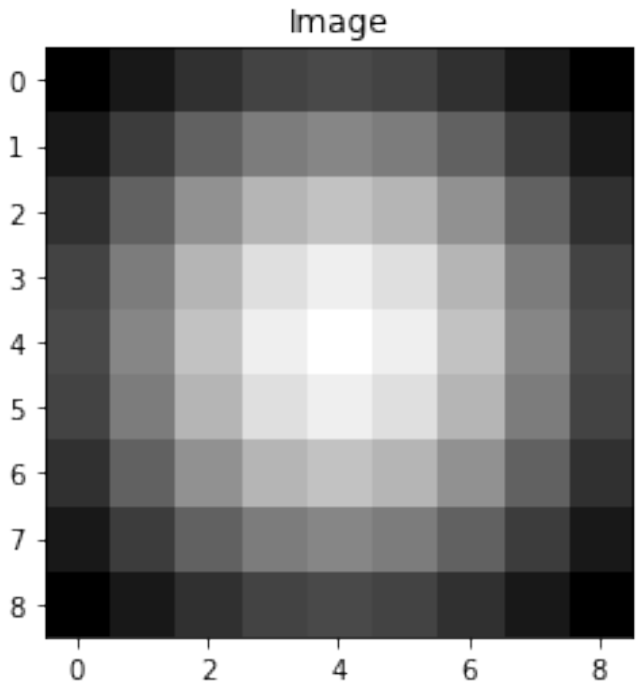
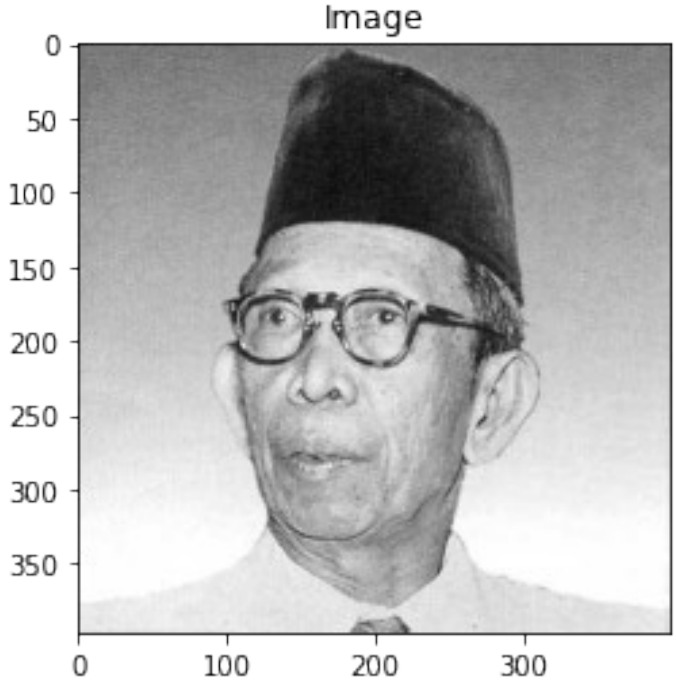
```
import numpy as np
import cv2
import argparse
import matplotlib.pyplot as plt
import math

def gaussian_kernel(size, sigma=1, verbose=False):

    kernel_1D = np.linspace(-(size // 2), size // 2, size)
    for i in range(size):
        kernel_1D[i] = dnorm(kernel_1D[i], 0, sigma)
    kernel_2D = np.outer(kernel_1D.T, kernel_1D.T)

    kernel_2D *= 1.0 / kernel_2D.max()

    if verbose:
        plt.imshow(kernel_2D, interpolation='none',cmap='gray')
        plt.title("Image")
        plt.show()

    return kernel_2D

def convolution(image, kernel, average=False, verbose=False):
    if len(image.shape) == 3:
        print("Found 3 Channels : {}".format(image.shape))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        print("Converted to Gray Channel. Size :
{}".format(image.shape))
    else:
        print("Image Shape : {}".format(image.shape))

    print("Kernel Shape : {}".format(kernel.shape))

    if verbose:
        plt.imshow(image, cmap='gray')
        plt.title("Image")
        plt.show()

    image_row, image_col = image.shape
    kernel_row, kernel_col = kernel.shape

    output = np.zeros(image.shape)

    pad_height = int((kernel_row - 1) / 2)
    pad_width = int((kernel_col - 1) / 2)

    padded_image = np.zeros((image_row + (2 * pad_height), image_col +
(2 * pad_width)))

    padded_image[pad_height:padded_image.shape[0] - pad_height,
pad_width:padded_image.shape[1] - pad_width] = image
```

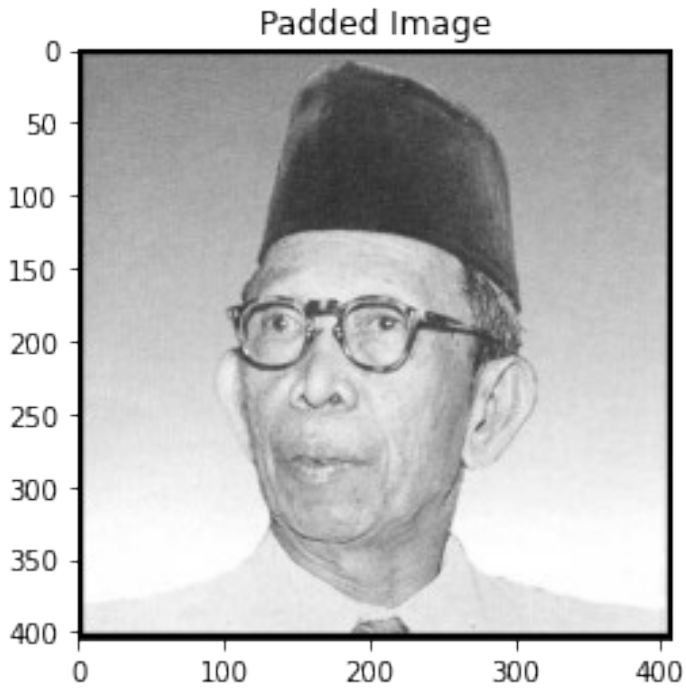
```
    if verbose:
        plt.imshow(padded_image, cmap='gray')
        plt.title("Padded Image")
        plt.show()

    for row in range(image_row):
        for col in range(image_col):
            output[row, col] = np.sum(kernel * padded_image[row:row +
kernel_row, col:col + kernel_col])
            if average:
                output[row, col] /= kernel.shape[0] * kernel.shape[1]

    print("Output Image size : {}".format(output.shape))

    if verbose:
        plt.imshow(output, cmap='gray')
        plt.title("Output Image using {}X{} Kernel".format(kernel_row,
kernel_col))
        plt.show()

    return output

def dnorm(x, mu, sd):
    return 1 / (np.sqrt(2 * np.pi) * sd) * np.e ** (-np.power((x - mu)
/ sd, 2) / 2)


def gaussian_kernel(size, sigma=1, verbose=False):
    kernel_1D = np.linspace(-(size // 2), size // 2, size)
    for i in range(size):
        kernel_1D[i] = dnorm(kernel_1D[i], 0, sigma)
    kernel_2D = np.outer(kernel_1D.T, kernel_1D.T)

    kernel_2D *= 1.0 / kernel_2D.max()

    if verbose:
        plt.imshow(kernel_2D, interpolation='none', cmap='gray')
        plt.title("Kernel ( {}X{} )".format(size, size))
        plt.show()

    return kernel_2D


def gaussian_blur(image, kernel_size, verbose=False):
    kernel = gaussian_kernel(kernel_size,
sigma=math.sqrt(kernel_size), verbose=verbose)
    return convolution(image, kernel, average=True, verbose=verbose)

def mySobel(image, verbose=False):
  versitive_x = convolution(image, filterX, verbose)
```

```
  if verbose:
        plt.imshow(versitive_x, cmap='gray')
        plt.title("Horizontal Edge")
        plt.show()

  versitive_y = convolution(image, np.flip(filterY.T, axis=0),
verbose)

  if verbose:
        plt.imshow(versitive_y, cmap='gray')
        plt.title("Vertical Edge")
        plt.show()

  magnitude = np.sqrt(np.square(versitive_x) + np.square(versitive_y))

  magnitude *= 255.0 / magnitude.max()

  if verbose:
        plt.imshow(magnitude, cmap='gray')
        plt.title("Gradient Magnitude")
        plt.show()

  return versitive_x, versitive_y, magnitude

filterX = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
filterY = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])

image = cv2.imread("Ki_Hajar_Dewantara,_Kemdikbud.jpg")
image = gaussian_blur(image, 9, verbose=True)
mySobel(image, verbose=True)
```

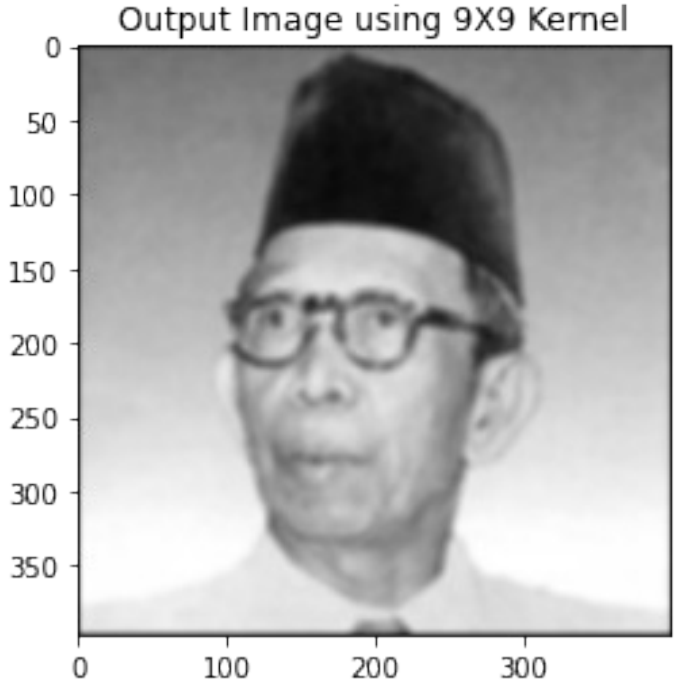



```
Found 3 Channels : (397, 400, 3)
Converted to Gray Channel. Size : (397, 400)
Kernel Shape : (9, 9)
```

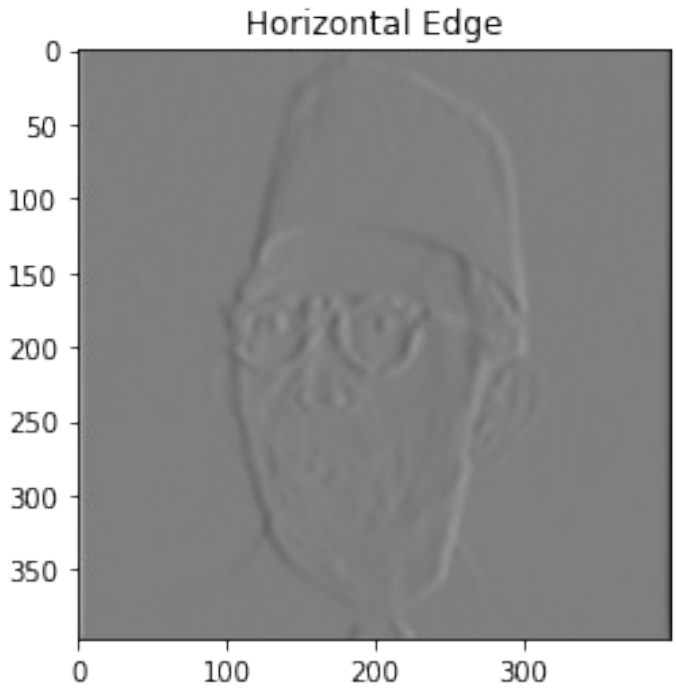




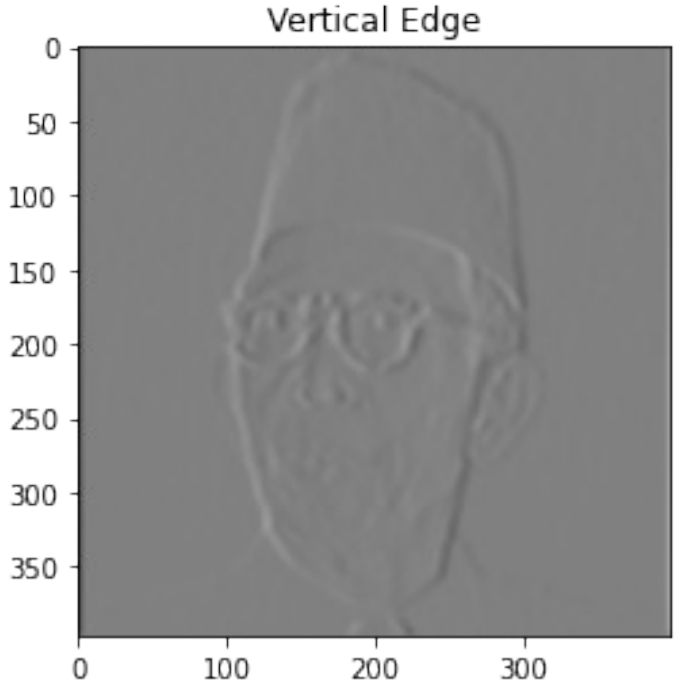

```
Output Image size : (397, 400)
```

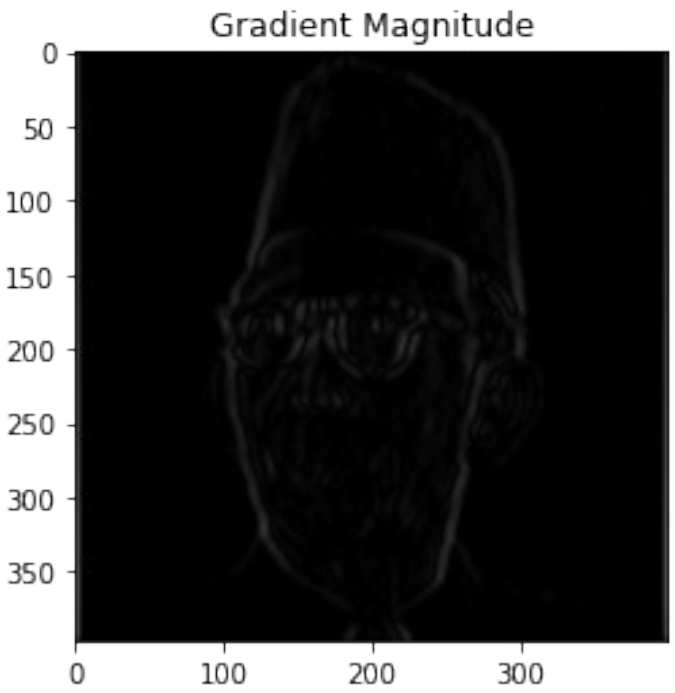



```
Image Shape : (397, 400)
Kernel Shape : (3, 3)
Output Image size : (397, 400)
```




```
Image Shape : (397, 400)
Kernel Shape : (3, 3)
Output Image size : (397, 400)
```






```
(array([[ 10.79568385,   3.9752507 ,   3.23369258, ...,  -3.43371522,
          -4.30029019, -11.68772144],
        [ 16.5454671 ,   6.10212987,   4.96871808, ...,  -5.23243923,
          -6.5481431 , -17.83272003],
        [ 19.38523819,   7.17550475,   5.85284135, ...,  -6.10647042,
          -7.63288508, -20.79051102],
        ...,
        [ 32.94154775,  12.24926934,   9.88034971, ...,  -9.58371972,
         -11.90359279, -33.08077289],
        [ 28.10510272,  10.44653298,   8.42190495, ...,  -8.20199181,
         -10.16818684, -28.24759709],
        [ 18.36259443,   6.82350839,   5.49912222, ...,  -5.37032334,
          -6.64826419, -18.46100272]]),
 array([[-10.79568385,  -3.9752507 ,  -3.23369258, ...,   3.43371522,
           4.30029019,  11.68772144],
        [-16.5454671 ,  -6.10212987,  -4.96871808, ...,   5.23243923,
           6.5481431 ,  17.83272003],
        [-19.38523819,  -7.17550475,  -5.85284135, ...,   6.10647042,
           7.63288508,  20.79051102],
        ...,
        [-32.94154775, -12.24926934,  -9.88034971, ...,   9.58371972,
          11.90359279,  33.08077289],
        [-28.10510272, -10.44653298,  -8.42190495, ...,   8.20199181,
          10.16818684,  28.24759709],
        [-18.36259443,  -6.82350839,  -5.49912222, ...,   5.37032334,
           6.64826419,  18.46100272]]),
 array([[ 64.2318302 ,  23.65182525,  19.23972542, ...,  20.429814  ,
          25.58573531,  69.53924826],
        [ 98.44171506,  36.30626605,  29.56272718, ...,  31.13180723,
```

```
  38.95994192, 106.10057325],
[115.33769845,  42.69259913,  34.8230568 , ...,  36.33209128,
  45.41390665, 123.69874778],
...,
[195.99461522,  72.88032878,  58.78580307, ...,  57.02092303,
  70.82363299, 196.8229727 ],
[167.21888224,  62.15446302,  50.108393  , ...,  48.79996048,
  60.49836768, 168.06669089],
[109.25320378,  40.59830194,  32.71850951, ...,  31.95218587,
  39.55563933, 109.83871045]]))
```