Outline

- 嵌入式應用: 網路攝影機
  - 觀看Raspberry Pi Camera的圖片
  - 影像辨識 (opencv)
    - 圖片旋轉, 裁切, 縮放
    - 人臉識別

觀看Raspberry Pi Camera的圖片

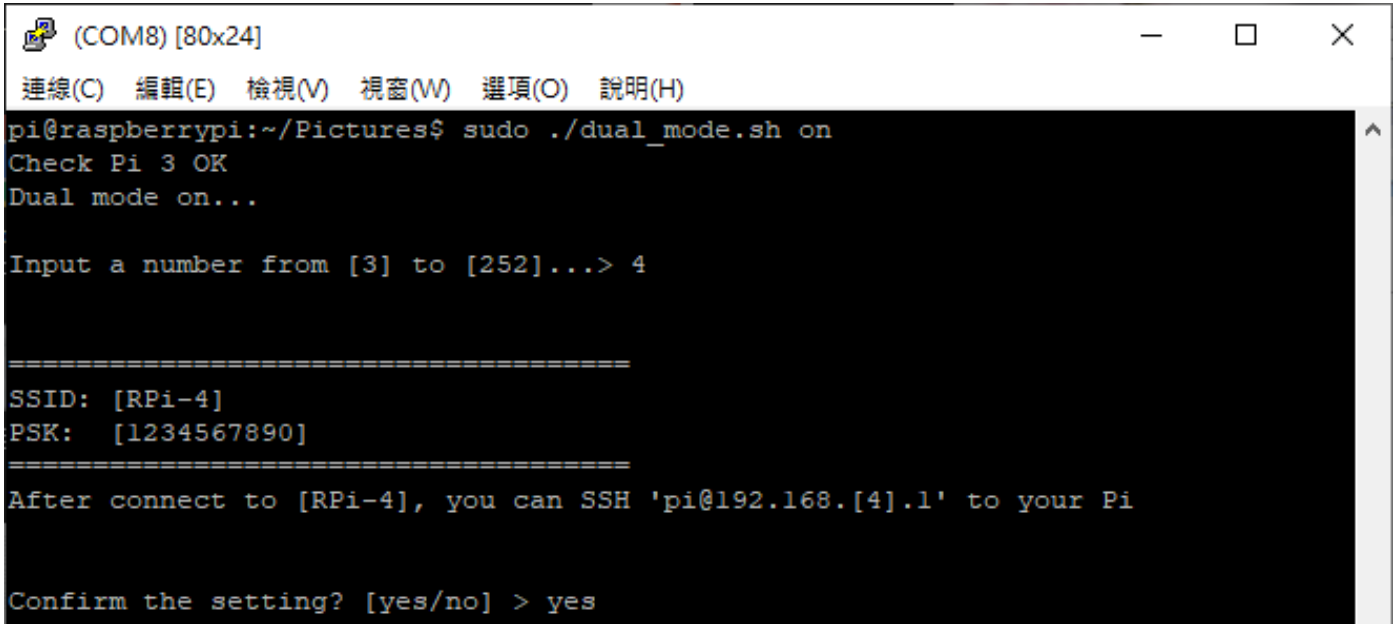
- python -m SimpleHTTPServer 8000
- winscp
- vnc

- Create Wi-Fi hotspot on PI
- View image in terminal

https://github.com/raspberrypi-tw/sh/blob/master/dual_mode.sh

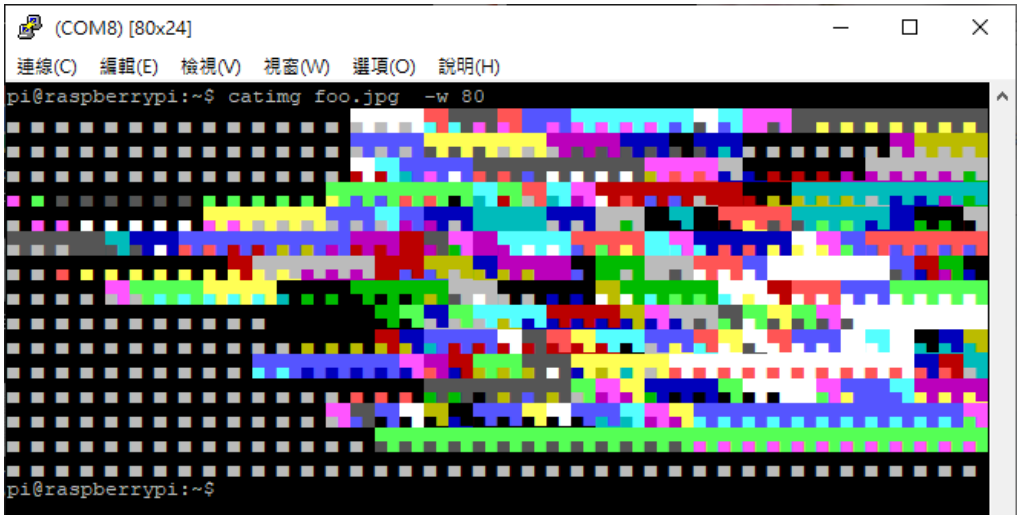
Create Wi-Fi hotspot on Pi

- wget https://raw.githubusercontent.com/raspberrypi-tw/sh/master/dual_mode.sh
- chmod +x dual_mode.sh
- sudo ./dual_mode.sh on # it will install related packages and reboot
- sudo ./dual_mode.sh off # it will reboot



View image in terminal

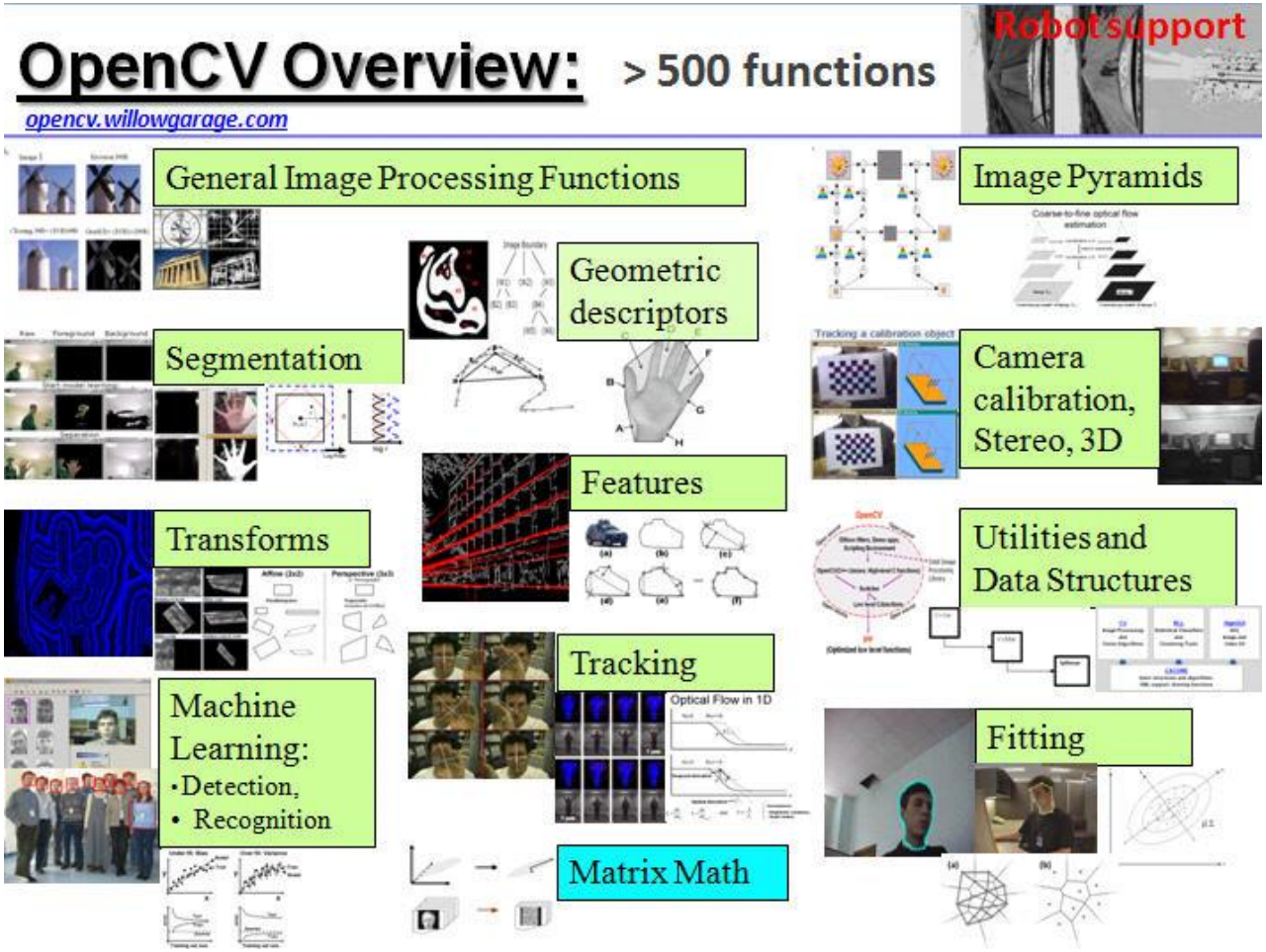
- Insanely fast image printing in your terminal
  - git clone https://github.com/posva/catimg
  - cd catimg/
  - cmake .
  - sudo make install
- Usage: catimg xxxxxxx.jpg -w 80



https://www.embedded-vision.com/technology/computer-vision-algorithms

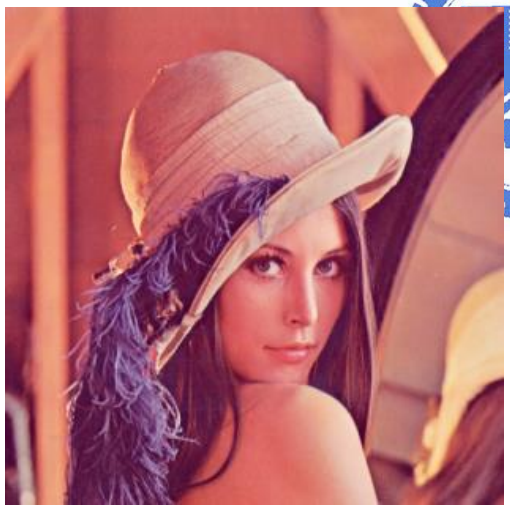
opencv

- Open Source Computer Vision Library



https://docs.opencv.org/2.4.13.7/doc/tutorials/introduction/display_image/display_image.html

preview



- Sample code

```
import cv2
import numpy as np
img = cv2.imread('lena256rgb.jpg')

cv2.imshow('preview', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

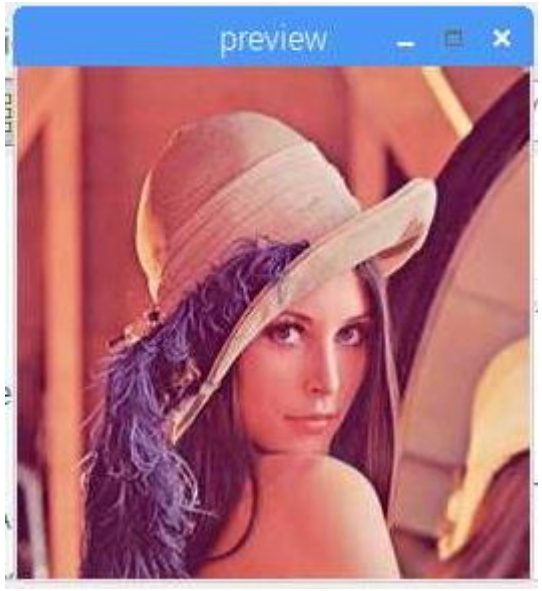
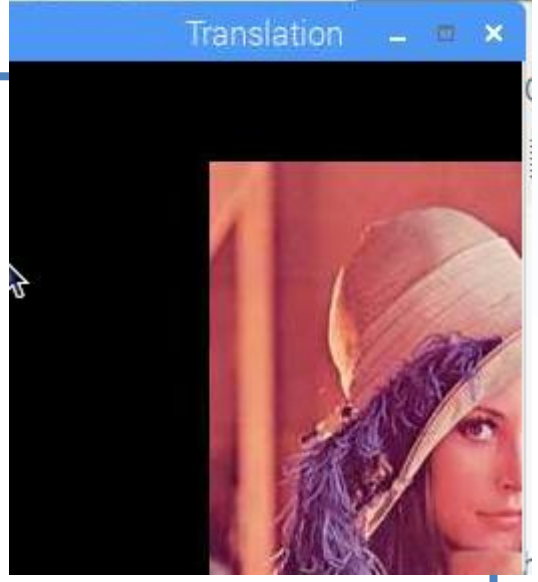


Fig source: https://upload.wikimedia.org/wikipedia/zh/3/34/Lenna.jpg

https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

translation



- Applies an affine transformation to an image.

```
import cv2
import numpy as np
img = cv2.imread('lena256rgb.jpg')
rows, cols = img.shape[:2]
M = np.float32([ [1,0,100], [0,1,50] ])
translation = cv2.warpAffine(img, M, (cols, rows))
cv2.imshow('Translation', translation)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

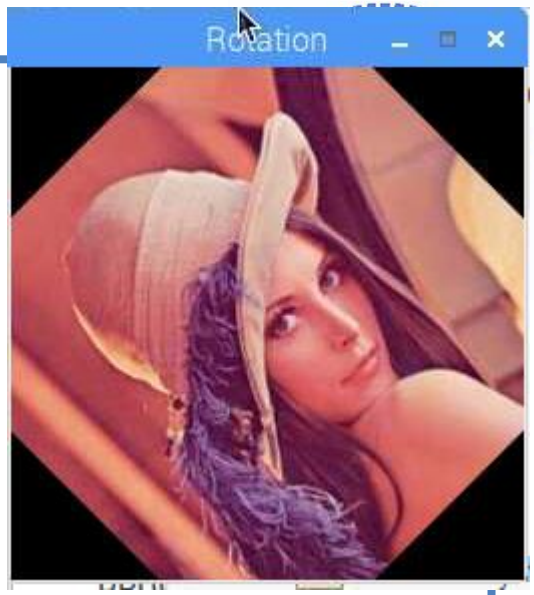
The function warpAffine transforms the source image using the specified matrix:

$$\mathtt{dst}(x,y) = \mathtt{src}(M_{11}x + M_{12}y + M_{13}, M_{21}x + M_{22}y + M_{23})$$

https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

rotation

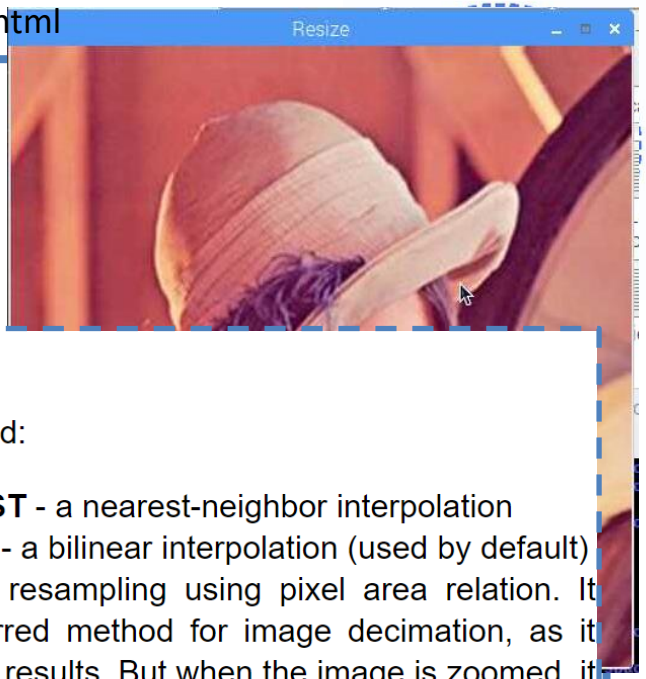
- Calculates an affine matrix of 2D rotation.




```
import cv2
import numpy as np

img = cv2.imread("lena256rgb.jpg")
rows, cols = img.shape[:2]

M = cv2.getRotationMatrix2D((cols/2, rows/2), 45, 1)
rotation = cv2.warpAffine(img, M, (cols, rows))

cv2.imshow('Rotation', rotation)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

https://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

resize



- Resizes an image.

- **interpolation** –

  interpolation method:

  - **INTER_NEAREST** - a nearest-neighbor interpolation
  - **INTER_LINEAR** - a bilinear interpolation (used by default)
  - **INTER_AREA** - resampling using pixel area relation. It may be a preferred method for image decimation, as it gives moire'-free results. But when the image is zoomed, it is similar to the `INTER_NEAREST` method.
  - **INTER_CUBIC** - a bicubic interpolation over 4x4 pixel neighborhood
  - **INTER_LANCZOS4** - a Lanczos interpolation over 8x8 pixel neighborhood

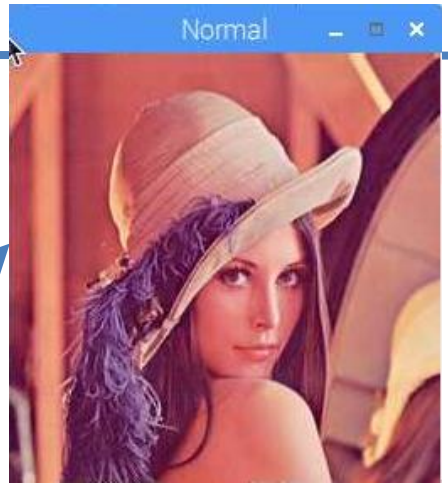
```
import cv2
import numpy as np

img = cv2.imread("lena256rgb.jpg")
rows, cols = img.shape[:2]

resize = cv2.resize(img, (2*rows, 2*cols), interpolation = cv2.INTER_CUBIC)
cv2.imshow('Resize', resize)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

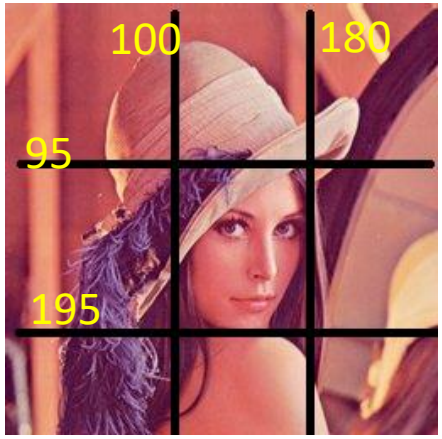
https://docs.opencv.org/2.4/doc/user_guide/ug_mat.html

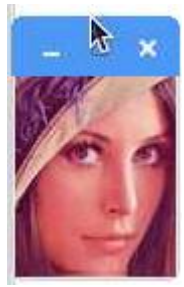
crop

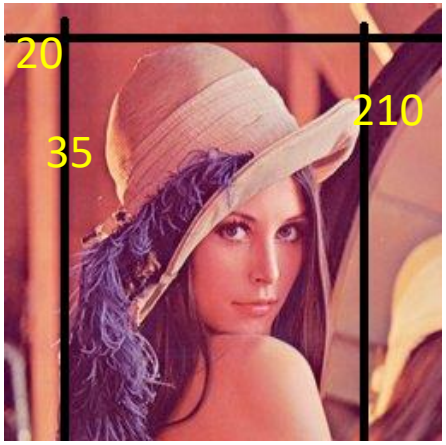
- Sample code

```
import cv2
import numpy as np

img = cv2.imread("lena256rgb.jpg")
cv2.imshow("Normal", img)
cv2.waitKey(0)

face = img[95:195, 100:180]
cv2.imshow("Face", face)
cv2.waitKey(0)

body = img[20:, 35:210]
cv2.imshow("Body", body)
cv2.waitKey(0)

cv2.destroyAllWindows()
```





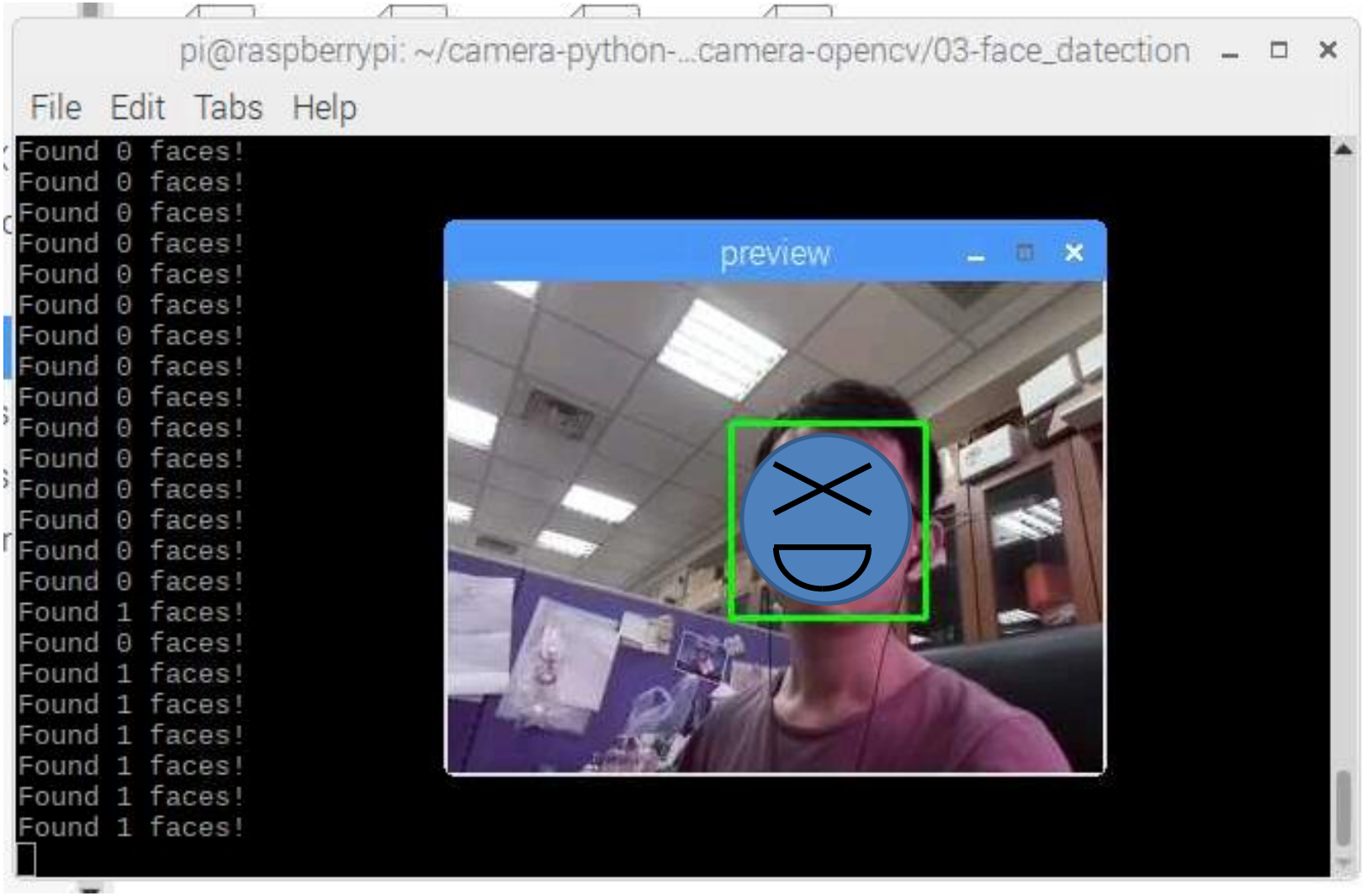




Install opencv

- Command
  - **sudo apt-get install python-opencv**
  - Download sample code and unzip it
  - Load module: **sudo modprobe bcm2835-v4l2**
- Two sample code
  - Analyze image
    - python image_face_detect.py **your_img** haarcascade_frontalface_default.xml
  - Analyze camera
    - python camera_face_detect.py haarcascade_frontalface_default.xml

Facial detection



Facial detection (python)

python camera_face_detect.py haarcascade_frontalface_default.xml

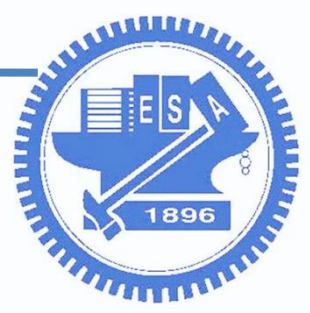
```
import sys
import cv2

cascPath = sys.argv[1]
faceCascade = cv2.CascadeClassifier(cascPath)

if cv2.__version__.startswith('2'):
    PROP_FRAME_WIDTH = cv2.cv.CV_CAP_PROP_FRAME_WIDTH
    PROP_FRAME_HEIGHT = cv2.cv.CV_CAP_PROP_FRAME_HEIGHT
    HAAR_FLAGS = cv2.cv.CV_HAAR_SCALE_IMAGE

....

cap = cv2.VideoCapture(0)
cap.set(PROP_FRAME_WIDTH, 320)
cap.set(PROP_FRAME_HEIGHT, 240)
```

Facial detection (python)

```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=HAAR_FLAGS
    )
```

Facial detection (python)

```
print "Found {0} faces!".format(len(faces))

    # Draw a rectangle around the faces
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)

    # Display the resulting frame
    cv2.imshow("preview", frame)

    if cv2.waitKey(1) & 0xFF == ord("q"):
        break

# When everything is done, release the capture
cap.release()
cv2.destroyAllWindows()
```

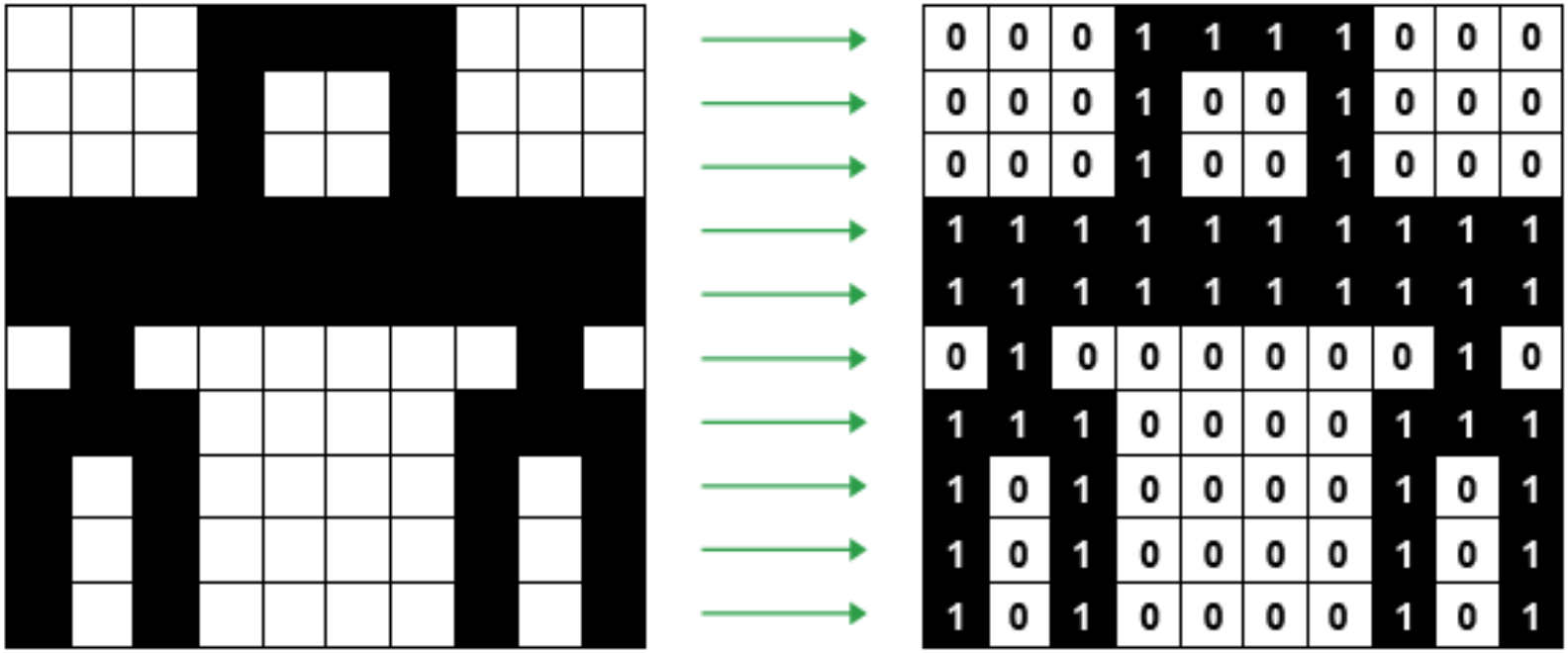


Cascade Classification

- Haar Feature-based Cascade Classifier for Object Detection
  - The object detector described below has been initially proposed by Paul Viola [Viola01] and improved by Rainer Lienhart [Lienhart02].
  - a classifier is trained with a few hundred sample views of a particular object (i.e., a face or a car), called positive examples
    - Output 1: the region is likely to show the object (i.e., face/car)
    - Output 0: otherwise

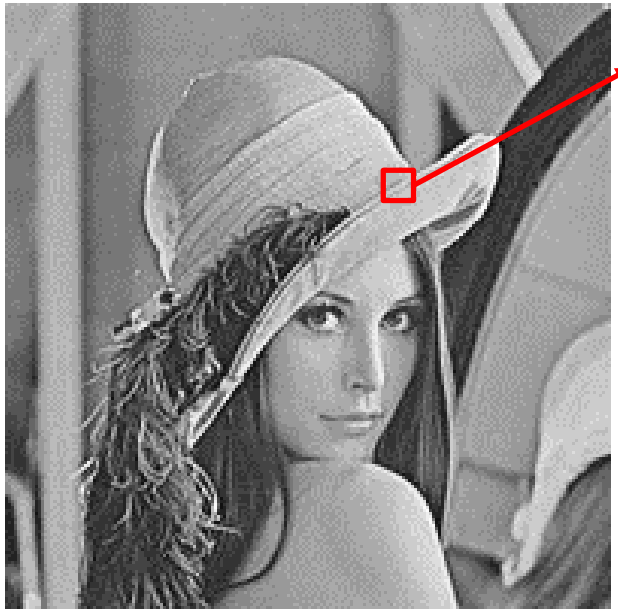
Bitmap images

- Example: black-and-white image



Bitmap images

- Example: grayscale picture
  - 8 bits per pixel
  - This pixel depth allows 256 different intensities

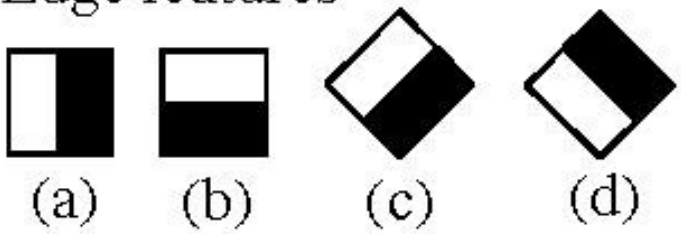


154	108	198	216	52
61	168	148	52	45
72	80	55	134	39
89	129	232	204	155
156	99	118	125	83

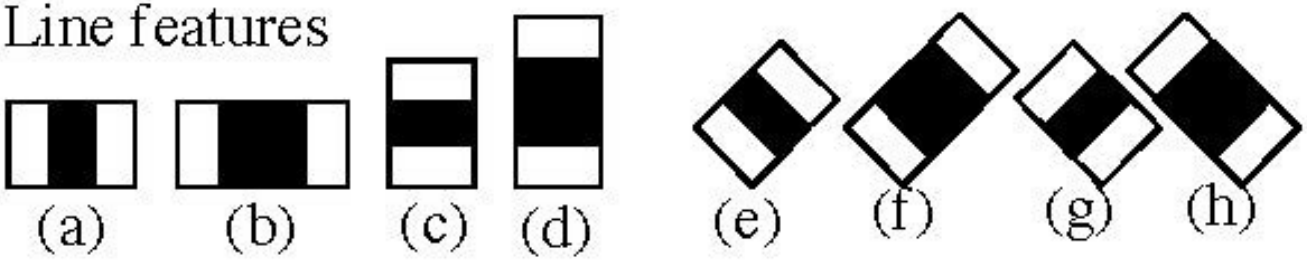
Fig source: https://zh.wikipedia.org/wiki/%E8%90%8A%E5%A8%9C%E5%9C%96

Haar-Like Features

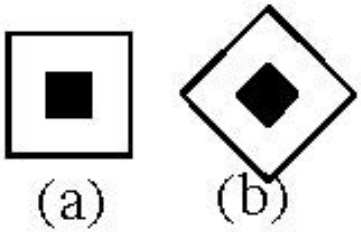
1. Edge features



2. Line features



3. Center-surround features



Find features

- Pick a scale (ex: 24x24 pixels) for the feature
- Slide it across the image
- Compute the average pixel values under the white area and the black area
- If the difference between the areas is above some threshold, the feature matches

Find features

1. Calculate the average of white/black pixel
2. Calculate the difference

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

image

0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

Edge feature

$$\Delta = black - whilte = 1$$

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

image

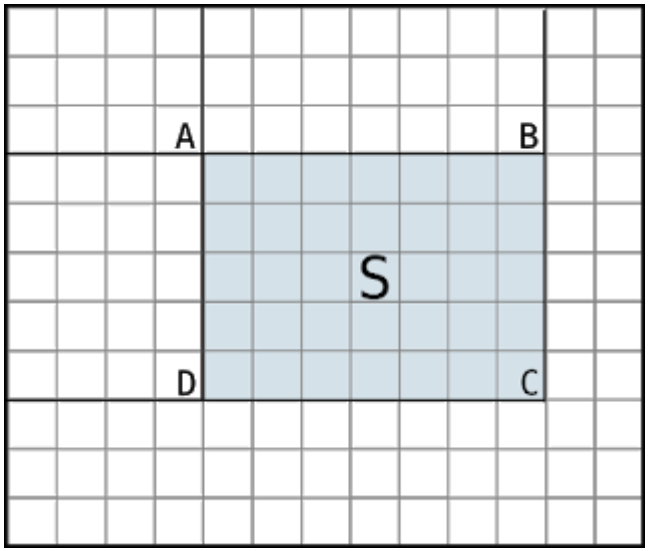
0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

Edge feature

$$\Delta = \frac{0.6 + 0.8 + \cdots}{8} - \frac{0.1 + 0.2 + \cdots}{8}$$
$$= 0.7375 - 0.175 = 0.56$$

Integral Image

- a quick and effective way of calculating the sum of values (pixel values) of a rectangular subset of a grid
- It can also used for calculating the average intensity within a given image.



Sum = Value(C) - Value(B) - Value(D) + Value(A)

Integral Image

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

Original image

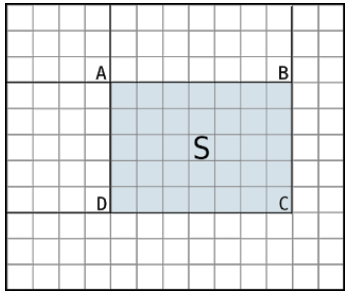
0.1	0.3	0.9	0.8
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

integral image

Integral Image

0.1	0.2	0.6	0.8
0.2	0.3	0.8	0.6
0.2	0.1	0.6	0.8
0.2	0.1	0.8	0.9

0.1 A	0.3	**0.9** B	0.8
0.3	0.8	2.2	3.6
0.5 D	1.1	**3.1** C	5.3
0.7	1.4	4.2	7.3



Calculate the are summation

0.1	0.3	0.9	0.8
0.3	0.8	2.2	3.6
0.5	1.1	3.1 C	5.3
0.7	1.4	4.2	7.3

0.1	0.3	0.9 B	0.8
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

0.1 A	0.3	0.9	0.8
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

0.1	0.3	0.9	0.8
0.3	0.8	2.2	3.6
0.5 D	1.1	3.1	5.3
0.7	1.4	4.2	7.3

Sum = Value(C) - Value(B) + Value(A) - Value(D)

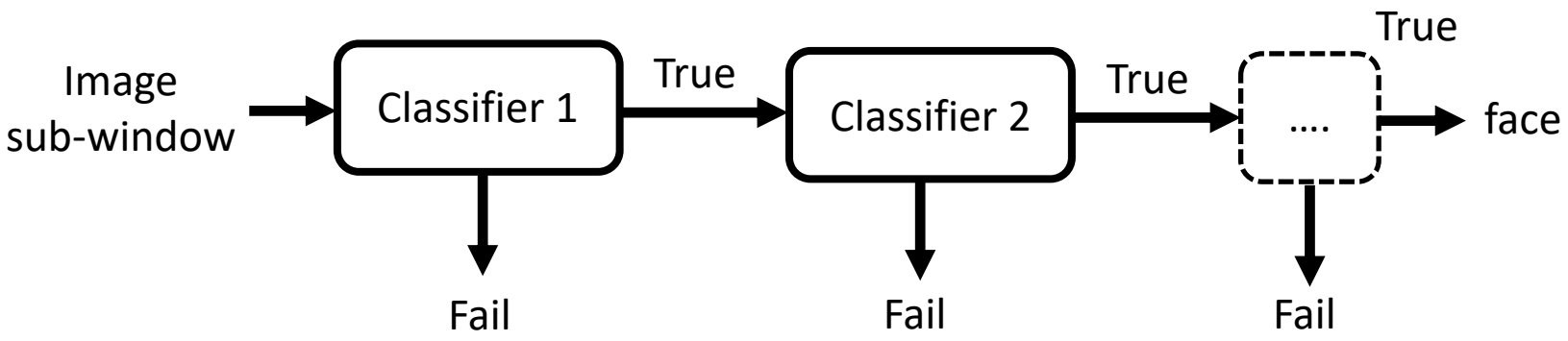
AdaBoost

- Adaptive Boosting
  - Try out multiple weak classifiers over several rounds
  - Select the best weak classifier in each round and combining the best weak classifiers to create a strong classifier

Data point	Classifier 1	Classifier 2	Classifier 3	…
P_1	Pass	Fail	Fail	…
P_2	Pass	Pass	Pass	…
P_3	Fail	Pass	Pass	…
…	…	…	…	…

Cascades

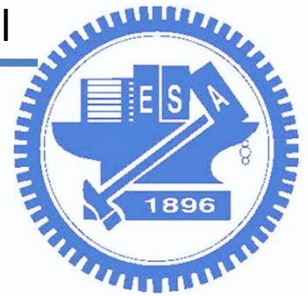
- Haar cascades consists of a series of weak classifiers
  - barely better than 50% correct
  - If an area passes a single classifier, go to the next classifier; otherwise, area doesn't match



Recall the code

```
while True:
    # Capture frame-by-frame
    ret, frame = cap.read()

    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    faces = faceCascade.detectMultiScale(
        gray,
        scaleFactor=1.1,
        minNeighbors=5,
        minSize=(30, 30),
        flags=HAAR_FLAGS
    )
```

https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html



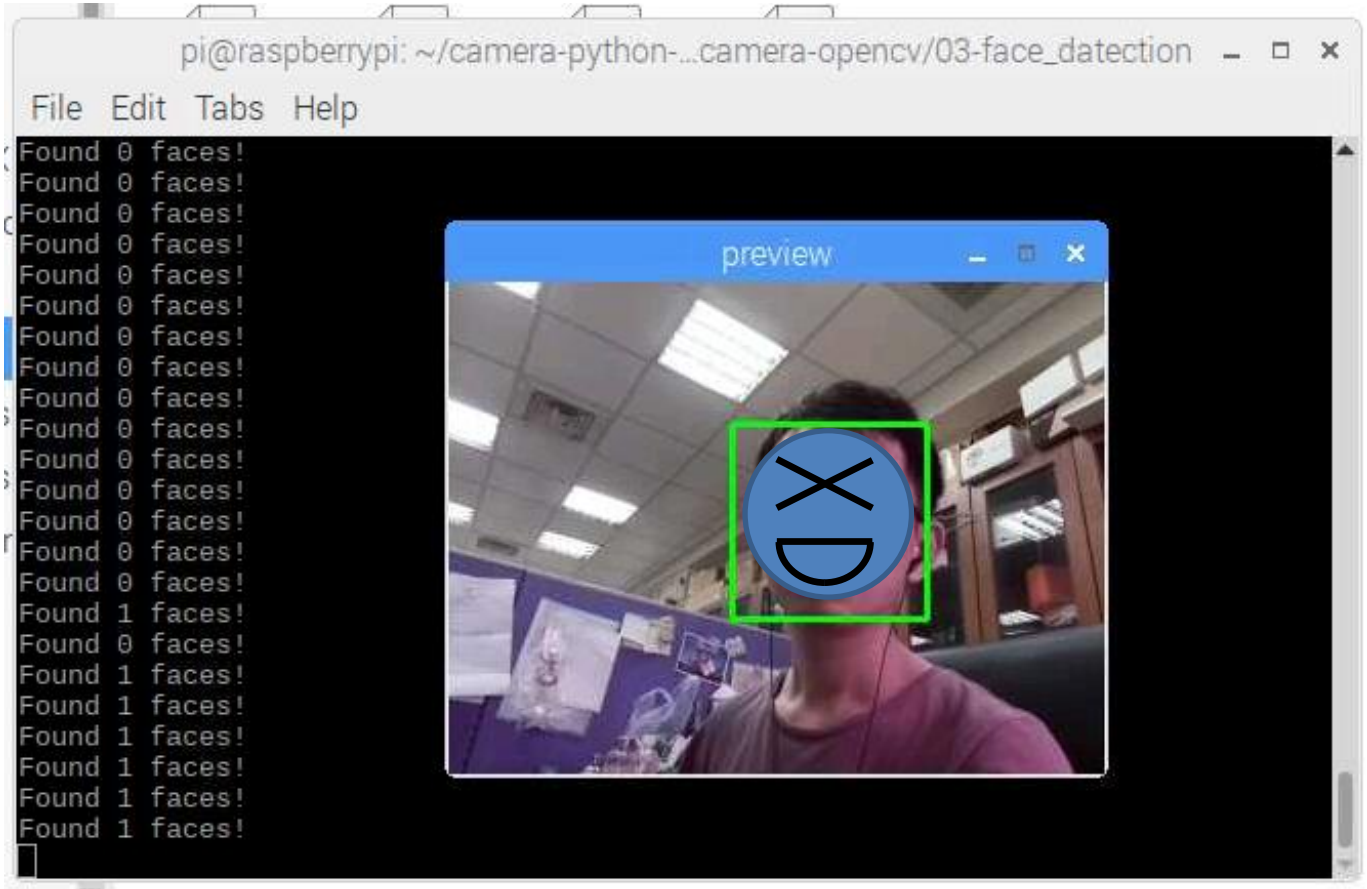
Related parameters

- **cascade** – Haar classifier cascade (OpenCV 1.x API only). It can be loaded from XML or YAML file using Load(). When the cascade is not needed anymore, release it using cvReleaseHaarClassifierCascade(&cascade).
- **image** – Matrix of the type CV_8U containing an image where objects are detected.
- **objects** – Vector of rectangles where each rectangle contains the detected object.
- **scaleFactor** – Parameter specifying how much the image size is reduced at each image scale.
- **minNeighbors** – Parameter specifying how many neighbors each candidate rectangle should have to retain it.
- **flags** – Parameter with the same meaning for an old cascade as in the function cvHaarDetectObjects. It is not used for a new cascade.
- **minSize** – Minimum possible object size. Objects smaller than that are ignored.
- **maxSize** – Maximum possible object size. Objects larger than that are ignored.

Try to use different **parameters**, you will get different results.

Face detection on PI

- python camera_face_detect.py haarcascade_frontalface_default.xml



Reference

- Online resource
  - Facial Detection
    - https://www.youtube.com/watch?v=sWTvK72-SPU
  - Computer Vision - Haar-Features
    - https://www.youtube.com/watch?v=F5rysk51txQ
  - Computer Vision - Integral Images
    - https://www.youtube.com/watch?v=x41KFOFGnUE
  - Recognition Part II: Face Detection via AdaBoost
    - https://courses.cs.washington.edu/courses/cse455/16wi/notes/15_FaceDetection.pdf

Reference

- [Viola01] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001.
  - http://research.microsoft.com/en-us/um/people/viola/Pubs/Detect/violaJones_CVPR2001.pdf
- [Lienhart02] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. IEEE ICIP, Vol. 1, pp. 900-903, Sep. 2002.
  - http://www.multimedia-computing.de/mediawiki//images/5/52/MRL-TR-May02-revised-Dec02.pdf