CS 5330

Pattern Recognition and Computer Vision

Project 1: Real-time filtering

Name: Vidya Ganesh
NUID: 002766414

Overview of the overall project:

This project's main goal is to become familiar with the opencv package's features, such as the mat type, read, display, and write functions for images and videos, as well as how to apply the fundamentals to build real-time video filters. The implementation of the filters entails the alteration of the RGB values at each pixel. By averaging the values of each RGB channel, we first produced a grayscale filter. Then, using a magnitude filter for the black edges and a blurQuantize filter for the blurred and quantized color, we constructed a filter for cartoonization. The following section contains the finished product.

For loops are used to iterate through each row and column by the size of each pixel value in order to access the pixels. In order to manipulate the values of pixels at the proper locations and show pixels at the proper locations, careful attention is paid to the data type of RGB values at each pixel. The project also focuses on filters to simplify or modify visual content in images/videos while emphasizing or preserving the perceptually important information.

Required Image 1: Shows the original and **cvtColor** version of the greyscale video. This was done using the opencv function cvtColor.

Fig Set 1: The image in the right is the window streaming the grey video using the **cvtColor** function and the left image is from the normal live video.

Required image 2: shows the customized greyscale video using the **greyscale** function

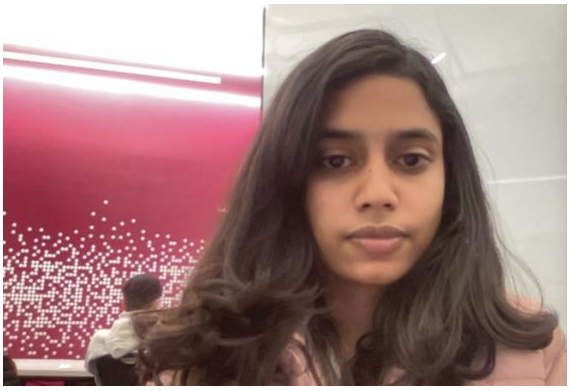


Fig Set 2: This is the screenshot of the video captured with the alternative grey filter. It is achieved by averaging each color channel in the greyscale function.

Required Image 3: shows the original and the blurred image using the **blur5X5** function.


Fig Set 3: The first image is the screenshot of the original video. The second image is the screenshot of the video with the blur5x5 filter. It can be observed that the second image from the video is blurred.

Required Image 4: shows the original and the gradient magnitude image using **blur5x5.**


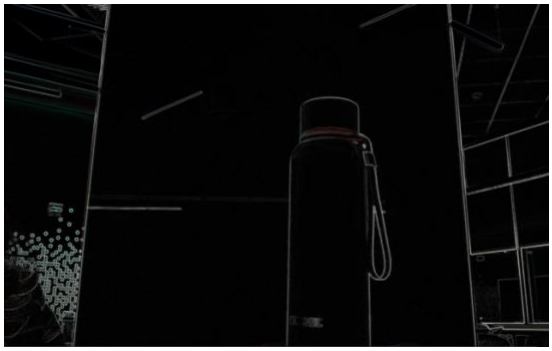

Fig Set 4: The first image is the screenshot of the original video and the second is the screenshot of the video captured with the gradient magnitude filter. This filter combines xsobel and ysobel , which highlights the edges where the colors change the most.

Required Image 5: shows the original and the blurred/quantized image using the magnitude function

Fig Set 5: The first image is the screenshot of the original video and the second is the screenshot of the video captured with the blurred/quantized filter. It can be observed that in the second image, the change of colors becomes less smooth and the resolution is lower.

Required Image 6: shows the original and the cartoonized image using the **cartoon** fuction.

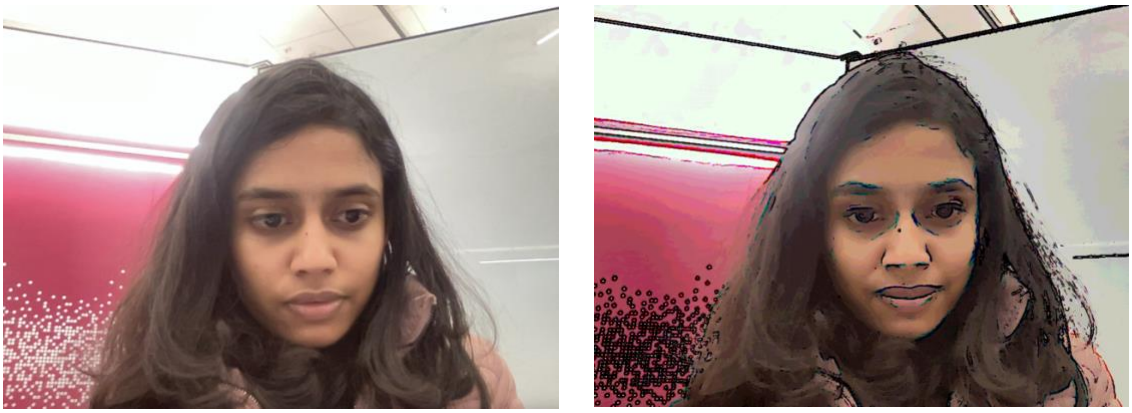

Fig Set 6: The first image is the screenshot of the original video and the second is the screenshot of the video captured with the cartoon filter. It can be observed that in the second image, the image looks cartoonized.

Required Image 7: show the original and the modified image in your report. If you want to demonstrate using a video, post it on Google drive and put the URL in your readme.txt file.


Fig Set 7: The first image is the screenshot of the original video and the second is the screenshot of the video captured with the cartoon filter. It can be observed that in the second image, the image has all the white colors inverted to black and the other way round for black colors and so on.

Extensions

Extension 1: Allows you to provide any desired caption to your video stream


Fig Set 8: The first image is the screenshot of the original video and the second is the screenshot of the video captured with the caption filter. It can be observed that in the second image, the image has a caption that was given in the command line while executing. In this way we can add any desired caption to the video output.

Extension 2: Allows you to rotate the video stream to any desired angle


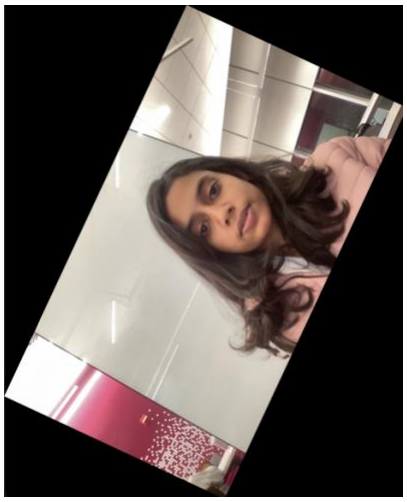

Fig Set 9: The first image is the screenshot of the original video and the second is the screenshot of the video captured with the caption filter. It can be observed that in the second image, the image has been rotated by some angle (here 60 degree) using the rotate video function.

Reflections:

I learned more about the opencv packages—particularly the objects and functions they offer—through this project. I also learned more about C++. Additionally, I learnt how to manipulate images from scratch by using filters and altering the values of each pixel's color channel. I was able to comprehend the justification for the opencv functions as a result. I wonder whether there will be any faster approaches to handling all pixels simultaneously, similar to what we can do with numpy.

Acknowledgements:

Professor Bruce, TA Mrudula and TA Tanmay helped me with the segmentation faults.