Convolution Image

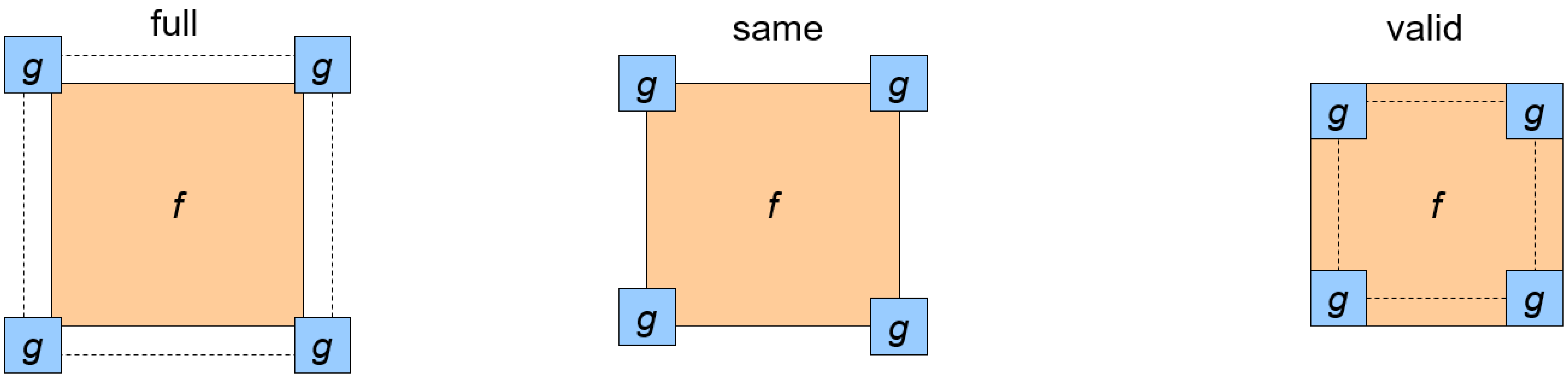
Image Processing

Convolution

MATLAB's Image Processing Toolbox (IPT) has two built-in functions that can be used to implement 2D convolution:

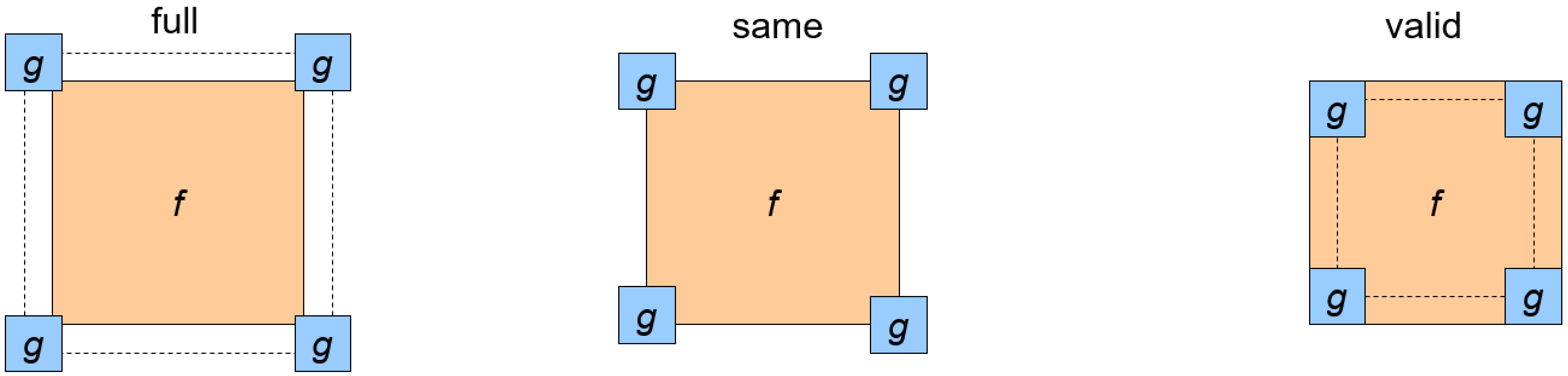
- `conv2`: It computes the 2D convolution between two matrices. In addition to the two matrices, it takes a third parameter that specifies the size of the output:
  - `full`: Returns the full 2D convolution (default).
  - `same`: Returns the central part of the convolution of the same size as A.
  - `valid`: Returns only those parts of the convolution that are computed without the zero-padded edges.
- `filter2`: It rotates the convolution mask (which is treated as a 2D FIR filter) 180° in each direction to create a convolution kernel and then calls `conv2` to perform the convolution operation.

Note: using zero padding



Convolution

- C = conv2(A,B) computes the two-dimensional convolution of matrices A and B.
- The size of C is determined as follows:
    - if [ma,na] = size(A),[mb,nb] = size(B), and [mc,nc] = size(C),
    - then mc = max([ma+mb-1,ma,mb]) and nc = max([na+nb-1,na,nb]).



Convolution: imfilter

Linear filters are implemented in MATLAB using two functions: `imfilter` and—optionally—`fspecial`.

The syntax for `imfilter` is

```
g = imfilter(f, h, mode, boundary_options, size_options);
```

where

- `f` is the input image.
- `h` is the filter mask.
- `mode` can be either `'conv'` or `'corr'`, indicating, respectively, whether filtering will be done using convolution or correlation (which is the default);
- `boundary_options` refer to how the filtering algorithm should treat border values. There are four possibilities:
  1. `X`: The boundaries of the input array (image) are extended by padding with a value `X`. This is the default option (with `X` = 0).
  2. `'symmetric'`: The boundaries of the input array (image) are extended by mirror-reflecting the image across its border.

Convolution: imfilter (Cont'd)

3. `'replicate'`: The boundaries of the input array (image) are extended by replicating the values nearest to the image border.
4. `'circular'`: The boundaries of the input array (image) are extended by implicitly assuming the input array is periodic, that is, treating the image as one period of a 2D periodic function.

- `size_options`: There are two options for the size of the resulting image: `'full'` (output image is the full filtered result, that is, the size of the extended/padded image) or `'same'` (output image is of the same size as input image), which is the default.
- `g` is the output image.

Convolution vs Correlation

Command Window

```
>> A

A =

     1     2     3
     4     5     6
     7     8     9

>> kernel

kernel =

    -1     0     1
    -2     0     2
    -1     0     1
```

```
>> conv2(A, kernel, 'valid')

ans =

    -8

>> filter2(A, kernel, 'valid')

ans =

     8
```

Convolution

Command Window

```
>> a = [0 0 0 1 0 0 0];
>> f = [1 2 3 4 5];
>> g = imfilter(a, f, 'full', 'conv');
>> g

g =

     0     0     0     1     2     3     4     5     0     0     0
```

Command Window

```
>> g = imfilter(a, f, 'same', 'conv')

g =

     0     1     2     3     4     5     0
```

Konvolusi

		0	0	0	1	0	0	0		
5	4	3	2	1						
		0	1							

		0	0	0	1	0	0	0		
	5	4	3	2	1					
		0	1							

		0	0	0	1	0	0	0		
		5	4	3	2	1				
		0	1	2						

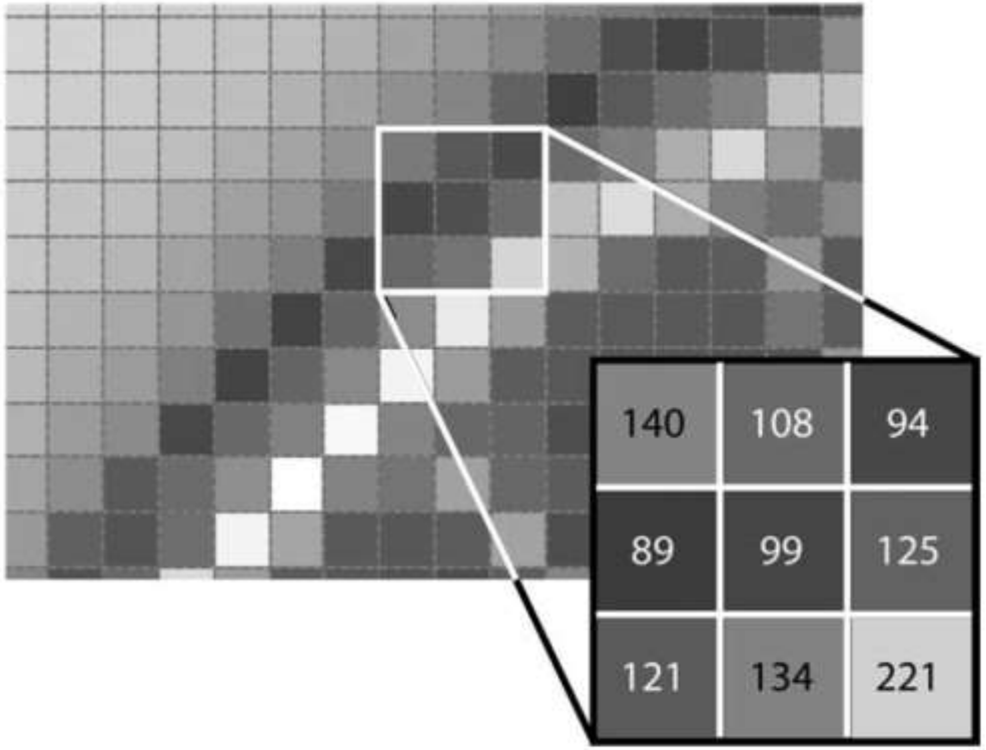
		0	0	0	1	0	0	0		
			5	4	3	2	1			
		0	1	2	3					

		0	0	0	1	0	0	0		
				5	4	3	2	1		
		0	1	2	3	4				

		0	0	0	1	0	0	0		
					5	4	3	2	1	
		0	1	2	3	4	5			

*	*	0	0	0	1	0	0	0	*	*
						5	4	3	2	1
*	*	0	1	2	3	4	5	0	*	*

Correlation



–1	0	1
–2	0	2
–1	0	1

```
x = [140 108 94;89 99 125;121 134 221]
y = [-1 0 1;-2 0 2;-1 0 1]
z = imfilter(x,y,'corr')
```

Correlation

```
x =

   140   108    94
    89    99   125
   121   134   221

>> xcircular = repmat(x, 3, 3)

xcircular =

   140   108    94   140   108    94   140   108    94
    89    99   125    89    99   125    89    99   125
   121   134   221   121   134   221   121   134   221
   140   108    94   140   108    94   140   108    94
    89    99   125    89    99   125    89    99   125
   121   134   221   121   134   221   121   134   221
   140   108    94   140   108    94   140   108    94
    89    99   125    89    99   125    89    99   125
   121   134   221   121   134   221   121   134   221
```

```
>> z = imfilter(x, y, 'corr', 'circular', 'full')

z =

    -4  -186   190    -4  -186
    41   -85    44    41   -85
    -1  -125   126    -1  -125
    -4  -186   190    -4  -186
    41   -85    44    41   -85

>> z = imfilter(xcircular, y, 'corr', 'same')

z =

   315   -56    54     2   -56    54     2   -56  -315
   440   126    -1  -125   126    -1  -125   126  -440
   475   190    -4  -186   190    -4  -186   190  -475
   449    44    41   -85    44    41   -85    44  -449
   440   126    -1  -125   126    -1  -125   126  -440
   475   190    -4  -186   190    -4  -186   190  -475
   449    44    41   -85    44    41   -85    44  -449
   440   126    -1  -125   126    -1  -125   126  -440
   367   236   -36  -200   236   -36  -200   236  -367
```

Convolution

`fspecial` is an IPT function designed to simplify the creation of common 2D image filters. Its syntax is `h = fspecial(type, parameters)`, where

- `h` is the filter mask.
- `type` is one of the following:
  - `'average'`: Averaging filter
  - `'disk'`: Circular averaging filter
  - `'gaussian'`: Gaussian low-pass filter
  - `'laplacian'`: 2D Laplacian operator
  - `'log'`: Laplacian of Gaussian (LoG) filter
  - `'motion'`: Approximates the linear motion of a camera
  - `'prewitt'` and `'sobel'`: horizontal edge-emphasizing filters
  - `'unsharp'`: unsharp contrast enhancement filter

Convolution

- `parameters` are optional parameters that vary depending on the `type` of filter, for example, mask size, standard deviation (for `'gaussian'` filter), and so on. See the IPT documentation for full details.

Command Window

```
>> h = fspecial('prewitt')

h =

     1     1     1
     0     0     0
    -1    -1    -1
```

```
>> h = fspecial('gaussian', 5, 0.4)

h =

    0.0000    0.0000    0.0000    0.0000    0.0000
    0.0000    0.0016    0.0371    0.0016    0.0000
    0.0000    0.0371    0.8450    0.0371    0.0000
    0.0000    0.0016    0.0371    0.0016    0.0000
    0.0000    0.0000    0.0000    0.0000    0.0000
```

Smoothing Filters in The Spatial Domain

```
clear; clc;
I = imread('cameraman.tif');
figure, subplot(1,2,1), imshow(I), title('Original Image');
fn = fspecial('average');
I_new = imfilter(I, fn);
subplot(1,2,2), imshow(I_new), title('Filtered Image');
```

Image Smoothing

The mean filter we just implemented was a uniform filter—all coefficients were equivalent. The nonuniform version of the mean filter gives the center of the mask (the pixel in question) a higher weighted value, while all other coefficients are weighted by their distance from the center. This particular mask cannot be generated by the `fspecial` function, so we must create it ourselves.

```
fn2 = [1 2 1; 2 4 2; 1 2 1]
fn2 = fn2 * (1/16)
```

1/9 ×

1	1	1
1	1	1
1	1	1

1/16 ×

1	2	1
2	4	2
1	2	1

FIGURE 10.14 Uniform and nonuniform averaging masks.

Uniform vs Non-Uniform Average Filter

```
I_new2 = imfilter(I,fn2);
figure, subplot(1,2,1), imshow(I_new), title('Uniform Average');
subplot(1,2,2), imshow(I_new2), title('Non-uniform Average');
```

Gaussian Filter: Image Smoothing

The Gaussian filter is similar to the nonuniform averaging filter in that the coefficients are not equivalent. The coefficient values, however, are not a function of their distance from the center pixel, but instead are modeled from the Gaussian curve.


```
fn_gau = fspecial('gaussian',9,1.5);
figure, bar3(fn_gau,'b'), ...
title('Gaussian filter as a 3D graph');
```

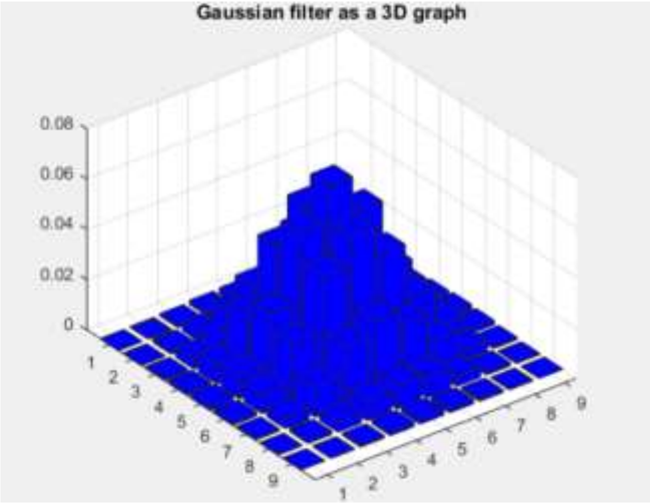


Image Smoothing

The convolution mask for a 3 × 3 mean filter is given by

$$h(x, y) = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{10.9}$$

Modified Mask Coefficients The mask coefficients from equation (10.9) can be modified, for example, to give more importance to the center pixel and its 4-connected neighbors:

$$h(x, y) = \begin{bmatrix} 0.075 & 0.125 & 0.075 \\ 0.125 & 0.2 & 0.125 \\ 0.075 & 0.125 & 0.075 \end{bmatrix} \tag{10.10}$$

The Laplacian

The Laplacian of an image $f(x, y)$ is defined as

$$\nabla^2(x, y) = \frac{\partial^2(x, y)}{\partial x^2} + \frac{\partial^2(x, y)}{\partial y^2} \tag{10.12}$$

where the second derivatives are usually approximated—for digital signals—as

$$\frac{\partial^2(x, y)}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y) \tag{10.13}$$

and

$$\frac{\partial^2(x, y)}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y) \tag{10.14}$$

The Laplacian

which results in a convenient expression for the Laplacian expressed as a sum of products:

$$\nabla^2(x, y) = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y) \tag{10.15}$$

This expression can be implemented by the convolution mask below:

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

Image Sharpening: Lapacian Kernel

$$g(x,y)=\begin{cases} f(x,y)-\nabla^2 f(x,y) & \textit{If the center coefficient is negative} \\ f(x,y)+\nabla^2 f(x,y) & \textit{If the center coefficient is positive} \end{cases}$$

Where f(x,y) is the original image

$\nabla^2 f(x,y)$ is Laplacian filtered image

g(x,y) is the sharpen image

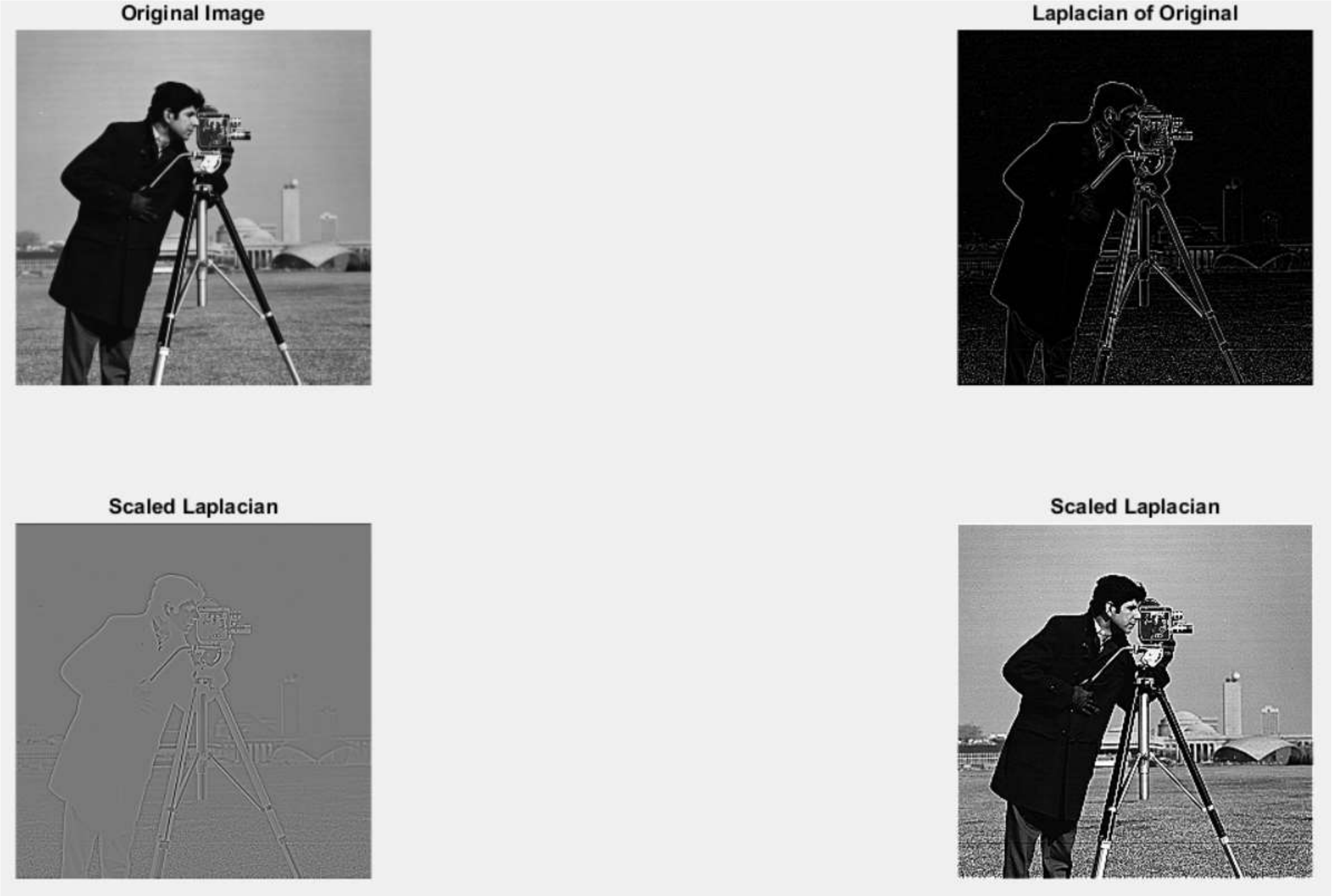
```
clear; clc;
I = imread('cameraman.tif');
Id = im2double(I);
figure,
subplot(2, 2, 1), imshow(Id), title('Original Image');

f = fspecial('laplacian', 0);
I_filt = imfilter(Id,f);
subplot(2, 2, 2), imshow(I_filt), title('Laplacian of Original');
subplot(2, 2, 3), imshow(I_filt, []), title('Scaled Laplacian');
%I_sharp = imsubtract(Id, I_filt);
I_sharp = imadd(Id, -1*I_filt);
subplot(2, 2, 4), imshow(I_sharp), title('Scaled Laplacian');
```

`fspecial` creates Laplacian filters using

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

$$\nabla^2 = \frac{4}{(\alpha+1)}\begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

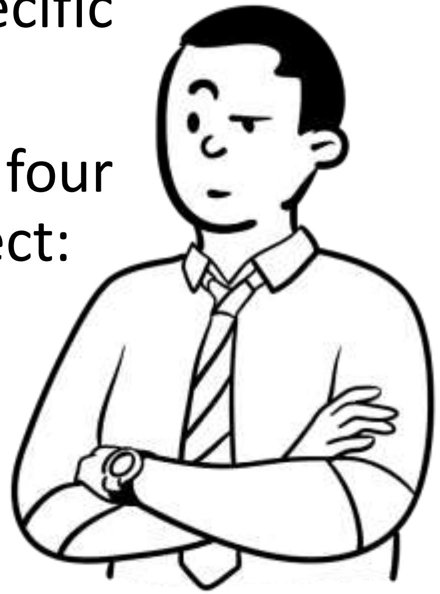
Image Sharpening



Directional Difference Filters

- Directional difference filters are similar to the Laplacian high-frequency filter discussed earlier. The main difference is that—as their name suggests—directional difference filters emphasize edges in a specific direction.
- There filters are usually called *emboss filters*. There are four representative masks that can be used to implement the emboss effect:

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



Emboss


emboss horizontal
emboss diagonal 45 degree
emboss vertical
emboss diagonal 135 degree

The Laplacian: Composite

$$g(x,y) = f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1) + 4f(x,y)$$

$$g(x,y) = 5f(x,y) - f(x+1,y) - f(x-1,y) - f(x,y+1) - f(x,y-1)$$

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

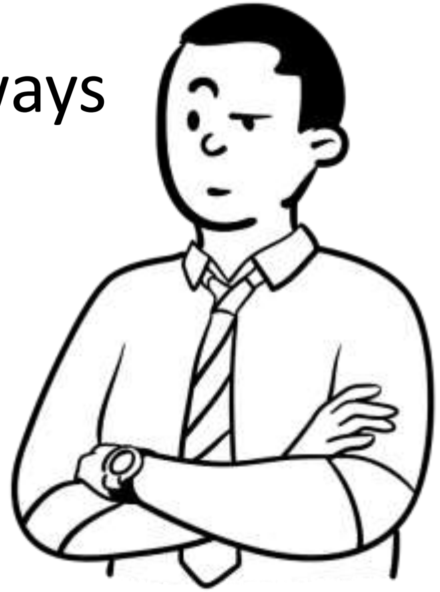
-1	-1	-1
-1	9	-1
-1	-1	-1

```
f2 = [0 -1 0; -1 5 -1; 0 -1 0];
I_sharp2 = imfilter(Id, f2);
figure,
subplot(1, 2, 1), imshow(Id), title('Original Image');
subplot(1, 2, 2), imshow(I_sharp2), title('Composite Laplacian');
```

The Laplacian: Composite

Unsharp Masking

- Unsharp masking is a simple process of substracting a blurred image from its original to generate a sharper image.
- Although the concept is straightforward, there are three ways it can be implemented.



Unsharp #1

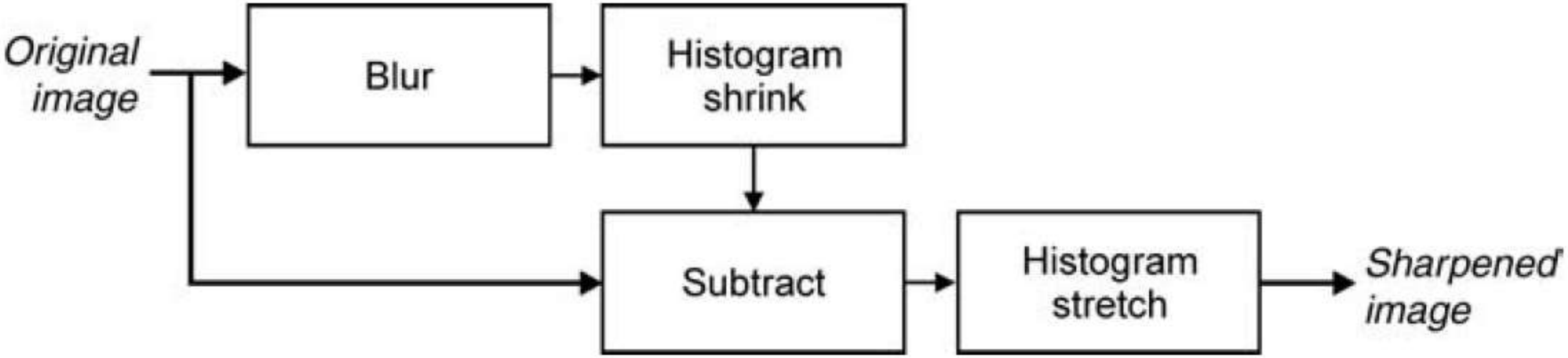


FIGURE 10.16 Unsharp masking process including histogram adjustment.

```
I = imread('moon.tif');
f_blur = fspecial('average',5);
I_blur = imfilter(I,f_blur);
figure, subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I_blur), title('Blurred Image');
```

Unsharp #1

- We must now shrink the histogram of the blurred image. The amount by which we shrink the histogram will ultimately determine the level of enhancement in the final result. In our case, we will scale the histogram to range between 0.0 and 0.4, where the full dynamic grayscale range is [0.0 1.0].

```
I_blur_adj = imadjust(I_blur,stretchlim(I_blur),[0 0.4]);
   I_sharp = imsubtract(I,I_blur_adj);
I_sharp_adj = imadjust(I_sharp);
subplot(1,3,3), imshow(I_sharp_adj), title('Sharp Image');
```

Unsharp #2

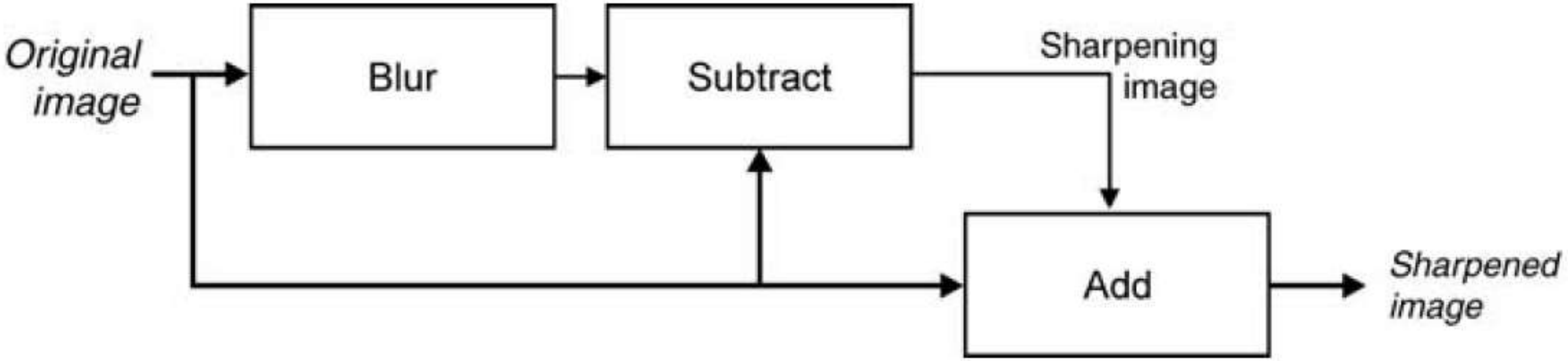


FIGURE 10.17 Unsharp masking process with sharpening image.

```
I_sharpening = imsubtract(I,I_blur);
```

```
I_sharp2 = imadd(I,I_sharpening);
figure, subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I_sharp2), title('Sharp Image');
```

Unsharp #3

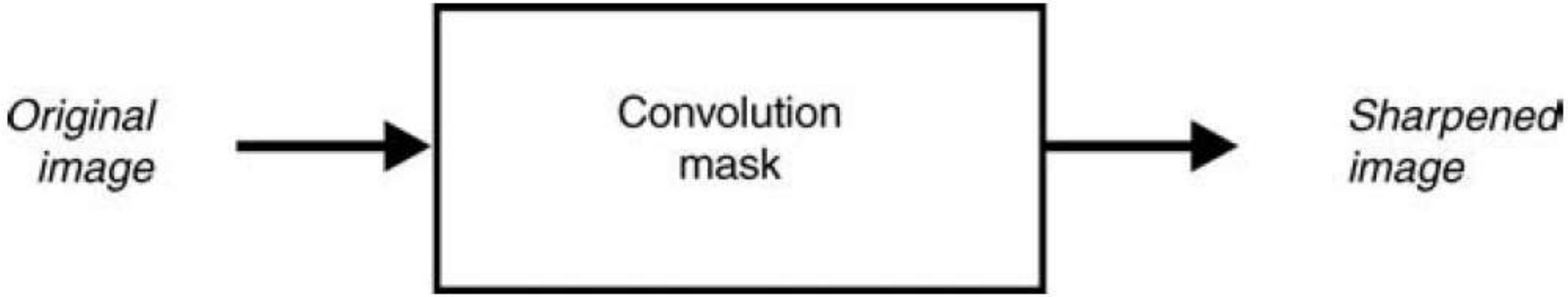


FIGURE 10.18 Unsharp masking process using convolution mask.

```
f_unsharp = fspecial('unsharp');
```

```
I_sharp3 = imfilter(I,f_unsharp);
figure, subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I_sharp3), title('Sharp Image');
```

High Boost Filtering

- High-boost filtering is a sharpening technique that involves creating a sharpening image and adding it to the original image.
- The mask used to create the sharpening image is illustrated in Figure 10.19. Note that there are two versions of the mask: one that does not include the corner pixels and nother that does.

0	–1	0
–1	A+4	–1
0	–1	0

–1	–1	–1
–1	A+8	–1
–1	–1	–1

FIGURE 10.19 High-boost masks with and without regard to corner pixels.

High Boost Filter

- Create a high-boost mask (where A = 1) and apply it to the moon image

```
f_hb = [0 -1 0; -1 5 -1; 0 -1 0];
I_sharp4 = imfilter(I,f_hb);
figure, subplot(1,2,1), imshow(I), title('Original Image');
subplot(1,2,2), imshow(I_sharp4), title('Sharp Image');
```

You may have noticed that when $A = 1$, the high-boost filter generalizes to the composite Laplacian mask. As the value of A increases, the output image starts to resemble an image multiplied by a constant.

High Boost Filter

```
f_hb2 = [0 -1 0; -1 7 -1; 0 -1 0];
I_sharp5 = imfilter(I,f_hb2);
I_mult = immultiply(I,3);
figure, subplot(1,3,1), imshow(I), title('Original Image');
subplot(1,3,2), imshow(I_sharp5), title('High Boost, A = 3');
subplot(1,3,3), imshow(I_mult), title('Multiplied by 3');
```

Show that a high-boost mask when A = 3 looks similar to the image simply multiplied by 3.

Lampiran

`fspecial` creates Gaussian filters using

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2 + n_2^2)}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{h_g(n_1, n_2)}{\sum_{n_1} \sum_{n_2} h_g}$$

`fspecial` creates Laplacian of Gaussian (LoG) filters using

$$h_g(n_1, n_2) = e^{\frac{-(n_1^2 + n_2^2)}{2\sigma^2}}$$

$$h(n_1, n_2) = \frac{(n_1^2 + n_2^2 - 2\sigma^2) h_g(n_1, n_2)}{2\pi\sigma^6 \sum_{n_1} \sum_{n_2} h_g}$$

`fspecial` creates Laplacian filters using

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

$$\nabla^2 = \frac{4}{(\alpha+1)} \begin{bmatrix} \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \\ \frac{1-\alpha}{4} & -1 & \frac{1-\alpha}{4} \\ \frac{\alpha}{4} & \frac{1-\alpha}{4} & \frac{\alpha}{4} \end{bmatrix}$$

`fspecial` creates averaging filters using

```
ones(n(1),n(2))/(n(1)*n(2))
```

Lampiran

Histogram Sliding

This technique consists of simply adding or subtracting a constant brightness value to all pixels in the image. The overall effect will be an image with comparable contrast properties, but higher or lower (respectively) average brightness.

Histogram Stretching

$$s = \frac{r - r_{\min}}{r_{\max} - r_{\min}} \times (L - 1)$$

Histogram Shrinking

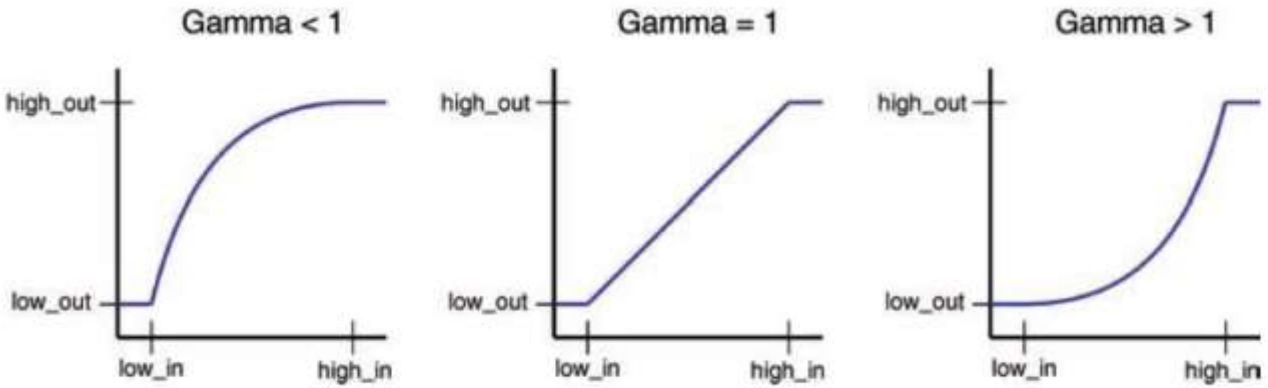
Mathematically,

$$s = \left[\frac{s_{\max} - s_{\min}}{r_{\max} - r_{\min}}\right](r - r_{\min}) + s_{\min}$$

Lampiran

Histogram stretching and shrinking can be achieved through use of the `imadjust` function. The syntax for the function is as follows:

```
J = imadjust(I,[low_in; high_in],[low_out; high_out], gamma)
```



Any values below `low_in` and above `high_in` are *clipped* or simply mapped to `low_out` and `high_out`, respectively. Only values in between these limits are affected by the curve. Gamma values less than 1 create a weighted curve toward the brighter range, and gamma values greater than 1 weight toward the darker region. The default value of gamma is 1.

Terima Kasih