Haobo Zhang

09/18/2024

CS 5330

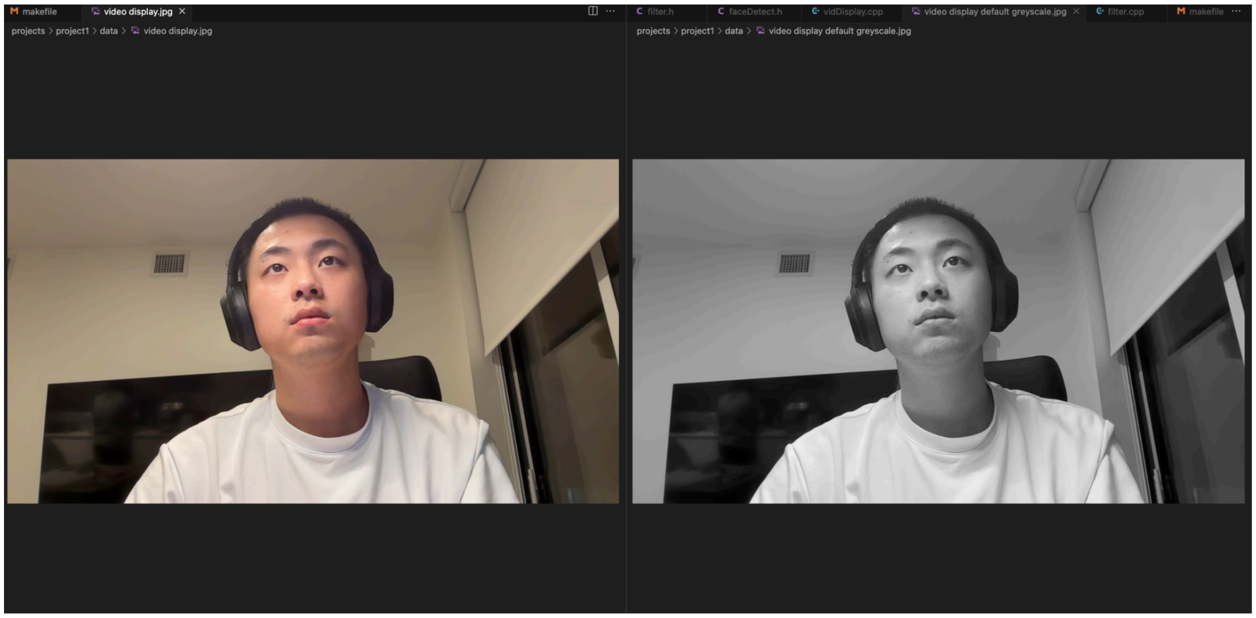
Project 1 Report

Summary

This project focuses on using C++ and OpenCV for real-time image and video processing. The tasks include reading and displaying images, capturing live video, and applying various filters to the video stream. It implements pixel-based filters such as greyscale, using both the OpenCV cvtColor function and a custom method by setting all color channels to 255 - Red. Additionally, it involves creating blur filters, Sobel filters for edge detection, calculating gradient magnitudes, and developing a function to blur and quantize color images. The project also incorporates face detection using a pre-trained Haar cascade classifier. Three additional filters were added: color inversion, applying greyscale to all areas except the face, and blurring and quantizing the face area in the live stream. These tasks explore OpenCV's Mat structure and direct pixel manipulation, focusing on performance optimization and creative image processing techniques.

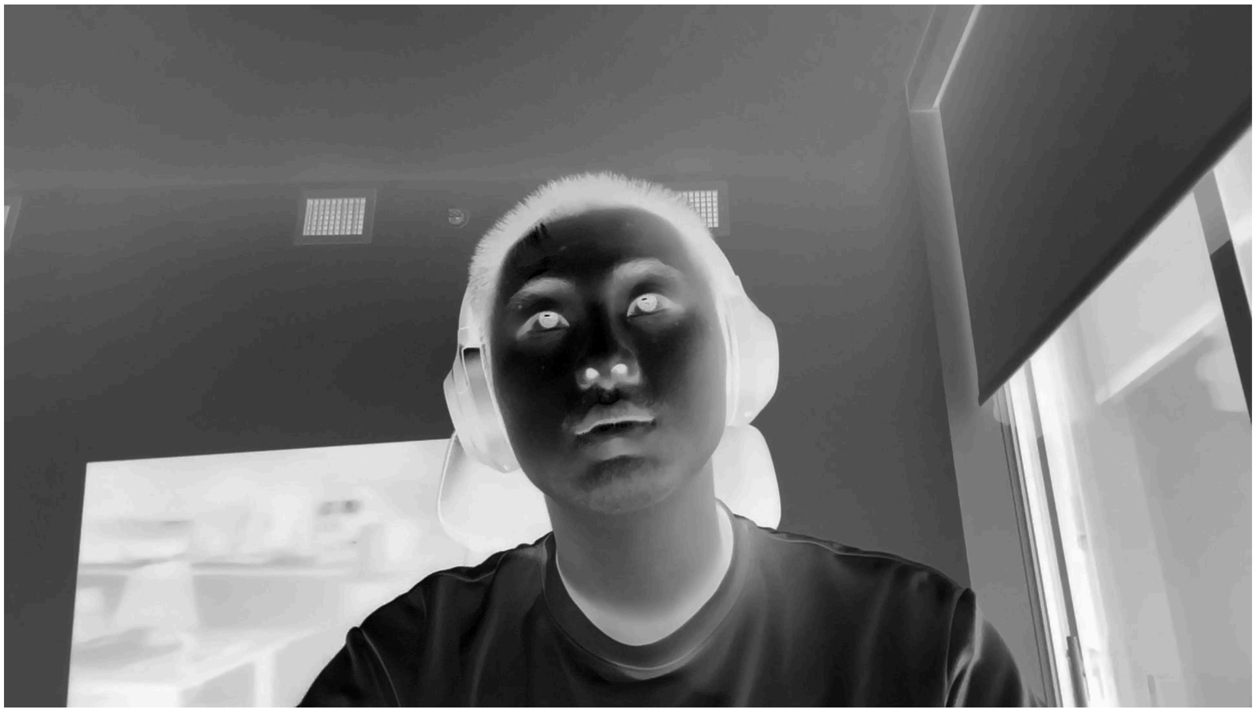
Required Images:

Required Image 1: show the original and cvtColor version of the greyscale image


makefile
video display.jpg
projects > project1 > data > video display.jpg
filter.h
faceDetect.h
vidDisplay.cpp
video display default greyscale.jpg
filter.cpp
makefile
projects > project1 > data > video display default greyscale.jpg

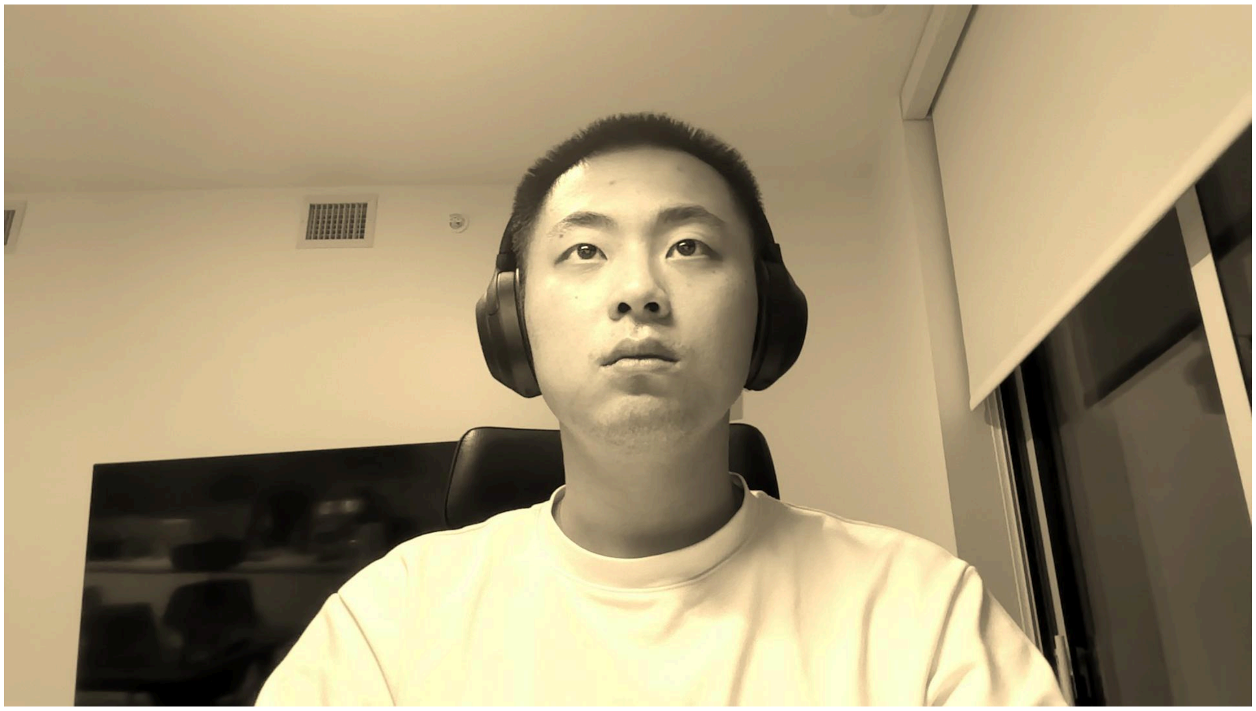
Required Image 2: show your customized greyscale image.

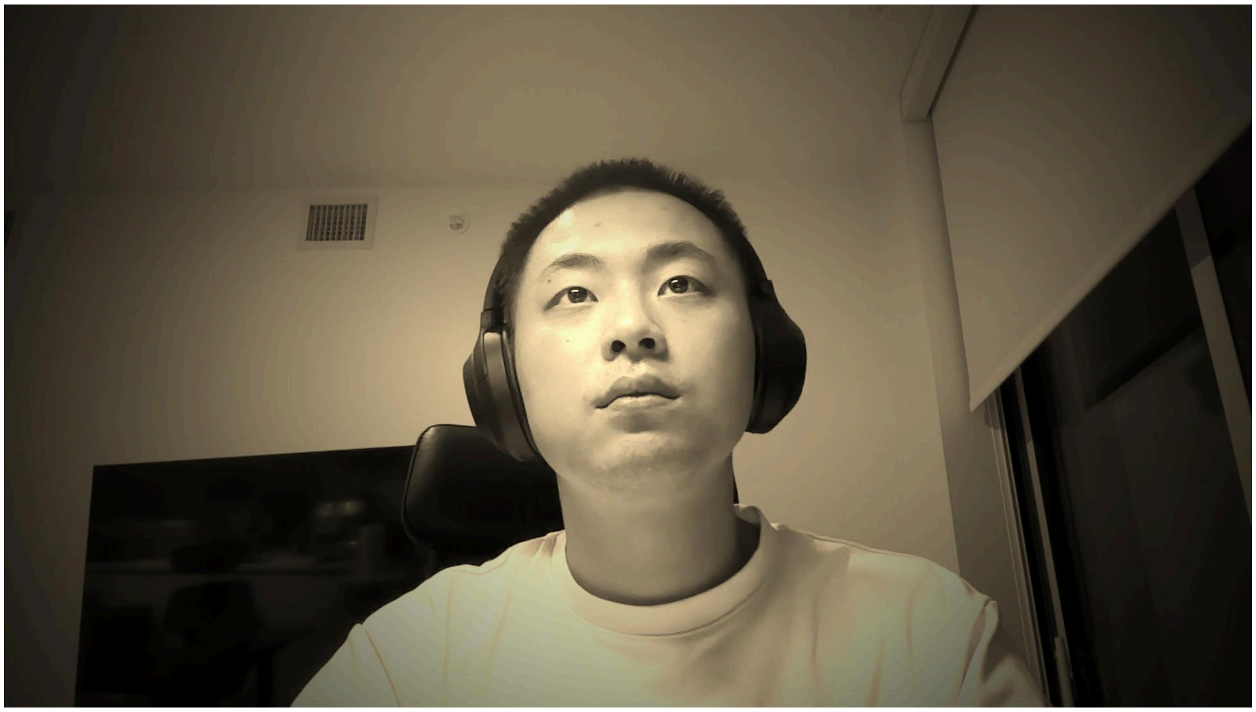
I generated an alternative greyscale image by inverting the red channel for each pixel (255-Red) and assigning this inverted value to all three color channels. This approach creates an inverted effect where areas with high red intensity in the original image become darker, and areas with low red intensity become lighter. The result looks like a greyscale image with reversed black and white.



Required Image 3: show your sepia tone filter with vignetting

Before replacing the original frame, I create a ‘dst’ frame to store manipulated data from the filter function. This way can ensure it uses the original RGB value for computing the new RGB value.

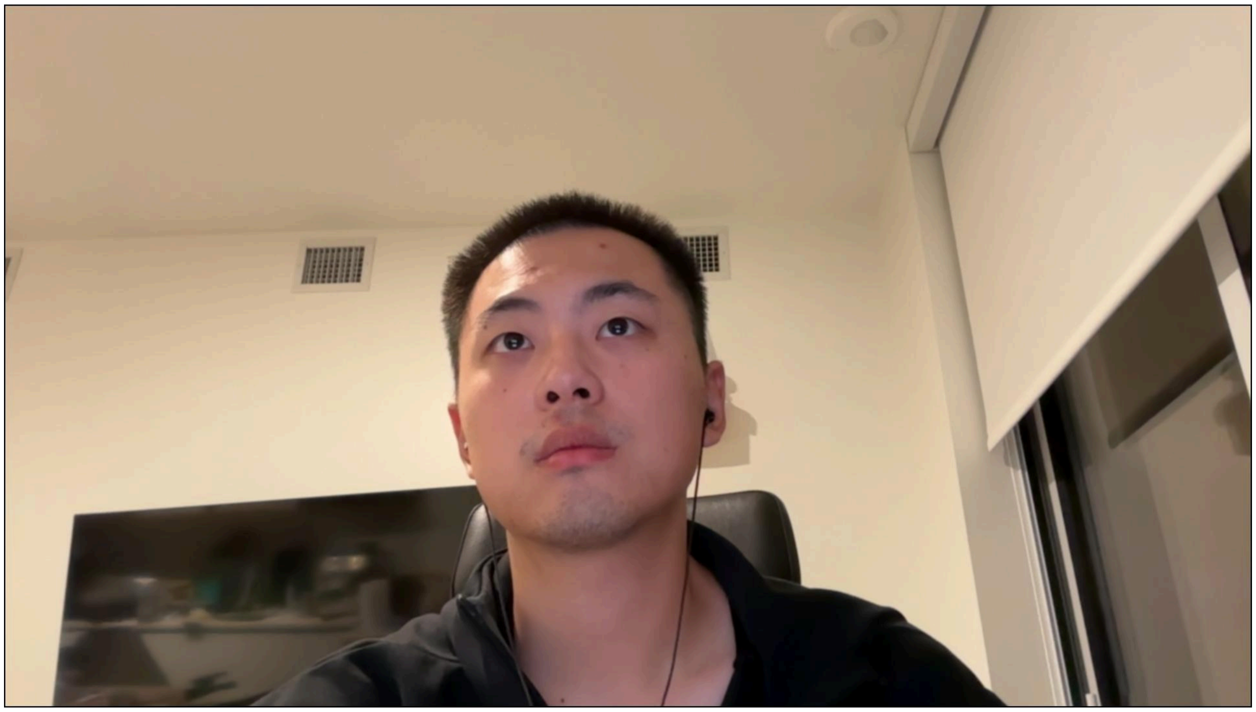




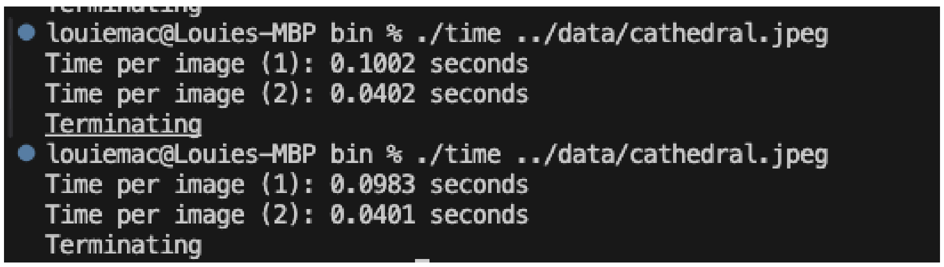
Required Image 4:

Original image:

Blurred image:



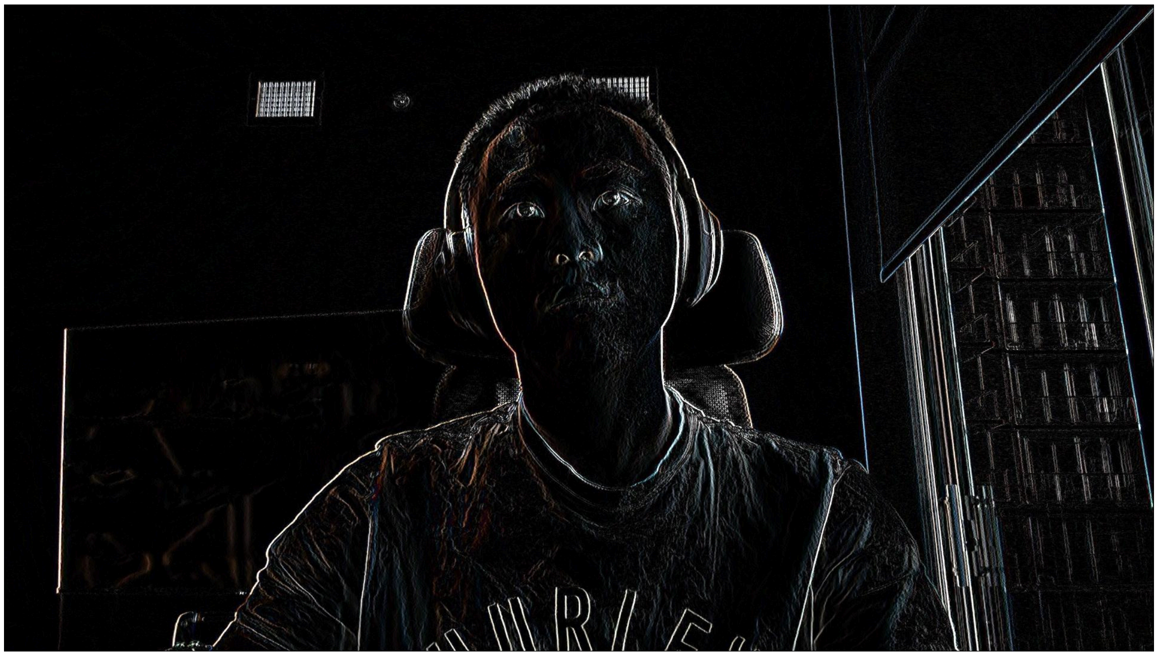
Timing report:



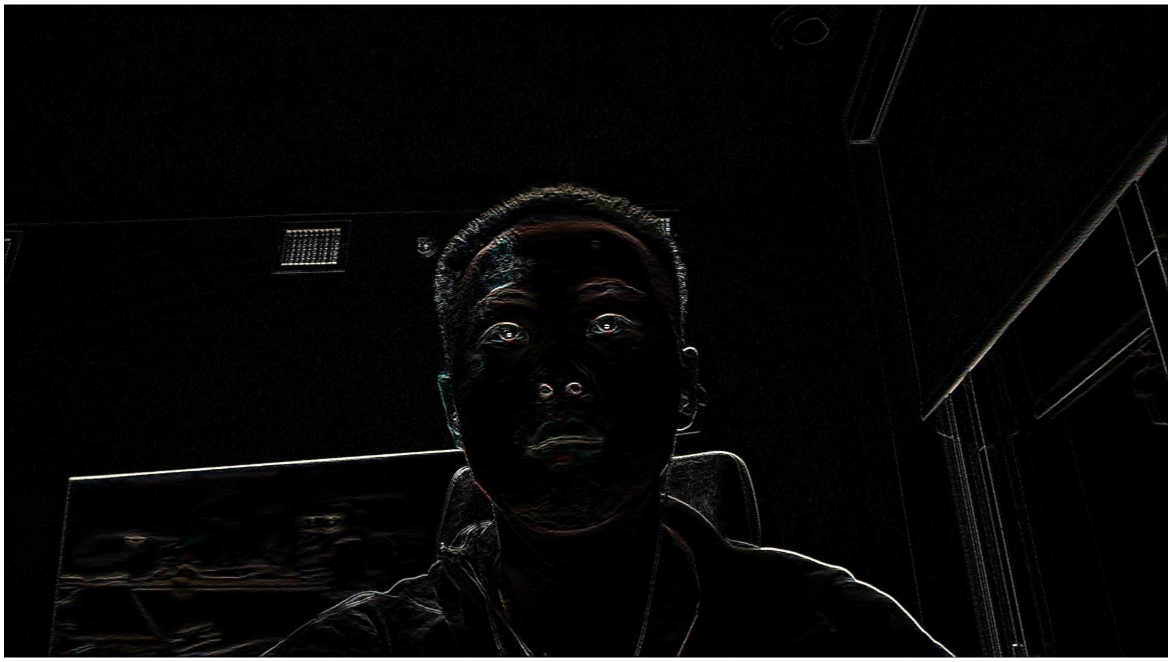
```
louiemac@Louies-MBP bin % ./time ../data/cathedral.jpeg
Time per image (1): 0.1002 seconds
Time per image (2): 0.0402 seconds
Terminating
louiemac@Louies-MBP bin % ./time ../data/cathedral.jpeg
Time per image (1): 0.0983 seconds
Time per image (2): 0.0401 seconds
Terminating
```

Required Image 5:

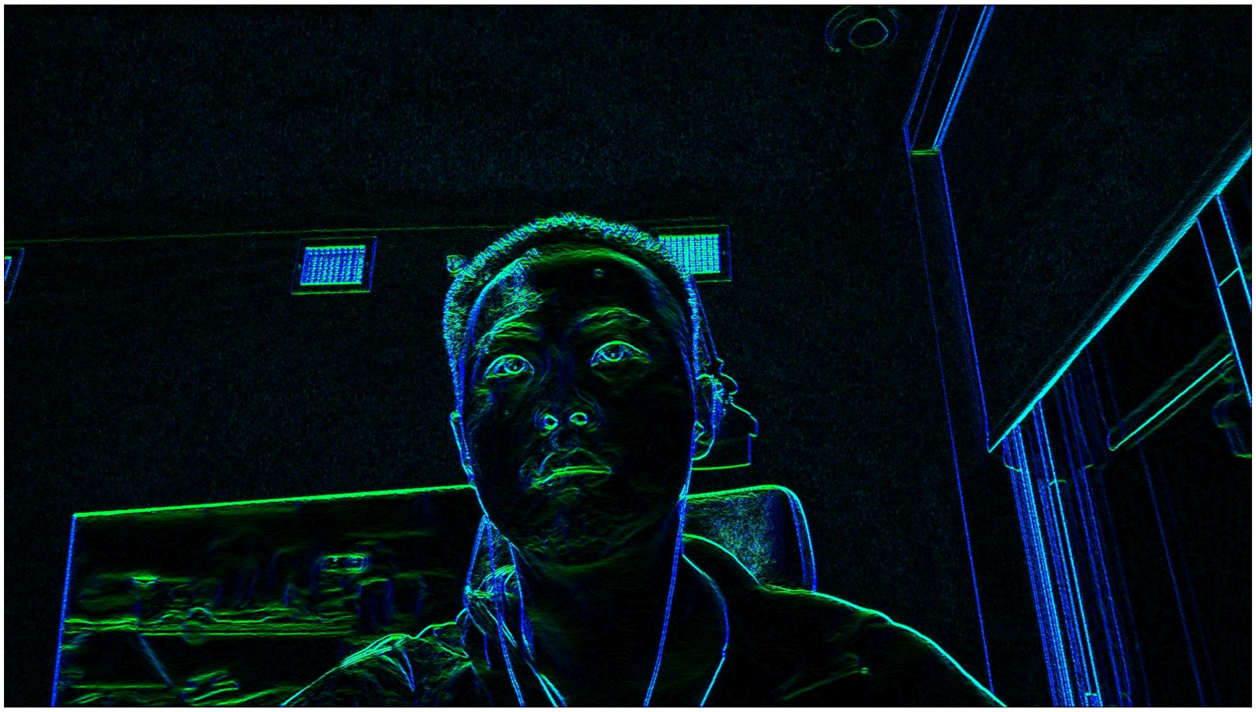
Soble X:



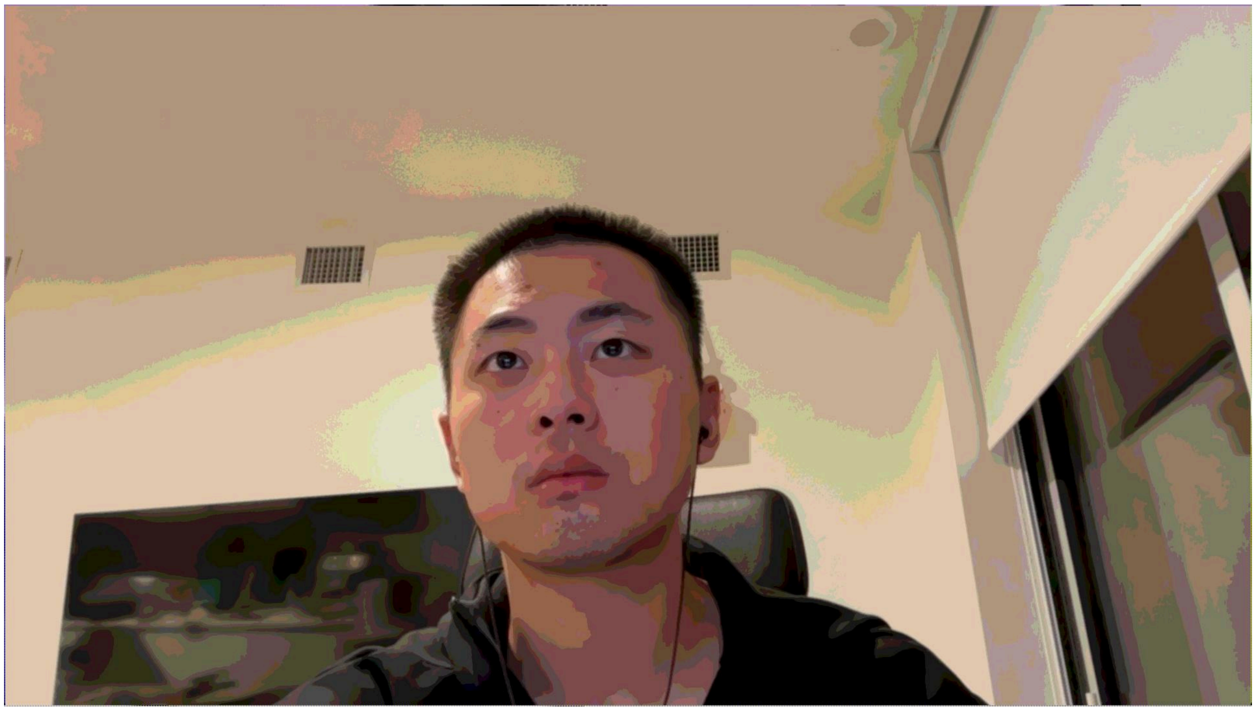
Soble Y:



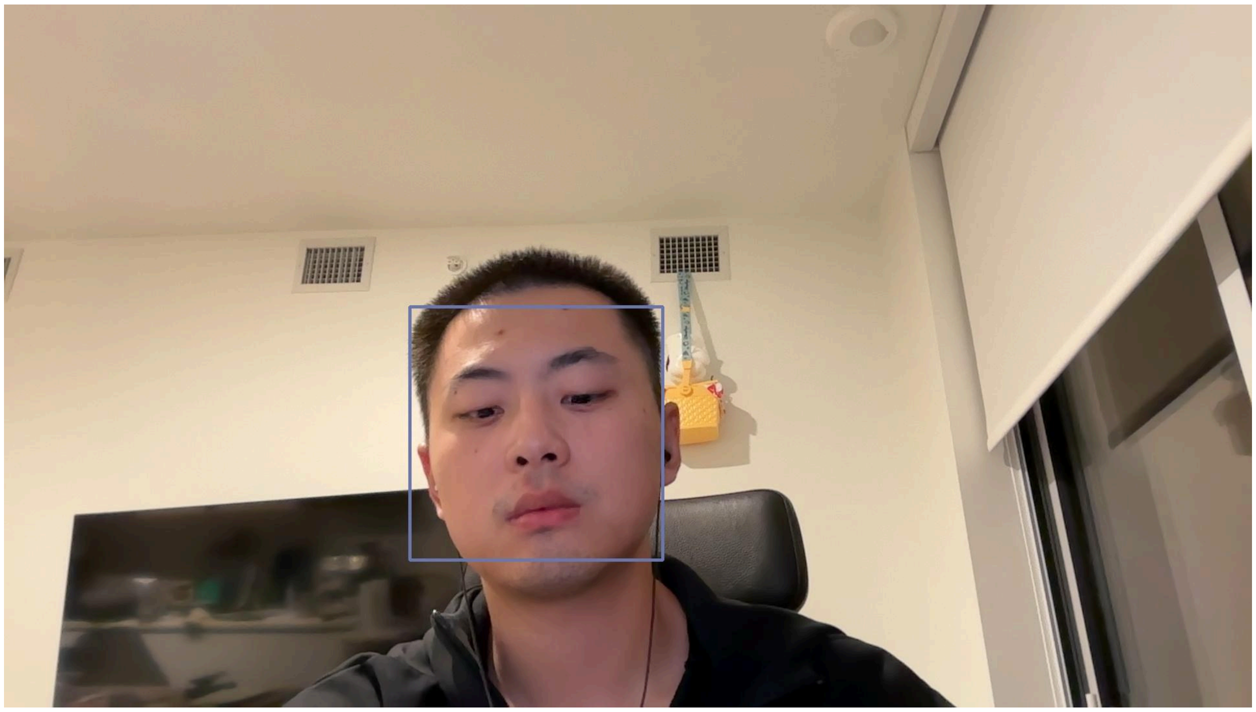
Gradient magnitude image:



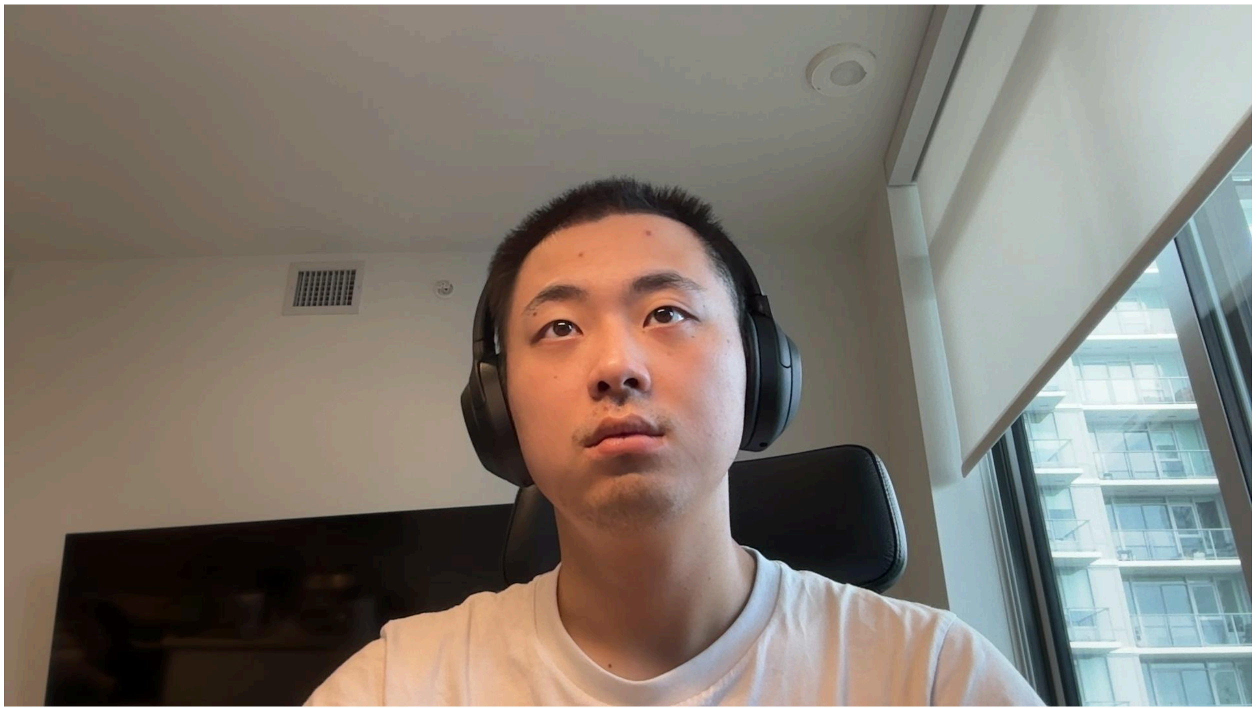
Required Image 6:
Origin Image



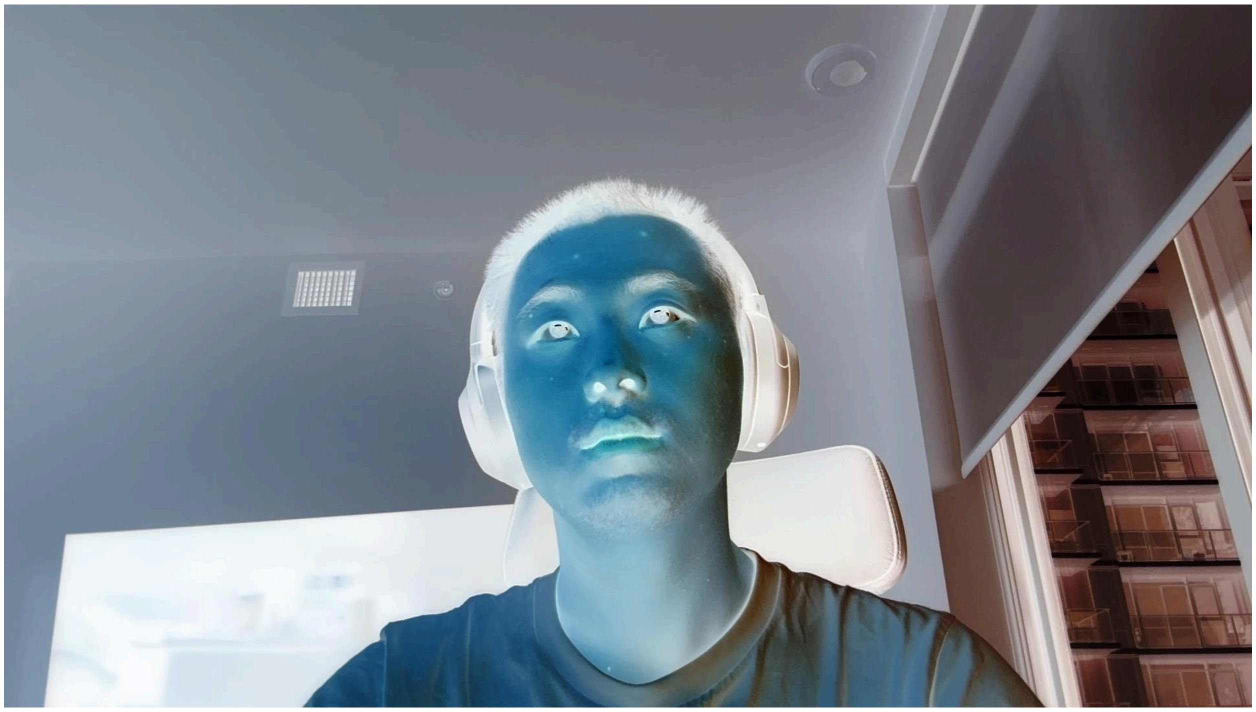
Required Image 7:
Face detector



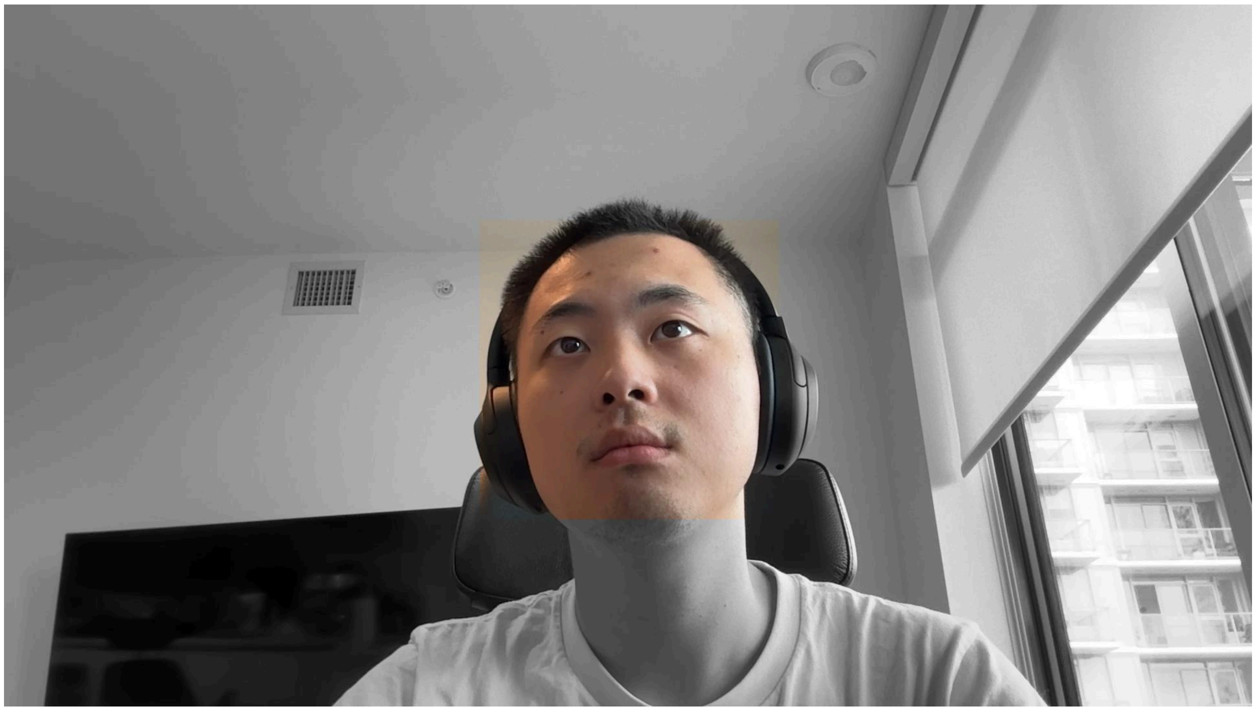
Required Image for effects of choice:
Original Image:



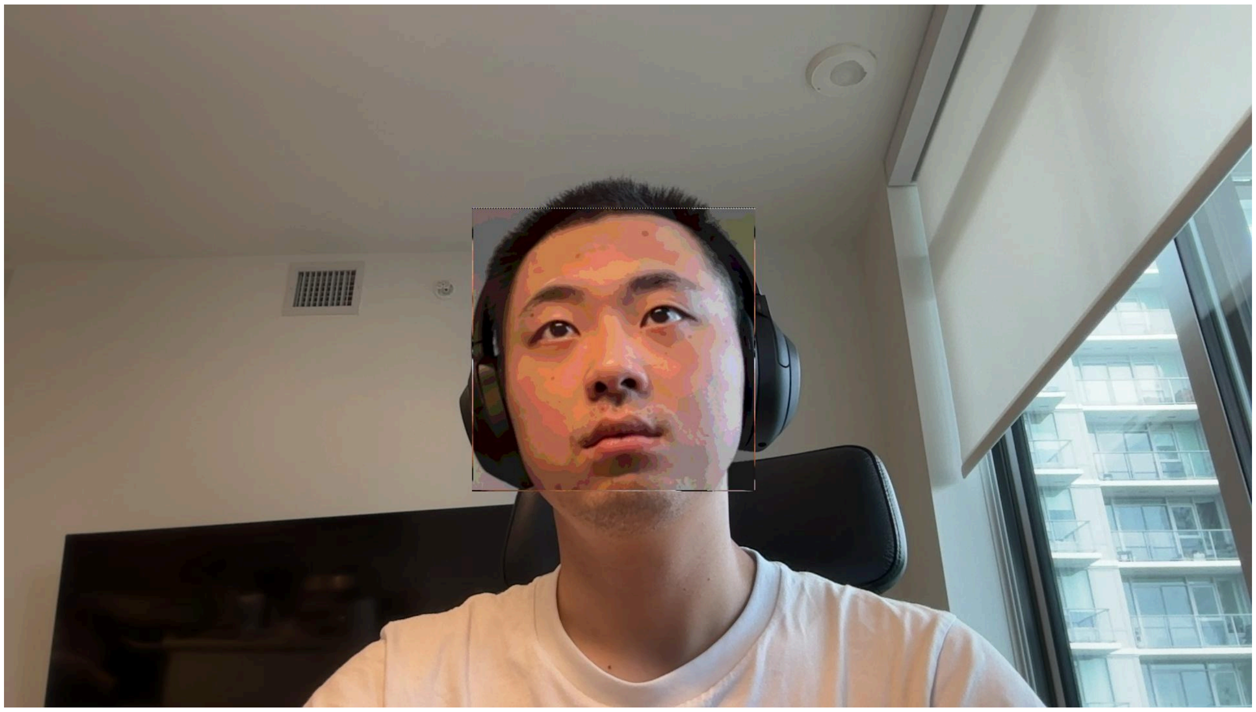
Invert color:



Color face only:



Blurred and quantized:



What I learned

Through this project, I gained a deeper understanding of using C++ and OpenCV for image processing and manipulation. I learned how to work with OpenCV's Mat structure and directly manipulate pixels to create custom filters. By implementing tasks like reading and displaying images, capturing live video, and applying filters such as grayscale, blur, and Sobel edge detection, I gained hands-on experience in real-time video processing.

I also learned that when applying filters that require neighboring pixel values, we shouldn't modify the original frame directly, as this could result in multiple modifications of the same data. Additionally, I discovered that OpenCV does not store data in the RGB order but rather in BGR, which can lead to output format changes depending on the processing method. It's important to consider output format conversion when necessary. When handling color data, it's crucial to ensure that the pixel values remain within the 0-255 range, and steps must be taken to prevent values from exceeding this limit.

By experimenting with different methods to optimize performance, such as using separable filters for faster blur operations, I also learned how to combine face detection with image processing through custom filters like color inversion and applying effects to specific areas (e.g., the face). This project improved my skills in OpenCV for both creative and real-time applications, while also teaching me to handle performance constraints and data range limitations effectively.

Material and people you consulted:

"Makefile Tutorial." https://cs.colby.edu/maxwell/courses/tutorials/maketutor/.

"OpenCV Tutorials" https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html.

ChatGPT

Peiyao Tao