Lab 3

Name: Faiza

Student ID: 2021-1-60-089

Course code: CSE438

Course title: Digital Image Processing

Section: 2

1. Using the following image, solve questions a - f.

a) Read and show the image.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import cv2
image_path = "/kaggle/input/faiza10lab3/Picture1.jpg"

img = cv2.imread(image_path)
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img)
plt.axis("off")
plt.show()
```

Output:

b) Show the matrix form of the image.

Code:

```
img = cv2.imread(image_path)

# Convert from BGR → RGB (for correct color display)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

# Display the image
plt.imshow(img_rgb)
plt.axis("off")
plt.title("Uploaded Image")
plt.show()

# Show matrix details
print("Image shape:", img_rgb.shape)   # (height, width, 3)
print("Data type:", img_rgb.dtype)     # Usually uint8

# Print a small section of the matrix
print("\nMatrix sample (top-left 5x5 pixels):\n", img_rgb[:5, :5])
```

Output:

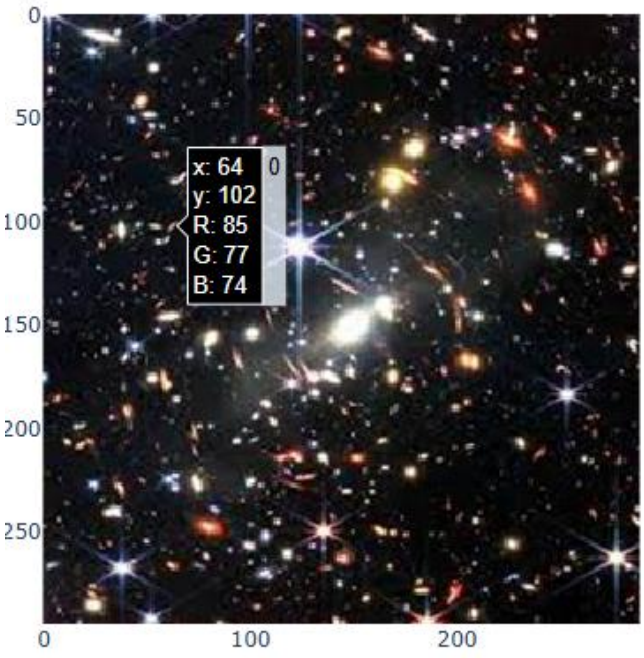
```
Image shape: (295, 289, 3)
Data type: uint8

Matrix sample (top-left 5x5 pixels):
 [[[232 234 255]
  [240 242 255]
  [251 251 255]
  [250 250 255]
  [253 250 255]]

 [[157 159 180]
  [198 200 221]
  [239 239 255]
  [218 218 242]
  [169 166 193]]

 [[ 71  76  98]
  [129 134 156]
  [200 202 227]
  [178 180 205]
  [ 90  90 118]]

 [[ 43  47  72]
  [ 84  88 113]
  [158 162 187]
  [161 163 188]
  [ 79  80 108]]

 [[ 66  72  98]
  [ 74  80 106]
  [117 123 149]
  [117 121 148]
  [ 50  54  81]]]
```

c) Show the pixel information by hovering the cursor on the image.

Code:

```
import plotly.express as px
img = cv2.imread(image_path)
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

fig = px.imshow(img_rgb, title="Hover over the image to see pixel info")

fig.update_traces(
    hovertemplate="x: %{x}<br>y: %{y}<br>R: %{customdata[0]}<br>G: %{customdata[1]}<br>B: %{customdata[2]}",
    customdata=np.reshape(img_rgb, (img_rgb.shape[0], img_rgb.shape[1], 3))
)

fig.show()
```

Output:



d) Find the value of the pixel (10, 78).

Code:

```
img = cv2.imread(image_path)

img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

x, y = 78, 10

pixel_value = img_rgb[y, x]

print(f"Pixel value at (10, 78) [R, G, B]: {pixel_value}")
```

Output:

```
Pixel value at (10, 78) [R, G, B]: [23 25 24]
```

e) Show the size of the image.

Code:

```
height, width, channels = img.shape

print(f"Image size: {width} x {height}")
print(f"No. of color channels: {channels}")
```

Output:

```
Image size: 289 x 295
No. of color channels: 3
```

f) Show the all the information of the image.

Code:

```
img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

height, width, channels = img_rgb.shape
total_pixels = height * width

dtype = img_rgb.dtype

min_vals = img_rgb.min(axis=(0,1))
max_vals = img_rgb.max(axis=(0,1))
mean_vals = img_rgb.mean(axis=(0,1))

x, y = 78, 10
```

```
pixel_value = img_rgb[y, x]

print("=== Image Information ===")
print(f"Dimensions: {width} x {height}")
print(f"Number of color channels: {channels}")
print(f"Total pixels: {total_pixels}")
print(f"Data type: {dtype}")
print(f"Pixel value at (10,78) [R,G,B]: {pixel_value}")
print(f"Minimum per channel [R,G,B]: {min_vals}")
print(f"Maximum per channel [R,G,B]: {max_vals}")
print(f"Mean per channel [R,G,B]: {mean_vals}")
```

Output:

```
=== Image Information ===
Dimensions: 289 x 295
Number of color channels: 3
Total pixels: 85255
Data type: uint8
Pixel value at (10,78) [R,G,B]: [23 25 24]
Minimum per channel [R,G,B]: [0 0 0]
Maximum per channel [R,G,B]: [255 255 255]
Mean per channel [R,G,B]: [38.68856959 35.76892851 37.44262507]
```

2. Using the following images, solve questions a - i.

RGB Image

Grayscale Image

Indexed Image

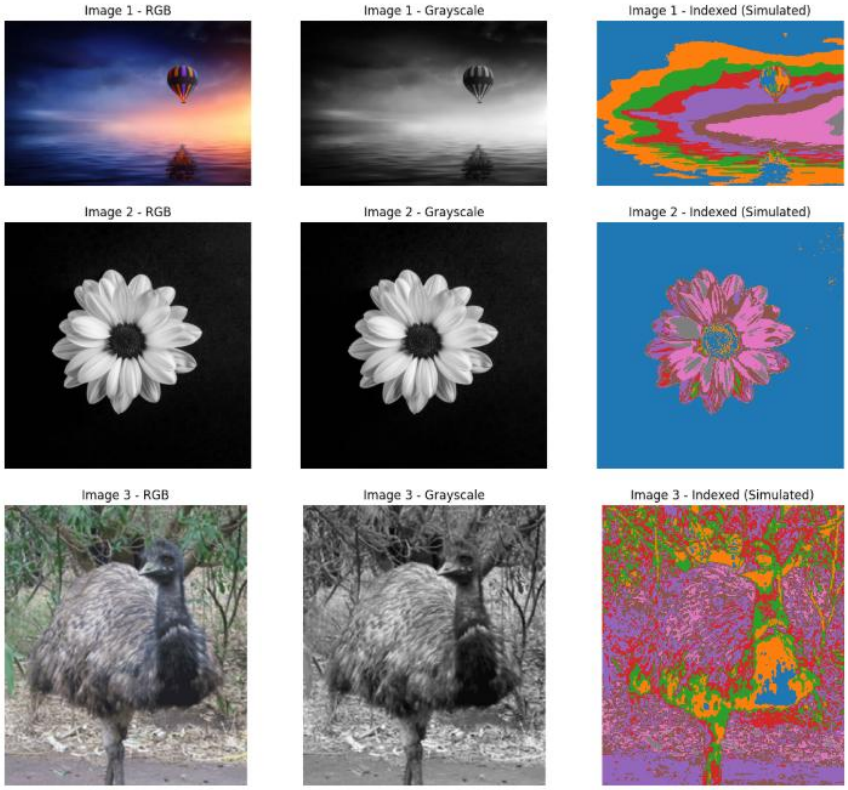
a) Read and show all three types of images (RGB, Grayscale, and Indexed).

Code:

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

```
from matplotlib.colors import ListedColormap

# List of image paths
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

# Loop through each image
for idx, path in enumerate(image_paths, start=1):
    # Read RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert to Grayscale
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

    # Simulate Indexed Image (quantize grayscale to 8 levels)
    indexed_image = np.floor_divide(gray_image, 32)  # values 0–7
    colormap = ListedColormap(plt.cm.tab10.colors[:8])

    # Display images
    plt.figure(figsize=(15, 5))

    plt.subplot(1, 3, 1)
    plt.imshow(rgb_image)
    plt.title(f"Image {idx} - RGB")
    plt.axis('off')

    plt.subplot(1, 3, 2)
    plt.imshow(gray_image, cmap='gray')
    plt.title(f"Image {idx} - Grayscale")
    plt.axis('off')

    plt.subplot(1, 3, 3)
    plt.imshow(indexed_image, cmap=colormap, vmin=0, vmax=7)
    plt.title(f"Image {idx} - Indexed (Simulated)")
    plt.axis('off')

    plt.show()
```

Output:



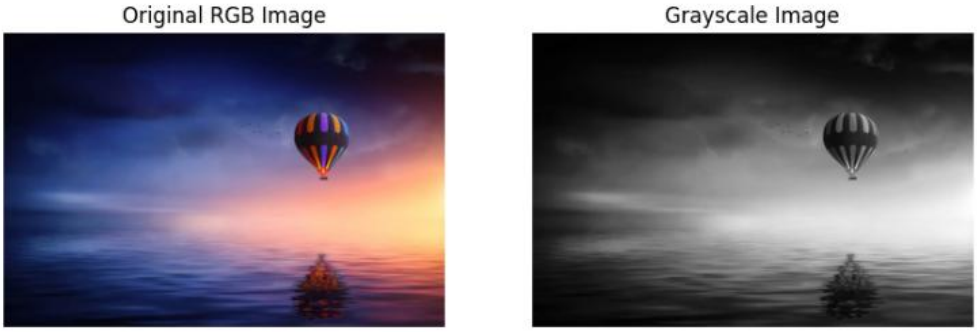
b) Turn the RGB image to Grayscale image.

Code:

```
image_path = r'/kaggle/input/faiza10lab3/Picture2.png'
rgb_image = cv2.imread(image_path)

# Check if image is loaded
if rgb_image is None:
    print("Error: Could not load image.")
else:
    # Convert BGR (OpenCV default) → RGB for correct display
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert RGB → Grayscale
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

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

    # Original RGB
    plt.subplot(1, 2, 1)
    plt.imshow(rgb_image)
```

```
plt.title("Original RGB Image")
plt.axis('off')

# Grayscale
plt.subplot(1, 2, 2)
plt.imshow(gray_image, cmap='gray')
plt.title("Grayscale Image")
plt.axis('off')

plt.show()
```

Output:



c) Turn the Indexed image to Grayscale image.

Code:

```
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'//kaggle/input/faiza10lab3/Picture4.tif'
]

# Loop through images
plt.figure(figsize=(15, 5))
for idx, path in enumerate(image_paths, start=1):
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    # Convert BGR → RGB for correct color display
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert RGB → Grayscale
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)
```

```
    # Display Grayscale image
    plt.subplot(1, 3, idx)
    plt.imshow(gray_image, cmap='gray')
    plt.title(f"Image {idx} - Grayscale")
    plt.axis('off')

plt.show()
```

Output:

d) Turn the Indexed image to RGB image.

Code:

```
# List of image paths
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

# Create a colormap for Indexed → RGB
colormap = np.array(plt.cm.tab10.colors)[:, :3]  # RGB only, 0-1 range

plt.figure(figsize=(15, 5))

for idx, path in enumerate(image_paths, start=1):
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert RGB → Grayscale
```

```
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

    # Simulate Indexed Image (quantize grayscale to 8 levels)
    indexed_image = np.floor_divide(gray_image, 32).astype(int)  # 0-7

    # Convert Indexed → RGB
    indexed_rgb = colormap[indexed_image]  # shape: H x W x 3, RGB in 0-1

    # Display
    plt.subplot(1, 3, idx)
    plt.imshow(indexed_rgb)
    plt.title(f"Image {idx} - RGB from Indexed")
    plt.axis('off')

plt.show()
```

Output:

e) Convert the Grayscale image to a Binary image.

Code:

```
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

plt.figure(figsize=(15, 5))

for idx, path in enumerate(image_paths, start=1):
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    # Convert BGR → RGB
```

```
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert RGB → Grayscale
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

    # Convert Grayscale → Binary
    # Using a threshold value of 128 (can be adjusted)
    _, binary_image = cv2.threshold(gray_image, 128, 255,
cv2.THRESH_BINARY)

    # Display Binary image
    plt.subplot(1, 3, idx)
    plt.imshow(binary_image, cmap='gray')
    plt.title(f"Image {idx} - Binary")
    plt.axis('off')

plt.show()
```

Output:

f) Show the inverted form of that Binary image.

Code:

```
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

plt.figure(figsize=(15, 5))

for idx, path in enumerate(image_paths, start=1):
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue
```

```
    # Convert BGR → RGB
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert RGB → Grayscale
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

    # Convert Grayscale → Binary
    _, binary_image = cv2.threshold(gray_image, 128, 255,
cv2.THRESH_BINARY)

    # Invert Binary image
    inverted_binary = cv2.bitwise_not(binary_image)

    # Display Inverted Binary image
    plt.subplot(1, 3, idx)
    plt.imshow(inverted_binary, cmap='gray')
    plt.title(f"Image {idx} - Inverted Binary")
    plt.axis('off')

plt.show()
```

Output:

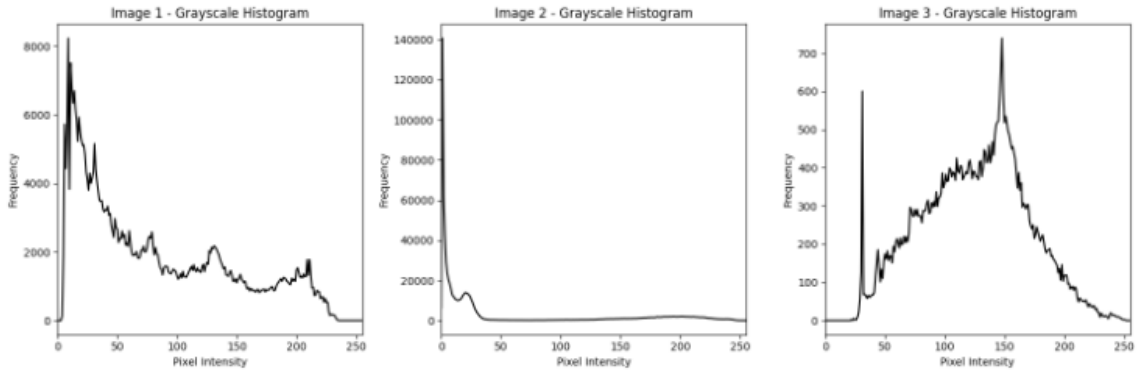
g) Show the histogram of the Grayscale image.

Code:

```
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

plt.figure(figsize=(15, 5))

for idx, path in enumerate(image_paths, start=1):
```

```
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    # Convert BGR → RGB
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Convert RGB → Grayscale
    gray_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2GRAY)

    # Compute histogram
    hist = cv2.calcHist([gray_image], [0], None, [256], [0, 256])

    # Display histogram
    plt.subplot(1, 3, idx)
    plt.plot(hist, color='black')
    plt.title(f"Image {idx} - Grayscale Histogram")
    plt.xlim([0, 255])
    plt.xlabel("Pixel Intensity")
    plt.ylabel("Frequency")

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

Output:



h) Invert the RGB image.

Code:

```
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

plt.figure(figsize=(15, 5))

for idx, path in enumerate(image_paths, start=1):
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    # Convert BGR → RGB
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Invert RGB image
    inverted_rgb = 255 - rgb_image

    # Display inverted image
    plt.subplot(1, 3, idx)
    plt.imshow(inverted_rgb)
    plt.title(f"Image {idx} - Inverted RGB")
    plt.axis('off')

plt.show()
```

Output:

i) Blur the RGB image.

Code:

```
image_paths = [
    r'/kaggle/input/faiza10lab3/Picture2.png',
    r'/kaggle/input/faiza10lab3/Picture3.jpg',
    r'/kaggle/input/faiza10lab3/Picture4.tif'
]

plt.figure(figsize=(15, 5))

for idx, path in enumerate(image_paths, start=1):
    # Load RGB image
    rgb_image = cv2.imread(path)
    if rgb_image is None:
        print(f"Error: Could not load image at {path}")
        continue

    # Convert BGR → RGB
    rgb_image = cv2.cvtColor(rgb_image, cv2.COLOR_BGR2RGB)

    # Apply Gaussian Blur (kernel size 5x5)
    blurred_rgb = cv2.GaussianBlur(rgb_image, (5, 5), 0)

    # Display blurred image
    plt.subplot(1, 3, idx)
    plt.imshow(blurred_rgb)
    plt.title(f"Image {idx} - Blurred RGB")
    plt.axis('off')

plt.show()
```

Output: