

djitellogy

```

def keyboard(self, key):
    #global is_flying
    print("key:", key)
    fb_speed = 40
    lf_speed = 40
    ud_speed = 50
    degree = 30
    if key == ord('1'):
        self.takeoff()
        #is_flying = True
    if key == ord('2'):
        self.land()
        #is_flying = False
    if key == ord('3'):
        self.send_rc_control(0, 0, 0, 0)
        print("stop!!!!")
    if key == ord('w'):
        self.send_rc_control(0, fb_speed, 0, 0)
        print("forward!!!!")
    if key == ord('s'):
        self.send_rc_control(0, (-1) * fb_speed, 0, 0)
        print("backward!!!!")

```

將 keyboard\_djitello.py 中的 function 複製到你的 code 上 (lab06.py)

並在你的 while 迴圈下面加入此行以便強制控制無人機

```

if key != -1:
    keyboard(drone, key)

```

# Tello 控制 速度 function

```
send_rc_control(self, left_right_velocity,  
forward_backward_velocity, up_down_velocity,  
yaw_velocity)
```

Send RC control via four channels. Command is sent every  
self.TIME\_BTW\_RC\_CONTROL\_COMMANDS seconds.

務必確認輸入的值為int

```
drone.send_rc_control(0, int(z_update) , int(y_update) , int(yaw_update) )
```

## Parameters:

Name	Type	Description	Default
left_right_velocity	int	-100~100 (left/right)	required
forward_backward_velocity	int	-100~100 (forward/backward)	required
up_down_velocity	int	-100~100 (up/down)	required
yaw_velocity	int	-100~100 (yaw)	required

# Tello 控制 移動距離 function

`move(self, direction, x)`

Tello fly up, down, left, right, forward or back with distance x cm. Users would normally call one of the `move_x` functions instead.

Parameters:

Name	Type	Description	Default
<code>direction</code>	<code>str</code>	up, down, left, right, forward or back	<i>required</i>
<code>x</code>	<code>int</code>	20-500	<i>required</i>

`rotate_clockwise(self, x)`

Rotate x degree clockwise.

Parameters:

Name	Type	Description	Default
<code>x</code>	<code>int</code>	1-360	<i>required</i>

Source code in `djitellopy/tello.py`

`rotate_counter_clockwise(self, x)`

Rotate x degree counter-clockwise.

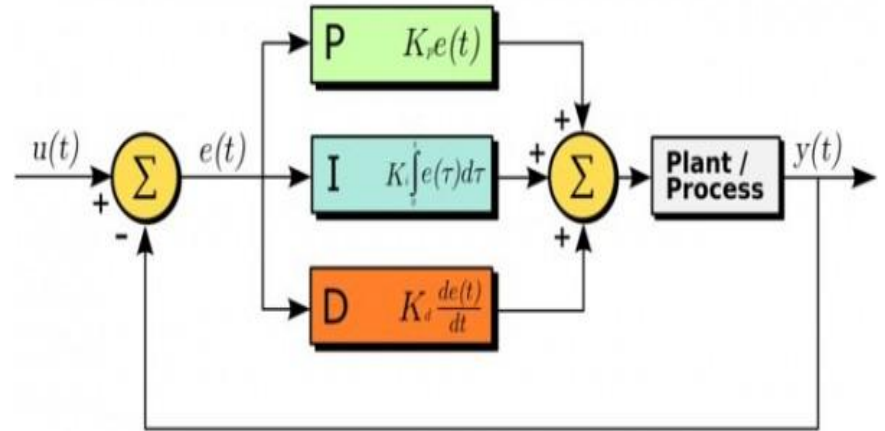
Parameters:

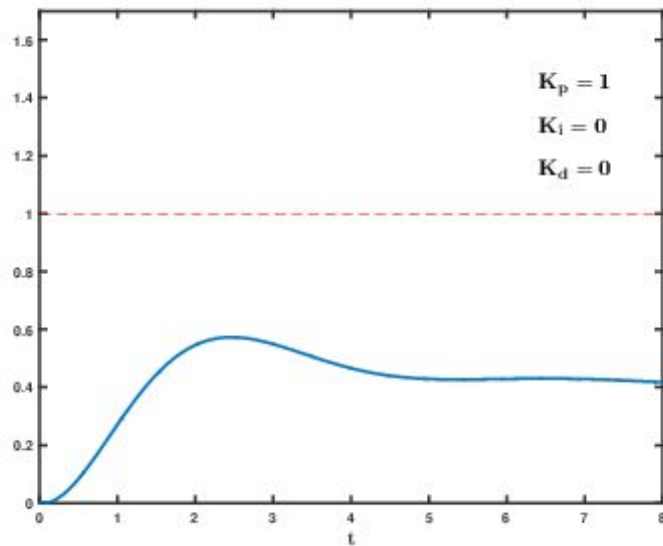
Name	Type	Description	Default
<code>x</code>	<code>int</code>	1-360	<i>required</i>

# PID Control

# Theory

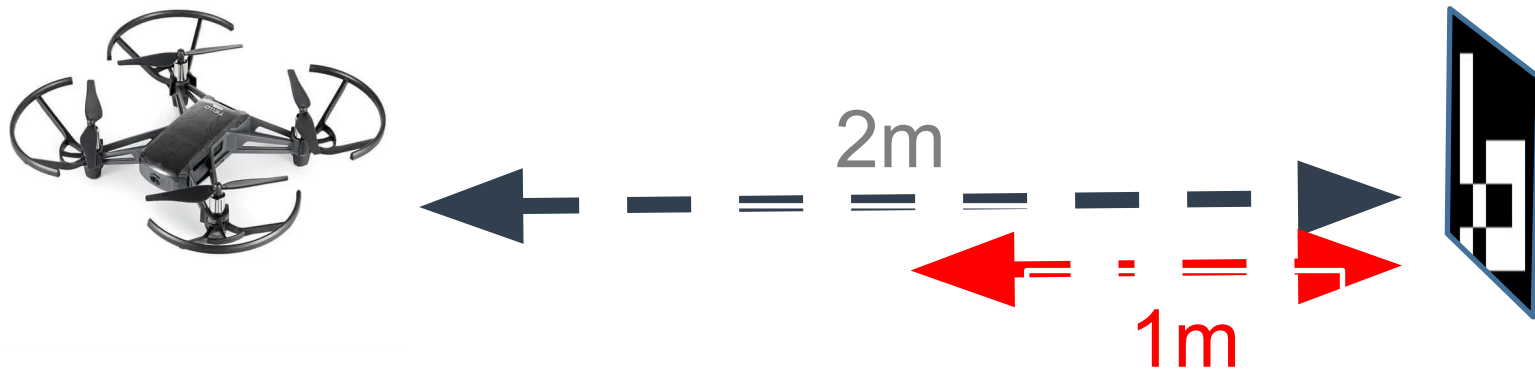
$$u = \underbrace{K_p e}_{\text{Proportional Term}} + \underbrace{K_i \int_0^t e dt}_{\text{Integral Term}} + \underbrace{K_d \frac{d}{dt} e}_{\text{Differential Term}}$$





調整方式	上升時間	超調量	穩態誤差	穩定性
$\uparrow K_p$	減少 $\downarrow$	增加 $\uparrow$	減少 $\downarrow$	變差 $\downarrow$
$\uparrow K_i$	小幅減少 $\searrow$	增加 $\uparrow$	大幅減少 $\downarrow\downarrow$	變差 $\downarrow$
$\uparrow K_d$	小幅減少 $\searrow$	減少 $\downarrow$	變動不大 $\rightarrow$	變好 $\uparrow$

# Step1:



欲修正的誤差(error):  $2\text{m} - 1\text{m} = 1\text{m}$



## Step2:



利用PID去smooth原本的誤差

1m  $\longrightarrow$  0.4m

將誤差轉換成速度給無人機

## Step3:



根據無人機飛行的  
狀況調整PID

1. 先把I, D設為0
2. P: 無人機會停在你設定的距離附近
3. I: 無人機會在設定的距離附近抖動
4. D: 停止抖動

# pyimagesearch

將pyimagesearch資料夾複製到與lab06.py同目錄下

名稱	修改日期	類型	大小
__pycache__	2022/3/29 下午 02:13	檔案資料夾	
__init__.py	2020/6/16 下午 12:01	PY 檔案	0 KB
objcenter.py	2020/6/16 下午 12:01	PY 檔案	2 KB
pid.py	2020/6/16 下午 12:01	PY 檔案	2 KB

```
import cv2
import numpy as np
#import tello
import time
import math
from djitellopy import Tello
from pyimagesearch.pid import PID
```

from pyimagesearch.pid import PID

# pyimagesearch

在main中宣告會用到的pid

```
def main():
    ...# Tello
    ...drone = Tello()
    ...drone.connect()
    ...#time.sleep(10)
    ...
    ...global is_flying
    ...# Get the parameters of camera calibration
    ...fs = cv2.FileStorage("calibrateCamera.xml", cv2.FILE_STORAGE_READ)
    ...intrinsic = fs.getNode("intrinsic").mat()
    ...distortion = fs.getNode('distortion').mat()

    ...z_pid = PID(kP=0.7, kI=0.0001, kD=0.1)
    ...y_pid = PID(kP=0.7, kI=0.0001, kD=0.1)
    ...yaw_pid = PID(kP=0.7, kI=0.0001, kD=0.1)

    ...yaw_pid.initialize()
    ...z_pid.initialize()
    ...y_pid.initialize()
```

kP, kI, kD的值可以自己挑整,  
以下為助教的範例宣告值

# pyimagesearch

```
z_update = tvec[i,0,2] - 100
print("org_z: " + str(z_update))
z_update = z_pid.update(z_update, sleep=0)
print("pid_z: " + str(z_update))
if z_update > max_speed_threshold:
    z_update = max_speed_threshold
elif z_update < -max_speed_threshold:
    z_update = -max_speed_threshold
drone.send_rc_control(0, int(z_update//2), 0, 0)
```

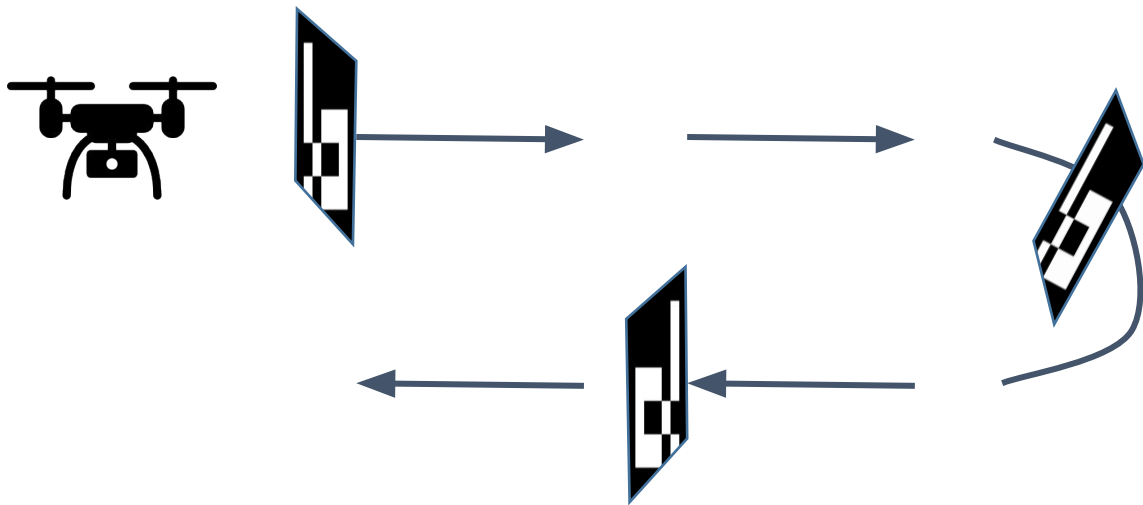
使用時, 先算出想要丟給無人機的速度, 算出其pid值之後再丟給無人機

最好限制其最高速度以防失控

# Lab06

# 追蹤marker移動

1. 鍵盤可以控制無人機移動(20%)
2. 無人機追蹤人手持的Marker移動, 六個方向+旋轉都要可以正常運作(80%)



# 估計無人機與marker的距離

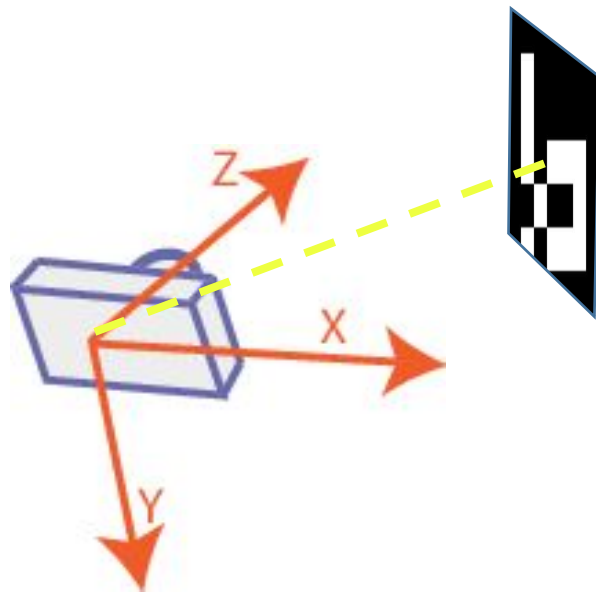
tvec: 上次lab估距離

rvec: 旋轉矩陣

1. 用R乘以Z軸(0, 0, 1)得到Z'
2. 將Z'投影到XZ平面得到向量V
3. 求出Z與V的夾角(rad)轉換成degree

```
dst = cv2.Rodrigues(src)
```

```
math.atan2(z, x)
```





# 注意事項

- 撰寫自動飛行的程式碼時, 一定也要有 keyboard control 功能, 且要有最高優先權, 確保自動飛行狀況不佳時仍能手動控制。