數位影像處理
Digital Image Processing

Chapter 3
Spatial Filtering

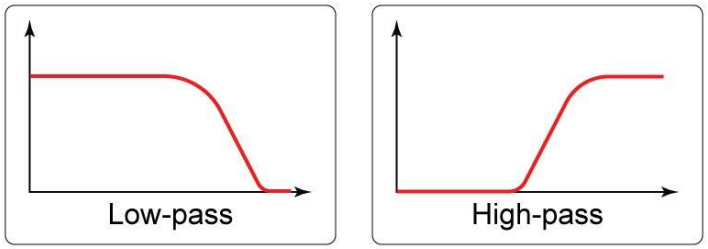
Ming-Han Tsai
Department of Information Engineering and Computer Science,
Feng Chia University

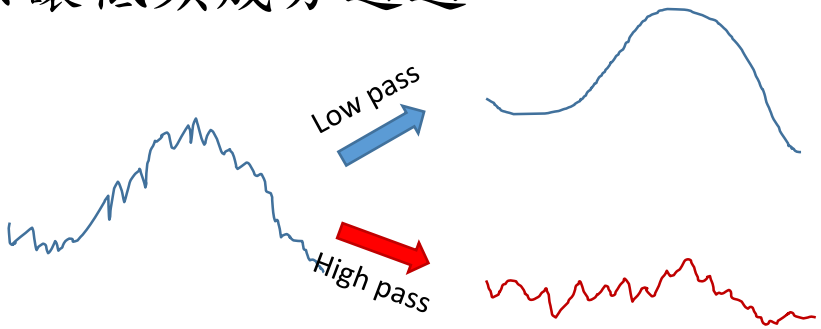
3 空間濾波 Spatial Filtering

- 1. 簡介
- 2. 空間濾波技術
- 3. 空間相關性與迴旋積
- 4. 平滑空間濾波器

1. 簡介

- 濾波器(filter)一詞是從頻域(frequency domain)發展而來。
- 濾波：指接受(通過)或拒絕某些頻率成分
  - 高通濾波器(highpass filter)只讓高頻成分通過
  - 低通濾波器(lowpass filter)只讓低頻成分通過





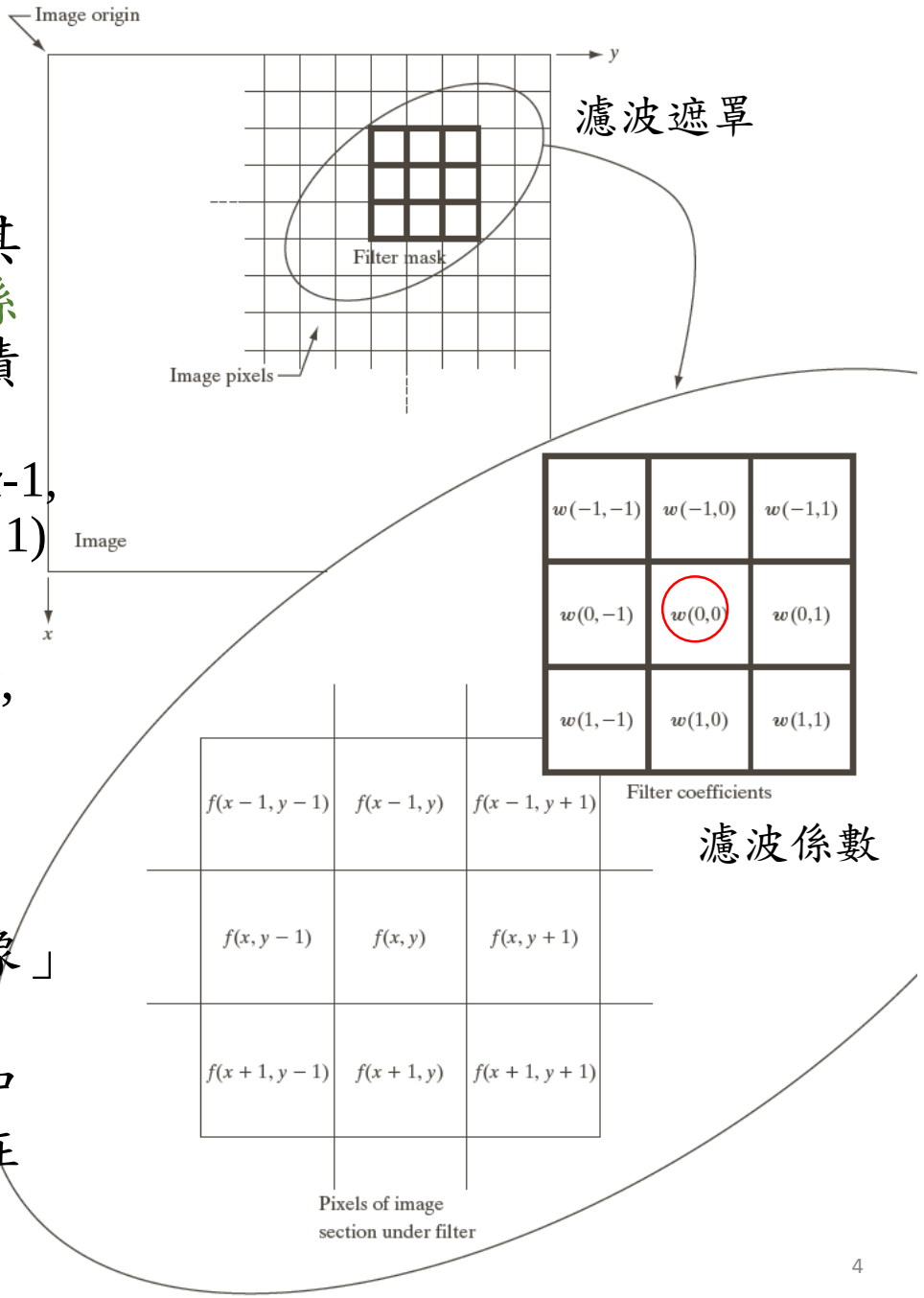
- 空間濾波器(spatial filter)：亦稱遮罩(mask)、核心(kernel)、模板(template)、視窗(window)…

2. 空間濾波技術

- 濾波處理：影像中的一點(x, y)，其濾波響應(response) $g(x, y)$是濾波係數與濾波器所圍繞之影像像素乘積的總和:

$$g(x, y) = w(-1,-1)\, f(x-1, y-1) + w(-1,0)\, f(x-1, y) + \ldots + w(0,0)\, f(x, y) + w(1, 1)\, f(x+1, y+1)$$

- 濾波器的中央係數$w(0, 0)$與像素(x, y)對齊。



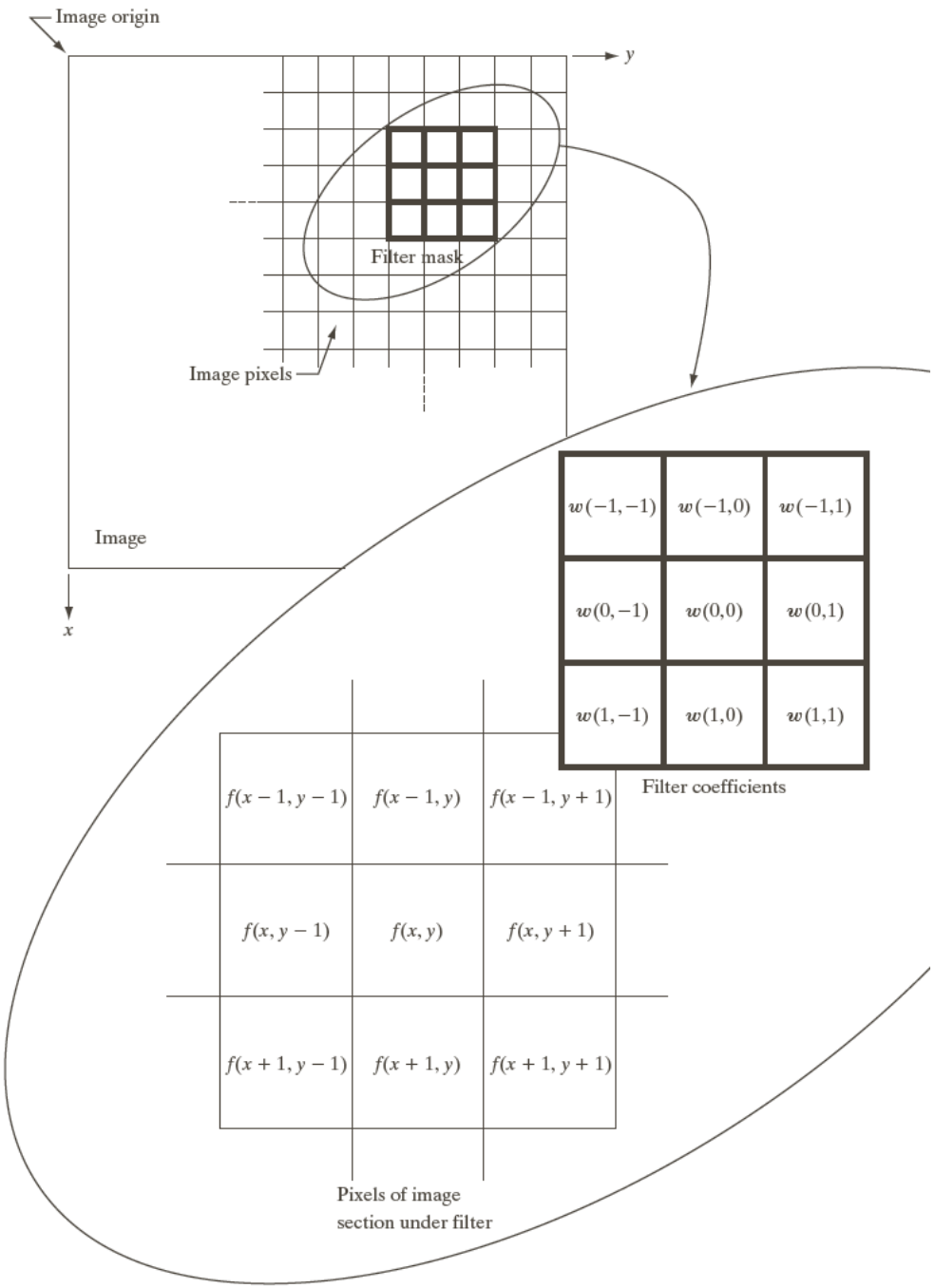
注意

- 濾波計算過程需有一張「新的影像」儲存結果。
- 很少把濾波後的像素取代原影像中對應位置上的值，因為當濾波還在執行時，這會改變影像的內容。

3. 空間相關性與迴旋積

- 相關性(correlation)：如上頁之運算，把濾波遮罩在影像上移動，並在每個位置上計算乘積的和。
- $g(x, y) = w(-1,-1)\, f(x-1, y-1)$
  $+ w(-1,0)\, f(x-1, y)+\ldots$
  $+w(0,0)\, f(x, y)$
  $+w(1, 1)\, f(x+1, y+1)$

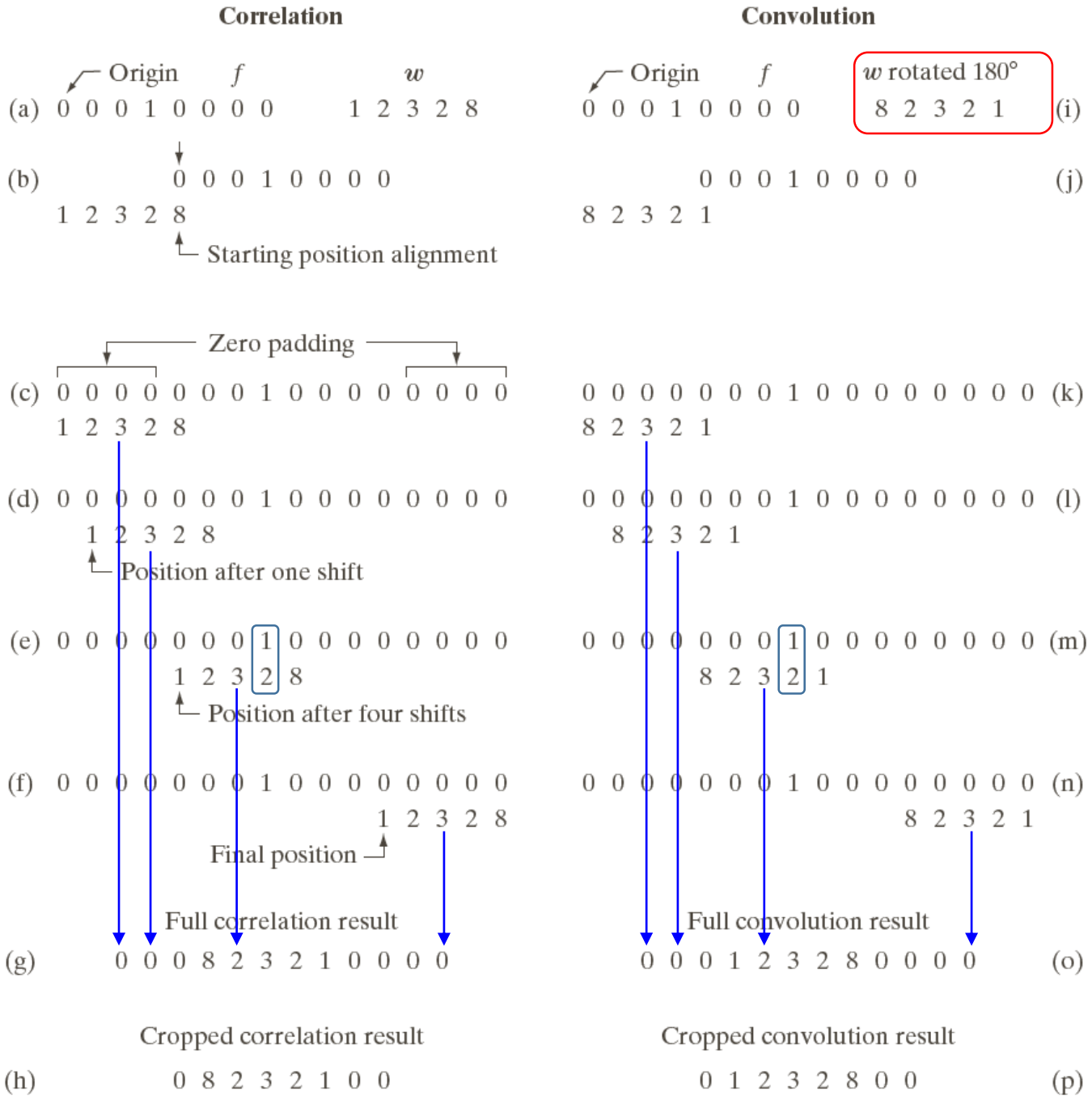
- 迴旋積(convolution)：計算方式相同，但濾波器要先旋轉180度。



一維範例

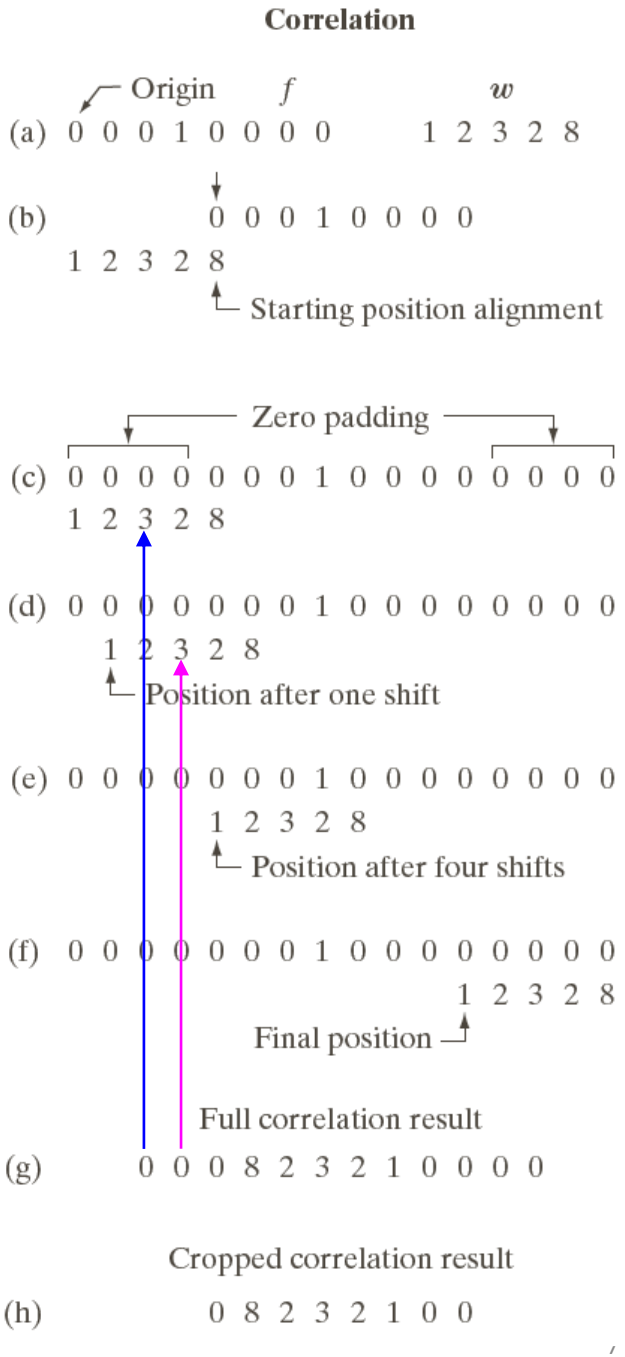
f: function

w: filter



3. 空間相關性與迴旋積

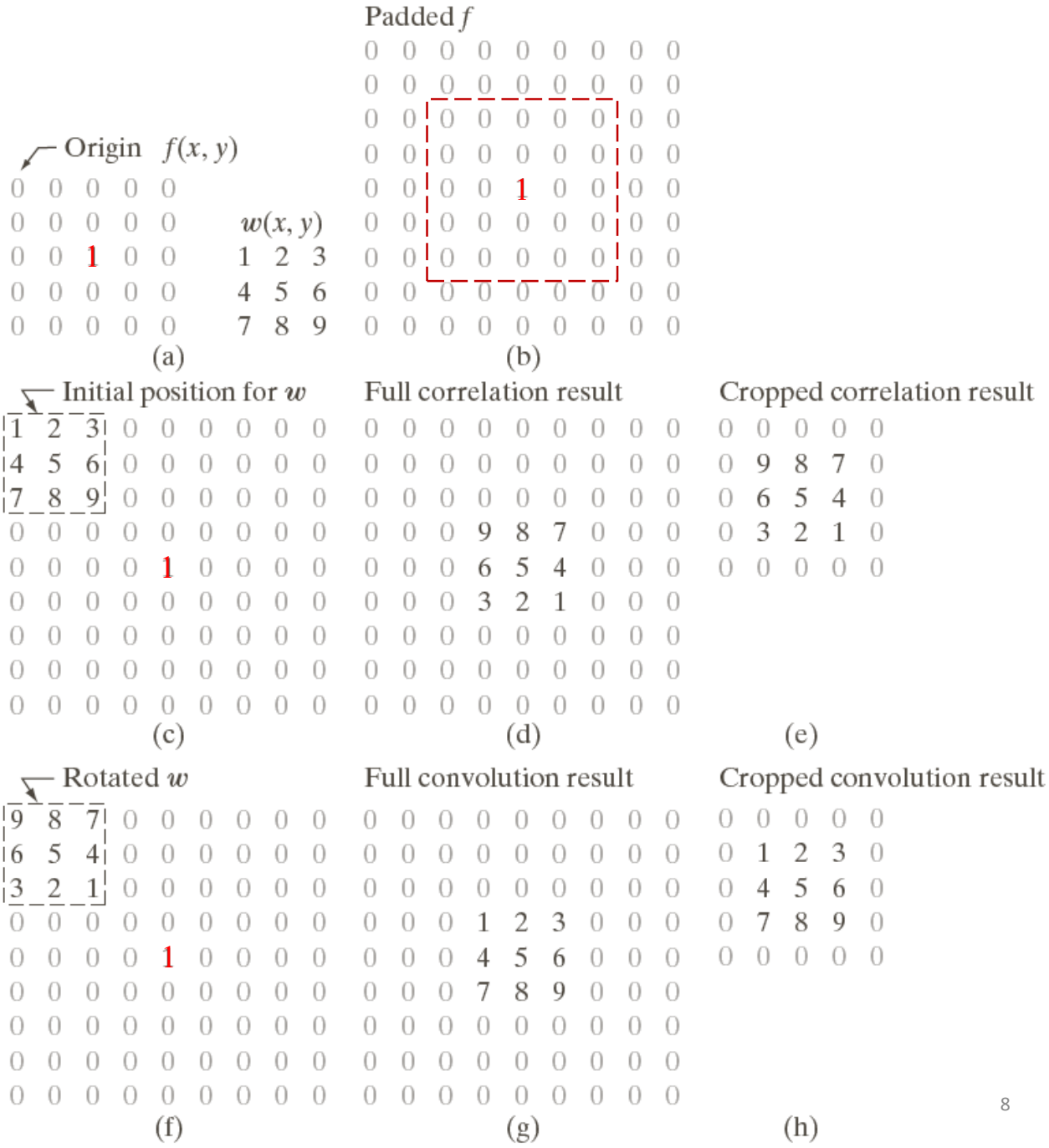
- 相關性是濾波器位移(displacement)的函數
  - 相關性第一個值對應濾波器的零位移
  - 第二個值對應到一個單位的位移
- 一個濾波器*w*與一個有單一個1，其它為0的函數 (稱「離散單位脈衝」discrete unit impulse)進行相關性運算後，得到一旋轉180度的*w*。
- 若將濾波器先作旋轉，則能在脈衝處產生與原函數一樣的訊號→迴旋積



二維範例

f: function

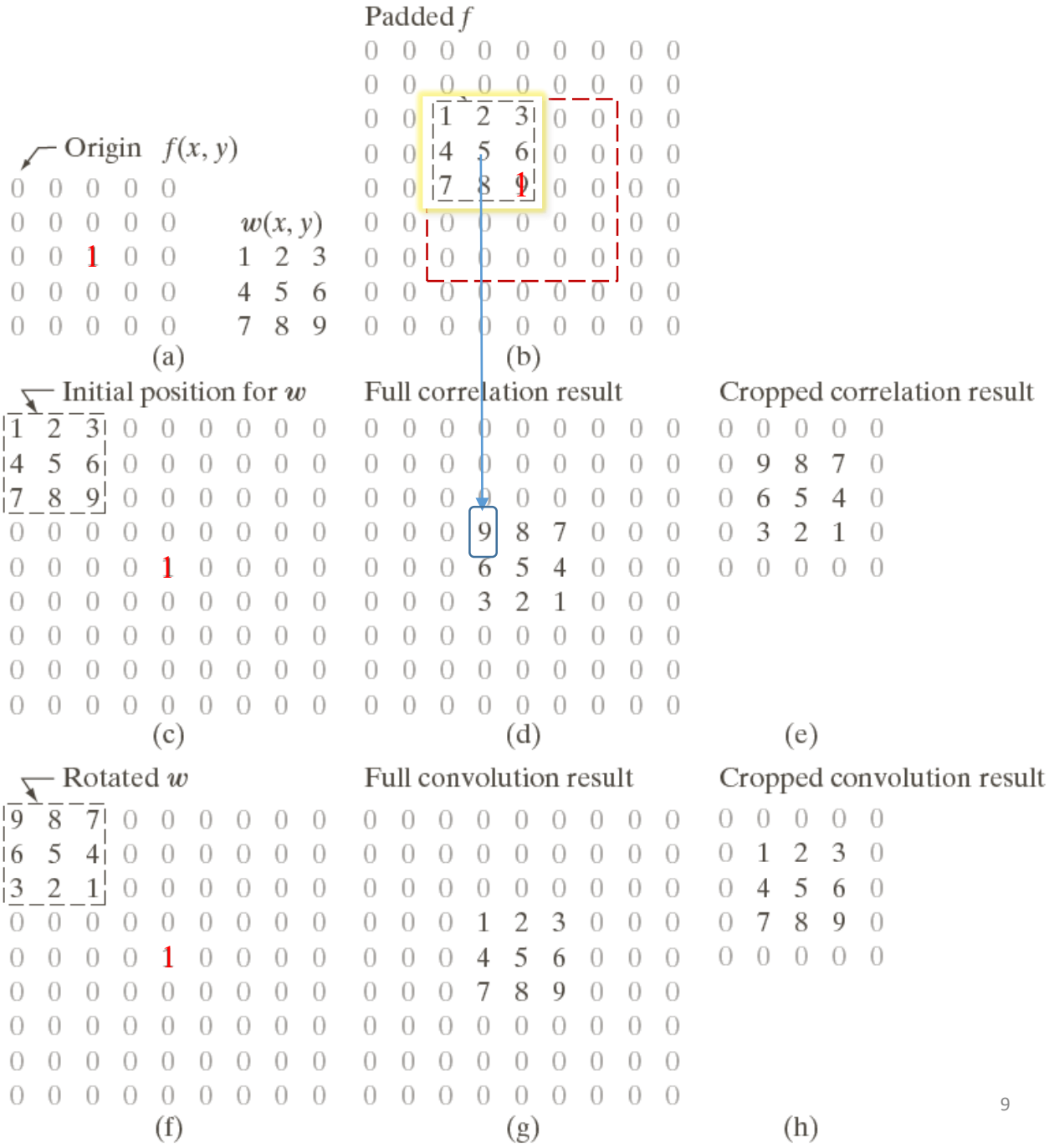
w: filter



二維範例

f: function

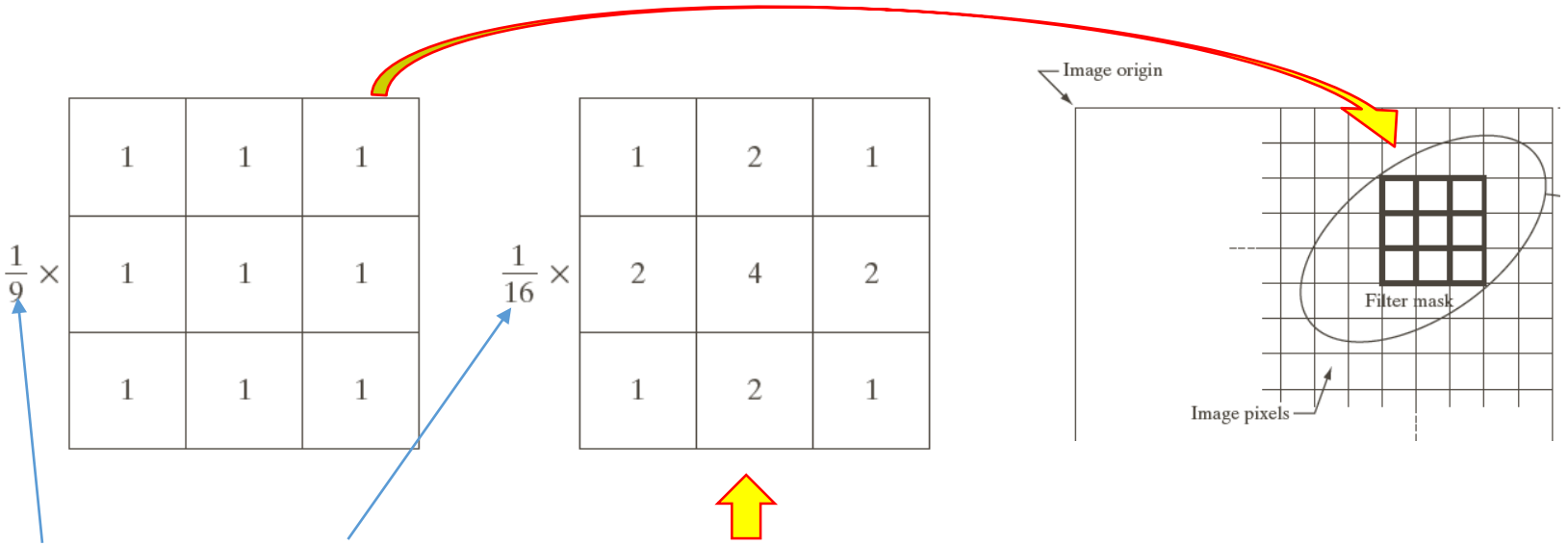
w: filter



- 若濾波遮罩是對稱的，則相關性與迴旋積會有相同結果。
- 在一些文獻中可能會提到「將影像與遮罩作迴旋積」，但可能指的是前述「滑動＋求積和」的過程。

4. 平滑空間濾波器

- 平滑(smoothing)濾波器用於模糊化與減少雜訊
- 平均濾波器(averaging filter)→線性濾波器



正規化，讓總和為1。

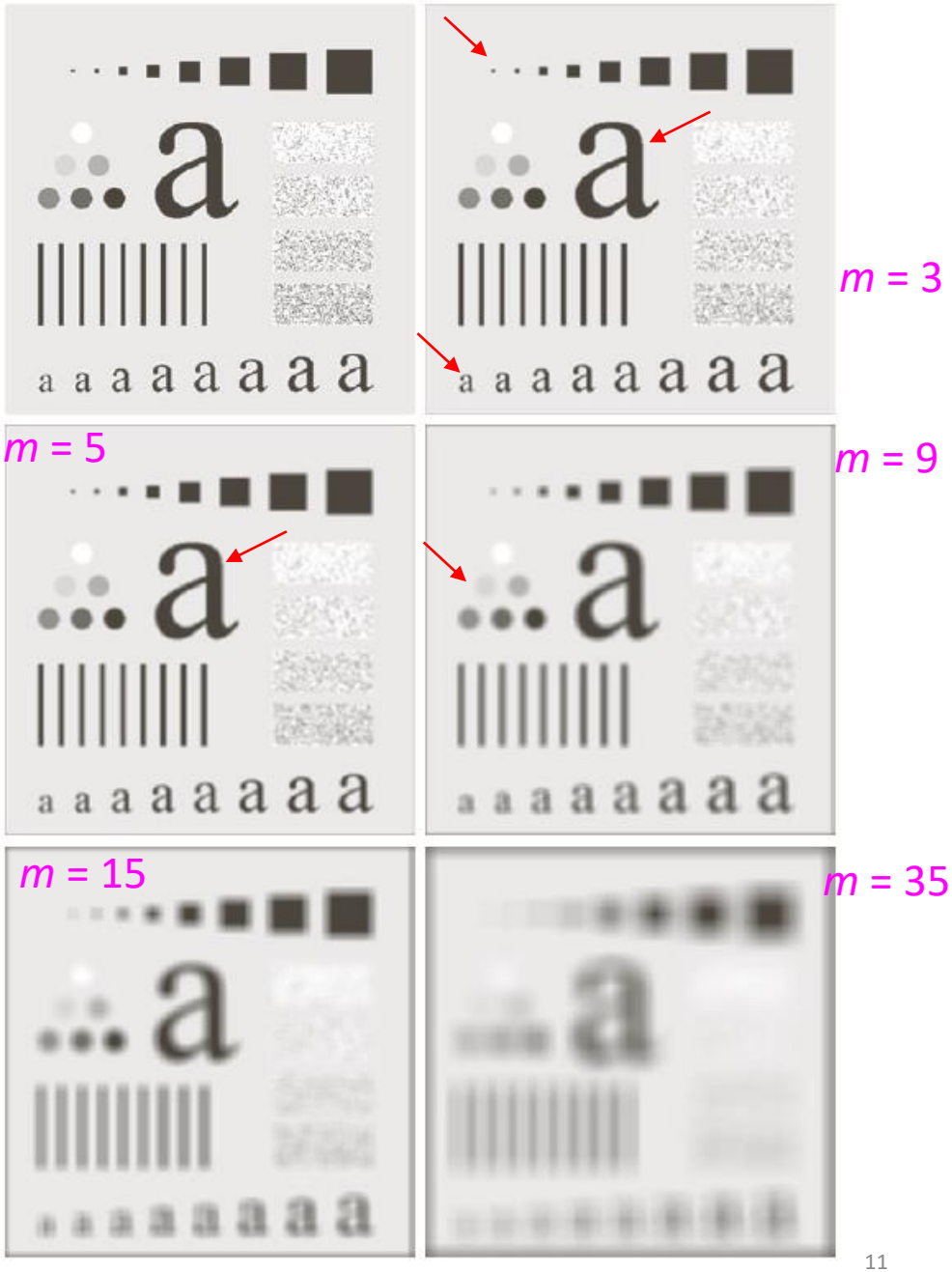
權重平均(weighted average)，給某些像素更高的重要性（權重）

4. 平滑空間濾波器

1	1	1
1	1	1
1	1	1

- 原影像：500x500
- 正方形平均濾波器
- 遮罩邊長 m = 3, 5, 9, 15, 35
- 圖上端黑正方形大小： 3, 5, 9, 25, 35, 45, 55。間隔25像素。

- m=3，輕微模糊，與遮罩同樣大小的細節影像較大。
  字元鋸齒邊緣被平滑化。
- m=5，類似，模糊略增。
- m=9，黑圈已不易與背景區分。
- m=15 and 35，小方塊、圓圈、雜訊矩形已混合到背景中。



OpenCV: Image Blur - Average

- Smooth an image by replacing each pixel by the average pixel value computed over a rectangular neighborhood.
- OutputArray **dst = blur**(InputArray **src**, Size **ksize**, Point **anchor**=Point(-1,-1), int **borderType**=BORDER_DEFAULT)
  - **src** – input image
  - **ksize** – blurring kernel size.
  - **anchor** – anchor point; default value Point(-1,-1) means that the anchor is at the kernel center.
  - **borderType** – border mode used to extrapolate pixels outside of the image.

➔ result=cv2.blur(image, (3,3));

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Such a matrix is sometimes called a kernel or a mask.

OpenCV: Image Blur - Average

Image smoothing by block averaging

Average Blur

抹平!

3*3

7*7

11*11

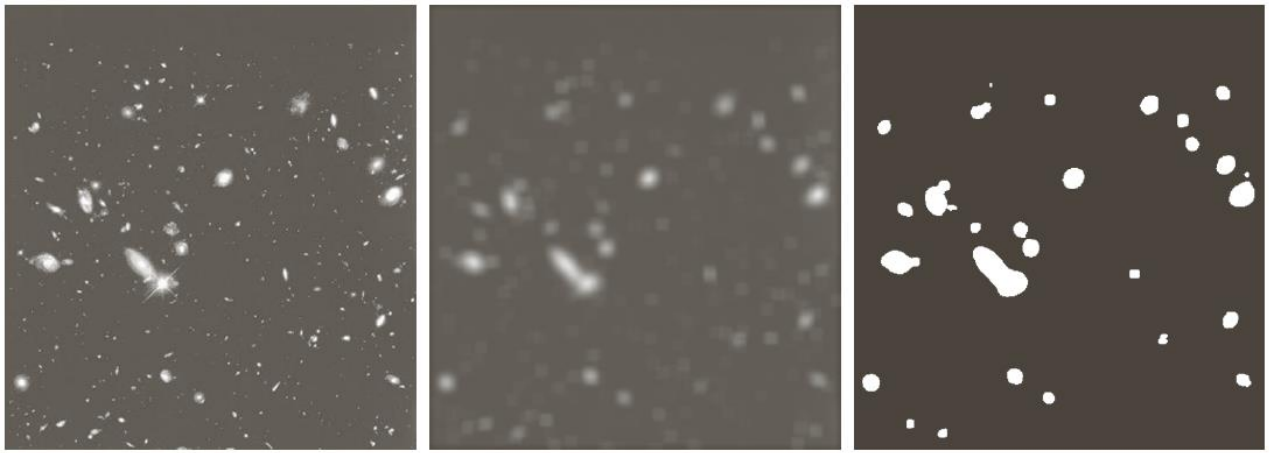
4. 平滑空間濾波器

- 空間平均的一個重要應用：為了獲得感興趣物體的整體表示而將一影像模糊，使較小的物體融入背景，較大物體則呈「斑點狀」，而較易偵測到。

門檻化 (Thresholding)

- $f(x, y)$: 像素值，$g(x, y)$: thresholding後的結果
- T: 門檻值

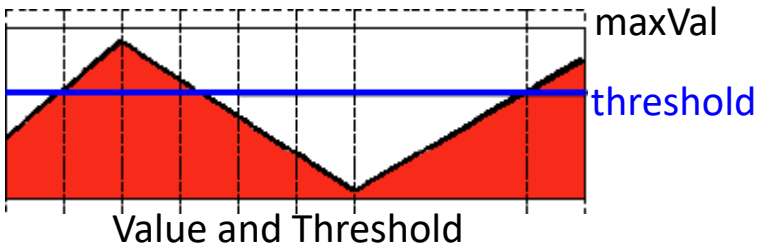
$$g(x,y) = \begin{cases} 255, if\ f(x,y) > T \\ 0, if\ f(x,y) \le T \end{cases} \quad or\ g(x,y) = \begin{cases} 1, if\ f(x,y) > T \\ 0, if\ f(x,y) \le T \end{cases}$$



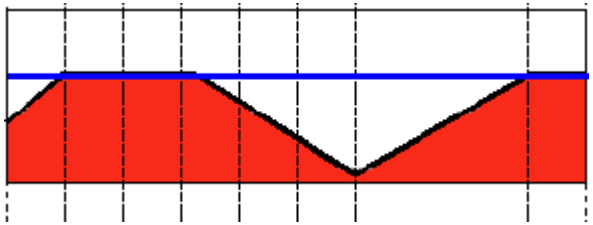
(a)哈伯望遠鏡之太空影像，528x485
(b)用15x15的平均遮罩作濾波
(c) 對(b)做門檻化

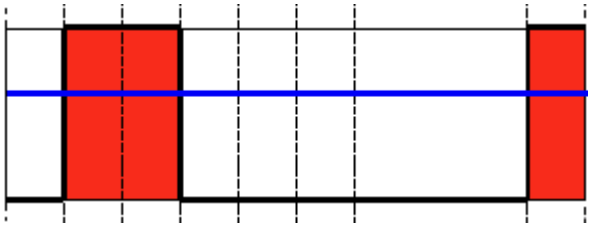
OpenCV Threshold Types

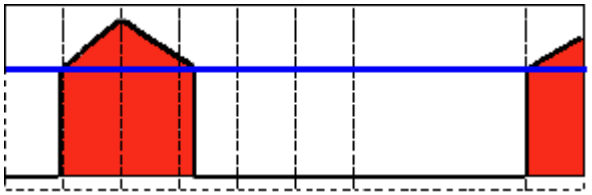
- dst = threshold (src, threshold, max_value, threshold_type);

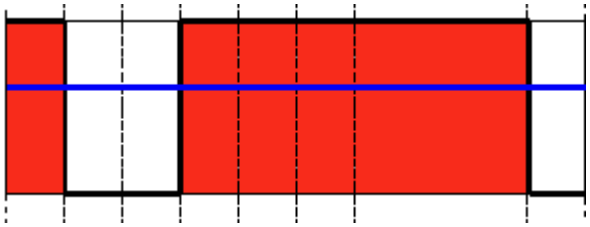


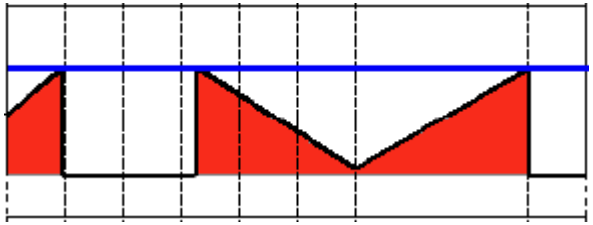
Value and Threshold


THRESH_TRUNC


THRESH_BINARY


THRESH_TOZERO


THRESH_BINARY_INV


THRESH_TOZERO_INV

Thresholding 的應用

- cvtColor (src, CV_BGR2GRAY);
- threshold (src, threshold, max_value, threshold_type);

cvtColor (frame, gray, CV_BGR2GRAY);

frame (3 channels)

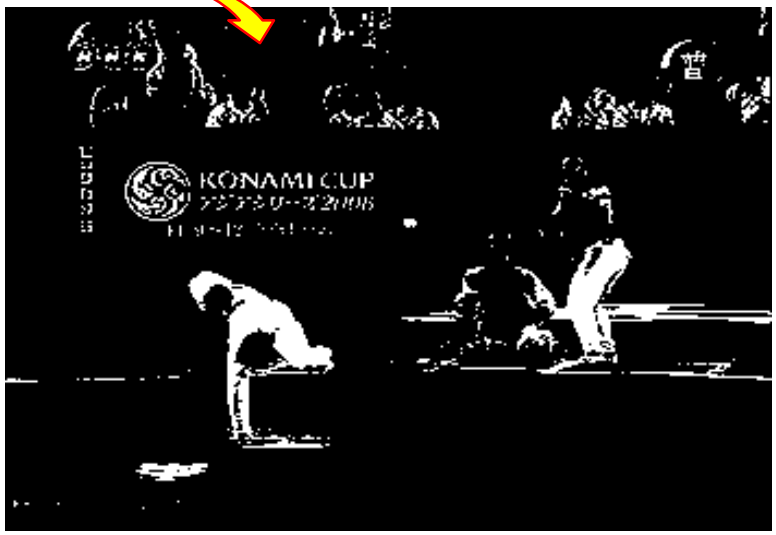
gray (1 channel)

Thresholding 的應用

- cvtColor (src, dst, CV_BGR2GRAY);
- threshold (src, dst, threshold, max_value, threshold_type);

threshold (gray, white, 200, 255, **THRESH_BINARY**);

gray

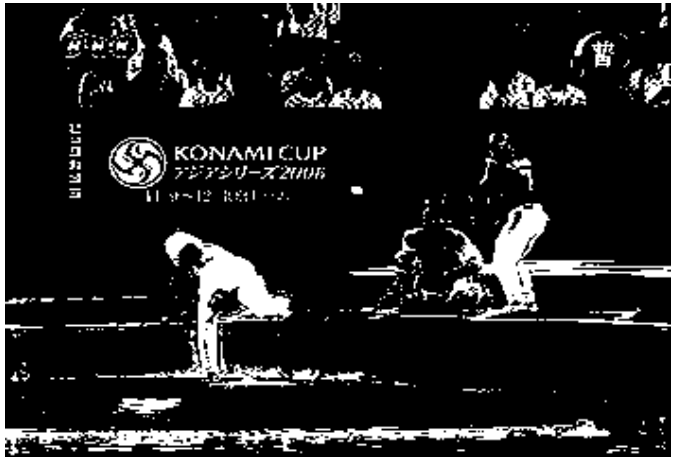


White (threshold = 200)

Thresholding 的應用

- Different thresholds

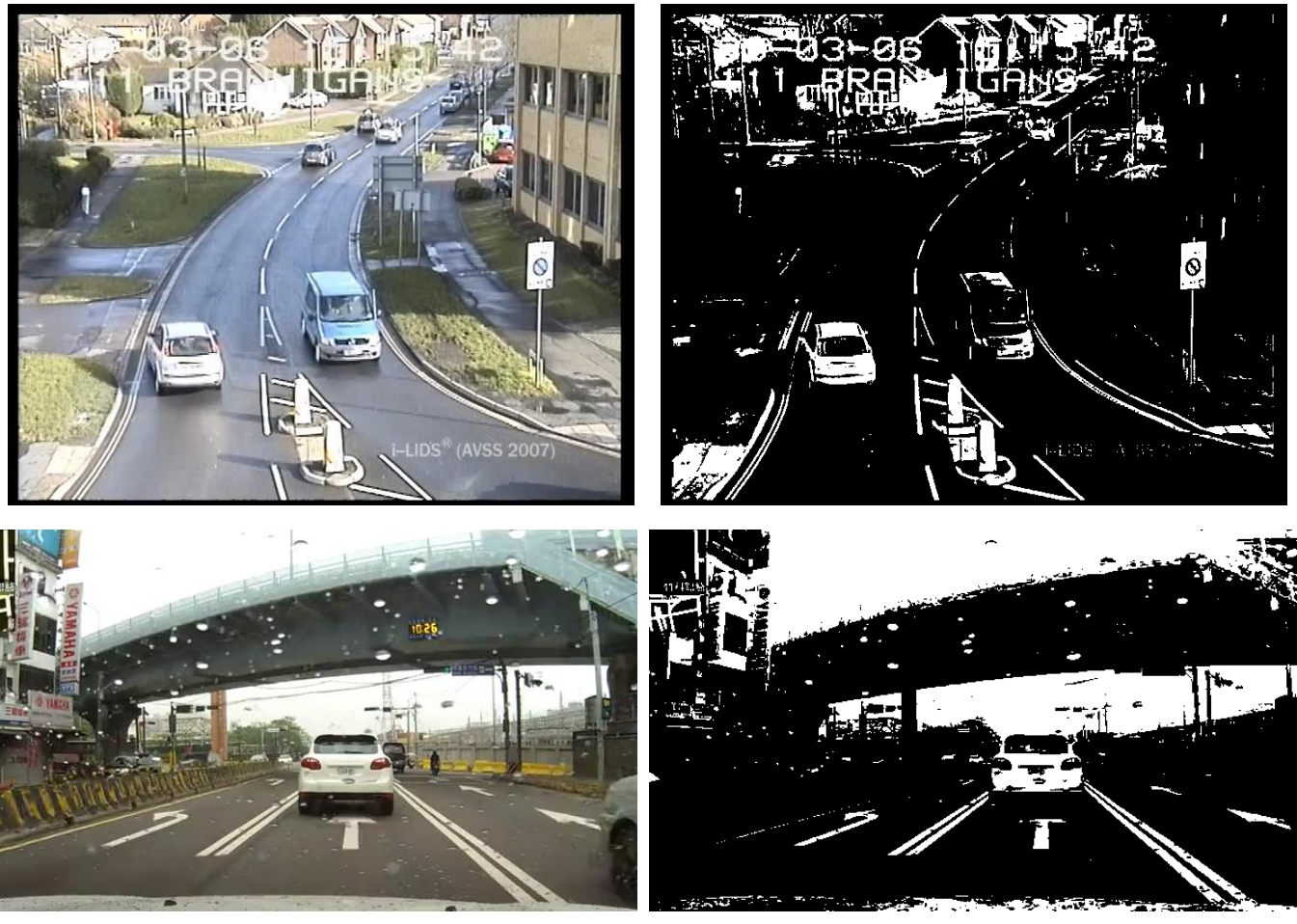
(threshold = 200)



(threshold = 180)

Thresholding 的應用

Thresholding 的應用

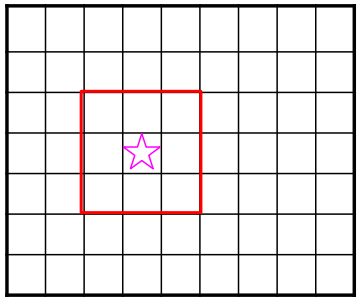


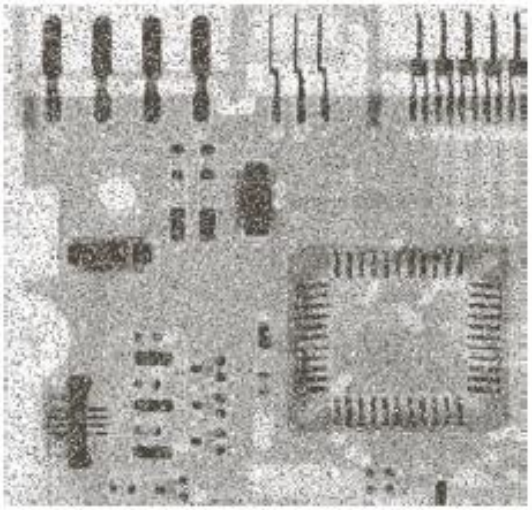
4. 平滑空間濾波器

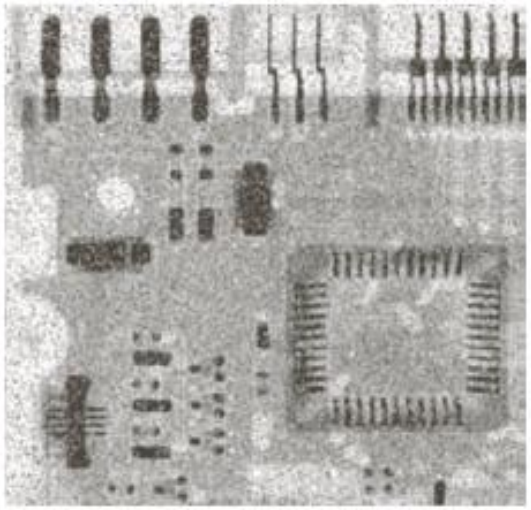
23, 23, 24, 25 ,25, 27, 27, 30, 235
Median = 25
Avg = 48.8

非線性濾波器

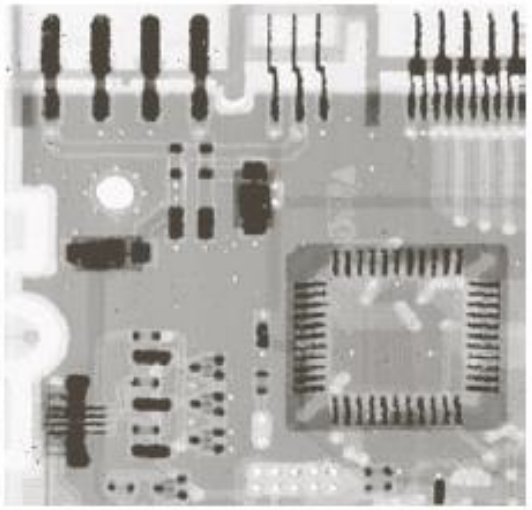
- 以中值濾波器(median filter)最常用
- 在一遮罩內，取中間值當作結果。
- 有效去除脈衝雜訊(impose noise)，因為在影像上脈衝雜訊看起來有黑點與白點，又稱椒鹽雜訊(salt-and-pepper noise)







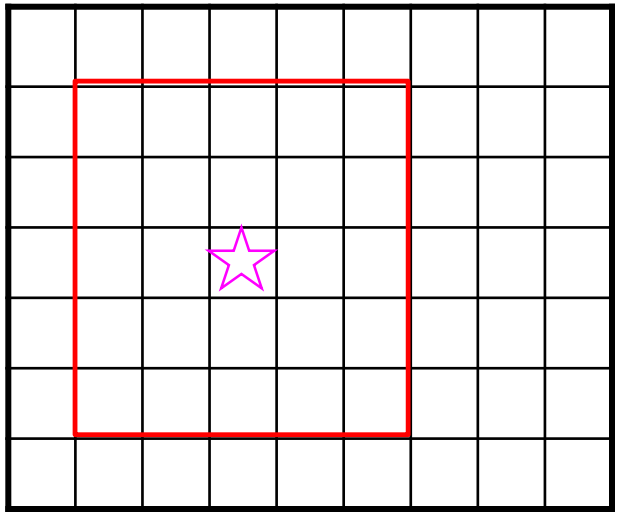
3x3 average filter



3x3 median filter

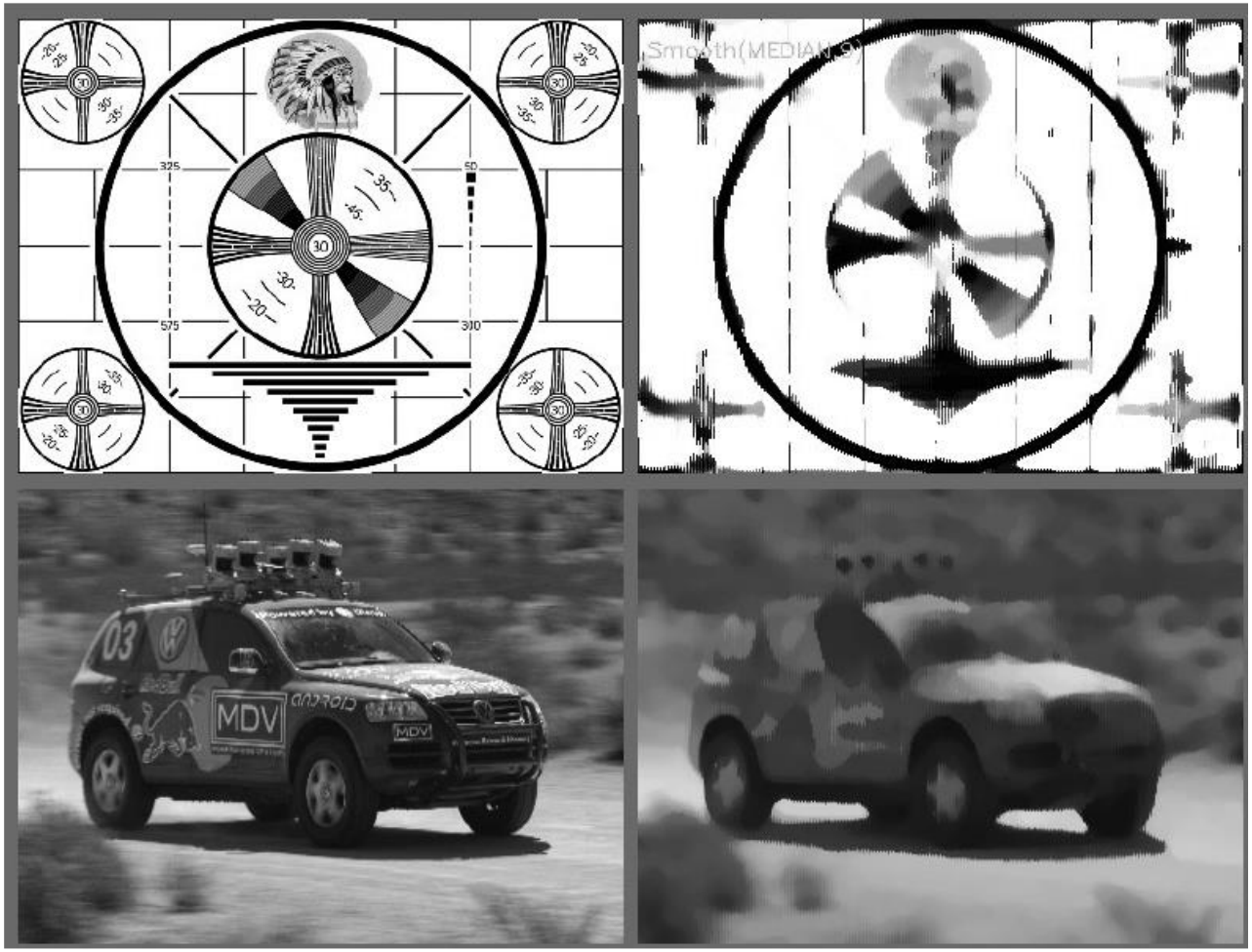
Image Blur - Median

- The median filter replaces each pixel by the median or "middle" pixel (as opposed to the mean pixel) value in a square neighborhood around the center pixel.
- OutputArray **dst medianBlur**(InputArray **src**, int **ksize**)
  - **src** – input image.
  - **dst** – output image of the same size and type as src.
  - **ksize** – aperture linear size; it must be odd and greater than 1, for example: 3, 5, 7 ...
    It is **int → square kernal**



→result =
cv2.medianBlur(image, 5);

Image Blur - Median



Median Blur

1	1	1
1	2	3
1	1	1

1	1	1
1	1	3
1	1	1

挑出中位數 = 1

3*3

7*7

11*11

Image Blur - Median

- Simple blurring by averaging can be sensitive to image noise, especially large isolated outlier points.
- Median filtering is able to ignore the outliers by selecting the middle points.
  - particularly useful to combat salt-and-pepper noise

Practice

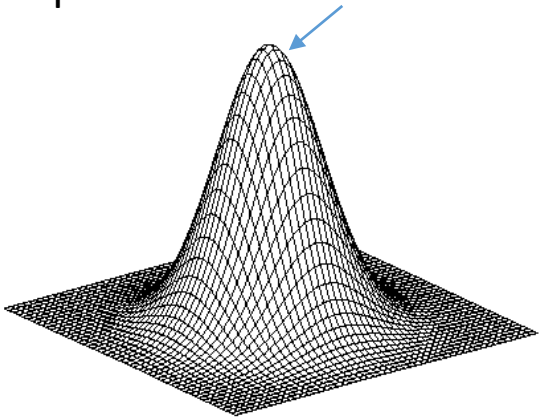
- 寫個小程式比較Average Filter跟Medium Filter的效果
- OutputArray **dst** = **blur**(InputArray **src**, Size **ksize**, Point **anchor**=Point(-1,-1), int **borderType**=BORDER_DEFAULT)
  - Result = cv2.blur(image, (3,3));
- OutputArray **dst** **medianBlur**(InputArray **src**, Size **ksize**)
  - result = cv2.medianBlur(image, 3);
- 改變kernel大小

OpenCV Functions

Image Blur – Gaussian filter

- Gaussian filtering is probably the most useful.
  - convolve each point in the input array with a Gaussian kernel and then summing to produce the output array.

The pixels closer to the center are weighted more.



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

5x5 kernel, $\sigma_x = \sigma_y = 1$

Recall → 4. 平滑空間濾波器

- 平滑(smoothing)濾波器用於模糊化與減少雜訊
- 平均濾波器(averaging filter)→線性濾波器

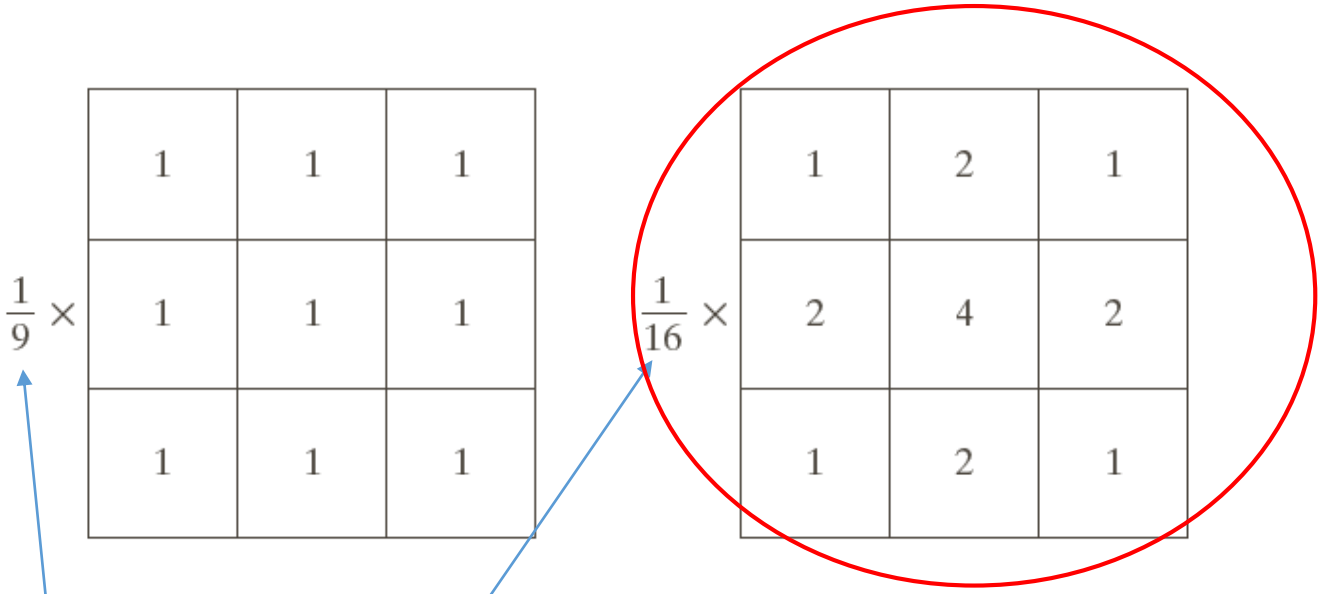
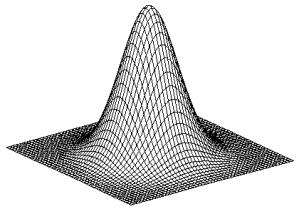


Image Blur – Gaussian filter

- OutputArray **dst** **GaussianBlur**(InputArray **src**, Size **ksize**, double **sigmaX**, double **sigmaY**=0, int **borderType**=BORDER_DEFAULT)
  - **ksize** – Gaussian kernel size.
    - ksize.width and ksize.height can differ but they both must be positive and odd.
    - Or, they can be zero's and then they are computed from sigma* .
  - **sigmaX** – Gaussian kernel standard deviation in X direction.
  - **sigmaY** – Gaussian kernel standard deviation in Y direction.
    - if sigmaY is zero, it is set to be equal to sigmaX
    - if both sigmas are zeros, they are computed from ksize.width and ksize.height, respectively.



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

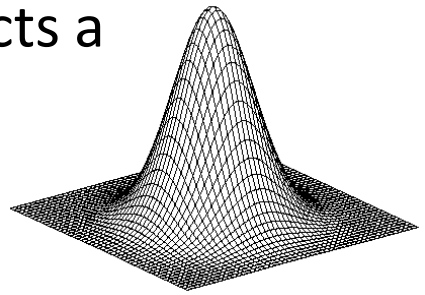
Image Blur – Gaussian filter

Image Blur – Bilateral filter

- Bilateral filtering →edge-preserving smoothing.
- A typical motivation for Gaussian smoothing:
  - pixels in a real image should vary slowly over space and thus be correlated to their neighbors
  - random noise can be expected to vary greatly from one pixel to the next (i.e., noise is spatially uncorrelated).
- It is in this sense that Gaussian smoothing reduces noise while preserving signal.
- Unfortunately, this method breaks down near edges, where you do expect pixels to be uncorrelated with their neighbors.

  →Gaussian smoothing smoothes away the edges.

Image Blur – Bilateral filter



- Like Gaussian smoothing, bilateral filtering constructs a weighted average of each pixel and its neighboring components.
- The weighting has two components:
  - the first is the same weighting used by Gaussian smoothing, based on the spatial distance from the center pixel
  - The second is also a Gaussian weighting but is based on the difference in intensity from the center pixel.
- You can think of bilateral filtering as Gaussian smoothing that weights more similar pixels more highly than less similar ones.
- The effect of this filter is typically to turn an image into what appears to be a watercolor painting of the same scene.
- This can be useful as an aid to segmenting the image.

http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/MANDUCHI1/Bilateral_Filtering.html

Comparison – Gaussian filter

Image Blur – Bilateral filter

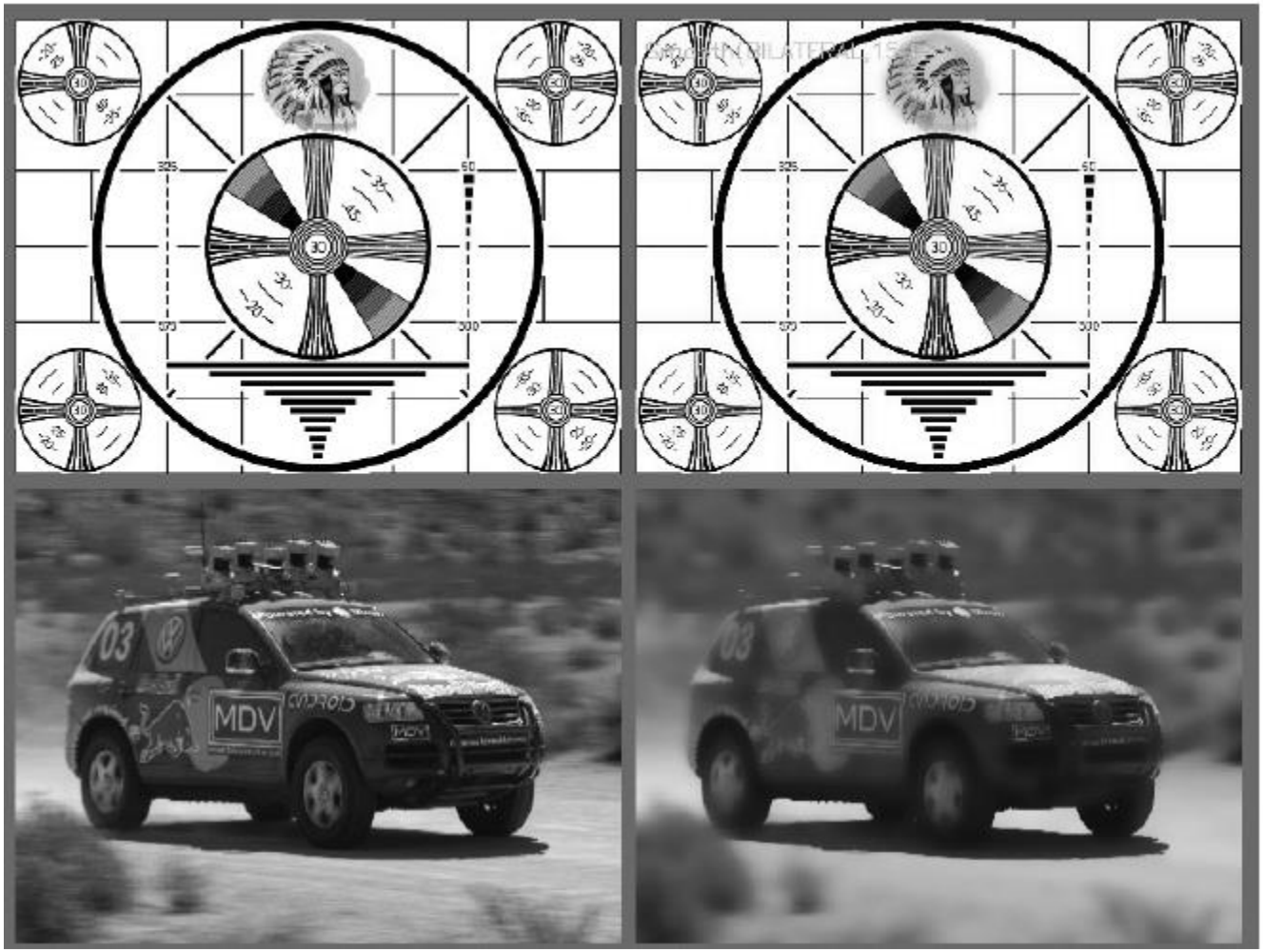
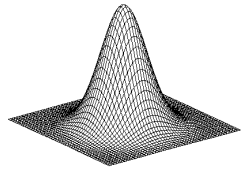


Image Blur – Bilateral filter

Image Blur – Bilateral filter

Image Blur – Bilateral filter



$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$

- void **bilateralFilter**(InputArray **src**, OutputArray **dst**, int **d**, double **sigmaColor**, double **sigmaSpace**, int **borderType**=BORDER_DEFAULT)
  - **d** – Diameter of each pixel neighborhood that is used during filtering.

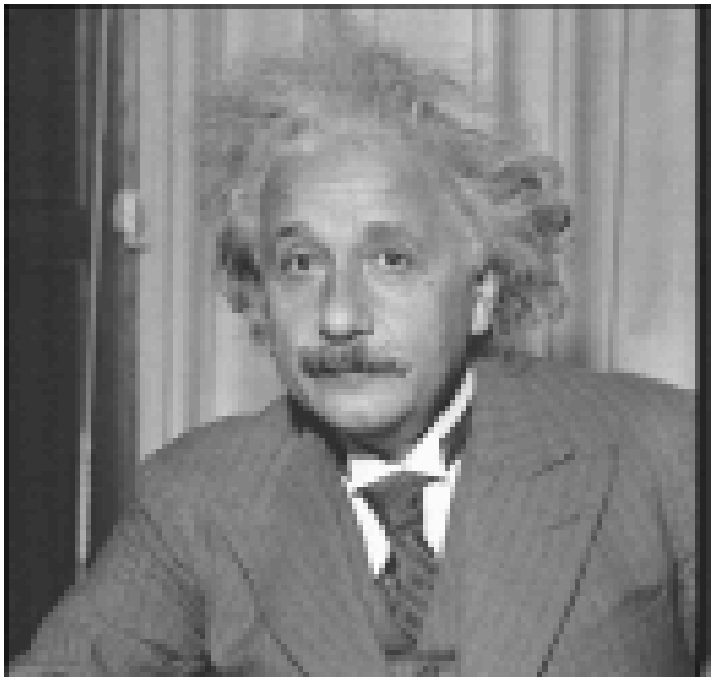
  →Large filters (d > 5) are very slow. Recommend: use d=5 for real-time applications.

  - **sigmaColor** – Filter sigma in the color space. A larger value of the parameter means that farther colors within the pixel neighborhood will be mixed together, resulting in larger areas of semi-equal color.
  - **sigmaSpace** – Filter sigma in the coordinate space. A larger value of the parameter means that farther pixels will influence each other as long as their colors are close enough.

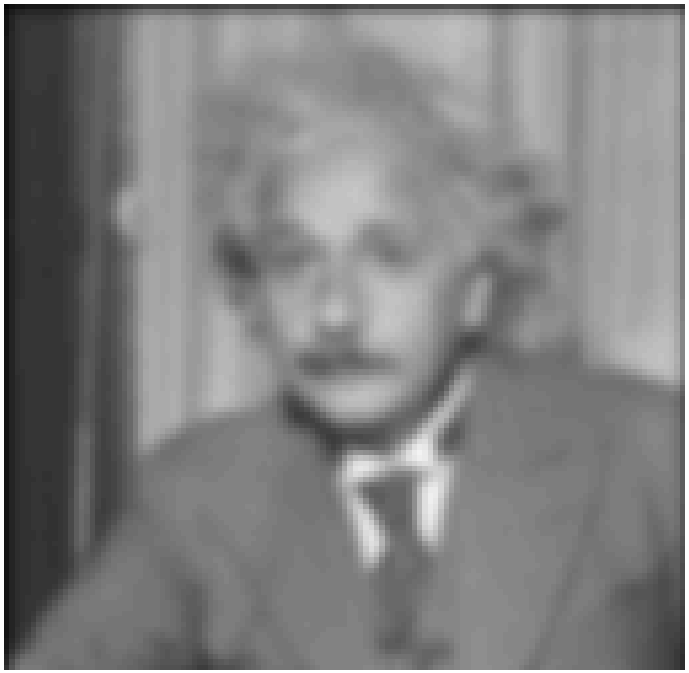
  →For simplicity, you can set the 2 sigma values to be the same.

  If they are small (< 10), the filter will not have much effect, whereas if they are large (> 150), they will have a very strong effect, making the image look "cartoonish".

Some applications



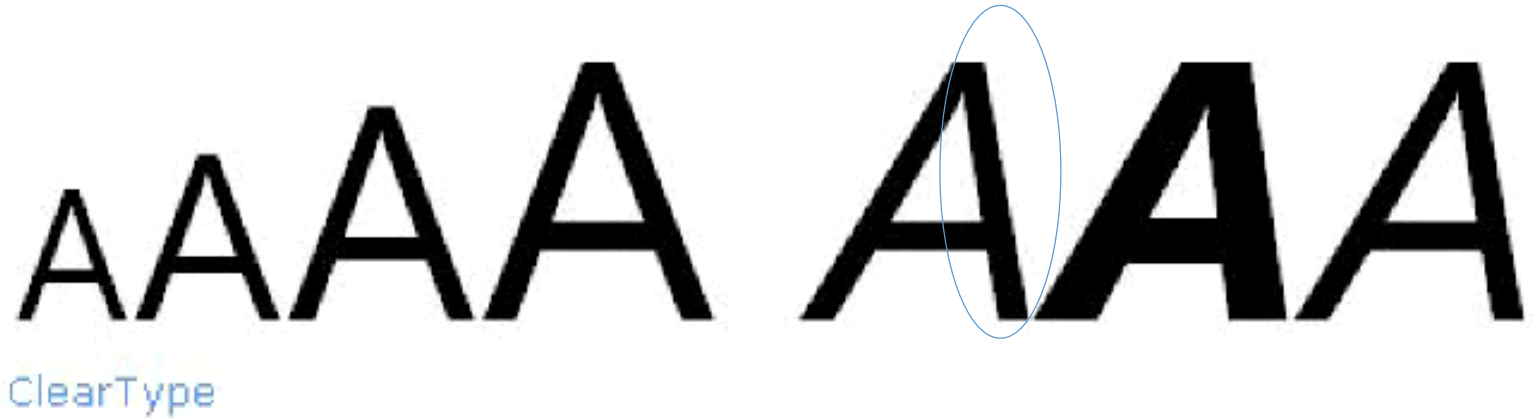
Gaussian blur



Downsample

Downsample

Clear Type Font



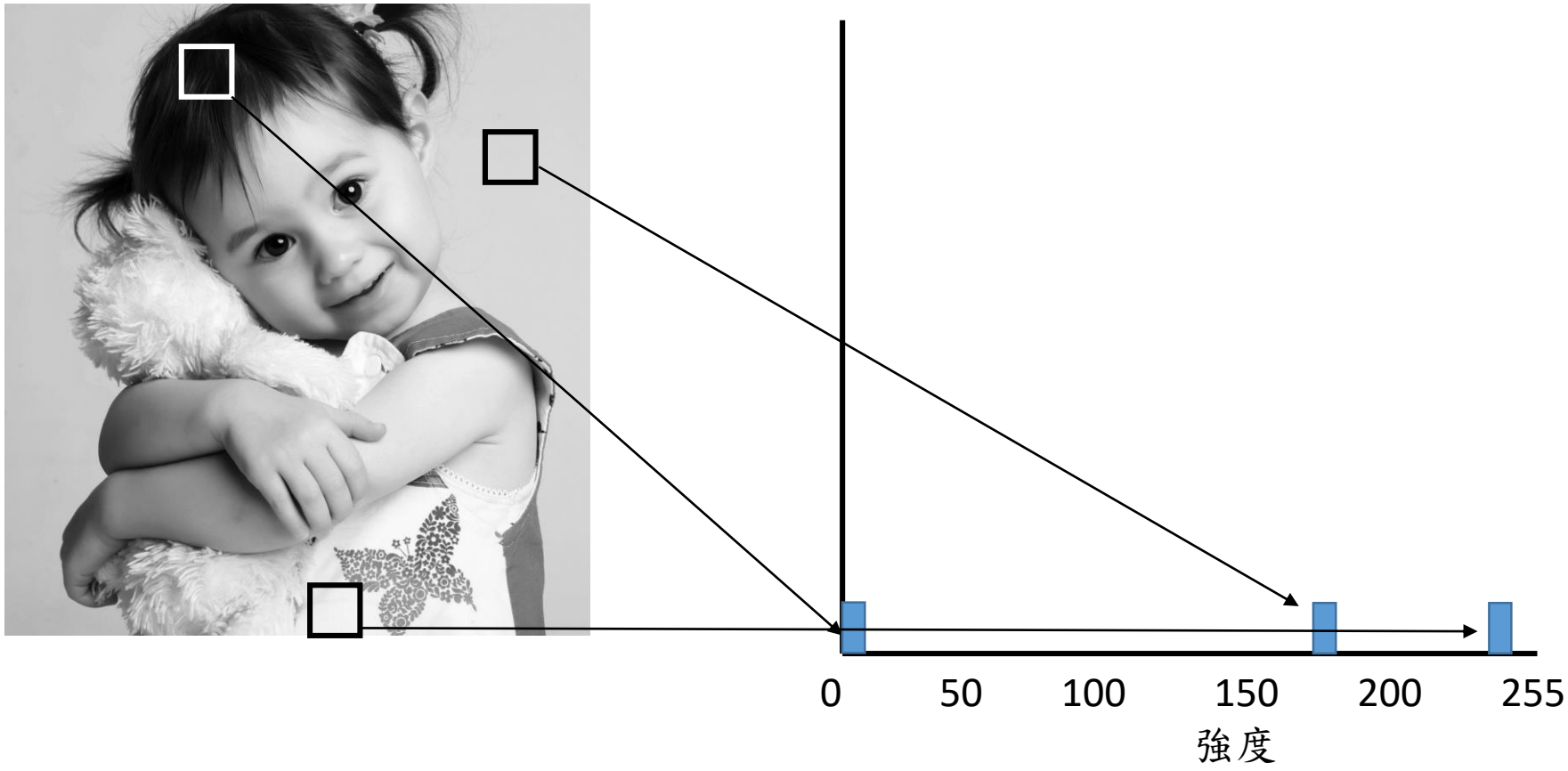
ClearType

None

Clear Type Font

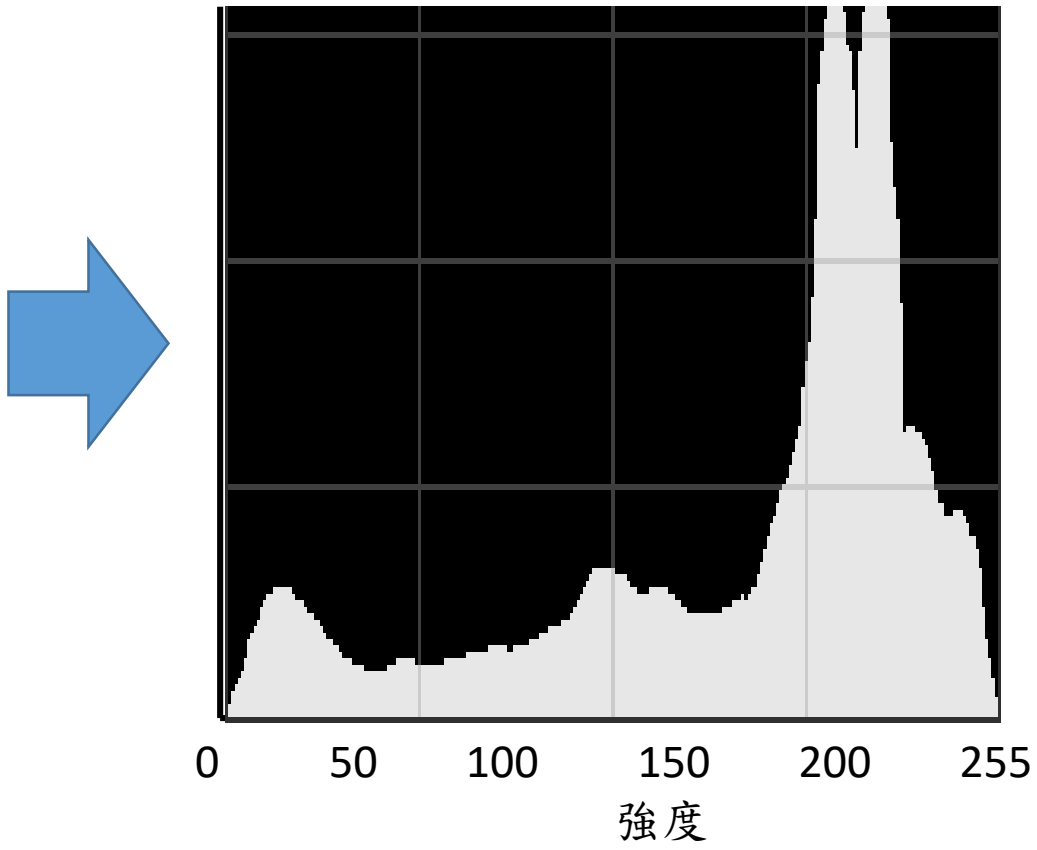
直方圖(Histogram)

- 將影像上的強度值統計的結果

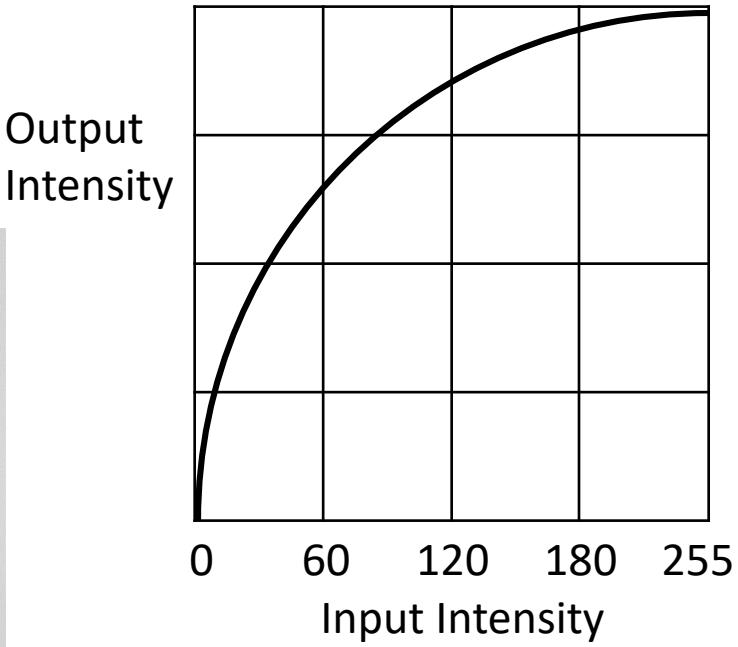


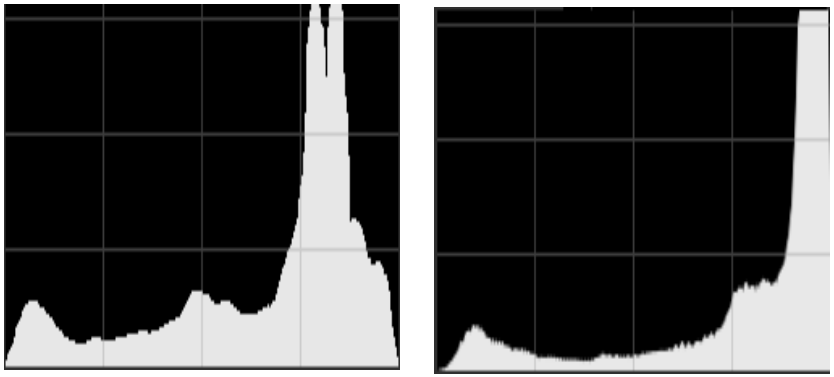
直方圖(Histogram)

- 將影像上的強度值統計的結果



直方圖調整





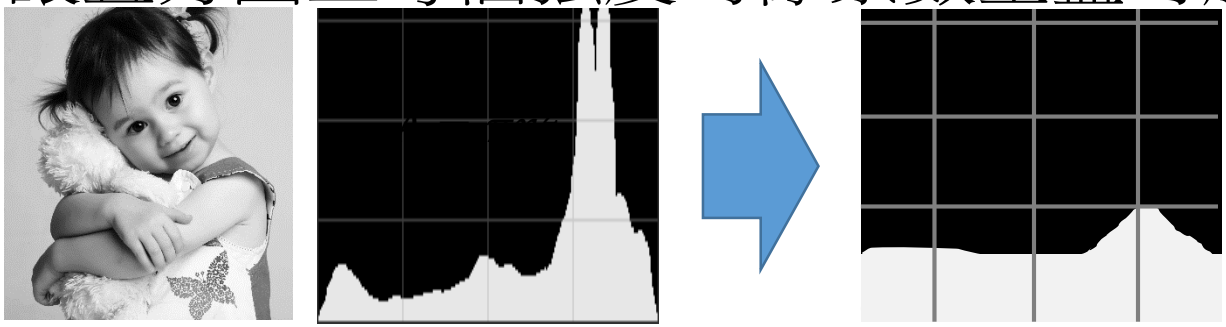
彩色

圖 6.36 對於 CMYK 彩色影像做色彩平衡修正

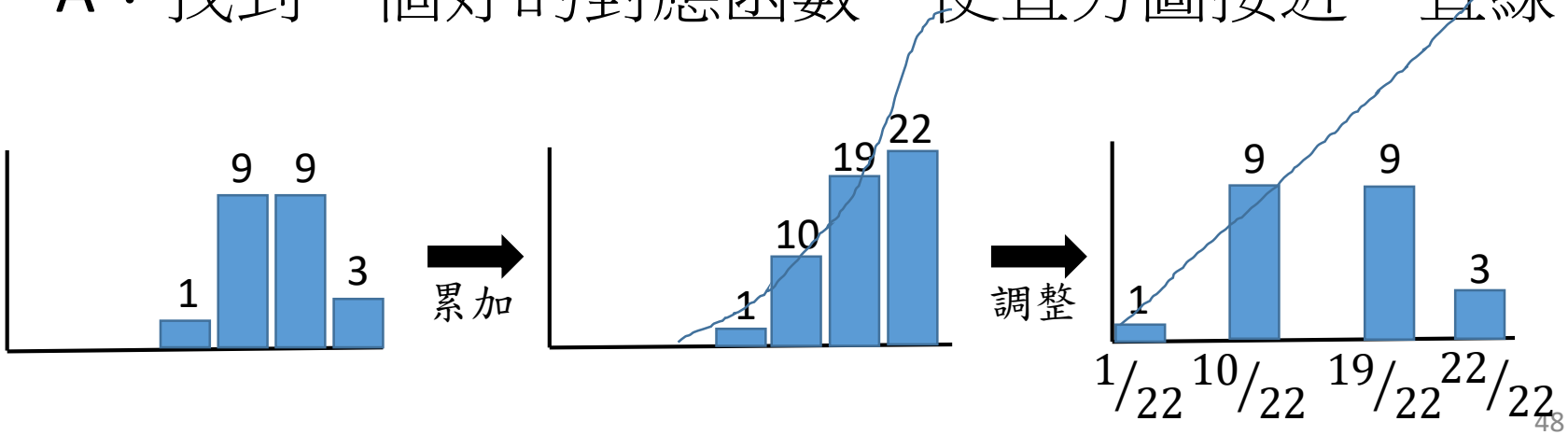
Histogram Equalization

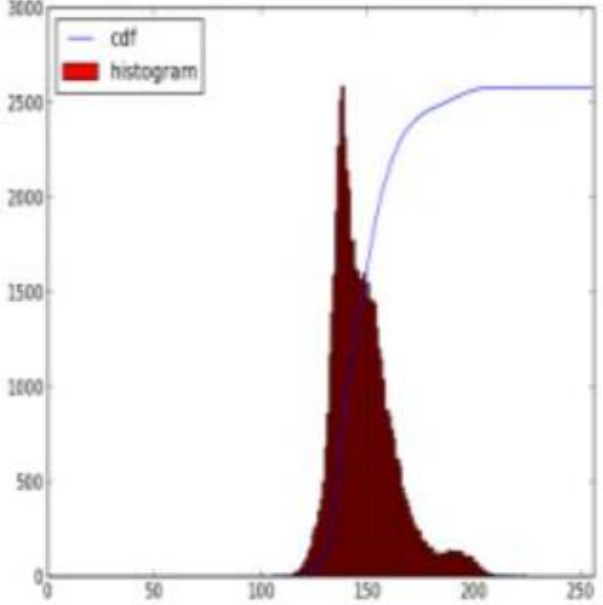
當影像強度其中在某一區(偏亮/偏暗)時我們希望讓畫面中的顏色強度較平均，較可看出細節

Q：如何讓直方圖上每個強度的像素數量盡可能相等？

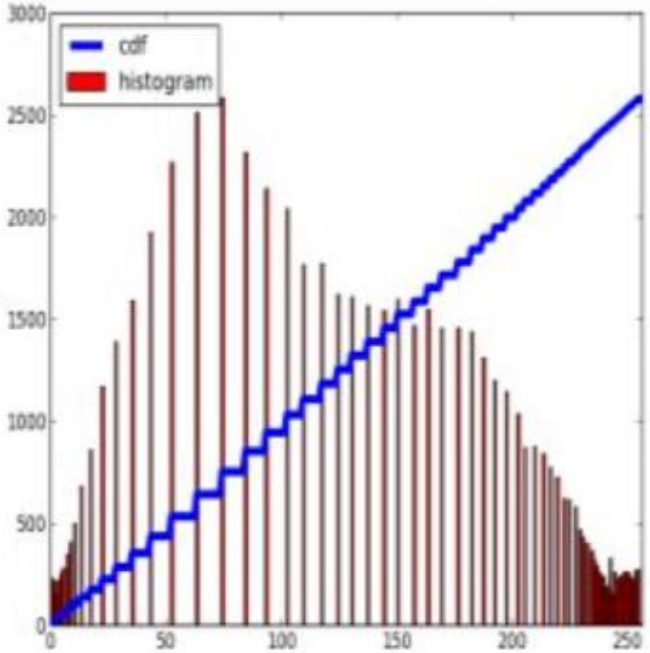


A：找到一個好的對應函數，使直方圖接近一直線




cdf
histogram
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250

image


cdf
histogram
3000
2500
2000
1500
1000
500
0
0
50
100
150
200
250

image

cv2.calcHist(images, channels, mask, histSize, ranges)

- images：要處理的圖片檔
- channels：指定產生的直方圖類型。例：[0]→灰階，[0, 1, 2]→RGB三色。
- mask：optional，只計算遮罩的部分影像。
- histSize：要切分的像素強度值範圍，預設為256。每個channel皆可指定一個範圍。例如，[32,32,32] 表示RGB三個channels皆切分為32區段。
- ranges：X軸(像素強度)的範圍，預設為[0,256]（最後那個值是表示<256）。

cv2.calcHist(img,[0],None,[256], [0, 256])

```
def calcAndDrawHist(image, color):
    hist= cv2.calcHist([image], [0], None, [256], [0.0,255.0])
    minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(hist)
    histImg = np.zeros([256,256,3], np.uint8)
    hpt = int(0.9* 256);

    for h in range(256):
        intensity = int(hist[h]*hpt/maxVal)
        cv2.line(histImg,(h,256), (h,256-intensity), color)

    return histImg;
```

```
def calcAndDrawHist(image, color):
    hist= cv2.calcHist([image], [0], None, [256], [0.0,255.0])
    minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(hist)
    histImg = np.zeros([256,256,3], np.uint8)
    hpt = int(0.9* 256);

    for h in range(256):
        intensity = int(hist[h]*hpt/maxVal)
        cv2.line(histImg,(h,256), (h,256-intensity), color)

    return histImg;



img = cv2.imread("image.jpg",0)


his = calcAndDrawHist(img,(255,255,255))
cv2.imshow("histogram",his)
cv2.waitKey()
```

```
eqimg = cv2.equalizeHist(img)
his2 = calcAndDrawHist(eqimg,(255,255,255))
cv2.imshow("img",img)
cv2.imshow("eqimg",eqimg)
cv2.imshow("his",his)
cv2.imshow("equalizeHist", his2)
cv2.waitKey()
```

Practice

- 比較Average Blur, Gaussian Blur 以及Medium Blur 的差別 1.Building.jpg 2.Face.jpg 3.自己挑一張影像

```
import numpy as np
import cv2

img = cv2.imread("building.jpg",0)

AverageBlur = cv2.blur(img,(5,5))
GaussianBlur = cv2.GaussianBlur(img,(5,5),0)
MediumBlur = cv2.medianBlur(img,5)

cv2.imshow("original",img)
cv2.imshow("AverageBlur",AverageBlur)
cv2.imshow("GaussianBlur",GaussianBlur)
cv2.imshow("MediumBlur",MediumBlur)
cv2.waitKey()
```