Raghul S

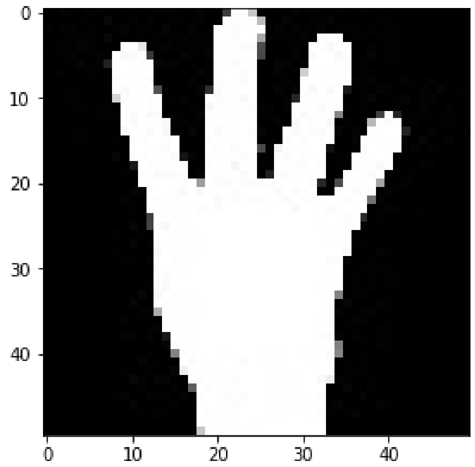
CB.EN.U4CSE19140

Exercise 1

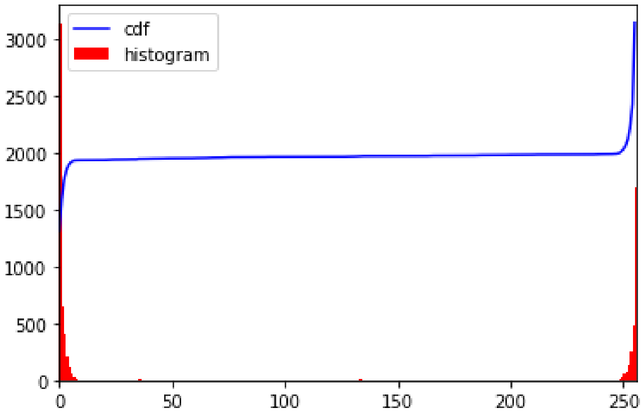
```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
from skimage.io import imshow, imread
from google.colab.patches import cv2_imshow
```

```
#Histogram Equalization
path= "/content/drive/MyDrive/Computer Vison/1.jpg"
img = cv.imread(path)
imshow(img)
```

```
<matplotlib.image.AxesImage at 0x7ff2363e0750>
```



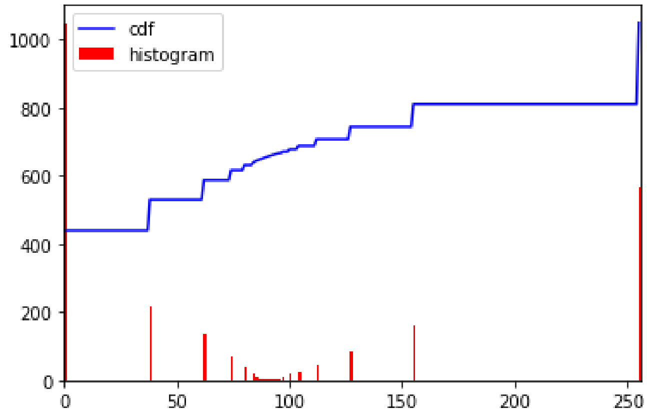
```
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(img.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



```
grayimg = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

equ=cv.equalizeHist(grayimg)
equ
```

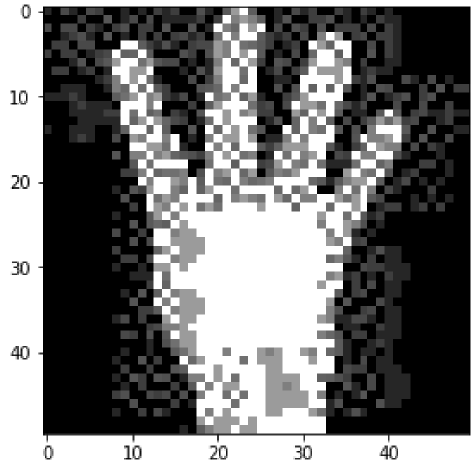
```
array([[38,  0, 74, ...,  0,  0,  0],
       [ 0,  0,  0, ...,  0,  0,  0],
       [74,  0, 74, ...,  0,  0,  0],
       ...,
       [ 0,  0,  0, ...,  0,  0,  0],
       [ 0,  0,  0, ...,  0,  0,  0],
       [ 0,  0,  0, ...,  0,  0,  0]], dtype=uint8)
```

```
hist,bins = np.histogram(equ.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()
plt.plot(cdf_normalized, color = 'b')
plt.hist(equ.flatten(),256,[0,256], color = 'r')
plt.xlim([0,256])
plt.legend(('cdf','histogram'), loc = 'upper left')
plt.show()
```



```
imshow(equ)
```

```
<matplotlib.image.AxesImage at 0x7ff2366e0290>
```



1) Histogram equalization is a method to process images in order to adjust the contrast of an image by modifying the intensity distribution of the histogram.

2)The main objective of this technique is to give a linear association to the cumulative probability function associated to the image.

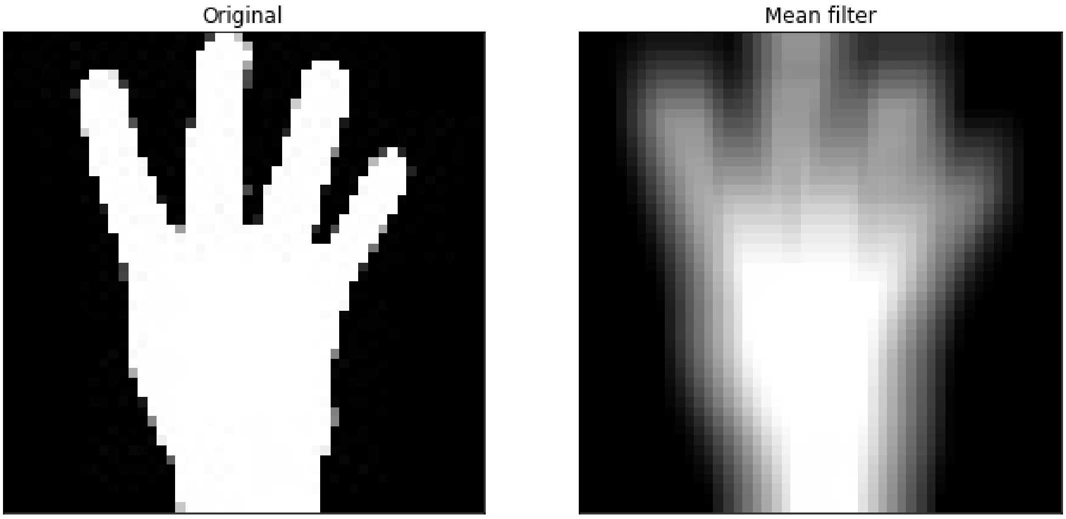
3) Histogram Equalization technique mainly relies on the Cumulative Probability Distribution Function cdf.

4)CDF= SUM OF ALL THE PROBANILITIES IN THE IMAGE DOMAIN

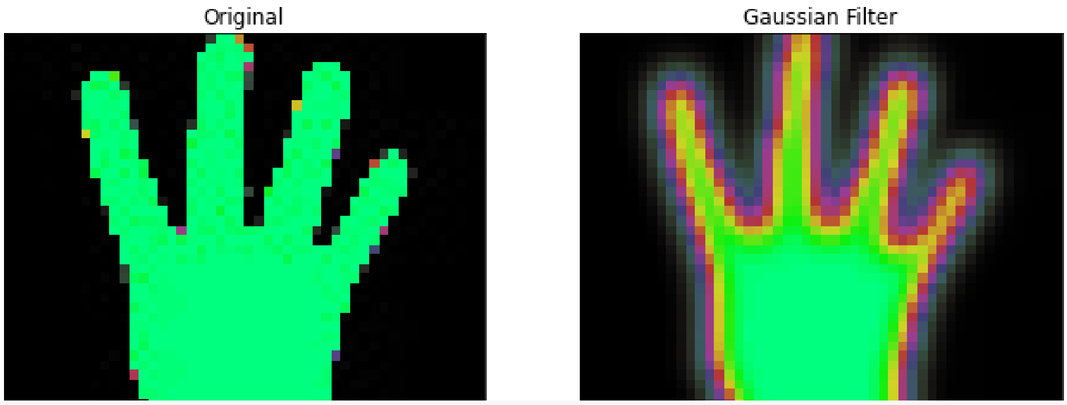
5) This technique is mainly used in enhancing the visual qualities of Gray scale images and also improving the contrast of X-ray and MRI images

Exercise 2

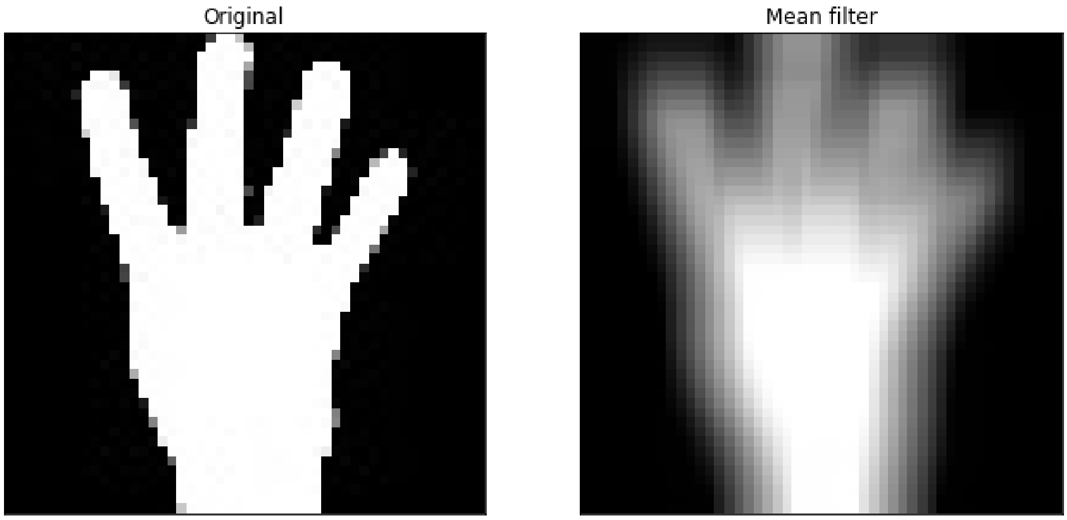
```
def mean_filter(path):
  image = cv.imread(path)
  image = cv.cvtColor(image, cv.COLOR_BGR2HSV)
  figure_size = 9
  new_image = cv.blur(image,(figure_size, figure_size))
  plt.figure(figsize=(11,6))
  plt.subplot(121), plt.imshow(cv.cvtColor(image, cv.COLOR_HSV2RGB)),plt.title('Original')
  plt.xticks([]), plt.yticks([])
  plt.subplot(122), plt.imshow(cv.cvtColor(new_image, cv.COLOR_HSV2RGB)),plt.title('Mean filter')
  plt.xticks([]), plt.yticks([])
  plt.show()
path= "/content/drive/MyDrive/Computer Vison/1.jpg"
mean_filter(path)
```



```
def gaussian_filter(path):
  image=cv.imread(path)
  figure_size = 9
  new_image = cv.GaussianBlur(image, (figure_size, figure_size),0)
  plt.figure(figsize=(11,6))
  plt.subplot(121), plt.imshow(cv.cvtColor(image, cv.COLOR_HSV2RGB)),plt.title('Original')
  plt.xticks([]), plt.yticks([])
  plt.subplot(122), plt.imshow(cv.cvtColor(new_image, cv.COLOR_HSV2RGB)),plt.title('Gaussian Filter')
  plt.xticks([]), plt.yticks([])
  plt.show()

gaussian_filter(path)
```



```
def median_filter(path):
  image=cv.imread(path)
  figure_size = 9
  new_image = cv.medianBlur(image, figure_size)
  plt.figure(figsize=(11,6))
  plt.subplot(121), plt.imshow(cv.cvtColor(image, cv.COLOR_HSV2RGB)),plt.title('Original')
  plt.xticks([]), plt.yticks([])
  plt.subplot(122), plt.imshow(cv.cvtColor(new_image, cv.COLOR_HSV2RGB)),plt.title('Median Filter')
  plt.xticks([]), plt.yticks([])
  plt.show()

mean_filter(path)
```



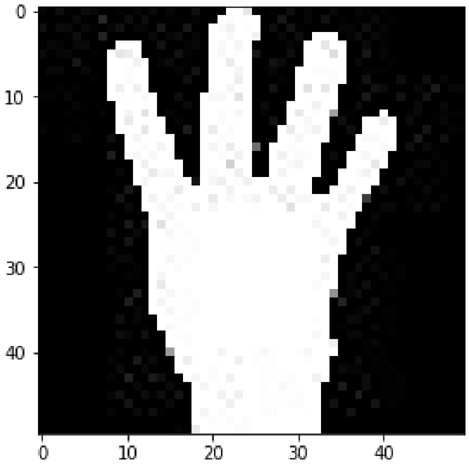
Exercise 3

```
#Sharpening An Image using OpenCV Library in Python
kernel = np.array([[0, -1, 0],
                   [-1, 5,-1],
                   [0, -1, 0]])
image_sharp = cv.filter2D(src=img, ddepth=-1, kernel=kernel)
image_sharp
```

```
array([[[ 5,  5,  5],
        [ 0,  0,  0],
        [14, 14, 14],
        ...,
        [ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0]],

       [[ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0],
        ...,
        [ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0]],

       [[15, 15, 15],
        [ 0,  0,  0],
        [11, 11, 11],
        ...,
        [ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0]],

       ...,

       [[ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0],
        ...,
        [ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0]],

       [[ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0],
        ...,
        [ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0]],

       [[ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0],
        ...,
        [ 0,  0,  0],
        [ 0,  0,  0],
        [ 0,  0,  0]]], dtype=uint8)
```

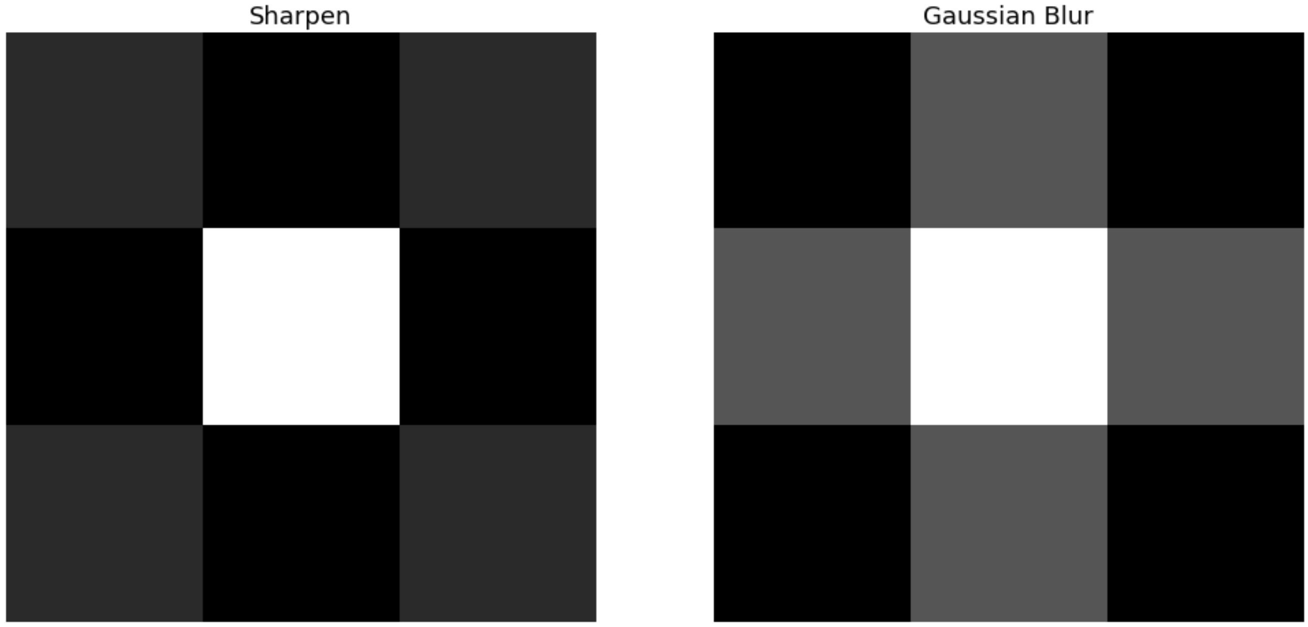
```
imshow(image_sharp)
```

```
<matplotlib.image.AxesImage at 0x7ff23643bcd0>
```



```
# Blurring and Sharpening
import matplotlib.pyplot as plt
from skimage.color import rgb2yuv, rgb2hsv, rgb2gray, yuv2rgb, hsv2rgb
from scipy.signal import convolve2d
```

```
hand= imread(path)
#Sharpen
sharpen = np.array([[0, -1, 0],
                    [-1, 5, -1],
                    [0, -1, 0]])
# Gaussian Blur
gaussian = (1 / 16.0) * np.array([[1., 2., 1.],
                                  [2., 4., 2.],
                                  [1., 2., 1.]])
fig, ax = plt.subplots(1,2, figsize = (17,10))
ax[0].imshow(sharpen, cmap='gray')
ax[0].set_title(f'Sharpen', fontsize = 18)

ax[1].imshow(gaussian, cmap='gray')
ax[1].set_title(f'Gaussian Blur', fontsize = 18)

[axi.set_axis_off() for axi in ax.ravel()];
```

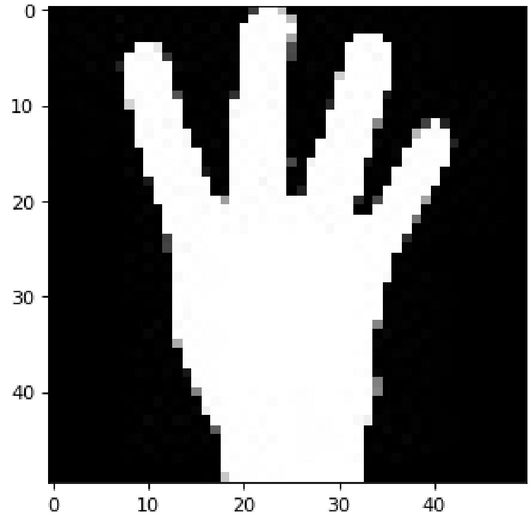


```
def multi_convolver(image, kernel, iterations):
    for i in range(iterations):
        image = convolve2d(image, kernel, 'same', boundary = 'fill',
                           fillvalue = 0)
    return image
multi_convolver(hand, gaussian, 2)
```

```
array([[0.2265625 , 0.4375    , 0.609375  , ..., 0.        , 0.        ,
        0.        ],
       [0.41015625, 0.7109375 , 0.9921875 , ..., 0.        , 0.        ,
        0.        ],
       [0.5078125 , 0.87109375, 1.265625  , ..., 0.        , 0.        ,
        0.        ],
       ...,
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ],
       [0.        , 0.        , 0.        , ..., 0.        , 0.        ,
        0.        ]])
```

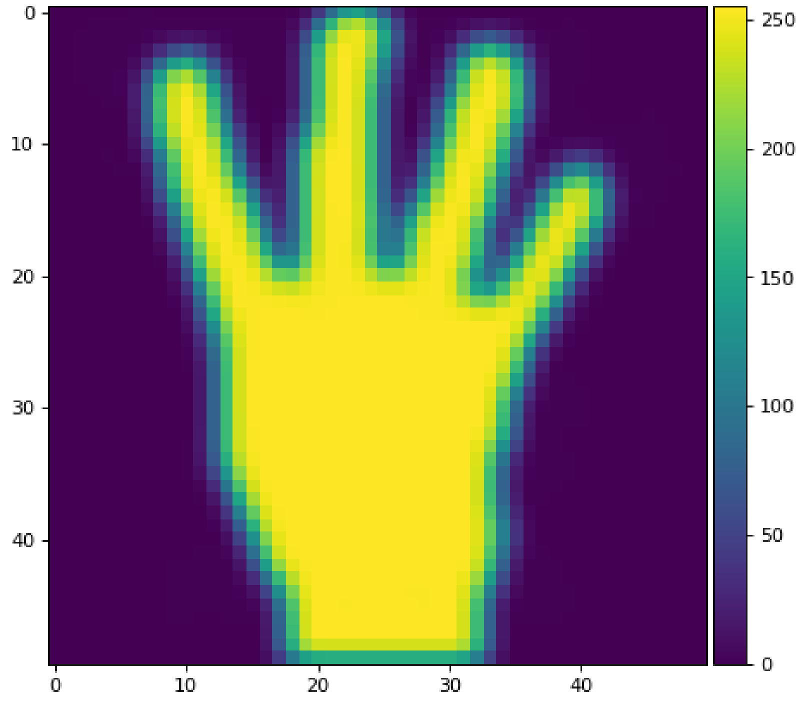
```
hand_grey = rgb2gray(hand)
plt.figure(num=None, figsize=(4, 4), dpi=80)
imshow(hand_grey);
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarning: The behavior of rgb2
  """Entry point for launching an IPython kernel.
```

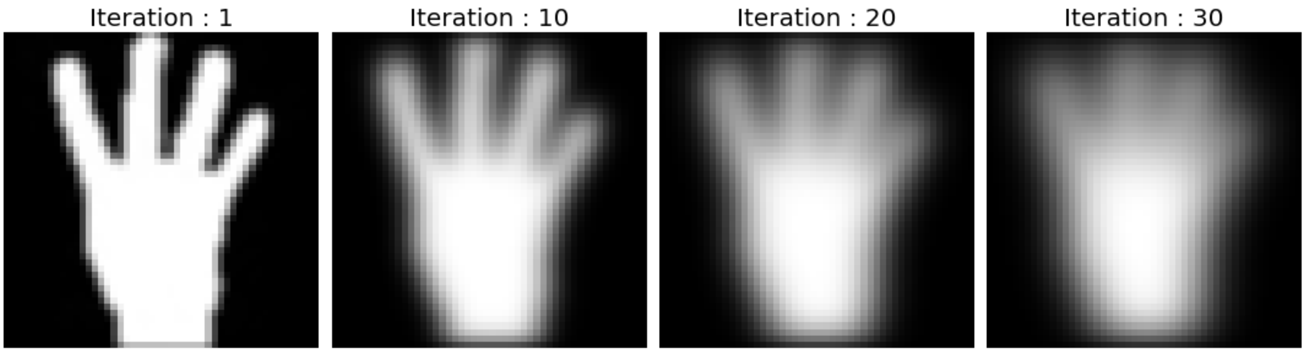


```
convolved_image = multi_convolver(hand_grey, gaussian, 2)
plt.figure(num=None, figsize=(6, 6), dpi=80)
imshow(convolved_image);
```

```
/usr/local/lib/python3.7/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: F
  lo, hi, cmap = _get_display_range(image)
```



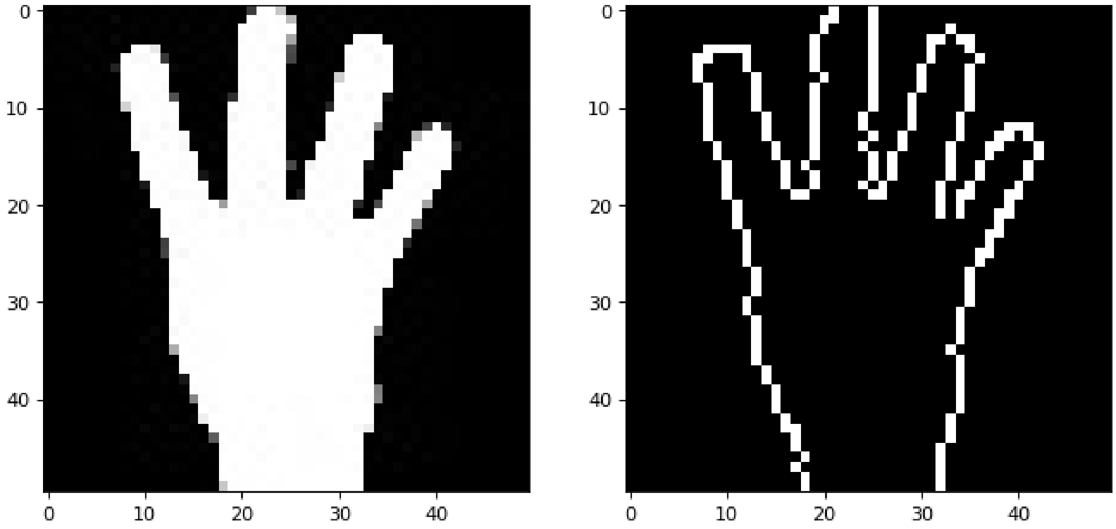
```
#gaussian bluring
def convolution_plotter(image, kernel):
    iterations = [1,10,20,30]
    f_size = 20

    fig, ax = plt.subplots(1,4, figsize = (15,7))
    for n, ax in enumerate(ax.flatten()):
        ax.set_title(f'Iteration : {iterations[n]}', fontsize =
                     f_size)
        ax.imshow(multi_convolver(image, kernel, iterations[n]),
                  cmap='gray')
        ax.set_axis_off()
    fig.tight_layout()

convolution_plotter(hand_grey, gaussian)
```



Exercise 4

```
#Simple Edge Detection Model
def simple_edge_detection(image):
   edges_detected = cv.Canny(image , 100, 200)
   images = [img , edges_detected]
   location = [121, 122]
   plt.figure(num=None, figsize=(10, 6), dpi=80)
   for loc,edge_image in zip(location, images):
      plt.subplot(loc)
      plt.imshow(edge_image, cmap='gray')
   cv.imwrite('edge_detected.png', edges_detected)
   plt.savefig('edge_plot.png')
   plt.show()
```

```
simple_edge_detection(img)
```



✓ 0s completed at 09:34