影像處理
Chapter 03 空間濾波器

Chien-Chang Chen
Tamkang University
Department of Computer Science and Information Engineering

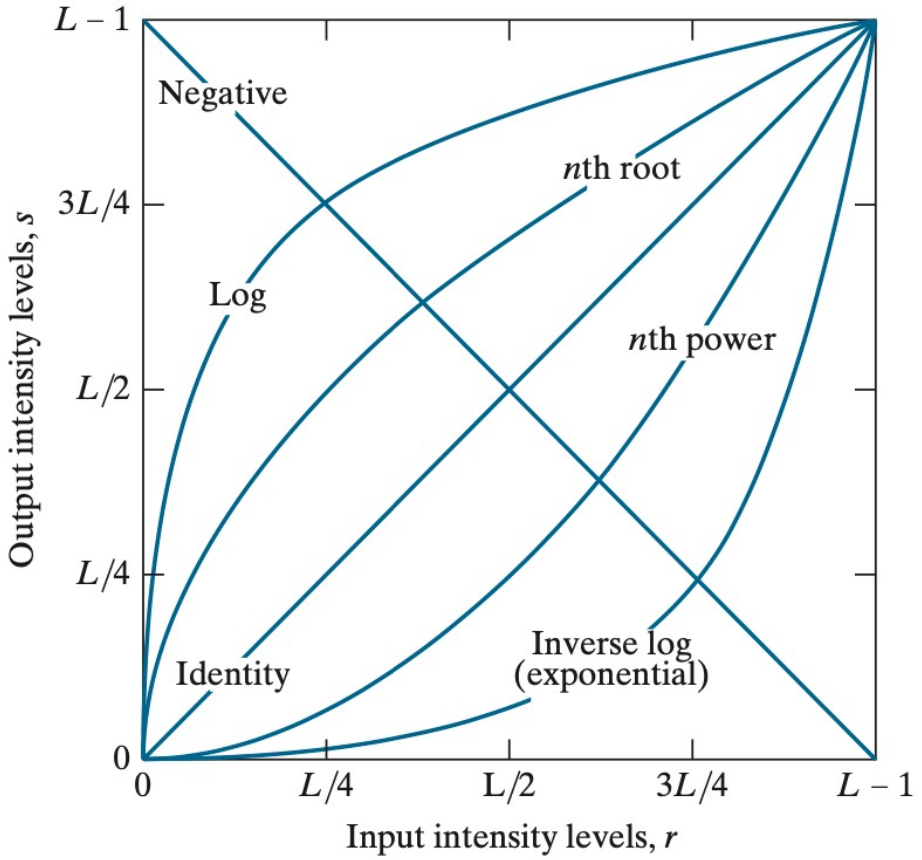
Image Enhancement

- *Enhancement* is the process of manipulating an image so that the result is more suitable than the original for a specific application

- For example,
  - Image too dark => brighten
  - Image too bright => darken

Some Basic Intensity Transform

FIGURE 3.3
Some basic intensity transformation functions. Each curve was scaled *independently* so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.



Intensity transformation

- Image Negative
- Log Transformation

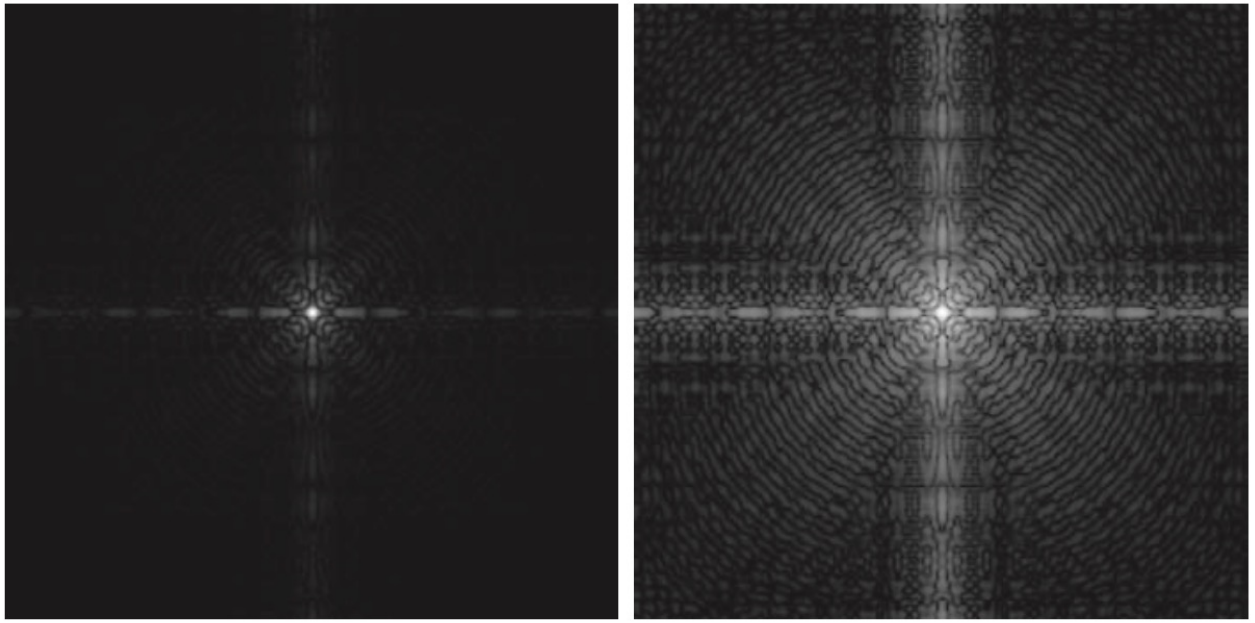
Log Transformation

$$s = c \log(1+r)$$

where c is a constant and it is assumed that $r \geq 0$

a b

FIGURE 3.5
(a) Fourier spectrum displayed as a grayscale image. (b) Result of applying the log transformation in Eq. (3-4) with $c = 1$. Both images are scaled to the range $[0, 255]$.



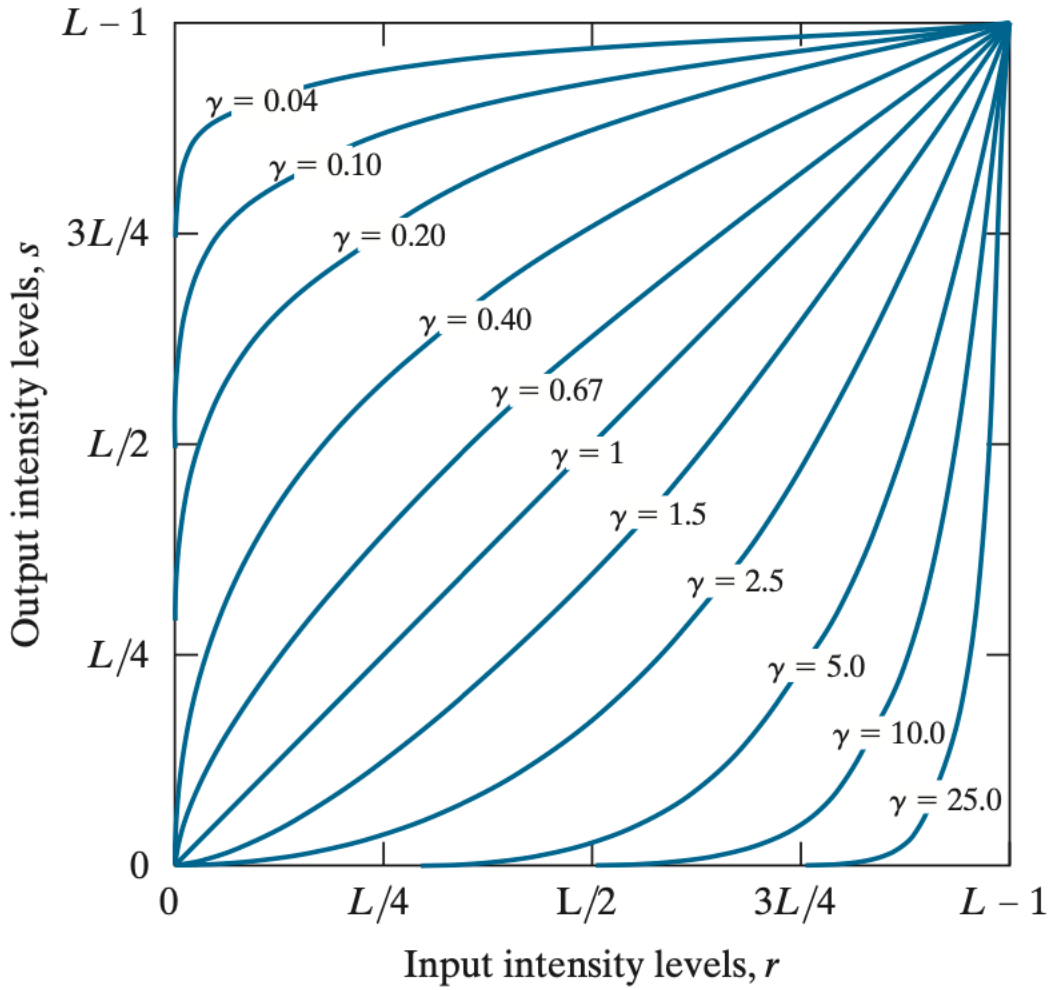
Gamma Correction伽瑪矯正

$$s = cr^{\gamma}$$

where c and γ are positive constants

- 伽瑪矯正的擴充形式 $\boldsymbol{s = c(r + \varepsilon)^{\gamma}}$，增加位移參數
- 解決相片過度曝光或曝光不足
- 輸入與輸出範圍(0~255)
- $\boldsymbol{c}$值隨著 $\boldsymbol{\gamma}$ 值調整，若 $\boldsymbol{\gamma}$=2.0
- $\boldsymbol{r = 0, s = 0 \Longrightarrow 0 = c \cdot 0^2 \Longrightarrow c = 0}$
- $\boldsymbol{r = 255, s = 255 \Longrightarrow 255 = c \cdot 255^2 \Longrightarrow c = \frac{1}{255}}$

FIGURE 3.6
Plots of the gamma equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). Each curve was scaled *independently* so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.



Check IP03.ipynb

- 3.1 Gamma function plot

原始影像　　$\gamma = 0.5$

$\gamma = 0.2$　　$\gamma = 0.1$

圖 5-3　伽瑪矯正 ($\gamma < 1$)

Check IP03.ipynb

- 3.2 Gamma Correction
  - Please use for loop to calculate Gamma correction of an image.

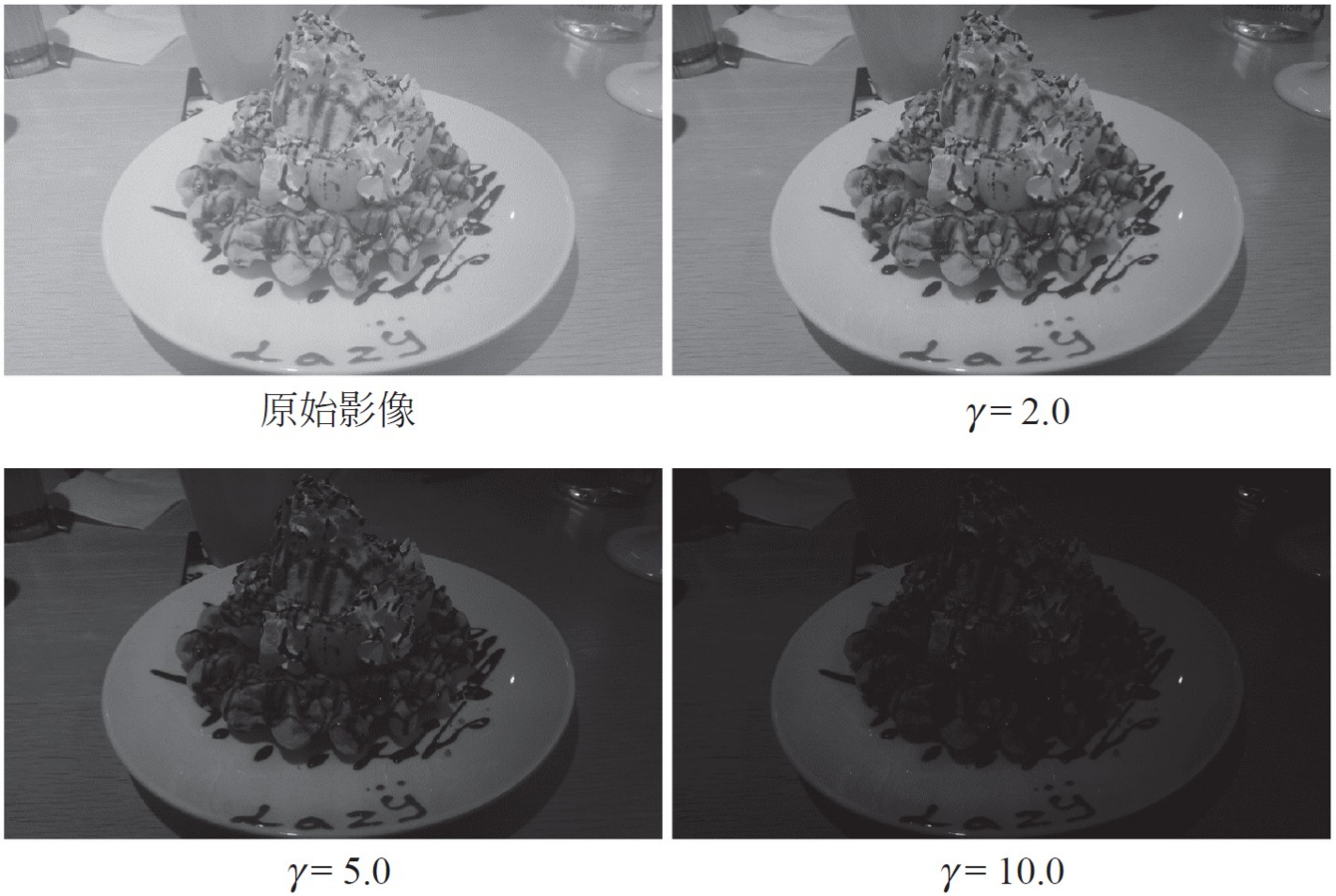
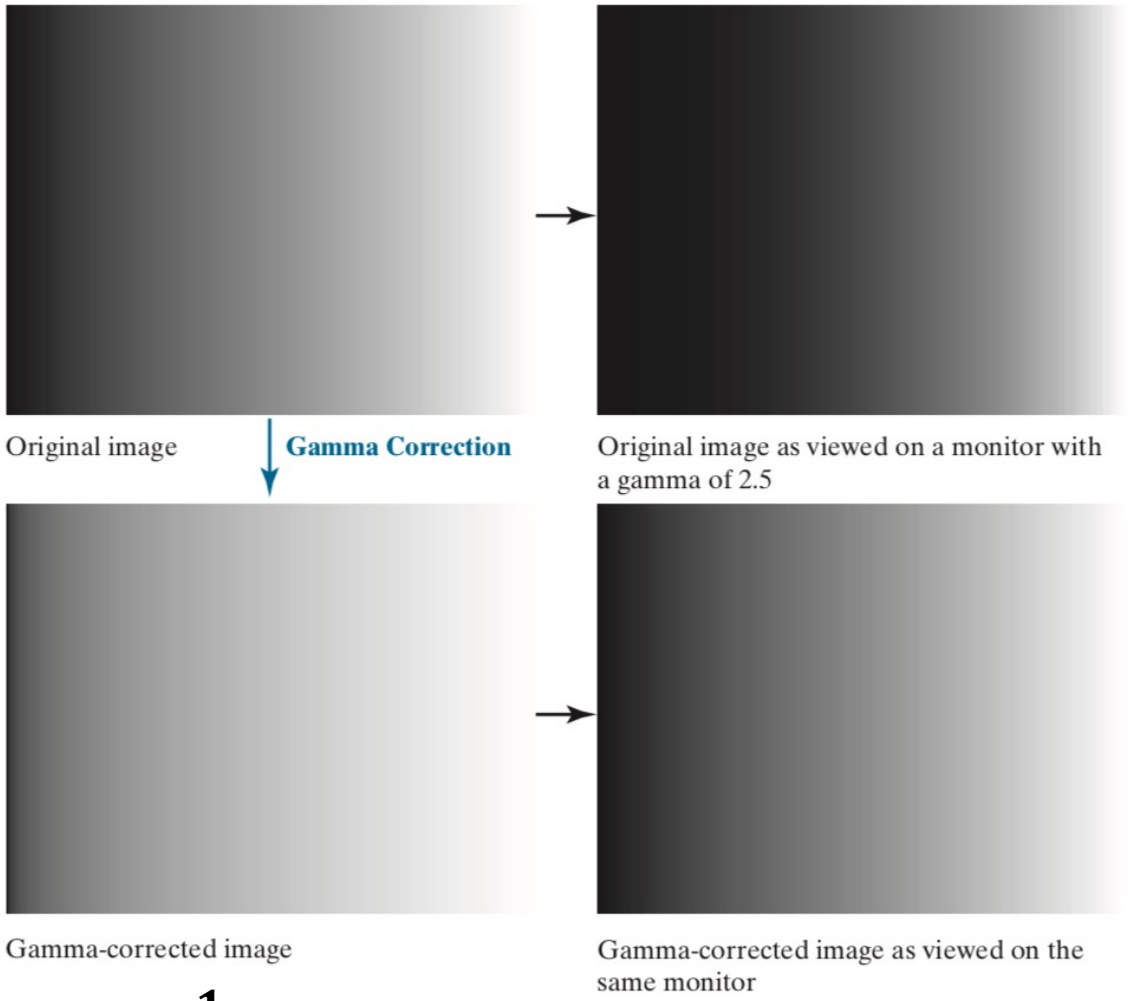


圖 5-4 伽瑪矯正 ($\gamma > 1$)

a b
c d

FIGURE 3.7
(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).



$$\gamma = 2.5$$

$$\gamma = \frac{1}{2.5}$$

練習

- 伽瑪矯正：$s = c \cdot r^{\gamma}$

(a)若$\gamma = 0.5$，c =?

$$r = 0 \Rightarrow s = c \cdot r^{\gamma} = c \cdot (0)^{0.5} = 0$$

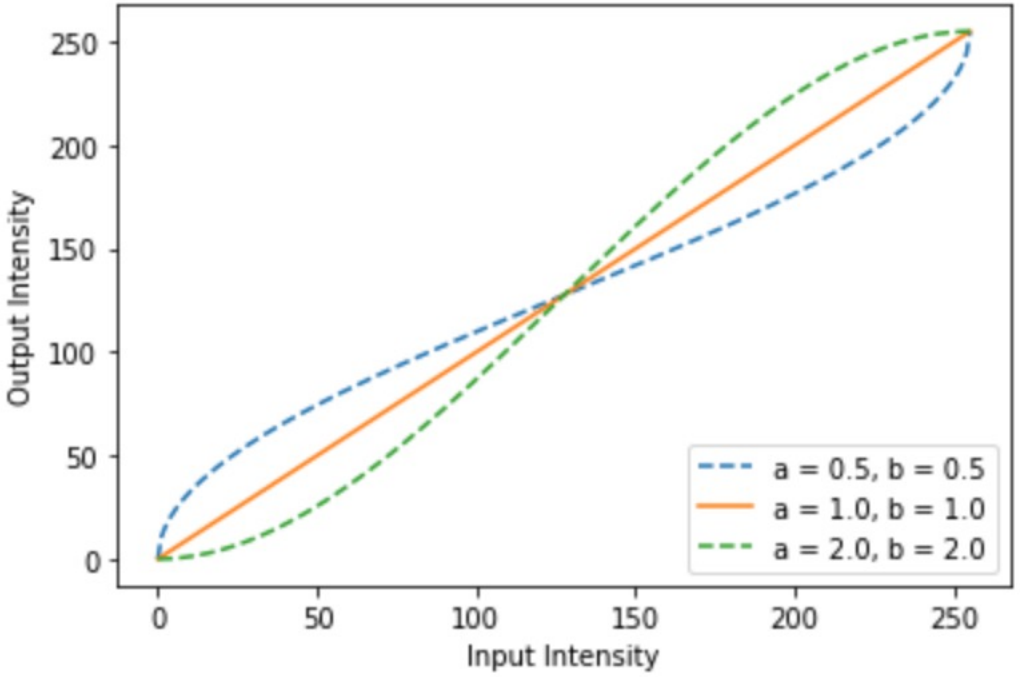
$$r = 255 \Rightarrow s = c \cdot r^{\gamma} = c \cdot (255)^{0.5} = 255 \Rightarrow c = \frac{255}{\sqrt{255}} \approx 15.9687$$

(b)當$r = 100$，s =?

$$s = c \cdot r^{\gamma} = \frac{255}{\sqrt{255}} \cdot (100)^{0.5} = 159.687 \approx 160 \text{ (四捨五入)。}$$

Beta函數

- 影像對比
- 暗的地方是否夠暗、亮的地方是否夠亮
- 不完整Beta函數 (Incomplete Beta Function)
- $\boldsymbol{\beta(x;a,b) = \int_0^x t^{a-1}(1-t)^{b-1}dt}$



原始影像　　$a = 0.5$、$b = 0.5$　　$a = 2.0$、$b = 2.0$

圖 5-6　Beta 矯正

- $\beta(x; a, b) = \int_0^x t^{a-1}(1-t)^{b-1}dt$
- 若a,b<1，減弱影像的對比
- 若a,b>1，增強影像的對比
- a,b值愈小(或愈大)，對比調整愈明顯

Check IP03.ipynb

- 3.3 Beta Function
- 3.4 Beta Correction
  - Please use for loop to calculate Beta correction of an image.

直方圖處理

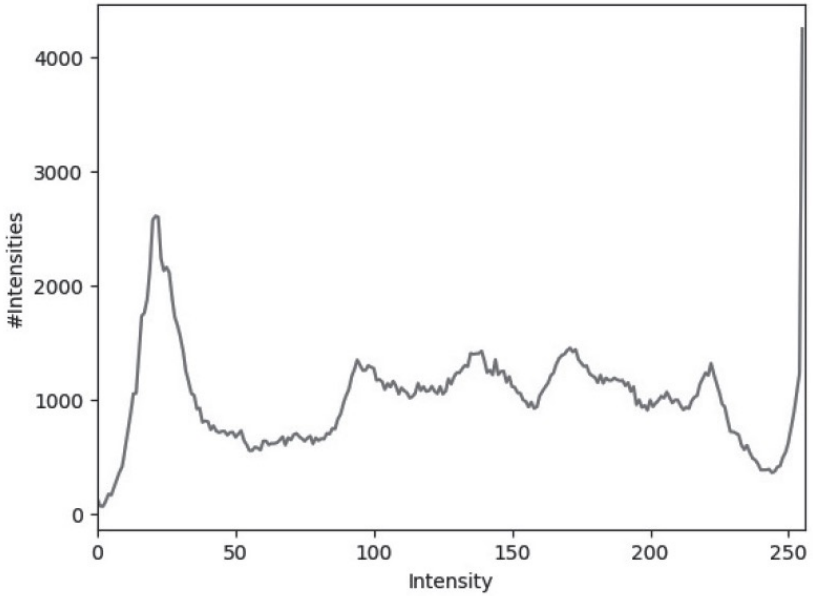
定義 直方圖

數位影像的**直方圖** (Histogram) 可以定義爲：

$$h(r_k) = n_k$$

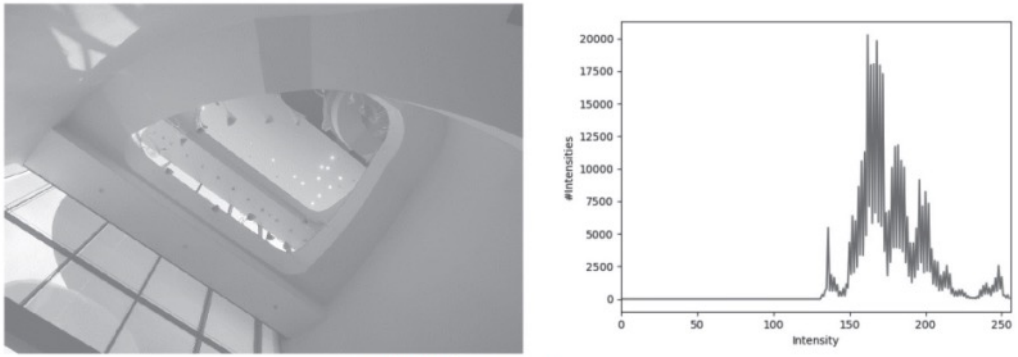
其中，r_k 爲第 k 個強度值，n_k 爲強度 r_k 的像素個數。

原始影像

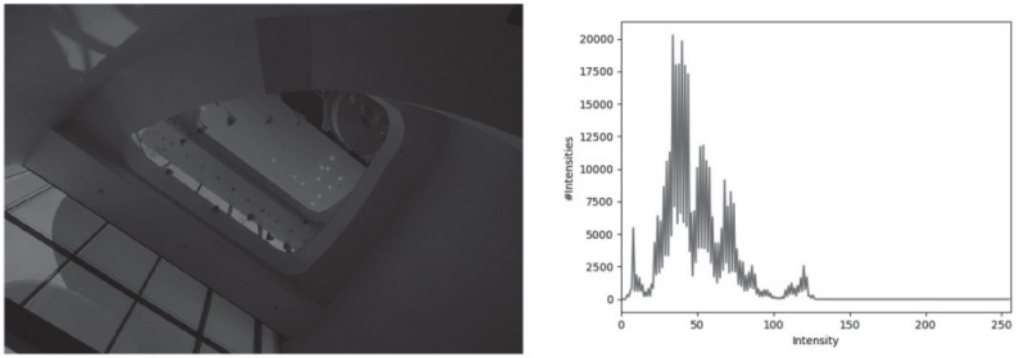


直方圖

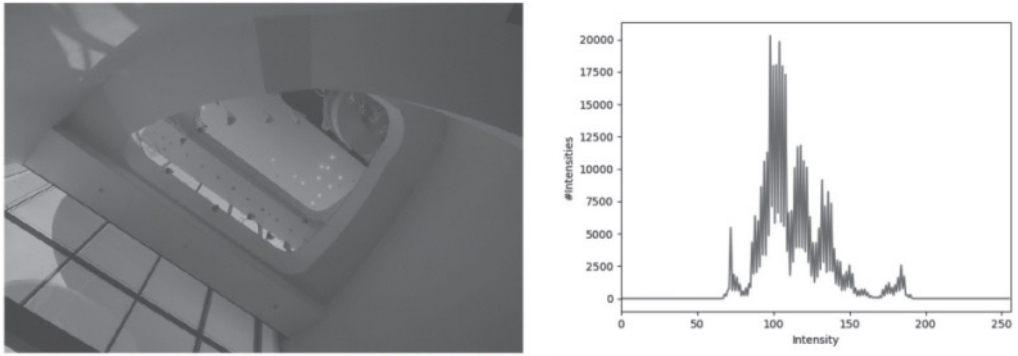
圖 5-7　Lenna 影像與直方圖



過度曝光 (Over Exposure)



曝光不足 (Under Exposure)

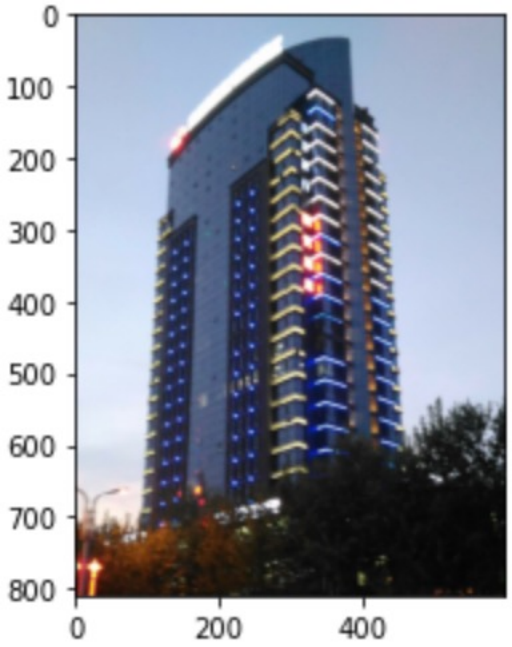


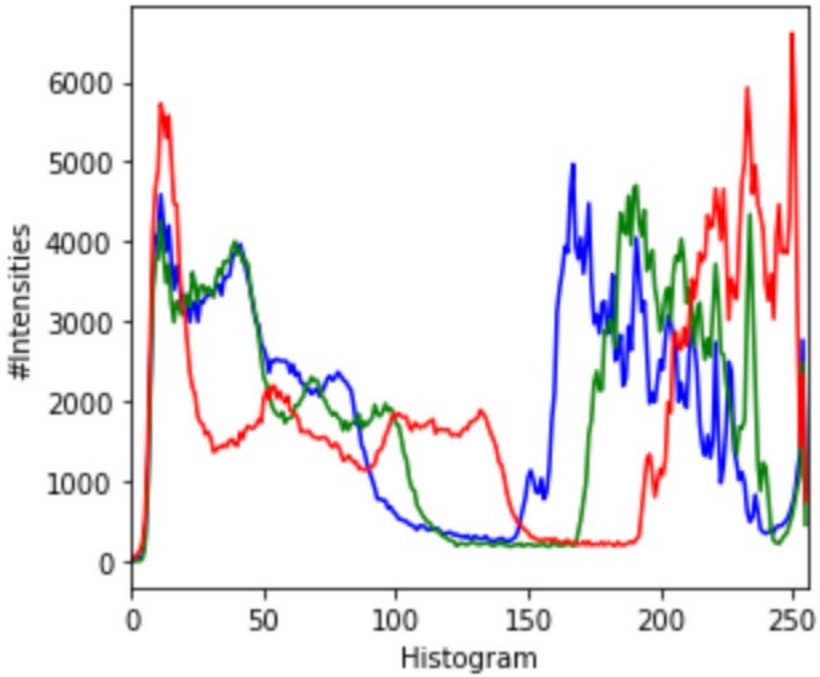
低對比 (Low Contrast)

圖 5-8　拍攝條件與直方圖

Check IP03.ipynb

- 3.5 Histogram





Check IP03.ipynb

- 3.6 Histogram
  - Subprogram returns image Histogram, and display Histogram in main()

Check IP03.ipynb

- 3.7 Image Histogram
  - Finish image histogram subprogram with following requirements:
  - 1. Use for-loop (no OpenCV) to calculate Histogram from pixel colors
  - 2. Subprogram returns image Histogram
  - 3. Display Histogram in main()

過度曝光　　直方圖等化

曝光不足　　直方圖等化

低對比　　直方圖等化

圖 5-9　直方圖等化

TABLE 3.1
Intensity distribution and histogram values for a 3-bit, 64×64 digital image.

r_k	n_k	$p_r(r_k) = n_k / MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

EXAMPLE 3.5: Illustration of the mechanics of histogram equalization.

It will be helpful to work through a simple example. Suppose that a 3-bit image ($L = 8$) of size 64×64 pixels ($MN = 4096$) has the intensity distribution in Table 3.1, where the intensity levels are integers in the range $[0, L-1] = [0, 7]$. The histogram of this image is sketched in Fig. 3.19(a). Values of the histogram equalization transformation function are obtained using Eq. (3-15). For instance,

$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33$$

Similarly, $s_1 = T(r_1) = 3.08$, $s_2 = 4.55$, $s_3 = 5.67$, $s_4 = 6.23$, $s_5 = 6.65$, $s_6 = 6.86$, and $s_7 = 7.00$. This transformation function has the staircase shape shown in Fig. 3.19(b).

At this point, the s values are fractional because they were generated by summing probability values, so we round them to their nearest integer values in the range $[0, 7]$:

$$s_0 = 1.33 \rightarrow 1 \quad s_2 = 4.55 \rightarrow 5 \quad s_4 = 6.23 \rightarrow 6 \quad s_6 = 6.86 \rightarrow 7$$
$$s_1 = 3.08 \rightarrow 3 \quad s_3 = 5.67 \rightarrow 6 \quad s_5 = 6.65 \rightarrow 7 \quad s_7 = 7.00 \rightarrow 7$$

CDF
Cumulative Density Function
計算累積密度函數

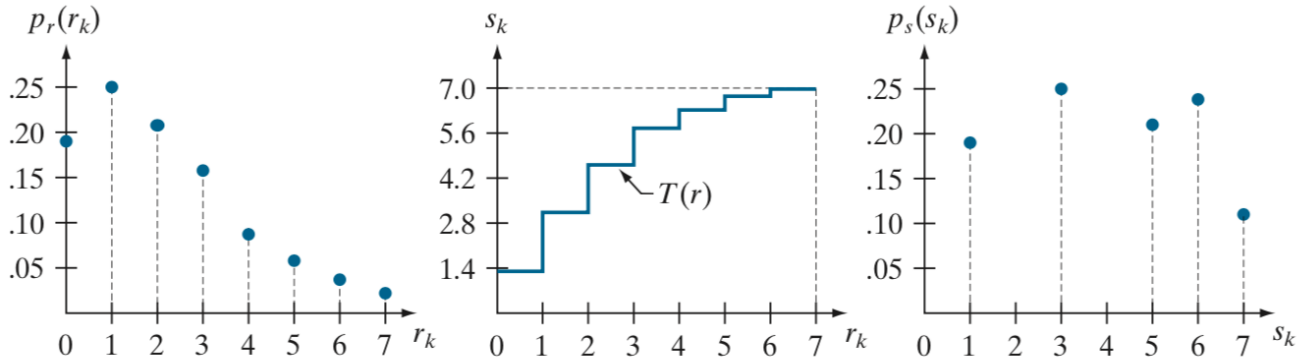


FIGURE 3.19 Histogram equalization. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

PDF
Probability Density Function
機率密度函數
↑
Input Histogram

CDF_equlization =
CDF*255/CDF.max()

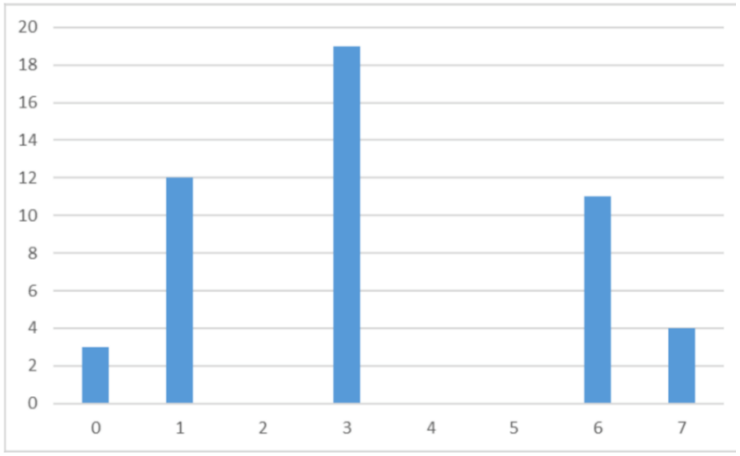
↓
Output Histogram

練習

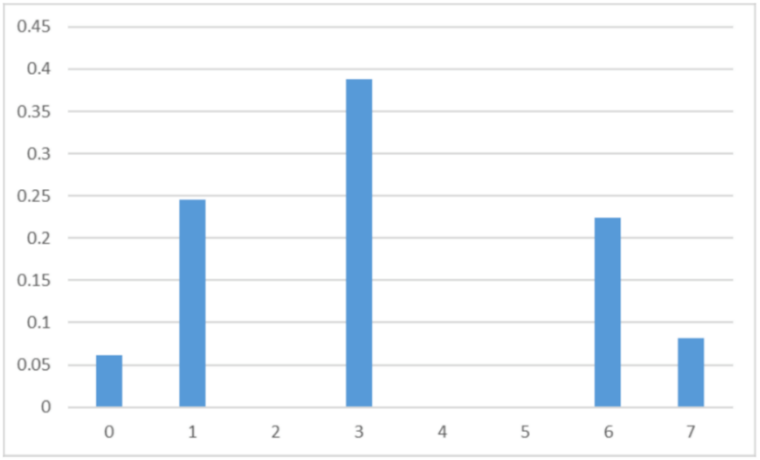
0	0	1	1	1	3	3
0	1	1	1	3	3	3
1	1	1	3	3	3	6
1	1	3	3	3	6	6
1	3	3	3	6	6	6
3	3	3	6	6	7	7
3	3	6	6	6	7	7

- 回答下列問題
- **(a) 計算直方圖**
- **(b) 計算PDF**
- **(c) 計算CDF**
- **(d) 計算直方圖等化的結果**

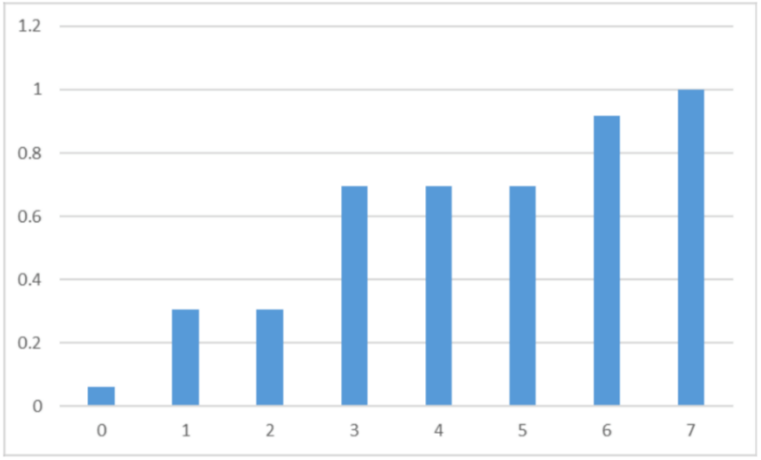
直方圖



) PDF



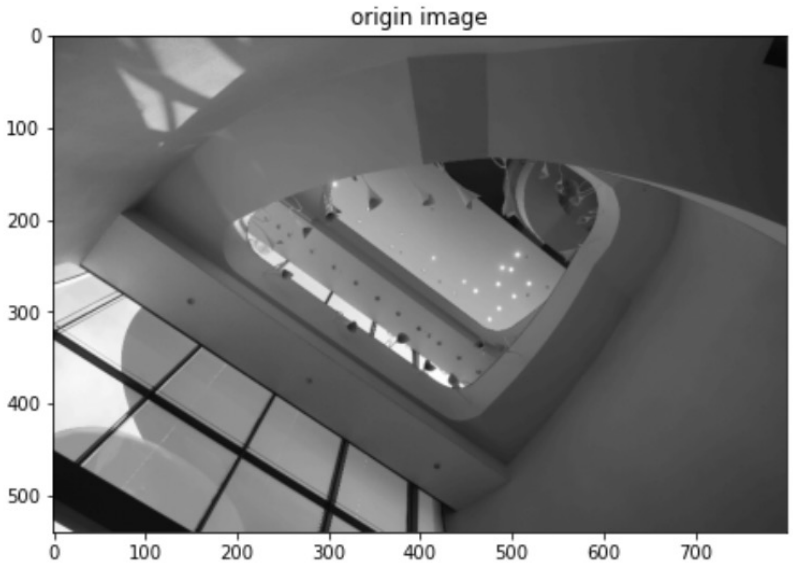
CDF

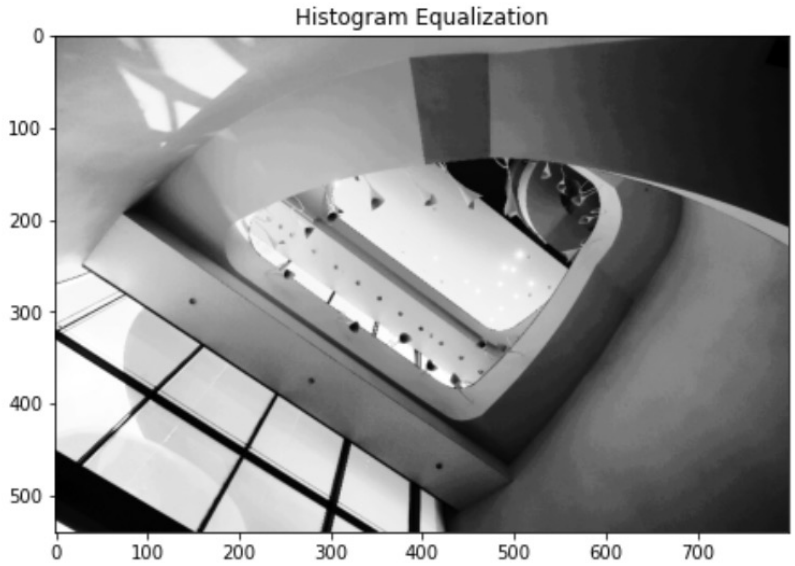


r_k	s_k
0	$0.428 \rightarrow 0$
1	$2.143 \rightarrow 2$
2	$2.143 \rightarrow 2$
3	$4.857 \rightarrow 5$
4	$4.857 \rightarrow 5$
5	$4.857 \rightarrow 5$
6	$6.429 \rightarrow 6$
7	$7 \rightarrow 7$

Check IP03.ipynb

3.8 Histogram Equalization





Check IP03.ipynb

- 3.9 Homework Image Equalization
  - Finish image equalization (myEqualization) by your histogram subprogram with following requirements:
  - 1. Use myHistogram in 3.7 calculate image Histogram
  - 2. cdf can be done by using np.cumsum()
  - 3. Subprogram returns cdf, cdf_equalization, andf_equ4
  - . Display image, cdf, cdf_equlization, quantized image in main()

影像濾波 Image Filtering

- 影像濾波
- Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbors.
- **線性濾波**
- If the operation performed on the image pixels is linear, then the filter is called a *linear spatial filter*.
- **非線性濾波**
- Otherwise, the filter is a *nonlinear spatial filter*.

Figure 3.28 illustrates the mechanics of linear spatial filtering using a 3×3 kernel. At any point (x, y) in the image, the response, $g(x, y)$, of the filter is the sum of products of the kernel coefficients and the image pixels encompassed by the kernel:

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + \ldots + w(0,0)f(x,y) + \ldots + w(1,1)f(x+1,y+1) \quad (3\text{-}30)$$

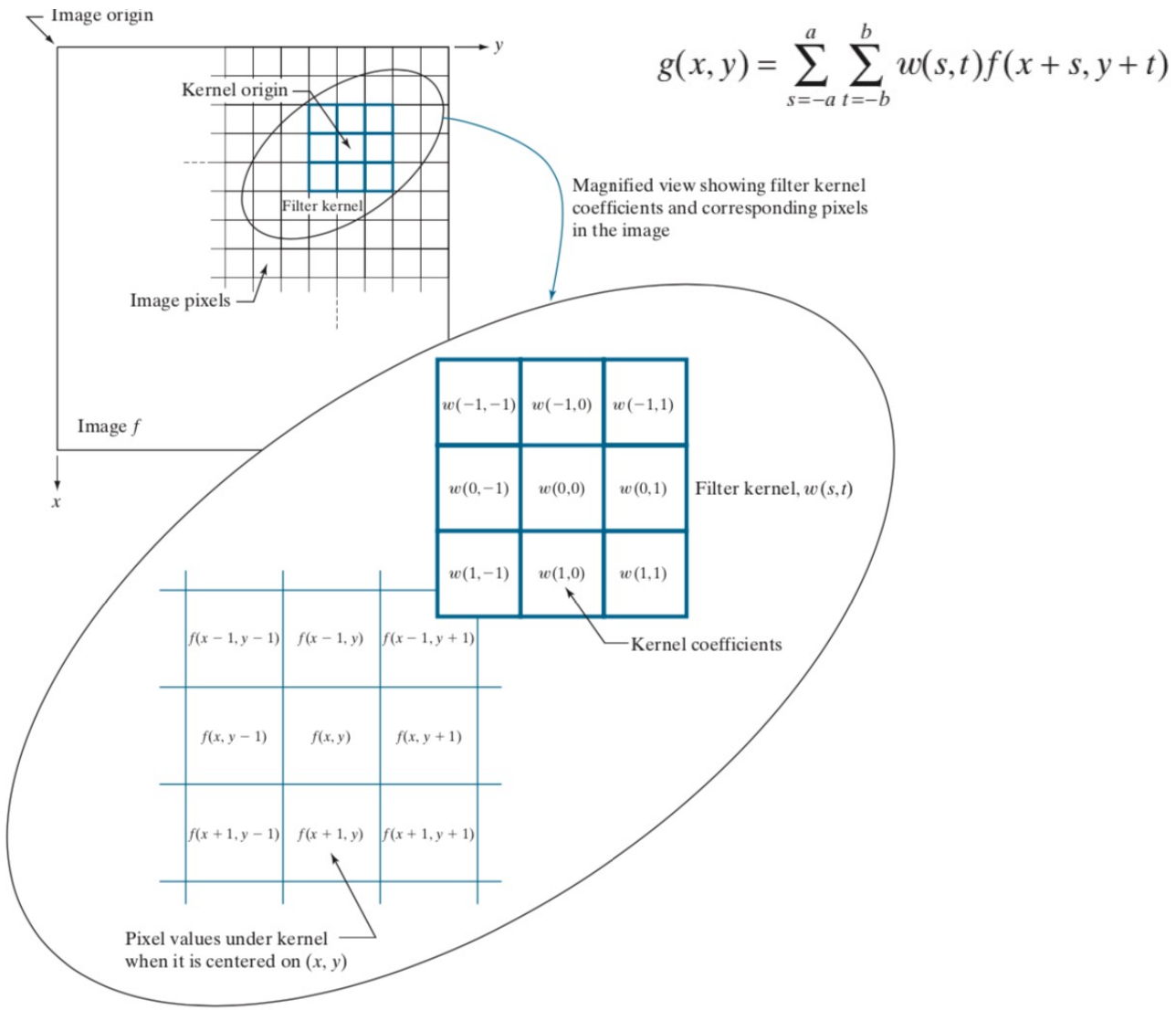


FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 kernel. The pixels are shown as squares to simplify the graphics. Note that the origin of the image is at the top left, but the origin of the kernel is at its center. Placing the origin at the center of spatially symmetric kernels simplifies writing expressions for linear filtering.

定義 **卷積**

給定兩個函數 $x(t)$ 與 $h(t)$ ，則**卷積** (Convolution) 可以定義爲：

$$y(t)=\int_{-\infty}^{\infty} x(\tau)\cdot h(t-\tau)\,d\tau=\int_{-\infty}^{\infty} h(\tau)\cdot x(t-\tau)\,d\tau$$

或

$$y(t)=x(t)*h(t)$$

其中，星號 * 爲卷積運算符號。

定義 **卷積**

給定數位訊號 $x[n]$ 與 $h[n]$，則**卷積** (Convolution) 可以定義爲：

$$y[n]=\sum_{k=-\infty}^{\infty} x[k]\cdot h[n-k]=\sum_{k=-\infty}^{\infty} h[k]\cdot x[n-k]$$

或

$$y[n]=x[n]*h[n]$$

其中，星號 * 爲卷積運算符號。

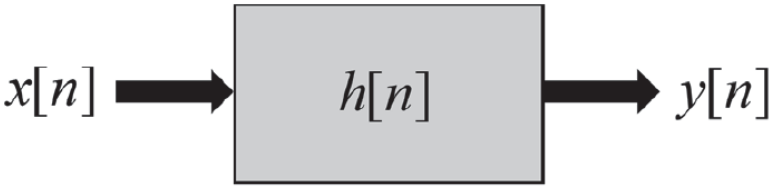


圖 5-10　離散時間域的卷積運算

$$x = \{1, 2, 4, 3, 2, 1, 1\}, n = 0, 1, ..., 6$$

$$h = \{1, 2, 3, 1, 1\}, n = 0, 1, 2, 3, 4$$

=

x	h
0 0 0 0 1 2 4 3 2 1 1 0 0 0 0	1 2 3 1 1

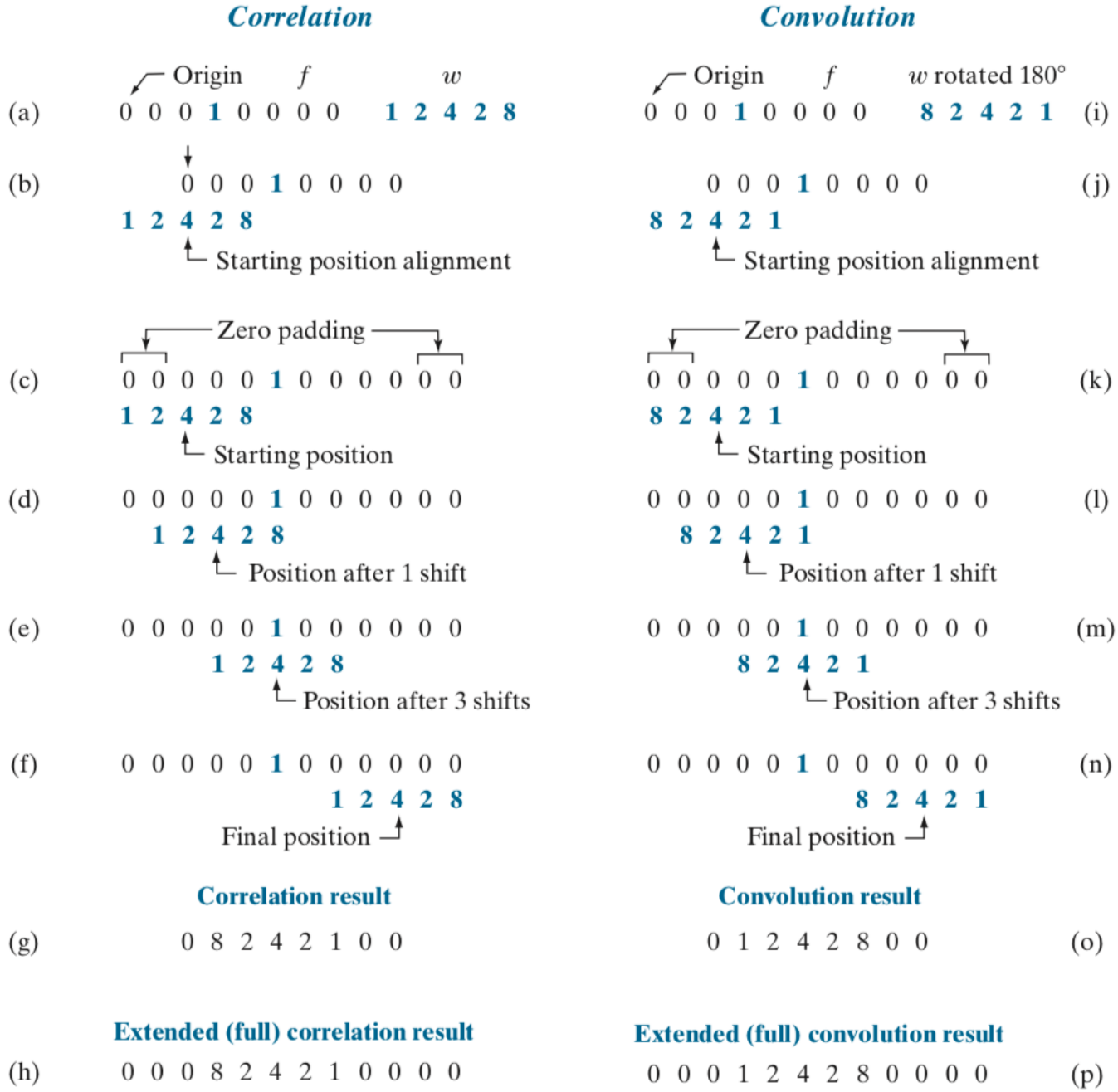
x	h
0 0 0 0 1 2 4 3 2 1 1 0 0 0 0	1 2 3 1 1
1 1 3 2 1	

x	h
0 0 0 0 1 2 4 3 2 1 1 0 0 0 0	1 2 3 1 1
1 1 3 2 1	

n	0	1	2	3	4	5	6	7	8	9	10
$y[n]$	1	4	11	18	23	20	16	10	6	2	1

n	0	1	2	3	4	5	6
$x[n]$	1	2	4	3	2	1	1
$y[n]$	11	18	23	20	16	10	6

FIGURE 3.29
Illustration of 1-D correlation and convolution of a kernel, w, with a function f consisting of a discrete unit impulse. Note that correlation and convolution are functions of the variable x, which acts to *displace* one function with respect to the other. For the extended correlation and convolution results, the starting configuration places the right-most element of the kernel to be coincident with the origin of f. Additional padding must be used.



Check IP03.ipynb

- 3.10 1-D Convolution

定義 卷積 (2D)

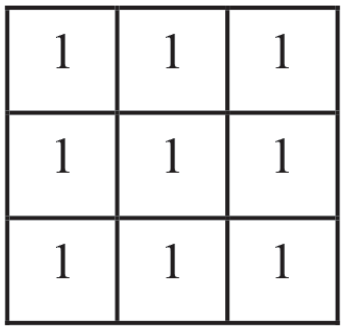
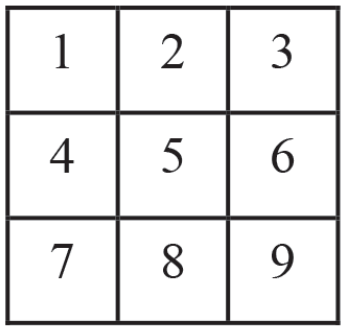
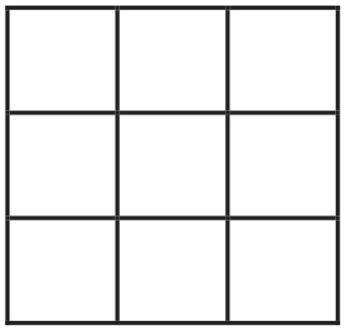
二維的**卷積** (Convolution) 可以定義爲：

$$g(x,y) = f(x,y) * h(x,y)$$

或

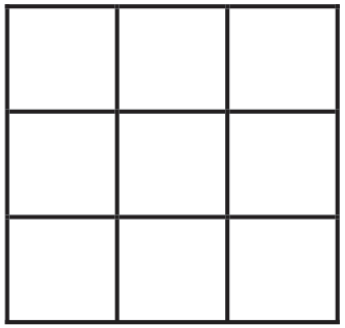
$$g(x,y) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} h(s,t) \cdot f(x-s, y-t)$$

其中，星號 * 爲卷積運算符號。

$f(x, y)$	$h(x, y)$	$g(x, y)$
		

$f(x,y)$	$h(x,y)$	$g(x,y)$
0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	1 2 3 4 5 6 7 8 9	

$f(x,y)$	$h(x,y)$	$g(x,y)$
0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	9 8 7 6 5 4 3 2 1	



$f(x,y)$	$h(x,y)$	$g(x,y)$
0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	9 8 7 6 5 4 3 2 1	12

$f(x,y)$	$h(x,y)$	$g(x,y)$
0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	9 8 7 6 5 4 3 2 1	12 21

$f(x,y)$	$h(x,y)$	$g(x,y)$
0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 0 0 0	9 8 7 6 5 4 3 2 1	12 21 16 27 45 33 24 39 28

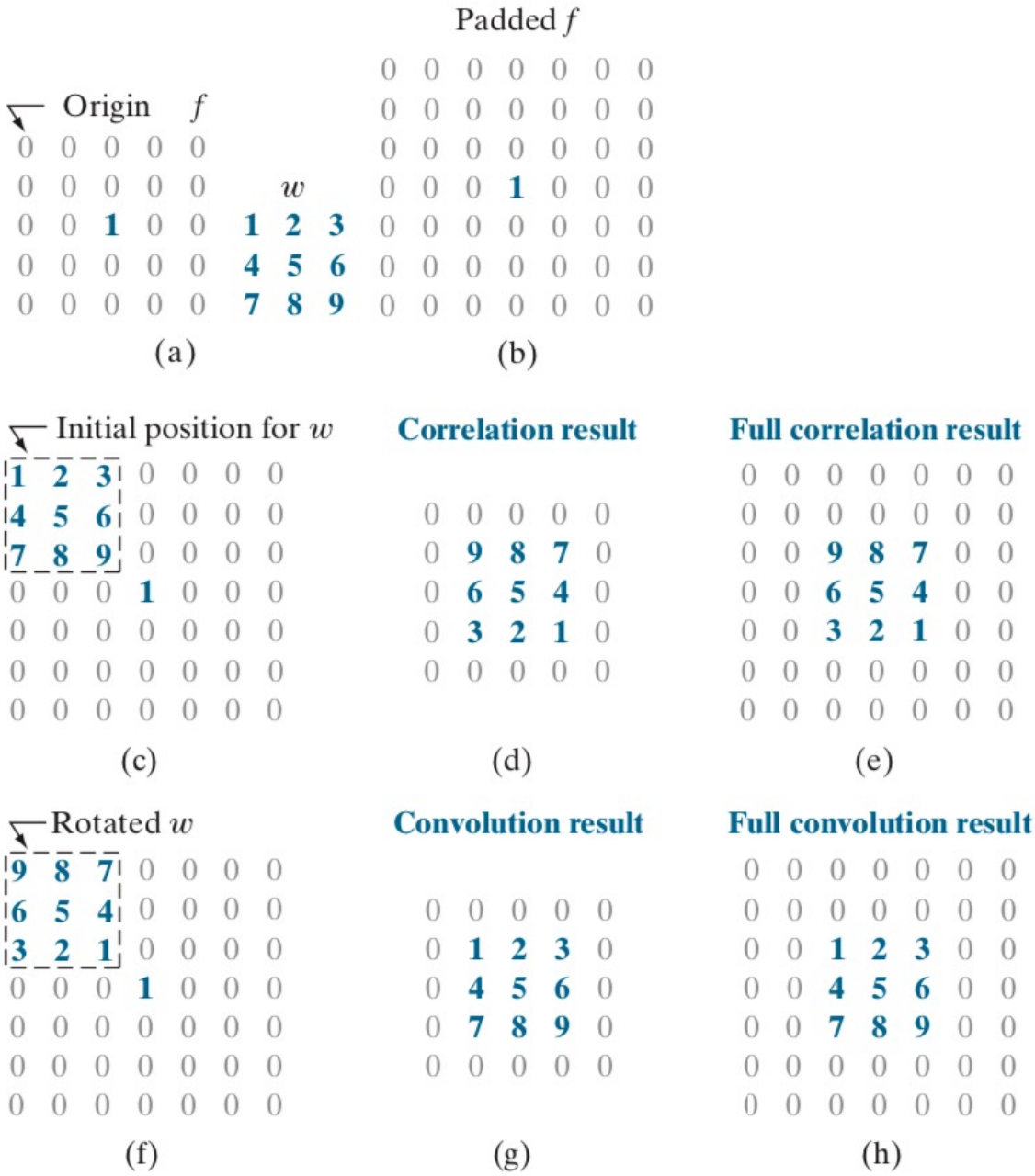


FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D kernel with an image consisting of a discrete unit impulse. The 0's are shown in gray to simplify visual analysis. Note that correlation and convolution are functions of x and y. As these variable change, they *displace* one function with respect to the other. See the discussion of Eqs. (3-36) and (3-37) regarding full correlation and convolution.

Check IP03.ipynb

- 3.11 2-D Convolution

練習

- 假設數位影像和濾波器如下，試求二維卷積的結果

1	2	3
1	2	3
1	2	3

$f(x, y)$

1	1	1
1	2	1
1	1	1

$h(x, y)$

$g(x, y)$

7	14	13
10	20	18
7	14	13

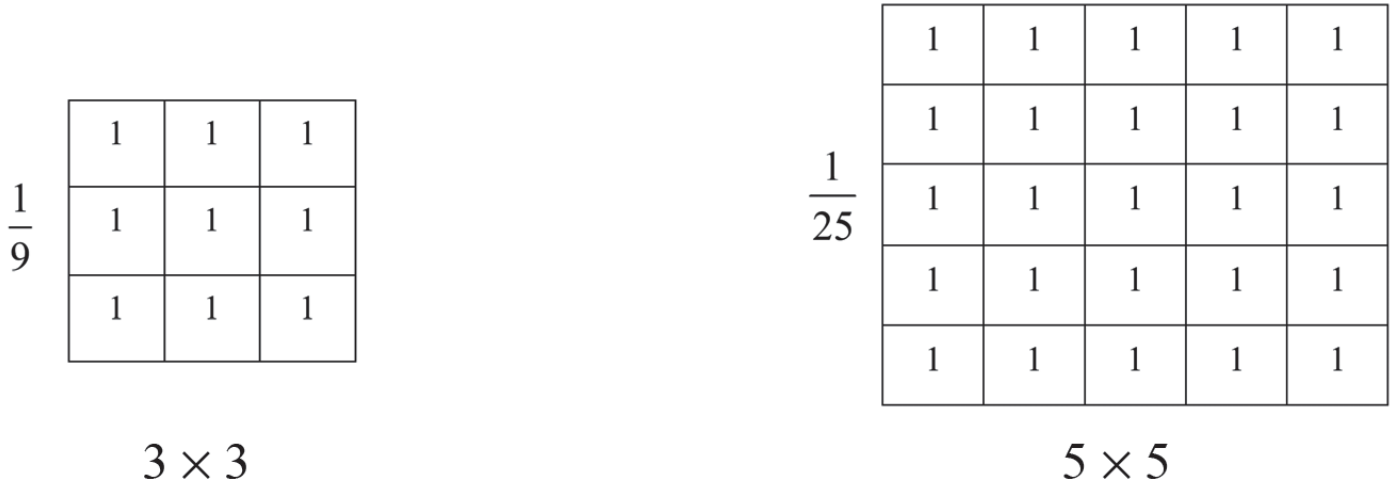


圖 5-11　典型的平均濾波器

- **平均濾波器(Average Filtering)**
- 最簡單的濾波器
- 模糊化處理(Blur)
- 也稱為Box filter

圖 5-12　平均濾波

Check IP03.ipynb

- 3.12 Average Filtering

Check IP03.ipynb

- 3.13 Please write an Average Filter without using OpenCV function
  - input parameters including a (height) and b (width)

定義 高斯函數 (2D)

高斯函數 (Gaussian Function) 可以定義爲：

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

其中，σ 稱爲標準差。

- 濾波器大小通常由標準差決定
- **比較理想的選擇是$[-3\sigma, 3\sigma]$**
- **若$\sigma = 1.0$，濾波器範圍為$[-3, 3]$，濾波器大小為7×7**
- **相同大小的濾波器，高斯濾波器保留的影像資訊會比平均濾波器多**

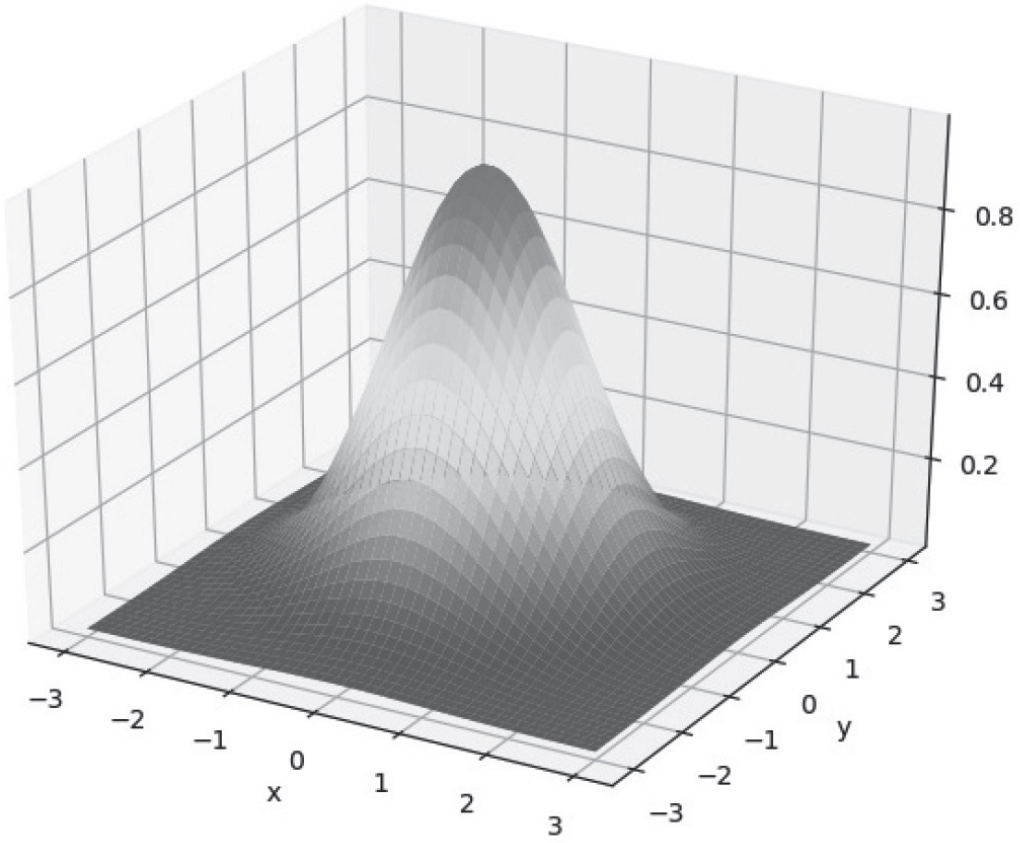


圖 5-13　典型的高斯函數 (標準差 $\sigma=1$)

原始影像　　5 × 5

11 × 11　　21 × 21

圖 5-14　高斯濾波

Check IP03.ipynb

- 3.14 Gaussian function
  - OpenCV 提供的高斯濾波器稱為 GaussianBlur
  - 會依據濾波器的大小，自動計算標準差

$$w(s,t) = G(s,t) = Ke^{-\frac{s^2 + t^2}{2\sigma^2}}$$

By letting $r = [s^2 + t^2]^{1/2}$ we can write Eq. (3-45) as

$$G(r) = Ke^{-\frac{r^2}{2\sigma^2}}$$

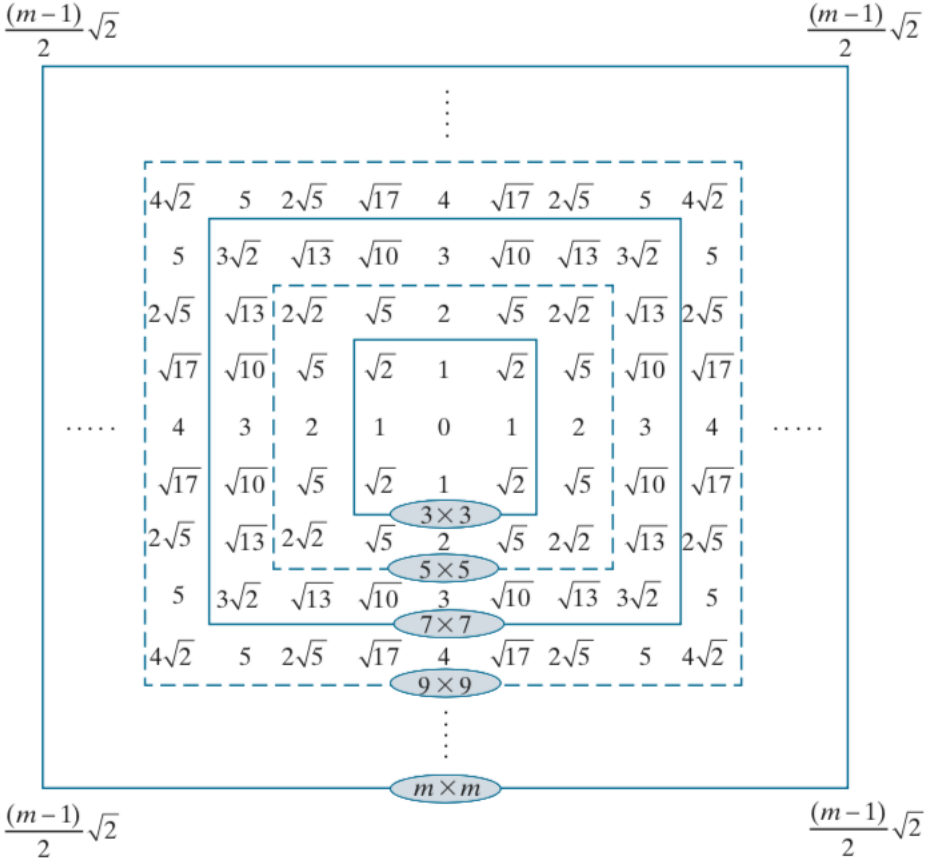
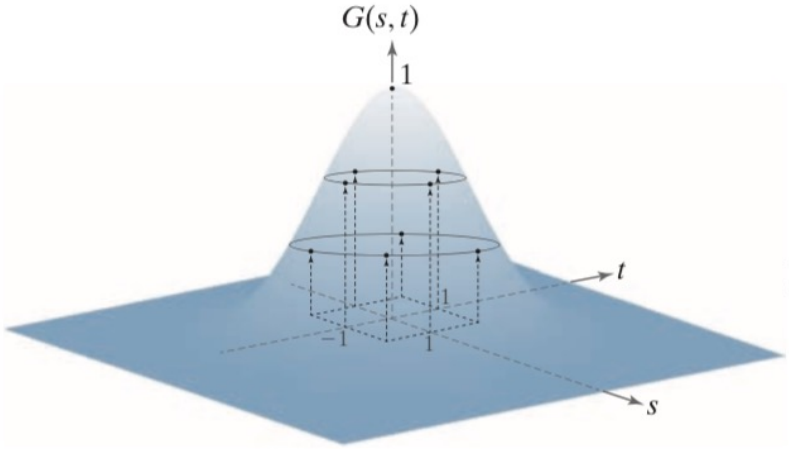


FIGURE 3.34 Distances from the center for various sizes of square kernels.

a b

FIGURE 3.35
(a) Sampling a Gaussian function to obtain a discrete Gaussian kernel. The values shown are for $K = 1$ and $\sigma = 1$. (b) Resulting 3×3 kernel [this is the same as Fig. 3.31(b)].



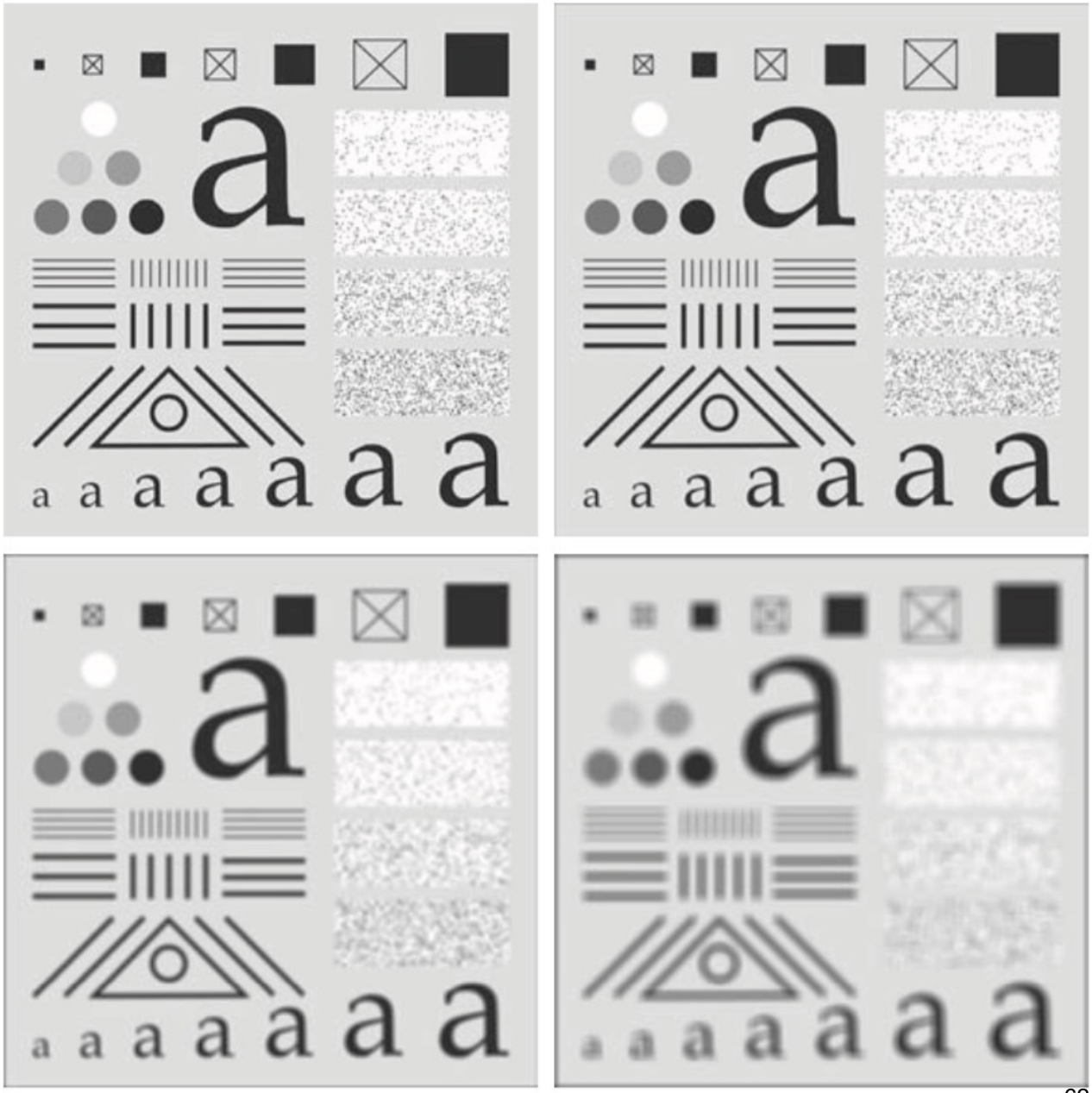
$$\frac{1}{4.8976} \times \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1.0000 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix}$$

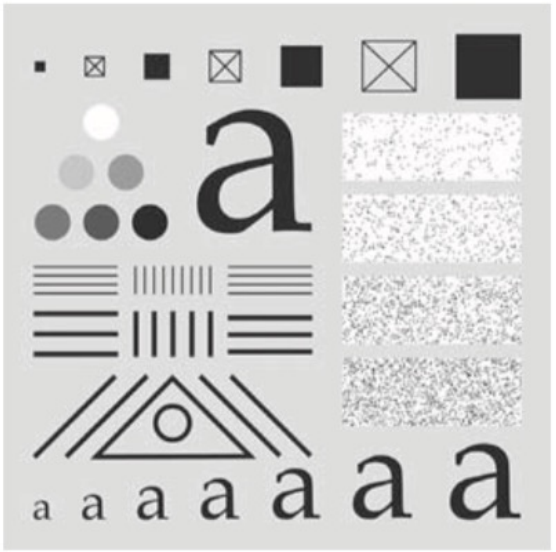
- **Box filtering:**
- Advan.
- Simple
- suitable for quick experimentation
- often yield smoothing results that are visually acceptable
- limitations
- poor approximations to the lowpass blurring characteristics of lenses
- box filters favor blurring along perpendicular directions, images with a high level of detail, or with strong geometrical components, the directionality of box filters often produces undesirable results.

- **Gaussian filtering:**
- circularly symmetric
- isotropic
- response is independent of orientation

a b
c d

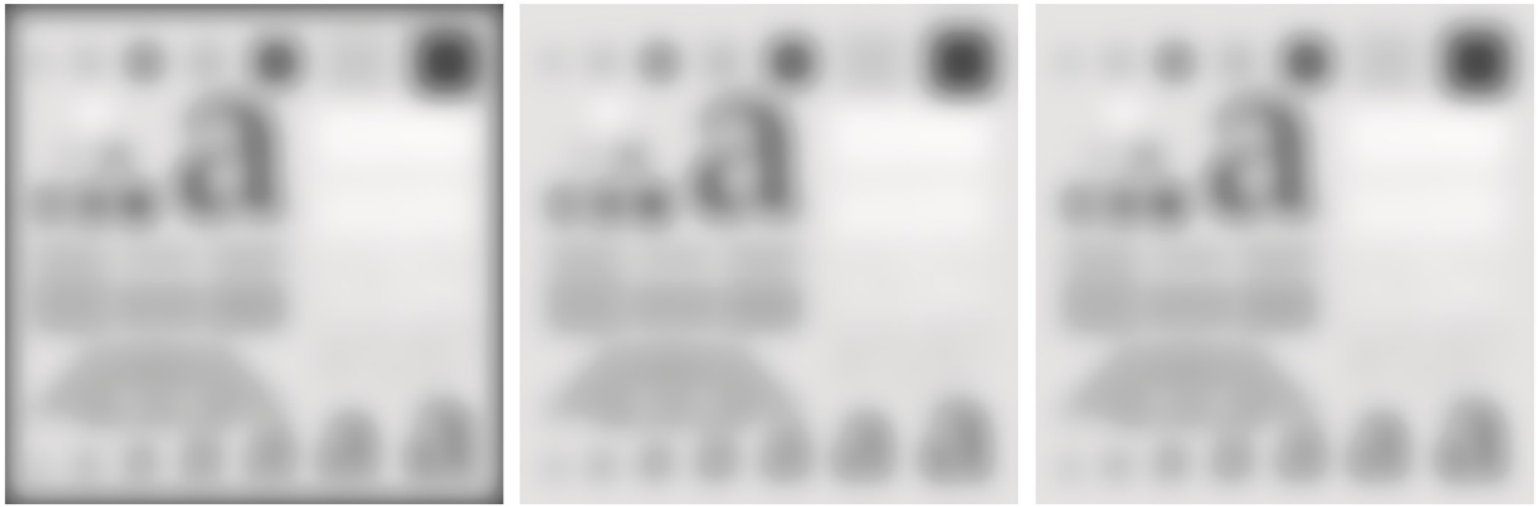
FIGURE 3.33
(a) Test pattern of size 1024×1024 pixels.
(b)-(d) Results of lowpass filtering with box kernels of sizes 3×3, 11×11, and 21×21, respectively.





a b c

FIGURE 3.36 (a)A test pattern of size 1024×1024. (b) Result of lowpass filtering the pattern with a Gaussian kernel of size 21×21, with standard deviations $\sigma = 3.5$. (c) Result of using a kernel of size 43×43, with $\sigma = 7$. This result is comparable to Fig. 3.33(d). We used $K = 1$ in all cases.



a b c

FIGURE 3.39 Result of filtering the test pattern in Fig. 3.36(a) using (a) zero padding, (b) mirror padding, and (c) replicate padding. A Gaussian kernel of size 187×187, with $K = 1$ and $\sigma = 31$ was used in all three cases.

定義 影像梯度

影像梯度 (Image Gradient) 可以定義爲：

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

其中，

$\frac{\partial f}{\partial x}$ 爲 x 方向的一階導函數；$\frac{\partial f}{\partial y}$ 爲 y 方向的一階導函數。

- 影像梯度大小(**Magnitiude**)
- $\boldsymbol{M(x,y) = \sqrt{g_x^2 + g_y^2}}$ 或
- $\boldsymbol{M(x,y) \approx |g_x| + |g_y|}$
- **梯度方向則為** $\boldsymbol{\theta = tan^{-1}\left(\frac{g_y}{g_x}\right)}$

$$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix} / 4$$

–1	0
0	1

0	–1
1	0

Roberts

–1	–1	–1
0	0	0
1	1	1

–1	0	1
–1	0	1
–1	0	1

Prewitt

–1	–2	–1
0	0	0
1	2	1

–1	0	1
–2	0	2
–1	0	1

Sobel

45^o Sobel

$$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

–1	–1	–1
1	–2	1
1	1	1

–1	1	1
–1	–2	1
–1	1	1

Robinson

–5	–5	–5
3	0	3
3	3	3

–5	3	3
–5	0	3
–5	3	3

Kirsch

圖 5-15　梯度運算子

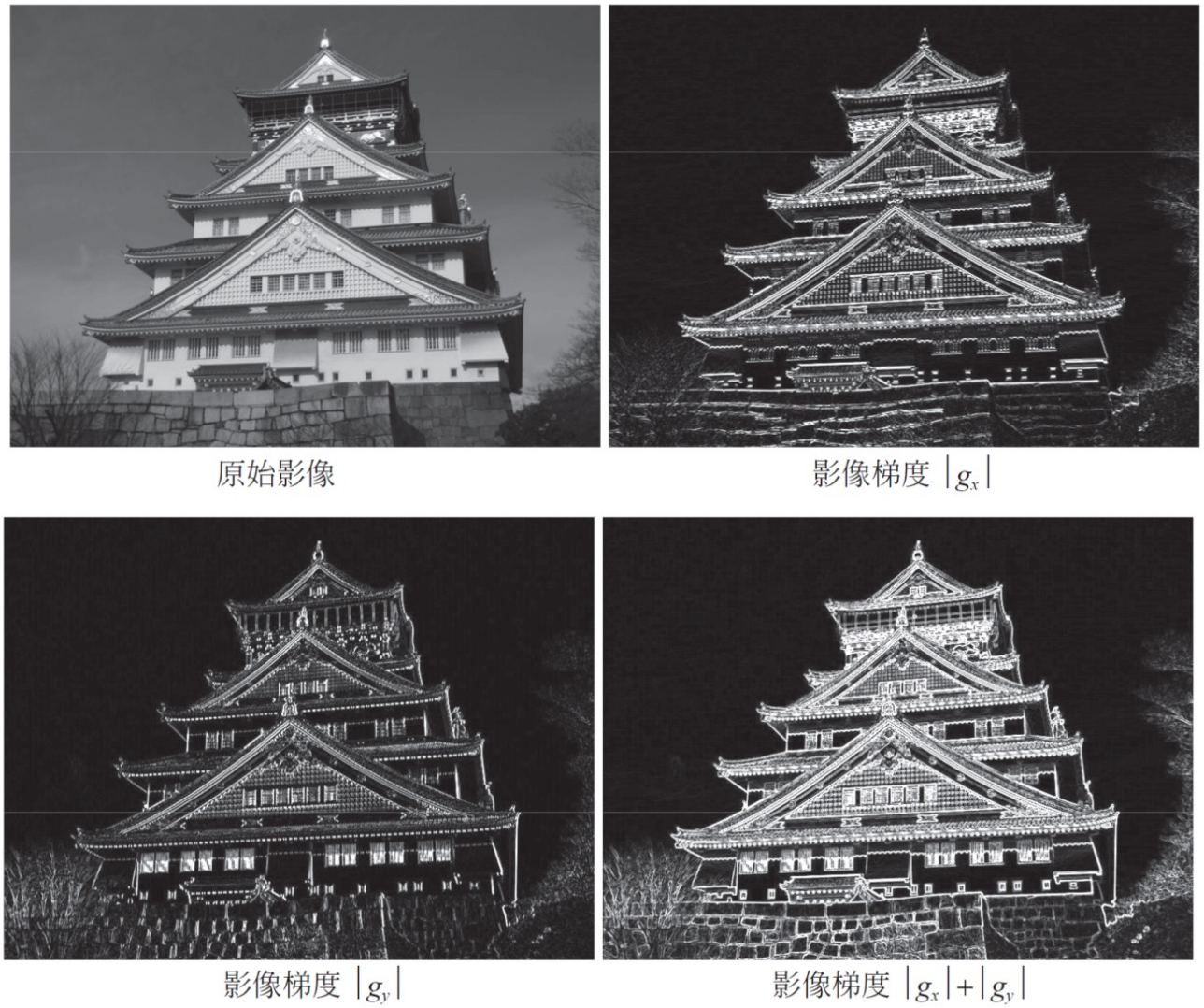


圖 5-16　影像梯度

練習

- 給定下列影像，試計算下列紅字的相關數值

1	1	3	3	3
1	1	3	3	3
1	3	3	3	3
3	3	3	6	6
3	3	6	6	7

g_x

–1	–2	–1
0	0	0
1	2	1

g_y

–1	0	1
–2	0	2
–1	0	1

Sobel

- (a) 計算 Sobel 濾波器的影像梯度$\boldsymbol{M(x,y)} \approx |\boldsymbol{g_x}| + |\boldsymbol{g_y}|$
- (b) 計算 Sobel 濾波器的影像梯度方向$\boldsymbol{\theta} = \tan^{-1}\frac{\boldsymbol{g_y}}{\boldsymbol{g_x}}$

Check IP03.ipynb

- 3.15 Image Gradient

- 3.16 Image Gradient Experiments
  - In function: Sobel_gradient, add two options
    - 1. direction = 4: calculate Square Root of sum of g_x^2 and g_y^2

$$g = \sqrt{g\_x^2} + \sqrt{g\_y^2}$$

    - 2. direction = 5: return theta

定義 **拉普拉斯運算子**

拉普拉斯運算子 (Laplacian) 可以定義爲：

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

其中，

$\frac{\partial^2 f}{\partial x^2}$ 爲 x 方向的二階導函數；$\frac{\partial^2 f}{\partial y^2}$ 爲 y 方向的二階導函數。

- 使用***Laplacian***找出邊緣
- 使用此函數除了傳入灰階影像外，
- 亦須指定輸出的影像浮點格式***CV_64F***或***CV_32F***

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y)$$

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

0	1	0
1	−4	1
0	1	0

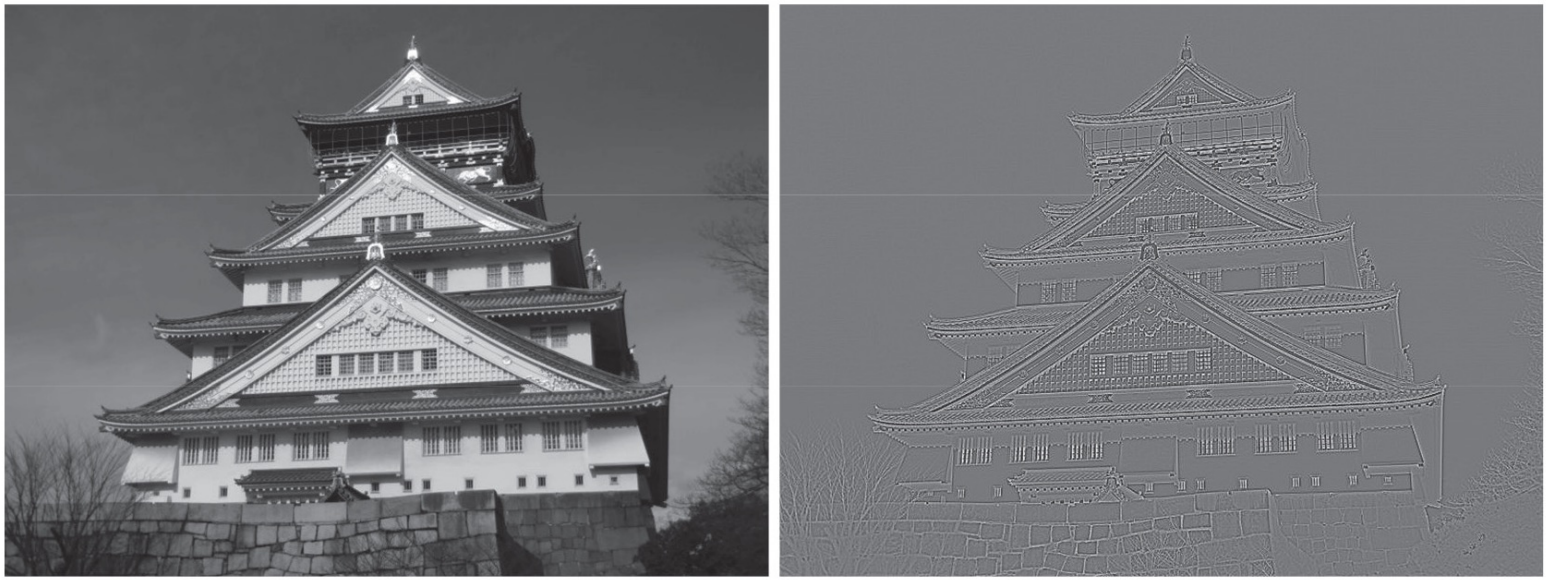
圖 5-17　拉普拉斯運算子

0	1	0
1	−4	1
0	1	0

Because the Laplacian is a derivative operator, it highlights sharp intensity transitions in an image and de-emphasizes regions of slowly varying intensities. This will tend to produce images that have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background. Background features can be "recovered" while still preserving the sharpening effect of the Laplacian by adding the Laplacian image to the original. As noted in the previous paragraph, it is important to keep in mind which definition of the Laplacian is used. If the definition used has a negative center coefficient, then we *subtract* the Laplacian image from the original to obtain a sharpened result. Thus, the basic way in which we use the Laplacian for image sharpening is

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)] \tag{3-63}$$

where $f(x, y)$ and $g(x, y)$ are the input and sharpened images, respectively. We let $c = -1$ if the Laplacian kernels in **Fig. 3.51(a)** or



原始影像　　影像銳化

圖 5-18　使用拉普拉斯運算子之影像銳化

Check IP03.ipynb

- 3.17 Laplacian

0	–1	0
–1	5	–1
0	–1	0

圖 5-19　混合拉普拉斯運算子

原始影像　　影像銳化

圖 5-20　使用混合拉普拉斯運算子的影像銳化

Check IP03.ipynb

0	−1	0
−1	5	−1
0	−1	0

圖 5-19　混合拉普拉斯運算子

- 3.18 Laplacian Sharpening
  - Finish the following Laplacian sharpening filters

原始影像

影像銳化

定義 非銳化遮罩

非銳化遮罩 (Unsharp Masking) 可以定義爲：

$$g(x,y) = f(x,y) + k \cdot g_{mask}(x,y)$$

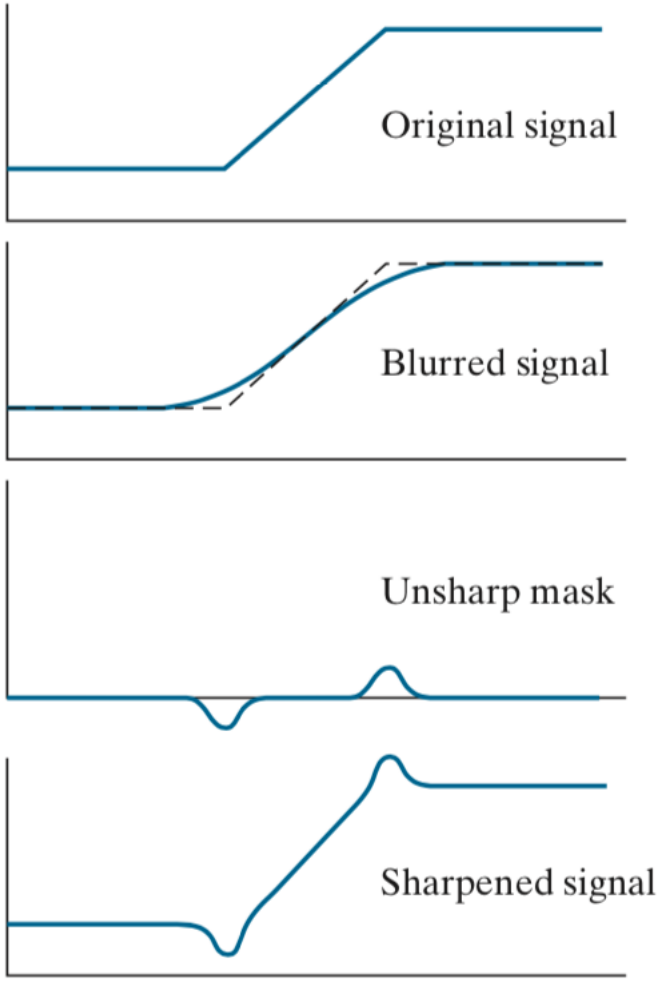
其中，

$$g_{\text{mask}}(x,y) = f(x,y) - \bar{f}(x,y)$$

稱爲**非銳化遮罩** (Unsharp Mask)，$\bar{f}(x,y)$ 爲對原始影像平滑化的結果。參數 k 是用來調整銳化的程度：$k = 1$ 時稱爲**非銳化遮罩** (Unsharp Masking)；$k > 1$ 稱爲**高增濾波** (Highboost Filtering)。

a
b
c
d

FIGURE 3.48
1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).



a b c
d e

FIGURE 3.49 (a) Original image of size 600×259 pixels. (b) Image blurred using a 31×31 Gaussian lowpass filter with $\sigma = 5$. (c) Mask. (d) Result of unsharp masking using Eq. (3-56) with $k = 1$. (e) Result of highboost filtering with $k = 4.5$.

$$g(x, y) = f(x, y) + k g_{\text{mask}}(x, y) \qquad (3\text{-}56)$$

Figure 3.49(e) shows the result of using Eq. (3-56) with $k = 4.5$.

原始影像

非銳化遮罩

圖 5-21　非銳化遮罩

Check IP03.ipynb

- 3.19 un-sharpening

- 3.20 Un-sharpening Experiments
  - Finish Un-sharpening function with the following form options:
  - 1. form = 1: (cv2.GaussianBlur) nxn Gaussian Blur filter
  - 2. form = 2: (cv2.blur) nxn average filter
  - 3. k: degree of un-sharpening mask

原始影像

高增濾波

圖 5-22　高增濾波

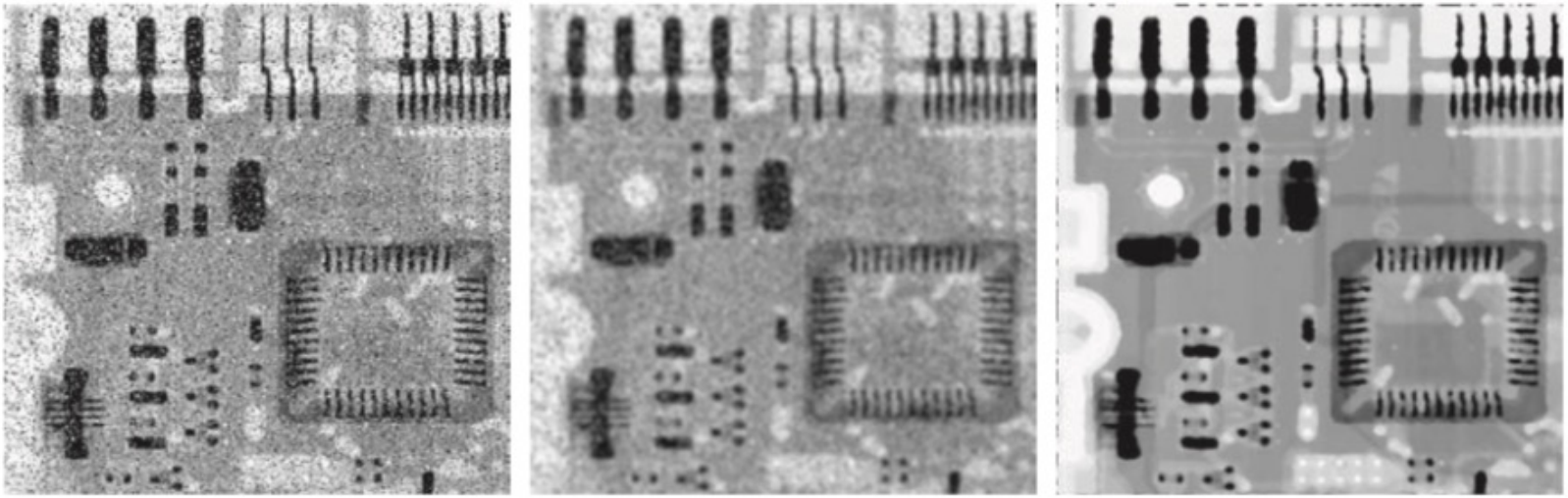
中值濾波器(Median Filter)

Median Filter

The best-known order-statistic filter in image processing is the *median filter*, which, as its name implies, replaces the value of a pixel by the median of the intensity levels in a predefined neighborhood of that pixel:

$$\hat{f}(x,y) = \underset{(r,c)\in S_{xy}}{\text{median}}\{g(r,c)\} \qquad (5\text{-}27)$$

where, as before, S_{xy} is a subimage (neighborhood) centered on point (x,y). The value of the pixel at (x,y) is included in the computation of the median. Median filters



a b c

FIGURE 3.43 (a) X-ray image of a circuit board, corrupted by salt-and-pepper noise. (b) Noise reduction using a 19×19 Gaussian lowpass filter kernel with $\sigma = 3$. (c) Noise reduction using a 7×7 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Check IP03.ipynb

- 3.21 salt-and-pepper noise
- 3.22 Median filter
  - Please design a Python program to acquire median filter results of 3.21