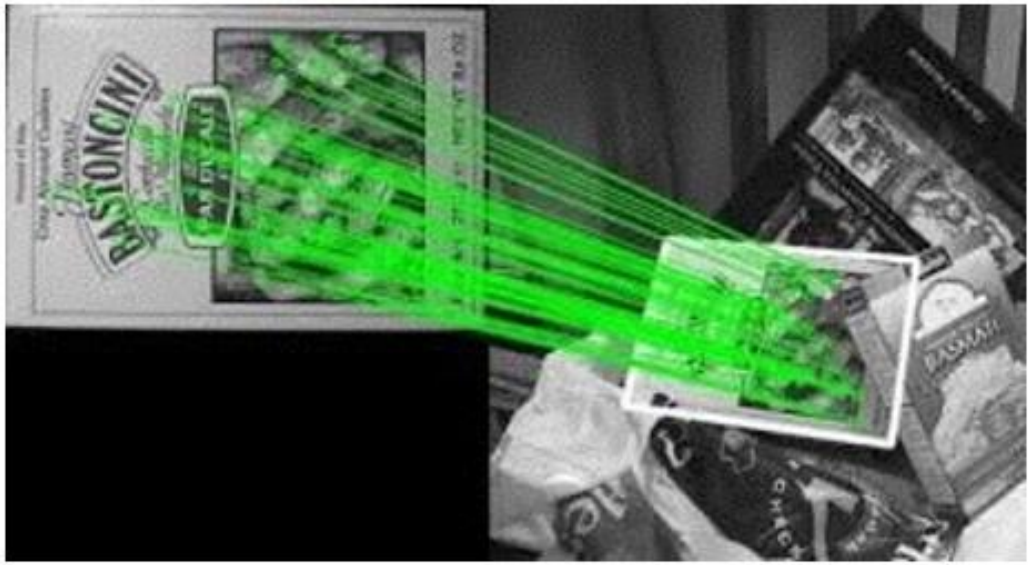
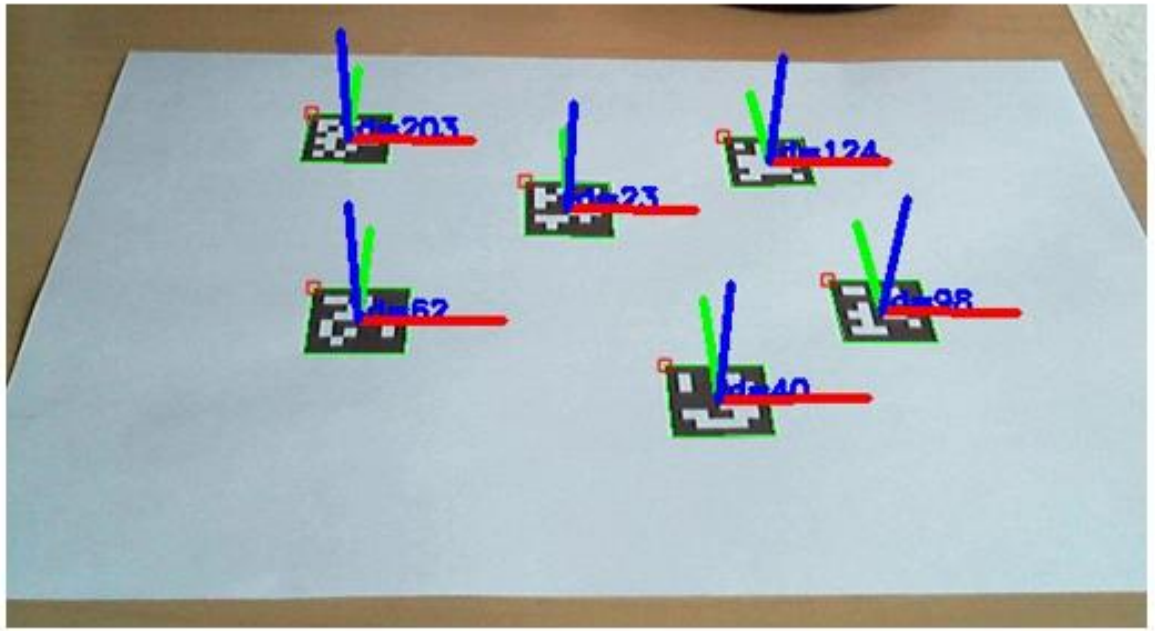
OPENCV BASICS

- core. The Core Functionality
- imgproc. Image Processing
- imgcodecs. Image file reading and writing
- videoio. Media I/O
- highgui. High-level GUI and Media I/O
- video. Video Analysis
- calib3d. Camera Calibration and 3D Reconstruction
- features2d. 2D Features Framework
- objdetect. Object Detection
- ml. Machine Learning
- flann. Clustering and Search in Multi-Dimensional Spaces
- photo. Computational Photography
- stitching. Images stitching
- cuda. CUDA-accelerated Computer Vision
- cudaarithm. CUDA-accelerated Operations on Matrices
- cudabgsegm. CUDA-accelerated Background Segmentation
- cudacodec. CUDA-accelerated Video Encoding/Decoding
- cudafeatures2d. CUDA-accelerated Feature Detection and Description
- cudafilters. CUDA-accelerated Image Filtering
- cudaimgproc. CUDA-accelerated Image Processing
- cudaoptflow. CUDA-accelerated Optical Flow
- cudastereo. CUDA-accelerated Stereo Correspondence
- cudawarping. CUDA-accelerated Image Warping
- shape. Shape Distance and Matching
- superres. Super Resolution
- videostab. Video Stabilization
- viz. 3D Visualizer
- bioinspired. Biologically inspired vision models and derivated tools
- cvv. GUI for Interactive Visual Debugging of Computer Vision Programs
- datasets. Framework for working with different datasets
- face. Face Recognition
- Binary descriptors for lines extracted from an image
- optflow. Optical Flow Algorithms
- reg. Image Registration
- rgbd. RGB-Depth Processing
- Saliency API
- surface_matching. Surface Matching

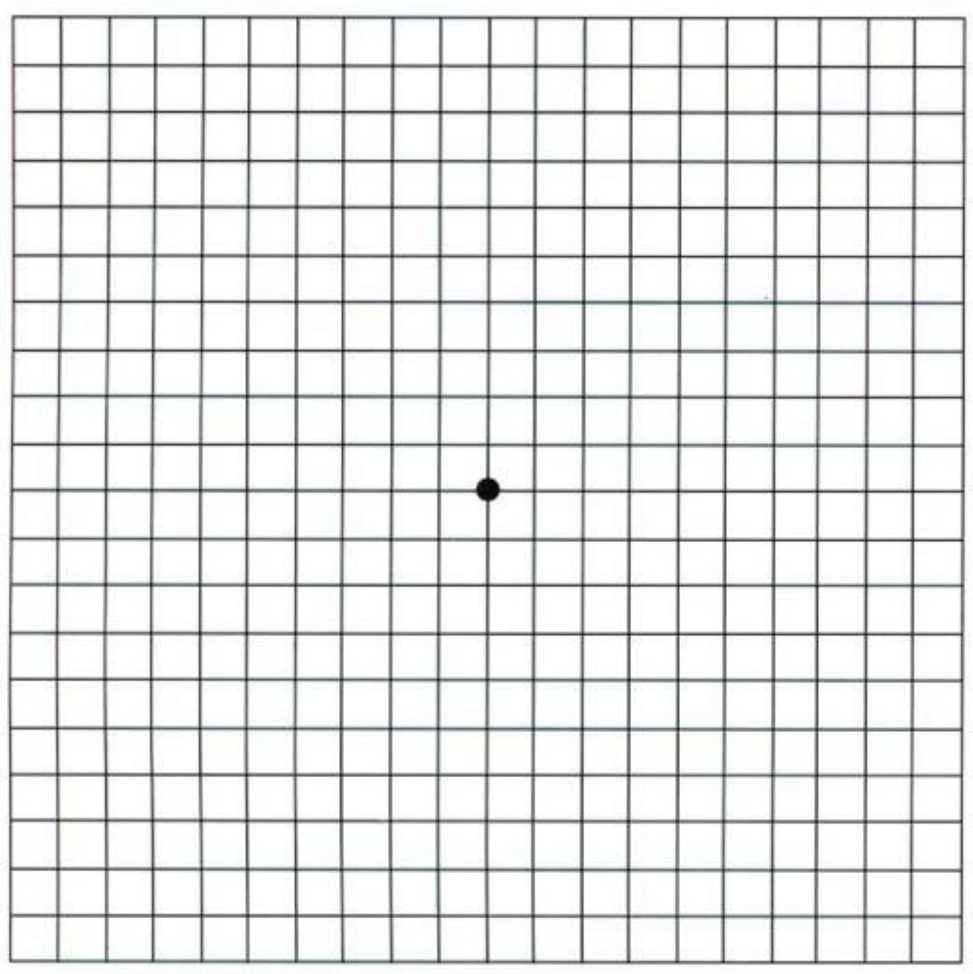
feature detection





pattern recognition

MAT



rows: 長
cols: 寬
type: 像素型態
channels: 通道數

NORMAL:

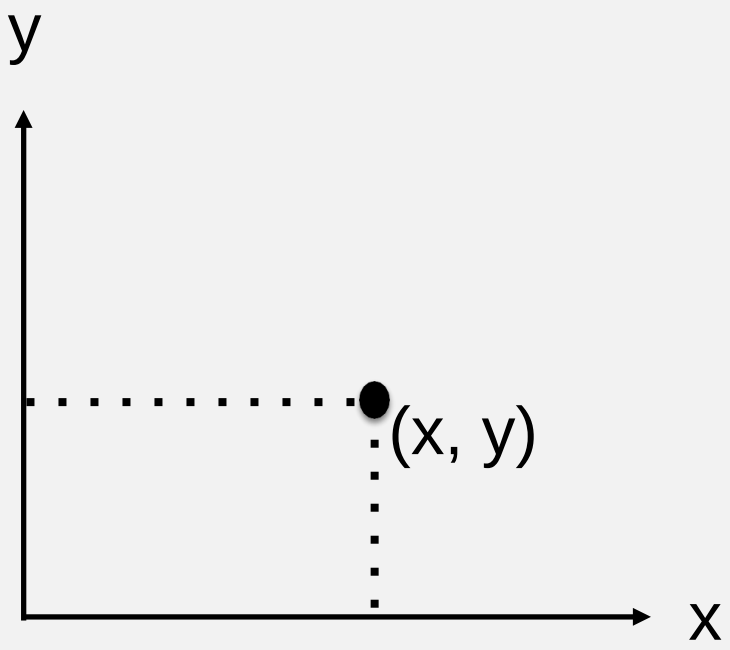


image:

```
0/0---X--->
 |
 |
 Y
 |
 |
 V
```

```
0/0---column--->
  |
  |
row
  |
  |
  V
```

MAT VALUE ACCESS

	Column 0	Column 1	Column ...	Column m
Row 0	0,0	0,1	...	0, m
Row 1	1,0	1,1	...	1, m
Row ...	...,0	...,1	...	..., m
Row n	n,0	n,1	n,...	n, m

3-channel : B, G, R

	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m

```python
import numpy as np
import cv2

#read
image = cv2.imread("image.jpg")
#show
cv2.imshow("My Image", image)

#按下按鍵關閉顯示視窗
cv2.waitKey(0)
cv2.destroyAllWindows()

#save
cv2.imwrite("output.jpg", image)
```

標頭引入

```
import numpy as np
import cv2
```

讀寫圖片

讀取:

```
img = cv2.imread('image.jpg')
```

儲存:

```
cv2.imwrite('output.jpg', img)
```

顯示圖片

顯示影像:

等待按鍵輸入:

```
# 顯示圖片
cv2.imshow('My Image', img)

# 按下任意鍵則關閉所有視窗
cv2.waitKey(0)
cv2.destroyAllWindows()
```

開一個指定大小的黑圖
複製圖片

```
blank_image = np.zeros((height,width,3), np.uint8)
```

```
newImage = myImage.copy()
```

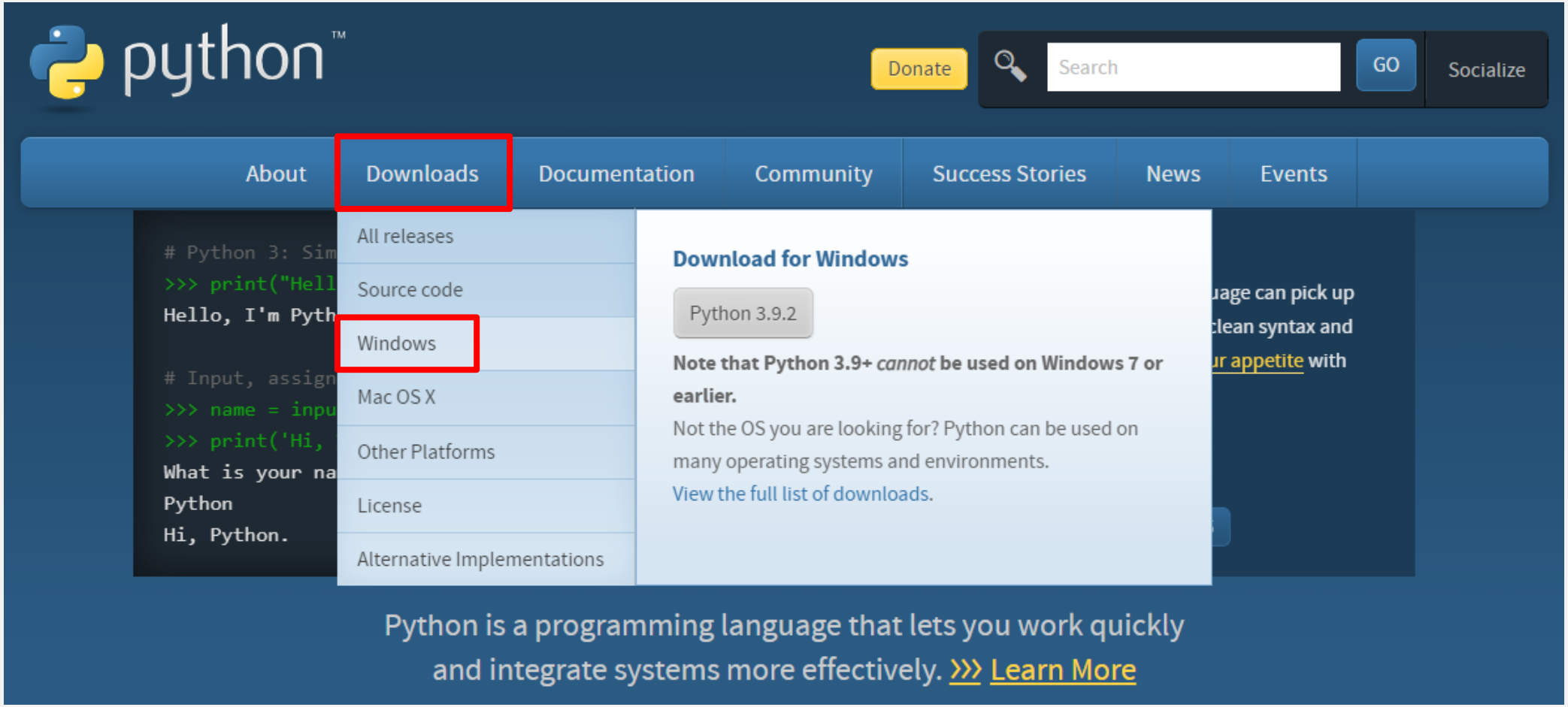
操作像素

image[row, col, channel]

WINDOWS10
PYTHON 3 & OPENCV

1. 下載PYTHON

到官網點選Downloads

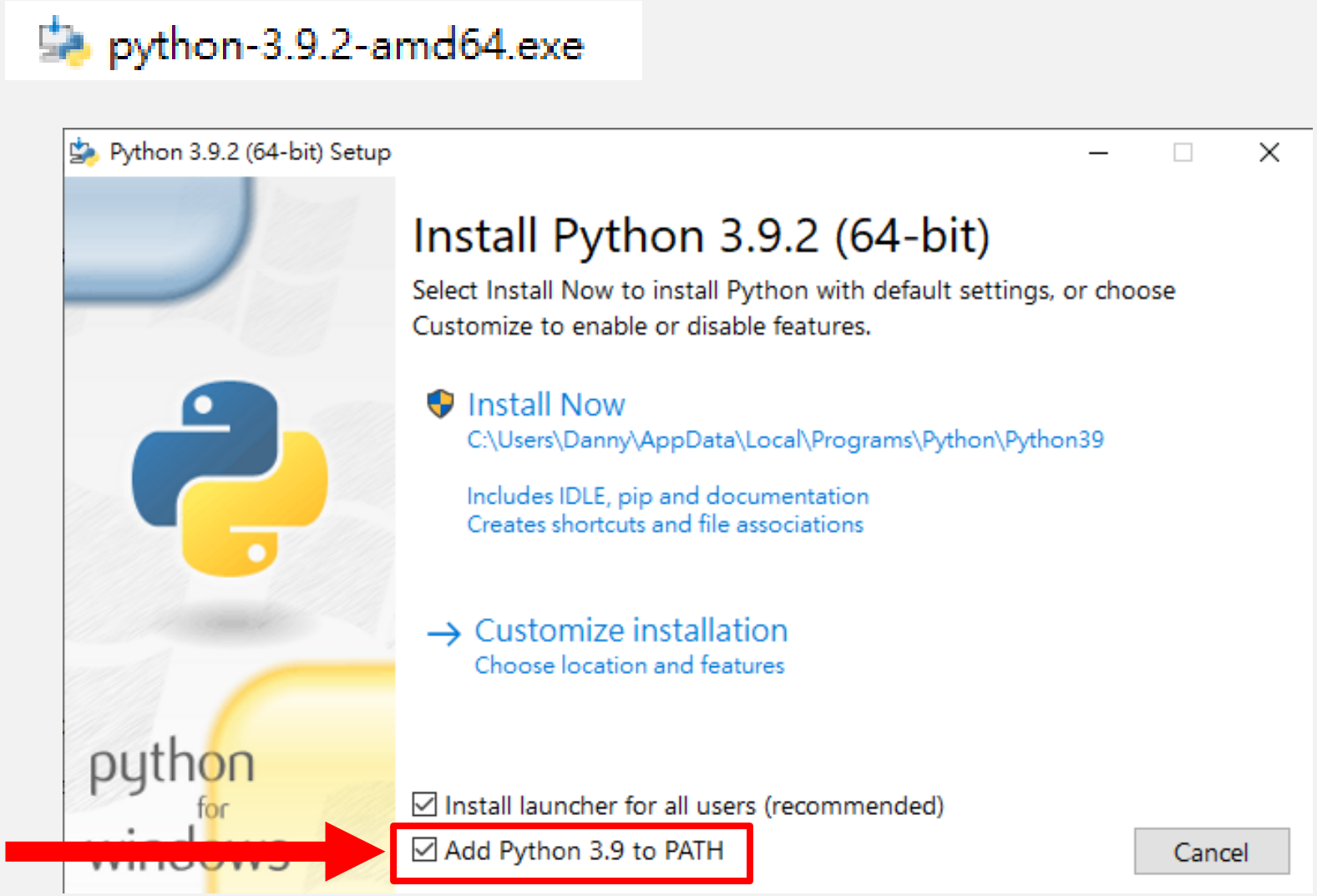


1. 下載PYTHON

選擇python 3

2. 安裝軟體

- 點選並安裝
- 加入環境變數



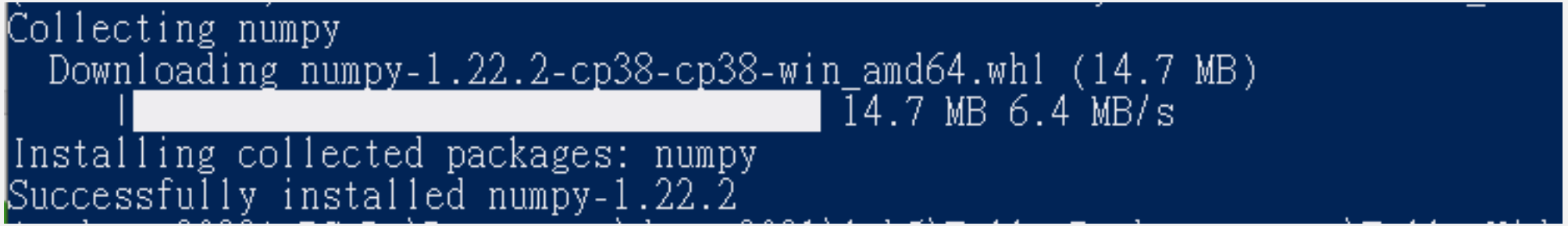
3. 安装OPENCV

- pip install opencv-python**==4.4.0.46**
- Test :

```python
import cv2

img = cv2.imread('kobe.jpg')

cv2.imshow('My Image', img)
cv2.waitKey(0)
cv2.destroyAllWindows()

```

4. 安装NUMPY

- pip install numpy



HOMEWORK 1

將圖片分成九宮格

對每一格進行指定的操作

原圖

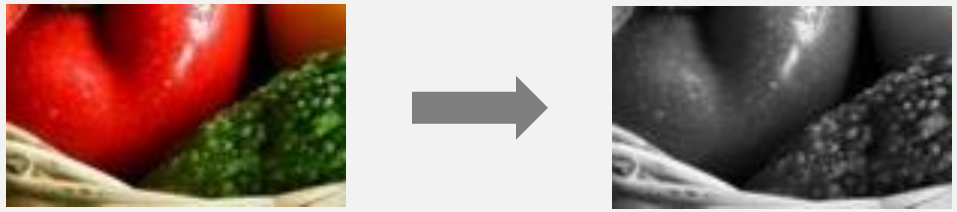
結果示意圖

EXCHANGE POSITION

- 交換指定兩格的內容 (20%)

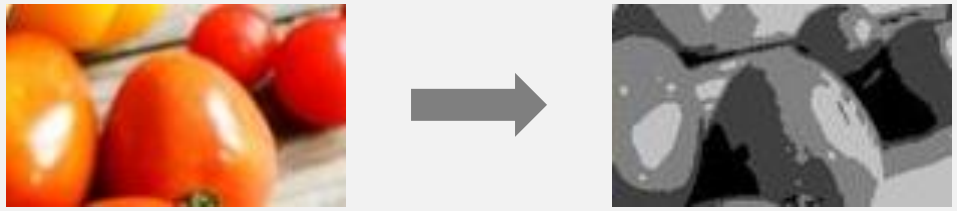
GRAY SCALE

- 將此格轉成灰階影像 (10%)
  - Hint: (R + G + B) / 3, 如果影像有3個channel, 則3個channel設一樣的值



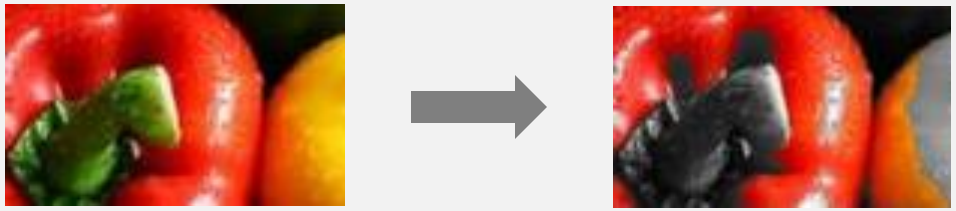
INTENSITY RESOLUTION

- 將此格轉為灰階影像，再把灰階的Intensity resolution降為4 (256→4) (10%)

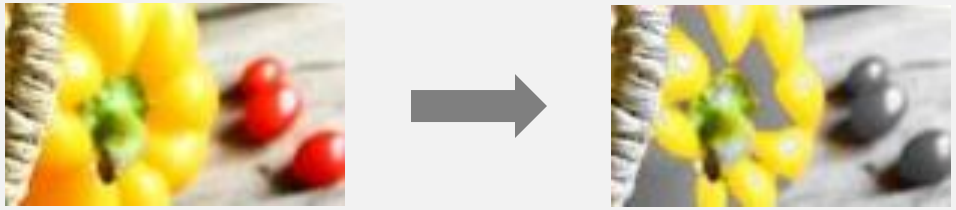


COLOR FILTER

- 紅色濾鏡: 只保留此格圖片的紅色區塊，其餘轉為灰階 (10%)
  - Hint: R > 150 and R * 0.6 > B and R * 0.6 > G

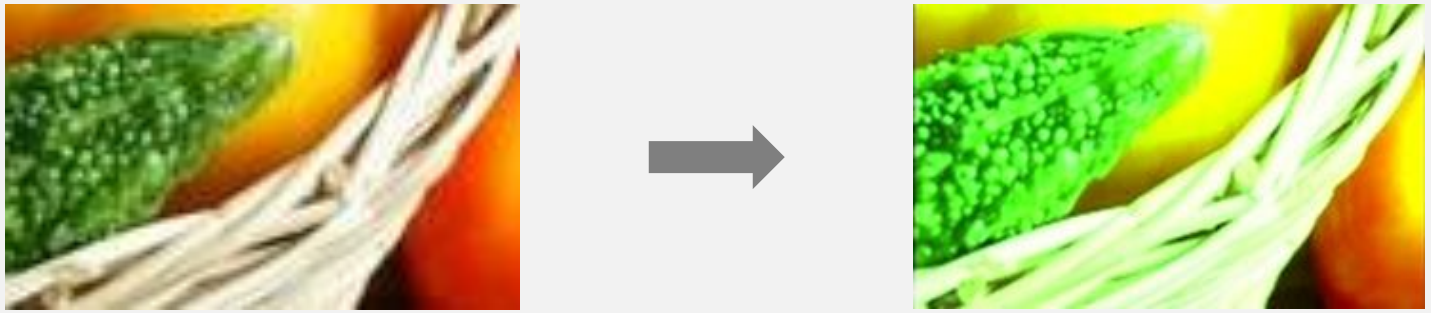


- 黃色濾鏡: 只保留此格圖片的黃色區塊，其餘轉為灰階 (10%)
  - Hint: (G + R) * 0.3 > B and abs(G - R) < 50



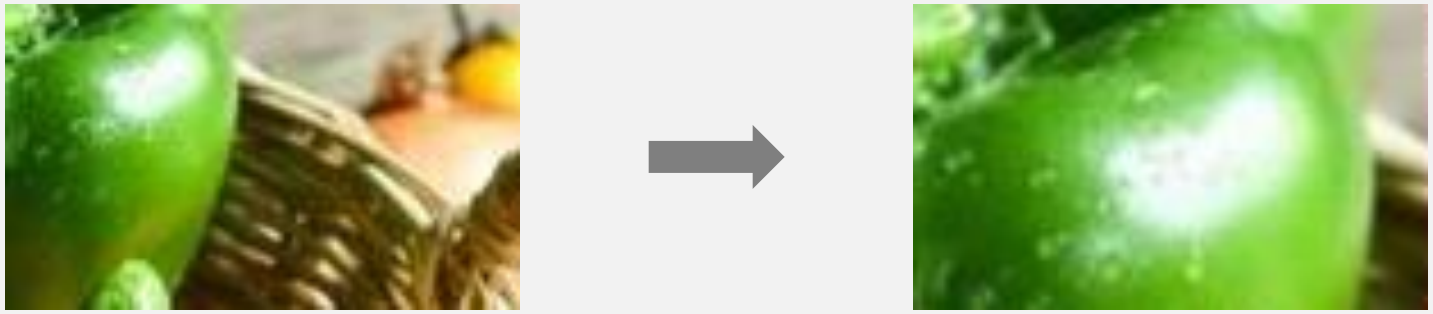
CHANNEL OPERATION

- 將此格的綠色值放大2倍 (10%)



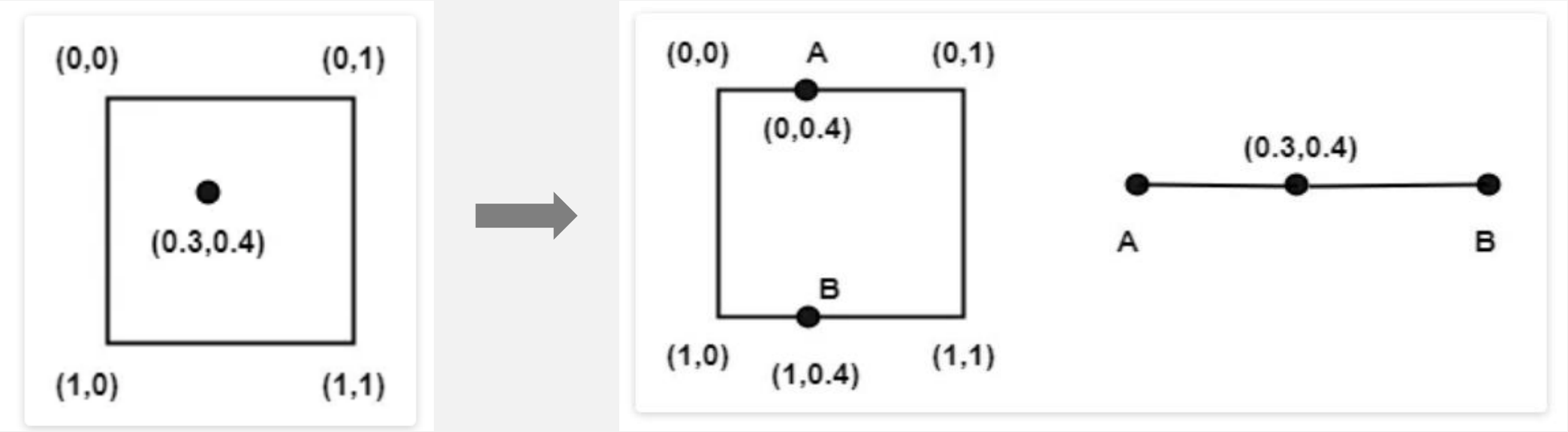
BILINEAR INTERPOLATION

- 將此格用 Bilinear interpolation 放大2倍，只需保留放大後左上角和原格子大小相同的部分 (10%)



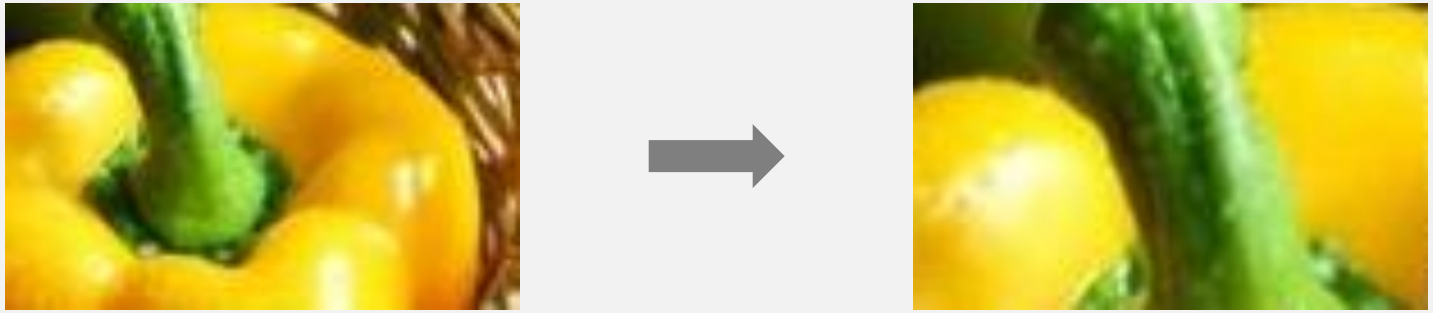
BILINEAR INTERPOLATION

- 根據輸出影像的像素位置，找到輸入影像中最鄰近的四個點，再利用雙線性內插法求出輸出影像的像素強度。



BICUBIC INTERPOLATION

- 將此格用 Bicubic interpolation 放大2倍，只需保留放大後左上角和原格子大小相同的部分(10%)



BICUBIC INTERPOLATION

- 三次樣條插值 (Cubic spline interpolation)
- 如果一個函數 f(x) 在 x=0 和 x=1 的位置上的函數值已知，利用一個三次方的多項式去計算這條曲線在 [0,1] 上的值

$$f(x) = ax^3 + bx^2 + cx + d$$

$$f'(x) = 3ax^2 + 2bx + c$$

$$f(0) = d$$

$$f(1) = a + b + c + d$$

$$f'(0) = c$$

$$f'(1) = 3a + 2b + c$$

$$a = 2f(0) - 2f(1) + f'(0) + f'(1)$$

$$b = -3f(0) + 3f(1) - 2f'(0) - f'(1)$$

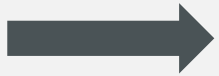
$$c = f'(0)$$

$$d = f(0)$$

BICUBIC INTERPOLATION

- 假設我們有四個值，分別 p0, p1, p2, p3 分別代表 x=-1, x=0, x=1 和 x=2 位置的函數值, 此時並不知道他的導數，因此用這條直線的斜率來近似替代這個位置導數

$$
\begin{aligned}
f(0) &= p_1 \\
f(1) &= p_2 \\
f'(0) &= \frac{p_2 - p_0}{2} \\
f'(1) &= \frac{p_3 - p_1}{2}
\end{aligned}
$$

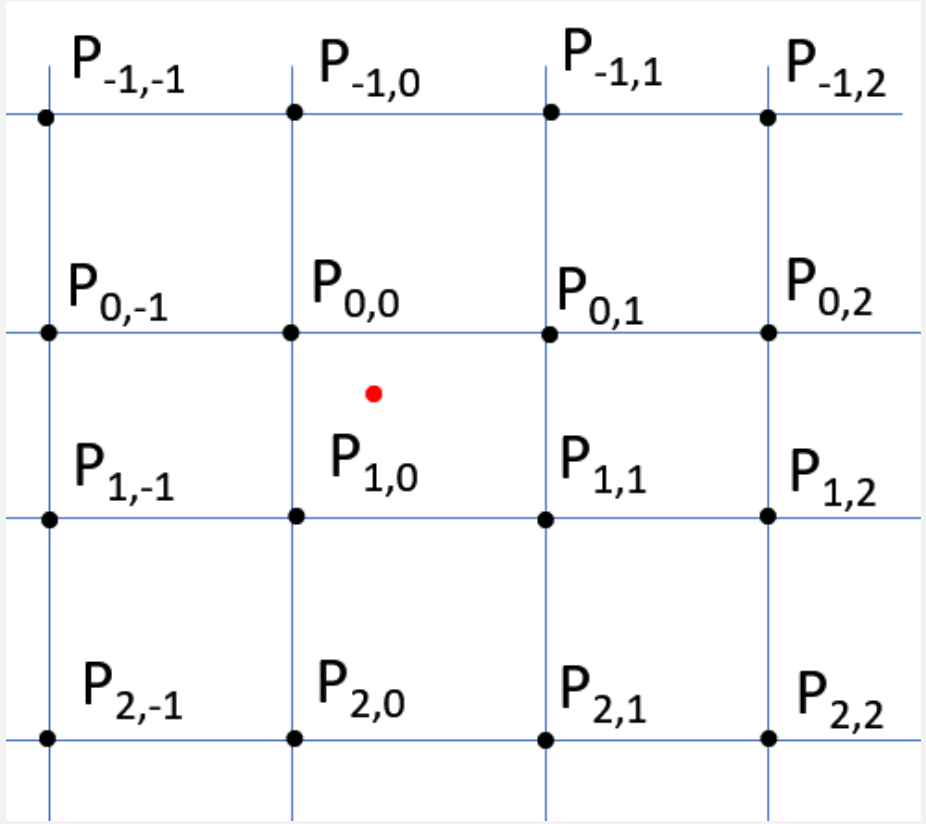


$$
\begin{aligned}
a &= -\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3 \\
b &= p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3 \\
c &= -\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2 \\
d &= p_1
\end{aligned}
$$

$$
f(p_0, p_1, p_2, p_3, x) = (-\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3)x^3 + (p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3)x^2 + (-\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2)x + p_1
$$

BICUBIC INTERPOLATION

- 雙三次插值法就是二維的三次樣條插值



GRADING POLICY

- Each grid – 10%
  - Call function: 6%
  - Implement by yourself: 10%
- Report – 10%

SUBMISSION

- Report
  - 包含 Method 、 Result 、 Feedback 三部分
  - 至多3頁
  - Filename: STUDENT_ID.pdf
- Code
  - Filename: STUDENT_ID.zip
- Deadline: 3/17 Fri. 10:10 a.m.