BCSE417P Fall 2024 Tasks

Nimalan S – 21BAI1291

August 16, 2024

Problem 1

(a) Compute Basic Statistics and Convert an Image into Different Color Spaces.

Code:

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

def process_image(image_path):

    img = cv2.imread(image_path)


    means, stds = cv2.meanStdDev(img)
    means = means.flatten()
    stds = stds.flatten()


    hist_b = cv2.calcHist([img], [0], None, [256], [0, 256])
    hist_g = cv2.calcHist([img], [1], None, [256], [0, 256])
    hist_r = cv2.calcHist([img], [2], None, [256], [0, 256])


    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    lab_img = cv2.cvtColor(img, cv2.COLOR_BGR2Lab)
```

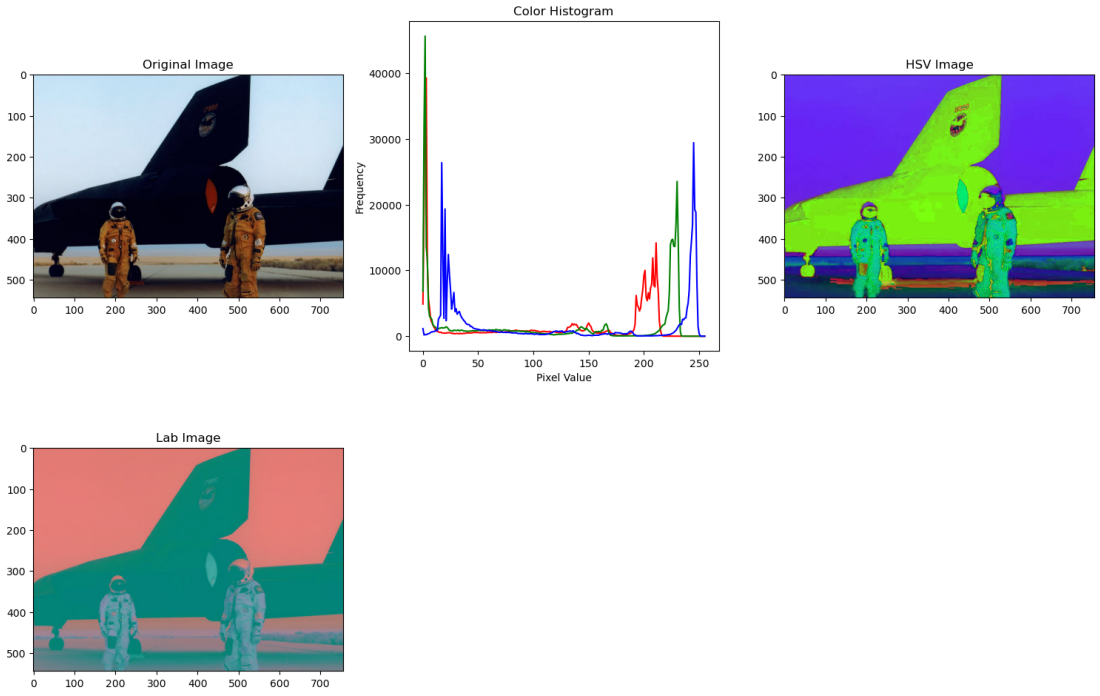
```
    plt.figure(figsize=(15, 10))


    plt.subplot(231)
    plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
    plt.title('Original Image')


    plt.subplot(232)
    plt.plot(hist_r, color='r')
    plt.plot(hist_g, color='g')
    plt.plot(hist_b, color='b')
    plt.title('Color Histogram')
    plt.xlabel('Pixel Value')
    plt.ylabel('Frequency')


    plt.subplot(233)
    plt.imshow(hsv_img)
    plt.title('HSV Image')


    plt.subplot(234)
    plt.imshow(lab_img)
    plt.title('Lab Image')


    print(f"Means: R={means[2]:.2f}, G={means[1]:.2f}, B={means[0]:.2f}")
    print(f"Standard Deviations: R={stds[2]:.2f}, G={stds[1]:.2f}, B={stds[0]:.2f}"

    plt.tight_layout()
    plt.show()

image_path = r"D:\Twitter pics\Twitter\blackbird.jpg"
process_image(image_path)
```

Outputs:

- The mean and standard deviation for each color channel:

```
Means: R=110.25, G=115.03, B=124.55
Standard Deviations: R=87.89, G=98.68, B=101.37
```

- The combined image showing the original image and the outputs in different color spaces:



Observations:

- The mean and standard deviation values provide insights into the brightness and contrast of each color channel.
- The histogram shows the distribution of pixel intensities for each color channel, which can indicate if an image is underexposed or overexposed.
- The HSV color space is useful for color-based segmentation and filtering as it separates chromatic content (color) from intensity. The Lab color space is closer to human vision, which is beneficial for tasks requiring accurate color representation.

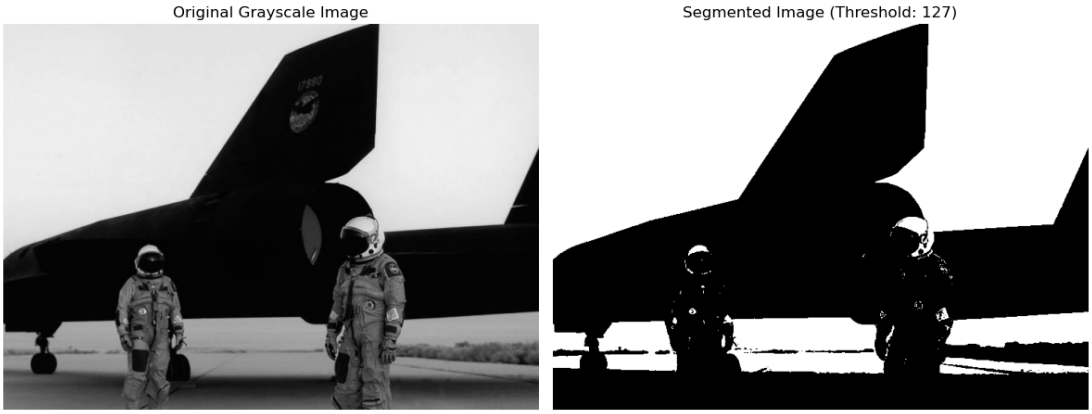
(b) Simple Image Segmentation Using Thresholding.

Code:

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

def segment_image(image_path, threshold):

    with Image.open(image_path) as img:
        original = np.array(img.convert('L'))


    segmented = np.where(original > threshold, 255, 0).astype(np.uint8)


    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    ax1.imshow(original, cmap='gray')
    ax1.set_title('Original Grayscale Image')
    ax1.axis('off')

    ax2.imshow(segmented, cmap='gray')
    ax2.set_title(f'Segmented Image (Threshold: {threshold})')
    ax2.axis('off')

    plt.tight_layout()
    plt.show()

    return original, segmented

image_path = r"D:\Twitter pics\Twitter\blackbird.jpg"
threshold_value = 127

original, segmented = segment_image(image_path, threshold_value)
```

Outputs:

- The original grayscale image and the segmented image after applying global thresholding:



Observations:

- Thresholding is a basic image segmentation technique that divides the image into foreground and background based on a fixed intensity threshold.
- The chosen threshold value of 127 separates the image into two regions: pixels below the threshold are turned black (0), and pixels above the threshold are turned white (255).
- This method works well for images with a clear contrast between foreground and background.

(c) Color-Based Segmentation **Code:**

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

#in hsv red has 2 distinct ranges so we use 2 upper and 2 lower bounds
def segment_image(image_path, lower1, upper1, lower2, upper2):

    original = cv2.imread(image_path)


    hsv = cv2.cvtColor(original, cv2.COLOR_BGR2HSV)


    lower_mask = cv2.inRange(hsv, lower1, upper1)
    upper_mask = cv2.inRange(hsv, lower2, upper2)


    full_mask = cv2.add(lower_mask, upper_mask)


    kernel = np.ones((5,5), np.uint8)
    full_mask = cv2.morphologyEx(full_mask, cv2.MORPH_CLOSE, kernel)
    full_mask = cv2.morphologyEx(full_mask, cv2.MORPH_OPEN, kernel)


    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    ax1.imshow(cv2.cvtColor(original, cv2.COLOR_BGR2RGB))
    ax1.set_title('Original Image')
    ax1.axis('off')

    ax2.imshow(full_mask, cmap='gray')
    ax2.set_title('Segmented Image')
    ax2.axis('off')

    plt.tight_layout()
    plt.show()

    return original, full_mask
```

```
image_path = r"D:\Twitter pics\Twitter\hongmen_banquet.jpg"
lower1 = np.array([0, 100, 20])
upper1 = np.array([10, 255, 255])

lower2 = np.array([160, 100, 20])
upper2 = np.array([179, 255, 255])

original, segmented = segment_image(image_path, lower1, upper1, lower2, upper2)
```

Outputs:

- The original image and the segmented image after applying colour based masks:

Observations:

- The image has been converted to HSV and masks have been generated for the ranges which contain the colour red [0, 10] and [160, 180]. This helps us to segment parts with certain colours from images.

All code uploaded to: Github