CSS5330 Pattern Recognition and Computer Vision

Project 1: Real time filtering

Francis Jacob Kalliath

1. A short description of the overall project

In this project I learned the basic functionality of OpenCV in C++ language which includes opening, capturing, manipulating, and writing to an image, and implemented various filters. This project had a wide implementation of different kinds of filters on live images, and all of them were implemented from scratch by looping through every pixel using a nested for loop. Some filters that I implemented as a part of this project include blur filter, Gaussian filter, Sobel filter X and Y, blur and quantization of an image, cartoonization of an image, and implementing a sparkle over negative images all of which are sub-parts of the project. Saving a particular frame and recording a particular section of the video were also implemented in this project.

2. Images collected from the output of the filters

TASKS:

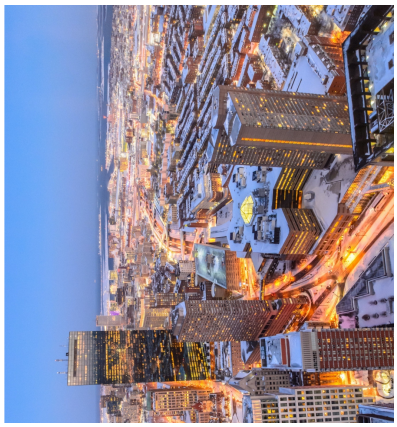
Task I

In this task I took the saved image from the project 1 folder and displayed it in a frame. If the alphabet ‘q’ was detected then the frame will be closed.

Additional functionality:

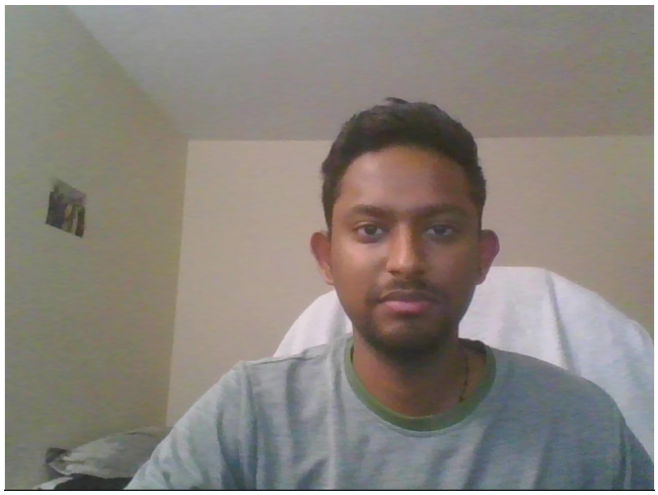
when the alphabet ‘r’ is clicked then the image is rotated by 90° clockwise direction

- When the alphabet ‘a’ is clicked, the image is located in counterclockwise direction 90°



- When the alphabet ‘o’ is clicked then the image is resized

Task 2



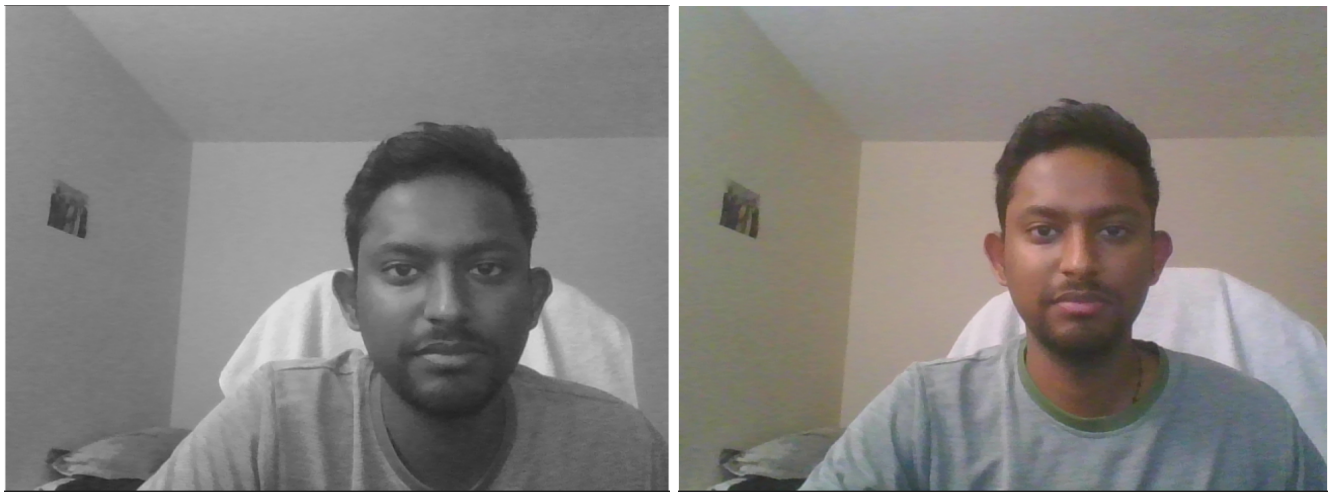
In this task, I collected frames in the loop to capture video and when the alphabet ‘q’ was clicked The video frame stopped and the tab was closed. When the alphabet s is clicked then the frame is saved in the current directory.

The OpenCV cvtColor() function uses the below formula and substitutes the RGB values in the formula to obtain the Grey scaled corresponding color.

L = 0.299*R* + *0.587*G + 0.114*B

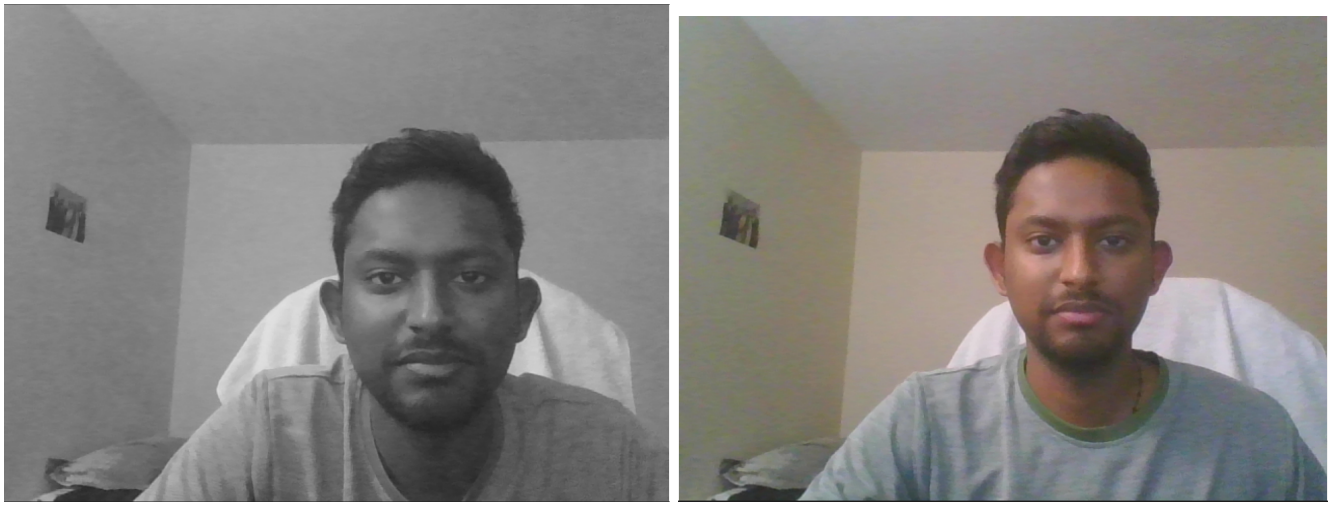
Where L is the luminosity value, R is the red channel value, G is the green channel value, and B is the blue channel value.

Task 3



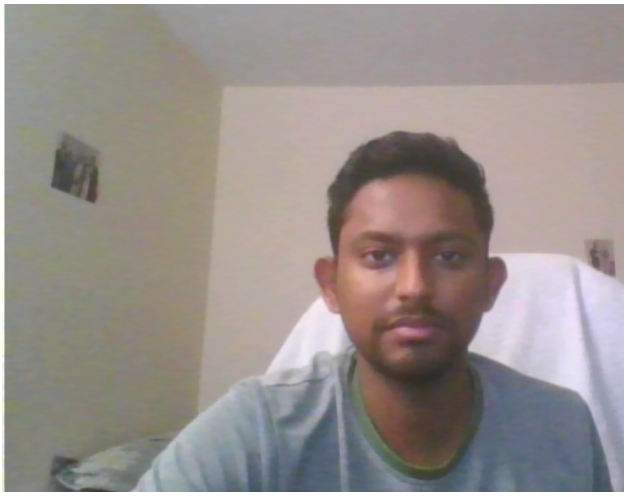
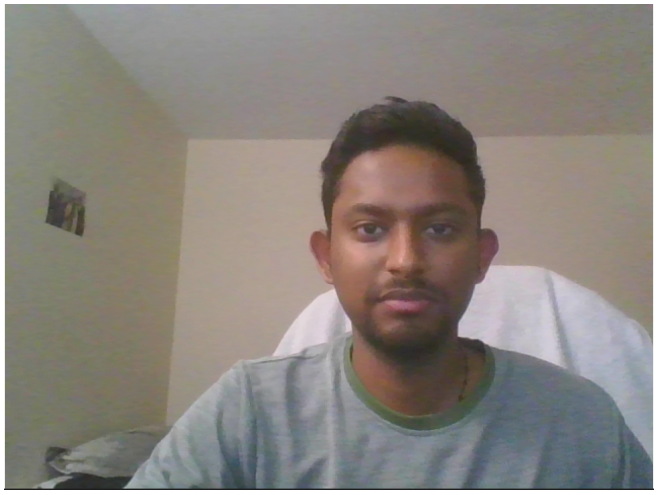
In this task when the alphabet ‘g’ is pressed the video turns into grayscale video.

Task 4



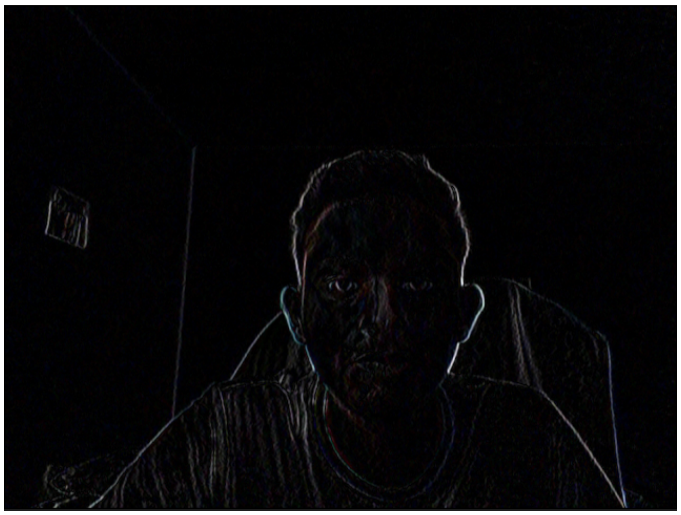
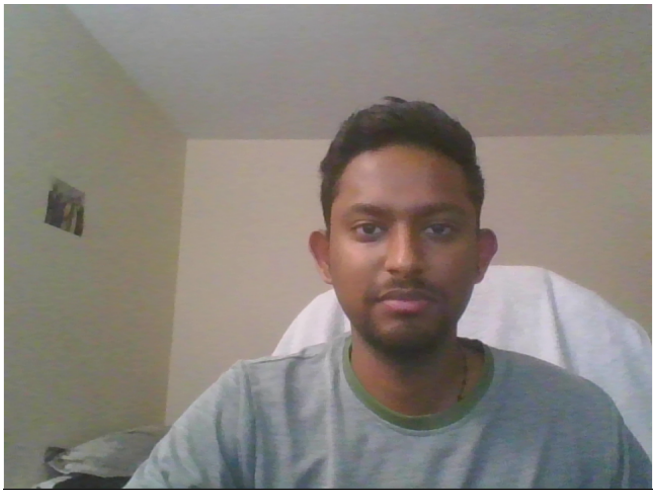
In this task when the alphabet ‘h’ is pressed the video stream turns into a different greyscale. Here the green channel color is copied into the other two color channel

Task 5

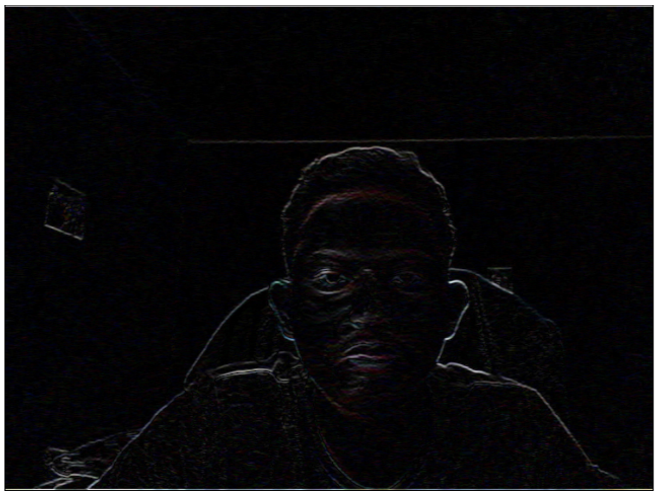
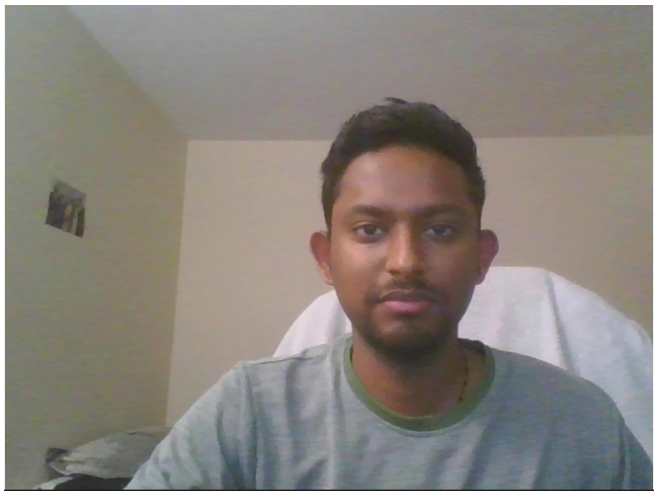



In this task when the alphabet ‘b’ is pressed the image gets blurred as a Gaussian filter is applied as separable is 1x5 filters

Task 6

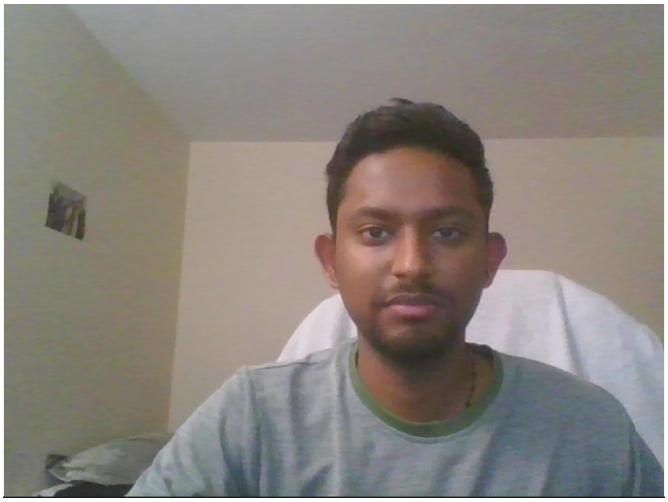



In this step of the task when the alphabet ‘x’ is pressed the sobel filter X is applied and only the vertical lines are visible

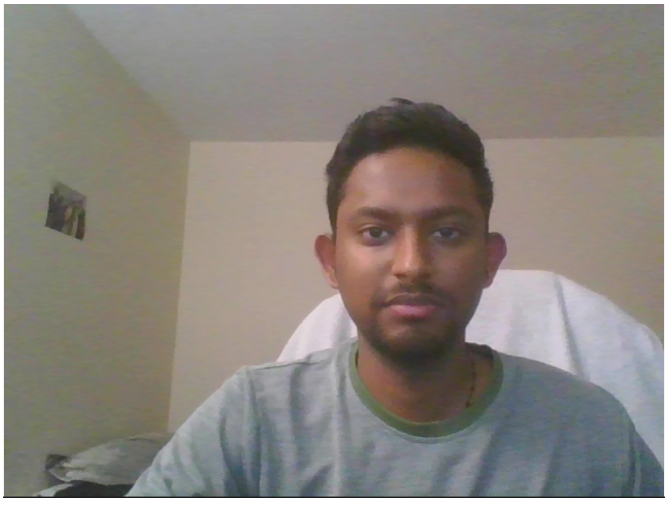
In this step of the task when the alphabet ‘y’ is pressed the sobel filter Y is applied and only the horizontal lines are visible.

Task 7

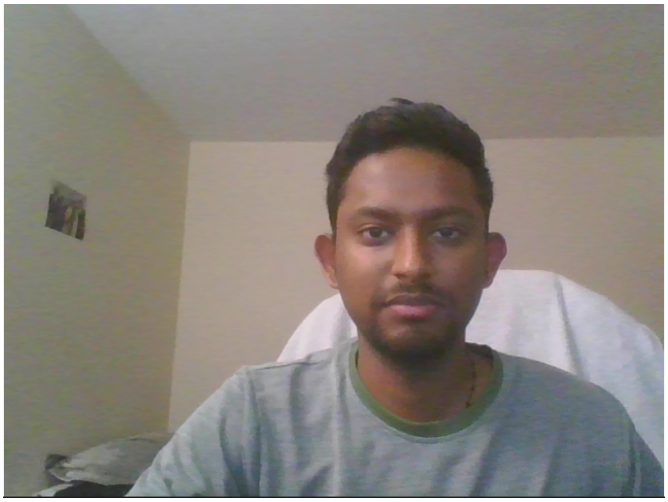
In this task when the alphabet ‘m’ is pressed the gradient magnitude image is displayed on the frame

Task 8

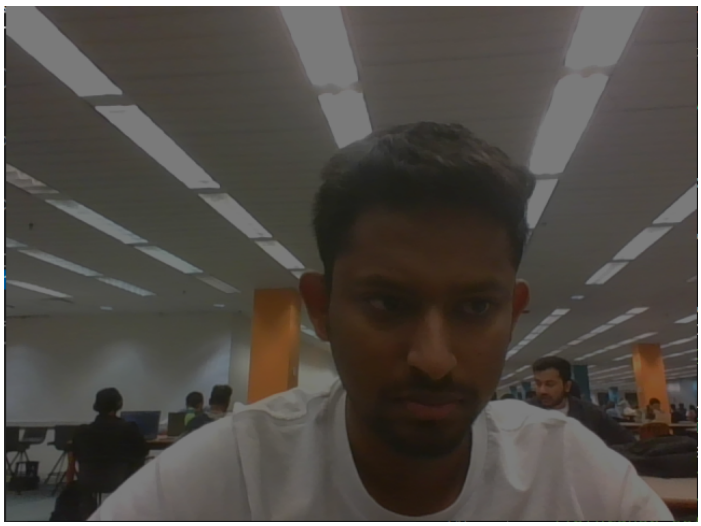
In this task when the alphabet ‘l’ is pressed the an image that is blur and quantized color image is obtained

Task 9

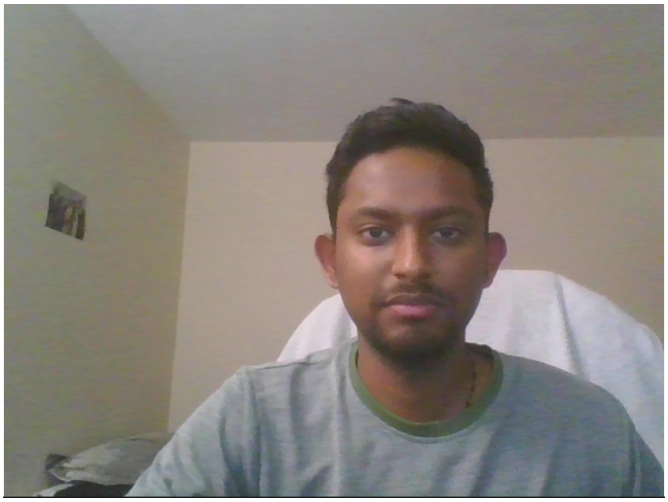
In this task when the alphabet ‘c’ is pressed the image gets cartoonized by detecting edges and applying blur/quantize filters to it.

Task 10




In this task when the alphabet ‘w’ is pressed a prompt in the terminal asks for the alpha and beta value. Once you key in alpha press enter then key in the beta value then press enter then the output frame displays the image with the entered brightness and contrast.

3. Description and example image for extension

In this task when the alphabet a is pressed the edges are detected a sparkle is added to it and then a negative image is created and then color map is generated for the image and then the output is displayed.
In this task initially I am converting the frames to grayscale. Then I am using the openCV function Canny() to obtain the edges in the frame. After this I am looping through all the pixels and assigning the strong edges with white colors to give the image a sparkling effect at the sharp edges. Then I am converting the image to a negative image by accessing all the pixels. At the end I am applying applyColorMap() function to give a colorful shade to the output image.
The output detects edges very accurately and also detects the surface changes or textures very precisely. The additional color mag makes the visualization lot more colorful

4. Reflection of the learnings

In this project I got hands-on exposure to the C++ coding in the OpenCV package. Some of the most important concepts for computer vision like opening, capturing, manipulating and writing into images were all an important learning in this project.
Some of the most important learning from this project are:

1. Capturing an image from a live video feed and doing some operation on it when a key is entered
2. Applying the concept of convolution from scratch and applying filters on the images to get required output helps visualize how convolution works.
3. Observing the effect of filters on the image and how applying different filters together helps up obtain different kinds of output image
4. This project not only gives me a hands on practices of applying fillers but also gives me an idea of image processing and saving the content on the device

5. Acknowledgement of the material

- GeeksforGeeks
- W3 School
- HackerRank
- Quora
- Stackoverflow
- Git