Raspberry Pi Camera & Object Detection

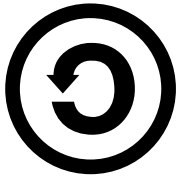
台灣樹莓派 <sosorry@raspberrypi.com.tw>
2021/04/16 @NCTU

CC (Creative Commons)

姓名標示 — 非商業性 — 相同方式分享

姓名標示 — 你必須給予 適當表彰、提供指向本授權條款的連結，以及 指出（本作品的原始版本）是否已被變更。你可以任何合理方式為前述表彰，但不得以任何方式暗示授權人為你或你的使用方式背書。

非商業性 — 你不得將本素材進行商業目的之使用。



相同方式分享 — 若你重混、轉換本素材，或依本素材建立新素材，你必須依本素材的授權條款來散布你的貢獻物。

關於我們

- Raspberry Pi 官方經銷商

about 台灣樹莓派

- 專注於 Raspberry Pi 應用與推廣
- 舉辦社群聚會 / 工作坊 / 讀書會 / 黑客松
- Website:
  - https://www.raspberrypi.com.tw/
- Facebook:
  - 搜尋 RaspberryPi.Taiwan
  - https://www.facebook.com/RaspberryPi.Taiwan

分享 x 教學

- COSCUP,MakerConf,PyCon,HKOSCon 講者
- 投影片
  - http://www.slideshare.net/raspberrypi-tw/presentations
- 程式碼
  - https://github.com/raspberrypi-tw

▶ 學習路徑 ◀

Pi選購指南 → Pi設定安裝 → Linux系統管理 / Python程式設計

I/O硬體控制

- GPIO學習套件 (初)
- 感測器學習套件 (基礎/進階) (中)
- 空氣盒子套件 (PiM25) (初)
- Win10開發套件 (初)
- 智慧開關套件 (初)
- Linux Driver 學習套件 (進)

無線/IoT

- RFID/NFC 門禁系統 (初)
- LoRa IoT 閘道器套件 (初)
- 生理資訊 監控IoT(藍牙) (初)
- 毫米波人流/熱點監控 (mmWave) (初)

相機/影像處理

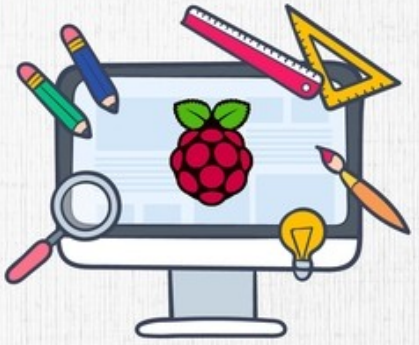
- 特色相機改裝套件 (初)
- 寵物小車套件 (初)
- 自控機器手臂套件 (中)
- 小鴨車套件 (Duckietown) (進)

人工智慧

- 驢車套件 (DonkeyCar) (初)
- AIY Vision Kit (中)
- Intel神經運算棒 (中)
- Google Coral USB加速器 (中)

語音/訊號處理

- 智慧音箱套件 (初)
- AIY Voice Kit (初)



(初) 初階課程　(中) 中階課程　(進) 進階課程

大綱

- 人臉偵測和物件辨識介紹
  - 人臉偵測歷史和比較
  - Haar Features 人臉偵測原理
  - 使用 Haar Features 做物件偵測
  - Dlib 人臉偵測原理
  - 使用 Dlib 做物件偵測

今日環境

- 硬體：Raspberry Pi 4B
- 作業系統：2021-01-11-raspios-buster-armhf-full.img

- 為了可以使用USB轉TTL傳輸線
  - 修改/boot/config.txt，新增三行
    - dtoverlay=pi3-miniuart-bt
    - core_freq=250
    - enable_uart=1

```
55 # Enable audio (loads snd_bcm
56 dtparam=audio=on
57 dtoverlay=pi3-miniuart-bt
58 core_freq=250
59 enable_uart=1
```

新增三行

  - 修改/boot/cmdline.txt，將quiet splash的quiet移除

```
1 dwc_otg.lpm_enable=0 console=serial0,115200
  console=tty1 root=/dev/mmcblk0p2 rootfstype=
  ext4 elevator=deadline fsck.repair=yes rootw
  ait quiet splash plymouth.ignore-serial-cons
  oles quiet init=/usr/lib/raspi-config/init_r
  esize.sh
```

刪除 quiet

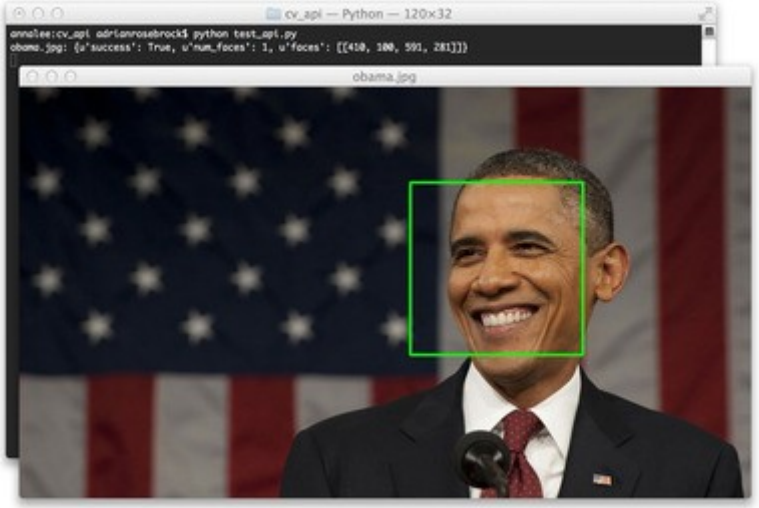
安裝今日所需軟體

- `$ sudo apt-get update`
- `$ sudo apt-get install -y cmake python3-opencv libjasper-dev python3-skimage libatlas3-base`
- `$ sudo pip3 install imutils dlib`

電腦如何分辨東西？

- 顏色
- 特徵
- 學習

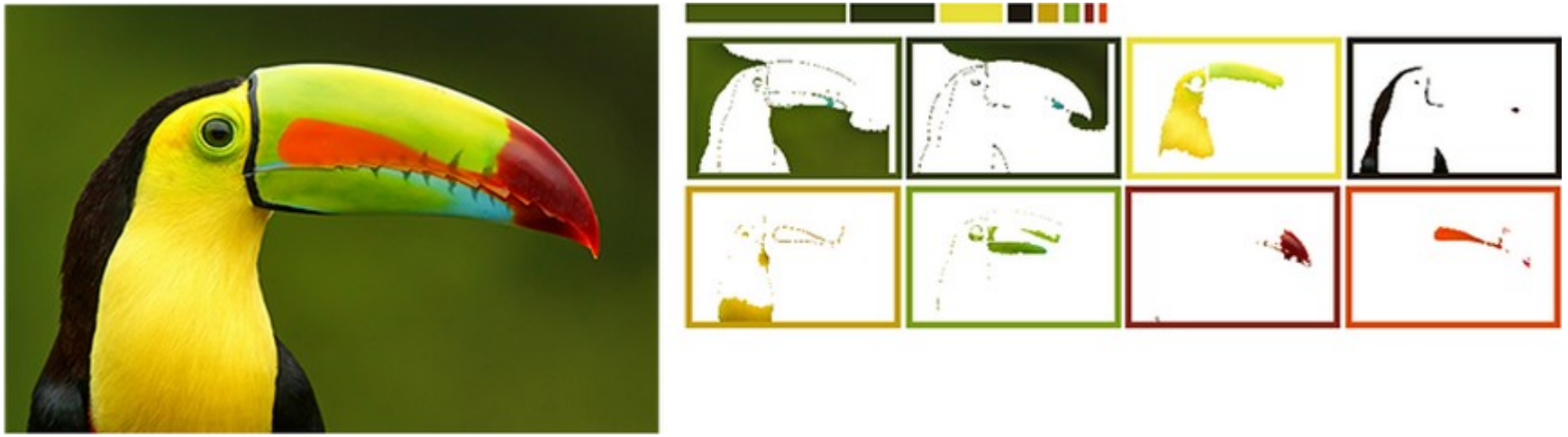
. . .



http://www.pyimagesearch.com/

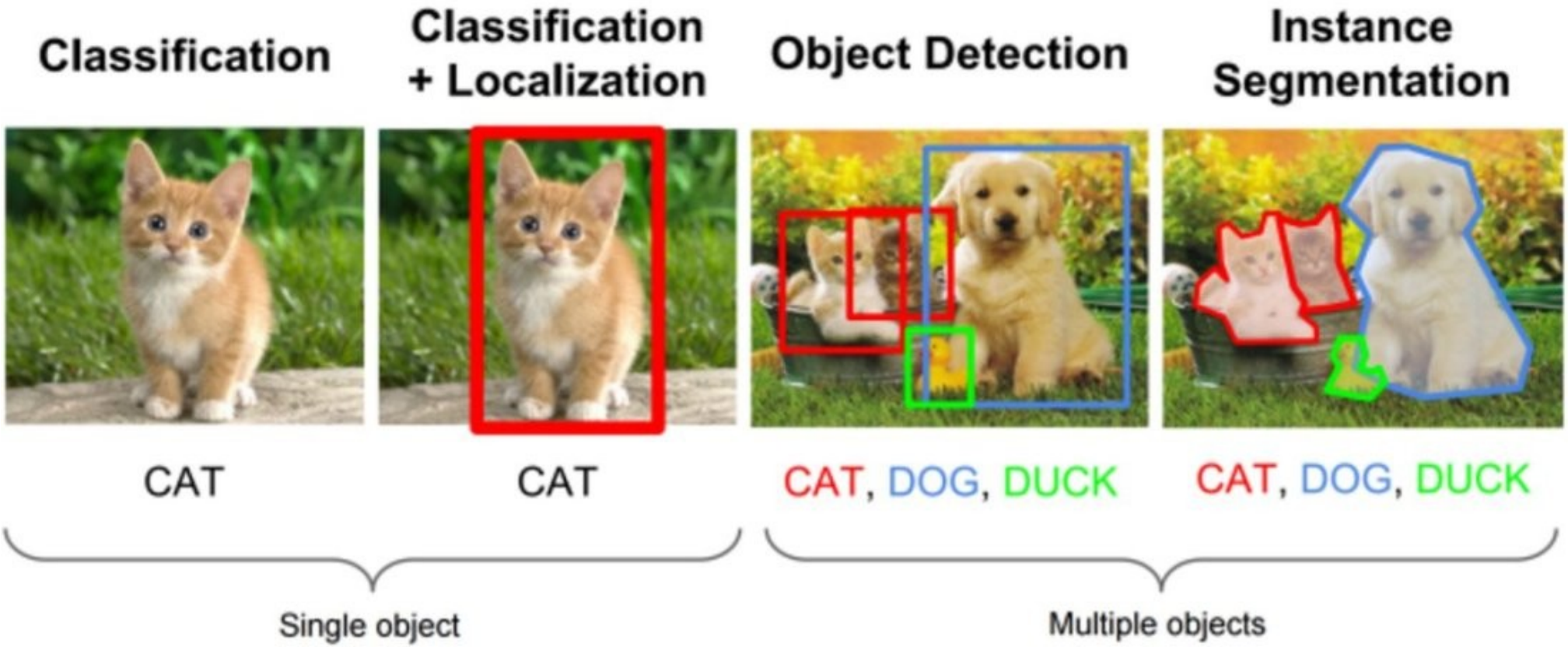
影像處理的意義

- 分離與萃取資訊
- 增強或平滑訊號
- 作為分群、特徵識別等電腦視覺應用的前處理



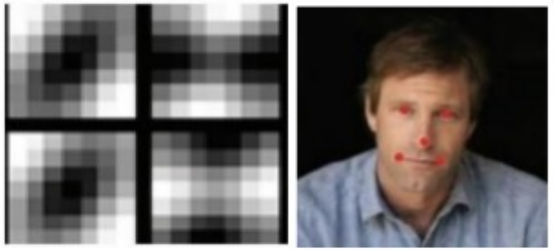
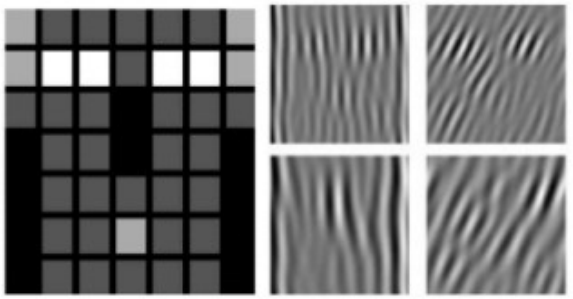
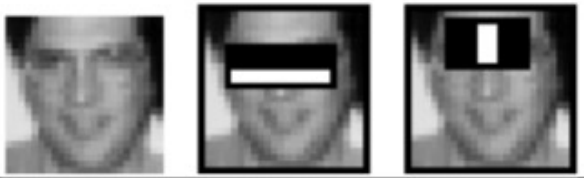
http://mkweb.bcgsc.ca/color-summarizer/

電腦視覺的應用



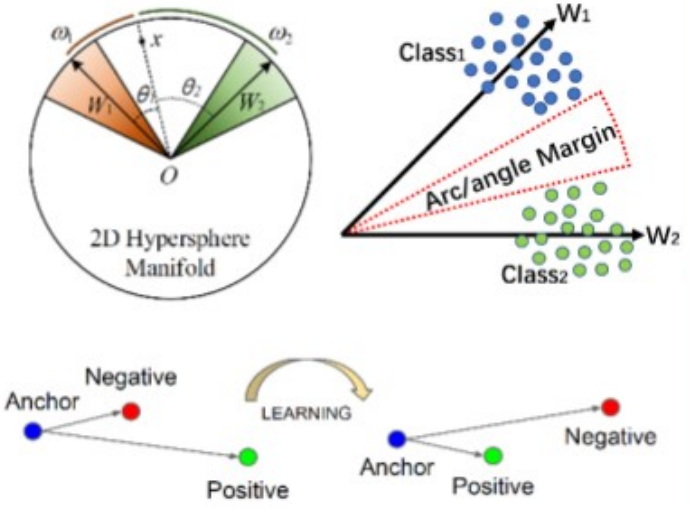
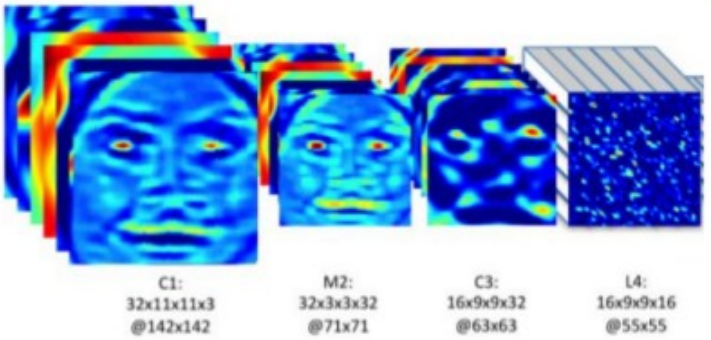
https://medium.com/lis-computer-vision-blogs/computer-vision-application-dc8412d9d392

人臉特徵發展歷史

Shallow	**PCANet** (*2015*) 86.28% (LFW) **LE (Learning-based Descriptor)** (*2010*) 84.45% (LFW)		Improved distinctiveness and compactness of codebook. Representation not robust to complex non-linear nature of face
Local Handcraft	**Local Binary Patterns** *2004* 66-79% (FERET) **Gabor Wavelets** *2002* >70% (LFW)		Robust to illumination and expression Removed the need for manual annotation Manually designing optimal encoding method and codebook is very difficult Susceptible to surface issues such as blurring.Results in uneven distribution which reduces informativity and compactness
	Haar Features *2001* 93.9% (Detection on MIT-CMU test set)		Provided method of detecting faces efficiently and effectively. Pioneered boosting based detection methods. Sensitive to illumination, pose, image quality
Holistic	**EigenFaces** *1991* 60.02% (LFW)		Simple, efficient method of recognizing faces in constrained environments. Relatively ineffective in face recognition in unconstrained conditions due to lack of robustness to lighting, pose, expression and image quality changes

https://arxiv.org/abs/1907.12739

人臉特徵發展歷史

Deep Learning	**ArcFace** *2018* 99.83% (LFW) **SphereFace** *2017* 99.42% (LFW) **FaceNet** *2015* 99.63% (LFW)	
	DeepFace *2014* 97.35% (LFW)	

Current state of the art on LFW and MegaFace Challenge. Although identification rate on the latter is still low (82.55% on SphereFace), verification rate is close to 100%. Significant improvements in handling expression, pose, illumination and occlusion.	
High computational cost due to extensive use of GPUs and very deep network architectures. Issues of poor annotation and noise, together with image quality affect performance	
Surpassed human verification accuracy in unconstrained settings for the first time. Commenced movement to focus research on deep learning methods such as CNNs	
High computational cost and suffers from loss of accuracy in situations involving spoofing, cross-pose, cross-age, low-resolution and make-up	

https://arxiv.org/abs/1907.12739

實驗一：人臉偵測 (1)

目的：瞭解 OpenCV 中機器學習函式

人臉偵測與人臉識別

- Facial Detection:
  Where is the face?

- Facial Recognition:
  Who is this?

https://www.youtube.com/watch?v=sWTvK72-SPU

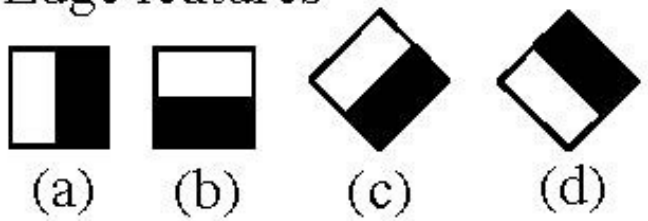
Haar Feature Cascade

- 由 Viola & Jones 提出，並由 Lienhart & Maydt 改善
- 採監督式學習的類神經網路演算法，並有以下特色
  - 特徵比對 (Haar features)
  - 積分影像計算 (Integral Image)
  - 串接分類器 (Cascade)
  - 學習機制 (AdaBoost)

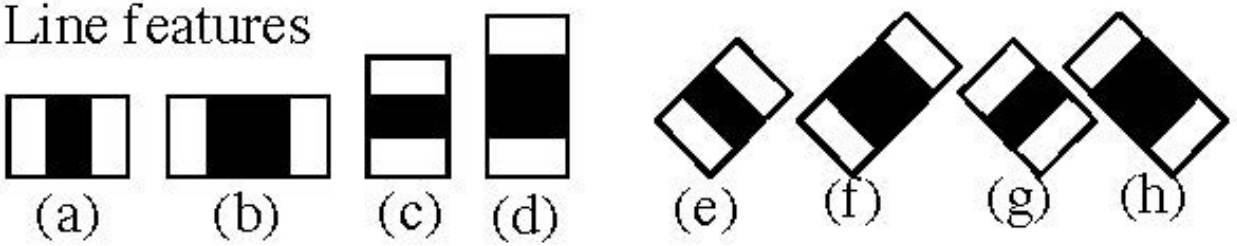
http://www.open-electronics.org/raspberry-pi-and-the-camera-pi-module-face-recognition-tutorial/

Haar-Like Features

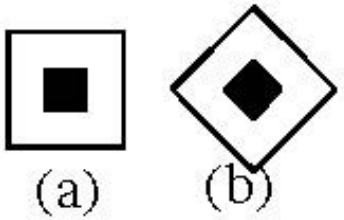
1. Edge features



2. Line features



3. Center-surround features



4. Special diagonal line feature used in [3,4,5]

http://archive.cnx.org/contents/d13cd00c-cabf-402b-844a-0cbe92aaef3f@2/algorithms

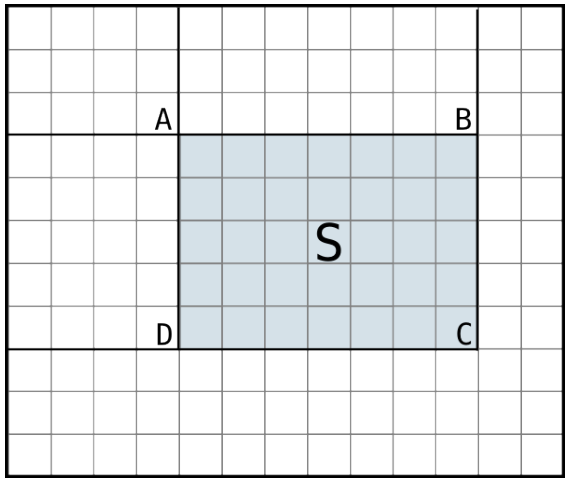
特徵比對

- 定義一個 24x24 的檢測窗口
- 每個特徵由左到右 , 由上到下滑動比對
- 計算特徵裡黑色和白色區域的灰階值
- 如果兩個區域灰階值的差大於門檻值 , 該特徵保留

https://www.youtube.com/watch?v=sWTvK72-SPU

積分影像計算

- 計算灰階值可以先將灰階值的加總 (積分) 算一輪
- 任何大小的矩形灰階值加總可由四頂點的矩形灰階值加減取得



Original

5	**2**	**3**	4	1
1	**5**	**4**	2	3
2	2	1	3	4
3	5	6	4	5
4	1	3	2	6

Integral

5	7	10	14	15
6	13	**20**	26	30
8	17	25	34	42
11	25	39	52	65
15	30	47	62	81

$5 + 2 + 3 + 1 + 5 + 4 = 20$

Original

5	2	3	4	1
1	**5**	**4**	**2**	3
2	**2**	**1**	**3**	4
3	5	6	4	5
4	1	3	2	6

$5 + 4 + 2 + 2 + 1 + 3 = 17$

Integral

5	7	10	**14**	15
6	13	20	26	30
8	17	25	**34**	42
11	25	39	52	65
15	30	47	62	81

$34 - 14 - 8 + 5 = 17$

http://computervisionwithvaibhav.blogspot.tw/2015/08/viola-jones-in-nut-shell.html

Haar Cascade

- Cascade 是一連串 Haar-like features 的組合所形成的分類器 (classifier)
- Haar Cascade = Classifier

https://www.youtube.com/watch?v=sWTvK72-SPU

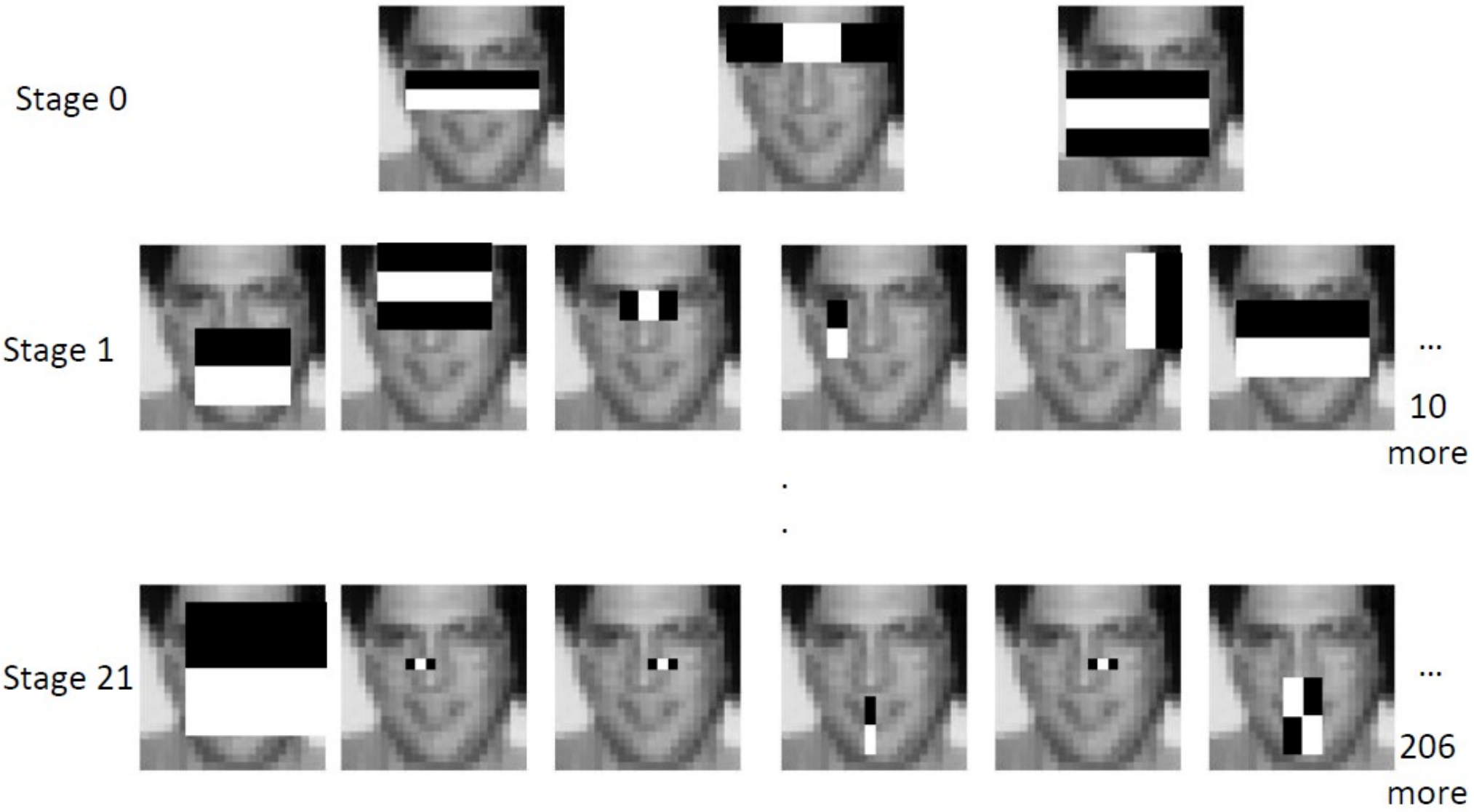
弱分類器

- Feature:  and 
- Classifier:  

- **單一的分類器準確度不夠，因此也稱為是弱分類器**

https://www.youtube.com/watch?v=sWTvK72-SPU

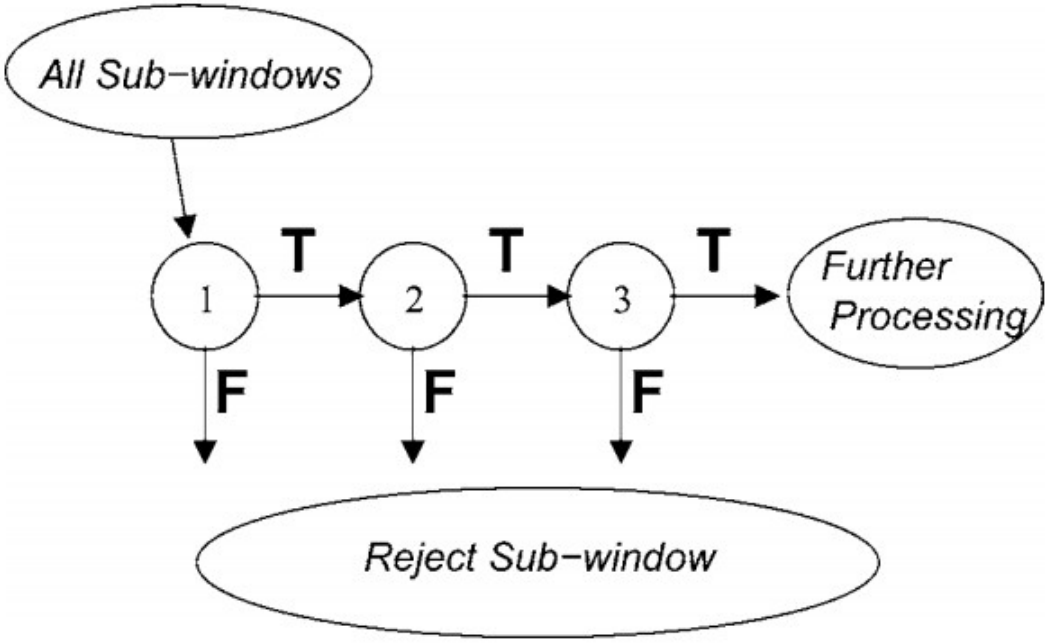
串接多個弱分類器



https://www.researchgate.net/post/Can_someone_answer_a_few_questions_on_Haar_Features_and_Adaboost

串接弱分類器的好處

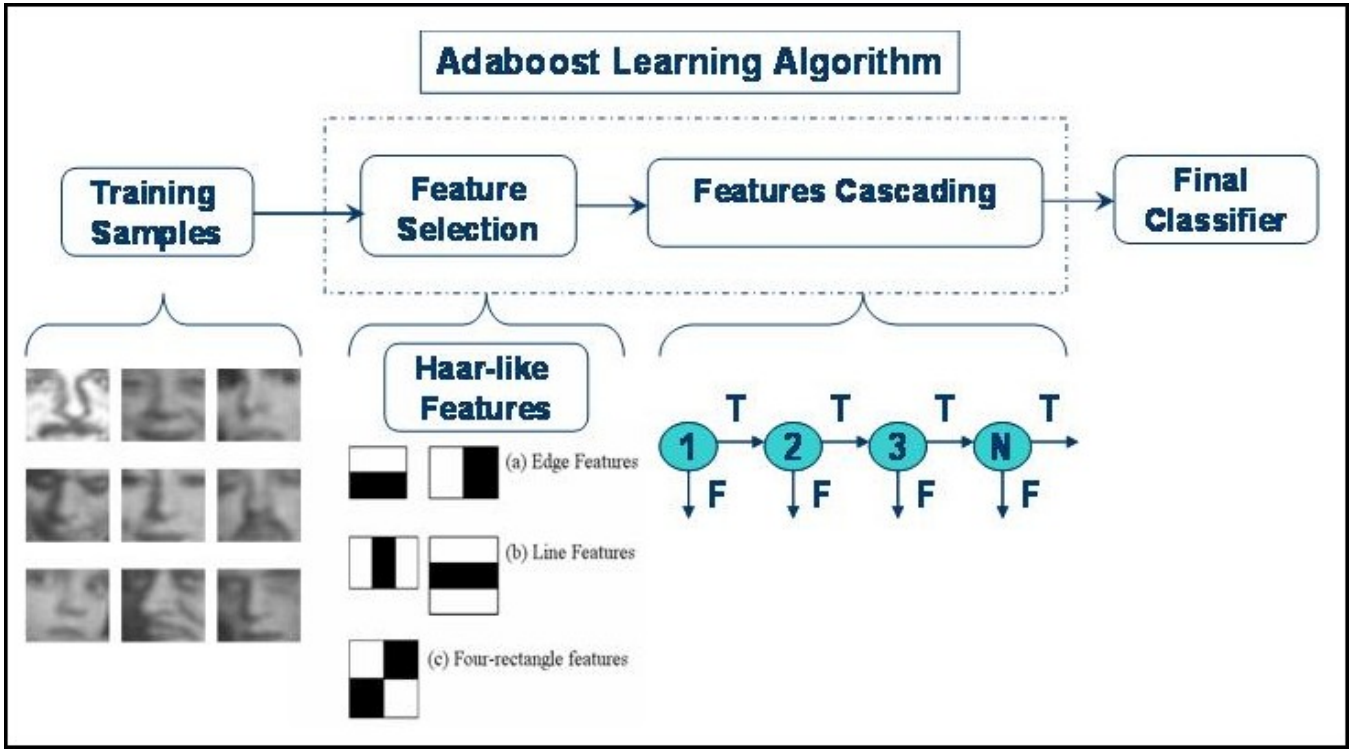
- 單一分類器的準確度雖然只略大於 50%，但如果同時通過多個弱分類器，其準確度將會大副提昇
- 使用弱分類器的好處是可以將未達到門檻值的特徵 (feature) 在早期就先去除掉



https://www.youtube.com/watch?v=sWTvK72-SPU

AdaBoost(Adaptive Boosting)

- Adaboost 在學習階段可以組合效果好的分類器，並根據監督式的學習過程調整不同分類器的權重，用來建立適合的偵測模型



https://www.youtube.com/watch?v=sWTvK72-SPU

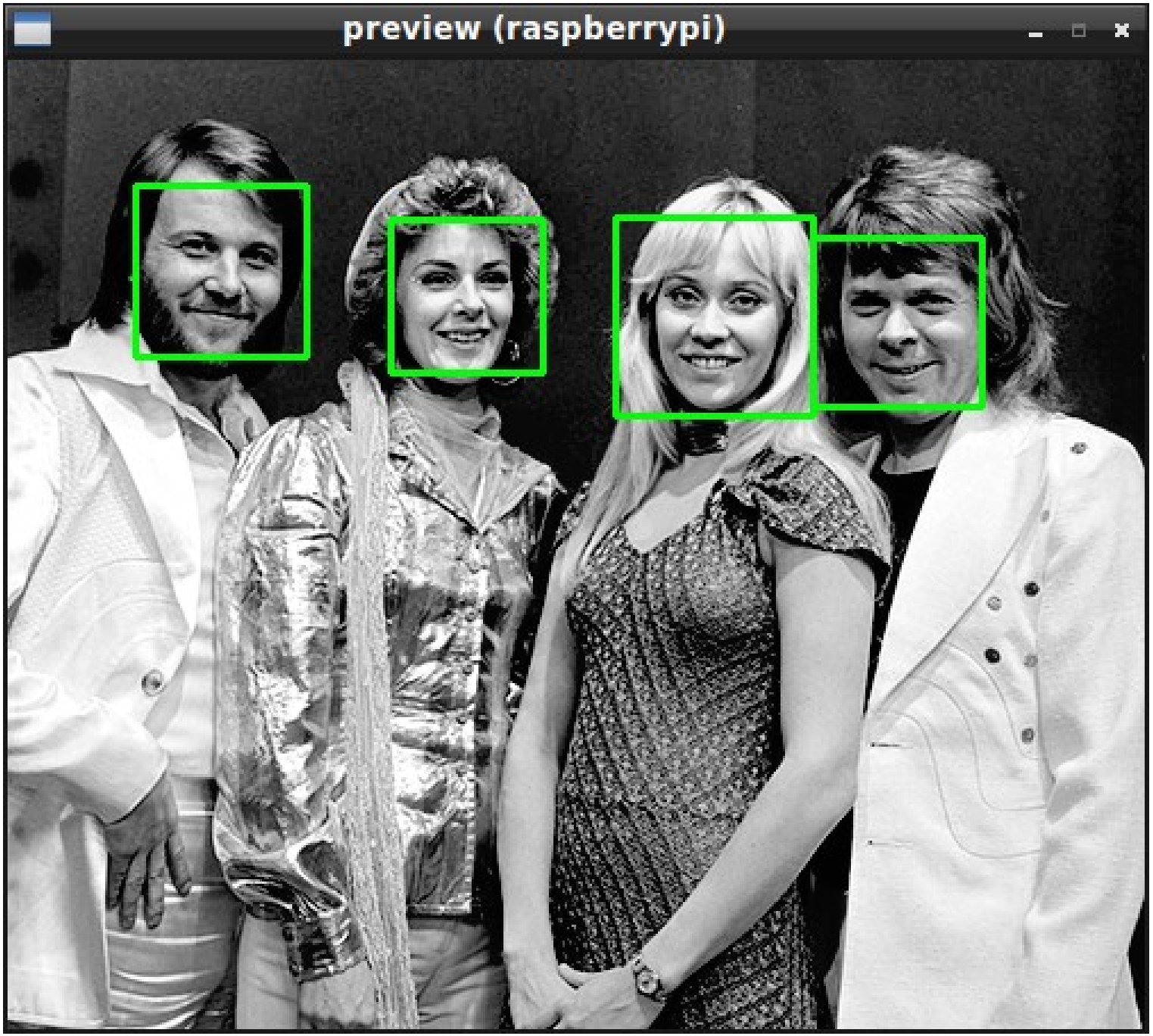
載入圖檔並辨識

```
faceCascade = cv2.CascadeClassifier(sys.argv[2])
image = cv2.imread(sys.argv[1])
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30),
    flags = cv2.cv.CV_HAAR_SCALE_IMAGE
)

for (x, y, w, h) in faces:
    cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

https://sites.google.com/site/5kk73gpu2012/assignment/viola-jones-face-detection

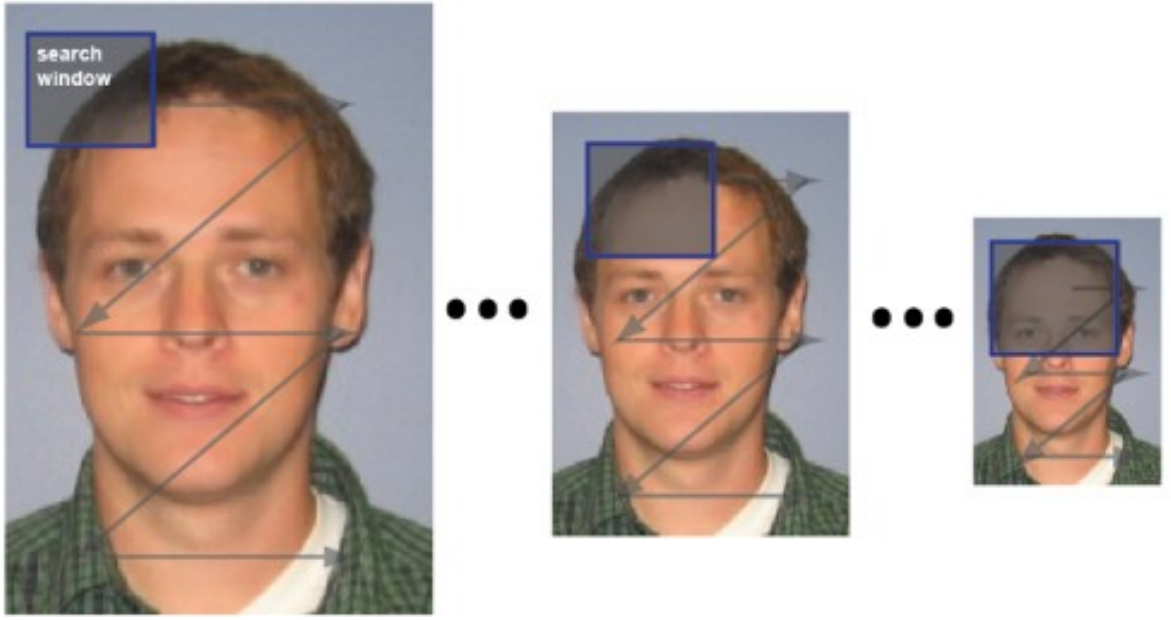
DEMO
image_face_detect.py

```
$ cd ~/face_detect/opecv_demo
$ python3 image_face_detect.py abba.png haarcascade_frontalface_default.xml
```


preview (raspberrypi)

可調整的參數

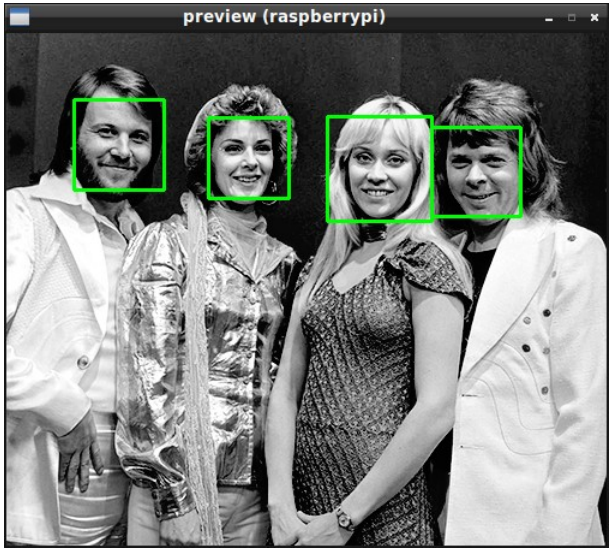
- `scaleFactor`：檢測視窗縮放比率
- `minNeighbors`：檢測區域鄰域內最少包含的檢測出的備選人臉區域 (次數)
- `minSize`：被檢測物體的最小尺寸



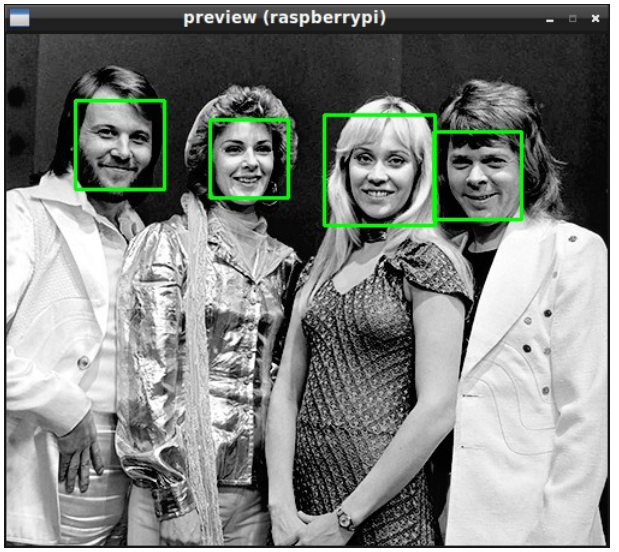
https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html

scaleFactor

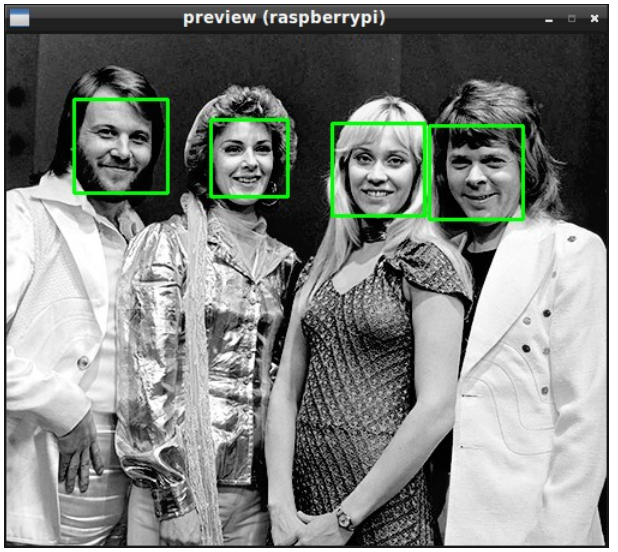
- minNeighbors=5, minSize=(30, 30)



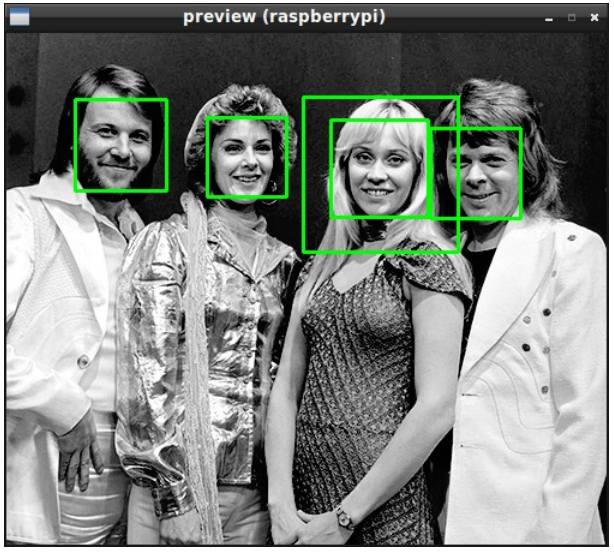
scaleFactor=1.1



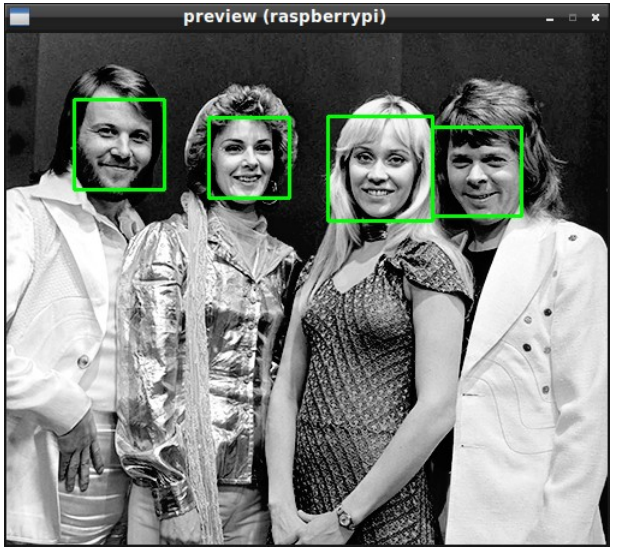
1.2



1.3



1.4

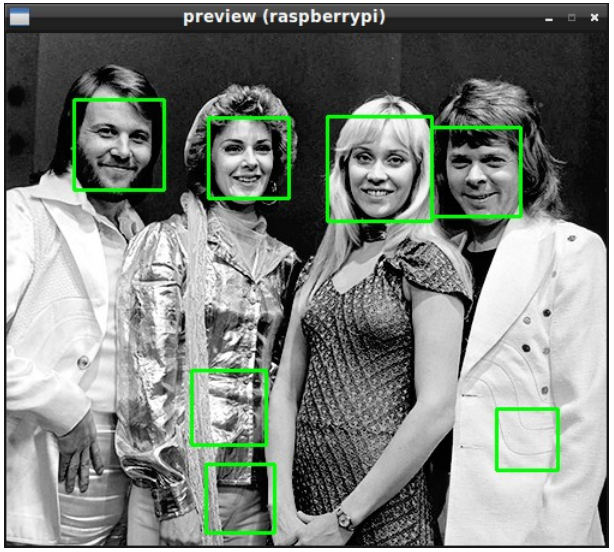


1.5

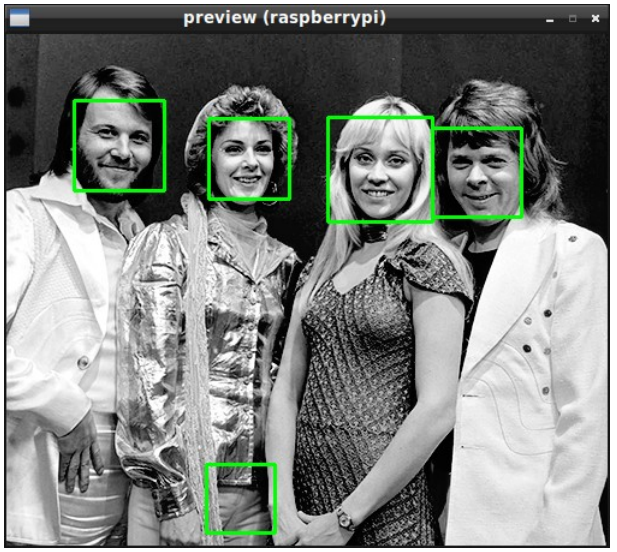
1.6

minNeighbors

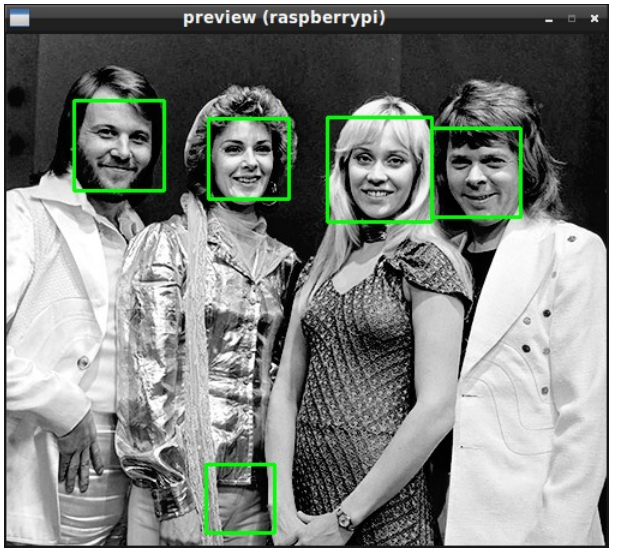
- scaleFactor=1.1, minSize=(30, 30)



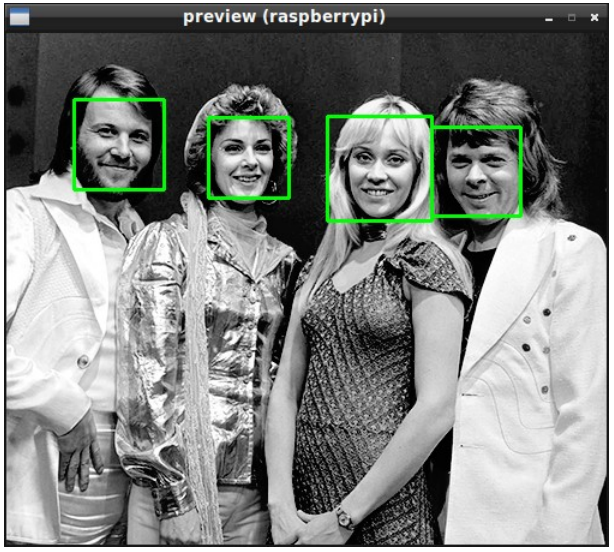
minNeighbors=1



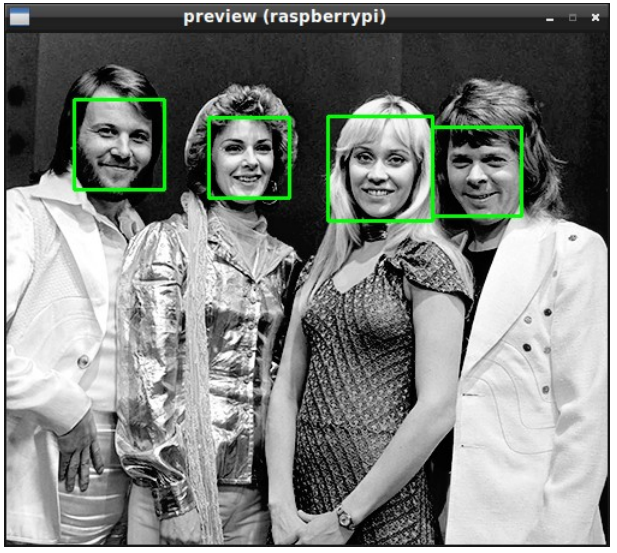
2



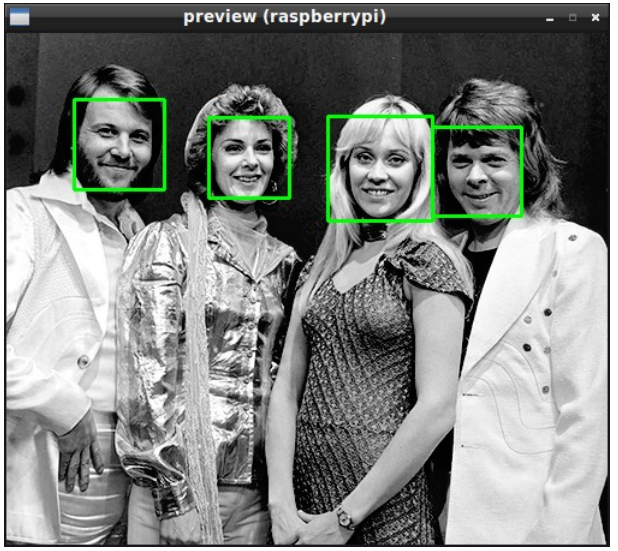
3



5



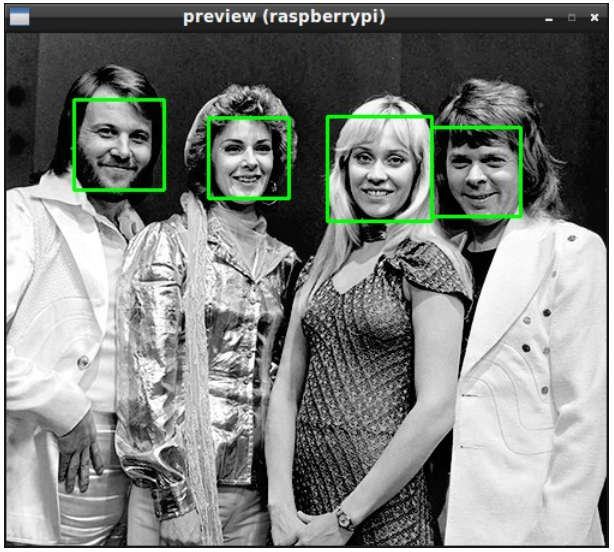
10



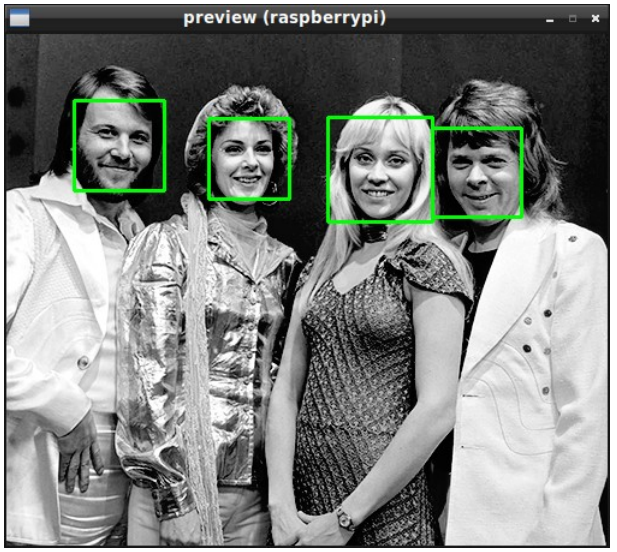
20

minSize(x, y)

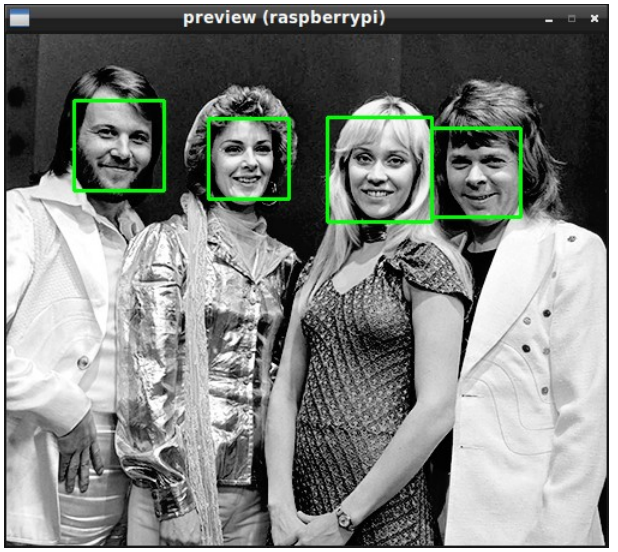
- scaleFactor=1.1, minNeighbosr=5



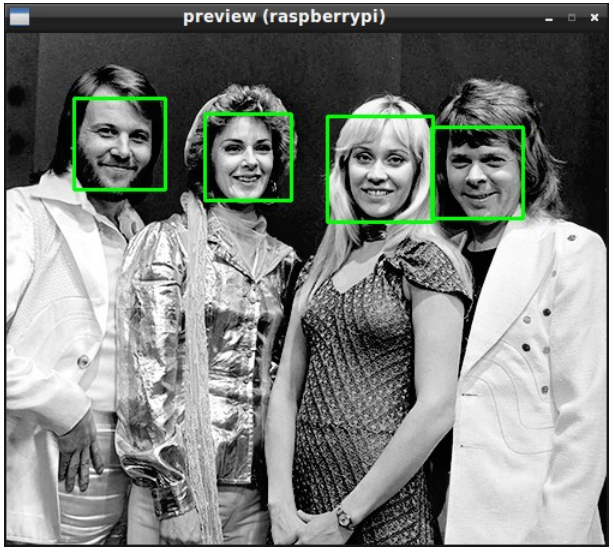
minSize=(15, 15)



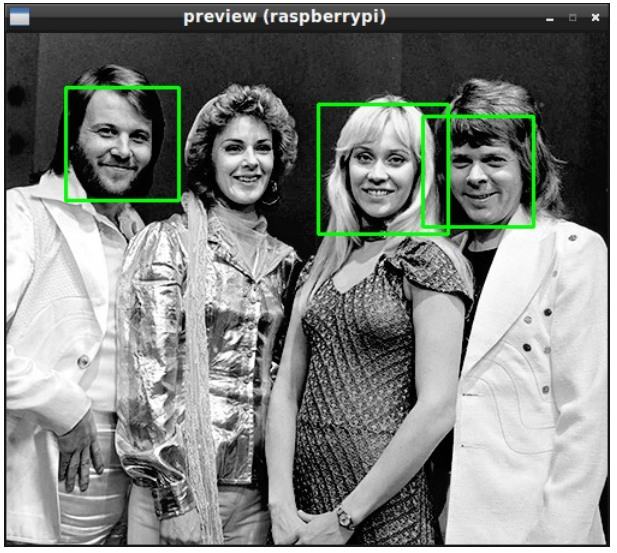
(30, 30)



(60, 60)



(90, 90)

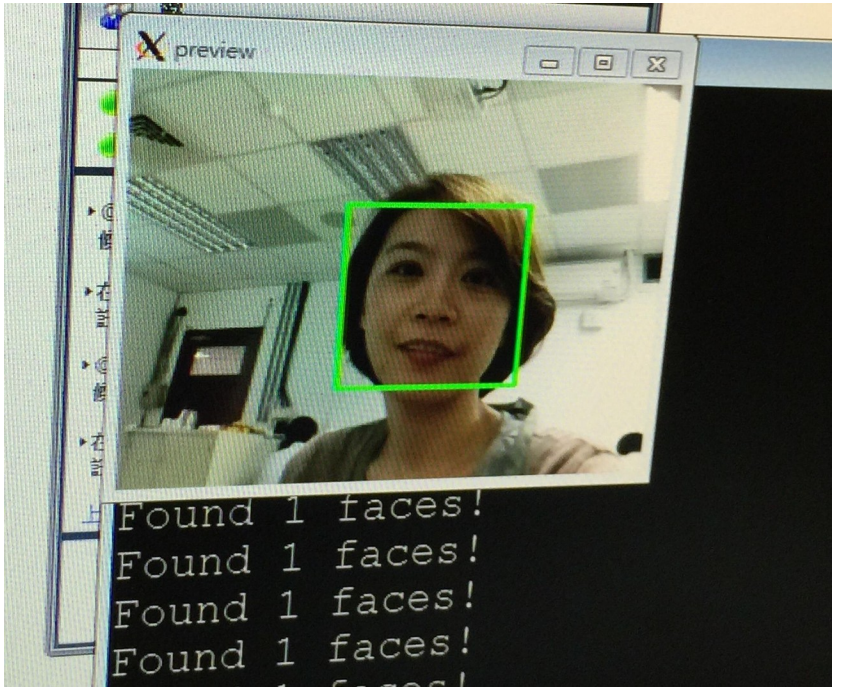


(120 ,120)

(150, 150)

讀取 Camera 並辨識

```
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=cv2.cv.CV_HAAR_SCALE_IMAGE
    )

    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)
```



DEMO
camera_face_detect.py

```
$ cd ~/face_detect/opecv_demo
$ python3 camera_face_detect.py haarcascade_frontalface_default.xml
```

觀察不同參數的影響

DEMO
camera_face_detect_param.py

```
$ cd ~/face_detect/opecv_demo
$ python3 camera_face_detect_param.py haarcascade_frontalface_default.xml
```

訓練自己的 Haar 分類器

如果不是使用 OpenCV3.x 編譯

- `$ cd ~`
- `$ wget http://raspberrypi-tw.s3.amazonaws.com/download/opencv3.4.10.tar.gz`
- `$ tar zxvf opencv3.4.10.tar.gz`
- `$ sudo cp -f opencv3.4.10/bin/* /usr/local/bin`
- `$ sudo cp -f opencv3.4.10/lib/* /usr/local/lib`

常見資料來源

- Caltech101
  - http://www.vision.caltech.edu/Image_Datasets/Caltech101/
- CIFAR-10
  - https://www.cs.toronto.edu/~kriz/cifar.html
- BioID Face Database
  - https://www.bioid.com/facedb/
- ImageNet
  - http://www.image-net.org/

BioID-Face Database

- 免費，開源的人臉影像資料庫
- 為真實世界資料，有多種照明 / 背景 / 人臉大小
- 包含 1521 張灰階影像檔 (384x286 pixel)，為 23 位受測者的正面影像
- 影像檔案使用 PGM(portable gray map) 格式
- 眼睛位置為純文字檔 (座標位置)

https://ftp.uni-erlangen.de/pub/facedb/readme.html

PGM 格式

- PBM 格式用 ASCII 碼表示單色點陣圖（1988 年）
- PBM 是單色，PGM 是灰度圖，PPM 使用 RGB 顏色
- 檔案包含格式，圖像尺寸和實際資料

檔案描述子	類型	編碼
P1	點陣圖	ASCII
P2	灰度圖	ASCII
P3	像素圖	ASCII
P4	點陣圖	二進位
P5	灰度圖	二進位
P6	像素圖	二進位

```
P1
# This is an example bitmap of the letter "J"
6 10
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 1 0
1 0 0 0 1 0
0 1 1 1 0 0
0 0 0 0 0 0
0 0 0 0 0 0
```

PBM

```
P2
6 6
255
0 0 0 150 0 0
0 0 0 150 0 0
0 0 0 150 0 0
0 150 0 150 0 0
0 150 150 150 0 0
0 0 0 0 0 0
```

PGM

https://zh.wikipedia.org/wiki/PBM 格式

用 Python 的 PIL 套件將 PGM 轉成 JPG

```
PHOTO_DIR = "BioID"
NEG_DIR   = "neg"

def pgm2jpg(in_dir, out_dir):
    filepaths = glob.glob(os.path.join(in_dir, '*.pgm'))

    for fp in filepaths:
        im = Image.open(fp)
        im = im.resize((150, 150), Image.ANTIALIAS)
        fn = ntpath.basename(fp)[0:10] + '.jpg'
        out_file = os.path.join(out_dir, fn)
        im.save(out_file)

if __name__ == "__main__":
    pgm2jpg(PHOTO_DIR, NEG_DIR)
```

使用前先移動 BioID 目錄

```
$ mv ~/BioID/ ~/face_detect/opencv_demo
```

DEMO
pgm_to_jpg.py

```
$ cd ~/face_detect/opecv_demo
$ python3 pgm_to_jpg.py
```

執行結果

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ python3 pgm_to_jpg.py
neg/BioID_0723.jpg
neg/BioID_0101.jpg
neg/BioID_1400.jpg
neg/BioID_0350.jpg
neg/BioID_0907.jpg
neg/BioID_0695.jpg
neg/BioID_1196.jpg
neg/BioID_0076.jpg
neg/BioID_1020.jpg
neg/BioID_0169.jpg
neg/BioID_0137.jpg
neg/BioID_1097.jpg
neg/BioID_0726.jpg
neg/BioID_1508.jpg
neg/BioID_0943.jpg
neg/BioID_0047.jpg
neg/BioID_0865.jpg
neg/BioID_0015.jpg
neg/BioID_0415.jpg
neg/BioID_0950.jpg
neg/BioID_1175.jpg
neg/BioID_0002.jpg
neg/BioID_0911.jpg
```

目錄結構

- $ tree

```
├── data
│   ├── cascade.xml
│   ├── params.xml
│   ├── stage0.xml
│   └── ...
├── neg
│   ├── BioID_0000.jpg
│   ├── ...
│   └── BioID_1520.jpg
├── neg.txt
├── pos.vec
├── pos
│   ├── 0001_0015_0016_0089_0089.jpg
│   ├── ...
│   └── pos.txt
└── sample.jpg
```

產生負樣本圖片

訓練 Haar Cascade 步驟

- 建立負樣本
  - 例如從 BioID-Face Database 取得人臉
  - 將 pgm 格式轉成 jpg 格式
- 建立正樣本
  - 拍照取得單一正樣本
  - 用 opencv_createsamples 產生多組正樣本
- 根據正樣本轉換成向量格式
- 使用 opencv_traincascade 訓練分類器
- 實際測試模型

拍照取得單一正樣本，按 q 存檔

```
import cv2
import imutils

cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    frame = imutils.resize(frame, 320)
    cv2.imshow("preview", frame)
    if cv2.waitKey(1) & 0xFF == ord("q"):
        cv2.imwrite("capture.jpg", frame)
        break

cap.release()
cv2.destroyAllWindows()
```

按 'q' 存檔離開

DEMO
camera_capture.py

```
$ cd ~/face_detect/opecv_demo
$ python3 camera_capture.py
```

執行結果

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ python3 camera_capture.py
```

按 'q' 存檔離開

框出正様本

```
def callback(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        ix, iy = x, y
    elif event == cv2.EVENT_LBUTTONUP:
        if mode == True:
            cv2.rectangle(img, (ix,iy), (x,y), (0,255,255), 0)
            crop = img[iy:y, ix:x]
            cv2.imwrite("sample.jpg", crop)

if __name__ == "__main__":
    img = cv2.imread(imagePath)
    img_copy = img.copy()
    cv2.setMouseCallback("sample", callback)

    while True:
        cv2.imshow("sample", img)
```

按 'q' 存檔離開

DEMO
extract_sample.py

```
$ cd ~/face_detect/opecv_demo
$ python3 extract_sample.py capture.jpg
```

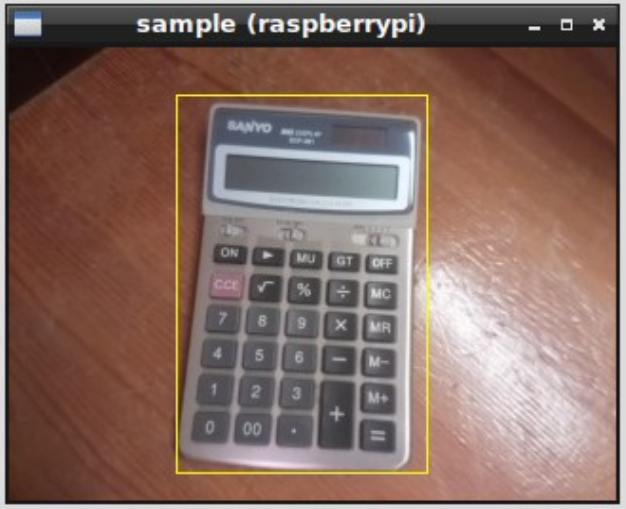
執行結果

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ python3 extract_sample.py
 capture.jpg
Dn => ix= 88 iy= 25
Up => ix= 220 iy= 225
```

選取外框後按 'q' 存檔離開



目錄結構

- $ tree

```
├── data
│   ├── cascade.xml
│   ├── params.xml
│   ├── stage0.xml
│   └── ...
├── neg
│   ├── BioID_0000.jpg
│   ├── ...
│   └── BioID_1520.jpg
├── neg.txt
├── pos.vec
├── pos
│   ├── 0001_0015_0016_0089_0089.jpg
│   ├── ...
│   └── pos.txt
└── sample.jpg
```

建立負樣本檔案列表

- `$ find neg -iname "*.jpg" > neg.txt`

目錄結構

- $ tree

```
├── data
│   ├── cascade.xml
│   ├── params.xml
│   ├── stage0.xml
│   └── ...
├── neg
│   ├── BioID_0000.jpg
│   ├── ...
│   └── BioID_1520.jpg
├── neg.txt
├── pos.vec
├── pos
│   ├── 0001_0015_0016_0089_0089.jpg
│   ├── ...
│   └── pos.txt
└── sample.jpg
```

產生負樣本檔案列表

快速建立正樣本
$ opencv_createsamples

pi@raspberrypi: ~

File Edit Tabs Help

```
pi@raspberrypi:~ $ opencv_createsamples
Usage: opencv_createsamples
  [-info <collection_file_name>]
  [-img <image_file_name>]
  [-vec <vec_file_name>]
  [-bg <background_file_name>]
  [-num <number_of_samples = 1000>]
  [-bgcolor <background_color = 0>]
  [-inv] [-randinv] [-bgthresh <background_color_threshold = 80>]
  [-maxidev <max_intensity_deviation = 40>]
  [-maxxangle <max_x_rotation_angle = 1.100000>]
  [-maxyangle <max_y_rotation_angle = 1.100000>]
  [-maxzangle <max_z_rotation_angle = 0.500000>]
  [-show [<scale = 4.000000>]]
  [-w <sample_width = 24>]
  [-h <sample_height = 24>]
  [-maxscale <max sample scale = -1.000000>]
  [-rngseed <rng seed = 12345>]
pi@raspberrypi:~ $
```

opencv_createsamples 參數說明

- -img: 單張正樣本圖片路徑
- -bg: 合成背景圖片路徑
- -info: 建立正樣本所存放的路徑
- -maxxangle: 對樣本圖片的 x 軸方向最大扭曲弧度
- -num: 要建立的正樣本數量
- -vec: 要訓練的正樣本路徑 (向量格式)
- -w: 建立樣本圖片的寬度 (需和訓練一致)
- -h: 建立樣本圖片的高度 (需和訓練一致)

- $ opencv_createsamples -img sample.jpg -bg neg.txt -info pos/pos.txt -maxxangle 0.3 -maxyangle 0.3 -maxzangle 0.3 -num 100

執行結果

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ opencv_createsamples -img
 sample.jpg -bg neg.txt -info pos/pos.txt -maxxangle 0.3 -maxyangle 0.3 -maxzang
le 0.3 -num 100
Info file name: pos/pos.txt
Img file name: sample.jpg
Vec file name: (NULL)
BG  file name: neg.txt
Num: 100
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 0.3
Max y angle: 0.3
Max z angle: 0.3
Show samples: FALSE
Width: 24
Height: 24
Max Scale: -1
RNG Seed: 12345
Create test samples from single image applying distortions...
Open background image: neg/BioID_0477.jpg
Open background image: neg/BioID_1365.jpg
Open background image: neg/BioID_1062.jpg
```

目錄結構

- $ tree

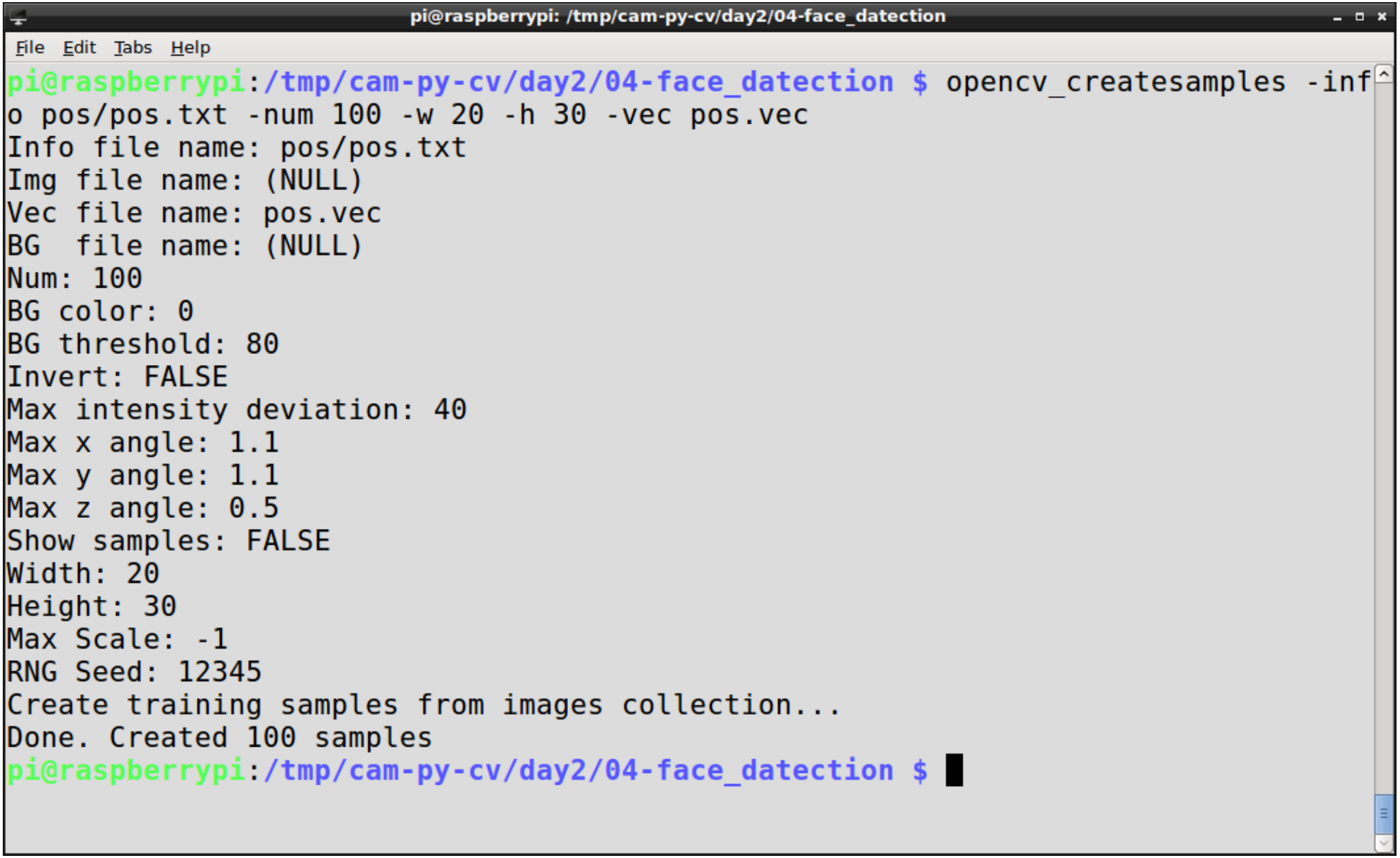
```
├── data
│   ├── cascade.xml
│   ├── params.xml
│   ├── stage0.xml
│   └── ...
├── neg
│   ├── BioID_0000.jpg
│   ├── ...
│   └── BioID_1520.jpg
├── neg.txt
├── pos.vec
├── pos
│   ├── 0001_0015_0016_0089_0089.jpg
│   ├── ...
│   └── pos.txt
└── sample.jpg
```

產生正樣本圖片和正樣本檔案列表

將正樣本轉換成向量格式檔案

- `$ opencv_createsamples -info pos/pos.txt -num 100 -w 20 -h 30 -vec pos.vec`

執行結果


pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection
File Edit Tabs Help
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ opencv_createsamples -inf
o pos/pos.txt -num 100 -w 20 -h 30 -vec pos.vec
Info file name: pos/pos.txt
Img file name: (NULL)
Vec file name: pos.vec
BG file name: (NULL)
Num: 100
BG color: 0
BG threshold: 80
Invert: FALSE
Max intensity deviation: 40
Max x angle: 1.1
Max y angle: 1.1
Max z angle: 0.5
Show samples: FALSE
Width: 20
Height: 30
Max Scale: -1
RNG Seed: 12345
Create training samples from images collection...
Done. Created 100 samples
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $

目錄結構

- $ tree

```
├── data
│   ├── cascade.xml
│   ├── params.xml
│   ├── stage0.xml
│   └── ...
├── neg
│   ├── BioID_0000.jpg
│   ├── ...
│   └── BioID_1520.jpg
├── neg.txt
├── pos.vec
├── pos
│   ├── 0001_0015_0016_0089_0089.jpg
│   ├── ...
│   └── pos.txt
└── sample.jpg
```

訓練 Cascade
$ opencv_traincascade

pi@raspberrypi: ~

File Edit Tabs Help

```
pi@raspberrypi:~ $ opencv_traincascade
Usage: opencv_traincascade
  -data <cascade_dir_name>
  -vec <vec_file_name>
  -bg <background_file_name>
  [-numPos <number_of_positive_samples = 2000>]
  [-numNeg <number_of_negative_samples = 1000>]
  [-numStages <number_of_stages = 20>]
  [-precalcValBufSize <precalculated_vals_buffer_size_in_Mb = 1024>]
  [-precalcIdxBufSize <precalculated_idxs_buffer_size_in_Mb = 1024>]
  [-baseFormatSave]
  [-numThreads <max_number_of_threads = 4>]
  [-acceptanceRatioBreakValue <value> = -1>]
--cascadeParams--
  [-stageType <BOOST(default)>]
  [-featureType <{HAAR(default), LBP, HOG}>]
  [-w <sampleWidth = 24>]
  [-h <sampleHeight = 24>]
--boostParams--
  [-bt <{DAB, RAB, LB, GAB(default)}>]
```

opencv_traincascade 參數說明

- -data: 訓練完成的分類器儲存路徑
- -vec: 具有正樣本的檔案
- -bg: 負樣本圖片的儲存路徑
- -numStages: 訓練的分類器的級數
- -numPos: 每個分類階段訓練使用的正樣本數
- -numNeg: 每個分類階段訓練使用的負樣本數
- -featureType: 指定特徵型別，可用HAAR(預設)/LBP/HOG
- -w: 訓練樣本圖片的寬度(需和建立時一致)
- -h: 訓練樣本圖片的高度(需和建立時一致)

- `$ opencv_traincascade -data data -vec pos.vec -bg neg.txt -numStages 10 -numPos 100 -numNeg 100 -w 20 -h 30`

開始訓練

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ opencv_traincascade -data
 data -vec pos.vec -bg neg.txt -numStages 10 -numPos 100 -numNeg 100 -w 20 -h 30

PARAMETERS:
cascadeDirName: data
vecFileName: pos.vec
bgFileName: neg.txt
numPos: 100
numNeg: 100
numStages: 10
precalcValBufSize[Mb] : 1024
precalcIdxBufSize[Mb] : 1024
acceptanceRatioBreakValue : -1
stageType: BOOST
featureType: HAAR
sampleWidth: 20
sampleHeight: 30
boostType: GAB
minHitRate: 0.995
maxFalseAlarmRate: 0.5
weightTrimRate: 0.95
maxDepth: 1
maxWeakCount: 100
mode: BASIC
```

可能會提早結束

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
|   2|         1|       0.2|
+----+----------+----------+
END>
Training until now has taken 0 days 0 hours 0 minutes 48 seconds.

===== TRAINING 5-stage =====
<BEGIN
POS count : consumed   100 : 100
NEG count : acceptanceRatio    100 : 0.00209039
Precalculation time: 3
+----+----------+----------+
|  N |    HR    |    FA    |
+----+----------+----------+
|   1|         1|      0.24|
+----+----------+----------+
END>
Training until now has taken 0 days 0 hours 0 minutes 56 seconds.

===== TRAINING 6-stage =====
<BEGIN
POS count : consumed   100 : 100
NEG count : acceptanceRatio    0 : 0
Required leaf false alarm rate achieved. Branch training terminated.
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $
```

結束在第六個 stage

目錄結構

- $ tree

```
├── data
│   ├── cascade.xml
│   ├── params.xml
│   ├── stage0.xml
│   └── ...
├── neg
│   ├── BioID_0000.jpg
│   ├── ...
│   └── BioID_1520.jpg
├── neg.txt
├── pos.vec
├── pos
│   ├── 0001_0015_0016_0089_0089.jpg
│   ├── ...
│   └── pos.txt
└── sample.jpg
```

訓練完成的結果
cascade.xml 是特徵模型

DEMO
camera_haar_detect.py

```
$ cd ~/face_detect/opecv_demo
$ python3 camera_haar_detect.py data/cascade.xml
```

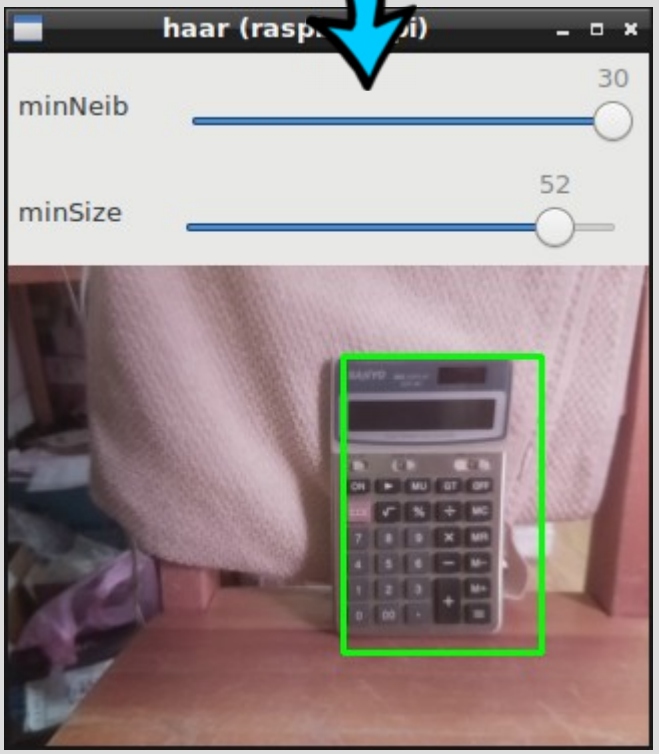
執行結果

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ python3 camera_haar_detec
t.py data/cascade.xml
[ INFO:0] Initialize OpenCL runtime...
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
Found 0 faces!
```

調整 minNeighbors 和 minSize



視覺化 haar 分類器比對過程

- 使用 opencv_visualisation 將 haar 分類器比對過程轉成圖檔
- 需要先將正樣本轉成 24x24 灰階格式
- 產生的圖會在 data 目錄下

按 'q'(兩次) 存檔離開

DEMO

resize.py

```
$ cd ~/face_detect/opecv_demo
$ python3 resize.py sample.jpg
```

執行結果

pi@raspberrypi: /tmp/cam-py-cv/day2/04-face_datection

File Edit Tabs Help

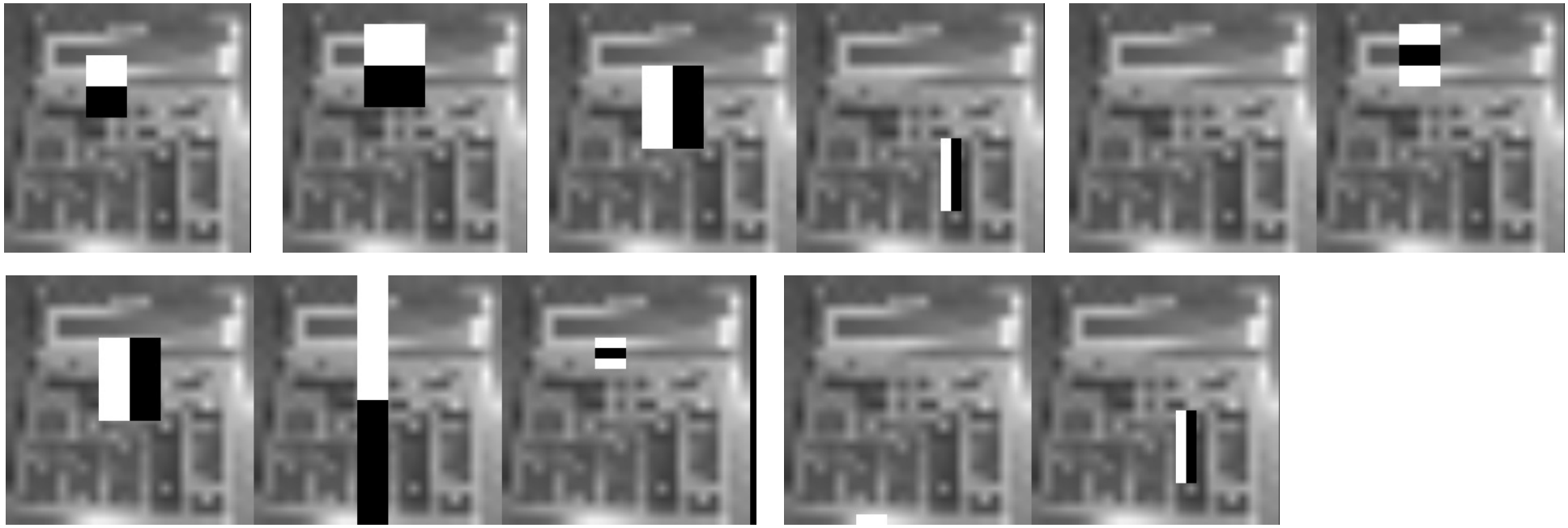
```
pi@raspberrypi:/tmp/cam-py-cv/day2/04-face_datection $ python3 resize.py sample.
jpg
```

出現圖檔以後按 'q'

24x24 灰階圖檔 (出現後再按 'q')

視覺化 haar 分類器比對過程

- `$ opencv_visualisation -d=data/result -i=resized.jpg -m=data/cascade.xml`
- 產生的圖會在 data 目錄下



實驗二：人臉偵測(2)

目的：瞭解 Dlib 中機器學習方法

Dlib

- Dlib 是一套包含網路處理，使用者圖形界面，資料結構，線性代數，機器學習，影像處理，資料探勘，XML 和文本解析等功能的函式庫
- 使用 C++ 撰寫的跨平台函式庫，由 Davis King 在 2002 年開發維護
- 廣泛應用在工業及學術界，也用在嵌入式系統和大型運算架構中
- 提供 C++ 和 Python API
- Boost 授權，但資料集需商業授權

http://dlib.net/

一切都從邊緣偵測開始

- 如何找出灰階有劇烈變化的邊界？

http://www.likecool.com/Winter_in_Krak_w_photographed_by_Marcin_Ryczek--Pic--Gear.html

灰階變化 = 梯度 (gradient)

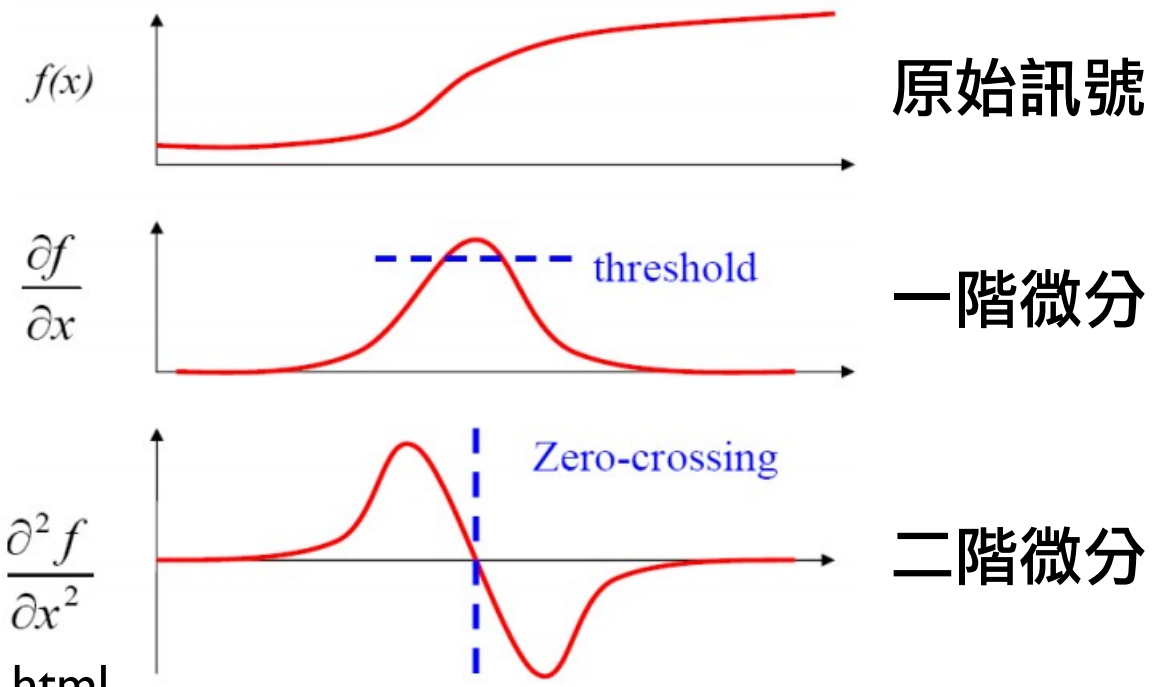
- 梯度可從一階微分 (derivative) 和二階微分求得

(a) Step edge

(b) Ramp

(c) Roof edge

(d) Real edge



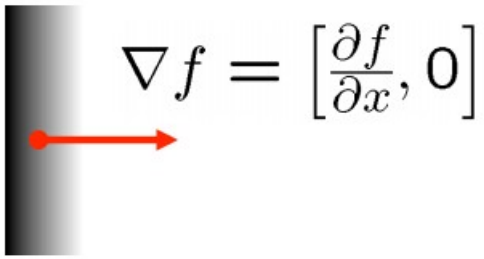
https://miac.unibas.ch/SIP/07-Segmentation.html

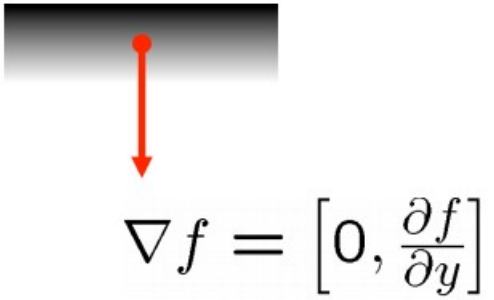
離散資料使用有限差分計算梯度

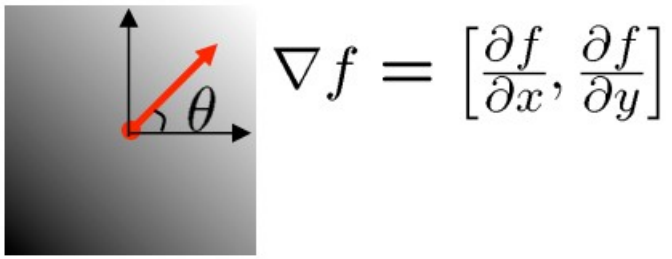
- 梯度包含強度(magnitude), 方向(direction)
- 有限差分(FDM)對X方向的梯度(gradient)計算

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y)-f(x,y)}{1}$$

- 不同方向的梯度計算





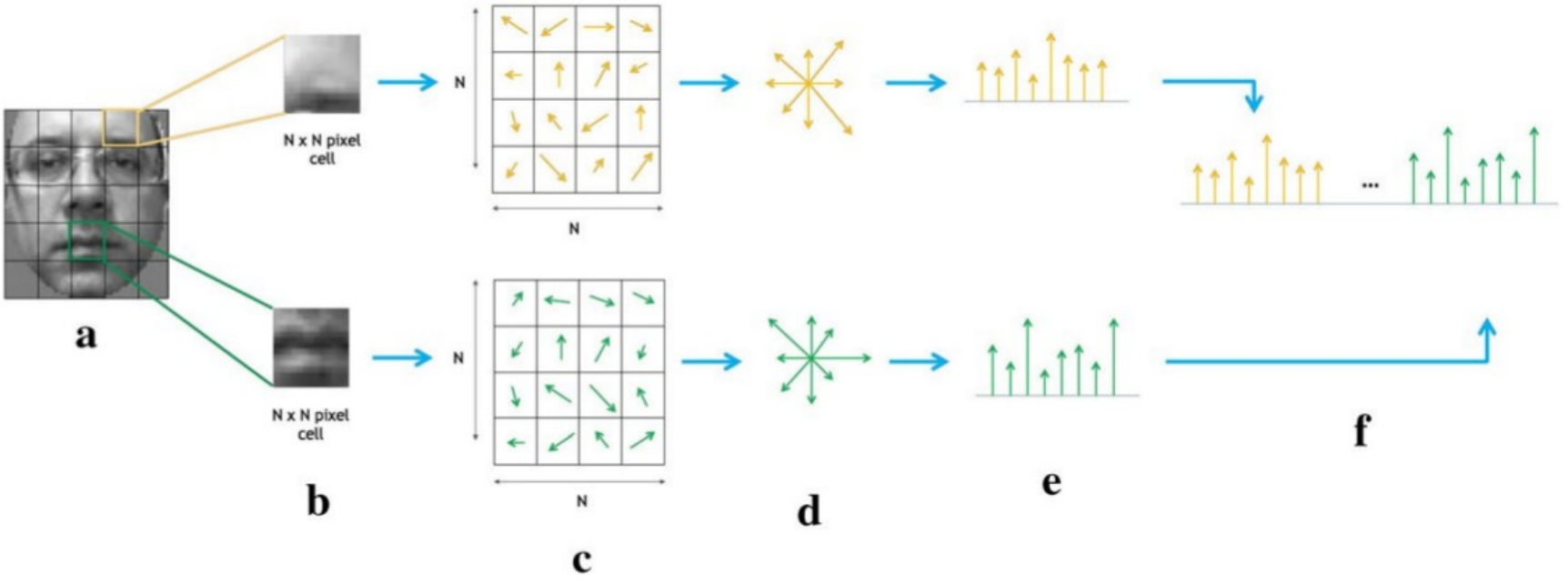


https://www.saltycrane.com/blog/2008/04/how-to-use-pythons-enumerate-and-zip-to/

HOG

- Histogram of Oriented Gradient

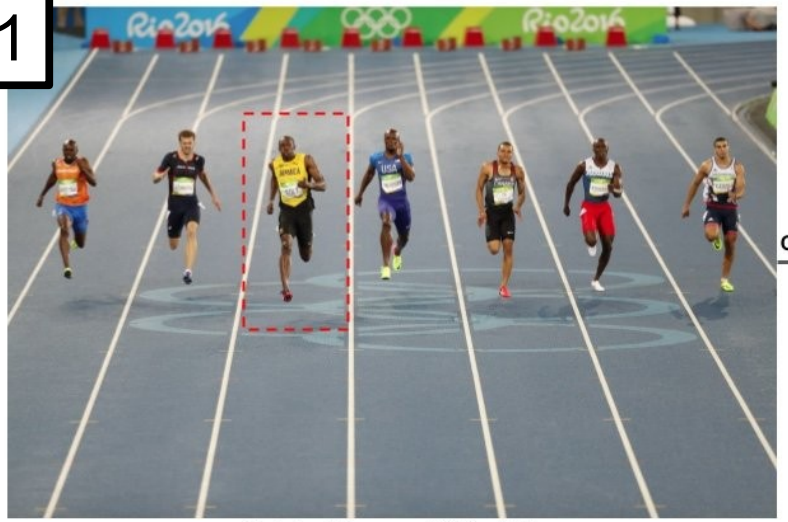
- 方向梯度直方圖，用來描述圖形的特徵
- 計算檢測視窗的梯度強度和方向的統計值當作特徵
- 常搭配 SVM 做物件辨識



https://learnopencv.com/histogram-of-oriented-gradients/

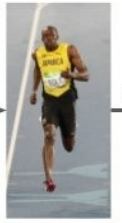
用 HOG 描述圖形特徵

1



Original Image : 720 x 475

Crop



100 x 200

Resize

64 x 128

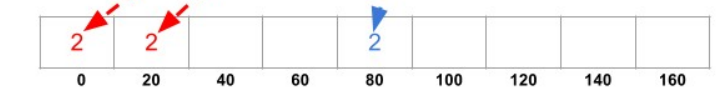
3

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

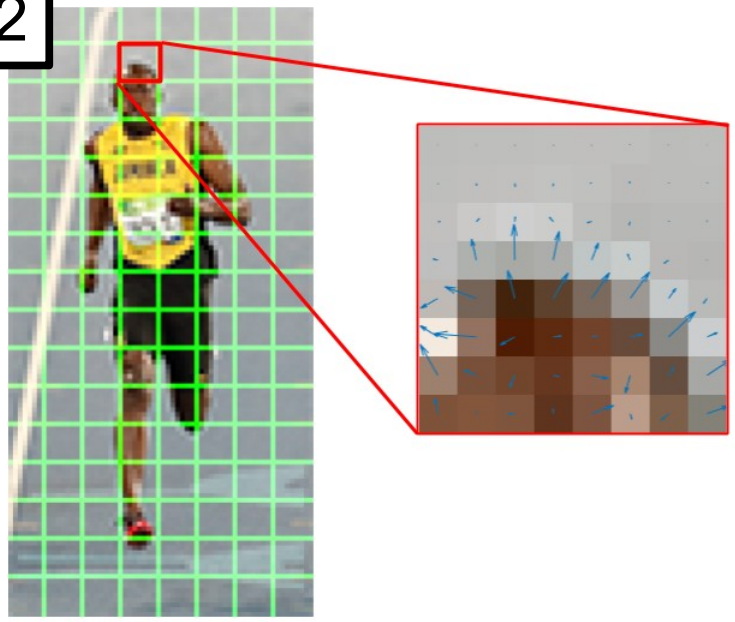
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude



Histogram of Gradients

2



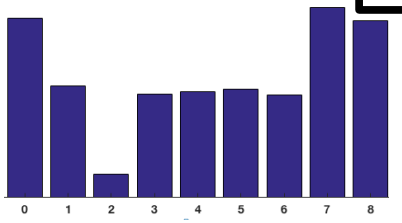
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

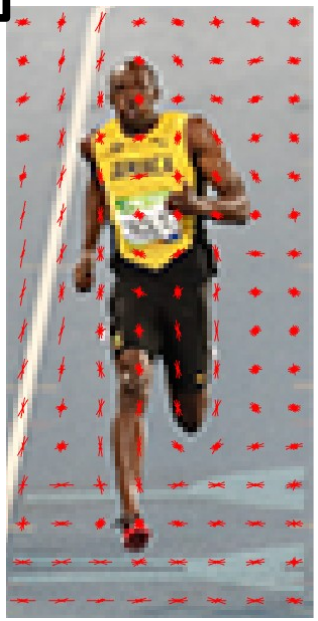
Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

4





https://learnopencv.com/histogram-of-oriented-gradients/

畫出圖片的 HoG

```
import matplotlib.pyplot as plt
from skimage.feature import hog

fd, hog_image = hog(image, orientations=8, pixels_per_cell=(16, 16),
                    cells_per_block=(1, 1), visualize=True, multichannel=True)

fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(8, 4), sharex=True, sharey=True)

ax1.axis('off')
ax1.imshow(image, cmap=plt.cm.gray)
hog_image_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

ax2.axis('off')
ax2.imshow(hog_image_rescaled, cmap=plt.cm.gray)
plt.show()
```

https://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html

DEMO
plot_hog.py

```
$ cd ~/face_detect/dlib_demo
$ python3 plot_hog.py
```

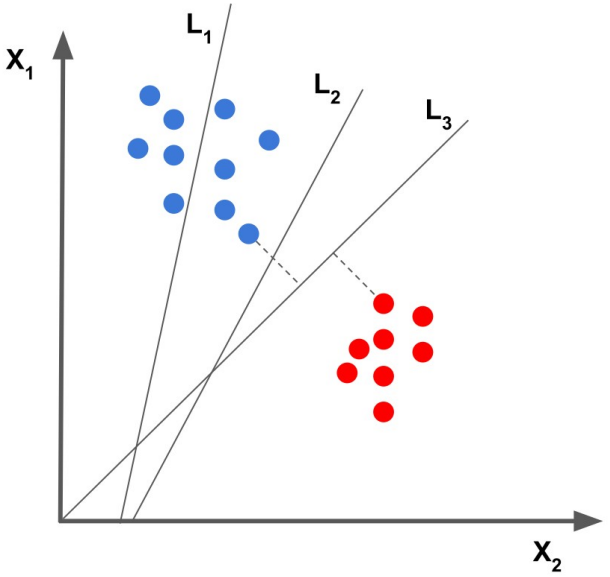
Figure 1

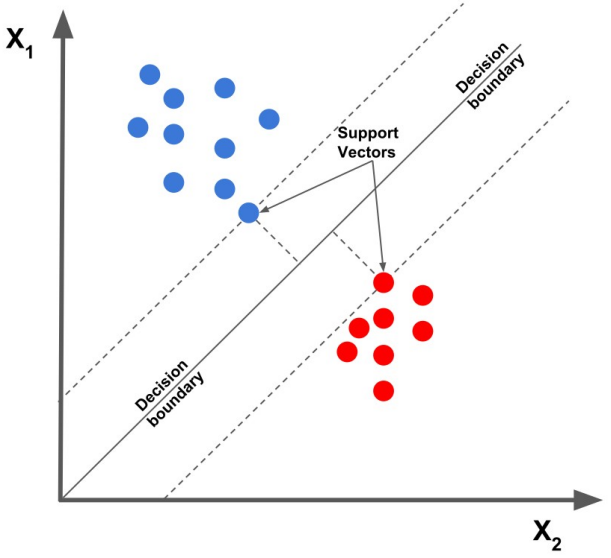
Input image

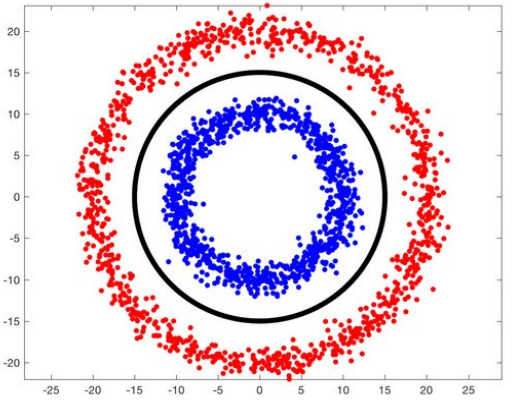
Histogram of Oriented Gradients

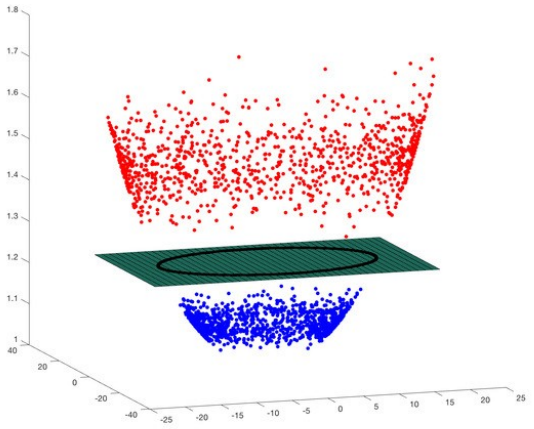
SVM
- Support Vector Machines

- 支援向量機是一種監督式學習
- 使用統計風險最小化的原則來估計一個分類的超平面 (hyperplane)





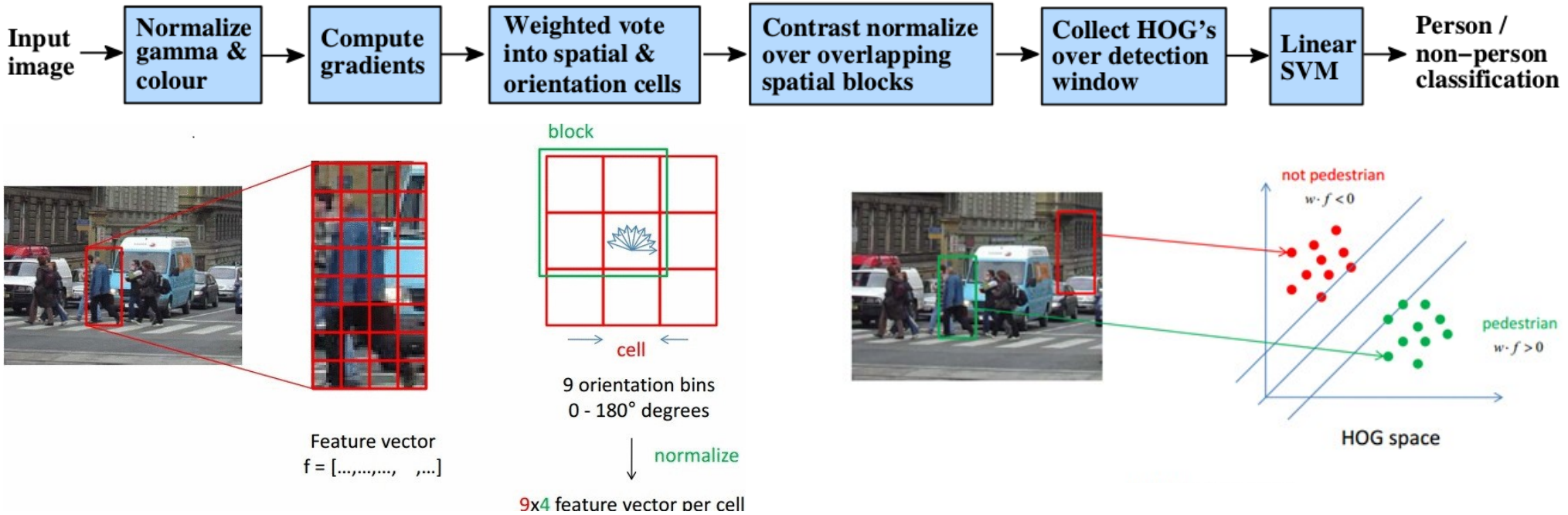


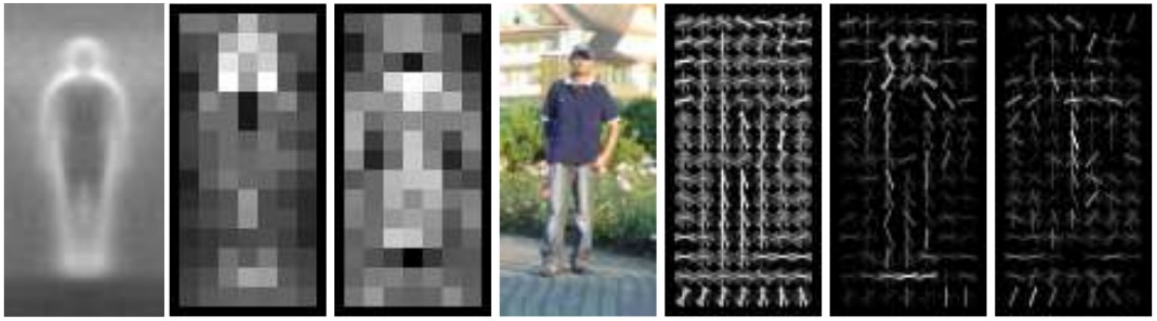


https://learnopencv.com/support-vector-machines-svm/

以 HOG + SVM 做行人檢測

- 源自 CVPR05 論文





https://hal.inria.fr/file/index/docid/548512/filename/hog_cvpr2005.pdf

載入圖檔並辨識

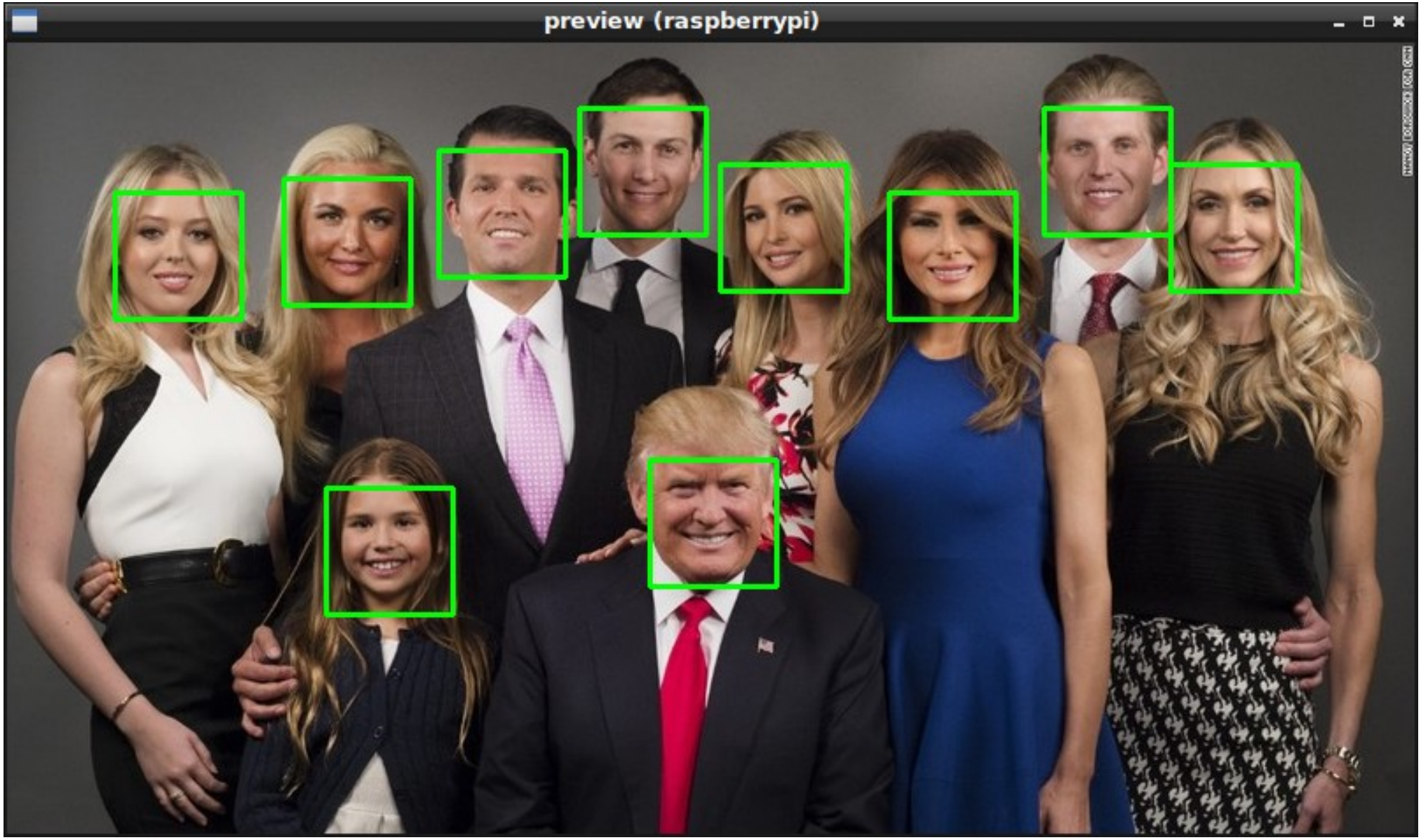
```
image = cv2.imread(sys.argv[1])
detector = dlib.get_frontal_face_detector()
faces = detector(image, 0)



for face in faces:
    x1 = face.left() # left point
    y1 = face.top() # top point
    x2 = face.right() # right point
    y2 = face.bottom() # bottom point

    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

上採樣的次數，設為 1 可偵測更多人臉

http://dlib.net/dlib/image_processing/frontal_face_detector.h.html

DEMO
image_face_detect.py

```
$ cd ~/face_detect/dlib_demo
$ python3 image_face_detect.py
```



https://edition.cnn.com/2016/07/18/opinions/trump-family-production-dantonio/index.html

使用 detector.run() 觀察人臉資訊

```
image = cv2.imread(sys.argv[1])
detector = dlib.get_frontal_face_detector()
faces, scores, idx = detector.run(image, 0, 0)
```

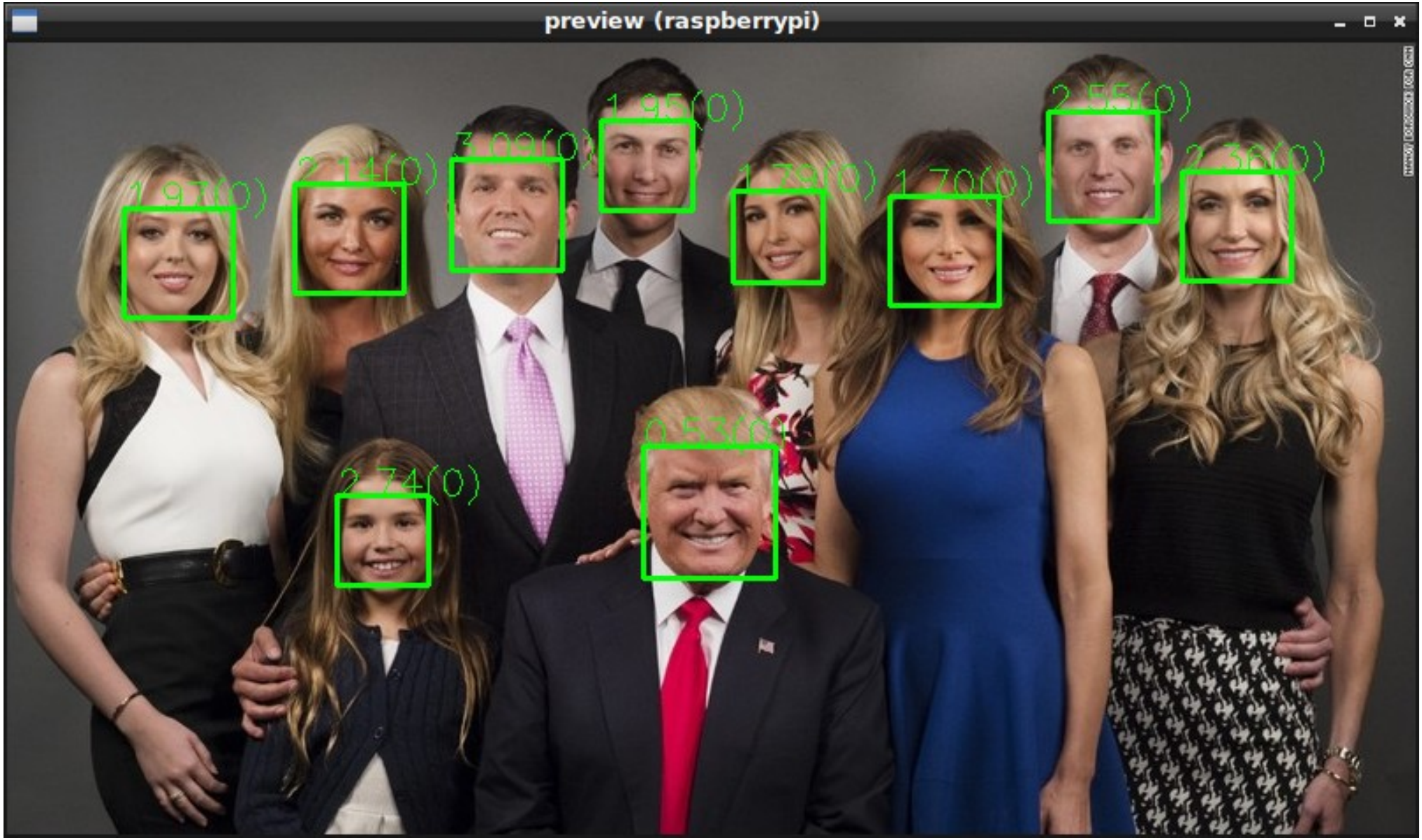
上採樣的次數　　分數門檻值

```
for i, face in enumerate(faces):
    x1 = face.left() # left point
    y1 = face.top() # top point
    x2 = face.right() # right point
    y2 = face.bottom() # bottom point

    text = "%2.2f(%d)" % (scores[i], idx[i])
    cv2.rectangle(image, (x1, y1), (x2, y2), (0, 255, 0), 2)
    cv2.putText(image, text, (x1, y1), cv2.FONT_HERSHEY_SIMPLEX,
          0.7, (0, 255, 0), 1)
```

http://dlib.net/dlib/image_processing/frontal_face_detector.h.html

DEMO
image_face_detect_run.py

```
$ cd ~/face_detect/dlib_demo
$ python3 image_face_detect_run.py
```


preview (raspberrypi)
1.97(0)
2.14(0)
3.09(0)
1.95(0)
1.79(0)
1.70(0)
2.55(0)
2.36(0)
2.74(0)
0.53(0)

可調整的參數

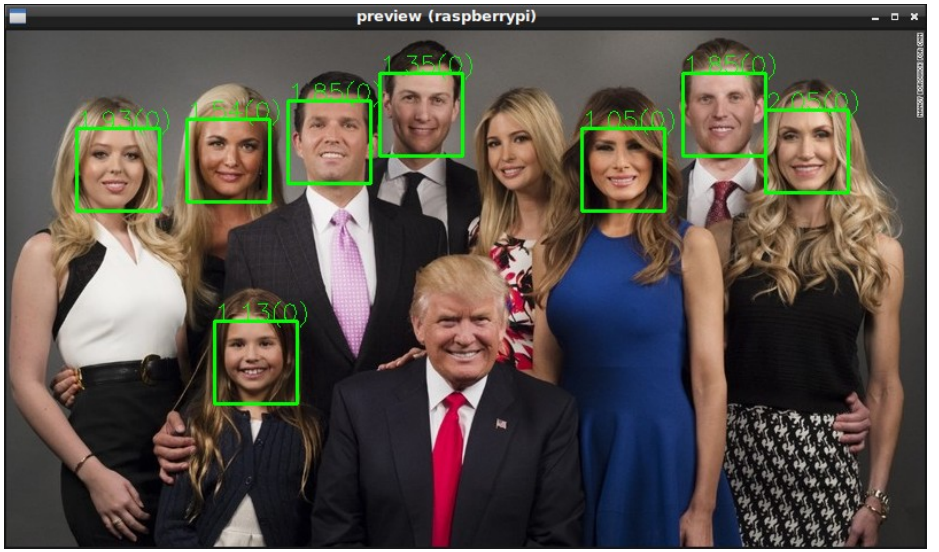
- `upsample`: 上採樣的次數 (通常 =0)
- `threshold`: 人臉門檻值 (通常 >0.12)

https://github.com/davisking/dlib/blob/master/python_examples/face_detector.py

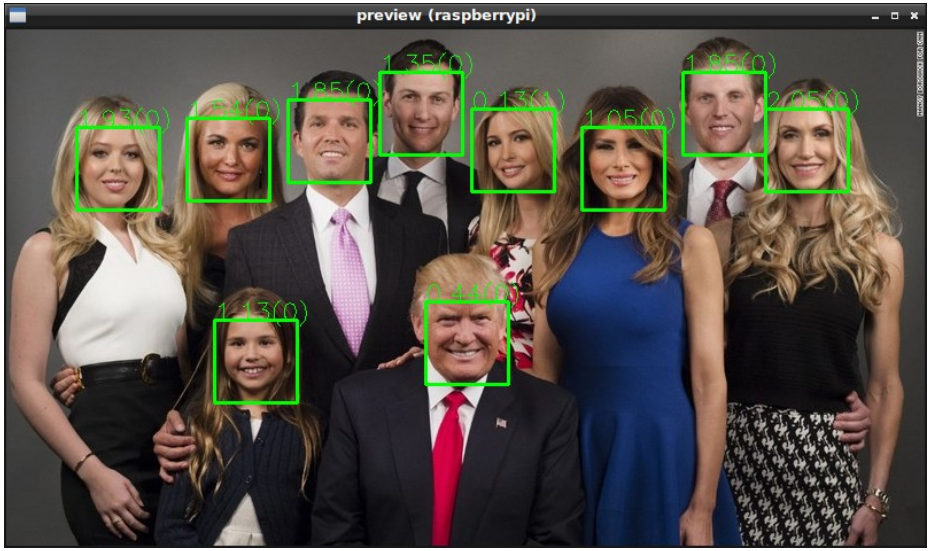
threshold

- upsample = 0

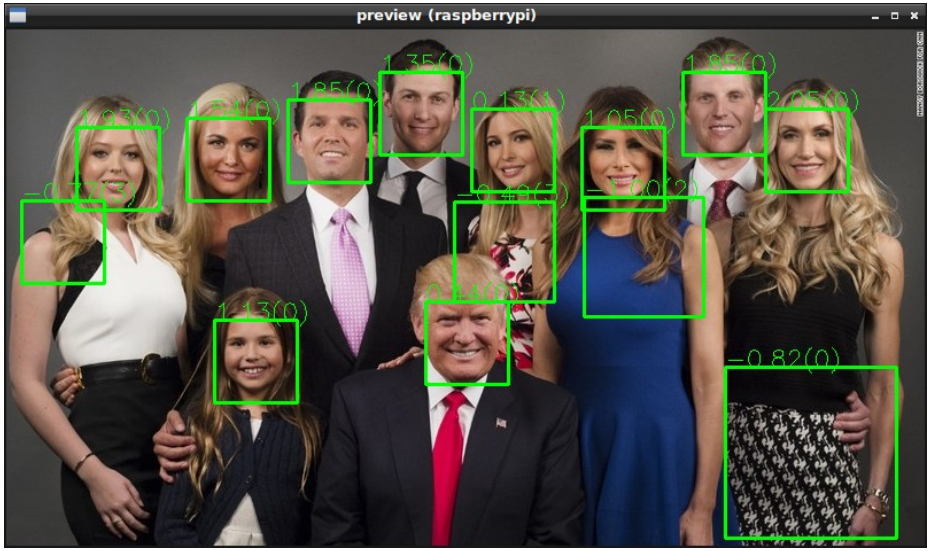
threshold =2



threshold =1

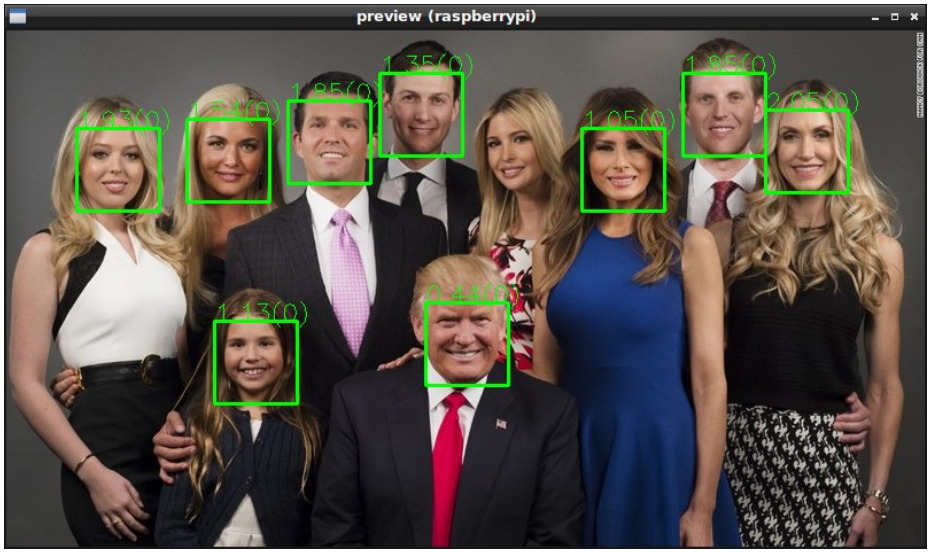


threshold =0

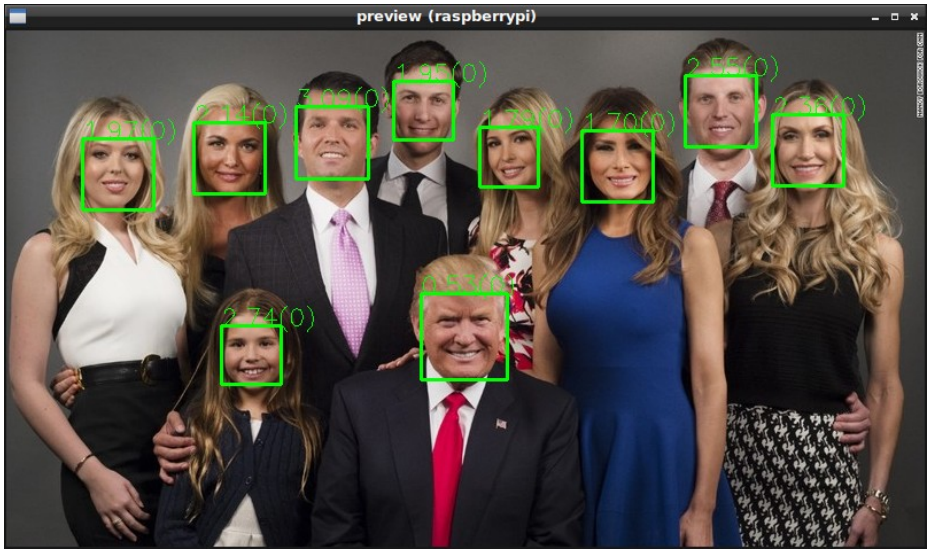


threshold =-1

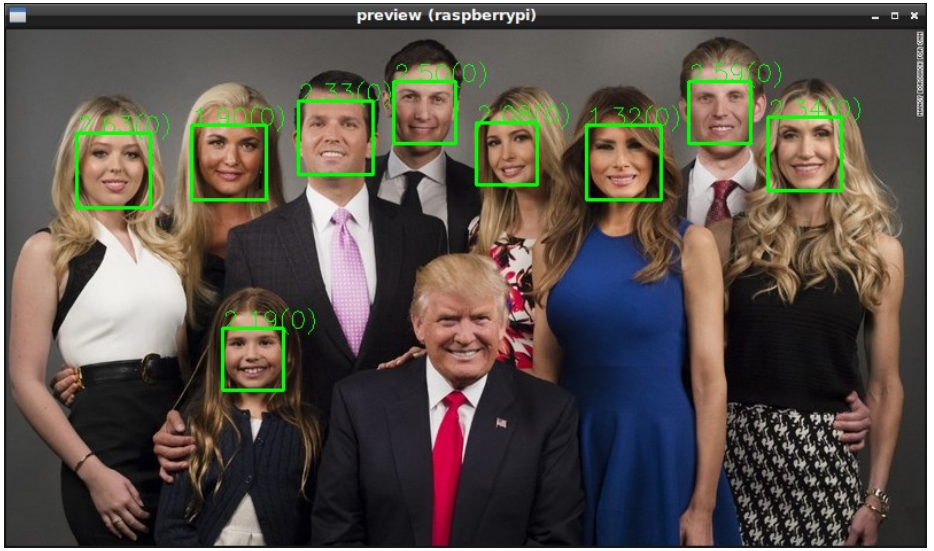
upsample
- threshold= 0.2



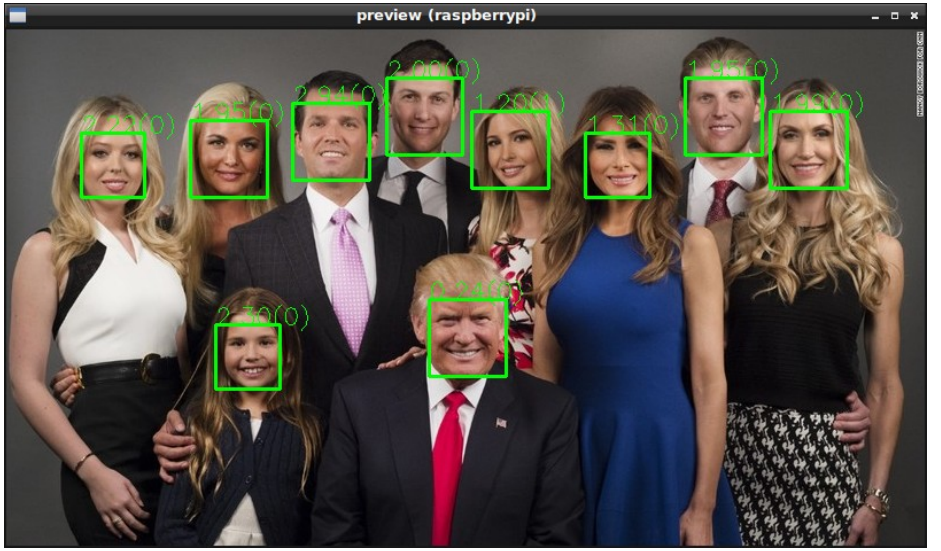
upsample =3



upsample =2



upsample =1



upsample =0

讀取 Camera 並辨識

```
cap = cv2.VideoCapture(0)
detector = dlib.get_frontal_face_detector()

while True:
    ret, frame = cap.read()
    faces, scores, idx = detector.run(frame, 0, 0)

    for i, face in enumerate(faces):
        x1 = face.left()
        y1 = face.top()
        x2 = face.right()
        y2 = face.bottom()

        text = "%2.2f(%d)" % (scores[i], idx[i])
        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
        cv2.putText(frame, text, ...)
```

DEMO
camera_face_detect.py

```
$ cd ~/face_detect/dlib_demo
$ python3 camera_face_detect.py
```

使用預先訓練好的 HoG Filters

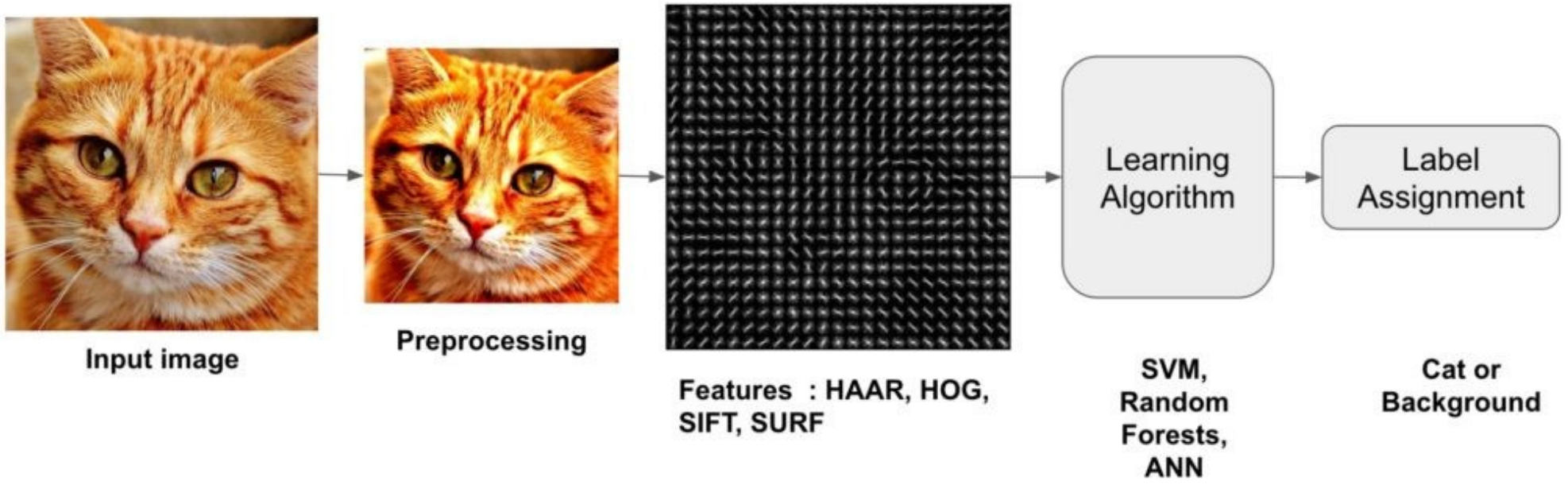
- dlib.get_frontal_face_detector()
  - 5個HoG filters
    - front looking(0)
    - left looking(1)
    - right looking(2)
    - front looking but rotated left(3)
    - front looking but rotated right(4)
  - 已內嵌在frontal_face_detector.h

- 資料集下載：
  - https://github.com/davisking/dlib-models

https://github.com/davisking/dlib/blob/master/dlib/image_processing/frontal_face_detector.h

訓練自己的 HoG Filters

操作流程



https://learnopencv.com/training-a-custom-object-detector-with-dlib-making-gesture-controlled-applications

Hand Images Databases

- Mutah 大學提供的免費，開源的手掌影像資料庫
- 為真實世界資料，有多種照明 / 背景 / 人臉大小
- 200 位受試者，3000 張 RGB 影像 (TIFF 格式)
- 資料庫以男女，年齡，拍攝裝置取名
- 提供兩種解析度照片
  - MOHI 高解析度照片 (3264x2448)
  - WEHI 低解析度照片 (640x480)

https://www.mutah.edu.jo/biometrix/hand-images-databases.aspx

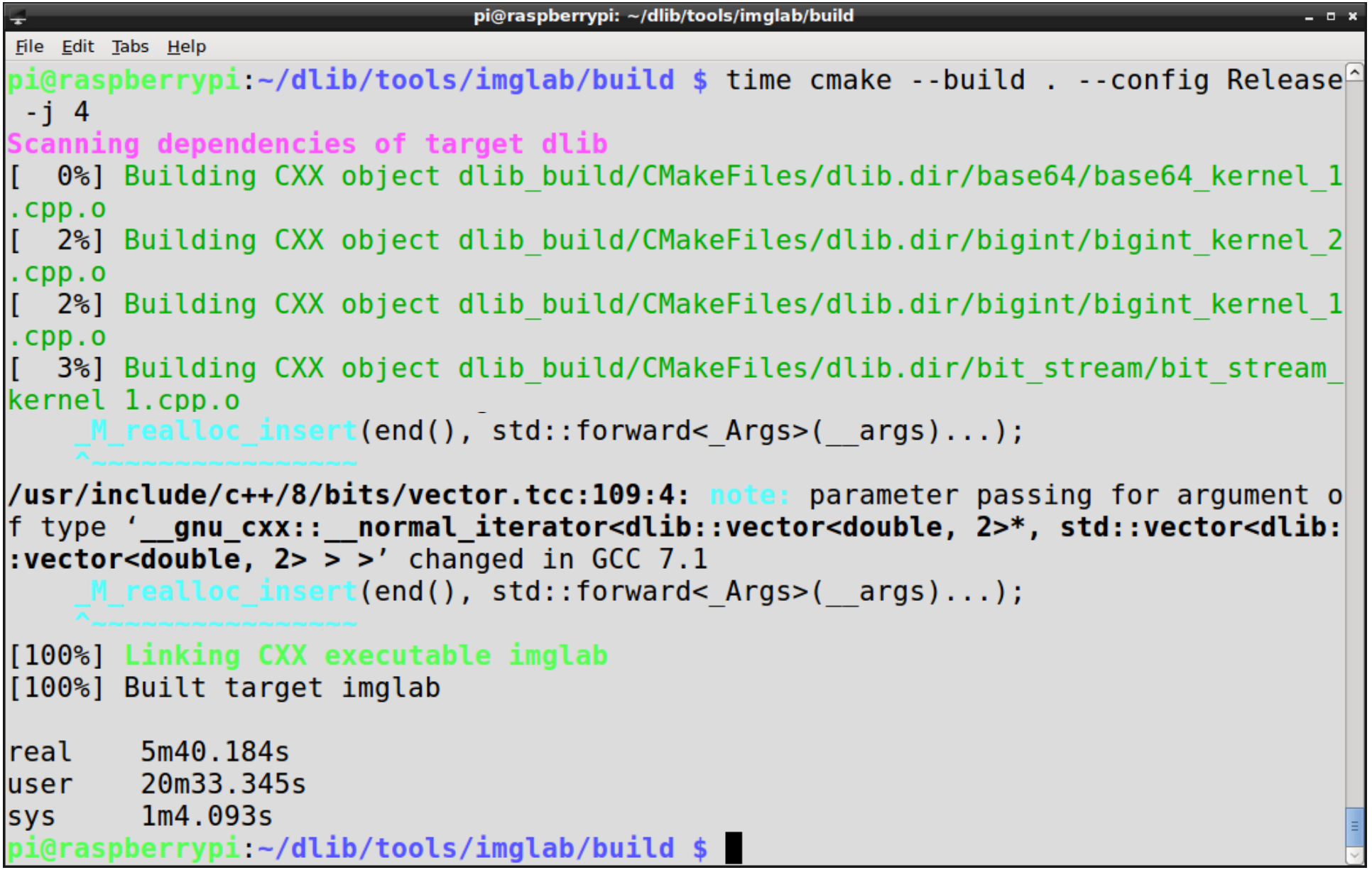
解壓縮 WEHI 影像資料庫

- `$ cd ~/face_detect/dlib_demo`
- `$ tar zxvf ~/WEHI-S1-P1-50.tar.gz`

編譯 imglab 標注工具

- `$ git clone http://github.com/davisking/dlib.git`
- `$ cd dlib/tools/imglab`
- `$ mkdir build`
- `$ cd build`
- `$ time cmake ..`
- `$ time cmake --build . --config Release -j 4`

編譯 imglab 標注工具



使用 imglab 工具

- $ cp ~/dlib/tools/imglab/build/imglab ~/face_detect/dlib_demo

- $ cd ~/face_detect/dlib_demo

  # 根據影像資料庫來建立 hand-dataset.xml 和 testing.xml
- $ ./imglab -c ./WEHI-S1-P1-50/hand-dataset.xml ./WEHI-S1-P1-50
- $ ./imglab -c ./WEHI-S1-P1-50/testing.xml ./WEHI-S1-P1-50

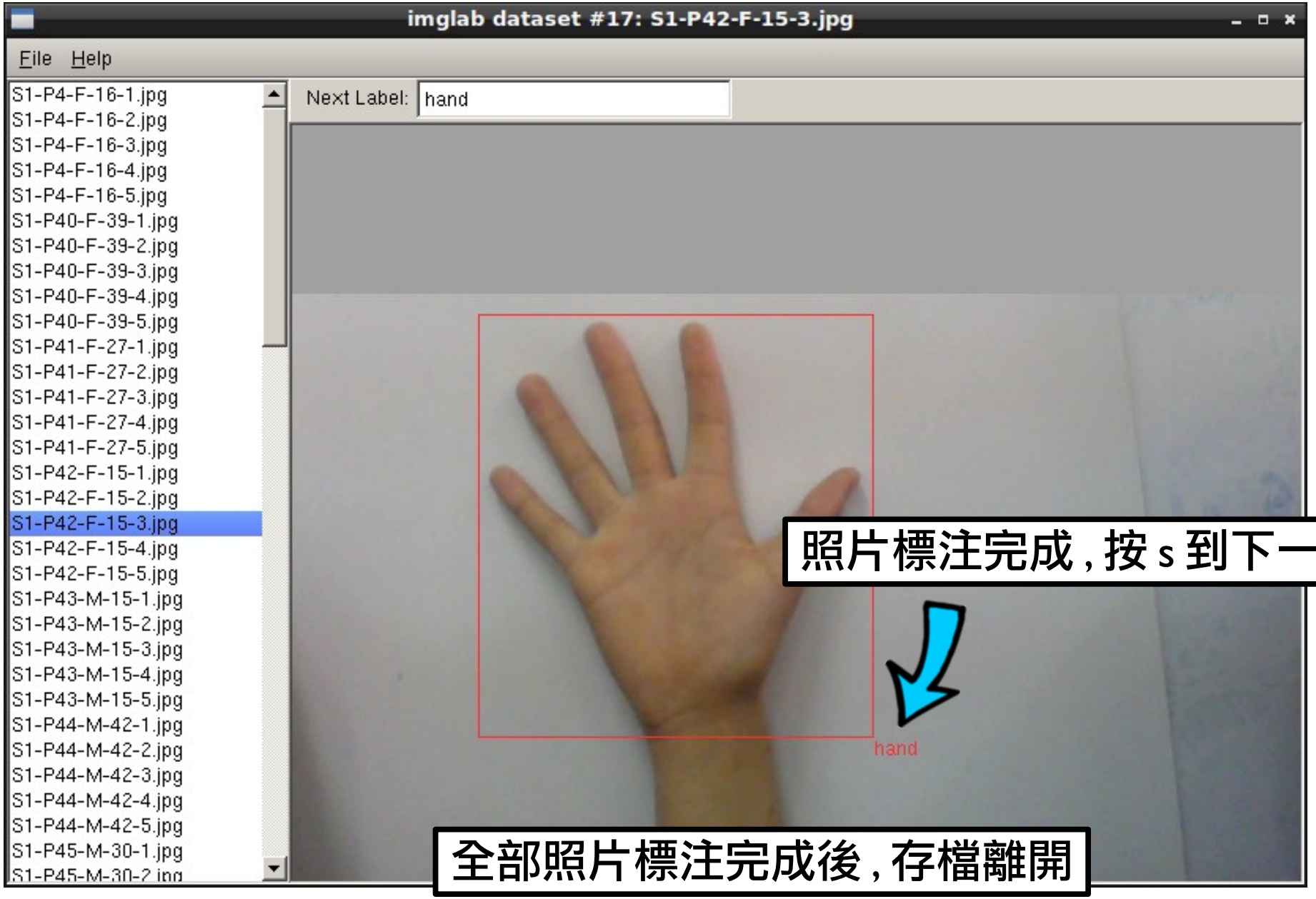
  # 標注影像資料庫並將結果儲存到 hand-dataset.xml(使用 80% 做訓練)
- $ ./imglab ./WEHI-S1-P1-50/hand-dataset.xml

  # 標注影像資料庫並將結果儲存到 testing.xml(使用 20% 做測試)
- $ ./imglab ./WEHI-S1-P1-50/testing.xml

使用 imglab 工具

1. 先建立 label

使用 imglab 工具


imglab dataset #17: S1-P42-F-15-3.jpg
File Help
Next Label: hand
S1-P4-F-16-1.jpg
S1-P4-F-16-2.jpg
S1-P4-F-16-3.jpg
S1-P4-F-16-4.jpg
S1-P4-F-16-5.jpg
S1-P40-F-39-1.jpg
S1-P40-F-39-2.jpg
S1-P40-F-39-3.jpg
S1-P40-F-39-4.jpg
S1-P40-F-39-5.jpg
S1-P41-F-27-1.jpg
S1-P41-F-27-2.jpg
S1-P41-F-27-3.jpg
S1-P41-F-27-4.jpg
S1-P41-F-27-5.jpg
S1-P42-F-15-1.jpg
S1-P42-F-15-2.jpg
S1-P42-F-15-3.jpg
S1-P42-F-15-4.jpg
S1-P42-F-15-5.jpg
S1-P43-M-15-1.jpg
S1-P43-M-15-2.jpg
S1-P43-M-15-3.jpg
S1-P43-M-15-4.jpg
S1-P43-M-15-5.jpg
S1-P44-M-42-1.jpg
S1-P44-M-42-2.jpg
S1-P44-M-42-3.jpg
S1-P44-M-42-4.jpg
S1-P44-M-42-5.jpg
S1-P45-M-30-1.jpg
S1-P45-M-30-2.jpg
hand
照片標注完成，按 s 到下一張
全部照片標注完成後，存檔離開

檢查 hand-dataset.xml 和 testing.xml

- 如果圖片是手，一定要標注

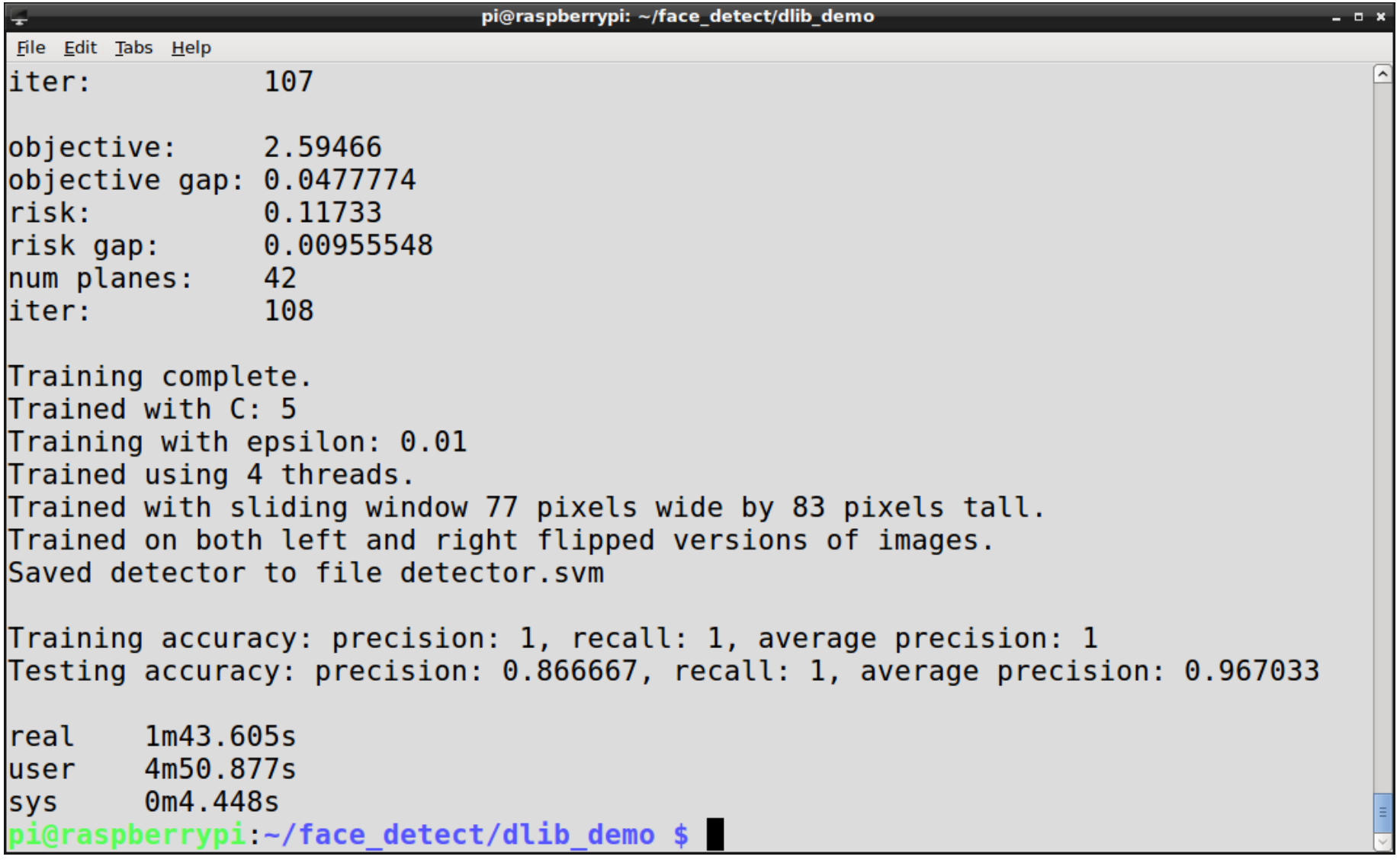
pi@raspberrypi: ~/face_detect/dlib_demo/WEHI-S1-P1-50

File Edit Tabs Help

```
 1 <?xml version='1.0' encoding='ISO-8859-1'?>
 2 <?xml-stylesheet type='text/xsl' href='image_metadata_stylesheet.xsl'?>
 3 <dataset>
 4 <name>imglab dataset</name>
 5 <comment>Created by imglab tool.</comment>
 6 <images>
 7   <image file='S1-P4-F-16-1.jpg'>
 8     <box top='173' left='137' width='237' height='266'>
 9       <label>hand</label>
10     </box>
11   </image>
12   <image file='S1-P4-F-16-2.jpg'>
13     <box top='174' left='135' width='241' height='278'>
14       <label>hand</label>
15     </box>
16   </image>
17   <image file='S1-P4-F-16-3.jpg'>
18     <box top='179' left='125' width='239' height='256'>
19       <label>hand</label>
20     </box>
21   </image>
```

根據影像和標注檔案使用 SVM 訓練

- $ time python3 train_object_detector.py ./WEHI-S1-P1-50



使用 train_simple_object_detector()

```
import dlib

options = dlib.simple_object_detector_training_options()
options.add_left_right_image_flips = True


options.C = 5
options.num_threads = 4

training_xml_path = os.path.join(faces_folder, "hand-dataset.xml")
testing_xml_path = os.path.join(faces_folder, "testing.xml")

dlib.train_simple_object_detector(training_xml_path, "detector.svm",
options)

dlib.test_simple_object_detector(training_xml_path, "detector.svm")
dlib.test_simple_object_detector(testing_xml_path, "detector.svm")
```

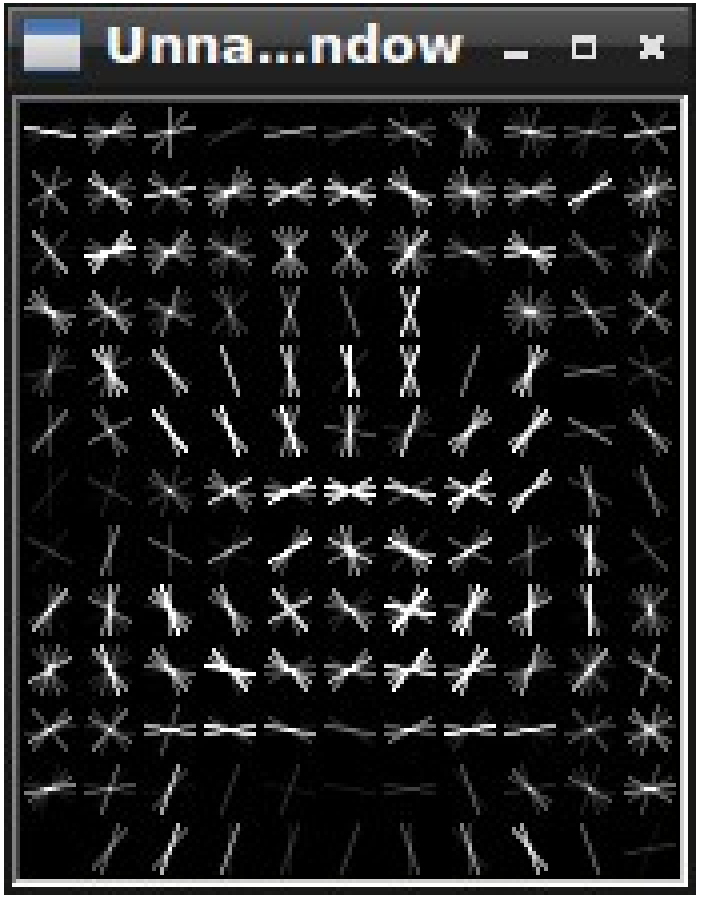
C 是 SVM 懲罰係數 , 表示對誤差容忍度
如果太高可能會 overfitting, 建議 1-5 之間

DEMO
train_object_detector.py

```
$ cd ~/face_detect/dlib_demo
$ python3 train_object_detector.py ./WEHI-S1-P1-50
```

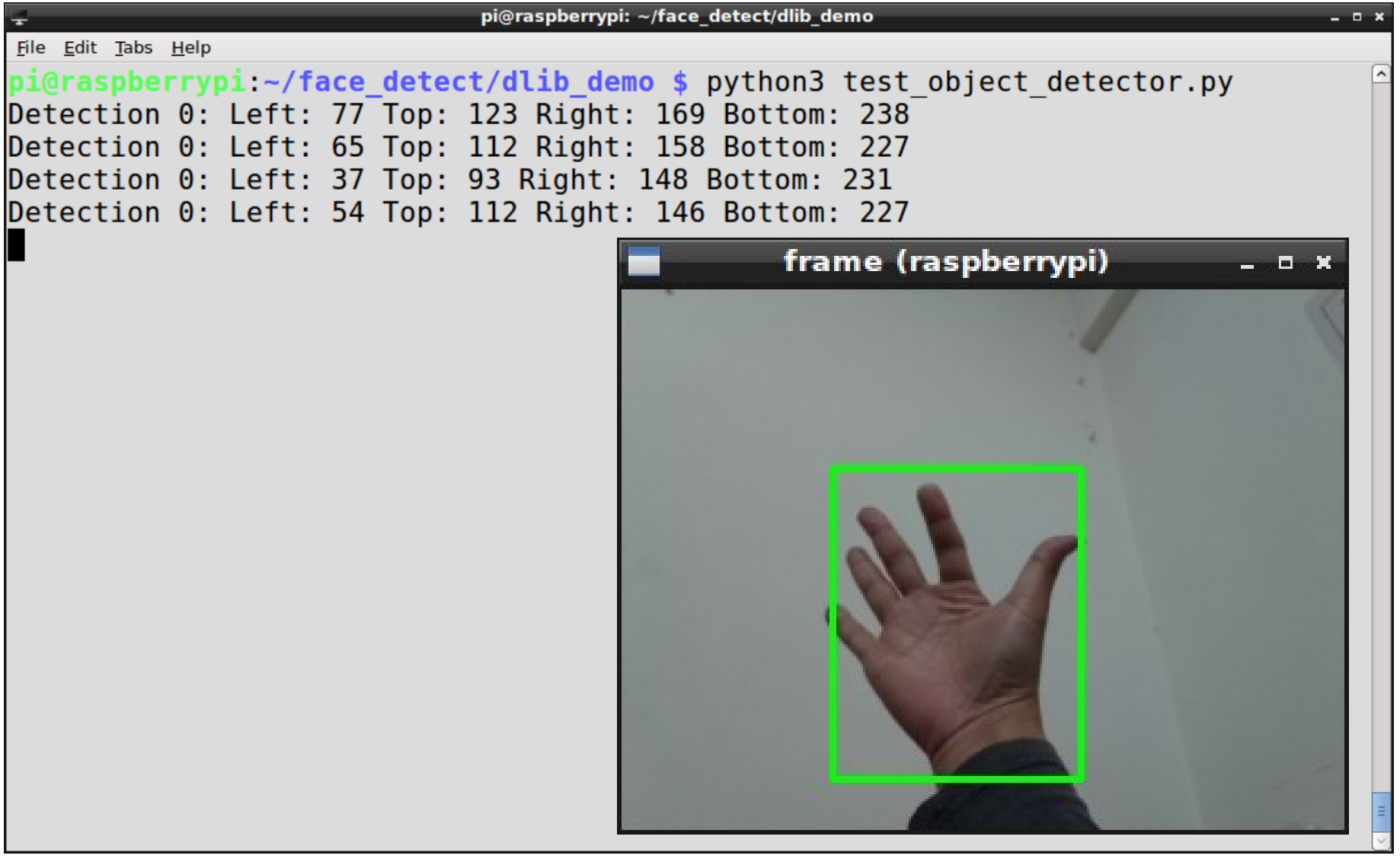
觀察學到的 HoG Fliter 結果

- `$ python3 test_object_detector.py`



測試 detector.svm

- `$ python3 test_object_detector.py`



讀取 Camera 並辨識

```
cap = cv2.VideoCapture(0)
detector = dlib.simple_object_detector("detector.svm")

while True:
    ret, frame = cap.read()
    rects = detector(frame)

    for k, d in enumerate(rects):
        x1 = d.left()
        y1 = d.top()
        x2 = d.right()
        y2 = d.bottom()

        cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 255, 0), 2)
```

DEMO
test_object_detector.py

```
$ cd ~/face_detect/dlib_demo
$ python3 test_object_detector.py
```

小結

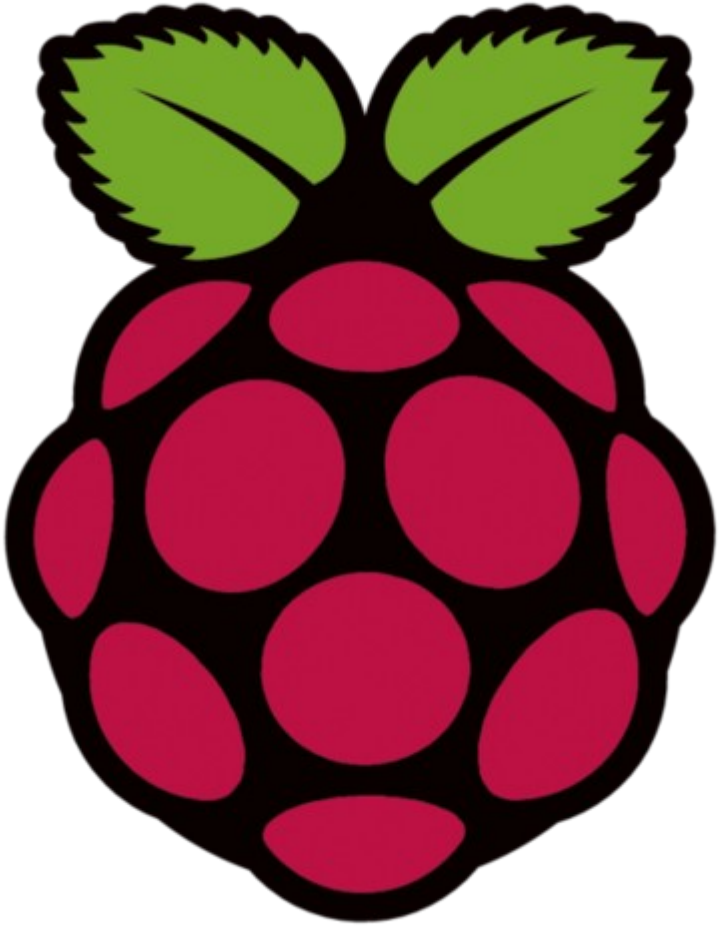
- Haar 和 HoG 都沒有旋轉不變性和尺度不變性，需自行建立訓練資料
- 好的資料內容和標記是影響結果好壞的關鍵
- 對於單純的環境，使用機器學習就有不錯的效果

練習

- 使用 HoG + SVM 訓練杯子辨識器
- 可以用 camera_capture.py 拍照儲存影像 (640x480)

```
$ cd ~/face_detect/dlib_demo
$ python3 camera_capture.py
```

Raspberry Pi Rocks the World



Thanks