Real-time filtering

Kruthika Gangaraju

Description:

In OpenCV, image filtering is done to modify the image. These modifications are performed because they can help in extracting necessary information from the image. We can blur, sharpen, remove unwanted objects, etc. for the necessary final image. In this project, we implemented different filters by extracting pixel values and applying the necessary filter to each pixel. This helped us understand how a filter works. In order to apply a filter, we needed to know about kernel and convolutions. A kernel basically helps us how the pixels involved in a computation go through those changes to obtain the desired results. Convolution, when performed using the kernel matrix and the corresponding image matrix, produces the value for the output image pixel. After understanding these, it became very easy. First task was to extract the pixel values. This is done by using a simple for loop run over the rows and columns of the source image and using the .at<Vec3b> function. Once we extract the pixel, we need to multiply the kernel matrix with a matrix surrounding the pixel with the same dimensions. To simply this task, I used separable filters, i.e., applied the filter twice, first horizontally, then vertically. For the blur and sobel filters, we have different kernel matrices, so the logic would be the same, just the values would differ. Once these matrices are obtained, it is easy to perform further applications since they are just derived from these two. Lastly, we can either save a frame from the video (using imwrite() function or save a video sequence (using .write() function)

Required images:

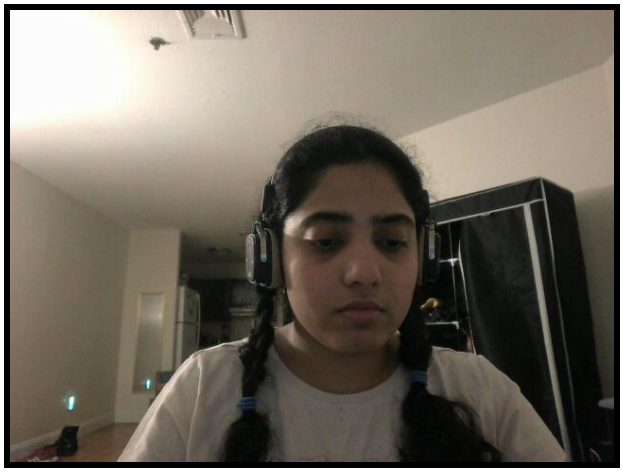

Fig 1: Original image

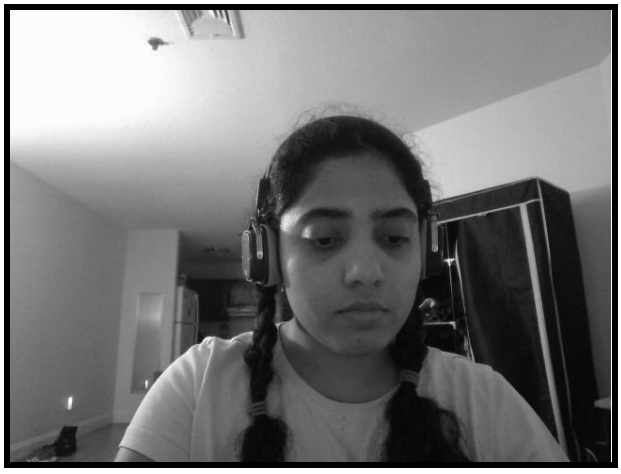

Fig 2: Grayscale image

Required image 1: The grayscale image was obtained simply by using the cvtColor() function. It takes the output image and changes the color scale of the original image to any desired result, for example, in this case, a grayscale image.

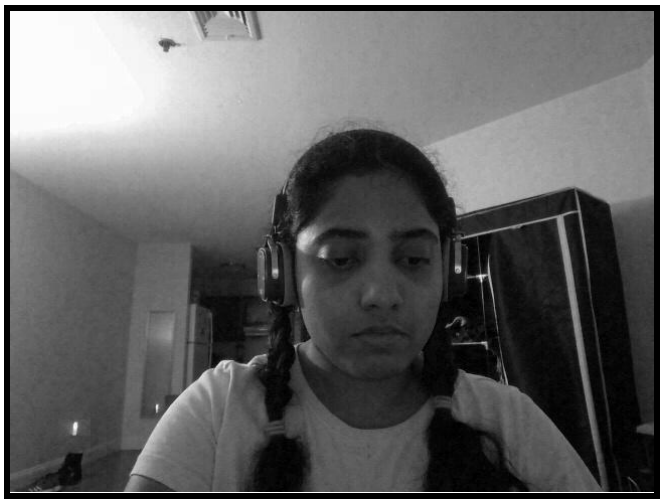

Fig 3: Alternate Grayscale image

Required image 2: An alternate method of achieving grayscale is by copying one channel to the other two channels. Here, I copied the blue channel to the red and green channels.

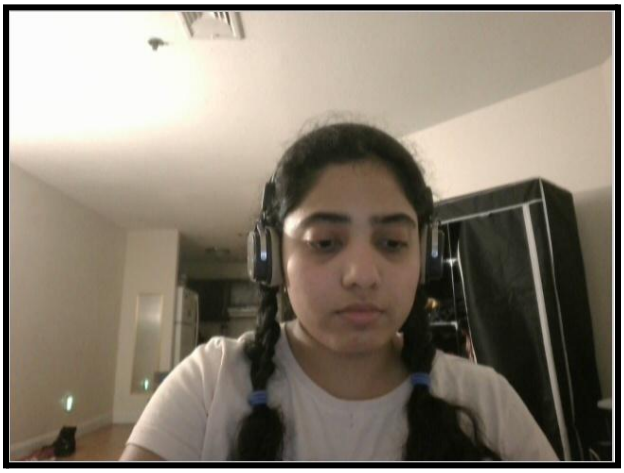

Fig 4: Blur image

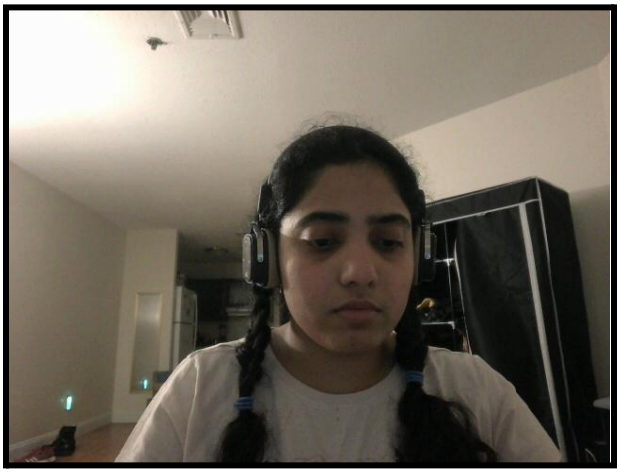

Fig 5: Original image

Required image 3: As discussed earlier, a blur image can be obtained by accessing the pixel values and performing convolution using a kernel matrix. Here, I used two separable filters (horizontal and vertical) whose resultant would give a 5x5 kernel matrix. The kernel matrix used was {1,2,4,2,1}.

Fig 6: Gradient magnitude image

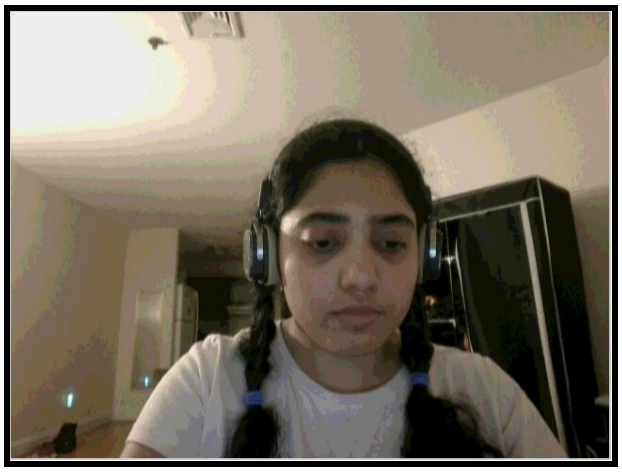

Fig 7: Blur quantized image

Required image 4: A gradient magnitude can be obtained by performing the following operation: $\sqrt{(sobelx)^2 + (sobely)^2}$. Sobel filters in x and y directions can be achieved by using a 3x3 kernel matrix: {-1,0,1} (horizontal) x {1,2,1} (vertical) for x direction and its transpose for y direction. Using the same logic as for blur, we can get images for Sobel x and Sobel y, then perform the given operation to obtain a gradient magnitude image.

Required image 5: An image can be quantized by setting threshold levels and changing all the pixel values to a singular value for that particular threshold limit. Here, a blur image is taken, the threshold level is set to 15 and the filter is applied using the given logic.

Fig 8: Cartoon image

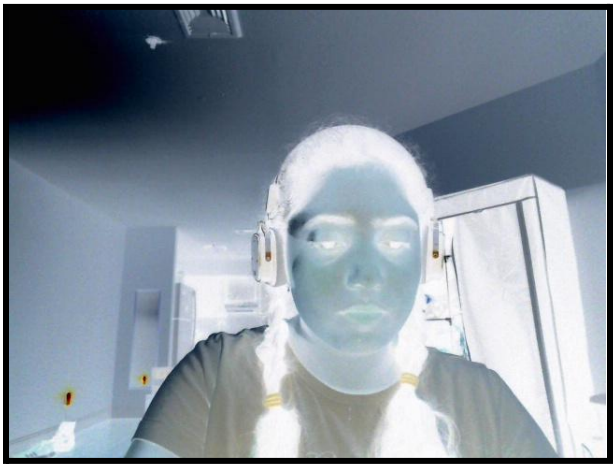

Fig 9: Negative image

Required image 6: A cartoon image follows the same logic as the blur quantized image, except for changing values to some randomly calculated pixel value, we choose a threshold limit to change all the pixels having value greater than the threshold to black.

Required image 7: For the special filter, I chose to make the image negative. It has a very simple logic. I just changed each channel value to its inverse, i.e., just subtracted the channel from 255.

Acknowledgements:
Firstly I would like to thank Prof. Bruce Maxwell for introducing this topic and always being available to clarify any doubts I encountered. Next I would like to thank the TAs, especially Heet Sakaria, David Anderson, Santosh Vasa, Guanang Su and Ravina Lad, for helping me out understand the problems and helping me with the errors I kept getting. Finally I would like to mention a few sites that helped me understand few OpenCV and C++ functions:

- Image Filtering Using Convolution in OpenCV - GeeksforGeeks
- Read, Write and Display a video using OpenCV | (learnopencv.com)
- How to put a text in an image in OpenCV using C++? (tutorialspoint.com)
- Image Filtering Using Convolution in OpenCV | LearnOpenCV #
- Change the contrast and brightness of an image using OpenCV in Python (tutorialspoint.com)
- How To Install OpenCV C++ and Use It With Microsoft Visual Studio - YouTube