

人臉辨識實戰

上課講師：莊啓宏

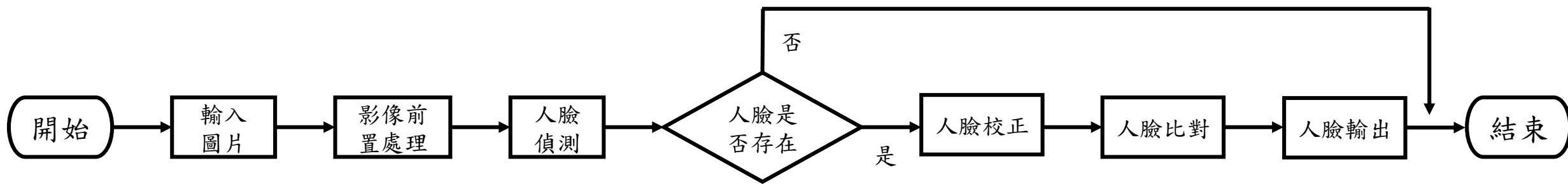
摘要

- 人臉辨識是基於人的臉部特徵資訊進行身份辨識的一種生物辨識技術，是用攝像機或攝影機擷取含有人臉的影像或視訊流，並自動在影像中檢測和追蹤人臉，進而對檢測到的人臉進行臉部辨識的一系列相關技術，通常也叫作人像辨識、面部辨識。

- 在研究人臉辨識過程中，經常看到FDDB和LFW這兩個縮寫簡稱，但很多人不知道它們到底指的是什麼？
- (1)FDDB的全稱為Face Detection Data Set and Benchmark，是由麻塞諸塞大學電腦系維護的一套公開資料庫，為來自全世界的研究者提供一個標準的人臉檢測評測平台。它是全世界最具權威的人臉檢測評測平台之一，包含2845幅圖片，共有5171個人臉作為測試集。測試集範圍包括：不同姿勢、不同解析度、旋轉和遮檔等圖片，同時包括灰階圖和彩色圖，標準的人臉標注區域為橢圓形。值得注意的是，目前FDDB所公佈的評測集也代表了目前人臉檢測的世界最高水準。

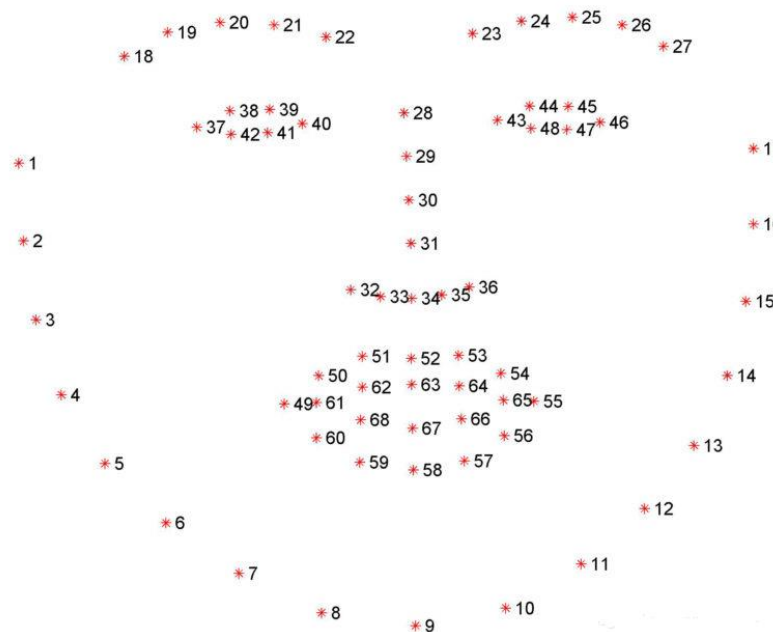
- (2) LFW全名Labeled Faces in the Wild，是由麻塞諸塞大學於2007年建立，用於評測非限制條件下的人臉辨識演算法性能，它也是人臉辨識領域使用最廣泛的評測集合。該資料集由13000幅全世界知名人士在自然場景中的具有不同朝向、表情和光源的人臉圖片組成，共有5000多人，其中有1680人有2幅或2幅以上人臉圖片。每幅人臉圖片都有其唯一的姓名ID和序號加以區分。LFW測試正確率代表了人臉辨識演算法在處理不同姿態、光線、角度、遮擋等情況下辨識人臉的綜合能力。

人臉辨識步驟



- 影像前置處理
 - 人臉光線補償
 - 灰階轉換
 - 濾波
- 人臉偵測
 - 檢測一張圖片中人臉是否存在

- 人臉校正
 - 又可以稱人臉矯正、人臉扶正等(但目前可以以巨量資料樣本訓練取勝)
- 人臉特徵定位
 - 人臉特徵點定位是指在檢測到圖片中人臉的位置之後，在圖片中定位能夠代表圖片中人臉的關鍵位置的點。
 - 常用的人臉特徵點是由左右眼、左右嘴角、鼻子這5個點組成的5點人臉特徵點，以及包括人臉及嘴唇等輪廓組成的68點人臉特徵點等。



下頷線[1, 17]

左眼眉毛[18, 22]

右眼眉毛[23, 27]

鼻樑[28, 31]

鼻子[32, 36]

左眼[37, 42]

右眼[43, 48]

上嘴唇外邊緣[49, 55]

下嘴唇外邊緣[56, 60]

下嘴唇內邊緣[61, 65]

上嘴唇內邊緣[66, 68]

- 人臉特徵提取

- 人臉特徵提取（Face Feature Extraction）也稱人臉表徵，它是對人臉進行特徵建模的過程。人臉特徵提取是將一幅人臉影像轉化為可以表徵人臉特點的特徵，具體表現形式為一串固定長度的數值。
- 人臉特徵提取過程的輸入是「一幅人臉圖」和「人臉五官關鍵點座標」，輸出是人臉對應的數值串（特徵）。
- 人臉特徵提取演算法實現的過程為：首先將五官關鍵點座標進行旋轉、縮放等操作來實現人臉對齊，然後再提取特徵並計算出數值串。

- 人臉比對


- 人臉比對（Face Compare）演算法實現的目的是衡量兩個人臉之間的相似度。
- 這一過程又分為兩類：一類是**確認**，是一對一進行影像比較的過程；另一類是**辨認**，是一對多進行影像匹配對比的過程。


人臉檢測和關鍵點定位


- DLIB介紹：
- dlib是一套包含了機器學習、計算機視覺、圖像處理等的函式庫，使用C++開發而成，目前廣泛使用於工業及學術界，也應用在機器人、嵌入式系統、手機、甚至於大型的運算架構中，而且最重要的是，它不但開源且完全免費，而且可跨平台使用（Linux、Mac OS、Windows），並且除了C++之外還提供了Python API，因此如果我們想要建立一套物件偵測系統，dlib是相當適合的平台。

如何安裝 DLIB

- (1) pip install Cmake
- (2) pip install boost
- (3) 下載你需要安裝 dlib 的版本（可到 <https://github.com/sachadee/Dlib> 下載）


 main ▾

 1 branch





 0 tags

Go to file

Code ▾

 sachadee Update README.md

44f92c0 on Nov 21, 2021 22 commits

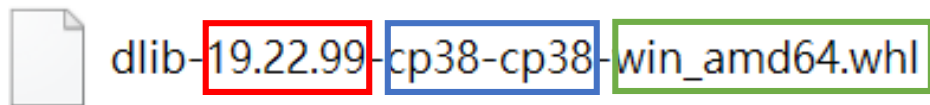
	README.md	Update README.md	2 years ago
	dlib-19.22.99-cp37-cp37m-win_amd6...	Add files via upload	2 years ago
	dlib-19.22.99-cp38-cp38-win_amd64....	Add files via upload	2 years ago
	dlib-19.22.99-cp39-cp39-win_amd64....	Add files via upload	2 years ago

- (4)將下載的檔案放置你要執行的目錄底下（例如我放在 C:\Users\JackyChuang 下）
- (5)執行 `pip install dlib-19.22.99-cp38-cp38-win_amd64.whl`

```
(tensorflow2_gpu) C:\Users\JackyChuang>pip install dlib-19.22.99-cp38-cp38-win_amd64.whl
Processing c:\users\jackychuang\dlib-19.22.99-cp38-cp38-win_amd64.whl
Installing collected packages: dlib
Successfully installed dlib-19.22.99
```

- (6)執行 `import dlib` 是否成功

```
import dlib
```



版本號碼
Python版本
-cp38 就是
Python3.8

64就是Win64位元作業系統

人臉檢測實現

- 首先利用Dlib函數庫的正向人臉檢測器get_frontal_face_detector進行人臉檢測，提取人臉外部矩形框，利用訓練好的Dlib的68點特徵預測器，進行人臉68點面部輪廓特徵提取，把所辨識出來的人臉輪廓點給標記出來。其程式處理流程如圖下所示。

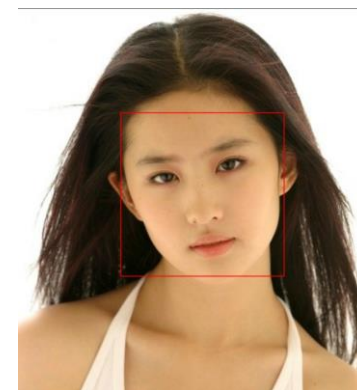
讀取影像檔 `img = io.imread("xxx.jpg")`

利用Dlib正向人臉檢測器檢測
`detector = dlib.get_frontal_face_detector()`
`faces = detector(img,1)`

利用Dlib 68點特徵預測器
`predictor = dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")`
`shape = predictor(img, faces[i])`

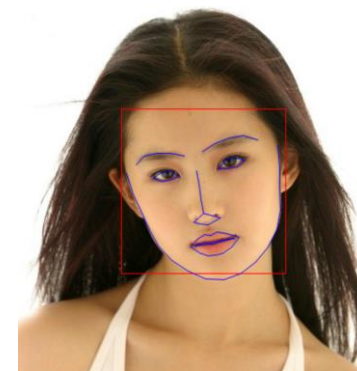
繪製矩形輪廓

`win.add_overlay(faces)`



繪製面部輪廓

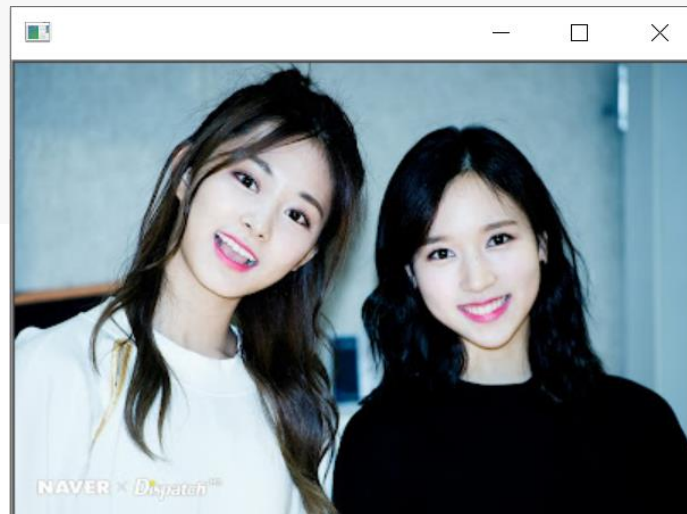
`win.add_overlay(shape)`



pip install scikit-image

```
import dlib
from skimage import io
# 使用 DLib 的正面人臉檢測器 front_face_detector (取得預設的臉部偵測器)
detector = dlib.get_frontal_face_detector()
# 根據shape_predictor方法載入68個特徵點模型，此方法為人臉表情識別的偵測器 (DLib 的68點模型)
modelname = "shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(modelname)
img = io.imread("twice.jpg")
# 生成 DLib 的影像視窗
win = dlib.image_window()
# 顯示要檢測的影像
win.set_image(img)
# 使用 detector 檢測器來檢測影像中的人臉
faces = detector(img,1)
print("人臉數:",len(faces))
```

人臉數: 2



```
# 繪製矩形輪廓
```

```
win.add_overlay(faces)
```

```
# 繪製兩個 overlay, 人臉外接矩形框與面部特徵框
```

```
for i, d in enumerate(faces):
```

```
    print("第", i+1, "個人臉的矩形框座標 :", "left:", d.left(),  
          "right:", d.right(), "top:", d.top(), "bottom:", d.bottom())
```

```
# 使用 predictor 來計算面部輪廓
```

```
    shape = predictor(img, faces[i])
```

```
# 繪製面部輪廓
```

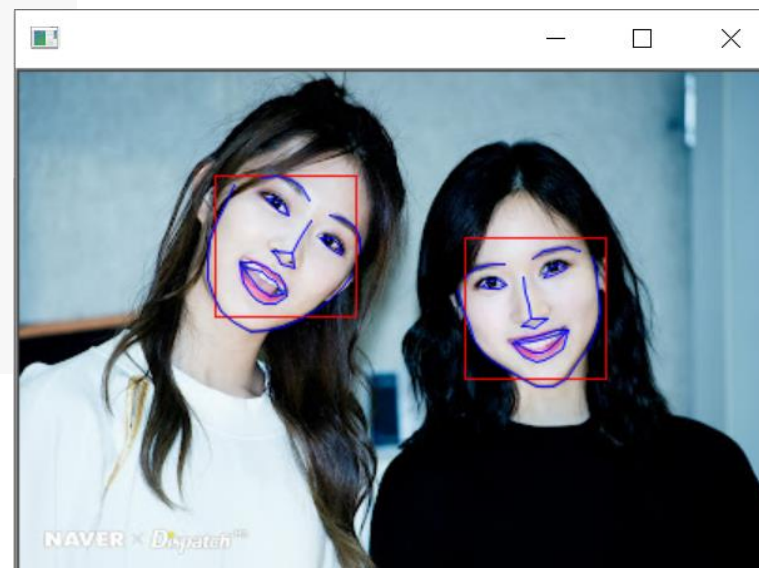
```
    win.add_overlay(shape)
```

```
# 保持影像
```

```
dlib.hit_enter_to_continue()
```

第 1 個人臉的矩形框座標 : left: 104 right: 179 top: 55 bottom: 130

第 2 個人臉的矩形框座標 : left: 237 right: 312 top: 88 bottom: 163



detector = dlib.get_frontal_face_detector()

1. 功能：人臉檢測畫框
2. 參數：無
3. 返回值：默認的人臉檢測器

predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

1. 功能：標記人臉關鍵點
2. 參數：'data/data_dlib/shape_predictor_68_face_landmarks.dat'：68個關鍵點模型地址
3. 返回值：人臉關鍵點預測器

dets = detector(img,1)

1. 功能：通過detector()來檢測圖片中的人臉區域
2. 參數：第一個參數img為圖片對象；第二個參數為上採樣的值。

採樣實際上也就是將一個尺寸較小的圖像通過算法使之變為尺寸較大的圖像，這裡我們將它的值設為1，能會幫助我們檢測到更多的人臉。

返回的dets表示模型檢測到的img中的所有人脸區域。因為一張圖片中可能有多張人臉，也就意味著可能會有多個人臉區域了，所以dets是一個數組。對於dets中的每一個元素d，都可以通過d.left(), d.top(), d.right(), d.bottom()分別獲取到d對應的人臉區域的最左，最上，最右和最下的四個值。

shape = predictor(img, box)

1. 功能：定位人臉關鍵點
2. 參數：img：一個numpy ndarray，包含8位灰度或RGB圖像
3. box：開始內部形狀預測的邊界框返回值：68個關鍵點的位置

使用WebCam 進行人臉偵測

```
import cv2 as cv
import dlib
# 使用 DLib 的正面人臉檢測器 front_face_detector (取得預設的臉部偵測器)
detector = dlib.get_frontal_face_detector()
# 利用官方提供的模型建構特徵提取器
modelname = "shape_predictor_68_face_landmarks.dat"
predictor = dlib.shape_predictor(modelname)
# 選擇第一隻攝影機
cap = cv.VideoCapture(0)
# 當攝影機打開時，對每個frame進行偵測
while cap.isOpened():
    # 讀出frame資訊
    ret, frame = cap.read()
    # 給68特徵點辨識取得一個轉換顏色的frame (BGR必須轉換成RGB)
    img = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    # 使用 detector 檢測器來檢測影像中的人臉
    faces = detector(img,0)
    # 使用 enumerate 函數遍歷序列中的元素以及他們的下標
    # k 即為人臉序號
    for k, d in enumerate(faces):
        # 使用 predictor 進行人臉關鍵點辨識, shape 為傳回的結果
        shape = predictor(img, d)
        # 繪製68個特徵點 (shape.part(i)是第i個特徵點)
        for index, pt in enumerate(shape.parts()):
            pt_pos = (pt.x, pt.y)
            cv.circle(frame, pt_pos, 1, (0,255,0),2)
            font = cv.FONT_HERSHEY_SIMPLEX
            cv.putText(frame, str(index+1),pt_pos,font,0.3,(0,0,255),1,cv.LINE_AA)
    cv.imshow("Frame", frame)
    key = cv.waitKey(10)
    if key == 27: # 當按下 Esc 鍵時離開
        print(key)
        break
cap.release()
cv.destroyAllWindows()
```


人臉辨識實戰

- 首先先安裝 face_recognition
- 直接執行命令即可：`pip --default-timeout=1000 install face_recognition`（加上了控制超時的參數）
- face_recognition 是 GitHub 主流的人臉辨識工具套件之一，該軟體套件使用Dlib函數庫中最先進的人臉辨識深度學習演算法，在LFW資料集中有99.38%的準確率。
- face_recognition實現人臉辨識的想法：
 - (1)給定想要辨識的人臉的圖片並進行編碼（每個人只需要一幅），並將這些不同的人臉編碼建構成一個串列。編碼其實就是將人臉圖片映射成一個128維的特徵向量。
 - (2)OpenCV讀取視訊並迴圈每一幀圖片，將每一幀圖片編碼後的128維特徵向量與前面輸入的人臉函數庫編碼串列裡的每個向量內積來衡量相似度，根據設定值來計算是否是同一個人。

- (3) 對辨識出來的人臉打標籤。人臉辨識實現的想法是採用HOG方法檢測輸入影像中的人臉。雖然使用CNN或HOG方法在量化面部之前（對面部編碼）都可以檢測輸入影像中的人臉，但CNN方法更準確（但更慢），而HOG方法更快（但不太準確）。雖然CNN人臉檢測更準確，但在沒有使用GPU執行的情況下即時檢測速度太慢。
- 人臉辨識實際上是對人臉進行編碼後再去計算兩張人臉的相似度，每張人臉是一個128維的特徵向量，最後利用兩個向量的內積來衡量相似度。

```
import os
import face_recognition
# 已知人臉照片的檔案路徑
path = "c:\\FaceImage"
files = os.listdir(path)
# 從目錄讀取所有的檔案至 files 中
know_names = []
know_faces = []
for file in files:
    filename = str(file)
    print(filename)
    know_names.append(filename)
    image = face_recognition.load_image_file(path+"\\ "+filename)
    encoding = face_recognition.face_encodings(image)[0]
    know_faces.append(encoding)

unknown_Image = face_recognition.load_image_file("c:\\TestFace\\test.jpg")
unknown_face = face_recognition.face_encodings(unknown_Image)[0]
results = face_recognition.compare_faces(know_faces, unknown_face, tolerance=0.36)
print("辨識結果如下 :")
for i in range(len(know_names)):
    print(know_names[i]+":", end=" ")
    if results[i]:
        print("相同")
    else:
        print("不相同")
```