嵌入式系統設計概論與實作

曾煜棋、吳昆儒

National Yang Ming Chiao Tung University

Last week

- 嵌入式應用: 網路攝影機
  - Raspberry Pi Camera
  - Python + OpenCV
  - Calculate FPS
  - 建立網路串流

This week

- 嵌入式應用: 網路攝影機
  - 影像辨識 (opencv)
    - 圖片旋轉, 裁切, 縮放
    - 人臉識別
    - 人臉輪廓識別

Requirement

```
# this should be done in the last class
sudo apt-get install python3-opencv
pip3 install imutils
pip3 install numpy
pip3 install dlib
```

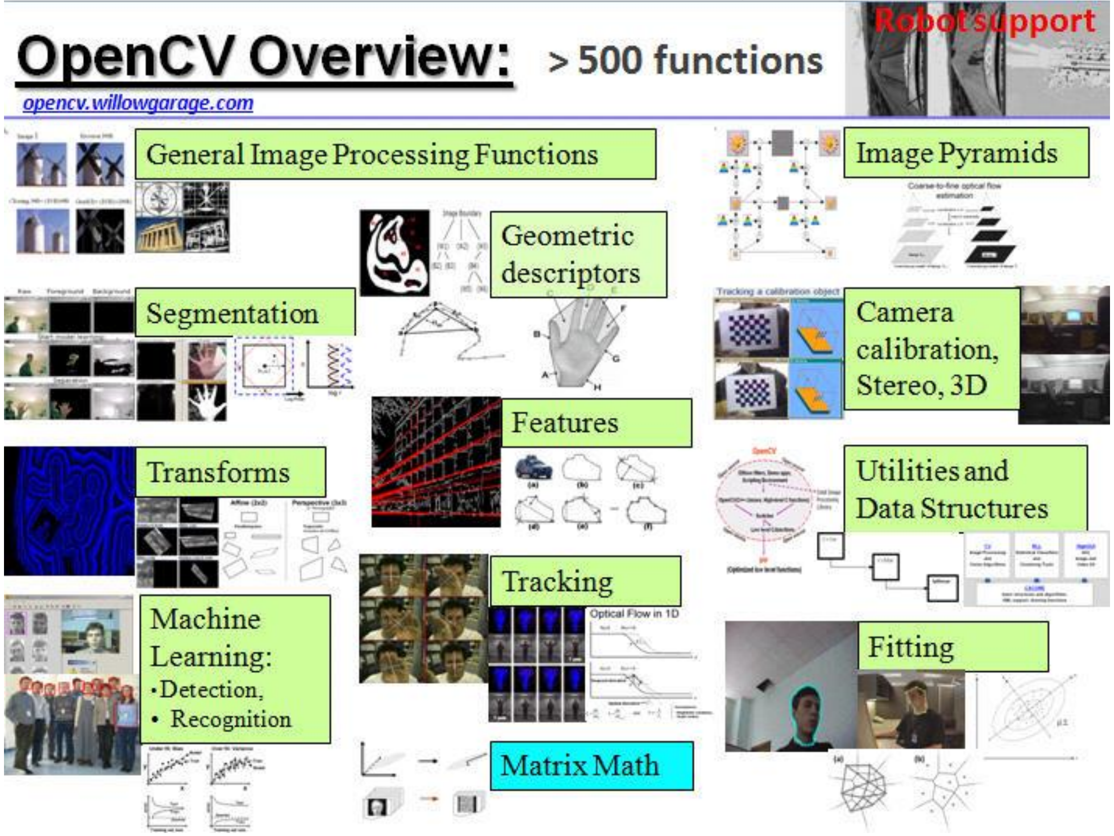
Err: Failed building wheel for dlib
Sol: **pip3 install --upgrade pip**

Err: CMake must be installed to build the following extensions: dlib
Sol: **sudo apt-get install cmake**

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

OpenCV

□ Open Source Computer Vision Library



OpenCV Overview: > 500 functions

opencv.willowgarage.com

Robot support

General Image Processing Functions

Image Pyramids

Geometric descriptors

Segmentation

Camera calibration, Stereo, 3D

Features

Transforms

Utilities and Data Structures

Tracking

Machine Learning:
- Detection,
- Recognition

Fitting

Matrix Math

Install OpenCV:

- ~~Python2: sudo apt-get install python-opencv~~
- Python3: sudo apt-get install python3-opencv

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()
```

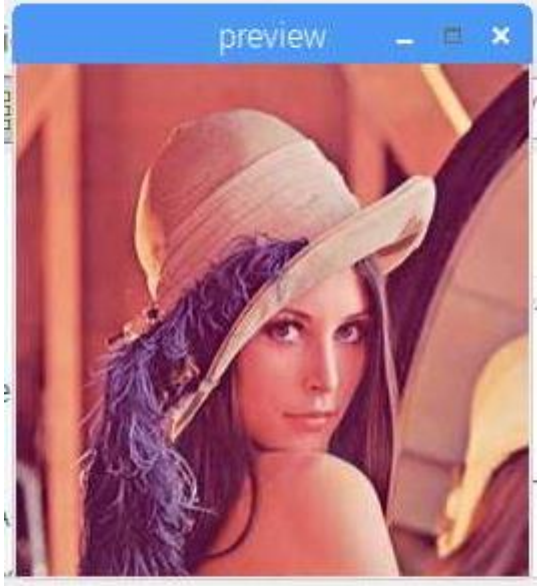
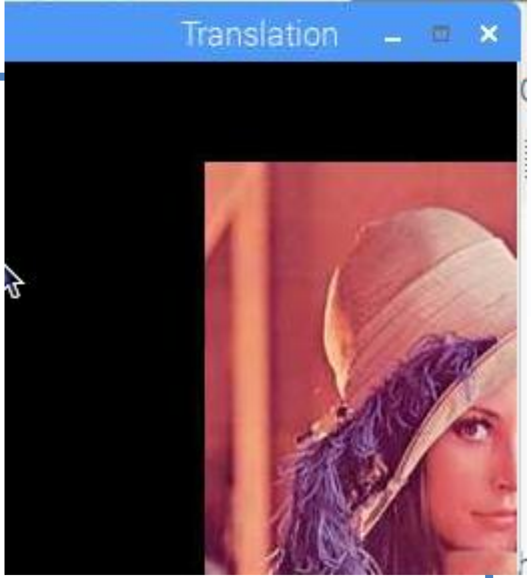


Fig source: https://upload.wikimedia.org/wikipedia/zh/3/34/Lenna.jpg

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

Translation



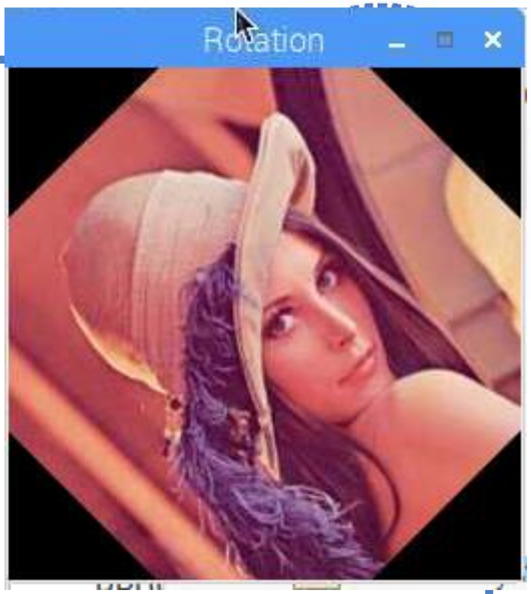
- Applies an affine transformation to an image.

```
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.

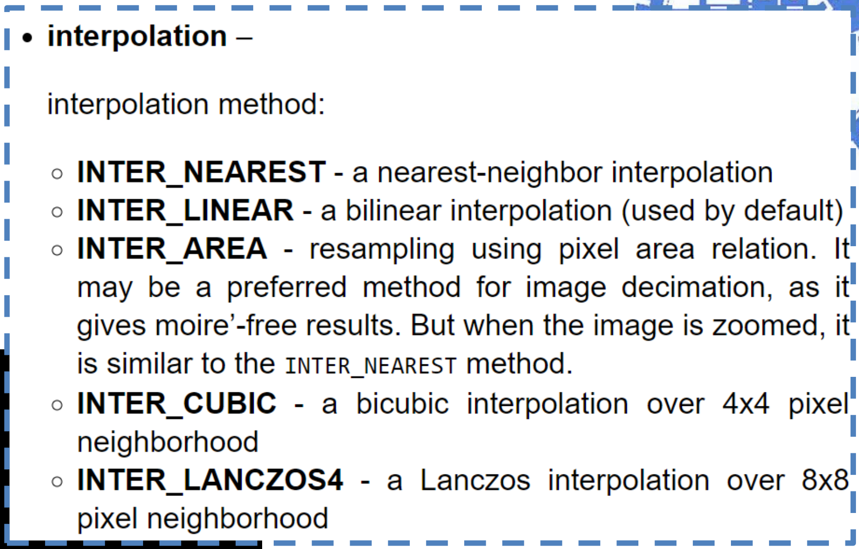
```
import cv2
import numpy as np

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.

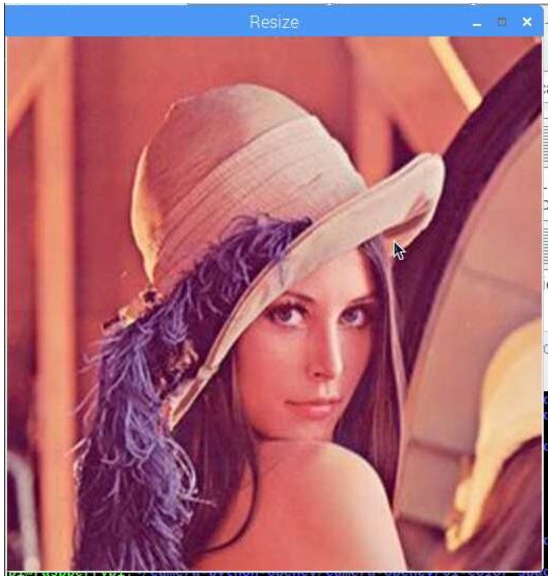
```
import cv2
import numpy as np

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()
```



- **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

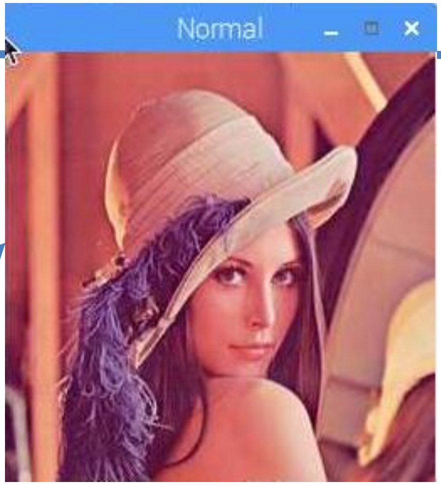


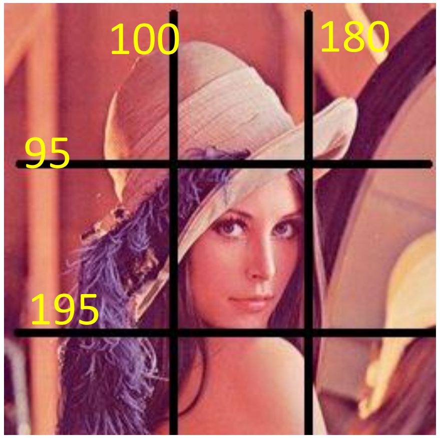
https://docs.opencv.org/2.4/doc/user_guide/ug_mat.html

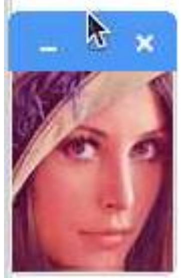
Crop

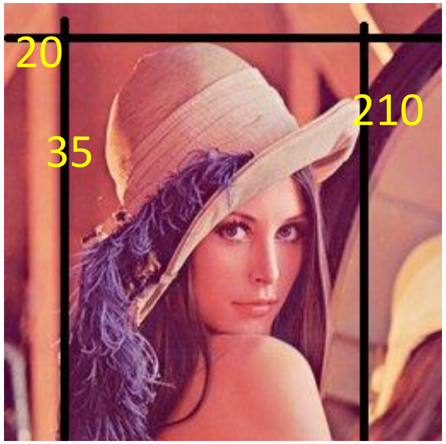
- □ Sample code

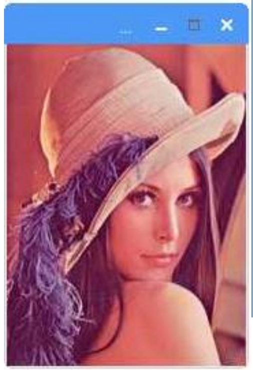
```
import cv2
import numpy as np

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()
```









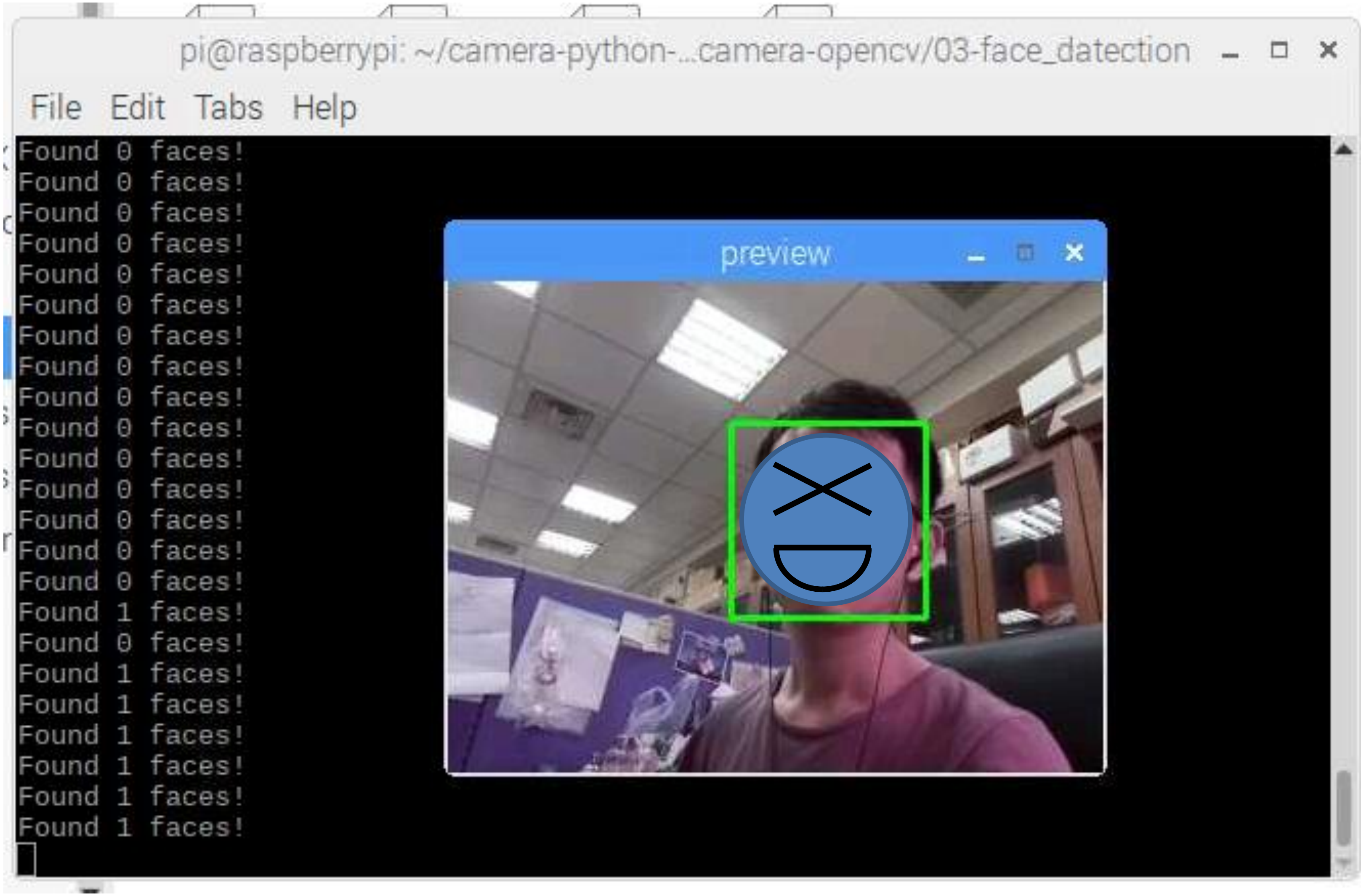


Install opencv

- Command
  - sudo apt-get install python3-opencv
  - Download sample code and unzip it
  - Load module: sudo modprobe bcm2835-v4l2

- Two sample code
  - Analyze image
    - python 1.1image_face_detect.py
  - Analyze stream from camera
    - python 1.2camera_face_detect.py

1. Facial detection



pi@raspberrypi: ~/camera-python-...camera-opencv/03-face_datection
File Edit Tabs Help
preview

```
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 1 faces!
Found 0 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
Found 1 faces!
```

1. Facial detection (python)

python 1.1image_face_detect.py

```
import sys
import cv2

imagePath = "img.jpg"

# Create the haar cascade
cascPath = "model/haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascPath)

# Read the image
image = cv2.imread(imagePath)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

1. Facial detection (python)

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30)
)

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

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

cv2.imshow("preview", image)
cv2.waitKey(0)

cv2.destroyAllWindows()
```

Face detection flow

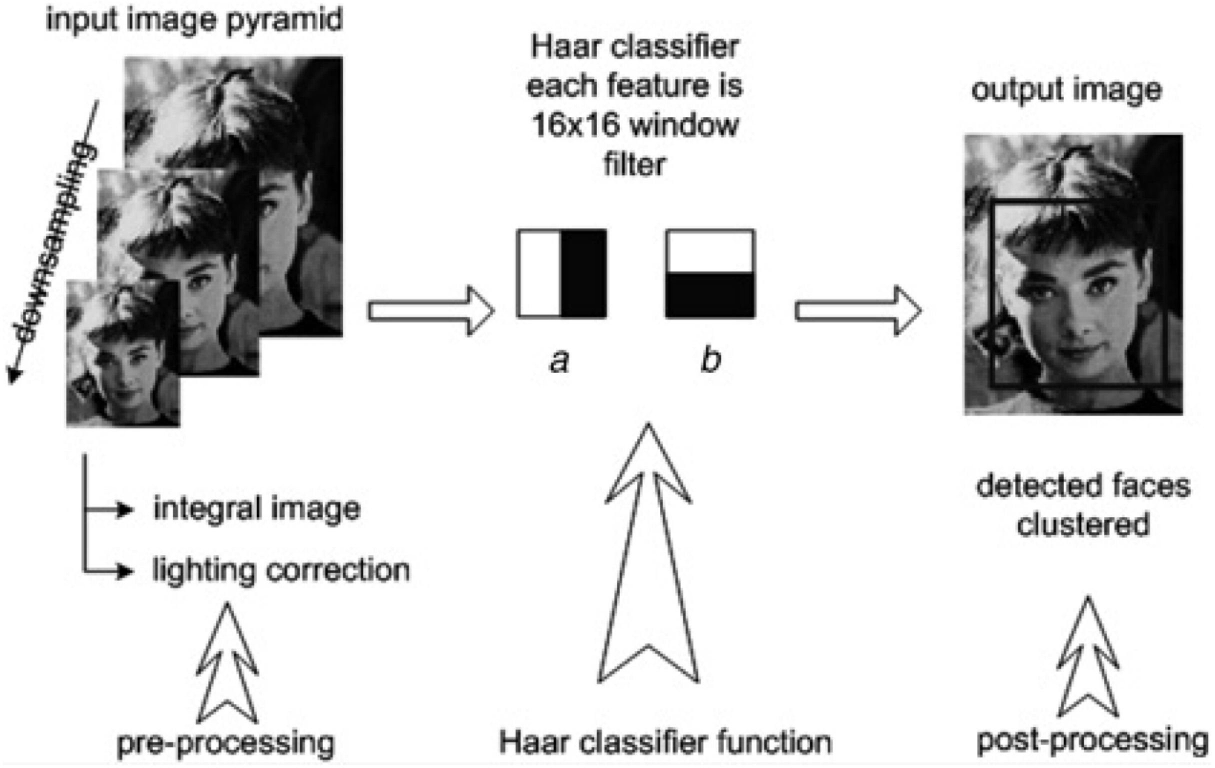


Figure 1 *Face detection flow based on the Haar classifier*

Paper: Field programmable gate array-based Haar classifier for accelerating face detection algorithm https://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

A. Cascade Classification

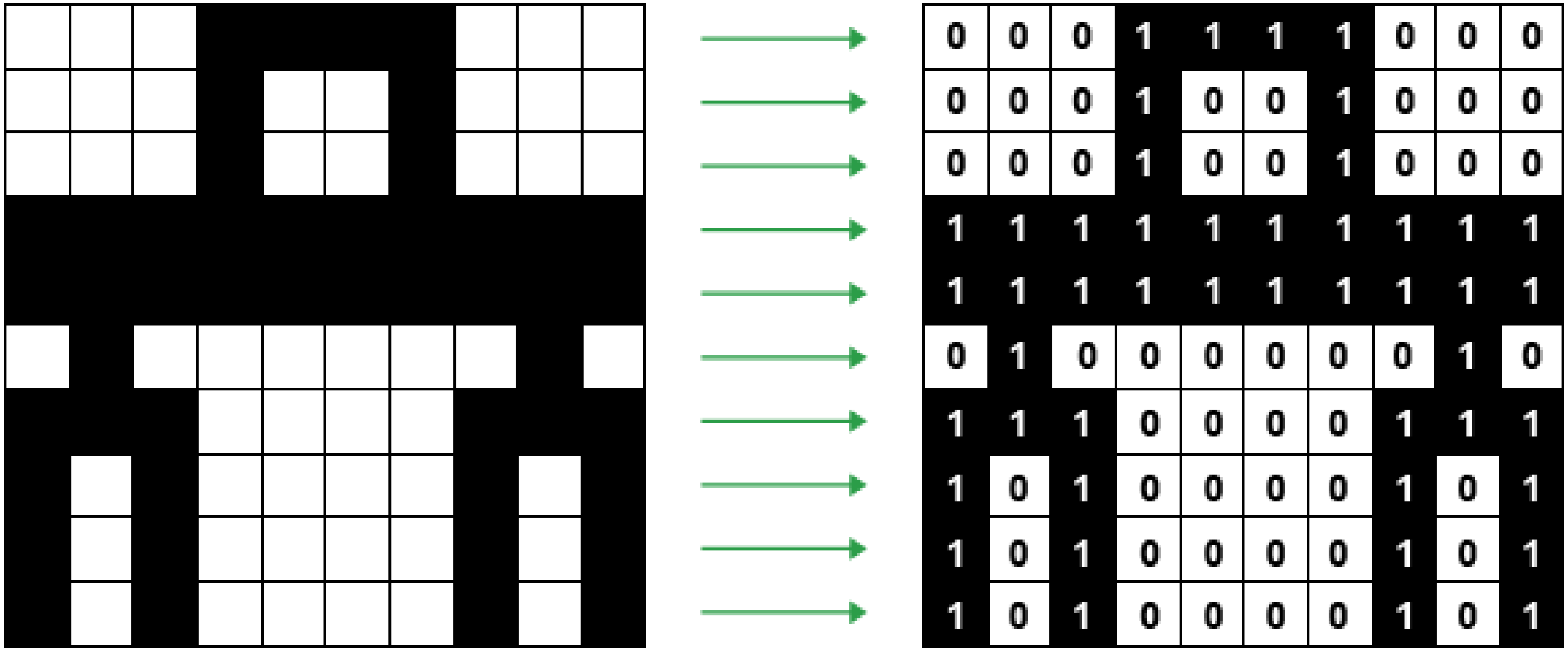
- Haar Feature-based Cascade Classifier for Object Detection
  - The object detector described below has been initially proposed by Paul Viola [Viola01] and improved by Rainer Lienhart [Lienhart02].
  - A classifier is trained with a few hundred sample views of a particular object (i.e., a face or a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

- [Viola01] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. IEEE CVPR, 2001. https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.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://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.214.9150&rep=rep1&type=pdf

https://www.bbc.com/bitesize/guides/zqyrq6f/revision/1

Bitmap images

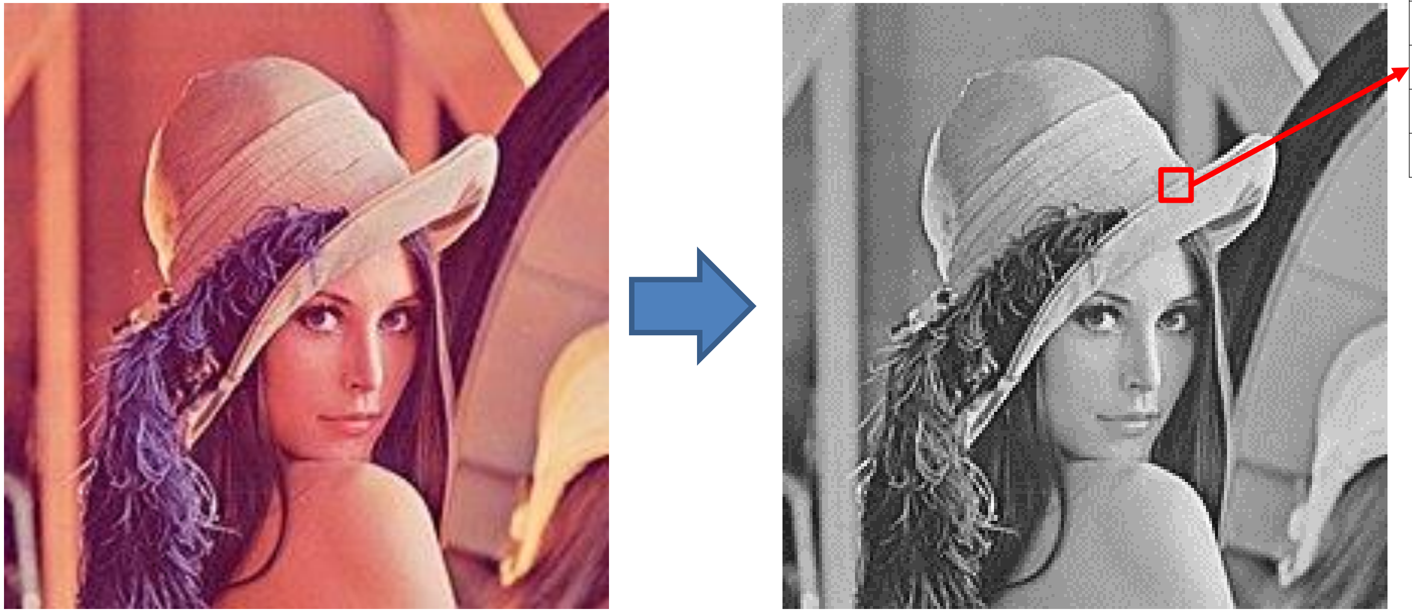
- Example: black-and-white image



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

Bitmap images

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

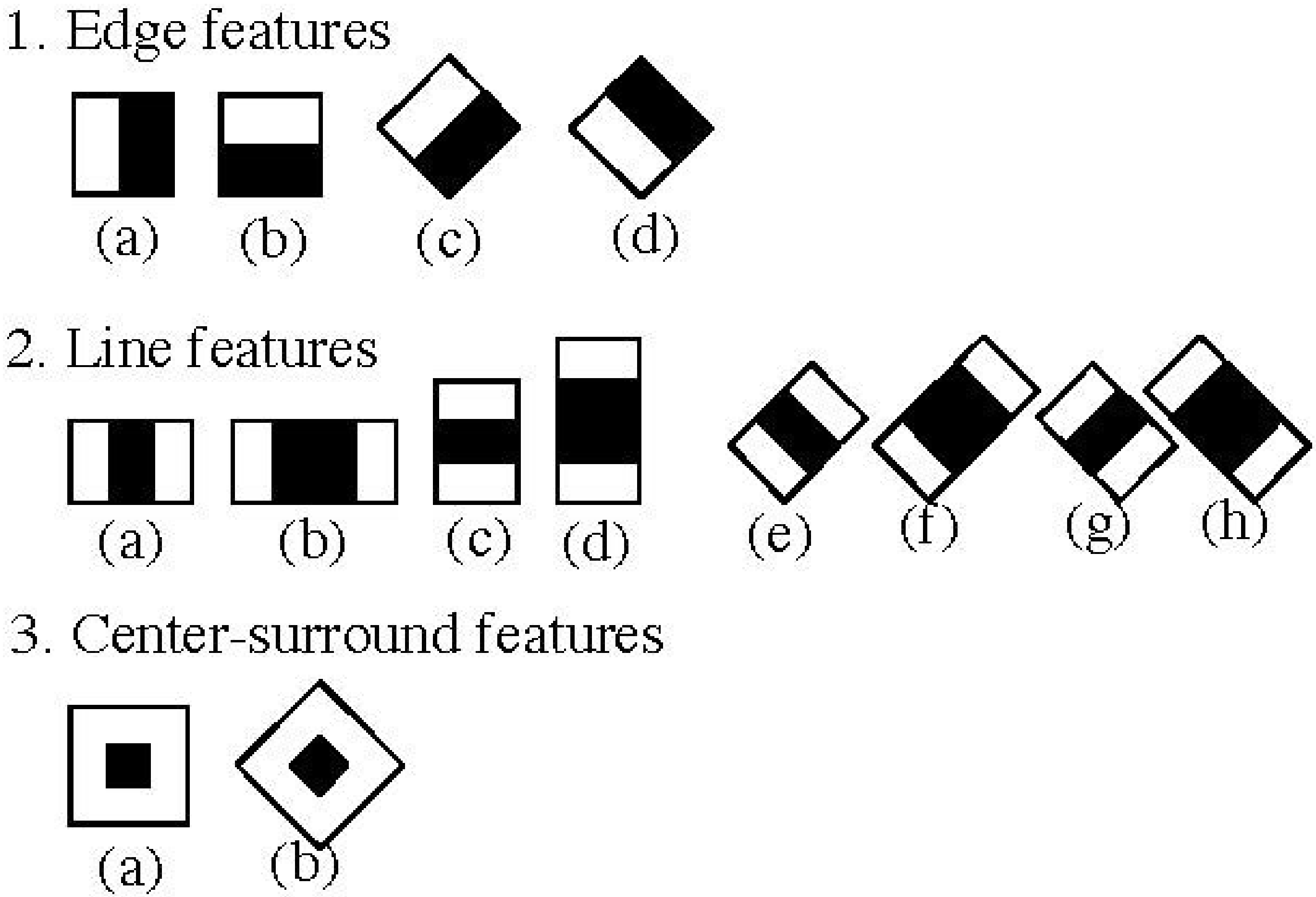


154	108	198	216	52
61	168	148	52	45
72	80	55	134	39
89	129	232	204	155
156	99	118	125	83

Camera sees this

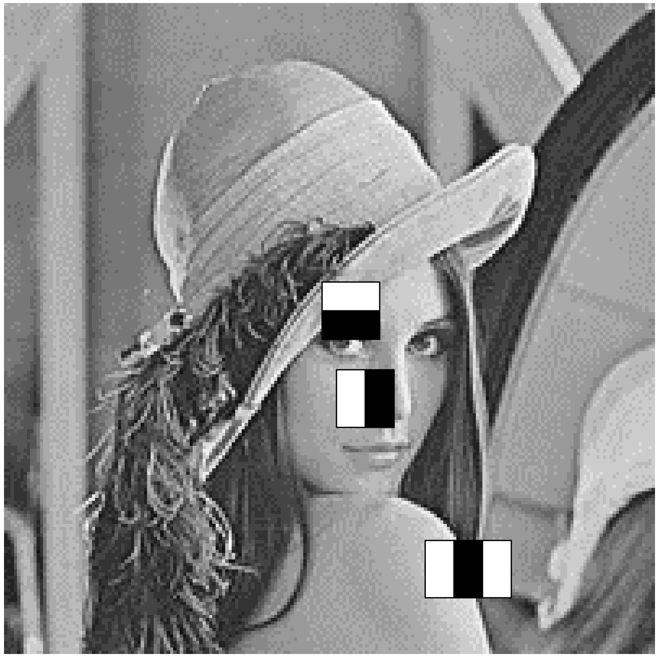
Fig source: https://zh.wikipedia.org/wiki/%E8%90%8A%E5%A8%9C%E5%9C%96

Haar-Like 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

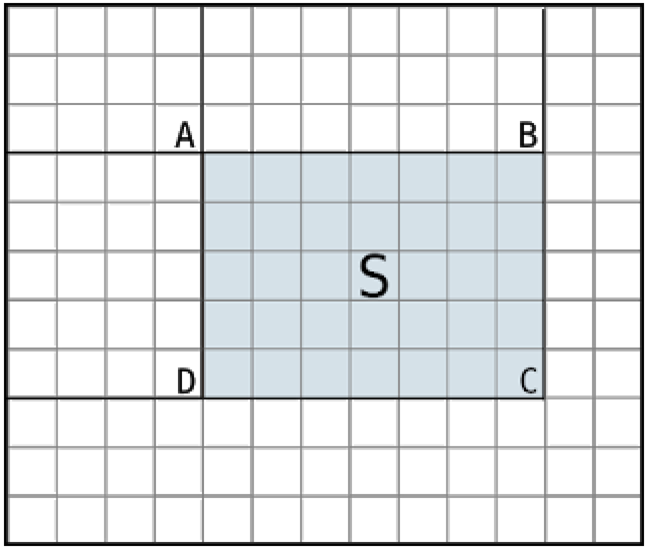
0	0	1	1
0	0	1	1
0	0	1	1
0	0	1	1

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

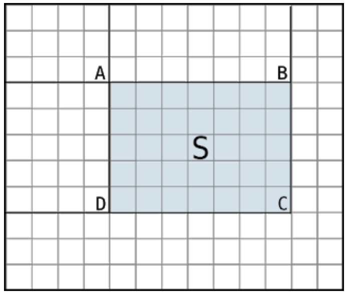
0.1	0.3	0.9	1.7
0.3	0.8	2.2	3.6
0.5	1.1	3.1	5.3
0.7	1.4	4.2	7.3

integral image

Integral Image

0.1	0.2	0.6	1.7
0.2	0.3	0.8	3.6
0.2	0.1	0.6	5.3
0.2	0.1	0.8	7.3

0.1 A	0.3	**0.9** B	1.7
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	1.7
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	1.7
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	1.7
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	1.7
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)

Discussion

- How to calculate the area summation by integral image?
  - 1. Write down the value of integral image
  - 2. Sum = Value(C) - Value(B) + Value(A) - Value(D) = ?????

0.1	0.2	0.3	0.5	0.8
0.2	0.2	0.3	0.7	0.9
0.1	0.2	0.4	0.7	0.9
0.3	0.1	0.2	0.8	0.2
0.1	0.2	0.4	0.6	0.1

→

???

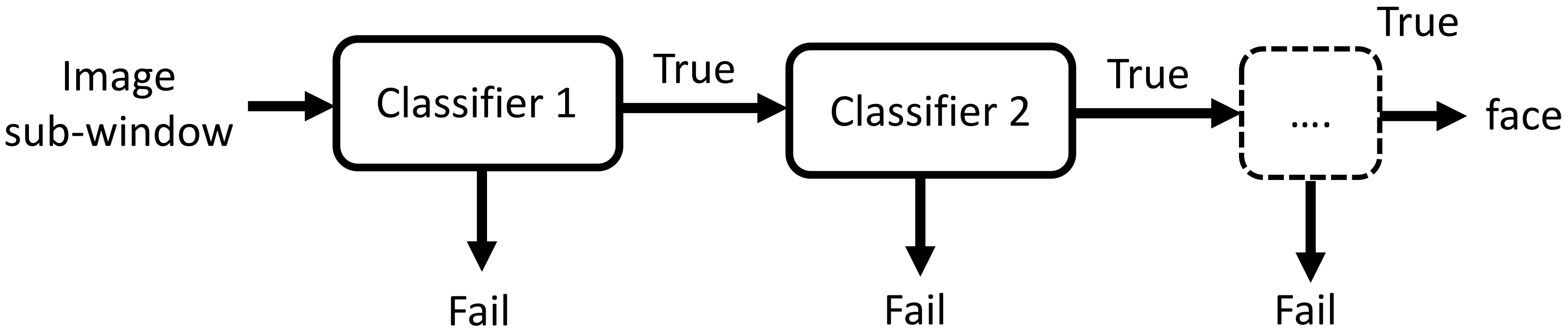
B. AdaBoost

- 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	…
…	…	…	…	…

C. 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

```
# Detect faces in the image
faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor=1.1,
    minNeighbors=5,
    minSize=(30, 30)
)
```

- **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.
- **minSize** – Minimum possible object size. Objects smaller than that are ignored.

https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html

Related parameters

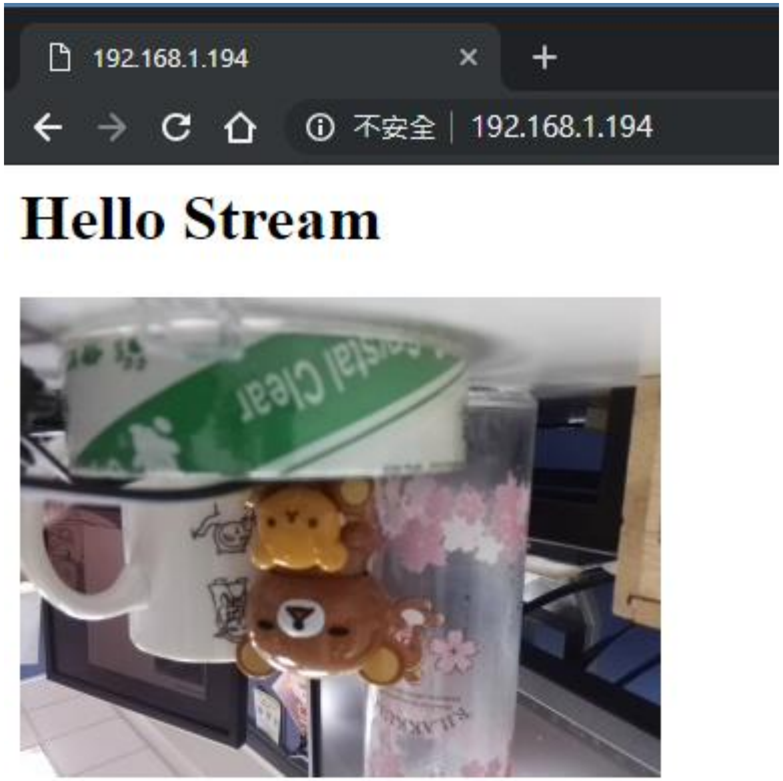
CascadeClassifier::detectMultiScale

- **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, the rectangles may be partially outside the original image.
- **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.
  If maxSize == minSize model is evaluated on single scale.

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

Previous class

- MJPG on PI

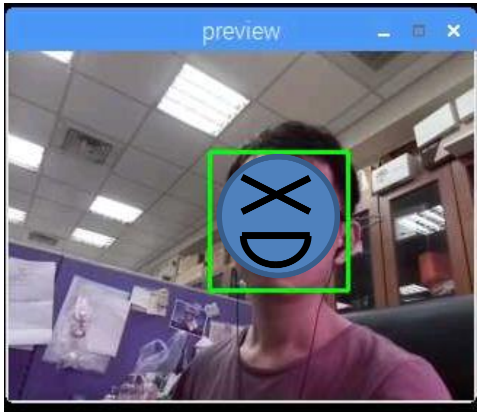


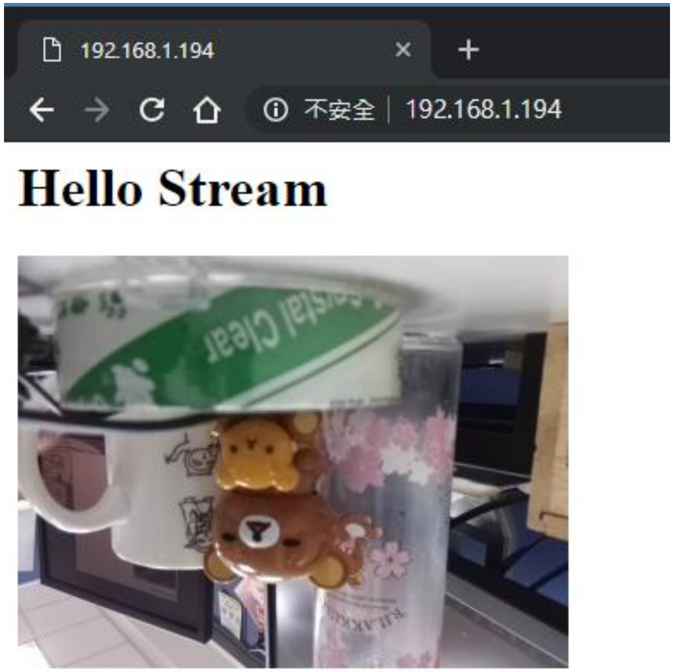
Hello Stream

No stream? You might need: **sudo modprobe bcm2835-v4l2**

Quiz 1

- Based on the sample code:
  - 1. MJPG with Picamera module
  - 2. Facial detection
- Combine them together!





Hello Stream

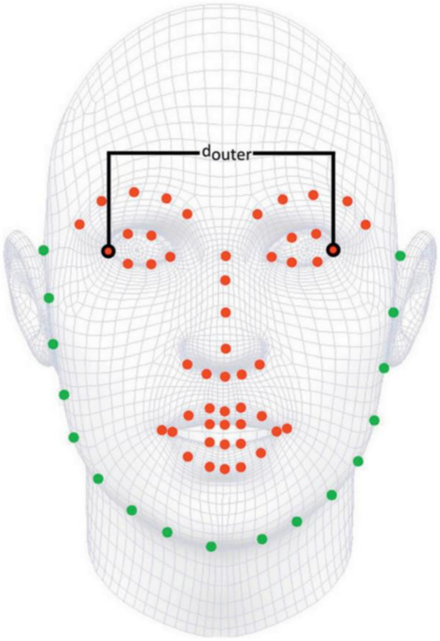
We can use browser to watch the facial detection results.

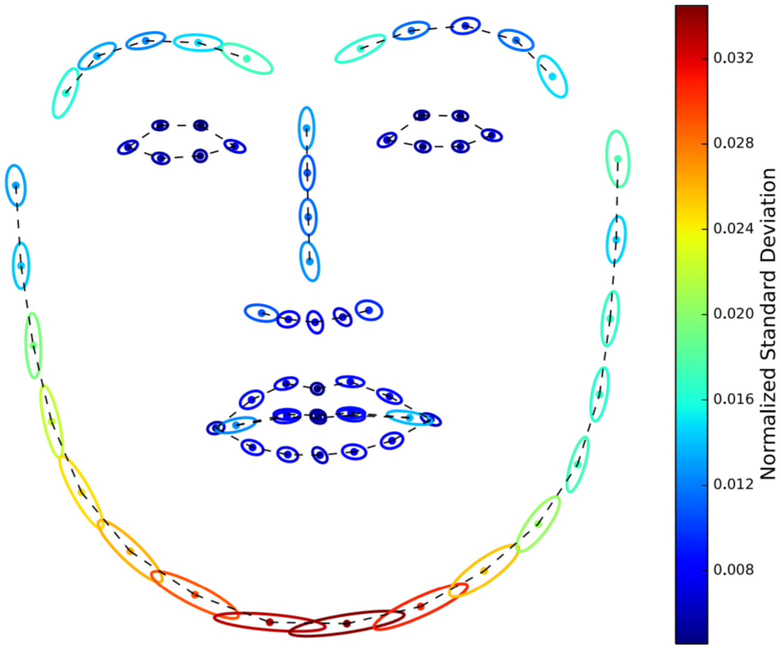
https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/

2. Facial landmark

□ 300 Faces In-The-Wild Challenge: database and results

https://ibug.doc.ic.ac.uk/media/uploads/documents/sagonas_2016_imavis.pdf





Two steps for capturing facial landmark:

A. Face detection (OpenCV or dlib)

B. Draw landmark (dlib)

A1. Face detection (OpenCV)

```
# load OpenCV's Haar cascade for face detection,
# (This is faster than dlib's built-in HOG detector, but less accurate.)
detector_file = "model/haarcascade_frontalface_default.xml“
detector = cv2.CascadeClassifier(detector_file)

# detect faces in the grayscale frame by opencv's method: (x, y, w, h)
rects = detector.detectMultiScale(gray, scaleFactor=1.1,
    minNeighbors=5, minSize=(30, 30),
    flags=cv2.CASCADE_SCALE_IMAGE)
```

http://dlib.net/face_detection_ex.cpp.html
http://dlib.net/python/index.html#dlib.get_frontal_face_detector
https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html

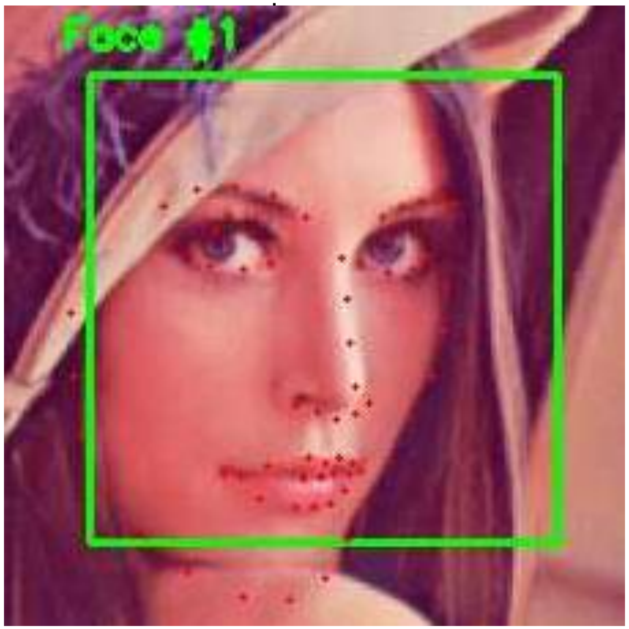
A2. Face detection (dlib)

```
# initialize dlib's face detector (HOG-based)
# then create the facial landmark predictor
detector = dlib.get_frontal_face_detector()

# load the input image, resize it, and convert it to grayscale
image = cv2.imread("img.jpg”)
image = imutils.resize(image, width=500)

# cvtColor: Converts an image from one color space to another.
# Here, convert a RGB image to gray
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# detect faces in the grayscale image
# The 1 in the second argument indicates that we should upsample the image 1 time.
# This will make everything bigger and allow us to detect more faces.
rects = detector(gray, 1)
```

https://github.com/jrosebr1/imutils/blob/master/imutils/face_utils/helpers.py
http://dlib.net/face_detector.py.html

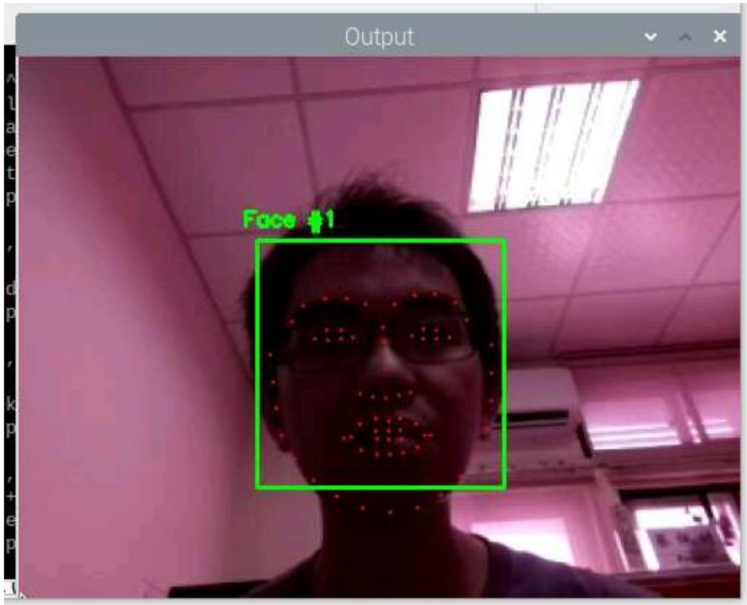
B. Draw landmark

```
# loop over the face detections
face_counter = 0
for (x, y, w, h) in rects:
    # construct a dlib rectangle object from the Haar cascade bounding box
    rect = dlib.rectangle(int(x), int(y), int(x + w), int(y + h))

    # determine the facial landmarks for the face region,
    # then convert the facial landmark (x, y)-coordinates to a NumPy array
    shape = predictor(gray, rect)
    shape = face_utils.shape_to_np(shape)

    # convert dlib's rectangle to a OpenCV-style bounding box
    # [i.e., (x, y, w, h)], then draw the face bounding box
    (x, y, w, h) = face_utils.rect_to_bb(rect)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)

    # show the face number
    cv2.putText(image, "Face #{}".format(face_counter + 1), (x - 10, y - 10),
        cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

    # loop over the (x, y)-coordinates for the facial landmarks and draw them on the image
    for (x, y) in shape:
        cv2.circle(image, (x, y), 1, (0, 0, 255), -1)

    face_counter = face_counter + 1
```



Quiz2: Picamera + facial landmark

- In this sample, it **uses image** to draw facial landmark.
  - image = cv2.imread("img.jpg")
- **Use PI camera to achieve online landmark detection.**
  - Hint: vs = PiVideoStream().start()



Summary

- Practice Lab
  - (opencv example, facial detection, facial landmark)

- Write down the answer for discussion
  - **Discussion**
    - **How to calculate the area summation by integral image?**
  - Deadline: Before 4/30, 12:00

- Write code for Quiz 1&2, then demonstrate it to TAs
  - **Quiz1: MJPG + facial detection**
  - **Quiz2: Picamera + facial landmark**
  - Deadline: Before 4/23, 15:10

- **Next week is midterm!!**

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

- Online resource
  - Increasing Raspberry Pi FPS with Python and OpenCV
    - https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/
  - Raspberry Pi: Facial landmarks + drowsiness detection with OpenCV and dlib
    - https://www.pyimagesearch.com/2017/10/23/raspberry-pi-facial-landmarks-drowsiness-detection-with-opencv-and-dlib/
  - [人臉辨識] 使用5 Facial Landmarks進行臉孔校正
    - https://makerpro.cc/2019/08/face-alignment-through-5-facial-landmarks/

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

- □ 300 Faces In-The-Wild Challenge: database and results, 2016
  - □ https://ibug.doc.ic.ac.uk/media/uploads/documents/sagonas_2016_imavis.pdf