Digital Image Processing

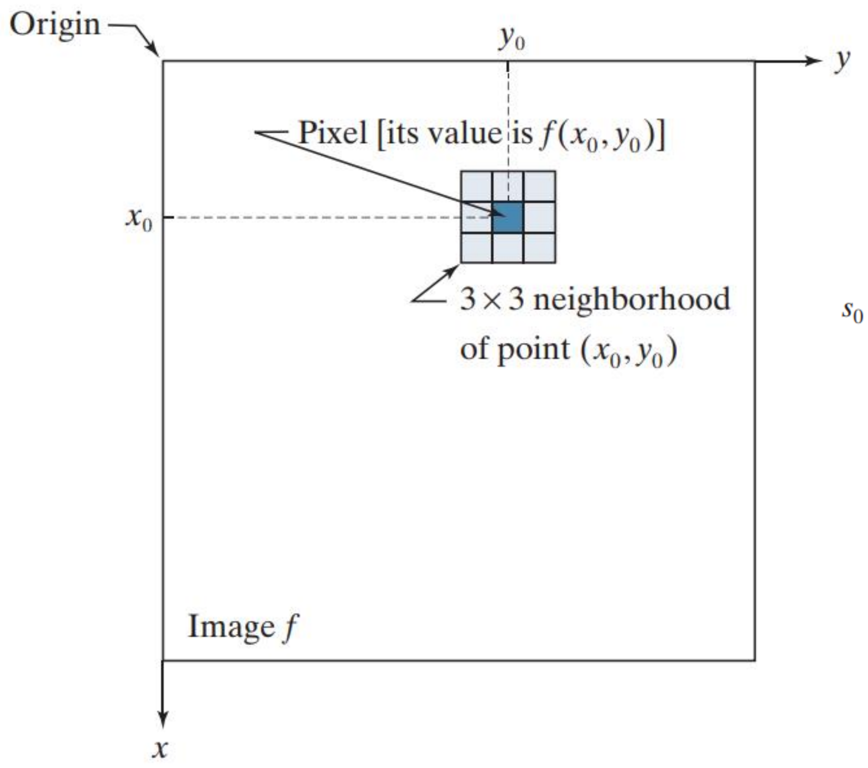
Chapter 3
Intensity Transformations and Spatial Filtering

Yasin Fakhar

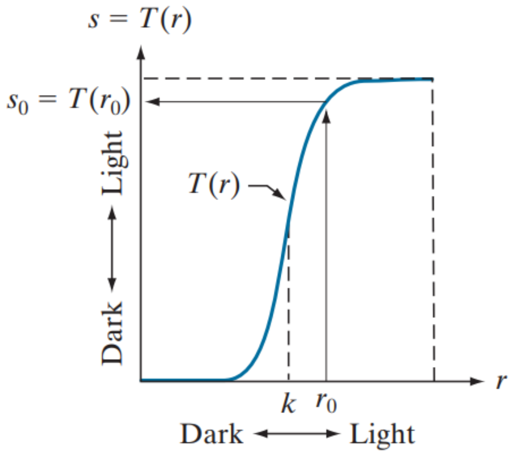
The term spatial domain refers to the image plane itself, and image processing methods in this category are based on direct manipulation of pixels in an image.

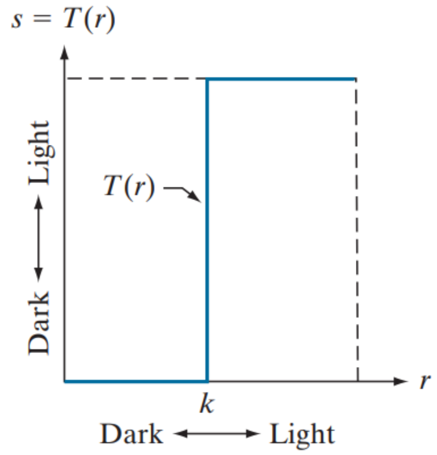
categories of spatial processing

- Intensity transformations : operate on single pixel
    - contrast manipulation
    - image thresholding
- Spatial filtering : performs operations on the neighbourhood of every pixels
    - image smoothing
    - sharpening


Origin
y_0
y
Pixel [its value is $f(x_0, y_0)$]
x_0
3×3 neighborhood of point (x_0, y_0)
Image f
x

$$s = T(r)$$





contrast manipulation image thresholding

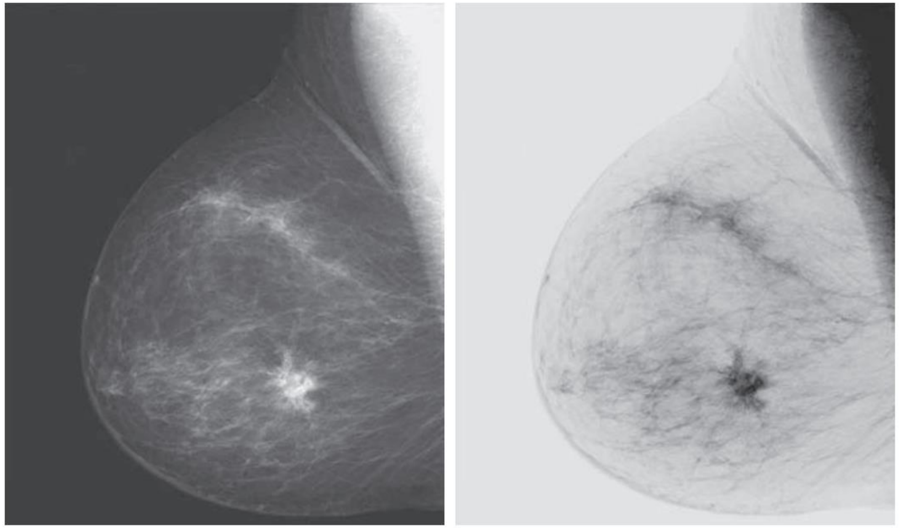
Generally we use contrast manipulation for image enhancement

Some Basic Intensity Transformation Functions

Image Negative

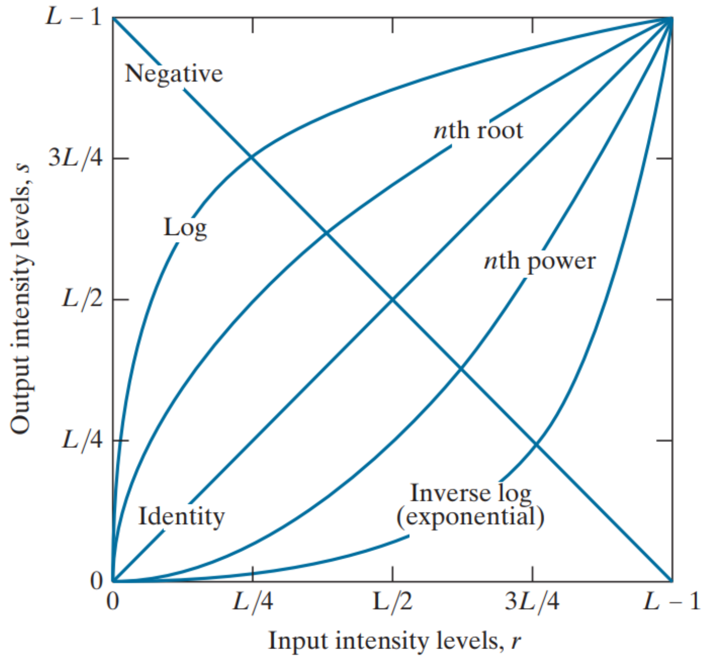
$S = L - 1 - r$

Intensity : $[0 , L-1]$



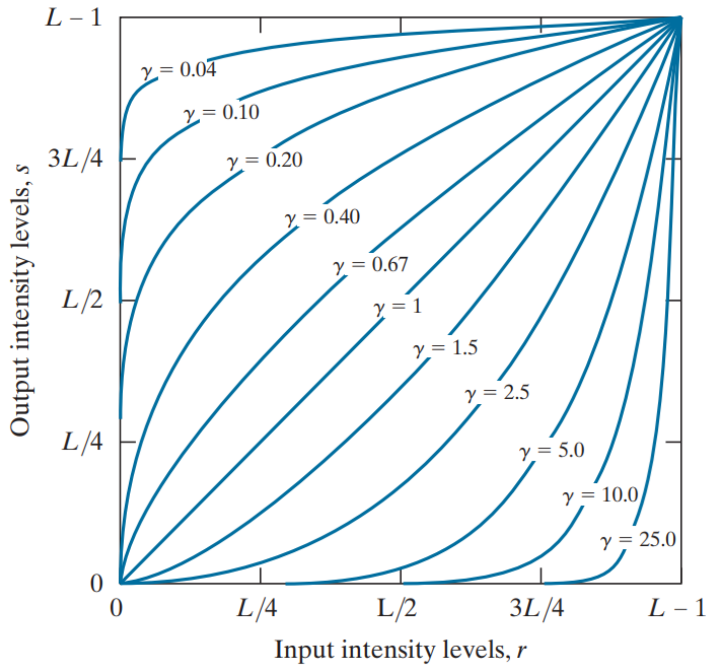
Log Transformation

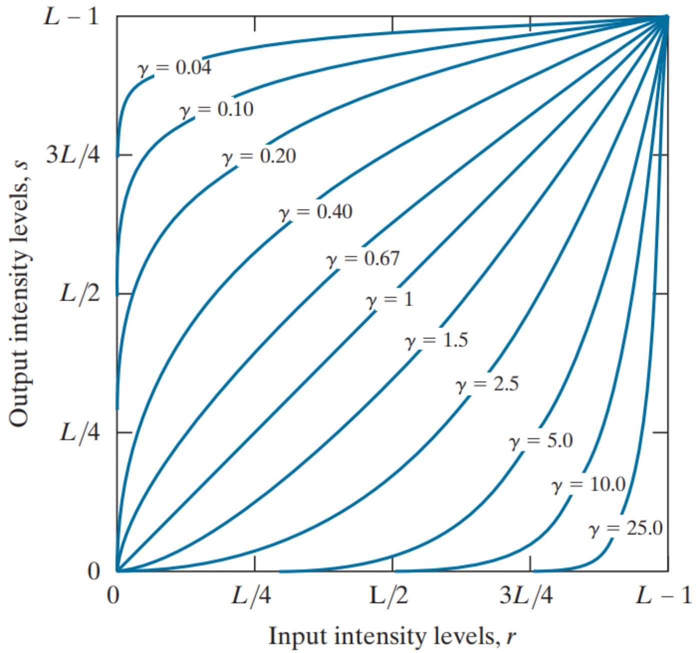
$$s = c\log(1+r)$$



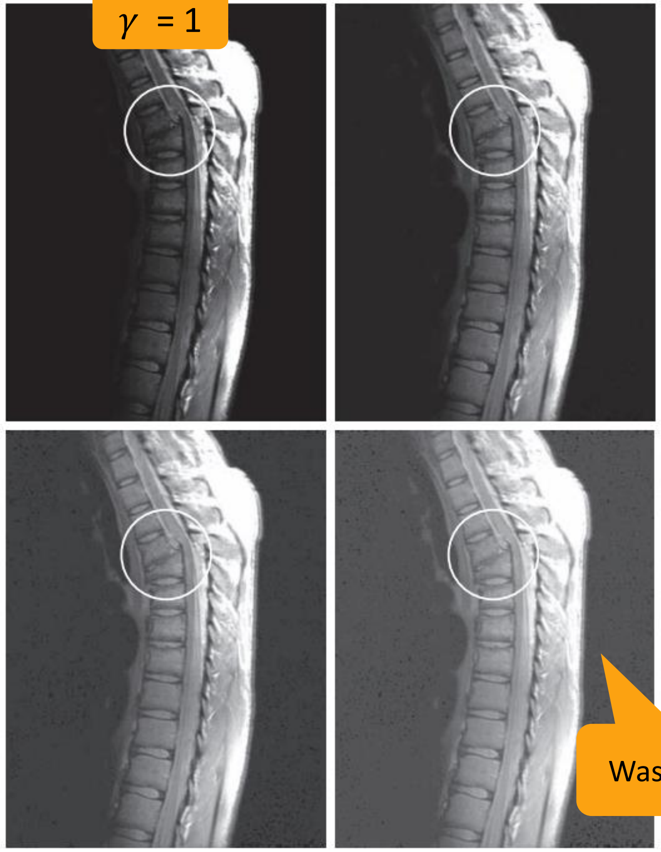
Power-law (Gamma) Transformation

$$s = cr^{\gamma} \qquad c,\gamma > 0$$



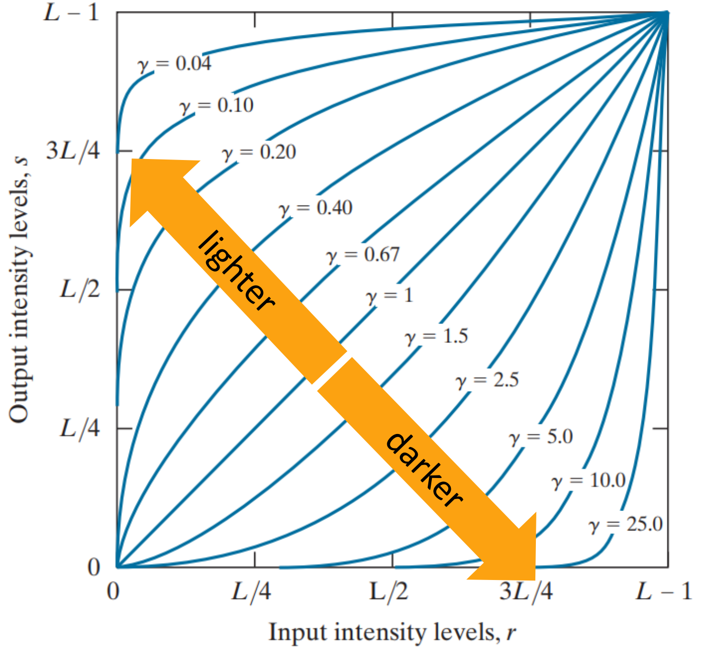


$\gamma = 1$

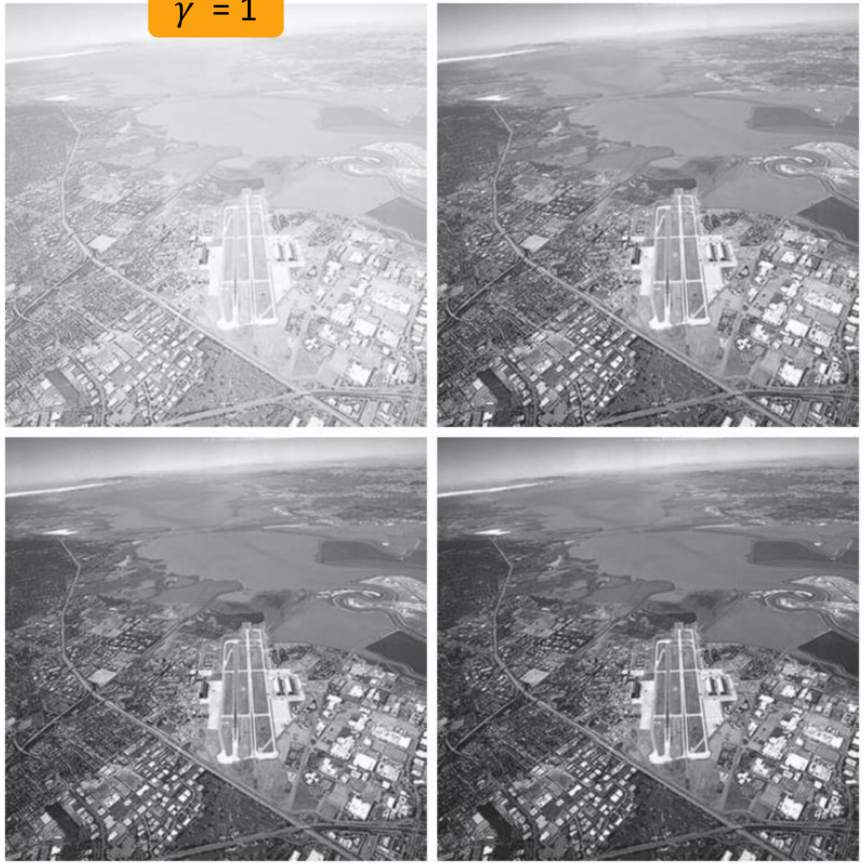


$\gamma = 0.6$ $\gamma = 0.4$ $\gamma = 0.3$

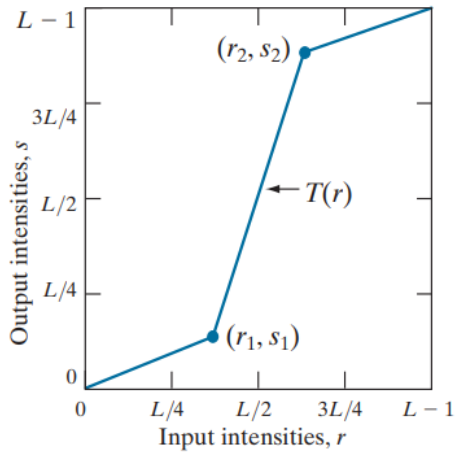
Washted-out

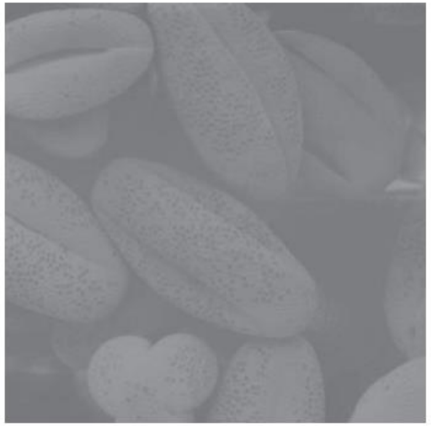


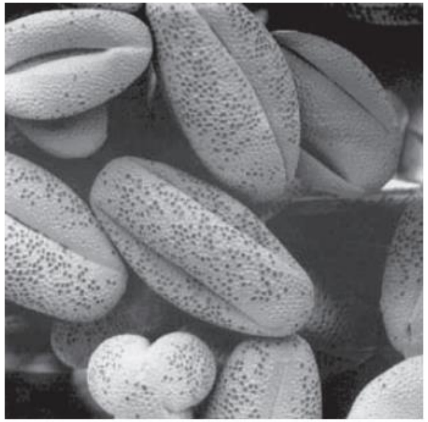
$\gamma = 1$



$\gamma = 3$ $\gamma = 4$ $\gamma = 5$




Original image


Contrast stretching

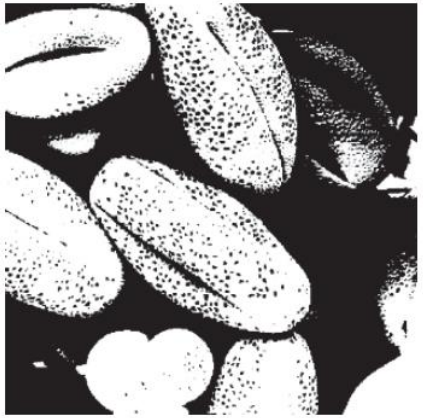
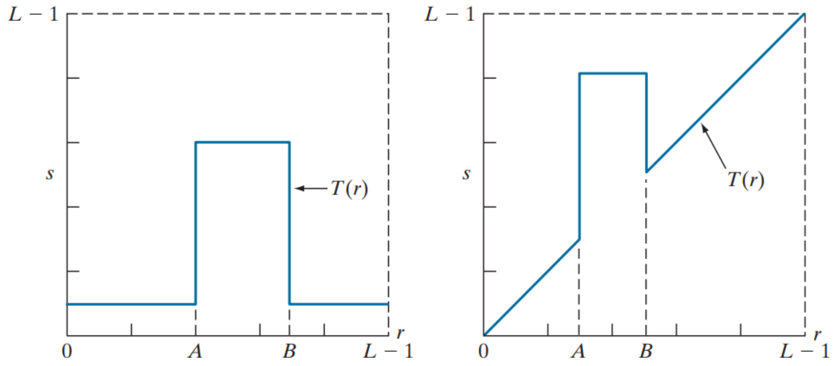
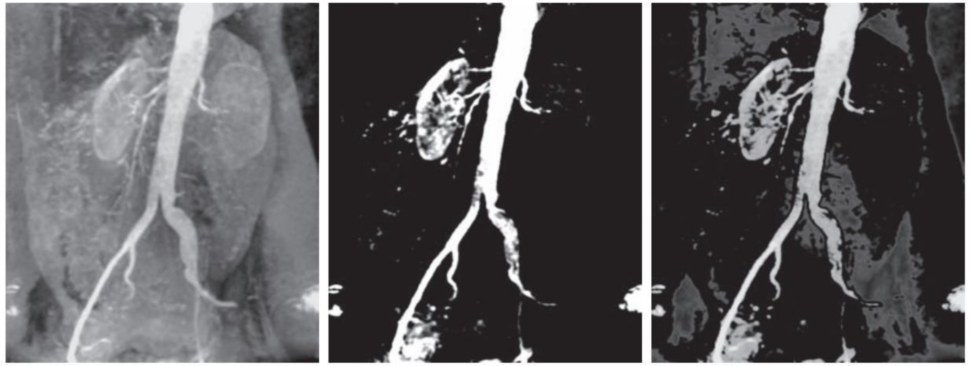


Image thresh holding

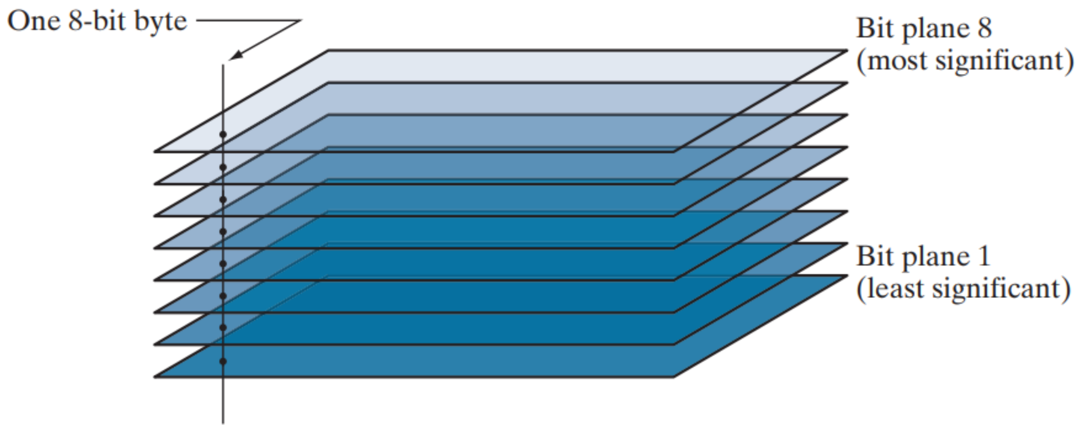
There are applications in which it is of interest to highlight a specific range of intensities in an image. Some of these applications include enhancing features in satellite imagery, such as masses of water, and enhancing flaws in X-ray images. The method, called intensity-level slicing

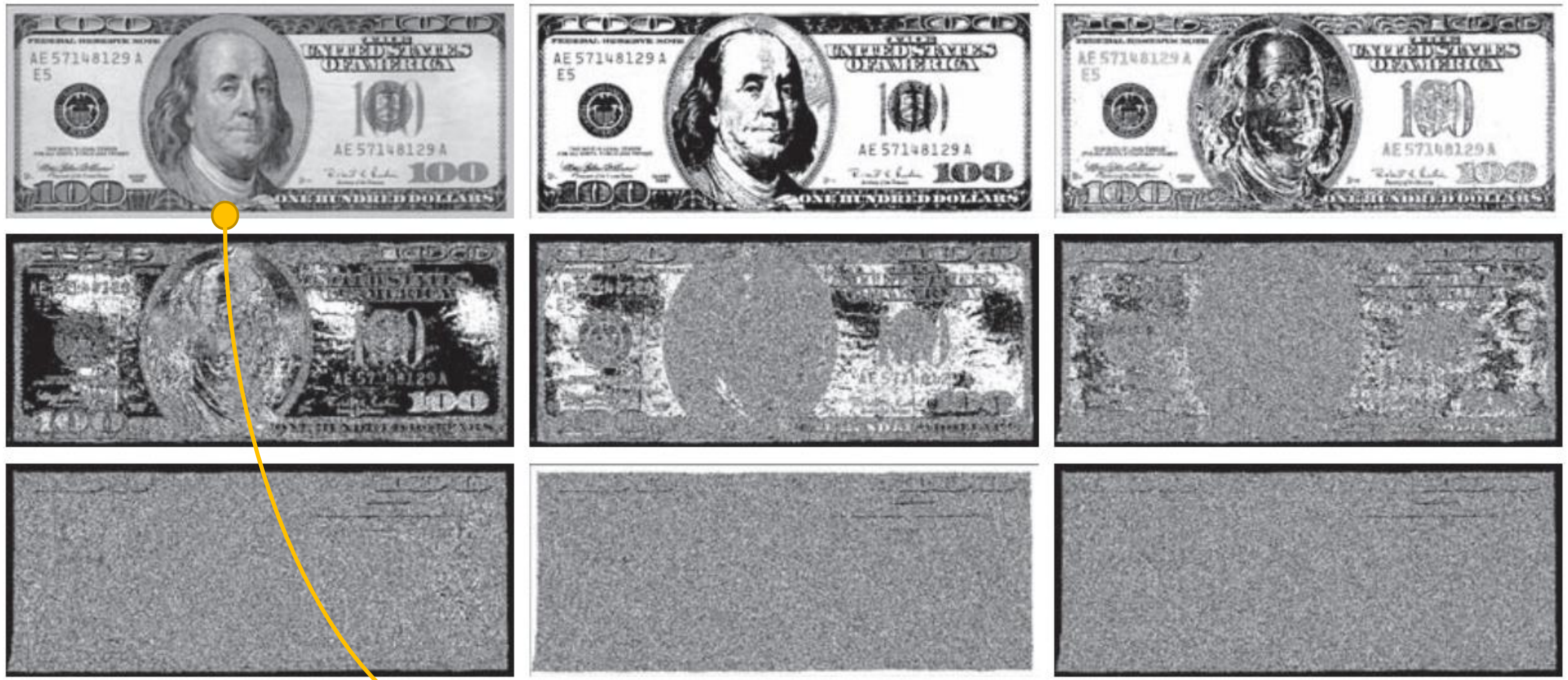




(a) Aortic angiogram. (b) Result of using a slicing transformation of the type illustrated in Fig. 3.11(a), with the range of intensities of interest selected in the upper end of the gray scale. (c) Result of using the transformation in Fig. 3.11(b), with the selected range set near black, so that the grays in the area of the blood vessels and kidneys were preserved.

Bit-Plane Slicing





197 → 1 1 0 0 0 0 1 0

in this example, storing the four highest-order bit planes would allow us to reconstruct the original image in acceptable detail. Storing these four planes instead of the original image requires 50% less storage.

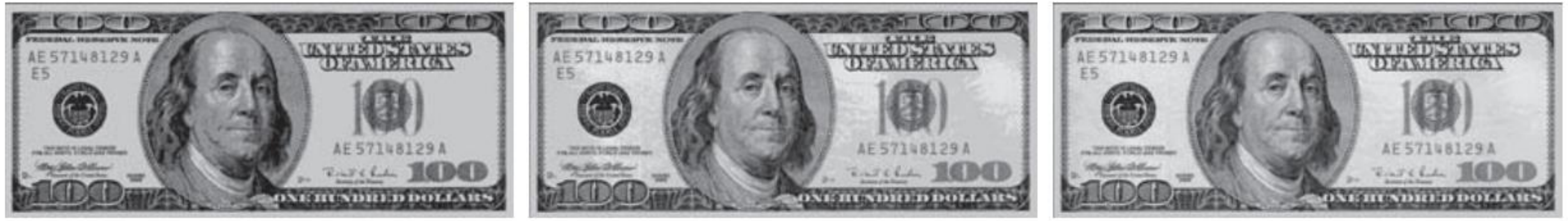
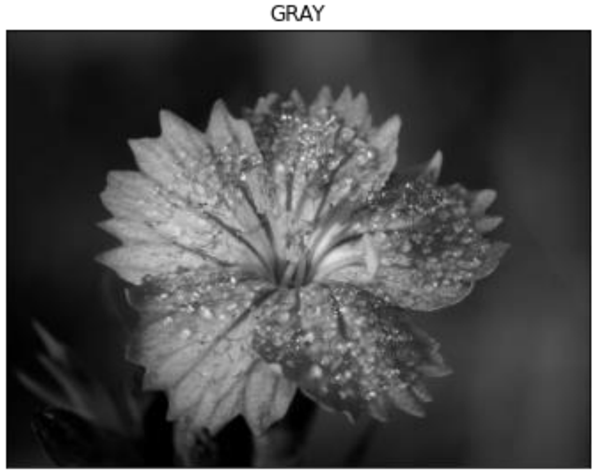


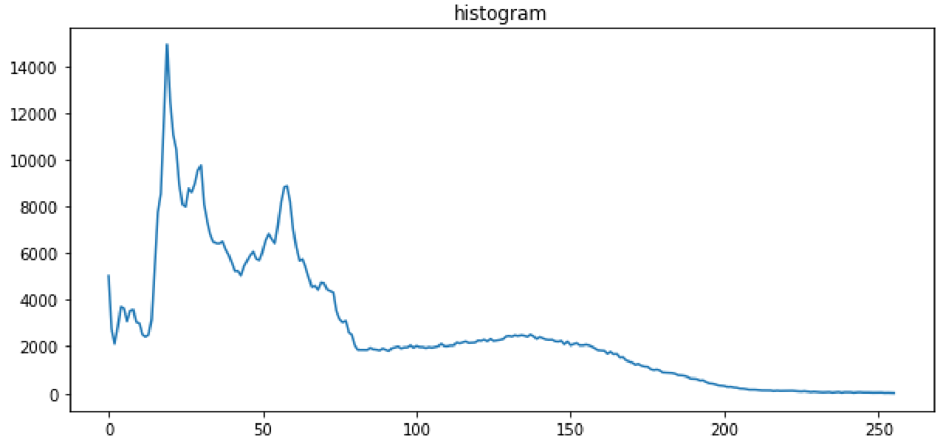
Image reconstructed from bit planes: (a) 8 and 7; (b) 8, 7, and 6; (c) 8, 7, 6, and 5.

✓ ✓ ✓ ✓ ✗ ✗ ✗ ✗

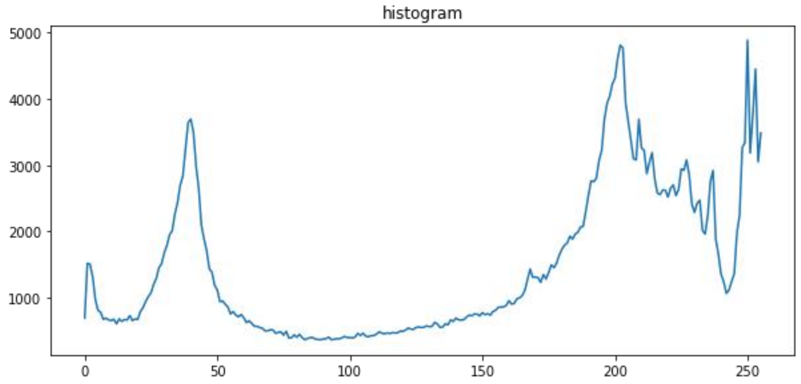
Histogram

the histogram of an image normally refers to a histogram of the pixel intensity values. This histogram is a graph showing the number of pixels in an image at each different intensity value found in that image. For an 8-bit gray scale image there are 256 different possible intensities, and so the histogram will graphically display 256 numbers showing the distribution of pixels amongst those grayscale values.





(720, 960)



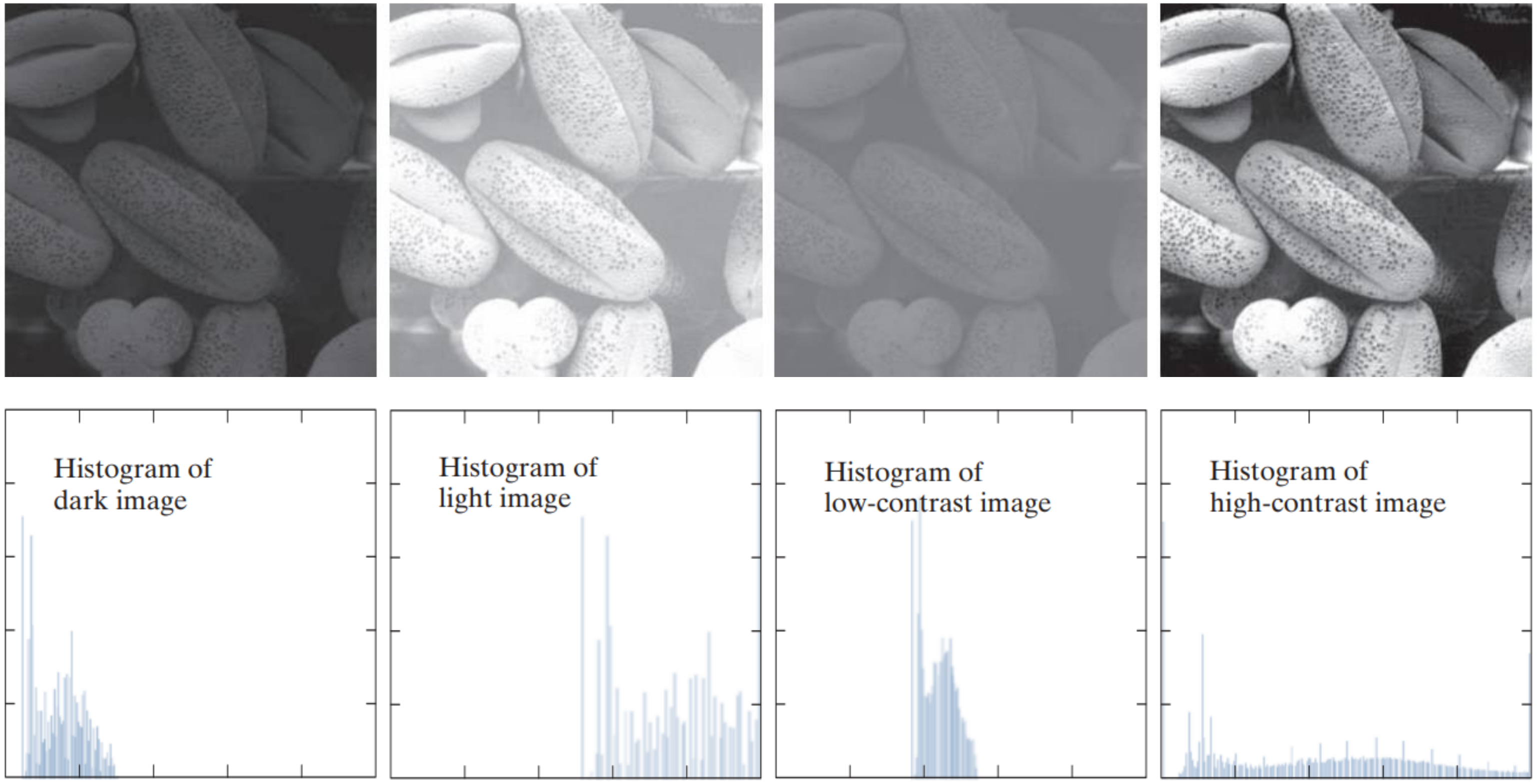
unnormalized histogram

$$h(r_k) = n_k \quad for\ k = 0,1,2,\dots,(L-1)$$

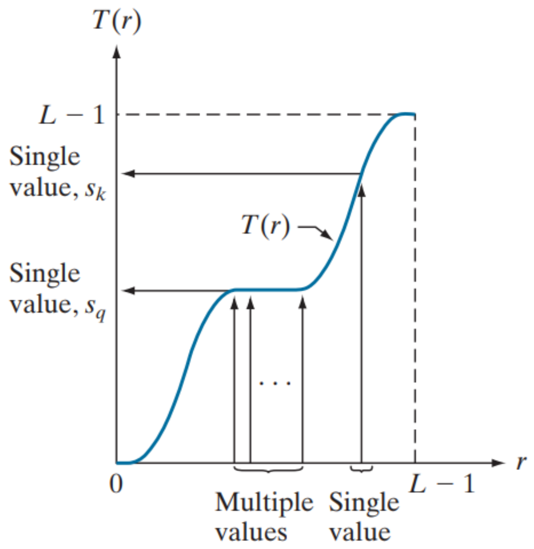
normalized histogram

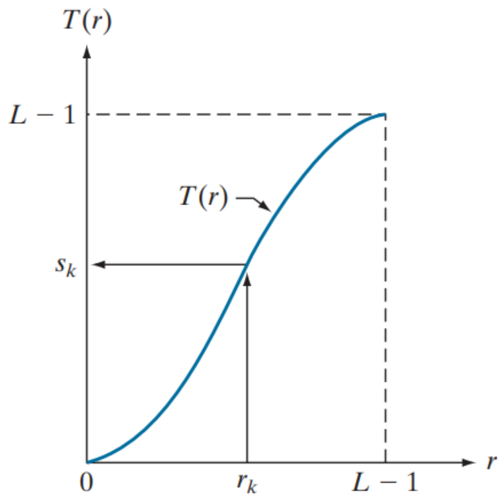
$$h(r_k) = \frac{n_k}{MN} \quad for\ k = 0,1,2,\dots,(L-1)$$

$$\sum_{0}^{L-1} h(r_k) = 1$$


Histogram of dark image
Histogram of light image
Histogram of low-contrast image
Histogram of high-contrast image

incomplete

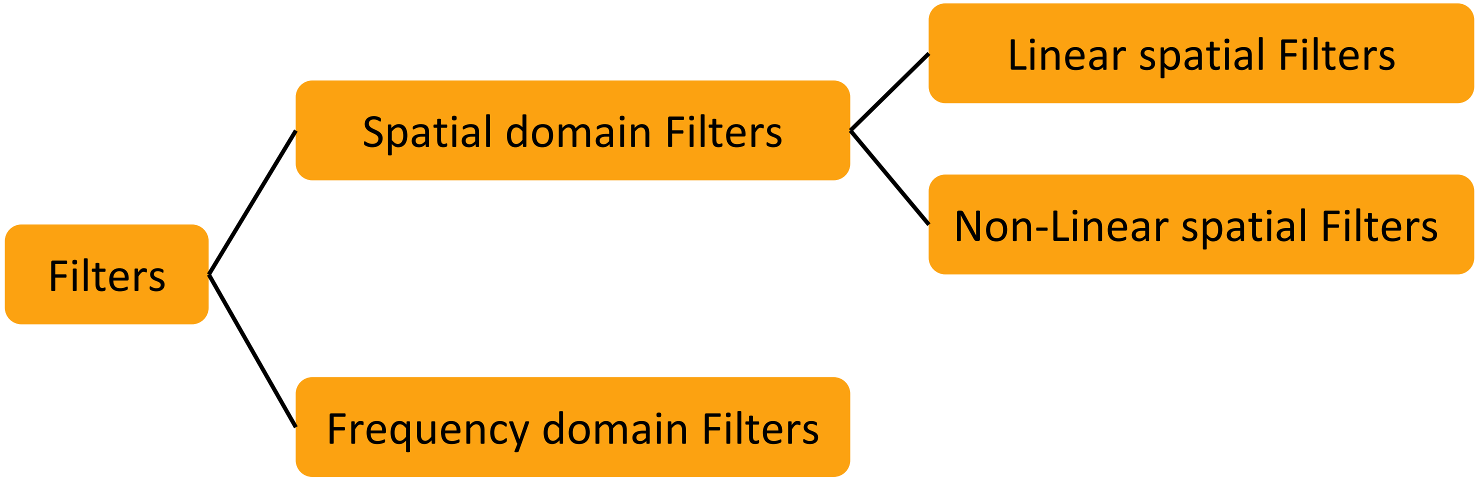




$$s = T(r) \qquad 0 \le r \le L-1$$

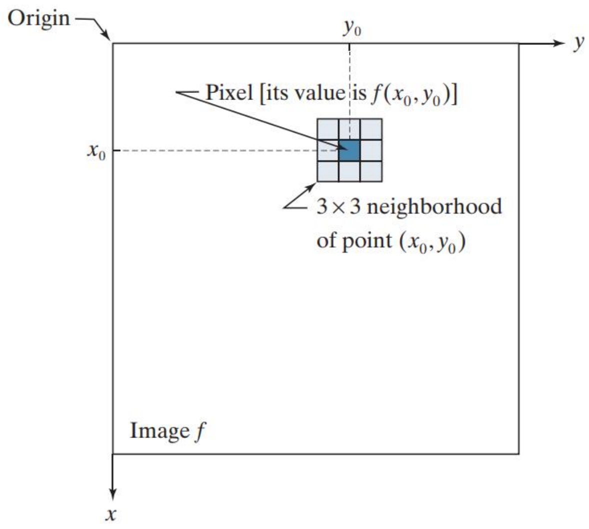
$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

$$s = T(r) = (L-1)\int_0^r p_r(w)\,dw$$


Filters
Spatial domain Filters
Frequency domain Filters
Linear spatial Filters
Non-Linear spatial Filters

Spatial Filtering

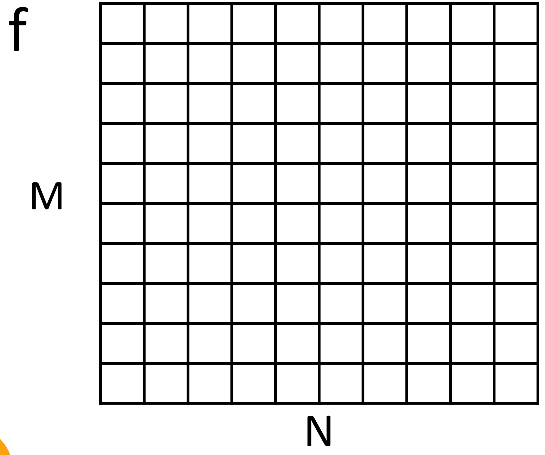
filtering examples in this section deal mostly with image enhancement.
The name filter is borrowed from frequency domain processing where "filtering" refers to passing, modifying, or rejecting specified frequency components of an image.
Spatial filtering modifies an image by replacing the value of each pixel by a function of the values of the pixel and its neighbours. If the operation performed on the image pixels is linear, then the filter is called a linear spatial filter. Otherwise, the filter is a nonlinear spatial filter

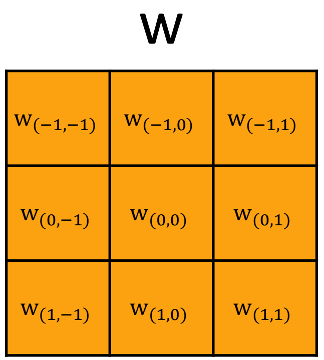

Origin
y_0
y
Pixel [its value is $f(x_0, y_0)$]
x_0
3×3 neighborhood of point (x_0, y_0)
Image f
x

Spatial correlation

A linear spatial filter performs a sum-of-products operation between an image f and a filter kernel, w. The kernel is an array whose size defines the neighborhood of operation, and whose coefficients determine the nature of the filter.

Other terms used to refer to a spatial filter kernel are mask, template, and window.





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

or

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)f(x+s,y+t)$$

m = 2a+1 and n = 2b+1

Spatial correlation

1	0	1
0	1	0
1	0	1

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

1x1	1x0	1x1	0	0
0x0	1x1	1x0	1	0
0x1	0x0	1x1	1	1
0	0	1	1	0
0	1	1	0	0

4		

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b} w(s,t)f(x+s,y+t)$$

correlation

Consists of moving the center of a kernel over an image, and computing the sum of products at each location.

convolution

Same as correlation, except that the correlation kernel is rotated by 180°

correlation

1	2	3
4	5	6
7	8	9

Rotate 180° →

convolution

9	8	7
6	5	4
3	2	1

1-D example

Correlation

w

1 2 **4** 2 8

Convolution

w

8 2 **4** 2 1

Zero padding

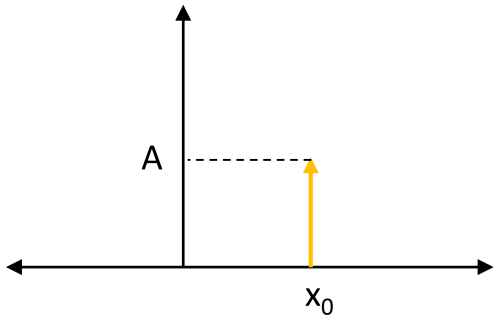
```
0 0 0 0 0 1 0 0 0 0 0 0
| | | | |
1 2 4 2 8

    0 8 2 4 2 1 0 0
```

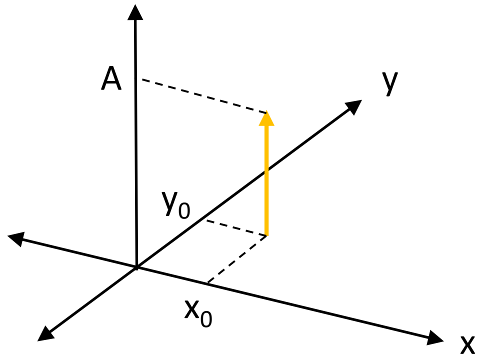
```
0 0 0 0 0 1 0 0 0 0 0 0
| | | | |
8 2 4 2 1

    0 1 2 4 2 8 0 0
```

correlating a kernel w with a function that contains all 0's and a single 1 (discrete unit impulse) yields a copy of w, but rotated by 180°

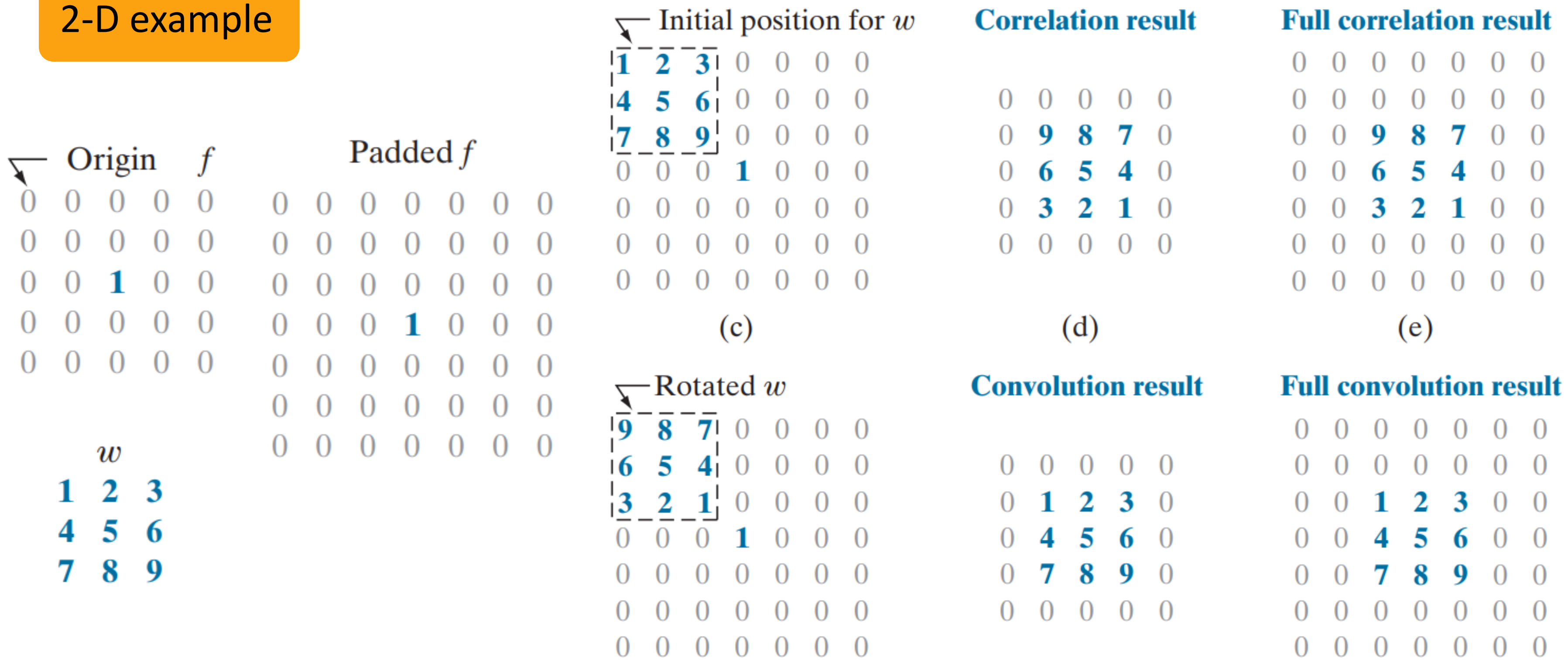
Discrete unit impulse

$$\delta(x - x_0, y - y_0) = \begin{cases} A & \text{if } x = x_0 \text{ and } y = y_0 \\ 0 & \text{otherwise} \end{cases}$$





2-D example



Origin f

$$\begin{matrix} 0&0&0&0&0\\ 0&0&0&0&0\\ 0&0&1&0&0\\ 0&0&0&0&0\\ 0&0&0&0&0 \end{matrix}$$

w

$$\begin{matrix} 1&2&3\\ 4&5&6\\ 7&8&9 \end{matrix}$$

Padded f

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Initial position for w

$$\begin{matrix} 1&2&3&0&0&0&0\\ 4&5&6&0&0&0&0\\ 7&8&9&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(c)

Correlation result

$$\begin{matrix} 0&0&0&0&0\\ 0&9&8&7&0\\ 0&6&5&4&0\\ 0&3&2&1&0\\ 0&0&0&0&0 \end{matrix}$$

(d)

Full correlation result

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&9&8&7&0&0\\ 0&0&6&5&4&0&0\\ 0&0&3&2&1&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

(e)

Rotated w

$$\begin{matrix} 9&8&7&0&0&0&0\\ 6&5&4&0&0&0&0\\ 3&2&1&0&0&0&0\\ 0&0&0&1&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Convolution result

$$\begin{matrix} 0&0&0&0&0\\ 0&1&2&3&0\\ 0&4&5&6&0\\ 0&7&8&9&0\\ 0&0&0&0&0 \end{matrix}$$

Full convolution result

$$\begin{matrix} 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0\\ 0&0&1&2&3&0&0\\ 0&0&4&5&6&0&0\\ 0&0&7&8&9&0&0\\ 0&0&0&0&0&0&0\\ 0&0&0&0&0&0&0 \end{matrix}$$

Correlation

$$(w \star f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Convolution

$$(w \bigstar f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Property	Convolution	Correlation
Commutative	$f \star g = g \star f$	—
Associative	$f \star (g \star h) = (f \star g) \star h$	—
Distributive	$f \star (g + h) = (f \star g) + (f \star h)$	$f \,☆\, (g + h) = (f \,☆\, g) + (f \,☆\, h)$

So from this time we use convolution to filter the image in spatial domain

Separable kernels

$$\mathbf{c} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$w = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = c \,.\, r^T$$

$$\mathbf{r} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

It turns out that the product of a column vector and a row vector is the same as the 2-D convolution of the vectors image of size M × N and a kernel of size m × n $\Rightarrow$ M × N × m × n multiplications and additions.

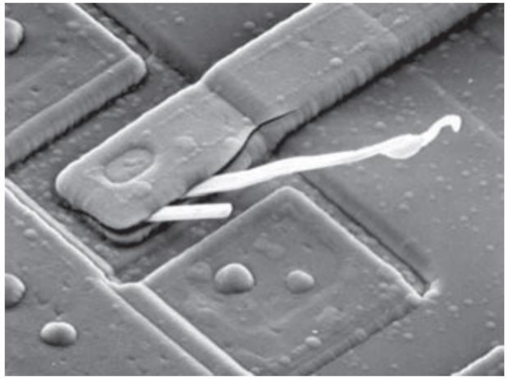
$$w \star f = (w_1 \star w_2) \star f = (w_2 \star w_1) \star f = w_2 \star (w_1 \star f) = (w_1 \star f) \star w_2$$

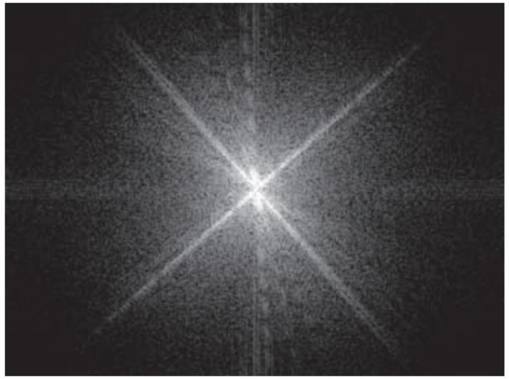
image of size M × N and two kernels of size m and n $\Rightarrow$ M × N × (m + n) multiplications and additions.

$$C = \frac{MNmn}{MN(m+n)} = \frac{mn}{m+n}$$

Spatial vs Frequency domain

An image in spatial domain is the intensity level of each pixel
An image in frequency domain is the frequency intensity level of pixels
The tie between spatial- and frequency-domain processing is the Fourier transform. We use the Fourier transform to go from the spatial spatial domain to frequency domain to return to the spatial domain we use
the inverse Fourier transform



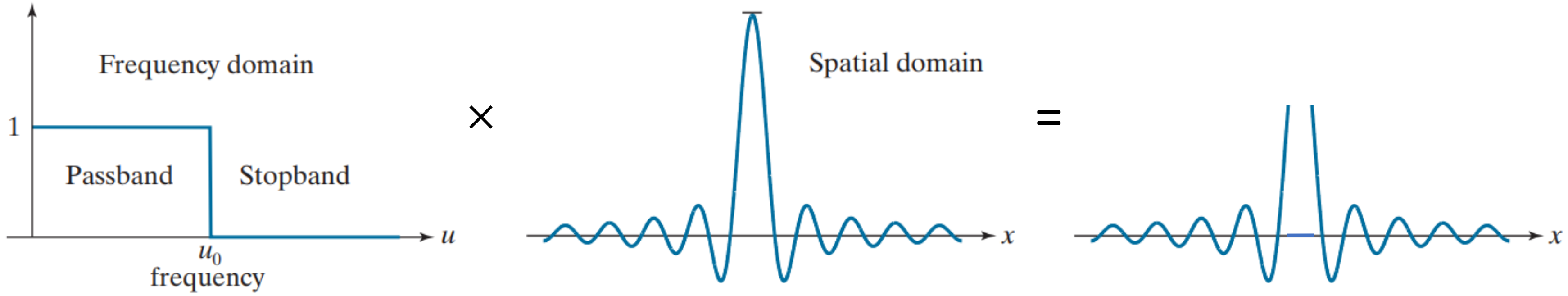


A function (e.g., an image) satisfying some mild conditions can be expressed as the sum of sinusoids of different frequencies and amplitudes. Thus, the appearance of an image depends on the frequencies of its sinusoidal components—change the frequencies of those components, and you will change the appearance of the image.
For example, regions of an image with intensities that vary slowly (e.g., the walls in an image of a room) are characterized by sinusoids of low frequencies. Similarly, edges and other sharp intensity transitions are characterized by high frequencies. Thus, reducing the high frequency components of an image will tend to blur it.

1. Convolution, which is the basis for filtering in the spatial domain, is equivalent to multiplication in the frequency domain, and vice versa.
2. An impulse of strength *A* in the spatial domain is a constant of value *A* in the frequency domain, and vice versa.

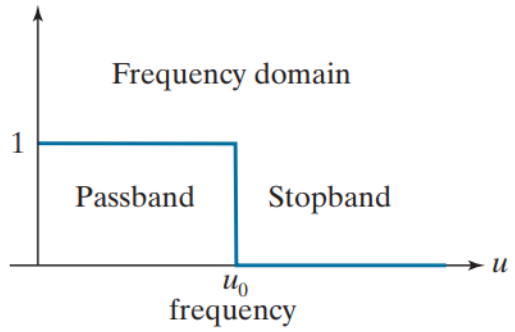
Low pass filter in frequency domain

It eliminates all frequencies above u_0 , while passing all frequencies below this value. This cause elimination of high frequencies and make the image blurred.

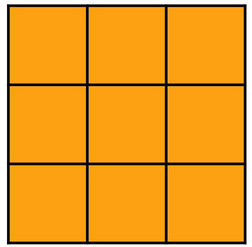


Low pass filter in spatial domain

Because of the duality between the spatial and frequency domains, we can obtain the same result in the spatial domain by convolving the equivalent spatial domain filter kernel with the input spatial function.



Inverse furrier transform



One approach is based on formulating filters based on mathematical properties. For example, a filter that computes the average of pixels in a neighbourhood blurs an image.

A second approach is based on sampling a 2-D spatial function whose shape has a desired property. For example, we will show in the next section that samples from a Gaussian function can be used to construct a weighted-average (lowpass) filter.

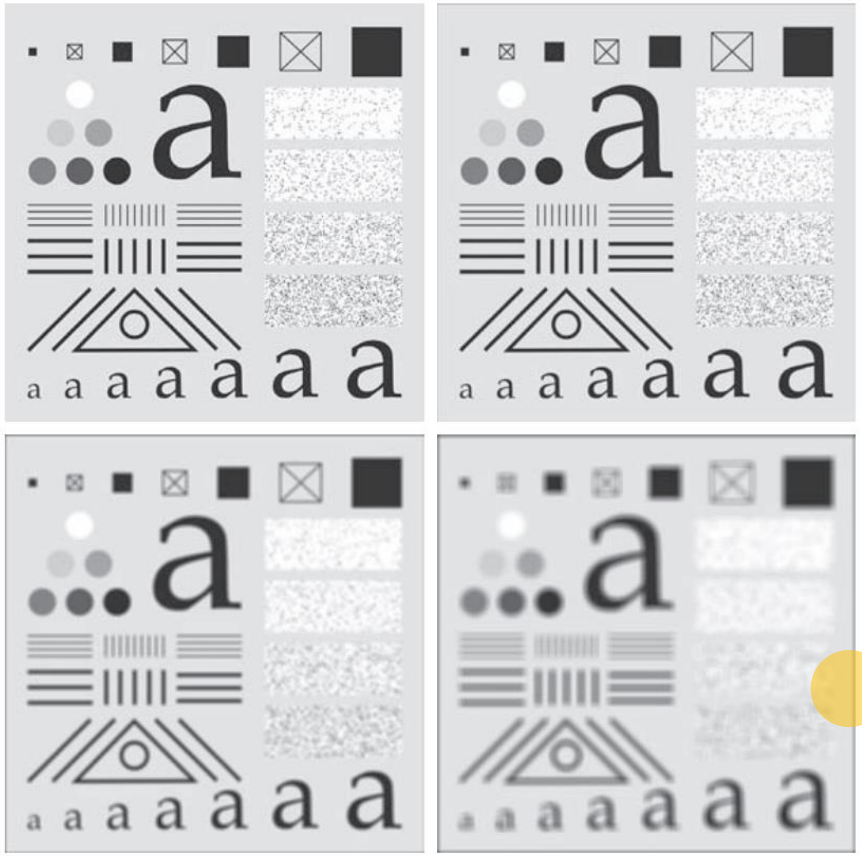
A third approach is to design a spatial filter with a specified frequency response.

Box Filter kernel

The simplest, separable lowpass filter kernel is the box kernel, whose coefficients have the same value (typically 1)

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \times \frac{1}{9} = \begin{bmatrix} 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \\ 0.11 & 0.11 & 0.11 \end{bmatrix}$$

First, the average value of an area of constant intensity would equal that intensity in the filtered image, as it should. Second, normalizing the kernel in this way prevents introducing a bias during filtering



21 × 21

3 × 3

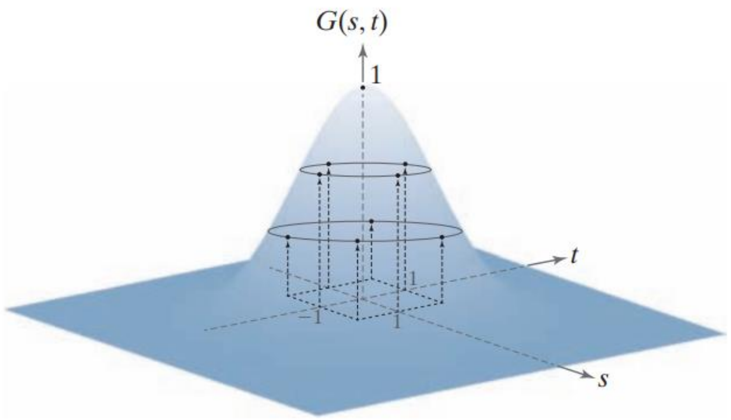
11 × 11

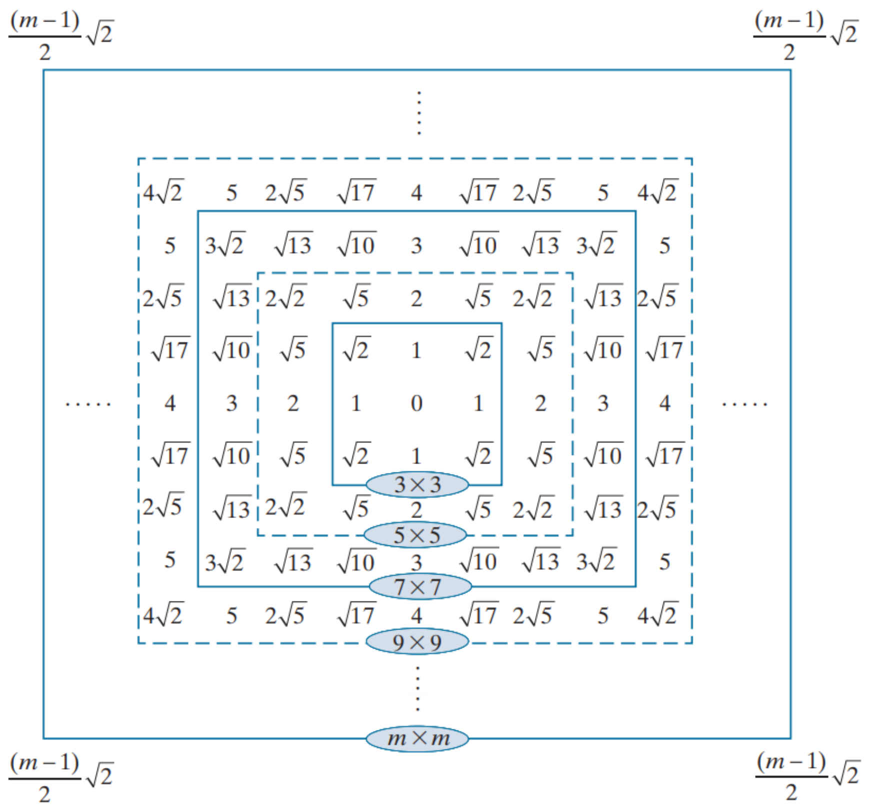
?

Gaussian kernel

In applications involving images with a high level of detail, or with strong geometrical components, the directionality of box filters often produces undesirable results

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



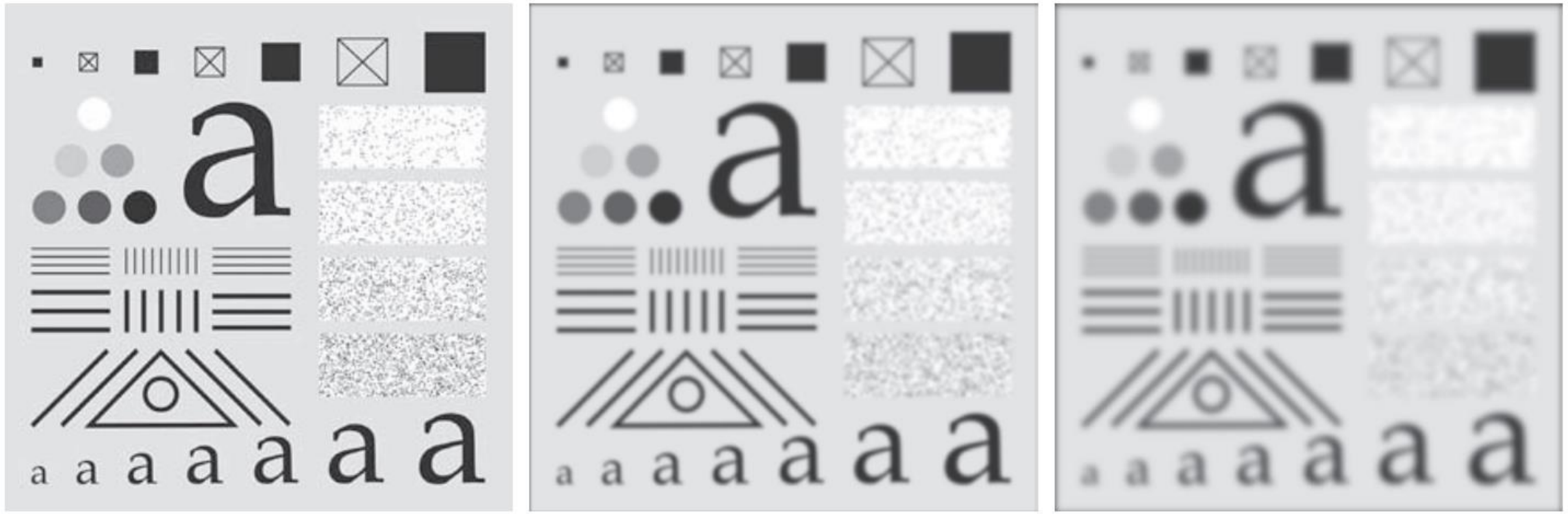

$\frac{(m-1)}{2}\sqrt{2}$
$\frac{(m-1)}{2}\sqrt{2}$
$4\sqrt{2}$ 5 $2\sqrt{5}$ $\sqrt{17}$ 4 $\sqrt{17}$ $2\sqrt{5}$ 5 $4\sqrt{2}$
5 $3\sqrt{2}$ $\sqrt{13}$ $\sqrt{10}$ 3 $\sqrt{10}$ $\sqrt{13}$ $3\sqrt{2}$ 5
$2\sqrt{5}$ $\sqrt{13}$ $2\sqrt{2}$ $\sqrt{5}$ 2 $\sqrt{5}$ $2\sqrt{2}$ $\sqrt{13}$ $2\sqrt{5}$
$\sqrt{17}$ $\sqrt{10}$ $\sqrt{5}$ $\sqrt{2}$ 1 $\sqrt{2}$ $\sqrt{5}$ $\sqrt{10}$ $\sqrt{17}$
4 3 2 1 0 1 2 3 4
$\sqrt{17}$ $\sqrt{10}$ $\sqrt{5}$ $\sqrt{2}$ 1 $\sqrt{2}$ $\sqrt{5}$ $\sqrt{10}$ $\sqrt{17}$
$2\sqrt{5}$ $\sqrt{13}$ $2\sqrt{2}$ $\sqrt{5}$ 2 $\sqrt{5}$ $2\sqrt{2}$ $\sqrt{13}$ $2\sqrt{5}$
5 $3\sqrt{2}$ $\sqrt{13}$ $\sqrt{10}$ 3 $\sqrt{10}$ $\sqrt{13}$ $3\sqrt{2}$ 5
$4\sqrt{2}$ 5 $2\sqrt{5}$ $\sqrt{17}$ 4 $\sqrt{17}$ $2\sqrt{5}$ 5 $4\sqrt{2}$
3×3
5×5
7×7
9×9
$m\times m$
$\frac{(m-1)}{2}\sqrt{2}$
$\frac{(m-1)}{2}\sqrt{2}$

$$\begin{bmatrix} \sqrt{2} & 1 & \sqrt{2} \\ 1 & 0 & 1 \\ \sqrt{2} & 1 & \sqrt{2} \end{bmatrix}_{3\times 3}$$

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

For K=1 and σ=1

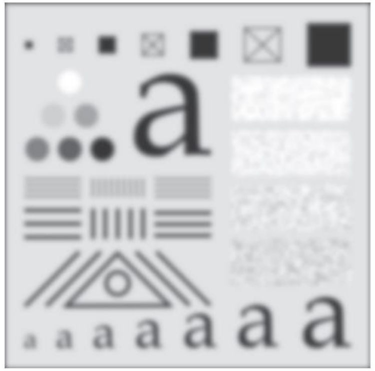
$$\longrightarrow \begin{bmatrix} 0.3679 & 0.6065 & 0.3679 \\ 0.6065 & 1 & 0.6065 \\ 0.3679 & 0.6065 & 0.3679 \end{bmatrix} \times \frac{1}{4.8976}$$

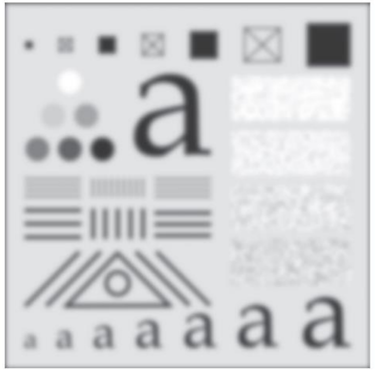


21*21 with K=1 and σ=3.5

43*43 with K=1 and σ=7

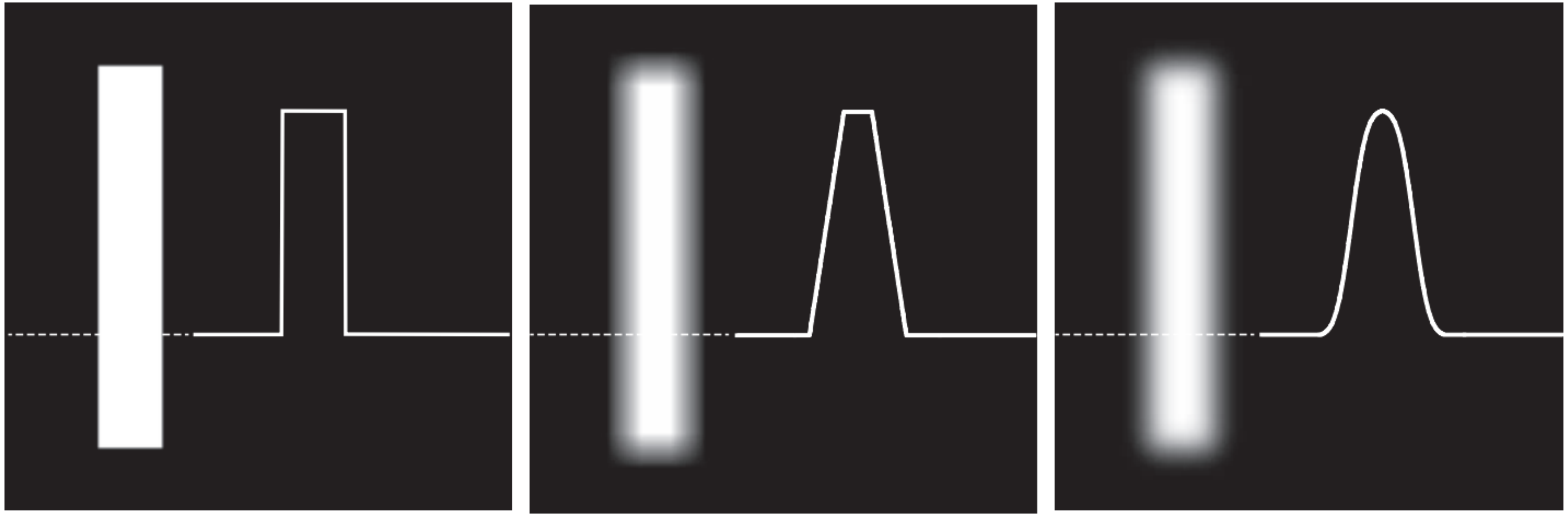
we know that the values of a Gaussian function at a distance larger than 3σ from the mean are small enough that they can be ignored. This means that if we select the size of a Gaussian kernel to be $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$ (the notation $\lceil c \rceil$ is used to denote the ceiling of c; that is, the smallest integer not less than c), we are assured of getting essentially the same result as if we had used an arbitrarily large Gaussian kernel. Viewed another way, this property tells us that there is nothing to be gained by using a Gaussian kernel larger than $\lceil 6\sigma \rceil \times \lceil 6\sigma \rceil$ for image processing. Because typically we work with kernels of odd dimensions, we would use the smallest odd integer that satisfies this condition (e.g., a 43×43 kernel if $\sigma = 7$).


43×43 with k=1 and $\sigma = 7$


85×85 with k=1 and $\sigma = 7$

difference

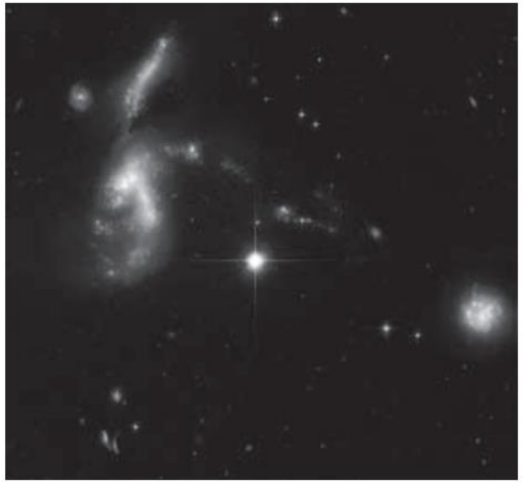
Box Filter vs Gaussian Filter

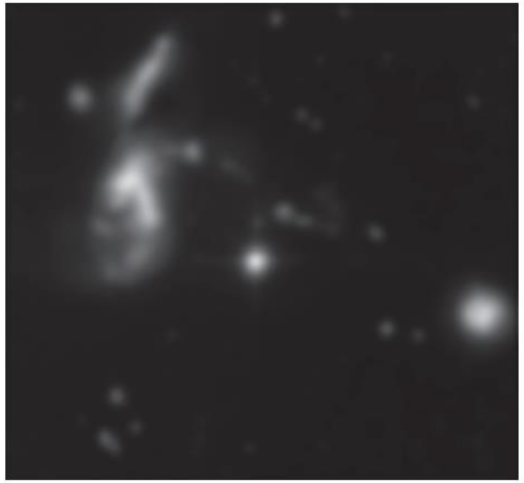


Box Filter

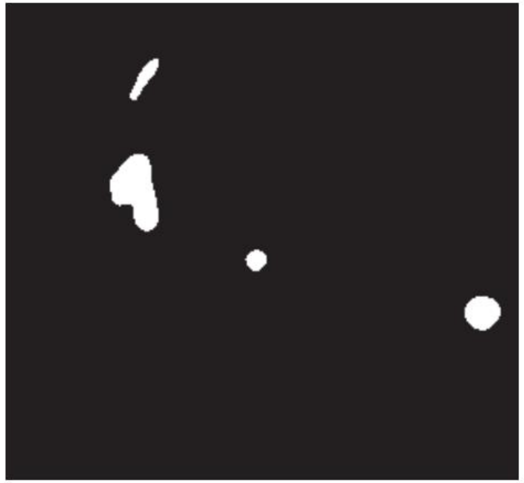
Gaussian Filter

Using lowpass filtering and thresholding for region extraction





the result of filtering the original image with a Gaussian kernel of size 151 × 151 (approximately 6% of the image width) and standard deviation = 25



result of thresholding the filtered image with a threshold T = 0.4

Shading correction (flat-field correction) using lowpass filtering

Consider the 2048 × 2048 checkerboard image in Fig. 3.42(a), whose inner squares are of size 128 × 128 pixels. Figure 3.42(b) is the result of lowpass filtering the image with a 512 × 512 Gaussian kernel (four times the size of the squares), K = 1, and sd = 128 (equal to the size of the squares). This kernel is just large enough to blur-out the squares (a kernel three times the size of the squares is too small to blur them out sufficiently)

is the result of dividing (a) by (b)

Important note

Thinking of computation time, if it took **30 sec** to process a set of images using the two 1-D separable components of the Gaussian kernel, it would have taken **2.2 hrs** to achieve the same result using a nonseparable lowpass kernel

Order-static (nonlinear) filters

Order-statistic filters are nonlinear spatial filters whose response is based on ordering (ranking) the pixels contained in the region encompassed by the filter. Smoothing is achieved by replacing the value of the center pixel with the value determined by the ranking result. The best-known filter in this category is the median filter, which, as its name implies, replaces the value of the center pixel by the median of the intensity values in the neighborhood of that pixel (the value of the center pixel is included in computing the median).

Median filtering

$1\ 6\ 1\ 7\ 8\ 5\ 6\ 4\ 2 \xrightarrow{\text{sort}} 1\ 1\ 2\ 4\ \boxed{5}\ 6\ 6\ 7\ 8$

$$\begin{matrix} 1 & 6 & 1 \\ 2 & 8 & 5 \\ 3 & 4 & 2 \end{matrix} \xrightarrow{\text{sort}} \begin{matrix} 1 & 1 & 2 \\ 2 & \boxed{3} & 4 \\ 5 & 6 & 8 \end{matrix}$$

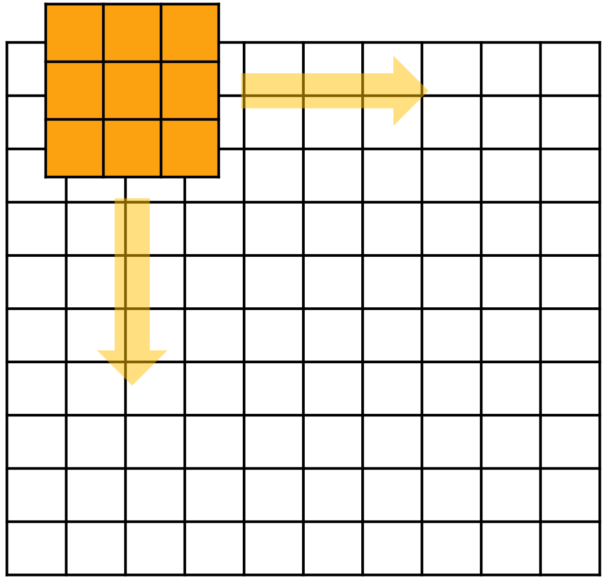
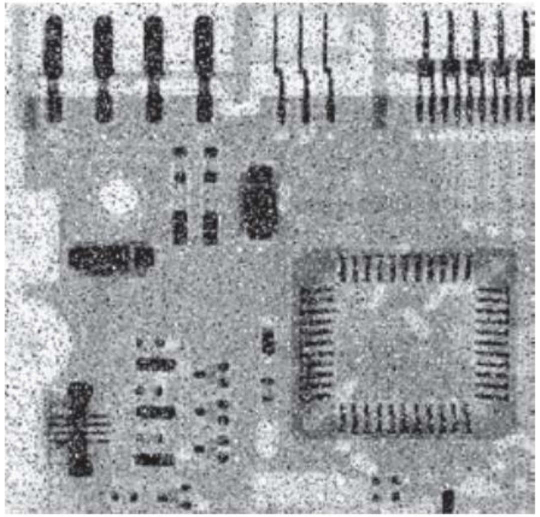
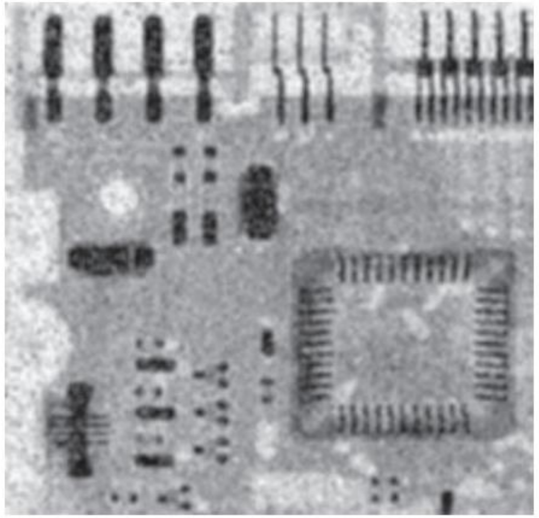


Image with S&P noise

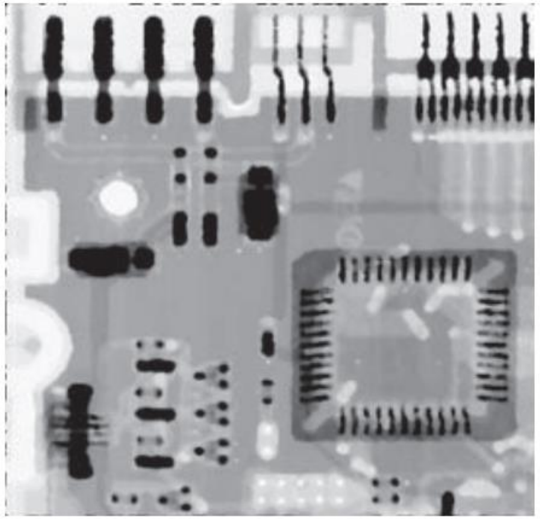
With Gaussian filter

With median filter





19 × 19 Gaussian lowpass filter kernel with sd = 3



7 × 7 median filter

Derivation of image

$$f(x+\Delta x) = f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{(\Delta x)^3}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \cdots\cdots$$

where Δx is the separation between samples of f. For our purposes, this separation is measured in pixel units. Thus, following the convention is Δx = 1

$$f(x+1) = f(x) + \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{1}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \cdots\cdots$$

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

$$f(x+\Delta x) = f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{(\Delta x)^3}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \cdots\cdots$$

$\Delta x = 1$

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

Forward difference

$\Delta x = -1$

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

backward difference

$$\frac{\partial f(x)}{\partial x} = f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

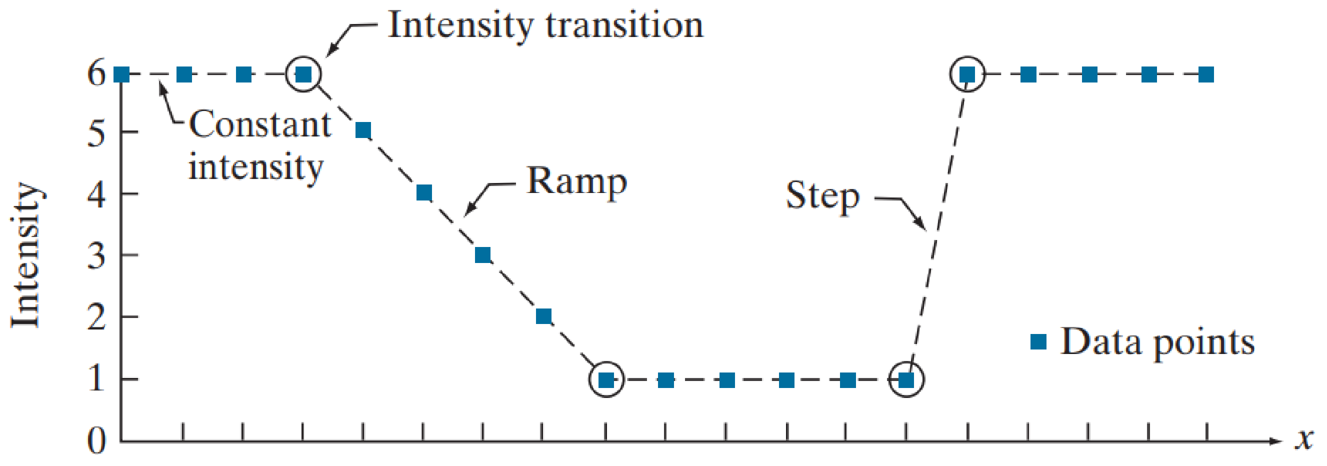
central difference

Second Derivation

$$f(x+1) = f(x) + \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2} + \frac{1}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \cdots\cdots$$

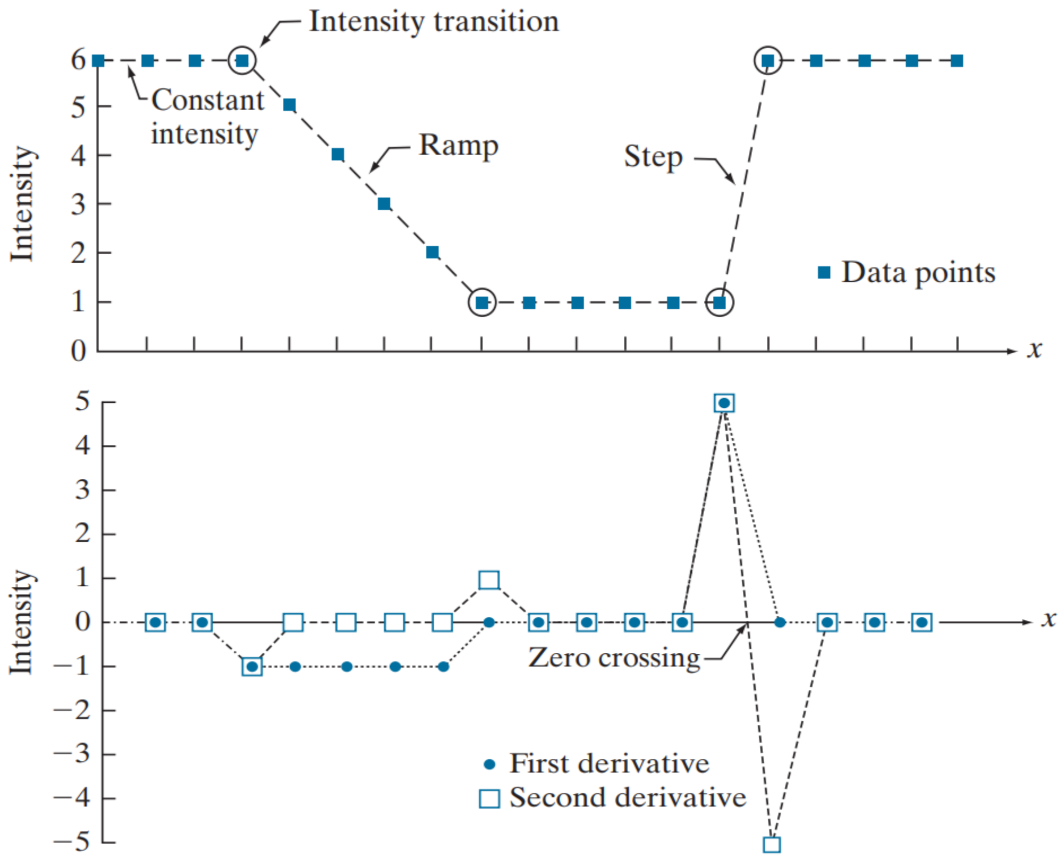
$$f(x-1) = f(x) - \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial x^2} - \frac{1}{3!}\frac{\partial^3 f(x)}{\partial x^3} + \cdots\cdots$$

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



Values of scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6	6	6 → x
1st derivative		0	0	−1	−1	−1	−1	−1	0	0	0	0	0	5	0	0	0	0	
2nd derivative		0	0	−1	0	0	0	0	1	0	0	0	0	5	−5	0	0	0	

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



This zero crossing property is quite useful for locating edges

Derivation of image

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

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

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

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

Second Derivation of image

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

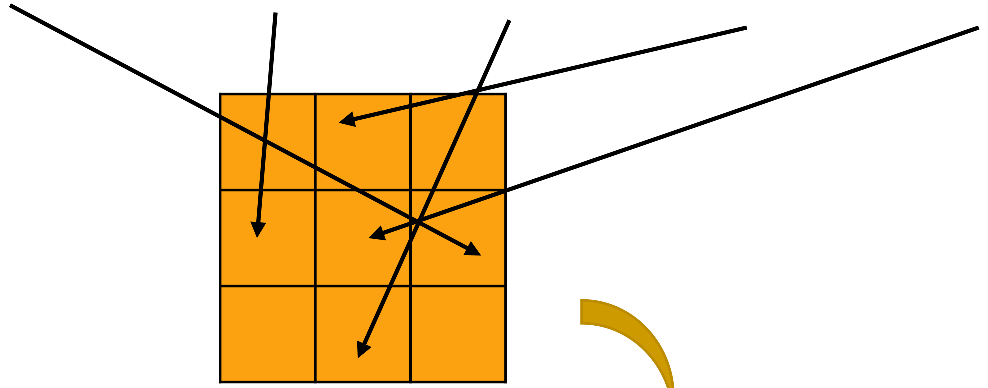
$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) - 2f(x,y) + f(x-1\,,y)$$

$$+$$

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

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

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



0	1	0
1	-4	1
0	1	0

0	1	0
1	−4	1
0	1	0

(1)

1	1	1
1	−8	1
1	1	1

(2)

Kernel used to implement an extension of laplacian

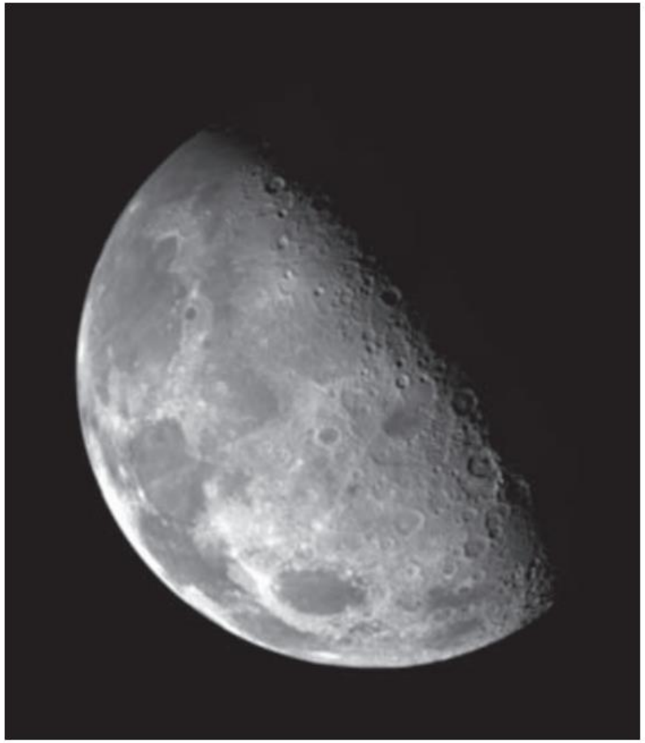
0	−1	0
−1	4	−1
0	−1	0

(3)

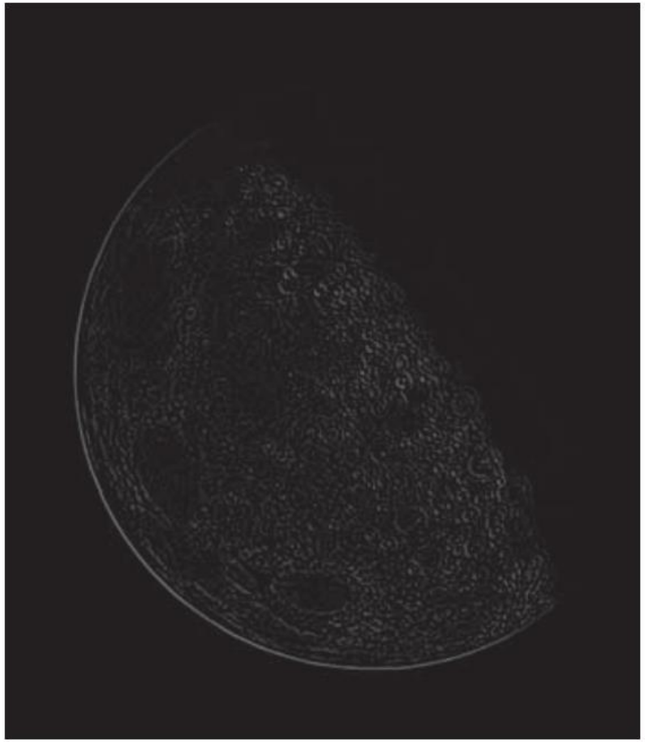
Using Δx or Δy = -1

−1	−1	−1
−1	8	−1
−1	−1	−1

(4)



Original



Laplacian

Sharpening using Laplacian

$$g(x,y) = f(x,y) + c\left[\nabla^2 f(x,y)\right]$$

c = -1

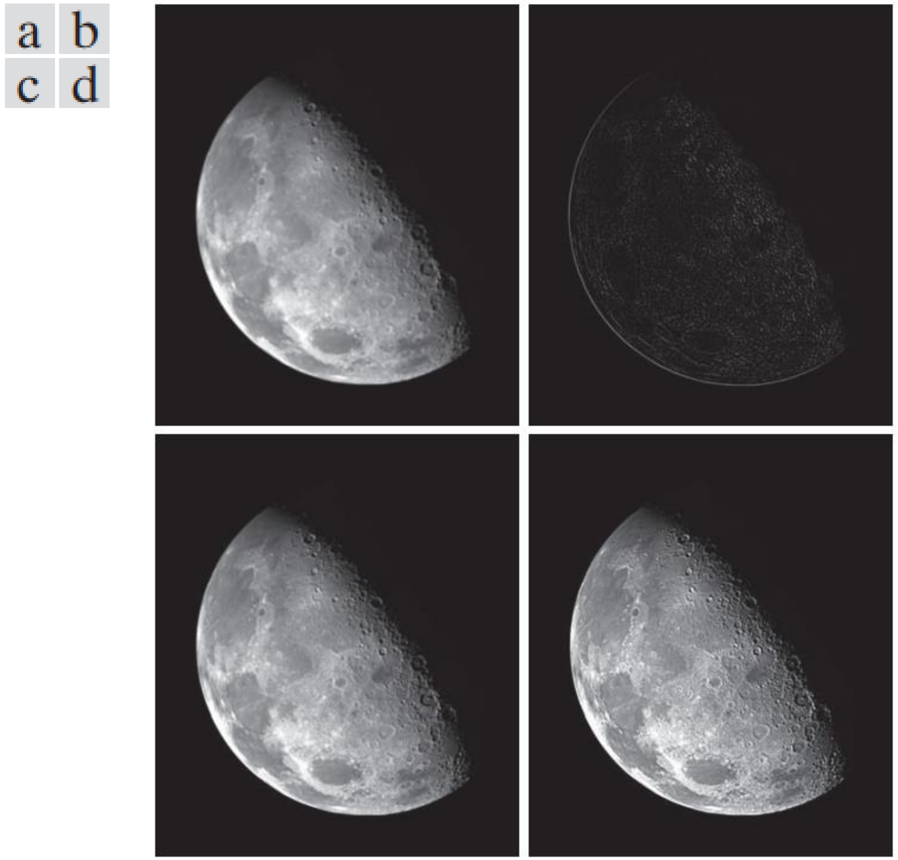
0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

c = 1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

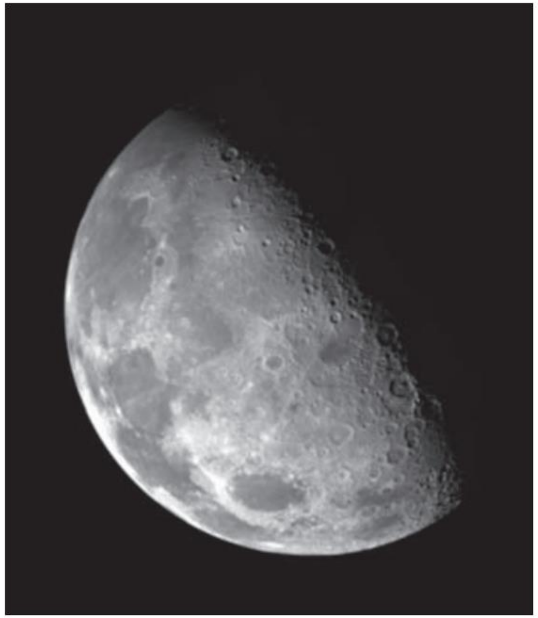


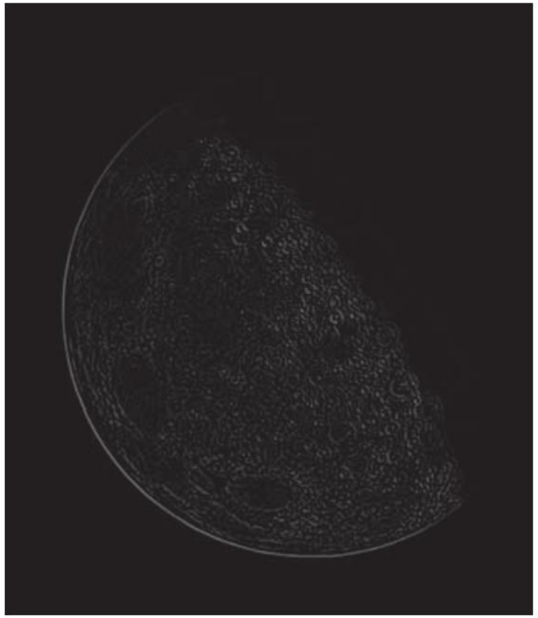
(a) Blurred image of the North Pole of the moon.

(b) Laplacian image obtained using the kernel (1).

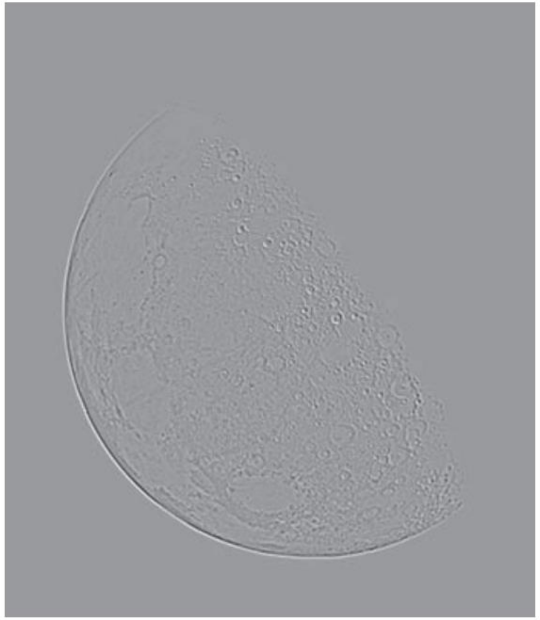
(c) Image sharpened using Laplacian equation with c = −1.

(d) Image sharpened using the same procedure, but with the kernel (2)





Some pixels have negative values



rescaled to the full [0, 255] range of intensity values

Unsharp masking and highboost filtering

Subtracting an unsharp (smoothed) version of an image from the original image is process that has been used since the 1930s by the printing and publishing industry to sharpen images. This process, called unsharp masking, consists of the following steps:

1.Blur the original image.
2. Subtract the blurred image from the original (the resulting difference is called the mask.)
3. Add the mask to the original

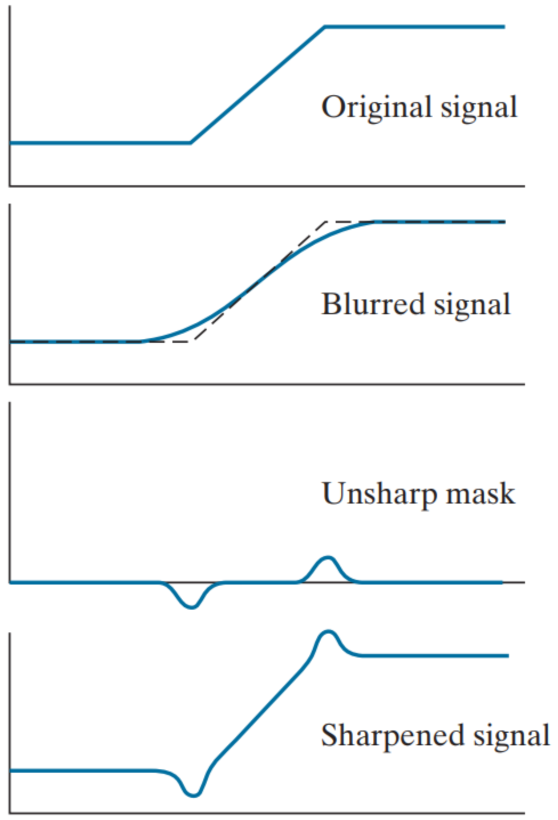
$$g_{\text{mask}}(x, y) = f(x, y) - \bar{f}(x, y)$$

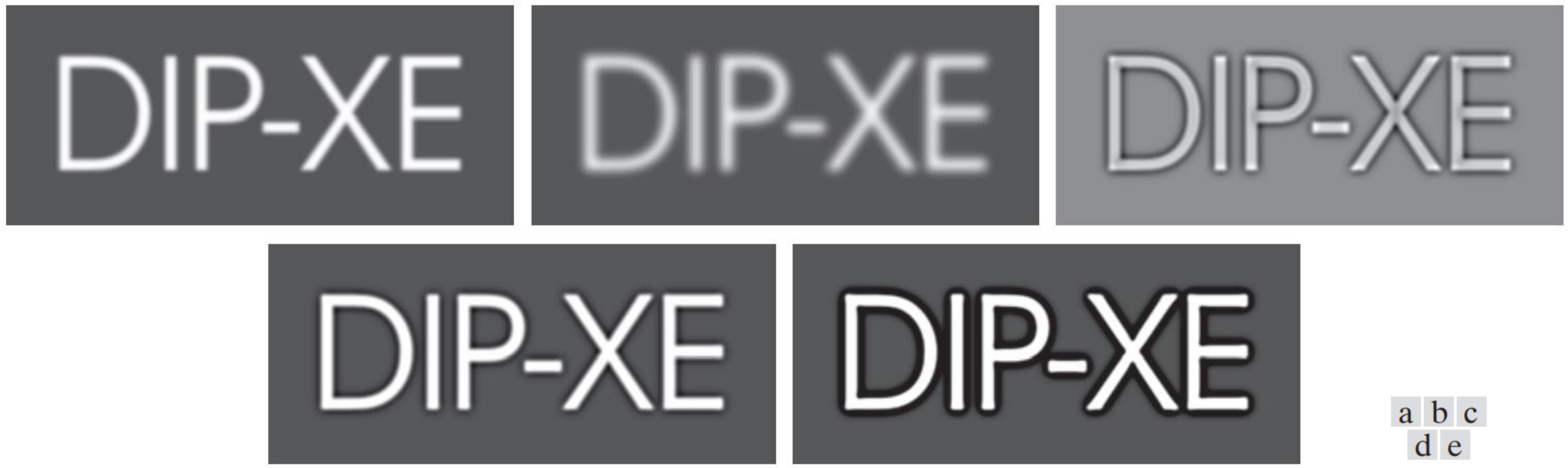
$$g(x, y) = f(x, y) + k\, g_{\text{mask}}(x, y)$$

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

When k = 1 we have unsharp masking, as defined above. When k > 1, the process is referred to as highboost filtering. Choosing k < 1 reduces the contribution of the unsharp mask.

note that the unsharp mask is similar to what we would obtain using a second-order derivative





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

First order derivation of image

$$\nabla f \equiv \operatorname{grad}(f) = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

It is common practice to refer to this image as the gradient image

The magnitude (length) of vector ∇f :

$$M(x,y) = \|\nabla f\| = \operatorname{mag}(\nabla f) = \sqrt{g_x^2 + g_y^2}$$

$$M(x,y) \approx |g_x| + |g_y|$$

Because the components of the gradient vector are derivatives, they are linear operators. However, the magnitude of this vector is not, because of the squaring and square root operations. On the other hand, the partial derivatives are not rotation invariant, but the magnitude of the gradient vector is.

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

−1
1

−1	1

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

Roberts [1965] are one of the earliest attempts to use 2-D kernels with a diagonal preference

$$g_x = \frac{\partial f}{\partial x} = (z_9 - z_5)$$

$$g_y = \frac{\partial f}{\partial y} = (z_8 - z_6)$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	0
0	1

0	−1
1	0

prewitt kernels

$$g_x = \frac{\partial f}{\partial x} = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

sobel kernels

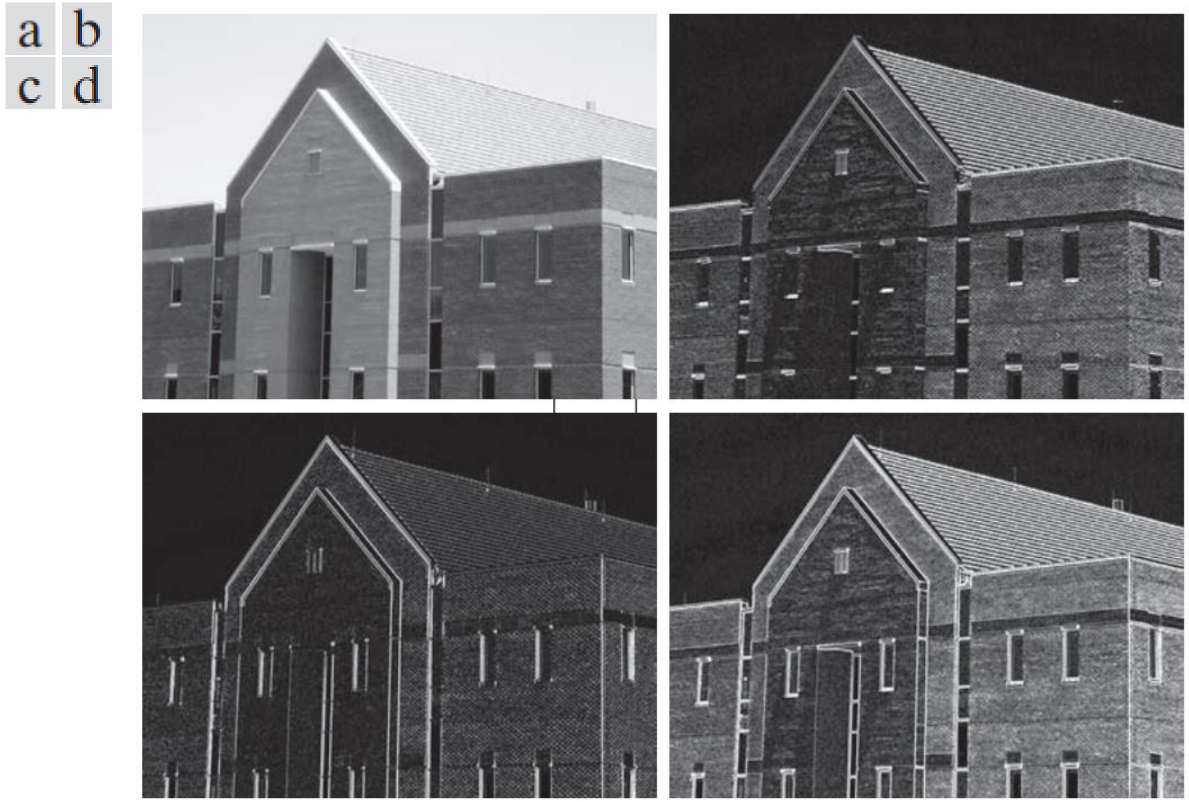
$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Images convolved with all of the kernels using for gradients will have negative values in general.



(a) Image of size 834 × 1114 pixels, with intensity values scaled to the range [0, 1]

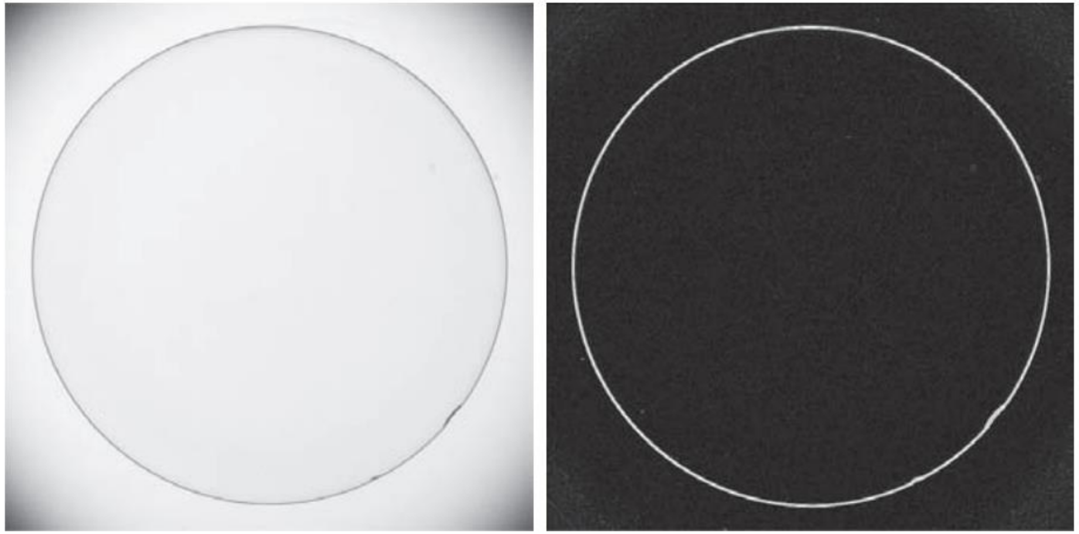
(b) gx , the component of the gradient in the x-direction, obtained using the Sobel kernel to filter the image.

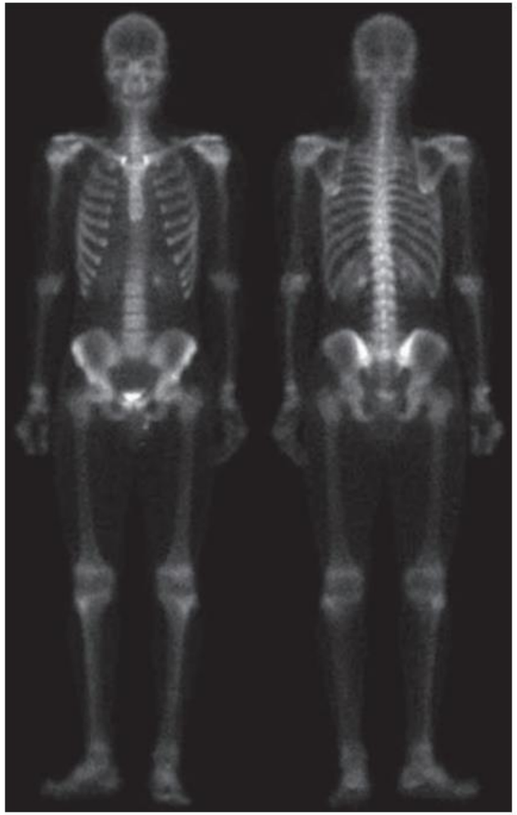
(c) gy obtained using the sobel kernel

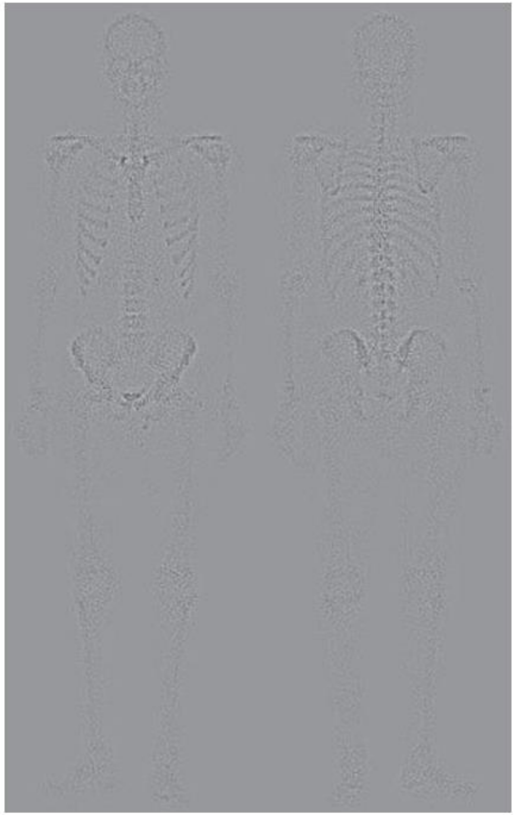
(d) The gradient image, |gx| + |gy|

Using the gradient for edge enhancement

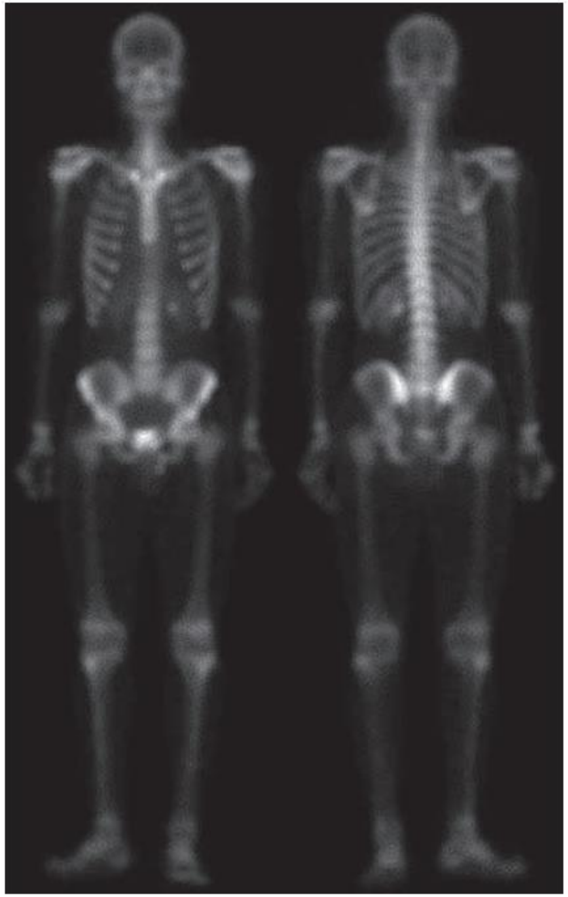
Figure below is an optical image of a contact lens, illuminated by a lighting arrangement designed to highlight imperfections, such as the two edge defects in the lens boundary seen at 4 and 5 o’clock







−1	−1	−1
−1	8	−1
−1	−1	−1

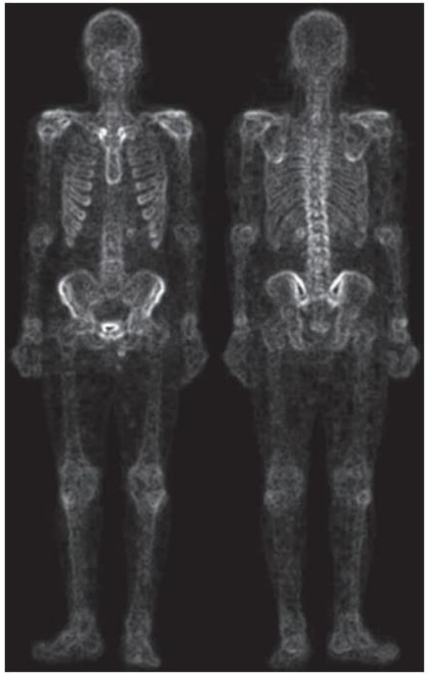


We would expect a rather noisy sharpened image if we added laplacian to original image. This is confirmed by the result in this picture. One way that comes immediately to mind to reduce the noise is to use a median filter. However, median filtering is an aggressive nonlinear process capable of removing image features. This is unacceptable in medical image processing.

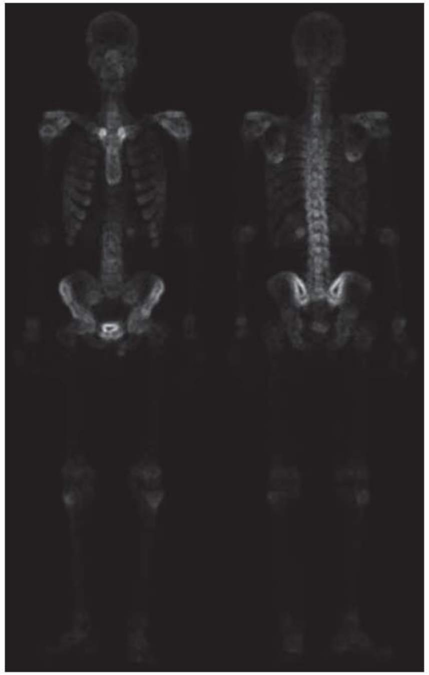
The response of the gradient to noise and fine detail is lower than the Laplacian’s and can be lowered further by smoothing the gradient with a lowpass filter

The idea, then, is to smooth the gradient and multiply it by the Laplacian image. In this context, we may view the smoothed gradient as a mask image. The product will preserve details in the strong areas, while reducing noise in the relatively flat areas. This process can be interpreted roughly as combining the best features of the Laplacian and the gradient. The result is added to the original to obtain a final sharpened image.

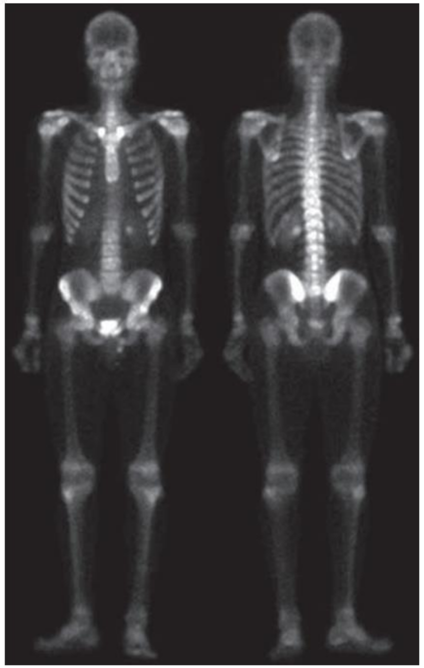
(a)



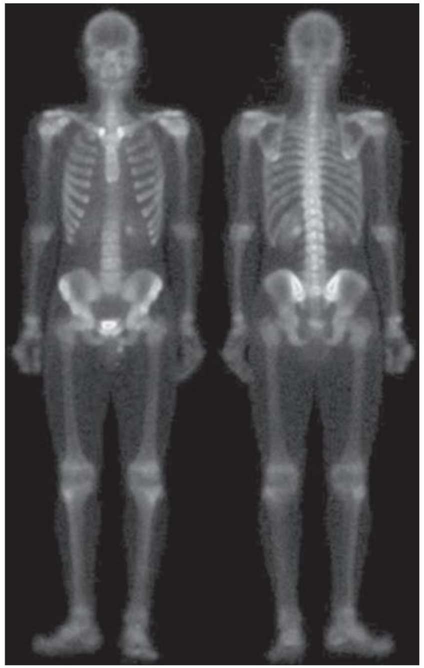
(b)



(c)



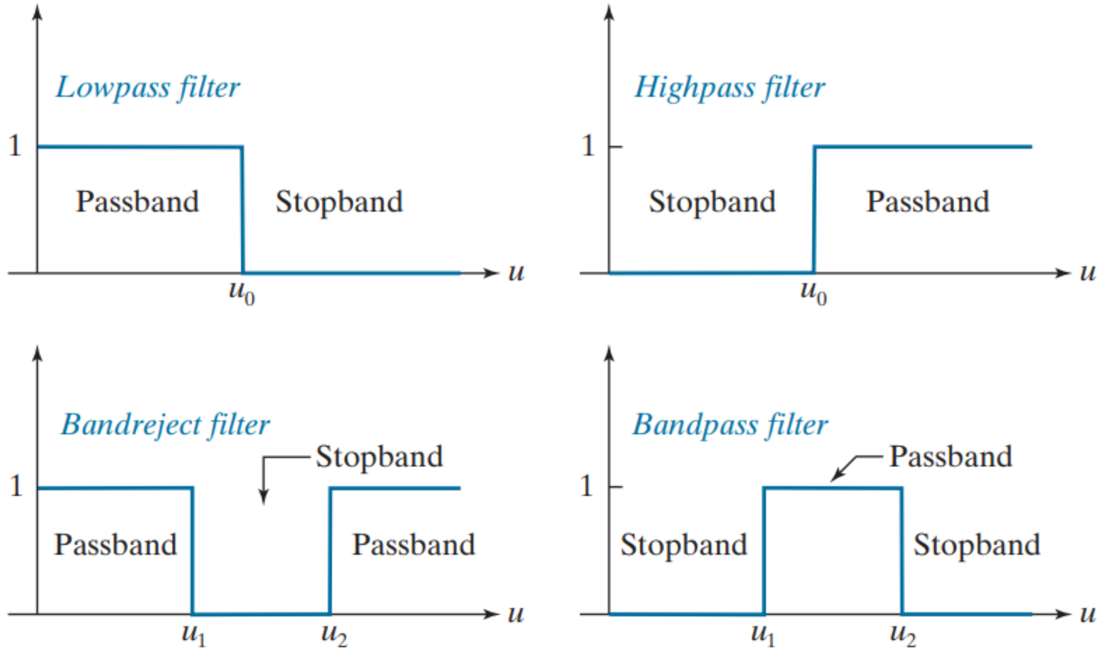
(d)



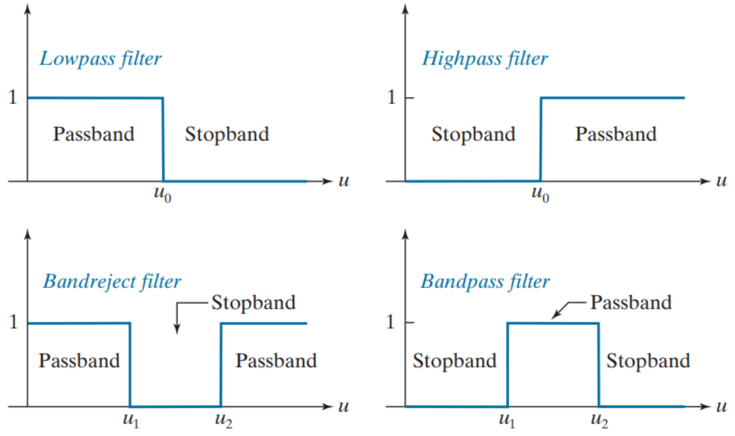
(a) Sobel image smoothed with a 5 × 5 box filter. (b) Mask image formed by the product of (a) and laplacian. (c) Sharpened image obtained by the adding images (a) and (b). (d) Final result obtained by applying a gamma transformation with gamma = 0.5 and c=1

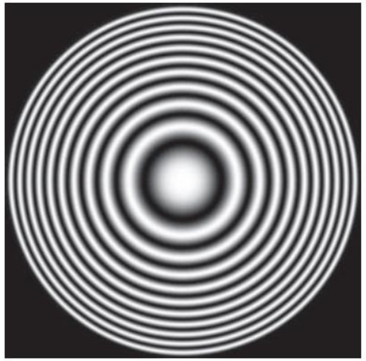
Spatial and frequency-domain linear filters are classified into four broad categories:

- Low pass filter
- high pass filter
- Band pass filter
- Band reject filter

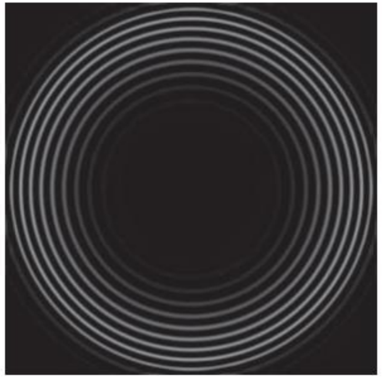


Filter type	Spatial kernel in terms of lowpass kernel, lp
Lowpass	$lp(x,y)$
Highpass	$hp(x,y) = \delta(x,y) - lp(x,y)$
Bandreject	$br(x,y) = lp_1(x,y) + hp_2(x,y)$ $= lp_1(x,y) + [\delta(x,y) - lp_2(x,y)]$
Bandpass	$bp(x,y) = \delta(x,y) - br(x,y)$ $= \delta(x,y) - [lp_1(x,y) + [\delta(x,y) - lp_2(x,y)]]$

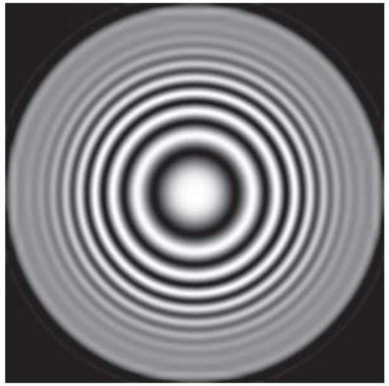




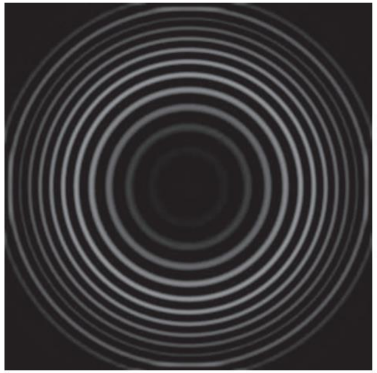
high pass filter



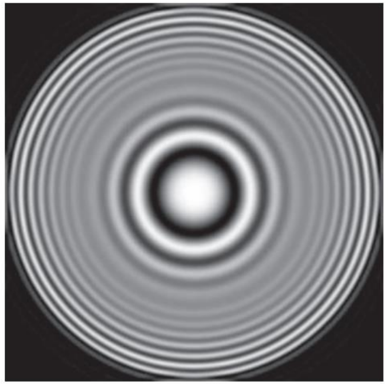
Low pass filter



Band pass filter



Band reject filter



END