Assignment 1

Author 1, Ethan Eisenga
Affiliation: Florida Polytechnic University

A report on assignment 1 which consists of loading, editing, and saving an image and its histogram and a preview for changes to contrast and brightness levels

Report GIRS-2024-23-01

Department of Computer Science
Florida Polytechnic University, Florida, USA

January 2024

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

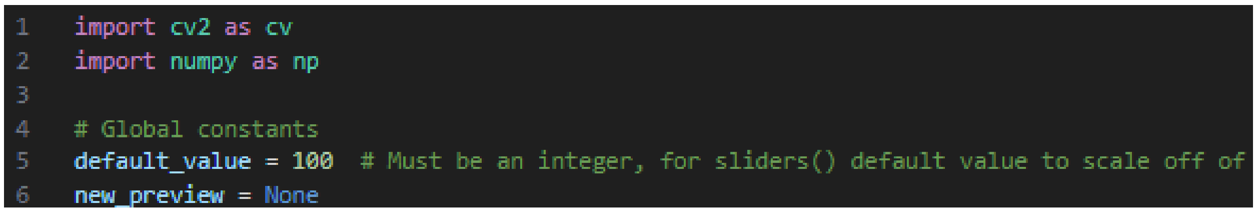
Images of a dog are loaded to the screen with histograms for the user to analyze. Contrast and Brightness sliders are what the user can control to modify the image. The previewed changes can be viewed with strictly the image on the right and its respective histogram. If wished to exit without saving, any key but ’s’ can be pressed. If ’s’ is pressed, the requested changes (seen in the preview) will be applied to the initial image without the use of restoring a backup image.

2 PROGRAM

An in depth analysis of each partition in the program.

2.1 PRE-PROCESSING

OpenCV and Numpy are the only libraries that were imported. default_value is created as a global constant which represents the default setting of the trackbars/sliders and affects their lower and upper bounds. new_preview is created as a global constant so that it can be updated by the following update_images method globally. See Figure 2.1.



```
1  import cv2 as cv
2  import numpy as np
3
4  # Global constants
5  default_value = 100  # Must be an integer, for sliders() default value to scale off of
6  new_preview = None
```

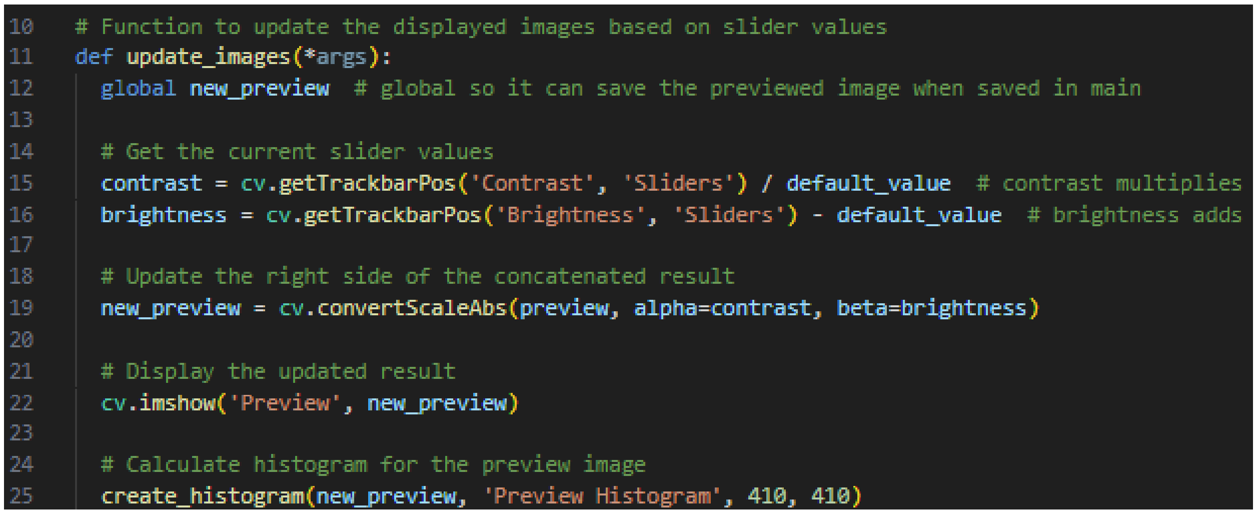
Figure 2.1. Lines 1-6: Pre-Processing which includes importing libraries and declaring global constants

2.2 METHODS

Methods used in the program that were not imported by another library, although inspired by the cited references.

2.2.1 update_images()

This method is called whenever an image needs to be updated on the display, therefore, whenever the trackbar has its slider repositioned. The contrast and brightness are equated differently from their trackbar positions as brightness acts additively, while contrast acts multiplicatively. The image is reloaded with the new contrast and brightness values and displayed to the window. A new histogram must be generated with the new preview, and is therefore called. See Figure 2.2.



```
10  # Function to update the displayed images based on slider values
11  def update_images(*args):
12      global new_preview  # global so it can save the previewed image when saved in main
13
14      # Get the current slider values
15      contrast = cv.getTrackbarPos('Contrast', 'Sliders') / default_value  # contrast multiplies
16      brightness = cv.getTrackbarPos('Brightness', 'Sliders') - default_value  # brightness adds
17
18      # Update the right side of the concatenated result
19      new_preview = cv.convertScaleAbs(preview, alpha=contrast, beta=brightness)
20
21      # Display the updated result
22      cv.imshow('Preview', new_preview)
23
24      # Calculate histogram for the preview image
25      create_histogram(new_preview, 'Preview Histogram', 410, 410)
```

Figure 2.2. Lines 10-25: update_images() method which updates the images whenever the trackbar's sliders are adjusted

2.2.2 create_histogram()

This method is called whenever a histogram needs to be created or updated. It will be called initially before the trackbar slider is repositioned, and again whenever the trackbar does slide. The inputted image is first split into separate R, G, and B planes. The histogram is created separately for each plane before each are normalized. Lines are created for each plane and then shown and created at a specified window location. See Figure 2.3.

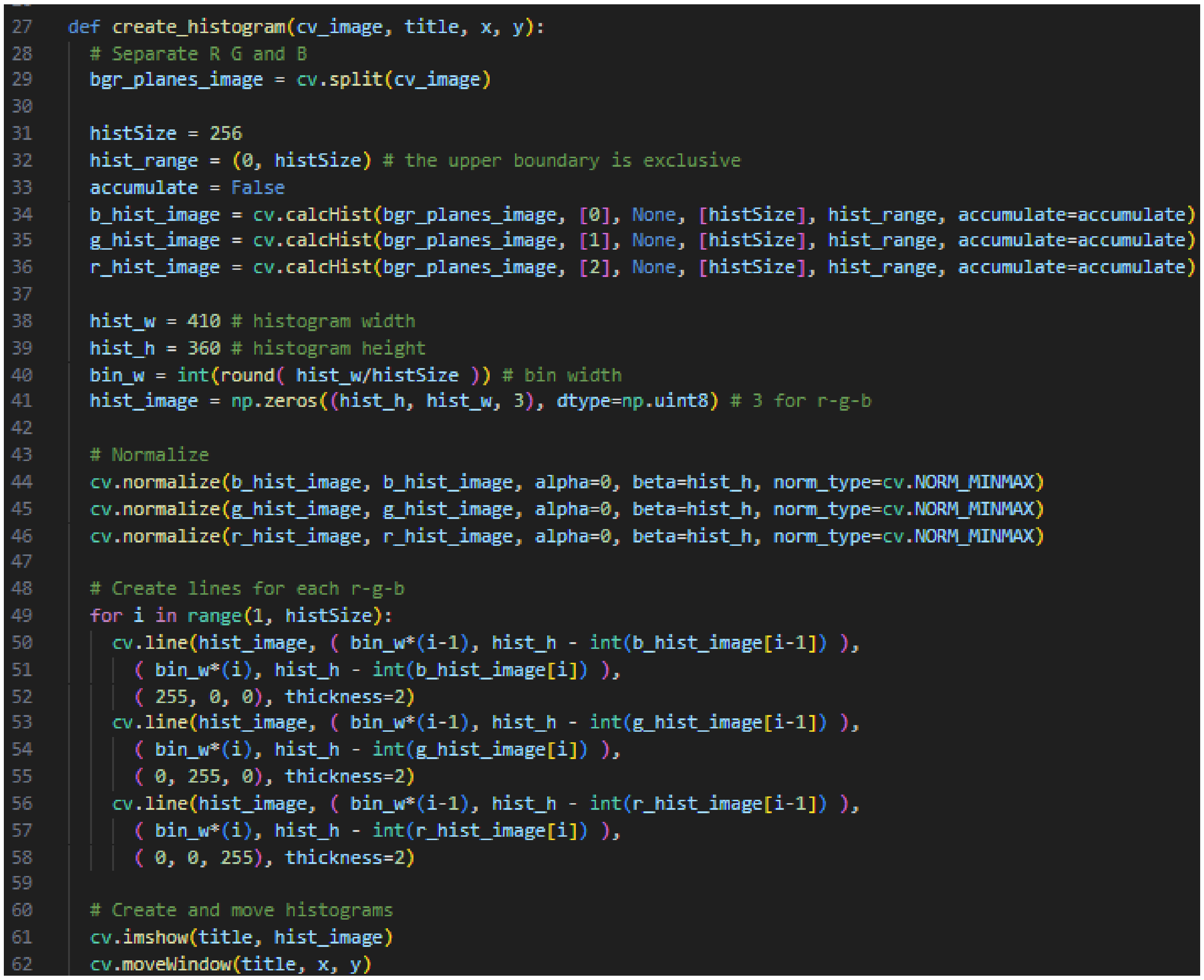
2.3 IMAGE AND PREVIEW IMAGES

The initial image, dog.bmp, is loaded and the preview is a copy of it. Windows are created for each, moved to respective locations, and their images displayed to the user. See Figure 2.4.

Each trackbar, contrast and brightness, first have a window created and moved to the right of both the initial image and the preview. Lastly, the trackbars are created and are told to use update_images() function whenever the trackbar slider is moved.

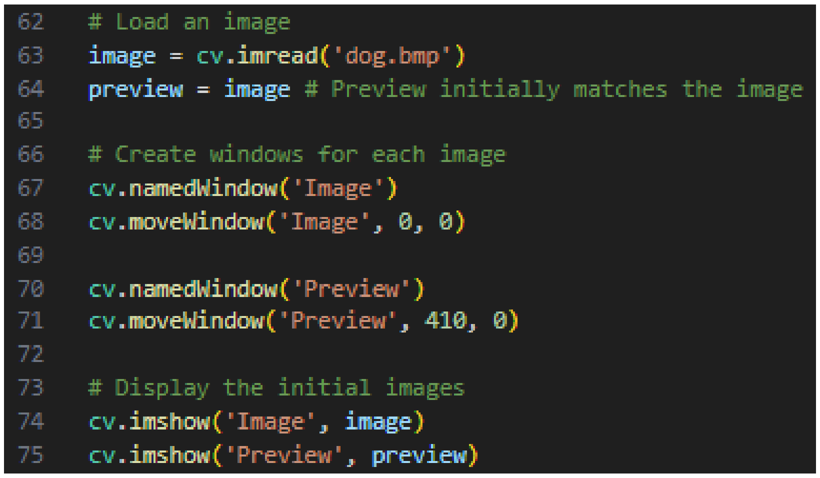
2.4 TRACKBARS

See Figure 2.5.



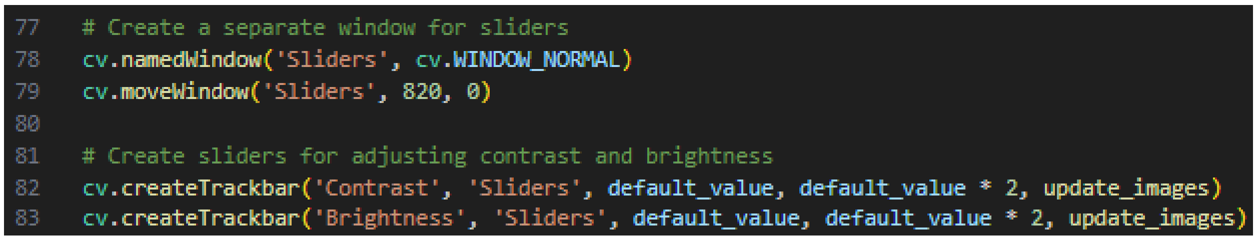
```
27  def create_histogram(cv_image, title, x, y):
28      # Separate R G and B
29      bgr_planes_image = cv.split(cv_image)
30
31      histSize = 256
32      hist_range = (0, histSize) # the upper boundary is exclusive
33      accumulate = False
34      b_hist_image = cv.calcHist(bgr_planes_image, [0], None, [histSize], hist_range, accumulate=accumulate)
35      g_hist_image = cv.calcHist(bgr_planes_image, [1], None, [histSize], hist_range, accumulate=accumulate)
36      r_hist_image = cv.calcHist(bgr_planes_image, [2], None, [histSize], hist_range, accumulate=accumulate)
37
38      hist_w = 410 # histogram width
39      hist_h = 360 # histogram height
40      bin_w = int(round( hist_w/histSize )) # bin width
41      hist_image = np.zeros((hist_h, hist_w, 3), dtype=np.uint8) # 3 for r-g-b
42
43      # Normalize
44      cv.normalize(b_hist_image, b_hist_image, alpha=0, beta=hist_h, norm_type=cv.NORM_MINMAX)
45      cv.normalize(g_hist_image, g_hist_image, alpha=0, beta=hist_h, norm_type=cv.NORM_MINMAX)
46      cv.normalize(r_hist_image, r_hist_image, alpha=0, beta=hist_h, norm_type=cv.NORM_MINMAX)
47
48      # Create lines for each r-g-b
49      for i in range(1, histSize):
50          cv.line(hist_image, ( bin_w*(i-1), hist_h - int(b_hist_image[i-1]) ),
51              ( bin_w*(i), hist_h - int(b_hist_image[i]) ),
52              ( 255, 0, 0), thickness=2)
53          cv.line(hist_image, ( bin_w*(i-1), hist_h - int(g_hist_image[i-1]) ),
54              ( bin_w*(i), hist_h - int(g_hist_image[i]) ),
55              ( 0, 255, 0), thickness=2)
56          cv.line(hist_image, ( bin_w*(i-1), hist_h - int(r_hist_image[i-1]) ),
57              ( bin_w*(i), hist_h - int(r_hist_image[i]) ),
58              ( 0, 0, 255), thickness=2)
59
60      # Create and move histograms
61      cv.imshow(title, hist_image)
62      cv.moveWindow(title, x, y)
```

Figure 2.3. Lines 27-62: create_histogram() method which creates/updates histogram windows for a specified cv image



```
62  # Load an image
63  image = cv.imread('dog.bmp')
64  preview = image # Preview initially matches the image
65
66  # Create windows for each image
67  cv.namedWindow('Image')
68  cv.moveWindow('Image', 0, 0)
69
70  cv.namedWindow('Preview')
71  cv.moveWindow('Preview', 410, 0)
72
73  # Display the initial images
74  cv.imshow('Image', image)
75  cv.imshow('Preview', preview)
```

Figure 2.4. Lines 62-75: Loading of the initial and preview images

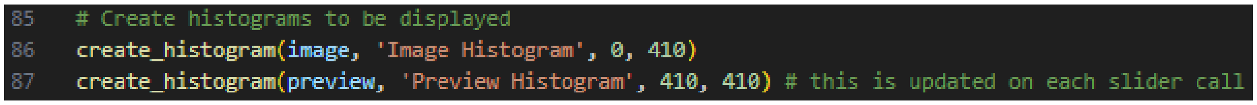


```
77  # Create a separate window for sliders
78  cv.namedWindow('Sliders', cv.WINDOW_NORMAL)
79  cv.moveWindow('Sliders', 820, 0)
80
81  # Create sliders for adjusting contrast and brightness
82  cv.createTrackbar('Contrast', 'Sliders', default_value, default_value * 2, update_images)
83  cv.createTrackbar('Brightness', 'Sliders', default_value, default_value * 2, update_images)
```

Figure 2.5. Lines 77-83: Trackbar creation

2.5 HISTOGRAMS

The histogram for both the image and preview are created initially below their respective images. Recall that most of the grunt work is done within the create_histogram() method and that it is called additionally when the slider updates the images. See Figure 2.6.

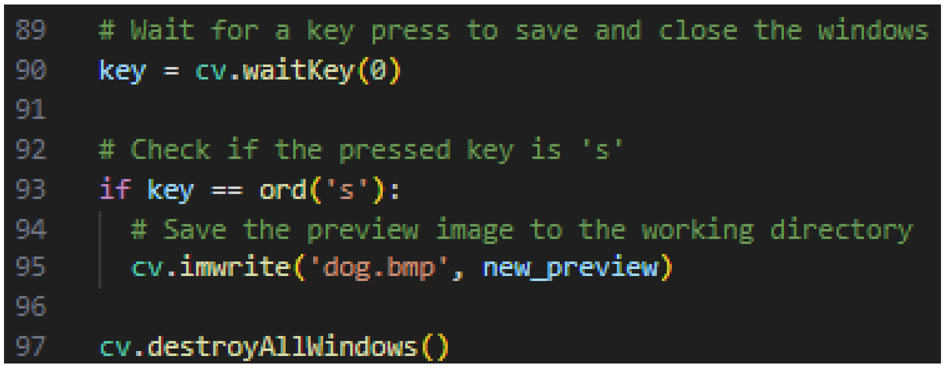


```
85  # Create histograms to be displayed
86  create_histogram(image, 'Image Histogram', 0, 410)
87  create_histogram(preview, 'Preview Histogram', 410, 410) # this is updated on each slider call
```

Figure 2.6. Lines 85-87: Histograms creation

2.6 SAVE AND EXIT

When any key is pressed all windows will be destroyed. However, if the 's' key is pressed then the preview image is written to the home directory with the same file name as the initial file, overwriting it. It is recommended to save a backup of dog.bmp so the loaded image can be restored when needed. See Figure 2.7.



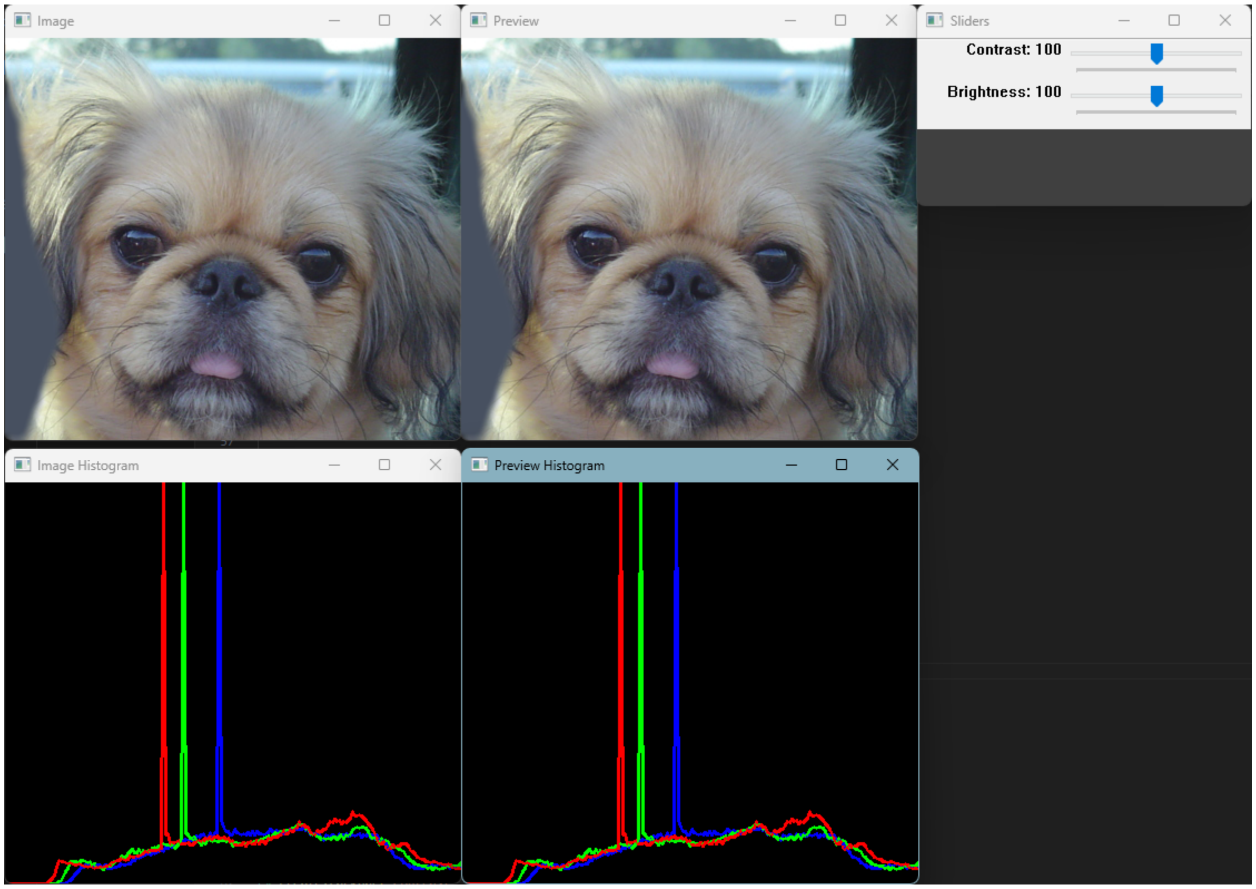
```
89  # Wait for a key press to save and close the windows
90  key = cv.waitKey(0)
91
92  # Check if the pressed key is 's'
93  if key == ord('s'):
94      # Save the preview image to the working directory
95      cv.imwrite('dog.bmp', new_preview)
96
97  cv.destroyAllWindows()
```

Figure 2.7. Lines 89-97: Saving and/or Exiting

3 FUNCTIONALITY

3.1 INITIAL VIEW

The program is run initially, without any changes or sliding either of the trackbars. The initial image is in the top left, with the preview of the modified image (initially identical) on its right. Each image has their respective histograms below, with a line each for their RGB planes. Lastly, the trackbar window is on the far right. See Figure 3.1.



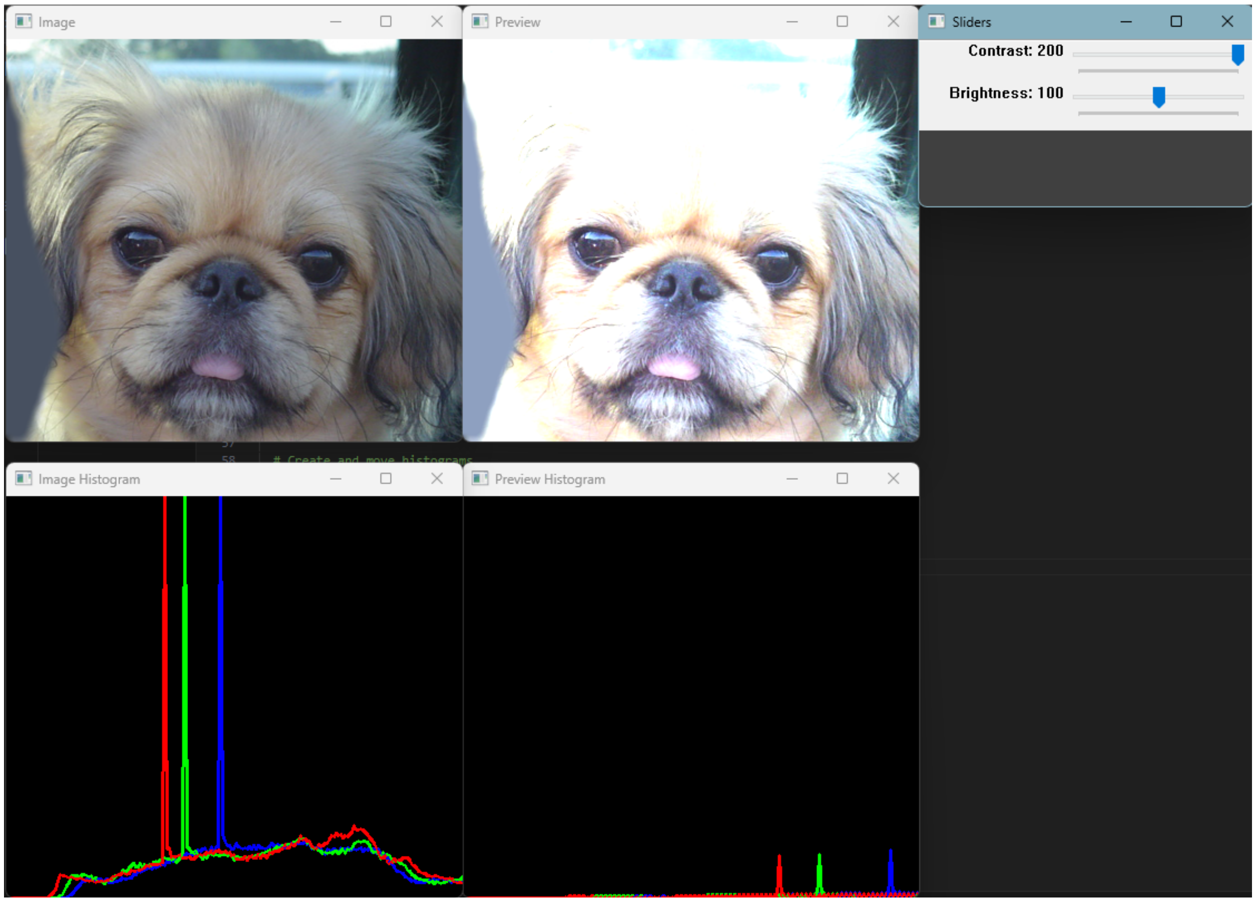
Contrast: 100

Brightness: 100

Figure 3.1. Initial view of the image, preview, their respective histograms, and the trackbar before any changes are implemented

3.2 CHANGING CONTRAST

The contrast slider is adjusted from the 100 initial to 200 (max), and therefore the image's contrast is much greater and appears multiplicatively brighter. See Figure 3.2.



Contrast: 200
Brightness: 100

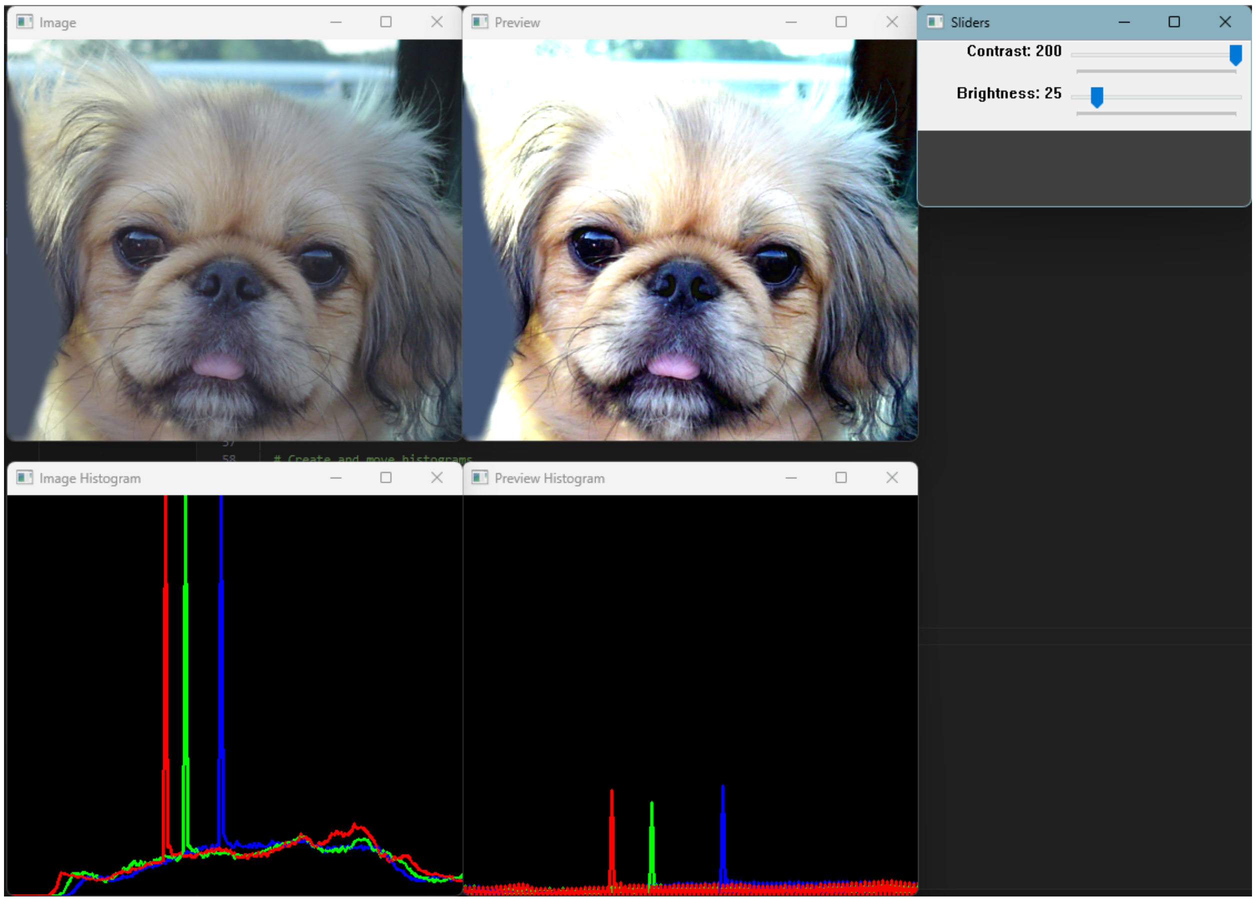
Figure 3.2. Changing the contrast of the previewed image from 100 to 200

3.3 CHANGING BRIGHTNESS

The brightness slider is then adjusted from the 100 initial to 25, and therefore the image's brightness is much darker and appears additively dimmer. See Figure 3.3.

3.4 POST VIEW

The 's' key is pressed to save and exit the program, which also creates dog.bmp file in the home directory with the content of the previewed/modified image, overwriting the old dog.bmp file.

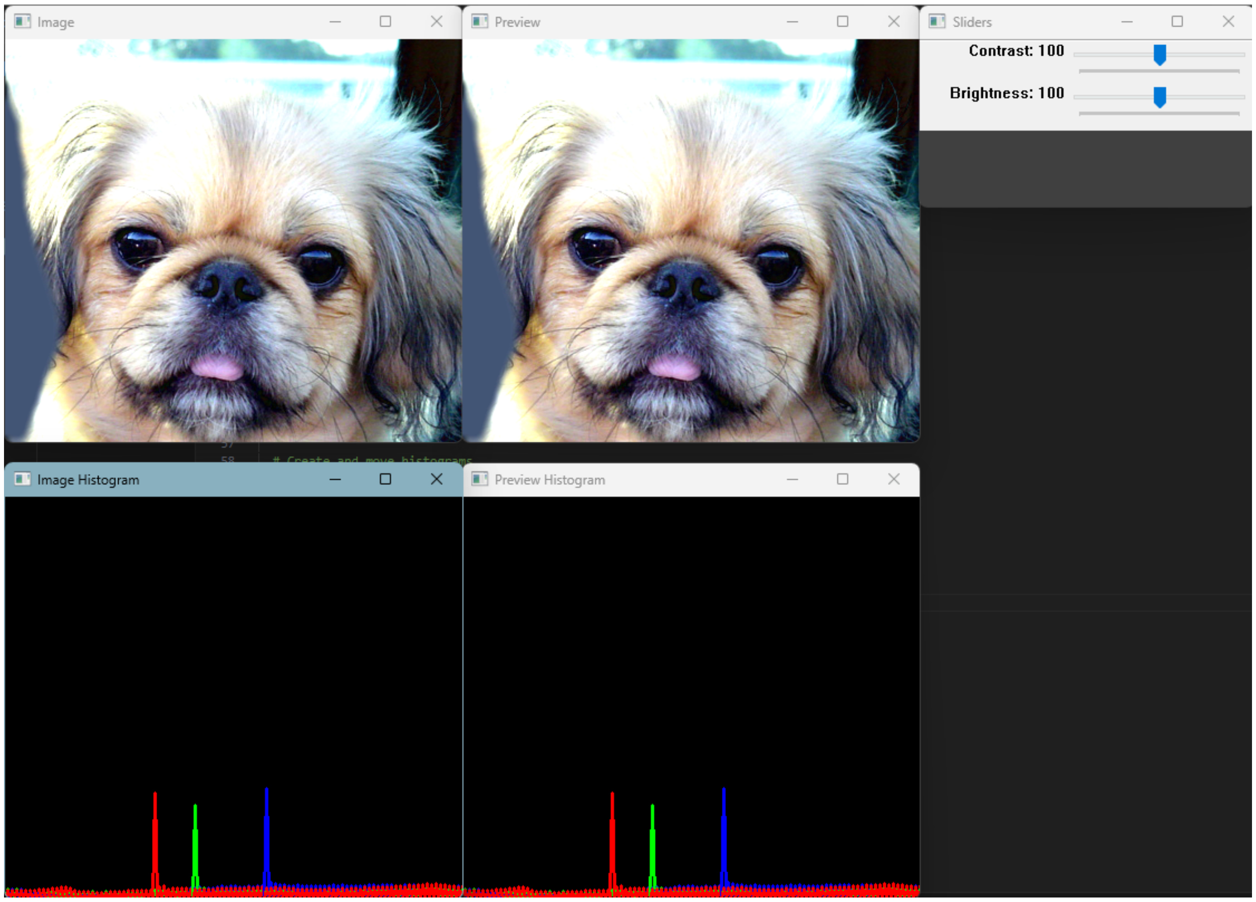


Contrast: 200

Brightness: 25

Figure 3.3. Changing the brightness of the previewed image from 100 to 25

As seen, the images are now identical to the previouw preview and have the adjusted histogram. However, the contrast and brightness sliders are reset to their default value of 100 on their scale of 0 to 200. See Figure 3.4.



Contrast: 100

Brightness: 100

Figure 3.4. Post view of the image, preview, their respective histograms, and the slider being reset after saving the requested changes

4 SUMMARY

The dog.bmp image is loaded with the initial window and the preview window, with histograms of each and a trackbar for contrast and brightness that affect strictly the preview window. When contrast trackbar is slid, the image is multiplicatively modified. When the brightness trackbar is slid, the image is additively modified. When any key is pressed, the program exits. If the key pressed was ’s’, the previewed modified image is loaded to the home directory, overwriting the previous initial dog.bmp image.

REFERENCES

[1] How to change the contrast and brightness of an image using OpenCV in Python?, Shahid Akhtar Khan, 2023, tutorialspoint.com

[2] Python OpenCV – moveWindow() Function, geeksforgeeks, 2023, geeksforgeeks.org

[3] Qt New Functions High-level GUI, OpenCV, 2024, docs.opencv.org

[4] Histogram Comparison, OpenCV, 2024, docs.opencv.org