Samuel Lee

CS5330

Project 1

1/26/2024

Project Overview

This application is split into two main parts: imgDisplay.cpp and vidDisplay.cpp. imgDisplay.cpp simply takes an image as a terminal argument and displays it. It also allows the user to copy a version of the image.

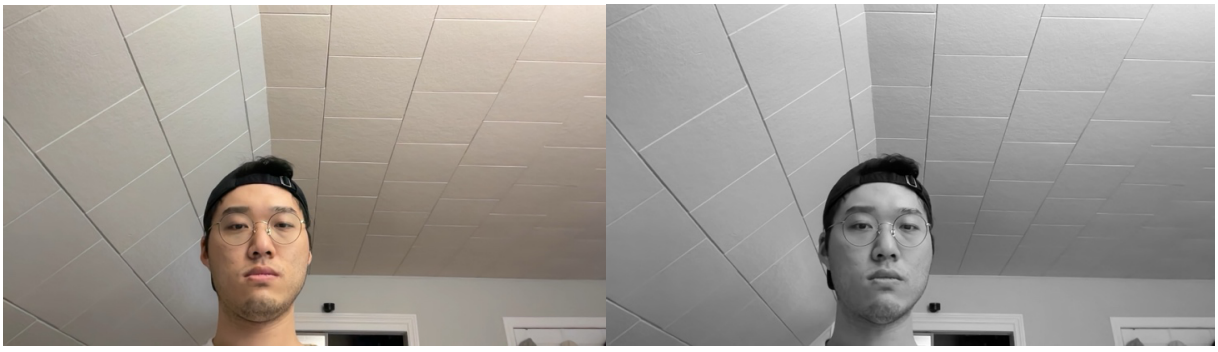
vidDisplay.cpp is more complex. The application opens an instance of the video camera attached to the device, from which the user can apply many filters and features. Such filters include sepia, blurred, greyscale, custom greyscale, greenify, quantize, and median. Other features include saving a screenshot, face tracking, face covering, and eye activity tracking.

Overall, this application allows users to apply different filters and features to live video.

Project Description and Images

Task 3: Original Image vs Greyscale Image

Original (left) Greyscale (right)



Task 4: Customized Greyscale Image

- For my custom greyscale image, I used different weights for how RGB is calculated than the standard greyscale filter.

- This is the standard that cvtColor follows

$$\text{RGB[A] to Gray:} \quad Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

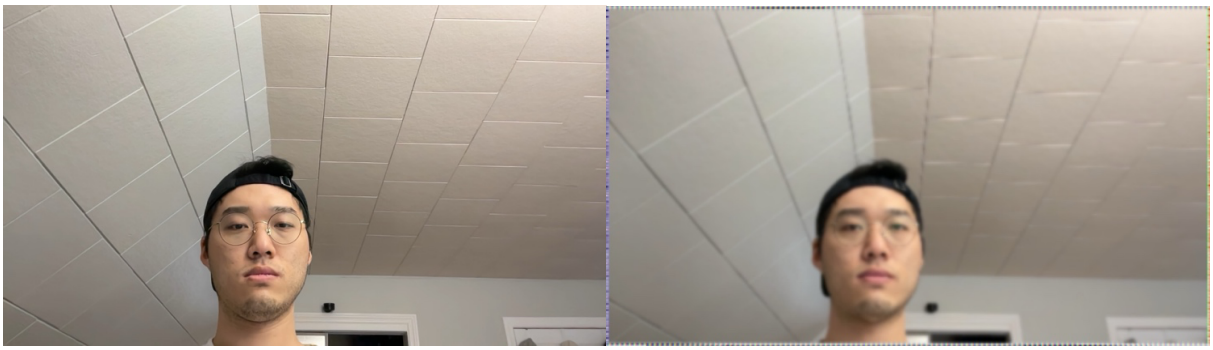
- I used R: 0.2, G: 0.6, B: 0.2
- My greyscale image has more ratio of blue in it and less red.

Standard Greyscale (left) Custom Greyscale (right)

Task 5: Sepia Image

- For my sepia image filter, I created a nested loop by rows and columns. I made sure to make the filter efficient by using pointers. Using the ratios the assignment provided, I calculated the RGB values by multiplying the ratios by the actual channel values. This gave me the sepia tone filter RGB channel values, which I assigned to the destination pointer pixel. I had to make sure my values were under 255 since the output was an unsigned char 8-bit image.

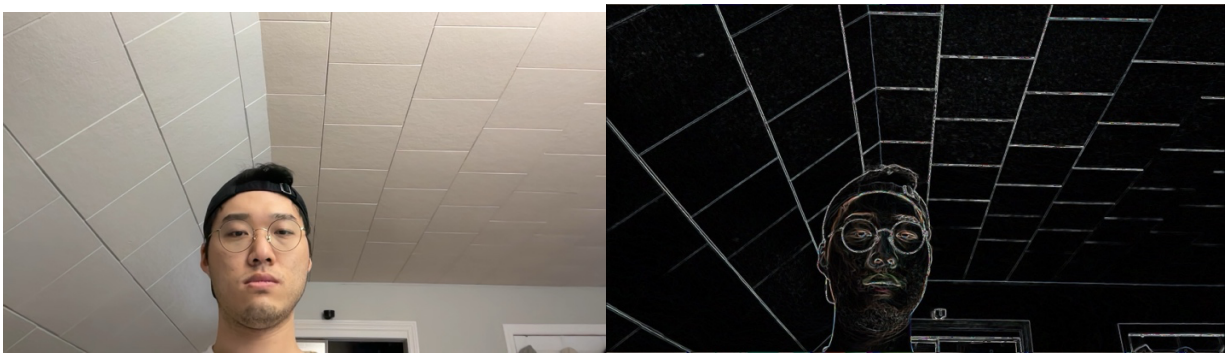
Task 6: Original Image(left) vs Blurred Image (right)



Task 6: Timing Screenshot

```
Time per image (1): 0.0796 seconds
Time per image (2): 0.0025 seconds
```

Task 8: Original Image(left) vs Gradient Magnitude Image(right)

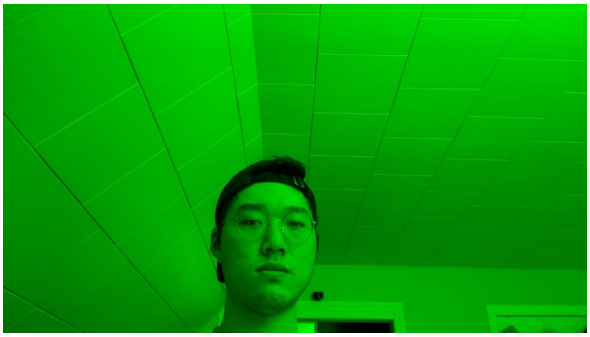


Task 9: Original Image(left) vs Blurred/Quantized Image(right)

Task 10: Image with Face Detection

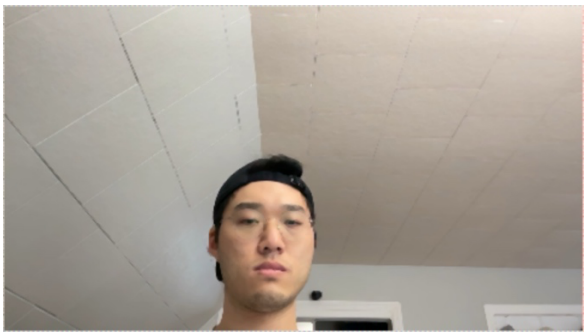
Task 11: Original vs Greenify Filter

- This filter makes all red and blue channels equal to zero but leaves the green channel information.



Task 11: Original Median Filter

- This filter implements a 3x3 median filter.



Task 11: Face Cover Detection

- This feature allows for face covering by first detecting the face then putting a white block on the face.

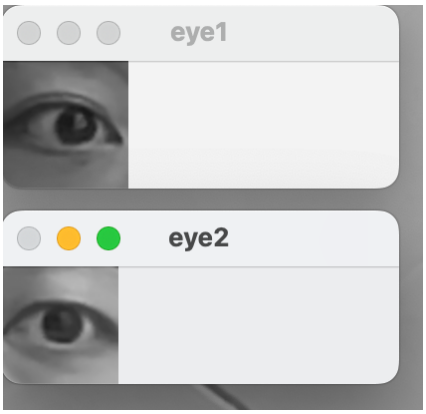
Extension Description and Images

I chose to further delve into object detection. With limited time and resources, I could not implement a machine learning model for object detection, but I was able to learn how to implement object detection through Haar cascade. By initially studying the example code Professor posted, I was intrigued to dive deeper into this subject.

Based on my research, I learned that Haar cascade is a machine learning object detection method used to identify objects such as faces, hands, and more. It's a pretty efficient method to filter out regions of an image that most likely does not have the object of interest. The "cascade" structure makes this method efficient because if the region of interest does not fit the criteria of a classifier within a cascade (which are a series of classifiers in a specific order), the region is disregarded. If the region fits the criteria, it will move on to the next classifier and so on.

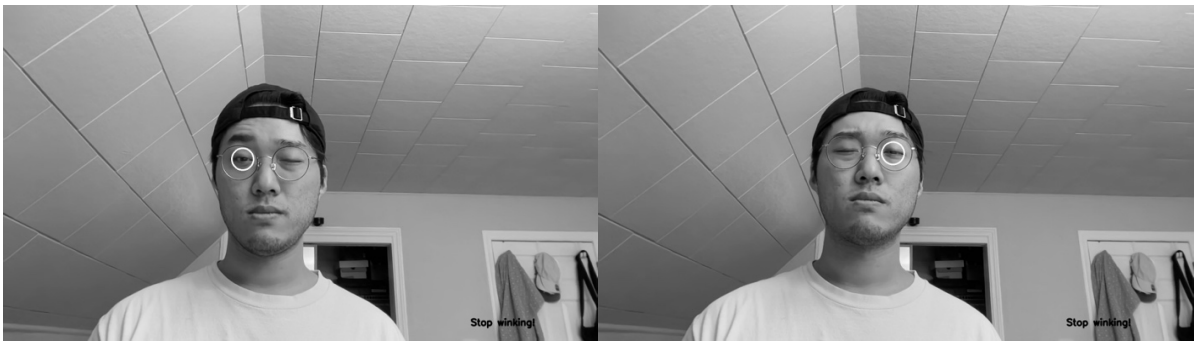
Using this knowledge, I implemented an eye activity detector. The feature I implemented happens when the user types 't.' It efficiently detects if the subject's eyes are closed, looking left, looking right, or winking. Utilizing the Haar cascade method along with OpenCV libraries and C++, I was able to implement this interesting feature. The current activity/status of the eyes are show. I added this feature in the given module 'faceDetect.cpp.' The new functions I created are called determineGazeDirection, compareRectBySize, and detectEyes. Here are some screenshots:

Eyes



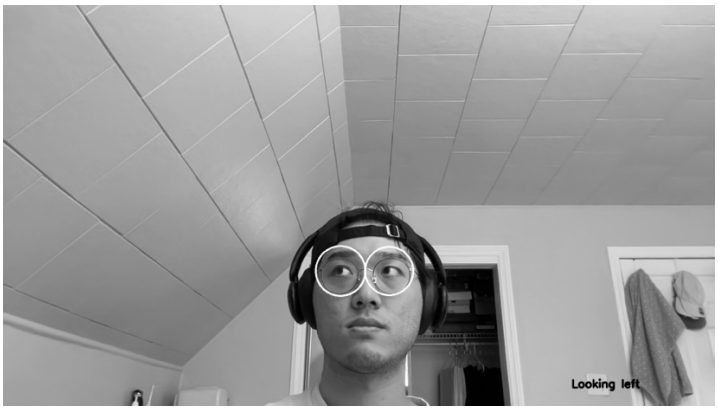
No Face Detected

Winking (both eyes)



Looking Right

Looking Left



Eyes Closed

Reflection

This project helped me tie the conceptual with practical applications. The theories and concepts I learned in class initially did not make sense to me because I couldn't see how these could be applied to programming. After reading online resources and the OpenCV documentation, I was able to connect theory with practice. Most importantly, this project helped me understand Sobel and Gaussian better.

Acknowledgments

- OpenCV documentation
- Office hours
- Class readings
- Sample code from the project and from the Professor