

# 多媒體技術與應用

## Spring 2022

Instructor : Yen-Lin Chen(陳彥霖), Ph.D.

Professor

Dept. Computer Science and Information Engineering

National Taipei University of Technology



# Lecture 1

OpenCV 常用影像處理技術簡介



# OpenCV 簡介



# OpenCV 簡介

- OpenCV：最早是由Intel公司所開發的開放原始碼電腦影像視覺函式庫( Open Source Computer Vision Library )，它由一系列 C 函數和 C++ 構成，實現了圖像處理和電腦視覺方面的很多通用演算法。
- 最新版本為4.5.5。
- 更多關於OpenCV：<https://opencv.org/>







# OpenCV跨平台性

- OpenCV可以在 Windows、MacOS、Linux、Android與IOS等作業系統，只要選擇OpenCV來當作影像開發函式庫，那麼所開發出來的程式將能輕鬆移植至其他作業平台。
- OpenCV支援的使用環境如下：
  - Visual C++ .net
  - Eclipse IDE
  - C++ Builder IDE
  - DevCpp IDE
  - Visual C++ and Microsoft's DirectShow
  - Xcode (Mac)
  - Python
  - Linux



# OpenCV程式語言

- OpenCV使用C++語言編寫，它的主要介面也是C++語言，但是依然保留了大量的C語言介面。OpenCV也有大量的Python，Java 和 MATLAB / OCTAVE(版本2.5)的介面。這些語言的API介面函式可以通過線上文件獲得。現在也提供對於C#, Ch, Ruby的支援。
- 所有新的開發和演算法都是用C++介面。
- OpenCV從第三版開始已加入深度學習的模組 (OpenCV 3.3)。

# 基礎影像分析 – 應用OpenCV於車道線偵測為例

車道辨識在自駕車領域是非常重要的主題，除了傳統的影像處理手段之外，現今的研究也常常使用機器學習的網路模型辨識出車前攝影機影像中的車道線。

Radius of Curvature = 627  
Vehicle is 0.40 m left of center







# 基礎影像分析 – 應用 OpenCV 於車道線偵測為例(機器學習流程)

雖然機器學習非常方便、速度快又可達到高準確率，但即使是使用機器學習的方法，仍須對影像作前處理。前處理大概包含有：

- 色彩轉換（灰階/HSV）
- 門檻值二值化
- 物件標示

而常用的影像相關機器學習架構有CNN、YOLO等。





# 基礎影像分析 – 應用 OpenCV 於車道線偵測為例(傳統影像處理)

在傳統影像處理的手段中，車道辨識大概可以包含以下步驟：

1. 將RGB影像轉為灰階/HSV影像
2. 對影像作增強/降噪、以形態學方法放大或縮小特徵
3. 以門檻值做二值化篩選出車道線
4. 以邊緣檢測方法抓出車道線的形狀

在電腦視覺領域上，最常被使用的影像套件則是OpenCV。



# OpenCV 常用功能介紹



# OpenCV常用語法－導言

- 影像處理的基本要素就是圖片，在OpenCV中，C++版是使用Mat的資料型態儲存圖片資料，Python版則是使用NumPy的ndarray來儲存。
- 每個影像處理的步驟都可以使用Mat或ndarray儲存起來，顯示出來相互比較。
- 官方document: <https://docs.opencv.org/4.1.0/index.html>





# OpenCV – 常用語法(numpy.ndarray) (Python)

- NumPy是 Python 的運算套件，提供陣列 (ndarray)、矩陣 (matrix).....等常用的數學工具，運算速度比 Python 內建的資料格式快很多。





# OpenCV – 常用語法(numpy.ndarray) (Python)

- `numpy.ndarray(<shape>, <dtype>[, <buffer>, <offset>, <strides>, <order>])`
  - `shape`: 指定陣列長寬
  - `dtype`: 指定資料型態

<code>int8</code>	8位元整數，有負號(-128~127)
<code>int16</code>	16位元整數，無負號(-32768~32767)
<code>uint8</code>	8位元整數，無負號(0~255)
<code>uint16</code>	16位元正數，無負號(0~65535)
<code>float16</code>	半精度浮點數，包括1個符號位元，5個指數位，10個尾數位
<code>float32</code>	單精確度浮點數，包括：1個符號位元，8個指數位，23個尾數位

NumPy常用資料型態



# OpenCV – 常用語法(numpy.ndarray) (Python)

- `numpy.ndarray(<shape>, <dtype>[, <buffer>, <offset>, <strides>, <order>])`
  - `buffer`: 用於初始化陣列的buffer
  - `offset`: buffer中用於初始化陣列的首個資料的偏移量
  - `strides`: int型tuple，每個軸的下標增加1時,資料指標在記憶體中增加的位元組數
  - `order`: 用於決定初始化陣列的順序，"C"為row優先，"F"為column優先
  - 例:`img = numpy.ndarray(shape=(1080, 1920), dtype=numpy.uint8)`





# OpenCV – 常用語法(numpy.ndarray) (Python)

- 其他常用的ndarray如下:

<code>numpy.array()</code>	回傳一陣列
<code>numpy.empty()</code>	回傳一陣列其每個元素皆未初始化
<code>numpy.zeros()</code>	回傳一陣列其每個元素皆為0
<code>numpy.ones()</code>	回傳一陣列其每個元素皆為1

- 各項函式詳細介紹請見下頁

參考資料:<https://numpy.org/doc/stable/reference/generated/numpy.ndarray.html>



# OpenCV – 常用語法(numpy.array) (Python)

- `numpy.array(<shape>, <dtype>[, <copy>, <order>, <subok>, <ndmin>, <like>])`
  - `shape`: 指定陣列長寬
  - `dtype`: 指定資料型態
  - `copy`: If true (default), then the object is copied. Otherwise, a copy will only be made if `__array__` returns a copy
  - `order`: 用於決定初始化陣列的順序，” C”為row優先，” F”為column優先
  - `subok`: If True, then sub-classes will be passed-through, otherwise the returned array will be forced to be a base-class array (default).
  - `ndmin`: 指定陣列的維度
  - `like`: Reference object to allow the creation of arrays which are not NumPy arrays.
  - 例: `img = numpy.array(shape=(1080, 1920), dtype=numpy.uint8)`



# OpenCV – 常用語法(numpy.empty) (Python)

- `numpy.empty(<shape>, <dtype>[, <order>, <like>])`
  - `shape`: 指定陣列長寬
  - `dtype`: 指定資料型態
  - `order`: 用於決定初始化陣列的順序，” C”為row優先，” F”為column優先
  - `like`: Reference object to allow the creation of arrays which are not NumPy arrays.
  - 例: `img = numpy.empty(shape=(1080, 1920), dtype=numpy.uint8)`





# OpenCV – 常用語法(numpy.zeros) (Python)

- 創建全黑的影像圖
- `numpy.zeros(<shape>, <dtype>[, <order>, <like>])`
  - `shape`: 指定陣列長寬
  - `dtype`: 指定資料型態
  - `order`: 用於決定初始化陣列的順序，” C”為row優先，” F”為column優先
  - `like`: Reference object to allow the creation of arrays which are not NumPy arrays.
  - 例: `img = numpy.zeros(shape=(1080, 1920), dtype=numpy.uint8)`



# OpenCV – 常用語法(numpy.ones) (Python)

- `numpy.ones(<shape>, <dtype>[, <order>, <like>])`
  - `shape`: 指定陣列長寬
  - `dtype`: 指定資料型態
  - `order`: 用於決定初始化陣列的順序，” C”為row優先，” F”為column優先
  - `like`: Reference object to allow the creation of arrays which are not NumPy arrays.
  - 例: `img = numpy.ones(shape=(1080, 1920), dtype=numpy.uint8)`



# OpenCV – 常用語法(numpy.shape()) (Python)

- shape函式是numpy.core.fromnumeric中的函式，它的功能是讀取矩陣的長度或寬度，比如shape[0]就是讀取矩陣的第一維度的長度。
- 例：
  - `img = numpy.zeros(shape=(1080, 1920), dtype=numpy.uint8)`
  - `height = img.shape[0]`                      #回傳1080
  - `width = img.shape[1]`                      #回傳1920





# OpenCV – 常用語法(Mat)(C++)

- Mat的資料結構中，包含長、寬、像素型態、像素深度、通道數等資訊，以下為Mat之成員變數及函式。

成員變數	定義
rows	影像列的數量，也就是影像高
cols	影像行的數量，也就是影像寬

- OpenCV影像尺寸：`Size Mat::size() const`
- 影像之大小 `Size(cols, rows)`，cols和rows分別為寬和高。
- OpenCV通道數：`int Mat::channels() const`
- 影像之通道數：灰階圖為1，彩色圖為3。



# OpenCV – 常用語法(Mat)(C++)

- OpenCV像素型態：`int Mat::type() const`

CV_8U	8位元整數，無負號，通道數1
CV_8S	8位元整數，有負號，通道數1
CV_16U	16位元整數，無負號，通道數1
CV_32F	32位元浮點數，通道數1
CV_8UC3	8位元整數，無負號，通道數3



# OpenCV – 常用語法(Mat)(C++)

## OpenCV Mat函式：

- **Mat(int rows, int cols, int type, const cv::Scalar &s)**
  - rows：影像之高度。
  - cols：影像之寬度。
  - type：影像之型態。
  - s：像素值，為像素強度，灰階或BGR。
- 使用方法
  - Mat img1(480, 720, CV\_8);
  - Mat img1(480, 720, CV\_8, Scalar(128));
  - Mat img1(480, 720, CV\_8UC3, Scalar(256, 128, 0));





# OpenCV – 常用語法(Mat)(C++)

## 影像複製：

- OpenCV 等號多載 `Mat& Mat::operator = (const Mat& img)` 。
- OpenCV 影像複製：`Mat::copyTo(Mat& img) const` 。
- `img`：輸入影像，左邊影像和右邊影像相同。
- OpenCV 影像複製：`Mat Mat::clone() const` 。
- `img`：輸出影像，輸出影像會有相同的長、寬、像素值。
- `Mat img1(480, 720, CV_8U);`
- `Mat img2, img3, img4;`
- `img2 = img1;` // 第一種
- `img1.copyTo(img3);` // 第二種
- `img4 = img1.clone();` // 第三種(最常使用)



# OpenCV – 常用語法(Mat)(C++)

## 操作像素

- 我們可用at()得到或改變某個像素值，這函式使用模板，所以使用時除了輸入位置，還必須輸入影像的像素型態，使用at()函式時，輸入參數順序為at(高,寬)。
- OpenCV改變像素：template T& Mat::at(int i, int j)
- at<type> : type 有 uchar, unsigned char, float, double
- Mat Image (720, 480, CV\_8U, Scalar(100));
- Image.at<uchar>(400, 100) = 255; //把位置(400,100)的值改成255



# OpenCV – 常用語法(imread)(Python)

- 讀取圖片

- `cv2.imread(<filename>[, <flags>])`

- filename:檔案路徑(含檔名、副檔名)

- flags:

- `cv2.LOAD_IMAGE_ANYDEPTH` - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.

- `cv2.LOAD_IMAGE_COLOR` - If set, always convert image to the color one(常用)

- `cv2.LOAD_IMAGE_GRAYSCALE` - If set, always convert image to the grayscale one

- .....

- 例: `frame = cv2.imread(“./pic1.jpg”)`





# OpenCV – 常用語法(imwrite)(Python)

- 儲存影像成圖片

- `cv2.imwrite( <filename>, <img>, [, <params>])`

- filename:輸出的檔案路徑(含檔名、副檔名)

- img:欲輸出的圖片(ndarray格式)

- params :

- `cv2.IMWRITE_JPEG_QUALITY` - For JPEG, it can be a quality from 0 to 100 (the higher is the better). Default value is 95.

- `cv2.IMWRITE_JPEG_PROGRESSIVE` - Enable JPEG features, 0 or 1, default is False.

- `cv2.IMWRITE_JPEG_OPTIMIZE` - Enable JPEG features, 0 or 1, default is False.

- `cv2.IMWRITE_JPEG_RST_INTERVAL` - JPEG restart interval, 0 - 65535, default is 0 - no restart.

- .....

- 例: `cv2.imwrite( “./pic1.jpg”, frame)`



# OpenCV – 常用語法(imshow)(Python)

- 顯示圖片
  - `cv2.imshow(<window_name>, <img>)`
    - `window_name`:顯示的視窗名稱
    - `img`:欲輸出的圖片(ndarray格式)
    - 例: `cv2.imshow(“showPic1”, frame)`



# OpenCV – 常用語法(imread, imwrite)(C++)

- 讀取圖片

- `cv::imread(<路徑>, <flag>);`

- flags:

- CV\_LOAD\_IMAGE\_ANYDEPTH - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.
    - CV\_LOAD\_IMAGE\_COLOR - If set, always convert image to the color one(常用)
    - CV\_LOAD\_IMAGE\_GRAYSCALE - If set, always convert image to the grayscale one
    - Ex: `frame = cv::imread("./picInput.jpg", CV_LOAD_IMAGE_COLOR)`

- 儲存影像成圖片

- `cv::imwrite(<路徑>, <影像(Mat)>);`

- Ex: `cv::imwrite("./pic1.jpg", frame);`





# OpenCV – 常用語法(imshow)(C++)

- 顯示圖片
  - `cv::imshow(<window_name>, <img>)`
    - `window_name`:顯示的視窗名稱
    - `img`:欲輸出的圖片(Mat格式)
    - 例:`cv::imshow(“showPic1”, frame)`



# OpenCV – 常用語法(resize) (Python)

## OpenCV改變尺寸大小

- `cv2.resize(<src>, <dsiz>[, <dst>[, <fx>[, <fy>[, <interpolation>]]]])`
  - `src`:輸入影像
  - `dsiz`:輸出尺寸，當輸入為0時，`fx`、`fy`皆不可為0，  
`dsiz=Size(round(fxsrc.cols), round(fysrc.rows))`
  - `fx`:水平縮放比率，當輸入為0，`fx=(double)dsiz.width/src.cols`。
  - `fy`:垂直縮放比率，當輸入為0時，`fy=(double)dsiz.height/src.rows`。
  - `interpolation`:插值方式。



# OpenCV – 常用語法(resize) (Python)

- interpolation：插值方式
- 當影像改變大小时，要將像素填入新的影像。
  - CV\_INTER\_NEAREST：最鄰近插點法。
  - CV\_INTER\_LINEAR：雙線性插值(預設)。
  - CV\_INTER\_AREA：臨域像素再取樣插值。
  - CV\_INTER\_CUBIC：雙立方插值，4x4大小的補點。
  - CV\_INTER\_LANCZOS4：Lanczos插值，8x8大小的補點。
- [https://docs.opencv.org/4.1.0/da/d54/group\\_\\_imgproc\\_\\_transform.html#ga47a974309e9102f5f08231edc7e7529d](https://docs.opencv.org/4.1.0/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d)



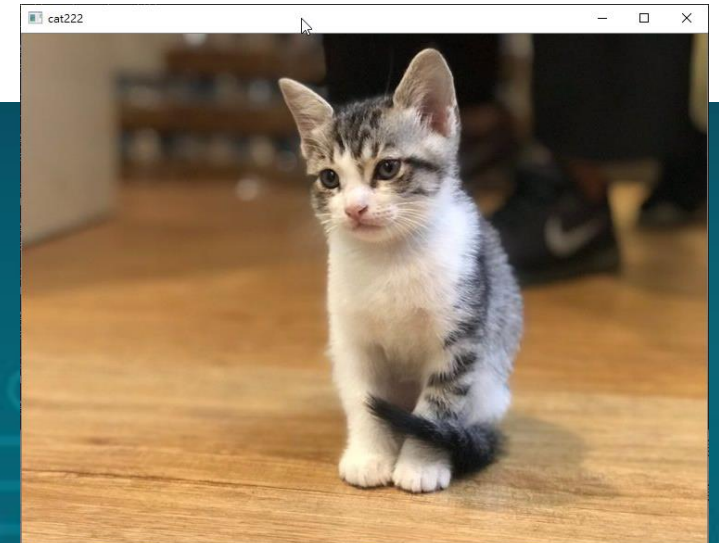
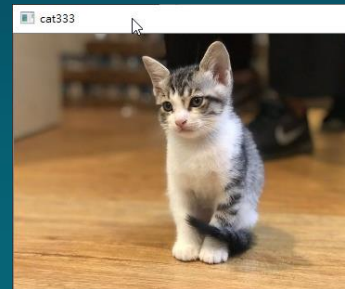


# OpenCV – 常用語法(resize) (Python)

- 範例程式:

- [4]讀檔
- [5]顯示原始圖片
- [7, 8]取得原始圖片寬長
- [9, 10]長寬各縮小0.5倍
- [11]顯示縮小後的圖片
- [13]暫停直到輸入任意鍵
- [14]關掉所有視窗

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("cat111.jpg")
5 cv2.imshow("cat222", img)
6
7 width = img.shape[1]
8 height = img.shape[0]
9 img2 = cv2.resize(img, (int(width*0.5), int(height*0.5)), \
10                    interpolation=cv2.INTER_CUBIC)
11 cv2.imshow("cat333", img2)
12
13 cv2.waitKey(0)
14 cv2.destroyAllWindows()
15
```





# OpenCV – 常用語法(resize)(C++)

## OpenCV改變尺寸大小

- `void cv::resize(InputArray src, OutputArray dst, Size dsize, double fx=0, double fy=0, int interpolation=INTER_LINEAR)`
- `src` : 輸入影像。
- `dst` : 輸出影像：型態會和輸入圖相同。
- `dsize` : 輸出尺寸，當輸入為0時，`fx`、`fy`皆不可為0，  
`dsize=Size(round(fxsrc.cols), round(fysrc.rows))`
- `fx` : 水平縮放比例，當輸入為0，`fx=(double)dsize.width/src.cols`。
- `fy` : 垂直縮放比例，當輸入為0時，`fy=(double)dsize.height/src.rows`。
- `interpolation` : 插值方式。



# OpenCV – 常用語法(resize)(C++)

- interpolation：插值方式
- 當影像改變大小时，要在原本的像素之間填入新的像素。
  - CV\_INTER\_NEAREST：最鄰近插點法。
  - CV\_INTER\_LINEAR：雙線性插值(預設)。
  - CV\_INTER\_AREA：臨域像素再取樣插值。
  - CV\_INTER\_CUBIC：雙立方插值，4×4大小的補點。
  - CV\_INTER\_LANCZOS4：Lanczos插值，8×8大小的補點。





# OpenCV – 常用語法(waitKey) (Python)

- `cv2.waitKey(ms)`。
- `waitKey`無限制時間等待一個鍵盤輸入（當參數 $\leq 0$ ）或延遲毫秒等候時間（參數 $> 0$ ）。
- 由於作業系統在切換執行緒之間具有最小時間，因此該功能將不會精準的等待正確的設定的延遲(ms)，只保證等待時間至少設定的延遲（ms），取決於當時在您的平台上執行的內容。
- 如果在指定的延遲時間過去之前沒有按下任何鍵，它將返回-1，若有按下按鍵，他將返回按下的按鍵ASCII。



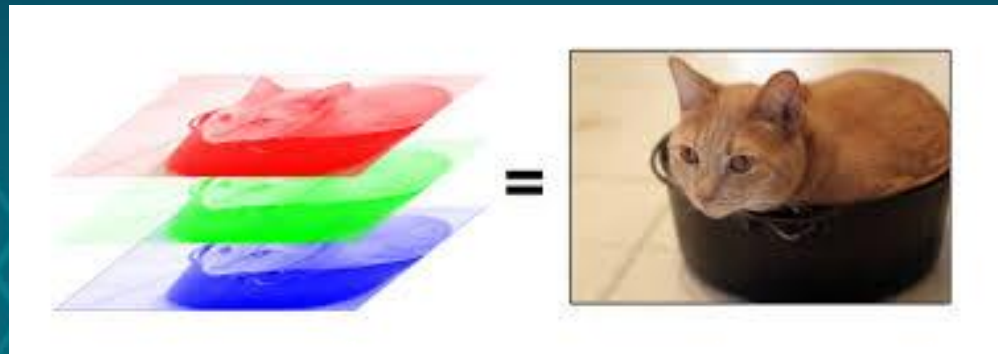
# OpenCV – 常用語法(waitKey)(C++)

- `cv::waitKey(ms)`。
- `waitKey`無限制時間等待一個鍵盤輸入（當參數 $\leq 0$ ）或延遲毫秒等候時間（參數 $> 0$ ）。
- 由於作業系統在切換執行緒之間具有最小時間，因此該功能將不會精準的等待正確的設定的延遲(ms)，只保證等待時間至少設定的延遲（ms），取決於當時在您的平台上執行的內容。
- 如果在指定的延遲時間過去之前沒有按下任何鍵，它將返回-1，若有按下按鍵，他將返回按下的按鍵ASCII。



# 影像通道與色彩空間簡介

- 影像通道說明:
- 一張彩色的影像有3個通道(R, G, B)，即一個二維陣列上的每一個像素點都有R, G, B三個值(值域0~255)，也可以想成是一個三維的陣列。
- 一張黑白的影像只有1個通道，即一個二維陣列上的每一個像素點只有一個灰階值(值域0~255)，也可以想成是一個二維的陣列。



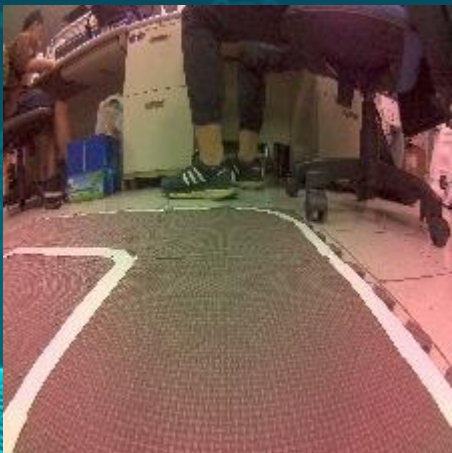
<https://blog.csdn.net/cywtriancai/article/details/80601407>





# 灰階(BGR TO GRAY)-簡介

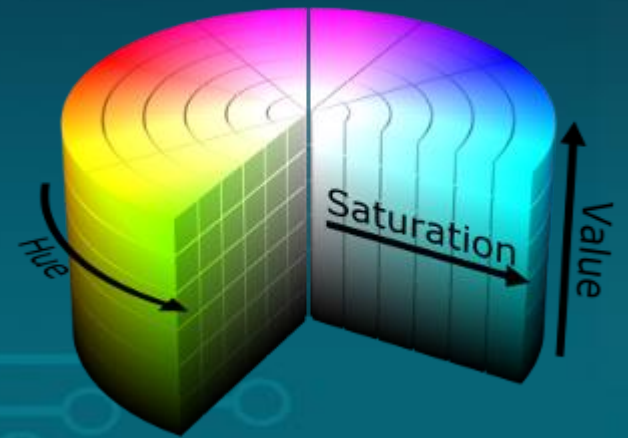
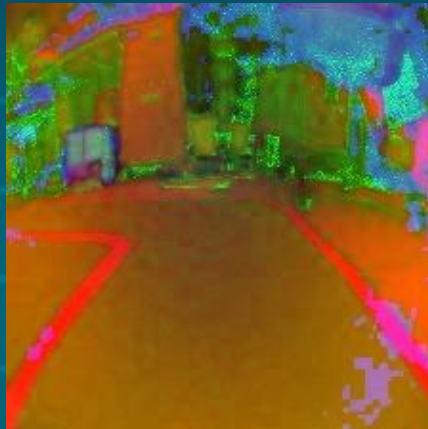
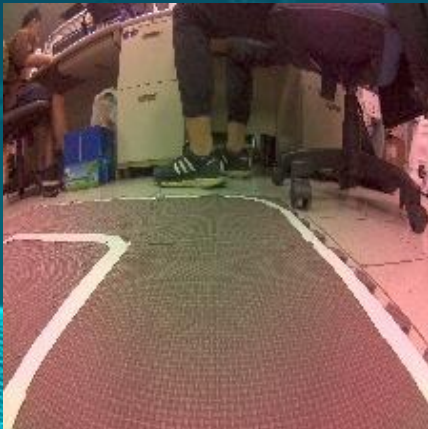
- 將彩色的影像轉化為灰階。
- 彩色轉灰階原理：
  - 由於人眼對綠色的敏感度最大，對藍色敏感度最小，因此綠色的權重分配會較大，藍色較小，如下公式為彩色轉灰階的標準。
- BGR to Gray公式： $Y=0.299R + 0.587G + 0.114B$





# HSV(BGR TO HSV)-簡介

- 將BGR的影像轉化為HSV。
- HSV是一種將RGB色彩轉為圓柱坐標系的表示法：
  - H：色相(Hue)，是色彩的基本屬性，如紅色、黃色等。
  - S：飽和度(Saturation)，是指色彩的純度，越高色彩越純，低則逐漸變灰，取0-100%的數值。
  - V：明度(Value)，取0-100%的數值。





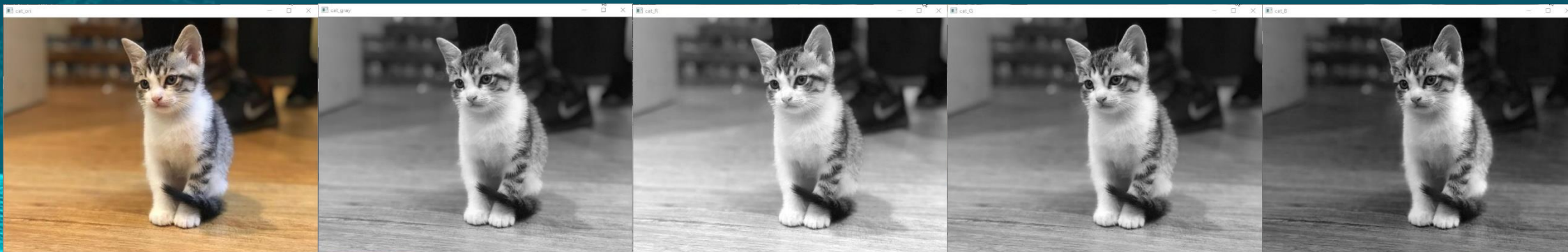


# OpenCV – 常用語法(channels)

- 範例程式:

- [4]讀圖(imread預設通道順序為B,G,R)
- [5]顯示圖片
- [7]RGB空間轉灰度空間(3通道→單通道)
- [8]顯示圖片
- [10, 11, 12]分別取得R, G, B通道(此時img\_R, img\_G, img\_B皆為單通道的圖)
- [13, 14, 15]分別顯示圖片(圖片為單通道，所以顯示出來為灰階圖)

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("cat.jpg")
5 cv2.imshow("cat_ori", img)
6
7 img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
8 cv2.imshow("cat_gray", img_gray)
9
10 img_R = img[:, :, 2]
11 img_G = img[:, :, 1]
12 img_B = img[:, :, 0]
13 cv2.imshow("cat_R", img_R)
14 cv2.imshow("cat_G", img_G)
15 cv2.imshow("cat_B", img_B)
16
17 cv2.waitKey(0)
18 cv2.destroyAllWindows()
```



原始圖(3通道)

灰階圖(單通道)

R(單通道)

G(單通道)

B(單通道)



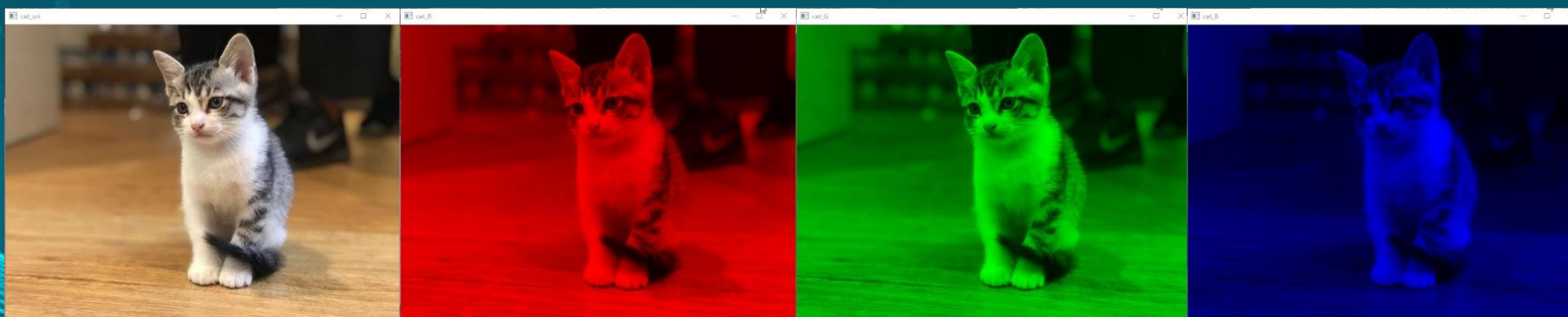


# OpenCV – 常用語法(channels)

- 範例程式:

- [4]讀圖(imread預設通道順序為B,G,R)
- [5]顯示圖片
- [7, 8]取得原始圖長寬
- [10, 11, 12]創建3個3維陣列( height, width, channel)
- [13, 14, 15]分別取得R, G, B通道(此時img\_R, img\_G, img\_B皆為3通道的圖)
- [16, 17, 18]分別顯示圖片(圖片為3通道，所以顯示出來為有顏色的圖)

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("cat.jpg")
5 cv2.imshow("cat_ori", img)
6
7 height = img.shape[0]
8 width = img.shape[1]
9
10 img_R = np.zeros((height, width, 3), np.uint8)
11 img_G = np.zeros((height, width, 3), np.uint8)
12 img_B = np.zeros((height, width, 3), np.uint8)
13 img_R[:, :, 2] = img[:, :, 2]
14 img_G[:, :, 1] = img[:, :, 1]
15 img_B[:, :, 0] = img[:, :, 0]
16 cv2.imshow("cat_R", img_R)
17 cv2.imshow("cat_G", img_G)
18 cv2.imshow("cat_B", img_B)
19
20 cv2.waitKey(0)
21 cv2.destroyAllWindows()
```



原始圖(3通道)

R (3通道)

G (3通道)

B (3通道)



# OpenCV – 常用語法(cvtColor)(Python)

- `dst = cv2.cvtColor(<src>, <code>[, <dst>[, <dstCn>]])`
  - src: 來源圖像(ndarray)
  - code: 色彩空間轉換參數
  - dst: 輸出圖像(ndarray)
  - dstCn: 影像通道數
  - 例: `img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)`

色彩空間轉換參數	說明
<code>cv2.COLOR_BGR2GRAY</code>	從RGB空間轉換為灰度空間
<code>cv2.COLOR_BGR2HSV</code>	從RGB空間轉換為HSV空間
<code>cv2.COLOR_BGR2YUV</code>	從RGB空間轉換為YUV空間
<code>cv2.COLOR_BGR2YCrCb</code>	從RGB空間轉換為YCrCb空間



# OpenCV – 常用語法(cvtColor)(C++)

- Mat為BGR而不是RGB格式，所以輸入參數通常使用CV\_BGR2GRAY
- cvtColor(const Mat& src, Mat& dst, int code)
  - src：輸入影像。
  - dst：輸出影像，尺寸大小和深度會與輸入影像相同。
  - code：指定在何種色彩空間轉換，比如CV\_BGR2GRAY、CV\_GRAY2BGR、CV\_BGR2HSV等。





# OpenCV – 常用語法(繪製圖形) (Python)

- 繪製線段:

- `cv2.line(<img>, <pt1>, <pt2>, <color>[, <thickness>[, <lineType>[, <shift>]])`
  - `img`: 來源影像
  - `pt1`: 線段起始座標
  - `pt2`: 線段終點座標
  - `color`: 指定線段顏色(B, G, R)
  - `thickness`: 指定線段粗細
  - `lineType`: 指定線段型態
  - `shift`: 點坐標中的小數位數
  - 例: `img = cv2.line(img, (2, 2), (10, 10), (255, 0, 0))`



# OpenCV – 常用語法(繪製圖形) (C++)

- 繪製線段:
  - `cv::line(InputOutputArray img, Point pt1, Point pt2, Scalar& color, int thickness, int lineType, int shift)`
    - img:來源影像
    - pt1:線段起始座標
    - pt2:線段終點座標
    - color:指定線段顏色(B, G, R)
    - thickness:指定線段粗細
    - lineType:指定線段型態
    - shift:點坐標中的小數位數



# OpenCV – 常用語法(繪製圖形) (Python)

- 繪製方框:
  - `cv2.rectangle(<img>, <pt1>, <pt2>, <color>[, <thickness>[, <lineType>[, <shift>]])`
  - `img`: 來源影像
  - `pt1`: 左上角頂點座標
  - `pt2`: 右下角頂點座標
  - `color`: 指定方框顏色(B, G, R)
  - `thickness`: 指定方框粗細
  - `lineType`: 指定方框型態
  - `shift`: 點坐標中的小數位數
  - 例: `img = cv2.rectangle(img, (2, 2), (10, 10), (255, 0, 0))`





# OpenCV – 常用語法(繪製圖形) (C++)

- 繪製方框:

- `cv::rectangle(InputOutputArray img, Point pt1, Point pt2, Scalar& color, int thickness, int lineType, int shift)`
- `img`: 來源影像
- `pt1`: 左上角頂點座標
- `pt2`: 右下角頂點座標
- `color`: 指定方框顏色(B, G, R)
- `thickness`: 指定方框粗細
- `lineType`: 指定方框型態
- `shift`: 點坐標中的小數位數



# OpenCV – 常用語法(繪製圖形) (Python)

- 繪製圓圈:

- `cv2.circle(<img>, <center>, <radius>, <color>[, <thickness>[, <lineType>[, <shift>]]])`
- `img`: 來源影像
- `center`: 圓心座標
- `radius`: 半徑
- `color`: 指定圓圈顏色(B, G, R)
- `thickness`: 指定圓圈粗細
- `lineType`: 指定圓圈型態
- `shift`: 點坐標中的小數位數
- 例: `img = cv2.circle(img, (2, 2), (10, 10), (255, 0, 0))`



# OpenCV – 常用語法(繪製圖形) (C++)

- 繪製圓圈：
  - `cv::circle(InputOutputArray img, Point center, int radius, Scalar& color, int thickness, int lineType, int shift)`
  - `img`: 來源影像
  - `center`: 圓心座標
  - `radius`: 半徑
  - `color`: 指定圓圈顏色(B, G, R)
  - `thickness`: 指定圓圈粗細
  - `lineType`: 指定圓圈型態
  - `shift`: 點坐標中的小數位數





# OpenCV – 常用語法(繪製圖形) (Python)

- 繪製文字:

- `cv2.putText(<img>, <text>, <org>, <fontFace>, <fontScale>, <color>[, <thickness>[, <lineType>[, <bottomLeftOrigin>]])`
- `img`: 來源影像
- `text`: 欲寫出來的文字內容
- `org`: 寫出文字的位置(文字左下角座標)
- `fontFace`: 字型
- `fontScale`: 字體大小
- `color`: 指定字體顏色(B, G, R)
- `thickness`: 指定字體粗細
- `lineType`: 指定字體型態
- `bottomLeftOrigin`: 為布林值，若為true表示會從左下開始寫字，反之從左上開始寫字
- 例: `img = cv2.putText(img, "Hello", (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1, cv2.LINE_AA)`



# OpenCV – 常用語法(繪製圖形) (C++)

- 繪製文字:

- `cv::putText(InputOutputArray img, String& text, Point org, int fontFace, double fontScale, Scalar& color, int thickness, int lineType, bool bottomLeftOrigin)`
- `img`: 來源影像
- `text`: 欲寫出來的文字內容
- `org`: 寫出文字的位置(文字左下角座標)
- `fontFace`: 字型s
- `fontScale`: 字體大小
- `color`: 指定字體顏色(B, G, R)
- `thickness`: 指定字體粗細
- `lineType`: 指定字體型態
- `bottomLeftOrigin`: 為布林值，若為true表示會從左下開始寫字，反之從左上開始寫字

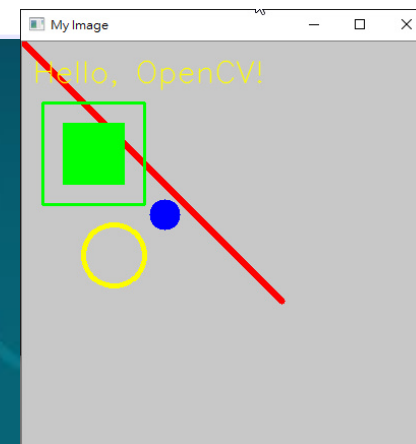


# OpenCV – 常用語法(繪製圖形)

- 範例程式:

- [4]創建400x400大小的3通道圖片
- [5]圖片每個像素值指定為200
- [7]畫線
- [8, 9]畫方框
- [10, 11]畫圓圈
- [14]寫字
- [14, 15]後面的\是跳脫字元，若要換行要打\  
1
- [17]顯示圖片
- [18]暫停直到輸入任意鍵
- [19]關閉所有視窗

```
1 import cv2
2 import numpy as np
3
4 img = np.zeros((400, 400, 3), np.uint8)
5 img.fill(200)
6
7 cv2.line(img, (0, 0), (255, 255), (0, 0, 255), 5)
8 cv2.rectangle(img, (20, 60), (120, 160), (0, 255, 0), 2)
9 cv2.rectangle(img, (40, 80), (100, 140), (0, 255, 0), -1)
10 cv2.circle(img, (90, 210), 30, (0, 255, 255), 3)
11 cv2.circle(img, (140, 170), 15, (255, 0, 0), -1)
12
13 text = 'Hello, OpenCV!'
14 cv2.putText(img, text, (10, 40), cv2.FONT_HERSHEY_SIMPLEX, \
15             1, (0, 255, 255), 1, cv2.LINE_AA)
16
17 cv2.imshow('My Image', img)
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```







# OpenCV – 常用語法(讀寫影片) (Python)

- 讀取影片

- `cv2.VideoCapture(<filename>)`
  - filename:檔案路徑(含檔名、副檔名)
  - 例: `cap = cv2.VideoCapture (“video123.avi”)`

- 寫入影片

- `cv.VideoWriter(<filename>, <fourcc>, <fps>, <frameSize>[, <isColor>])`
  - filename:檔案路徑(含檔名、副檔名)
  - fourcc:檔案編碼格式
  - fps:指定影片的fps
  - frameSize:指定影片的解析度
  - isClolor:若給值為0，會以灰階寫入影片frame。若無給值，會以原色彩寫入影片frame
  - 例: `output_movie = cv2.VideoWriter('output.avi', cv2.VideoWriter_fourcc(*'XVID'), fps, (1920, 1080) )`



# OpenCV – 常用語法(讀寫影片)

- 範例程式碼:

- [1]讀影片

- [3]設定輸出影片編碼

- [4]設定輸出影片格式

- [6]用while迴圈取得輸入影片的每一個frame

- [7]取得當前frame

- [8]若有取得frame

- [9]將frame寫入輸出影片

- [12, 13]釋放輸入及輸出影片

```
1  cap = cv2.VideoCapture('input.avi')
2
3  fourcc = cv2.VideoWriter_fourcc(*'XVID')
4  output = cv2.VideoWriter('output.avi', fourcc, 30, (1920, 1080))
5
6  while(cap.isOpened()):
7      ret, frame = cap.read()
8      if ret == True:
9          output.write(frame)
10     else:
11         break
12 cap.release()
13 output.release()
```



# OpenCV – 常用語法(讀寫影片)(C++)

- 讀取影片

- `cv::VideoCapture(String& filename)`
  - filename:檔案路徑(含檔名、副檔名)

- 寫入影片

- `cv::VideoWriter(String& filename, int fourcc, double fps, Size frameSize, bool isColor)`
  - filename:檔案路徑(含檔名、副檔名)
  - fourcc:檔案編碼格式
  - fps:指定影片的fps
  - frameSize:指定影片的解析度
  - isClolor:若給值為0，會以灰階寫入影片frame。若無給值，會以原色彩寫入影片frame

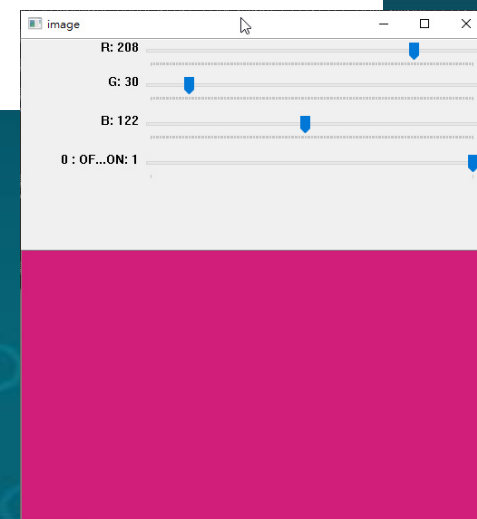




# OpenCV – 調色盤

- 範例程式：
  - [7]創建300x512解析度的3通道空白圖
  - [8]命名視窗名稱
  - [10~15]建立4個scroll bar代表R, G, B值、開關
  - [17]顯示圖片
  - [19]按下”esc”鍵結束程式
  - [22~25]取得R, G, B, 開關值
  - [26~29]若開關值==0，顯示黑色；反之顯示給定的R, G, B值所產生出來的顏色圖

```
1 import cv2
2 import numpy as np
3
4 def nothing(x):
5     pass
6
7 # Create a black image, a window
8 img = np.zeros((300,512,3), np.uint8)
9 cv2.namedWindow('image')
10 # create trackbars for color change
11 cv2.createTrackbar('R','image',0,255,nothing)
12 cv2.createTrackbar('G','image',0,255,nothing)
13 cv2.createTrackbar('B','image',0,255,nothing)
14 # create switch for ON/OFF functionality
15 switch = '0 : OFF \n1 : ON'
16 cv2.createTrackbar(switch, 'image',0,1,nothing)
17 while(1):
18     cv2.imshow('image',img)
19     k = cv2.waitKey(1) & 0xFF
20     if k == 27:
21         break
22     # get current positions of four trackbars
23     r = cv2.getTrackbarPos('R','image')
24     g = cv2.getTrackbarPos('G','image')
25     b = cv2.getTrackbarPos('B','image')
26     s = cv2.getTrackbarPos(switch,'image')
27     if s == 0:
28         img[:] = 0
29     else:
30         img[:] = [b,g,r]
```





# OpenCV – 滑鼠游標畫圖(1)

## • 範例程式:

- [8]定義函式draw\_circle()
- [10~12]滑鼠左鍵按下的事件
- [13~18]滑鼠游標移動的事件、畫方框或圓形
- [19~24]滑鼠左鍵放開的事件、畫方框或圓形
- [26]建立512x512解析度的3通道圖
- [27]設定視窗名稱
- [28]設定滑鼠事件函式
- [30]顯示圖片
- [32~33]按下”m”鍵時切換mode(畫方框或圓形)
- [34]按下”esc”鍵結束程式

```
1 import cv2
2 import numpy as np
3
4 drawing = False # true if mouse is pressed
5 mode = True # if True, draw rectangle. Press 'm' to toggle to curve
6 ix,iy = -1,-1
7 # mouse callback function
8 def draw_circle(event,x,y,flags,param):
9     global ix,iy,drawing,mode
10    if event == cv2.EVENT_LBUTTONDOWN:
11        drawing = True
12        ix,iy = x,y
13    elif event == cv2.EVENT_MOUSEMOVE:
14        if drawing == True:
15            if mode == True:
16                cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
17            else:
18                cv2.circle(img,(x,y),5,(0,0,255),-1)
19    elif event == cv2.EVENT_LBUTTONUP:
20        drawing = False
21        if mode == True:
22            cv2.rectangle(img,(ix,iy),(x,y),(0,255,0),-1)
23        else:
24            cv2.circle(img,(x,y),5,(0,0,255),-1)
25
26 img = np.zeros((512,512,3), np.uint8)
27 cv2.namedWindow('image')
28 cv2.setMouseCallback('image',draw_circle)
29 while(1):
30     cv2.imshow('image',img)
31     k = cv2.waitKey(1) & 0xFF
32     if k == ord('m'):
33         mode = not mode
34     elif k == 27:
35         break
36 cv2.destroyAllWindows()
```





# OpenCV — 全觀

- 目前OpenCV包含如下幾個部份：
  - core : The Core Functionality.
  - Imgproc : Image Processing Library. It include :
    - Image Filtering
    - Geometric Image Transformations
    - Miscellaneous Image Transformations
    - Histograms
    - Structural Analysis and Shape Descriptors
    - Motion Analysis and Object Tracking
    - Feature Detection
    - Object Detection
  - highgui : High-level GUI and Media I/O Library.



# OpenCV Image Processing



# 二值化(Binarization)-簡介

- 二值化(Binarization)是影像分割的一種方法，我們會將影像分為兩個部分，一個是感興趣的部分(前景)，以及不感興趣的部分(背景)，會依據門檻值(threshold)當作分割的標準，通常會以強度超過門檻值的像素當作前景，反之則為背景。
- 門檻值的算法主要分兩類：
  - 固定門檻值：直接給定一個灰階值當門檻值，再用這個門檻值進行二值化。
  - 自適應門檻值(Otsu決定法)：會依據輸入影像計算出較合適的門檻值，再用這個門檻值進行二值化。



## 二值化-利用門檻值進行影像分割

- 在一灰階影像當中，每一獨立區塊的產生取決於亮度（灰階值）的不同，利用這樣的特性，我們可以設立一固定的門檻值T，以該值為條件，將影像轉換為二值影像（只包含0及1之二值影像，或0及255之灰階影像），並可協助我們在後續處理中擷取出感興趣的影像區塊。

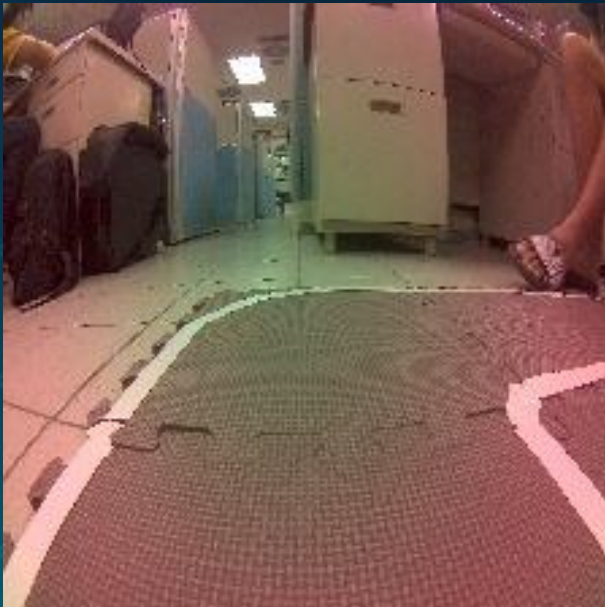
$$g(x, y) = \begin{cases} 1 & f(x, y) \geq T \\ 0 & f(x, y) < T \end{cases}$$

$$g(x, y) = \begin{cases} 0 & f(x, y) \geq T \\ 1 & f(x, y) < T \end{cases}$$

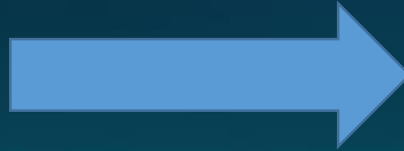




# 二值化-利用門檻值對車道影像進行分割



$$g(x, y) \geq 170$$



$$g(x, y) = \begin{cases} 1 & f(x, y) \geq 170 \\ 0 & f(x, y) < 170 \end{cases}$$



# 二值化-OpenCV

## OpenCV 固定門檻值(threshold)二值化

- `retval, dst = cv2.threshold(<src>, <thresh>, <maxval>, <type>[, <dst>])`
  - `retval`：只是一個return值，並不會使用到
  - `dst`：輸出影像(ndarray)，尺寸大小、深度會和輸入影像相同。
  - `src`：輸入影像(ndarray)，只能輸入單通道，8位元或32位元浮點數影像。
  - `thresh`：門檻值。
  - `maxval`：二值化結果的最大值。
  - `type`：二值化操作型態，共有THRESH\_BINARY、THRESH\_BINARY\_INV、THRESH\_TRUNC、THRESH\_TOZERO、THRESH\_TOZERO\_INV五種。
  - 例：`ret, img_out = cv2.threshold(img, 170, 255, cv2.THRESH_BINARY)`
  - [https://docs.opencv.org/4.1.0/d7/d1b/group\\_\\_imgproc\\_\\_misc.html#gae8a4a146d1ca78c626a53577199e9c57](https://docs.opencv.org/4.1.0/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57)



# 二值化-OpenCV

常用五種門檻值參數介紹：

threshold門檻值參數	說明
cv2.THRESH_BINARY	大於門檻值的像素設為最大值(maxval)，否則設為0。
cv2.THRESH_BINARY_INV	大於門檻值的像素設為0，否則設為最大值(maxval)。
cv2.THRESH_TRUNC	大於門檻值的像素設為門檻值，否則不變。
cv2.THRESH_TOZERO	大於門檻值的像素值不變，否則設為0。
cv2.THRESH_TOZERO_INV	大於門檻值的像素值設為0，否則不變。
cv2.THRESH_OTSU	自適應決定門檻值，大於門檻值的像素設為最大值(maxval)，否則不變。





# 二值化-OpenCV

- 範例程式:

- [4] 以灰階讀入圖片(後面參數0)
- [5] THRESH\_BINARY: 將大於門檻值的像素設為255，否則設為0。
- [6] THRESH\_BINARY\_INV: 大於門檻值的像素設為0，否則設為255。
- [7] THRESH\_TRUNC: 大於門檻值的像素設為127，否則不變。
- [8] THRESH\_TOZERO: 大於門檻值的像素不變，否則設為0
- [9] THRESH\_TOZERO\_INV: 大於門檻值的像素設為0，否則不變
- [10] THRESH\_OTSU: 自適應決定門檻值，大於門檻值的像素設為255，否則不變。

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread("cat.jpg", 0)
5 ret, out1 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)
6 ret, out2 = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY_INV)
7 ret, out3 = cv2.threshold(img, 127, 255, cv2.THRESH_TRUNC)
8 ret, out4 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO)
9 ret, out5 = cv2.threshold(img, 127, 255, cv2.THRESH_TOZERO_INV)
10 ret, out6 = cv2.threshold(img, 127, 255, cv2.THRESH_OTSU)
11
12 cv2.imshow("THRESH_BINARY", out1)
13 cv2.imshow("THRESH_BINARY_INV", out2)
14 cv2.imshow("THRESH_TRUNC", out3)
15 cv2.imshow("THRESH_TOZERO", out4)
16 cv2.imshow("THRESH_TOZERO_INV", out5)
17 cv2.imshow("THRESH_OTSU", out6)
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```



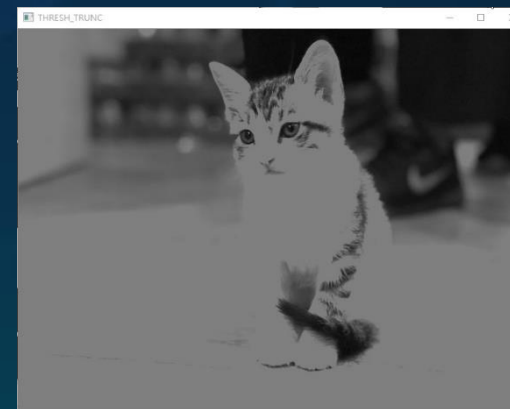
# 二值化-OpenCV



THRESH\_BINARY



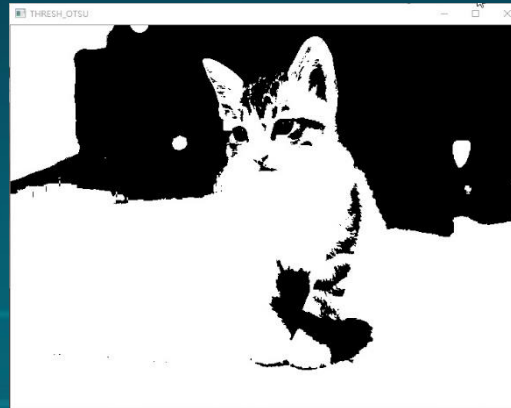
THRESH\_BINARY\_INV



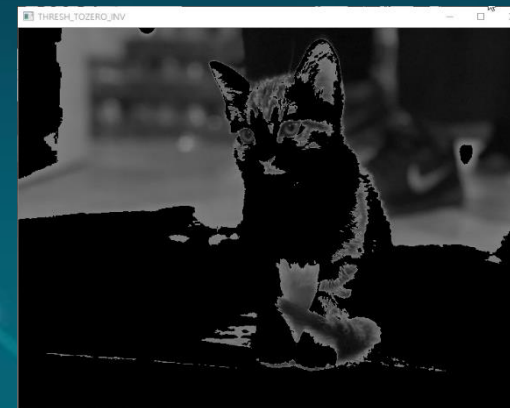
THRESH\_BINARY\_TRUNC



THRESH\_TOZERO



THRESH\_TOZERO\_INV



THRESH\_OTSU



# 影像增強(Image Enhancement)

- 清晰的影像，是指能夠清楚反映被攝景物的明亮程度和細微色彩差別的影像。
- 應用實例：當夜間拍攝路面號誌或車道線時，因色彩相似或亮度不足以致融入背景等情況下，影像就很難看的清楚。對於這種影像，如果把號誌或線條的色彩和亮度進行適當處理，使其與背景產生微秒的差別，就可以使所需特徵更加明確，以供後續辨識作業。
- 像這樣通過增強亮度或色彩等各種包含於影像中的資訊，如使影像顯得更美觀、易於判讀、突出某些部位等，而加強影像的某些特點，這種處理過程稱為影像增強。





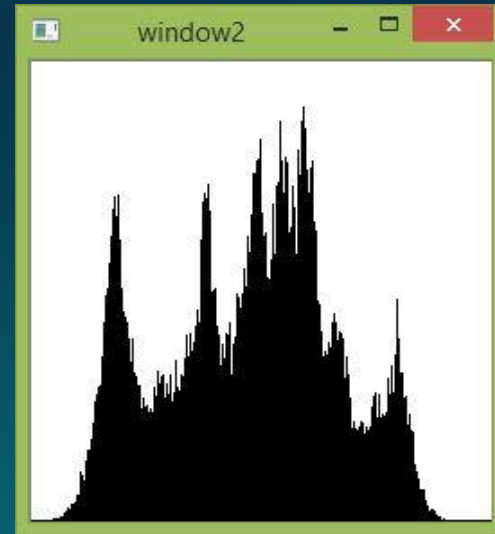
# 影像增強-範例





# 直方圖(Histogram)-簡介

- 直方圖是(Histogram)一個表現影像中像素分布的統計表，橫軸為影像中所有的像素值，假設是8位元影像，那其範圍為0到255，縱軸每個像素在影像中的個數。



- 直方圖是影像的一個重要特性，可以透過直方圖觀察影像中像素值的變化，看出影像是否太暗或過曝，又或者分布太過集中。





# 直方圖-OpenCV

OpenCV calcHist()函式：

- `hist = cv2.calcHist(<images>, <channels>, <mask>, <histSize>, <ranges>[, <hist>[, <accumulate>]])`
  - `images`：輸入影像，可以一個或多個影像，每張影像的尺寸和深度必須相同。
  - `channels`：直方圖通道。
  - `mask`：遮罩(可有可無)。
  - `hist`：輸出的直方圖。
  - `dims`：直方圖的維度，必須為正數。
  - `histSize`：直方圖橫軸數目。
  - `ranges`：直方圖的強度範圍，以8位元無負號的影像為例子:範圍為[0, 255]。
  - `uniform`：各維度取值是否一致。
  - `accumulate`：如果設定為true的話，在呼叫calcHist()這函式的時候，hist的內容不會被清掉。
  - 例: `hist = cv2.calcHist(gray_img, 0, None, 256, [0, 255])`





# 直方圖等化(Histogram Equalization)-簡介

- 我們可透過拉伸直方圖，使直方圖覆蓋所有強度範圍，這種方法的確能提高影像對比度，但是在多數情況，影像模糊不是因為過窄的強度範圍，而是某區間的像素強度比例過高，這時可以製作一個映射表，使得調整之後的影像，能平均使用所有的強度，進而增加影像的整體對比度。
- 要把灰階直條圖等化，只需要把原影像中像素少的部份進行壓縮，而將像素數較多的部份拉伸開來即可。





# 直方圖等化-流程

- 計算輸入影像的直方圖。
- 將直方圖歸一到所有bin的總合為255。
- 計算直方圖累計表(CDF)。
- 用直方圖累計表完成各強度的映射，所以假設強度30所累積的比例為20%，映射的強度即為 $255 \times 0.2$ ，由於我們直方圖歸一化到255，所以假設強度30所累積的值為20，映射的強度即為20。



# 直方圖等化-演算法

- 下面利用一個簡單的例子來說明平坦化的演算法。先考慮在灰階為0~7之處，各有如下圖所示數量的像素的情況。本例中為像素數平均值 $40 \div 8 = 5$ 。然後，從原影像灰階最高處開始，每5個像素為一組，順序分配給新的灰階級，本例中是從新的灰階7開始分配。

灰階級	7	6	5	4	3	2	1	0
原影像的各級像素數	0	4	9	11	5	7	4	0
解釋階級高處開始，每5個像素為一組，依順序分配給新的灰階級								
等化後的各級的像素數	5	5	5	5	5	5	5	5

- 而從灰階為5處的像素中，選擇出1個像素的方法有兩種：
  - 1) 隨機選擇。
  - 2) 從周圍平均灰階高的像素開始順序選擇
- 從演算法上考慮，(1)比(2)稍顯複雜，而(2)與(1)相比，具有雜訊數較少的特點。

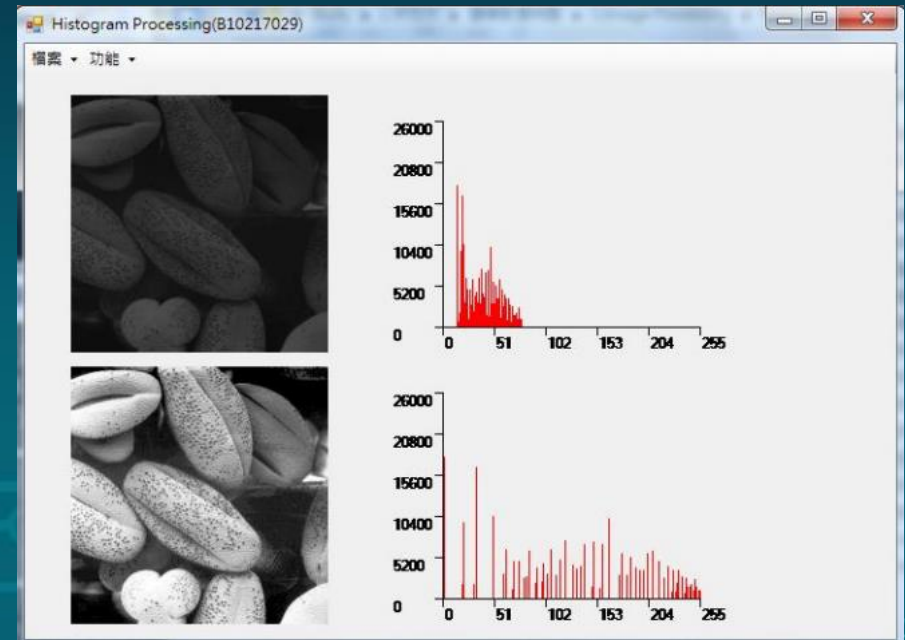




# 直方圖等化-OpenCV

## OpenCV直方圖等化函式

- `dst = cv2.equalizeHist(<src>[, <dst>])`
  - `src`：輸入影像，8位元單通道圖。
  - `dst`：輸出影像，和輸入影像尺寸、型態相同。
  - 例：`img_out = cv2.equalizeHist(img)`





# Otsu(自適應門檻值)-簡介

- Otsu演算法又稱為「大津演算法」，是一種自動的門檻值決定法則，常用在選擇影像二值化的門檻值上，當然Otsu並不是只能使用在影像二值化上，只要是選擇門檻值都能使用Otsu。
- Otsu 的這個方法在許多應用上時常被提起，主要是應用於影像的二值化處理，這個方法可以自動找出一個適當臨界值，讓不同的群組分開。



# Otsu(自適應門檻值)-流程

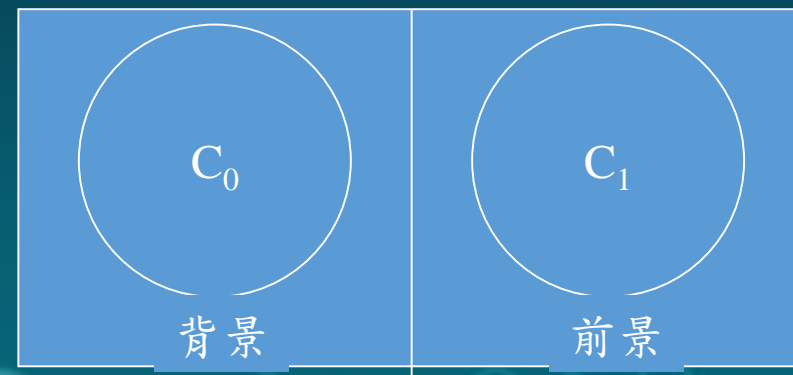
- 先計算影像的直方圖。
- 把直方圖強度大於門檻值的像素分成一組，把小於門檻值的像素分成另一組。
- 分別計算這兩組的組內變異數，並把兩個組內變異數相加。
- 將0-255依序當作門檻值來計算組內變異數和，總和值最小的就是結果門檻值。





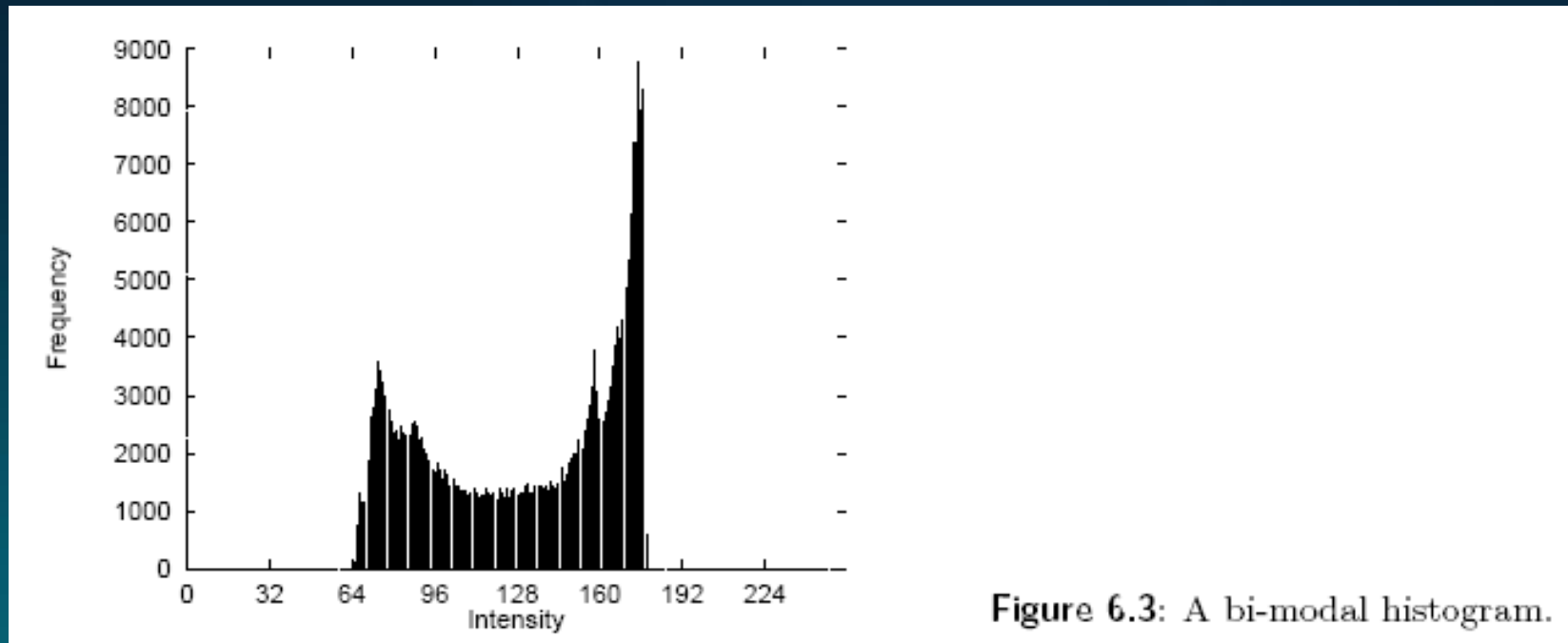
# Otsu(自適應門檻值)決定法(1-1)

- 底下將以決定一個門檻值為例，來帶出Otsu的想法。假若 $T^*$ 為最佳門檻值，我們利用 $T^*$ 把一影像分成 $C_0$ 及 $C_1$ 前後景二區，而該 $T^*$ 值的決定將會滿足下列兩項條件：
  1.  $C_0$ 及 $C_1$ 之間的「類別間變異數Between-class variance」為最大。
  2.  $C_0$ 內的變異數加上 $C_1$ 內的「類別內變異數Within-class variance」之和為最小。



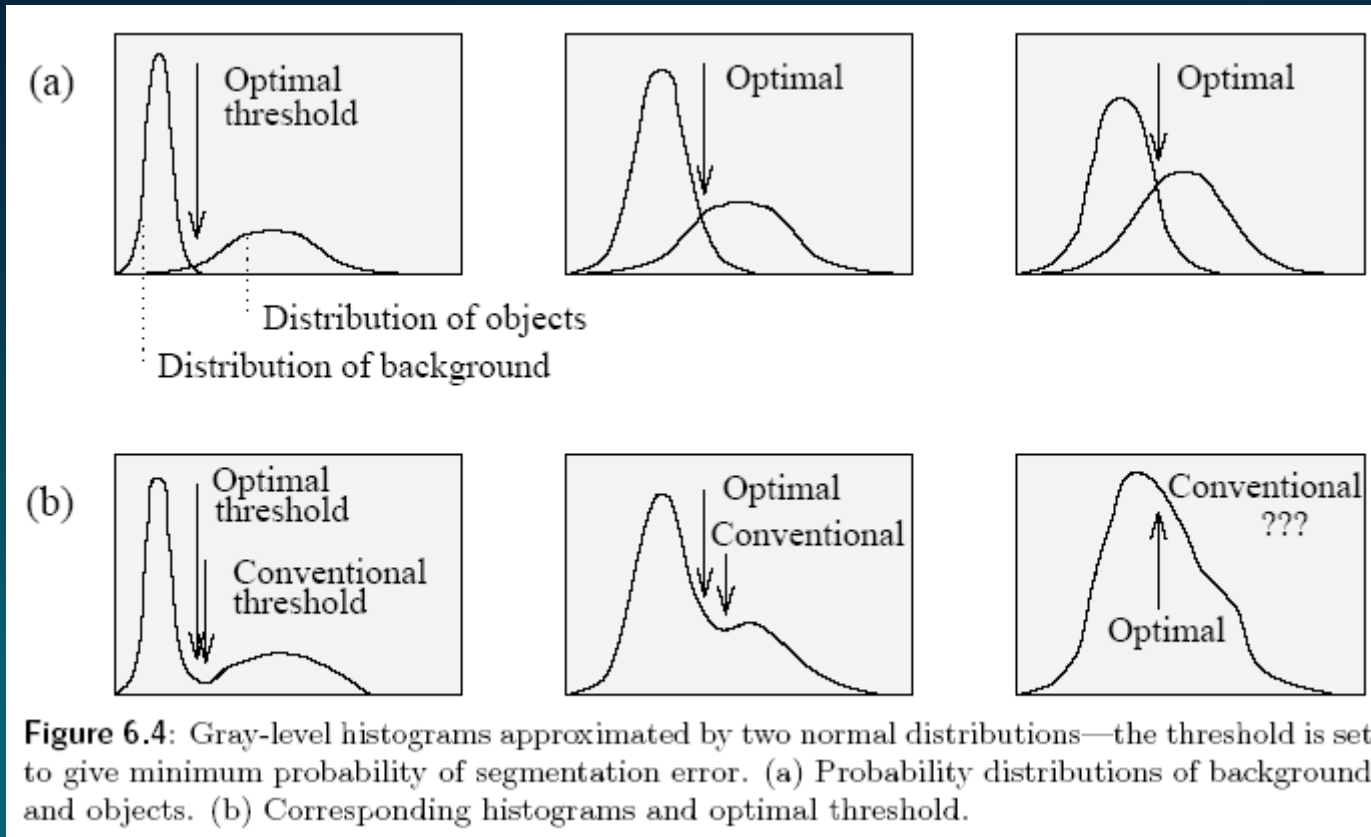


# Otsu(自適應門檻值)決定法(1-2)





# Otsu(自適應門檻值)決定法(1-3)







## Otsu(自適應門檻值)決定法(2)

- 令處理影像的大小為 $N$ ，而灰階值個數為 $I$ 。則灰階值為 $i$ 的出現機率可表示為：

$$P(i) = \frac{n_i}{N}$$

- $n_i$ 表示灰階值 $i$ 出現在影像中的次數，且 $i$ 的範圍介於 $0 \leq i \leq I - 1$ ，而根據機率原理又得知

$$\sum_{i=0}^{I-1} P(i) = 1$$



# Otsu(自適應門檻值)決定法(3)

- 假設 $C_0$ 及 $C_1$ 內的像素個數佔的比率(累進機率)分別為

$$\begin{aligned} w_0 &= \Pr(C_0) = \sum_{i=0}^{T^*} P(i) \\ w_1 &= \Pr(C_1) = \sum_{i=T^*+1}^{I-1} P(i) \end{aligned} \quad w_0 + w_1 = 1$$

- 接著可以求出 $C_0$ 及 $C_1$ 之期望值為

$$\mu_0 = \sum_{i=0}^{T^*} \frac{P(i) * i}{w_0} = \frac{\mu_*}{w_0}$$

$$\mu_1 = \sum_{i=T^*+1}^{I-1} \frac{P(i) * i}{w_1} = \frac{\mu_T - \mu_*}{1 - w_0}$$



# Otsu(自適應門檻值)決定法(4)

- 利用 $\mu_0$ 及 $\mu_1$ ，進一步算出 $C_0$ 及 $C_1$ 的變異數為

$$\sigma_0^2 = \sum_{i=0}^{T^*} (i - \mu_0)^2 \frac{P(i)}{w_0}$$

$$\sigma_1^2 = \sum_{i=T^*+1}^{I-1} (i - \mu_1)^2 \frac{P(i)}{w_1}$$

- 而 $C_0$ 及 $C_1$ 之內變異數和為

$$\sigma_W^2 = w_0 \sigma_0^2 + w_1 \sigma_1^2$$

- $C_0$ 及 $C_1$ 之間的類別間變異數亦可表示為

$$\sigma_B^2 = w_0 (\mu_0 - \mu_{T^*})^2 + w_1 (\mu_1 - \mu_{T^*})^2 = w_0 w_1 (\mu_0 - \mu_1)^2$$

- 此處 $\mu_{T^*}$ 為整個原始影像的平均值，可用下式求得

$$\mu_T = \sum_{i=0}^{I-1} \frac{n_i * i}{N} = \frac{1}{N} \sum_{i=0}^{I-1} n_i * i$$





## Otsu(自適應門檻值)決定法(5)

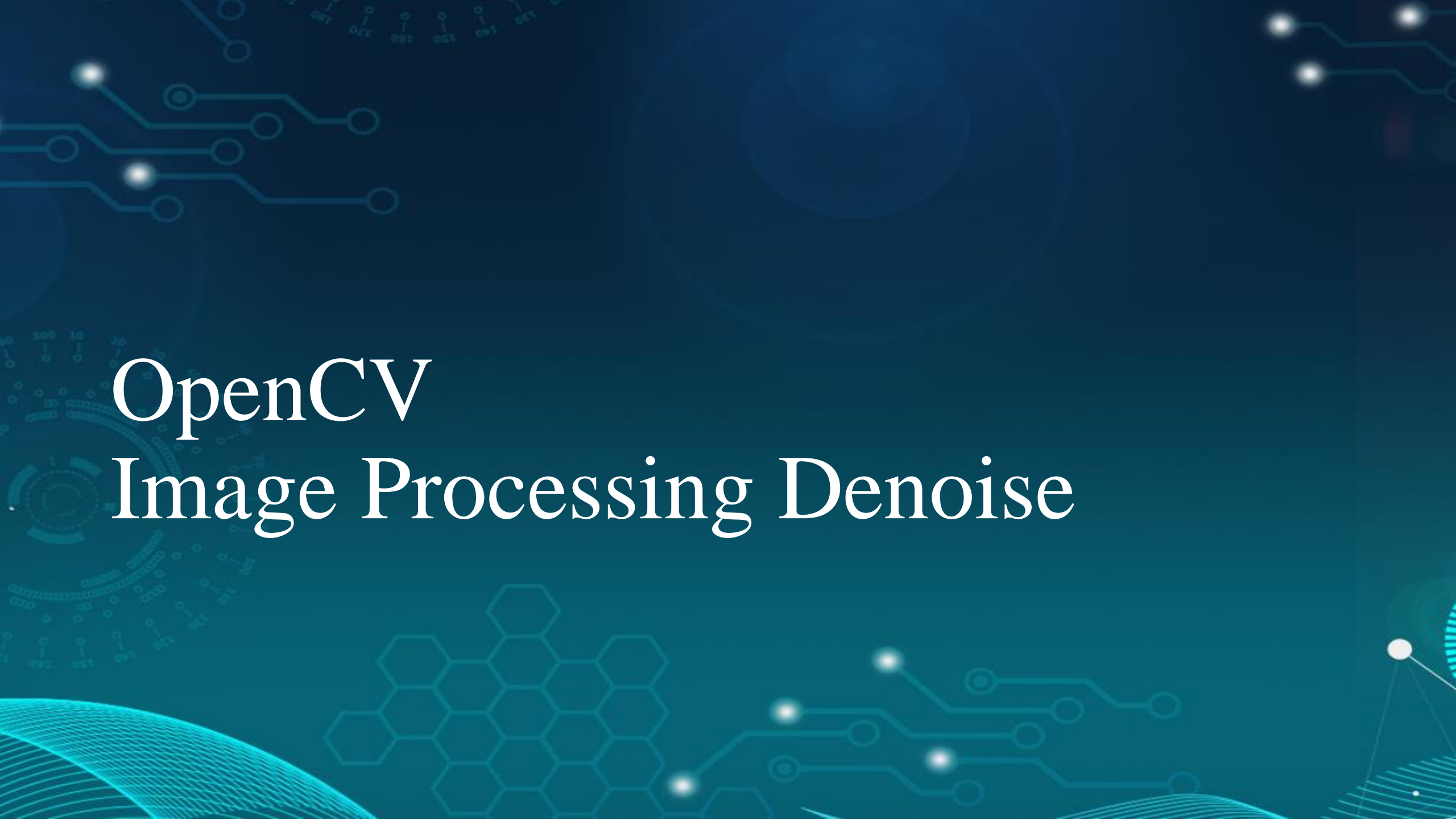
- 最後，我們可以驗證出 $\sigma_B^2$ 、 $\sigma_W^2$ 和 $\sigma_{T^*}^2$ 之間存在有這樣的關係，此處 $\sigma_{T^*}^2$ 為原始影像的變異數。
- 由於 $\sigma_{T^*}^2$ 為一定值， $C_0$ 和 $C_1$ 之間的變異數最大化問題等於 $C_0$ 和 $C_1$ 內的變異數和的最小化問題。那就考慮如何找到一個最佳化的 $T^*$ 來使得 $C_0$ 和 $C_1$ 之間的變異數 $\sigma_B^2$ 為最大就夠了。
- 我們使用的方法是在0至 $I - 1$ 之間，一個一個將灰階值代入 $\sigma_B^2$ 式子內，等全部 $I$ 個灰階值都代入完，再從可獲得最大的 $\sigma_B^2$ 類別間變異量值所對應的灰階值做為 $T^*$ 。這樣決定的 $T^*$ 就是將原始影像分割為 $C_0$ 和 $C_1$ 兩區的最佳門檻值。



# Otsu(自適應門檻值)OpenCV

一樣是用cv2.threshold()函式，使用方式也一樣，只是最後一個參數增加cv2.THRESH\_OTSU，目前otsu只能使用在8位元圖。

- src：輸入影像，只能輸入單通道。
- dst：輸出影像，尺寸大小、深度會和輸入影像相同。
- thresh：門檻值。
- maxval：二值化結果的最大值。
- type：二值化操作型態，共有THRESH\_BINARY、THRESH\_BINARY\_INV、THRESH\_TRUNC、THRESH\_TOZERO、THRESH\_TOZERO\_INV五種。
- type從上述五種結合CV\_THRESH\_OTSU，
- 範例：THRESH\_BINARY | CV\_THRESH\_OTSU



# OpenCV

## Image Processing Denoise





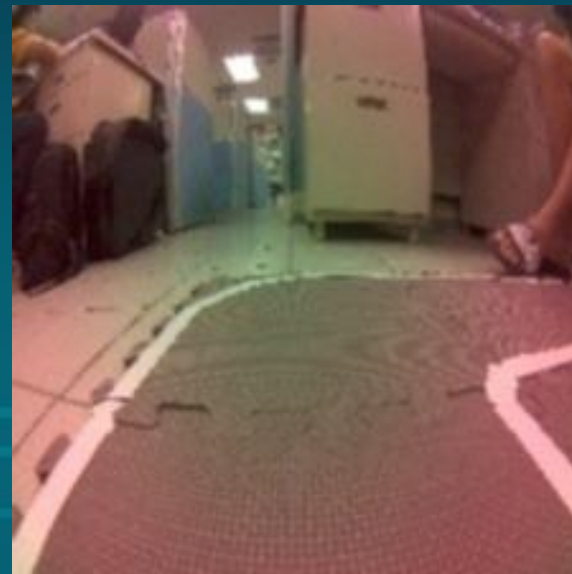
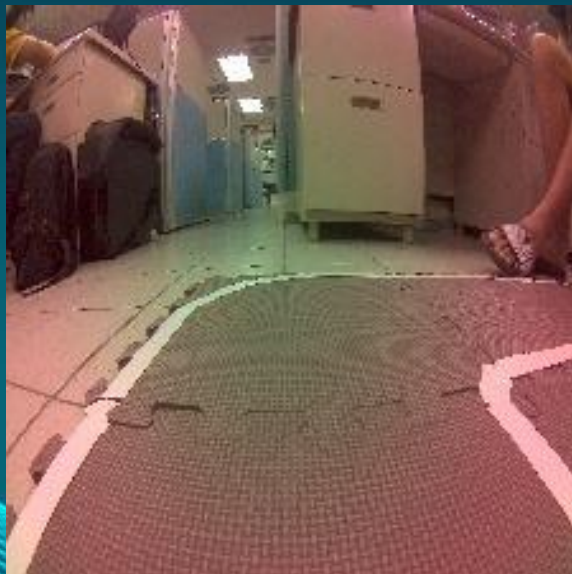
# 影像去雜訊(Image Denoising)

- 影像在傳輸的過程中，由於受到通道、劣質取樣系統、雜訊等其他的干擾影響，導致影像變得不清晰，因此我們需要對影像使用濾波去除雜訊。
- 雜訊產生的原因決定了雜訊與影像訊號的關係。而減少雜訊的方法可分為兩種：一種是在空間域做處理；另一種則是在頻率域上做處理。
- 在執行影像濾波時，需要以一定的細節模糊做為代價，因此要如何濾除影像的雜訊，又可以保持影像的細節是一個重要的課題。



# 影像去雜訊-以車道線偵測為例

- 在車道線偵測的過程中，因所使用的車上相機不一定能保有穩定的圖片訊號，容易造成影像中留有許多雜訊，以至於難以取出車道線的特徵。
- 為了讓車道線的特徵能更完整，我們可以先使用濾波去除影像中的雜訊。下圖以高斯濾波器作為示範。





# 濾波器常見種類

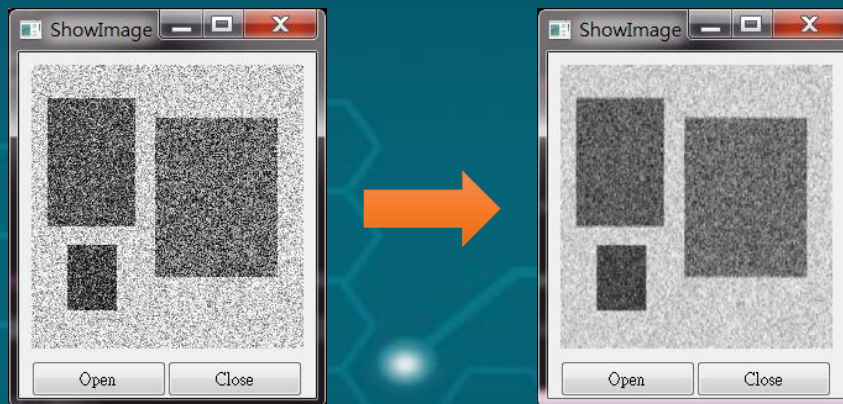
- 均值濾波器(Mean Filter)
- 中值濾波器(Median Filter)
- 高斯濾波器(Gaussian Filter)
- 雙邊濾波器(Bilateral Filter)





# 均值濾波(Mean Filter)

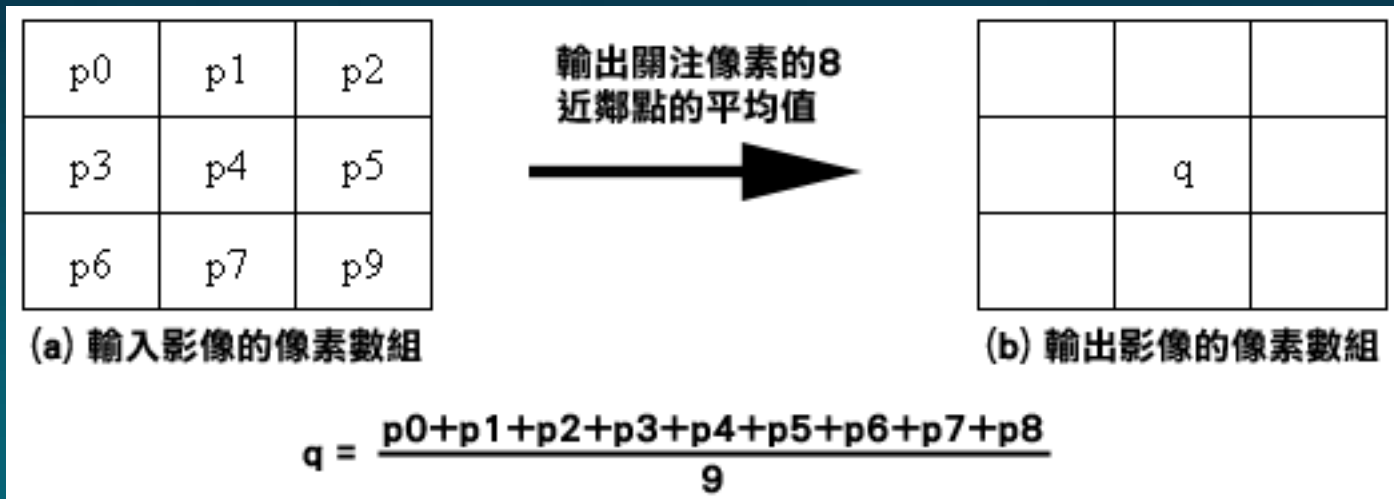
- 又稱為平滑線性濾波器(averaging filters)或低通濾波器(lowpass filters)。
- 濾波器遮罩(mask)，將鄰近的區域中的平均值(灰階)，取代區域中的每一個像素，這樣的程序產生在灰階上「銳利」變化降低的影像。隨機雜訊通常在灰階上含有銳利的變化，所以均值濾波器常常使用在減少雜訊。
- 但是邊緣也在灰階上含有銳利變化的特性，所以均值濾波器有模糊邊緣的缺點。





# 均值濾波(Mean Filter)

- 平滑法是最簡單的雜訊去除法，它採取把某像素的值置換為該像素周圍3x3個像素的濃度的平均值的方法，將整張影像予以平滑化。

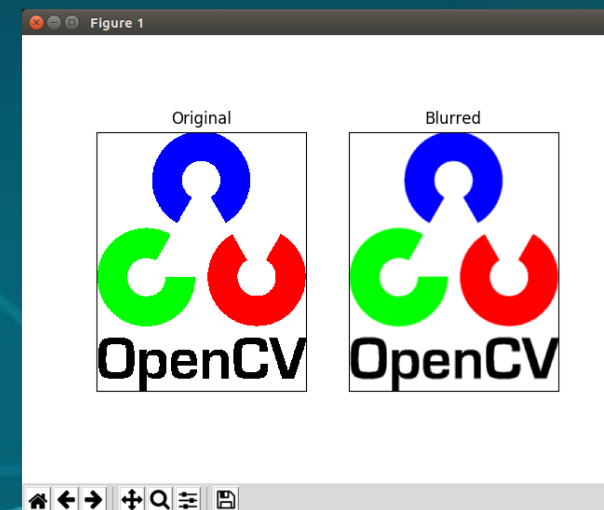




# 均值濾波-OpenCV

OpenCV blur()函式：

- `dst = cv2.blur(<src>, <ksize>[, <dst>[, <anchor>[, <borderType>]])`
  - `src`：輸入影像。
  - `dst`：輸出影像會和輸入影像尺寸、型態相同。
  - `ksize`：kernel大小，可分別指定長和寬。
  - `anchor`：錨點，預設為`Point(-1,-1)`，代表錨點在kernel的中心
  - `borderType`：邊界類型，邊界模式用來推斷影像外的像素
  - 例：`blur = cv2.blur(img, (5, 5))`

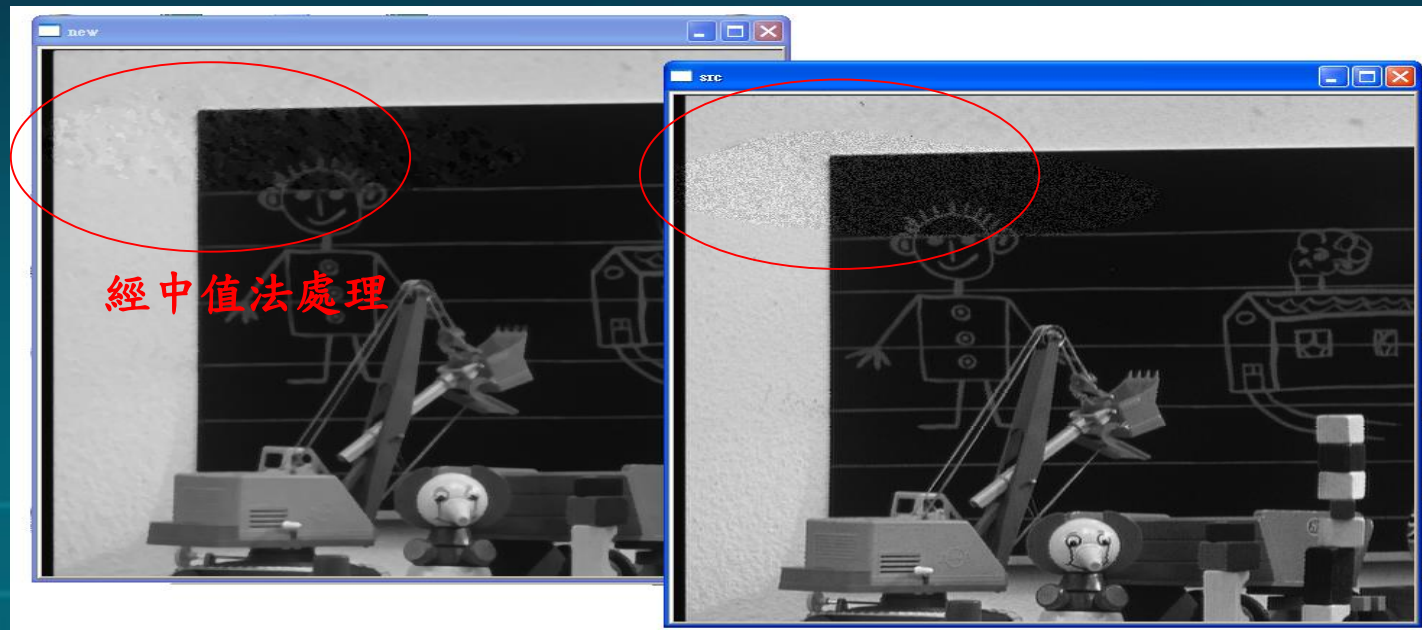






# 中值濾波(Median Filter)

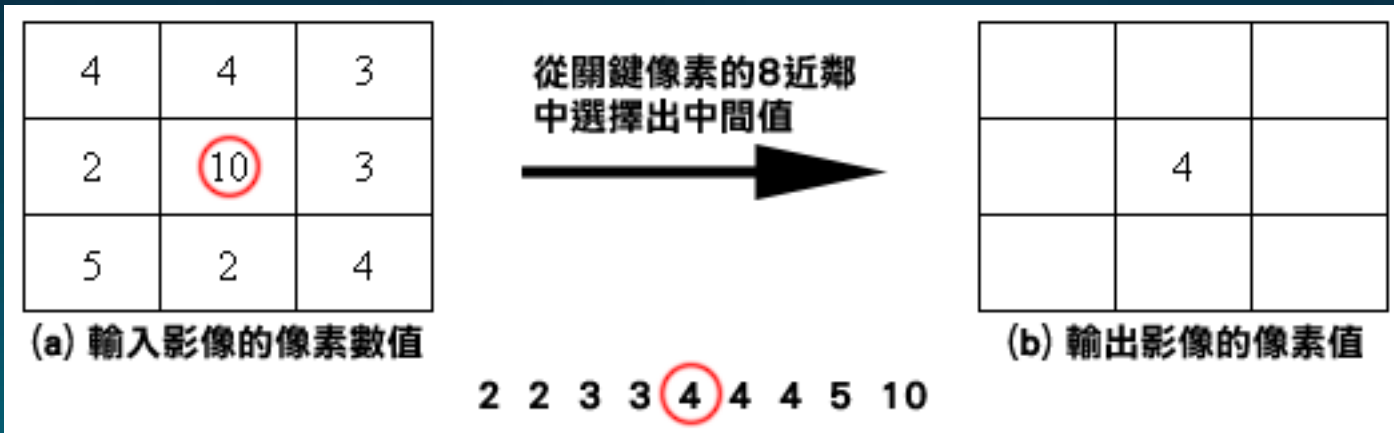
- 將像素的值用該像素近鄰灰階的中間值來取代。
- 在脈衝雜訊(impulse noise) (又稱椒鹽雜訊(salt and pepper noise))出現時，中值濾波器能有效地去除雜訊，而且與均值濾波器相比有較輕微的模糊化。






# 中值濾波(Median Filter)

- 將某像素的值與周圍3x3像素做大小順序排列，隨後取出中間值並取代原有像素。

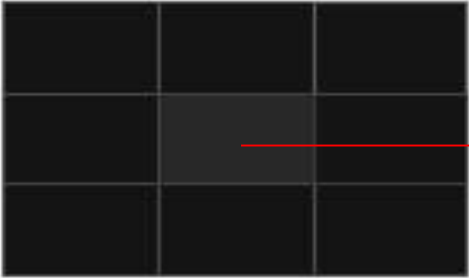




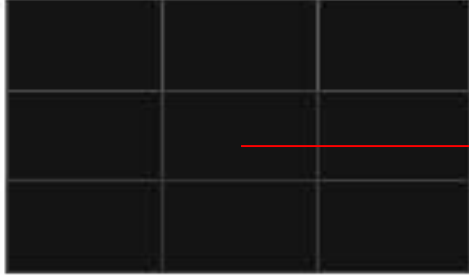
# 中值濾波(Median Filter)

20	20	20	
20	200	20	
20	20	20	

20	20	20	
20	40	20	
20	20	20	

20	20	20	
20	20	20	
20	20	20	

原始雜訊影像

使用平滑法處理之結果

使用中值法處理之結果





# 中值濾波 - OpenCV

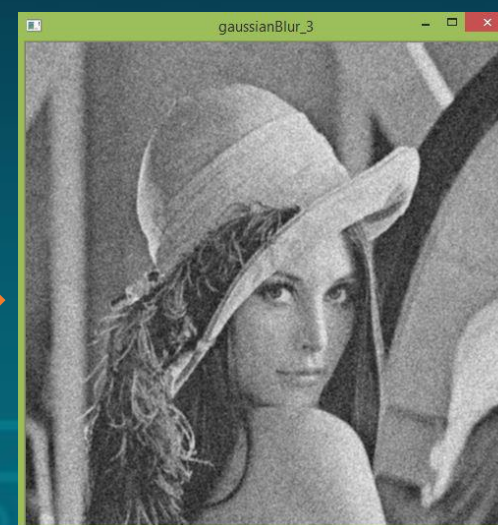
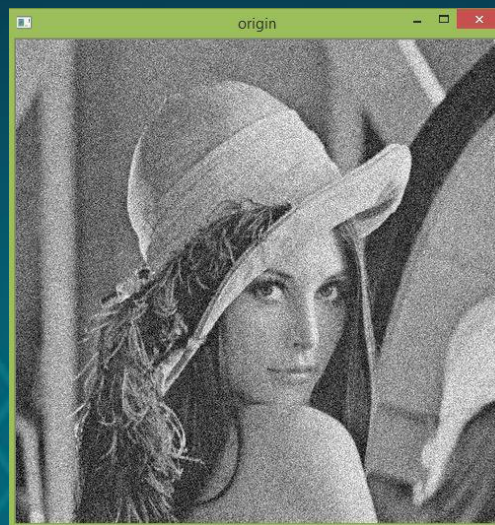
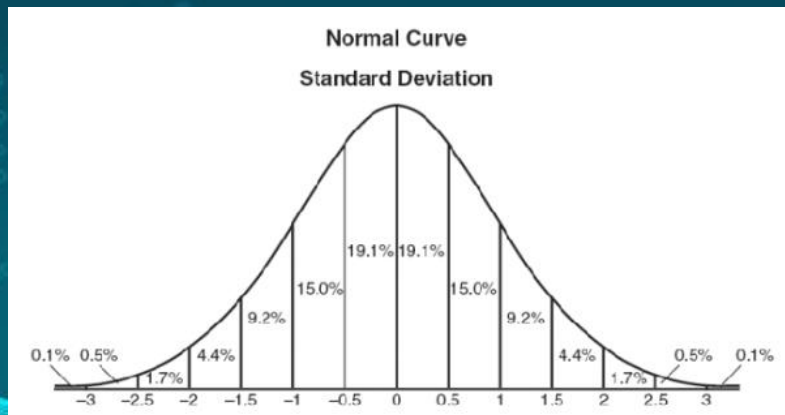
OpenCV medianBlur() 函式：

- `dst = cv2.medianBlur(<src>, <ksize>[, <dst>])`
  - `src`：輸入影像。
  - `dst`：輸出影像會和輸入影像尺寸、型態相同。
  - `ksize`：kernel 大小，必須為大於1的正奇數(例如給定7，就會自動使用7x7的kernel，因為kernel必須是正方形)。
  - 例：`blur3 = cv2.medianBlur(img, 15)`



# 高斯濾波(Gaussian Filter)

- 高斯濾波的每個像素的值都是周圍相鄰像素值的加權平均。原始像素為中心，有最大的高斯分布值，所以有最大的權重，相鄰像素隨著距離原始像素越來越遠，其權重也越來越小如下圖。這樣進行模糊處理，跟其它的濾波器相比能更有效地保留邊緣。





# 高斯濾波(Gaussian Filter)

- 以下為高斯函數：

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

- 其中 $\mu$ 是 $x$ 的均值， $\sigma$ 是 $x$ 的標準差。
- 每次計算時當前像素為原點，因此公式簡化為：

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





# 高斯濾波(Gaussian Filter)

- 一般影像都是二維，所以使用二維分布，以下為二維高斯函式：

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

- 計算權重值，假設中心為(0,0)：

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,-1)	(0,-1)	(1,-1)



# 高斯濾波(Gaussian Filter)

- 計算權重值，假設 $\sigma$ 為2，經由高斯運算，其權重矩陣如下：

0.0309874	0.0351134	0.0309874
0.0351134	0.0397887	0.0351134
0.0309874	0.0351134	0.0309874

- 然後求其加權平均，這9個點權重總和等於 0.3041919，因此要分別除0.3041919讓總和等於1：

0.1018679	0.1154317	0.1018679
0.1154317	0.1309013	0.1154317
0.1018679	0.1154317	0.1018679



# 高斯濾波(Gaussian Filter)

- 有了權重就能計算高斯模糊，假設有以下9個點：

100	95	110
120	90	100
110	115	120

- 將像素點乘上權重：

$100 \times 0.1018679$	$95 \times 0.1154317$	$110 \times 0.1018679$
$120 \times 0.1154317$	$90 \times 0.1309013$	$100 \times 0.1154317$
$110 \times 0.1018679$	$115 \times 0.1154317$	$120 \times 0.1018679$





# 高斯濾波(Gaussian Filter)

- 得到高斯模糊後的值：

10.18679	10.9660115	11.205469
13.851804	11.781117	11.54317
0.1018679	13.2746455	12.224148

- 以下為3x3常用模板 $\sigma$ 為0.8：

1/16	2/16	1/16
2/16	4/16	2/16
1/16	2/16	1/16



# 高斯濾波-OpenCV

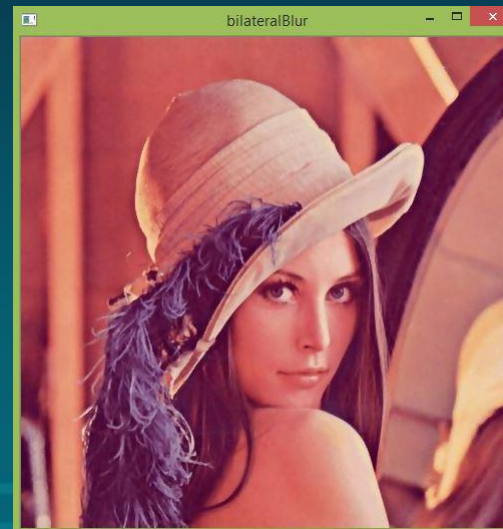
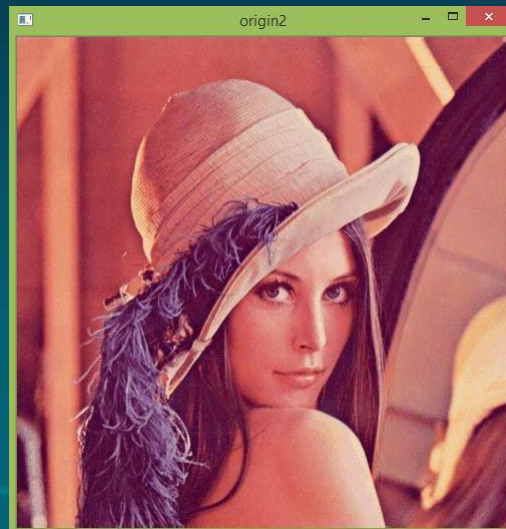
OpenCV GaussianBlur() 函式：

- `dst = cv2.GaussianBlur(<src>, <ksize>, <sigmaX>[, <dst>[, <sigmaY>[, <borderType>]])`
  - `src`：輸入影像。
  - `dst`：輸出影像會和輸入影像尺寸、型態相同。
  - `ksize`：kernel大小，長寬可以不同，但是都必須為正的奇數。
  - `sigmaX`：x方向的標準差。
  - `sigmaY`：y方向的標準差。
  - 例：`blur2 = cv2.GaussianBlur(img, (15,15), 0)`



# 雙邊濾波(Bilateral Filter)

- 和均值濾波及中值濾波有所不同，雙邊濾波器除了使用像素之間幾何上的靠近程度之外，還多考慮像素之間光度及色彩上的差異，使得雙邊濾波器能夠有效的將影像上的雜訊濾除，同時保存影像上的邊緣資訊。







# 雙邊濾波(Bilateral Filter)

- 雙邊濾波包含了兩個函式，一個是採用空間幾何(和高斯濾波相似)，另一個是像素差值(光度/色彩差異)。
- 以下為雙邊濾波公式：

$$I_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I_p - I_q|) I_q$$

- $p$ 為目標像素。
- $q$ 為目標像素之周圍像素。
- $I_p$ 為目標像素之色彩。
- $I_q$ 為目標像素之周圍像素之色彩。
- $S$ 為目標像素之權重計算範圍。
- $G_{\sigma_s}$ 為高斯濾波，加權根據距離。
- $G_{\sigma_r}$ 為高斯濾波，加權根據像素色差。
- $W_p$ 為 $G_{\sigma_s}$ 和 $G_{\sigma_r}$ 相乘。



# 雙邊濾波(Bilateral Filter)

- 假設矩陣為 $8 \times 8$ ，雙邊取值為 $5 \times 5$ ，距離 $\sigma$ 為10，像素 $\sigma$ 為30，我們取左上 $5 \times 5$ 先進行計算。

170	160	170	165	160	145	145	160
160	90	90	85	165	145	145	160
170	95	70	95	170	160	140	140
165	90	85	85	170	160	135	160
170	170	160	160	165	160	160	160
140	160	160	160	140	140	135	135
160	135	145	140	160	160	135	135
160	135	135	140	140	140	140	160



# 雙邊濾波(Bilateral Filter)

- 計算距離權重值跟高斯相同(雙邊通常模板大小大於為5x5以上，在這裡以5x5大小作為範例)：

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

- 計算權重值，假設中心為(0,0)：

(-2, 2)	(-1, 2)	(0, 2)	(1, 2)	(2, 2)
(-2, 1)	(-1, 1)	(0, 1)	(1, 1)	(2, 1)
(-2, 0)	(-1, 0)	(0, 0)	(1, 0)	(2, 0)
(-2, -1)	(-1, -1)	(0, -1)	(1, -1)	(2, -1)
(-2, -2)	(-1, -2)	(0, -2)	(1, -2)	(2, -2)





# 雙邊濾波(Bilateral Filter)

- 計算權重值，假設 $\sigma$ 為3，經由高斯運算，其權重矩陣如下：

0.011339	0.013395	0.014160	0.013395	0.011339
0.013395	0.015824	0.016728	0.015824	0.013395
0.014160	0.016728	0.017684	0.016728	0.014160
0.013395	0.015824	0.016728	0.015824	0.013395
0.011339	0.013395	0.014160	0.013395	0.011339



# 雙邊濾波(Bilateral Filter)

- 計算像素差值權重(一樣使用高斯函式，這邊使用一維)：

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

- 計算權重值，假設中心為(3, 3)：

170	160	170	165	160
160	90	90	85	165
170	95	70	95	170
165	90	85	85	170
170	170	160	160	165



# 雙邊濾波(Bilateral Filter)

- 與(3, 3)之像素差值為：

100	90	100	95	90
90	20	20	15	95
100	25	0	25	100
95	20	15	15	100
100	100	90	90	95

- 計算權重值，假設 $\sigma$ 為30，經由高斯運算，其權重矩陣如下：

0.000051	0.000148	0.000051	0.000088	0.000148
0.000148	0.010648	0.010648	0.011736	0.000088
0.000051	0.009397	0.013298	0.009397	0.000051
0.000088	0.010648	0.011736	0.011736	0.000051
0.000051	0.000051	0.000148	0.000148	0.000088





# 雙邊濾波(Bilateral Filter)

- 將距離權重值乘上像素權重值，得到計算雙邊濾波的權重。

5.78289E-07	1.98246E-06	7.2216E-07	1.17876E-06	1.67817E-06
1.98246E-06	0.000168494	0.00017812	0.00018571	1.17876E-06
7.2216E-07	0.000157193	0.000235162	0.000157193	7.2216E-07
1.17876E-06	0.000168494	0.00019632	0.00018571	6.83145E-07
5.78289E-07	6.83145E-07	2.09568E-06	1.98246E-06	9.97832E-07

- 分別將權重乘上，該位置之像素。

9.83091E-05	0.000317194	0.000122767	0.000194495	0.000268508
0.000317194	0.015164456	0.016030777	0.015785389	0.000194495
0.000122767	0.014933337	0.016461328	0.014933337	0.000122767
0.000194495	0.015164456	0.016687184	0.015785389	0.000116135
9.83091E-05	0.000116135	0.000335309	0.000317194	0.000164642



# 雙邊濾波(Bilateral Filter)

- 將乘上權重之像素總和除以權重之總和，得到最後計算之結果。  
乘上權重之像素總和為0.144046367，權重之總和為0.001651341，  
相除後的結果值為87.22993746。
- 得到新值如右圖(綠色)

170	160	170	165	160	145	145	160
160	90	90	85	165	145	145	160
170	95	87	95	170	160	140	140
165	90	85	85	170	160	135	160
170	170	160	160	165	160	160	160
140	160	160	160	140	140	135	135
160	135	145	140	160	160	135	135
160	135	135	140	140	140	140	160



# 雙邊濾波(Bilateral Filter)

- 之後再往下一個點繼續計算。

170	160	170	165	160	145	145	160
160	90	90	85	165	145	145	160
170	95	70	95	170	160	140	140
165	90	85	85	170	160	135	160
170	170	160	160	165	160	160	160
140	160	160	160	140	140	135	135
160	135	145	140	160	160	135	135
160	135	135	140	140	140	140	160





# 雙邊濾波-OpenCV

OpenCV bilateralFilter() 函式：

- `dst = cv.bilateralFilter(<src>, <d>, <sigmaColor>, <sigmaSpace>[, <dst>[, <borderType>]])`
  - `src`：輸入影像。
  - `dst`：輸出影像會和輸入影像尺寸、型態相同。
  - `d`：過程中各像素會使用到的鄰域直徑大小，5以上
  - `sigmaColor`：該參數的較大值意味著像素鄰域內的更多顏色將被混合在一起，會得到較大的半等色區域10以上150以下。
  - `sigmaSpace`：坐標空間中的過濾器sigma。參數越大意味著只要其顏色足夠近，更遠的像素就會相互影響。
  - `borderType`：邊界類型，邊界模式用來推斷影像外的像素
  - 例：`blur4 = cv2.bilateralFilter(img, 19, 75, 75)`



# 常見濾波器-OpenCV

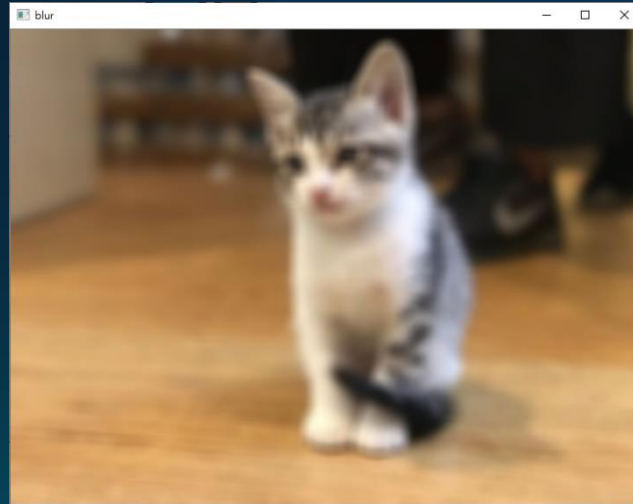
- 範例程式：
  - [4]讀入圖片
  - [6]以15x15大小的kernel進行均值濾波。
  - [7]以15x15大小的kernel進行高斯濾波。
  - [8]以15x15大小的kernel進行中值濾波。
  - [9]進行雙邊濾波。

```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('cat.jpg')
5
6 blur1 = cv2.blur(img, (15,15))
7 blur2 = cv2.GaussianBlur(img, (15,15), 0)
8 blur3 = cv2.medianBlur(img, 15)
9 blur4 = cv2.bilateralFilter(img, 19, 75, 75)
10
11 cv2.imshow("blur", blur1)
12 cv2.imshow("gaussianBlur", blur2)
13 cv2.imshow("medianBlur", blur3)
14 cv2.imshow("bilateralFilter", blur4)
15 cv2.waitKey(0)
16 cv2.destroyAllWindows()
```

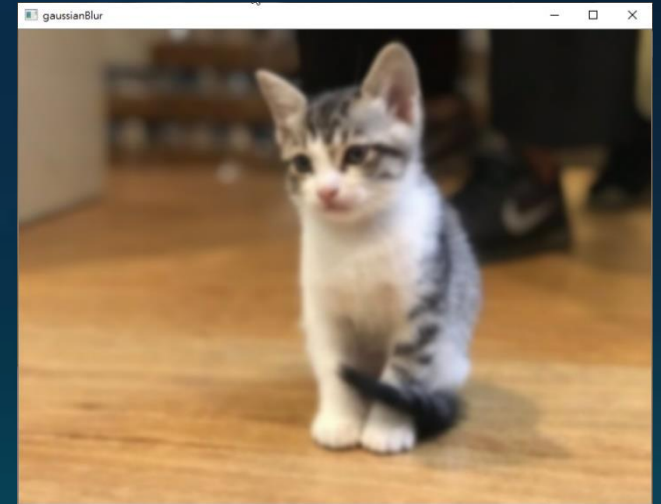




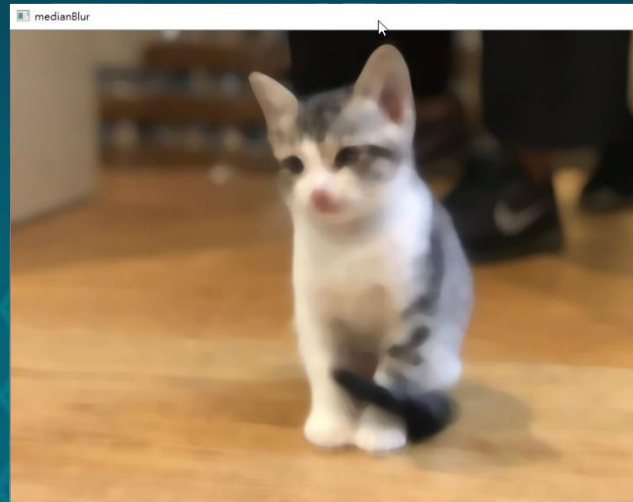
# 常見濾波器-OpenCV



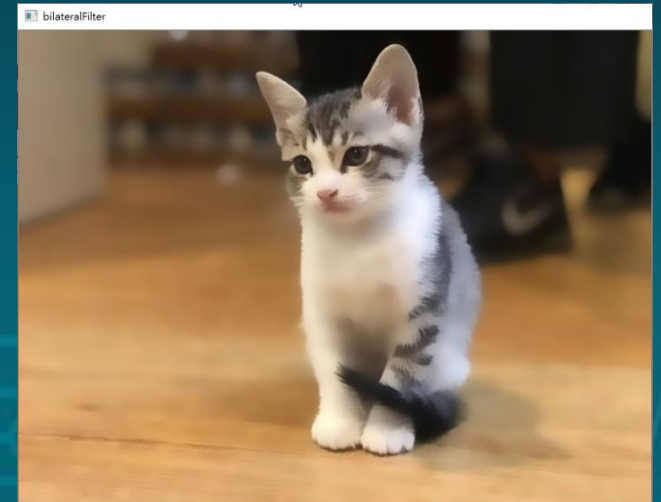
Blur(均值濾波)



GaussianBlur(高斯濾波)



MedianBlur(中值濾波)



BilateralFilter(雙邊濾波)





# 應用型態學演算法進行影像雜訊濾除

## 二值影像的雜訊去除

- 二值影像的雜訊，稱為椒鹽雜訊，它來自英語 salt and pepper noise 一詞的翻譯。當然這種雜訊也能用中值濾波法將其除去，另外，利用它的二值性，有稱為膨脹、收縮的處理方法。
- 膨脹(Dilation)是指某像素 $p$ 的近鄰中，若有一個為 1，則將 $p$ 置為 1。
- 收縮(Erosion)是指某像素 $p$ 的近鄰中，若有一個為 0，就將 $p$ 置為 0。
- Closing(閉運算)：膨脹→收縮(除去黑色雜訊，白色雜訊依然殘留)
- Opening(開運算)：收縮→膨脹(除去白色雜訊，黑色雜訊依然殘留)



# 膨脹(Dilation)

- 膨脹法表示位於某個點時是否有偵測到物件(以A當作kernel)，以下為其公式，假設A為3x3矩陣B為9x9矩陣。

$$A \oplus B = \{x | B_x \cap A \neq \emptyset\}.$$

- 綠色及紅色為原影像



1	1	1
1	1	1
1	1	1

A

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

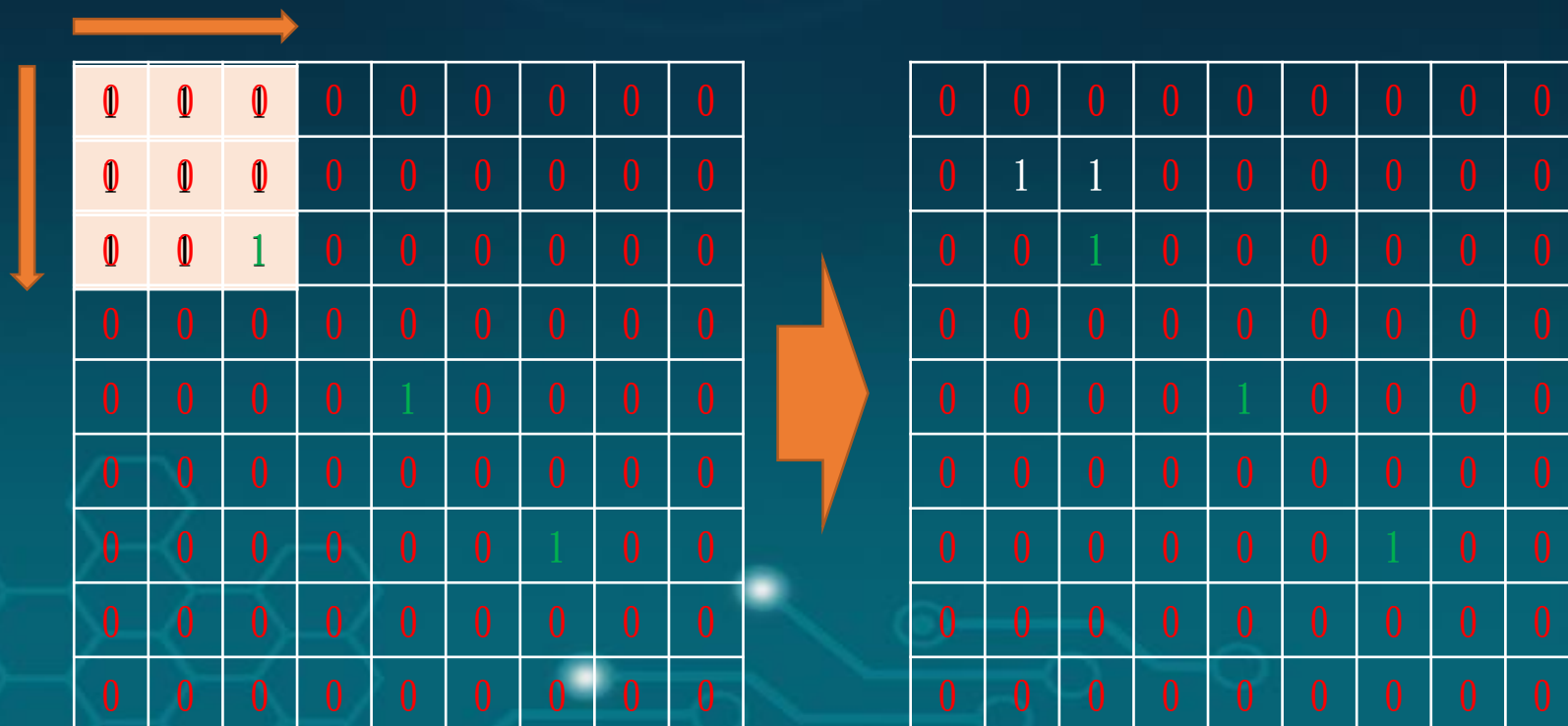
B







- 綠色及紅色為原影像。
- 黑色為走訪後。
- 接著繼續走訪。







- 最終走訪結果。
- 綠色及紅色為原影像。
- 黑色為走訪後。

The diagram illustrates a transformation of a 9x9 grid. On the left, the initial state is a sparse grid with most cells containing red '0's. A few cells contain green '1's: at row 3, column 3; row 5, column 5; and the bottom-right 3x3 block (rows 7-9, columns 7-9). Orange arrows indicate a transformation from left to right. On the right, the transformed state is a dense grid where many more cells contain green '1's, particularly in the top-right and bottom-right regions, while the bottom-right 3x3 block remains orange.





# 膨脹-OpenCV

OpenCV dilate() 函式：

- `dst = cv2.dilate(<src>, <kernel>[, <dst>[, <anchor>[, <iterations>[, <borderType>[, <borderValue>]]]]])`
  - `src`：輸入影像。
  - `dst`：輸出影像，和輸入影像尺寸、型態相同。
  - `kernel`：結構元素，預設為3×3的矩陣，越大膨脹效果越明顯。
  - `anchor`：原點位置，預設為結構元素的中央。
  - `iterations`：執行次數，執行越多次膨脹效果越明顯。
  - 例：`out = cv2.dilate(img, np.ones((5, 5)), iterations=3)`

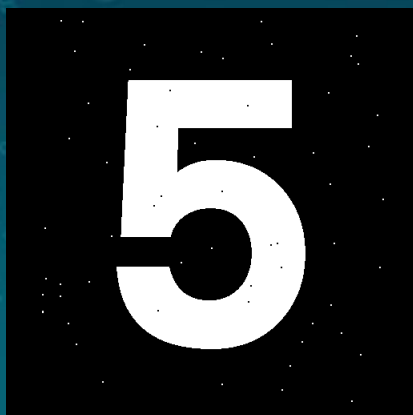


# 侵蝕(Erosion)

- 侵蝕法表示位於某個點時是否有偵測到全部物件(kernel A)，以下為其公式，假設A為3x3矩陣B為9x9矩陣。

$$A \ominus B = \{x | B_x \subseteq A\}$$

- 綠色及紅色為原影像



1	1	1
1	1	1
1	1	1

A

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

B











- 完全走訪後之結果。
- 綠色及紅色為原影像。
- 黑色為走訪後。

The diagram illustrates a transformation of a 9x9 grid. The left grid shows a sparse matrix with 1s in the first 6 rows and columns 2-6, and 0s elsewhere. The right grid shows the resulting dense matrix after a transformation, with 1s in the first 6 rows and columns 2-6, and 0s elsewhere. A large orange arrow points from the left grid to the right grid.





# 侵蝕 - OpenCV

OpenCV erode()函式:

- `dst = cv2.erode(<src>, <kernel>[, <dst>[, <anchor>[, <iterations>[, <borderType>[, <borderValue>]]]]])`
  - `src` : 輸入影像。
  - `dst` : 輸出影像，和輸入影像寸、型態相同。
  - `kernel` : 結構元素，預設為3x3的矩陣，越大侵蝕效果越明顯。
  - `anchor` : 原點位置，預設為結構元素的中央。
  - `iterations` : 執行次數，預設為1次，執行越多次侵蝕效果越明顯。
  - 例：`out = cv2.erode(img, np.ones((5, 5)), iterations=3)`



# 形態學(Morphology)

- 主要用於二值化後的影像，根據使用者的目的，用來凸顯影像的形狀特徵，像邊界和連通區域等，同時像細化、像素化、修剪毛刺等技術也常用於影像的預處理和後處理，形態學操作的結果除了影像本身，也和結構元素的形狀有關，結構元素和空間域操作的濾波概念類似。
- 在機器學習與影像處理的過程中常被用來前處理。



# 形態學(Morphology)

有以下幾個種類其公式算法如下：

- OPEN

- 公式： $A \circ B = (A \ominus B) \oplus B$
- $\text{dst} = \text{open}(\text{src}, \text{element}) = \text{dilate}(\text{erode}(\text{src}, \text{element}))$
- 去除小雜訊

- CLOSE

- 公式： $A \bullet B = (A \oplus B) \ominus B$
- $\text{dst} = \text{close}(\text{src}, \text{element}) = \text{erode}(\text{dilate}(\text{src}, \text{element}))$
- 去除小洞

- GRADIENT

- 公式： $A \oplus B - A \ominus B$
- $\text{dst} = \text{morph}(\text{src}, \text{element}) = \text{dilate}(\text{src}, \text{element}) - \text{erode}(\text{src}, \text{element})$
- 找輪廓





# 形態學(Morphology)

- TOPHAT

- 公式： $T_w(f) = f - f \circ b,$
- $\text{dst} = \text{tophat}(\text{src}, \text{element}) = \text{src} - \text{open}(\text{src}, \text{element})$
- 輸入影像及型態學Open之間的差異。
- TOPHAT被用於各種影像處理，如特徵提取，背景均衡，影像增強等。

- BLACKHAT

- 公式： $T_b(f) = f \bullet b - f,$
- $\text{dst} = \text{blackhat}(\text{src}, \text{element}) = \text{close}(\text{src}, \text{element}) - \text{src}$
- 型態學Close及輸入影像之間的差異。



# 形態學-OpenCV

OpenCV morphologyEx()函式：

- `dst = cv2.morphologyEx(src, op, kernel[, dst[, anchor[, iterations[, borderType[, borderValue]]]])` )
  - `src`：輸入影像。
  - `op`：操作種類，決定要進行何種型態學操作(open、close等等)。
  - `kernel`：結構元素。
  - `dst`：輸出影像，和輸入影像尺寸、型態相同。
  - `anchor`：原點位置，預設為結構元素的中央。
  - `iterations`：執行次數，預設為1次。
  - 例：見下頁



# 形態學-OpenCV

操作種類如下：

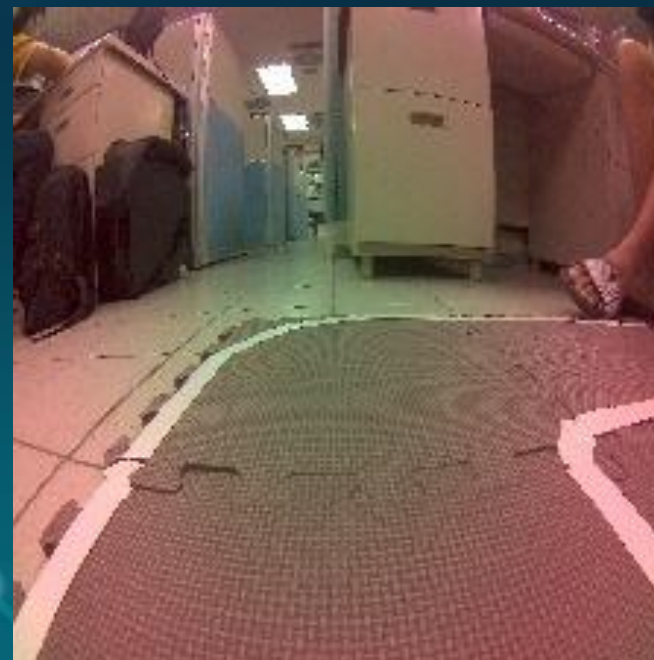
- Opening
  - 例：`opening = cv2.morphologyEx(img, cv2.MORPH_OPEN, np.ones((5, 5)))`
- Closing
  - 例：`closing = cv2.morphologyEx(img, cv2.MORPH_CLOSE, np.ones((5, 5)))`
- Gradient
  - 例：`gradient = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, np.ones((5, 5)))`
- Top Hat
  - 例：`tophat=cv2.morphologyEx(img, cv2.MORPH_TOPHAT, np.ones((5, 5)))`
- Black Hat
  - 例：`blackhat=cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, np.ones((5, 5)))`





# 形態學-以車道線擷取為例

- 在接下來的範例中，我們使用各種形態學的演算法來示範在車道線的擷取上這些演算法能提供什麼樣的幫助。
- 首先將原圖進行轉灰階與二值化的步驟，將圖片較亮的部分過濾出來，再分別使用不同演算法展示結果。





# 形態學-OpenCV

## • 範例程式：

- [5]決定一個5x5的kernel。
- [7, 13]使用5x5的kernel進行Opening運算，迭代3次。
- [8, 14]使用5x5的kernel進行Closing運算，迭代3次。
- [9, 15]使用5x5的kernel進行Gradient運算，迭代3次。
- [10, 16]使用5x5的kernel進行Top Hat運算，迭代3次。
- [11, 17]使用5x5的kernel進行Black Hat運算，迭代3次。

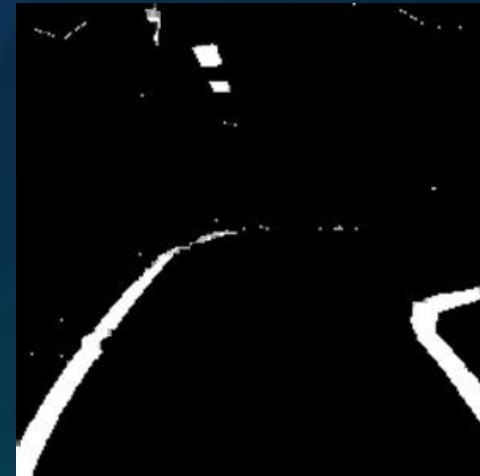
```
1 import cv2
2 import numpy as np
3
4 img = cv2.imread('road.png',0)
5 kernel = np.ones((5,5), np.uint8)
6
7 out3 = cv2.morphologyEx(img, cv2.MORPH_OPEN, kernel, iterations=3)
8 out4 = cv2.morphologyEx(img, cv2.MORPH_CLOSE, kernel, iterations=3)
9 out5 = cv2.morphologyEx(img, cv2.MORPH_GRADIENT, kernel, iterations=3)
10 out6 = cv2.morphologyEx(img, cv2.MORPH_TOPHAT, kernel, iterations=3)
11 out7 = cv2.morphologyEx(img, cv2.MORPH_BLACKHAT, kernel, iterations=3)
12
13 cv2.imshow("open", out3)
14 cv2.imshow("close", out4)
15 cv2.imshow("gradient", out5)
16 cv2.imshow("tophat", out6)
17 cv2.imshow("blackhat", out7)
18 cv2.waitKey(0)
19 cv2.destroyAllWindows()
```



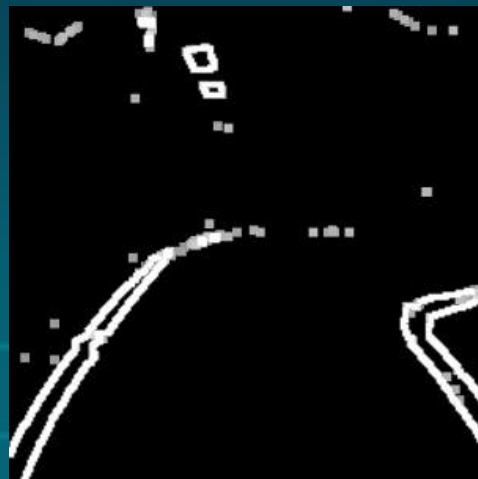
# 形態學-OpenCV



Opening 運算



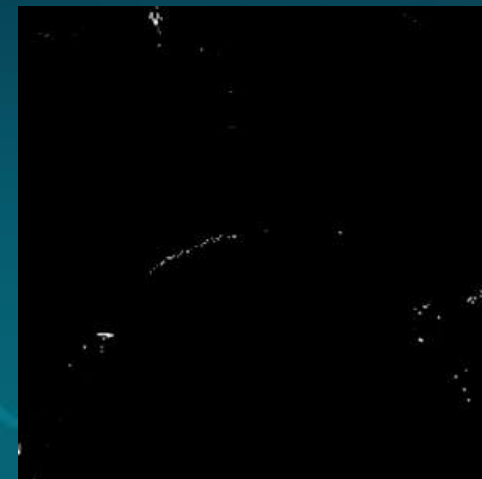
Closing 運算



Gradient 運算



Top Hat 運算



Black Hat 運算