

**De La Salle University- Manila**

**Gokongwei College of Engineering**

Lab Activity Number : Drawings and Histograms

Lab Activity Title : 2

Date Performed : September 17, 2024

Date Submitted : September 24, 2024

Instructor : Jay Robert B. del Rosario, PhD

Name of Student : Collado, Andrei Vincent V.

Student ID : 12149616

Subject / Section : LBYCPF3 - EQ1

Remarks: (do not write anything here)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Please delete those marked in RED and change the font color into black.

**Objectives:**

. To gain proficiency in using OpenCV's drawing functions, such as cv2.line(), cv2.rectangle(), and cv2.circle(), to create and manipulate shapes on a digital canvas. 2. To compute and interpret both grayscale and color histograms for images, using OpenCV for data processing and Matplotlib for visualization, enhancing understanding of pixel intensity distributions.

**Part I. Simulations**

**Example 1: drawing.py**

Codes:

import numpy as np

import cv2

canvas = np.zeros((300, 300, 3), dtype="uint8")

green = (0, 255, 0)

red = (0, 0, 255)

blue = (255, 0, 0)

cv2.line(canvas, (0, 0), (300, 300), green)

cv2.imshow("Canvas", canvas)

cv2.waitKey(0)

cv2.line(canvas, (300, 0), (0, 300), red, 3)

cv2.imshow("Canvas", canvas)

cv2.waitKey(0)

cv2.rectangle(canvas, (10, 10), (60, 60), green)

cv2.imshow("Canvas", canvas)

cv2.waitKey(0)

# Red rectangle with a thickness of 5

cv2.rectangle(canvas, (50, 200), (200, 225), red, 5)

cv2.imshow("Canvas", canvas)

cv2.waitKey(0)

# Blue filled rectangle

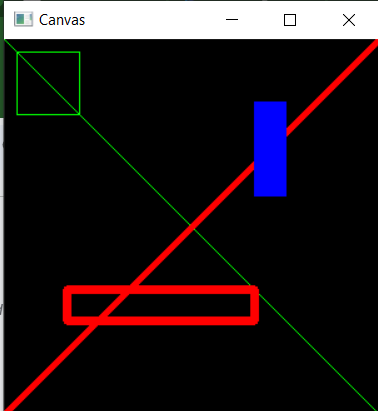
cv2.rectangle(canvas, (200, 50), (225, 125), blue, -1)

cv2.imshow("Canvas", canvas)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output Screenshot:**

****

**Example 2: grayscale\_histogram.py**

Codes:

from matplotlib import pyplot as plt

import cv2

image\_path = r'Examples\test.jpg'

# Load the image

image = cv2.imread(image\_path)

# Check if the image was loaded properly

if image is None:

print(f"Error: Could not load image from {image\_path}")

else:

# Convert to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Show the original grayscale image

cv2.imshow("Original", gray\_image)

# Compute the grayscale histogram

hist = cv2.calcHist([gray\_image], [0], None, [256], [0, 256])

# Plot the grayscale histogram

plt.figure()

plt.title("Grayscale Histogram")

plt.xlabel("Bins")

plt.ylabel("# of Pixels")

plt.plot(hist)

plt.xlim([0, 256])

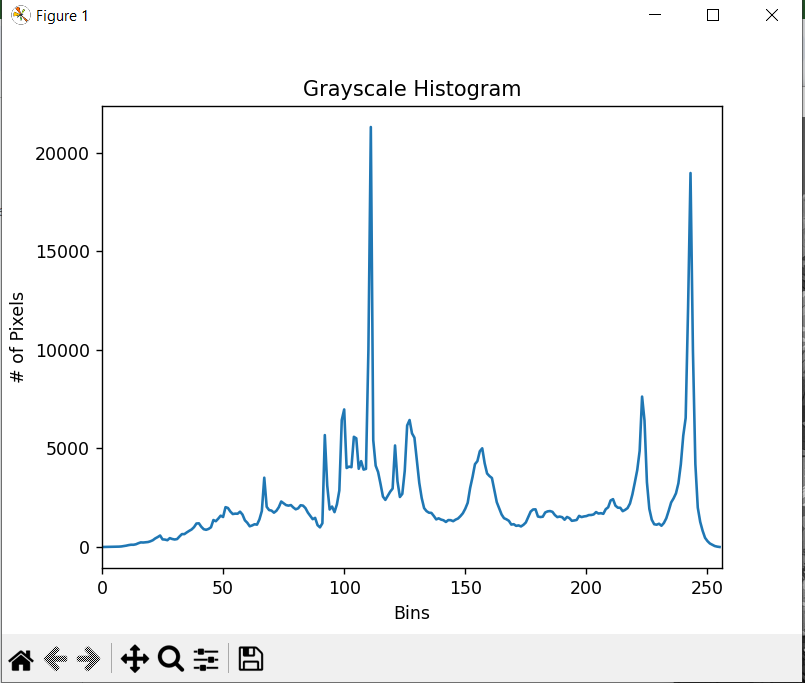
plt.show()

# Wait for a keypress and close the windows

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output Screenshot:**

****

****

**Example 3: color\_histograms.py**

**Codes:**

from matplotlib import pyplot as plt

import cv2

image\_path = r'Examples\test.jpg'

# Load the image

image = cv2.imread(image\_path)

# Check if the image was loaded properly

if image is None:

print(f"Error: Could not load image from {image\_path}")

else:

# Show the original image

cv2.imshow("Original", image)

chans = cv2.split(image)

colors = ("b", "g", "r")

# Plot the color histograms

plt.figure()

plt.title("Color Histograms")

plt.xlabel("Bins")

plt.ylabel("# of Pixels")

# Plot histogram for each channel

for (chan, color) in zip(chans, colors):

hist = cv2.calcHist([chan], [0], None, [256], [0, 256])

plt.plot(hist, color=color)

plt.xlim([0, 256])

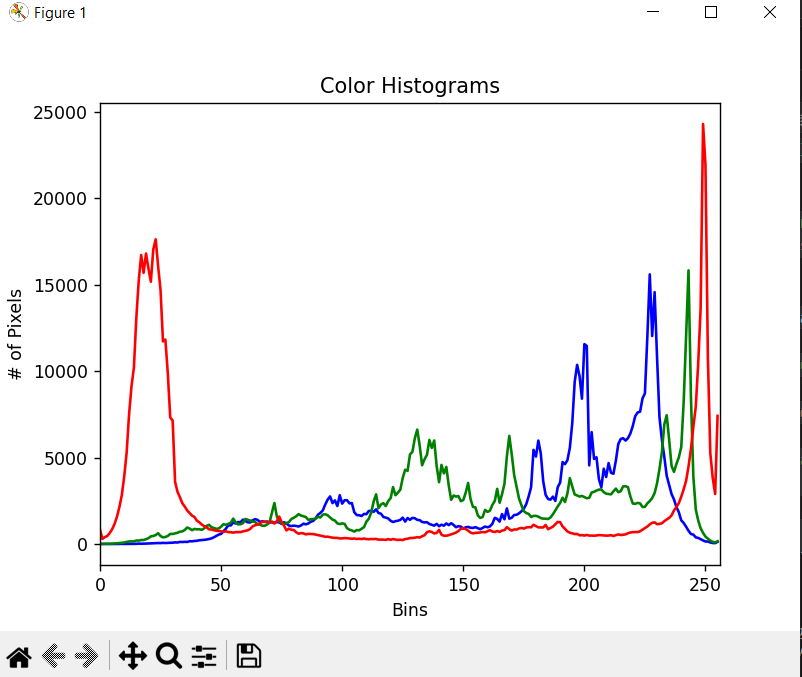
plt.show()

# Wait for a keypress and close the image window

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output Screenshot:f**

****

**Example 4 3d\_histogram.py**

**Codes:**

from mpl\_toolkits.mplot3d import Axes3D

import matplotlib.pyplot as plt

import numpy as np

import cv2

image\_path = r'Examples\test.jpg'

# Load the image

image = cv2.imread(image\_path)

# Check if the image was loaded properly

if image is None:

print(f"Error: Could not load image from {image\_path}")

else:

hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

# Normalize the histogram for plotting

ratio = 5000 / np.max(hist)

# Iterate through the 3D histogram and plot points

for x in range(hist.shape[0]):

for y in range(hist.shape[1]):

for z in range(hist.shape[2]):

if hist[x, y, z] > 0:

ax.scatter(x, y, z, s=hist[x, y, z] \* ratio, color=(x / 7, y / 7, z / 7))

ax.set\_xlabel('Red Channel')

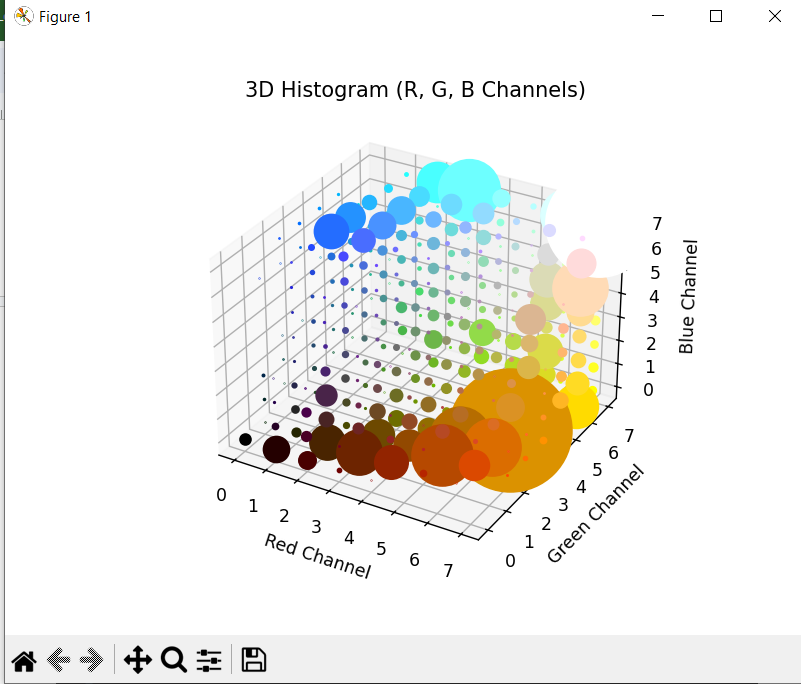
ax.set\_ylabel('Green Channel')

ax.set\_zlabel('Blue Channel')

plt.title("3D Histogram (R, G, B Channels)")

plt.show()

**Output Screenshot:**

****

**Example 5: equalize.py**

**Codes:**

import cv2

import numpy as np

image\_path = r'Examples\test.jpg' # Replace with your image path

image = cv2.imread(image\_path)

if image is None:

print(f"Error: Could not load image from {image\_path}")

else:

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

equalized\_image = cv2.equalizeHist(gray\_image)

result = np.hstack((gray\_image, equalized\_image))

cv2.imshow("Original vs Equalized", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Output Screenshot:  
**

**Part II. Group Activity (4 tasks)**

**Task 1: Create and Draw Shapes Using OpenCV:**

**Codes:**

**Output Screenshot:**

**Discussion/Q and A:**

* Question: How does changing the thickness parameter in the cv2.line() and cv2.rectangle() functions affect the appearance of the shapes? What happens when you use a negative value for the thickness?

**Task 2: Histogram Analysis of an Image:**

**Codes:**

**Output Screenshot:**

**Discussion/Q and A:**

* **Question: What do the peaks and valleys in the histogram indicate about the contrast and brightness of the image? How does the histogram change when you convert an image from color to grayscale?**

**Task 3: 3D Histogram Visualization:**

**Codes:**

**Output Screenshot:**

**Discussion/Q and A:**

* **Question: What insights can you gain from visualizing a 3D histogram about the color distribution in an image?**

**How does this differ from analyzing individual 2D histograms for each color channel?**

* + **answer**

**Part III. Individual Activity**

**Codes**

import cv2

import numpy as np

import matplotlib.pyplot as plt

from mpl\_toolkits.mplot3d import Axes3D

import os

# Function to draw shapes on a blank canvas

def draw\_shapes():

canvas = np.zeros((400, 400, 3), dtype="uint8") # Create a blank canvas (400x400)

# Draw a green line from the top-left to the bottom-right corner

cv2.line(canvas, (0, 0), (400, 400), (0, 255, 0), thickness=5)

# Draw a red rectangle with a thickness of 3 pixels

cv2.rectangle(canvas, (50, 50), (150, 150), (0, 0, 255), thickness=3)

# Draw a filled blue circle with radius 50 pixels

cv2.circle(canvas, (200, 200), 50, (255, 0, 0), thickness=-1)

# Show the canvas

cv2.imshow("Canvas with Shapes", canvas)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Function to compute and display grayscale histogram

def grayscale\_histogram(image):

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

hist = cv2.calcHist([gray\_image], [0], None, [256], [0, 256])

plt.figure()

plt.title("Grayscale Histogram")

plt.xlabel("Pixel Intensity")

plt.ylabel("Frequency")

plt.plot(hist)

plt.xlim([0, 256])

plt.show()

# Function to compute and display color histograms (R, G, B)

def color\_histograms(image):

chans = cv2.split(image) # Split the image into R, G, B channels

colors = ("b", "g", "r")

plt.figure()

plt.title("Color Histograms")

plt.xlabel("Bins")

plt.ylabel("# of Pixels")

for (chan, color) in zip(chans, colors):

hist = cv2.calcHist([chan], [0], None, [256], [0, 256])

plt.plot(hist, color=color)

plt.xlim([0, 256])

plt.show()

# Function to compute and display 2D histogram (Red vs Green)

def two\_d\_histogram(image):

chans = cv2.split(image)

hist = cv2.calcHist([chans[1], chans[2]], [0, 1], None, [32, 32], [0, 256, 0, 256])

plt.imshow(hist, interpolation="nearest")

plt.title("2D Histogram (Green vs Red)")

plt.colorbar()

plt.show()

# Function to compute and display 3D histogram (R, G, B channels)

def three\_d\_histogram(image):

hist = cv2.calcHist([image], [0, 1, 2], None, [8, 8, 8], [0, 256, 0, 256, 0, 256])

fig = plt.figure()

ax = fig.add\_subplot(111, projection='3d')

for i in range(hist.shape[0]):

for j in range(hist.shape[1]):

for k in range(hist.shape[2]):

if hist[i, j, k] > 0:

ax.scatter(i, j, k, s=hist[i, j, k], color=(i / 8, j / 8, k / 8))

ax.set\_xlabel('Red Channel')

ax.set\_ylabel('Green Channel')

ax.set\_zlabel('Blue Channel')

plt.title("3D Histogram (R, G, B Channels)")

plt.show()

# Function to apply and display histogram equalization

def histogram\_equalization(image):

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

eq\_image = cv2.equalizeHist(gray\_image)

result = np.hstack((gray\_image, eq\_image)) # Side-by-side comparison

cv2.imshow("Original vs Equalized", result)

cv2.waitKey(0)

cv2.destroyAllWindows()

# Function to run the workflow

def main():

# Predefined image path

image\_path = r"photo1.png" # Replace with actual image path

# Check if file exists

if not os.path.exists(image\_path):

print(f"Error: File not found at {image\_path}")

return

# Load the image

image = cv2.imread(image\_path)

if image is None:

print("Error: Unable to load the image. Please check the file format.")

return

draw\_shapes()

grayscale\_histogram(image)

color\_histograms(image)

two\_d\_histogram(image)

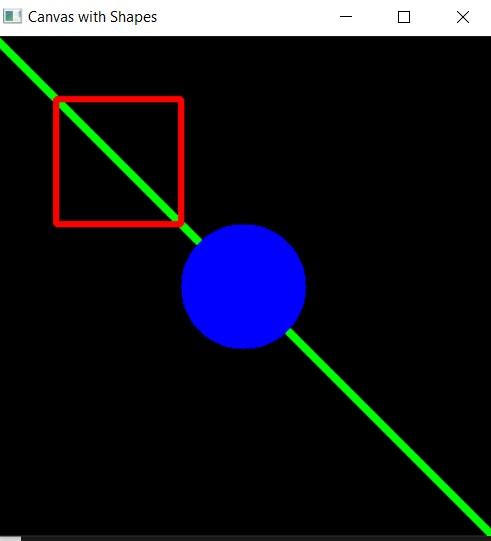
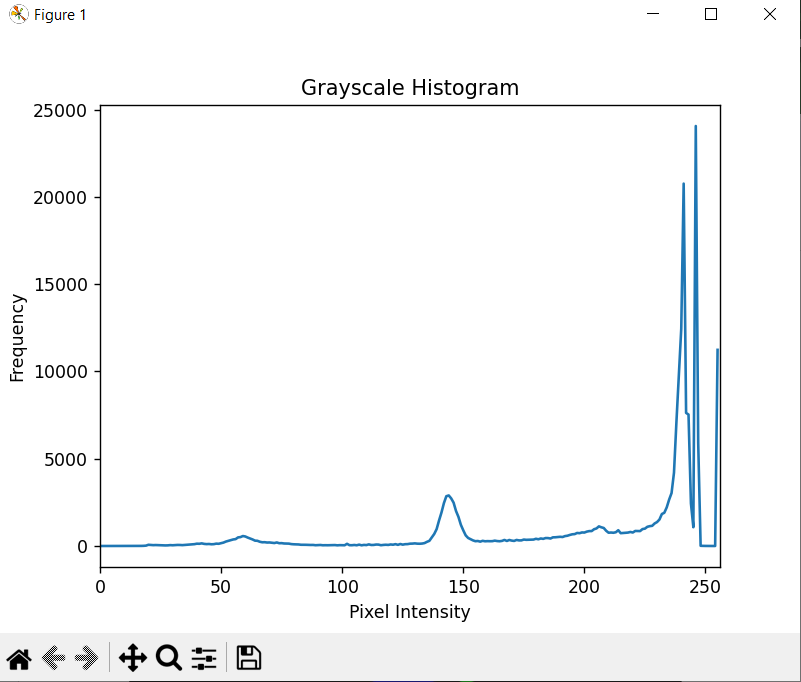
three\_d\_histogram(image)

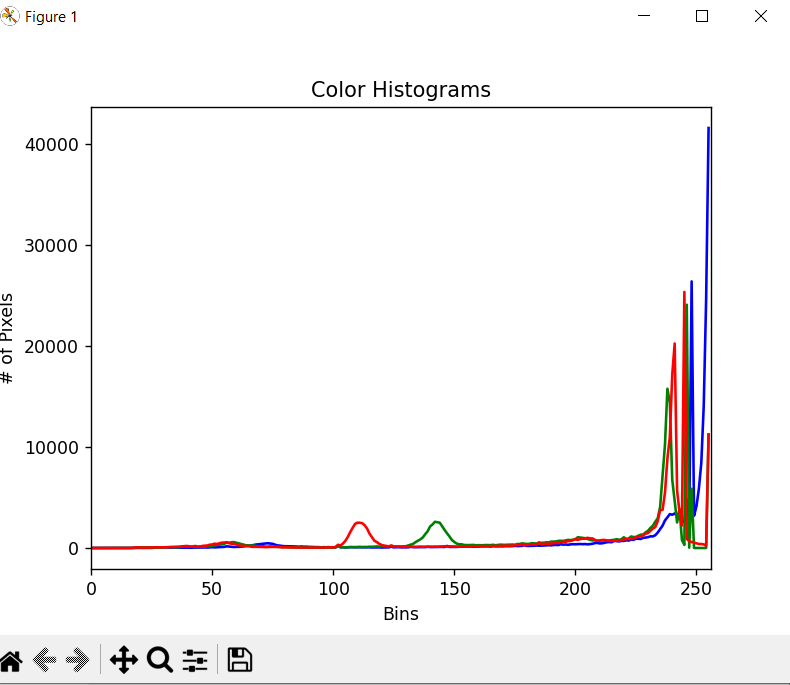
histogram\_equalization(image)

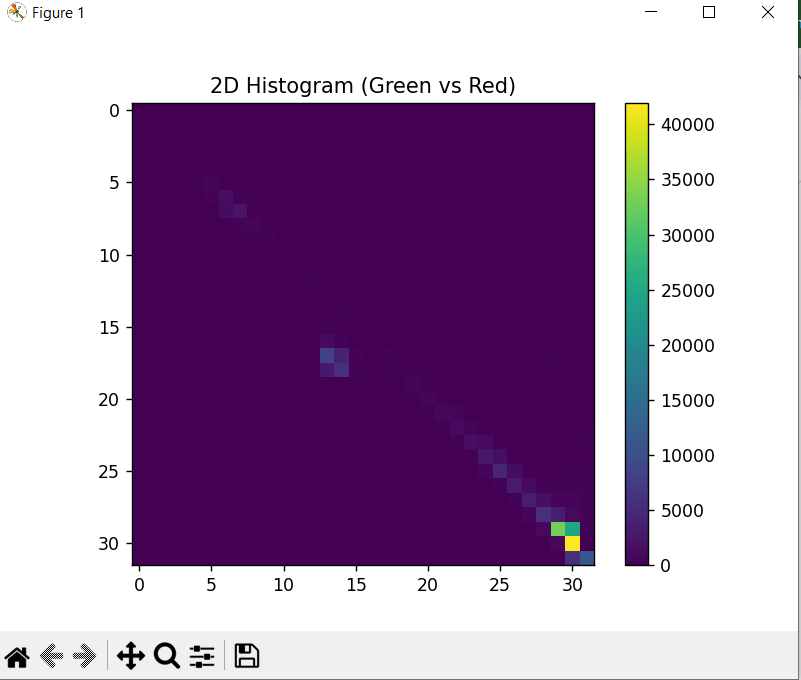
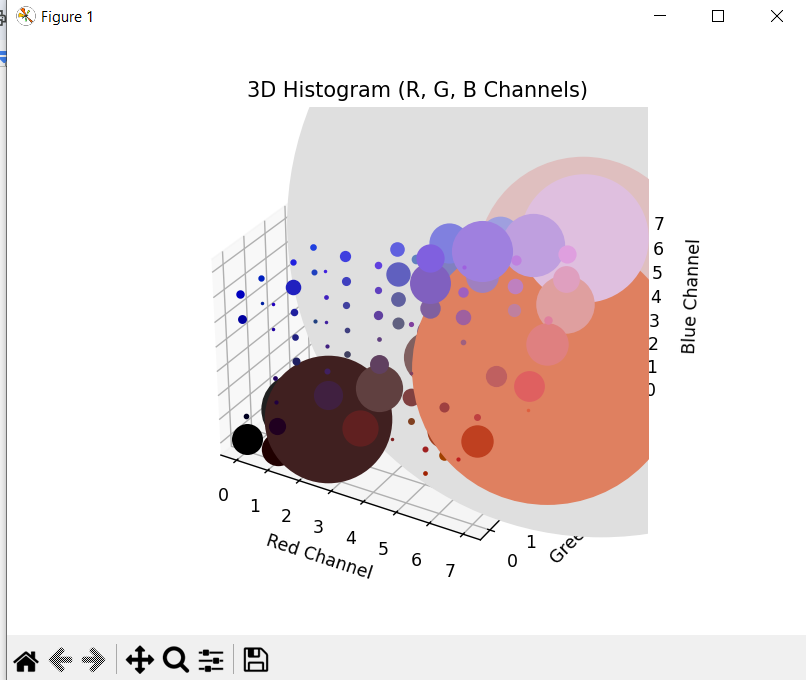
if \_\_name\_\_ == "\_\_main\_\_":

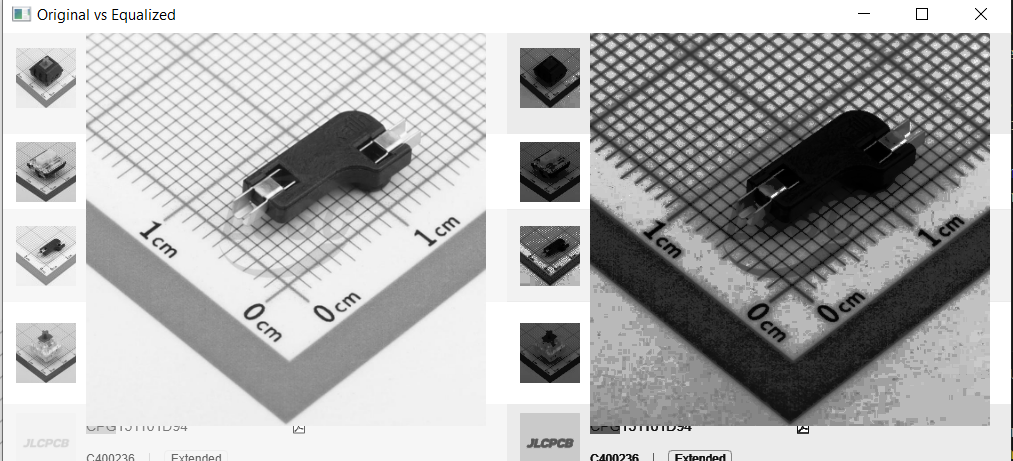
main()

**Output Screenshot:**

**  
**

****

**  
**

****

**Discussion/Q and A:**

* **what challenges did you encounter in ensuring that the**

**different Python scripts and OpenCV functions worked together smoothly? How did you resolve any issues with**

**code compatibility, data formats, or function parameters across the different components?**

Conclusion: