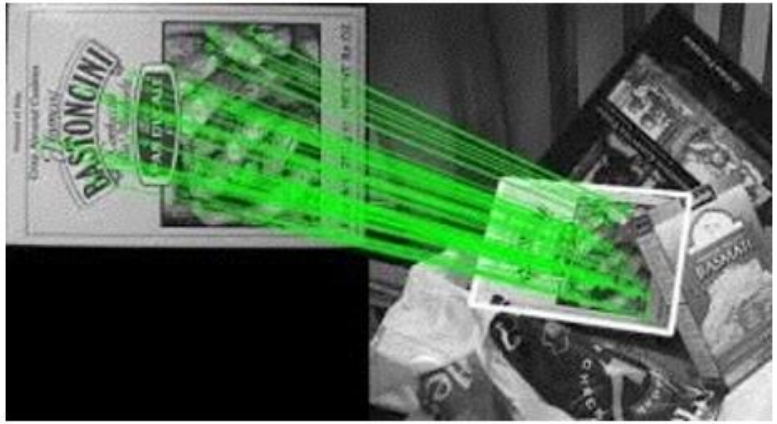
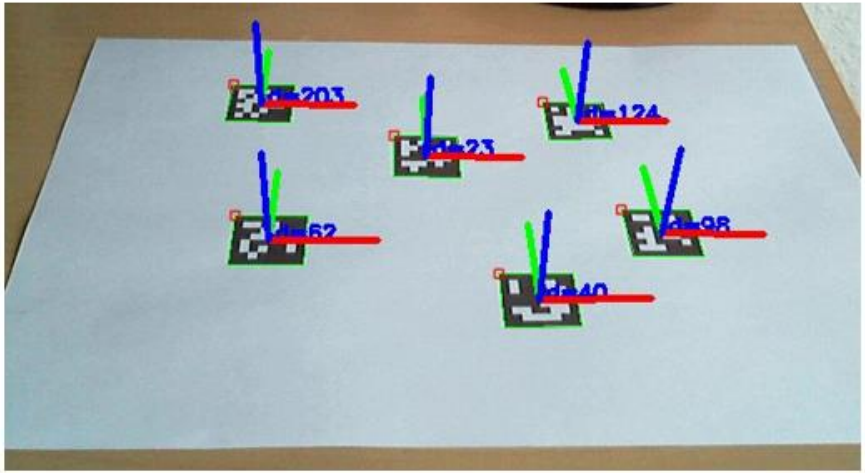
3/12 lab

1. OpenCV introduction
2. OpenCV installation
3. Lab1

- 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

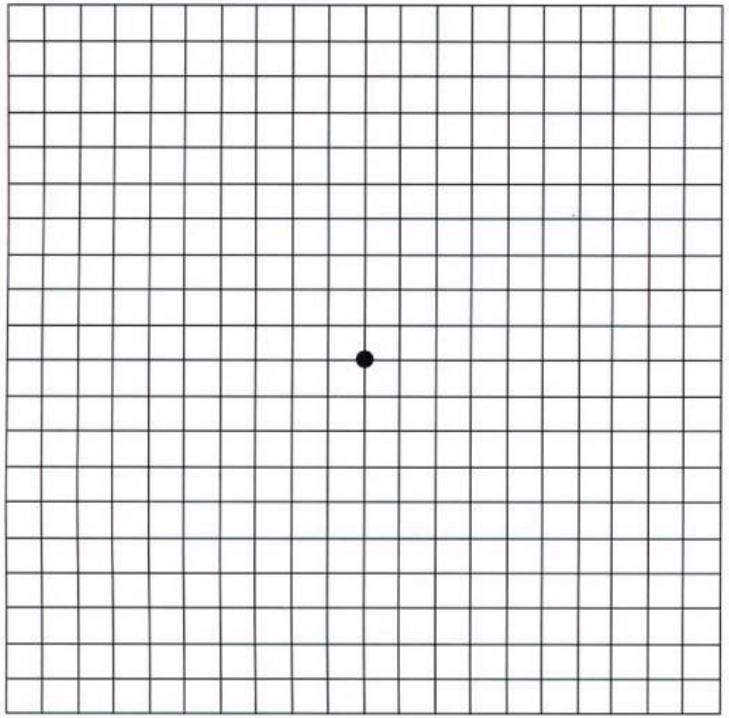
feature detection





pattern recognition

Mat



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

normal:

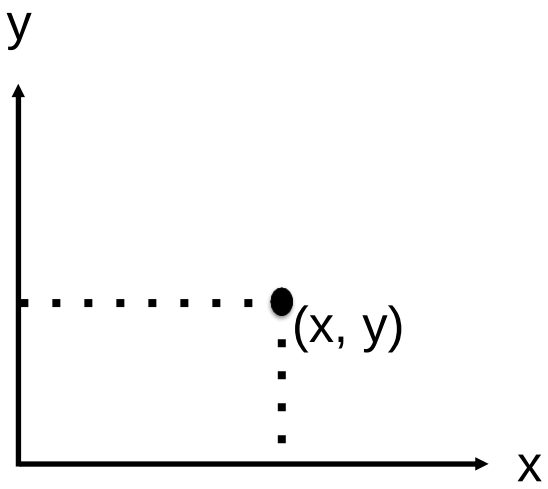


image:

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

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

Mat(int rows, int cols, int type, const cv::Scalar &s)

type:
CV_8U, CV_8S, CV_16U, CV_32F, CV_8UC3

```
Mat img1(240, 320, CV_8U);
Mat img2(240, 320, CV_8U, Scalar(100));
Mat img3(240, 320, CV_8UC3, Scalar(200,100,0));
```

“=“ and “clone”

```
Mat img1(240, 320, CV_8U, Scalar(100));
Mat img2, img3;
img2 = img1;
img3 = img1.clone();
```

Python

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

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

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

標頭引入

```cpp
#include <opencv2/opencv.hpp>
using namespace cv;

int main(){
  Mat img = imread( .... );
  imwrite( .... );

     ... ...

  imshow("Display window", img);
  waitKey(0);
  return 0;
}
```

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

讀寫圖片

讀取:

```
Mat imread(
    const string& filename,
    int flags=1),          flag > 0: three channel
                           flag = 0: gray scale
```

儲存:

```
bool imwrite(
    const string& filename,
    InputArray img,
    const vector& params=vector())
```

讀寫圖片

讀取:

```
Mat image;
image = imread("girl.jpg", 1 );
```

儲存:

```
imwrite( "girl backup.jpg", image );
```

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

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

顯示圖片

秀出影像:

```
void imshow(
    const string& window_name,
    InputArray mat)
```

等待按鍵輸入:

```
imshow("Display window", img);
waitKey(0);
```

```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()
```

操作像素

灰階:

Mat gray_img(100, 100, **CV_8U**, Scalar(100));

gray_img.at<**uchar**>(30,20) =255;

彩色:

Mat color_img(100, 100, **CV_8UC3**, Scalar(200,100,0));

img.at<**Vec3b**>(30,20)[0] =255;

image[row, col, channel]

OpenCV
in
Visual Studio

for Linux:

https://gist.github.com/MarcWang/0547f87cf777b6576275

https://www.learnopencv.com/install-opencv3-on-ubuntu/

for MacOS:

https://www.learnopencv.com/install-opencv3-on-macos/

Step 1 Download OpenCV

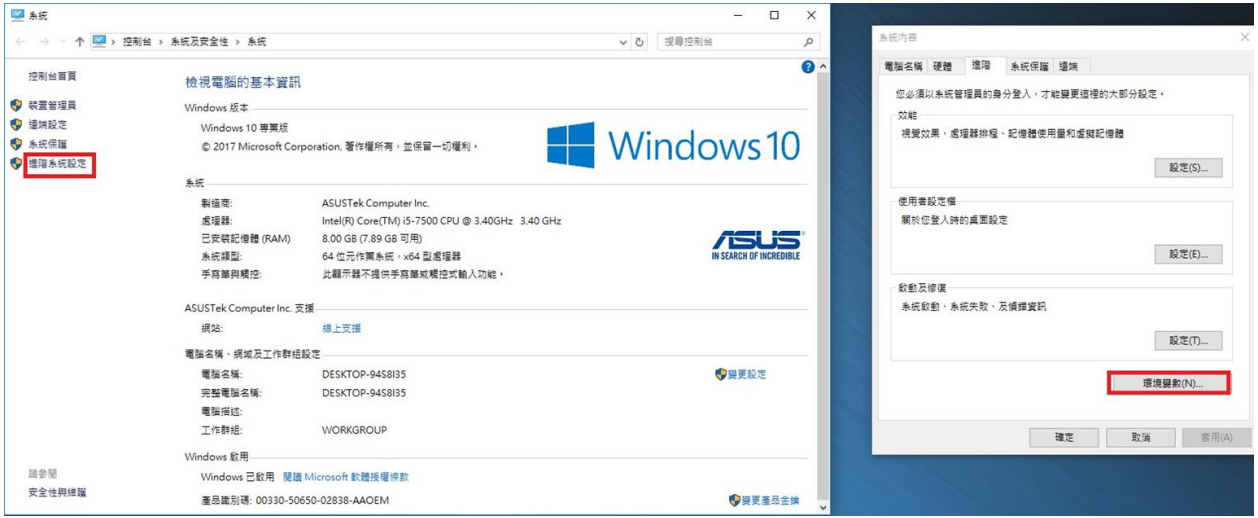
- OpenCV 3.4.0 download link:
  https://downloads.sourceforge.net/project/opencvlibrary/opencv-win/3.4.0/opencv-3.4.0-vc14_vc15.exe?r=https%3A%2F%2Fopencv.org%2Fopencv-3-4.html&ts=1519635075&use_mirror=nchc

- Extract it to a proper directory

  ex. C:\opencv

Step 2 System Path Setting

- 電腦 > 右鍵內容 > 進階系統設定 > 進階 >環境變數



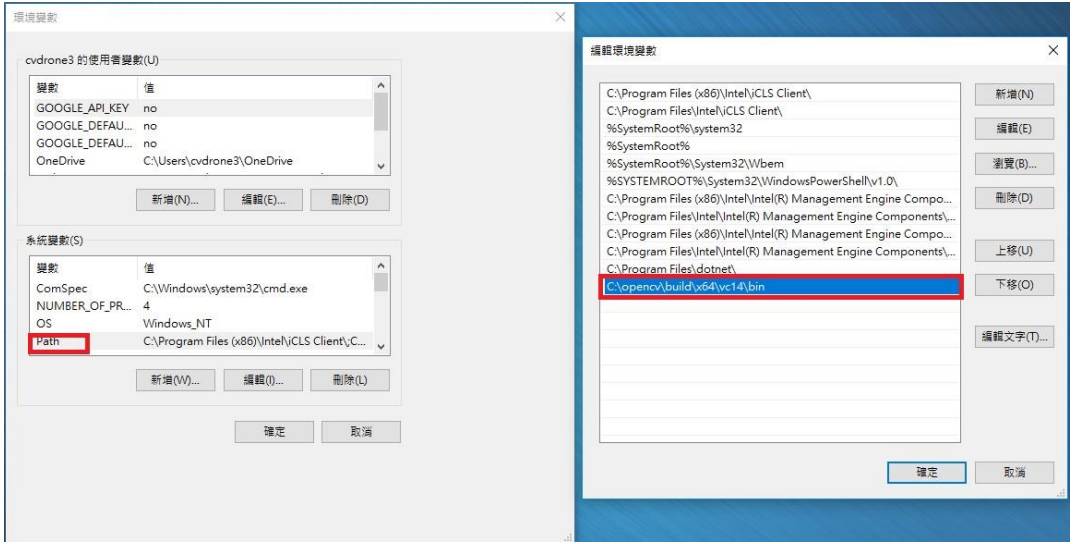
Step 2 System Path Setting

- 系統變數 > Path > 編輯 > 新增opencv資料夾路徑

  Ex: C:\opencv\build\x64\vc15\bin

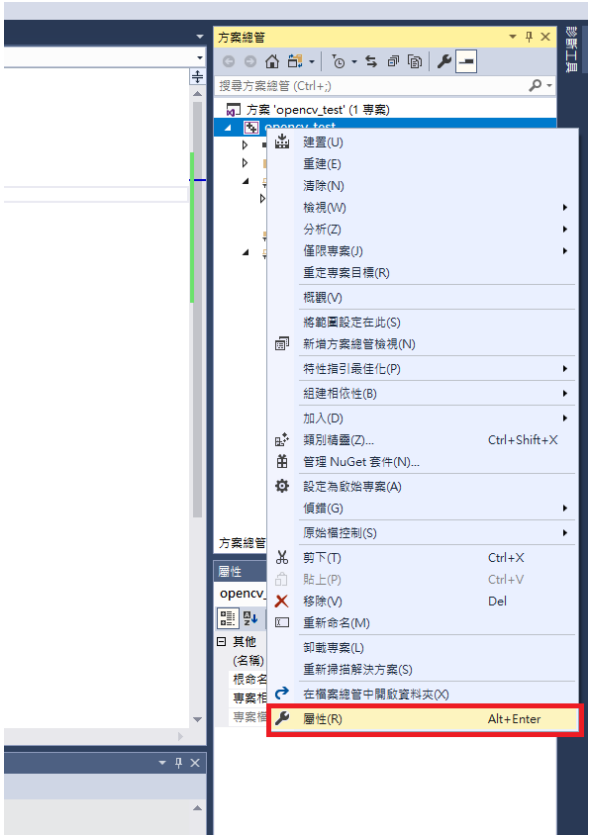
  (vc15 for visual studio 2017)

- 重新開機



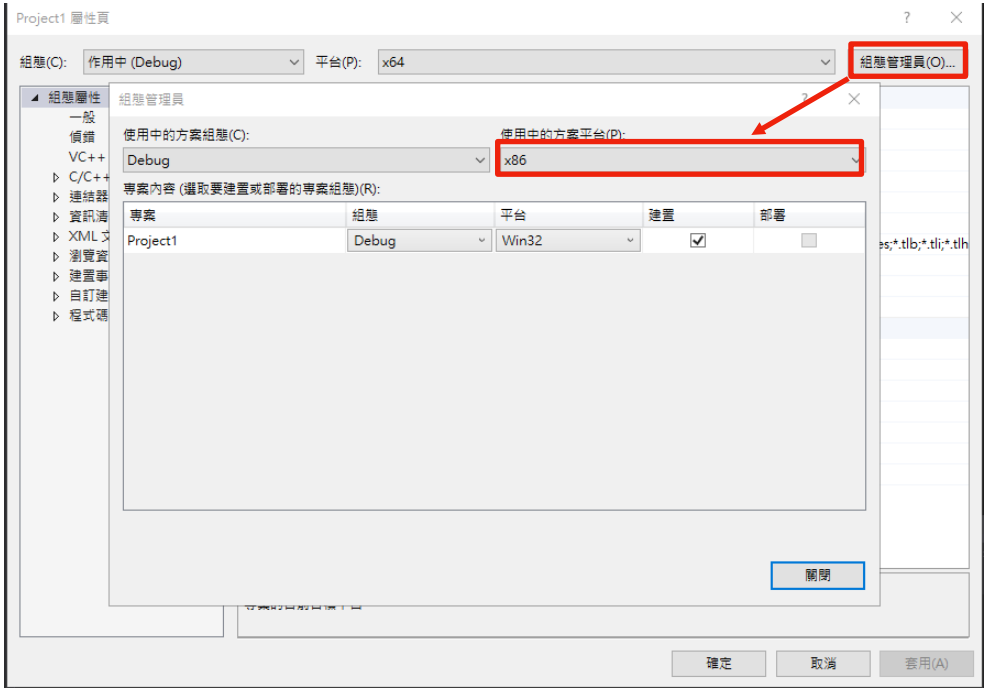
Step 2 Project properties

- Create a new Project
- > 專案 > 右鍵 > 屬性



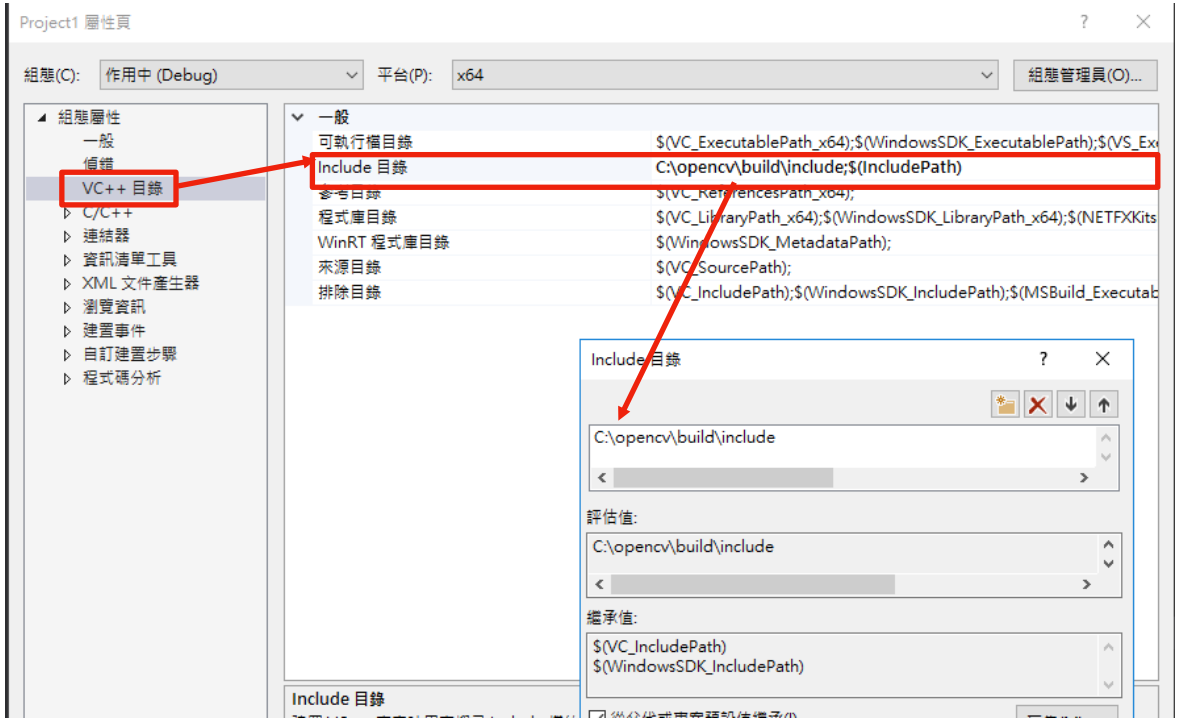
Step 2 Project properties

- 組態管理員 > 新增 x64 (Win32 平台)



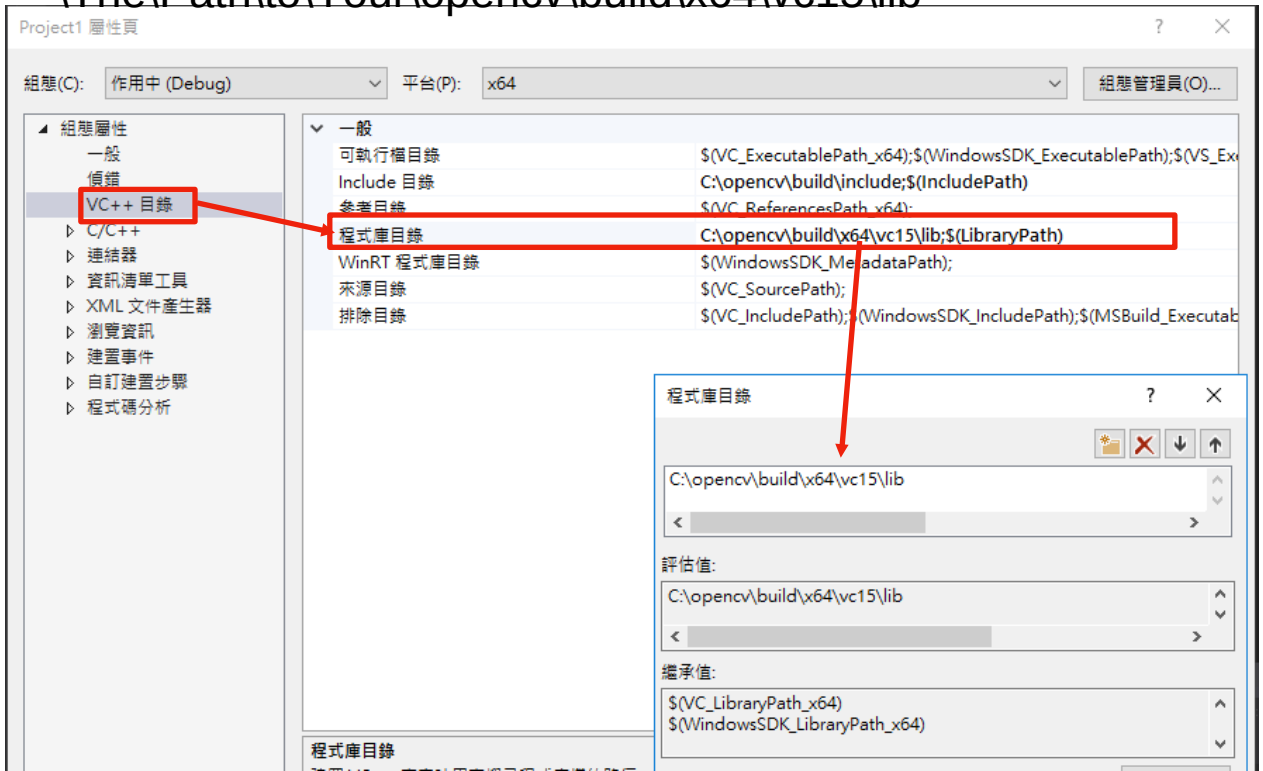
Step 2 Project properties

- VC++ 目錄 > include 目錄 > 編輯 > 編輯opencv路徑 ~~\The\Path\to\Your~~\opencv\build\include



Step 2 Project properties

- VC++ 目錄 > 程式庫目錄 > 編輯 > 新增opencv路徑 \The\Path\to\Your\opencv\build\x64\vc15\lib



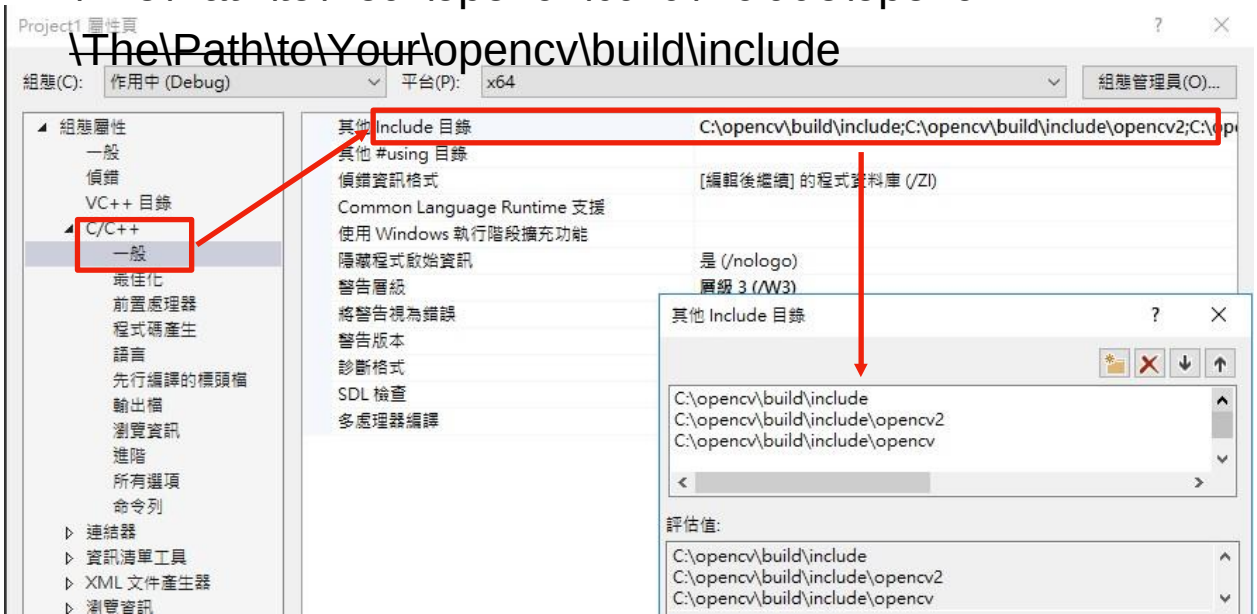
Step 2 Project properties

- C/C++ > 一般> 其他include目錄 > 編輯 > 新增以下三項

  \The\Path\to\Your\opencv\build\include

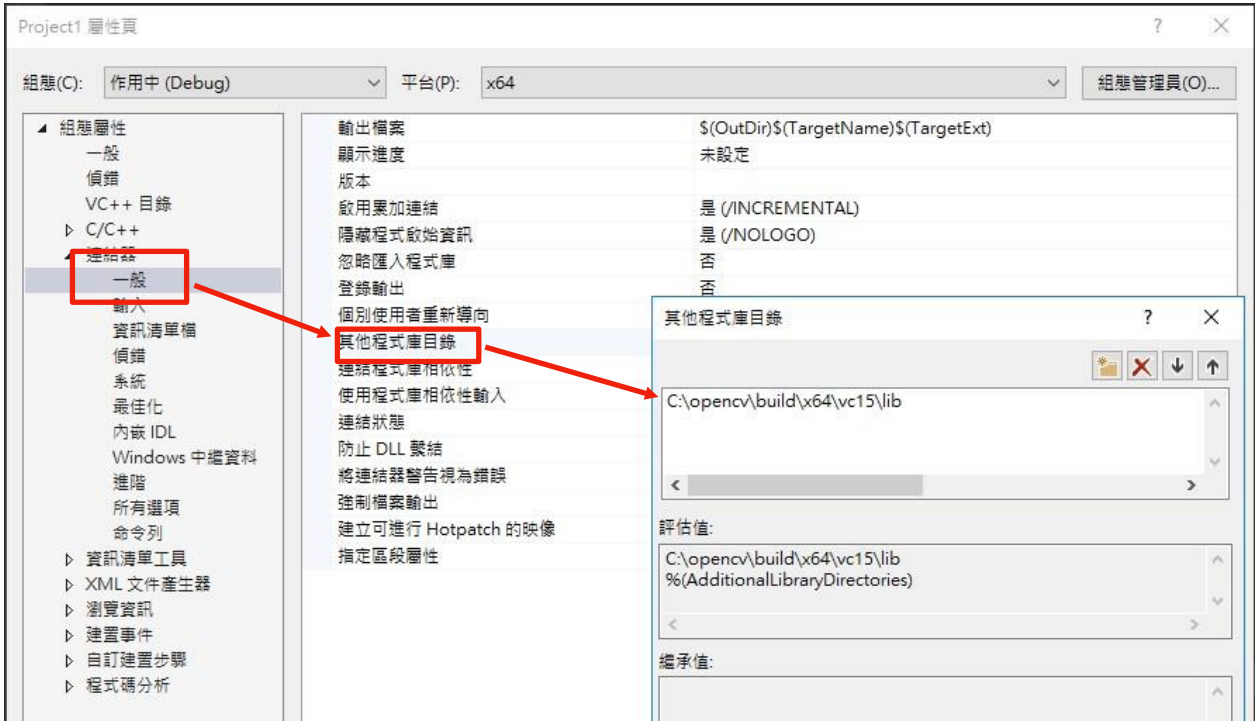
  \The\Path\to\Your\opencv\build\include\opencv

  \The\Path\to\Your\opencv\build\include



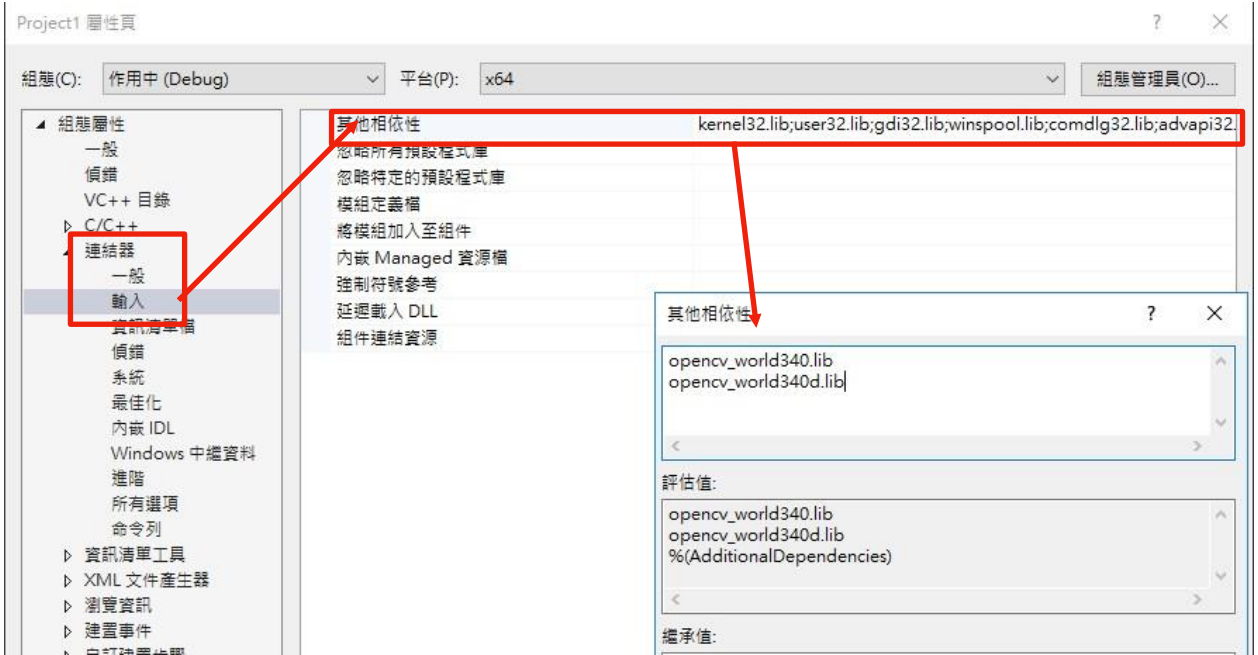
Step 2 Project properties

- 連結器 > 一般 > 其他程式庫目錄 > 編輯 > 新增~~\The\Path\to\Your~~\opencv\build\x64\vc15\lib



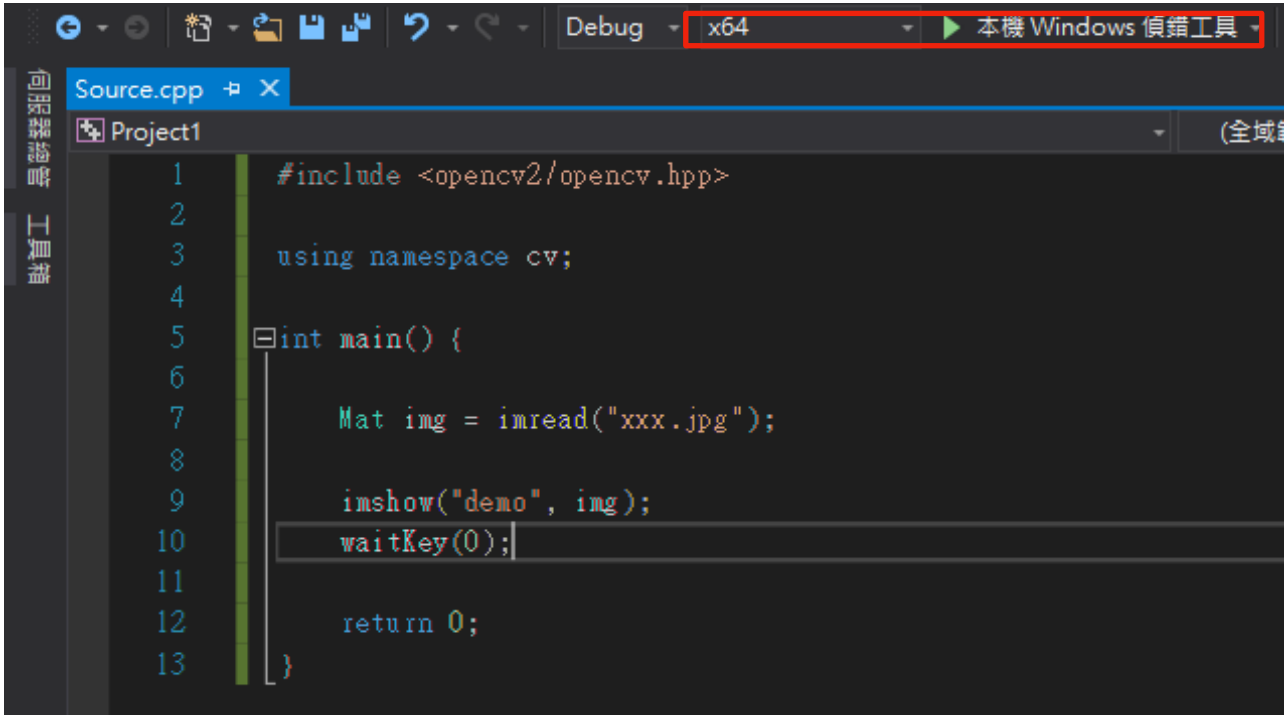
Step 2 Project properties

- 連結器 > 輸入 > 其他相依性 > 編輯 > 新增 opencv_world340.lib
  opencv_world340d.lib



Step 3 Test

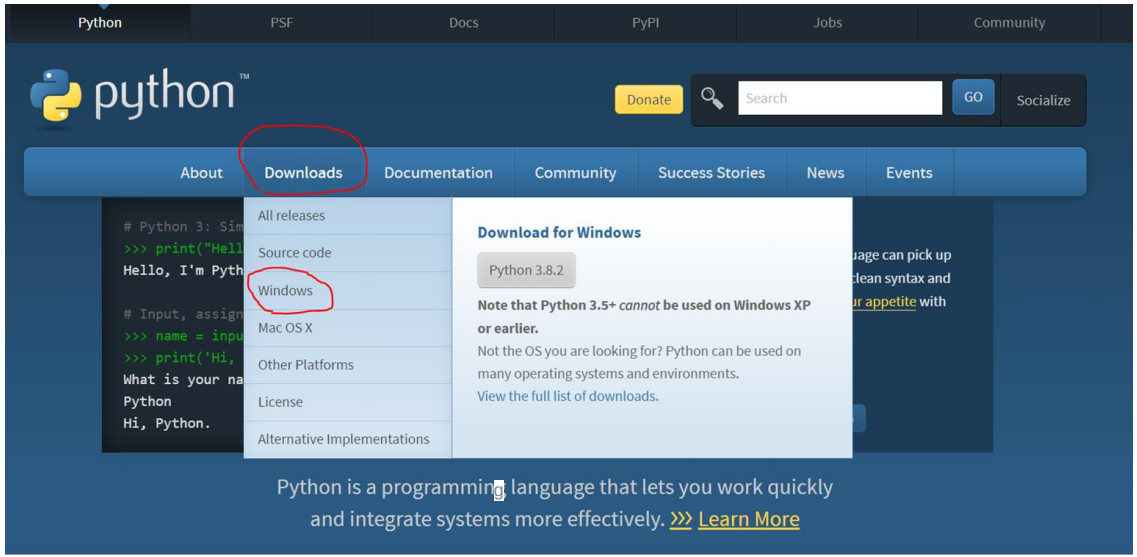
- 建置專案前將x86改為x64



- Windows10
- python 2.7 & opencv

一、下載python

到官網點選Downloads



一、下載python

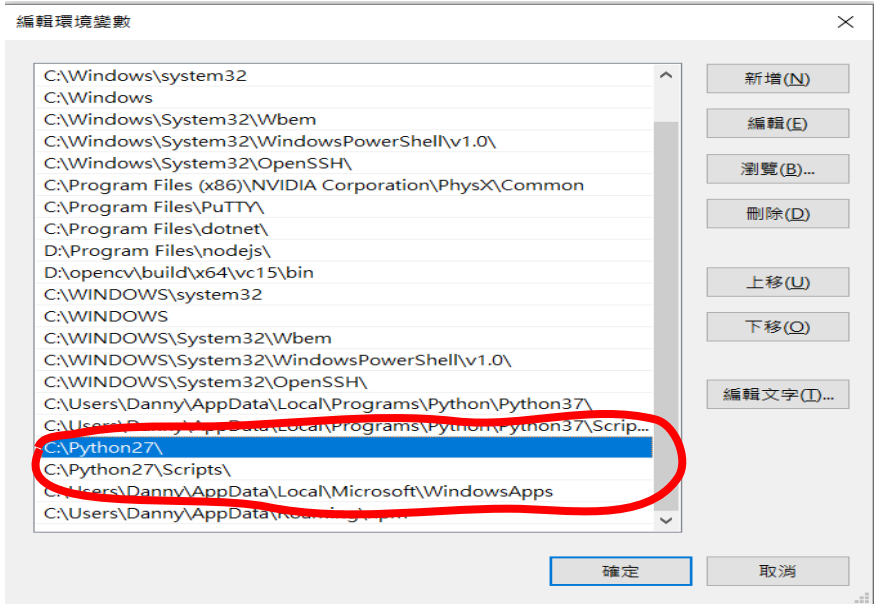
選擇python 2.7

二、安裝軟體

點選並安裝

三、配置環境

新增環境變數



四、安裝opencv

- pip install opencv-python
- Test :

```
1  import cv2
2
3  img = cv2.imread('kobe.jpg')
4
5  cv2.imshow('My Image', img)
6  cv2.waitKey(0)
7  cv2.destroyAllWindows()
8
```

Lab 01

彩色影像轉灰階影像
nearest neighbor Interpolation
Bilinear Interpolation

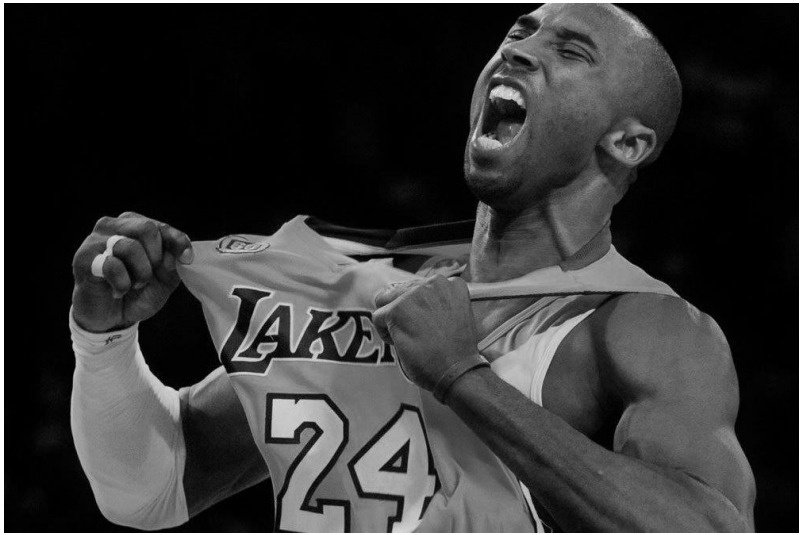
1. 彩色影像轉灰階影像(20%)

- 取得單通道影像中，像素(i, j)的強度:
  uchar intensity = img.at<uchar>(i, j);

- 取得三通道影像中，像素(i, j)的紅色強度:
  uchar intensity = img.at<Vec3b>(i, j)[2];

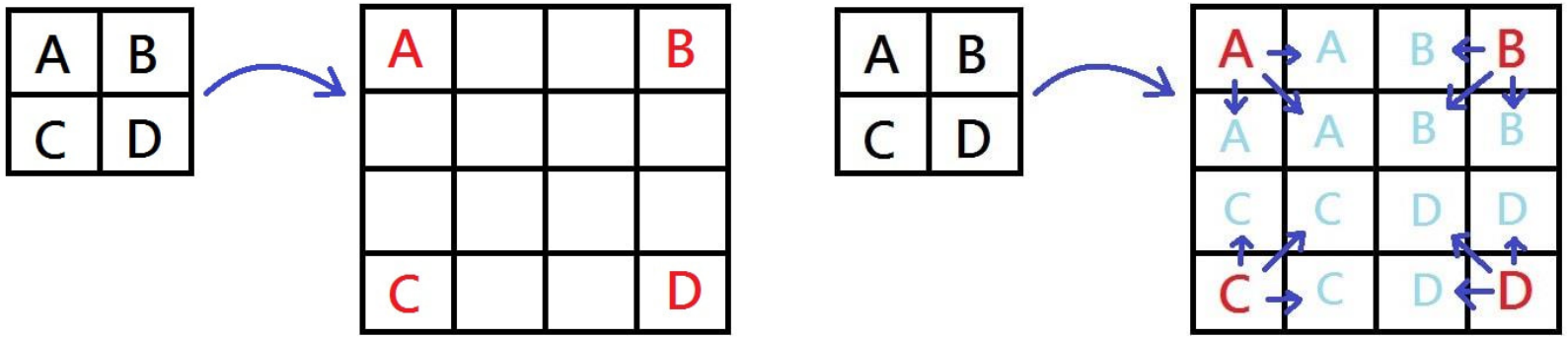
1. 彩色影像轉灰階影像(20%)

- Gray = (R + G + B) /3



2. Interpolation - 最近相鄰內插法 (40%)

- 根據輸出影像的像素位置,找到輸入影像中最鄰近的點,即當作輸出影像的像素強度。
- 以下圖為例

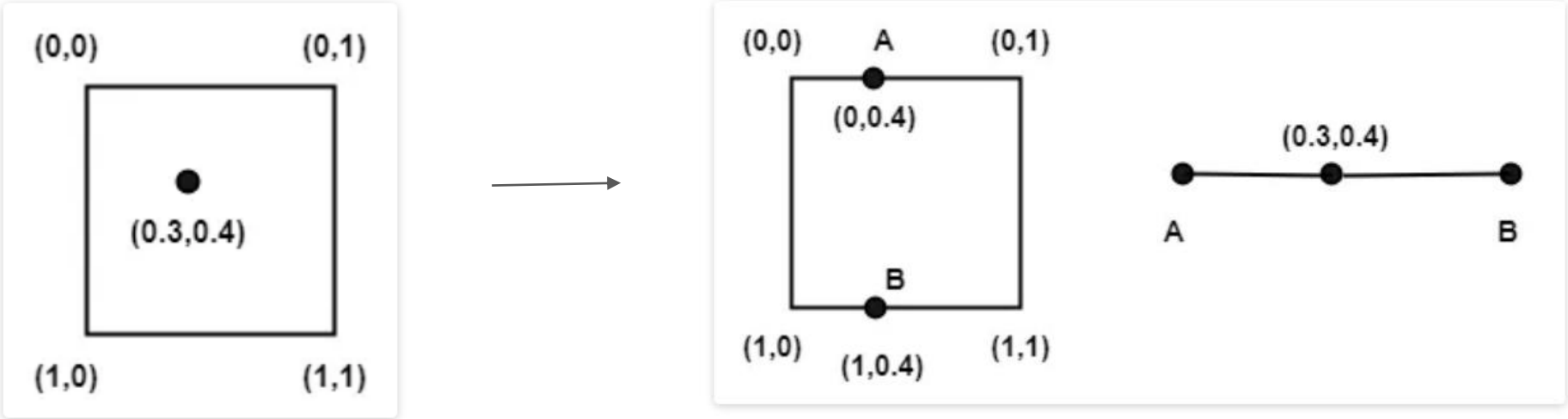


2. Interpolation - 最近相鄰內插法 (40%)

- 將照片放大3倍

3. Interpolation - 雙線性內插法 (40%)

- 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。



3. Interpolation - 雙線性內插法 (40%)

- 右圖為輸入影像
  下圖為
  倍率0.7之結果

下次demo

- 最鄰近點 – 放大3倍
- 雙線性 – 縮小0.7倍