

碁峯資訊

版權聲明：本教學投影片僅供教師授課講解使用，投影片內之圖片、文字及其相關內容，未經著作權人許可，不得以任何形式或方法轉載使用。

```
import pygame, random, math, time

class Ball(pygame.sprite.Sprite):
    dx = 0
    dy = 0
    x = 0
    y = 0
    direction = 0
    speed = 0

    def __init__(self, sp, srx, sry, radium, color):
        pygame.sprite.Sprite.__init__(self)
        self.speed = sp
        self.x = srx
        self.y = sry
        self.image = pygame.Surface([radium*2, radium*2])
        self.image.fill((255,255,255))
        pygame.draw.circle(self.image, color, (radium,radium), radium, 0)
        self.rect = self.image.get_rect()
        self.rect.center = (srx,sry)
        self.direction = random.randint(40,70)

    def update(self):
        radian = math.radians(self.direction)
        self.dx = self.speed * math.cos(radian)
        self.dy = -self.speed * math.sin(radian)
        self.x += self.dx
        self.y += self.dy
        self.rect.x = self.x
        self.rect.y = self.y
        if(self.rect.left <= 0):
            self.rect.left = 0
        elif(self.rect.top <= 0):
            self.rect.top = 0
        elif(self.rect.bottom >= screen.get_height()-10):
            self.rect.bottom = screen.get_height()-10
        else:
            return False

    def bounceup(self):
        self.direction = (180 - self.direction) * 360

    def bouncelr(self):
        self.direction = (180 - self.direction) * 360

class B(pygame.sprite.Sprite):
    def __init__(self):
        self.rect = self.image.get_rect()
        self.rect.x = x
        self.rect.y = y

class Pad(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load("media\\pad.png")
        self.image.convert()
        self.rect = self.image.get_rect()
        self.rect.x = int((screen.get_width() - self.rect.width)/2)
        self.rect.y = screen.get_height() - self.rect.height - 20

    def update(self):
        pos = pygame.mouse.get_pos()
        self.rect.x = pos[0]
        if self.rect.x > screen.get_width() - self.rect.width:
            self.rect.x = screen.get_width() - self.rect.width

def gameOver(message):
    global running
    text = font1.render(message, 1, (255,0,255))
    screen.blit(text, (screen.get_width()/2-100,screen.get_height()/2-20))
    pygame.display.update()
    time.sleep(3)
    running = False

pygame.init()
font = pygame.font.SysFont('SimHei', 20)
font1 = pygame.font.SysFont('SimHei', 32)
soundhit = pygame.mixer.Sound("media\\hit.wav")
soundpad = pygame.mixer.Sound("media\\pad.wav")
```

Python

Machine Learning and Deep Learning with Python

機器學習與深度學習特訓班

看得懂也會做的AI人工智慧實戰

12

無所遁形術： 即時車牌影像辨識

12-1 專題方向

12-2 車牌號碼機器學習訓練資料

12-3 建立車牌辨識系統

12.1 專題檢視

首先將車牌圖片的檔案名稱更改為車牌號碼，如此在擷取車牌號碼文字後，可使用檔案名稱文字做為車牌號碼文字的標記。

接著撰寫程式 (`cropPlate.py`) 擷取車牌號碼圖形，車牌號碼圖形檔仍使用車牌號碼做為檔案名稱。

使用 `opencv` 的輪廓偵測功能可取得車牌號碼中文字輪廓的位置，再分別將文字擷取出來。車牌號碼文字為大寫英文字母及數字，新式車牌英文字母沒有「O」及「I」，因此有 24 個字母及 10 個數字共計 34 個文字，也就是機器學習時分為 34 類。分別以這 34 個文字建立資料夾，將擷取的车牌號碼文字圖形存入對應的資料夾中。

目前的資料量太少，無法進行機器學習訓練，所以要撰寫程式 (`makedata.py`) 增加資料數量：複製原始圖片，然後在圖片上隨機加入一些雜點，就能產生不同圖片。

本專題將各分類圖片擴增到 500 筆左右，全體資料數量有一萬七千多筆。資料準備齊全後就進行機器學習訓練建立模型，然後就可用模型來辨識未知車牌的號碼了！

12.2 車牌號碼機器學習訓練資料

12.2.1 原始圖片轉換尺寸

請複製書附光碟本章範例到硬碟中進行後續操作：`<realPlate_sr>` 資料夾含有 61張數位相機拍攝的相片，用於建立機器學習訓練資料；

`<predictPlate_sr>` 資料夾含有 5 張數位相機拍攝的相片，是讓機器學習模型實際測試的預測車牌。為了方便後續判斷車牌號碼辨識是否正確，所有圖片的檔案名稱已更改為車牌號碼，如此只要看檔案名稱即可得知該圖片的車牌號碼。

◆ 首先 `resize.py` 將所有數位相機拍攝的相片尺寸轉換為 300x225 像素圖形，以便讓 `<haar_carplate.xml>` 模型偵測。

12.2.2 擷取車牌號碼圖形

- ◆ **cropPlate.py** 利用前一章建立的車牌號碼 Haar 特徵分類器模型 (haar_carplate.xml)，就可自動框選出車牌號碼，進而將車牌號碼圖形擷取下來。

12.2.3 以輪廓偵測分割車牌號碼文字

若要分別擷取車牌的文字，可使用 `opencv` 的輪廓偵測功能取得車牌號碼中文字輪廓的位置，再分別擷取文字。

`opencv` 使用 `findContours` 方法取得圖片的輪廓，語法為：

```
尋找變數 = cv2.findContours( 圖片 , 偵測模式 , 輪廓算法 )
```

`findContours` 方法的傳回值是含有 3 個元素的串列，輪廓資訊存於第 2 個元素。例如偵測 `<test.jpg>` 圖片的輪廓存於 `contours` 變數中：

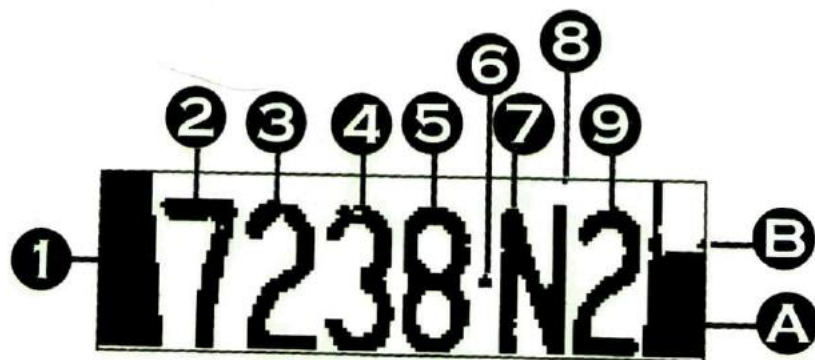
```
contours1 = cv2.findContours('test.jpg', cv2.RETR_EXTERNAL,  
                             cv2.CHAIN_APPROX_SIMPLE)  
contours = contours1[1]  # 第二個元素
```

`contours` 也是一個串列，每一個元素就是一個圖形輪廓。

一個圖形的輪廓資訊非常複雜，`opencv` 提供 `boundingRect` 方法，將圖形輪廓資訊以一個最小矩形包圍起來，我們只要擷取該矩形，就擷取了該輪廓的圖形。

使用 `boundingRect` 的語法為：

```
cv2.boundingRect( 輪廓資訊 )
```



[(0, 0, 15, 40) 1 (14, 6, 18, 34) 2 (34, 6, 17, 34) 3
 (53, 7, 16, 33) 4 (71, 7, 15, 33) 5 (88, 22, 3, 3) 6
 (93, 6, 15, 34) 7 (107, 0, 1, 1) 8 (110, 6, 15, 34) 9
 (126, 0, 14, 40) A (138, 13, 2, 2) B

批次擷取車牌號碼文字圖形

- ◆ **cropNum.py**可擷取單一車牌的每一個號碼圖片(利用前面所介紹的擷取原理)。
- ◆ **cropNum_all.py**可將<cropPlate>資料夾中所有車牌號碼檔案的每一個號碼圖片都擷取出來，存放於<cropNum>資料夾。為了方便核對擷取結果是否正確，將以車牌號碼建立資料夾，再將擷取的每一個號碼圖片檔案儲存於資料夾中。

12.2.4 建立類神經網路學習分類文字庫

辨識車牌號碼機器學習也是分類式機器學習：拆解車牌號碼文字，再辨識個別文字。

車牌號碼文字為大寫英文字母及數字，新式車牌英文字母沒有「O」及「I」，因此有 24 個字母及 10 個數字共計 34 個文字，也就是類神經網路學習時分為 34 類。

另外，新式車牌與舊式車牌的字型不同，蒐集車牌時，最好能包含新式車牌與舊式車牌的所有文字。(舊式車牌英文字母仍有「O」及「I」，但加入後反而對於「0(零)」與「1(壹)」的辨識造成更大錯誤，因此予以排除。)書附光碟的車牌資料已包含新舊車牌所有車牌號碼文字。

◆ **makefont.py** 程式會先在 **<platefont>** 資料夾內以大寫英文字母及數字為名稱建立 34 個資料夾，再將所有車牌擷取的文字檔案分別存入對應的資料夾中。

12.2.5 擴充機器學習分類文字庫

觀察擷取的車牌號碼文字圖片，可發現車牌號碼文字圖片有一些雜點，可能是原始車牌就有的髒污，也可能是處理圖片轉換產生的。我們可以複製原始圖片，然後在圖片上隨機加入一些雜點，這樣就產生了不同圖片。

目前各分類圖片的數量差異很大，擴充資料時將各分類圖片數量調整到差不多。

- ◆ **makedata.py** 程式將各分類圖片擴增到 500 筆左右，全體資料數量有一萬七千多筆存放於 **<data>** 資料夾，如果使用者要修改資料數量，只要修改 21 列程式即可。

12.3 建立車牌辨識系統

12.3.1 建立車牌號碼辨識模型

到此，車牌號碼辨識的一萬七千多筆全部位於 **<data>** 資料夾，訓練資料共分為**34個**號碼或英文字母，分別存放在**34個**資料夾當中。

訓練前，讀取訓練資料車牌個別號碼或圖片**(140x40x1)**時，就以資料夾名稱做為該圖片的標記(label)。

◆ 請用「pip list」檢查是否已安裝「imutils」模組，若未安裝請執行下列命令安裝：

```
pip install imutils
```

◆ **train.py**程式訓練神經網路模型辨識車牌上的個別號碼或英文字母(圖片)，完成後，將神經網路模型存檔。

12.3.2 使用車牌號碼模型

模型建立完成後，就可使用模型來預測車牌號碼了！

◆ **predict.py**程式以 12.1.3 節擷取「9238N2」車牌號碼文字圖形產生的 **<cropMono>** 資料夾為辨識對象，查看是否可以正確辨識該車牌擷取下來的個別號碼或英文字母(圖片)。

12.3.3 車牌辨識系統

- ◆ **recogPlate.py** 程式結合擷取車牌圖形、分割車牌號碼圖形，再利用車牌號碼神經網路模型辨識車牌號碼，就完成車牌辨識系統了！
- ◆ **recogPlate.py** 程式僅使用 `<predictPlate>` 資料夾的一張車牌 (3M6605.jpg)，此車牌並未包含在訓練資料中，而是全新的車牌，做為預測用。

12.3.4 批次辨識車牌

- ◆ **recogPlate_all.py** 程式提供同時辨識多個車牌功能：將要辨識的車牌置於同一個資料夾中(`<predictPlate>` 資料有 5 張車牌圖片)，程式會一一辨識，並列出圖片檔案路徑(檔案名稱就是車牌號碼) 及辨識結果，讓使用者查看辨識是否正確。