Image Processing with Python: Image Effects using Convolutional Filters and Kernels

How to blur, sharpen, outline, or emboss a digital image?

1st step:

As usual, we import libraries such as **numpy** and **matplotlib**. Additionally, we import specific functions from the **skimage** and scipy.signal library.

Python Code:

```
import numpy as np

import matplotlib.pyplot as plt

from skimage.io import imread, imshow

from skimage.color import rgb2gray

from skimage.transform import rescale

from scipy.signal import convolve2d
```

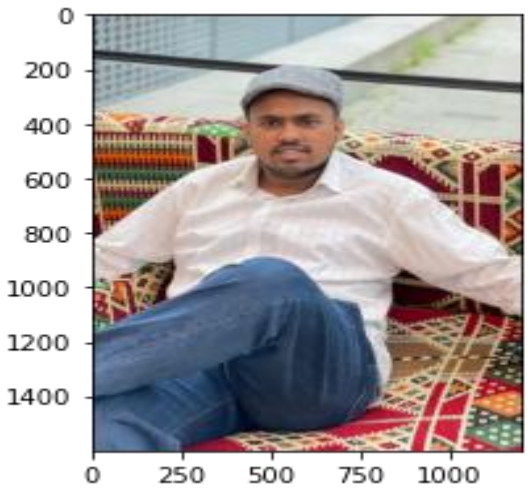
2nd Step:

I am going to use an image of my campus day program in Germany at Bremen University on 15th July.

Python Code:

```
my_day = imread('cday.jpg')

plt.imshow(my_day, cmap='gray’);
```

Output:



3rd Step:

To ensure that the effects of the filters and kernels are visually evident, let us rescale the image down to 10% of its original size.

Python Code:

```
r_scaled = rescale(my_day[:,:,0], 0.10)

g_scaled = rescale(my_day[:,:,1], 0.10)

b_scaled = rescale(my_day[:,:,2], 0.10)

my_day_scaled = np.stack([r_scaled, g_scaled, b_scaled], axis=2)

my_day_gray = rescale(rgb2gray(my_day), 0.10)
```

4th Step:

We have also defined a function that will apply the convolution function in all channels of the image, as shown below:

Python Code:

```
def rgb_convolve2d(image, kernel):

    red = convolve2d(image[:,:,0], kernel, 'valid')
```

```
    green = convolve2d(image[:,:,1], kernel, 'valid')
    blue = convolve2d(image[:,:,2], kernel, 'valid')
    return np.stack([red, green, blue], axis=2)
```

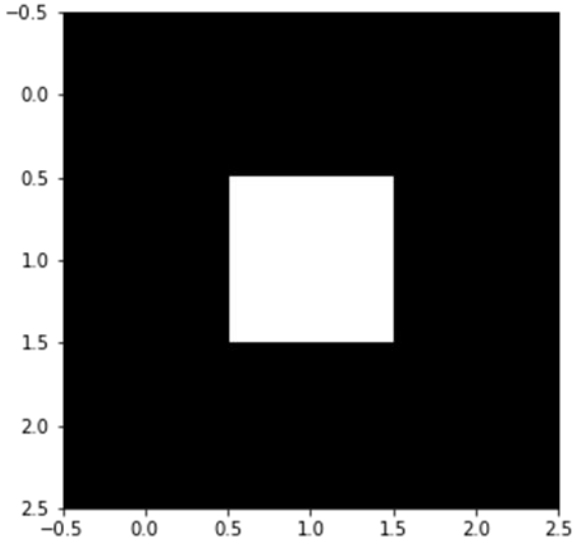
5th Step:

Now, let's try to apply the identity filter to the image of my image.

Python Code:

```
identity = np.array([[0, 0, 0],
                     [0, 1, 0],
                     [0, 0, 0]])
conv_im1 = rgb_convolve2d(my_day_scaled, identity)
fig, ax = plt.subplots(1,2, figsize=(12,5))
ax[0].imshow(identity, cmap='gray')
ax[1].imshow(abs(conv_im1), cmap='gray');
```

Output:



As expected, nothing happens! As the filter's name suggests, the identity kernel will return the input image itself.

6th Step:

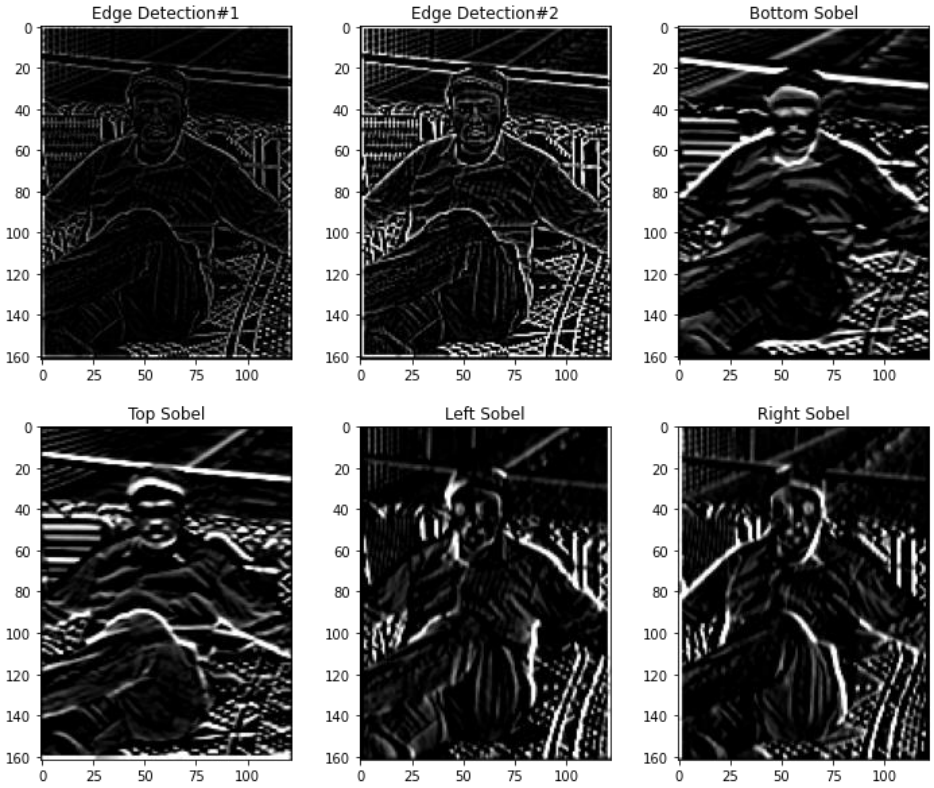
Now, let's try edge detection filters on the grayscale image of my image.

Python Code:

```
# Edge Detection1

kernel1 = np.array([[0, -1, 0],

                    [-1, 4, -1],

                    [0, -1, 0]])

# Edge Detection2

kernel2 = np.array([[-1, -1, -1],

                    [-1, 8, -1],

                    [-1, -1, -1]])

# Bottom Sobel Filter

kernel3 = np.array([[-1, -2, -1],

                    [0, 0, 0],

                    [1, 2, 1]])

# Top Sobel Filter

kernel4 = np.array([[1, 2, 1],

                    [0, 0, 0],

                    [-1, -2, -1]])
```

```
# Left Sobel Filter
kernel5 = np.array([[1, 0, -1],
                    [2, 0, -2],
                    [1, 0, -1]])
# Right Sobel Filter
kernel6 = np.array([[-1, 0, 1],
                    [-2, 0, 2],
                    [-1, 0, 1]])
kernels = [kernel1, kernel2, kernel3, kernel4, kernel5, kernel6]
kernel_name = ['Edge Detection#1', 'Edge Detection#2',
               'Bottom Sobel', 'Top Sobel',
               'Left Sobel', 'Right Sobel']figure, axis = plt.subplots(2,3, figsize=(12,10))
for kernel, name, ax in zip(kernels, kernel_name, axis.flatten()):
    conv_im1 = convolve2d(my_day_gray,
                          kernel[::-1, ::-1]).clip(0,1)
    ax.imshow(abs(conv_im1), cmap='gray')
    ax.set_title(name)
```

Output:



7th Step:

Now, let's try other types of kernel operators on the original image of my image.

Python Code:

```
# Sharpen

kernel7 = np.array([[0, -1, 0],

                    [-1, 5, -1],

                    [0, -1, 0]])

# Emboss

kernel8 =  np.array([[-2, -1, 0],

                     [-1,  1, 1],

                     [ 0,  1, 2]])
```

```
# Box Blur

kernel9 = (1 / 9.0) * np.array([[1, 1, 1],

                                [1, 1, 1],

                                [1, 1, 1]])

# Gaussian Blur 3x3

kernel10 = (1 / 16.0) * np.array([[1, 2, 1],

                                  [2, 4, 2],

                                  [1, 2, 1]])

# Gaussian Blur 5x5

kernel11 = (1 / 256.0) * np.array([[1, 4, 6, 4, 1],

                                   [4, 16, 24, 16, 4],

                                   [6, 24, 36, 24, 6],

                                   [4, 16, 24, 16, 4],

                                   [1, 4, 6, 4, 1]])

# Unsharp masking 5x5

kernel12 = -(1 / 256.0) * np.array([[1, 4, 6, 4, 1],

                                    [4, 16, 24, 16, 4],

                                    [6, 24, -476, 24, 6],

                                    [4, 16, 24, 16, 4],

                                    [1, 4, 6, 4, 1]])kernels = [kernel7, kernel8, kernel9, kernel10, kernel11, kernel12]
```

```
kernel_name = [‘Sharpen’, ‘Emboss’, ‘Box Blur’,
               ‘3x3 Gaussian Blur’, ‘5x5 Gaussian Blur’,
               ‘5x5 Unsharp Masking’]
figure, axis = plt.subplots(2,3, figsize=(12,10))
for kernel, name, ax in zip(kernels, kernel_name, axis.flatten()):
    conv_im1 = rgb_convolve2d(my_day_scaled,
                              kernel[::-1, ::-1]).clip(0,1)
    ax.imshow(abs(conv_im1), cmap=’gray’)
    ax.set_title(name)
```

Output: