Computer Vision

Assignment (1)

Under The Supervision of:

Dr. Ahmed Badawi

Team (4) Members:

1. Fady Mohsen

2. Camellia Marwan

3. Lamees Mohamed

4. Lama Zakaria

5. Rana Ibrahim

Table of Contents

1st: Noise Addition 3

A. Uniform Noise: 3

B. Gaussian Noise 4

C. Salt and Pepper Noise 6

2nd: Filtering Noisy Images 7

A. Average Filter 7

B. Gaussian Filter 8

C. Median Filter 8

3rd: Edge Detection 9

A. Sobel Edge Detection 9

B. Prewitt Edge Detection 9

C. Roberts Edge Detection 10

D. Canny Edge Detection 10

4th: Histogram and Distribution Curve 13

5th: RGB Histogram and Cumulative Distribution Function 13

Implementation: 14

6th: Global & Local Thresholding 15

7th: Frequency Domain Filters 16

Low Pass Filter (Smoothing): 16

I. Ideal Low Pass Filter: 16

II. Gaussian Low Pass Filter: 17

III. Butterworth Low Pass Filter: 18

High Pass Filter (Sharpening): 18

I. Ideal High Pass Filter: 18

II. Gaussian High Pass Filter: 19

III. Butterworth High Pass Filter: 19

8th: Hybrid Images: 20

9th: Image Equalization 21

10th: Image Normalization 22

1st: Noise Addition

Adding noise to images is a common technique used to simulate real-world noise introduced by electronic sensors and to test the robustness of image processing algorithms.

Three types of noise commonly added to images are:

A. Uniform Noise:

Uniform noise is added by adding random values sampled from a uniform distribution to each pixel intensity in the image.

$$p(z) = \begin{cases} \dfrac{1}{b-a} & \text{if } a \le z \le b \\ 0 & \text{otherwise} \end{cases}$$

Equation 1: Uniform Noise

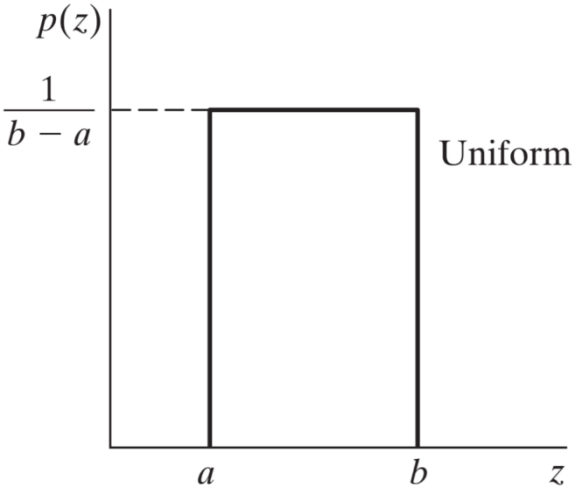


Figure 1: Uniform Noise

Observations and Results

Uniform noise introduces a consistent level of randomness across the image, affecting all pixel intensities equally. Visually, it appears as a grainy pattern superimposed on the image.

Figure 2: Image Before Adding Noise

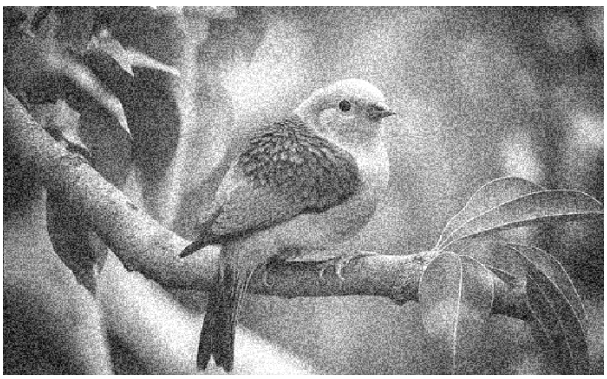


Figure 3: Image After Adding Noise

The range of values used to generate the random noise determines the amplitude or intensity of the noise added to the image. A wider range results in more significant variations in pixel intensities and, consequently, more noticeable noise in the image.

Figure 4: Original Image Before Adding Uniform Noise

Figure 5: Image After Adding Uniform Noise With Bigger Range

B. Gaussian Noise

Gaussian noise, also known as Gaussian white noise is generated by adding random values sampled from a Gaussian (normal) distribution to the pixel intensities of an image. These random values are sampled from a Gaussian distribution with a specified mean and standard deviation, the mean of the Gaussian distribution often corresponds to zero, meaning that the noise has an average intensity of zero, while the standard deviation controls the amplitude or intensity of the noise.

$$g(x,y) = f(x,y) + n(x,y)$$

Equation 2: Gaussian Noise - Part (1)

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}} \qquad -\infty < z < \infty$$

Equation 3: Gaussian Noise - Part (2)

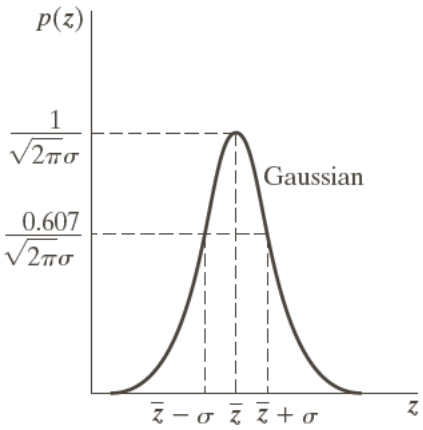


Figure 6: Gaussian Graph

Observations and Results

Gaussian noise manifests as a random variation in pixel intensities across the image. It can cause blurring, affecting sharpness and loss of detail, as well as degrade image quality by introducing unwanted artifacts.

Figure 8: Original Image Before Adding Gaussian Noise

Figure 7: Original Image After Adding Gaussian Noise

The severity of Gaussian noise depends on the standard deviation of the noise distribution: higher standard deviations result in more intense noise with greater variation in pixel values.

Figure 10: Original Image Before Adding Gaussian Noise

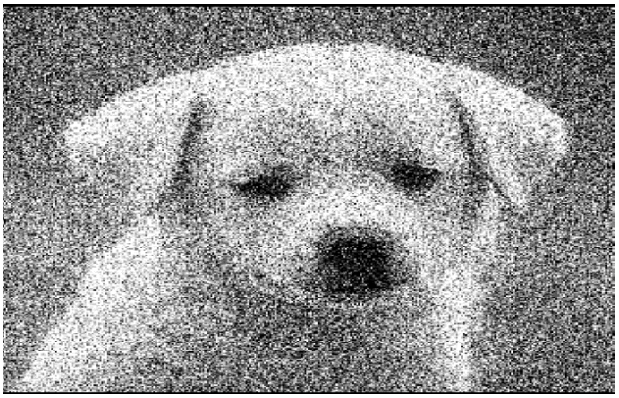
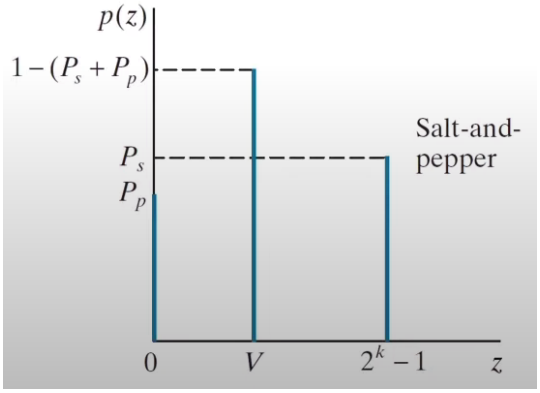


Figure 9: Image after adding uniform noise with bigger standard deviation

C. Salt and Pepper Noise

Salt and pepper noise derives its name from the visual analogy of sprinkled salt and pepper grains in a dish. In digital images, salt and pepper noise manifests as randomly occurring bright (salt) and dark (pepper) pixels, which appear as isolated spots or clusters distributed across the image.

$$p(z) = \begin{cases} P_s & \text{for } z = 2^k - 1 \quad \text{salt} \\ P_p & \text{for } z = 0 \quad \text{pepper} \\ 1 - (P_s + P_p) & \text{for } z = V \quad \text{unchanged} \end{cases}$$



Observations and Results

The presence of bright and dark spots degrades image quality and disrupts the visual appearance of the image, making it appear corrupted. It can obscure important image features and details, making it challenging to interpret or analyse the image effectively.

Figure 12: Original Image Before Adding Salt & Pepper Noise

Figure 11: Image After Adding Salt & Pepper Noise

2nd: Filtering Noisy Images

Filtering noisy images is a crucial step in image processing aimed at reducing or removing unwanted noise while preserving important image features.

Here we explore three common low-pass filters used for filtering noisy images:

A. Average Filter

The average filter, also known as the box filter, is a simple and widely used technique for smoothing images and reducing noise. It operates by replacing each pixel in the image with the average value of its neighbouring pixels within a defined kernel or window.

Observations and Results

By replacing each pixel value with the average of its neighbours, the average filter effectively smooths out noise and reduces high-frequency variations in the image. This smoothing operation helps to blur sharp edges and small details in the image, making it appear smoother and less noisy.

Figure 14: Image with uniform noise

Figure 13: Image after applying average filter (3x3 kernel)

Figure 16: Using 5x5 kernel

Figure 15: Using 11x11 kernel

B. Gaussian Filter

The Gaussian filter is a more advanced smoothing technique that applies a weighted average to neighbouring pixels, with weights determined by a Gaussian distribution. Unlike the average filter, which assigns equal weights to all pixels within the kernel, the Gaussian filter gives more weight to nearby pixels and less weight to distant ones. This preserves edges and important image features better than the average filter.

Observations and Results

The Gaussian filter smooths the image by blurring sharp transitions and high-frequency variations while preserving gradual changes in intensity. This smoothing effect helps to reduce noise and make the image appear more visually pleasing. However, unlike the average filter, which can produce noticeable blurring, the Gaussian filter achieves smoother results with minimal loss of detail.

Figure 17: Image with uniform noise

Figure 18: Image after applying gaussian filter

C. Median Filter

The median filter is a nonlinear filtering technique that replaces each pixel value with the median value of its neighbouring pixels within a defined kernel. Unlike linear filters such as the average and Gaussian filters, the median filter does not blur edges.

Observations and Results

The median filter is particularly effective in removing salt and pepper noise without significantly blurring the image, making it suitable for preserving fine details.

Figure 19: Image with salt & pepper noise

Figure 20: Image after applying median filter

3rd: Edge Detection

The goal of edge detection is to identify the boundaries of objects within an image, where abrupt changes in intensity occur.

Sobel, **Prewitt**, and **Roberts** these methods use convolutional filters to calculate the gradient of an image, highlighting areas of high intensity change that are likely to correspond to edges.

Canny Edge Detection is known for its effectiveness in detecting a wide range of edges while minimizing false positives.

A. Sobel Edge Detection

Two 3x3 convolution kernels are applied to the image, one for detecting horizontal changes and the other for vertical changes.

$$\text{Horizontal Sobel kernel: } \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \textit{Vertical Sobel Kernel}: \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Image after Horizontal Kernel **Image after Vertical Kernel**

B. Prewitt Edge Detection

$$\text{Horizontal Prewitt kernel: } \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \text{Vertical Prewitt Kernel: } \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Image after Horizontal Kernel **Image after Vertical Kernel**

C. Roberts Edge Detection

Horizontal Roberts kernel: $\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$, Vertical Roberts Kernel: $\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$

Image after Horizontal Kernel

Image after Vertical Kernel

D. Canny Edge Detection

The Canny edge detection algorithm is a multi-step process used to detect a wide range of edges in images. It is known for its ability to produce precise results with low error rates.

The algorithm consists of the following steps:

1- **Preprocessing**
2- **Gradient Calculation**
3- **Gradient Magnitude and Direction**
4- **Non-maximum Suppression**
5- **Double Thresholding and Hysteresis**

Pre-processing

The preprocessing is done by Applying a Gaussian blur to the image using a 5x5 kernel with a standard deviation of 4, its purpose is to Reduce noise in the image to improve edge detection accuracy.

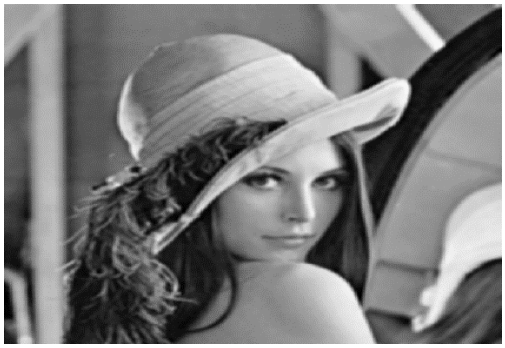


Figure 21: Image after Preprocessing Step

Gradient Calculations

Gradients are used to determine the intensity changes in different directions, which can indicate the presence of edges. We calculate the gradients in the x and y directions using the Sobel operator.

Image after Gradient Calculations

Gradient Magnitude and Direction

The gradient magnitude represents the strength of the edge at each pixel. A high gradient magnitude indicates a strong intensity change, which is likely to be an edge.

The gradient direction indicates the orientation of the edge at each pixel. The direction is important for non-maximum suppression.

Non-Maximum Suppression

The purpose of is to thin out the edges to keep only the most prominent ones. For each pixel, compare its gradient magnitude with its neighbours in the gradient direction. Keep the pixel value if it is the maximum, otherwise, set it to 0. The idea is to take the pixels which are almost perpendicular direction to the angle of the main pixel.

Figure 22: Image after Non-Maximum Suppression

Double Thresholding and Hysteresis

The purpose of Double Thresholding is to classify edge pixels as strong, weak, or non-edges based on two thresholds:

1- **High Threshold:** Set to 9% of the maximum gradient magnitude.
2- **Low Threshold:** Set to 5% of the maximum gradient magnitude.

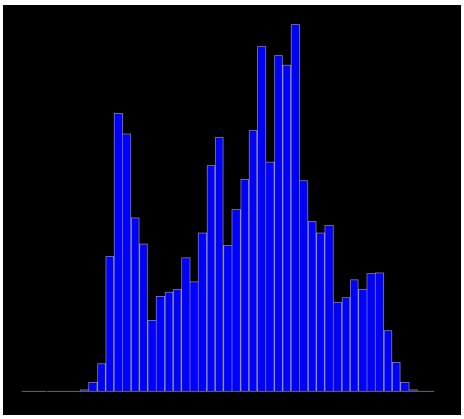
Pixels with a gradient magnitude above the high threshold are classified as strong edges, while those between the high and low thresholds are classified as weak edges.

Pixels with a gradient magnitude between the low and high threshold are considered weak edges. These pixels only assign pixel value of 255, if they are connected to a strong edge, else assign pixel value of 0 to it, this process is called hysteresis.

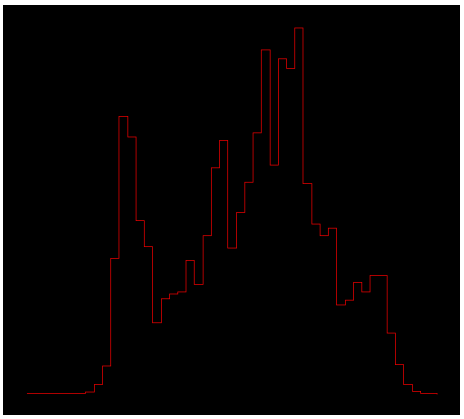
Figure 23: Final Canny Edge Detected Image

4th: Histogram and Distribution Curve

- The **histogram** visually represents the frequency of occurrence of different pixel intensity values in an image. We calculate a histogram with 50 bins, each representing a range of pixel intensities.
- The **distribution curve** provides a smoothed representation of the histogram, showing the overall distribution of pixel intensities.



Histogram Curve



Distribution Curve

5th: RGB Histogram and Cumulative Distribution Function

RGB histograms are graphical representations of the distribution of pixel intensities in colour images across the red, green, and blue channels. They provide valuable insights into the colour composition of an image by visualizing the frequency of occurrence of different colour values.

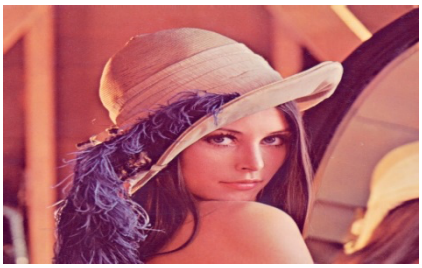
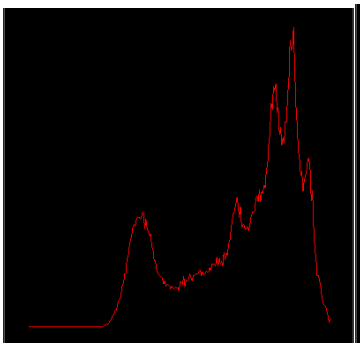
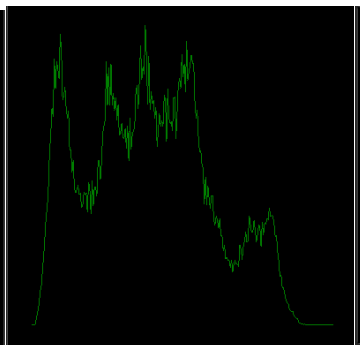


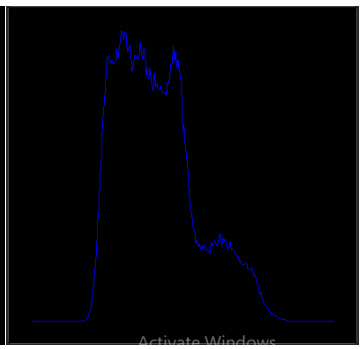
Image Before Processing



Red Channel

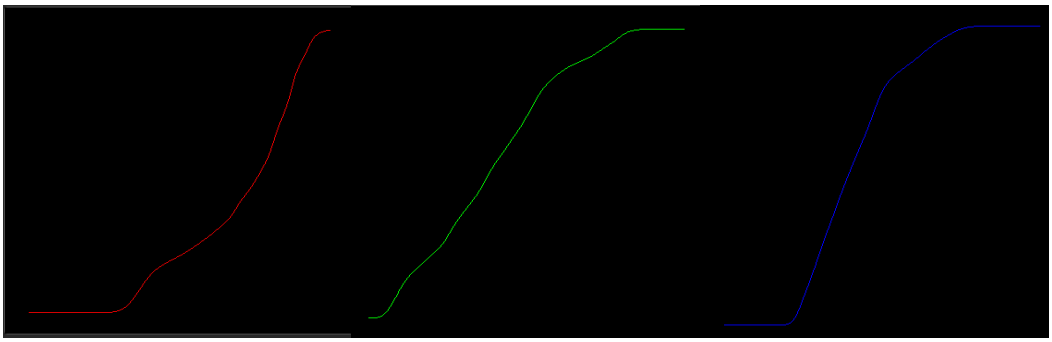


Green Channel



Blue Channel

Cumulative functions, often derived from histograms, represent the cumulative distribution of pixel intensities in an image. They depict the cumulative sum of pixel counts up to a certain intensity level, providing insights into the overall distribution characteristics.



Red Channel **Green Channel** **Blue Channel**

Implementation:

1. **Histogram Computation:** Iterate over each pixel in the image and accumulate counts for different color values in the red, green, and blue channels.
2. **Plotting:** Plot the histogram values against the corresponding color intensities for each channel using plotting libraries such as Matplotlib or PyQtGraph.
3. **Visualization:** Display the histograms alongside the original image to facilitate visual inspection and analysis.

6th: Global & Local Thresholding

Global thresholding is a basic image segmentation technique where a single threshold value is applied uniformly to the entire image to separate foreground objects from the background. Pixels with intensities above the threshold are classified as foreground, while those below are classified as background.

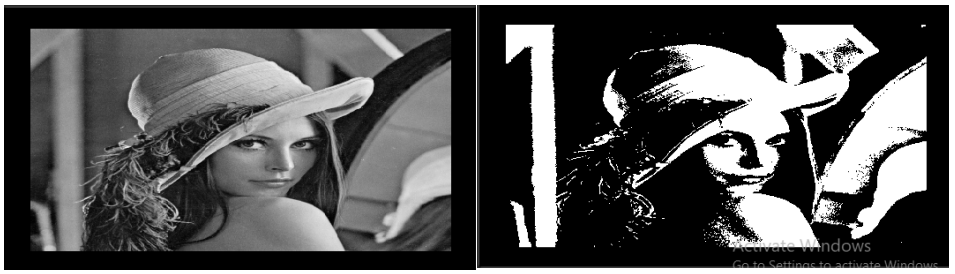


Image Before Edit **Image After Edit**

Local thresholding, also known as adaptive thresholding, is a more sophisticated segmentation technique that computes threshold values for individual image regions based on local pixel neighbourhoods. It allows for better adaptation to variations in illumination and contrast across different parts of the image.

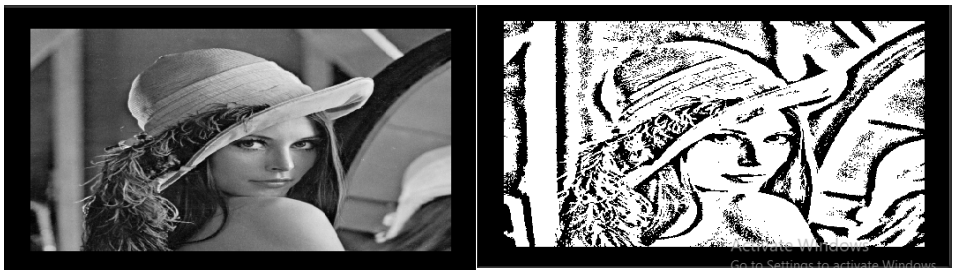
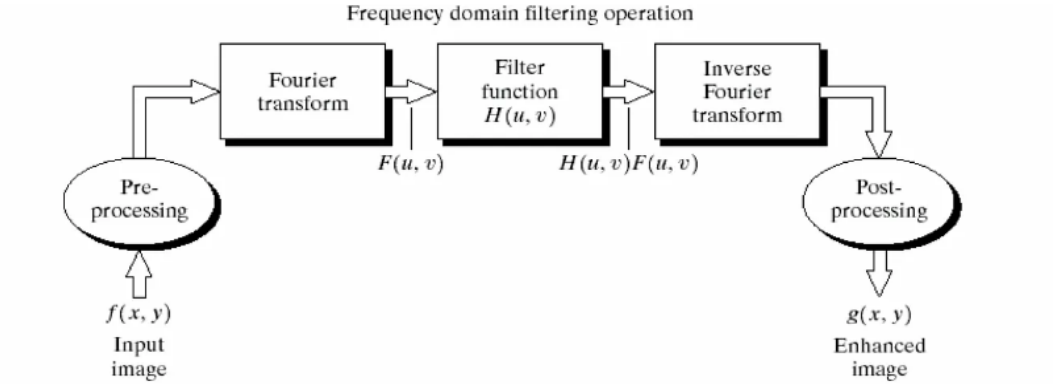


Image Before Edit **Image After Edit**

7th: Frequency Domain Filters

Frequency Domain Filters are used for smoothing and sharpening of image by removal of high or low frequency components. Sometimes it is possible of removal of very high and very low frequency. Frequency domain filters are different from spatial domain filters as they basically focus on the frequency of the images **(**get Fourier transform of image**)**. It is basically done for two basic operations i.e., Smoothing and Sharpening.



Low Pass Filter (Smoothing):

Low pass filter removes the high frequency components that means it keeps low frequency components. It is used for smoothing the image. It is used to smoothen the image by attenuating high frequency components and preserving low frequency components.

I. Ideal Low Pass Filter:

Simply cut off all high frequency components that are a specified distance D0 from the origin of the transform.

The transfer function for the ideal low pass filter**:**

$$H(u,v) = \begin{cases} 1 & D(u,v) \le D_o \\ 0 & D(u,v) > D_o \end{cases}$$

Where D(u, v) is given as :

$$D(u,v) = \sqrt{(u - M/2)^2 - (v - N/2)^2}$$

M: Number of rows, **N**: Number of columns, **D0**: Cut off Frequency

Gray scale Image

After Filtering with 21 radius

II. Gaussian Low Pass Filter:

The Gaussian filter has a smooth frequency response with no ripples, making it effective for noise reduction and smoothing applications.

The transfer function for the gaussian low pass filter**:**

$$H(u,v) = e^{-D^2(u,v)/2D_o{}^2}$$

Image Gray scale

Image After Filtering with 64 radius

III. Butterworth Low Pass Filter:

The transfer function for BLPF of order n:

$$H(u,v) = \frac{1}{1+(D(u,v)/D_o)^{2n}}$$

Image Gray scale

Image After Filtering with 64 Radius

High Pass Filter (Sharpening):

High pass filter removes the low frequency components that means it keeps high frequency components. It is used for sharpening the image. It is used to sharpen the image by attenuating low frequency components and preserving high frequency components.

I. Ideal High Pass Filter:

Simply cut off all low frequency components that are a specified distance D0 from the origin of the transform.

The transfer function for the ideal high pass filter**:**

$$H(u,v) = \begin{cases} 0 & D(u,v) \le D_o \\ 1 & D(u,v) > D_o \end{cases}$$

Image Gray scale

Filtering with 28 Radius

II. Gaussian High Pass Filter:

The transfer function for the gaussian high pass filter**:**

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_o{}^2}$$

Image Gray scale

Filtering with 28 radius

III. <u>Butterworth High Pass Filter:</u>

The transfer function for BLPH of order n:

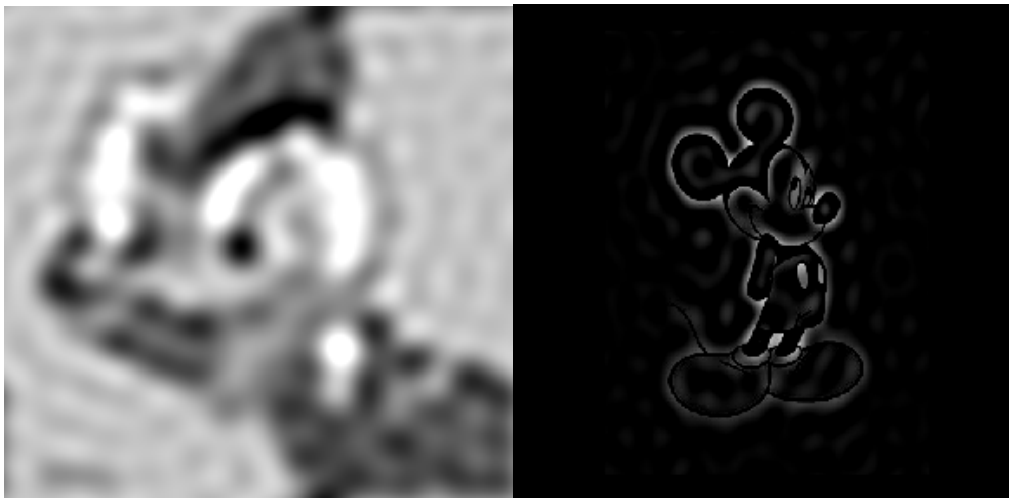
$$H(u,v) = 1 - \frac{1}{1 + (D(u,v)/D_o)^{2n}}$$

Image Gray scale

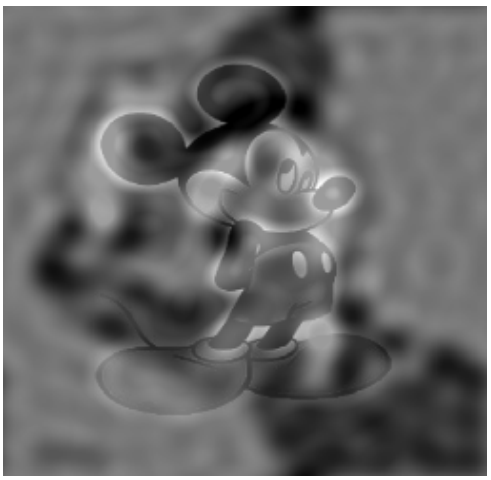
Filtering with 28 radius

8th: Hybrid Images:

Hybrid Images is combining two images with different frequency content to create a single image that conveys different interpretations at different viewing distances. The basic idea is to merge the low-frequency components of one image with the high-frequency components of another image.
Apply a low-pass filter to the first image to retain its low-frequency components. Similarly, apply a high-pass filter to the second image to isolate its high-frequency components.



After Applying LPF **After Applying HPF**



Result

9th: Image Equalization

Image equalization, often referred to as histogram equalization, is a technique used in image processing to improve the contrast of an image. It works by redistributing the intensity values of an image so that they span a wider range, making the details in both dark and bright areas more visible.

The process involves the following steps:

1. **Histogram Calculation:** Compute the histogram of the image, which is a graph showing the distribution of pixel intensities.
2. **Cumulative Distribution Function (CDF):** Calculate the cumulative distribution function from the histogram. The CDF indicates the cumulative probability of each intensity level.
3. **Equalization:** Use the CDF to map the original pixel values to new values that spread more evenly across the available intensity range. This mapping is done such that the new histogram of the image has a roughly uniform distribution.
4. **Reassign Pixel Values:** Replace each pixel in the original image with its new intensity value from the equalization step.

Here are two images for illustration:

- **Origin:** The first image is a grayscale photograph with poor contrast, where details in both dark and light areas are hard to distinguish due to the limited range of intensity values.
- **After Equalization:** The second image is the result of applying histogram equalization to the first image, showcasing enhanced contrast with more distinguishable details in both dark and light regions and a wider range of intensity values.

10th: Image Normalization

- Image normalization, also known as intensity normalization or contrast stretching, is a process used in image processing to adjust the range of pixel intensity values in an image. The goal of image normalization is to improve the contrast and visibility of features in the image by scaling the intensity values to a specified range, typically [0, 255] for 8-bit grayscale images or [0, 1] for floating-point images.
- Normalization is particularly useful in situations where the image has low contrast, or the intensity values are concentrated in a narrow range. By spreading the intensity values across a wider range, normalization can make details in the image more visible and enhance the overall appearance of the image.

The process of image normalization typically involves the following steps:

1. **Determine the Range:** Find the minimum and maximum intensity values in the original image.
2. **Apply the Normalization Formula:** Use a linear transformation to scale the intensity values from the original range to the desired target range. For example, to normalize pixel values to the range [0, 255], the formula would be: $\boldsymbol{Normalized\ Value} = \frac{(\boldsymbol{Original\ Value - Min\ Value})\ \boldsymbol{x\ 250}}{\boldsymbol{Max\ Value - Min\ Value}}$
3. **Reassign Pixel Values:** Replace each pixel in the original image with its normalized intensity value.

Here are two more images for illustration:

- **Origin:** The first image is a grayscale photograph with a narrow range of intensity values, resulting in low contrast and lack of detail visibility.
- **After Normalization:** The second image is the result of applying intensity normalization to the first image, showcasing improved contrast with a wider range of intensity values, making details in the image more visible and enhancing the overall appearance.