Computer Science and Data Analytics

Course: Introduction to Computer Vision

Student: Ali Asgarov

Supervisor: Dr. Jamal Hasanov

Assignment 2

Digital Images and Color Spaces

Introduction

In this assignment, we explore the field of digital image processing by undertaking a series of experiments focused on color manipulation. These activities aim to deepen our understanding of how digital images can be transformed and analyzed through various techniques. The experiments include:

1. **Grayscale Conversion**: We convert color images to grayscale using two methods: the weighted average based on human perception and a simple average method. This allows us to compare their impacts both visually and quantitatively.

2. **Color Quantization**: Implementing color quantization algorithms to reduce the number of colors in an image, exploring the effects on image quality and file size.

3. **Adjustments of Hue, Saturation, Brightness, and Lightness:** Modifying these color properties within their valid ranges to understand their influence on image appearance.

4. **Color Similarity Using CIEDE Algorithm**: Utilizing the CIEDE color closeness algorithm to identify similar colors within an image, enhancing our understanding of color space and color difference metrics.

This report documents our findings and insights gained from these experiments, contributing to our knowledge of digital image processing and color theory.

Grayscale Conversion

In this experiment we are converting the **RGB** images to the **Grayscale** images with using two methods.

As part of the report, we performed grayscale conversion on the original image using two different methods: the NTSC formula and the averaging method.

NTSC Formula:

This method applies weights to each color channel (red, green, and blue) according to the NTSC standard.

The formula used is: **gray = 0.299 * R + 0.587 * G + 0.114 * B**, where **R**, **G**, and **B** are the red, green, and blue channels of the image, respectively.

After applying the formula, we obtain a grayscale image.

color_closeness.py | grayscale_converter.py

grayscale_converter.py > ...

```python
from PIL import Image
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import argparse

# NTSC formula
def grayscale_NTSC(img):
    np_img = np.array(img)
    gray_image = 0.299 * np_img[:, :, 0] + 0.587 * np_img[:, :, 1] + 0.114 * np_img[:, :, 2]
    return Image.fromarray(np.uint8(gray_image))

# Averaging method
def grayscale_average(img):
    np_img = np.array(img)
    gray_image = np.mean(np_img, axis=2)
    return Image.fromarray(np.uint8(gray_image))


if __name__ == "__main__":
    parser = argparse.ArgumentParser(description='Processing images.')
    parser.add_argument('-i', '--image', type=str, default='input_images/image_0.jpeg', help='Image path')

    # Parsing parameters.
    args = parser.parse_args()
    image_path = f"input_images/{args.image}"

    image = Image.open(image_path)

    image_grayscale_NTSC = grayscale_NTSC(image)
    image_grayscale_average = grayscale_average(image)

    image.show()
    image_grayscale_NTSC.show()
    image_grayscale_average.show()
```

Figure 1. Grayscale convertor screenshot.

Averaging Method:

In this method, we simply take the average of the RGB values for each pixel to obtain the grayscale intensity.

This is calculated as: **gray = (R + G + B) / 3** for each pixel.

The result is a grayscale image where each pixel's intensity is the average of its RGB values.

We applied both methods to the original image, resulting in two grayscale images: one using the NTSC formula (**image_grayscale_NTSC**) and the other using the averaging method (**image_grayscale_average**).

The resulting grayscale images were then displayed alongside the original image for comparison. This visualization allows us to observe the differences in grayscale conversion methods and their impact on the overall appearance of the image.

Example 1. image_7.jpeg:

Figure 2. Original Image

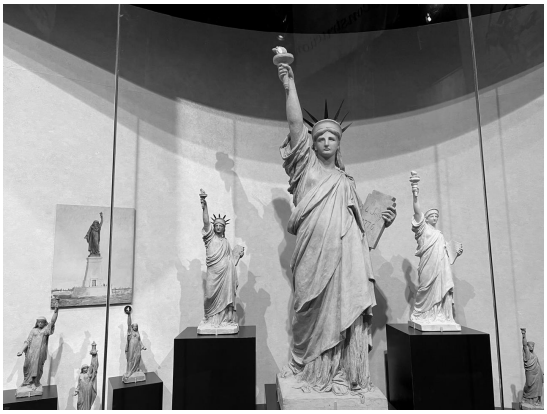


Figure 3. Averaging Method

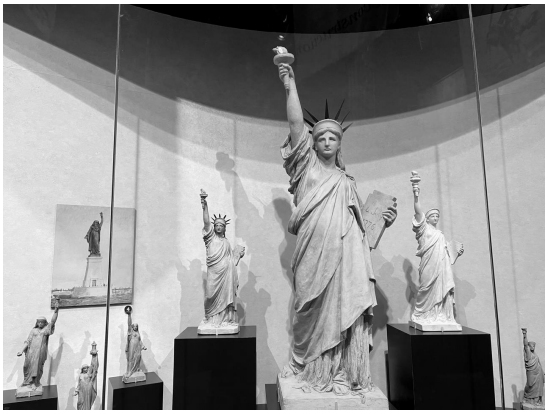


Figure 4. NTSC

Here, the resulting grayscale images were displayed alongside the original image for comparison. This visualization allows us to observe the differences in grayscale conversion methods and their impact on the overall appearance of the image.

Example 2. image_3.jpeg:

Figure 5. Original Image

Figure 6. Averaging Method

Figure 7. NTSC Output

Color Quantization

1. K-means quantization:

Converts the input image into the LAB color space.

Applies K-means clustering to group similar colors into a specified number of clusters.

Assigns each pixel to the nearest cluster centroid.

Converts the clustered LAB values back to RGB color space.

Returns the quantized image.

2. Uniform quantization:

Divides the RGB color space into a specified number of bins.

Assigns each pixel to the closest bin centroid.

Returns the quantized image.

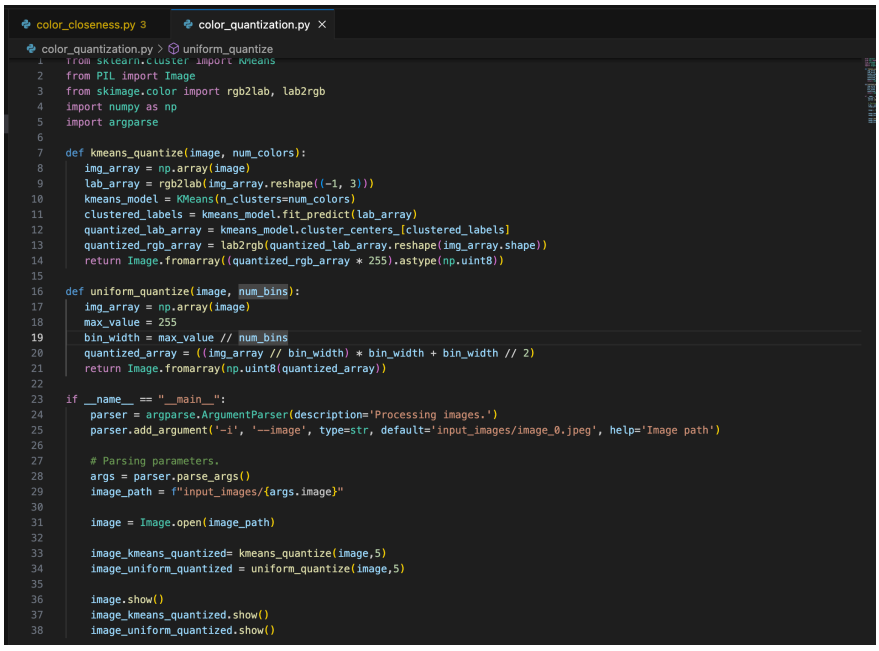


Figure 8. Quantization screenshot.

Example 1. image_4.jpg - number of colors / nbins = 4

Figure 9. Original Image

Figure 10. Uniform Output

Figure 11. KMeans Output

Example 2. image_5.jpg - number of colors / nbins = 4

Figure 12. Original image.

Figure 13. Uniform Quantization

Figure 14. KMeans Output

Hue, Saturation, Brightness and Lightness

Hue Adjustment:

Modifying the hue component of the image, changing the perceived color tone.

The hue adjustment is applied by converting the image to the HSV color space, adjusting the hue value, and then converting it back to RGB.

Saturation Adjustment:

Altering the saturation of colors in the image, affecting their intensity and vividness.

Similar to hue adjustment, it operates in the HSV color space, modifying the saturation value accordingly.

Brightness Adjustment

Adjusting the overall brightness of the image.

Implemented within the HSV color space by modifying the value component, which represents brightness.

Lightness Adjustment

Changing the lightness of colors in the image.

Achieved by converting the RGB image to HLS color space, adjusting the lightness factor, and then converting it back to RGB.

These functionalities offer a convenient way to manipulate the color appearance of images, allowing users to tailor them according to their preferences or specific requirements.

Example 1 - image_0.jpg with 0.1 for hue, 0.1 for saturation and 0.1 for brightness and 10 for lightness adjustment factors.

Figure 19. Original Image

Figure 18. Hue changed image.

Figure 17. Saturation changed image.

Figure 15. Lightening changed image

Figure 16. Brightness changed image.

Color closeness

In this task of the assignment we've implemented a color picker tool and a function to highlight similar colors within an image.

Color Picker Functionality:

We utilize **matplotlib** to display the image and allow users to pick a color by clicking on it. Upon clicking, the RGB value of the picked color is printed.

Highlighting Similar Colors

The **highlight_similar_colors** function takes the picked color and an image as input. It converts the picked color to the LAB color space, as it provides perceptually uniform color differences. Then, it calculates the color difference (Delta E) between the picked color and every pixel in the image. Pixels with a color difference below a certain threshold are considered similar and highlighted in green.

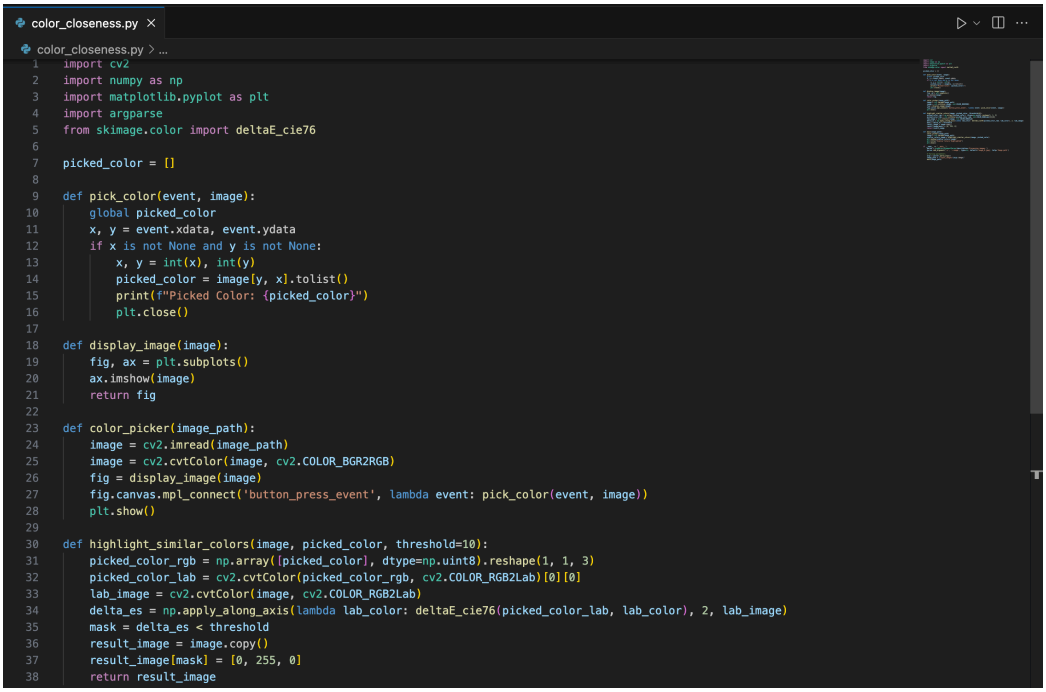
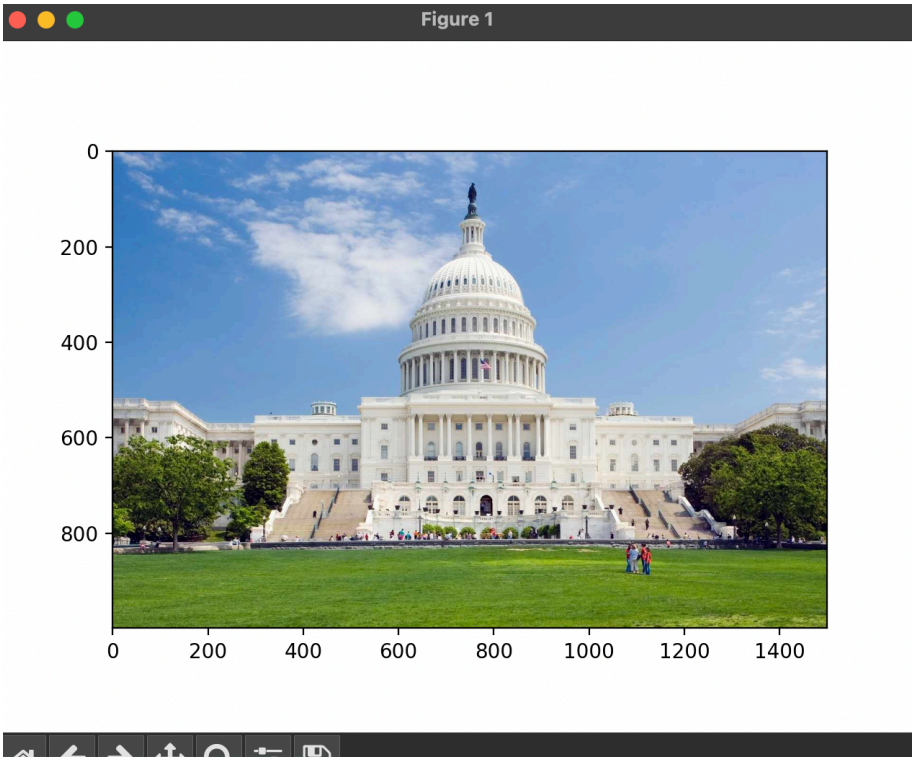


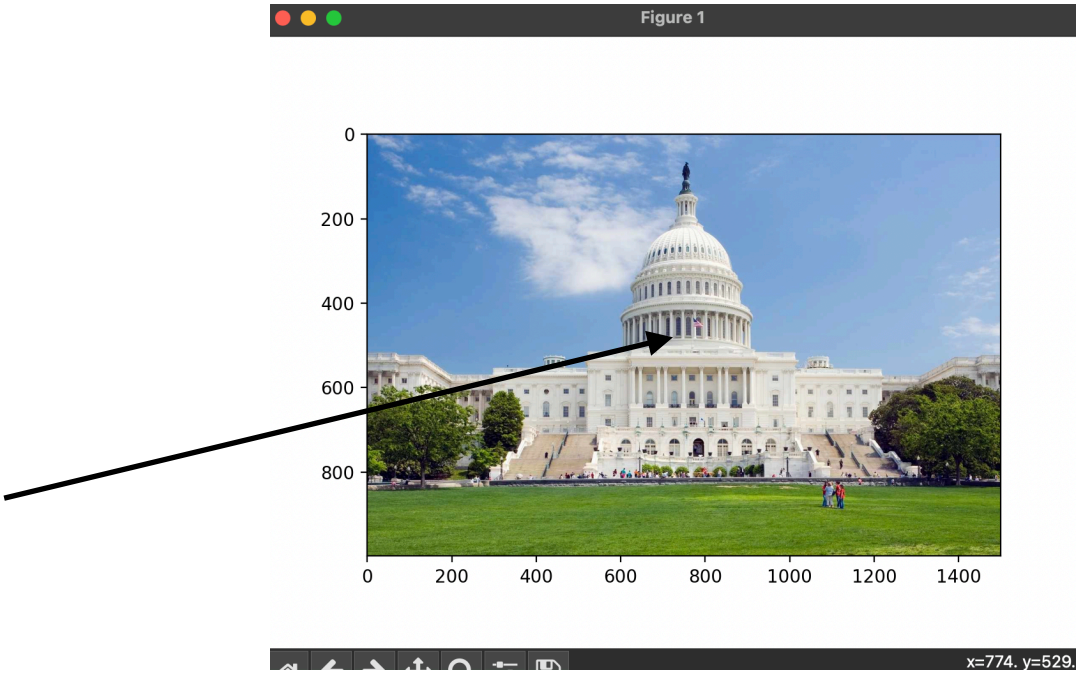
Figure 20. Color closeness.

This task provides a useful tool for visualizing and identifying similar colors within an image.

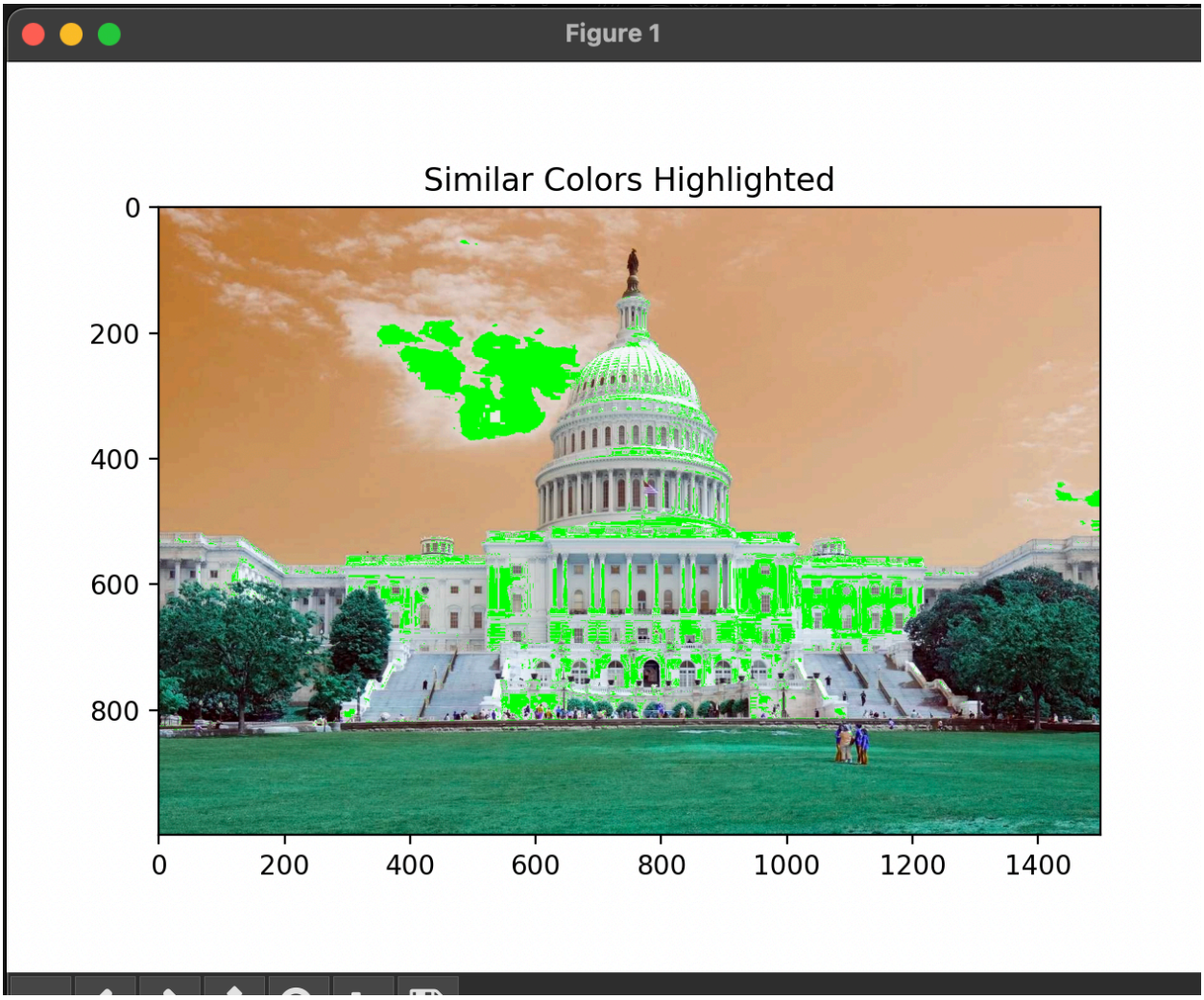
Example 1 - image_0.jpg



Picking color now:

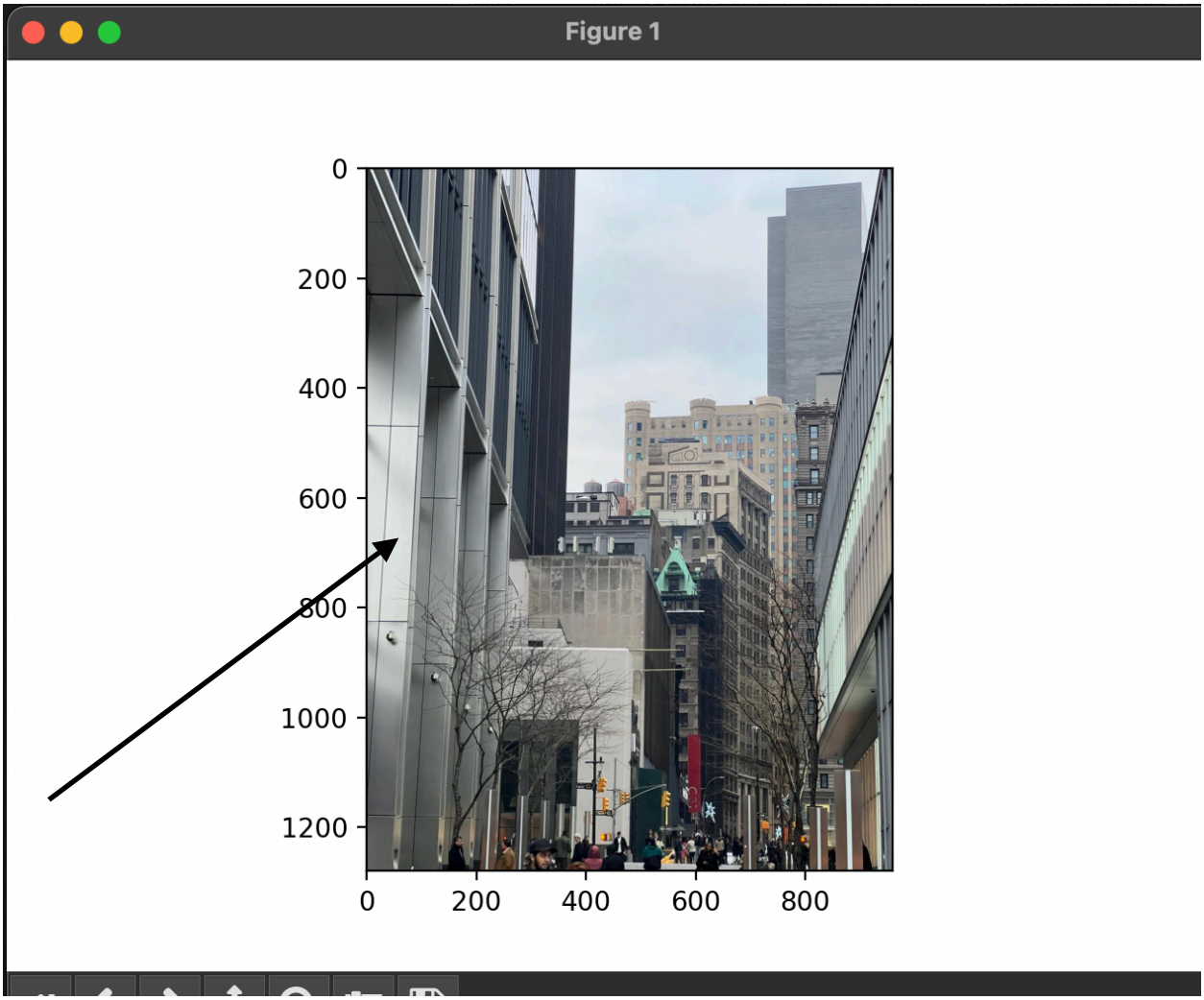


Output:

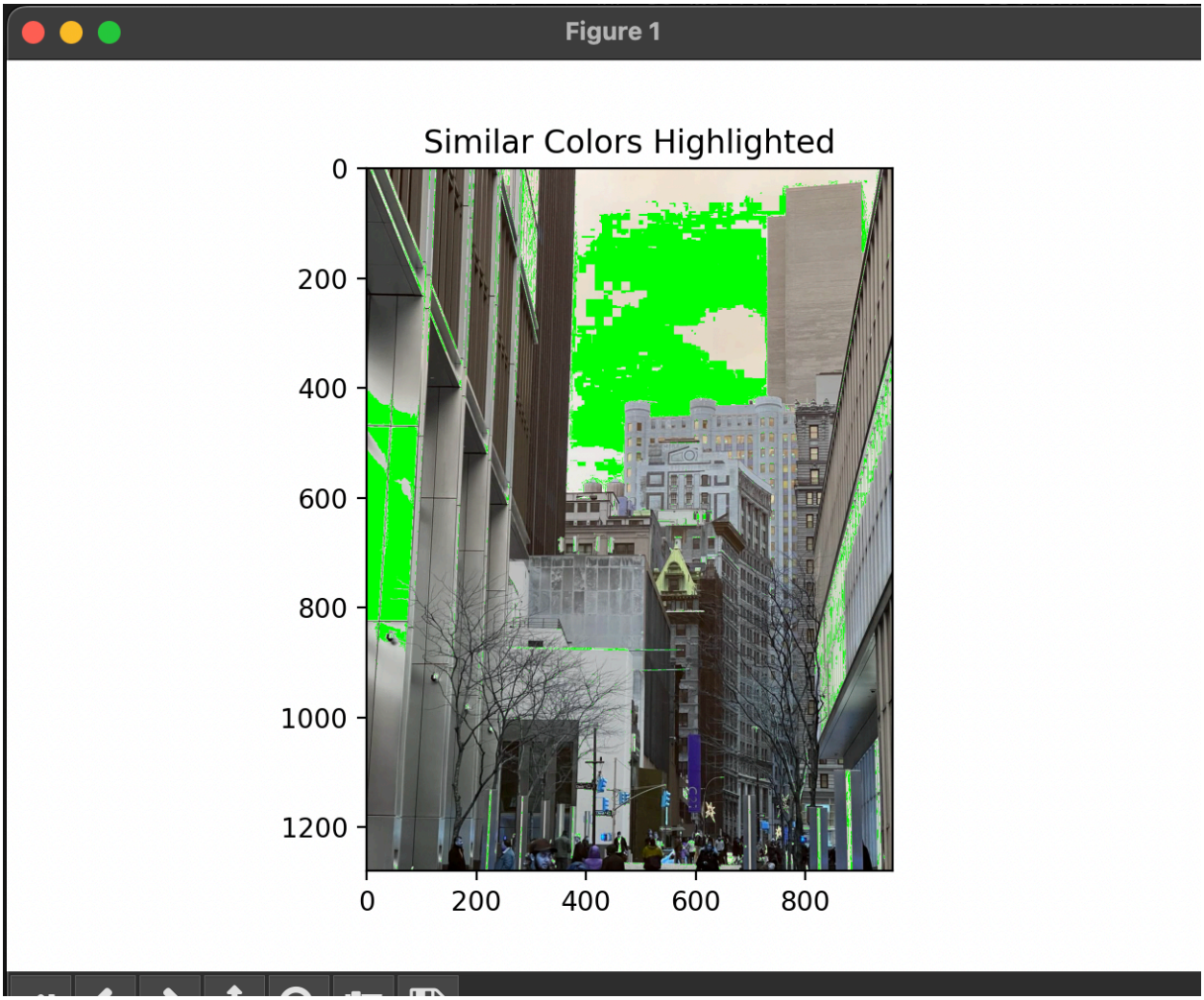


Similar - close colors with the threshold 10 has been spotted with green colors.

One more example.



Output



Conclusion

In this assignment we have learned and experimented how to convert image from RGB to grayscale without doing abstraction but with actual formulas and also we have learned about quantization methods which are very important for image processing. Also we have learned about how to change saturation, hue, value and lightness of the image. We have also experimented how to find similar colors to the pointed color in the image. Overall, assignment was very helpful to understand image processing techniques much more in details.