Michael Hart
CS5330 Spring 2023
Project 1: Real Time Filtering

Project Overview:

The purpose of Project 1 (Teal Time Filtering) was to demonstrate the basics of image processing in OpenCV with C++ by implementing several video filters to augment live video. The project output is four C++ files (imgDiplay.cpp, vidDisplay.cpp, filter.cpp, vidDisplay.h). The imgDisplay.cpp file compiles to a program that reads in an image and saves it when the users presses 's'. The vidDisplay.cpp files compiles to a program that can display live video, augment the live video using 13 different video filters, and allows the user to save an image from the live video at any time (using any video filter). The vidDisplay.cpp program allows users to toggle between any filter mode and save images from the video at any time by pressing 's'. Images are saved as filename "savedImage_[filter].jpg" (e.g., "savedImage_cartoon.jpg" or "savedImage_blur.jpg")[1]. The 13 filter modes implemented and their keys are:

'g' - Greyscale	'm' – Gradient Magnitude	'e' – Histogram Equalization
'h' – Alternate Greyscale	'l' = Quantize/Blur	'z' – Color threshold Sliders
'b' – Blur5x5	'c' – Cartoonize	'a' – Brightness Adjustment
'x' – SobelX	'n' – Negative	
'y' – SobelY	't' – Median Blur	

[1] You will need to update the directory/path variable to the location on your machine you wish to save the files. See README.txt

Images:

Image 1: Greyscale using cv::cvtColor

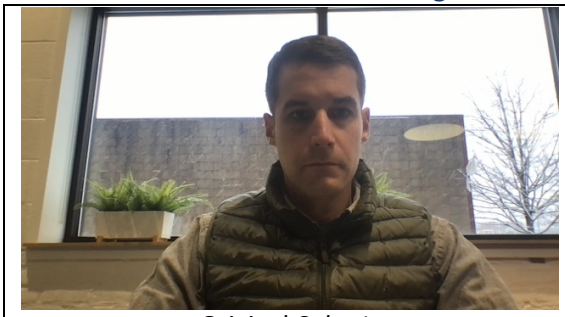
Original

Greyscale Image using cvtColor

This mode can be enabled by pressing 'g'.

Michael Hart
CS5330 Spring 2023
Project 1: Real Time Filtering

Image 2: Customized Greyscale Image


Original Color Image


Customized Greyscale Image

The alternate Greyscale filter was implemented by taking the average of each color channel at each pixel. The resulting image is slightly different than the Greyscale image implemented using OpenCV's cvtColor function. This mode can be enabled by pressing 'h'.

Image 3: Blurred Image

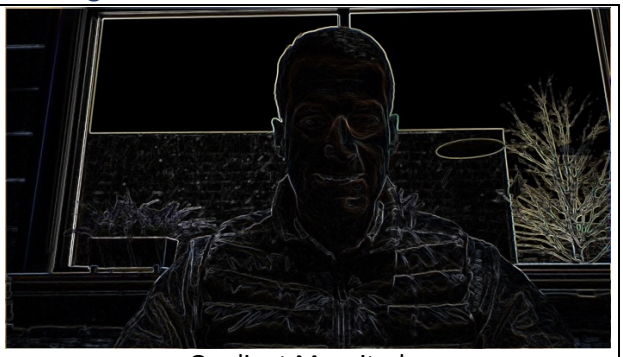

Original


Blurred

This was implemented using a 5X1 separable Gaussian filter of [1,2,4,2,1]. This mode can be enabled by pressing 'b'.

Image 4: Gradient Magnitude


Original


Gradient Magnitude

The Gradient Magnitude filter was implemented by combining a separable Sobel-X and separable Sobel-Y filter to determine the total gradient magnitude. The separable filters used were [-1,0,1] and [1,2,1]. This mode can be enabled by pressing 'm'. The SobelX mode can be enabled by pressing 'x' and the SobelY mode can be enabled by pressing 'y'.

Michael Hart
CS5330 Spring 2023
Project 1: Real Time Filtering

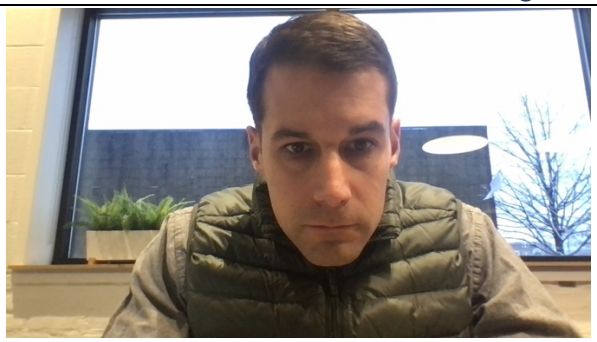
Image 5: Blurred/Quantized


Original


Blurred/Quantized

The Blurred/Quantized filter was implemented by splitting the range of each color channel into 15 quantiles. This mode can be enabled by pressing ‘l’.

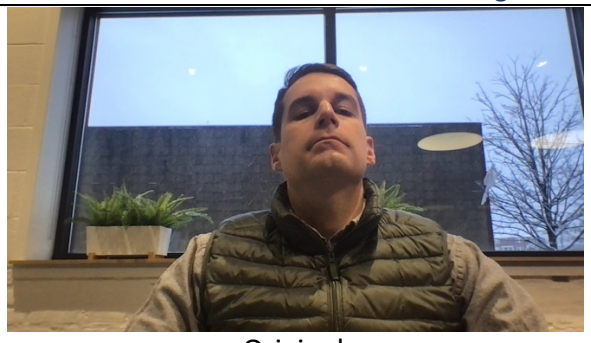
Image 6: Cartoonized


Original


Cartoonized

The Cartoonized filter was implemented by splitting each color channel into 8 quantiles and setting the magThreshold to 15. This mode can be enabled by pressing ‘c’.

Image 6: Negative Image


Original

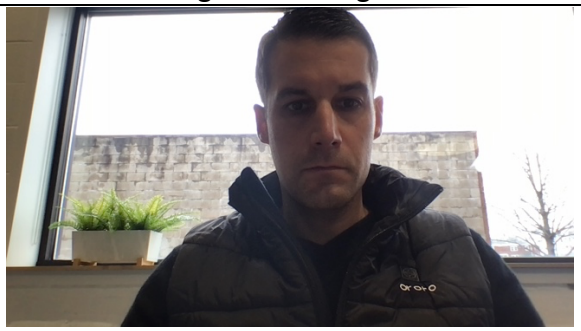

Negative Image

The Negative filter was implemented by displaying the inverse of each color channel c (255 - c). This mode can be enabled by pressing ‘n’.

Michael Hart
CS5330 Spring 2023
Project 1: Real Time Filtering

Extensions:

1. Smoothing effect using 5X5 median filter.



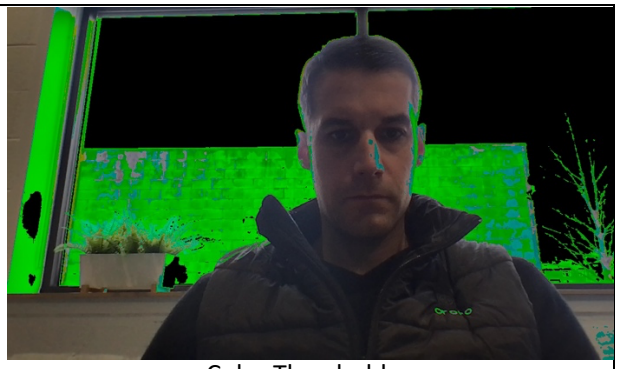
Original

Smooth with 5X5 median filter

I implemented a smoothing filter using a 5X5 median filter. Press 't' to toggle on/off the effect. The resulting image is smoothed in the areas where the pixel color are similar. Users can press 's' to save the image when in "medianBlur" mode.

2. Trackpads to change color thresholds

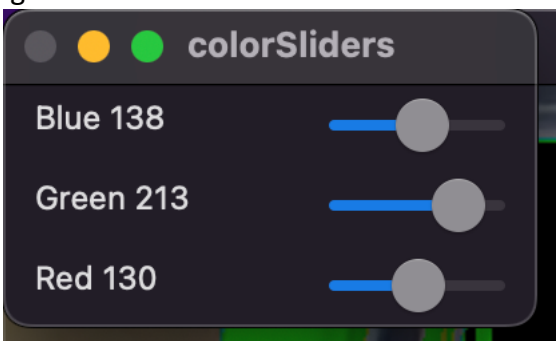
Original



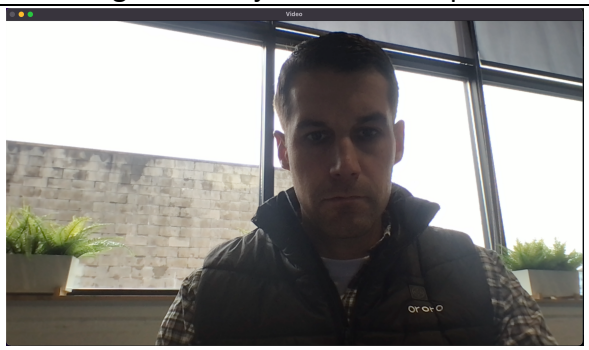
Color Thresholds

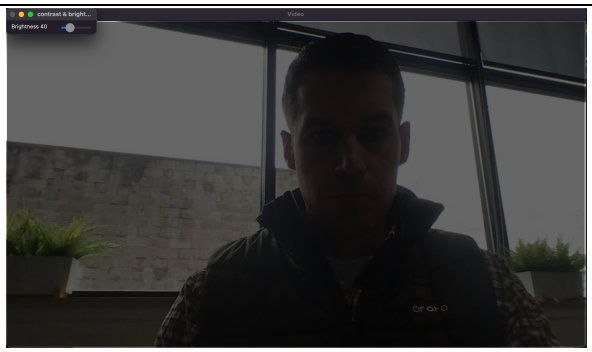
In this extension, I added a feature whereby when the user presses 'z', a window with 3 trackpads appears that the user can adjust to edit the colors displayed by the live video. The slider values affect the threshold values of green, blue, and red color values whereby color channel values are replaced with a 0 if the original image's color value is below the set threshold. Users can press 's' to save an image as well.

Michael Hart
CS5330 Spring 2023
Project 1: Real Time Filtering

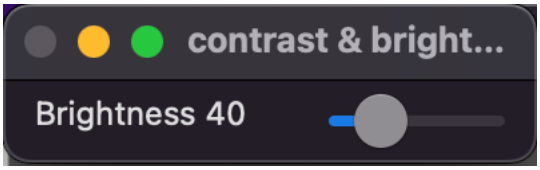


3. Brightness Adjustment Trackpads


Original


Brightness Adjustment

In this extension, I added a feature to allow the user to adjust the brightness of the video. Users can press ‘a’ change the live video to brightness mode where they can adjust the brightness between -100% and +100% via the trackpad. Users can press ‘s’ to save a still image when the brightness is adjusted.



4. Histogram Equalization


Original


Histogram equalization

Michael Hart
CS5330 Spring 2023
Project 1: Real Time Filtering

In this extension, I implemented a histogram equalization of the intensity values from scratch. When the user presses ‘e’, it toggles the “histEqualization” mode. This was done by creating a histogram of the intensity values for the image and normalizing them so that the resulting image intensity values map to the cumulative weighted intensity values. The resulting image is more balanced in terms of contrast and brightness. Users can press ‘s’ to save a still image when the histogram equalization is applied.

What I learned:

I learned many things about image processing, OpenCV, and C++ from this project. With regards to image processing, I learned about the math and logic behind different filters & effects, as well as how to implement them on images and live video. I learned how to read an image into memory and access individual pixels and color channels. I learned how to use different data types, color channel values, and color spaces and when it is appropriate to use them (e.g., uchar/CV_8U/Vec3b vs. signed short/CV_16S/Vec3s). I learned how to implement several types of filters including point operators (e.g., greyscale), separable linear filters (e.g., sobelX), and neighborhood filters (e.g., median). This was my first-time using C++ and I learned a lot about the language itself including syntax, structure, and the use of pointers. I implemented the filter functions using pointers to improve real-time performance and learned how to do “pointer math” to access the pixel color values above/below and left/right of the pixel in focus.

Acknowledgements:

I consulted several articles on the following sites:

- https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- https://www.tutorialspoint.com/
- https://www.geeksforgeeks.com/
- https://www.stackoverflow.com/