**車牌辨識系統**

學生:黃翔毓 指導老師:王維倫

陳映宇

蔡易辰

正修科技大學資訊工程系

**摘要**

在現今的社會，汽機車已經成為每個人的交通工具，車輛的使用也變得越來越頻繁，但也衍生出了相當多的交通及治安問題，例如：交通違規、車禍肇事、車輛竊盜等情形。當案件發生後，我們藉由車牌號碼，就能夠快速的找出車主，以利警政單位及監理機關執行公權力。以交通執法為例，現在所使用超速與闖紅燈自動照像設備，將取得違規影像，再以人工來判斷車牌號碼的方式進行違規舉發。可是近年來車輛成長快速，警政單位投入大量的人力在車牌號碼判讀，增加了警察人員的工作負擔。因此，本論文為提出一個自動化的車牌辨識系統，就可以有效減少人力及政府的財政負擔。此車牌辨識系統可應用於違規車輛的車牌辨識、車輛門禁管制及停車場的車輛管理等。

**目錄**

摘要.............................................................................................................1

目錄.............................................................................................................2

1. 緒論. .............................................................................................3

1.1 專題題目....................................................................................3

1.2 研究動機與目的.........................................................................3

1. 相關研究......................................................................................4

2.1 邊緣偵測...................................................................................4

2.2 二值化......................................................................................5

2.3 樣板比對…...............................................................................6

2.4 Open CV…….……………………………………………...……...7

第 三 章 軟體需求分析………………………………………………………..8

3.1 軟體設計………………………………………………………...…...8

第 四 章 測試功能……………………………………………………...……..12

4.1 辨識畫面………………………………………………………….....14

第五章總結………………………………………………………………..…...…16

參考文獻資料………………………………………………………………..…...17

1. **緒論**

**1.1 專題題目 車牌辨識系統**

**1.2 研究動機與目的**

**動機:**

由於汽車已經變成人們主要的交通工具,車輛的管理顯得更為重要,傳統用人工進行管理的方法不符合經濟成本,於是想藉由對車牌的辨識,應用於對車輛的管理,並藉由實作影像處理,了解影像處理的原理,鑒於甚多的應用車 牌大多以人工的方式來判讀，當判讀時間長，或是讀取車牌號碼數量大時，錯誤率就會提高，加上處理程序曠日費時，十分的不便，如果可以將所拍到的影像直接的利用電腦來做處理與辨識，對於降低管理人員的負擔，減 少人力成本的花費，可以有顯著的助益。

**目的:**

我們研究的目的是為減少人力成本的支出,也可以達到兩全其美的辦法。首先利用即時截取的方法，拍取所需要的照片，接著進行照片的車牌偵測，利用影像的邊界，和照片中灰階和二值化後的陰影，去判斷出車牌的位置，接著使用影像處理來分割背景和車牌上的字體，利用分割下來的字體，藉著採取字體行跟列的黑點個數，此流程可以成功的得知照片中車牌的字型。透過此系統來達到管理的功能，利用即時擷取，再進行車牌辨識功能，可以將每輛車都正確的管理，達到節省人力的方法。

**第二章 相關研究**

### 2.1邊緣偵測

邊緣偵測的目的是標識數字圖像中亮度變化明顯的點。圖像屬性中的顯著變化通常反映了屬性的重要事件和變化。利用相鄰像素的差距找出物體的邊緣，因此像素差距越明顯就能更清楚的表達出邊緣。找出正確、明顯的邊緣就能更有效率的辨識車牌位置。車牌會因為拍攝角度不同，而產生不一樣的形狀，也會因為天氣、亮度造成拍攝的結果而有所不一樣。

sobel邊緣偵測使用一個 2 的權重值，是想給予更多的重要性來達成某種程度的平滑，主要是用 3x3 的矩陣來找出邊緣像素的灰階值，如果一個點和其他的八個點是相同，表示沒有變化，其值等於零。如果兩邊的值不同，Gx輸出就不會等於零。對於每點的Gx 與 Gy 計算結果會累加起來存於另一張圖上，以確保兩個方向的邊緣變化都會被找到如下

-1 -2 +1 -1 +0 +1

Gx= +0 +0 +0 Gy= -2 +0 +2

+1 +2 +1 -1 +0 +1

* + 1. Ga 為水平遮罩係數(b)Gb 為垂直遮罩係數

#### **二值化**

二值化是圖像分割的一種方法。在二值化圖像的時候把大於某個臨界灰度值得像素灰度設為灰度極大值,把小於這個值得像素灰度設為灰度及小值, 從而實現二值化。

二值化的執行是利用 pixel-by-pixel operation 把一張灰階圖片變成一張只有黑與白二值化(canny)的圖。



#### **樣板比對**

形狀檢測辨識最直接的方法，就是把所有標準圖形事先儲存在電腦裡。當電腦遇到一個待測圖形時，就把該圖形與所有事先儲存的圖形庫資料一一叫出來比對，比對出來最接近該待測圖形的標準圖形即可達到形狀檢測辨識的目的。而簡單的比對方法，就是把兩個圖形放置在同一個位置上，然後令兩個影像相減，或是計算兩個影像之間重疊的面積，通常這樣做也會耗去很多時間。這樣比對的方法常常需要精確的定位，否則兩個影像也許只是位置錯開了，但內容是相同的，相減後許多點素仍會殘留灰度值，如此會把同一個影像的兩種不同形式判定為兩個不同的影像。因此在形狀檢測辨識以前應該對標準圖形作一些結構的分析，然後根據結構的特徵來從

事形狀檢測辨識與比對，而樣板（template）即可視為具有影像結構特徵之子影像，利用樣板來從事形狀比對的工作，有助於簡化整個形狀檢測辨識的流程。



#### 

#### **2.4 Open CV**

OpenCV 的全稱是 Open Source Computer Vision Library， 是一個跨平台[的計算機視覺](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89)庫。OpenCV 是由[英特爾公司](http://zh.wikipedia.org/wiki/%E8%8B%B1%E7%89%B9%E5%B0%94)發起並參與開發，以 [BSD 許可證](http://zh.wikipedia.org/wiki/BSD%E8%AE%B8%E5%8F%AF%E8%AF%81)授權發行，可以在商業和研究領域中免費使用。OpenCV 可用於開發實時的[圖像處理](http://zh.wikipedia.org/wiki/%E5%9B%BE%E5%83%8F%E5%A4%84%E7%90%86)、[計算機視覺](http://zh.wikipedia.org/wiki/%E8%AE%A1%E7%AE%97%E6%9C%BA%E8%A7%86%E8%A7%89)以及[模式識別](http://zh.wikipedia.org/wiki/%E6%A8%A1%E5%BC%8F%E8%AF%86%E5%88%AB)程序。該程序庫也可以使用英特爾公司的 [IPP](http://zh.wikipedia.org/w/index.php?title=%E9%9B%86%E6%88%90%E6%80%A7%E8%83%BD%E5%9F%BA%E5%85%83&amp;action=edit&amp;redlink=1) 進行加速處理。

OpenCV 項目最早由英特爾公司於 1999 年啟動， 致力於 CPU 密集型的任務，是一個包括如[光線追蹤](http://zh.wikipedia.org/wiki/%E5%85%89%E7%B7%9A%E8%BF%BD%E8%B9%A4)和 [3D 顯示](http://zh.wikipedia.org/w/index.php?title=3D%E6%98%BE%E7%A4%BA&amp;action=edit&amp;redlink=1)的計劃的一部分。早期 OpenCV 的主要目標是

* 為推進機器視覺的研究，提供一套開源且優化的基礎庫。不重複發明輪子。
* 提供一個共同的基礎庫，使得開發人員的代碼更容易閱讀和轉讓，促進了知識的傳播。
* 通過提供一個不需要開源或免費的軟體許可，促進商業應用軟體的開發。
* OpenCV 現在也集成了對 [CUDA](http://zh.wikipedia.org/wiki/CUDA) 的支持.

OpenCV 的第一個[預覽版本](http://zh.wikipedia.org/wiki/Alpha)於 2000 年在 [IEEE Conference on Computer](http://zh.wikipedia.org/w/index.php?title=Conference_on_Computer_Vision_and_Pattern_Recognition&amp;action=edit&amp;redlink=1) [Vision and Pattern Recognition](http://zh.wikipedia.org/w/index.php?title=Conference_on_Computer_Vision_and_Pattern_Recognition&amp;action=edit&amp;redlink=1) 公開，並且後續提供了五個[測試版本](http://zh.wikipedia.org/wiki/%E6%B5%8B%E8%AF%95%E7%89%88%E6%9C%AC)。1.0 版本於 2006 年發布。

OpenCV 的第二個主要版本是 2009 年 10 月的 OpenCV 2.0。該版本的主要更新包括 [C++](http://zh.wikipedia.org/wiki/C%2B%2B)介面，更容易、更類型安全的模式，新的函數，以及對現有實現的優化(特別是多核心方面)。現在每 6 個月就會有一個官方版本[[1]](http://zh.wikipedia.org/wiki/OpenCV" \l "cite_note-1)， 並由一個商業公司贊助的獨立小組進行開發。

在 2012 年 8 月，OpenCV 的支持由一個非盈利性組織(OpenCV.org)來提供， 並保留了一個開發者網站 和用戶網站。

OpenCV 用 [C++](http://zh.wikipedia.org/wiki/C%2B%2B)語言編寫，它的主要介面也是 C++語言，但是依然保留了大量的 [C 語言](http://zh.wikipedia.org/wiki/C%E8%AF%AD%E8%A8%80)介面。該庫也有大量的

[Python,](http://zh.wikipedia.org/wiki/Python_(programming_language)) [Java](http://zh.wikipedia.org/wiki/Java_(programming_language)) and [MATLAB](http://zh.wikipedia.org/wiki/MATLAB)/[OCTAVE](http://zh.wikipedia.org/wiki/GNU_Octave) (版本 2.5)的介面。這些語言的 API 介面函數可以通過在線文檔獲得。[[4]](http://zh.wikipedia.org/wiki/OpenCV" \l "cite_note-Cdocs-4)現在也提供對於 [C#](http://zh.wikipedia.org/wiki/C_Sharp_(programming_language)), [Ch,](http://zh.wikipedia.org/w/index.php?title=Ch_(computer_programming)&amp;action=edit&amp;redlink=1)[[5]](http://zh.wikipedia.org/wiki/OpenCV#cite_note-5) [Ruby](http://zh.wikipedia.org/w/index.php?title=Ruby_(programming_language)&amp;action=edit&amp;redlink=1) 的支持。

所有新的開發和演算法都是用 C++介面。一個使用 [CUDA](http://zh.wikipedia.org/wiki/CUDA) 的 [GPU](http://zh.wikipedia.org/w/index.php?title=Graphics_processing_unit&amp;action=edit&amp;redlink=1) 介面也於 2010 年 9 月開始實現

**第三章 軟體需求分析**

我們以嵌入式系統Raspberry Pi和Open Soure的影像library，OpenCV去做車牌的辨識系統，而Raspberry Pi是linux的作業系統，我們只要從OpenCV下載下來之後，裝進去Raspberry Pi，即可使用。

**3.1 軟體設計**

表1、Raspberry Pi硬體規格

|  |  |  |  |
| --- | --- | --- | --- |
|  | Raspberry pi B | Raspberry Pi3 B |  |
| SoC | Broadcom BCM2835 | Broadcom BCM2837 |  |
| CPU | 700 MHz;ARM1176JZF-S core | 1.2 GHz 64-bit quad-core ARM Cortex-A53 |  |
| 記憶體 | 512MB SDRAM | 1GB LPDDR2（和 GPU 共享） |  |
| 工作電流 | 700 mA | 800 mA |  |
| 儲存 | SD / MMC / SDIO card slot | microSD |  |
| 視訊輸出 | Composite RCA; HDMI | Composite RCA; HDMI |  |
| 作業系統 | Raspbian,OpenLEC, Fedora, Raspbmc | Raspbian,OpenLEC, Fedora, Raspbmc,Windows10 |  |



圖1、靜態流程圖



圖2、車牌定位流程圖



圖3、字元擷取流程圖



圖4、字元辨識流程圖



圖5、動態車牌辨識流程圖

**第四章 測試功能**

車牌定位，主要是將一張車輛影像，找到車牌的所在位置，雖然每張車輛不可能都是相同距離或是相同角度，但是車牌的大小面積或長寬比是差不多相同的，所以用連通區域法找出最大連通區域，找出最大面積。本論文之車牌定位步驟：

Step1:將圖像灰階化，再透過5\*5高斯濾波器去掉雜訊，最後用Sobel   
 做邊緣偵測。

Step2:將圖像做適應式二值化以及3\*1矩形形態學。

Step3:最後將形態學後的圖像用OpenCV的輪廓方式，找出車牌的位置。而如果沒找到車牌，將查看二值化是否已經達到預先設定的範圍，如果沒有就將範圍加大，到達設定範圍還未找到車牌，則判定未找到車牌。

字元擷取之主要目的，為基於前一個步驟之車牌位置的物件資訊，在原始灰階影像中之相對位置，進行字元擷取的處理。

大部分比較普遍的方法都是用投影法來，將車牌作水平投影，投影出字元的相對位置，然後再對字元進行垂直水平切割。為了讓字元比較凸顯且減少環境光線的干擾，所以我用的不是一般固定的二值化，而是用適應式二值化且用反向二值化來處理，因為用反向會使黑色的部分直接變白，字元也會更明顯可見，然後我則是用OpenCV裡的輪廓找尋，因為車牌的字元面積相似不遠，所以我們設些判斷條件，將面積太小或過大的濾除掉，接下來就剩面積相似不遠的輪廓，剩下面積與字元相似的，再用長寬比例來去除。

車牌辨識，在辨識之前，我先用一個分類器，將數字0~9和A~Z分類，而我用SVM分類器和HOG特徵，做一個XML的訓練庫。而訓練的樣本是用機車的車牌的字樣所訓練而成。

辨識的HOG特徵是近年被廣泛使用的一種特徵，其基本概念為將像素的梯度。Dalad and Triggs 所提出的以HOG為特徵的行人偵測系統的特徵架構是以三種不同大小的區域為單位，分別是cell、block、window， cell是一個以固定大小、數量的像素所組成的區域，cells之間沒有重疊(overlap)；block 是一個以固定大小、數量的cells所組成的區域，blocks之間允許重疊；window是一個以固定大小、數量的blocks所組成的區域，也就是俗稱的ROI，windows之間允許重疊。最後辨識的特徵向量是由一整個window的特徵所組成。

最後將擷取出的字元，透過剛剛利用SVM與HOG所訓練的資料庫，一個一個字元辨識，然後再將車牌的字元，利用字串合併，將其合併為一起，統一輸出於終端機。

**4.1 辨識畫面**



圖6、Sobel邊緣偵測



圖7、形態學之圖像



圖8、紅色部分為車牌定位到的結果



圖9、紅色部分為車牌擷取到的結果

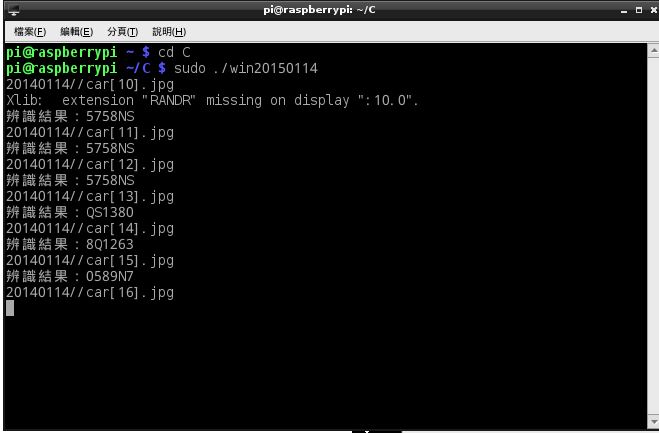


圖10、輸出結果

**第五章 總結**

慢慢隨著經濟發展越來越富裕，生活素質提高，使得有車的人越來越多，相對的在交通上衍生出更多的問題，尤其在停車方面，很多地方都需要有管理員來管理停車場，所以此專題的設計研發出以電腦代替人類判讀車牌進而衍生出可自動化管理的車牌辨識系統。

在現在這個發展迅速的時代，台灣平均每戶人家都有一台汽車，我們希望利用現今科技的方便性，讓車輛管理能夠資訊系統化

**創作過程的問題大致分於以下幾類**

1. 環境光線問題

因為每個地方，光線會有所不同，而在影像處理中，光線會使的影像變化是最大的，不管是太亮或太暗，所處理的方式，都是不一樣，所以只能用一些機制，讓它動態去找尋。

1. 雜訊問題

因為我是以找字元邊緣的方式去擷取字元，所以只要汙點太過於多，就很容易多擷取到汙點部分。

**參考文獻**

[1] "車牌辨識系統"，逢甲大學，資 訊 工 程 學 系，九 十 九 年。

[2][陳一昌](http://search.books.com.tw/exep/prod_search.php?key=%B3%AF%A4%40%A9%F7&amp;f=author)，"車牌影像辨識系統與號牌設計改進配合措施之探討"，2004 年

03 月 01 日。

[3]林泰良，"智慧型車牌定位與字串分割"，國立臺灣大學電機工程學研究所碩士論文，2000。

[4]洪健詠，"車牌辨識系統"， 立德管理學院資訊工程學系專題研究報告，

2004。

[5]維基百科，OpenCV，<http://zh.wikipedia.org/wiki/OpenCV>。

[5] "車牌辨識系統"，國立彰化師範大學電子工程系。