Student Name: Anumanchi Agastya Sai Ram Likhit
Student ID: 20048

Computer Vision
Assignment 1

Question 1

The following code explain the process of applying the filter on an image without using inbuilt libraries.

```
import numpy as np
import matplotlib.pyplot as plt

def apply_filter(original_image, filter):
    image_height, image_width = original_image.shape
    filter_height, filter_width = filter.shape
    filtered_image = np.zeros_like(original_image)

    for i in range(1, image_height - 1):
        for j in range(1, image_width - 1):
            region = original_image[i-1:i+2, j-1:j+2]
            filtered_value = np.sum(region * filter)
            filtered_image[i, j] = filtered_value

    return filtered_image
```

The function takes note of the dimensions of the original image and the filter and next it creates an empty image of same shape as the original image to store the filtered image pixels,now the function iterates through the original image, skipping the border pixels, as the filter is assumed to be of size 3x3.For each pixel, a 3x3 region around it is extracted and the extracted region is multiplied by the filter matrix, and the result is summed to obtain the filtered value for the current pixel.The filtered value is assigned to the corresponding pixel in the filtered image and this way we get the Filtered image.
Now below we will see how the image and the pixel values change according to the applied filters provided to us and discuss what the filters are doing.

0.1 Filter - 1

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 1: Original Image

```
array([[ 0,  0,  0, ...,  0,  0,  0],
       [ 0, 83, 82, ..., 27, 30,  0],
       [ 0, 81, 80, ..., 25, 32,  0],
       ...,
       [ 0, 14, 13, ..., 38, 46,  0],
       [ 0, 16, 15, ..., 56, 77,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 2: Filtered Image

This filter retains the original image, effectively doing nothing. The central value of 1 ensures that each pixel in the output will have the same value as the corresponding pixel in the input.

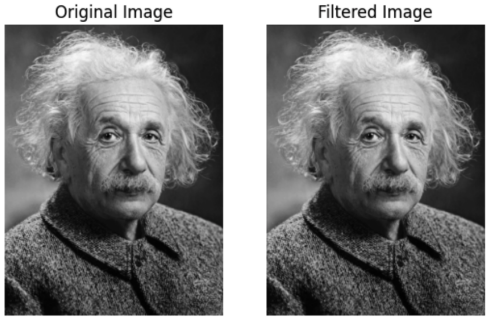

Figure 3: Comparison of original image with image after applying filter 1

If we see the image we find no change in the image visually to the naked eye because the image size is very big and having so many pixels,the shape of the image is $(2666, 2000)$.

0.2 Filter - 2

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 4: Original Image

```
: array([[ 0,  0,  0, ...,  0,  0,  0],
         [ 0, 82, 78, ..., 30, 33,  0],
         [ 0, 80, 76, ..., 32, 38,  0],
         ...,
         [ 0, 13, 12, ..., 46, 68,  0],
         [ 0, 15, 12, ..., 77, 93,  0],
         [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 5: Filtered Image

This filter shifts the pixel values one position to the left, creating a motion effect towards the left. If we see the image we find no change in the image visually to the naked eye because the image size is very big and having so many pixels.

0.3 Filter - 3

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

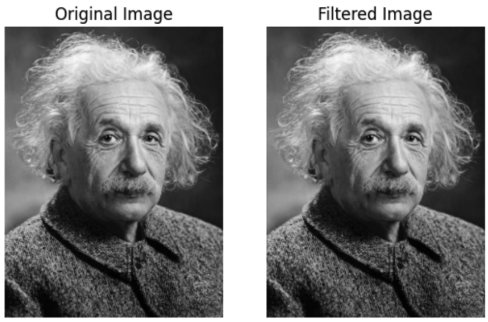

Figure 6: Comparison of original image with image after applying filter 2

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 7: Original Image

```
array([[ 0,  0,  0, ...,  0,  0,  0],
       [ 0, 80, 83, ..., 25, 27,  0],
       [ 0, 79, 81, ..., 22, 25,  0],
       ...,
       [ 0, 15, 14, ..., 46, 38,  0],
       [ 0, 16, 16, ..., 45, 56,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 8: Filtered Image

This filter shifts the pixel values one position to the right, creating a motion effect towards the right.

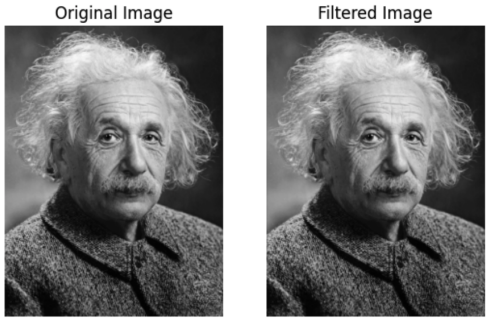

Figure 9: Comparison of original image with image after applying filter 3

If we see the image we find no change in the image visually to the naked eye because the image size is very big and having so many pixels.

0.4 Filter - 4

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 10: Original Image

```
array([[ 0,  0,  0, ...,  0,  0,  0],
       [ 0, 84, 79, ..., 26, 23,  0],
       [ 0, 82, 78, ..., 30, 33,  0],
       ...,
       [ 0, 14, 12, ..., 41, 58,  0],
       [ 0, 13, 12, ..., 46, 68,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 11: Filtered Image

This filter shifts the pixel values one position upward to the right, creating a unique spatial transformation.

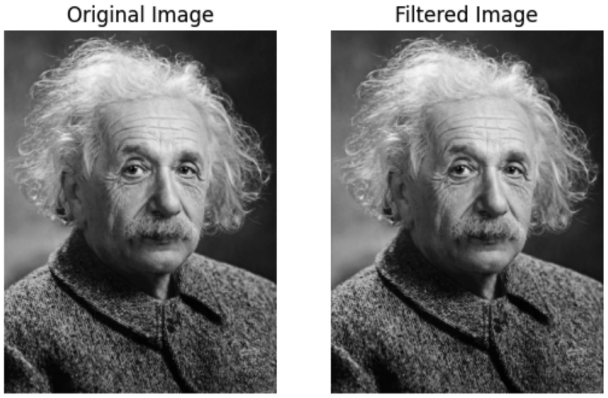

Figure 12: Comparison of original image with image after applying filter 4

If we see the image we find no change in the image visually to the naked eye because the image size is very big and having so many pixels.

0.5 Filter - 5

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 13: Original Image

```
: array([[ 0,  0,  0, ...,  0,  0,  0],
         [ 0, 82, 78, ..., 30, 33,  0],
         [ 0, 80, 76, ..., 32, 38,  0],
         ...,
         [ 0, 13, 12, ..., 46, 68,  0],
         [ 0, 15, 12, ..., 77, 93,  0],
         [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 14: Filtered Image

This filter shifts the pixel values one position downward to the right, creating a diagonal shifting

effect.

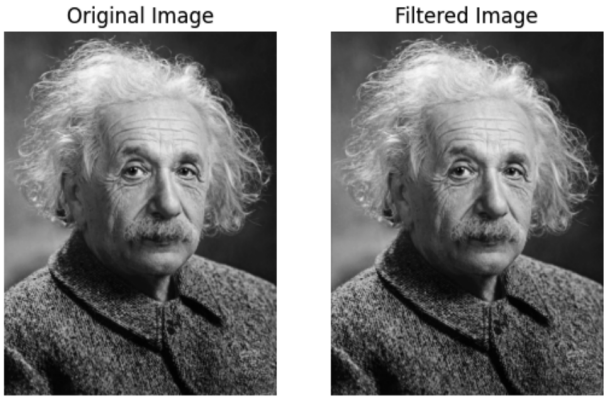

Figure 15: Comparison of original image with image after applying filter 5

If we see the image we find no change in the image visually to the naked eye because the image size is very big and having so many pixels.

Question 2

0.6 Filter 1

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

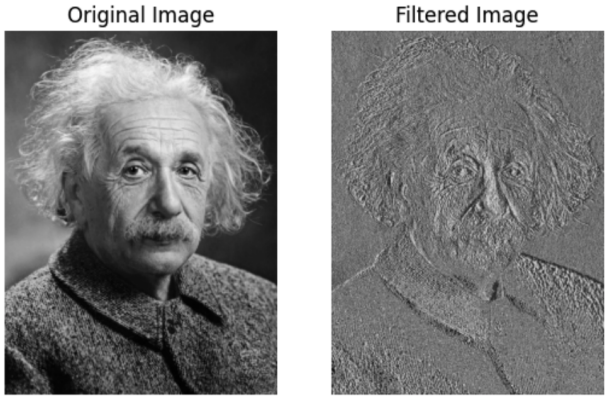

Figure 16: Comparison of original image with image after applying Vertical Sobel filter

When this filter is applied to an image,The positive values on the left side of the filter are used to detect vertical edges where the intensity increases from left to right,while the negative values on the right side of the filter are used to detect vertical edges where the intensity decreases from left to right,this filter is called Sobel filter and its more used in detecting edges and more particularly vertical edges within an image.

0.7 Filter 2

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

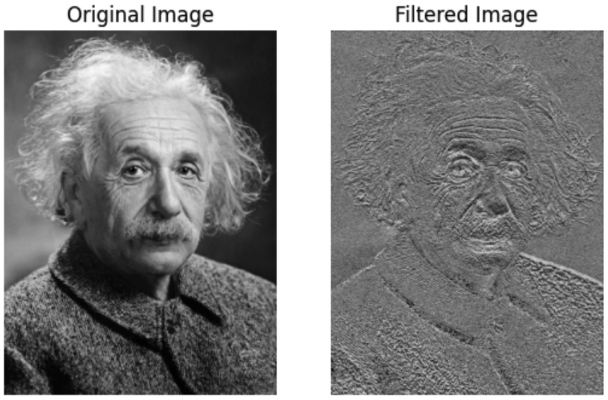

Figure 17: Comparison of original image with image after applying filter Horizontal Sobel filter

When this filter is applied to an image,The positive values on the top side of the filter are used to detect horizontal edges where the intensity increases from top to bottom,while the negative values on the bottom side of the filter are used to detect horizontal edges where the intensity decreases from top to bottom,this filter is also called Sobel filter and its more used in detecting edges and more particularly horizontal edges within an image.

Question 3

0.8 Filter 1

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 18: Original Image

```
array([[ 0,  0,  0, ...,  0,  0,  0],
       [ 0, 81, 80, ..., 27, 29,  0],
       [ 0, 79, 78, ..., 26, 30,  0],
       ...,
       [ 0, 14, 13, ..., 47, 57,  0],
       [ 0, 15, 13, ..., 55, 68,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 19: Filtered Image

it takes the average of the pixel values in a 3×3 neighborhood around each pixel in the original image.Specifically, it adds up the values of a pixel and its eight immediate neighbors, then divides the sum by 9,that is it simply replaces the image pixel with the average of the surrounding pixel values.

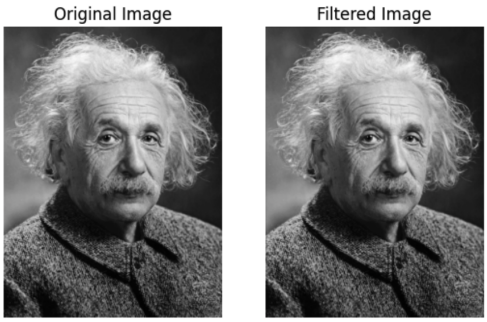

Figure 20: Comparison of original image with image after applying filter

The effect of this filter is to smooth the image, reducing sharp edges and noise.

0.9 Filter 2

$$\frac{1}{15}\begin{bmatrix}1 & 2 & 1\\ 2 & 3 & 2\\ 1 & 2 & 1\end{bmatrix}$$

```
: array([[ 80,  84,  84, ...,  28,  26,  23],
         [ 80,  83,  82, ...,  27,  30,  33],
         [ 79,  81,  80, ...,  25,  32,  38],
         ...,
         [ 15,  14,  13, ...,  38,  46,  68],
         [ 16,  16,  15, ...,  56,  77,  93],
         [ 16,  16,  15, ...,  59,  83, 100]], dtype=uint8)
```

Figure 21: Original Image

```
array([[ 0,  0,  0, ...,  0,  0,  0],
       [ 0, 81, 80, ..., 27, 29,  0],
       [ 0, 79, 79, ..., 26, 30,  0],
       ...,
       [ 0, 14, 13, ..., 46, 55,  0],
       [ 0, 15, 14, ..., 55, 69,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

Figure 22: Filtered Image

This is an Gaussian filter this also does the same job as the above filter,it does averaging but giving different weights to the surrounding pixel values.

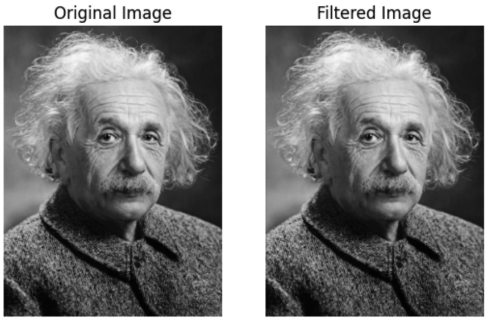

Figure 23: Comparison of original image with image after applying filter

But here due to the large number of pixels of the image we cant see the noticable change.