Project-1

-Kirti Kshirsagar

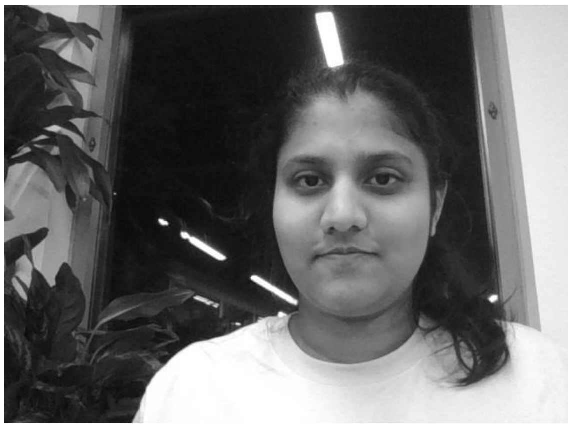
Project Description:

This project-1 aims to enhance and manipulate video streams through a collection of functions for adding filters, face detectors and effects. Leveraging the OpenCV library, the program encompasses various image processing tasks, starting from basic operations like grayscale conversion and blur filters using the opencv function to more intricate effects like sepia tones and edge detection. The different functionalities applied include customizable 5x5 blur filters, custom greyscale, Sobel filters for edge detection, and quantization for artistic effects. Additionally, the project incorporates facial detection and highlighting using a Cascade Classifier. This project experimented with transformations like negative image creation, box filters for a unique blurring effect, and sharpening filter using Laplacian kernel to emphasize edges. The modular structure allows easy integration of new features and makes it a versatile platform for exploring real-time video manipulation. Overall, the project provides a comprehensive exploration for computer vision learners like me, enabling the creation of visually engaging and creatively modified video content.

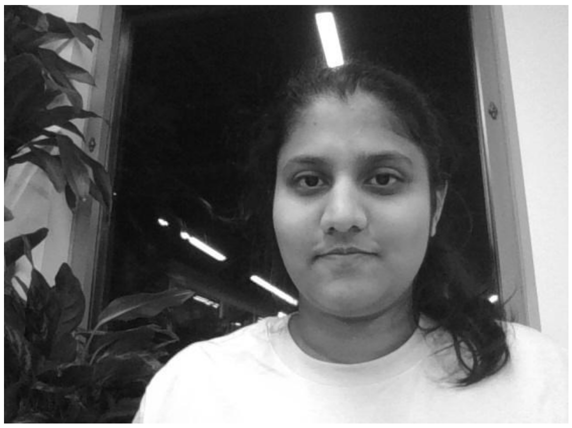
Tasks:

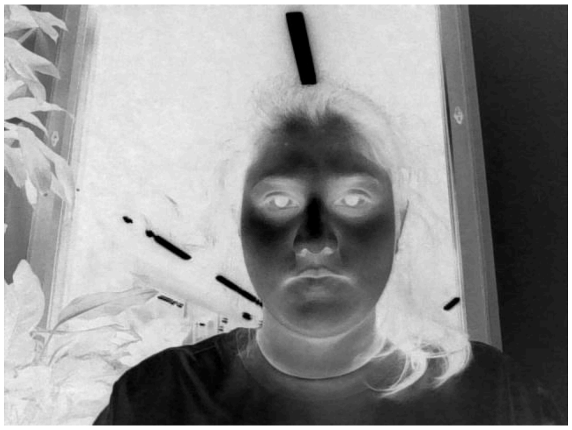
- Task 1- Implemented keypress q for quitting and s for saving a screenshot.
- Task 3- Grayscale using cvtColor function
  In the cvtColor function, the RGB to grayscale conversion is typically performed using the formula:
  $$Y=0.299*R+0.587*G+0.114*B$$
  Here, the weights 0.299, 0.587, and 0.114 represent the luminance contribution of each color channel to the resulting grayscale pixel value. These weights are based on the perceived intensity of each color channel to the human eye.

Original Image

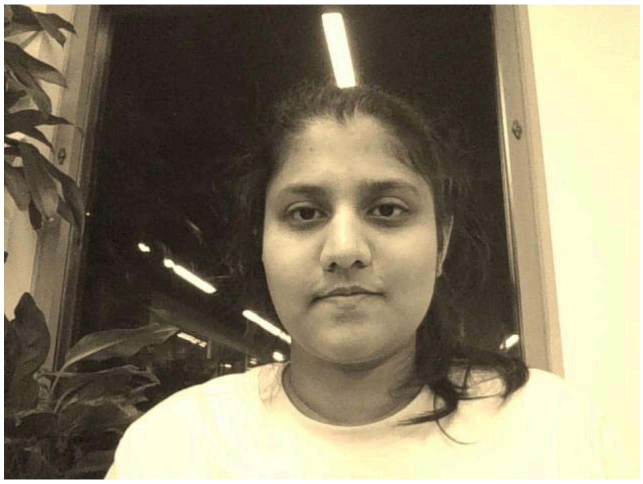

grayscale Image

- Task 4 - Custom greyscale version
  So the implementation I chose for this was to subtract the red channel from 255 and copy the value to all three color channels. This method creates a different kind of grayscale as cvtColor employs a weighted sum of the color channels to create the grayscale representation. Also, in the implementation of custom grayscale conversion, we manually iterate pixel-wise.


Using cvtColor grayscale function


Using custom Grayscale function

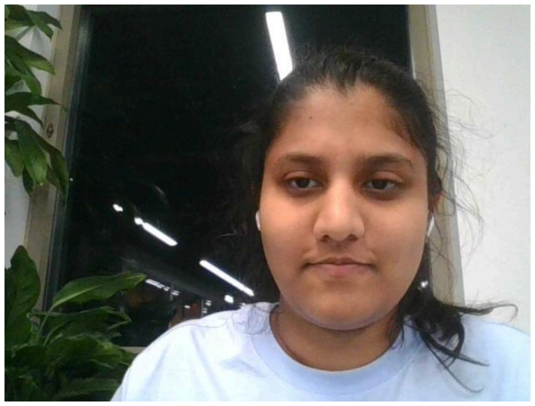
- Task 5 - Sepia Tone Filter


Sepia Tone Filter

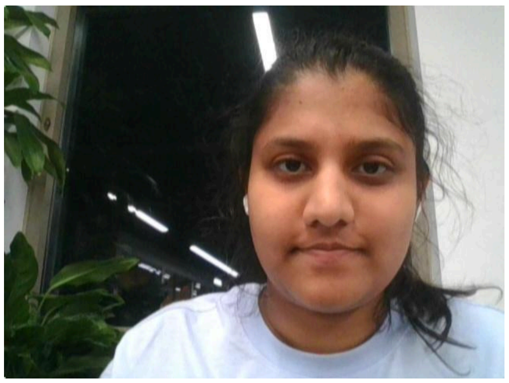
  I have created new variables to store the calculated value and am using the original RGB values in the computation. I can ensure this because I am using the pixel values of each channel stored in a different separate variable altogether, known as 'pixel'.

- Task 6 - Blur Effect
  I was not able to compare the timing for both sub-task of this task because of some issue with sys/time.h not being compatible with windows. But still I can ensure my blur

method using two separable filters is faster than 5*5 filtered blur function. Since, I have used ptr<> in the blur5x5_2 and only at<> in the blur5x5_1 function, the blur5x5_2 is faster.

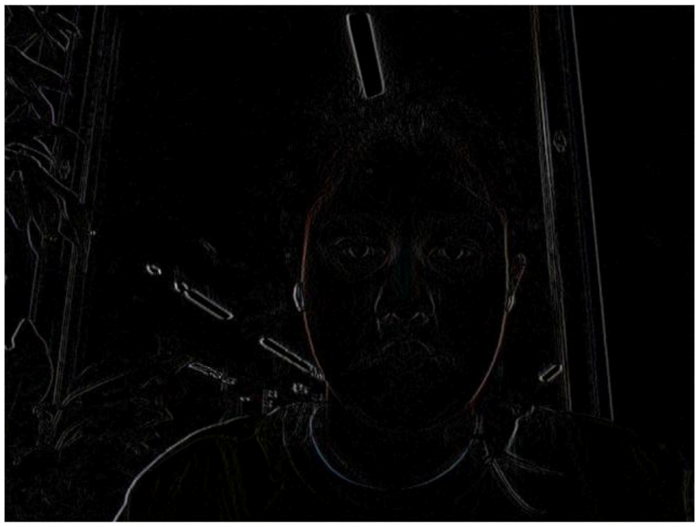


Original

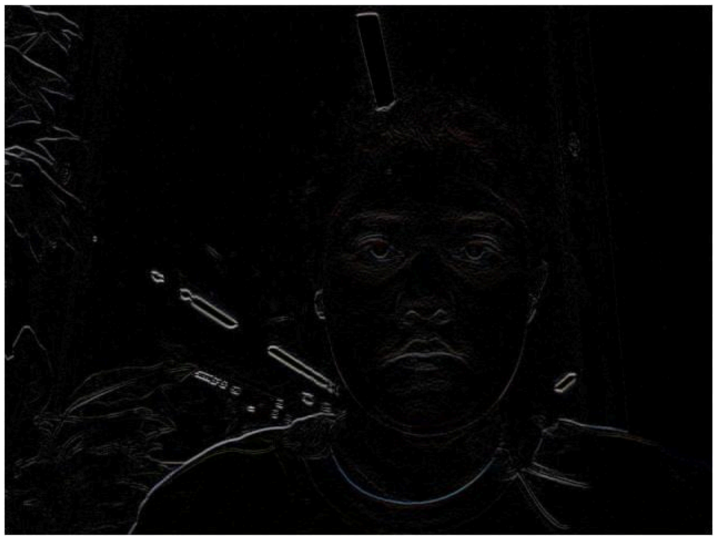


Blur Image

- Task 7- Sobel-x and Sobel-y functions

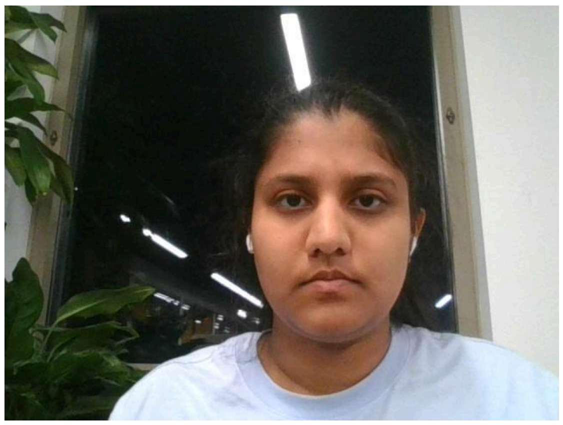


Sobel-x

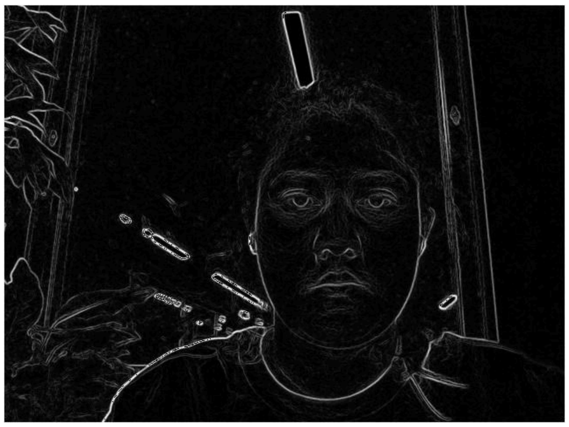


Sobel-y

- Task 8- Magnitude

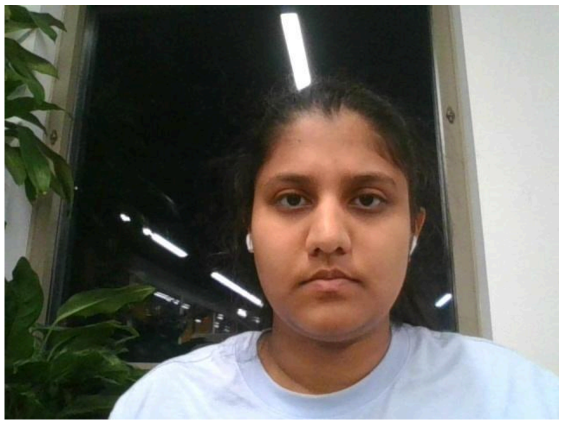


Original

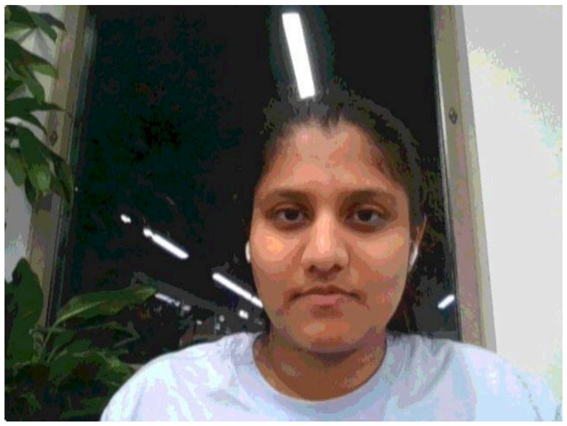


Magnitude

- Task 9- Blurred and Quantized image

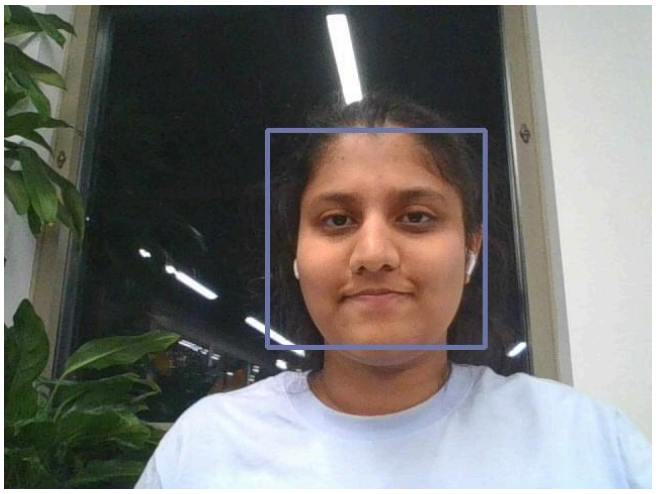


Original



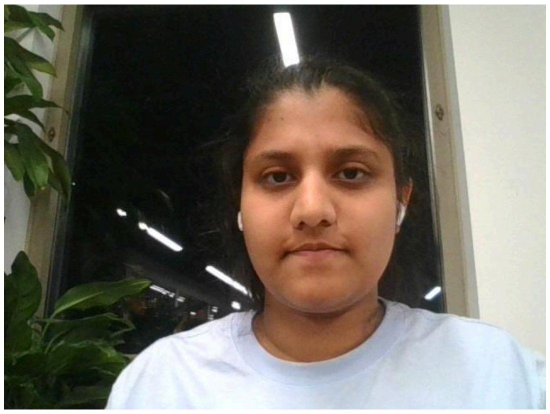
Blur-Quantized

- Task 10- Face Detection

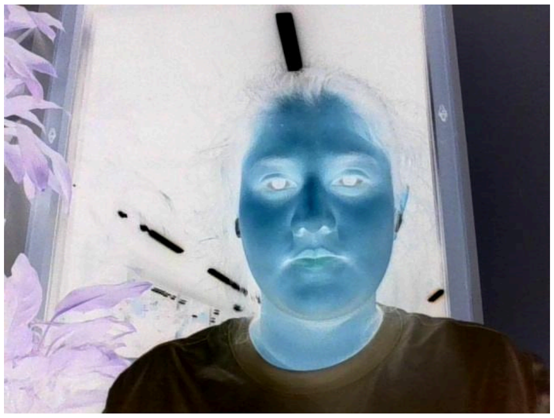


Face detection using box drawing

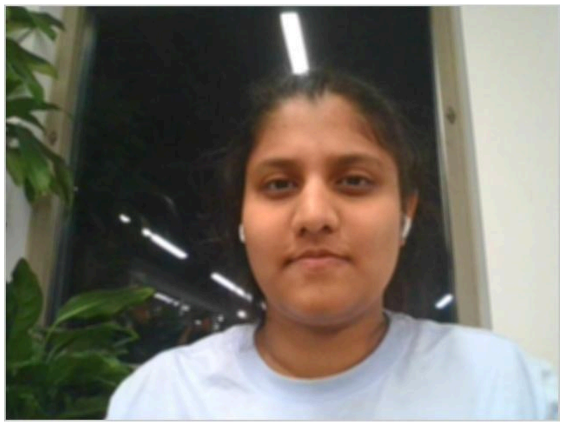
- Task 11 -



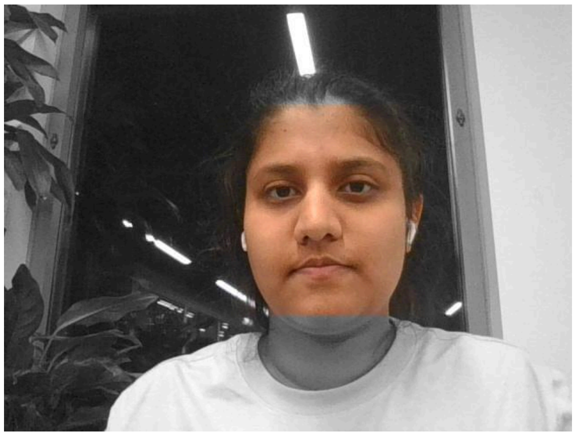
Original



Negative Image



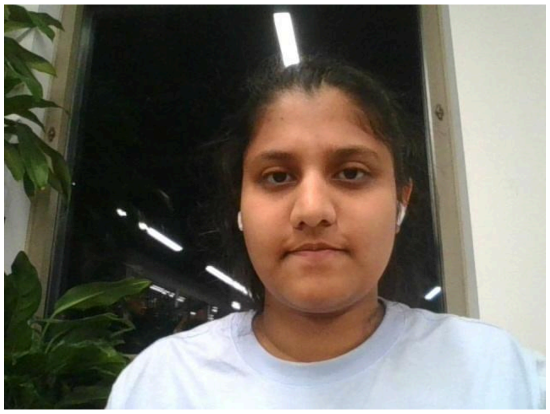
Box Filtered Image

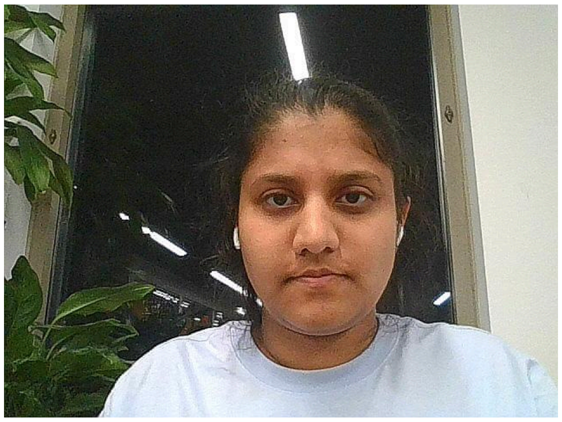


Only Face is colorful, rest is grayscale

- Extension -

Task 1. Applied sharpen filter using Laplacian Kernel.


Original


Sharpened Image

Task2. I have implemented recording of the live camera feed whenever the user wishes to, which even includes the keypress effects of filters and manipulations. The recording starts when the user presses ‘r’ and stops upon keypress ‘p’. This file is then saved. I’ll be adding a google drive link of the recorded video.
Video Link -
https://drive.google.com/file/d/1HuC4J5c5XWaNXUuzbflr6IzGDnlZ7wMe/view?usp=sharing

Reflection:

Through building and working with the image and video processing program in C++ and OpenCV, I gained valuable insights into the practical implementation of computer vision algorithms. Understanding the basics of filters, effects, and convolution operations deepened my understanding of image manipulation techniques. I even got to know about the significance of convolution with respect to applying image filtration techniques by executing tasks such as blurring/ sharpening of images and frames.
This Project gave me an opportunity to hands-on experience, broadened my understanding and cleared some basic concepts from computer vision for me.

Acknowledgement of material consulted:

- OpenCV Documentation
- https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html
- Stack Overflow

Installations:

So I am working on Windows 10, Visual Studio Code, Msys2, CMake, OpenCv 4 and C++.