دانشگاه بوعلی سینا

# Computer vision

## Exercise 1

**By:**

Zahra Sharafian

Professor Hassan Khotanlou

Fall 2023

# TABLE OF CONTENTS

# Question 1

## a)

**Pixel replication:** is also known as nearest neighbor interpolation because using it we can only replicate the neighboring pixels. In the pixel replication method, new pixels are produced through the existing pixels or already given pixels of the input image. Every pixel of the input picture is replicated n times row-wise and n times column-wise, and the input image gets zoomed.

**Zero-order hold:** is also known as zoom twice because we can only zoom twice by this method. We pick two adjacent elements from a row in the zero-order hold method. We then add these elements. After the addition, we divide the result by two. Then the resulting number is placed between the adjacent elements we picked earlier.

## b)

Pixel replication: We import the OpenCV library, which provides functions for image processing and computer vision tasks. We load the image 'einstein.jpg', retrieve its dimensions, and calculate the new dimensions after zooming. The image is then resized using the pixel replication method, creating a zoomed image. The code displays the zoomed and original images in separate windows.

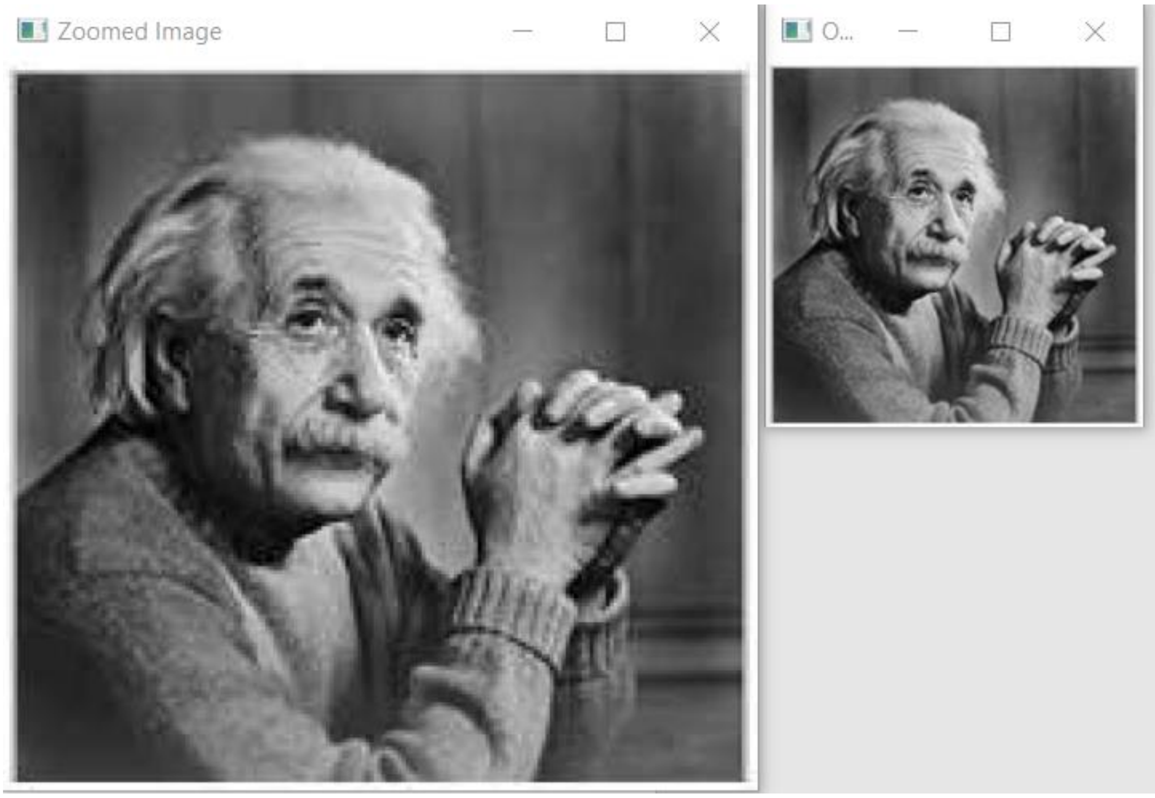


*Figure 1 :Zoomed image and original image using pixel replication*

Zero-order hold: We load the image 'einstein.jpg' using the cv2.imread function and store it in the variable image then retrieve the height and width of the image using the shape attribute of the image array. We calculate the new dimensions of the image after zooming and use the cv2.resize function to resize the image to the new dimensions using the zero-order hold method. Then we display the zoomed and original images using the cv2.imshow function in separate windows.

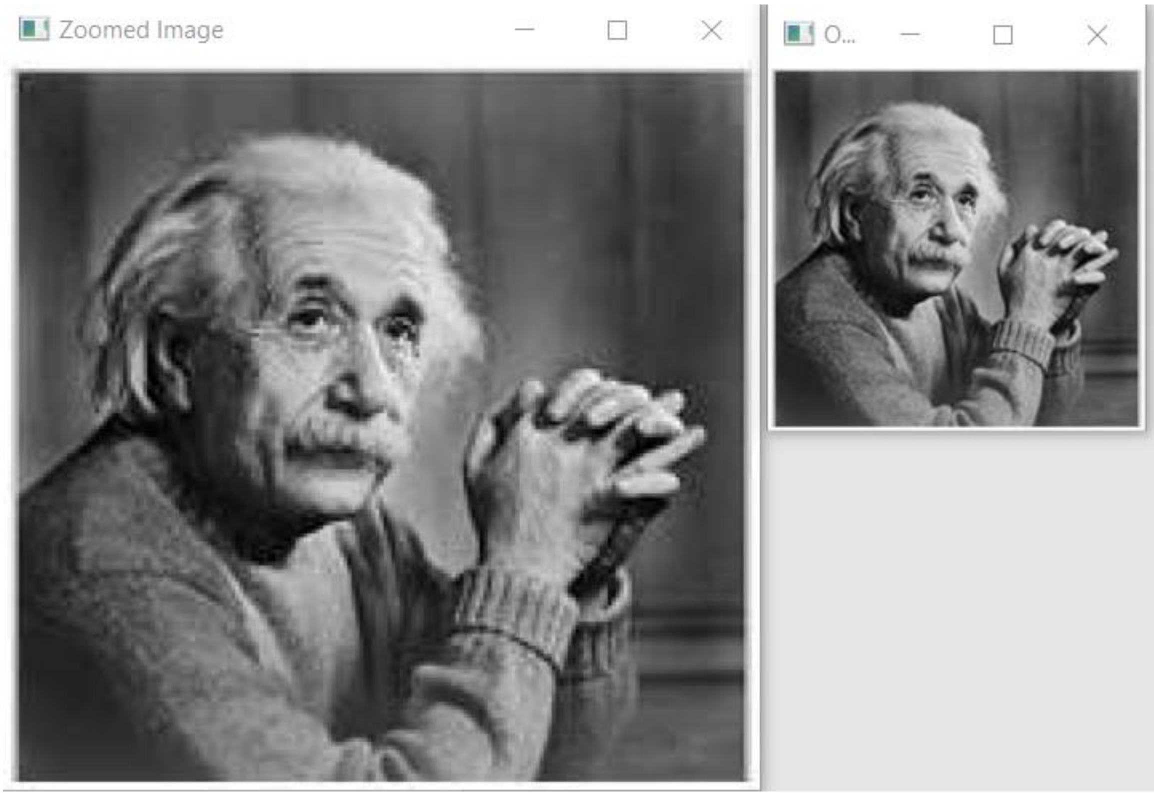


*Figure 22: Zoomed image and original image using zero-order hold*

## c)

Pixel replication method:

a) Advantages: The pixel replication method is computationally efficient since it involves duplicating pixels from the original image to create the zoomed image. It does not require complex calculations or interpolation algorithms.
b) Disadvantages: The pixel replication method can result in a loss of image quality, especially when zooming in. The zoomed image may appear blocky or pixelated due to the duplication of pixels.

Zero-order hold method:

a) Advantages: The zero-order hold method can preserve sharp edges and fine details in the image, resulting in better image quality compared to some other interpolation methods. It avoids blurring and smoothening of details.
b) Disadvantages: The zero-order hold method can introduce aliasing artifacts in the zoomed image, particularly if the zoom factor is large. These artifacts appear as jagged edges or staircase-like patterns around high-frequency areas in the image.

- The pixel replication method is suitable for certain types of images where preserving fine details or smoothness is not critical. It can be used for simple graphics, icons, or images with large solid regions. However, it is not recommended for images with intricate textures, gradients, or complex details, as it may lead to a noticeable loss of quality and visual artifacts.

- The zero-order hold method is suitable for images with sharp edges, fine details, and high-frequency components. It can work well for certain types of graphics, line art, or images with distinct boundaries. However, it may not be the best choice for images with complex textures or smooth gradients.

## d)

When increasing the zoom factor in both the pixel replication and zero-order hold methods, certain effects can be observed in terms of image quality and detail preservation.

In pixel replication method As the zoom factor increases, the pixel replication method can result in a more noticeable loss of image quality. The image may appear more blocky and pixelated due to the repeated duplication of pixels. The lack of interpolation and the limited number of unique pixels can lead to a significant degradation in visual fidelity. With higher zoom factors, the pixel replication method struggles to preserve fine details in the image. The limited number of unique pixels available for replication can cause the loss of subtle textures, gradients and details.

The zero-order hold method tends to preserve sharp edges and fine details reasonably well, even with increasing zoom factors. However, it's important to note that the method's effectiveness in detail preservation diminishes as the zoom factor becomes larger. Complex textures and subtle details may still suffer from some degree of blurring or distortion.


*Figure3 : zero-order hold (zoom factor of 4)*

# e)

An improved zooming method that can provide better image quality and detail preservation compared to pixel replication and zero-order hold is the bilinear interpolation method. Bilinear Interpolation is a simple interpolation technique in which we fill the gaps between pixels using the neighbor pixels.

# Question 2

## a)

for normalizing the image first we read 'low_contrast.jpg'. To enhance the image, the pixel values are normalized to a range of 0 to 1. This ensures that the image has consistent brightness levels. Then, a figure with two subplots is created. In the first subplot, the original image is displayed. The second subplot showcases the normalized image

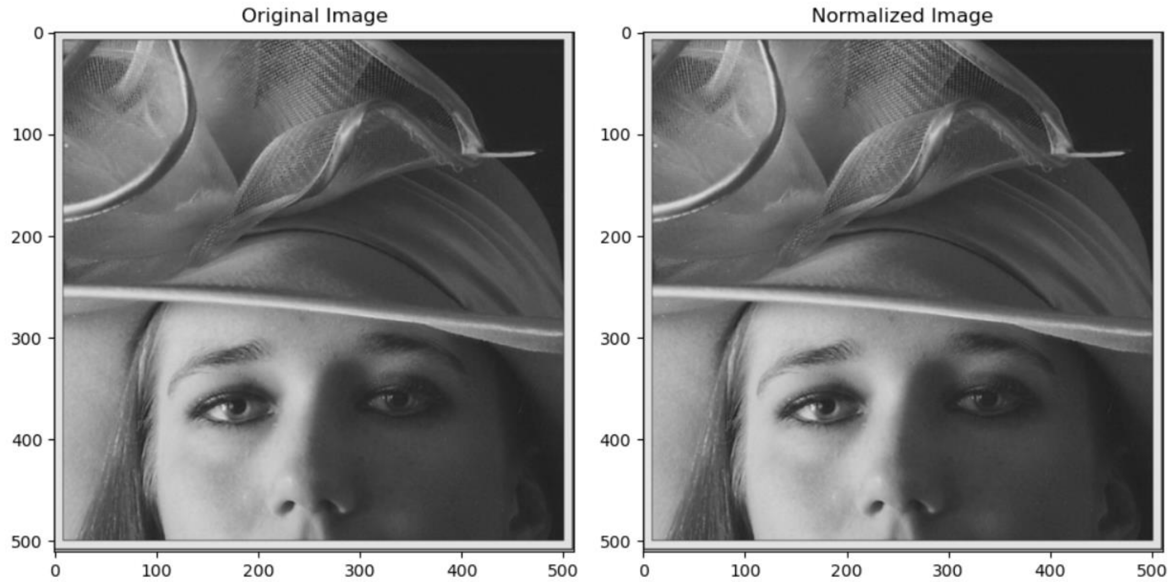


*Figure 4: original and normalized image*

## b)

1. Logarithmic Correction: Logarithmic correction is a technique used to enhance the contrast of an image by applying a logarithmic function to the pixel values. The goal of logarithmic correction is to redistribute the pixel values in such a way that the darker regions of the image are expanded more than the brighter regions, resulting in an increased overall contrast.
2. Gamma Correction: Gamma correction is a technique used to adjust the relationship between the pixel values of an image and the displayed brightness levels. It is based on the power-law function, where each pixel value is transformed by raising it to a power.

## c)

first we read the 'low_contrast.jpg' image in grayscale. Logarithmic contrast stretching is applied to the image using a formula that involves a constant 'c' and the logarithm function. The constant 'c' is chosen to ensure that the enhanced pixel values are scaled to the range of 0 to 1.

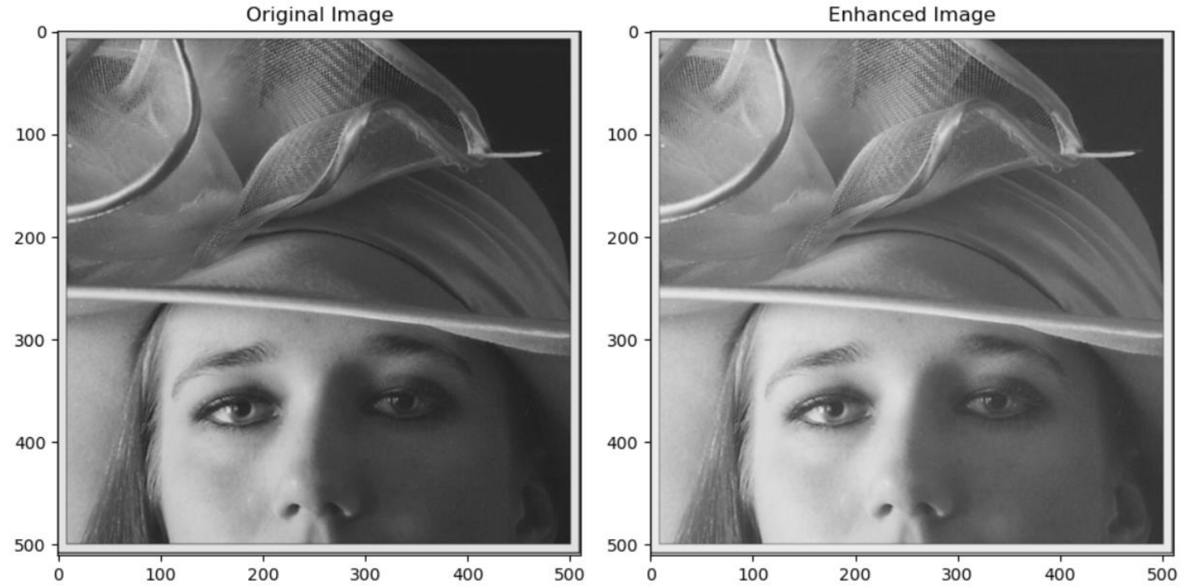


*Figure 5: original and enhanced image using Logarithmic correction*

## d)

first the 'low_contrast.jpg' image is read in grayscale. Gamma correction is then applied to the image, using a chosen 'gamma' value to control the intensity of the correction. A higher 'gamma' value brightens the image, while a lower value darkens it.

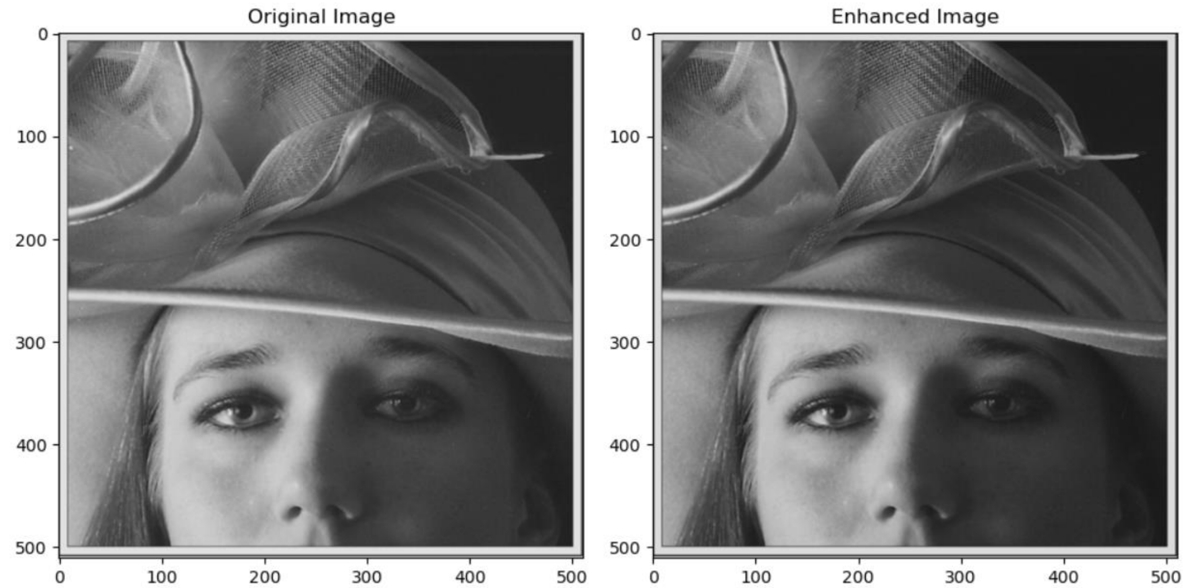


*Figure 6: original and enhanced image using Gamma correction*

# Question 3

## a)

first the 'lena_gray.gif' image is opened and converted to grayscale using the PIL library. The grayscale image is then converted to a NumPy array. The pixel values of the image array are normalized to be within the range of 0 to 1. The code creates a figure with two subplots: one for displaying the original image and the other for displaying the normalized image.

## b)

The histogram of the original image provides valuable insights into how pixel intensities are distributed. The spread of the histogram reflects the image's contrast. A wider spread suggests a broader range of intensity values, leading to higher contrast. histogram displays multiple peaks, it indicates the presence of different groups of intensity values.


*Figure 7: histogram of original image*

c)

The original image is read and converted to grayscale. It is then transformed into a NumPy array for efficient manipulation. The histogram of the original image is calculated to count the frequency of pixel intensities. Histogram equalization is performed by mapping the pixel intensities to their corresponding normalized CDF values, enhancing the image's contrast. The equalized image is reshaped, and its histogram is calculated and plotted.

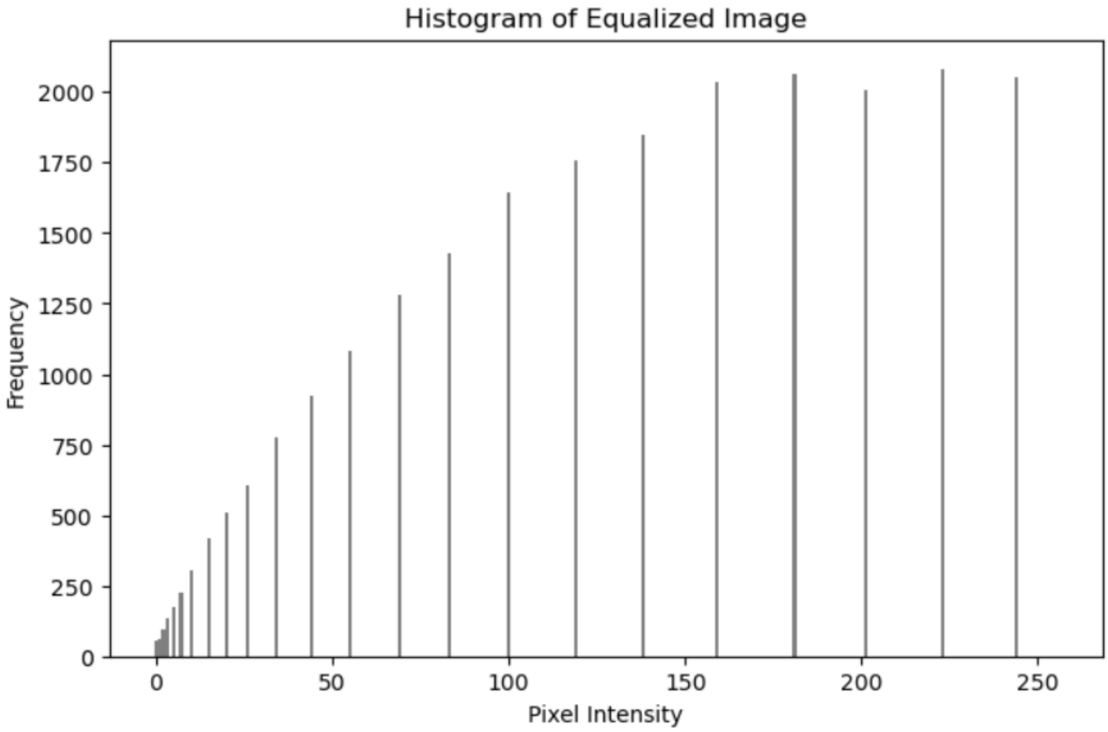


*Figure 8 : histogram of equalized image*

d)

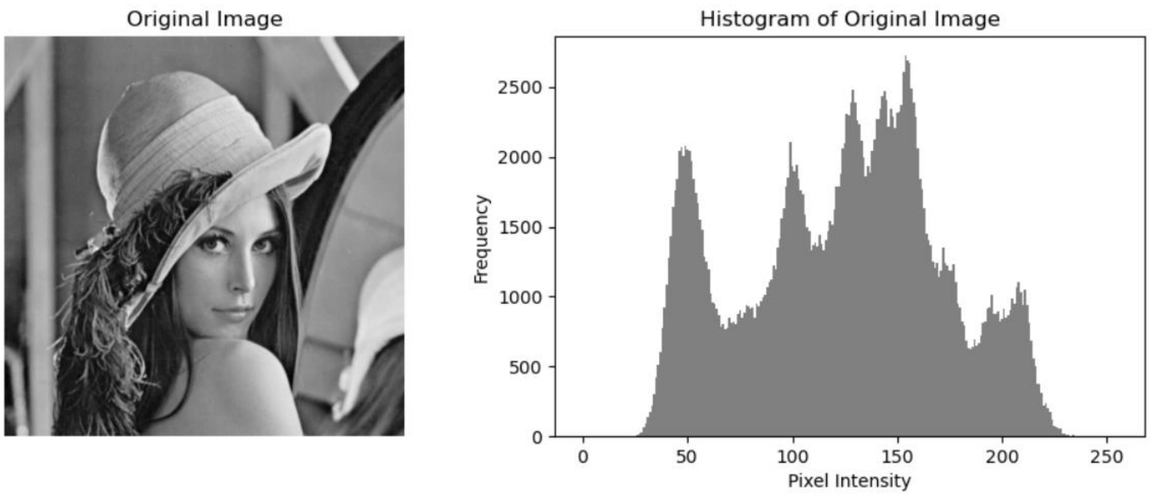


*Figure 9: original image*

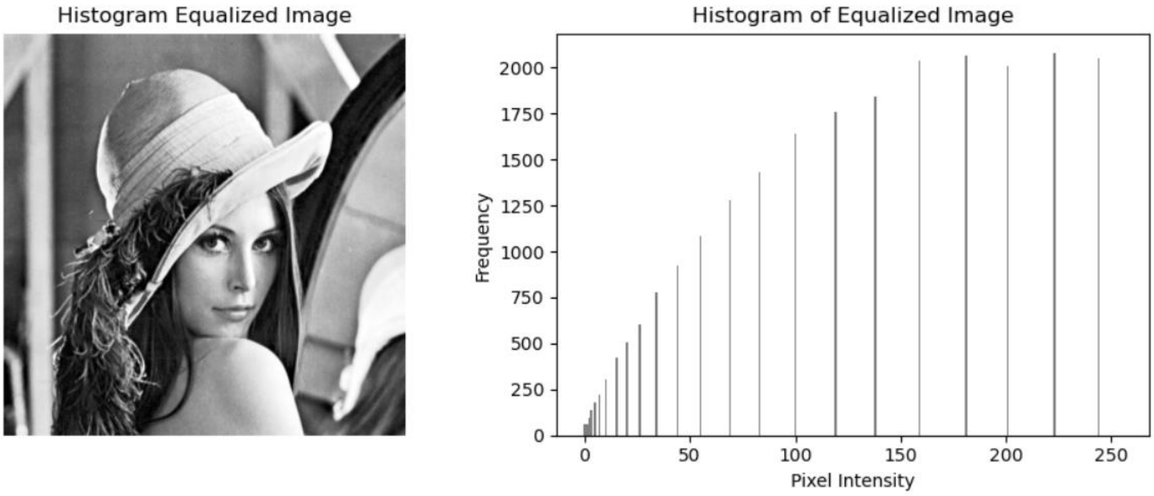


*Figure 10: equalized image*

# Question 4

## a & b )

The code manipulates an image of squares. It first loads the image and converts it to grayscale. Then, the image is converted to a NumPy array. The code defines translation parameters for both the x and y directions, as well as a rotation angle. It performs translation on the small square using np.roll(), which shifts the pixels in the image array. For the large square, it uses PIL's rotate() method after converting the image array back to a PIL image. The resulting rotated image is converted back to a NumPy array.

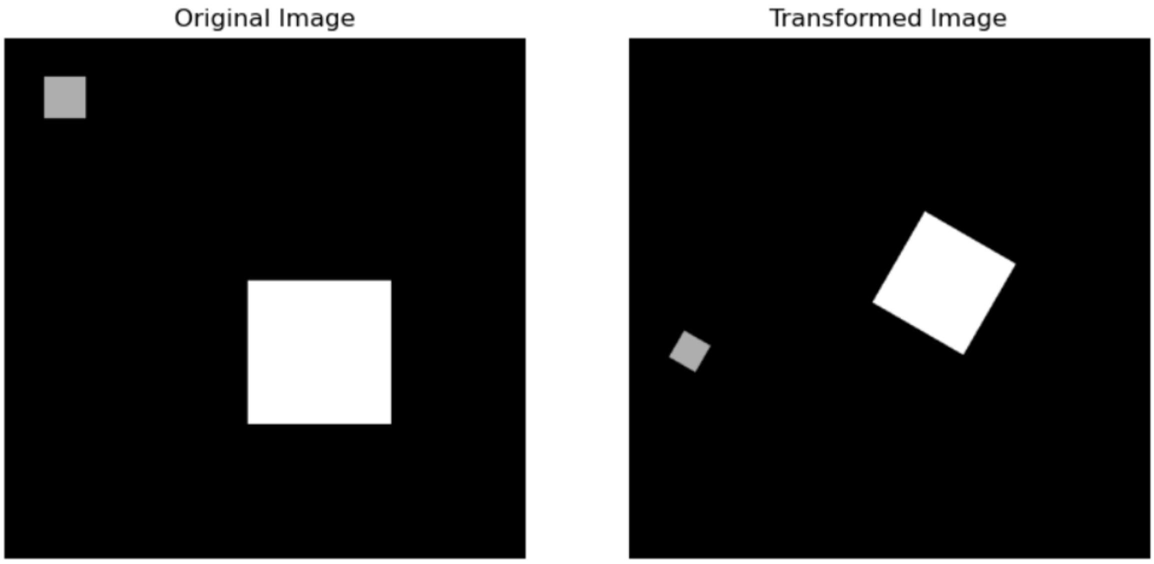


*Figure 11: original and transformed image*