CS 5330 – Pattern Recognition and Computer Vision

Assignment 1 – Image and Video Manipulation using Computer Vision

Name – Haritha Selvakumaran

Overview: This project is about developing an image and video processing and manipulation application using OpenCV along with C++ libraries. The program constantly captures the video in the form of stream and works on each frame based on the key pressed by the user. It primarily has a few features such as greyscale conversion, sepia filter, Sobel filters, detection of face, etc. There are also a few additional features where the user can choose cartoony filter, paintify filters, texture synthesis filters and embossing effect. In addition, the user can also save the images or videos and as well as convert the video back to its original form. The filter.cpp and filter.h contain all the manipulation functions, while the vidDisplay.cpp is the main file that checks the key pressed and calls the corresponding filters accordingly. The details of each filter with results are as follows-

No-filter photo for reference:

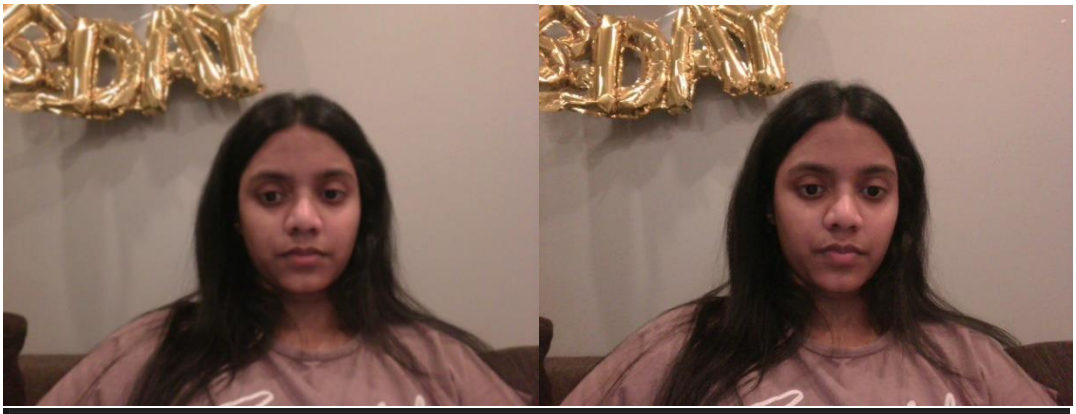
1. **Grayscale filter –** When pressed ‘g’, the vidDisplay.cpp converts RGB image to grayscale by multiplying scalar 0.299 to red pixels, 0.587 to green pixels, which also signifies our eye is more sensitive to green, and 0.114 to blue pixels. There’s also alternate grayscale, which is a customized grayscale and gets activated when pressed ‘h’. Over here the weighted sum is changed as we multiply red with 0.5, green with 0.3, and blue with 0.4 which makes the picture much brighter.

a. Grayscale filer b. Alternate grayscale filter

2. **Sepia filter –** This filter gives an antique camera look to the images. The filter was implemented by fetching all the red, green, and blue channel values pixelwise and computing new red, green, and blue pixel values using a weighted sum of the old values. These values were then clamped and assigned to the destination image.

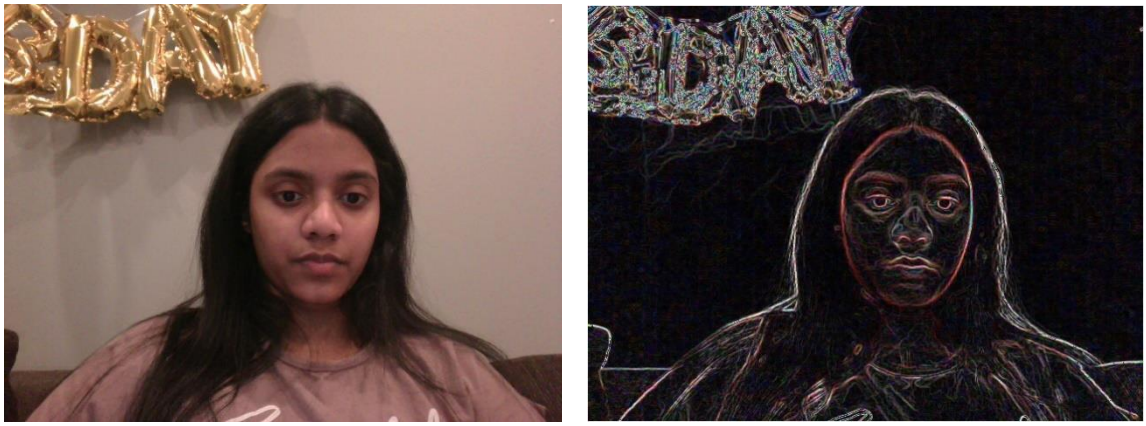
c. Sepia filter

3. **Blur filter –** Here, we implemented 2 blur filters using ‘b’ key press. One was the naïve approach blur3x3_1 where we multiply the Gaussian filter over the image. The other blur3x3_2 function is where we use the gaussian filter as a separable filter which makes the convolution faster based on the time reports as well.



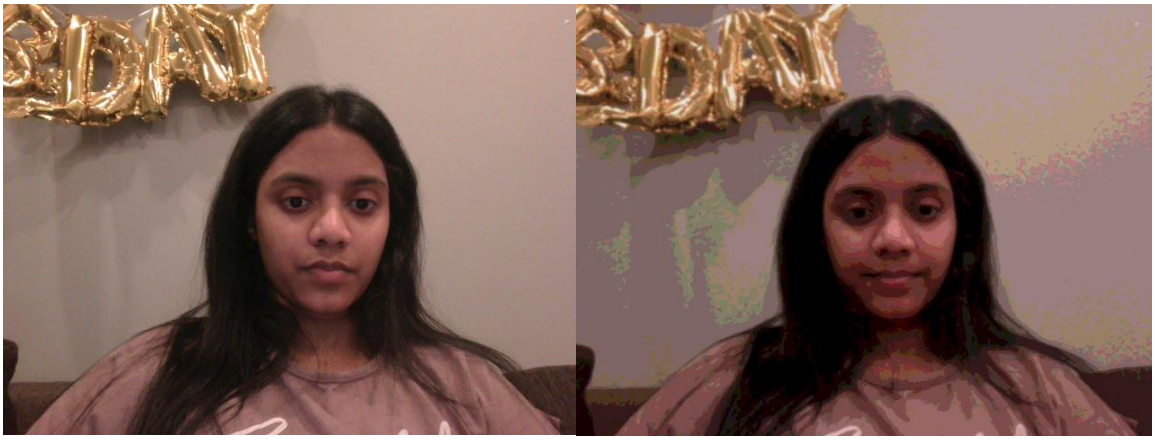
d. Blur filter besides normal filter followed by the time taken for both functions.

4. **Magnitude** – We implemented magnitude using keypress ‘m’ and using the Sobel x and Sobel y filters, merges them, and uses the Euclidean distance.



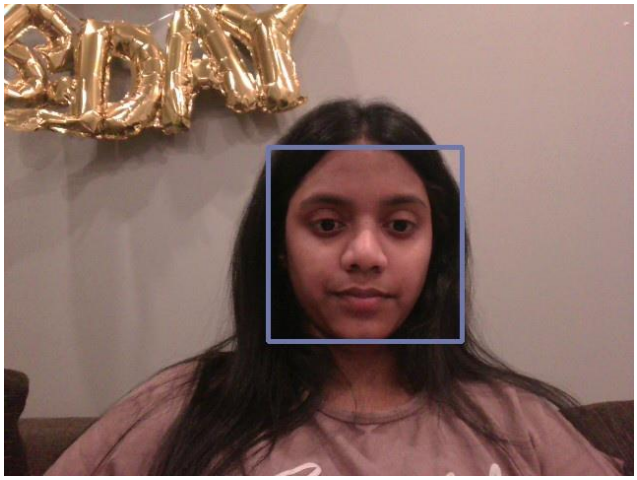
e. Normal image vs magnitude filter image

5. Quantize filter – This filter takes in a color image, blurs it, and adds quantization to it based on the levels given by the user. This filter is applied using the 'l' keypress.



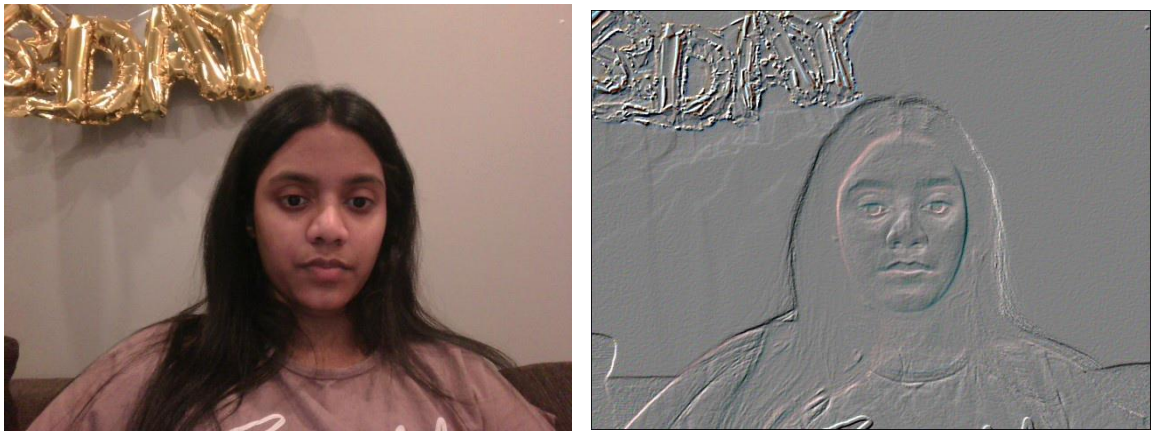
f. Normal image vs quantized image

6. Face detected – When clicked 'f' button, the face detect.cpp file is activated which when draws a bounding box around the faces detected in the videocam



g. Box showing face detection

7. Emboss effect – Implemented the emboss effect when pressed ‘e’, which uses the Sobel x and Sobel y outputs and uses a dot product which is calculated by using 0.7071 as the weighted sum for pixels.

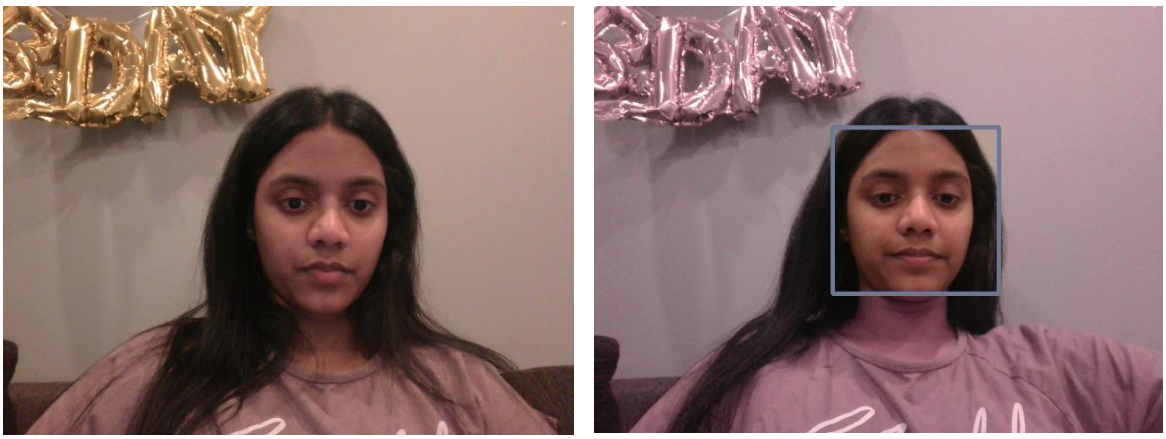


Normal image vs Emboss image

8. Negative effect – This effect is when we invert the pixel values in an image and can be used by ‘c’ keypress.

Normal effect vs Negative effect

9. Color Faces – This effect is where only the face is in the usual color, while the background is turned into a completely different color/grayscale. This filter is activated when clicked ‘d’.

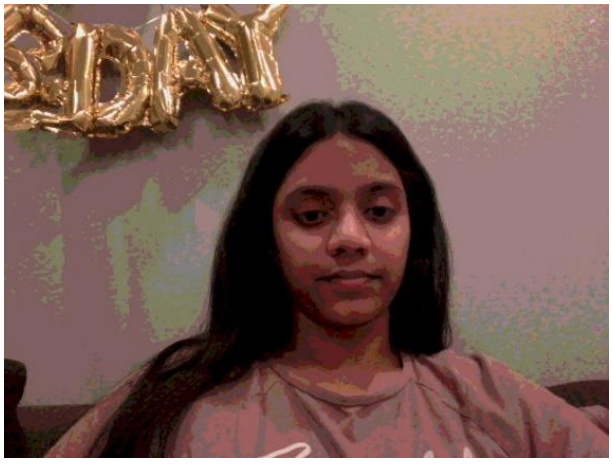


Extension –

1. Save images – In the project, when pressed ‘s’ the user can save that frame along with the filter that is currently being applied. All the images in the report were saved in this method.
2. Save videos – When the user clicks ‘v’, they can save a small video along with the filters.
3. Applied a few filters additional filters –
    a. **Cartoonify** – This filter is activated when pressed ‘a’.

    b. **Texture Synthesis –** This filter is activated when pressed 't’. Tried to implement a simple texture synthesis, where the image is overlapped with a texture.

    c. **Paintify** – The filter is activated when pressed ‘z’, which uses quantization and canny detection.



Short Reflection – Throughout this project, I explored the various functionalities that OpenCV has to offer using C++ libraries. Also learnt how accessing each pixel and pixelwise manipulations work, and fundamental image manipulation concepts work. Grasped understanding of many custom filters such as Sobel filter, blur using Gaussian filter, and grayscale filters. Due to which I was able to implement much more complex filters such as the cartoonize filter or the paintify filter. Also, learnt how saving images and videos work along with the filters. Overall, this project sparked an interest to explore OpenCV for broader applications.

Acknowledgements

OpenCV Documentation - https://docs.opencv.org/4.x/

Stack Overflow and online community