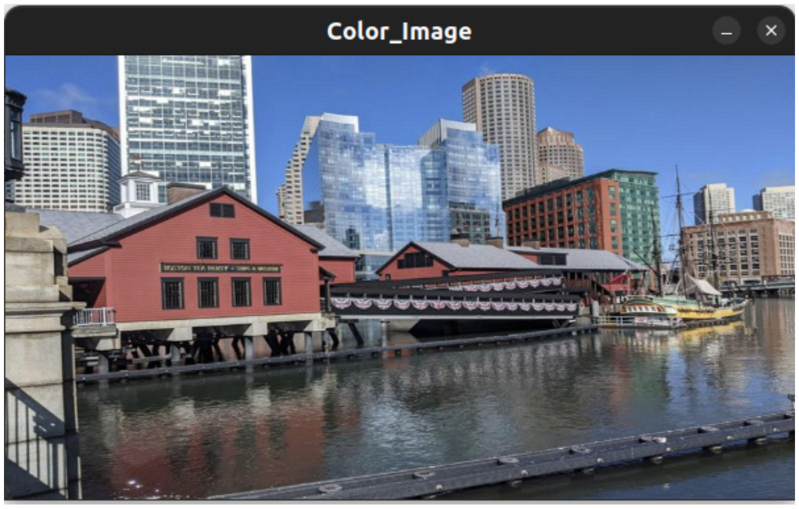
Project 1: Real-time filtering

Project 1: Real-time filtering

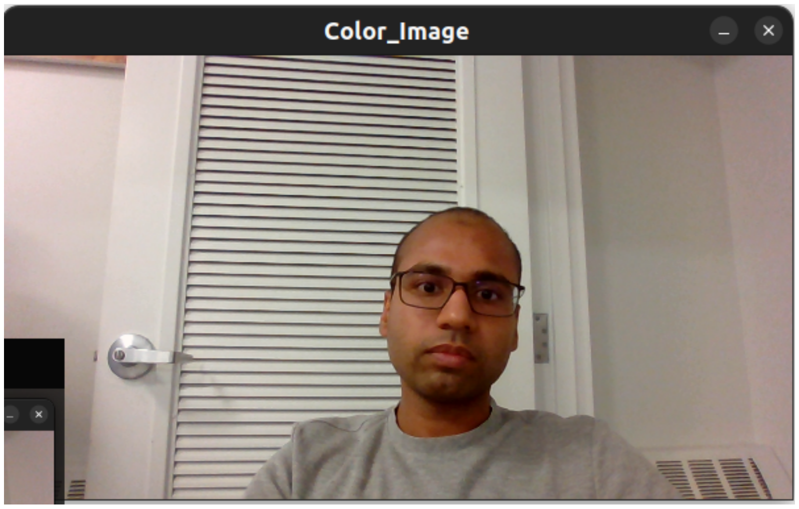
Description: The project consists of implementing many filters from scratch using scratch using C++ and OpenCV. The filters include greyscale conversion using weighted average method, 5x5 Gaussian filter as separable 1x5 filters, 3x3 Sobel X and 3x3 Sobel Y filter as separable 1x3 filters, generates a gradient magnitude image from the X and Y Sobel images, blurs and quantizes a color image, live video cartoonization, and image negatiion. As part of extension, I have implemented changing brightness, image sharpening, drawing a sketch, and captioning. I wrote vidDisplay.cpp, filter.cpp, and filter.h files to apply filters on Live Video. imgDisplay.cpp file reads a saved image using command line argument and displays it on a window. If it is not passed, live video will be shown.

1) Read an image from a file and display it.

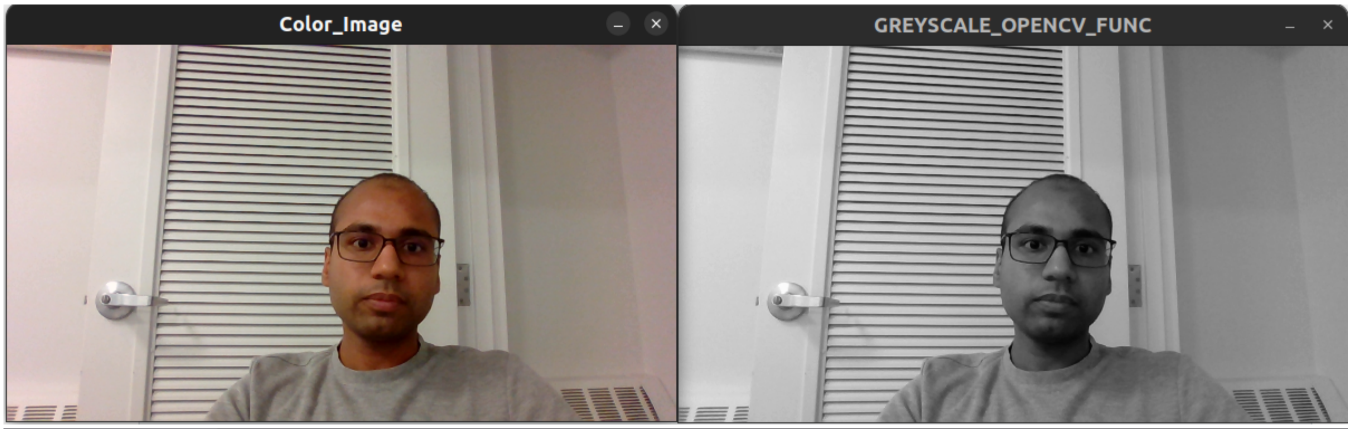
Below is the still image used for the extension. Different filters are applied to the live video as well as still image.

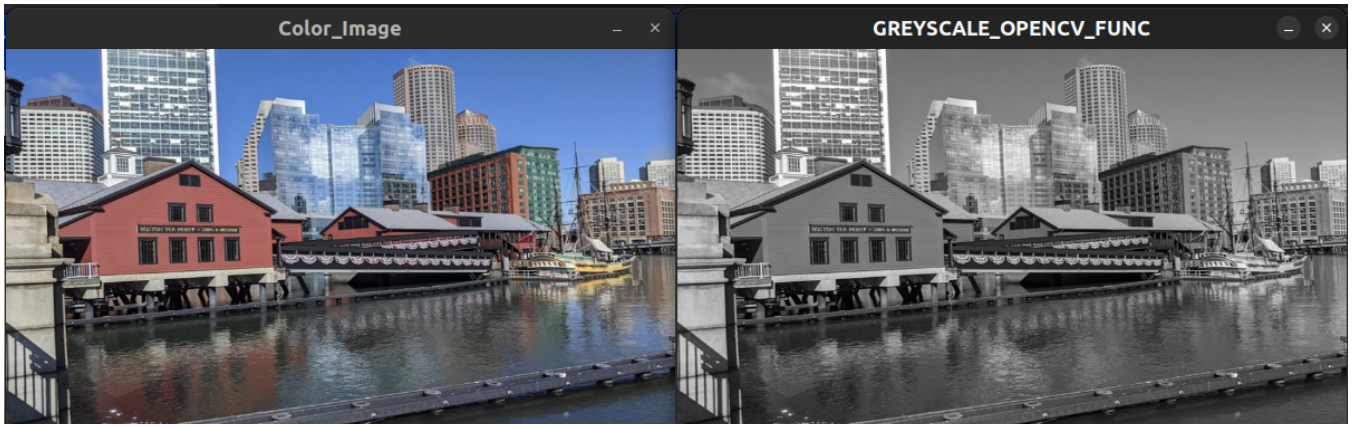


2) Display live video

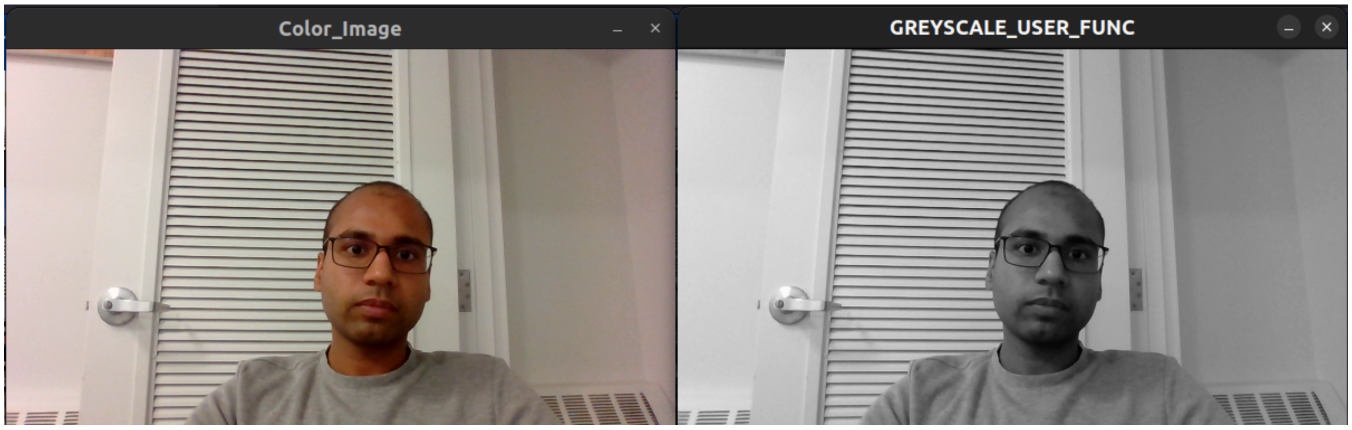


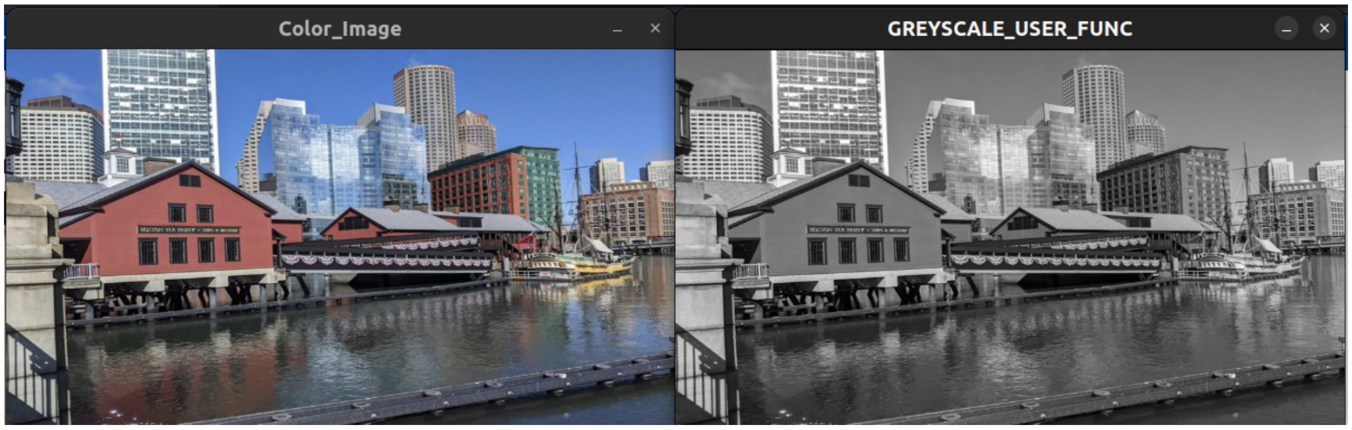
3) Display greyscale live video



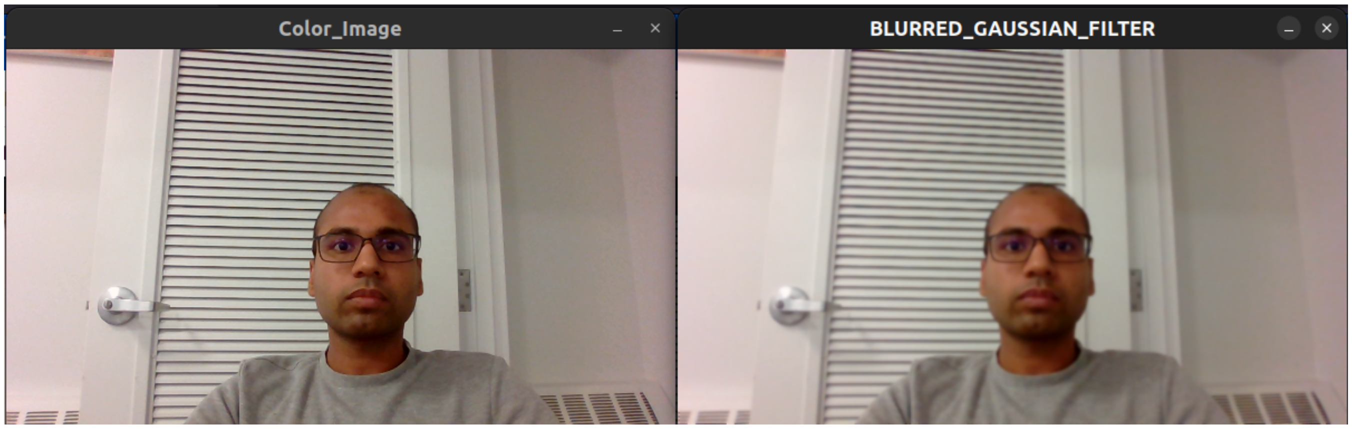


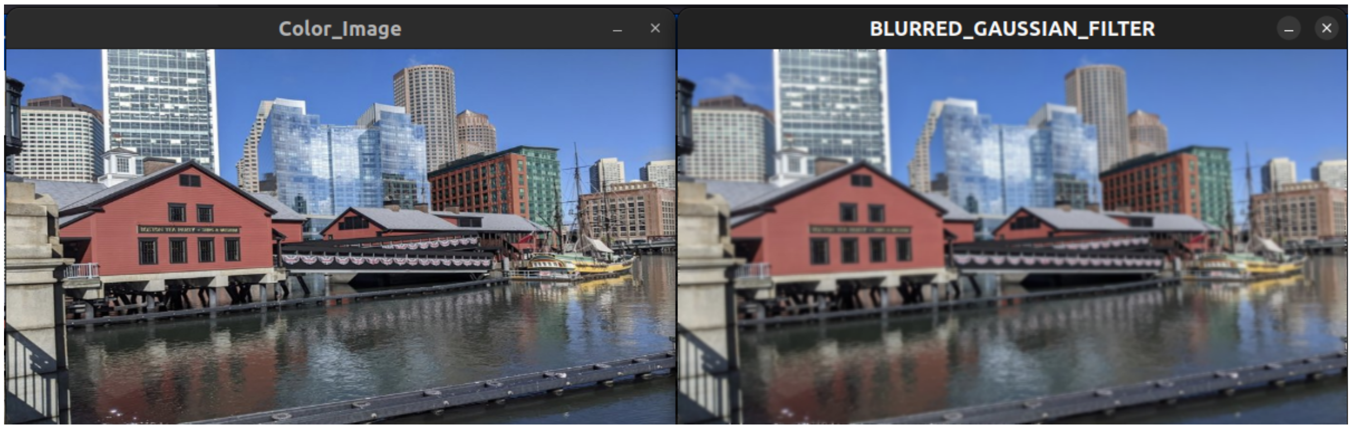
4) Display alternative greyscale live video



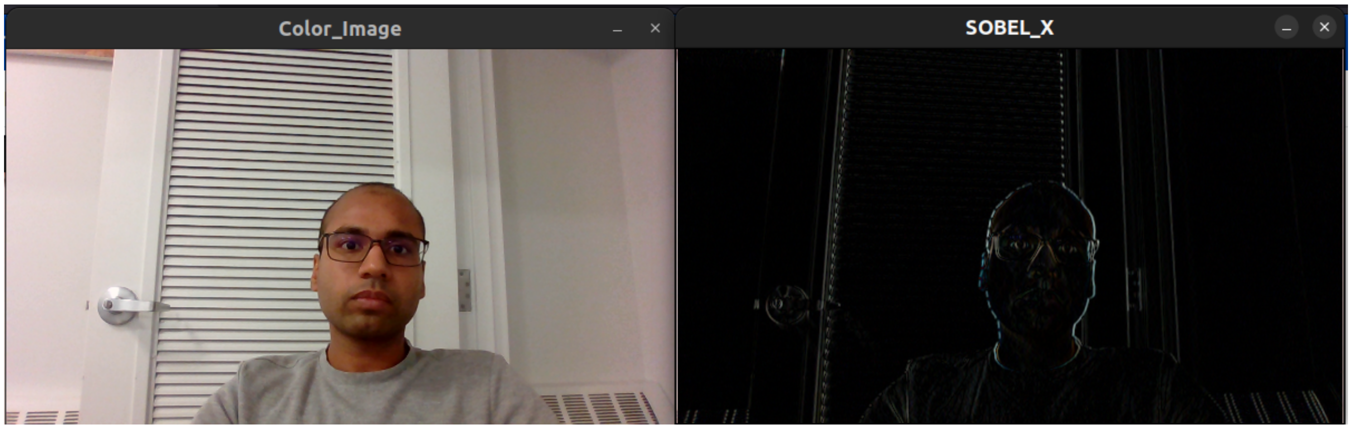


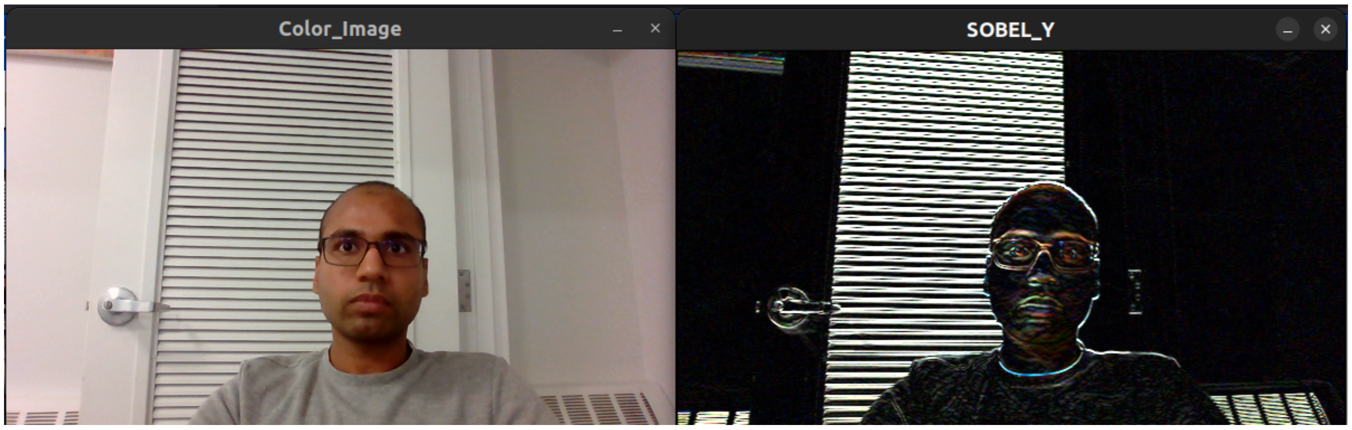
5) Implement a 5x5 Gaussian filter as separable 1x5 filters

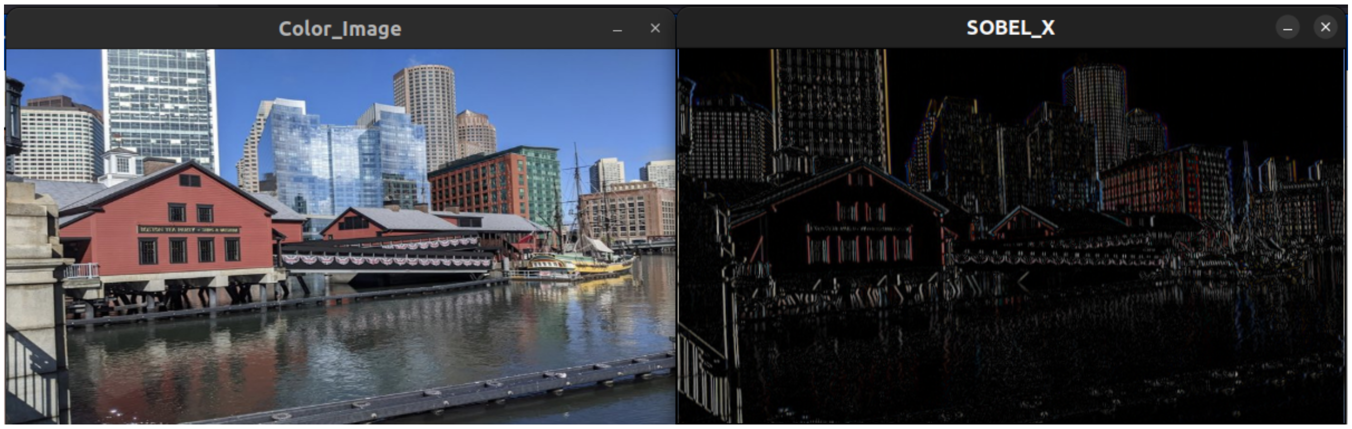


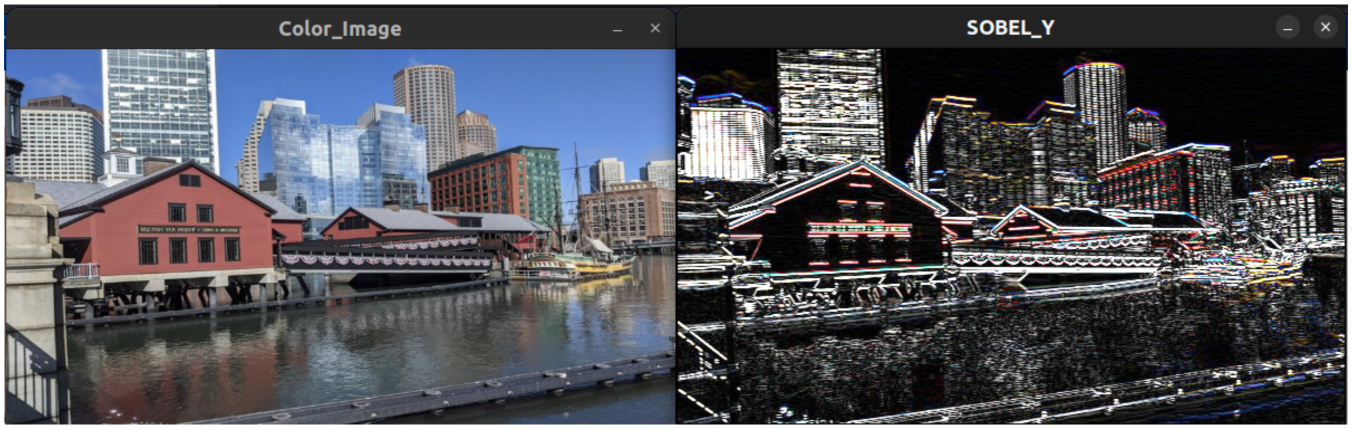


6) Implement a 3x3 Sobel X and 3x3 Sobel Y filter as separable 1x3 filters

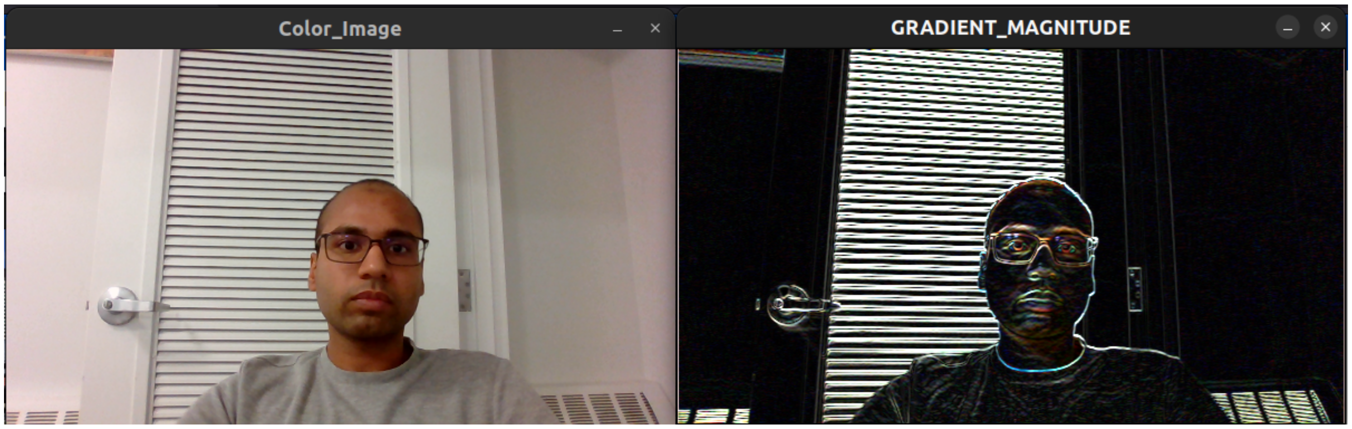


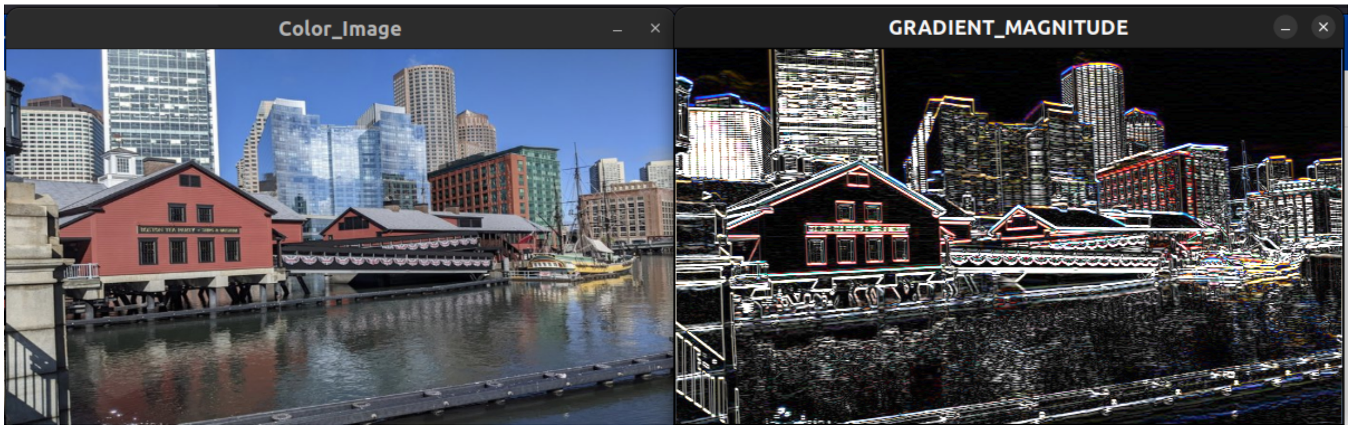




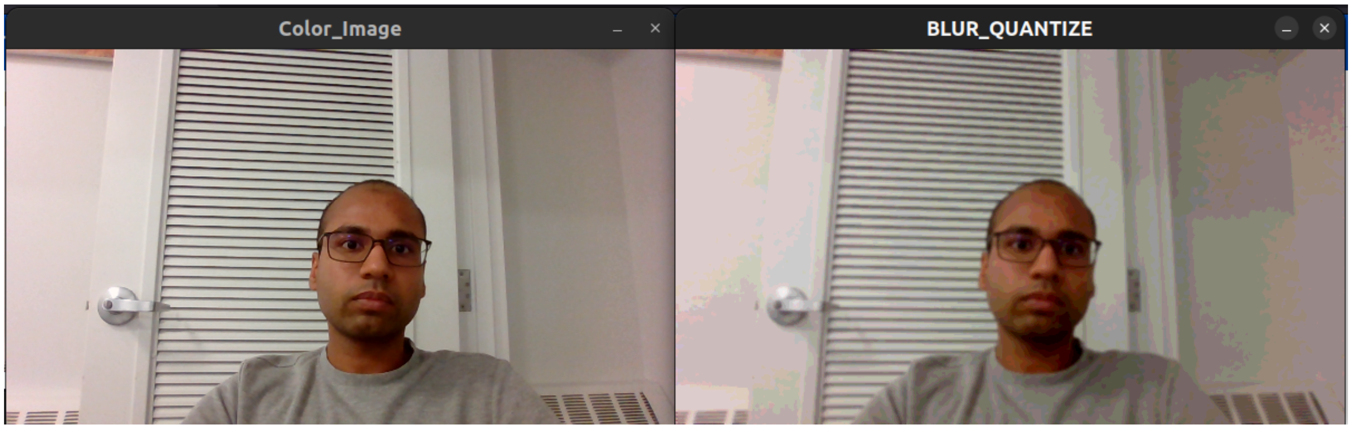


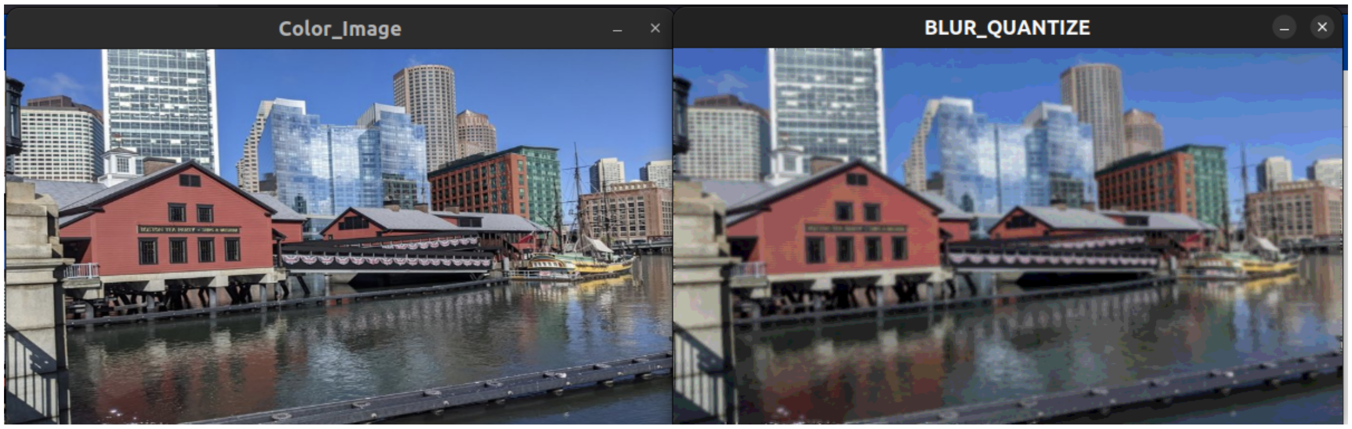
7) Implement a function that generates a gradient magnitude image from the X and Y Sobel images



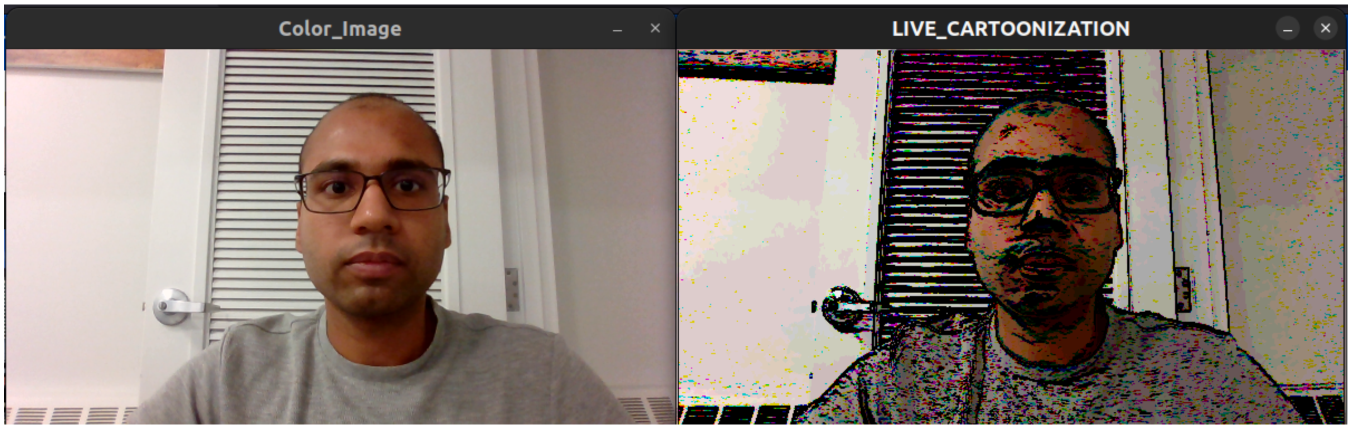


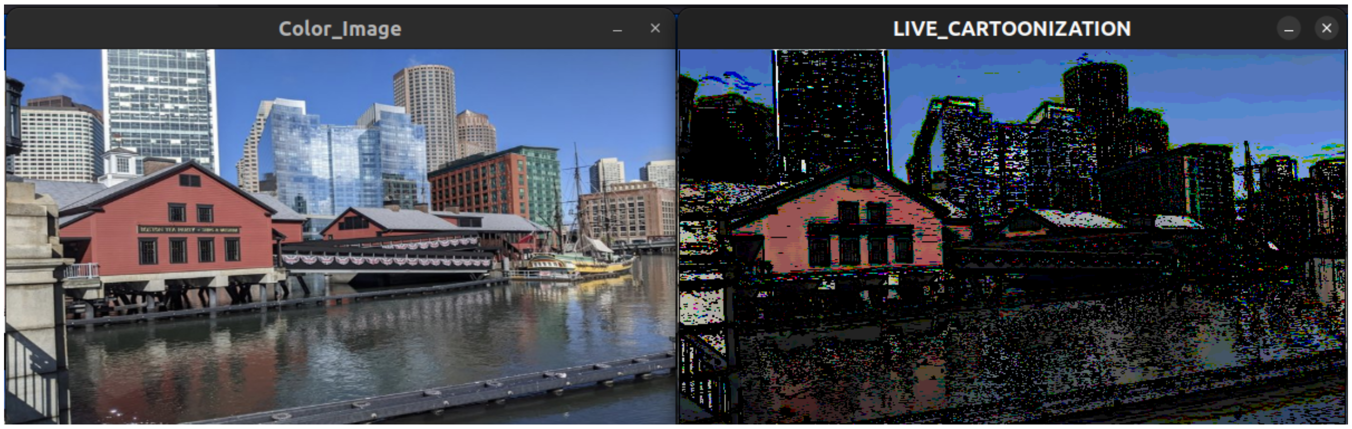
8) Implement a function that blurs and quantizes a color image



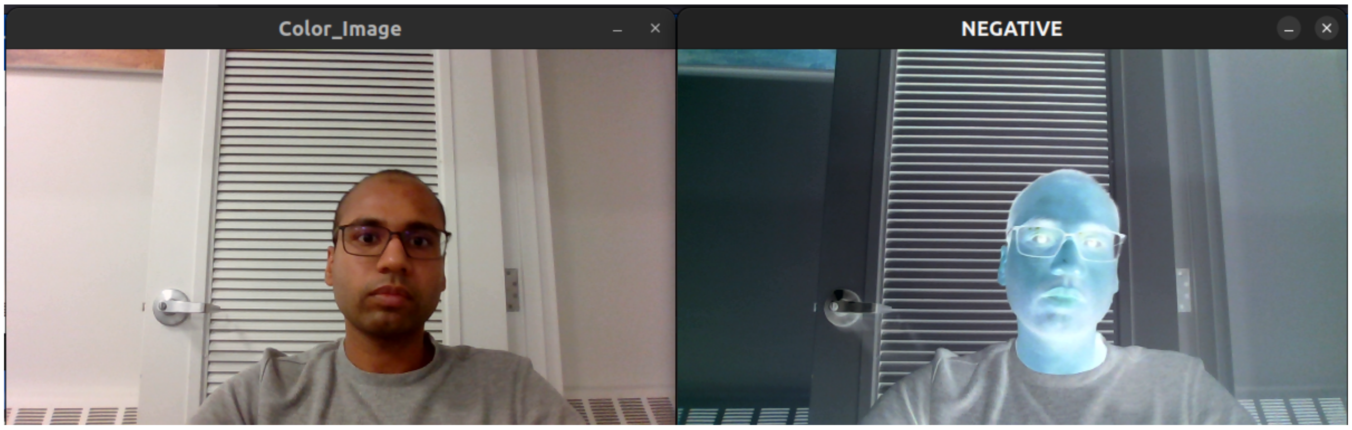


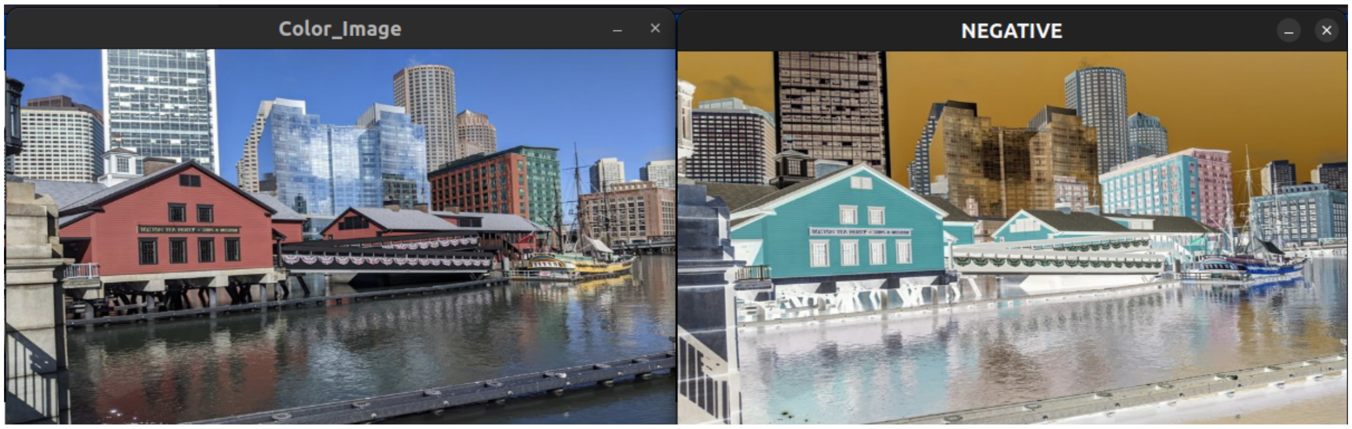
9) Implement a live video cartoonization function using the gradient magnitude and blur/quantize filters



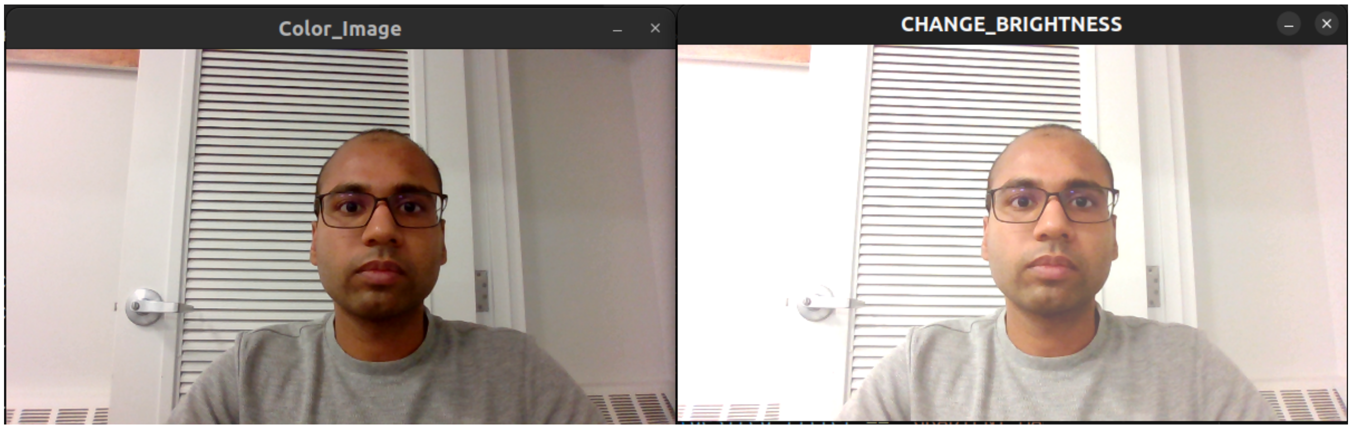


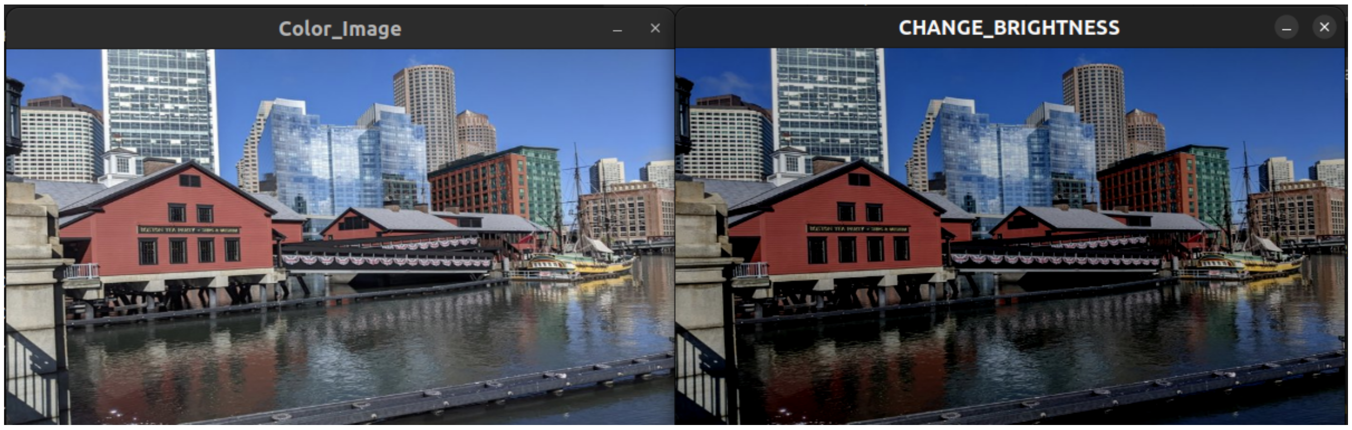
10) Pick another effect to implement on your video: Making the image a negative of itself.



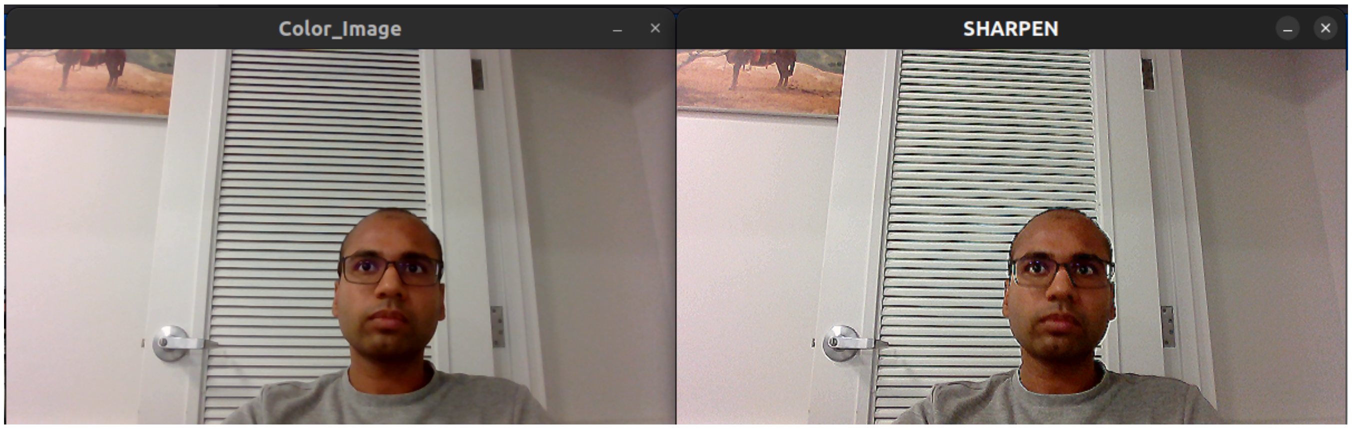


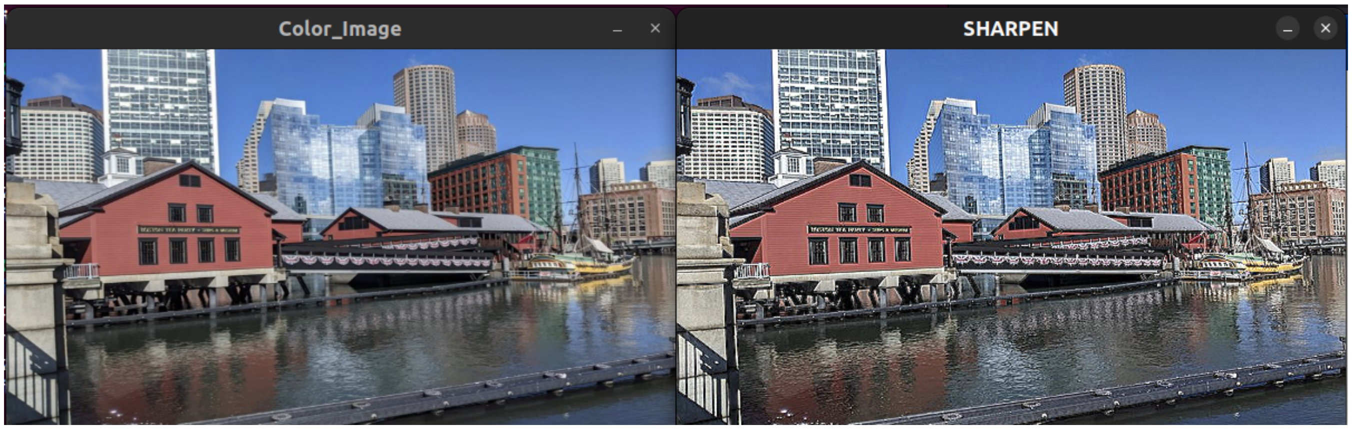
EXTENSION 1) Increase/ decrease brightness when the user presses UP/ DOWN arrow key respectively on the keyboard



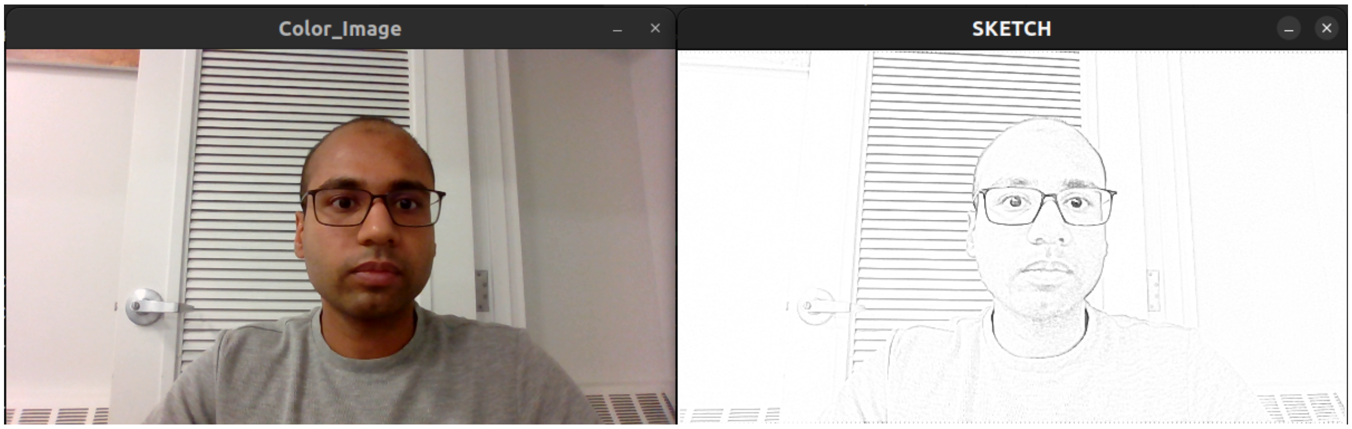


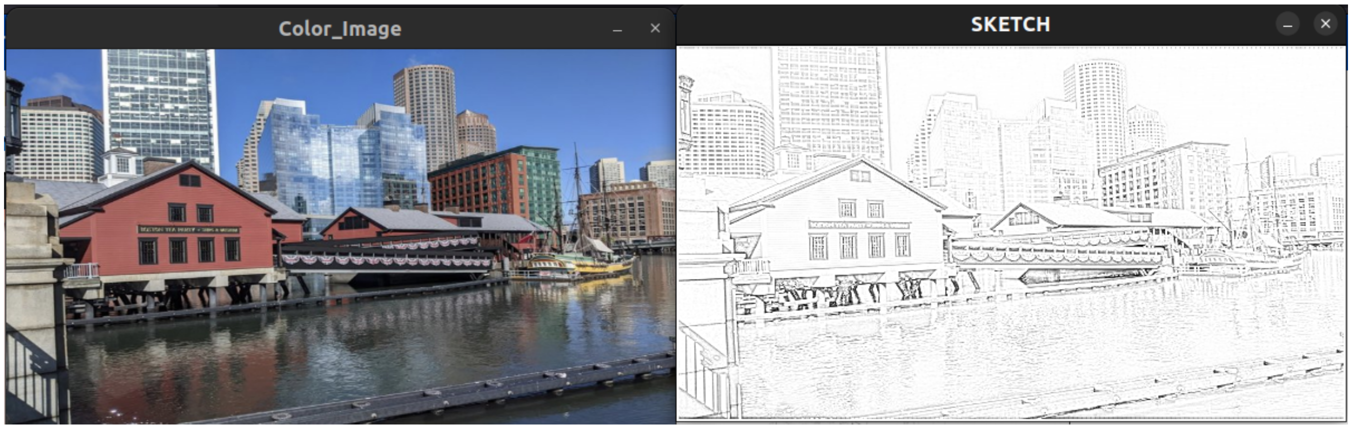
EXTENSION 2) Sharpen the image when the user presses the key 1 on the keyboard



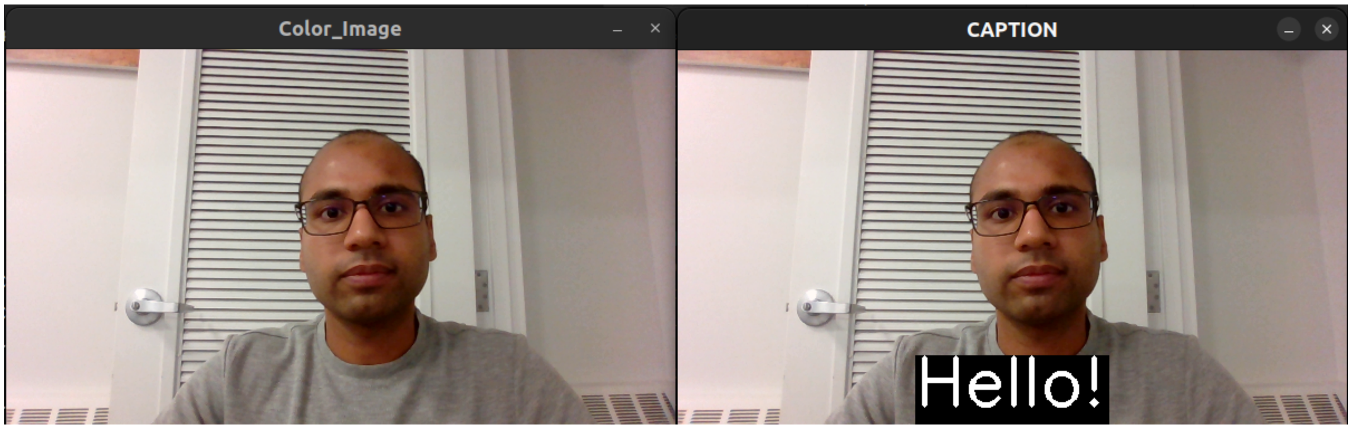


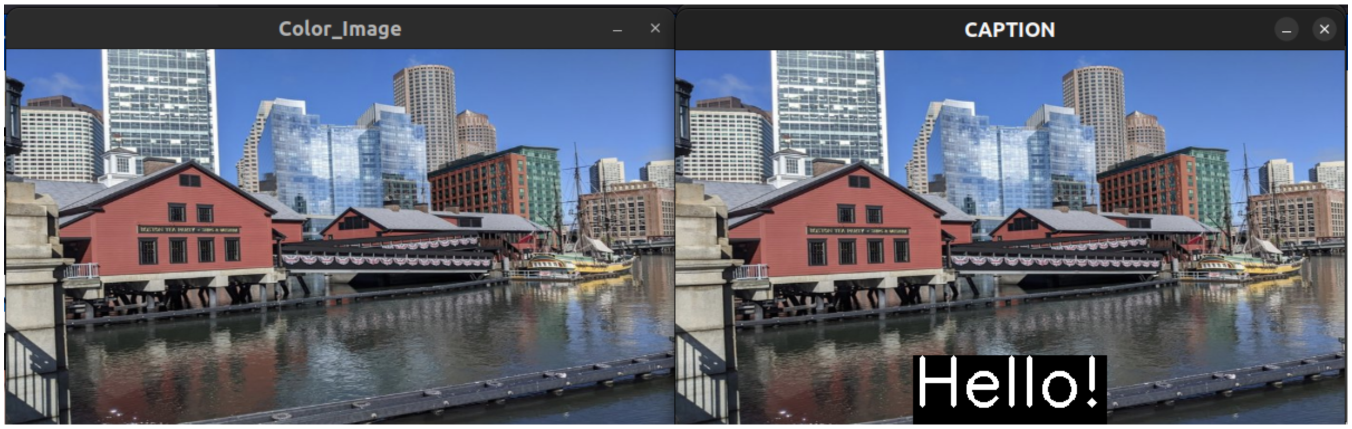
EXTENSION 3) Make a pencil sketch of the image when the user presses the key 2 on the keyboard





EXTENSION 4) Get input from the user and add a caption to the image when the user presses the key 3 on the keyboard





EXTENSION 5) Saving any filtered video when the user presses the key 4 on the keyboard and stop when q is pressed

The binary generates three files: s*aved_video_original.avi, saved_video_color.avi* ,and *saved_video_grey.avi* . Video gets saved to only two files at a time depending on whether the filter's output is color or greyscale. Original video gets saved into s*aved_video_original.avi* always when an effect is into action and 4 is pressed.

Reflection :

I learned different methods to extract data from images, implementing various filters from scratch by reading each pixel and performing mathematical operations on it, and also eliminating high-frequency noise from images. I strongly believe that this project made me ready for future computer vision and deep learning projects as the hidden layers mainly consist of filtered output.

References:

I have seen some videos on C++ and OpenCV documentation for function reference and their sample code for an idea about the usage of functions. I used Stack overflow to resolve bugs in the code, and Wikipedia pages to see the process to perform filters.