IVP-Computer Assignment 1

Abirami Sivakumar(as16288)

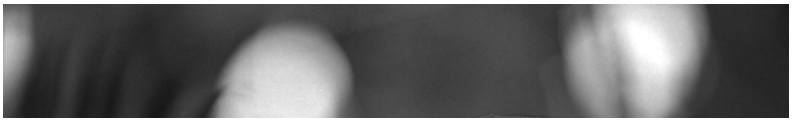
```
import matplotlib.pyplot as plt
import numpy as np
import cv2
from google.colab.patches import cv2_imshow
```

(a) Load an RGB color image and display the RGB image as well as the R, G, and B channels separately. Describe what you observe in each component.

```
img=cv2.imread('/content/color image.png')
cv2_imshow(img)
```

```
blue = img[:, :, 0]
green = img[:, :, 1]
red = img[:, :, 2]
```

```
cv2_imshow(blue)
```



Areas with strong blue have higher intensity in this image. The regions in blue in the original image are highlighted in a white tone.

```
cv2_imshow(green)
```

Areas with strong green have higher intensity in this image. The regions in green in the original image are highlighted in a white tone.

```
cv2_imshow(red)
```

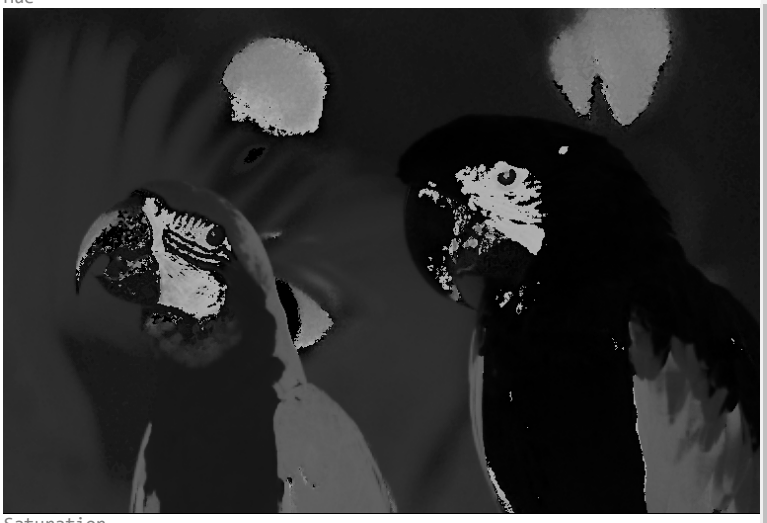
Areas with strong red have higher intensity in this image. The regions in red in the original image are highlighted in a white tone.

(b) Convert the RGB image to the HSV colorspace and display the H, S, and V channels separately. Describe your observations.

```
image_hsv=cv2.cvtColor(img,cv2.COLOR_BGR2HSV)
cv2_imshow(image_hsv)
```

```
#Displaying H,s,V channels separately
h,s,v = cv2.split(image_hsv)
print("Hue")
cv2_imshow(h)
print("Saturation")
cv2_imshow(s)
print("Value")
cv2_imshow(v)
```

Hue



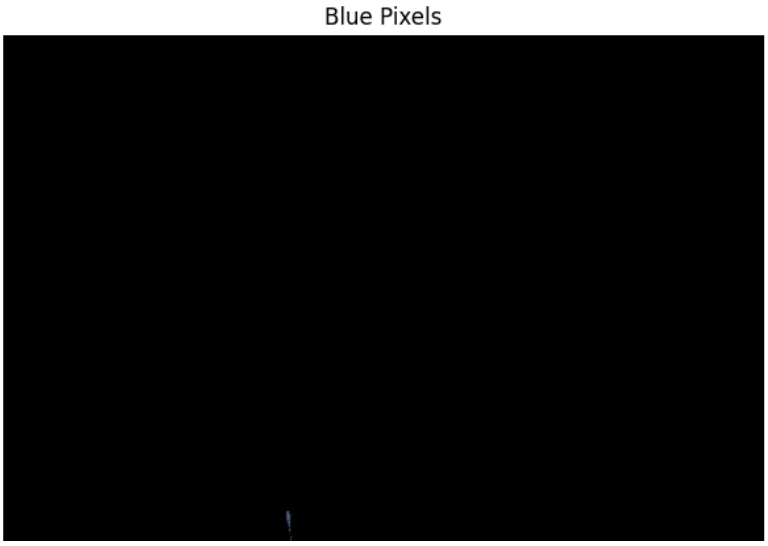
Saturation

Value

In Hue Channel we see variations in color, and objects with different colors will have different intensity values in this channel. In Saturation channel white areas correspond to highly saturated regions, while black areas correspond to desaturated regions. Higher values indicate vivid colours while lower values represent neutral colours. The Value channel shows us variation in brightness in the image.

(c) Detect blue pixels, defined by having a hue in range [110; 130]. Your detected pixels should be in the form of a binary mask. Use this mask to display only the blue pixels of your original image, with the non-blue pixels displayed as black.

```
# Define the lower and upper bounds of the blue hue range
lower_blue = np.array([110, 50, 50])
upper_blue = np.array([130, 255, 255])

# Create a binary mask for the blue pixels
blue_mask = cv2.inRange(image_hsv, lower_blue, upper_blue)

# Create an output image with only the blue pixels displayed
blue_pixels = cv2.bitwise_and(img, img, mask=blue_mask)

# Display the blue pixels on a black background
black_background = np.zeros_like(img)
result = cv2.add(black_background, blue_pixels)

# Display the result
plt.figure(figsize=(12, 5))

plt.subplot(1, 2, 1)
plt.imshow(cv2.cvtColor(result, cv2.COLOR_BGR2RGB))
plt.title('Blue Pixels')
plt.axis('off')

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



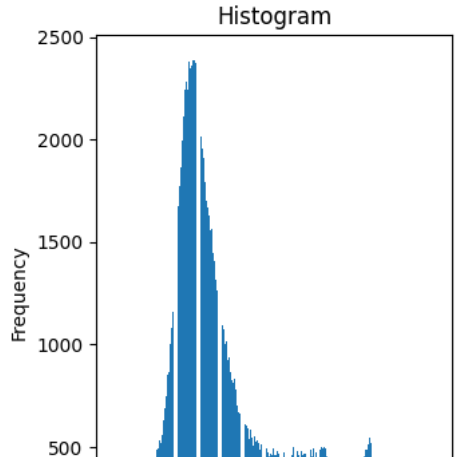
Problem 2

(a) Load and display a low-contrast grayscale image

```
low_img=cv2.imread('/content/low_contrast_image.png',cv2.IMREAD_GRAYSCALE)
cv2_imshow(low_img)
```

(b) Calculate and plot the image's histogram. Comment on the relation between the image's contrast and its histogram

```
hist = np.zeros(256, dtype=int)
for pixel_value in low_img.ravel():
    hist[pixel_value] += 1


# Plot the histogram manually
plt.subplot(1, 2, 2)
plt.bar(range(256), hist)
plt.title('Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')

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

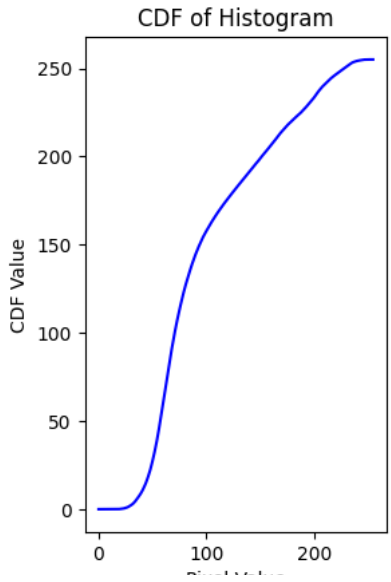


The histogram provides insight into an image's contrast. A well-distributed histogram with prominent peaks indicates good contrast, while a flat or spread-out histogram suggests low contrast.The peaks or spikes in the histogram correspond to the most common pixel intensities in the image. A high-contrast image typically has a histogram with well-defined peaks at both ends of the intensity scale Understanding the histogram can help you make adjustments to improve an image's visual quality.

c) Recall, we derived in lecture that the cumulative distribution function (CDF) of the histogram is the theoretic optimal histogram-equalization transformation function. Plot the CDF of your histogram. Is this an appropriate equalization function? Why or why not?

```
# Calculate the CDF (cumulative distribution function)
cdf = np.cumsum(hist)

# Normalize the CDF to range [0, 255]
cdf_normalized = ((cdf - cdf.min()) * 255) / (cdf.max() - cdf.min())

plt.subplot(1, 2, 2)
plt.plot(cdf_normalized, color='b')
plt.title('CDF of Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('CDF Value')
```

```
Text(0, 0.5, 'CDF Value')
```



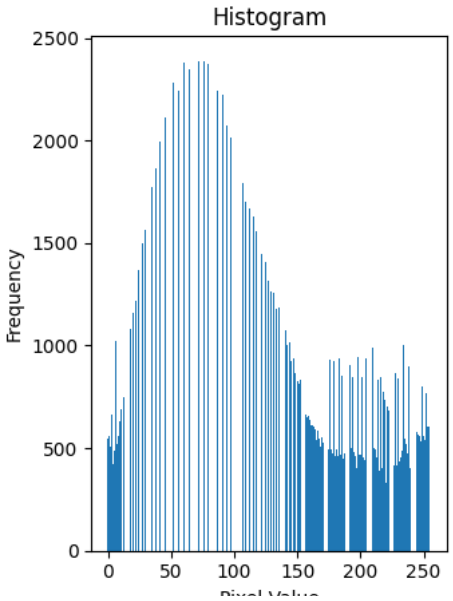
The CDF spans the full range of intensity values (0-255). This ensures that the entire intensity spectrum is utilized in the equalized image.The CDF is also monotonically increasing, meaning that as pixel values increase, their corresponding CDF values also increases.

(d) Apply the CDF as a transformation function to your image1. Display your transformed image and its histogram. Has your image's histogram been equalized? Comment on the your original image and histogram vs. the transformed image and histogram.

```
transformed_img=cdf_normalized[low_img]
cv2_imshow(transformed_img)
transformed_img
```

```
array([[ 10.91505236,  27.09852915, 101.01501933, ..., 196.55021664,
        197.21578876, 200.32564556],
       [ 27.09852915,  48.95340704, 160.83062934, ..., 194.30081411,
        192.76322692, 196.55021664],
       [ 41.75597956,  91.03639225, 194.9878563 , ..., 192.76322692,
        191.26362524, 192.76322692],
       ...,
       [ 38.46114986,  52.71727515,  64.23018633, ...,  32.45118231,
         27.09852915,  27.09852915],
       [ 16.6525152 ,  35.38102085,  52.71727515, ...,  29.6848466 ,
         27.09852915,  24.62286514],
       [ 18.43453086,  52.71727515,  68.12782948, ...,  29.6848466 ,
```

```
#Displaying Transformed image's histogram
hist = np.zeros(256, dtype=int)
for pixel_value in transformed_img.ravel():
    hist[int(pixel_value)] += 1
# Plot the histogram manually
plt.subplot(1, 2, 2)
plt.bar(range(256), hist)
plt.title('Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```



The Histogram equalized image has better contrast and the pixel intensity has been spread out to have a wider range. The dark and light regions appear to be more balanced. This is a good histogram equalization technique. If we want to further improve it we can use adaptive equalization.