第六屆旺宏科學獎

成果報告書

參賽編號： SA6-424

作品名稱：MIA 自動飛靶

姓　名　：王柏崴

關鍵字　：　電腦視覺、機械、拋體

目錄

壹、 摘要

本研究地目的希望以電腦視覺加上拋體計算，使電腦能控制氫氣炮射擊空中拋體飛靶。

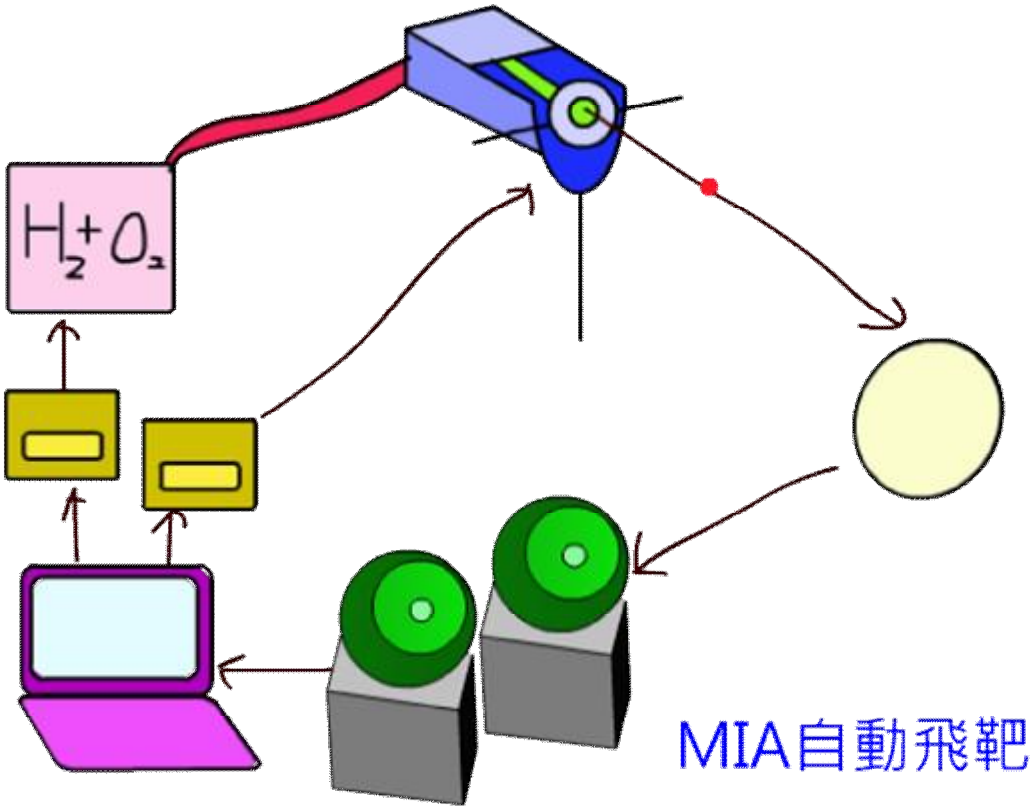
在導出了自拋體位置與速度推得射擊角度的公式之後，研究電腦視覺以得到此資訊，其中自己推導出了 camera calibration 的方程式，試圖以線性方法解之，並分析此方法的誤差所在。自兩攝影機同時取得圖像，利用立體視覺的概念，推得飛行中物體的位置以及速度。想出了只要用一隻相機就可以抓到拋體位置的構想。

做出了能夠考量電腦運算、馬達移動等時間的實時系統，解得如何操作砲管以射擊拋體的方法。

設計出許多高效率的電解器以及防爆閥，使得燃料的補充更快更安全。最新一代的電解器命名為菊花，有 21 個微型的電解槽。

實際以壓克力及生活中的材料做出了砲台，可以連續射擊、自動換彈、自動瞄準，維持燃料比例固定。

本研究設計了一塊 8051 實驗板，用以操作電解器及點火裝置。並為 servo motor 的操控板寫了 driver，可以準確的控制砲台的角度。



貳、 研究動機

某日在社團內想著，如何能將工研社內各領域的知識統合，讓一個學術性的社團也能有容易展現的作品。突發奇想，希望能以氫氧空氣砲搭配電腦視覺、物理運動學及自動控制，做出自動瞄準系統，實踐自動射擊拋體飛靶。

參、 研究目的

利用電腦運算以及機器實作，達成自動瞄準射擊飛靶之目的。

一. 找出射擊方向與拋體運動關係的方程式。
二. 實作解方程式的程式。
三. 實作射擊控制模擬系統 MIA。
四. 實作視覺定位 stereo vision。
五. 實作氫氧空氣砲。
六. 實作控制晶片程式。
七. 測試射擊機器效能。

肆、 研究設備器材

電腦、8051 晶片(Winbond W78E58B)、WebCam(Quickcam Sphere MP)、伺服馬達(S3003)、壓克力條、螺絲、鑽孔機、鑽頭、連環盒、塑膠氣密盒、塑膠管、NaOH、4mm 粗銅線、塑膠蘭花瓶、細管、熱融膠、繼電器、電源供應器、電子點火槍、BB 彈、線材、金屬棒、吸管、塑膠滴管、燒杯、瓦斯噴燈、美工刀、伺服馬達控制板。

伍、 研究過程或方法

一. 找出射擊方向與拋體運動關係的方程式

目的: 從拋體位置、速度、加速度，直接推得砲管仰角轉角。

1. 簡化模型

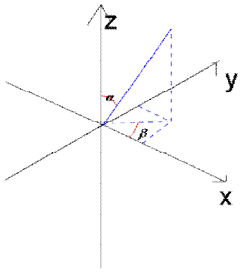
首先假設物體呈拋體運動，無空氣阻力，且槍管發射速率固定，槍管長度為零，發射所需時間為零，令槍管口所在位置為原點，則：

$$\vec{L}_{obj}+\vec{v}_{obj}t+\frac{1}{2}\vec{g}t^2=\vec{v}_{gun}t+\frac{1}{2}\vec{g}t^2 \overset{\text{兩邊消去}}{\Rightarrow} \vec{L}_{obj}+\vec{v}_{obj}t=\vec{v}_{gun}t$$

其中 $\vec{L}$:位置　$\vec{v}$:速度　t:時間　$\vec{g}$:重力加速度　obj:飛靶物體　gun: 空氣砲

再令 $|\vec{v}_{gun}|=v_{gun}=const.$ 且 $\vec{v}_{gun}=(v_{gun}\cos a\cos b,\ \ v_{gun}\cos a\sin b,\ \ v_{gun}\sin a)$

其中 a，$0\le a\le\frac{1}{2}p$ 為槍管垂直角度，b，$-\frac{p}{2}\le b\le\frac{p}{2}$ 為槍管水平角度，如下圖:



現在方程式中有 3 個未知數 a 、 b 、 t，在三維空間中將可以得到三個方程式，可解：

令 $\vec{L}_{obj}$ =(x, y, z), $\vec{v}_{obj}$ =(a, b, c), v_{gun} 簡稱 v, 得 $\begin{cases} x+at = v\cos a\ \cos b \cdot t \\ y+bt = v\cos a\ \sin b \cdot t \\ z+ct = v\sin a \cdot t \end{cases}$

又 $\frac{z+ct}{vt} = \sin a \Rightarrow \cos^2 a = 1-\left(\frac{z+ct}{vt}\right)^2$，且 $\left(\frac{x+at}{vt\cos a}\right)^2+\left(\frac{y+bt}{vt\cos a}\right)^2 = \cos^2 b+\sin^2 b = 1$

兩式合併得 $(x+at)^2+(y+bt)^2 = v^2t^2\left(1-\left(\frac{z+ct}{vt}\right)^2\right)$，展開整理得 t 之二次方程

$(x+at)^2+(y+bt)^2+(z+ct)^2 = v^2t^2$ ，解得 t(負則不合), 可能有兩組有效解，其中

$a = \sin^{-1}\left(\frac{z+ct}{vt}\right)$，則 $b = \cos^{-1}\left(\frac{x+at}{vt\cos a}\right)$

二. 作解方程式的程式

函式稱為 MIA 的核心，在 a 解不合或 t<0 或擊中位置高度<0 時將回傳不可。

Input: 一飛行物件 obj，含其位置向量 $\vec{L}$ 及速度向量 $\vec{v}$

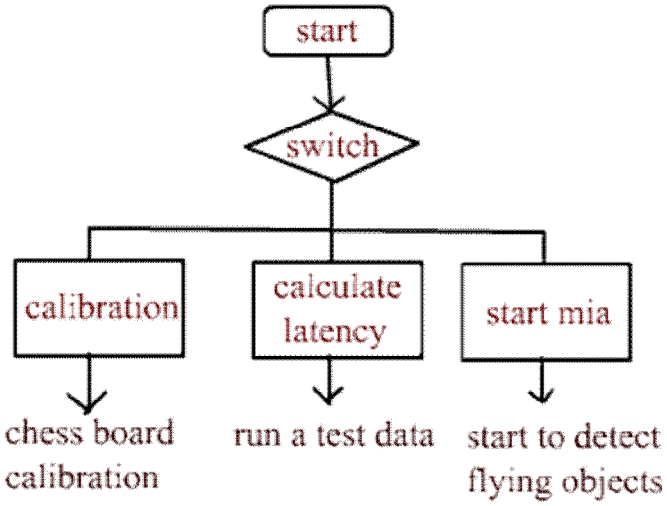
Output: 回傳不可能射中或兩組解(槍管仰角 a 、水平角 b)中較易射中的值

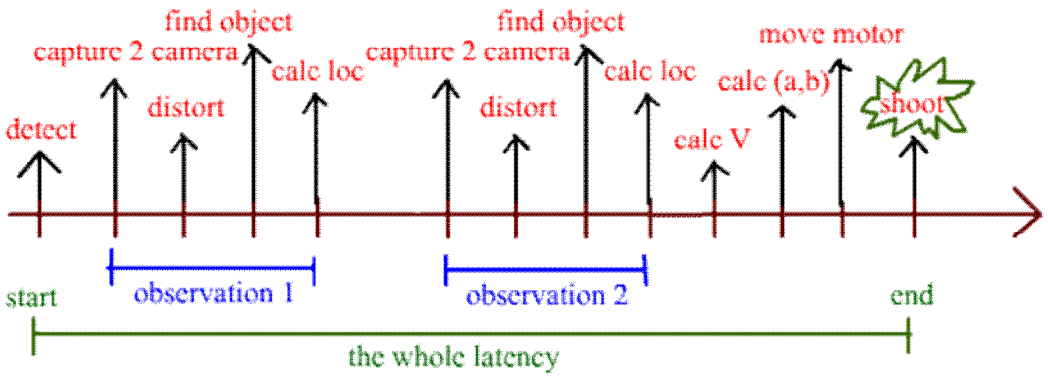
如上解方程，若運算過程中 sin, cos 值不合法、仰角小於 0 度或落地前無法射中 則傳回 0。若有兩組解，取需時較小者回傳，因誤差較小。

使用時，先計算機器就定位所需最長時間，將之定為延遲時間常數，於呼叫解方程函數時，先將 obj 的資料依此平移，可去除準備需佔用的時間。

三. 實作射擊控制模擬系統 MIA

由於馬達移動以及填充燃料需要時間，所以需要將物體的位置向後平移一段時間。





time line

四. 實作視覺定位 Stereo Vision

有一套滿好用的電腦視覺函式庫 Open Computer Vision(OpenCV)，是 Intel 開發的 Open Source 電腦視覺庫，支援跨平台，我選他來實作相機校正、濾波。裡面內建的 GUI 並不好用，於是我改寫了 Simple Direct Media Library(SDL)，實作較方便的顯示板。另外，裡面的相機擷取有些速度上的問題，故直接使用 Video For Linux 2(V4L2)寫 Webcam 的輸入。

1. 實做 Webcam 輸入

在 Linux 上，用 V4L2 的 API 作為視覺輸入的介面。此時，我第一次接觸到了 ioctl 的用法。我的相機 Quickcam Sphere MP 以 UVC 為驅動程式。我學習 luvcview 這個程式，以記憶體映射取得影像，這樣有助於取得影像速度的提升。我將原本 luvcview 裡面的資料格式重新包裝，方便多個相機的輸入。

2. 實作 GUI

在 SDL 與 OpenCV 間，完成了其圖片格式間的轉換(IplImage ⟷YUYV)，以便顯示及處理。學習了 SDL 的互動方法，完成簡單但快速的 GUI。實做了讓 Quickcam Sphere MP 可以轉動頭部的方法。

3. 實作 Camera Calibration 校正參數

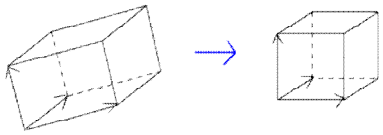
因為由攝影機取得的圖像與針孔模型有些差異，像是

- 一個 pixel 的長寬不同
- 攝影機不一定在原點
- 圖像中心不一定是攝影機中心
- 攝影機的投影平面不一定平行於 xz 平面

所以我需要假設一個轉換矩陣，要有平移、伸縮、推移、旋轉等功能。數學告訴我這些操作只是線性變換，安啦。

為了最快速的運算，我推了一遍方程式，也利用線性方法解出相機的參數(見附錄)，然後用人工選取圖片中的顏色來 floodfill，取得點的位置。但是由於此法的誤差太大，只好改採用 OpenCV 的 Calibration: cv3dTarckerCalibration，以最小平方法減少誤差。

cv3dTrackerCalibraion 利用 chessboard 類似西洋棋盤的平板來做找出資料點。利用校正產生出來的資料可以產生一組直角座標系。然而這組座標系的長度不一定和公制單位一樣，也不一定符合重力場方向。於是在校正完之後，再做一次線性變換，由使用者在畫面上選取理想的 x 單位、y 單位、原點。此處使用滑鼠選取座標，再 floodfill 相近顏色的區域。



4. 兩眼視差法

利用兩架攝影機實做出物體三維定位，又稱 Stereo Vision。

假設攝影機上每一像素的取得方法為球殼投影至平面，則可將畫面上的每一點對應到兩個角度，得空間中一射線。以兩個 camera 得二射線取交點，對球形座標與三維方塊座標轉換，求得物體的三維定位:

$$m\vec{u} = n\vec{v} + \vec{r}$$

其中 $\vec{u}$ 、$\vec{v}$ 為物體對該攝影機得到射線向量，$\vec{r}$ 為二攝影機鏡頭距離之向量

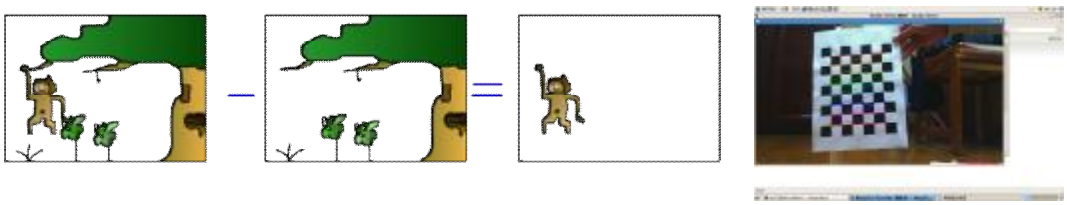
於三維空間中得三等式，兩未知數，可解得 m, n，推得物體之定位。

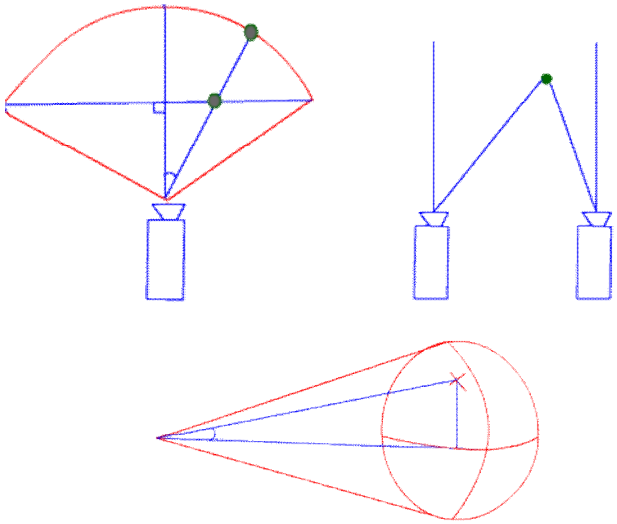
由前處校正參數可得等

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a(v_{a1}x' + v_{a2}z' + v_{a3}) + v_{a4} \\ a(v_{a5}x' + v_{a6}z' + v_{a7}) + v_{a8} \\ a(v_{a9}x' + v_{a10}z' + v_{a11}) + v_{a12} \end{bmatrix} = \begin{bmatrix} b(v_{b1}x' + v_{b2}z' + v_{b3}) + v_{b4} \\ b(v_{b5}x' + v_{b6}z' + v_{b7}) + v_{b8} \\ b(v_{b9}x' + v_{b10}z' + v_{b11}) + v_{b12} \end{bmatrix}$$

，此時只有二未知數 a, b，將 x, z 式聯立解得，再算出 x, y, z。

我以 OpenCV 的 cv3dTrackerLocateObject 實作。物體辨識以現在畫面與背景畫面相減而得到位置，其中使用了平滑化，型態學濾波，且背景會隨時間慢慢改變。





追蹤兩次此物體，經由時間差我們可以得知此物當時之速度向量。
輸入位置給 MIA 解方程，以計算物體的質心。

5. 求畫面中特定鮮明顏色物體的畫面位置且去除雜訊的演算法

除了使用 OpenCV 提供的濾波方法，我還使用了利用周圍 pixel 懲罰雜訊的方法，見附錄。此法可以將零星的雜訊消除，以免極端值的出現。

6. 由顏色取點的方法

比較畫面上顏色與標準顏色的距離(此處將 rgb 三色當作向量)，再篩選最適範圍，可以使用顏色或色調選取畫面中的點，但此法只有在背景單純且目標與背景色差大時適用。我利用單色保麗龍板上面釘圖釘來測試 threshold 的最佳值。

7. 單眼旋轉

因 Quickcam Sphere MP 實在是太貴了，加上他又有鏡頭旋轉功能而未利用，想出此方程，只要一個鏡頭旋轉拍三次，就可以知道拋體的位置，速度。拋體專用。
令攝影機鏡頭每 d 秒拍一次，每次位置差 wd 度，每次拍照可得向物體一射線，對於拋體有方程式:

令 $\vec{u}_n = (\cos b_n \cos a_n, \sin b_n \cos a_n, \sin a_n)$，表示第 n 次拍攝的射線的單位向量，則

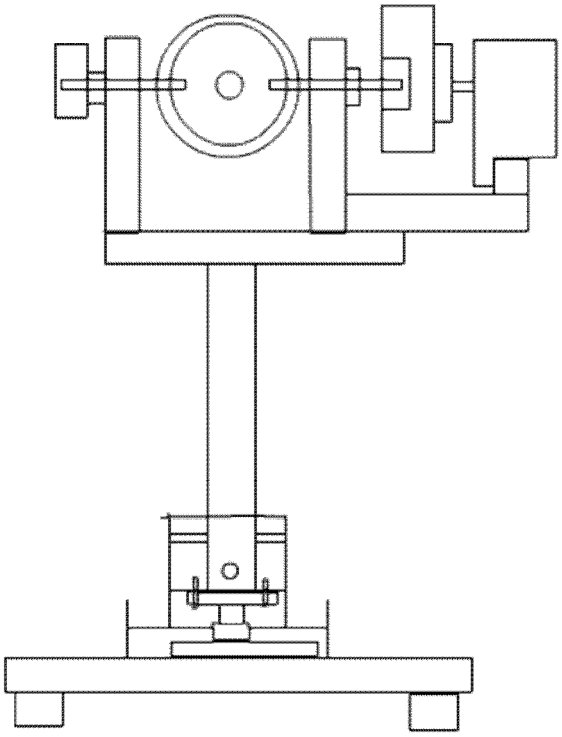
$r_n \vec{u}_n = \vec{L} + \vec{v}\, t_n + \frac{1}{2}\vec{g}\, t_n^2$，已知 a_n, b_n(轉角), $\vec{g}$(重力加速度), t_n(時差)，n 式有 3n 道不重複方程式，有 6+n 的變數，得 n=3 時可解得 $\vec{L}, \vec{v}$。
因為這只是個 9 元聯立方程組，先用克拉瑪公式檢驗有無實數解，再用高斯消去法解得。看似很神奇，只用一隻眼睛就能得到所有資訊，與前人的方法不同。其實這是因為重力加速度保持一定的原故。此法要求相機轉的角度精準。

8. 實作高精度解方程式

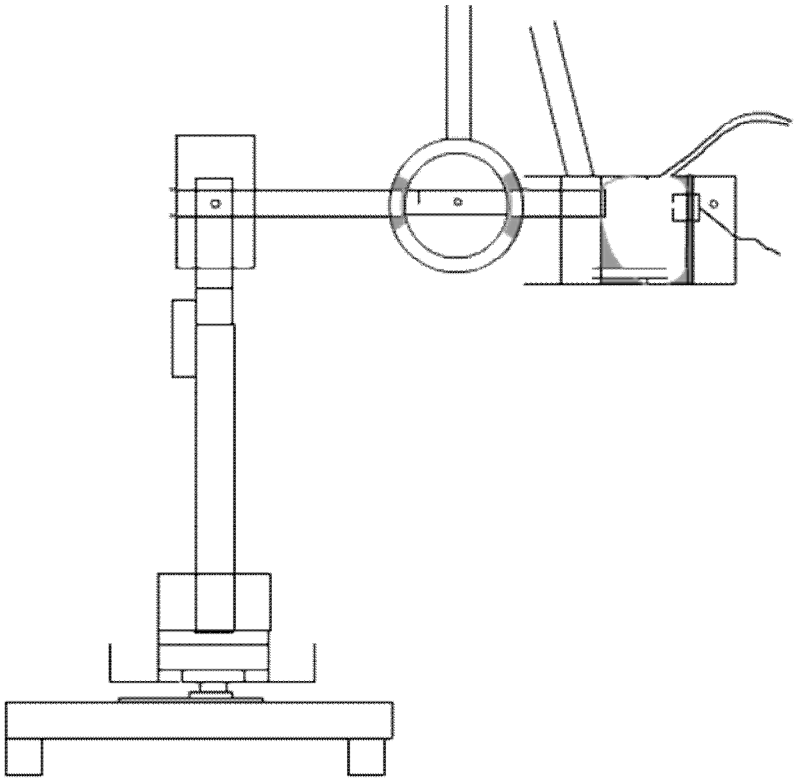
由於前面的 camera calibration 要解一個 20 元一次方程式，且浮點數系統經過越多次計算誤差將越大，我決定利用 GNU 的 GMP 大數運算庫寫線性方程式的解以減少運算誤差。我實做了 Cramer 解以及 Gauss-Gordan 解，並改良了 Gauss-Gordan 解法交換行的速度，使用自定長度浮點數結構 mpf。經過分析後發現，我的 DLT 方程式在運算誤差最小化後仍存在矛盾：因為是線性的關係，所以圖片上位置的誤差被放大。

五. 實作容易控制的氫氧空氣砲

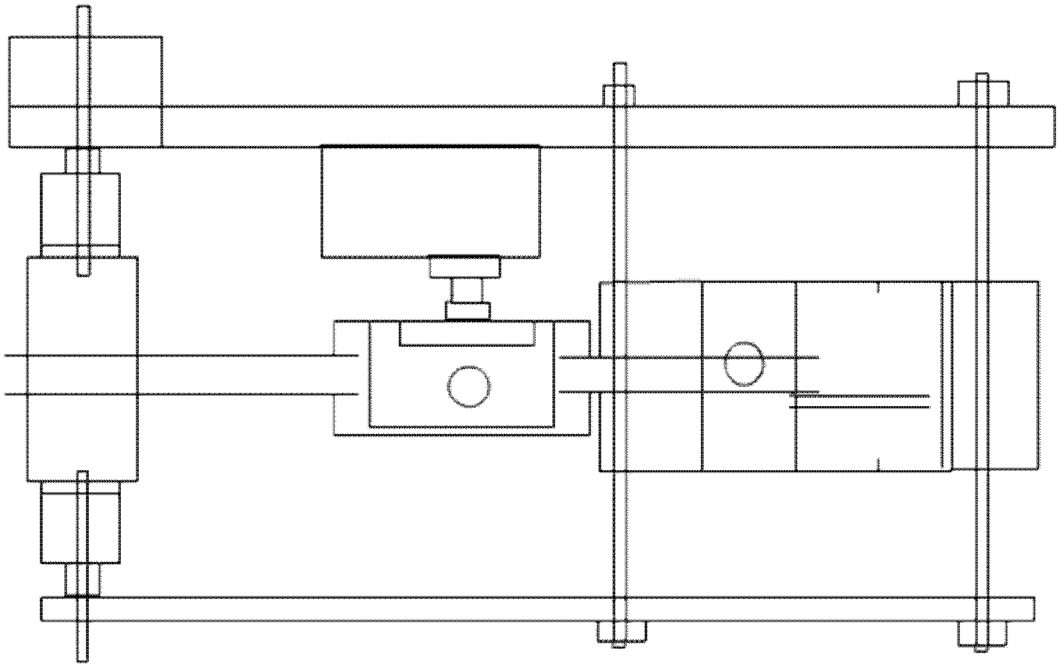
三視圖與照片



前視圖



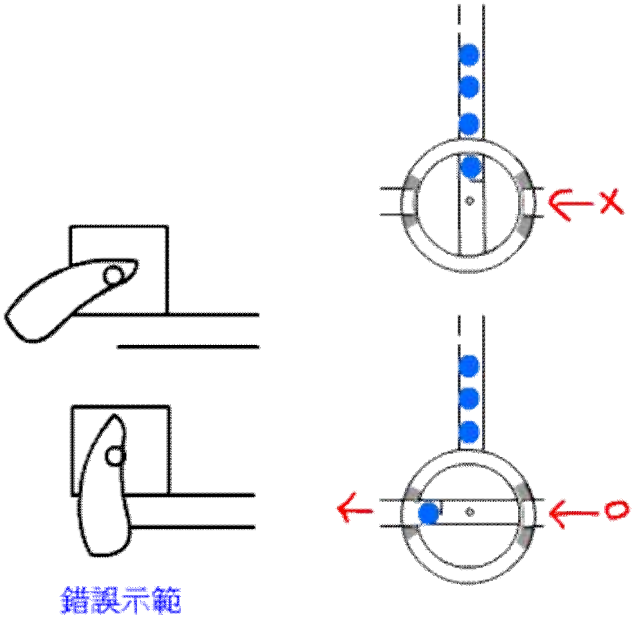
側視圖



上視圖

換彈器兼氣密閥

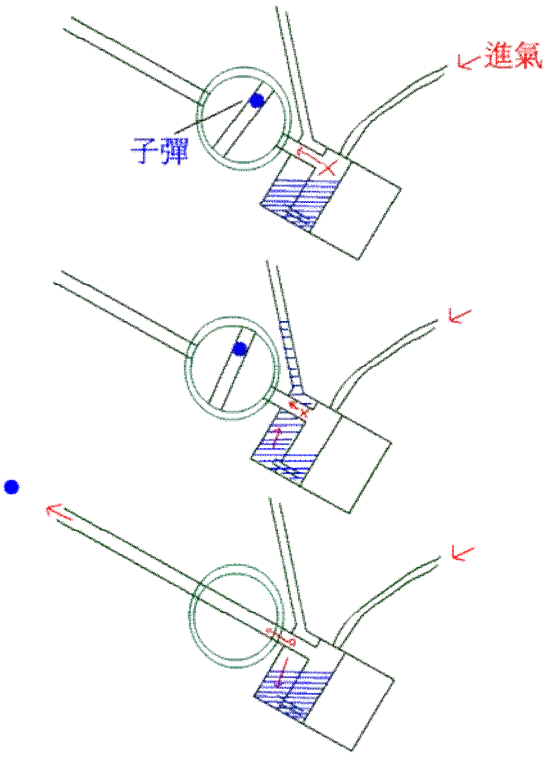
一開始的構想頗蠢，想說在炮管前面裝一支大姆哥狀的塞子，再利用槓桿傳動換彈器，但是實作起來十分麻煩而好笑。現在利用伺服馬達轉動內輪，可同時達到氣密和換彈效果。氣密部灰色特難做。我先在內輪塗上肥皂，再用熱融膠黏合，模型冷卻後包上膠帶減少摩擦力，之後就可以輕鬆的將內輪轉動。



錯誤示範

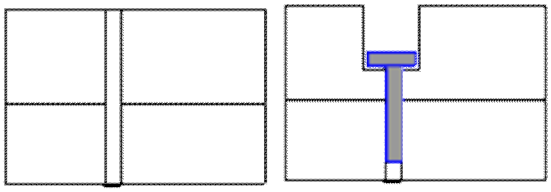
氣體比率調節

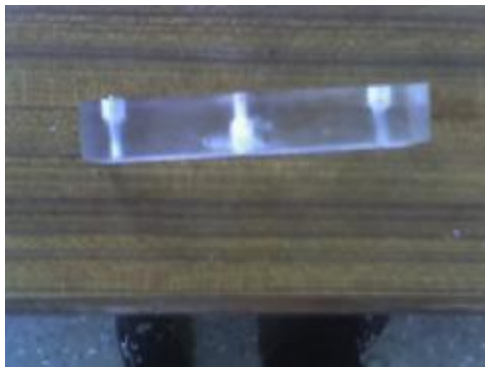
以水壓力差的方式，使得每次充氣時水的部分會被置換為氫氧混合氣，發射完後因為壓力差水又倒流回來。如此可用來保持空氣與氫氧混氣的比例一致，以確保子彈出管速度。



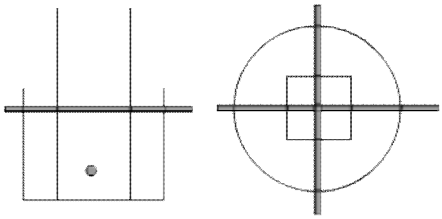
螺絲釘與十字固定

整個主機我都用螺絲釘來連接，這樣方便攜帶。因為塑膠柱較長，螺絲釘較短，我想出了這個接法，只要鑽兩次即可。



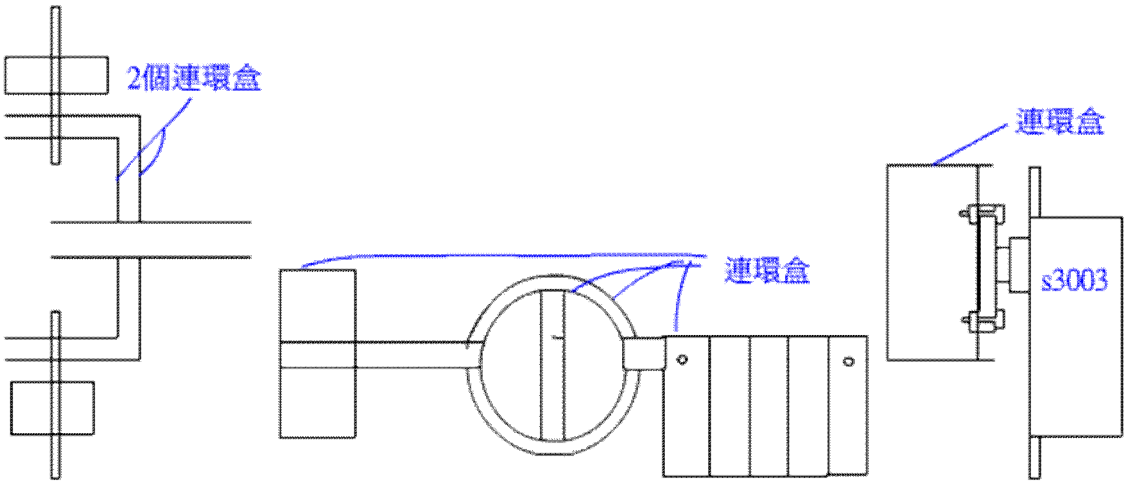


在連接壓克力柱與連環盒時，我想出了可以讓他們兩個連接的更緊的辦法:用兩個四驅車軸承十字固定。



連環盒的妙用

在本作中，大部分的主體由連環盒構成，如: 砲口旋轉處，炮身，與馬達連接處。



新的關節

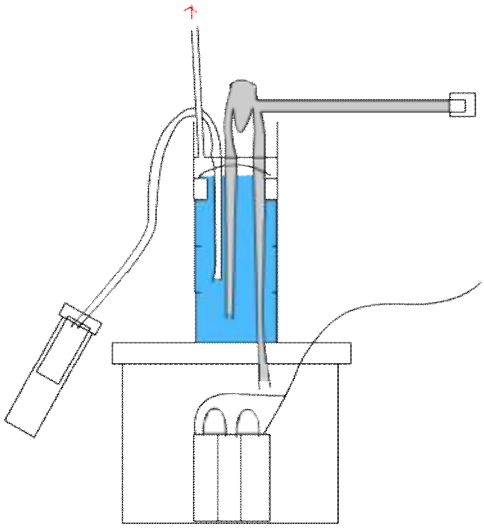
之前底座的固定只由一隻伺服機支撐，有些搖晃。一直在尋找替代的方案。恰巧被我發現有一種牙籤盒可以當作承軸，改造後解決了底座搖晃的問題。

防爆閥的原理:

將氫氧爆炸區與電解槽以水隔離，避免在發射點火時電解槽自爆。鐵網是用來防止水滴噴濺至輸出管口。

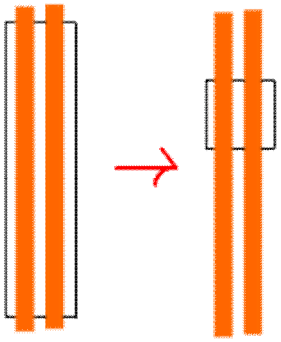
外接小蘭花瓶方便注水，外接水平的管子則用在機器關閉後，防止熱空氣冷縮，吸走防爆水，使電解液變稀薄，電解效率降低。箭頭處為出氣口。

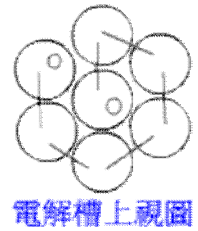
此防爆閥到最後還是爆了。所以有第二代的產生。



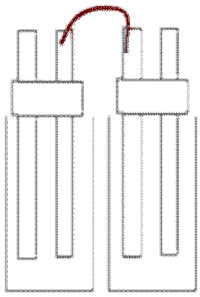
Cell 的原理

因為電源供應器的電流量有限制，故將多個電解槽串聯，形成 Cell，以充分利用電源供應器。如此即使電流只有 1 安培，但流經 4 個槽，產量為 4 單位。若是水乾了，可打開蓋子加水。Cell 的電極為雙粗銅電線，是家庭常用的那一種，製作方便，取代方便。





電解槽上視圖



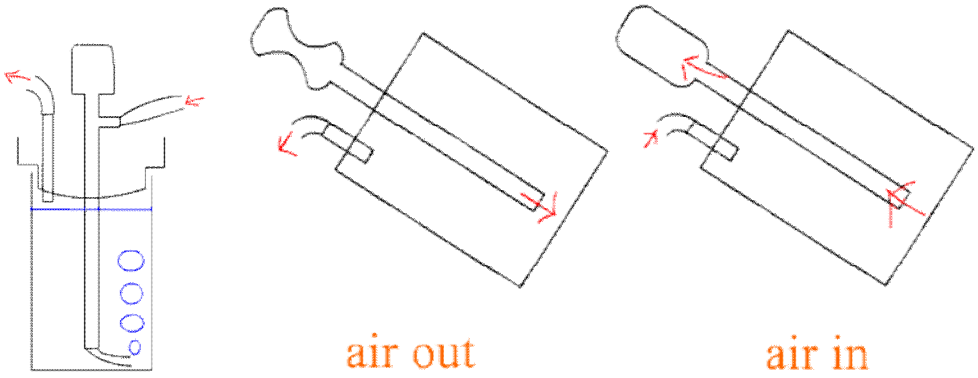
電極連接方法

防爆閥二代

因為防爆閥爆了，所以想出了新的設計。

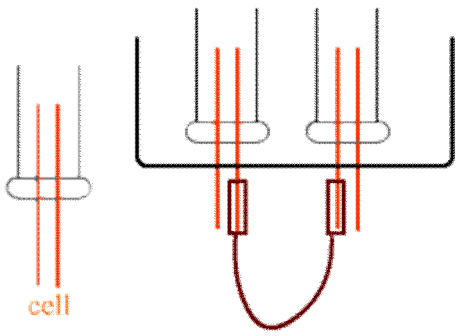
檢討前代失敗的原因，可能是因為氣泡冒上去的路徑太窄的關係，所以利用牙籤罐做出了以下耐用的防爆閥。上面的是塑膠滴管，外接氣體輸入。短的是滴管頭，接氣體輸出。這

個設計的好處就是簡潔，而且要加減水的時候只要把裝在上面的滴管當作幫浦就好了。



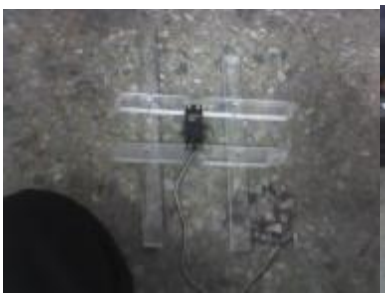
電解器二代

上戴的電解氣面臨了一個問題:電極間的銲接。由於 NaOH 實在很猛烈，有可能與焊錫 Sn 結合成白色的 $Sn(OH)_2$，所以焊接點不可以在電解器內，要不然一陣子之後電阻會變很大。為了使效率更好，連發速率更快，我把 cell 由 7 個增加到 21 個，且改用針腳取代銅電線，用杜邦街頭配線。製作的難度在 cell 的電解液不能相通。我用吸管加上熱熔膠及 AB 膠製造出了很多顆像是電解電容一樣的 cell，集結起來放入電解器中，做了 4 次才成功。此電解器外型很好看，配上綠色的杜邦線就像一朵菊花，吐著金色的花蕊(雖然黏起來就看不到了)，故命名為菊花電解器。



底座的演替

原本是一個井字型的塑膠條夾住一隻伺服機，改良後我把 8051 板，伺服馬達控制板，新關節都放進去底座裡面(合體)，這樣比較不占空間，且方便集線。



Old

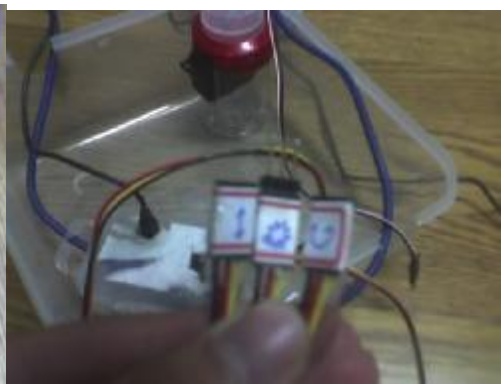


New

l 集線的創意

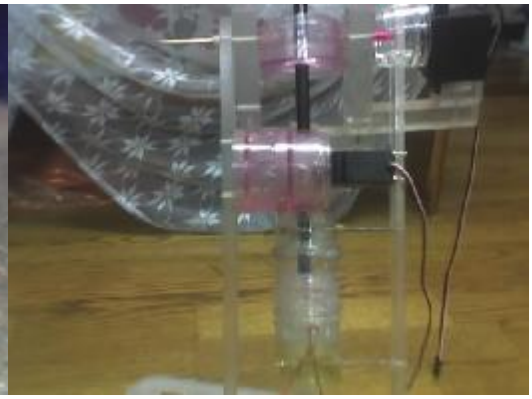
左圖是電源及 USB

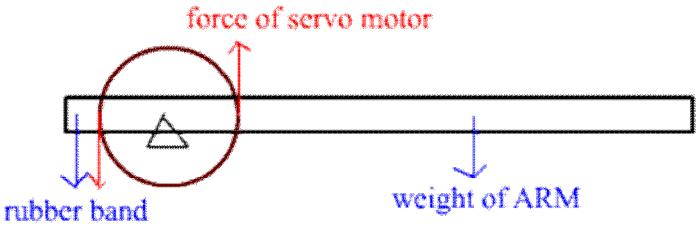
右圖是伺服馬達的線，左至右是上下，換器，左右

l 完成的 ARM

完成的砲身加上換彈器就像機器人的手臂(的位置)。加了一個連環盒，輔助換彈器保持不動。由於 ARM 有點重，我在負責上下的伺服馬達加上橡皮筋以減少其所需負擔的力矩。



六. 實作控制晶片程式

I 8051 控制板

採用 WinBond 的W78E516B 單晶片，我設計了一塊實驗板，命名為 ITRS，有以下功能：

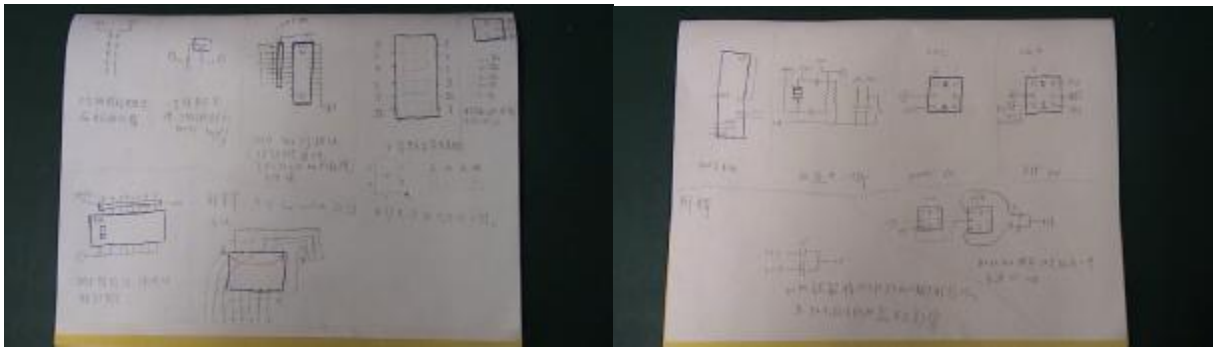
1. LED*8
2. 七節顯示器*4
3. DIP*1 (指撥開關 8 鈕)
4. WinBond ISP 快速切換功能
5. 由 5v 變壓器供應電源
6. IO 模組化設定，要用再跳線
7. LED 與 七節+LED 開關切換
8. 方便外接的針腳
9. Low Active

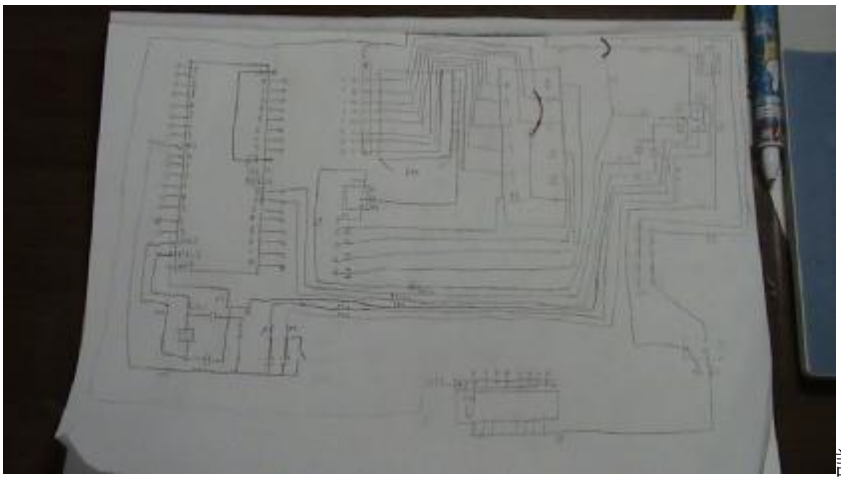
其實這是我為社團設計的板子，我所要用的功能只有 IO 而已。

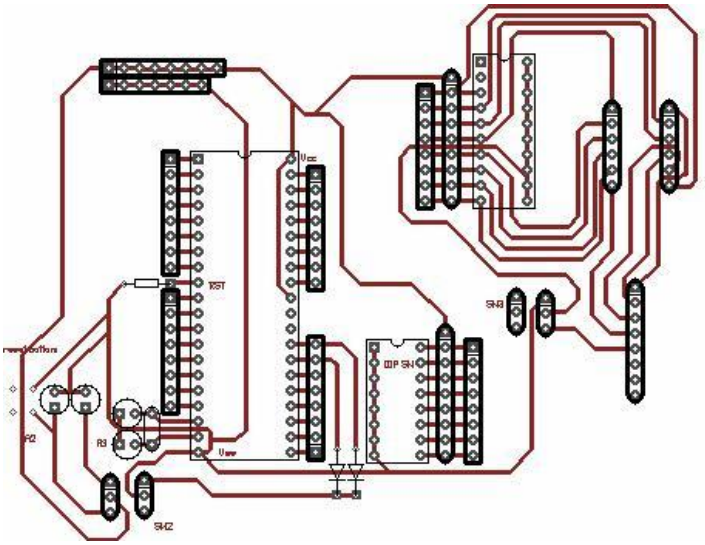
另外我在下面又外加了一塊板子，有兩個繼電器，用來操縱電子點火器及電源供應器。

由於我只會用 GSCHEM 設計電路，下面的 PCB 是請學長幫忙 layout 的。


這是原型


設計手稿


設計手稿(假想的 layout)


實際的 PCB layout

十軸伺服馬達操控板

原本這部分是要用 8051 自己寫 Power Management(PWM)，但是寫出來發現很難控制脈波，就買了控制板。這是學長的朋友開發的，用來做無人飛行器控制，可以精確的控制角度以及角速度，但是只有 Windows 驅動版本。此操控板採用 Human Interface Device(HID)的 USB 介面，我參考了他的操作碼之後幫他在 Linux 底下寫了驅動程式，順便學會了 USB 控制。這是我第一次寫 Driver，滿興奮的。

軟體

我用 linux 底下的 SDCC 作為編譯器實做出了一個 client，以 8051 的 RX, TX 透過 UART 介面與 PC 的 COM 通訊。當命令傳過來時會產生一個中斷，此時判別並執行。

PC 端使用 Python 收發資料，再用 C 執行 Pyhon 程式，因為對於序列復通訊的實作 Python 比 C 省力多了，而且找到範例可用。


UART

陸、 討論及應用

❙ 雙眼與單眼旋轉的比較

	雙眼	單眼旋轉
拍攝次數	2	3
拍攝張數	2*2=4	3*1=3
需要相機	2	1
靜物定位	可	不可
拋體定位	可	可

❙ Cramer 法以及 Gauss-Jordan 法在浮點數系統解方程式誤差的比較

測試結果發現，Cramer 法最大的弱點就是其複雜度為 O(n!)，而 Gauss-Jordan 為 O(n^3)，但若時間允許，Cramer 法的乘除次數較少，加法次數較多，而 Gauss-Jordan 的乘除次數較多。以本次 Camera Calibration 方程式(10 元)的線性解法發現，若使用 ANSI C 的 double 格式，Cramer 法的誤差(就純代入資料而言)約在10^{-4}，而 Gauss-Gordon 法可能會無解。

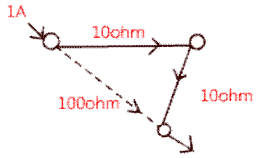
最佳的方法還是改變資料格式，例如使用大數運算庫，這樣就可以安心使用 Gauss-Gordon 法。

❙ 電解器效率的探討

首先我們欲得到的是最大效率，即單位時間內通過電解槽 cell 的電流最多。但是電源供應器限制電流，故欲使效率增大，應改變功率 P 維持電流 I，也就是增大電阻 R。

電解槽中的正負離子移動，將與水分子產生碰撞，產生熱 H。依照此模型可推得兩電極之間距離要盡可能靠近。又欲 R 大，故串連電解 cell 以增大效率。

探討若 cell 之間的電解液不小心相通，則電流的流動如下，實線為預定路線。



假設遵守歐姆定律，則原來(通過電流*cell)=2，通過虛線後 (通過電流*cell)=11/6

此狀況下電阻變小，總電流變大，流經每個電解槽的電流和變小，效率降低，電流量分配不均，很明顯可以看到兩極的電解速度增加。

柒、 結論

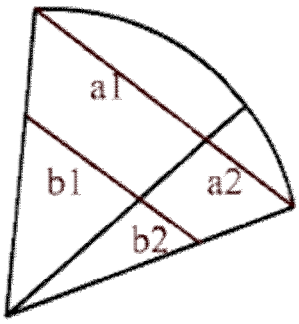
MIA 結合了數學、物理、化學、資訊、電子等領域，以及一年半來在社辦做的實驗，可以說是既深又廣，一開始就是因為說出這個構想沒人相信我能達成，才專心研究此題目。資訊方面從準確的時間至電腦視覺，單晶片控制，以及數學中球型座標系統的概念，矩陣，化學的電解，物理的拋體，電子控制輸出入等。裡面每一件材料都是自己完成的，包括拆了好幾輛四驅車拿到的軸承，裝電子零件用的連環盒，吸管，剛好符合吸管大小的滴管等，挖了一陣子才找到堪用的材料。換彈器兼氣密閥的構想更是在蠢主意中脫穎而出，想了好久才想到還可以將連環盒如此利用。看到砲台主體終於站起來的時候，感覺很驕傲。在高中時代完成一台機器人是我的一個夢想，我也盡力達成。

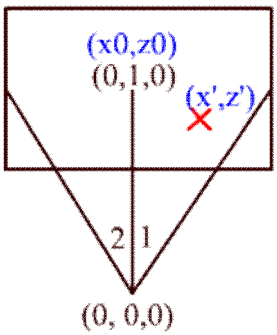
捌、 附錄

一、 Camera Calibration 參數的推導

在此先敘述基本的模型。基本假設為物體由遠方圓周面上向鏡頭投影，成像在一個長方框裡。由三角形的相似性質可得只要物體在同一向鏡頭射線上，就會在此長方框內同一比例位置成像，不論長方框的大小，因其大小也與具鏡頭距離成比例。

我們令鏡頭在(0, 0, 0)，此長方框在鏡頭前(0, 1, 0)的地方，則物體在位置(x, y, z)與畫面上位置(x', z')有以下關係





$$x=\left((x'-x'_0)\times\frac{\tan\angle 1}{(\frac{w}{\tan\angle 1+\tan\angle 2}\times\tan\angle 1)}\right)\times a=\left((x'-x'_0)\times\frac{\tan\angle 1+\tan\angle 2}{w}\right)\times a=(x'gp+q)\times a$$

$$y=a$$

同理$z=(z'\times r+s)\times a$

其中 x'_0 是指由鏡頭對成像處的矩形做垂線對應的 x 螢幕座標，也就是指中心點。因為鏡頭上下的可見角度可能不一樣，故如此處理。同理得 z 式

p,q,r,s 的確定方法可由人工得到，但是太不精準，所以就設計了校正程序來產生這些參數。令一物體在第一鏡頭前(x, y, z)處，x, y, z 已知，得 $\begin{cases} x=(px'+q)a=(px''+q)b+i \\ y=a=b+j \\ z=(rz'+s)a=(rz''+s)b+k \end{cases}$，其中 (x',z') 為第一鏡頭的畫面座標，(x'',z'') 為第二鏡頭的。因兩台相機型號相同，由一式平移可得此式，又 (i,j,k) 為以第一鏡頭為原點第二鏡頭的位置。此為 $2n+3$ 元一次聯立方程式，n 為聯立組數，則提供二組資料可解得。

解法：

$\begin{cases} (y_1x'_1)p+(a_1)q=x_1 \\ (y_2x'_2)p+(a_2)q=x_2 \end{cases}$ 得 p, q，同理得 r,s，再解 $\begin{cases} (px_1''+q)b_1+i=x_1 \\ (rz_1''+s)b_1+k=z_1 \\ (px_2''+q)b_2+i=x_2 \\ (rz_2''+s)b_2+k=z_2 \end{cases}$ 得 b_1,b_2,i,k，再 $y_1-b_1=j$ 得解。

此校正的好處是兩隻攝影機除了參數之外，連相對位置都找到了，不假人工。若校正正確，以後再平移的時候只會有常數誤差。

剛剛我們嚐試解決長寬的比例問題，但是測試結果並不盡理想，比對畫面之後發現兩個攝影機的中心點並不如預期的同一個水平面上，這時有兩個方法可解決:

1. 利用 QuickCam Sphere 的特殊功能，直接旋轉鏡頭對齊校正。
2. 加入攝影機鏡頭經緯參數 a,b 。

法 1 可行，但是比較難保證兩者的平行，精確度不好。故我採用法 2。一開始我假設 a,b ，原來三點(x, y, z)設出三角方程式，想解 n+4 元三角方程式，每次提供 3n 條，但是因為帶有一堆根號，解起來很煩。剛好下學期要學矩陣，我先修了一下，發現有個好東西——轉角矩陣，大大減輕了我思緒的繁複(好工具就是如此，要不然人類就全用相似三角形解幾何了)。這讓我很容易的列出一個 n+4 元三角方程式，每次提供 3n 條，理論上只要 2 個點就可解，但是還是太煩，因為三角和多項式混合在一起。我想到一個很暴力的解法: 我把它做代數變換，換成多元一次多項方程，反正增加變數沒在怕的，只要我代入多一些點就好。於是我把他代數變換，造出了一個 12 元二次方程式，再換成 21 元一次方程式，以高斯消去法解得，再向上解回去。如下:

延續上次的變換代入，其中 a,b 為與 x 軸夾角，仰角。

令 $\begin{bmatrix} x_r \\ y_r \\ z_r \end{bmatrix}=\begin{bmatrix} (px'+q)a+i \\ a+j \\ (rz'+s)+k \end{bmatrix}$，則

$$\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}\cos a & -\sin a & 0\\ \sin a & \cos a & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos b & 0 & -\sin b\\ 0 & 1 & 0\\ \sin b & 0 & \cos b\end{bmatrix}\begin{bmatrix}x_r\\y_r\\z_r\end{bmatrix}$$

$$=\begin{bmatrix}\cos a\cos b & -\sin a & -\cos a\sin b\\ \sin a\cos b & \cos a & -\sin a\sin b\\ \sin b & 0 & \cos b\end{bmatrix}\begin{bmatrix}x_r\\y_r\\z_r\end{bmatrix}$$

$$=\begin{bmatrix}t_1 & t_2 & t_3\\ t_4 & t_5 & t_6\\ t_7 & 0 & t_8\end{bmatrix}\begin{bmatrix}x_r\\y_r\\z_r\end{bmatrix}$$

$$=\begin{bmatrix}a(\boxed{t_1p}x'+\boxed{t_3r}z'+\boxed{t_1q+t_3s+t_2})+\boxed{t_1i+t_2j+t_3k}\\ a(\boxed{t_4p}x'+\boxed{t_6r}z'+\boxed{t_4q+t_6s+t_5})+\boxed{t_4i+t_5j+t_6k}\\ a(\boxed{t_7p}x'+\boxed{t_8r}z'+\boxed{t_7q+t_8s})+\boxed{t_7i+t_8k}\end{bmatrix}$$

我把攝影機參數(不變)，變數(每次要解)，係數(傳入的已知)分開整理變換，得

$\begin{bmatrix}x\\y\\z\end{bmatrix}=\begin{bmatrix}a(v_1x'+v_2z'+v_3)+v_4\\ a(v_5x'+v_6z'+v_7)+v_8\\ a(v_9x'+v_{10}z'+v_{11})+v_{12}\end{bmatrix}$，這是一個 12+n 元 2 次方程式，每次提供 3 條方程。我把 a 消掉，得 $a=\frac{x-v_4}{v_1x'+v_2z'+v_3}=\frac{y-v_8}{v_5x'+v_6z'+v_7}=\frac{z-v_{12}}{v_9x'+v_{10}z'+v_{11}}$，展開得

$$\begin{cases}xx'v_5+xz'v_6+xv_7-x'v_4v_5-z'v_4v_6-v_4v_7\\ -yx'v_1-yz'v_2-yv_3+xv_1v_8+zv_2v_8+v_3v_8=0\\ zx'v_5+zz'v_6+zv_7-x'v_{12}v_5-z'v_{12}v_6-v_{12}v_7\\ -yx'v_9-yz'v_{10}-v_{11}+x'v_9v_8+z'v_{10}v_8+v_{11}v_8=0\end{cases}$$

，直接把二次項當新變數，得一 21 元 1 次齊次方程式。發現是齊次，也就是有比例或者全為零，不能讓電腦直接跑高斯。我選了共同項 v_5，令 $v_5=t$，同除之，得 20 元 1 次方程式，每次提供 2 式，要 10 個點才能得。解這個電腦最在行，用了一個高斯消去法以及檢查是否可能解。又把它改良，使得列互換更快，不詳述。

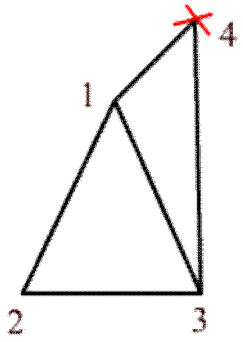
到這邊我們得到了 20 元對 v_5 的比例。代入有 a 的式子可化為型如 $\begin{cases}t(am+n)=x\\ t(ao+p)=y\end{cases}$，其中 a,t 未知，消去 t 再解 a，得 v_5，得 20 元的值。再經過簡單的除法，得 $v_1\sim v_{12}$ 的值，其中有四個變數重複解，可以用來驗證精確度。

這樣省去複雜的二次運算，以後要由螢幕上的點解實際位置，就直接帶入

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a(v_1x'+v_2z'+v_3)+v_4 \\ a(v_5x'+v_6z'+v_7)+v_8 \\ a(v_9x'+v_{10}z'+v_{11})+v_{12} \end{bmatrix}$$

。我畫了一張紙，上面有幾個圓，前後移動(因為若點共平面將無解)，當做標準座標系裡的點。讓兩個攝影機同時抓，這樣相對位置也有了。

其實這個解法還蠻浪費的，但是因為點的數目並不是瓶頸，故忽略之。我用 GMP 大數運算庫寫了 cramer 方法以及 Gauss-Jordan 方法解得校正參數。但是發現這個方法的誤差滿大的，因為：我給他的點太多了，而太多點的誤差將在此線性方程式累積；舉例來說：若有三個點，則加入第四個共平面的點時若有誤差，即帶入值不合，將造成矛盾。而越多的點若只用線性方法可能造成越多的誤差。



二、 求畫面中特定鮮明顏色物體的畫面位置且去除雜訊的演算法

明明攝影機傳入的圖片裡的圖形就是這麼清楚的在那個方位，若是只用平均，電腦抓到的位置就是會比實際位置偏了一點，因為雜訊也可以參加平均。我設計的方法利用該 pixel 周圍的臨近點來懲罰他的值，可有效縮小雜訊影響。

Procedure catch_obj_by_color(img , color) return x, y

Inputs: img, the image
color, the color we'd like to choose

utput: (x,y), the coordination of the object in image

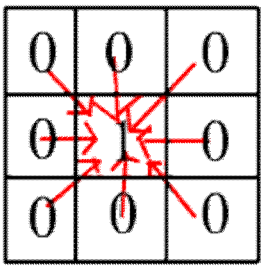
Local variables: thold, the threshold
range, the tolerance
min, the min element in data2, initialed as 無窮大
data[w][h], a array to store the distance
data2[w][h]; a array to store the modified distance
sx, sy, sn, n, i, j

For each pixel(i, j) in img
data[i][i] = (pixel[i][j],r – color.r)^2+ (pixel[i][j].g-color.g)^2+(pixel[i][j].b-color.b)^2

For each element E[i][j] in data
n=0

```
        For each neighbor of E ( 九宮格的範圍)
            If(E> threshold)
                n = n+1
        data2[i][j] = data[i][j]<<n
        if(data2[i][j] < min)
            min = data2[i][j]
    for each element E[i][j] in data2
        if(min+range > E[i][j])
            sx = sx+i;
            sy = sy+j
            n = n+1
    (x, y) = (sx/n, sy/n)

    Return (x, y)
```



玖、 參考資料

一、數學課本(三、五)

二、物理課本(三、五)

三、OpenCV Wiki

四、C 語言於資料結構及演算法的應用 河西朝雄 著 ISDN: 957-527-488-1

五、C++ 程式設計 張耀仁 著 ISDN: 986-421-475-6

六、Luvcview by Laurent Pinchart && Michel Xhaard

七、V4l2 API document http://www.thedirks.org/v4l2/

八、V4l HOWTO http://pages.cpsc.ucalgary.ca/~sayles/VFL_HowTo/

九、SDL tutorial by http://lazyfoo.net/SDL_tutorials/index.php

十、Artificial Intelligence: A Modern Approach 參考如何寫虛擬碼

十一、 OpenCV China

十二、 8051

十三、 Hiddev example http://www.charmed.com/txt/hiddev.txt

十四、 Tracker calibration example by Micah Dowty @ micah@navi.cx

十五、 Introduction to calibration http://kwon3d.com/theory/calib.html

十六、 Example of stereo vision http://arti.vub.ac.be/~bartj/vision.html
十七、 LatteBox for servo motor control board http://groups.google.com/group/lattebox-e-service
十八、 V4L2 example
http://www.messinalug.org/mediawiki/index.php/Video4Linux_HowTo:_Come_realizzare_un_semplice_frame_grabber
十九、 Servo motor datasheet http://www.societyofrobots.com/actuators_servos.shtml
二十、 PWM introduction http://yukuan.blogspot.com/2006/12/motor-controlling-pwms.html
二十一、 Quickcam sphere mp PTZ
http://osdir.com/ml/linux.drivers.uvc.devel/2006-10/msg00056.html
二十二、 C 語言常見問題集 http://twpug.net/docs/ccfaq/ccfaq.html