中華民國第 61 屆中小學科學展覽會

作品說明書

高級中等學校組　電腦與資訊學科

第一名

052505

彩色二維條碼手持產品開發之探討

學校名稱：國立嘉義高級中學

作者：	指導老師：
高二 李艾登	黃冠夫

關鍵詞：影像辨識、彩色二維條碼、機器學習

得獎感言

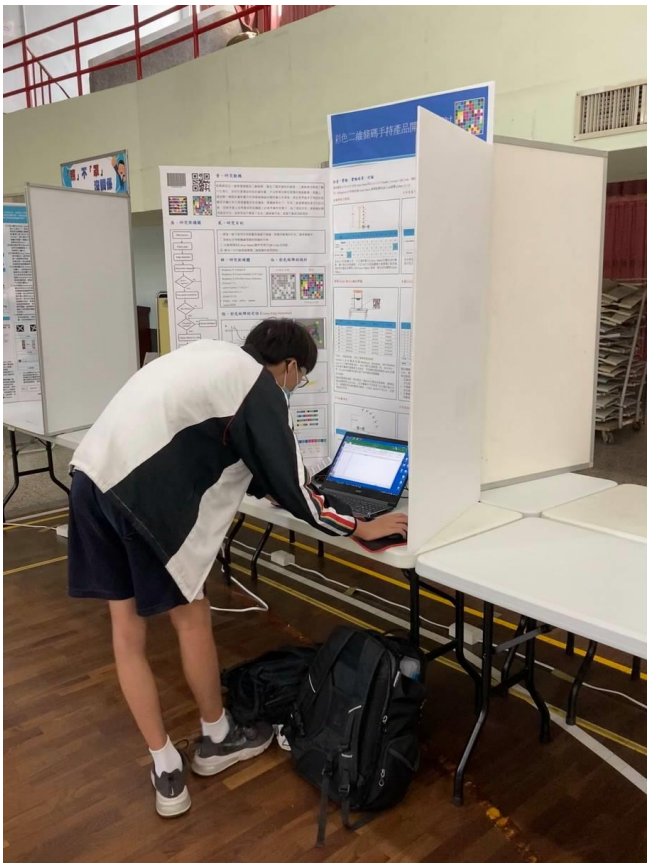
我住在嘉義，離嘉義市幾十公里外的嘉義，所能得到的資源並不多，大部分知識都是透過老師傳授，台灣並不缺教育資源，但是珍貴的資源都集中在離我百公里外的大城市裡，能突破困境獲取得資源的唯一途徑就是通過網路，所幸的是我有英文優勢。當國中畢業時參加英文檢定，多益 945 分，雅思 6.5 分，我把我的英文能力當作一個工具，用來拓寬我的視野。

在國中會考結束後當晚，我開始在 Coursera 平台上的密西根大學，密集的修了三門 Python 基礎課程，偶然之下發現了哈佛大學的知名電腦課程 CS50，CS50 課程很有趣，內容也不難，問題在課程結束的時候有一個期末專題程式作業 ，我沒有足夠的信心可以完成，直到上完全部課程，我還是遲遲不敢註冊。事後我一直在四處尋找有潛力的題目，最後在英國的泰特現代美術館，我找到了。

上了嘉義高中不出幾天我就找到了指導老師，設備組長兼地科老師，雖然老師並不會寫程式，但是老師願意每個禮拜五晚上撥空讓我留校與他討論，並且在遇到問題時提供解決問題之建議與設備，長達兩年的時間，老師常常為了我一個人多待在學校好幾個小時，非常感謝老師。

在做這個科展題目之前，我程式經驗只限於前面 Python 課程的作業和考試，沒有做過一個完整 project，剛開始，常常因為一個簡單的錯誤就可以讓我卡關好久，在每周五與老師有約的壓力下，我只能轉向 Google 大神尋求幫助，因此我在網路學會了很多東西，像是不同的顏色表示方式，電腦的影像辨識，Linux 作業系統的操作、如何進行 debug，並且還學了很多程式知識，像是 openCV, image recognition, machine learning 與人工智慧等等，在實驗後期進入精進實驗過程中，我也學會了搜尋與閱讀期刊。

持續大量閱讀期刊後，蜂擁而至的知識與未知讓我興奮不已，我想我會繼續在學習電腦程式這條路上持續走下去。



摘要

數位二維條碼的使用日益普及，然而使用手機掃瞄時，常受限於掃瞄距離及拍攝角度，使用上不夠便利。在不增加拍攝限制且增加二維條碼儲存容量的前提下，本研究嘗試設計一款 8 色 10×10 模組的彩色二維碼，命名為 Colour Matrix。其四個頂角的模組具特定顏色，用來作為定位的特徵，其他模組用於儲存資料。此研究也成功在樹莓派平台上開發專屬的顏色辨識及編碼與解碼應用程式。相較於常用的二維條碼 QR Code，Colour Matrix 的拍攝傾角限制較小但拍攝距離較短。升級拍攝鏡頭後的實驗則發現鏡頭解像力明顯影響 Colour Matrix 辨識率，以及研究中所使用的邊緣偵測方法具有極限，有待進一步深入探討。

壹、 研究動機

QR Code (Quick Response Code)是一種非常方便可以在小空間內儲存資料的方式，被廣泛應用於各行各業。然而在一般手機使用上時有其缺點，相信多數人都有以下經驗:在一段距離之外要掃描一個 QR Code 時，雖然已經盡量把 QR Code 放到手機的掃瞄範圍之內，但是仍然無法辨識出來。原因是一般掃描 QR Code 的 APP 有掃描距離內方格數量與圖形大小的限制。此外 QR Code 最初的設計所能儲存的資料量較少，因為使用越來越廣，近期的研究為了將資料處存量提升，因此設計出了許多種不同種類，更複雜的 QR Code 或 Matrix。

迄今已經有許多種設計被提出以解決上述問題，可區分為以下列數種。第一種是試圖增進硬體在 QR Code 的辨識準確度，例如 2020 台北國際自動化工業展示中，利用精密的儀器在受到控制的環境下，可以精準且快速地從視野中找出數個 QR Code，缺點就是昂貴，而且辨識環境有許多的限制，得在特殊規格與設計的打燈環境中，才能進行掃描。第二種是擴大模組，第一代的 QR Code 版本 1 為 21×21 模組（模組為 QR 碼中的最小單元），每增加一個版本，長寬各增加 4 個模組，目前最大的版本 40 為 177×177 模組。第三種是試圖在有限的空間內增加更多的資料，比方說 HCC2D(High Capacity Colored 2-Dimensional Code)，裡用顏色來增加資料密度，擺脫了原本的二進位制，來到 4、8 甚至 16 進位制。如此一來資料量相較同樣格數的 QR Code 資料量比大了數倍。

但用顏色來增加資料密度的設計有一個致命的缺點，就是，到目前為止，就算是 Microsoft 等大公司，仍然無法讓 HCC2D 在室外光線下進行掃描，因為光線無法被控制造成色彩失真，所以顏色辨識在實際應用上有其難度，研發工作相繼在 2016 年截止。

在工資較低廉的國家，並不是所有的商家，小型工廠，網拍業者都願意花錢購買整套的二維條碼系統，較符合需要是一個簡單手持裝置甚至是手機，就能增進品管和降低出貨錯誤造成的退貨支出，若能設計出一款 Matrix 可以利用顏色來存資料，並且可以用手持裝置來掃描的二維條碼系統，相信一定能打開另一片尚未被開發的藍海市場。

貳、 研究目的

開發一套可使用手持裝置快速進行掃描、辨識及解碼的彩色二維條碼(Colour Matrix)套件。具體步驟如下:

一、設計一款新型式 Colour Matrix 及其相應編碼，能儲存更多資料。

二、發展在手持裝置處理顏色辨識的方法。

三、比較開發的 Colour Matrix 套件和現行 QR Code 的條件限制。

參、 研究設備及器材

一、 研究硬體

Raspberry Pi 4 Model B	Raspberry Pi Camera Module v2
Micro SD card(128GB)	Logitech C922 Pro Stream WebCam
3A type-c 充電插頭	Micro HDMI to HDMI 轉接線
筆記型電腦(乙太網路)	USB 鍵盤/滑鼠
Double A 影印紙 80 磅	乙太網路線
HP LaserJet Pro CP1025 彩色印表機列印解析度 600 x 600 dpi	PASCO 無線光感應器 
電腦螢幕	

二、 軟體與工具

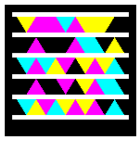
1. Raspberry Pi OS GNU/Linux 10 (buster)

2. Python 3.7.3
3. opencv-python 3.4.6.2
4. scikit-learn 0.23.1 (SK learn)
5. Pyzbar 0.1.8
6. Webcamoid
7. 相關套件： numpy、scipy、tqdm、pillow 、pandas、matplotlib。

肆、 文獻探討

一、 數位行動條碼發展史

表格 1..數位條碼種類

一維條碼 (Code 39)	二維條碼 (QR Code)	彩色二維條碼 (Microsoft)
		

最早開始發展的行動條碼是一維條碼，由平行的黑白線條組成，掃描條碼後可對應至特定的文字、符號或數字。由於具保密性，讀取的錯誤率低，又可快速處理資料，因此被廣泛應用。一維條碼可存放的資料量約 20 characters(字元)，資料量不足是其設計上的缺點。較新式的二維條碼在設計上增加了垂直方向這個維度的空間，可存放的資料量較一維條碼多，且處理速度更快。

最常見的二維條碼是 1994 年由日本 Denso-Wave 公司發明的 QR (quick response) code，基本結構是一個由模組(module)構成的正方形，黑色模組代表 1，白色模組代表 0。模組解碼後可以對應到網址、手機號碼、簡訊、字串等。第 1 代的 QR Code 設計為 21×21 模組，每增加一個版本，長寬各增加 4 個模組，目前發展至第 40 代 QR Code，為 177×177 模組。資料量方面，第 1 代 QR Code 可存放 35 個字元，第 40 代的 QR Code 版本可存放高達 2953 個字元的 8 位元組訊息或 7089 字元的數字(Mishra & Mathuria, 2017)。

為了增加辨識度，QR Code 在左上、右上及左下角各設計了一個定位標記(圖 1)，另外還有數個較小的校正標記。為了降低錯誤率，QR Code 中並不是所有的模組都被

用來儲存資料，而是保留部分模組用來容錯，因此當部分模組毀損時，QR Code 仍可以被正確解碼。

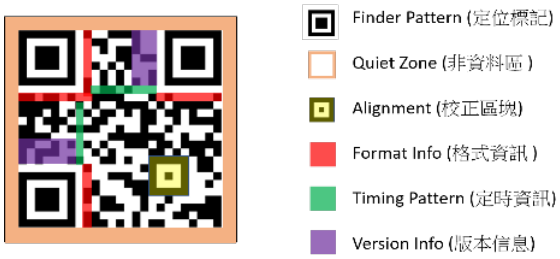


圖 1.QR Code 結構圖

為了在二維條碼有限的區域內儲存更多的資訊，最簡單的做法是增加模組的數量，但這會提高掃描及辨識的困難度。另一個方法是提高單一模組的資料量，例如使用顏色模組來取代黑白模組。使用顏色模組的彩色二維碼雖可增加資料量，但在實務上解碼彩色二維碼會比黑白二維碼困難，程序也較複雜。首先，擷取彩色影像過程中，物體實際顏色會受到環境光源的色溫及照度影響，產生顏色失真。此外，相鄰模組之間的顏色可能會互相干擾，產生混色及誤判(Yang, Xu, Deng, Loy, & Lau, 2018)。

彩色二維碼在文獻中的資料相當少，設計時模組可以選用三角形或正方格(John & Raahemifar, 2015)。目前網路上可以查詢到的彩色二維碼產品只有微軟在 2007-2013 年開發，利用 CMYK4 色的小三角形製作的 5 x 10 模組的高容量彩色二維碼，商品名為 Microsoft Tag("High Capacity Color Barcodes (HCCB)," 2007)，但目前相關資訊在微軟網站上已停止更新。

二、 電腦彩色視覺

電腦的世界是由 0 和 1 組成，所以它需要將”看” 到的影像先轉成 0 和 1 的數位訊號才能讀懂。電腦視覺的產生需要利用鏡頭感光元件來接收光線及感應顏色，利用影像擷取卡來將類比視訊轉為數位訊號，再將訊號輸出至電腦主機裡進行分析處理(陳志強, 2018)。在電腦眼中，色彩是一組 RGB 數值，R 指紅色(Red)、G 指綠色(Green)、B 是藍色(Blue)。RGB 模式中，所有顏色都是由紅、綠、藍按照不同的比例疊加而成。RGB 的 “值”指的是顏色的強度，從 0 到 255 共分 256 級。當 RGB 值為 0 或 255 時，此時色彩的飽和度(彩度)位於極端，這種情況共有 8 個(圖 2)。理論上，這 8 個極端色彩應該是比較容易被電腦分辨出的顏色，因此被選用來製作本實驗的 Colour Matrix。

圖 2. 8 個具極端彩度的顏色及其相對的 RGB 數值。

三、 彩色辨識

在不同光源及亮度下，人的眼睛及大腦能藉由生活經驗及試誤學習，判斷正確的顏色。但是，電腦沒有這個能力。電腦感知的顏色是利用接收到的 0 和 1 數位訊號算出來的，而不是”看”出來的，當計算的結果不準確，電腦就會看到錯誤的顏色。因此，會發生在清晨拍的彩色數位照片偏紅而在黃昏拍的照片偏黃的情形。由於目前鏡頭及相機主要被設計來拍攝人物及風景，賣點是高畫質影像及賞心悅目的色彩，顏色是否失真反而是較不被重視的問題(Akkaynak et al., 2014) ，因此，解決顏色辨識將是開發彩色二維碼時的一項挑戰。

開發彩色二維碼的過程需要經過編碼、彩色影像輸出、影像擷取、彩色濾光片 RGB 分色、分色資料提取、還原原始彩色影像，再經過解碼對應出資訊內容。為了使顏色提取正確，可以使用影像後處理技術，例如 luminance enhancement、thresholding binarization、salt and pepper refinement algorithm 等(Bhardwaj, Kumar, Verma, Jindal, & Bhondekar, 2016)。為了克服 RGB 值會因環境照度與照相條件而異的問題，辨識彩色二維碼顏色則可以使用機器學習的分類演算法，如隨機森林演算法(random forest)、支援向量機演算法(support vector machine)、差商演算法(quadratic discriminant analysis)等(Yang et al., 2016)，讓電腦預測及校正所讀取的顏色資訊。

四、 影像處理-影像切割(Image segmentation)

在一張圖片中搜尋目標影像時，可以針對目標影像的特徵來進行適當的切割，以簡化待處理的資訊，加速後續圖像的分析。常見的影像切割方法為:

(一) 閾值分割法:

閾值分割法是一種利用設定一個或多個灰度閾值，將整張影像依設定的灰階閾值劃分為不同區域的分割技術(Image Thresholding)。在一張 8 bits(28 =256)灰階圖中，最暗(灰階值=0)到最亮(灰階值=255)共有 256 種亮度變化。圖 3 為二值化的灰度閾值分割法，當設定灰階值的閾值=100 時，可將圖片分為灰階值<100 及≥100 兩種區塊。

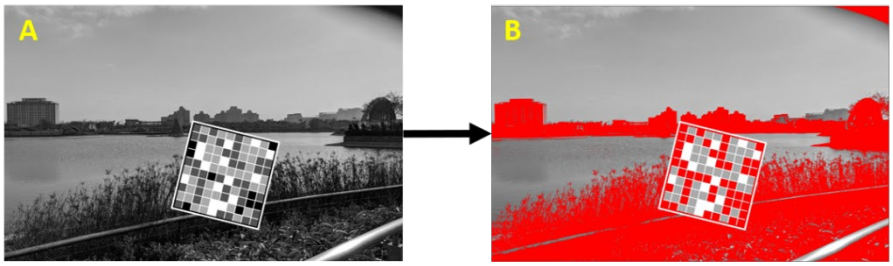


圖 3.(A)原始影像 (B)閾值分割法(灰階值<100 低於 100 的區域以紅色標示)。

(二) 邊緣檢測(Edge detection):

圖像中具有不同灰度的兩個區域在交界處會存在明顯的灰階值差異，因此，可以利用這種灰階值的梯度(gradient)落差作為區塊邊緣的特徵。Canny edge detection 是一種常見的邊緣檢測演算法，原理是先利用高斯濾波器來濾出邊緣的梯度值，計算可用來區分相鄰兩區塊邊緣的梯度極大值及極小值。當灰度梯度較極大值高時，認定為區塊邊緣(圖 4A 藍線 a 段)；灰度梯度介於極大值及極小值之間但是和灰度梯度大於極大值像素相連的(圖 4A 藍線 b 段)，認定為區塊邊緣；灰度梯度介於極大值及極小值之間但是未和灰度梯度大於極大值像素相連的(圖 4A 綠線)，則非區塊邊緣；灰度梯度較極小值低的，亦為非區塊邊緣(圖 4A 洋紅線)。圖 4C 利用邊緣檢測方法來分類圖片中的物體，Colour Matrix 的邊緣可效偵測出來。

圖 4. Canny edge detection (A)邊緣連線原理 (B)原始影像 (C)邊緣檢測法。

(三) 其他的影像切割技術

包括區域分裂合併及區域生長法(Region growth)等。

五、 影像處理-透視轉換(Perspective Transformation)

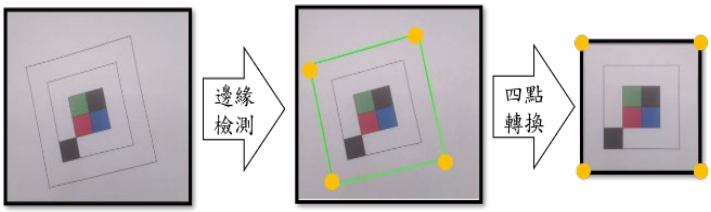


圖 5.(A)拍攝圖影 (B))邊緣檢測 (C)四點轉換。

3D 的實物轉換成 2D 的平面影像的過程中，距離遠近的資訊會因壓縮而消失，物體形狀也會扭曲而形變。這很像繪畫構圖時，為了在 2D 圖片裡呈現物體的遠近及形狀，反而必須將現實中原本互相平行的線條設計成會匯集於遠方的一個點(消逝點)，利用幾何學的概念在平面圖像裡呈現出立體空間中物體位置及形狀。當掃描 Colour Matrix 後，圖片中線與線的交叉點座標及線條的尺寸會失真。因此，當在現實世界中掃描 Colour Matrix 後，圖片必須轉換回正面鳥瞰圖，才能對應到模組的座標，進行模組顏色解碼的動作。此實驗使用四點轉換(4-point perspective transform)來進行轉換。

六、 機器學習

機器學習是一種利用人工智慧技術，讓電腦經由處理及學習大量資料，訓練分辨資料特徵及產生解決問題的能力。所以當新的數據出現時，便能根據過往訓練的經驗(模型)進行預測。進行機器學習時，會預先保留一部分資料留作測試集，用來評估演算法的表現。其他資料則用來用於訓練演算法的模型，稱為訓練集。訓練集與試集比例可以選擇 1:2 或 1:4 等。模型的預測能力是評估模型好壞的依據。最常見的指標是準確率(Accuracy)，準確率越高的模型其預測能力通常也越好。由於個人經驗有限，進行顏色辨識時，先考慮較常見的機器學習分類演算法，包括決策樹、k－近鄰演算法(k-nearest neighbor, KNN) 和多層感知器(multilayer perceptron, MLP)。

決策樹是一種利用像樹一樣具有分支及葉片的結構圖來做決策的分類方法(圖 6A)，樹的每個分支節點代表判斷條件，分類結果則連結到下一層的決策節點或是分類終點(葉片) (Quinlan, 1996)。

KNN 是一種簡單的演算法。當處理的是分類問題時，分類結果將由 K 個鄰近樣本以多數決方法決定；當處理的是樣本是數值時，預測值則由鄰近 K 個樣本平均值來決定(Altman, 1992)。以圖 6B 為例，當要判斷一個未知物體的顏色(X)是屬於青、洋紅或紅色時，可以先將訓練用的青、洋紅及紅色數值繪於座標系統中，再比對未知顏色的座標位置和哪幾個顏色的座標點最接近。比對後發現它和青色的特徵分布距離最短，便預測未知顏色為青色。進行 KNN 演算法時，可以選擇的參數除了 K 值，還可以對 K 個鄰近樣本的貢獻進行加權，賦予較近距離的樣本較大的距離加權。這種改良版的 KNN 稱為加權 KNN。

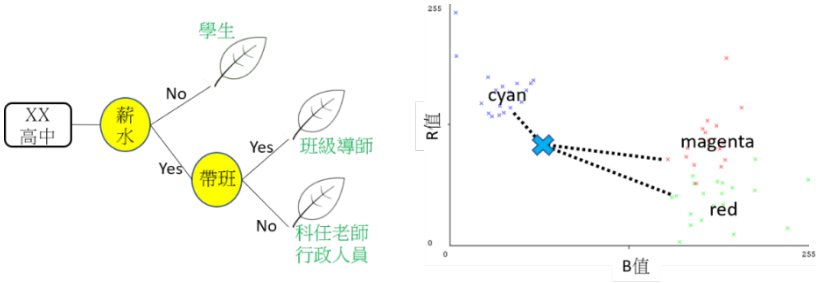


圖 6. A 決策樹　　B . k－近鄰演算法(k=1)

MLP 是一種類神經網路，包含一個輸入層、數個隱藏層及一個輸出層(圖 7)。文獻中這種方法曾被成功用來判斷汽車的顏色(Zhang et al., 2018) 。MLP 的設計是模仿人類神經系統的傳遞，網路中的節點扮演著神經元的功能(圖 7 紅色圓圈)，節點和上層及下層間有對應的連結。節點在處理輸入及輸出訊號的過程中，會學習到與上下層之間的連結強度，並修正每條迴路之權重值，進行最佳化推演。

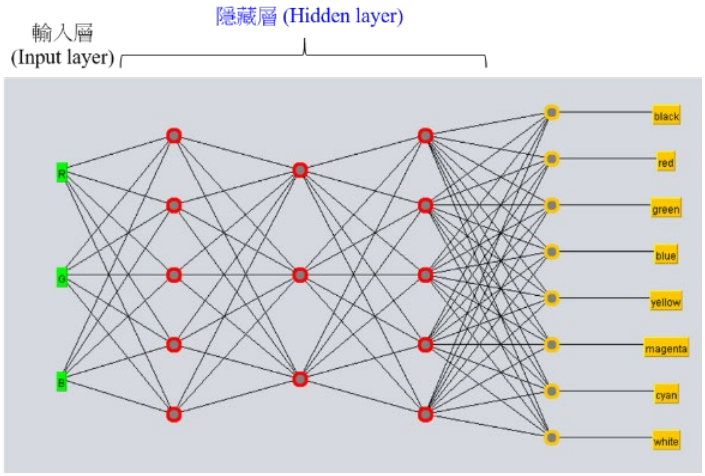


圖 7.多層感知器

伍、 研究過程及方法

進行手持彩色二維碼的研究過程中，我參考了網路及參考文獻自己設計了一款彩色二維碼，稱之為Colour Matrix。Colour Matrix 編碼與解碼的方法參考 QR code 的設計。影像處理部分是使用網路上搜尋到的開源軟體來偵測及擷取條碼。顏色辨識是自己收集各種環境條件下的顏色資料，並使用機器學習的方法所開發出來。圖 8 為此實驗的研究架構圖。

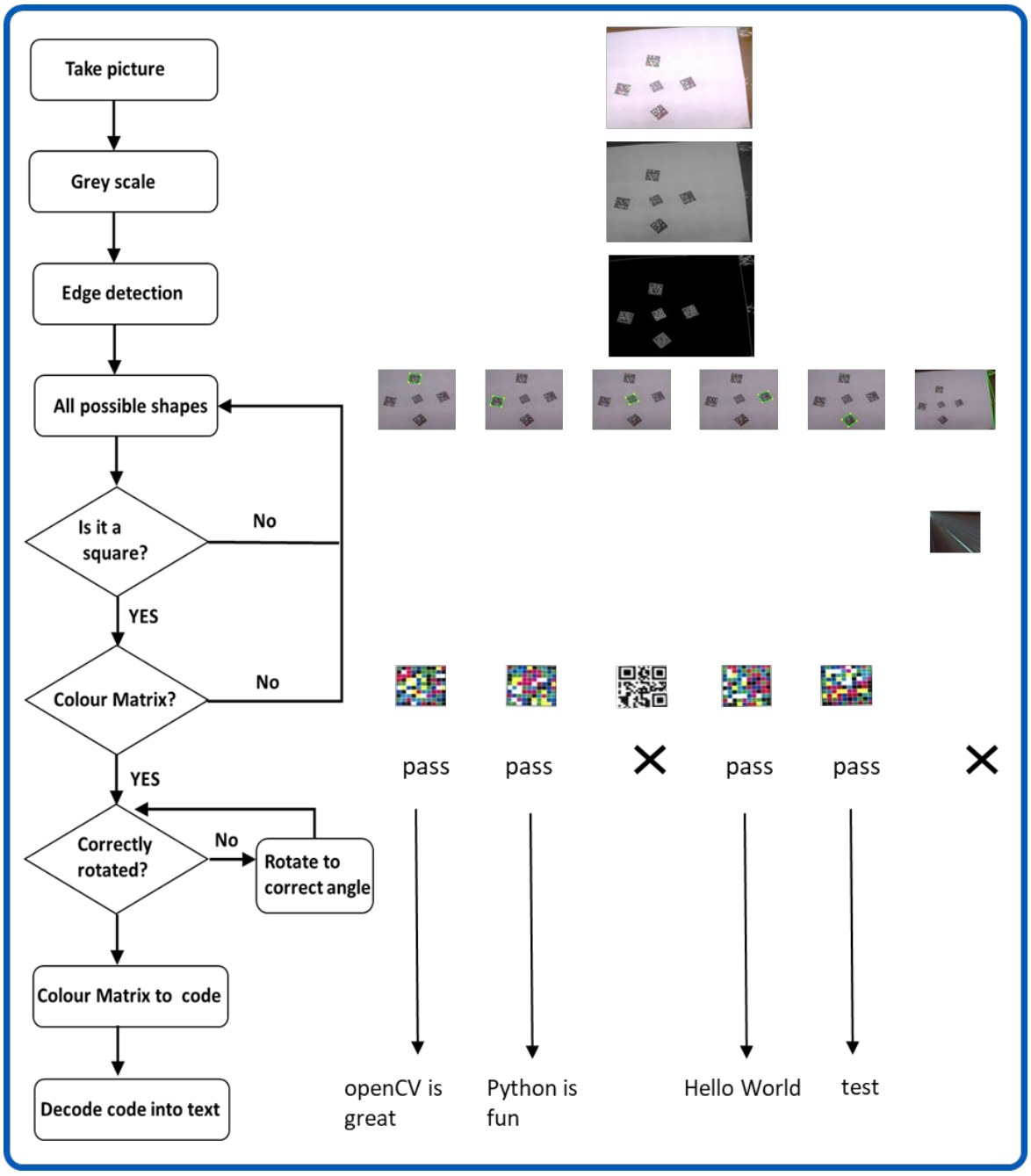


圖 8.研究架構圖

一、 軟硬體選擇

本實驗目的是要製作手持彩色二維碼平台，需使用支援影像輸入及後處理的硬體。評估的硬體包括筆電、Arduino 及 Raspberry Pi(簡稱 RPi)。由於經費有限，加上目標是要能做為手持裝置，因此開發板式如 Arduino 及 RPi 是較適合的平台。實驗最後選用 RPi 作為開發的平台，因為 RPi 有專屬的相機 Raspberry Pi Camera(Pi Cam)，鏡頭為 Sony IMX219 8-megapixel sensor，支援 800 萬像素拍照，畫質和市售低階手機差不多，加上 RPi 也搭配 8Pi Cam 的軟體，使用方便。RPi 另一個優勢是支援 Linux 作業系統及 python environment。由於 python 提供許多影像處理及機器學習套件，加上網路上有非常多 python 學習管道，可望降低開發門檻。

表格 2.硬體規格比較表

硬體	優點	缺點
筆記型電腦	64 位元微處理器 記憶體 4GB 起 作業系統(+) 內建顯示晶片	體積大，不適合手持 價格高，15000 起台幣
Arduino Uno	體積小 價格低，762 台幣	8 位元微處理器 記憶體 0.002MB 未內建顯示晶片，無法處理影像 作業系統(-)
Raspberry Pi 4	體積小 64 位元微處理器 記憶體 4GB 作業系統(+) Linux 內建顯示晶片	價格中等，2400 台幣

二、 彩色矩陣(Colour Matrix)的設計

實驗目標是要設計出一個比 QR Code 更實用的彩色二維碼，必須具備類似 QR Code 的功能，可以利用程式將它從一張照片中自動定位、擷取及判讀。也必須像 QR Code 一樣，可以從多個方向和角度掃描，希望可以增加掃描的距離。此外通常 QR Code 掃描器一次只能掃描一個，我還希望設計的彩色二維碼可以同時一次掃描多個，提高應用範圍。

我將自己設計的彩色二維碼命名為"Colour Matrix"，選用的為 10×10 的顏色模組(如圖 9 左)，顏色共 8 個，均為具高度飽和彩度的色彩。為了增加辨識率，Colour Matrix 的

四個頂角設計為定位格，功能類似 QR Code 中的定位標記(如圖 9 中)。Colour Matrix 定位格左上、右上、左下、右下的顏色分別是紅、黑、綠、藍。當符合這樣的規則時，程式將自動判定掃描到的圖像為 Colour Matrix。四個定位格的顏色同時也可以用來作為 Colour Matrix 轉正時的方向校正基準，協助程式呈現 Colour Matrix 鳥瞰圖。當 Colour Matrix 正確轉正後，讀取模組的順序由左而右、由上而下，如圖 9 右。

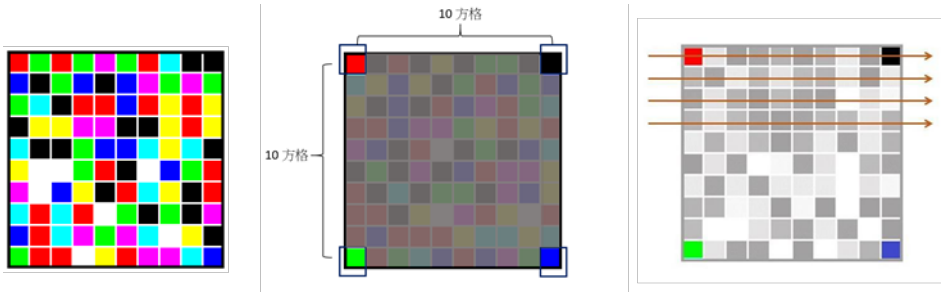
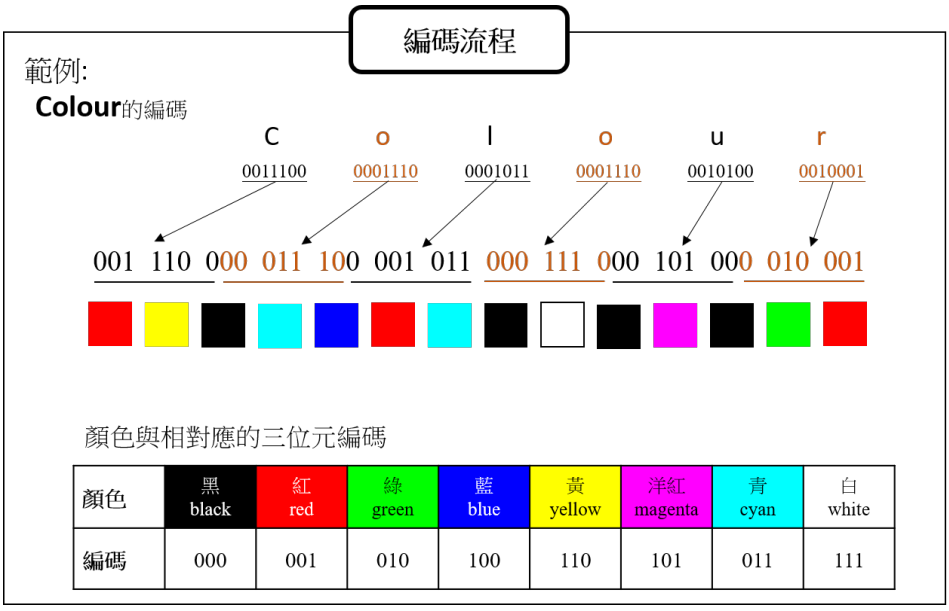


圖 9.(左) Colour Matrix 的範例。(中) 10×10 模組 Colour Matrix 規格。四個頂角模組為定位格，灰色半透明區為資料區模組。(右)Colour Matrix 的解碼順序。

三、 編碼設計

實驗的 Colour Matrix 選用 10×10 的模組，模組變化選用的是 8 個具高度飽和彩度的顏色，這 8 個顏色的特徵是在 RGB 數值上具極端值。顏色所對應的代碼及編碼原則如圖 10。由於設計的 Colour Matrix 有 8 個顏色，使用不用到三個模組就可以完整的儲存一個子元。相較之下，QR Code 只有黑與白色，所以須要 8 個模組才能儲存一個字元。



顏色	黑 black	紅 red	綠 green	藍 blue	黃 yellow	洋紅 magenta	青 cyan	白 white
編碼	000	001	010	100	110	101	011	111

圖 10.編碼流程圖

QR Code 的編碼方式為 8 位元編碼，有數字，英數，位元組，日語漢字模式及其他模式五種。在位元組模式中，編碼可以是 ISO-8859-1 字元或是 UTF-8 的編碼。ISO 8859-1 是國際標準化組織內 ISO/IEC 8859 的第一個 8 位字符集，以 ASCII 為基礎，可用來表示使用附加符號的拉丁字母，而 ASCII 同樣也是 UTF-8 的子集。因此我的構想是使用類似的編碼方式，資料讀出來後可以使用一串類似 ASCII to Binary Code 的編碼來解讀 Colour Matrix。

URL 中常見的字元由英文字母大小寫、數字 0-9、特殊符號加上終止碼共 86 個組成，至少需 7 位元編碼(即 2 的 7 次方=128)，才能完整的涵蓋，所以本實驗 Colour Matrix 的字元編碼採 7 位元制(表 3)。每個 10×10 模組的 Colour Matrix 在扣除四個定位點後會剩下 96 個模組，每個模組能儲存 8 個顏色(2 的 3 次方=8)，即 3 位元，因此共可儲存 288 個位元。字串最後須加上 END 字元(編碼為 1010110)，END 後不足 288 位元的部分使用亂數。Colour Matrix 的範例如圖 11。

表格 3.字元編碼本

字元	代碼	編碼	字元	代碼	編碼	字元	代碼	編碼	字元	代碼	編碼	字元	代碼	編碼
a	0	0000000	u	20	0010100	O	40	0101000	8	60	0111100	-	80	1010001
b	1	0000001	v	21	0010101	P	41	0101001	9	61	0111101	_	81	1010010
c	2	0000010	w	22	0010110	Q	42	0101010	!	62	0111110	.	82	1010011
d	3	0000011	x	23	0010111	R	43	0101011	*	63	1000001	~	83	1010100
e	4	0000100	y	24	0011000	S	44	0101100	"	64	1000000	SPACE	84	1010101
f	5	0000101	z	25	0011001	T	45	0101101	(	65	1000010	END	85	1010110
g	6	0000110	A	26	0011010	U	46	0101110	)	66	1000011			
h	7	0000111	B	27	0011011	V	47	0101111	;	67	1000100			
i	8	0001000	C	28	0011100	W	48	0110000	:	68	1000101			
j	9	0001001	D	29	0011101	X	49	0110001	@	69	1000110			
k	10	0001010	E	30	0011110	Y	50	0110010	&	70	1000111			
l	11	0001011	F	31	0011111	Z	51	0110011	=	71	1001000			
m	12	0001100	G	32	0100000	0	52	0110100	+	72	1001001			
n	13	0001101	H	33	0100001	1	53	0110101	$	73	1001010			
o	14	0001110	I	34	0100010	2	54	0110110	/	74	1001011			
p	15	0001111	J	35	0100011	3	55	0110111	?	75	1001100			
q	16	0010000	K	36	0100100	4	56	0111000	%	76	10011.01			
r	17	0010001	L	37	0100101	5	57	0111001	#	77	1001110			
s	18	0010010	M	38	0100110	6	58	0111010	[	78	1001111			
t	19	0010011	N	39	0100111	7	59	0111011	]	79	1010000			

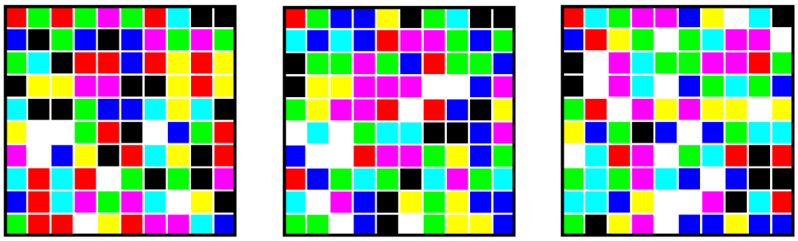
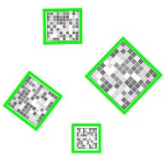
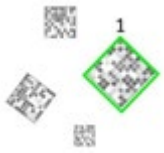
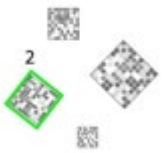
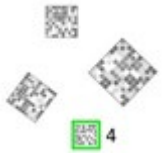


圖 11.10x10 模組 Colour Matrix 範例

四、 Colour Matrix 定位與確認

在圖片中要找出 Colour Matrix 的位置時，可以利用 Colour Matrix 的特徵。由於 Colour Matrix 是正方形，形變後通常以四邊形來呈現，於是可以使用 Canny edge detection 來辨識圖片中物件的邊緣線條，接著篩選出這些線條中可以組成一個四邊形的集合(表格 4)。如果拍攝角度在一定的範圍內，四邊的大小相差太多的也可以去除。四邊形的集合再進一步確認是否符合 Colour Matrix 的設計，如果是，就利用旋轉功能將 Colour Matrix 轉為鳥瞰圖進行後續流程。當圖片中找到的四邊形超過一個時，程式會依據四邊形面積大小排序，依此順序進入迴圈，進行後續處理。

表格 4.多張 Colour Matrix 時的辨識順序

原始影像	多張 Colour Matrix 辨識順序 (找到四邊形→依面積大小排序→進行後續處理)				
					

程式偵測到一個四邊形後，會先確認四邊形是否符合 Colour Matrix 的定義並進行解碼(圖 12)。首先，先進行 4 個頂角定位點的確認，步驟是先將四個定位點應該出現的地方 crop 下來，由機器學習進行顏色辨識流程，確認四個定位點的顏色順序是否跟 Colour Matrix 一樣，如果顏色順序對但方向不符合，就進行旋轉，直到旋轉至正確的角度為止，這樣就得到一個被確認過且轉正的 Colour Matrix。

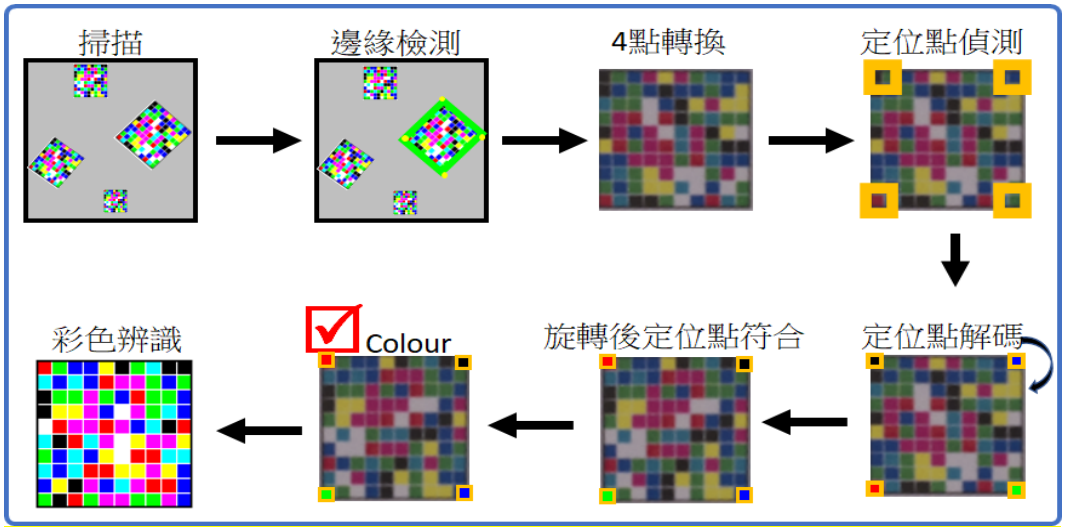


圖 12.Colour Matrix 的定位及辨識

五、 Colour Matrix 解碼流程

Colour Matrix 為 8 個顏色組成的 10×10 模組，包含著 4 個定位點跟 96 每個顏色模組，每個顏色模組都可以定應到一個三位元的代碼，利用對照表可以解出 Colour Matrix 的資訊，解碼範例如圖 13。

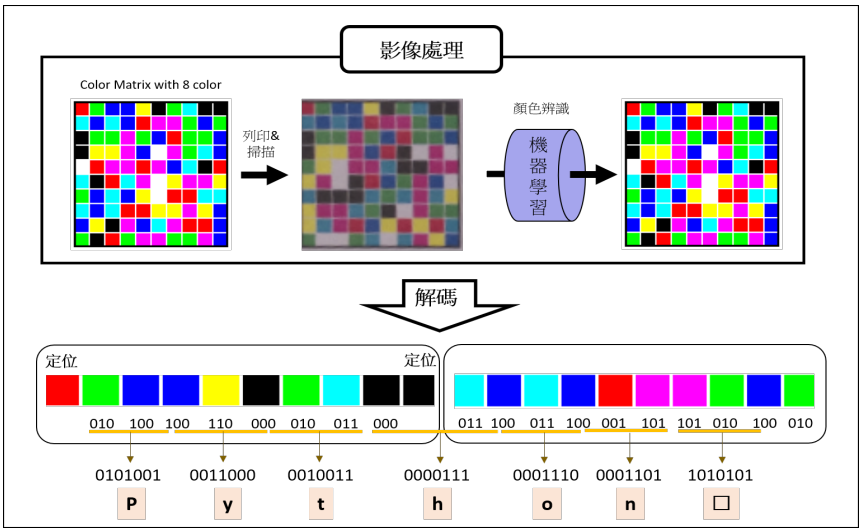


圖 13.Colour Matrix 解碼的流程圖

六、 實驗一、 建立顏色辨識的模型

電腦判斷顏色時是由一組 RGB 值決定，但實務上鏡頭在擷取顏色過程中會因環境光源及照度的影響而失真。本實驗採用機器學習的分類法來訓練及建立顏色辨識模型。訓練集的資料在盡可能多樣性的環境下進行，以提升資料的複雜性。這樣雖會降低訓練模型的正確性，但可提高預測的正確性。

[實驗步驟]:

1. 製作邊長 2 公分x2 公分，具 8 種顏色，10x10 模組的 Colour Matrix。
2. 在多種不同環境條件下拍攝 1 組影像用於建模(如圖 14) 。使用的
3. LED 燈具色溫 4000K，環境亮度為 100-700 Lux。
4. 使用 OpenCV 辨識物件位置並輸出 RGB 值。
5. 進行監督式學習，手動標示每個 RGB 值所對應的正確顏色。
6. 使用 SKlearn 建立顏色辨識的演算法，包括決策樹、KNN 及 MLP。
7. 重複上述流程，將被辨識錯誤的模組資料正確標示後，加入訓練
8. 集，直到能得到一個有效辨識顏色的訓練集。
9. 額外拍攝 2 張 Colour Matrix，作為測試集資料。

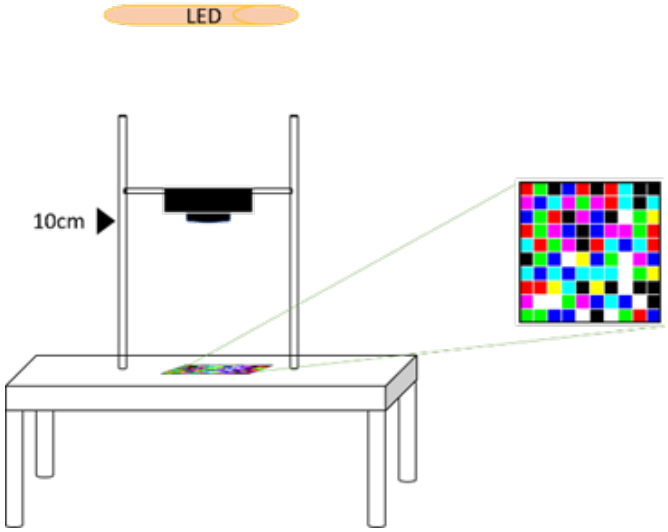


圖 14.實驗架設圖

七、 實驗二、 Colour Matrix 和 QR Code 的效能比較

為了測試所開發的 Colour Matrix 套件的效能，接下來以 model 2 version 1 QR Code 作為對照組，比較 Colour Matrix 及 QR Code 在拍攝距離、拍攝傾角及不同亮度下的掃

描限制。實驗中的 Colour Matrix 及 QR Code 邊長同為 2 公分x2 公分，儲存內容相同 - Python is fun 或 I Like Python。為讀取 QR Code，於樹莓派上安裝 QR Code 軟體 Pyzbar 0.1.8。pyzbar 是一個常用的模組，可同時解碼多組 QR Code 並提供 QR Code 邊緣的位置。

(一) 實驗二之一、掃描距離

[實驗步驟]:

1. 以 Pi Cam 拍攝桌面上 Colour Matrix。
2. Pi Cam 高度依序調整為距離桌面 5、7.5、10、12.5、15、17.5、20、22.5、25、27.5、30、32.5、35 公分，每個條件拍攝 3 次。
3. 進行影像解碼流程(如圖 13)。
4. 以 QR Code 替換 Colour Matrix 並按重複以上步驟，拍攝後以 Pyzbar 進行解碼。

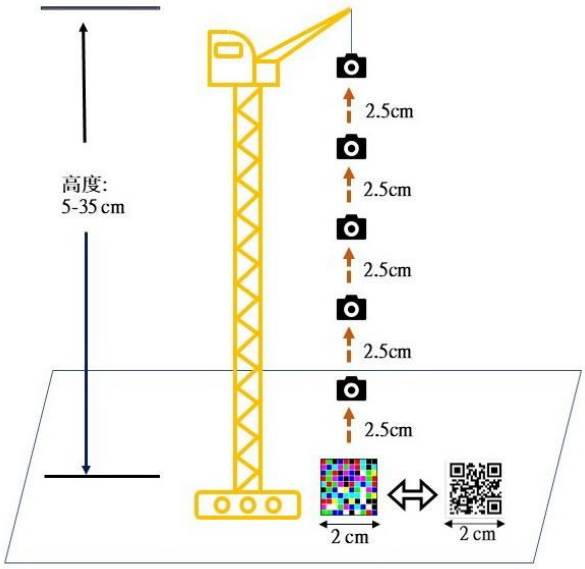


圖 15.改變 Pi Cam 拍攝距離

(二) 實驗二之二、桌面傾角

[實驗步驟]:

1. 拍攝桌面上 Colour Matrix，Pi Cam 固定在 Colour Matrix 正上方 10cm。
2. 依順時鐘方向傾斜桌面，使拍攝傾角分別為 0°、15°、30°、45°、60°、75°，每個條件拍攝 3 次。

3. 進行影像解碼流程(如圖 13)。
4. 以 QR Code 替換 Colour Matrix 並按重複以上步驟，拍攝後以 Pyzbar 進行解碼。

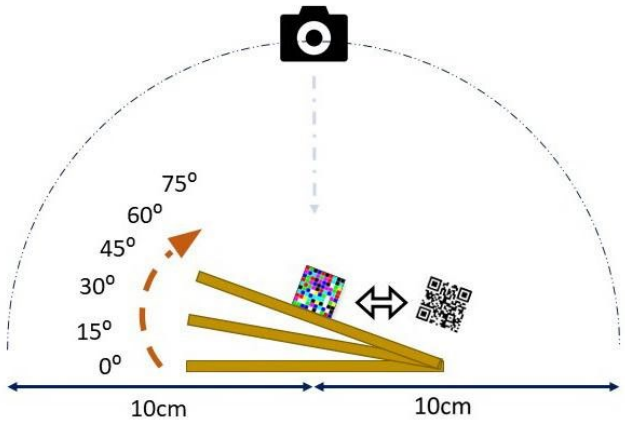


圖 16.不同傾角桌面的 Color Matrix 、QR Code 的解碼成功率:

(三) 實驗二之三、環境照度

[實驗步驟]:

1. 拍攝桌面上 Colour Matrix，Pi Cam 固定在 Colour Matrix 正上方 10cm。
2. 使用色溫 4000K 燈源，照度分別為 100、200、300、400、500、600 及 700 Lux，每個條件拍攝 3 次。
3. 進行影像解碼流程(如圖 13)。
4. 以 QR Code 替換 Colour Matrix 並按重複以上步驟，拍攝後以 Pyzbar 進行解碼。

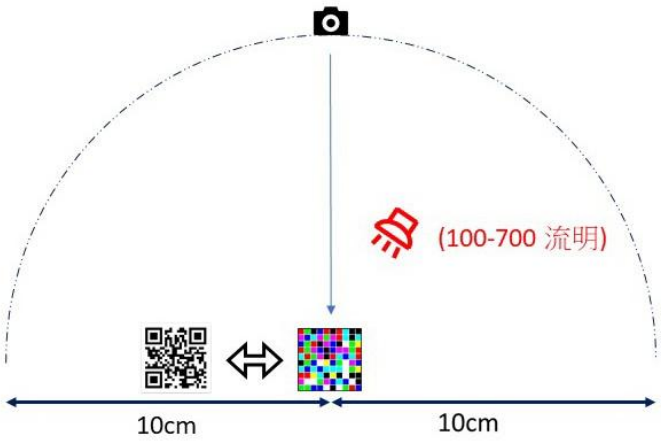


圖 17.不同亮度下辨識能力

八、 實驗三、 拍攝多張 Colour Matrix 辨識

市面上所提供的 QR code 掃描及解碼器，都只有支援單張 QR code 的，如能同時掃描及解碼多張條碼，將有機會提高條碼的應用範圍。下列實驗目的是為了測試所開發的 Colour Matrix 套件在進行多張 Colour Matrix 辨識的效能。

[實驗步驟]:

1. 列印 Colour Matrix 於 A4 紙張，每列 10 個，共 7 列。
2. 拍攝桌面上 Colour Matrix，Pi Cam 固定在 Colour Matrix 正上方 20cm 處，進行 10 次拍攝。
3. 進行影像解碼流程。
4. 分別輸出邊框辨識及解碼結果。

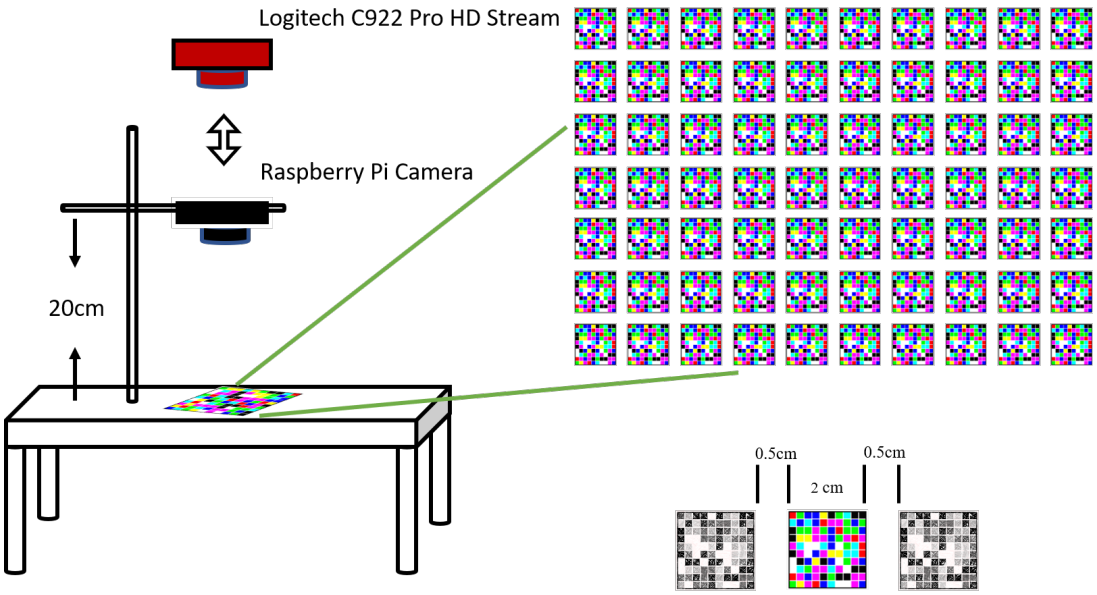


圖 18.拍攝多張 Colour Matrix 辨識

九、 實驗四 、鏡頭測試

初期研究會採用 Pi Cam 鏡頭進行測試，理由是因為樹莓派內建 Pi Cam 驅動程式，可隨插即用，操作方便加上操作的程式也相對簡單易懂。但 Pi Cam 是一個相對規格陽春的鏡頭，雖然配備有很好的 Sony CCD，但是因採用塑膠鏡頭規格，拍攝品質不及玻璃鏡頭，照片相對上景深較淺又不清晰。Pi Cam 另一項缺點是只有手動對焦功能，拍攝過程較為耗時。

進行了前面數個實驗後，由實驗結果推斷如果能改良"相機"這個因素，或許有機會可以提升辨識率，因此決定要找一個新的鏡頭進行實驗。在所有台灣可取得且支援樹莓派的 webcam 中("RPi USB Webcams," 2020)，Logitech C922 Pro Stream WebCam 解析度最高，有玻璃鏡頭，可提供 Full HD 1080P 畫質，而且能自動對焦。但是 Logitech C922 不支援 Pi Cam 的拍照軟體，只能用 OpenCV 來進行拍照。但是最後發現 OpenCV 無法進行自動對焦，因此最後是使用軟體 Webcamoid 來進行手動對焦拍照。

[實驗步驟]:

1. 同實驗三。

陸、 研究結果

一、 實驗一、建立顏色辨識的模型

(一) 建立訓練集

在多樣化拍攝條件下，選取資料來建立訓練集，共 713 筆資料，分布如圖 19。

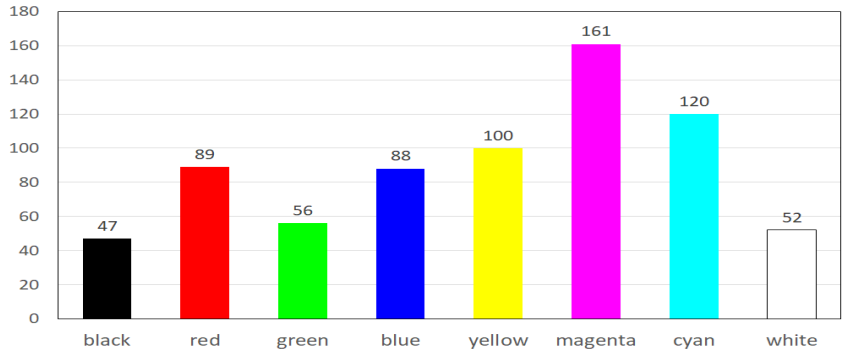


圖 19.各顏色的訓練資料數量

(二) 評估機器學習演算法的效能

使用機器學習法進行顏色辨識時，KNN(k=3)及 MLP(hidden layer =1) 正確率均高達 99.0%，J48 分類的正確較差一些，為 98.0%。由於 KNN 較 MLP 簡單而效能好，而訓練的模型正確率相同，所以 KNN 被選用來進行接下來的顏色辨識實驗。此部分的顏色辨識機器學習先在筆電上進行測試，接下來實驗完全在手持裝置 RPi 上執行，事後將所

有實驗數據存成 csv 檔，輸入 RPi 的 SK learn，用來訓練後續實驗顏色辨識的機器學習分類法，並將 KNN 之最佳模型以 pickle library 儲存。

(三) 建立高品質訓練集

表格 5 為不同環境照度下 8 個模組顏色的 RGB 數值。可發現相同顏色在不同照度下具有明顯差異的RGB數值，這可能導致顏色預測失準。例如紅色理論上R值為255，但實際測得的 R 值介於 132 和 161 之間；而紅色的 G 值和 B 值原本應該皆為 0，但在 200Lux 下，竟然 G 值高達 48 而 B 值高達 62。

此外，同樣照度下各顏色的 RGB 值範圍有時也不符預期。例如當照度為 500 Lux 時，紅色為(132,32,45)，其中 R 值為 132，明顯高於 G 值和 B 值，這樣的情況符合我的期待。但此條件下，綠色為(81,109,54)，其中 G 值只有 109，與 R 值 81 非常接近，和理論上的綠色 RGB 值(0,255,0)有所出入。藍色的 RGB 理論值為(0,0,255)，實際測量為(38,39,92)，其中最高的為 B 值只有 92，比預期低。此外，黑色在 200 Lux 下會近似深灰色，而白色則接近極淺的灰色。由此結果可知高亮度的的環境會造成畫面偏紅，對綠色的辨識不利，且普遍藍色 RGB 的數值均偏低。因此，增加訓練集資料多樣化對進行正確預測有其必要性。

表格 5.不同照度下 RGB 值。括弧內的三個值為 RGB 值之平均

照度 / 顏色	200 Lux	300 Lux	400 Lux	500 Lux
紅	(161,48,62)	(136,39,53)	(135,37,51)	(132,32,45)
綠	(109,137,72)	(93,120,61)	(91,116,62)	(81,109,54)
藍	(58,56,115)	(44,44,100)	(45,45,97)	(38,39,92)
黑	(56,47,47)	(45,39,41)	(43,37,39)	(34,30,32)
白	(188,180,182)	(177,166,172)	(166,159,164)	(165,158,164)

二、 實驗二、Colour Matrix 和 QR Code 的效能比較

(一) 實驗二之一、掃描距離

當鏡頭和 Colour Matrix 的距離在 30 公分之內時，Colour Matrix 可被正確偵測到，並成功解碼，顯示程式運作正常。但是當距離超過 30 公分時，邊緣檢測無法成功執行，原因可能是影像邊緣太過模糊，以致於程式偵測的到的邊緣並非實際上 Colour

Matrix 的邊框。當掃描距離在 QR code 的 5 公分內時，QR code 無法被偵測出，距離 5-30 公分時則可以正確辨識及解碼。

表格 6.解像力測試結果

距離		5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30
Colour Matrix	邊緣檢測	O	O	O	O	O	O	O	O	O	O	O
	解碼	O	O	O	O	O	O	O	O	O	O	X
QR code	解碼	X	O	O	O	O	O	O	O	O	O	O

(二) 實驗二之二、拍攝傾角

當 Pi Cam 以 0 至 60 度傾角拍攝 Colour Matrix 時，邊框辨識及解碼成功；當拍攝傾角為 75 度時，邊框辨識及解碼失敗。當使用拍攝傾角 0 至 45 度拍攝 QR code 時，解碼成功；當拍攝傾角為 60 度或以上時，解碼失敗。推測造成 Colour Matrix 邊框辨識及解碼失敗的原因可能是拍攝鏡頭的極限或程式無法克服 Colour Matrix 的影像形變。

表格 7.不同桌面傾角

頃角		0°	15°	30°	45°	60°	75°
Colour Matrix	邊框辨識 (正確/全部)	3/3	3/3	3/3	3/3	3/3	0/3
	解碼 (正確/全部)	3/3	3/3	3/3	3/3	3/3	0/3
QR Code	解碼 (正確/全部)	3/3	3/3	3/3	3/3	0/3	0/3

(三) 實驗二之三、環境照度

在環境光源為 4000K 的 LED 燈及一般室內照度 100-700Lux 下，Colour Matrix 及 QR Code 皆能被掃描成功且正確解碼。顯示實驗一中所建立的顏色辨識模型確實能成功克服日常的環境照度對顏色 RGB 值的干擾，所以本實驗所開發的 Colour Matrix 設計在室內環境下使用基本上是可行的。

表格 8.不同亮度下辨識能力結果

亮度LUX		100	200	300	400	500	600	700
Colour Matrix	邊框辨識 (正確/全部)	3/3	3/3	3/3	3/3	3/3	3/3	3/3
	解碼 (正確/全部)	3/3	3/3	3/3	3/3	3/3	3/3	3/3
QR Code	解碼 (正確/全部)	3/3	3/3	3/3	3/3	3/3	3/3	3/3

三、 實驗三、拍攝多張 Colour Matrix 辨識

多張 Colour Matrix 的實驗中，邊框辨識的成功率為 74.3 ± 5.0%，解碼成功率為 70.2 ± 6.1%，而邊框辨識成功的 Colour Matrix 絕大多數在後續的彩色辨識及解碼步驟也都是成功的。結果顯示邊框辨識應該是決定實驗成功與否的關鍵步驟。

表格 9.Pi Cam 多張 Colour Matrix 辨識

	總數	邊框辨識	解碼成功
1	70	49 (70.0%)	49 (70.0%)
2	70	55 (78.6%)	52 (74.3%)
3	70	55 (78.6%)	51 (72.9%)
4	70	50 (71.4%)	43 (61.4%)
5	70	48 (68.6%)	43 (61.4%)
6	57	45 (78.9%)	43 (75.4%)
7	57	47 (82.5%)	46 (80.7%)
8	70	49 (70.0%)	48 (68.6%)
9	70	52 (74.3%)	50 (71.4%)
10	70	49 (70.0%)	46 (65.7%)
平均值	67.4	49.9 (74.3%)	47.1 (70.2%)
標準差	5.5	3.2 (5.0%)	3.4 (6.1%)

四、 實驗四、鏡頭測試

當使用 Logitech C922 代替 Pi Cam 時，Colour Matrix 的邊框辨識率從 74.3%提高為 99.1%，解碼成功率則從為 70.2%提高為 92.4%。結果顯示高階鏡頭確實能優化 Colour Matrix 的邊框辨識率，進而增加解碼成功的機會。

表格 10.Logitech C922 多張 Colour Matrix 辨識

	CM總數	邊框辨識成功	解碼成功
1	45	44 (97.8%)	44 (97.8%)
2	45	45 (100%)	43 (95.6%)
3	45	45 (100%)	42 (93.3%)
4	45	45 (100%)	44 (97.8%)
5	45	45 (100%)	40 (88.9%)
6	45	44 (97.8%)	41 (91.1%)
7	45	45 (100%)	41 (91.1%)
8	45	44 (97.8%)	41 (91.1%)
9	45	44 (97.8%)	41 (91.1%)
10	45	45 (100%)	39 (86.7%)
平均值	45	44.6(99.1%)	41.6 (92.4%)
標準差	0	0.5(1.1%)	1.6 (3.7%)

柒、 討論

一、 高品質訓練集的建立決定顏色辨識的成功率

在檢視實驗一的數據時，發現顏色的 RGB 值在不同拍攝環境下，數值分布的範圍很大，有時和理論值相去甚遠，有時數值會超乎預期。為了讓建立的機器學習模型能適用於各式各樣的情況下進行正確的顏色辨識，訓練時便需提供足夠、有代表性且多樣化的資料，訓練出的模型預測能力才會好。然而，過多的訓練資料同時也可能會造成過度擬和(overfitting)的現象，因此，建立一個高品質訓練集相當重要。

為了建立一個高品質訓練集，本實驗訓練資料的來源是收集在不同的光源、明暗、遠近、拍攝角度下所得到的 Colour Matrix 影像 RGB 值，再手動標記所對應的顏色，建立第一版的訓練集資料。接著，拍攝數張 Colour Matrix，挑選預測錯誤的資料，進行正確標記後加入更新版的訓練集。重複步驟多次後，觀察到顏色辨識正確率提升至一個穩定程度後，停止收集訓練集的新資料。結果顯示，依此步驟所建立的訓練集能有效在不同拍設環境下進行正確顏色辨識。

二、 訓練模型的參數最佳化

SK learn 的 KNN 分類器，可以調整的參數除了 k 值外，還有權重的參數。權重的參數可以設定為 uniform、distance 或是由使用者自己定義。預設是 uniform，意思是均等的權重，所有的鄰近點的權重都是相等的。distance 是不均等的權重，距離近的點比距離遠的點的影響大。這個實驗的結果是 k=3 時，正確率最高。而權重方面則是

distance 略優於 uniform。因為若 k 值較小，對鄰近的點會非常敏感，因此本研究中 k 取 3，權重採用 distance，其他參數設定則使用預設值。

表格 11.不同參數設定下的 KNN 模型正確率

K WEIGHT	1	2	3	4	5	6	7	8	9	10
DISTANCE	97.4	97.4	99.0	99.0	99.0	99.0	99.0	99.0	99.0	99.0
UNIFORM	97.4	95.8	98.4	94.2	99.0	99.0	99.0	99.0	99.0	99.0

註:上表中數字單位為%

三、 解碼失敗的原因探討

(一) 未偵測出 Colour Matrix

本研究使用的邊框辨識流程是先找出圖像中邊緣的集合，再利用 cv2.approxPolyDP 的方法找出近似的多邊形，如果 Colour Matrix 的邊緣過於模糊、或印有 Colour Matrix 的紙不是很平整、或紙張有摺痕時，會造成找到的多邊形不是四邊形。當程式沒找到四邊形時，就不會進行後續流程(如圖 20)。

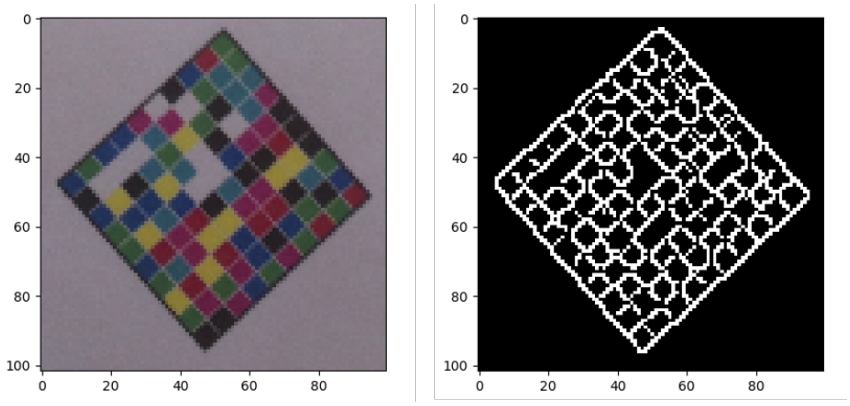


圖 20.Colour Matrix 頂角在邊緣辨識時呈現弧形，導致無法被判斷為四邊形

(二) 定位點顏色辨識失敗

已知相同顏色在不同環境亮度下會產出不同的 RGB 數值。因此，在一個較暗的環境中，黑色-RGB 座標為 (50, 50, 50)，會因為顏色失真而顏色偏向深藍(例如 RGB 座標為 (50, 50, 76))，所以有可能被演算法辨識為藍色。為了避免發生在 RGB 三維空間中的

同一個座標點對應到不同的顏色，可以增加用來訓練演算法的資料量，以及在更多元環境亮度下取得訓練模型用的資料。此外，選取最佳的演算法參數也會影響正確率。

(三) 解碼失敗

即使成功找到 Colour Matrix 的四邊形區域也進行了正確裁切，但由於圖像形變過大，仍可能導致辨識失敗。這常發生在相機拍攝的角度過大時，拍攝的 Colour Matrix 會變形而無法被修正，導致程式在抓取四邊形的四個頂角座標時，不符合實際上 Colour Matrix 的四個定位點位置，造成影像透視轉換失敗。或是當邊緣不清楚的時候，影像解碼時未辨識出 Colour Matrix。此外，當圖像色彩模糊時，也可能發生辨識失敗。

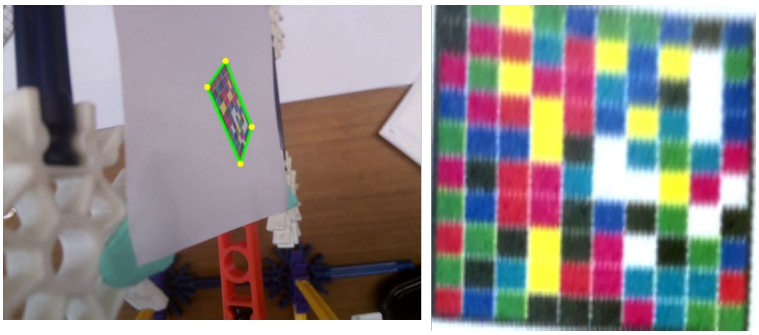


圖 21.左:拍攝角度過大時， Colour Matrix 可以被邊緣辨識 右:因為頂角未被正確抓取，導致影像透視轉換失敗。

(四) 邊緣偵測的限制

由實驗三、四的結果發現，高階鏡頭能增進 Colour Matrix 流程中邊緣檢測能力，進而提升解碼成功率。在掃描距離方面，初步的測試結果顯示更換高階鏡頭只讓可操作距離增加 5 公分(見表格 12)，在實際使用中幫助有限。由這兩個結果推測，更換鏡頭對改善 Colour Matrix 邊框辨識的效果可能侷限於掃描距離在 32.5 公分以內。如果要進一步提高邊框辨識的正確性，可能得嘗試使用人工智慧，例如 Convolutional Neural Network (CNN)，這類方法來取代傳統的 Canny Edge Detection。

表格 12.不同鏡頭對掃描距離的影響

距離		2.5	5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30	32.5	35
Pi Cam	邊緣檢測	X	O	O	O	O	O	O	O	O	O	O	O	X	X
	解碼	X	O	O	O	O	O	O	O	O	O	O	X	X	X
Logitech C922	邊緣檢測	X	O	O	O	O	O	O	O	O	O	O	O	O	X
	解碼	X	O	O	O	O	O	O	O	O	O	O	O	O	X

捌、 結論

1. 本研究設計了一款 8 色 10×10 模組的 Colour Matrix 及編碼方式，並成功在樹莓派平台上開發專屬的顏色辨識及解碼流程。所開發的顏色預測模型可用於照度 100-700 Lux 的室內環境。據我所知，目前市面上尚無相似功能之手持裝置可供比較。

2. 本研究收集高品質的訓練集資料，並測試了決策樹、KNN 及 MLP 三種不同的機器學習演算法對顏色辨識的效能，發現 KNN 是簡單而有效的方法。本研究比較 Colour Matrix 與 QR Code 掃描距離，發現 Colour Matrix 掃描距離無法勝過 QR Code，但有效距離仍可達 32.5 公分。

3. 研究比較 Colour Matrix 與 QR Code 的視角限制，發現 Colour Matrix 在拍攝傾角 0 至 60 度間能正確解碼，而 QR Code 則在拍攝傾角 0 至 45 度間才能正確解碼，顯示 QR Code 較容易受到拍攝傾角的限制。

4. 本研究所建立的顏色辨識模型不受鏡頭周邊變形與不同環境亮度影響。

5. 本研究程式的設計具可支援多張 Colour Matrix 掃描與解碼的優勢。

玖、 使用限制與未來展望

這是我除了交資訊課作業和參加考試之外寫的第一個程式，在撰寫過程中跌跌撞撞吃盡苦頭。在研究結束時，終於發現我只上過 Scratch 和自學 Python 程式，卻沒有基礎電腦入門的基礎 ，以致於在過程中犯了許多最基礎的錯誤，寫的程式邏輯也是非常直白，有經驗的軟體工程師應該不可能如此撰寫。此外，也沒學過影像處理與影像辨識，全部是靠網路自學。如果有機會再修一些課程，適當的老師教導下，換個方式重

新撰寫程式，經過數百億、千億次的機器學習經驗，我相信我的 Colour Matrix 一定會有更好的表現。

一、 使用限制

1. Colour Matrix 需使用彩色輸出，實務上會較使用黑白印刷的 QR code 受限。
2. Colour Matrix 辨識和解碼侷限於拍攝距離 32.5 公分(含)及拍攝傾角 60 度(含)內。

二、 未來展望

1. 改良 Colour Matrix 的定位格設計以增加 Colour Matrix 的辨識性。例如可以放大定位格的尺寸或改變定位格的設計。
2. 使用其他深度學習或人工智慧的方式來進行邊框辨識，例如使用 Convolutional Neural Network (CNN)來取代本實驗中的邊緣偵測，提高邊框辨識的正確性，避免 Canny Edge Detection 的缺點。
3. 加入容錯的設計，來降低因顏色辨識錯誤所造成的解碼失敗。
4. 本實驗只使用了基本的影像處理技術，未來還可以加入 image enhancement、image filtering 等技術，來改善 Colour Matrix 的辨識率。

壹拾、 參考文獻資料及其他

Akkaynak, D., Treibitz, T., Xiao, B., Gürkan, U. A., Allen, J. J., Demirci, U., & Hanlon, R. T. (2014). Use of commercial off-the-shelf digital cameras for scientific data acquisition and scene-specific color calibration. *Journal of the Optical Society of America. A, Optics, image science, and vision, 31*(2), 312-321. Retrieved May 18 2021 from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4028365/

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician, 46*(3), 175-185. Retrieved May 18 2021 from http://www.jstor.org/stable/2685209

Bhardwaj, N., Kumar, R., Verma, R., Jindal, A., & Bhondekar, A. P. (2016). Decoding algorithm for color QR code: a mobile scanner application. *2016 international conference on recent*

trends in information technology (ICRTIT), 1-6. Retrieved May 18 2021 from https://ieeexplore.ieee.org/document/7569561

High Capacity Color Barcodes (HCCB). (2007). Retrieved from https://www.microsoft.com/en-us/research/project/high-capacity-color-barcodes-hccb/?from=http%3A%2F%2Fresearch.microsoft.com%2Fprojects%2Fhccb%2F

John, R. A., & Raahemifar, K. (2015). Designing a 2D color barcode. *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 297-301. Retrieved May 18 2021 from https://ieeexplore.ieee.org/document/7129203

Mishra, A., & Mathuria, M. (2017). A Review on QR Code. *Int. J. Comput. Appl, 164*(9), 17-19. Retrieved May 18 2021 from https://www.ijcaonline.org/archives/volume164/number9/mishra-2017-ijca-913739.pdf

Quinlan, J. R. (1996). Improved use of continuous attributes in C4.5. *J. Artif. Int. Res., 4*(1), 77 – 90. Retrieved May 18 2021 from https://www.jair.org/index.php/jair/article/view/10157/24078

RPi USB Webcams. (2020, September 15 2020). Retrieved from https://elinux.org/RPi_USB_Webcams

Yang, Z., Cheng, Z., Loy, C. C., Lau, W. C., Li, C. M., & Li, G. (2016). Towards robust color recovery for high-capacity color QR codes. *2016 IEEE International Conference on Image Processing (ICIP)*, 2866-2870. Retrieved May 18 2021 from https://ieeexplore.ieee.org/document/7532883

Yang, Z., Xu, H., Deng, J., Loy, C. C., & Lau, W. C. (2018). Robust and fast decoding of high-capacity color QR codes for mobile applications. *IEEE Transactions on Image Processing, 27*(12), 6093-6108. Retrieved May 18 2021 from https://ieeexplore.ieee.org/document/8412566

Zhang, Q., Zhuo, L., Li, J., Zhang, J., Zhang, H., & Li, X. (2018). Vehicle color recognition using Multiple-Layer Feature Representations of lightweight convolutional neural network. *Signal Processing, 147*, 146-153. Retrieved May 18 2021 from https://www.sciencedirect.com/science/article/pii/S016516841830029X

陳志强. (2018). 視網膜動力學之研究及其應用. *中研院訊第 1668 期*. Retrieved May 18 2021 from https://newsletter.sinica.edu.tw/

【評語】052505

此作品開發一套可使用手持裝置快速進行掃描、辨識及解碼的彩色二維條碼(Colour Matrix)套件，在科學研究、探討、和實驗都相當完整且有應用價值，未來可在使用更多數目的顏色會增高辨認錯誤率的這方面進一步進行探討和實驗分析。

作品簡報

作品編號: 052505
作品組別: 高級中等學校組
科　　別: 電腦與資訊學科

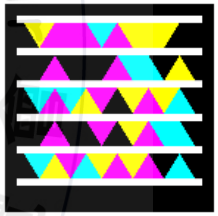
彩色二維條碼手持產品開發之探討

研究動機

- QR Code:
  - 支援單張掃描 ，但實務上有多張條碼掃描的需求。
  - 使用黑白模組為二進位制資料儲存，容量受限。
  - QR Code大小決定掃描距離。最佳掃描距離 : 邊長= 10 : 1
- 透過彩色二維條碼有機會改善QR code限制，但目前市面上無商品化產品。
- 網路上只查到Microsoft曾短暫發表過彩色二維條碼產品
  - Microsoft Tag (High Capacity Colour Barcode)。

(Harald Eversmann,2019)

(: 依規定不能出現QR Code)



Microsoft Tag

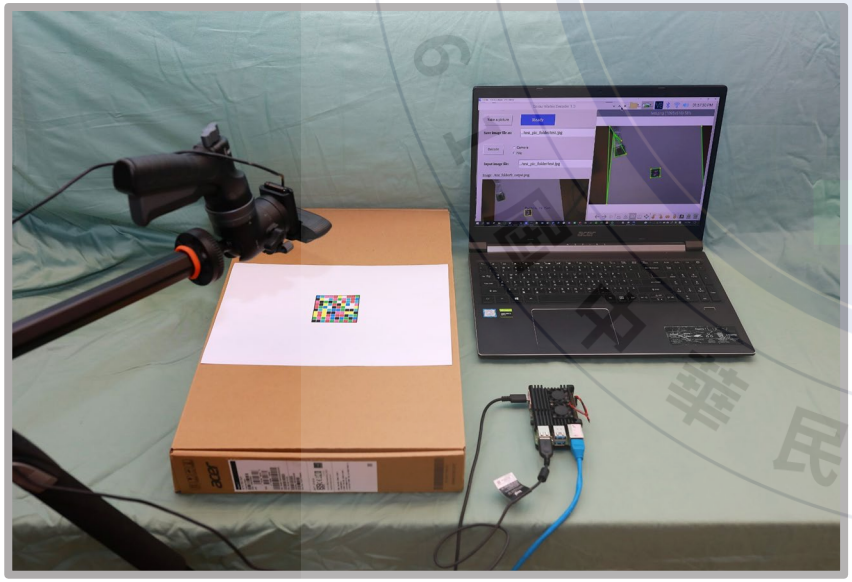
目的

- 設計一款彩色二維條碼，命名為"**Colour Matrix**"，利用多種顏色儲存資料。
- 開發Colour Matrix在手持裝置上讀取的解決方案，執行程式測試及效能優化。
- 以QR Code為對照組，測試Colour Matrix的效能與極限。

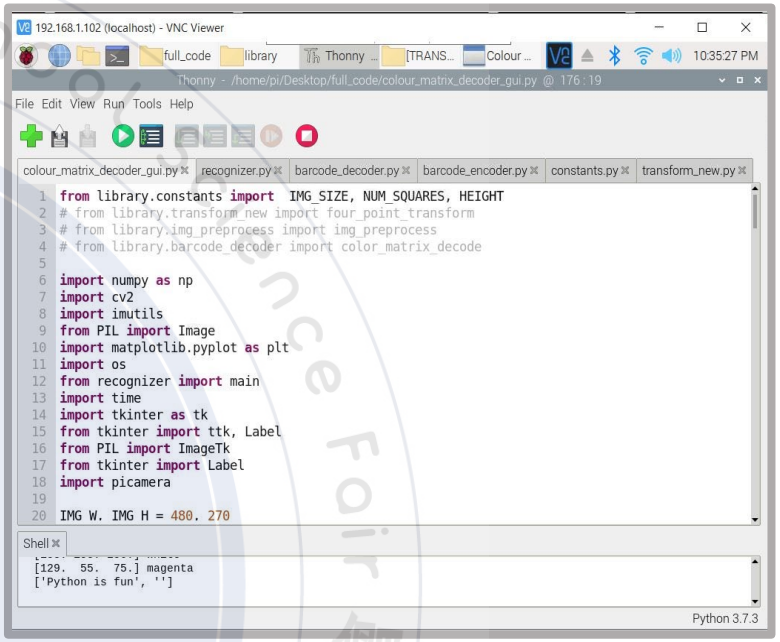
開發環境

- Raspberry Pi OS GNU (簡稱RPi)/Linux 10 (buster)
- Python 3.7.3
- opencv-python 3.4.6.2
- scikit-learn 0.23.1 (SK learn)
- Pyzbar 0.1.8
- Webcamoid
- 相關套件： numpy、scipy、pillow 、 pandas、matplotlib

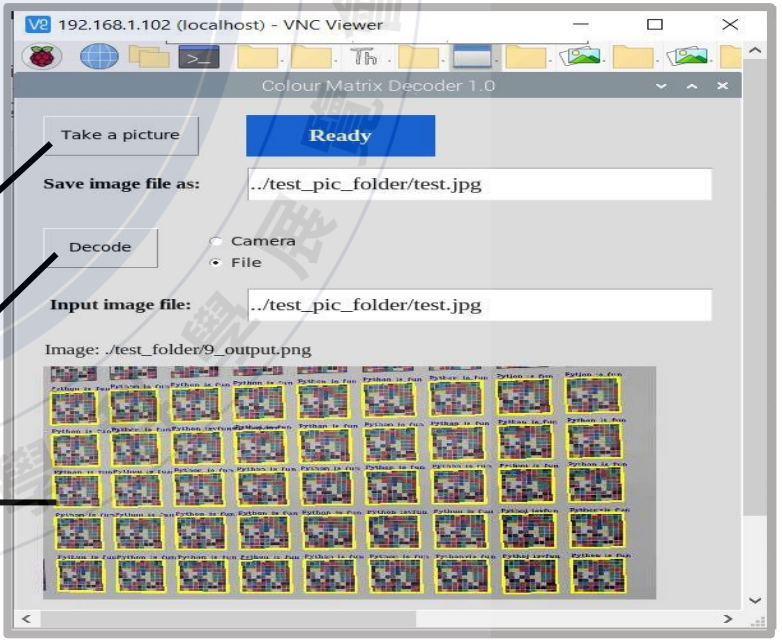
硬體設置: Raspberry Pi 4 Model B



Thonny程式碼編輯器



使用者介面



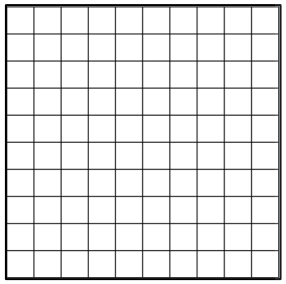
拍照

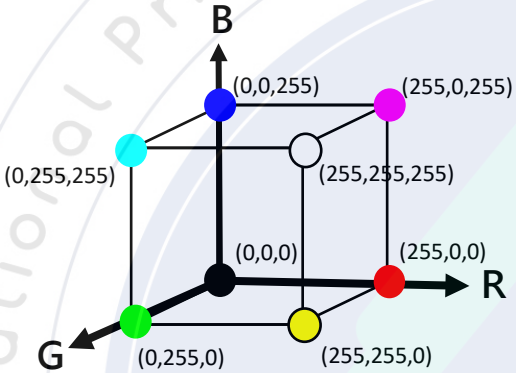
解碼

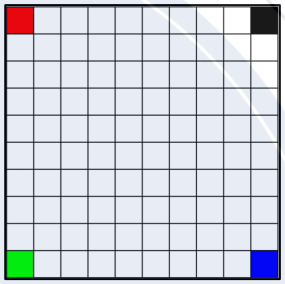
結果顯示

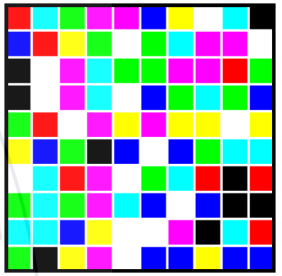
Colour Matrix模組設計

10X10 modules ✚ 8-colours ✚ 4 definition points ➡ Colour Matrix









Colour Matrix編碼流程

以”Colour”為例，使用右側編碼表，將文字轉換成顏色編碼。

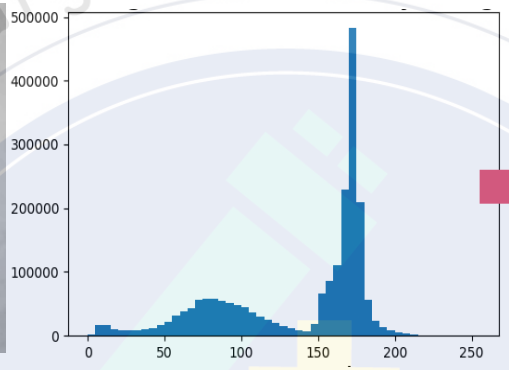
C	o	l	o	u	r
0011100	0001110	0001011	0001110	0010100	0010001

Colours	Encoding
	000
	001
	010
	100
	110
	101
	011
	111

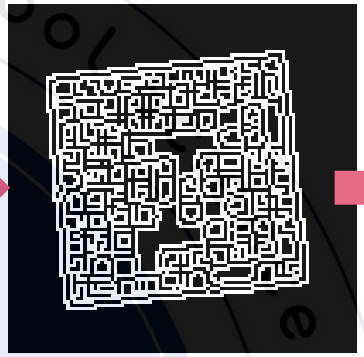
影像前處理流程

Scan image

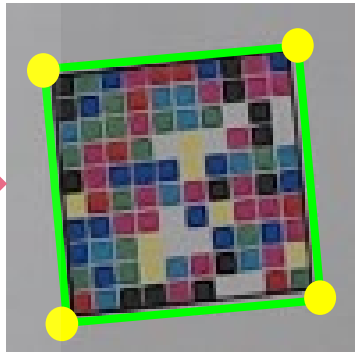
CLAHE gray



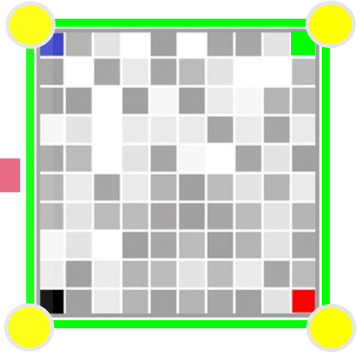
Histogram



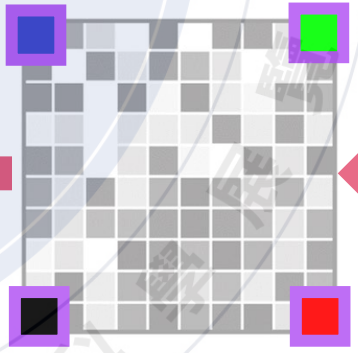
Canny edge +
Adaptive threshold



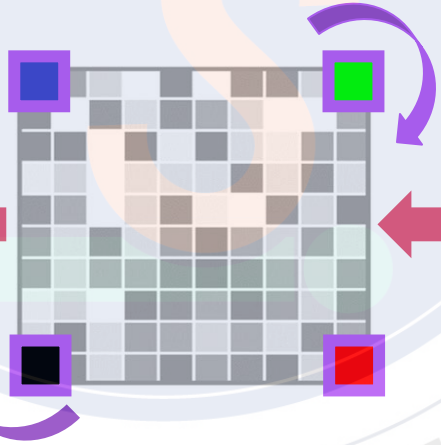
Get image contours
(only get squares)



Four-point
transform



Detect features



Rotate to match

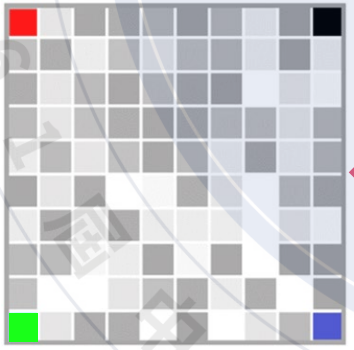


Image registration

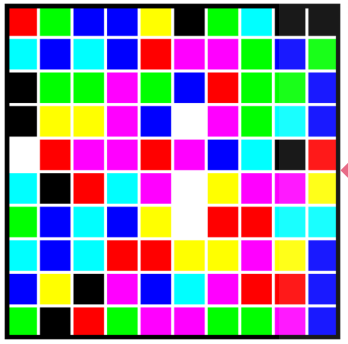
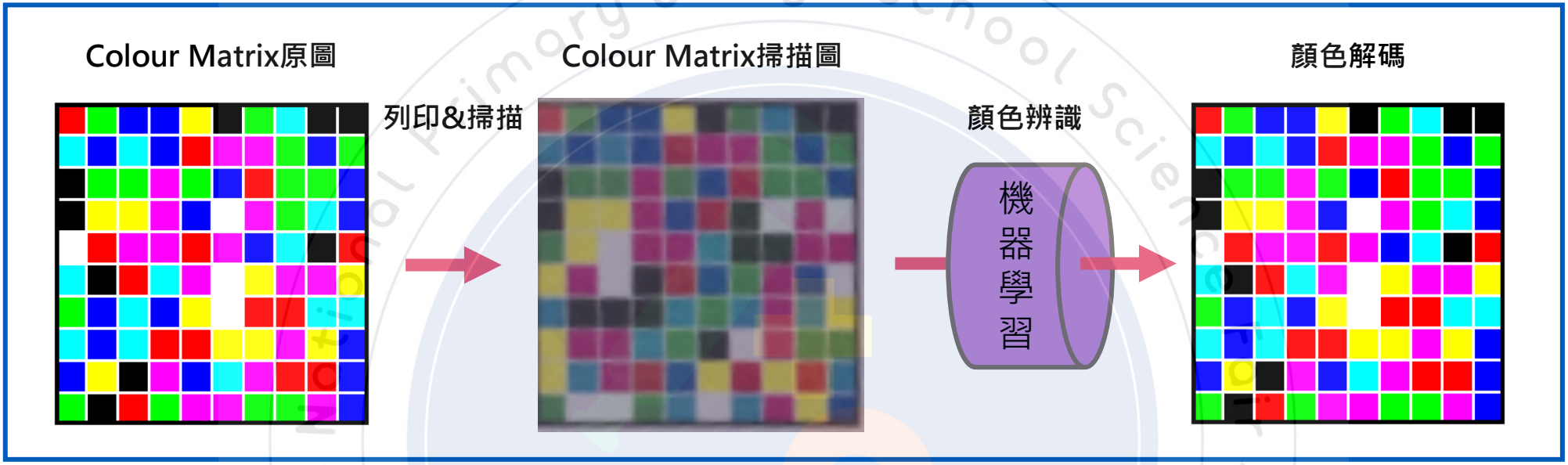


Image display

使用機器學習演算法進行色彩辨識



演算法效能分析

Model	Decision tree	MLP (Layer = 1)	kNN (k = 3)
Accuracy (%)	98.0	99.0	97.4 - 99.0

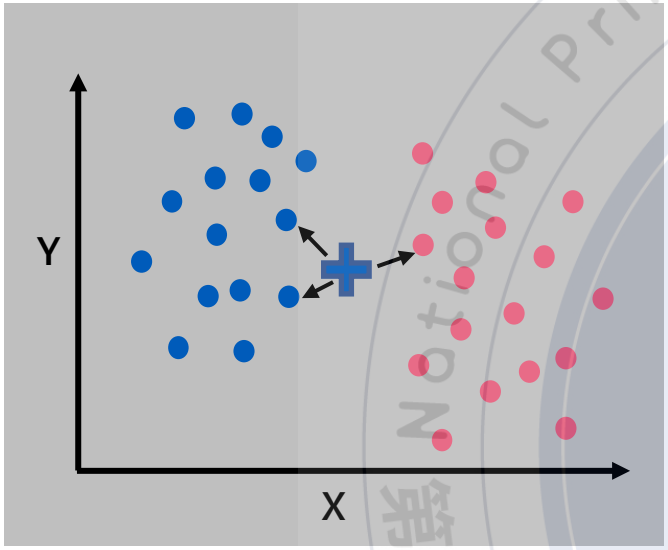
MLP: multilayer perceptron
kNN: k-nearest neighbors

kNN參數調整結果

k	1	3	5	7	9
Weighted	97.4	99.0	99.0	99.0	99.0
Uniform	97.4	98.4	99.0	99.0	99.0

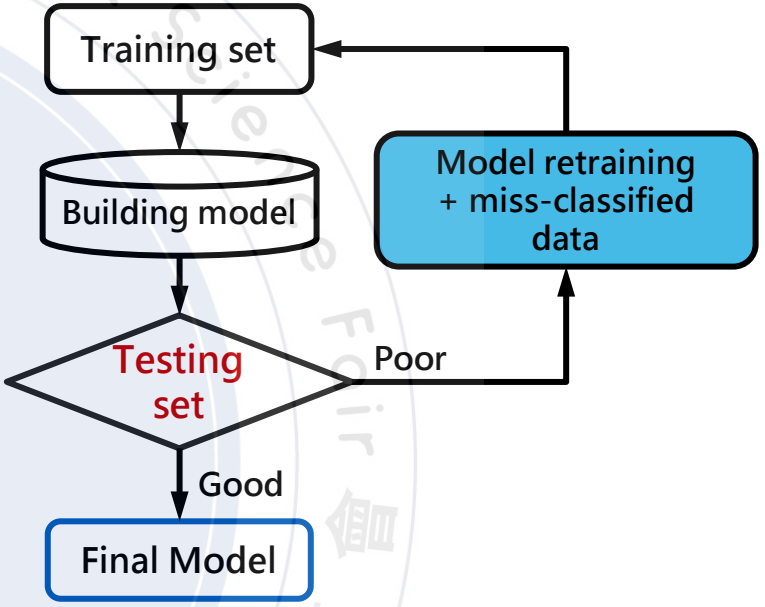
監督式學習(Supervised Learning)

k-nearest neighbors (kNN)

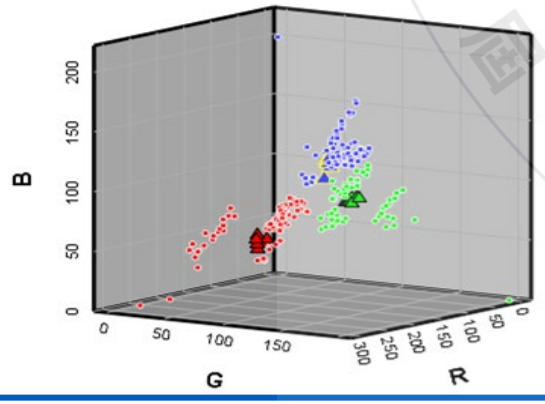


相同顏色在不同環境照度下具相異RGB座標，為提高kNN演算法的顏色預測正確性，透過模型再訓練學習分類錯誤的資料(miss-classified observations)，增加訓練集資料多樣化。

Model retraining



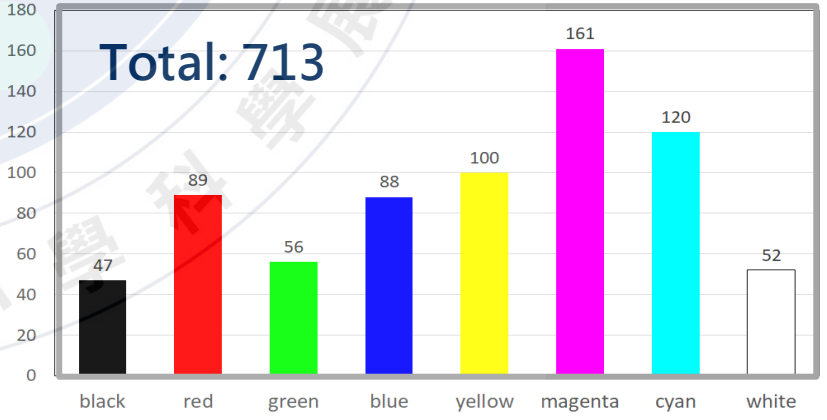
RGB空間中的訓練資料分布



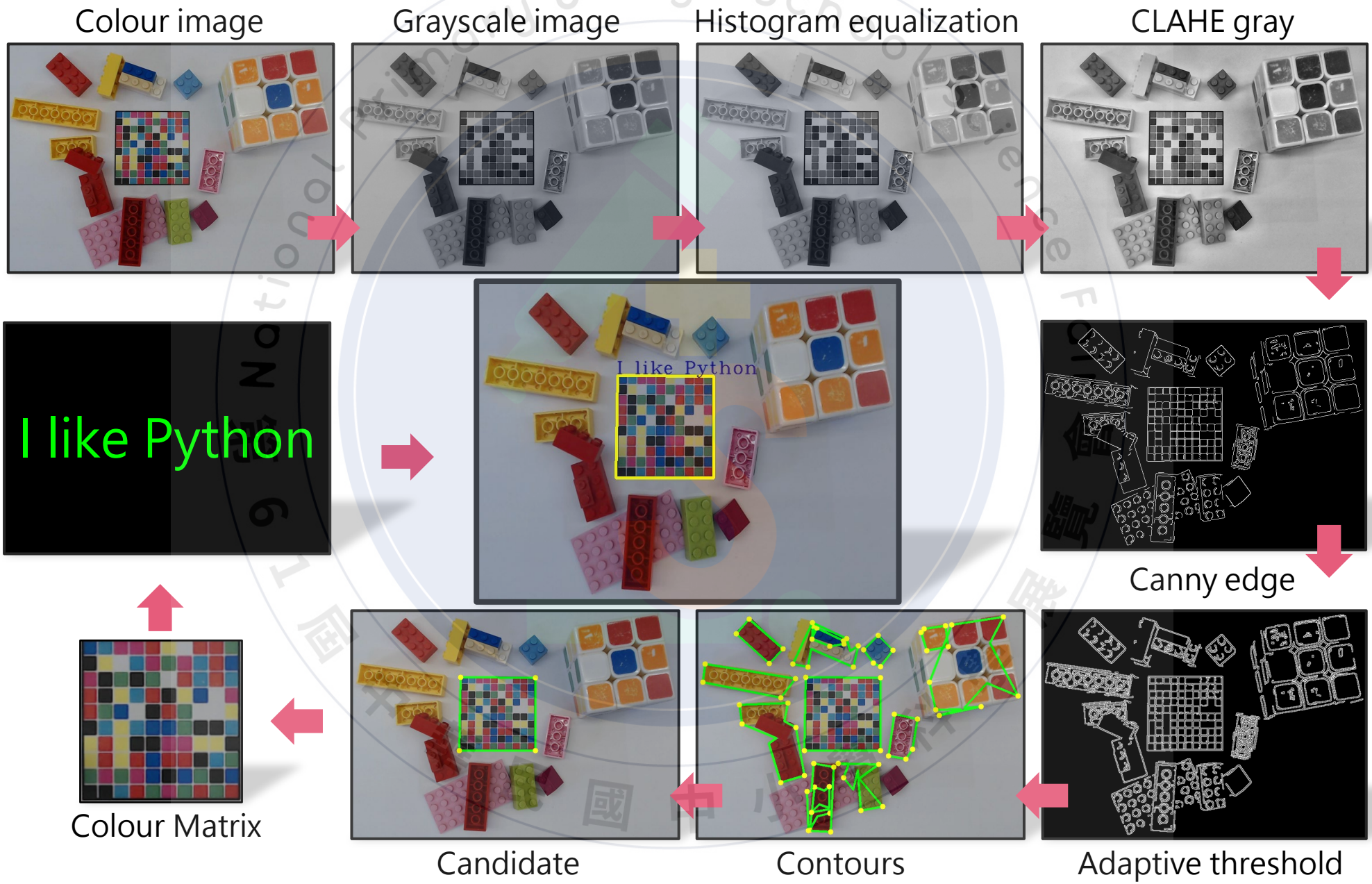
Triangle : original data

Dots : new labeled data (miss-classified observations)

訓練集資料分布

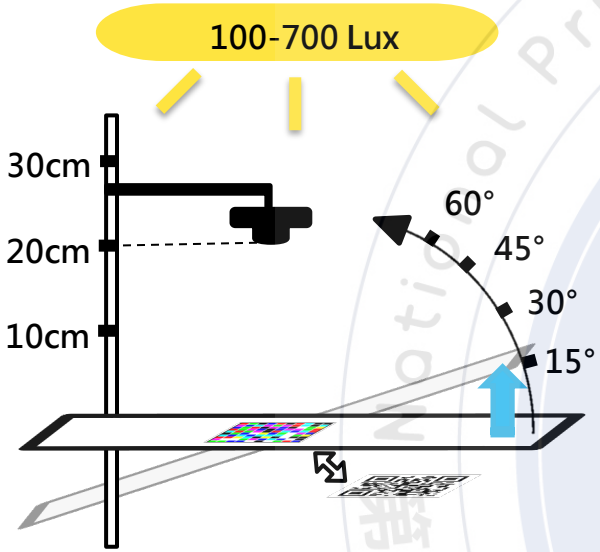


開發方案的"效能實測"


Colour image
Grayscale image
Histogram equalization
CLAHE gray
I like Python
I like Python
Canny edge
Colour Matrix
Candidate
Contours
Adaptive threshold

實驗結果-單張二維碼

Colour Matrix vs QR code



掃描距離評比

Distance (cm)		5	7.5	10	12.5	15	17.5	20	22.5	25	27.5	30
Colour Matrix	Edge	O	O	O	O	O	O	O	O	O	O	O
	Decode	O	O	O	O	O	O	O	O	O	O	X
QR code	success	X	O	O	O	O	O	O	O	O	O	O

掃描傾角評比

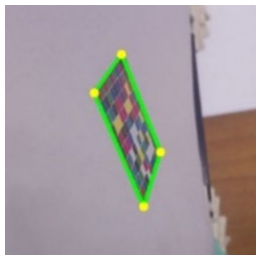
Angle (°)		0°	15°	30°	45°	60°	75°
Colour Matrix	Edge	3/3	3/3	3/3	3/3	3/3	X
	Decode	3/3	3/3	3/3	3/3	3/3	0/3
QR code	success	3/3	3/3	3/3	3/3	0/3	0/3

環境照度評比

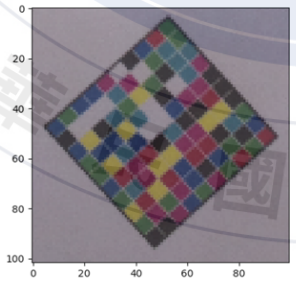
Lux		100	200	300	400	500	600	700
Colour Matrix	Edge	3/3	3/3	3/3	3/3	3/3	3/3	3/3
	Decode	3/3	3/3	3/3	3/3	3/3	3/3	3/3
QR code	success	3/3	3/3	3/3	3/3	3/3	3/3	3/3

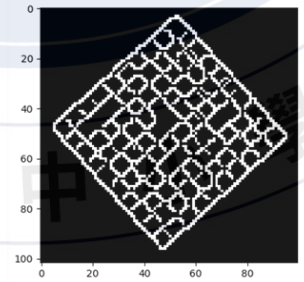
辨識/解碼錯誤檢討

1. 影像形變過大，definition points辨識失敗
--> 邊緣辨識的限制



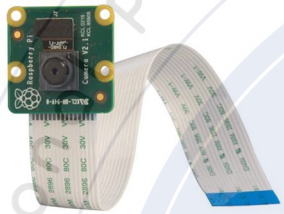
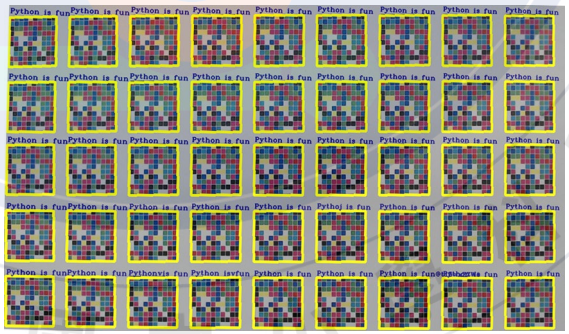
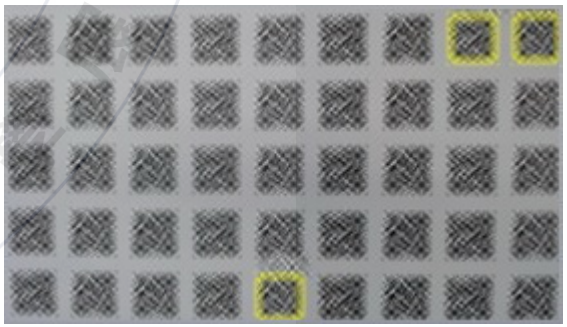
2. 圖像辨識為非四邊形
--> 邊緣辨識失敗





解決方法: 可以加入image enhancement、image filtering等技術，來改善Colour Matrix的辨識率。

挑戰多張掃描

	Colour Matrix		QR Code
鏡頭	Pi Cam v2 	Logitech C922 	Logitech C922 
規格	塑膠鏡頭/手動對焦	玻璃鏡頭/自動對焦	玻璃鏡頭/自動對焦
解析度	Full HD/1080p30fps	Full HD/1080p60fps	Full HD/1080p60fps
邊框辨識率	74.3% →	99.1%	---
解碼成功率	70.2% →	92.4%	6.7%
結果			 已馬賽克處理 (依規定不能出現QR Code)

結論

- 本研究設計一款8色10×10模組的**Colour Matrix**，整合RPi平台及開源程式，利用Python以及多個資源包撰寫800行程式，成功開發出一套完整的彩色二維條碼產品。
- 完成具有視覺化的使用者介面。
- 本研究成功在RPi平台上開發專屬的顏色辨識及解碼流程。
- 本研究測試結果，kNN為最佳顏色辨識的機器學習演算法。
- 本研究挑選照度100-700Lux之間的資料來增加訓練集的多樣性。此外，還挑選被機器學習分類錯誤的資料(miss-classified observations)，重新進行標示後加入訓練集，進行模型再訓練(Model retraining)。透過優化訓練集，成功提高顏色辨識正確性。
- 本研究設計的邊框辨識流程用來尋找Colour Matrix 的有效辨識距離可達32.5公分，支援拍攝傾角0至60度，與pyzbar解碼QR Code的能力相當。
- 本研究開發的Colour Matrix方案在多張掃描與解碼方面，效能明顯較使用pyzbar辨識QR Code佳，可擴增條碼的使用範圍。
- 本研究所建立的顏色辨識方法不受鏡頭周邊變形的影響。
- 同樣空間，Colour Matrix儲存容量為QR Code 的3.4倍。

未來展望

- 改進Colour Matrix設計，以提高辨識成功率。例如放大或改變定位格設計。
- 選用其他邊框辨識方法來克服Canny Edge + Adaptive Threshold的極限。可考慮深度學習或人工智慧的演算法，例如Convolutional Neural Network (CNN)。
- 加入容錯設計，降低因Colour Matrix毀損或顏色辨識錯誤所造成的解碼失敗。

參考文獻資料及其他

1. High Capacity Color Barcodes (HCCB). (2007). https://www.microsoft.com/
2. John RA, et al. (2015). Designing a 2D color barcode. Paper presented at the IEEE 28th Canadian Conference on Electrical and Computer Engineering.
3. Bhardwaj N, et al. (2016). Decoding algorithm for color QR code: a mobile scanner application. Paper presented at the international conference on recent trends in information technology.
4. Yang Z, et al. (2018). Robust and fast decoding of high-capacity color QR codes for mobile applications. IEEE Transactions on Image Processing, 27(12), 6093-6108.
5. Eversmann H. (2019). *Computation offloading of augmented reality In warehouse order picking*. Bachelor's thesis, University of Twente, Netherlands.