Assignment-2: Image Effect Implementation (Part-2)

Sumaiya Tarannum Noor
Student ID: 2425410650
email: sumaiya.tarannum@nothsouth.edu

February 23, 2025

I. INTRODUCTION

In the modern digital world, images play a vital role in communication, education, and various industries, making Digital Image Processing (DIP) an indispensable technology. We can say, Digital image processing is the use of a digital computer to process digital images through an algorithm [1]. DIP involves the application of algorithms to process and analyze digital images, improving their quality, extracting meaningful information, and enabling automation of visual tasks. It enhances image contrast, sharpens details, reduces noise, and restores lost information, making images clearer and more interpretable. Moreover, DIP is a crucial component of artificial intelligence and computer vision, as it allows machines to recognize patterns, detect objects, and make automated decisions with high precision. Dewangan et el in [9] says, Digital image processing is a highly popular and fast-evolving field within computer science engineering. Its expansion is driven by advancements in digital imaging technology, computer processing, and data storage devices. Industries that once relied on analog imaging are now transitioning to digital systems due to their flexibility and cost-effectiveness. Key sectors benefiting from this shift include medicine, video production, photography, remote sensing, and security surveillance.

The importance of DIP extends across multiple domains, offering benefits that improve efficiency, security, and visual representation. In the medical field, it is used in MRI scans, X-rays, and CT imaging to enhance diagnostics and detect diseases. Security systems employ DIP for facial recognition, surveillance, and biometric authentication. In the automotive industry, particularly in self-driving cars, image processing ensures safe navigation by detecting obstacles and road signs. Social media platforms leverage DIP for automatic photo enhancement, object tagging, and filter application. Furthermore, satellite imaging and remote sensing utilize DIP to monitor environmental changes, forecast weather patterns, and aid in disaster management. With the growing reliance on digital visuals, DIP continues to revolutionize how we interact with and interpret images.Several image processing techniques can be applied to enhance, restore, and transform images for different purposes. These techniques not only improve visual quality but also add artistic and functional value to images. Below are four important image effects along with their implementation and applications.

Histogram Equalization is a technique that enhances the contrast of an image by redistributing pixel intensity values more evenly. Abdullah et el in [8], proposed a smart contrast enhancement technique based on conventional histogram equalization (HE) algorithm.

The **Old Image Effect** simulates the appearance of an aged photograph by adding noise, adjusting brightness, and reducing saturation.

The **Sketch Effect** transforms an image into a pencil sketch by emphasizing edges and removing colors.

Lastly, the **Edge Glow Effect** enhances the outlines of objects by detecting edges and adding a glowing effect. Each of these image processing techniques serves unique purposes, demonstrating the versatility and importance of digital image manipulation in various fields, from medical and security applications to artistic and entertainment industries.

METHODOLOGY

This study focuses on implementing and analyzing four distinct image processing techniques: **Histogram Equalization, Old Image Effect, Sketch Effect, and Edge Glow Effect**. Each technique serves a unique purpose in enhancing, restoring, or transforming images for better visualization and artistic effects. The implementation follows a systematic approach, starting with image preprocessing, applying specific transformations, and evaluating the output.

For **Histogram Equalization**, contrast enhancement is achieved by redistributing pixel intensity values to ensure a more uniform brightness distribution. The **Old Image Effect** simulates aging by modifying color tones, adding noise, and introducing blur. The **Sketch Effect** extracts edge details while removing color to create a pencil-drawn appearance. Lastly, the **Edge Glow Effect** enhances object outlines by detecting edges and overlaying them with a glow effect.

Each method is implemented using MATLAB, leveraging image processing libraries to apply filters, transformations, and intensity adjustments. The effectiveness of these techniques is analyzed based on visual improvements and their applicability in various domains, such as medical imaging, photography, and digital art.

We will be discussing all five image effets one by one below:

A. *Histogram Equalization*

Histogram Equalization is a digital image processing technique used to enhance the contrast of an image by redistributing pixel intensity values more evenly. In a typical image, certain intensity ranges may dominate, leading to low contrast and loss of detail. By equalizing the intensity distribution, this technique enhances details in both bright and dark regions, making the image more visually balanced and clearer.

This effect is widely applied in **medical imaging**, where it improves visibility in X-rays and MRI scans, **surveillance systems** to enhance details in low-light footage, and **remote sensing** for better interpretation of satellite images. Additionally, it is beneficial in **computer vision applications**, where it aids in object detection and recognition by enhancing feature visibility.

1) How It Works: **Pixel Intensity Distribution** Each pixel in a grayscale image has an intensity value $I(x, y)$ ranging from **0 (black) to 255 (white)**. In color images, this value is split across three channels: **Red (R), Green (G), and Blue (B)**, each with the same intensity range. Histogram Equalization modifies this distribution to improve contrast.

Computing the Histogram We can get the Histogram Equalization Equation from [3]. A histogram represents the frequency of each intensity level in an image. The probability of a pixel having an intensity i is given by:

$$P(i) = \frac{\text{Number of pixels with intensity } i}{\text{Total number of pixels}} \tag{1}$$

This helps determine how pixel values are distributed across the image.

Cumulative Distribution Function (CDF) The **Cumulative Distribution Function (CDF)** is calculated to transform pixel intensities. It is computed as:

$$CDF(i) = \sum_{j=0}^{i} P(j) \tag{2}$$

The CDF helps in mapping old intensity values to new ones, ensuring a uniform distribution of intensities.

Intensity Mapping and Transformation The new intensity $I'(x, y)$ is obtained by applying the transformation:

$$I'(x, y) = \text{round}\left(\frac{CDF(I(x, y)) - CDF_{\min}}{(M \times N) - CDF_{\min}} \times 255\right) \tag{3}$$

where:

- $CDF_{\min}$ is the minimum nonzero CDF value.
- M and N are the image dimensions.
- $I(x, y)$ is the original pixel intensity.
- $I'(x, y)$ is the new, equalized intensity.

Applying to Color Images For **RGB images**, Histogram Equalization is applied separately to each color channel:

$$R'(x, y) = \text{Equalize}(R(x, y)), \quad G'(x, y) = \text{Equalize}(G(x, y)), \quad B'(x, y) = \text{Equalize}(B(x, y)) \tag{4}$$

This ensures balanced contrast across all channels, improving overall image clarity.

Final Image Output The resulting image has improved contrast, with enhanced visibility in previously dark or bright regions. This technique is widely used in **medical imaging, night vision enhancement, remote sensing, and photography**, where improving image clarity is crucial for analysis and interpretation.

B. Algorithm for Histogram Equalization

Algorithm 1 Histogram Equalization

1: **Input:** Grayscale image I of size $M \times N$
2: **Output:** Equalized image I'
3: Compute histogram H of image I
4: Compute probability distribution $P(i) = \frac{H(i)}{M \times N}$
5: Compute cumulative distribution function (CDF):

$$CDF(i) = \sum_{j=0}^{i} P(j)$$

6: Normalize CDF to obtain transformation function:

$$T(i) = \text{round}\left(\frac{CDF(i) - CDF_{\min}}{(M \times N) - CDF_{\min}} \times 255 \right)$$

7: Apply transformation to obtain equalized image:

$$I'(x, y) = T(I(x, y))$$

8: Return equalized image I'

C. Old Image

An old image often suffers from **degradation** due to factors such as aging, environmental exposure, and improper storage. These images may exhibit **fading, noise, discoloration, and loss of details**, making it difficult to analyze or restore them effectively.

Restoring old images is crucial in fields such as **historical preservation**, where valuable photographs and documents need to be maintained. It is also widely used in **digital archiving** and **forensic analysis**, where enhancing degraded images can reveal hidden details.

1) Common Issues in Old Images: **Fading and Loss of Contrast** Over time, images lose their original contrast and sharpness due to chemical reactions and environmental exposure. This results in a low dynamic range, making details hard to distinguish.

Noise and Grain Old images often contain noise and grain due to the degradation of photographic materials. This can include **salt-and-pepper noise** (random white and black pixels) or **Gaussian noise** (random variations in intensity).

Discoloration and Color Shifts Photographic materials degrade unevenly, leading to unwanted color shifts. For example, an image may develop a yellowish or bluish tint due to the fading of certain dyes.

Tears, Scratches, and Missing Parts Physical damage such as **scratches, tears, and missing sections** further deteriorate the image quality. These defects need digital reconstruction techniques to restore missing details.

2) Restoration Techniques: **Contrast and Brightness Adjustment** Using histogram equalization or adaptive contrast enhancement, the faded image can be **rebalanced** to improve visibility and restore lost details.

Noise Reduction and Smoothing Noise can be reduced using techniques such as **Gaussian blurring, median filtering, and wavelet-based denoising**. These methods help in removing unwanted grain while preserving important image features.

Color Correction and Enhancement Color restoration techniques use **white balance adjustment, color histogram matching, and deep learning models** to correct discoloration and recover original hues.

Image Inpainting and Repair For damaged or missing sections, **image inpainting** techniques such as **patch-based synthesis** and **deep learning-based restoration** help reconstruct missing details seamlessly.

Final Restored Image After applying restoration techniques, the resulting image is clearer, with improved contrast, reduced noise, and corrected colors. This enhances the usability of old photographs for **historical records, personal memories, and forensic investigations**.

D. Algorithm for Old Image Restoration

Algorithm 2 Old Image Restoration

1: **Input:** Degraded Image I.
2: **Output:** Restored Image I_r.
3: **for** each pixel (x, y) in I **do**
4:    Apply noise reduction: $I_n(x, y) = \text{Denoise}(I(x, y))$
5:    Adjust contrast: $I_c(x, y) = \text{Equalize}(I_n(x, y))$
6:    Perform color correction: $I_{cc}(x, y) = \text{CorrectColor}(I_c(x, y))$
7:    Apply inpainting for missing parts: $I_r(x, y) = \text{Inpaint}(I_{cc}(x, y))$
8: **end for**
9: **Return** I_r

E. Sketch Effect

A sketch effect transforms an image into a hand-drawn sketch, mimicking pencil strokes and shading. This effect is widely used in digital art, stylized photography, and creative design, enhancing images by converting them into artistic representations.

1) Characteristics of Sketch Effect: **Edge Detection and Contours** The key feature of a sketch effect is edge detection, where prominent edges and contours are identified. This process helps in distinguishing boundaries and structures within an image, resembling pencil outlines.

Shading and Texture Simulation To achieve a realistic sketch effect, shading techniques such as cross-hatching and intensity variation are applied. These methods simulate the depth and texture of hand-drawn sketches.

Monochrome and Colored Sketches Sketch effects can be monochrome (black and white) or color-based. Monochrome sketches highlight structural details, while colored sketches maintain the original hues in a hand-drawn style.

Noise Reduction and Smoothing To refine the sketch effect, noise reduction techniques such as Gaussian blurring and bilateral filtering are used. These methods help in eliminating unnecessary details while preserving essential features.

2) Sketch Effect Techniques: **Edge Enhancement and Thresholding** Using methods like Canny edge detection and adaptive thresholding, an image's edges are extracted and converted into bold strokes resembling pencil sketches.

Gradient-Based Shading Gradient-based techniques simulate shading variations by adjusting intensity values. This enhances the depth and realism of the sketch effect.

Pencil Stroke Simulation By applying stroke-based filters and texture overlays, an image is transformed into a hand-drawn representation. This includes techniques such as stroke-based rendering and pattern replication.

F. Algorithm for Sketch Effect

Algorithm 3 Sketch Effect Transformation

1: **Input:** Original Image I.
2: **Output:** Sketch Image I_s.
3: Convert image to grayscale: $I_g = \text{Grayscale}(I)$
4: Apply Gaussian blur: $I_b = \text{GaussianBlur}(I_g)$
5: Compute edge map using Canny detection: $I_e = \text{CannyEdges}(I_g)$
6: Apply adaptive thresholding: $I_t = \text{Threshold}(I_e)$
7: Simulate stroke texture: $I_s = \text{StrokeEffect}(I_t)$
8: **Return** Sketch Image I_s

G. Edge Glow Effect

The Edge Glow effect is a digital technique that highlights the edges of an image, producing a glowing or illuminated outline around objects and structures. It enhances the visibility of edges, creating a glowing aura effect that adds depth and intensity to images. This effect is commonly used in artistic photography, graphic design, and visual media to emphasize contours and add a dramatic flair.

1) Attributes of Edge Glow Effect: **Edge Detection and Gradient Calculation** A key feature of the Edge Glow effect is the detection of edges using gradient-based methods. The edges are identified by calculating the first-order derivatives in both horizontal and vertical directions, typically using filters like Sobel operators.

Glow Simulation and Intensity Adjustment To simulate the glow effect, intensity adjustments are made to the detected edges. These adjustments amplify the edges and create a soft, glowing halo around them, giving the image a radiant, illuminated look.

Blurring and Smoothing To enhance the edge glow, Gaussian blur is applied to the gradient image. This blurring process smooths out the edges, blending the glow into the surrounding pixels, giving it a natural and soft appearance.

Final Blending and Output The final edge glow effect is achieved by blending the original image with the blurred gradient image. The blend combines the sharp features of the original image with the soft glow from the edge detection process.

2) Edge Glow Effect Techniques: **Sobel Edge Detection** Sobel operators are used to compute the first-order derivatives of the image in both horizontal (G_x) and vertical (G_y) directions. The edge strength is calculated by averaging the absolute values of these derivatives.

Gaussian Blur for Glow Simulation Gaussian blur is applied to the gradient image to smooth out the sharp edges and create a soft glow effect around the detected edges. This gives the edges a glowing aura that blends seamlessly with the image.

Image Blending The final step involves blending the original image with the blurred gradient image. The blend enhances the edges while maintaining the natural appearance of the original image, with a noticeable glow around the contours.

H. Algorithm for Edge Glow Effect

Algorithm 4 Edge Glow Effect Transformation

1: **Input:** Original Image I.
2: **Output:** Edge Glow Image I_g.
3: Convert image to grayscale: $I_{gray} = \text{Grayscale}(I)$
4: Apply Sobel operator in horizontal direction: $G_x = \text{SobelX}(I_{gray})$
5: Apply Sobel operator in vertical direction: $G_y = \text{SobelY}(I_{gray})$
6: Calculate gradient magnitude: $G = \frac{|G_x| + |G_y|}{2}$
7: Apply Gaussian blur: $G_b = \text{GaussianBlur}(G)$
8: Blend original image with blurred gradient: $I_g = 0.55 \times I + 0.45 \times G_b$
9: **Return** Edge Glow Image I_g

II. Experiment

All of my experiments were performed using Matlab. All four effects have their own methods to be perfected. This section is dedicated to describing the details of the experiments for all four effects.

A. Input

As our experiments are about image effects, input images play a very important part in their performance. The input imges play a key role in making the image effects useful. An input image can show some importantaspects of using any certain image effects. A perfect input image can make the use of a certain image effect more useful, purposeful and meaningful.

A perfect input image can make an image effect look more purposeful and meaningful. Image processing plays very important roles in our daily life. In this experiment, we are performing the use of four image effects, that are: Histogram Equalization, Old Image, Sketch and Edge Glow. All of these effects has their own imporatnt part in our life.

Let us discuss a bit about what are the use of all these effect, what are the perfect inputs for each of them and why that matters for their best performance: Here is a chart showing the different input images for different experiments:

1) Histogram Equalization: Histogram equalization is primarily used to enhance image contrast by redistributing pixel intensity values. This technique is widely applied in medical imaging, such as X-rays, MRIs, and CT scans, to improve visibility and highlight crucial details. It is also used in satellite imaging to enhance terrain and object visibility in remote sensing applications. Additionally, biometric systems and face recognition benefit from histogram equalization to distinguish features more effectively. In night vision and surveillance, this technique helps improve image clarity in low-light conditions.

What is a perfect image for Histogram Equalization: A gray-scale image is a perfect input for this image effect to perform perfecty.

Why it matters: A grayscale image is a perfect input for histogram equalization because it has only one intensity channel, making it easy to compute and adjust pixel values directly. Unlike color images, which require conversion to different color

spaces before applying histogram equalization, grayscale images allow for a straightforward transformation. This technique enhances contrast by redistributing pixel intensities, improving visibility in dark or bright regions. Additionally, since grayscale images contain no color information, histogram equalization does not introduce unnatural color artifacts, ensuring a natural and effective contrast enhancement.

2) Old Image: The old image effect is designed to simulate an aged, vintage appearance by adding sepia tones, noise, and faded colors. It is commonly used in photo restoration to recreate historical photographs or enhance aged images. The film and game industries use this effect to create realistic historical scenes or retro-style visuals. Additionally, social media platforms and photo-editing apps like Instagram and Photoshop provide old image effects to give pictures a nostalgic and artistic look. This effect is also popular in advertising and design to evoke emotions associated with past eras.

What is a perfect image for the Old Image Effect An RGB image with vibrant colors is a perfect input for this image effect to perform effectively.

Why it matters An RGB image is ideal for the old image effect because it contains rich color information, allowing for accurate manipulation of tones and fading effects. Since the effect relies on desaturation, sepia toning, and the addition of noise or grain, starting with a colorful image ensures a more visually striking transformation. Unlike grayscale images, which lack color depth, RGB images enable a smoother and more natural transition into vintage hues, preserving important details while achieving the desired aged appearance.

3) Sketch Effect: The sketch effect transforms images into pencil-drawn sketches, capturing fine details and enhancing contrasts. This effect is commonly used in art and design to create illustrations, concept art, or stylized portraits. It is also utilized in the film and animation industries for character design and concept visualization. Social media platforms and photo-editing apps, such as Instagram and Photoshop, provide sketch filters to give images an artistic, hand-drawn touch. This effect is especially valued for its ability to emphasize intricate details, creating a soft yet expressive look that conveys an artistic style.

What is a perfect image for the Sketch Effect: An RGB image with strong contrasts, light and shadow play, and rich details is a perfect input for the sketch effect.

Why it matters: An RGB image is ideal for the sketch effect because it contains detailed color and tonal information, allowing for a solid foundation in creating high-contrast sketches. Images with rich light and shadow play are essential, as the effect relies on emphasizing fine details and textures. The contrast in RGB images ensures that the pencil-drawn aesthetic remains dynamic and realistic. Grayscale images, in comparison, lack the depth and tonal variety, making RGB images the best choice for achieving a detailed and visually striking sketch effect.

4) Edge Glow Effect: The edge glow effect is used to highlight the edges of objects in an image by applying a glowing or neon-like effect to them. This effect creates a high-contrast, visually striking appearance that draws attention to the outlines and contours of objects. The edge glow effect is commonly used in graphic design, visual art, and advertising to emphasize shapes and add a dynamic, futuristic look. It is also popular in digital art, where glowing edges can be used to enhance the mood or theme of an image.

What is a perfect image for the Edge Glow Effect: Images that contain many visible vertical and horizontal lines are perfect inputs for the edge glow effect.

Why it matters: Images with visible vertical and horizontal lines are ideal for the edge glow effect because these lines create well-defined edges that can be enhanced with the glowing effect. The presence of clear and sharp edges makes it easier to isolate and emphasize contours, ensuring a more striking and pronounced glow. Images with intricate patterns or geometric designs benefit from the effect, as the lines become more prominent and visually captivating. Unlike images with less defined edges, those with prominent lines provide a more powerful and impactful edge glow transformation, making the image stand out with bold, radiant outlines.

TABLE I Input images for each experiment:

No.	Experiment Name	Input Image
1	Histogram Equalization	
2	Old Image Effect	
3	Sketch Effect	
4	Edge Glow Effect	

B. Process of the Experiments

In this section I will discuss the process for all the experiments:

1) Histogram Equalization: **Step 1: Clearing the Environment** - Clear the command window, remove all variables from memory, and close all open figure windows.

Step 2: Reading and Displaying the Image - Read the input image file. - Retrieve the dimensions of the image ($M \times N$). - Display the original image.

Original Image

Fig. 1: Original Image

Step 3: High-Pass Filtering (Edge Detection) - Define a Laplacian kernel filter [6]. - Apply the filter to the image to detect edges using convolution. - Display the edge-detected image.

High-Pass Filter (Edges)

Fig. 2: High-Pass Filter (Edge Detection)

Step 4: Low-Pass Filtering (Blurring) - Define a Gaussian filter of size 5×5 with standard deviation $\sigma = 1$. - Apply the Gaussian filter to blur the image. - Display the blurred image.
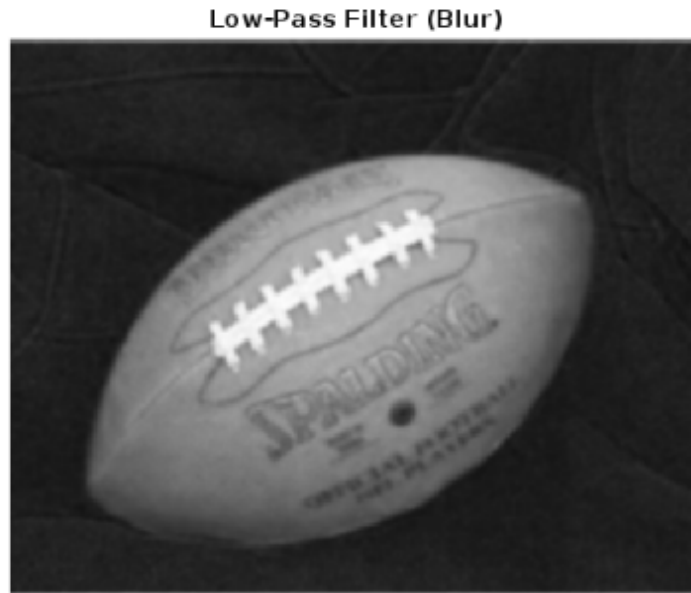
Fig. 3: Low-Pass Filter (Blurred Image)

Step 5: Compute Histogram and Probability Distribution - Define the number of intensity levels ($L = 256$). - Compute the histogram of the grayscale image. - Normalize the histogram to get the probability distribution function (PDF). - Display the histogram of the original image.
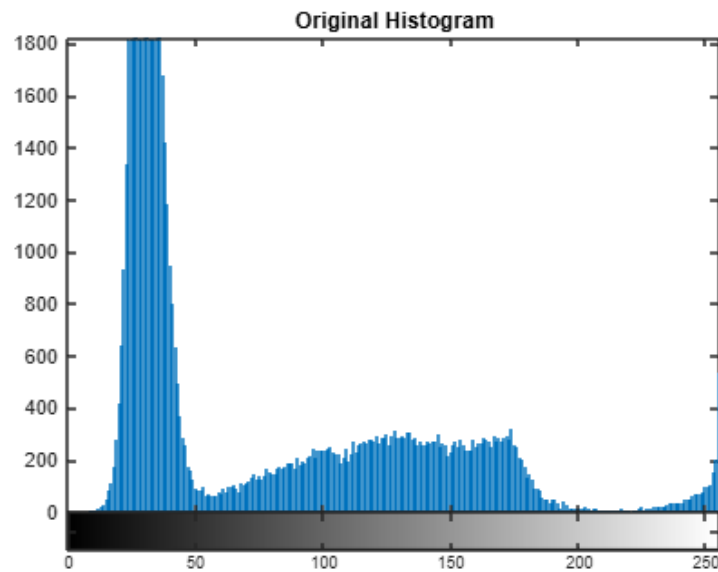
Fig. 4: Original Image Histogram

Step 6: Compute Cumulative Distribution Function (CDF) - Compute the cumulative distribution function (CDF) from the histogram. - Compute the transformation function for histogram equalization using $s_k = (L - 1) \times CDF$.

Step 7: Apply Histogram Equalization - Use the transformation function to map intensity values. - Display the histogram-equalized image.
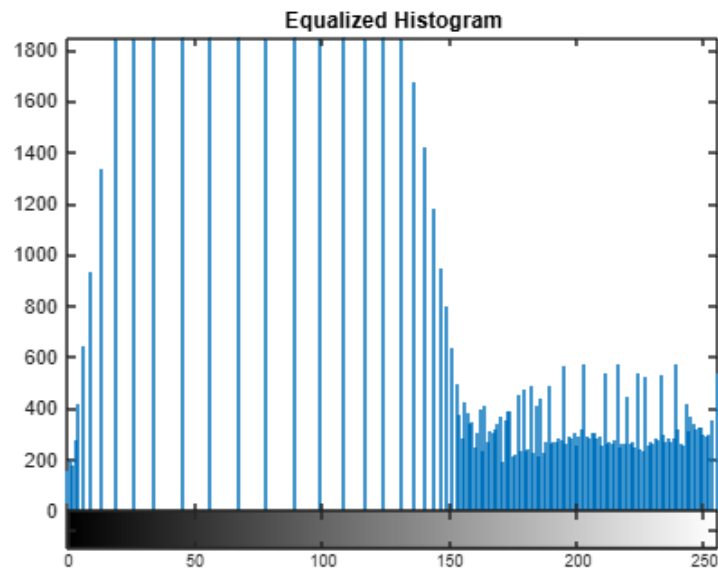
Fig. 5: Histogram Equalized Image

- Display the histogram after equalization.
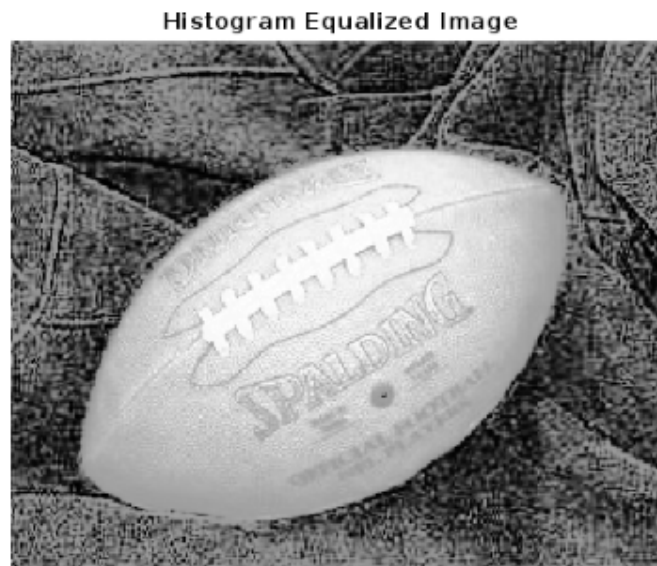
Fig. 6: Equalized Histogram

2) Old Image: **Step 1: Read the Image and Apply Sepia Filter** - Read the input image file. - Retrieve the dimensions of the image ($M \times N$). - Define the sepia filter matrix [7]:

$$S = \begin{bmatrix} 0.393 & 0.769 & 0.189 \\ 0.349 & 0.686 & 0.168 \\ 0.272 & 0.534 & 0.131 \end{bmatrix}$$

- Apply the sepia transformation by computing:

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = S \times \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Convert pixel values to ensure they remain within valid intensity ranges.
Step 2: Display the Original and Sepia Images - Display the original image. - Display the sepia-filtered image.

Fig. 7: Original Image

Fig. 8: Sepia Effect Image

Step 3: Generate and Display Gaussian Noise - Create a Gaussian noise matrix [2] with zero mean and small variance.
- Display the generated noise.

Gaussian Noise

Fig. 9: Gaussian Noise

Step 4: Blend the Noise with the Sepia Image and Show the Result - Convert the sepia image to double precision for blending. - Add the Gaussian noise to simulate an old, degraded appearance. - Ensure pixel values remain within the valid range $[0, 1]$. - Display the sepia image, the noise, and the final "old image" effect.

Old Image Effect

Fig. 10: Final Old Image Effect

3) Sketch: **Step 1: Read and Convert the Image** - Read the input image file. - Convert the image to double precision for accurate computations. - Convert the image to grayscale to remove color information.

Step 2: Apply Gaussian Blur to Reduce Noise - Apply a Gaussian filter with a small standard deviation ($\sigma = 0.5$) to smooth the image. - The blurring helps in noise reduction, improving edge detection.

Step 3: Apply the Sobel Operator for Edge Detection - Define the Sobel operator [4] for detecting edges along the X and Y directions:

$$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = S_x^T$$

- Apply the Sobel filter to detect horizontal and vertical edges. - Compute the gradient magnitude using:

$$\text{edges} = \sqrt{(\text{edgeX})^2 + (\text{edgeY})^2}$$

- Normalize the edge intensities to bring values within the range $[0, 1]$. - Display the detected edges.
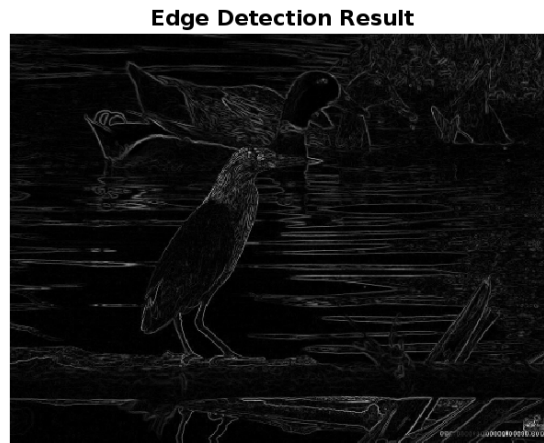
Fig. 11: Edge Detection Result

Step 4: Invert the Edges to Get a Sketch Effect - Invert the edge image to create a white-background effect using:

$$\text{invertedEdges} = 1 - \text{edges}$$

- Display the final sketch effect.
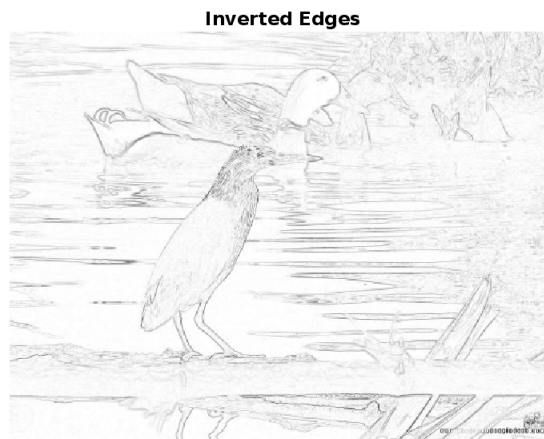
Fig. 12: Inverted Edges (Sketch Effect)

4) Edge Glow: **Step 1: Clearing the Environment** - Clear the command window. - Remove all variables from memory. - Close all open figure windows.

Step 2: Read and Convert the Image - Read the input image file. - Convert the image to grayscale for edge detection.
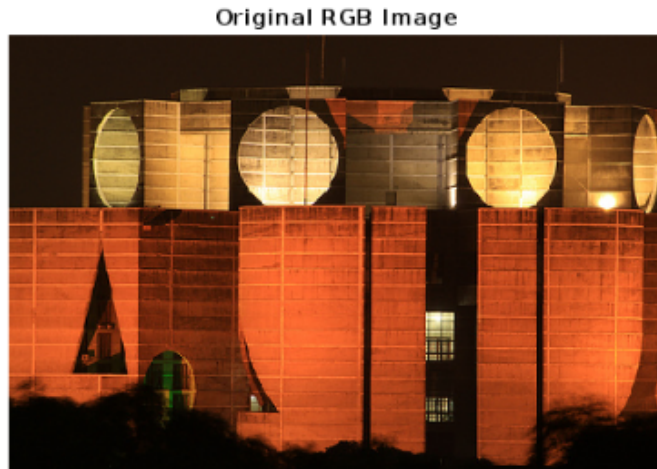
Fig. 13: Original RGB Image
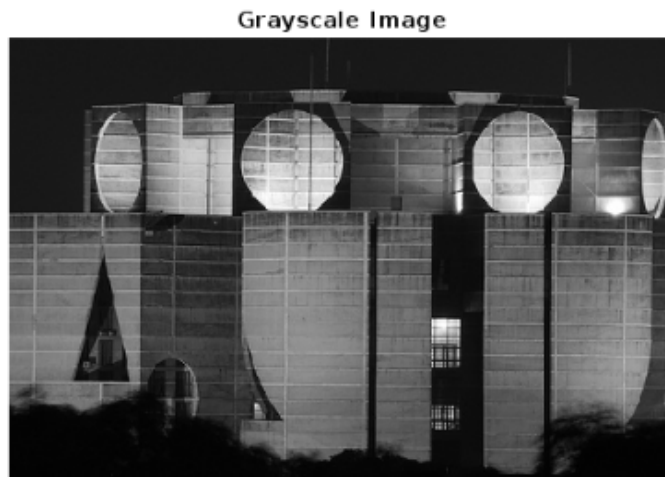
Fig. 14: Grayscale Image

Step 3: Compute First-Order Derivatives Using Sobel Operator - Apply Sobel kernels [4] to compute horizontal (G_x) and vertical (G_y) gradients. - The Sobel operators are:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

- Compute the absolute gradient images and take the average.

Fig. 15: Gx (First-order Derivative)
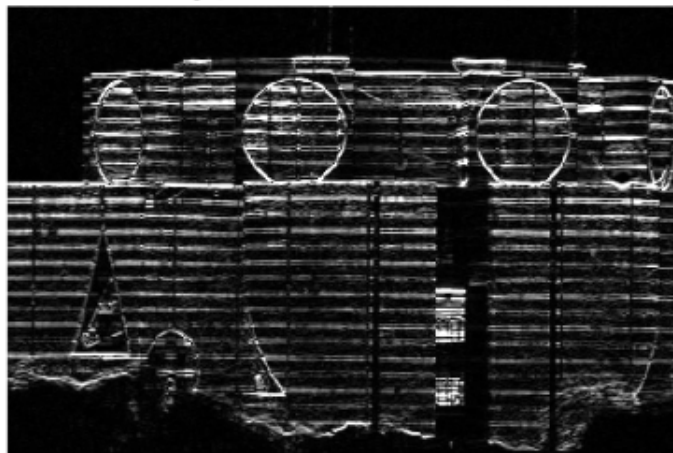
Fig. 16: Gy (First-order Derivative)
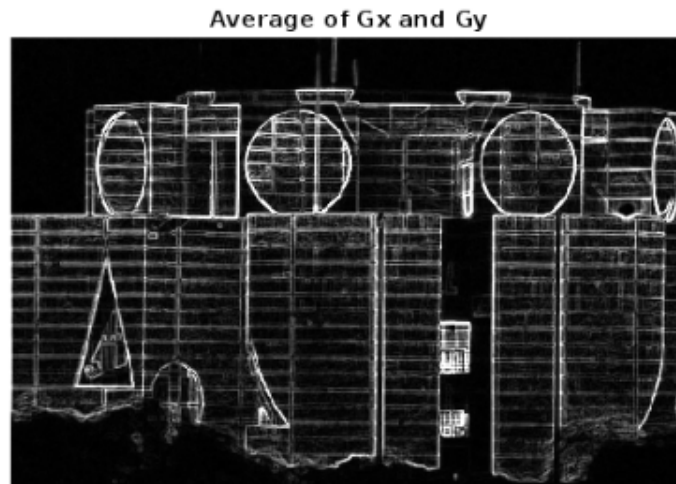
Fig. 17: Average of Gx and Gy

Step 4: Apply Gaussian Blurring - Use a Gaussian filter [5] with a standard deviation of $\sigma = 2$ to blur the gradient image. - This helps in smoothening the edges and enhancing the glow effect.
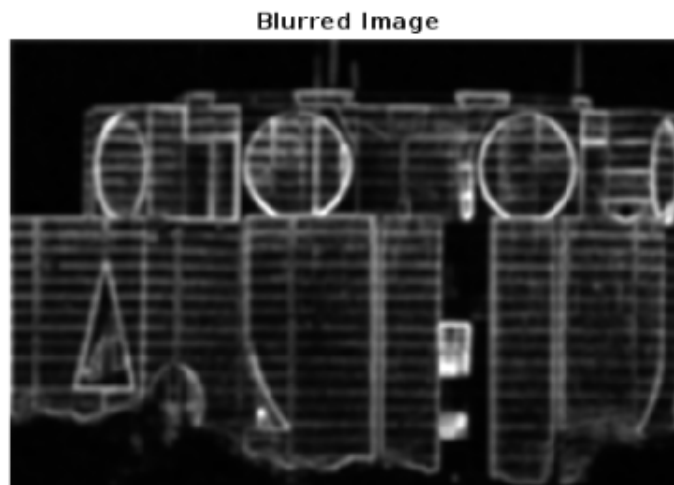
Fig. 18: Blurred Image

Step 5: Blend the Original Image with the Blurred Gradient Image - Compute the final blended output using:

$$\text{Final Output} = 0.55 \times \text{Original Image} + 0.36 \times \text{Blurred Image}$$

- The result enhances the edges and provides a glowing effect.
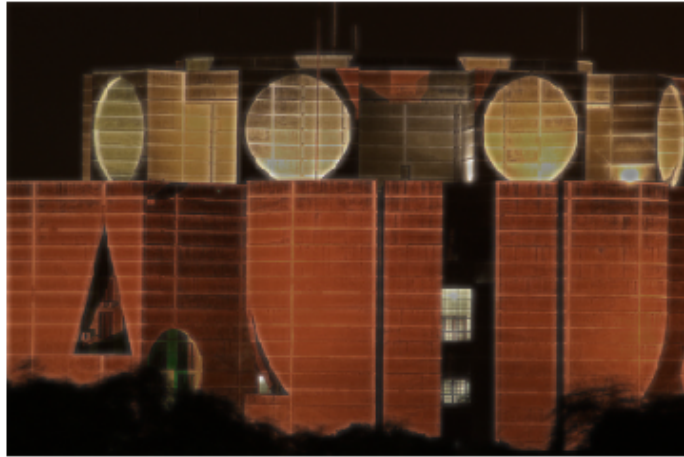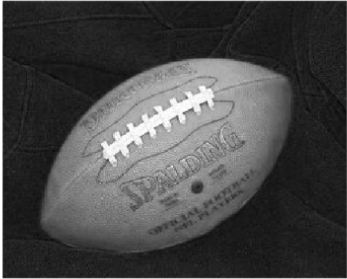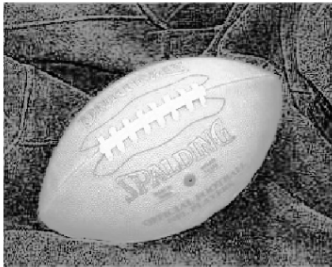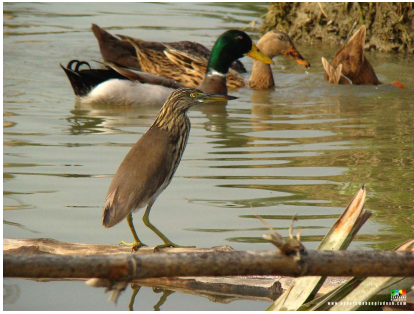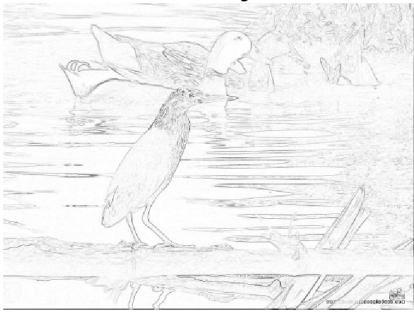
Fig. 19: Final Blended Output

C. Resultant Outputs

    All the resultant outputs along with their inputs are shown in the next table:

TABLE II Image Effects and Their Corresponding Inputs and Outputs

Effect Name	Input Image	Output Image
Histogram Equalization		
Old Image		
Sketch		
Edge Glow		

III. CONCLUSION

Image processing is a part of our daily life. From taking nice photos to medical imaging to gaming, it is used everywhere. Some of the image processing techniques are called image effects. It is important that we use them in the right way for right purpose.

In conclusion, Digital Image Processing (DIP) is a transformative technology that enhances the quality and utility of digital images across multiple domains. By improving image clarity, enabling pattern recognition, and automating visual tasks, DIP plays a critical role in industries such as medicine, security, automotive, and entertainment. The adoption of various image enhancement techniques—such as Histogram Equalization, Old Image Effect, Sketch Effect, and Edge Glow Effect—further illustrates the vast potential of DIP, not only in improving visual quality but also in providing functional value across different sectors. As technology continues to advance, DIP will remain central in shaping how we process, interpret, and interact with digital visuals, driving innovation and efficiency in countless applications.

IV. APPENDIX

Code for Histogram Equalization:

```matlab
clc;
clear;
close all;

% Read the image
img = imread('footBall_orig.jpg');
[M, N] = size(img);

figure, imshow(img), title('Original RGB Image');

% High-Pass Filtering
hp_filter = [0 -1 0; -1 4 -1; 0 -1 0];
edges = imfilter(double(img), hp_filter, 'replicate');

figure, imshow(uint8(edges)), title('High-Pass Filter (Edges)');

% Low-Pass Filtering
lp_filter = fspecial('gaussian', [5 5], 1);
blurred = imfilter(double(img), lp_filter, 'replicate');

figure, imshow(uint8(blurred)), title('Low-Pass Filter (Blur)');

% Compute Histogram and Probability Distribution
L = 256; % Intensity levels (0-255)
histogram = imhist(gray_img);
pr = histogram / (M * N);

figure, imhist(img), title('Original Histogram');

% Compute Cumulative Distribution Function (CDF)
cdf = cumsum(pr);
sk = round((L - 1) * cdf);

equalized_img = sk(double(img) + 1);

figure, imshow(uint8(equalized_img)), title('Histogram Equalized Image');

figure, imhist(uint8(equalized_img)), title('Equalized Histogram');
```

Code for Old Image:

```matlab
% Step 1: Read the image and apply sepia filter
im = imread('flower.jpg');


[rows, cols] = size(im(:,:,1));

% Sepia filter matrix
sepia_filter = [0.393, 0.769, 0.189;
                0.349, 0.686, 0.168;
                0.272, 0.534, 0.131];

sepia_im = zeros(rows, cols, 3);

% Apply the sepia filter using nested loops
for i = 1:rows
    for j = 1:cols
        R = im(i, j, 1);
        G = im(i, j, 2);
        B = im(i, j, 3);

        sepia_im(i, j, 1) = sepia_filter(1, 1) * R + sepia_filter(1, 2) * G +
            sepia_filter(1, 3) * B;
        sepia_im(i, j, 2) = sepia_filter(2, 1) * R + sepia_filter(2, 2) * G +
            sepia_filter(2, 3) * B;
        sepia_im(i, j, 3) = sepia_filter(3, 1) * R + sepia_filter(3, 2) * G +
            sepia_filter(3, 3) * B;
    end
end

sepia_im = uint8(min(sepia_im, 255));

% Step 2: Show the original and sepia images separately
figure(1);
imshow(im);
title('Original Image');

figure(2);
imshow(sepia_im);
title('Sepia Effect');

% Step 3: Create Gaussian noise and show it
noise = imnoise(zeros(size(sepia_im)), 'gaussian', 0, 0.01);

figure(3);
imshow(noise, []);
title('Gaussian Noise');

% Step 4: Blend the sepia image with Gaussian noise
old_im = im2double(sepia_im) + noise;

old_im = min(max(old_im, 0), 1);

% Display the blended image (old image effect) separately
figure(4);
imshow(old_im);
title('Old Image Effect');
```

Code for Sketch:

```
1
2  im = imread('sketch_in.jpg');
3  im_double = im2double(im);
4  im_gray = rgb2gray(im_double);
5  % Blurring to reduce noise
6  im_blur = imgaussfilt(im_gray, 0.5);
7  % Step 1: Apply the Sobel operator for edge detection
8  sobelX = [-1 0 1; -2 0 2; -1 0 1];
9  sobelY = sobelX';
10 % Apply the Sobel operator to get edges
11 edgeX = conv2(im_blur, sobelX, 'same');
12 edgeY = conv2(im_blur, sobelY, 'same');
13 % Combine the edges
14 edges = sqrt(edgeX.^2 + edgeY.^2);
15 % Normalize edge intensities
16 edges = edges / max(edges(:));
17 % Display the edge detection result
18 figure(1);
19 imshow(edges), title('Edge Detection Result');
20 % Step 2: Invert to get white background
21 invertedEdges = 1 - edges;
22 % Display the inverted edges
23 figure(2);
24 imshow(invertedEdges), title('Inverted Edges');
```

Code for Edge Glow:

```matlab
clc; clear; close all;

img = imread('Edge_Glow_in.jpg');
gray_img = rgb2gray(img);

figure, imshow(img), title('Original RGB Image');
figure, imshow(gray_img), title('Grayscale Image');

% First-order derivatives using Sobel operators
gx = imfilter(double(gray_img), [-1 0 1; -2 0 2; -1 0 1], 'same');
gy = imfilter(double(gray_img), [-1 -2 -1; 0 0 0; 1 2 1], 'same');

% Display Gx and Gy
figure, imshow(uint8(abs(gx))), title('Gx (First-order Derivative)');
figure, imshow(uint8(abs(gy))), title('Gy (First-order Derivative)');

gradient_avg = (abs(gx) + abs(gy)) / 2;

% Display gradient average image
figure, imshow(uint8(gradient_avg)), title('Average of Gx and Gy');

% Apply Gaussian blur
blurred_img = imgaussfilt(gradient_avg, 2);

% Display the blurred image
figure, imshow(uint8(blurred_img)), title('Blurred Image');

% Blend the input image and the blurred image
final_output = uint8(0.55 * double(img) + 0.36 * blurred_img);

% Display the final output
figure, imshow(final_output), title('Final Blended Output');
```

REFERENCES

[1] Digital image processing - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Digital_image_processing. [Accessed 23-02-2025].
[2] Gaussian Noise - GeeksforGeeks — geeksforgeeks.org. https://www.geeksforgeeks.org/gaussian-noise/. [Accessed 23-02-2025].
[3] Histogram equalization - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Histogram_equalization. [Accessed 22-02-2025].
[4] Sobel operator - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Sobel_operator. [Accessed 23-02-2025].
[5] Spatial Filters - Gaussian Smoothing — homepages.inf.ed.ac.uk. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm. [Accessed 23-02-2025].
[6] Spatial Filters - Laplacian/Laplacian of Gaussian — homepages.inf.ed.ac.uk. https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm. [Accessed 23-02-2025].
[7] SVG Sepia filter effect: mid-tones only — stackoverflow.com. https://stackoverflow.com/questions/41986812/svg-sepia-filter-effect-mid-tones-only. [Accessed 23-02-2025].
[8] M. Abdullah-Al-Wadud, M. H. Kabir, M. A. A. Dewan, and O. Chae. A dynamic histogram equalization for image contrast enhancement. *IEEE transactions on consumer electronics*, 53(2):593–600, 2007.
[9] S. K. Dewangan. Importance & applications of digital image processing. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 7(7):316–320, 2016.