資訊勘測 期末專題報告

人臉辨識

電機四 B97901031 向思蓉

B97901094 蔡知芸

一、專題目的

利用 data mining 的技術去分析與分類人臉 feature，建構人臉辨識系統，可辨識出不同的人，並且可以辨識出沒有在資料庫中的人臉，並加入資料庫中。

二、使用工具

1. datasets

這次 project 中的 datasets 是由我們自己建立的，內容為台大電機系二十多名同學的臉，每一個人包含 10~15 張人臉照片，每一張有不同的表情。

2. OpenCV

OpenCV 主要是用在影像部分，從擷取影像到處理影像，甚至是人臉辨識，都可以透過 OpenCV 提供的 library 實現，在這次的 project 中，我們主要用 OpenCV 實現兩部分：

a. 視訊影像

利用 CvCapture 函式，並加上 while 迴圈，便可以持續從 webcam 鏡頭擷取到每毫秒的視訊影像。再透過鍵盤事件，我們可以存取想要的圖片。

b. Haar cascade classifier

利用 20*20 大小的 haar-like features(OpenCV 有已經 train 好的 haar-like feature)，針對當前圖片去尋找是否有符合 haar-like features 的部分，如果找不到，就將圖片縮小 1.1 倍（可自行調整），直到小到比 threshold（可自行調整）小為止。另外為了提升 haar cascade 的準確度，通常會把兩到三個 classifier 串聯，以臉為例，找到臉後，判斷區域內有沒有眼睛、鼻子等，可以避免背景中類似人臉的東西被誤判。

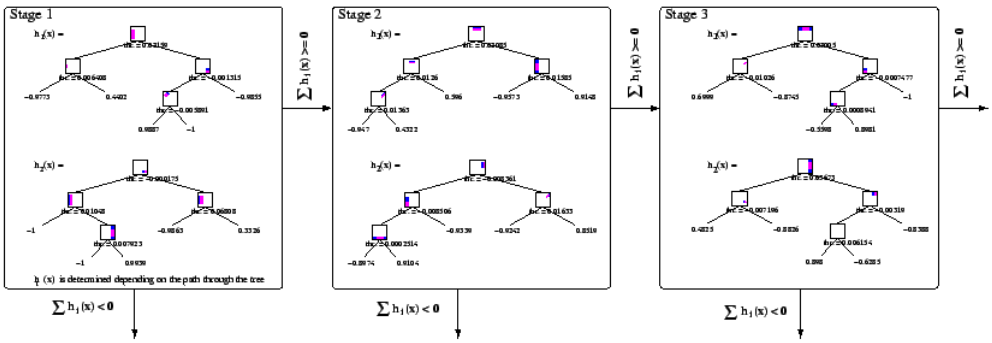
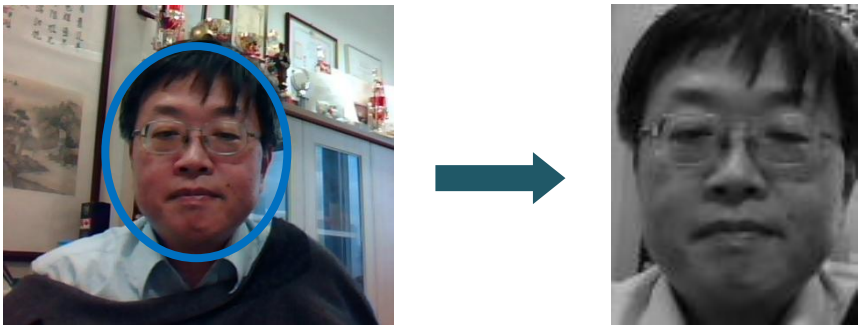


Figure: The first three stages of a cascade of classifiers to detect a ball. Every stage contains several simple classifier trees that use Haar-like features with a threshold and return values of $\sum h(x)$.

c. 影像處理

在上一步的 haar cascade 中取得人臉區域後，可以知道人臉的半徑與中心，利用

OpenCV 的影像處理函式，我們只擷取人臉部分，將每一張人臉都調整至 90*117 的大小，並且將圖片轉成灰階圖片。



3. Webcam

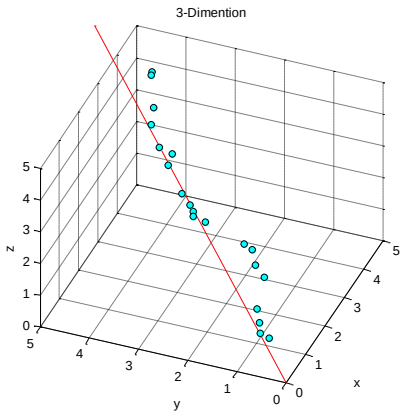
由於 OpenCV 在視訊影像的擷取上常受到硬體設施的限制，許多筆電內建的視訊鏡頭無法使用，故在這次的 project 中我們使用了 Logitech 的視訊鏡頭。

4. MATLAB

為達我們的分辨人臉圖像的目的，一些 supervised classifier（例如 support vector machine, SVM）會是很好的工具。然而一張圖太過大張（例如在我們的 project 中使用 90×117 像素的人臉圖片，相當於 10530 維的向量），對於 classifier 來說不但過於吃力，裡面所包含的資訊也太雜亂，分類的效果並不好。因此在訓練 classifier 之前，有必要將資料做適當的降維處理。在我們的 project 中，我們參考數篇論文並且使用 MATLAB 實際運用 PCA 及 LDA 的概念，以下將做詳細說明：

a. PCA(Principal Component analysis)

我們希望在高維度的空間中找到一些投影向量，將每一筆資料投影上去，用這些投影係數取代原本的資料，如果投影向量的數量較原資料的維度



小，而且這些投影係數不會使原資料間的關係和特性消失，那麼就達到我們所希望降維的目的。

如右圖，紅色線是一條很好的投影向量，當藍點都投影在紅線上時可以被拉到最開，也就是投影點之間的變異數 variance 最大，原本三維的資料可以被降成僅一維取而代之。

假設有 n 筆 d 維的資料 $\mathbf{X_0} = \{\mathbf{x}_i\}_{i=1\sim n} \in \mathbb{R}^{d\times n}$，通常 n<d(在我們的 project 中，n= 人數 × 10，d=90 × 117=10530)，我們希望找到投影向量 $\boldsymbol{\omega}$ 使得投影後 variance 最大：

$$\underset{\boldsymbol{\omega}}{\operatorname{argmax}} \operatorname{Var}(\boldsymbol{\omega}^{\mathrm{T}}\mathbf{x})$$

為求極值，自然的方法就是求微分值為零。定義目標函數

$$\begin{aligned} J(\boldsymbol{\omega}) &= Var(\tilde{\mathbf{x}}) \\ &= Var(\boldsymbol{\omega}^{\mathrm{T}}\mathbf{x}) \\ &= E[(\boldsymbol{\omega}^{\mathrm{T}}\mathbf{x} - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\mu})^{2}] \\ &= E[(\boldsymbol{\omega}^{\mathrm{T}}\mathbf{x} - \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\mu})(\mathbf{x}^{\mathrm{T}}\boldsymbol{\omega} - \boldsymbol{\mu}^{\mathrm{T}}\boldsymbol{\omega})] \\ &= \boldsymbol{\omega}^{\mathrm{T}}E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}]\boldsymbol{\omega} \\ &= \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{\omega}, \end{aligned}$$

其中 $\mu = \mathrm{E}[\mathrm{x}]$，$\Sigma \in \mathbb{R}^{d\times d}$ 是 x 的 covariance matrix。

我們所求即為

$$\arg_{\boldsymbol{\omega}} \frac{\partial J(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = 0$$

如果我們預設$\boldsymbol{\omega}$是單位向量，即 $\|\boldsymbol{\omega}\| = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} = 1$，下一個推導會用上一點小 trick，若另定義一函數

$$L(\boldsymbol{\omega}, \lambda) = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{\omega} - \lambda(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega} - 1)$$

則$L(\boldsymbol{\omega}, \lambda) = \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{\omega} = J(\boldsymbol{\omega})$, 原本$\frac{\partial J(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = 0$ 的問題就變成

$$\frac{\partial L(\boldsymbol{\omega}, \lambda)}{\partial \boldsymbol{\omega}} = 2\boldsymbol{\Sigma}\boldsymbol{\omega} - 2\lambda\boldsymbol{\omega} = 0$$

$$\left(\begin{array}{l} \because \text{covariance matrix } \boldsymbol{\Sigma} = \boldsymbol{\Sigma}^{\mathrm{T}} \\ \therefore \dfrac{\partial \boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\Sigma}\boldsymbol{\omega}}{\partial \boldsymbol{\omega}} = (\boldsymbol{\Sigma} + \boldsymbol{\Sigma}^{\mathrm{T}})\boldsymbol{\omega} = 2\boldsymbol{\Sigma}\boldsymbol{\omega} \end{array} \right)$$

$$\boldsymbol{\Sigma}\boldsymbol{\omega} = \lambda\boldsymbol{\omega}$$

即是典型的特徵值/特徵向量 (eigenvalue/eigenvector) 問題，我們想求的投影向量 $\boldsymbol{\omega}$ 就是 covariance matrix $\boldsymbol{\Sigma}$ 的特徵向量。

然而，$\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ 在我們的 project 中仍然是一個非常龐大的矩陣 (d=10530)，難以在電腦上進行特徵值分析。因此我們還需要另外的方法才能加以實做。

實際運算時，

$$\begin{aligned}\boldsymbol{\Sigma} &= E[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}] \\ &= \frac{1}{\mathrm{n}}\mathbf{X}\mathbf{X}^{\mathrm{T}},\end{aligned}$$

$\mathbf{X} = \{\mathbf{x}_i - \mu(\mathbf{x}_i)\}_{i=1\sim n}$，即每一向量減去自己的平均。為了運算方便，訂 $\Sigma = \mathrm{X}\mathrm{X}^{\mathrm{T}}$（稍後將證明 $\frac{1}{\mathrm{n}}$ 並不重要），則

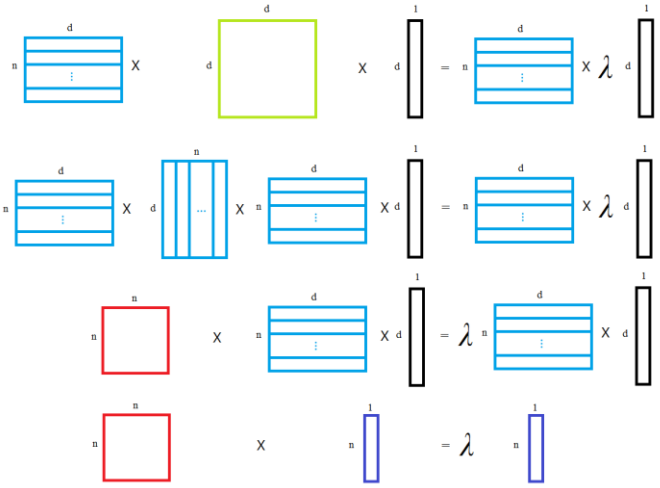
$\boldsymbol{\Sigma}\boldsymbol{\omega} = \lambda\boldsymbol{\omega}$

$\mathrm{X}\mathrm{X}^{\mathrm{T}}\boldsymbol{\omega} = \lambda\boldsymbol{\omega}$

(前面 $\times \mathbf{X}^{\mathrm{T}}$)

$\mathbf{X}^{\mathrm{T}}\mathbf{X}\mathbf{X}^{\mathrm{T}}\boldsymbol{\omega} = \lambda\mathbf{X}^{\mathrm{T}}\boldsymbol{\omega}$

$\mathbf{X}^{\mathrm{T}}\mathbf{X}\boldsymbol{\omega}' = \lambda\boldsymbol{\omega}'$



如此一來 $\omega' = \mathrm{X}^{\mathrm{T}}\omega \in \mathbb{R}^{\mathrm{n}}$也是特徵向量，不但和原本的 ω 有相同的特徵值，且 ω' 對應的矩陣是$\mathbf{X}^{\mathrm{T}}\mathbf{X} \in \mathbb{R}^{\mathrm{n}\times\mathrm{n}}$，維度較小。在我們的 project 中，n 大約為 200~300（視人數而不同），進行 300×300 矩陣的特徵值分析對於 MATLAB 而言是很輕鬆的工作。

求得 $\boldsymbol{\omega}' \in \mathbb{R}^{\mathrm{n}}$後，接下來要求 $\boldsymbol{\omega}$，由

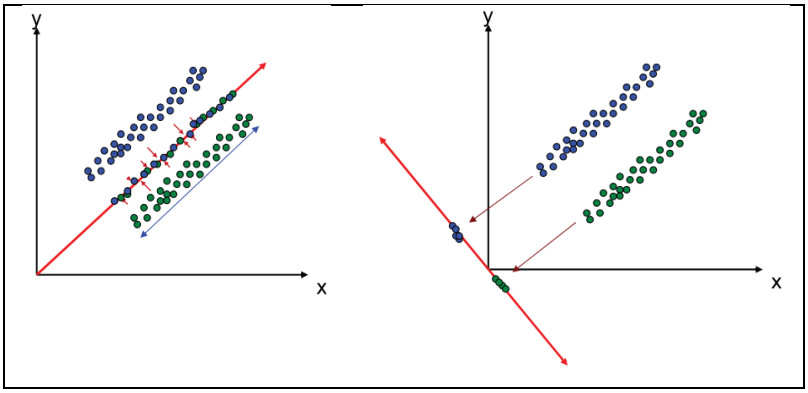
$$\begin{cases}\mathbf{X}\mathbf{X}^{\mathrm{T}}\boldsymbol{\omega} = \lambda\boldsymbol{\omega} \\ \boldsymbol{\omega}' = \mathbf{X}^{\mathrm{T}}\boldsymbol{\omega}\end{cases}$$
$$\Rightarrow \lambda\boldsymbol{\omega} = \mathbf{X}\boldsymbol{\omega}'$$

由於我們只需要單位化的 $\boldsymbol{\omega}$，計算 $\mathbf{X}\boldsymbol{\omega}'$ 並將其單位化即得之，上式的 λ 以及先前省略的係數 $\frac{1}{\mathrm{n}}$ 並不重要。

得到 n-1 個單位特徵向量 $\boldsymbol{\omega}$ 後，將資料 $\mathbf{X} = \{\mathbf{x}_i - \mu(\mathbf{x}_i)\}_{i=1\sim n} \in \mathbb{R}^{\mathrm{d}\times\mathrm{n}}$ 分別投影上去可得到較低維度的向量 $\mathrm{Y} = \{\mathbf{y}_i\}_{i=1\sim n} \in \mathbb{R}^{(\mathrm{n}-1)\times\mathrm{n}}$，且 $\{\mathbf{y}_i\}$ 之間的 variance 最大，保留了不同人臉圖像的差距資訊，使得 classifier 能更有效分類。因為分析過程中運用到特徵向量運算，所以$\{\mathbf{y}_i\}$被稱為 eigenface。

b. LDA(linear discriminate analysis)

雖然 PCA 已能有效降維，同時保留資料間特性，但是從上述的說明可以看出，PCA 對於所有類別的資料一視同仁。如下圖左，藍點與綠點分別代表兩組資料，從肉眼就可輕易看出分類，然而如果我們先使用 PCA 降維，將兩組資料投影在紅色向量上，雖然投影點之間的 variance 可以達到最大，但是屬於兩組的投影點反而混雜在一起，classifier 難以加以區別，因此 predict 效果不會最好。



在我們的 project 中，類別是很重要的資訊。由上述的說明可以看出我們希望的不是投影點間 variance 最大，最理想的投影結果是**同組內 variance 達到極小，而不同組間 variance 達到極大**。如此一來 classifier 的效果可以達到最好。如上圖右裡面的紅色向量就是理想的投影向量。

因此我們的目標函數應該修改為：

$$J(\boldsymbol{\omega}) = \frac{\text{variance between classes}}{\text{variance within classes}}$$
$$\boldsymbol{\omega} = \underset{\boldsymbol{\omega}}{\operatorname{argmax}} J(\boldsymbol{\omega})$$

首先探討分子部分，意即 c 組資料中，各組資料經過投影後的平均 $\tilde{\mu}_i = \boldsymbol{\omega}^{\mathbf{T}}\boldsymbol{\mu}_{\mathbf{i}}$($\boldsymbol{\mu}_{\mathbf{i}}$為第 i 組資料之平均)分得越開越好，即

$$\begin{aligned}
&\sum_{i=1}^{c} \|\tilde{\mu}_i - \tilde{\mu}\|^2 \\
&= \sum_{i=1}^{c} (\tilde{\mu}_i - \tilde{\mu})(\tilde{\mu}_i - \tilde{\mu})^{\mathrm{T}} \\
&= \sum_{i=1}^{c} (\boldsymbol{\omega}^{\mathbf{T}}\boldsymbol{\mu}_i - \boldsymbol{\omega}^{\mathbf{T}}\boldsymbol{\mu})(\boldsymbol{\omega}^{\mathbf{T}}\boldsymbol{\mu}_i - \boldsymbol{\omega}^{\mathbf{T}}\boldsymbol{\mu})^{\mathrm{T}} \\
&= \boldsymbol{\omega}^{\mathbf{T}} \left(\sum_{i=1}^{c} (\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^{\mathrm{T}} \right) \boldsymbol{\omega} \\
&= \boldsymbol{\omega}^{\mathrm{T}} \mathbf{S}_{\mathbf{B}} \boldsymbol{\omega}
\end{aligned}$$

再來分母部分，即

$$\begin{aligned}
&\sum_{i=1}^{c} \mathrm{Var}(\tilde{x}_i) \\
&= \sum_{i=1}^{c} \mathrm{Var}(\omega^T x_i) \\
&= \sum_{i=1}^{c} E[(\omega^T x_i - \omega^T \mu_i)(\omega^T x_i - \omega^T \mu_i)^T] \\
&= \omega^T \left(\sum_{i=1}^{c} \sum_{j=1}^{n_i} (x_{ij} - \mu_i)(x_{ij} - \mu_i)^T \right) \omega \\
&= \boldsymbol{\omega^T S_W \omega}
\end{aligned}$$

所以

$$J(\boldsymbol{\omega}) = \frac{\boldsymbol{\omega^T S_B \omega}}{\boldsymbol{\omega^T S_W \omega}}$$

$$\frac{\partial J(\boldsymbol{\omega})}{\partial \boldsymbol{\omega}} = \frac{2\boldsymbol{\omega^T S_W \omega S_B \omega} - 2\boldsymbol{\omega^T S_B \omega S_W \omega}}{(\boldsymbol{\omega^T S_W \omega})^2} = 0$$

$$\left(\because \mathbf{S_B}, \mathbf{S_W} \text{皆為對稱矩陣} \quad \therefore \frac{\partial \boldsymbol{\omega^T S_B \omega}}{\partial \boldsymbol{\omega}} = 2\mathbf{S_B \omega} \right)$$

$$\frac{\boldsymbol{\omega^T S_W \omega S_B \omega} - \boldsymbol{\omega^T S_B \omega S_W \omega}}{\boldsymbol{\omega^T S_W \omega}} = 0$$

$$\frac{\boldsymbol{\omega^T S_W \omega}}{\boldsymbol{\omega^T S_W \omega}} \mathbf{S_B \omega} - \frac{\boldsymbol{\omega^T S_B \omega}}{\boldsymbol{\omega^T S_W \omega}} \mathbf{S_W \omega} = 0$$

$$\mathbf{S_B \omega} - J(\boldsymbol{\omega}) \mathbf{S_W \omega} = 0$$

$$\mathbf{S_W^{-1} S_B \omega} = J(\boldsymbol{\omega}) \boldsymbol{\omega}$$

左邊 $\mathbf{S_W^{-1} S_B}$ 為一$\mathbb{R}^{d \times d}$的矩陣，而右邊$J(\boldsymbol{\omega})$為一純數，如同 PCA，我們的問題又簡化為單純的$\mathbb{R}^{d \times d}$矩陣特徵值分析。注意到不為零的特徵值數量就是 $\mathbf{S_W^{-1} S_B}$ 的階數，即 $\min\{\mathrm{rank}(\mathbf{S_B}), \mathrm{rank}(\mathbf{S_W})\} = \mathrm{rank}(\mathbf{S_B}) = c - 1$。因此在計算時我們可使用 MATLAB 函數 eigs 取前$c - 1$個特徵向量即可，增加系統效率。

c. FisherFace = PCA + LDA

然而 $\mathbf{S_W^{-1} S_B}$ 是 d×d 的龐大矩陣，在我們的 project 中 d=10530，對 MATLAB 來說無法進行特徵值運算。因此我們選擇將資料 $\mathbf{X} = \{\mathbf{x}_i - \mu(\mathbf{x}_i)\} \in \mathbb{R}^{d \times n}$先經過 PCA 處理成較低維的$\mathbf{Y} = \{\mathbf{y}_i\} \in \mathbb{R}^{(n-1) \times n}$，再進行後續的 LDA 運算，此時$\mathbf{S_W^{-1} S_B}$將會是僅僅(n-1)×(n-1)的矩陣，在我們的 project 中約為 299×299，求得特徵向量

後加以投影，就得到效率和效果兼具的更低維度向量$\mathbf{Z} = \{\mathbf{z}_i\} \in \mathbb{R}^{(c-1)\times n}$，適合 classifier 運作。

實做時有以下幾點值得注意：

i. 彩色圖片雖有三個維度，但因有用的資訊如臉的輪廓、五官深淺主要呈現在明暗變化，也就是 YIQ 的 Y 層，而 I、Q 的色彩資訊並不那麼重要，如下圖所呈現。因此我們先將人臉圖片由 RGB 轉為 YIQ，並只保留灰階層進行分析。

	RGB 表示 (資訊分散在三層)
	YIQ 表示 (資訊集中在 Y 層)

ii. 人臉拍攝時可能身處於不同亮度的環境，較亮環境造成較大的像素值，因此同一人在不同光線下將被分在不同的類別。然而我們分類的依據應該要是人臉亮暗變化，而非亮暗的絕對值，因此我們以減去平均值的 $\mathbf{X} = \{\mathbf{x}_i - \mu(\mathbf{x}_i)\}$ 取代原本的資料進行分析。

步驟	
I.	計算 $\mathbf{X} = \{\mathbf{x}_i - \mu(\mathbf{x}_i)\}$
II.	計算 $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ 的特徵向量 $\boldsymbol{\omega}'$
III.	計算 $\mathbf{X}\boldsymbol{\omega}'$ 並單位化，得 $\boldsymbol{\omega}_{\mathbf{PCA}}$
IV.	計算 $\boldsymbol{\omega}_{\mathbf{pca}}{}^{\mathrm{T}}\mathbf{X}$，得 $Y = \{\mathbf{y}_i\}$　若不需 PCA feature 則可省略此步驟
V.	計算 $\mathbf{S_B} = \sum_{i=1}^{c} \boldsymbol{\omega}_{\mathbf{PCA}}{}^{\mathrm{T}}(\boldsymbol{\mu}_i - \boldsymbol{\mu})(\boldsymbol{\mu}_i - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\omega}_{\mathbf{PCA}}$
VI.	計算 $\mathbf{S_W} = \sum_{i=1}^{c}\sum_{j=1}^{n_i} \boldsymbol{\omega}_{\mathbf{PCA}}{}^{\mathrm{T}}(\mathbf{x_{ij}} - \mu_i)(\mathbf{x_{ij}} - \mu_i)^{\mathrm{T}}\boldsymbol{\omega}_{\mathbf{PCA}}$
VII.	計算 $\mathbf{S_W}^{-1}\,\mathbf{S_B}$ 的特徵向量 $\boldsymbol{\omega}_{\mathbf{LDA}}$
IIX.	計算 $\boldsymbol{\omega}_{\mathbf{LDA}}^{\mathbf{T}}\boldsymbol{\omega}_{\mathbf{PCA}}{}^{\mathrm{T}}\mathbf{X}$，得 $\mathbf{Z} = \{\mathbf{z}_i\}$

5. libSVM (http://www.csie.ntu.edu.tw/~cjlin/libsvm/)

   由台大資工系教授林智仁開發，可以進行 linear 的 SVM，其優點在於快速，並且支援多項功能與開發平台，如 probability 估算、cross validation、不同種類之 kernel model 等等。

   在這次的 project 中，我們將低維度 LDA 向量 $\mathbf{Z} = \{\mathbf{z}_i\}$做為 SVM model training 的 feature 向量，使用 non-linear 的 radial basis function: exp(-gamma*|u-v|^2)，並且透過 cross validation 選擇最佳的參數 gamma、c。

   libSVM 支援 estimation probability，因此我們在 train model 時便記錄下正確分類的 estimation probability，取所有圖片中的最小值並乘以 0.7 做為 predict 的 threshold，如果預測結果的 estimation probability 沒有超過此 threshold，我們便視此張臉為陌生人。

   由上述可看出，我們的 model 是彈性可變化的，隨著 class 數量變多，threshold 會自動變小，並不需要手動調整。

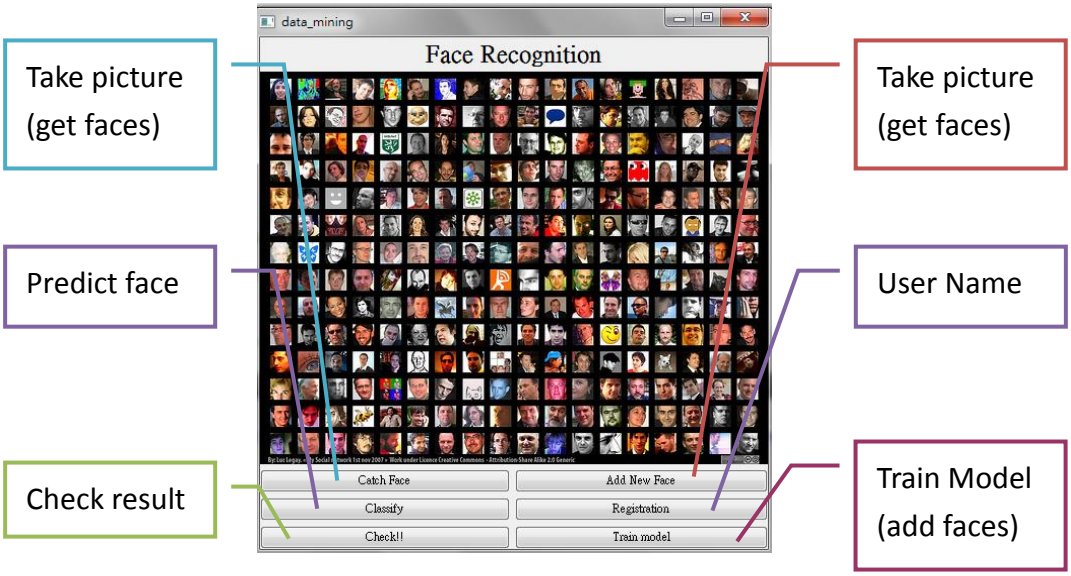
6. QT

   QT 是由 NOKIA 公司開發，不但擁有了完善的 C++圖形函式庫，而且近年來的版本逐漸整合了資料庫、OpenGL 函式庫、多媒體函式庫、網路、指令碼函式庫、XML 函式庫、WebKit 函式庫等等，其核心函式庫也加入了行程間通訊、多執行緒等模組，相同的程式碼可以在任何支援的平台上編譯與執行，而不需要修改原始碼。會自動依平台的不同，表現平台特有的圖形介面風格。

   在這次的 project 中，利用 QT 製作圖形化使用者界面。利用 QPushButton、QLabel、QSplitter 等圖形工具，以及 QProcess 等 library 使得按鈕事件可以執行 exe 檔與 batch 檔案。

三、專題架構

1. User Interface



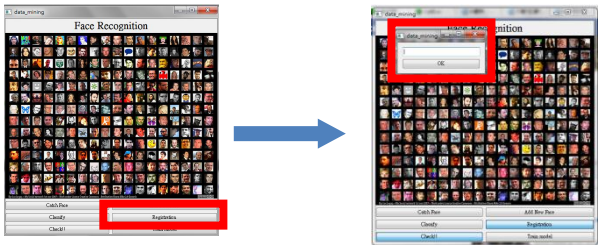
Step.1 點選”Check Face”，當視訊鏡頭顯示出藍圈標示出目標人臉後，按下”s”即可以儲存圖片，按下”q”則離開程式。程式關閉後，出現將來 predict 要用的圖片

Step.2 點選”Classify”，開始 predict

Step.3 點選”Check!!”，會以文字標籤的方式顯示 predict 的結果。

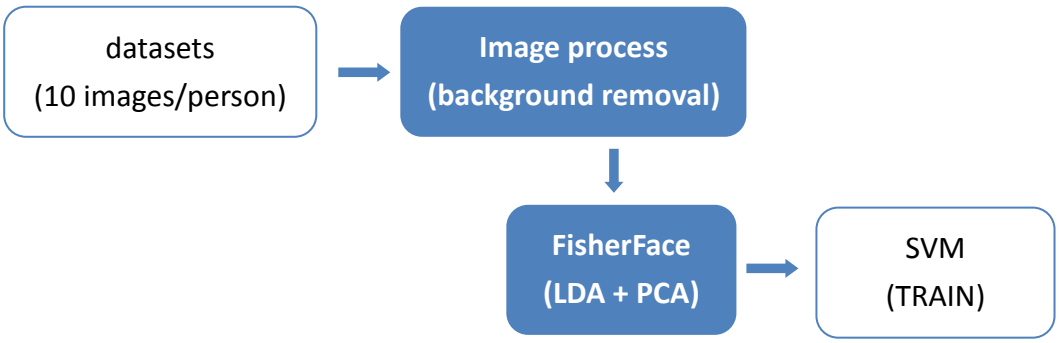
Step.4 如果目標預測為”陌生人”，則可以將此人加入人臉資料庫中，按下”Add New Face”，開啟視訊鏡頭，每按一次”a”便儲存一張圖片，大約需要 10~15 張相片。

Step.5 按下”Register”後，會出現輸入方框，輸入想要加入資料庫的人臉的姓名標籤，按下確定即可。

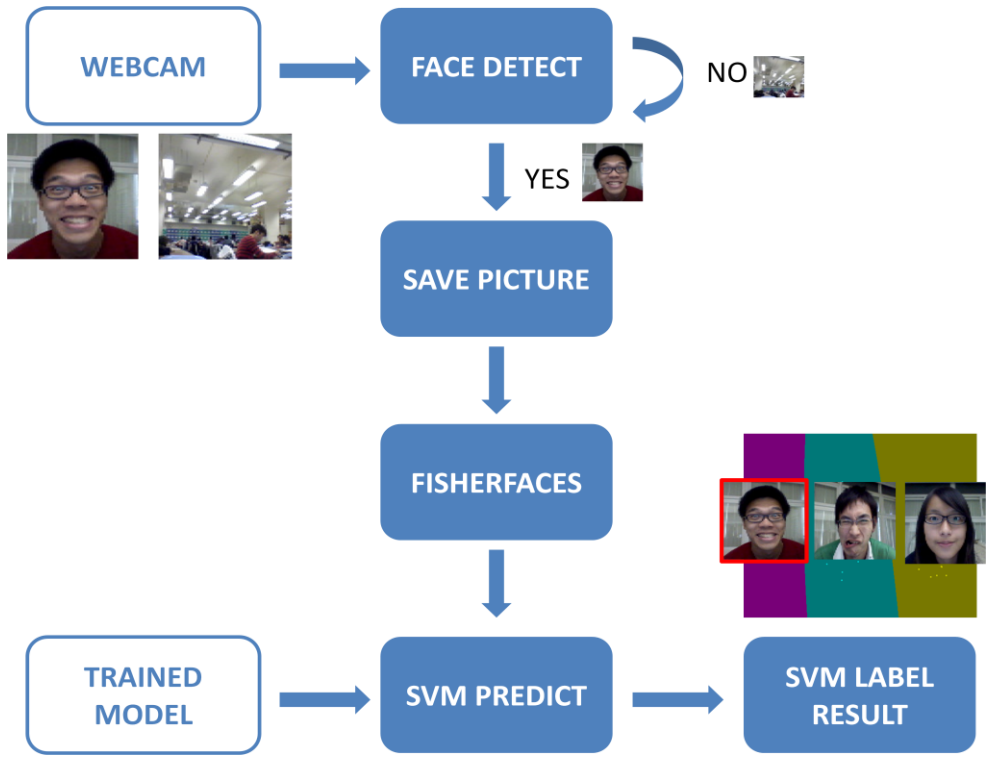


Step.6 最後，按下”Train Model”，便可以重新 train 一次人臉資料庫，產生出新的 SVM model。

2. Train model



3. Test model



四、專題成果

針對我們自己蒐集的 data set，我們做了以下實驗

* SVM kernel: radial basis exp(-gamma*|u-v|^2)

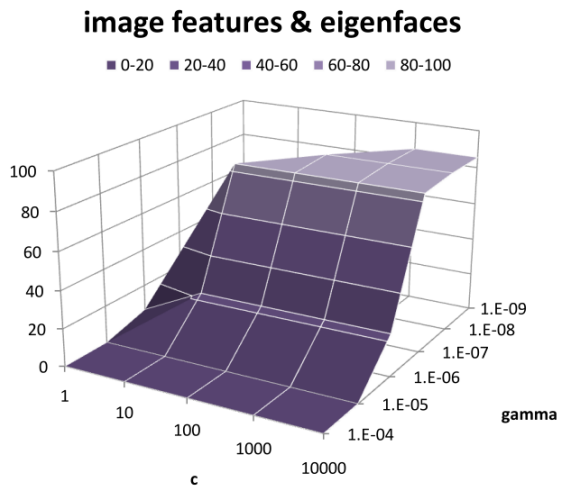
1. 比較原 image、eigenface(PCA)、fisherface(PCA+LDA)做為 feature 向量的辨識成功率(%)：

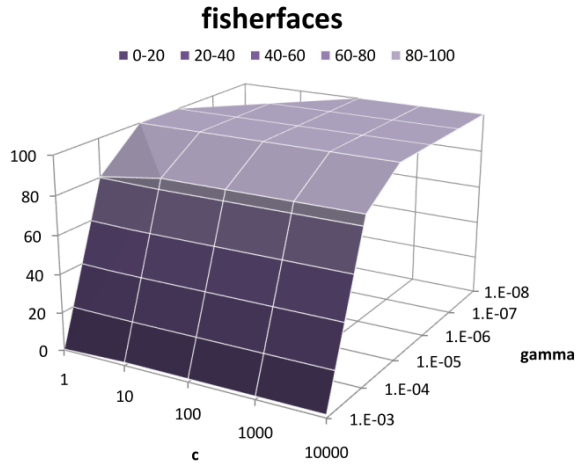
image、eigenface (same results)

c ＼ g	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
1	0	0	6.25	3.75	0	0
10	0	0	22.5	83.75	15	0
100	0	0	22.5	83.75	85.42	15.42
1000	0	0	22.5	83.75	85.42	85.42
10000	0	0	22.5	83.75	85.42	85.42

fisherface

c ＼ g	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}	10^{-8}
1	0.42	80.42	100	100	0	0
10	1.67	85	100	100	100	0
100	1.67	85	100	100	100	100
1000	1.67	85	100	100	100	100
10000	1.67	85	100	100	100	100





雖然在 training 及 testing 時 PCA 因維度較原圖低而有速度較快的優勢，但上面數據顯示用原圖或 PCA 的辨識效果都差不多，而經過 PCA+LDA 處理的 fisherface 除了維度最低，辨識效果也最好。

2. 比較總類別數量、單一類別資料量造成的影響：

辨識成功率(%)

gamma=10^{-6}，c=1

張/人 人數	3	5	8	10
5	100	100	100	100
10	97.14	100	100	100
15	89.29	100	100	100
20	85.71	93.33	94.44	100
25	82.86	92	93.33	100
30	76.19	91.11	92.59	94.44

當人數增加，每人所需要的 training data 也越多，才可提高辨識成功率。實際 DEMO 時，data set 維持在 25~30 人之間，因此我們取一個人 10 張圖片，作 PCA+LDA 得到低維的 fisherface 用以 train SVM model。在不同光線環境下，

- 辨識出已知面孔：≅85%
- 辨識出陌生面孔：≅90%

此外，當使用者在不遮蔽五官的前提下變換髮型(例如將長髮綁起)、或是戴/不戴眼鏡，仍然超過 70%的辨識成功率。

五、參考資料

1. Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19,NO.7, JULY 1997

2. Phillip Ian Wilson, and John Fernandez, FACIAL FEATURE DETECTION USING HAAR CLASSIFIERS, CCSC: South Central Conference, 2006

3. A guide for beginners: A practical guide to support vector classification C.-W. Hsu, C.-C. Chang, C.-J. Lin.

4. Chih-Chung Chang and Chih-Jen Lin, LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and technology, :27:1--27:27,

2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

5. OpenCV Document

(http://www-nh.scphys.kyoto-u.ac.jp/~hayata/opencv_doc/index.htm)