無人機與電腦視覺專題

物理三乙 羅芷銨 (10912222)
指導教授：吳啓彬教授
(繳交日期:112.9/16)

從零開始製作無人機，並且使用 ESP32 來當作控制的核心，在數據的處理中，除了互補濾波之外，還使用了高斯濾波的方法來處理。再來還使用 ESP32 cam 獲取影像，並且參考了 ORB-SLAM 的部分系統修改並寫入 ESP32 中進行運算來取得一些必要數據，以用來避免**磁力計、GPS 失效時的致命問題**。雖然成功的飛行，只可惜因爲元件記憶體與效能限制，目前電腦視覺無法實時的使用在無人機上面。

目的

無人機的用途非常多，除了一般最大衆的空拍外，還包括救援、戰略等，而按照不同平台構型來分類，無人機可分爲固定翼無人機、無人直升機和多旋翼無人機三大種類。其中固定翼無人機與無人直升機的結構較像，其主要的差別在於固定翼無人機是給予前進的推進力，額不是像無人直升機一樣是向上的，所以固定翼無人機並不可以停留在同一個地方，另外多旋翼無人機比較前兩個，這具備著更高的機動性。

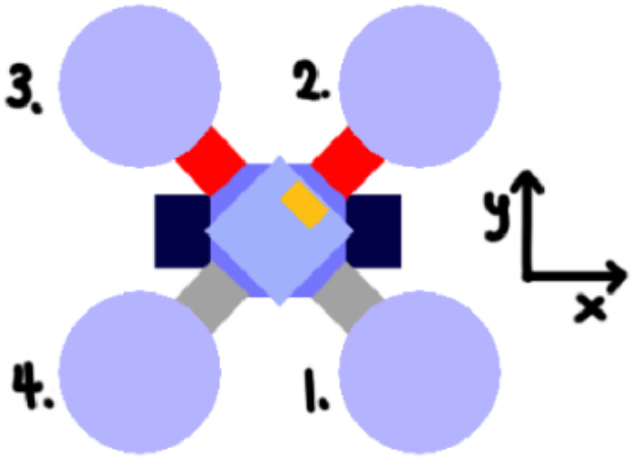
而這次的專題主要是爲了學習以及實現無人機的技術，並且使用 ESP32 來當作無人機的控制核心，其中無人機上加裝了使用 ESP32CAM 來拍攝影像的電腦視覺系統，期望能排除飛行的收不到 GPS、磁力受干擾等致命問題。

原理

一、飛行原理

1. **如何上升:**
   四軸無人機主要靠每個軸的無刷馬達來帶動螺旋槳，螺旋槳會旋轉推動空氣而產生推進力。(剛開始起飛時，因爲地板與無人機距離的比較近，而因爲螺旋槳的旋轉，會產生大量的風壓進入無人機下方的空間中，造成螺旋槳在地面與高空同樣的轉速下，在地面的上升力道要更大)

2. **如何移動:**
   四軸無人機主要是靠每個無刷馬達的不同轉速來達成，圖（一）爲一個簡略的四軸無人機示意圖，如果要往 x 軸飛的話，那就讓 1、2 編號的無刷馬達減速， 3、4 編號的無刷馬達加速，這樣無人機就會往 x 軸傾斜，上升的力道也會從垂直向上變成偏向 x 軸。這要先說到無人機是有分正槳與反槳的，通常會讓 1、3 同一組， 2、4 同一組，並且這兩組其中一組是正槳，另外一組是反槳，這樣進行移動時，產生出的力矩就會相互抵消。

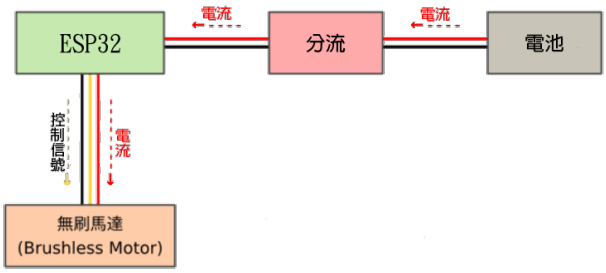


圖（一）. 四軸無人機示意圖

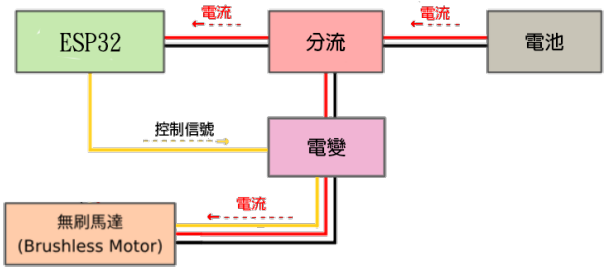
3. **如何旋轉:**
   讓 1、3 編號的無刷馬達減速， 2、4 編號的無刷馬達加速，這樣就會產生某一個方向 (看搭配) 的力矩，使得四軸無人機旋轉。

4. **如何控制無刷馬達轉速:**
   當然如果要操控馬達的轉速，可以直接用 ESP32 去連接馬達，如圖（二），但是這麼做會有一個問題，那就是電流會非常大，這足以超過 ESP32 或其他控制系統的負荷能力，所以才要有 ESC 的存在，如圖（三），電流不會直接流入 ESP32 中，ESP32 只有負責控制而已，另外 ESC 有三條線連通到無刷

馬達當中，這三條會輪流爲通電、接地、斷路的狀態，如果想要看這是如何運作的，可以看我做的 multisim 模擬 (附錄 1)。



圖（二）. 無電變情況



圖（三）. 有電變情況

5. **如何遙控**

無人機上除了要有控制系統外，還會需要能與外界控制器連接的方式，所以這次的專題才會用 ESP32 來做，利用 ESP32NOW 的功能來連接兩個 ESP32，進行遠端控制。

6. **如何控制才能穩定飛行**

要使無人機穩定的飛行，必須要有陀螺儀、加速度計、磁力計、氣壓計，這四項缺一不可，在我的程式中有些元件還扮演了互補的角色，另外要定位無人機就必須要用 GPS，將這些資訊傳入 ESP32 中處理後，最後會使用 PID 的方法處裡數據後回饋到轉速上。

7. **PID 控制**

PID 控制是用來對一個系統變化量進行穩定的方式，主要是由 P 比例、I 積分、D 微分所構成的，用物理角度來說的話，使用 P 就會使一個系統產生簡諧運動，D 就像是阻尼，一般來說在簡單的環境下只需要 P、D 就能夠使系統穩定，但是有可能會有誤差、外力因素的影響，需要再加上 I 才能穩定，例如無人機被風吹到左邊，而使用 I 就會產生一個反方向的力使的無人機飛回右邊，或是 X、Y 軸的初始位置錯誤，使用 I 後，經過一定時間的飛行後就會逐漸導正。

8. **X、Y 軸旋轉**

首先先將加速度計的數值處理一下，公式是 :

$$X_A = tan^{-1}(\frac{AY}{\sqrt{AX^2 + AZ^2}}) - \mu_1$$
$$Y_A = tan^{-1}(\frac{-AX}{\sqrt{AY^2 + AZ^2}}) - \mu_2 \qquad (1)$$

AX, AY, AZ 是加速度計讀到的數值，μ_1, μ_2 是與眞實情況的誤差值，這要自己去測量。這公式也很好理解，就是將加速度方向投影到不同軸平面，再用比例關係丟到 arctan 就可以求出角度。

再來因爲加速度計的值準確卻不精確，所以要利用陀螺儀與加速度計去做濾波。

$$X = T(X + GX \cdot \delta) + (1 - T)X_A$$
$$Y = T(Y + GY \cdot \delta) + (1 - T)Y_A \qquad (2)$$

GX, GY, GZ 是陀螺儀讀到的數值，X, Y 是旋轉角度，Δ 指的是程式循環一次要多少秒 (我的固定會是 0.006)，T 值介於 0~1 之間，通常要接近 1 才會有明顯的效果，我是設定 0.995。

最後我還有加上一個類似高斯濾波的方法，但是在這個部份效果不明顯，在磁力計比較明顯。

9. **Z 軸旋轉**

這主要是利用磁力計來找出地磁方向來控制目前的 Z 軸旋轉。最初最重要的是要先校準磁力計，不然無法正確的得到正確的旋轉角度。

而找出地磁方向的公式爲:

$$\begin{aligned} Xh =& MX \cdot cos(X_O) \\ &+ MY \cdot sin(Y_O) \cdot sin(X_O) \\ &+ MZ \cdot cos(Y_O) \cdot sin(X_O) \\ Yh =& MY \cdot cos(Y_O) \\ &+ MZ \cdot sin(Y_O) \end{aligned} \qquad (3)$$

$$Z_A = \begin{cases} tan^{-1}(\frac{-Yh}{Xh}) & \frac{-Yh}{Xh} >= 0 \\ \pi + tan^{-1}(\frac{-Yh}{Xh}) & \frac{-Yh}{Xh} < 0 \end{cases}$$

因爲此數值準確但不精確，所以要與陀螺儀的 Z 軸去做濾波。

$$Z = T(Z + GZ \cdot \alpha \cdot \delta) + (1 - T)Z_A \quad (4)$$

Z 即爲 z 軸的旋轉角度，T 值我設定爲 0.85，α 是受到的力矩比例，這是可以用計算得到，也可以直接去測量得出。
雖然濾波了，但是其實 PID 輸出的 D 值的結果還不算很平穩，而我也想了一個方法解決這個問題。這方法就是利用高斯分布的權重來看要加多少比例，如果離誤差標準差越遠，代表不是雜訊的機率越小，所以將 (4) 修改如下。

$$Z = T(Z + GZ \cdot \alpha \cdot \delta) \quad (5)$$
$$+ (1 - T)(Z + (Z_A - Z)(1 - e^{-\frac{(Z_A - Z)^2}{2\sigma^2}}))$$

10. **高度**
利用氣壓計來獲取數據，再把它轉爲目前高度，這部分我都交由 DFRobot BMP3XX 的庫來解決了。

再來是利用加速度計累加得出垂直速度，這樣的話就可以與氣壓計數值得出的速度去互補，增強穩定性 (D 的穩定性)，要注意的是因爲氣壓計會有延遲，所以加速度計累加得出的速度數值要稍微 delay 下，算法爲下。

$$Xr = |AY \cdot sin(X_O)|$$
$$Yr = |AX \cdot sin(Y_O)|$$
$$Zr = AZ \cdot cos(X_O) \cdot cos(Y_O)$$
$$V_A = Xr + Yr + Zr \quad (6)$$

V_A 就是垂直加速度，然後先將數值用 list 紀錄起來，過個幾個迴圈後再使用。再來先將高度數值濾波後計算出加速度數值。

$$H_s = T(H_s + V_{Ar} \cdot \delta) + (1 - T)\frac{dH}{dt} \quad (7)$$

H_s 是輸出的垂直速度，H 是高度數值，V_{Ar} 是前 r 個的垂直加速度數值，r 我設定是 30。取得目前的 H 值後，再用糾正後的速度值再跟高度值去做濾波。

$$H = T \cdot (H_{last} + H_s \cdot \delta) + (1 - T)H \quad (8)$$

H_{last} 是上一個迴圈的高度數值，上面 T 我都是設定 0.995。

11. **GPS 位置**
將數值處理完後後丟到 PID 裡後用來控制 X, Y 軸角度的目標值，並且限定 P 值的結果只能只能 $\pm 10°$。

要在 GPS 使用濾波的話，要先利用 X、Y 軸角度值帶入 sin 中就得出了加速度的大小，並且利用與高度一樣的濾波方法，測試的話就一定要起飛才可以測試，所以一定要 SD 卡才能存取數據，也要避免在有風的地方測試，因爲風會擾亂準確度。

$$GPSX_s = T(GPSX_s + X_{Ar} \cdot \delta) \quad (9)$$
$$+ (1 - T)\frac{dGPSX}{dt}$$

$GPSX$ 是 GPS 數據經過 Z 軸數據進行旋轉後的 X 軸數值，$GPSX_s$ 是高度數值的加速度，X_{Ar} 是前 r 個的 X 軸加速度數值，r 我設定是 27。

$$GPSX = T(GPSX_{last} + GPSX_s \cdot \delta)$$
$$+ (1 - T)GPSX \quad (10)$$

$GPSX_{last}$ 是上一個迴圈的數值，上面 T 我都是設定 0.96。

二、電腦視覺部分的原理

OpenCV 是個開源的電腦視覺庫，裡面含有非常多能用於影像辨識的程式，而 ORB 就是其中一個，主要是用來檢測出兩張圖片的特徵關係，其他類似的方法還有 Harris、SIFT、SURF，但是這三種方法都不比 ORB 還要快，而目前 OpenCV 的同時定位與地圖構建 (SLAM) 的也採用 ORB 算法，名爲 ORB-SLAM。

而找出兩張圖片的特徵關係後，就可以利用兩張圖片的轉換矩陣 (單應矩陣) 來推出圖片的相對旋轉與位移。

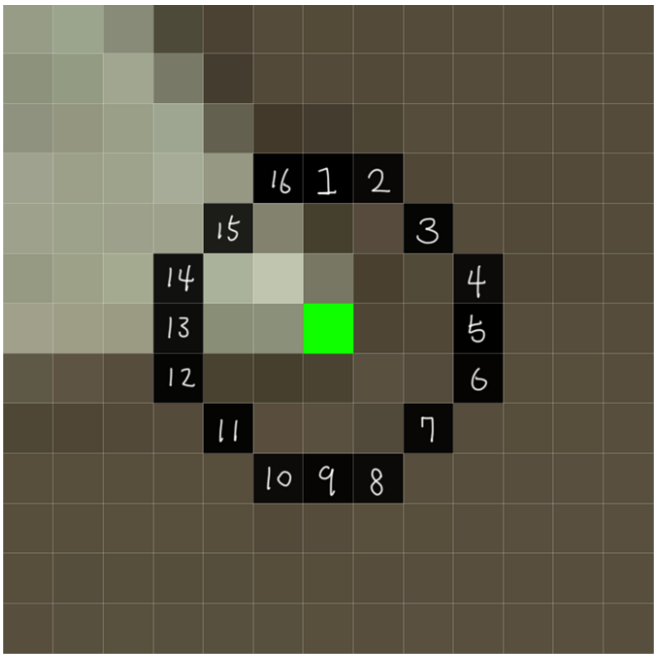
1. **ORB 檢測**
   具體方法是利用 FAST 檢測兩圖的 key point ，並且用 BRIEF 描述子來配對兩者的 key point (特徵點，比如桌子角點)，但是要考慮圖片可能會被旋轉過，所以在 FAST 檢測中，必須找出各自 key point 的質心方向，這樣就可以使 BRIEF 做更精確的配對。

2. **Fast 檢測**
   假定以半徑爲 3 的像素點，並且以順時針爲順序標記 (如圖（四）)，並利用在 1,9 兩點，篩出其像素點灰階值 (0~255) 相差超過 10 的點，之後再來篩掉以半徑爲三的 5,13、3,11、7,15 的點、 2,10、4,12、6,14、8,16 的點，出來的結果會有兩種情況，一個是中心像素小周圍大，另一個是中間大周圍小，再來算出他們的脚點分數，此爲即爲候選點，最後再利用四元樹篩掉密集的特徵點 (以角點分數)，以漸少計算量 (但我不是這麼做)。

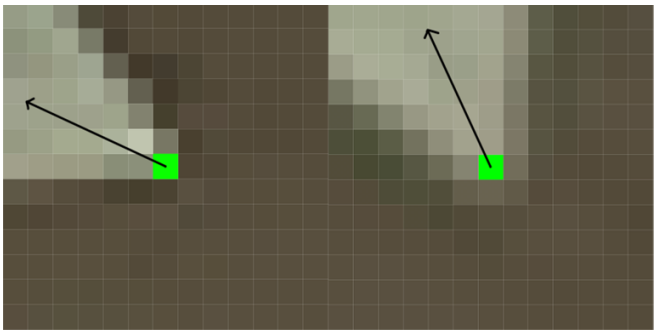
3. **質心方向計算**利用以下公式取得質心位置。

$$
\begin{aligned}
m_{10} &= \sum_{n=-15}^{15} x_n (C_n - C_c) \\
m_{01} &= \sum_{n=-15}^{15} y_n (C_n - C_c)
\end{aligned}
\tag{11}
$$

此公式大致的意思就是以半徑爲 15 的全部點乘一個權重去做灰階值相減的加總，C_c 是中心灰階值，並且離圓心越遠，權重 x_n, y_n 越大。取得的 $(m_{10}、m_{01})$ 就是質心方向，如圖（五）。



圖（四）. 16 個點視意圖



圖（五）. 質心方向

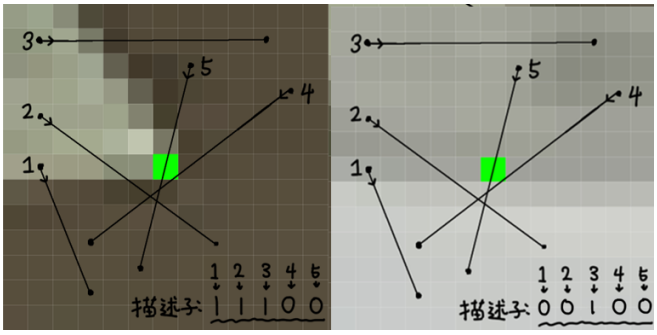
4. **harris 分數**
   利用以下公式取得 harris 分數。

$$
M = \sum_{x=-15}^{15} \sum_{y=-15}^{15} \begin{vmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{vmatrix} \tag{12}
$$

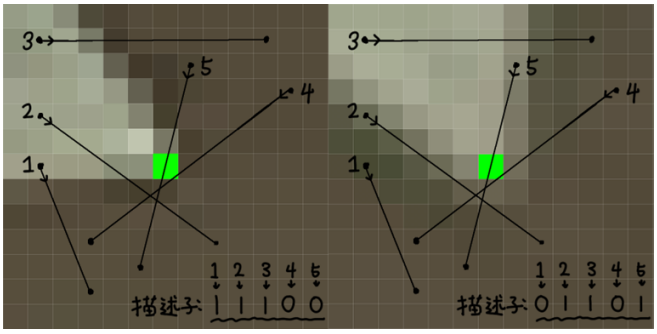
M 爲角點分數，I_x、I_y 是 x、y 方向的梯度，此公式的意義就有點像是在計算此圖像的梯度響應值，並且同個 key point 經由旋轉後，此 harris 分數並不會區別太大，所以可以利用 harris 分數來減少 BRIEF 配對的計算量，方法就是不去配 harris 分數太小的 key point。

5. **BRIEF 描述配對**

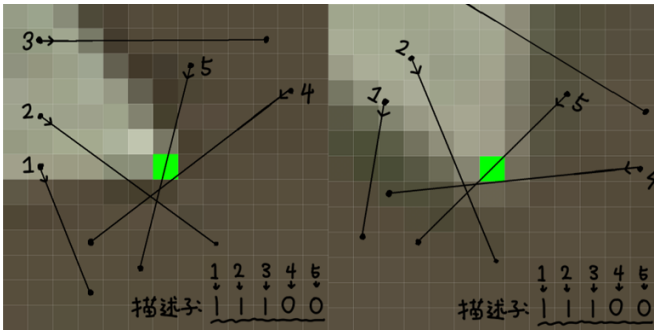
類似漢明碼的運作方式，每個 key point 都有他的 BRIEF 描述子群 (256 個)，描述的方式是以固定的點組去比較像素點大小，比他大就是 1，比他小就是 0 (如圖（六）)。但是爲了避免點與點之間可能有經過旋轉 (如圖（七）)，必須利用質心方向來旋轉這些固定的點組 (如圖（八）)，以免造成配對失效。(將圖片高斯模糊能夠更精準的描述)



圖（六）. BRIEF 描述子



圖（七）. BRIEF 描述子 - 旋轉問題



圖（八）. BRIEF 描述子 - 解決方法

6. **單應矩陣與本質矩陣**

(a) 單應矩陣是用來描述兩照片之間的關係，簡單來說就可以利用前一幀照片去乘單應矩陣來還原當前幀照片。

(b) 本質矩陣是用來描述兩相機空間之間的關係，利用四個 key point 位置之間所形成的形狀來判斷旋轉角度與位移。

兩矩陣都可以用來找出相機之間的關係，但是各有優缺點，本質矩陣不能處理平面場景，單應矩陣不能處理非平面的場景。

7. **找出單應矩陣並還原角度**

單應矩陣關係式如公式 (13)，解出來後會得到公式 (14) ，並用此公式去推算出單應矩陣值，最少需要 4 個 key point 配對才可以求出單應矩陣 (因爲有 8 個自由度)，接下來利用奇异值分解 (SVD Jacobi 特徵值算法) 來求出此方程式，但是爲了更好的準確度，要使用隨機 8 個 key point 配對來利用隨機抽樣一致 (RANSAC) 找還原後的誤差值最小的結果，並且誤差不能超過一定閾值，假如都不符合，此幀作廢。求出單應矩陣後，利用相機內參矩陣來還原世界空間，並且求出位移與旋轉。

$$\begin{bmatrix} x_1' \\ y_1' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -y_1x_1' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -y_2x_2' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -y_3x_3' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -y_4x_4' \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix} \tag{14}$$

8. **相機內參矩陣**

如圖 15 表示的一樣，f_x 與 f_y 爲相機座標反映到世界座標的縮放大小，c_x 與 c_y 是圖片的中心像素點位置，ESP32CAM 的 OV2640 相機在 320x240 解析度的內參矩陣我測量出來如圖所示：

$$\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 400 & 0 & 160 \\ 0 & 400 & 120 \\ 0 & 0 & 1 \end{bmatrix} \tag{15}$$

所需材料

1. **動力部分**

   (a) **鋰電池** :3S 11.1V 2800mAh 40c 鋰電池，3s 鋰電池代表說是三顆標準電壓 3.7V 的鋰電池串聯在一起，40c 代表這顆電池有自身容量幾倍的放電能力，例如說我這就有 2800x40=112A 的放電能力。我買的電調是 30A 的，所以鋰電池最好選大約 30x4(120)A 放電能力的電池。

   (b) **ESC** :BLHeli-S 30A，而此電變在使用前必須要使用 BLHeliSuite 軟體來控制一些參數，例如說溫度牆、訊號寬度範圍、馬達轉向等等。當此電變連接上電源與訊號後，如果此時的訊號爲最小值，那馬達就會發出『低中高 低 高 (Hz) 』的聲音，聽到此聲音代表飛行準備就緒，當訊號寬度加大後，就可以順利開始運作。

   (c) **無刷馬達** : 新西達 2212 1400KV。無刷馬達的旋轉方式是線圈不動，更改磁極的方式來做旋轉，並且此種類馬達都會有 KV 值，KV 值表示的是電壓每升高 1 伏特，轉速增加的數值，而 KV 值越高代表馬達內部的磁極越少，因此給予訊號後的馬達轉速快，但是扭力小。空拍機通常 KV 值要比穿越機低，因爲空拍機的螺旋槳通常都比較大，需要更大的扭力。

   (d) **螺旋槳** : 乾豐 ABS 正反槳 8045，8045 意味著直徑 8 英寸，螺距爲 4.5 英寸，並且螺旋槳使用前，建議用螺旋槳平衡器後用膠帶平衡螺旋槳，不然螺旋槳在旋轉時會震動。

   (e) **F330 機架** : 對角線距離爲 33 公分，所以最大只能容忍 9 英寸的螺旋槳，底盤有兩片部分露出的金屬片，可以讓電池可以直接焊在上面。

2. **控制部分**關於控制端其實原先是用 Arduino 與無線電收發模組來進行控制的，但是因爲爲了電腦視覺系統以及方便程度，後來才改用 ESP32。

   (a) **控制端** : NodeMCU-32S，就是 ESP32，只是比原廠的小一點。

   (b) **遙控器** : JoyStick 與按鈕模組，JoyStick 可分爲 X 軸與 Y 軸，使 Arduino 接收類比訊號，除此之外，這也能往中間壓下傳送訊號，但是可惜的是因爲這是遊戲專用的模組，所以在斜向方向的 X、Y 軸壓到底後並沒有很線性。另外因爲 ESP32 的 ADC2 已經被 WiFi 佔用了，所以只剩下 6 個通道可以連接這些模組，蠻可惜的就是了，或是可以設計個電路來用也是不錯。

   (c) **天線** : 用於控制端。

3. **飛控部分**

   (a) **飛控端** : NodeMCU-32S。

   (b) **9 軸感測器** : MPU-9250，9 意味者陀螺儀 3 軸、加速度計 3 軸、磁力計 3 軸的總和。

   (c) **氣壓計** : BMP390L，此氣壓計傳感器還內置了一個溫度傳感器，以補償由於溫度變化而對壓力產生的影響。要注意的是氣壓計不可以直射太陽，並且在使用時建議再多加一個罩子，防止風吹到導致氣壓不穩。

   (d) **GPS** : NEO-M9N，使用的是原廠的天線，GPS 是由訊號接收器接收 GPS 衛星的電磁波訊號，並且藉由都卜勒效應的原理，進而算出用戶的位置與海拔高度，通常用戶端的 GPS 必須要接收到 4 顆 (或以上) 的衛星的電磁波訊號才可以定位。
   使用時先用原廠的軟體 u-center 更改參數，包括設定 10 Hz、調整 Baud、只使用 GNGGA 語句之類的。

   (e) **天線** : 用於飛控端。

   (f) **相機** : ESP32CAM，用於估計三維位置，還有遠端畫面功能。

4. **其他**

   (a) **蜂鳴器** : 這主要是確保電池電量的方法，我把它設定成只要鋰電池當中的單層電量低於 3.7V 就發出嗶嗶的聲音，避免過度使用導致鋰電池損耗。

(b) **線**：主要用於連接元件與主板的導線。

(c) **降壓模組**：將鋰電池的電壓降到 5V 給飛控端與控制端使用。

(d) **鋰電池充電器**：買了鋰電池後的必須品。

(e) **轉接頭**：主要是鋰電池充電器與電池的轉接頭。

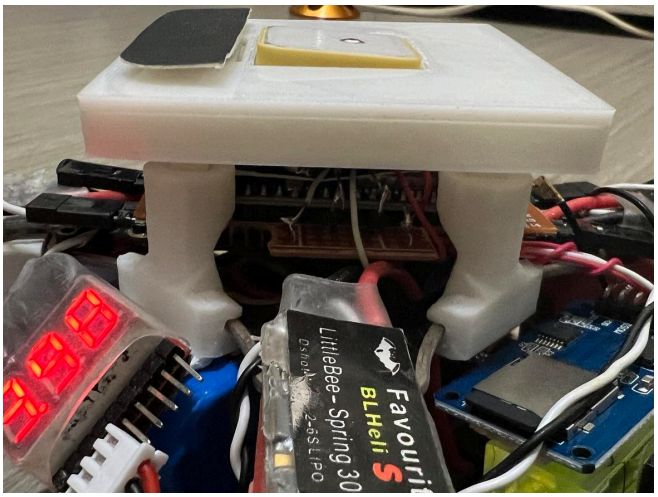
(f) **螺旋槳平衡器**：一個很像很小的扯鈴的裝置，利用磁浮浮空，將螺旋槳安裝上去能夠知道螺旋槳是否有平衡。

結構

圖（九）. 無人機整體樣貌

目前無人機的結構如（九），圖中中下角是 ESP32CAM ，用於拍攝影像與視覺系統，卡在中間的藍色部分爲電池，其他四軸都有一個 ESC、無刷馬達、螺旋槳，其中一軸下面有綁用於飛控端的天線。原本是有腳架在下面，但是因爲電池無法撐太久，也就是說重量太重，所以將很多部分優化了一次，其中就有拆掉腳架來節省重量，重量從 900 g 降到 750 g 。

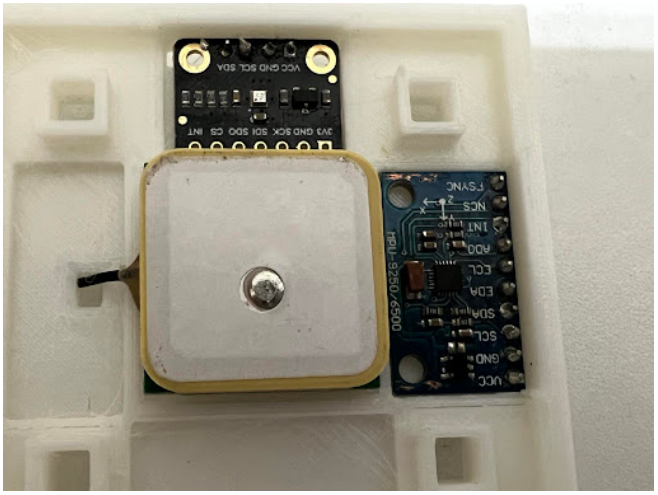
（十）中間爲擺放飛控端的地方，右下方的是 MicroSD 卡模組，左下爲低電壓警報器，中下的爲 ESC，頂端白色部分是使用 3D 列印來製作的，其中 GPS、GPS 天線、9 軸感測器、氣壓計都在這裡面，內部的結構如（十一），會使用 3D 列印的主要一個原因是要固定 9 軸感測器，而且也沒那麼多空間可以擺其他元件，這樣的作法就有位置可以放了，而且也讓無人機看起來整齊一些。另外



圖（十）. 無人機平台下面構造

在這平台氣壓計的頂端有孔洞，這是爲了使空氣能夠流通，上面也貼了一個黑色的紙避免被光線直射。

組裝不是一個難事，但是元件該擺在哪裡就是重要的問題了，現在的樣子是我最滿意的組合，如果我技術再好一點的話，也需機身也可以自己設計。



圖（十一）. 無人機平台內部構造

測試過程

一、無人機部分

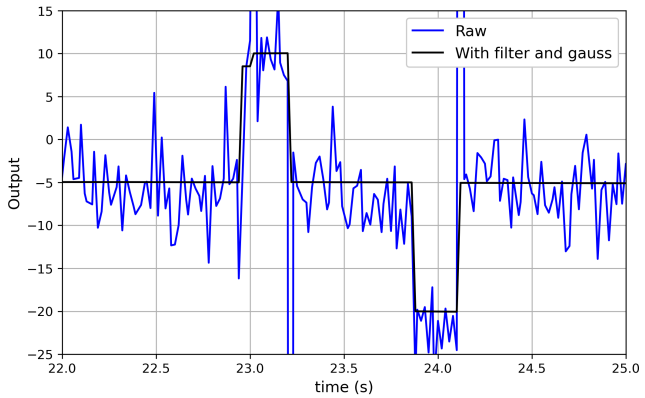
前期如何測試無人機是個問題，因爲我沒有那些專業的工具可以使用，我想到比較好的方式是將一個竿子與地面平行的懸空在距離地面高 10 cm，並且把無人機的底部中間兩端綁在欄杆上，注意不要綁太緊，要能夠使無人機飛離竿子的程度，這樣測試

就不怕會一直摔了。

到了能夠飛起來的程度以後，可以將數據紀錄到 SD 卡上，以下的測試方法都是使用 SD 卡讀取數值。

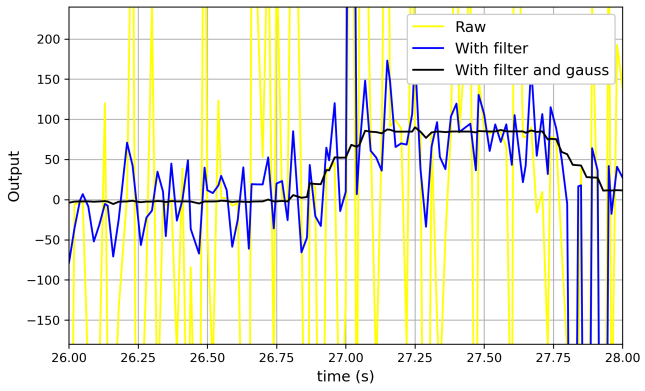
1、2、3 的測試過程都是放在一個靜止的平面上，並且調整目標的數值，看 PID 在有沒有濾波的情況下，Output 的結果與時間作圖的關係，這個無人機四個馬達的總和輸出值平均在 400 左右就可以飛離地面，再來其他的元件就會調整各個馬達的 Output 數值，來達到平衡。

1. **加速度計濾波前後變化**
   圖（十二）在 22.9 秒、23.9 秒時對目標值調整，所以裡當然的輸出值也會變動。



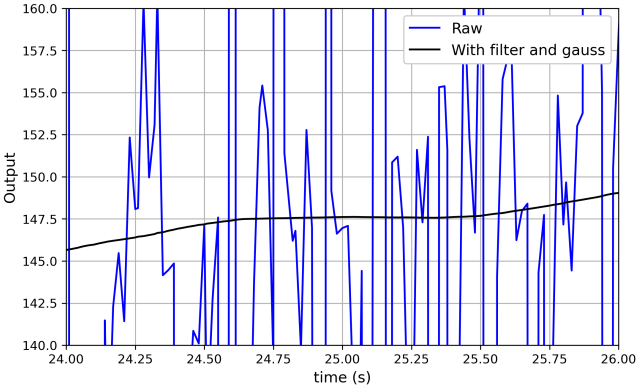
圖（十二）. 加速度計濾波前後變化

2. **磁力計未濾波、濾波、濾波加高斯的變化**
   圖（十三）在 26.8 秒、27.8 秒時對目標值調整。



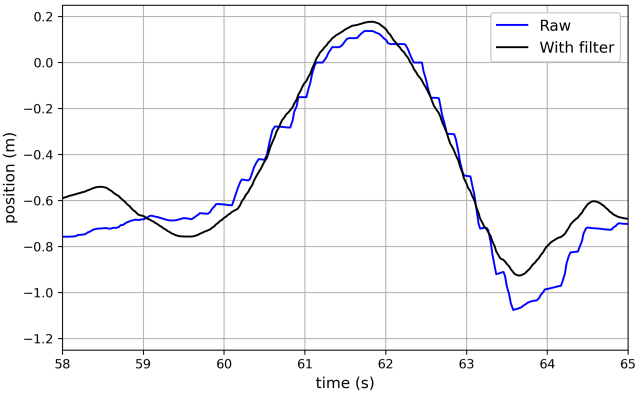
圖（十三）. 磁力計未濾波、濾波、濾波加高斯變化

3. **氣壓計未濾波、濾波加高斯的前後變化**
   圖（十四）並未對目標值調整。



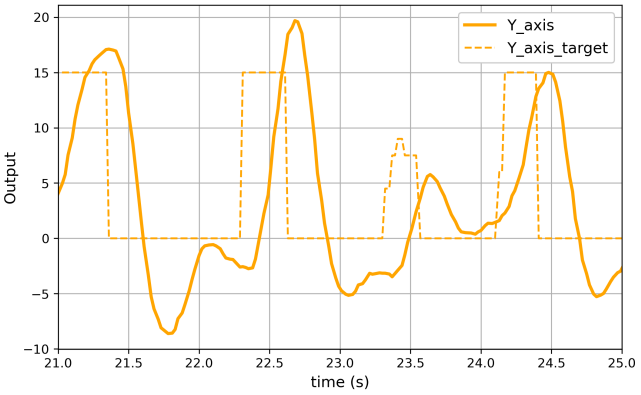
圖（十四）. 氣壓計未濾波、濾波加高斯的前後變化

4. **GPS 濾波變化**
   飛行時的測試結果。



圖（十五）. GPS 未濾波、濾波後的前後變化

5. **元件延遲測試**
   可以看出目標值與輸出值，相差了 0.1 秒左右。

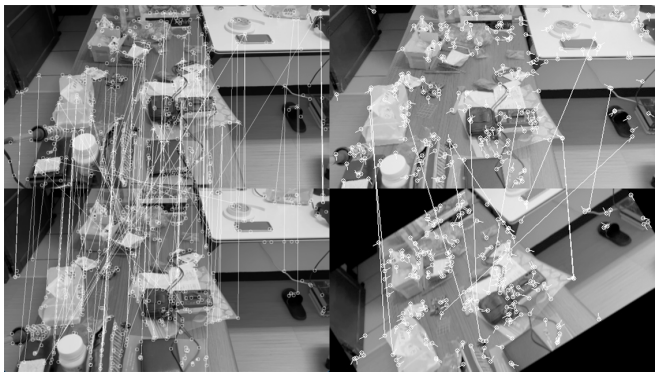


圖（十六）. 元件延遲測試

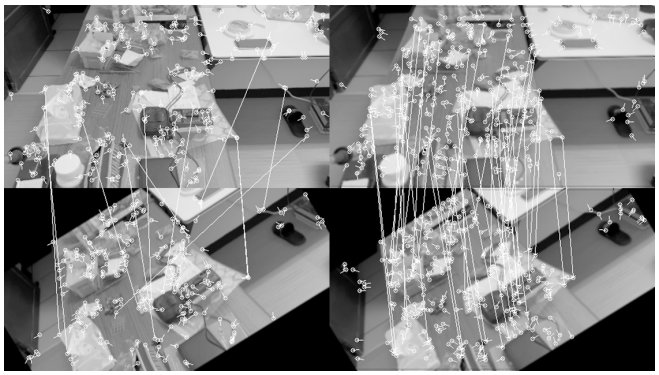
二、電腦視覺部分

因爲兩個原因，無法直接使用 OpenCV 的程式碼，第一是因爲爲了理解與更好的控制 ORB 的運作；第二是發現 OpenCV C++ 的 ORB 算法非常的慢 (每次 0.5s)，所以我重新寫了一個類似 ORB 算法的 C++ 程式，運行的步驟爲 : 對圖像梯度 → 篩選 → 角點分數 → 角點質心 → BRIEF → BRIEF 配對，而 BRIEF 只記錄 64 個描述子。

出來的效果在圖像未旋轉的情況下還算可以，但是只要經過旋轉，就無法精確匹配，由圖（十七）所示，所以後來還是照 OpenCV 的 ORB 去改 (後來發現其是非常的快，之前覺得慢是因爲紀錄到了初始化過程)，結果如圖（十八）所示。



圖（十七）. 配對結果 (左 (位移)，右 (旋轉))



圖（十八）. 旋轉 z 軸 (左 (原本)，右 (修改後))

OpenCV 的過程就如同原理一樣，與之不同的是我的 Fast 檢測並不是去找特徵點四周 3x3 是否有其他特徵點、四元樹篩選，而是將此圖像分爲 30x30 的區塊，並且使區塊內就只能有一個角點分數最大的特徵點，當然要這麼做的原因是因爲全搜無法控制特徵點的數量，會造成後面 BRIEF 配對計算量增大許多；另外一個不同的點是我只記錄 128 個描述子 (原本要 256 個)，以減少 BRIEF 配對計算量。

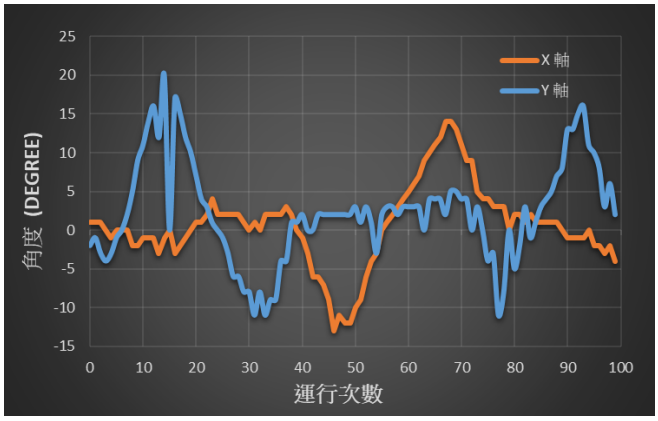
完成 ORB 算法之後，算是完成了一半的工程，另外一半就是要求出單應矩陣與分解出旋轉與位移，ORB-SLAM 的步驟爲 : 特徵點 → SVD → 單應矩陣 → 相機空間還原三維空間 → 分解旋轉與位移。

完成 C++ 程式後，接下來要移植到 ESP32CAM 當中，但是在移植的過程中遇到一個問題，那就是動態記憶體不足的問題，ESP32CAM 可使用的就只有 327680 bytes 的動態記憶體，320x240 圖片佔了 320x240x2 bytes，特徵點總共佔了 30x30x(2x16+4x5)x2 bytes，紀錄配對總共佔了 900x5x4 bytes，總共加起來已經 265200 bytes 了，再加一些其他的就超過了 ESP32CAM 的負荷量，所以後來我就只劃分 26x26 個區塊，並且不直接紀錄兩張圖片特徵點，只記錄前一幀，並且當前幀直接去做運算並取代前一幀的特徵點，爲了避免 BRIEF 配對到相同幀的特徵點，特徵點增加了一個 bool 屬性，並且判斷是相同 bool 才去做 BRIEF 配對，紀錄配對也調整成只記錄 150 個。

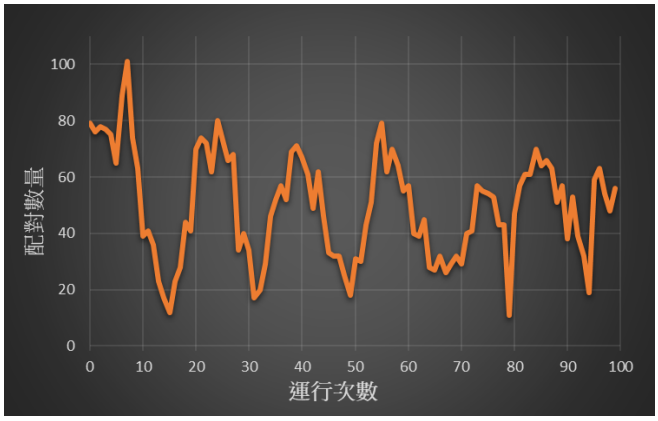
成功移植在 ESP32CAM 後，爲了更好的觀察角度變化，我以第一幀作爲標準，測量後面的 100 幀結果 (正常的使用情況下並不會鎖定一幀，而是會比較前一幀與當前幀的結果)，並且判斷是否成功配對了 16 個特徵點以上，如果成功就計算他在 x、y 軸的旋轉角度，因爲 x、y 軸相較於 垂直 z 軸更難去判別旋轉方向，並且只要單應矩陣有些微的誤差，就會導致旋轉判斷成了移動。

圖（十九）是在雜物較多的地方進行測試的配對數，並以後 100 幀的數據來進行 x、y 做圖，這數據當中有兩幀因爲特徵點不足 16 個所以失敗 (分別在第 15 幀與第 79 幀)，如果失敗，則設定 x、y 都是 0°，配對數量與運行次數關係如同圖（二十），x、y 軸角度與運行次數關係如同圖**??**，此 100 幀總共運行了 35 ~ 40 秒左右，而一秒快的可以執行快 4 次，慢則 2 次，視照片的複雜度 (特徵點數目) 所定。

1. **雜物較多時的測試結果**



圖（十九）. 角度與運行次數關係



圖（二十）. 配對數量與運行次數關係

2. **實際應用在無人機的測試結果**
   可看附錄 (4) 即是實際應用的影片與結果。

結果分析與討論

從圖（十二）到（十五）可以看到，濾波對無人機穩定性的影響非常的大，沒有濾波的話就無法穩定，而我使用的高斯濾波方法對磁力計的效果非常明顯。

經由我的測試後，通常在元件沒有任何延遲的情況下， PID 參數可以調到非常高，越高無人機月可以達到穩定，但是在有延遲的情況下，延遲越大，PID 參數只能越小，所以也會導致操作不靈敏的情況，而這個**無人機的 x、y、z 軸數據延遲達到了 0.1 秒**，發生延遲的問題經由測試後，延遲應該是出在 ESC 元件上，MPU-9250 使用蜂鳴器做聲音延遲測試測出幾乎不會有延遲。

GPS 的濾波數據雖然看起來誤差變大了，但是其實 GPS 也並不是一定準確，會受到大氣的干擾影響，所以重點是數據不會是階梯狀的就好。

由圖可得知確實可以測量 x、y 軸的角度，只要角度越大，配對的數量就會隨之下降，大約在 15° 以內幾乎都可以成功找出旋轉角度。

而剛剛在原理中有講到，單應矩陣無法處裡非平面的場景，而經由我的實測發現確實如此，但是只有在移動上會有問題，我可以正常來測試旋轉是因爲對於旋轉來說，整個世界就是平面的，就如同全景攝像機的一幀一樣。

最後有一個無法解決的問題，那就是移動攝像機的時候影像會有飄移模糊，配對數就會大幅下降，而且在無人機實際應用下受震動的影響會更誇張，但是這也合理，因爲原本使用 GRAYSCALE 模式的幀數就不是很高 (我猜是因爲未編碼，而且數據量大)，一張照片需要 0.1 ~ 0.2 秒的時間來生成，這樣飄移是必然的。

總結

無人機做到現在，也只是到了能飛的程度，雖然有使用高斯濾波來穩定數據，但是因爲 ESC 有延遲的原因，造成在控制上不穩定，在有風的清況下會更嚴重，如果眞的要做出更好、穩定的方式，可能第一步就是要更換元件。

再來電腦視覺系統目前也還無法使用在無人機上，如果只用來傳輸影像的話，使用手機連接的距離也無法太遠，或者可以一樣利用 ESP-NOW 來連接螢幕觀看影像? 總之目前還有很多地方需要改進。

附錄

1. ESC multisim 模擬:
   `https://www.multisim.com/content/4Vkx8FGz53veqVyvDch3GN/bldc-motor-drive/`

2. 所有程式碼:
https://github.com/Wattgo-Real/Collections/tree/main/Drone/MainControl

3. Joop Brokking 教學影片:
https://www.youtube.com/@Joop_Brokking

4. 無人機影像測試影片:
https://youtu.be/XDx7J83GrS4

5. ORB-slam 3 開源原始碼:
https://github.com/UZ-SLAMLab/ORB_SLAM3

6. ORB-slam 原碼解析 ORB 特徵提取（二）:
https://blog.csdn.net/weixin_45947476/article/details/122799429

7. ORB-slam 裡爲什麼要分單應矩陣和本質矩陣陣來分別恢復 R 和 T :
https://www.zhihu.com/question/53891132