CS5330 Project 1: Real-Time Filtering Report

Submission by - Sriram Pandi

Task-1) Read an image from a file and display it -

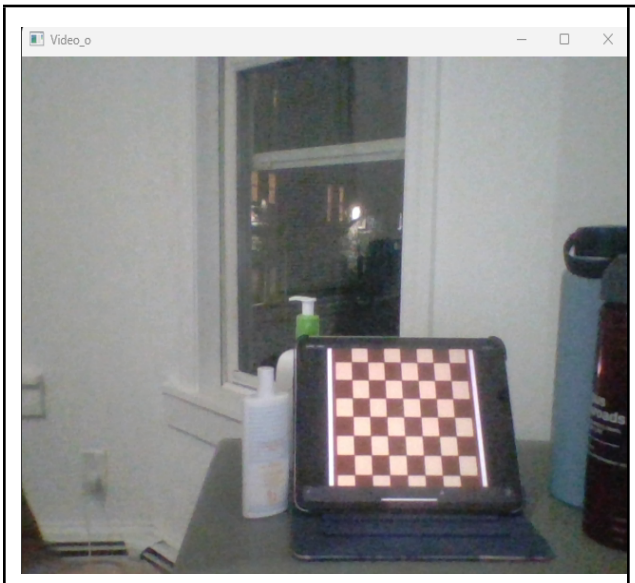
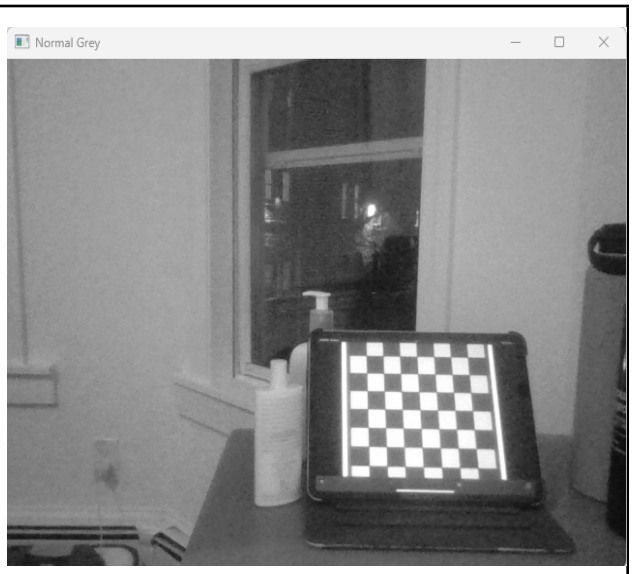
A file was created with the name imgDisplay.cpp and a path of the image file was given to display on a window and added an option to close the window and quit if typed ‘q’.

Task-2) Display live video -

Used the code snippet given in the question and included the ability to add save option to save the specific frame.

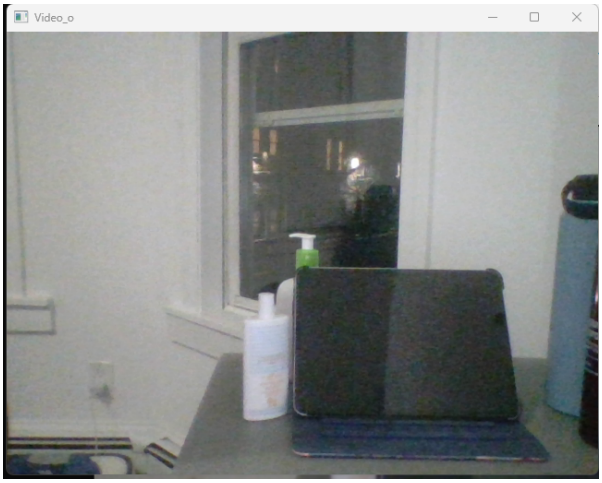
Task-3)

Obtained a greyscale image of the original frame using the function cvtColor(img, res, COLOR_BGR2HSV) and displayed the output in a window named normal Grey.

	
Original Image	**Normal Gray scale image**

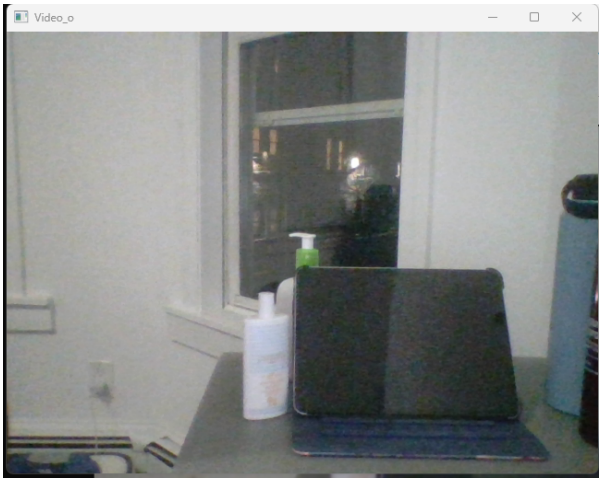
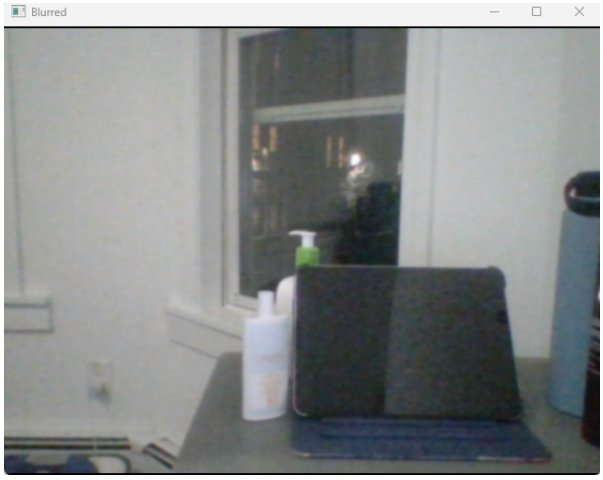
Task-4)

A function was written that converts a color image (src) to greyscale (dst). It does this by looping over each row and column of the src image, using a pointer to access each pixel. For each pixel, the code calculates its greyscale value using a weighted average of its R, G, and B components, where the weighting factors used are 0.6 for the green channel, 0.2 for the red channel and 0.2 for the blue channel. The result is stored in the dst image.

	
Original Image	**Alternate-Grey scale Image**

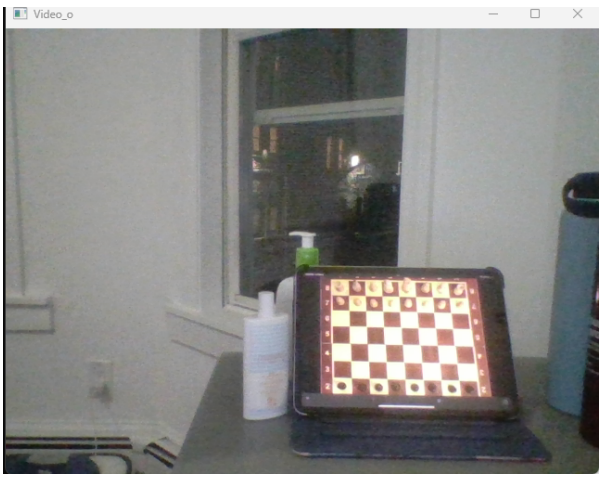
Task-5) Gaussian blur

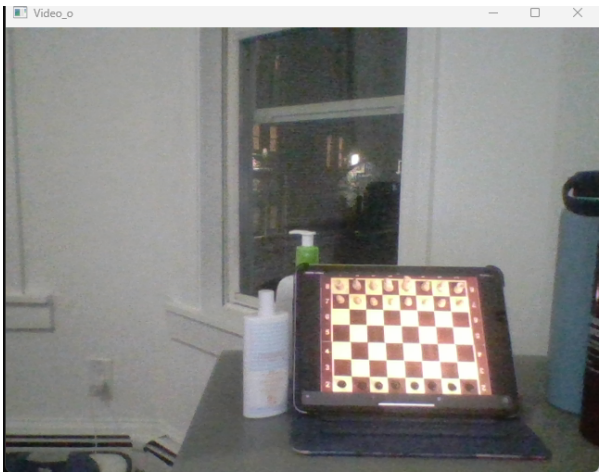
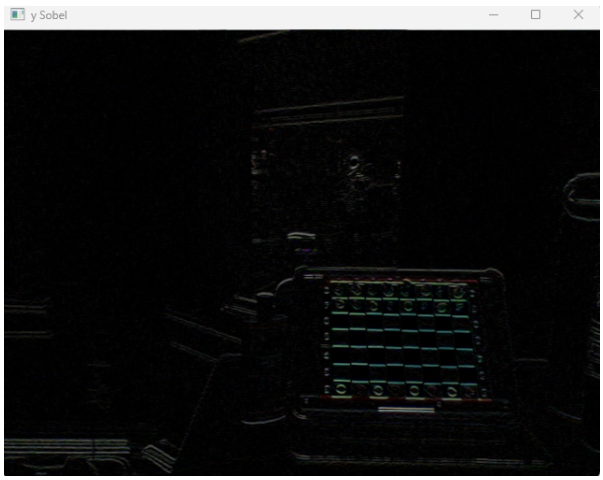
Implemented a 5x5 Gaussian filter as separable 1x5 filters ([1 2 4 2 1] vertical and horizontal). Implemented the function by accessing pixels,.The input is a color image (type vec3b) and the output is also a color image (type vec3b).if the user types 'b' it displays a blurred version of the image (in color).

	
Original Image	**Gaussian Blurred image**

Task-6)

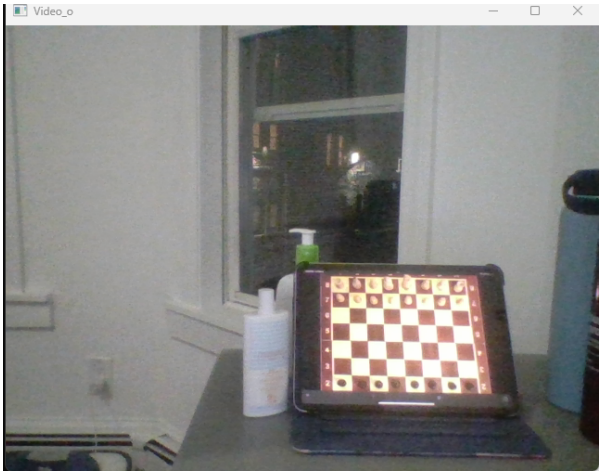
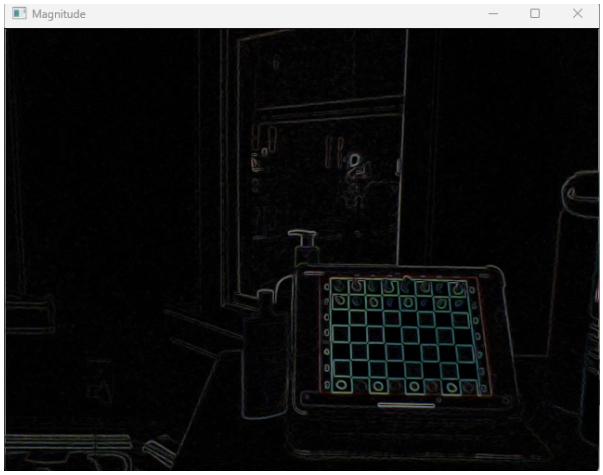
created two new functions, sobel-x ,sobel-y. Each implements a 3x3 Sobel filter, either horizontal (X) or vertical (Y). The X filter is positive right and the Y filter is positive up. Both the input and output images are color images, but the output is of type 16S (signed short) because the values would be in the range [-255, 255]. If the user types X key then, the 'x' key shows the X Sobel (shows the absolute value), and the 'y' key shows the Y Sobel.

	
Original Image	**Sobel-x image**

	
Original Image	**Sobel-y Image**

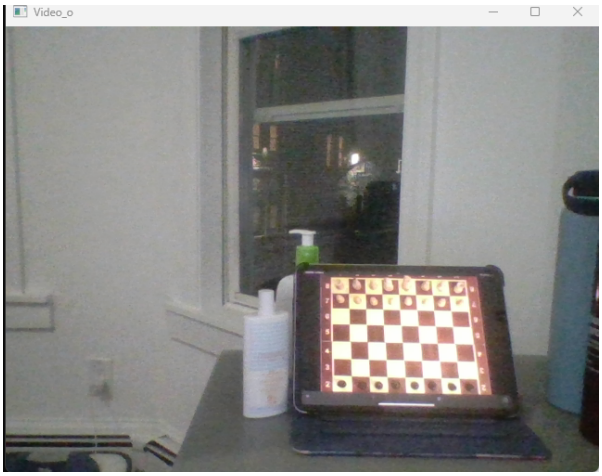
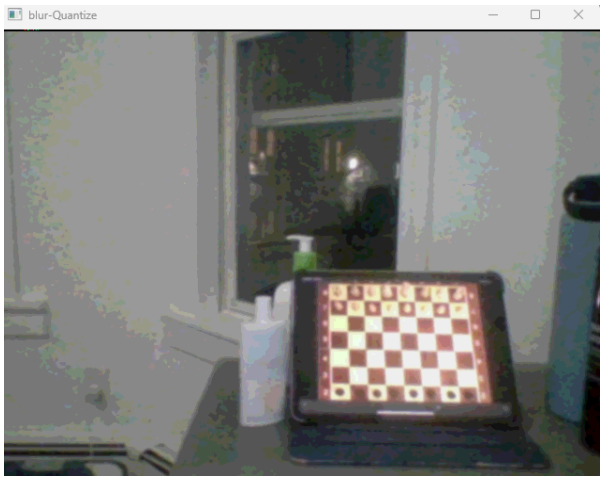
Task-7)

A function was written that generates a gradient magnitude image using Euclidean distance for magnitude: I = sqrt(sx*sx + sy*sy). This is still a color image. The two input images will be 3-channel signed short images, but the output is a uchar color image suitable for display.in order to achieve that we converted the output to absscale.

	
Original Image	**Gradient Magnitude Image**

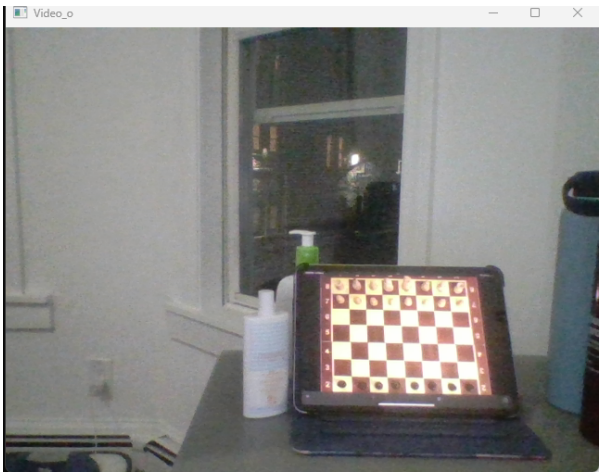
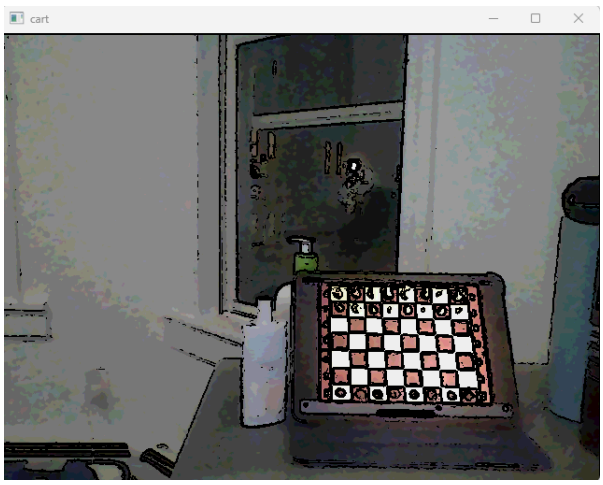
Task-8)

A function was written that takes in a color image, blurs the image, and then quantizes the image into a fixed number of levels as specified by a parameter.First a variable bucket was calculated using b = 255/levels. Then a color channel value x was taken and execute xt = x / b,followed by executing xf = xt * b. After executing this for each pixel and each color channel, the image was left with only levels**3 possible color values.If pressed “l” then the image gets blur quantized. The default level taken was 15.

	
Original Image	**Blur-Quantized image**

Task-9)

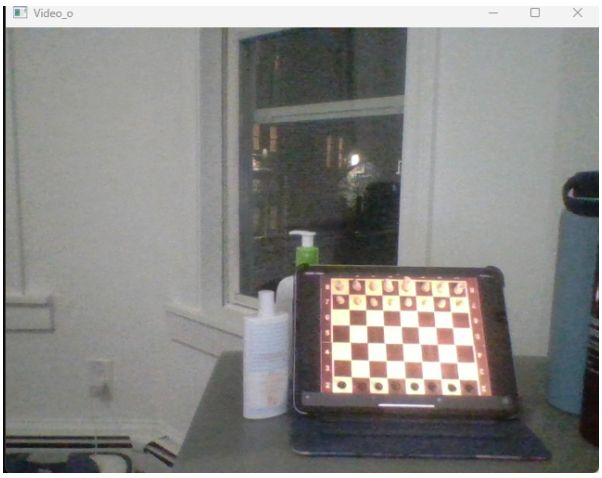
A function that cartoonize a color image was written. This is done by first calculating the gradient magnitude followed by blurring and quantizing the image. Finally, changed the blurred and quantized image by setting pixels with a gradient magnitude larger than a threshold to black i.e ‘0’.Here the threshold value was taken to be 20.

	
Original Image	**Cartoon image**

Task-10) Another effect to implement on your video

Alternate effect-"Making the image a negative of itself"

This program computes the negative value of every pixel by subtracting the pixel value from 255, which is the maximum value for an 8-bit image channel. The result is stored in the corresponding pixel in the destination image. By pressing 'n', the negative image is displayed.

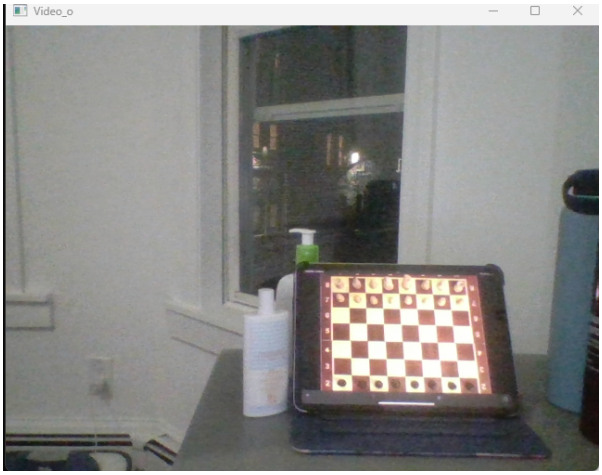
	
Original Image	**Negative image**

Extension-1)

The following extension can be titled as color corectiona and thresholding. Explaining about the method, the method receives two input images (src and dst) and applies thresholding and color correction to the input image src. A gamma correction with a gamma value of 1.2 was used as the adjustment.The image is changed from BGR to YCrCb color space, and the Y channel is corrected.

The image is then changed back to grayscale and BGR after being rectified. The grayscale picture is then given a binary threshold, transferred to the output image dst, and finally inverted.I was just trying to use multiple inbuilt functions and get a new filter.

And it turned out to be the below filter.

	
Original Image	**Color corrected and thresholded image**

Extension-2)

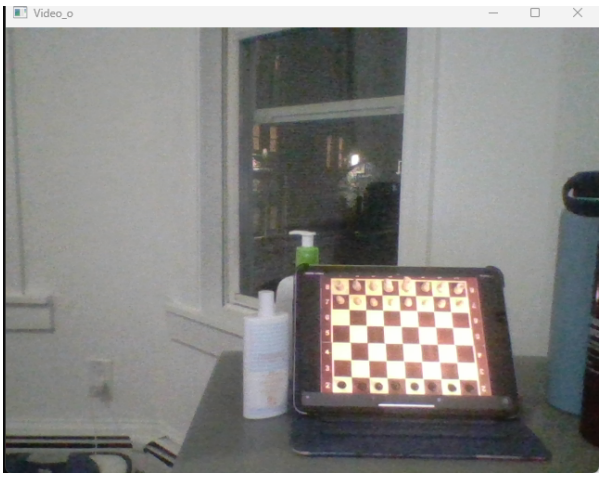
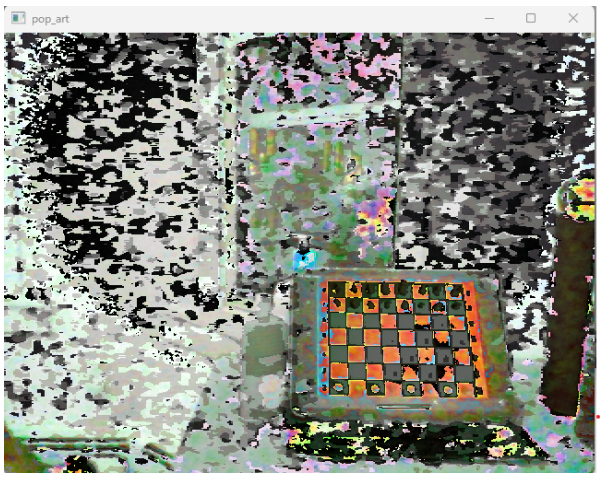
Inspired by a pop art filter, I have tried this filter. Explaining about the implementation

This program's function applies the pop art effect to an image. Src and res are the two input parameters required. The function will return the processed image as res, and the original image as src.

The function first changes the input image's src color space from BGR to HSV. The cvtColor function is used to do this. The function then manipulates the HSV values for each pixel in the image by looping through all of the pixels in the image. To do this, distinct transformations are first applied to each range of hue values after which they are divided into ranges. As an example the function multiplies the hue and saturation values by two and adds 50 to the value. if the hue value is between 0 and 30.The function then applies cvtColor to convert the image back to BGR color space before returning the edited image. Similarly multiple such conditions were checked and finally the output was converted back to BGR format.

The expected output was a bit different but it turned out to be a new filter as below.

By pressing ‘w’ after running the code, the pop art filter output is displayed.

	
Original Image	**Pop-art effect image**

Reflection of what you learned:

Was able to get acquainted with data-types, basic C++ syntaxes, pointer concepts etc.

Acknowledgement of any materials or people you consulted for the assignment:

- For basic Understanding of the pop art filter the following resource was used
https://www.analytics-link.com/post/2019/07/11/creating-pop-art-using-opencv-and-python
- For understanding some concepts i have used Open ai's chatGPT
- Discussed some coding issues with course-mate Srinivas Peri.