

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

Sumaiya Tarannum Noor

Student ID: 2425410650

email: sumaiya.tarannum@nothsouth.edu

February 22, 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. 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 al in [2] 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.

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 effects 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 [1]. A histogram represents the frequency of each intensity level in an image. The probability of a pixel having an intensity i is given by:

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

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:

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

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:

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

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:

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

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 extbfdegradation due to factors such as aging, environmental exposure, and improper storage. These images may exhibit extbffading, noise, discoloration, and loss of details, making it difficult to analyze or restore them effectively.

Restoring old images is crucial in fields such as extbfhistorical preservation, where valuable photographs and documents need to be maintained. It is also widely used in extbfdigital archiving and extbfforensic 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 extbfsalt-and-pepper noise (random white and black pixels) or extbfGaussian 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 extbfscratches, 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 extbfrebalanced to improve visibility and restore lost details.

Noise Reduction and Smoothing Noise can be reduced using techniques such as extbfGaussian 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 extbfwhite 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, extbfimage inpainting techniques such as extbfpatch-based synthesis and extbfdeep learning-based restoration help reconstruct missing details seamlessly.

3) *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 extbfhistorical records, personal memories, and forensic investigations.

4) *Algorithm for Old Image Restoration:*

D. 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:*

2) *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.

3) *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.

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  $I_r$ 

```

4) *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.

5) *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.

6) *Sketch Effect Techniques:*

7) *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.

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

9) *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.

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  $I_s$ 

```

10) *Algorithm for Sketch Effect:*

E. 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) *Characteristics of Edge Glow Effect:*

2) *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.

3) *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.

4) *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.

5) *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.

6) *Edge Glow Effect Techniques:*

7) *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.

8) *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.

9) *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.

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
-

10) *Algorithm for Edge Glow Effect*:

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 describe the details of the experiments for all four effects.

A. Input

Here is a chart for showing the different input images for different experiments:

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.

IV. APPENDIX

Code for Posterize:

```

1  % Read the image
2  im = imread('MY_IMAGE.jpg');
3
4  % Convert the image to double
5  im_double = im2double(im);
6
7  % Define the number of Colours for posterization
8  colours = 4;
9
10 % Quantize the image
11 posterized_im = round(im_double * (colours - 1)) / (colours - 1);

```

```
12
13 % Convert back to uint8 for display
14 posterized_im_uint8 = im2uint8(posterized_im);
15
16 % Display the original and posterized images
17 figure;
18 subplot(1, 2, 1);
19 imshow(im);
20 title('Original Image');
21
22 subplot(1, 2, 2);
23 imshow(posterized_im_uint8);
24 title(['Posterized Image']);
```




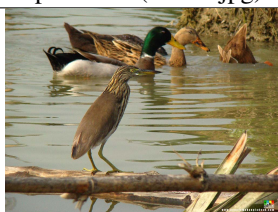
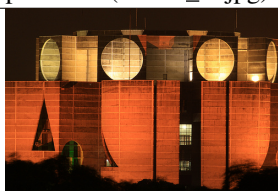
No.	Experiment Name	Image
1	Histogram Equalization	 <p>Fig. 1: Histogram Equalization Experiment (foot-Ball_orig.jpg)</p>
2	Old Image Effect	 <p>Fig. 2: Old Image Effect Experiment (MY_IMAGE.jpg)</p>  <p>Fig. 3: Old Image Effect Experiment (flower.jpg)</p>
4	Sketch Effect	 <p>Fig. 4: Sketch Effect Experiment (sketch_in.jpg)</p>
5	Edge Glow Effect	 <p>Fig. 5: Edge Glow Effect Experiment (Edge_Glow_in.jpg)</p>

TABLE I: Summary of Image Processing Experiments

Code for Nightvision:

```
1  % Read the image
2  im = imread('MY_IMAGE.jpg');
3
4  % Split the image into R, G, B channels
5  R = im(:,:,1);
6  G = im(:,:,2);
7  B = im(:,:,3);
8
9  % Apply the transformations
10 % Output R = Input G / 2
11 R_out = G / 2;
12
13 % Output B = 2 x Output R
14 B_out = 2 * R_out;
15
16 % Output G = 2 x Output B
17 G_out = 2 * B_out;
18
19 % Combine the channels into a new image
20 nightvision_im = cat(3, R_out, G_out, B_out);
21
22 % Convert the image back to uint8
23 nightvision_im_uint8 = im2uint8(nightvision_im);
24
25 % Display the original and nightvision images
26 figure;
27 subplot(1, 2, 1);
28 imshow(im);
29 title('Original Image');
30
31 subplot(1, 2, 2);
32 imshow(nightvision_im_uint8);
33 title('Nightvision Effect');
```


Code for Photocopy :

```

1  % Read the image
2  im = imread('MY_IMAGE.jpg');
3
4  % Convert the image to grayscale
5  gray_im = rgb2gray(im);
6
7  % Define the threshold
8  threshold = 100;
9
10 % Initialize the output image
11 output_im = zeros(size(gray_im));
12
13 % Apply the threshold transformation
14 for i = 1:size(gray_im, 1)
15     for j = 1:size(gray_im, 2)
16         if gray_im(i, j) > threshold
17             output_im(i, j) = 255;
18         else
19             output_im(i, j) = gray_im(i, j) * (threshold - gray_im(i, j)) / (
                threshold^2);
20         end
21     end
22 end
23
24 % Convert the output image to uint8 for display
25 output_im_uint8 = uint8(output_im);
26
27 % Display the original, grayscale, and transformed images
28 figure;
29 subplot(1, 3, 1);
30 imshow(im);
31 title('Original Image');
32
33 subplot(1, 3, 2);
34 imshow(gray_im);
35 title('Grayscale Image');
36
37 subplot(1, 3, 3);
38 imshow(output_im_uint8);
39 title('Photocopy Effect');

```

Code for Vignetting :

```

1  % Read the image
2  im = imread('MY_IMAGE.jpg');
3
4  % Get the dimensions of the image
5  [rows, cols, ~] = size(im);
6
7  % Calculate the center of the image
8  center_x = floor(cols / 2);
9  center_y = floor(rows / 2);
10
11 % Calculate the maximum distance (M) from the center to any corner
12 M = sqrt(center_x^2 + center_y^2);
13
14 % Initialize the output image
15 output_im = zeros(size(im), 'like', im);
16
17 % Apply the vignette effect
18 for i = 1:rows
19     for j = 1:cols
20         % Calculate the distance (D) from the current pixel to the center
21         D = sqrt((j - center_x)^2 + (i - center_y)^2);
22
23         % Calculate the darkness weight
24         weight = 1 - D / M;
25
26         % Apply the weight to each channel of the pixel
27         output_im(i, j, :) = weight * double(im(i, j, :));
28     end
29 end
30
31 % Convert the output image back to uint8 for display
32 output_im_uint8 = uint8(output_im);
33
34 % Display the original and vignetted images
35 figure;
36 subplot(1, 2, 1);
37 imshow(im);
38 title('Original Image');
39
40 subplot(1, 2, 2);
41 imshow(output_im_uint8);
42 title('Vignetting Effect');

```

Code for Mirror :

```
1 % Read the image
2 im = imread('MY_IMAGE.jpg');
3
4 % Reverse the order of the pixels in each row to create a mirror image
5 mirror_im = im(:, end:-1:1, :);
6
7 % Display the original and mirrored images
8 figure;
9 subplot(1, 2, 1);
10 imshow(im);
11 title('Original Image');
12
13 subplot(1, 2, 2);
14 imshow(mirror_im);
15 title('Mirror Image');
```

REFERENCES

- [1] Histogram equalization - Wikipedia — en.wikipedia.org. https://en.wikipedia.org/wiki/Histogram_equalization. [Accessed 22-02-2025].
- [2] S. K. Dewangan. Importance & applications of digital image processing. *International Journal of Computer Science & Engineering Technology (IJCSET)*, 7(7):316–320, 2016.