Image Enhancement in Spatial Domain

Nischal Regmi

Everest Engineering College

2022

Spatial and Frequency Domains

- *Spatial Domain* refers to the representation of images as matrices where each element corresponds to intensity values.
- *Frequency Domain* refers to the representation of images as matrix, but each matrix element corresponds to the amplitude of some frequency content – this will be dealt in the next chapter.
- Some of the image processing and pattern recognition algorithms are easier to understand and implement in the spatial domain, and some in the frequency domain.
- In this chapter, we study how image to 'enhance' images using spatial domain techniques.

Image Enhancement

- 'Image Enhancement' refers to improve the quality of images for better visualization.
- The output quality of image enhancement algorithms is subjective. It is up to the human viewer who assess whether the resulting image is enhanced or not, and to select the appropriate values of the algorithm parameters.
- Spatial domain enhancement techniques could be either of three types, that uses information of a single pixel only (*point processing*), that uses the information of pixel and its neighborhood (*processing*), and that uses global information of the image.

Note: We will consider only gray scale algorithms in this chapter.

Point Processing Methods

- A point processing algorithm, and each step requires the information of single pixel at each step. Thus we think such an algorithm as a point-to-point transformation of the form

$$s = T(r)$$

  where r is the input pixel and s is the output pixel.
- A neighborhood based enhancement algorithm, at each step, requires information of a singe pixel and its neighborhood.
- A global information based algorithm, at each step, requires the information of the whole image.

Basic Enhancement Methods

Image Negative

Given that a gray scale has L levels $[0, L-1]$, the negative transformation is defined as

$$s = L - 1 - r$$

Log Transformation Log transformation is defined as

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

where w is a constant. Log transformation is used to highlight the pixels whose brightness is very low compared to the bright pixels. (How?)

Power-Law (Gamma) Transformation Power-law transformation is defined as

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

where c and r are positive constants. Similar to the log transformation, power-law transformation is used to highlight the pixels whose brightness is very low compared to the bright pixels. (How?)

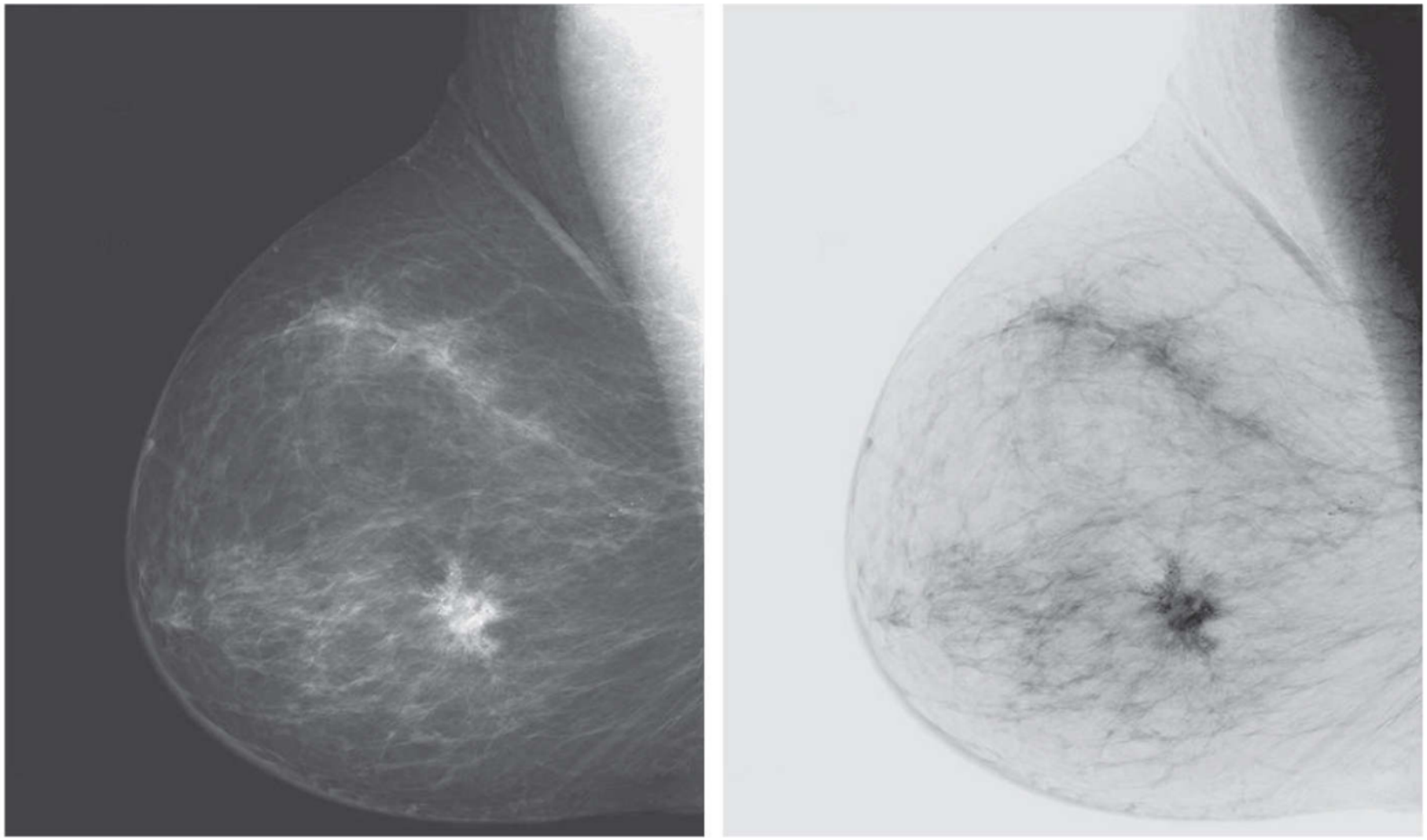


Figure: Digital image of a mammogram (left) and its negative (right)

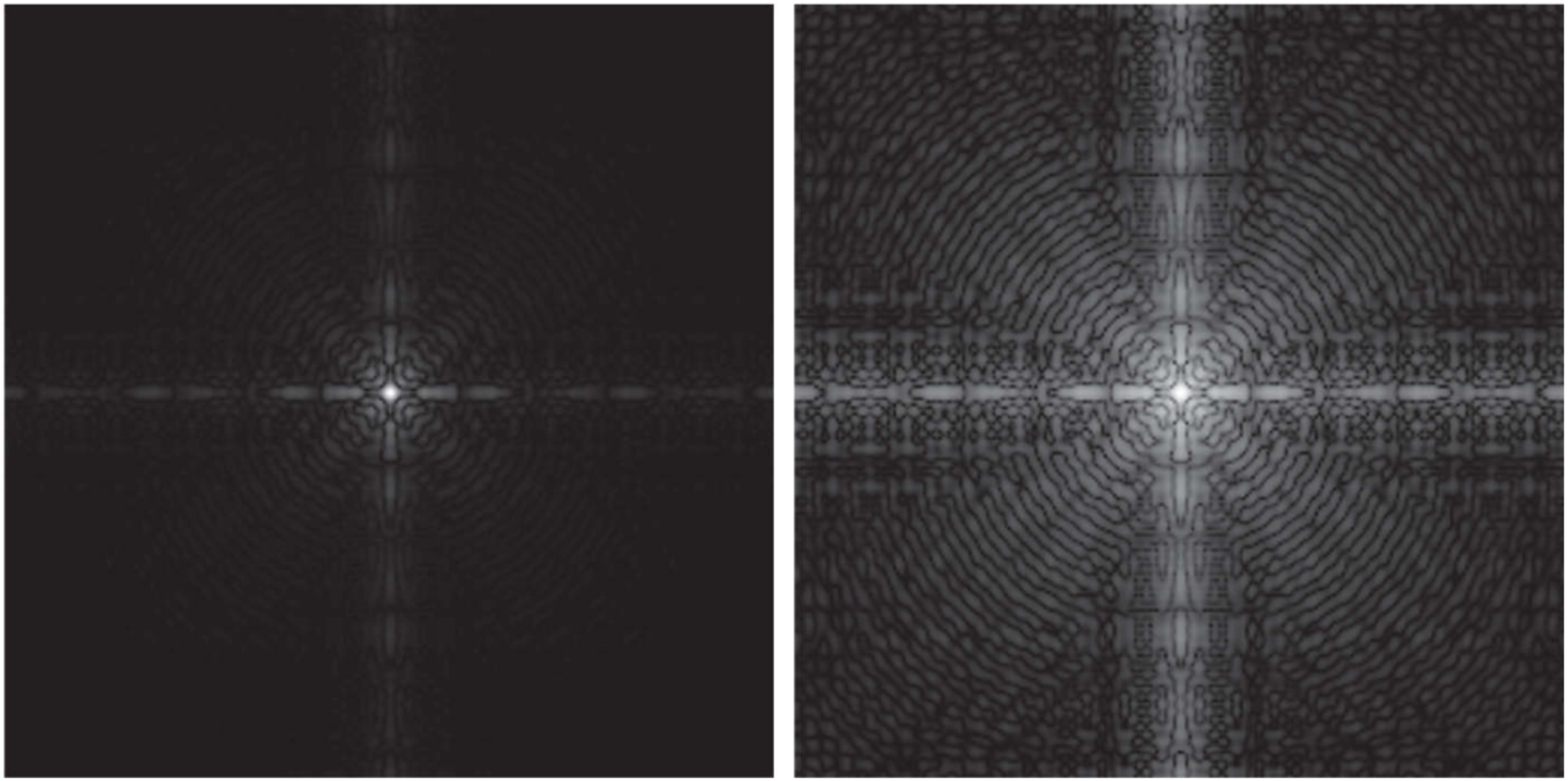


Figure: Image of a Fourier transform plot (left), and the log transform of the image (right)

Contrast Stretching and Thresholding

- *Contrast stretching* expands the range of intensity levels in an image so that it spans the ideal full intensity range.
- Contrast stretching can be achieved by the *piecewise linear transformation* shown in the figure (next slide).
- The transformation is characterized by two points (r_1, s_1) and (r_2, s_2).
  - If $r_1 = s_1$ and $r_2 = s_2$, then the transformation results in no change in the pixel intensities.
  - If $r_1 = r_2$, $s_1 = 0$, and $s_2 = L - 1$, the transformation becomes a *thresholding function.*
  - Intermediate values of (r_1, s_1) and (r_2, s_2) produce various levels of contrast stretching.

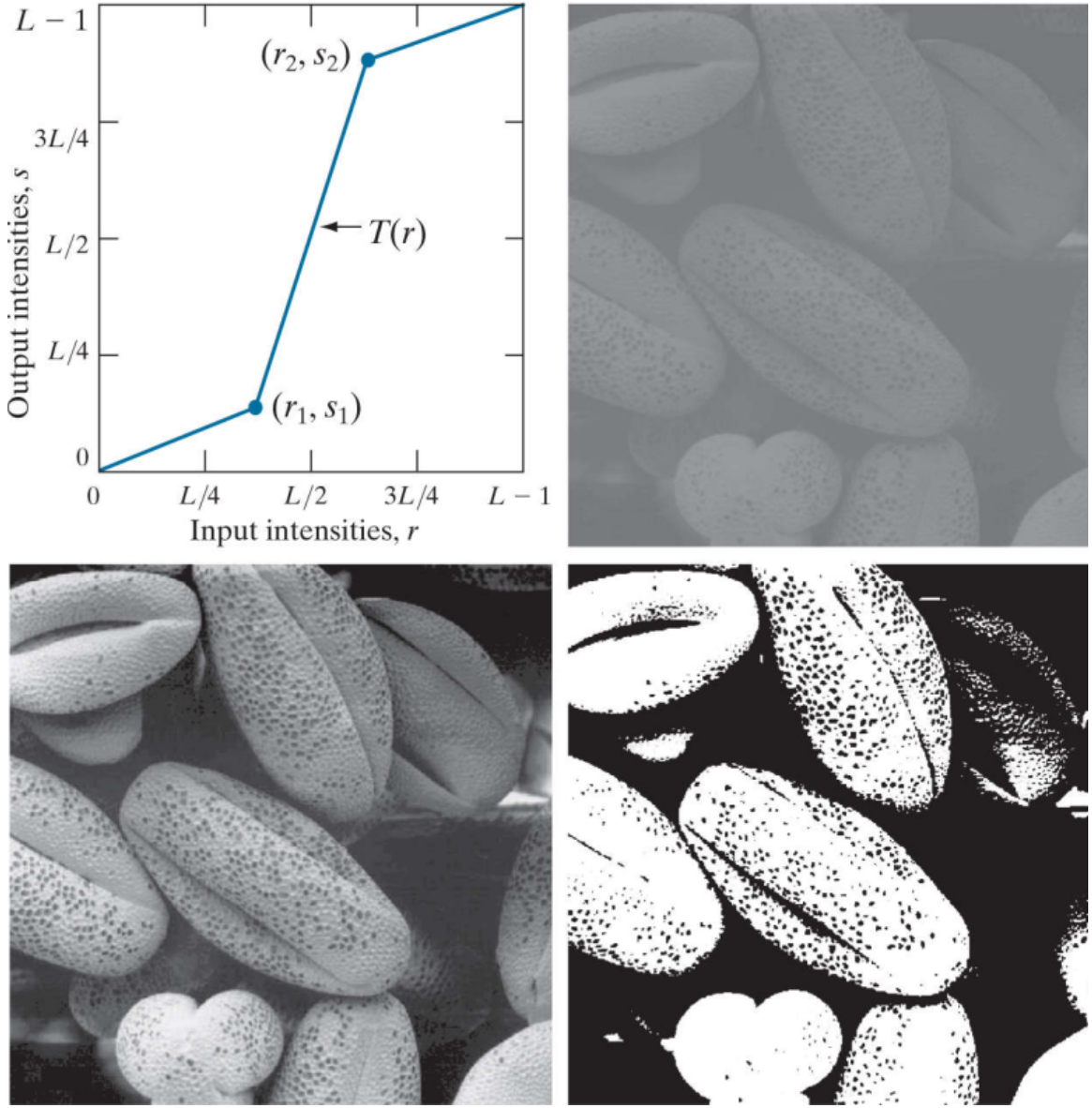


Figure: Contrast stretching function (top left), original image (top right), result of contrast stretching (bottom left), and result of thresholding (bottom right)

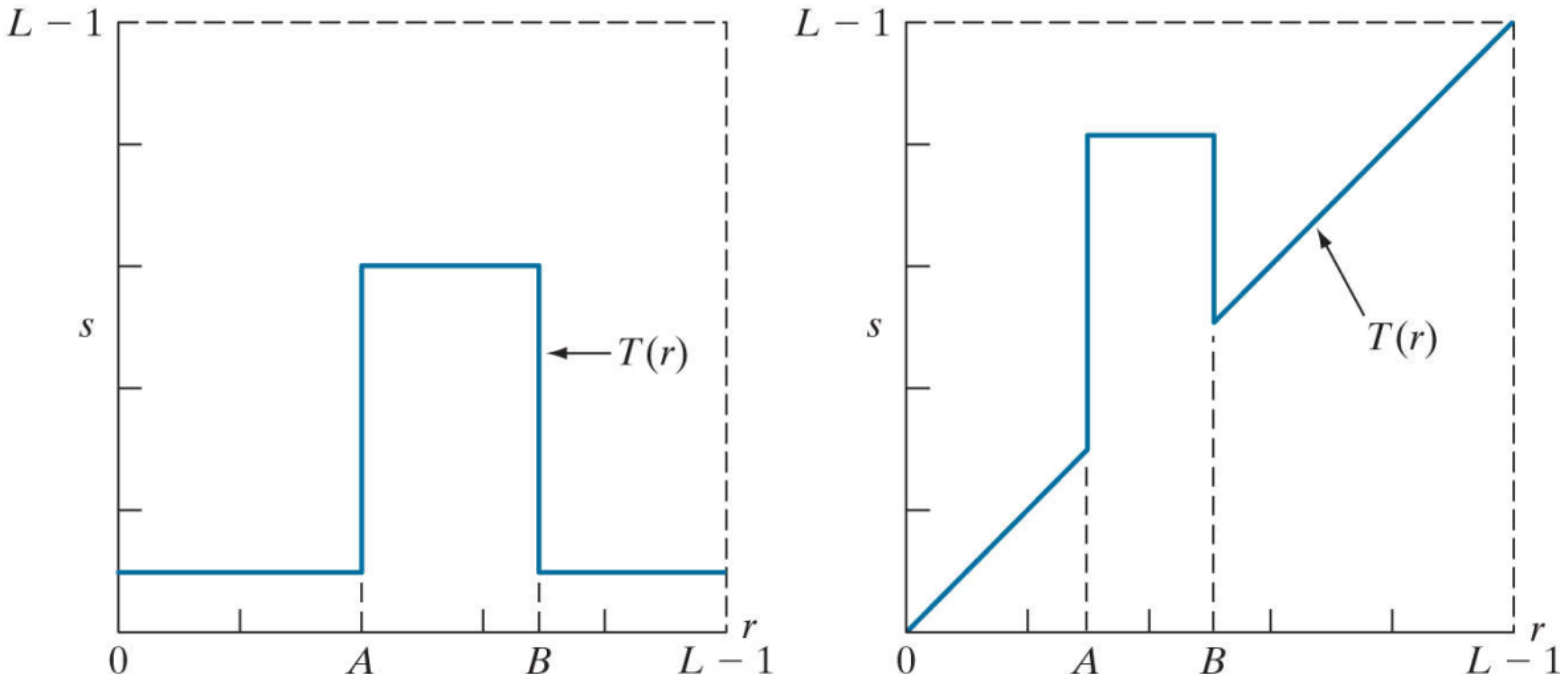
- Thresholding is not an image enhancement process. But it is required in morphological image processing which aims to identify shapes and boundaries of objects in an image.
- Thresholding transformation can be written in a simple form as following.

$$s = \begin{cases} c_l & \text{if } r \leq \theta \\ c_h & \text{otherwise} \end{cases}$$

where r and s are input and output pixel intensities, θ is the threshold, c_l and c_h are constants. We usually let $c_l = 0$ and $c_h = L - 1$ to create a binary image where all pixel having intensities higher than θ are black, and the rest are white.

Intensity-Level Slicing

Intensity-Level slicing highlights pixels within certain range of intensities.



The transformation function of the left figure highlights the pixels with intensities in the range $[A, B]$ and makes all other pixels black. The function in the right figure highlights the pixels with intensities in the range $[A, B]$, but leaves other pixels intact.

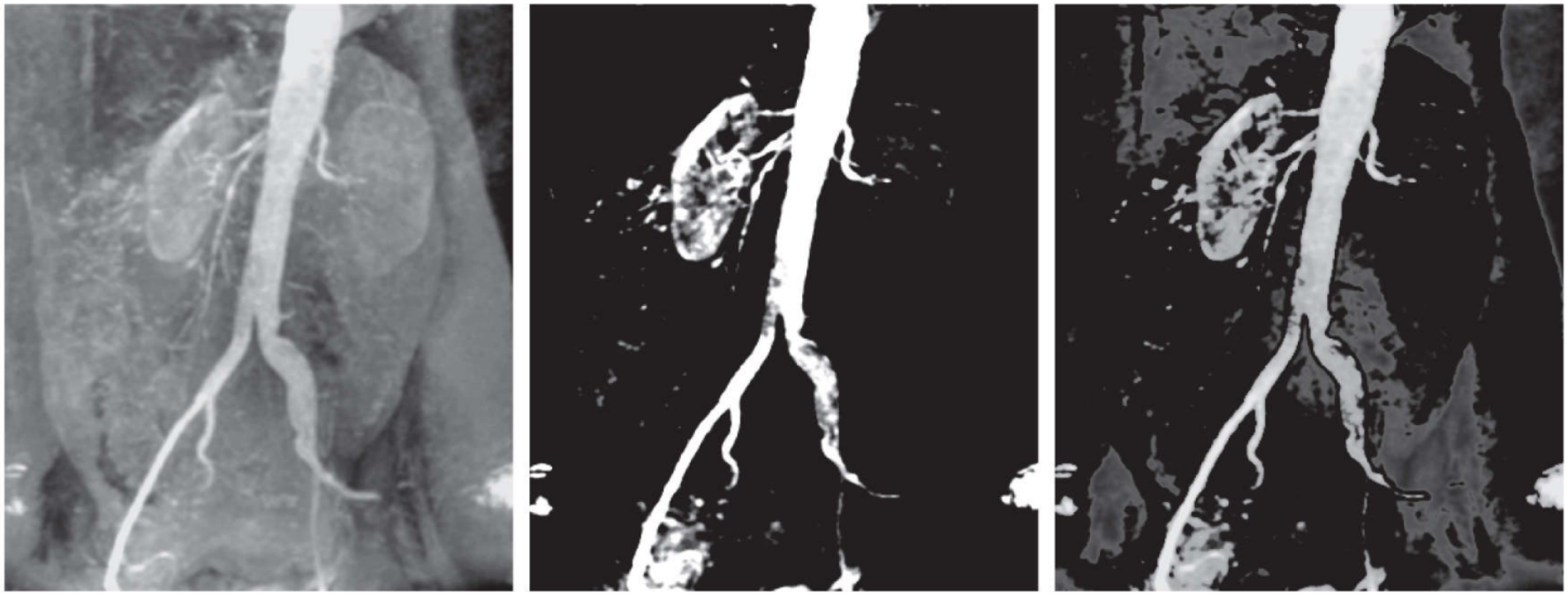
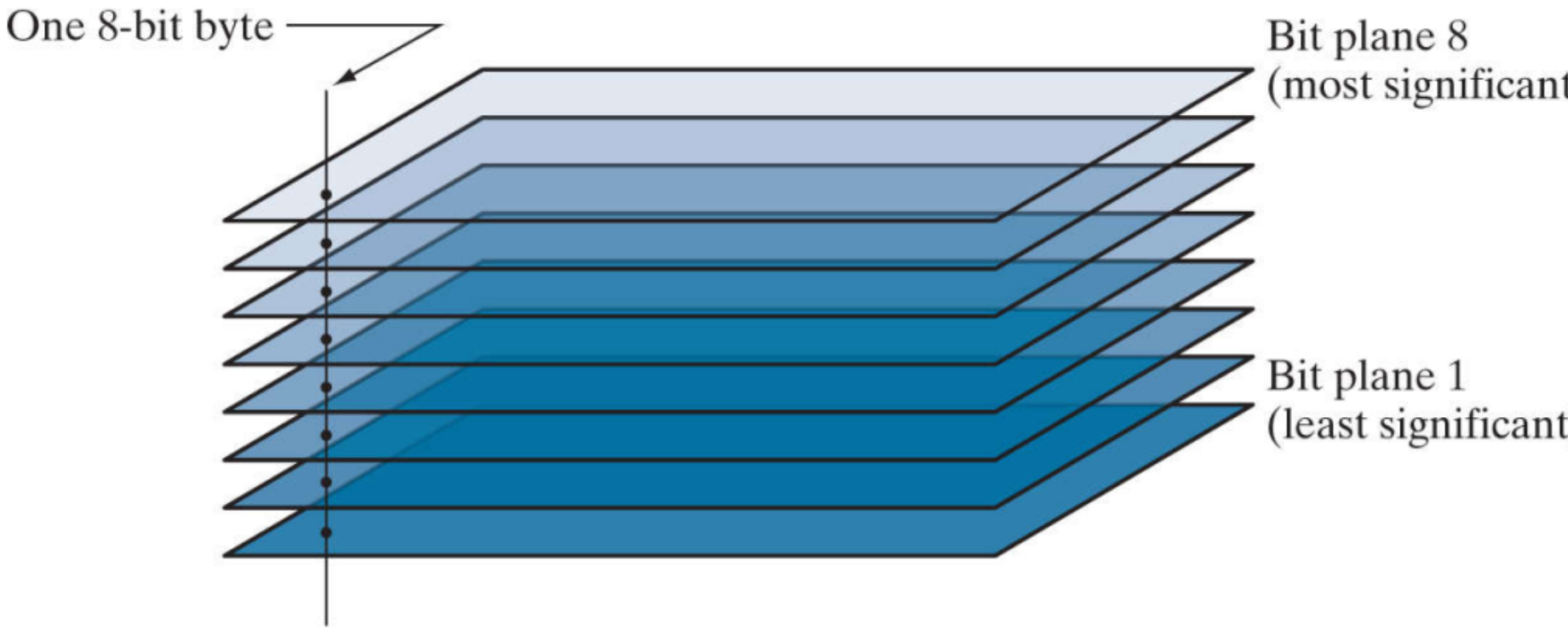


Figure: Left: Original image, Center: highlighting certain pixels blackening other, Right: highlighting certain pixels and keeping others intact

Bit-Plane Slicing

Bit-plane slicing creates separate images for each pixel bit. For an 8-bit image, there will be eight images after bit plane slicing.



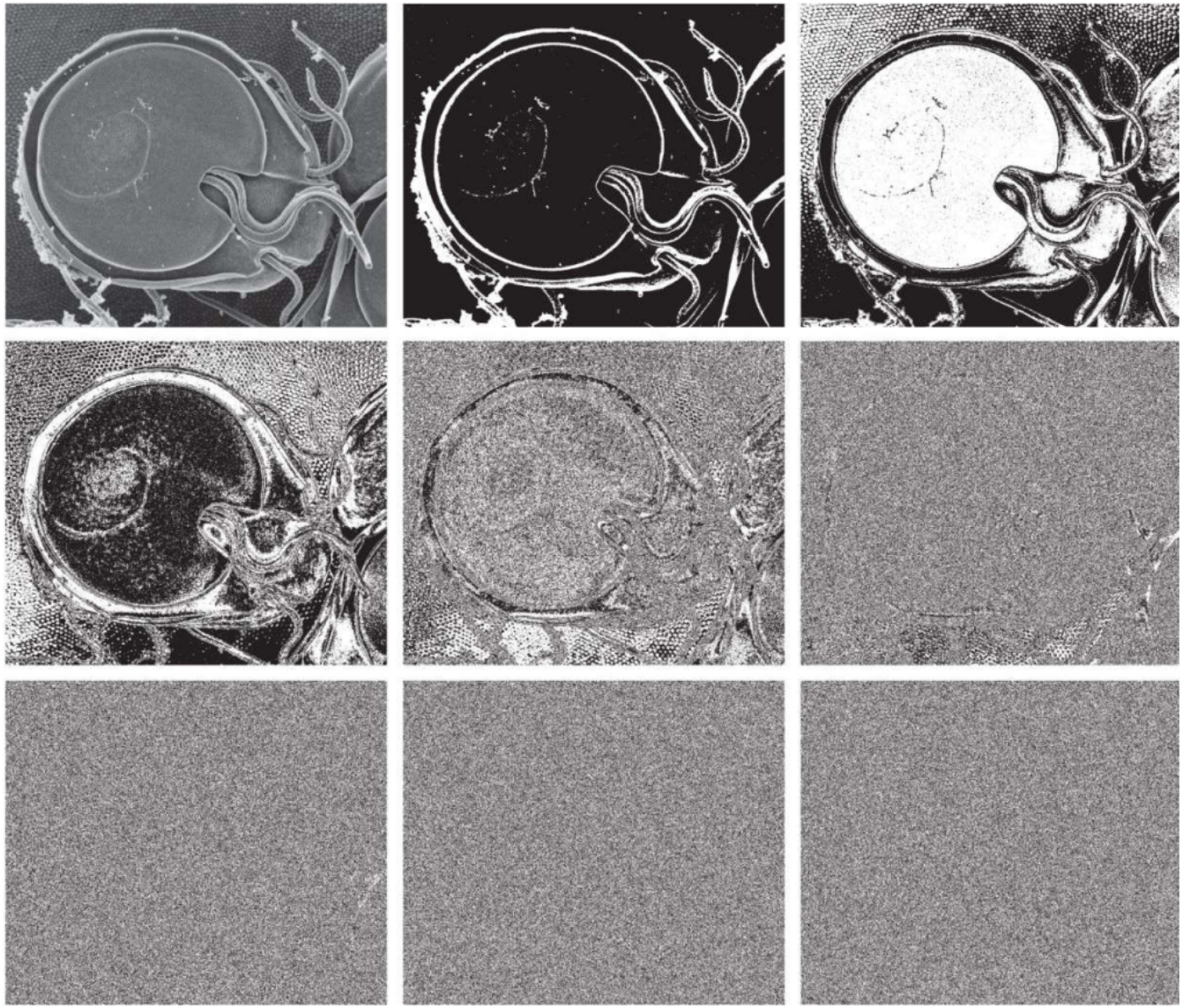


Figure: Original image (top left), and images corresponding to bits 8 to 1.

Histogram Processing

Histogram processing refers to the analysis of intensity distribution in an image.
Let r_k, for $k = 0, 1, \ldots, L-1$, denote the intensities of an L-level digital image, $f(x, y)$. The *unnormalized histogram* of f is define as

$$h(r_k) = n_k \quad \text{for } k = 0, 1, \ldots, L-1$$

where n_k is the number of pixels in f with intensity r_k. The *normalized histogram* of f is defined as

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

where M and N are the number of rows and columns in the image, respectively.

Histogram Equalization

- Histogram equalization tries to changes to reallocate the pixel intensities such that pixels with all intensities are equally probable.
- Assuming that an image can have continuous intensity values, we can derive the transformation $s = T(r)$ for a perfect histogram equalization.
  - Mathematically, this would mean that the probability density of the intensities in the resulting image is uniformly distribution in the interval $[0, L-1]$, irrespective of the distribution in the original image.
- To preserve the order of pixel intensities in the original and the transform image, we require that $T(r)$ is a *monotonic increasing function*

Let the intensities are real numbers having values in the interval $[0, L-1]$. Let $p_r(x)$ and $p_s(x)$ denote the intensity distribution in the original and transformed image. Define the transformation function as T

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

We can prove that the transformation function $T(r)$ defined in equation 1 results in a perfectly equalized image.
From probability theory, we know

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

Now,

$$\begin{aligned}\frac{ds}{dr} &= \frac{dT(r)}{dr} \\ &= (L-1)\frac{d}{dr}\left[\int_0^r p_r(w)dw\right] \\ &= (L-1)p_r(r) \quad \text{(Using Leibnizś rule)}\end{aligned}$$

Finally,

$$\begin{aligned}p_s(s) &= p_r(r)\left|\frac{dr}{ds}\right| \\ &= p_r(r)\left|\frac{1}{(L-1)p_r(r)}\right| \qquad \text{(using previous derivation)} \\ &= \frac{1}{L-1} \quad 0 \le s \le L-1\end{aligned}$$

Thus, the probability distribution of the intensities in the transformed image, $p_s(s)$ is uniform.

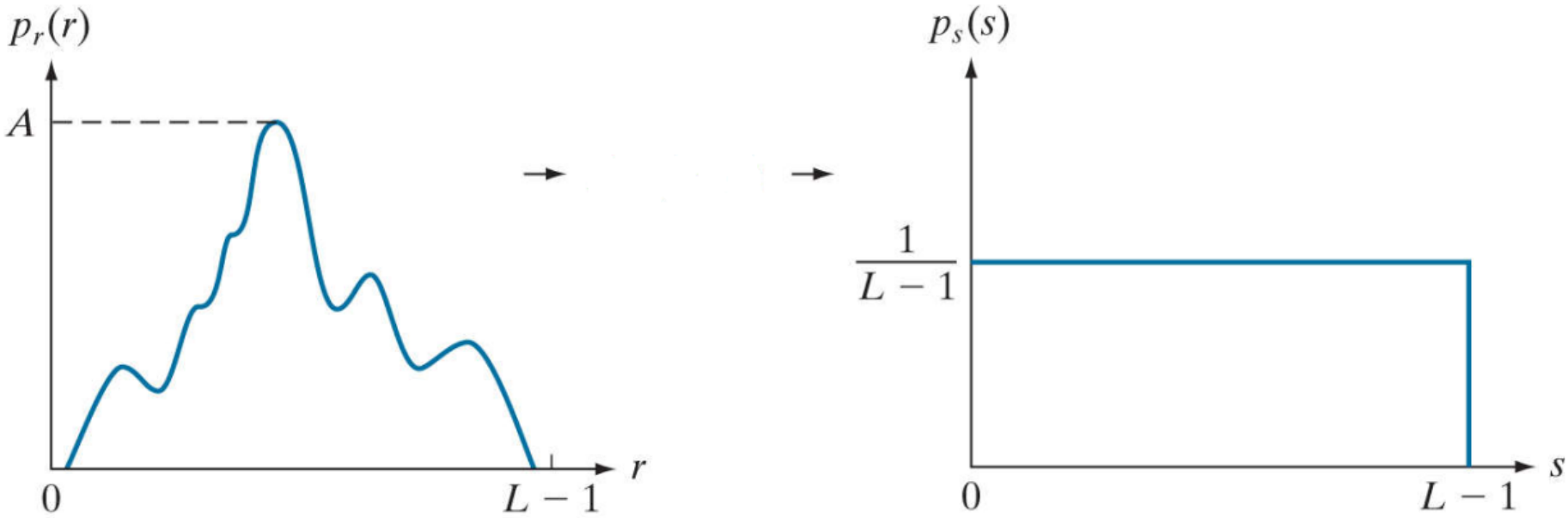


Figure: Intensity distribution of input image (left), whatever it be, is transformed into uniform distribution (right) using equation 1

Summary: If the pixel intensities are continuous in the interval $[0, L-1]$, then we can achieve a perfect histogram equalization using

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

But digital images cannot have continuous intensities, so we use the discrete version

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \quad k = 0, 1, \ldots, L-1$$

where, as defined earlier,

$$p_r(r_k) = \frac{n_k}{MN}$$

Spatial Filtering

Filters

- The term *filter* denotes a procedure that removes some frequency component from a signal, in our context, an image.
- The term 'filter' is more relevant if we work in the frequency domain. However, there are spatial domain algorithms for filtering.
- Spatial filtering algorithms are both used for noise reduction and enhancement.
- Every spatial filtering algorithm has its counterpart in the frequency domain.

Correlation and Convolution – 1D Case

Correlation and convolution are the fundamental operations in signal processing.

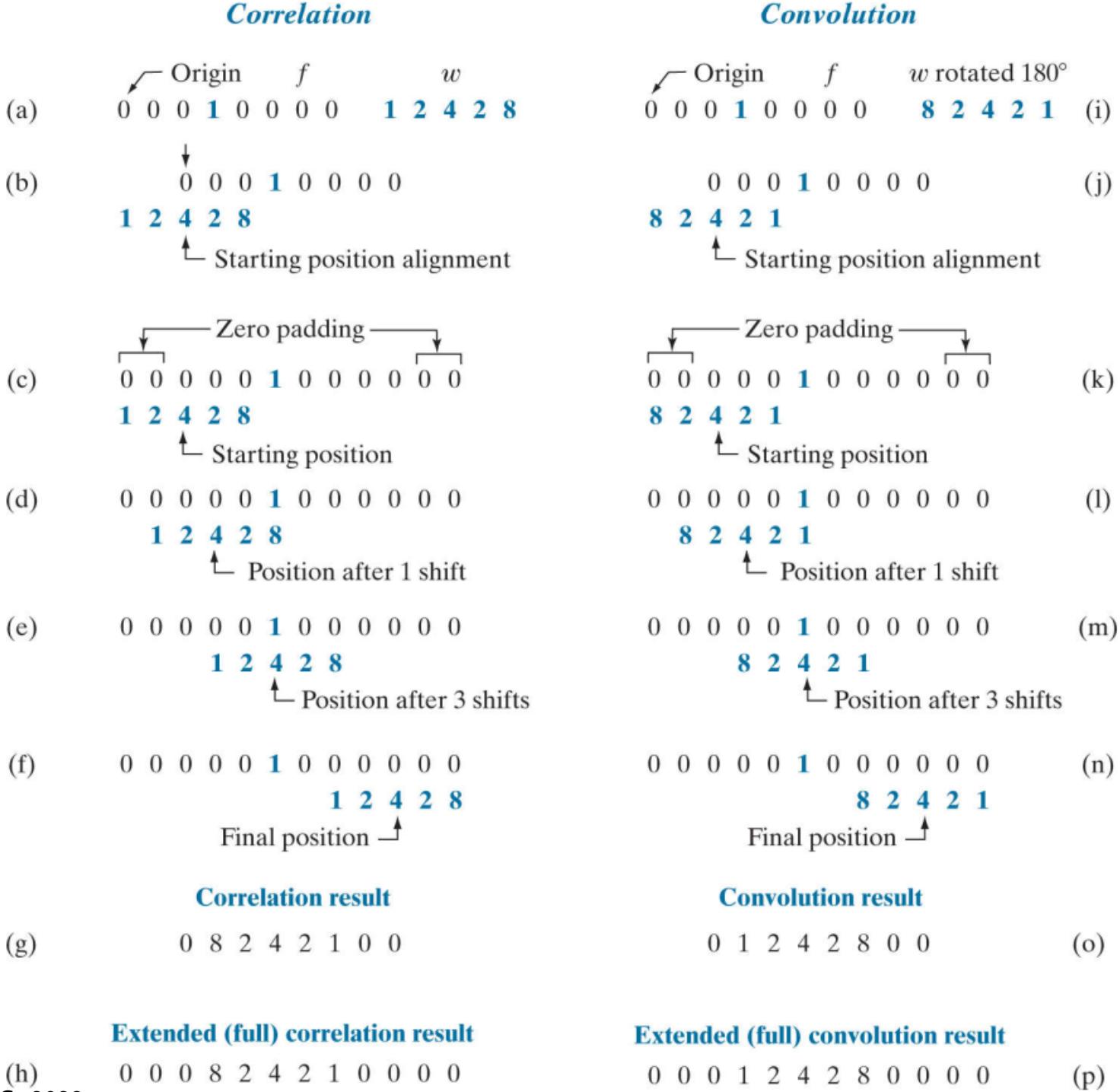
Correlation of the function $f(x)$ with the kernel $w(x)$ is defined as

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

Convolution of the function $f(x)$ with the kernel $w(x)$ is defined as

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

Note: When $f(x)$ is defined over a finite range, we cannot calculate convolution and correlation at the end points. For this, we either skip the calculations at the end points, or use the method of *zero padding*. Also, note that the kernel can be intuitively thought as a vector of odd length.


Correlation
Convolution
Origin
f
w
w rotated 180°
(a)
0 0 0 1 0 0 0 0
1 2 4 2 8
(i)
8 2 4 2 1
(b)
(j)
Starting position alignment
Zero padding
(c)
(k)
Starting position
(d)
(l)
Position after 1 shift
(e)
(m)
Position after 3 shifts
(f)
(n)
Final position
Correlation result
Convolution result
(g)
0 8 2 4 2 1 0 0
0 1 2 4 2 8 0 0
(o)
Extended (full) correlation result
Extended (full) convolution result
(h)
0 0 0 8 2 4 2 1 0 0 0 0
0 0 0 1 2 4 2 8 0 0 0 0
(p)

Correlation and Correlation – 2D Case

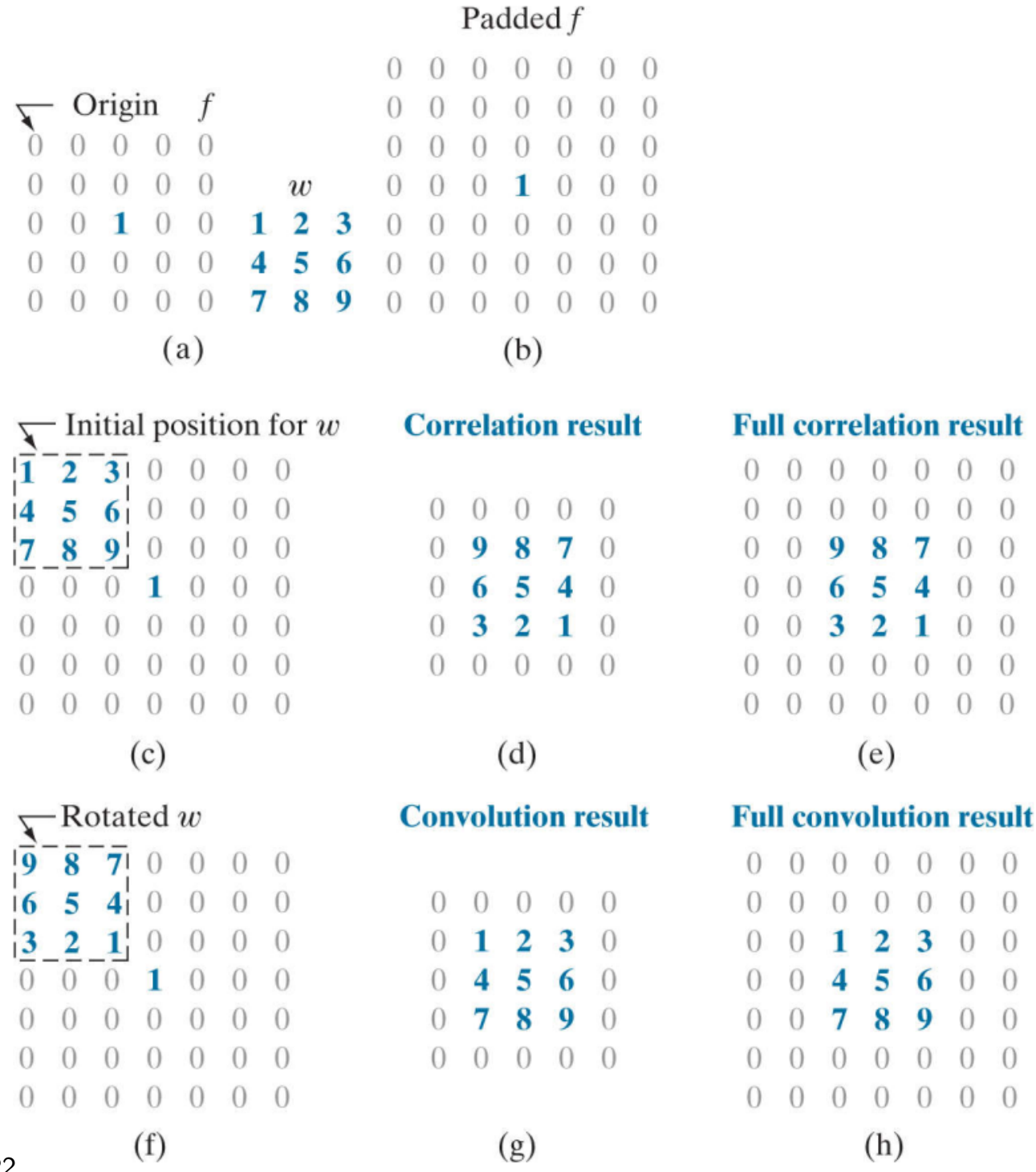
In the two-dimensional case, correlation is of a function $f(x, y)$ with the kernel $f(x, y)$ is defined as

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

Convolution is defined as as

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

Note: As with the one dimensional case, to calculate the correlation and convolution at the boundary points, either we need to zero-pad the original image, or we should ignore the boundary points. As with the 1D case, the kernel can be thought as a matrix with odd number of rows and columns.


Origin
f
w
(a)
Padded f
(b)
Initial position for w
(c)
Correlation result
(d)
Full correlation result
(e)
Rotated w
(f)
Convolution result
(g)
Full convolution result
(h)

Basics of Spatial Filtering

- The procedure of spatial filtering is similar to convolution/correlation.
- Many of the spatial filters, like averaging, are actually convolution with different kernel.
- Some of the spatial filters, like median filters, mimic the convolution algorithm, where calculations are done by conceptually sliding a window.

Averaging Filter

- Averaging is achieved by convolution of an image with a kernel of $N \times N$ size, N odd, where each element of the kernel has the value $1/N^2$. For example, a 3×3 kernel for averaging is

$$w = \begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix}$$

- Averaging filter does not enhance the image. On the contrary, averaging results in a blurred image.
- Averaging is used to reduce *Gaussian noise* in the image.

Median Filter

- A $m \times m$ median filter operates in the following way
  1. For each pixel, identify the pixel values in its $m \times m$ neighborhood; for the boundary pixels, we zero-pad the images (alternatively, we could also ignore the boundary pixels).
  2. Let ξ be the median value of the neighborhood.
  3. Replace the pixel being considered by the median, ξ.
- Median filter belongs the the type of *order-statistic filters*.
- Median filter is used to reduce impulse noise, also called as salt-pepper noise, from the image.

Spatial Low and High Pass Filters

- Any spatial filter that blurs the image is a low pass filter. For example, averaging is a low pass filter.
  - A spatial low pass filter corresponds to the removal of high-frequency components from the image in the frequency domain.
- Any spatial filter that sharpens the image is a high pass filter. For example, Laplacian and gradient are high pass filters.
  - A spatial high pass filter corresponds to the removal of low-frequency components from the image in the frequency domain.

Derivatives as High Pass Filters

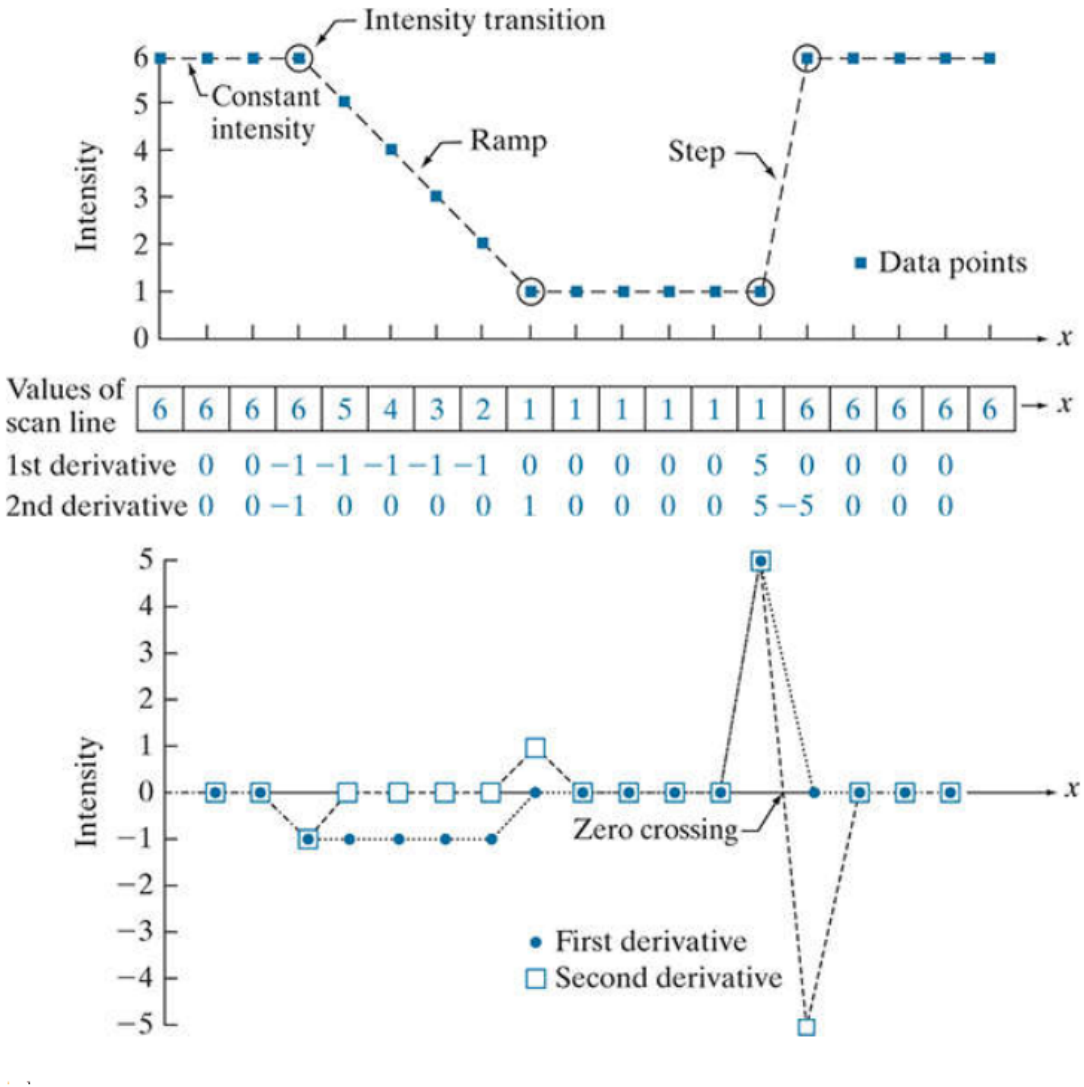
- For an image $f(x, y)$, the first order derivatives are

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

  The second order derivatives are

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

- Derivatives have higher values at the points of sharp intensity transitions; thus derivatives can be used for edge detection.
- Derivatives are high pass filters, because image regions with high frequency components have high values of derivatives.



(a) A section of a horizontal scan line from an image, showing ramp and step edges, as well as constant segments. (b)Values of the scan line and its derivatives. (c) Plot of the derivatives, showing a zero crossing. In (a) and (c) points were joined by dashed lines as a visual aid.

Laplacian Filter

The Laplacian of a function $f(x, y)$ is defined as

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

Since,

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

We get

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

We can compute image Laplacian, based on the previous equation, using convolution of the image $f(x, y)$ with the following kernel

$$w = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Note:

- Above formulation of image Laplacian does not consider change of intensity along the diagonal direction. If we consider the diagonal direction, we get an eight-point formula for the Laplacian. The kernel for the eight-point Laplacian is shown in the next slide.
- Also, the Laplacian defined in above equation is positive if the point considered is darker than its surrounding. One could reverse the signs to have a Laplacian kernel with the opposite interpretation.
- Thus, there could be eight possible Laplacian kernels; we need to be careful while interpreting the values of these operators.

0	1	0
1	−4	1
0	1	0

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

Figure: Left two images: four- and eight-point Laplacian kernels. Right two images: same kernels with reverse interpretation for intensity change

Laplacian Enhancement

- The value of image Laplacian is higher where there is sharp transition in intensity, i.e. where the image is more detailed.
- Thus, addition of the Laplacian to the original image enhances the details in an image.
- Let $f(x, y)$ be the image. The enhanced image, $g(x, y)$ can be obtained as

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

where $c = -1$ if we take a Laplacian kernel with negative center, and $c = 1$ if we take a Laplacian kernel with positive center. (Why?)

Image Gradient

Given an image $f(x, y)$, its gradient is

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

Gradient is a vector, whereas pixel intensities are scalar. So we compute the magnitude of the gradient

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

However, computing squares and square-root could be computationally extensive, because there are thousands of pixels in any realistic image. So we use the following approximation

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

The kernels for the gradients g_x and g_y can be obtained in various ways. Refer to the figure (in the following slide) for the notation used.

Robert's Cross Gradient Operator

$$g_x = (z_9 - z_5)$$
$$g_y = (z_8 - z_6)$$

Sobel's Operator

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$
$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

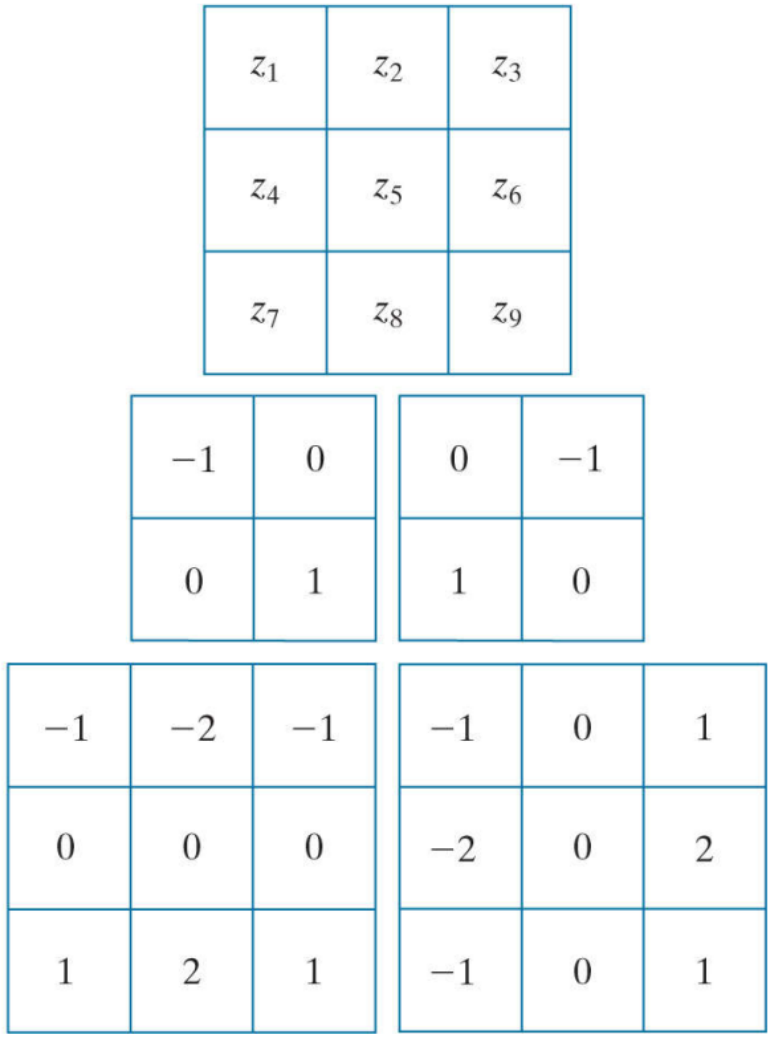


Figure: Top: neighborhood, we need to calculate gradient at g_5. Middle: kernels (g_x and g_y) for Robert's cross operator. Bottom: kernels (g_x and g_y) for Sobel's operator.

Combining Spatial Enhancement Methods

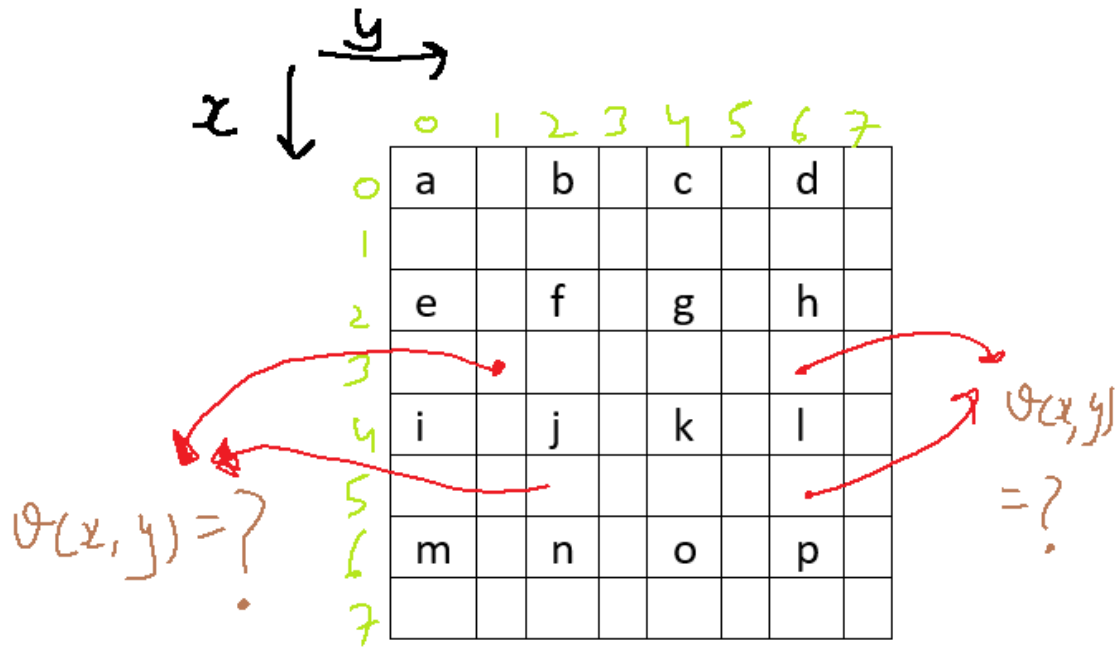
- Single image enhancement technique may not fulfill the needs of real applications.
- In practical situations, we may need to apply more than one enhancement techniques.

Image Interpolation (Magnification/Zooming)

The Basic Setup

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Original Image



Empty spaces after zooming two times

Problem: Find $\theta(x, y)$ for all empty cells.

Image Interpolation Techniques

Nearest Neighbor Interpolation

- For each empty cell generated after enlarging the image, assign the intensity value from the nearest non-empty cell.

Bilinear Interpolation Let (x, y) the coordinates in the enlarged image, and $v(x, y)$ being non-empty, fit the following nonlinear regression model. Suppose we need to calculate the intensity at point (x_p, y_p). Let (x_1, y_1), (x_2, y_2), (x_3, y_3) and (x_4, y_4) be the coordinates of four nearest non-empty cells from (x_p, y_p).
We first solve the following set of linear equations for a, b, c, and d

$$ax_1 + by_1 + cx_1y_1 + d = v(x_1, y_1)$$
$$ax_2 + by_2 + cx_2y_2 + d = v(x_2, y_2)$$
$$ax_3 + by_3 + cx_3y_3 + d = v(x_3, y_3)$$
$$ax_4 + by_4 + cx_4y_4 + d = v(x_4, y_4)$$

Then we find the intensity at $v(x_p, y_p)$ as

$$v(x_p, y_p) = ax_p + bx_py_p + cx_py_p + d$$

Bicubic Interpolation The idea of bicubic interpolation extends the bilinear interpolation. Instead of fitting a four-parameter formula, bicubic interpolation fits a sixteen-parameter formula using the values of sixteen nearest non-empty cells from the empty cell being considered. The formula to be fitted is

$$v(x, y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$$

Note that in this equation, the sixteen parameters to be estimated are $a_{0,0}, a_{0,1}, \ldots, a_{15,15}$.