CS5330: PROJECT 1

VIDEO SPECIAL EFFECTS

Team Members:
Name: Ravi Shankar Sankara Narayanan (NUID: 001568628)
Name: Vishaq Jayakumar (NUID: 002737793)

Short Description:

This project is a real-time video processing application that uses OpenCV. It captures video from the integrated webcam in our laptops, applied various filters tot eh video frames and displays the processed video in real-time.

The filters include grayscale, sepia, blur, Sobel X and Y, magnitude, quantize, emboss, negative, color pop, cartoon, and face detection. The filter to be applied is determined by the key pressed by the user.

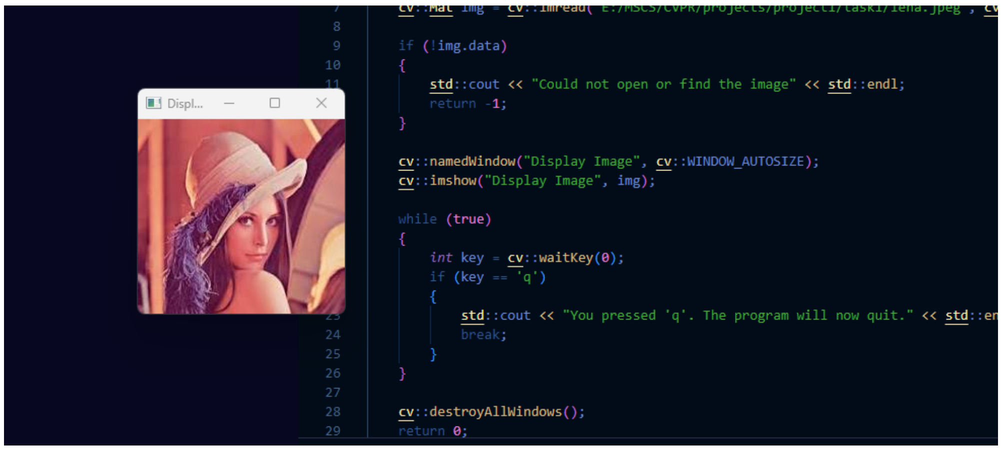
For the extensions: We have added a filter to create a sketch effect over the frame. We have provided features like real-time increase and decrease of brightness and contrast. And finally, the program allows the user to record a video of the webcam feed from the program. While recording the video, the user can implement and play with the various filters we have provided.

We have also added appropriate text messages when each filter is applied and when video is recorded to make it more intuitive for the user.

Task Output:

1. **Task 1: Display an image**

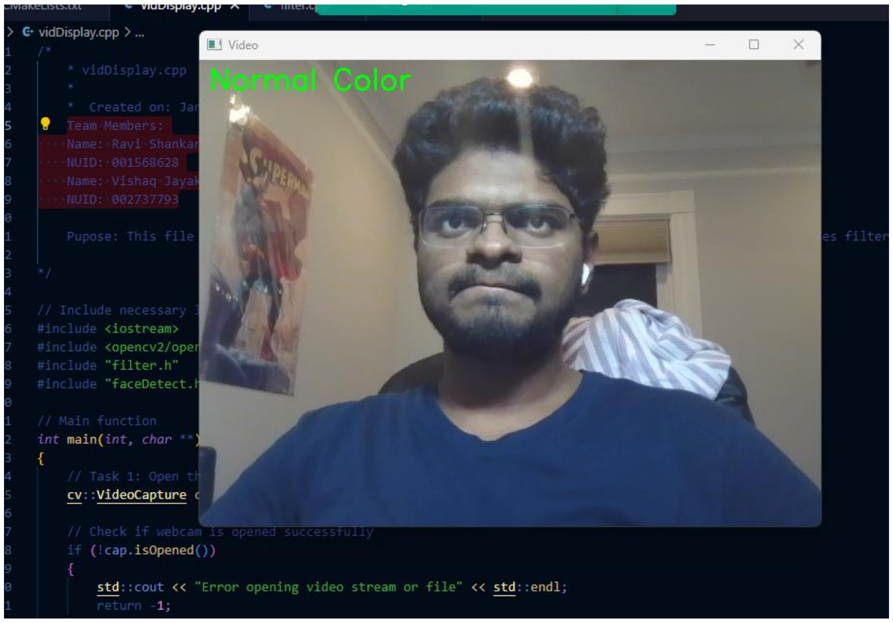
Our code can read an image stored on our computer and display it. A screenshot of our program displaying an image is attached below.



```
cv::Mat img = cv::imread("E:/MSCS/CVPR/projects/project1/task1/lena.jpeg", cv

if (!img.data)
{
    std::cout << "Could not open or find the image" << std::endl;
    return -1;
}

cv::namedWindow("Display Image", cv::WINDOW_AUTOSIZE);
cv::imshow("Display Image", img);

while (true)
{
    int key = cv::waitKey(0);
    if (key == 'q')
    {
        std::cout << "You pressed 'q'. The program will now quit." << std::en
        break;
    }
}

cv::destroyAllWindows();
return 0;
```

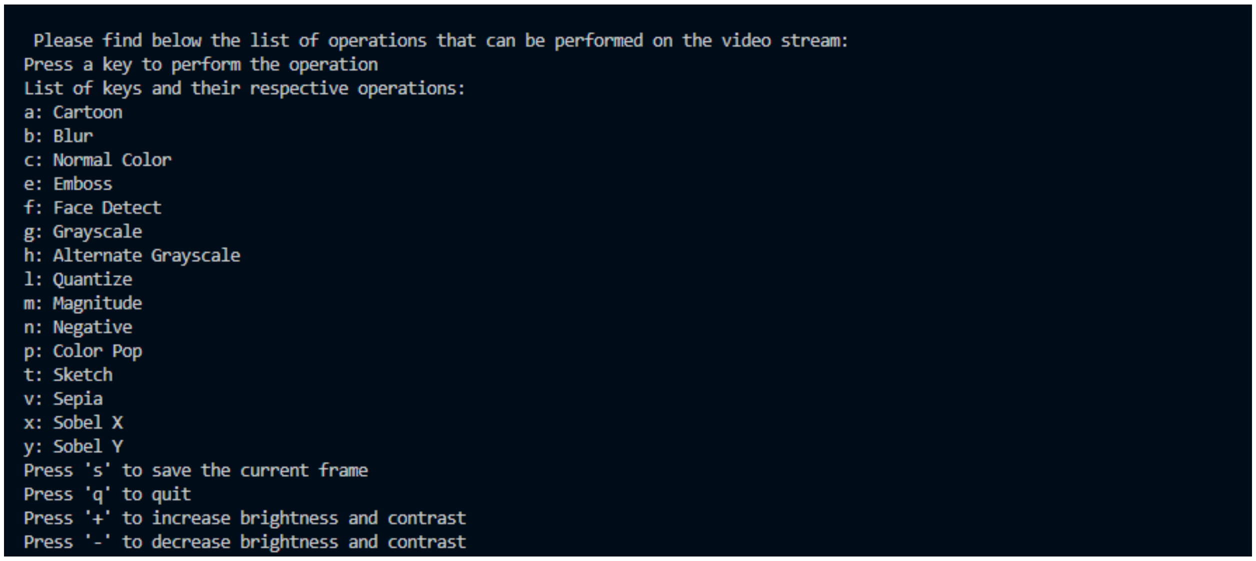
2. Task 2: Display Live video

Our code can connect with the webcam and display the data. A screenshot of our program displays the webcam feed is attached below. Now using our program, we can apply multiple filters.



```
/*
 * vidDisplay.cpp
 *
 *  Created on: Jar
 Team Members:
    Name: Ravi Shankar
    NUID: 001568628
    Name: Vishaq Jayak
    NUID: 002737793

 Pupose: This file                                es filter

*/

// Include necessary l
#include <iostream>
#include <opencv2/oper
#include "filter.h"
#include "faceDetect.h

// Main function
int main(int, char **)
{
    // Task 1: Open th
    cv::VideoCapture c

    // Check if webcam is opened successfully
    if (!cap.isOpened())
    {
        std::cout << "Error opening video stream or file" << std::endl;
        return -1;
```

When our program runs, it will display the details of the keys and their respective functions in the terminal. The screenshot is attached below:

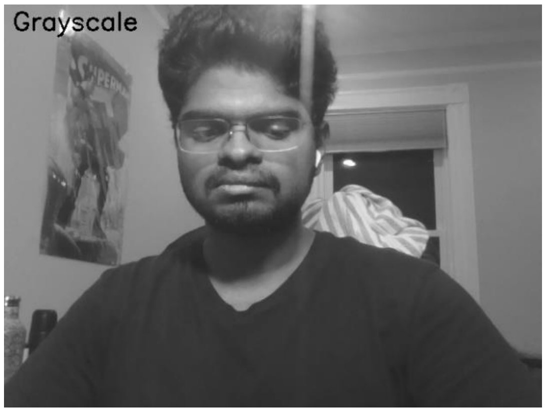


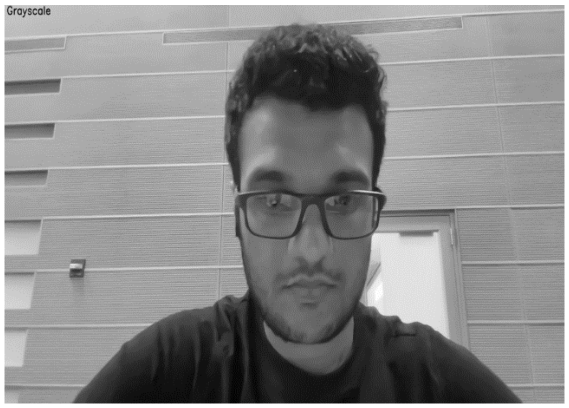
```
 Please find below the list of operations that can be performed on the video stream:
Press a key to perform the operation
List of keys and their respective operations:
a: Cartoon
b: Blur
c: Normal Color
e: Emboss
f: Face Detect
g: Grayscale
h: Alternate Grayscale
l: Quantize
m: Magnitude
n: Negative
p: Color Pop
t: Sketch
v: Sepia
x: Sobel X
y: Sobel Y
Press 's' to save the current frame
Press 'q' to quit
Press '+' to increase brightness and contrast
Press '-' to decrease brightness and contrast
```

When the user clicks on a key, the respective filter will be applied over the webcam feed in real-time.

3. **Task 3: Greyscale filer**

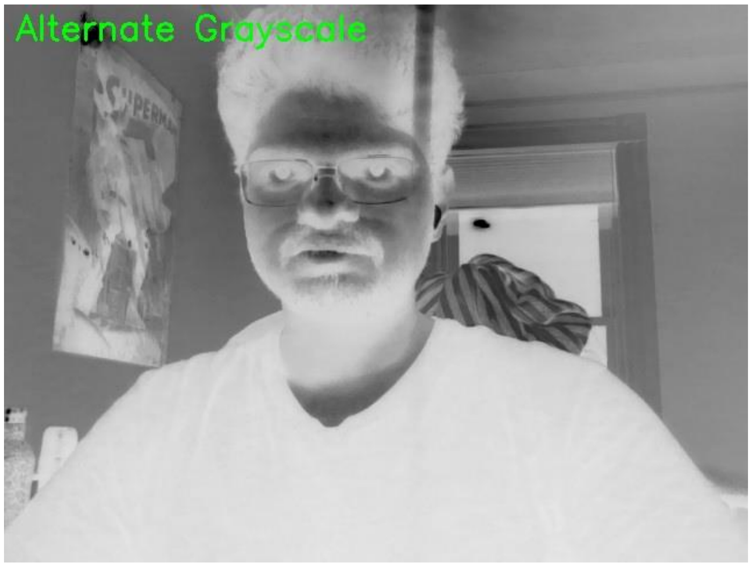
Our code can connect with the webcam and convert the live feed to greyscale and display the output. Example images are attached below.





4. **Task 4: Alternate Greyscale filer**

Our code can apply the alternate greyscale filter to the live feed and display the output. The output image is displayed below.

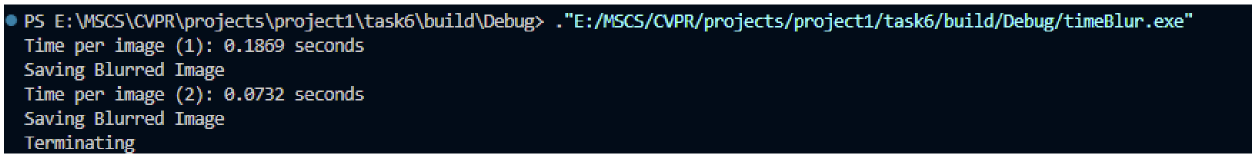


5. Task 5: Sepia filter

Our code can also apply the sepia filter successfully over the live feed. The output images are attached below.

6. Task 6: Blur filter

The time comparison of both blur5x5_1 and blur5x5_2 is attached below.



```
● PS E:\MSCS\CVPR\projects\project1\task6\build\Debug> ."E:/MSCS/CVPR/projects/project1/task6/build/Debug/timeBlur.exe"
  Time per image (1): 0.1869 seconds
  Saving Blurred Image
  Time per image (2): 0.0732 seconds
  Saving Blurred Image
  Terminating
```

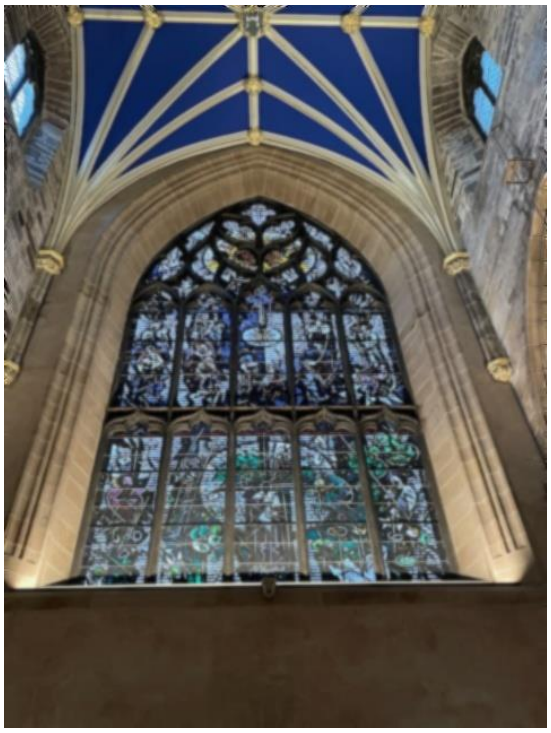
As expected, the blur5x5_2 function is faster than blur5x5_1 because it uses a separable filter, which reduces the computational complexity.

In blur5x5_1, the 5x5 kernel is applied to each pixel in the image, which involves 25 multiplications and additions for each pixel.

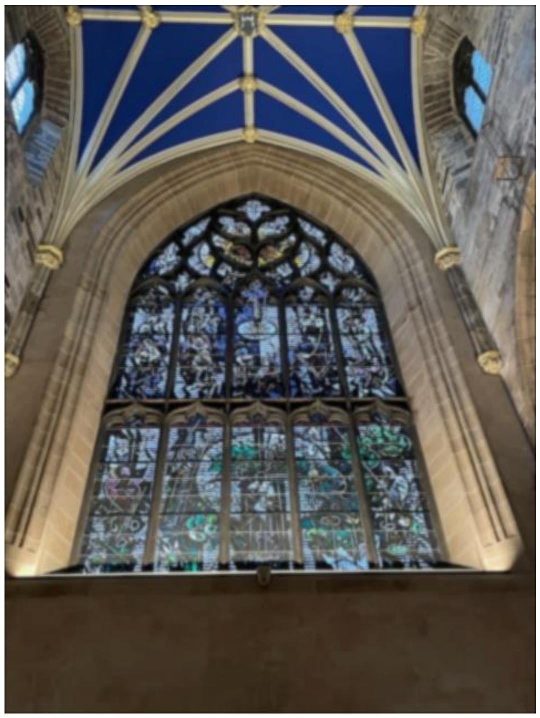
In blur5x5_2, the 5x5 kernel is separated into two 1x5 kernels. The first 1x5 kernel is applied horizontally, and then the second 1x5 kernel is applied vertically. This involves 5 multiplications and additions for each pixel, twice (once for the horizontal pass and once for the vertical pass), for a total of 10 operations per pixel.

So, blur5x5_2 is faster because it performs fewer operations per pixel. This is a common technique used in image processing to speed up convolution operations.

The blurred image from blur5x5_1 is attached below:



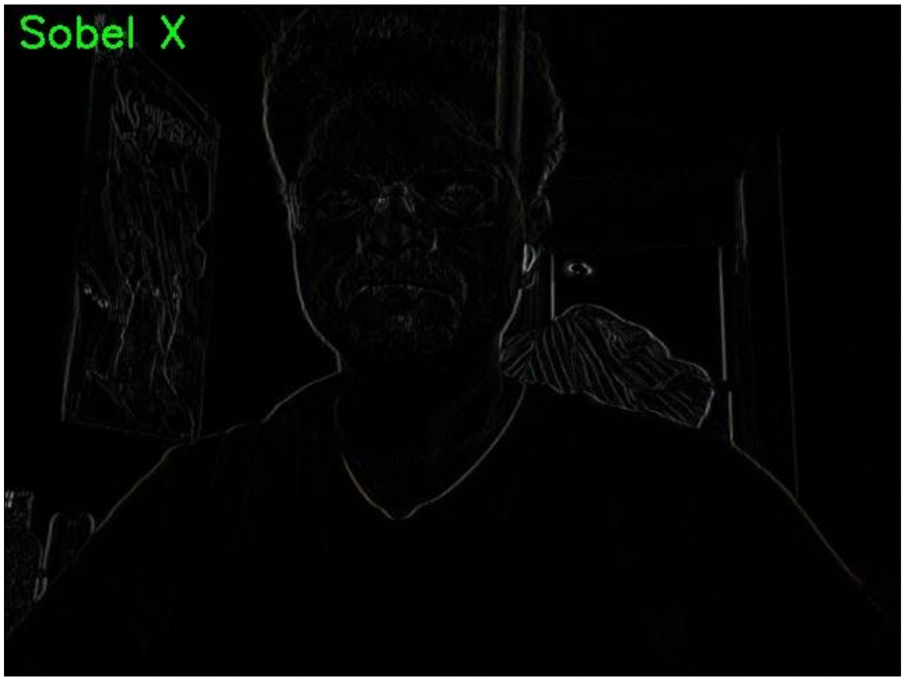
The blurred image from blur5x5_2 is attached below:

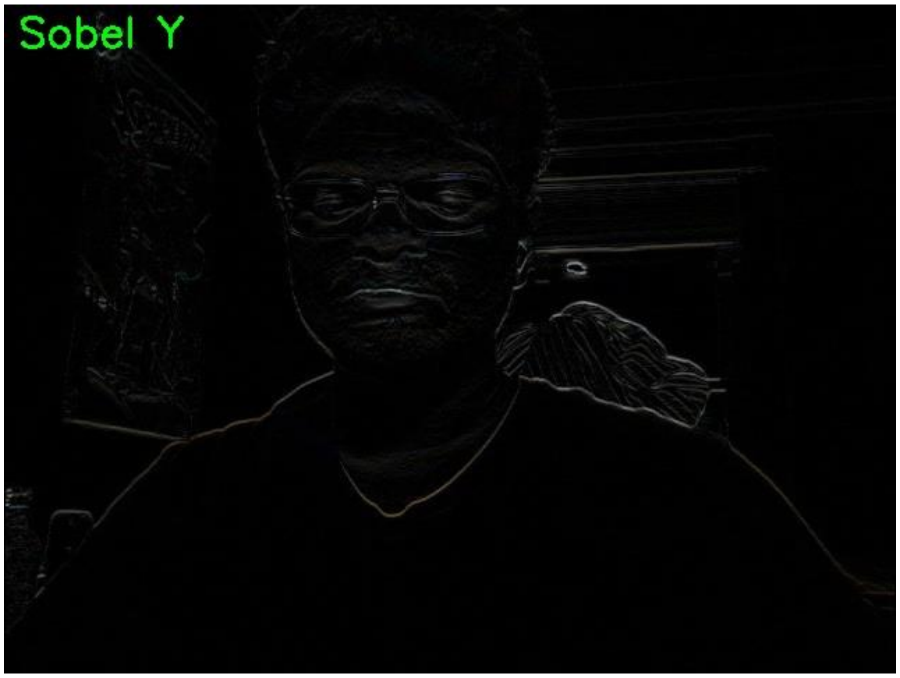


We applied the blur5x5_2 filter to vidDisplay program and the output is attached below. The filter has successfully blurred the live feed from webcam.

7. Sobel X and Y filter

Our code can successfully apply the Sobel X and Y filter in real-time over the webcam feed. The output images are attached below.

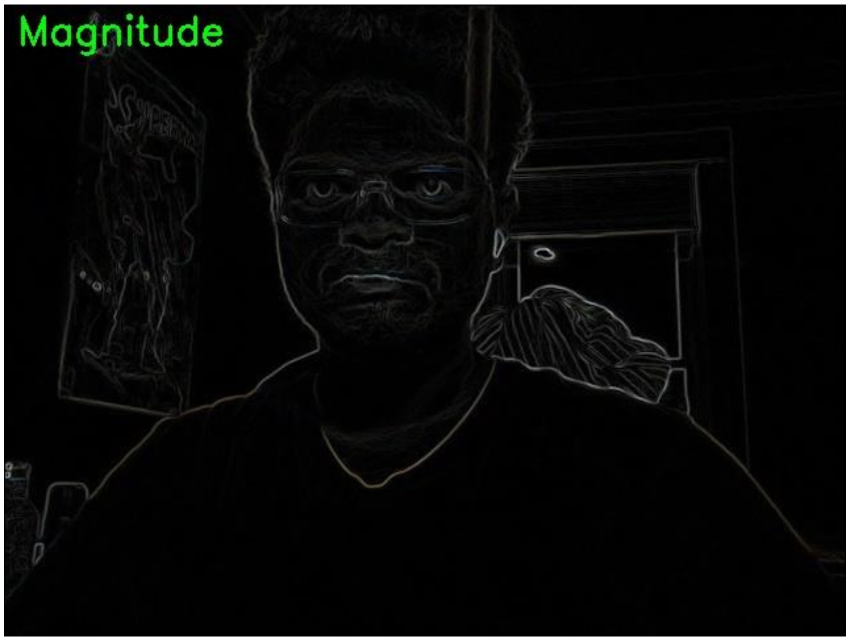




On close observation we can find that in Sobel X filter all vertical edges are displayed whereas for Sobel Y filter all horizontal edges are well detected.

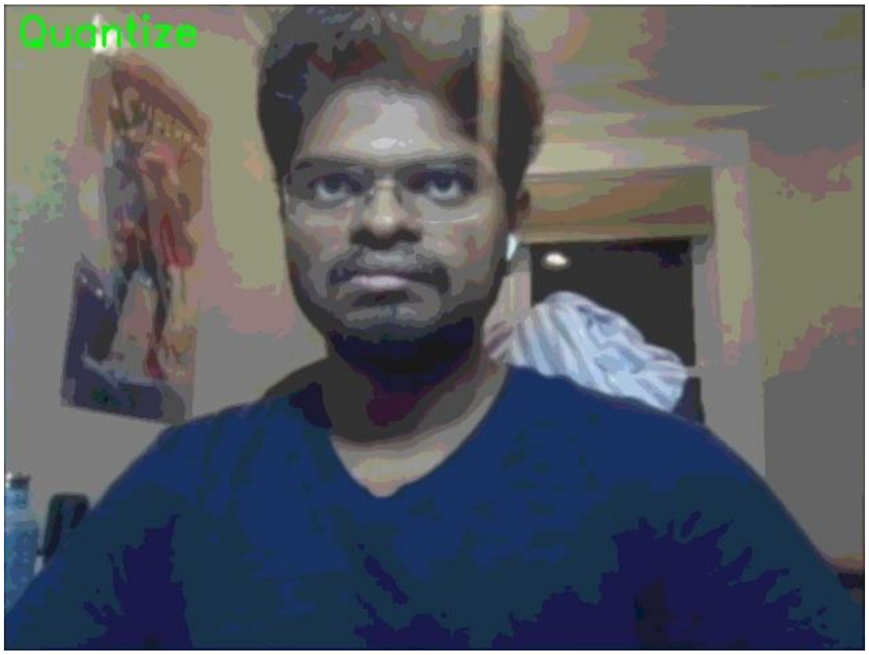
8. Gradient Magnitude Filter

Our code can apply a gradient magnitude filter over the live feed and display the output. The output is attached below.



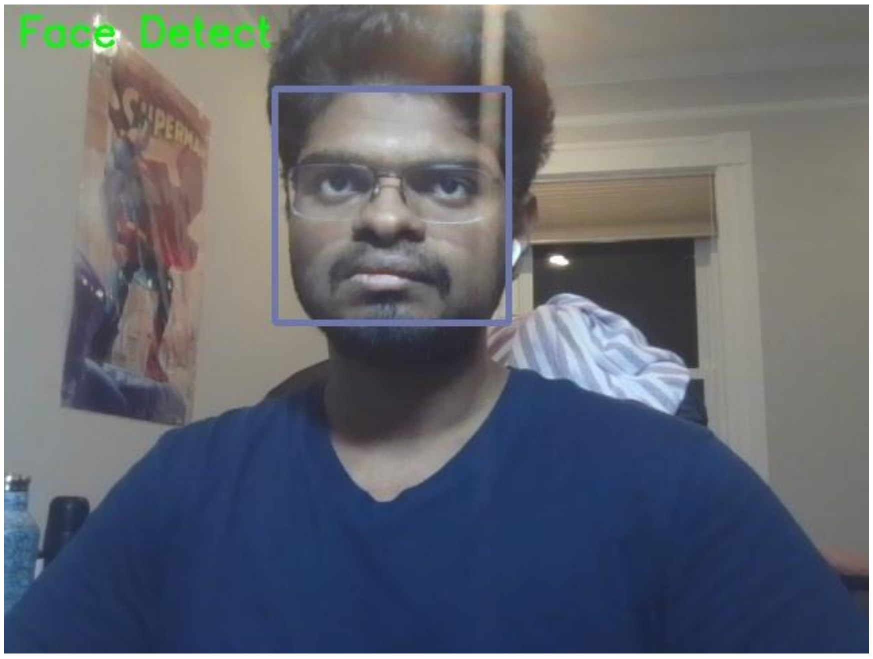
9. **Blur and Quantization**

Our code can apply a blur and quantization filter over the live feed and display the output. The output is attached below.



10. **Face detection**

The face detection code provided by the professor was integrated into our code and our code can identify faces and draw a box around it. The output is attached below.

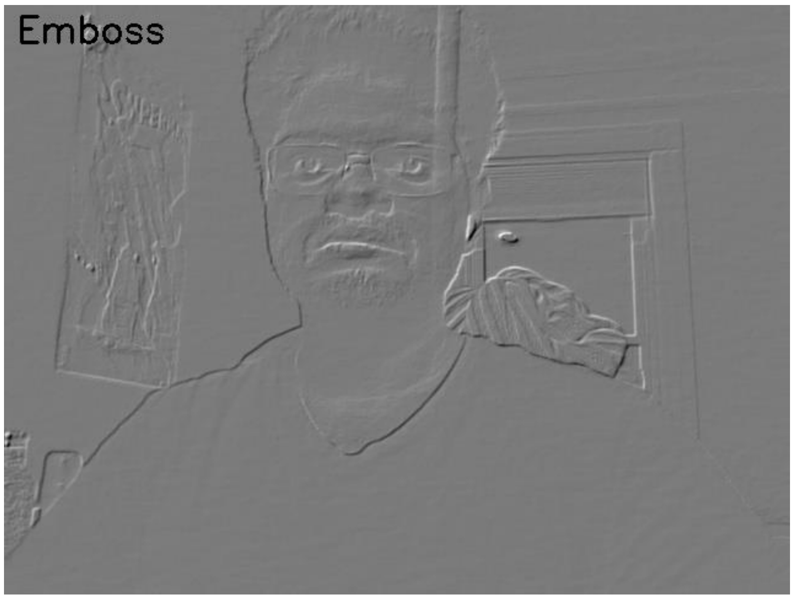


11. Additional Filters

For task 11, we have implemented four filters.

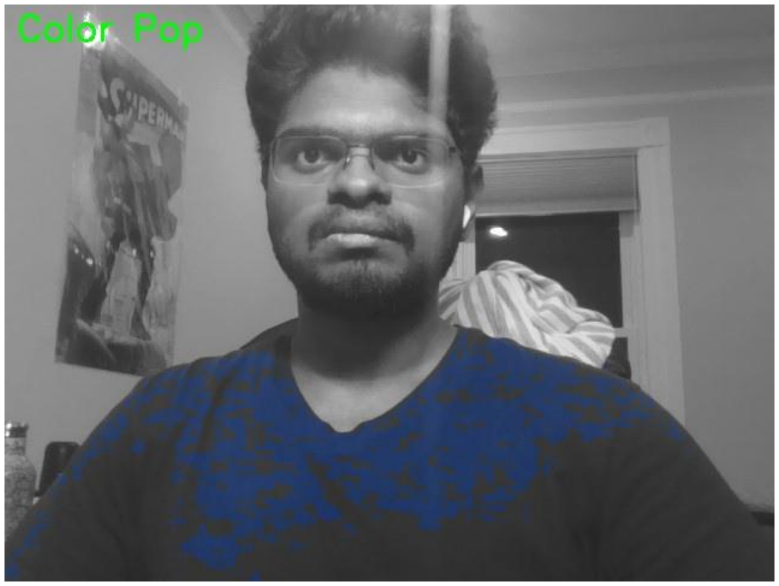
Emboss Filter:

Our code can apply an emboss filter over the live feed and display the output. The output is attached below.



Color Pop Filter:

In this filter I have set a range of HSV values to be dominant (In this example, blue is dominant). So, anything within the specified range will appear as colored image in the result whereas everything else will be greyscale. The output is attached below.

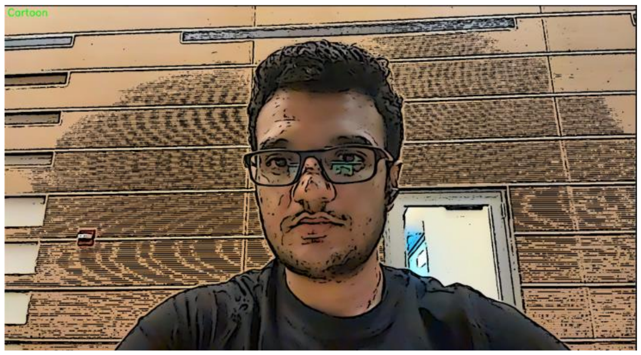


Negative Filter:

Our code can apply a negative filter over the live feed and display the output. The output is attached below.

Cartoon Filter:

Our code can apply a cartoon filter over the webcam feed in real-time. The output image is attached below.

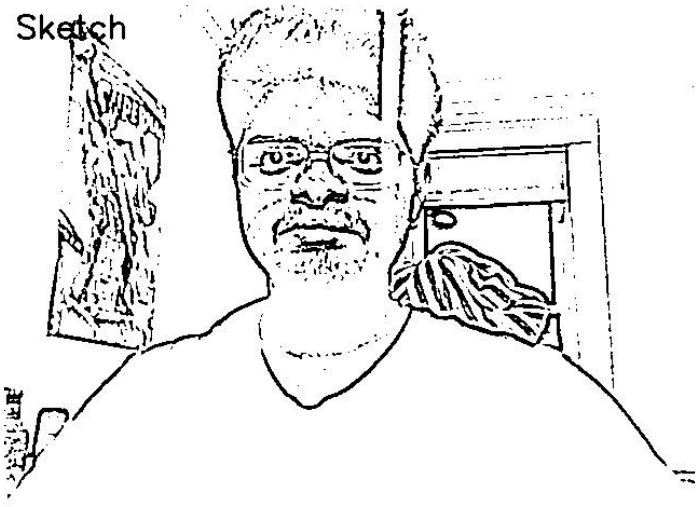


12. Extensions:

For the extensions we have added another new filter to our code. We have also added a feature to increase and decrease the brightness and contrast of the output feed and the user can also now save a video of the output.

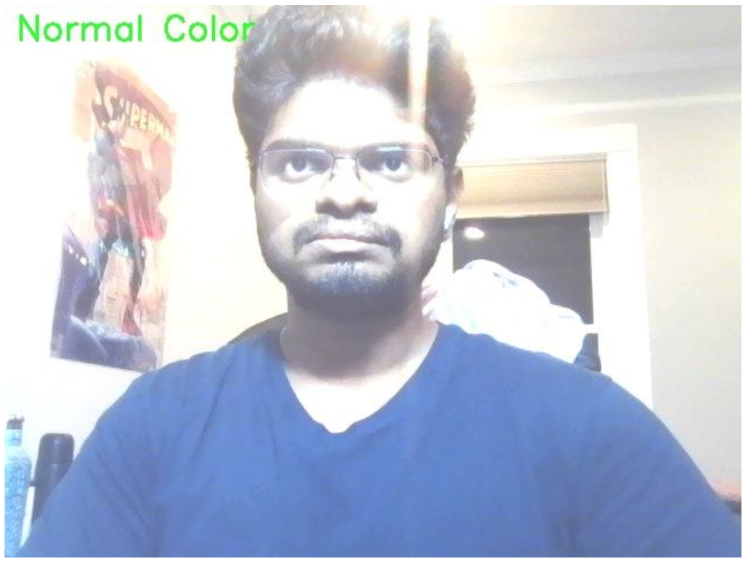
Sketch Filter:

Our code can apply a sketch filter over the live feed and display the output. The output is displayed below.

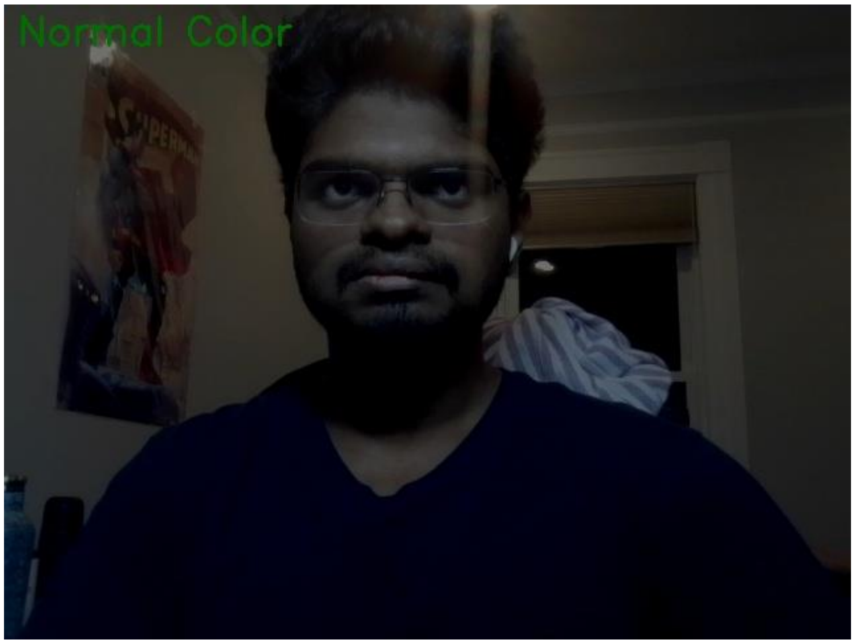


Increase Brightness and Contrast:

When the user presses the '+' button on the keyboard, the brightness and contrast increases. The output is displayed below.



When the user presses the '-' button on the keyboard, the brightness and contrast decreases. The output is displayed below.



Video Recording:

Our program can record from webcam even when applying filters. When the user first presses the 'r' button on keyboard the recording begins. A text 'Video recording' is displayed on the frame and when the user presses 'r' again the recording is stopped and saved. We successfully tested this feature and the link to the recording is attached below:

Link: https://drive.google.com/file/d/14n1nhQKOr5KGpJlilCirkR48yBgnvGRu/view?usp=sharing

Reflection:

Ravi Shankar Sankara Narayanan:

Through this project, I deepened my understanding of OpenCV and its application in C++. Although I had previous experience with computer vision projects, this project served as a comprehensive refresher of the fundamental concepts. Interestingly, the most challenging aspect was not the coding itself, but rather the configuration of CMake for project building. This experience, while initially confusing, boosted my confidence and proficiency in handling similar tasks in the future. Now, I feel well-equipped to undertake more complex computer vision projects using C++, having gained a solid foundation and practical experience from this project.

Vishaq Jayakumar:

This project has been a significant learning experience, enhancing our understanding of image manipulation techniques and its significance in computer vision. We encountered various challenges from setting up the necessary libraries for the project to implementing these image manipulation techniques. Personally, I learnt a lot about CMake and its significance in building a C++ based project. Maintaining a directory hierarchy and support for multiple library dependance is some of the key features offered by CMake.

Acknowledgements:

https://docs.opencv.org/4.5.1/d9/df8/tutorial_root.html