EEL 5820: Digital Image Processing
Fall 2022

Homework #1

Image Negative Transform

Submitted by:
Maryna Veksler
5848285

Department of Electrical and Computer Engineering
09/19/2022

Objective:

Apply negative transformation to an image. Practice in reading/writing images and manipulating its pixels.

Method:

The negative transform of the image refers if subtracting pixel values from a maximum pixel value (i.e., 250), and replacing a corresponding pixel with a result. The overall formula of negative transform can be expressed as

$$255 - f(x, y),$$

where *f(x,y)* indicates a pixel of the image *f* at the position *(x, y)*.

The formula can be applied to both grayscale and RGB images. However, in the second scenario, it is required to apply the formula to pixels from each plane, R, G, and B

Results:

Example 1: The original image represents a structural pattern. While the original representation indicates that it is black and white image, when processed and analyzed, the shape of the image is as follows: heigh – 360, width – 640, and 3 channels are present – RGB.

Figure 1.1. Original Image (RGB)

Figure 1.2. Negative transform (RGB)

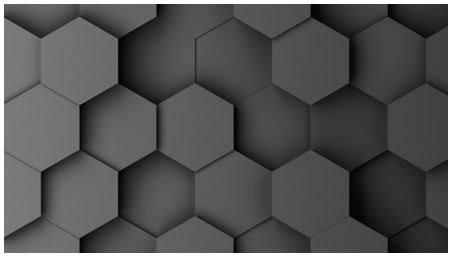

Figure 1.3. Original Image grayscale

Figure 1.4. Negative Transform Grayscale

Example 2: The original image represents a logo of a bird. While the original representation indicates that it is black and white image, when processed and analyzed, the shape of the image is as follows: heigh – 128, width – 128, and 3 channels are present – RGB.

Figure 2.1. Original Image (RGB)

Figure 2.2. Negative transform (RGB)

Figure 2.3. Original Image grayscale

Figure 2.4. Negative Transform Grayscale

Example 3: The original image represents a portrait of the lady. While the original representation indicates that it is black and white image, when processed and analyzed, the shape of the image is as follows: heigh – 584, width – 644, and 3 channels are present – RGB.

Figure 3.1. Original Image (RGB)

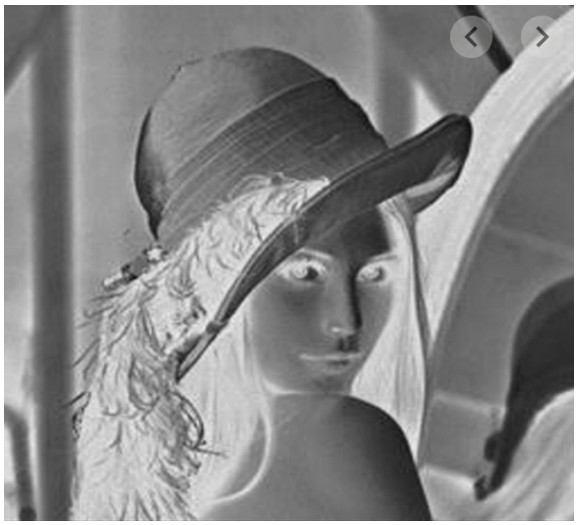

Figure 3.2. Negative transform (RGB)

Figure 3.3. Original Image grayscale

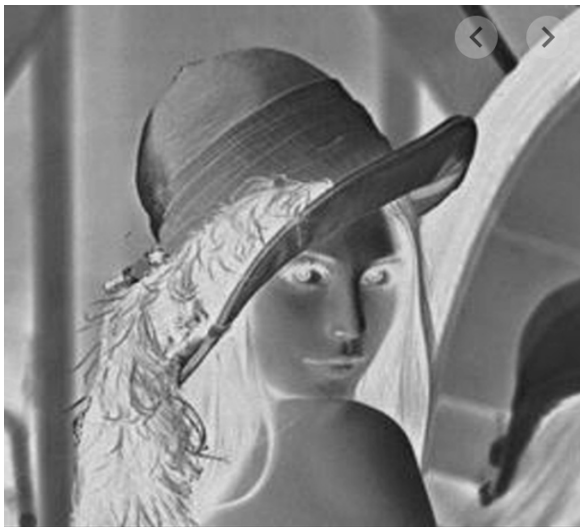

Figure 3.4. Negative Transform Grayscale

Example 4: The original image represents a city photo. The original representation is an RGB image with heigh – 481, width – 770.

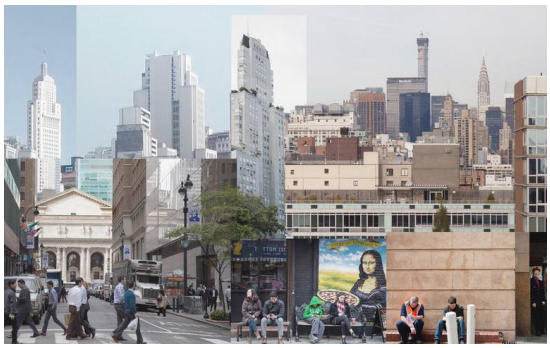

Figure 4.1. Original Image (RGB).

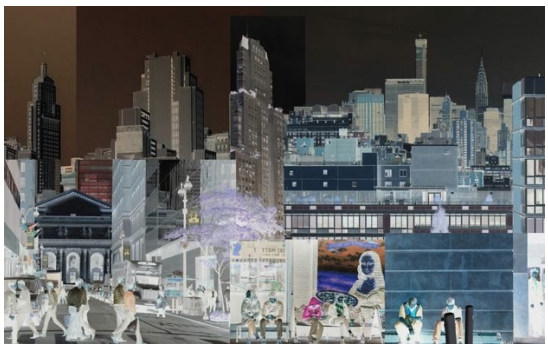

Figure 4.2. Negative transform (RGB)

Figure 4.3. Original Image grayscale

Figure 4.4. Negative Transform Grayscale

Discussion:

The negative transform is implemented for 4 different images. Examples 1, 2, and 3 illustrate images that appear as a gray scale image, while still having 3-colored channel characteristics. For each image we apply negative transform in two scenarios. First, the negative transform is applied to the original image (Figures 1.2, 2.2, 3.2); then the original images are converted to the grayscale (Figures 1.3, 2.3, 3.3); and the negative transform is applied again. The results indicate that negative transform for the original and grayscale images are visually identical in cases when the original image appears as black and white.

On the contrary, example 4 demonstrates the effect of the negative transform when applied to a bright RGB image. Figure 4.2. illustrates the negative transform of the original image. Unlike previous examples of the negative transform when applied to a truly RGB images, we also applied the transform to the gray scale version of the image. It is evident that in this scenario the transforms of the same image significantly differ for the original RGB and gray scale versions.

Program:

We implemented the program using Python programming language. We selected opencv and numpy libraries for the image processing. The implementation of our program consists of two

functions: gray_transform (image, im_name) and rgb_transform (image, im_name). The function are called from the main() function.

```
import cv2
import numpy as np
import argparse

def gray_transform (image, im_name):
    # get the height and width of the image
    height, width = image.shape

    # create new empty image array to store pixels after the negative transform
    new_image = np.empty((image.shape), np.float64)

    for i in range(height):
        for j in range(width):

            pixel = 255 - image[i,j]
            new_image[i, j, ...] = pixel

    # save the new image
    cv2.imwrite(im_name, new_image)

def rgb_transform (image, im_name):
    # get the height and width of the image
    height, width, channel = image.shape

    # create new empty image array to store pixels after the negative transform
    new_image = np.empty((image.shape), np.float64)

    for i in range(height):
        for j in range(width):

            pixel_r = 255 - image[i,j, 0]
            pixel_g = 255 - image[i,j, 1]
            pixel_b = 255 - image[i,j, 2]

            new_image[i, j, 0] = pixel_r
            new_image[i, j, 1] = pixel_g
            new_image[i, j, 2] = pixel_b

    # save the new image
    cv2.imwrite(im_name, new_image)

parser = argparse.ArgumentParser(
    description='Make predictions with signature network',
    formatter_class=argparse.ArgumentDefaultsHelpFormatter)
```

```
parser.add_argument('--image_path', type=str, required=True, help='Path to the
original image.')

if __name__ == "__main__":
    args = parser.parse_args()
    image_path = args.image_path

    # Read the image using cv2 library
    image = cv2.imread(image_path)

    # Check it the input image is grayscale or RGB
    # If image has less than 3 channels, the program only takes negative transform of
the gray scale image
    # Else the program (1) takes negative transform of the original image, (2)
converts the original image to the grayscale,
    # (3) takes negative transform of the gray scale version
    if len(image.shape) < 3:

        gray_transform(image, "gray_transform.jpg")

    else:
        rgb_transform(image, "rgb_transform.jpg")

        # get a gray scale version of the RGB image
        gray_image_1 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        # save gray scale image version
        cv2.imwrite("gray_version.jpg", gray_image_1)

        gray_transform(gray_image_1, "gray_transform.jpg")
```