中華民國第 61 屆中小學科學展覽會

作品說明書

國中組　生活與應用科學(一)科

第二名

032807

AI 影像辨識輔助視力量測系統

學校名稱：新北市立林口國民中學

作者：	指導老師：
國二 蔡奕章	邱佩璇

關鍵詞：AI 影像辨識、視力檢測、Yolov3

摘要

近年來因為 COVID-19 緣故，學生線上學習的比例提高，但礙於疫情無法頻繁至眼科做視力追蹤，故本研究利用 Yolov3 建立可辨識手勢的 AI 模型，並結合視力量測規則，研發出無人協助即可達成即時量測視力的人工智慧系統。

研究結果顯示，本系統所得之視力檢測值平均準確度可達 96.645%，相較傳統量測視力之總誤差平均值僅有 0.096，且每人次的量測時間僅需約五分鐘。完成檢測後，系統會自動將量測結果上傳至雲端資料庫，並可利用本研究設計的視力追蹤 App 進行視力變化查詢。

綜合上述，本研究可在一般家庭裡使用、紀錄及追蹤視力變化，未來更期許可進一步配合使用在醫療診所或學校，協助醫護人員量測視力，減輕醫護人員的工作量。

壹、研究動機

本研究開始於 2020 年 COVID-19 開始爆發時，結束於 2021 年 COVID-19 造成臺灣三級警戒，研究初期恰逢開學時期，學生要進行眼科回診，我發現在這樣的「視力檢查高峰期」讓學生花費許多時間進行等候，因為量測視力需要一位醫護人員協助量測視力，非常耗費人力和時間，因此我想設計一套 AI 系統來幫助醫護人員減少工作量，增加視力檢查的效率。

一般傳統的量測視力方法，需要一張標準視力表，然後透過量測人員的指導及指示進行量測，當量測人員指到或顯示視力視標時，待測者需用手或口語明確指出視標 E 缺口的方向，量測人員須檢測待測者的手勢方向來判斷其回答是否正確，然後根據標準量測視力的規則進行下一個視標 E 的量測（衛生福利部國民健康署，2016）。

當前人工智慧在很多領域皆有重大的發展，尤其是在影像辨識的方面，例如：開源的 Yolov3 的 AI 模型在目標檢測方面達到了很好的效果，可以辨識出各種物品的圖片。於是本研究開始思考，利用 Yolov3 模型能不能辨識出檢測視力時受測者比的手勢？如果能正確的辨識出受測者的手勢，再配合量測視力的規則，就可以達到自動量測視力的成效。

綜合上述，本研究想要利用人工智慧影像辨識的功能，來輔助視力的檢測，期望達到協助醫護人員減低工作量，也可以讓一般的家庭使用這一套系統，即使沒有專業的護士協助，也可以在家自行量測視力並取得正確的量測結果。

貳、研究目的

本研究希望可以在電腦螢幕上顯示標準的視標 E，並利用人工智慧的影像辨識來辨識待測人員回答的手勢方向，達到自動判斷待測者的回答是否符合正確的視標 E 缺口方向並將標準視力的量測規則寫入程式判斷中，讓系統自行判斷下一個要顯示的測試視標。完成視力量測之後，進一步將結果上傳到雲端資料庫，使醫生及受測者皆可透過電腦或手機 App 查詢量測結果，期望可以達到不需要量測人員指導及判斷，即可更簡便地進行視力量測。研究中將探討下列各項研究目的：

一、利用視力量測定義編寫出自動產生視標 E 之程式碼。

二、使用 Yolov3 進行 AI 模型訓練並判讀待測者手勢方向。

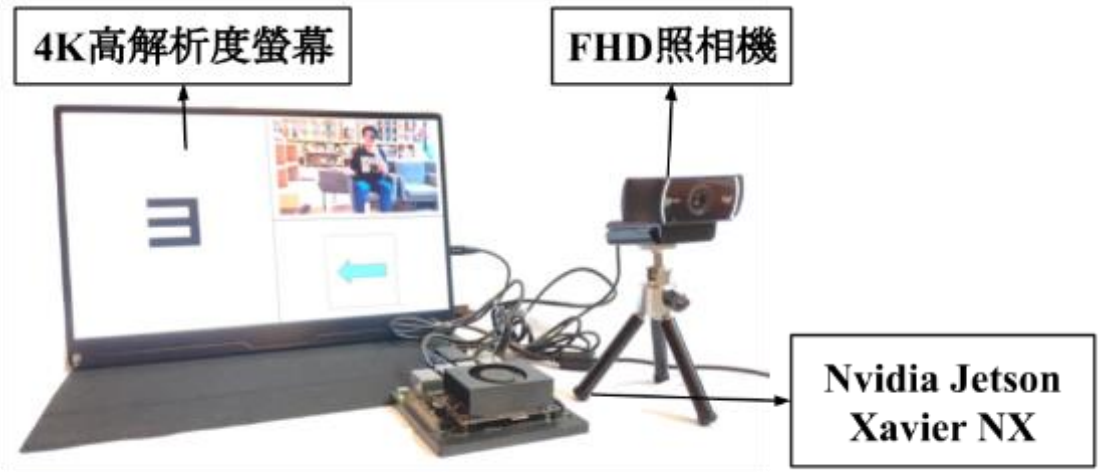
三、利用 python3 程式語言將視力量測規則程式化。

四、將系統搭載在 Nvidia Jetson Xavier NX 邊緣運算系統並進行裝置整合。

五、針對本系統實際量測視力值與傳統檢測值進行誤差分析。

六、利用 google sheets 製作雲端資料庫並以 App inventor2 和 python3 製作可查詢視力的 App。

參、研究設備及器材



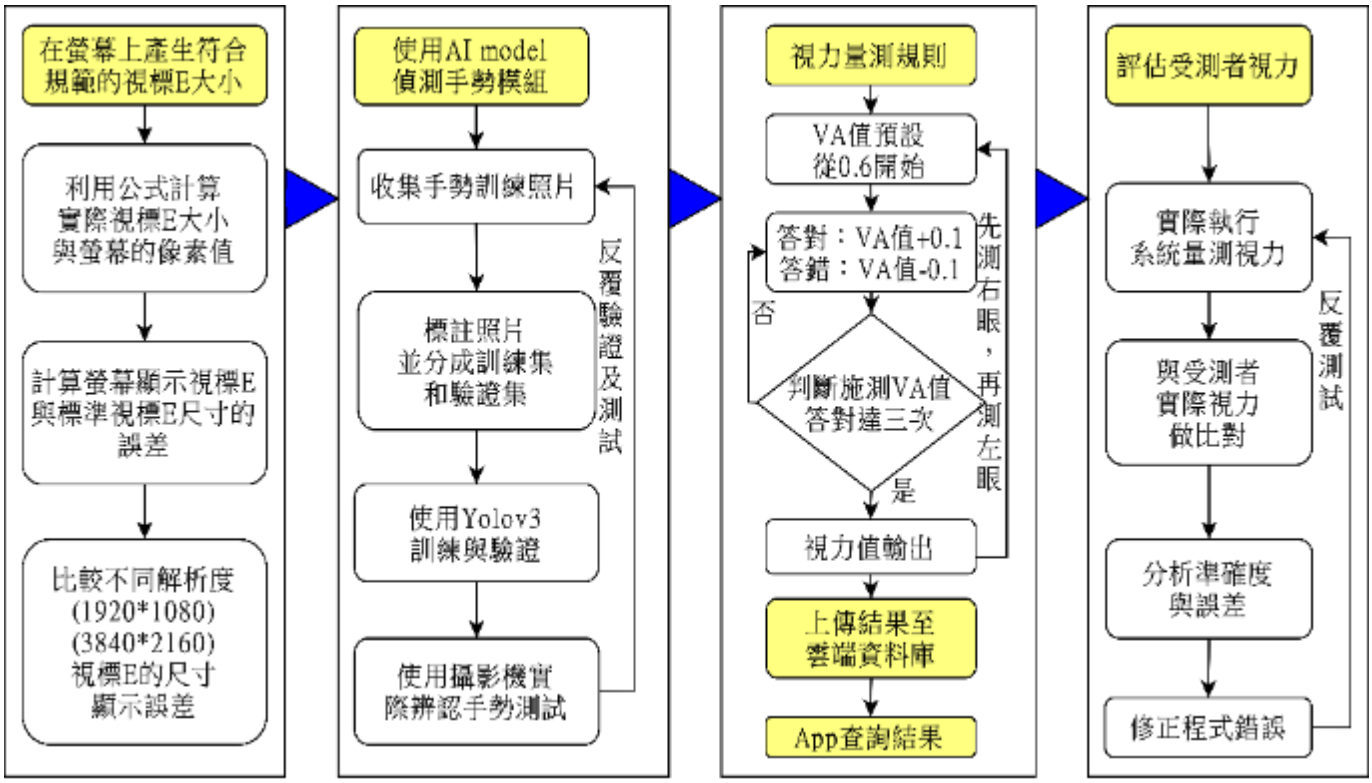
圖一、研究設備及器材配置

表一、研究設備及器材整理表

項目	規格或型號	用途
4K 螢幕	物理尺寸：15 吋 解析度：3840×2160	支援解析度 3840×2160 和 1920×1080
USB WEB 攝影機	Logitech C922 Pro	擷取影像 支援解析度 1920×1080
AI 邊緣運算平台	Nvidia Jetson Xavier NX	將系統整合、AI 運算平台
電競筆電	ASUS GL503V	撰寫程式、報告、進行實驗
Android 手機	Oppo R15 Pro	進行 App 開發及測試
AI 模型	Yolov3	AI 影像辨識模型
電腦程式編譯器	Python3	編譯程式碼
電腦文件編輯器	Notepad++	撰寫程式
影像處理函式庫	Opencv	進行影像處理
手機 App 製作編輯器	App Inventor2	製作手機 App

肆、研究過程或方法

本研究的研究過程及方法整體實現方式如下列圖二所示，圖二中黃色部分為研究目的，研究中首先針對須於螢幕上產生符合規範的視標 E 大小進行設計，接著使用 AI model 偵測手勢模組，並配合視力量測規則建立系統，實際量測後將受試者視力值匯至雲端資料庫，並設計視力查詢 App 以建立即時視力追蹤機制。



圖二、研究過程及方法流程圖

一、利用視力量測定義編寫出自動產生視標 E 之程式碼

（一）求出視標 E 實際尺寸大小

要量測視力，要先有符合規範的視標 E 的尺寸大小，所以本研究需要一個可以求出標準視標 E 大小的方法。根據視力量測的定義：「1.0 的視力值代表受測者可『解析』視網膜上 1 分角大小的成像（許明木，2017）。」，也就是說：「當視力 VA(visual acuity)＝1.0 時，待測距離 6 公尺，可看清一弧分角度的距離大小（一弧分＝$\frac{1}{60}$度）」，所以根據該視力量測定義及圖三之示意，本研究可經由以下步驟推導出標準視標 E 尺寸大小的公式（L 為量測距離(mm)，X 為視標大小(mm)）：

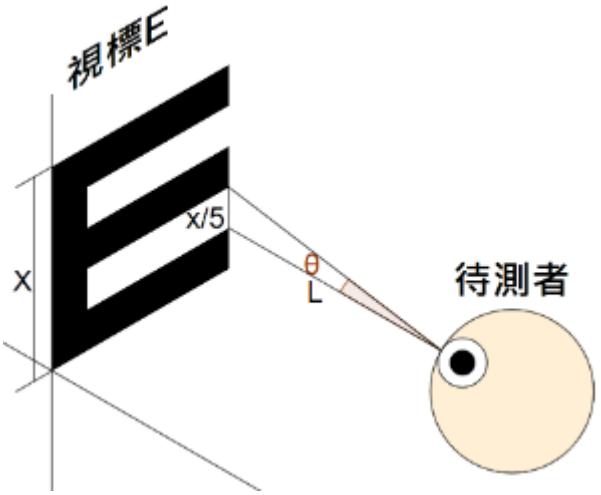
$$\tan\theta = \frac{\left(\frac{x}{5}\right)}{L}$$

$$\tan\theta \times L = \frac{x}{5}$$

$$x = \tan\theta \times L \times 5$$

$$\theta = \frac{1}{60 \times VA}$$

$$\therefore x = tan\left(\frac{1}{60 \times VA}\right) \times L(mm) \times 5$$



圖三、視標與檢查距離關係圖

有了上述公式之後，便可彈性的根據不同待測距離求出相對應的視標 E 尺寸大小，本研究為使居家使用時能有彈性調整之空間，在程式設定中，可針對不同之施測空間進行定點距離輸入，於視力實際量測時分別可採用待測距離為兩公尺及三公尺。

（二）算出視標 E 在螢幕上所需要的像素(pixel)

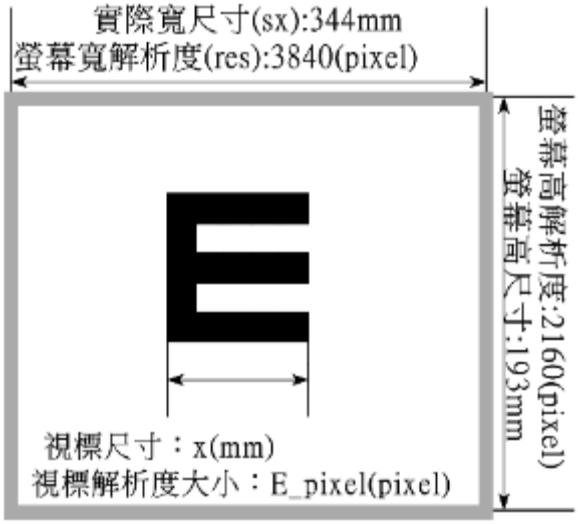
由於各種螢幕的實際尺寸及實際解析度可能有所不同，所以算出視標 E 實際大小之後，需要進行螢幕顯示所需的像素大小的換算。根據螢幕的實際大小和顯示的解析度，可利用其比例關係推導出下列公式（X 為實際視標 E 大小(mm)，res 為螢幕的寬解析度，sx 為螢幕寬的尺寸，E_pixel 為顯示在螢幕上的視標 E 的大小(pixel)）：

$$E\_pixel : x = res : sx$$

$$E\_pixel \times sx = x \times res$$

$$E\_pixel = x \times \frac{res}{sx}$$

如圖四所示。求出 E_pixel 後，即可用電腦程式 python 及 openCV 在電腦螢幕上，根據不同的待測距離，如本研究採用的兩公尺及三公尺距離，畫出各個視力值的標準視標 E 大小。



圖四、螢幕實際尺寸與解析度

二、使用 Yolov3 進行 AI 模型的訓練並判讀待測者手勢方向

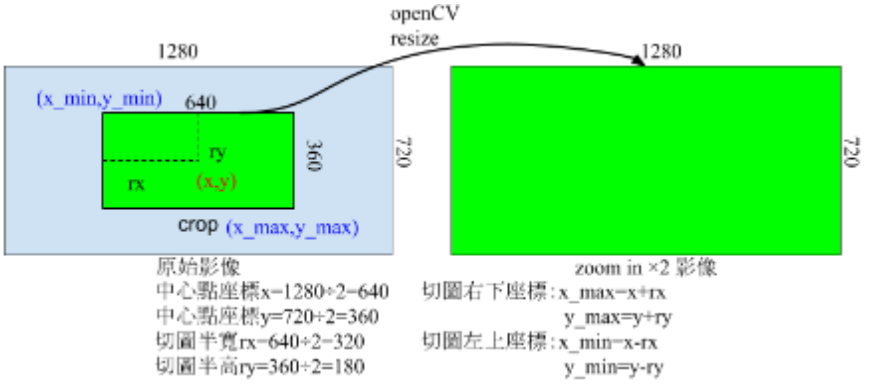
在螢幕上可顯示出標準視標 E 大小之後，可初步使用鍵盤進行視標 E 切換測試，但由於使用鍵盤較為不方便，因此本研究使用 Yolov3 AI model 來判斷待測者手勢取代鍵盤操控。

Yolov3 本身為一個可利用影像辨識多種物件的 AI model，但是其預設可辨識的物件種類不適合本研究使用，本研究需要的是可辨識手勢的 AI model，故將 Yolov3 重新訓練成可辨識手勢的 AI model，而訓練 AI model 必須先收集手勢的訓練照片，接著對收集之照片進行標註，反覆訓練及驗證模型以確保達成足夠之判讀準確度，以下分列三點進行描述：

（一）收集訓練照片

本研究在收集訓練照片時，考量到手勢出現的背景複雜度，可能會影響到 AI 模型的辨識效果，所以在收集照片時，會針對不同衣服顏色的手勢背景進行廣泛收集，並達到符合規定的測試環境光源（許明木等人，2017）。

另外在收集照片時，由於一般的 USB 攝影機之鏡頭配置多為廣角鏡頭，使得遠距拍攝時被拍攝物體會顯得過小，即使本研究已經把測試距離拉到三公尺及兩公尺，受測者比的手勢在鏡頭內之影像仍會很小，不利 AI 辨識。所以本研究模擬數位變焦的方式，把待測手勢拉近兩倍。其作法是：先取出原圖的中心點(x，y)，原圖的寬與長設定為 1280×720(pixel)。再將原圖寬與長除以 2，得到裁切圖的寬與長大小(640×320)（如圖五左邊綠色區塊），接著定位左上角(x_min，y_min)及右下角(x_max，y_max)的點的座標，用 openCV 將裁切框擷取出來，再利用 resize()函式，將它放大兩倍，放大回原圖的寬與長，這樣就可獲得放大兩倍的視野（如圖五右邊綠色區塊）。如此在收集訓練資料時，即以這樣的放大倍率做圖像資料收集，當 AI 模型訓練好之後，做推論時，也是以這樣的放大倍率做手勢辨別，有助於提高 AI model 的辨識率。



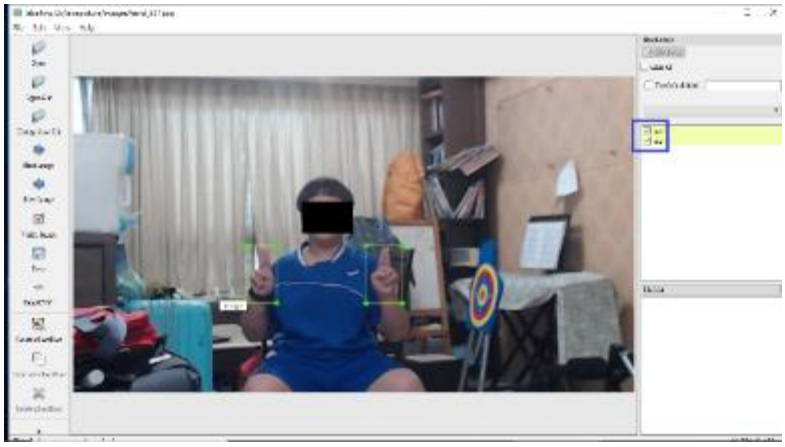
圖五、放大公式解釋圖

（二）對訓練照片進行標注

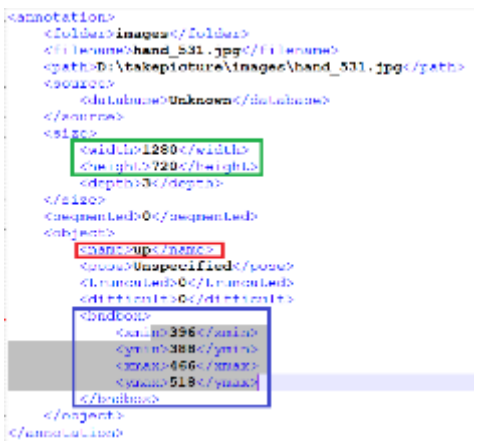
在收集大量的圖片資料後，必須進行圖片資料的標註以便進行訓練。因為訓練 AI model 必須先準備有標準答案的資料讓 AI 進行學習，也必須保留一部分資料做為測試。所以本研究請不同受試者進行照片樣本收集，並使用 labelimg 軟體工具進行標註。labelimg 軟體工具為一開源免費可進行圖片標註的工具，如下圖六所示，圖中兩個綠色的小框即為人工標註的辨識範圍，而右邊藍色框列的「up」即為標籤類別名稱。

而圖七為標註好照片 labelimg 工具存下來的.xml 檔，綠色框代表的是圖片原始大小的寬與高為 1280×720 pixel，而紅色框內的「up」即為標籤的類別名稱，藍色框則代表的是標註物件的邊界框座標及範圍。

當將這張照片送進 AI model 訓練時，即是告知 AI model 這張圖片裡有兩個手勢代表「up」的類別，讓 AI model 可以進行學習，而沒標註到的範圍 AI model 都將視為背景。



圖六、labelimg 進行標註的圖片



圖七、標註後的標籤檔

如圖八，本研究對每個人拍攝六種手勢，包含手勢上下左右，分別代表四種方向、手勢五代表不知道、手勢 ok 代表下一個，不區分左右手，所以共有六種類別(class)，總共收集了 631 張照片，每一張照片標註左右手兩個標籤，因此總共有 1262 個標籤可供 AI model 進行訓練及測試。

圖八、標註訓練照片（分別為上、下、左、右代表方向、五代表不知道、ok 代表下一個）

（三）使用 Yolov3 模型進行訓練及驗證

本研究使用官方版的 Yolov3 開源程式碼進行 AI model 訓練。硬體設備則使用具有 GPU GTX 1070、8GB 顯卡記憶體的電競筆電來進行 AI model 的訓練。將本研究收集到的 631 張訓練照片之 80%設定為訓練數據集、20%則設定為驗證照片數據集，並讓 AI 進行 10000 個週期訓練，訓練時間約需 8 至 10 個小時，訓練完後可獲得用來辨識手勢的 AI model。而遵循 Yolov3 程式碼官網所描述之訓練步驟(Alexey, 2020)，對於本研究的 AI model 進行測試及訓練的步驟如下：

1. 下載 Yolov3 預訓練好的權重檔(darknet53.conv.74)

2. 修改 Yolov3.cfg 檔，yolov3.cfg 為訓練神經網路的設定檔

   (1) 設定訓練圖片批次大小 batch=8

   (2) 設定訓練圖片子批次大小 subdivisions=8

   (3) 設定最大訓練週期 max_batches=10000

   (4) 設定神經網路輸入大小，width=608，height=608

   (5) 設定類別數量 class=6

3. 準備本研究辨識手勢的圖片檔及標籤檔

4. 執行以下指令，進行本研究的 AI model 訓練

   指令碼：./darknet detector train data/obj.data yolov3.cfg darknet53.conv.74 -map

   (1) ./darknet detector 為 Yolov3 的主程式

   (2) 參數 train 為進行訓練

   (3) data/obj.data 為放置訓練圖片集及標籤集的目錄

   (4) Yolov3.cfg 為訓練及測試神經網路的設定檔

   (5) darknet53.conv.74 為原始 Yolov3 預訓練好的權重檔

   (6) map 為訓練過程計算出的平均精準度

5. 訓練完成之後會得到新的權重檔(.weights)

接著使用訓練好的 AI model 進行手勢辨識，執行下列指令，可開啟網路攝影機進行手勢辨識，操作步驟如下：

指令碼：./darknet detector demo cfg/obj.data cfg/yolov3.cfg yolov3.weights -c 0

(1) ./darknet detectorA.為 Yolov3 的主程式

(2) .data/obj.data 為放置訓練圖片集及標籤集的目錄

(3) cfg/Yolov3.cfg 為訓練及測試神經網路的設定檔

(4) Yolov3.weights 為訓練好手勢辨識的權重檔

(5) -c 0 為使用 USB WEB 攝影機進行影像輸入

三、利用 python3 程式語言將視力量測規則程式化

根據視力量測規則，受測者須比出視標 E 的缺口方向，回答正確就跳下一排，錯誤就往上，同一排裡面最多測五次，答對數過半，即可得受測者之視力值。舉例來說：當受測者右眼受檢時 0.5 那一列全對、0.6 那一列答對 3 個（過半數過關）、0.7 那一列答對 2 個（沒過半數，不過關），則該受測者右眼的視力值記錄為 0.6，即以比出過半數之最小列視標記錄為視力值（黃麗恩，2009）。

本研究將上述的視力量測規則轉換成量測視力的 python 程式，以及配合 AI model 的手勢辨識功能，即可讓系統自動判斷及切換下一個視標 E。

四、將系統搭載在 Nvidia Jetson Xavier NX 邊緣運算系統並進行裝置整合

Nvidia Jetson Xavier NX 為一個邊緣運算裝置為小尺寸模組系統，只需消耗 10 或 15 瓦能源就可以開發出多模型人工智慧應用程式，高達 21 兆次運算的加速運算（21TOPS），可提供足夠效能，讓使用者並行多個現代類神經網路，及處理多個高解析度感應器的資料，滿足完整人工智慧系統的需求(Nvidia, 2019)。

本研究將在電競筆電上訓練好的 AI 模型及程式移植到 Jetson NX，將原本 FP32 的模型利用 Nvidia TensorRT 轉成 FP16 的模型，可從 4~5FPS 達到 8~10FPS，達到加快推論的速度的目的(Yqlbu, 2020)。

五、針對本系統實際量測視力值與傳統檢測值進行誤差分析

整個系統開發完成後，本研究接著會進行系統之實際視力量測值與評估，並計算系統量測結果和醫院量測結果的誤差值。

量測視力前，先確認環境燈光充足（須為 700 燭光以上），本系統可依施測場地之需求，輸入待測距離（三公尺或兩公尺），且待測者眼睛高度必須和螢幕顯示器同高。在邀請受試者協助量測時，會於量測前紀錄最新一次醫院量測結果，以便探討誤差值。在分析誤差方面，本研究將資料分成學齡階段（低於 18 歲）和成年階段（高於 18 歲（含）），並針對誤差值進行分析。

六、利用 google sheets 製作雲端資料庫並以 App inventor2 和 python3 製作可查詢視力的 App

為了方便將視力量測試結果儲存及查詢，並且達到隨時隨地都可查詢視力紀錄的目的，本研究需要一個網路資料庫。而 google sheets 是一個免費且使用簡單的表單式資料庫，故本研究利用 google sheets 當成視力量測結果的網路資料庫，當系統實際量測時會自動將視力的結果上傳到該資料庫。

有了上述網路資料庫，本研究便可利用 App Inventor 2 開發讓受測者可以觀察或追蹤視力的手機 App，並將此 App 命名為：「視力追蹤 App」，受測者可下載該手機 App 並透過網路隨時隨地的查詢視力結果及追蹤視力變化記錄。

此外，研究中亦使用 python3 開發出可讓醫生方便瀏覽所有受測者視力量測結果的電腦端 App，在使用上較手機 App 多了更多介面與功能，且電腦端的 App 有較大的畫面顯示，可讓醫護人員方便查詢瀏覽所有受測者的視力結果，並進行資料統計及分析。

伍、研究結果

一、利用視力量測定義編寫出自動產生視標 E 之程式碼

本研究利用視標 E 的公式，可彈性設定待測距離為三公尺或兩公尺，產生出的視標 E 在螢幕上顯示如圖九所示。根據不同的待測距離及不同的 VA 值可分別算出不同的視標 E 大小。本研究使用 15 吋的 4K 螢幕呈現視標 E，並且在表二計算出視標 E 顯示在螢幕上實際所占的 pixel 數值及實際尺寸大小。例如待測距離為三公尺，VA=0.1 所產生的視標 E 大小，實際占用螢幕 487 個 pixel，而且顯示出的實際尺寸大小為 43.62mm（標準值為 43.63mm）。此螢幕顯示的尺寸大小與標準的視標 E 尺寸仍有誤差，不過誤差很小，平均各個 VA 的視標 E 尺寸大小的誤差在 4K 螢幕上只有 0.72%；而待測距離兩公尺時，誤差為 0.81%。

圖九、使用公式產生的所有視標 E 圖片

表二是使用本系統在 15 吋的 4K 螢幕上產生出來的各個視標 E 在待測距離為三公尺及兩公尺時，實際在螢幕上所佔的 pixel 數值以及顯示尺寸大小。本研究之視力量測程式編寫附錄三、產生所有視標 E 的程式。

表二、視標 E 在螢幕上所佔的 pixel 數值及實際顯示尺寸大小

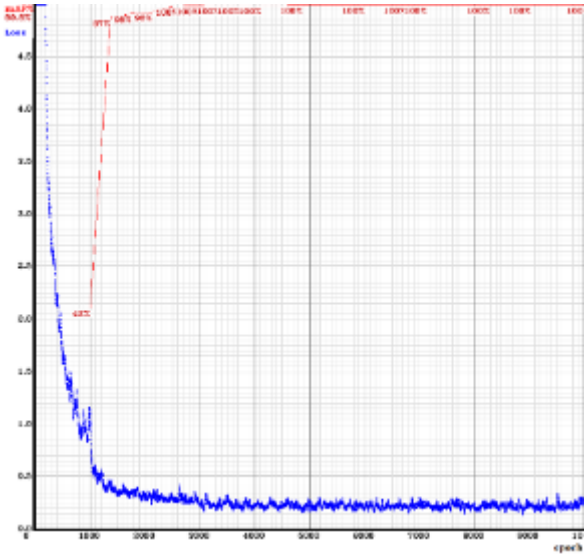
視力值(VA)	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	1.2	1.5	
待測距離三公尺標準(mm)	43.63	21.82	14.54	10.91	8.73	7.27	6.23	5.45	4.85	4.36	3.64	2.91	平均誤差 0.72
螢幕顯示視標 E 像素數(pixel)	487	243	162	121	97	81	69	60	54	48	40	32	
螢幕實際顯示視標 E 大小(mm)	43.63	21.77	14.51	10.84	8.69	7.26	6.18	5.38	4.84	4.30	3.58	2.87	
誤差(%)	0.01	0.22	0.22	0.63	0.42	0.22	0.84	1.45	0.22	1.45	1.45	1.45	
待測距離兩公尺標準(mm)	29.09	14.54	9.70	7.27	5.82	4.85	4.16	3.64	3.23	2.91	2.42	1.94	平均誤差 0.81
螢幕顯示視標 E 像素數(pixel)	324	162	108	81	64	54	46	40	36	32	27	21	
螢幕實際顯示視標 E 大小(mm)	29.03	14.51	9.68	7.26	5.73	4.84	4.12	3.58	3.23	2.87	2.42	1.88	
誤差(%)	0.22	0.22	0.22	0.22	1.45	0.22	0.84	1.45	0.22	1.45	0.22	2.99	

二、使用 Yolov3 進行 AI 模型訓練並判讀待測者手勢方向

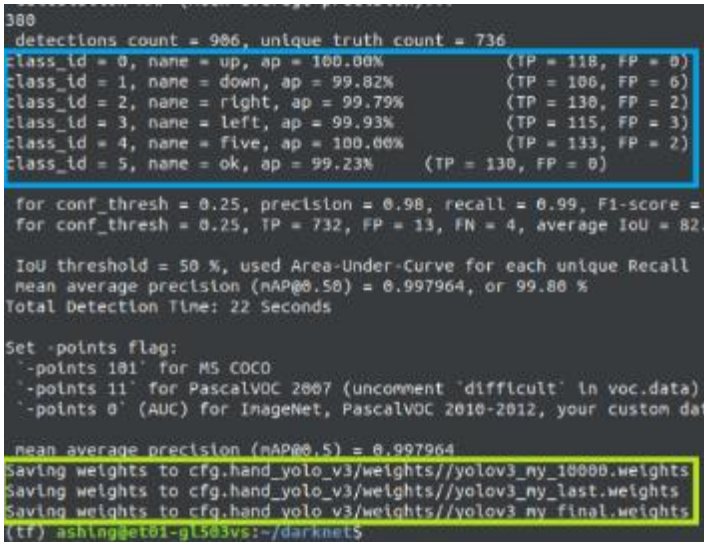
使用官方版的 Yolov3 進行本研究的 AI model 訓練結果如下：

在執行訓練程式時便會出現下圖十，顯現出訓練過程的 loss（藍色線），及測試數據集的各類別平均精準度 mAP（紅色線），由此圖可看出此本研究 AI model 訓練到最後，各類別平均精準度 mAP 可達到 99%以上。由於 Yolov3 是一個成熟的 AI model，本身的模型架構已有考慮並解決過度適合（overfiting）的 AI model 訓練的問題，所以本研究在應用上只考慮測試數據集的各類別平均精準度 mAP，由於 mAP 已達 99%，故可信賴該 AI model 已訓練成功，可實際應用。而實際在使用 Yolov3 辨識手勢時，本研究會判斷連續三次偵測到同一手勢方向，才確認為該手勢方向，可再減少手勢偵測被誤判的可能性。

圖十一為 AI model 訓練結束時在電腦終端機顯現出的資訊，藍色框內的訊息為 AI 訓練過程對各個類別「up」，「down」，「left」，「right」，「five」，「ok」進行平均準確度 mAP 的評估。AI model 訓練完成之後，如圖十一下列綠色框的訊息，產生的 yolov3_my_10000.weights 檔案即為 AI model 所需要的權重檔，使用此權重檔即可進行本研究系統的手勢辨識。



圖十、Yolov3 訓練過程 loss 及 mAP

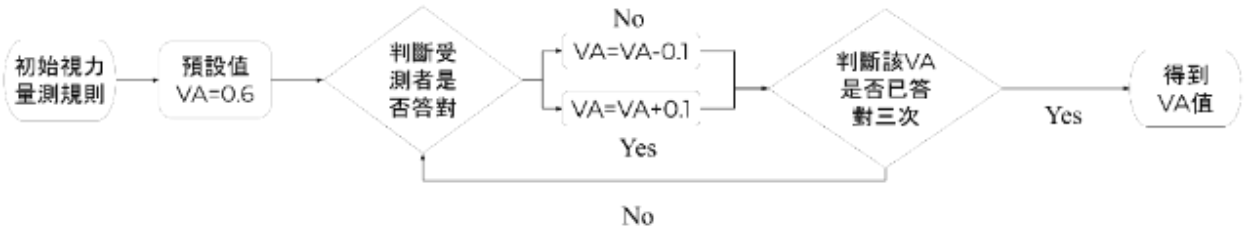


圖十一、Yolov3 訓練過程顯示的訊息

三、利用 python3 程式語言將視力量測規則程式化

如同在醫院或診所傳統手比的量測方式，圖十二為視力量測規則的流程圖。視力量測的規則為：正確就跳往視力表的下一排，如同本程式顯示下一個(VA+0.1)的視標 E，而錯誤就往回視力表上一層跳(VA-0.1)，視力表中同一排視標裡，最多施測五次，答對三個過半的 VA 值即為受測者的視力。

而在本研究中，為方便居家彈性空間調整使用，增設定點距離輸入，使用者可因應不同的施測環境空間大小，自行設定施測距離為三公尺或兩公尺，以讓系統呈現出適合該施測距離的視標 E 來進行施測。本研究之視力量測程式編寫於圖十三。

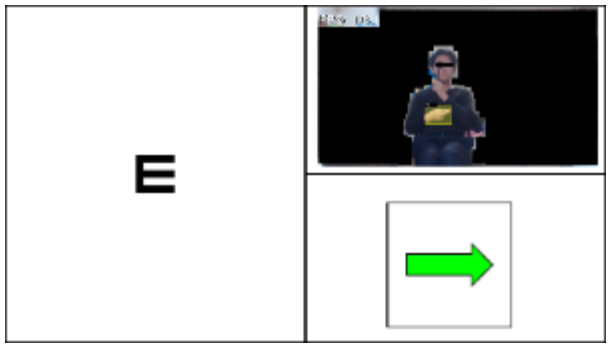


圖十二、視力量測規則流程圖

圖十三、視力量測規則程式碼

四、將系統搭載在 Nvidia Jetson Xavier NX 邊緣運算系統並進行裝置整合

將前面所述的軟體功能使用 python 開發完成後，為了系統小型化，並降低成本及考量系統功耗。本研究將系統軟體程式從電競筆電整合到 Nvidia Jetson Xavier NX 上執行，圖十四為本系統執行時的縮圖，左邊區域顯示出標準視標 E，右上角利用 opencv 開啟 USB 攝影機擷取影像，並同時顯示 Yolov3 AI model 手勢偵測的狀態，右下角顯示出 AI model 手勢辨識出的方向，讓受測者確認所比方向是否與 AI model 辨識結果一致，經過上述軟硬體整合後，便可開始進行實測（如圖十五）。



圖十四、系統軟體執行時的畫面縮圖

圖十五、系統實際測試的圖片

五、針對本系統實際量測視力值與傳統檢測值進行誤差分析

當完成系統後，本研究進行系統的實際量測，燈光環境為一般室內充足的日光燈光源，待測距離為三公尺及兩公尺，先測右眼再測左眼，右眼量測完時，會先休息 20 秒再進行左眼測試。測定受測者視力值後，再將所測得之受測者平均視力值及誤差值進行計算，在量測之前先記錄受測者最近一次在醫院診所的量測值，以利進行比對分析。

在待測距離為三公尺時總共有效量測人次為 38 次，施測年齡分布在 6 至 44 歲（如表三）；而在待測距離為二公尺時總共有效量測人次為 11 次，施測年齡分布在 12 至 43 歲（如表四）。根據測量結果，無論是在待測距離三公尺或兩公尺皆可獲得不錯的準確率，因此使用系統時，可彈性調整待測距離來因應場地空間不同的問題。

表三、待測距離三公尺的誤差比較表

年齡層分布	人次	醫院量測		系統量測		平均誤差			總誤差	準確率
		左眼	右眼	左眼	右眼	左眼	右眼	平均		
學齡階段	24	1.063	1.129	1.113	1.175	0.092	0.113	0.102	0.097	**96.645**
成年階段	14	0.843	0.936	0.914	0.886	0.114	0.064	0.089		

表四、待測距離兩公尺的誤差比較表

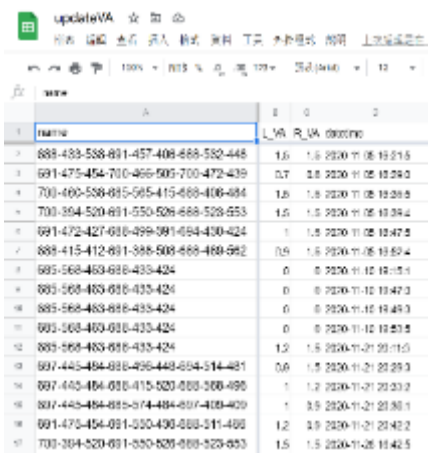
年齡層分布	人次	醫院量測		系統量測		平均誤差			總誤差	準確率
		左眼	右眼	左眼	右眼	左眼	右眼	平均		
學齡階段	5	0.960	1.000	1.040	0.960	0.150	0.100	0.125	0.084	**98.014**
成年階段	6	0.960	1.000	0.850	0.933	0.080	0.020	0.050		

六、利用 google sheets 製作雲端資料庫並以 App inventor2 和 python3 製作可查詢視力的 App

在進行實際量測時，本研究會自動將量測結果上傳至雲端資料庫。google sheets 為開源、簡單、方便使用的輕量資料庫，使用其當作雲端資料庫，配合本研究所製作的 App 便可隨時隨地透過網路查詢視力量測的結果。

圖十六為雲端資料庫的存取畫面，為防止個資外洩，左邊欄位為姓名加密過後的數值，中間兩個欄位為左眼量測視力的結果及右眼量測的結果，右邊的欄位為量測完畢時的日期及時間。圖十七為 python 撰寫的電腦 App，可供醫護人員使用。圖十八則為利用 App Inventor 所撰寫的手機 App，可供受測者使用，而電腦版視力查詢 App 之詳細程式碼如附錄一。

在兩種不同 App 介面的使用上，最大差異在於電腦端的 App 有較大的畫面顯示，可讓醫護人員方便查詢瀏覽所有受測者的視力結果，並進行資料統計及分析；而手機 App 則可供受測者方便、迅速、及時的查詢視力量測結果，而手機版視力查詢 App 之詳細程式碼如附錄二。



圖十六、雲端資料庫

圖十七、電腦 App 查詢視力

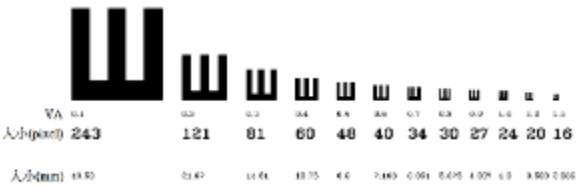
圖十八、手機 App 查詢視力

陸、討論

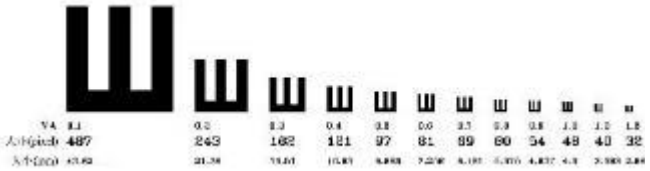
一、探討標準視標 E 和螢幕顯示出來的視標 E 有誤差的原因

（一）比較不同解析度顯示視標 E 大小的誤差

由於電腦螢幕顯示的大小單位是用 pixel 為單位，且這個值須為正整數，所以顯示出來的視標 E 和標準視標 E 大小會有誤差，為了進一步研究這個誤差，本研究使用 15 吋 1920×1080 解析度螢幕及 15 吋 3840×2160 解析度螢幕來探討誤差。



圖十九、1920×1080 的螢幕產生出來的視標 E



圖二十、3840×2160 的螢幕產生出來的視標 E

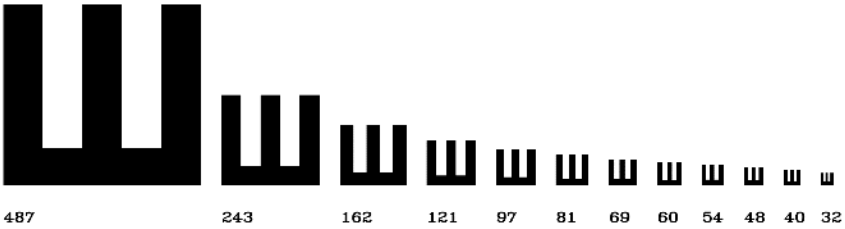
表五是計算兩個螢幕顯示的視標 E 尺寸誤差和平均誤差百分比，由表五可得知在解析度 1920×1080 的解析度下平均誤差為 1.15%，在 3840×2160 的解析度下平均誤差為 0.79%，故本研究所採用 4k 的解析度，以更貼近實際公式解的視標 E 尺寸。

表五、不同解析度的螢幕顯示視標 E 大小(mm)和標準視標 E 大小(mm)的誤差和誤差平均值

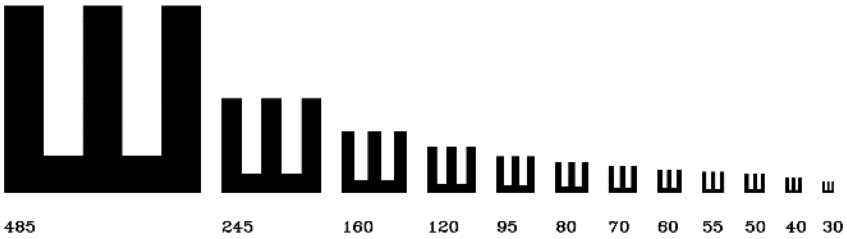
視力值 (VA)	標準大小 (mm)	1920×1080 的螢幕顯示視標 E 尺寸(mm)	誤差(%)	3840×2160 的螢幕顯示視標 E 尺寸(mm)	誤差(%)
0.1	43.633	43.538	0.220	43.627	0.014
0.2	21.817	21.679	0.630	21.769	0.219
0.3	14.544	14.513	0.219	14.513	0.219
0.4	10.908	10.750	1.451	10.840	0.630
0.5	8.727	8.600	1.451	8.690	0.425
0.6	7.272	7.167	1.451	7.256	0.219
0.7	6.233	6.092	2.273	6.181	0.835
0.8	5.454	5.375	1.451	5.375	1.451
0.9	4.848	4.838	0.219	4.838	0.219
1.0	4.363	4.300	1.451	4.300	1.451
1.1	3.967	3.942	0.630	3.942	0.630
1.2	3.636	3.583	1.451	3.583	1.451
1.3	3.356	3.225	3.915	3.315	1.246
1.4	3.117	3.046	2.273	3.046	2.273
1.5	2.909	2.867	1.451	2.867	1.451
		平均誤差	1.369	平均誤差	0.849

（二）討論顯示視標 E 會有粗細大小不同的問題

根據標準視標 E 的規範，視標 E 的大小需為 5×5 的正方形，所以換算出來的視標 E 大小的 pixel 值如果不是 5 的倍數就會產生粗細不同的現象（如圖二十一），因此本研究使用另一種方法來修正視標 E 粗細不同的問題，將非 5 的倍數的 pixel 值設定為最接近 5 的倍數的值，並重新計算與標準視標 E 的誤差，結果可發現，雖修正粗細大小不同的問題（如圖二十二），但由於有額外增加 pixel 的誤差值，導致總平均視標 E 誤差大小也會增加，例如：原本 VA=0.1，視標 E pixel 值為 487，經過粗細大小不同修正後變為 485，額外增加了兩個 pixel 的誤差值。考量在實際施測視力時，視標 E 的大小較粗細度對視力判讀的準確度來的更有影響性，本研究採以使用未修正粗細不同的視標 E 大小進行實驗。



圖二十一、未修正粗細不同的視標 E 樣本圖片



圖二十二、已修正粗細不同的視標 E 樣本圖片

表六、比較已修正與未修正粗細不同的視標 E 尺寸大小誤差

視力值 (VA)	標準大小 (mm)	pixel	4K 未修正粗細不同的視標 E 尺寸(mm)	誤差(%)	pixel	4K 已修正粗細不同的視標 E 尺寸(mm)	誤差(%)
0.1	43.633	487	43.627	0.014	485	43.448	0.425
0.2	21.817	243	21.769	0.219	245	21.948	0.602
0.3	14.544	162	14.513	0.219	160	14.333	1.451
0.4	10.908	121	10.840	0.630	120	10.750	1.451
0.5	8.727	97	8.690	0.425	95	8.510	2.478
0.6	7.272	81	7.256	0.219	80	7.167	1.451
0.7	6.233	69	6.181	0.835	70	6.271	0.602
0.8	5.454	60	5.375	1.451	60	5.375	1.451
0.9	4.848	54	4.838	0.219	55	4.927	1.628
1.0	4.363	48	4.300	1.451	50	4.479	2.655
1.2	3.636	40	3.583	1.451	40	3.583	1.451
1.5	2.909	32	2.867	1.451	30	2.688	7.611
			平均誤差	0.716		平均誤差	2.138

二、探討使用網路資料庫會產生的資安問題

使用 google sheets 網路資料庫，雖然有簡單、方便、且免費等優點，但其缺點卻是，需要先把網路資料庫設為公開，才可將結果上傳上去，也就是只要任何人知道該 google sheets 網路資料庫的網址，便可瀏覽結果。因此會產生個資外洩的問題，為了解這個個資外洩的問題，本研究將系統設計成先把姓名加密，加密完後才會將結果上傳到雲端資料庫。而將結果下載到本研究製作的 App 時再進行解密，便可顯示正確的受測者姓名。

在做姓名加密的時候，會使用 python 將姓名的字串資料轉換成 unicode 編碼，並使用簡單的線性轉換，代入公式$y = ax + b$（x 為未加密的 unicode 編碼，y 為已加密的 unicode 編碼），得到加密的值，這個值會被存到 google sheets 網路資料庫，取代原本的姓名資料。

而在 App 端做姓名解密時，會先把值代入公式$\frac{y-b}{a} = x$，x 為復原的 unicode 編碼，再將其轉換回字串資料的姓名，即可完成解密，App 端即可顯示正確的姓名。

三、探討使用 App Inventor2 製作 App 的問題

本研究使用 App Inventor2 製作使用者可查詢視力量測結果的 App，必須從雲端資料庫中下載加密過後的姓名資料，然後在 App Inventor2 作解密還原的動作，因為 App Inventor2 沒有把 unicode 轉成字串的積木功能，所以本研究製作了可將 unicode 解密，再轉成姓名字串的功能積木。製作自建功能積木的方法如下：

（一）安裝 jdk：jdk 是 Java 開發工具組。

（二）安裝 ant：ant 是用來將寫完的自定義的 Extension 打包成 aix 文件，打包完成以後可以直接導入到 app inventor 中使用。

（三）下載 App Inventor2 源碼：可到 Github 下載程式碼

（四）找到 appinventor-sources-master\appinventor\components\src 這個路徑，創建解密積木的模組。

（五）在此路徑下：appinventor-sources-master\appinventor 執行 ant extensions 命令進行編譯程式及打包。

（六）打包成功後可產生解密積木的模組 aix。

（七）最後在 App Inventor2 中導入該模組 aix，即可使用解密功能的程式積木。

四、探討使用 Yolov3 AI model 辨識手勢的問題

本研究在前期測試使用 Yolov3 AI model 辨識手勢時，一開時發現 AI model 辨識手勢有左右不分的現象，致使左右方向辨識率不佳。經研究後發現在訓練 Yolov3 AI model 的時候，該 model 會將訓練照片作左右翻轉的資料增強，因此會造成左右方向的手勢的類別標籤混淆。解決的方法為當訓練 AI model 時，將 Yolov3 左右翻轉的訓練參數（filp）關掉，再重新訓練，即可避免該問題。

五、探討使用 Nvidia Jetson Xavier NX AI 邊緣運算裝置

本研究使用 Nvidia Jetson Xavier NX 進行系統開發及整合，主要有低功耗、AI 運算力高、體積小、平價等優點。如表七所示，雖然 Jetson Nano 價格較低，但執行 Yolov3 AI model 的效能經實測過後只有 3FPS，未達本研究所需的需求，因為如果一秒內 AI model 只能判斷三次手勢方向是否正確，將會影響整體測試速度，而在本研究中，經實際操作測試一秒內 AI model 至少須能偵測 6~8 次手勢方向的結果，才不會影響整體測試流暢度。相關硬體設備中，以 Jetson Xavier AGX 效能最好，但價格昂貴。故本研究選擇 Jetson Xavier NX 做為系統整合開發平台，經實測過後，其效能可達 8~10FPS，雖未達實時（Realtime（30FPS 的效能，但已達本研究的需求，每秒 AI model 可偵測受測者 8~10 個手勢方向，實測上已可順暢的進行測試。

表七、Nvidia Jetson 系列產品規格比較

	Jetson Nano	**Jetson Xavier NX**	**Jetson Xavier AGX**
			AGX Xavier
AI 效能	472 GFLOPS	21 TOPS	**32 TOPS**
GPU	128-core NVIDIA Maxwell™ GPU	384-core NVIDIA Volta™ GPU with 48 Tensor Cores	**512-core NVIDIA Volta™ GPU with 64 Tensor Cores**
CPU	Quad-core ARM® Cortex®-A57 MPCore processor	6-core NVIDIA Carmel ARM®v8.2 64-bit CPU 6MB L2 + 4MB L3	**8-core NVIDIA Carmel Arm®v8.2 64-bit CPU 8MB L2 + 4MB L3**
記憶體	4 GB 64-bit LPDDR4 25.6GB/s	8 GB 128-bit LPDDR4x 51.2GB/s	**32 GB 256-bit LPDDR4x 136.5GB/s**
功耗	**5W | 10W**	10W | 15W	10W | 15W | 30W
機械規格	**69.6 mm x 45 mm**	**69.6 mm x 45 mm**	100 mm x87 mm
價錢(NT$)	**1650**	13300	23500

六、比較傳統量視力和本系統量視力

本研究進行本系統實際量測後再比較傳統手比量測的差異，整理成表八。在本系統進行實際量測前，需先記錄受測者醫院或診所量測的視力值，以求有正確的比較基準。並且本系統實際量測時，也必須符合醫院或診所的受測環境光源，需有 700 燭光以上的標準。

本研究將醫院或診所量測的視力值當成標準答案，將本系統量測的結果與之作比較，準確度可達 96.645%。在量測所需時間方面，本系統每次量測只約 5 分鐘，而且不需排隊等待，而傳統手比的時間雖然也是約五分鐘，但是必須在醫院或診所排隊等待量測，相較起來本系統彈性程度更佳，可因應不同環境空間選擇不同的測量距離，且準確率高。

而雖然本系統使用 Jetson Xavier NX 比一般視力燈箱單一價格高，但使用簡單不須有醫護人員一直在旁指導，並且可在家即可自行操作使用。

表八、傳統醫院量測和本系統量測的差異比較表

	傳統手比量測	本研究系統
一人次量測時間（分鐘）	5	5
量測準確度（%）	接近 100	96.645
人力耗費（人數）	每個人次皆需 1 人力指導	1 醫護人員可同時啟動 多台機器同時量測
設備成本估計（元）	約 3500（視力燈箱）	22000（本系統）
方便性	須有人指導協助	可在家自行操作量測
量測距離	固定三公尺或六公尺	可彈性選擇兩公尺或三公尺 （居家操作方便）

七、未來展望

綜合比較，本研究將各個版本的系統功能及使用的硬體整理成表九，首先本研究開發第一代系統時，使用鍵盤來當作受測者回答的媒介，以先期確認視標 E 顯示功能正常，接著第二代導入 Yolov3 AI 辨識手勢的功能，並將系統移植到 Jetson NX，接著第三代將訓練好的 Yolov3 進行優化，使辨識效能可達 10FPS，最後第四代導入上傳網路資料庫及增加 App 查詢的功能。

表九、各代系統比較表

	第一代	第二代	第三代	第四代
軟體部分	使用鍵盤操作方向	AI 辨識手勢操作方向	優化 AI 模型	新增視力查詢 App 及網路資料庫功能
硬體部分	使用電競筆電	電競筆電、NX	NX	NX、手機
效能部分	操作不方便	AI 辨識速度 4~5FPS	8~10FPS	8~10FPS

本研究目前主要是開發軟體系統來實現本研究目的，未來在 App 的使用開發上，可再開發出更多的功能，例如電腦端的 App 可提供醫護人員統計及分析受測者視力的功能。在 AI model 偵測的速度上也有機會再進行優化，讓整個測試系統更加順暢。

未來如果有更新更有效率的硬體平台，也可將本研究的開發軟體系統移植到新的硬體平台上，進一步降低系統成本，使系統不受限於使用 Nvidia Jetson Xavier NX 的硬體平台。再者，可將系統功能再擴充，例如可利用 Yolov3 AI model 進行手勢數字辨識，即可增加辨色力異常的檢測功能。

柒、結論

本研究利用 Yolov3 AI model 以及 Nividia Jetson xavier NX 邊緣運算平台實作一 AI 影像辨識輔助視力量測系統，系統開發整合完成後進行人員實際量測視力，並將量測結果與傳統量測方式做比較。研究結論包括：

一、系統硬體部分

本研究將系統實作在 Nvidia Jetson Xavier NX 平台上，並採用 4K 顯示器，以及高解析度相機，整體架構相對於使用電競筆電，可達到**低成本，低功耗，體積小**等優點，經實測過後，效能可達到 8~10FPS，該效能讓受測者進行測試時，不會受限於 AI model 辨識速度，讓整體測試流程更加順暢。

二、系統軟體部分

本研究開發可在電腦顯示器上顯示標準大小的視標 E 的程式，並使用 Yolov3 製作可辨識手勢的 AI model，再配合視力量測規則，達到**無人協助即可進行視力量測**。經實測過後，在待測距離三公尺時，本系統平均誤差大約為 0.096，準確率達 96%；待測距離兩公尺時，平均誤差大約為 0.084，準確度為 98.014%，可根據環境空間**彈性選擇待測距離**。

三、雲端資料庫及 App 應用程式的部分

本研究將系統設計成量測完視力時自動將量測結果上傳到雲端資料庫儲存，並**將姓名加密來保護個資**，並使用 Python3 和 App inventor2 製作視力查詢的 App，可**查詢歷來的視力量測紀錄**。

綜合以上的研究結果，本研究所設計的系統可達到輔助量測視力的成果，可期望減輕醫護人員的工作負擔，尤其在 covid-19 疫情期間無法外出到眼科診所追蹤視力，在一般家庭使用上，可預先檢測視力是否有出現問題，以及追蹤及記錄視力的變化，**達到及早發現，及早治療的預防功能。**

本研究主要是開發軟體系統，目前搭載在 Nvidia Jetson Xavier NX 上，未來如果有更新更有效率的硬體平台，也可將本研究的開發軟體系統移植到新的硬體平台上，進一步降低系統成本。也可將系統功能再擴充，例如可利用 Yolov3 AI model 進行**手勢數字**辨識，即可增加辨色力異常的檢測功能。

捌、參考文獻資料

一、許明木、莊素貞、鄭靜瑩、陳賢堂、王俊諺、吳承臻、林則豪、許淑貞、連政炘、葉志偉（2017）。**低視力學**（2 版）。臺北：五南出版社。

二、衛生福利部國民健康署（2016）。**兒童視力篩檢及矯治指引**。取自 https://www.hpa.gov.tw/Pages/ashx/File.ashx?FilePath=~/File/Attach/1077/File_8248.pdf

三、黃麗恩（2009）。**老年性白內障之視覺模擬測試平台建構**。取自臺灣博碩士論文知識加值系統。

四、Alexey(2020)。**Darknet**。取自 https://github.com/AlexeyAB/darknet

五、Yqlbu(2020)。**TRT-yolov3**。取自 https://github.com/yqlbu/TRT-Yolov3

六、Nvidia(2019)。**NVIDIA 宣布推出 Jetson Xavier NX 全球尺寸最小的邊緣 AI 超級電腦**。取自 https://blogs.nvidia.com.tw/2019/11/06/nvidia-introducing-jetson-xavier-nx/

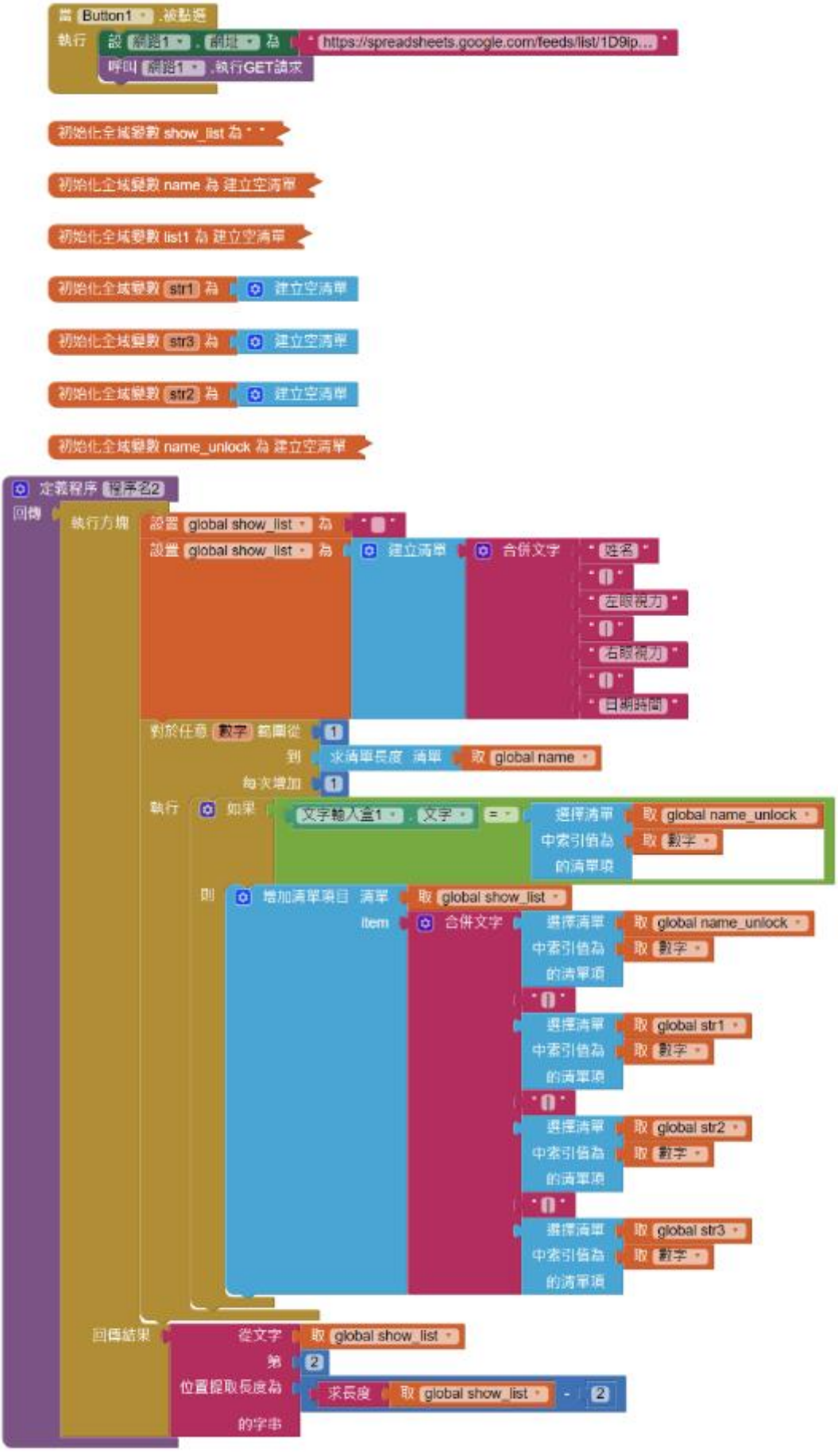
玖、附錄

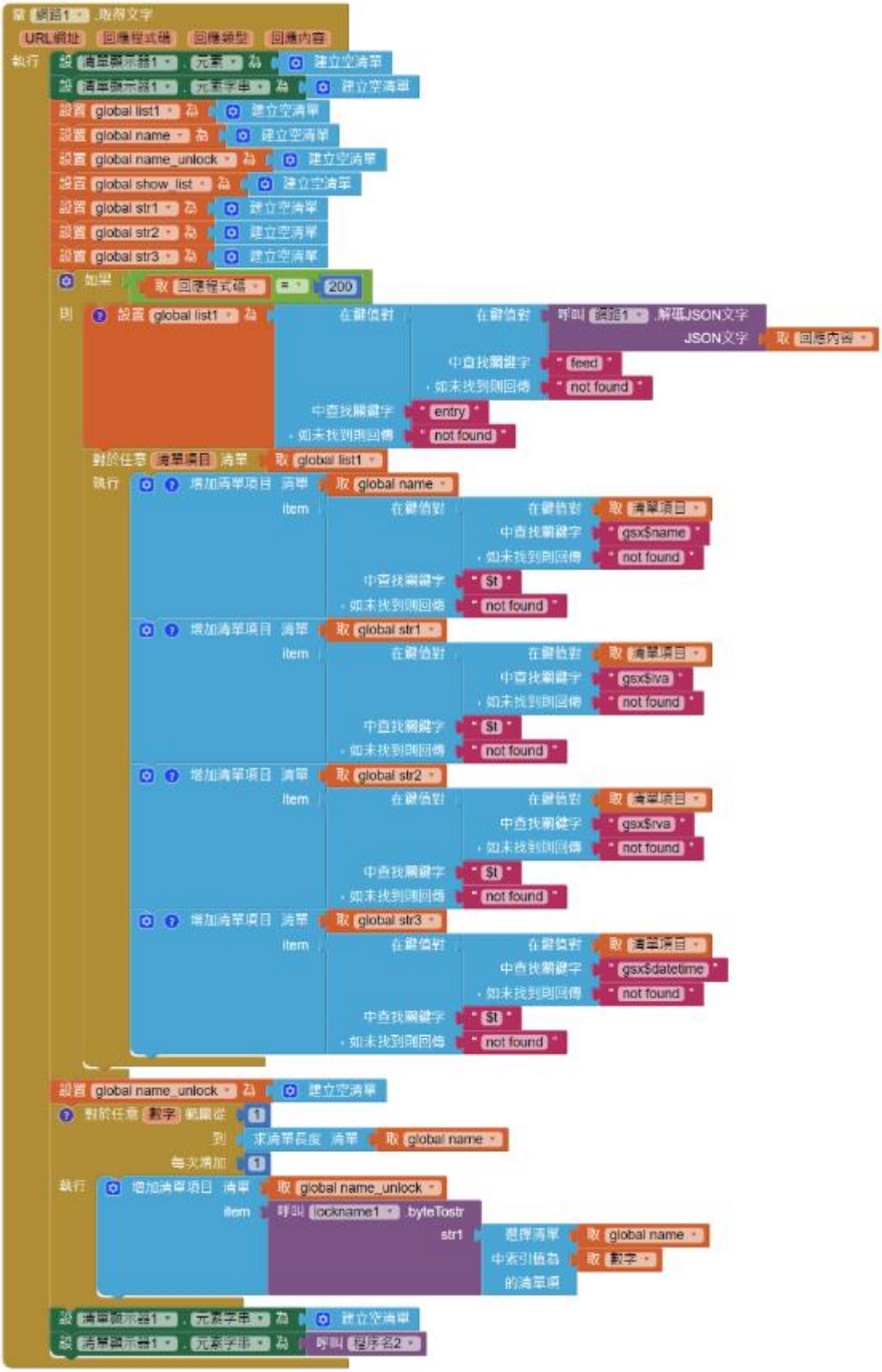
一、電腦版視力查詢 App 程式碼

```
#made at 0601
#new======
#you can scan name

import PySimpleGUI as sg
import sys
import time
import datetime
import gspread
import json
from oauth2client.service_account import ServiceAccountCredentials as SAC
import re

def getdata():
	try:
		GDriveJSON = 'updateVA.json'
		GSpreadSheet = 'updateVA'
		scope =	['https://spreadsheets.google.com/feeds','https://www.googleapis.com/auth/drive']
		key = SAC.from_json_keyfile_name(GDriveJSON, scope)
		gc = gspread.authorize(key)
		worksheet = gc.open(GSpreadSheet).sheet1
		return worksheet
	except Exception as ex:
		print('update fail', ex)
		sys.exit(1)


worksheet=getdata()
list_of_lists = worksheet.get_all_values()
record_len=len(list_of_lists)-1



def	covert2name(name_str):
	name_str_list=name_str.split('-')

	name_str_list=[int((int(i)-1)/3) for i in name_str_list]

	name_bytes = bytearray(name_str_list)
	#print('name_bytes' ,name_bytes.decode("utf8"))
	return name_bytes.decode("utf8")
def double_to_single(list_a,getname=''):
	list_a_len=len(list_a)
	list_b=[]
	for i in range(1,list_a_len):
		#txt = "name	|L_VA	|R_VA	|datetime	"
		cname=covert2name(list_a[i][0])
		string_name=cname+"	"*10
		string_name=string_name[0:8]
		string_lva=list_a[i][1]+" "*20
		string_lva=string_lva[0:20]
		string_rva=list_a[i][2]+" "*20
		string_rva=string_rva[0:20]

		string_va=string_name+string_lva+string_rva+list_a[i][3]
		if(getname=='' or (getname == cname)):
			list_b.append(string_va)
	if(len(list_b)==0):
		list_b.append(("can't find "+getname))
	return list_b

output_str='_OUT_'
#sg.theme('DarkBrown5') #背景設置 DarkBrown5，默認或者這一行不設置為銀河灰
#sg.theme('DarkGreen5')
fsize= 24
layout=[]
layout += [[sg.Input(key='name',font=("Helvetica", fsize)),sg.Button('reload',font=("Helvetica", fsize)),sg.Button('Exit',font=("Helvetica", fsize))]]
layout += [[sg.Text('視力量測紀錄表	共有'+str(record_len-1)+"筆資料",font=("Helvetica", fsize)),]]
```

```
layout += [[sg.Text(' ',font=("Helvetica", fsize))]+[sg.Text('姓名'+ ' '*12+'左眼視力'+ ' '*12+'右眼視力'+ ' '*12+'量測時間',font=("Helvetica", fsize))                ]]
#姓名解密機
#二為症列轉一為症列
list_data=double_to_single(list_of_lists)

layout += [[sg.Listbox(key="_list_",values=(list_data),font=("Helvetica", fsize), size=(300, 20))]]

window = sg.Window('Window Title', layout,size=(3072,1728))
#窗口標題和 layout 布局

while True:
        #定義窗口 while 循環里的事件和數值
        event,values = window.read() #讀出

        if event is None or event == 'Exit': #定義窗口關閉
                break
        elif event == 'reload':
                print('============================================================')
                print('reloading')
                worksheet=getdata()
                list_of_lists = worksheet.get_all_values()
                record_len=len(list_of_lists)-1
                #姓名解密機
                getname=values['name']
                #二為症列轉一為症列
                list_data=double_to_single(list_of_lists,getname)
                window['_list_'].update(list_data)#put list in 'name'



window.close()
print(event)
```

二、手機 App 程式碼


當 Button1 .被點選
執行 設 網路1 . 網址 為 "https://spreadsheets.google.com/feeds/list/1D9ip..."
呼叫 網路1 .執行GET請求
初始化全域變數 show_list 為 ""
初始化全域變數 name 為 建立空清單
初始化全域變數 list1 為 建立空清單
初始化全域變數 str1 為 建立空清單
初始化全域變數 str3 為 建立空清單
初始化全域變數 str2 為 建立空清單
初始化全域變數 name_unlock 為 建立空清單
定義程序 程序名2
回傳
執行方塊
設置 global show_list 為 ""
設置 global show_list 為 建立清單 合併文字 "姓名" "0" "左眼視力" "0" "右眼視力" "0" "日期時間"
對於任意 數字 範圍從 1
到 求清單長度 清單 取 global name
每次增加 1
執行 如果 文字輸入盒1 . 文字 = 選擇清單 取 global name_unlock 中索引值為 取 數字 的清單項
則 增加清單項目 清單 取 global show_list
item 合併文字 選擇清單 取 global name_unlock 中索引值為 取 數字 的清單項
"0"
選擇清單 取 global str1 中索引值為 取 數字 的清單項
"0"
選擇清單 取 global str2 中索引值為 取 數字 的清單項
"0"
選擇清單 取 global str3 中索引值為 取 數字 的清單項
回傳結果 從文字 取 global show_list
第 2
位置提取長度為 求長度 取 global show_list - 2
的字串


網路1 取得文字
URL網址 回應程式碼 回應類型 回應內容
執行
設 清單顯示器1 . 元素 為 建立空清單
設 清單顯示器1 . 元素字串 為 建立空清單
設置 global list1 為 建立空清單
設置 global name 為 建立空清單
設置 global name_unlock 為 建立空清單
設置 global show_list 為 建立空清單
設置 global str1 為 建立空清單
設置 global str2 為 建立空清單
設置 global str3 為 建立空清單
如果 取 回應程式碼 = 200
則 設置 global list1 為 在鍵值對 在鍵值對 呼叫 網路1 .解碼JSON文字
JSON文字 取 回應內容
中查找關鍵字 "feed"
, 如未找到則回傳 "not found"
中查找關鍵字 "entry"
, 如未找到則回傳 "not found"
對於任意 清單項目 清單 取 global list1
執行 增加清單項目 清單 取 global name
item 在鍵值對 在鍵值對 取 清單項目
中查找關鍵字 "gsx$name"
, 如未找到則回傳 "not found"
中查找關鍵字 "$t"
, 如未找到則回傳 "not found"
增加清單項目 清單 取 global str1
item 在鍵值對 在鍵值對 取 清單項目
中查找關鍵字 "gsx$va"
, 如未找到則回傳 "not found"
中查找關鍵字 "$t"
, 如未找到則回傳 "not found"
增加清單項目 清單 取 global str2
item 在鍵值對 在鍵值對 取 清單項目
中查找關鍵字 "gsx$rva"
, 如未找到則回傳 "not found"
中查找關鍵字 "$t"
, 如未找到則回傳 "not found"
增加清單項目 清單 取 global str3
item 在鍵值對 在鍵值對 取 清單項目
中查找關鍵字 "gsx$datetime"
, 如未找到則回傳 "not found"
中查找關鍵字 "$t"
, 如未找到則回傳 "not found"
設置 global name_unlock 為 建立空清單
對於任意 數字 範圍從 1
到 求清單長度 清單 取 global name
每次增加 1
執行 增加清單項目 清單 取 global name_unlock
item 呼叫 lockname1 byteTostr
str1 選擇清單 取 global name
中索引值為 取 數字
的清單項
設 清單顯示器1 . 元素字串 為 建立空清單
設 清單顯示器1 . 元素字串 為 呼叫 程序名2

【評語】032807

1. 本作品應用 AI 影像辨識系統作為視力檢查手勢判斷為其研究創意，其整合一完整視力評估系統，從影像顯示解析到手勢辨識，以實作數據校正作品，可以應用於實際場合。

2. 對於裝置進行視力檢查正確性的成效，與傳統方法的對照中，建議統計每個樣本以兩種方法測試後的差異程度與數量，而且最好以相同的目視裝置作為控制變因，進行比較，此外應注意相關公式數值計算的有效位數。

3. 由於此裝置成本較傳統裝置高，建議可做成本效益分析。

4. 總體而言，此為一件完成度很高的科展作品，值得嘉獎。

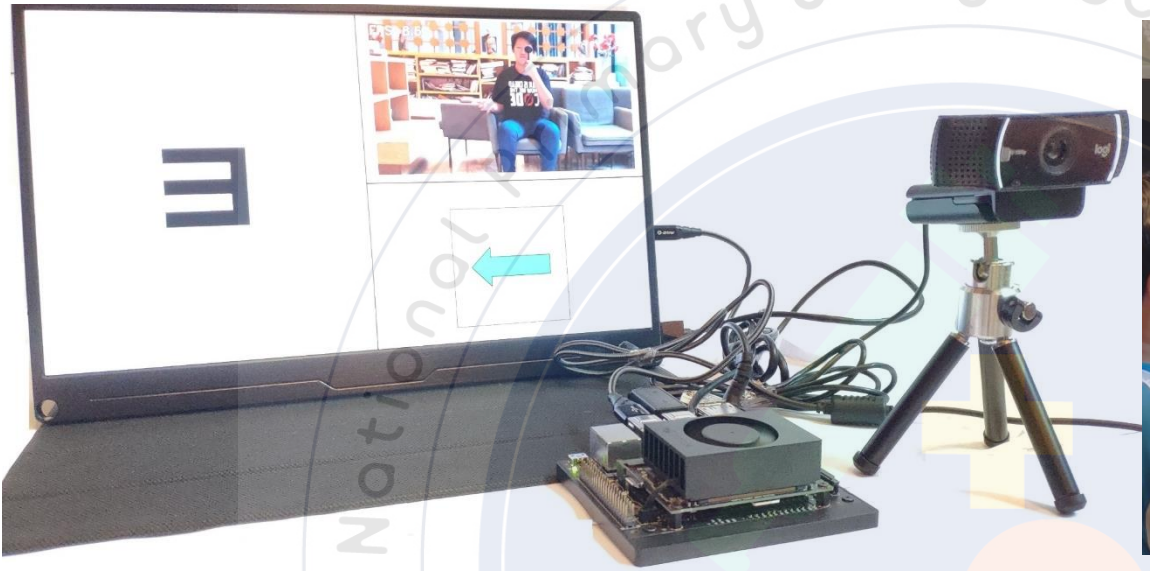
作品簡報

AI影像辨識
輔助視力量測系統

國中組

生活與應用科學科(一)

研究摘要

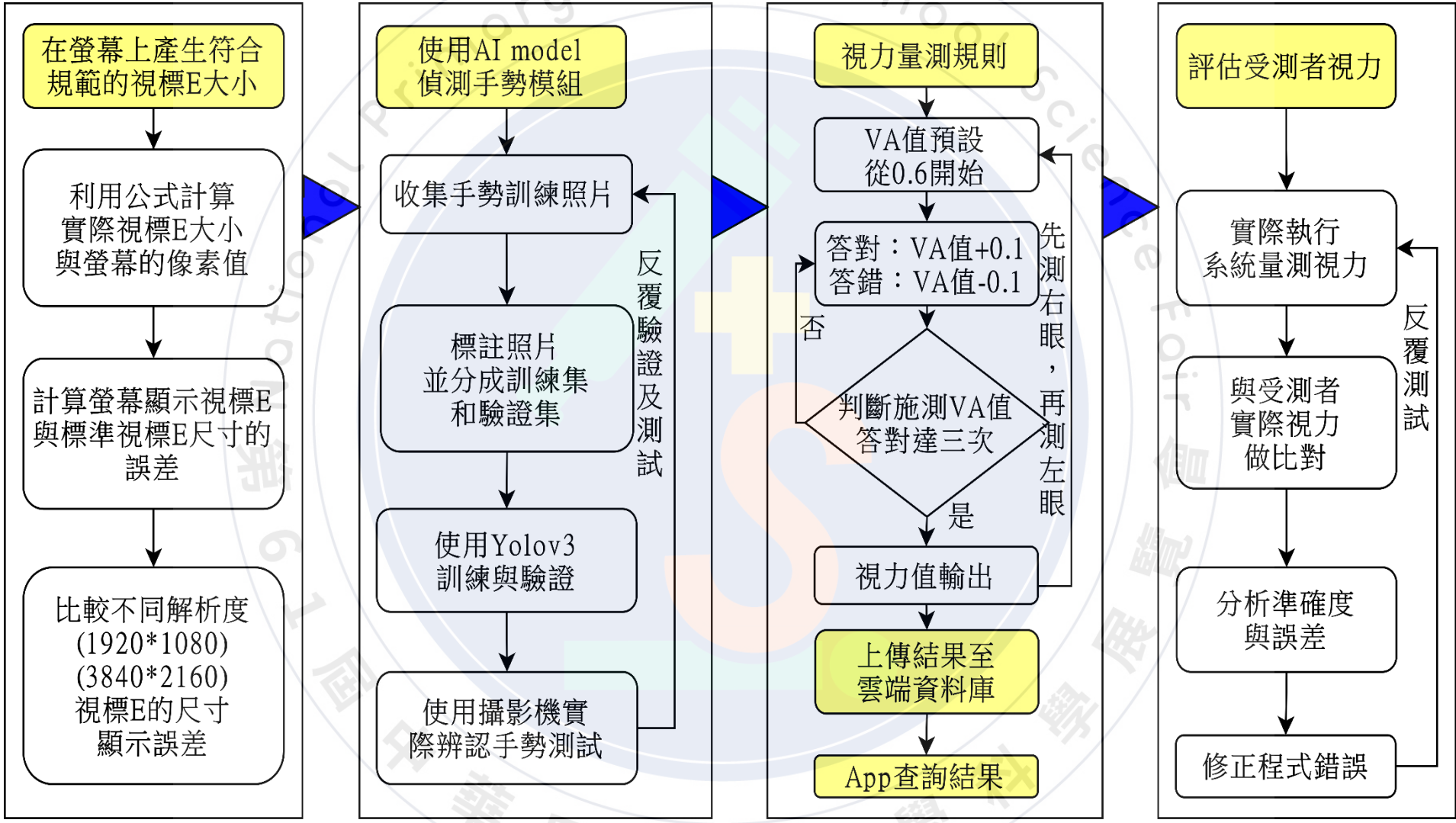


圖一、AI視力量測系統硬體配置

圖二、視力量測系統實測圖

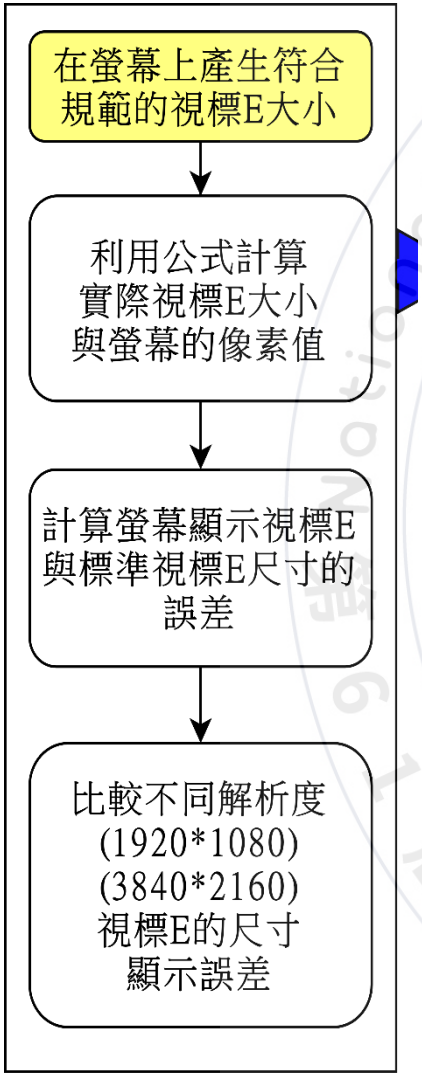
- 疫情期間線上課程、居家辦公用眼量大增
- 為減少醫護人員工作量、無人協助即可自行在家視力量測
- 根據環境空間彈性選擇待測距離，低成本、低功耗、體積小
- 視力量測總平均誤差值0.097、準確率可達96.645%以上
- 量測資料連結雲端系統，並結合App進行視力追蹤

研究目的及流程

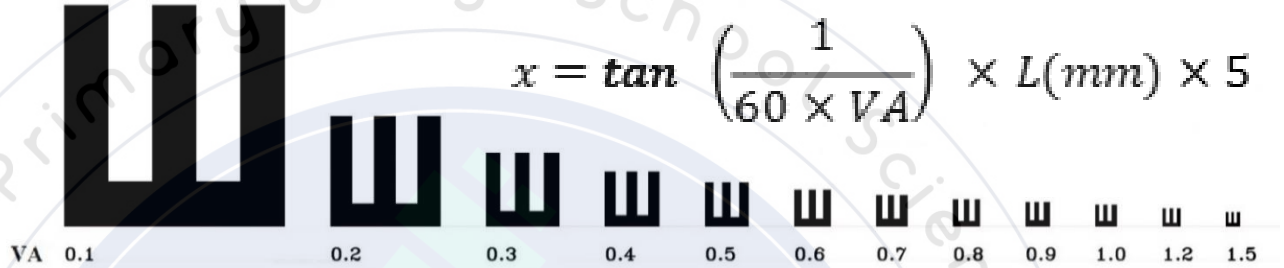


圖三、研究過程及方法流程圖

一、編寫出自動產生視標E的程式碼

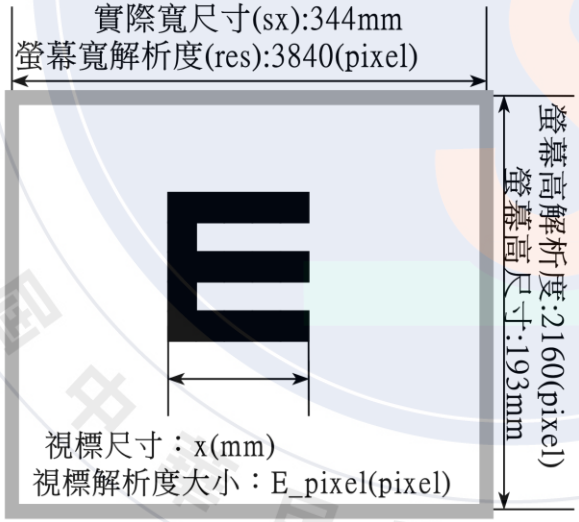


$$x = \tan\left(\frac{1}{60 \times VA}\right) \times L(mm) \times 5$$

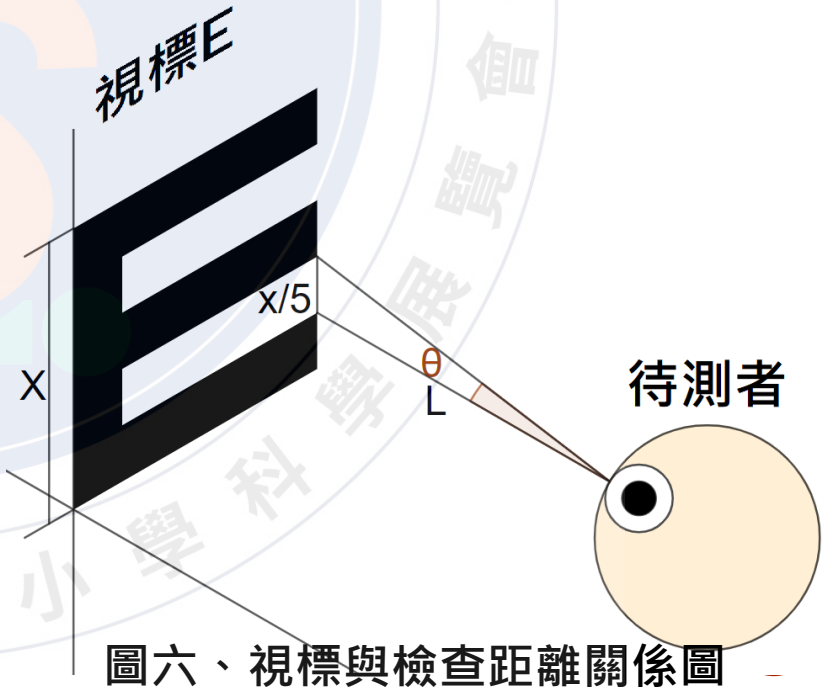


圖四、視標E樣本圖片

- 當視力VA=1.0，代表受測者可『解析』視網膜上1分角大小的成像，指可在待測距離6公尺看清楚一弧分距離大小。



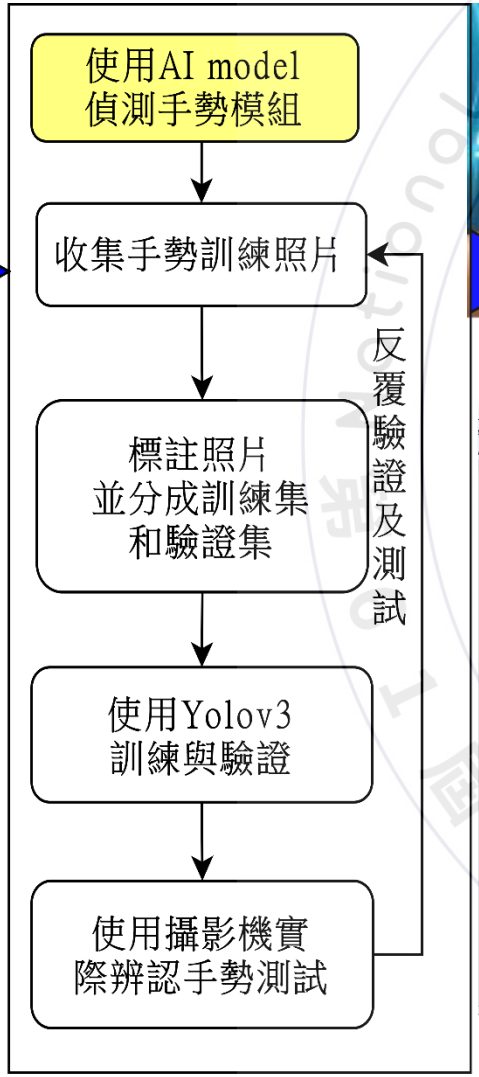
圖五、螢幕實際尺寸與解析度

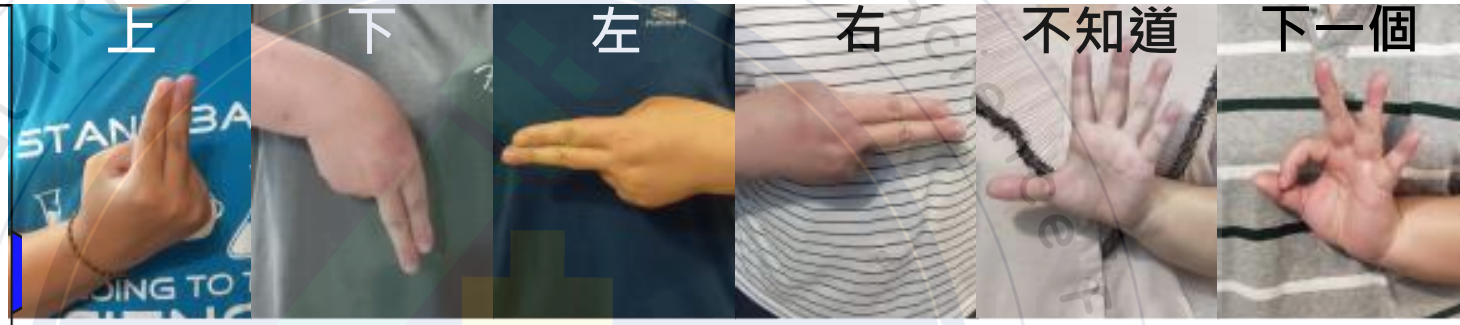


圖六、視標與檢查距離關係圖

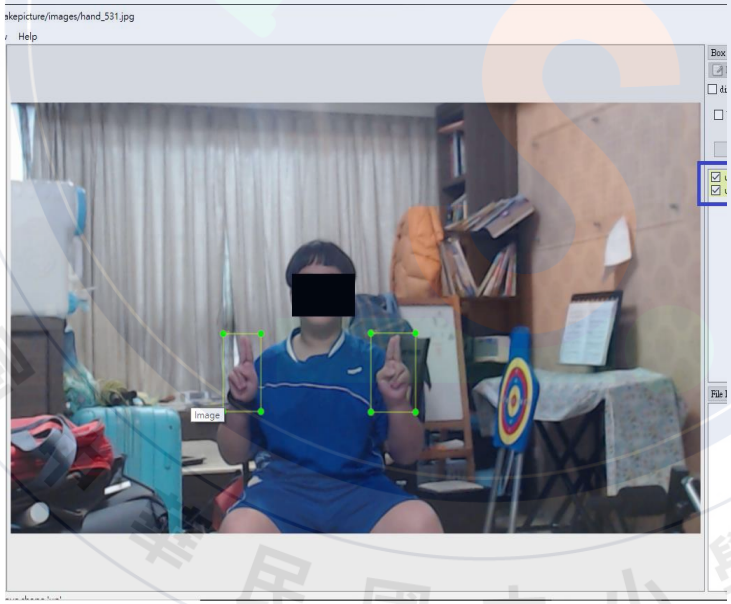
二、訓練可判讀待測者手勢方向的AI模型

為訓練AI model，本研究收集六種手勢照片共631張，進行10000個AI訓練週期。訓練完畢後測試資料準確度可達99%（如圖九紅線部分）。

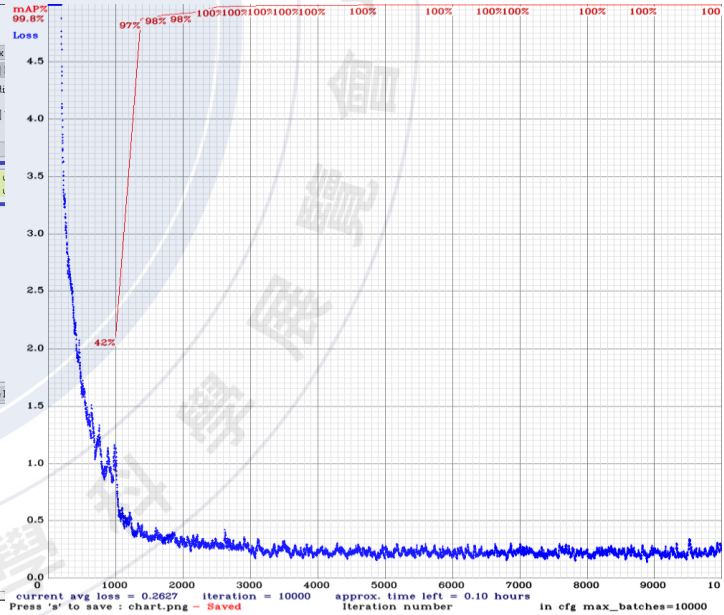




圖七、標註訓練照片（共六種手勢）



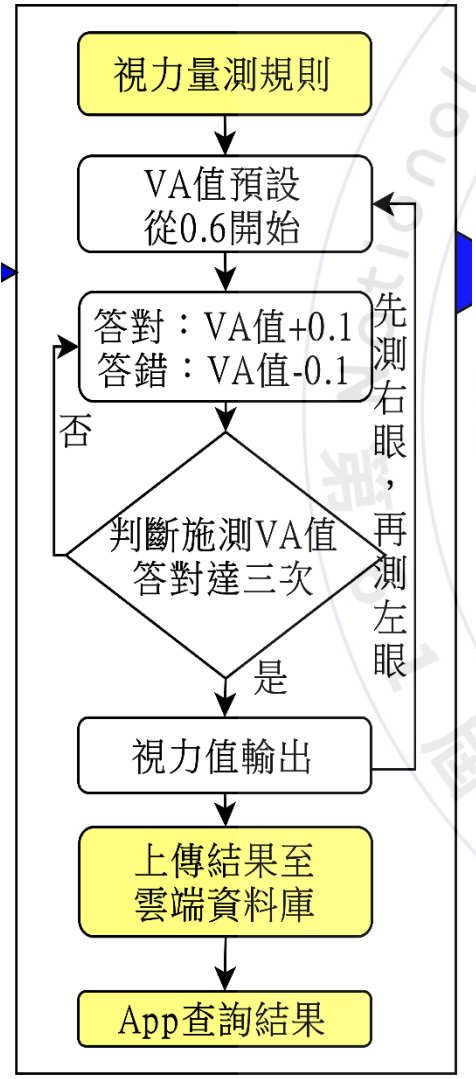
圖八、labelimg進行標註圖片



圖九、AI訓練過程

三、將視力量測規則程式化

本研究增設定點距離輸入，使用者可因應不同的施測環境空間，自行設定施測距離（三公尺、兩公尺），方便居家彈性空間調整使用。



視力量測規則如下：

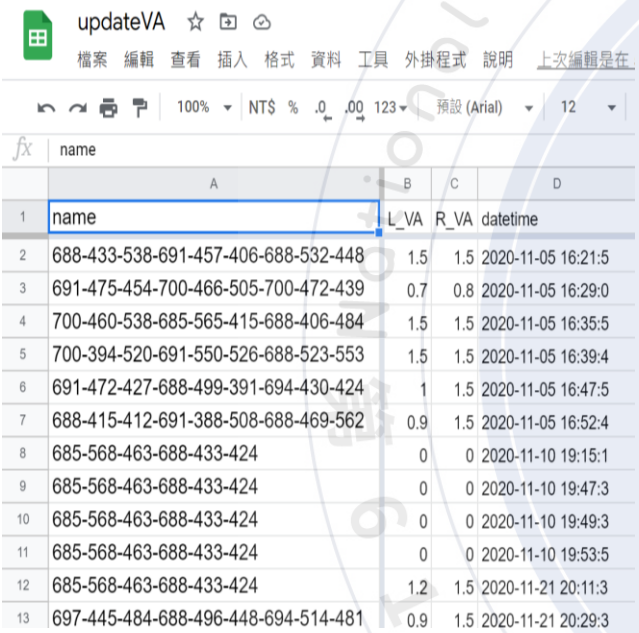
- 受測者答對：跳下一排視標E檢測區(VA+0.1)
- 受測者答錯：跳上一排視標E檢測區(VA-0.1)
- 答對數過半的最大VA值即為量測結果

四、製作雲端資料庫與視力查詢APP

- 當系統實際量測時會自動將視力量測值上傳至google sheets 雲端資料庫。
- 配合本研究所製作的App可隨時隨地透過網路查詢視力量測結果。

本研究使用google sheets製作雲端資料庫，並使用Python3和App Inventor2製作視力查詢App，考量個資問題，在施測時，會附加製作姓名加密系統。

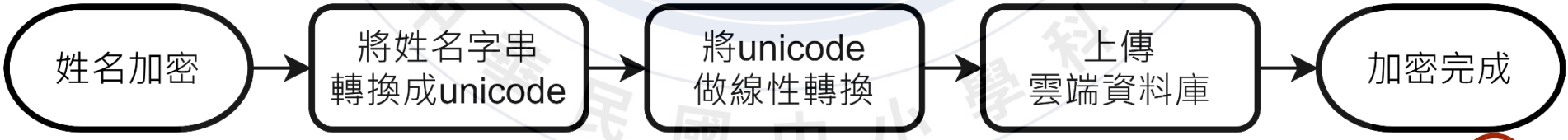
- **電腦版App**：可讓醫護人員瀏覽所有受測者的視力結果，並進行資料統計及分析。
- **手機版App**：可供受測者方便、迅速、及時的查詢視力量測結果。



圖十、雲端資料庫

圖十一、電腦版App（醫生用）

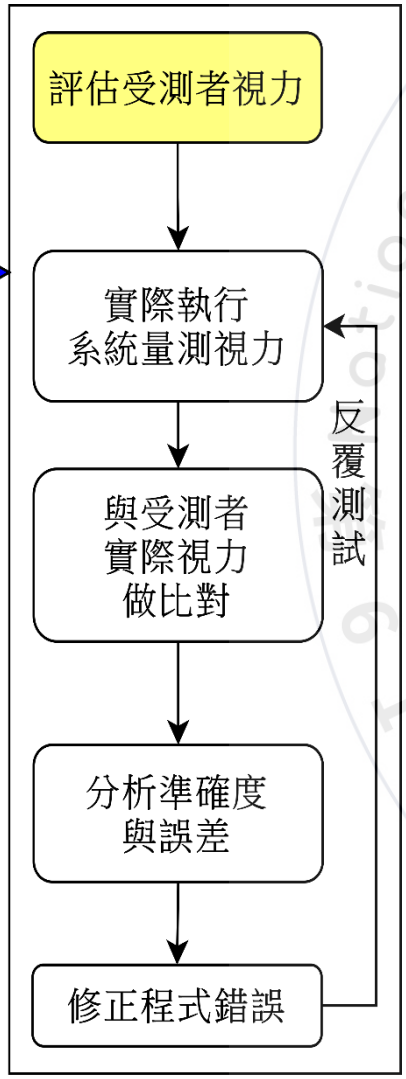
圖十二、手機板App（受測者用）



圖十三、加密流程圖

五、將系統搭載在JETSON NX 並進行整合

系統硬體搭載完畢後如圖十四所示，而螢幕顯示軟體系統執行畫面。



圖十四、系統整合硬體圖片

本研究選擇Jetson Xavier NX做為系統整合開發平台，效能可達8~10FPS，每秒AI model可偵測受測者8~10個手勢方向，實測上已可順暢的進行測試。

六、將系統視力量測值與傳統量測做比較

本研究之施測年齡層分兩個層次：學齡階段（~17歲）及成年階段（18~歲）。根據測量結果，無論是在待測距離三公尺或兩公尺準確率皆能達水準，因此可使用本系統彈性調整待測距離來因應場地空間不同的問題。

表一、系統量測與傳統測量之平均誤差表

年齡層	人次	平均誤差(VA)			總誤差(VA)	準確率(%)
		左眼	右眼	平均		
測試距離三公尺						
學齡階段	24	0.092	0.113	0.102	0.097	96.645
成年階段	14	0.114	0.064	0.089		
測試距離兩公尺						
學齡階段	5	0.150	0.100	0.125	0.084	98.014
成年階段	6	0.080	0.020	0.050		

六、將系統視力量測值與傳統量測做比較

本研究準確度可達96.645%，量測所需時間約5分鐘，且不需排隊等待量測，受測者可自行操作，更可因應不同環境空間選擇不同的測量距離，相較傳統量測彈性應變更佳。

表二、傳統醫院量測和本系統量測的差異比較表

	傳統手比量測	本研究系統
量測時間	5 分鐘	5 分鐘
量測準確度	接近100%	96.645%
人力耗費	每個人次皆需1人力指導	1醫護人員可同時啟動 多台機器同時量測
設備成本	約3500元（視力燈箱）	22000元（本系統）
方便性	須有人指導協助	可在家自行操作量測
量測距離	固定三公尺或六公尺	可彈性選擇 兩公尺或三公尺 （居家操作方便）

一、系統硬體部分

本研究將系統實作在Jetson NX，**達到低成本，低功耗，體積小**等優點，且效能達8~10FPS，符合研究需求。

二、系統軟體部分

本研究寫出可產生標準視標E的程式，並訓練可辨識手勢的AI model接著程式化視力量測規則。實際量測**準確率達96%，且可彈性調整量測距離**。

三、雲端資料庫及App應用程式的部分

本研究使用google sheets製作雲端資料庫，並製作**視力查詢App**，考量個資問題，進一步製作加密系統**加密姓名**。

四、未來展望

本研究未來希望可將系統移植到效能更好、價格更便宜的硬體平台；或者為這個系統擴充更多功能，例如：**色盲色弱檢測**。

結論

- **高準確率之視力輔助量測系統**
- **可望減輕醫護人員的工作負擔**
- **預先檢測、追蹤及記錄視力變化**
- **及早發現、及早治療的預防功能**

- 許明木、莊素貞、鄭靜瑩、陳賢堂、王俊諺、吳承臻、林則豪、許淑貞、連政炘、葉志偉（2017）。低視力學（2版）。臺北：五南出版社。
- 衛生福利部國民健康署（2016）。兒童視力篩檢及矯治指引。取自 https://www.hpa.gov.tw/Pages/ashx/File.ashx?FilePath=~/File/Attach/1077/File_8248.pdf
- 黃麗恩（2009）。老年性白內障之視覺模擬測試平台建構。取自臺灣博碩士論文知識加值系統。
- Alexey(2020)。Darknet。取自https://github.com/AlexeyAB/darknet
- Yqlbu(2020)。TRT-yolov3。取自https://github.com/yqlbu/TRT-Yolov3
- Nvidia(2019)。NVIDIA宣布推出Jetson Xavier NX全球尺寸最小的邊緣AI超級電腦。取自https://blogs.nvidia.com.tw/2019/11/06/nvidia-introducing-jetson-xavier-nx/

參考資料