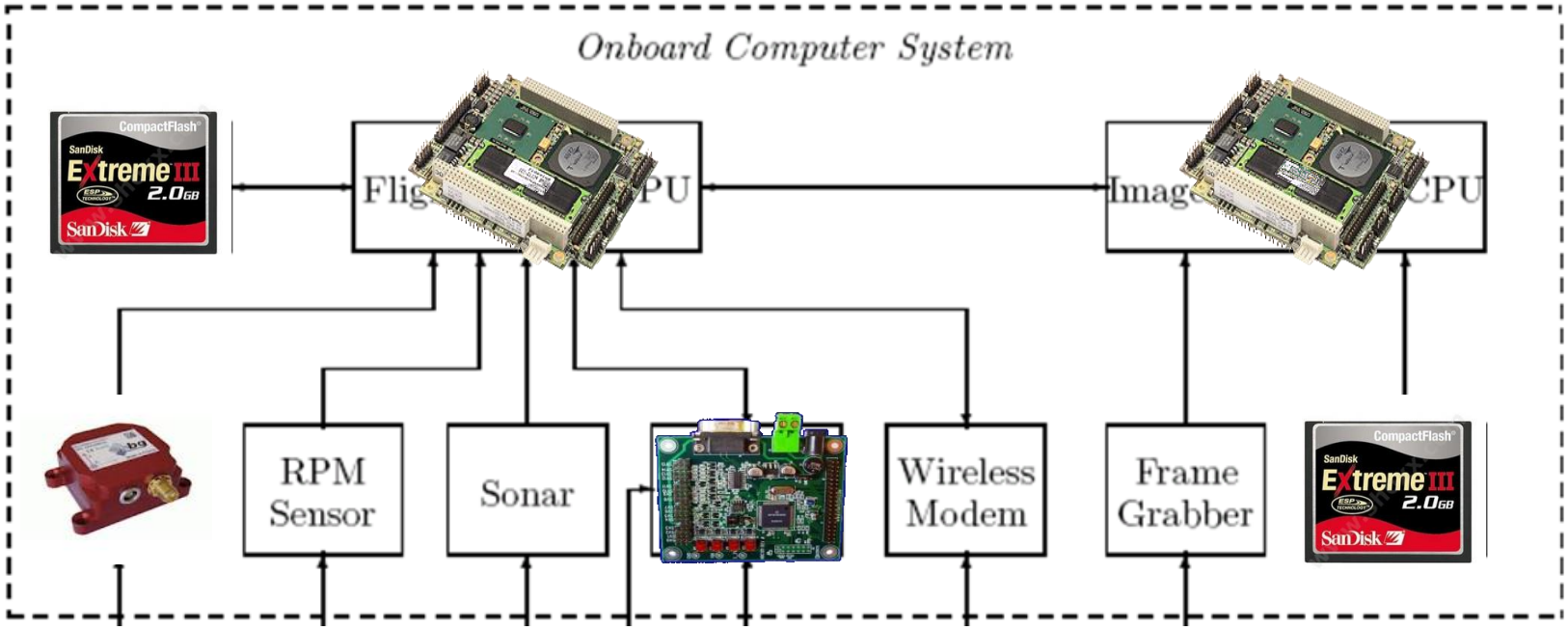
IG-500N INS/GPS

Reason for selection:

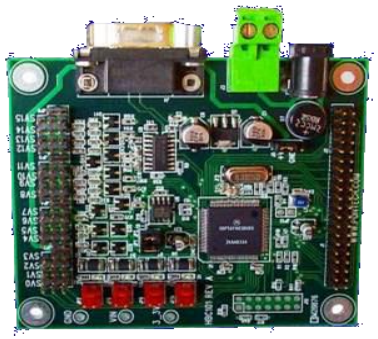
1. Complete data package for modeling and control
2. Precise calibration for raw sensors
3. Built-in Extended Kalman Filter
4. Sufficient update rate up to 100 Hz
5. Anti-vibration and EMI design

Construction of Avionic System… Components Selection

Step 2. Hardware components selection:



HBC-101 Servo Ctrl Bd

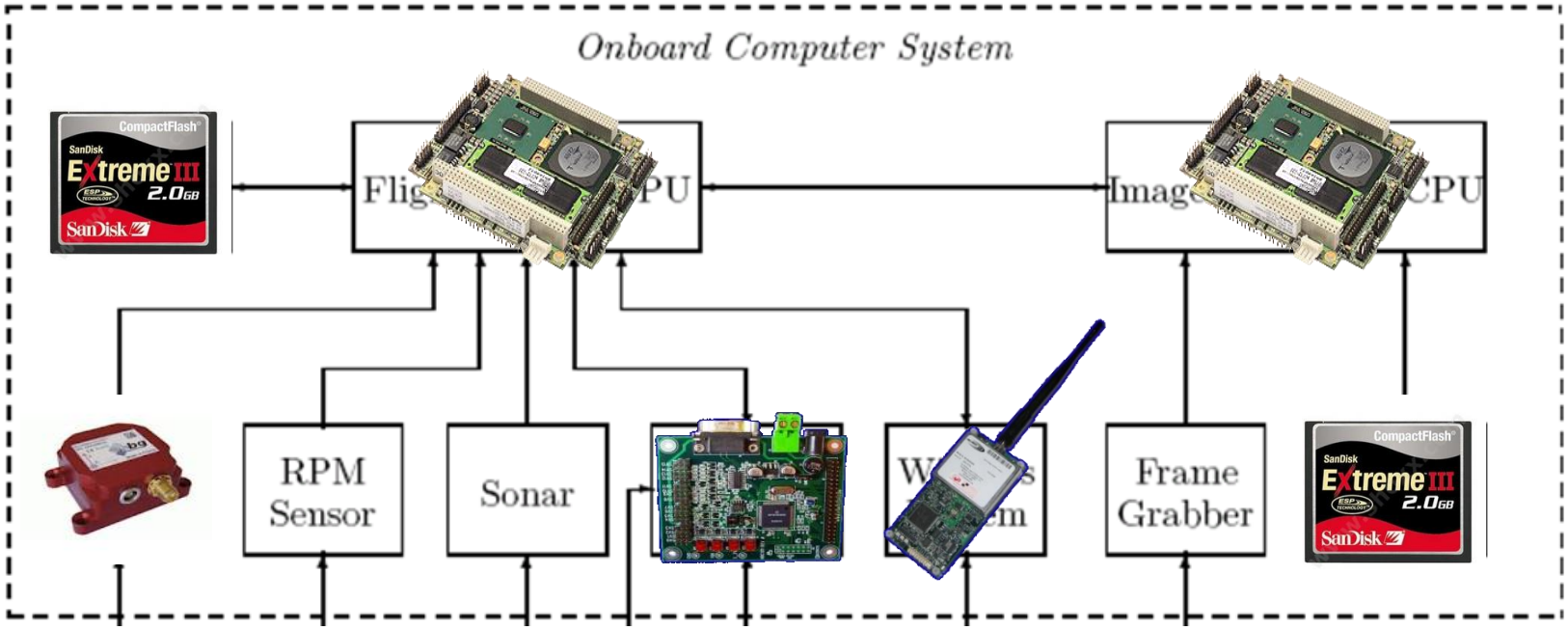


Reason for selection:

1. Input signal recording for modeling and control
2. Reliable manual/auto switching function
3. Sufficient input/output channels for extension
4. High A/D resolution (16-bit)

Construction of Avionic System… Components Selection

Step 2. Hardware components selection:



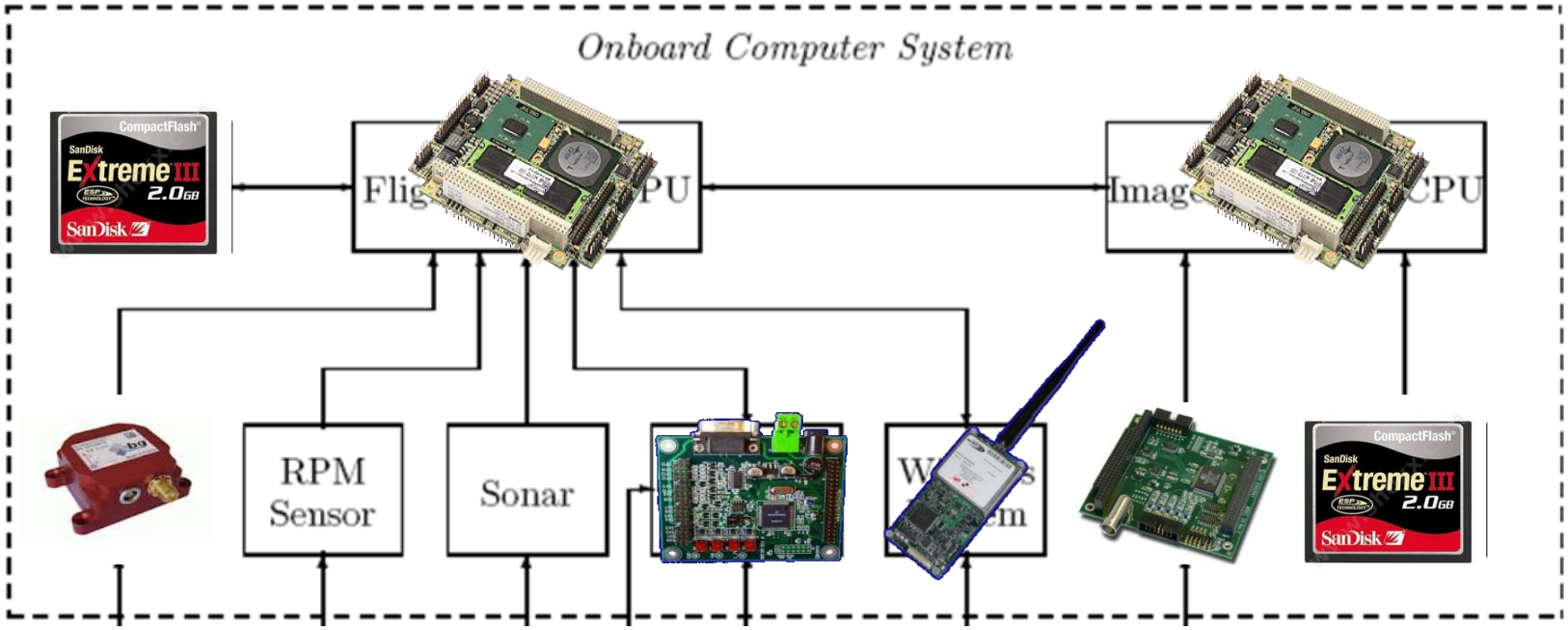
Freewave IM-500 Mdm

Reason for selection:

1. Extremely long range up to 32 km
2. Small size and ultra light weight
3. Plug-in-and-play configuration
4. Sufficient throughput rate

Construction of Avionic System…
Components Selection

Step 2. Hardware components selection:



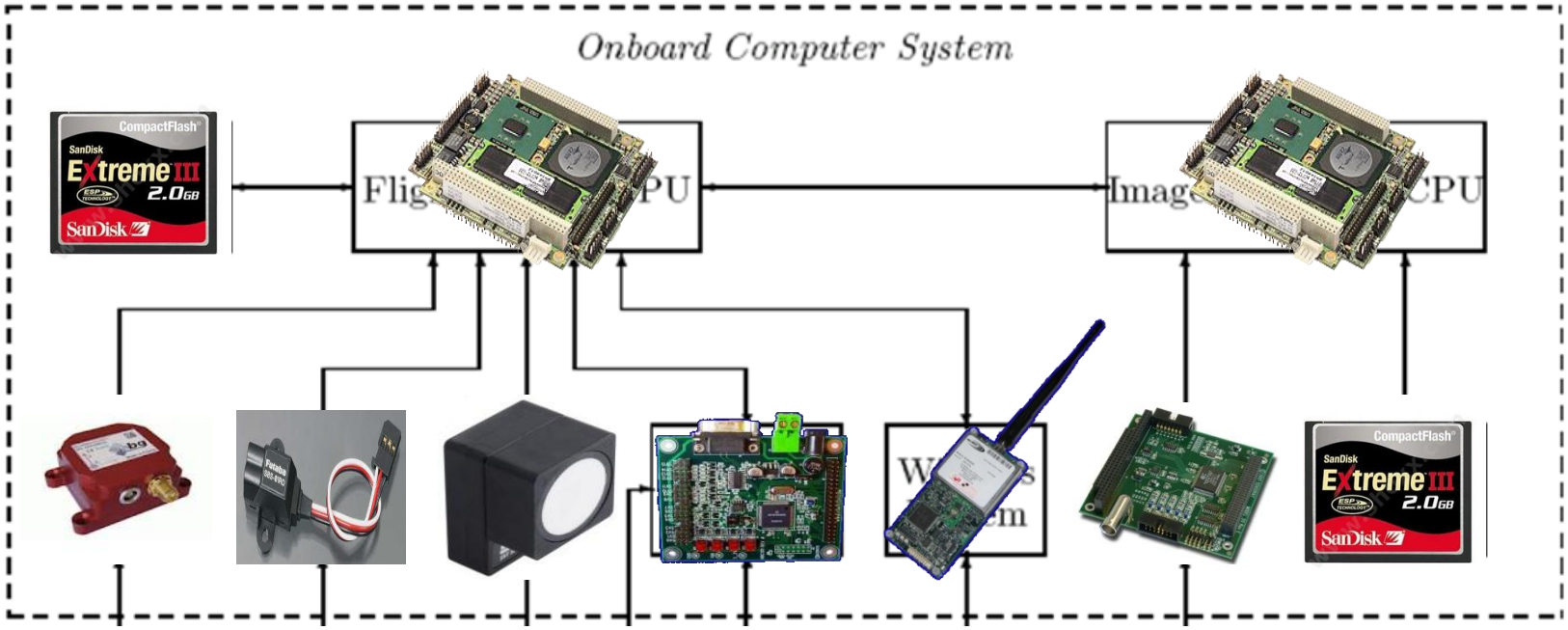
Colory 104 Frame Grab

Reason for selection:

1. Sufficient resolution up to 720*576 pixels
2. Processing rate up to 30 FPS
3. Plug-in-and-play configuration
4. Parallel image processing to 2 formats

Construction of Avionic System…
Components Selection

Step 2. Hardware components selection:



SNT UPK Sonar

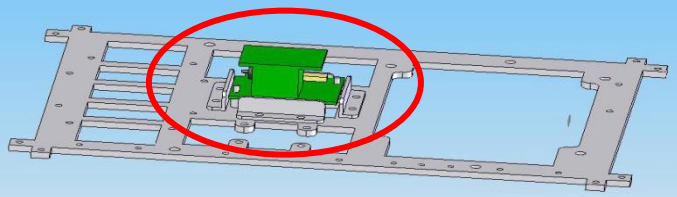
Futaba RPM sensor

Reason for selection:

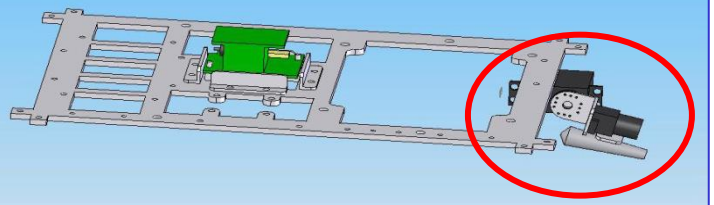
1. Light weight and small size
2. Sufficient precision
3. Easy for customization and mounting

Step 3. Layout Design and Integration:

Determining the location of INS/GPS

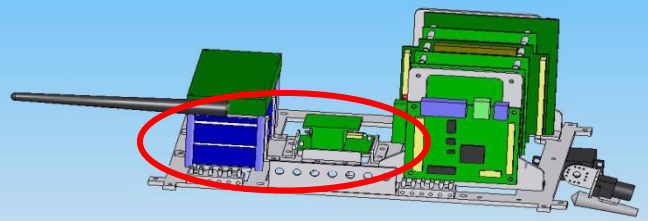


Determining the location of the camera and laser range finder

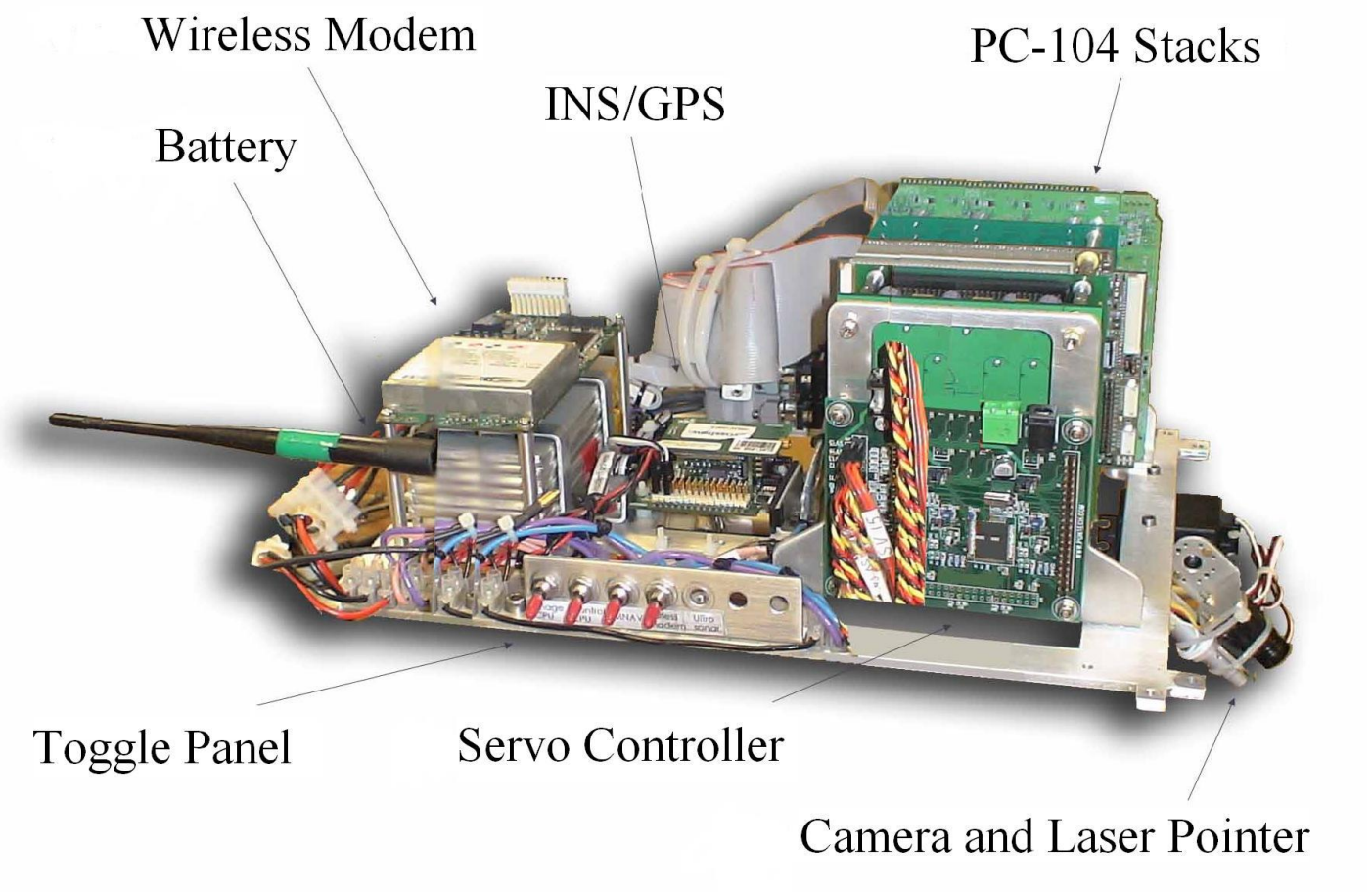


CG balancing

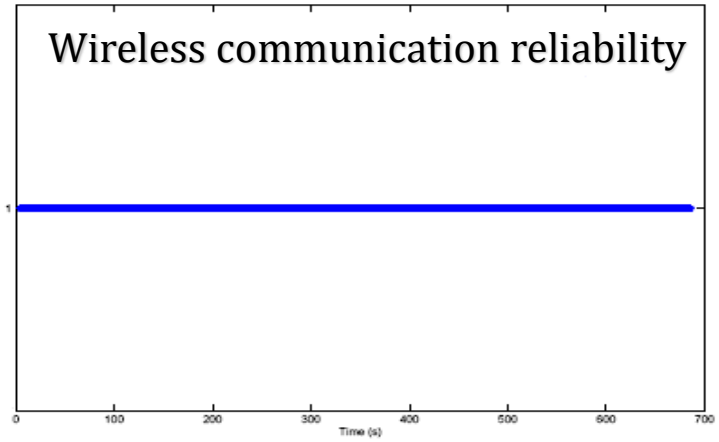
Locating the remaining components

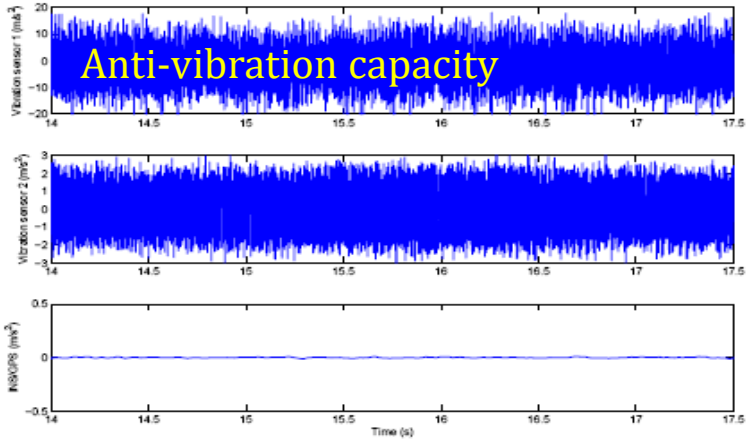


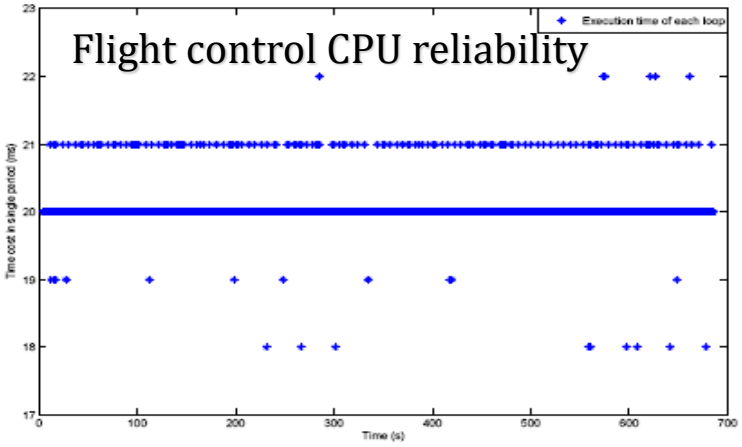
Step 3. Layout Design and Integration:

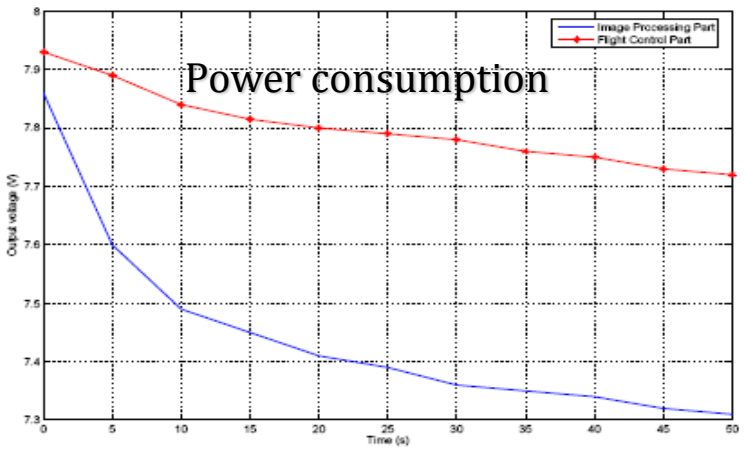


Step 4. Reliability Evaluation:



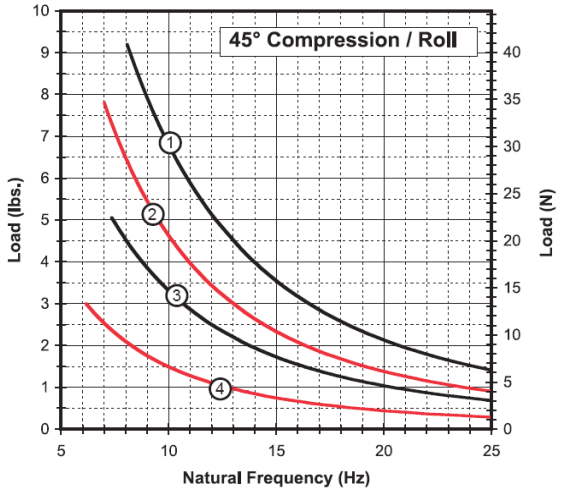


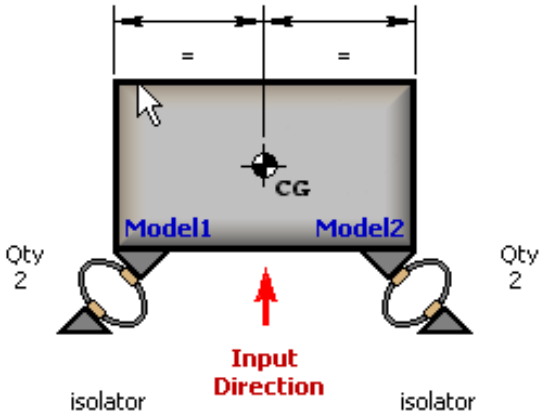


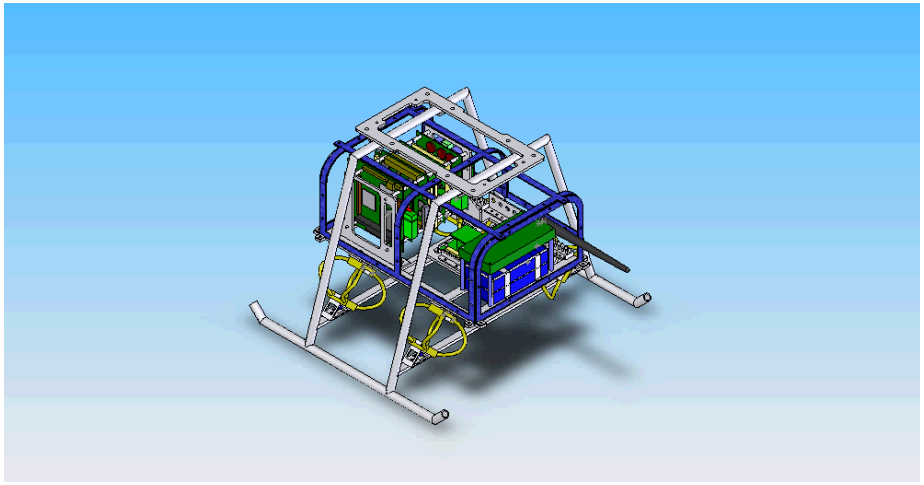


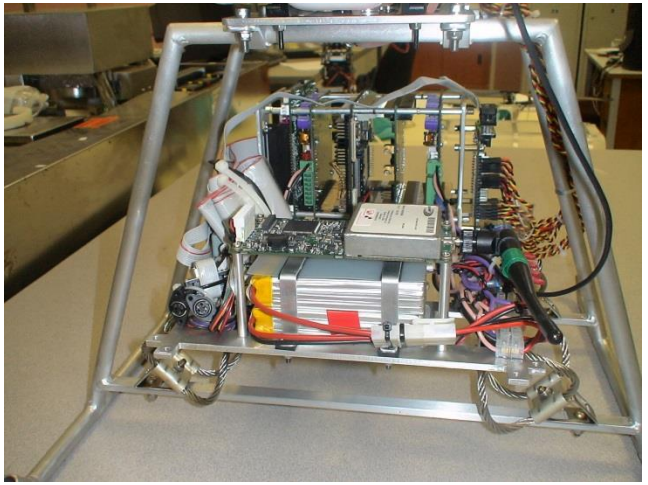
Construction of Avionic System…
Reliability Evaluation

Vibration isolation mount



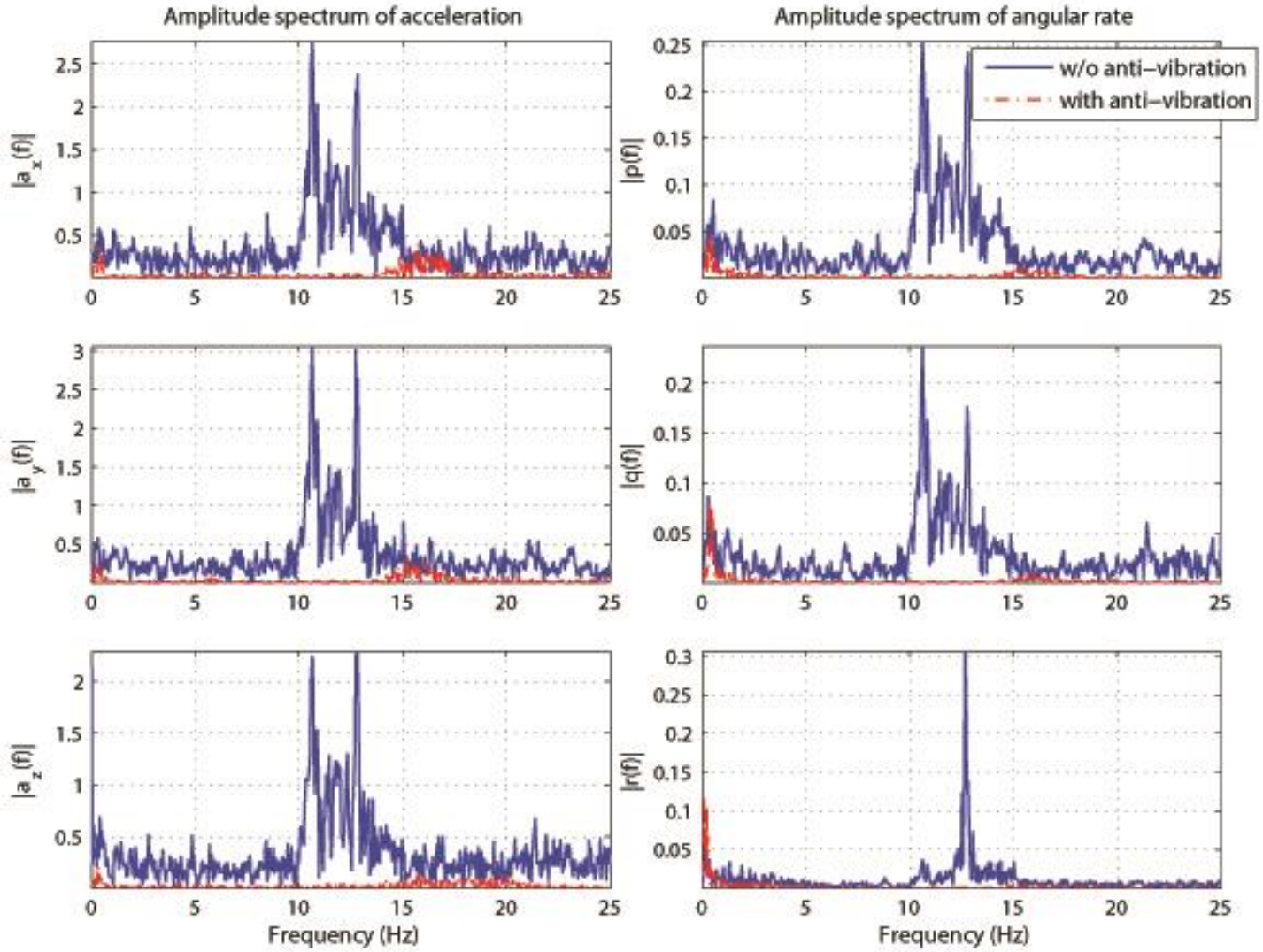






Construction of Avionic System… Reliability Evaluation

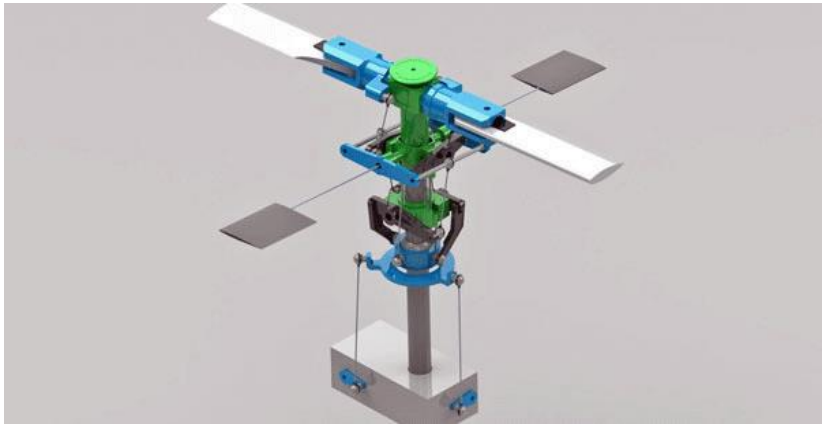
Anti-vibration test

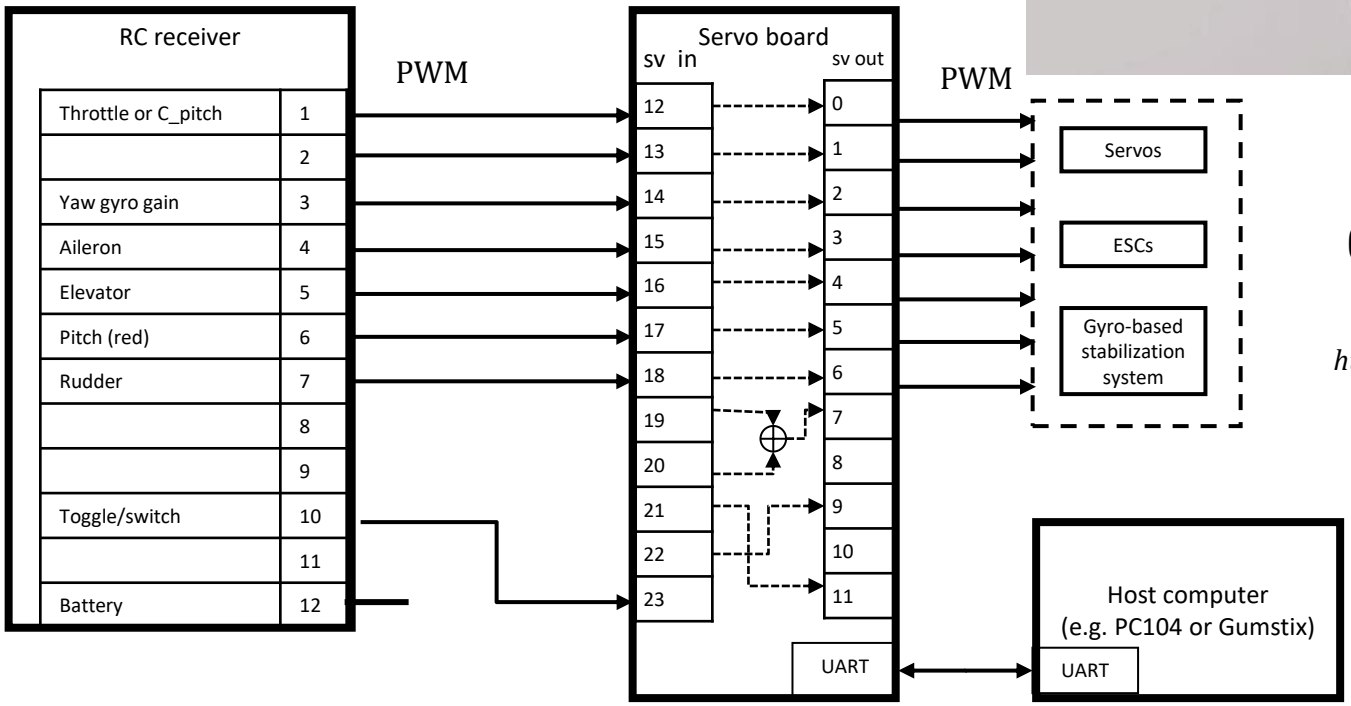


Actuator Management

Actuator management is to realize smooth switching between the manual control mode and the automatic control mode. The requirements for the actuator management are listed below:

- Reliable switching function
- Sufficient input/output channels
- Capability of recording actuator's input signal
- High resolution





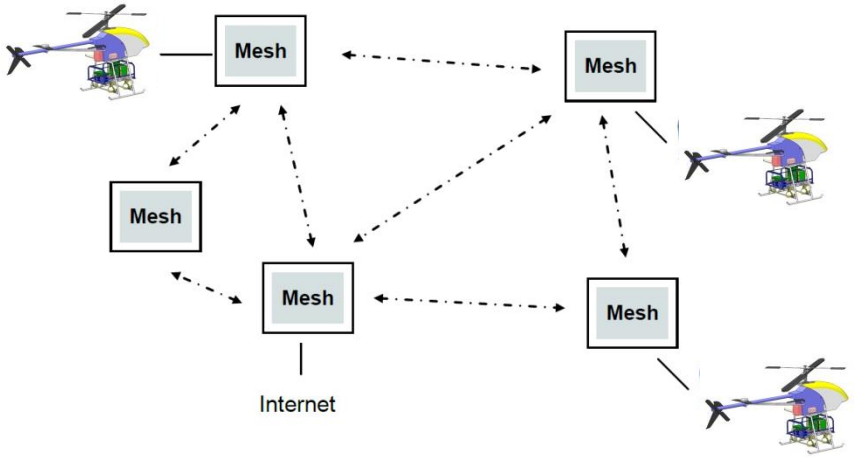
An example of cyclic-collective-pitch-mixing (CCPM) swashplate control for helicopter main blade. *(adopted from http://www.tburrow.com/portfolio_swash.html#)*

Communication Unit

The typical values of data bandwidth and communication range of those communication devices are summarized

Communication module	Data bandwidth	Range	Protocol
UART interface	115200 bps	32 km	UART
802.11g	54 Mbps	250 m	TCP/IP
3G modem	300 kbps	Hundred km	TCP/IP
4G modem	5 Mbps	Hundred km	TCP/IP

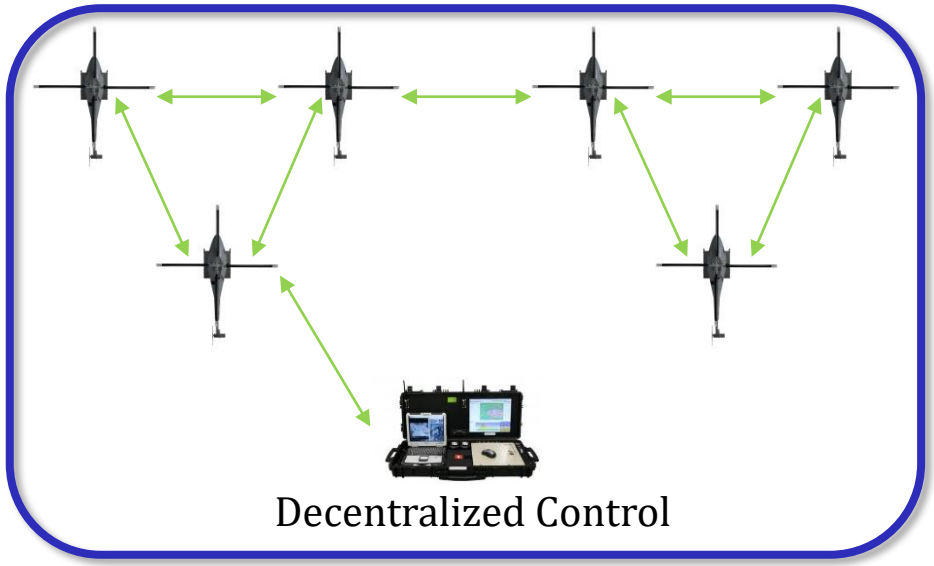
Access point to station

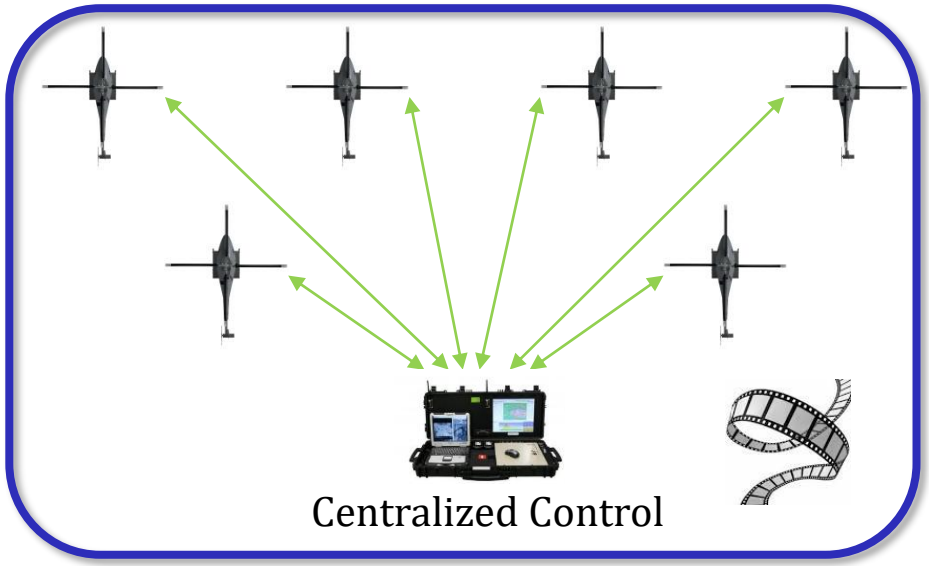


Mesh Network

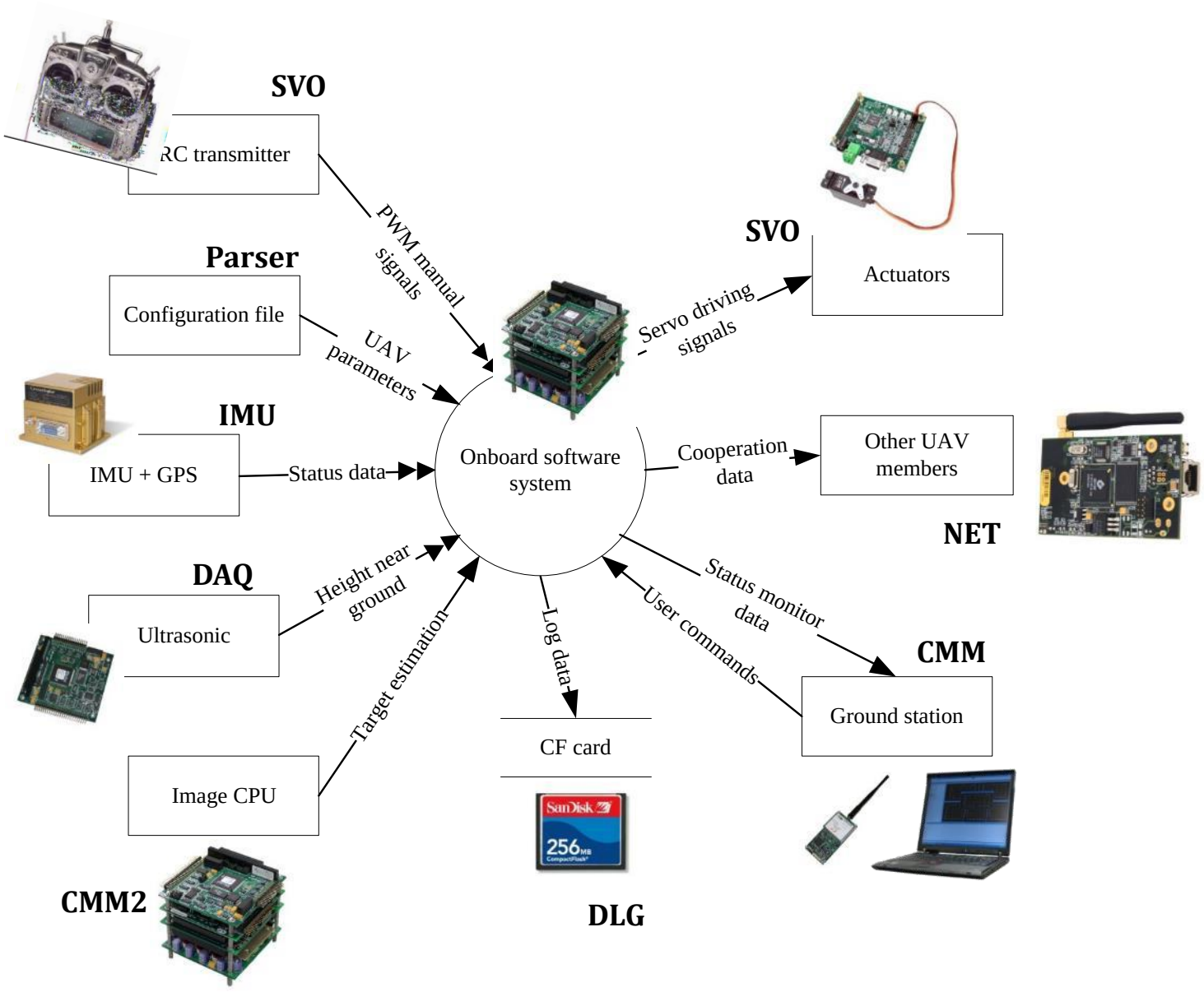
Communication Unit

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.

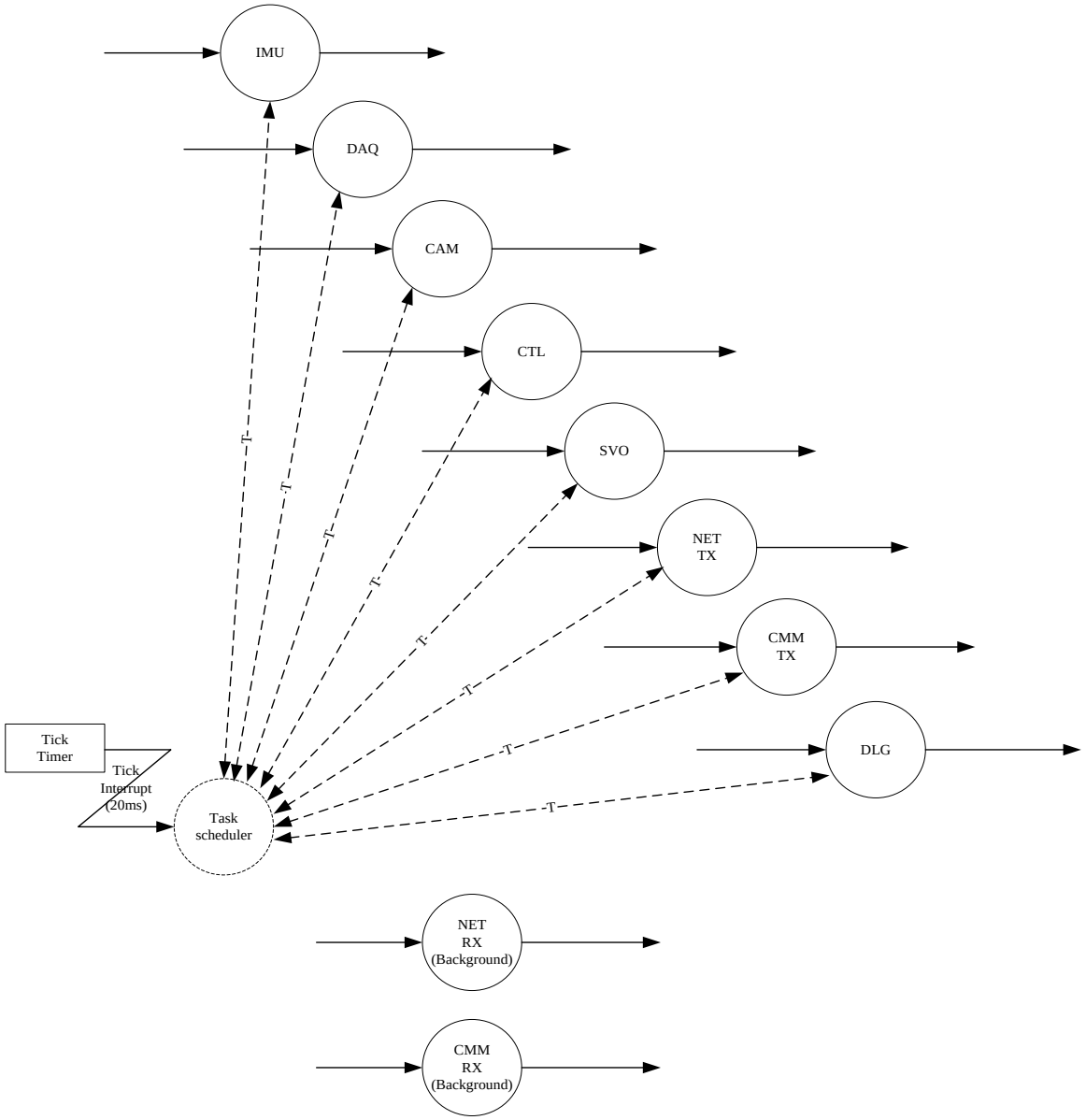




Onboard Software – Modular Design



Onboard Software – Task Scheduling



1. Multitasking (Multithreading) is necessary for real-time software

2. Task synchronization is realized via signal, IMU -> DAQ -> CAM -> CTL -> SVO -> NET -> CMM -> DLG

3. Background tasks are scheduled by the RTOS scheduler, will be activated once no other threads are working

4. Each working thread represents a state in the whole system behaviors

5. Task scheduler manages the state transitions

Onboard Software System – Performance Test

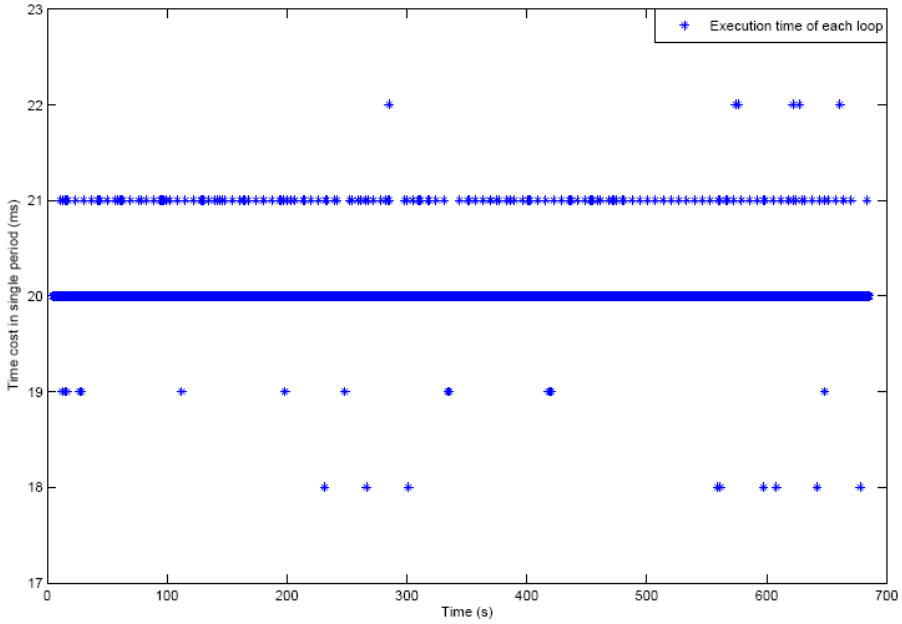


Figure 3.15: Time intervals between each loop.

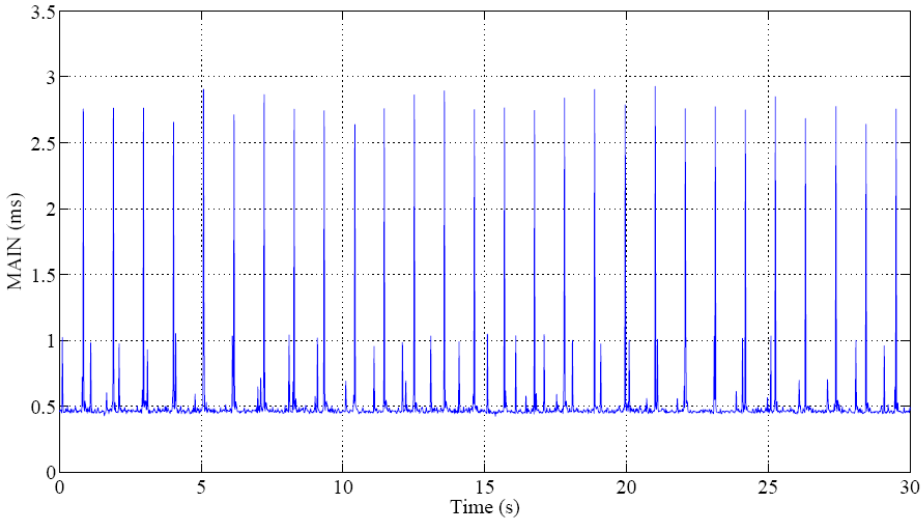
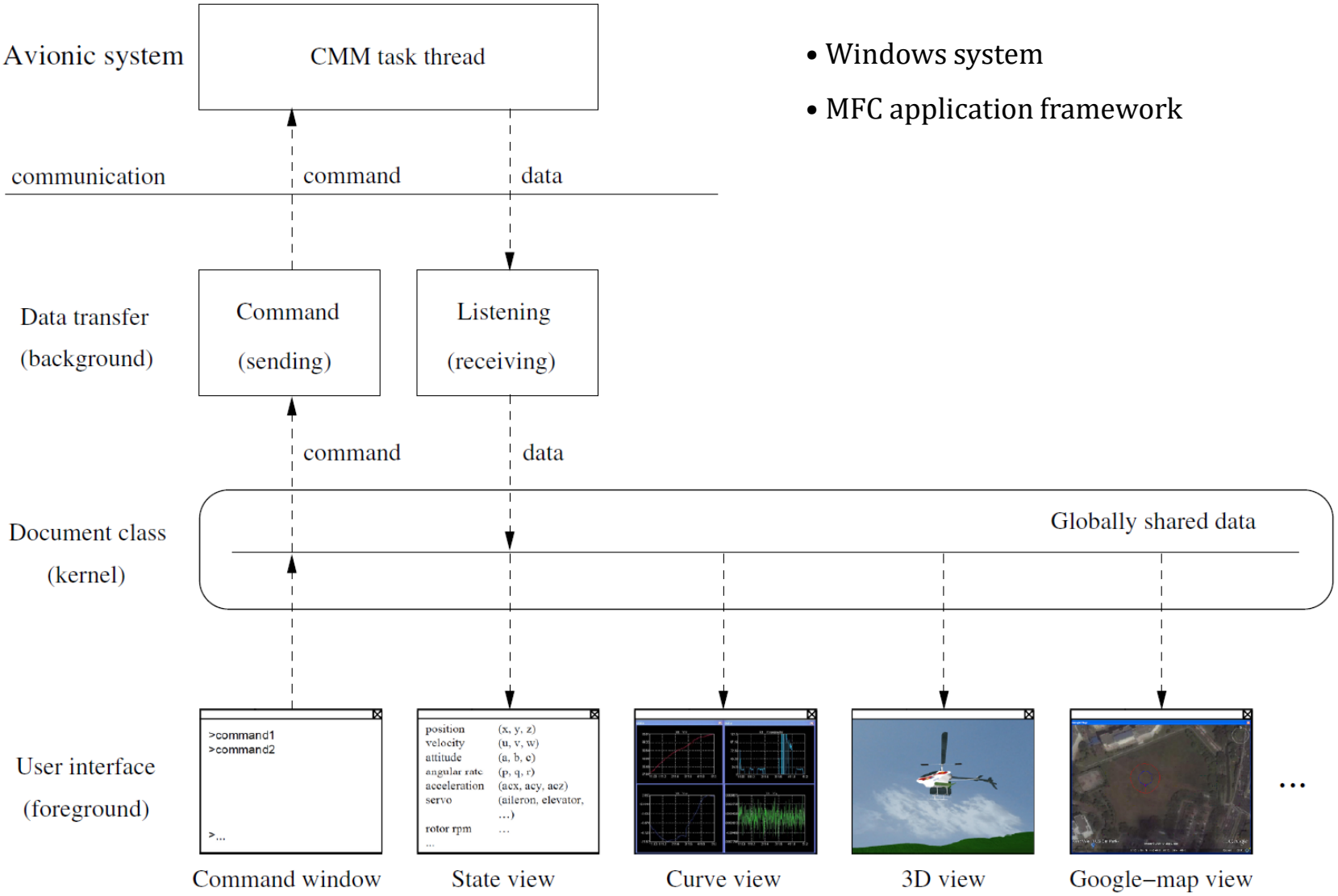


Figure 3.16: Time consumption statistics for main loop.

CPU maximum working load

$$\text{usage}_{\max} = \tau_{\max}/T_{\text{cyc}} = 2.93/20 = 14.65\%$$

Ground Control System – Software Framework

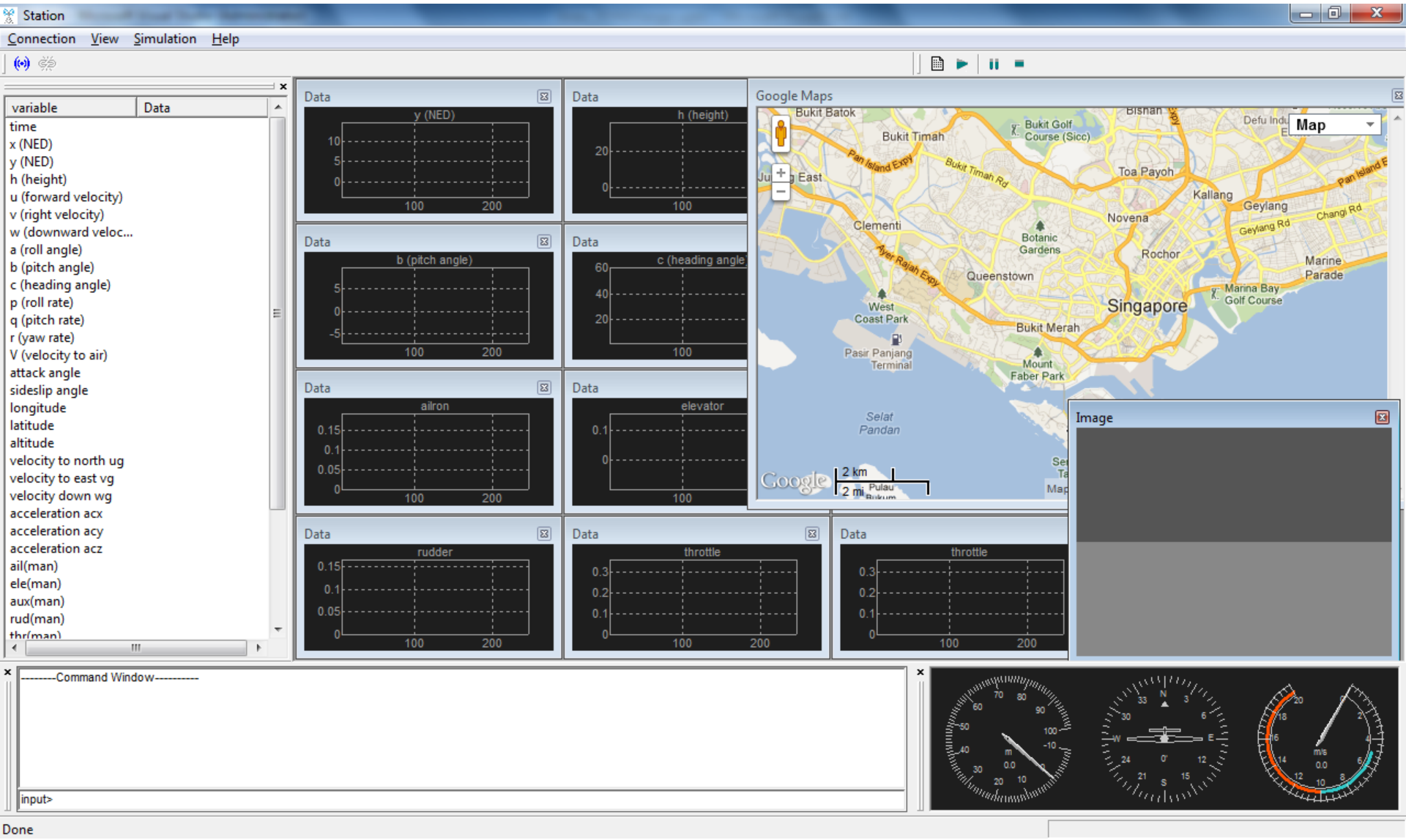


- Windows system
- MFC application framework

Software System – Safety Features

Safety Mode	Description	Triggering Events
Automatically return home	The UAV will cease its mission and follow a pre-defined safety trajectory to land on 'home' location autonomously.	▪ Mission related sensor faulty (1 second or outlier occurs) ▪ Loss of remote transmitter signal (>10 seconds without re-gaining communication link) ▪ UAV hit 'soft geo-fence' boundary ▪ Etc.
Automatically land	The UAV will land gradually on its current location and stop engine when landed with or without GPS navigation.	▪ Low battery (single cell less than 3.6V) ▪ Mission control board and redundant GPS sensor both faulty ▪ Bad control performance (> 17 meters position tracking error with respect to the reference at any given point of time) ▪ Etc.
Flight termination	The UAV will immediately turn off all the motors.	▪ The UAV exits 'hard geo-fence' ▪ Low level autopilot "lock-up" ▪ UAV attitude is more than 45 degrees ▪ Etc.

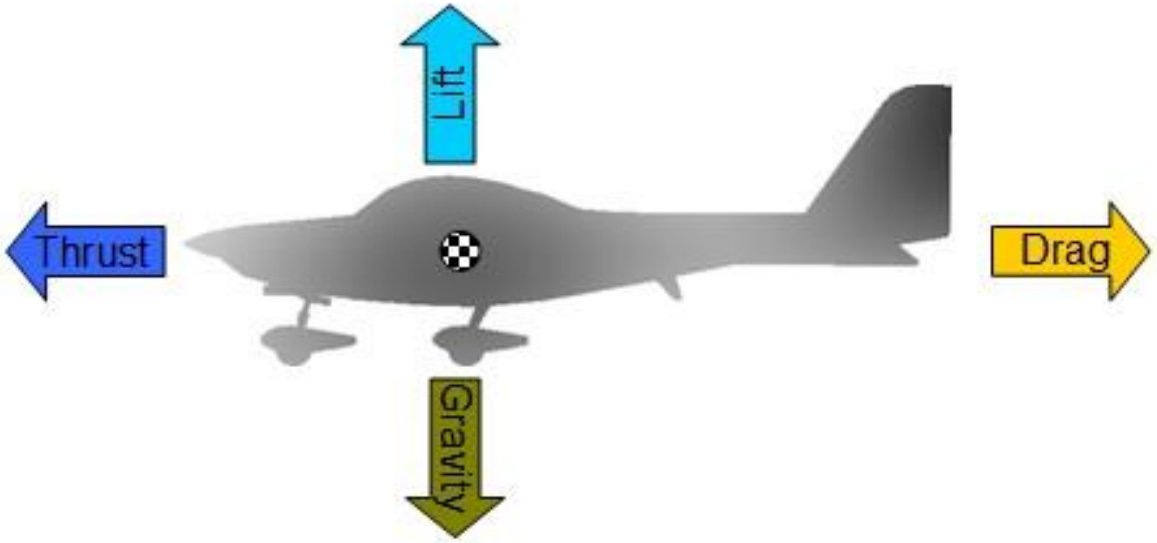
Ground Control System – Information Monitoring



What is a **flight dynamics model**?

A flight dynamics model is a set of equations that explains

1. How aerodynamic forces are generated on the aircraft body?
2. How actuators and external factors affect these forces?
3. How these forces affect the aircraft's motion?



Why do we need a **flight dynamics model**?

1. Design and tuning of autonomous flight controller

Most control techniques are model based. An accurate flight dynamics model can assist flight control design efficiently in terms of parameter tuning and performance evaluation.

2. Design and modification of aircraft structure

Aerodynamic stability, physical limits and characteristics can be analyzed based on the obtained model. Mechanical modification or re-structuring can be applied if there is inherent deficiencies in the original structure.

UAV Flight Dynamics Modeling

How do we obtain a **flight dynamics model**?

1) First-principles Modeling Approach

- Build mathematical model by manipulating equations of physics
- Complex, nonlinear, rigorous with physical meanings
- Test-bench experiments

2) System Identification Approach

- Build mathematical model by fitting measurement data
- Simplified, linear, black-box
- Flight test experiments

3) Combination of the above two approaches

UAV Flight Dynamics Modeling

How do we obtain a **flight dynamics model**?

1) First-principles Modeling Approach

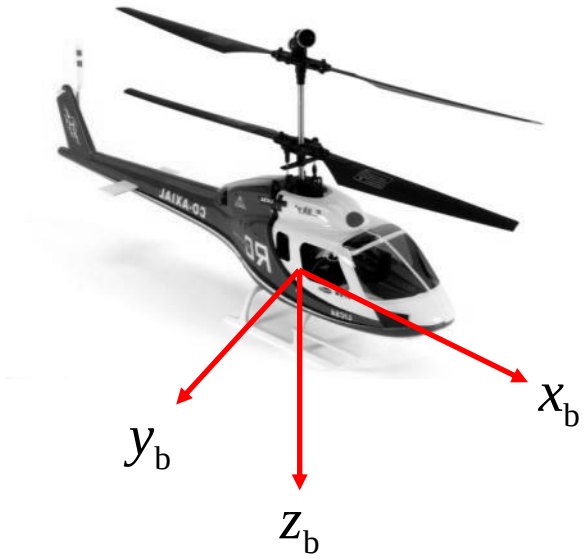
- Build mathematical model by manipulating equations of physics
- Complex, nonlinear, rigorous with physical meanings
- Test-bench experiments

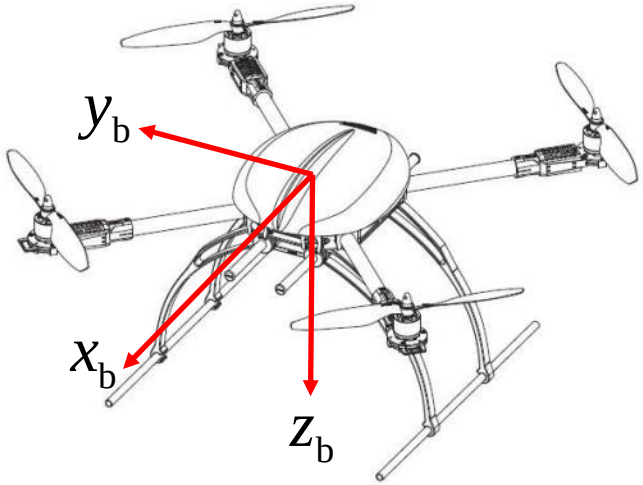
2) System Identification Approach

- Build mathematical model by fitting measurement data
- Simplified, linear, black-box
- Flight test experiments

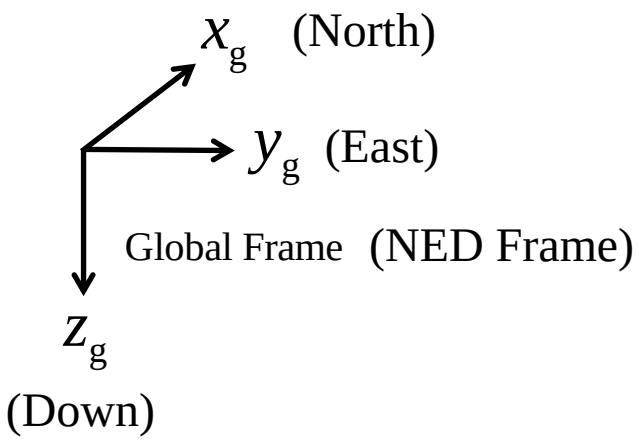
3) Combination of the above two approaches

Body Frame



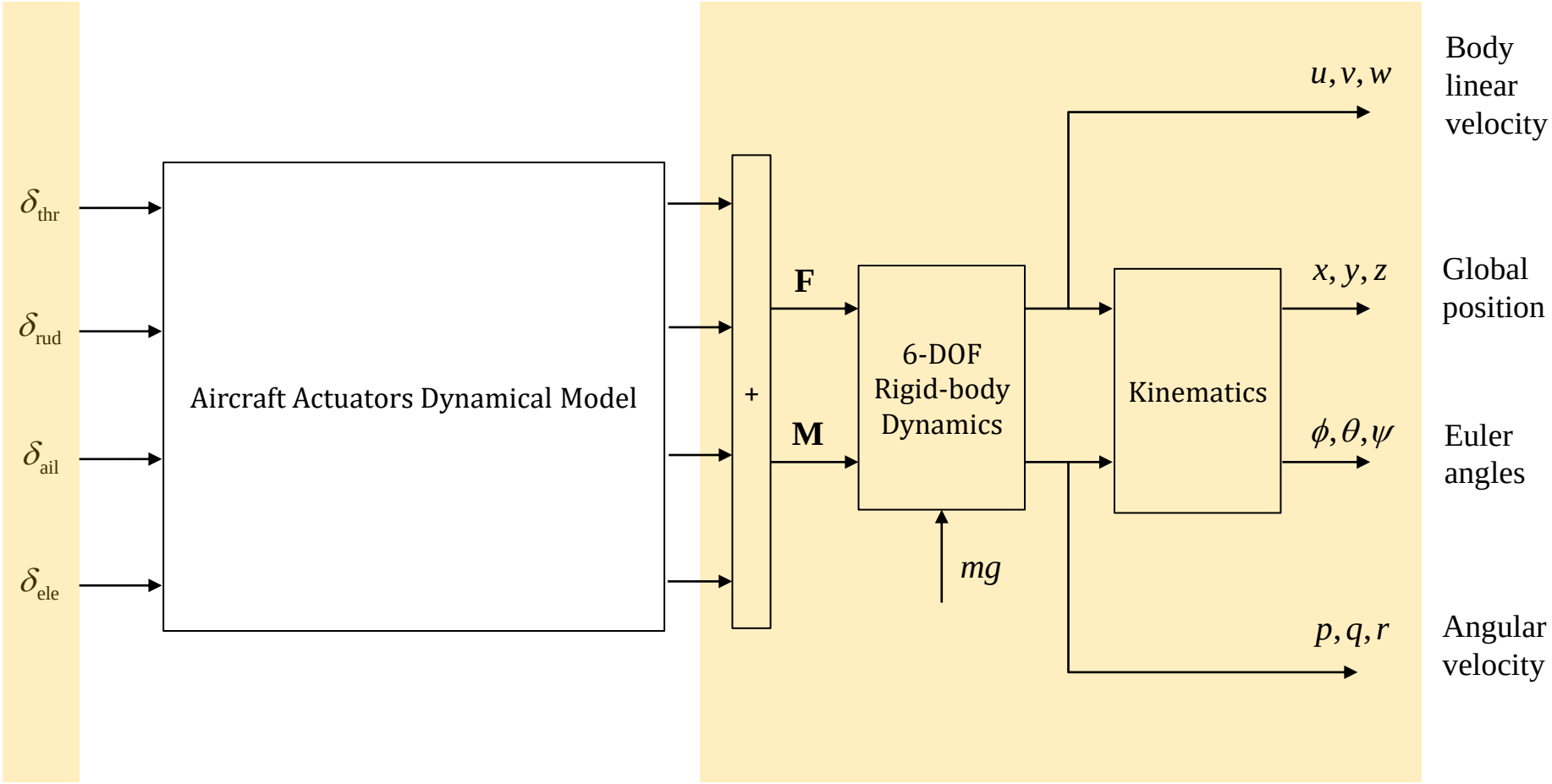


Ground Frame



First-principles Modeling Approach

General aircraft model structure



First-principles Modeling Approach

Rigid-body dynamics

Body-frame acceleration

Body-frame resultant force

$$\begin{pmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \end{pmatrix} = \frac{1}{m}\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

Cross-coupling between axes

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \mathbf{J}^{-1}\left\{\begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} - \begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \mathbf{J}\begin{pmatrix} p \\ q \\ r \end{pmatrix}\right\}$$

Body-frame angular acceleration

Body-frame resultant torque

First-principles Modeling Approach

Kinematics

Derivative of global-frame position

$\mathbf{R}_{n/b}$

Body-frame velocity

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \begin{bmatrix} c_\psi c_\theta & c_\psi s_\theta s_\phi - s_\psi c_\phi & c_\psi s_\theta c_\phi + s_\psi s_\phi \\ s_\psi c_\theta & s_\psi s_\theta s_\phi + c_\psi c_\phi & s_\psi s_\theta c_\phi - c_\psi s_\phi \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{bmatrix} 1 & s_\phi s_\theta / c_\theta & c_\phi s_\theta / c_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$

Derivative of Euler angles

$\mathbf{S}^{-1}$

Body-frame angular velocity

Different types of rotorcraft UAV

1) Multi-rotor (quadrotor) UAV

- Multiple rotors to create difference in torque
- Multiple rotors to create thrust

2) Fixed-pitch Co-axial UAV

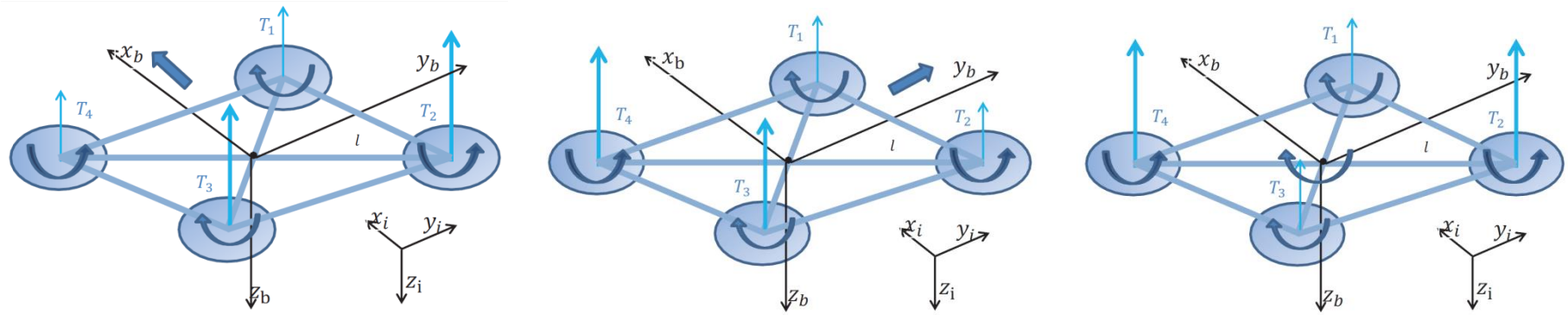
- Two counter rotating main rotors to create thrust and balance torque
- Two servo motors to control direction of flight

3) Single-rotor UAV

- One main rotor for the lift
- One tail rotor to balance heading angle
- Multiple servo motors to control collective pitch and direction of flight

First-principles Modeling Approach

Quadrotor working principles

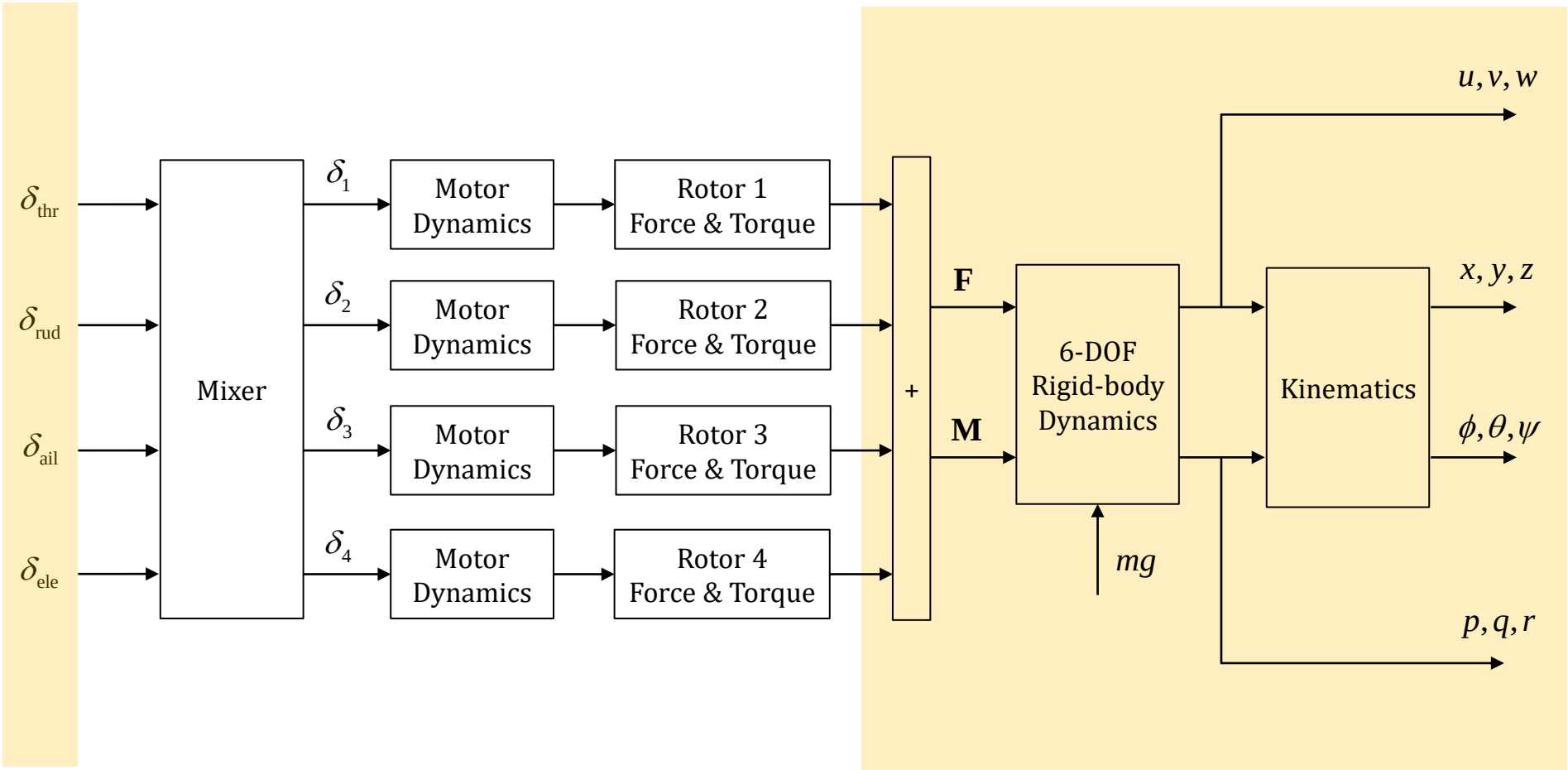


(a) Pitching (b) Rolling (c) Yawing

- Symmetric structure
- All types of motion controlled by adjusting motor speeds
- All thrust forces in the UAV body frame z-axis direction
- Open-loop dynamics fast and unstable

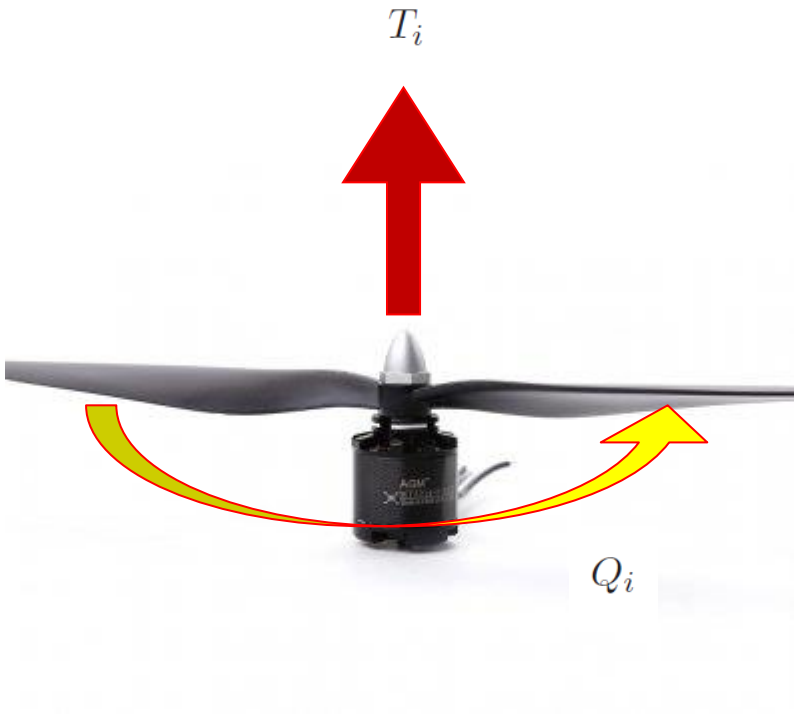
First-principles Modeling Approach

Quadrotor model structure



First-principles Modeling Approach

Motor and propeller



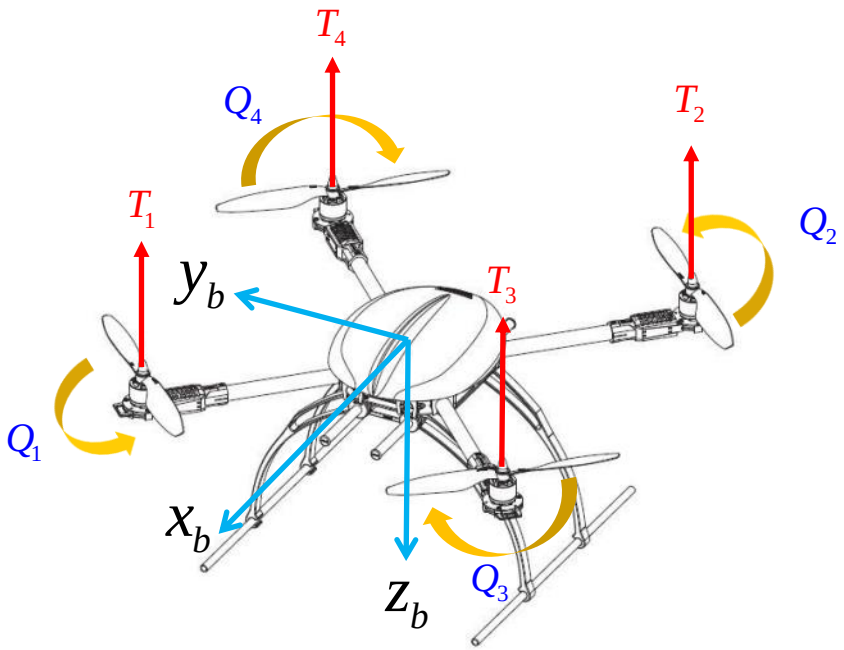
$$T_i = C_T \rho A R^2 \Omega_i^2,$$
$$Q_i = C_Q \rho A R^3 \Omega_i^2,$$

$$T_i = k_T \Omega_i^2,$$
$$Q_i = k_Q \Omega_i^2,$$

- Assume no flapping dynamics
- Air density remains constant

First-principles Modeling Approach

Quadrotor force & torque decomposition



$$\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = -\begin{pmatrix} 0 \\ 0 \\ |T_1|+|T_2|+|T_3|+|T_4| \end{pmatrix} + mg \begin{pmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{pmatrix}$$

$$\begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} (|T_2|+|T_3|-|T_1|-|T_4|)\times l/\sqrt{2} \\ (|T_1|+|T_3|-|T_2|-|T_4|)\times l/\sqrt{2} \\ |Q_1|+|Q_2|-|Q_3|-|Q_4| \end{pmatrix}$$

$$T_i = k_T \Omega_i^2,$$

$$Q_i = k_Q \Omega_i^2,$$

1) **Multi-rotor (quadrotor) UAV**
   - Multiple rotors to create difference in torque
   - Multiple rotors to create thrust

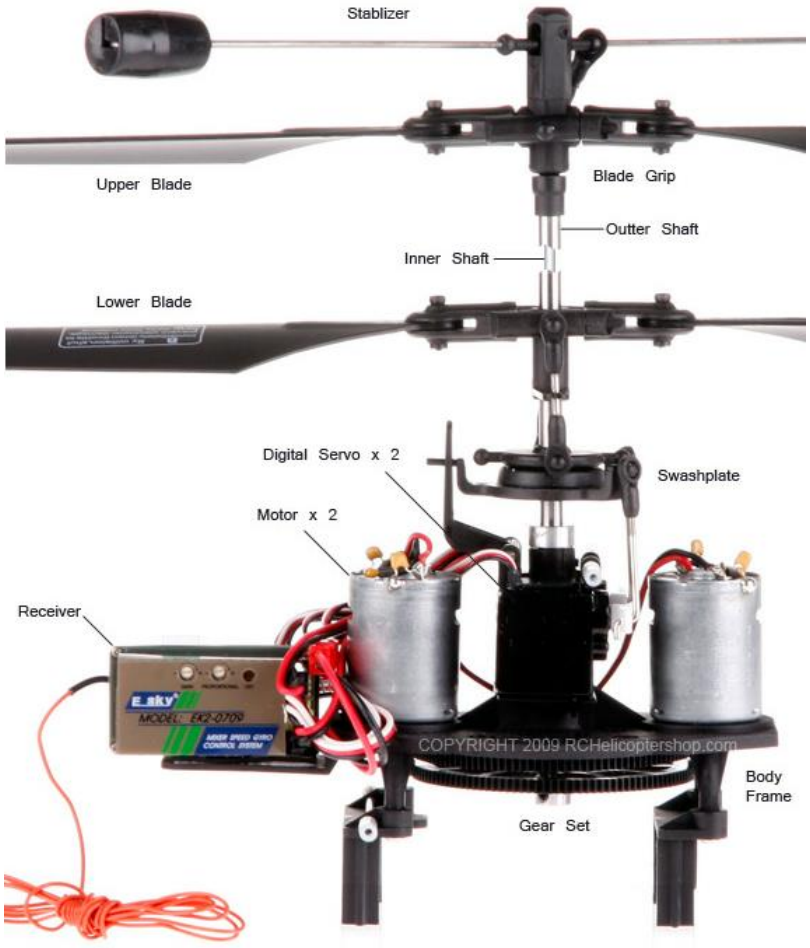
2) **Fixed-pitch Co-axial UAV**
   - Two counter rotating main rotors to create thrust and balance torque
   - Two servo motors to control direction of flight

3) **Single-rotor UAV**
   - One main rotor for the lift
   - One tail rotor to balance heading angle
   - Multiple servo motors to control collective pitch and direction of flight

First-principles Modeling Approach

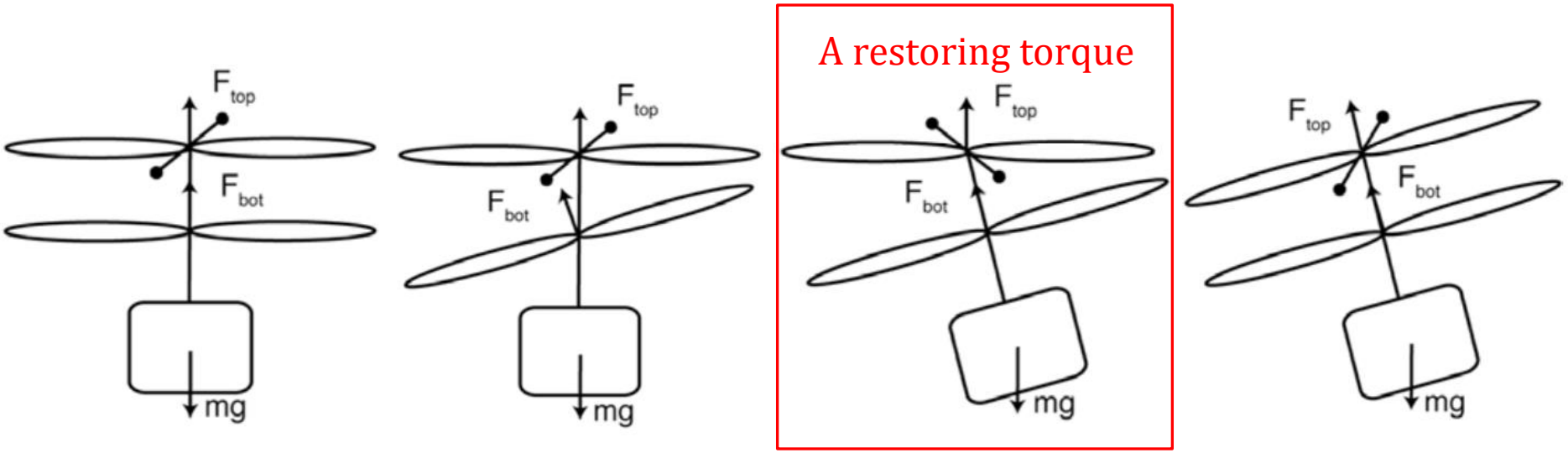
Coaxial helicopter working principles

- Dual main rotors spin in opposite directions
- Rotational speed can be controlled
  - Sum → Heave motion
  - Difference → Yaw motion
- Bottom rotor attached to swashplate
  - Servo 1 → Roll motion
  - Servo 2 → Pitch motion
- Top rotor attached to stabilizer bar
  - Roll & Pitch motion damped



First-principles Modeling Approach

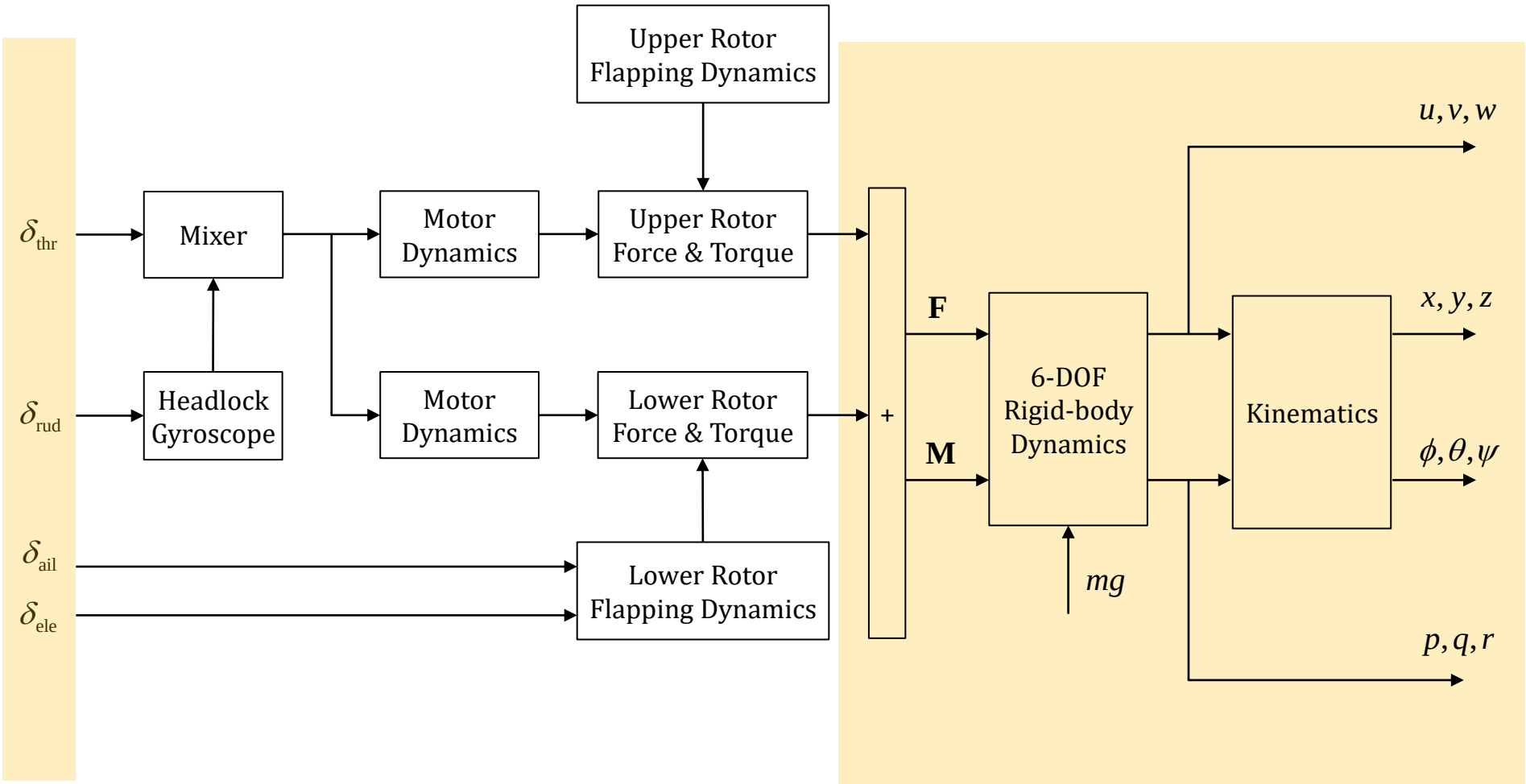
Coaxial helicopter working principles (stabilizer bar)



- Roll and pitch dynamics slowed down
- Stability increased (inherently stable)
- Maneuverability decreased

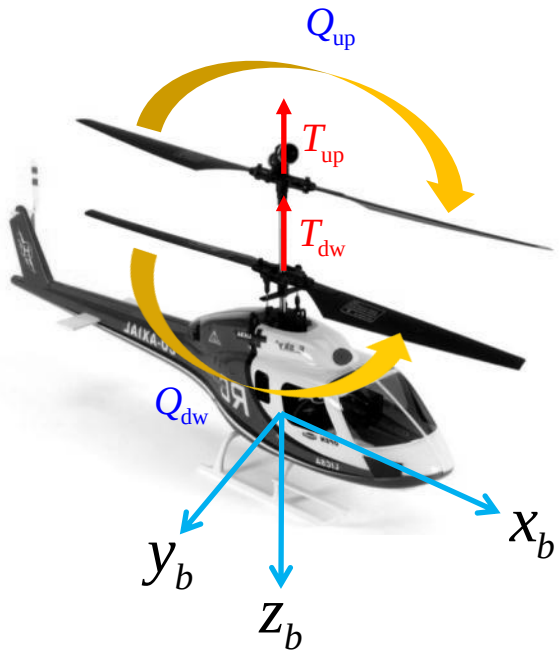
First-principles Modeling Approach

Coaxial helicopter model structure



First-principles Modeling Approach

Coaxial helicopter force & torque decomposition



$$T_i = k_T \Omega_i^2,$$
$$Q_i = k_Q \Omega_i^2,$$

$$\begin{pmatrix} F_x \\ F_y \\ F_z \end{pmatrix} = |T_{up}| \begin{pmatrix} -\sin a_{up} \\ \sin b_{up} \\ -\cos a_{up} \cos b_{up} \end{pmatrix} + |T_{dw}| \begin{pmatrix} -\sin a_{dw} \\ \sin b_{dw} \\ -\cos a_{dw} \cos b_{dw} \end{pmatrix} + mg \begin{pmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{pmatrix}$$

$$\begin{pmatrix} M_x \\ M_y \\ M_z \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -l_{up} \end{pmatrix} \times |T_{up}| \begin{pmatrix} -\sin a_{up} \\ \sin b_{up} \\ -\cos a_{up} \cos b_{up} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ -l_{dw} \end{pmatrix} \times |T_{dw}| \begin{pmatrix} -\sin a_{dw} \\ \sin b_{dw} \\ -\cos a_{dw} \cos b_{dw} \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ |Q_{up}| - |Q_{dw}| \end{pmatrix}$$

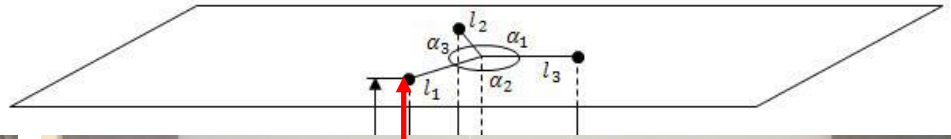
First-principles Modeling Approach

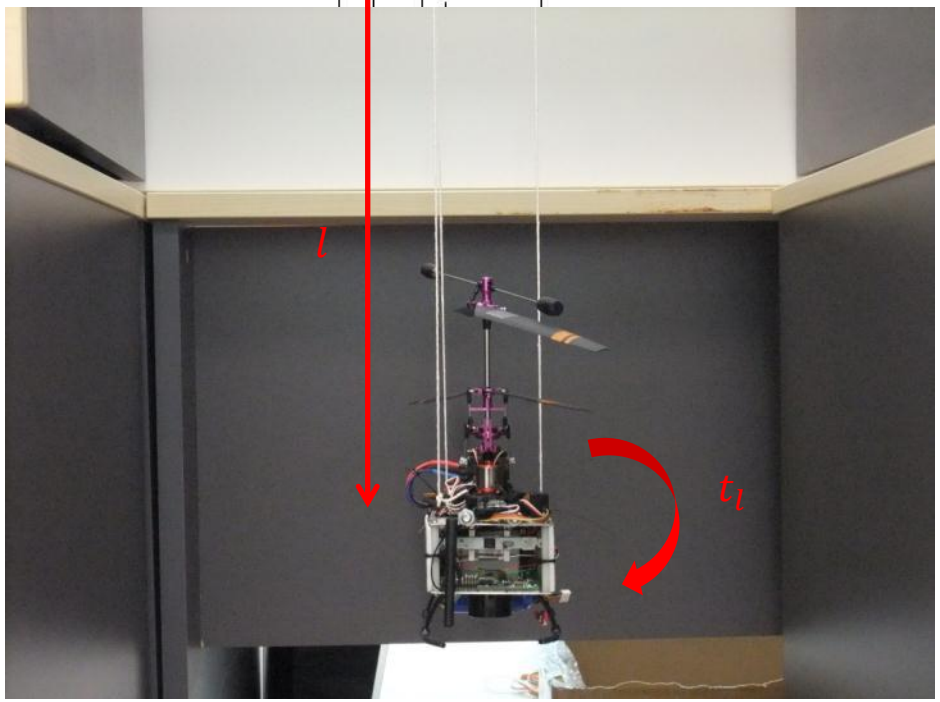
Parameters identification

Mass

$$m = 0.934 \text{ kg}$$

Moment of inertial



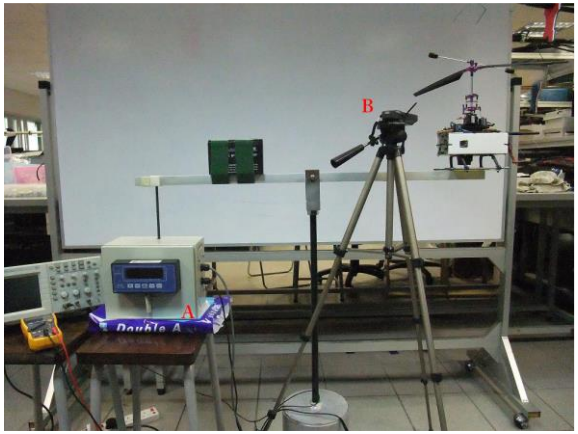


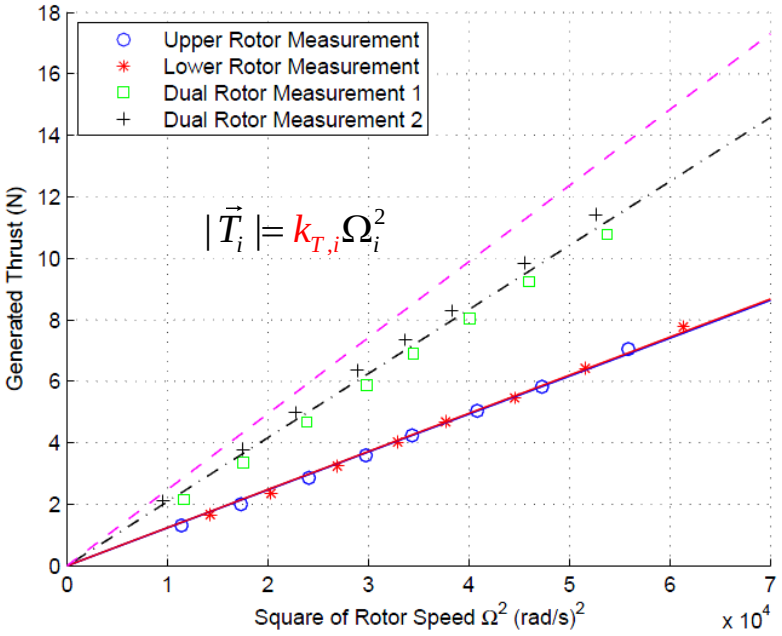
$$J_{zz} = \frac{mgl_1l_2l_3t_l^2}{4\pi^2 l} \cdot \frac{l_1 \sin\alpha_1 + l_2 \sin\alpha_2 + l_3 \sin\alpha_3}{l_2l_3 \sin\alpha_1 + l_1l_3 \sin\alpha_2 + l_1l_2 \sin\alpha_3}$$

First-principles Modeling Approach

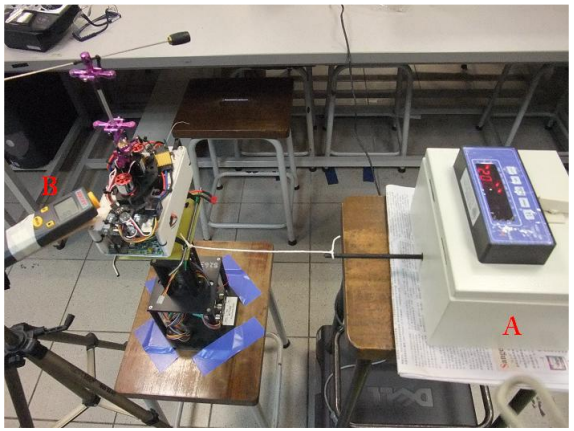
Parameters identification

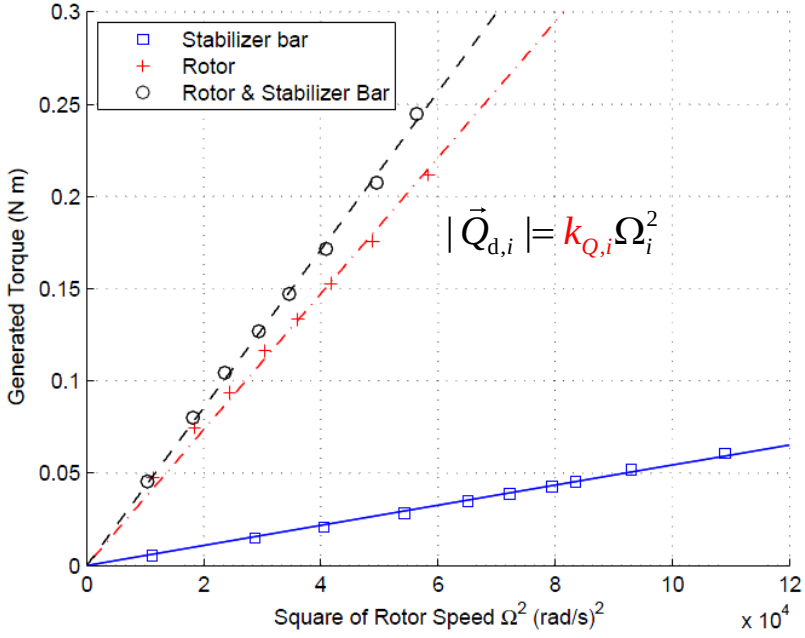
Thrust





Torque

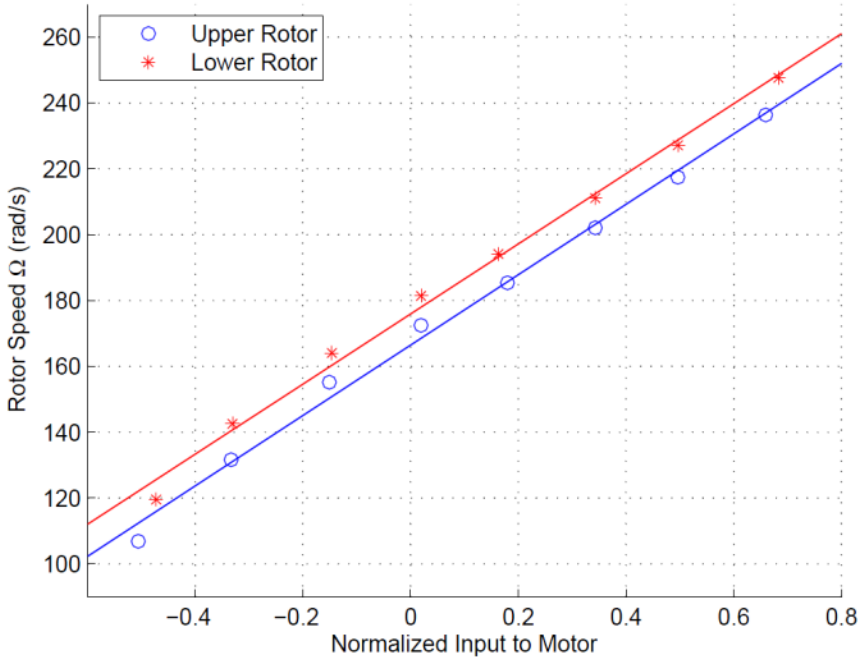


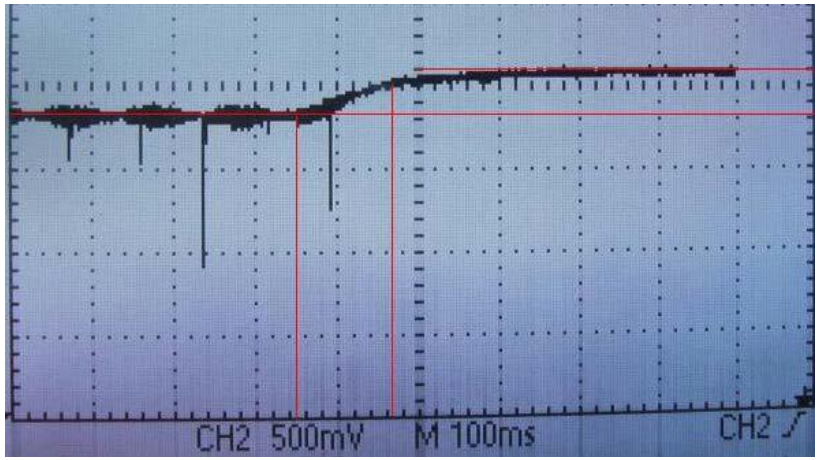


First-principles Modeling Approach

Parameters identification

Motor dynamics



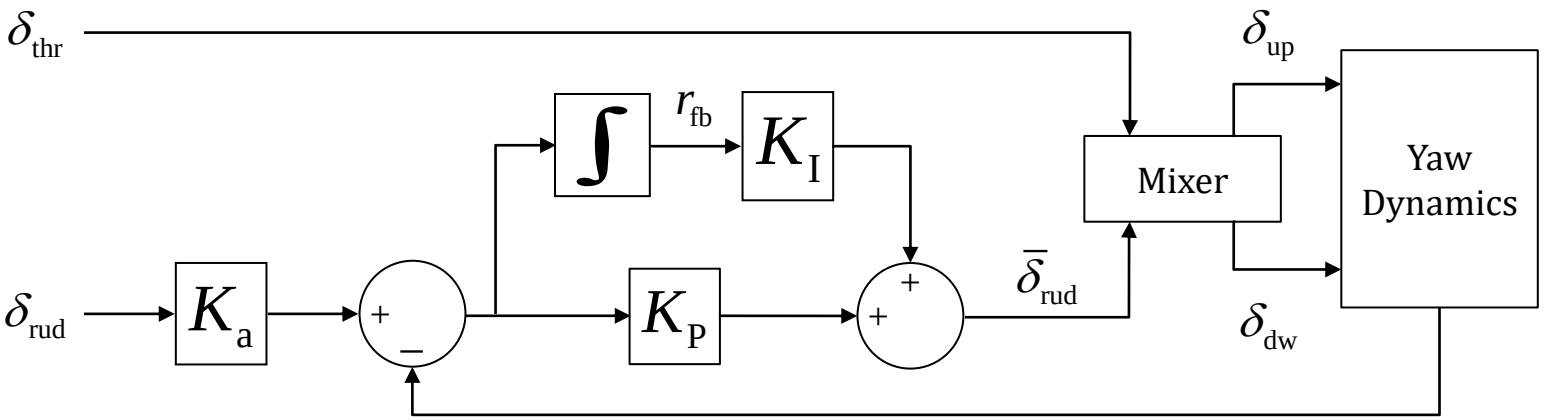


$$\dot{\Omega}_{\text{up}} = \frac{1}{\tau_{\text{mt}}}(m_{\text{up}}\delta_{\text{up}} + \Omega^{*}_{\text{up}} - \Omega_{\text{up}})$$

$$\dot{\Omega}_{\text{dw}} = \frac{1}{\tau_{\text{mt}}}(m_{\text{dw}}\delta_{\text{dw}} + \Omega^{*}_{\text{dw}} - \Omega_{\text{dw}})$$

First-principles Modeling Approach

Mixer & headlock gyro dynamics



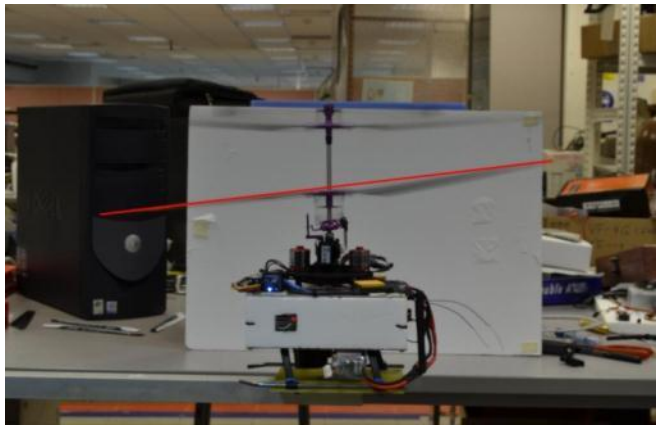
$$\bar{\delta}_{rud} = K_P(K_a\delta_{rud} - r) + K_I r_{fb}$$

$$\dot{r}_{fb} = K_a\delta_{rud} - r$$

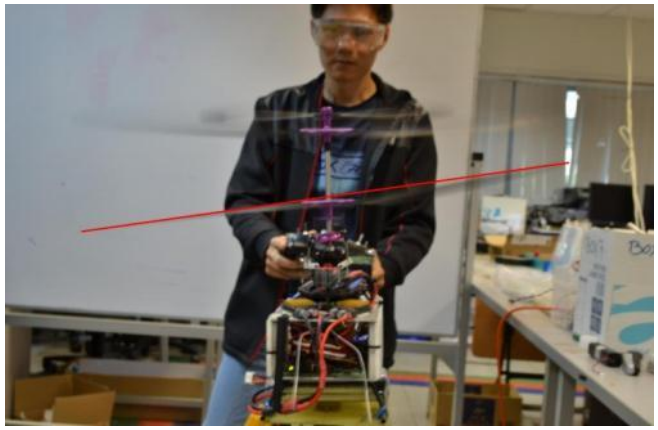
$$\delta_{up} = \delta_{thr} + \bar{\delta}_{rud}$$

$$\delta_{dw} = \delta_{thr} - \bar{\delta}_{rud}$$

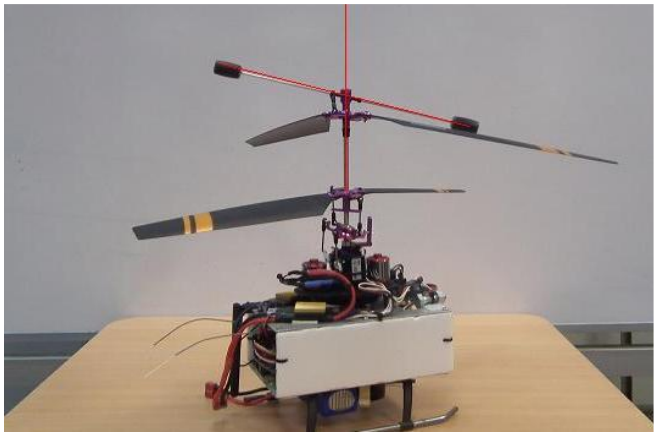
Tip-path-plane dynamics



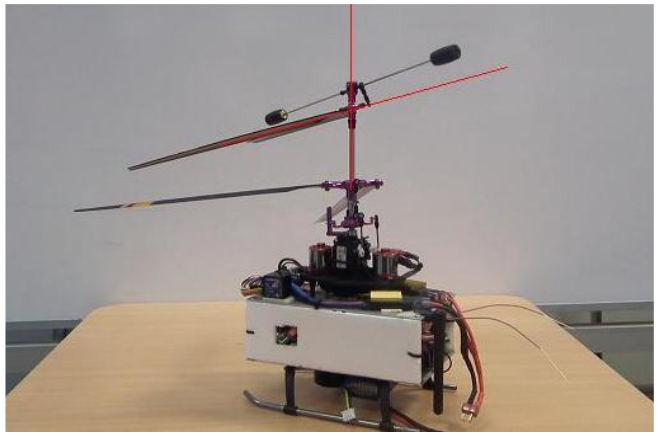
$$a_{\text{dw}} = A_{a,\text{dw}}\,\delta_{\text{ele}} + A_{b,\text{dw}}\,\delta_{\text{ail}}$$



$$b_{\text{dw}} = B_{b,\text{dw}}\,\delta_{\text{ail}} + B_{a,\text{dw}}\,\delta_{\text{ele}}$$



$$\dot{a}_{\text{up}} = -\frac{1}{\tau_{\text{up}}} a_{\text{up}} - A_{a,\text{up}} q - A_{b,\text{up}} p$$



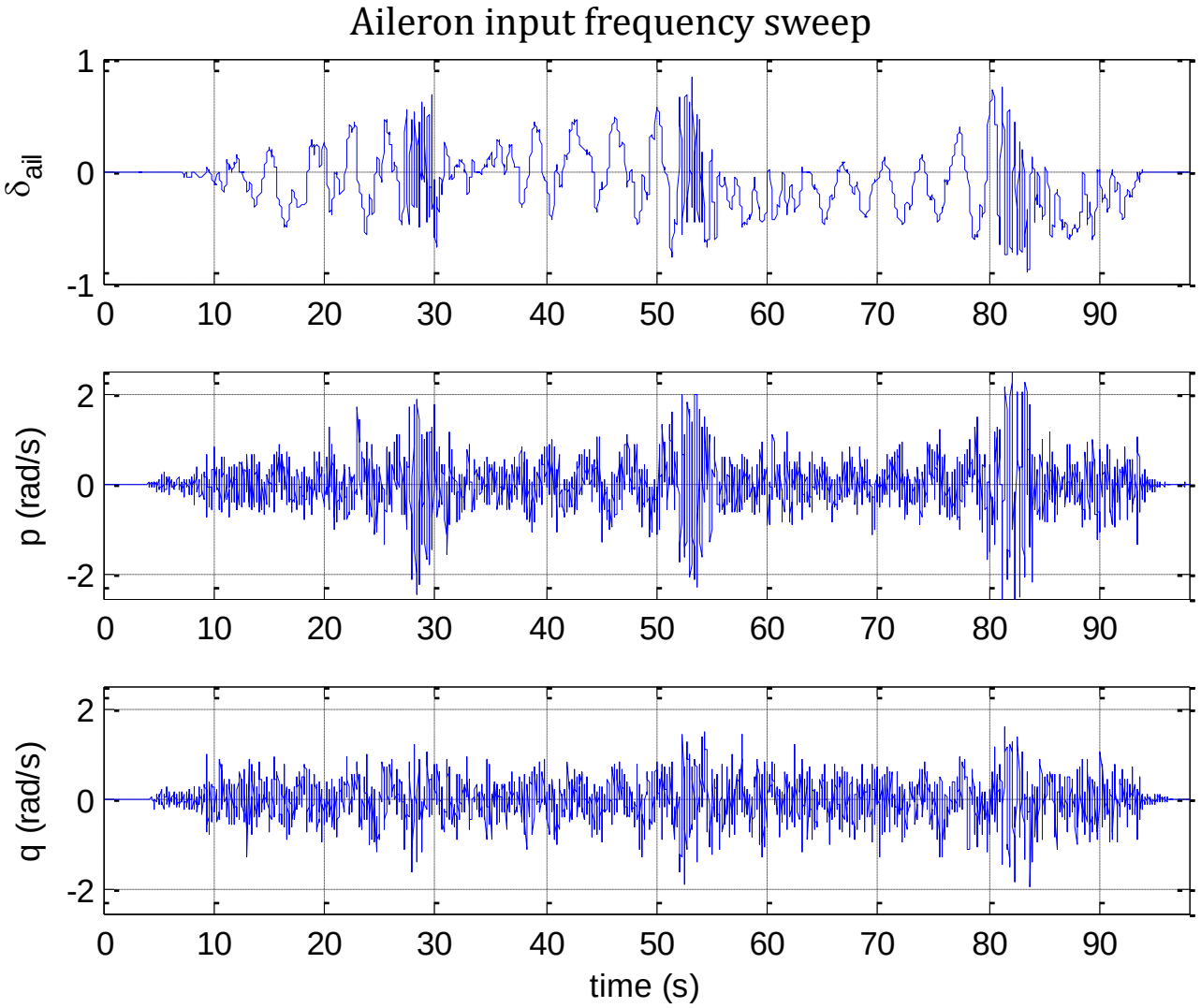
$$\dot{b}_{\text{up}} = -\frac{1}{\tau_{\text{up}}} b_{\text{up}} - B_{b,\text{up}} p - B_{a,\text{up}} q$$

Tip-path-plane dynamics

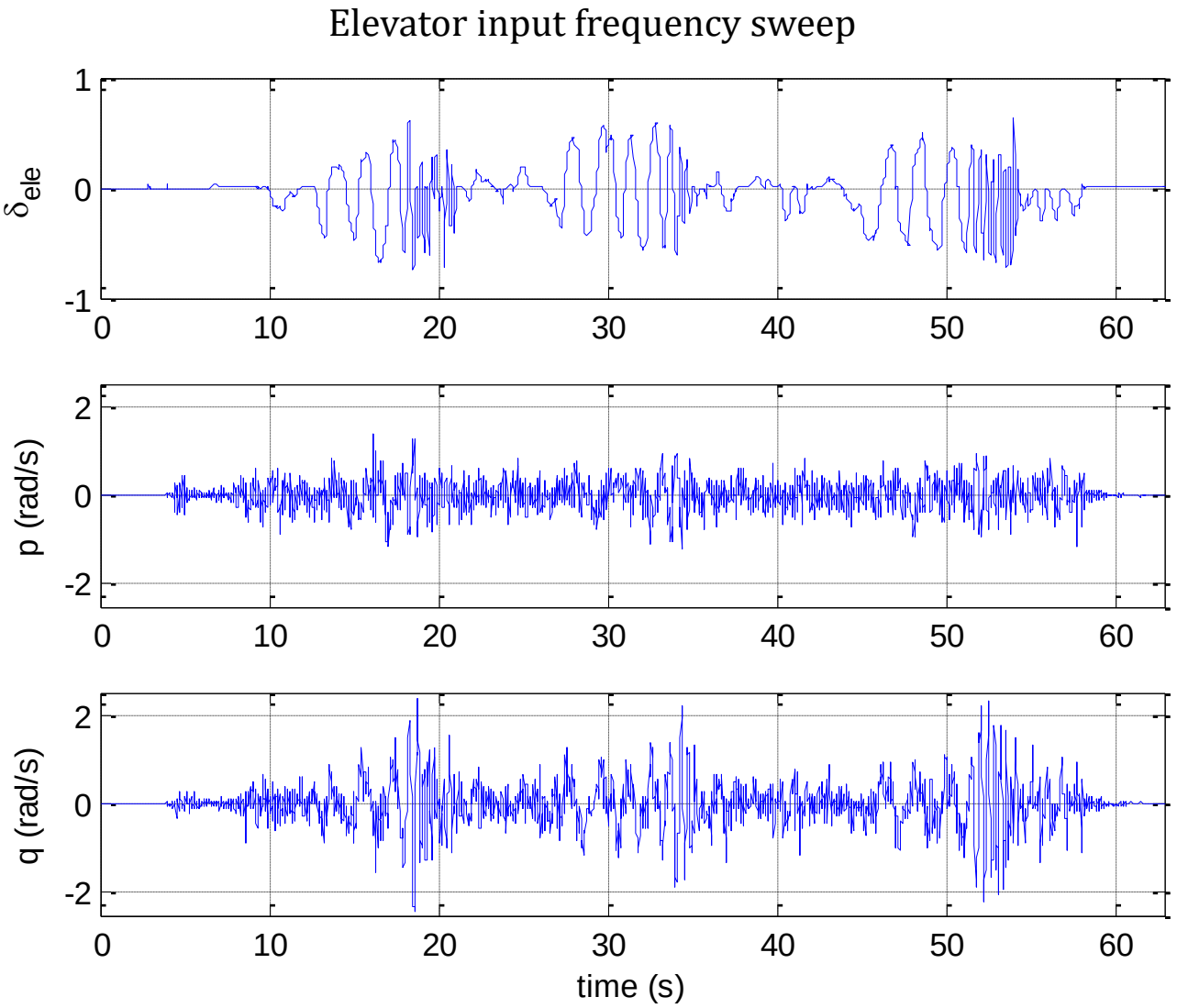
$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{a}_{\text{up}} \\ \dot{b}_{\text{up}} \end{pmatrix} = \begin{bmatrix} \frac{-X_{\text{dw}}B_{p,\text{dw}}}{J_{xx}} & 0 & 0 & \frac{X_{\text{up}}}{J_{xx}} \\ 0 & \frac{-X_{\text{dw}}A_{q,\text{dw}}}{J_{yy}} & \frac{X_{\text{up}}}{J_{yy}} & 0 \\ -A_{b,\text{up}} & -A_{a,\text{up}} & -\frac{1}{\tau_{\text{sb}}} & 0 \\ -B_{b,\text{up}} & -B_{a,\text{up}} & 0 & -\frac{1}{\tau_{\text{sb}}} \end{bmatrix} \begin{pmatrix} p \\ q \\ a_{\text{up}} \\ b_{\text{up}} \end{pmatrix} + \begin{bmatrix} \frac{X_{\text{dw}}B_{b,\text{dw}}}{J_{xx}} & \frac{X_{\text{dw}}B_{a,\text{dw}}}{J_{xx}} \\ \frac{X_{\text{dw}}A_{b,\text{dw}}}{J_{yy}} & \frac{X_{\text{dw}}A_{a,\text{dw}}}{J_{yy}} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} \delta_{\text{ail}} \\ \delta_{\text{ele}} \end{pmatrix}$$

$$\begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{a}_{\text{up}} \\ \dot{b}_{\text{up}} \end{bmatrix} = \begin{bmatrix} -17.19 & 0 & 0 & 934.1 \\ 0 & -5.360 & 291.3 & 0 \\ 0.2745 & -0.49 & -5 & 0 \\ -0.49 & -0.2745 & 0 & -5 \end{bmatrix} \begin{bmatrix} p \\ q \\ a_{\text{up}} \\ b_{\text{up}} \end{bmatrix} + \begin{bmatrix} -102.48 & -38.08 \\ -11.73 & 31.95 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_{\text{ail}} \\ \delta_{\text{ele}} \end{bmatrix}$$

Tip-path-plane dynamics

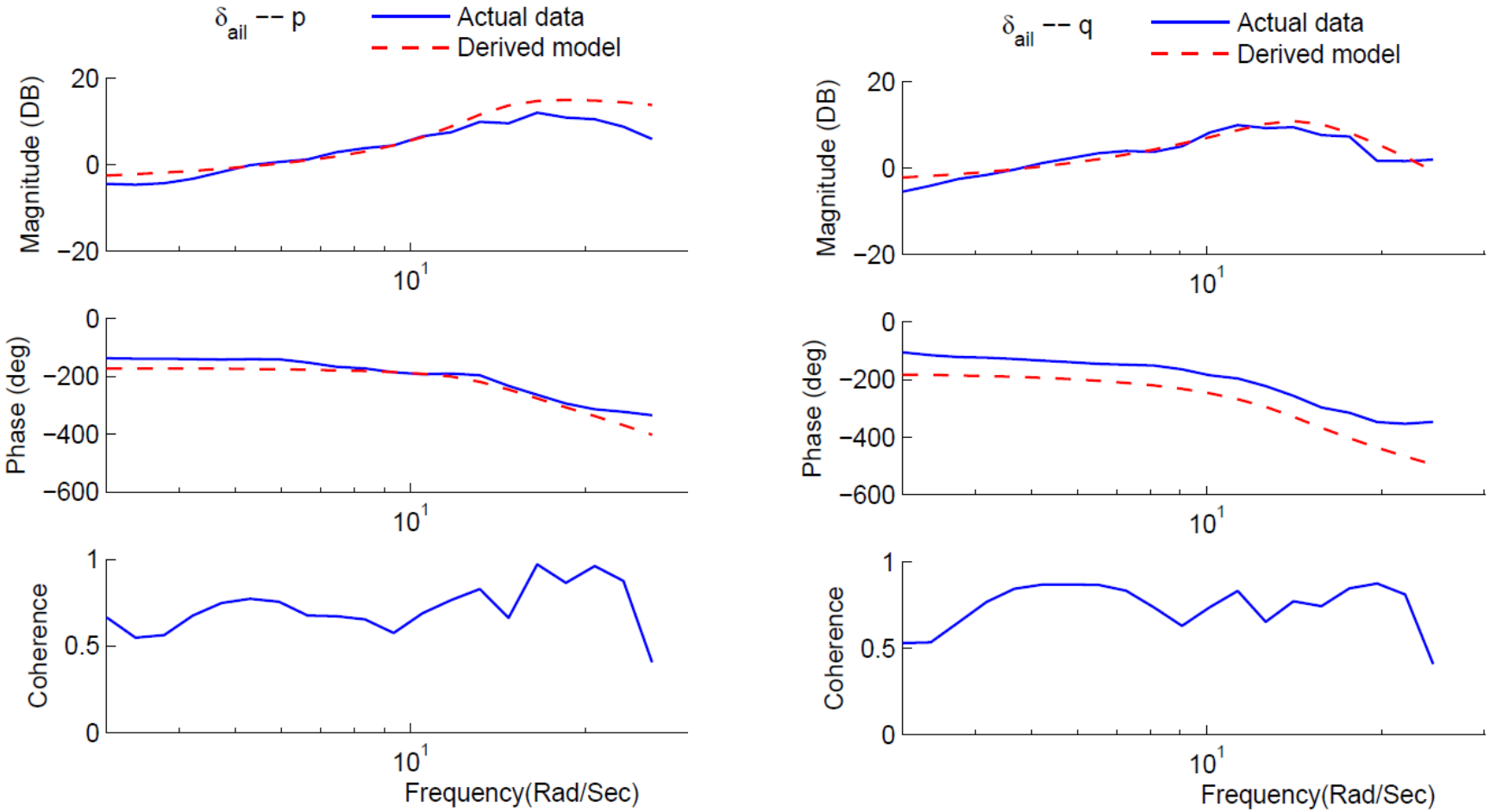


Tip-path-plane dynamics



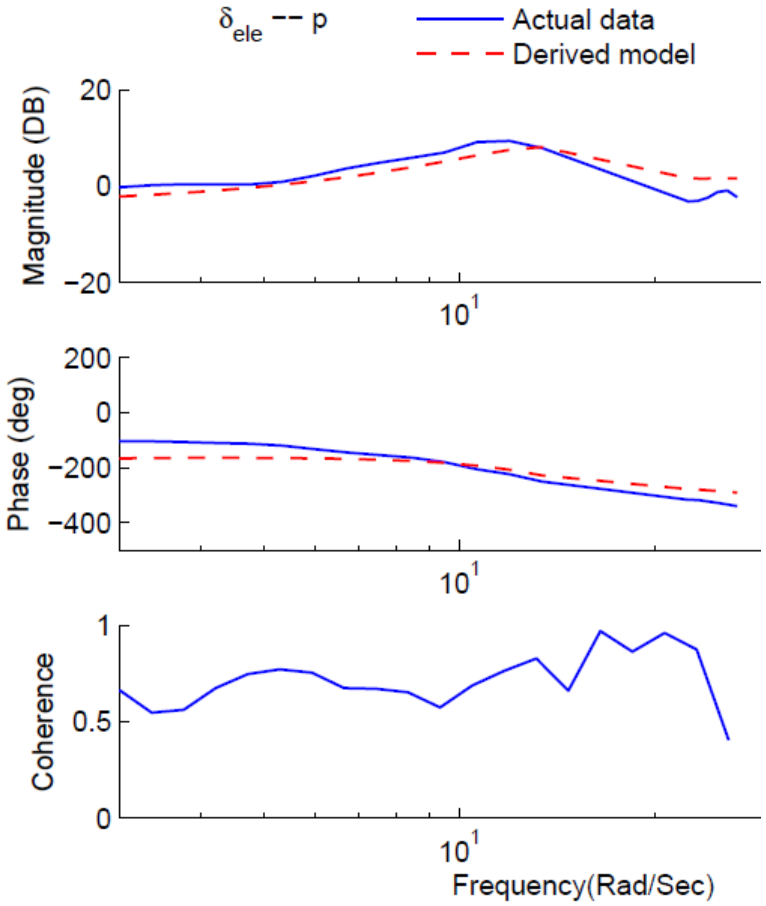
Tip-path-plane dynamics

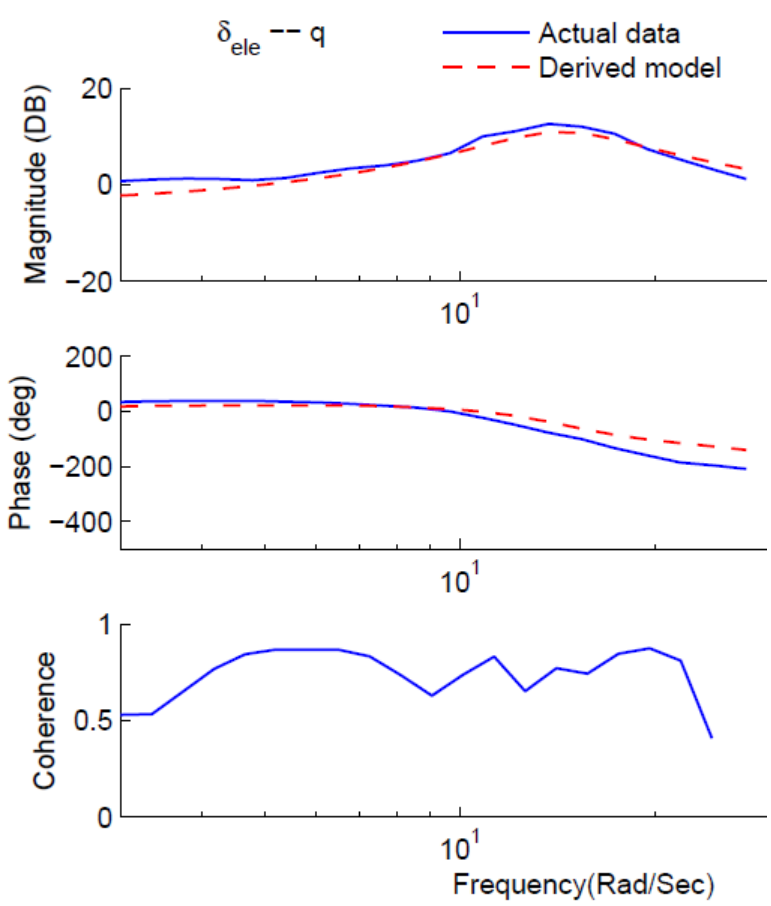
Parameter identification via aileron frequency sweep



Tip-path-plane dynamics

Parameter identification via elevator frequency sweep





First-principles Modeling Approach

Identified parameters of coaxial helicopter

1. Direct measurement
2. Test-bench experiment
3. Flight test data

$\rho = 1.204$ $m = 0.977$ $R = 0.250$ $g = 9.781$	$m_{\text{up}} = 106.90$ $m_{\text{dw}} = 106.45$ $\tau_{\text{sb}} = 0.2$ $\tau_{\text{mt}} = 0.12$	$\Omega^*_{\text{up}} = 203.38$ $\Omega^*_{\text{dw}} = 217.88$ $K_\beta = 4.377$
$S_{fx} = 0.00835$ $S_{fy} = 0.01310$ $S_{fz} = 0.01700$	$K_{\text{a}} = 6.4267$ $K_{\text{P}} = 0.667/K_{\text{a}}$ $K_{\text{I}} = 0.713/K_{\text{a}}$	$J_{xx} = 0.0059$ $J_{yy} = 0.0187$ $J_{zz} = 0.0030$
$l_{up} = 0.195$ $l_{dw} = 0.120$	$J_{\text{up}} = 6.8613 \cdot 10^{-4}$ $J_{\text{dw}} = 3.2906 \cdot 10^{-4}$	$A_q = 0.0204$ $B_p = 0.0204$
$k_{T,\text{up}} = 1.23 \cdot 10^{-4}$ $k_{T,\text{dw}} = 8.50 \cdot 10^{-5}$ $k_{Q,\text{up}} = 4.23 \cdot 10^{-6}$ $k_{Q,\text{dw}} = 3.68 \cdot 10^{-6}$	$A_{a,\text{up}} = 0.4900$ $A_{b,\text{up}} = -0.2745$ $B_{a,\text{up}} = 0.2745$ $B_{b,\text{up}} = 0.4900$	$A_{a,\text{dw}} = 0.1217$ $A_{b,\text{dw}} = -0.0450$ $B_{a,\text{dw}} = -0.0450$ $B_{b,\text{dw}} = -0.1217$

Different types of rotorcraft UAV

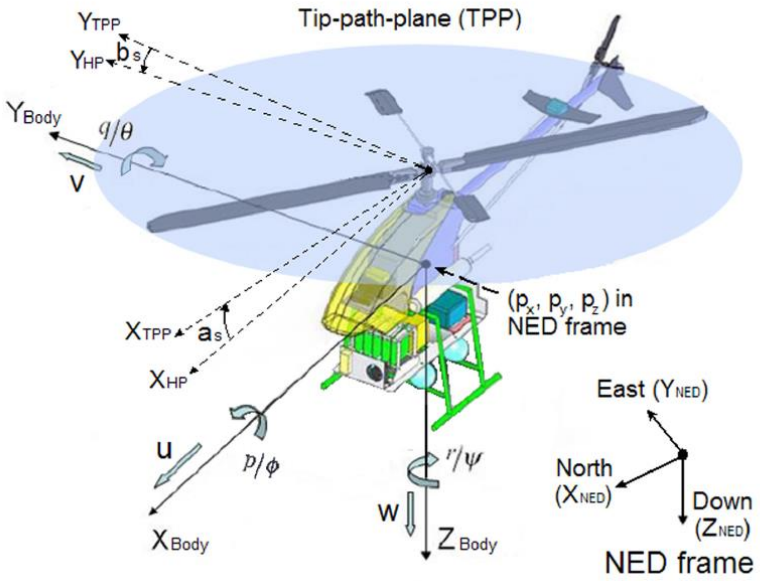
1) **Multi-rotor (quadrotor) UAV**
   - Multiple rotors to create difference in torque
   - Multiple rotors to create thrust

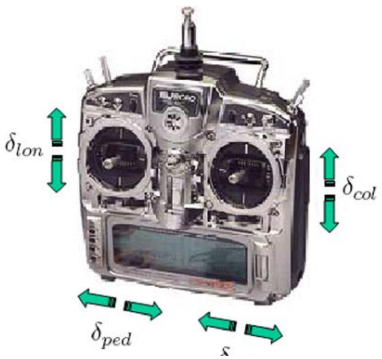
2) **Fixed-pitch Co-axial UAV**
   - Two counter rotating main rotors to create thrust and balance torque
   - Two servo motors to control direction of flight

3) **Single-rotor UAV**
   - One main rotor for the lift
   - One tail rotor to balance heading angle
   - Multiple servo motors to control collective pitch and direction of flight

First-principles Modeling Approach

Single-rotor helicopter





Variable	Physical description	Unit
p_x, p_y, p_z	Position vector along NED-frame x, y, and z axes	m
u, v, w	Velocity vector along body-frame x, y, and z axes	m/s
p, q, r	Roll, pitch, and yaw angular rates	rad/s
ϕ, θ, ψ	Euler angles	rad
a_s, b_s	Longitudinal and lateral tip-path-plane (TPP) flapping angle	rad
$\delta_{\text{ped,int}}$	Intermediate state in yaw rate gyro dynamics	NA
δ_{lat}	Normalized aileron servo input $(-1, 1)$	NA
δ_{lon}	Normalized elevator servo input $(-1, 1)$	NA
δ_{col}	Normalized collective pitch servo input $(-1, 1)$	NA
δ_{ped}	Normalized rudder servo input $(-1, 1)$	NA

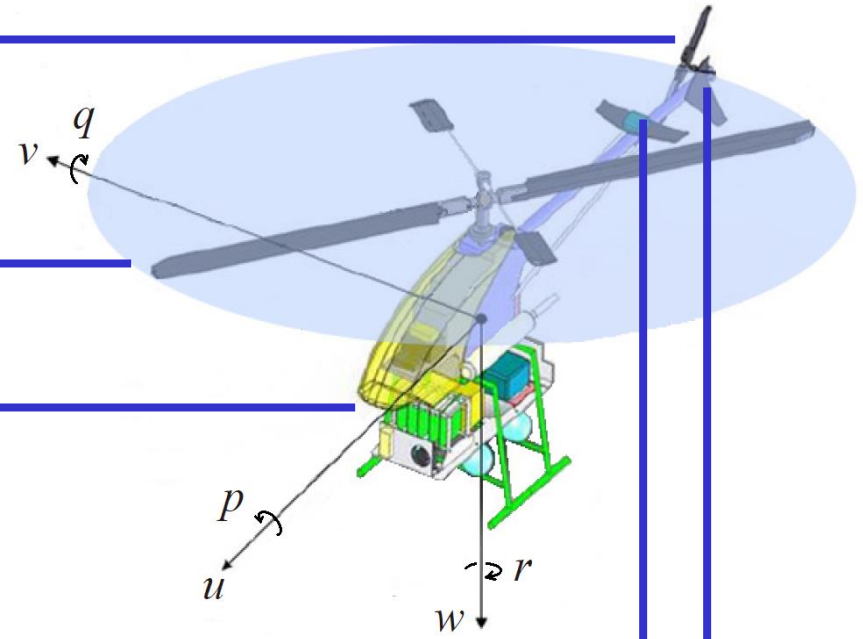
First-principles Modeling Approach

Single-rotor helicopter

Forces and Moments

$$F = F_{mr} + F_{tr} + F_{fus} + F_{hf} + F_{vf}$$

$$M = M_{mr} + M_{tr} + M_{fus} + M_{hf} + M_{vf}$$



$()_{mr}$ for the main rotor $()_{tr}$ for the tail rotor $()_{fus}$ for the fuselage $()_{hf}$ for the horizontal fin $()_{vf}$ for the vertical fin

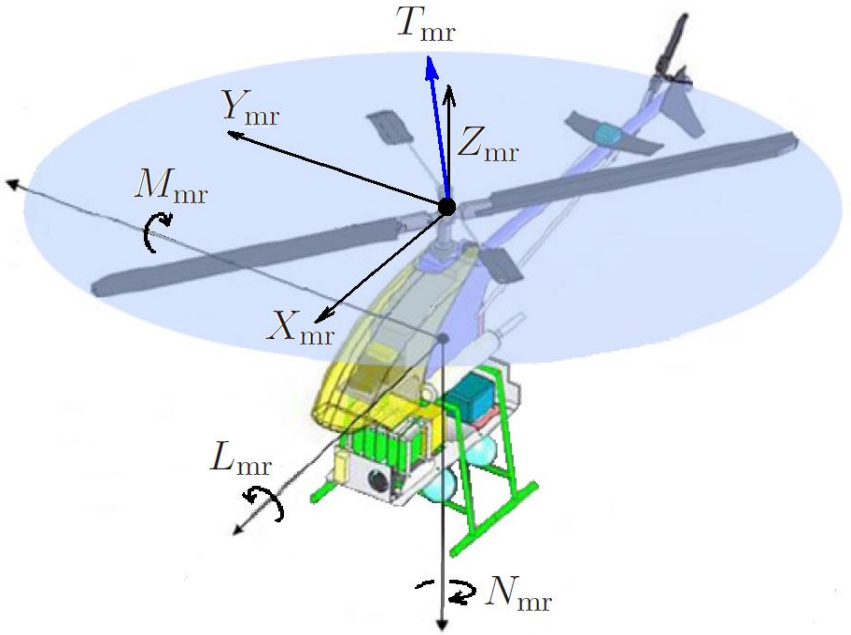




G. Cai, B. M. Chen, and T. H. Lee, *Unmanned Rotorcraft Systems*, New York: Springer, 2011.

Main rotor forces and moments

$$
\begin{aligned}
X_{\mathrm{mr}} &= -T\sin(a_{\mathrm{s}}) \\
Y_{\mathrm{mr}} &= T\sin(b_{\mathrm{s}}) \\
Z_{\mathrm{mr}} &= -T\cos(a_{\mathrm{s}})\cos(b_{\mathrm{s}}) \\
L_{\mathrm{mr}} &= (K_\beta + TH_{\mathrm{mr}})\sin(b_{\mathrm{s}}) \\
M_{\mathrm{mr}} &= (K_\beta + TH_{\mathrm{mr}})\sin(a_{\mathrm{s}}) \\
N_{\mathrm{mr}} &= -(P_{\mathrm{pr}} + P_{\mathrm{i}} + P_{\mathrm{pa}} + P_{\mathrm{c}})/\Omega
\end{aligned}
$$



$$
\begin{aligned}
T_{\mathrm{mr}} &= \frac{\rho\,\Omega_{\mathrm{mr}}\,R_{\mathrm{mr}}^2\,C_{\mathrm{l}\alpha,\mathrm{mr}}\,b_{\mathrm{mr}}\,c_{\mathrm{mr}}}{4}(w_{\mathrm{bl,mr}} - v_{\mathrm{i,mr}}) \\
v_{\mathrm{i,mr}}^2 &= \sqrt{\left(\frac{\hat{v}_{\mathrm{mr}}^2}{2}\right)^2 + \left(\frac{T_{\mathrm{mr}}}{2\,\rho\,\pi\,R_{\mathrm{mr}}^2}\right)^2} - \frac{\hat{v}_{\mathrm{mr}}^2}{2} \\
\hat{v}_{\mathrm{mr}}^2 &= u_{\mathrm{a}}^2 + v_{\mathrm{a}}^2 + w_{\mathrm{r,mr}}(w_{\mathrm{r,mr}} - 2\,v_{\mathrm{i,mr}}) \\
w_{\mathrm{r,mr}} &= w_{\mathrm{a}} + a_{\mathrm{s}}\,u_{\mathrm{a}} - b_{\mathrm{s}}\,v_{\mathrm{a}} \\
w_{\mathrm{bl,mr}} &= w_{\mathrm{r,mr}} + \frac{2}{3}\Omega_{\mathrm{mr}}\,R_{\mathrm{mr}}\,\theta_{\mathrm{col}}
\end{aligned}
$$

$$
\begin{aligned}
P_{\mathrm{pr}} &= \frac{\rho\Omega R^2 C_{\mathrm{D0}} b_{\mathrm{mr}} c_{\mathrm{mr}}}{8}\left[(\Omega R)^2 + 4.6(u_{\mathrm{a}}^2 + v_{\mathrm{a}}^2)\right] \\
P_{\mathrm{i}} &= T v_{\mathrm{i}} \\
P_{\mathrm{c}} &= -mgw_{\mathrm{a}} \\
P_{\mathrm{pa}} &= |X_{\mathrm{fus}}u_{\mathrm{a}}| + |Y_{\mathrm{fus}}v_{\mathrm{a}}| + |Z_{\mathrm{fus}}(w_{\mathrm{a}} - v_{\mathrm{i}})|.
\end{aligned}
$$

Initial values: $T(0) = mg, \quad v_{\mathrm{i}}(0) = \dfrac{mg}{2\rho\pi R^2}, \quad \hat{v}(0) = 0.$

Tail rotor forces and moments

$$T_{\mathrm{tr}} = \frac{\rho\,\Omega_{\mathrm{tr}}\,{R_{\mathrm{tr}}}^2\,C_{l\alpha,\mathrm{tr}}\,b_{\mathrm{tr}}\,c_{\mathrm{tr}}}{4}(w_{\mathrm{bl,tr}} - v_{\mathrm{i,tr}})$$

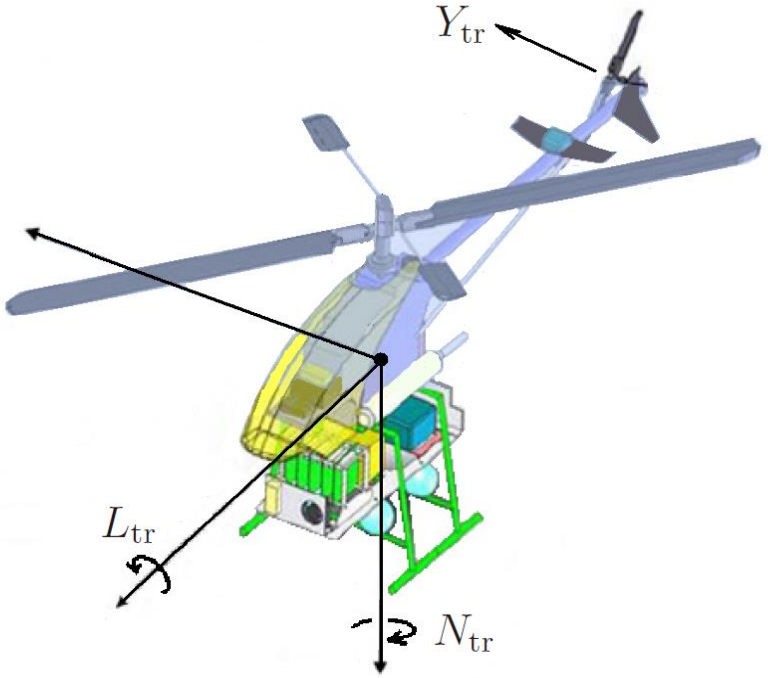
$$v_{\mathrm{i,tr}}^2 = \sqrt{\left(\frac{\hat{v}_{\mathrm{tr}}^2}{2}\right)^2 + \left(\frac{T_{\mathrm{tr}}}{2\rho\,\pi\,R_{\mathrm{tr}}^2}\right)^2} - \frac{\hat{v}_{\mathrm{tr}}^2}{2}$$

$$\hat{v}_{\mathrm{tr}}^2 = (w_{\mathrm{a}} + qD_{\mathrm{tr}})^2 + u_{\mathrm{a}}^2 + w_{\mathrm{r,tr}}(w_{\mathrm{r,tr}} - 2v_{\mathrm{i,tr}})$$

$$w_{\mathrm{r,tr}} = v_{\mathrm{a}} - rD_{\mathrm{tr}} + pH_{\mathrm{tr}}$$

$$w_{\mathrm{bl,tr}} = w_{\mathrm{r,tr}} + \frac{2}{3}\Omega_{\mathrm{tr}}R_{\mathrm{tr}}\theta_{\mathrm{ped}}$$

$$\theta_{\mathrm{ped}} = K_{\mathrm{ped}}\bar{\delta}_{\mathrm{ped}} + \theta_{\mathrm{ped},0}$$



Initial values: $$Y_{\mathrm{tr}}(0) = \frac{1}{\Omega D_{\mathrm{tr}}}\left(\frac{\rho\Omega^3R^4C_{\mathrm{D0}}b_{\mathrm{mr}}c_{\mathrm{mr}}}{8} + \frac{m^2g^2}{2\rho\pi R^2}\right), \quad v_{\mathrm{i,tr}}(0) = \frac{Y_{\mathrm{tr}}(0)}{2\rho\pi R_{\mathrm{tr}}^2}, \quad \hat{v}_{\mathrm{tr}}(0) = 0$$

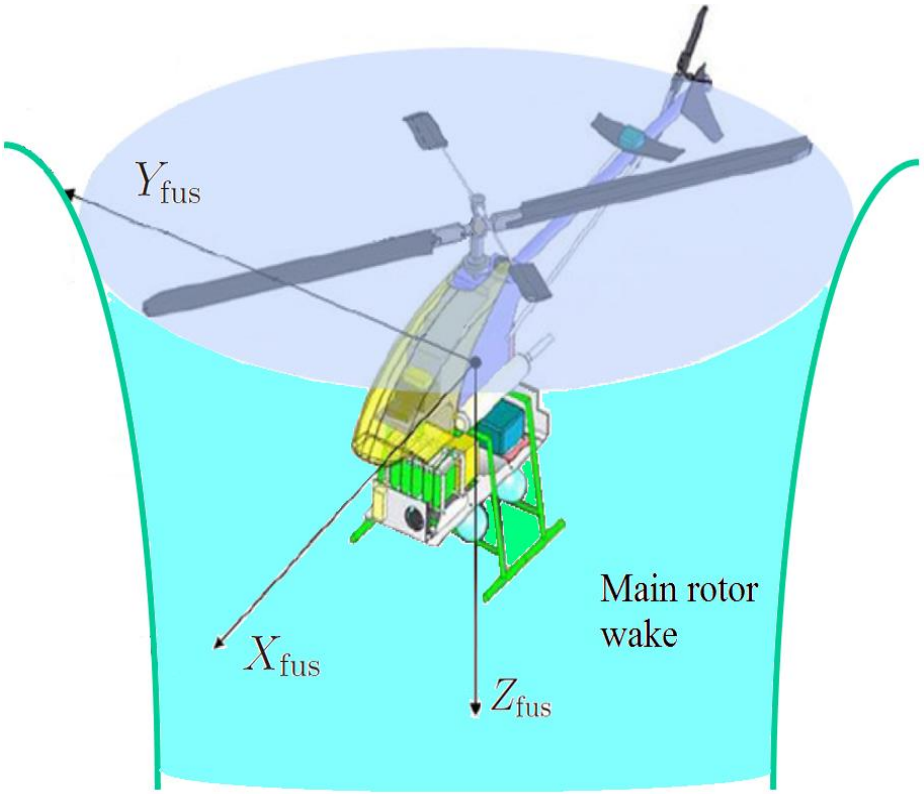
$$Y_{\mathrm{tr}} = -T_{\mathrm{tr}} \qquad L_{\mathrm{tr}} = -Y_{\mathrm{tr}}H_{\mathrm{tr}}, \quad N_{\mathrm{tr}} = Y_{\mathrm{tr}}D_{\mathrm{tr}}$$

Fuselage forces

$$X_{\text{fus}} = \begin{cases} -\frac{\rho}{2} S_{\text{fx}}\, u_{\text{a}}\, v_{\text{i,mr}}, & \text{if } |u_{\text{a}}| \leq v_{\text{i,mr}} \\ -\frac{\rho}{2} S_{\text{fx}}\, u_{\text{a}}\, |u_{\text{a}}|, & \text{if } |u_{\text{a}}| > v_{\text{i,mr}} \end{cases}$$

$$Y_{\text{fus}} = \begin{cases} -\frac{\rho}{2} S_{\text{fy}}\, v_{\text{a}}\, v_{\text{i,mr}}, & \text{if } |v_{\text{a}}| \leq v_{\text{i,mr}} \\ -\frac{\rho}{2} S_{\text{fy}}\, v_{\text{a}}\, |v_{\text{a}}|, & \text{if } |v_{\text{a}}| > v_{\text{i,mr}} \end{cases}$$

$$Z_{\text{fus}} = -\frac{\rho}{2} S_{\text{fz}}\, (w_{\text{a}} - v_{\text{i,mr}})\, |w_{\text{a}} - v_{\text{i,mr}}|$$

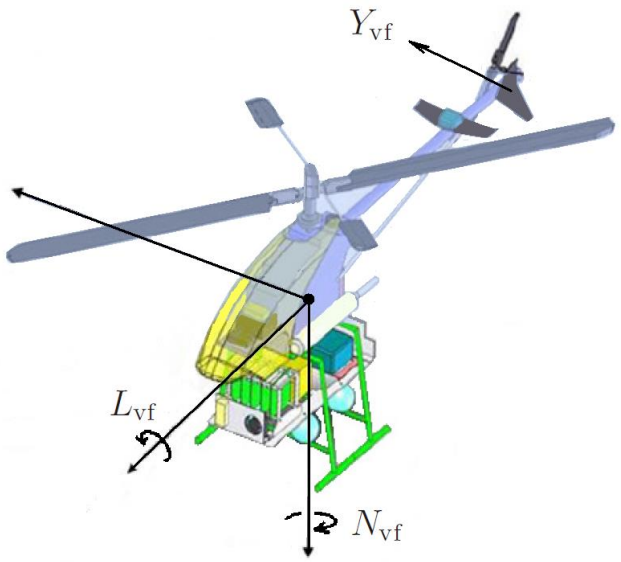


Vertical fin forces and moments

$$Y_{\mathrm{vf}} = \begin{cases} -\frac{\rho}{2} C_{\mathrm{l}\alpha,\mathrm{vf}} S_{\mathrm{vf}} v_{\mathrm{vf}} |u_{\mathrm{a}}|, & \text{if } \left|\frac{v_{\mathrm{vf}}}{u_{\mathrm{a}}}\right| \le \tan(\alpha_{\mathrm{st}}) \\ -\frac{\rho}{2} S_{\mathrm{vf}} v_{\mathrm{vf}} |v_{\mathrm{vf}}|, & \text{if } \left|\frac{v_{\mathrm{vf}}}{u_{\mathrm{a}}}\right| > \tan(\alpha_{\mathrm{st}}) \text{ (stalled)} \end{cases}$$

where $v_{\mathrm{vf}} = v_{\mathrm{a}} + v_{\mathrm{i,tr}} - r D_{\mathrm{vf}}$

$$L_{\mathrm{vf}} = Y_{\mathrm{vf}} H_{\mathrm{vf}}, \quad N_{\mathrm{vf}} = Y_{\mathrm{vf}} D_{\mathrm{vf}}$$

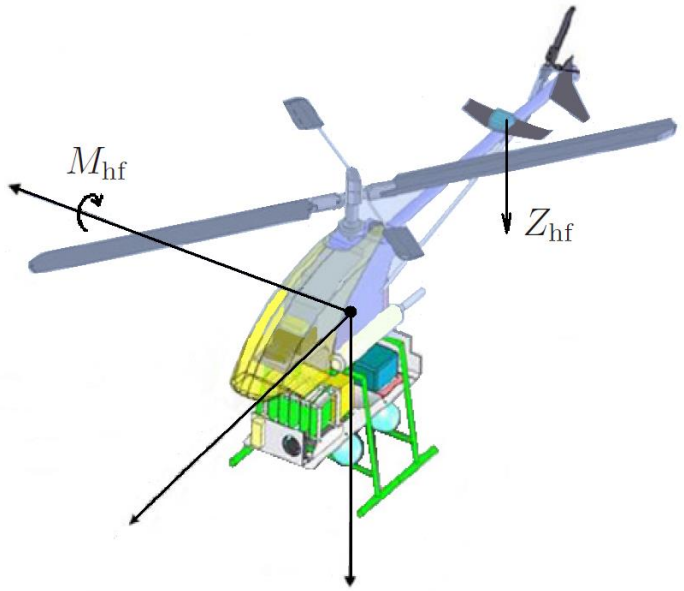


Horizontal fin forces and moments

$$Z_{\mathrm{hf}} = \begin{cases} -\frac{\rho}{2} C_{\mathrm{l}\alpha,\mathrm{hf}} S_{\mathrm{hf}} w_{\mathrm{hf}} |u_{\mathrm{a}}|, & \text{if } \left|\frac{w_{\mathrm{hf}}}{u_{\mathrm{a}}}\right| \le \tan(\alpha_{\mathrm{st}}) \\ -\frac{\rho}{2} S_{\mathrm{hf}} w_{\mathrm{hf}} |w_{\mathrm{hf}}|, & \text{if } \left|\frac{w_{\mathrm{hf}}}{u_{\mathrm{a}}}\right| > \tan(\alpha_{\mathrm{st}}) \text{ (stalled)} \end{cases}$$

where $w_{\mathrm{hf}} = w_{\mathrm{a}} + q D_{\mathrm{hf}} - v_{\mathrm{i,mr}}$

$$M_{\mathrm{hf}} = Z_{\mathrm{hf}} H_{\mathrm{hf}}$$



First-principles Modeling Approach

Single-rotor helicopter

Direct measurement

Parameter	Physical meaning
$I_{mr} = 0.055$ kg·m^2	Main blade inertia w.r.t. rotor hub
$I_{sb} = 0.004$ kg·m^2	Stabilizer bar inertia w.r.t. rotor hub
$R = 0.7$ m	Main rotor radius
$R_{sb,in} = 0.231$ m	Stabilizer bar inner radius
$R_{sb,out} = 0.312$ m	Stabilizer bar outer radius
$R_{tr} = 0.128$ m	Tail rotor radius
$S_{fx} = 0.103$ m^2	Effective longitudinal fuselage drag area
$S_{fy} = 0.900$ m^2	Effective lateral fuselage drag area
$S_{fz} = 0.084$ m^2	Effective vertical fuselage drag area
$S_{hf} = 0.011$ m^2	Horizontal fin area
$S_{vf} = 0.007$ m^2	Vertical fin area
$b = 2$	Main blade number
$b_{tr} = 2$	Tail blade number
$c = 0.062$ m	Main blade chord length
$c_{sb} = 0.059$ m	Stabilizer bar chord length
$c_{tr} = 0.029$ m	Tail rotor chord length
$g = 9.781$ N·kg^{-1}	Acceleration of gravity
$m = 9.750$ kg	Helicopter mass
$n_{tr} = 4.650$	Gear ratio of the tail rotor to the main rotor
$\rho = 1.290$ kg/m^3	Air density

Ground Tests

Parameter	Physical meaning	
$D_{hf} = 0.751$ m	Horizontal fin location behind the CG	*CG Location*
$D_{tr} = 1.035$ m	Tail rotor hub location behind the CG	
$D_{vf} = 0.984$ m	Vertical fin location behind the CG	
$H_{mr} = 0.337$ m	Main rotor hub location above the CG	
$H_{tr} = 0.172$ m	Tail rotor hub location above the CG	
$H_{vf} = 0.184$ m	Vertical fin location above the CG	
$A_{lon} = 0.210$ rad	Direct linkage gain from δ_{lon} to main blade deflection	*Airfoil deflection*
$B_{lat} = 0.200$ rad	Direct linkage gain form δ_{lat} to main blade deflection	
$C_{lon} = 0.560$ rad	Linkage gain from δ_{lon} to stabilizer bar deflection	
$D_{lat} = 0.570$ rad	Linkage gain from δ_{lat} to stabilizer bar deflection	
$K_{sb} = 1$	Ratio of main blade deflection to stabilizer bar TPP titling angle	
$I_{xx} = 0.249$ kg·m^2	Rolling moment of inertia	*Inertia*
$I_{yy} = 0.548$ kg·m^2	Pitching moment of inertia	
$I_{zz} = 0.787$ kg·m^2	Yawing moment of inertia	
$K_I K_a = 8.499$ rad	Product of integral gain K_I and scaling factor K_a	*Collective Pitch curve*
$K_P K_a = 1.608$ rad	Product of proportional gain K_P and scaling factor K_a	
$K_{col} = -0.165$ rad	Ratio of θ_{col} to δ_{col}	
$K_{ped} = 1$	Ratio of θ_{ped} to δ_{ped}	
$\theta_{col,0} = 0.075$ rad	θ_{col} value when δ_{col} is zero	
$\theta_{ped,0} = 0.143$ rad	θ_{ped} value when δ_{col} is zero	

First-principles Modeling Approach

Single-rotor helicopter

NUS National University of Singapore

Wind Tunnel Data

Parameter	Physical meaning
$C_{l\alpha,\mathrm{hf}} = 2.85\ \mathrm{rad}^{-1}$	Horizontal fin lift curve slope
$\hat{C}_{l\alpha,\mathrm{mr}} = 5.71\ \mathrm{rad}^{-1}$	Main blade lift curve slope
$\hat{C}_{l\alpha,\mathrm{sb}} = 2.23\ \mathrm{rad}^{-1}$	Stabilizer bar lift curve slope
$\hat{C}_{l\alpha,\mathrm{tr}} = 2.23\ \mathrm{rad}^{-1}$	Tail blade lift curve slope
$C_{l\alpha,\mathrm{vf}} = 2.85\ \mathrm{rad}^{-1}$	Vertical fin lift curve slope

Actual Flight Tests

Parameters	Physical meaning
$A_{\mathrm{b}_{\mathrm{s}}} = 9.720\ \mathrm{sec}^{-1}$	Coupling effect
$B_{\mathrm{a}_{\mathrm{s}}} = 10.704\ \mathrm{sec}^{-1}$	Coupling effect
$K_{\mathrm{I}} = 2.2076$	Integral gain of the embedded PI controller
$K_{\mathrm{P}} = 0.4177$	Proportional gain of the embedded PI controller
$K_{\mathrm{a}} = -3.85\ \mathrm{rad}$	Scaling factor of the amplifier circuit
$K_{\beta} = 112.84\ \mathrm{N{\cdot}m}$	Main rotor spring constant
$C_{l\alpha,\mathrm{mr}} = 5.73\ \mathrm{rad}^{-1}$	Main blade lift curve slope
$C_{l\alpha,\mathrm{sb}} = 2.13\ \mathrm{rad}^{-1}$	Stabilizer bar lift curve slope
$C_{l\alpha,\mathrm{tr}} = 2.81\ \mathrm{rad}^{-1}$	Tail blade lift curve slope
$\Omega_{\mathrm{mr}} = 193.73\ \mathrm{rad}$	Main rotor rotating speed
$\Omega_{\mathrm{tr}} = 900.85\ \mathrm{rad}$	Main rotor rotating speed

UAV Flight Dynamics Modeling

How do we obtain a **flight dynamics model**?

1) **First-principles Modeling Approach**
   - Build mathematical model by manipulating equations of physics
   - Complex, nonlinear, rigorous with physical meanings
   - Test-bench experiments
2) **System Identification Approach**
   - Build mathematical model by fitting measurement data
   - Simplified, linear, black-box
   - Flight test experiments
3) **Combination of the above two approaches**

System Identification Approach

Steps

System Identification Approach is suitable for obtaining linear dynamics model and can be conducted in both time-domain and frequency-domain.

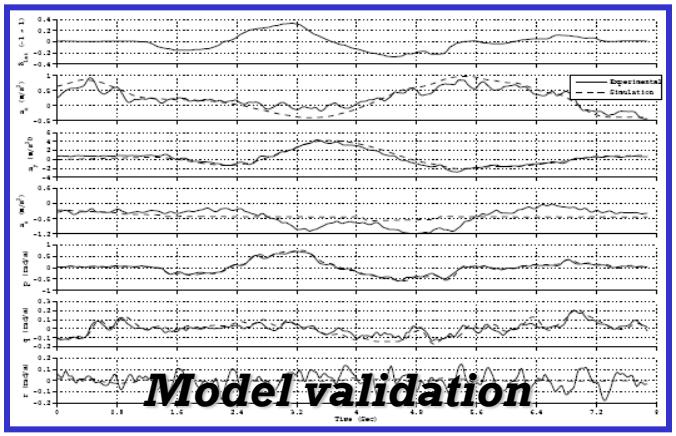
$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

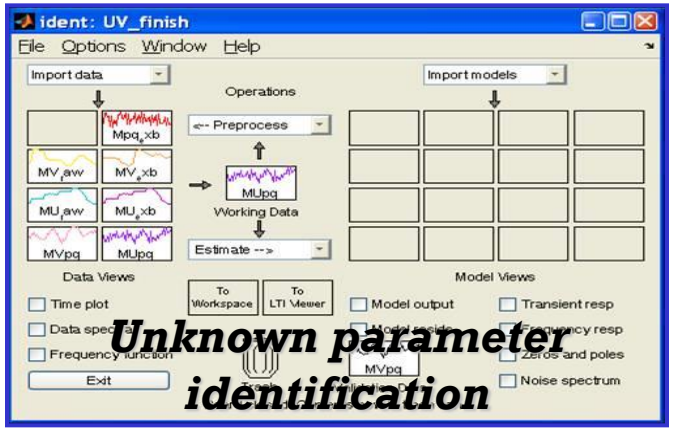
$$\mathbf{x} = [u \quad v \quad p \quad q \quad \phi \quad \theta \quad a_s \quad b_s \quad w \quad r \quad r_{fb}]',$$

$$\mathbf{u} = [\delta_{lat} \quad \delta_{lon} \quad \delta_{col} \quad \delta_{ped}]',$$

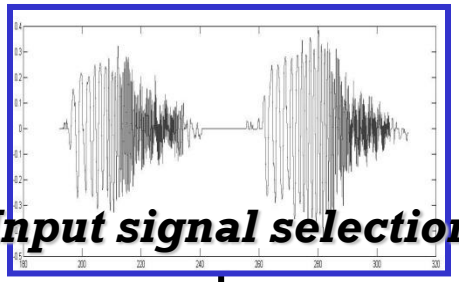
$$A = \begin{bmatrix} X_u & 0 & 0 & 0 & -g & X_{a_s} & 0 & 0 & 0 & 0 \\ 0 & Y_v & 0 & 0 & g & 0 & 0 & Y_{b_s} & 0 & 0 & 0 \\ L_u & L_v & 0 & 0 & 0 & 0 & L_{a_s} & L_{b_s} & 0 & 0 & 0 \\ M_u & M_v & 0 & 0 & 0 & 0 & M_{a_s} & M_{b_s} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1/\tau & A_{b_s} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & A_{b_s} & -1/\tau & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Z_w & 0 & 0 \\ 0 & N_v & 0 & 0 & 0 & 0 & 0 & 0 & N_r & N_{rfb} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1/\tau \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A_{lat} & A_{lon} & 0 & 0 \\ B_{lat} & B_{lon} & 0 & 0 \\ 0 & 0 & Z_{col} & 0 \\ 0 & 0 & 0 & N_{ped} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Model structure determination



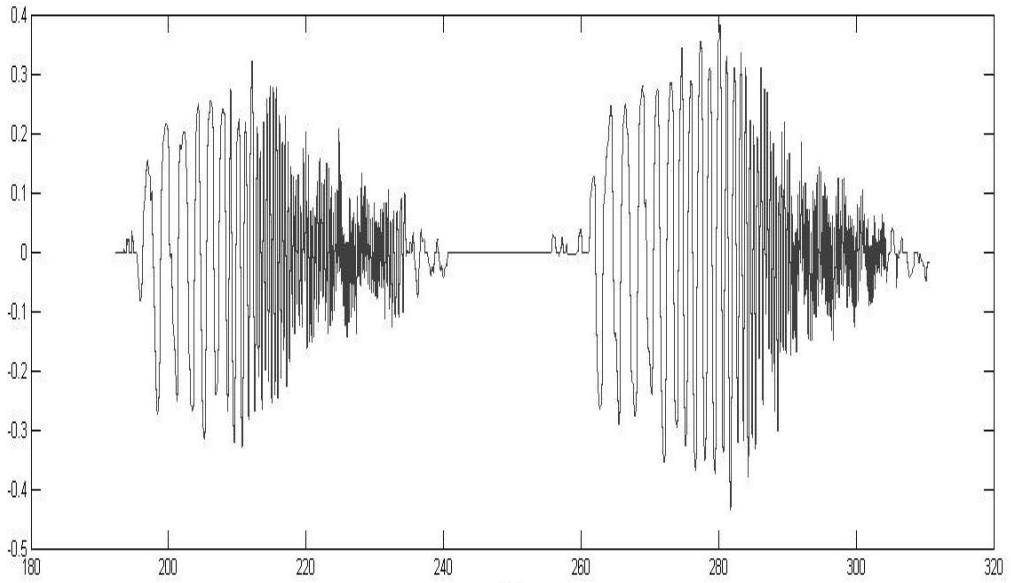


Step 1: Data collection and preprocessing:



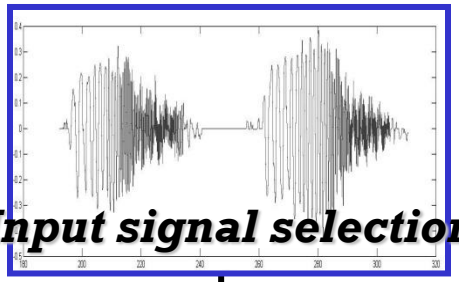
Input signal selection

Data preprocessing



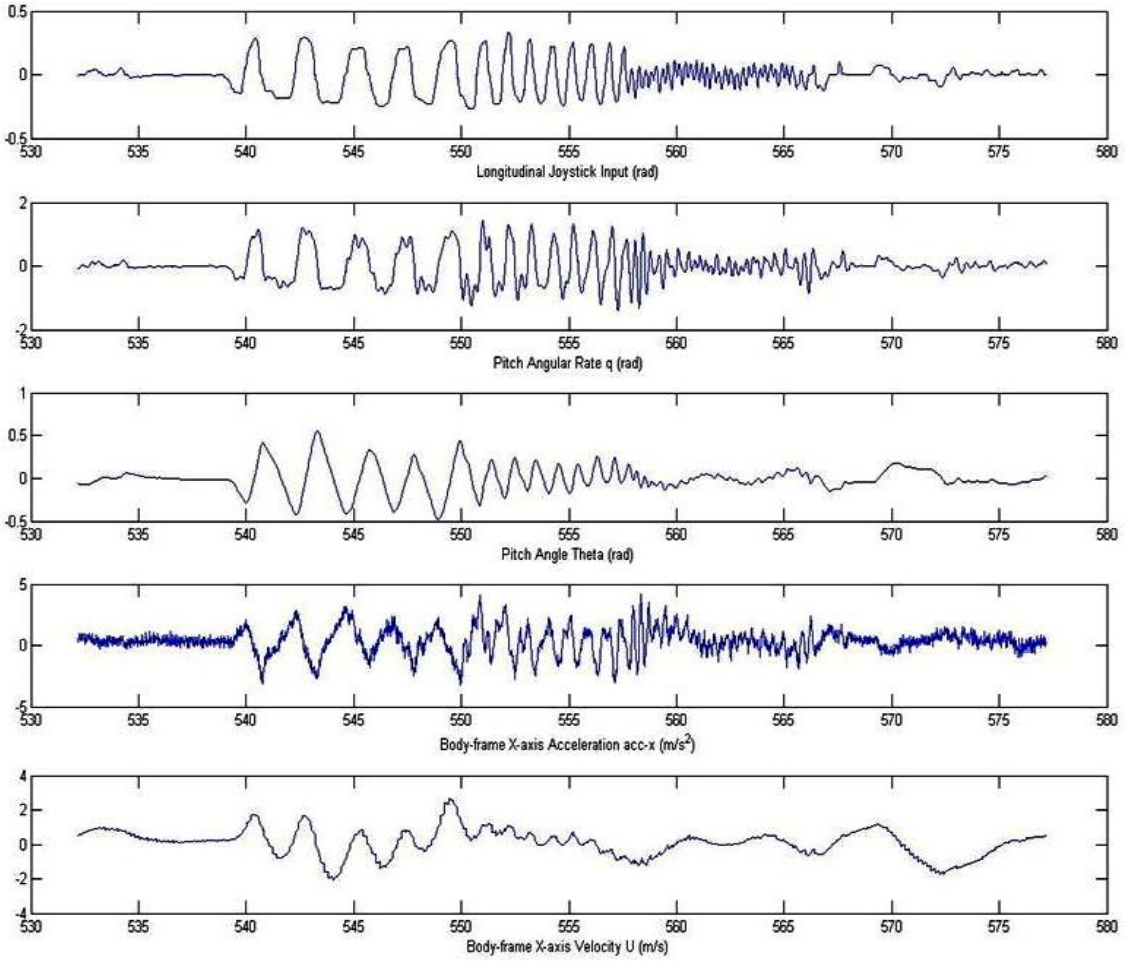
Frequency sweep input signal

Step 1: Data collection and preprocessing:



Input signal selection

Data preprocessing

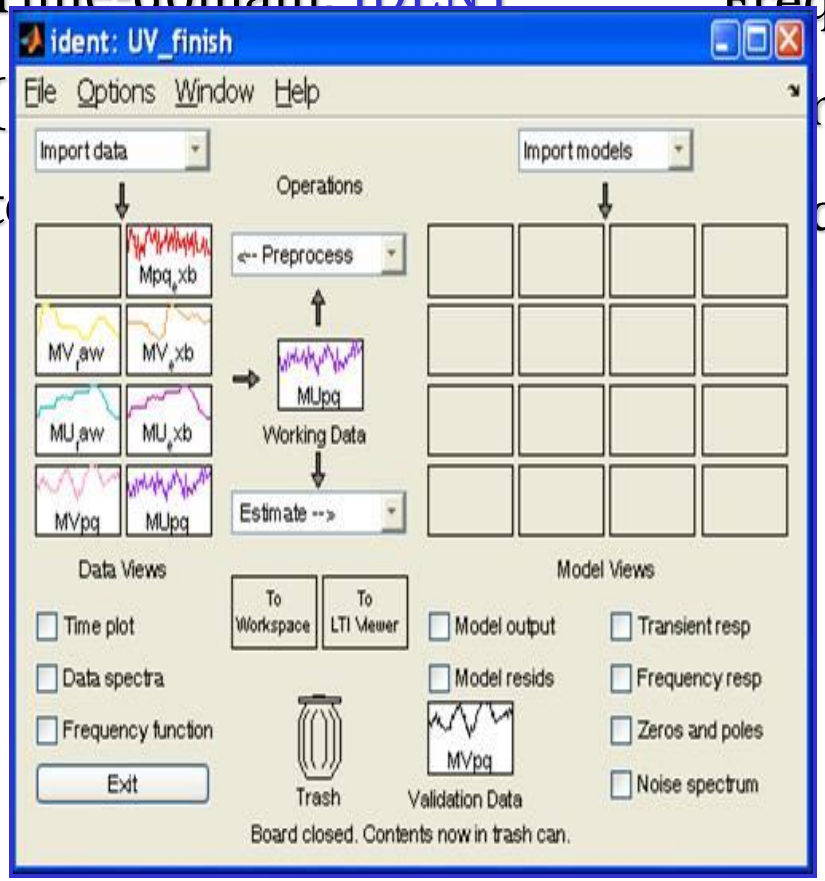


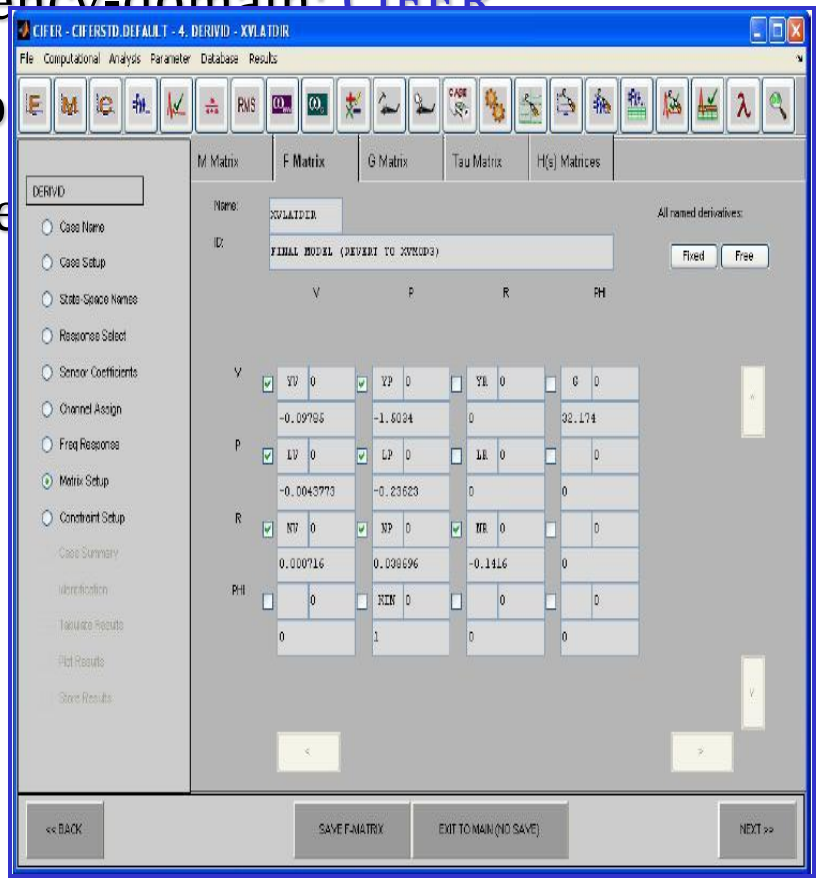
Output responses

Toolkits:

Time-domain: IDENT

Frequency-domain: CIFER

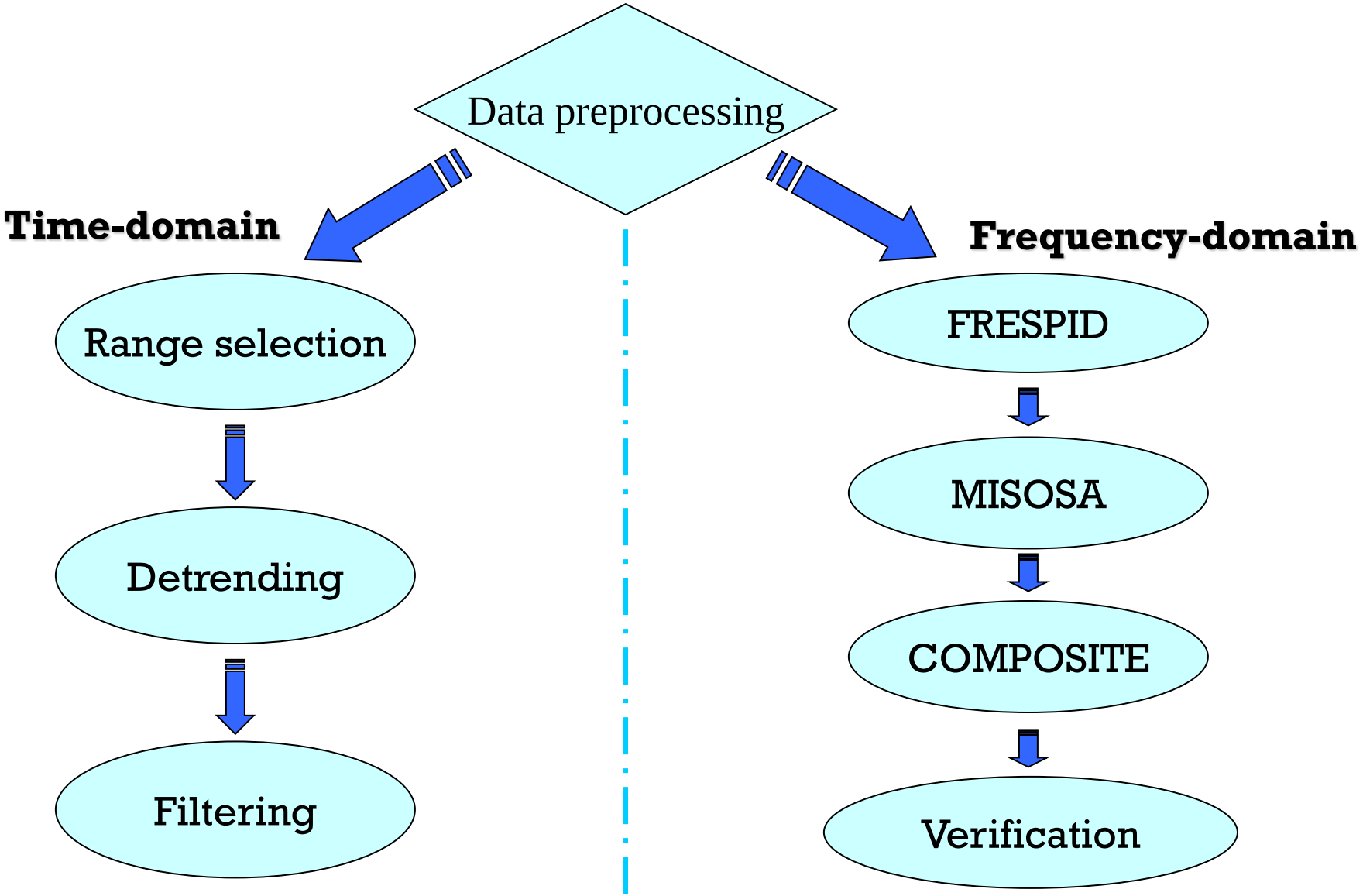




IDENT **CIFER**

System Identification Approach

Case Study: single-rotor helicopter



Step 2: Model structure determination:

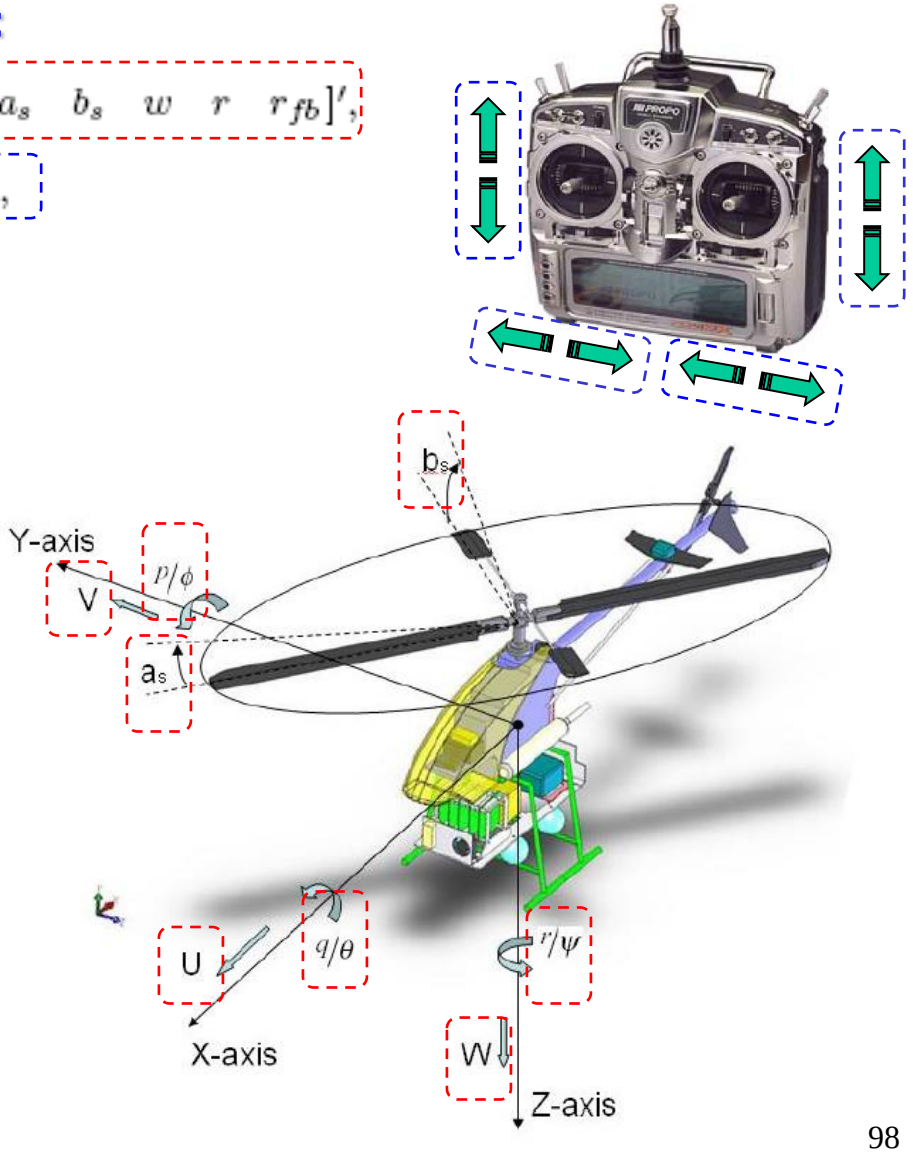
$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u}$$

$$\mathbf{x} = [u \quad v \quad p \quad q \quad \phi \quad \theta \quad a_s \quad b_s \quad w \quad r \quad r_{fb}]',$$

$$\mathbf{u} = [\delta_{lat} \quad \delta_{lon} \quad \delta_{col} \quad \delta_{ped}]',$$

$$A = \begin{bmatrix} X_u & 0 & 0 & 0 & 0 & -g & X_{a_s} & 0 & 0 & 0 & 0 \\ 0 & Yv & 0 & 0 & g & 0 & 0 & Y_{b_s} & 0 & 0 & 0 \\ L_u & L_v & 0 & 0 & 0 & 0 & L_{a_s} & L_{b_s} & 0 & 0 & 0 \\ M_u & M_v & 0 & 0 & 0 & 0 & M_{a_s} & M_{b_s} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1/\tau & A_{b_s} & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & B_{a_s} & -1/\tau & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & Z_w & Z_r & 0 \\ 0 & N_v & 0 & 0 & 0 & 0 & 0 & 0 & 0 & N_r & N_{rfb} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & K_r & K_{rfb} \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A_{lat} & A_{lon} & 0 & 0 \\ B_{lat} & B_{lon} & 0 & 0 \\ 0 & 0 & Z_{col} & 0 \\ 0 & 0 & 0 & N_{ped} \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$



Step 3: Unknown parameter identification:

1). Angular rate dynamics; 2). Horizontal velocity dynamics;

3). Yaw dynamics; 4). Heave dynamics;

a. Time-domain: Hover model of a single-rotor helicopter

$$A=\begin{bmatrix}
-0.1778 & 0 & 0 & 0 & 0 & -9.7807 & -9.7807 & 0 & 0 & 0 & 0\\
0 & -0.3104 & 0 & 0 & 9.7807 & 0 & 0 & 9.7807 & 0 & 0 & 0\\
-0.3326 & -0.5353 & 0 & 0 & 0 & 0 & 75.764 & 343.86 & 0 & 0 & 0\\
-0.1903 & -0.2490 & 0 & 0 & 0 & 0 & 172.62 & -59.958 & 0 & 0 & 0\\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & -1 & 0 & 0 & -8.1222 & 4.6535 & 0 & 0 & 0\\
0 & 0 & -1 & 0 & 0 & 0 & -0.0921 & -8.1222 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.6821 & -0.0535 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -0.2892 & -5.5561 & -36.674\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2.7492 & -11.112
\end{bmatrix},$$

$$B=\begin{bmatrix}
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0.0496 & 2.6224 & 0 & 0\\
2.4928 & 0.1741 & 0 & 0\\
0 & 0 & 7.8246 & 0\\
0 & 0 & 1.6349 & -58.4053\\
0 & 0 & 0 & 0
\end{bmatrix},$$

b. Frequency-domain: Hovering model of a single-rotor helicopter

Extra metrics for parameter accuracy evaluation:

1). Cramer-Rao bound (CR% < 20%); 2). Insensitivity (I% < 10%)

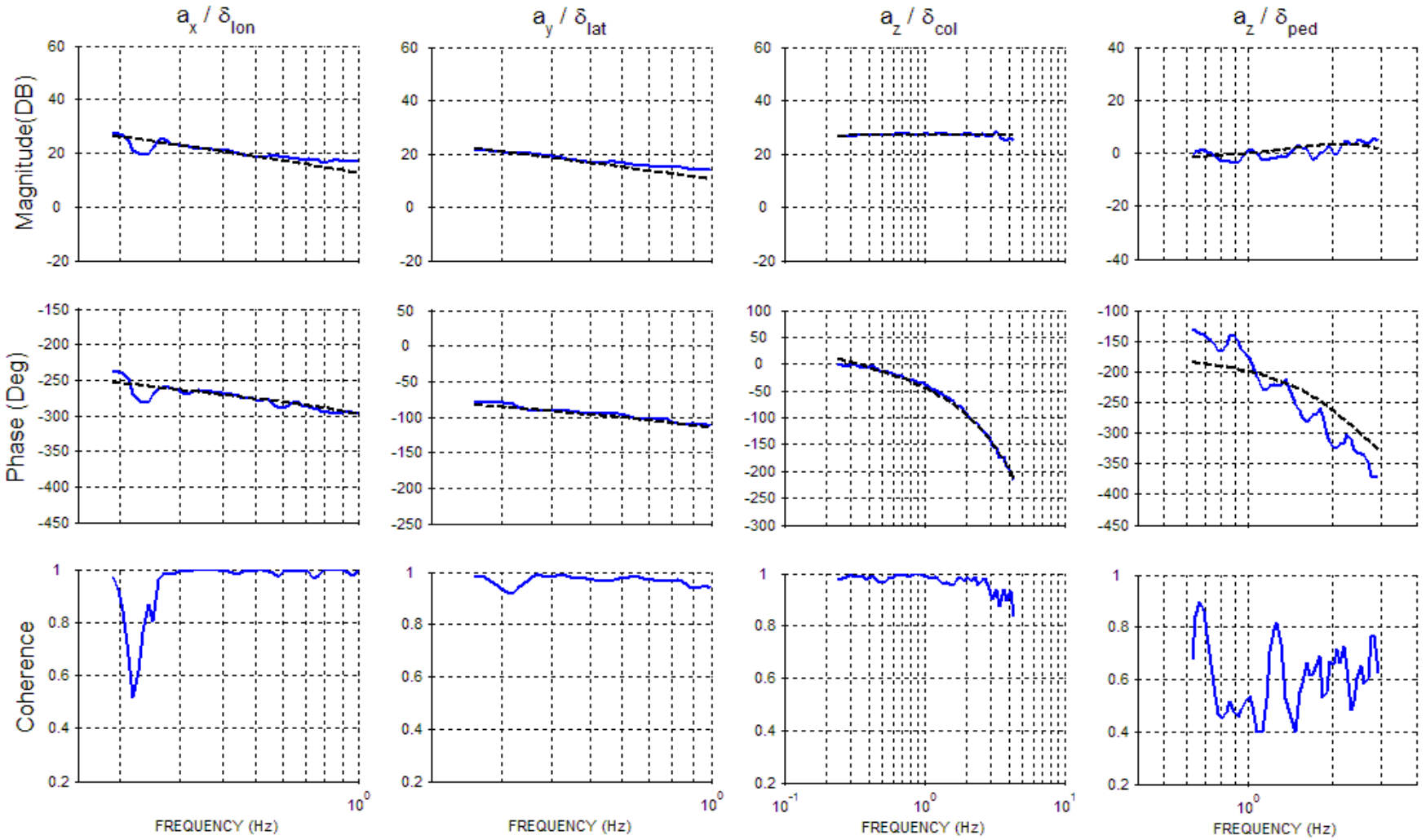
Parameter	Value	CR%	I%
X_u	-0.5198	20.31	9.827
X_{a_s}	-9.7807 !	——	——
Y_v	-0.4225	19.51	9.231
Y_{b_s}	9.7807 !	——	——
L_u	-0.3262 !	——	——
L_v	-0.0675 *	——	——
L_{a_s}	0 +	——	——
L_{b_s}	439.4000	2.489	1.020
M_u	-0.0814 *	——	——
M_v	-2.1210	18.02	7.143
M_{a_s}	275.1000	2.794	1.067
M_{b_s}	-42.1100	14.17	4.562
τ	-4.7790	7.678	2.367
a_{b_s}	2.3850	13.67	4.309
b_{a_s}	0	——	——
Z_w	-0.5996	19.33	12.13
Z_r	0.1103	6.369	2.514
N_r	-4.6150	7.515	2.441
N_{rfb}	154.5000 *	——	——
N_v	0	——	——
K_r	-1.0830	5.805	1.929
K_{rfb}	-9.2300 *	——	——
A_{lat}	0.0610	16.62	8.979
A_{lon}	2.8080	2.731	1.319
B_{lat}	2.1810	2.687	1.202
B_{lon}	0 +	——	——
Z_{col}	22.9400	3.776	1.856
N_{ped}	-154.5000	6.663	1.833

+ : Eliminated in model structure determination

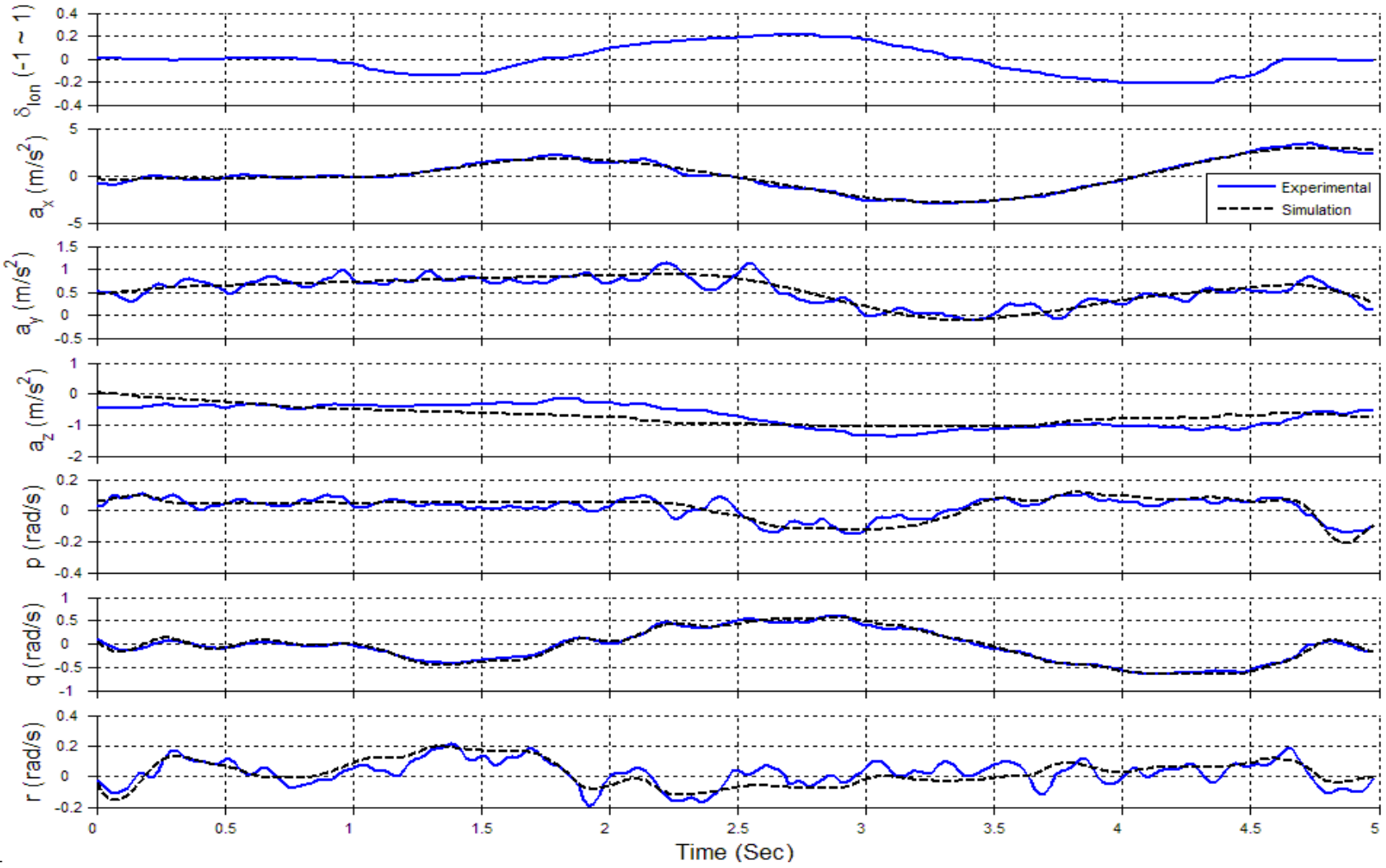
! : Fixed empirically

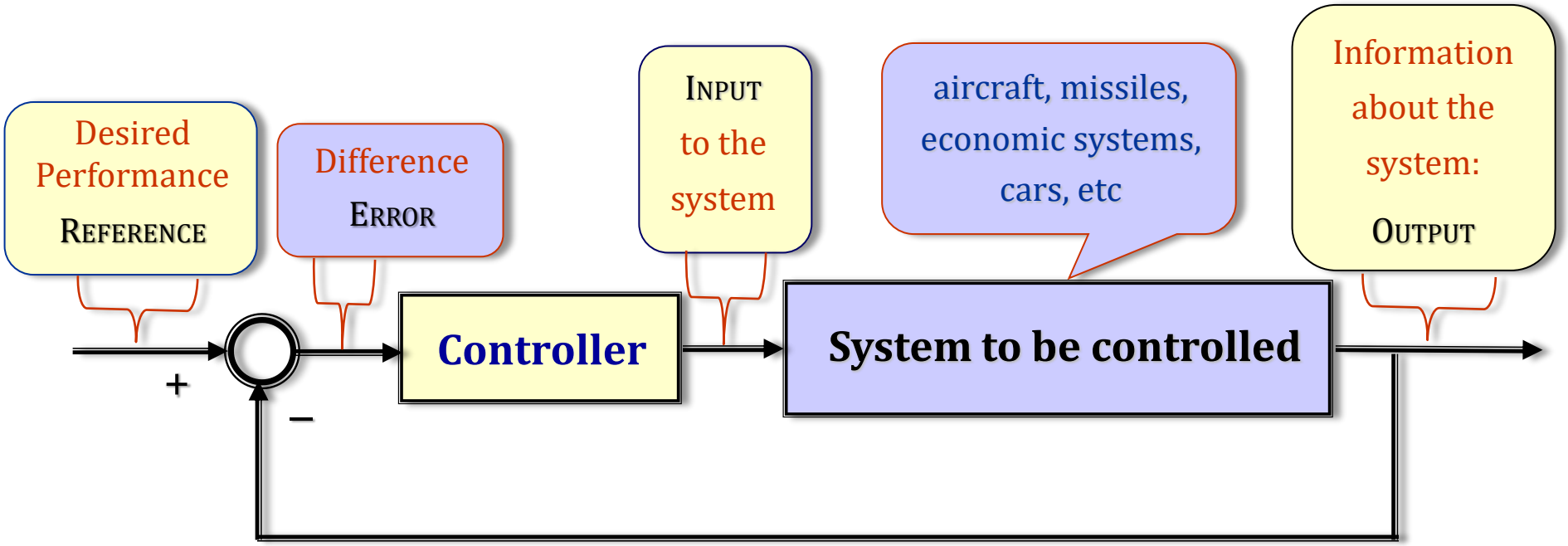
* : Tied to other parameters

Frequency responses matching



Step 4: Model validation



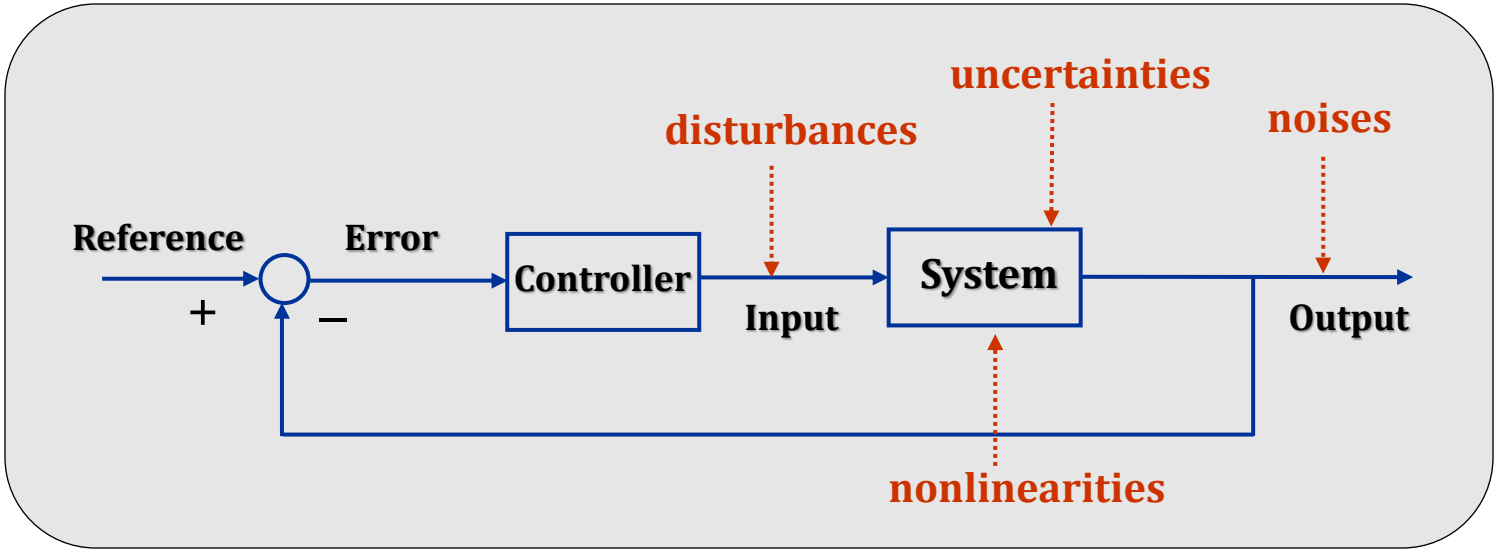


Objective: To make the system **OUTPUT** and the desired **REFERENCE** as close as possible, i.e., to make the **ERROR** as small as possible.

Key Issues: (1) How to describe the system to be controlled? **(Modeling)**

(2) How to design the controller? **(Control)**

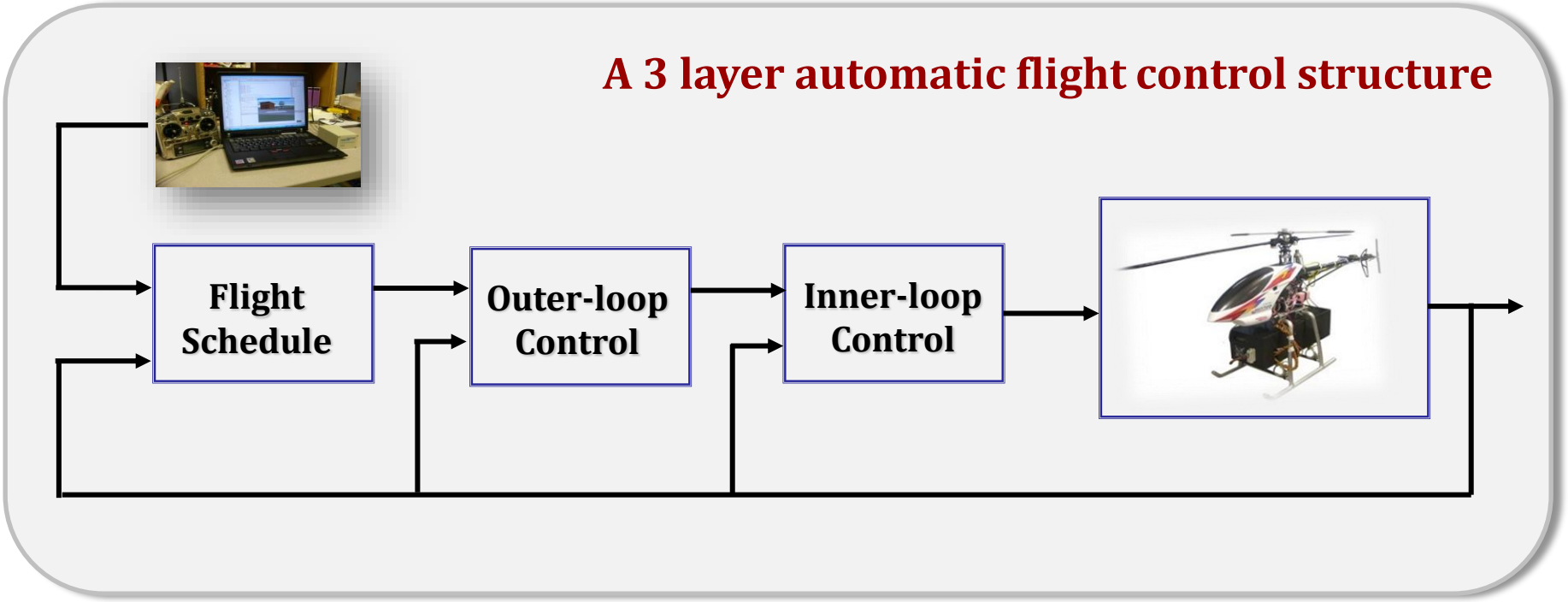
There are many other factors of life have to be carefully considered when designing a control system for real-life problems. These factors include:



These real-life factors, i.e., disturbances, uncertainties, nonlinearities, give birth of many modern control techniques…

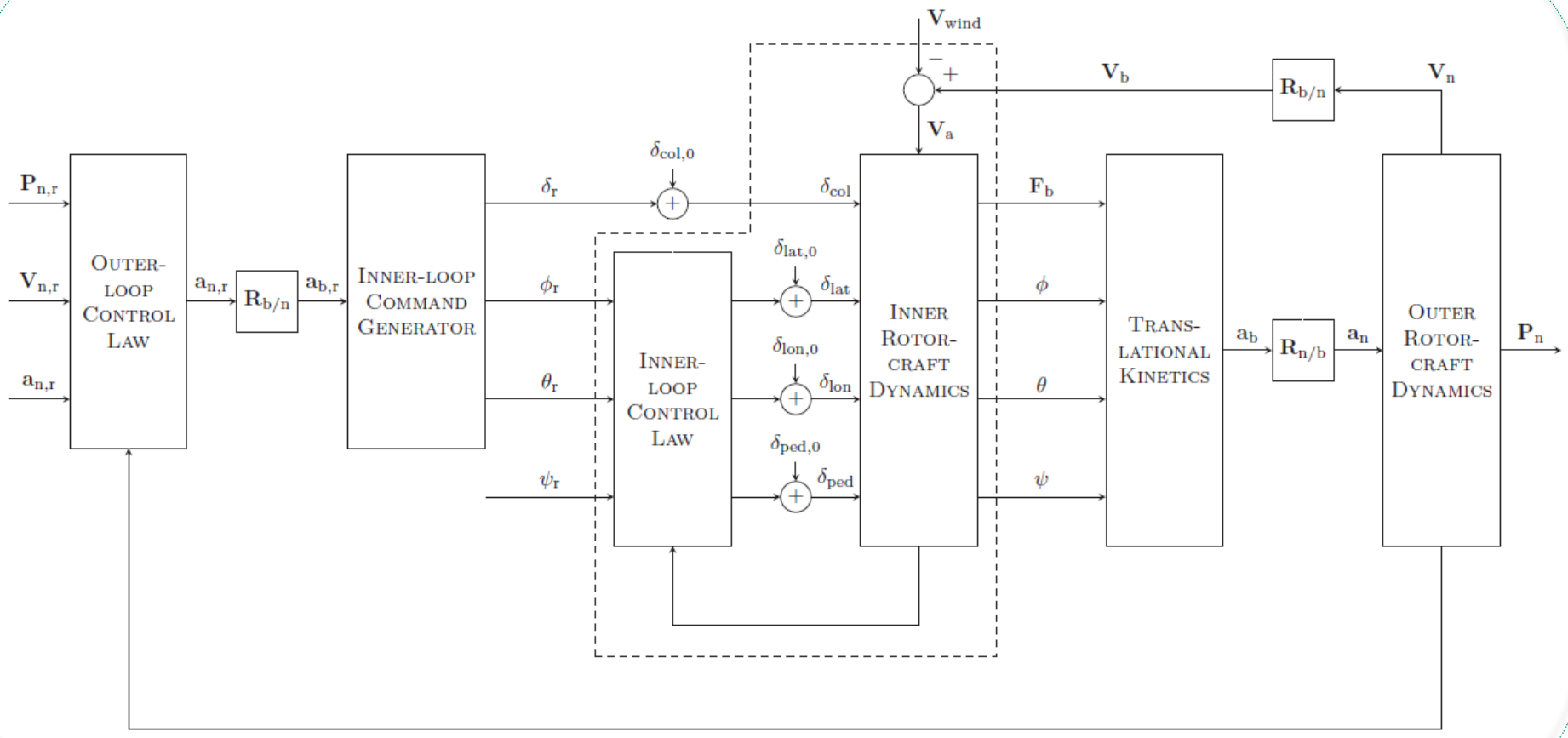
The following is my personal view on the clarification of control techniques:

- Classical control: PID control, developed in 1940s and utilized heavily for in industrial processes. Examples: everywhere in life…
- Optimal control: Linear quadratic regulator (LQR), H_2 optimal control, Kalman filter, developed in 1960s to achieve certain optimal performance and boomed by *NASA Apollo Project*.
- Robust control: H_∞ control, developed in 1980s and 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., researched heavily in 1990s, developed to handle systems with unknown models. Examples: economic systems, social systems...



- Inner loop control is to guarantee the stability of the aircraft attitude
- Outer loop control is to control the aircraft position
- Flight schedule is to generate references for flight missions

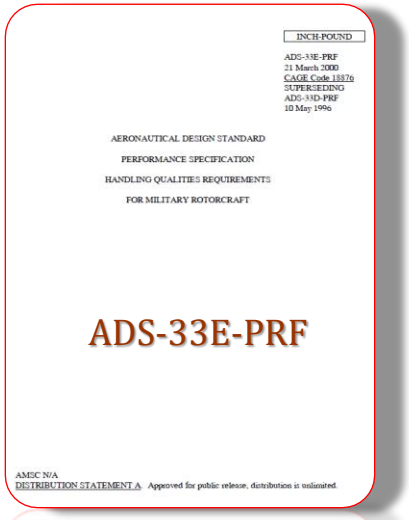
Detailed structure of inner- and outer-loop control



Inner-loop control is to stabilize the overall aircraft and to control its attitude

We adopt the design specifications set by the USA military organization for military rotorcraft, which place strict requirements on:

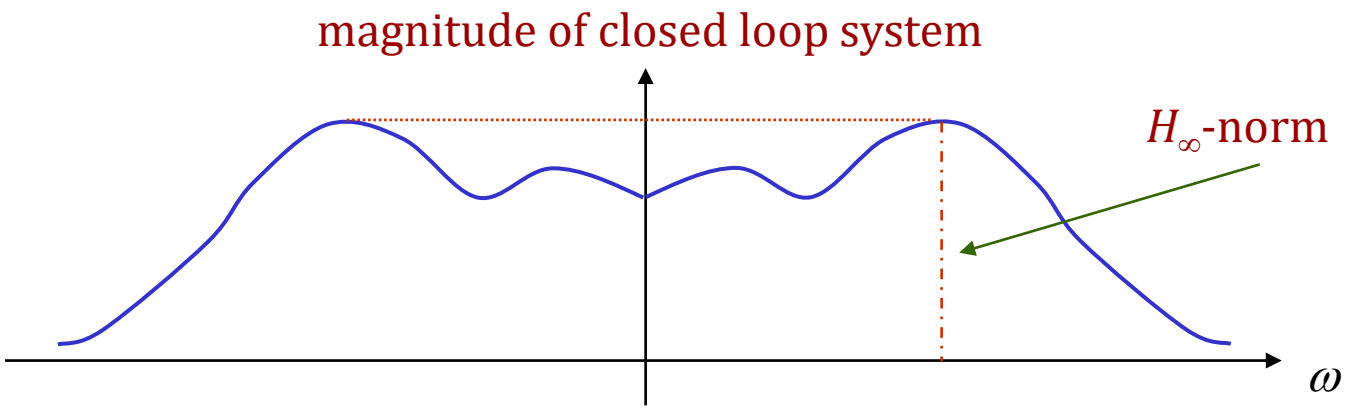
- closed-loop stability
- bandwidth of pitch and roll attitude response
- coupling effect between roll and pitch channels
- coupling effect from heave control to yaw response
- disturbance rejection
- quickness of pitch, roll and yaw responses
- attitude hold for spike disturbance input

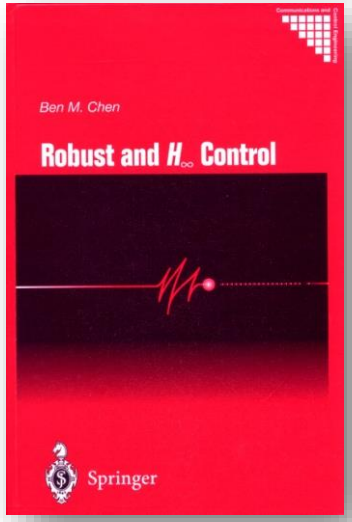


We aim to achieve **Level 1 performance** in all specifications in accordance with the military standards set by U.S. Army Aviation (ADS-33E-PRF).

Control technique adopted for the inner loop

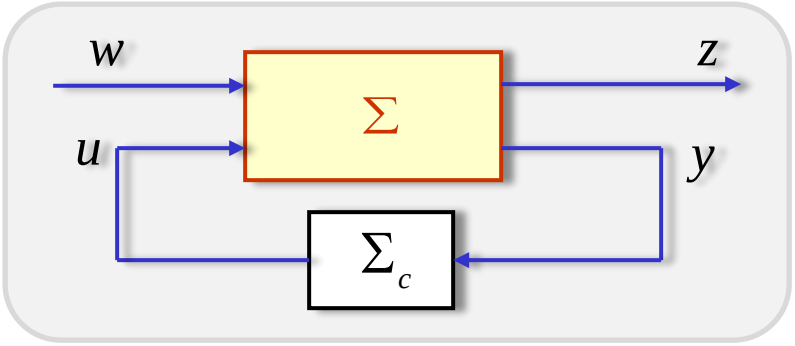
For the inner loop, we treat wind gusts as disturbance entering the system and then formulate the overall problem as an H_∞ control problem, which is to design a control law such that when it is applied to the given system, the resulting closed-loop system is stable and the effect of the disturbance to the output to be controlled (the position of the aircraft in our case) is minimized in H_∞ sense:





H_∞ optimization is closely and commonly related to ***robust control***…

Consider a stabilizable and detectable linear time-invariant system Σ with a proper controller Σ_c



where

$$\Sigma: \begin{cases} \dot{x} = A\,x + B\,u + E\,w \\ y = C_1\,x + \quad + D_1\,w \\ z = C_2\,x + D_2\,u \end{cases} \qquad \Sigma_c: \begin{cases} \dot{v} = A_{\text{cmp}}\,v + B_{\text{cmp}}\,y \\ u = C_{\text{cmp}}\,v + D_{\text{cmp}}\,y \end{cases}$$

$$\begin{cases} x \Leftrightarrow \text{state variable} \\ y \Leftrightarrow \text{measurement} \\ z \Leftrightarrow \text{controlled output} \end{cases} \qquad \begin{cases} u \Leftrightarrow \text{control input} \\ w \Leftrightarrow \text{disturbance} \\ v \Leftrightarrow \text{controller state} \end{cases}$$

The problem of H_∞ control is to design a control law Σ_c such that when it is applied to the given plant with disturbance, i.e., Σ, we have

- The resulting closed loop system is internally stable (this is necessary for any control system design).
- The H_∞-norm of the resulting closed-loop transfer function from the disturbance to the controlled output is as small as possible, i.e., the effect of the disturbance on the controlled output is minimized.

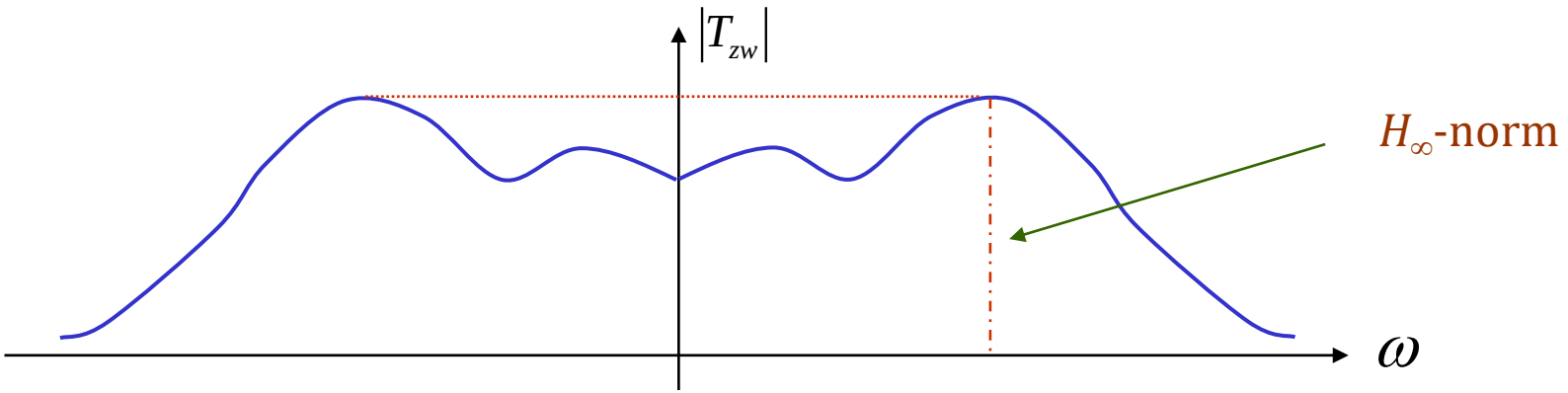
Note: A transfer function is a function of frequencies ranging from 0 to ∞. It is hard to tell if it is large or small. The common practice is to measure its norms instead. H_2-norm and ***H_∞*-norm** are two commonly used norms in measuring the sizes of a transfer function.

Interpretation of H_∞ norm

Given a stable and proper transfer function $T_{zw}(s)$, its H_∞-norm is defined as

$$\|T_{zw}\|_\infty = \sup_{0\le\omega<\infty} \sigma_{\max}\left[T_{zw}(j\omega)\right]$$

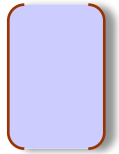
where $\sigma_{\max}[T_{zw}(j\omega)]$ denotes the maximum singular value of $T_{zw}(j\omega)$. For a SISO transfer function $T_{zw}(s)$, it is equivalent to the magnitude of $T_{zw}(j\omega)$. Graphically,



Note: The H_∞-norm is the worst case gain in $T_{zw}(s)$. Thus, minimization of the H_∞-norm of $T_{zw}(s)$ is equivalent to the minimization of the worst case (gain) situation on the effect from the disturbance w to the controlled output z.

H_∞ control: regular vs. singular cases

Most results in H_∞ control deal with a so-called a regular problem or regular case because it is simple. An H_∞ control problem is said to be **regular** if the following conditions are satisfied,

1. D_2 is of maximal column rank, i.e., D_2 is a tall and full rank matrix 
2. The subsystem (A,B,C_2,D_2) has no invariant zeros on the imaginary axis;
3. D_1 is of maximal row rank, i.e., D_1 is a fat and full rank matrix 
4. The subsystem (A,E,C_1,D_1) has no invariant zeros on the imaginary axis.

An H_∞ control problem is said to be **singular** if it is not regular, i.e., at least one of the above 4 conditions is not satisfied.

Solution to regular H_∞ state feedback problem

Given $\gamma > \gamma_\infty^*$ (see the note below), solve the following algebraic Riccati equation

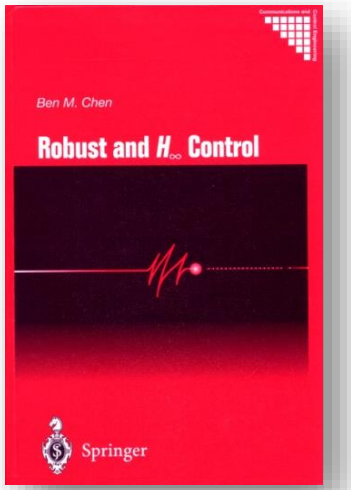
$$A^{\mathrm{T}}P + PA + C_2^{\mathrm{T}}C_2 + PEE^{\mathrm{T}}P/\gamma^2 - (PB + C_2^{\mathrm{T}}D_2)\left(D_2^{\mathrm{T}}D_2\right)^{-1}(D_2^{\mathrm{T}}C_2 + B^{\mathrm{T}}P) = 0$$

for a unique positive semi-definite solution $P \geq 0$. The H_∞ state feedback law is then given by

$$u = F\,x = -(D_2^{\mathrm{T}}D_2)^{-1}\left(D_2^{\mathrm{T}}C_2 + B^{\mathrm{T}}P\right)x$$

The resulting closed-loop system $T_{zw}(s)$ has the following property: $\|T_{zw}\|_\infty < \gamma$.

Note: The computation of the best achievable H_∞ attenuation level, γ_∞^*, is very complicated. For certain cases, γ_∞^* can be computed exactly. Generally, γ_∞^* can only be obtained using some iterative algorithms. One method is to keep solving the Riccati equation for different values of γ until it hits γ_∞^* for which and any $\gamma < \gamma_\infty^*$, the Riccati equation does not have a solution. See Chen (2000) for details.



Solution to singular H_∞ state feedback problem

Step 1: Given a $\gamma > \gamma_\infty^*$, choose $\varepsilon = 1$.

Step 2: Define the corresponding $\tilde{C}_2$ and $\tilde{D}_2$

$$\tilde{C}_2 := \begin{bmatrix} C_2 \\ \varepsilon I \\ 0 \end{bmatrix} \quad \text{and} \quad \tilde{D}_2 := \begin{bmatrix} D_2 \\ 0 \\ \varepsilon I \end{bmatrix}$$

Step 3: Solve the following Riccati equation for $\tilde{P}$:

$$A^{\mathrm{T}}\tilde{P} + \tilde{P}A + \tilde{C}_2^{\mathrm{T}}\tilde{C}_2 + \tilde{P}EE^{\mathrm{T}}\tilde{P}/\gamma^2 - (\tilde{P}B + \tilde{C}_2^{\mathrm{T}}\tilde{D}_2)\left(\tilde{D}_2^{\mathrm{T}}\tilde{D}_2\right)^{-1}(\tilde{D}_2^{\mathrm{T}}\tilde{C}_2 + B^{\mathrm{T}}\tilde{P}) = 0$$

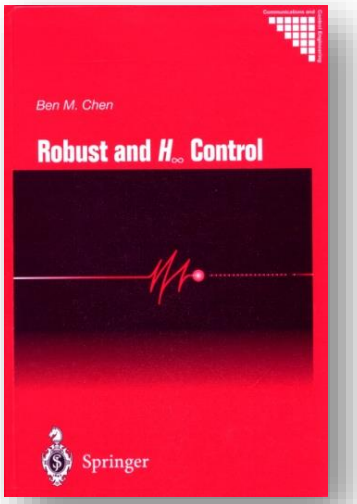
Step 4: If $\tilde{P} > 0$, go to Step 5. Otherwise, reduce the value of ε and go to Step 2.

Step 5: Compute the required state feedback control law

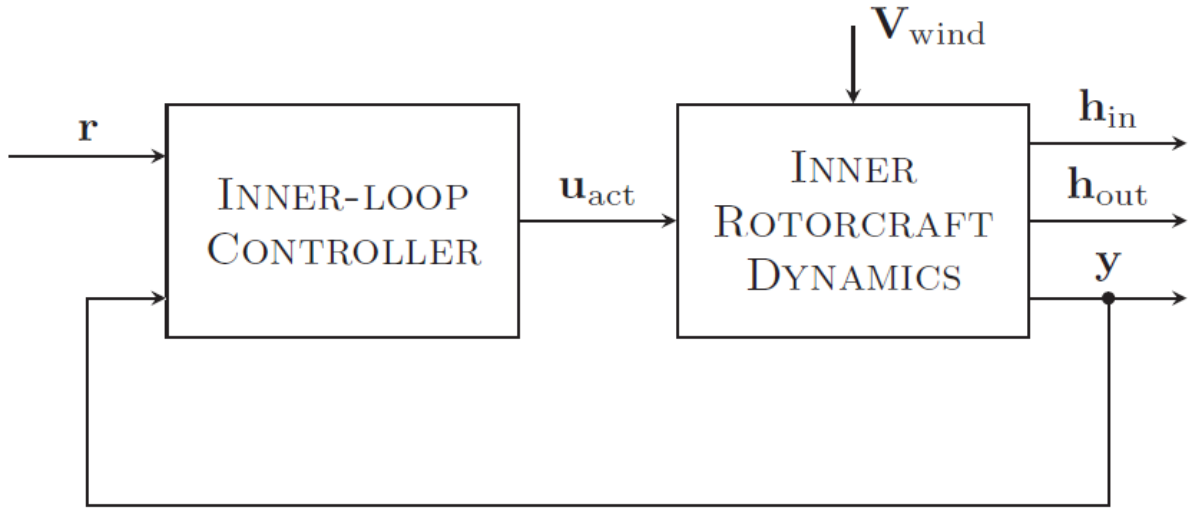
$$u = \tilde{F}\,x = -(\tilde{D}_2^{\mathrm{T}}\tilde{D}_2)^{-1}\left(\tilde{D}_2^{\mathrm{T}}\tilde{C}_2 + B^{\mathrm{T}}\tilde{P}\right)x$$

The resulting closed-loop system $T_{zw}(s)$ has: $\| T_{zw} \|_\infty < \gamma$.

More general results for the singular case can be found in Chen (2000).



Inner-loop control system design setup



$$\mathbf{x} = [\phi \quad \theta \quad p \quad q \quad a_s \quad b_s \quad r \quad \delta_{\text{ped,int}} \quad \psi]^T$$

$$\mathbf{u}_{\text{act}} = [\delta_{\text{lat}} \quad \delta_{\text{lon}} \quad \delta_{\text{ped}}]^T$$

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

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

No gain scheduling is required in our flight control system!

Inner-loop linearized model at hover

$$\begin{cases} \dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} + E\mathbf{w} \\ \mathbf{y} = C_1\mathbf{x} \\ \mathbf{h}_{\text{out}} = C_{\text{out}}\mathbf{x} \end{cases}$$

$$\mathbf{y} = \begin{pmatrix} \phi \\ \theta \\ p \\ q \\ r \\ \psi \end{pmatrix}$$

$$\mathbf{h}_{\text{out}} = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix}$$

$$A = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0.0009 & 0 & 0 \\ 0 & 0 & 0 & 0.9992 & 0 & 0 & -0.0389 & 0 & 0 \\ 0 & 0 & -0.0302 & -0.0056 & -0.0003 & 585.1165 & 11.4448 & -59.529 & 0 \\ 0 & 0 & 0 & -0.0707 & 267.7499 & -0.0003 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.0000 & -3.3607 & 2.2223 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 2.4483 & -3.3607 & 0 & 0 & 0 \\ 0 & 0 & 0.0579 & 0.0108 & 0.0049 & 0.0037 & -21.9557 & 114.2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0.0389 & 0 & 0 & 0.9992 & 0 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 43.3635 \\ 0 & 0 & 0 \\ 0.2026 & 2.5878 & 0 \\ 2.5878 & -0.0663 & 0 \\ 0 & 0 & -83.1883 \\ 0 & 0 & -3.8500 \\ 0 & 0 & 0 \end{bmatrix} \qquad E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0001 & 0.1756 & -0.0395 \\ 0.0000 & 0.0003 & 0.0338 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ -0.0002 & -0.3396 & 0.6424 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

H_∞ state feedback control law…

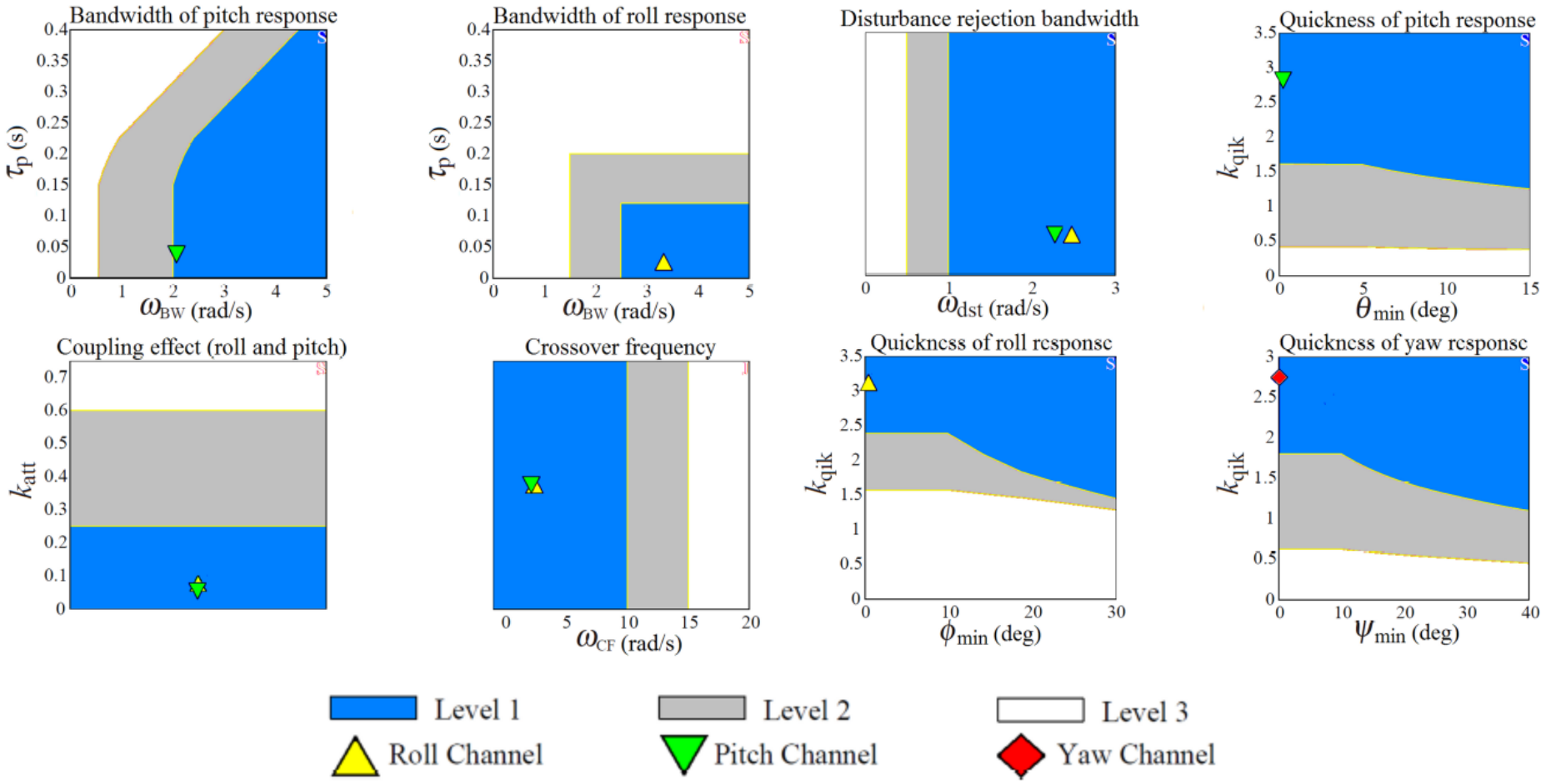
$$\mathbf{u} = F\,\mathbf{x} + G\,(\mathbf{r} - \mathbf{h}_{\text{out,trim}})$$

$$F = \begin{bmatrix} -1.0368 & -0.0604 & -0.0230 & -0.0083 & -0.2857 & -2.6165 & -0.0312 & 0.0499 & -0.0746 \\ 0.0760 & -0.9970 & 0.0174 & -0.0378 & -1.8340 & -0.1130 & 0.0026 & 0.0024 & -0.0169 \\ -0.0002 & -0.0185 & -0.0066 & 0.0004 & 0.0353 & 0.0990 & 0.0044 & 0.2295 & 0.2441 \end{bmatrix}$$

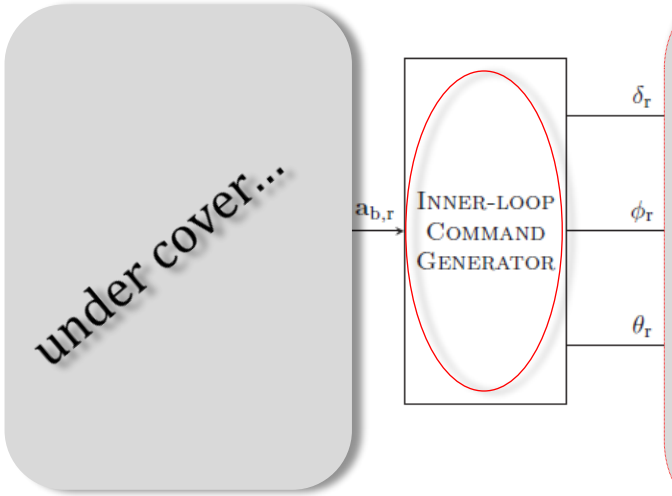
$$G = [C_{\text{out}}(A - BF)^{-1}B]^{-1} = \begin{bmatrix} 1.0368 & 0.0604 & 0.0746 \\ -0.0760 & 0.9970 & 0.0169 \\ 0.0002 & 0.0185 & -0.2441 \end{bmatrix}$$

The above state feedback control law is to be implemented together with a properly designed reduced-order observer for the state variables that cannot be measured…

Evaluation results with the standard set by U.S. Army Aviation…



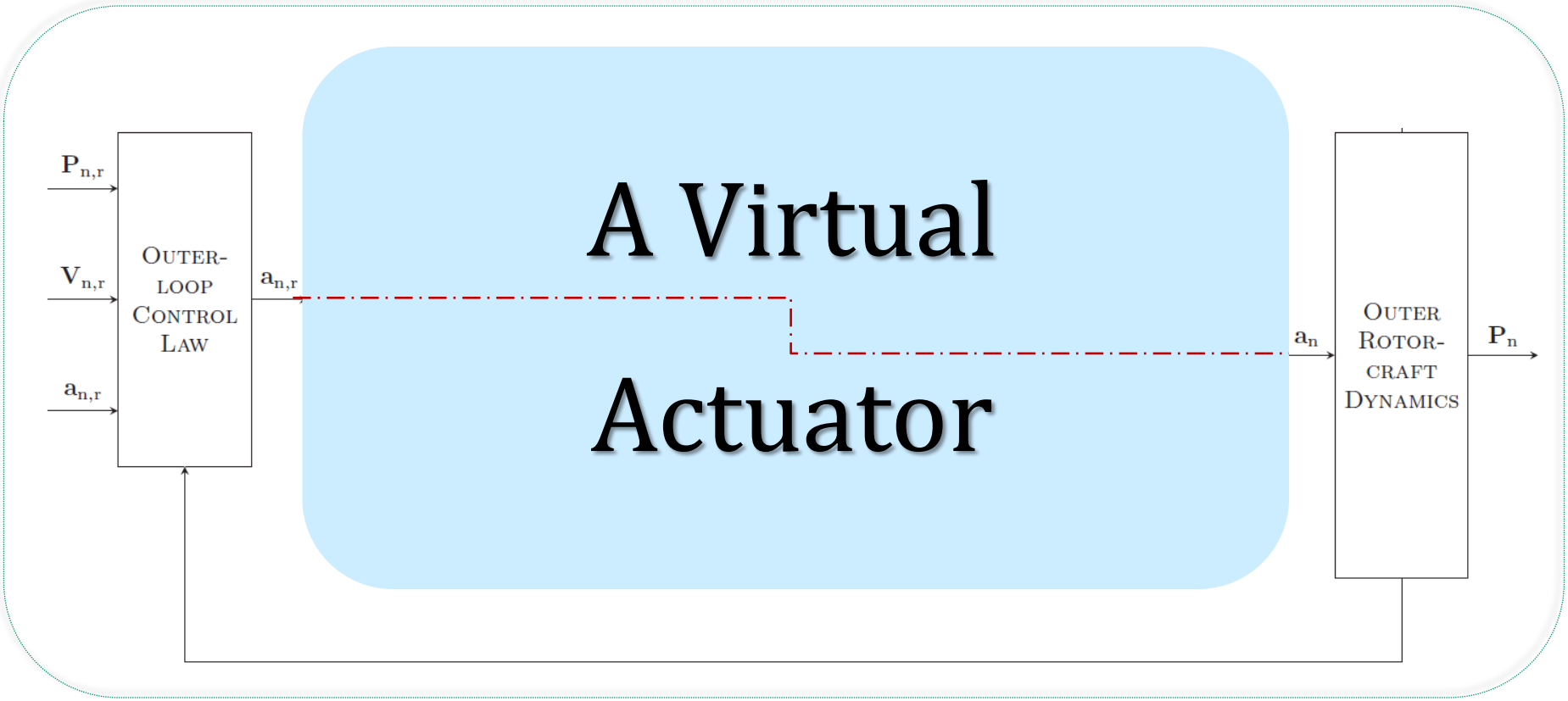
Inner-loop command generator



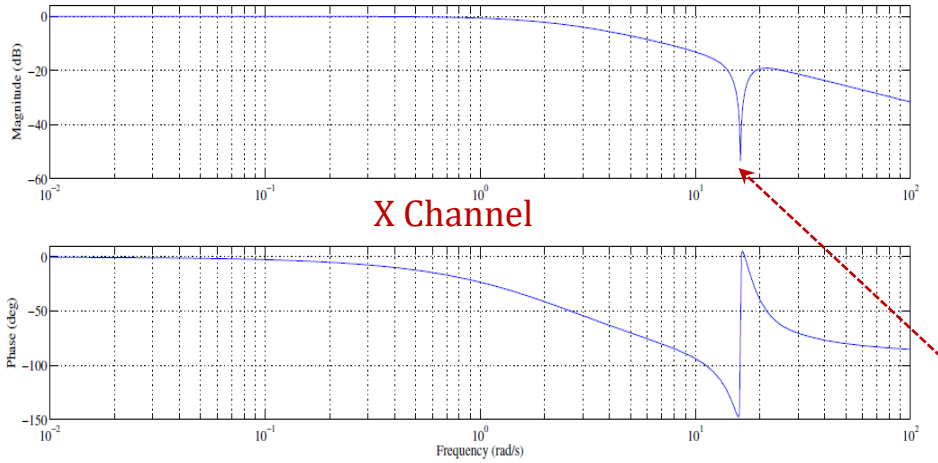
The inner-loop command generator is given as

$$\begin{pmatrix} \delta_r \\ \phi_r \\ \theta_r \end{pmatrix} = \begin{bmatrix} -0.0001 & 0.0019 & 0.0478 \\ 0.0022 & -0.1031 & -0.0048 \\ 0.1022 & 0 & 0.0002 \end{bmatrix} \mathbf{a}_{b,r}$$

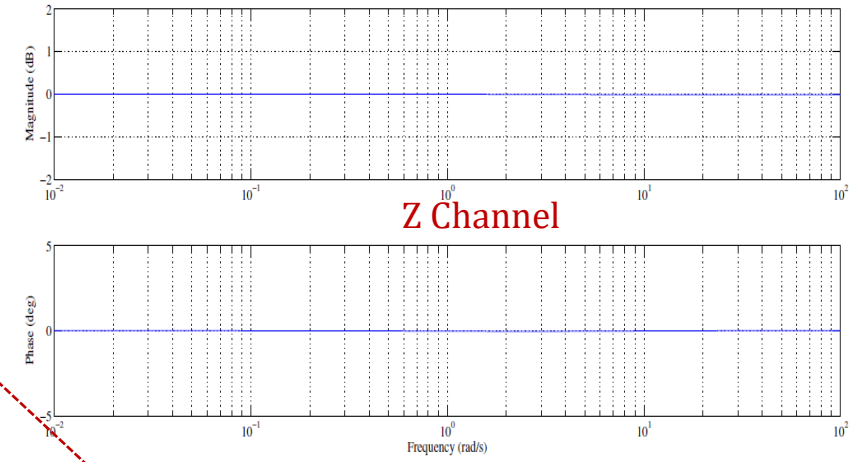
The outer-loop control is to control the position of the aircraft and at the same time to generate necessary commands for the inner-loop control system…



Properties of the virtual actuator

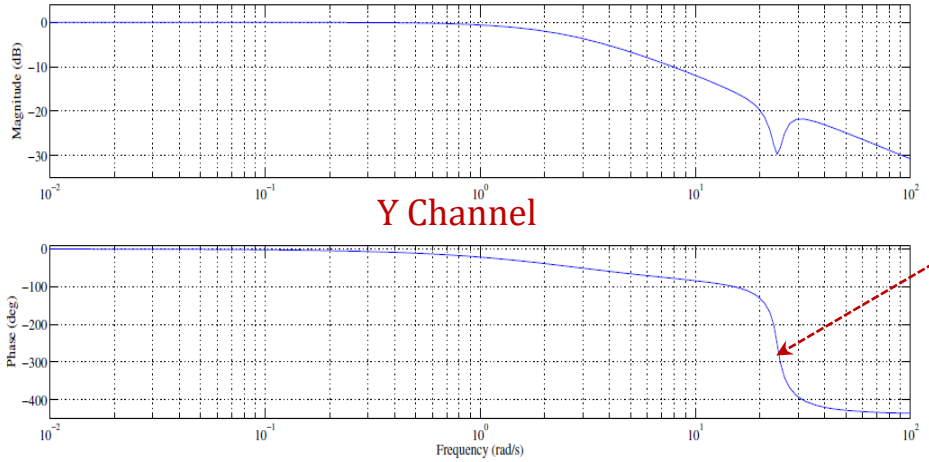


X Channel



Z Channel

Frequency response of the virtual actuator…



Y Channel

Unstable Zeros!

From practical point of view, it is safe to ignore them so long as the outer-loop bandwidth is within 1 rad/sec…

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

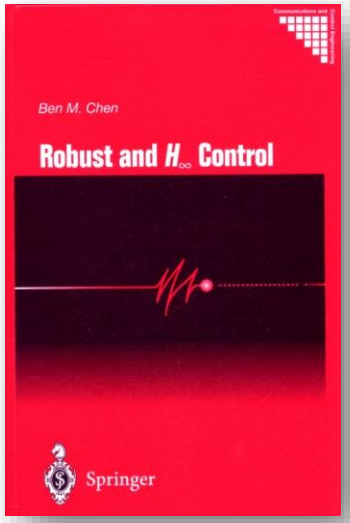
$$\begin{pmatrix} \dot{p}_* \\ \dot{v}_* \end{pmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{pmatrix} p_* \\ v_* \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} a_*$$

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

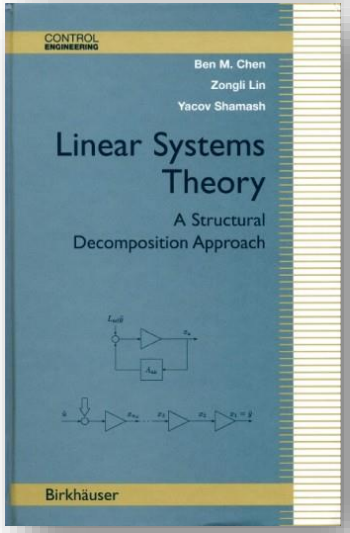
For such a simple system, it can be controlled by almost all the control techniques available in the literature, which include the most popular and the simplest one such as PID control…

Control technique adopted for the outer loop

The outer-loop control system is designed using the so-called robust and perfect tracking (RPT) control technique developed by Chen and his co-workers. It is to design a controller such that the resulting closed-loop system is stable and the controlled output almost perfectly tracks a given reference signal in the presence of any initial conditions and external disturbances.



One of the most interesting features in the RPT control method is its capability 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.



The RPT control renders flight formation of multiple UAVs a trivial task.

Consider the following continuous-time system:

$$\Sigma : \begin{cases} \dot{\boldsymbol{x}} = A\,\boldsymbol{x} + B\,\boldsymbol{u} + E\,\boldsymbol{w}, & \boldsymbol{x}(0) = \boldsymbol{x}_0, \\ \boldsymbol{y} = C_1\,\boldsymbol{x} + D_1\,\boldsymbol{w} \\ \boldsymbol{h} = C_2\,\boldsymbol{x} + D_2\,\boldsymbol{u} + D_{22}\,\boldsymbol{w} \end{cases} \tag{8.1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the state, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input, $\boldsymbol{w} \in \mathbb{R}^q$ is the external disturbance, $\boldsymbol{y} \in \mathbb{R}^p$ is the measurement output, and $\boldsymbol{h} \in \mathbb{R}^\ell$ is the output to be controlled. Given the external disturbance $\boldsymbol{w} \in L_{\boldsymbol{p}}$, $\boldsymbol{p} \in [1, \infty)$, and any reference signal vector $\boldsymbol{r} \in \mathbb{R}^\ell$ with $\boldsymbol{r}, \dot{\boldsymbol{r}}, \cdots, \boldsymbol{r}^{(\kappa-1)}$, $\kappa \geq 1$, being available, and $\boldsymbol{r}^{(\kappa)}$ being either a vector of delta functions or in $L_{\boldsymbol{p}}$, the RPT problem for the system in (8.1) is to find a parameterized dynamic measurement control law of the following form:

$$\begin{cases} \dot{\boldsymbol{v}} = A_{\rm cmp}(\varepsilon)\boldsymbol{v} + B_{\rm cmp}(\varepsilon)\boldsymbol{y} + G_0(\varepsilon)\boldsymbol{r} + \cdots + G_{\kappa-1}(\varepsilon)\boldsymbol{r}^{(\kappa-1)} \\ \boldsymbol{u} = C_{\rm cmp}(\varepsilon)\boldsymbol{v} + D_{\rm cmp}(\varepsilon)\boldsymbol{y} + H_0(\varepsilon)\boldsymbol{r} + \cdots + H_{\kappa-1}(\varepsilon)\boldsymbol{r}^{(\kappa-1)} \end{cases} \tag{8.2}$$

such that when the controller of (8.2) is applied to the system of (8.1), we have the following

1. There exists an $\varepsilon^* > 0$ such that the resulting closed-loop system with $\boldsymbol{r} = 0$ and $\boldsymbol{w} = 0$ is asymptotically stable for all $\varepsilon \in (0, \varepsilon^*]$.
2. Let $\boldsymbol{h}(t, \varepsilon)$ be the closed-loop controlled output response and let $\boldsymbol{e}(t, \varepsilon)$ be the resulting tracking error, i.e., $\boldsymbol{e}(t, \varepsilon) := \boldsymbol{h}(t, \varepsilon) - \boldsymbol{r}(t)$. Then, for any initial condition of the state, $\boldsymbol{x}_0 \in \mathbb{R}^n$,

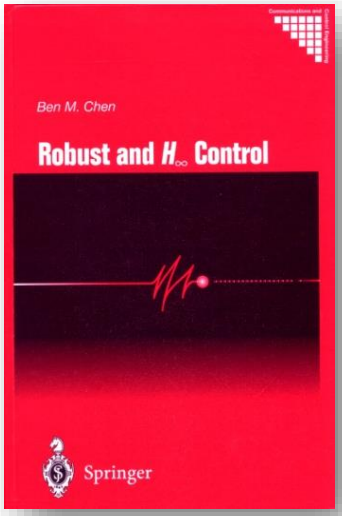
$$\|\boldsymbol{e}\|_{\boldsymbol{p}} = \left(\int_0^\infty |\boldsymbol{e}(t)|^{\boldsymbol{p}}\, dt \right)^{1/\boldsymbol{p}} \to 0 \text{ as } \varepsilon \to 0. \tag{8.3}$$

Robust to disturbance

RPT Control

Perfect in Tracking

The necessary and sufficient condition for the solvability of the general RTP control problem can be found in Chen (200). For the case when $D_1 = 0$, the conditions are relatively simple and are given as



1. (A, B) is stabilizable and (A, C_1) is detectable.
2. $D_{22} = 0$.
3. (A, B, C_2, D_2) is right invertible and of minimum phase.
4. $\mathrm{Ker}\,(C_2) \supset \mathrm{Ker}\,(C_1)$.

where Ker(X) represents the null space of a constant matrix X.

We note that the last condition is automatically satisfied if the control output $\boldsymbol{h}$ of the given system is part of its measurement output $\boldsymbol{y}$.

Step 1. For a sufficiently small scalar ε_0, we define

$$\tilde{\boldsymbol{C}}_2 = \begin{bmatrix} \boldsymbol{C}_2 \\ \varepsilon I_{\kappa\ell+n} \\ 0 \end{bmatrix}, \quad \tilde{\boldsymbol{D}}_2 = \begin{bmatrix} \boldsymbol{D}_2 \\ 0 \\ \varepsilon I_m \end{bmatrix},$$

$$\tilde{\boldsymbol{A}} = \begin{bmatrix} \tilde{A}_0 & 0 \\ 0 & A \end{bmatrix}, \quad \tilde{A}_0 = -\varepsilon_0 I_{\kappa\ell} + \begin{bmatrix} 0 & I_\ell & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & I_\ell \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

Step 2. Then, solve for positive-definite solution for the following Riccati equation

$$\boldsymbol{P}\tilde{\boldsymbol{A}} + \tilde{\boldsymbol{A}}^{\mathrm{T}}\boldsymbol{P} + \tilde{\boldsymbol{C}}_2^{\mathrm{T}}\tilde{\boldsymbol{C}}_2 - \left(\boldsymbol{P}\boldsymbol{B} + \tilde{\boldsymbol{C}}_2^{\mathrm{T}}\tilde{\boldsymbol{D}}_2\right)\left(\tilde{\boldsymbol{D}}_2^{\mathrm{T}}\tilde{\boldsymbol{D}}_2\right)^{-1}\left(\boldsymbol{P}\boldsymbol{B} + \tilde{\boldsymbol{C}}_2^{\mathrm{T}}\tilde{\boldsymbol{D}}_2\right)^{\mathrm{T}} = 0$$

Step 3. The required state feedback control law that solves the RPT problem is

$$\boldsymbol{u} = F(\varepsilon)\boldsymbol{x} + H_0(\varepsilon)\boldsymbol{r} + \cdots + H_{\kappa-1}(\varepsilon)\boldsymbol{r}^{(\kappa-1)}$$

where $\tilde{\boldsymbol{F}}(\varepsilon) = -\left(\tilde{\boldsymbol{D}}_2^{\mathrm{T}}\tilde{\boldsymbol{D}}_2\right)^{-1}\left(\boldsymbol{P}\boldsymbol{B} + \tilde{\boldsymbol{C}}_2^{\mathrm{T}}\tilde{\boldsymbol{D}}_2\right)^{\mathrm{T}} = [H_0(\varepsilon) \quad \cdots \quad H_{\kappa-1}(\varepsilon) \quad F(\varepsilon)]$.

Outer-loop control law for the UAV

$$a_{x,n} = -\begin{bmatrix} \frac{\omega_{n,x}^2}{\varepsilon_x^2} & \frac{2\zeta_x\omega_{n,x}}{\varepsilon_x} \end{bmatrix}\begin{pmatrix} x_n \\ u_n \end{pmatrix} + \left(\frac{\omega_{n,x}^2}{\varepsilon_x^2}\right)x_{n,r} + \left(\frac{2\zeta_x\omega_{n,x}}{\varepsilon_x}\right)u_{n,r} + a_{x,n,r}$$

$$a_{y,n} = -\begin{bmatrix} \frac{\omega_{n,y}^2}{\varepsilon^2} & \frac{2\zeta_y\omega_{n,y}}{\varepsilon} \end{bmatrix}\begin{pmatrix} y_n \\ v_n \end{pmatrix} + \left(\frac{\omega_{n,y}^2}{\varepsilon^2}\right)y_{n,r} + \left(\frac{2\zeta_y\omega_{n,y}}{\varepsilon}\right)v_{n,r} + a_{y,n,r}$$

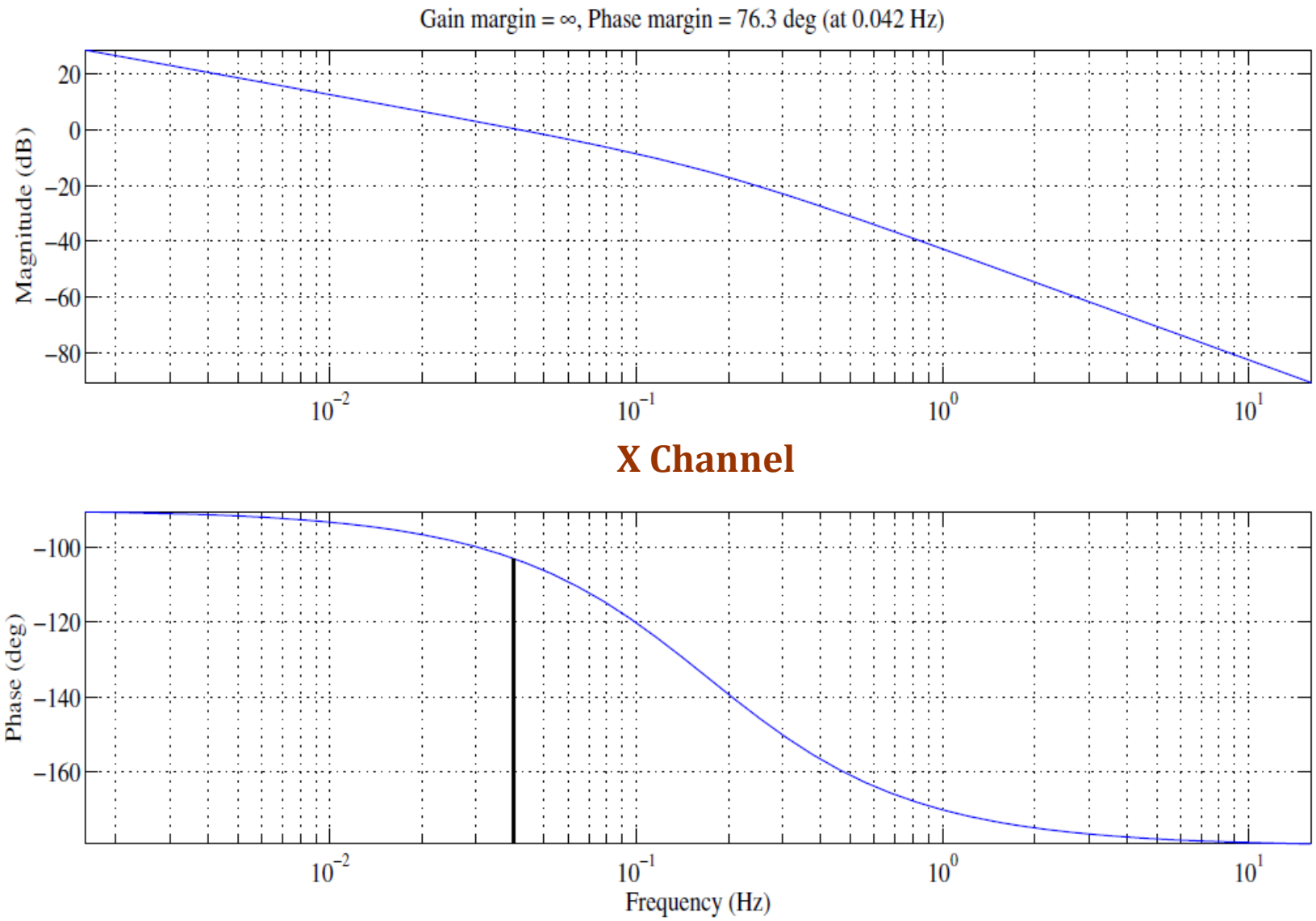
$$a_{z,n} = -\begin{bmatrix} \frac{\omega_{n,z}^2}{\varepsilon^2} & \frac{2\zeta_z\omega_{n,z}}{\varepsilon} \end{bmatrix}\begin{pmatrix} z_n \\ w_n \end{pmatrix} + \left(\frac{\omega_{n,z}^2}{\varepsilon^2}\right)z_{n,r} + \left(\frac{2\zeta_z\omega_{n,z}}{\varepsilon}\right)w_{n,r} + a_{z,n,r}$$

$$\varepsilon_x = \varepsilon_y = \varepsilon_z = 1$$

$$\zeta_x = 1, \quad \zeta_y = 1, \quad \zeta_z = 1.1$$

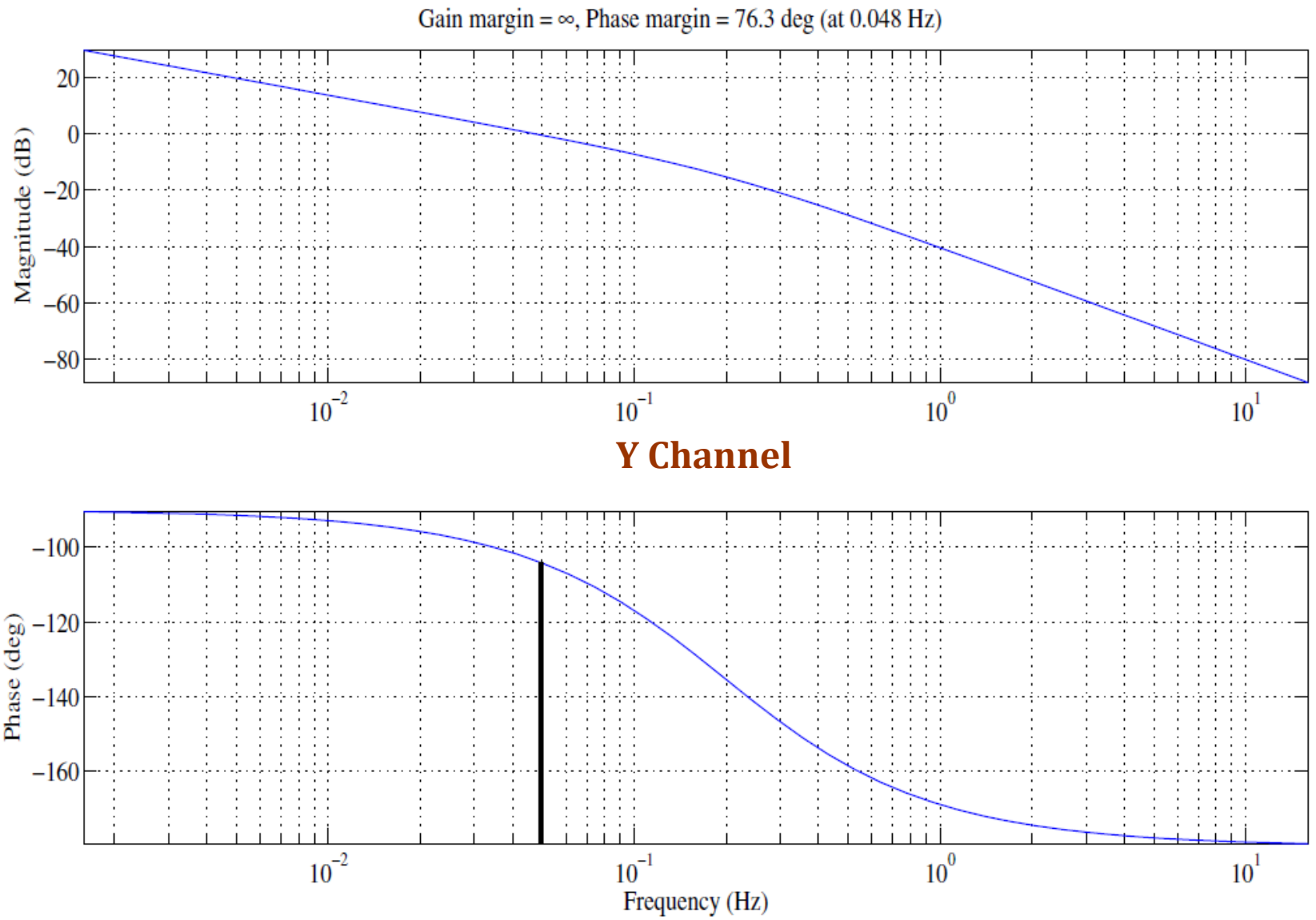
$$\omega_{n,x} = 0.54, \quad \omega_{n,y} = 0.62, \quad \omega_{n,z} = 0.78$$

Outer-loop control system gain and phase margins

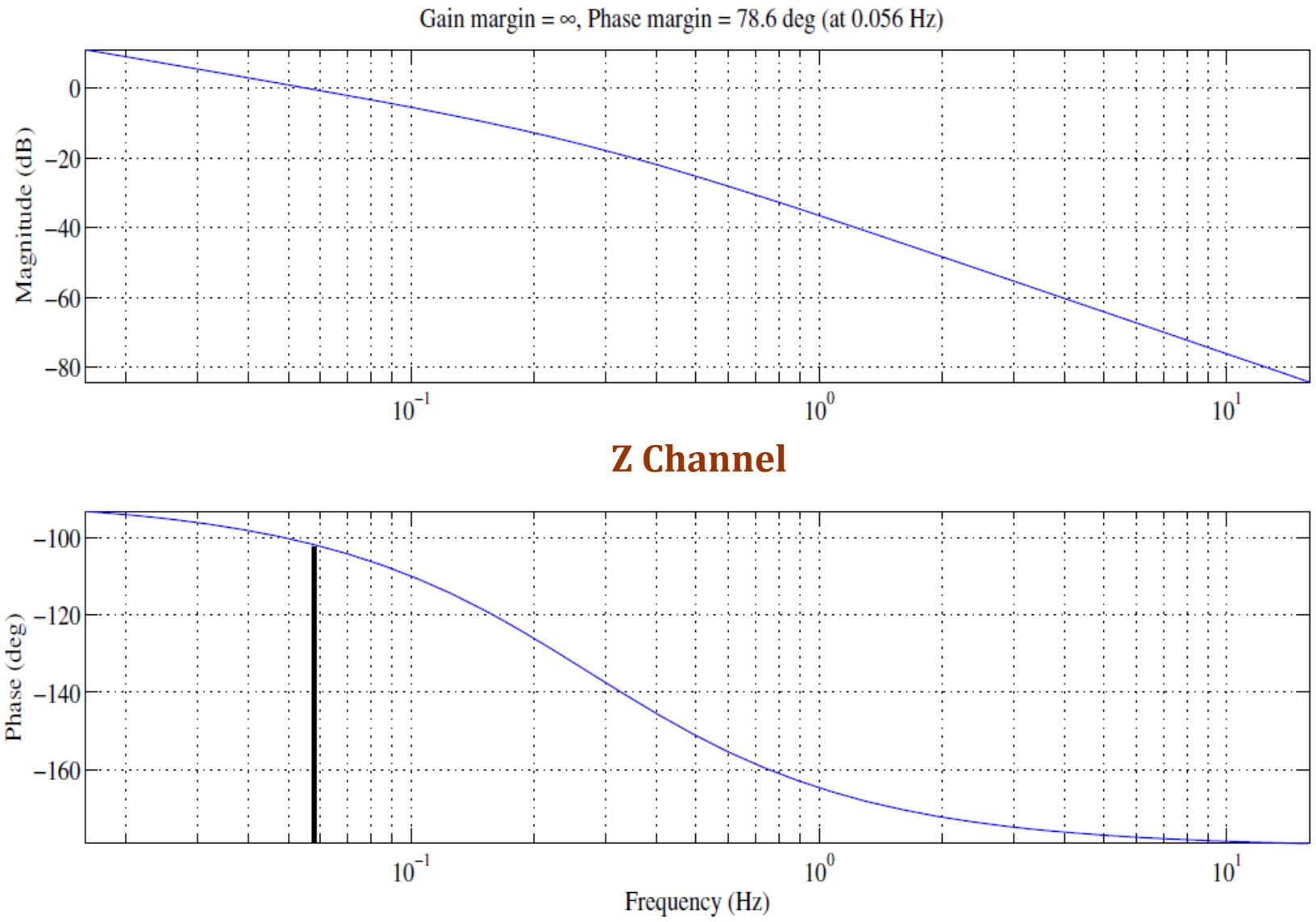


X Channel

Outer-loop control system gain and phase margins



Outer-loop control system gain and phase margins



Z Channel

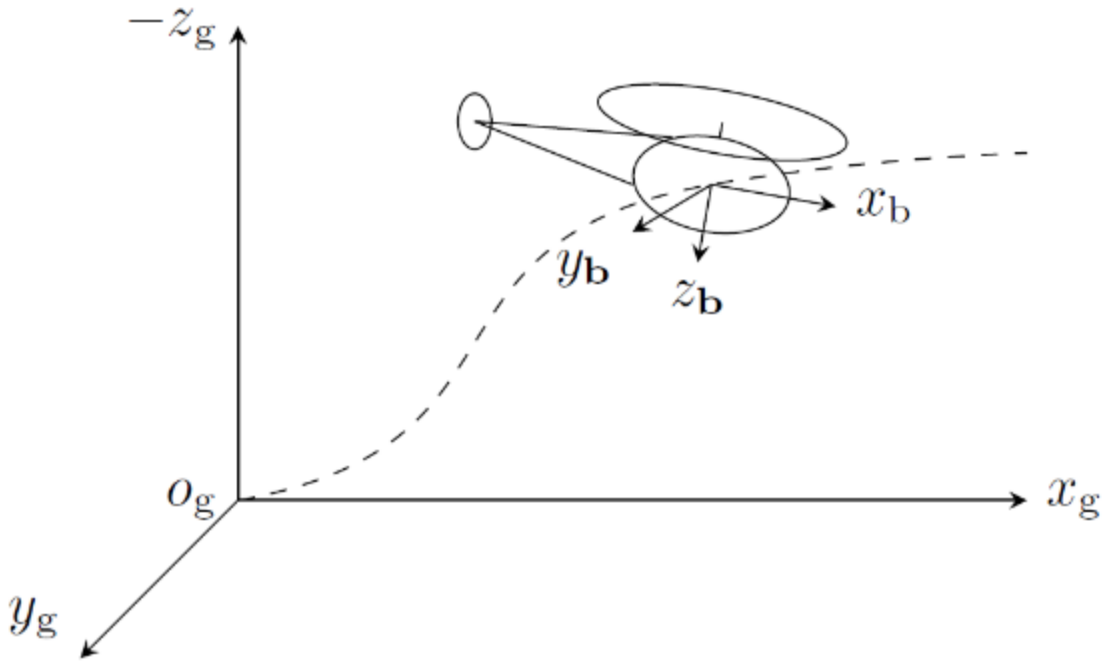
Single-rotor Helicopter

Quadrotor

What is Navigation?

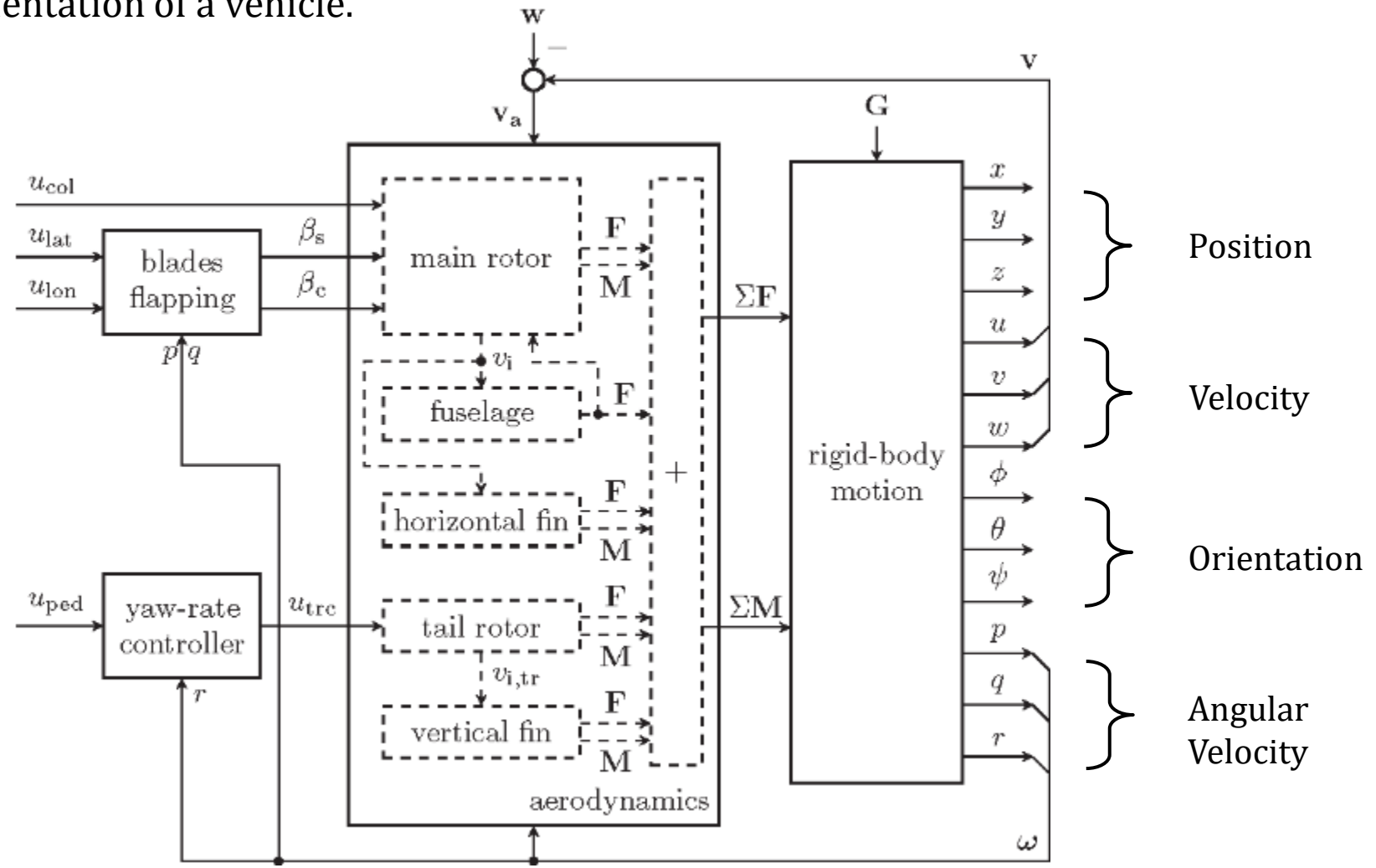
In a broader sense, navigation is the study of monitoring and controlling the movement of an object from one place to another. It is to answer the following three questions:

- **Where is the object?**
- **Where should the object be?**
- **How to bring the object back to the correct path?**



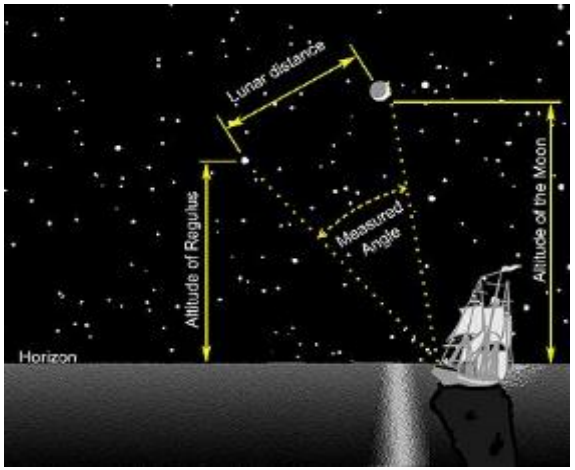
What is Navigation?

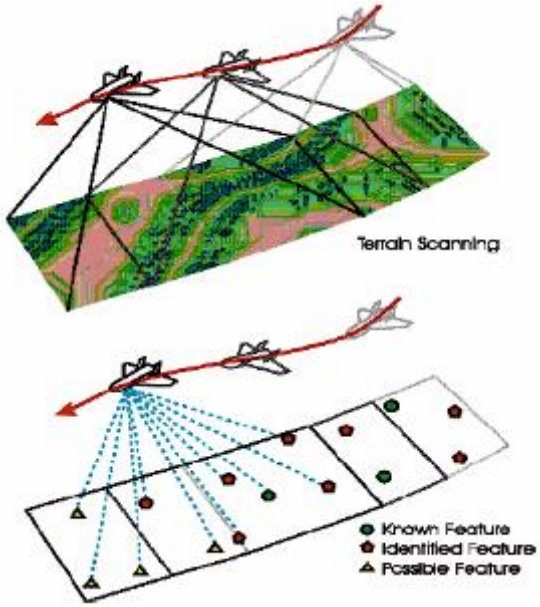
In a narrower sense, navigation refers to the study of determining the position and orientation of a vehicle.

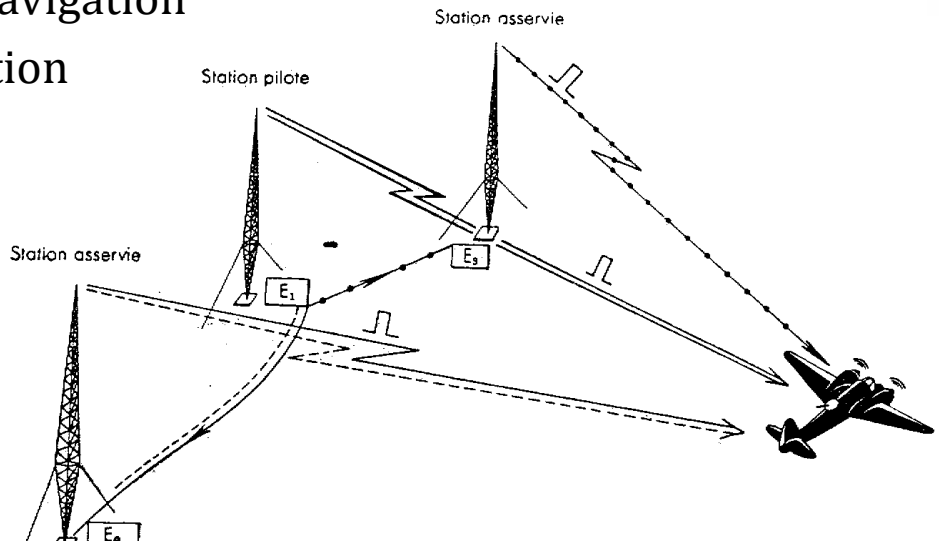


Different Methods of Navigation

- Landmarks recognition
- Celestial Navigation
- Radio Navigation
  - Signal-based navigation
  - Satellite navigation
- Inertial Navigation
  - Gimbaled INS
  - Strapdown INS
- Laser Aided Navigation
- Visual Navigation
- …

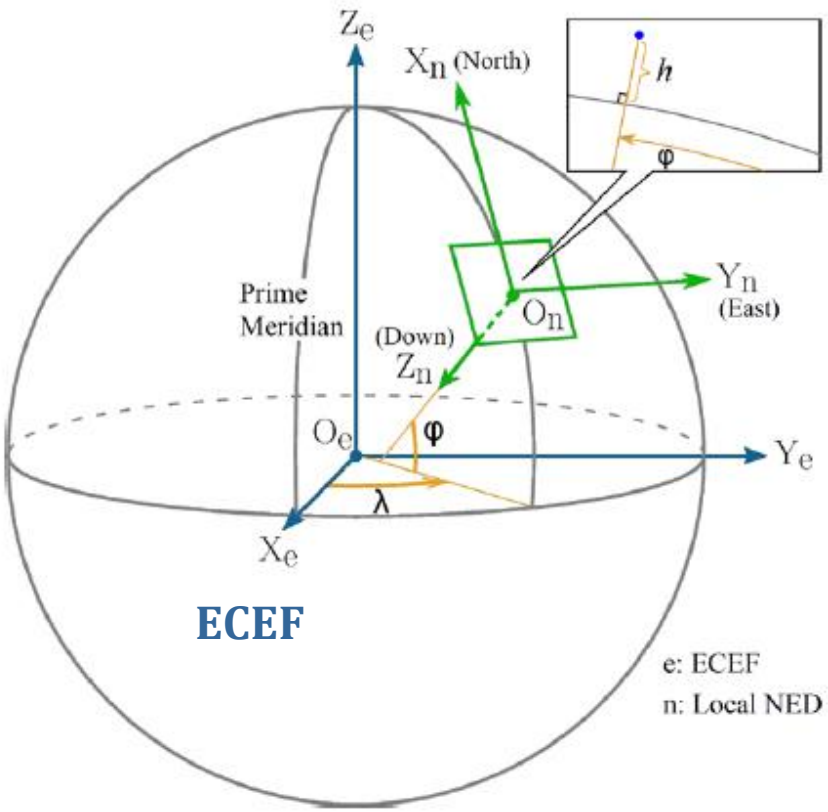






Coordinate Systems

Geodetic, earth-centered earth-fixed (ECEF) & local north-east-down (NED)

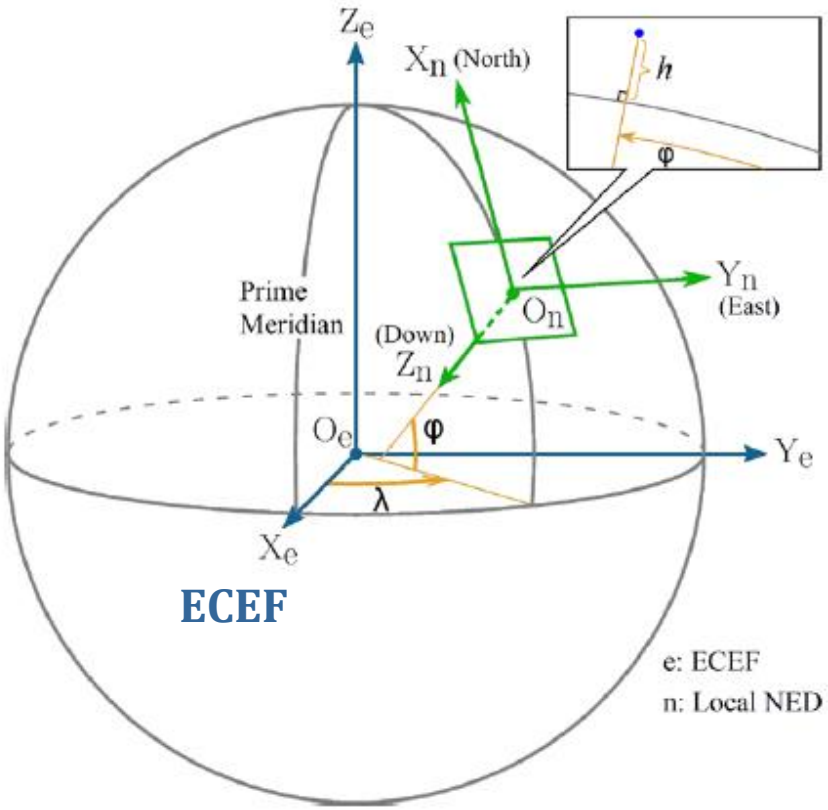


Geodetic: $\mathbf{P}_g = \begin{bmatrix} \lambda \\ \varphi \\ h \end{bmatrix}$

- Origin at centre of mass of Earth
- λ: The **longitude** measures the rotational angle between the Prime Meridian and the measured point.
- φ:The **latitude** measures the angle between the equatorial plane and the normal of the reference ellipsoid that passes through the measured point.
- h: The **altitude** is the local vertical distance between the measured point and the reference ellipsoid.

Coordinate Systems

Geodetic, earth-centered earth-fixed (ECEF) & local north-east-down (NED)

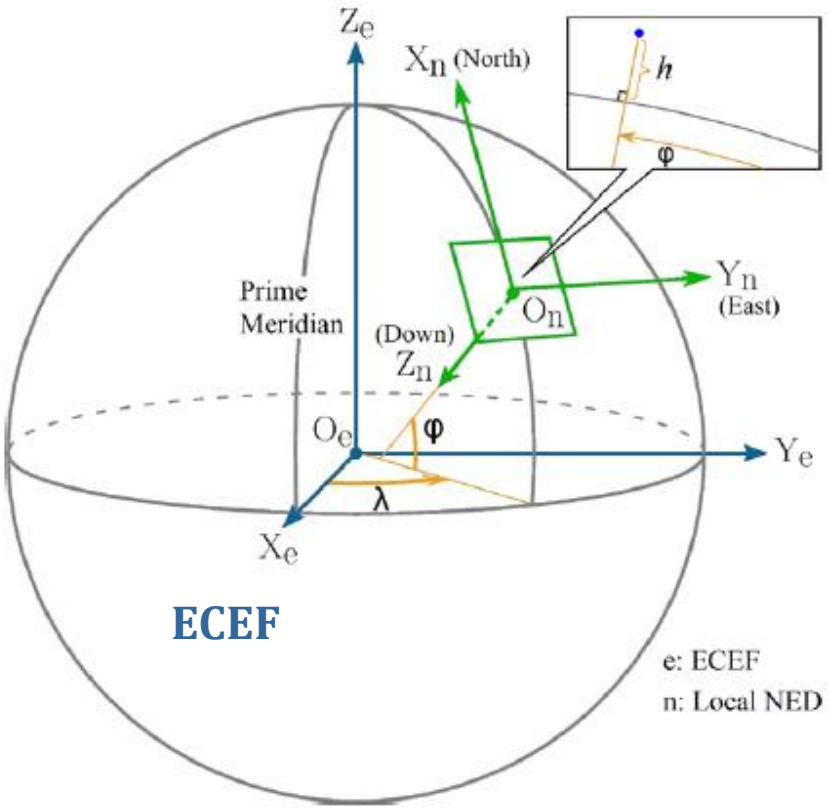


$$\textbf{ECEF: } \mathbf{P}_e = \begin{bmatrix} X_e \\ Y_e \\ Z_e \end{bmatrix}$$

- Origin at centre of mass of Earth
- The Z-axis extends through true north
- The X-axis intersects the sphere of the earth at 0° latitude (the equator) and 0° longitude (prime meridian in Greenwich)
- The Y-axis intersects the sphere of the earth at 0° latitude (the equator) and 90° longitude

Coordinate Systems

Geodetic, earth-centered earth-fixed (ECEF) & local north-east-down (NED)



Local NED: $\mathbf{P}_n = \begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix}$

- The origin is arbitrarily fixed to a point on the earth's surface.
- The X-axis points toward the ellipsoid north (geodetic north).
- The Y-axis points toward the ellipsoid east (geodetic east).
- The Z-axis points downward along the ellipsoid normal.

Conversion from Geodetic to ECEF Coordinates

$$\begin{cases} X_e = (N(\varphi) + h)\cos\varphi\cos\lambda \\ Y_e = (N(\varphi) + h)\cos\varphi\sin\lambda \\ Z_e = \left(\frac{b^2}{a^2}\mathrm{N}(\varphi) + \mathrm{h}\right)\sin\varphi \end{cases}$$

where $N(\varphi) = \dfrac{a^2}{\sqrt{a^2\cos^2\varphi + b^2\sin^2\varphi}} = \dfrac{a^2}{\sqrt{1 - e^2\sin^2\varphi}}$

a: Equatorial radius (semi-major axis)

b: Polar radius (semi-minor axis)

$e^2 = 1 - \frac{b^2}{a^2}$: Square of the first numerical eccentricity of ellipsoid

$N(\varphi)$: Prime vertical radius of curvature

Conversion from ECEF to Local NED Coordinates

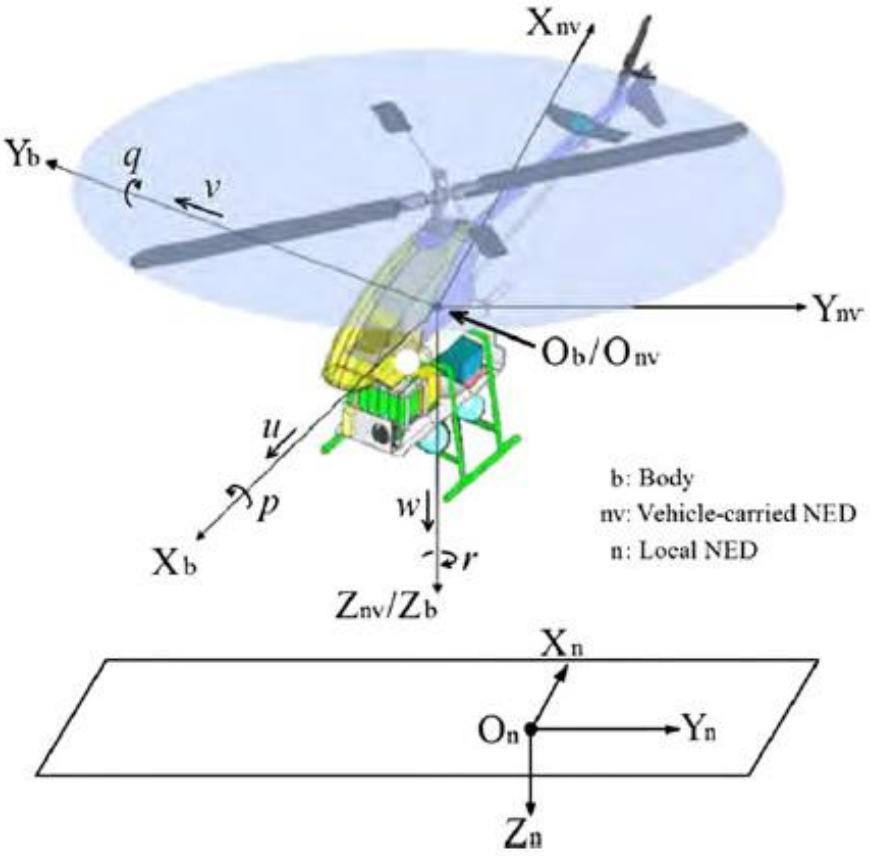
(Zero) reference point in ECEF coordinates

$$\begin{bmatrix} X_n \\ Y_n \\ Z_n \end{bmatrix} = R^T \left(\begin{bmatrix} X_e \\ Y_e \\ Z_e \end{bmatrix} - \begin{bmatrix} X_{e0} \\ Y_{e0} \\ Z_{e0} \end{bmatrix} \right)$$

where $$R = \begin{bmatrix} -\sin\varphi\cos\lambda & -\sin\lambda & -\cos\varphi\cos\lambda \\ -\sin\varphi\sin\lambda & \cos\lambda & -\cos\varphi\sin\lambda \\ \cos\varphi & 0 & -\sin\varphi \end{bmatrix}$$

Coordinate Systems

Body coordinate systems

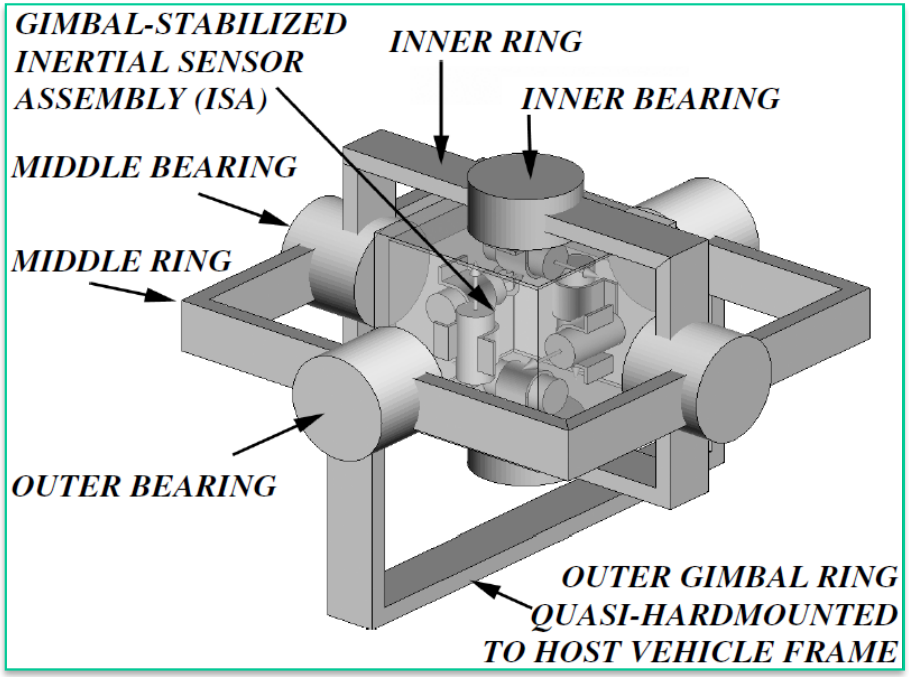


Body: $\mathbf{V}_b = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$

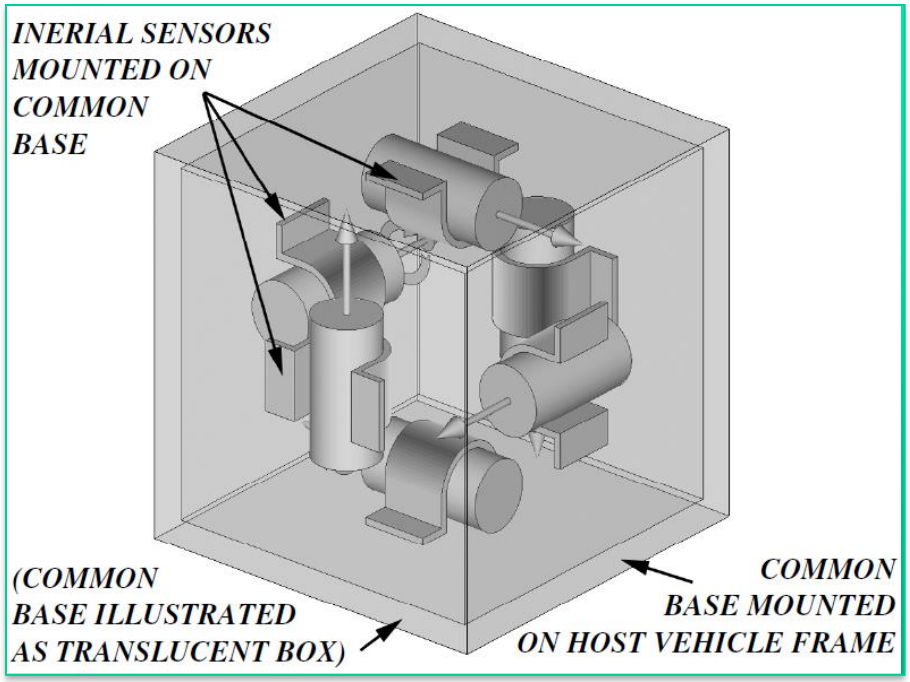
- The origin (denoted by O_b) is located at the centre of gravity (CG) of the vehicle.
- The X-axis (denoted by X_b) points forward.
- The Y-axis (denoted by Y_b) points to the right side of the vehicle.
- The Z-axis (denoted by Z_b) points downward to comply with the right-hand rule.

Inertial Measurement Unit

Inertial Measurement Units (IMUs) is a self-contained system that measures linear and angular motion usually with **a triad of gyroscopes** and **triad of accelerometers**. An IMU can either be gimbaled or strapdown, outputting the integrating quantities of angular velocity and acceleration in the sensor/body frame.



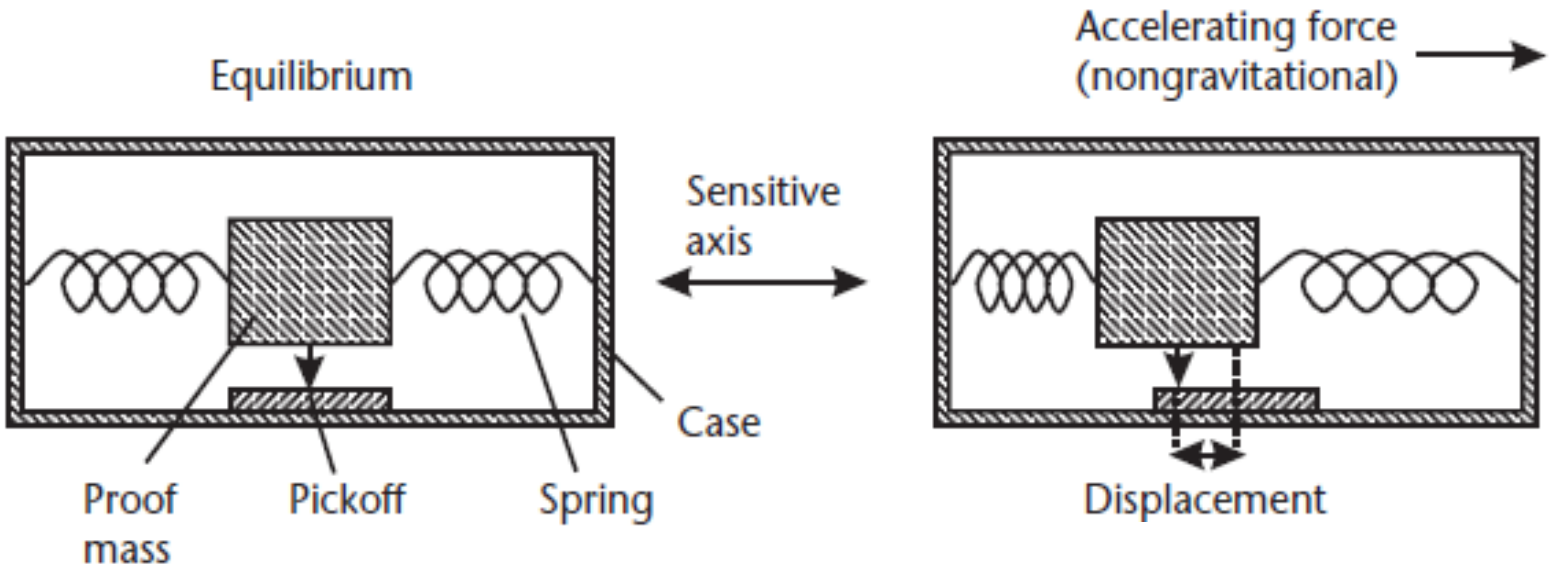
Gimballed IMU



Strapdown IMU

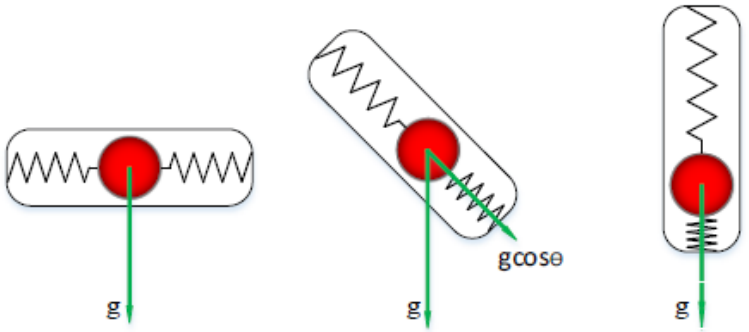
Accelerometer

By attaching a mass to a spring, measuring its deflection, we get a simple accelerometer.



Basic principle: mass inertia effect

- Measure voltage of piezo cristals
- Changes in capacity
- Laser deflection
- Pendulous Integrating (PIGA)



Accelerometer

Define noise model of measurement vector $a^* = (a^*_x, a^*_y, a^*_z)$

$$\underbrace{\vec{a}^*}_{\text{measurement}} = \underbrace{\vec{b}_C + s \cdot M}_{\text{calibration}} \cdot \underbrace{\vec{a} + \vec{b}}_{\text{estimation}} + \underbrace{W\vec{n} + \vec{o}}_{\text{modelling}}$$

Where: a^* = measured acceleration vector

b_c = long term constant bias

s = scale factor

M = misalignement correction matrix

b = short term constant bias

W=crosscorrelation matrix for white noise vector n

o = other (environmental) influences

Simplified model after calibration

$$\mathbf{a}_{\mathrm{nb,IMU}} = \mathbf{a}_{\mathrm{nb}} - \mathbf{b}_a - \mathbf{w}_a,$$

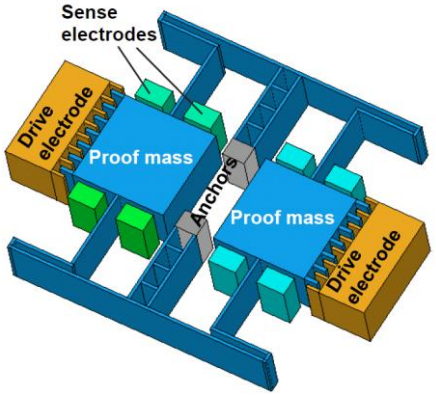
where $\mathbf{b}_a \in R^3$ is unknown but constant measurement biases; and $\mathbf{w}_a \in R^3$ is zero-mean Gaussian white noises.

Gyroscope

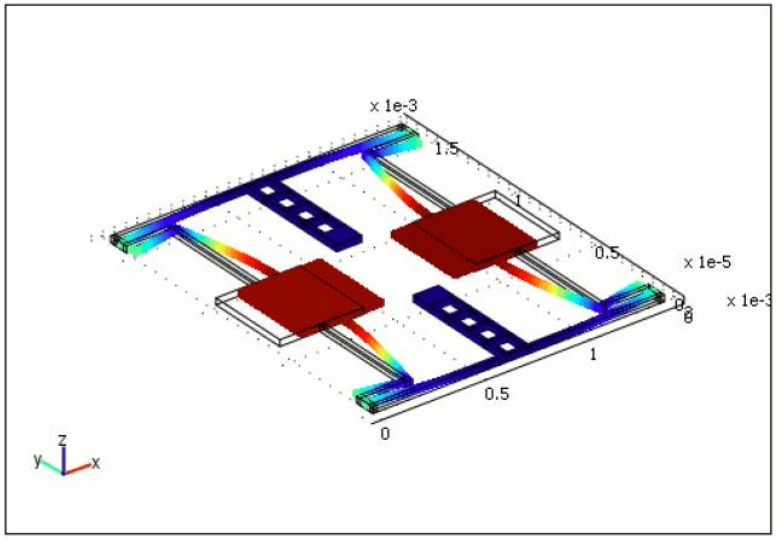
Micro Electro-Mechanical System

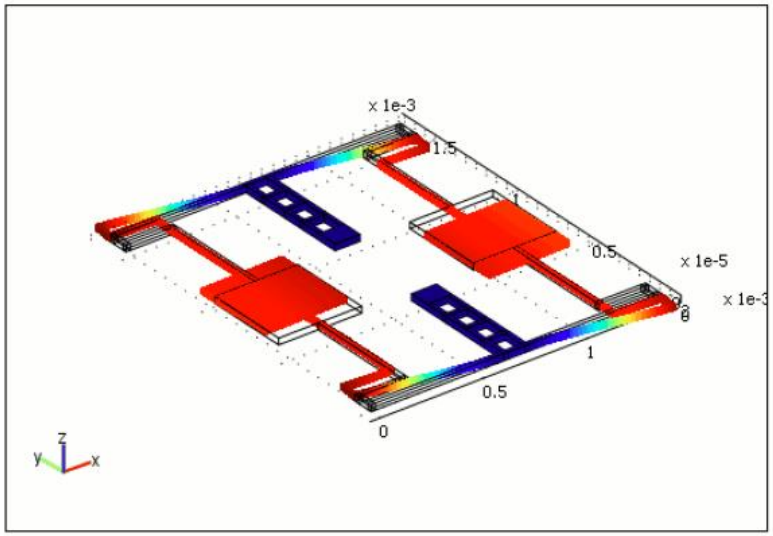
MEMS gyroscopes use the Coriolis effect to directly measure angular velocity:

- Proof masses vibrate at frequency ω_r giving the position $p = A\sin(\omega_r t)$
- The Coriolis effect introduces the acceleration $a = -2(v \times \omega)$ with the velocity $v = \dot{p} = A\omega_r \cos(\omega_r t)$



- We can measure the introduced acceleration with an accelerometer
- Error rates of ~1°/h when EKF fused (0.01°/h sensors exist)





Rate Gyro

Define noise model of measurement vector $\omega^* = (\omega^*_x, \omega^*_y, \omega^*_z)$

$$\underbrace{\vec{\omega}^*}_{\text{measurement}} = \underbrace{\vec{b}_C + s \cdot M}_{\text{calibration}} \cdot \underbrace{\vec{\omega} + \vec{b}}_{\text{estimation}} + \underbrace{W\vec{n}}_{\text{modelling}} + \vec{o}$$

Where: ω^* = measured angular velocities

b_c = long term constant bias

s = scale factor

M = misalignment correction matrix

b = short term constant bias

W= crosscorrelation matrix for white noise vector n

o = other (environmental) influences

Simplified model after the calibration

$$\boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/n,IMU}} = \boldsymbol{\omega}^{\mathrm{b}}_{\mathrm{b/n}} - \mathbf{b}_{\omega} - \mathbf{w}_{\omega},$$

where $\mathbf{b}_\omega \in R^3$ is unknown but constant measurement biases; and $\mathbf{w}_\omega \in R^3$ is zero-mean Gaussian white noises.

Global Positioning System

The Global Positioning System (GPS) is a space-based navigation system that provides **location** and **time** information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.

- Based on time and the known position of specialized satellites;
- Independent of time of day or weather;
- Begin in 1973 and fully operational since April 1995;
- Currently 30 active satellites (6 paths, 55 deg. inclination);
- Carrier frequency is 1575.42 MHz (1227.6 MHz).

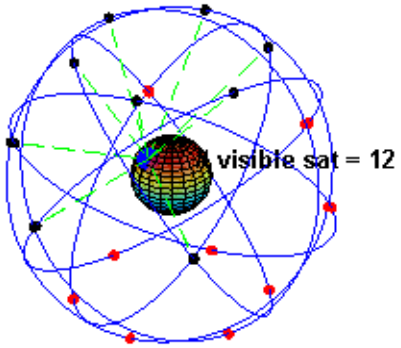
Other Similar Systems

- GLONASS – Russian
- BEIDO – China
- GAILILEO – Europe

Global Positioning System

Space Satellites:

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



Control Station:

- Master control for satellites
- Update satellite position and atomic clocks

User Reception:

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

Global Positioning System

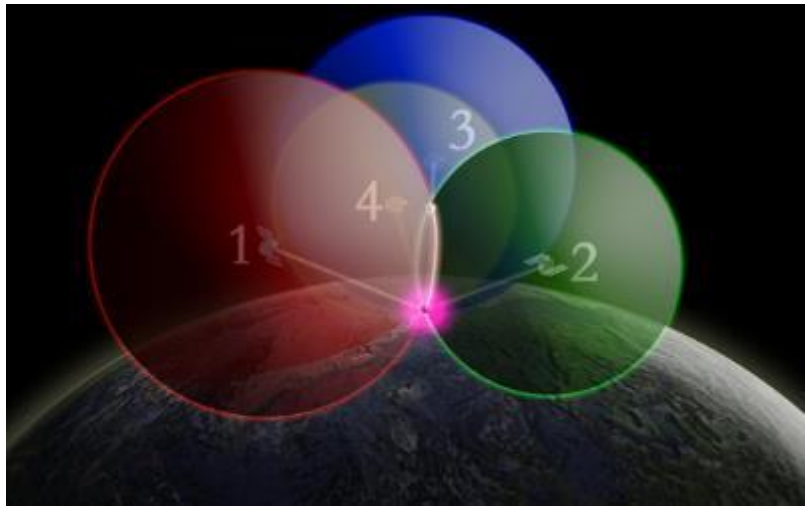
Navigation Equation

The receiver uses messages received from satellites to determine the satellite positions and time sent. For *n* satellites, the equations to satisfy are:

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

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

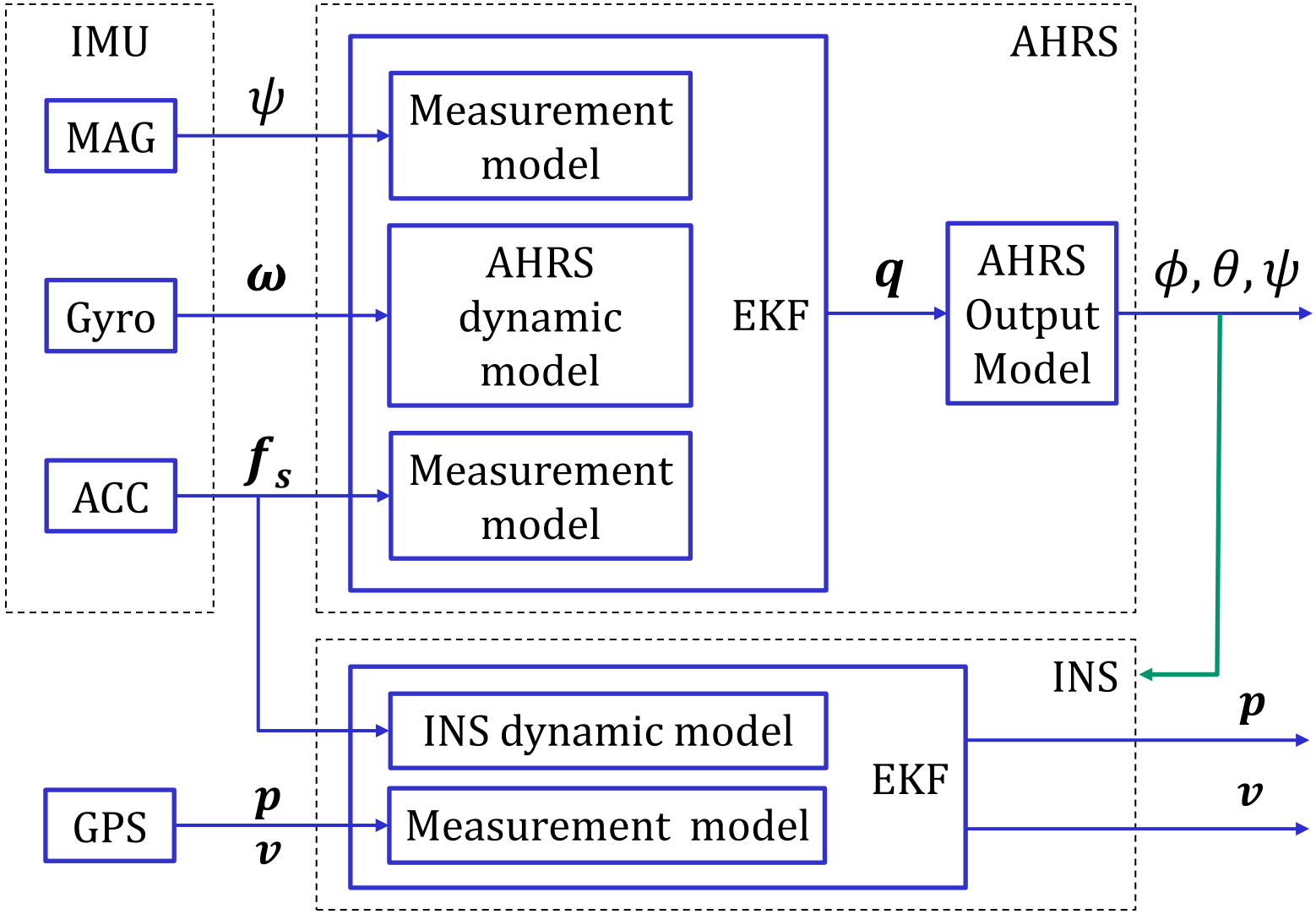
$$\left(\hat{x},\hat{y},\hat{z},\hat{b}\right)=\underset{(x,y,z,b)}{\arg\min}\sum_i\left(\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2}+bc-p_i\right)^2$$

where

$p_i=\left(\tilde{t}_i-s_i\right)c$ is called the pseudo-range

For 3D localization, we need at least 4 satellites!

GPS-Aided Inertial Navigation System



What is Kalman Filter?

Given the linear dynamical system:

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k)$$
$$y(k) = H(k)x(k) + w(k)$$

$x(k)$ is the n-dimensional state vector (unknown)

$u(k)$ is the m-dimensional input vector (known)

$y(k)$ is the p-dimensional output vector (known, measured)

$F(k), G(k), H(k)$ are appropriately dimensioned system matrices (known)

$v(k), w(k)$ are zero-mean, white Gaussian noise with (known) covariance matrices $Q(k), R(k)$

the Kalman Filter is a recursion that provides the 'best' estimate of the state vector *x*.

Kalman Filter

Why use Kalman Filter?

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k)$$
$$y(k) = H(k)x(k) + w(k)$$

- State estimation via probability optimization from both inputs and measurements
- Noise smoothing
- Realtime scalable (computes next estimate using only most recent measurement)

How does Kalman Filter work?

$$x(k+1) = F(k)x(k) + G(k)u(k) + v(k)$$
$$y(k) = H(k)x(k) + w(k)$$

1. Prediction based on last estimate:

$$\hat{x}(k+1|k) = F(k)\hat{x}(k|k) + G(k)u(k)$$
$$\hat{y}(k) = H(k)\hat{x}(k+1|k)$$

2. Calculate correction based on prediction and current measurement:

$$\Delta x = f\left(y(k+1), \hat{x}(k+1|k)\right)$$

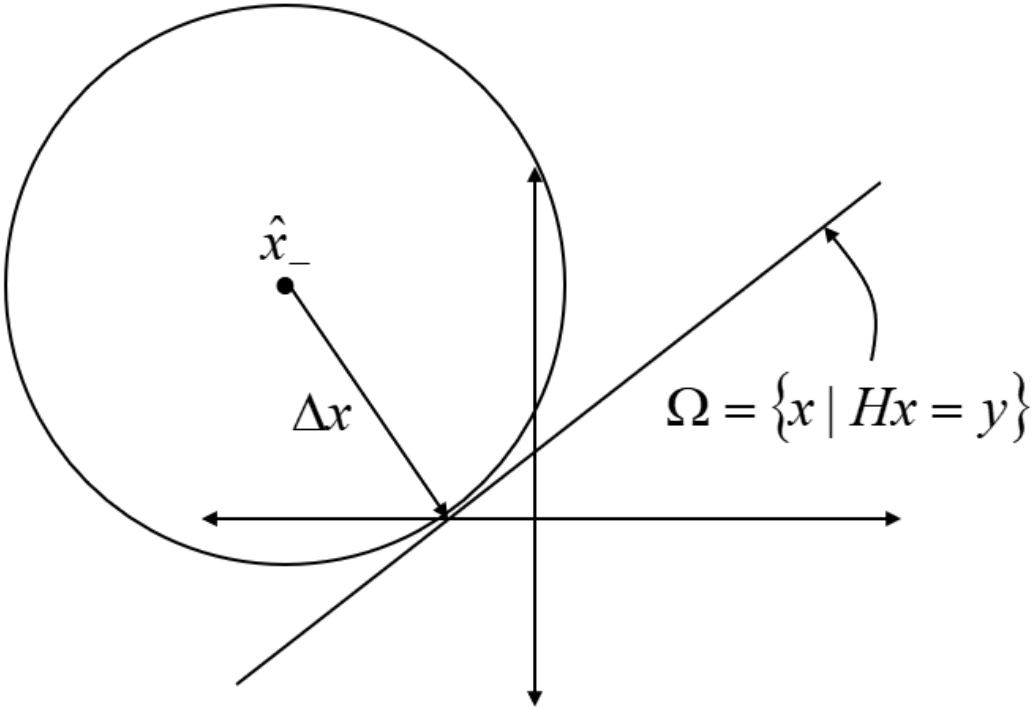
3. Update prediction: $\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + \Delta x$

Kalman Filter (A Simple Special Case)

If measurement has zero noise and variance of *x* is the same in all axis ...

$$y = Hx$$

Given prediction $\hat{x}_-$ and output y, find Δx so that $\hat{x} = \hat{x}_- + \Delta x$ is the "best" estimate of x.



$$\Delta x = H^T \left(HH^T \right)^{-1} \nu$$

Kalman Filter (A Simple Special Case)

System:

$$x(k+1) = Fx(k) + Gu(k) + v(k)$$
$$y(k) = Hx(k)$$

1. Prediction:

$$\hat{x}(k+1 \mid k) = F\hat{x}(k \mid k) + Gu(k)$$

2. Compute correction:

$$\Delta x = H^T \left(HH^T\right)^{-1} \left(y(k+1) - H\hat{x}(k+1 \mid k)\right)$$

3. Update:

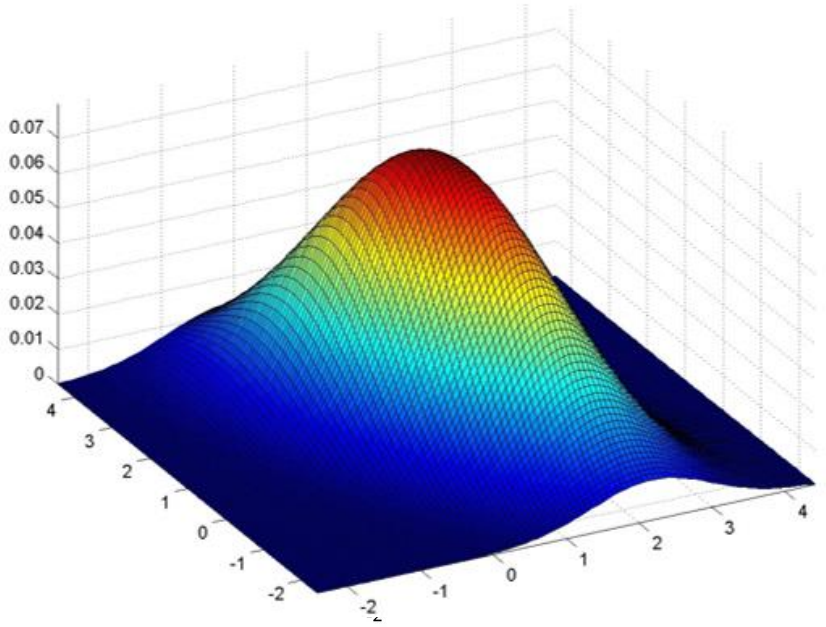
$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + \Delta x$$

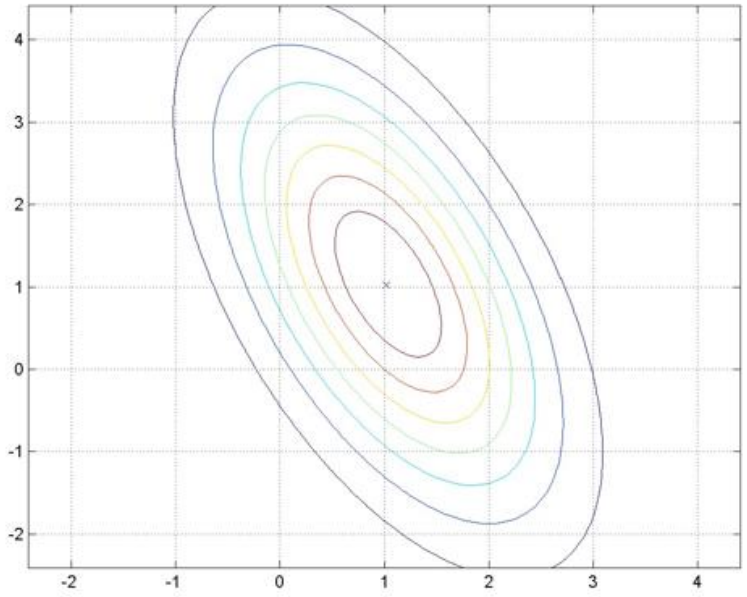
Kalman Filter (A More Complicated Case)

If we consider variance of *x* follows a joint Gaussian distribution p(x).

$$p(x) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} e^{\frac{-1}{2}\left((x-\hat{x})^T P^{-1}(x-\hat{x})\right)}$$

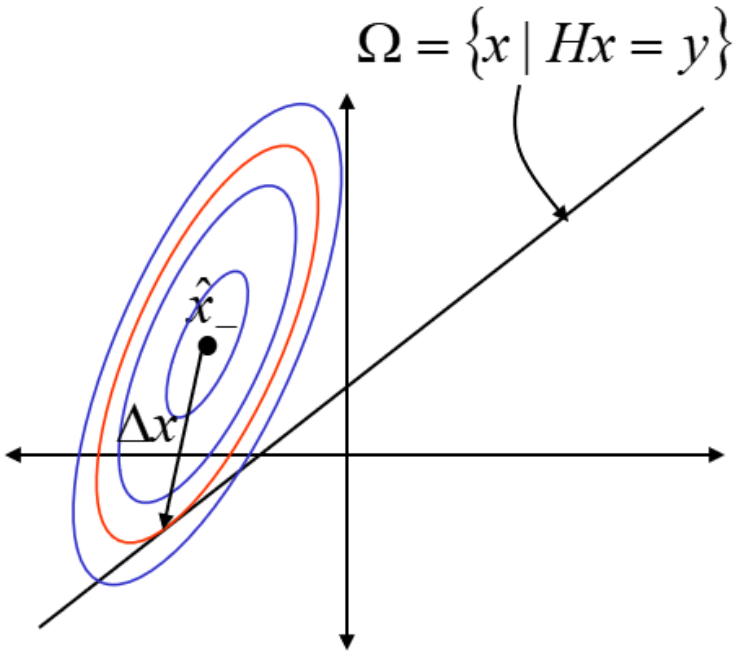
where $P = E\left((x-\hat{x})(x-\hat{x})^T\right)$ is the *covariance matrix*





Kalman Filter (A More Complicated Case)

Given prediction $\hat{x}_-$, covariance P, and output y, find Δx so that $\hat{x} = \hat{x}_- + \Delta x$ is the "best" (i.e. most probable) estimate of x.



$$p(x) = \frac{1}{(2\pi)^{n/2}|P|^{1/2}} e^{\frac{-1}{2}\left((x-\hat{x})^T P^{-1}(x-\hat{x})\right)}$$

The most probable Δx is the one that :

1. satisfies $\hat{x} = \hat{x}_- + \Delta x$
2. minimizes $\Delta x^T P^{-1} \Delta x$

Kalman Filter (A More Complicated Case)

Suppose we define a new inner product on R^n to be :

$$\langle x_1, x_2 \rangle = x_1^T P^{-1} x_2 \qquad \text{(this replaces the old inner product } x_1^T x_2\text{)}$$

Then we can define a new norm $\|x\|^2 = \langle x, x \rangle = x^T P^{-1} x$

The $\hat{x}$ in Ω that minimizes $\|\Delta x\|$ is the orthogonal projection of $\hat{x}$_ onto Ω, so Δx is orthogonal to Ω.

$$\Rightarrow \langle \omega, \Delta x \rangle = 0 \text{ for } \omega \text{ in } T\Omega = null(H)$$

$$\langle \omega, \Delta x \rangle = \omega^T P^{-1} \Delta x = 0 \text{ iff } \Delta x \in column(PH^T)$$

Kalman Filter (A More Complicated Case)

Assuming that Δx is linear in $\nu = y - H\hat{x}_-$

$$\Delta x = PH^T K \nu$$

The condition $y = H(\hat{x}_- + \Delta x) \Rightarrow H\Delta x = y - H\hat{x}_- = \nu$

Substitution yields :

$$H\Delta x = \nu = HPH^T K \nu$$

$$\Rightarrow K = \left(HPH^T\right)^{-1}$$

$$\therefore \quad \Delta x = PH^T \left(HPH^T\right)^{-1} \nu$$

Kalman Filter (A More Complicated Case)

Hence, we can create a better state observer following the same 3 steps, but with the additional involvement of covariance matrix P.

Step 1: Prediction

$$\hat{x}(k+1 \mid k) = F\hat{x}(k \mid k) + Gu(k)$$

P also needs to be updated as:

$$P(k \mid k) = E\left((x(k) - \hat{x}(k \mid k))(x(k) - \hat{x}(k \mid k))^T\right)$$

and

$$P(k+1 \mid k) = E\left((x(k+1) - \hat{x}(k+1 \mid k))(x(k+1) - \hat{x}(k+1 \mid k))^T\right)$$

Kalman Filter (A More Complicated Case)

$$
\begin{aligned}
P_{k+1}^{-} &= E\left((x_{k+1} - \hat{x}_{k+1}^{-})(x_{k+1} - \hat{x}_{k+1}^{-})^T\right) \\
&= E\left((Fx_k + Gu_k + v_k - (F\hat{x}_k + Gu_k))(Fx_k + Gu_k + v_k - (F\hat{x}_k + Gu_k))^T\right) \\
&= E\left((F(x_k - \hat{x}_k) + v_k)(F(x_k - \hat{x}_k) + v_k)^T\right) \\
&= E\left(F(x_k - \hat{x}_k)(x_k - \hat{x}_k)^T F^T + 2F(x_k - \hat{x}_k)v_k^T + v_k v_k^T\right) \\
&= FE\left((x_k - \hat{x}_k)(x_k - \hat{x}_k)^T\right)F^T + E\left(v_k v_k^T\right) \\
&= FP_k F^T + Q
\end{aligned}
$$

$$
\boxed{P(k+1 \mid k) = FP(k \mid k)F^T + Q}
$$

Kalman Filter (A More Complicated Case)

From step 1 we get $\hat{x}(k+1|k)$ and $P(k+1|k)$.

Now we use these to compute Δx :

$$\Delta x = P(k+1|k)H\left(HP(k+1|k)H^T\right)^{-1}\left(y(k+1) - H\hat{x}(k+1|k)\right)$$

For ease of notation, define W so that

$$\Delta x = W\nu$$

Kalman Filter (A More Complicated Case)

$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + W\nu$$

$$P_{k+1} = E\left((x_{k+1} - \hat{x}_{k+1})(x_{k+1} - \hat{x}_{k+1})^T\right)$$

$$= E\left((x_{k+1} - \hat{x}^-_{k+1} - W\nu)(x_{k+1} - \hat{x}^-_{k+1} - W\nu)^T\right)$$

(many steps of linear equation manipulations …)

$$P(k+1 \mid k+1) = P(k+1 \mid k) - WHP(k+1 \mid k)H^T W^T$$

Kalman Filter (A More Complicated Case)

Many steps of linear equation manipulations …

$$
\begin{aligned}
P_{k+1} &= E\left((x_{k+1}-\hat{x}_{k+1})(x_{k+1}-\hat{x}_{k+1})^T\right) \\
&= E\left((x_{k+1}-\hat{x}^-_{k+1}-W\nu)(x_{k+1}-\hat{x}^-_{k+1}-W\nu)^T\right) \\
&= E\left(\left((x_{k+1}-\hat{x}^-_{k+1})-W\nu\right)\left((x_{k+1}-\hat{x}^-_{k+1})-W\nu\right)^T\right) \\
&= E\left((x_{k+1}-\hat{x}^-_{k+1})(x_{k+1}-\hat{x}^-_{k+1})^T-2W\nu(x_{k+1}-\hat{x}^-_{k+1})^T+W\nu(W\nu)^T\right) \\
&= P^-_{k+1}+E\left(-2WH(x_{k+1}-\hat{x}^-_{k+1})(x_{k+1}-\hat{x}^-_{k+1})^T+WH(x_{k+1}-\hat{x}^-_{k+1})(x_{k+1}-\hat{x}^-_{k+1})^T H^T W^T\right) \\
&= P^-_{k+1}-2WHP^-_{k+1}+WHP^-_{k+1}H^T W^T \\
&= P^-_{k+1}-2P^-_{k+1}H^T\left(HP^-_{k+1}H^T\right)^{-1}HP^-_{k+1}+WHP^-_{k+1}H^T W^T \\
&= P^-_{k+1}-2P^-_{k+1}H^T\left(HP^-_{k+1}H^T\right)^{-1}\left(HP^-_{k+1}H^T\right)\left(HP^-_{k+1}H^T\right)^{-1}HP^-_{k+1}+WHP^-_{k+1}H^T W^T \\
&= P^-_{k+1}-2WHP^-_{k+1}H^T W^T+WHP^-_{k+1}H^T W^T
\end{aligned}
$$

Kalman Filter (A More Complicated Case)

System:

$$x(k+1) = Fx(k) + Gu(k) + v(k)$$
$$y(k) = Hx(k)$$

1. Predict

$$\hat{x}(k+1|k) = F\hat{x}(k|k) + Gu(k)$$
$$P(k+1|k) = FP(k|k)F^T + Q$$

2. Correction

$$W = P(k+1|k)H\left(HP(k+1|k)H^T\right)^{-1}$$
$$\Delta x = W\left(y(k+1) - H\hat{x}(k+1|k)\right)$$

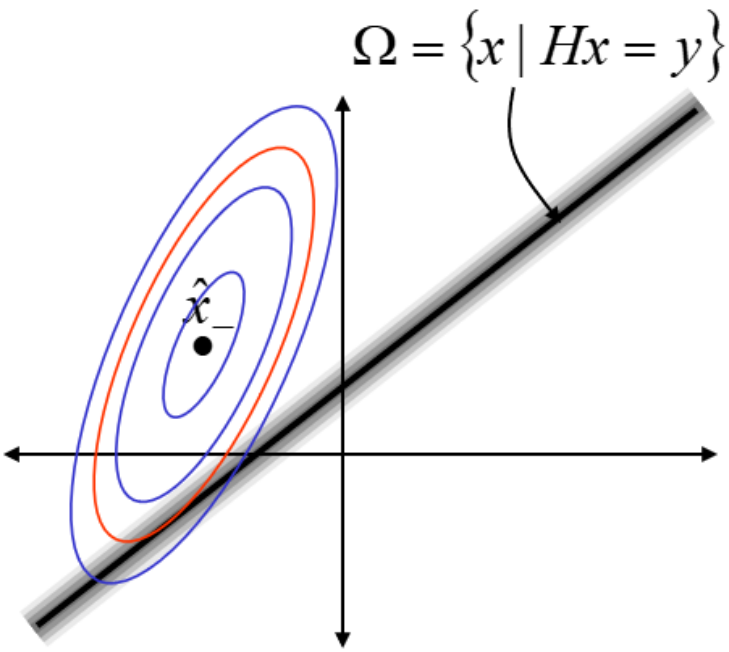
3. Update

$$\hat{x}(k+1|k+1) = \hat{x}(k+1|k) + W\nu$$
$$P(k+1|k+1) = P(k+1|k) - WHP(k+1|k)H^TW^T$$

Kalman Filter (The Ultimate General Case)

If measurement has noises too …

$$y = Hx + w$$



You do not have a hyperplane to directly aim for, you can't solve this with simple algebra!

You must solve an optimization problem.

The answer:

$$\Delta x = PH^T\left(HPH^T + R\right)^{-1}\left(y - H\hat{x}_-\right)$$

Kalman Filter (The Ultimate General Case)

System:

$$x(k+1) = Fx(k) + Gu(k) + v(k)$$
$$y(k) = Hx(k) + w(k)$$

1. Predict

$$\hat{x}(k+1 \mid k) = F\hat{x}(k \mid k) + Gu(k)$$
$$P(k+1 \mid k) = FP(k \mid k)F^T + Q$$

2. Correction

$$S = HP(k+1 \mid k)H^T + \mathrm{R}$$
$$W = P(k+1 \mid k)HS^{-1}$$
$$\Delta x = W\left(y(k+1) - H\hat{x}(k+1 \mid k)\right)$$

3. Update

$$\hat{x}(k+1 \mid k+1) = \hat{x}(k+1 \mid k) + W\nu$$
$$P(k+1 \mid k+1) = P(k+1 \mid k) - WSW^T$$

Extended Kalman Filter

The extended Kalman filter is used for system state estimation with nonlinear dynamics or is usually seen as the nonlinear version of Kalman filter. We consider a given system characterized by

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$$

and

$$\mathbf{y}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{v}_k,$$

where $\mathbf{x}_k$ is the true state vector at time k, $\mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1})$ and $\mathbf{h}(\mathbf{x}_k)$ are some nonlinear (or linear) functions, and $\mathbf{w}$ and $\mathbf{v}$ are the process and measurement noises. The noises, $\mathbf{w}$ and $\mathbf{v}$, are assumed to be independent, zero mean, and with normal probability distributions

$$\mathbf{w}_k \sim \mathrm{N}(0, \mathbf{Q}_{\mathrm{cov}})$$

and

$$\mathbf{v}_k \sim \mathrm{N}(0, \mathbf{R}_{\mathrm{cov}}),$$

where $\mathbf{Q}_{\mathrm{cov}}$ and $\mathbf{R}_{\mathrm{cov}}$ are the process and measurement noise covariance matrices.

Extended Kalman Filter

The computation process of Extended Kalman Filter involve two steps:

1. **Prediction Step**: The prediction step uses the previous time step to produce an estimation of the state at the current time step. In this stage we have the predicted state estimate and the predicted estimated covariance as follows:

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1})$$

and

$$\mathbf{P}_{k|k-1} = \mathbf{F}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{F}_{k-1}^{\mathrm{T}} + \mathbf{Q}_{\mathrm{cov}}$$

where $\mathbf{F}_{k-1}$, the Jacobian matrix of the partial derivative of the function $\mathbf{f}$ with respect to $\mathbf{x}$ at time $k-1$, is defined by

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

Extended Kalman Filter

2. Correction Step: In the correction step, the current predicted estimate is combined with current observation information to refine the state estimate. At time instance k, we need to compute the Kalman gain $\mathbf{K}_k$ and then update the state estimate and error covariance as follows:

$$\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}}(\mathbf{H}_k\mathbf{P}_{k|k-1}\mathbf{H}_k^{\mathrm{T}} + \mathbf{R}_{\mathrm{cov}})^{-1},$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_{k|k-1}),$$

and

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_{k|k-1},$$

where $\mathbf{H}_k$, the Jacobian matrix of the partial derivative of the function **h** with respect to **x** at time k, is defined by

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}_{k|k-1}}$$

EKF: Attitude and Heading Reference System

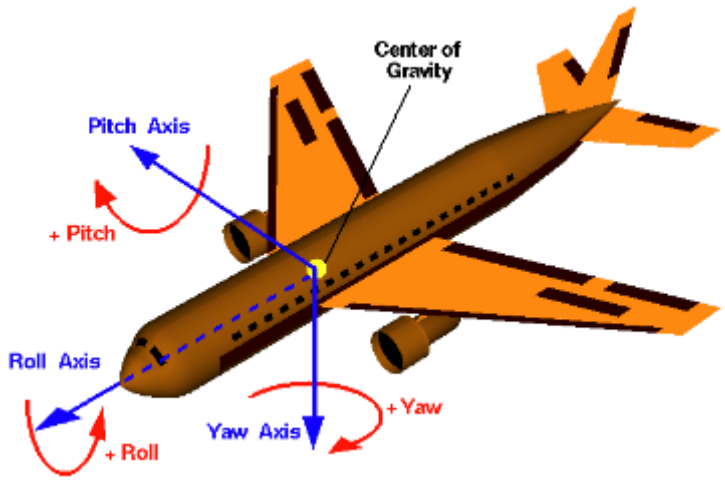
Euler angles

- roll, pitch, yaw
- no redundancy
- gimbal lock singularities

Angular rate: $\boldsymbol{\omega}$

Roll
Pitch
Yaw

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \underbrace{\begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}}_{\mathbf{H}} \begin{pmatrix} p \\ q \\ r \end{pmatrix}$$



where $$\sec\theta = \frac{1}{\cos\theta}$$

$$\boldsymbol{\Phi}(t_k) = \boldsymbol{\Phi}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \mathbf{H}\boldsymbol{\omega}\mathrm{d}t$$

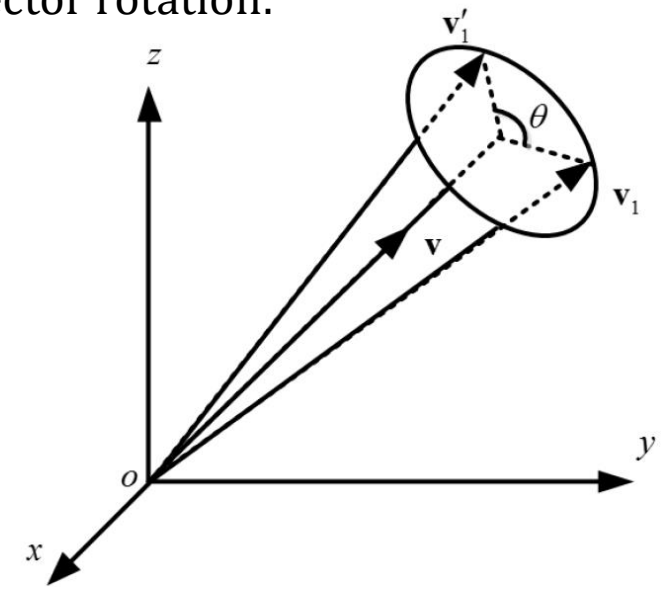
EKF: Attitude and Heading Reference System

Quaternions

- Generally considered the "best" representation;
- Redundant (4 values), but only by one DOF;
- Stable interpolations of rotations possible;
- A rotation is represented by a unit quaternion, which is a four-element unit vector:

$$\mathbf{q} = (q_0, q_1, q_2, q_3) = (s_q, \mathbf{v}_q)$$
$$q_0^2 + q_1^2 + q_3^2 + q_4^2 = 1$$

Vector rotation:



$$q_0 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2}$$

$$q_1 = \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2}$$

$$q_2 = \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2}$$

$$q_3 = \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2}$$

$$\begin{bmatrix} 0 \\ \mathbf{v}_1' \end{bmatrix} = \mathbf{q} \otimes \begin{bmatrix} 0 \\ \mathbf{v}_1 \end{bmatrix} \otimes \mathbf{q}^{-1}$$

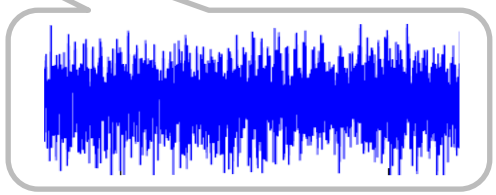
where

$$\mathbf{q} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \mathbf{v}\sin\frac{\theta}{2} \end{bmatrix}$$

EKF: Attitude and Heading Reference System

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{pmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -p & -q & -r \\ p & 0 & r & -q \\ q & -r & 0 & p \\ r & q & -p & 0 \end{bmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix}$$

$$\dot{\mathbf{q}} = \frac{1}{2}\mathbf{\Omega}\mathbf{q}$$



Discretization

$$\mathbf{q}(t_k) = \mathbf{q}(t_{k-1}) + \int_{t_{k-1}}^{t_k} \frac{1}{2}\mathbf{\Omega}\mathbf{q}\mathrm{d}t$$

Re-normalization

$$q_0^2 + q_1^2 + q_3^2 + q_4^2 = 1$$

EKF: Attitude and Heading Reference System

Augmentation with gyro bias:

$$\begin{pmatrix} \dot{q}_0 \\ \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \\ \dot{b}_\mathrm{p} \\ \dot{b}_\mathrm{q} \\ \dot{b}_\mathrm{r} \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \begin{bmatrix} 0 & b_\mathrm{p} - p & b_\mathrm{q} - q & b_\mathrm{r} - r \\ p + b_\mathrm{p} & 0 & r - b_\mathrm{r} & b_\mathrm{q} - q \\ q + b_\mathrm{q} & b_\mathrm{r} - r & 0 & p - b_\mathrm{p} \\ r + b_\mathrm{r} & q - b_\mathrm{q} & b_\mathrm{p} - p & 0 \end{bmatrix} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} \\ \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \end{pmatrix}$$

EKF: Attitude and Heading Reference System

Constrained by accelerometer measurement:

$$\begin{pmatrix} f_{\mathrm{u}} \\ f_{\mathrm{v}} \\ f_{\mathrm{w}} \end{pmatrix} = \begin{pmatrix} -2g(q_1q_3 - q_0q_2) \\ -2g(q_0q_1 + q_2q_3) \\ -g(q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}$$

Acceleration

Constrained by magnetometer measurement:

$$\psi = \arctan \frac{2(q_1q_2 + q_0q_3)}{q_0^2 + q_1^2 - q_2^2 - q_3^2}$$

Heading

EKF: Attitude and Heading Reference System

$$
\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{f}_{\mathrm{c}}(\mathbf{x}, \boldsymbol{\omega}, \mathbf{w}) & p(\mathbf{w}) &\sim \mathrm{N}(0, \mathbf{Q}) \\
\mathbf{y} &= \mathbf{h}_{\mathrm{c}}(\mathbf{x}, \mathbf{v}) & p(\mathbf{v}) &\sim \mathrm{N}(0, \mathbf{R})
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{x}_k &= \mathbf{x}_{k-1} + \int_{t_{k-1}}^{t_k} \mathbf{f}_{\mathrm{c}}(\mathbf{x}_{k-1}, \boldsymbol{\omega}_{k-1}, \mathbf{w}_{k-1}) \mathrm{d}t \\
&= \mathbf{f}_{\mathrm{d}}(\mathbf{x}_{k-1}, \boldsymbol{\omega}_{k-1}, \mathbf{w}_{k-1}) \\
\mathbf{y}_k &= \mathbf{h}_{\mathrm{d}}(\mathbf{x}_k, \mathbf{v}_k)
\end{aligned}
$$

EKF: Attitude and Heading Reference System

$$
\begin{aligned}
\dot{\mathbf{x}} &= \mathbf{f}_c(\mathbf{x}, \boldsymbol{\omega}, \mathbf{w}) & p(\mathbf{w}) &\sim \mathrm{N}(0, \mathbf{Q}) \\
\mathbf{y} &= \mathbf{h}_c(\mathbf{x}, \mathbf{v}) & p(\mathbf{v}) &\sim \mathrm{N}(0, \mathbf{R})
\end{aligned}
$$

$$
\mathbf{A} = \frac{\partial \mathbf{f}_c}{\partial \mathbf{x}} \quad \mathbf{W} = \frac{\partial \mathbf{f}_c}{\partial \mathbf{w}} \quad \mathbf{C} = \frac{\partial \mathbf{h}_c}{\partial \mathbf{x}} \quad \mathbf{V} = \frac{\partial \mathbf{h}_c}{\partial \mathbf{v}}
$$

$$
\Downarrow \qquad \Downarrow \qquad \Downarrow \qquad \Downarrow
$$

$$
\mathbf{F}_k \qquad \mathbf{W}_k \qquad \mathbf{H}_k \qquad \mathbf{V}_k
$$

EKF: Attitude and Heading Reference System

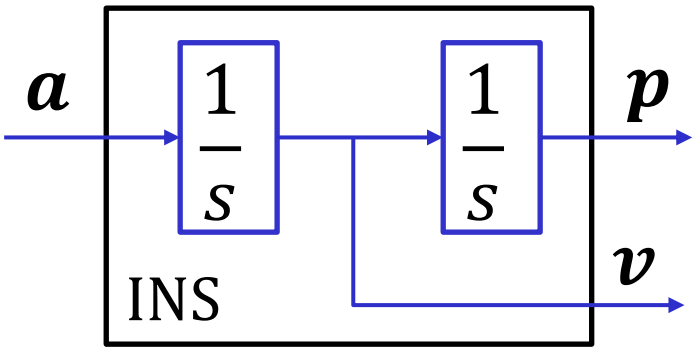
- Time update (prediction):

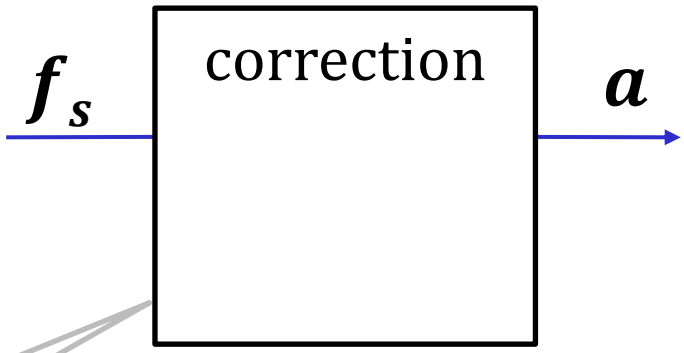
$$\begin{aligned}\hat{\mathbf{x}}_{k|k-1} &= \mathbf{f}_{\mathrm{d}}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_{k-1}, 0) \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k' + \mathbf{W}_k \mathbf{Q}_k \mathbf{W}_k'\end{aligned}$$

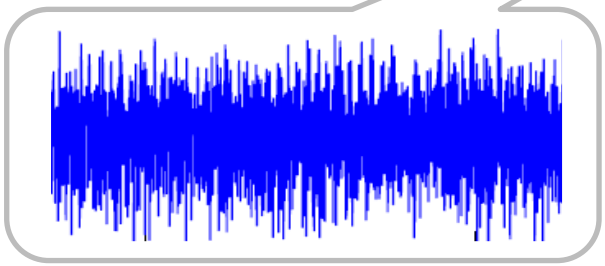
- Measurement update (correction):

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_{k|k-1} \mathbf{H}_k' (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k' + \mathbf{V}_k \mathbf{R}_k \mathbf{V}_k')^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k [\mathbf{y}_k - \mathbf{h}_{\mathrm{d}}(\hat{\mathbf{x}}_{k|k-1})] \\ \mathbf{P}_{k|k} &= \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1}\end{aligned}$$

EKF: Inertial Navigation System



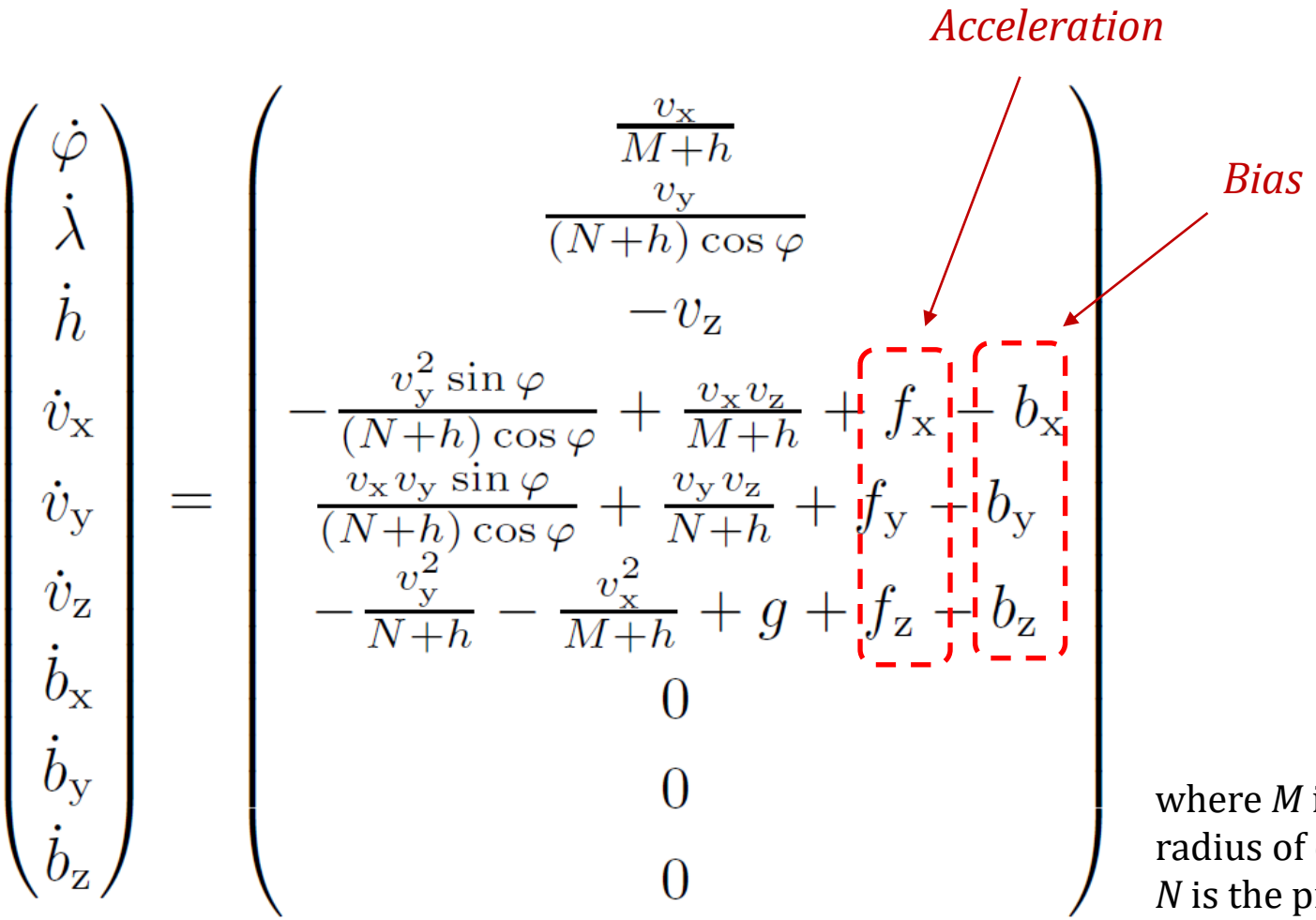




$$\boldsymbol{p} = (\varphi \quad \lambda \quad h)$$

$$\boldsymbol{v} = (v_{\mathrm{x}} \quad v_{\mathrm{y}} \quad v_{\mathrm{z}})$$

EKF: Inertial Navigation System



where M is the meridian radius of curvature and N is the prime vertical radius of curvature

EKF: Inertial Navigation System

If a strapdown IMU is used,

$$\begin{pmatrix} f_{\mathrm{x}} \\ f_{\mathrm{y}} \\ f_{\mathrm{z}} \end{pmatrix} = \mathbf{R}_{\mathrm{nb}} \begin{pmatrix} f_{\mathrm{u}} \\ f_{\mathrm{v}} \\ f_{\mathrm{w}} \end{pmatrix}$$

$$\begin{aligned} \mathbf{R}_{\mathrm{nb}} &= \mathbf{R}_{\psi}\mathbf{R}_{\theta}\mathbf{R}_{\phi} \\ &= \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \end{aligned}$$

EKF: Inertial Navigation System

Process Model:

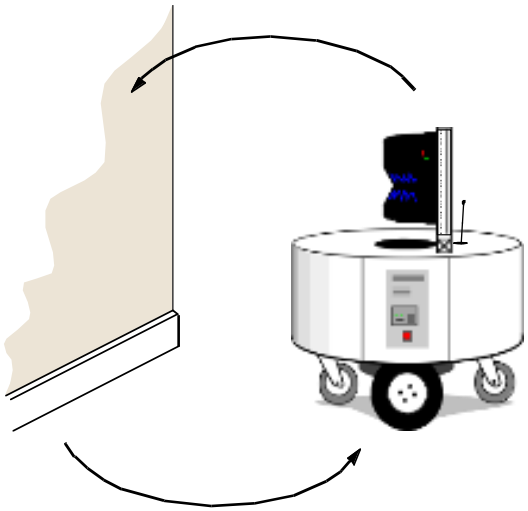
$$\begin{pmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \\ \dot{v}_\mathrm{x} \\ \dot{v}_\mathrm{y} \\ \dot{v}_\mathrm{z} \\ \dot{b}_\mathrm{x} \\ \dot{b}_\mathrm{y} \\ \dot{b}_\mathrm{z} \end{pmatrix} = \begin{pmatrix} \frac{v_\mathrm{x}}{M+h} \\ \frac{v_\mathrm{y}}{(N+h)\cos\varphi} \\ -v_\mathrm{z} \\ -\frac{v_\mathrm{y}^2 \sin\varphi}{(N+h)\cos\varphi} + \frac{v_\mathrm{x} v_\mathrm{z}}{M+h} + f_\mathrm{x} - b_\mathrm{x} \\ \frac{v_\mathrm{x} v_\mathrm{y} \sin\varphi}{(N+h)\cos\varphi} + \frac{v_\mathrm{y} v_\mathrm{z}}{N+h} + f_\mathrm{y} - b_\mathrm{y} \\ -\frac{v_\mathrm{y}^2}{N+h} - \frac{v_\mathrm{x}^2}{M+h} + g + f_\mathrm{z} - b_\mathrm{z} \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Measurement Model:

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{v} \end{pmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{p} \\ \mathbf{v} \\ \mathbf{b} \end{pmatrix}$$

What is SLAM?

SLAM is the process by which a robot or vehicle **builds a map** of the environment and, at the same time, uses this map to **compute its location**



- SLAM is a chicken-or-egg problem:
  → A map is needed for localizing a robot
  → A pose estimate is needed to build a map

- SLAM is **the most fundamental problem** for robots to become **autonomous**

- The history of SLAM dates to the **mid-eighties** (stone-age of mobile robotics)

Mathematical Model of SLAM

Given:

- The robot's controls

$$\mathbf{U}_{0:k} = \{\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k\}$$

- Relative observations

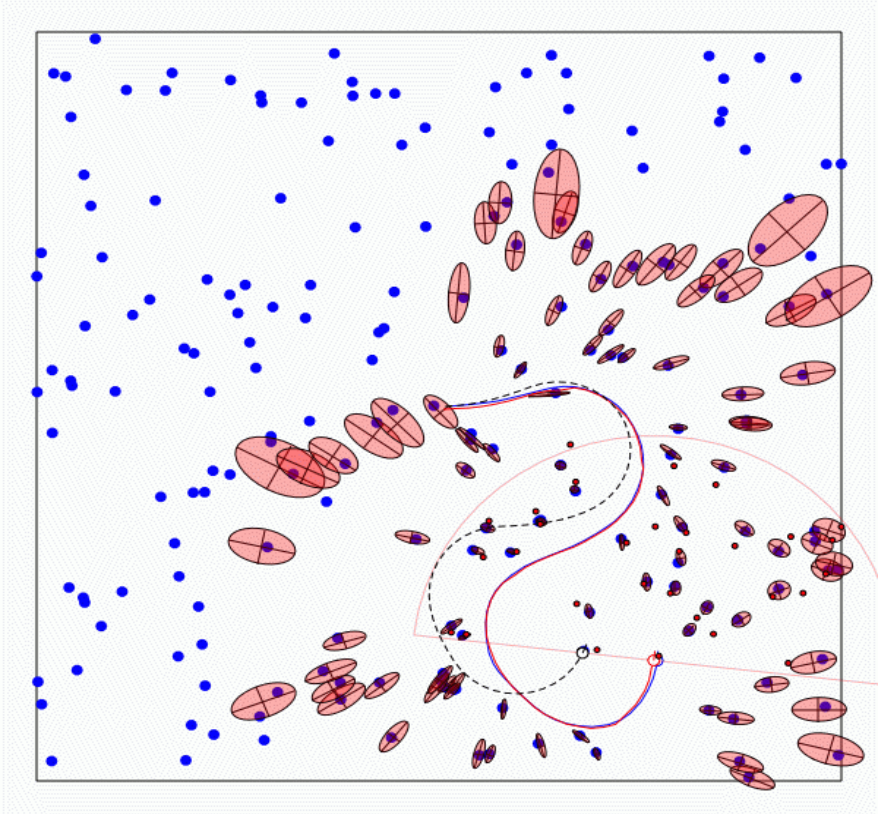
Compute:

- Map of features

$$\mathbf{m} = \{\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n\}$$

- Path of the robot

$$\mathbf{X}_{0:k} = \{\mathbf{x}_0, \mathbf{x}_1, \cdots, \mathbf{x}_k\}$$



Mathematical Model of SLAM

- **Full SLAM:** Estimates entire path and map!

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t})$$

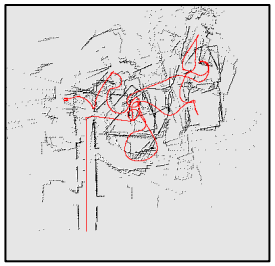
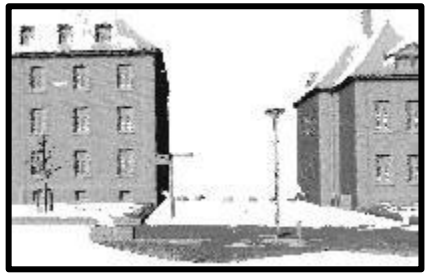
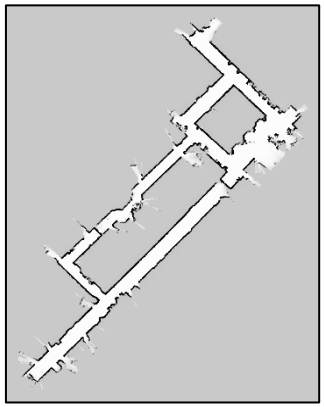
- **Online SLAM:** Estimates most recent pose and map!

$$p(x_t, m \mid z_{1:t}, u_{1:t}) = \int\int \dots \int p(x_{1:t}, m \mid z_{1:t}, u_{1:t})\, dx_1 dx_2 \dots dx_{t-1}$$

Integrations (marginalization) typically done one at a time

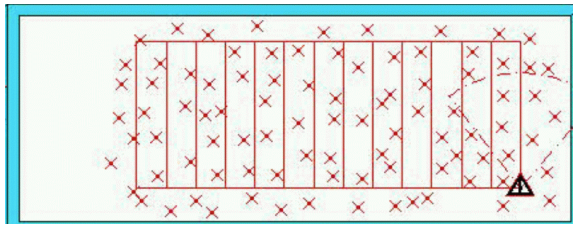
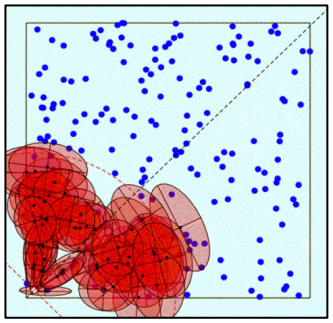
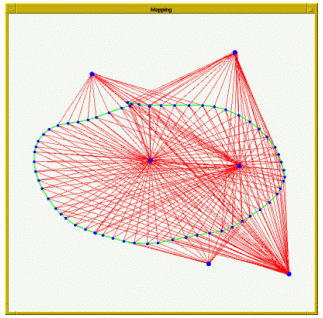
Map Representation of SLAM

- Grid maps or scans

[Lu & Milios, 97; Gutmann, 98: Thrun 98; Burgard, 99; Konolige & Gutmann, 00; Thrun, 00; Arras, 99; Haehnel, 01;…]

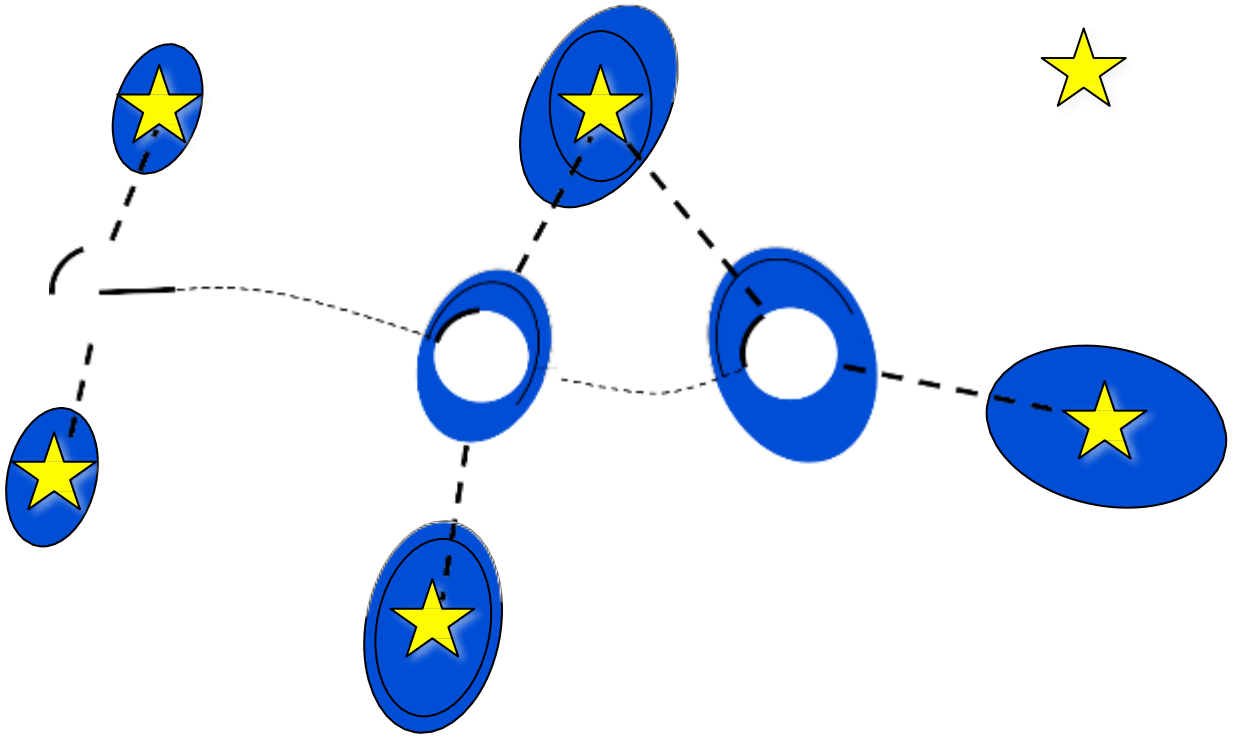
- Landmark-based

[Leonard et al., 98; Castelanos et al., 99: Dissanayake et al., 2001; Montemerlo et al., 2002;…

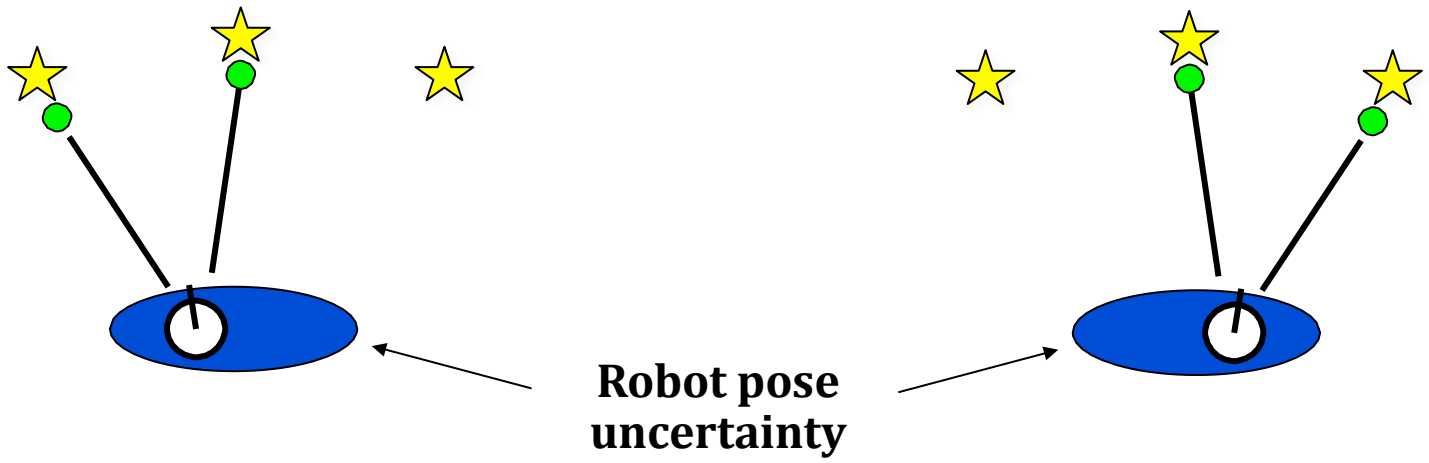
Why is SLAM Challenging?

SLAM: robot position and map are both unknown



Robot position error correlates errors in the map

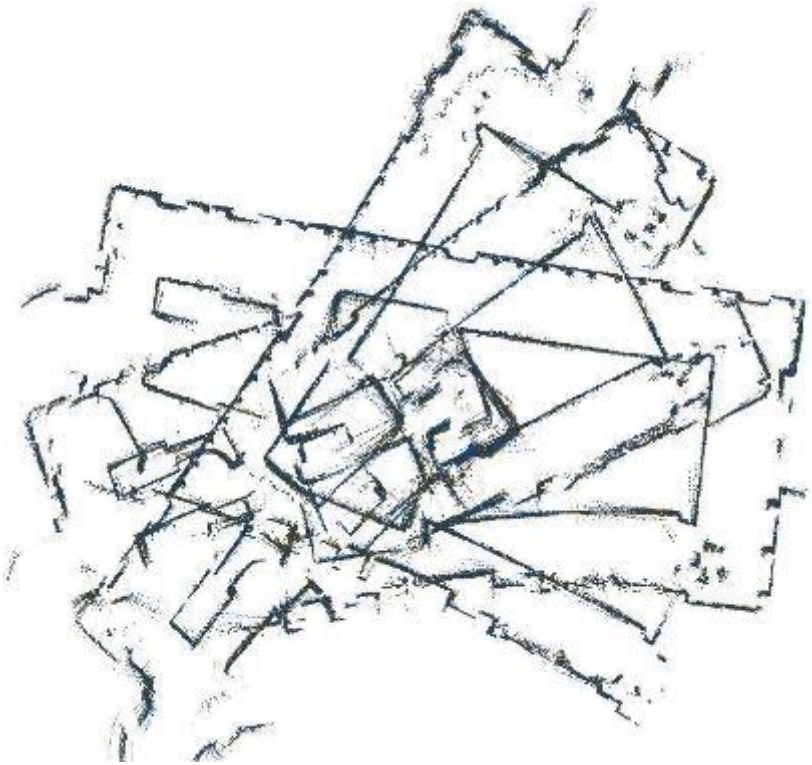
Why is SLAM Challenging?

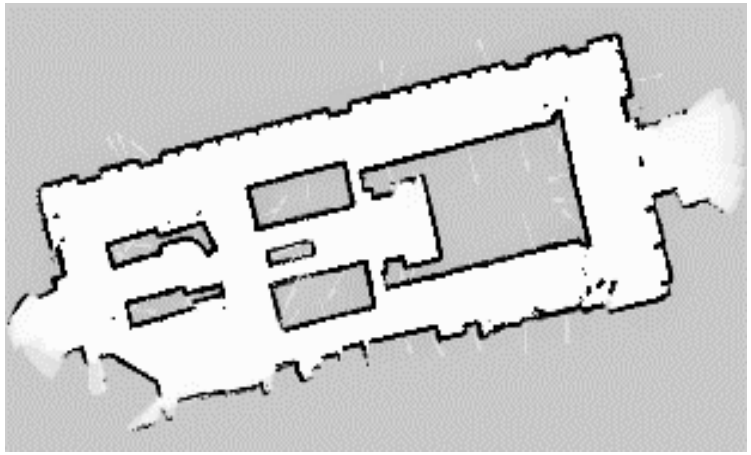


- In practice, the mapping between observations and landmarks is unknown
- Picking wrong data associations can have catastrophic consequences
- Pose error correlates data associations

Why is SLAM Challenging?

- Small local error accumulate to large global errors
- This is usually tolerable for navigation
- However, when **closing loops**, global error does matter

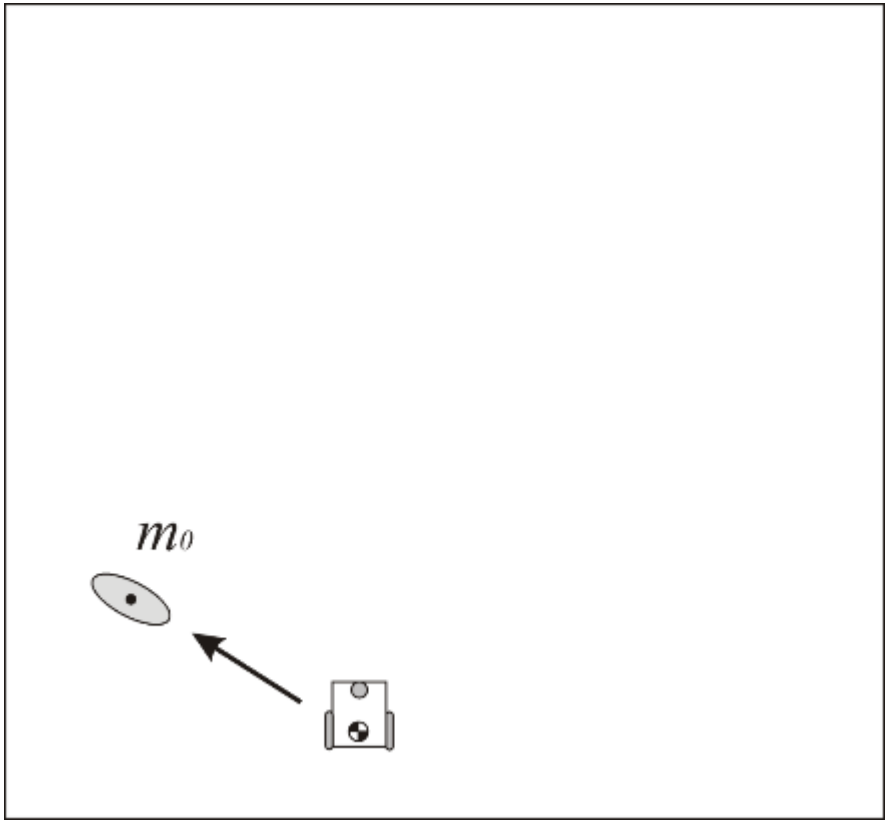




A Typical Simplified SLAM Process

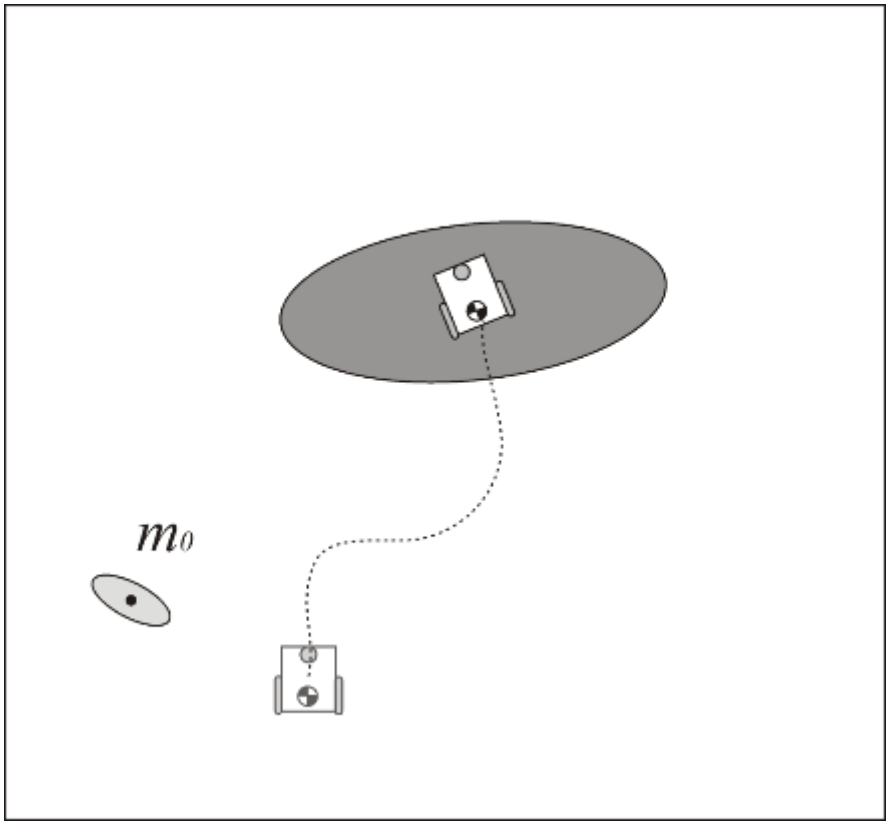
- Assume that the robot uncertainty at its initial location is zero.

- From this position, the robot observes a feature which is mapped with an uncertainty related to the exteroceptive sensor error model



A Typical Simplified SLAM Process

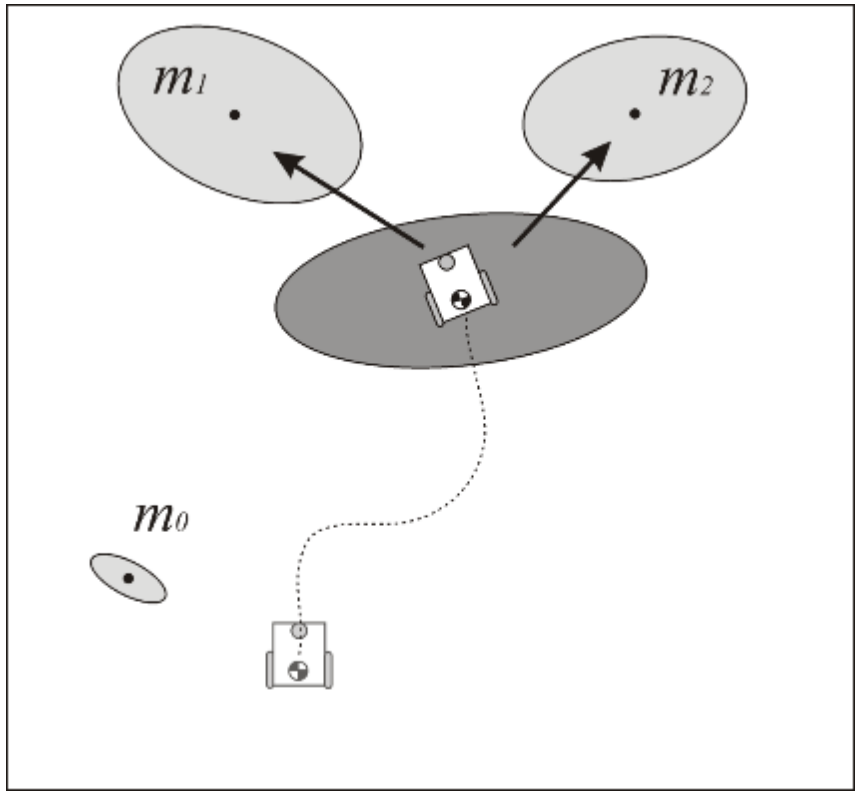
- As the robot moves, its pose uncertainty increases under the effect of the errors introduced by its odometry



A Typical Simplified SLAM Process

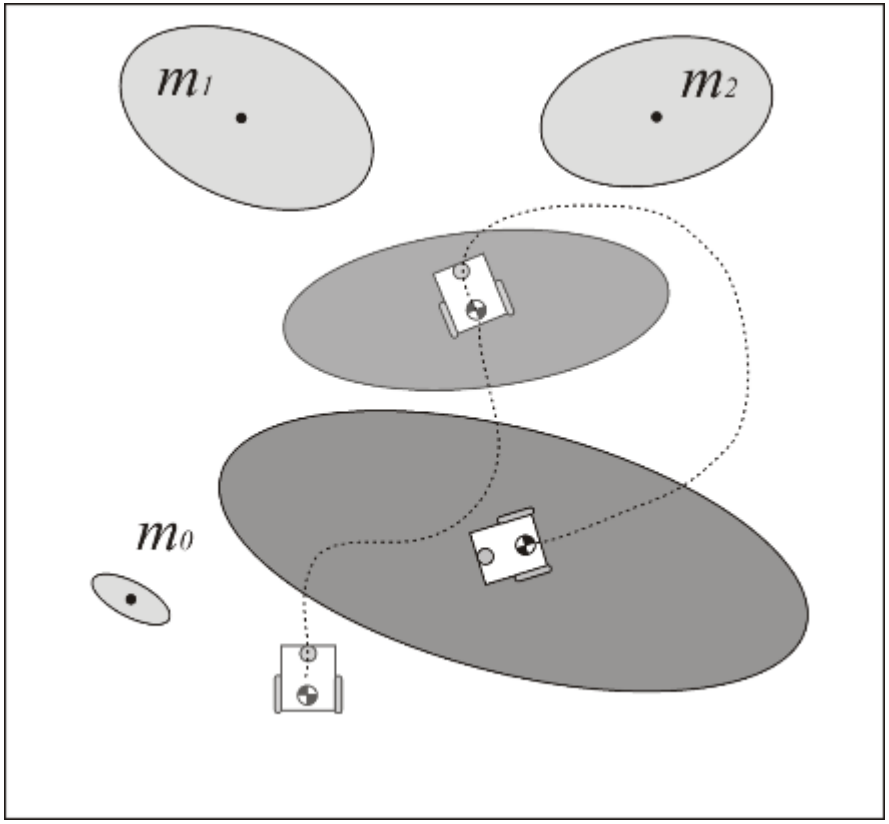
- At this point, the robot observes two features and maps them with an uncertainty which results from the combination of measurement error with the robot pose and sensor uncertainty

- From this, we can notice that the map becomes correlated with the robot position estimate. Similarly, if the robot updates its position based on an observation of an imprecisely known feature in the map, the resulting position estimate becomes correlated with the feature location estimate.



A Typical Simplified SLAM Process

- The robot moves again and its uncertainty increases under the effect of the errors introduced by the odometry

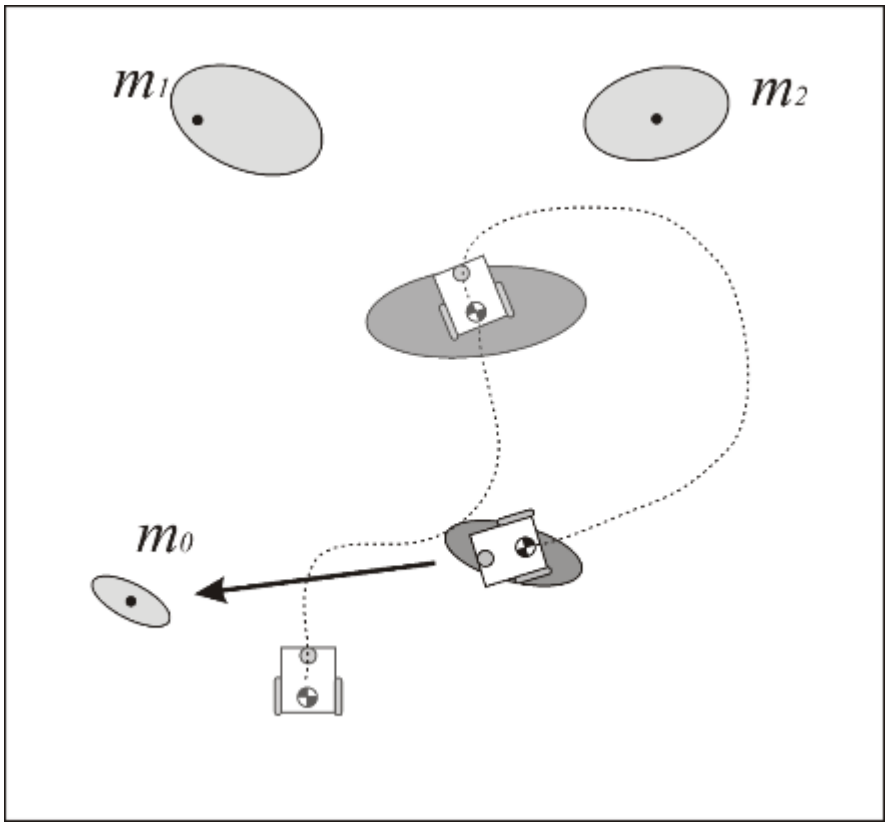


A Typical Simplified SLAM Process

- In order to reduce its uncertainty, the robot must observe features whose location is relatively well known. These features can be landmarks that the robot has already observed before.

- Such observation is called *loop closure detection.*

- When a loop closure is detected, the robot pose uncertainty shrinks. At the same time, the map is updated and the uncertainty of other observed features also reduce



SLAM Paradigms

Most SLAM algorithms fall into three categories:

1. **SLAM based on Extended Kalman Filter (EKF SLAM)**
2. **SLAM based on Particle Filter (FastSLAM)**
3. **SLAM based on Graph Optimization (GraphSLAM)**

EKF SLAM: Overview

- The EKF SLAM proceeds exactly like the standard EKF that we have seen for robot localization, with the only difference that it uses an **extended state vector $\boldsymbol{y_t}$** which comprises both the robot pose $\boldsymbol{x_t}$ and the position of all the features $\boldsymbol{m_i}$ in the map, that is:

$$y_t = [x_t, m_1, \ldots, m_{n-1}]^T$$

- If we sense 2D line-landmarks, the size of $\boldsymbol{y_t}$ would be *3+2n*, since we need three variables to represent the robot pose and *2n* variables for the *n* line-landmarks having vector components (α_i, r_i)

$$y_t = [x_t, y_t, \theta_t, \alpha_0, r_0, \ldots, \alpha_{n-1}, r_{n-1}]^T$$

- As the robot moves and takes measurements, the state vector and covariance matrix are updated using the standard equations of the extended Kalman filter

EKF SLAM: Prediction Step

- For the prediction step, the robot pose is updated using the odometric position update formula

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \end{bmatrix}$$

- The position of the features, will conversely remain unchanged. Hence, we can write the prediction model of the EKF SLAM as

$$\begin{bmatrix} \hat{x}_t \\ \hat{y}_t \\ \hat{\theta}_t \\ \hat{\alpha}_0 \\ \hat{r}_0 \\ \dots \\ \hat{\alpha}_{n-1} \\ \hat{r}_{n-1} \end{bmatrix} = \begin{bmatrix} x_t \\ y_t \\ \theta_t \\ \alpha_0 \\ r_0 \\ \dots \\ \hat{\alpha}_{n-1} \\ r_{n-1} \end{bmatrix} + \begin{bmatrix} \frac{\Delta S_r + \Delta S_l}{2} \cos(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r + \Delta S_l}{2} \sin(\theta_{t-1} + \frac{\Delta S_r - \Delta S_l}{2b}) \\ \frac{\Delta S_r - \Delta S_l}{b} \\ 0 \\ 0 \\ \dots \\ 0 \\ 0 \end{bmatrix}$$

$$\hat{P}_t = F_y P_{t-1} F_y^T + F_u Q_t F_u^T$$

EKF SLAM: Measurement Update Step

- The observation model is the same as the normal EKF localization:

$$\hat{z}_i = \begin{bmatrix} \hat{\alpha}_i \\ \hat{r}_i \end{bmatrix} = h(x)$$

- The measurement update step is also in the same manner as the conventional EKF:

$$y_t = \hat{y}_t + K_t(z - h(x))$$

$$P_t = \hat{P}_t - K_t \Sigma_{IN} K_t^T$$

where

$$\Sigma_{IN} = HPH^T + R$$

$$K_t = PH(\Sigma_{IN})^{-1}$$

EKF SLAM: Other Insights

- At the beginning of the process, when the robot/vehicle takes the first measurements, the covariance matrix is populated by assuming that these (initial) features are uncorrelated, which implies that the off-diagonal elements are all zeros.

$$P_0 = \begin{bmatrix} P_x & 0 & 0 & \dots & 0 & 0 \\ 0 & P_{m_0} & 0 & \dots & 0 & 0 \\ 0 & 0 & P_{m_1} & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & \dots & P_{m_{n-2}} & 0 \\ 0 & 0 & 0 & \dots & 0 & P_{m_{n-1}} \end{bmatrix}$$

EKF SLAM: Other Insights

- However, when the robot starts to move and takes new measurements, both the robot pose and features start becoming correlated.

$$\hat{P}_t = F_y P_{t-1} F_y^T + F_u Q_t F_u^T$$

- Accordingly, the covariance matrix becomes *non-sparse*.

$$P_0 = \begin{bmatrix} P_x & P_{xm_0} & P_{xm_1} & \dots & P_{xm_{n-2}} & P_{xm_{n-1}} \\ P_{xm_0} & P_{m_0} & P_{m_0m_1} & \dots & P_{m_0m_{n-2}} & P_{m_0m_{n-1}} \\ P_{xm_1} & P_{m_0m_1} & P_{m_1} & \dots & P_{m_1m_{n-2}} & P_{m_1m_{n-1}} \\ \dots & \dots & \dots & \dots & \dots & \dots \\ P_{xm_{n-2}} & P_{m_0m_{n-2}} & P_{m_1m_{n-2}} & \dots & P_{m_{n-2}} & P_{m_1m_{n-1}} \\ P_{xm_{n-1}} & P_{m_0m_{n-1}} & P_{m_1m_{n-1}} & \dots & P_{m_{n-2}m_{n-1}} & P_{m_{n-1}} \end{bmatrix}$$

- The existence of this correlation can be explained by recalling that the uncertainty of the features in the map depends not only on the uncertainty associated to the robot pose, but also on the uncertainty of other features that have been used to update the robot pose. This means that when a new feature is observed this contributes to correct not only the estimate of the robot pose but also that of the other features. The more observations are made, the more the correlations between the features will grow, the better the solution to SLAM.

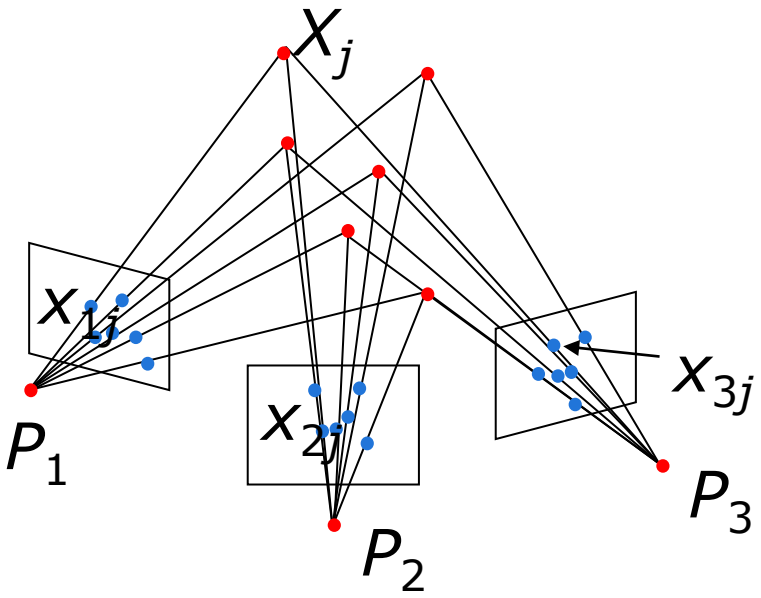
EKF SLAM: A Vision-Based Example

This example is based on the following two papers:

- Andrew J. Davison, Real-Time Simultaneous Localization and Mapping with a Single Camera, ICCV 2003.
- Nicholas Molton, Andrew J. Davison and Ian Reid, Locally Planar Patch Features for Real-Time Structure from Motion, BMVC 2004.

EKF SLAM: A Vision-Based Example

- Take several images of the same object
- Visual features are extracted from all frames and matched among them
- Both camera motion and 3D structure can be recovered by fusing the overall information via solving an optimization problem
- In the computer vision community, this process is called Structure From Motion (SFM)



EKF SLAM: A Vision-Based Example

Visual feature extraction and matching

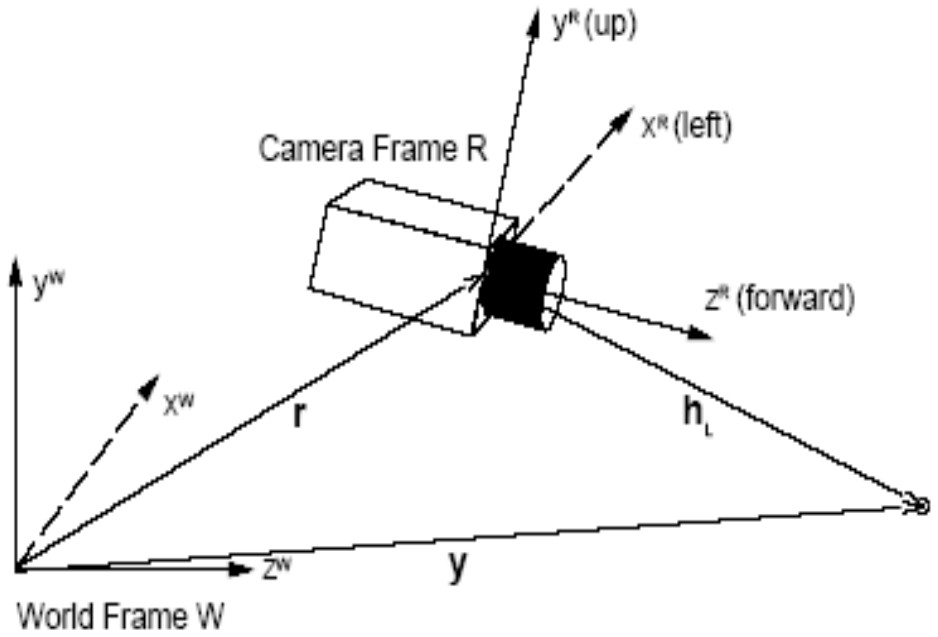
EKF SLAM: A Vision-Based Example

The state contains both the camera's pose and the 3D positions landmark points :

$$\hat{\mathbf{x}} = \begin{pmatrix} \hat{\mathbf{x}}_v \\ \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \vdots \end{pmatrix}$$

where

$$\mathbf{x}_v = \begin{pmatrix} \mathbf{r}^W \\ \mathbf{q}^{WR} \\ \mathbf{v}^W \\ \omega^W \end{pmatrix}$$



r is the camera position **r** = (x, y, z) and **q** is the camera orientation represented in quaternion. **v** and **ω** are the translational and rotational velocity.

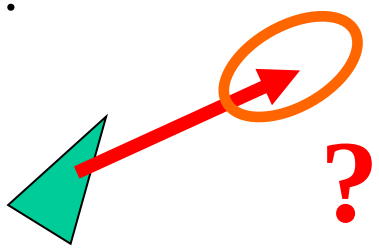
EKF SLAM: A Vision-Based Example

$$\hat{x}_t = f(x_{t-1}, u_t)$$

$$\hat{P}_t = F_y P_{t-1} F_y^T + F_u Q_t F_u^T$$

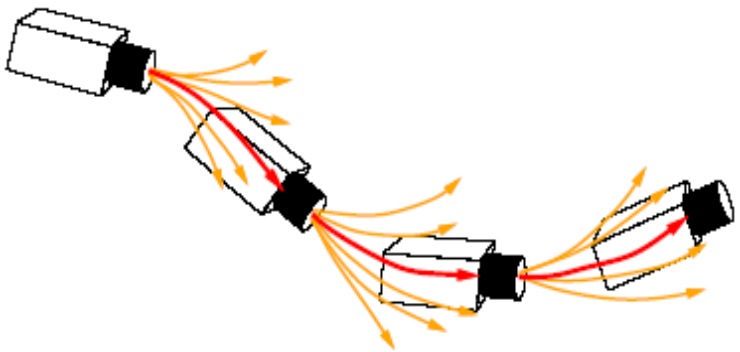
How do we compute the next camera position?

$$\hat{x}_t = f(x_{t-1}, u_t) = ?$$



Davison uses a **constant velocity model**, i.e. motion at time t-1 equals to motion at time t.

EKF SLAM: A Vision-Based Example



$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}_{new}^W \\ \mathbf{q}_{new}^{WR} \\ \mathbf{v}_{new}^W \\ \omega_{new}^W \end{pmatrix} = \begin{pmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^W + \mathbf{\Omega}^W)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ \omega^W + \mathbf{\Omega}^W \end{pmatrix}$$

Model the acceleration as a process of zero mean and Gaussian distribution:

$$\mathbf{n} = \begin{pmatrix} \mathbf{V}^W \\ \mathbf{\Omega}^W \end{pmatrix} = \begin{pmatrix} \mathbf{a}^W \Delta t \\ \alpha^W \Delta t \end{pmatrix}$$

By setting the covariance matrix of *n* to small or large values, we define the smoothness or rapidity of the motion we expect. In practice these values were used:

$$\begin{bmatrix} 4\ m/s \\ 6\ rad/s \end{bmatrix}$$

EKF SLAM: A Vision-Based Example

Observation: when a new feature is measured, both the camera pose and the covariance matrix will be updated as:

$$x_t = \hat{x}_t + K_t(z - h(x))$$

$$P_t = \hat{P}_t - K_t \Sigma_{IN} K_t^T$$

where

$$\Sigma_{IN} = HPH^T + R$$

$$K_t = PH(\Sigma_{IN})^{-1}$$

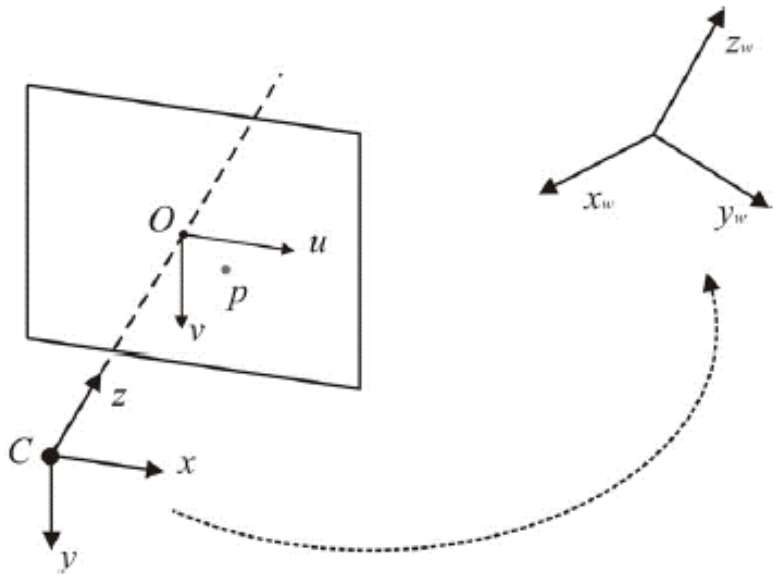
EKF SLAM: A Vision-Based Example

- By predicting the next camera pose, we can also predict where each features is going to appear:

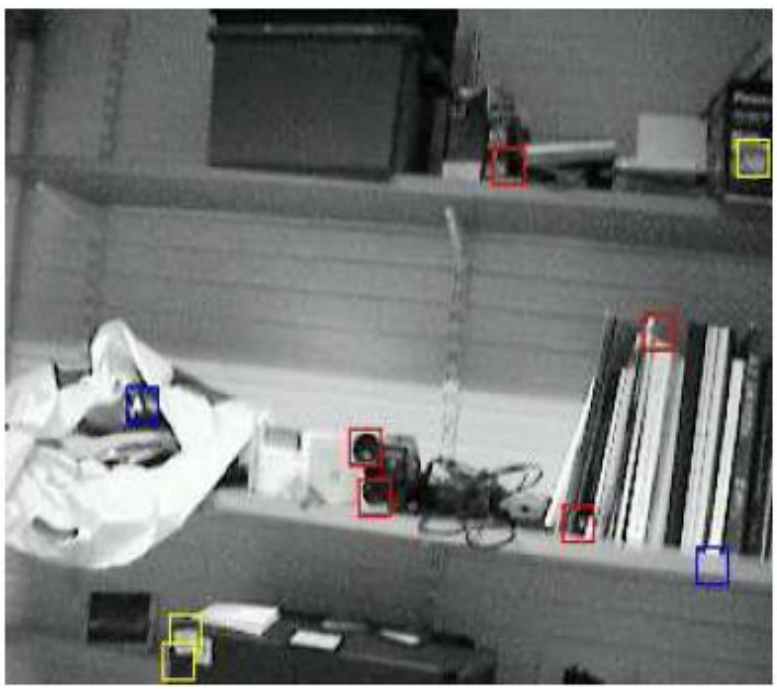
- To compute the predicted observation we need to compute *h(x)*

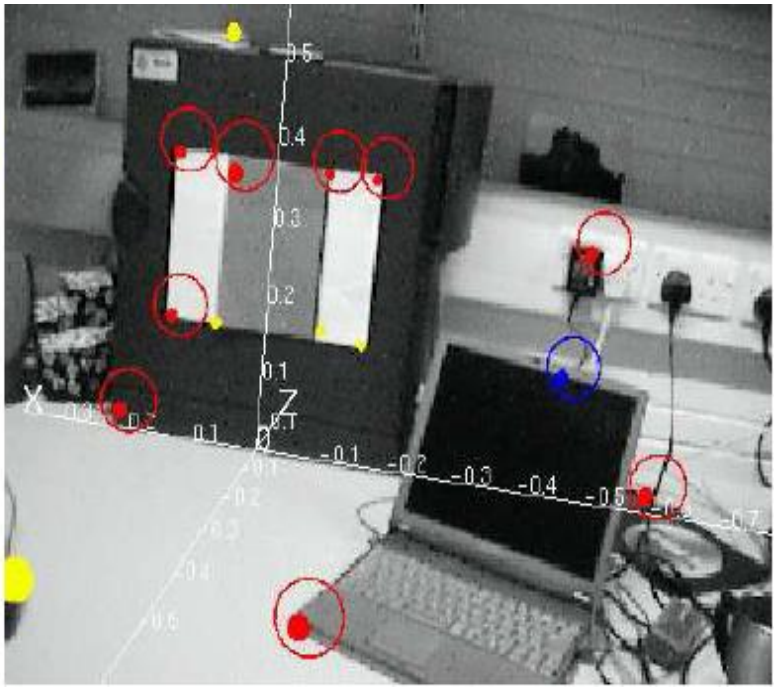
- *h(x) i*s the perspective projection equation:

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \cdot R \begin{bmatrix} x_w \\ y_w \\ z_w \end{bmatrix} + T$$



EKF SLAM: A Vision-Based Example





- At each frame, the features occurring at previous step are searched in the elliptic region where they are expected to be according to the motion model (Normalized Sum of Square Differences is used for matching)

- Once the features are matched, the entire state of the system is updated according to EKF

EKF SLAM: Drawbacks

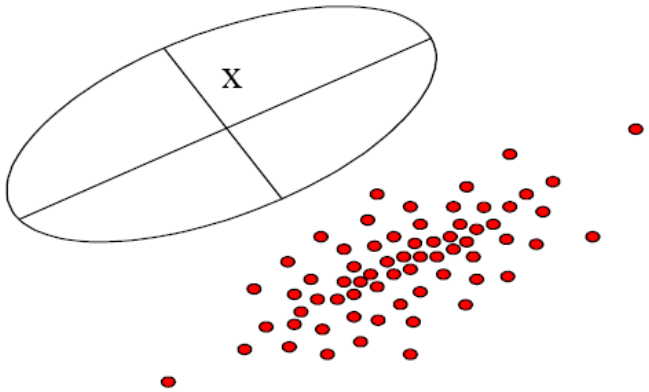
- The state vector in EKF SLAM is much larger than the state vector in EKF localization where only the robot pose was being updated. This makes EKF SLAM computationally very expensive.

- Maps in EKF SLAM are feature-based. As new features are observed, they are added to the state vector. Thus, the noise covariance matrix grows quadratically, with size $(3+2n)^2$. For computational reasons, the size of the map is therefore usually limited to less than 1,000 features.

- EKF assumes that all noises are Gaussian distributed. This assumption can be too strong for some practical cases.

FastSLAM

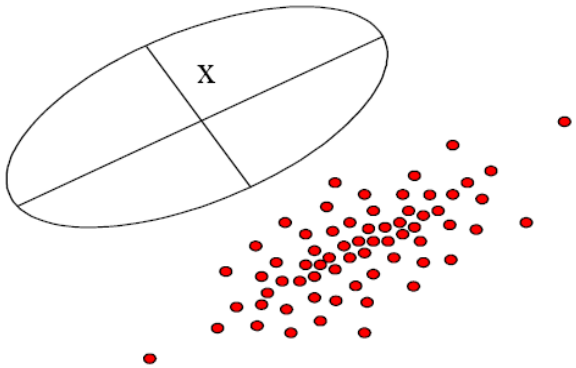
- It solves the SLAM problem using particle filters
- Each particle k: Estimate mean and covariance of robot path and each of the n features: $P^{[k]}\,(X_t^{[k]}\,\mu^{[k];}\,\Sigma_1^{[k]}\,\ldots\,\mu^{[k]}\,\Sigma_n^{[k]}\,)$



probability distribution (ellipse) as particle set (red dots)

Particle filter update

- In the update step a new particle distribution, given motion model and controls applied is generated.



a) For each particle:

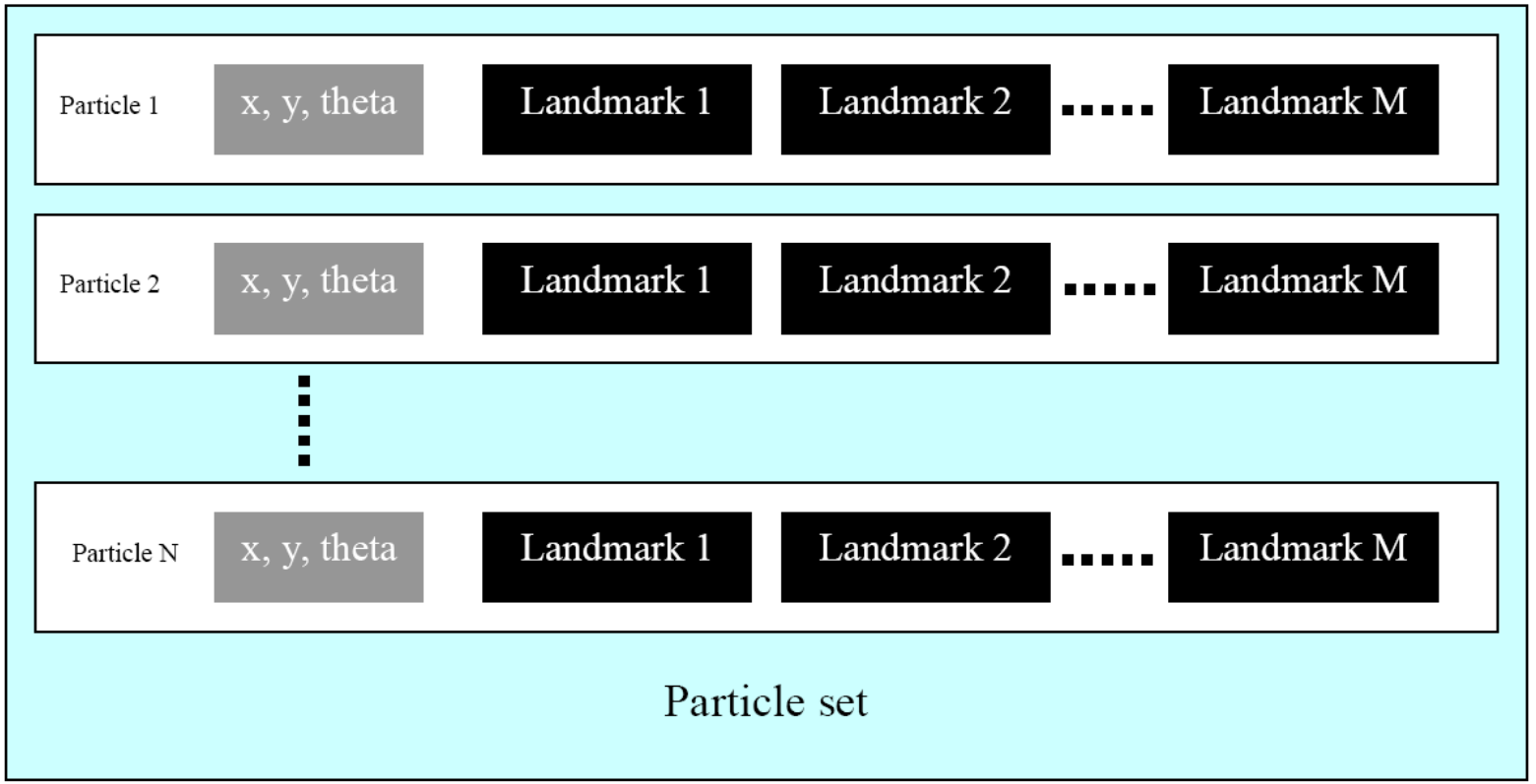
1. Compare particle's prediction of measurements with actual measurements
2. Particles whose predictions match the measurements better are given a higher weight

b) Filter resample:

- Resample particles based on weight
- Filter resample
  - Assign each particle a weight depending on how well its estimate of the state agrees with the measurements and randomly draw particles from previous distribution based on weights creating a new distribution.

FastSLAM: Particle Set

The robot posterior is solved using a Rao Blackwellized particle filtering using landmarks. Each landmark estimation is represented by a 2x2 EKF (Extended Kalman Filter). Each particle is "independent" (due the factorization) from each other and maintains the estimate of M landmark positions.



FastSLAM: A LIDAR-Based Example

- Dimensions: 35 cm in height and 86 cm in diagonal length (tip to tip)
- Bare platform weight: 1.3 kg
- Maximum take-off weight: 2.9 kg
- Endurance: 10 – 15 min hovering

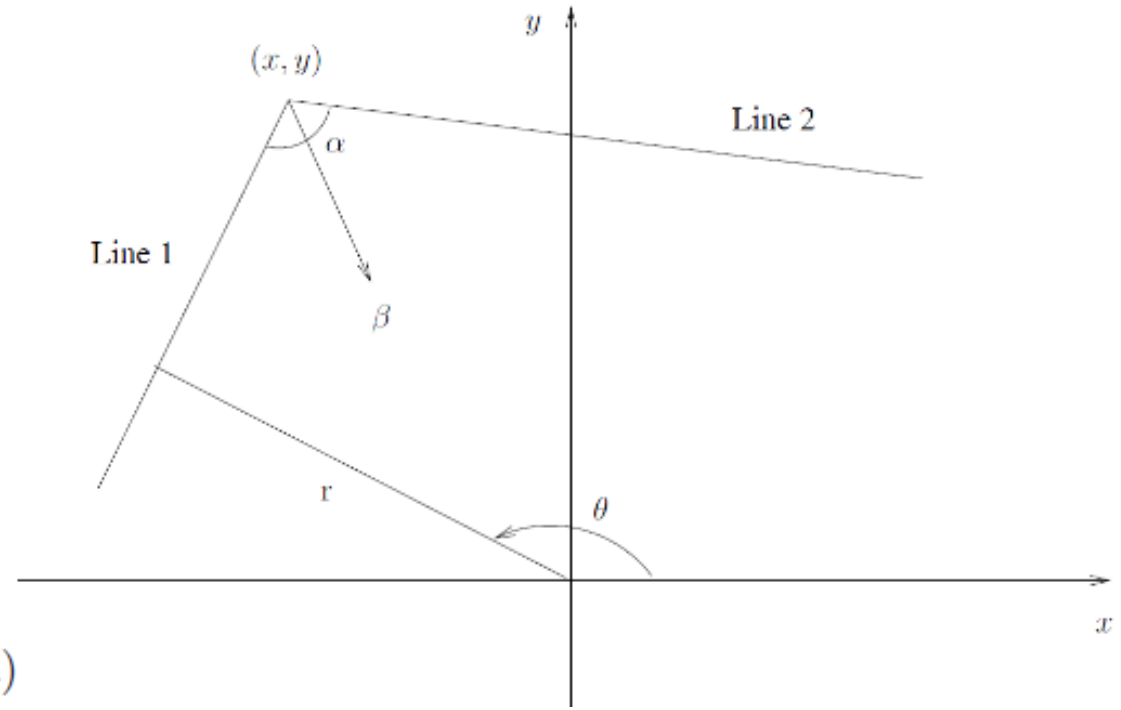
- Power source: 12 VDC
- Light source: Semiconductor laser diode (λ = 905 nm)
- Detection Range: 30 m, 270°
- Accuracy: ±30mm (0.1 to 10 m), ±50mm (10 to 30 m)
- Angular Resolution: 0.25°
- Scan Time: 25 msec/scan
- Interface: USB2.0
- Weight: 370 g

FastSLAM: A LIDAR-Based Example

Feature extraction:

Lines:

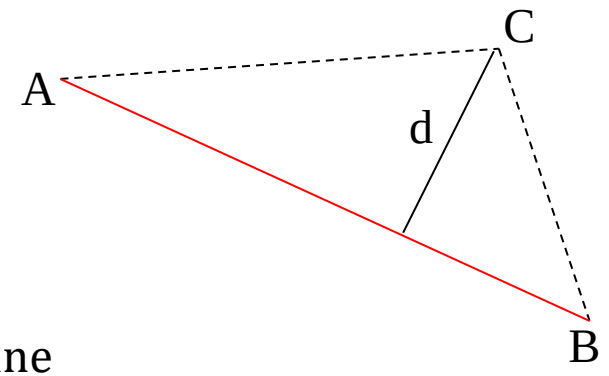
$$
\begin{aligned}
z_l &= \mathcal{N}(r, \theta; \mu_l^z, \Sigma_l^z) \\
\mu_l^z &= \{\mu_r, \mu_\theta\} \\
\Sigma_l^z &= \begin{bmatrix} \sigma_{rr} & \sigma_{r\theta} \\ \sigma_{\theta r} & \sigma_{\theta\theta} \end{bmatrix}
\end{aligned}
$$



Corners:

$$
\begin{aligned}
z_c &= \mathcal{N}(x, y, \alpha, \beta; \mu_c^z, \Sigma_c^z) \\
\mu_c^z &= \{\mu_x, \mu_y, \mu_\alpha, \mu_\beta\} \\
\Sigma_c^z &= \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\alpha} & \sigma_{x\beta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\alpha} & \sigma_{y\beta} \\ \sigma_{\alpha x} & \sigma_{\alpha y} & \sigma_{\alpha\alpha} & \sigma_{\alpha\beta} \\ \sigma_{\beta x} & \sigma_{\beta y} & \sigma_{\beta\alpha} & \sigma_{\beta\beta} \end{bmatrix}
\end{aligned}
$$

FastSLAM: A LIDAR-Based Example



Feature extraction:

Clustering - *Split-and-Merge*:

1. Start with all input points
2. Connect the first point and the last point with a line
3. Calculate the perpendicular distances of all other intermediate points with respect to the line segment obtained in the previous step
4. Search for the point that has the largest distance and compare this distance with a defined threshold
5. If the maximum distance is less than the threshold, all points between the first point and the last point belong to the same line; Else, recursively call Step 1 with (first point, max point) and (max point, last point)

FastSLAM: A LIDAR-Based Example

Feature extraction:

- Line fitting and covariance estimation

  (S. T. Pfister, S. I. Roumeliotis and J. W. Burdick, "Weighted Line Fitting Algorithms for Mobile Robot Map Building and Efficient Data Representation," International Conference on Robotics and Automation, pp. 1304-1311, 2004)

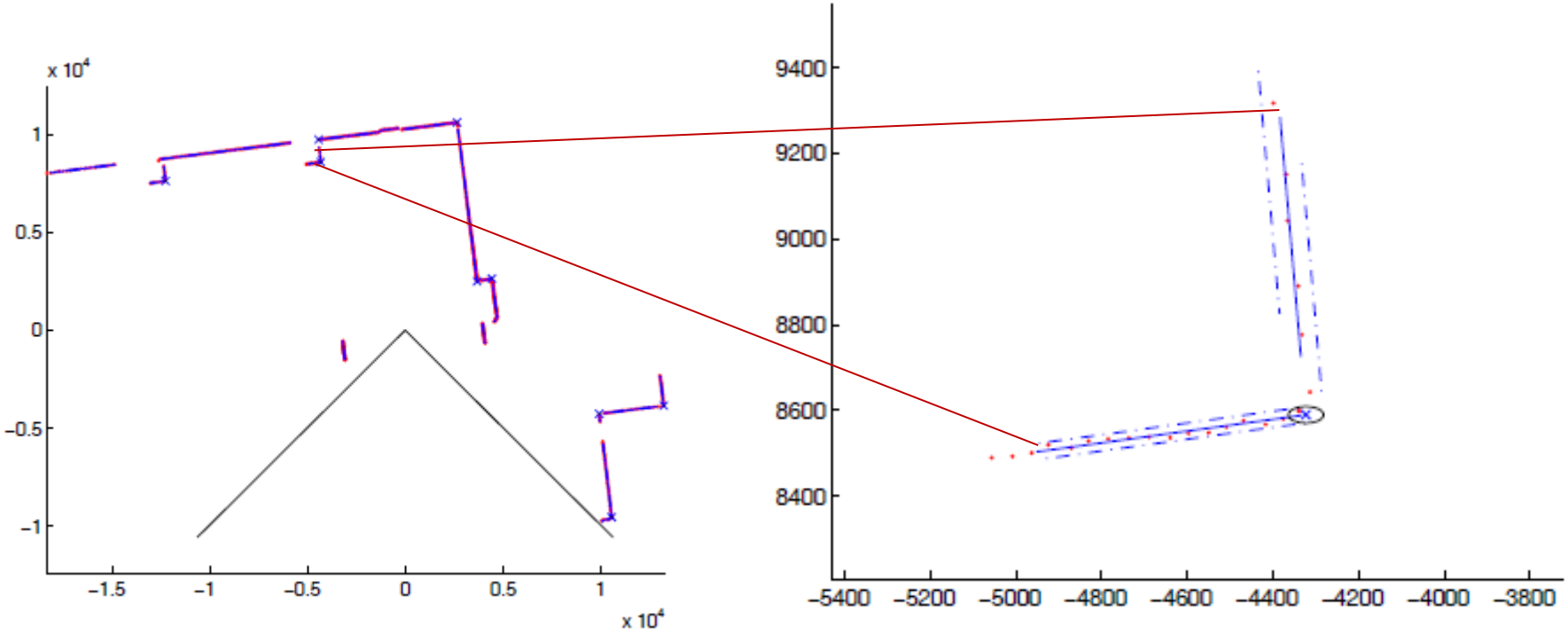
  1. Assume an initial heading of the line by connecting the first and the last point
  2. Find the radial position and its variance based on the weight obtained by projecting uncertainties of the points onto the perpendicular direction of the line
  3. Calculate the iterative increment for heading
  4. Repeat Step 1 – 3 until heading converges
  5. Calculate the remaining terms in the covariance matrix for line parameters

- Corner extraction and covariance estimation

  (P. Nunez, R. Vazquez-Martin, J. C. del Toro, A. Bandera and F. Sandoval, "Natural Landmark Extraction for Mobile Robot Navigation Based On an Adaptive Curvature Estimation," Journal of Robotics and Autonomous Systems, Vol. 56, pp. 247-264, 2008)

FastSLAM: A LIDAR-Based Example

Feature extraction:



FastSLAM: A LIDAR-Based Example

Motion Estimation and Proposal Generation

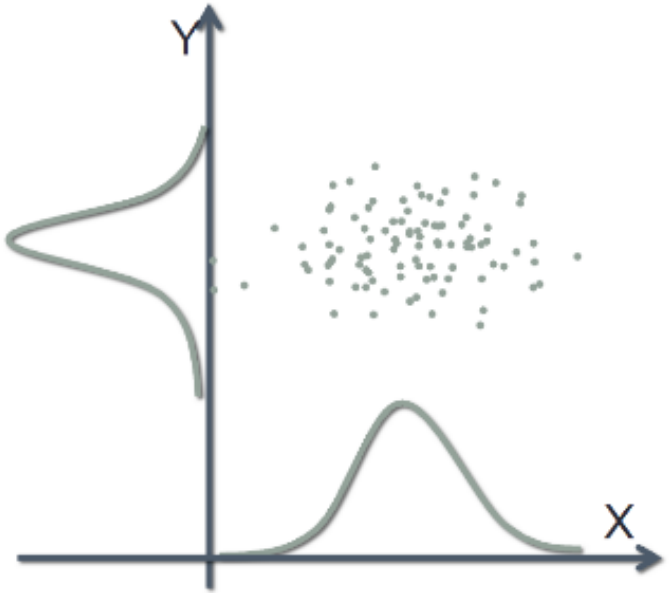
1. Check corner feature correspondences based on pair-wise Mahalanobis distances
2. Organize corner features in such a way that p_i in the previous frame corresponds to q_i in the current frame
3. Calculate the centroid of the feature points for both frames, denoted by $\bar{p}$ and $\bar{q}$
4. Form matrix $P = [p_1 - \bar{p}, p_2 - \bar{p}, p_3 - \bar{p}, \ldots, p_{\max} - \bar{p}]$
5. Form matrix $\mathrm{Q} = [q_1 - \bar{q}, q_2 - \bar{q}, q_3 - \bar{q}, \ldots, q_{\max} - \bar{q}]$
6. Let $[U, S, V] = SVD(PQ^T)$ and $d = sign(\det(PQ^T))$
7. Then $R = V[1\ 0; 0\ d]U^T$ and $T = R\bar{p} - \bar{q}$, that leads to $\bar{c_t} = -\mathrm{atan2}(\mathrm{R}(1), \mathrm{R}(2))$ and $\Delta\bar{x_t} = T(1), \Delta\bar{y_t} = T(2)$

FastSLAM: A LIDAR-Based Example

Motion Estimation and **Proposal Generation**

$$
\begin{aligned}
\Delta c_t^{[m]} &= \Delta \bar{c}_t (1 + \alpha_1 \text{randN}(1)) + \alpha_2 \text{randN}(1) \\
\Delta x_t^{[m]} &= \Delta \bar{x}_t (1 + \alpha_3 \text{randN}(1)) + \alpha_4 \text{randN}(1) \\
\Delta y_t^{[m]} &= \Delta \bar{y}_t (1 + \alpha_3 \text{randN}(1)) + \alpha_4 \text{randN}(1)
\end{aligned}
$$

$$
\begin{aligned}
P_t^{[m]}.c &= P_t^{[m]}.c + \Delta c_t^{[m]} \\
P_t^{[m]}.x &= P_t^{[m]}.x + \cos(P_t^{[m]}.c)\Delta x_t^{[m]} - \sin(P_t^{[m]}.c)\Delta y_t^{[m]} \\
P_t^{[m]}.y &= P_t^{[m]}.y + \sin(P_t^{[m]}.c)\Delta x_t^{[m]} + \cos(P_t^{[m]}.c)\Delta y_t^{[m]}
\end{aligned}
$$



FastSLAM: A LIDAR-Based Example

Per-particle Data Association:

- Every particle has its own data association hypothesis
- Extra robustness but worse complexity
- Methods of data association:

1. Consider each feature independently

   - Individual compatibility nearest neighbor (ICNN)

2. Consistent association – no duplicated associations

   - Sequential compatibility nearest neighbor (SCNN)

   - Joint compatibility branch and bound (JCBB)

3. Consider data association in the previous iteration, maximize the whole probability history

FastSLAM: A LIDAR-Based Example

Per-particle Measurement Update:

No. of corners

M particles

$(N_c + N_l) \times M$ EKFs:

No. of lines

$$
\begin{aligned}
Z_{n,t} &= \Sigma_{n,t-1}^{[m]} + \Sigma_{n,t}^{z} \\
K_{n,t}^{[m]} &= \Sigma_{n,t-1}^{[m]} Z_{n,t}^{-1} \\
\mu_{n,t}^{[m]} &= \mu_{n,t-1}^{[m]} + K_{n,t}^{[m]} \left(z_{n,t} - \hat{z}_{n,t} \right) \\
\Sigma_{n,t}^{[m]} &= \left(I - K_{n,t}^{[m]} \right) \Sigma_{n,t-1}
\end{aligned}
$$

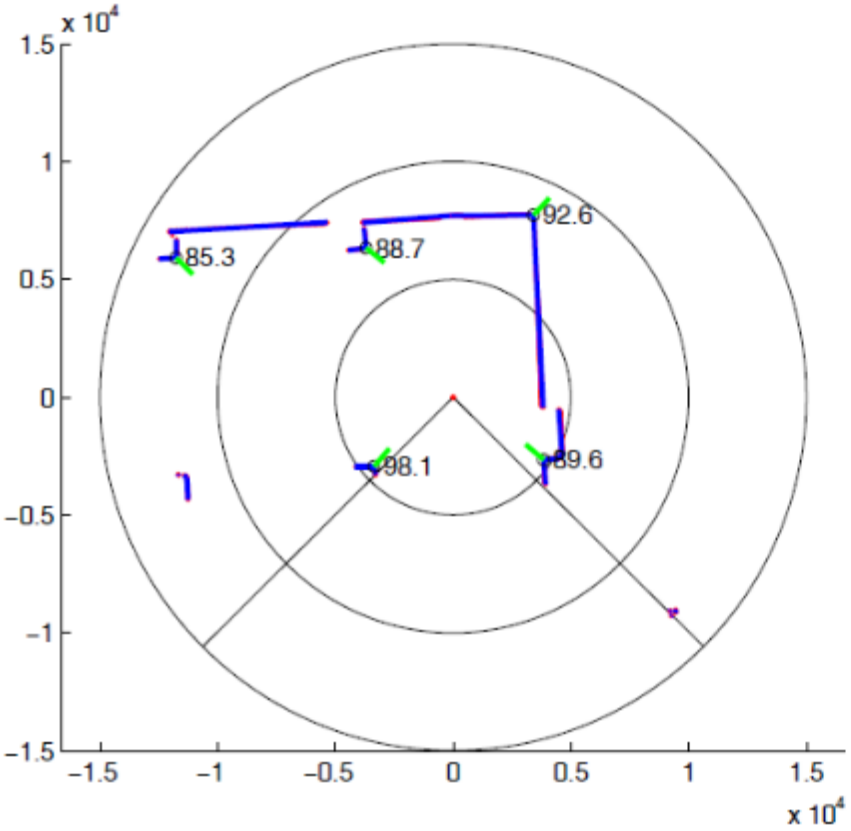
FastSLAM: A LIDAR-Based Example

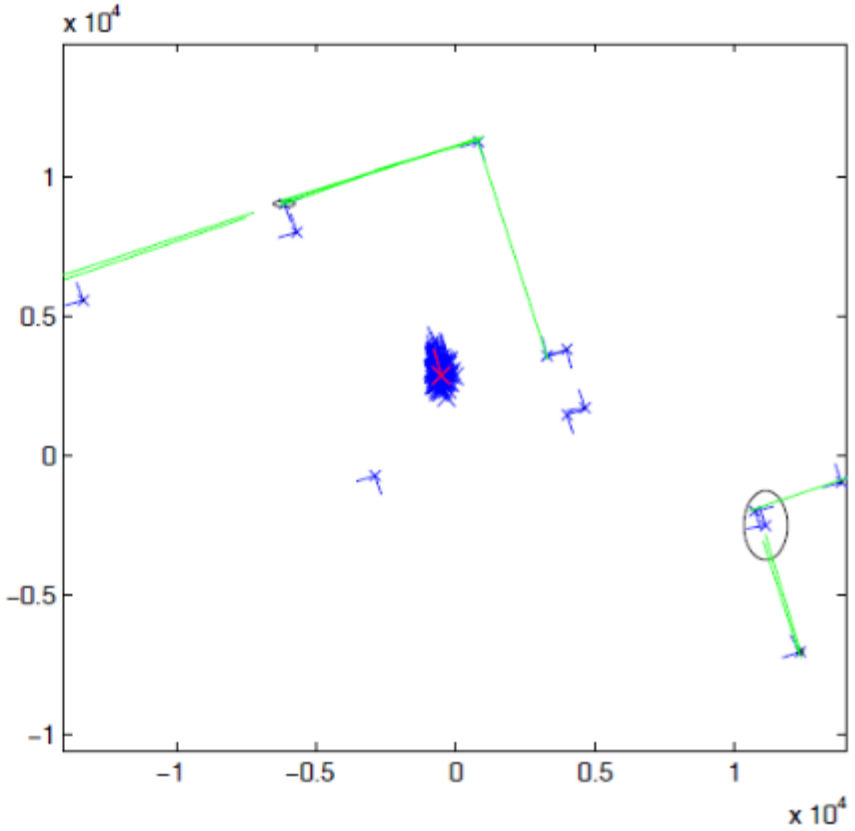
Particle Importance Weighting and Resampling:

$$w_t^{[m]} = \frac{1}{\sqrt{2\pi Z_{n,t}}} \exp\left\{-\frac{1}{2}(z_{n,t} - \hat{z}_{n,t})^{\mathrm{T}} [Z_{n,t}]^{-1} (z_{n,t} - \hat{z}_{n,t})\right\}$$

1. Calculate W, the total weight of all M particles
2. Generate a random number W' between 0 and W
3. Deduct W by $w_t^{[1]}, w_t^{[2]}, w_t^{[3]}, \ldots, w_t^{[i]}$ one by one until the result hits W'
4. Particle i in the old set will be chosen as one element in the new set
5. Repeat Step 2 to Step 4 M times to generate a whole new set of M particles

FastSLAM: A LIDAR-Based Example

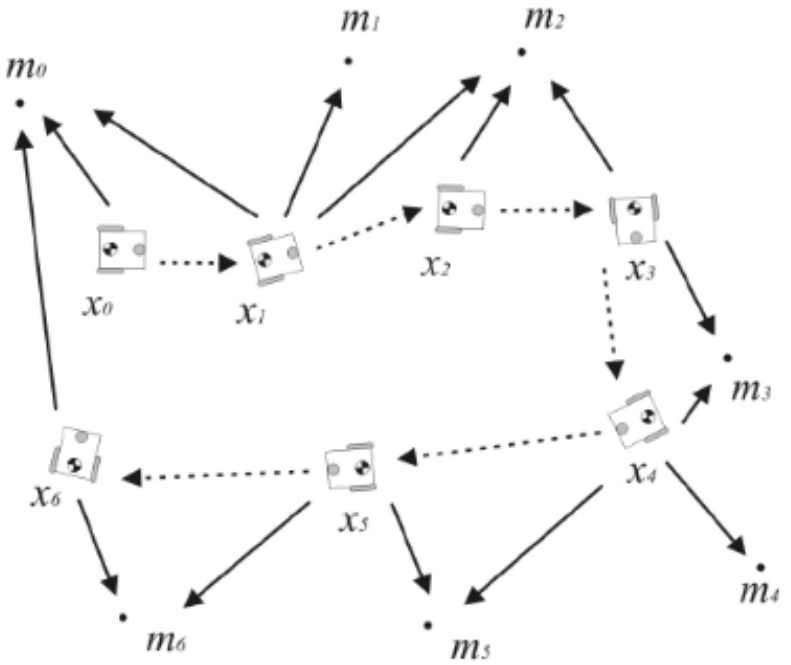




FastSLAM: A LIDAR-Based Example

GraphSLAM

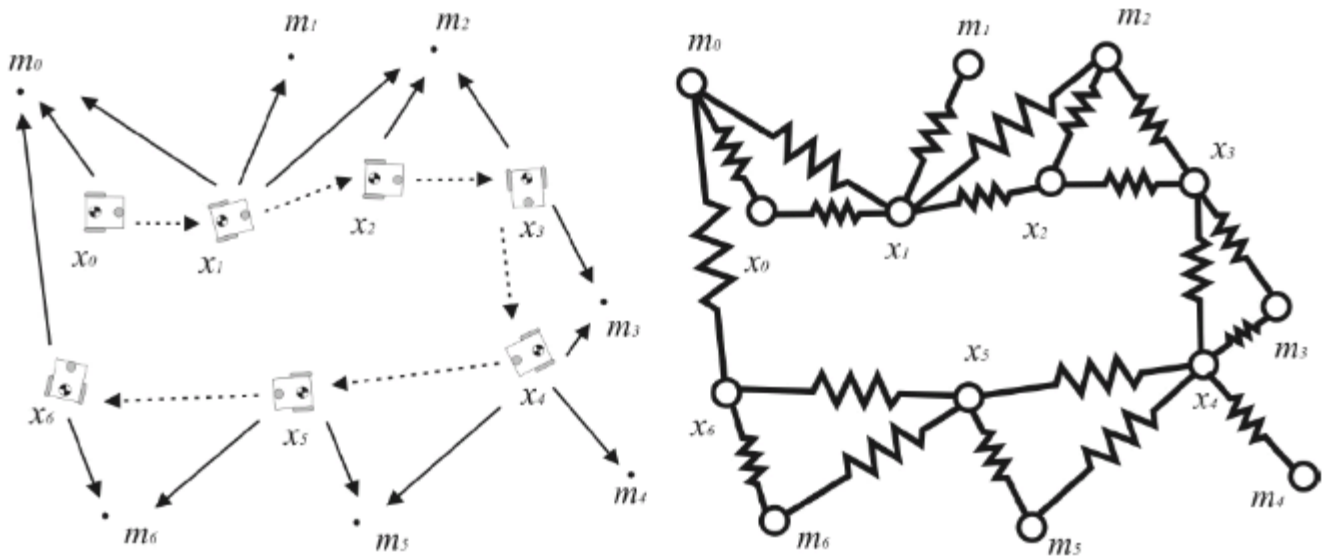
- **GraphSLAM** was born from the intuition that the SLAM problem can be interpreted as a sparse graph of **nodes and constraints** among the nodes.
- The nodes of the graph are the robot locations and the features in the map.
- The constraints are the relative position between consecutive robot poses, (given by the odometry input ***u***) and the relative position between the robot locations and the features observed from those locations.



GraphSLAM

- The critical idea of GraphSLAM is that the constraints are not rigid constraints but soft/flexible constraints.

- It is by relaxing these constraints that we can compute the solution to the full SLAM problem, that is, the best estimate of the robot path and the environment map. By saying this in other words, GraphSLAM represents robot locations and features as the nodes of an elastic net. The SLAM solution can then be found by computing the state of minimal energy of this net.



Thank You!