Chapter 3: พื้นฐานการประมวลผลภาพดิจิตอลด้วยโปรแกรม MATLAB (Fundamental of DIPUM)

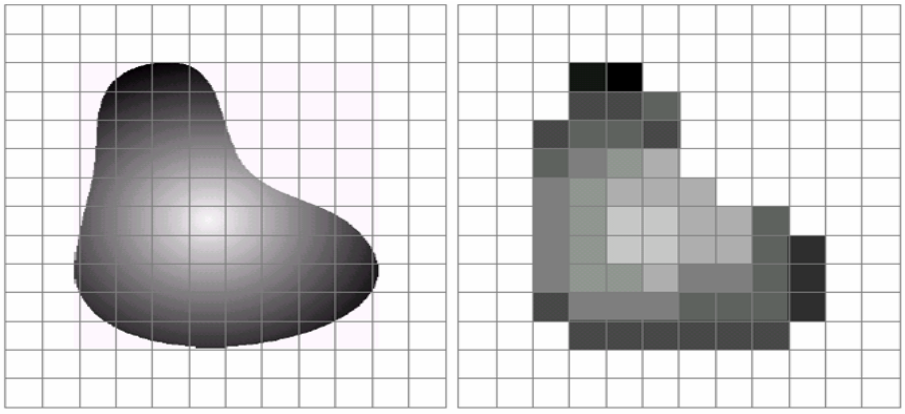
3.1 การแทนรูปภาพดิจิตอล (Digital Image Representation)

An image may be defined as a two-dimensional function, *f (x, y)*, where *x* and *y* are *spatial* (plane) *coordinates*, and the amplitude of *f* at any pair of coordinates *(x, y)* is called the *intensity* of the image at that point.

รูปภาพสามารถนิยามได้ในรูปของฟังก์ชัน *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

An image may be continuous with respect to the *x*- and *y*-coordinates, and also in amplitude. Converting such an image to digital form requires that the coordinates, as well as the amplitude, be digitized. Digitizing the coordinate values is called *sampling*; digitizing the amplitude values is called *quantization*. Thus, when *x, y*, and the amplitude values of *f* are all finite, discrete quantities, we call the image a *digital image*

จากลักษณะของพิกัด *(x, y)* และค่าของ amplitude แล้วรูปภาพถือว่ามีลักษณะที่ต่อเนื่อง ซึ่งการแปลงรูปภาพที่ต่อเนื่องไปเป็นแบบดิจิตอลต้องทำการดิจิไทซ์ (digitized) ทั้งค่าของพิกัดจุด (coordinates) และค่าของ amplitude การ ดิจิไทซ์ ค่าของพิกัดจุด (coordinates) นั้นเรียกว่าการ *sampling* ส่วนการ ดิจิไทซ์ ค่าของ amplitude เรียกว่าการ *quantization* ดังนั้นเมื่อ ค่า *x, y,* และค่า amplitude ของ *f* มีขนาดที่จำกัด นั่นคือเป็นค่าแบบดิสครีต เราจึงเรียกรูปภาพที่ผ่านการ ดิจิไทซ์ แล้วว่า *digital image* ดังตัวอย่างในรูปที่ 3.1(b) ซึ่งได้มาจากการ *sampling และการ quantization* ภาพที่ต่อเนื่องในรูปที่ 3.1(a)



(a) *(b)*

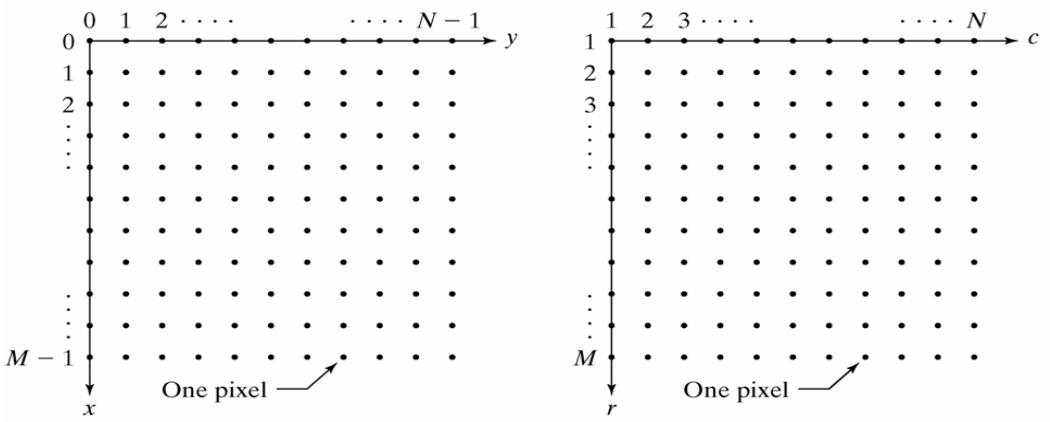
รูปที่ 3.1 (a) ภาพที่มีลักษณะต่อเนื่อง (b) ภาพดิจิตอลที่ได้จากการ sampling และการ quantization

- **ข้อตกลงเกี่ยวกับรูปแบบของพิกัด (Coordinate Conventions)**

ผลลัพธ์ที่ได้จากการ sampling และ quantization รูปภาพนั้นถือว่าเป็นเมตริกซ์ของตัวเลข สมมุติรูปภาพ *f(x, y)* ผ่านการ sampling มาแล้วมีขนาด *M* แถว และ *N* คอลัมน์ นั่นคือเป็นรูปภาพที่มีขนาด *M* x *N* โดยค่าของพิกัดจุด *(x,y)* เป็นค่าแบบดิสครีต หนังสือเกี่ยวกับ image processing โดยทั่วไปนั้น กำหนดให้พิกัดเริ่มต้นของ

รูปภาพอยู่ที่ตำแหน่ง *(x, y)* = (0,0) ดังนั้นช่วงของค่าพิกัดในแกน *x* จึงอยู่ที่ 0 ถึง *M*-1 และ ช่วงของค่าพิกัดในแกน *y* อยู่ที่ 0 ถึง *N-1* ดังรูปที่ 3.2(a)

แต่รูปแบบของพิกัดจุกที่ใช้ใน Image Processing Toolbox (IPT) ของ MATLAB นั้นแตกต่างกันออกไปคือ ใช้สัญลักษณ์ *(r,c)* แทนแทนที่จะใช้ *(x,y)* โดย *r* แทน row และ *c* แทน column และกำหนดให้พิกัดเริ่มต้นของรูปภาพอยู่ที่ตำแหน่ง *(r, c)* = (1,1) แทน ช่วงของค่าพิกัดในแนว *r* จึงอยู่ที่ 1 ถึง *M* และช่วงของค่าพิกัดในแนว *c* อยู่ที่ 1 ถึง *N* ดังแสดงในรูปที่ 3.2(b)



(a) *(b)*

รูปที่ 3.2 (a) รูปแบบของพิกัดที่ใช้ในหนังสือ image processing ทั่วไป (b) รูปแบบของพิกัดที่ใช้ใน Image Processing Toolbox (IPT) ของ MATLAB

- **การแทนรูปภาพด้วย Matrices (Images as Matrices)**

จากรูปแบบของพิกัดที่ใช้ในหนังสือ image processing ทั่วไปสอดคล้องตามในรูป 3.1(a) นั้นเมื่อนำค่าของ amplitude *f(x,y)* ของแต่ละพิกัดมาแทนด้วย matrix จึงแสดงได้ดังด้านล่าง

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

ซึ่งสมาชิกแต่ละตัวใน matrix นั้นเรียกว่า *element, picture element, pixel,* หรือ *pel.*

ส่วนรูปแบบของพิกัดที่ใช้ใน MATLAB ซึ่งสอดคล้องตามในรูป 3.1(b) นั้นเมื่อนำค่าของ amplitude *f(x,y)* ของแต่ละพิกัดมาแทนด้วย matrix จึงแสดงได้ดังด้านล่าง

$$\mathtt{f} = \begin{bmatrix} \mathtt{f(1,1)} & \mathtt{f(1,2)} & \cdots & \mathtt{f(1,N)} \\ \mathtt{f(2,1)} & \mathtt{f(2,2)} & \cdots & \mathtt{f(2,N)} \\ \vdots & \vdots & & \vdots \\ \mathtt{f(M,1)} & \mathtt{f(M,2)} & \cdots & \mathtt{f(M,N)} \end{bmatrix}$$

โดยที่ค่า ณ ตำแหน่ง *f*(0,0) ที่อ้างอิงในหนังสือ image processing ทั่วไปจะเท่ากันกับค่า ณ ตำแหน่ง f(1,1) ของ MATLAB

3.2 การอ่านไฟล์รูปภาพเข้ามาใน MATLAB (Reading Images)

- ใช้ฟังก์ชัน imread ในการอ่านไฟล์รูปภาพเข้ามาใน MATLAB

  Syntax:
  imread('filename')

  - ตัวอย่าง:

    ถ้าภาพ rose_original.tif อยู่ใน current directory ใช้
    >> f1 = imread('rose_original.tif');

    ถ้าภาพ rose_original.tif ไม่ได้อยู่ใน current directory แต่อยู่ใน folder Myimages ของ drive C ใช้
    >> f1 = imread('C:\Myimages\rose_original.tif');

    ตัวอย่างข้างต้นเป็นการใช้คำสั่ง imread อ่าน file ภาพ ชื่อ rose_original.tif เข้ามาเก็บไว้ที่ตัว f1 ดังนั้น ผลลัพธ์ที่ได้คือจะ มีตัวแปร f1 เกิดขึ้นใน Workspace ดังรูปแบบ

Name	Size	Bytes	Class
f1	1024x1024	1048576	uint8 array

    ซึ่งข้อมูลที่แสดงใน Workspace ดังกล่าว สามารถใช้คำสั่ง whos ที่ Command Windows ได้เช่นกันดังรูปแบบ

    >> whos f1
    ผลลัพธ์ที่ได้คือ

Name	Size	Bytes	Class
f1	1024x1024	1048576	uint8 array

    Grand total is 1048576 elements using 1048576 bytes

    ถ้าต้องการดูเฉพาะขนาดของภาพสามารถใช้คำสั่ง

    >> size(f1)

    ผลลัพธ์ที่ได้คือ

    ```
    ans =
        1024    1024
    ```

    ถ้าต้องการเอาขนาดของภาพมาเก็บไว้ที่ตัวแปร M และ N เมื่อ M คือแถว และ N คือคอลัมน์ สามารถใช้คำสั่ง

    >> [M N] = size(f1)

    ผลลัพธ์ที่ได้คือ

    ```
    M =
        1024
    N =
        1024
    ```

3.3 การแสดงผลรูปภาพ (Displaying Images)

- ใช้ฟังก์ชัน imshow ในการแสดงผลรูปภาพออกหน้าจอหลังจากได้อ่านไฟล์รูปนั้นเข้ามาแล้วด้วยฟังก์ชัน imread

  Syntax:
  imshow(f,G)
  เมื่อ f คือตัวแปรที่เก็บข้อมูลภาพ (image array) และ G คือ ค่าตัวเลขที่บอกว่าต้องการให้แสดงผลออกมาด้วยค่าสีกี่ระดับ (intensity level)

  Syntax:
  imshow(f)
  เป็นการแสดงผลโดยใช้ค่า G เป็นค่า default นั่นคือ 256 ระดับ

  Syntax:
  imshow(f, [low high])
  เป็นการแสดงผลโดยกำหนดให้เป็นสีดำ (ค่า = 0) ในทุกจุดที่มีค่าน้อยกว่าหรือเท่ากับ low และสีขาว (ค่า = 255) ในทุกจุดที่มีค่ามากกว่าหรือเท่ากับ high

  Syntax:
  imshow(f, [])
  เป็นการแสดงผลโดยให้ค่า low คือค่าต่ำสุดของ array f และค่า high คือค่าสูงสุดของ array f

  - ตัวอย่าง:

    ```
    >> imshow(f1, 2)
    >> imshow(f1, 10)
    >> imshow(f1, 256), figure, imshow(f1)
    >> imshow(f1, [100 150])
    >> imshow(f1, [])
    ```

  - ทดลองพิมพ์คำสั่งต่อไปนี้ แล้วสังเกตผลลัพธ์:

    ```
    >> g1 = imread('chest_xray.tif');
    >> imshow(g1), figure, imshow(g1, [])
    ```

3.4 การเขียนไฟล์รูปภาพ (Writing Images)

- ใช้ฟังก์ชัน imwrite ในการขียนไฟล์รูปภาพ (เมื่อต้องการ save ไฟล์รูปภาพเก็บไว้)

  Syntax:
  imwrite(f,'filename')

  - ตัวอย่าง:

    ```
    >> imwrite(f1, 'newrose.jpg');
    >> imfinfo('rose_original.tif')
    >> imfinfo('newrose.jpg')
    ```

3.5 Data Classes

Lists the various data classes supported by MATLAB and IPT for representing pixel values

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

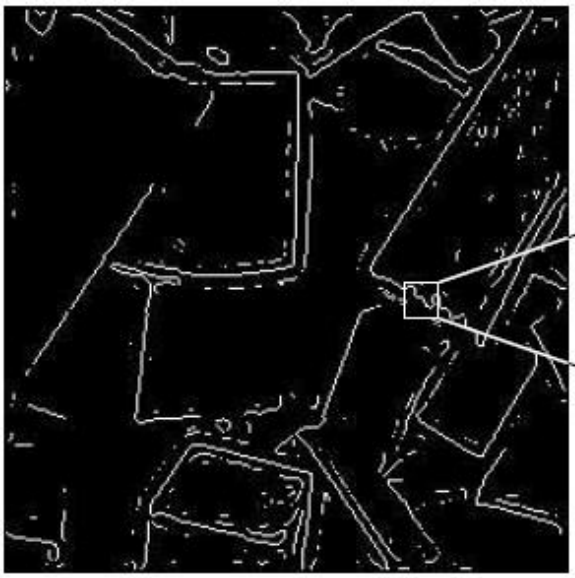
3.6 Image Types

สามารถแบ่งชนิดของรูปภาพดิจิตอลออกเป็น 4 ชนิดด้วยกันคือ Binary image, Intensity image (Grayscale Image), RGB image และ Indexed (color) image

Image Type	Storage Class	Interpretation
Binary	logical	An array of zeros (0) and ones (1)
Intensity[1] (Grayscale)	double	An array of floating-point values. The typical range of values is [0, 1].
	uint8 or uint16	An array of integers. The typical range of values is [0, 255] or [0, 65535].
Indexed[1]	double	An array of integers in the range [1, *p*]
	uint8 or uint16	An array of integers in the range [0, *p*-1]
RGB (Truecolor)	double	An m-by-n-by-3 array of floating-point values in the range [0, 1].
	uint8 or uint16	An m-by-n-by-3 array of integers in the range [0, 255] or [0, 65535].

- **Binary Images**

Binary images have a very specific meaning in MATLAB. A binary image is a logical array of 0s and 1s. Thus, an array of 0s and 1s whose values are of data class, say, uint8 is not considered a binary image in MATLAB.



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

- **Intensity Images**

230	229	232	234	235	232	148
237	236	236	234	233	234	152
255	255	255	251	230	236	161
99	90	67	37	94	247	130
222	152	255	129	129	246	132
154	199	255	150	189	241	147
216	132	162	163	170	239	122

- **RGB Images**

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Red

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Blue

- **Index Color Images**

4	5	5	5	5	5
5	4	5	5	6	6
5	5	5	0	8	9
5	5	5	5	11	11
5	5	5	8	16	20
8	11	11	26	33	20
11	20	33	33	58	37

Indices

0.1211	0.1211	0.1416
0.1807	0.2549	0.1729
0.2197	0.3447	0.1807
0.1611	0.1768	0.1924
0.2432	0.2471	0.1924
0.2119	0.1963	0.2002
0.2627	0.2588	0.2549
0.2197	0.2432	0.2588
⋮	⋮	⋮

Colour map

3.7 Converting between Data Classes and Image Types

- **Converting between Data Classes**

Name	Converts Input to:	Valid Input Image Data Classes
`im2uint8`	`uint8`	`logical`, `uint8`, `uint16`, and `double`
`im2uint16`	`uint16`	`logical`, `uint8`, `uint16`, and `double`
`mat2gray`	`double` (in range [0, 1])	`double`
`im2double`	`double`	`logical`, `uint8`, `uint16`, and `double`
`im2bw`	`logical`	`uint8`, `uint16`, and `double`

- **Converting between Image Classes and Types**

Function	Description
dither	Create a binary image from a grayscale intensity image by dithering; create an indexed image from an RGB image by dithering
gray2ind	Create an indexed image from a grayscale intensity image
grayslice	Create an indexed image from a grayscale intensity image by thresholding
im2bw	Create a binary image from an intensity image, indexed image, or RGB image, based on a luminance threshold
ind2gray	Create a grayscale intensity image from an indexed image
ind2rgb	Create an RGB image from an indexed image
mat2gray	Create a grayscale intensity image from data in a matrix, by scaling the data
rgb2gray	Create a grayscale intensity image from an RGB image
rgb2ind	Create an indexed image from an RGB image

3.8 Array Indexing

- ตัวอย่างการใช้ array indexing ใน MATLAB กับการเรียกดูข้อมูลภาพ:

```
>> f1 = imread('rose_original.tif');
>> imshow(f1)

>> f2 = f1(end:-1:1, :);
>> figure, imshow(f2)

>> f3 = f1(257:768, 257:768);
>> figure, imshow(f3)

>> f4 = f1(1:2:end, 1:2:end);
>> figure, imshow(f4)

>> f4 = f1(1:16:end, 1:16:end);
>> figure, imshow(f4)

>> plot(f1(512, :))
```

3.9 Operators in MATLAB

Table of array and matrix operators in MATLAB

Operator	Name	MATLAB Function	Comments and Examples
+	Array and matrix addition	plus(A, B)	a + b, A + B, or a + A.
–	Array and matrix subtraction	minus(A, B)	a – b, A – B, A – a, or a – A.
.*	Array multiplication	times(A, B)	C = A.*B, C(I, J) = A(I, J)*B(I, J).
*	Matrix multiplication	mtimes(A, B)	A*B, standard matrix multiplication, or a*A, multiplication of a scalar times all elements of A.
./	Array right division	rdivide(A, B)	C = A./B, C(I, J) = A(I, J)/B(I, J).
.\	Array left division	ldivide(A, B)	C = A.\B, C(I, J) = B(I, J)/A(I, J).
/	Matrix right division	mrdivide(A, B)	A/B is roughly the same as A*inv(B), depending on computational accuracy.
\	Matrix left division	mldivide(A, B)	A\B is roughly the same as inv(A)*B, depending on computational accuracy.
.^	Array power	power(A, B)	If C = A.^B, then C(I, J) = A(I, J)^B(I, J).
^	Matrix power	mpower(A, B)	See online help for a discussion of this operator.
.'	Vector and matrix transpose	transpose(A)	A.'. Standard vector and matrix transpose.
'	Vector and matrix complex conjugate transpose	ctranspose(A)	A'. Standard vector and matrix conjugate transpose. When A is real A.' = A'.
+	Unary plus	uplus (A)	+A is the same as 0 + A.
–	Unary minus	uminus (A)	–A is the same as 0 – A or –1*A.
:	Colon		Discussed in Section 2.8.

Table of relational operator

Operator	Name
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
~=	Not equal to

Table of logical operator

Operator	Name
&	AND
|	OR
~	NOT

Table of logical function

Function	Comments
`xor` (exclusive OR)	The `xor` function returns a 1 only if both operands are logically different; otherwise `xor` returns a 0.
`all`	The `all` function returns a 1 if all the elements in a vector are nonzero; otherwise `all` returns a 0. This function operates columnwise on matrices.
`any`	The `any` function returns a 1 if any of the elements in a vector is nonzero; otherwise `any` returns a 0. This function operates columnwise on matrices.

- ทดลองพิมพ์คำสั่งต่อไปนี้แล้วสังเกตผลลัพธ์:

```
>> A = rand(4,4)
>> A = A*10
>> A = round(A)
>> B = rand(4,4)
>> B = B*10
>> B = round(B)

>> C = A + B
>> D = A - B

>> E = A .* B
>> F = A * B

>> G = A ./ B
>> H = A / B

>> I = A .^2
>> J = A .^B
>> K = A^2

>> L = A’
>> M = A.’

>> N = -A
>> O =  +A

>> P = A>= 4
>> all(P)
>> any(P)

>> ~P
>> P | ~P
>> P & ~P
>> xor(P, ~P)
>> xor(P, P)
```

3.10 Image Arithmetic

Table of the image arithmetic supported by IPT in MATLAB

Function	Description
`imadd`	Adds two images; or adds a constant to an image.
`imsubtract`	Subtracts two images; or subtracts a constant from an image.
`immultiply`	Multiplies two images, where the multiplication is carried out between pairs of corresponding image elements; or multiplies a constant times an image.
`imdivide`	Divides two images, where the division is carried out between pairs of corresponding image elements; or divides an image by a constant.
`imabsdiff`	Computes the absolute difference between two images.
`imcomplement`	Complements an image. See Section 3.2.1.
`imlincomb`	Computes a linear combination of two or more images. See Section 5.3.1 for an example.

- **Adding Image:** imadd

  Syntax:
  Z = imadd(X,Y)

```
>> X = uint8([ 255 0 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imadd(X,Y)
Z =
255    50   125
94   255   150
```

- ตัวอย่าง: Add a constant to an image.

```
>> I = imread('rice.tif');
>> J = imadd(I,100);
>> subplot(1,2,1), imshow(I)
>> subplot(1,2,2), imshow(J)
```

- ตัวอย่าง: Add two images together and specify an output class

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

>> J = imread('cameraman.tif');
>> imshow(J)

>> K = imadd(I, J);
>> L = imadd(I, J, 'uint16');
>> imshow(K,[]), figure, imshow(L,[])

>> subplot(1,2,1); imshow(K,[]), xlabel('unit8');
>> subplot(1,2,2); imshow(L,[]), xlabel('uint16');
```

- **Subtracting Image:** imsubtract

  Syntax:

  Z = imsubtract(X,Y)

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imsubtract(X,Y)
Z =
205    0   25
0   175   50
```

- ตัวอย่าง: Subtract a constant to an image.

```
>> I = imread('rice.tif');
>> J = imsubtract(I,85);
>> subplot(1,2,1), imshow(I)
>> subplot(1,2,2), imshow(J)
```

- ตัวอย่าง: Estimate and subtract the background of the rice image:

```
>> I = imread('rice.tif');
>> blocks = blkproc(I,[32 32],'min(x(:))');
>> background = imresize(blocks,[256 256],'bilinear');
>> Ip = imsubtract(I,background);
>> subplot(1,2,1); imshow(I,[]), xlabel('Before');
>> subplot(1,2,2); imshow(Ip,[]), xlabel('After');
```

- **Multiplying Images:** immultiply

  Syntax:

  Z = immultiply(X,Y)

```
>> I = imread('moon.tif');
>> J = immultiply(I,0.5);
>> subplot(1,2,1), imshow(I)
>> subplot(1,2,2), imshow(J)
```

- **Deviding Image:** imdivide

  Syntax:

  Z = imdivide(X,Y)

```
>> I = imread('rice.tif');
>> blocks = blkproc(I,[32 32],'min(x(:))');
>> background = imresize(blocks,[256 256],'bilinear');
>> Ip = imdivide(I,background);
>> imshow(Ip,[]) % [] = let imshow scale data automatically
```

- **Complement Image**: imcomplement

  Syntax:

  IM2 = imcomplement(IM)

- ตัวอย่าง: Create the complement of a uint8 array.

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> X2 = imcomplement(X)
>> X2 =
    0   245   180
  211    30   155
```

- ตัวอย่าง: Reverse black and white in a binary image.

```
>> bw = imread('text.tif');
>> bw2 = imcomplement(bw);
>> subplot(1,2,1),imshow(bw)
>> subplot(1,2,2),imshow(bw2)
```

- ตัวอย่าง: Create the complement of an intensity image.

```
>> I = imread('bonemarr.tif');
>> J = imcomplement(I);
>> imshow(I), figure, imshow(J)
```

➢ **Absolute Different Image:** imabsdiff

Syntax:

```
Z = imabsdiff(X,Y)
```

```
>> X = uint8([ 255 10 75; 44 225 100]);
>> Y = uint8([ 50 50 50; 50 50 50 ]);
>> Z = imabsdiff(X,Y)
Z =
205    40    25
6   175    50
```

- ตัวอย่าง: Display the absolute difference between a filtered image and the original.

```
>> I = imread('cameraman.tif');
>> J = uint8(filter2(fspecial('gaussian'), I));
>> K = imabsdiff(I,J);
>> imshow(K,[])
```