Assignment 1 Report

Presented

To the Professor of Florida Polytechnic University,

In Fulfillment of the Requirements
for the Computer Vision Brightness and Contrast Image Adjustment

Computer Engineering

Supervised By: Dr. Abid

Written By: Paul Patullo

Date: January 27, 2025

ABSTRACT

This assignment creates a tool to adjust the brightness and contrast of images using sliders. The user can see the image displayed on the left hand side along with the histogram of that image below while real-time changes occur in the image and its histogram to the secondary image on the right. The tool also includes a ”Save” feature that stores the adjusted image and histogram. The change occurs to the original image not within the current program but when the program is ran a second time will the file remember the data sliders adjusted, storing the information for when the file is reopened.

ACKNOWLEDGMENTS

I would like to thank the developers and contributors of the libraries used in this project, including Tkinter, OpenCV, and Matplotlib, for providing powerful tools that allowed us to utilize their libraries in making this assignment. I would also like to acknowledge the open-source community for their resources and documentation, which helped guide me through the coding process.

CONTENTS

1 INTRODUCTION

1.1 THE IMAGE

This project required creating an image edition GUI, generating its histogram, creating brightness and contrast sliders, and a ”save” button which all stemmed around this image.

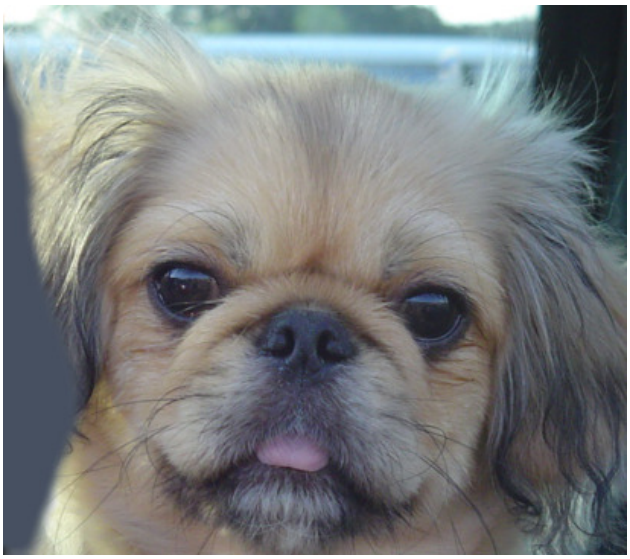

Figure 1.1. A picture of a dog

1.2 FIRST STEPS

When first approaching this assignment, I imported the opencv library into python and tried to use simple functions such as

image = cv2.imread(pathtoimage.jpg, cv2.IMREADGRAYSCALE) imagecopy = image.copy()
hist = cv2.calchist(imagecopy, [0], None, [256], [0, 256])

```
plt.plot(hist)
plt.xlim([0, 256])
plt.show()
cv2.waitKey(0)
cv2.destroyAllWindows()
```

However, I encountered an issue with OpenCV where the image, its copy, and their respective histograms did not display as expected. Instead of opening simultaneously, each window appeared one at a time, requiring the previous one to be closed before the next would open.

After researching various solutions, I discovered an alternative approach involving the use of the Tkinter library. This library enabled me to create a single window with four distinct frames: one for each image (original and copy) and one for each corresponding histogram. Tkinter also facilitated precise alignment of the frames, making the GUI setup much easier to manage.

Tkinters function:

```
imageframe = tk.Frame(root)
imageframe.pack(side = tk.BOTTOM)
```

The issues continued when the provided file was in .bmp format, which is incompatible with Tkinter. To resolve this, I converted the .bmp file into a readable format, adjusted the image size to meet specific requirements, and assigned it to a variable. This allowed me to display both the original image and its copy correctly. Below, you can view the code I created, along with the resulting output.

```
dogImage = r C:Users dogImage.bmp
img = Image.open(dogImage)
resizedimg = img.resize((300, 300)) Resized image

bmpimage = ImageTk.PhotoImage(resizedimg) Converts .bmp file into a readable format
imgcopy = resizedimg.copy() Copys image
imgcopytk = ImageTk.PhotoImage(imgcopy) Reformats copied image

displaylabel = tk.Label(root, image= bmpimage)
displaylabel.place(relx = 0.17, rely = 0.3, anchor = 'w') Displays First Dog Image
displaylabelcopy = tk.Label(root, image= imgcopytk)
displaylabelcopy.place(relx = 0.58, rely = 0.3, anchor = 'w')
```

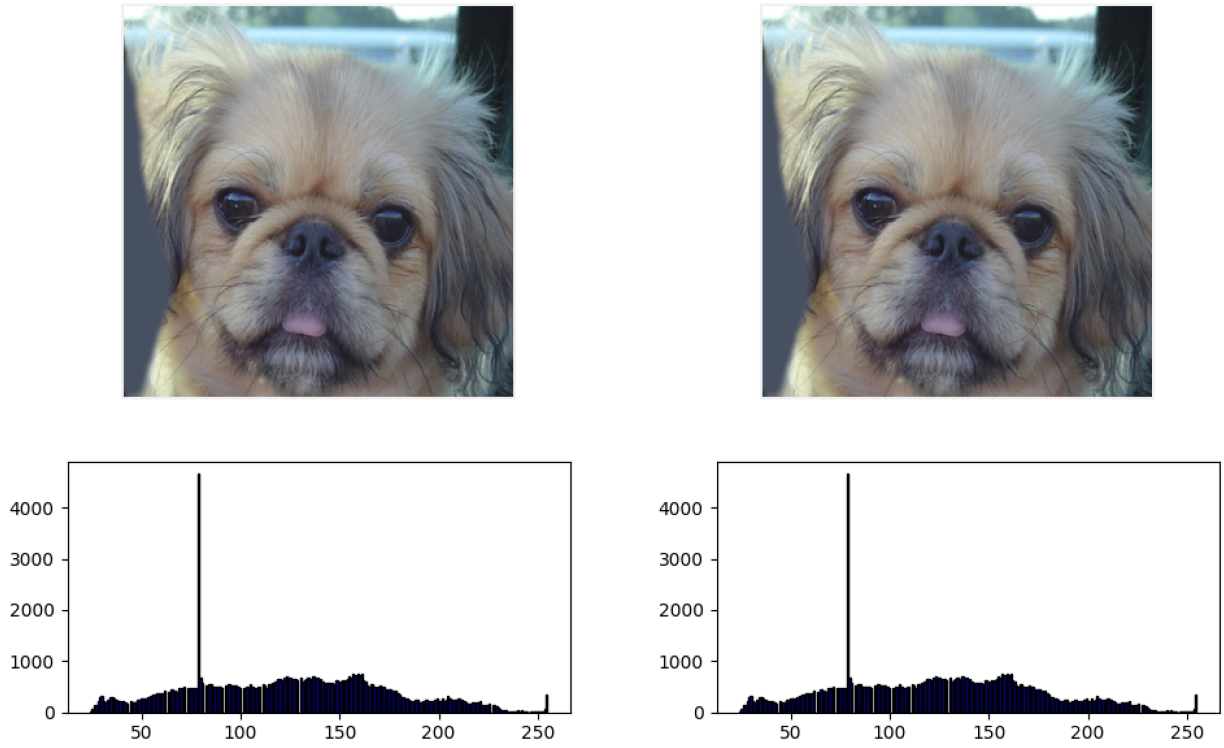

Figure 1.2. Added both images with histograms to the window

1.3 CREATING SLIDER BARS

This was the most challenging step I accomplished in the project. I had to research how to create a slider bar and, beyond that, how to link it to a controller that calculates the equivalent positions for adjusting brightness and contrast. After implementing the slider, I encountered errors because an image cannot be directly modified unless it is stored in an array. The formula I created needed to adjust each pixel within the array, enabling a smooth transition between the slider bar adjustments and the image.

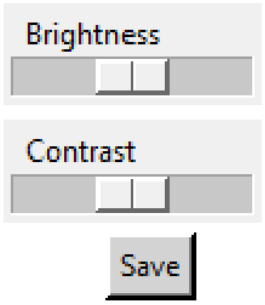

Figure 1.3. Added both images with histograms to the window

In the second stage, the histogram needed to update dynamically with each pixel change to provide an accurate visual representation. To achieve this, I integrated an updating histogram function into the slider function. This function detected changes, cleared the flattened array, and redrew the histogram, ensuring that the displayed histogram accurately reflected the image modifications.

The data plotted in my histogram is accurate, as it corresponds directly to the slider positions. When the brightness slider is moved fully to the left, the image becomes completely black, and the histogram reflects this by showing all the pixel values concentrated in a single bar. The same occurs when the slider is moved fully to the right, turning the image completely white, and the histogram displays a single bar for this frequency as well.

For the contrast slider, moving it to the left turns the image gray, resulting in one bar in the histogram. However, when sliding the bar to the right, the image shows a mix of extreme light and dark areas, causing the histogram frequencies to shift significantly but not into a single bar. This variation in the histogram confirmed that the calculations were performed correctly.

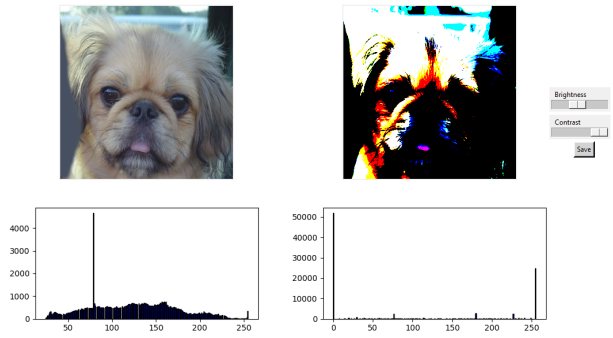

Figure 1.4. Incorporated contrast and brightness sliding bars that updating the histogram

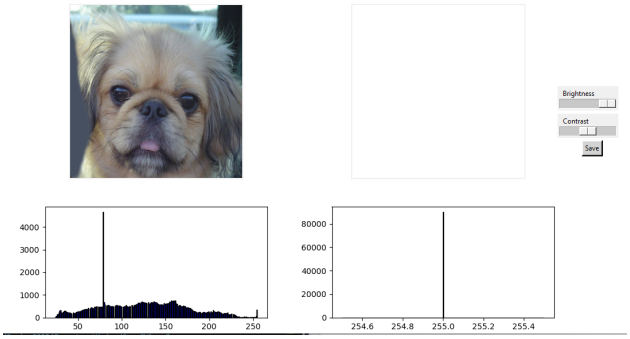

Figure 1.5. Brightness at 1

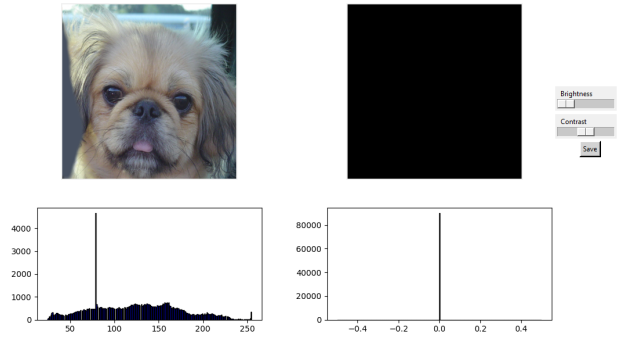

Figure 1.6. Brightness at 0

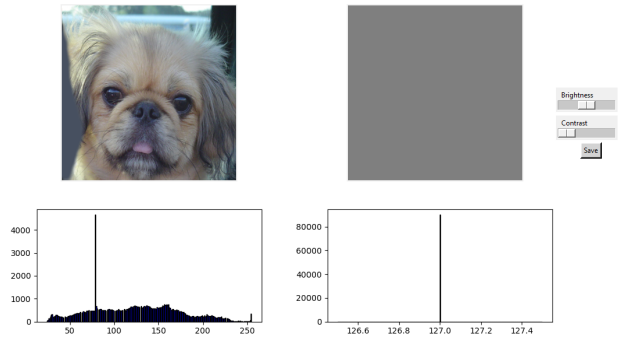

Figure 1.7. Contrast at 0

1.4 IMPLEMENTING THE SAVE BUTTON

The save buttom should have been an easy implementation process however, I could not understand why my program did not update and overright the original file tyle as my code clearly overrights it with the new data from the copied file over.

imgcopy.save(dogImage) Save the modified image back to the original location
canvashist2.save(canvashist1)
print(f”Image saved to dogImage”)

References

https://stackoverflow.com/questions/28595958/creating-trackbars-to-scroll-large-image-in-opencv-python
https://techtutorialsx.com/2020/12/13/python-opencv-add-slider-to-window
https://matplotlib.org/stable/api/widgets$_{a}pi.htmlmatplotlib.widgets.Slider$
$https://www.geeksforgeeks.org/changing-the-contrast-and-brightness-of-an-image-using-python-opencv$
$https://www.geeksforgeeks.org/open-a-new-window-with-a-button-in-python-tkinter$
$https://stackoverflow.com/questions/39482273/changing-image-on-button-click$
$https://www.geeksforgeeks.org/changing-the-contrast-and-brightness-of-an-image-using-python-opencv/$
$https://www.geeksforgeeks.org/python-creating-a-button-in-tkinter/$
$https://www.geeksforgeeks.org/opencv-python-tutorial/$