MANIPAL
ACADEMY of HIGHER EDUCATION
(Institution of Eminence Deemed to be University)

MDS6401 / Segment 01

IMAGE REPRESENTATION AND ENHANCEMENT – PART I

MANIPAL ACADEMY of HIGHER EDUCATION (Institution of Eminence Deemed to be University)

Table of Contents

Introduction

Image enhancement is a critical aspect of digital image processing that aims to improve image quality and visual appeal. This field involves various techniques designed to enhance the interpretability and presentation of images for applications in medical imaging, remote sensing, consumer photography, and more. The primary goals of image enhancement include amplifying important details, reducing noise, and improving the visual appeal of images. This topic focuses on the detailed concepts of image enhancement. It will cover fundamental concepts, key objectives, techniques used, and practical examples to illustrate the impact and necessity of these processes in enhancing digital images across different applications.

Learning Objectives

At the end of this topic, you will be able to:

- Explain the basic concepts of image enhancement
- Elaborate on the spatial and frequency domains in image processing
- Explain the various histogram processing and matching techniques
- Outline the concepts of multiple filters

1. Concepts of Image Enhancement

Image enhancement refers to the collection of techniques employed to improve images' utility and aesthetic appeal. This process is crucial in various fields, including medical imaging, remote sensing, and consumer photography, to enhance the interpretability or presentation of digital images. The primary goals of image enhancement include:

- **Highlighting Interesting Details:** This involves amplifying specific features within an image to make them more prominent or easier to perceive. This is particularly valuable in scientific imaging, where essential but subtle details must be emphasised.
- **Removing Noise:** Digital images often contain unwanted information or distortions known as noise. Image enhancement helps clean up these images by reducing or eliminating noise and improving image clarity and quality.
- **Improving Visual Appeal:** For images in advertising, media, or personal use, enhancing an image to make it more visually appealing is crucial. This could involve adjusting colours, contrast, brightness, and other visual elements to make images more engaging and pleasant.

1.1 Objectives

The primary objective of image enhancement is to render images more 'lookable' or visually appealing, similar and more suitable for further analysis or presentation. This involves:

- Detail Amplification: Bringing out subtle details in images that may be overlooked or obscured.
- Noise Reduction: Identifying and eliminating unwanted artefacts that can degrade the image. These artefacts, often called 'noise', can manifest as random dots or irregular features that do not reflect the actual content of the scene.
- Contrast Enhancement: Adjusting the image's contrast to make the distribution of tones (from dark to light) more distinct, which helps distinguish features and structures within the image.

1.2 Techniques

- **Pre-processing and Post-processing:** Images captured from cameras undergo several behind-the-scenes adjustments before they reach the viewer. These adjustments may include scaling, cropping, and colour correction to enhance the image's overall quality.
- **Morphological Operations:** These techniques process image structures, enhancing the shapes and edges within the image to make features more distinguishable.
- **Artefacts Removal:** Artifacts that appear randomly and are not part of the original scene are removed to improve the clarity and accuracy of the visual representation.

1.3 Examples

- **Bone Images:** In medical imaging, enhancing the contrast and clarity of bone images can significantly affect diagnosis. An improved image provides a clearer distinction between different tissue densities, making it easier to identify abnormalities.

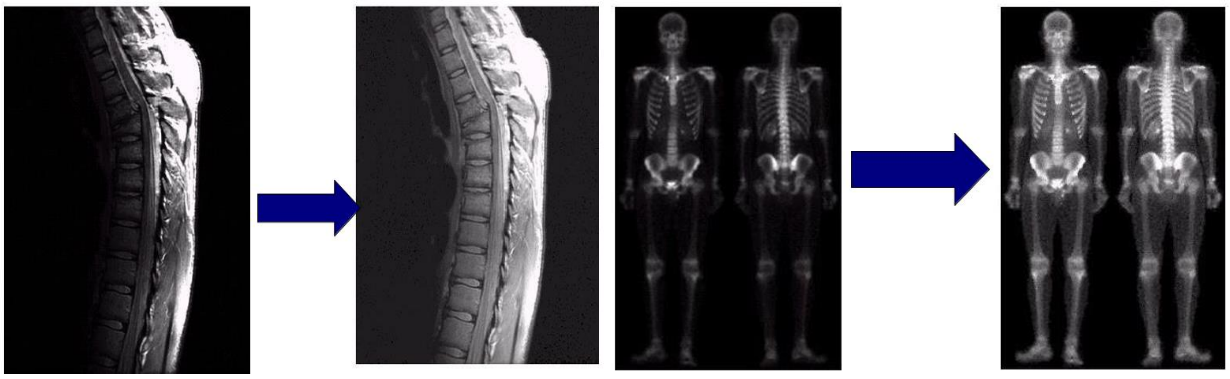


Figure 1: Bone Image Enhancement

- **Chip Images:** In technical fields, images of electronic components like chips may contain noise that obscures critical details. Applying filters can reduce this noise, resulting in cleaner and more precise visual data.

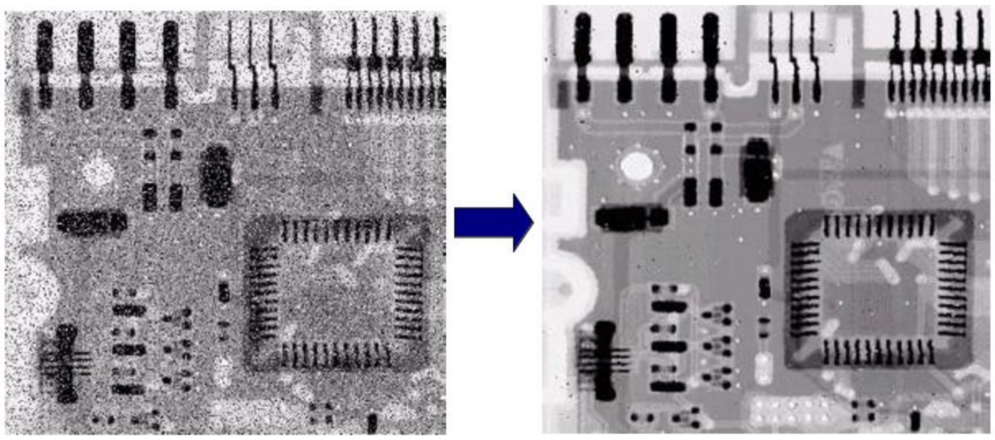


Figure 2: Chip image

- **Aerial City Images:** Enhancing aerial photographs of cities can help better interpret urban layouts. Adjustments to brightness and contrast can reveal details not visible in the original image.

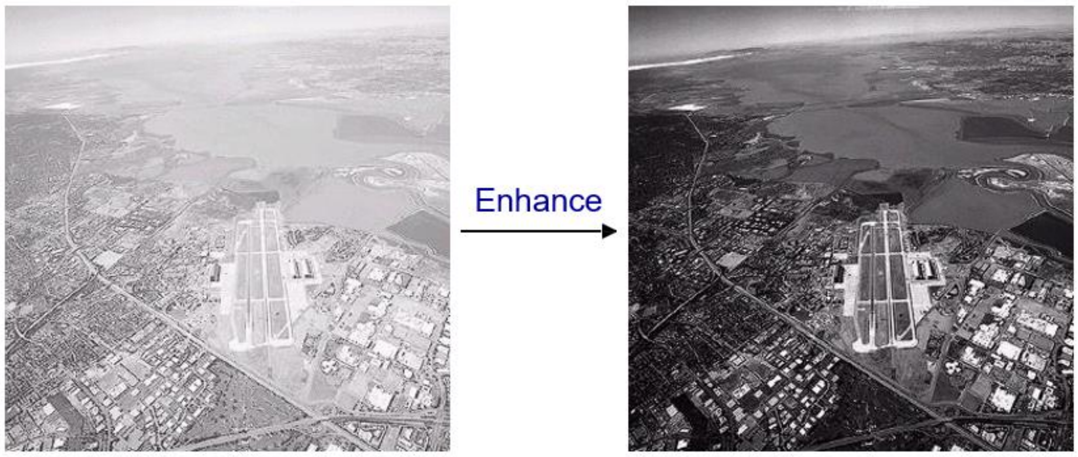


Figure 3: Aerial City

2. Spatial and Frequency Domains

In digital image processing, manipulation can occur within two primary domains: the spatial and frequency domains. The spatial domain directly manipulates the pixels within an image, often utilising information from neighbouring pixels or the overall image composition. Standard techniques include point processing, where individual pixels are adjusted, and neighbourhood processing, which involves considering the surrounding pixels (the neighbours) for operations like spatial filtering. This is similar to enhancing or diminishing certain features based on their surrounding context, effectively altering the image's visual presentation at the pixel level.

Conversely, the frequency domain transforms the image into a frequency representation, usually through mathematical transformations such as the Fourier Transform. In this domain, the image is analysed and manipulated based on the frequency of the pixel intensity variations rather than their spatial arrangement. This approach is beneficial for identifying and enhancing specific frequency components of an image, which can be crucial for removing noise or emphasising certain textures or shapes within the image.

2.1 Histogram Processing

Histogram processing, a spatial domain technique, adjusts the image's contrast and brightness by modifying the pixel intensity distribution. This is achieved by mapping the frequencies of each intensity level within the image, often visualised through a histogram. Histograms represent how many pixels in an image have specific intensity values, which can be crucial for tasks like enhancing underexposed or overexposed images. Adjusting the histogram can significantly affect the visual quality of the image. Similarly, it is more suitable for further analysing or improving its aesthetic appeal.

Let us focus on histogram usage and how pixel values are distributed across different images.

Example 1:

0	4	8	10	12
12	16	5	0	16
4	16	8	5	10
10	0	4	12	16
12	0	16	0	8

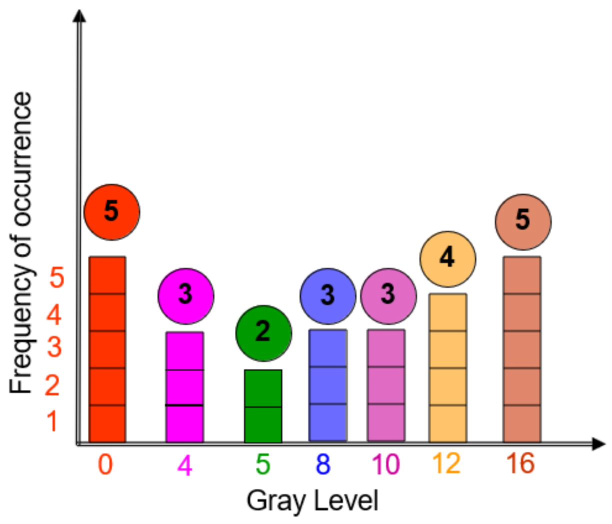


Figure 4: Matrix with Grey Level Histogram

Figure 4 shows a matrix with discrete pixel values alongside a histogram representing the frequency of occurrence for each grey level in the matrix. The histogram visually represents how often each grey level appears, which helps understand the image's contrast and brightness distribution.

Example 2:

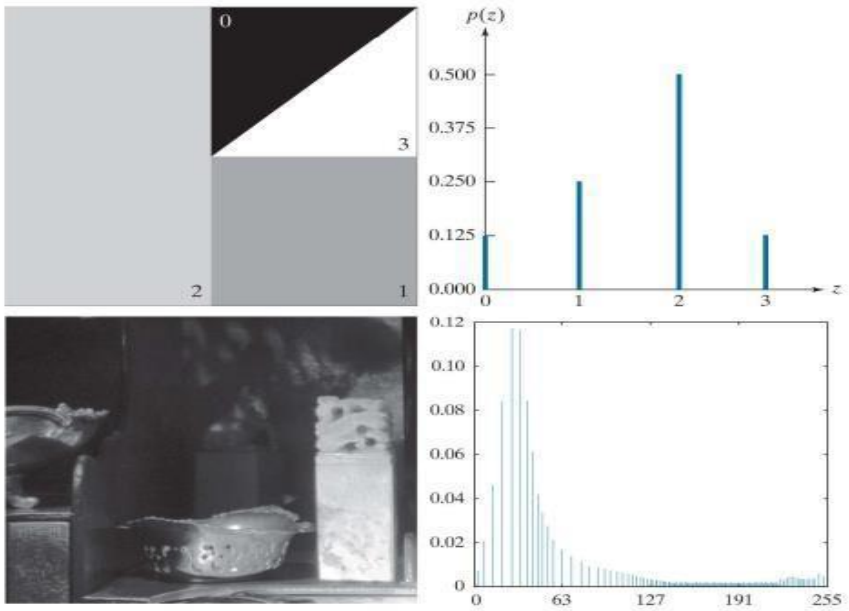


Figure 5: Underexposure and Overexposure

Figure 5 shows underexposed, correctly exposed, and overexposed scenes, typically including histograms to indicate pixel intensities' distribution. An underexposed image histogram would skew towards darker values (left side of the histogram), a correctly exposed image would have a more balanced histogram, and an overexposed image would skew towards brighter values (right side of the histogram).

Example 3:

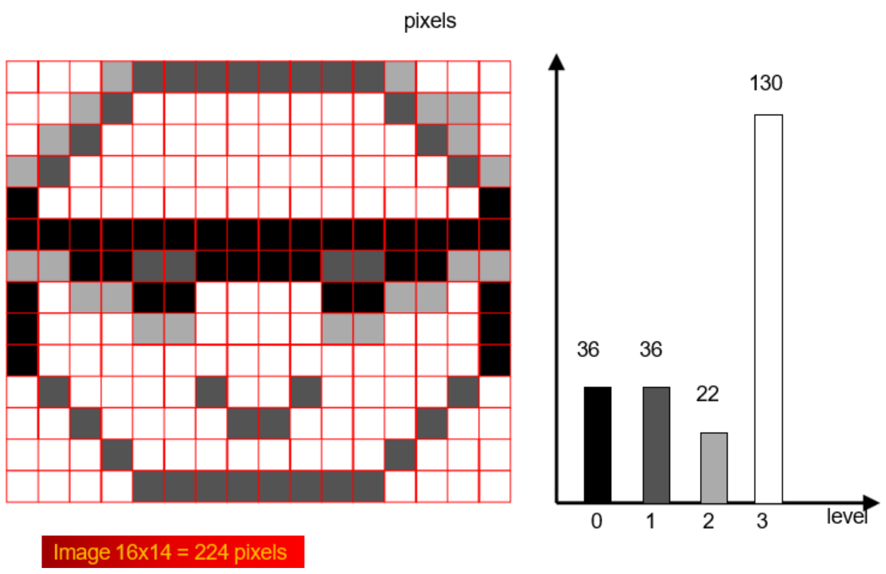


Figure 6: Pixel Grids and Associated Histograms

Figure 6 shows a grid of pixels, typically magnified to show individual pixel values, accompanied by histograms that count the occurrence of each pixel value. These are useful for analysing the texture and detail at a very localised level within an image.

Example 4:

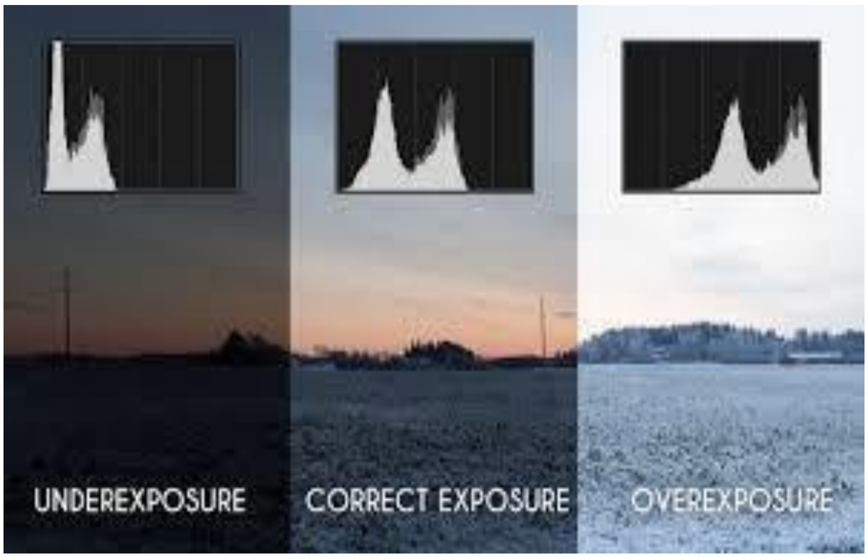


Figure 7: Comparative Analysis of Exposure Levels

Figure 7 demonstrates three exposure levels of a scene, each with its histogram. The first image shows underexposure, with most pixels concentrated towards the dark end, resulting in a dark scene with lost details. The middle image is correctly exposed with a balanced distribution of tones, capturing the scene accurately. The third image is overexposed, with pixel values skewed towards brightness, causing a washed-out appearance and loss of detail in highlights. Histograms provide a visual tool for assessing and adjusting exposure in images.

Example 5:

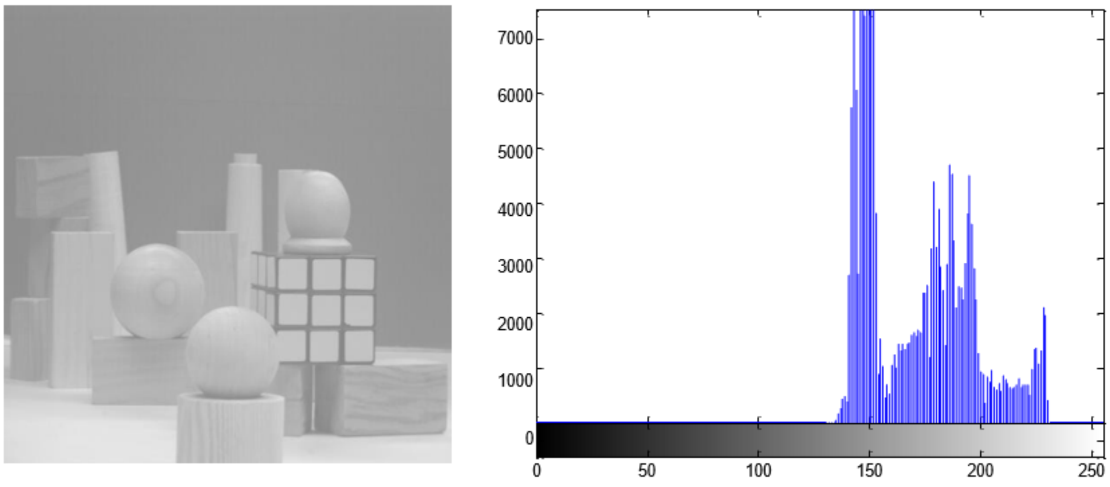


Figure 8: Histogram of Overexposed Image

Figure 8 on the left shows a greyscale photo of geometric objects. The histogram on the right represents the distribution of pixel intensity values across the image, measured from 0 (black) to 255 (white). The histogram exhibits two significant peaks: a high peak around the mid-range and another substantial presence in the higher range, indicating a concentration of mid-tone and lighter pixels, respectively, corresponding to the grey and lighter areas visible in the image.

Example 6:

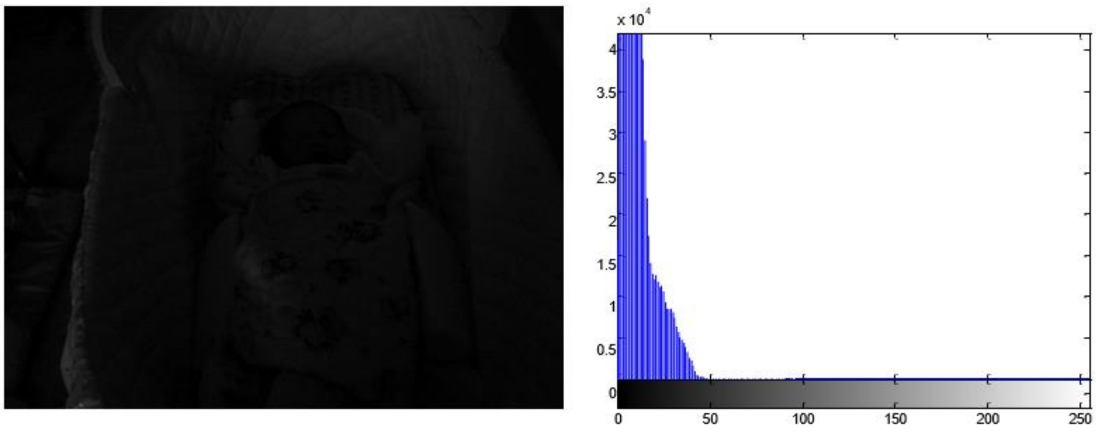


Figure 9: Histogram of Underexposed Image

Figure 9 on the left is a dark, low-light photo of a baby in a crib, predominantly showing darker shades. The histogram on the right illustrates the distribution of pixel intensity values for this image. Most of the bars in the histogram are heavily skewed towards the left, indicating a high concentration of darker pixels. The peak near the zero mark on the histogram suggests that most of the image's pixels are very close to black, corroborating the underexposed nature of the photograph. This visualisation aids in understanding the need for potential brightness enhancement or exposure correction to bring out more details in the image.

Image analysis using histograms can be performed using Python for image processing tasks; the OpenCV library is an essential tool. Begin by installing the library via the command line with pip install opencv-python. After installation, import it into the Python script using import cv2. Specifically, the cv2.calcHist() function is instrumental for calculating image histograms. The imhist() function serves a similar purpose for MATLAB users, allowing for straightforward histogram generation of images.

2.1.1 Histogram Equalisation

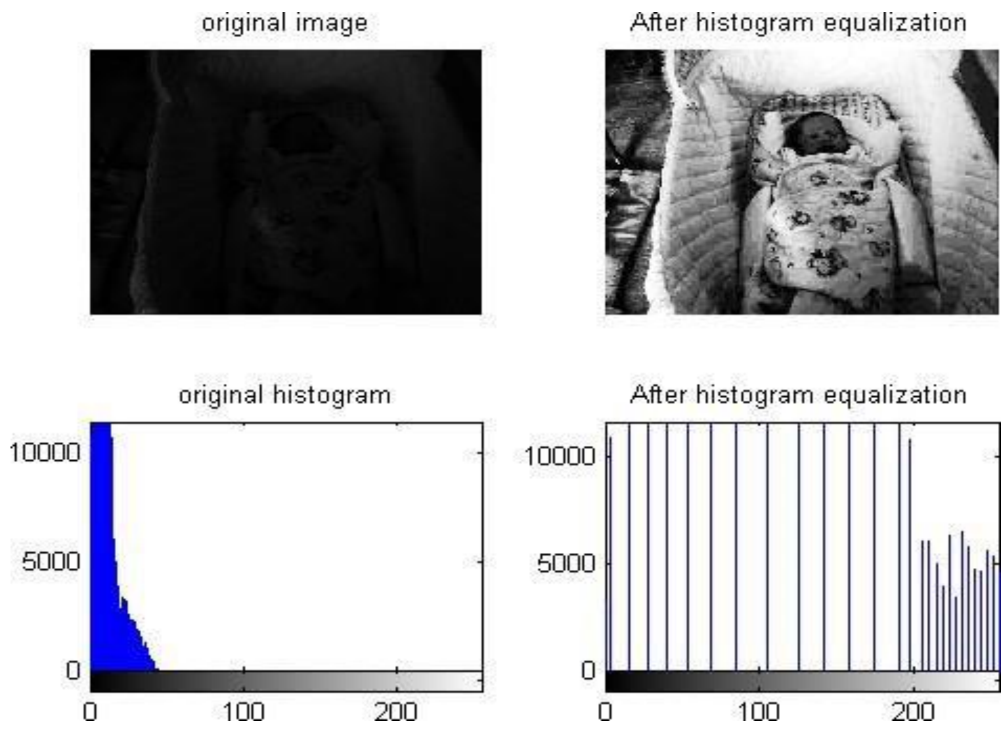


Figure 10: Histogram Equalisation

The provided Figure 10 illustrates the impact of histogram equalisation on an underexposed image. The original image, predominantly dark due to abundant black pixels, does not reveal the underlying objects. Histogram equalisation, a process designed to improve image visibility, has been applied to adjust the pixel intensity distribution. This technique redistributes pixel values across the available range to utilise the full spectrum more effectively.

The original histogram, skewed towards the lower end, confirms the underexposure, with most pixel values clustered near zero. Post-equalisation, the histogram shows a uniform distribution across the spectrum from 0 to 255, demonstrating a successful enhancement where pixel intensities are balanced, enhancing both contrast and detail visibility. This process is crucial in medical imaging and photography, where details in darker areas are critical for accurate interpretation.

A histogram in the context of an image represents the frequency distribution of its pixels' intensity values. This distribution is fundamental to many image processing techniques, including histogram equalisation. Here's how it works:

$$h(r_k) = n_k$$

where,

r_k is the k^{th} intensity value

n_k is the number of pixels in the image with intensity r_k

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

- Intensity Levels (r): In an image, each pixel carries an intensity value, typically ranging from 0 (black) to 255 (white) in an 8-bit image. Each distinct intensity level is denoted by 'r'.
- Frequency of Intensity (n_k): The ' n_k ' value represents the number of times a particular intensity appears in the image. For instance, if '0' (black) occurs 780 times, then for intensity level 'r=0', ' n_k=780'.
- Probability Density Function (PDF): To normalise the histogram, we use a PDF, which adjusts each n_k value. This is calculated by dividing the n_k value by the number of pixels in the image (MxN, where M and N are the image's dimensions). This

normalisation helps adjust the range of the histogram to [0,1], similar to the range normalisation in statistics.

Let us consider Figure 11, which illustrates the transformation of a Probability Mass Function (PMF) into a uniform distribution through histogram equalisation, a technique employed in image processing to enhance contrast.

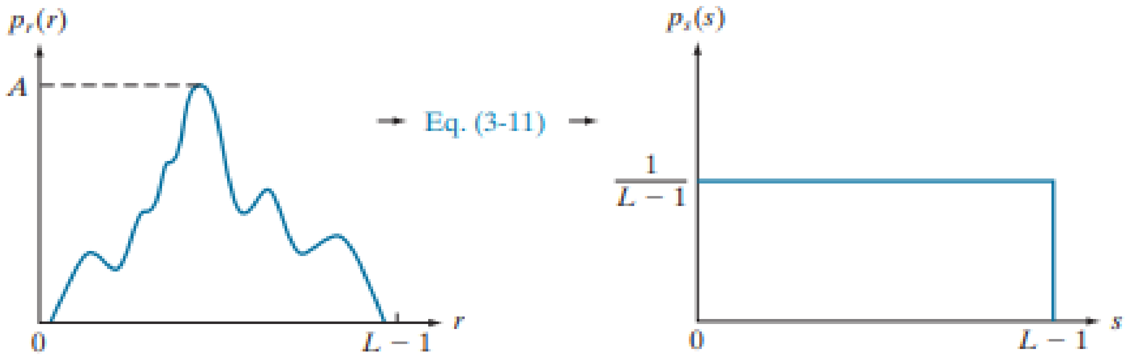


Figure 11: Transformation from Non-Uniform to Uniform Probability Distribution via Histogram Equalisation
(Source: Mathematical Derivation: Rafael C. Gonzalez, Richard E. Woods, Digital Image Processing, (4e), Pearson Education, 2018. (Chapter-3, Page-134))

- Left Graph (p_r (r)): This graph represents the PMF of pixel intensities in an image. Here, 'r' represents pixel intensities ranging from 0 to L-1, where L is the maximum intensity level (often 255 for an 8-bit image). The y-axis measures the probability of each intensity occurring within the image. The shape of the curve indicates the distribution of these intensities, showing peaks where specific intensities occur more frequently.
- Transformation (Eq. 3-11): This equation symbolises the mathematical operation applied to the PMF to achieve histogram equalisation. The transformation aims to modify the intensity levels to distribute more uniformly across the available range, effectively improving the image's contrast.
- Right Graph (p_s (s)): After applying the histogram equalisation process, the PMF is transformed into a nearly uniform distribution across all intensity levels from 0 to L-1. The y-axis shows that each intensity level is equally likely, which is ideal for

maximising the dynamic range and enhancing visual details in areas previously dominated by shadows or highlights.

Intensity Transformation:

The histogram equalisation process begins with intensity mapping, where pixel values are adjusted to enhance image contrast. The image intensities, denoted as r within a range of [0, L-1] where 0 represents black and L−1 represents white, undergo transformation using a function $T(r)$ to yield new intensity values s. This function ensures that s remains within the same bounds as r.

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

The transformation function $T(r)$ must be monotonic, ensuring that it never decreases as r increases. This mapping facilitates a more uniform distribution of intensity values, similar to the image, and is easier to analyse by enhancing visual contrast and details. The transformation is represented graphically where the initial intensity distribution $P_r(r)$ can be uneven or skewed and is transformed into a more uniformly distributed $P_s\ (s)$, essential in applications requiring precise image analysis such as medical imaging or digital photo enhancement.

This function ideally satisfies three fundamental properties:

- Monotonic Increase: The function must consistently rise without decline over the interval [0, L-1].
  T (r) is a monotonic increasing function in the interval $0 \le r \le L - 1$
- Range Preservation: Both the original and transformed intensities span the full range from 0 to L−1.
  $0 \le T(r) \le L - 1$ for $0 \le r \le L - 1$
- Reversibility: The transformation is invertible, meaning there exists a function $T^{-1}\ (s)$ that can revert s back to r, preserving the ability to recover the original intensity values.
  r=T−1 (s)

The discrete form of the transformation is,

$$S_k = T(r_k) = (L-1)_{j=0}^{\sigma k} p_r(r_j) \quad where\ k = 0,1,2,3..L-1$$

In this context, L represents the total number of possible intensity levels in the image, such as 256 for an 8-bit image. The transformation function $T(r_k)$ calculates a new pixel intensity S_k for each original intensity r_k by cumulatively summing the probabilities $p_r(r_j)$ up to the current intensity level k.

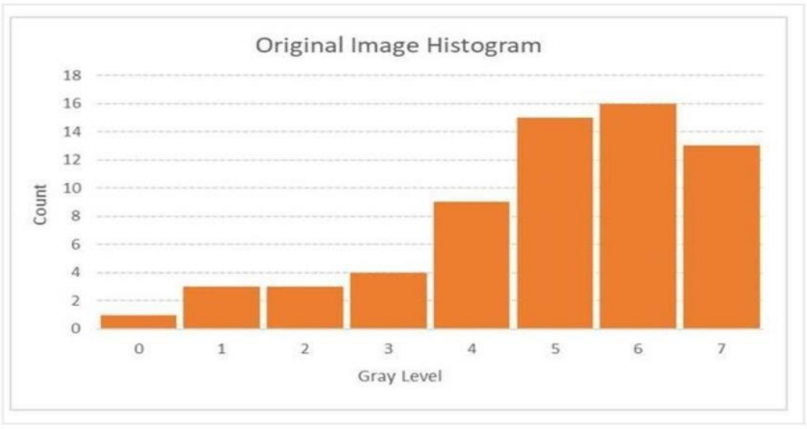
This cumulative sum effectively redistributes the pixel intensities across the available range, aiming to achieve a uniform distribution across all intensities. The factor $(L–1)$ scales the cumulative probability to span the entire intensity range, ensuring that the transformed values S_k also lie within the range from 0 to L–1. This transformation adjusts the image's contrast by spreading the most frequently occurring intensity values. This is particularly useful in images where specific intensities dominate, making it difficult to discern details. By equalising the histogram, lower contrast regions gain higher visibility, enhancing the overall visual perception of the image.

Histogram Equalisation Examples

Example 1:

Let us suppose we have a 3-bit, 8 x 8 greyscale image. The greyscale range is 2^3 = 8 intensity values (i.e. grey levels) because the image is 3 bits. We label these intensity values 0 through 7. Below is the histogram of this image.

Gray Level (i.e. Intensity)	Count
0	1
1	3
2	3
3	4
4	9
5	15
6	16
7	13
	64



Explanation:

Consider a 3-bit greyscale image with dimensions 8 x 8. In this context, a "3-bit" designation implies that each pixel can represent one of 2^3 or 8 possible intensity levels, ranging from 0 to 7. This discrete range allows the image to display varying shades of grey, though limited by the bit depth.

For each intensity level, the distribution might be as follows: an intensity level of 0 occurs once, level 2 occurs three times, level 5 occurs fifteen times, and level 7 occurs thirteen times. This distribution suggests a skew towards higher intensity values, with more occurrences of the darkest (0) and the brightest (7) shades, which are less frequent.

Let us solve using the below tabular column approach

Table 1: Transformation of Greyscale Values Through a Histogram Equalisation Process

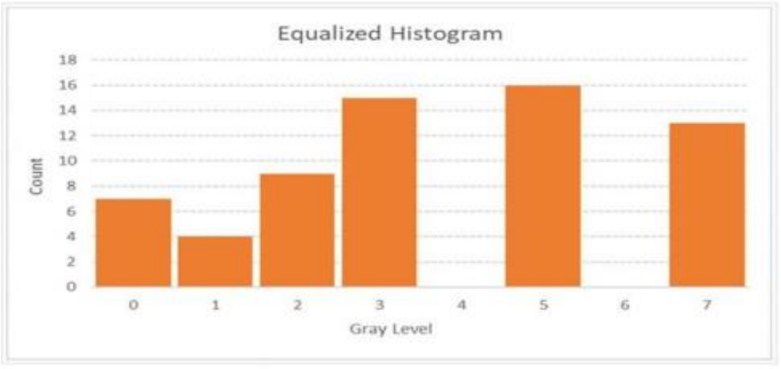
Grey Level (K = 8 total levels)	Count	Probability Density Function (Count/Total Count)	Cumulative Distribution Function (CDF)	(K - 1) * CDF	FLOOR((K - 1) * CDF)
0	1	1/64 = 0.0156 $\longrightarrow$	0.0156	0.109	0
1	3	3/64 = 0.0469	(0.0156 + 0.0469) = 0.0625	0.438	0
2	3	3/64 = 0.0469	(0.0625 + 0.0469) = 0.1094	0.766	0
3	4	4/64 = 0.0625	(0.1094 + 0.0625)= 0.1719	1.203	1
4	9	9/64 = 0.1406	(0.1719 + 0.1406) = 0.3125	2.188	2
5	15	15/64 = 0.2344	(0.3125 + 0.2344) = 0.5469	3.828	3
6	16	16/64 = 0.25	(0.5469 + 0.25) = 0.7969	5.578	5
7	13	13/64 = 0.2031	(0.7969 + 0.2031) = 1.0000	7	7
	64	**1.000**			

Table 1 primarily details the greyscale levels ranging from zero to seven, representing the pixel intensity levels in an 8-level greyscale image. The 'Count' column records the frequency of each pixel level's occurrence within the image. Subsequently, the probability density function (PDF) is computed by dividing each count by the total pixel count, in this case, 64. This results in values such as 1/64 for a single occurrence.

The cumulative distribution function (CDF), crucial for understanding the probability distribution across the image, is calculated by summing the PDF values progressively. For example, the initial CDF value remains 0.0156, with subsequent values aggregating the preceding PDFs, ultimately summing up to 1, as expected in a probability distribution.

Further, the transformation (K-1) * CDF is applied where K represents the total number of greyscale levels. This transformation scales the CDF to span the greyscale range, resulting in values that are then floored. Flooring, a rounding operation, maps these continuous values to discrete greyscale levels. This operation can be executed in programming contexts using functions such as the floor in MATLAB or equivalent functions in Python's NumPy library. Below is the histogram representation of the updated grey level count values:

Gray Level	Count
0	7
1	4
2	9
3	15
4	0
5	16
6	0
7	13
	64



Example 2:

Suppose a 3-bit image (L=8) of size 64 × 64 pixels (MN = 4096) has the intensity distribution shown in the following table.

Get the histogram equalisation transformation function and give the $p_s(s_k)$ for each s_k.

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

Explanation:

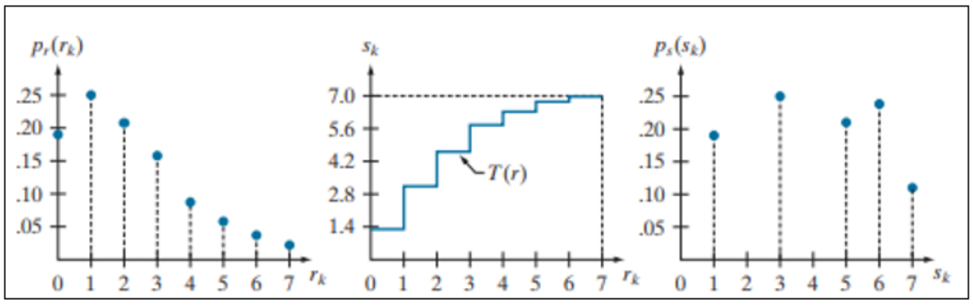
The given table displays pixel intensities (r_k) ranging from 0 to 7 with their corresponding frequencies (n_k) and probabilities ($p(r_k) = \frac{n_k}{MN}$), where MN is the total number of pixels.

The transformation S_k = T(r_k) applies the formula S_k = $(L-1)\sum_{j=0}^{k} p(r_j)$, where L is the number of possible intensity levels (8 for a 3-bit image). This function cumulatively distributes the pixel intensities to achieve a uniform histogram.

Initial values S_0 to S_7 are calculated based on cumulative probabilities, showing incremental increases to ensure each intensity level is mapped proportionally to its cumulative probability.

$$s_0 = T(r_0) = 7 * 0.19 = 1.35 \qquad s_0 = 1$$
$$s_1 = T(r_1) = 7 * (0.19 + 0.25) = 3.08 \qquad s_1 = 3$$
$$s_2 = T(r_2) = 7 * (0.19 + 0.25 + 0.21) = 4.55 \qquad s_2 = 5$$
$$s_3 = 5.67 \qquad s_3 = 6$$
$$s_4 = 6.23 \qquad s_4 = 6$$
$$s_5 = 6.65 \qquad s_5 = 7$$
$$s_6 = 6.86 \qquad s_6 = 7$$
$$s_7 = 7 \qquad s_7 = 7$$

There is a gradual increase in the values from S_4 to S_7. Once the values are calculated, they must be rounded to the nearest value. By plotting the values, we get the three graphs below.



Left Graph ($p(r_k)$):

- This graph shows the probability density function (PDF) for the image's original pixel intensities (rk). Each point represents the probability of occurrence for each intensity level from 0 to 7.
- The vertical axis denotes probability values, while the horizontal axis lists the greyscale levels.

Middle Graph (S_k Vs $T(r)$):

- This graph illustrates the transformation T(r) applied to each pixel intensity r_k. The step-like function depicts how each original pixel value is mapped to a new value S_k to achieve a more uniform histogram.
- The transformation is designed to equalise the histogram, spreading the pixel intensities more uniformly across the available range, as indicated by the graph's steps.

Right Graph ($p(S_k)$):

- This graph displays the PDF of the transformed pixel intensities (S_k). After applying T(r), the histogram is expected to be more uniform, as shown by the relatively equal heights of the bars across all intensity levels.
- This equalisation effect enhances the global contrast of the image, making the details in lower and higher-intensity areas more visible.

Key Observations:

- The transformation function $T(r)$ effectively redistributes the pixel intensities to combat areas of the image that may be underrepresented.

- The resulting histogram ($p(S_k)$) demonstrates a more balanced spread across intensity levels than the original, which is crucial for enhancing image quality, particularly in medical imaging and photographic correction.

There are methods for conducting histogram equalisation in two distinct programming platforms: OpenCV Python and MATLAB. For OpenCV Python, the function cv2.equalizeHist() is utilised to perform histogram equalisation, which adjusts the contrast of an image by equalising its histogram distribution. In MATLAB, the equivalent functionality is achieved using the histeq() function, serving the same purpose of enhancing image contrast through histogram equalisation.

2.1.2 Histogram Matching

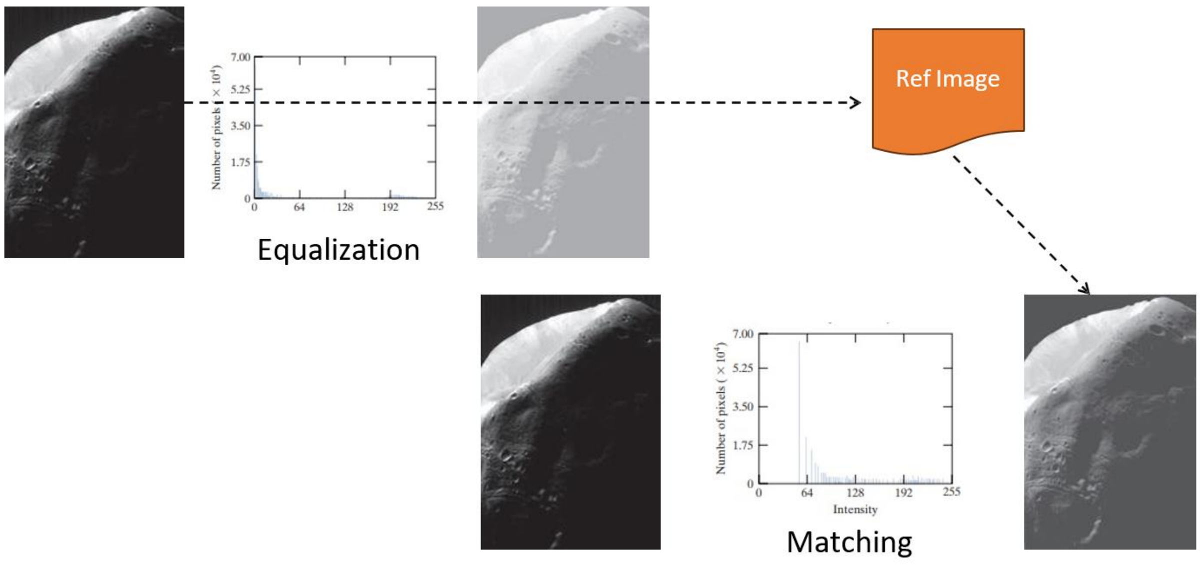


Figure 12: Process of Histogram Matching

Figure 12 depicts the process of histogram matching. Initially, histogram equalisation is performed to enhance the image, illustrated by the transition from the original image to a brighter version with a flatter histogram. However, this equalisation often alters the image's contrast and visual details. To address this, histogram matching is employed, utilising a reference image to guide the adjustment. This technique modifies the equalised image to more closely resemble the reference image's contrast characteristics, as shown by the matched image and its histogram. This approach ensures that while the histogram is equalised, the original image's structural integrity and visual essence are preserved.

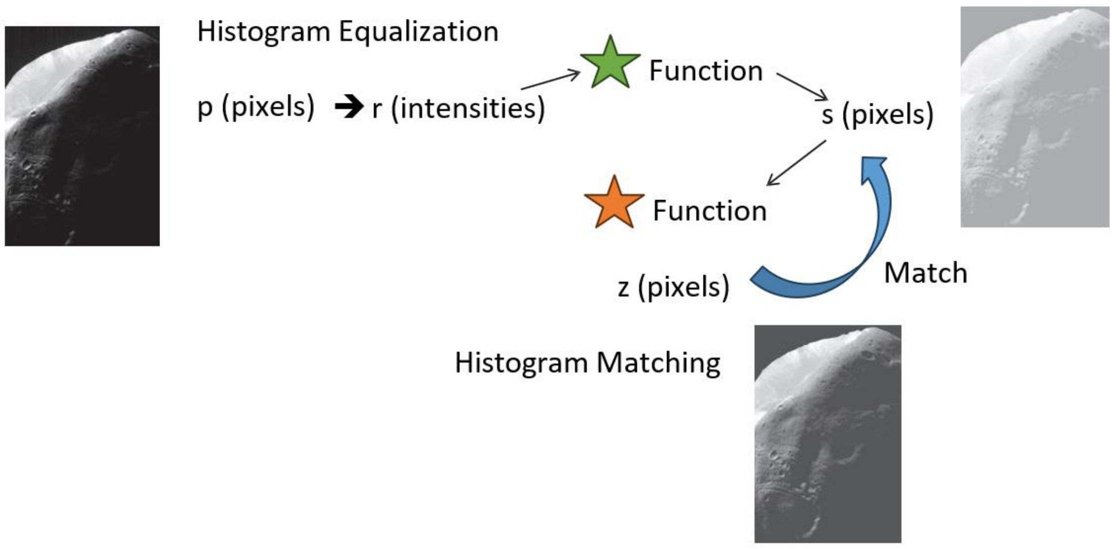


Figure 13: Process Flow of Histogram Equalisation and Matching in Image Enhancement

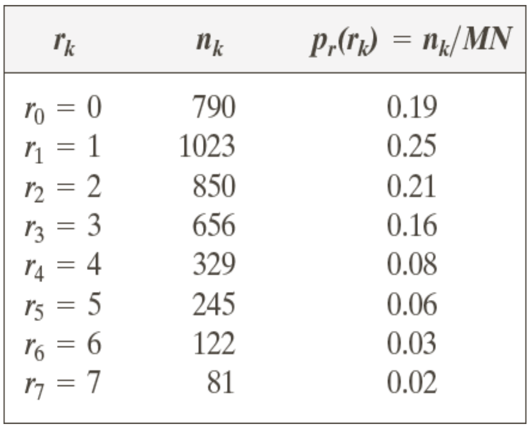
Figure 13 illustrates the two-stage image enhancement process through histogram equalisation followed by matching. Initially, pixel values from the original image are transformed through a function $T(r)$ from pixel values p to intensities r, resulting in a new image represented as s pixels. This transformation aims to equalise the image's histogram, as shown in the upper right part of Figure 13.

Subsequently, histogram matching is applied, which further processes the s pixels using another transformation function to produce z pixels, ensuring the output image has a histogram that matches a reference image. This process not only equalises but also adjusts the image to maintain visual consistency with the reference, as seen in the final image in the lower section of the illustration. This dual-step approach enhances the image while preserving its structural characteristics relative to a reference histogram.

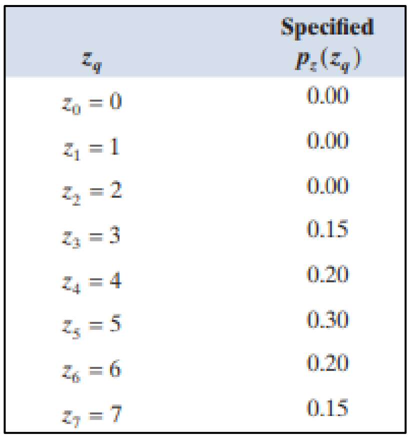
Example Problem for Histogram Matching

Suppose a 3-bit image (L=8) of size 64 × 64 pixels (MN = 4096) has the intensity distribution shown in the following table. This histogram is desired to be transformed to have the values specified in the tables below.

Get the histogram equalisation transformation function and give the $p_s(s_k)$ for each s_k.



r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02



z_q	Specified $p_z(z_q)$
$z_0 = 0$	0.00
$z_1 = 1$	0.00
$z_2 = 2$	0.00
$z_3 = 3$	0.15
$z_4 = 4$	0.20
$z_5 = 5$	0.30
$z_6 = 6$	0.20
$z_7 = 7$	0.15

Explanation:

The above tables provide information about the distributions to be matched (For Example, 0.19 is to be matched with 0.00, etc.)

The equalisation calculation is to be done as below,

$$s_0 = T(r_0) = 7 * 0.19 = 1.35 \qquad s_0 = 1$$
$$s_1 = T(r_1) = 7 * (0.19 + 0.25) = 3.08 \qquad s_1 = 3$$
$$s_2 = T(r_2) = 7 * (0.19 + 0.25 + 0.21) = 4.55 \qquad s_2 = 5$$
$$s_3 = 5.67 \qquad s_3 = 6$$
$$s_4 = 6.23 \qquad s_4 = 6$$
$$s_5 = 6.65 \qquad s_5 = 7$$
$$s_6 = 6.86 \qquad s_6 = 7$$
$$s_7 = 7 \qquad s_7 = 7$$

The below equation is used for histogram matching operation,

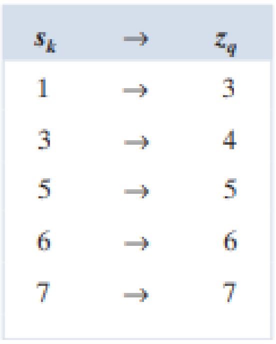
$$G(z_q) = (L-1)\sum_{j=0} p_z(z_i)$$

The substitution solving the equation and rounding off the values to their nearest value is as below,

$$G_0 = T(z_0) = 7 * 0 = 0 \qquad G_0 = 0$$
$$G_1 = T(z_1) = 7 * 0 = 0 \qquad G_1 = 0$$
$$G_2 = T(z_2) = 7 * 0 = 0 \qquad G_2 = 0$$
$$G_3 = T(z_3) = 7 * 0.15 = 1.05 \qquad G_3 = 1$$
$$G_4 = 2.45 \qquad G_4 = 2$$
$$G_5 = 4.55 \qquad G_5 = 5$$
$$G_6 = 5.95 \qquad G_6 = 6$$
$$G_7 = 7 \qquad G_7 = 7$$

During this process, we converted from the R to the S domain and the histogram matched from the S to the Z domain.

After the histogram matching, we get the below conversion,



s_k	$\rightarrow$	z_q
1	$\rightarrow$	3
3	$\rightarrow$	4
5	$\rightarrow$	5
6	$\rightarrow$	6
7	$\rightarrow$	7

2.2 Intensity Transformations

The concept of intensity transformation is used in image enhancement. It contrasts two approaches: spatial domain, where pixel values are directly altered, and frequency domain, involving transformation into a frequency space before processing. This transformation enhances images by modifying pixel intensity values, represented as g(x,y)=T[f(x,y)], to improve visual quality.

There are two primary methods for image processing. The first method, point processing, involves directly modifying individual pixel values and reverting to the original image structure. The second method, neighbourhood processing, employs a kernel or a small cubic function that traverses through the entire image, applying transformations to capture localised effects before producing the final result.

Spatial Filtering:

The concept of spatial filtering is integral to image processing. Spatial filtering manipulates the image by passing or rejecting specific frequency components. It has various filters, such as low-pass, smoothing, and high-pass filters. These filters are implemented using kernels, masks, templates, or windows. These kernels move across the image, applying transformations to a defined neighbourhood of pixels at each step. This operation can be linear or non-linear, affecting the intensity values based on the underlying filter design, thus effectively modifying the image's spatial domain characteristics.

Types of Spatial Filtering:

Consider several practical examples to illustrate point processing and mask operations in image filtering. In point processing, operations occur on a pixel-by-pixel basis. Conversely, mask operations involve computing the sum of values within a neighbourhood around a central pixel. For instance, consider a 3x3 kernel centred on a target pixel; this method aggregates information from the nine nearby pixels to perform various filtering techniques, such as calculating the mean, maximum, or median value representing the central pixel in the output image. This approach is extended in convolution operations used in neural networks, where the impact of each surrounding pixel is weighted differently, fundamentally altering the convolutional process based on the weights assigned to these kernels.

2.2.1 Point Processing

Point processing operations, also called zero-memory operations, involve direct modifications to individual pixels without the need to recall past values or states. This is contrasted with neighbourhood processing, which requires memory to store specific weights for pixel values due to its reliance on surrounding pixel data.

In point processing, one common approach is using linear functions where a scalar multiplier modifies a pixel's value. For example, producing a negative image involves subtracting a pixel's value from the maximum possible value, effectively reversing its intensity on the spectrum.

More complex transformations like logarithmic and power law adjustments also introduce non-linearity into the processing. Logarithmic transformations gradually compress the dynamic range of darker areas more than lighter ones, enhancing details in shadows. On the other hand, power law (or gamma) transformations adjust the image through a gamma function, which can dramatically alter the visual emphasis across its intensity spectrum. These methods showcase the versatility of point processing in adjusting image attributes directly at the pixel level.

Point Processing Techniques:

- **Image Negative**

The concept of image negatives is a basic yet powerful image processing technique. This process involves subtracting pixel values from the maximum intensity level, L−1, to highlight darker regions and enhance contrast. The transformation, represented by s=L−1−r, inversely maps pixel intensities, turning dark areas bright and vice versa. For example, dark regions in the original image, as depicted in Figure 14, like hair or clothing, become significantly lighter in the output image, showcasing the effectiveness of this method in enhancing obscured details. This transformation is depicted graphically and exemplified with before-and-after images, illustrating the stark transformation from original to negative.

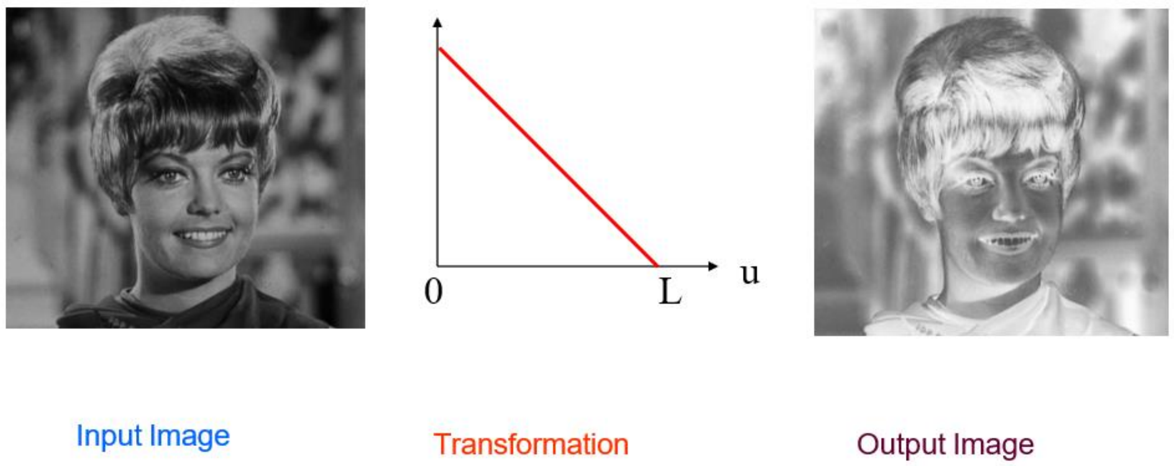


Figure 14: Image Negative

- **Log Transformation:**

Log transformation is commonly applied to expand the dynamic range of images. By employing the formula S=clog(1+r), where S represents the output pixel values and r denotes the input pixel intensities, the transformation stretches the range of lower intensities more than higher ones. This approach is particularly beneficial for enhancing details embedded in darker areas of an image, effectively brightening them without compromising the integrity of lighter areas. The curve on the graph, as in Figure 15, shows how lower intensity values in the input are mapped to a broader range in the output, a principle crucial in improving visual perception of details in images.

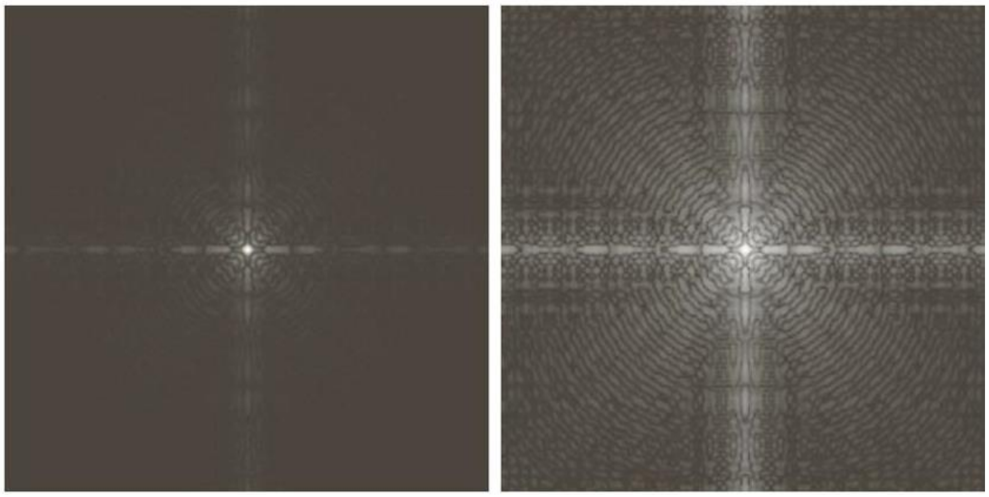


Figure 15: Log Transformation

- **Power-law (Gamma) Transformation:**

The formula s = c * r^{γ}dictates the transformation, where s is the output intensity, r is the input intensity, c is a constant, and γ (gamma) is the exponent that governs the transformation curve's nature. The set of curves displayed in Figure 16 shows the effect of various gamma values on an image's intensity levels.

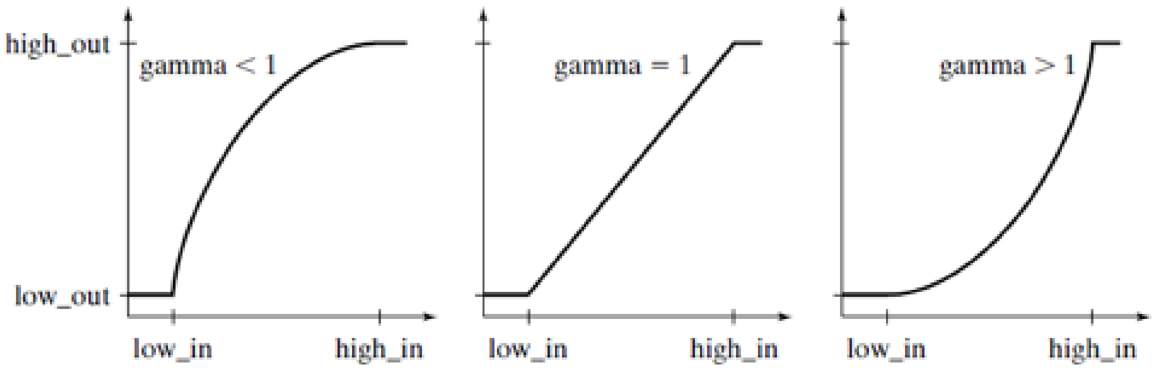


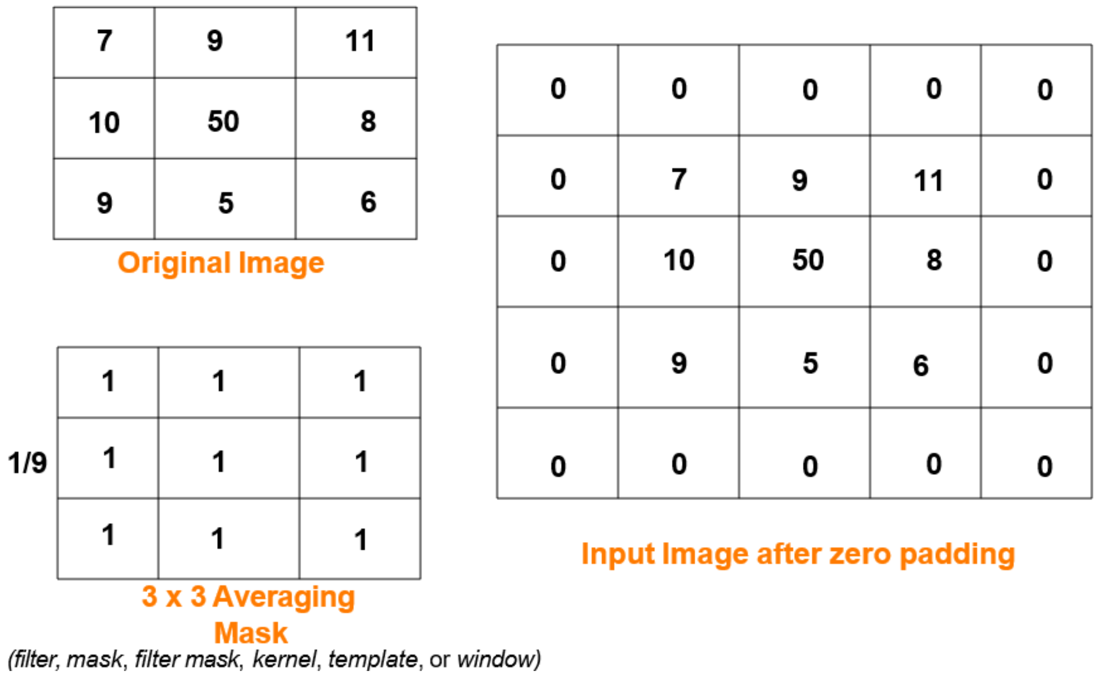
Figure 16: Gamma Values on Image Intensity Levels

The curve shapes represent different gamma values: gamma < 1 enhances dark regions and compresses the pixel value in the bright regions, gamma = 1 maintains the linear relationship between input and output intensities, and gamma > 1 amplifies highlights while compresses the pixel values in the darkening shadows.

2.2.2 Neighbourhood Processing

Neighbourhood processing involves a centred pixel and its surrounding pixels within a predefined kernel size. The operation executed within this kernel yields the response for each pixel, and this process is iterated across the entire image. Linear spatial filtering refers to these operations when they employ linear computations. Conversely, the process is classified as non-linear when the computations are not linear. This method ensures comprehensive image processing, as the emphasised points on critical computations indicate.

- **Illustration of Spatial Filtering**



7	9	11
10	50	8
9	5	6

Original Image

1/9

1	1	1
1	1	1
1	1	1

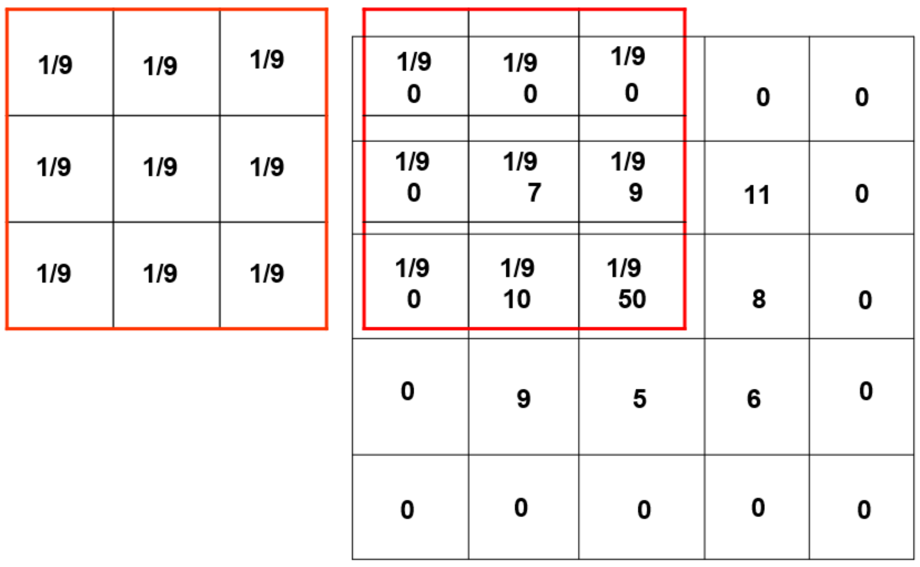
3 x 3 Averaging Mask

(filter, mask, filter mask, kernel, template, or *window)*

0	0	0	0	0
0	7	9	11	0
0	10	50	8	0
0	9	5	6	0
0	0	0	0	0

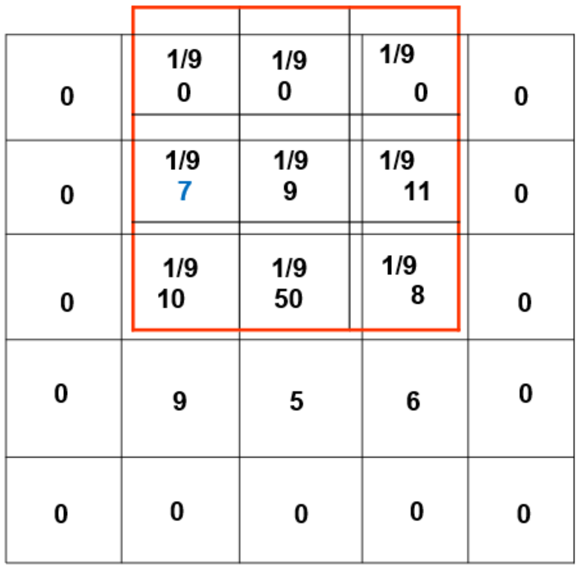
Input Image after zero padding

Let us illustrate the application of a 3x3 averaging mask to an original image matrix for smoothing purposes. To accommodate the mask application along the image borders, the input image is extended through zero padding, creating a border of zeros around the original pixel values. This preparation ensures that each pixel in the original image, including those at the edges, is processed uniformly. This results in a smoother output image by averaging the pixel values within the mask's coverage.



0 x 1/9 + 0 x 1/9 + 0 x 1/9 + 0x 1/9 + 7 x 1/9+ 9 x 1/9+ 0 x 1/9+10 x 1/9+ 50 x 1/9 = 8.4

The averaging mask, which consists of nine entries valued at 1/9, is used to calculate the average of the pixels under it, helping to smooth the image. The calculation shown in the image applies the mask to a section of the input image to produce a new pixel value for the centre pixel. The sum of the products of corresponding elements of the mask and the pixels under it is 8.4, which becomes the new value of the pixel that was originally 50 in the centre of the input matrix. This operation is typically repeated across all the pixels in the image to generate a smooth output.



0 x 1/9+ 0 x 1/9 +0 x 1/9+ 7 x 1/9 +9 x 1/9 +11 x 1/9 +10 x 1/9+50 x 1/9+ 8 x 1/9 = 10.5

The filter, composed of each element having a value of 1/9, covers a portion of the image grid to perform an averaging operation. In the calculation displayed:

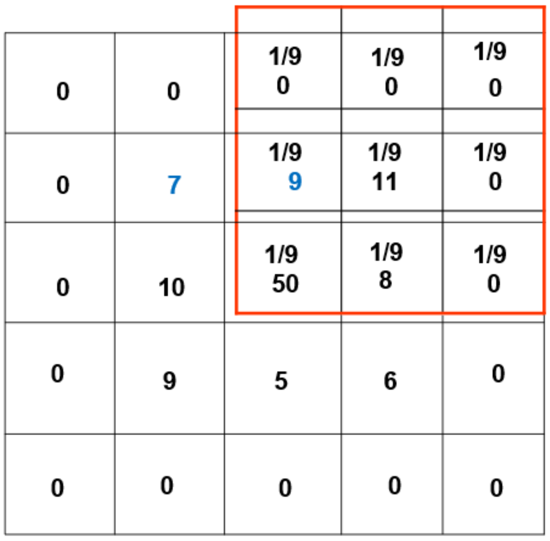
Each pixel value within the red box is multiplied by 1/9.

The specific values processed by the filter are 0, 0, 0 at the top, 7, 9, 11 in the middle, and 10, 50, and 8 at the bottom.

The result of the averaging computation, shown at the bottom, is calculated as follows:

0 × 1/9 + 0 × 1/9 + 0 × 1/9 + 7 × 1/9 + 9 × 1/9 + 11 × 1/9 + 10 × 1/9 + 50 × 1/9 + 8 × 1/9 = 10.5

The resulting value, 10.5, is then used to replace the centre pixel of the filter's coverage area in the output image, achieving a local smoothing effect. This process illustrates how convolutional operations use image processing filters to modify pixel values based on their local neighbourhoods.



0	0	1/9 0	1/9 0	1/9 0
0	7	1/9 9	1/9 11	1/9 0
0	10	1/9 50	1/9 8	1/9 0
0	9	5	6	0
0	0	0	0	0

0 x 1/9+ 0 x 1/9 +0 x 1/9+9 x 1/9 +11 x 1/9+0 x 1/9 +50 x 1/9 +8 x 1/9 +0 x 1/9 = 8.6

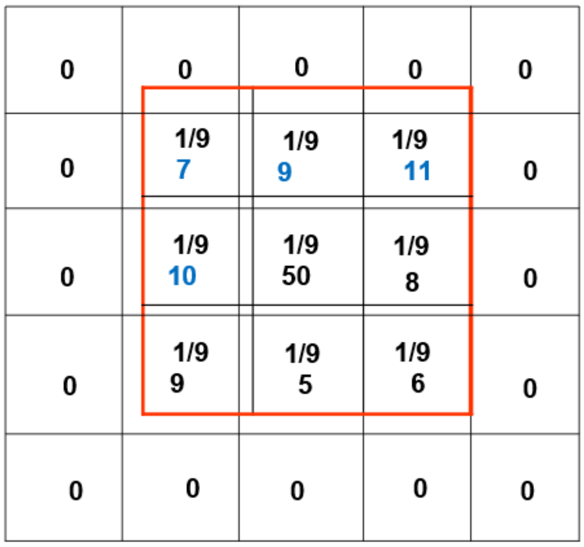
This step focuses on a region near the edge where zero-padding is involved. This process is used to smooth the image and is a standard operation in image processing to reduce noise or detail.

Here's the breakdown of the operation:

- The 3x3 averaging filter, each element with a value of 1/9, is applied to a portion of the image that includes zero-padded cells. This is necessary because the processed region includes parts of the image edge without enough adjacent pixels.
- The region under the filter includes pixel values from the zero-padded area and the image itself: 0, 0, 0 across the top; 7, 9, 11 in the middle; and 10, 50, 8 at the bottom.
- The calculation for the new pixel value is done by averaging all these values:

$$(0+0+0+7+9+11+10+50+8)\times 1/9=10.555$$

- The computed average value, 8.6 (after rounding down or approximating), will replace the pixel value at the centre of this mask in the output image. This is a part of the convolution process where the influence of surrounding pixels, including artificial zeros, is considered to determine the new pixel value, resulting in smoothing effects near the edges or corners of the image.



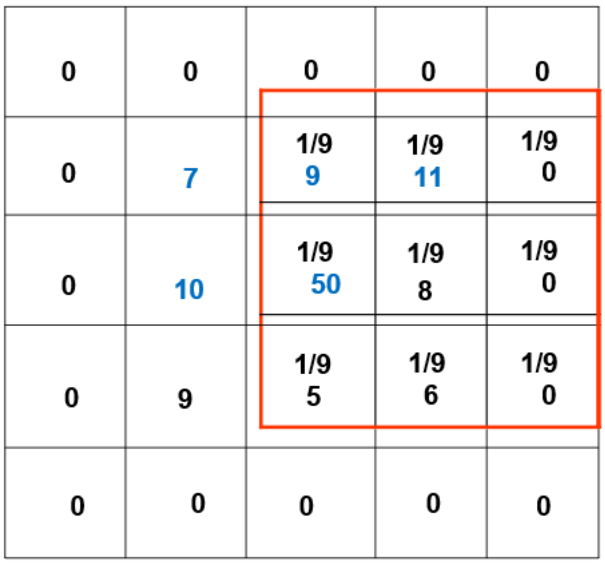
0	0	0	0	0
0	1/9 7	1/9 9	1/9 11	0
0	1/9 10	1/9 50	1/9 8	0
0	1/9 9	1/9 5	1/9 6	0
0	0	0	0	0

7 x 1/9 +9 x 1/9 +11 x 1/9 +10 x 1/9 +50 x 1/9+8 x 1/9+9 x 1/9 +5 x 1/9+ 6 x 1/9 = 12.7

This step focuses on the following steps:

- The 3x3 averaging mask, with each element having a value of 1/9, overlays a portion of the image, including the actual image data and zero-padded edges.
- The region under consideration for the averaging operation includes the following pixel values: 0 (padded), 7, 9, 11 on the top row; 0 (padded), 10, 50, 8 in the middle row; and 0 (padded), 9, 5, 6 on the bottom row.
- The calculation for determining the new pixel value at this position in the output image involves averaging all these pixel values:
- $(0\times 1/9)+(7\times 1/9)+(9\times 1/9)+(11\times 1/9)+(0\times 1/9)+(10\times 1/9)+(50\times 1/9)+(8\times 1/9)+(0\times 1/9)+(9\times 1/9)+(5\times 1/9)+(6\times 1/9)=12.7$

This computed average, 12.7, will be used as the new value for the pixel at the filter's centre in the processed image. This averaging process helps reduce noise and blur the image, effectively smoothing the local variations in pixel intensities.



9 x 1/9 + 11 x 1/9 + 0 x 1/9 +50 x 1/9 +8 x 1/9 + 0 x 1/9 +5 x 1/9+6 x 1/9+0 x 1/9 = 9.8

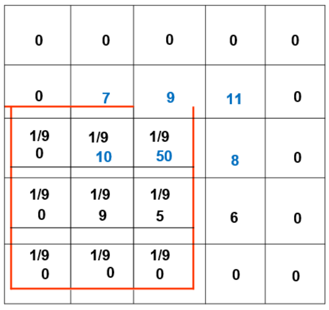
Here is the detailed calculation for the new pixel value:

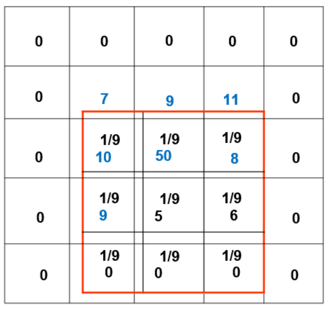
- Pixel Values under the Filter: The filter's centre covers the pixel value of 50. Surrounding it are pixel values 9, 11, 10, 8, 5, and 6, with 0s in the positions corresponding to the padded areas.
- Calculation: Each pixel value under the filter is multiplied by 1/9 (since it's an averaging mask), and then all these values are summed to calculate the average:

(9× 1/9)+(11× 1/9)+(0× 1/9)+(10× 1/9)+(50× 1/9)+(8× 1/9)+(5× 1/9)+(6× 1/9)+(0× 1/9) =9.8

This computed value, 9.8, becomes the new value for the central pixel in the processed image at this specific location. This averaging process aims to blur and reduce the noise in the image, making the overall image smoother by blending the pixel values within local neighbourhoods.

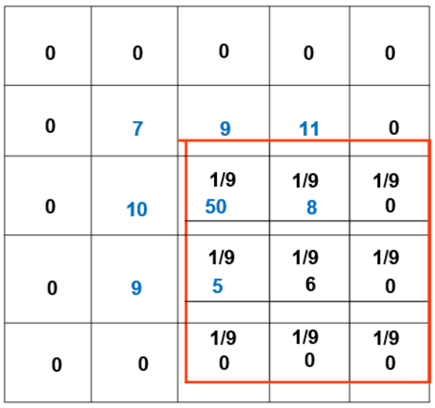
The process continues with the remaining values as below:





0 x 1/9 +10 x 1/9 +12 x 1/9 +0 x 1/9 +9 x 1/9+ 5 x 1/9+0 x 1/9 + 0x 1/9 +0 x 1/9 = 8.2

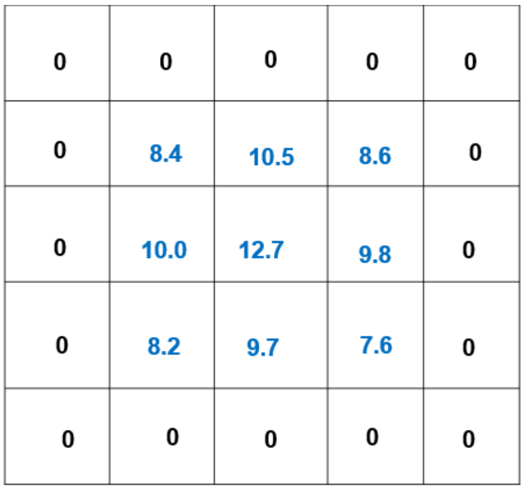
10 x 1/9 +50 x 1/9 + 8x 1/9 + 9 x 1/9 +5 x 1/9+ 6 x 1/9+0 x 1/9+0 x 1/9+0 x 1/9 = 9.7



0	0	0	0	0
0	7	9	11	0
0	10	1/9 50	1/9 8	1/9 0
0	9	1/9 5	1/9 6	1/9 0
0	0	1/9 0	1/9 0	1/9 0

12 x 1/9 + 5 x 1/9 + 0 x 1/9 + 4 x 1/9 +6 x 1/9+0 x 1/9 + 0 x 1/9 + 0 x 1/9 + 0 x 1/9 = 7.6

```
[[ 8.444445 10.555555 8.666667 ]
 [10. 12.777778 9.888889 ]
 [ 8.222222 9.777778 7.6666665]]
```



0	0	0	0	0
0	8.4	10.5	8.6	0
0	10.0	12.7	9.8	0
0	8.2	9.7	7.6	0
0	0	0	0	0

The result of Averaging Filter is given below,

7	9	11
10	50	8
9	5	6

Original Image

8	11	9
10	13	10
8	10	8

Image after Spatial Averaging

```
[[ 8.444445 10.555555 8.666667 ]
 [10. 12.777778 9.888889 ]
 [ 8.222222 9.777778 7.6666665]]
```

- **Smoothing**

Smoothing is a fundamental image-filtering technique used to enhance the clarity of images by reducing noise and unwanted artefacts. This process is particularly useful in medical imaging, such as brain CT scans, where extraneous white regions can obscure diagnostic details. Employing a Gaussian filter, for instance, helps integrate pixel values with their neighbours, effectively diminishing white patches and enhancing the overall image quality. This technique adjusts the intensity across the image to produce a cleaner output without the artefacts, leveraging different kernel sizes, such as a 5x5, to vary the level of smoothness. Various filters, like Gaussian or simple averaging, can yield different outcomes depending on their mathematical functions.

3. Average Versus Weighted Average Filters

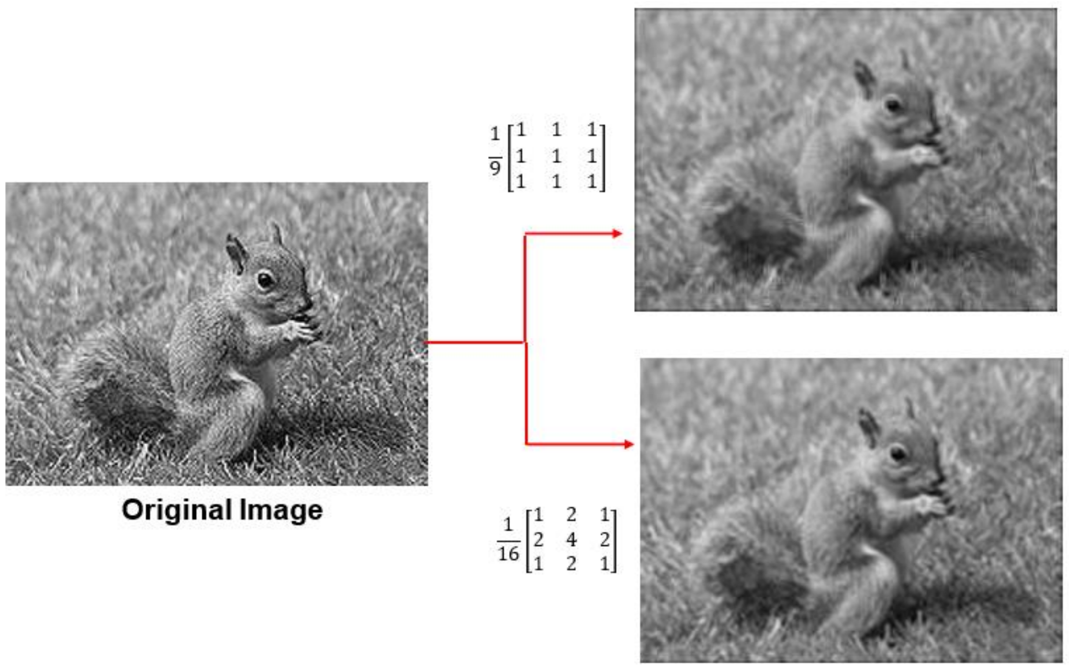

Original Image
$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$
$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$

Figure 17: Average Versus Weighted Average Approaches

The first example illustrates a simple averaging approach, utilising a uniform 1/9 value across a 3x3 matrix. Conversely, the second example employs a weighted filter with a distinct configuration of [1, 2, 1; 2, 4, 2; 1, 2, 1], demonstrating the allocation of different weights to each pixel.

These weights are not arbitrarily chosen but are often derived from established filter designs available in various image processing software or can be custom-defined based on specific processing requirements. These predefined filters are instrumental for multiple tasks and are readily available.

In more advanced contexts such as deep learning, filter weights start as random values and are optimised during training to achieve desired results. Unlike these adaptable models, traditional filters like the Sobel operator have fixed weights designed to address specific image features. In the upcoming presentation on frequency modulation, I will further discuss how these filters function and explore their implementation in different scenarios, offering an opportunity to delve into the practicalities of selecting and applying optimal filter configurations for image enhancement.

4. Non-Linear Filters

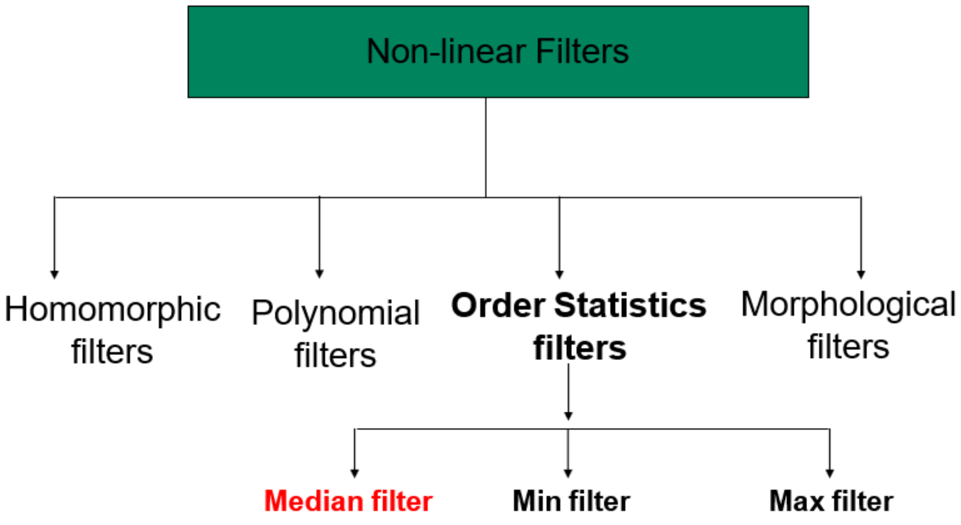


Figure 18: Categories of Non-Linear Filters

Figure 18 categorises various types of non-linear filters used in image processing. These filters are grouped into four main types:

- **Homomorphic Filters** - These filters manage variations in illumination and enhance contrast in images.
- **Polynomial Filters** - These implement operations based on polynomial functions on the pixel values, enhancing specific image features.
- **Order Statistics Filters** - This category includes filters that depend on the ranking of pixel values within a neighbourhood. Key examples are:
  - **Median Filter** - Replaces a pixel's value with the median value of its neighbourhood, effective for removing noise like salt and pepper noise.

- **Min Filter** - Takes the minimum value from the pixel's neighbourhood, which helps erode image regions.
- **Max Filter** - Uses the maximum value from the neighbourhood, which helps dilate image features.
- **Mean Filter**- Replaces the centre pixel value with the mean of all pixel points.

- **Morphological Filters** - These filters process images based on their shapes. They are used in edge detection and noise removal tasks and typically utilise structuring elements to probe and transform the input image.

These filters are essential for enhancing image quality and extracting useful information from the visuals by manipulating pixel values non-linearly.

Let us explore the Mean and Median filters in detail:

Mean Filter:

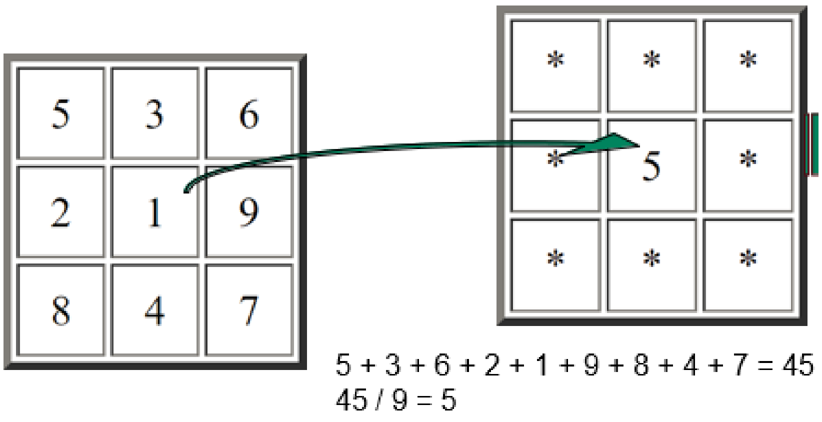
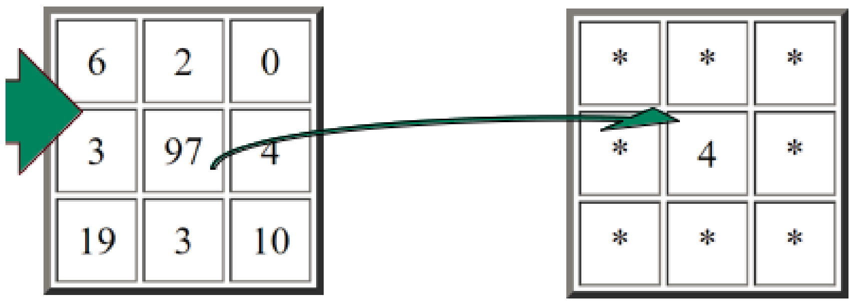


Figure 19: Mean Filter Example

This filter calculates the average of all pixel values within the 3x3 window and replaces the centre pixel value with this average. For example, as per the Figure 19, the sum of the values (5, 3, 6, 2, 1, 9, 8, 4, 7) is 45. Dividing by 9 (the number of pixels in the window), the result is 5, which becomes the new value of the centre pixel.

Median Filter:



0, 2, 3, 3, **4**, 6, 10, 15, 97

Figure 20: Median Filter Example

Unlike the mean filter, the median filter replaces the centre pixel value with the median (middle value) of all pixel values in the window. As per Figure 20, the values are first sorted (0, 2, 3, 3, 4, 6, 10, 15, 97), and the median here is 4, which replaces the centre pixel.

Variations of Median filter:

- Weighted Median filter
- Centre Weighted Median filter
- Adaptive Centre Weighted Median filter

Both filters are used for different purposes: the mean filter for smoothing the image by reducing intensity variations between pixels, and the median filter for reducing noise while preserving edges because it is less sensitive to extreme values that do not represent most of a local area.

5. Summary

In this topic, we discussed:

- Image Enhancement – Spatial and Frequency Domain Technique
- Histogram processing and matching
- Intensity Transformation Function
- Point Processing
- Linear/Logarithmic/Power-Law Transform

- Neighbourhood Processing
- Smoothing Spatial Filter-LPF
- Categories of Non-Linear Filters
- Average vs Weighted Average filter
- Median and Mean filter