Project 1: Real-time filtering

by Yixiang Xie for CS5330 CVPR

Project Description

This is the first project for the class CS5300 Pattern Recognition & Computer Vision. The idea of the project is to familiarize students with C++ and OpenCV package, and the related mechanics of operations of images, such as opening, capturing, manipulating and writing images.
The prerequisite is setting up the environment for C++ development and OpenCV dependency. The first task is to read an image and display it, and to be able to capture keyboard input to receive interactive commands from user. Starting the second task, we need to read from camera feed and display live video. User can save image from the feed. Then we need to use the OpenCV function to convert the feed to grayscale. We also need to implement our own grayscale filter, Gaussian blur filter, Sobel X and Sobel Y filters, gradient magnitude filter (from Sobel X and Sobel Y), quantize filter (from blur) and finally a cartoonization filter (from magnitude and quantize filter).

Required Images

1. Grayscale using cvtColor function

Original image | Grayscale version using cvtColor function

2. Grayscale using the average of BGR channel

Original image | Grayscale version using avg of three channels

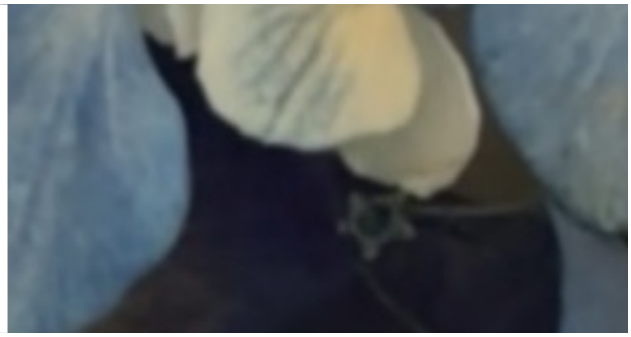
The reason why I implement the grayscale as the average of red, green and blue channel here is that some images are bright in one channel but dark in other channels, e.g. an image of an apple. If we choose either channel and copy that channel to the other two to create a grayscale image, there is a chance that channel is the dark channel and the image is just plain black. Calculating the average of three channels can avoid this. I also did some research by comparing different ways of implementing the grayscale filter in the extension part.

3. 5x5 Gaussian blurred image

Original image

Gaussian blurred image

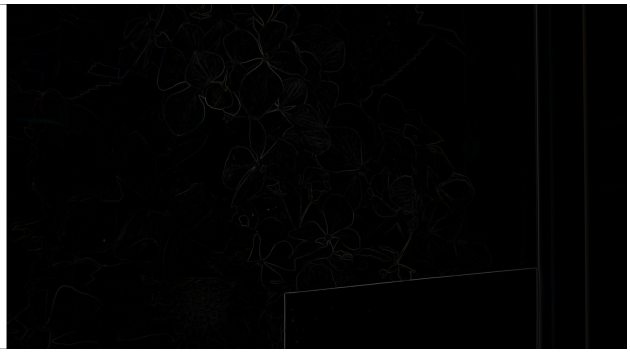
Original image (cropped)



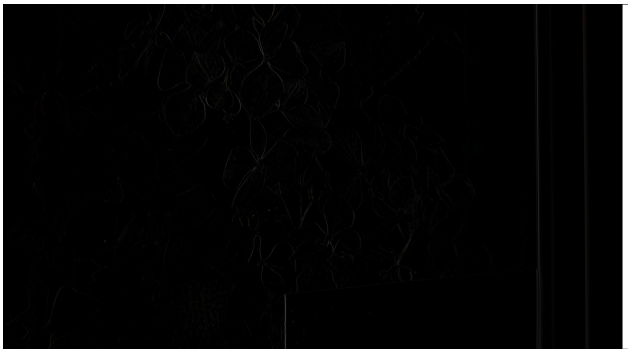
Gaussian blurred image (cropped)

4. Gradient magnitude image

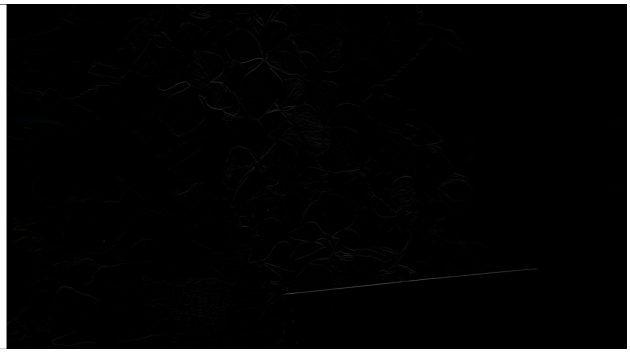
Original image



Gradient magnitude image

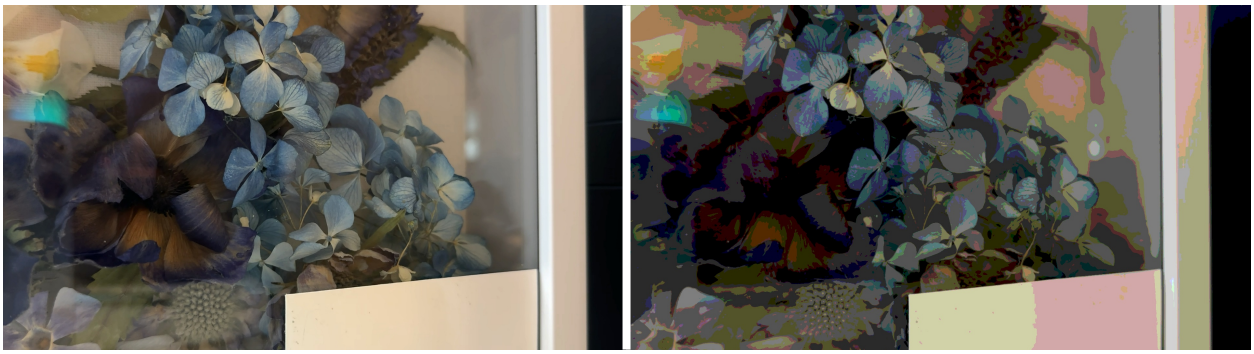


Sobel X image



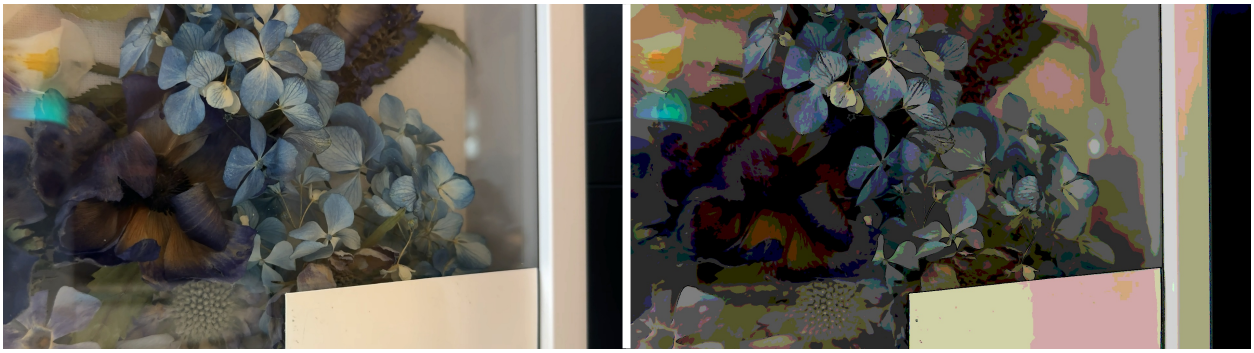
Sobel Y image

5. Blurred and quantized image



Original image | Blurred and quantized image (levels=6)

6. Cartoonized image



Original image | Cartoonized image (levels=6 & threshold=15)

7. Adjusting brightness

Original image | Brightness increased | Brightness decreased

Extensions

1. Explore different implementations of grayscale transform

Inspired by Task 4, I wanted to know what is the difference between several kinds of grayscale transformation: copying red channel to the other two (using red channel), using blue channel, using green channel, using the average of three color channel, using the value channel of HSV color space and using the L channel of Lab color space. The results are as the following images.

Original image

Grayscale using red channel

Using blue channel

Using green channel

Using avg of BGR channels

Using value channel of HSV

Using L channel of Lab

2. Adjusting contrast

Inspired by Task 10 adjusting brightness, I also implemented adjusting contrast.

Original image | Contrast increased | Contrast decreased

Reflections

In this project, I learned to use makefile and CMake to compile and link a C++ program. I learned to use OpenCV to operate on images and videos. Particularly, I learned about reading images and writing images, reading from camera feed and display a video, color spaces, image data types, pixel and color channel accessing, etc. I also learned about several filters and how to implement them and optimize the implementation.

Acknowledgments

imgDisplay displays a photography by Declan Sun on Unsplash.
Thanks to Chenjie Wu for help setting up the OpenCV environment.