110-1 (Fall 2021)
自動控制實驗
Laboratory for Automatic Control
Lecture 8
Final Project: Unmanned Aerial Vehicles (UAVs) Manipulations

教師：　劉浩澧　hlliu@ntu.edu.tw
助教：　洪聲仰　shengyang@ntu.edu.tw
實驗室：　學新館R422

本單元之目標

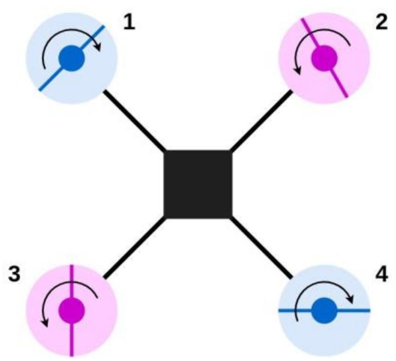
- 理論及工具學習:
  - UAV介紹
  - UAV物理模型
  - 介紹測試載具—Quadcopter UAV
- 期末專題: 四旋翼UAV控制
- 本單元共進行6-7周

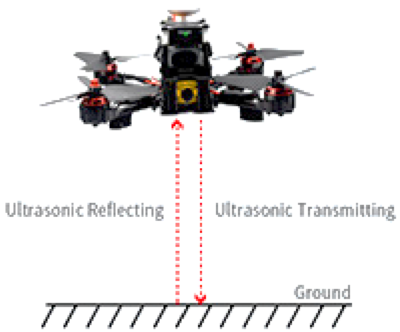
Introduction to Unmanned Aerial Vehicles (UAVs)

Drones or Unmanned Aerial Vehicles (UAVs)

- Drones, or Unmanned Aerial Vehicles (UAVs) are aircraft that are pilotless
- Some are winged aircraft, and some are helicopters that rely on one or more rotor blades for lift
- Application
  - Aerial photography and video, surveillance, security/police work, search and rescue, farming, defense, entertainment, package delivery to flying cars, …
- Helicopters with multiple rotors are a popular platform for these vehicles due to the simplicity of the vehicle hardware and maintenance, ability to hover, and the vertical takeoff and landing capability.

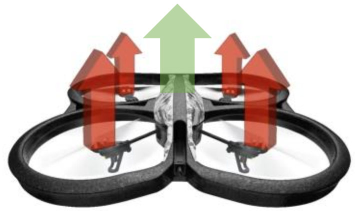
- The quadrotor consists of four rotors, with one pair rotating **clockwise** (CW), and the other pair rotating **counter-clockwise** (CCW)
- By independently controlling each **rotor's speed**, it is possible to command the attitude of the vehicle along with the **translation** and **altitude**
- Sensor
    - **GPS**: 3-D position and velocity sensing
    - **Inertial sensors**: that provide pitch, roll, yaw angles, and their angular rates.
    - **Inertial Measurement Units (IMUs):** contain 3-D accelerometers and 3-D gyros at a minimum, but sometimes contain 3-D magnetometers
    - **Ultrasound**: height detection





Moving the Drone

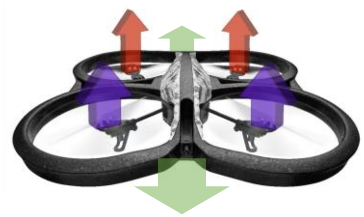
- **Take off**
  - Need a net upward force → Motors generate thrust that is greater than the weight, making the quad rise upwards



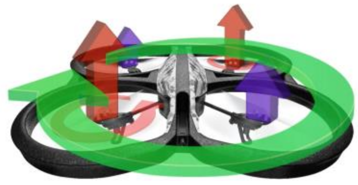
- **Hover**
  - Motors generate Thrust, the thrust should equal the weight of the system; the two forces cancel and our drone Hovers

- **Roll**
  - to roll towards the left, the thrust is increased on the motors on the right, also decrease the thrust on the motors on the Left.

- **Pitch (Towards us**)
  - the thrust to the rear motors is increased, this creates a net forward force which causes the Drone to pitch downward.; also decrease the power to the two front motors to keep the angular momentum conserved.



- **Yaw (Clockwise)**
  - Increase the thrust on the anti-clockwise moving motors, decrease the thrust on clockwise Rotating Motors. the CCW thrust results anti-clockwise Torque, and the quad rotates clockwise to conserve the angular momentum.



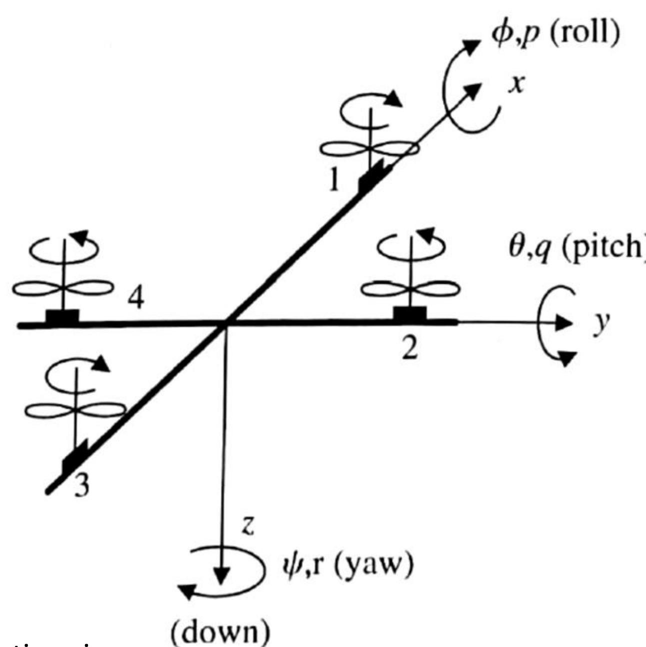
https://hackernoon.com/quadcopter-physics-explained-468ee44ba40b

Modeling a Drone

- **System linearization** can be done assume the body-fixed coordinate system stays essentially level through the motion and the pitch and roll angles remain small; we also assume the angular motions are reasonably small
- The longitudinal motion $(x, u, \theta, \text{and } q)$ is uncoupled from the lateral motion $(y, v, \phi, \text{and } p)$
- The longitudinal x-axis equations are:

$$\begin{bmatrix} \dot{x} \\ \dot{u} \\ \dot{q} \\ \dot{\theta} \\ \dot{T}_\theta \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & X_u & 0 & -g_0 & 0 \\ 0 & M_u & 0 & 0 & M_\theta \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -a \end{bmatrix} \begin{bmatrix} x \\ u \\ q \\ \theta \\ T_\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a \end{bmatrix} T_{lon}$$

$$\begin{bmatrix} \theta_m \\ x_m \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ q \\ \theta \\ T_\theta \end{bmatrix}$$

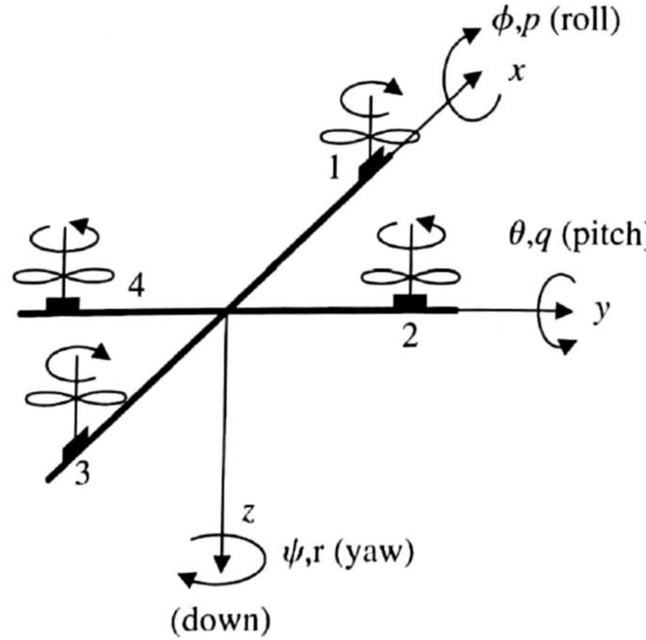


$x = Xm$ = measured position in the drone frame x direction
u = velocity in the drone frame x direction
q = angular rate about the positive drone frame y-axis, or pitch rate
$\theta = \theta m$ = measured pitch angle from horizontal
X_u = partial derivative of the aerodynamic force in x, direction with respect to perturbations in u
M_u = partial derivative of the aerodynamic (Pitching) moment with respect to perturbations in u
$M_\theta = 1 / I_y$
T_θ = pitching moment around +y axis from rotors 1 and 3
T_{lon} = pitching torque command for rotors 1 and 3
g_0= gravity; a = delay in the rotors producing the changed thrust and resulting torque

- The lateral y-axis equations are:

$$\begin{bmatrix} \dot{y} \\ \dot{v} \\ \dot{p} \\ \dot{\phi} \\ \dot{T_\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & Y_v & 0 & g_0 & 0 \\ 0 & L_v & 0 & 0 & L_\phi \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -a \end{bmatrix} \begin{bmatrix} y \\ v \\ p \\ \phi \\ T \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a \end{bmatrix} T_{Lat}$$

$$\begin{bmatrix} \phi_m \\ y_m \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} y \\ v \\ p \\ \phi \\ T \end{bmatrix}$$



$y = ym$ = measured position iri the drone frame y direction
v= velocity in the drone frame y direction
p = angular rate about the positive drone frame x-axis, or roll rate
$\Phi = \Phi_m$ = measured roll-angle from horizontal
Y_v = partial derivative of the aerodynamic force in y direction with respect to perturbations in v,
Lv = partial derivative of the aerodynamic (rolling) moment with respect to perturbations in v
$L_\Phi = 1/I_x$
T_Φ = rolling moment around +x-axis from rotors 2 and 4,
T_{lat} = rolling torque command for rotors 2 and 4
a= delay in the rotors producing the changed thrust and resulting torque; g_o = gravity

- The rotational - z-axis equations are:

$$\begin{bmatrix} \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} r \\ \psi \end{bmatrix} + \begin{bmatrix} 1/I_z \\ 0 \end{bmatrix} T_\psi$$

$$[\psi_m] = [0 \quad 1] \begin{bmatrix} r \\ \psi \end{bmatrix}$$

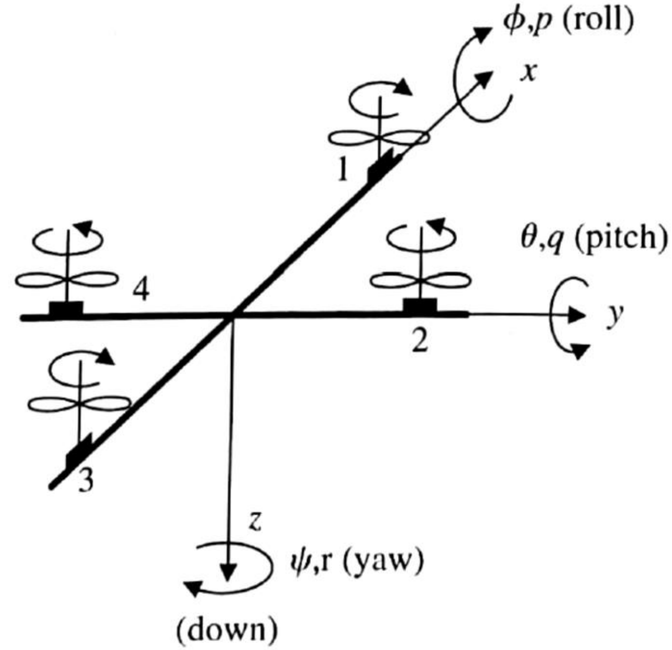
r = angular rate about positive drone frame z-axis, or yaw rate
$\psi = \psi_m$ = measured azimuth angle of the drone frame x-axis with respect to North,
T_ψ= commanded yawing moment

- The altitude dynamics are

$$\begin{bmatrix} \dot{h} \\ \dot{w} \\ \dot{F_h} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & Z_h \\ 0 & 0 & -a \end{bmatrix} \begin{bmatrix} h \\ w \\ F_h \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ a \end{bmatrix} F_{alt}$$

$$h_m = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} h \\ w \\ F_h \end{bmatrix}$$

$h = hm$ = vertical position
w = vertical, z-axis, velocity
$Z_h = 1/mo, mo$ = mass ofthe vehic1e
F_h= vertical thrust
F_{alt} = commanded thrust from all rotors



- To determine position in an earth-fixed frame for the case when there is no yaw rate, *r,* the transformation matrix based on the rotation of the level body-fixed frame is required

$$\begin{bmatrix} \dot{x_E} \\ \dot{y_E} \end{bmatrix} = \begin{bmatrix} cos(\psi) & -sin(\psi) \\ +sin(\psi) & cos(\psi) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

X_E = position in the earth-fixed frame with the x-axis pointing north
Y_E = position in the earth-fixed frame with the y-axis pointing east

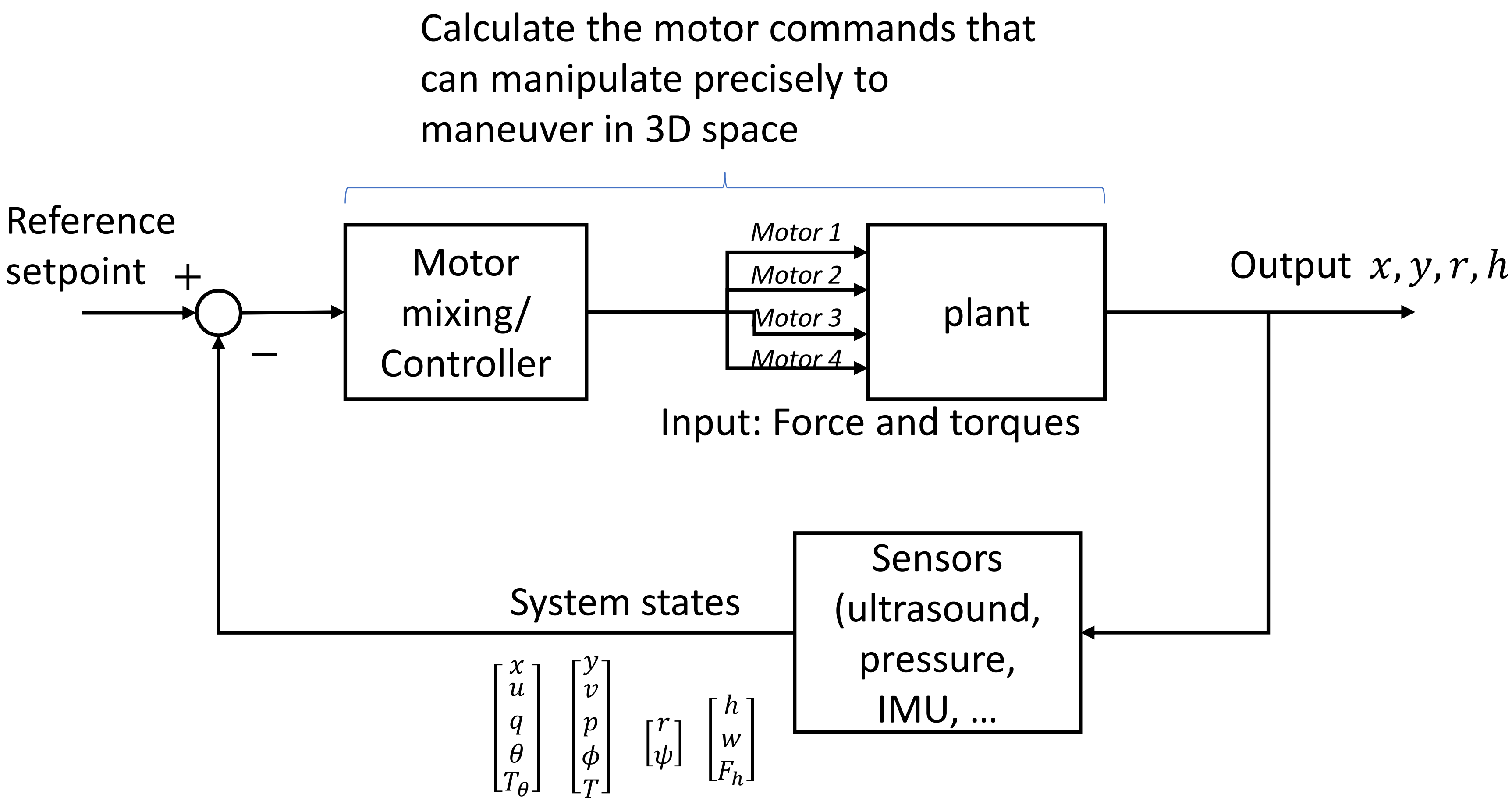
- Motor mixing

Motor$_{\text{front, right}}$ (δT_1) = Thrust (F_{alt}) + Yaw (T_Ψ) + Pitch (T_θ) + Roll (T_ϕ)

Motor$_{\text{front, left}}$ (δT_2) = Thrust (F_{alt}) - Yaw (T_Ψ) + Ritch (T_θ) - Roll (T_ϕ)

Motor$_{\text{back, right}}$ (δT_3) = Thrust (F_{alt}) - Yaw (T_Ψ) - Ritch (T_θ) + Roll (T_ϕ)

Motor$_{\text{back, left}}$ (δT_4) = Thrust (F_{alt}) + Yaw (T_Ψ) - Ritch (T_θ) - Roll (T_ϕ)



Try a PID controller for a single axis

- The parameters used were

M_u=1.1
$X_u = -0.25$
M_θ=0.02
$g_0 = 32.2$
$a = 20$

Produce the TF

$$\frac{\theta_m(s)}{T_{lon}(s)} = 0.4\frac{(s+0.25)}{(s-1.6\pm 2.8j)(s+3.4)(s+20)}$$

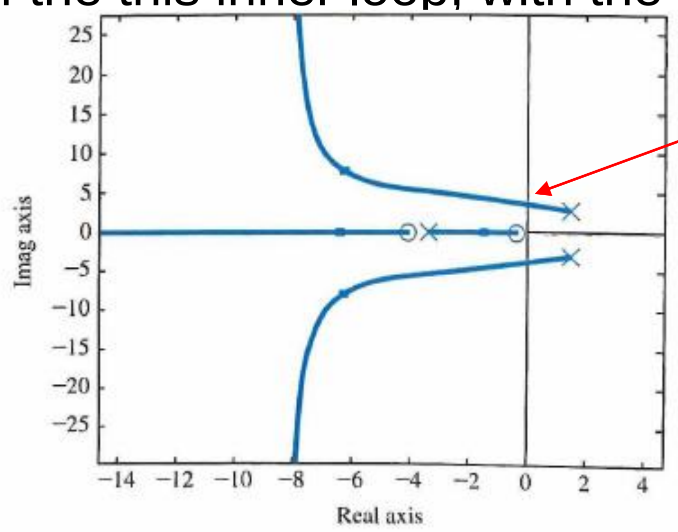
$$\frac{x_m(s)}{T_{lon}(s)} = -13\frac{1}{s(s-1.6\pm 2.8j)(s+3.4)(s+20)}$$

Unstable pole!

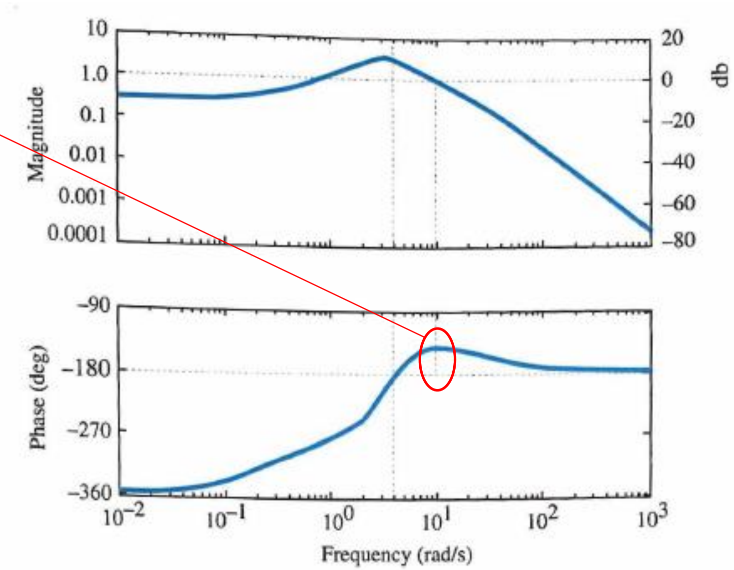
- The use of sisotool allows to find the PD controller (under feedback of θ and $\dot{\theta}$)

$D_{c1lon}(s) = 500(s+4)$ Inner loop controller

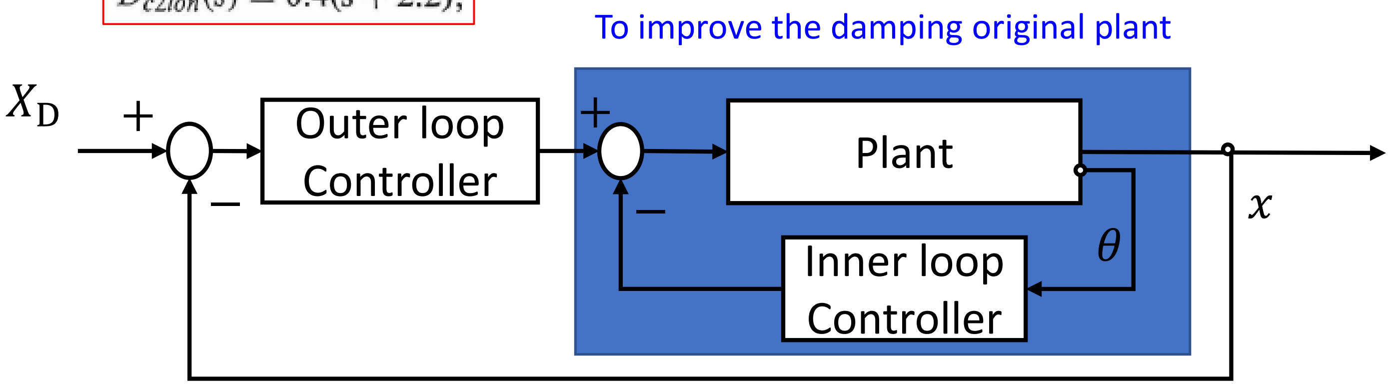
This produces a damping $\xi \sim 0.6$, and ωn =10 rad/sec for the oscillatory roots of the this inner loop, with the PM of 40°



Lowering the gain by a factor of 3.2 would cause an instability while there is no upper limit

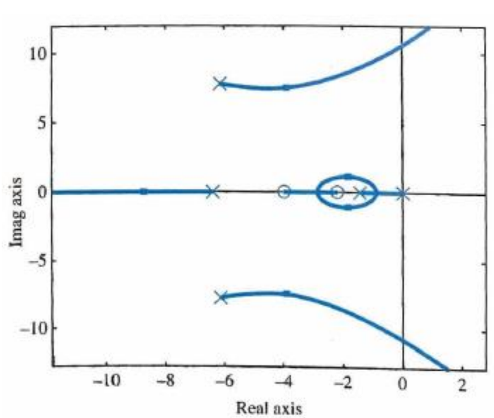


- For the outer loop, we wish to command a change in position; therefore our measurement will be x.
- with the addition of the inner-loop feedback, we find that an outer-loop PD controller $D_{c2lon}(s) = 0.4(s + 2.2)$,

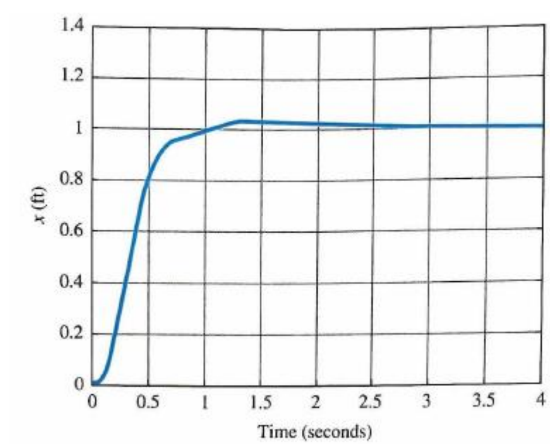


- The rise time is approximately 0.6 seconds and the sett1ing time is approximately 3 seconds, with an overshoot of less than 5%.

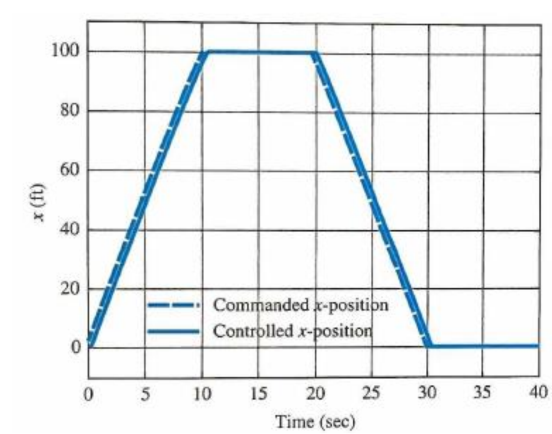
Root locus



Step response

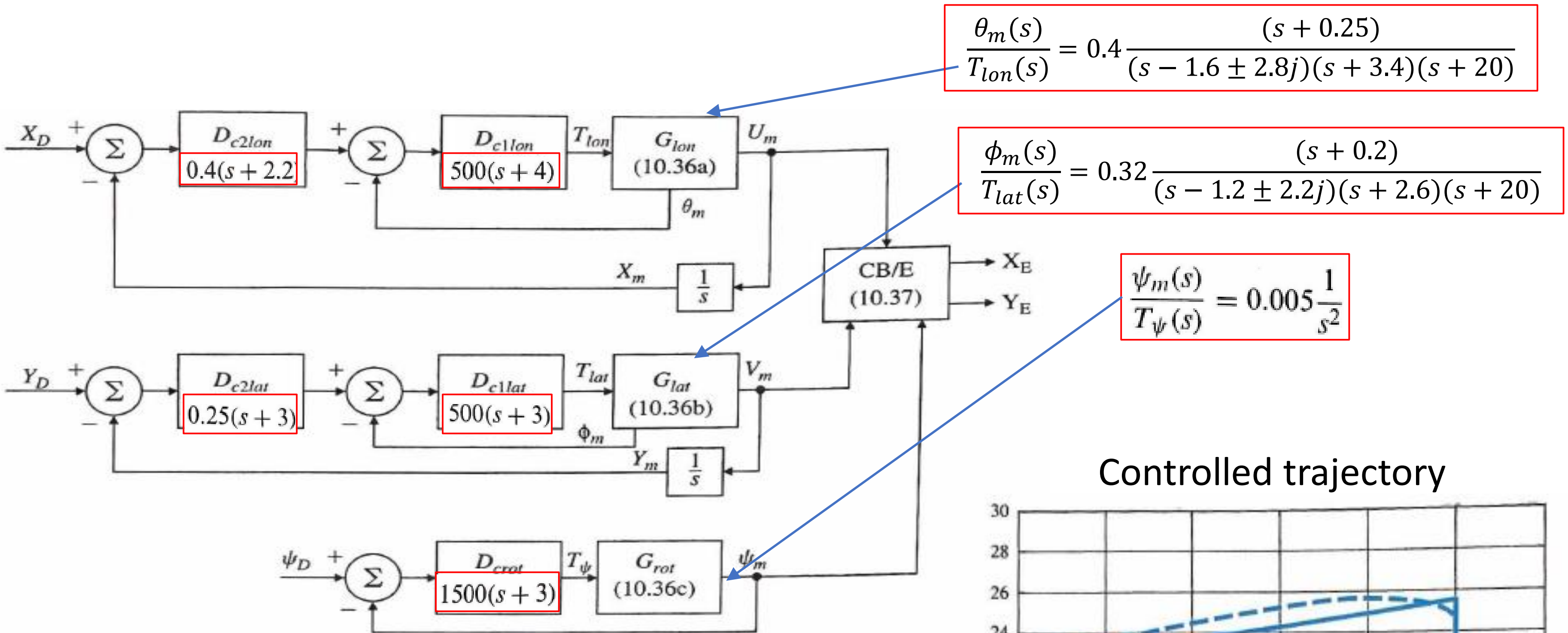


Trajectory follow



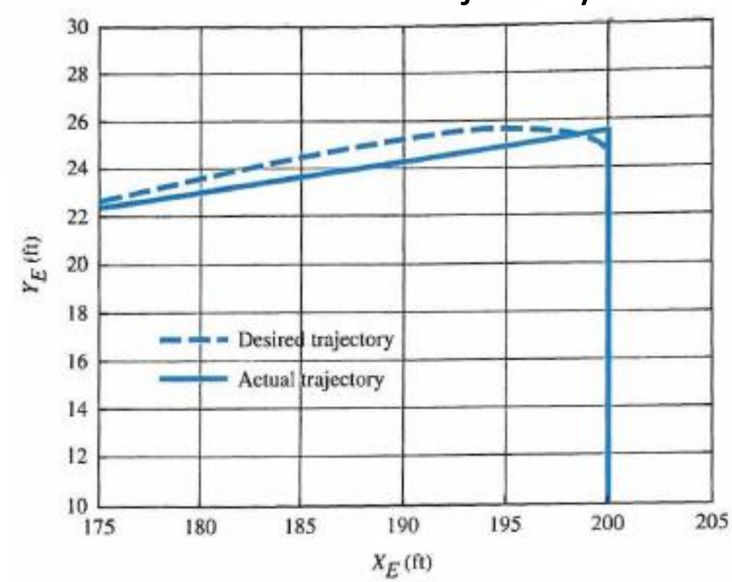
Try PID controllers for 2-D motion in the horizontal plane

- X_D and Y_D design can be decoupled, ad design scenarios are similar



- It is generally acceptable to use the uncoupled equations for purposes of control system design!

Controlled trajectory



Try an optimal design

- Find the transfer function

$$\begin{bmatrix} \dot{x} \\ \dot{u} \\ \dot{q} \\ \dot{\theta} \\ \dot{T}_\theta \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & X_u & 0 & -g_0 & 0 \\ 0 & M_u & 0 & 0 & M_\theta \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -a \end{bmatrix} \begin{bmatrix} x \\ u \\ q \\ \theta \\ T_\theta \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ a \end{bmatrix} T_{lon}$$

$$\begin{bmatrix} \theta_m \\ x_m \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ u \\ q \\ \theta \\ T_\theta \end{bmatrix}$$

$$\frac{x\,(s)}{T_{lon}(s)} = \frac{-12.88}{s(s - 1.6 \pm 2.8j)(s + 3.37)(s + 20)}$$

- Selecting the following LQR weighting matrices,

$$\mathbf{Q} = \rho \mathbf{C}^T \mathbf{C}, \mathbf{R} = 1, \mathbf{C} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \mathbf{K} = \begin{bmatrix} -49020 & 6204 & 126690 & -316230 & 2.66 \end{bmatrix}$$

$$\mathbf{N_x} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, N_u = 0, \bar{N} = N_u + \mathbf{K}\mathbf{N_x} = -3.1623e5, \mathbf{M} = \mathbf{B}\bar{N} \qquad q = 1e7, \quad \mathbf{B}_1 = \mathbf{B}, \qquad \mathbf{L} = \begin{bmatrix} 148 \\ -44 \\ -23 \\ 17 \\ -170 \end{bmatrix}$$

Results in the dynamic controller transfer function for the longitudinal axis given by

$$D_c(s) = \frac{1.59e7(s + 9.3 \pm 3.1j)(s + 3.8)(s + 20)}{(s + 36.9)(s + 24.7 \pm 22.2j)(s + 2.2 \pm 24.5j)}$$

- The overall c1osed-loop system equations are

$$\begin{bmatrix}\dot{x}\\ \dot{\hat{x}}\end{bmatrix}=\begin{bmatrix}\boldsymbol{A} & -\boldsymbol{BK}\\ \boldsymbol{LC} & \boldsymbol{A}-\boldsymbol{BK}-\boldsymbol{LC}\end{bmatrix}\begin{bmatrix}x\\ \hat{x}\end{bmatrix}+\begin{bmatrix}\boldsymbol{B}\bar{N}\\ M\end{bmatrix}r$$

$$y=[\boldsymbol{C} \quad 0]\begin{bmatrix}x\\ \hat{x}\end{bmatrix}$$

- The lateral y-axis state-space equations result in the transfer function:

$$\begin{bmatrix}\dot{y}\\ \dot{v}\\ \dot{p}\\ \dot{\phi}\\ \dot{T_\phi}\end{bmatrix}=\begin{bmatrix}0 & 1 & 0 & 0 & 0\\ 0 & Y_v & 0 & g_0 & 0\\ 0 & L_v & 0 & 0 & L_\phi\\ 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & -a\end{bmatrix}\begin{bmatrix}y\\ v\\ p\\ \phi\\ T\end{bmatrix}+\begin{bmatrix}0\\ 0\\ 0\\ 0\\ a\end{bmatrix}T_{Lat}$$

$$\begin{bmatrix}\phi_m\\ y_m\end{bmatrix}=\begin{bmatrix}0 & 0 & 0 & 1 & 0\\ 1 & 0 & 0 & 0 & 0\end{bmatrix}\begin{bmatrix}y\\ v\\ p\\ \phi\\ T\end{bmatrix}$$

$$\frac{y\,(s)}{T_{lat}(s)}=\frac{10.3}{(s-1.2\pm 2.2j)(s+2.6)(s+20)}$$

- Choosing the following LQR weighting matrices

$$\mathbf{Q}=\rho\mathbf{C}^T\mathbf{C},\mathbf{R}=1,\mathbf{C}=[\,0 \quad 0 \quad 0 \quad 1 \quad 0\,], \qquad \rho=1e\text{^}10 \qquad q=1e10, \quad \mathbf{B}_1=\mathbf{B},$$

$$\mathbf{N_x}=\begin{bmatrix}0\\ 0\\ 0\\ 1\\ 0\end{bmatrix}, N_u=0,\ \bar{N}=N_u+\mathbf{K}\mathbf{N_x}=3.16e5, \mathbf{M}=\mathbf{B}\bar{N}. \qquad \mathbf{L}=\begin{bmatrix}733\\ 1539\\ 261\\ 38\\ 52342\end{bmatrix}$$

- This results in the lateral dynamic controller transfer function

$$D_c(s) = \frac{-9.76e7(s+2\pm j5.4)(s+5)(s+20)}{(s+46)(s+30.6\pm j28.9)(s+0.78\pm j31)}.$$

- The rotational z-axis state-space equations are as

$$\begin{bmatrix} \dot{r} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} r \\ \psi \end{bmatrix} + \begin{bmatrix} 1/I_z \\ 0 \end{bmatrix} T_\psi,$$

$$[\psi_m] = \begin{bmatrix} 0 & 1 \end{bmatrix}\begin{bmatrix} r \\ \psi \end{bmatrix}$$

$$\rightarrow \quad \frac{\psi_m(s)}{T_\psi(s)} = 0.005\frac{1}{s^2}$$

- Selecting the ensuing LQR weighting matrices

$$\mathbf{Q} = \rho\mathbf{C}^T\mathbf{C}, \mathbf{R} = 1, \mathbf{C} = [0 \quad 1], \quad \rho = 1e10, \quad q = 1e10, \mathbf{B}_1 = \mathbf{B},$$

$$\mathbf{K} = \begin{bmatrix} 11246 & 316227 \end{bmatrix}$$

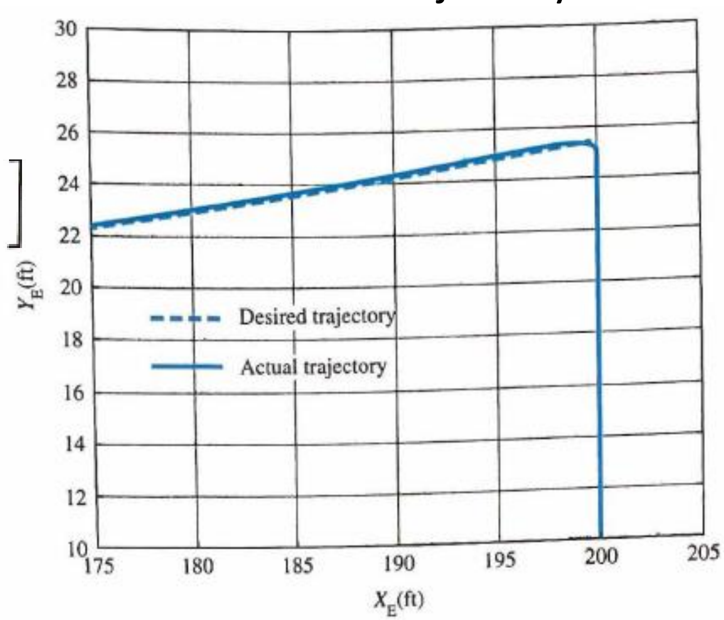
$$\mathbf{N_x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, N_u = 0, \bar{N} = N_u + \mathbf{KN_x} = 3.16e5, \mathbf{M} = \mathbf{B}\bar{N}.$$

$$\mathbf{L} = \begin{bmatrix} 500 \\ 31 \end{bmatrix}$$

- The dynamic controller transfer function for the rotational dynamics

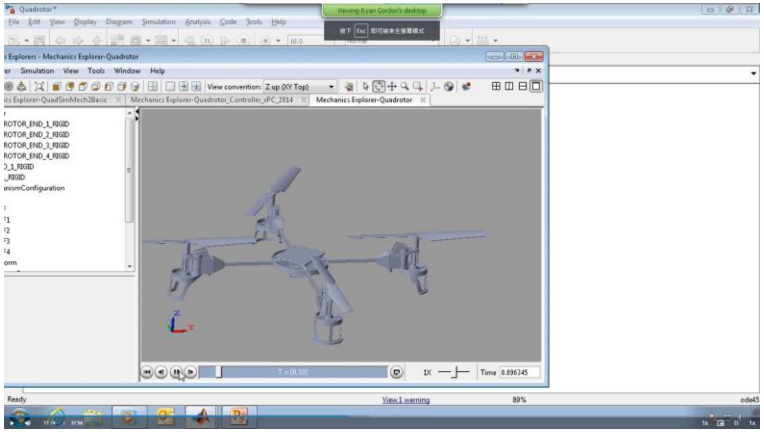
$$D_c(s) = \frac{-1.59e7(s+10)}{(s+43.9\pm 43.9j)}$$

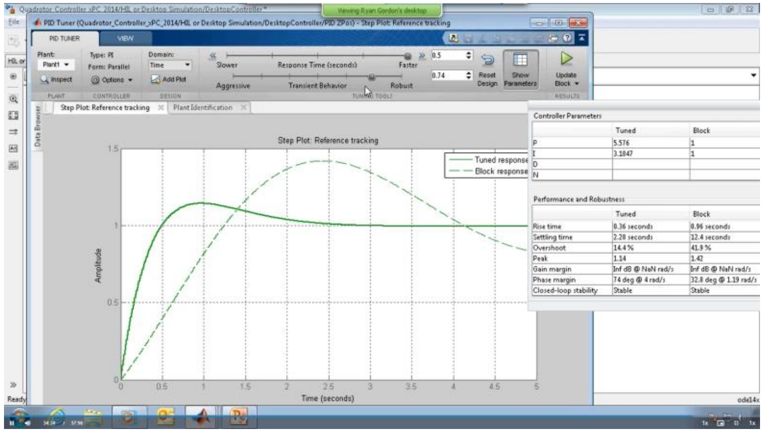
Controlled trajectory

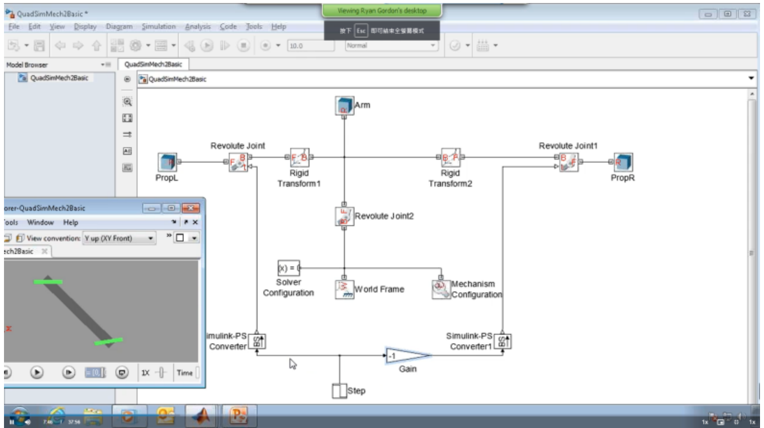


Quadcopter Simulation and Control using MATLAB Simulink

- https://www.mathworks.com/videos/quadcopter-simulation-and-control-made-easy-93365.html

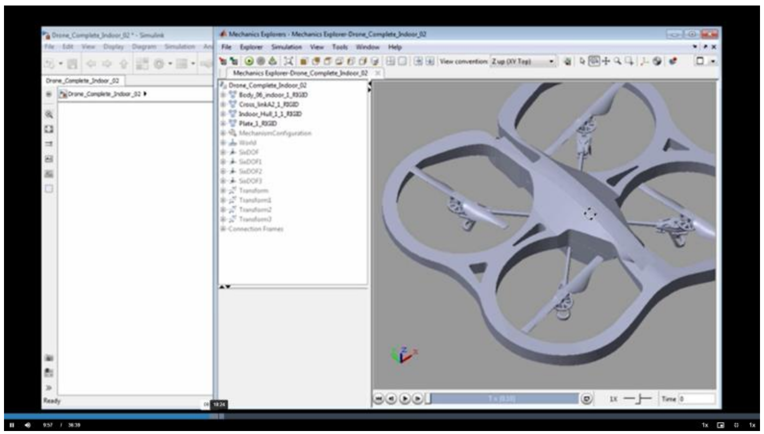


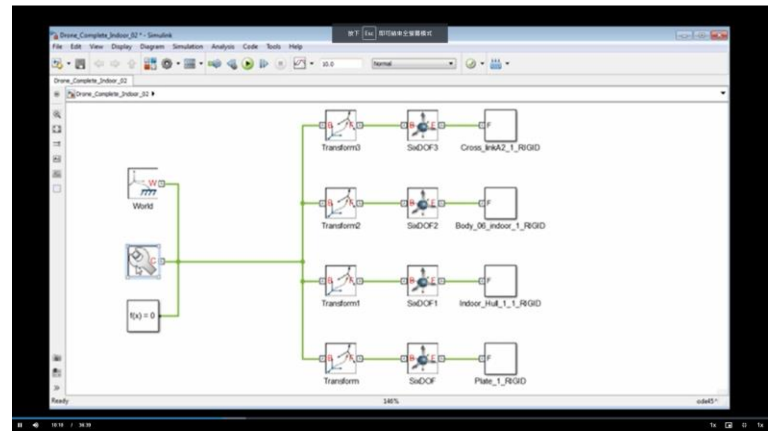


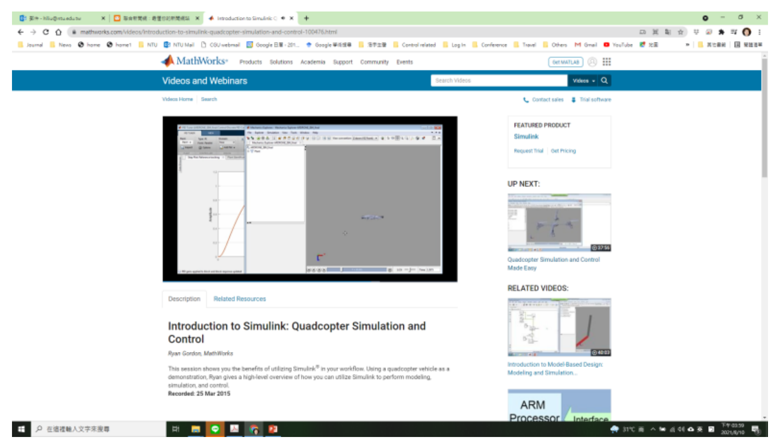


Quadcopter Simulation and Control using MATLAB Simulink

- https://www.mathworks.com/videos/introduction-to-simulink-quadcopter-simulation-and-control-100476.html

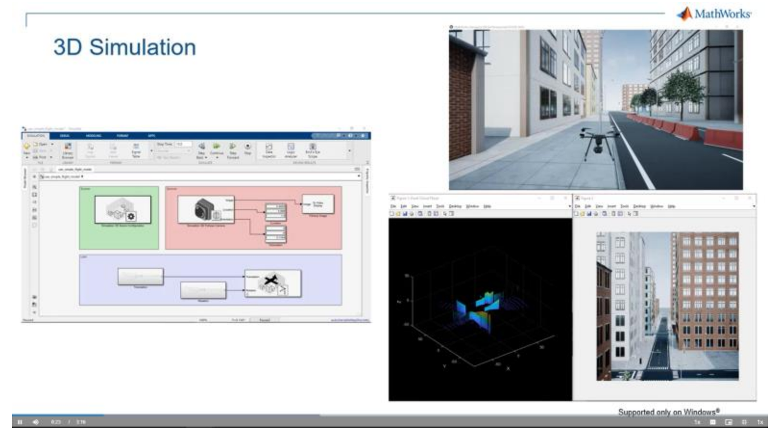


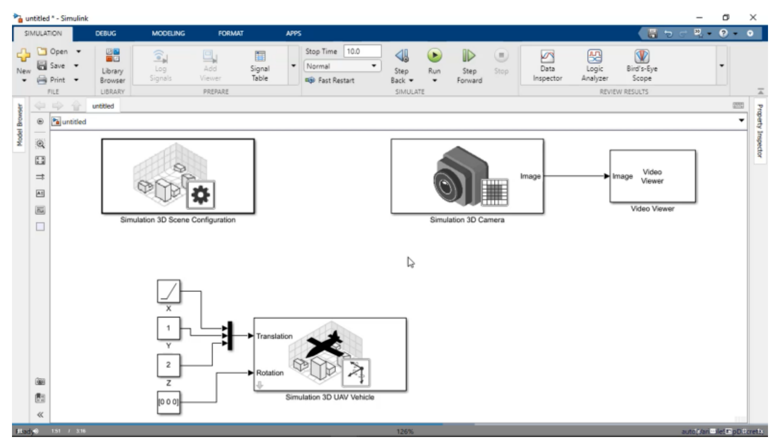




UAV toolbox: Simulate drone algorithms in a virtual environment

- https://www.mathworks.com/discovery/drone-simulation.html





期末專題: UAV工具介紹

Tello Edu Quadcopter Drone (Ryze Inc.)

- **技術規格**
  - 起飛重量 (含槳保護罩)：87 g
  - 最大水準飛行速度：28.8 km/h
  - 最常飛行時間：13 分鐘 (無風環境、15 km/h 勻速飛行時測得）
  - 具備電子影像穩定功能，可拍攝 500 萬像素照片和 720p 影像
  - 使用升級版 SDK 2.0 開啟更多編程可能性(支援以MATLAB ROS Toolbox 開發)
  - 可編寫程式進行編隊飛行
  - 可使用手機或平板電腦上的 app 控制
- **相機**
  - 照片最大解析度：2592x1936 (500萬畫素)
  - 錄影解析度：HD1280x720 30p(720p)視頻
  - 格式：MP4
  - 100m 圖傳距離

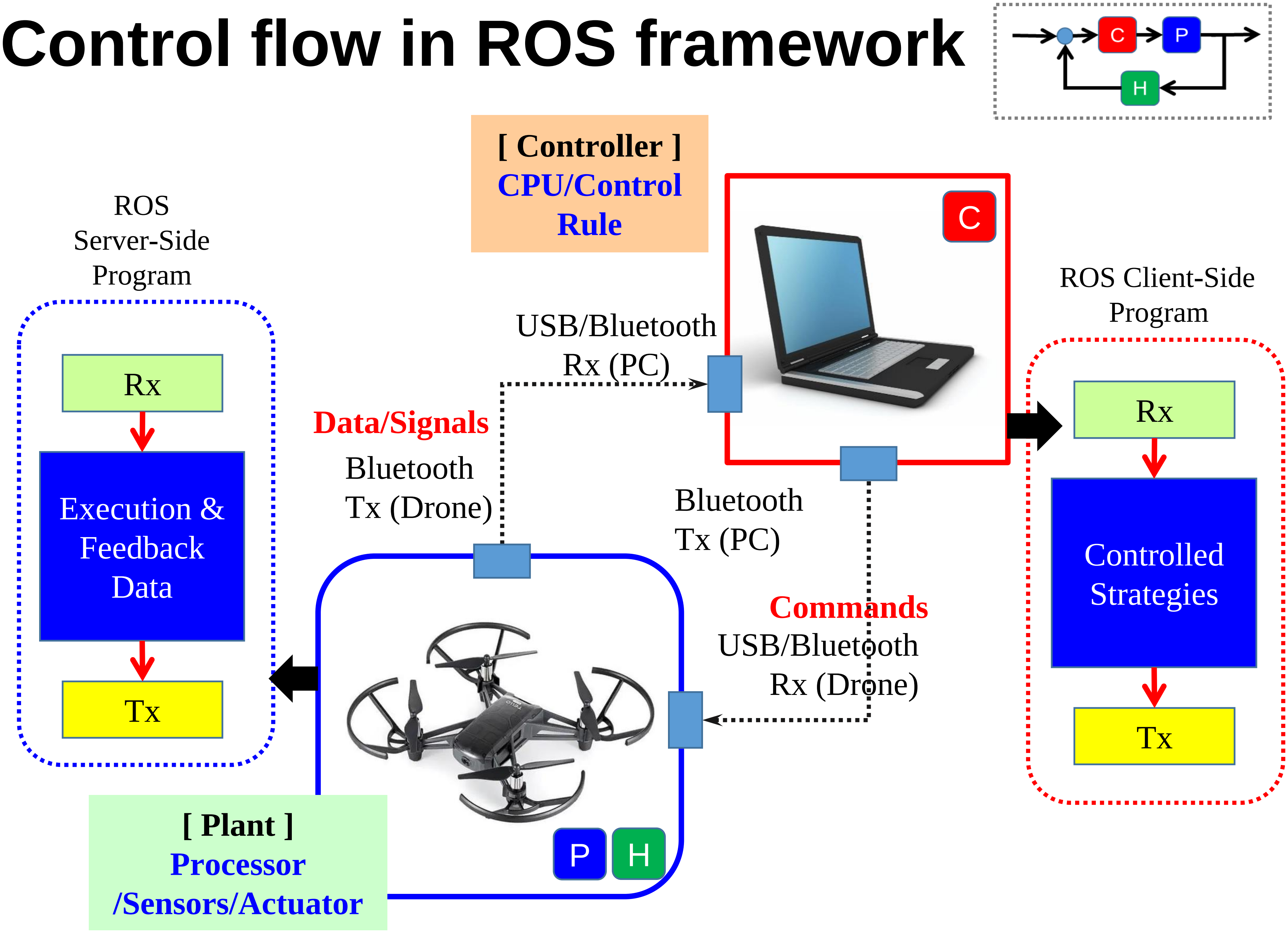
- **電池**
  - 容 量：1100 mAh
  - 電 壓：3.8 V
  - 電池類型：Lipo
  - 能 量：4.18 Wh
  - 電池整體重量：25±2 g
  - 最大充電功率：10 W

https://www.ryzerobotics.com/zh-tw/tello

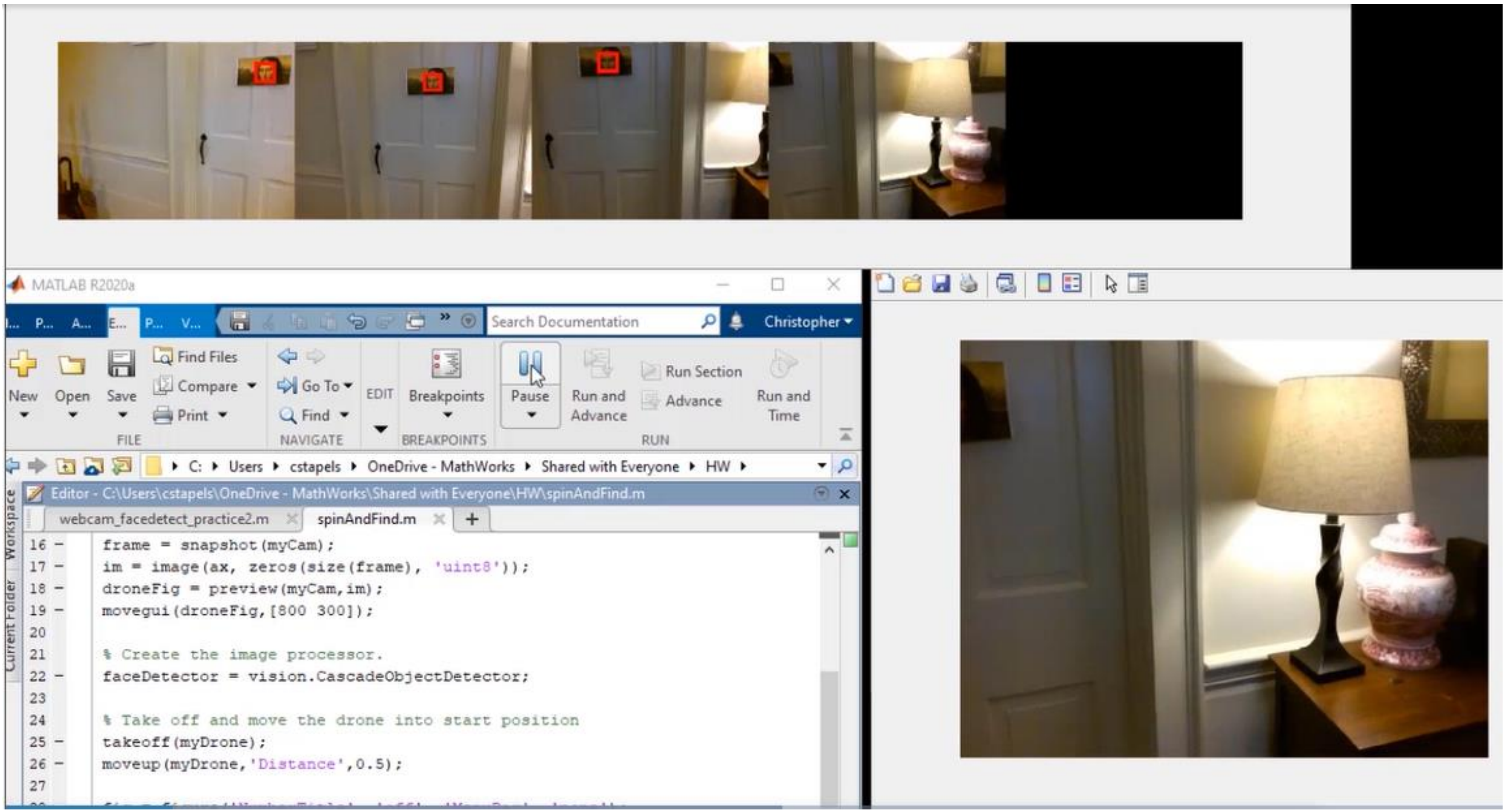
Quadcopter Drone with MATLAB

- Ryze Tello Drone Support from MATLAB
- https://www.mathworks.com/hardware-support/tello-drone-matlab.html
- Step:
  - Connect to Ryze Tello drone over WiFi
  - Use MATLAB commands to take off, move in a specified direction, turn, and land
  - Flip drone in one of four directions
  - Stream images and live video from drone's camera
  - Read flight data including speed, height, orientation, and battery level
  - If you are using the Tello EDU model, you can also perform these tasks in MATLAB:
    - Send abort command to shut off drone motors in case of emergency
    - Fly diagonally along multiple axes at once

Control flow in ROS framework



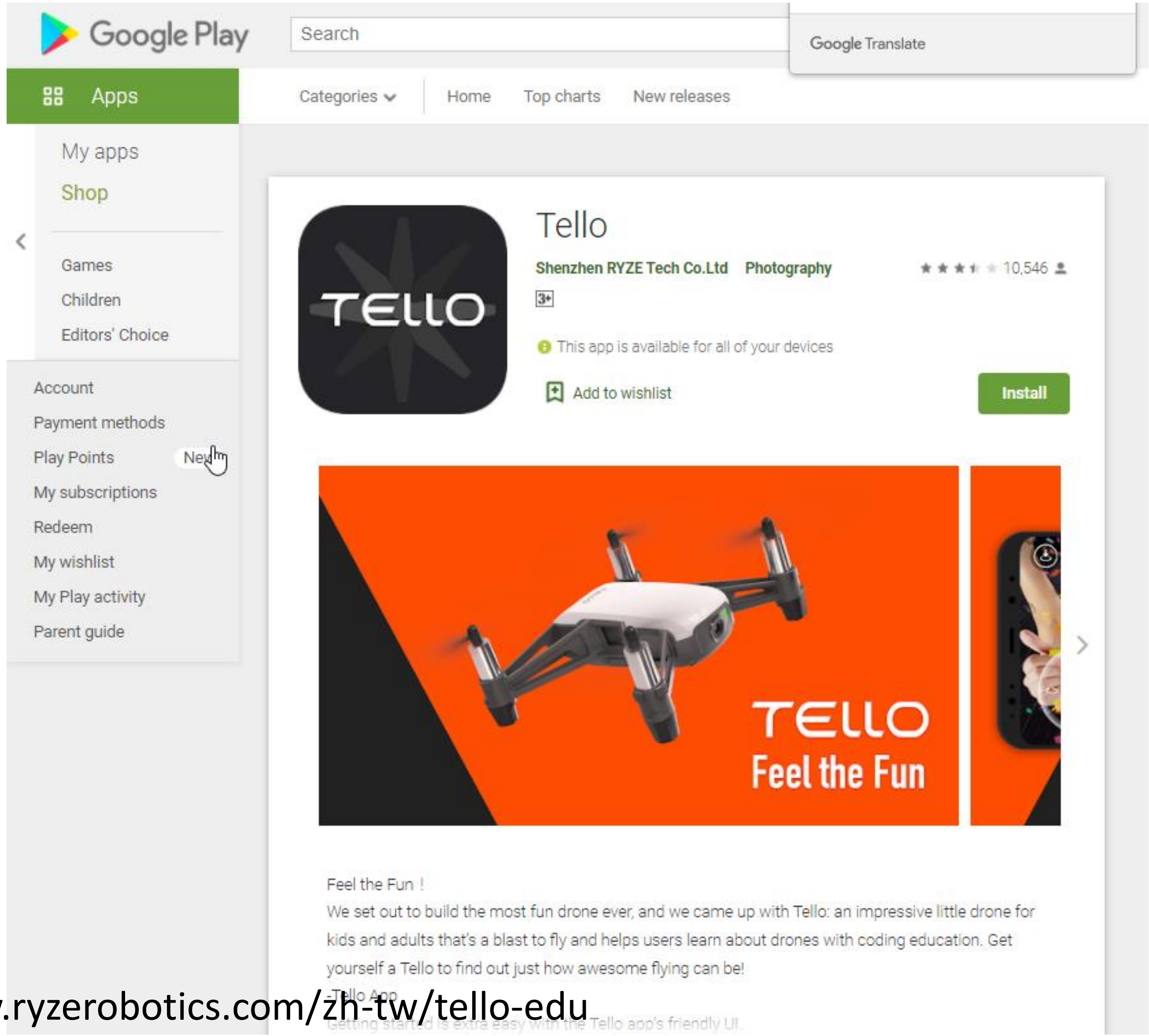
Control Ryze Tello Drones from MATLAB demonstration



https://www.mathworks.com/videos/control-ryze-tello-drones-from-matlab-1595582029947.html

期末專題: 四旋翼UAV控制

Tello mobile app. (Update Drone Firmware)



https://www.ryzerobotics.com/zh-tw/tello-edu

MATLAB Support Package for Ryze Tello Drones Installation

- https://www.mathworks.com/matlabcentral/fileexchange/74434-matlab-support-package-for-ryze-tello-drones?s_tid=srchtitle_Tello_2

MATLAB Support Package for Ryze Tello Drones

by MathWorks MATLAB Hardware Team STAFF

★★★★★ (5)
1.1K Downloads
Updated 22 Sep 2021

Control Ryze Tello drone from MATLAB and acquire sensor and image data

+Follow | Learn More | Download

Ryzeio.mlpkginstall

Overview | Reviews (5) | Discussions (23)

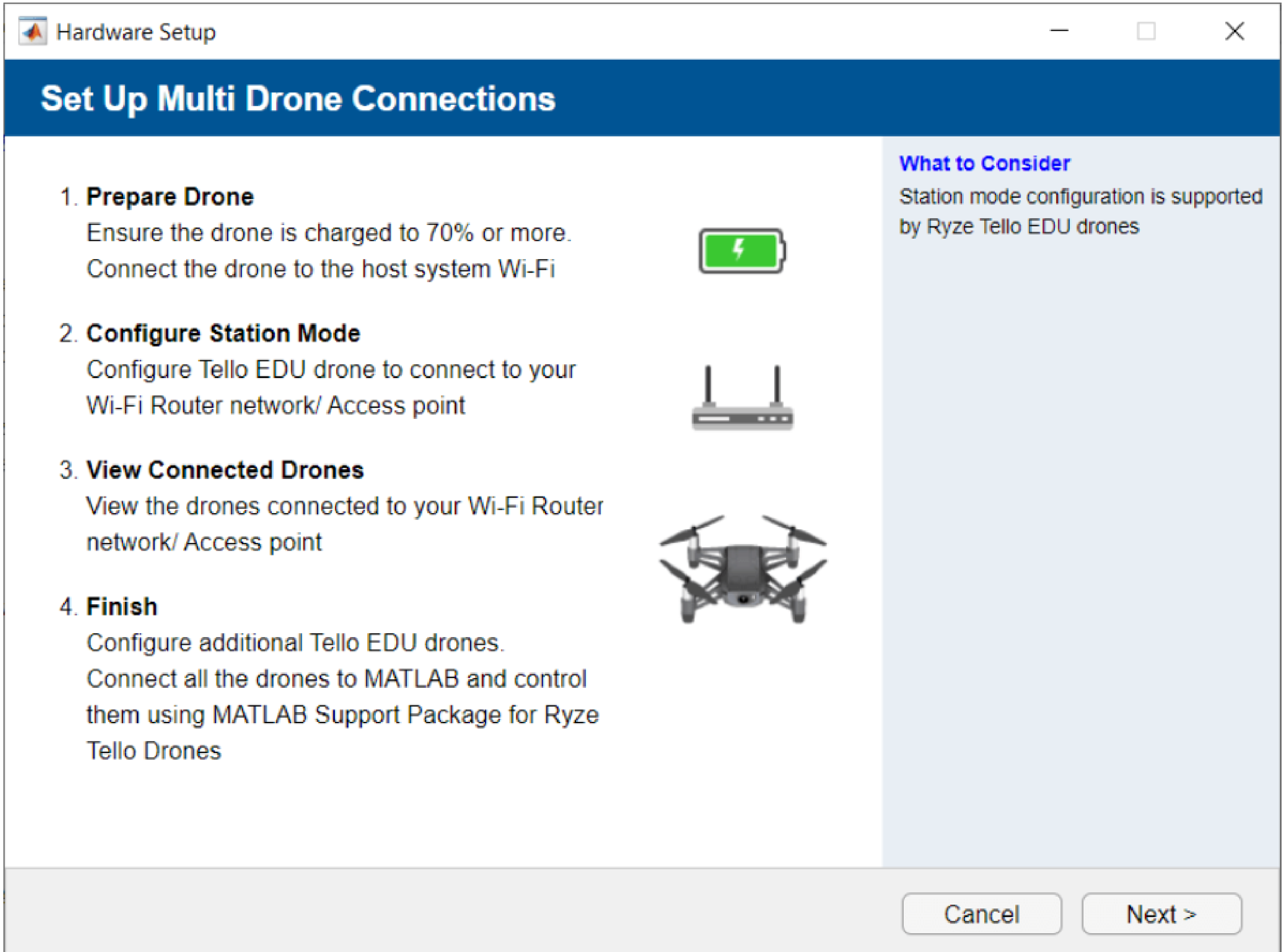
The MATLAB® Support Package for Ryze Tello drones provides programming interfaces that enable you to control a DJI Ryze Tello drone from MATLAB. You can pilot the drone by sending commands to control its direction and orientation. You can also read navigation data and process image data using MATLAB commands.

Features

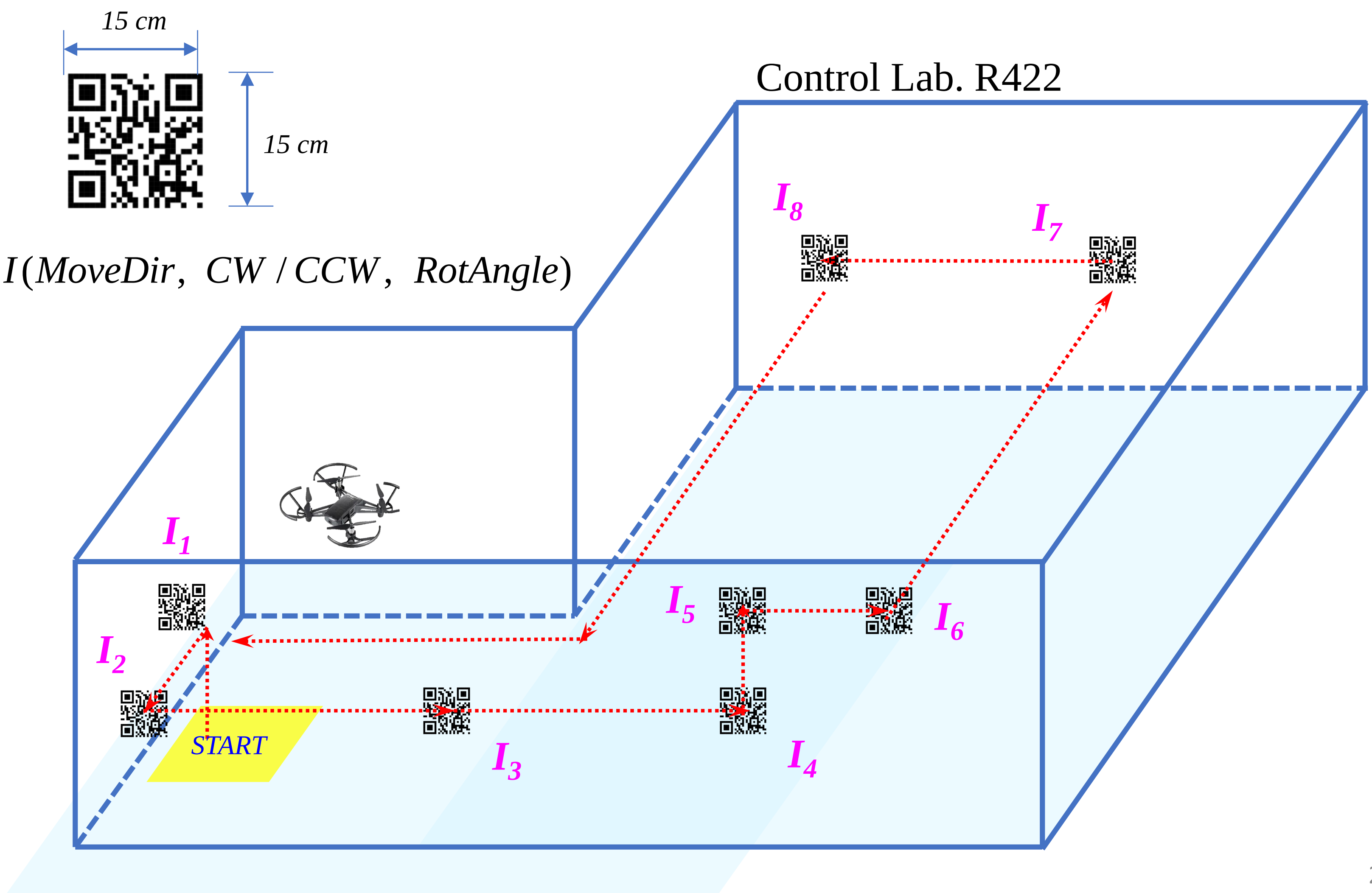
- Use MATLAB commands to take off, move in a specified direction, turn, and land.
- Flip drone in one of four directions.
- Stream images and live video from the drone's camera.
- Read flight data including speed, height, orientation, and battery level.

To know more about the supported drones, capabilities, and features, visit Ryze Tello Drone Support from MATLAB

PC-Based Wi-Fi Network Configuration



Project I: 2D-Barcode Positioning



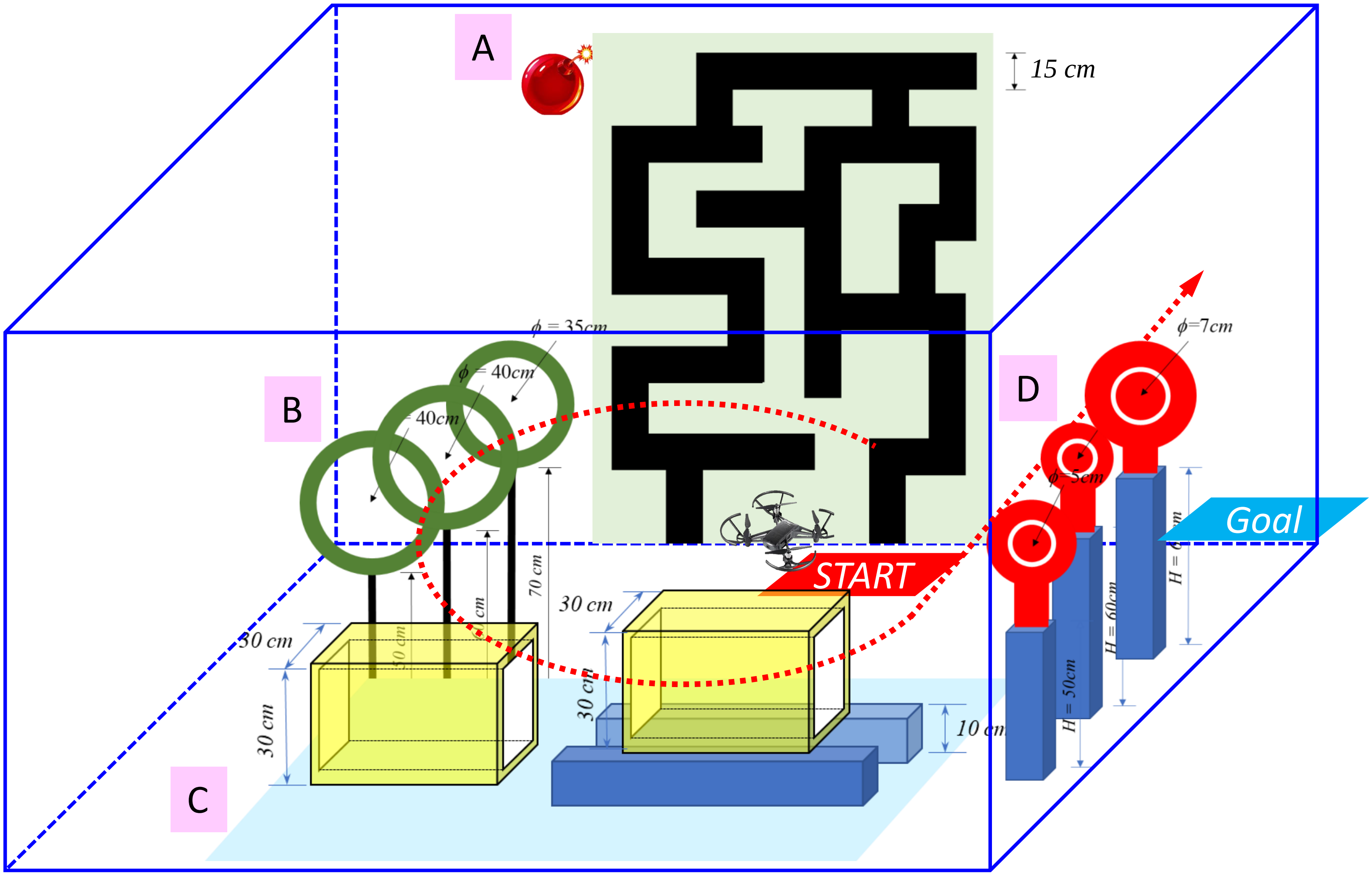
二維條碼定位

- 出發點在I_1位置地面(鏡頭朝向條碼)起飛。
- 每個二維條碼接包含資訊有三個

  $I(MoveDir, CW/CCW, RotAngle)$

  分別下一定位移動方向、四旋翼轉向以及轉角度。
- 每次移動依照順序為$I_1\ I_2\ I_3 \ldots I_1$(需經過每個指定定點)
- 需具有辨識與解讀二維條碼能力
- 具有搜尋條碼能力

Project II: Four Scenarios (A,B,C,D) Test



四種不同情境的測試

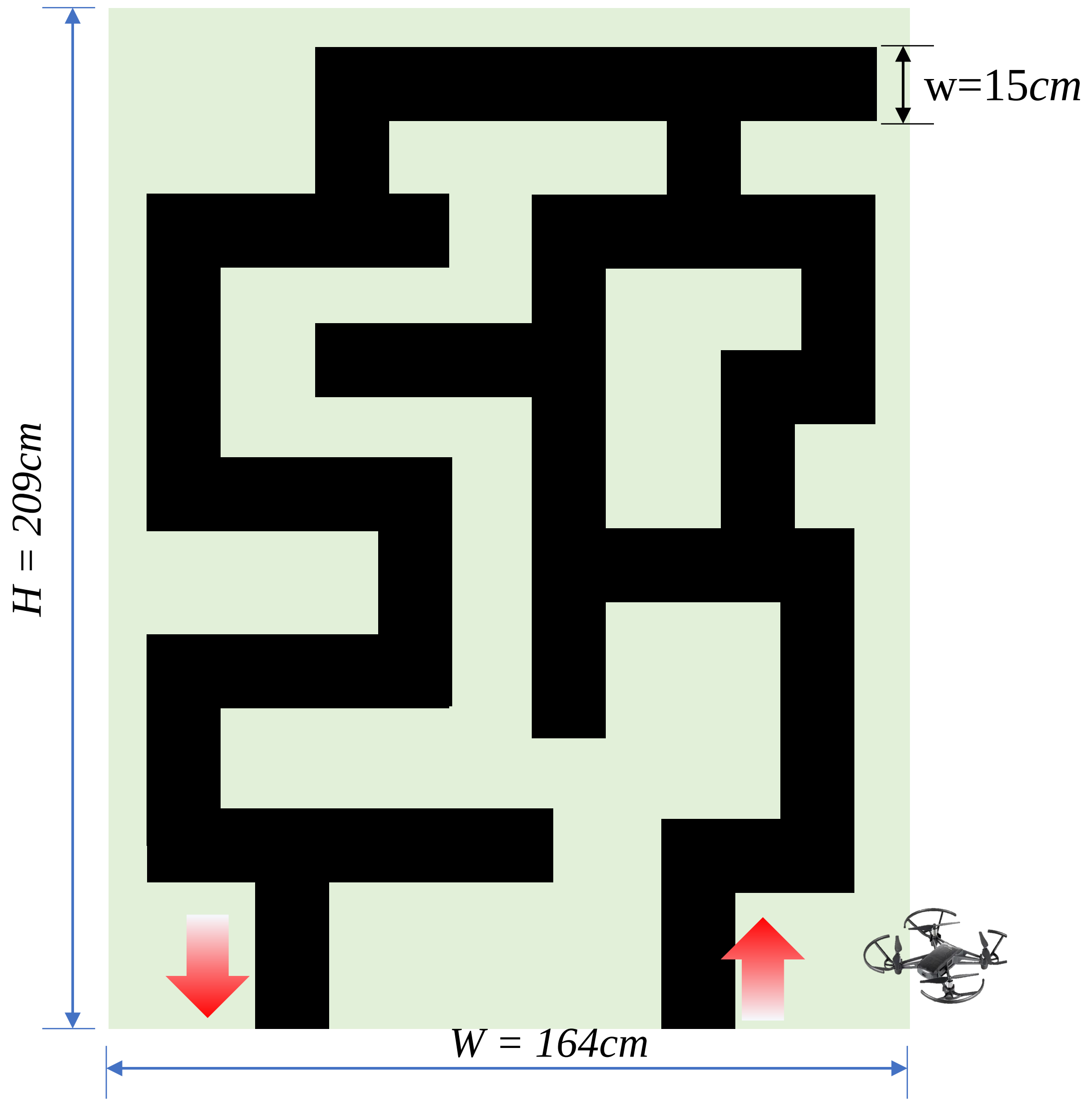
Case A:

1. 將鏡頭轉向路徑，利用影像循跡，沿路徑穩定飛行。需和地圖面保持距離h，也需要維持在路徑軌跡(寬度*15cm*)中線位置d。
2. 需要對影像像素校準。定義像素和距離的關係。
3. 利用PID控制保持定距h和定位d的控制器。
4. 需要具有搜尋路徑的功能。
5. 行進方向由右下方出發，由左下方出口。

Case A: Maze-Tracking Path

Scale：$\frac{1}{11.615}$

W*H =
164*209 cm

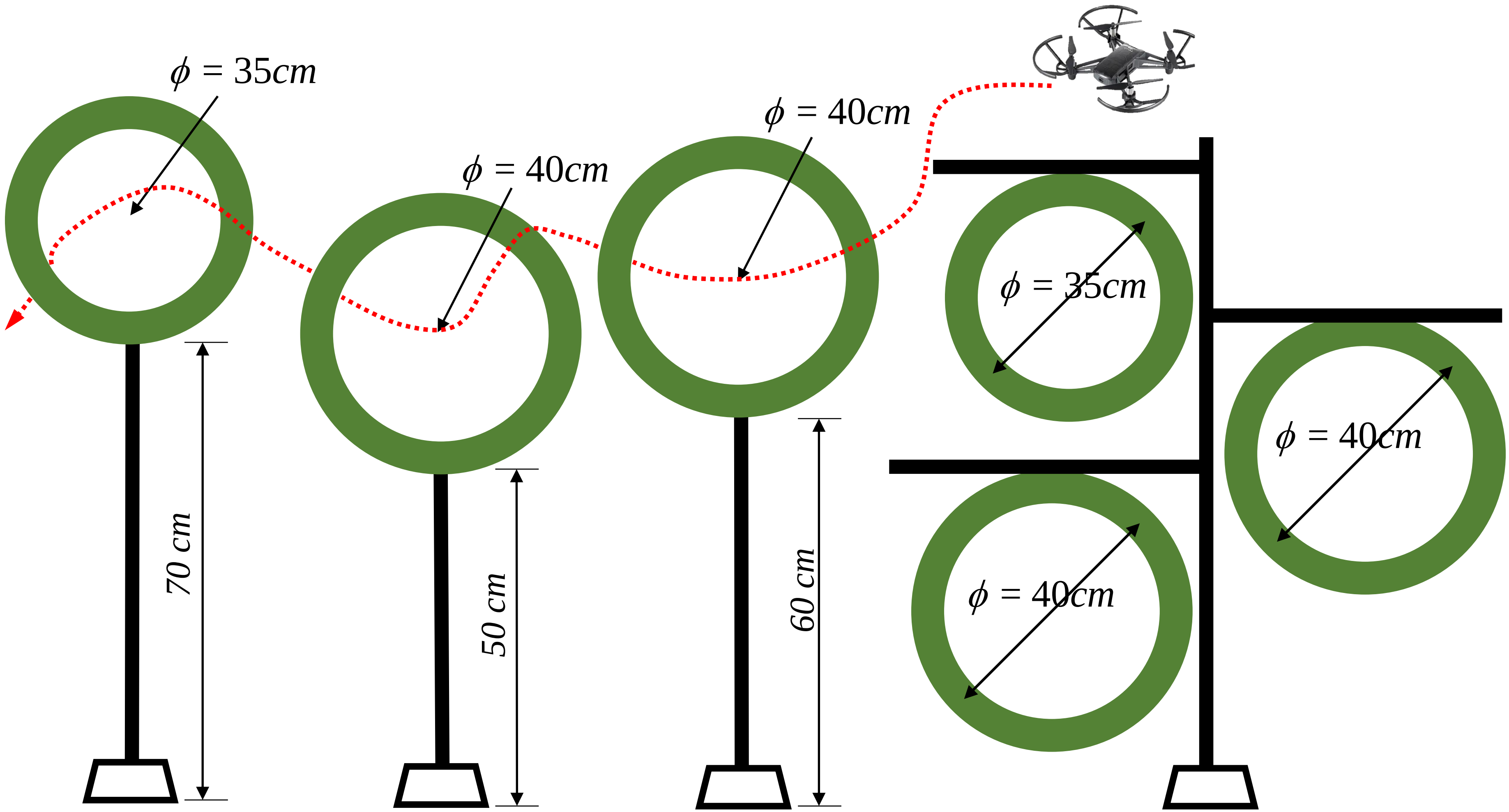


四種不同情境的測試

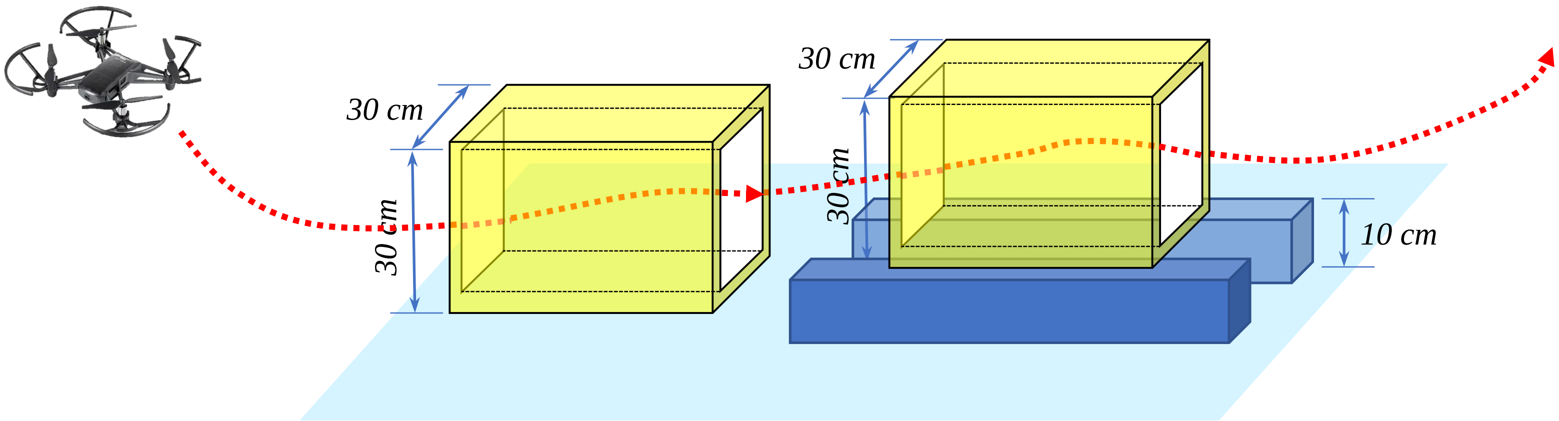
Case B:

1. 將鏡頭轉向前方，利用影像定位規畫飛行路徑。需通過圓圈中心點(考量圓直徑30, 40 cm以及機身20cm*20cm*5cm)。
2. 需要對影像像素校準。定義像素和距離的關係。
3. 依序通過三個圓圈。

Case B: Crossing Circles



Case C: Through the Tunnels



Case C:

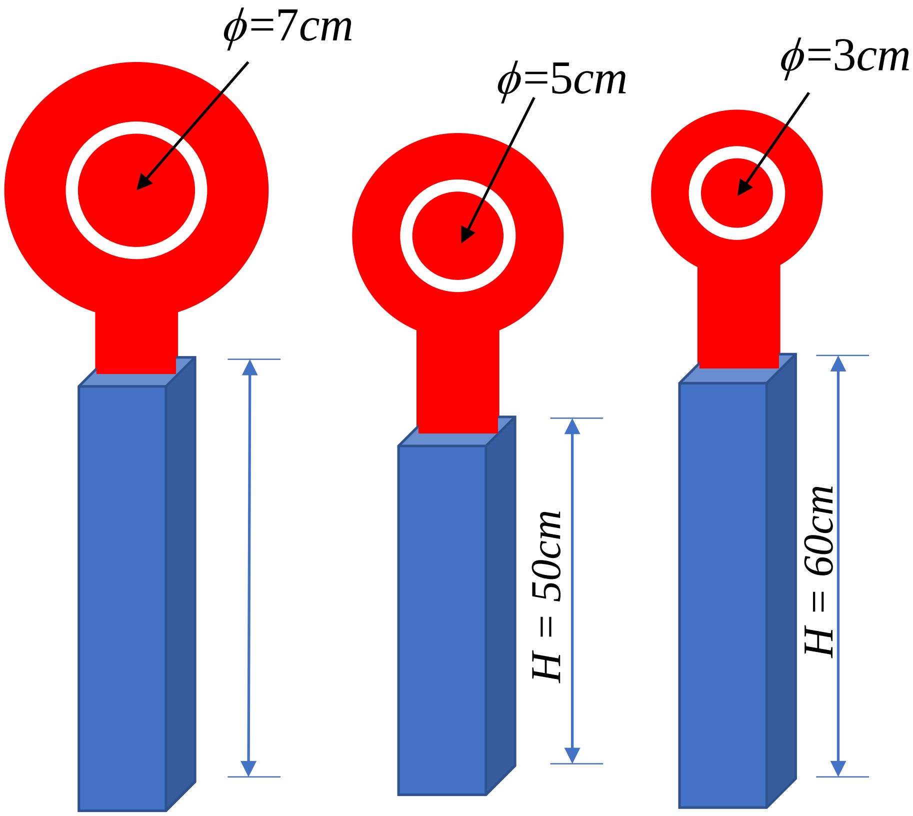
1. 將鏡頭轉向前方，利用影像定位規畫飛行路徑。需通過隧道中心點(考量入口大小*30 cm*見方以及機身*20cm*20cm*5cm*)。
2. 需要對影像像素校準。定義像素和距離的關係。
3. 依序通過兩個隧道。

四種不同情境的測試

Case D:

1. 將鏡頭轉向前方，利用影像定位將目標靶心”推倒”。標靶大小(圓直徑*7cm*, *5 cm*以及*3cm*)。
2. 需要對四旋翼加裝一推桿(自行加裝)。
3. 需要判斷是否推倒目標，可以重複”推倒”動作”，直到完成目的為止。
4. 完成後降落地面結束所有測試。

Case D: Targets Shooting



評分標準：

1. 以單一情境執行完成度與完整度
2. 整合所有情境執行完成度與連貫性
3. 進行穩定度以及完成全程時間
4. 控制系統架構完整性
5. 程式架構完整性(可重現性)
6. 報告說明之完整性