**2019 Footprintku校園達人秀**

**〈以I為名，AI動起來〉**

**初賽報告**

中華民國108年 月 日

隊名： 圖片辨辨辨

1. 為維護比賽公平性，**請勿於報告中洩露比賽隊伍之學校與教授名稱。**
2. 敬請依此格式撰寫報告，若有不適用之項目，**該項目請留空白，切勿刪除。**
3. 內文字體大小以12字體，行距以1.5行距為原則。
4. 報告得以中文或英文撰寫。
5. 報告內容篇幅以15頁為限(不含封面、目錄、圖目錄、表目錄、摘要及參考文獻，摘要不得超過一頁)。
6. 請以PDF為檔案格式提交。

目錄

[中文摘要 5](#_Toc27609932)

[壹、 緒論 6](#_Toc27609933)

[1-1、 研究背景與動機 6](#_Toc27609934)

[1-2、 研究目的與方法 6](#_Toc27609935)

[1-3、 研究流程與架構 6](#_Toc27609936)

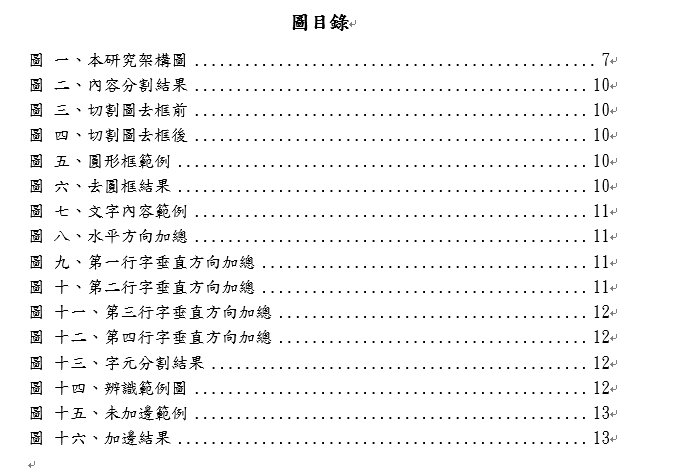
[貳、 背景知識與相關文獻 8](#_Toc27609937)

[參、 研究方法 9](#_Toc27609942)

[肆、 研究結果與分析 13](#_Toc27609943)

[伍、 結論與未來展望 14](#_Toc27609944)

[陸、 參考文獻 15](#_Toc27609945)





# 中文摘要

具影像辨識功能的演算法是在國內外都被列為重要的智慧工業與生活應用的重要技術。本次研究主要分為二階段，第一階段先取得表格內容跟標籤位置，並將兩者對應，第二階段以文字辨識方法取得表格跟標籤內容。

針對第一階段，我們使用OpenCV對圖像做前處理，並運用Numpy處理矩陣與計算。圖片大致分為兩種，以格線分開，或以間隔分開。以格線所分開的圖片可直接依框線切割，另外的圖像有固定間隔為特徵。利用這些特徵，能將各個座標格分開，並利位置對應其座標。

第二階段則利用pytesseract來辨識每個座標格的內容，並將其輸出到execl。

關鍵字：ORC, pytesseract, 座標辨識, 圖片前處理, OpenCV, Numpy, Pillow, Pytesseract, Pytorch

1. 緒論
   1. 研究背景與動機

隨著AI在人們日常生活中逐漸佔有重要的角色，圖像辨識也逐漸廣泛應用在生活中。例如手機的相機功能讓人們隨身攜帶紀錄影像，此時人們若想將所見的知識留存，圖片轉換為文字便能方便編輯與註記。因此若有優秀的圖片轉文字軟體，藉由AI處理可減少人們花在打字的時間，也方便資訊的流通與紀錄。

* 1. 研究目的與方法

本次研究藉由圖像處理與字元解析，來實現多張圖像內的文字辨識。先辨識各種不同顏色、樣式的圖片，再取得圖像表格中的內容與其對應的座標，最後將其結果寫入Excel檔，並以提高辨識準確率為主要目標。團隊以Python作為工具，使用openCV、Numpy、Pytorch、 Pytesseract...等套件找出圖片特徵，對圖片做前處理、訓練模型，調整參數並反覆實驗，使辨識準確率提高。

* 1. 研究流程與架構

在研究流程上，團隊先參考多種方法後，決定將工作分為兩部分同時進行， 一部分處理圖片，二值化、切割、去噪...等前處理以取得表格內容，另一部分尋找適合辨識文字的套件與模型，最後將兩者整合並輸出。

本研究架構如圖一所示，在物件偵測的研究中，團隊找到主流的方法像是YOLO、connented component、cv.findContours；原本團隊用於切割字串做訓練方法，團隊也發現或許可以用在取得表格位置上。

到研究的後半段，團隊決定選用Pytesseract作為辨識方法。為了提升準確率，除了調整Pytesseract的參數，還對圖片前處理部分進行改善，例如：調整文字的背景、將圖片切塊後重組、將圖片放大，最後將兩者整合並輸出。

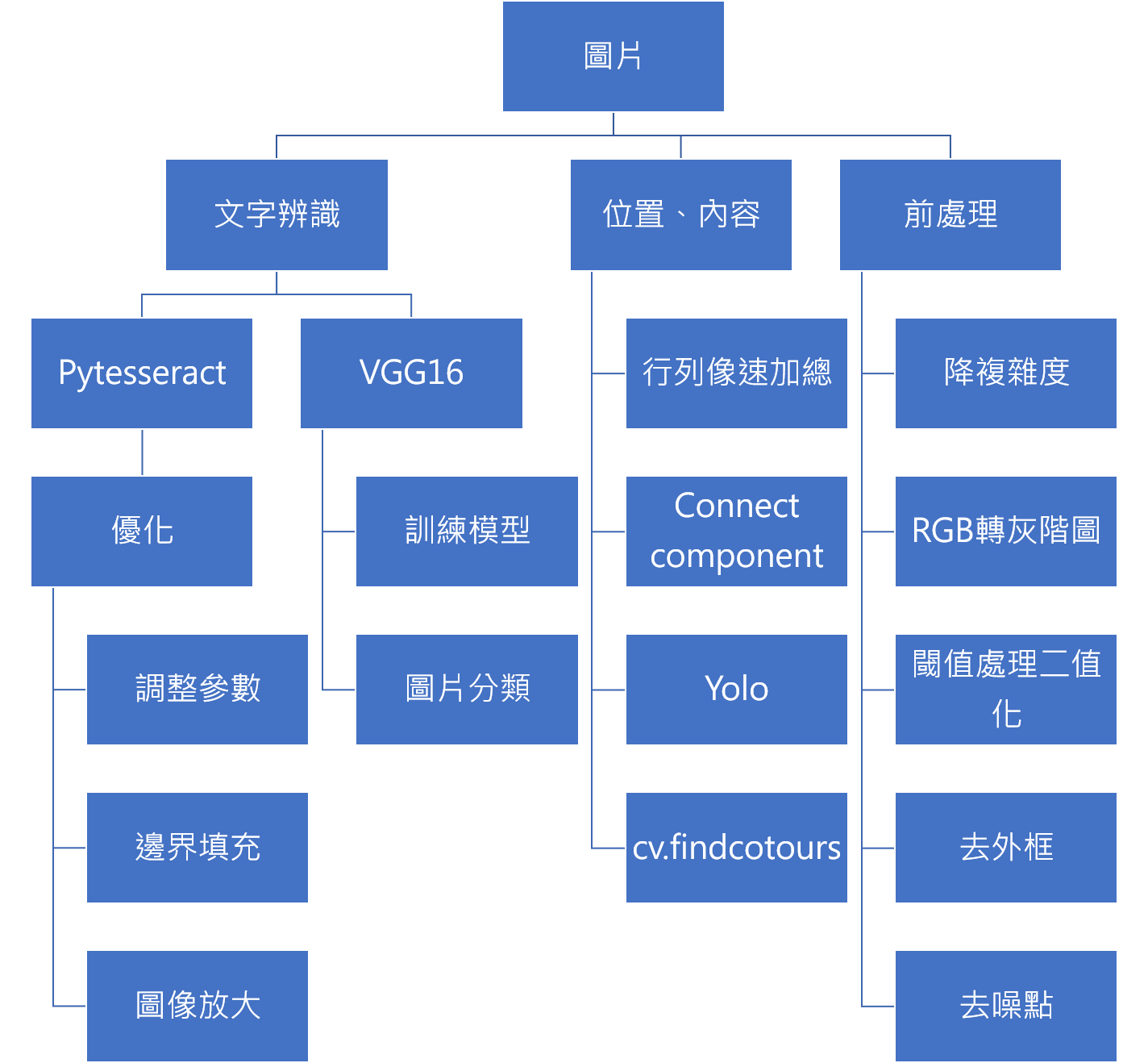


圖 一、本研究架構圖

1. 背景知識與相關文獻

在表格內容切割方面，我們使用影像垂直與水平投影分析、連通域分析(connected component)、影像輪廓查詢(cv.findcontours) 、以及Adaptive thresholding等方法。

* + 影像垂直與水平投影分析

此方法是使用在字串辨識時，將字元分割的的方法，文字間會有間隔，因此對於像素不為0的位置設為起始點，為0的位置設為終點，就可獲得字元所在位置。

* + 連通域分析(connected component)

為圖論內經典的方法，將連接起來的像素視為一個物件，opencv也有支援的函式。

* + 影像輪廓查詢cv.findcontours

為opencv內的一個函數，可以用來偵測圖片內的物件輪廓，並將其存為一個物件。

* + Adaptive thresholding

是一種決定threshold的方式，決定threshold 的目的在於將圖片二值化，也因此決定threshold相當重要。比起傳統的方法為global threshold，adaptive thresholding決定threshold的方式會考慮其附近的pixel，以達到考慮不同區塊差異的效果

文字辨識軟體方面，Pytesseract套件是支援python的tesseract，而tesseract是一套開源的OCR軟體，目前由google管理，這個模型已經經由多種字型的dataset訓練過，是相當成熟的模型，目前已更新到第四版，並被利用在各種應用上。VGG為Deep learning中的一大經典模型，他主要的貢獻是將CNN透過較小的Conv堆疊使模型能夠變得更 “深”，是目前CNN圖像分類的主流模型。

1. 研究方法
   1. 預處理

對於要偵測以及辨識的圖片，只留下其重要的特徵，可以減少資訊量。對於文字辨識來說，不需要顏色資訊，因此可以將彩圖轉為灰階圖降低複雜度。

灰階圖可再藉由閾值處理將背景及物件分為0與1，減少運算量。傳統的二值化方法為利用單一的閥值，會忽略局部的變化，例如陰影、顏色的影響，故團隊利用自適應閾值(adaptive threshold)方法對圖片二值化，更適合內容複雜度高的圖片。

* 1. 取得布局位置

在座標判斷以及切割部分，利用像素加總的方式來獲取座標格與標籤的位置資訊，圖片經過像素行列方向加總，空白間隔處值為零，先將不為零即有物件的位置設為切割上界，再選定下個為零的位置設為切割下界，可以獲得內容分布的位置。這個方法的優點在於計算量相當的小，極具效率。

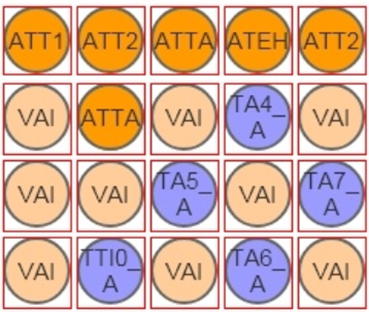


圖 二、內容分割結果

* 1. 消除框線

在文字辨識中，多餘的框線會影響辨識跟偵測的結果，故選擇將框線去除。作法則是利用矩陣水平及垂直加總，框線在水平及垂直加總後，會產生極大值，即可得到位置資訊，並令其為零消除框線。

在我們遇到的圖片中，不只有方框，也有圓框的的資料，這外圍的框一樣會影響辨識率。由於我們間隔的方式，已經是圓的切線，代表我們可以取得圓的半徑，利用半徑，我們可以有效的將圓框去除。

|  |  |
| --- | --- |
| 圖 三、切割圖去框前 | 圖 四、切割圖去框後 |
| 圖 五、圓形框範例 | 圖 六、去圓框結果 |

* 1. 切字元

字句通常整齊排列且會有分行的特性，每行的字元又可以垂直的空白分割，利用此特性並可將字元分割。以圖X為例，透過水平方向的加總，以選定閥值可將分為四個區塊(圖1)，會發現依序為第一行字，第一行底線，第二行字，第二行底線；再將分割後的四張圖片個別垂直加總(如圖2.3.4.5)，將字元與字元之以零為閥值分割，結果如圖6。

|  |  |
| --- | --- |
| 圖 七、文字內容範例 | 圖 八、水平方向加總 |
| 圖 九 、第一行字垂直方向加總 | 圖 十、第二行字垂直方向加總 |
| 圖 十一、第三行字垂直方向加總 | 圖 十二、第四行字垂直方向加總 |



圖 十三、字元分割結果

* 1. 辨識
     1. 調整參數

Pytesseract有提供不同參數可以調整，根據不同情況使用，可以提高正確率。

|  |  |  |
| --- | --- | --- |
| 圖 十四、辨識範例圖 | 未加參數(預設)  (config='-psm 3 ') | 輸出:  3\_1\_1\_1\_10 |
| config='-psm 7 digits' | 輸出:  BLLRXP1 |

表 一、調整參數結果

* + 1. 圖片邊緣加寬

原圖測試無法辨識出任何結果，在經過邊界填充後，Pytesseract就可以辨識了(如附表)。合理推測與其模型訓練方式有關，一般來說訓練字元模型時，不會恰好切齊字元的邊界，故經過邊界填充較不影響Pytesseract辨識結果

|  |  |  |
| --- | --- | --- |
| 原圖 | 圖 十五、未加邊範例 | 輸出: |
| 處理後 | 圖 十六、加邊結果 | 輸出:  JK |

表 二、加邊辨識結果

* + 1. 圖片放大

我們發現若將圖片resize成較大的圖片，辨識效果會更好，們將151\*117的圖片長寬都增加20倍，如此可以增加其辨識率

1. 研究結果與分析

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 二值化後 | 去除圓外框 | | 再分割 | |
|  |  | |  | |
|  |  |  | |  |

表 三、字元切割結果

|  |  |  |
| --- | --- | --- |
|  | **python 完整程式執行時間(秒)** | **exe執行檔測試時間(秒)** |
| **FPK\_1** | **40** | **73** |
| **FPK\_2** | **534** | **172** |
| **FPK\_3** | **28** | **99** |
| **FPK\_4** | **183** | **177** |
| **FPK\_5** | **47** | **94** |
| **FPK\_6** | **142** | **161** |
| **FPK\_7** | **90** | **114** |
| **FPK\_8** | **36** | **53** |
| **FPK\_9** | **27** | **63** |
| **FPK\_10** | **81** | **95** |

表 四、各圖執行時間

1. 結論與未來展望

本研究主要利用像素加總切割方法、圖片前處理、與Pytesseract辨識解決本次影像辨識問題，其優點如下

* 減少訓練時間

訓練模型是相當費時費力的，並且不同結構，不同dataset，都會影響辨識出來的結果，團隊使用pytesseract進行辨識，使得此實驗是每個人都可重複的，不會因為model的不同，而影響實驗結果。

* 有效率的位置取得

團隊用像素加總切割方法來取得位置，在通用的情況下，要使用這種方法來取得物件是相當複雜的，然而因為表格對齊的特性，使得這種方法相當適合作為這裡的切格方法，加上其計算複雜度低，運算量很小，能在極短的時間就做到其應有的效果。

相關缺點如下

* pytesseract輸出類別的限制

pytesseract是針對文件辨識用，對於特殊的需求，例如本次辨識的圖片中字元有overline的案例，需以其他方式判別。

* 前處理複雜

為了要讓pytesseract有更好的結果，團隊嘗試多種圖像前處理，有些看似能夠改進的方法，對pytesseract卻不一定有效。

未來可突破之處，Pytesseract是經過多筆資料與時間所訓練出來的模型，跟團隊自己所訓練出來的model相比，相對成熟穩定許多，因此團隊決定使用它作為團隊的辨識方法，但遇到辨識錯誤率高的情況，要修改Pytesseract的模型就相對困難，團隊只能選擇在前處理多下心思。

1. 參考文獻

加入報告當中參考文獻、請使用IEEE2006年樣式插入引文與排列參考文獻

# 引用的項目

|  |  |
| --- | --- |
| [1] | T. Lung-Yu, “影像垂直/水平投影分析,” [線上]. Available: http://honglung.pixnet.net/blog. |
| [2] | M. B. Dillencourt, H. Samet 且 M. Tamminen., “A general approach to connected-component labeling for arbitrary image representations,” *Journal of the ACM (J. ACM),* pp. 253-280, April 1992. |
| [3] | “OpenCV 3.0.0-dev documentation,” [線上]. Available: https://docs.opencv.org/3.0-beta/modules/imgproc/doc/structural\_analysis\_and\_shape\_descriptors.html#connectedcomponents. |
| [4] | “OpenCV 2.4.13.7 documentation,” [線上]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/structural\_analysis\_and\_shape\_descriptors.html?highlight=findcontours#findcontours. |
| [5] | K. Simonyan 且 A. Zisserman, “ Very Deep Convolutional Networks for Large-Scale Image Recognition,” 於 *International Conference on Learning Representations* , 2015 . |
| [6] | “虫数据,” [線上]. Available: http://chongdata.com/articles/?p=32. |
| [7] | “OpenCV-Python Tutorials,” [線上]. Available: https://docs.opencv.org/master/d7/d4d/tutorial\_py\_thresholding.html. |