Assignment 1

Abigail Stucki
Computer Science Undergraduate, Florida Polytechnic University

An assignment on using computer visualization and analysis to alter and display an image in real-time.

Department of Computer Science
Florida Polytechnic University, Florida, USA

January 2024

CONTENTS

1 INTRODUCTION

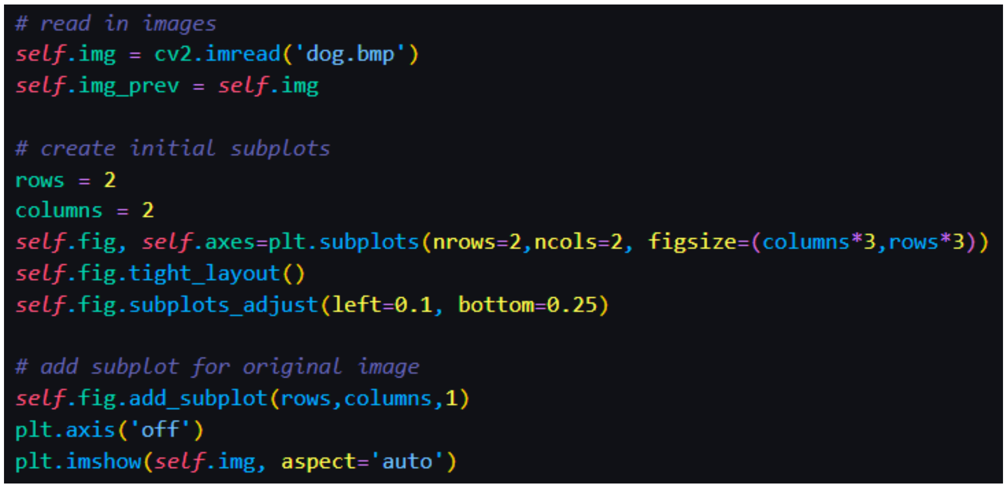
Assignment 1 of the Computer Vision course at Florida Polytechnic University involves numerous key components and libraries in order to properly comprehend the introductory concepts upon which the class is built. The assignment is intended to demonstrate the implementation of knowledge on topics such as working with images, image adjustments, active previewing, and saving functionality.

The essential tasks to be executed in the program included:

1. Load the given image on the screen.
2. Draw the histogram of that image on the screen.
3. Load the same given image as a preview on the screen.
4. Draw the histogram of the same preview image on the screen.
5. Add two scroll bars, one to increase or decrease the brightness and one to increase or decrease the contrast.
6. Add the save functionality, either by button or by pressing any key or combination of keys.
7. When the code is reloaded, it shows the image with new settings in both windows.

1.1 LOADING ORIGINAL IMAGE

Loading the given image onto the screen required the usage of the Python libraries ”cv2” and ”matplotlib.” The first steps involved declaring variables to hold the image, for both the original image and the altered/preview image. A figure was then created to hold subplots and axes to allow for a clean visual presentation. Said figure was declared as a 2x2, allowing for the two images and two histograms to be later displayed. A specific subplot was then initiated to hold the original image without labeled axes.

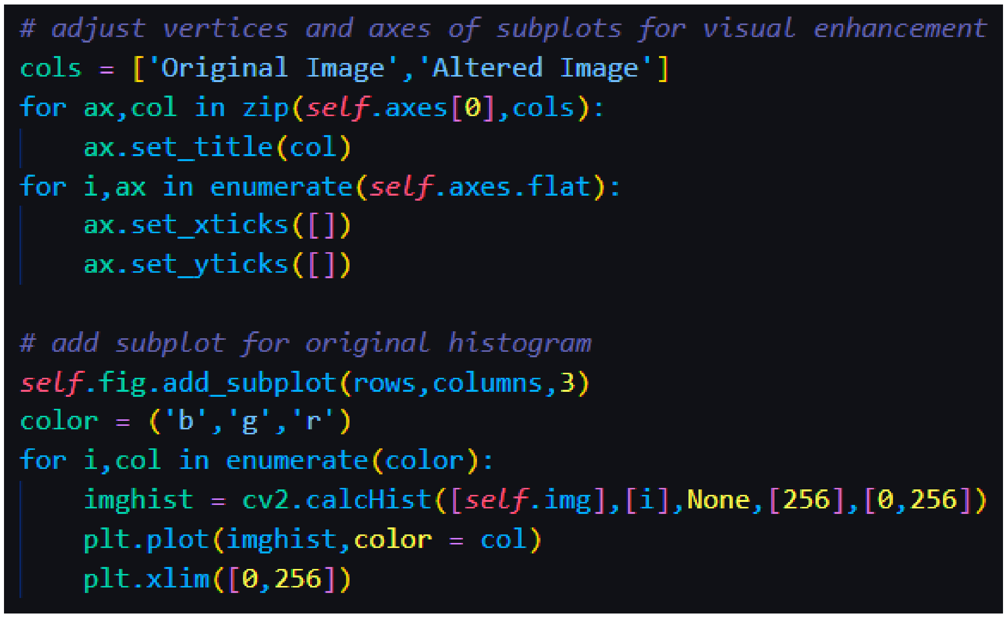


```
# read in images
self.img = cv2.imread('dog.bmp')
self.img_prev = self.img

# create initial subplots
rows = 2
columns = 2
self.fig, self.axes=plt.subplots(nrows=2,ncols=2, figsize=(columns*3,rows*3))
self.fig.tight_layout()
self.fig.subplots_adjust(left=0.1, bottom=0.25)

# add subplot for original image
self.fig.add_subplot(rows,columns,1)
plt.axis('off')
plt.imshow(self.img, aspect='auto')
```

Figure 1.1. This code initializes the image to a variable, creates a subplot to hold the various components, and shows the image on the window.

1.2 DRAWING ORIGINAL HISTOGRAM

Drawing a histogram of the original image onto the screen required the usage of the Python libraries ”cv2” and ”matplotlib.” Axes were created to label the original and altered images as such. RGB histograms were calculated for the original image to show each color, then condensed into one plot.



```
# adjust vertices and axes of subplots for visual enhancement
cols = ['Original Image','Altered Image']
for ax,col in zip(self.axes[0],cols):
    ax.set_title(col)
for i,ax in enumerate(self.axes.flat):
    ax.set_xticks([])
    ax.set_yticks([])

# add subplot for original histogram
self.fig.add_subplot(rows,columns,3)
color = ('b','g','r')
for i,col in enumerate(color):
    imghist = cv2.calcHist([self.img],[i],None,[256],[0,256])
    plt.plot(imghist,color = col)
    plt.xlim([0,256])
```

Figure 1.2. This code demonstrates setting the axes and labels, then creating a histogram containing all necessary data displayed in a visually appealing manner.

1.3 LOADING ALTERED IMAGE

Loading the altered image onto the screen required the usage of the Python libraries ”cv2” and ”matplotlib.” As the variables and figure had already been initialized, the next step was to initiate a specific subplot to hold the original image. In the Python file, these lines of code are placed directly below the code of Figure 1.1 which allows the image to display with no axes or ticks.

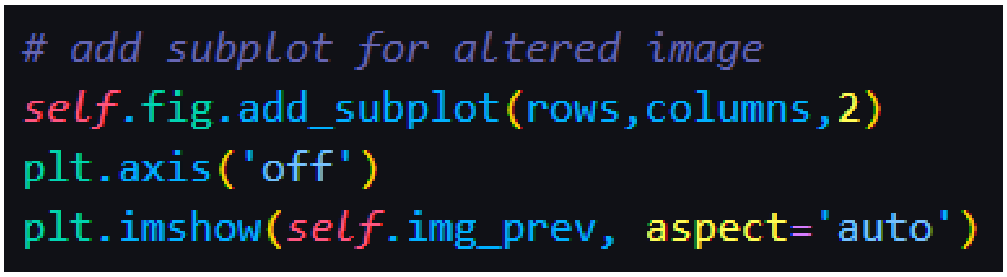

```
# add subplot for altered image
self.fig.add_subplot(rows,columns,2)
plt.axis('off')
plt.imshow(self.img_prev, aspect='auto')
```

Figure 1.3. This code creates a subplot to hold the various components of the altered image and shows the image on the window.

1.4 DRAWING ALTERED HISTOGRAM

Drawing a histogram of the altered image onto the screen required the usage of the Python libraries ”cv2” and ”matplotlib.” As the axes had already been created a few lines prior, this step only required calculating the RGB histograms for the altered image to show each color and then condensing them into one plot.

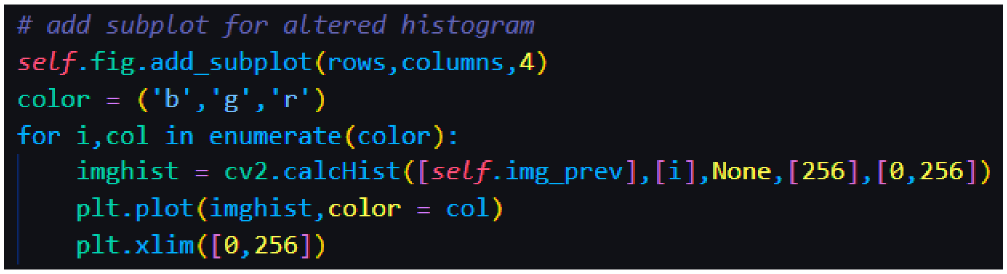
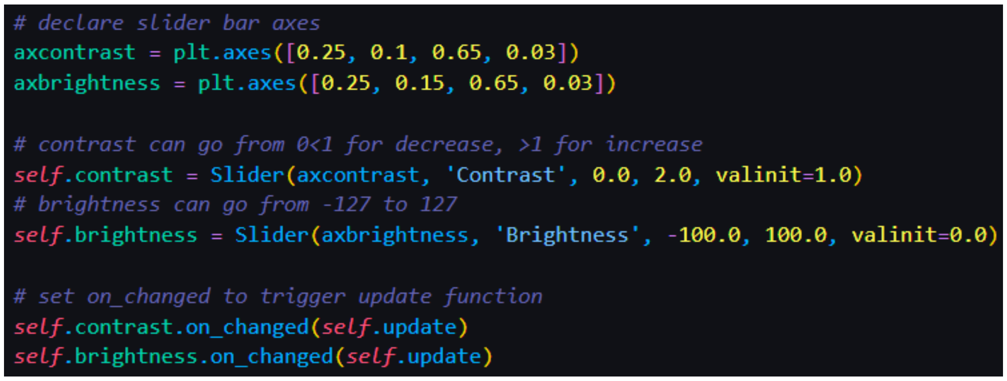

```
# add subplot for altered histogram
self.fig.add_subplot(rows,columns,4)
color = ('b','g','r')
for i,col in enumerate(color):
    imghist = cv2.calcHist([self.img_prev],[i],None,[256],[0,256])
    plt.plot(imghist,color = col)
    plt.xlim([0,256])
```

Figure 1.4. This code demonstrates creating a histogram containing all necessary data displayed in a visually appealing manner.

1.5 ALTERATION OF IMAGE BRIGHTNESS AND CONTRAST

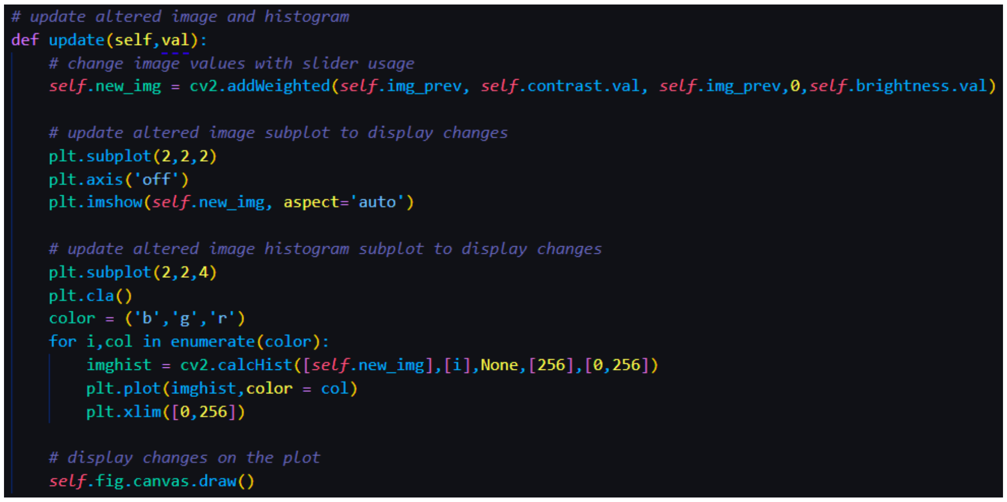
Creating two scrolling bars (Sliders) required the usage of the Python libraries ”cv2” and ”matplotlib.” The axes for the scrolling bars are declared and used to create each Slider. One Slider is used to alter the image contrast from a range of 0.0-2.0. The second Slider is used to alter the image brightness from a range of -100.0 to 100.0. Each variable is then set to call ”self.update” upon any trigger of direct interaction.



```
# declare slider bar axes
axcontrast = plt.axes([0.25, 0.1, 0.65, 0.03])
axbrightness = plt.axes([0.25, 0.15, 0.65, 0.03])

# contrast can go from 0<1 for decrease, >1 for increase
self.contrast = Slider(axcontrast, 'Contrast', 0.0, 2.0, valinit=1.0)
# brightness can go from -127 to 127
self.brightness = Slider(axbrightness, 'Brightness', -100.0, 100.0, valinit=0.0)

# set on_changed to trigger update function
self.contrast.on_changed(self.update)
self.brightness.on_changed(self.update)
```

Figure 1.5. This code demonstrates adding scrolling bars to the window, allowing for interactive alterations to image brightness and contrast.

The ”update” function created a variable to hold the new image with altered brightness and contrast using cv2’s ”addWeighted” function. The subplot to display the altered image is updated. The subplot displaying the histogram of the altered image is also updated to display an accurate histogram of the presented image preview. The figure is then updated with the new changes to subplots.

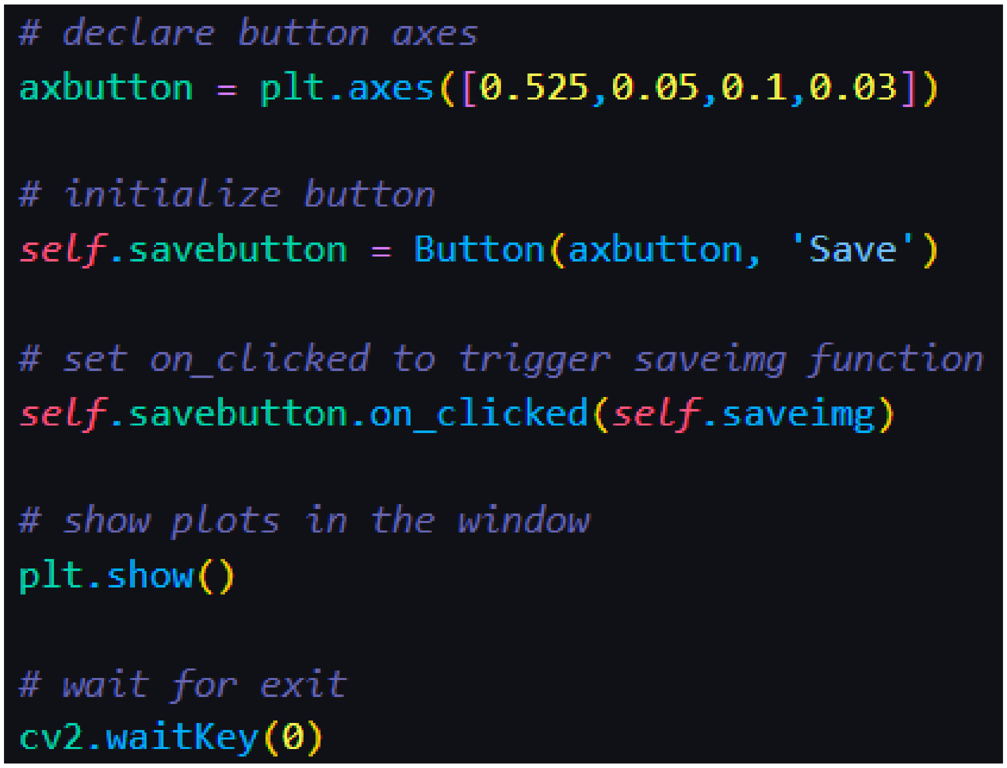


```
# update altered image and histogram
def update(self,val):
    # change image values with slider usage
    self.new_img = cv2.addWeighted(self.img_prev, self.contrast.val, self.img_prev,0,self.brightness.val)

    # update altered image subplot to display changes
    plt.subplot(2,2,2)
    plt.axis('off')
    plt.imshow(self.new_img, aspect='auto')

    # update altered image histogram subplot to display changes
    plt.subplot(2,2,4)
    plt.cla()
    color = ('b','g','r')
    for i,col in enumerate(color):
        imghist = cv2.calcHist([self.new_img],[i],None,[256],[0,256])
        plt.plot(imghist,color = col)
        plt.xlim([0,256])

    # display changes on the plot
    self.fig.canvas.draw()
```

Figure 1.6. This code demonstrates the changes to the altered image and the altered image histogram chosen via Sliders.

1.6 SAVE FUNCTIONALITY

Creating a button to save the image required the usage of the Python libraries ”cv2” and ”matplotlib.” The button axes were declared and used to create a Button in an appropriate location. The variable is then set to call ”self.saveimg” upon the trigger of clicking the designated Button.

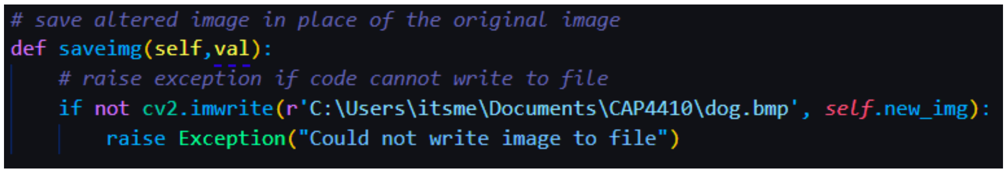


```python
# declare button axes
axbutton = plt.axes([0.525,0.05,0.1,0.03])

# initialize button
self.savebutton = Button(axbutton, 'Save')

# set on_clicked to trigger saveimg function
self.savebutton.on_clicked(self.saveimg)

# show plots in the window
plt.show()

# wait for exit
cv2.waitKey(0)
```

Figure 1.7. This code demonstrates the creation of a Button to allow a user to save the altered image.

1.7 SAVE IMPLEMENTATION FOR RELOADING

Saving the altered image in place of the original image required usage of the Python libraries ”cv2” and ”matplotlib.” This method is called by the code of Figure 1.7 and writes the altered image to the given file path in place of the original image. The code contains an Exception in case of file or file path errors to flag any file writing issues. Upon re-starting the program, the code will use the new image as the original image.



```python
# save altered image in place of the original image
def saveimg(self,val):
    # raise exception if code cannot write to file
    if not cv2.imwrite(r'C:\Users\itsme\Documents\CAP4410\dog.bmp', self.new_img):
        raise Exception("Could not write image to file")
```

Figure 1.8. This code demonstrates the saving functionality of the altered image. The provided file path includes the .bmp file matching the original image, replacing the original image with the newly altered image. This allows for reloaded code to display the new image for further changes.

1.8 VISUALIZATION AND RESULTS

This section contains the displayed output of the code in differing scenarios to demonstrate the features of the program.

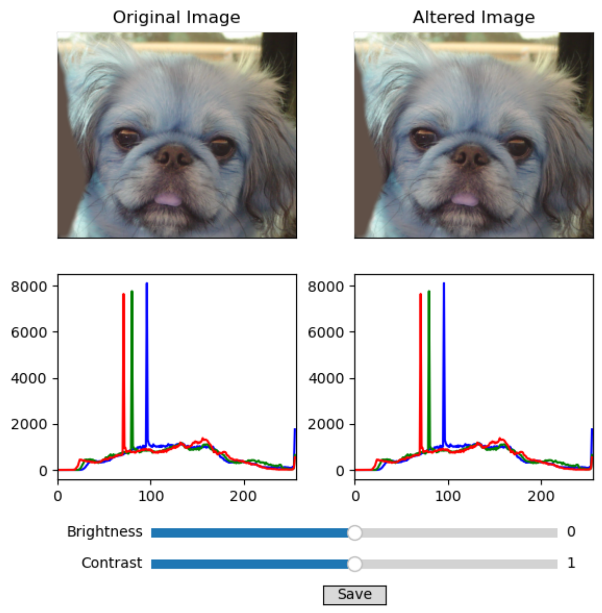
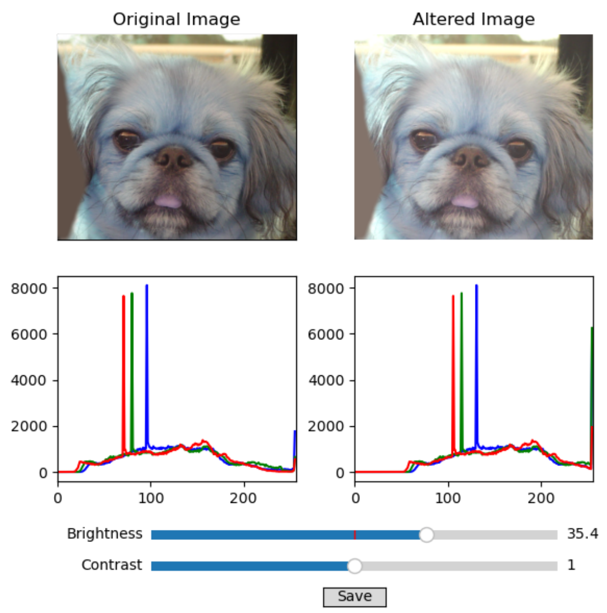
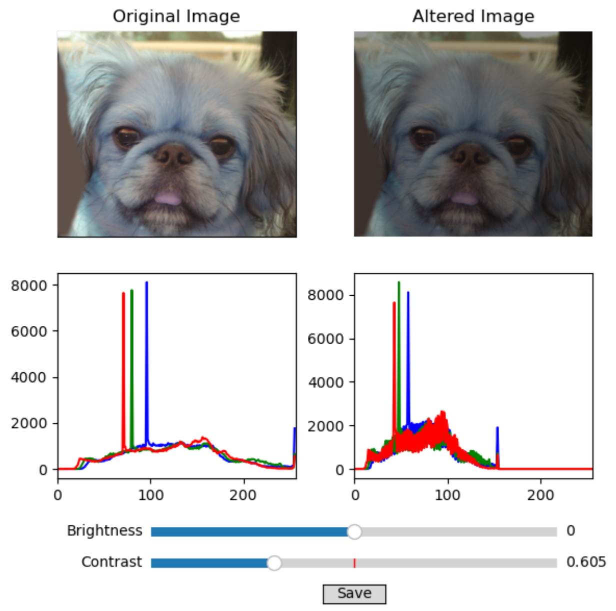


Figure 1.9. This figure displays the output upon program initialization. Visible above are the original image provided in the code and its associated histogram, the altered image and its associated histogram, slider bars for real-time image optimization, and a button with the capability of saving the altered image.



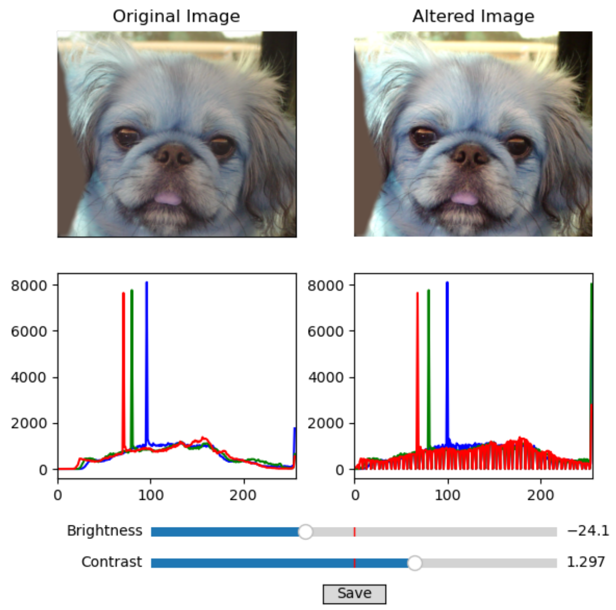
Brightness 35.4
Contrast 1

Figure 1.10. This figure displays the output upon adjusting the slider bar for brightness. The altered image and corresponding histogram display the visual adjustments matching the scale chosen via the slider.



Brightness 0
Contrast 0.605

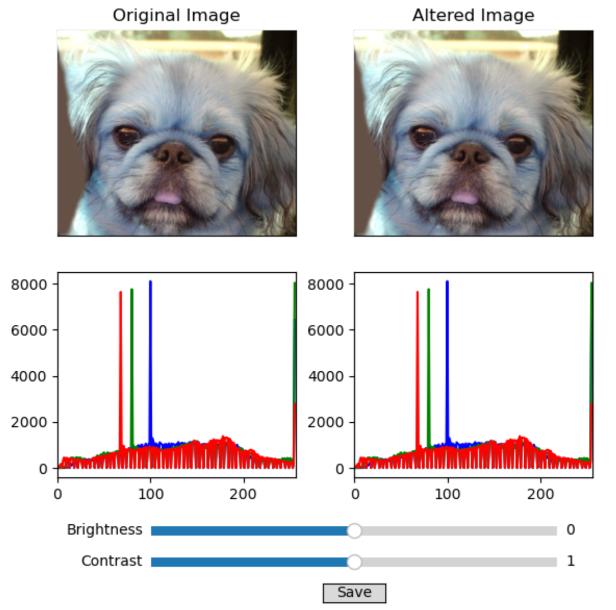
Figure 1.11. This figure displays the output upon adjusting the slider bar for contrast. The altered image and corresponding histogram display the visual adjustments matching the scale chosen via the slider.



Brightness −24.1

Contrast 1.297

Figure 1.12. This figure displays the output upon adjusting the slider bar for both brightness and contrast. The altered image and corresponding histogram display the visual adjustments matching the scales chosen via the slider.



Brightness 0

Contrast 1

Figure 1.13. This figure displays the output after saving changes to brightness and contrast made in the prior program execution and re-opening the program. The original image is replaced via code with the last saved alteration of the image.