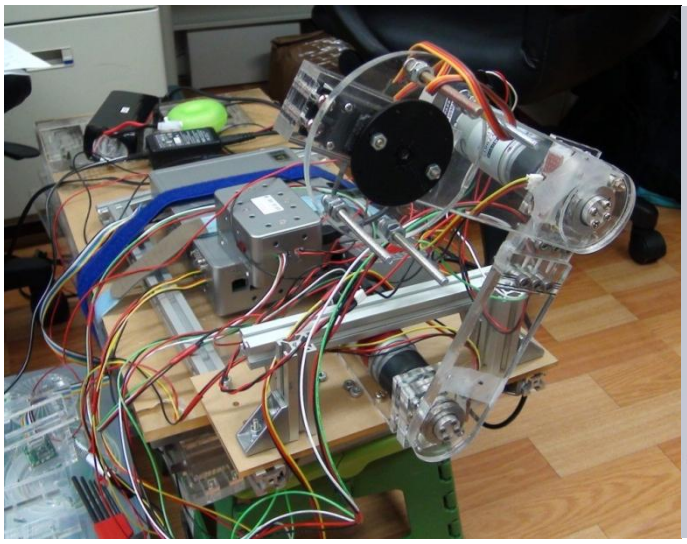
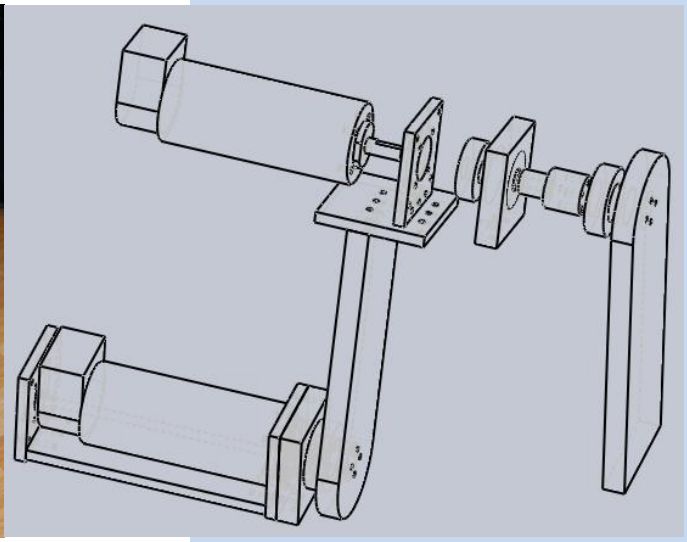
# 機器人簡介

# 期末報告

# Chicken Head Project






## 第五組

黃群凱 r01522805 控制組碩一

周敬凱 r01522801 控制組碩一

謝環宇 r01631003 生機所碩一

# 目錄



# 分工

| 姓名 | 比重 | 工作 |
|---|---|---|
| 黃群凱 r01522805 | 33% | 逆運動學與控制法 MATLAB 模擬 程式撰寫 |
| 周敬凱 r01522801 | 33% | 機電整合 FPGA IMU、CAM 訊號處理 SW 動畫模擬 |
| 謝環宇 r01631003 | 33% | 機構設計、製作與改良(第二版) 影片製作與剪接 |

## 一、Chicken Head 簡介

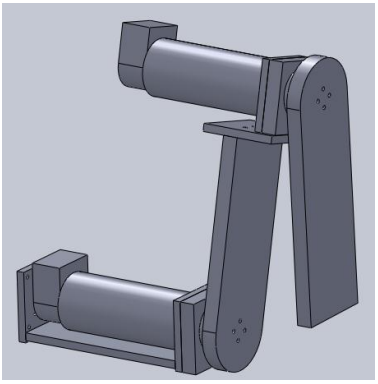
雞有一種現象叫做「Head Stabilizing」，當抓住雞的身體並且上下左右移動時，雞的頭會保持在同一個位置不動，不只雞有這現象，像是鴿子、白鷺鷥、秧雞等鳥類也有這種現象，這種現象來自於這類動物在走路或跑步時「Head-bobbing」的行為。Head-bobbing 是某些鳥類在行走時，頭部會有一前一後擺動的現象，仔細觀察會發現分為「thrust phase」與「hold phase」。在 thrust phase 時，頭部向前伸，而當 hold phase 時，身體再向前移動，此時頭部不會移動，次現象經過觀察，推論其目的有二：其一是提供移動時視覺深度的資訊，其二則是穩定影像，避免在移動時遭受天敵攻擊。

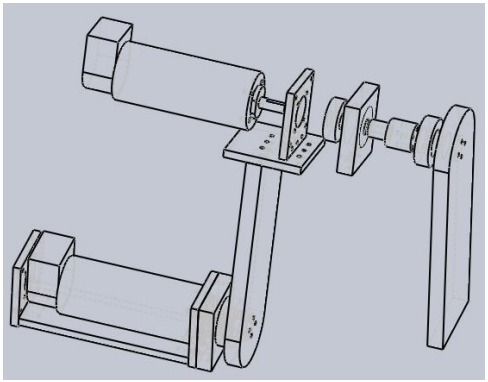
## 二、設計動機與目的

我們的設計靈感就是來自於雞的身體在任意移動甚至轉動的情況下，頭部依然可以保持不動，於是我們想設計出一個機器人，當我們移動或轉動他的基座(Base)時，亦不受到影響，又因為我們在課堂上學習了機械手臂的運動學，想要加以應用，於是乎我們設計了一個簡單的 2R 手臂來模擬雞的脖子，並且在手臂末端裝上相機系統，模擬雞的眼睛，並且在基座上裝設 IMU 感測器得知基座的狀態，試圖透過感測器的資訊配合課堂上所學的機械手臂正、逆運動學的運算，模擬雞的「Head Stabilizing」。

## 三、機構想法與設計

由於我們想要利用上課所學的運動學以及機械的結構去模擬雞頭的運作，因此想利用平行式的機械手臂來模擬雞頭在平面上的移動。而在現有的材料下，以 DC 直流馬達和壓克力等，決定設計兩軸的機械手臂。桿件的長度預期為 15cm，而且第二連桿的末端點位置需要和第一個馬達軸的中心一致。

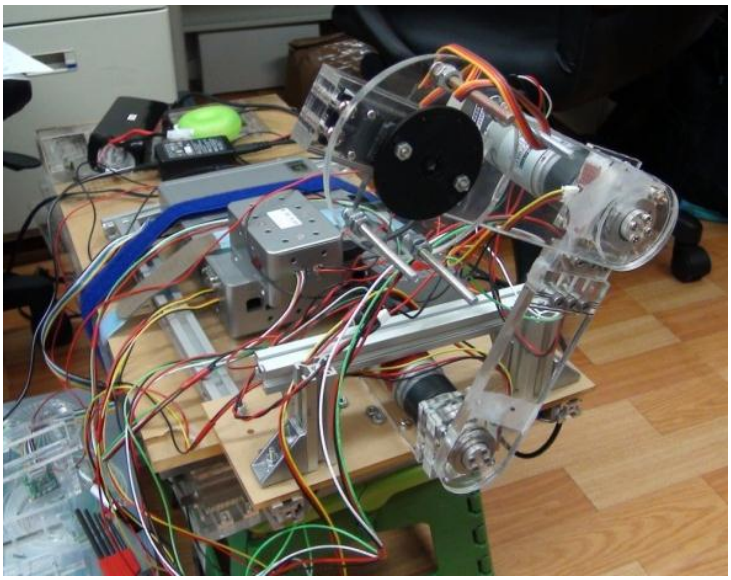




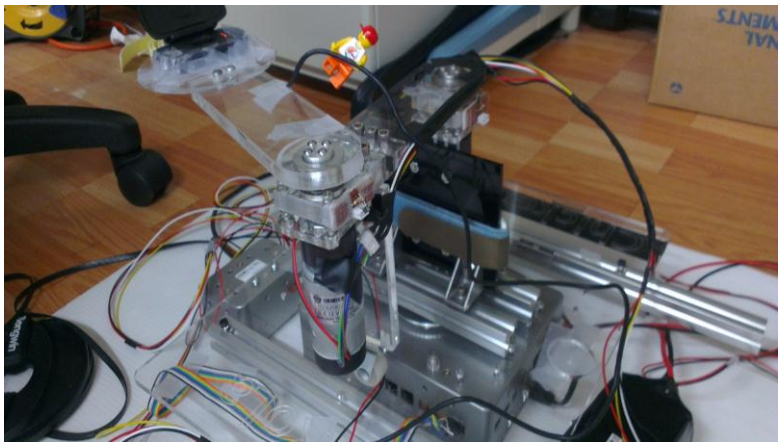
在軸跟桿件的連結上，有裝上滾珠軸承以及止推軸承，可以分散來自於相對於軸的側向和徑向力。

在末端的位置上安裝了攝影機當作雞頭的眼睛

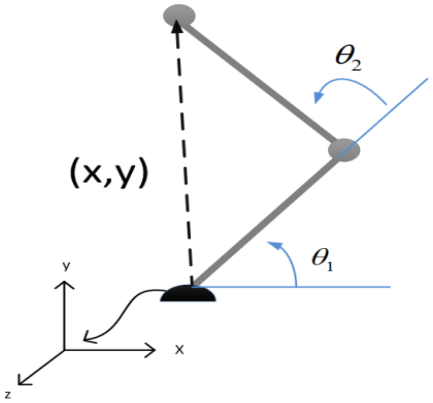
第一版完成的機構



之後在測試過程中發現到，重力的因素常常導致機構不穩定，容易讓軸承部分鬆脫，且馬達需要提供較大的扭力，考量此因素，於是我們將機構改為水平放置，並在第二顆馬達下方加上舵輪，減輕機構的負荷(第二版，如下圖所示)。



## 四、手臂的正、逆運動學




$$FK$$

$$x = l_1\cos(\theta_1) + l_2\cos(\theta_1+\theta_2)$$

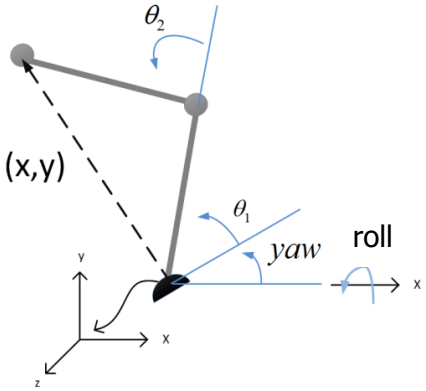
$$y = l_1\sin(\theta_1) + l_2\sin(\theta_1+\theta_2)$$

$$IK$$

$$\theta_1 = \tan^{-1}(\frac{y}{x}) - \tan^{-1}(\frac{l_2\sin\theta_2}{l_1+l_2\cos\theta_2})$$

$$\theta_2 = \cos^{-1}(\frac{x^2+y^2-l_1^{\,2}-l_2^{\,2}}{2l_1l_2})$$

上圖為基本的 2R 手臂的正逆運動學關係式，但我們的手臂因為基座會被轉動，所以公式要稍加改變，若我們加上了基座 roll 與 yaw 方向的轉動後，新的運動學方程式會變為如下圖所示：




$$FK$$

$$x = l_1\cos(\theta_1+yaw) + l_2\cos(\theta_1+yaw+\theta_2)$$

$$y = [l_1\sin(\theta_1+yaw) + l_2\sin(\theta_1+yaw+\theta_2)]\cos(roll)$$

$$IK$$

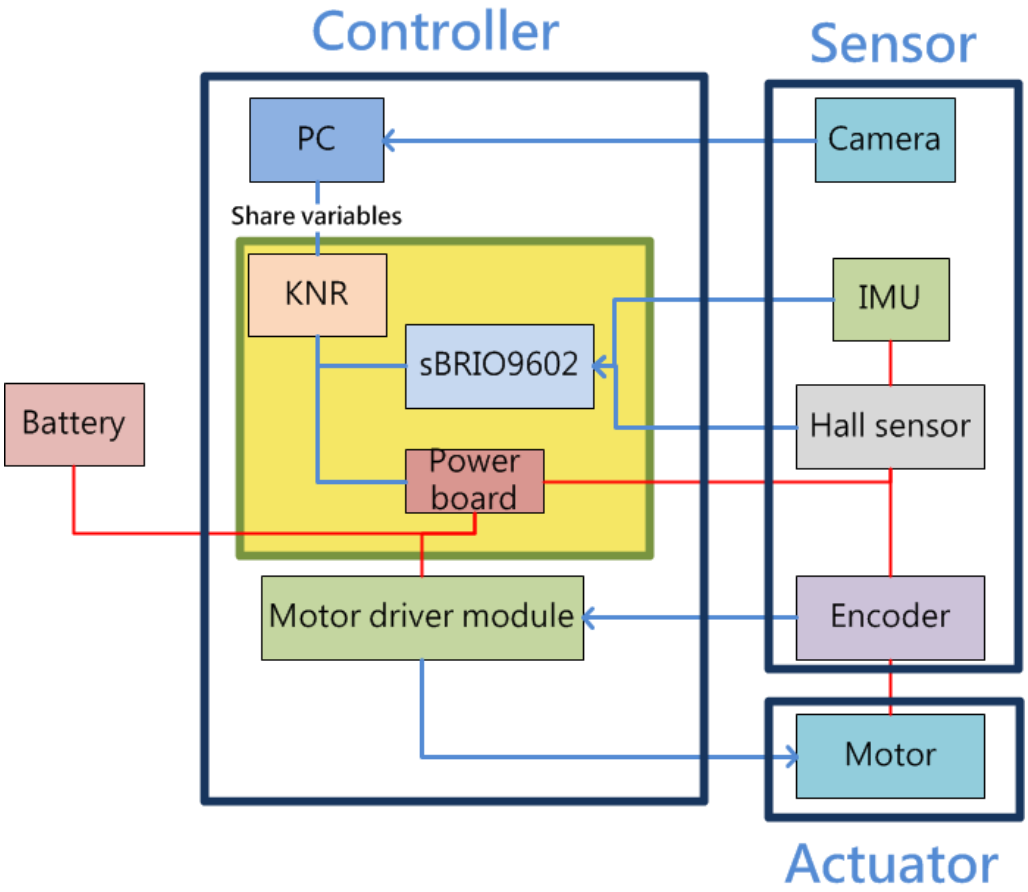
$$\theta_1 = \tan^{-1}(\frac{y}{x\cos(roll)}) - \tan^{-1}(\frac{l_2\sin\theta_2}{l_1+l_2\cos\theta_2}) - yaw$$

$$\theta_2 = \cos^{-1}(\frac{x^2+(\frac{y}{\cos(roll)})^2-l_1^{\,2}-l_2^{\,2}}{2l_1l_2})$$

從新的式子中可發現，yaw 方向的旋轉與$\theta_1$相同，故兩個角度相加即可，而 roll 方向的角度變化，可視為 y 的高度在 x-y 平面上做投影，故是乘上一個 cos(roll)。

## 五、系統架構

### ◯硬體架構示意圖




整體的系統架構如上圖，KNR 為一個控制核心 sbRIO9602 與 Power board 組成。紅色線為電源供應、藍色線為訊號，電池供應 24V 電壓至 Power board，由 Power board 上的 convertor 轉為 5V 供應給各個 sensors 使用。Motor driver module 主要由 polulu VNH3SP30 H bridge 晶片組成，藉由控制 PWM duty cycle 使馬達加減速。

在致動器部分，我們使用祥儀 DC 有刷 IG32-PGM01 減速比 27:1，內建增量型磁式編碼器。

在 Sensor 部分，慣性量測系統(IMU)為實驗室自製的模組，內部晶片使用 ANALOG DEVICE 公司生產的三軸加速規 ADXL335 與一軸陀螺儀 ADXRS620，整顆 IMU 由一個三軸加速規與三個一軸陀螺儀組成，提供六軸資訊以便計算出身體姿態。

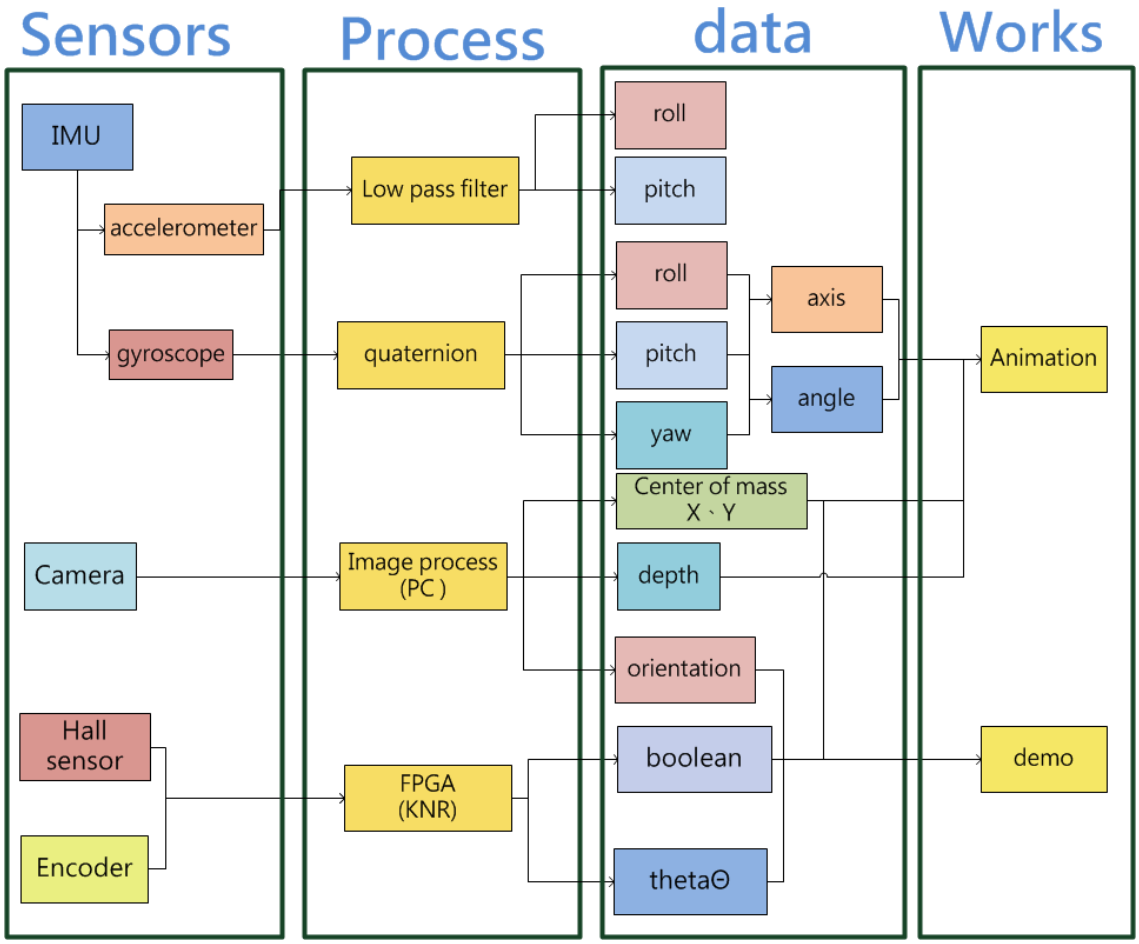
Hall sensor 使用 SS443A UNIPOLAR 由於在 OUTPUT 的類型屬於 sink 因此在訊號腳需並聯一顆 pull up 的電阻。

Encoder 使用祥儀內建的增量型磁式編碼器由 A、B 兩個 channel 組成各有 7 個 poles，由於 A、B 相位相差 90 度在解碼後可得到 4 倍的解析度，在加上 27:1

的減速比，最後一圈可以有$7 \times 4 \times 27 = 756\,{}^{counts}/_{cycle}$，但由於手臂放大齒輪箱背隙在使用上精度仍顯不足。

相機系統使用 Logitech 網路攝影機 C525，最高有1280x720 pixels，並且 fps 可達 30 速度足以提供追蹤目標使用，且可與 NI VISION Assistant 相容。但相機本身並不與 KNR 相容，且相機資訊也不經由 KNR 處理，因此我們將相機系統連接 PC，在 PC 端做影像處理後將控制訊號傳至 KNR，在傳輸過程中透過 Network published shard variable 傳輸，利用此方法可減少 KNR 的運算量，但缺點為機器人並不是全自主的與環境互動。

## ○程式架構

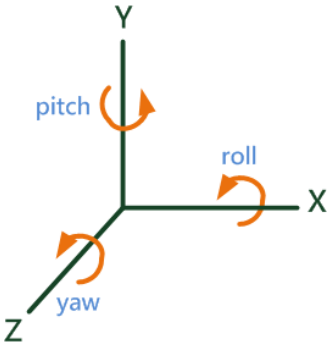


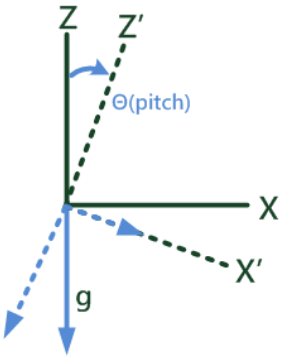

我們藉由 sensor 量測到的原始訊號經過程式運算後得到有用的資訊並將此資訊應用與電腦動畫與實際 demo 上，上圖即處理 sensor 的程式架構圖。

## 六、感測器與後處理

### ○IMU

由於加速規的訊號包含高頻的雜訊，在使用上我們讓他通過 Low pass filter 濾掉高頻雜訊，並藉由加速規自身座標得到重力加速度的分量可得 row 與 pitch 兩軸的角度資訊，且在身體座標不移動的狀況下，角度資訊為精準不發散的。




三軸陀螺儀可量測 roll、pitch、yaw 的角速度資訊，藉由積分我們可以得知機器人座標相對於世界座標的角度，但由於我們並不曉得在某一時刻 IMU 是先對 Z 軸旋轉還是先對 Y 軸旋轉，我們只知道在某一時刻的三軸角速度為多少，在無法使用 X-Y-Z fixed angle 旋轉矩陣來積分下，我們使用四元數法(quaternion)達到積分的目的。

四元數法為一種運算法則，四元數由一個實部與三個虛部組成

$$q = a + bi + cj + dk$$

其中 a、b、c、d 為實部，在計算上類似於右手座標系的概念

$$i^2 = -1\ j^2 = -1\ k^2 = -1\ ij = k\ ji = -k\ jk = i\ kj = -i\ ki = j\ ik = -j$$

也由於以上的關係，quaternion 在計算上並沒有交換率。

對於在三維空間的旋轉，可將 quaternion 描述成對一個單位向量(a, b, c)旋轉一個角度 $\theta$ 。

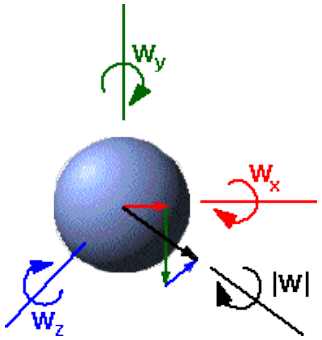
$$q = \cos\left(\frac{\theta}{2}\right) + a\sin\left(\frac{\theta}{2}\right)i + b\sin\left(\frac{\theta}{2}\right)j + c\sin\left(\frac{\theta}{2}\right)k$$

其中 $\|q\| = 1$ 藉由在某一時刻得知三軸角速度 $\omega_x$ 、$\omega_y$ 、$\omega_z$ 我們可將旋轉角度 $\theta$ 視為

$$\theta = |w(t)| * dt$$

其中 $|w(t)| = \sqrt{{w_x}^2 + {w_y}^2 + {w_z}^2}$ ，而單位向量(a, b, c)為

$$(a、b、c) = (w_x\hat{\imath} + w_y\hat{\jmath} + w_z\hat{k})/|w(t)|$$



因此整個 quaternion 可改寫成

$$q = \cos\left(\frac{|w(t)|dt}{2}\right) + \left(w_x\hat{\imath} + w_y\hat{\jmath} + w_z\hat{k}\right)/|w(t)|\sin\left(\frac{|w(t)|dt}{2}\right)$$

我們可將 quaternion 視為一個旋轉矩陣，更新旋轉矩陣的方式即是將新的旋轉$q'$

後乘於q，由於 IMU 測得的角速度資訊是建立在身體座標之下，所以我們將$q_{initial} = 1 + 0\mathrm{i} + 0\mathrm{j} + 0\mathrm{k}$設定成一開始世界座標與身體座標重合，在下一個時刻量測得知${}^{w}_{b}q$ 在下一個時刻量測得知${}^{b}_{b'}q$因此整個積分變為以下式子

$$q_{now} = q_{initial} \times {}^{w}_{b}q \times {}^{b}_{b'}q \times {}^{b'}_{b''}q \ldots$$

Quaternion 在乘法上可利用前述的關係式得到以下式子

$\mathrm{p} = p_0 + p_1 i + p_2 j + p_3 k$ $\mathrm{q} = q_0 + q_1 i + q_2 j + q_3 k$

$$\mathrm{p} \times \mathrm{q} = \begin{bmatrix} p_0 & -p_1 & -p_2 & -p_3 \\ p_1 & p_0 & -p_3 & p_2 \\ p_2 & p_3 & p_0 & -p_1 \\ p_3 & -p_2 & p_1 & p_0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

因此更新矩陣為

$$q_{new} = q_{old} \times q = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} \cos\left(\frac{|w(t)|dt}{2}\right) \\ \sin\left(\frac{|w(t)|dt}{2}\right) w_x/|w(t)| \\ \sin\left(\frac{|w(t)|dt}{2}\right) w_y/|w(t)| \\ \sin\left(\frac{|w(t)|dt}{2}\right) w_z/|w(t)| \end{bmatrix}$$

在得到${}^{w}_{b}q_{now}$後我們經由前述的關係將其轉換為三維空間的單位向量與旋轉量，即可供後端程式應用。

## ○ Camera

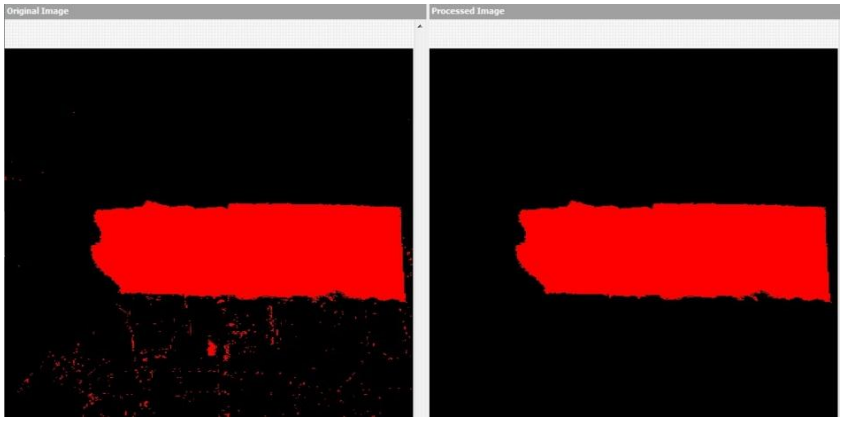
在影像處理部分，首先先將目標物在不同環境中照出 20 張照片，利用這些照片建立影像處理的邏輯，再利用 NI VISION assiatant 製作成 vi 實現 Real time 的控制。

1. 首先我們將影像從 RGB 轉換成 HSV 的色彩空間，HSV 分別為色相（H）、飽和度（S）、明度（V），其中色相即色彩的基本屬性，如黃色藍色等…飽和度指色彩的

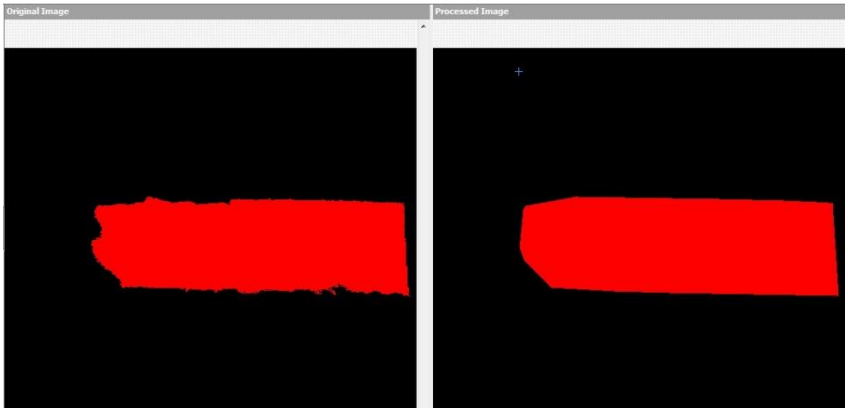
純度，越高色彩越純，低則逐漸變灰，明度即亮度。由於參數轉換的關係，HSV 將亮度、色相區分兩個獨立的參數將可以有效的抵抗環境光線的變化，對於標定物體的能力將會增強。

2. 將照片對其 HSV 的閥值空間做調整。

3. 在大致得知影像的位置我們再進一步使用 remove small object 將辨識錯誤的小物體濾掉。



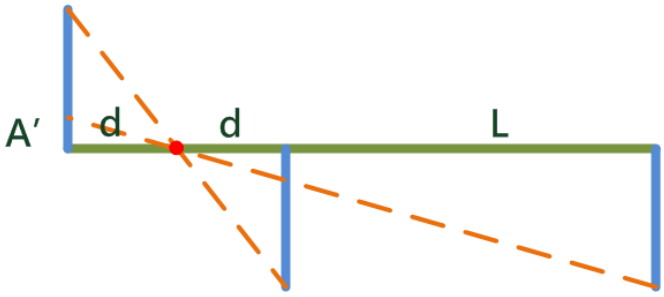

4. 最後使用 convex hull 將物體平滑化。




經由電腦計算我們可以得知物體的重心位置與面積，也可藉由長邊短邊的位置得到物體的角度(0~180)，由於 pixel 對應世界座標下的位移是隨著物體的位置而改變，因此我們要得到深度的資訊才能做有效的換算，由於只使用一台相機，為單眼視覺，我們必須確定目標物與相機的平面是互相平行的，藉由量測出目標物在距離相機 d 時佔據整個視野，利 pixel 面積的變化，我們可以藉由簡單的數

學關係得到正確的深度資訊。

$$Depth\ \frac{A'}{A} = \left(\frac{d}{d+L}\right)^2$$



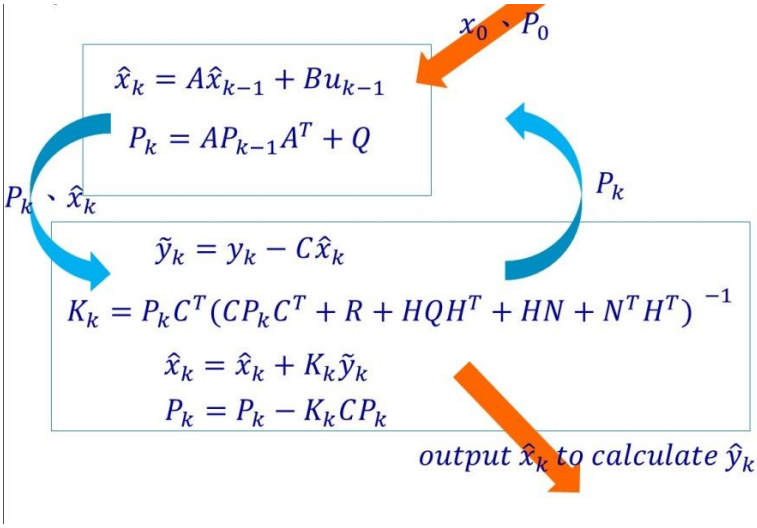

最後再利用深度資訊得到 pixel to millimeter 的正確轉換式。

## ○ Kalman filter

由於陀螺儀計算姿態的方法為積分角速度，而陀螺儀本身有 gyro bias 與 gyro drift 且陀螺儀對於溫度變化敏感，再經過長時間積分會累積誤差，因此必須有標準值可提供校正，而加速規中的 roll 與 pitch 在加速度沒有太大變化時將可校正陀螺儀的積分資訊，但 yaw 方向由於加速規軸與重力方向重合，因此我們無法校正 yaw 方向的值。在融合訊號上藉由角度與角速度兩者在物理上的相關性，我們可以使用 Kalman filter 融合兩個資訊。Kalman filter 將系統設定為：

$$\dot{x} = Ax + Bu + Gw$$
$$y = Cx + Du + Hw + v$$

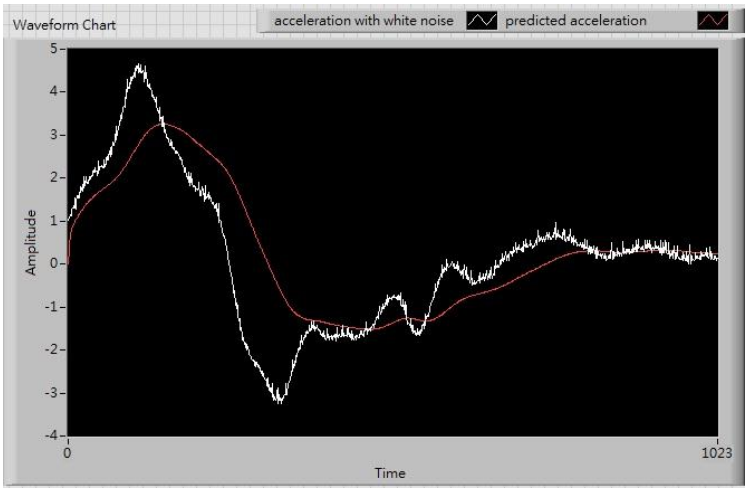
其中 w 為 process noise vector，v 為 measurement noise vector，G、H 表示 w 進入 model 的方向。實際運算上由於我們不知道 w 與 v，我們必須假設 w 與 v 的 covariance matrix Q、R，並藉由最小化 P(誤差相關矩陣)計算出來的 Kalman gain 得知 model 與 measurement 之間的權重。其數學如下



但實際使用上由於 Kalman filter Real Time 的計算對於控制器來說負荷太大，導致程式無法正常運作，且我們只能校正兩個角度的資訊，yaw 方向仍會隨時間飄移，因此最後我們並沒有使用 Kalman filter。

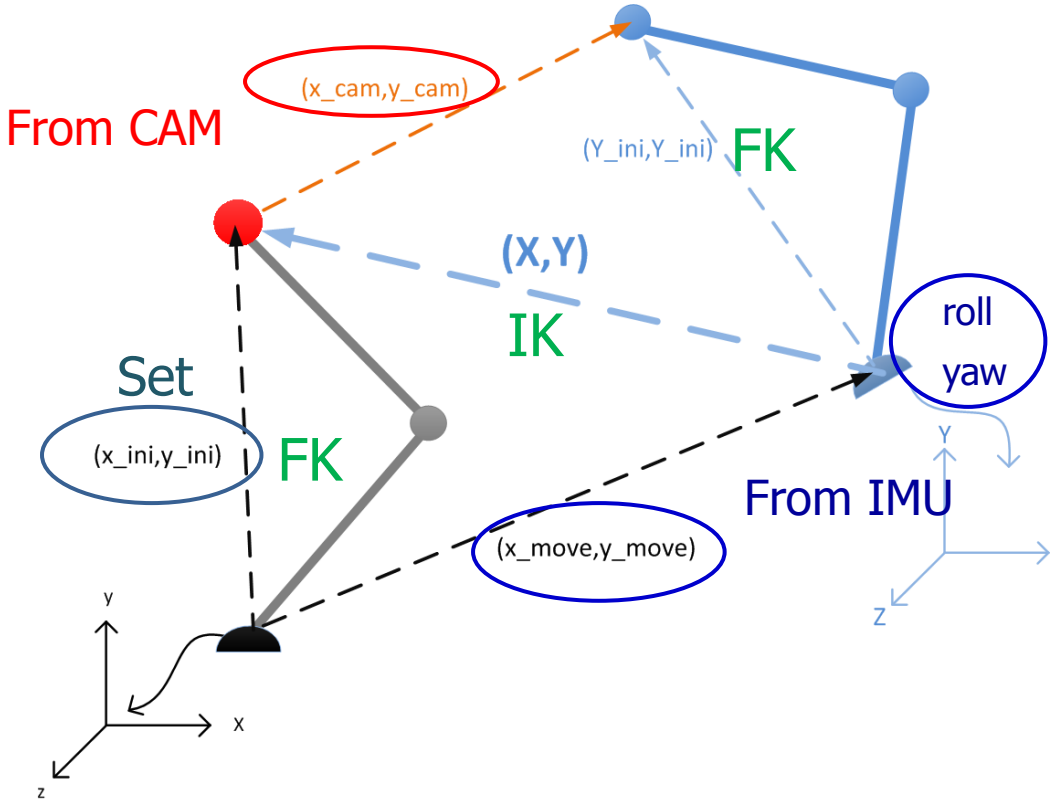
在加速度積分上由於我們不是車輛的系統有 encoder 可以校正加速規兩次積分後的位移，雖然導入 Kalman filter 可以濾掉 measurement 的雜訊，但由於沒有校正值可以使用，因此最後我們也放棄使用加速規的資訊。

不過我們仍做了實驗驗證 Kalman filter 濾波的功能(如下圖所示)，本次實驗我們將 IMU 對一軸移動 18cm 停止，紀錄其加速度值，白線為原始的資料，紅線為經過 Kalman filter 後的資訊，可以看出紅線有明顯的濾波效果，而經過濾波後的資訊經過積分後得到的位移與實際值在 1 秒內的誤差在 1cm 以內，但經過 10 秒後位移與實際值差距達 50cm 以上，因此我們無法使用此資訊。



## 七、控制方法與 LabView 程式

### ○ 控制方法示意圖

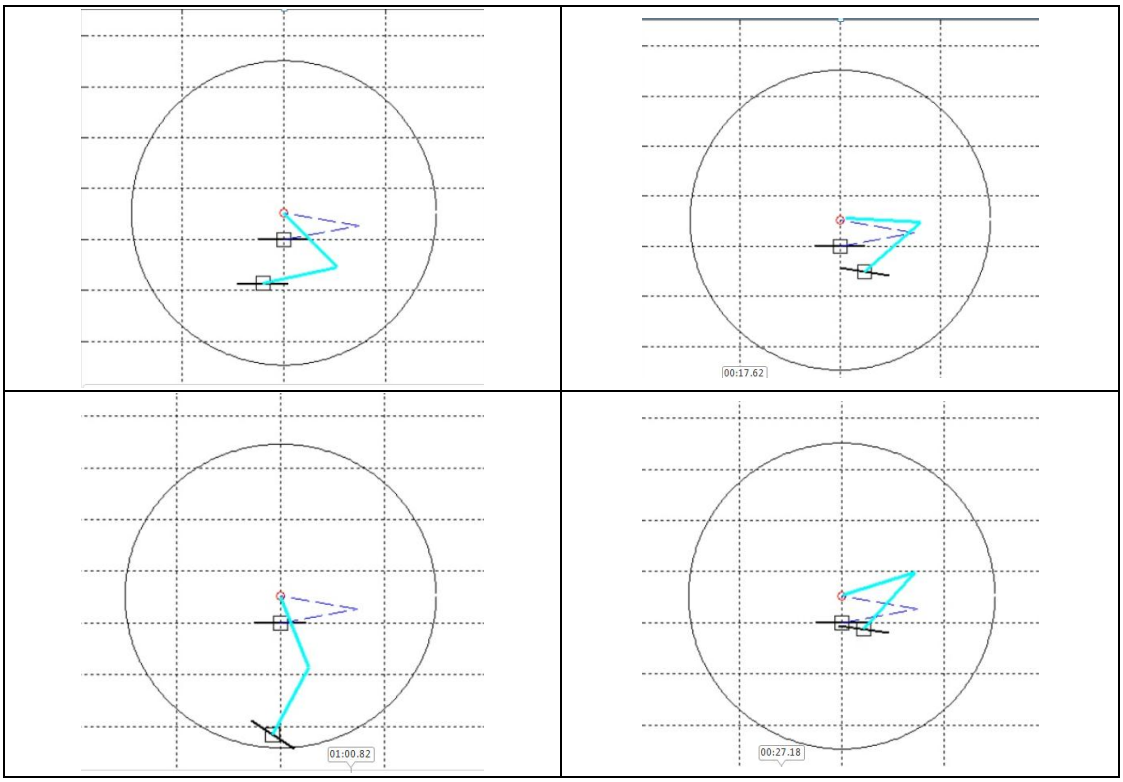



由上圖可見，灰色手臂為初始位置，透過 encoder 的角度資訊，經由 FK 可算出(x_ini, y_ini)，用此向量表示紅點位置，當手臂的 base 被移動(x_move, y_move)，再加上轉動了 roll 及 yaw 的角度後，變為藍色的手臂，新位置用心的坐標系(X, Y)表示，則新位置亦可用 encoder 的角度資訊加上 IMU 的角度資訊，透過 FK 可得知(X_ini, Y_ini)向量。(x_move, y_move)可由 IMU 對加速度積分得到(最後因積分誤差所以沒有使用)，(x_cam, y_cam)則可透過相機資訊做簡單的座標轉換獲得。

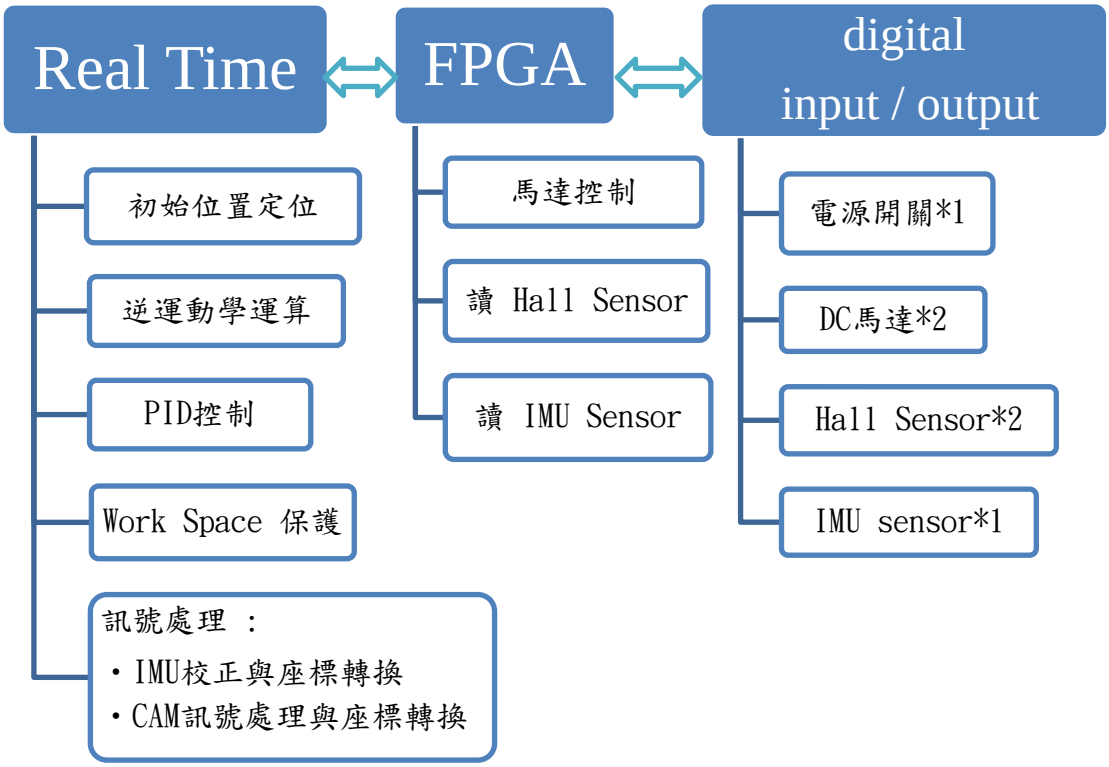
由上述的所有資訊，可以發現到，我們可以由兩個方向得到所要的目標位置(追紅點)向量(X, Y)：

$$(X,Y)=(X\_ini,Y\_ini)-(x\_cam,y\_cam)$$
$$(X,Y)=-(x\_move,y\_move)+(x\_ini,y\_ini)$$

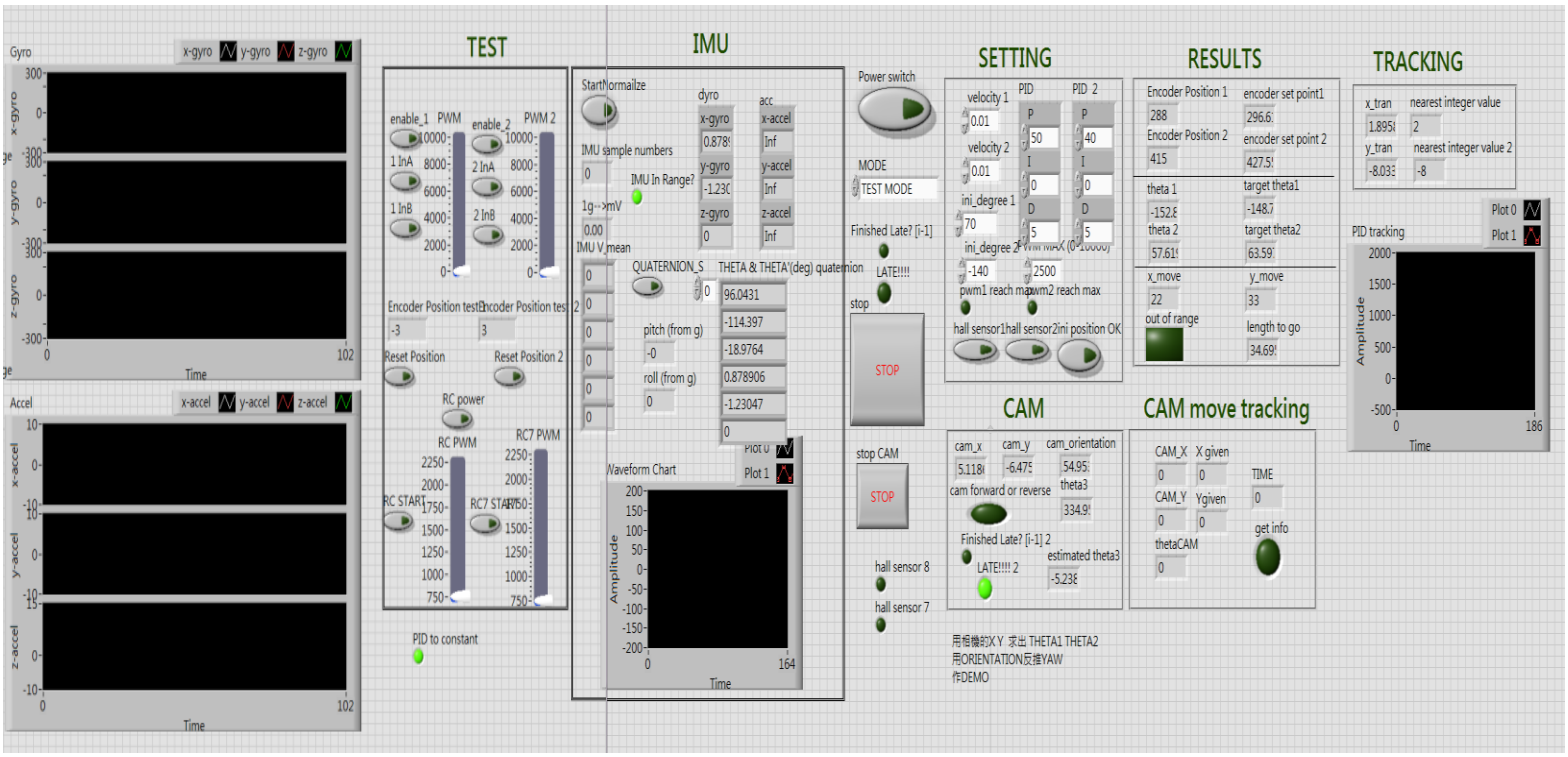
再將(X, Y)帶入 IK 做運算，即可得到手臂所需旋轉的$\theta_1$與$\theta_2$，而得到的新位置就是下一刻的初始位置，不斷的做連續的運算，即可模擬出我們所需要手臂追紅點的目的，我們可以用 MATLAB 做模擬驗證此方法的正確性與可行性(下圖為連續模擬之截圖，程式碼見附錄)。



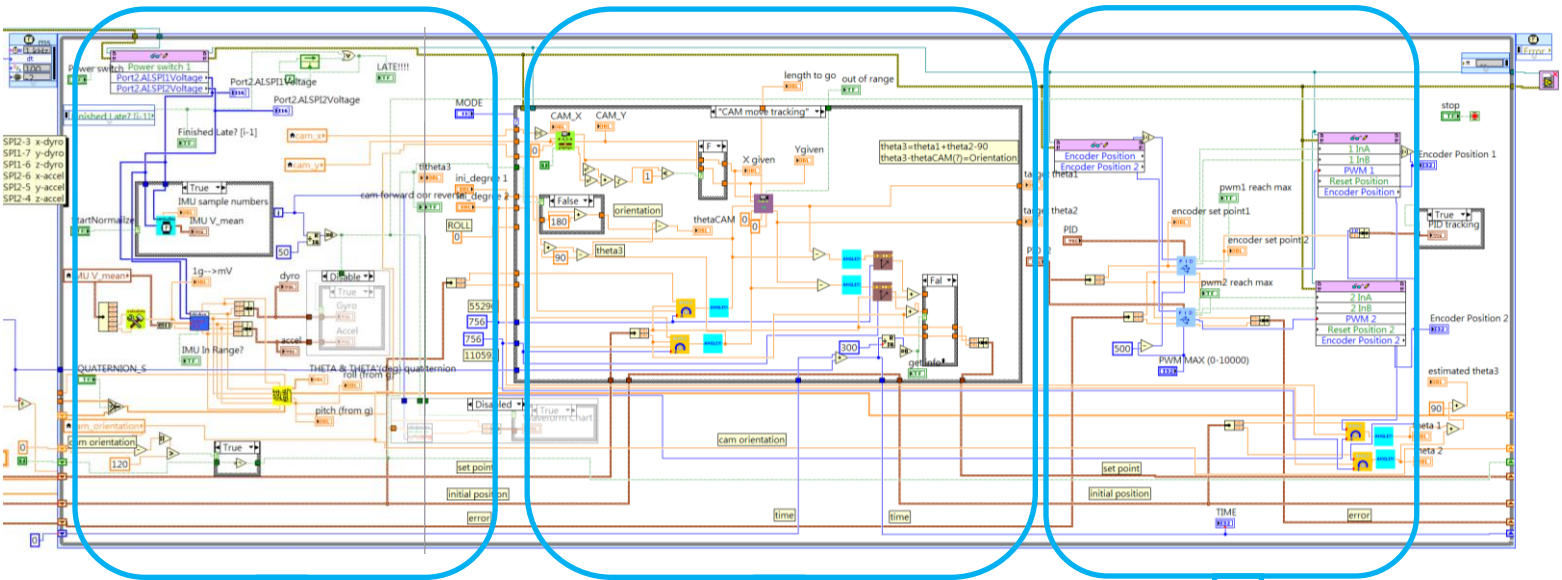

○ LabVIEW 程式架構圖



＊主程式 Front Panel



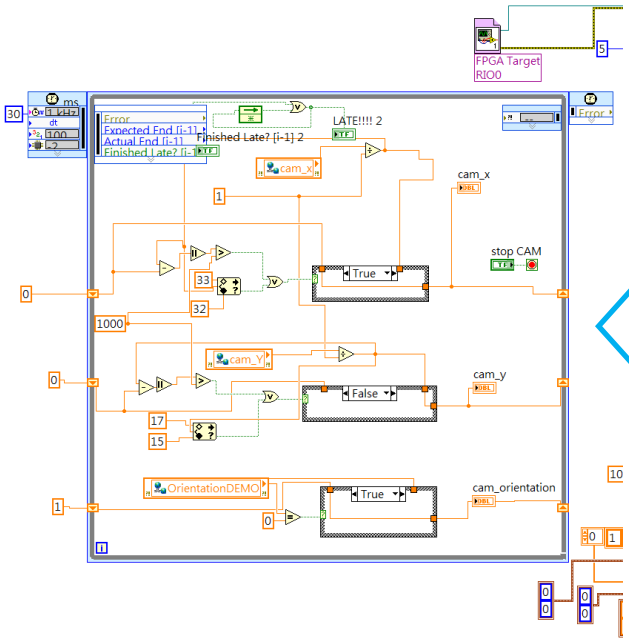

＊主程式 Block Daigram



1. CAM 訊號讀取
2. IMU 訊號校正與處理

主功能視窗：
手臂的 Inverse Kinematics 運算

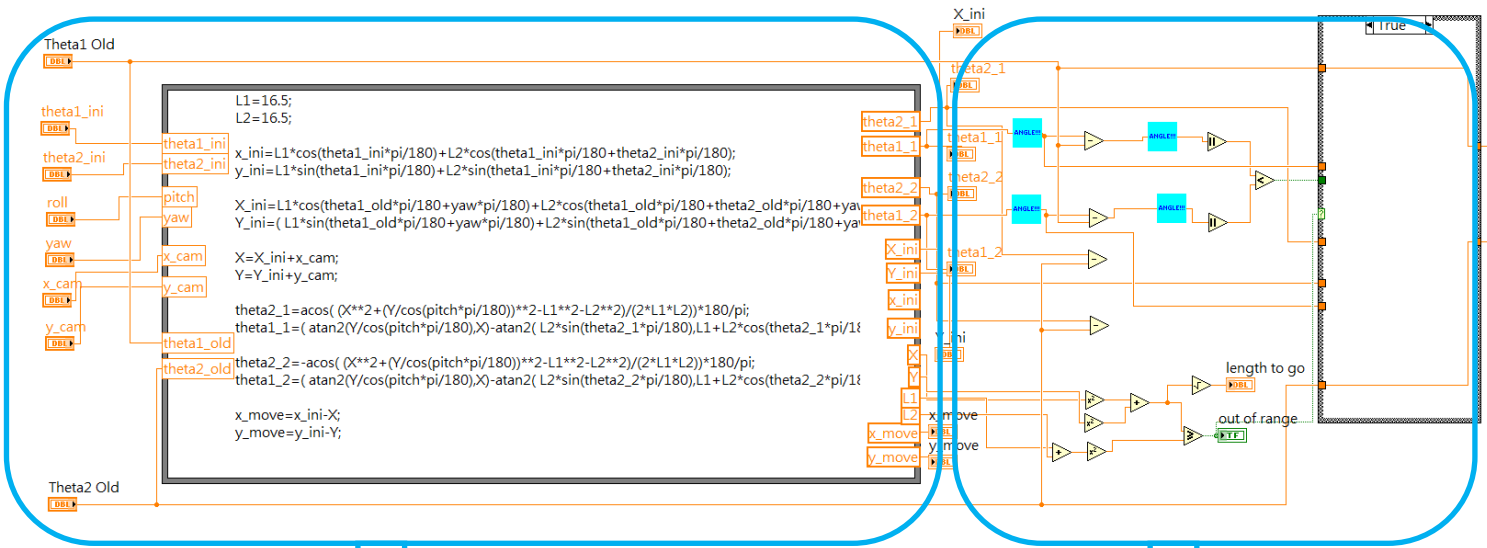
PID control




CAM 訊號處理：

將電腦端得到的相機資訊透過 Network published shard variable 傳輸進入控制板，此處將訊號做簡單的座標轉換，使其與機器人座標一致，並且因為相機有時候會沒有讀到訊號，而造成運算上的錯誤，所以此處有做處理，將沒讀到訊號時做保護，方法是當發現相機沒讀到紅點訊號時，則輸入上一個讀到的值。

＊主功能視窗：逆運動學 Block Daigram

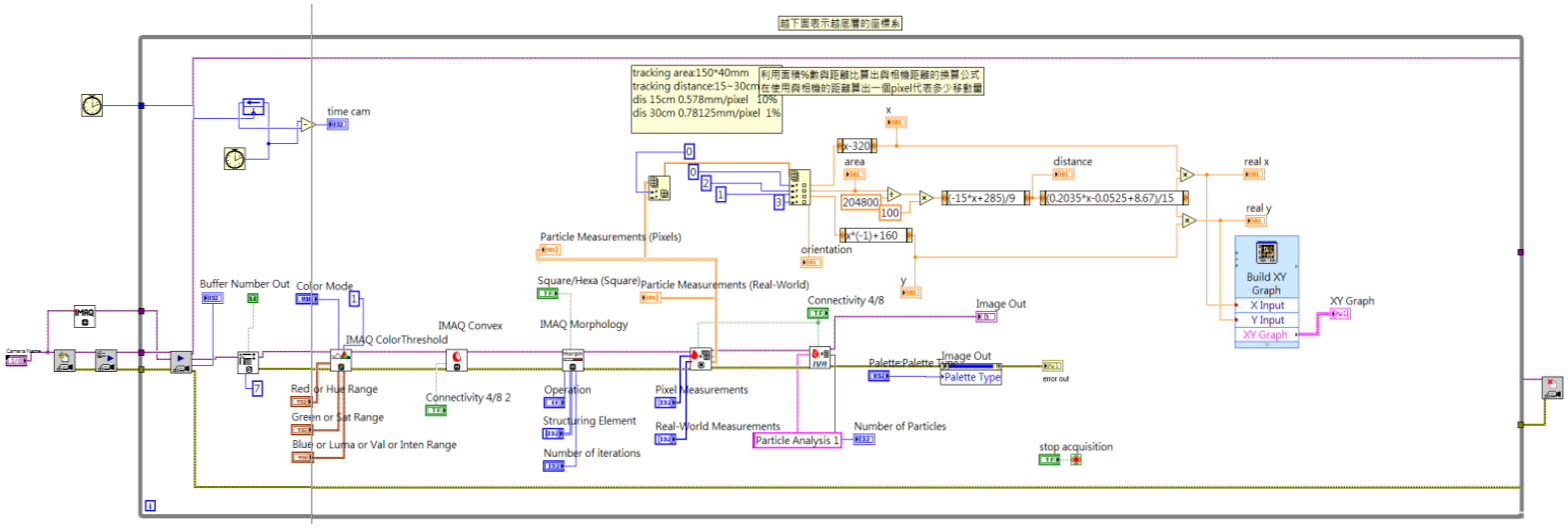



Inverse Kinematics

後處理

1. 角度最佳化，讓手臂不會旋轉不必要的角度（比如說命令為正轉 340 度，則最佳化為反轉 20 度）。
2. 從兩組 IK 解中，選擇出要快達到的一組。

＊電腦端相機資訊讀取、處理與傳輸




## 八、遭遇的困難

### ○訊號與電源干擾問題

由於廠商提供的 KNR 在製作 Power Board 時沒有將訊號電源與馬達電源確實隔開，在馬達轉動時產生的雜訊會導致訊號的地浮動，而 IMU 輸出類比訊號進入 ADC 此時 ADC 地位準也隨之浮動，導致量測的訊號誤差極大，而 KNR 已經包成完整的模組，我們無法拆開使訊號電源獨立供電，也因為這個問題，最後無法將 IMU 與馬達整合在同一個系統上。

### ○Encoder 安裝不易

由於祥儀馬達所附的 Encoder 精度嚴重不足，我們使用企誠編碼器模組 HS302 解析度為 512CPR，經減速比與 A、B 訊號的相位關係可達一圈 55296counts，但由於編碼器光盤孔與馬達軸大小不合，必須額外磨軸加工，但加工導致光盤偏心且光盤平面與軸不平行，且須將讀取頭固定在馬達上，但馬達後端不易固定，當馬達有震動時編碼器讀到的 counts 數會改變，導致 tracking 時的錯誤，光盤安裝需更精密的加工，未來會考慮委外加工，或者購買安裝解系度夠大的馬達模組。

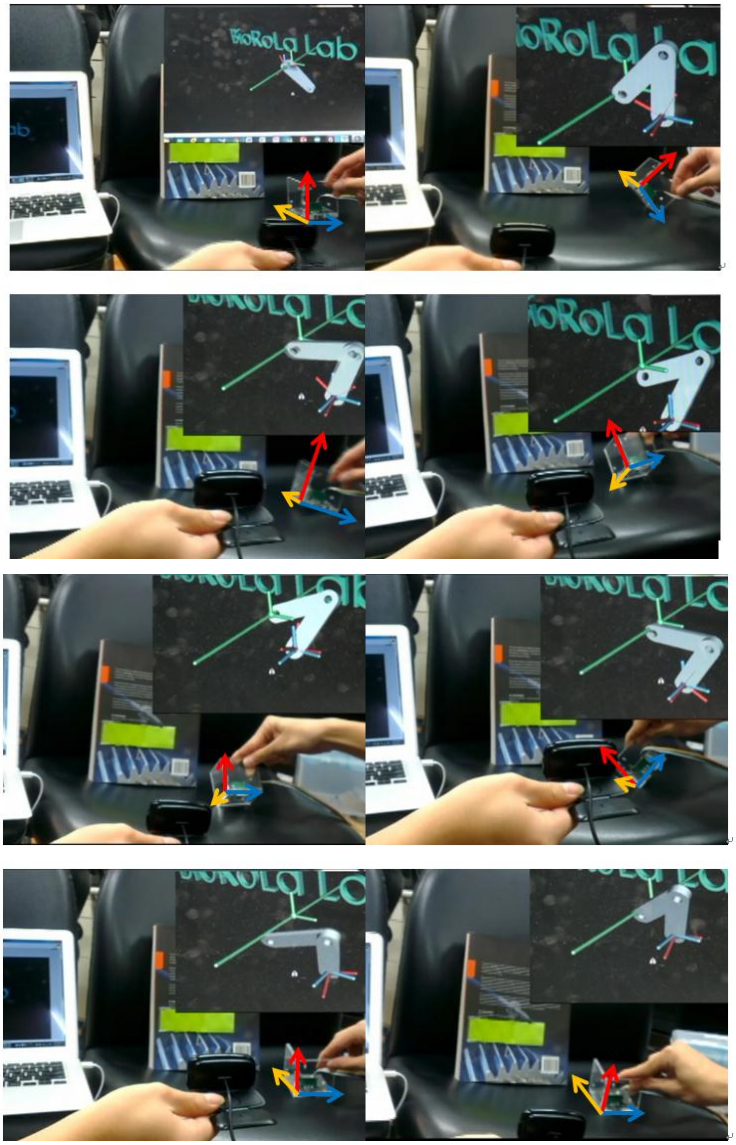
### ○相機資訊問題

相較於 encoder 資訊，相機資訊更新相對很慢，加上是從電腦端讀取再經由 shard variable 讀進程式中會有 delay，造成相機資訊與實際手臂狀態無法同步，造成控制方面的困擾。

## 九、結果呈現

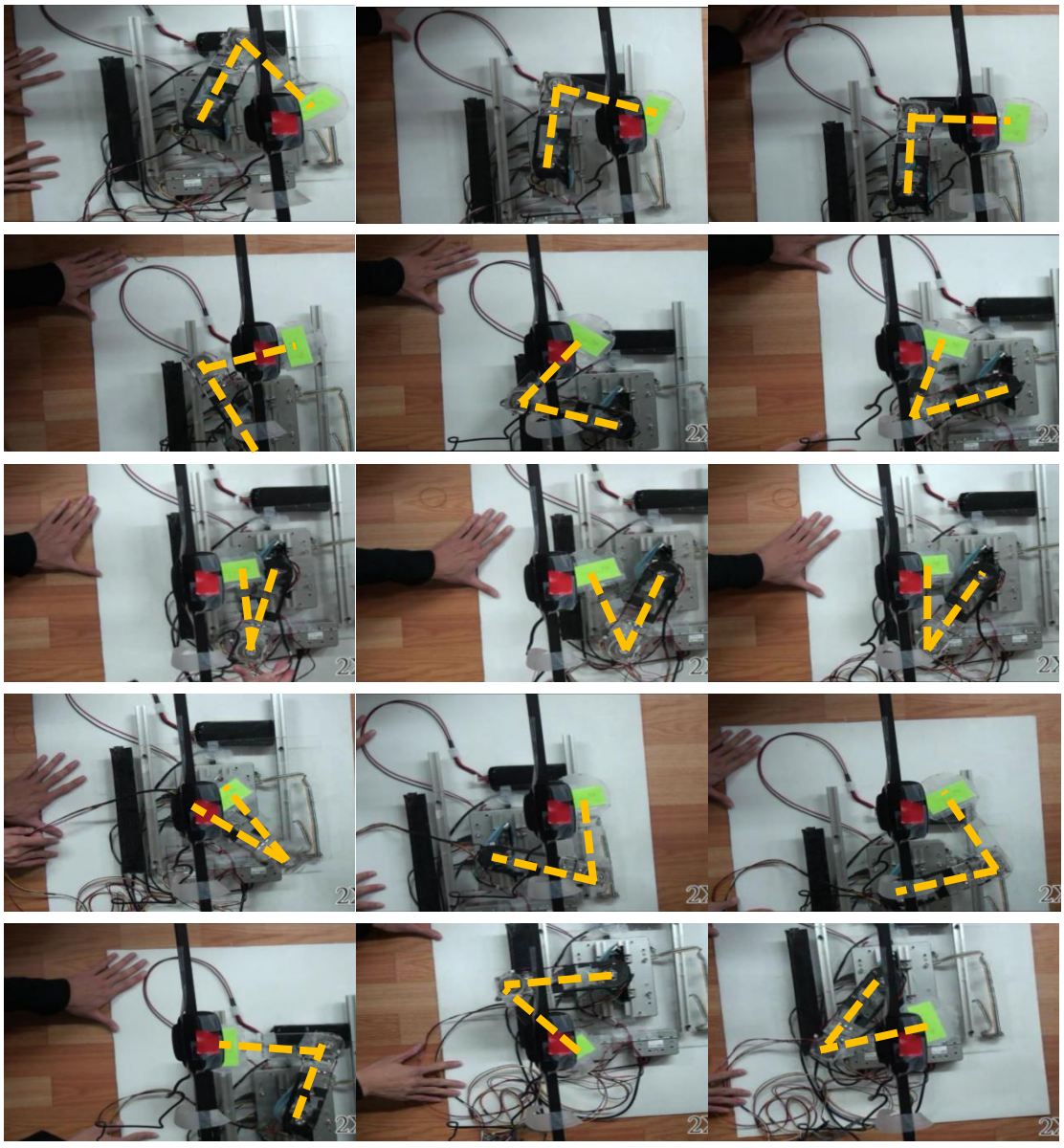
### ◯ Animation

我們將 Solidworks 繪製完成的 Prt 檔轉為向量型式 VDML 檔匯入 Labview 中利用 add object 將一個座標系建立在另一個座標系上，如此我們可以輸入相對的旋轉量不用再對每軸做座標轉換得到正確的動畫，在動畫模擬中我們主要想驗證相機深度資訊(Depth)與移動資訊(Center of Mass X、Y)與手臂 Inverse kinematic 在加上 IMU 資訊後的正確性。因此我們導入相機資訊(當作機身 base 的移動量，與 IMU row、pitch 資訊計算 Inverse kinematic 得出控制訊號$\theta_1$、$\theta_2$丟回動畫程式，並利用 IMU 的 quaternion 轉換成 axis-angle 的資訊旋轉機身姿態，與相機深度資訊以模擬真實世界機身在空間中的位置、姿態與手臂旋轉量。由實際模擬看出手臂末端點的確可以追蹤一個物體，且身體姿態轉動與相機的移動皆於實際狀態穩和，因此可以進一步的進行實際 demo 的動作(下圖為影片截圖)。



## ◯ 實際 Demo（此部分成果是在口頭報告完後才做出來）

在最後的呈現中，我們將相機架設於上方固定位置，此配置是因為相機訊號讀取太慢加上會 delay，此法可以讓相機不須經過座標轉換，控制上較不會出差錯，而將綠點至於雞頭上。在程式端方面，用簡單的策略希望讓相機的資訊可以盡量和手臂姿態的資訊同步，如此一來就可以達到 Tracking 的效果(下圖為影片截圖)。



## 十、參考文獻

[1] Chicken Head Stabilizing
http://a-chien.blogspot.tw/2009/10/chicken-head-stabilizing.html
[2] RESEARCH ARTICLE : Vision during head bobbing: are pigeons capable of shape discrimination during the thrust phase?
作者:Laura Jiménez Ortega · Katrin Stoppa ·Onur Güntürkün · Nikolaus F. Troje
[3] EuclideanSpace - building a 3D world
[4] RIOBotics - Putting the RIO in Robotics

## 十一、附錄

○ MATLAB 模擬程式程式碼:

```
%aviobj = avifile('simulation.avi');
runtimes=100;
n=1;
% base move
x_move=0;
y_move=0;
% base incline
pitch=0;
yaw=0;
% arm length
L1=15;
L2=15;
% ini condition
theta1_ini=10;
theta2_ini=160;
x_ini=L1*cosd(theta1_ini)+L2*cosd(theta1_ini+theta2_ini);
y_ini=L1*sind(theta1_ini)+L2*sind(theta1_ini+theta2_ini);
while n<=runtimes
clc
n
%x_move
%y_move
pitch
yaw
% chicken head see
```

```
x=x_move+L1*cosd(theta1_ini+yaw)+L2*cosd(theta1_ini+theta2_ini+yaw);
y=y_move+( L1*sind(theta1_ini+yaw)+L2*sind(theta1_ini+theta2_ini+yaw) )*cosd
(pitch);
% CAM
x_cam=x-x_ini;
y_cam=y-y_ini;
% new coordinate
X_ini=L1*cosd(theta1_ini+yaw)+L2*cosd(theta1_ini+theta2_ini+yaw);
Y_ini=( L1*sind(theta1_ini+yaw)+L2*sind(theta1_ini+theta2_ini+yaw) )*cosd(pi
tch);
% target
X=X_ini-x_cam; %=x_ini-x_move;
Y=Y_ini-y_cam; %=y_ini-y_move;
% IK calculate
% theta2=acosd( (X^2+(Y/cosd(pitch))^2-L1^2-L2^2) /(2*L1*L2) );
% theta1=( atan2(Y/cosd(pitch),X) - atan2( L2*sind(theta2) ,
L1+L2*cosd(theta2) ) )*180/pi-yaw;
%---------------theta optimization----------------------
theta2_1=acosd( (X^2+(Y/cosd(pitch))^2-L1^2-L2^2) /(2*L1*L2) );
theta1_1=( atan2(Y/cosd(pitch),X) - atan2( L2*sind(theta2_1) ,
L1+L2*cosd(theta2_1) ) )*180/pi-yaw;
theta1_1=OptimizationTheta(theta1_1);
theta2_2=-acosd( (X^2+(Y/cosd(pitch))^2-L1^2-L2^2) /(2*L1*L2) );
theta1_2=( atan2(Y/cosd(pitch),X) - atan2( L2*sind(theta2_2) ,
L1+L2*cosd(theta2_2) ) )*180/pi-yaw;
theta1_2=OptimizationTheta(theta1_2);
% choose a better solution from IK's two solutions
if n==1
    theta1_old=theta1_ini;
    theta2_old=theta2_ini;
end
delta_theta1_1=theta1_1-theta1_old;
delta_theta2_1=theta2_1-theta2_old;
delta_theta1_2=theta1_2-theta1_old;
delta_theta2_2=theta2_2-theta2_old;
delta_theta1_1=OptimizationTheta(delta_theta1_1);
delta_theta2_1=OptimizationTheta(delta_theta2_1);
delta_theta1_2=OptimizationTheta(delta_theta1_2);
```

```
delta_theta2_2=OptimizationTheta(delta_theta2_2);

if abs(delta_theta1_1)<abs(delta_theta1_2)
    theta1=theta1_1;
    theta2=theta2_1;
    delta_theta1=delta_theta1_1;
    delta_theta2=delta_theta2_1;
else
    theta1=theta1_2;
    theta2=theta2_2;
    delta_theta1=delta_theta1_2;
    delta_theta2=delta_theta2_2;
end
%-------------------------------------------------------
theta1
theta2
delta_theta1
delta_theta2
% movie generation
time=10;
t=[0 time];
 if n==1
   theta1vec=[theta1_ini theta1];
   theta2vec=[theta2_ini theta2];
 else
   theta1vec=[theta1_old theta1];
   theta2vec=[theta2_old theta2];
 end
Lb=5;
for i=1:360
tt(i)=i;
end
 for k=1:time
     % red spot (target)

plot(L1*cosd(theta1_ini)+L2*cosd(theta1_ini+theta2_ini),L1*sind(theta1_ini)+
L2*sind(theta1_ini+theta2_ini),'r o');
     hold on;
```

```
    % base
    plot(0,0,'square k','MarkerSize',10);
    hold on;
    plot([-Lb Lb],[0 0],'k','LineWidth',1.5);
    hold on;
    plot([0 L1*cosd(theta1_ini)],[0 L1*sind(theta1_ini)],'--');
    hold on;
    plot([L1*cosd(theta1_ini)
L1*cosd(theta1_ini)+L2*cosd(theta1_ini+theta2_ini)],[L1*sind(theta1_ini)
L1*sind(theta1_ini)+L2*sind(theta1_ini+theta2_ini)],'--');
    hold on;
    plot(x_move,y_move,'square k','MarkerSize',10);
    hold on;
    plot([x_move-Lb*cosd(yaw) x_move+Lb*cosd(yaw)],[y_move-Lb*sind(yaw)
y_move+Lb*sind(yaw)],'k','LineWidth',1.5);
    hold on;
%     plot([x_move x_move+L1*cosd(theta1_ini+yaw)],[y_move
y_move+( L1*sind(theta1_ini+yaw) )*cosd(pitch)],'b','LineWidth',1);
%     hold on;
%     plot([x_move+L1*cosd(theta1_ini+yaw)
x_move+L1*cosd(theta1_ini+yaw)+L2*cosd(theta1_ini+theta2_ini+yaw); ],[y_move
+( L1*sind(theta1_ini+yaw) )*cosd(pitch)
y_move+( L1*sind(theta1_ini+yaw)+L2*sind(theta1_ini+theta2_ini+yaw) )*cosd(p
itch)],'b','LineWidth',1);
%     hold on;
   % work region

plot(L1*cosd(theta1_ini)+L2*cosd(theta1_ini+theta2_ini)+30*cosd(tt),L1*sind(
theta1_ini)+L2*sind(theta1_ini+theta2_ini)+30*sind(tt)*cosd(pitch),'k')
   hold on;
   % creat movie
   THETA1(k)=spline(t,theta1vec,k);
   THETA2(k)=spline(t,theta2vec,k);

   plot([x_move x_move+L1*cosd(THETA1(k)+yaw)],[y_move
y_move+( L1*sind(THETA1(k)+yaw) )*cosd(pitch) ],'c','LineWidth',2);
   hold on;
   plot([x_move+L1*cosd(THETA1(k)+yaw)
```

```
x_move+L1*cosd(THETA1(k)+yaw)+L2*cosd(THETA1(k)+THETA2(k)+yaw)],[y_move+( L1
*sind(THETA1(k)+yaw) )*cosd(pitch)
y_move+( L1*sind(THETA1(k)+yaw)+L2*sind(THETA1(k)+THETA2(k)+yaw) )*cosd(pitc
h) ],'c','LineWidth',2);
    hold on
%
plot(x_move+L1*cosd(THETA1+yaw),y_move+( L1*sind(THETA1+yaw) )*cosd(pitch),'
k--');
%     hold on;
%
plot(x_move+L1*cosd(THETA1+yaw)+L2*cosd(THETA1+THETA2+yaw),y_move+( L1*sind(
THETA1+yaw)+L2*sind(THETA1+THETA2+yaw) )*cosd(pitch),'k--');
%     hold on;

    xlabel('X');
    ylabel('Y');
    title('X-Y');
    axis square;
    axis([-40+L1*cosd(theta1_ini)+L2*cosd(theta1_ini+theta2_ini)
40+L1*cosd(theta1_ini)+L2*cosd(theta1_ini+theta2_ini)
-40+L1*sind(theta1_ini)+L2*sind(theta1_ini+theta2_ini)
40+L1*sind(theta1_ini)+L2*sind(theta1_ini+theta2_ini)]);
    grid on;
    pause(0.01);
    hold off;
%   F(k+(n-1)*time)=getframe;
%   aviobj=addframe(aviobj,F(k+(n-1)*time));
 end

a=rand;
if a>rand
    a=1;
else
    a=-1;
end
% base move change
x_move=x_move+a*3*rand;
y_move=y_move+a*3*rand;
```

```
% base incline change
pitch=pitch+a*2*rand;
yaw=yaw+a*4*rand;

theta1_old=theta1;
theta2_old=theta2;

n=n+1;
end
%aviobj=close(aviobj);
----------------------
function theta_new=OptimizationTheta(theta)
if theta>180
    theta_new=theta-360;
elseif theta<-180
    theta_new=theta+360;
else
    theta_new=theta;
end
----------------------
```