```
# Read image
from matplotlib.image import imread
import matplotlib.pyplot as plt
import numpy as np
```

```
input_image = imread("/content/drive/MyDrive/Edge Detection/Image_01_RGB.jpg")
```

```
# Seperating Channels
r,g,b = input_image[:,:,0], input_image[:,:,1], input_image[:,:,2]

gamma = 1.04
r_const, b_const, g_const = 0.2126, 0.7152, 0.0722 #standard equation based on human perception
grayScale_image = r_const*r**gamma + g_const*g**gamma + b_const*b**gamma #for grayscale only one channel
```

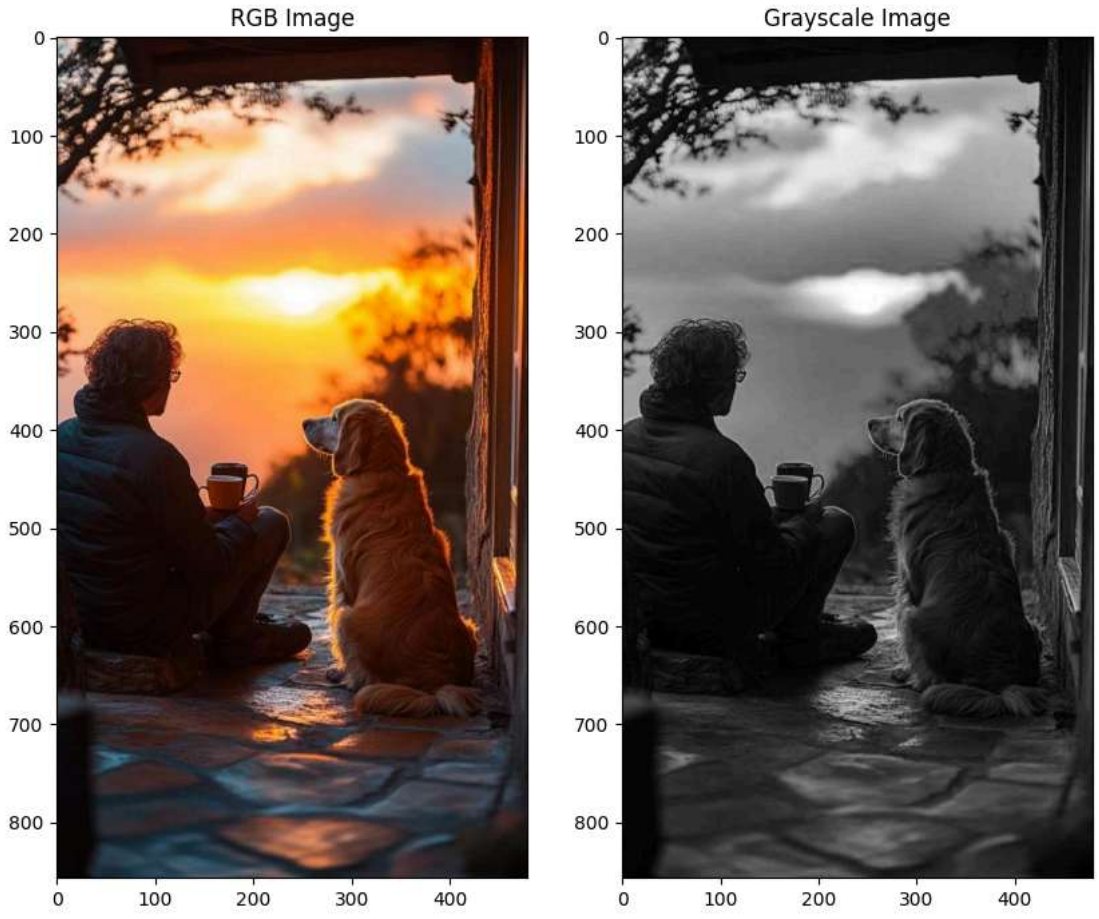
```
fig = plt.figure(figsize=(10,10))

image1, image2 = fig.add_subplot(1,2,1), fig.add_subplot(1,2,2)
image1.imshow(input_image)
image2.imshow(grayScale_image,cmap=plt.cm.get_cmap('gray'))

image1.set_title("RGB Image")
image2.set_title("Grayscale Image")

fig.show()
plt.show()
```

```
/tmp/ipython-input-1035785728.py:5: MatplotlibDeprecationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and wil
  image2.imshow(grayScale_image,cmap=plt.cm.get_cmap('gray'))
```




```
print(f"Matrix Shape: {grayScale_image.shape}")
```

```
Matrix Shape: (857, 480)
```

```
print(grayScale_image)
```

```
[[211.76024305 228.39678736 243.79522455 ...  21.18954926   8.81846961
    8.81846961]
 [227.1153505  232.24275324 237.37451776 ...  21.18954926   8.81846961
    8.81846961]
 [249.21497802 245.35845105 240.22004735 ...  21.18954926   8.81846961
    8.81846961]
 ...
 [  8.20814701   8.20814701   8.20814701 ...   6.60370942   6.60370942
    6.60370942]
 [  8.20814701   8.20814701   8.20814701 ...   6.60370942   6.60370942
    6.60370942]
 [  7.08422399   7.08422399   7.08422399 ...   6.60370942   6.60370942
    6.60370942]]
```

```
# Saving the image

np.save("matrix.npy", grayScale_image)
```

```
def grayscale_to_hex(grayscale_array, output_hex_file):

    # Convert to uint8 (0-255 range)

    if grayscale_array.dtype != np.uint8:

        normalized = np.clip(grayscale_array, 0, 255)
        grayscale_uint8 = normalized.astype(np.uint8)
    else:
        grayscale_uint8 = grayscale_array


    height, width = grayscale_uint8.shape
    pixels = grayscale_uint8.flatten()

    with open(output_hex_file, 'w') as f:
        for pixel in pixels:
            f.write(f"{pixel:02X}\n")



    # Create parameter file for Verilog
    param_file = output_hex_file.replace('.hex', '_params.txt')
    with open(param_file, 'w') as f:
        f.write(f"// Verilog parameters for image\n")
        f.write(f"parameter WIDTH = {width};\n")
        f.write(f"parameter HEIGHT = {height};\n")
        f.write(f"parameter TOTAL_PIXELS = {len(pixels)};\n")
        f.write(f"parameter ADDR_BITS = {int(np.ceil(np.log2(len(pixels))))};\n")


    return grayscale_uint8


# Convert the grayscale image to .hex file
hex_image = grayscale_to_hex(grayScale_image, "image01.hex")
```

```
good = 0
bad = 0
with open("/content/drive/MyDrive/Edge Detection/matrix.hex","r") as f:
    for line in f:
        s = line.strip()
        if not s:
            continue
        if s.lower().startswith("0x"):
            s = s[2:]
        try:
            int(s,16)
            good += 1
        except:
```

```
            bad += 1

print("good lines:", good, "bad lines:", bad)
```

```
good lines: 411360 bad lines: 0
```

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

# Set your expected dimensions
N, M = 480, 857  # width, height

def hex_to_img(path, width, height, strict=False):
    vals = []
    bad_lines = 0
    with open(path, "r") as f:
        for line in f:
            s = line.strip()
            if not s:
                continue
            if s.lower().startswith("0x"):
                s = s[2:]
            try:
                v = int(s, 16)
                vals.append(v & 0xFF)  # keep 8-bit
            except:
                bad_lines += 1

    arr = np.array(vals, dtype=np.uint8)
    total = width * height

    if strict and len(arr) != total:
        raise ValueError(f"{path}: got {len(arr)} samples, expected {total}")

    # pad/trim so we can always display
    if len(arr) < total:
        arr = np.pad(arr, (0, total - len(arr)), constant_values=0)
    else:
        arr = arr[:total]

    img = arr.reshape(height, width)
    return img, len(vals), bad_lines

# ---- Replace with your uploaded filenames ----
hex1 = "/content/drive/MyDrive/Edge Detection/image01.hex"
hex2 = "/content/drive/MyDrive/Edge Detection/output_pixels.hex"

img1, n1, b1 = hex_to_img(hex1, N, M)
img2, n2, b2 = hex_to_img(hex2, N, M)

plt.figure(figsize=(14, 6))

plt.subplot(1, 2, 1)
plt.title(f"Using High Level Language-Reference Image")
plt.imshow(img1, cmap="gray", vmin=0, vmax=255)
plt.axis("off")

plt.subplot(1, 2, 2)
plt.title(f"Using Edge Detection Algorithm ")
plt.imshow(img2, cmap="gray", vmin=0, vmax=255)
plt.axis("off")

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

Using High Level Language-Reference Image

Using Edge Detection Algorithm

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.