Name: Leaksmy Heng

Class: CS5330

Date: Jan 26 2025

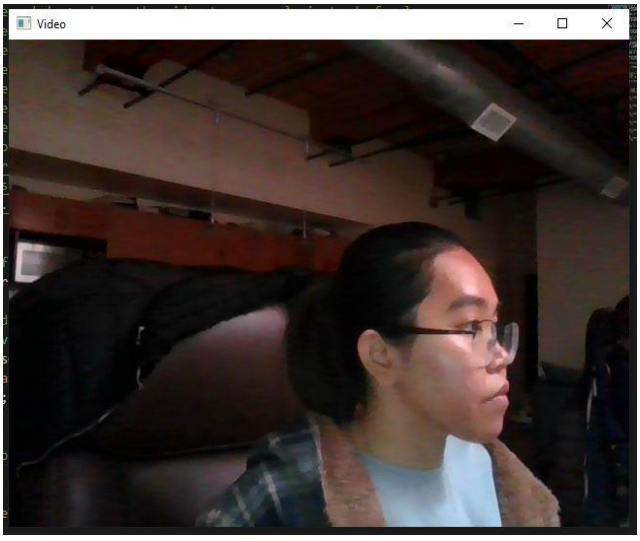
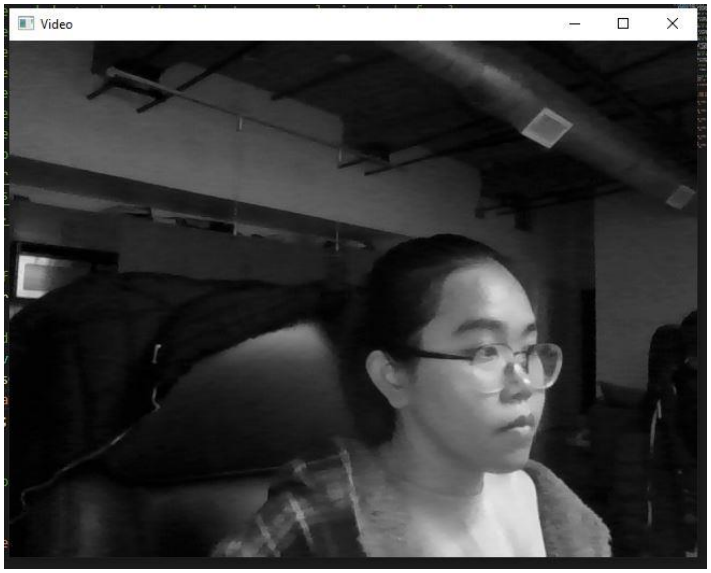
Project 1: Video-special effects

---

1. Short description of the overall project:

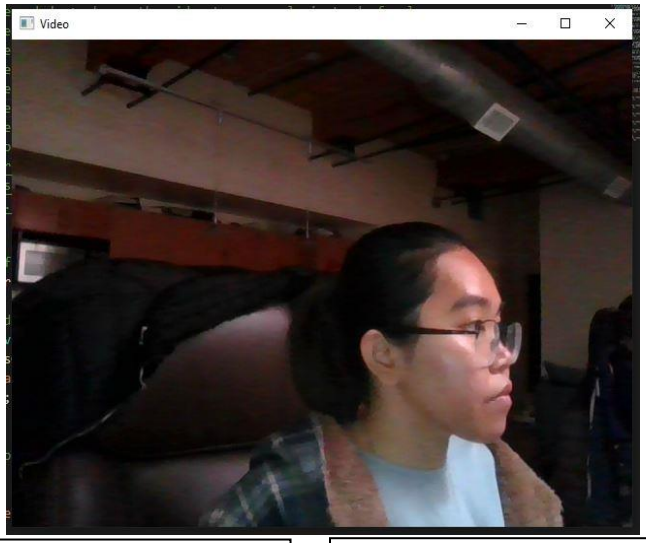
   This project has been incredibly challenging, yet enjoyable. The challenging parts were because I have never really used C++ before; therefore, I was not familiar with the syntax. On top of that, I had to map the mathematical approaches (convolution) into the code which I found incredibly difficult. 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 filters 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.

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

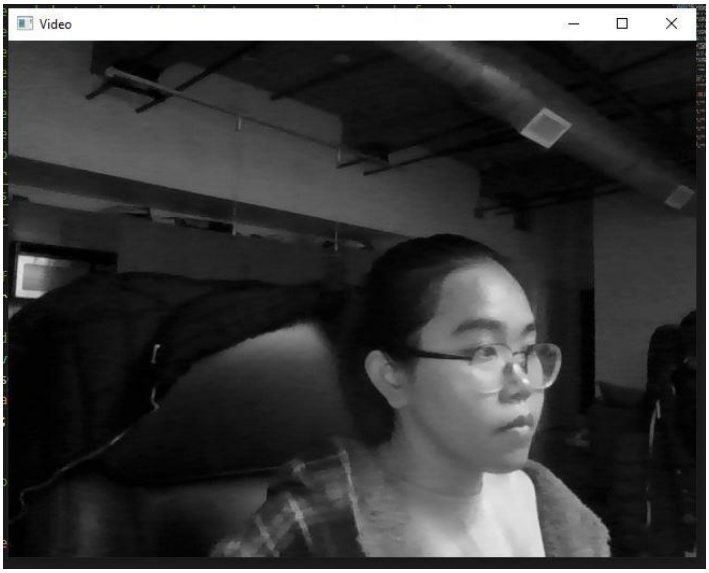
Colored Image	OpenCV Gray Image
	

b. **Required Image 2**: Compared between the colored image, grayscale image implemented by OpenCV, and the grayscale.
    i. 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).
    ii. 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.

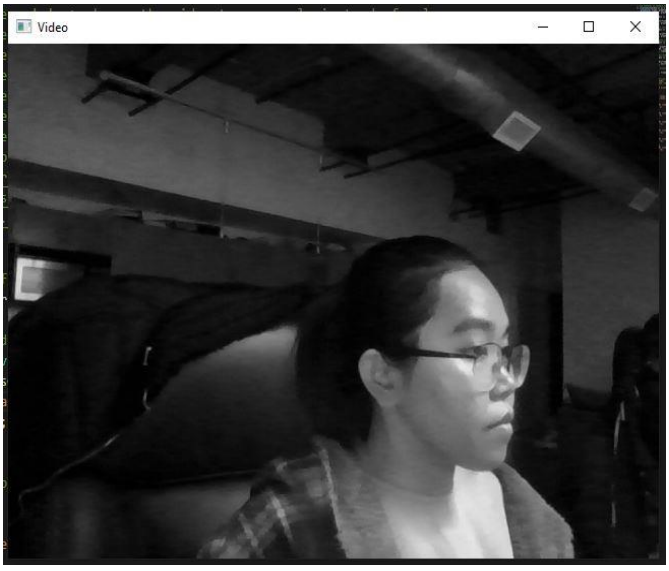
Colored Image



OpenCV Gray Image



My Gray Image



c. **Required Image 3**:

i. 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].

ii. **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. Finally, I multiplied the pixel by the vignette weight.

Image with Sepia Filter

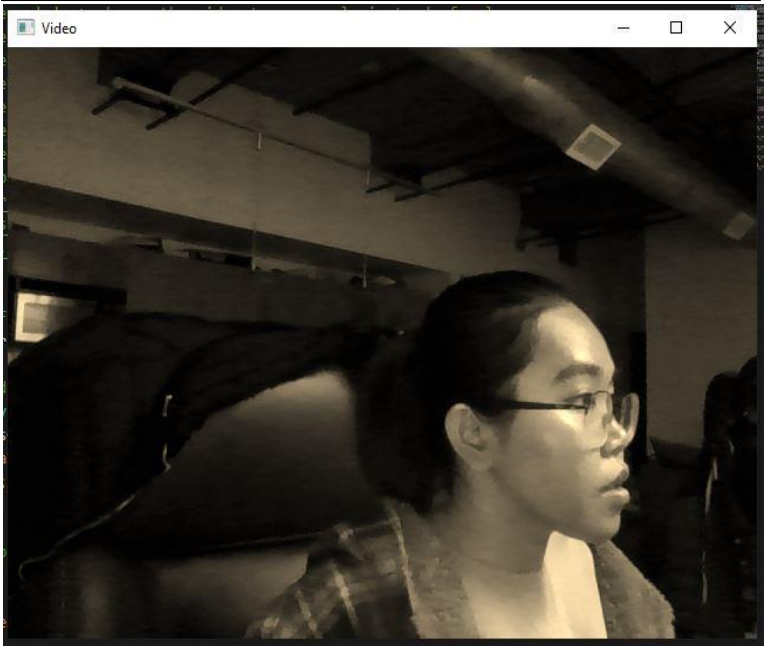
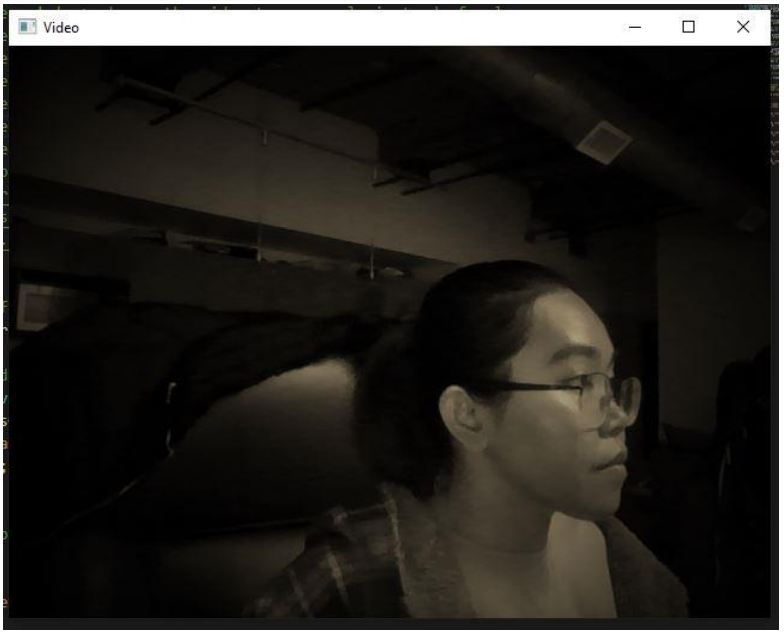
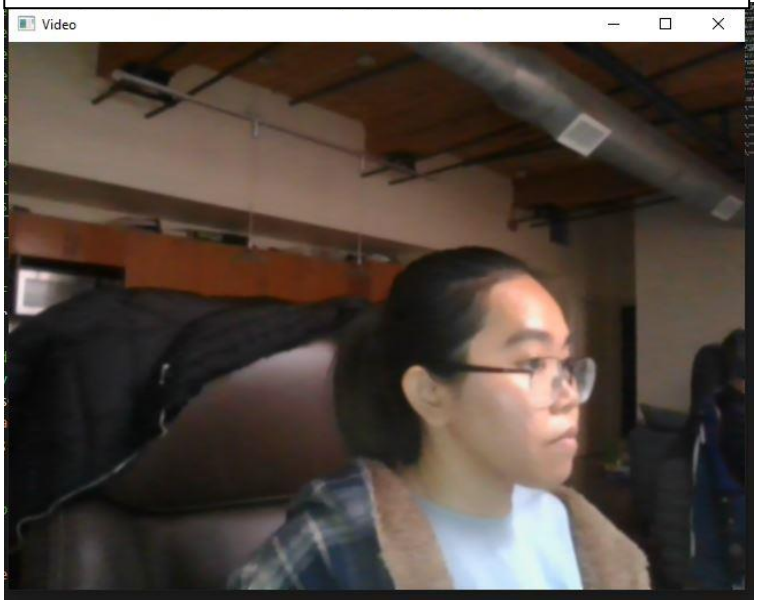


Image with Sepia Filter and Vignetting Edge



d. **Required Image 4:** Blurred filter and its 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, 0.2881 seconds and 0.0935 seconds respectively.

Image with Blurred filter

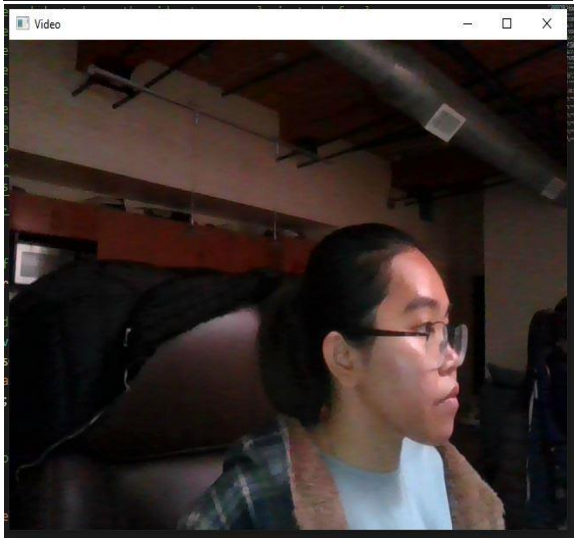


Timing Information

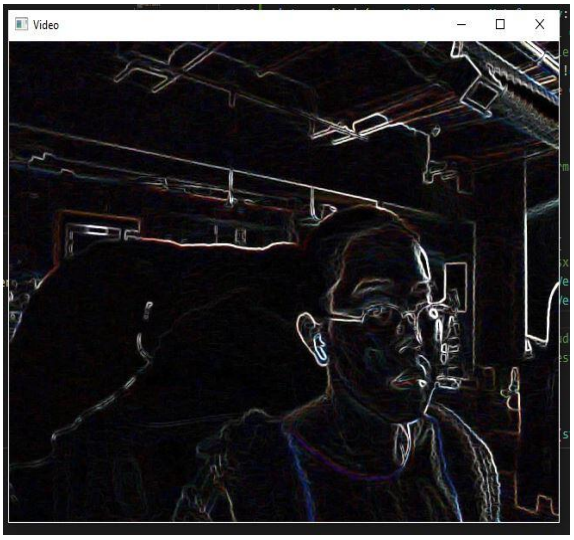
```
[build] MSBuild version 17.12.12+1cce77968 for .NET Framework
[build]
[build]   Building Custom Rule C:/Users/Leaksmy Heng/Documents/GitHub/CS5330/Computer_Vision/CMakeLists.txt
[build]   timeBlur.cpp
[build] C:\Users\Leaksmy Heng\Documents\GitHub\CS5330\Computer_Vision\timeBlur.cpp(42,5): warning C4996: 'strcpy': This function or variable may be unsafe. Consider using strcpy_s instead. To disable
deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details. [C:\Users\Leaksmy Heng\Documents\GitHub\CS5330\build\timeblurr.vcxproj]
[build]   filter.cpp
[build]   Generating Code...
[build]   timeblurr.vcxproj -> C:\Users\Leaksmy Heng\Documents\GitHub\CS5330\build\Debug\timeblurr.exe
[build]   Running the executable with an image file as an argument
[build]   Time per image (1): 0.2801 seconds
[build]   Time per image (2): 0.0935 seconds
[build]   Terminating
[driver] Build completed: 00:00:10.225
[build] Build finished with exit code 0
```

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

Original frame

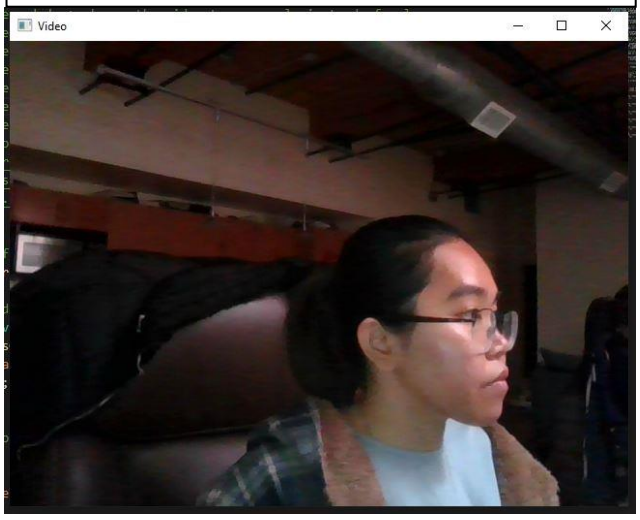


Generate gradient magnitude frame

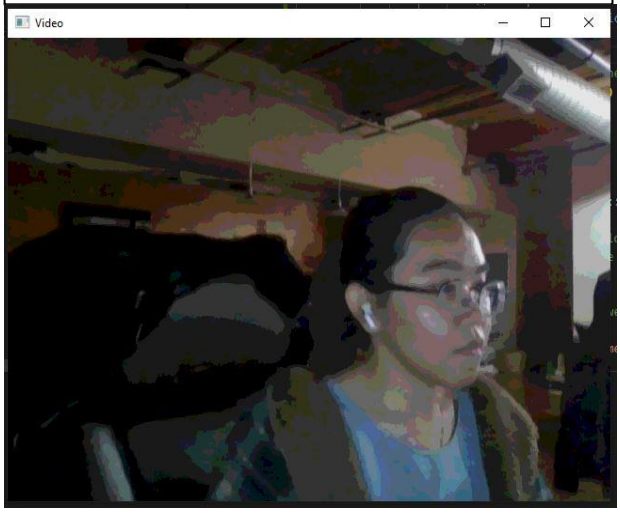


f. **Required Image 6:** Generate Blurred Quantize image.

Original frame

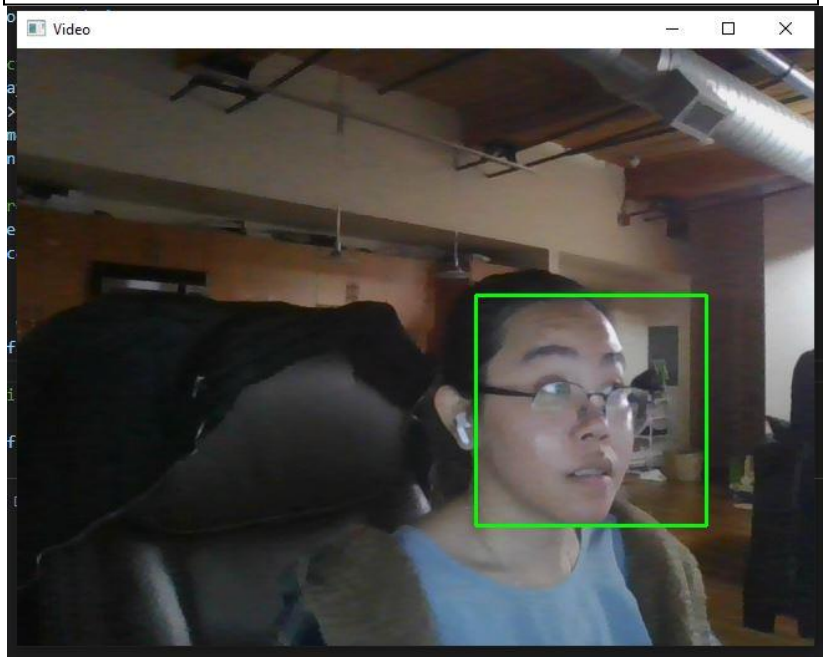


Blurred Quantize



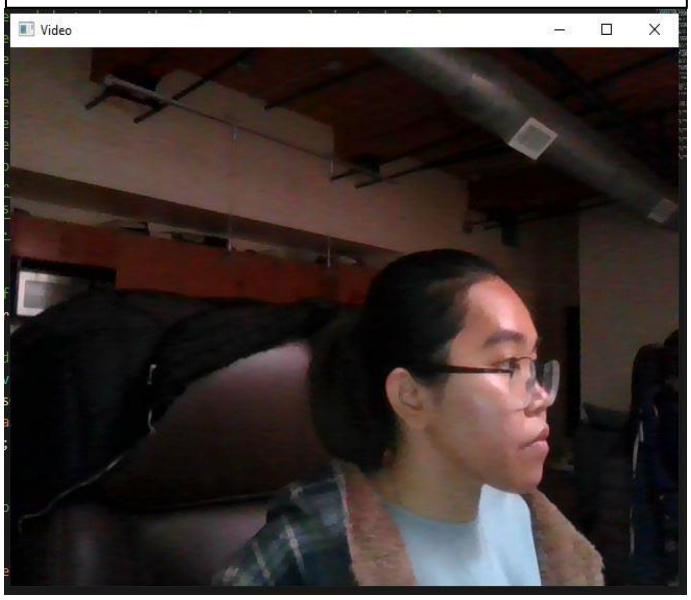
g. **Required Image 7:** Implementing facial detection.

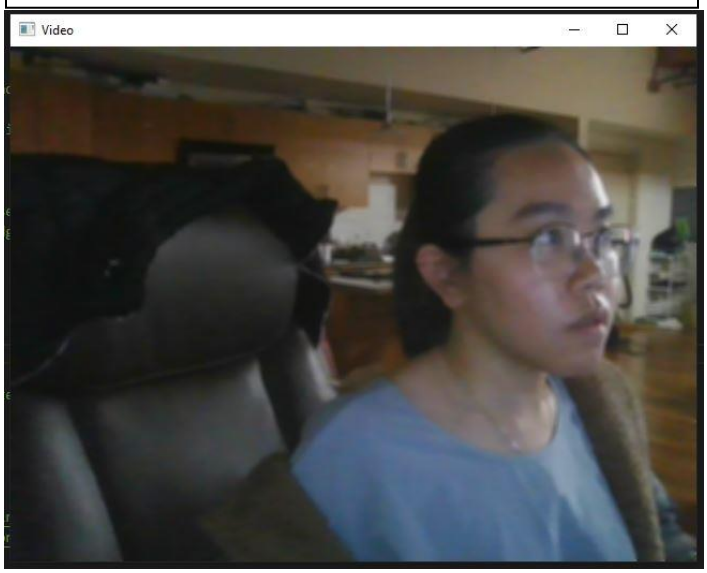
Face being detected on the camera



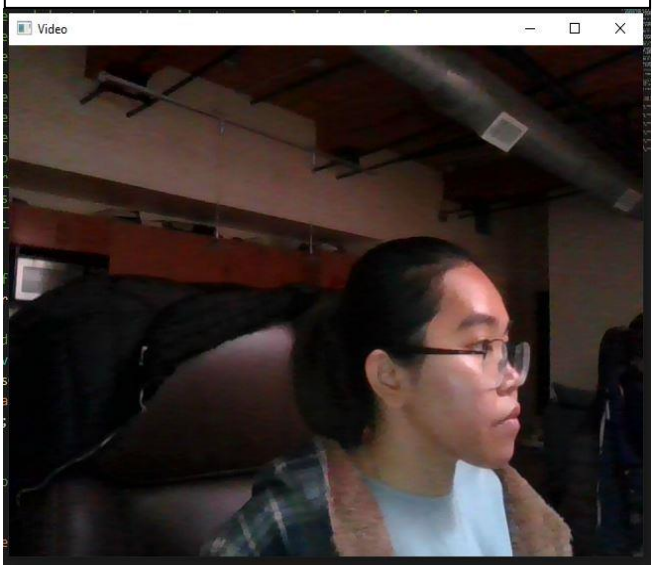
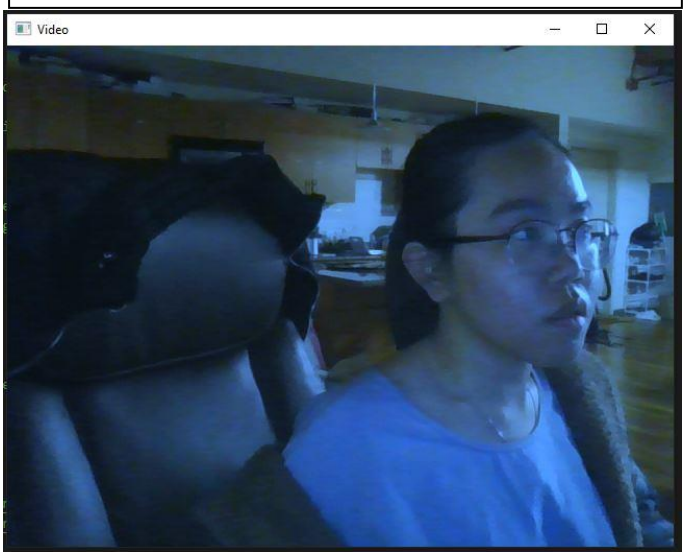
h. **Required Image 8:** I was not able to get the onnxruntime library working on my system. Therefore, I implemented low pass filter instead. Low pass filter creates a smoothing effect to the image by averaging nearly pixel value resulting in a blurred appearance as shown below.

Original frame	Low pass filter

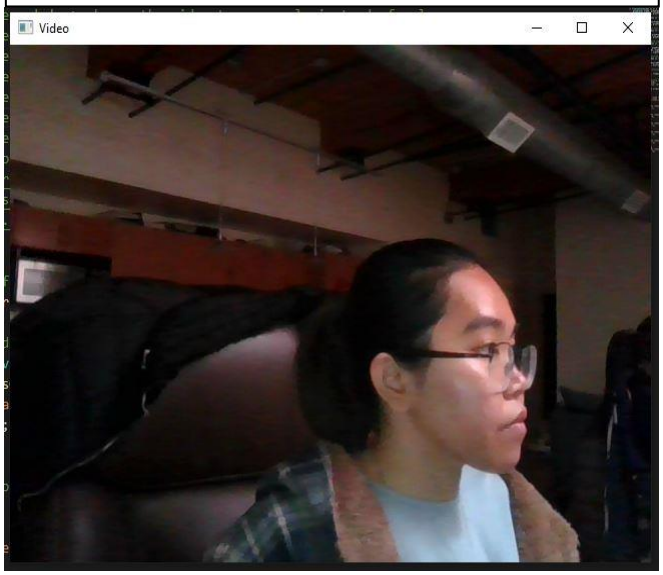
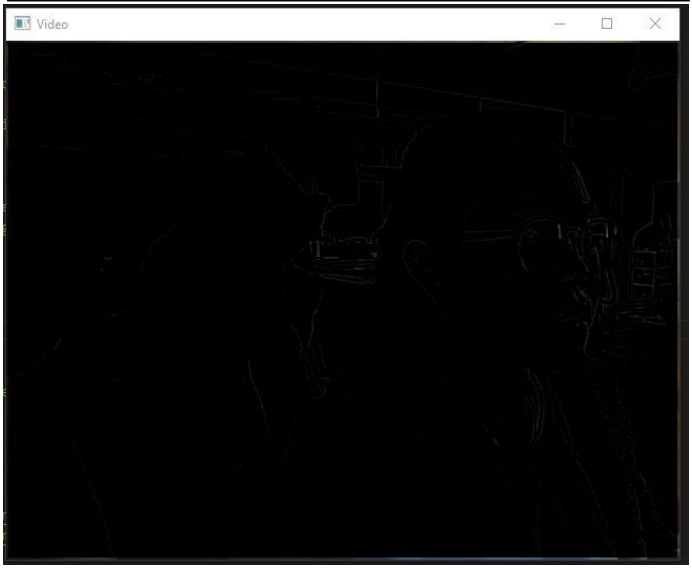




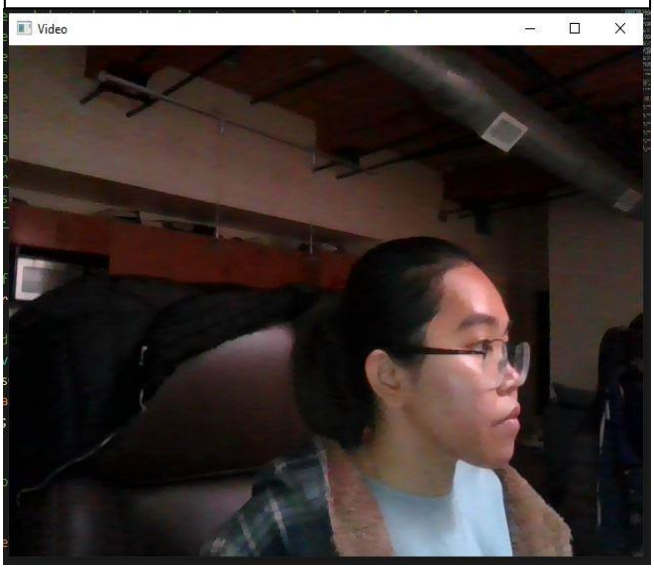
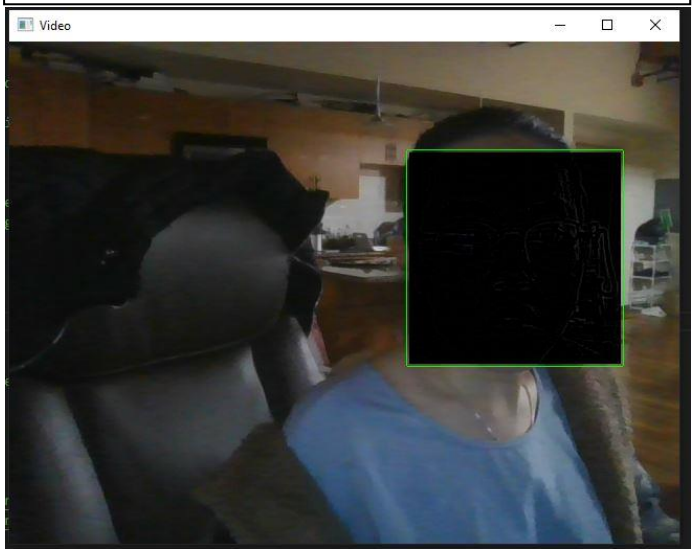
i. **Required Image 9:** This is the first effect applied in part 12 of the assignment. I implemented a cool tone filter which is a single-step pixel-wise modification. A cool tone filter. Cool Tone filter is implemented by increasing blue channel, and decrease red channel but leave green as is making the image has a blue / green tone to it.

Original frame	Cool Tone Filter
	

j. **Required Image 10:** This is the second effect applied in part 12 of the assignment. I implemented high pass filter. While low pass filter smooth the image by averaging nearly pixel value, high pass filter enhance the sharpness of the image by subtracting the low pass pixel from the original image (src) pixel.

Original frame	High Pass Filter
	

k. **Required Image 11:** This is the third effect applied in part 12 of the assignment. I implemented a high pass filter on a face detection. Therefore, the area that is not detected as face will have the regular RGB channels, but the facial region will have high pass filter apply to it.

Original frame	High Pass Filter on Facial Detection
	

3. Reflection of what I have learned

I've gained a lot of knowledge in various image processing techniques, including applying filters such as Gaussian, Sobel, and mean (through both low-pass and high-pass filters), face detection, etc. The first task, in particular, was quite challenging for me, especially when implementing the 5x5 blurred image, as I had to translate mathematical convolution into code, which I've never done before. I also explored different methods for converting images to grayscale. What truly fascinated me was learning about the coefficients used in the luminosity method, which are tailored to the sensitivity of human vision. I'm inspired by the time and effort scientists have invested in developing these techniques, and I find myself admiring their work. Considering this is our first project, I am really looking forward to learning more about image processing.

4. 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/

https://stackoverflow.com/questions/1696113/how-do-i-gaussian-blur-an-image-without-using-any-in-built-gaussian-functions

https://www.geeksforgeeks.org/difference-between-low-pass-filter-and-high-pass-filter/#

https://vineethvasu25.medium.com/digital-image-processing-smoothing-low-pass-filter-4b04f154631b

https://www.youtube.com/watch?v=-LD9MxBUFQo&list=PL2zRqk16wsdorCSZ5GWZQr1EMWXs2TDeu&index=6