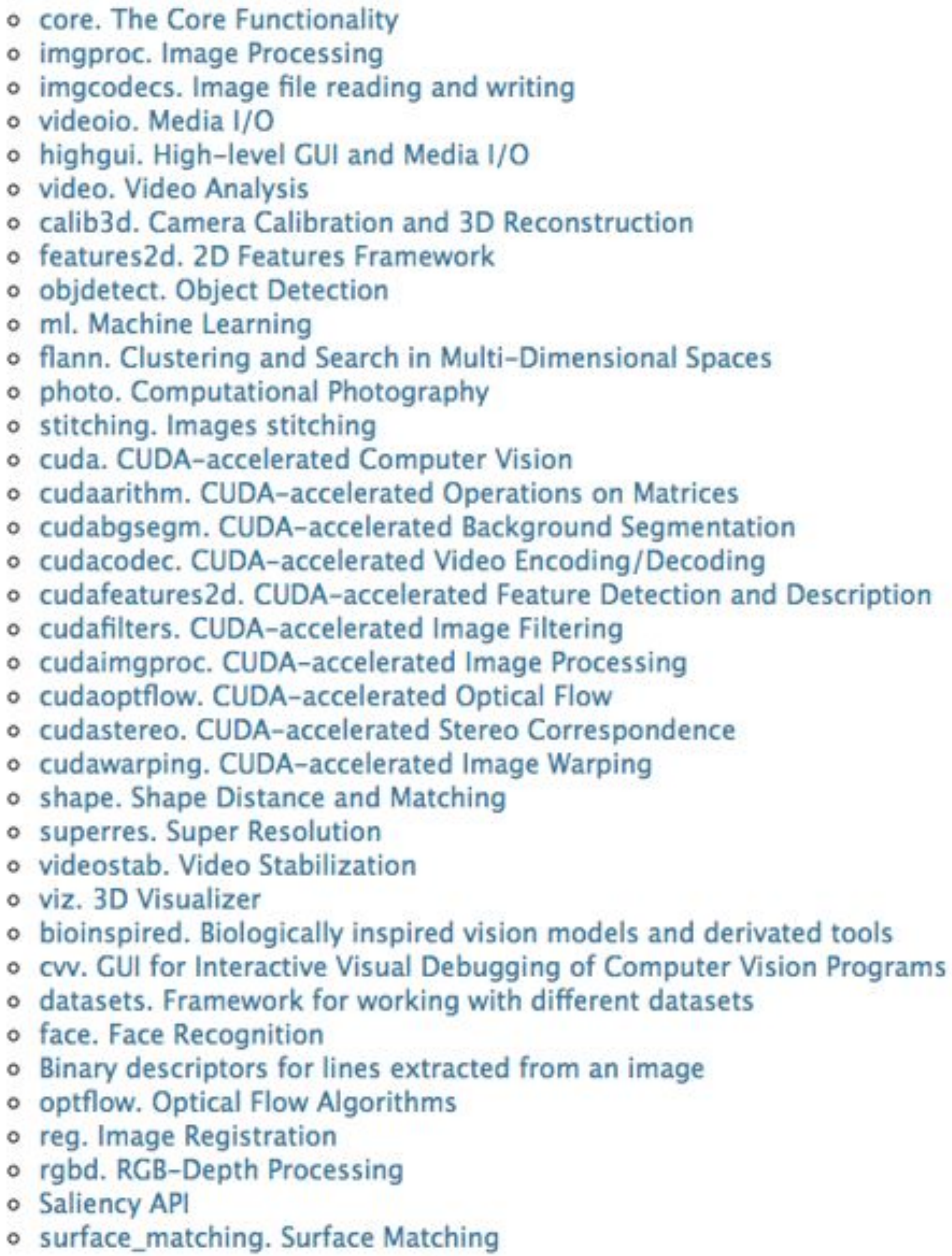
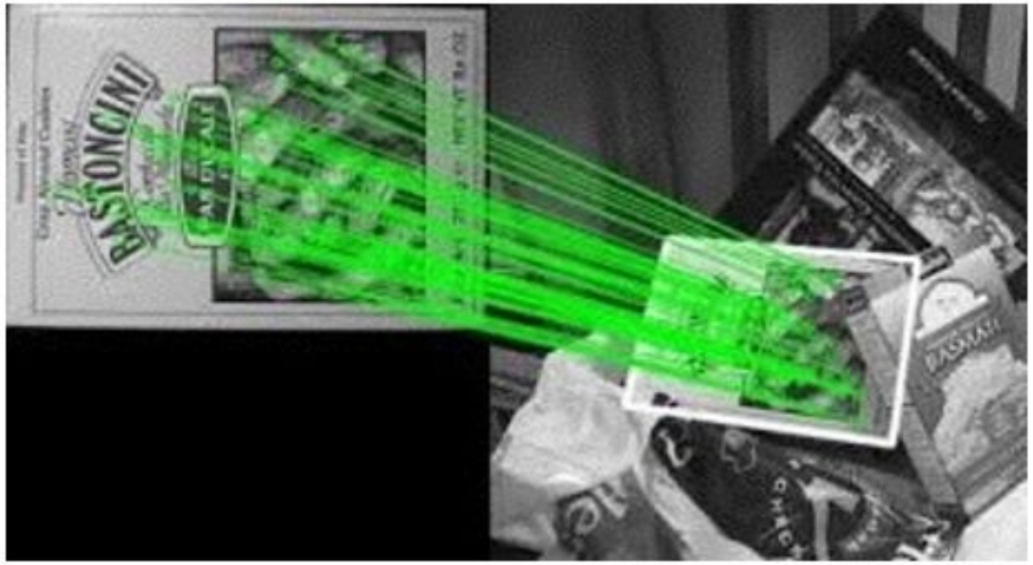
OPENCV BASICS

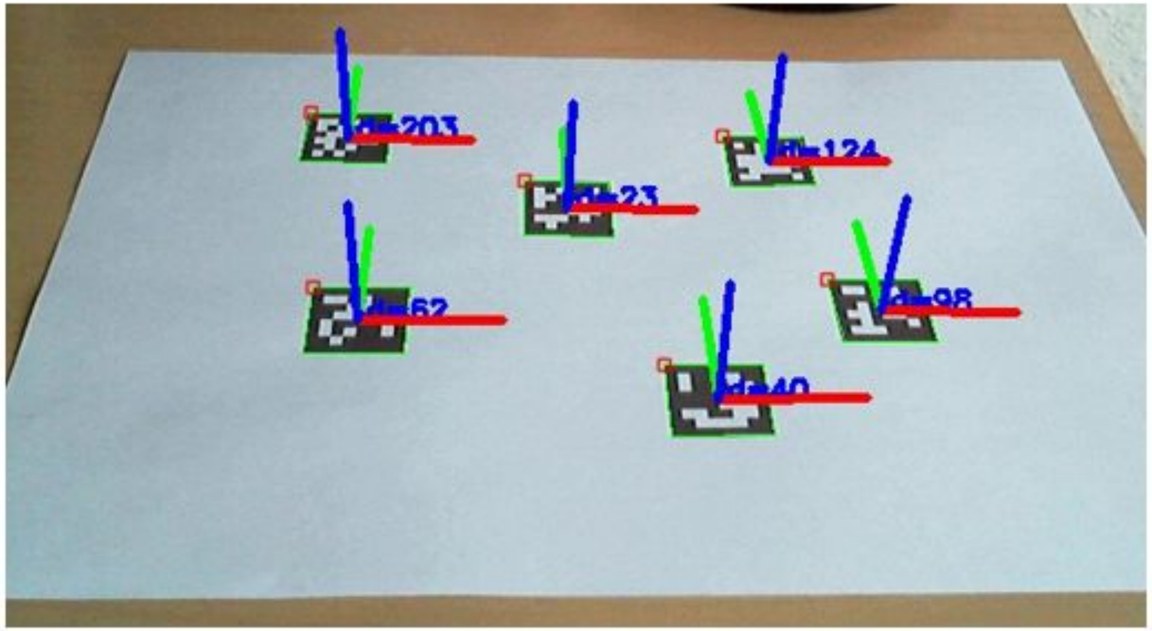
OpenCV



- 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

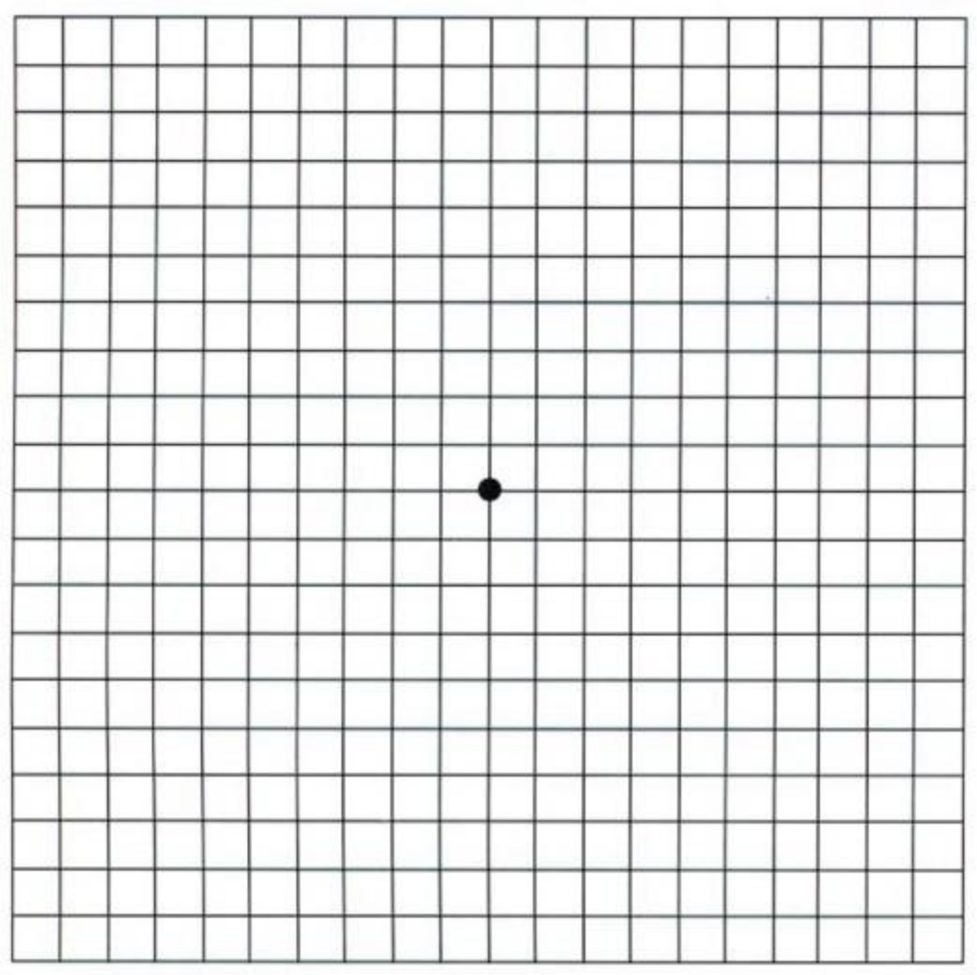
feature detection





pattern recognition

MAT



rows: 長

cols: 寬

type: 像素型態

channels: 通道數

NORMAL:

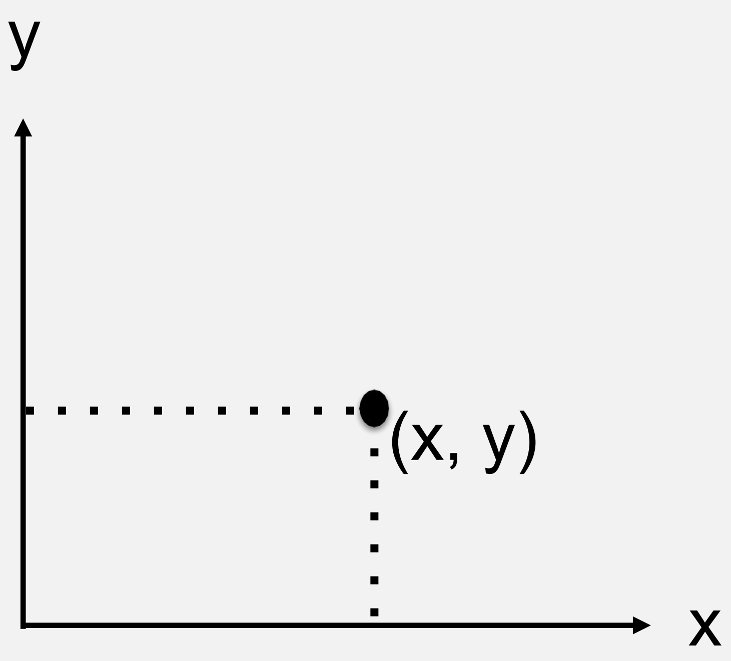
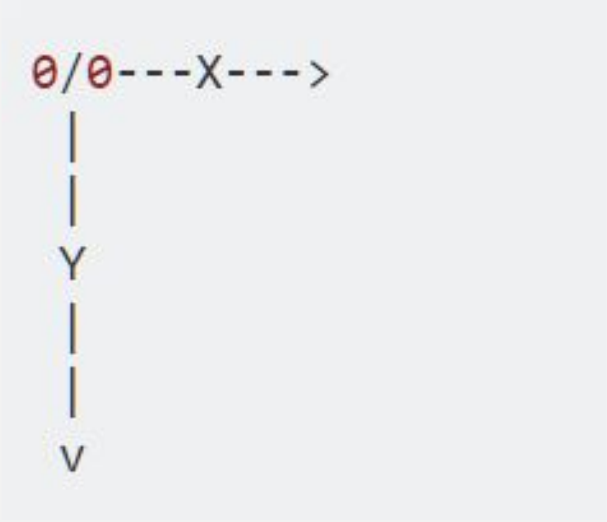
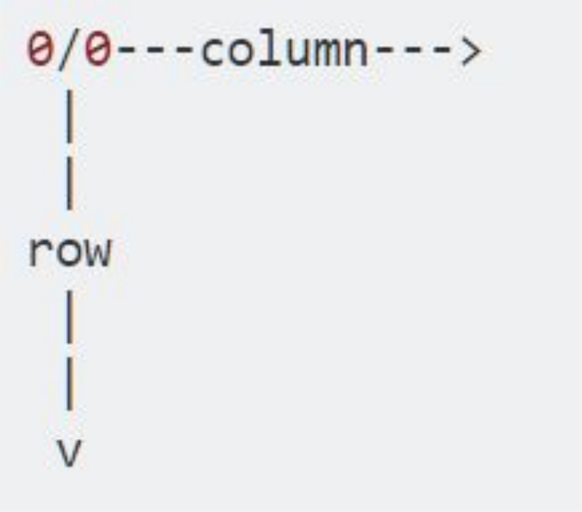


image:

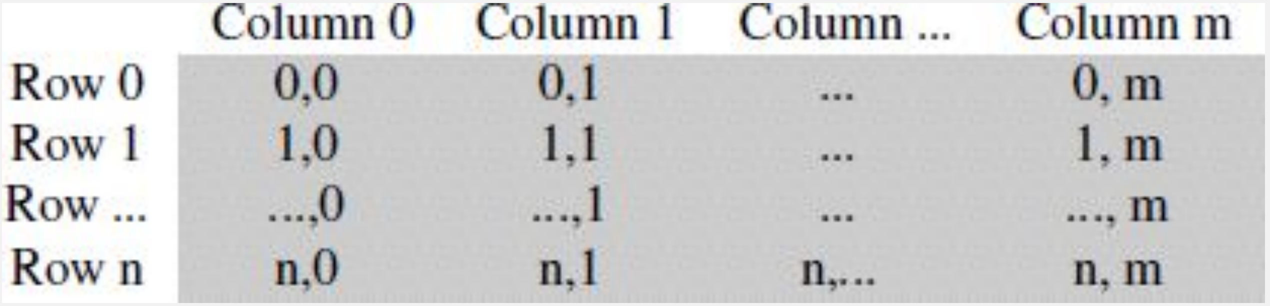


```
0/0---X--->
 |
 |
 Y
 |
 |
 v
```



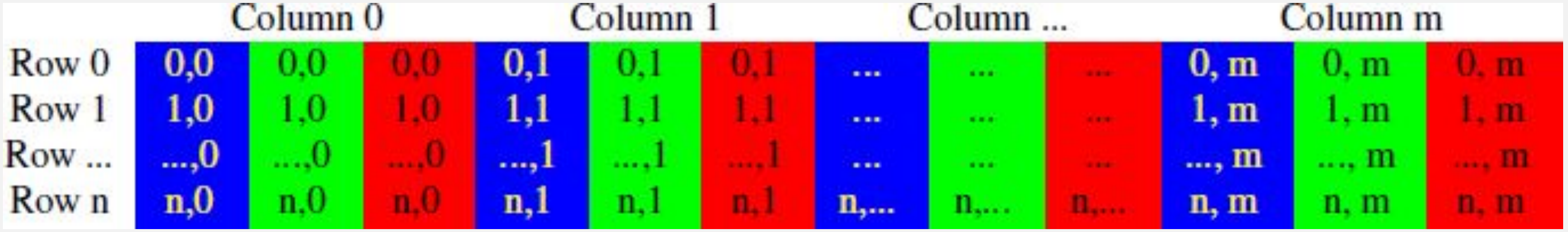
```
0/0---column--->
 |
 |
row
 |
 |
 v
```

MAT VALUE ACCESS

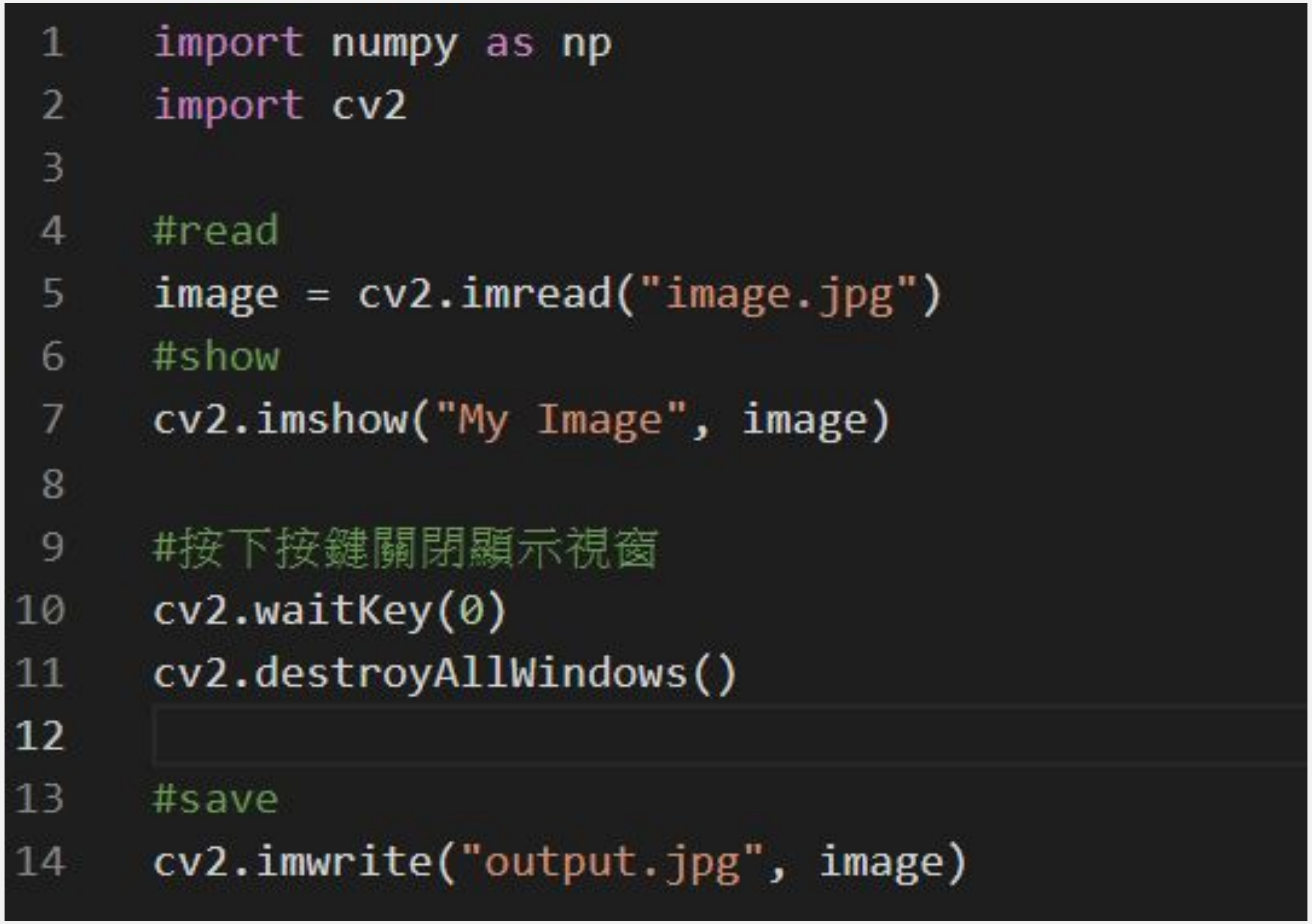


	Column 0	Column 1	Column ...	Column m
Row 0	0,0	0,1	...	0, m
Row 1	1,0	1,1	...	1, m
Row ...	...,0	...,1	...	..., m
Row n	n,0	n,1	n,...	n, m

3-channel : B, G, R

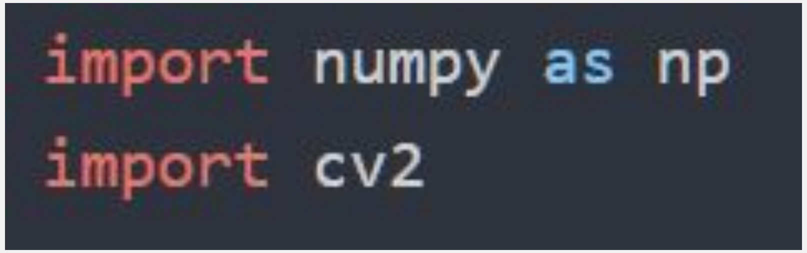


	Column 0			Column 1			Column ...			Column m		
Row 0	0,0	0,0	0,0	0,1	0,1	0,1	...	...	...	0, m	0, m	0, m
Row 1	1,0	1,0	1,0	1,1	1,1	1,1	...	...	...	1, m	1, m	1, m
Row ...	...,0	...,0	...,0	...,1	...,1	...,1	...	...	...	..., m	..., m	..., m
Row n	n,0	n,0	n,0	n,1	n,1	n,1	n,...	n,...	n,...	n, m	n, m	n, m



```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)
```

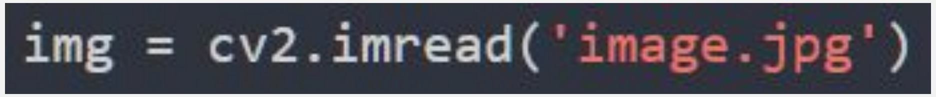
標頭引入



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

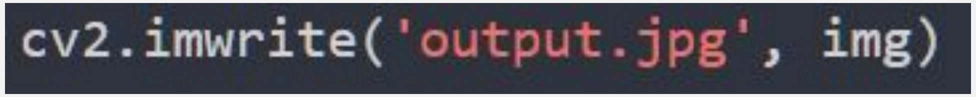
讀寫圖片

讀取:



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

儲存:

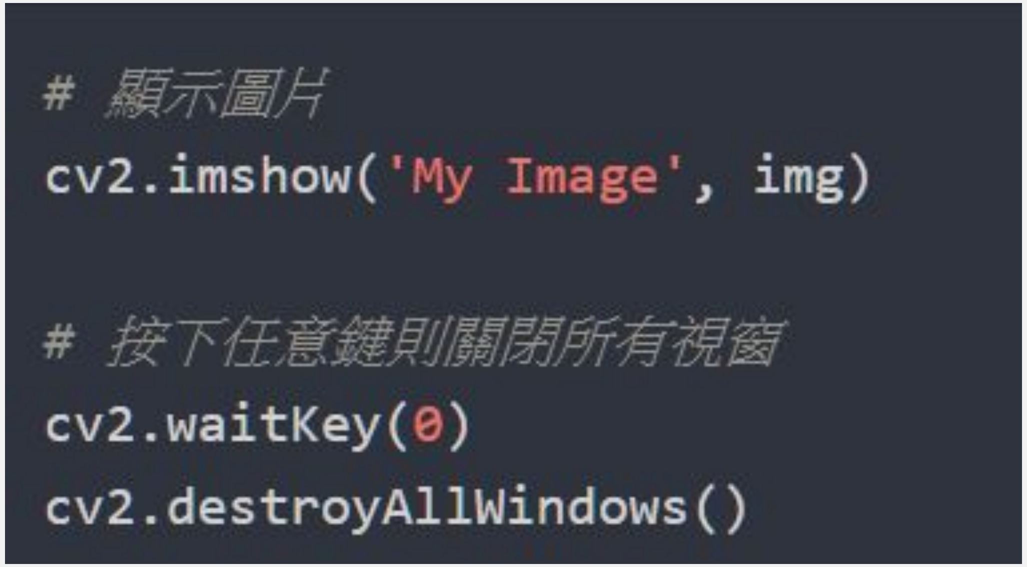


```
cv2.imwrite('output.jpg', img)
```

顯示圖片

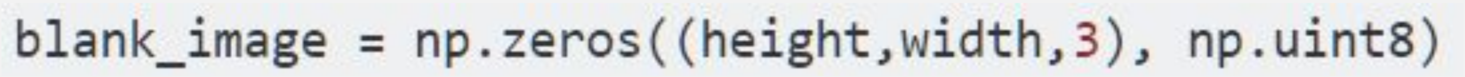
顯示影像：

等待按鍵輸入：



```python
# 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



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

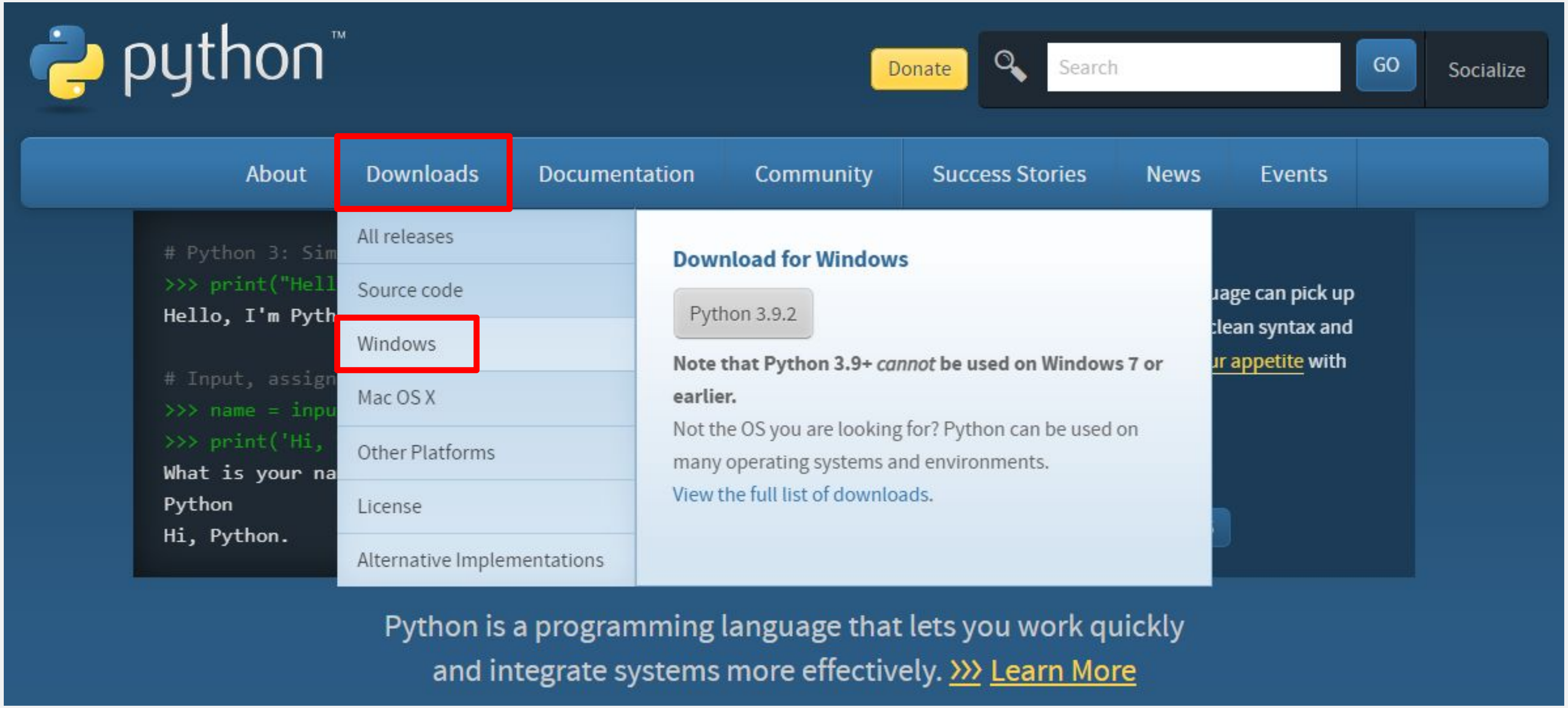
操作像素

image[row, col, channel]

WINDOWS10
PYTHON 3 & OPENCV

1. 下載PYTHON

到官網點選Downloads



```
# Python 3: Sim
>>> print("Hell
Hello, I'm Pyth

# Input, assign
>>> name = inpu
>>> print('Hi,
What is your na
Python
Hi, Python.
```

Download for Windows

Note that Python 3.9+ *cannot* be used on Windows 7 or earlier.

Not the OS you are looking for? Python can be used on many operating systems and environments.

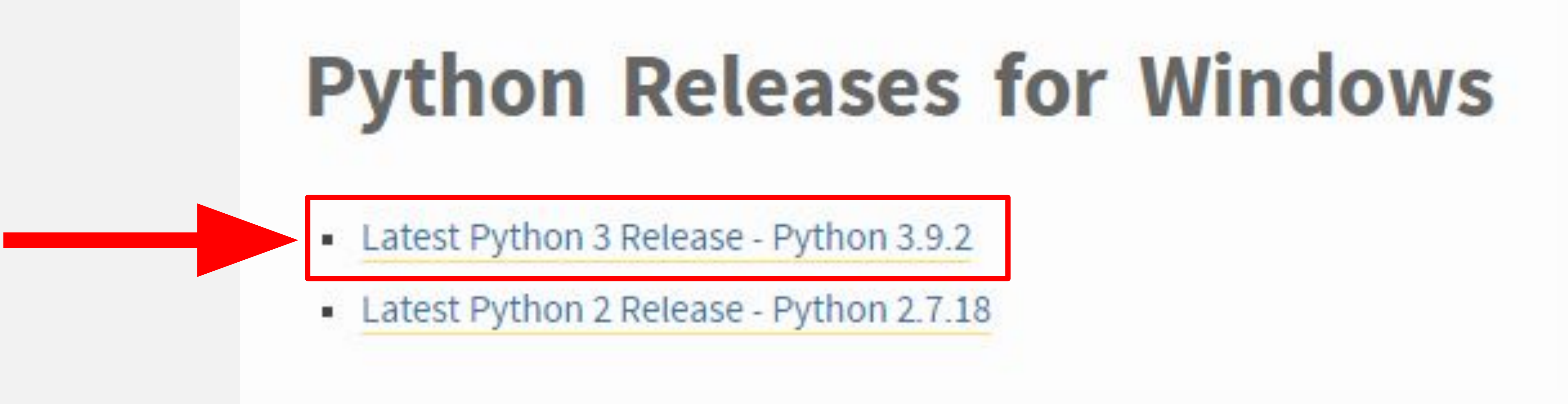
View the full list of downloads.

uage can pick up
clean syntax and
ur appetite with

Python is a programming language that lets you work quickly and integrate systems more effectively. >>> Learn More

1. 下載PYTHON

選擇python 3

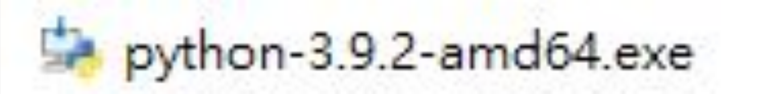


Python Releases for Windows

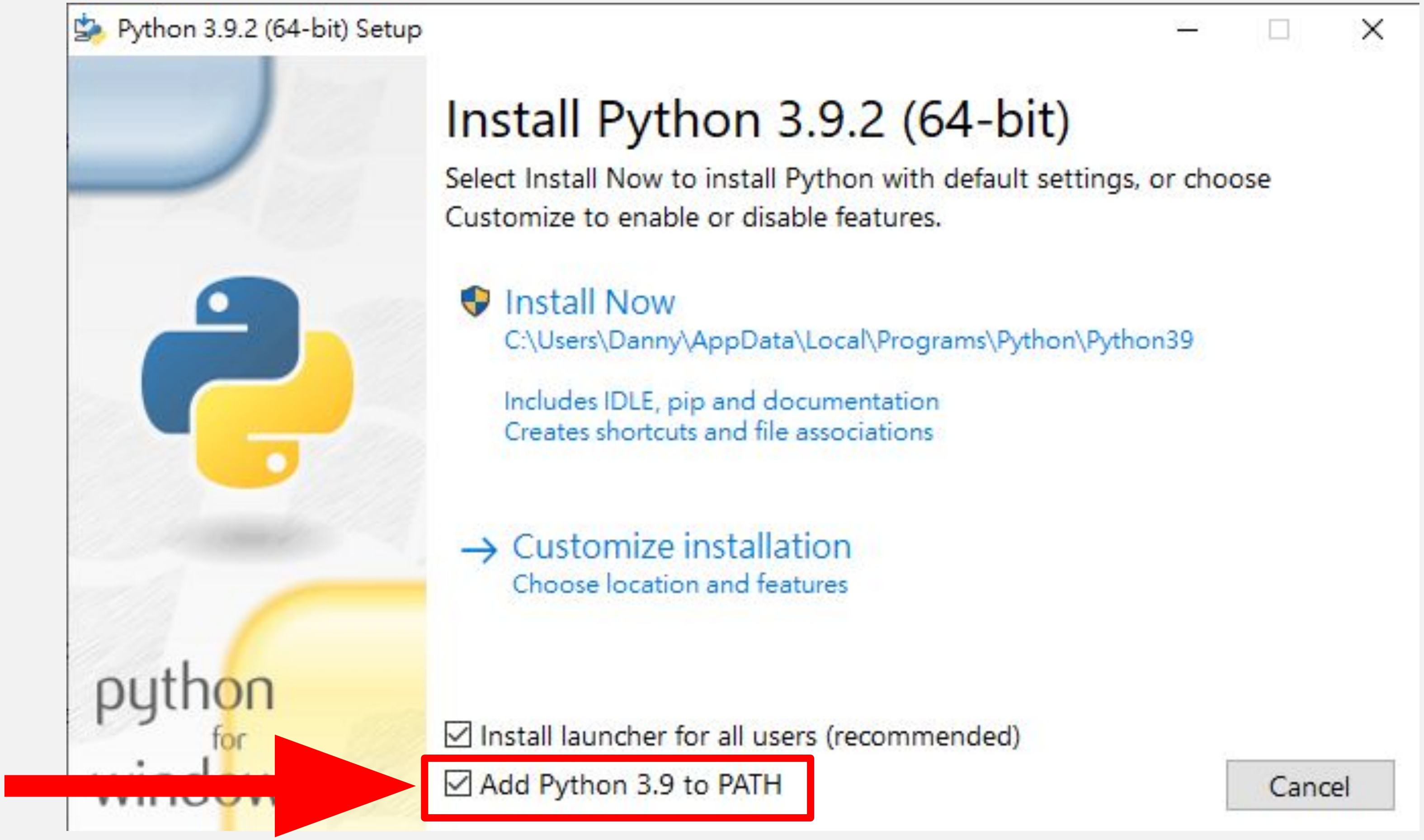
- Latest Python 3 Release - Python 3.9.2
- Latest Python 2 Release - Python 2.7.18

2. 安裝軟體

- 點選並安裝
- 加入環境變數



python-3.9.2-amd64.exe



Python 3.9.2 (64-bit) Setup

Install Python 3.9.2 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\Danny\AppData\Local\Programs\Python\Python39

Includes IDLE, pip and documentation
Creates shortcuts and file associations

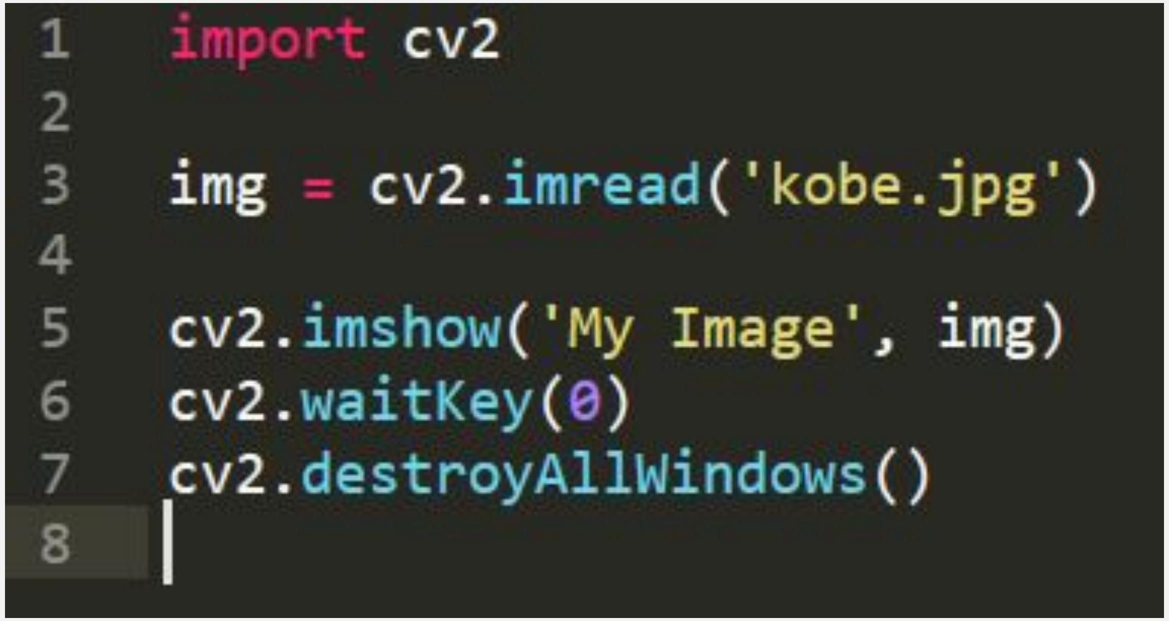
→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)

☑ Add Python 3.9 to PATH

3. 安裝OPENCV

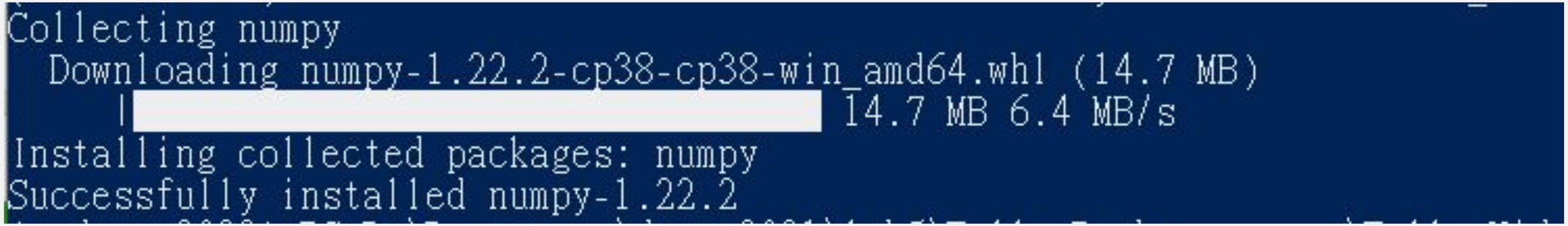
- pip install opencv-python**==4.4.0.46**
- Test :



```python
import cv2

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

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

```

4. 安裝NUMPY

- pip install numpy



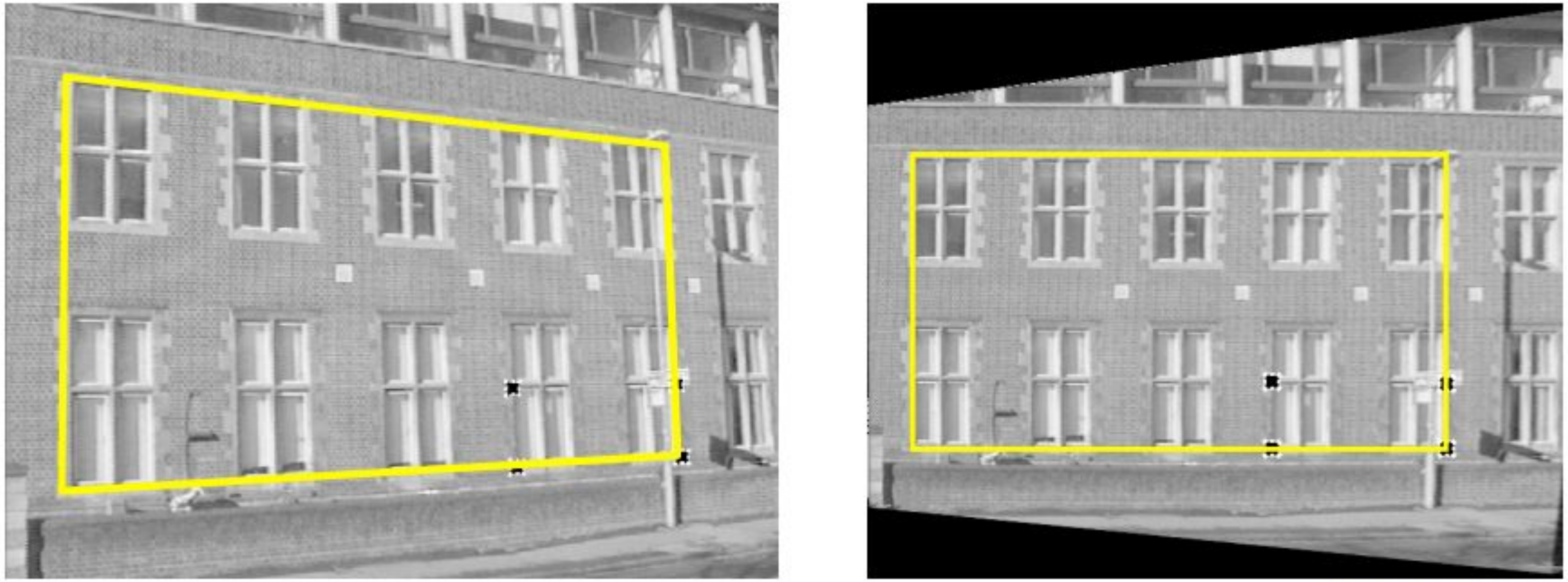
```
Collecting numpy
  Downloading numpy-1.22.2-cp38-cp38-win_amd64.whl (14.7 MB)
     |████████████████████████████████| 14.7 MB 6.4 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.22.2
```

HOMEWORK 1

CONTENT

- 使用不同的插值方法對影像中心順時鐘旋轉30度
- 使用不同的插值方法將影像warp至另一張影像
  - Nearest Neighbor Interpolation
  - Bilinear Interpolation
  - Bicubic Interpolation

IMAGE WARPING AND HOMOGRAPHY



from Hartley & Zisserman

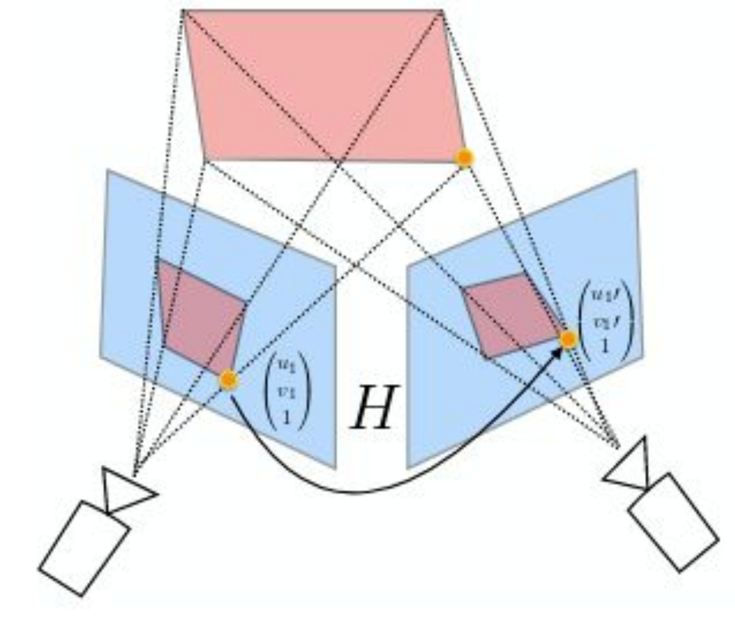

$\begin{pmatrix} u_1 \\ v_1 \\ 1 \end{pmatrix}$
H
$\begin{pmatrix} u_1{}' \\ v_1{}' \\ 1 \end{pmatrix}$

IMAGE WARPING AND HOMOGRAPHY

- Homogenous coordinates:
  - P’ = HP

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \underline{\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- We can get following equations:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

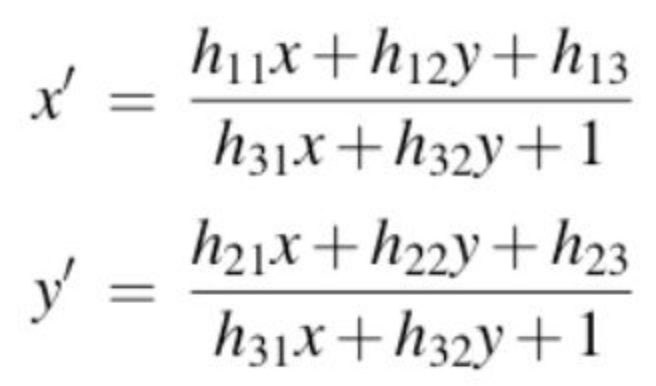
IMAGE WARPING AND HOMOGRAPHY

- We can restrict h33 to 1, then we need 4 pairs of points to solve the 8 variables

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \sim \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & \boxed{h_{33}} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

IMAGE WARPING AND HOMOGRAPHY

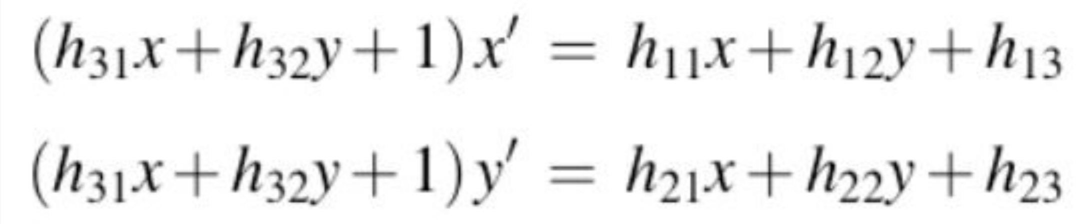
1. Setting h33 = 1



$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + 1}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + 1}$$

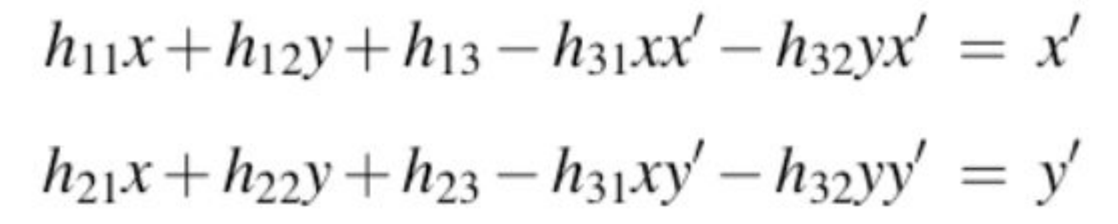
2. Multiplying through by denominator



$$(h_{31}x + h_{32}y + 1)x' = h_{11}x + h_{12}y + h_{13}$$

$$(h_{31}x + h_{32}y + 1)y' = h_{21}x + h_{22}y + h_{23}$$

3. Rearrange



$$h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' = x'$$

$$h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' = y'$$

$$\begin{aligned} h_{11}x + h_{12}y + h_{13} - h_{31}xx' - h_{32}yx' &= x' \\ h_{21}x + h_{22}y + h_{23} - h_{31}xy' - h_{32}yy' &= y' \end{aligned}$$

IMAGE WARPING AND HOMOGRAPHY

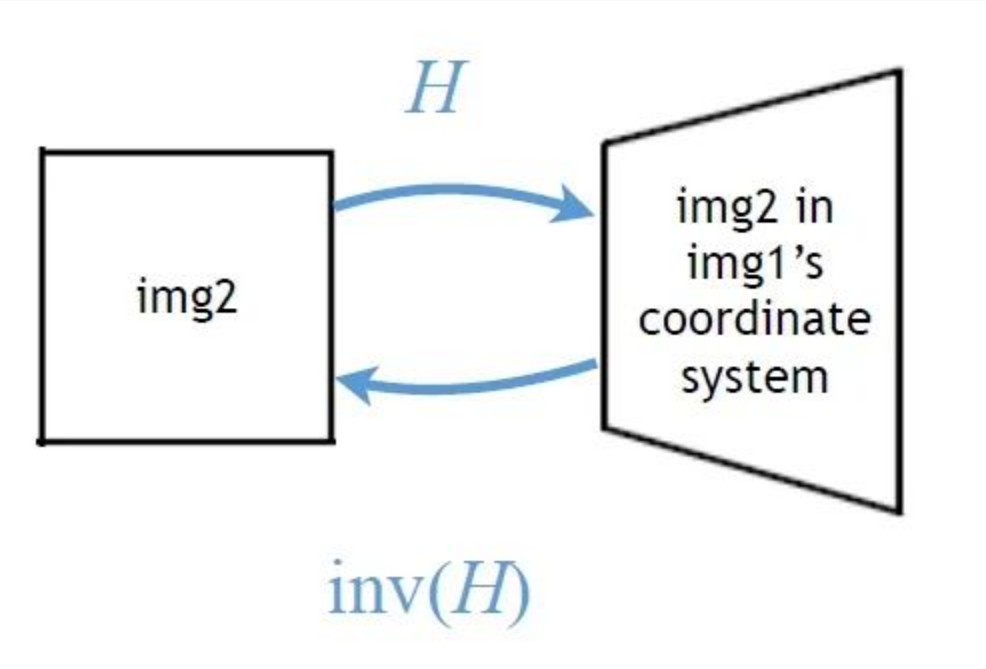
- Solve the Ah = b equation to get the homography matrix

$$\begin{array}{ccccc}
 & \textbf{2N x 8} & \textbf{8 x 1} & & \textbf{2N x 1} \\
\begin{array}{c} \textbf{Point 1} \\ \\ \\ \textbf{Point 2} \\ \\ \\ \textbf{Point 3} \\ \\ \\ \textbf{Point 4} \end{array} &
\begin{bmatrix}
x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 \\
0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 \\
x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x'_2 & -y_2x'_2 \\
0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y'_2 & -y_2y'_2 \\
x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x'_3 & -y_3x'_3 \\
0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y'_3 & -y_3y'_3 \\
x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x'_4 & -y_4x'_4 \\
0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y'_4 & -y_4y'_4
\end{bmatrix} &
\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \end{bmatrix} & = &
\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} \\
 & \mathbf{A} & \mathbf{h} & & \mathbf{b}
\end{array}$$

Source: https://www.cse.psu.edu/~rtc12/CSE486/lecture16.pdf

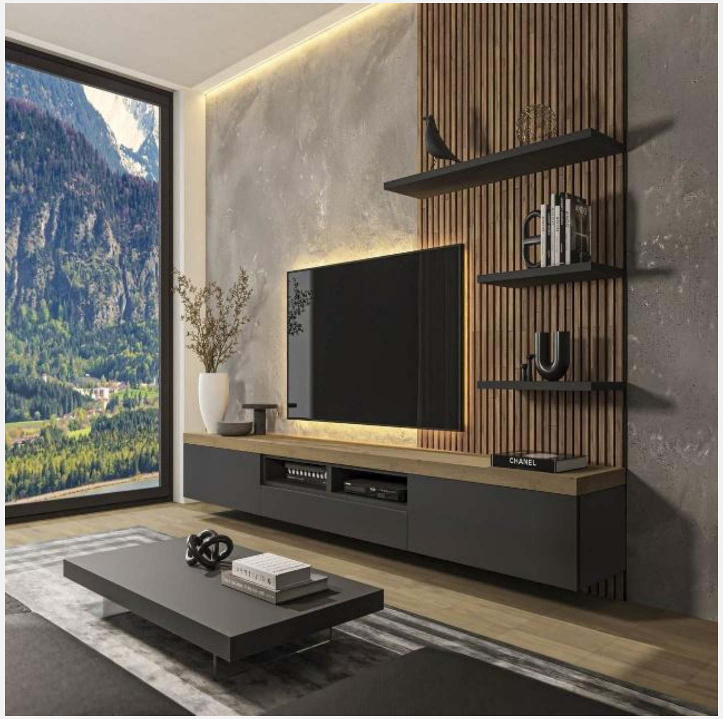
IMAGE WARPING AND HOMOGRAPHY

- Perform warping

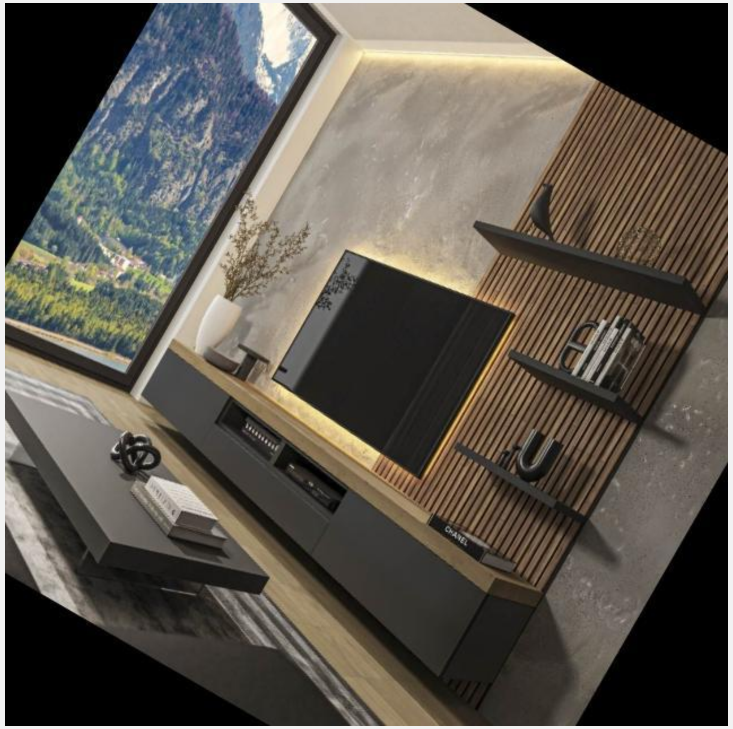


Source: https://yungyung7654321.medium.com/python實作自動全景圖拼接-automatic-panoramic-image-stitching-28629c912b5a

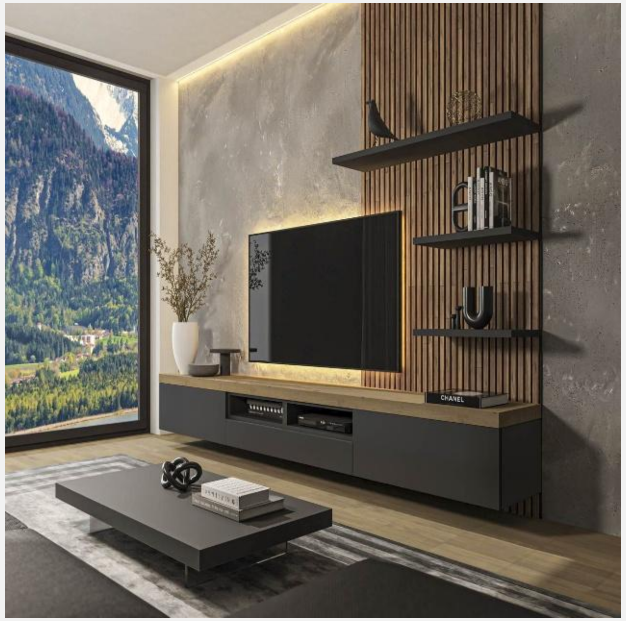
原圖

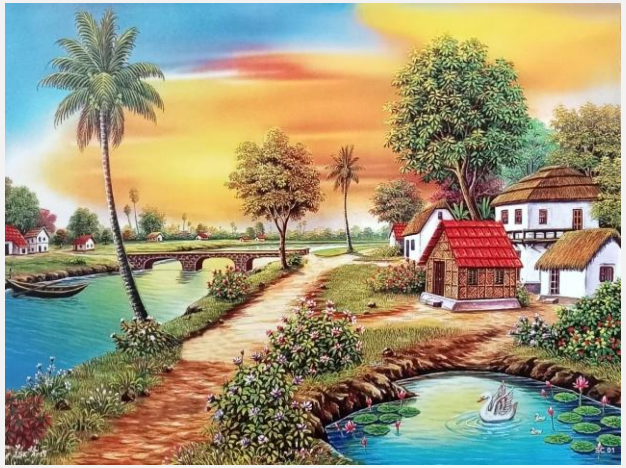


結果示意圖

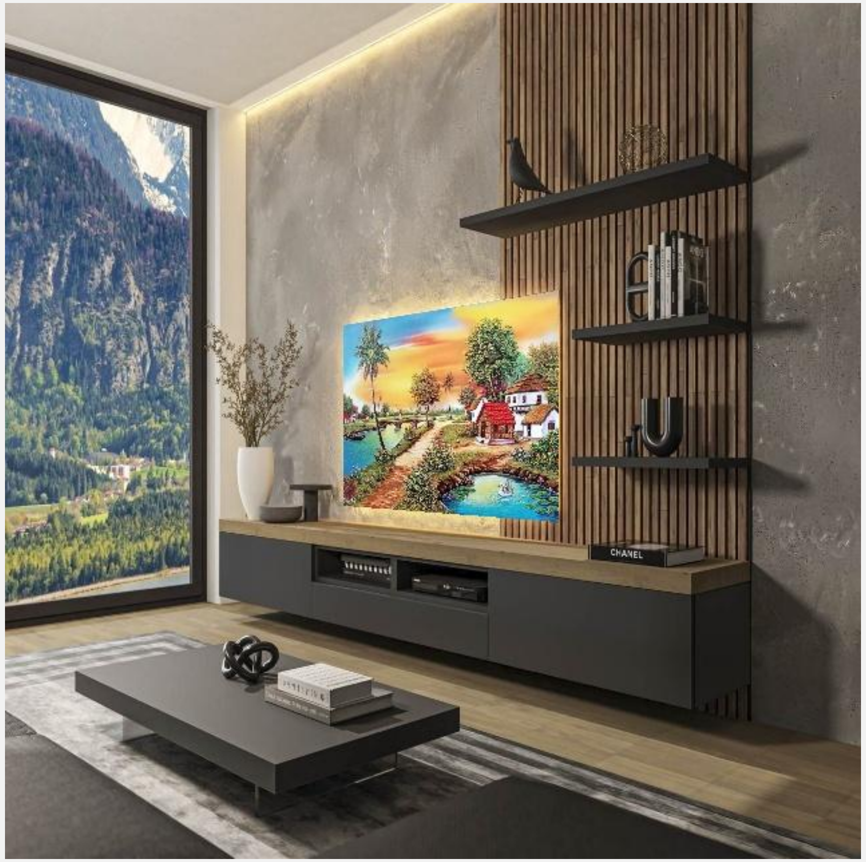


原圖





結果示意圖

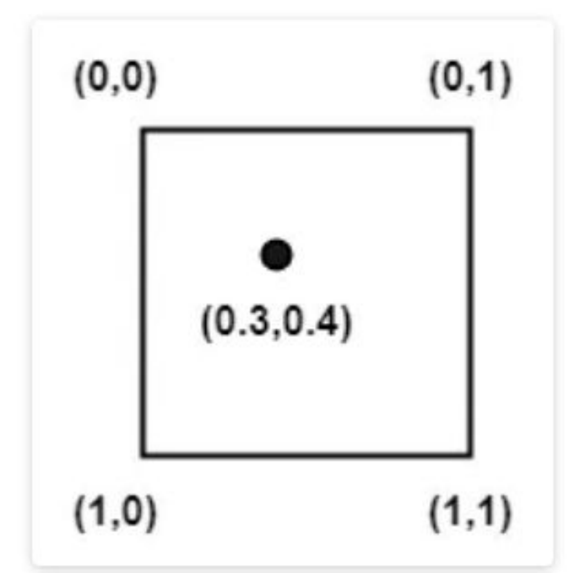


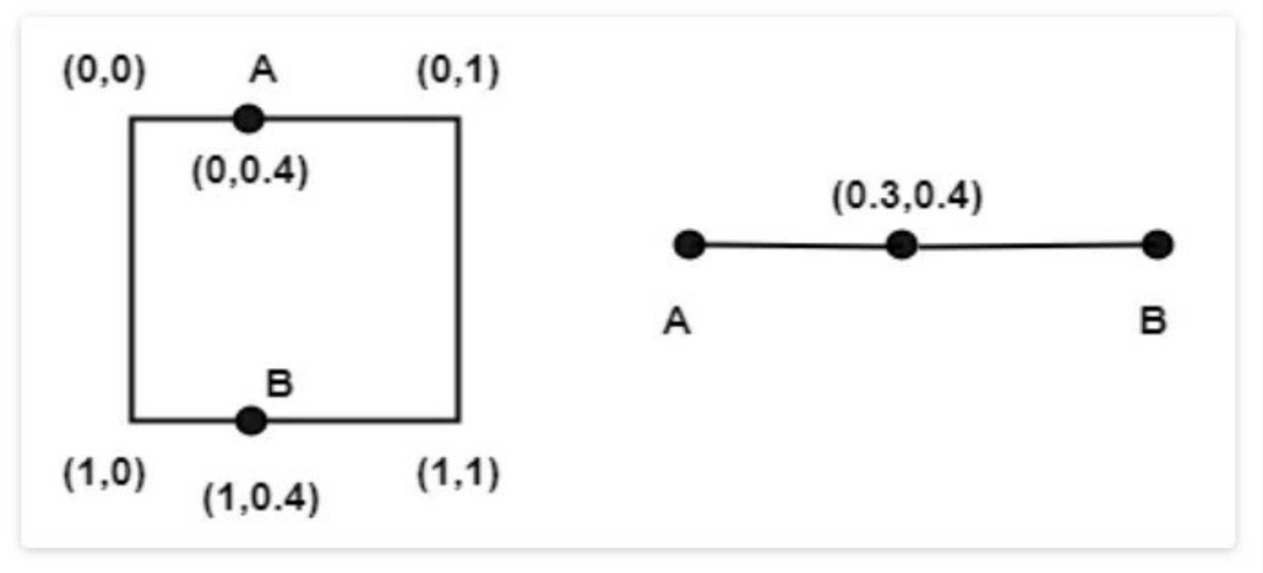
TV Coordinates

```
[(253, 241), (413, 215), (413, 387), (253, 375)]
```

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

- 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

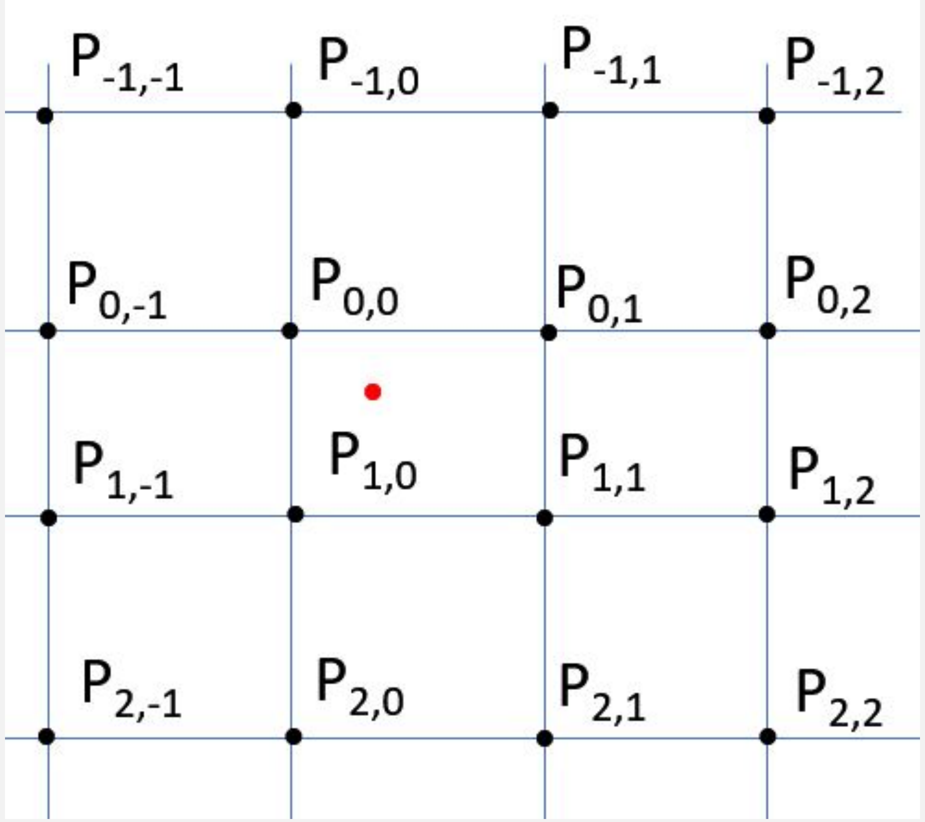
- 假設我們有四個值，分別 p0, p1, p2, p3 分別代表 x=-1, x=0, x=1 和 x=2 位置的函數值, 此時並不知道他的導數，因此用這條直線的斜率來近似替代這個位置導數

$$
\begin{aligned}
f(0) &= p_1 \\
f(1) &= p_2 \\
f'(0) &= \frac{p_2 - p_0}{2} \\
f'(1) &= \frac{p_3 - p_1}{2}
\end{aligned}
\quad\longrightarrow\quad
\begin{aligned}
a &= -\tfrac{1}{2}p_0 + \tfrac{3}{2}p_1 - \tfrac{3}{2}p_2 + \tfrac{1}{2}p_3 \\
b &= p_0 - \tfrac{5}{2}p_1 + 2p_2 - \tfrac{1}{2}p_3 \\
c &= -\tfrac{1}{2}p_0 + \tfrac{1}{2}p_2 \\
d &= p_1
\end{aligned}
$$

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

BICUBIC INTERPOLATION

- 雙三次插值法就是二維的三次樣條插值
- Hint: 記得注意overflow的問題
  - np.clip(img, 0, 255)
  - np.array(img, dtype=np.uint8)



GRADING POLICY

- Each method – 15%
  - 禁止使用 OpenCV (讀寫照片以外)的函式
- Report – 10%
  - 請比較不同方法結果的差異
- Format penalty: -10pts

SUBMISSION

- Report
  - 包含 Method 、Result 、Feedback 三部分
  - 至多3頁
  - Filename: STUDENT_ID.pdf
- Code
  - Filename: STUDENT_ID.zip
- Deadline: 3/21 Fri. 10:10 a.m.