Raspberry Pi Camera and OpenCV

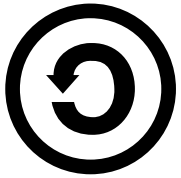
台灣樹莓派 <sosorry@raspberrypi.com.tw>
2018/01/17 @ITRI

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。

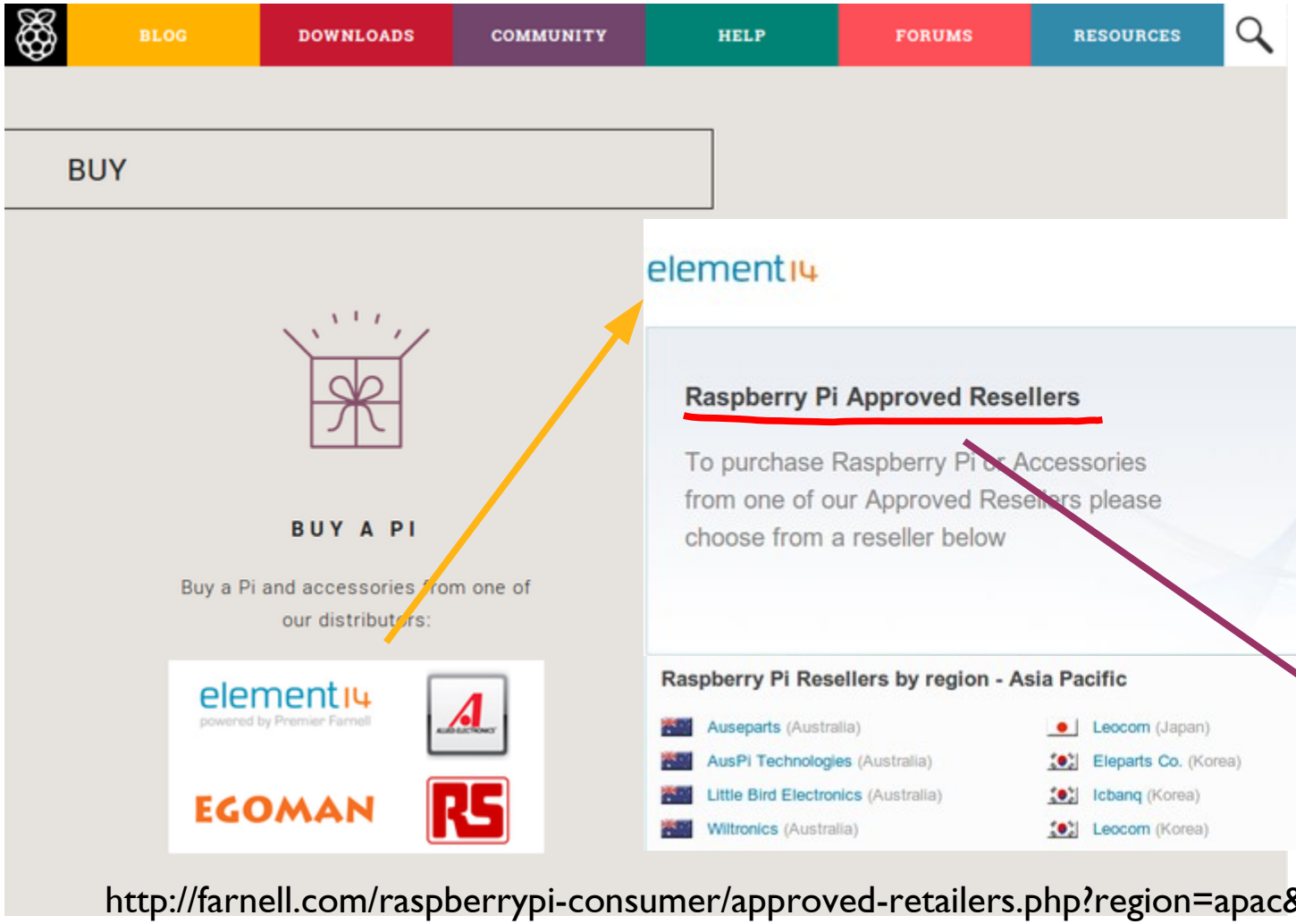
非商業性 — 你不得將本素材進行商業目的之使用。



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。

about 台灣樹莓派

- Raspberry Pi 官方經銷商



http://farnell.com/raspberrypi-consumer/approved-retailers.php?region=apac&MER=MER-LM-OB-RPICC-76315

社群 x 活動

- 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 講者
- 投影片
  - http://www.slideshare.net/raspberrypi-tw/presentations
- 程式碼
  - https://github.com/raspberrypi-tw

大綱

- 色彩空間與基本影像處理
  - 色彩空間介紹
  - 用 Python + OpenCV 做影像處理
- 常用影像處理方法
  - 平滑，侵蝕與膨脹
  - 找邊緣與找直線
  - 找重心與找輪廓
- 機器學習應用與綜合練習
  - 人臉偵測
  - 圖形分類

今日環境

- 硬體：Raspberry Pi 3
- 作業系統：2017-11-29-raspbian-stretch.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(已安裝)

- $ sudo apt-get update
- $ sudo apt-get install -y festival python-dev python-opencv python-pip x11vnc liblivemedia-dev libv4l-dev cmake libpthread-stubs0-dev vlc python-matplotlib
- $ sudo pip2 install request flask numpy

Types in Python2

- list: 內建型別
- array: 需要載入外部模組
  - python.array: from array import array
  - numpy.array: import numpy as np
  - python.array(I/O) vs.numpy.array(運算)
- matrix: 需要載入外部模組
  - from numpy import matrix

https://docs.python.org/2/library/stdtypes.html

numpy.array 運算

- numpy array 被稱為 ndarray, 常用屬性如下：
  - ndarray.ndim: 陣列的維度
  - ndarray.shape: 陣列的各維數大小
  - ndarray.size: 陣列元素的總個數
  - ndarray.dtype: 陣列元素類型
  - ndarray.itemsize: 陣列每個元素 byte 數

用互動模式瞭解 numpy.array 運算

- $ python

```
>>> import numpy as np
>>> a = np.arange(15).reshape(3, 5)
>>> a
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

常用屬性

```
>>> a.ndim
2
>>> a.shape
(3, 5)
>>> a.size
15
>>> a.dtype
dtype('int32')
>>> a.itemsize
4
```

自動轉型

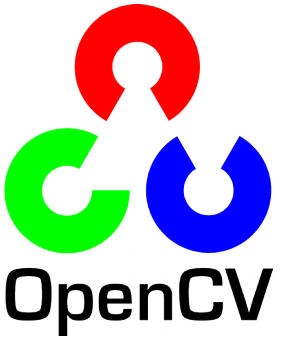
```
>>> b = np.array(["1", 2])
>>> b.ndim
1
>>> b.shape
(2,)
>>> b.size
2
>>> b.dtype
dtype('S1')
>>> b.itemsize
1
```

實驗 1：空間轉換與處理

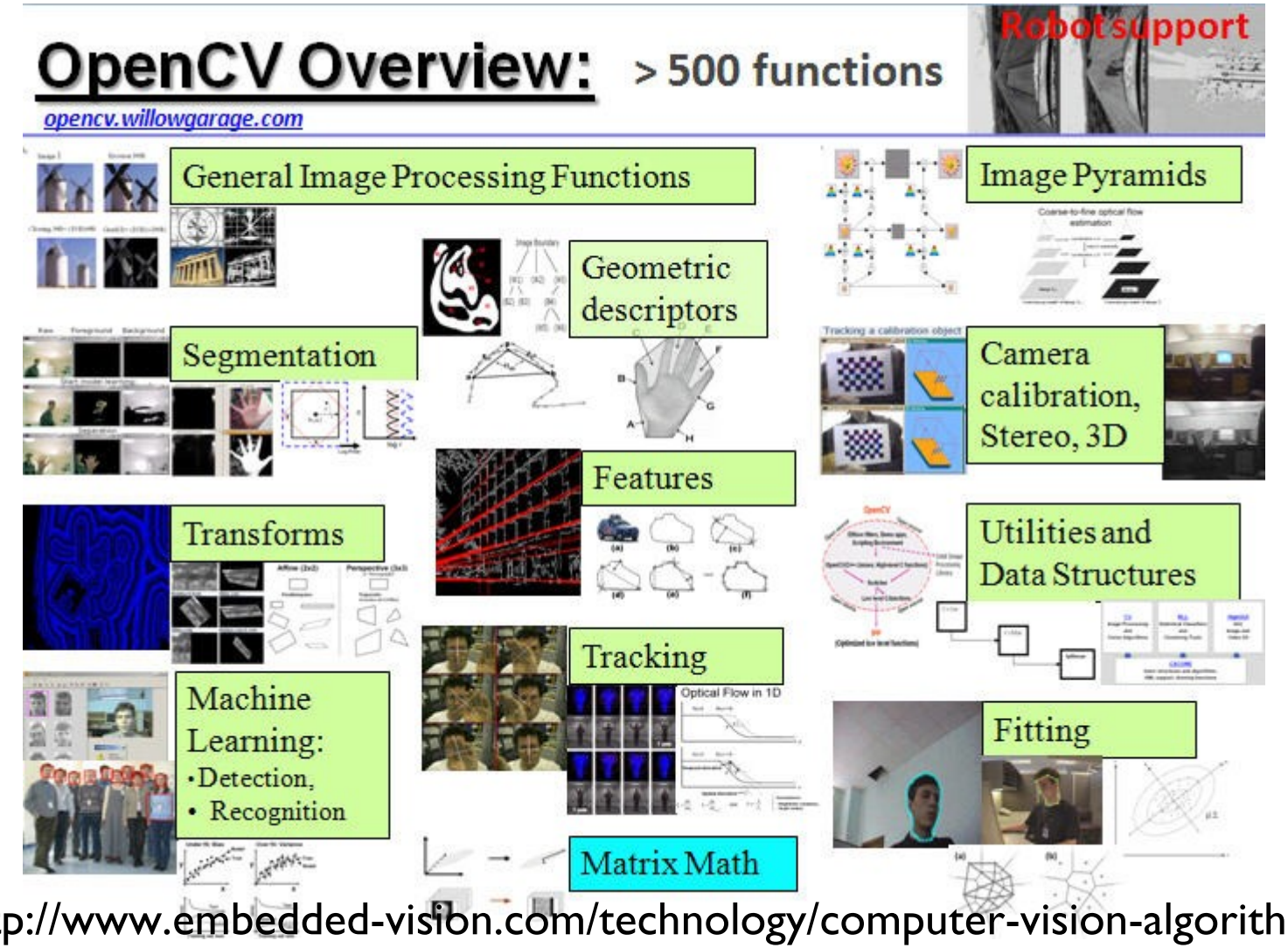
目的：數位影像處理入門

OpenCV

- Open Source Computer Vision Library



- 跨平台的計算機函式庫，主要由 C/C++ 撰寫

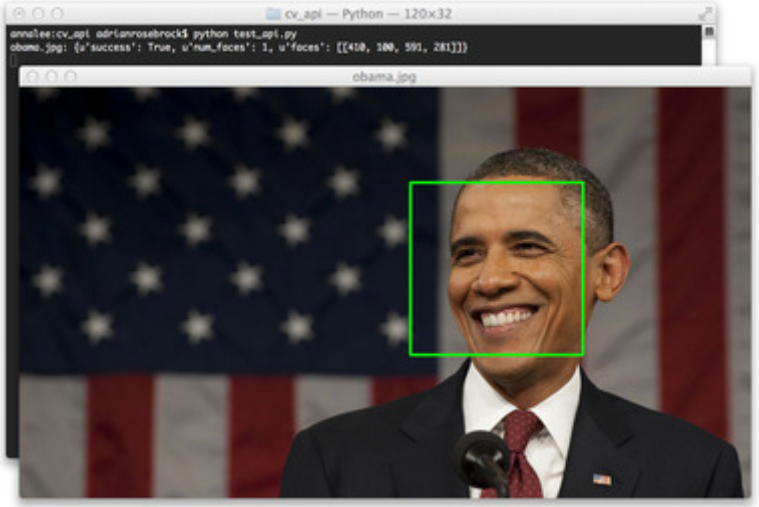


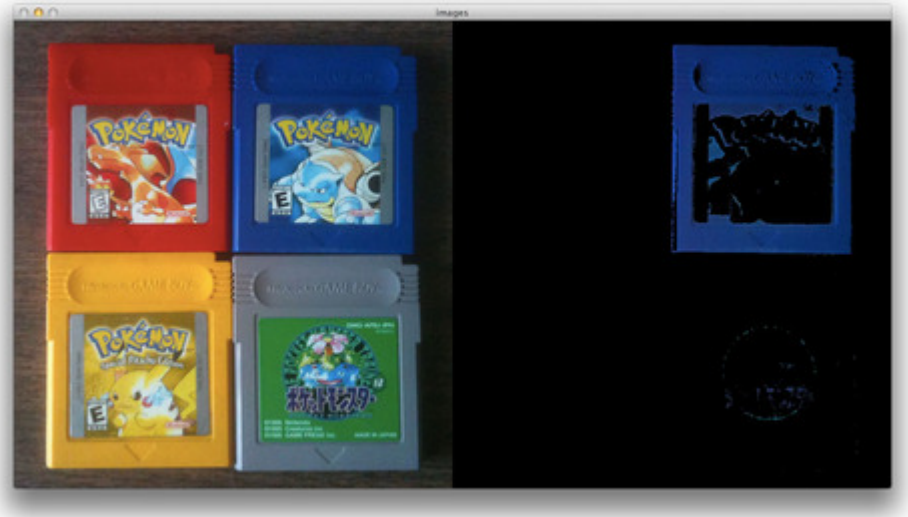
http://www.embedded-vision.com/technology/computer-vision-algorithms

電腦如何分辨東西？

- 顏色
- 特徵
- 學習

. . .

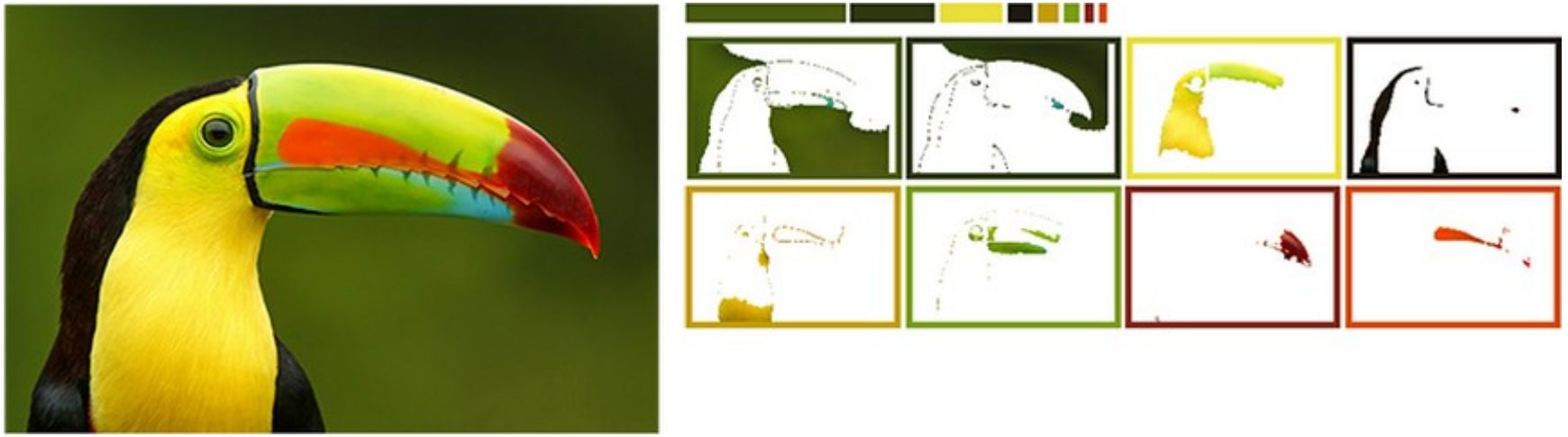




http://www.pyimagesearch.com/

影像處理的意義

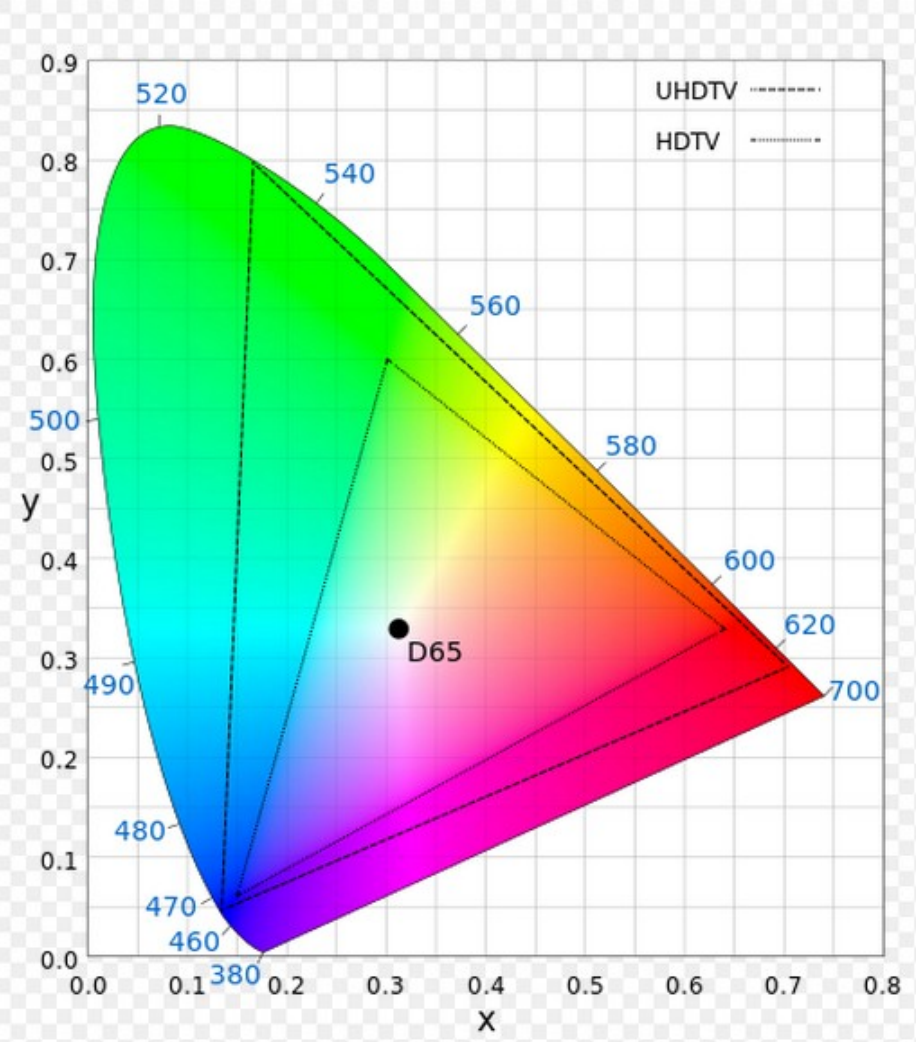
- 分離與萃取資訊
- 增強或平滑訊號
- 作為分群、特徵識別等應用的前處理



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

色彩空間 (Color Space)

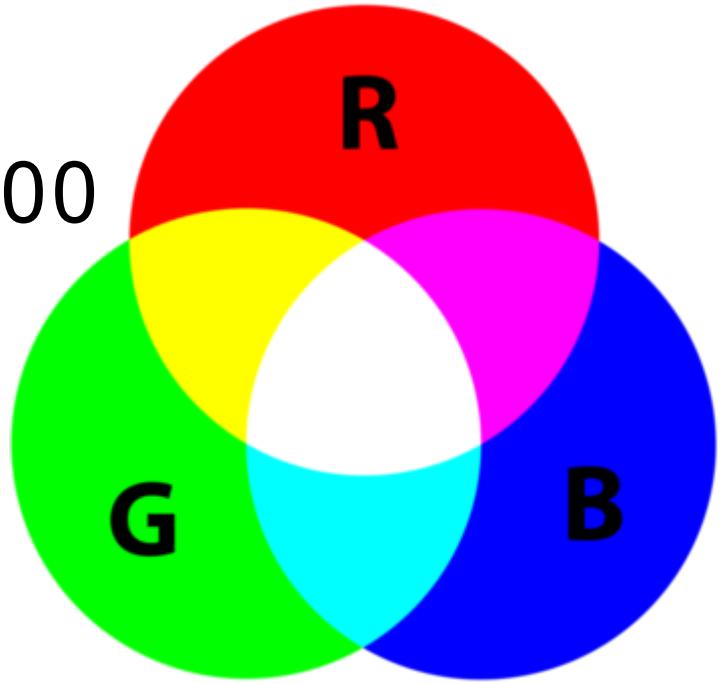
- RGB
- CMY(K)
- YUV(YCbCr)
- HSV



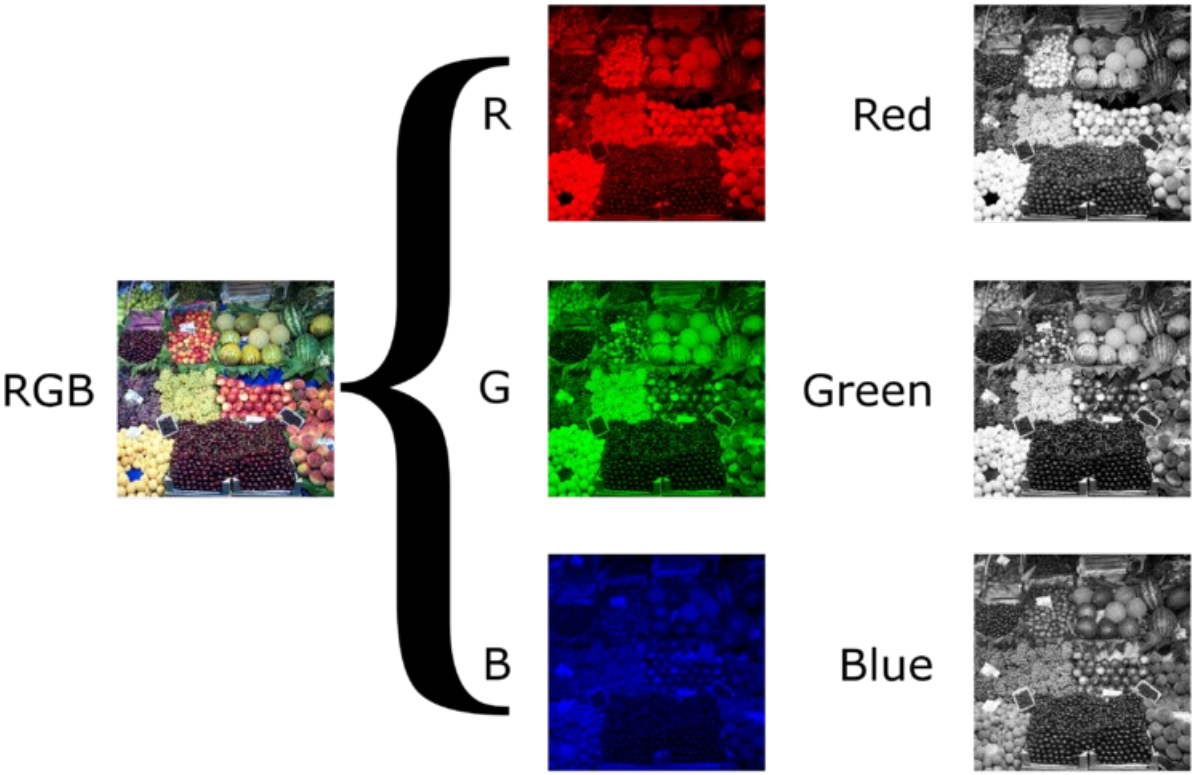
RGB

- R(Red), G(Green), B(Blue)

- R(紅),G(綠),B(藍)
- 光的三原色
- 疊加型混色的色彩模型
- 常用在電腦上表現色彩
  - 8-bit:(255,0,0),#FF0000



https://en.wikipedia.org/wiki/RGB_color_model



彩色，灰階，黑白

NUMBERS		
R 255 G 0 B 0	R 102 G 102 B 255	R 51 G 204 B 153
R 255 G 255 B 102	R 255 G 0 B 204	R 51 G 204 B 255
R 51 G 51 B 0	R 51 G 51 B 153	R 255 G 153 B 153

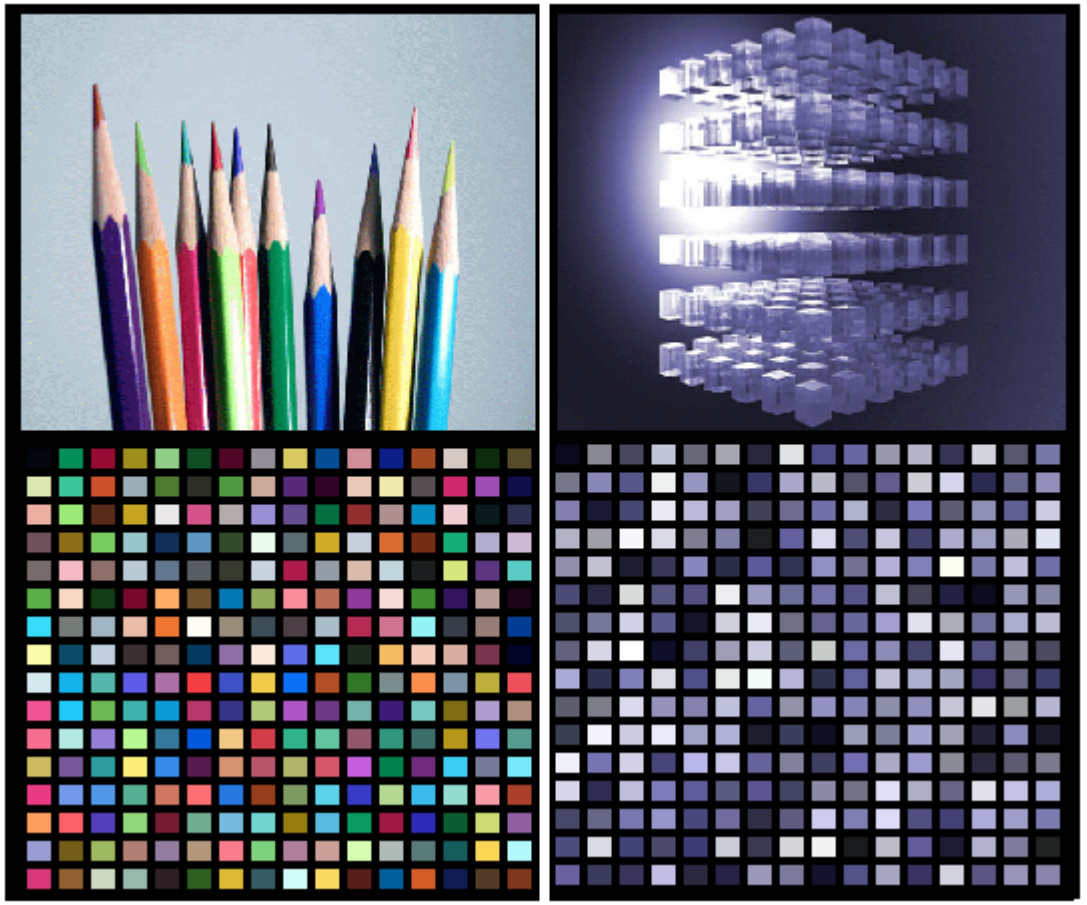
GRAY = 1 SET OF DIGITS		
11111111	11100110	11001101
10110100	10011011	01110011
01010000	00101000	00000000

BLACK & WHITE		
0	1	0
1	0	1
0	1	0

https://en.wikipedia.org/wiki/Grayscale

Indexed Color

- 從全彩 (True Color) 中挑選 256 個顏色

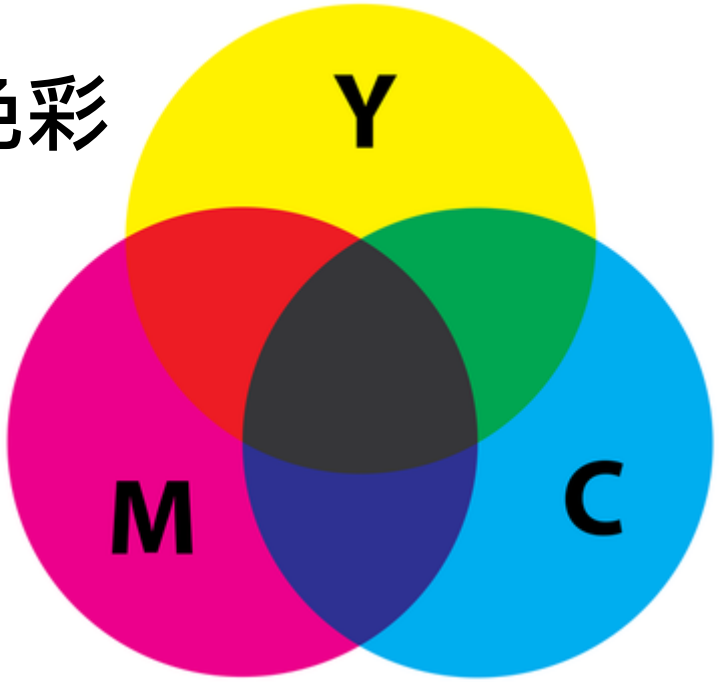


http://goo.gl/xijFPu

CMY(K)

- C(Cyan), M(Magenta), Y(Yellow), K(Black)

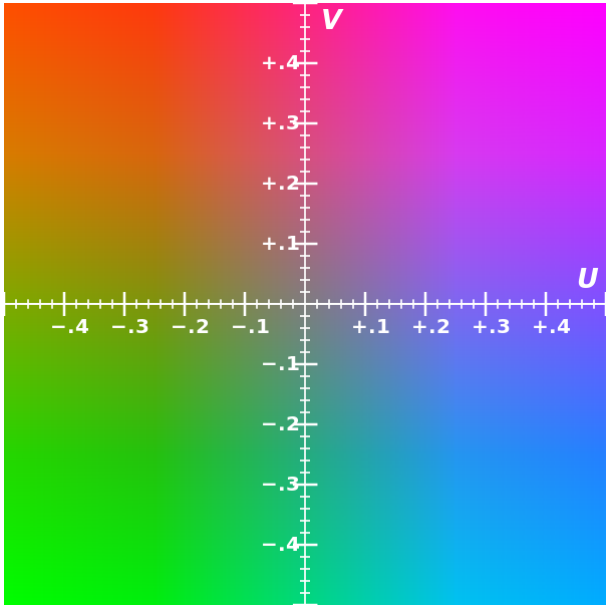
- C(青),M(紫),Y(黃),K(黑)
- 彩色墨水三原色
- 削減型混色的色彩模型
- 常用在印表機 / 影印機表現色彩
- 黑色 ,R=G=B=1



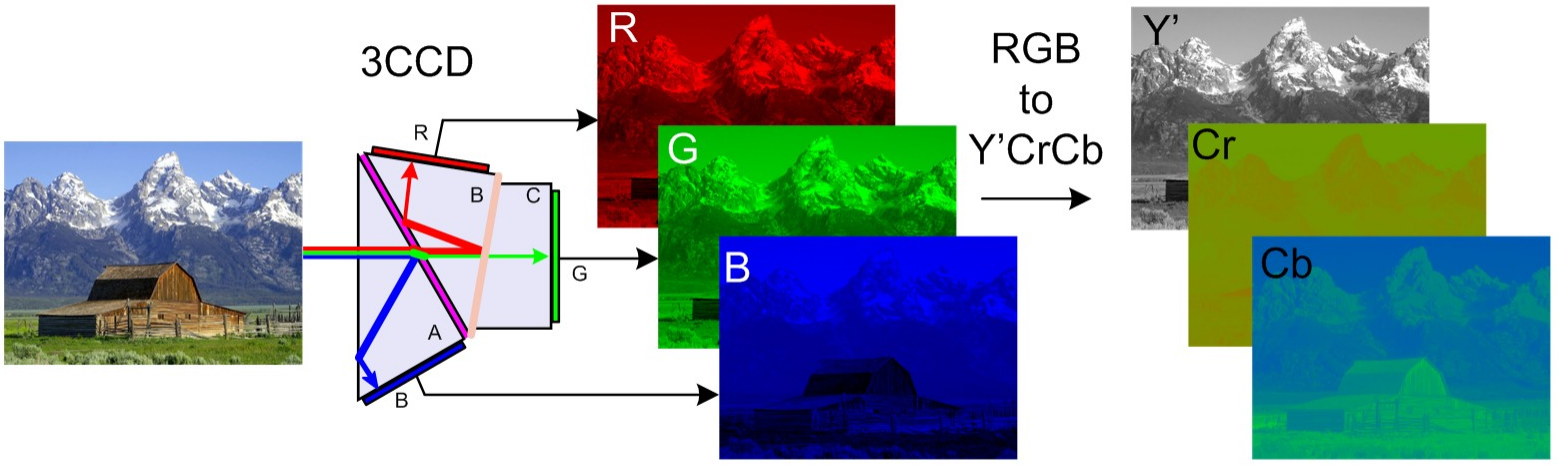
https://en.wikipedia.org/wiki/CMYK_color_model

YUV(YCbCr)

- Y(Luma), UV(Chroma)



- Y(亮度 / 輝度) , UV(色度)
- 常用在電視系統表現色彩
- 源自於 RGB 模型，表示色差訊號
  - YCbCr 是數位色差訊號
  - YPbPr 是類比色差訊號



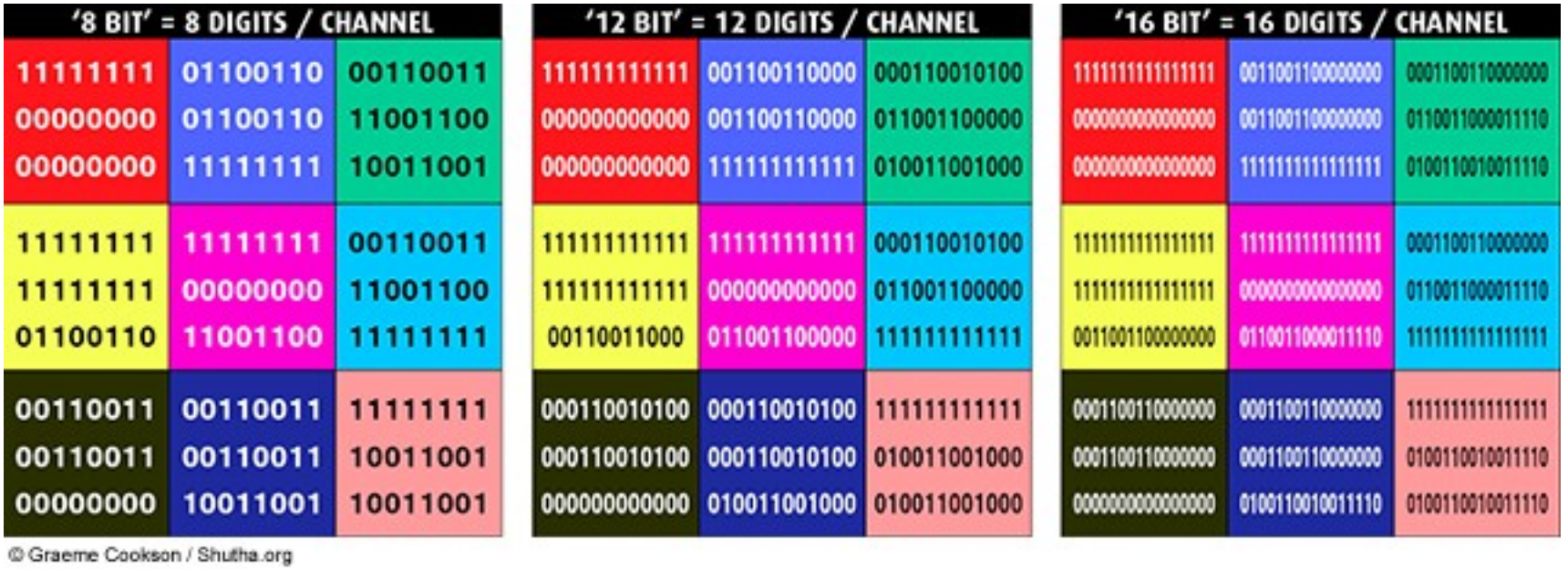
https://en.wikipedia.org/wiki/YUV

不同色彩空間儲存的資料量不相同

© Graeme Cookson / Shutha.org

http://www.shutha.org/node/789

同一色彩空間也有不同的色彩深度



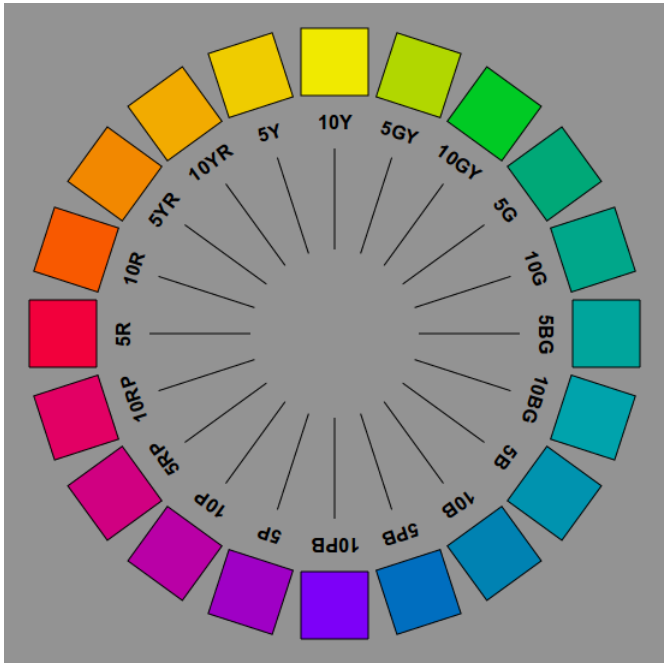
© Graeme Cookson / Shutha.org

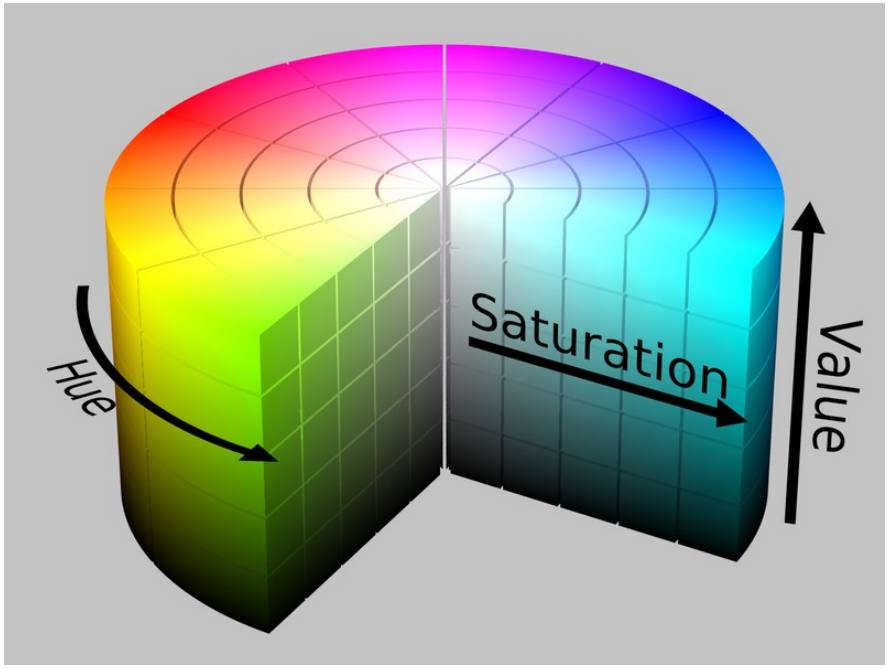
http://www.shutha.org/node/789

HSV

- H(Hue), S(Saturation), V(Value)

- H(彩度 ,0-179)
- S(飽和度 ,0-255)
- V(明度 ,0-255)
- 符合人對顏色的感知，常用在**數位影像處理**





https://en.wikipedia.org/wiki/HSL_and_HSV

色彩空間的轉換

- 以 RGB 為中心
  - RGB to CMY

$$\begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} 1-R \\ 1-G \\ 1-B \end{pmatrix}$$

  - CMY to RGB

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1-C \\ 1-M \\ 1-Y \end{pmatrix}$$

  - RGB to YUV

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.14713 & -0.28886 & 0.436 \\ 0.615 & -0.51499 & -0.10001 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

http://isis-data.science.uva.nl/koelma/horus/dox/horus2.0/refcpp/html/HxColConvert_8h.html

色彩空間的轉換

- 以 RGB 為中心
  - HSV to RGB

$$C = V \times S$$

$$X = C \times (1 - |(H / 60^\circ) \bmod 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0^\circ \le H < 60^\circ \\ (X, C, 0) & , 60^\circ \le H < 120^\circ \\ (0, C, X) & , 120^\circ \le H < 180^\circ \\ (0, X, C) & , 180^\circ \le H < 240^\circ \\ (X, 0, C) & , 240^\circ \le H < 300^\circ \\ (C, 0, X) & , 300^\circ \le H < 360^\circ \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

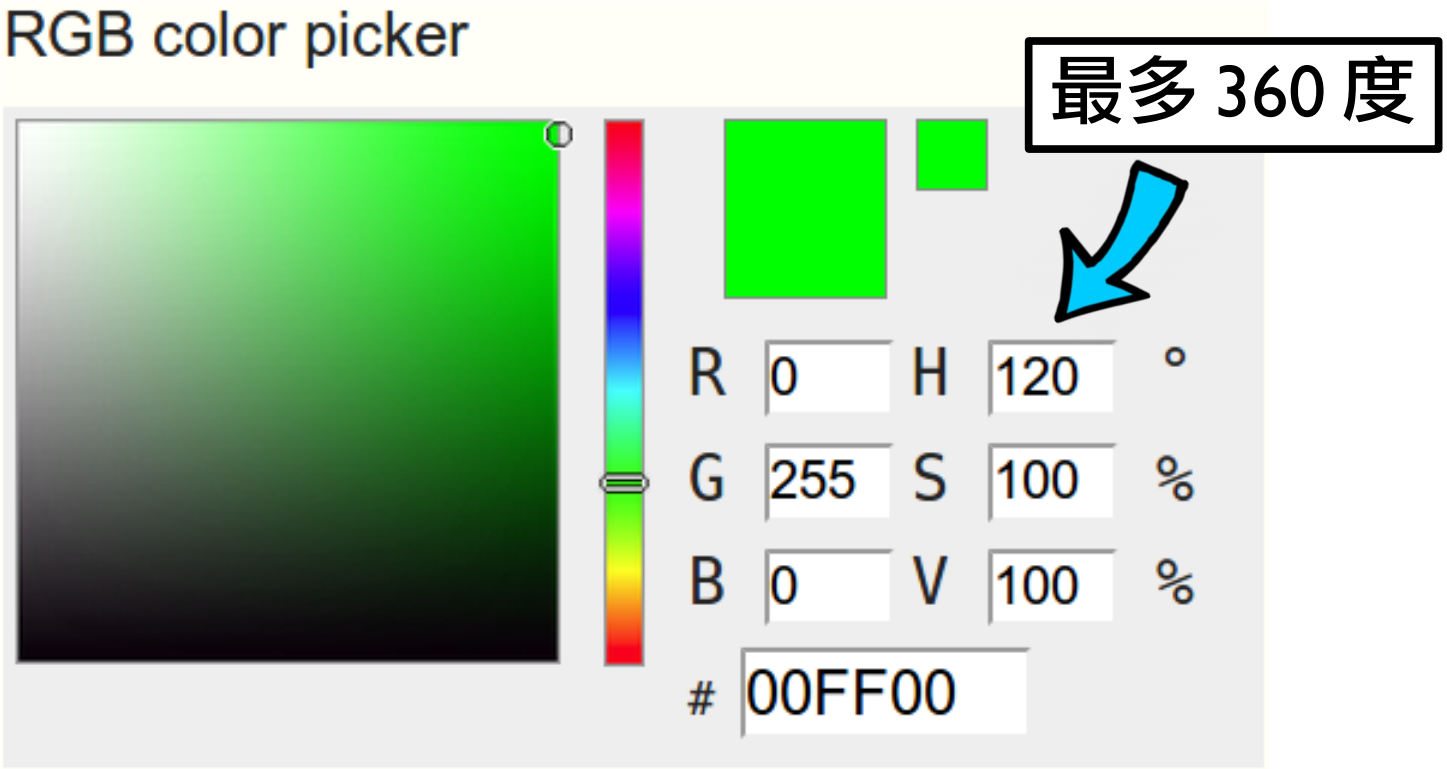
Color	Color name	(H,S,V)	Hex	(R,G,B)
	Black	(0°,0%,0%)	#000000	(0,0,0)
	White	(0°,0%,100%)	#FFFFFF	(255,255,255)
	Red	(0°,100%,100%)	#FF0000	(255,0,0)
	Lime	(120°,100%,100%)	#00FF00	(0,255,0)
	Blue	(240°,100%,100%)	#0000FF	(0,0,255)
	Yellow	(60°,100%,100%)	#FFFF00	(255,255,0)
	Cyan	(180°,100%,100%)	#00FFFF	(0,255,255)
	Magenta	(300°,100%,100%)	#FF00FF	(255,0,255)
	Silver	(0°,0%,75%)	#C0C0C0	(192,192,192)
	Gray	(0°,0%,50%)	#808080	(128,128,128)
	Maroon	(0°,100%,50%)	#800000	(128,0,0)
	Olive	(60°,100%,50%)	#808000	(128,128,0)
	Green	(120°,100%,50%)	#008000	(0,128,0)
	Purple	(300°,100%,50%)	#800080	(128,0,128)
	Teal	(180°,100%,50%)	#008080	(0,128,128)
	Navy	(240°,100%,50%)	#000080	(0,0,128)

http://www.rapidtables.com/convert/color/hsv-to-rgb.htm

線上轉換工具

- http://goo.gl/EV4l0z

- 將 RGB 的綠色轉成 HSV



除了看圖也看值

- 在 Python 中如何看 RGB 的綠色轉成 HSV 是多少？
- $ python

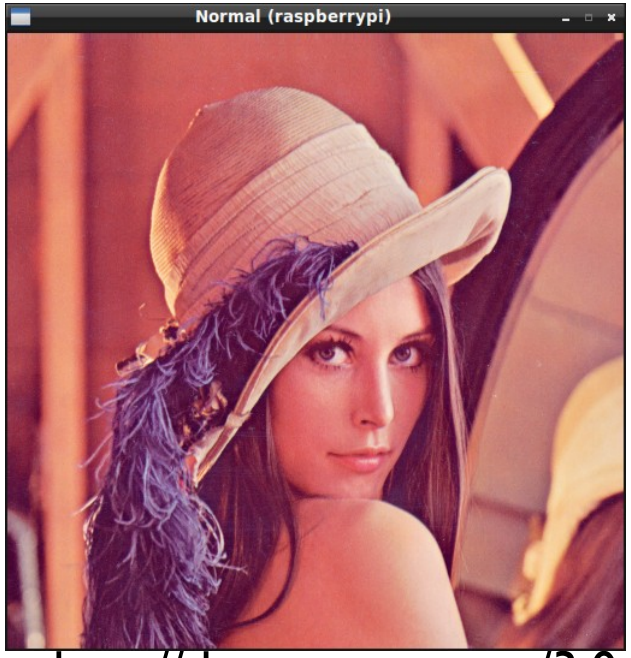
```
>>> import cv2
>>> import numpy as np
>>> green = np.array([[[0,255,0]]], dtype='uint8')
>>> hsv_green = cv2.cvtColor(green, cv2.COLOR_BGR2HSV)
>>> print hsv_green
[[[ 60 255 255]]]
```

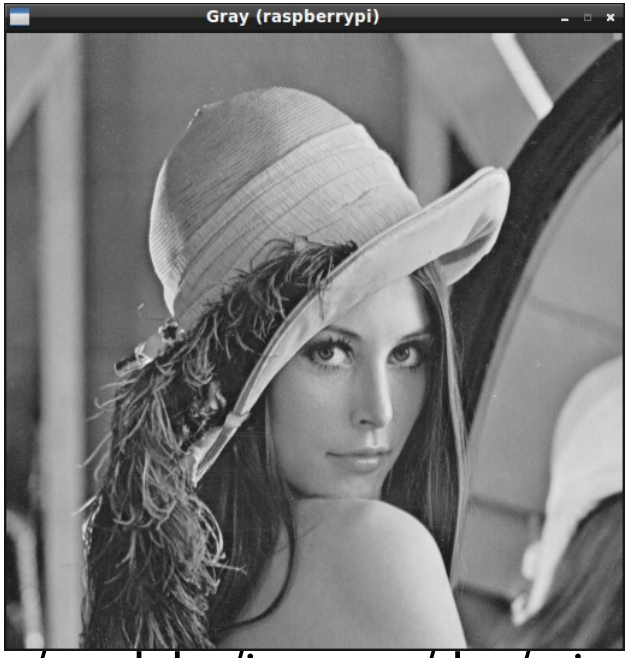
0-180

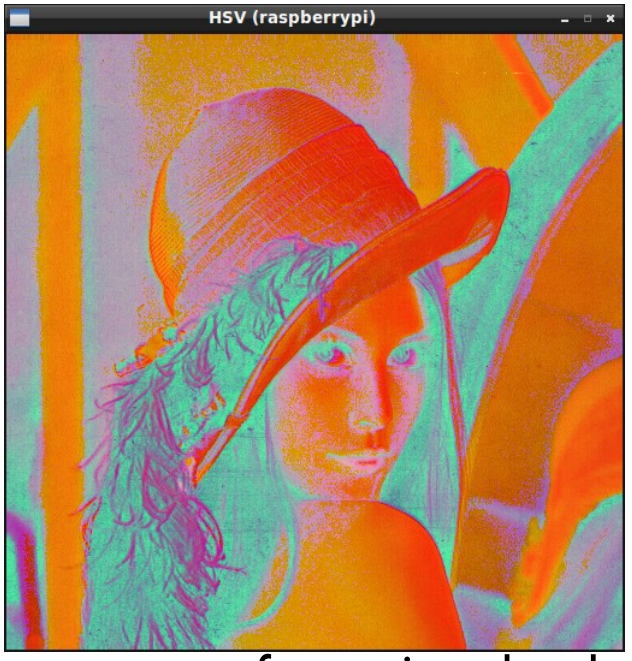
http://docs.opencv.org/3.2.0/df/d9d/tutorial_py_colorspaces.html

色彩空間的轉換

- `cv2.cvtColor(img,flag)`
  - RGB 轉灰階 ,`flag = cv2.COLOR_BGR2GRAY`
  - RGB 轉 HSV,`flag = cv2.COLOR_BGR2HSV`
  - HSV 轉 RGB,`flag = cv2.COLOR_HSV2BGR`







http://docs.opencv.org/3.0-beta/modules/imgproc/doc/miscellaneous_transformations.html

色彩空間轉換

```
image = cv2.imread(“lena256rgb.jpg”)
cv2.imshow("Normal", image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
cv2.imshow("HSV", hsv)

bgr = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
cv2.imshow("BGR", bgr)
```

BGR 轉灰階

BGR 轉 HSV

HSV 轉 BGR

影像出現後，按任意鍵會顯示下個影像

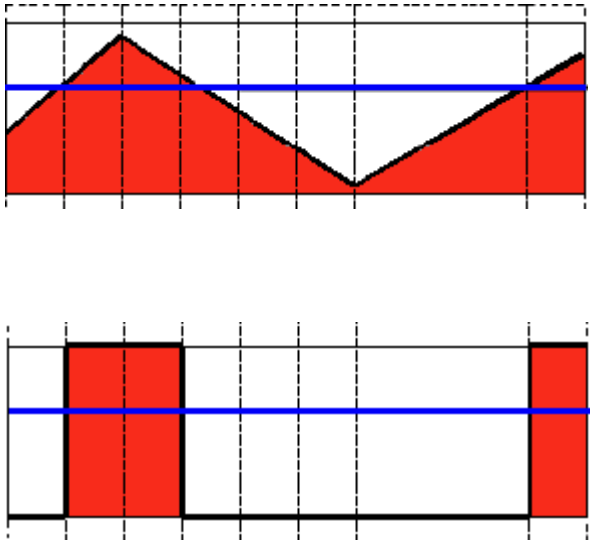
DEMO
color_space.py

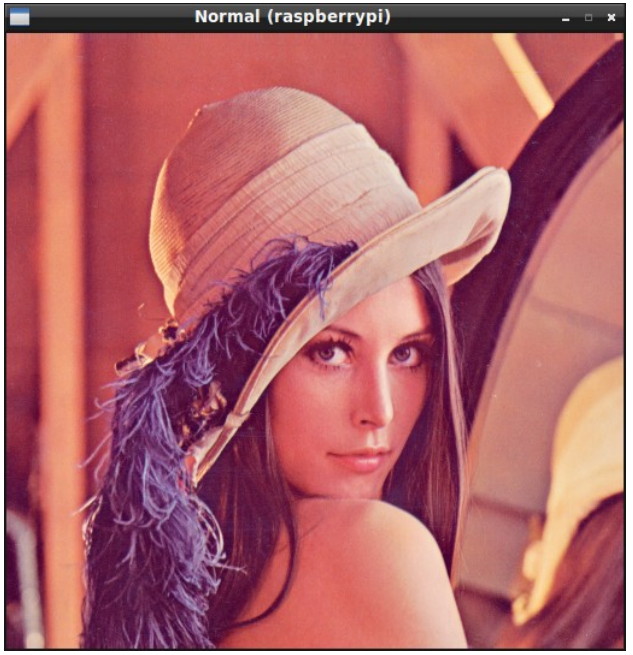
```
$ cd ~/cam-py-cv/day2/01-color_space
$ python color_space.py
```

二值化

- 將影像以強度 (threshold) 區分

- `cv2.threshold(img,thresh,maxval,type)`
- 從灰階轉成黑白，會有兩個回傳值
  - `cv2.threshold(gray,127,255,cv2.THRESH_BINARY)`

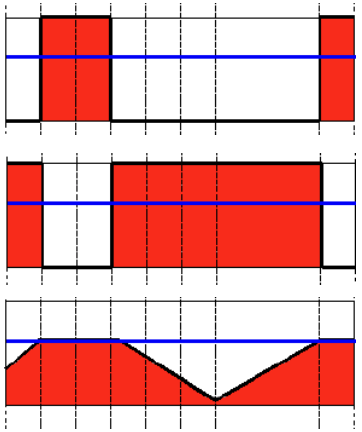


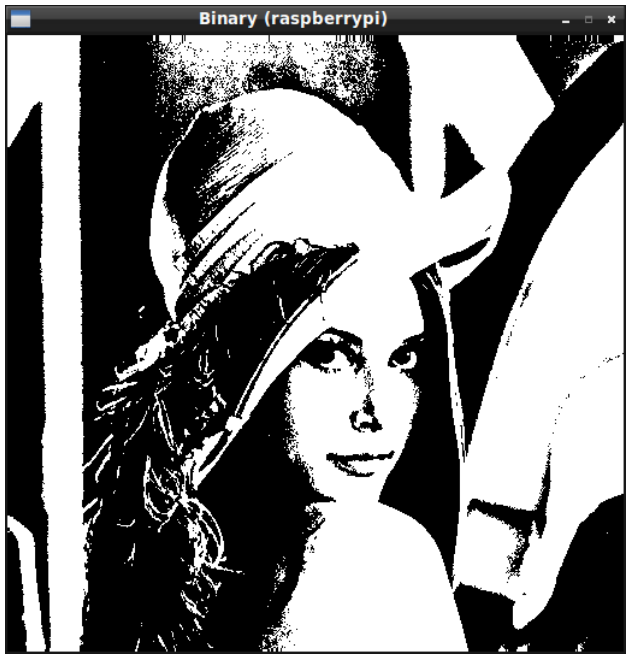


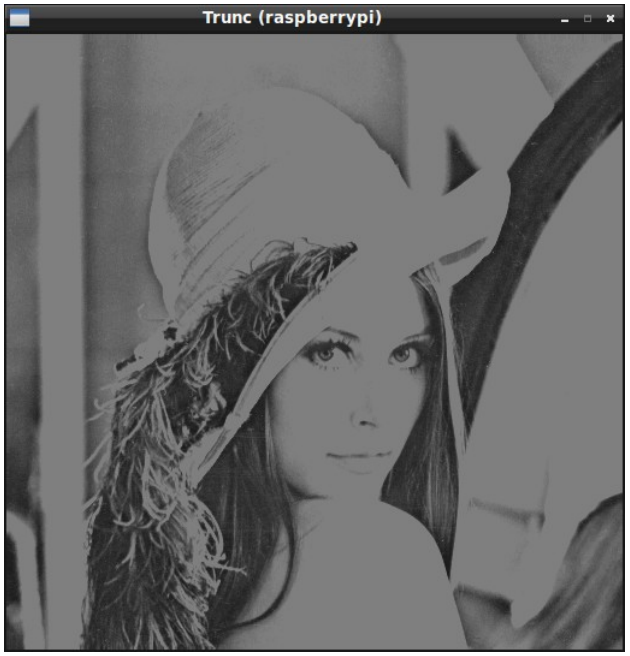
http://docs.opencv.org/3.0-beta/modules/imgproc/doc/miscellaneous_transformations.html

RGB 二值化的各種參數效果

- 轉成黑白，cv2.THRESH_BINARY
- 反向轉換，cv2.THRESH_BINARY_INV
- 超過刪除，cv2.THRESH_TRUNC







http://docs.opencv.org/trunk/d7/d4d/tutorial_py_thresholding.html

練習

- 修改 color_space.py 將灰階影像轉為二值化影像

```
# Convert BGR to Gray
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2.imshow("Gray", gray)
print "Gray image..."
cv2.waitKey(0)

# Threshold the gray image to binary image
# ret, binary = cv2.threshold(...)
```

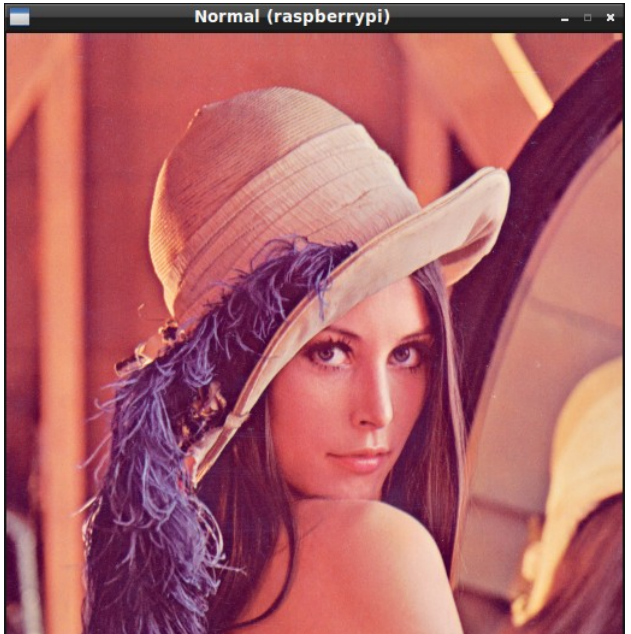
兩個回傳值

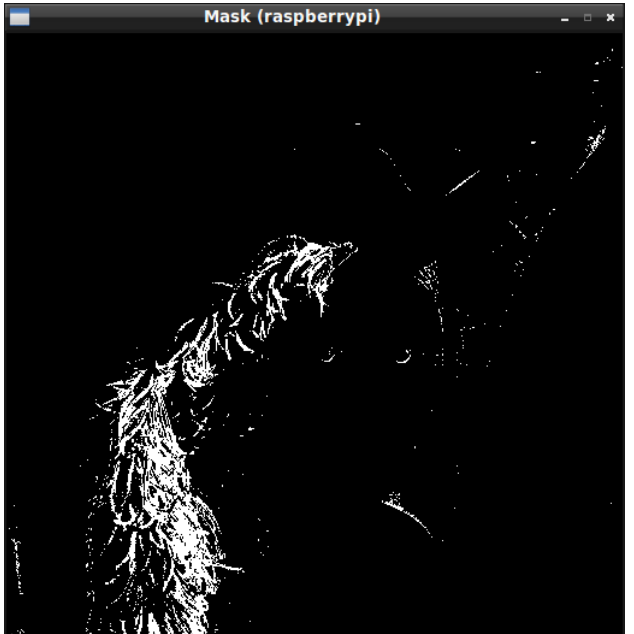
請完成這邊

另一種二值化

- 在 HSV 空間裡，根據區間 (lower/upper) 分割

- `cv2.inRange(img,lowerval,upperval)`
- 範例：只取出紫色部份
  - `lower=np.array([141,0,0])`
  - `upper=np.array([164,145,197])`
  - `binary=cv2.inRange(hsv,lower,upper)`





色彩空間轉換與二值化處理

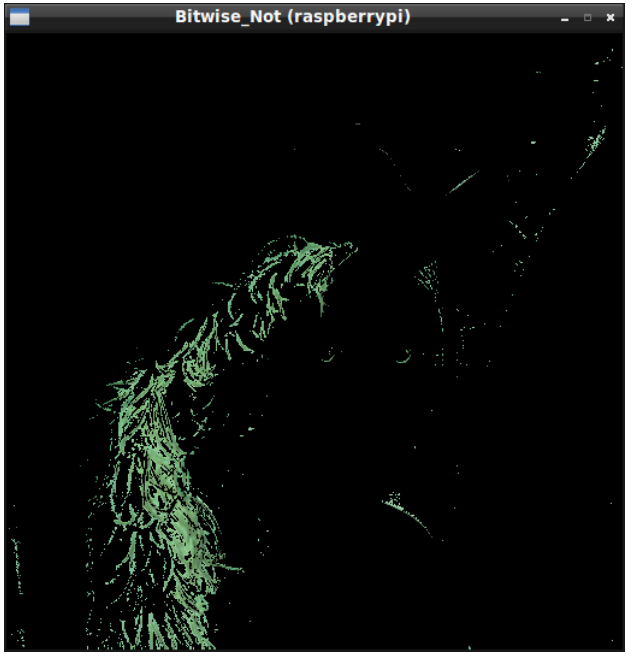
```
image = cv2.imread("lena256rgb.jpg")

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
lower = np.array( [141, 0, 0] )
upper = np.array( [164, 145, 197] )
binary = cv2.inRange(hsv, lower, upper)

cv2.imshow("Binary", binary)
print "HSV Binary image..."
cv2.waitKey(0)
```

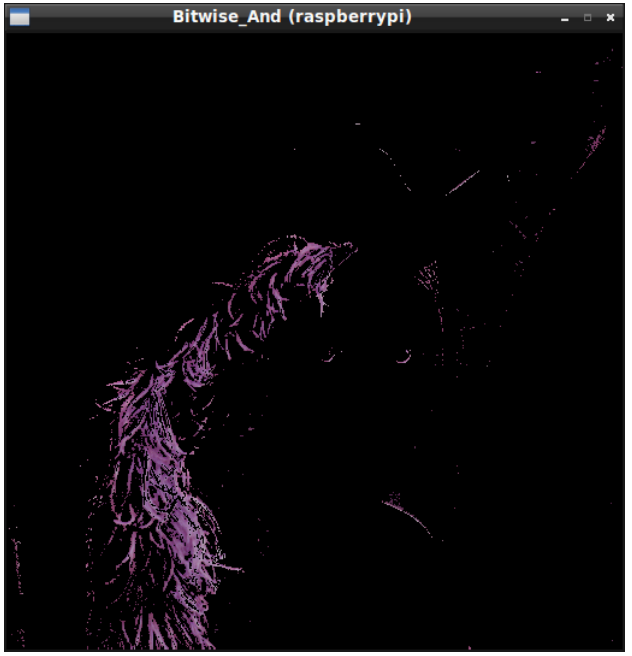
DEMO
hsv_binary.py

```
$ cd ~/cam-py-cv/day2/01-color_space
$ python hsv_binary.py
```

位元運算處理 (Bitwise)

- `cv2.bitwise_not(mask)  # 反向 mask 結果`
- `cv2.bitwise_not(src, mask)`
- `cv2.bitwise_and(src, dst, mask)`





http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_image_arithmetics/py_image_arithmetics.html

原圖與遮罩相疊

```
image = cv2.imread("lena256rgb.jpg")

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
lower = np.array([141, 0, 0])
upper = np.array([164, 145, 197])
binary = cv2.inRange(hsv, lower, upper)

bitwise_not = cv2.bitwise_not(binary)
cv2.imshow("Bitwise_not", bitwise_not)

bitwise_not = cv2.bitwise_not(image, mask=binary)
cv2.imshow("Bitwise_not", bitwise_not)
```

src dst mask

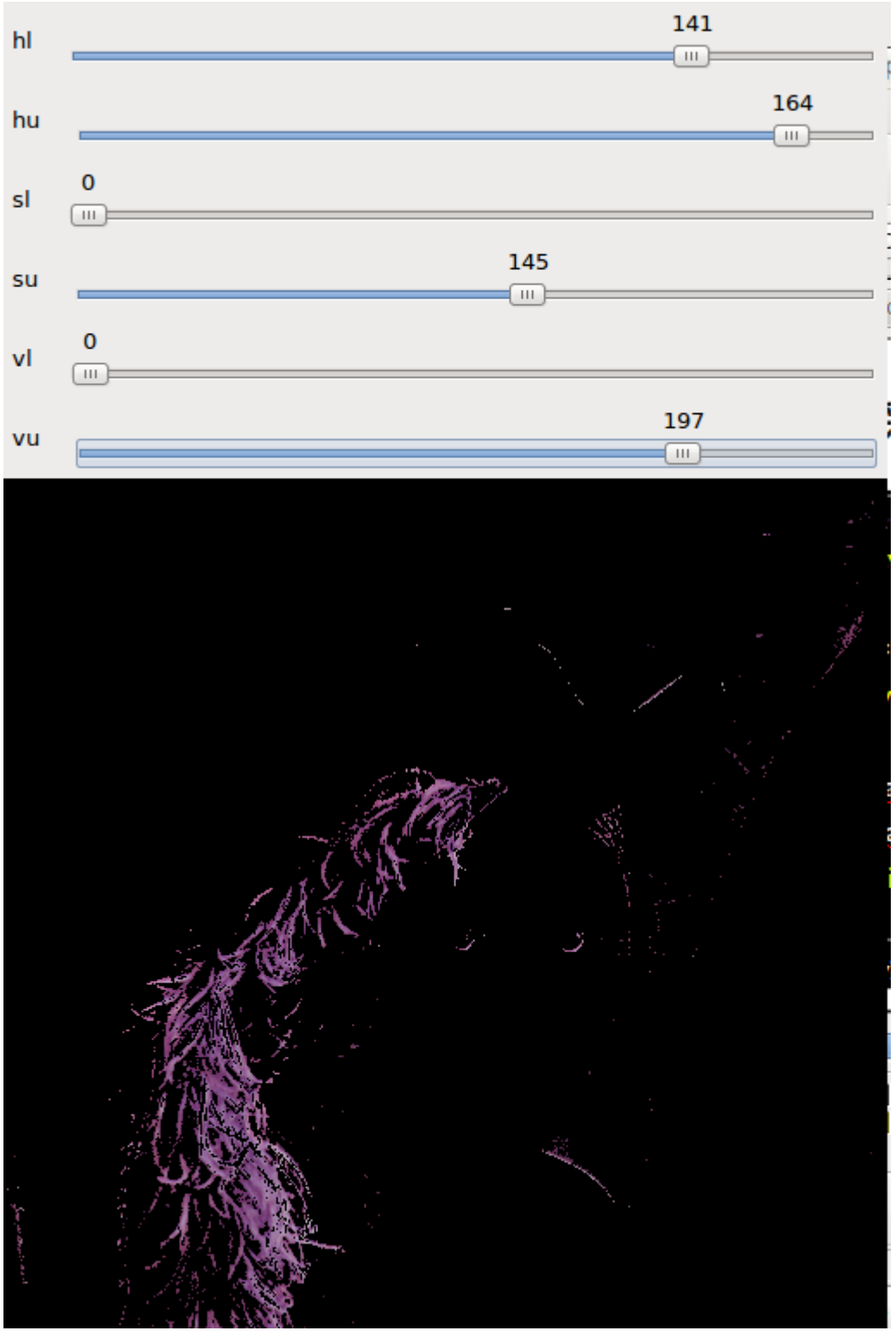
```
bitwise_and = cv2.bitwise_and(image, image, mask=binary)
cv2.imshow("Bitwise_and", bitwise_and)
```

DEMO
hsv_mask.py

```
$ cd ~/cam-py-cv/day2/01-color_space
$ python hsv_mask.py
```

HSV 的值？

- 即時調整



建立拉桿與顯示即時結果

```
cv2.namedWindow('hsv_demo')
h, s, v = 100, 100, 100
cv2.createTrackbar('hl', 'hsv_demo', 0,   179, nothing)
cv2.createTrackbar('hu', 'hsv_demo', 179, 179, nothing)

while True:
    hl = cv2.getTrackbarPos('hl', 'hsv_demo')
    hu = cv2.getTrackbarPos('hu', 'hsv_demo')

    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower = np.array([hl, sl, vl])
    upper = np.array([hu, su, vu])
    mask = cv2.inRange(hsv, lower, upper)
    result = cv2.bitwise_and(image, image, mask=mask)

    cv2.imshow("hsv_demo", result)
```

建立拉桿

讀取拉桿數值

DEMO
hsv_value.py

```
$ cd ~/cam-py-cv/day2/01-color_space
$ python hsv_value.py
```

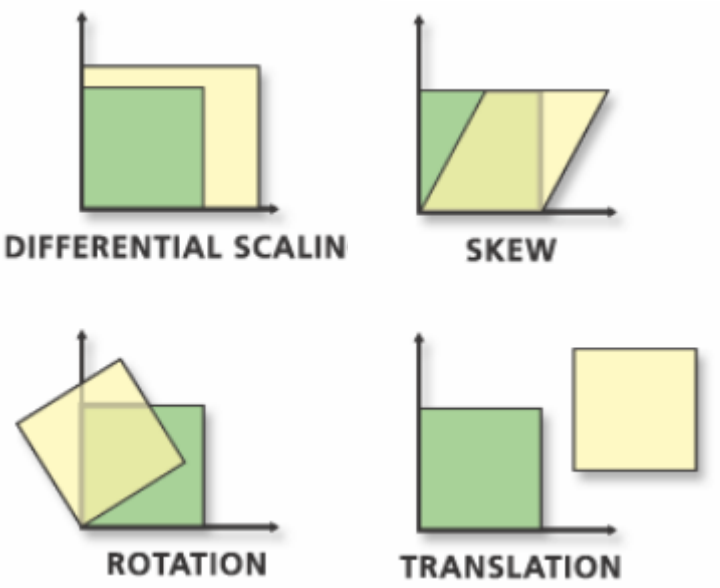
練習

- 執行以下指令，找出黃色，並紀錄 HSV 的 upper 和 lower
- `$ python hsv_value.py balloon.jpg`

https://pixabay.com/en/balloon-year-market-bloat-2216318/

2D Transformation

- 一個 2D 物件的轉換 (transformation) 包括
  - Position(translation)
  - Size(scaling)
  - Orientation(rotation)
  - Shapes(shear)



- 可用公式表示為 $\texttt{dst}(x,y) = \texttt{src}(f_x(x,y), f_y(x,y))$

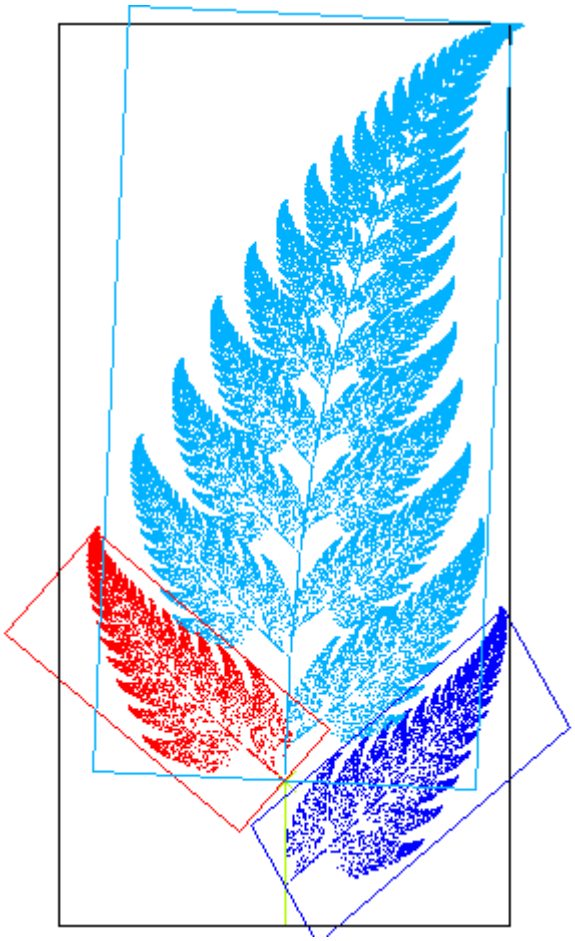
仿射變換 (Affine Transformation)

- 在幾何中，一個向量空間進行一次線性變換並接上一個平移，變換為另一個向量空間

- 為平移及線性映射的複合函數

$$\vec{y} = f(\vec{x}) = A\vec{x} + \vec{b}.$$

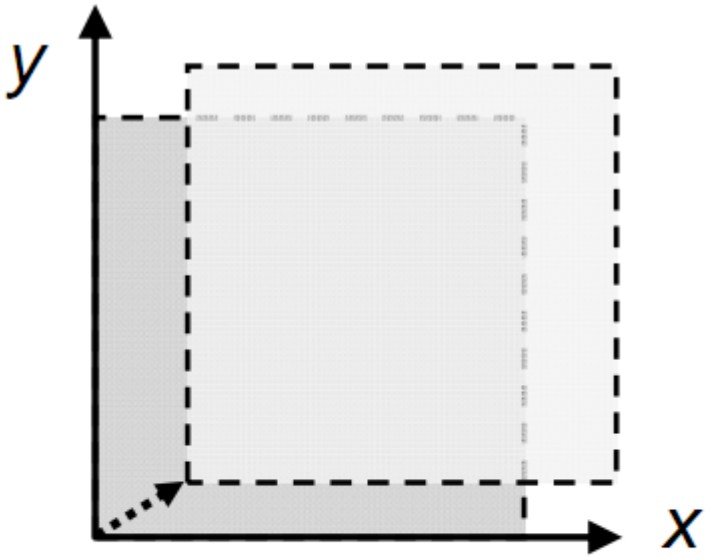
變化矩陣　平移



https://en.wikipedia.org/wiki/Affine_transformation

平移 (Translation)

- 將座標 (X,Y) 以位移向量 (tx,ty) 平移至 (X′,Y′)
  - X′ = X + tx
  - Y′ = Y + ty



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

https://ip.csie.ncu.edu.tw/course/IP/IP1402cp.pdf

矩陣表示式

- 將原本的轉換矩陣改為齊次座標

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} tx \\ ty \end{bmatrix}$$

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

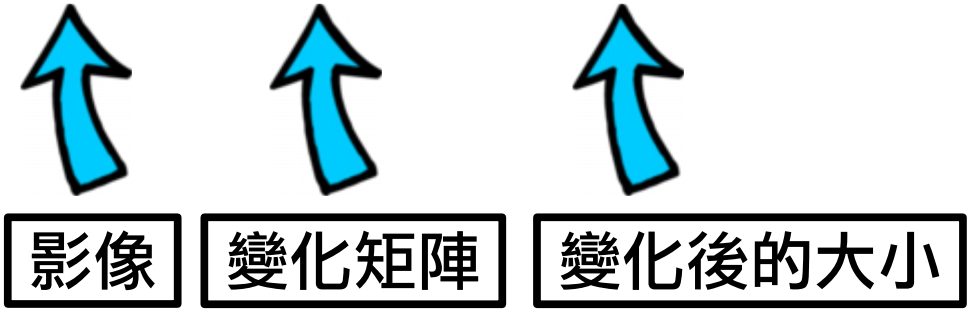
Use 3 x 1 vector

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & tx \\ 0 & 1 & ty \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

http://slideplayer.com/slide/10741621/

仿射變換

- `cv2.warpAffine(src, M, dsize)`



- 如果變換後的影像大小和輸入的不同，會用內插法自動調整像素間關係

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

將位移 (100, 50) 傳到 cv2.warpAffine()

```
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)
```

tx=100, ty=50

$$M = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \end{bmatrix}$$

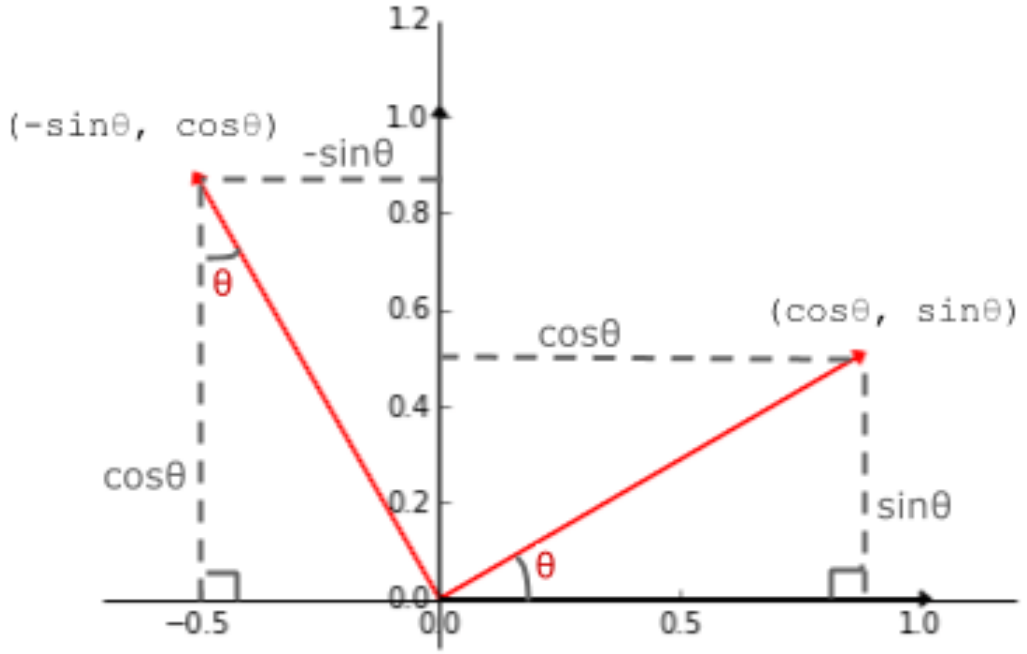
仿射矩陣

DEMO
translation.py

```
$ cd ~/cam-py-cv/day2/01-color_space
$ python translation.py
```

旋轉 (Rotation)

- 將影像旋轉 θ 角度，並且逆時針旋轉為正

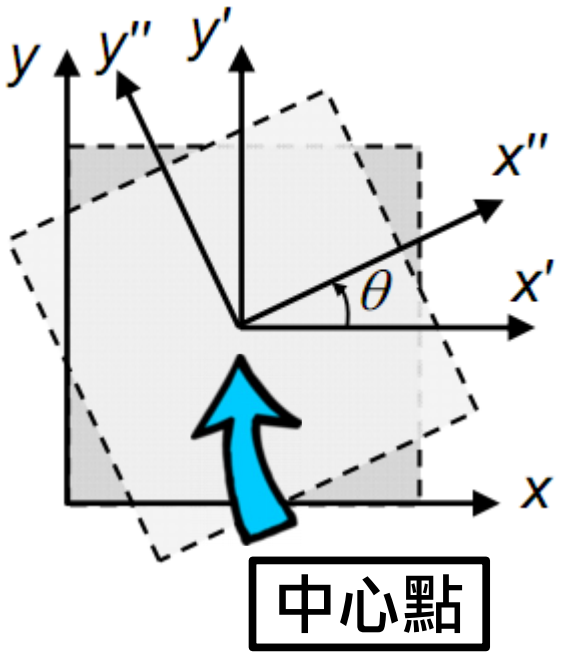


$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

https://alyssaq.github.io/2015/visualising-matrices-and-affine-transformations-with-python/

OpenCV 的旋轉矩陣

- 為了能在任意位置進行旋轉變換和縮放(scale)



$$M = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix}$$

$$M = \begin{bmatrix} \alpha & -\beta & (1-\alpha)center_x - \beta center_y \\ -\beta & \alpha & \beta center_x + (1-\alpha)center_y \end{bmatrix}$$

$$\alpha = scale * cos\theta \text{ and } \beta = scale * sin\theta$$

https://ip.csie.ncu.edu.tw/course/IP/IP1402cp.pdf

建立旋轉矩陣後傳到 cv2.warpAffine()

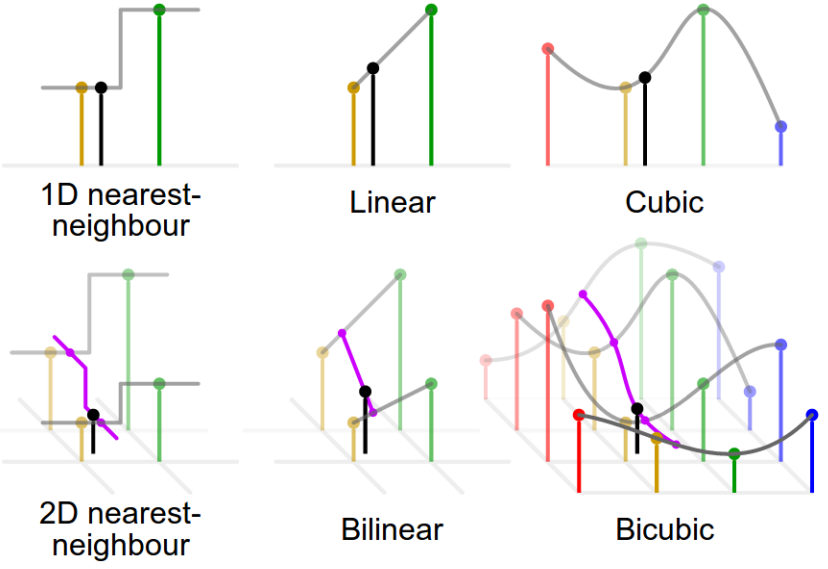
```
import cv2
import numpy as np

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)
```

DEMO
rotation.py

```
$ cd ~/cam-py-cv/day2/01-color_space
$ python rotation.py
```

縮放 (Resize)

- 透過內插法 (interpolation) 建立函數點
  - INTER_NEAREST: 鄰近內插
  - INTER_LINEAR: 線性內插 (預設)
  - INTER_AREA: 像素關係重採樣法 (縮小影像適用)
  - INTER_CUBIC: 三次內插 (放大影像適用)
  - INTER_LANCZOS4: lanczos4 內插



https://en.wikipedia.org/wiki/Linear_interpolation

三次內插適合放大影像時使用

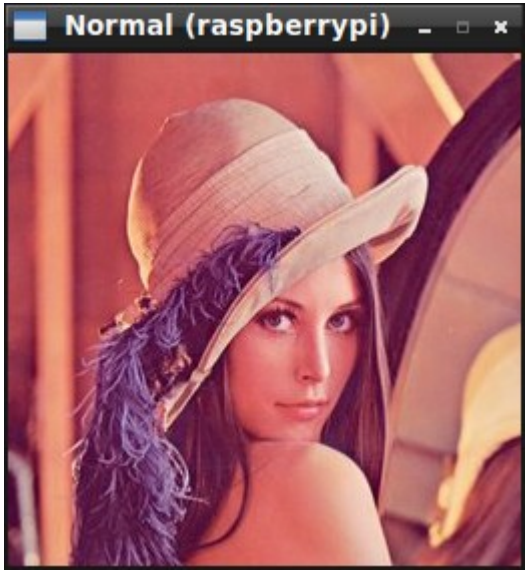
```
import cv2
import numpy as np

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)
```

DEMO
resize.py

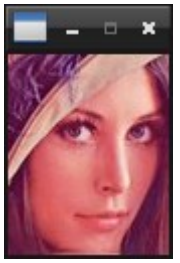
```
$ cd ~/cam-py-cv/day2/01-color_space
$ python resize.py
```

裁切 (Cropping)

- 選擇有興趣的部份 (Region of Interest, ROI)
- 可直接使用 numpy 的陣列切片取得結果



原圖



臉

身體

numpy 陣列切片

```
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)
```

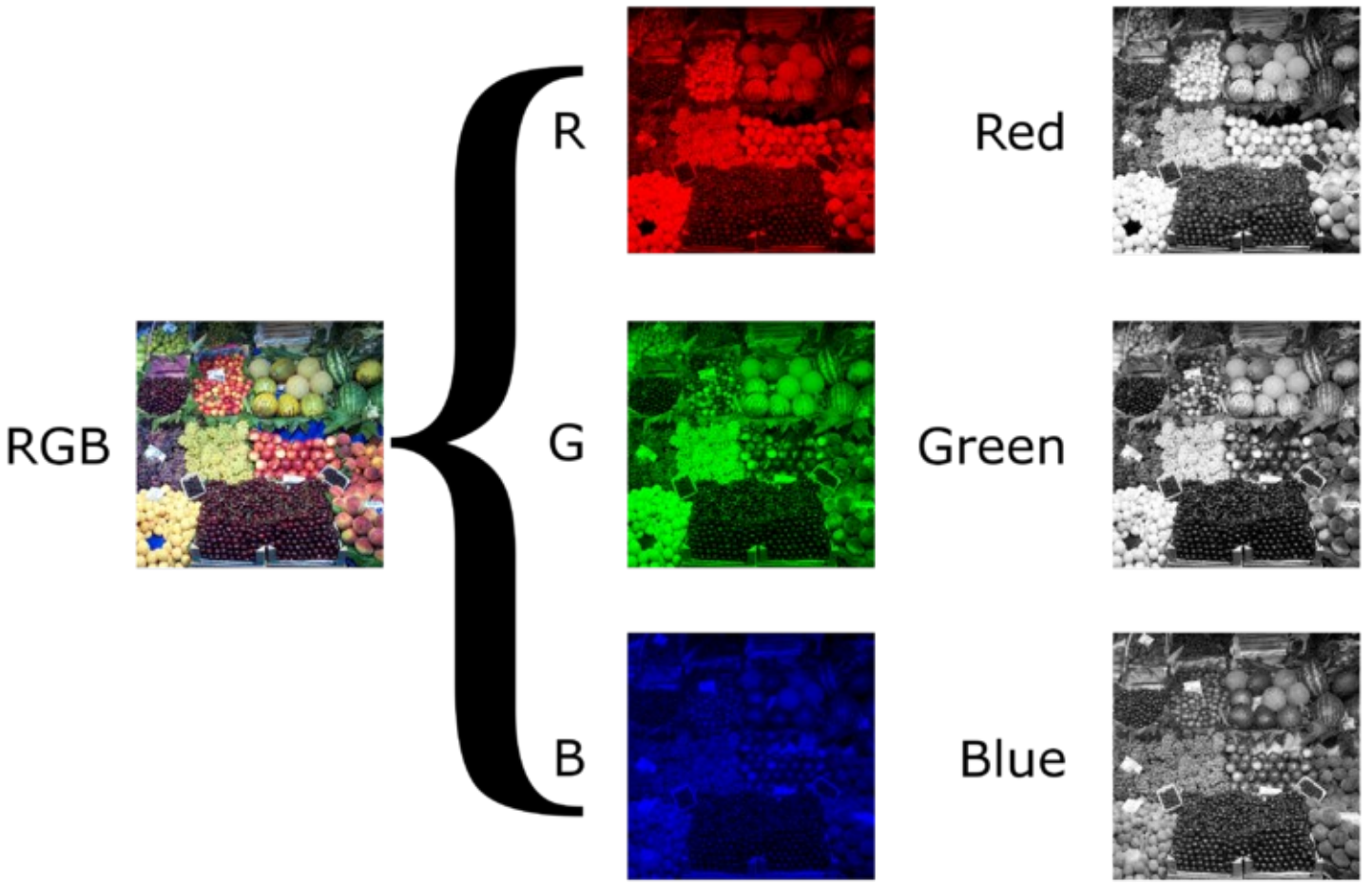
DEMO
crop.py

```
$ cd ~/cam-py-cv/day2/01-color_space
$ python crop.py
```

實驗 2：常用影像處理

目的：數位影像處理方法

單一通道是沒有顏色的

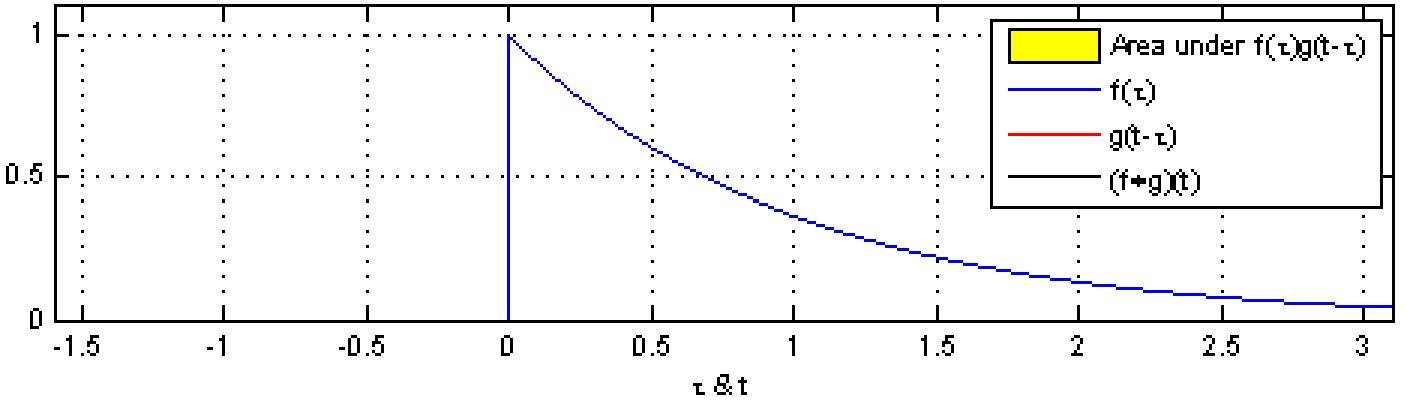


https://en.wikipedia.org/wiki/Grayscale

Convolution(卷積)

- 通過兩個函數 f 和 g 生成第三個函數的運算
- 是 f 與經過翻轉和平移的 g 的乘積函數圍成的面積
- 數學定義為：

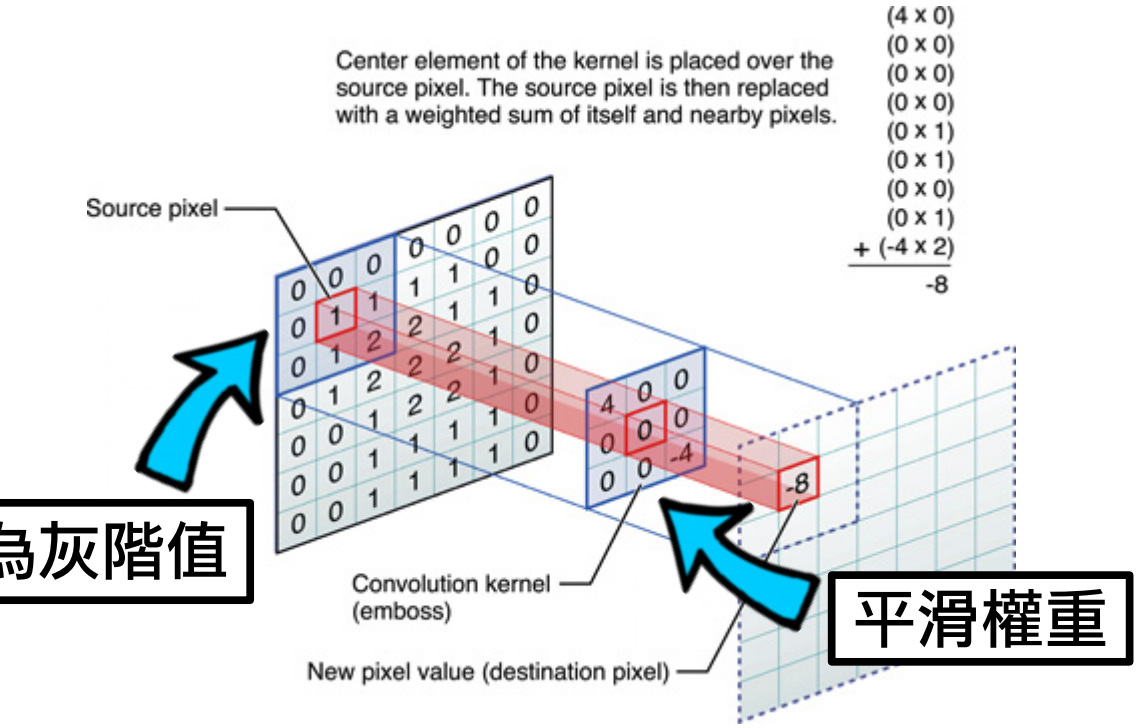
$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)\, g(t-\tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} f(t-\tau)\, g(\tau)\, d\tau.$$



https://en.wikipedia.org/wiki/Convolution

Convolution 在影像處理的應用

- 平滑的目的是為了去除雜訊，但對比度可能下降
- 每次要處理的像素區域稱為 Kernel



kernel，數字為灰階值

平滑權重

http://goo.gl/dck6tD

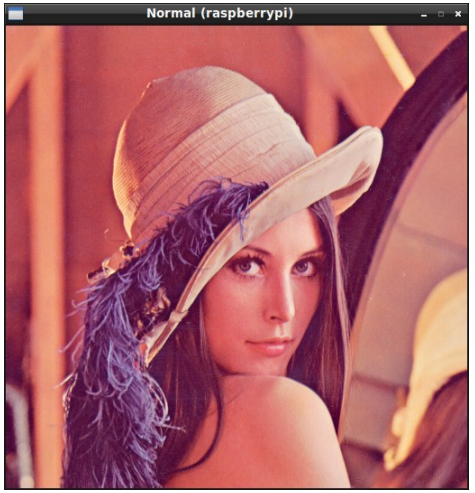
影像平滑方法

- 濾波器 (平滑化)
- 侵蝕 (Erode)
- 膨脹 (Dilate)

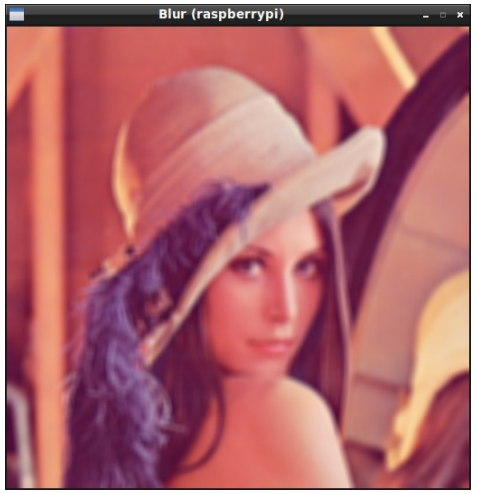
常見濾波器

- 線性濾波
  - 平均平滑 (blur)
  - 高斯平滑 (GaussianBlur)
- 非線性濾波
  - 中值濾波 (medianBlur)
  - 雙邊濾波 (bilateralFilter)

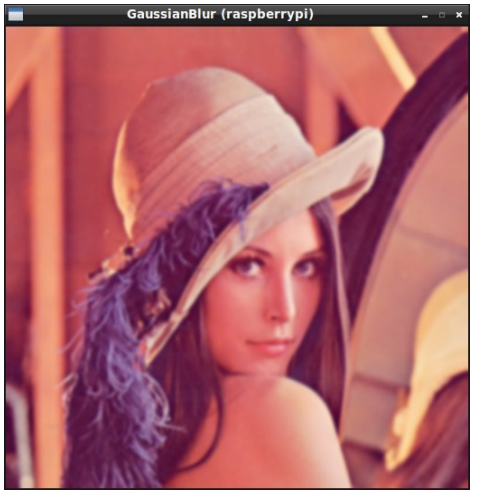
不同濾波器有不同的平滑效果



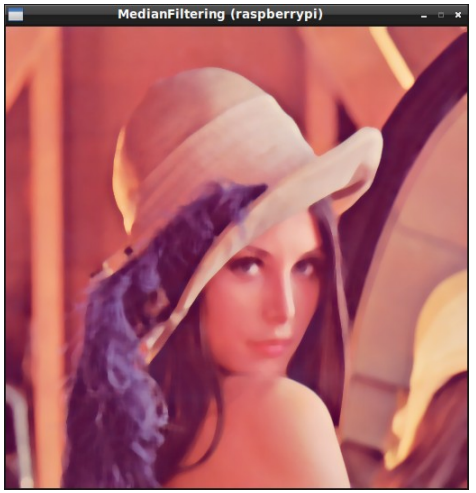
Normal



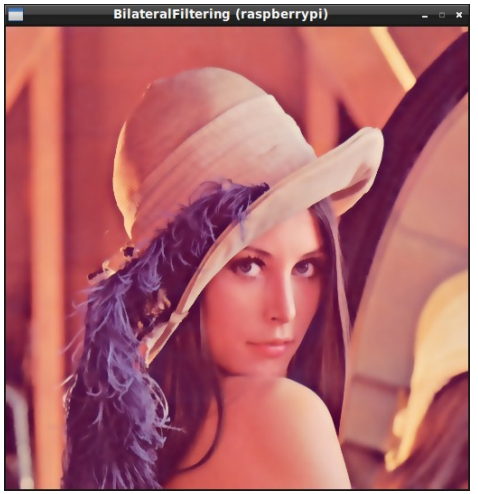
Box Blur



Gaussian Blur



Median Blur



Bilateral Blur

Original Image

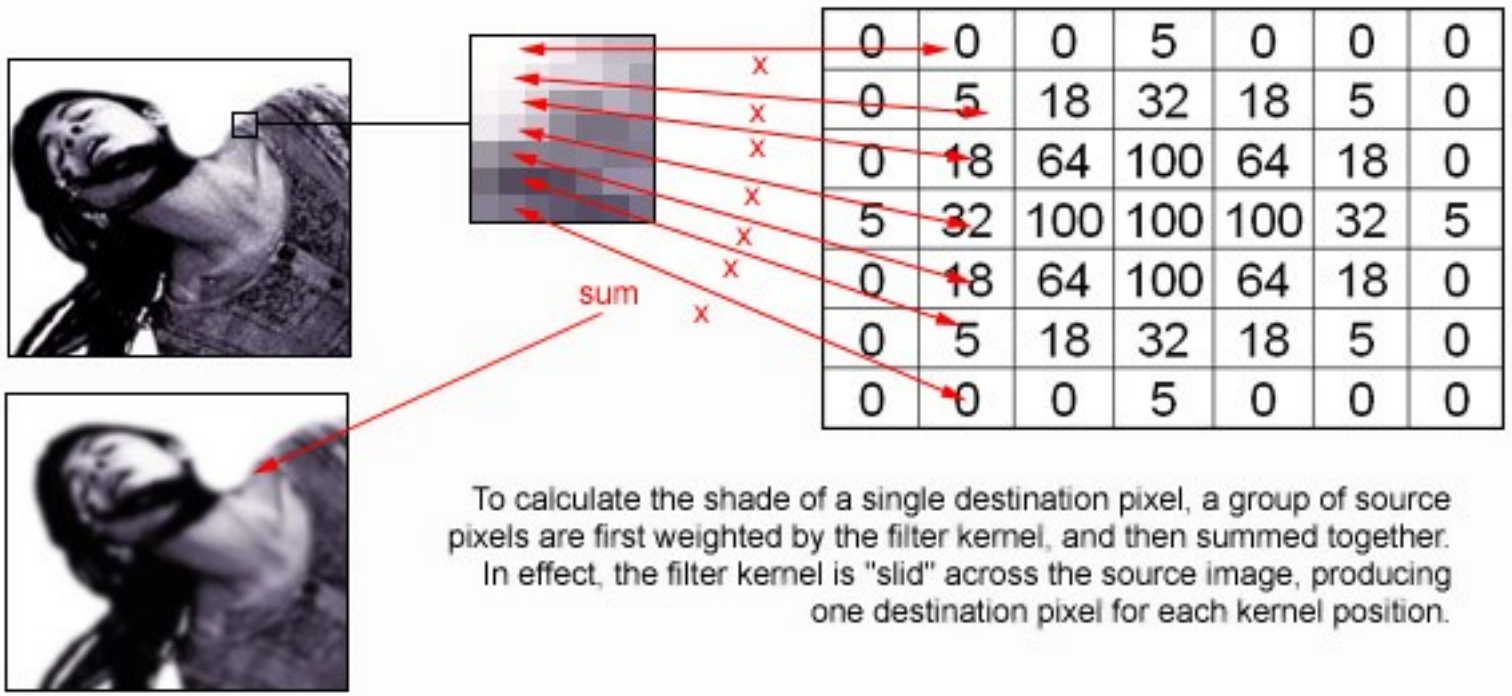
Box-filtered image

Gaussian-filtered image

Kernels used for blurring:
(note that the values shown have been scaled up to integers for clarity)

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0



To calculate the shade of a single destination pixel, a group of source pixels are first weighted by the filter kernel, and then summed together. In effect, the filter kernel is "slid" across the source image, producing one destination pixel for each kernel position.

http://www.gamasutra.com/view/feature/131511/four_tricks_for_fast_blurring_in_.php

高斯平滑 (GaussianBlur)

和中心點的距離

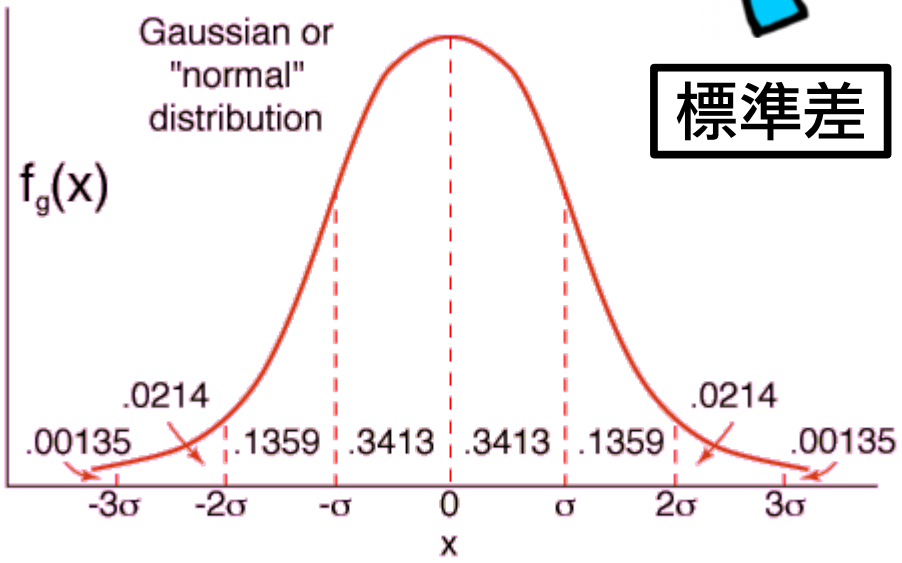
一維計算公式

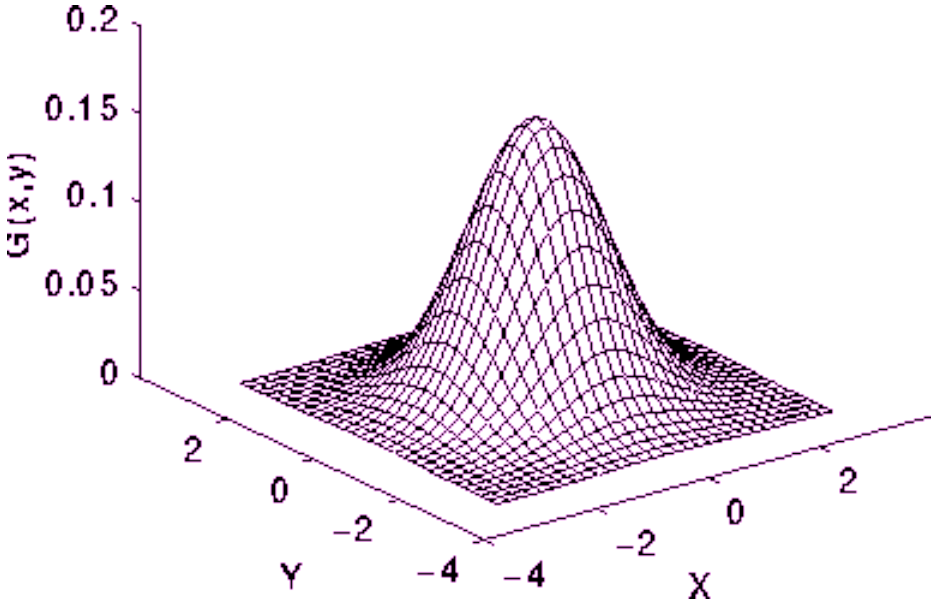
$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

標準差

二維計算公式

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$





http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm

高斯平滑效果

```
cv2.namedWindow('Gaussian_Blur')
cv2.createTrackbar('ksize', 'Gaussian_Blur', 0, 10, nothing)

image = cv2.imread("lena512rgb.png")

while True:
    ksize  = cv2.getTrackbarPos('ksize', 'Gaussian_Blur')
    image = cv2.imread(imagePath)
    blur = cv2.GaussianBlur(image, (2*ksize+1, 2*ksize+1), 0)
    cv2.imshow('Gaussian_Blur', blur)
```

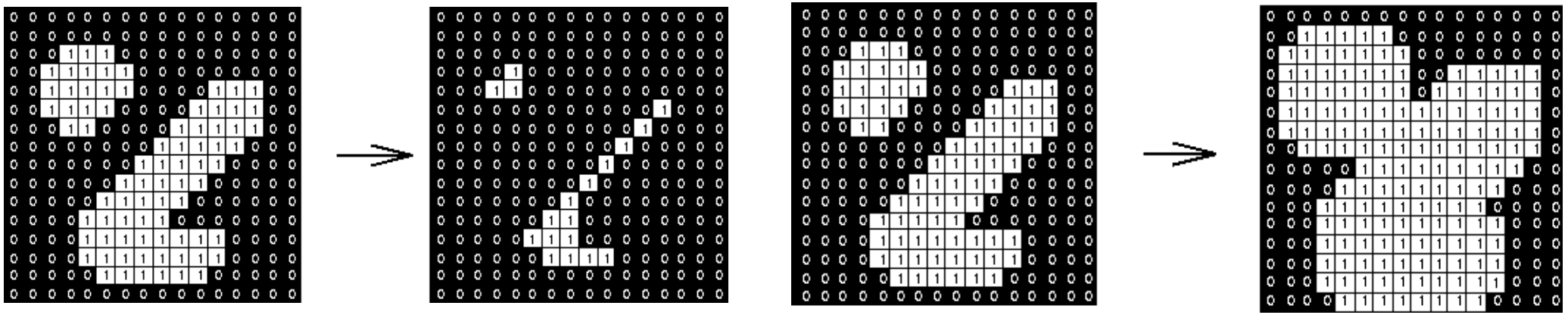
kernel size

DEMO
gaussian_blur.py

```
$ cd ~/cam-py-cv/day2/02-image_process
$ python gaussian_blur.py
```

從形態學的觀點

- 侵蝕 (Erode)
  - 消融物體的邊界
- 膨脹 (Dilate)
  - 擴大物體的邊界

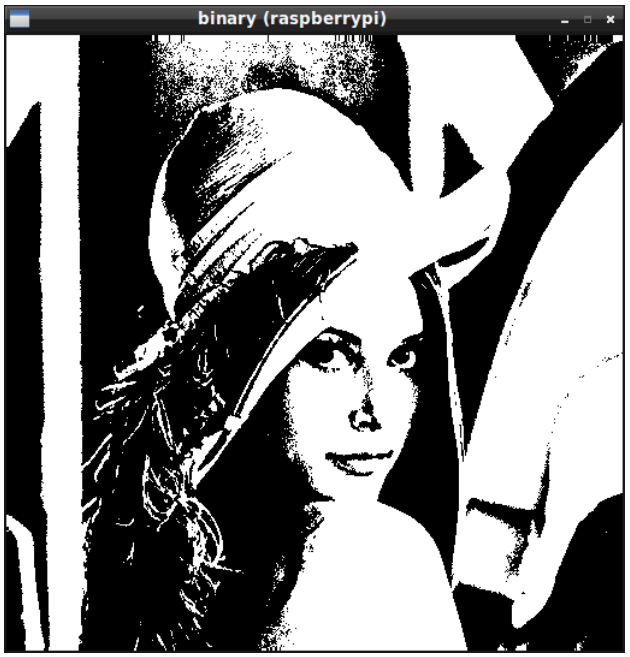


通常以奇數矩形為結構元素 (3x3, 5x5, 7x7...)，預設為 3x3

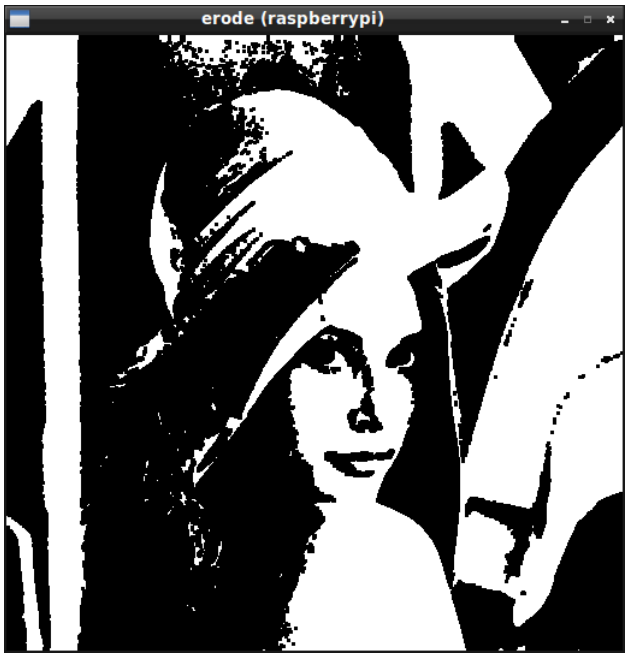
https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm

侵蝕 (Erode)

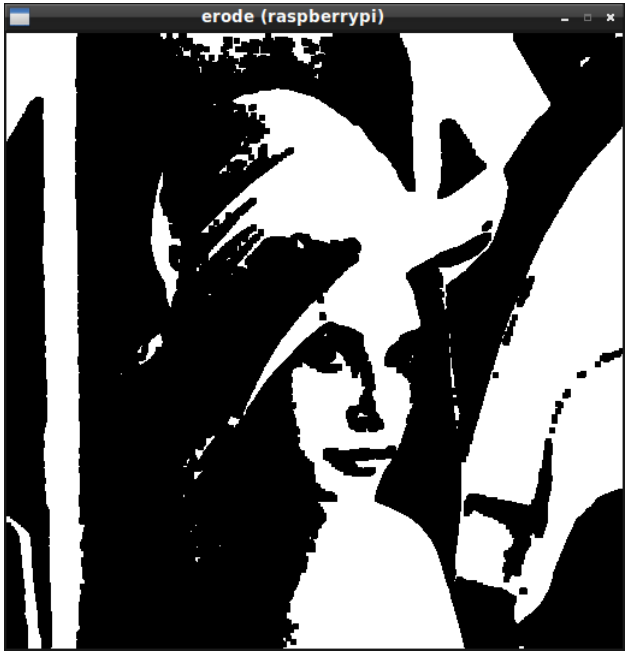
- `cv2.erode(src, kernel[, iterations])`
  - `iterations - number of times erosion is applied.`



黒白



iteration=1

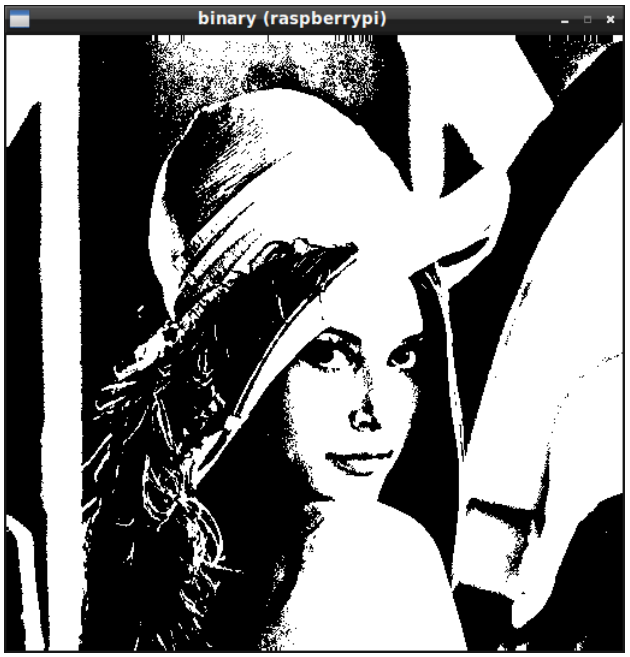


iteration=2

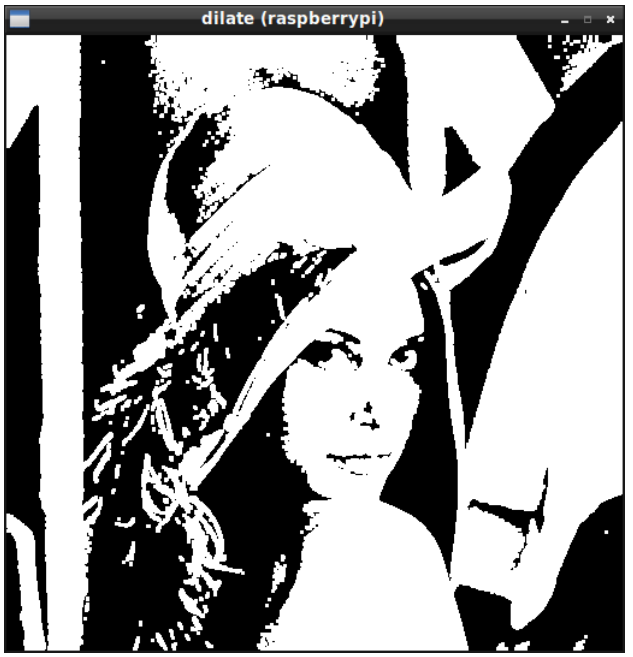
http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html

膨脹 (Dilate)

- `cv2.dilate(src, kernel[, iterations])`
  - `iterations – number of times dilation is applied.`



黑白



iteration=1

iteration=2

http://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html

侵蝕 / 膨脹效果

```
image = cv2.imread("lena512rgb.png”)
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(_, binary) = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY

kernel = np.ones((3,3),np.uint8)
erode  = cv2.erode(binary, kernel, iterations=1)
cv2.imshow("erode", erode)

dilate = cv2.dilate(binary, kernel, iterations=1)
cv2.imshow("dilate", dilate)
```

侵蝕

膨脹

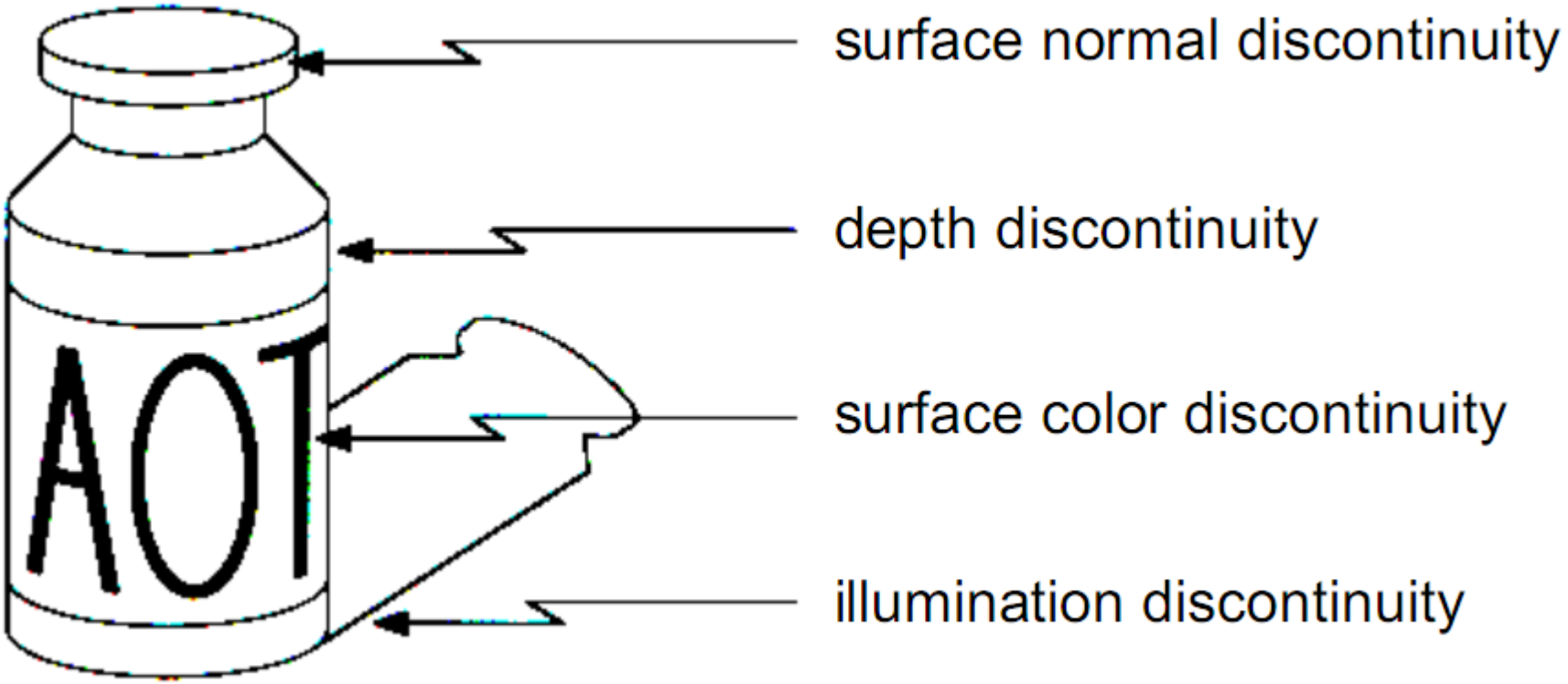
DEMO
erode_dilate.py

```
cd ~/cam-py-cv/day2/02-image_process
$ python erode_dilate.py
```

影像處理後的應用

邊緣

- 邊緣的產生來自於：



http://slazebni.cs.illinois.edu/spring16/lec07_edge.pdf

邊緣偵測

- 是為了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)

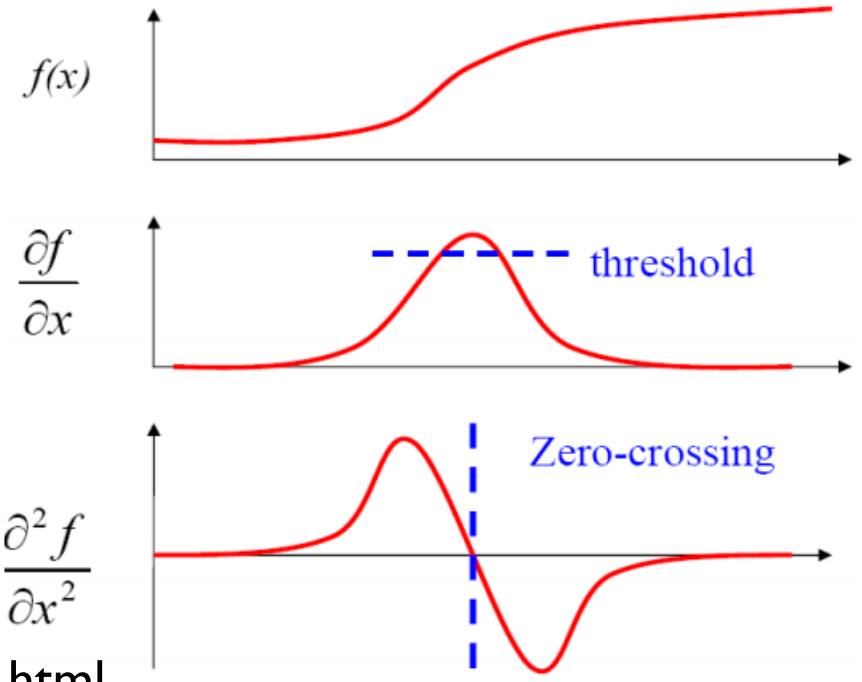
- 梯度可從一階微分 (derivative) 和二階微分求得

(a) Step edge

(b) Ramp

(c) Roof edge

(d) Real edge



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

常見邊緣檢測法

- 一階微分 (梯度法)
  - Roberts operator
  - Sobel operator
  - Prewitt operator
- 二階微分
  - Laplacian
  - Laplacian of Gaussian
  - Difference of Gaussian(DOG)
- 多級邊緣檢測
  - Canny edge detection

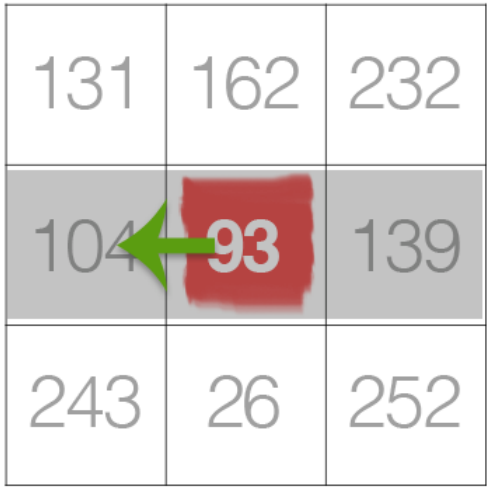
Canny 邊緣檢測 (Edge Detection)

- John F. Canny 所開發出的多級邊緣檢測演算法
- 優點：低錯誤率，高定位性，最小回應

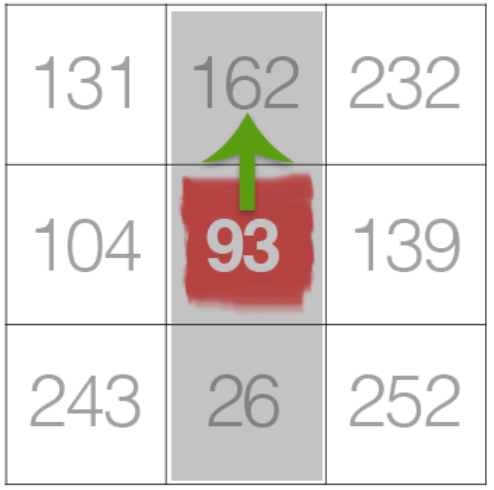
https://en.wikipedia.org/wiki/Canny_edge_detector
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.420.3300&rep=rep1&type=pdf

細小化邊緣曲線 (Edge Thinning)

- Non-maxima Suppression(非最大值抑制):
  - 一次比較 3x3 的像素
  - 依照梯度垂直或水平方向，取上下或左右兩像素
  - 將中心點和該兩像素比較，取大值



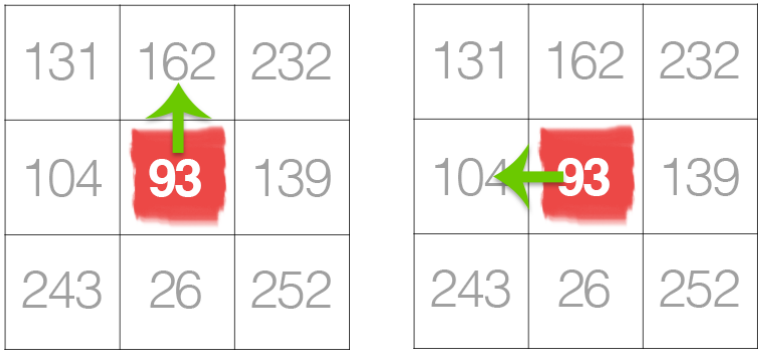
水平梯度，區域極大值 =139

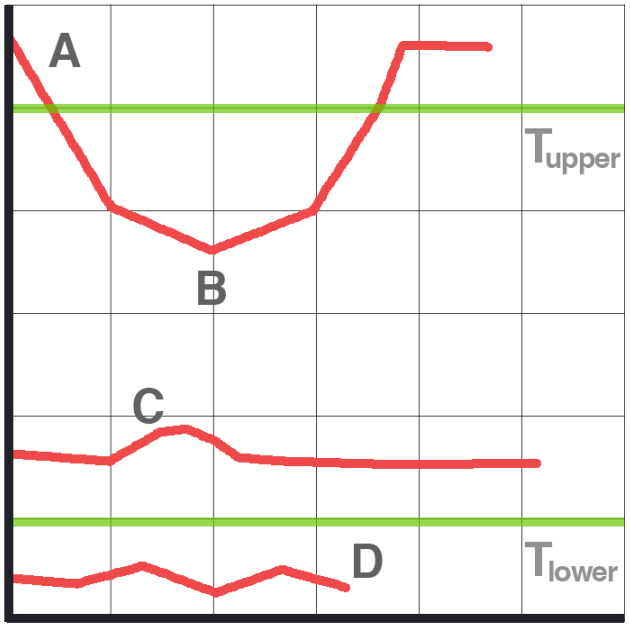


垂直梯度，區域極大值 =162

Canny Edge Detection 步驟

1. 去除雜訊 (常用高斯平滑濾波)
2. 計算梯度方向和強度 (Sobel)
3. 非最大抑制



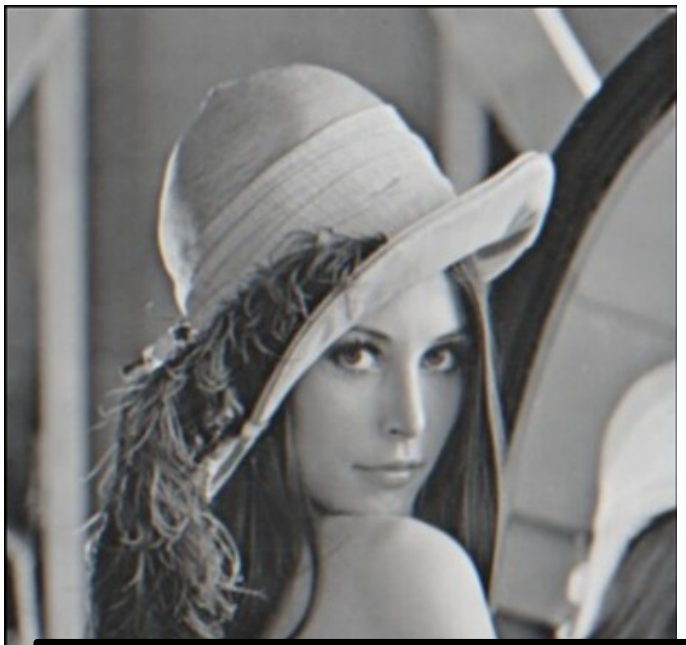


4. 判斷邊界

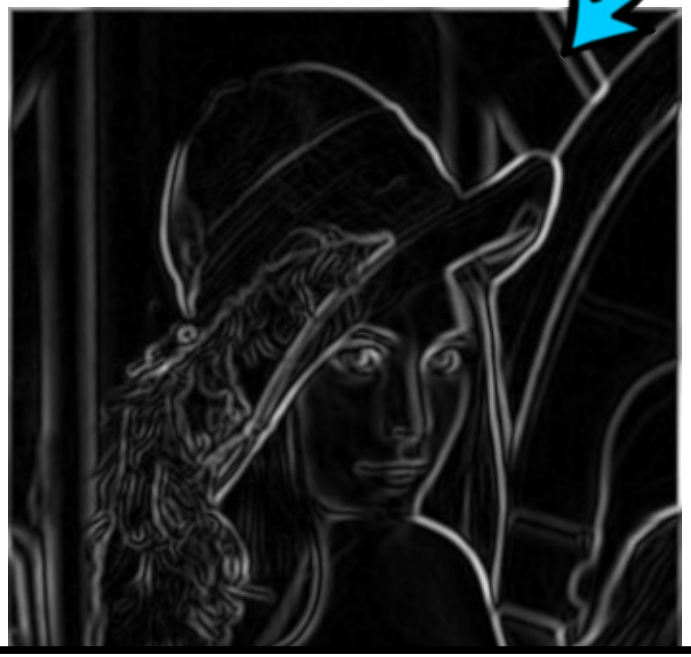
- `if pixel gradient > upper threshold, pixel = edge`
- `if pixel gradient < lower threshold, pixel != edge`
- `if lower < pixel gradient < upper &&`
  `neighbor > upper threshold, pixel = edge`

http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

Normal



去除雜訊 (Gaussian Blur)



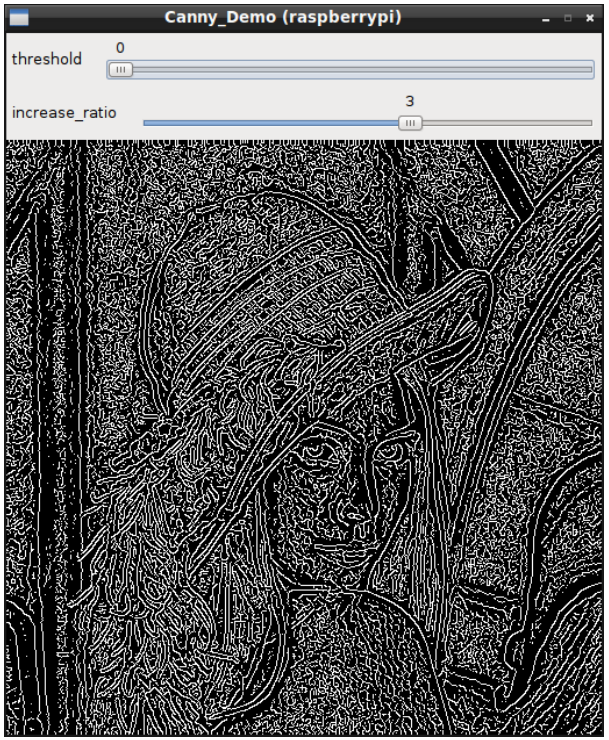
計算梯度方向和強度 (Sobel)

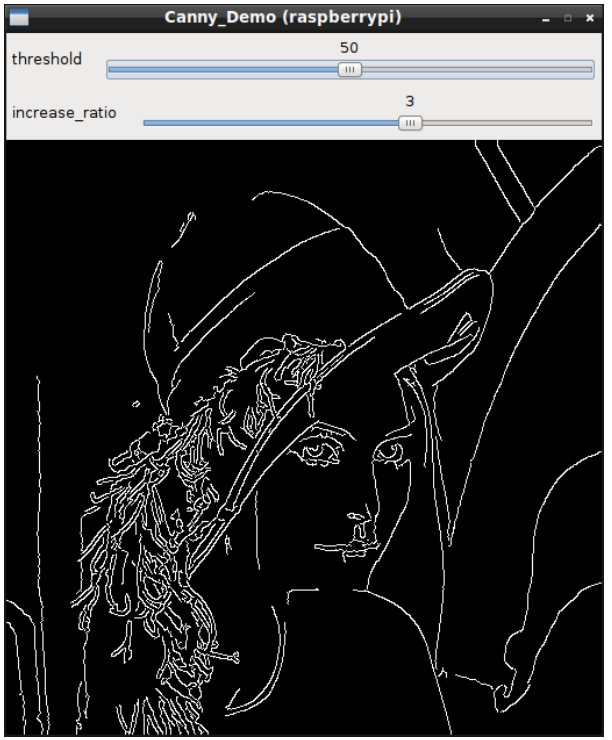
非最大抑制
(Non-maximum suppreson)

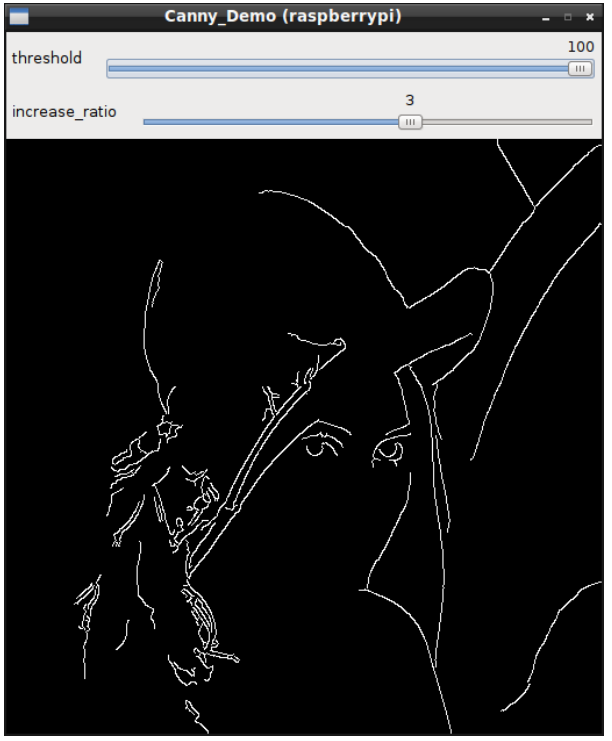
http://slideplayer.com/slide/8629030/

使用 Canny

- `cv2.Canny(img,lowerT,upperT,[apertureSize])`
  - 通常上下門檻值的比例在 2:1 到 3:1 之間
  - `apertureSize(Sobel kernel)` 預設是 3







http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/canny_detector/canny_detector.html

即時更新門檻值

```
cv2.createTrackbar('threshold', 'canny_demo', 0, 100, nothing)
cv2.createTrackbar('ratio',     'canny_demo', 0, 5,   nothing)
```

lower threshold = threshold
upper threshold = threshold x ratio

```
while True:
    threshold = cv2.getTrackbarPos('threshold', 'canny_demo')
    ratio     = cv2.getTrackbarPos('ratio',     'canny_demo')
    image = cv2.imread(imagePath)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    edges = cv2.GaussianBlur(gray, (5, 5), 0)
    edges = cv2.Canny(edges, threshold, threshold * ratio)
    cv2.bitwise_and(image, image, mask=edges)
```

DEMO
canny_edge_detect.py

```
$ cd ~/cam-py-cv/day2/02-image_process
$ python canny_edge_detect.py
```

找邊緣的目的是為了找線

Hough Transform

- P.Hough 所提出，用在二值化影像的形狀偵測
- 實做步驟為
  1. 空間映射（影像空間映射到參數空間，二維到一維）
  2. 座標轉換（笛卡兒座標轉換到極座標）
  3. 使用累加器(Accumulator)

(a) Input image (480×270)

(b) Edge of input image

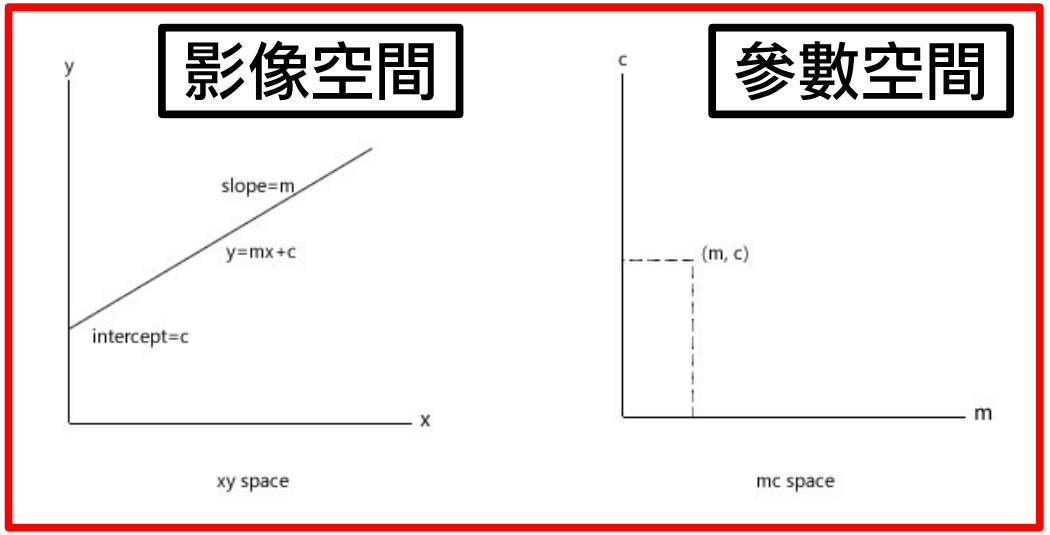
(c) Line detection (480×480 HS)

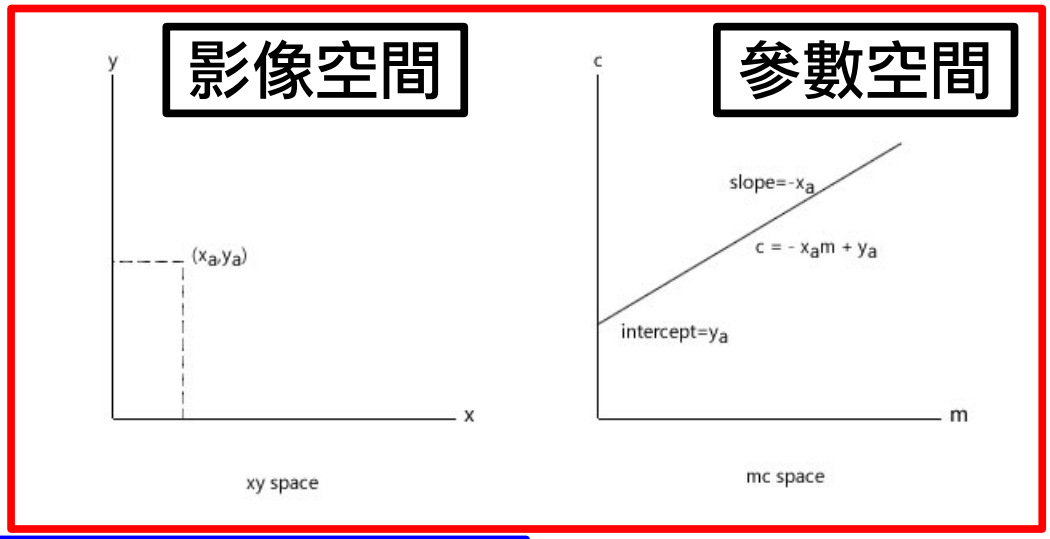
(d) Line detection (120×120 HS)

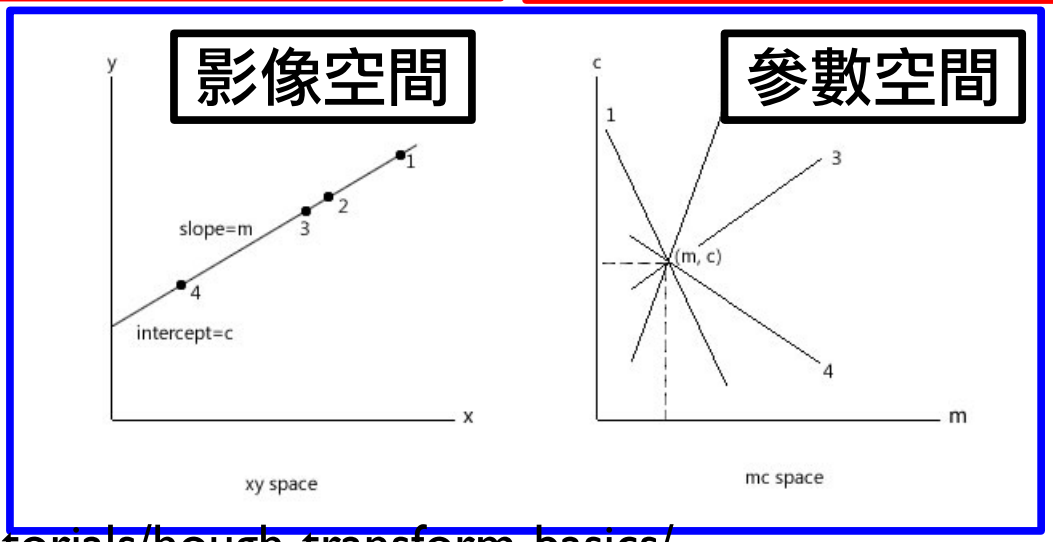
http://goo.gl/3O1tcq

空間映射

- 影像空間映射到參數空間





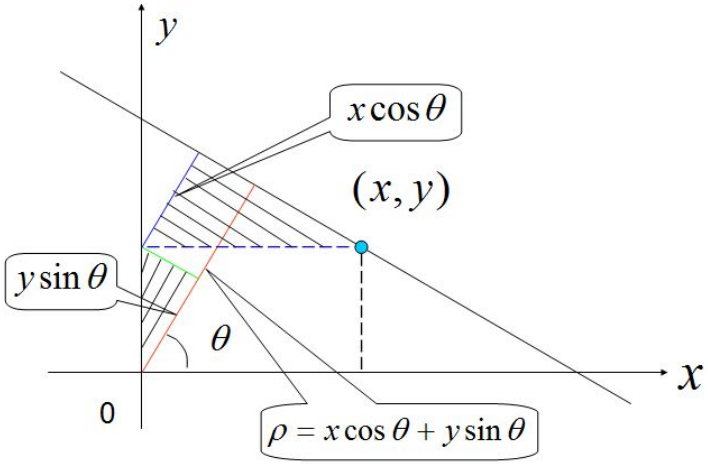


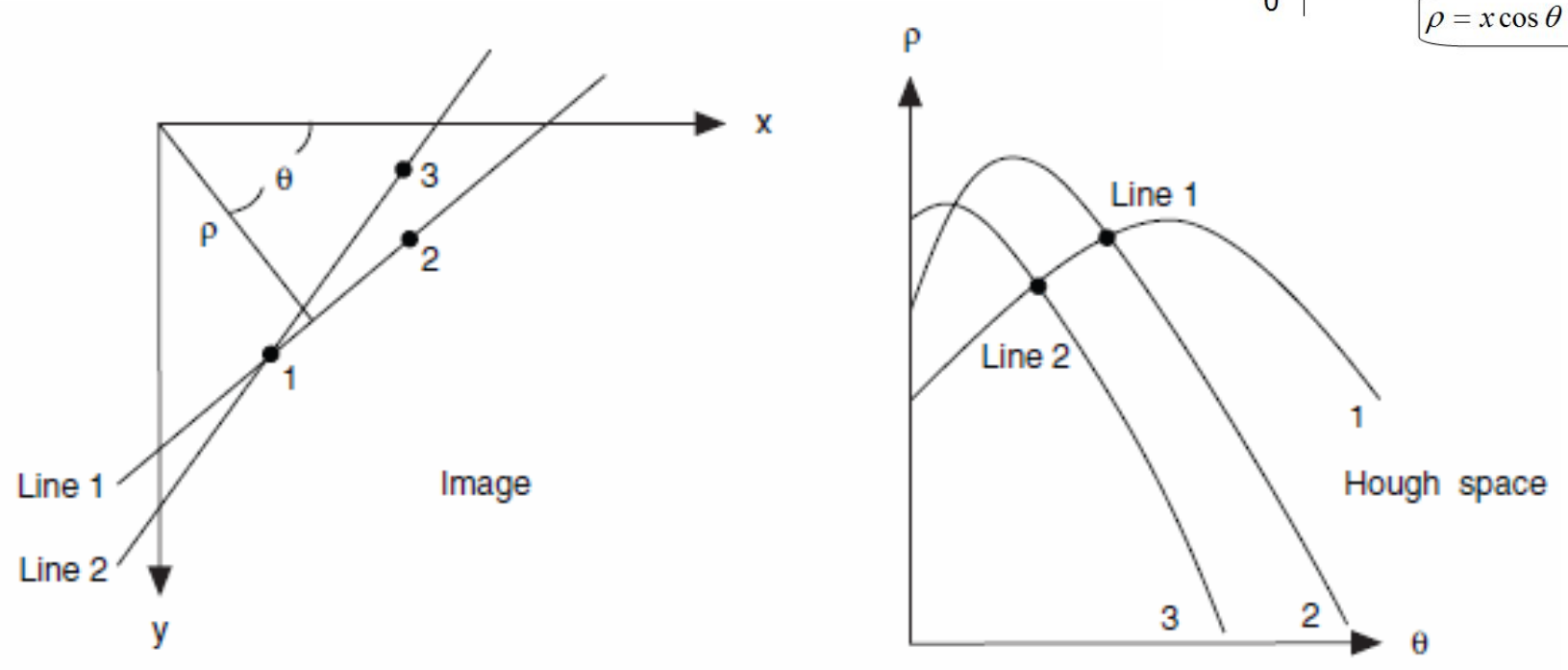
http://www.aishack.in/tutorials/hough-transform-basics/

座標轉換

$$\rho = x\cos\theta + y\sin\theta$$

- 笛卡兒座標轉換到極座標



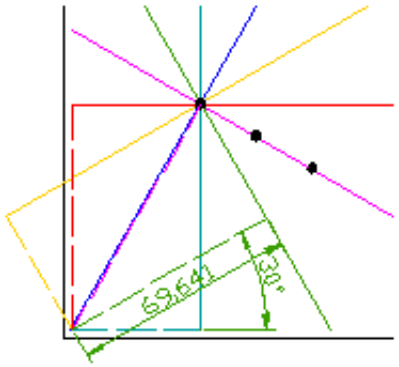


笛卡兒座標　　極座標

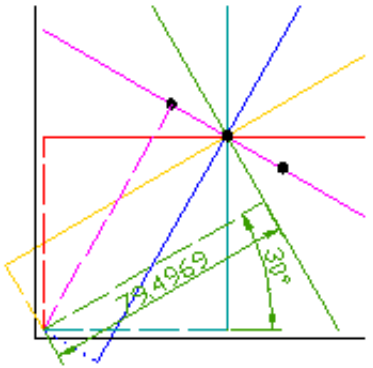
http://zone.ni.com/reference/en-XX/help/372916L-01/nivisionconcepts/edge_detection_concepts/

使用累加器

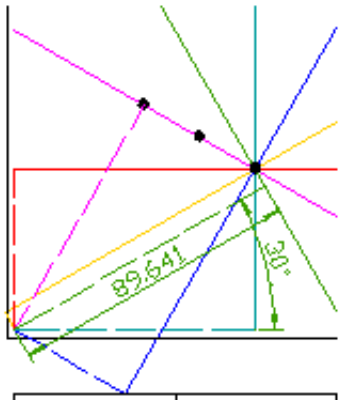
- 找出最大值（或超過門檻值）



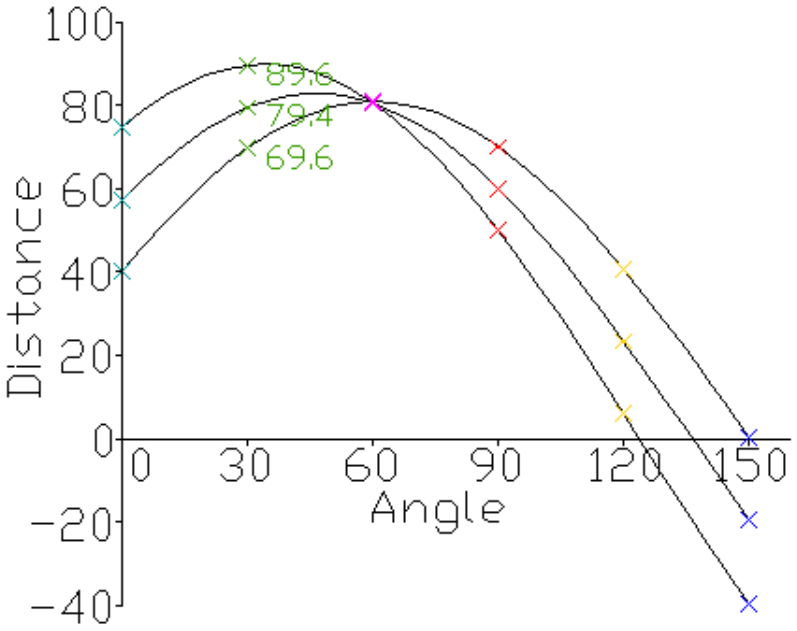
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5



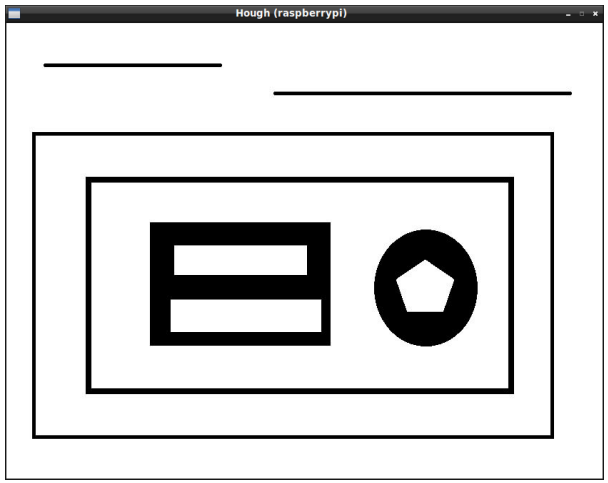
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6



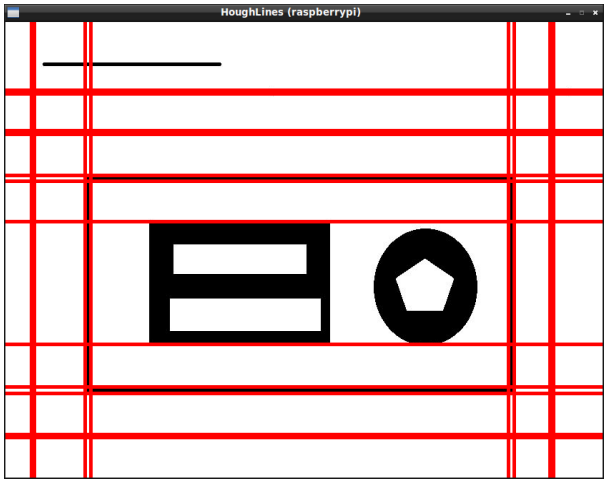
https://en.wikipedia.org/wiki/Hough_transform

HoughLines & HoughLinesP

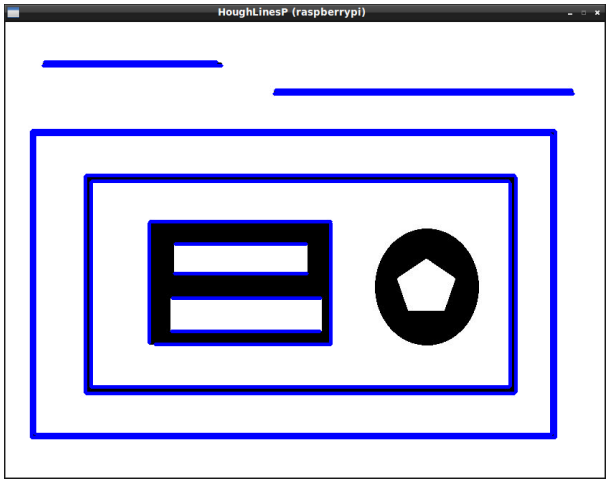
- HoughLines()，找出直線（無窮長）
- HoughLinesP()，找出線段



原始圖



HoughLines



HoughLinesP

http://ccw1986.blogspot.tw/2016/04/hough-transform-using-opencv.html

HoughLines

- 原型：cv2.HoughLines(image, rho, theta, threshold[, lines[, srn[, stn]]])
- 範例：cv2.HoughLines(edges, 1, np.pi/180, 250)
  - image：輸入影像（8 位元單通道二值化圖）
  - rho：距離解析度（極坐標中極徑 r 的最小單位）
  - theta：角度解析度（極坐標中極角 θ 的最小單位）
  - threshold：門檻值（超過此值的線才會存在 lines 裡）
  - lines：輸出結果（每條線都包含 r 和 θ）
  - srn：可有可無的距離除數
  - stn：可有可無的角度除數

http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html

找出直線

```
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
h, w = gray.shape

edges = cv2.Canny(gray, 50, 150, 3)
lines = cv2.HoughLines(edges, 1, np.pi/180, 250)[0]

for (rho, theta) in lines:
    x0 = np.cos(theta)*rho
    y0 = np.sin(theta)*rho
    pt1 = ( int(x0 + (h+w)*(-np.sin(theta))), int(y0 +
(h+w)*np.cos(theta)) )
    pt2 = ( int(x0 - (h+w)*(-np.sin(theta))), int(y0 -
(h+w)*np.cos(theta)) )
    cv2.line(image, pt1, pt2, (0, 0, 255), 3)
```

門檻值

DEMO
hough_find_lines.py

```
$ cd ~/cam-py-cv/day2/02-image_process
$ python hough_find_lines.py hough_demo.jpg
```

HoughLinesP

- 原型：cv2.HoughLinesP(image, rho, theta, threshold[, lines[, minLineLength[, maxLineGap]]])
- 範例：cv2.HoughLinesP(edges, 1, np.pi/180, 50, None, 100, 5)
  - image：輸入影像（8 位元單通道二值化圖）
  - rho：距離解析度（極坐標中極徑 r 的最小單位）
  - theta：角度解析度（極坐標中極角 θ 的最小單位）
  - threshold：門檻值（超過此值的線才會存在 lines 裡）
  - lines：輸出結果（每條線都包含 x1, y1, x2, y2 線段頂點）
  - minLineLength：線段最短距離（超過此值的線才會存在 lines 裡）
  - maxLineGap：最大間隔

http://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html

找出線段

```
gray = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)

edges = cv2.Canny(gray, 50, 150, 3)
plines = cv2.HoughLinesP(edges, 1, np.pi/180, 50, None,
100, 5)[0]

for pl in plines:
    cv2.line(image, (pl[0], pl[1]), (pl[2], pl[3]), (255
0, 0), 3)
```

門檻值

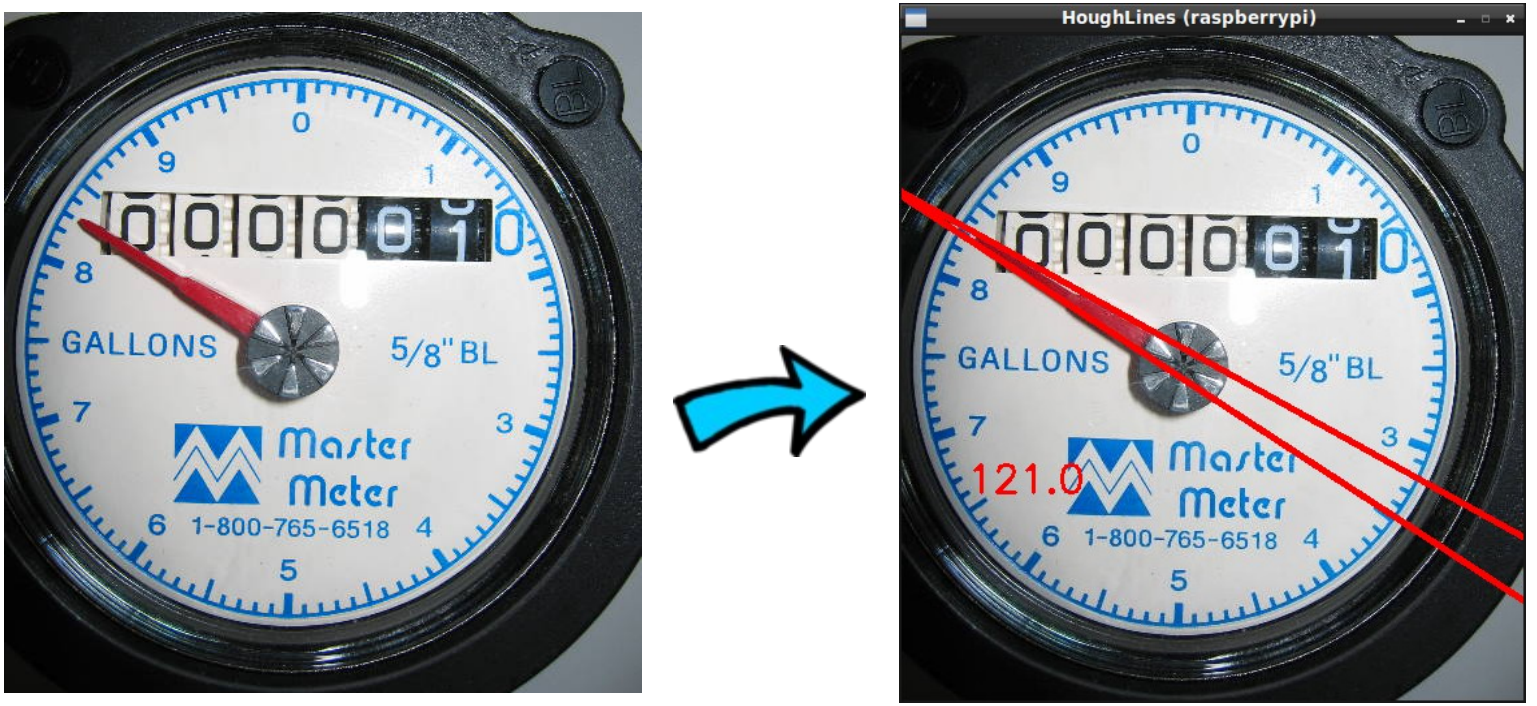
DEMO
hough_find_linesp.py

```
$ cd ~/cam-py-cv/day2/02-image_process
$ python hough_find_linesp.py hough_demo.jpg
```

練習

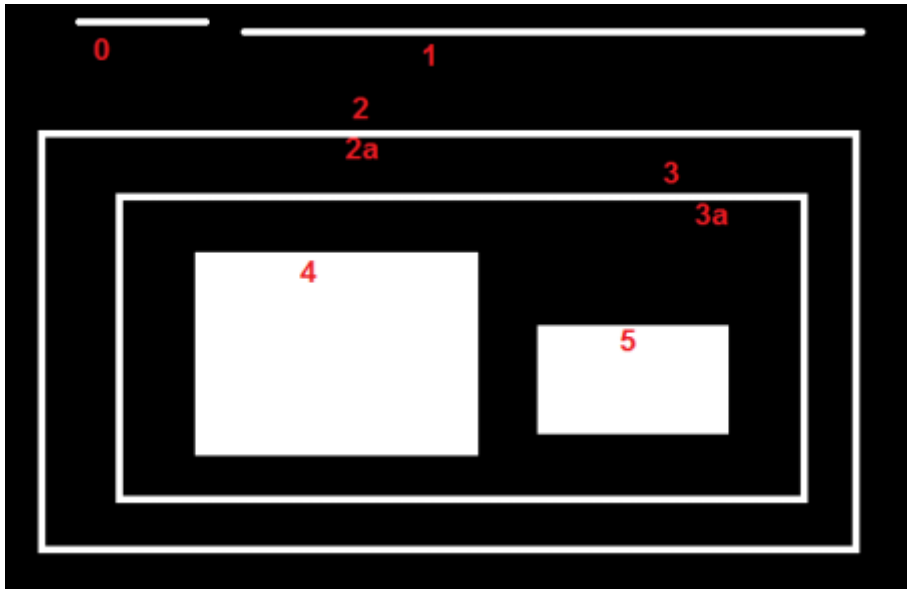
- 使用hsv_value.py加上canny_edge_detect.py加上hough_find_lines.py找出水表的指針角度(wm3.jpg)

```
$ python hsv_value.py wm3.jpg
$ python canny_edge_detect.py hsv_demo.png
$ python hough_find_lines.py canny_demo.png
```

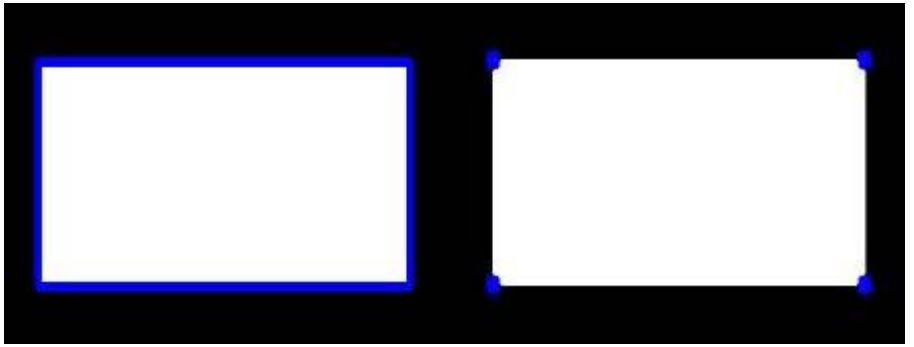


尋找輪廓 (findContours)

- `cv2.findContours(image, mode, method)`
  - `mode(` 輪廓檢索模式 `)`
  - `method(` 輪廓近似方法 `)`



輪廓檢索模式



輪廓近似方法

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html

參數說明

- mode(輪廓檢索模式)
  - CV_RETR_EXTERNAL: 只取最外層的輪廓
  - CV_RETR_LIST: 取得所有輪廓，不建立階層 (hierarchy)
  - CV_RETR_CCOMP: 取得所有輪廓，但只儲存成兩層的階層
  - CV_RETR_TREE: 取得所有輪廓，以全階層的方式儲存

- method(輪廓近似方法)
  - CV_CHAIN_APPROX_NONE: 儲存所有輪廓點
  - CV_CHAIN_APPROX_SIMPLE: 只留下頭尾點或對角頂點

http://monkeycoding.com/?p=615
http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_contours/py_contours_hierarchy/py_contours_hierarchy.html

畫輪廓 (drawContours)

- `cv2.drawContours(image,contours,contourIdx,color)`
  - `contourIdx = -1` 表示畫出所有輪廓點
  - 先轉為二值化影像能降低雜訊

二值化後找到的輪廓點

在原圖畫上輪廓點

http://docs.opencv.org/3.0-beta/modules/imgproc/doc/drawing_functions.html

找出輪廓並且全部畫出

```
image = cv2.imread("lena256rgb.jpg")


gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(_, binary) = cv2.threshold(gray, 127, 255,
cv2.THRESH_BINARY)


(contours, _) = cv2.findContours(binary,
cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
cv2.drawContours(image, contours, -1, (0,255,0), 3)
cv2.imshow("Contours", image)
```

DEMO
draw_contour.py

影像出現後，按任意鍵會顯示下個影像

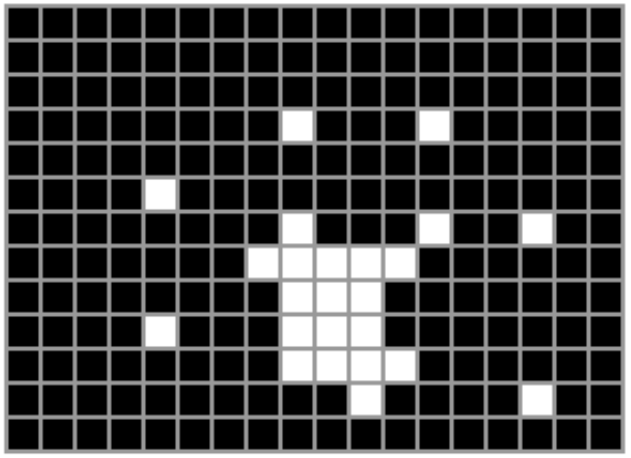
```
$ cd ~/cam-py-cv/day2/02-image_process
$ python draw_contour.py lena512rgb.png
```

找中心點

- 從矩(moment)的觀點來看
  - 物理學：表示作用力促使物體繞支點旋轉的趨向
  - 數學：用來描述資料分佈特徵
- Image moments是一串用來描述目標物件形狀的數值，例如面積，中心點(centroid)或旋轉角度
- 以二值化(黑白)的圖形找中心點

http://docs.opencv.org/3.1.0/d8/d23/classcv_1_1Moments.html

常見矩



- 空間矩 (spatial moments)

$$\mathbf{m}_{ji} = \sum_{x,y} \left(\mathbf{array}(x,y) \cdot x^j \cdot y^i \right)$$

- 中心矩 (central moments)

$$\mathbf{mu}_{ji} = \sum_{x,y} \left(\mathbf{array}(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i \right)$$

$$\bar{x} = \frac{\mathbf{m}_{10}}{\mathbf{m}_{00}},\ \bar{y} = \frac{\mathbf{m}_{01}}{\mathbf{m}_{00}}$$

中心點 (質心 / 圖心)

http://aishack.in/tutorials/image-moments/

找中心點與外框

```
image = cv2.imread("moment.jpg")

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
(_, binary) = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
(contours, _) = cv2.findContours(binary, cv2.RETR_TREE,
cv2.CHAIN_APPROX_NONE)

cnt = contours[0]

((x, y), radius) = cv2.minEnclosingCircle(cnt)
M = cv2.moments(cnt)

center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
cv2.circle(image, (int(x), int(y)), int(radius), (0, 255, 255), 2)
cv2.circle(image, center, 5, (0, 0, 255), -1)
```

取得輪廓點

第一組輪廓點

計算外框半徑

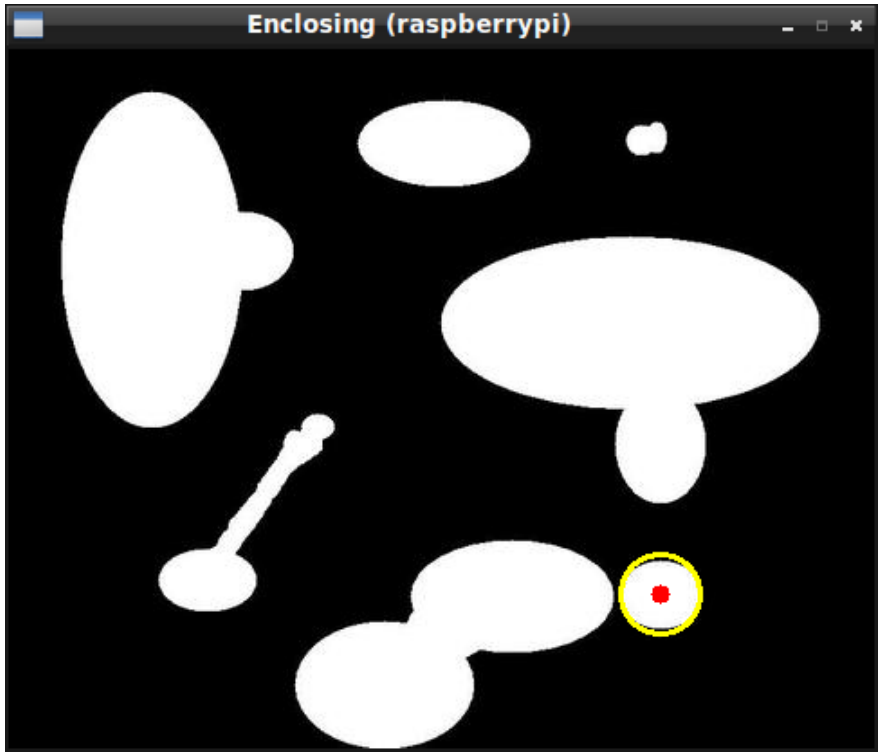
第一組輪廓點的矩

中心點

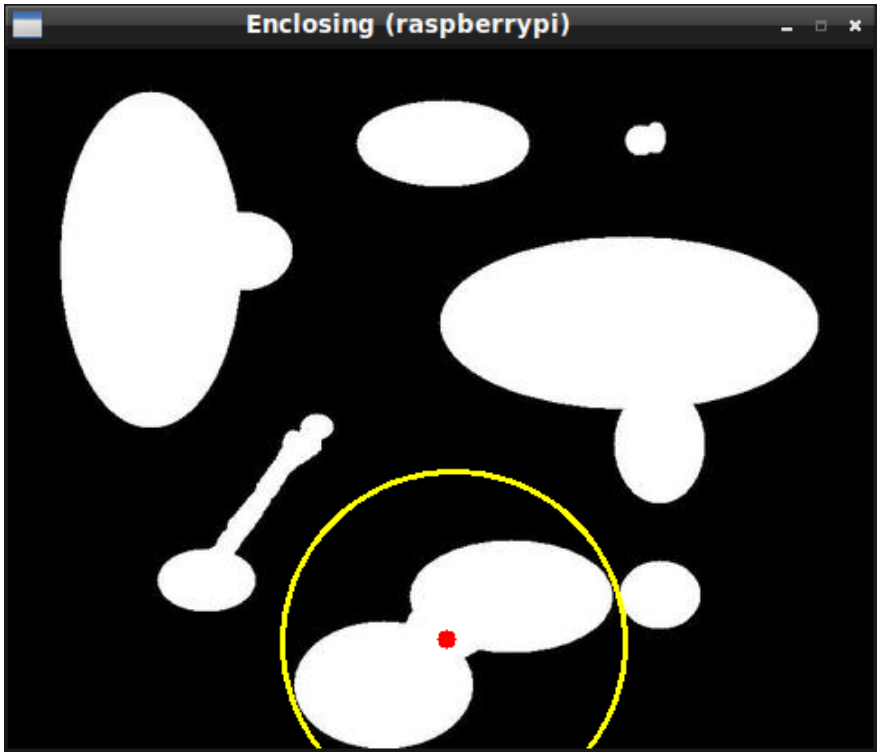
DEMO
draw_moment.py

```
$ cd ~/cam-py-cv/day2/02-image_process
$ python draw_moment.py moment.jpg
```

每組輪廓都可以找到找中心點



第一組輪廓點
`cnt = contours[0]`



第二組輪廓點
`cnt = contours[1]`

人臉偵測 (Face Detection)

實驗 3：人臉偵測

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

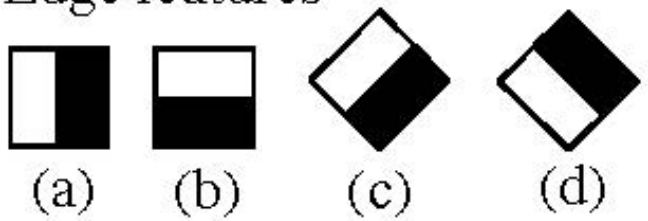
Haar Feature Cascade

- 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)

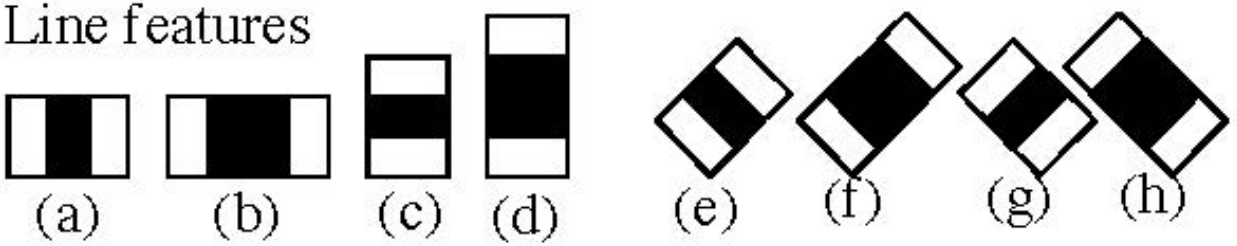
http://www.open-electronics.org/raspberry-pi-and-the-camera-pi-module-face-recognition-tutorial/

Haar-Like Features

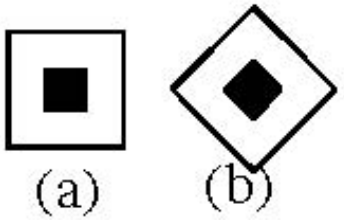
1. Edge features



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

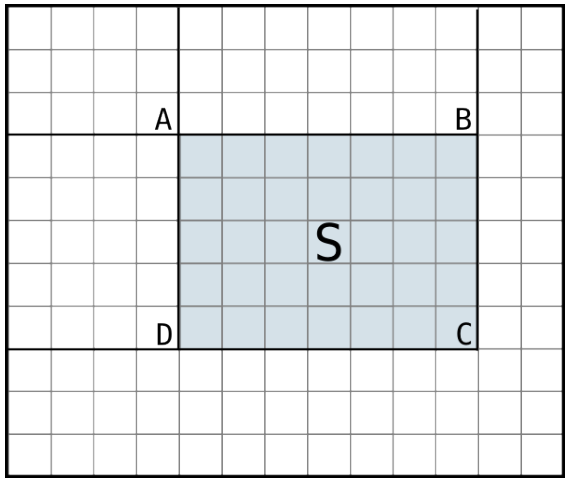
特徵比對

- 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

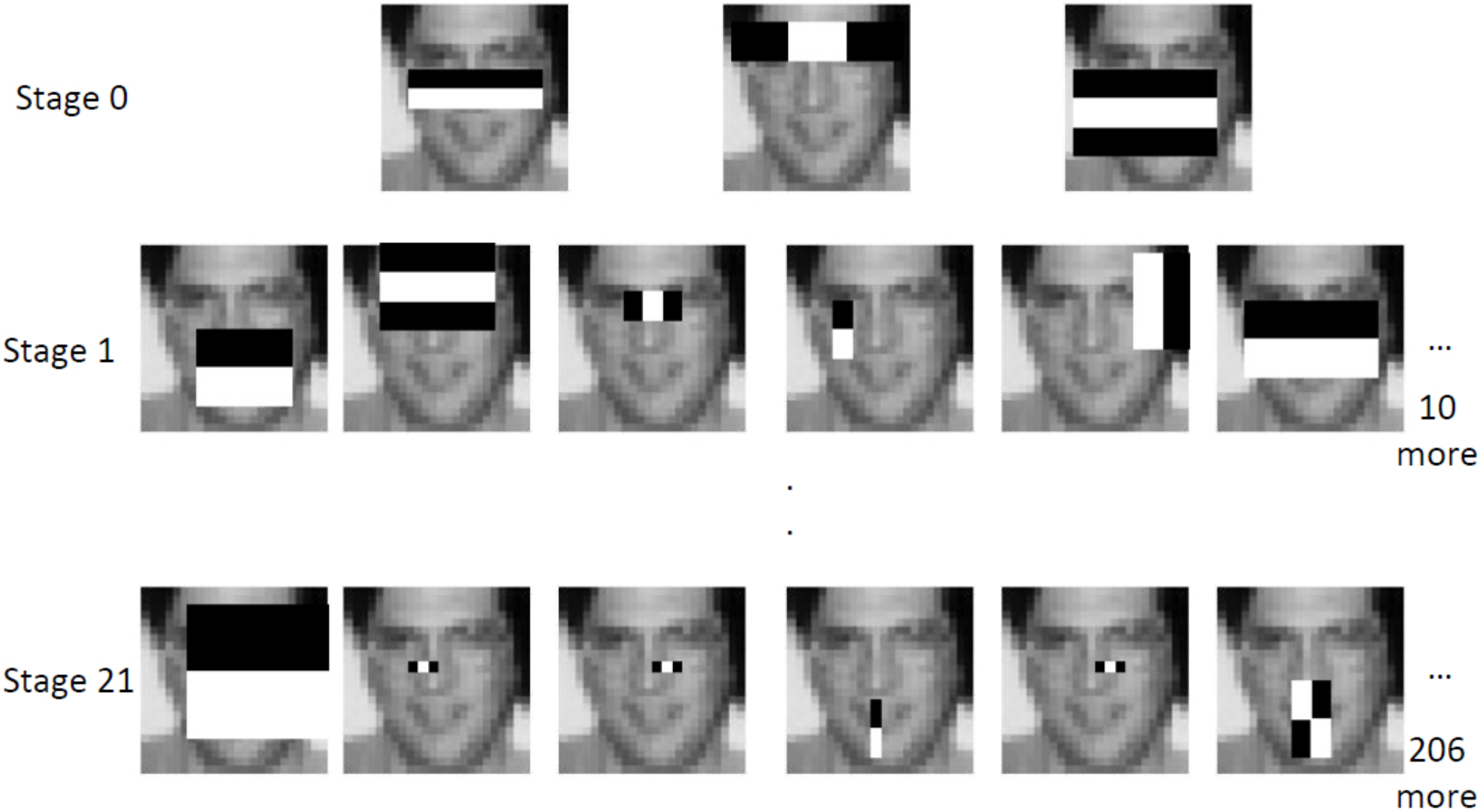
弱分類器

- 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

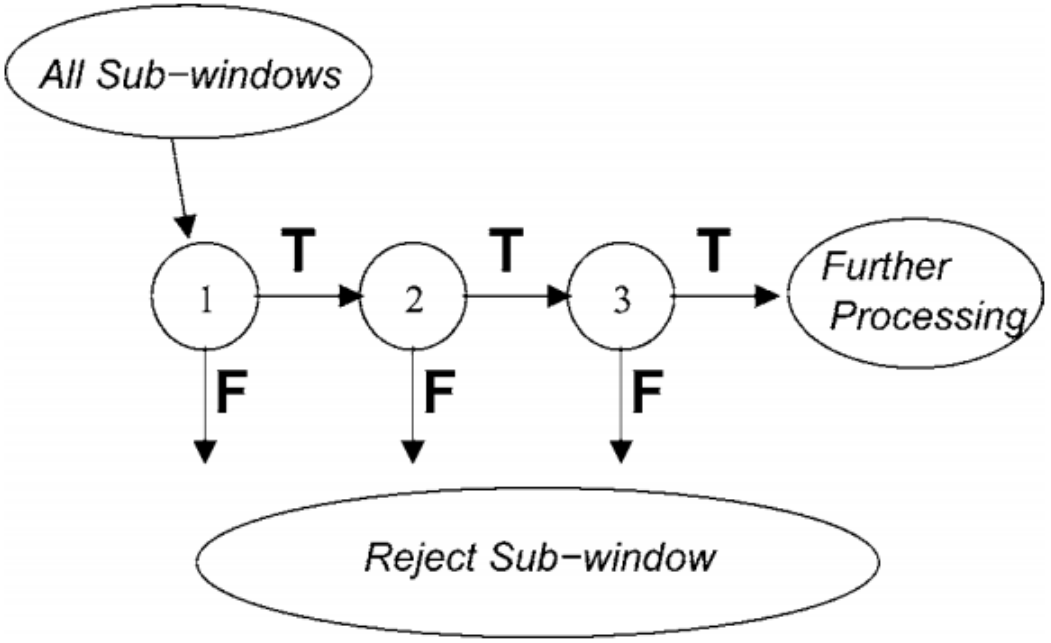
串接多個弱分類器



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

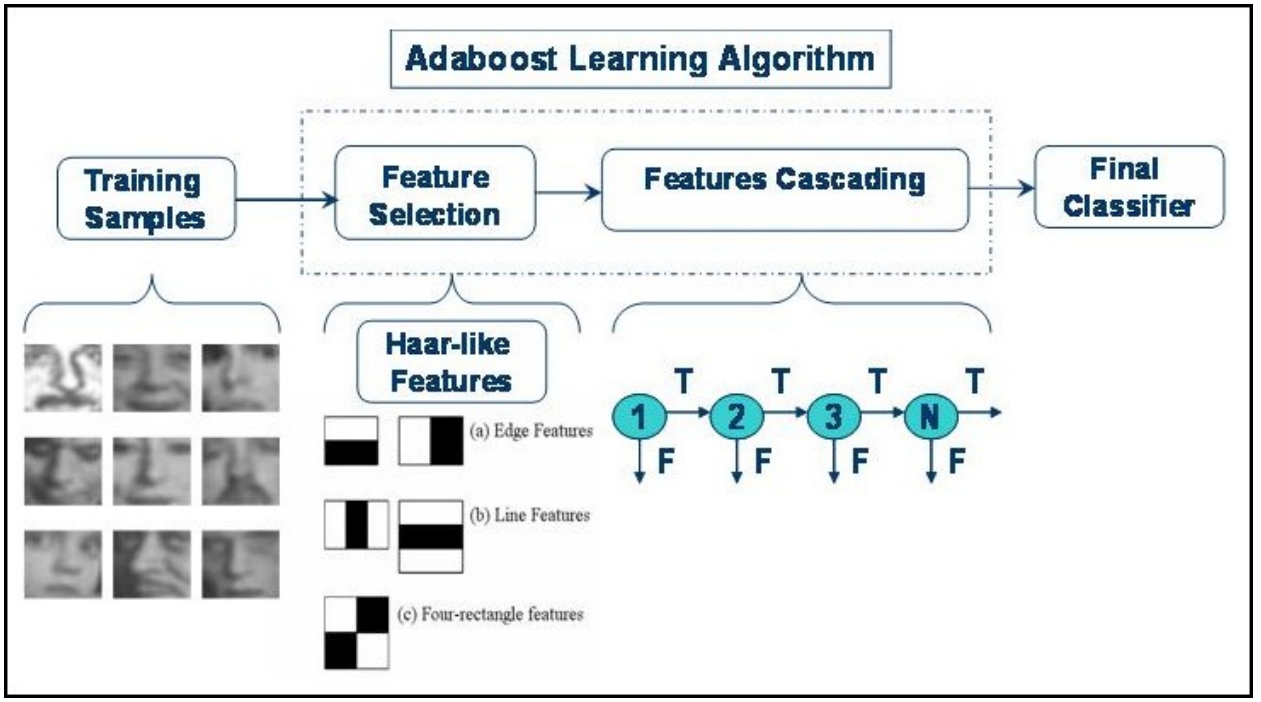
- 單一分類器的準確度雖然只略大於 50%，但如果同時通過多個弱分類器，其準確度將會大副提昇
- 使用弱分類器的好處是可以將未達到門檻值的特徵 (feature) 在早期就先去除掉



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

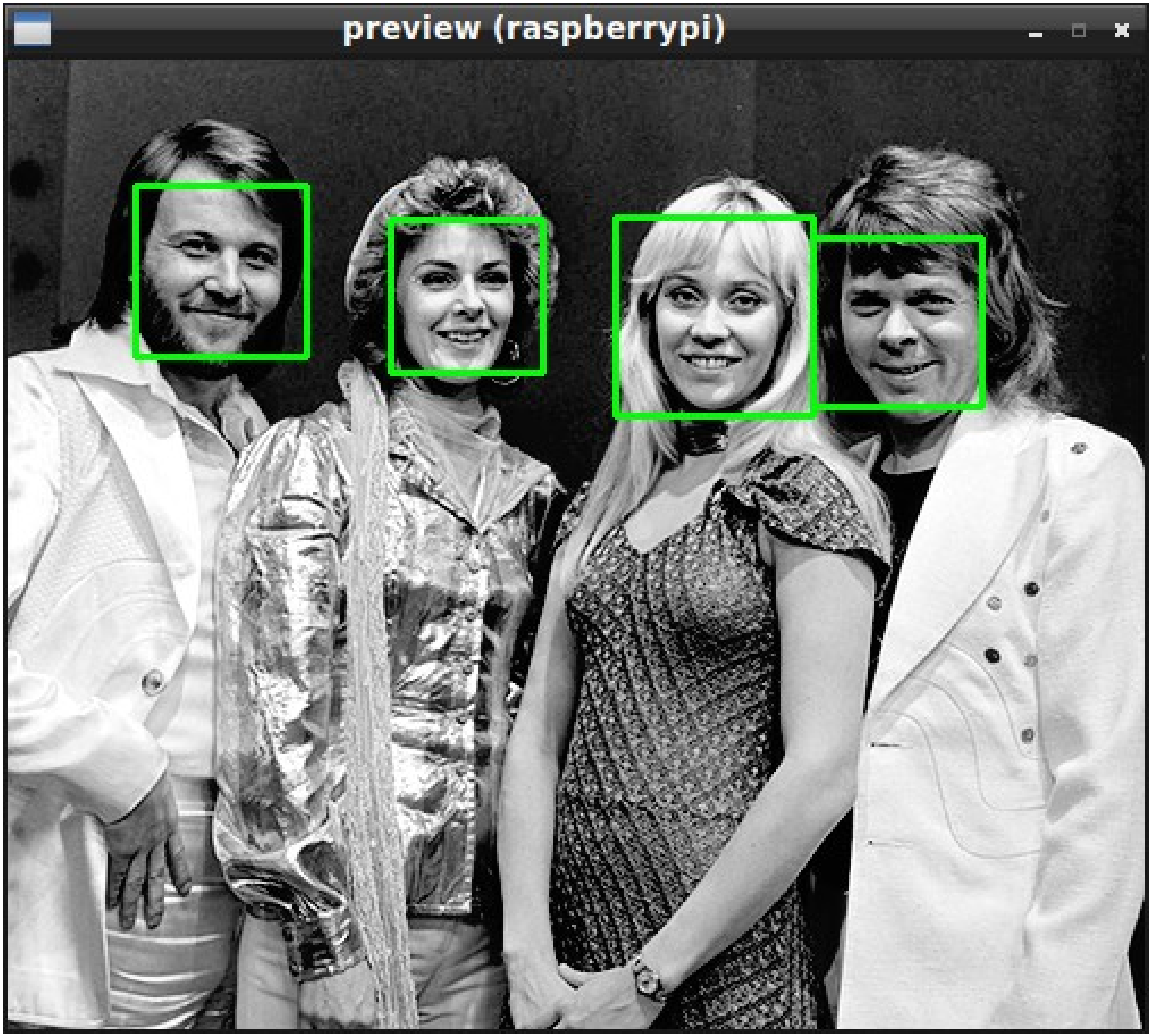
載入圖檔並辨識

```
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

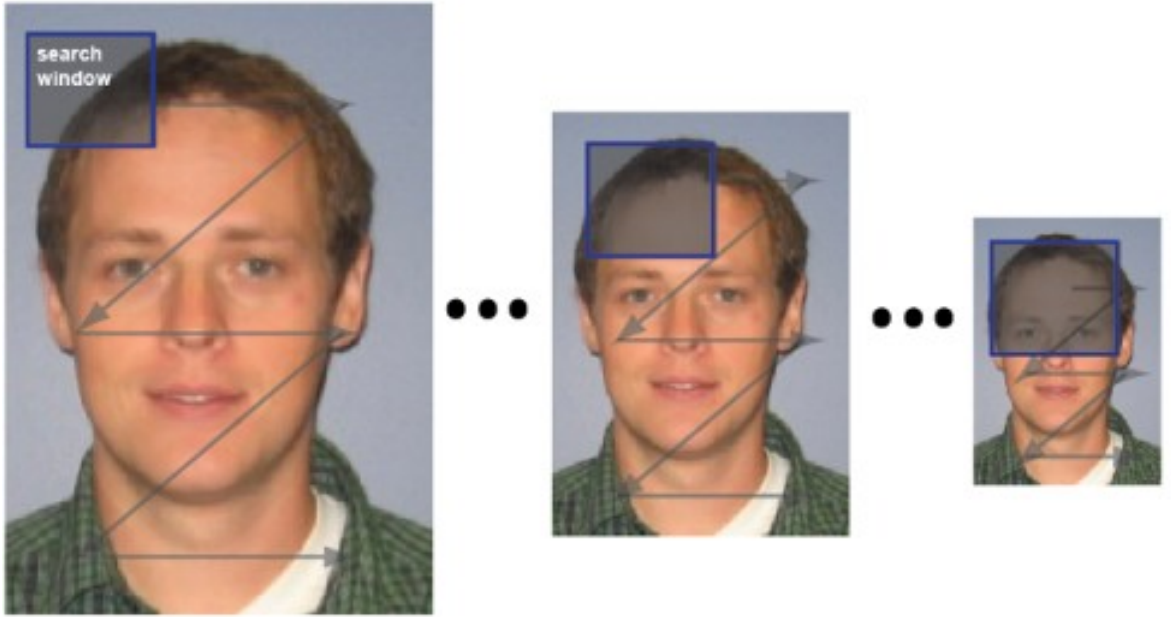
DEMO
image_face_detect.py

```
$ cd ~/cam-py-cv/day2/03-face_datection
$ python imag_face_detect.py abba.png haarcascade_frontalface_default.xml
```


preview (raspberrypi)

可調整的參數

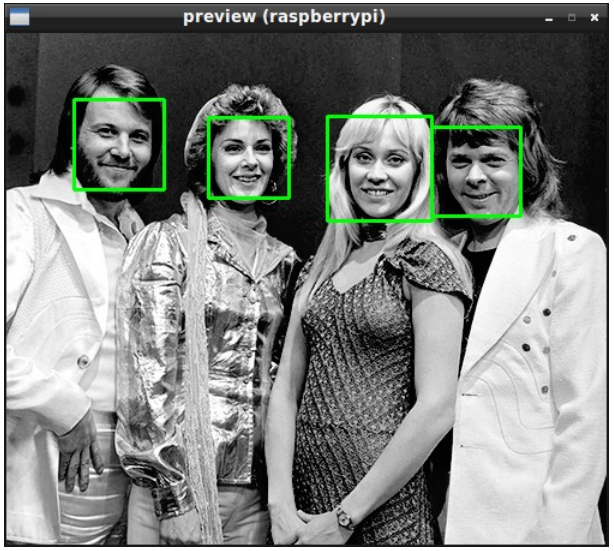
- `scaleFactor`：檢測視窗縮放比率
- `minNeighbors`：檢測區域鄰域內最少包含的檢測出的備選人臉區域 (次數)
- `minSize`：被檢測物體的最小尺寸



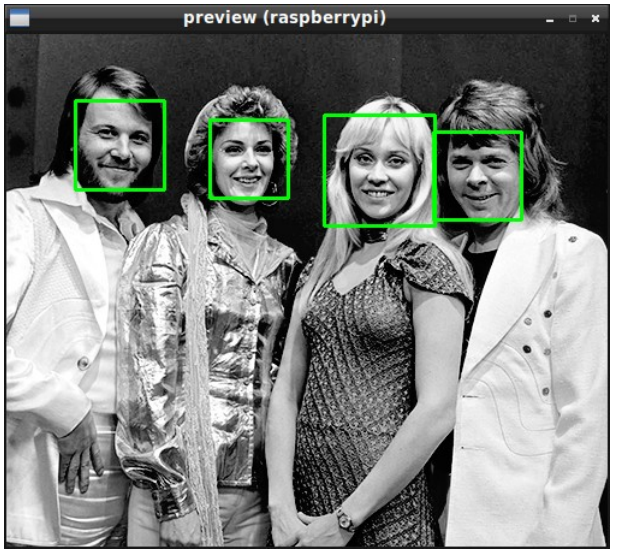
https://www.mathworks.com/help/vision/ref/vision.cascadeobjectdetector-class.html

scaleFactor

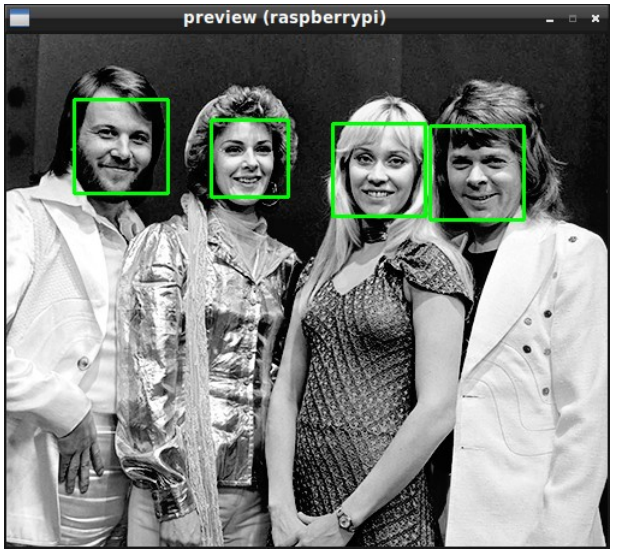
- minNeighbors=5, minSize=(30, 30)



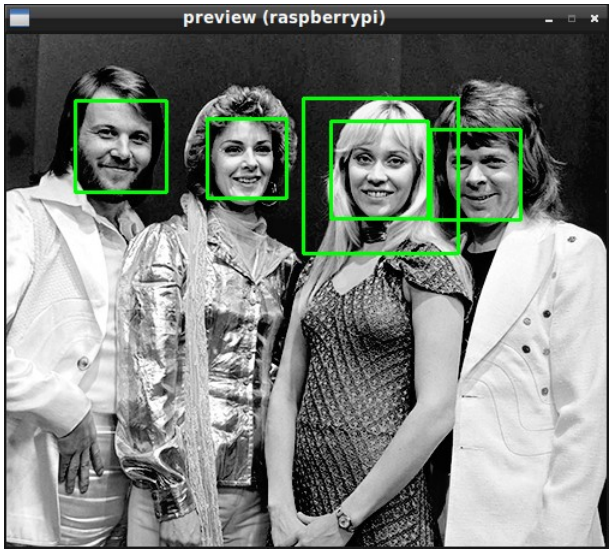
scaleFactor=1.1



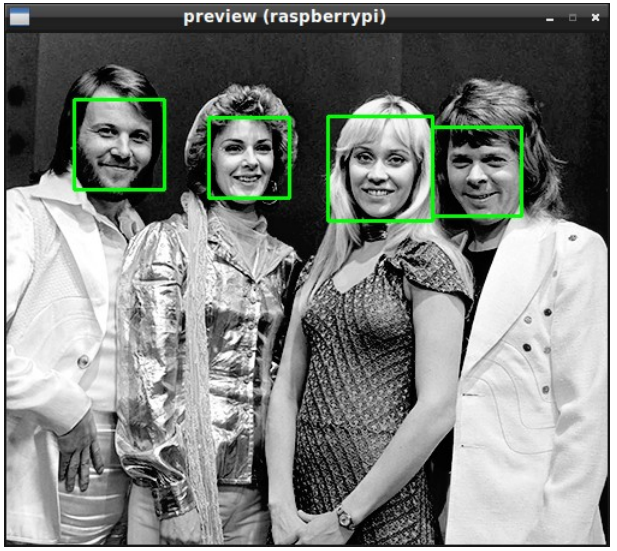
1.2



1.3



1.4

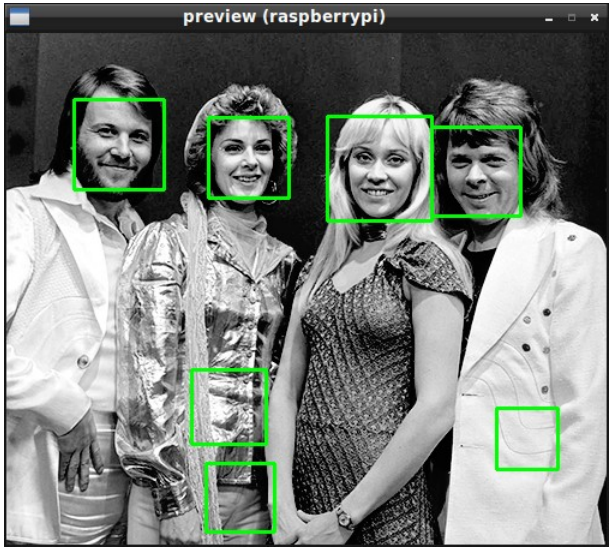


1.5

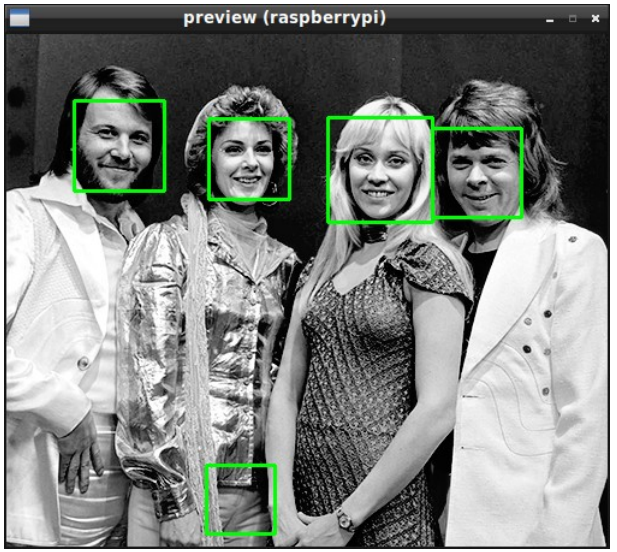
1.6

minNeighbors

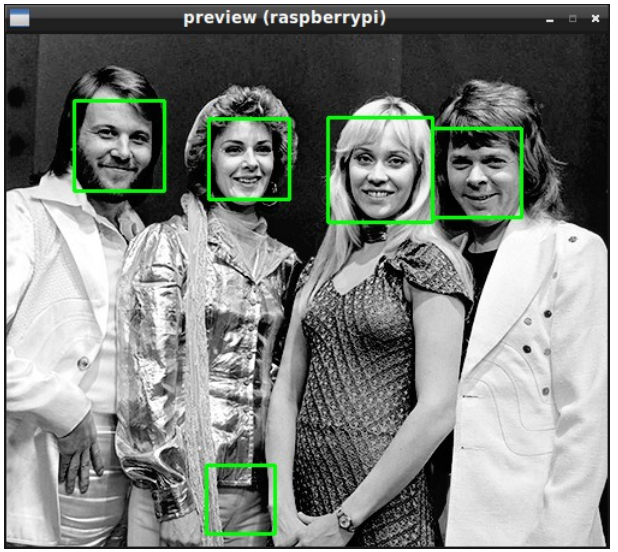
- scaleFactor=1.1, minSize=(30, 30)



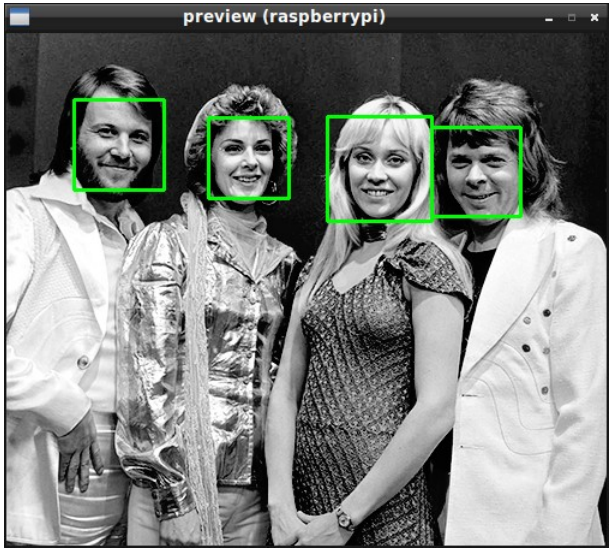
minNeighbors=1



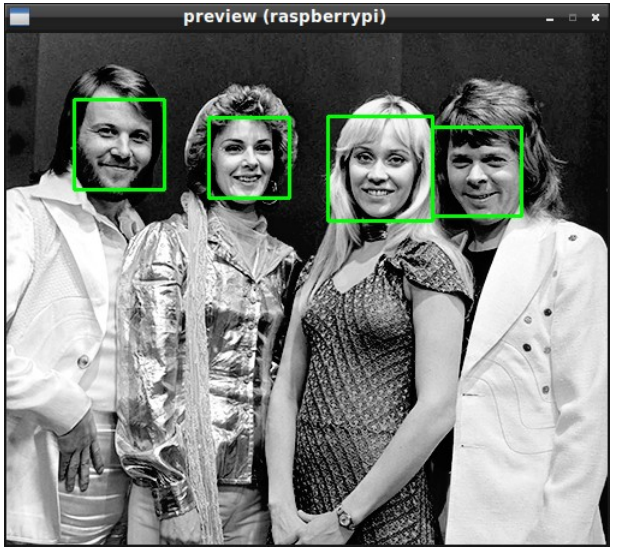
2



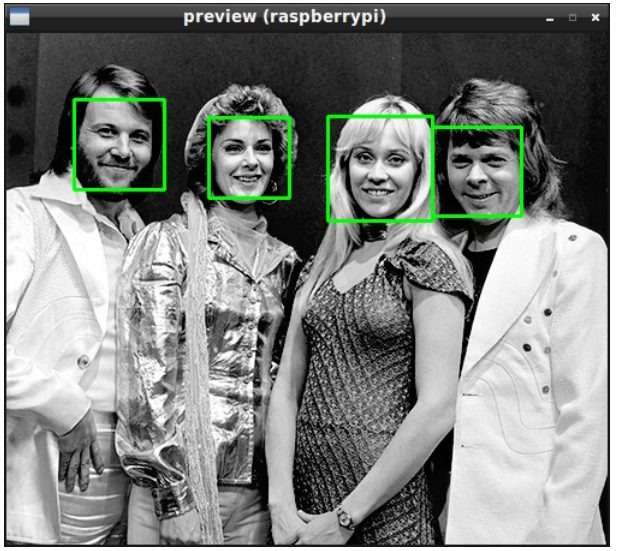
3



5



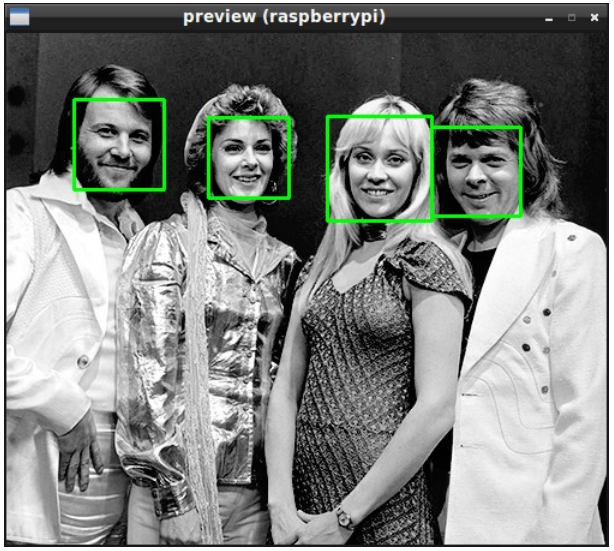
10



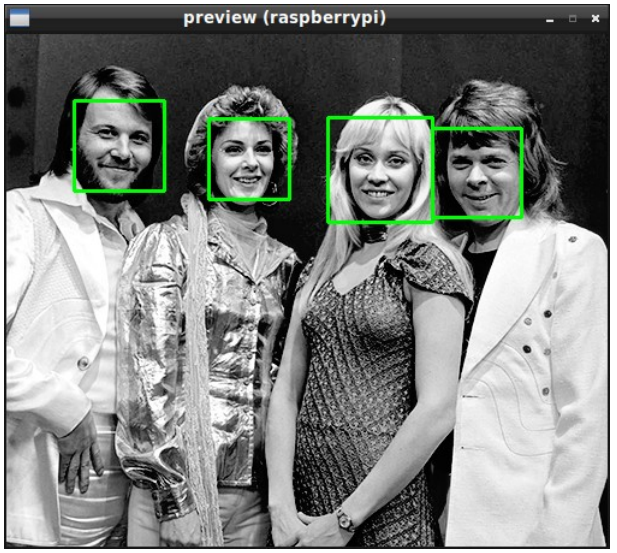
20

minSize(x, y)

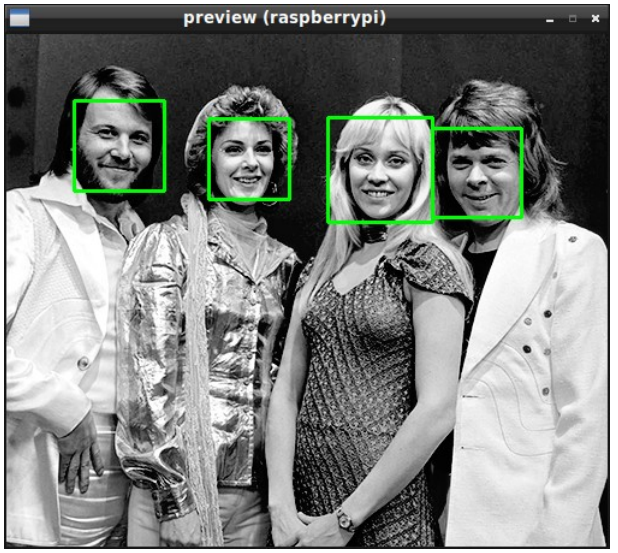
- scaleFactor=1.1, minNeighbosr=5



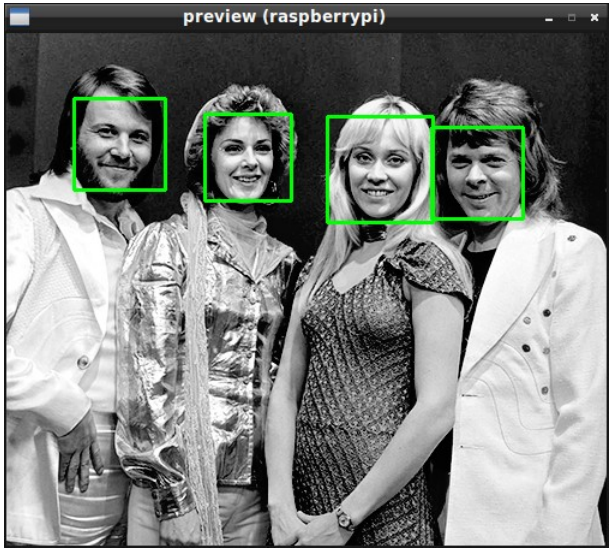
minSize=(15, 15)



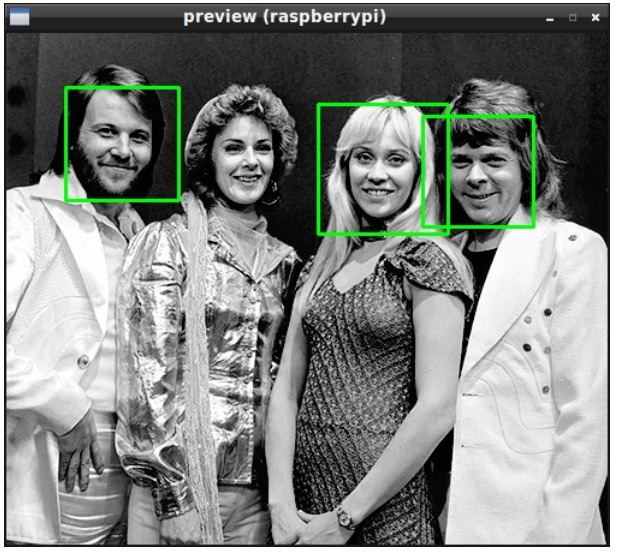
(30, 30)



(60, 60)



(90, 90)

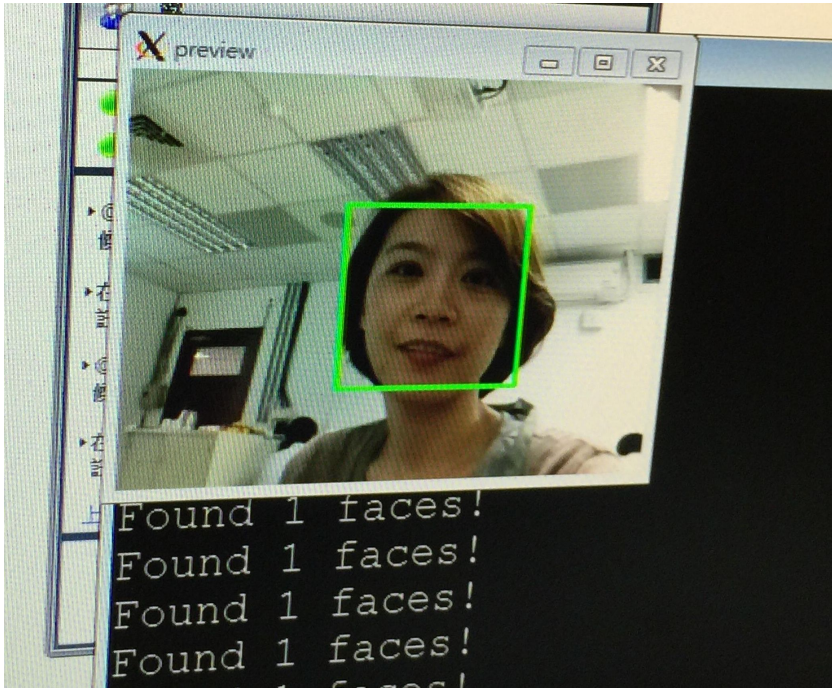


(120 ,120)

(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 ~/cam-py-cv/day2/03-face_datection
$ python camera_face_detect.py haarcascade_frontalface_default.xml
```

練習

- 將人臉辨識用在影像串流上 (修改 camera_pi.py)
- 請參考 Camera+Python 投影片

使用前記得要先載入模組
$ sudo modprobe bcm2835-v4l2

實驗 4 : KNN 最近鄰居法

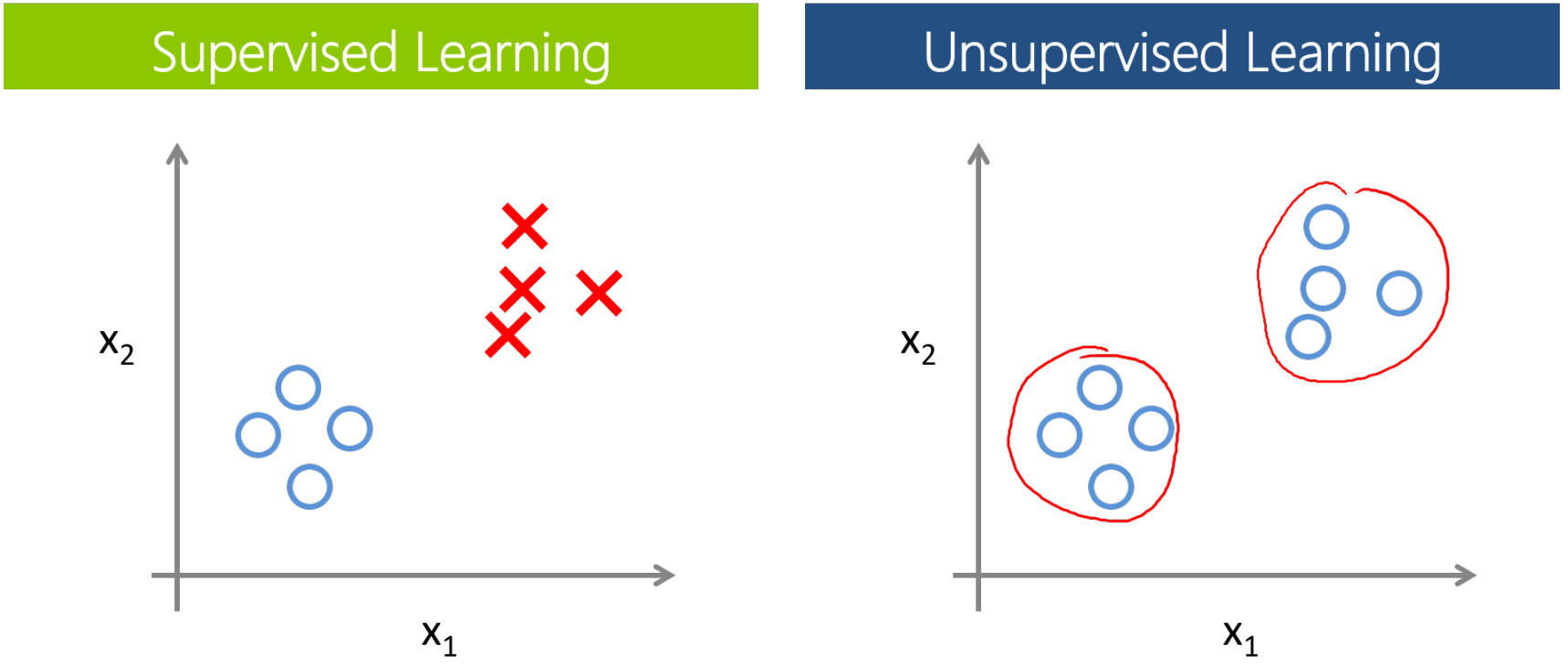
目的 : OpenCV 機器學習的 OCR

圖形分類

- 機器學習介紹
- kNN 演算法介紹
- 圖形分類介紹

機器學習方式

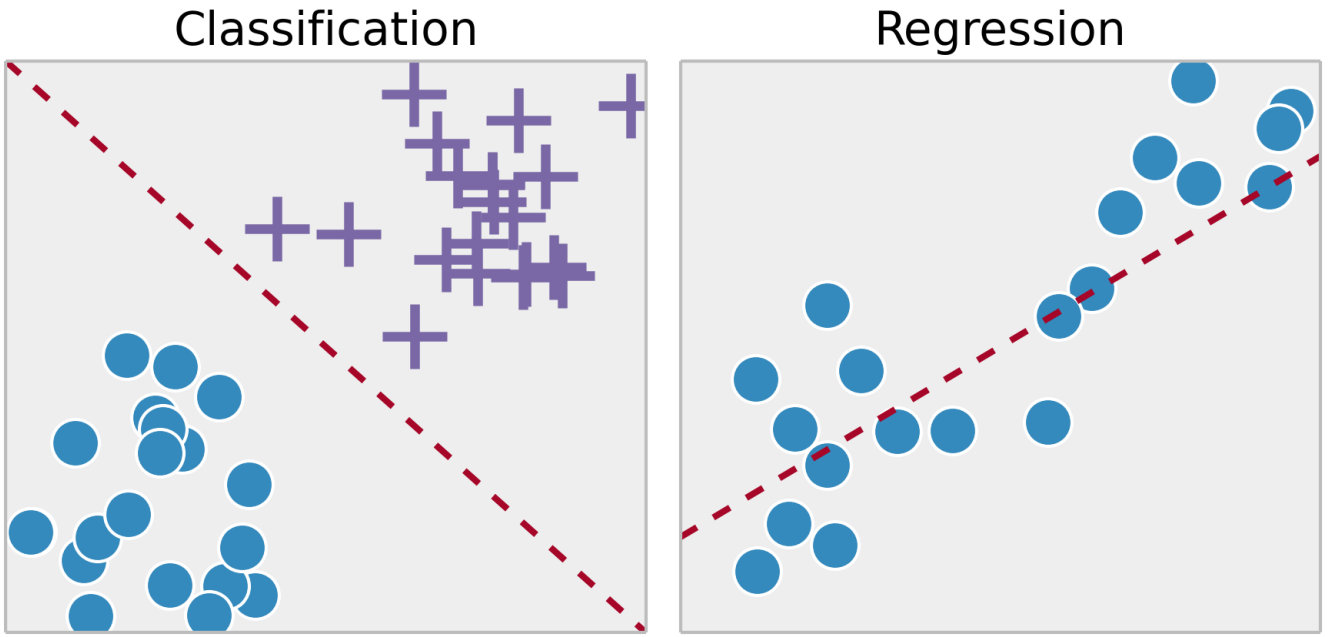
- 監督式學習 (supervised learning)
- 非監督式學習 (unsupervised learning)



https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/machine_learning.html

監督式學習

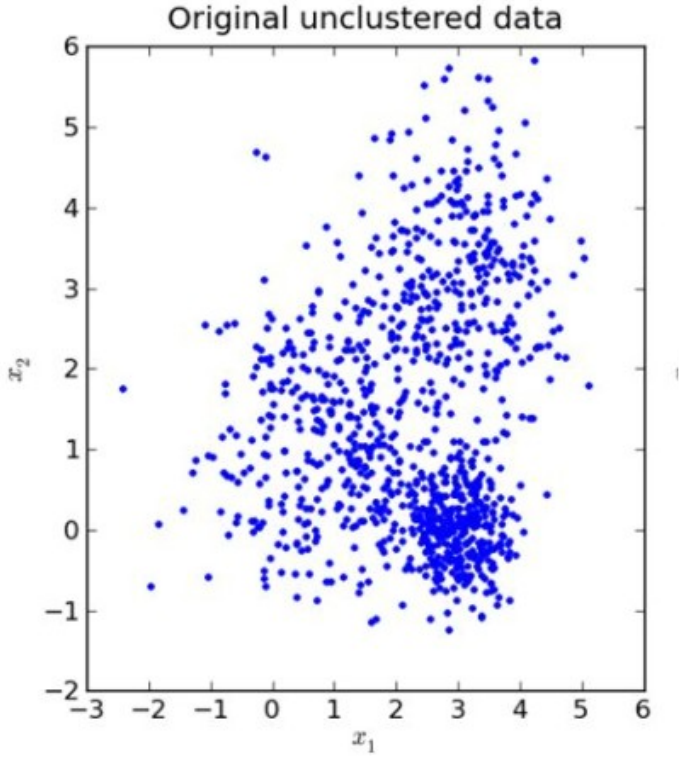
- 特色：有標準答案，每個資料都有明確的標籤
- 應用：分類(classification)，預測(prediction)
- 使用時機：根據輸入資料推測未來結果



https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/machine_learning.html

非監督式學習

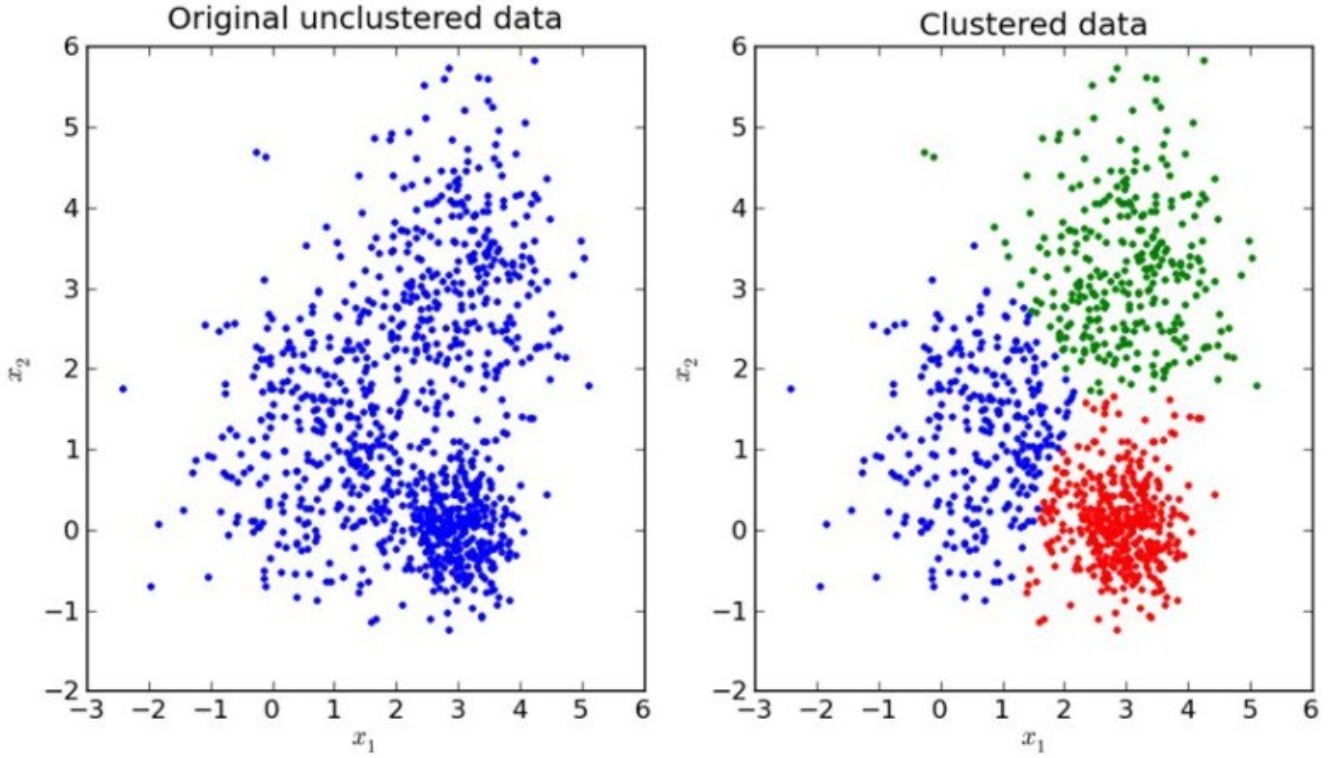
- 特色：沒有標準答案
- 應用：分群 (clustering)
- 使用時機：對資料還無法掌握，或是降低資料維度



http://www.frankichamaki.com/data-driven-market-segmentation-more-effective-marketing-to-segments-using-ai/

非監督式學習

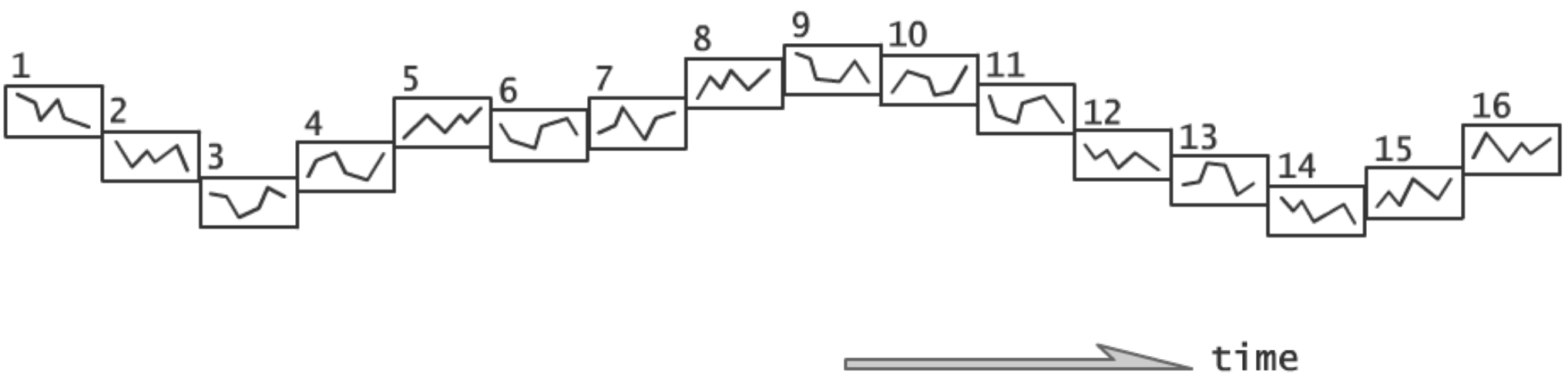
- 特色：沒有標準答案
- 應用：分群 (clustering)
- 使用時機：對資料還無法掌握，或是降低資料維度



http://www.frankichamaki.com/data-driven-market-segmentation-more-effective-marketing-to-segments-using-ai/

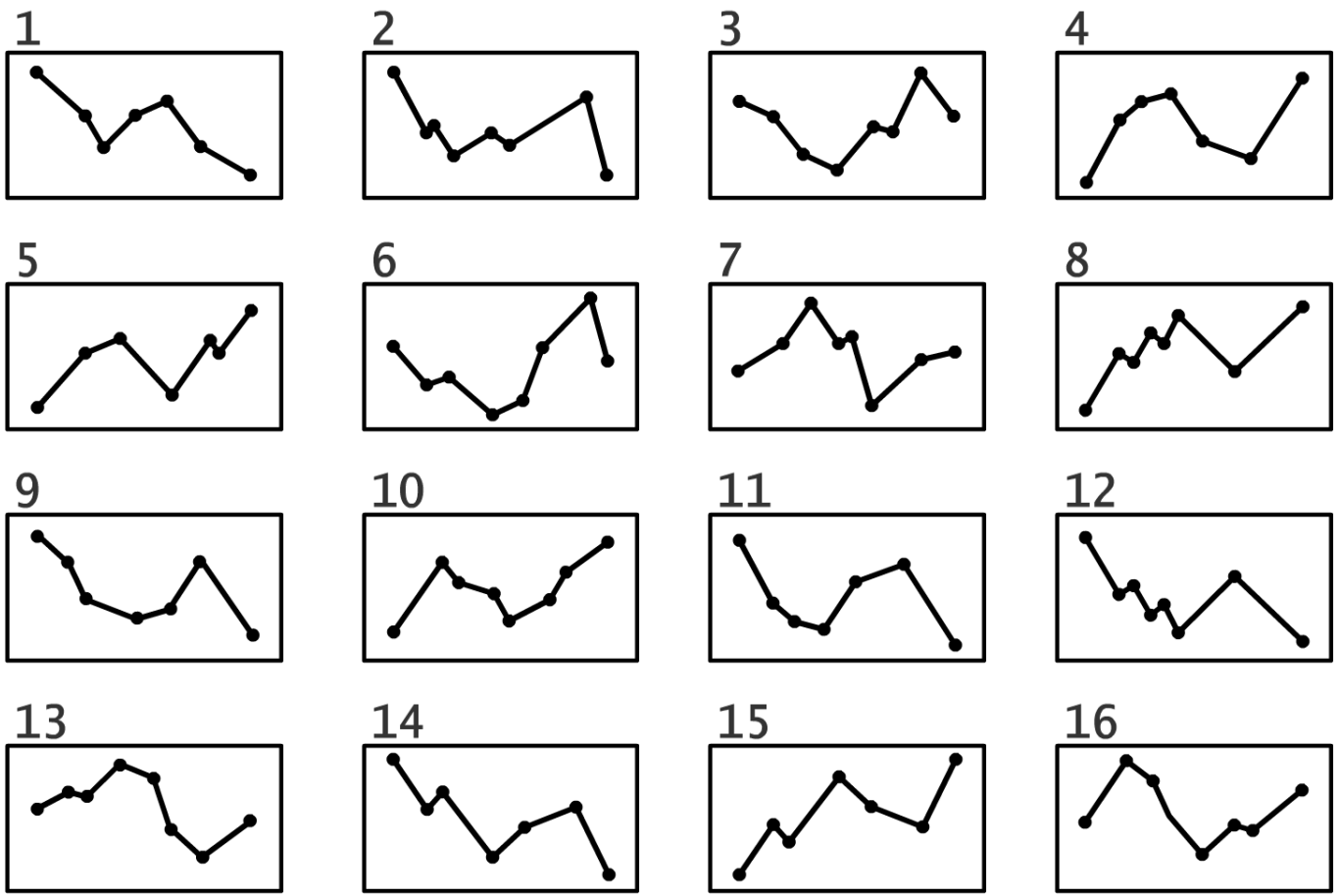
從股市線圖瞭解非監督式學習

- 每日的股市指數線圖為時間序列資料



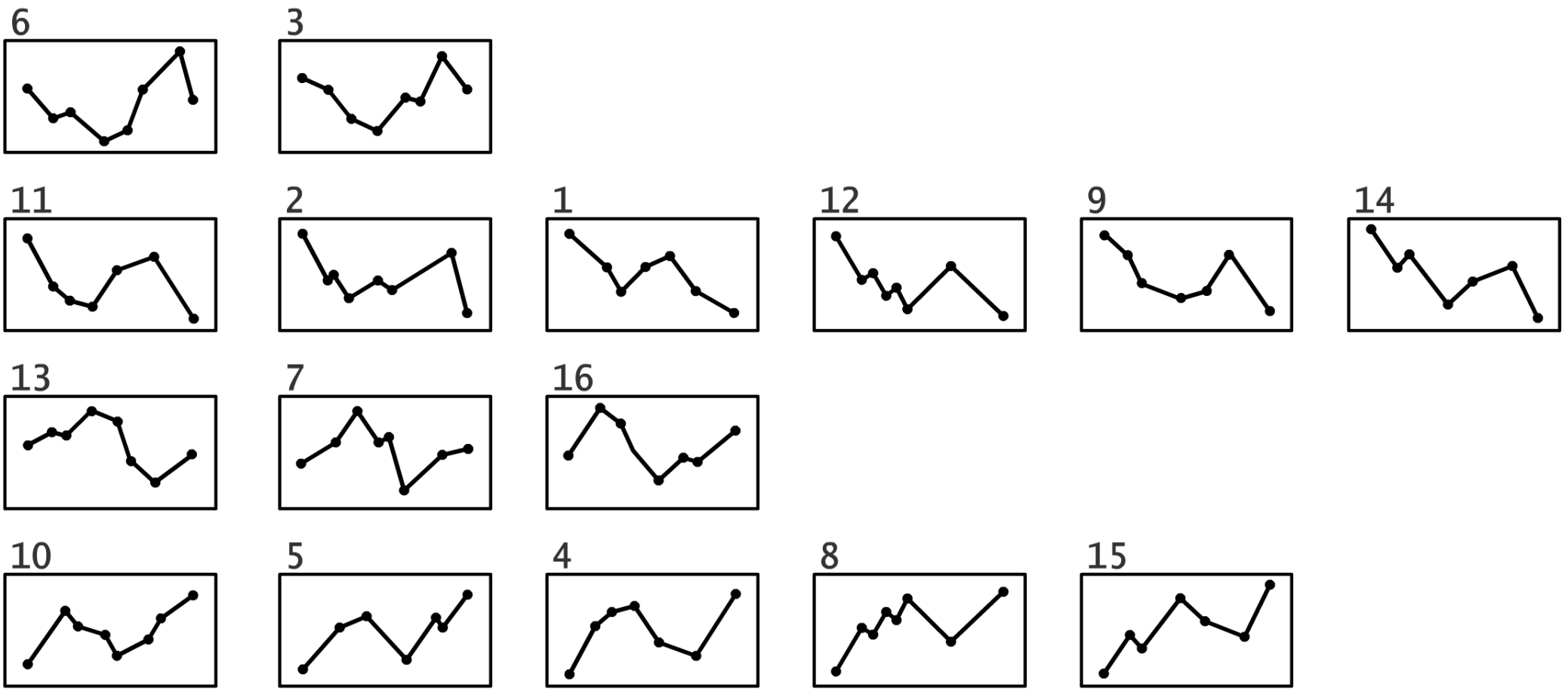
http://www.slideshare.net/sosorry/an-extended-twophase-architecture-for-mining-time-series-data

原始數據資料量太大難以分析



http://www.slideshare.net/sosorry/an-extended-twophase-architecture-for-mining-time-series-data

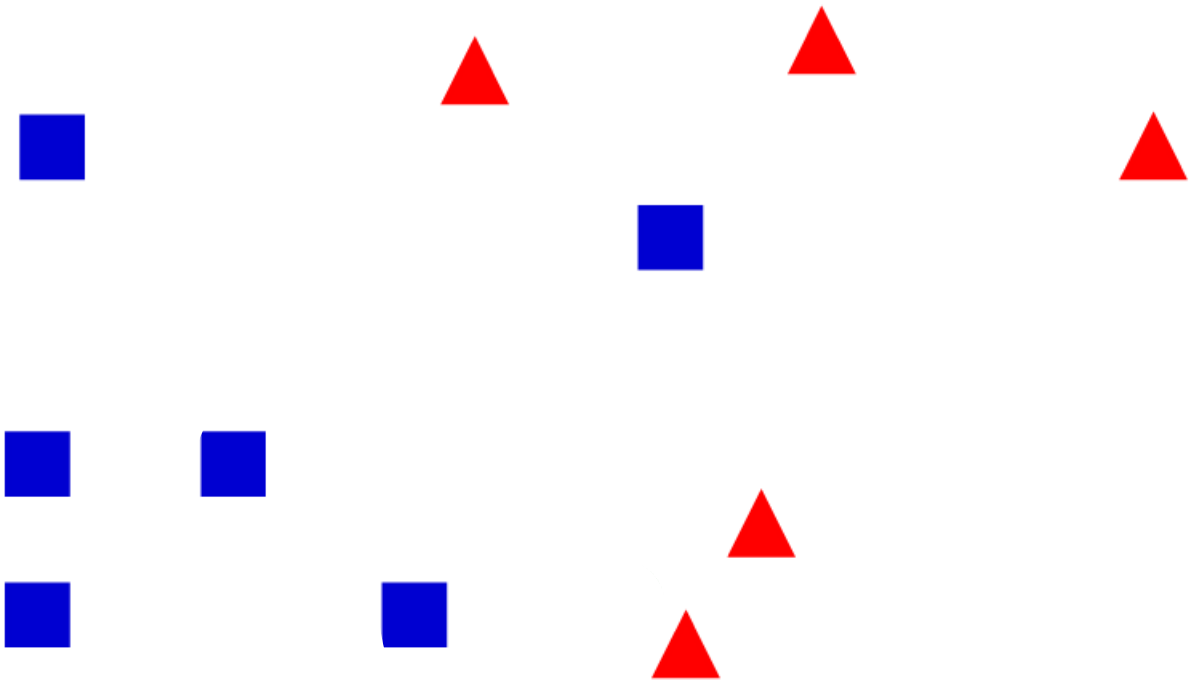
先分群以後幫助後續的資料處理



http://www.slideshare.net/sosorry/an-extended-twophase-architecture-for-mining-time-series-data

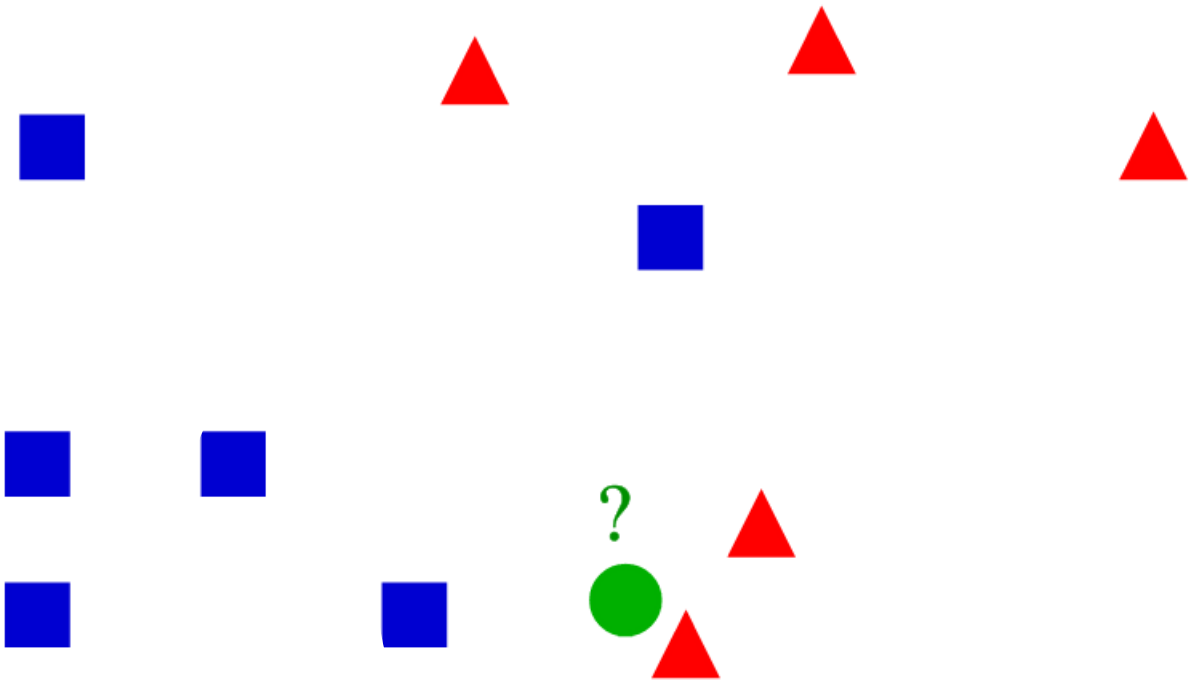
kNN = k-Nearest Neighbors
（非監督式學習）

在已經分類好的結果



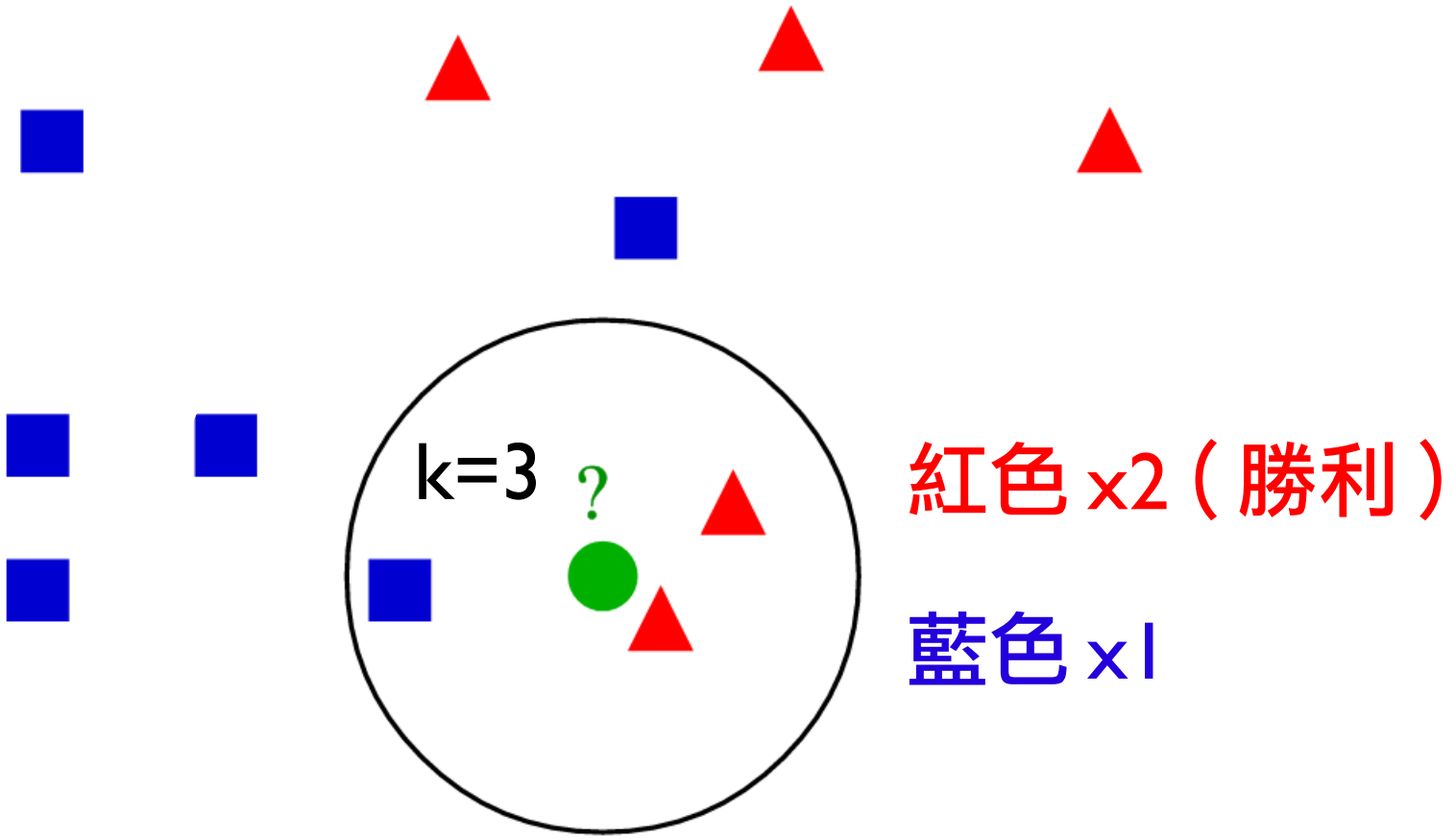
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

加入新資料 & 找 k 個最近的鄰居



https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

靠近的鄰居數目決定分類結果



https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

演算法

- 步驟：
  1. 根據距離找出最近的 k 個鄰居
  2. 根據鄰居分類決定資料結果

Input: D, the set of k training objects, and test object $z = (\mathbf{x}', y')$

Process:

Compute $d(\mathbf{x}', \mathbf{x})$, the distance between z and every object, $(\mathbf{x}, y) \in D$.

Select $D_z \subseteq D$, the set of k closest training objects to z.

Output: $y' = \underset{v}{\operatorname{argmax}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(v = y_i)$

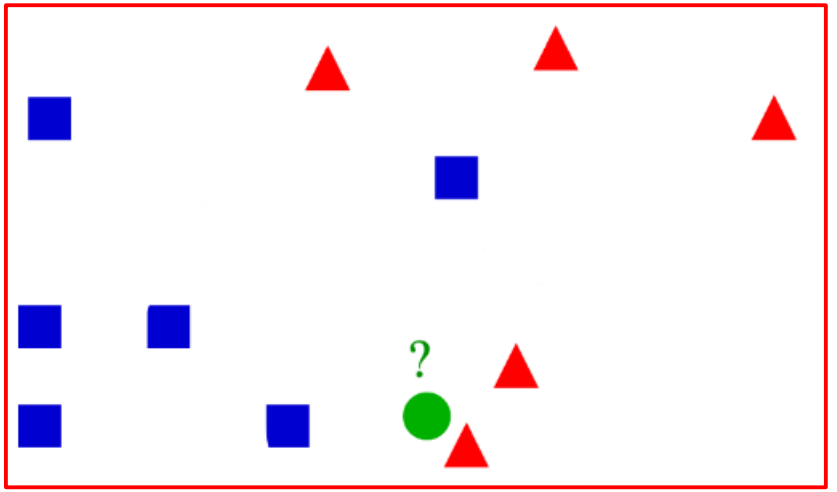
https://mariuszprzydatek.com/2014/05/27/k-nearest-neighbors-knn-algorithm-machine-learning/

視覺化二維資料分群

固定資料集

Training Data		
x	y	label
1.1	1.1	1
1.2	2.2	1
1.4	4.8	1
3.1	2.3	1
4.7	1.0	1
4.9	6.1	0
6.3	0.7	0
6.1	6.5	1
6.8	1.8	0
7.4	6.8	0
9.7	5.8	0

Test Data		
x	y	label
6	1	?



label=0 => Red Triangle

label=1 => Blue Rectangle

k-Nearest Neighbors 建構子

- `knn = cv2.KNearest()`

http://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html

訓練模型

- `knn.train(train_data, train_label)`
  - `train_data`: 特徵空間 (型態為 `float`)
  - `train_label`: 分類標籤

- 執行 `train()` 以後會維護一個搜尋樹結構

http://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html

找出最近的鄰居與分類

- `knn.find_nearest(newcomer, k)`
  - `newcomer`：要預測的資料
  - `k`：找到最近的鄰居數量

http://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html

根據特徵空間和標籤預測新進資料

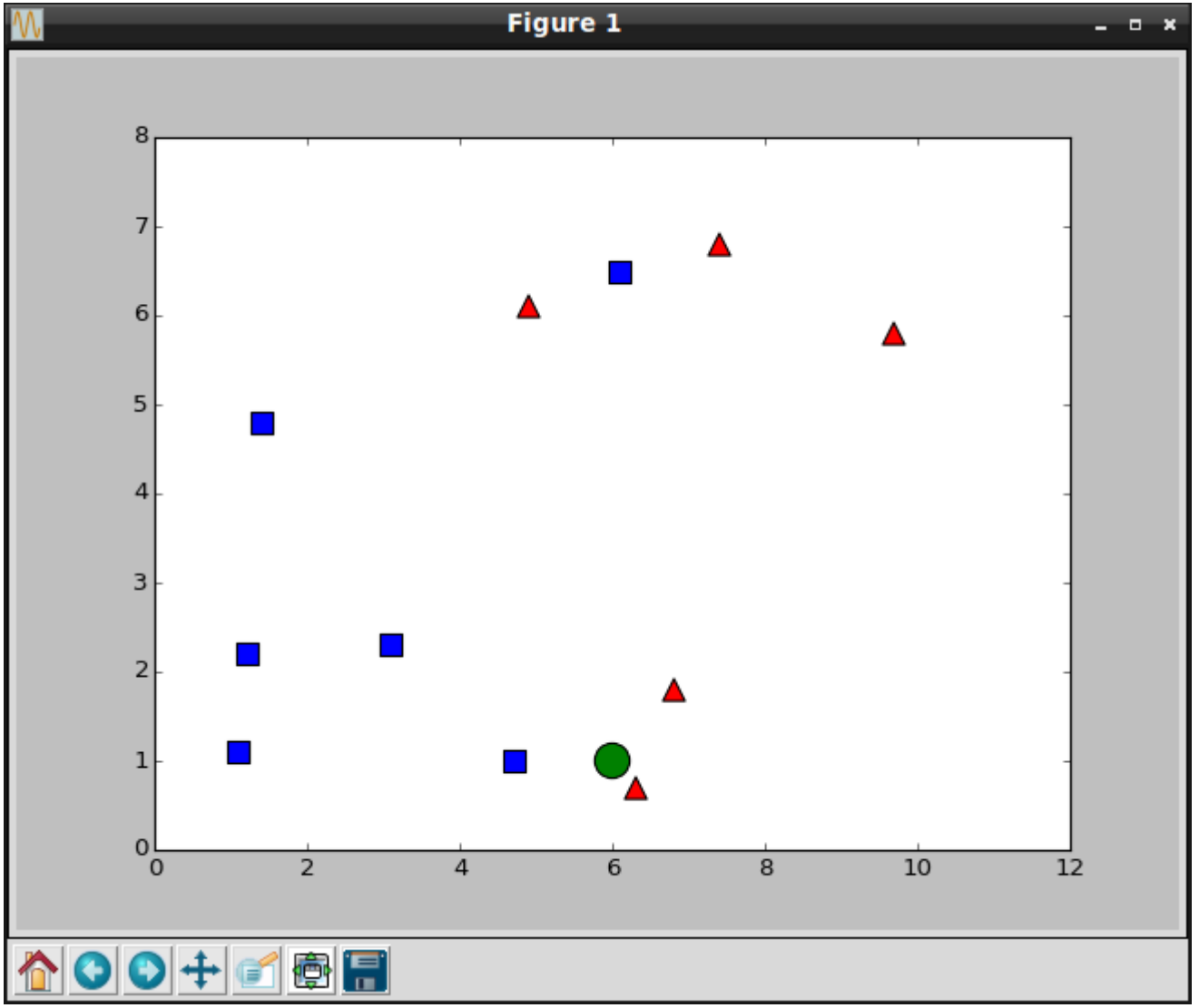
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

train = np.array([[ 1.1, 1.1], ...)
train_labels = np.array([[ 1.], ...)
...

knn = cv2.KNearest()
knn.train(train, train_labels)
ret, results, neighbors, dist = knn.find_nearest(newcomer, 3)

plt.show()
```

http://docs.opencv.org/2.4/modules/ml/doc/k_nearest_neighbors.html

knn_static.py

- 新進資料 [5, 8] 被分類到標籤 0(紅色三角形)

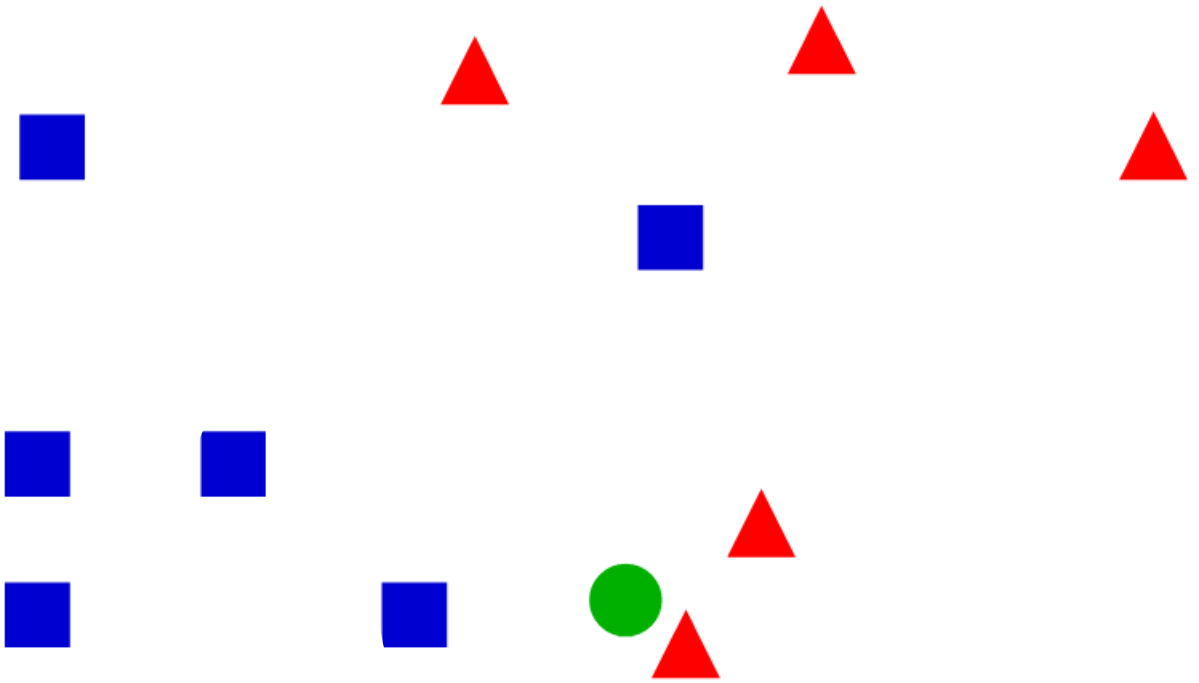
視覺化二維資料



DEMO
knn_static.py

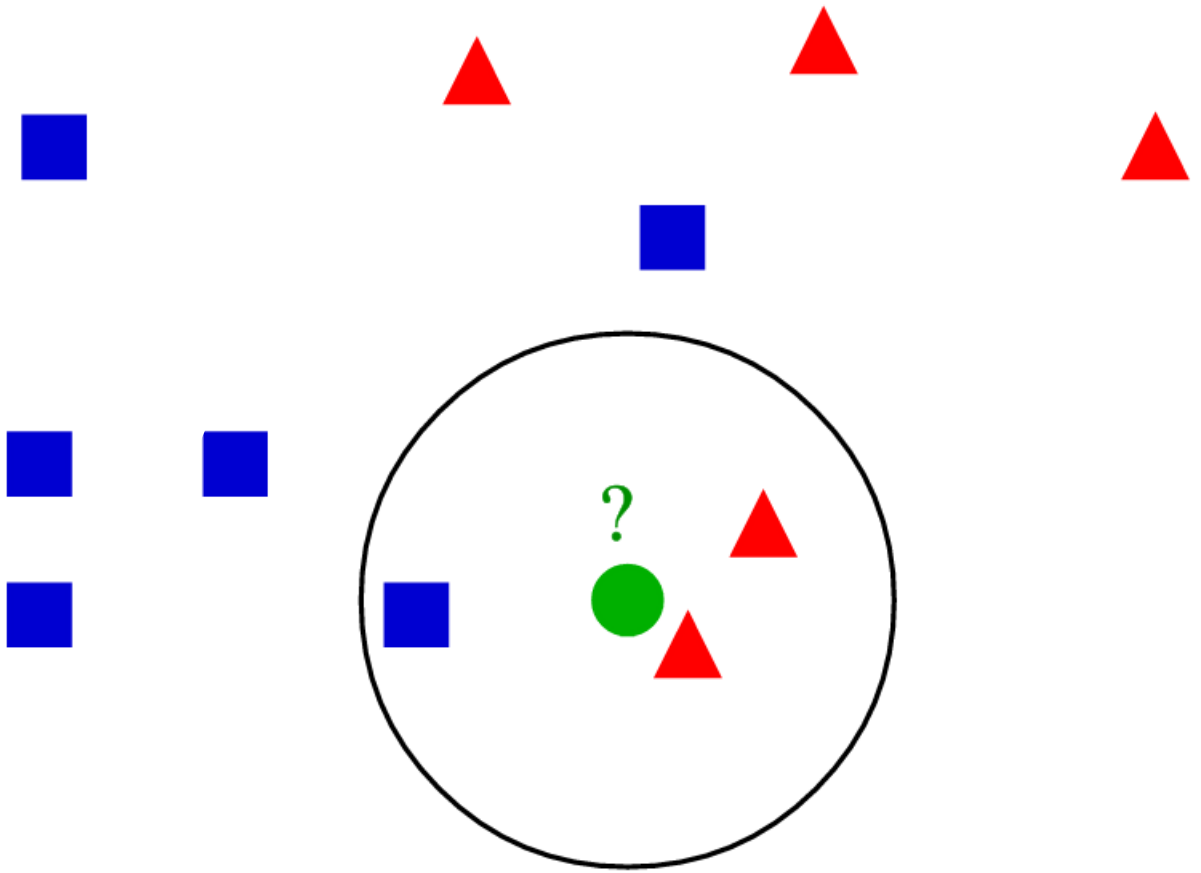
```
$ cd ~/cam-py-cv/day2/04-knn_ocr
$ python knn_static.py
```

找 k 個最近的鄰居



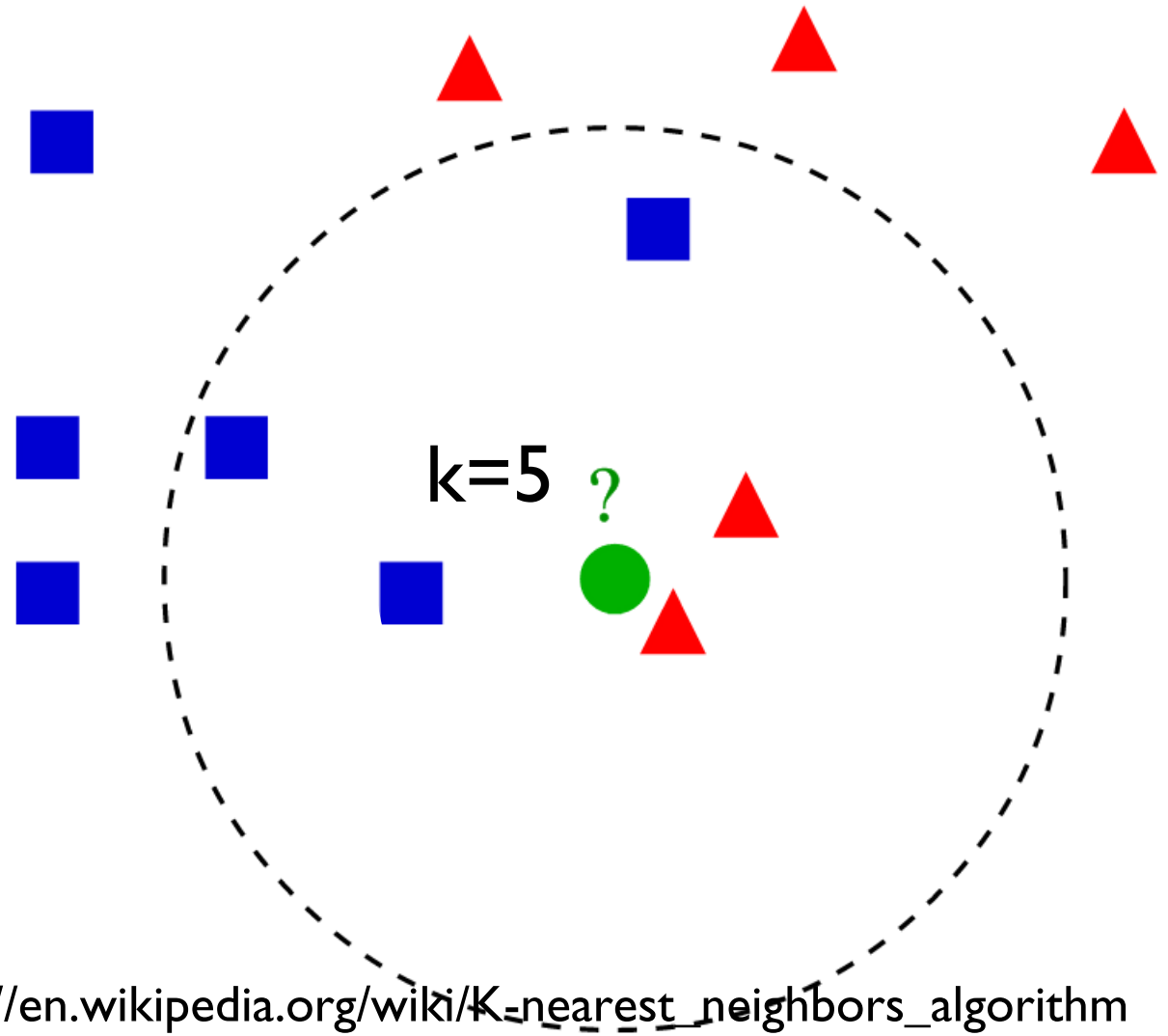
https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

k=3，應該是紅色吧？



https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

那如果 k=5 ？



https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

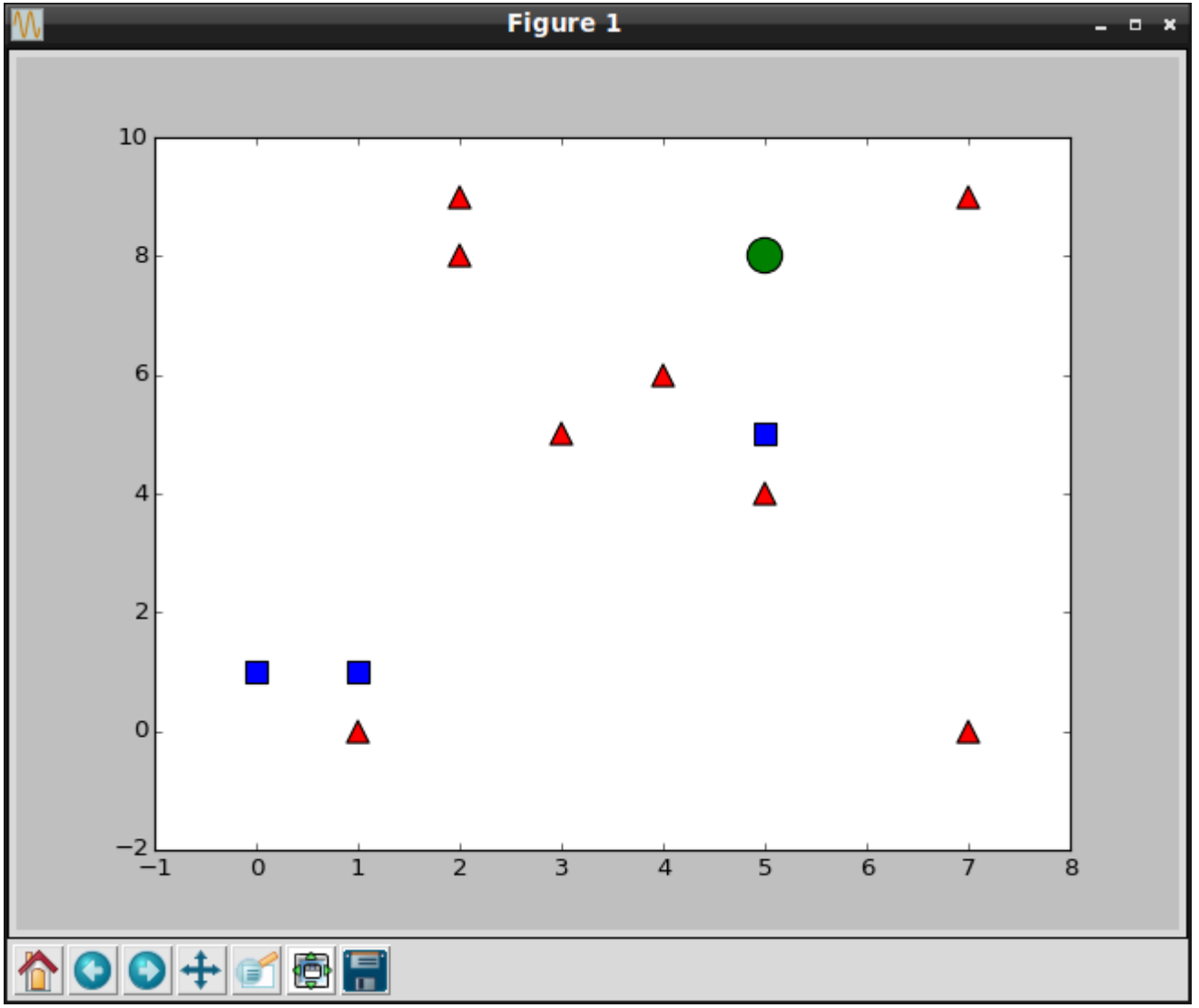
動態資料集

- 訓練資料
  - `train = np.random.randint(0,10,(11,2)).astype(np.float32)`
- 訓練標籤
  - `train_labels = np.random.randint(0, 2,(11,1)).astype(np.float32)`
- 新進資料
  - `newcomer = np.random.randint(0,10,(1, 2)).astype(np.float32)`

knn_random.py (每次都會不一樣)

- 新進資料 [5, 8] 被分類到標籤 0(紅色三角形)

視覺化二維資料 (每次都會不一樣)

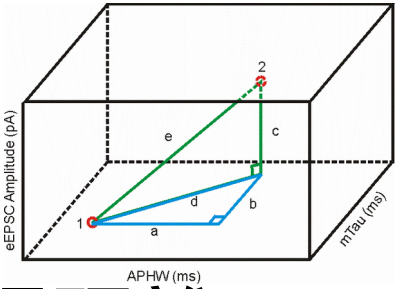


DEMO
knn_random.py

```
$ cd ~/cam-py-cv/day2/04-knn_ocr
$ python knn_random.py
```

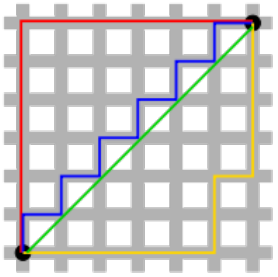
常用距離演算法

- 歐幾里得距離 (Euclidean Distance) => 預設



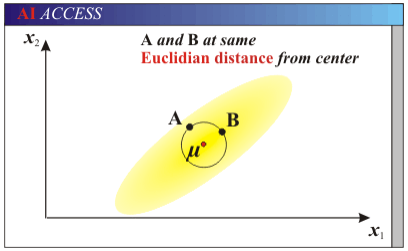
$$d_{12} = \sqrt{\sum_{k=1}^{n} (x_{1k} - x_{2k})^2}$$

- 曼哈頓距離 (Manhattan Distance)



$$d_{12} = \sum_{k=1}^{n} |x_{1k} - x_{2k}|$$

- 馬哈蘭距離 (Mahalanobis Distance)

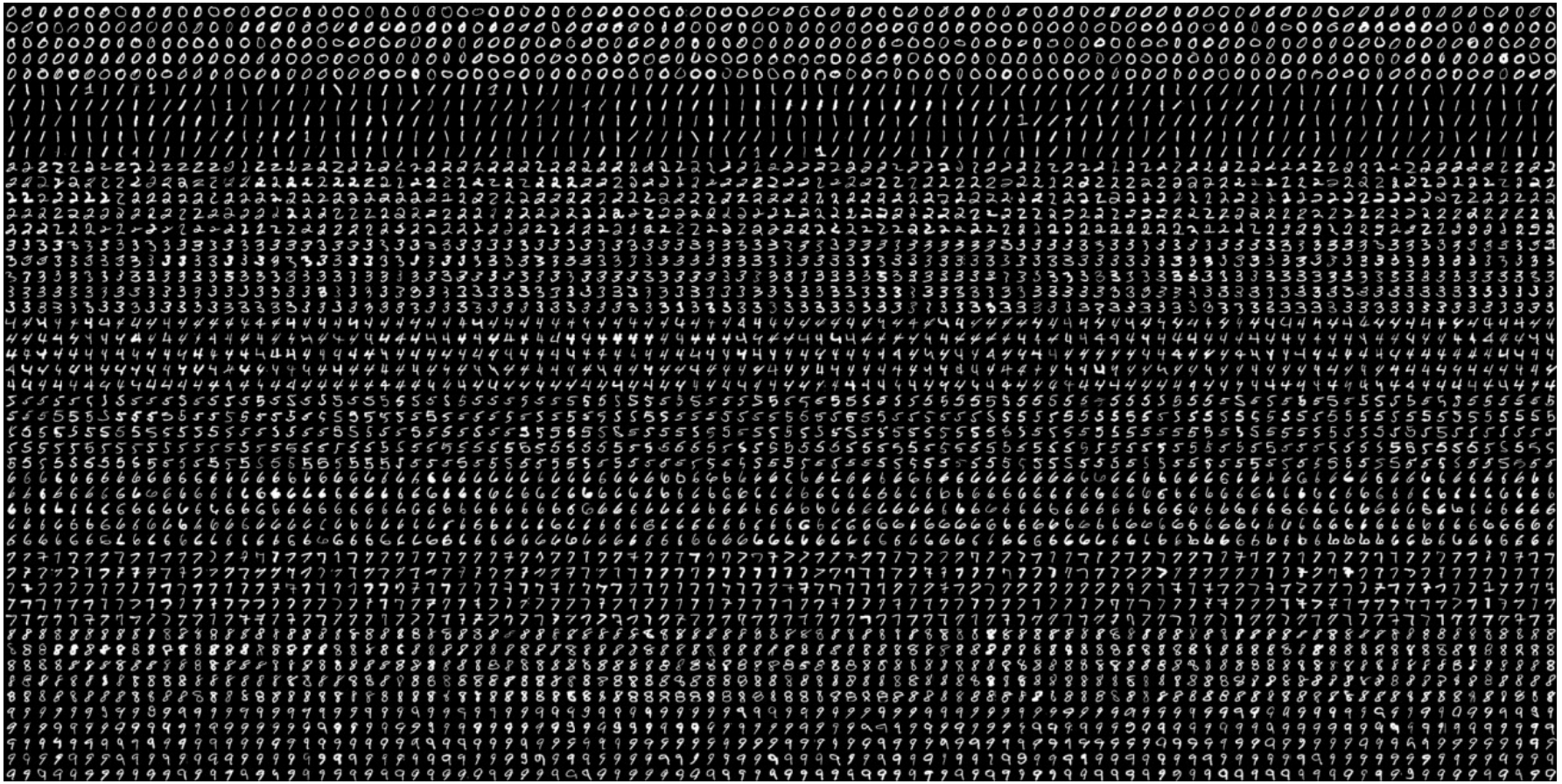


$$D(X_i, X_j) = \sqrt{(X_i - X_j)^T (X_i - X_j)}$$

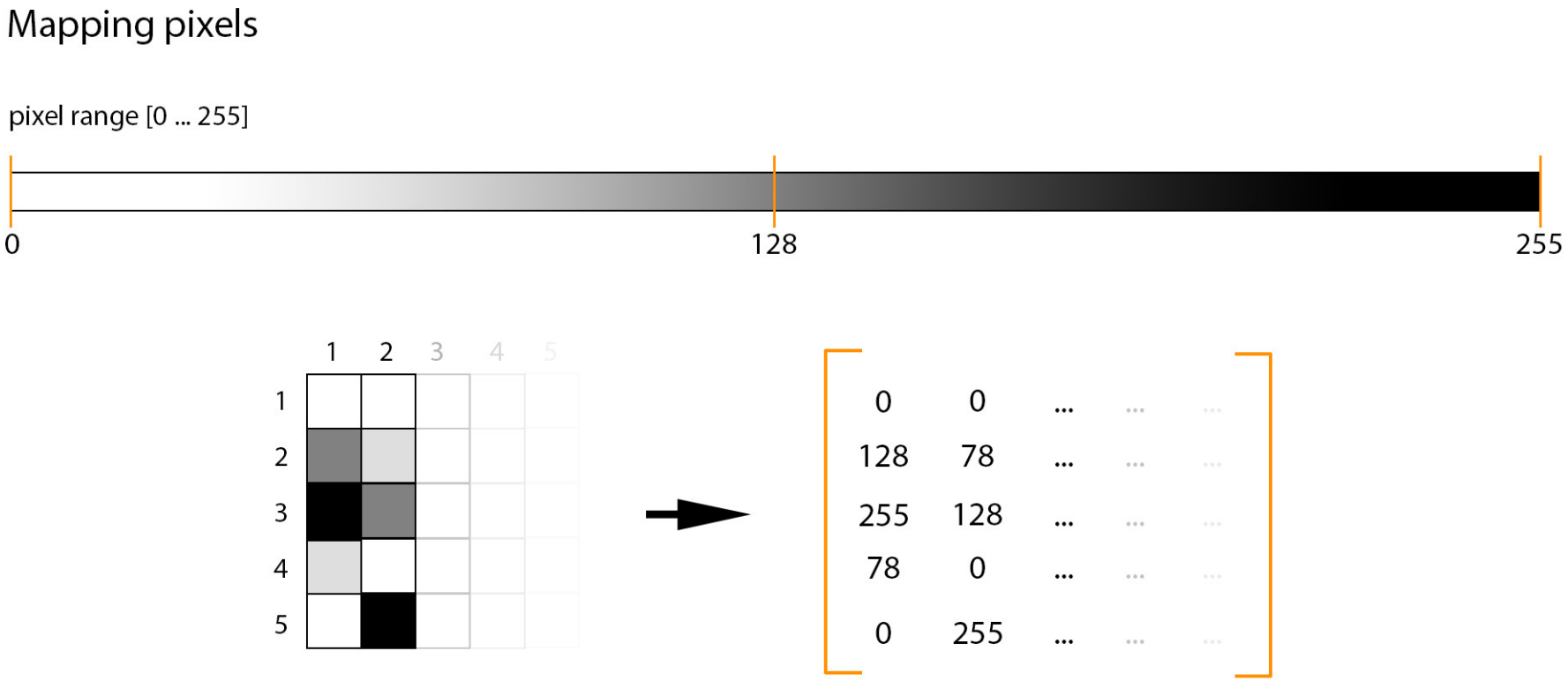
根據範例資料訓練與分類

OpenCV 原始檔內建手寫圖檔

- opencv2.4/samples/python2/data/digits.png
- 圖檔 2000x1000，每個數字佔 20x20，共 5000 個數字



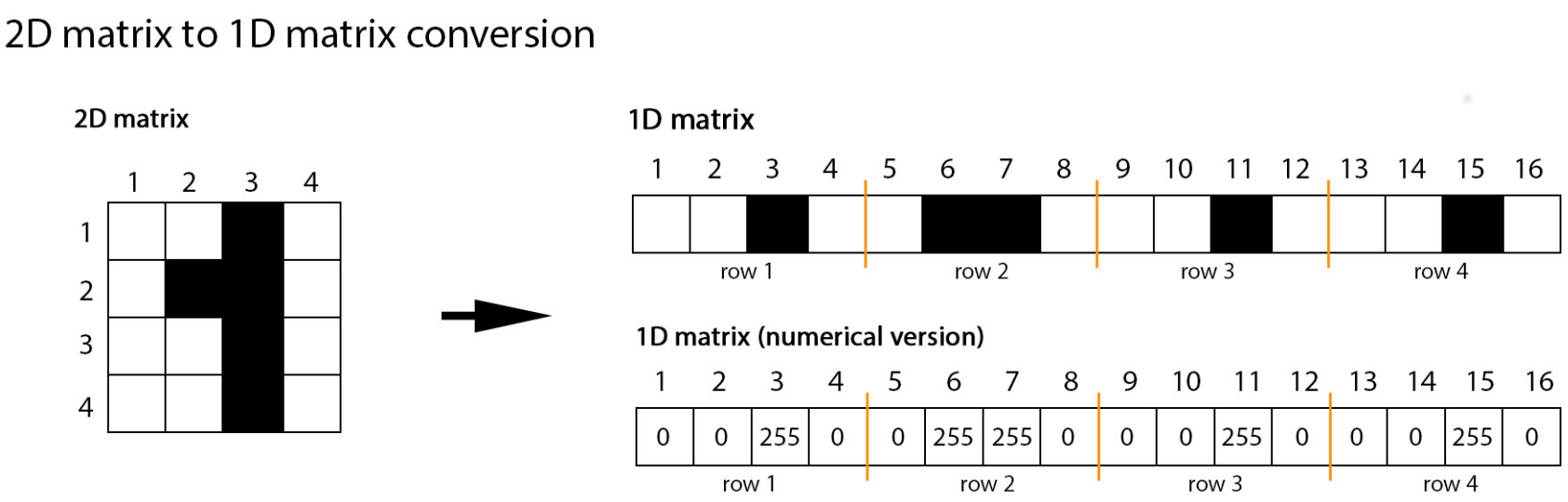
圖檔就是灰階矩陣



https://www.simplicity.be/article/recognizing-handwritten-digits/

2D 轉 1D

- 將外部載入圖檔轉為一維陣列



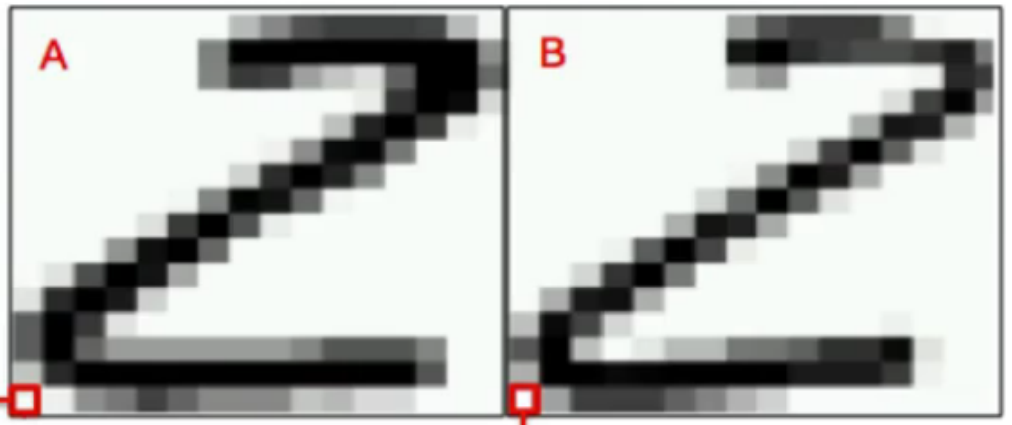

```C++
// read image file (grayscale)
cv::Mat 2dMat = cv::imread("test.jpg", 0);

// convert 2d to 1d
cv::Mat 1Mat = 2dMat.clone().reshape(1,1);
```

https://www.simplicity.be/article/recognizing-handwritten-digits/

計算相似度

- 16x16 bitmaps
- 8-bit grayscale
- Euclidian distance
  - over raw pixels

$$D(A,B)=\sqrt{\sum_r \sum_c \left(A_{r,c}-B_{r,c}\right)^2}$$



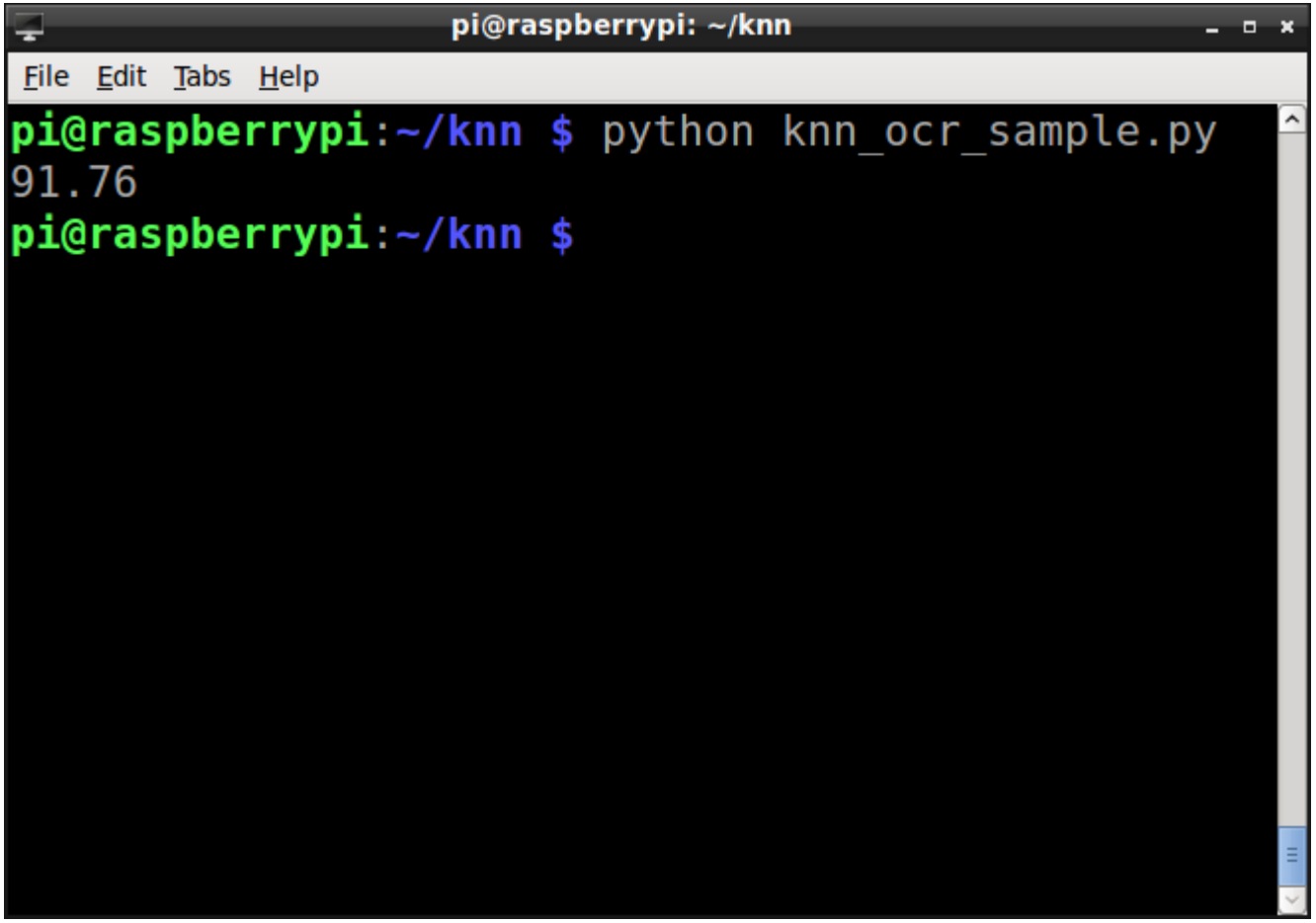
https://www.youtube.com/watch?v=ZD_tfNpKzHY

訓練和測試各佔 2500，切分後存檔

```
img = cv2.imread('data/digits.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

cells = [np.hsplit(row, 100) for row in np.vsplit(gray, 50)]
x = np.array(cells)
train = x[:, :50].reshape(-1, 400).astype(np.float32)
test  = x[:, 50:100].reshape(-1, 400).astype(np.float32)

k = np.arange(10)
train_labels = np.repeat(k,250)[:, np.newaxis]
test_labels = train_labels.copy()

knn = cv2.KNearest()
knn.train(train, train_labels)
ret, result, neighbours, dist = knn.find_nearest(test, k=5)

# Save the data
np.savez('knn_data.npz', train=train, train_labels=train_labels)
```

http://arbu00.blogspot.tw/2016/11/1-opencv-knn.html

knn_ocr_sample.py



- 測試資料準確度為 91.76

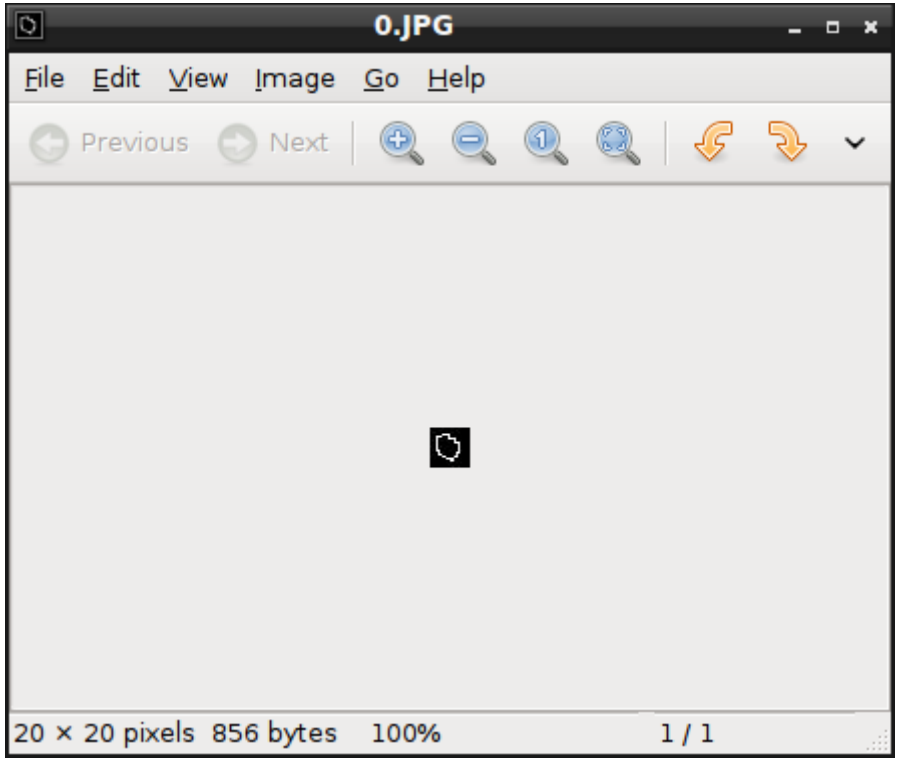
DEMO
knn_ocr_sample.py

```
$ cd ~/cam-py-cv/day2/04-knn_ocr
$ python knn_ocr_sample.py
```

根據範例資料測試手寫資料

使用小畫家新增手寫資料

- 每個數字佔 20x20，圖檔放在 data 目錄下
- 

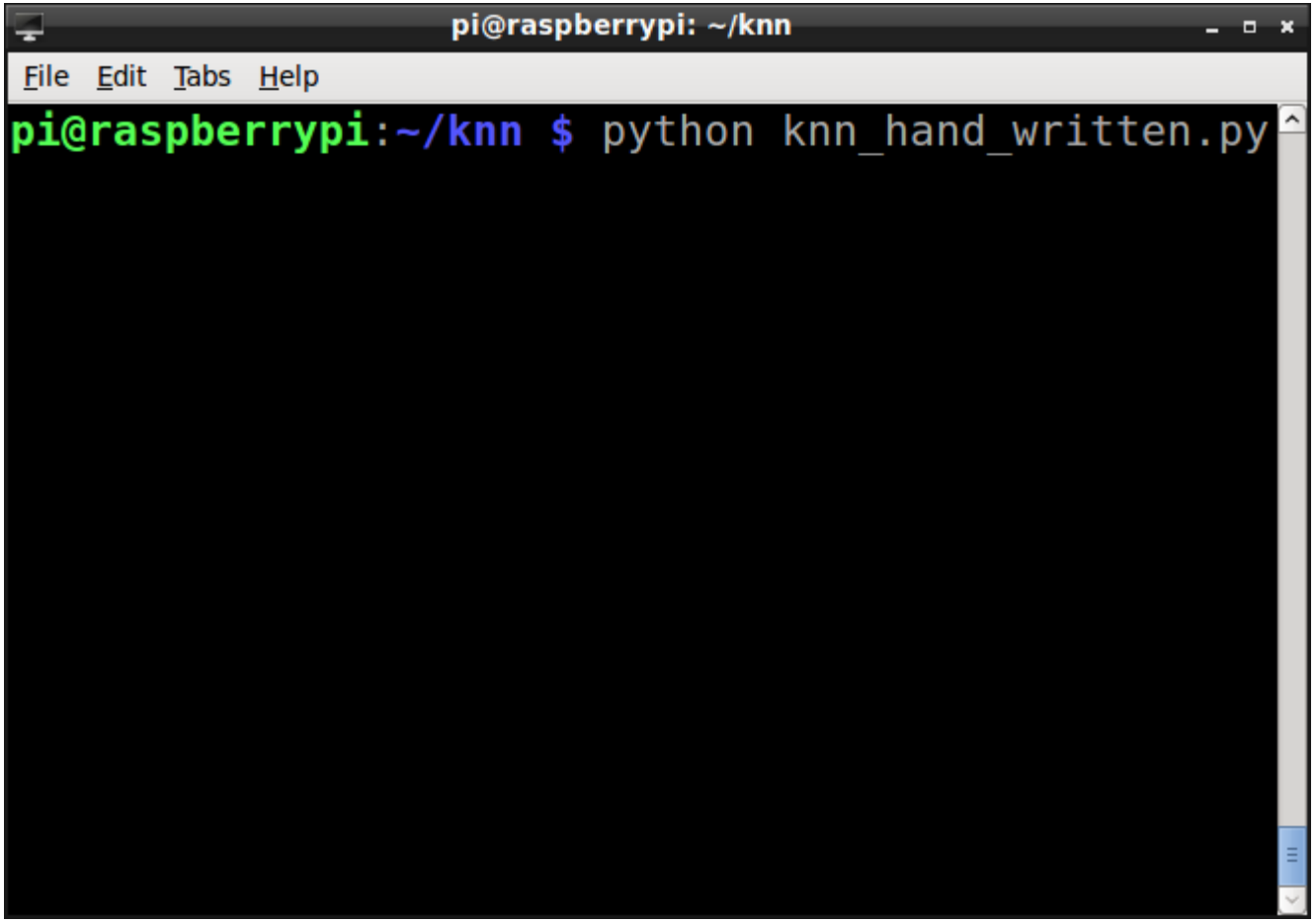


載入切分過的圖檔，對測試資料做 kNN

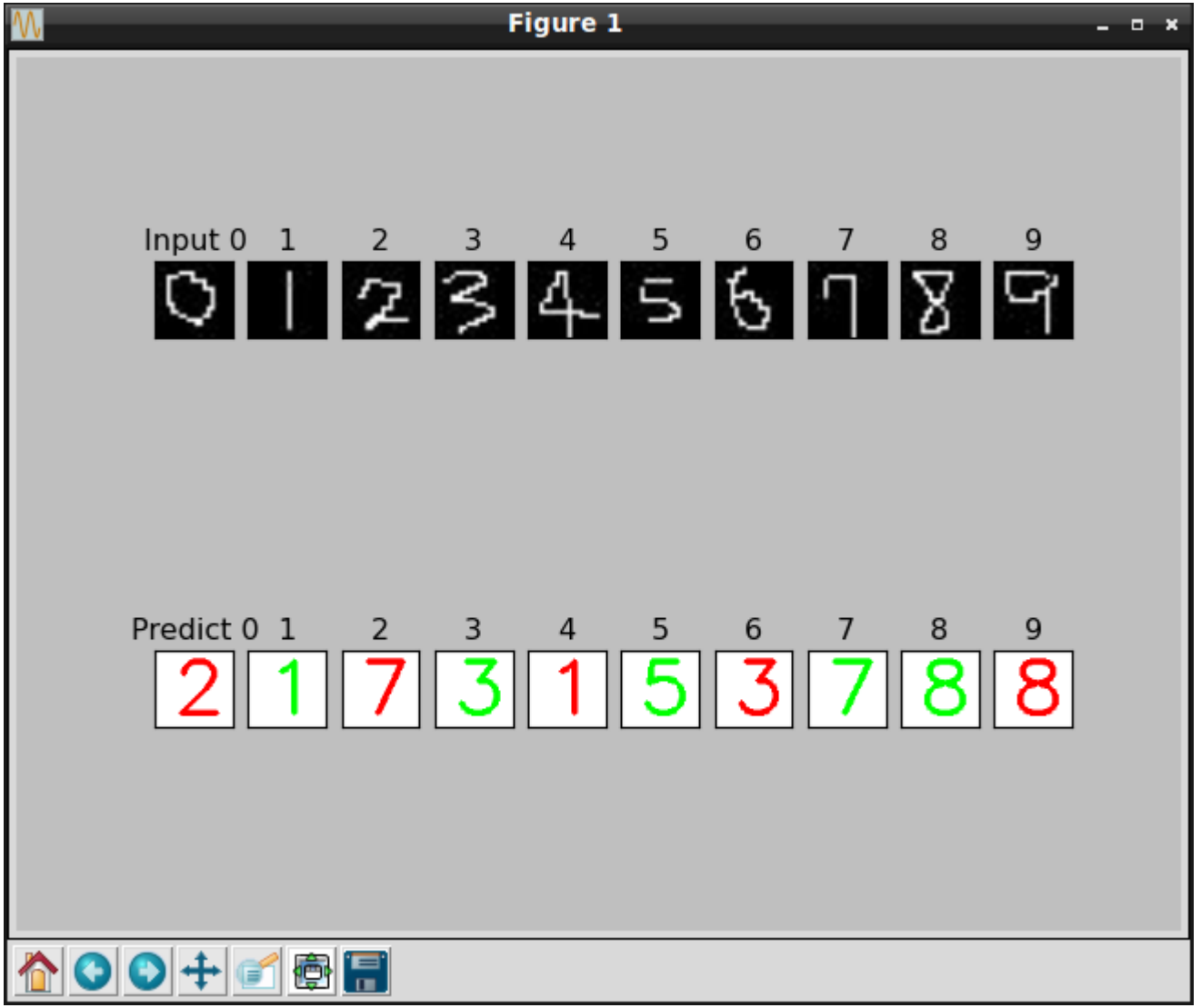
```
with np.load('knn_data.npz') as data:
    train = data['train']
    train_labels = data['train_labels']

knn = cv2.KNearest()
knn.train(train, train_labels)

for i in range(10):
    image[i] = cv2.imread( 'data/' + input_number[i], 0)
    test[i]  = image[i][:,:].reshape(-1, 400).astype(np.float32)
    ret, result[i], neighbours, dist = knn.find_nearest(test[i], k=5)
    image_result[i] = np.zeros((64, 64, 3), np.uint8)
    image_result[i][:,:] = [255, 255, 255]
    str[i] = str(result[i][0][0].astype(np.int32))

    if result[i][0][0].astype(np.int32) == i:
        cv2.putText(image_result[i], str[i], (15,52), font, 2, (0,255,0),3)
    else:
        cv2.putText(image_result[i], str[i], (15,52), font, 2, (255,0,0),3)
```

http://arbu00.blogspot.tw/2016/11/1-opencv-knn.html

knn_hand_written.py



分類結果



需要先執行過 knn_ocr_sample.py

DEMO
knn_hand_written.py

```
$ cd ~/cam-py-cv/day2/04-knn_ocr
$ python knn_hand_written.py
```

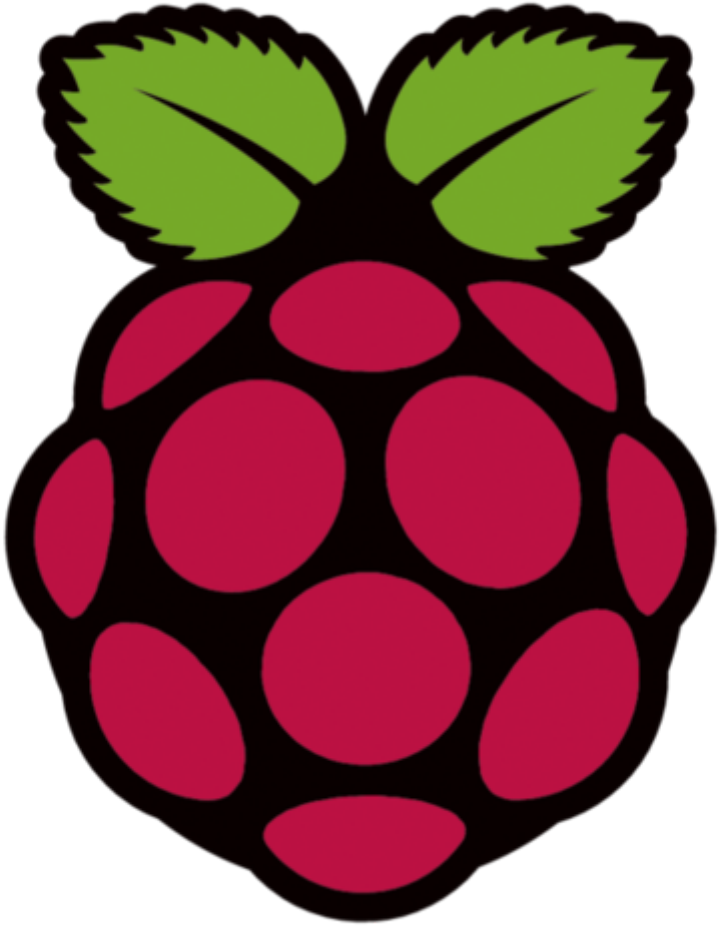
練習

- 步驟：

  1. 使用小畫家新增手寫資料(從0-9)，檔案名稱為*.JPG
  2. 檔案放到data目錄
  3. 測試knn_hand_written.py看分類結果

  (需要先執行過knn_ocr_sample.py)

Raspberry Pi Rocks the World



Thanks