CS 6476 - Computer Vision

Problem Set 0

Harsh Bhate (903424029)

Problem 1B (Using Python)

2.

(a).

```
x = np.random.permutation(1000)
```

The `np.random.permutation` function randomly permutes a sequence. In this case, the value `x` comprises of a list of values from 0 to 999 in a randomly shuffled order.

(b).

```
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = a[2,:]
```

The `np.array` function initializes a numpy array of the following value:

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

The command `a[2,:]` extracts the 3^{rd} row from `a`. That is,

$$b = \begin{bmatrix} 7 & 8 & 9 \end{bmatrix}$$

(c).

```
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = a.reshape(-1)
```

The `a.reshape(-1)` reshapes the 3×3 array `a` to a 9×1 array. Thus, b is a single dimensional array with values of `a` ordered by row. That is,

$$b = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \end{bmatrix}$$

In general, the `np.array.reshape` method of class `np.array` is used to convert the dimensions of a `np.array` object.

(d).

```
f = np.random.randn(5,1)
g = f[f>0]
```

The `np.random.randn` is used to generate a sample normal distribution of shape provided by the arguments. In this case, the function generates 5×1 samples of standard normal deviation. The line `f[f>0]` merely returns those samples in f whose values are larger than 0.

(e).

```
x = np.zeros(10)+0.5
y = 0.5*np.ones(len(x))
z = x + y
```

The function `np.zeros` is used to generate a `np.array` object of dimension specified by the arguments and values set to 0. Thus, the value of x is $\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$.
The function `np.ones` is used to generate a `np.array` object of dimension specified by the arguments and values set to 1. Thus, the valye of y is $\begin{bmatrix} 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 & 0.5 \end{bmatrix}$.
Finally, the value of z is merely the addition of the arrays x amnd y. Thus, z is $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$.

(f).

```
a = np.arange(1,100)
b = a[::-1]
```

The function `np.arange` generates an array with values from 1 to 99 in increments of one. That is,

$$a = \begin{bmatrix} 1 & 2 & \dots & 99 \end{bmatrix}$$

Next, the command `a[::-1]` merely reflects the array by its axis. In this case,

$$b = \begin{bmatrix} 99 & 98 & \dots & 1 \end{bmatrix}$$

3.

(a).

```
def all_outcomes(N):
    """Simulate the outcomes of a roll of a six-sided die over
    N trials using np.random.rand function
    Parameters
    ----------
    N : int
        Number of trials of the die
    Returns
    -------
    outcomes : np.array
        Array of outcomes of die-roll over N trials
    """
    SIDES_IN_DIE = 6
    prob_of_outcomes = np.random.rand(N, SIDES_IN_DIE)
    outcomes = []
    for roll_event in prob_of_outcomes:
        number_rolled = np.argmax(roll_event) + 1
        outcomes.append(number_rolled)
    return np.array(outcomes)
```

(b).

```
#!/usr/bin/python3

import numpy as np

y = np.array([1, 2, 3, 4, 5, 6])
NEW_SHAPE = (3,2)
z = y.reshape(NEW_SHAPE)
```

(c).

```
x = np.max(z)
dim = np.where(z == x)
r = dim[0][0]
c = dim[1][0]
```

(d).

```
#!/usr/bin/python3

import numpy as np

v = np.array([1, 8, 8, 2, 1, 3, 9, 8])
TARGET = 1
x = (v == TARGET).sum()
```

4.

(a).

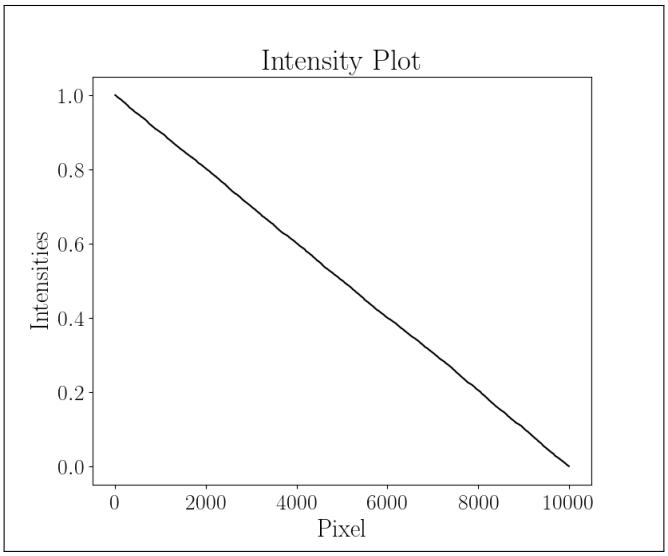

Figure 1: Plot of intensities in decreasing value

(b).

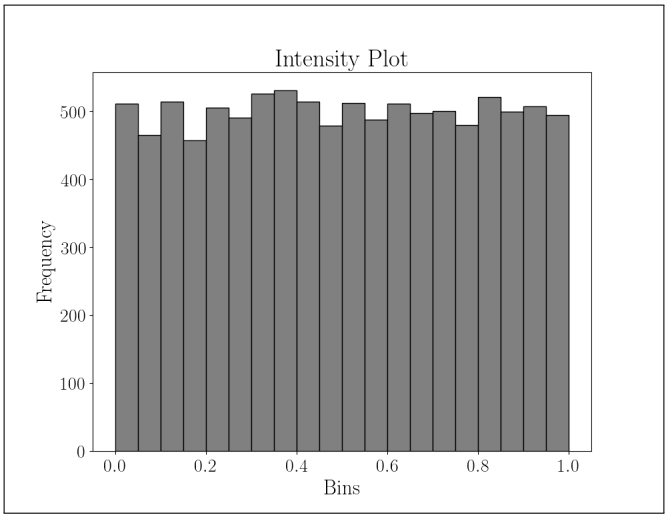

Figure 2: Histogram of the image

(c).

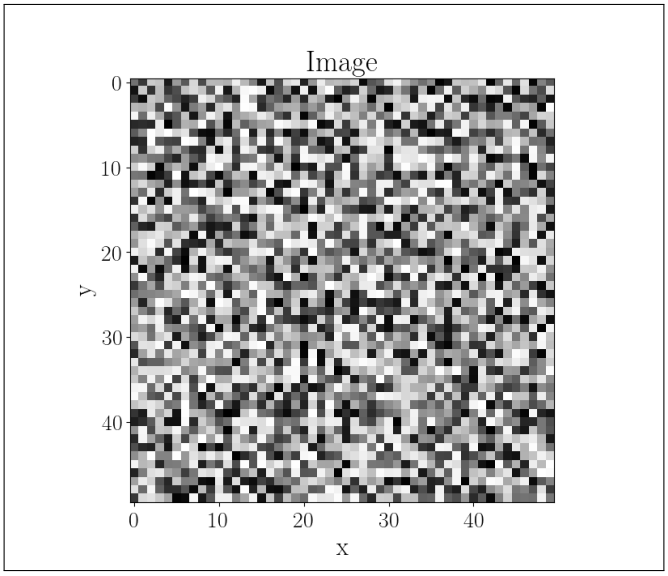

Figure 3: Cropped Image

(d).

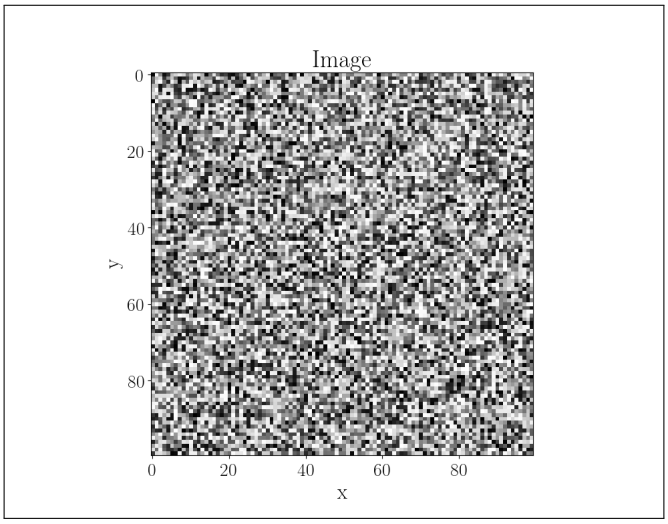

Figure 4: Mean Subtracted Image

(e).

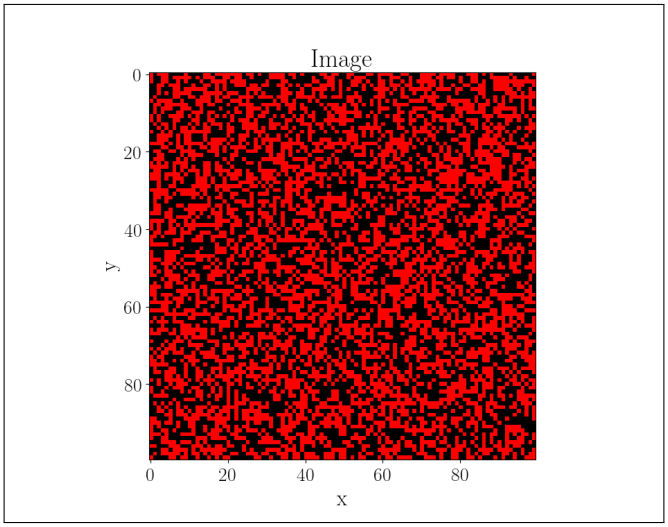

Figure 5: Thresholded RGB Image

Problem 2 (Using Python)

Figure 6: Original Image

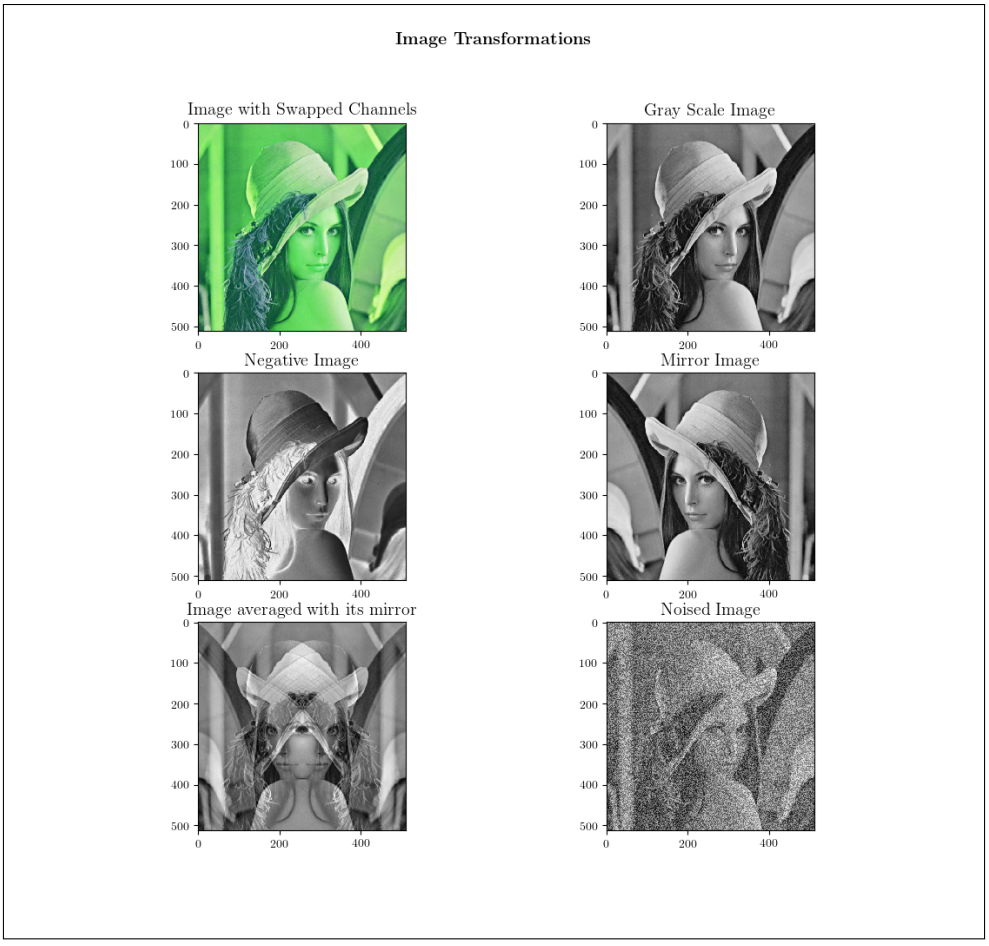

Figure 7: Original Image with various transformation