影像辨識期末報告

人臉辨識

學　　生：B10538008 王凱誼
B10538069 范勝凱
B10538080 李佩蓉

中華民國 109 年 1 月 7 日

1.Haar 特徵分類器

Haar 分類器

Haar 特徵是用於物體辨識的一種數位影像特徵，它反映了圖像局部的灰度變化。而 Haar 分類器是一項分類任務，訓練其分類物體是否存在來從而實現檢測，也是第一種即時的人臉檢測運算。因為它的速度快，即時性夠，所以我們利用此分類器做為路面箭頭的檢測。

如果直接使用圖像的強度(就是圖像每一個像素點的 RGB 值）去計算，會使得特徵的計算量很大。為了加快速度，我們利用 Haar 特徵，使用檢測窗口中指定位置的相鄰矩形，計算每一個矩形的像素和並取其差值。然後用這些差值來對圖像的子區域進行分類。

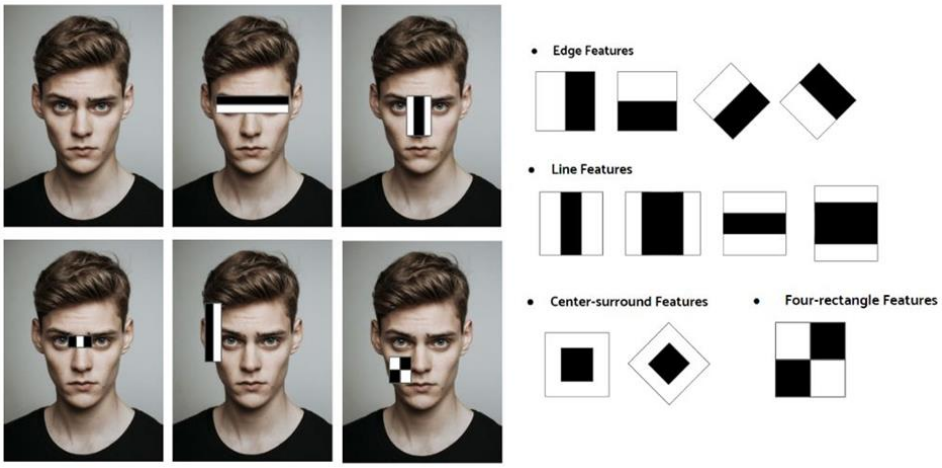


圖 1 特徵檢測及 Haar 矩形特徵圖

在目標檢測框架的檢測階段，一個與目標物體同樣尺寸的檢測窗口將在輸入圖像上滑動，在圖像的每一個子區域都計算一個 Haar 特徵。然後這個差值會與一個預先計算好的閾值進行比較，將目標和非目標區分開來(大於閥值為特徵)。因為這樣的一個 Haar 特徵是一個弱分類器（它的檢測正確率僅僅比隨機猜測強一點點），為了達到一個可信的判斷，就需要一大群這樣的特徵。在目標檢測框架中，就會將這些 Haar 特徵組合成一個分類器，最終形成一個強分類群(Adaboost 演算法)。

Haar 特徵最主要的優勢是它的計算非常快速。使用一個稱為積分圖的結構，任意尺寸的 Haar 特徵可以在常數時間內進行計算。如

下圖所示，左邊是原始圖像的像素值，右邊是積分圖像的像素值。從左上角開始計算給定矩形區域下像素的累加值。我們可以通過子矩形的值方便地得到某個區域的像素值總和。

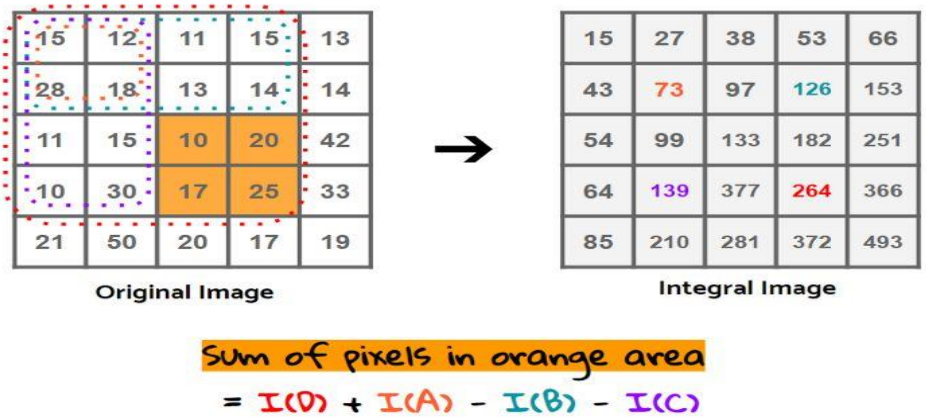


圖 2 積分圖像示意圖

AdaBoost 方法是一種疊代算法，在每一輪中加入一個新的弱分類器，直到達到某個預定的足夠小的錯誤率。每一個訓練樣本都被賦予一個權重，表明它被某個分類器選入訓練集的概率。如果某個樣本點已經被準確地分類，那麼在構造下一個訓練集中，它被選中的概率就被降低；相反，如果某個樣本點沒有被準確地分類，那麼它的權重就得到提高。通過這樣的方式，AdaBoost 方法能「聚焦於」那些較難分（更富信息）的樣本上。在具體實現上，最初令每個樣本的權重都相等，對於第 k 次疊代操作，我們就根據這些權重來選取樣本點，進而訓練分類器 Ck。然後就根據這個分類器，來提高被它分錯的的樣本的權重，並降低被正確分類的樣本權重。然後，權重更新過的樣本集被用於訓練下一個分類器 Ck。整個訓練過程如此疊代地進行下去。

基本特徵會在早期階段被識別出來，後期只識別有希望成為目標特徵的複雜特徵，也就是對 haar 特徵進行排序。在每一個階段，Adaboost 模型都將由集成弱分類器進行訓練。如果子部件或子窗口在前一階段被分類為“不像人臉的區域”，則將被拒絕進入下一步。通過上述操作，只須考慮上一階段篩選出來的特徵，從而實現更高的速度。其核心思想就是針對不同的訓練集訓練同一個弱分類器，然後把在不同訓練集上得到的弱分類器集合起來，構成一個最終的強分類器。

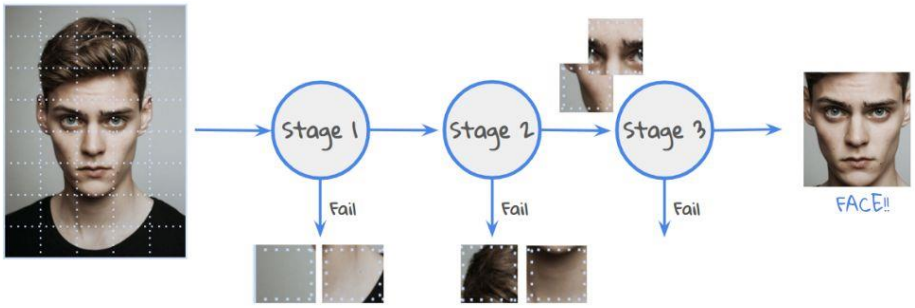


圖 3 分類器各階段示意圖

2.Eigenfaces

採用 PCA 的方法降維，是為了減少人臉圖像的表示，即將原始圖像投影到特徵空間，得到一系列降維圖像，取其主元素表示人臉，因其主元素有人臉的形狀，稱為“特徵臉”。

EigenFace 是一種基於統計特徵的方法，將人臉圖像視為隨機向量，並用統計方法辨別不同人臉特徵模式。

EigenFace 的基本思想是，從統計的觀點，尋找人臉圖像分佈的基本元素，即人臉圖像樣本集協方差矩陣的特徵向量，以此近似的表特徵人臉圖像，這些特徵向量稱為特徵臉。

下圖對特徵臉的應用進行了說明。從下圖可以看出，一組特徵臉基圖像（特徵臉 1~d）組成一個特徵臉子空間，任何一幅人臉圖像（減去平均人臉後）都可投影到該子空間，得到一個權值向量(§1~d)。

計算此向量和訓練集中每個人的權值向量之間的歐式距離，取最小距離所對應的人臉圖像的身份作為測試人臉圖像的身份。

而這裡所提到的一組特徵臉基圖像（也就是特徵臉，或者叫特徵向量），正是利用 PCA 所求得的協方差矩陣的特徵向量。具體可以參考主成分分析（PCA）和線性判別分析（LDA）原理簡介。

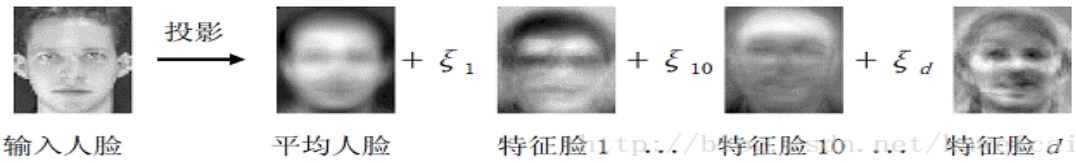


圖 4 分類器各階段示意圖

圖 5 Eigenfaces 流程圖

3.Fisherfaces

FisherFace 是一種基於 LDA(全稱 Linear Discriminant Analysis, 線性判別分析)的人臉識別算法，而 LDA 是 Ronald Fisher 於 193 年提出來的，所以 LDA 也被稱作是 Fisher Discriminant Analysis,也正因為如此，該人臉識別算法被稱為 FisherFace。

LDA 有和 PCA 相同的地方是，都有利用特徵值排序找到主元素的過程，但是不同的是 PCA 求的是協方差矩陣的特徵值，而 LDA 是求的是一個更為複雜的矩陣的特徵值。

其中需要注意的是在求均值時，和 PCA 也是有所不同的，LDA 對每個類別樣本求均值，而 PCA 是對所有樣本數據求均值，得到平均臉。

4.LBPH

LBPH 是利用局部二值模式直方圖的人臉識別算法。

LBP 是典型的二值特徵描述子，所以相比前面 EigenFace 和 FisherFace，更多的是整數計算，而整數計算的優勢是可以通過各種邏輯操作來進行優化，因此效率較高。

另外通常光照對圖中的物件帶來的影響是全局的，也就是說照片中的物體明暗程度，是往同一個方向改變的，可能是變亮或變暗，只是改變的幅度會因為距離光源的遠近而有所不同。

所以基本上局部相鄰(Local)的像素間，受光照影響後數值也許會改變，但相對大小不會改變，因此 LBP 特徵對光照影響不大。

圖 6 LBPH 流程圖

5.程式碼

訓練用:

```python
# -*- coding: utf-8 -*-
import cv2, os, sys
import numpy as np
from scipy.ndimage import filters

def read_images(path, sz=None):
    c = 0
    X, y = [], []

    for dirname, dirnames, filenames in os.walk(path):
        for subdirname in dirnames:
            subject_path = os.path.join(dirname, subdirname)
            for filename in os.listdir(subject_path):
                try:
                    if not filename.endswith('.jpg'):
                        continue
                    filepath = os.path.join(subject_path, filename)
                    im = cv2.imread(filepath, cv2.IMREAD_GRAYSCALE)
                    im = cv2.equalizeHist(im)
                    im = cv2.GaussianBlur(im,(3,3),1.5)
#                     im = filters.gaussian_filter(im, 10)
                    im = cv2.resize(im, (200, 200))
                    if sz is not None:
                        im = cv2.resize(im, (200, 200))
                    X.append(np.asarray(im, dtype=np.uint8))
                    y.append(c)
                except:
                    print("Unexpected error:", sys.exc_info()[0])
            c = c + 1
            print c
    return [X, y]
```

```python

def face_rec(img_path):
    a = range(0, 99)
    names = []
    names = a
    [X, y] = read_images(img_path)
    y = np.asarray(y, dtype=np.int32)
#     model = cv2.face.EigenFaceRecognizer_create()
#     model = cv2.face.FisherFaceRecognizer_create()
    model = cv2.face.LBPHFaceRecognizer_create()
    print "training now"
    model.train(np.asarray(X), np.asarray(y))
    print "training complete"
    print"save end"
    model.save("face_rec_LBPH.xml")
    print"save complete"

if __name__ == "__main__":
    face_rec('images/')

```

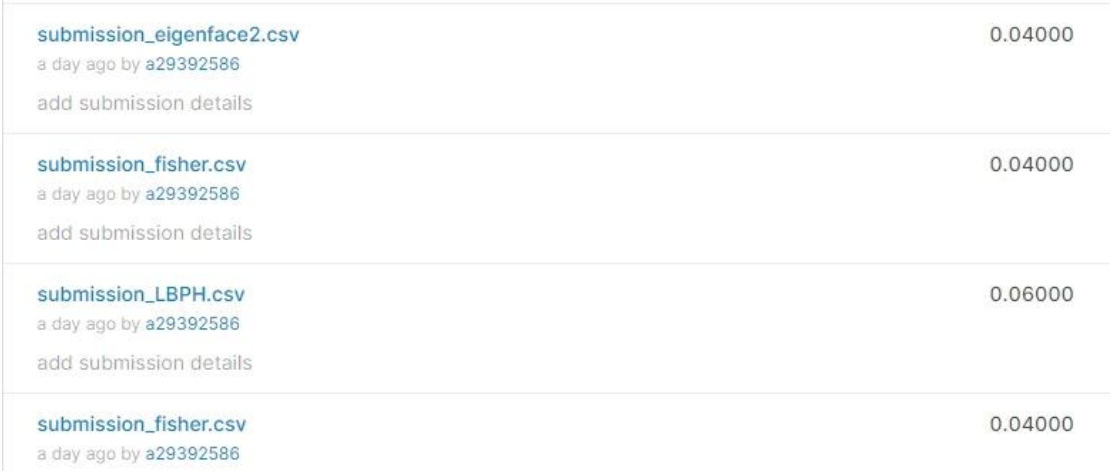
辨識用:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt
import os
import csv
import pandas as pd
# img = cv2.imread('C:\Users/nigge\Desktop\image recongnition\images/5498592/026.jpg')
detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# model = cv2.face.EigenFaceRecognizer_create()
# model = cv2.face.FisherFaceRecognizer_create()
model = cv2.face.LBPHFaceRecognizer_create()
print("roading xml now")
model.read("face_rec_LBPH.xml")
print("loading complete")
a = range(0, 99)
names = []
names = a
idList =[]
label = []
datas = []
name2= ['Id','Category']
test_images = os.listdir('images_test')
print('Found %d queries' % (len(test_images)))
```

```
for im in test_images:
    img = cv2.imread('images_test/'+im)

    signs = detector.detectMultiScale(img,
                                      scaleFactor=1.1,
                                      minNeighbors=5,
                                      minSize=(100, 100))
    if len(signs) > 0:
        for (x, y, w, h) in signs:
            cv2.rectangle(img,
                          (x, y), (x + w, y + h),
                          (0, 0, 255), 2)
            gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
            roi = gray[x: x + w, y: y + h]
            roi = cv2.equalizeHist(roi)
            roi = cv2.GaussianBlur(roi,(3,3),1.5)
#             roi = filters.gaussian_filter(roi, 10)
            roi = cv2.resize(roi, (200, 200), interpolation=cv2.INTER_LINEAR)
            params = model.predict(roi)

            try:
                print("Label: %s, Confidence: %.2f" % (params[0], params[1]))

                idList.append(im)
                label.append(params[0])
                datas.append(['images/'+im,params[0]])

            except:
                continue
                print ("nothing")
```

```
     else:
         print im
         print('nothing')
         idList.append(im)
         label.append('None')
         # datas.append([im, 'None'])
         datas.append(['images/'+im, 0])

test = pd.DataFrame(columns=name2,data=datas)
print (test)
test.to_csv('submission_LBPH.csv',encoding='gbk',index=False)


cv2.waitKey(0)
cv2.destroyAllWindows()
```

6.結果



7.遇到的問題

▶ Haar 分類器的 minNeighbors 參數調太小，同一張照片會很多非臉的部份被框到

→調高 minNeighbors 參數，盡量使一張照片只框出一個人臉

▶ 辨識成功率低

→影像預處理問題，雖有設定照片大小規一化，但在訓練集卻忘了只針對人臉做訓練，而是對整張圖訓練；所以在與測試圖片只取人臉部分做匹配時，造成辨識成功率低。

▶ 預處理效果不顯著

→再嘗試其他預處理方式。