# ABSU PREMACHANDRA 0137/ET

## IMAGE PROCESSING

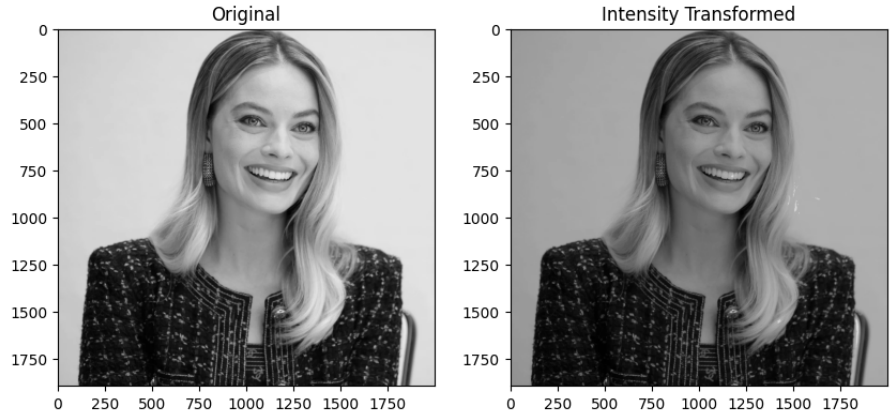
## ET3112

## ASSIGNMNET 1

QUESTION NO.01

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt

from google.colab import drive #Mount the google drive
drive.mount('/content/drive')
im1 = cv.imread('/content/drive/MyDrive/images/margot_golden_gray.jpg', cv.IMREAD_GRAYSCALE) #Load the image

assert im1 is not None #Check if the image is loaded

t = np.zeros(256,dtype = np.uint8)

t[0:230] = np.array([int(x*170/220) for x in range (230)])
t[230:256] = np.array([int(x*170/220 +40) for x in range(230,256)]) #Apply intensity transformation

im2 = t[im1]

fig, ax = plt.subplots(1,2,figsize = (10,10))
ax[0].imshow(im1, vmin = 0, vmax = 256, cmap='gray')
ax[0].set_title('Original')
ax[1].imshow(im2, vmin = 0, vmax = 256, cmap = 'gray')
ax[1].set_title('Intensity Transformed') #Plot the original and intensity transformed images
plt.show()
```

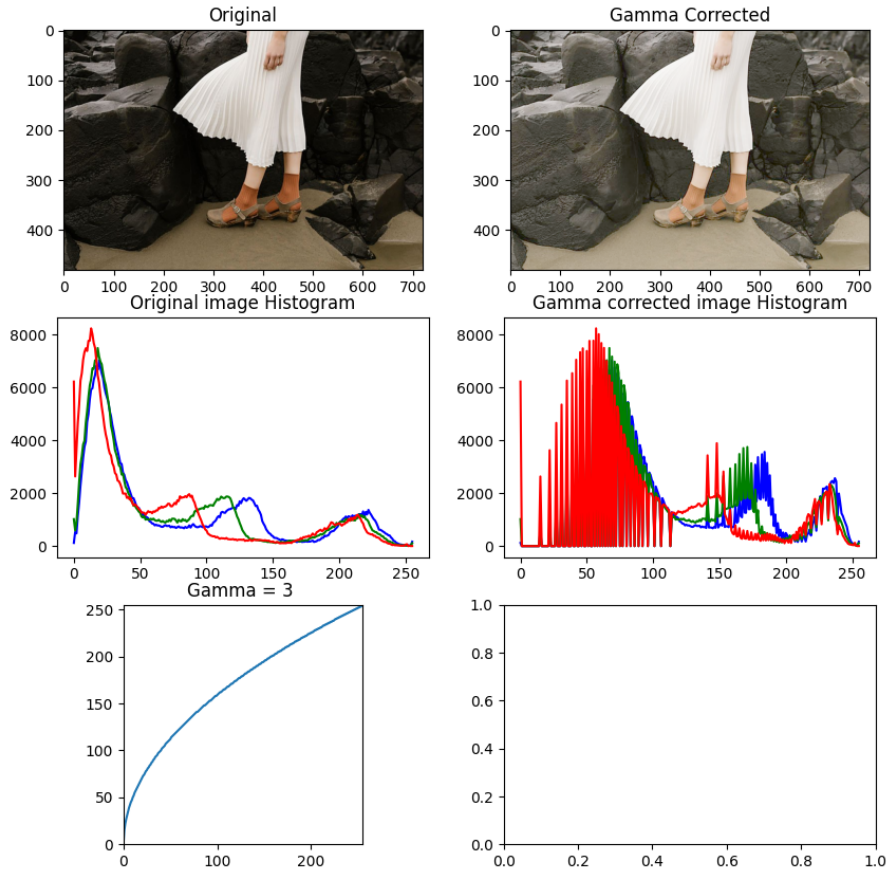
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```




## QUESTION NO.2

```
#Import necessary libraries
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
#Mount the google drive
from google.colab import drive
drive.mount('/content/drive')
im1 = cv.imread('/content/drive/MyDrive/images/highlights_and_shadows.jpg', cv.IMREAD_COLOR)#Read the image

assert im1 is not None #Check if the image is loaded

gamma = 0.5
table = np.array([(i/255.0)**(gamma)*255.0 for i in np.arange(0,256)]).astype('uint8')
g_corr = cv.LUT(im1,table) #Apply gamma correction

im1 = cv.cvtColor(im1, cv.COLOR_BGR2RGB) #Convert the image from BGR to RGB
g_corr = cv.cvtColor(g_corr, cv.COLOR_BGR2RGB)

fig, axarr = plt.subplots(3,2,figsize=(10,10))
axarr[0,0].imshow(im1)
axarr[0,0].set_title('Original')
axarr[0,1].imshow(g_corr)
axarr[0,1].set_title('Gamma Corrected') #Plot the original and gamma corrected images

color = ('b', 'g','r')
for i, c in enumerate(color):
  hist_org = cv.calcHist([im1],[i],None,[256],[0,256])
  axarr[1,0].plot(hist_org, color = c)
  axarr[1,0].set_title('Original image Histogram')
  hist_gamma = cv.calcHist([g_corr],[i],None,[256],[0,256])
  axarr[1,1].plot(hist_gamma, color = c) #Plot the histograms
  axarr[1,1].set_title('Gamma corrected image Histogram')

axarr[2,0].plot(table) #Plot gamma correction table
axarr[2,0].set_title('Gamma = 3 ')
axarr[2,0].set_xlim(0,255)
axarr[2,0].set_ylim(0,255)
axarr[2,0].set_aspect('equal')

plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```




## Question No.03

### Part a

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

from google.colab import drive
drive.mount('/content/drive')

image = cv.imread("/content/drive/MyDrive/images/spider.png", cv.IMREAD_COLOR)

assert image is not None

# Convert the image from BGR to HSV color space
hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)

# Split the image into hue, saturation, and value planes
hue, saturation, value = cv.split(hsv_image)

# Display the individual channels
plt.figure(figsize=(10, 10))

fig, ax = plt.subplots(1,3, figsize = (10,10))

ax[0].imshow(hue, cmap='hsv')
ax[0].set_title('Hue')
ax[0].axis('off')

#plt.subplot(1, 3, 1)
#plt.imshow(hue, cmap='hsv')
#plt.title('Hue')
#plt.axis('off')

ax[1].imshow(saturation, cmap = 'gray')
ax[1].set_title('Saturation')
ax[1].axis('off')

#plt.subplot(1, 3, 2)

#plt.title('Saturation')
#plt.axis('off')

ax[2].imshow(value, cmap='gray')
ax[2].set_title('Value')
ax[2].axis('off')

#plt.subplot(1, 3, 3)
#plt.imshow(value, cmap='gray')
#plt.title('Value')


plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
<Figure size 1000x1000 with 0 Axes>
```


## Part b

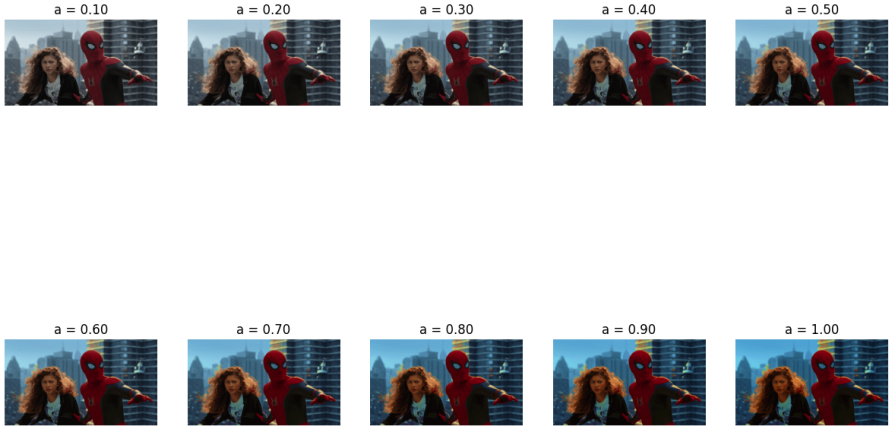
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

image = cv.imread("/content/drive/MyDrive/images/spider.png", cv.IMREAD_COLOR)

assert image is not None

# Convert the image from BGR to HSV color space
hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)

# Split the image into hue, saturation, and value planes
hue, saturation, value = cv.split(hsv_image)

# Define the intensity transformation function
def intensity_transformation(x, a=0.5, sigma=70):
    return np.clip(a * x + a * 128 * np.exp(-((x - 128) ** 2) / (2 * sigma ** 2)), 0, 255).astype(np.uint8)

# Apply the intensity transformation to the saturation plane
a = 0.5  # You can adjust this value as needed
enhanced_saturation = intensity_transformation(saturation, a=a)

# Recombine the channels
enhanced_hsv_image = cv.merge([hue, enhanced_saturation, value])

# Convert the enhanced HSV image back to BGR color space
enhanced_image = cv.cvtColor(enhanced_hsv_image, cv.COLOR_HSV2BGR)

# Display the original and enhanced images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(cv.cvtColor(enhanced_image, cv.COLOR_BGR2RGB))
plt.title('Vibrance-Enhanced Image (a={})'.format(a))
plt.axis('off')

plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```


## Part c

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')

# Read the image
image = cv2.imread("/content/drive/MyDrive/images/spider.png", cv2.IMREAD_COLOR)
assert image is not None

# Convert the image from BGR to HSV color space
hsv_image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# Split the image into hue, saturation, and value planes
hue, saturation, value = cv2.split(hsv_image)

# Define the intensity transformation function
def intensity_transformation(x, a=0.5, sigma=70):
    return np.clip(x + a * 128 * np.exp(-((x - 128) ** 2) / (2 * sigma ** 2)), 0, 255).astype(np.uint8)

# Define a range of 'a' values
a_values = np.linspace(0.1, 1.0, num=10)  # Values from 0.1 to 1.0 with 10 steps

# Plot each enhanced image corresponding to different values of 'a'
plt.figure(figsize=(15, 10))
for i, a in enumerate(a_values):
    # Apply the intensity transformation to the saturation plane
    enhanced_saturation = intensity_transformation(saturation, a=a)

    # Recombine the enhanced saturation with the original hue and value channels
    enhanced_hsv_image = cv2.merge([hue, enhanced_saturation, value])

    # Convert the enhanced HSV image back to BGR color space
    enhanced_image = cv2.cvtColor(enhanced_hsv_image, cv2.COLOR_HSV2BGR)

    # Display the enhanced image
    plt.subplot(2, 5, i + 1)
    plt.imshow(cv2.cvtColor(enhanced_image, cv2.COLOR_BGR2RGB))
    plt.title('a = {:.2f}'.format(a))
    plt.axis('off')

plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

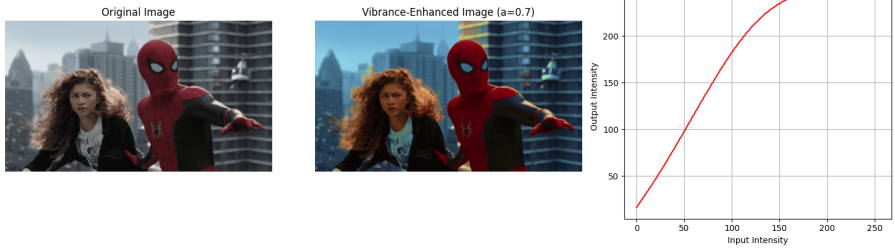


a = 0.7 is good

## Part d

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')

image = cv.imread("/content/drive/MyDrive/images/spider.png", cv.IMREAD_COLOR)

assert image is not None

# Convert the image from BGR to HSV color space
hsv_image = cv.cvtColor(image, cv.COLOR_BGR2HSV)

# Split the image into hue, saturation, and value planes
hue, saturation, value = cv.split(hsv_image)

# Define the intensity transformation function
def intensity_transformation(x, a=0.5, sigma=70):
    return np.clip(x + a * 128 * np.exp(-((x - 128) ** 2) / (2 * sigma ** 2)), 0, 255).astype(np.uint8)

# Apply the intensity transformation to the saturation plane
a = 0.7
enhanced_saturation = intensity_transformation(saturation, a=a)

# Recombine the enhanced saturation with the original hue and value channels
enhanced_hsv_image = cv.merge([hue, enhanced_saturation, value])

# Convert the enhanced HSV image back to BGR color space
enhanced_image = cv.cvtColor(enhanced_hsv_image, cv.COLOR_HSV2BGR)

# Display the original image, the vibrance-enhanced image, and the intensity transformation
plt.figure(figsize=(15, 5))

# Original Image
plt.subplot(1, 3, 1)
plt.imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB))
plt.title('Original Image')
plt.axis('off')

# Vibrance-Enhanced Image
plt.subplot(1, 3, 2)
plt.imshow(cv.cvtColor(enhanced_image, cv.COLOR_BGR2RGB))
plt.title('Vibrance-Enhanced Image (a={})'.format(a))
plt.axis('off')

# Intensity Transformation
x = np.arange(0, 256)
plt.subplot(1, 3, 3)
plt.plot(x, intensity_transformation(x, a=a), color='r')
plt.title('Intensity Transformation (a={})'.format(a))
plt.xlabel('Input Intensity')
plt.ylabel('Output Intensity')
plt.grid(True)

plt.tight_layout()
plt.show()
```

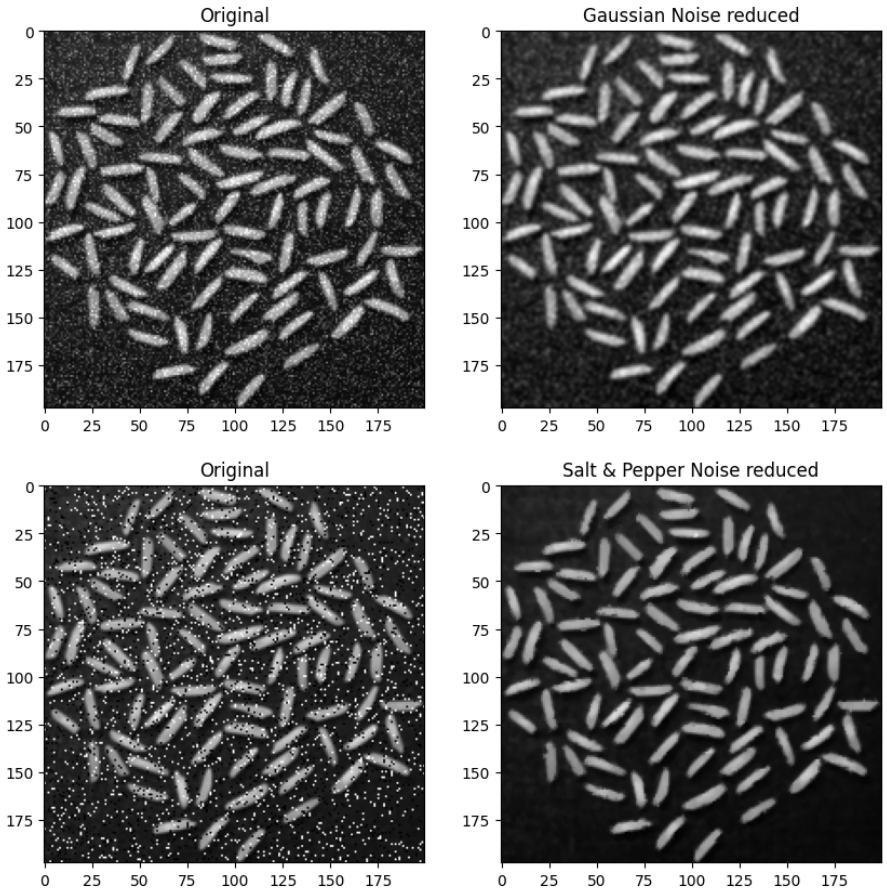
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```




## Question NO.05

```
import cv2 as cv
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount ('/content/drive')
image1 =cv.imread("/content/drive/MyDrive/images/rice_gaussian_noise.png", cv.IMREAD_GRAYSCALE)
image2 =cv.imread("/content/drive/MyDrive/images/rice_salt_pepper_noise.png", cv.IMREAD_GRAYSCALE)
assert image is not None
k = 3
Noise_reduced1 = cv.GaussianBlur(image1, (k,k), 0)
Noise_reduced2 = cv.medianBlur(image2, k)

fig, ax = plt.subplots(2,2,figsize = (10,10))
ax[0,0].imshow(image1, cmap = 'gray')
ax[0,0].set_title('Original')
ax[0,1].imshow(Noise_reduced1, cmap = 'gray')
ax[0,1].set_title('Gaussian Noise reduced')
ax[1,0].imshow(image2, cmap = 'gray')
ax[1,0].set_title('Original')
ax[1,1].imshow(Noise_reduced2, cmap = 'gray')
ax[1,1].set_title('Salt & Pepper Noise reduced')
plt.show()
```

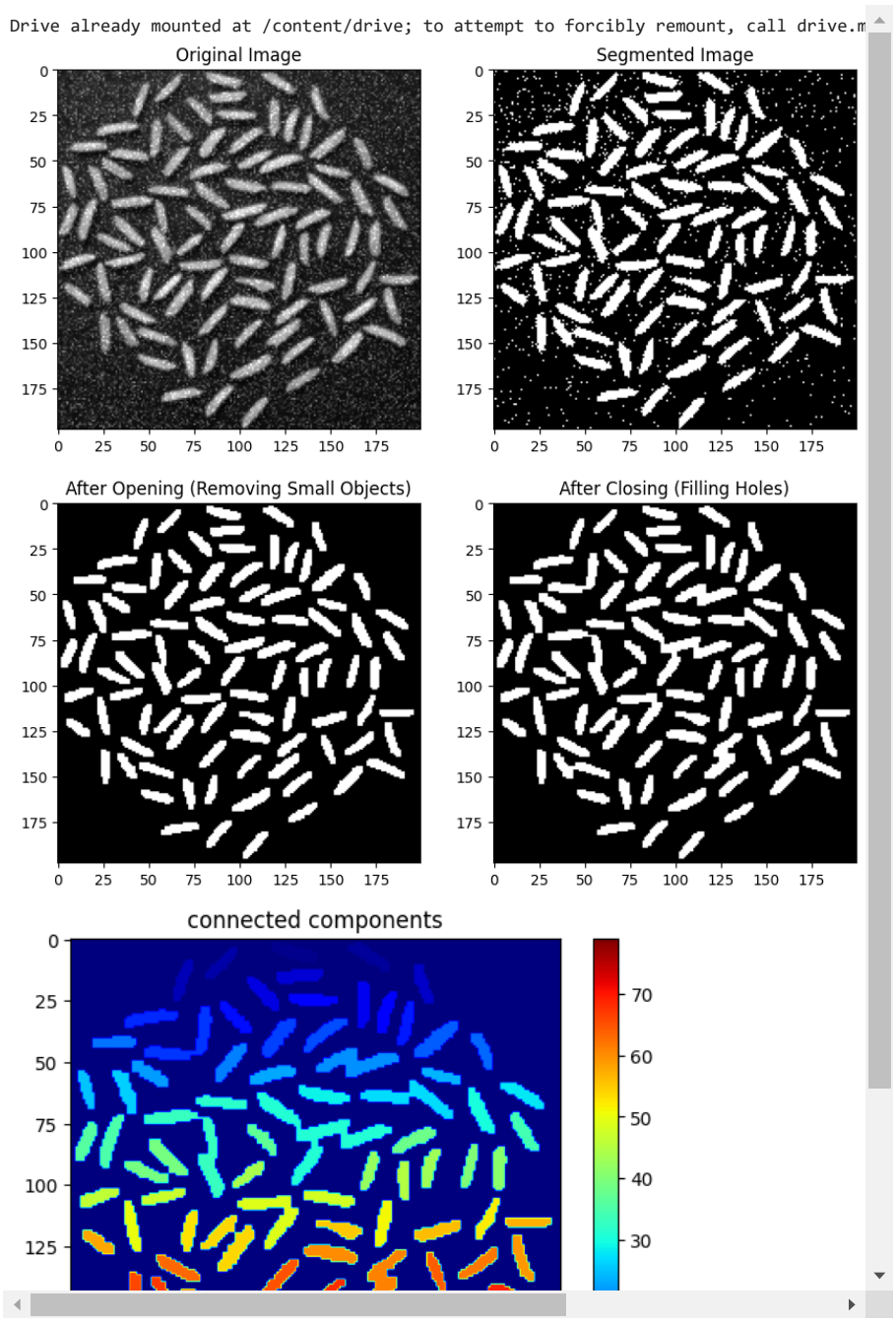
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```




```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount ('/content/drive')
# Load the image
im1 = cv.imread("/content/drive/MyDrive/images/rice_gaussian_noise.png", cv.IMREAD_GRAYSCALE)
# Threshold the image using Otsu's method
_, im2 = cv.threshold(im1, 0, 255, cv.THRESH_BINARY + cv.THRESH_OTSU)
k = 3
# Apply morphological opening to remove small objects
kernel = np.ones((k,k), np.uint8)
im3 = cv.morphologyEx(im2, cv.MORPH_OPEN, kernel)
# Apply morphological closing to fill holes
im4 = cv.morphologyEx(im3, cv.MORPH_CLOSE, kernel)
# Plot original, segmented, opened, and closed images
fig, ax = plt.subplots(2, 2, figsize=(10,10))
ax[0,0].imshow(im1, cmap='gray')
ax[0,0].set_title('Original Image')
ax[0,1].imshow(im2, cmap='gray')
ax[0,1].set_title('Segmented Image')
ax[1,0].imshow(im3, cmap='gray')
ax[1,0].set_title('After Opening (Removing Small Objects)')
ax[1,1].imshow(im4, cmap='gray')
ax[1,1].set_title('After Closing (Filling Holes)')
plt.show()
num_labels, labeled_images, stats, centroids = cv.connectedComponentsWithStats(im4, connectivity = 8)
plt.imshow(labeled_images, cmap = 'jet') #Display results
plt.colorbar()
plt.title('connected components')
plt.show()
```

```
num_rice_grains = num_labels
print("Number of rice grains = ", num_rice_grains)
```

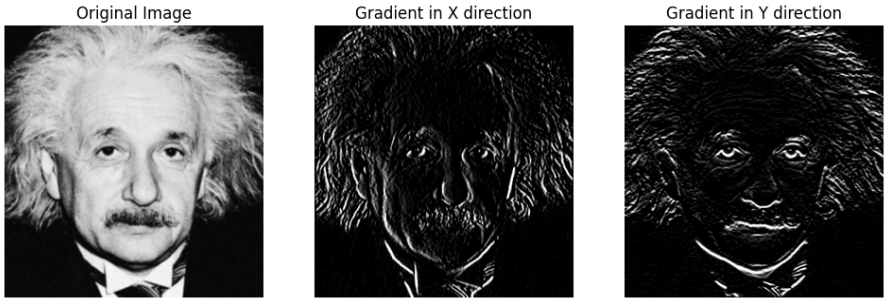
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.m
```




## Question 6

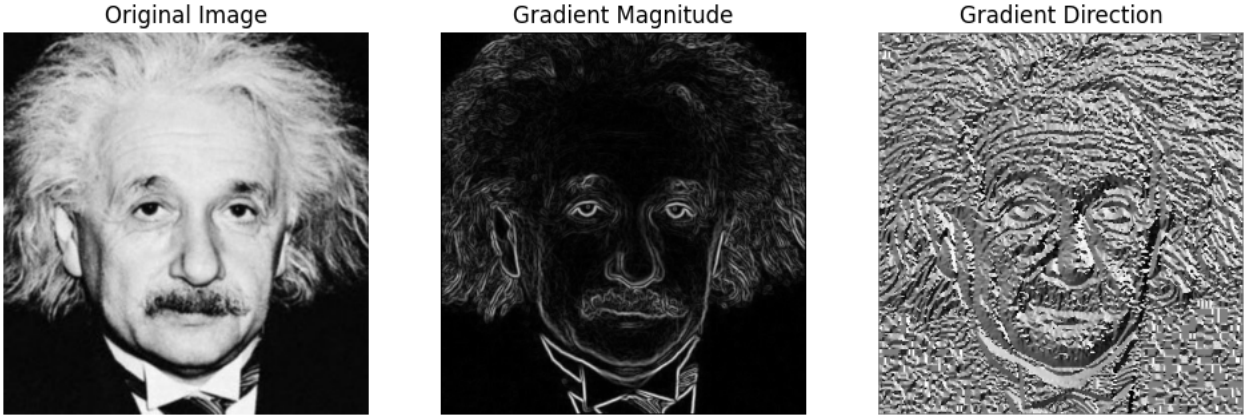
```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount("/content/drive")
image = cv.imread('/content/drive/MyDrive/images/einstein.png', cv.IMREAD_GRAYSCALE)
assert image is not None # Check if the image is loaded successfully
# Define the Sobel filter kernels
sobel_x = np.array([[-1, 0, 1],
[-2, 0, 2],
[-1, 0, 1]])
sobel_y = np.array([[-1, -2, -1],
[0, 0, 0],
[1, 2, 1]])
# Apply the Sobel operator using filter2D
gradient_x = cv2.filter2D(image, -1, sobel_x)
gradient_y = cv2.filter2D(image, -1, sobel_y)
# Compute the magnitude of gradients
gradient_magnitude = np.sqrt(np.square(gradient_x) + np.square(gradient_y))
# Display the results
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 3, 2)
plt.imshow(gradient_x, cmap='gray')
plt.title('Gradient in X direction')
plt.axis('off')
plt.subplot(1, 3, 3)
plt.imshow(gradient_y, cmap='gray')
plt.title('Gradient in Y direction')
plt.axis('off')
plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```



```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount("/content/drive")
image = cv.imread('/content/drive/MyDrive/images/einstein.png', cv.IMREAD_GRAYSCALE)
assert image is not None # Check if the image is loaded successfully
sobel_x = np.array([[-1, 0, 1],
[-2, 0, 2],
[-1, 0, 1]])
sobel_y = np.array([[-1, -2, -1],
[0, 0, 0],
[1, 2, 1]])
# Initialize gradient images
gradient_x = np.zeros_like(image, dtype=np.float32)
gradient_y = np.zeros_like(image, dtype=np.float32)
# Apply Sobel filter using nested loops
rows, cols = image.shape
for i in range(1, rows - 1):
  for j in range(1, cols - 1):
    gx = np.sum(sobel_x * image[i-1:i+2, j-1:j+2])
    gy = np.sum(sobel_y * image[i-1:i+2, j-1:j+2])
    gradient_x[i, j] = gx
    gradient_y[i, j] = gy
# Compute magnitude and direction of gradients
gradient_magnitude = np.sqrt(gradient_x**2 + gradient_y**2)
gradient_direction = np.arctan2(gradient_y, gradient_x)
# Display results
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 3, 2)
plt.imshow(gradient_magnitude, cmap='gray')
plt.title('Gradient Magnitude')
plt.axis('off')
plt.subplot(1, 3, 3)
plt.imshow(gradient_direction, cmap='gray')
plt.title('Gradient Direction')
plt.axis('off')
plt.show()
```

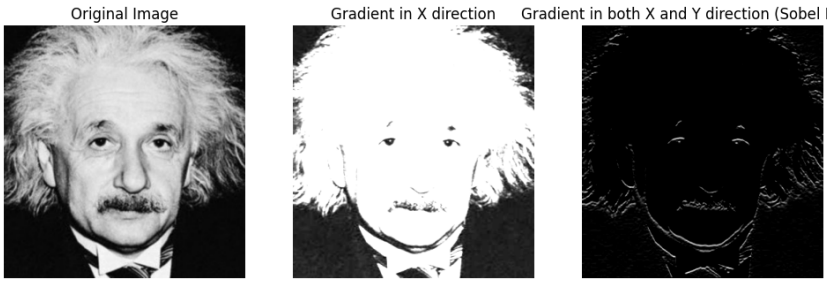
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```




```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount("/content/drive")
image = cv.imread('/content/drive/MyDrive/images/einstein.png', cv.IMREAD_GRAYSCALE)
assert image is not None
# Define the separable Sobel kernels
sobel_horizontal = np.array([1, 2, 1])
sobel_vertical = np.array([1, 0, -1])
# Convolve image with the horizontal kernel
gradient_x = cv2.filter2D(image, -1, sobel_horizontal.reshape(1, 3))
# Convolve the result with the vertical kernel
gradient_xy = cv2.filter2D(gradient_x, -1, sobel_vertical.reshape(3, 1))
# Display results
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')
plt.subplot(1, 3, 2)
plt.imshow(gradient_x, cmap='gray')
plt.title('Gradient in X direction')
plt.axis('off')
plt.subplot(1, 3, 3)
plt.imshow(gradient_xy, cmap='gray')
plt.title('Gradient in both X and Y direction (Sobel Filtered)')
plt.axis('off')
plt.show()
```

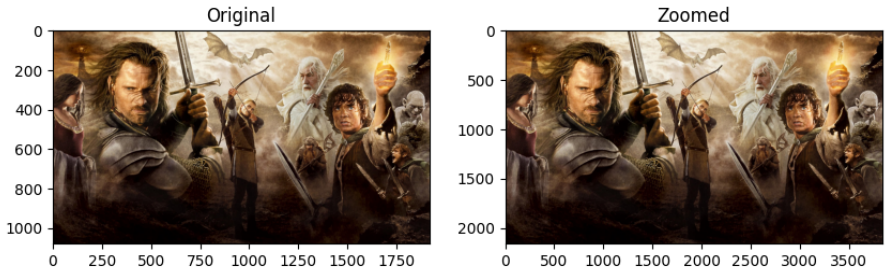
```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```




## Question 7

```
import cv2 as cv
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')

def zoom_image(image, scale_factor):
    # Calculate new dimensions
    new_width = int(image.shape[1] * scale_factor)
    new_height = int(image.shape[0] * scale_factor)
    # Resize the image using nearest-neighbor interpolation
    zoomed_image = cv.resize(image, (new_width, new_height), interpolation=cv.INTER_NEAREST)
    return zoomed_image

# Read the input image
image = cv.imread("/content/drive/MyDrive/images/im01.png", cv.IMREAD_COLOR)
assert image is not None

# Define the scaling factor (0 < s <= 10)
scale_factor = 2.0


if scale_factor <= 0 or scale_factor > 10:
    print("Error: Invalid scale factor. Scale factor must be in the range (0, 10].")
else:
    # Zoom the image
    zoomed_image = zoom_image(image, scale_factor)

    # Display the original and zoomed images
    fig, ax = plt.subplots(1, 2, figsize=(10, 10))
    ax[0].imshow(cv.cvtColor(image, cv.COLOR_BGR2RGB))
    ax[0].set_title('Original')
    ax[1].imshow(cv.cvtColor(zoomed_image, cv.COLOR_BGR2RGB))
    ax[1].set_title('Zoomed')
    plt.show()
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```




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

from google.colab import drive
drive.mount("/content/drive")
image = cv2.imread('/content/drive/MyDrive/images/im01.png', cv2.IMREAD_COLOR)
assert image is not None # Check if the image is loaded successfully

s = 2
height, width = image.shape[:2]  # Access only the first two elements of the shape tuple

new_width = width * s
new_height = height * s

interpolated_image = np.zeros((new_height, new_width, 3), dtype=np.uint8)  # Initialize with 3 channels

for y in range(new_height):
    for x in range(new_width):
        src_x = x * width // new_width
        src_y = y * height // new_height

        x0 = int(src_x)
        x1 = min(x0 + 1, width - 1)
        y0 = int(src_y)
        y1 = min(y0 + 1, height - 1)

        dx = src_x - x0
        dy = src_y - y0

        interpolated_values = (1 - dx) * (1 - dy) * image[y0, x0] + \
                              dx * (1 - dy) * image[y0, x1] + \
                              (1 - dx) * dy * image[y1, x0] + \
                              dx * dy * image[y1, x1]

        interpolated_image[y, x] = interpolated_values

fig, ax = plt.subplots(1, 2, figsize = (10,10))
```