程式設計（二）：期末個人專案說明書
數字影像辨識器
(Digit Image Recognizer)

江政欽
國立東華大學資工系

April 14, 2014

目錄

摘要

本專案期使修課同學藉由系統實作熟悉 C++ 物件導向之各種技巧，故希望同學能在作品中應用到以下各種技巧：

Remark

- 類別設計 (Class Design)
- 類別繼承 (Class Inheritance)
- 檔案輸出入
- 函數與運算子過載 (Function and Operator Overloading)
- 標準樣板 (Standard Template)

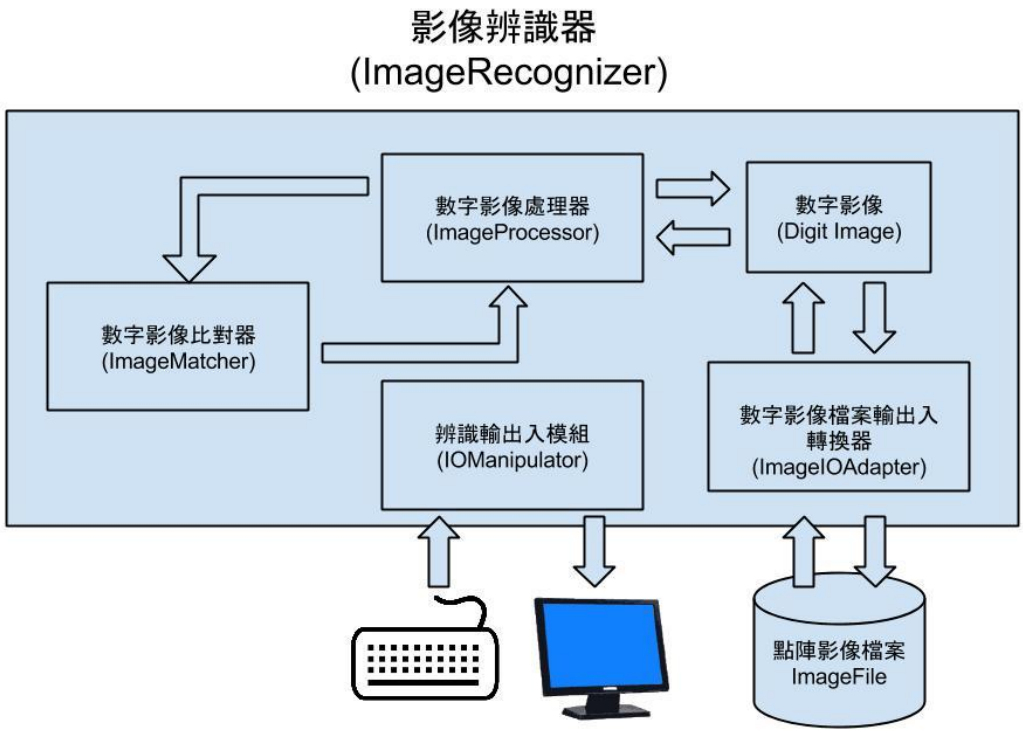
1 系統功能描述

本系統須含以下功能

系統功能

- 數字影像編輯：透過 Console 以文字編輯 $m \times n$ 數字點陣圖。
- 數字影像檔案輸出入：讀入或寫出數字影像點陣圖。
- 數字影像處理：數字影像大小縮放。
- 數字影像辨識：計算影像相似度，找出最相似影像。

Figure: 系統架構



2 系統模組規格

以下針對系統各模組的規格作一說明。

2.1 點陣影像資料結構

點陣圖 (Bitmap) 是以一個二維陣列來儲存影像像素 (Pixel) 的資料結構，在本專案中，點陣圖的像素可以是黑白影像或灰階影像，如果是白色像素，則像素值以 255 表示，如果是黑色像素，則像素值為 0，而介於 0 到 255 間的像素值則代表從暗到亮不

同的是度值。為能儲存一個影像，我們可以定義一個 Bitmap 類別，此 Bitmap 類別如下：

Code: Bitmap Class

```
#define MAX_HEIGHT 512 //點陣圖容許之最大高度
#define MAX_WIDTH 512 //點陣圖容許之最大寬是
class Bitmap {
public:
  int getWidth(); //取得影像寬度
  int getHeight(); //取得影像高度
  Bitmap(); //預設建構子
  Bitmap(int width, int height); //建構一個寬度，高度的點陣
      圖widthheight
  Bitmap(Bitmap &bmp); //拷貝建構子
  int& pixel(int y, int x); //取得像素(y,x)的Reference
private:
  int width; //影像寬度
  int height; //影像高度
  unsigned char[MAX_HEIGHT][MAX_WIDTH]; //二維像素值陣列
};
```

影像中像素均可以一個二維座標標定(y,x)，前者 y 為橫列，x 為縱行，且影像由上而下的橫列以 $y=0,1,\ldots$ 標定之，縱行由左而右則以 $x=0,1,\ldots$標定之。其中較值得注意的是 pixel(y,x) 這個函數回傳的是像素 (y,x) 的參考（Reference），而不是值（Value），依據課堂上老師教的觀念，Reference 可視同別名（Alias），所以在程式中我們既可以利用此函數來取得像素值，也可以利用它來設定像素值，使用方法如下例：

```
unsigned char value;
value = pixel(y,x); //取得像素(y,x)的值
pixel(y,x) = 255; //將像素(y,x)的值設成純白色
```

以上的成員資料設計僅供參考，各位同學可視需要予以調整設計。為方便後續的影像輸出入及影像處理與辨識，我們有必要在此 Bitmap 類別中設計相關之成員函數 (Member Functions) 和運算子（Operators），此部份的細節在後面各模組中會詳細規定。

2.2 點陣影像檔案輸出入模組

★ BitmapFileIO 類別

本模組主要用途是提供從檔案讀入一個點陣圖及將一個點陣圖寫入檔案中。基於物件導向原則，可將此點陣圖檔案輸出入模組設計為一個 BitmapFileIO 類別，其大致設計如下。

Code: BitmapFileIO Class

```
class BitmapFileIO {
public:
  void read(string filename);
          //filename: 檔名
          //進行讀檔，並將讀入之物件存入資料成員Bitmapbmp
  void write(string filename);
          //filename: 檔名
          //進行寫檔，將此類別之資料成員寫入檔案bmp
private:
  Bitmap bmp;    //儲存從檔案讀出或寫入檔案之影像
};
```

2.2.1 檔案輸出與輸入

灰階點陣圖的格式一般可以用 jpg 或 gif 檔儲存，但因為牽涉到壓縮和色彩量化等技術，非汝等大一生能力範圍之內（想一窺個中奧秘，歡迎日後修習本系聲色俱全、有趣好玩有錢途的「多媒體學程」），所以 CCC 選擇了最連高中生都懂的"pgm" 格式作為大家檔案輸出入的標準格式。pgm 格式其實是一個文字檔格式，它的檔案格式如下：

```
P2  ⇐ 這是必要的檔頭，標註此檔為pgm檔
# This is pgm format ⇐ #開頭的文字行均視為註解（可忽略之）
24 7 ⇐ 24是寬度，7是高度，單位均為像素
15 ⇐ 15代表最高亮度(白色像素)的像素值，黑色像素值一律為0
# 接著開始存放像素值
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15 15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

以上面的 pgm 內容為例，你可以是用可以觀看 pgm 檔的影像軟體（如 OpenSeeIt, http://openseeit.sourceforge.net/）打開來放大倍率來看，就會看到如下圖：

有了這樣的格式規範，相信聰明的你/妳一定很容易就能寫出可以讀出點陣圖的 pgm 讀檔程式吧！然後打鐵趁熱，再稍微多用那麼一點點時間，寫 pgm 檔的程式也立馬應聲出現的啦！但請記得 pgm 檔案的副檔名要取成".pgm" 喔！

為了方便將影像輸出入簡化，除了設計 BitmapFileIO 類別中的 read() 和 write() 外，我建議各位在學了 operator overloading(運算子過載) 後，練習針對 Bitmap 類別設計一個 Inseration operator(«) 和 Extraction operator (») 。有了這二個 operators 後，你的程式碼就可以寫成下面範例一樣：

Code: 點陣圖輸出入

```
ifstream in_file("infile.pgm");
ofstream out_file("outfile.pgm");
Bitmap bmp;
// do something
in_file >> bmp; //從讀入點陣圖到物件in_filebmp
out_file << bmp; //將物件寫出到bmpout_file
...
```

有了這二個運算子，其實你甚至可以不用設計 BitmapFileIO 這個類別。

2.3 點陣影像處理模組

為了方便辨識，影像讀入後我們必須對影像先作一些前處理的動作，其中最常包括的是：

★ 影像前處理

1. 影像黑白化（Binarization）：將灰階點陣圖轉換成非黑即白的黑白點陣圖
2. 大小正規化（Size Normalization）：將不同大小的點陣圖縮放到一個固定大小

完成前處理後，一種最直接的辨識方法就是將二個相同大小的點陣圖作對應像素相減後取絕對值，然後再將所有像素算出來的差異絕對值加總起來就可得到這二個點陣圖間的差異值，此差值越大就代表二點陣圖越不相似。計算公式如下

$$Diff(b_1, b_2) = \sum_{i=1}^{H} \sum_{j=1}^{W} |pixel_1(i,j) - pixel_2(i,j)| \tag{1}$$

其中 b_1, b_2 為二個被比對的點陣圖（寬度為 W，高度為 H），而 $pixel_1(i,j)$ 和 $pixel_2(i,j)$ 則分別為 b_1 和 b_2 上像素 (i,j) 的值。這種方法好處就是直接簡單，但缺點是當點陣圖變大時，像素變多了，所以相減的動作要做更多次，對速度會有負面影響。一種改善的方法是在二個影像上取出更精簡的特徵出來，然後以此特徵計算差異來作為比對依據。特徵通常會有很多數值，所以最好用一個一維陣列來儲存，在此，我們以一維 double 陣列來儲存。

在介紹前處理和特徵抽取的方法之前，你/妳必須將此模組設計為一個 ImageProc 類別，它大概的雛型如下：

Code: ImageProc Class

```
class ImageProc {
public:
  Bitmap& binarize(Bitmap bmp); //將bmp黑白化後傳回黑白點陣圖的結果
  Bitmap& normalize(Bitmap bmp, int new_w, int new_h);
  //將bmp縮放為寬度為new_w，高度為new_h的點陣圖
  double *extract(Bitmap bmp); //從bmp上抽取出一維特徵陣列
};
```

以下我們介紹如何完成這二個前處理動作。

2.3.1 點陣圖黑白化

點陣圖黑白化又叫做影像二值化，作法很簡單，只要設定一個像素值的門檻值，只要高過此門檻的像素就將其像素值改為 255(白色)，反之，若小於此門檻就將像素值改為 0（黑色）。要決定此門檻需要用到統計技巧，考量大家視數學如猛虎野獸，所以就不荼毒大家了，我們就以 $\frac{G_{max}}{2}$ 為門檻，其中 G_{max} 是點陣圖中最大的像素值。

點陣圖黑白化

$$pixel_{new}(y,x) = \begin{cases} 255, & \text{if } pixel_{old}(y,x) > \frac{G_{\max}}{2} \\ 0, & \text{otherwise.} \end{cases}$$

2.3.2 影像縮放

點陣圖的縮放其實很簡單，就是一個座標轉換而已，假設原點陣圖 B_1 大小為 $W_1 \times H_1$，我們想把它縮放為大小是 $W_2 \times H_2$ 的點陣圖 B_2。作法就是將 B_2 上每一個像素 $pixel_2(v,u)$ 的座標換算後，看 (v,u) 應該對應到哪個像素（假設 (y,x)），然後將 B_1 的 $pixel_1(y,x)$ 的像素值拷貝到 B_2 的 $pixel_2(v,u)$ 上即可完成。此座標換算公式應為：

點陣圖縮放

$$x = u\frac{W_1}{W_2} \tag{2}$$

$$y = v\frac{H_1}{H_2} \tag{3}$$

$$pixel_2(v,u) = pixel_1(y,x) \tag{4}$$

2.3.3 特徵抽取

可以用來辨識數字的特徵有上百種，每種都有優缺點，為了簡化大家實做的困難度，我們來使用一個叫做像素密度（Pixel Density）的特徵。這個特徵是把點陣圖平均分成四大塊：左上 (Upper-Left，代號 ul)、右上 (Upper-Right，代號 ur)、左下 (Lower-Left，代號 ll) 和右下 (Lower-Right，代號 lr)，然後計算出各塊的像素分佈比率，計算公式如下：

像素密度特徵

$$f_{ul} = N_{ul}/N \tag{5}$$

$$f_{ur} = N_{ur}/N \tag{6}$$

$$f_{ll} = N_{ll}/N \tag{7}$$

$$f_{lr} = N_{lr}/N \tag{8}$$

其中 N_{ul}, N_{ur}, N_{ll}, N_{lr} 分別為四塊區域內的白像素個數，而 N 則為此點陣圖內白像素總個數。

經過上述的特徵抽取後，我們得到四個密度值，因為都介於 0 到 1 間的小數，故可以一維之 double 陣列儲存之。這樣的特徵就不會因為點陣圖變大而讓特徵數變多，因

此有助於速度的提昇。如果同學們有創意，你/妳也可以設計自己的特徵，例如將區塊切割方法改變一下（如區塊切細或讓區塊間有部份重疊），或想出一個全新的特徵，也許你的辨識率就會因此而大幅提高，成績也就會扶搖直上喔！

2.4 點陣比對模組

一旦有了特徵之後，我們就可以設計辨識模組了，辨識模組的工作就是將輸入的待辨識點陣圖與數字 0 到 9 事先存好的標準樣板作差異度計算，然後從算出的差異度中挑出最小者就可得到辨識結果了。前文中，我們介紹了二種比對方法如下：

點陣圖比對方法

1. 點陣圖對應像素差異絕對值加總法：差異度計算如公式 (1) 所示，算出輸入點陣圖 (b) 與數字 0 到 9 的標準點陣圖樣版 $(b_0, b_1, \ldots, b_9)$ 的差異度後，若最小者為 $diff(b, b_m)$，則辨識結果就是數字 m。

2. 一維特徵向量差值法：假設輸入點陣圖的一維特徵陣列（或叫做特徵向量）為 $[f_1, f_2, \ldots, f_d]$，而數字 j 的標準特徵樣板為 $[t_1^j, t_2^j, \ldots, t_d^j]$，則二者間的向量差可依空間中二點座標的距離公式計算如下：

$$D_j = \|\mathrm{f} - \mathrm{t^j}\| = \sqrt{\sum_{k=1}^{d} (f_k - t_k^j)^2} \tag{9}$$

所以算出 D_0, D_1, ..., D_9 後，假設 D_m 是其中最小者，則辨識結果就是數字 m 囉！

基於上述設計，我們可以設計一個 Matcher 類別大致如下：

Code: Matcher Class

```
class Matcher {
public:
  double calcDiff(double *f1, double *f2); //計算特徵向量f1和特徵向量f2間的
      差異度
  double calcDiff(Bitmap b1, Bitmap b2);  //計算點陣圖b1和點陣圖b2間的差異
      度
  int recognize(Bitmap b, int method);
                        //針對輸入之點陣圖b，依據指定之方法(method)進行
                        //比對，傳回辨識結果之數字代號（0-9）
                        //method=1: 點陣圖樣版比對法，method=2:特徵向量比對法
};
```

2.5 辨識輸出模組

辨識結果出來了當然要把它顯示在螢幕上，這個部份 CCC 不像太限制大家的格式，但以下二個東西一定要秀在你的螢幕上

- 輸入的影像：如果你前面有過載 Bitmap 類別的 « 運算子，那你就可以直接使用 cout « bmp; 把點陣圖秀出來，如果沒有，那就請自行另外設計一個函數來做點陣圖的格式化輸出。這樣比較好檢驗你的辨識結果和輸入的點陣圖有否一致？

- 前五名候選字：比對通常不可能百分之一百正確，辨識錯誤發生時代表正確答案不是那個最小差異者，所以請依序輸出差異最小的前五名作為你的辨識結果，這樣方便讓使用者檢驗你的辨識結果是否很離譜？

3 進度規劃建議

本專案對各位同學其實不是難（沒用到太多的程式技巧，也沒有複雜的數學或演算法），但可能規模有一點點大，是個只要花時間就能完成的專案，為能讓同學適當調配時間來完成此專案，且避免嚴重排擠其他課業的時間，所以老師列出下表給大家參考，下表的時間規劃是以寬鬆為原則，只要依照時程按表施工，時程不會太緊，足供大家從容完成，若能順利完成，此專案將可為各位一年來的程式設計學習劃下完美句點，且以後可以拍胸脯大聲更人家說你的完成了一個一般資工系大一生不容易作到的實務系統，而且將來有興趣走多媒體領域中的影像處理、電腦視覺和圖像辨識等高科技的同學也可藉由這專案奠定不錯的專業基礎，以後拿這成果去推甄，一定很有加分效果。

Table 1: 進度規劃建議表

Module	Working Duration
Bitmap Class	0.5 week
BitmapFileIO Class	1 week
ImageProc Class	3 weeks
Matcher Class	2 weeks
BitmapOutput Class	1 weeks