# Assignment No.1

## Problem Statement

Read an 8 bit image and then apply different image enhancement techniques: (a) Brightness improvement (b) Brightness reduction (c) Thresholding (d) Negative of an image (e) Log transformation (f) Power Law transformation.

## Aim

To study and implement different spatial domain image enhancement techniques on a grayscale image, including brightness adjustment, thresholding, negative, log transformation, and power-law transformation, in order to improve image quality and understand their impact on pixel intensities.

## Objectives

1. To understand the concept of image enhancement in spatial domain.
2. To study brightness manipulation techniques.
3. To implement thresholding to separate object and background.
4. To generate negative images for contrast inversion.
5. To apply log transformation for low-intensity enhancement.
6. To apply power-law (gamma) transformation f+or contrast control.

## Expected Outcomes

1. Students will be able to modify brightness and contrast of an image.
2. Understand thresholding for segmentation.
3. Generate and interpret image negatives.
4. Apply log and gamma transforms for different contrast effects.
5. Compare original vs enhanced results.

## Theory

- $f(x,y)$ → the original pixel intensity at position $(x,y)$ (input image)
- $g(x,y)$ → the processed pixel intensity at position $(x,y)$ (output image after enhancement)
- $C$ → a positive constant used to adjust brightness

## 1. Brightness Improvement

Brightness refers to the overall lightness of an image. Brightness improvement is performed by adding a positive constant to all pixel values, which shifts the intensity histogram to the right. As a result, the image appears lighter and more visible, especially in darker regions. Care must be taken to clip values above 255 to avoid overflow.

**Formula:**

$$s = r + C$$

**Algorithm:**

1. Read the input grayscale image.
2. Select a positive constant *C*.
3. For each pixel, add *C* to its intensity value.
4. If value > 255, clip it to 255.
5. Display and save the enhanced image.

## 2. Brightness Reduction

Brightness reduction is the process of decreasing pixel intensities by subtracting a constant. This shifts the histogram to the left, making the image darker overall. It is useful in reducing over-exposed images or adjusting contrast in very bright scenes. Values lower than 0 are clipped to 0 to avoid underflow.

**Formula:**

$$s = r - C$$

**Algorithm:**

1. Read the input grayscale image.
2. Select a positive constant *C*.
3. For each pixel, subtract *C* from its intensity value.
4. If value < 0, clip it to 0.
5. Display and save the reduced-brightness image.

## 3. Thresholding

Thresholding is a basic segmentation technique that converts a grayscale image into a binary image. It separates objects from the background based on intensity values. Pixels greater than or equal to the threshold are turned white, while others are turned black. It is widely used in applications like document processing and object detection.

**Formula:**

$$s = \{1, \quad if\ r \geq T;\ 0, otherwise\ \}$$

**Algorithm:**

1. Read the input grayscale image.
2. Choose a threshold value *T* (e.g., 128).
3. For each pixel:

- `If value ≥` *T*, set it to 1.
- Otherwise, set it to 0.

## 4. Negative of an Image

Negative transformation produces the photographic negative of an image. It enhances the white details hidden in dark areas and vice versa. This is particularly useful in medical imaging (like X-rays) and photography to highlight subtle details. By subtracting each pixel value from 255, intensities are inverted.

**Formula:**

$$s = L - 1 - r$$

where, intensity level of image has a range [0 , L-1]

**Algorithm:**

1. Read the input grayscale image.
2. For each pixel, subtract its value from 255.
3. Store the result in a new image matrix.
4. Display and save the negative image.

## 5. Logarithmic Transformation

Log transformation is used to expand dark regions while compressing bright regions of an image. This makes low-intensity details more visible, while avoiding saturation of high-intensity pixels. It is especially effective when the dynamic range of the image is very large. The logarithmic curve maps low values strongly but compresses higher values.

**Formula:**

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

Where

c = constant

r >= 0

**Algorithm:**

1. Read the input grayscale image.
2. Find the maximum pixel intensity in the image.
3. Compute the scaling constant c*c*.
4. For each pixel, apply the log transformation formula.

## 6. Power-Law (Gamma) Transformation

Power-law transformation, also known as gamma correction, controls the contrast of an image in a non-linear way. By adjusting the gamma value, we can make the image appear either brighter or darker. This technique is widely used in display devices, photography, and image enhancement pipelines. Gamma < 1 enhances darker regions, while gamma > 1 suppresses them.

**Formula:**

$$s = c * (r)^{\gamma}$$

**Algorithm:**

1. Read the input grayscale image.
2. Normalize all pixel values between 0 and 1.
3. Choose a gamma value γ.
    - If $\gamma < 1$, image brightens.
    - $If\ \gamma > 1$, image darkens.
4. Apply the gamma correction formula.

## Pseudocode:

**Algorithm: Image Enhancement Techniques**

**Input:** 8-bit grayscale image f(x,y)
**Output:** Enhanced images using brightness adjustment, thresholding, negative, log transform, and power-law transform

**Step 1: Read and display the input image**

```
f ← ReadImage("sunset.jpg")
Display(f, title="Original Image")
```

**Step 2: Convert to Grayscale**

```
g ← ConvertToGrayscale(f)
SaveImage(g, "sunset_gray.jpg")
Display(g, title="Gray Image")
```

**Step 3: Brightness Adjustment**

```
bright_up ← g + 50
bright_down ← g - 50
Display(bright_up, "Brightness Increased")
Display(bright_down, "Brightness Decreased")
```

**Step 4: Negative Transformation**

```
For each pixel p in g
    neg(p) ← 255 - p
Display(neg, "Negative Image")
```

**Step 5: Thresholding**

```
For each pixel p in g
    If p > 128 → set to 255
    Else → set to 0
Display(thresh, "Thresholded Image")
```

**Step 6: Log Transformation**

$$c \leftarrow 255 / log(1 + max(g))$$

```
For each pixel p in g
    $log\_trans(p) \leftarrow c * log(1 + p)$
Convert result to 8-bit and Display(log_trans, "Log Transformed Image")
```

**Step 7: Power-Law (Gamma) Transformation**

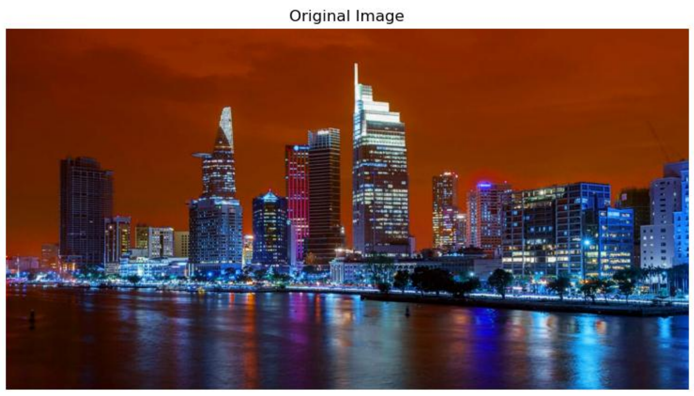
```
γ ← 0.5
For each pixel p in g
    $gamma\_trans(p) \leftarrow 255 * (p/255)^\wedge\gamma$
Display(gamma_trans, "Gamma Corrected Image")
```

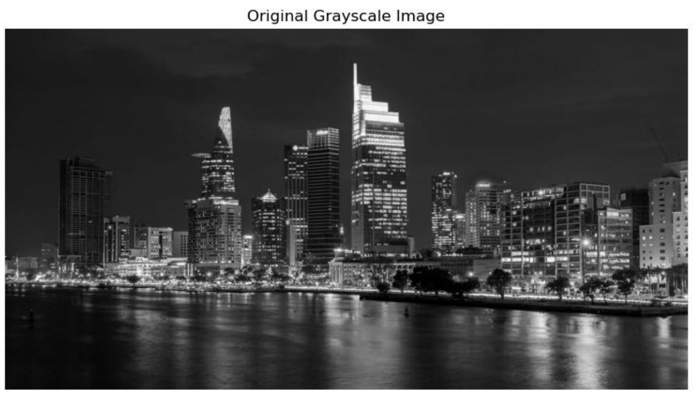
**Step 8: Display Results Together**

Arrange images in grid: Original, Brightness+, Brightness–, Threshold, Negative, Log, Gamma

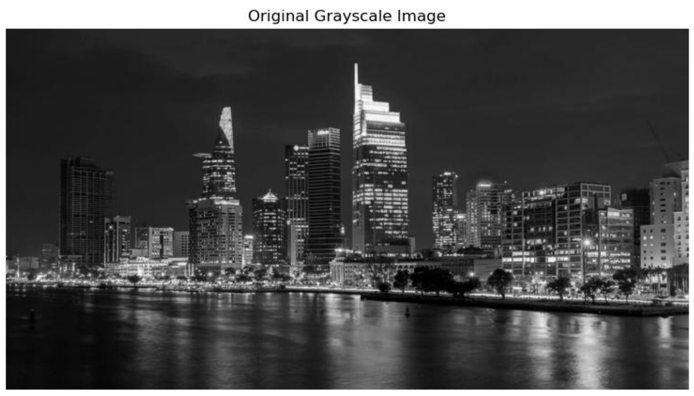
**End Algorithm**

**Results:**





**Original Image** **Grayscale Image**



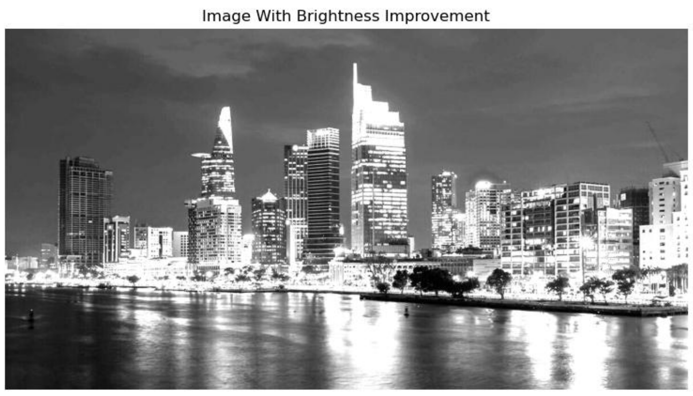
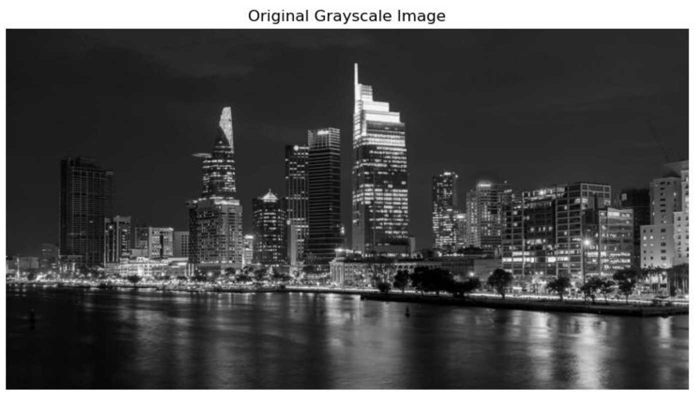


**Image with Enhanced Brightness**



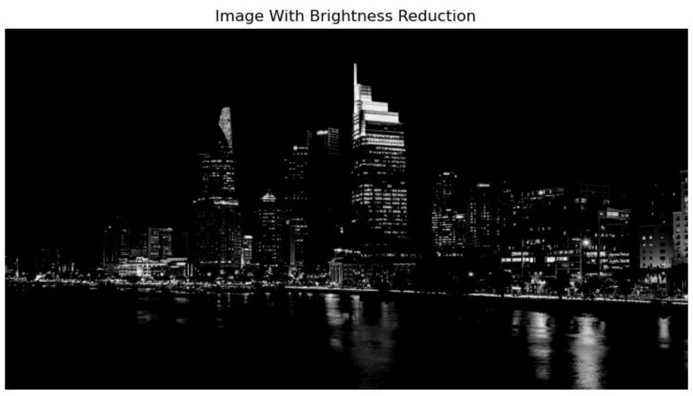


**Image with Reduced Brightness**

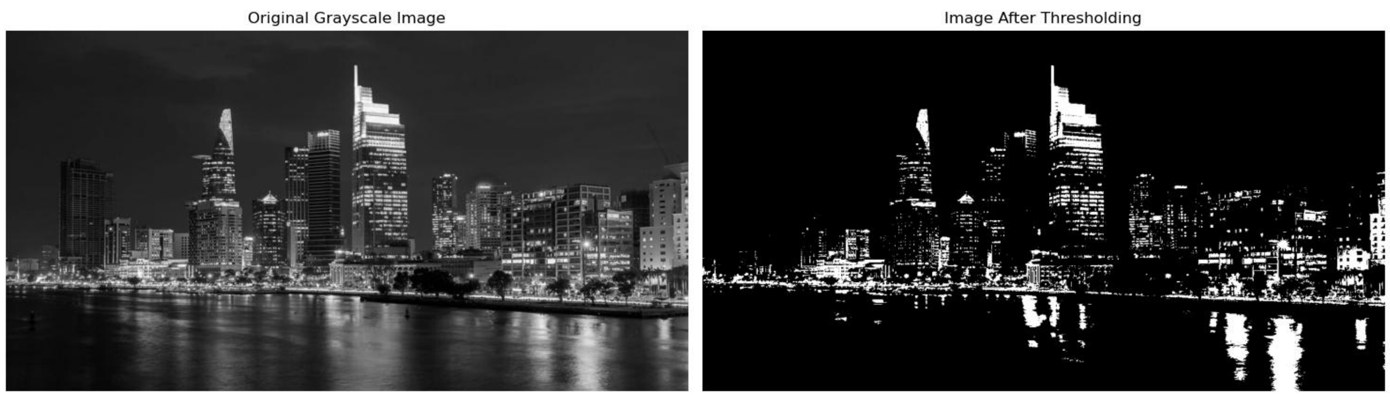


**Image After Thresholding**

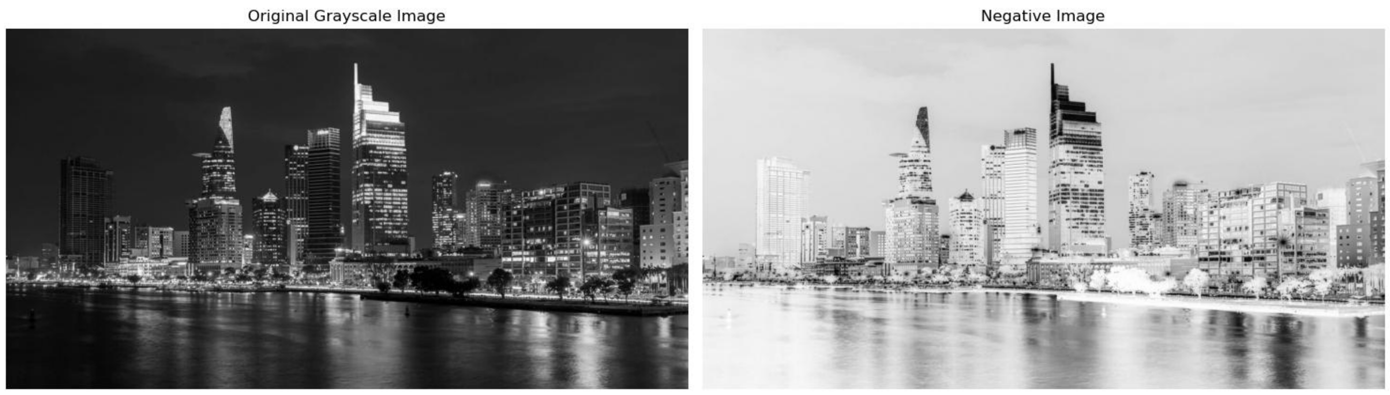


**Image After Negative Operation**

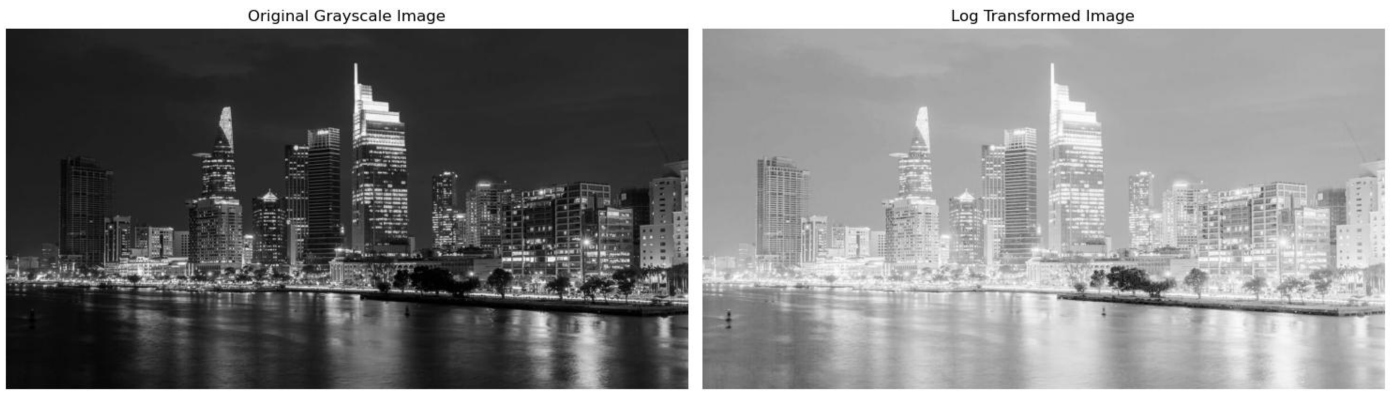


**Image After Log Transformation**

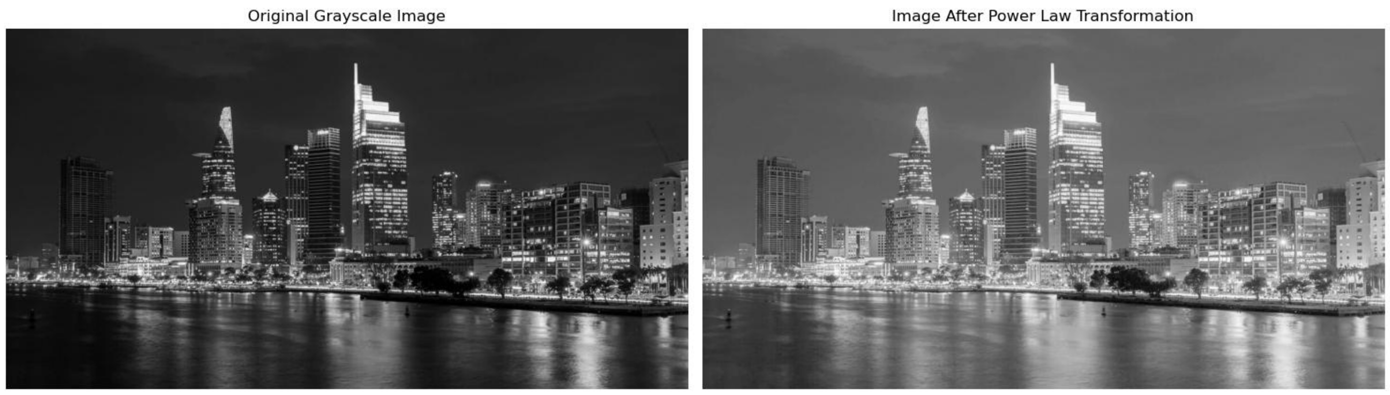
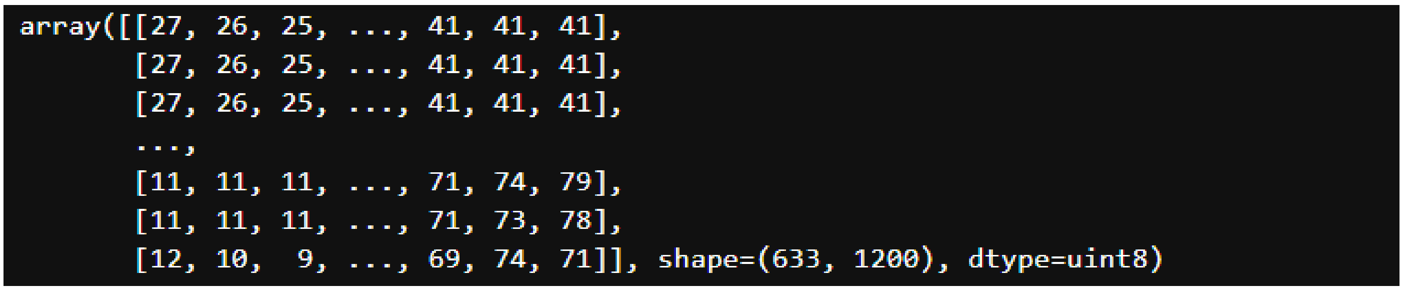


**Image After Power Low Transformation**

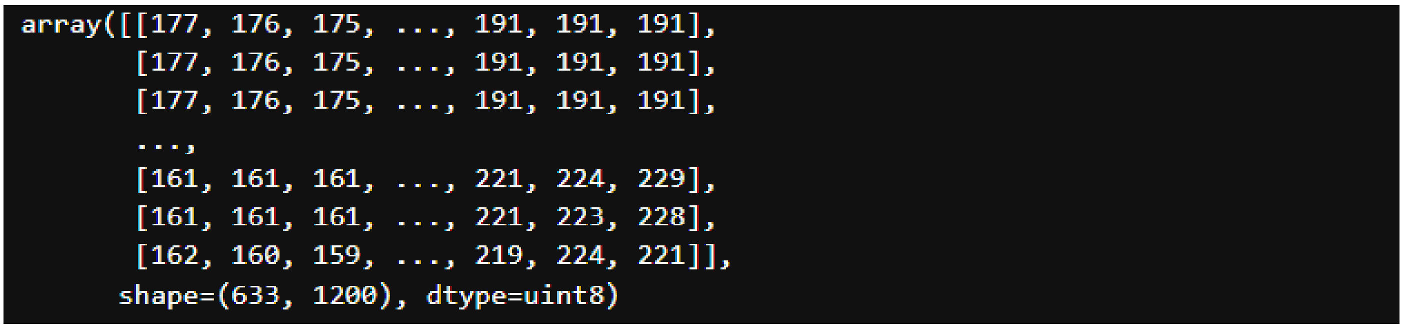
**Pixels:**

**Original Grayscale Image Pixel values:**



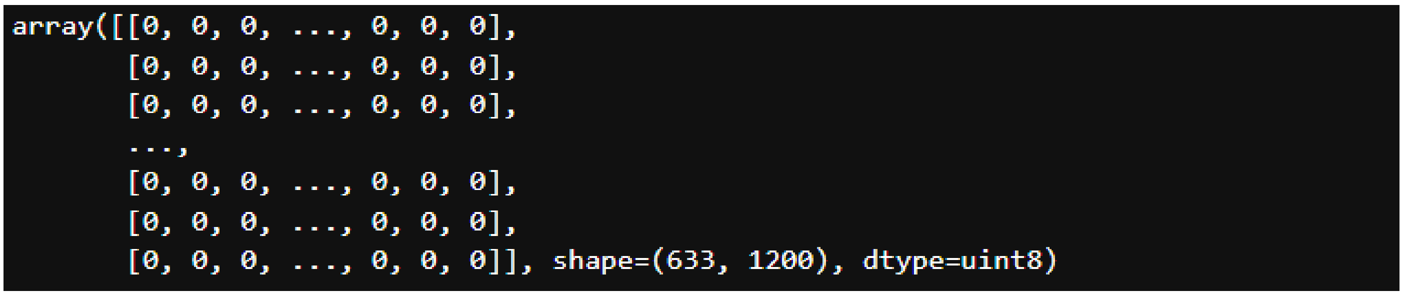
```
array([[27, 26, 25, ..., 41, 41, 41],
       [27, 26, 25, ..., 41, 41, 41],
       [27, 26, 25, ..., 41, 41, 41],
       ...,
       [11, 11, 11, ..., 71, 74, 79],
       [11, 11, 11, ..., 71, 73, 78],
       [12, 10,  9, ..., 69, 74, 71]], shape=(633, 1200), dtype=uint8)
```

**Brightness Increased Image Pixel values:**



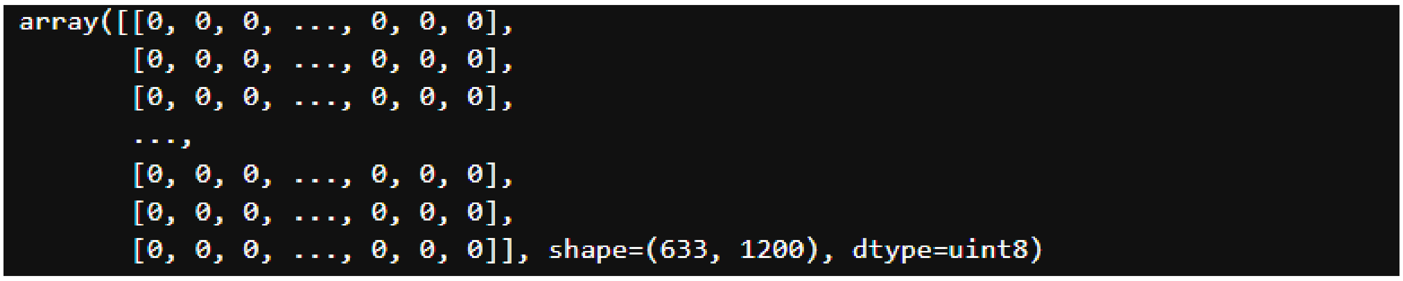
```
array([[177, 176, 175, ..., 191, 191, 191],
       [177, 176, 175, ..., 191, 191, 191],
       [177, 176, 175, ..., 191, 191, 191],
       ...,
       [161, 161, 161, ..., 221, 224, 229],
       [161, 161, 161, ..., 221, 223, 228],
       [162, 160, 159, ..., 219, 224, 221]],
      shape=(633, 1200), dtype=uint8)
```

**Brightness Reduced Image Pixel values:**



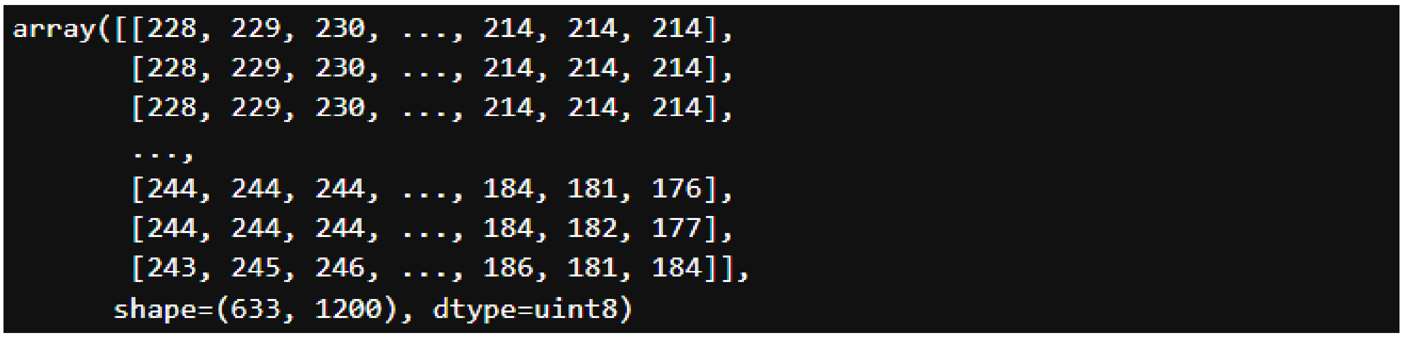
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], shape=(633, 1200), dtype=uint8)
```

**Thresholded pixel values:**



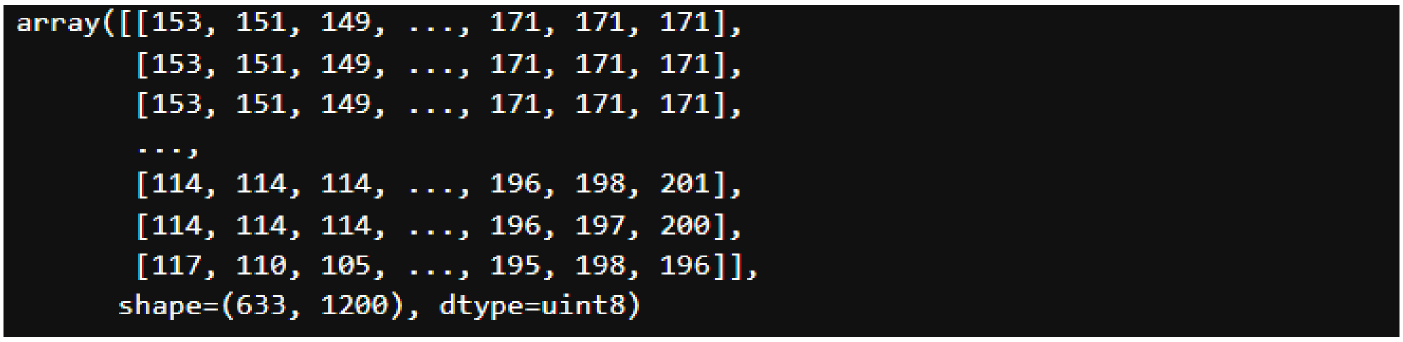
```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]], shape=(633, 1200), dtype=uint8)
```

**Negative Image Pixel Values:**



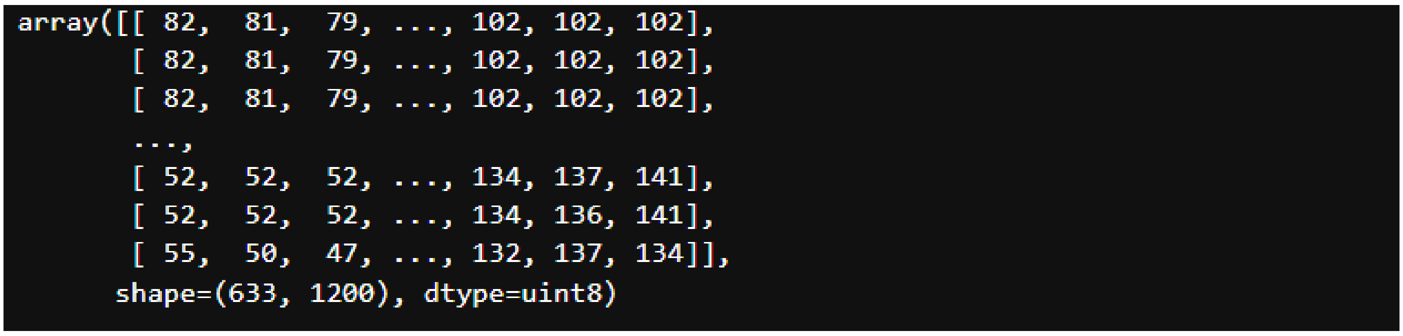
```
array([[228, 229, 230, ..., 214, 214, 214],
       [228, 229, 230, ..., 214, 214, 214],
       [228, 229, 230, ..., 214, 214, 214],
       ...,
       [244, 244, 244, ..., 184, 181, 176],
       [244, 244, 244, ..., 184, 182, 177],
       [243, 245, 246, ..., 186, 181, 184]],
      shape=(633, 1200), dtype=uint8)
```

**Log Transformed Image Pixel Values:**



```
array([[153, 151, 149, ..., 171, 171, 171],
       [153, 151, 149, ..., 171, 171, 171],
       [153, 151, 149, ..., 171, 171, 171],
       ...,
       [114, 114, 114, ..., 196, 198, 201],
       [114, 114, 114, ..., 196, 197, 200],
       [117, 110, 105, ..., 195, 198, 196]],
      shape=(633, 1200), dtype=uint8)
```

**Power Law Transformed Image Pixel Values:**



```
array([[ 82,  81,  79, ..., 102, 102, 102],
       [ 82,  81,  79, ..., 102, 102, 102],
       [ 82,  81,  79, ..., 102, 102, 102],
       ...,
       [ 52,  52,  52, ..., 134, 137, 141],
       [ 52,  52,  52, ..., 134, 136, 141],
       [ 55,  50,  47, ..., 132, 137, 134]],
      shape=(633, 1200), dtype=uint8)
```

## Conclusion

Through this experiment, we got to see how simple mathematical operations can completely change the way an image looks. Increasing or reducing brightness made the picture lighter or darker, while thresholding turned it into a clean black-and-white version. The negative transformation flipped the tones, which is really useful for highlighting hidden details. The log transformation brought out information in darker regions, and gamma correction showed how contrast can be fine-tuned just by adjusting the gamma value. Overall, this assignment gave a clear understanding of how different enhancement techniques help in improving the visibility and quality of images for real world applications.