Chapter 4: การปรับปรุงคุณภาพของภาพในโดเมนระยะทาง (Image Enhancement in Spatial Domain)

เป้าหมายหลักของการปรับปรุงคุณภาพของภาพดิจิตอลนั้นคือการทำให้ภาพผลลัพธ์ที่ได้จากการปรับปรุงคุณภาพมีความเหมาะสมกว่าภาพต้นฉบับเพื่อที่จะนำไปใช้งานต่อในกระบวนการอื่นๆ ได้ ซึ่งในขั้นตอนของการปรับปรุงคุณภาพของภาพนั้นไม่ได้จำกัดจำนวนครั้งและเทคนิคที่จะนำมาใช้ อีกทั้งขั้นตอนในการปรับปรุงคุณภาพนั้นก็ไม่ได้กำหนดตายตัวว่าจะต้องเลือกใช้เทคนิคใดก่อนหรือหลัง ซึ่งถ้าลำดับของการใช้งานเทคนิคมีความแตกต่างกัน ภาพผลลัพธ์ที่ได้ก็จะแตกต่างกันไปด้วย ดังนั้นการจะเลือกใช้เทคนิคใดๆ จึงขึ้นอยู่กับดุลพินิจของผู้ใช้ทั้งสิ้น แน่นอนว่าแต่ละเทคนิคจะมีลักษณะความสามารถในการทำงานที่แตกต่างกัน ผู้ใช้จึงจำเป็นต้องเรียนรู้ถึงหลักการทำงานของแต่ละเทคนิคให้เข้าใจชัดเจนก่อนเลือกใช้งาน เพื่อจะได้เลือกใช้ได้เหมาะสมและบรรลุผลตามเป้าหมายที่ต้องการ

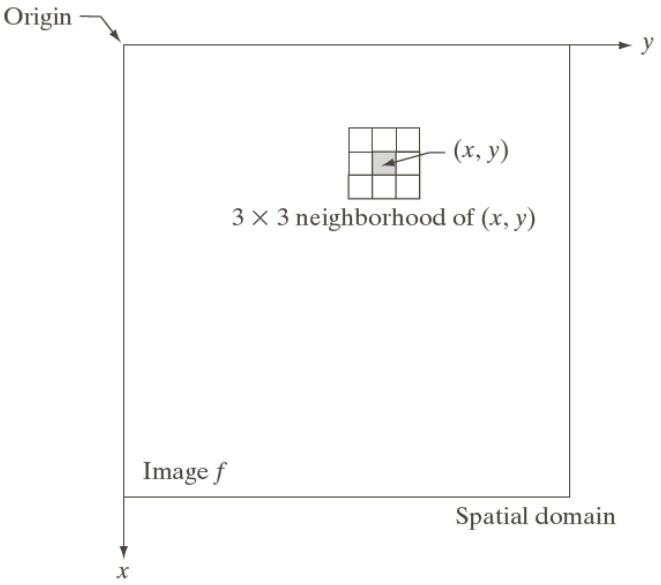
ในการปรับปรุงคุณภาพของภาพสามารถแบ่งออกได้เป็นเทคนิคใน 2 ประเภทใหญ่ๆ ด้วยกันตามโดเมนของข้อมูลภาพที่นำมาใช้งาน นั่นคือ

1. โดเมนระยะทาง (Spatial Domain) เป็นการประมวลผลโดยตรงกับพิกเซล (pixel) ของภาพ (เนื่องจากภาพดิจิตอลเกิดจากการชักตัวอย่าง (Sampling) ในแนวแกน x และ y ดังนั้นการประมวลผลในเชิงระยะทางจึงเกี่ยวข้องกับค่าในแนวแกนแกน x และ y นั่นคือเป็นการประมวลผลที่แต่ละพิกเซลของภาพโดยตรงนั่นเอง)
2. โดเมนความถี่ (Frequency Domain) เป็นการประมวลผลบนค่าองค์ประกอบความถี่ (Frequency Component) ซึ่งเป็นผลลัพธ์ที่ได้จากการแปลงข้อมูลภาพด้วยวิธีการแปลงแบบฟูริเยร์ (Fourier Transformation)

ในบทนี้เราจะกล่าวถึงเทคนิคต่างๆ ในโดเมนระยะทางเท่านั้น ส่วนเทคนิคบนโดเมนความถี่จะกล่าวถึงในบทต่อไป อย่างไรก็ตามผลลัพธ์ที่ได้จากเทคนิคของทั้งสองโดเมนนี้จะคล้ายคลึงกันมากเพราะข้อมูลบนโดเมนของความถี่นั้นก็ได้มาจากการแปลงข้อมูลในโดเมนระยะทางไปเป็นข้อมูลความถี่นั่นเอง

4.1 พื้นฐานการปรับปรุงภาพในโดเมนระยะทาง (Background of Image Enhancement in Spatial Domain)

จากที่เคยกล่าวไปแล้วว่ารูปภาพนั้นนิยามได้ในรูปของฟังก์ชัน $f(x, y)$ ใน 2 มิติเมื่อ x และ y เป็นของค่าระยะทางในแนวแกน x และแกน y และค่า amplitude ของ f ณ แต่ละพิกัด (x, y) เรียกว่า intensity ของภาพ ณ ตำแหน่งนั้นๆ รูปที่ 4.1 แสดงลักษณะการอ้างอิงตำแหน่งของรูปภาพ $f(x, y)$ โดยแสดงถึงลักษณะของ neighborhood (พิกเซลข้างเคียง) ขนาด 3x3 ของพิกเซล ณ ตำแหน่ง (x, y) ใดๆ ของภาพ



รูปที่ 4.1 Neighborhood (พิกเซลข้างเคียง) ขนาด 3x3 ของพิกเซล ณ ตำแหน่ง (x, y) ใดๆ ของภาพดิจิตอล

เทคนิคต่างๆ ในการปรับปรุงคุณภาพของภาพ $f(x, y)$ นั้นแทนด้วยฟังก์ชัน T หรือเรียกอีกอย่างว่า ฟังก์ชันการแปลง (Transformation Function) ที่มีอินพุตของฟังก์ชันคือ $f(x, y)$ โดยฟังก์ชันการแปลงจะกระทำบนกลุ่มพิกเซลข้างเคียง ณ ตำแหน่ง (x, y) และผลลัพธ์ที่ได้จากการแปลงด้วยฟังก์ชัน T ดังกล่าวคือ ภาพเอาต์พุต $g(x,y)$ ซึ่งสามารถเขียนอธิบายเป็นสมการได้ดังสมการที่ 4.1

$$g(x, y) = T[f(x, y)] \qquad \text{สมการ 4.1}$$

ในการดำเนินการด้วยฟังก์ชัน T นั้น จะกำหนดพิกเซลข้างเคียงสำหรับ (x, y) ใดๆ โดยพิจารณาภายในพื้นที่สี่เหลี่ยมจัตุรัสหรือสี่เหลี่ยมผืนผ้าเล็กๆ ที่อยู่รอบๆ (x, y) ดังแสดงในรูปที่ 4.1 ผลลัพธ์ที่ได้จากการดำเนินการด้วยฟังก์ชัน T คือค่าระดับความเทา (intensity) ค่าใหม่ที่จะกำหนดให้กับพิกเซล (x, y) นั่นเอง นั่นคือ ในการปรับปรุงคุณภาพ ณ ตำแหน่งพิกเซล (x,y)ใดๆ จะใช้ค่าระดับความเทาของพิกเซลข้างเคียงของ (x, y) ในการคำนวณ

พื้นที่สี่เหลี่ยมจัตุรัสหรือสี่เหลี่ยมผืนผ้าเล็กๆ ที่กำหนดเป็นพิกเซลข้างเคียงของพิกเซล (x, y) ใดๆ นั้นอาจเรียกว่า มาส์ก (Masks) ตัวกรอง (Filters) วินโดส์ (Windows) แทมเพรต (Tamplates) หรือเคอเนลล์ (Kernels) โดยในแต่ละเทคนิคจะนำมาส์กที่มีค่าสัมประสิทธิ์ที่ต่างกัน (ทำให้ได้ผลลัพธ์ที่ต่างกันในแต่ละเทคนิค) มาคูณกับค่าระดับความเทาของพื้นที่ภาพที่มาส์กนี้ทับอยู่ ค่าระดับความเทาของพิกเซล (x, y) ที่อยู่ ณ ตำแหน่งตรงกลางมาส์กจะถูกกำหนดโดยค่าใหม่ที่ได้จากการดำเนินการด้วยฟังก์ชัน T การคำนวณค่าระดับความเทาใหม่นั้นจะทำกับทุกๆ พิกเซลของภาพซึ่งทำได้โดยการกวาดมาส์กไปทั่วทั้งภาพ โดยเริ่มจากมุมบนซ้ายสุดของภาพ (Origin) แล้วขยับไปทีละพิกเซลไปทางขวามือ (ตามแนวแกน y ดังรูปที่ 4.1) เมื่อหมดหนึ่งแถวก็ขยับลงมาด้านล่างหนึ่งแถว (ตามแนวแกน x ดังรูปที่ 4.1) นั่นคือเป็นการเลื่อนมาส์กไปในทิศทางซ้ายไปขวา บนลงล่าง โดยที่พิกเซลบริเวณขอบของภาพนั้นมักจะกำหนดให้พิกเซลข้างเคียงที่อยู่นอกกรอบของภาพมีค่าระดับเทาเป็นศูนย์ (สีดำ) ทั้งหมด

ขนาดของมาส์กที่มีขนาดเล็กที่สุดคือ 1x1 ซึ่งหมายถึงการใช้ค่าระดับเทาของพิกเซล (x, y) เพียงค่าเดียว (ไม่พิจารณาพิกเซลข้างเคียง) ในการพิจารณาหาค่าระดับความเทาใหม่ให้กับตัวมันเอง เราเรียกฟังก์ชัน T ที่นำมาใช้งานในลักษณะนี้ว่า ฟังก์ชันการแปลงระดับความเทา (Gray-level or Intensity or Mapping Transformation

Function) และเรียกการประมวลผลในลักษณะนี้ว่า การประมวลผลจุด (Point Processing) ซึ่งสามารถเขียนในรูปสมการได้ดังสมการที่ 4.2

$$s = T(r) \qquad \text{สมการ 4.2}$$

เมื่อ r คือ ระดับความเทาตั้งต้นของพิกเซล (x, y) และ s คือ ระดับความเทาใหม่ของพิกเซล (x, y)

สำหรับมาส์กที่มีขนาดใหญ่กว่า 1x1 (มักอยู่ในรูปแบบ 2n+1 x 2n+1) เช่น 3x3, 5x5, … ซึ่งในการประมวลผลที่ใช้มาส์กขนาดใหญ่กว่า 1x1 นั้นเรียกว่าการประมวลผลมาส์กหรือการกรอง (Mask Processing or Filtering)

4.2 ฟังก์ชันพื้นฐานในการแปลงระดับความเทา (Basic Gray Level Transformation Functions)

สำหรับฟังก์ชันพื้นฐานในการแปลงค่าระดับความเทาที่ใช้ในการแปลงค่าระดับความเทาของภาพ เพื่อเป็นการปรับปรุงคุณภาพของภาพให้ดีขึ้นได้นั้น สามารถแบ่งออกได้เป็น

1. ฟังก์ชันการแปลงแบบเชิงเส้น (Linear Transformation Functions)
2. ฟังก์ชันการแปลงแบบลอการึทึม (Log Transformation Functions)
3. ฟังก์ชันการแปลงแบบยกกำลัง (Power-law Transformation Functions)
4. ฟังก์ชันการแปลงแบบเชิงเส้นต่อเนื่อง (Piecewise -linear Transformation Functions)

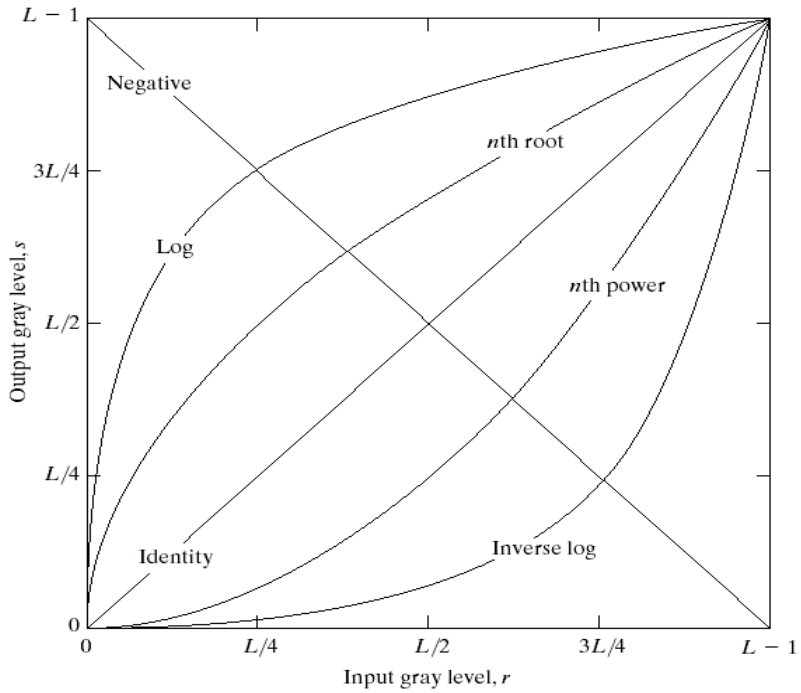
- **ฟังก์ชันการแปลงแบบเชิงเส้น (Linear Transformation Functions)**

  ประกอบด้วย

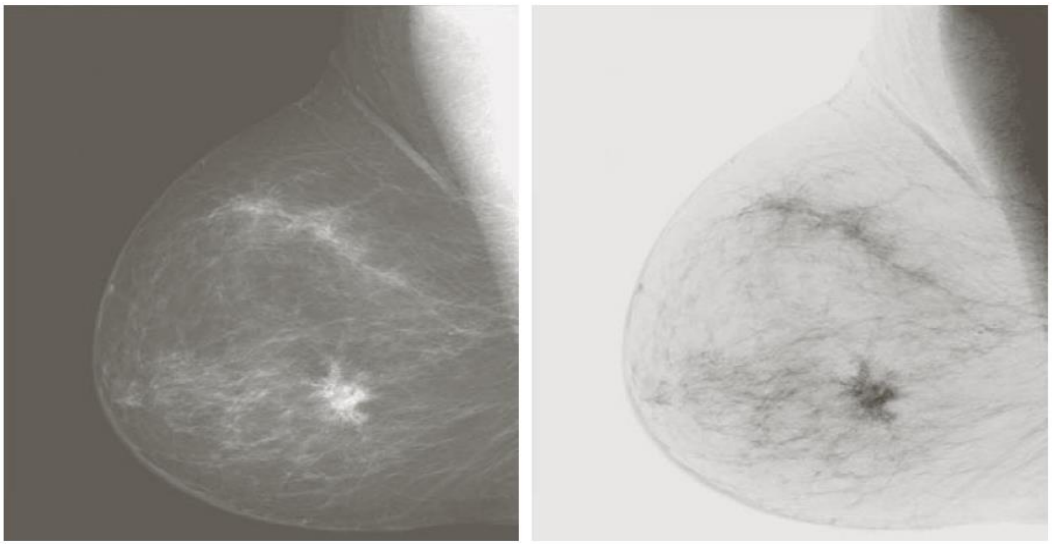
  - ฟังก์ชันการแปลงเอกลักษณ์ (Identity Transformation Function) ผลลัพธ์จากการแปลงที่ได้คือภาพต้นฉบับนั่นเอง
  - ฟังก์ชันการแปลงค่ากลับ (Negative Transformation Function) เป็นการกลับค่าระดับเทา เช่นถ้าช่วงระดับความเทาของภาพคือ [0, L-1] แล้วฟังก์ชันการแปลงค่ากลับสามารถเขียนได้ดังสมการ

    $$s = L - 1 - r$$

  การใช้ฟังก์ชันนี้เหมาะสำหรับการทำให้บริเวณที่มีสีขาวหรือเทาบนพื้นภาพสีดำมีความเด่นชัดมากขึ้นโดยเฉพาะภาพที่องค์ประกอบสีดำมีจำนวนมาก



รูปที่ 4.2 Input และ Output ของค่าระดับเทาจากการใช้ฟังก์ชันการแปลงค่าระดับเทาแบบเชิงเส้น ลอการิทึมและยกกำลัง



รูปที่ 4.3 ภาพแมมโมแกรมต้นฉบับ (ซ้าย) และภาพผลลัพธ์(ขวา) หลังการใช้ฟังก์ชันการแปลงค่ากลับ

- **ฟังก์ชันการแปลงแบบลอการึทึม (Log Transformation Functions)**

  ประกอบด้วย

  - ฟังก์ชันการแปลงแบบลอการึทึม (Log Transformation Functions)

    เป็นการเพิ่มความสว่างให้แก่แต่ละค่าระดับความเทา r ที่มีค่าต่ำๆ (สังเกตได้ว่าความชันของฟังก์ชันช่วง r ต่ำๆ จะมีค่าสูง) ซึ่งจะทำให้ pixel ที่มีระดับความเทาต่ำหรือมืดกลับสว่างมากขึ้น นั่นคือบริเวณค่า r ต่ำๆ จะถูกยืดค่าระดับความเทาออก (spreading) ในขณะที่ pixel ที่เป็นสีเทาสว่าง หรือ

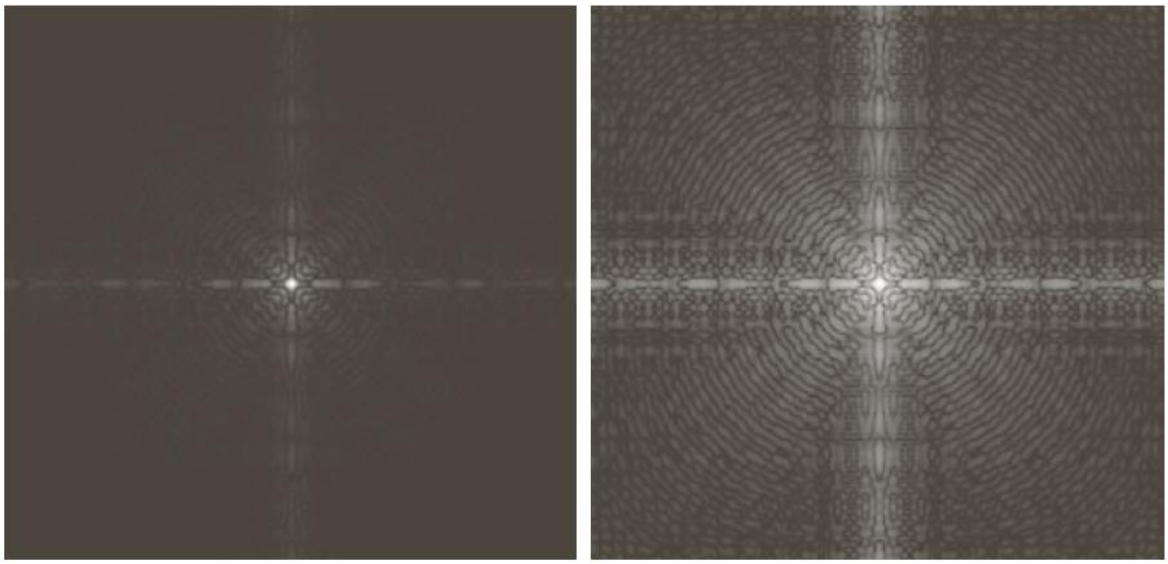
มีค่า r สูงๆ จะถูกบีบให้มีค่าลดลง กล่าวคือ pixel ที่เคยสว่างมากจะถูกทำให้สว่างน้อยลง ทำให้บริเวณที่มีค่า r สูงๆ นั้นถูกบีบอัดค่าระดับเทาลง (Compressing) แต่ระดับความเทาค่าสูงสุด (สีขาว) และต่ำสุด (สีดำ) ยังคงมีค่าเท่าเดิม ไม่เปลี่ยนแปลง โดยที่สมการของฟังก์ชันการแปลงแบบลอการึทึมคือ

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

เมื่อ c คือค่าคงที่และ $r >= 0$

ภาพที่เหมาะจะใช้ฟังก์ชันนี้ในการเพิ่มคุณภาพการมองเห็นได้แก่ ภาพที่มีช่วงระดับความเทาที่กว้างมากๆ (ค่าระดับความสว่างสูงสุดมีค่ามากๆ) เช่น Fourier Spectrum

- ฟังก์ชันการแปลงแบบอินเวอร์สลอการึทึม (Inverse-log Transformation Functions)
  จะทำงานตรงกันข้ามกับการทำงานของฟังก์ชันการแปลงแบบลอการึทึม กล่าวคือฟังก์ชันการแปลงแบบอินเวอร์สลอการึทึมจะทำให้ pixel ที่มีระดับความเทาต่ำมีค่ายิ่งต่ำลงไปอีก ในขณะที่ pixel ที่สว่างถูกทำให้สว่างมากขึ้น

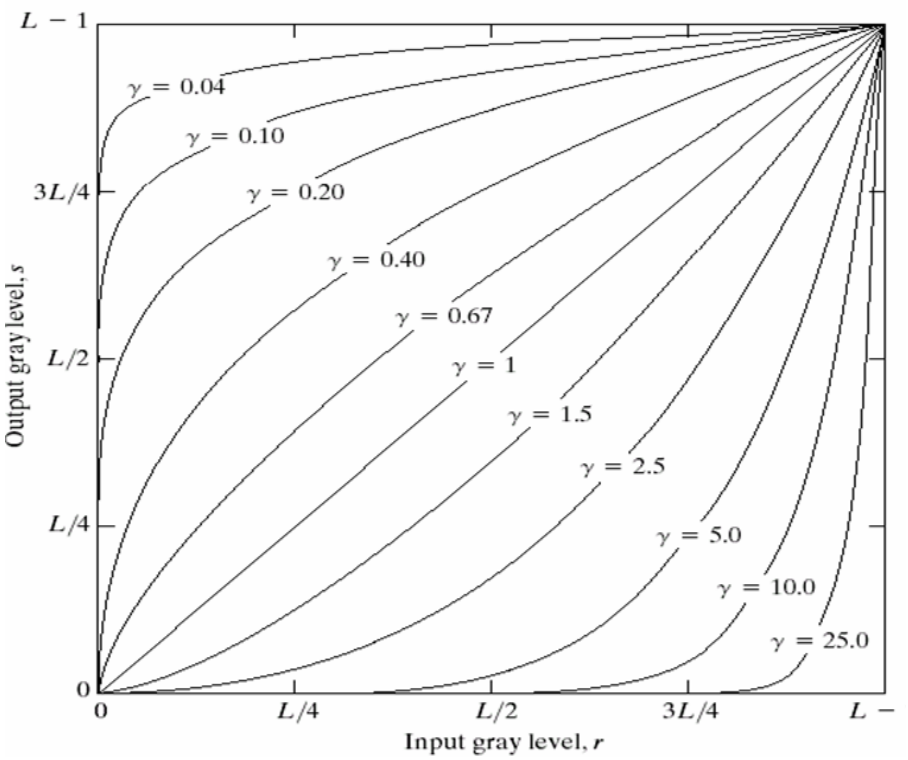


Fourier Spectrum ที่มีช่วงแอมปลิจูดเท่ากับ [0, 1.5 x 10^6] *หลังการใช้ฟังก์ชันการแปลงแบบลอการึทึมที่ c=1*

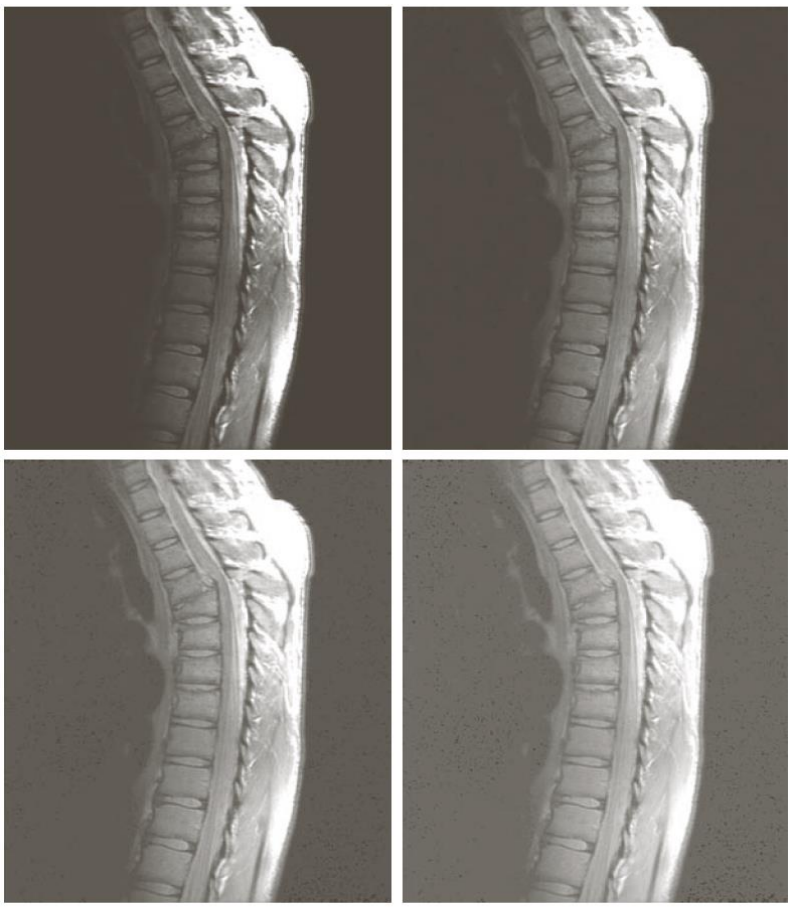
รูปที่ 4.4 ภาพ Fourier Spectrumหลังการใช้ฟังก์ชันการแปลงแบบลอการึทึมที่ c=1

- **ฟังก์ชันการแปลงแบบยกกำลัง (Power-law Transformation Functions)**

  ฟังก์ชันการแปลงแบบยกกำลังนั้นอยู่ในรูปของสมการ $s = cr^{\gamma}$



รูปที่ 4.5 ฟังก์ชันการแปลงแบบยกกำลังที่มีค่า γ ต่างๆ กันและค่า c=1

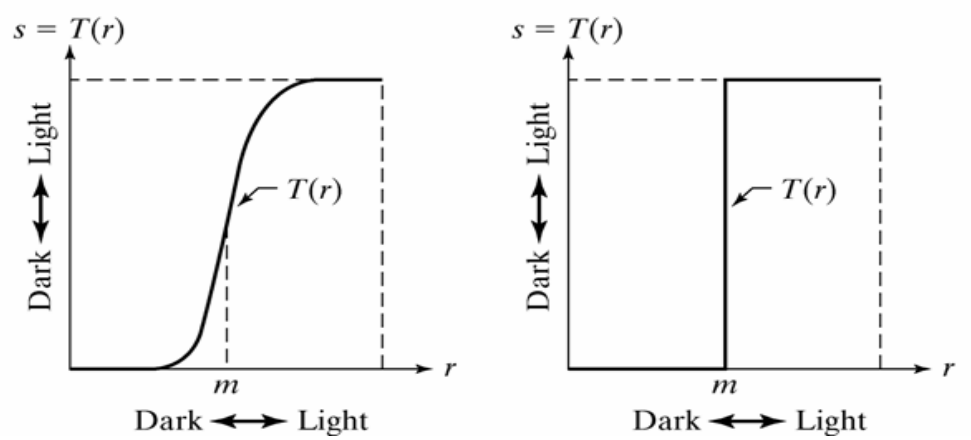


รูปที่ 4.6 ภาพจากการใช้ฟังก์ชันการแปลงแบบยกกำลังที่มีค่า r= 0.6, 0.4 และ 0.3 ตามลำดับ (ภาพบนซ้ายคือภาพต้นฉบับ) เมื่อค่า c=1

รูปที่ 4.7 ภาพจากการใช้ฟังก์ชันการแปลงแบบยกกำลังที่มีค่า r= 3, 4 และ 5 ตามลำดับ (ภาพบนซ้ายคือภาพต้นฉบับ) เมื่อค่า c=1

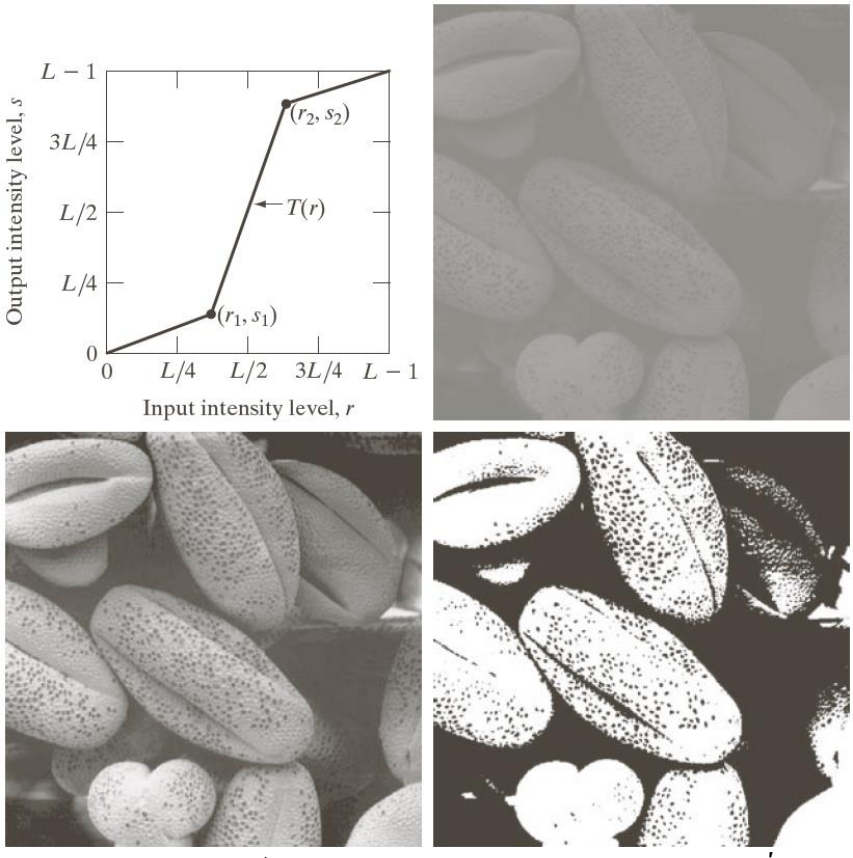
- **ฟังก์ชันการแปลงแบบเชิงเส้นต่อเนื่อง (Piecewise -linear Transformation Functions)**

  - ฟังก์ชันการแปลงแบบยืดความคมชัดเชิงเส้นต่อเนื่อง (Piecewise-Linear Transformation Function) เป็นการแปลงแบบยืดความแตกต่างของระดับความเทาในลักษณะรูปที่ 4.8 ซ้าย ซึ่งค่าระดับเทา r ที่มีค่าต่ำกว่า m จะถูกบีบค่าต่ำลงหรือมืดลง ส่วนค่า r ที่มากกว่า m จะถูกขยายให้มีค่าสูงขึ้นหรือสว่างขึ้น
  - ฟังก์ชันการแปลงแบบดีสครีต (Discrete Transformation Function) เป็นการแปลงแบบยืดระดับเทาที่ผลลัพธ์มีอยู่เพียง 2 ค่าคือ 0 และ 1 จึงทำให้ได้ภาพเป็นภาพขาวดำ จึงเรียกฟังก์ชันนี้ว่า ฟังก์ชันค่าแบ่ง (Thresholding Function)



(ซ้าย) ฟังก์ชันต่อเนื่องการแปลงแบบยืดระดับเทา (ขวา) ฟังก์ชันดีสครีตการแปลงแบบยืดระดับเทา

รูปที่ 4.8 ฟังก์ชันการแปลงแบบเชิงเส้นต่อเนื่อง



รูปที่ 4.9 ภาพจากการใช้ฟังก์ชันการแปลงแบบยืดความคมชัดเชิงเส้นต่อเนื่อง (ล่างซ้าย) และแบบฟังก์ชันค่าแบ่ง(ล่างขวา) เมื่อภาพบนขวาคือภาพต้นฉบับ

4.3 การประมวลผลอีสโตแกรม (Histogram Processing)

- **Histogram**

Histogram หมายถึง กราฟที่แสดงจำนวน pixel ของสี หรือของ intensity ค่าต่างๆ

$$h(r_k) = n_k$$

เมื่อ r_k คือค่าระดับเทาที่ k และ

n_k คือจำนวนพิกเซลในภาพที่มีค่าระดับความเทาเท่ากับ r_k

- Function imhist();

Syntax:
h = imhist(f);
imhist(f)

h= imhist(f, b);
imhist(f, b)

Example 1: Simple plot image histogram

```
>> I = imread('breast.tif');
>> imshow(I)
>> imhist(I)

>> figure, imhist(I, 4)
>> J = imhist(I, 4)

>> figure, imhist(I, 10)
>> K = imhist(I, 10)
```

Example 2: Various ways (imhist, bar, stem, and plot) to plot an image histogram

```
>> I = imread('breast.tif');
>> imshow(I)
>> imhist(I)    % Plot by the default imhist function

>> h = imhist(I);
>> h1 = h(1:10:256);
>> horz = 1:10:256;

>> figure, bar(horz, h1)          % Plot by function bar
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)            % gca: graphics current axis
>> set(gca, 'ytick', 0:2500:5000)

>> figure, stem(horz, h1, 'fill')          % Plot by function stem
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)
```

```
>> figure, stem(horz, h1, 'r--s', 'fill')  % Another plot by function stem
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)

>> h = imhist(I);
>> figure, plot(h)       % use the default value of plot function
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)

>> figure, plot(horz, h1, 'color', 'm', 'linestyle', 'none', 'marker', 'd')  %Another plot function
>> axis([0 255 0 5000])
>> set(gca, 'xtick', 0:50:255)
>> set(gca, 'ytick', 0:2500:5000)
```

Symbol	Color	Symbol	Line Style	Symbol	Marker
k	Black	–	Solid	+	Plus sign
w	White	– –	Dashed	o	Circle
r	Red	:	Dotted	*	Asterisk
g	Green	–.	Dash-dot	.	Point
b	Blue	none	No line	x	Cross
c	Cyan			s	Square
y	Yellow			d	Diamond
m	Magenta			none	No marker

Attributes for function stem and plot.

- **ความคมชัด (Contrast)**

Contrast หมายถึงความคมชัดของภาพ เป็นความแตกต่างระหว่าง สีที่มืดที่สุดในภาพกับสีที่สว่างที่สุดในภาพ ซึ่งรูปแบบที่แตกต่างกันของ contrast (Different types of contrast) เช่น

- ภาพที่มืด (dark image) จะมี histogram กองอยู่ไปทางซ้าย
- ภาพที่สว่าง (bright image) จะมี histogram กองอยู่ไปทางขวา
- ภาพที่มี contrast ต่ำ (low contrast) จะมี histogram กระจุกกันอยู่ในช่วงแคบๆ
- ภาพที่มี contrast สูง (high contrast) จะมี histogram กระจายกันอยู่ในช่วงกว้างๆ

Example: Image with different types of contrast

```
>> I = imread('pollen1.jpg');
>> imhist(I)
>> J = imread('pollen2.jpg');
>> figure, imhist(J)
>> K = imread('pollen3.jpg');
>> figure, imhist(K)
>> L = imread('pollen4.jpg');
>> figure, imhist(L)

>> figure
>> subplot(4,2,1); imshow(I,[]), title('pollen1');
>> subplot(4,2,2); imhist(I), text(125, 4000, 'dark image');
```

```
>> subplot(4,2,3); imshow(J,[]), title('pollen2');
>> subplot(4,2,4); imhist(J),text(40, 4000, 'bright image');
>> subplot(4,2,5); imshow(K,[]), title('pollen3');
>> subplot(4,2,6); imhist(K), text(150, 4000, 'low contrast');
>> subplot(4,2,7); imshow(L,[]), title('pollen4');
>> subplot(4,2,8); imhist(L), text(100, 3000, 'high contrast');
```

- **Histogram Processing**

หมายถึงกระบวนการปรับปรุง intensity ของรูปภาพเพื่อให้ได้ histogram ที่มีลักษณะตามต้องการ เช่น

- Histogram equalization เป็นการทำให้ histogram กระจายกันอย่างสม่ำเสมอตลอด
- Histogram matching เป็นการทำให้ histogram มีลักษณะเหมือนกราฟที่กำหนดไว้

- **Histogram Equalization**

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{N}$$

n_j = the number of pixels with intensity = j
N = the number of total pixels

ตัวอย่างการคำนวณ Histogram Equalization

Intensity	# pixels
0	20
1	5
2	25
3	10
4	15
5	5
6	10
7	10
Total	100

Accumulative Sum of P_r
20/100 = 0.2
(20+5)/100 = 0.25
(20+5+25)/100 = 0.5
(20+5+25+10)/100 = 0.6
(20+5+25+10+15)/100 = 0.75
(20+5+25+10+15+5)/100 = 0.8
(20+5+25+10+15+5+10)/100 = 0.9
(20+5+25+10+15+5+10+10)/100 = 1.0
1.0

Intensity (r)	No. of Pixels (n_j)	Acc Sum of P_r	Output value	Quantized Output (s)
0	20	0.2	0.2x7 = 1.4	1
1	5	0.25	0.25*7 = 1.75	2
2	25	0.5	0.5*7 = 3.5	3
3	10	0.6	0.6*7 = 4.2	4
4	15	0.75	0.75*7 = 5.25	5
5	5	0.8	0.8*7 = 5.6	6
6	10	0.9	0.9*7 = 6.3	6
7	10	1.0	1.0x7 = 7	7
Total	100			

Histogram Equalization ใน MATLAB

- Function histeq();

Syntax:

```
g = histeq(f, nlev);
```

f is the input image

nlev is the number of intensity levels specified for the output image (default value is 64 but should use 256).

Note: การใช้ ฟังก์ชัน histeq ของ Matlab นั้นถ้าไม่ใส่ parameter ที่กำหนด level จะ default ค่าให้เท่ากับ 64 ดังตัวอย่าง

```
>> histeq(I);
```

แต่เราควรใส่ parameter ที่กำหนด level เองให้เป็น 256 ซึ่งจะเป็นไปตามทฤษฎีของการทำ histogram equalization ดังตัวอย่าง

```
>> histeq(I, 256);
```

อย่างไรก็ตามผลลัพธ์ที่ได้จากการกำหนด level ทั้งสองแบบก็ไม่ได้แตกต่างกันมากนัก

Example1:

```
>> I = imread('tire.tif');
>> J = histeq(I); %default to 64 level
>> imshow(I), figure, imshow(J)
>> figure, imhist(J), ylim('auto')

>> K = histeq(I,256); %default to 64 level but should use 256
>> figure, imshow(K)
>> figure, imhist(K), ylim('auto')
```

Example2:

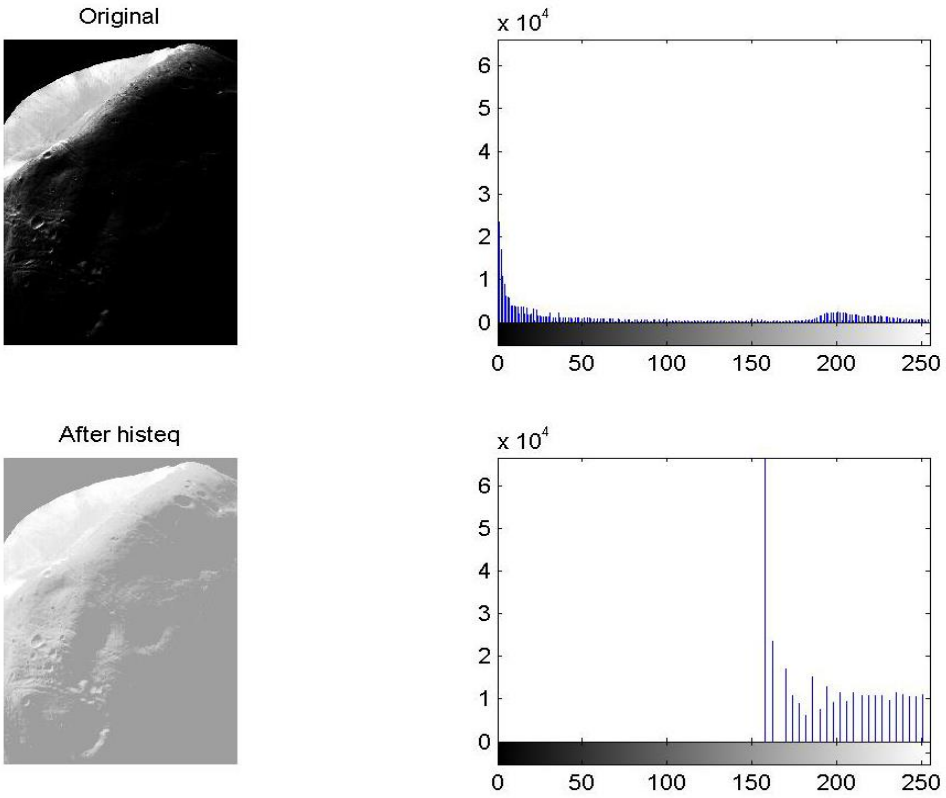
```
>> f = imread('pollen1.jpg');
>> imshow(f)
>> figure, imhist(f), ylim('auto')
>> g = histeq(f, 256);
>> figure, imshow(g);
>> figure, imhist(g), ylim('auto');
```

Example3:

```
>> I = imread('pollen1.jpg');
>> J = imread('pollen2.jpg');
>> K = imread('pollen3.jpg');
>> L = imread('pollen4.jpg');

>> figure
>> subplot(4,2,1); imshow(I,[]), title('pollen1');
>> subplot(4,2,2); imhist(I), text(125, 4000, 'dark image');
>> subplot(4,2,3); imshow(J,[]), title('pollen2');
>> subplot(4,2,4); imhist(J),text(40, 4000, 'bright image');
>> subplot(4,2,5); imshow(K,[]), title('pollen3');
>> subplot(4,2,6); imhist(K), text(150, 4000, 'low contrast');
>> subplot(4,2,7); imshow(L,[]), title('pollen4');
>> subplot(4,2,8); imhist(L), text(100, 3000, 'high contrast');

>> figure
>> subplot(4,2,1); histeq(I,256); II=histeq(I); title('histeq pollen1');
>> subplot(4,2,2); imhist(II);
>> subplot(4,2,3); histeq(J,256); JJ=histeq(J); title('histeq pollen2');
>> subplot(4,2,4); imhist(JJ);
>> subplot(4,2,5); histeq(K,256); KK=histeq(K); title('histeq pollen3');
>> subplot(4,2,6); imhist(KK);
>> subplot(4,2,7); histeq(L,256); LL=histeq(L); title('histeq pollen4');
>> subplot(4,2,8); imhist(LL);
```

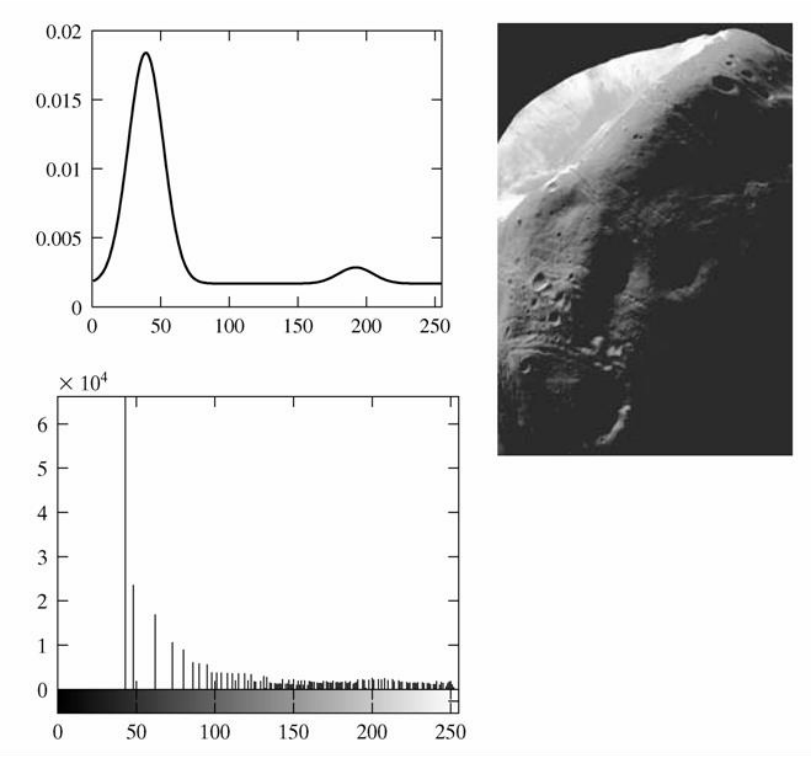
- **Histogram Matching**

ข้อจำกัดของ histogram equalization คือ เป็นการกำหนดฟังก์ชันความหนาแน่นความน่าจะเป็นของระดับความเทาของภาพผลลัพธ์เป็นลักษณะรูปแบบที่ตายตัว กล่าวคือ เป็นฟังก์ชันความหนาแน่นความน่าจะเป็นของระดับความเท่าแบบสม่ำเสมอเท่านั้น หมายความว่า ถึงแม้ค่า histogram ของภาพ input มีการเปลี่ยนแปลงไป ฟังก์ชันที่ใช้ในการแปลงค่าก็ไม่ได้มีการเปลี่ยนแปลงตามไปด้วย ทำให้ในบางครั้ง histogram equalization ก็ได้ผลลัพธ์ที่ไม่ดีในการปรับปรุงคุณภาพกับภาพบางแบบ ดังตัวอย่าง ภาพ Moon Phobos



การใช้ histogram equalization ที่ทำให้ภาพผลลัพธ์ที่ได้เป็นภาพแบบ low contrast ไป ซึ่งไม่ได้ทำให้การปรับปรุงคุณภาพของภาพดีขึ้น

ดังนั้นสำหรับภาพบางภาพหรือใน Applications บางชนิด เราอาจต้องการให้ภาพผลลัพธ์มีลักษณะเฉพาะของฟังก์ชันความหนาแน่นความน่าจะเป็นของระดับความเทาเป็นไปตามที่เราต้องการ ซึ่ง histogram equalization ไม่สามารถทำได้ แต่สามารถทำได้ด้วยวิธีที่เรียกว่า histogram matching หรืออาจเรียกว่าวิธีการกำหนดลักษณะเฉพาะของ histogram (histogram specification) ดังตัวอย่าง เมื่อภาพบนซ้ายคือ model ของลักษณะ histogram ของภาพผลลัพธ์ที่ผู้ใช้ต้องการ และถาพล่างซ้ายคือ histogram ที่ได้จากการทำ histogram matching ซึ่งได้ลักษณะของ histogram ที่ใกล้เคียงกับ model ที่ต้องการ ส่วนภาพทางขวาเป็นภาพผลลัพธ์ที่ได้จากการทำ histogram matching ซึ่งคุณภาพของภาพที่ได้ดีขึ้นมาก



Histogram Matching of Moon Phobos Image

Note: ใน Matlab นั้นก็สามารถใช้ฟังก์ชัน histeq ในการทำ histogram matching ได้แต่ต้องมี input ที่เป็น vector ของ model ของ histogram ที่ต้องการมาก่อน ดังตัวอย่าง

```
>> histeq(I, P);
```

เมื่อ P เป็น vector (row or column matrix) ซึ่งเป็น model ของ histogram ที่ต้องการ

4.4 Spatial Filtering

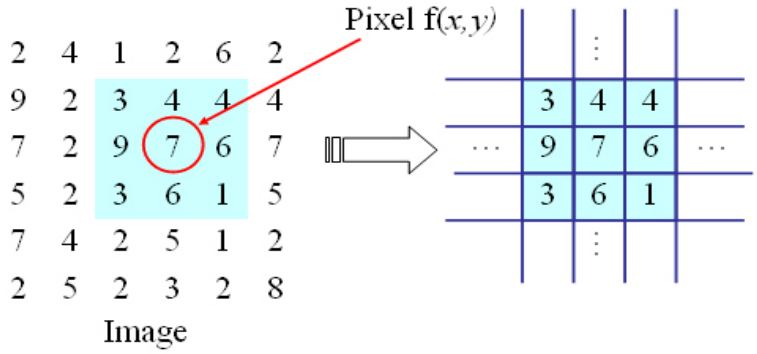
- **Fundamental of Spatial Filter**
  - **Basic Mask Processing**

Sometime we need to manipulate values obtained from neighboring pixels

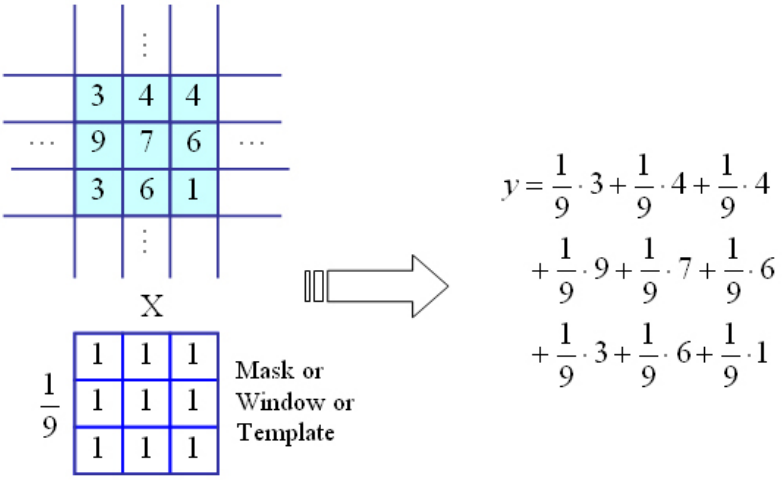
Example: How can we compute an average value of pixels in a 3x3 region center at a pixel *f*(x,y)?

Image

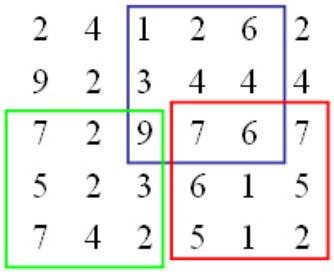
Step 1. Selected only needed pixels



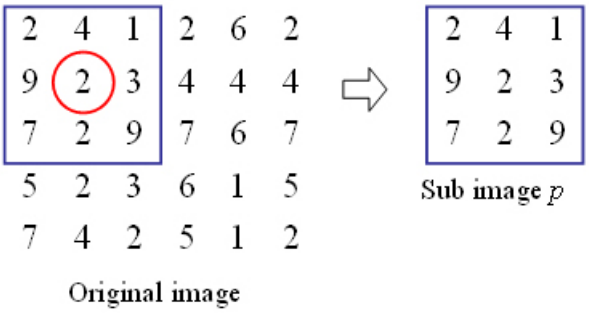
Step 2. Multiply every pixel by 1/9 and then sum up the values



- **Mask processing to every pixels**



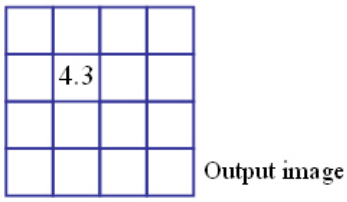
Step 1: Move the window to the first location where we want to compute the average value and then select only pixels inside the window.



Step 2: Compute the average value

$$y = \sum_{i=1}^{3}\sum_{j=1}^{3}\frac{1}{9}\cdot p(i,j)$$

Step 3: Place the result at the pixel in the output image



Step 4: Move the window to the next location and go to Step 2

Note: Wording *mask* was named by different words but the same meaning such as: *window, template, kernel, filter, filter mask or convolution filter.*

Mask can be any size of *m* x *n* that *m = 2a+1* and *n = 2b+1* where *a* and *b* are nonnegative integers

- **Spatial Filter in Matlab**

Example 1: Simple computes mean value directly from the generated data

```
>> x = uint8(10*magic(5))

x =

  170   240    10    80   150
  230    50    70   140   160
   40    60   130   200   220
  100   120   190   210    30
  110   180   250    20    90

>> mean2(x(1:3,1:3))

ans =

  111.1111

>> mean2(x(1:3,2:4))

ans =

  108.8889
```

Example 2: Use *imfilter* function of IPT (Image Processing Toolbox)
Syntax:
g = imfilter(f, w, filtering_mode, boundary_options, size_options)

Note: See the detail syntax of imfilter function by using help (>> help imfilter)

```
>> f = uint8(10*magic(5))

f =

  170   240    10    80   150
  230    50    70   140   160
   40    60   130   200   220
  100   120   190   210    30
  110   180   250    20    90

>> w = ones(3,3)/9

w =

    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111
    0.1111    0.1111    0.1111


>> imfilter(f, w)

ans =

    77    86    66    68    59
    88   111   109   129   106
    67   110   130   150   107
    68   131   151   149    86
    57   106   108    88    39

>> f = uint8(zeros(5))

f =

     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0
     0     0     0     0     0

>> f(3,3) = 1

f =

     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
```

```
>> w = [1 2 3;4 5 6; 7 8 9]

w =

    1    2    3
    4    5    6
    7    8    9

>> imfilter(f,w)

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0


>> imfilter(f,w,'corr')
ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0

>> imfilter(f,w,'conv')

ans =

    0    0    0    0    0
    0    1    2    3    0
    0    4    5    6    0
    0    7    8    9    0
    0    0    0    0    0

>> imfilter(f,w,'corr', 'same')

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0

>> imfilter(f,w,'corr', 'full')

ans =

    0    0    0    0    0    0    0
    0    0    0    0    0    0    0
    0    0    9    8    7    0    0
    0    0    6    5    4    0    0
    0    0    3    2    1    0    0
```

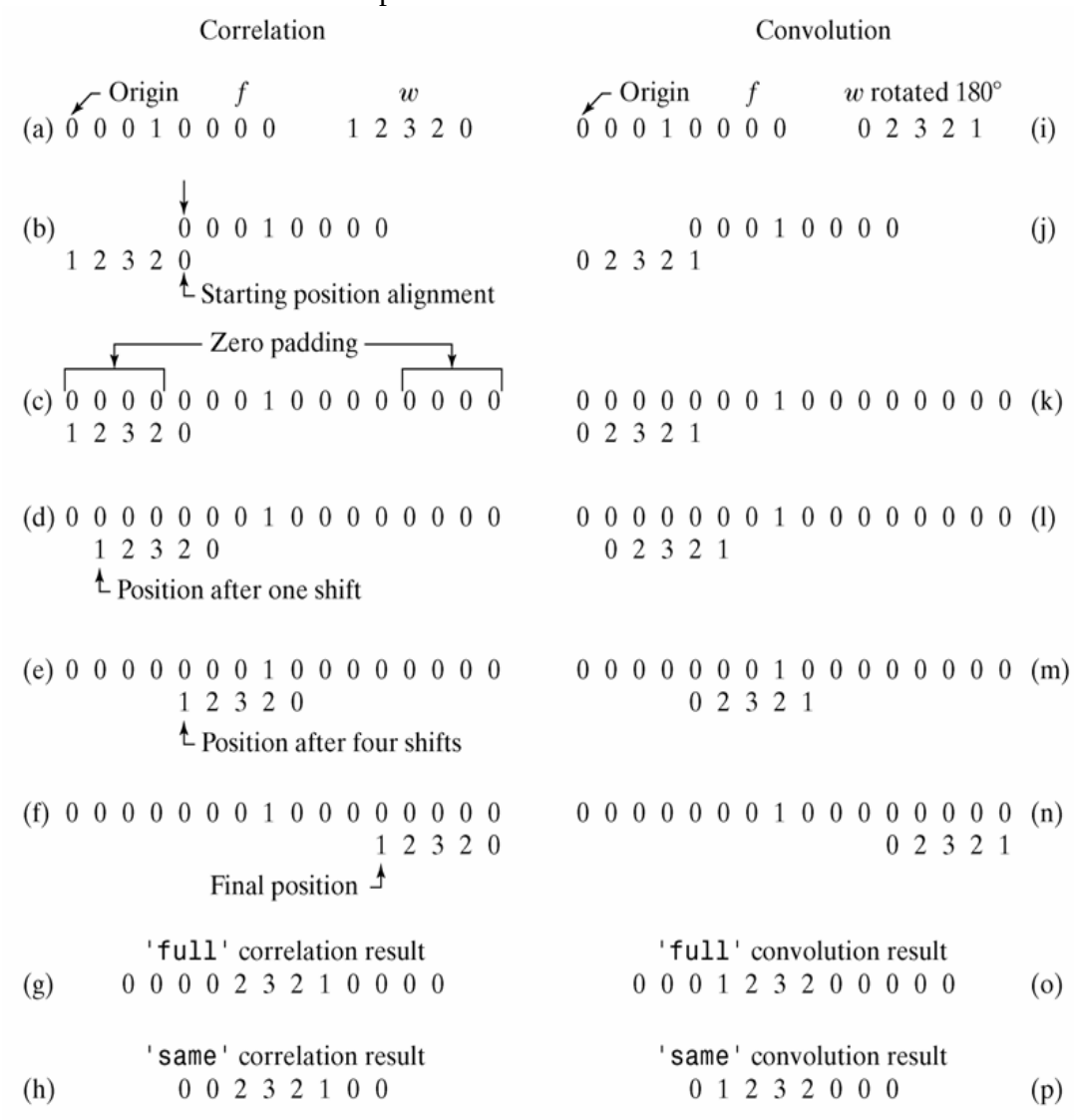
```
    0    0    0    0    0    0    0
    0    0    0    0    0    0    0
```

Example: Test for boundary options

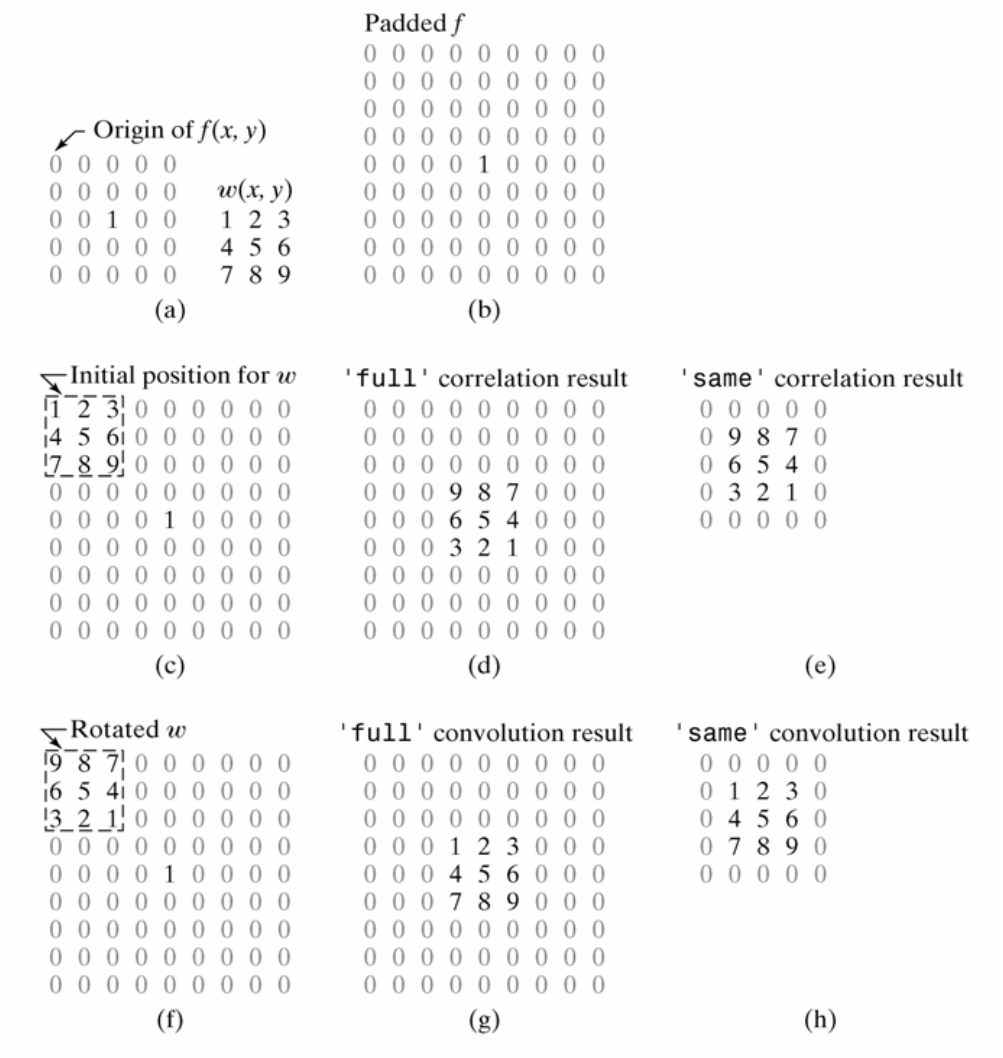
```
>> imfilter(f,w)

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0

>> imfilter(f,w,0)

ans =

    0    0    0    0    0
    0    9    8    7    0
    0    6    5    4    0
    0    3    2    1    0
    0    0    0    0    0


>> imfilter(f,w,1)

ans =

   17    6    6    6   21
   12    9    8    7   18
   12    6    5    4   18
   12    3    2    1   18
   29   24   24   24   33

>> imfilter(f,w,255)

ans =

  255  255  255  255  255
  255    9    8    7  255
  255    6    5    4  255
  255    3    2    1  255
  255  255  255  255  255
```

Options	Description
Filtering Mode	
`'corr'`	Filtering is done using correlation (see Figs. 3.13 and 3.14). This is the default.
`'conv'`	Filtering is done using convolution (see Figs. 3.13 and 3.14).
Boundary Options	
`P`	The boundaries of the input image are extended by padding with a value, `P` (written without quotes). This is the default, with value 0.
`'replicate'`	The size of the image is extended by replicating the values in its outer border.
`'symmetric'`	The size of the image is extended by mirror-reflecting it across its border.
`'circular'`	The size of the image is extended by treating the image as one period a 2-D periodic function.
Size Options	
`'full'`	The output is of the same size as the extended (padded) image (see Figs. 3.13 and 3.14).
`'same'`	The output is of the same size as the input. This is achieved by limiting the excursions of the center of the filter mask to points contained in the original image (see Figs. 3.13 and 3.14). This is the default.

Options for function imfilter



One-dimension correlation and convolution

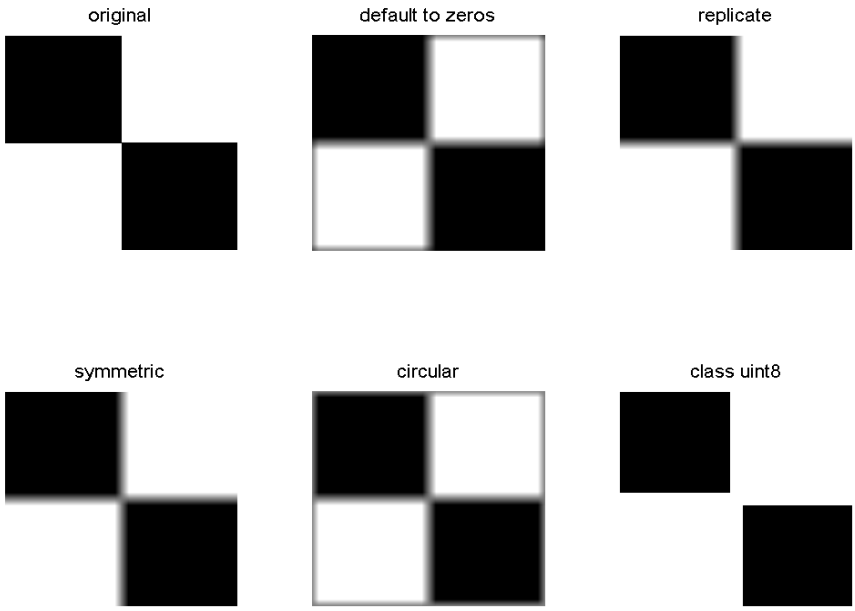


Two-dimension correlation and convolution

Example 3: Test boundary options of function *imfilter*

```
>> f = imread('block.tif');  %image of 'block.tif' can load from moodle
>> f = im2double(f);
>> subplot(2,3,1); imshow(f,[]), title('original');

>> w = ones(31);

>> gd = imfilter(f, w);
>> subplot(2,3,2); imshow(gd,[]), title('default to zeros');

>> gr = imfilter(f, w, 'replicate');
>> subplot(2,3,3); imshow(gr,[]), title('replicate');

>> gs = imfilter(f, w, 'symmetric');
>> subplot(2,3,4); imshow(gs,[]), title('symmetric');

>> gc = imfilter(f, w, 'circular');
>> subplot(2,3,5); imshow(gc,[]), title('circular');

>> f8 = im2uint8(f);
```

```
>> g8r = imfilter(f8, w, 'replicate');
>> subplot(2,3,6); imshow(g8r,[]), title('class uint8');
```



Result of Example 3

- **Linear Spatial Filter**
  - **Function *fspecial()*** ใช้ในการ generate filter mask ที่เป็นแบบ linear filter

    Syntax:

    w = fspecial('type', parameters)

    ซึ่ง type และ parameters สามารถกำหนดรูปแบบต่างๆ ได้จาก ตารางด้านล่าง โดยที่ในที่นี่จะกล่าวถึงเฉพาะ mask แบบต่อไปนี้คือ

    1. average ใช้ในการทำให้ภาพเลือนขึ้น (blurring image)
    2. gaussian ใช้ในการทำให้ภาพเลือนขึ้น (blurring image)
    3. laplacian ใช้ในการทำให้ภาพคมขึ้น (sharpening image)
    4. log ใช้ในการทำให้ภาพคมขึ้น (sharpening image)

Type	Syntax and Parameters
`'average'`	`fspecial('average', [r c])`. A rectangular averaging filter of size `r` × `c`. The default is 3 × 3. A single number instead of `[r c]` specifies a square filter.
`'disk'`	`fspecial('disk', r)`. A circular averaging filter (within a square of size `2r + 1`) with radius `r`. The default radius is 5.
`'gaussian'`	`fspecial('gaussian', [r c], sig)`. A Gaussian lowpass filter of size `r` × `c` and standard deviation `sig` (positive). The defaults are 3 × 3 and 0.5. A single number instead of `[r c]` specifies a square filter.
`'laplacian'`	`fspecial('laplacian', alpha)`. A 3 × 3 Laplacian filter whose shape is specified by `alpha`, a number in the range [0, 1]. The default value for `alpha` is 0.5.
`'log'`	`fspecial('log', [r c], sig)`. Laplacian of a Gaussian (LoG) filter of size `r` × `c` and standard deviation `sig` (positive). The defaults are 5 × 5 and 0.5. A single number instead of `[r c]` specifies a square filter.
`'motion'`	`fspecial('motion', len, theta)`. Outputs a filter that, when convolved with an image, approximates linear motion (of a camera with respect to the image) of `len` pixels. The direction of motion is `theta`, measured in degrees, counterclockwise from the horizontal. The defaults are 9 and 0, which represents a motion of 9 pixels in the horizontal direction.
`'prewitt'`	`fspecial('prewitt')`. Outputs a 3 × 3 Prewitt mask, `wv`, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: `wh = wv'`.
`'sobel'`	`fspecial('sobel')`. Outputs a 3 × 3 Sobel mask, `sv`, that approximates a vertical gradient. A mask for the horizontal gradient is obtained by transposing the result: `sh = sv'`.
`'unsharp'`	`fspecial('unsharp', alpha)`. Outputs a 3 × 3 unsharp filter. Parameter `alpha` controls the shape; it must be greater than 0 and less than or equal to 1.0; the default is 0.2.

Spatial filters (Mask types) supported by function *fspecial*

Example: Generate average masks

```
>> w = fspecial('average', [3 3])

w =

   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111

>> w = fspecial('average', 3)

w =

   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111
   0.1111   0.1111   0.1111

>> w = fspecial('average', [7 5])

w =

   0.0286   0.0286   0.0286   0.0286   0.0286
```

```
    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286
    0.0286    0.0286    0.0286    0.0286    0.0286
```

Example: Generate gaussian masks

```
>> w = fspecial('gaussian', [5 5], 1)

w =

    0.0030    0.0133    0.0219    0.0133    0.0030
    0.0133    0.0596    0.0983    0.0596    0.0133
    0.0219    0.0983    0.1621    0.0983    0.0219
    0.0133    0.0596    0.0983    0.0596    0.0133
    0.0030    0.0133    0.0219    0.0133    0.0030

>> w = fspecial('gaussian', [5 5], 2)

w =

    0.0232    0.0338    0.0383    0.0338    0.0232
    0.0338    0.0492    0.0558    0.0492    0.0338
    0.0383    0.0558    0.0632    0.0558    0.0383
    0.0338    0.0492    0.0558    0.0492    0.0338
    0.0232    0.0338    0.0383    0.0338    0.0232
```

```
>> w = fspecial('gaussian', [7 7], 3)

w =

    0.0113    0.0149    0.0176    0.0186    0.0176    0.0149    0.0113
    0.0149    0.0197    0.0233    0.0246    0.0233    0.0197    0.0149
    0.0176    0.0233    0.0275    0.0290    0.0275    0.0233    0.0176
    0.0186    0.0246    0.0290    0.0307    0.0290    0.0246    0.0186
    0.0176    0.0233    0.0275    0.0290    0.0275    0.0233    0.0176
    0.0149    0.0197    0.0233    0.0246    0.0233    0.0197    0.0149
    0.0113    0.0149    0.0176    0.0186    0.0176    0.0149    0.0113
```

Example: Plot the distribution of gaussian masks by surface plot

```
>> w1 = fspecial('gaussian',[50 50], 3);
>> surf(1:50, 1:50, w1)

>> w2 = fspecial('gaussian',[50 50], 9);
>> figure, surf(1:50, 1:50, w2)
```

Example: Generate laplacian masks

```
>> w = fspecial('laplacian')

w =

    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667

>> w = fspecial('laplacian',0.2)

w =

    0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667

>> w = fspecial('laplacian',1)

w =

    0.5000         0    0.5000
         0   -2.0000         0
    0.5000         0    0.5000
```

Example: Generate laplacian of Gaussian (loG) masks

```
>> w = fspecial('log')

w =

    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448

>> w = fspecial('log',5,0.5)

w =

    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448

>> w = fspecial('log',3,1)

w =

    0.1004   -0.0234    0.1004
   -0.0234   -0.3079   -0.0234
    0.1004   -0.0234    0.1004
```

- **Nonlinear Spatial Filter**

มาส์กแบบไม่เชิงเส้นนั้นเกิดจากการใช้ค่าทางสถิติ (statistic) เช่น ค่ากลาง ค่ามากสุด และค่าต่ำสุดของระดับความเทาของ pixel ที่อยู่ภายในมาส์กมากำหนดเป็นค่าระดับความเทาใหม่ของ pixel ในภาพ ณ ตำแหน่งที่อยู่ตรงกลางมาส์ก ในที่นี้จะกล่าวถึงเฉพาะมาส์กแบบค่ากลาง (median filter) ซึ่งใช้ในการทำให้ภาพเลือนขึ้น (blurring image) เช่นเดียวกันกับมาส์กแบบเชิงเส้นบางชนิด แต่ภาพที่ได้จากการใช้มาส์กแบบค่ากลางจะเลือนน้อยกว่าในขณะที่สามารถกำจัดสัญญาณรบกวนในภาพได้ผลดีมากกว่า โดยเฉพาะสัญญาณรบกวนชนิดเกลือและพริกไทย (salt and pepper noise)

- **Function *medfilt2()***

  Syntax:

  g = medfilt2(f, [m n], padopt)

4.5 การทำให้ภาพเลือน (Blurring)

การทำให้ภาพเลือนหรือการเบลอภาพนั้น เรียกอีกอย่างว่าการเกลี่ยภาพ (Smoothing) ซึ่งนอกจากจะมีผลทำให้ความคมชัดบริเวณขอบภาพน้อยลงแล้วยังทำให้สัญญาณรบกวน (noise) ภายในภาพลดลงตามไปด้วย โดยทั่วไปแล้วการเบลอภาพนี้จะใช้กำจัดวัตถุขนาดเล็กๆ ในภาพก่อนที่จะทำการแยกส่วนวัตถุขนาดใหญ่ที่ต้องการ (Object Extraction) นอกจากนี้ยังใช้ในการเชื่อมต่อช่องว่างระหว่าง 2 บริเวณที่อยู่ติดกันเข้าด้วยกัน ซึ่ง 2 บริเวณนั้นอาจเป็นอาจเป็นบริเวณของวัตถุอันเดียวกันแต่เกิดความผิดพลาดบางประการที่ทำให้บริเวณทั้งสองขาดออกจากกัน เป็นต้น

หลักการทั่วไปของการทำภาพเบลอนี้ คือ การคำนวณหาค่าระดับความเทาค่าใหม่ให้ตำแหน่ง f(x,y) โดยคำนวณจากค่าเฉลี่ยของระดับความเทาของพิกเซลข้างเคียงภายในมาส์ก การใช้ค่าเฉลี่ยนั้นจะมีผลมากบริเวณที่เป็นเส้นขอบของวัตถุ ตัวอย่างเช่น ถ้าวัตถุมีสีขาวส่วนพื้นหลังภาพคือสีดำหรือเทาแล้ว ระดับความเทาของบริเวณเส้นขอบภาพจะถูกลดค่าลงทำให้มีสีขาวหม่น (เนื่องด้วยค่าเฉลี่ยของบริเวณภายในมาส์กจะมีค่าต่ำกว่าค่าระดับความเทาเดิมของพิกเซลบนเส้นขอบ) ทำให้ความแตกต่างของระดับความเทา ณ บริเวณขอบวัตถุลดลงไป

บริเวณขอบภาพและส่วนที่เป็นสัญญาณรบกวนนั้นเป็นบริเวณที่มีการเปลี่ยนแปลงค่าระดับความเทาสูง ถ้าเราพิจารณาถึงองค์ประกอบความถี่ในบริเวณนี้ก็ย่อมต้องมีค่าสูง การเบลอภาพจึงเป็นการทำให้ค่าองค์ประกอบความถี่สูงของภาพตั้งต้นหายไป ดังนั้น การเบลอภาพจึงเทียบเท่ากับการกรองเอาความถี่สูงออกไป เหลือไว้แต่ส่วนที่มีองค์ประกอบความถี่ต่ำ หรือเรียกได้ว่าเป็นตัวกรองแบบผ่านต่ำ (Low pass Filter)

ตัวอย่างการใช้งาน average mask แสดงดังรูปใน Example 4 และ Example 5 ด้านล่าง ใน Example 4 ภาพซ้ายบนเป็นภาพตั้งต้นขนาด 500 x 500 พิกเซล ที่มีองค์ประกอบสี่เหลี่ยมจัตุรัสที่อยู่ด้านบน ขนาด 3 x 3 , 5 x 5 , 9 x 9 , 15 x 15 , 25 x 25 , 35 x 35 , 45 x 45 , และ 55 x 55 พิกเซลตามลำดับ ระยะห่างระหว่างแต่ละสี่เหลี่ยมเท่ากับ 25 พิกเซล ตัวอักษร a ที่อยู่ด้านล่างมีขนาด 10 , 12 , 14 , 16 , 18 , 20 , 22 และ 24 จุด ตามลำดับ ตัวอักษร a ตัวใหญ่ด้านบนสุดของภาพมีขนาด 60 จุด บริเวณตรงกลางภาพประกอบด้วยกลุ่มเส้นตรงขนาด 5 x 100 พิกเซลเรียงขนานกัน โดยมีระยะห่างระหว่างกันเท่ากับ 20 พิกเซล กลุ่มภาพวงกลมที่อยู่เหนือกลุ่มภาพเส้นตรงมีขนาดเส้นผ่าศูนย์กลางเท่ากับ 25 พิกเซลและมีระดับความเทาพิ่มทีละ

20% ภาพตั้งต้นนี้มีพื้นภาพเป็นสีเทาและมีสี่เหลี่ยมผืนผ้าขนาด 50 x 120 พิกเซลที่ประกอบด้วยสัญญาณรบกวนภายในอยู่ด้านซ้ายมือของภาพ จากการเบลอภาพตั้งต้นนี้ด้วยมาส์กขนาด 3 x 3 , 5 x 5 , 9 x 9 , 15 x 15 และ 35 x 35 ผลลัพธ์ที่ได้คือ ภาพที่เหลือในรูป เรียงจากบนลงล่างซ้ายไปขวา ตามลำดับ จะเห็นได้ว่ามาส์กขนาดใหญ่จะทำให้วัตถุมีขนาดเล็กกว่าหายไปได้ และทำให้เส้นตรงที่เรียงขนานกันเกิดการเชื่อมต่อด้วยพื้นที่สีเทา

ตัวอย่างการใช้งาน gaussian mask แสดงดังรูปใน Example 6 และ Example 7 ค่าสัมประสิทธิ์ของมาส์กน้ำหนักเฉลี่ยนั้นจะมีค่าสูงสุดที่ตรงกลางและจะมีค่าลดลงเมื่อระยะทางเทียบกับตำแหน่งกลางมาส์กมีค่าสูงขึ้น เหตุผลที่เป็นเช่นนี้ก็เพื่อไม่ทำให้เกิดการเบลอภาพมากจนเกินไป เนื่องจากพิกเซลข้างเคียงที่อยู่ไกลๆ จะไม่มีผลมากเท่ากับพิกเซลที่อยู่ใกล้กว่า ซึ่งสอดคล้องกับหลักการของภาพทั่วไปที่ว่าพิกเซลที่อยู่ใกล้กันมักมีค่าระดับความเทาใกล้เคียงกัน

- **ตัวอย่างการทำให้ภาพเลือนโดยใช้ Linear Spatial Filter**

  Example 4 และ Example 5 เป็นตัวอย่างการทำให้ภาพเลือนโดยใช้มาส์ก average ส่วน Example 6 และ Example 7 เป็นตัวอย่างการทำให้ภาพเลือนโดยใช้มาส์ก Gaussian

Example 4: Blurring image with average masks to ‘pattern.tif’

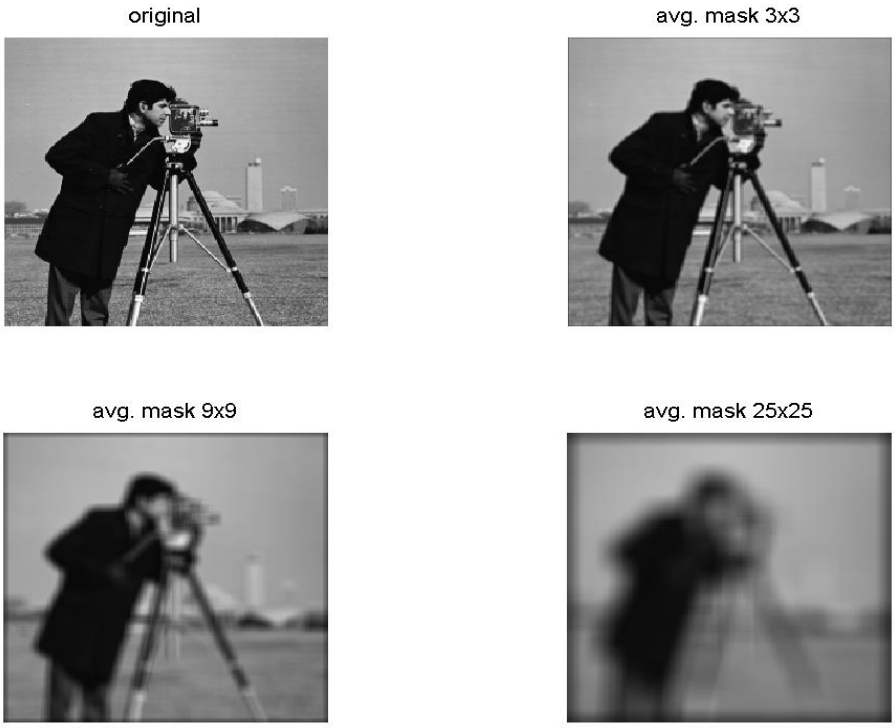
```
>> f = imread('pattern.tif');
>> f = im2double(f);
>> subplot(2,3,1); imshow(f), title('original');

>> w = fspecial('average', [3 3]);
>> gw3 = imfilter(f, w);
>> subplot(2,3,2); imshow(gw3), title('avg. mask 3x3');

>> w = fspecial('average', [5 5]);
>> gw5 = imfilter(f, w);
>> subplot(2,3,3); imshow(gw5), title('avg. mask 5x5');

>> w = fspecial('average', [9 9]);
>> gw9 = imfilter(f, w);
>> subplot(2,3,4); imshow(gw9), title('avg. mask 9x9');

>> w = fspecial('average', [15 15]);
>> gw15 = imfilter(f, w);
>> subplot(2,3,5); imshow(gw15), title('avg. mask 15x15');

>> w = fspecial('average', [35 35]);
>> gw35 = imfilter(f, w);
>> subplot(2,3,6); imshow(gw35), title('avg. mask 35x35');
```

Example 5: Blurring image with average masks to ‘cameraman.tif’

```
>> f = imread('cameraman.tif');
>> f = im2double(f);
>> subplot(2,2,1); imshow(f), title('original');
```

```
>> w = fspecial('average');
>> gw3 = imfilter(f, w);
>> subplot(2,2,2); imshow(gw3), title('avg. mask 3x3');

>> w = fspecial('average', 9);
>> gw9 = imfilter(f, w);
>> subplot(2,2,3); imshow(gw9), title('avg. mask 9x9');

>> w = fspecial('average', 25);
>> gw25 = imfilter(f, w);
>> subplot(2,2,4); imshow(gw25), title('avg. mask 25x25');
```

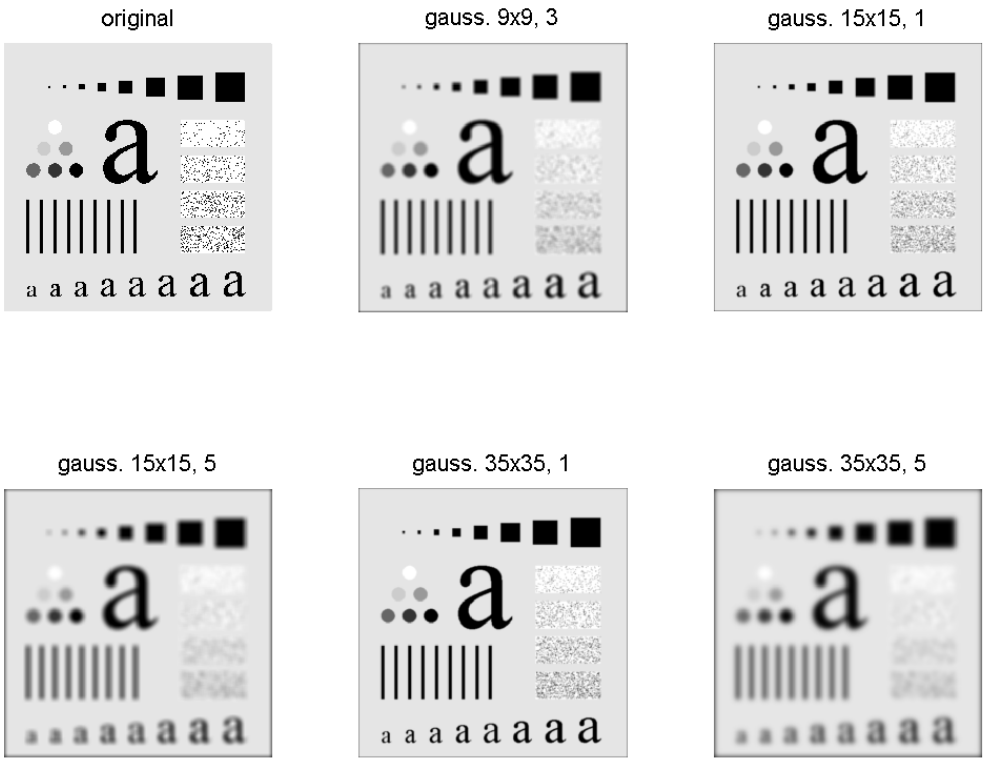
Result of Example 4



Result of Example 5

Example 6: Blurring image with gaussian masks to ‘pattern.tif’

```
>> f = imread('pattern.tif');
>> f = im2double(f);
>> subplot(2,3,1); imshow(f,[]), title('original');

>> w = fspecial('gaussian', [9 9], 3);
>> gw1 = imfilter(f, w);
>> subplot(2,3,2); imshow(gw1,[]), title('gauss. 9x9, 3');

>> w = fspecial('gaussian', [15 15], 1);
>> gw2 = imfilter(f, w);
>> subplot(2,3,3); imshow(gw2,[]), title('gauss. 15x15, 1');

>> w = fspecial('gaussian', [15 15], 5);
>> gw3 = imfilter(f, w);
>> subplot(2,3,4); imshow(gw3,[]), title('gauss. 15x15, 5');

>> w = fspecial('gaussian', [35 35], 1);
>> gw4 = imfilter(f, w);
>> subplot(2,3,5); imshow(gw4,[]), title('gauss. 35x35, 1');

>> w = fspecial('gaussian', [35 35], 5);
>> gw5 = imfilter(f, w);
>> subplot(2,3,6); imshow(gw5,[]), title('gauss. 35x35, 5');
```



Result of Example 6

Example 7: Blurring image with gaussian masks to ‘cameraman.tif’

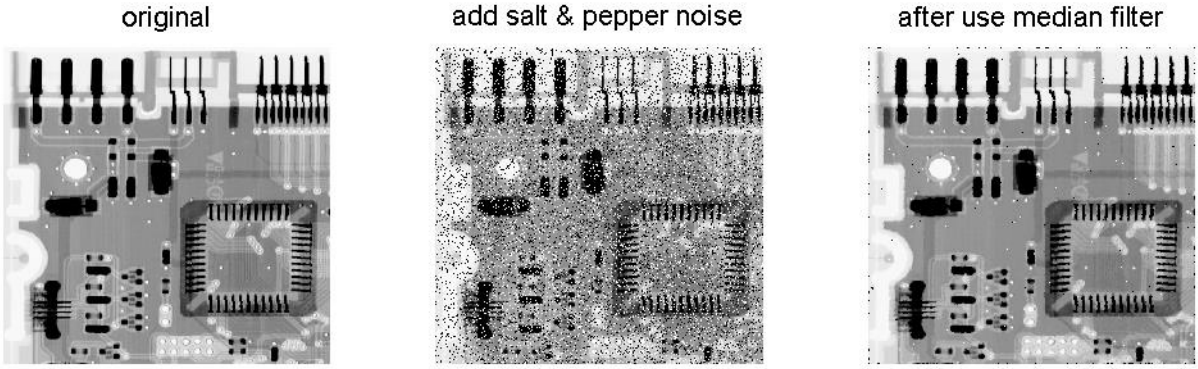
```
>> f = imread('cameraman.tif');
>> f = im2double(f);

>> w = fspecial('gaussian', [9 9], 1);
>> gw1 = imfilter(f, w);
>> subplot(2,2,1); imshow(gw1), title('gauss. 9x9, 1');

>> w = fspecial('gaussian', [9 9], 3);
>> gw2 = imfilter(f, w);
>> subplot(2,2,2); imshow(gw2), title('gauss. 9x9, 3');

>> w = fspecial('gaussian', [25 25], 1);
>> gw3 = imfilter(f, w);
>> subplot(2,2,3); imshow(gw3), title('gauss. 25x25, 1');

>> w = fspecial('gaussian', [25 25], 6);
>> gw4 = imfilter(f, w);
>> subplot(2,2,4); imshow(gw4), title('gauss. 25x25, 6');
```

Result of Example 7

- **ตัวอย่างการทำให้ภาพเลือนและการกำจัด noise โดยใช้ Nonlinear Spatial Filter**

Example 8 : Blurring image with gaussian masks to ‘cameraman.tif’

```
>> f = imread('board.tif');
>> subplot(1,3,1); imshow(f), title('original');

>> g = imread('st_pp_board.tif');
>> g = im2double(g);
>> subplot(1,3,2); imshow(g), title('add salt & pepper noise');

>> fn = medfilt2(g);
>> subplot(1,3,3); imshow(fn), title('after use median filter');
```



Result of Example 8

4.6 การทำให้ภาพคม (Sharpening)

การทำให้ภาพคมนั้นเป็นการกระทำที่ตรงกันข้ามกับการทำให้ภาพเบลอ กล่าวคือ เป็นการทำให้บริเวณที่มีการเปลี่ยนแปลงระดับความเทาสูง (บริเวณขอบภาพของวัตถุและสัญญาณรบกวน) มีความเด่นชัดมากขึ้น เราสามารถนำเทคนิคการทำให้ภาพคมมาใช้ปรับปรุงภาพที่ลักษณะเบลอจากการถ่ายภาพหรือความผิดพลาดในขั้นตอนการรับภาพ (Image Acquisition) ให้มีความคมชัดมากขึ้น การทำให้ภาพคมจึงเทียบเท่ากับการกรองเอาความถี่ต่ำออกไป เหลือไว้แต่ส่วนที่มีองค์ประกอบความถี่สูง หรือเรียกได้ว่าเป็นตัวกรองแบบผ่านสูง (High pass Filter)

หลักการของการทำให้ภาพคมชัดมีการใช้อัตราการเปลี่ยนแปลงของระดับความเทาของพิกเซลภายในมาส์ก หรือ อนุพันธ์เชิงระยะทาง (Spatial Differentiation) ซึ่งจะเห็นได้ว่าเป็นหลักการที่ตรงกันข้ามกับการเบลอภาพซึ่งทำการหาค่าผลรวม (Integration) ของค่าระดับความเทาของพิกเซลภายในมาส์ก

การทำให้ภาพคมชัดทำได้โดยการกำหนดให้ทำการขยายค่าระดับความเทาให้มีค่ามากขึ้นสำหรับบริเวณที่มีค่าอนุพันธ์สูง และในทางตรงกันข้ามบริเวณที่มีค่าอนุพันธ์ต่ำจะถูกลดค่าระดับความเทาลง

พิจารณาค่าอนุพันธ์อันดับที่ 1 (First – order Derivation) ของฟังก์ชัน f(x) ดังสมการต่อไปนี้

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

เราสามารถคำนวณหาค่าอนุพันธ์อันดับที่ 2 (Second – order Derivation) ได้ดังนี้

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \qquad \text{สมการ 2}$$

เนื่องจากสัญญาณภาพเป็นฟังก์ชันของตัวแปร x และ y ดังนั้น สมการที่ 1 และ 2 ในกรณีที่มีตัวแปรมากกว่าหนึ่งตัวแปรจะมีค่าเท่ากับสมการที่ 3 และ 4 ตามลำดับ โดยฟังก์ชันของทั้งสองสมการในที่นี้เป็นฟังก์ชันเชิงเส้นแบบต่อเนื่อง

Gradient operator : $\nabla f = \frac{\partial f(x,y)}{\partial x \partial y} = \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y}$ สมการ 3

Laplacian operation : $\nabla^2 f = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$ สมการ 4

ในกรณีที่เราพิจารณาภาพดิจิตอลซึ่งเป็นสัญญาณดีสครีต เราสามารถเขียนสมการ 4 ได้เป็น

$$\frac{\partial^2 f}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \qquad \text{สมการ 5}$$

$$\frac{\partial^2 f}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \qquad \text{สมการ 6}$$

เมื่อนำสมการ 5 3-37 และ 6 3-38 มารวมกัน จะได้

$$\nabla^2 f = [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)] \qquad \text{สมการ 7}$$

เมื่อนำค่าสัมประสิทธิ์ต่างๆของสมการ 7 ด้านขวามือมาใช้เป็นสัมประสิทธิ์ของมาส์กขนาด 3 x 3 เราจะได้มาส์กลาปลาเซียน (Laplacian Mask) ที่มีค่าดังแสดงในรูปที่ 1 รวมทั้งมาส์กที่ได้มีการปรับปรุงค่าสัมประสิทธิ์ของมาส์กลาปลาเซียน แสดงดังรูปที่ 2

0	1	0
1	-4	1
0	1	0

รูปที่ 1 มาส์กลาปลาเซียน

1	1	1
1	-8	1
1	1	1

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

รูปที่ 2 มาส์กลาปลาเซียนอื่นๆ

เนื่องจากภาพผลลัพธ์ที่ได้จากการใช้มาส์กลาปลาเซียนนั้นจะกำจัดส่วนที่เป็นพื้นภาพและส่วนที่มีการเปลี่ยนแปลงระดับความเทาต่ำ หากเรานำเอาภาพผลลัพธ์ที่ได้จากการใช้มาส์กลาปลาเซียนมารวมกับภาพตั้งต้นแล้วจะทำให้เราได้ภาพผลลัพธ์ที่มีรายละเอียดของภาพครบทุกส่วนอีกทั้งมีการทำให้บริเวณขอบภาพคมชัดมากยิ่งขึ้น การบวกผลลัพธ์จากการใช้มาส์กลาปลาเซียนกับภาพตั้งต้น เขียนเป็นสมการได้ดังนี้

$$g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) \\ f(x,y) + \nabla^2 f(x,y) \end{cases} \qquad \text{สมการ 8}$$

โดยสมการที่เป็นผลต่าง (-) นั้นจะใช้ในกรณีที่มาส์กลาปลาเซี่ยนมีค่าสัมประสิทธิ์ตรงกลางเป็นค่าลบและสมการที่เป็นผลบวก (+) นั้นจะใช้ในกรณีที่มาส์ก ลาปลาเซียนมีค่าสัมประสิทธิ์ตรงกลางเป็นค่าบวก เราสามารถกระจายสมการ 8 ได้เป็น

$$g(x,y) = f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) + 4f(x,y)] \qquad \text{สมการ 9}$$

$$= 5f(x,y) - [f(x+1,y) + f(x-1,y) + f(x,y+1) + f(xy-1)]$$

และเมื่อนำค่าสัมประสิทธิ์ในสมการที่ 9 มาเขียนเป็นมาส์กจะได้ดังรูปที่ 3

0	-1	0
-1	5	-1
0	-1	0

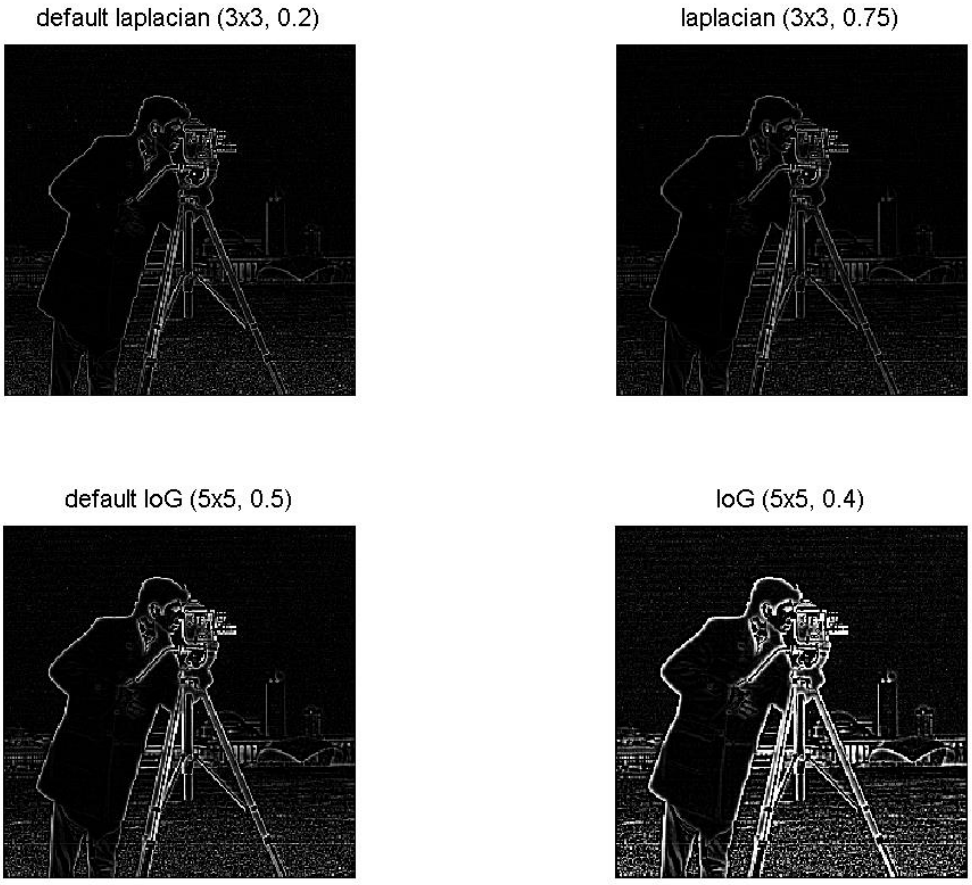
-1	-1	-1
-1	9	-1
-1	-1	-1

รูปที่ 3 มาส์กผลรวมค่าตั้งต้นกับลาปลาเซียน

- **ตัวอย่างการทำให้ภาพการทำให้ภาพคมโดยใช้ Linear Spatial Filter**

Example 9 : Sharpening image of ‘cameraman.tif’ by laplacian and log mask

```
>> f = imread('cameraman.tif');
>> f = im2double(f);

>> w = fspecial('laplacian');
>> fw1 = imfilter(f, w);
>> subplot(2,2,1); imshow(fw1), title('default laplacian (3x3, 0.2)');

>> w = fspecial('laplacian',0.75);
>> fw2 = imfilter(f, w);
>> subplot(2,2,2); imshow(fw2), title('laplacian (3x3, 0.75)');

>> w = fspecial('log');
>> fw3 = imfilter(f, w);
>> subplot(2,2,3); imshow(fw3), title('default loG (5x5, 0.5)');

>> w = fspecial('log', [5 5], 0.4);
>> fw4 = imfilter(f, w);
>> subplot(2,2,4); imshow(fw4), title('loG (5x5, 0.4)');
```



Result of Example 9