Name: Leaksmy Heng

Class: CS5330

Date: Jan 26 2025

Project 1: Video-special effects

1. Short description of the overall project:

This project has been incredibly challenging, yet enjoyable. The challenging parts

were because I have never really used C++ before; therefore, I was not familiar with the syntax. On top of that, I had to map the mathematical approaches (convolution) into the code which I found incredibly difficult. However, after implementing various filters in the code (Sobel, Gaussian, etc.), I became more familiar with the syntax and was able grasp it better. Despite all the challenges, I found this project incredibly enjoyable. The instant result once the filter work gives me a sense of accomplishment and makes me feel great about the progress I’ve made. I am also working on this project alone; therefore, I think I’ve learned a lot from this.

1. Required Images and description:
   1. **Required Image 1**: Comparison between the colored image, and grayscale image implemented by OpenCV

Image before pressing “g”

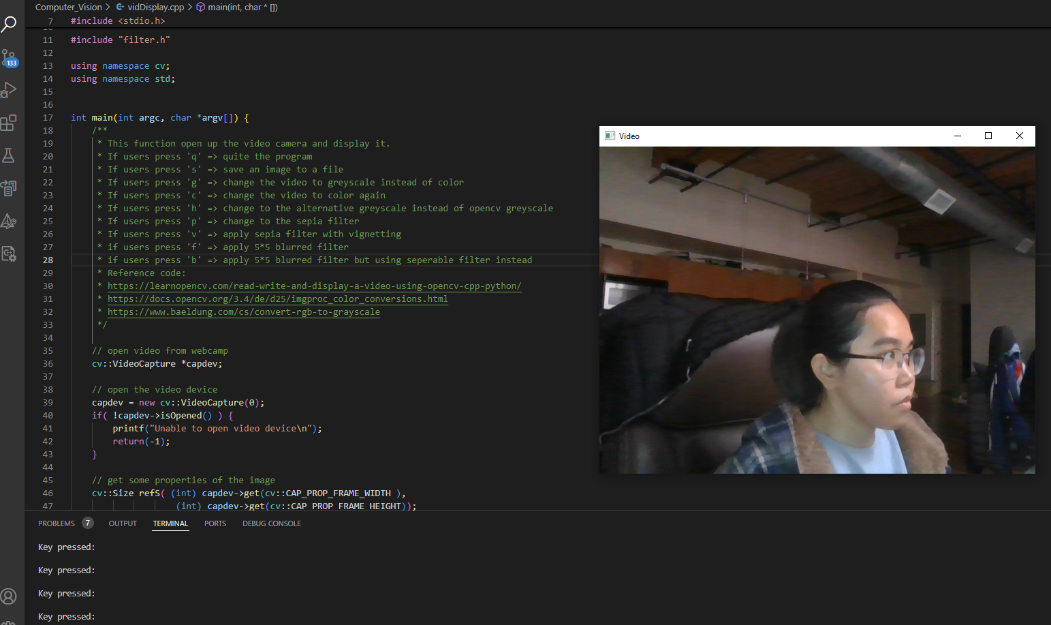
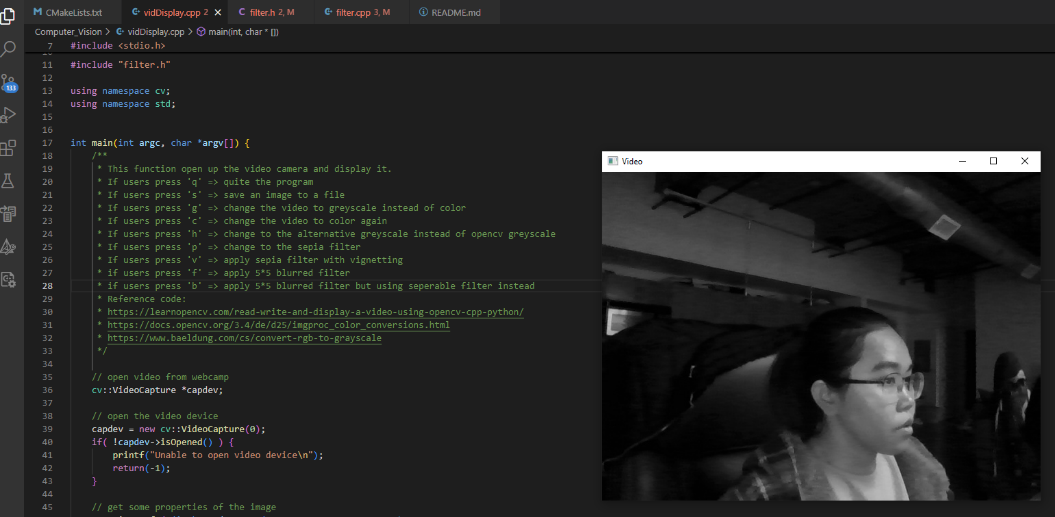
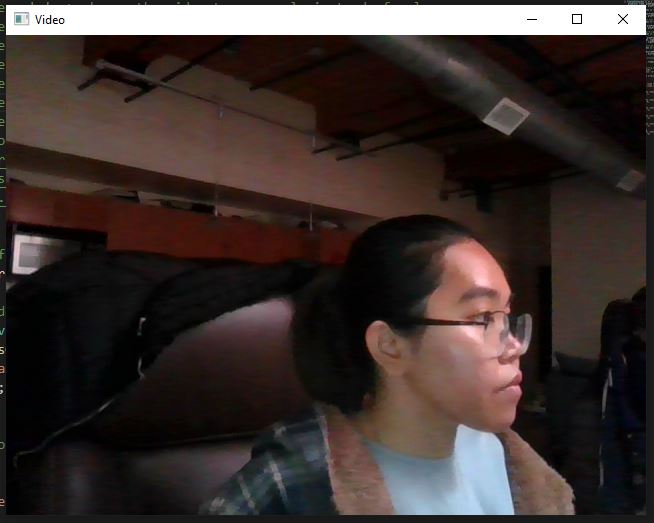
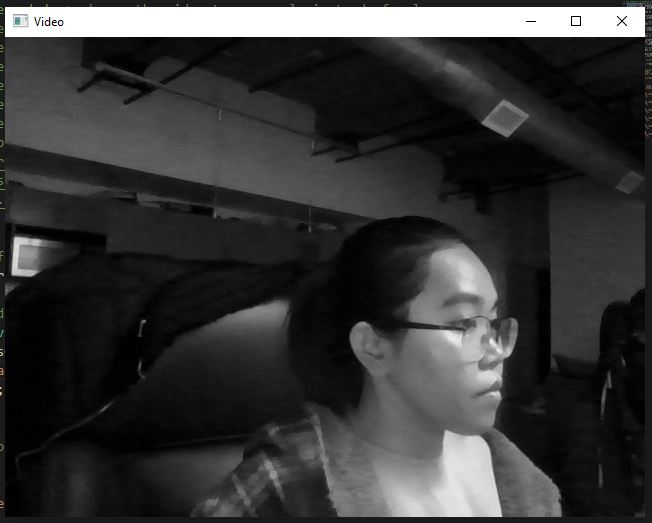


Image after pressing “g”

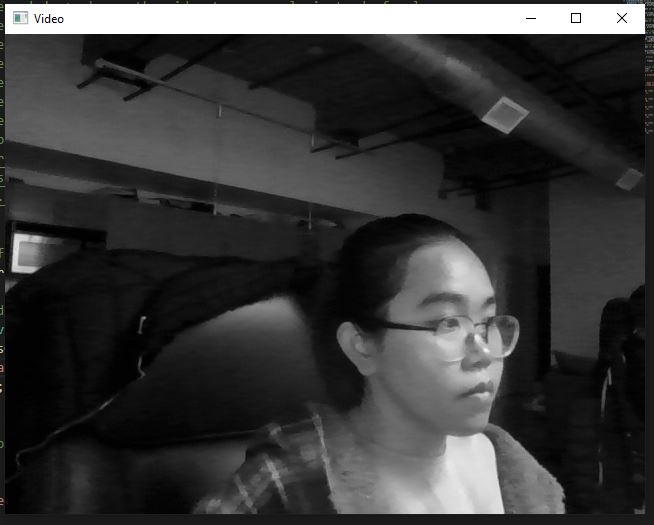


* 1. **Required Image 2**: Compared between the colored image, grayscale image implemented by OpenCV, and the grayscale.
     1. I used the average method to generate the alternative grayscale image. The formular for that is (R+G+B) / 3; whereas OpenCV uses the luminosity method where each of the R, G, B has its own weight. The formular for that is 0.3R + 0.59G + 0.11B (the coefficient here is round up).
     2. I’ve noticed that OpenCV grayscale method is lighter than the average method I used. This is because OpenCV accounts for the sensitivity of the human eyes to different colors. Therefore, it gives more weight to green, less to red and the least to blue.

My Gray Image



Colored Image



OpenCV Gray Image

* 1. **Required Image 3**:
     1. To ensure that the original RGB values are used to compute, I accessed it and stored it in the unsigned char named blue, green, and red. Blue stored pixel[0], green stored pixel[1], and blue stored pixel[2]. With each of the pixel stored, I then used it to compute newBlue, newGreen and newRed variable based on the coefficient used to compute the Sepia filter. Only after the newBlue, newRed and newGreen are computed that I assigned it to pixel[0], pixel[1], and pixel[2].
     2. **Extension**: Vignetting Edge is applied to the sepia filter. First, compute the radius of the image to apply the vignette. This step determines where the darkening effect starts to become noticeable. Then I loop through each row and column to calculate the distance between that coordinate to the center point that we have calculated from above. After that I computed the weight of the vignette. The smaller vignette weights result in darker area. Last but not least, I multiplied the pixel by the vignette weight.

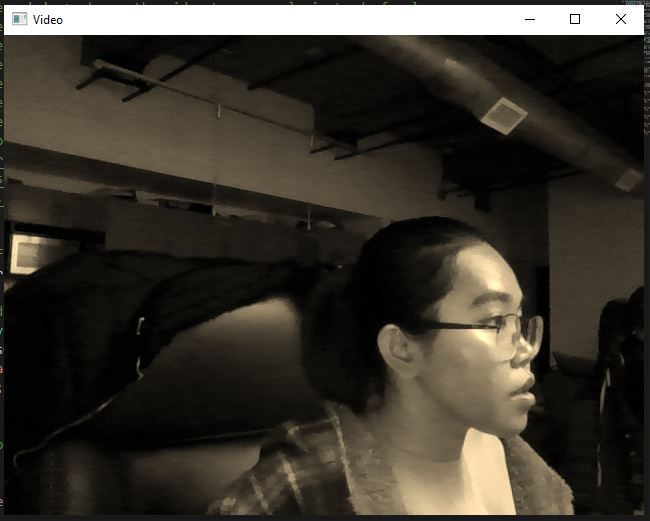


Image with Sepia Filter

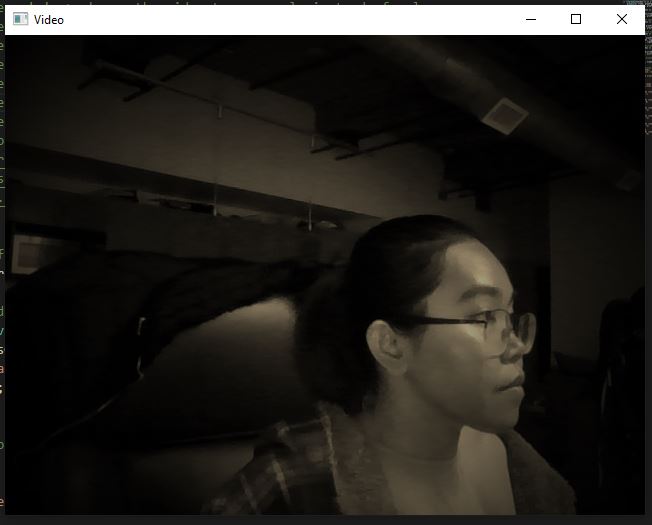


Image with Sepia Filter and Vignetting Edge

* 1. **Required Image 4:** Blurred filter and it timing information. As can be seen from the Timing Information image, the 5x5 gaussian filter using separable approach is more efficient compared to the regular 5x5 gaussian.

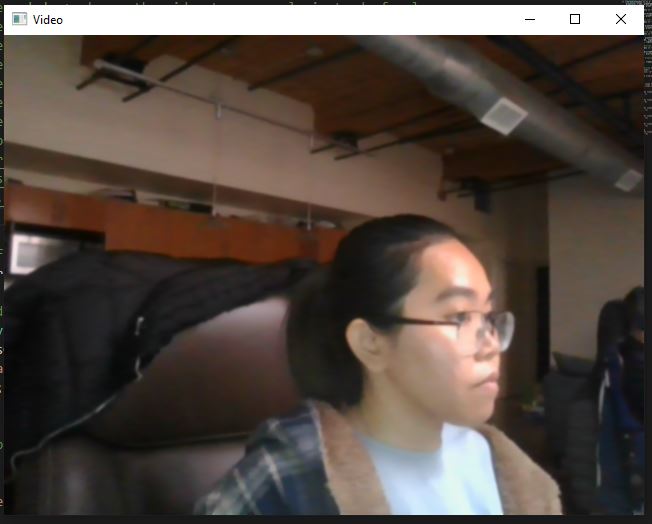
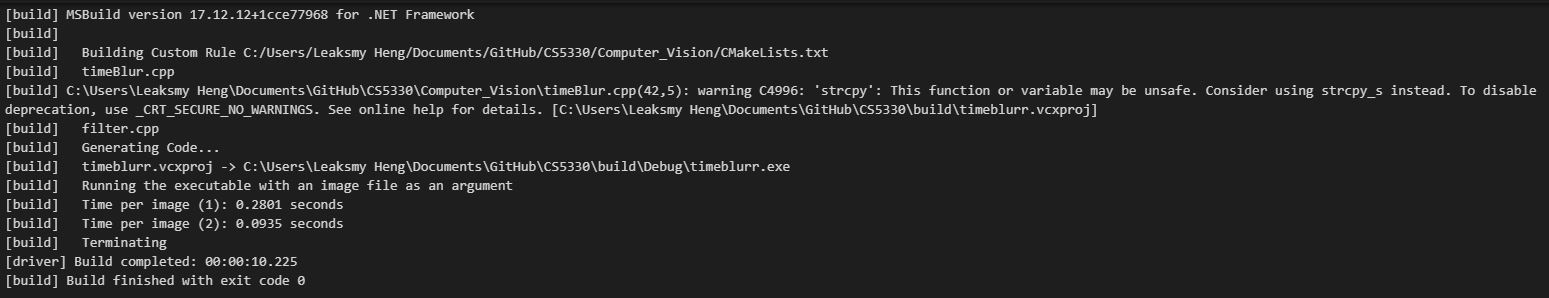
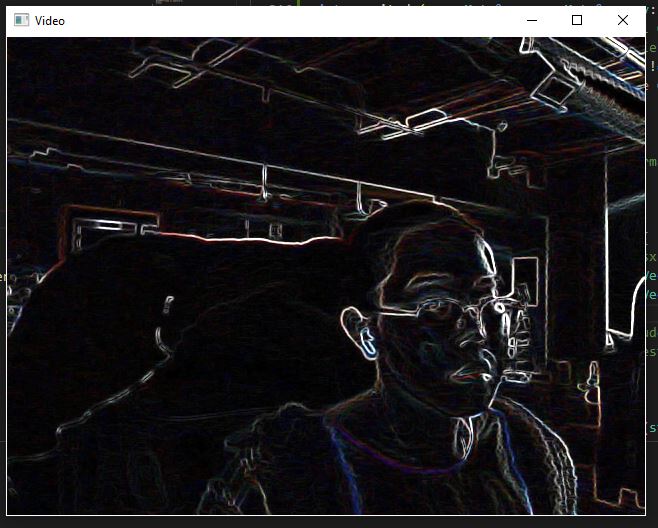


Image with Blurred filter

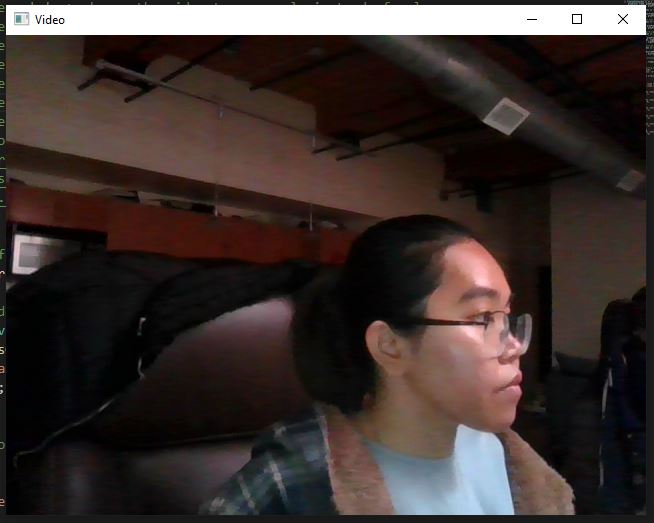
Timing Information



* 1. **Required Image 5**: Generate gradient magnitude image from x and y sobel frame



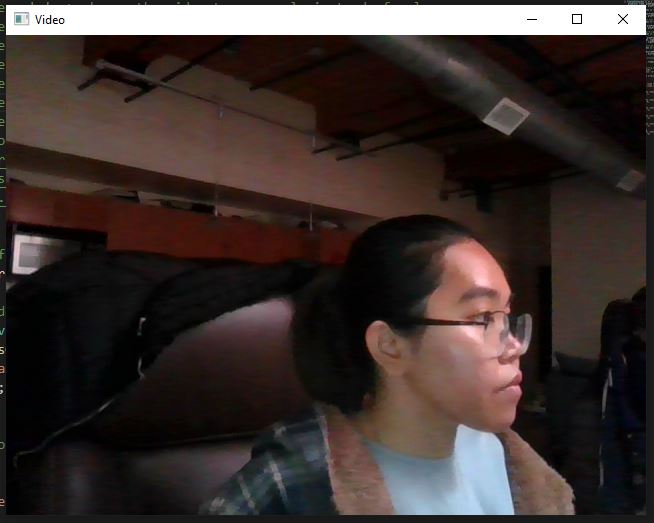
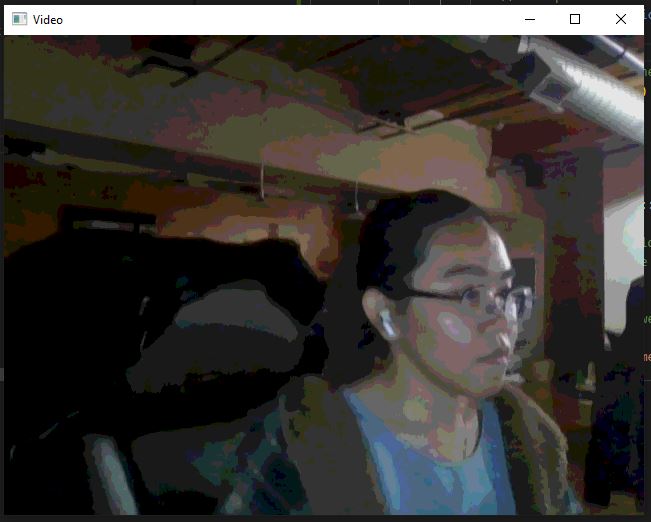
Generate gradient magnitude frame



Original frame

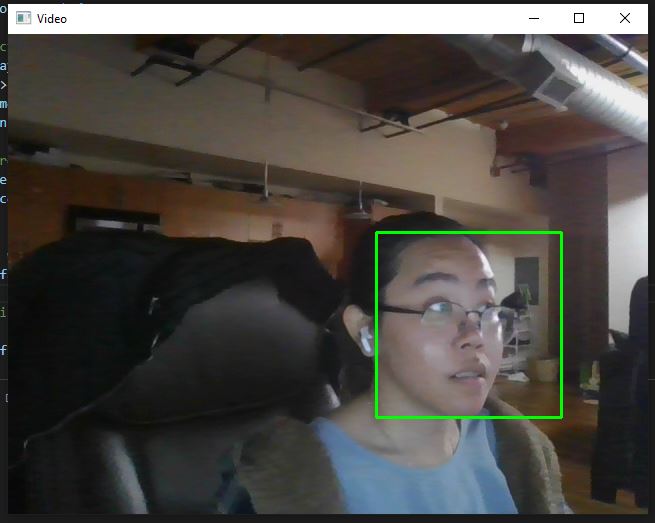
* 1. **Required Image 6:**

Blurred Quantize



Original frame

* 1. **Required Image 7:**



Face being detected on the camera

* 1. **Required Image 8:**

1. Reflection of what I have learned
2. Acknowledgements

<https://www.baeldung.com/cs/convert-rgb-to-grayscale>

<https://stackoverflow.com/questions/51818193/problems-with-using-a-rough-greyscale-algorithm>

<https://www.geeksforgeeks.org/gaussian-filter-generation-c/>