

Chapter 2: Digital Image Quality Enhancement

Dr. MAAMAR BEKHOUCHE

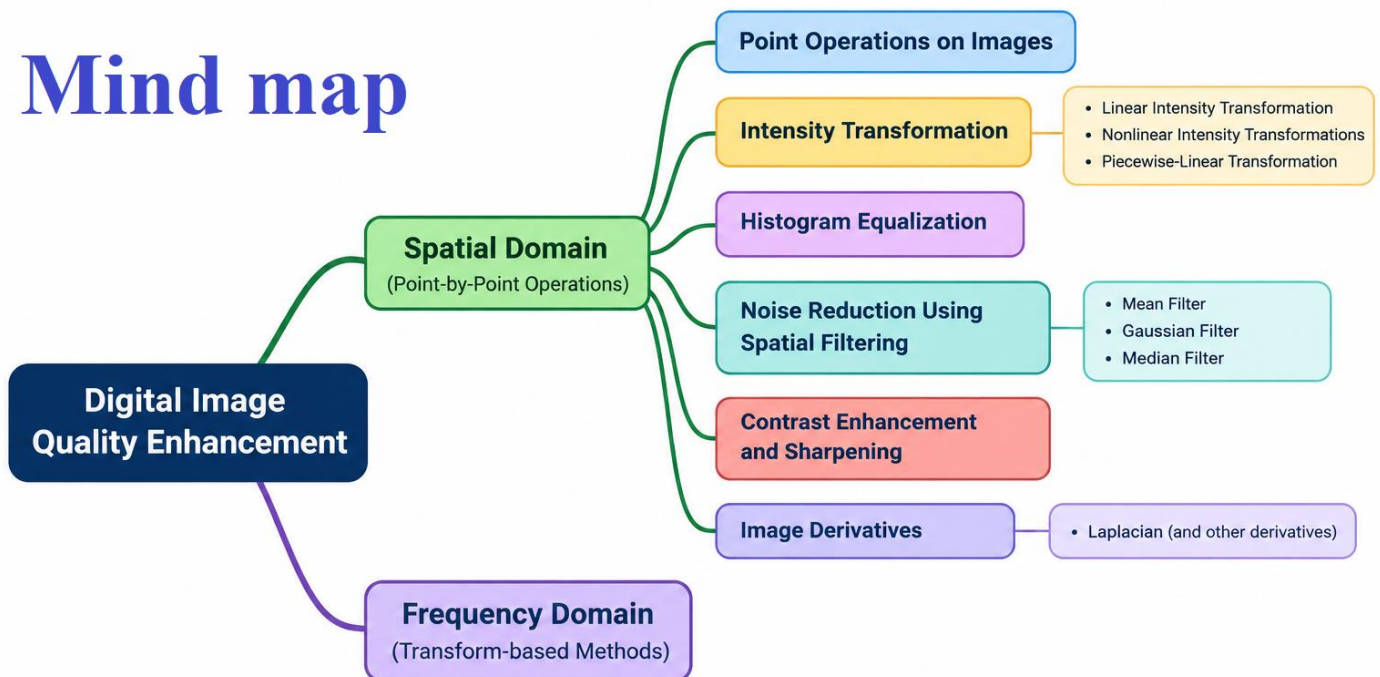
m.bekhouche@centre-univ-mila.dz

2026-2027

Image quality enhancement is a fundamental stage in digital image processing. Its primary objective is to modify the visual characteristics of an image to render it more suitable for a human observer or to optimize performance prior to automated analysis tasks such as segmentation, object detection, and pattern recognition.

This chapter explores essential spatial domain techniques: point operations, histogram processing, spatial filtering for noise reduction, and contrast enhancement using derivative operators.

Mind map



1. Point Operations on Images

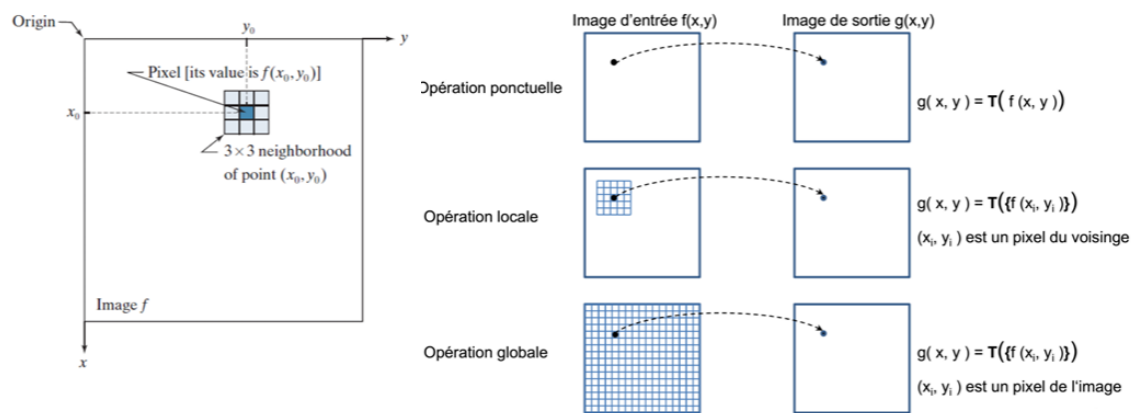
1.0. Introduction and Mathematical Formulation

Point operations (also known as intensity transformations) represent the simplest and most fundamental class of spatial processing techniques. In a point operation, the output pixel intensity depends strictly and exclusively on the intensity of the corresponding input pixel.

Let $f(x,y)$ denote the input image and $g(x,y)$ the resulting output image. A general point transformation is expressed as:

$$g(x, y) = T[f(x, y)]$$

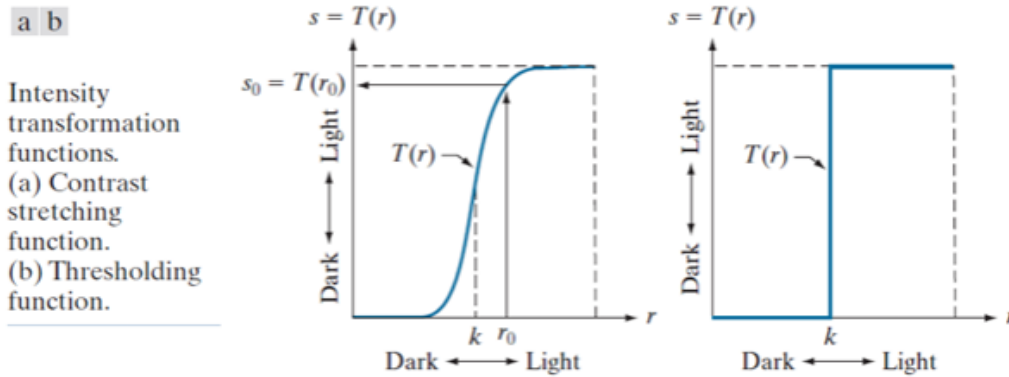
where T denotes the transformation operator applied to the intensity or color value at pixel location (x, y) .



From a neighborhood perspective, a point operation corresponds to a minimum neighborhood size of 1×1 . We can simplify the notation by operating directly on gray levels:

$$s = T(r)$$

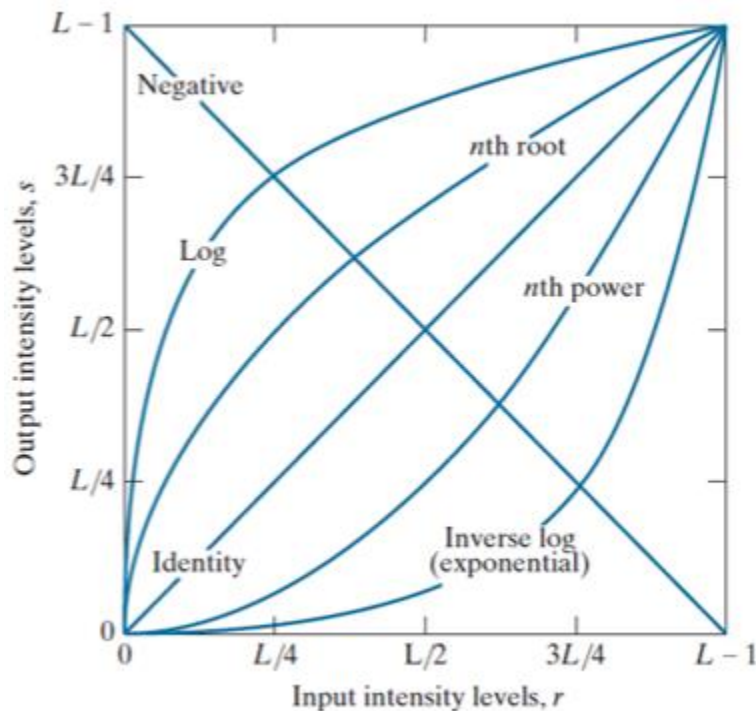
where $r \in [0, L-1]$ represents the input gray level, $s \in [0, L-1]$ represents the transformed output gray level, and L denotes the total number of available intensity levels (e.g., $L = 256$ for an 8-bit image).



1.1. Intensity Transformation

Intensity transformations are fundamental techniques in digital image processing used to modify the intensity values of individual pixels in an image. They operate directly on the input intensity (r) to produce a corresponding output intensity (s) according to a predefined transformation function: $s = T(r)$

Different transformation functions produce different effects on the intensity distribution of an image, such as preserving, reversing, compressing, or expanding specific intensity ranges. The following figure presents the geometric representation of some simple intensity transformations,



By appropriately modifying pixel intensities, these transformations can enhance image contrast, highlight specific intensity ranges, emphasize details, or improve the visual appearance of an image. Depending on the form of the transformation function, intensity transformations can be classified into several types, the following sections present some common examples of these transformations.

1.1.1. Linear Intensity Transformation

A linear transformation directly adjusts global brightness and contrast through an affine relationship:

$$s = a \cdot r + b$$

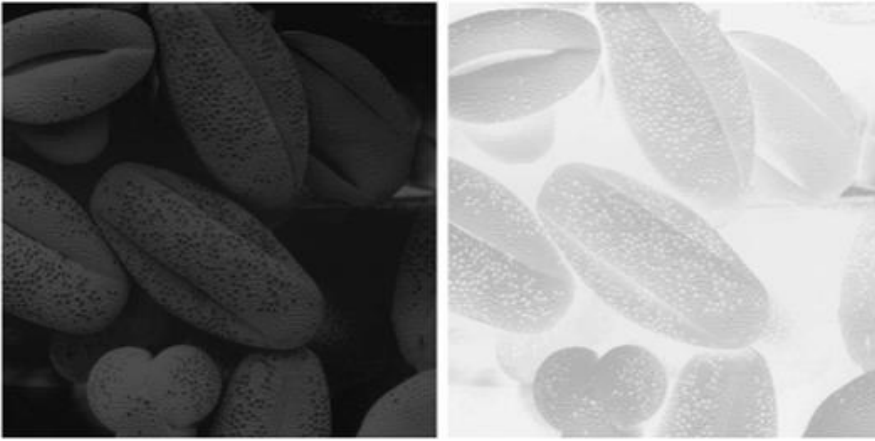
- **Parameter a (slope):** Controls image contrast.
If $a > 1$, contrast increases;
if $a < 1$, contrast decreases.
- **Parameter b (offset):** Controls image brightness.
If $b > 0$, the image becomes uniformly brighter.

- Image Negative

The negative of an image with intensity levels in the range $[0, L-1]$ is obtained by using the negative transformation function of the form:

$$s = L - 1 - r$$

This type of processing is used, for example, in enhancing white or gray detail embedded in dark regions of an image, especially when the black areas are dominant in size:



Original image and its negative counterpart

1.1.2. Nonlinear Intensity Transformations

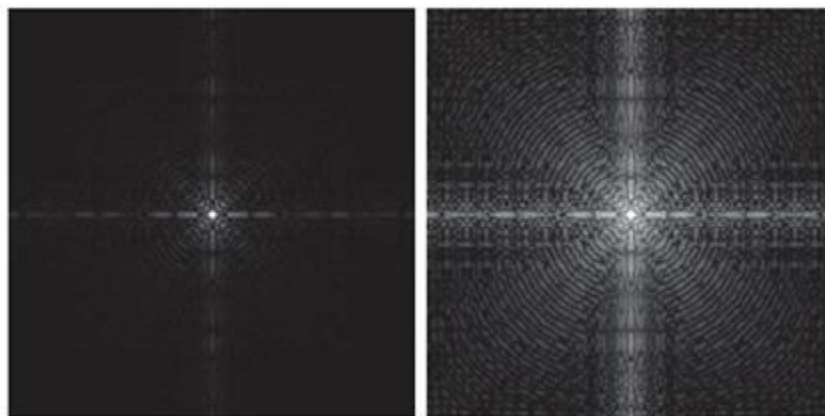
- Logarithmic Transformation

A logarithmic transformation compresses the dynamic range of high-intensity values while expanding low-intensity values. Its general mathematical form is:

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

where c is a positive scaling constant designed to map the output to the full dynamic range (typically $c = (L-1) / \log(1 + r_{\max})$). The addition of 1 ensures that $\log(1) = 0$ when $r = 0$.

We use a transformation of this type to expand the values of dark pixels in an image, while compressing the higher-level values. The opposite is true of the inverse log (exponential) transformation.



Effect of applying Log transform

- Power-Law (Gamma) Transformation

Power-law transformations, commonly referred to as gamma transformations, are defined as:

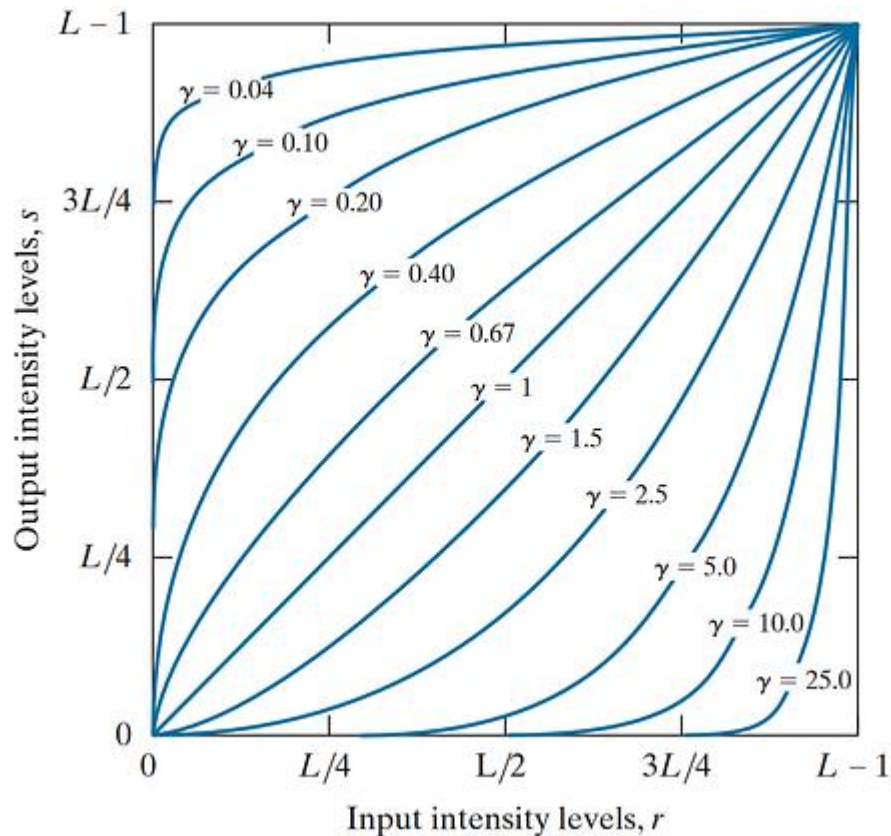
$$s = cr^\gamma$$

where c and γ are positive constants. Inputs are usually normalized to the interval $[0, 1]$.

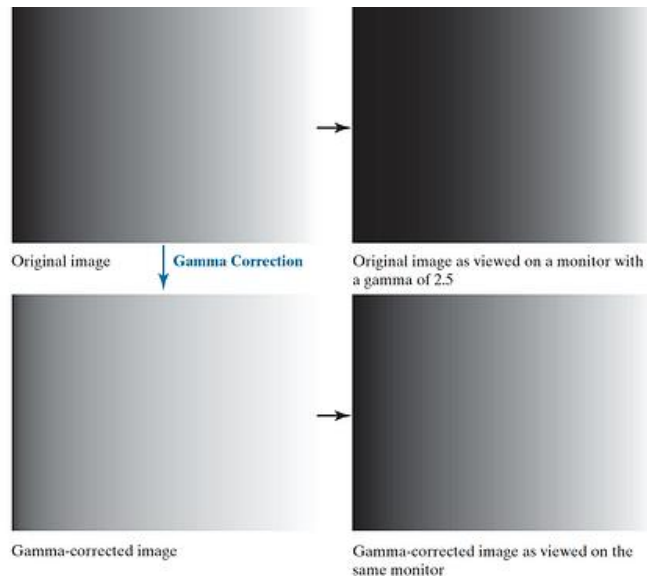
- $\gamma < 1$: Expands dark gray levels and compresses bright levels (similar to the logarithmic transformation).
- $\gamma > 1$: Compresses dark levels and expands bright levels.
- $\gamma = 1$: Identity linear transformation (when $c = 1$).

This transformation is indispensable for gamma correction across capture devices and monitors, whose inherent physical luminance responses are highly non-linear.

The following figure shows plots of s as a function of r for various values of γ :



The process used to correct these power-law response phenomena is called **gamma correction** or gamma encoding. For example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function. As you can see in the image below, such display systems would tend to produce images that are darker than intended:



Basic Gamma transformation:

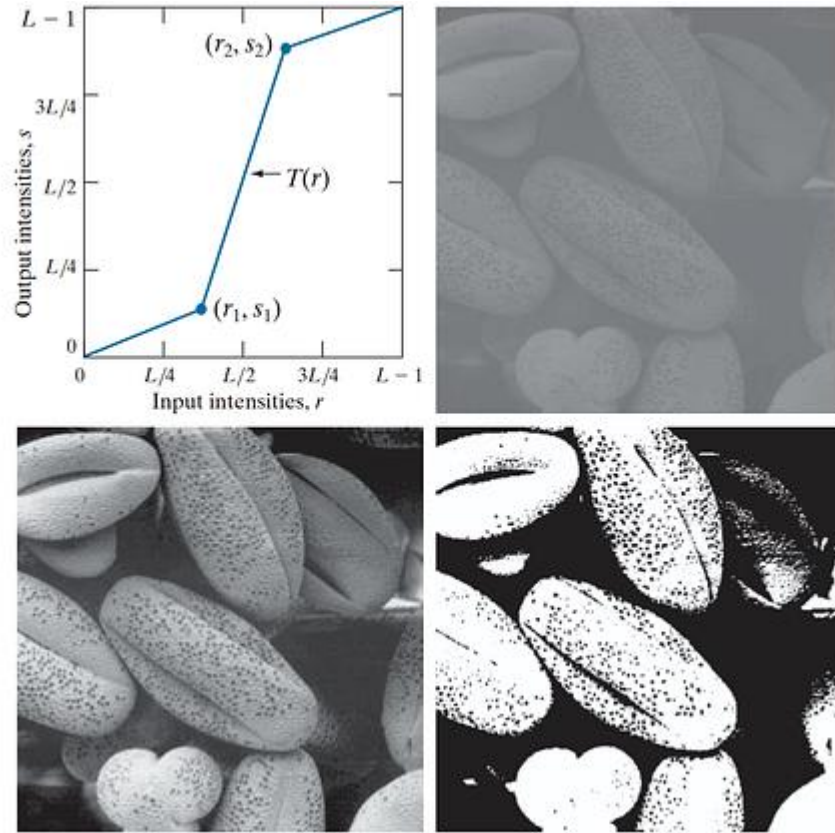


1.1.3. Piecewise-Linear Transformation

- Contrast Stretching**

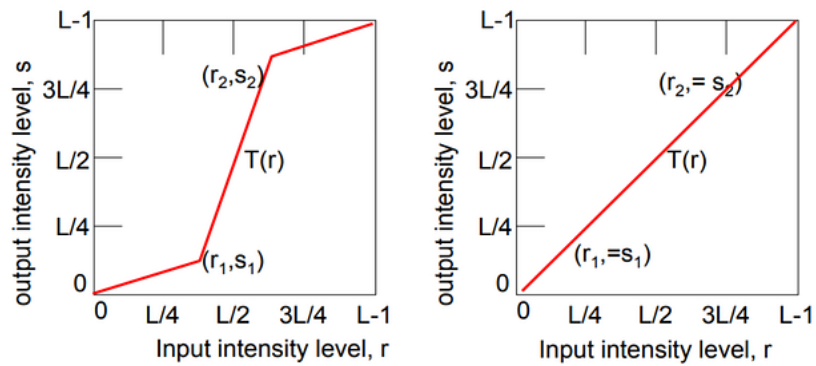
The advantage of piecewise functions over those discussed thus far is that the form of piecewise functions can be arbitrarily complex. Low-contrast images can result from poor illumination, lack of dynamic range in the imaging sensor, or even the wrong setting of a lens aperture during image acquisition. Contrast stretching expands the range of intensity levels in an image so that it spans the ideal full intensity range of the display device.

The figure shows a typical transformation used for contrast stretching:

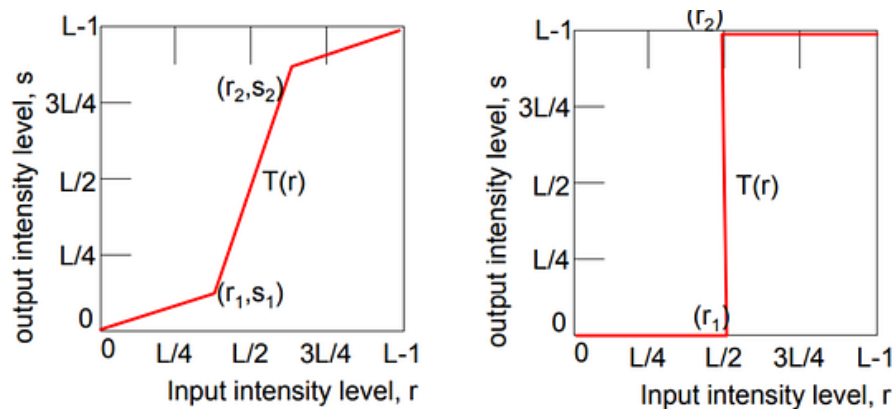


$$s = T(r) = \begin{cases} \frac{s_1}{r_1} r, & 0 \leq r \leq r_1 \\ \frac{s_2 - s_1}{r_2 - r_1} (r - r_1) + s_1, & r_1 < r \leq r_2 \\ \frac{(L-1) - s_2}{(L-1) - r_2} (r - r_2) + s_2, & r_2 < r \leq L-1 \end{cases}$$

If $r_1=s_1$ and $r_2=s_2$, the transformation is a linear function that produces no changes in intensity:

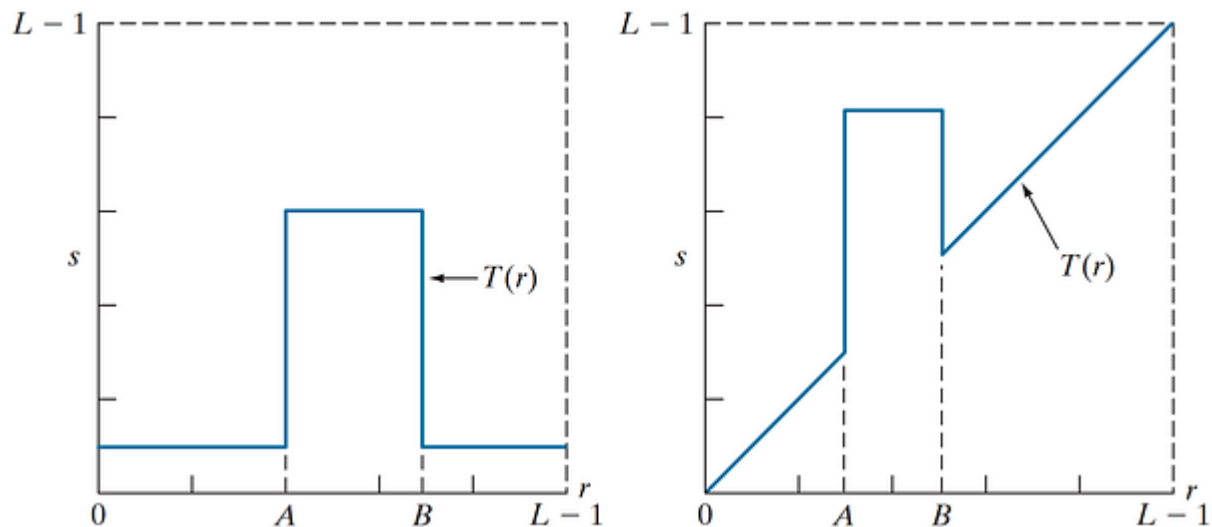


If $r_1=r_2$, $s_1=0$ and $s_2=L-1$, the transformation becomes a **thresholding function** that creates a binary image:



- Intensity level Slicing

There are applications where we want to highlight a specific range of intensities in an image (enhancing features in satellite imagery, and X-ray images). Here, intensity-level slicing can be implemented. Most implementations are variations of two basic themes:

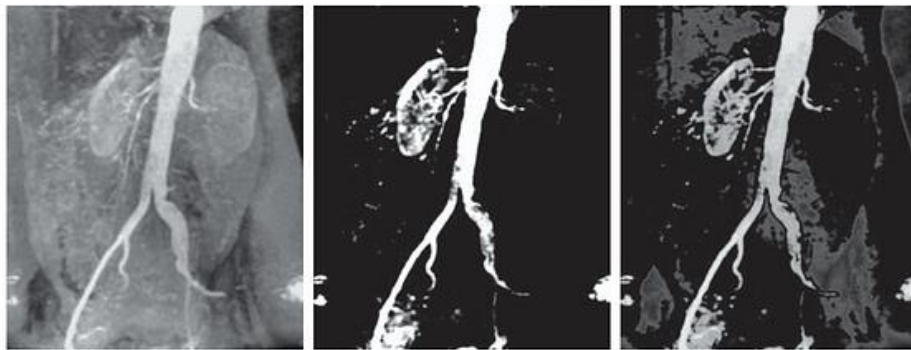


- Left: display in one value (say, white) all the values in the range of interest and in another (say, black) all other intensities

- Right: brightens (or darkens) the desired range of intensities, but leaves all other intensity levels in the image unchanged

Let A=3 and B=5. Without background preservation, we can use intensity slicing as follows:

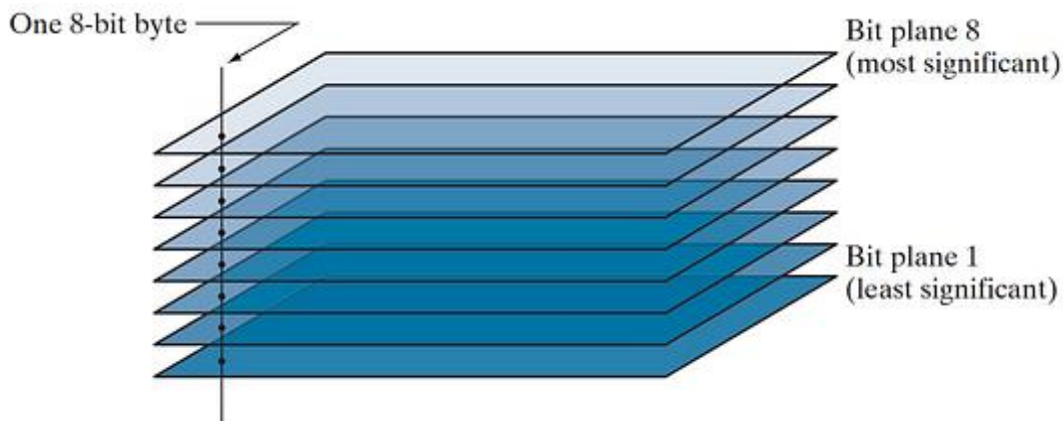
$$I = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \\ 3 & 4 & 5 & 6 \\ 4 & 5 & 6 & 7 \end{bmatrix} \quad I_1 = \begin{bmatrix} 0 & 0 & 7 & 7 \\ 0 & 7 & 7 & 7 \\ 7 & 7 & 7 & 0 \\ 7 & 7 & 0 & 0 \end{bmatrix} \quad I_2 = \begin{bmatrix} 1 & 2 & 7 & 7 \\ 2 & 7 & 7 & 7 \\ 7 & 7 & 7 & 6 \\ 7 & 7 & 6 & 7 \end{bmatrix}$$



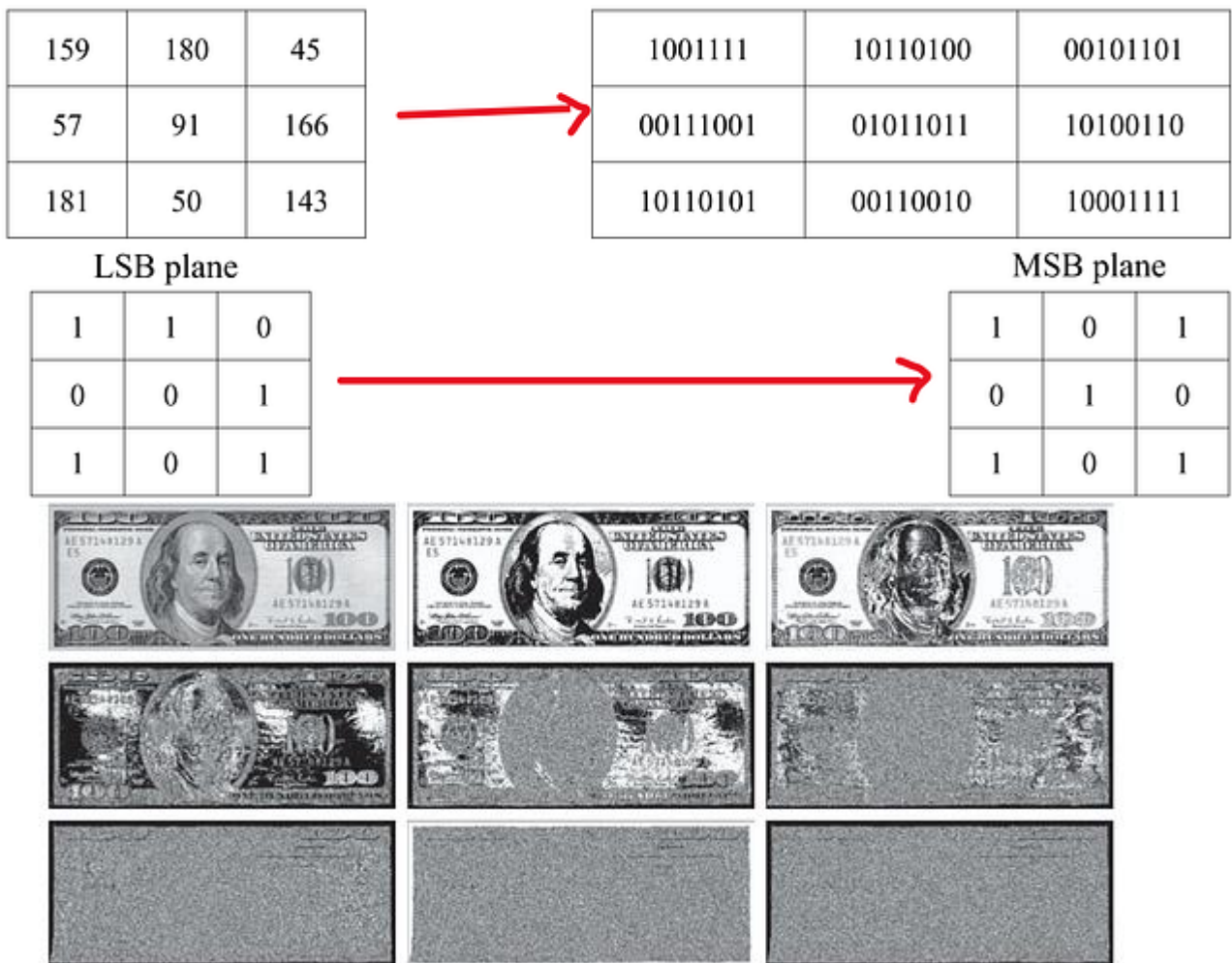
Original, left transformation and right transformation

- Bit plane slicing

Pixel values are integers composed of bits. Instead of highlighting intensity-level ranges, as above, we could highlight the contribution made to total image appearance by specific bits:



An 8-bit image may be considered as being composed of eight one-bit planes, with plane 1 containing the lowest-order bit of all pixels in the image, and plane 8, all the highest-order bits.

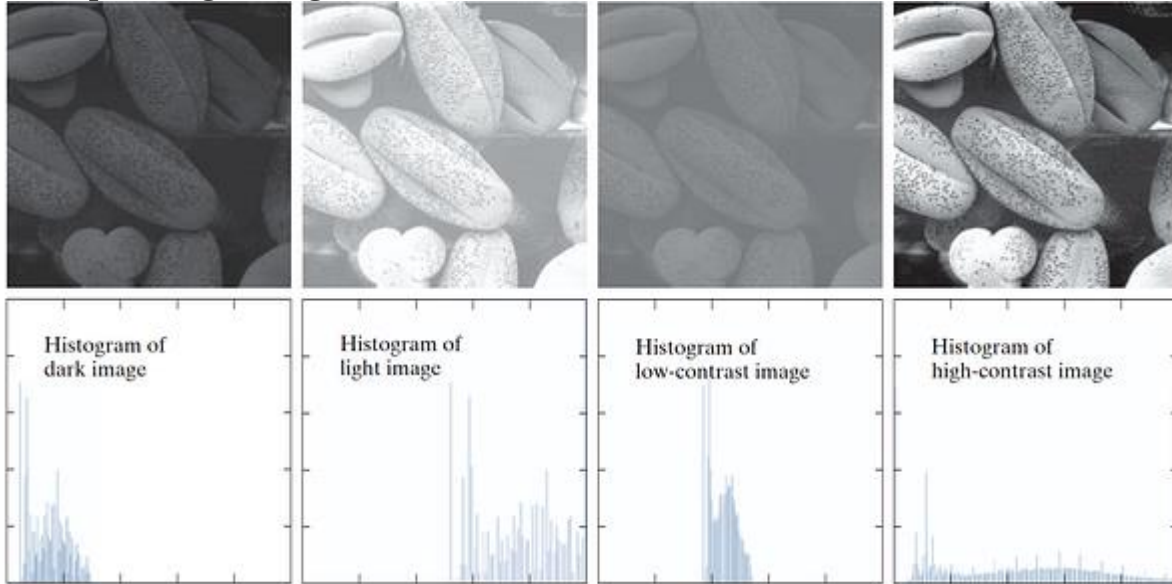


Original image and bit-plane slicing of planes 1–8

In the above figure, observe that the 4 higher-order bit planes, especially the first 2, contain a significant amount of the visually-significant data. The lower-order planes contribute to more subtle intensity details in the image. Also, The original image has a gray border. Notice that the corresponding borders of some of the bit planes are black (0), while others are white (1).

2. Histogram Processing

We can use a histogram to denote the intensities of an L-level digital image $f(x,y)$. Histogram shape is related to image appearance. The following figure shows images with four basic intensity characteristics: dark, light, low contrast, and high contrast and their corresponding histograms:



Intuitively, we can conclude that an image whose pixels tend to occupy the entire range of possible intensity levels, distributed uniformly, will have an appearance of high contrast and will exhibit a large variety of gray tones.

2.1. Histogram Equalization

let the variable r denote the intensities of an image in the range $[0, L-1]$ with $r = 0$ representing black and $r=L-1$ representing white. Let $p_r(r)$ and $p_s(s)$ denote the PDFs of intensity values r and s in two different images (p_r and p_s are different functions). Then the PDF of the transformed (mapped) variable s can be obtained as

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

Thus, we see that the PDF of the output intensity variable, s , is determined by the PDF of the input intensities and the transformation function used. Now we define a transformation function of particular importance in image processing:

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

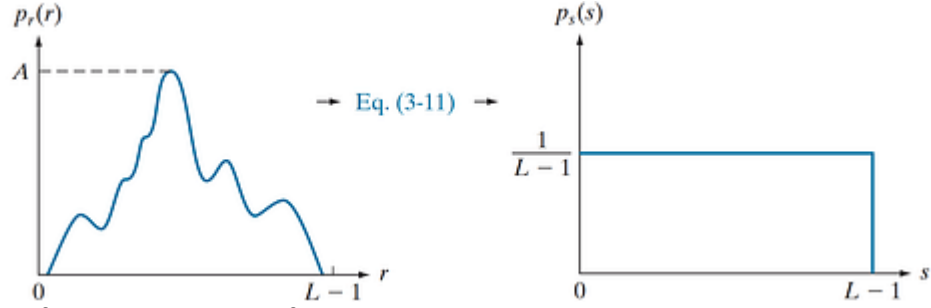
The integral on the right side is the **cumulative distribution function (CDF)** of random variable r . Now we can use the previous equation to find $p_s(s)$:

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

Substituting this result for dr/ds , and noting that all probability values are positive, gives the result:

$$\begin{aligned} p_s(s) &= p_r(r) \left| \frac{dr}{ds} \right| \\ &= p_r(r) \left| \frac{1}{(L - 1) p_r(r)} \right| \\ &= \frac{1}{L - 1} \quad 0 \leq s \leq L - 1 \end{aligned}$$

The form of $p_s(s)$ in the last line of this equation is a uniform probability density function. Thus, performing the intensity transformation yields a random variable, s , characterized by a uniform PDF. What is important is, $p_s(s)$ will always be uniform, independent of the form of $p_r(r)$.



Arbitrary PDF and result of applying the transformation

For discrete values, we work with probabilities and summations instead of probability density functions and integrals. The probability of occurrence of intensity level r_k in a digital image is approximated by:

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

where M and N are the number of image rows and columns, respectively, and MN is the total no. of pixels. n_k denotes the number of pixels that have intensity r_k . The discrete form of the transformation is given by:

$$s_k = T(r_k) = (L-1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L-1$$

Thus, a processed (output) image is obtained to map each pixel in the input image with intensity r_k into a corresponding pixel with level s_k in the output image. This is called a histogram equalization or histogram linearization transformation.

2.2. Histogram Matching (Specification)

When automatic enhancement is desired, Histogram Equalization is a good approach to consider because the results are predictable and the method is simple to implement. However, sometimes it is useful to be able to specify the shape of the histogram that we

wish the processed image to have. The method used to generate images that have a specified histogram is called histogram matching or histogram specification.



An image with a specified PDF can be obtained using the following procedure:

- Compute the histogram $p_r(r)$ of the given image, and use it to map the intensities in the input image to the intensities in the histogram-equalized image. Round the resulting values, s_k , to the integer range $[0, L-1]$

$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k p_r(r_j) \quad k = 0, 1, 2, \dots, L - 1$$

- Use the specified PDF $p_z(z)$ to obtain the function $G(z_q)$ for $q=0, 1, \dots, L-1$, where $p_z(z)$ are the values of the specified histogram.

$$G(z_q) = (L - 1) \sum_{i=0}^q p_z(z_i)$$

Round the values of G to integers in the range $[0, L-1]$. Store the rounded values of G in a lookup table

- For every value of s_k ($k=0, 1, \dots, L-1$) use the stored values of G from Step 2 to find the corresponding value of z_q so that $G(z_q)$ is closest to s_k . Store these mappings from s to z .
- Form the histogram-specified image by mapping every equalized pixel with value s_k to the corresponding pixel with value z_q in the histogram-specified image, using the mappings found in Step 3.

2.3. Example: Histogram Equalization

Consider a 3-bit grayscale image with pixel intensity levels ranging from 0 to 7 (i.e., $L = 8$ intensity levels). The original histogram of the image is given by the following table:

Input Histogram Data								
Intensity (r_k)	0	1	2	3	4	5	6	7
Number of pixels (n_k)	0	0	1	2	4	5	3	1

- **Total Number of Pixels (N):**

The total number of pixels in the image is the sum of pixel counts for each intensity level:

$$N = \sum_{k=0}^{L-1} n_k = 0 + 0 + 1 + 2 + 4 + 5 + 3 + 1 = 16$$

- **Probability of Each Intensity Level $p(r_k)$:**

The probability of occurrence of intensity level r_k is the ratio of its pixel count n_k to the total number of pixels N :

$$p(r_k) = \frac{n_k}{N}$$

- **Cumulative Distribution Function (CDF):**

The cumulative probability for intensity level k is the sum of probabilities up to k :

$$CDF(k) = \sum_{j=0}^k p(r_j) = \frac{\sum_{j=0}^k n_j}{N}$$

- **Equalization Transformation $T(r_k)$ and Rounding s_k :**

The new intensity level s_k corresponding to r_k is obtained by multiplying the CDF by $(L - 1)$ and rounding to the nearest integer:

$$s_k = \text{round}((L - 1) \times CDF(k)) = \text{round}\left(7 \times \frac{\sum_{j=0}^k n_j}{16}\right)$$

- **Detailed Step-by-Step Calculations**

- **Level $k = 0$ ($r_0 = 0$):** $n_0 = 0$ | $p(r_0) = 0/16 = 0.0$ | $CDF(0) = 0/16 \rightarrow s_0 = \text{round}(7 \times 0/16) = \text{round}(0) = 0$
- **Level $k = 1$ ($r_1 = 1$):** $n_1 = 0$ | $p(r_1) = 0/16 = 0.0$ | $CDF(1) = 0/16 \rightarrow s_1 = \text{round}(7 \times 0/16) = \text{round}(0) = 0$
- **Level $k = 2$ ($r_2 = 2$):** $n_2 = 1$ | $p(r_2) = 1/16 = 0.0625$ | $CDF(2) = 1/16 \rightarrow s_2 = \text{round}(7 \times 1/16) = \text{round}(0.4375) = 0$
- **Level $k = 3$ ($r_3 = 3$):** $n_3 = 2$ | $p(r_3) = 2/16 = 0.1250$ | $CDF(3) = 3/16 \rightarrow s_3 = \text{round}(7 \times 3/16) = \text{round}(1.3125) = 1$
- **Level $k = 4$ ($r_4 = 4$):** $n_4 = 4$ | $p(r_4) = 4/16 = 0.2500$ | $CDF(4) = 7/16 \rightarrow s_4 = \text{round}(7 \times 7/16) = \text{round}(3.0625) = 3$
- **Level $k = 5$ ($r_5 = 5$):** $n_5 = 5$ | $p(r_5) = 5/16 = 0.3125$ | $CDF(5) = 12/16 \rightarrow s_5 = \text{round}(7 \times 12/16) = \text{round}(5.2500) = 5$
- **Level $k = 6$ ($r_6 = 6$):** $n_6 = 3$ | $p(r_6) = 3/16 = 0.1875$ | $CDF(6) = 15/16 \rightarrow s_6 = \text{round}(7 \times 15/16) = \text{round}(6.5625) = 7$
- **Level $k = 7$ ($r_7 = 7$):** $n_7 = 1$ | $p(r_7) = 1/16 = 0.0625$ | $CDF(7) = 16/16 \rightarrow s_7 = \text{round}(7 \times 16/16) = \text{round}(7.0000) = 7$

- **Calculation Summary Table**

k	Level r_k	Pixels n_k	Probability $p(r_k)$	Cumulative $\sum n_j$	Formula $7 \times CDF$	Equalized Level s_k
0	0	0	0.0000 (0/16)	0	0.0000	0
1	1	0	0.0000 (0/16)	0	0.0000	0
2	2	1	0.0625 (1/16)	1	0.4375	0
3	3	2	0.1250 (2/16)	3	1.3125	1
4	4	4	0.2500 (4/16)	7	3.0625	3
5	5	5	0.3125 (5/16)	12	5.2500	5
6	6	3	0.1875 (3/16)	15	6.5625	7
7	7	1	0.0625 (1/16)	16	7.0000	7

- **Final Equalized Histogram**

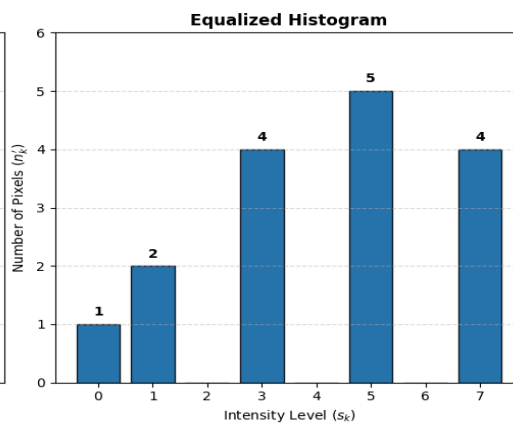
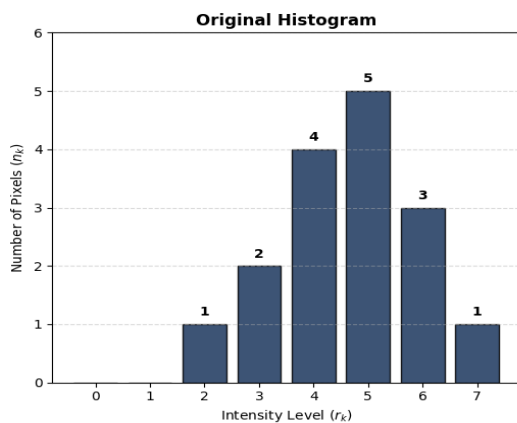
After mapping and combining pixel counts for intensity levels that share the same equalized value s_k , we obtain the new histogram:

- **Intensity 0:** $n_0 + n_1 + n_2 = 0 + 0 + 1 = 1$ pixel
- **Intensity 1:** $n_3 = 2$ pixels
- **Intensities 2, 4, 6:** 0 pixels (no pixels map to these intensities)
- **Intensity 3:** $n_4 = 4$ pixels
- **Intensity 5:** $n_5 = 5$ pixels
- **Intensity 7:** $n_6 + n_7 = 3 + 1 = 4$ pixels

Before	After
$r=0 : 0$ pixel	$s=0 : 1$ pixel
$r=1 : 0$ pixel	
$r=2 : 1$ pixel	
$r=3 : 2$ pixels	$s=1 : 2$ pixels
$r=4 : 4$ pixels	$s=3 : 4$ pixels
$r=5 : 5$ pixels	$s=5 : 5$ pixels
$r=6 : 3$ pixels	$s=7 : 4$ pixels
$r=7 : 1$ pixel	

Summary Equalized Histogram

New Intensity (s_k)	0	1	2	3	4	5	6	7
New Pixel Count (n_k')	1	2	0	4	0	5	0	4



3. Noise Reduction Using Spatial Filtering

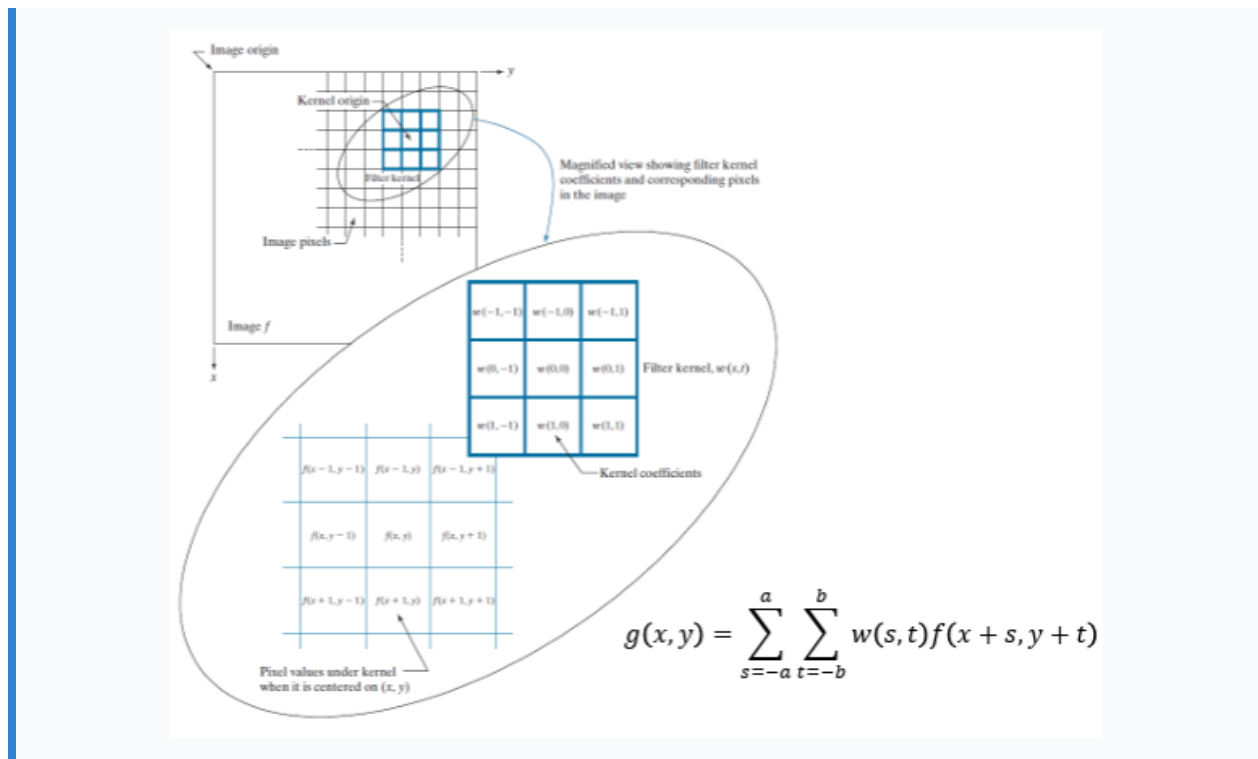
3.1. Origin and Characteristics of Image Noise

Image noise represents unwanted intensity variations introduced during image acquisition (sensor thermal noise, low lighting conditions), digitization, or transmission over noisy communication channels.

3.2. Principle of Spatial Filtering and Masks

Spatial filtering operates directly on pixel values. It involves sliding a small neighborhood window (kernel or mask, e.g., 3×3, 5×5) across the image and calculating a new intensity value for the central pixel.

A 3×3 linear kernel K is represented by a weighting matrix:



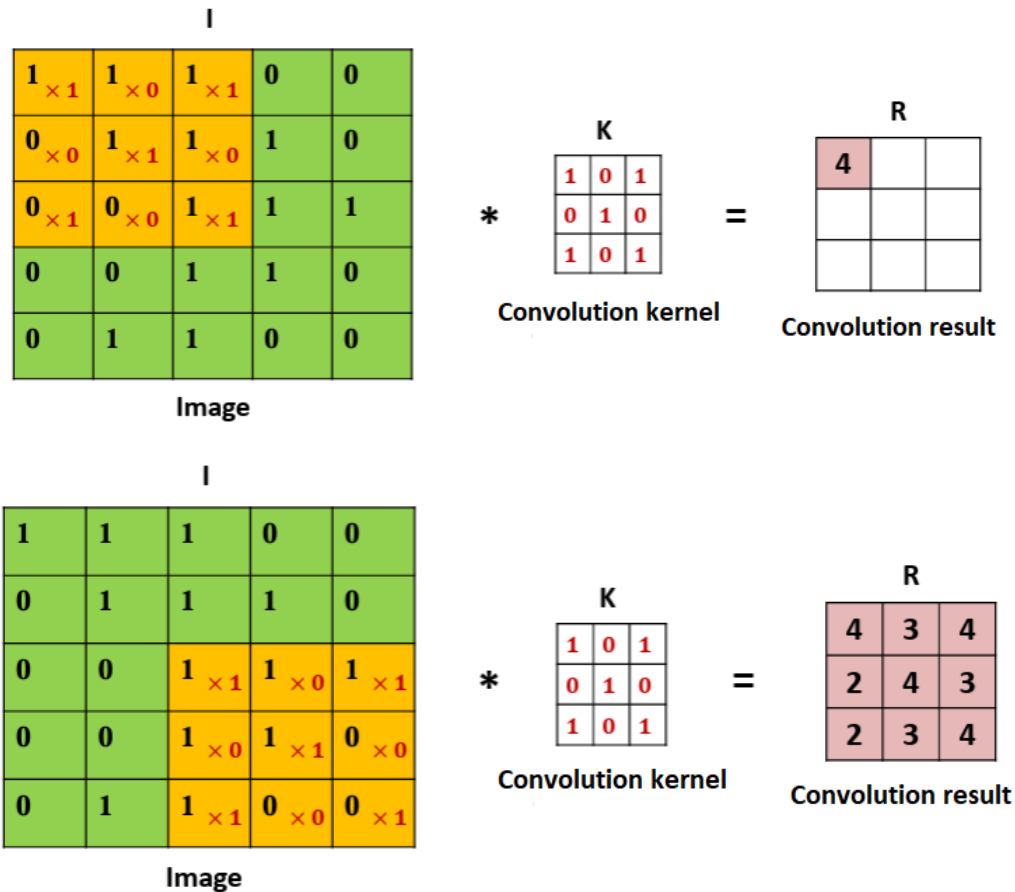
Mechanism of sliding a 3x3 kernel over an image neighborhood

3.3. Digital 2D Convolution

The mathematical formulation of the convolution operation is given by:

$$R(x, y) = (I * K)(x, y) = \sum_{m=-a}^a \sum_{n=-b}^b I(x-m, y-n) K(m, n)$$

where $I(x, y)$ denotes the input image, $K(m, n)$ is the filtering kernel, and $R(x, y)$ is the resulting image. The symbol $*$ represents the convolution operation, while $2a + 1$ and $2b + 1$ correspond to the height and width of the kernel, respectively.



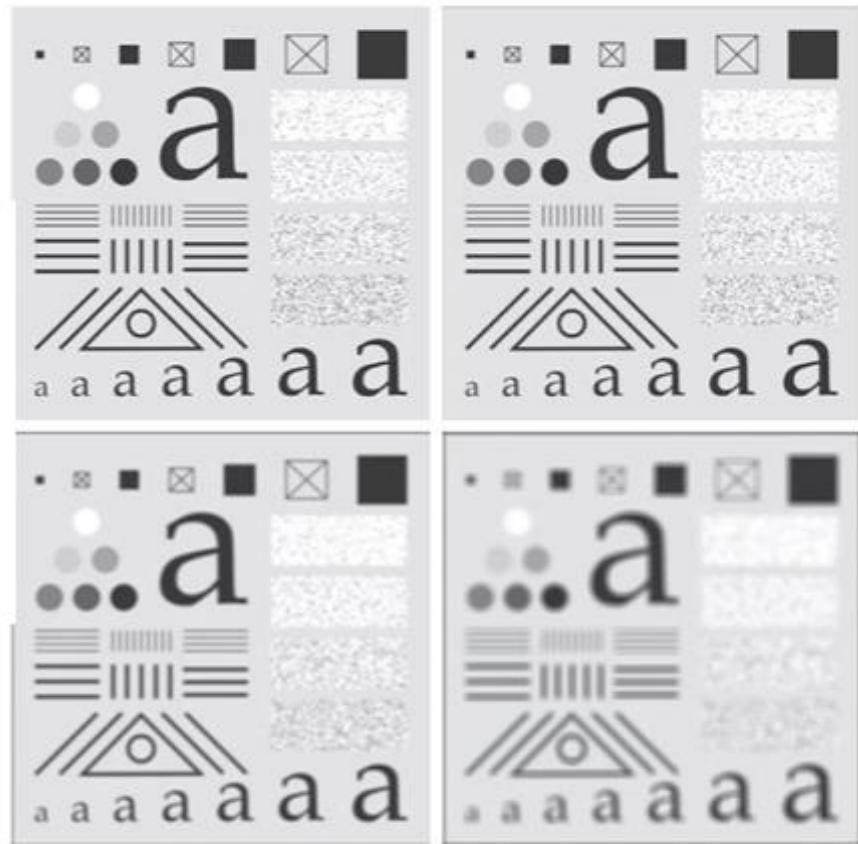
$$\text{Digital convolution } R = I * K$$

3.4. Smoothing (Blurring) Filters

Smoothing filters attenuate local intensity variations and high-frequency noise. The simplest linear smoothing filter is the box (mean) filter:

$$\frac{1}{9} \times \begin{array}{|c|c|c|} \hline \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \hline \end{array}$$

Mean Filter



All neighboring pixels contribute equally to the averaged output intensity.

3.5. Gaussian Filter

The Gaussian filter is an isotropic linear smoothing filter that assigns maximum weight to the central pixel, with weights decaying exponentially as a function of Euclidean distance.

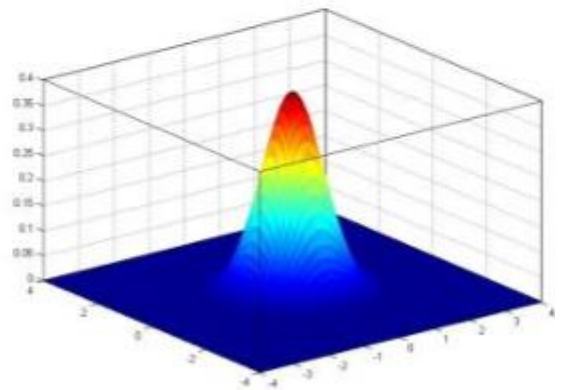
The 2D continuous Gaussian function is defined as:

Filtre gaussien

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Taille du filtre: $(4\sigma + 1) \times (4\sigma + 1)$

0,00	0,01	0,02	0,01	0,00
0,01	0,06	0,10	0,06	0,01
0,02	0,10	0,16	0,10	0,02
0,01	0,06	0,10	0,06	0,01
0,00	0,01	0,02	0,01	0,00



Exemple de filtre Gaussien $\sigma=1$

where σ (standard deviation) controls the spatial spread of the Gaussian bell curve and the degree of smoothing.

3.6. Median Filter (Non-Linear Filtering)

The median filter is a order-statistic non-linear filter. It replaces the central pixel value with the statistical median of the sorted intensity values in its neighborhood.

Algorithm for a 3×3 Neighborhood:

Example :

1 2 4 5 6 7 10 3 2 5 9 8 1 2 4 3 5 6 4 3 6

There are **21 values**, so the median is the **11th value** after sorting.

Sorted values: 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, **4**, 5, 5, 5, 6, 6, 6, 7, 8, 9, 10

So, **the median of all 21 values is 4.**

4. Contrast Enhancement and Sharpening

4.1. Introduction and Objectives

Image sharpening aims to highlight fine details, edges, and spatial intensity transitions that have been blurred during acquisition or filtering. While blurring corresponds to spatial integration (high-frequency attenuation), sharpening relies on spatial differentiation (high-frequency emphasis).

4.2. Image Derivatives

The first- and second-order derivatives of a one-dimensional function $f(x)$ can be approximated using the following formulas, respectively:

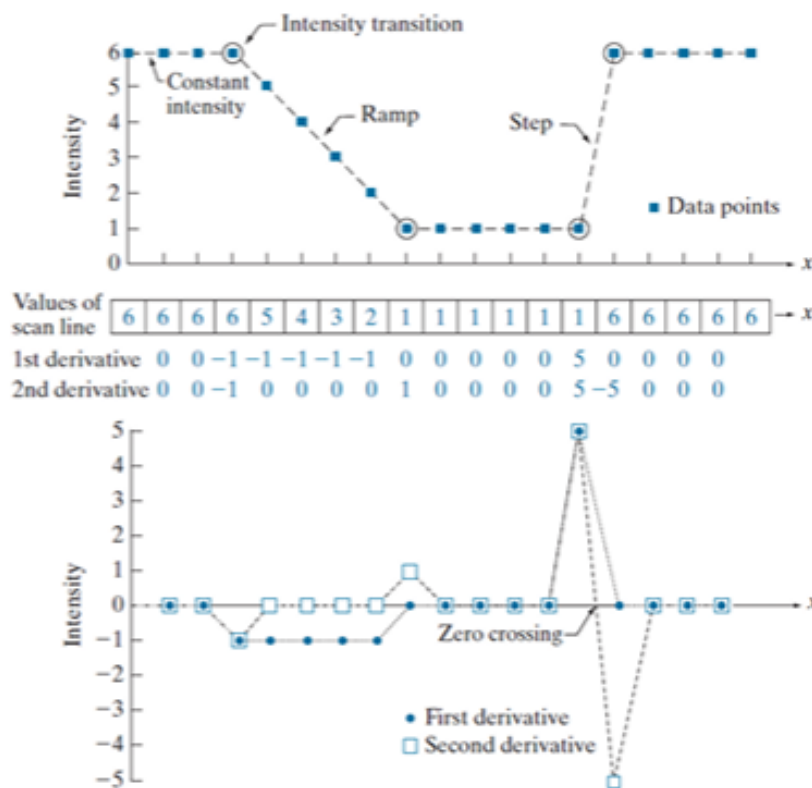
$$f'(x) = f(x+1) - f(x)$$

$$f''(x) = f(x+1) - 2f(x) + f(x-1)$$

a
b
c

FIGURE 3.44

(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.

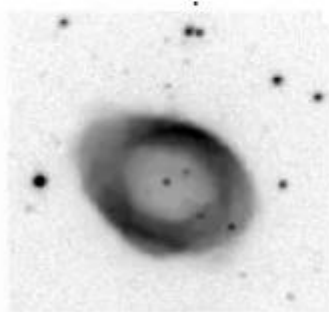


The **Laplacian filter** is a particular convolution filter used to emphasize image details characterized by rapid changes in intensity. It is therefore particularly effective for highlighting the edges and contours of objects, which explains its use in pattern recognition and, historically, in both military and civilian applications. From a mathematical point of view, the Laplacian is a **second-order derivative operator in two dimensions**. It is defined as follows:

$$L(x, y) = \frac{\partial^2 I(x, y)}{\partial x^2} + \frac{\partial^2 I(x, y)}{\partial y^2}$$

In image processing, the original image $I(x,y)$ is not a continuous function but a discrete function due to the digitization process. Nevertheless, the second derivative can still be approximated accurately. Three typical 3×3 kernels can be used for this purpose:

0	1	0	1	1	1	0	-1	0	-1	-1	-1
1	-4	1	1	-8	1	-1	4	-1	-1	8	-1
0	1	0	1	1	1	0	-1	0	-1	-1	-1



*

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

=



A detail worth noting is that the sum of all the elements of a Laplacian filter kernel is always zero. This implies that the filter does not change the average intensity of a constant image and is therefore commonly used to emphasize regions of rapid intensity variation, such as edges and fine details.