CS5330: Pattern Recognition and Computer Vision

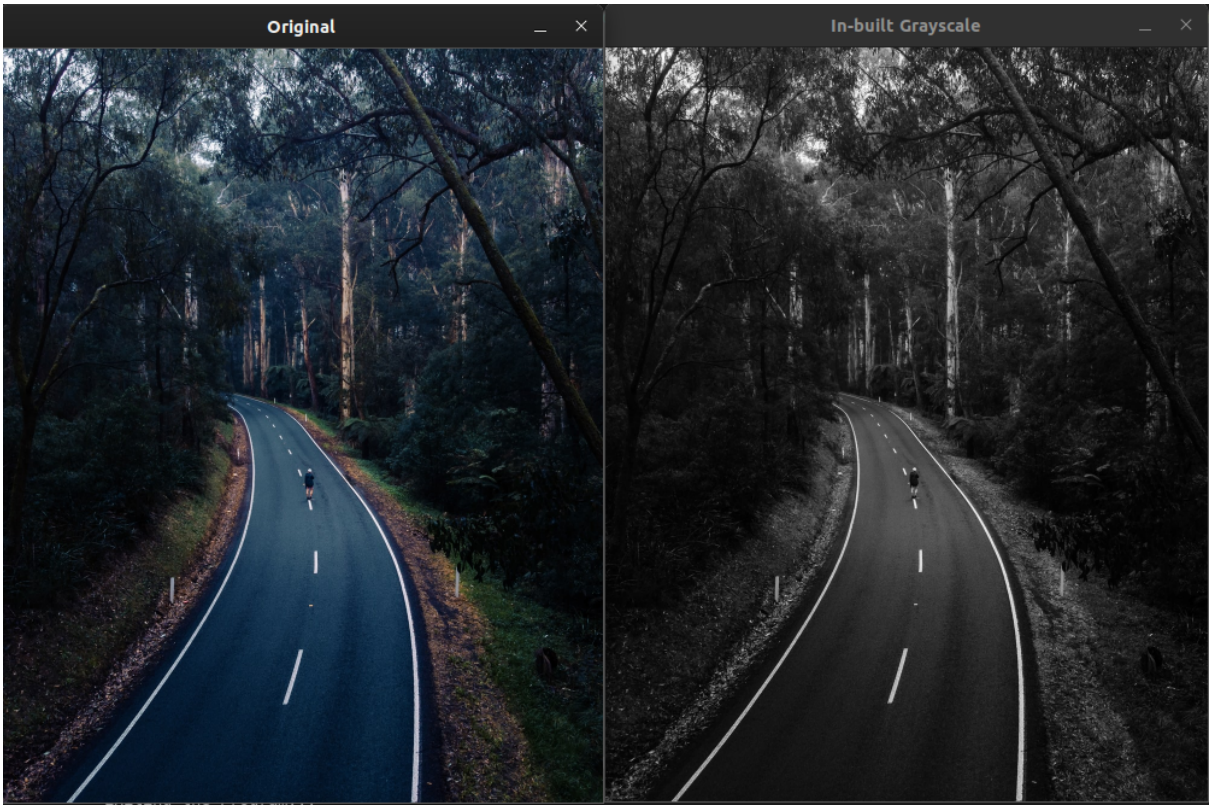
Project 1: Real-time filtering

Brief Description of the Project:

The purpose of this assignment was to familiarize us with C/C++, the OpenCV package, and the mechanics of opening, capturing, manipulating, and writing images. The whole assignment was divided into 10 tasks (Excluding Extensions) starting with displaying and saving a simple image to cartoonizing the image making use of all functions that we defined before. It was a well-structured assignment for beginners as it broke down the task of doing something as complicated (at least at beginner level) as cartoonizing an image that uses multiple image manipulating techniques into simple steps that were comparatively more feasible to achieve and didn't seem daunting. It helped us get familiar with the basic functions of OpenCV package and C++ language like displaying/saving an image, video streaming, grayscaling, blurring, application of filters, quantizing and then combining the results of all of the above to display a cartoonized image/video stream.

Required Image 1: Original and cvtColor version of Grayscale

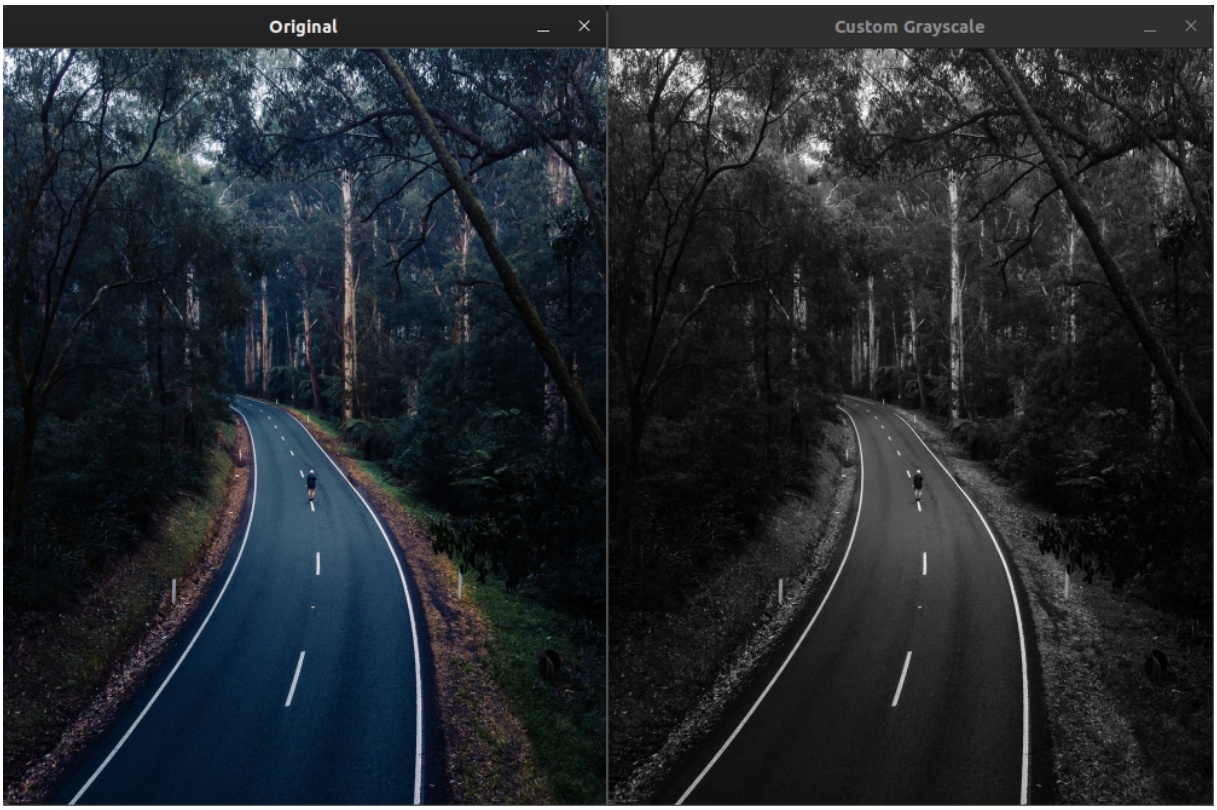
A side-by-side display of the original image and grayscale image converted using the cvtColor function of OpenCV.



Required Image 2: Customized Grayscale Image

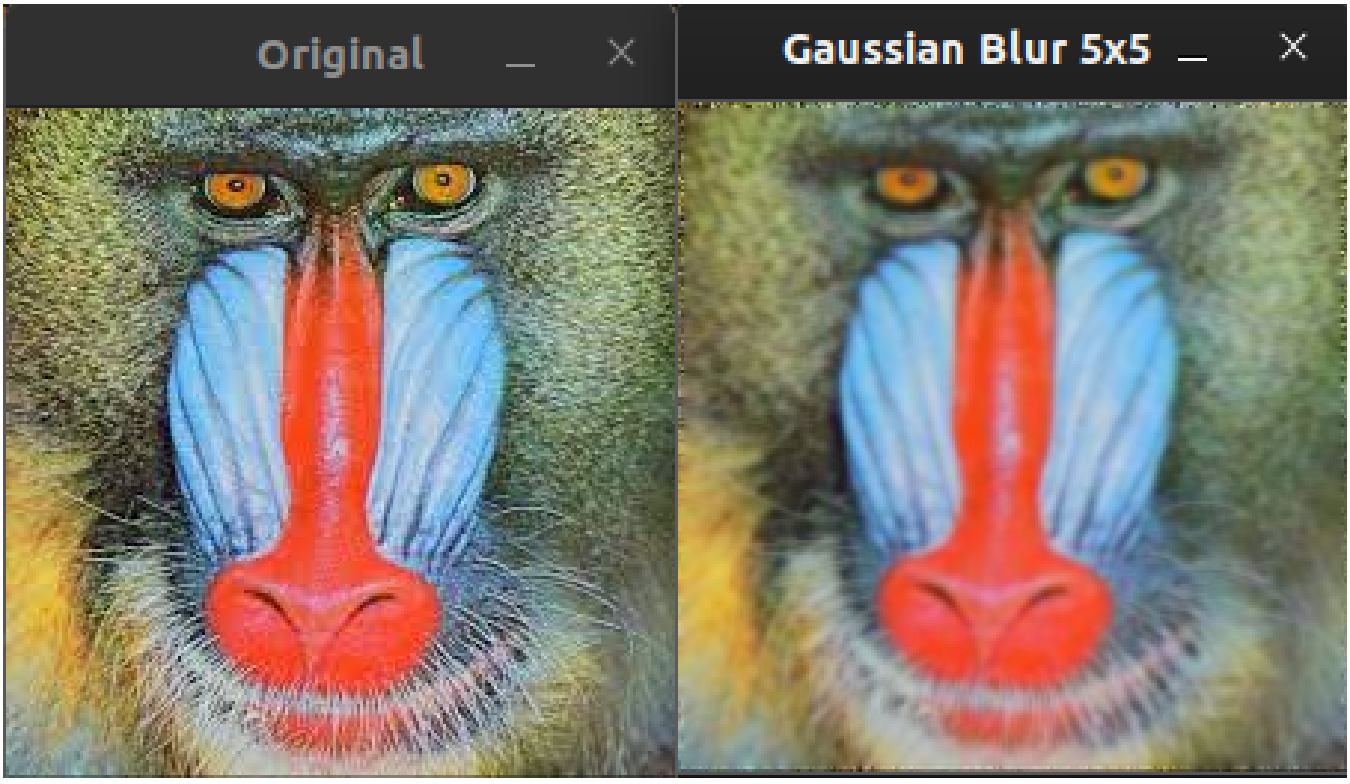
A side-by-side display of the original image and the grayscale image converted using a user-defined function. The image was generated using the following relation between the pixels of a grayscale image and the BGR channel:

Gray = 0.114*Pixel of Blue Channel + 0.587*Pixel of Green Channel + 0.299*Pixel of Red Channel.



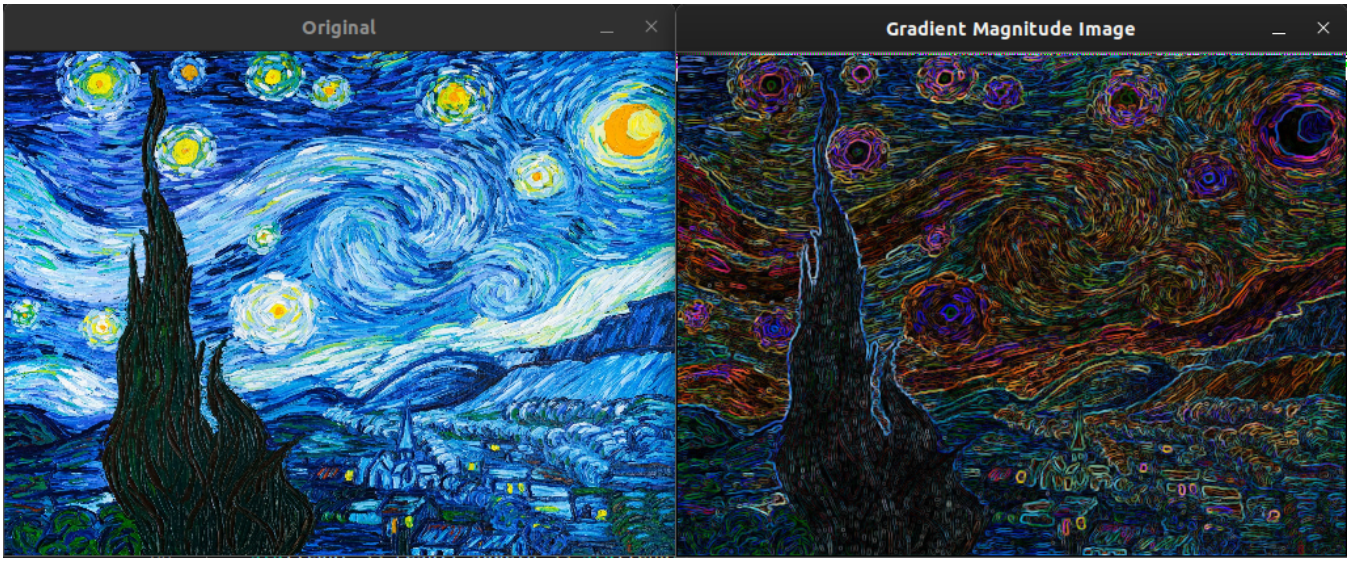
Required Image 3: Original and Blurred Image

A comparison between the original image and the blurred image where the blur is generated using a 5x5 Gaussian filter as separable 5x1 and 1x5 filters. The 1x5 filter is applied horizontally and the 5x1 filter is applied vertically.



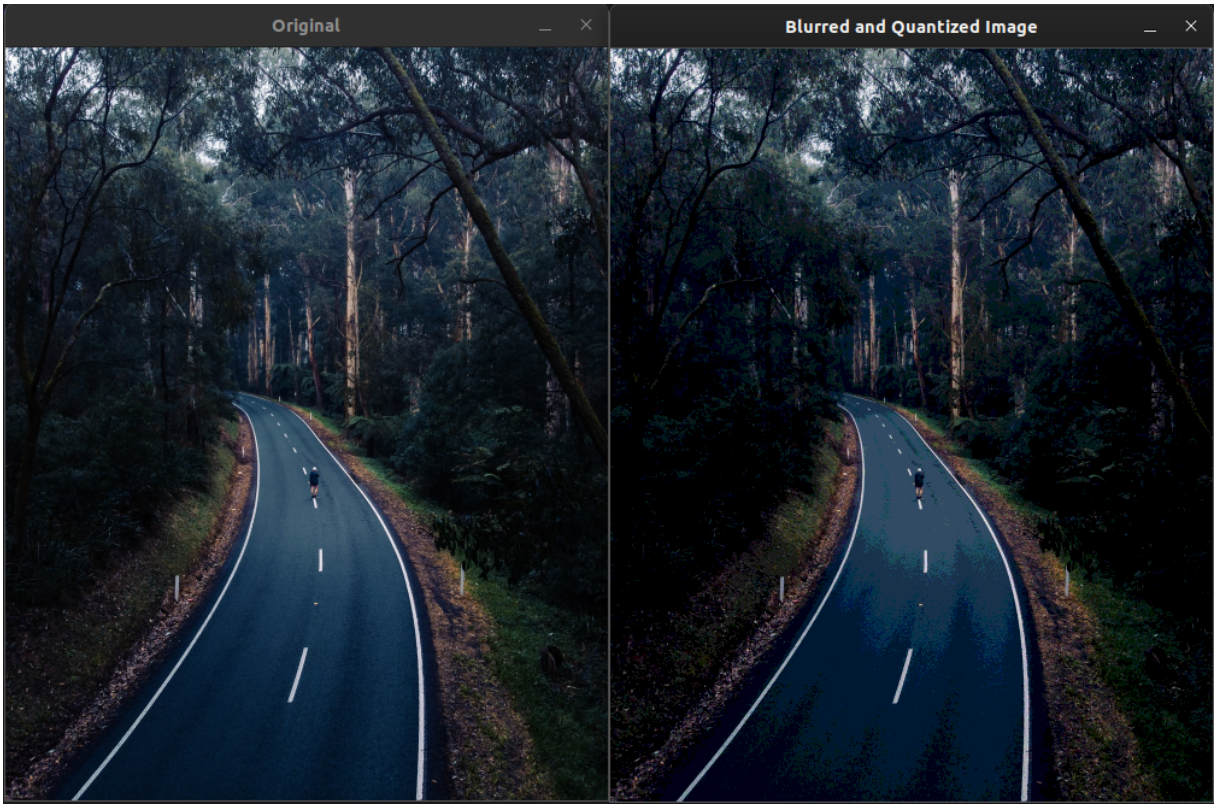
Required Image 4: Original and Gradient Magnitude Image

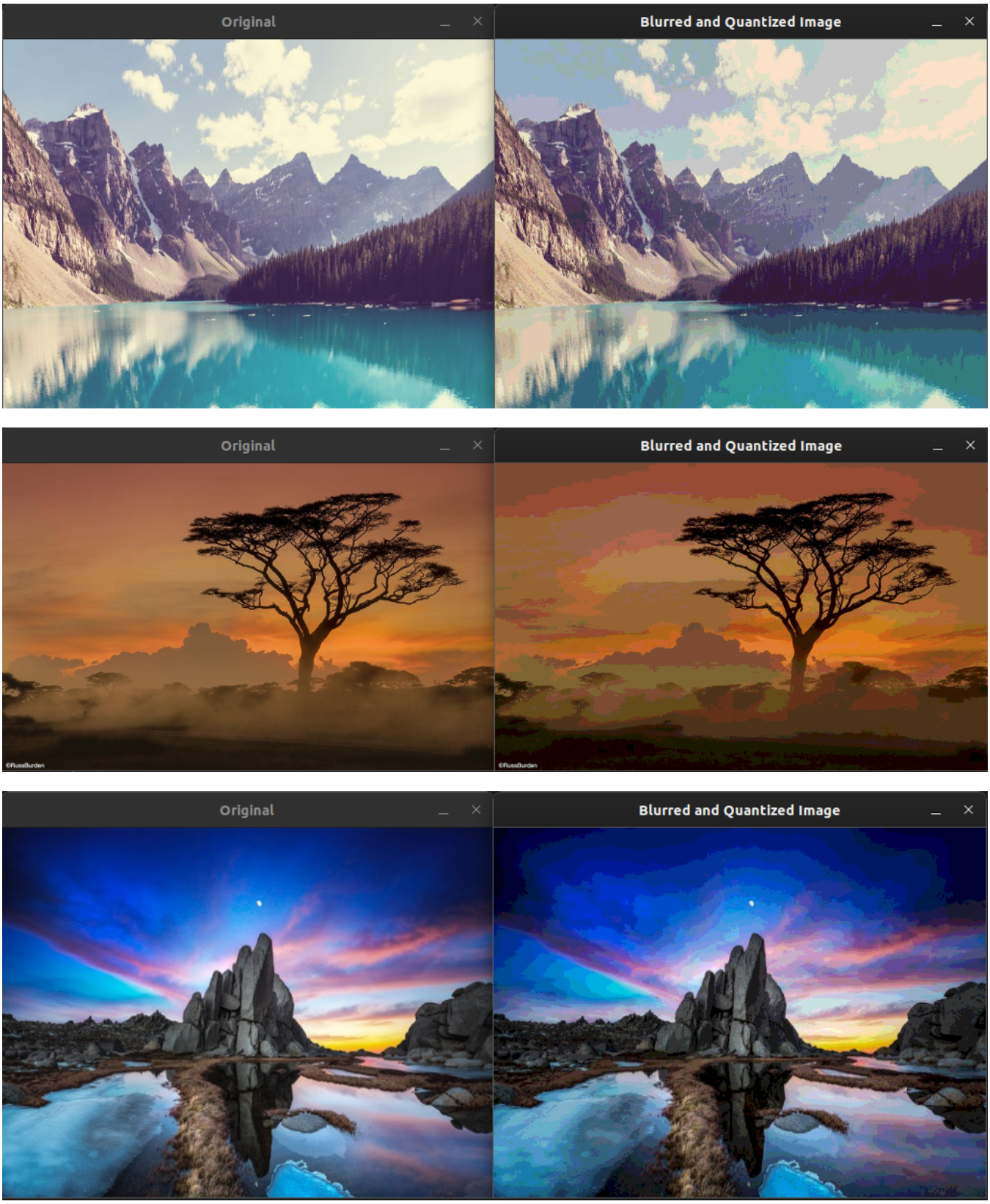
The below image shows the Gradient Magnitude Image generated with the help of 3x3 SobelX and SobelY separable filters. The pixel in Gradient Magnitude Image takes the value of the square root of the sum of squares of individual SobelX and SobelY filters resulting in an image that has strong edge values.



Required Image 5: Original and Blurred/Quantized Image

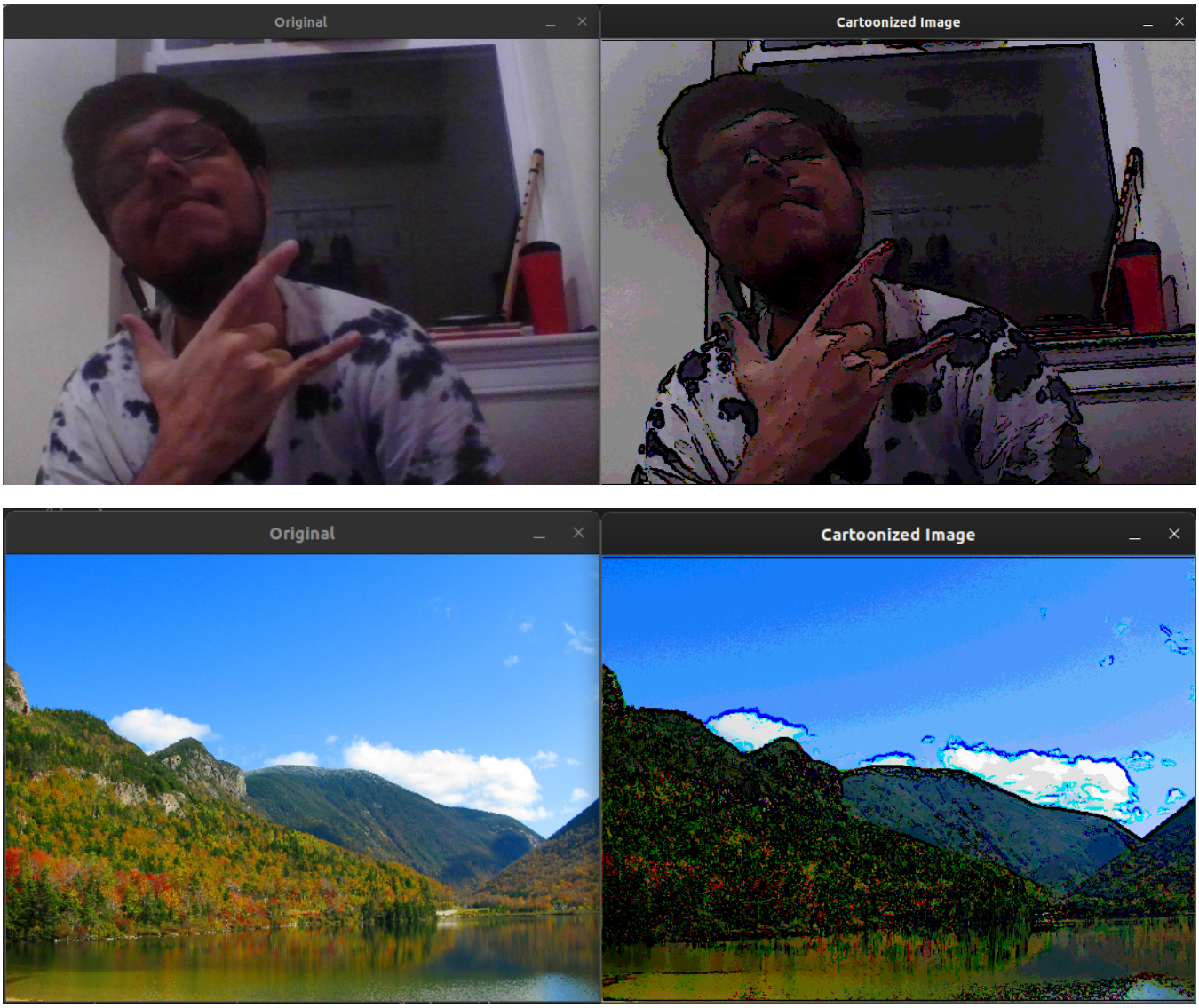
The following shows multiple examples of images that are blurred (5x5 Gaussian) and quantized (levels = 10) alongside their originals to make the difference in features apparent.





Required Image 6: Cartoonized Image

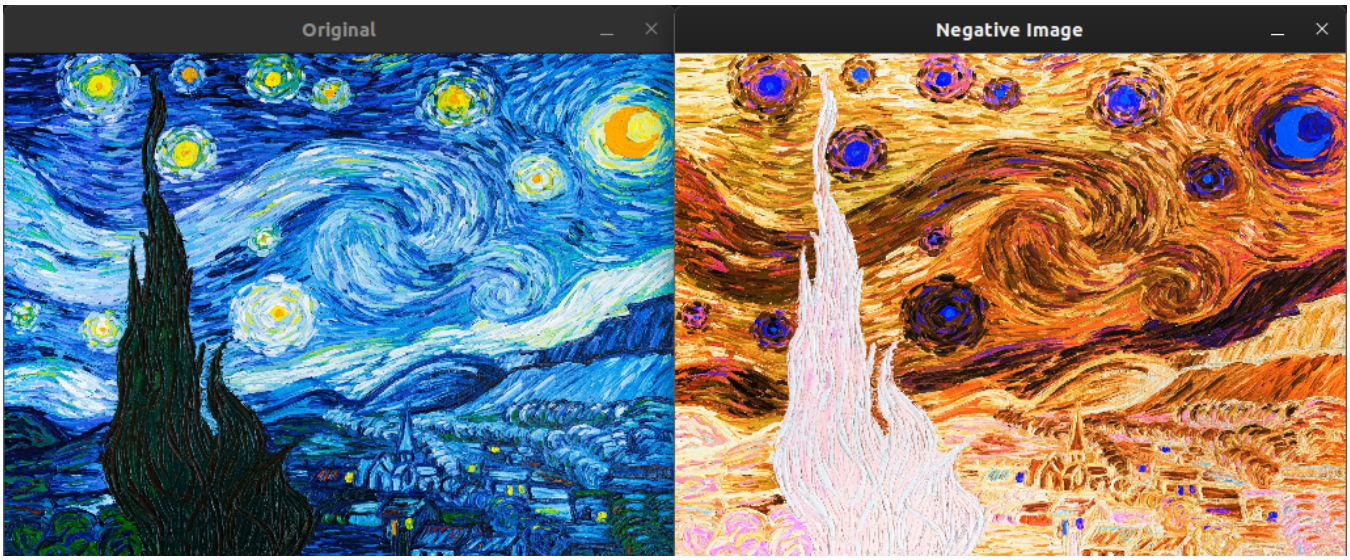
The following examples of images display side-by-side, the final cartoonized image along with their originals. The first example is saved from a video stream while the other is directly processed on the image. To create this effect, every function created before this point is utilised.

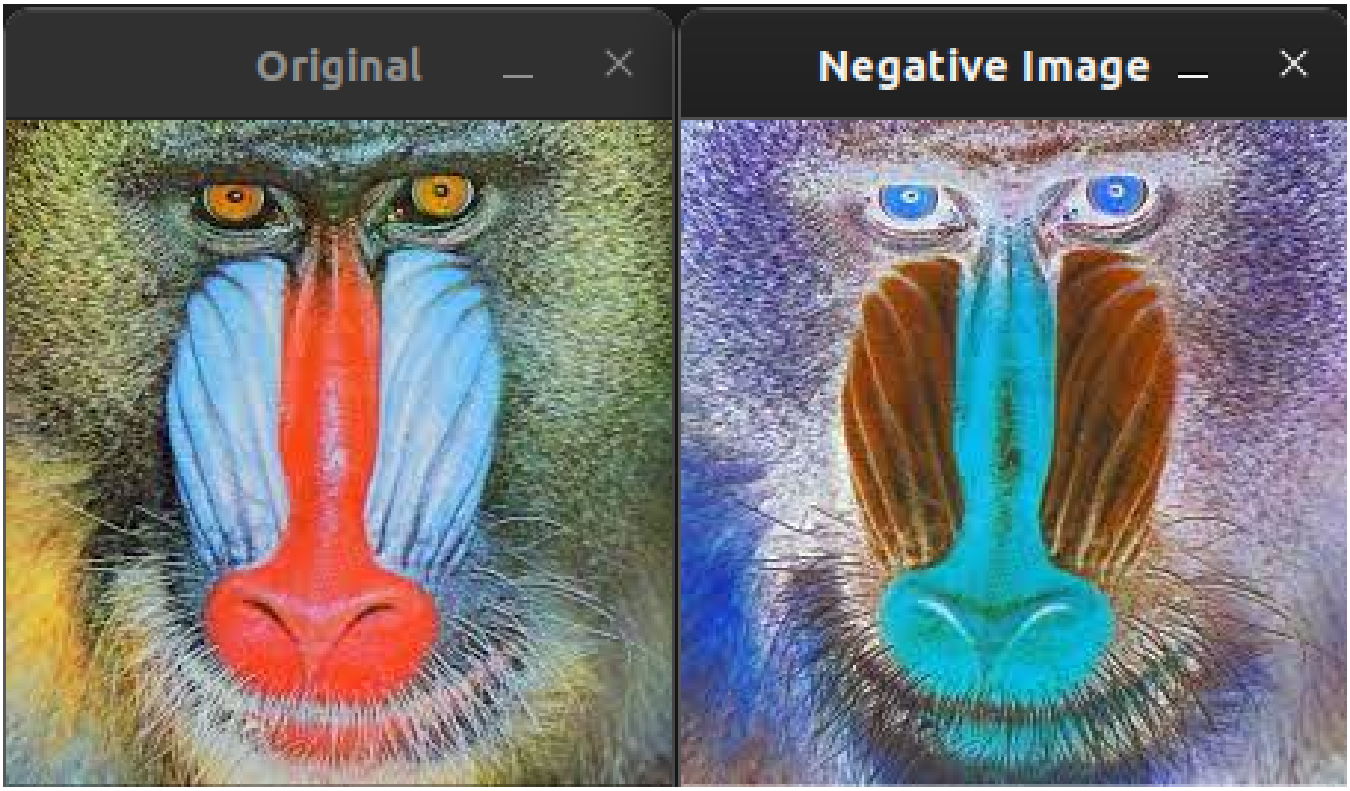


Required Image 7: Negative Image

The resultant images show the negative of the original images. It is a custom implementation wherein the value of every pixel is calculated as follows:

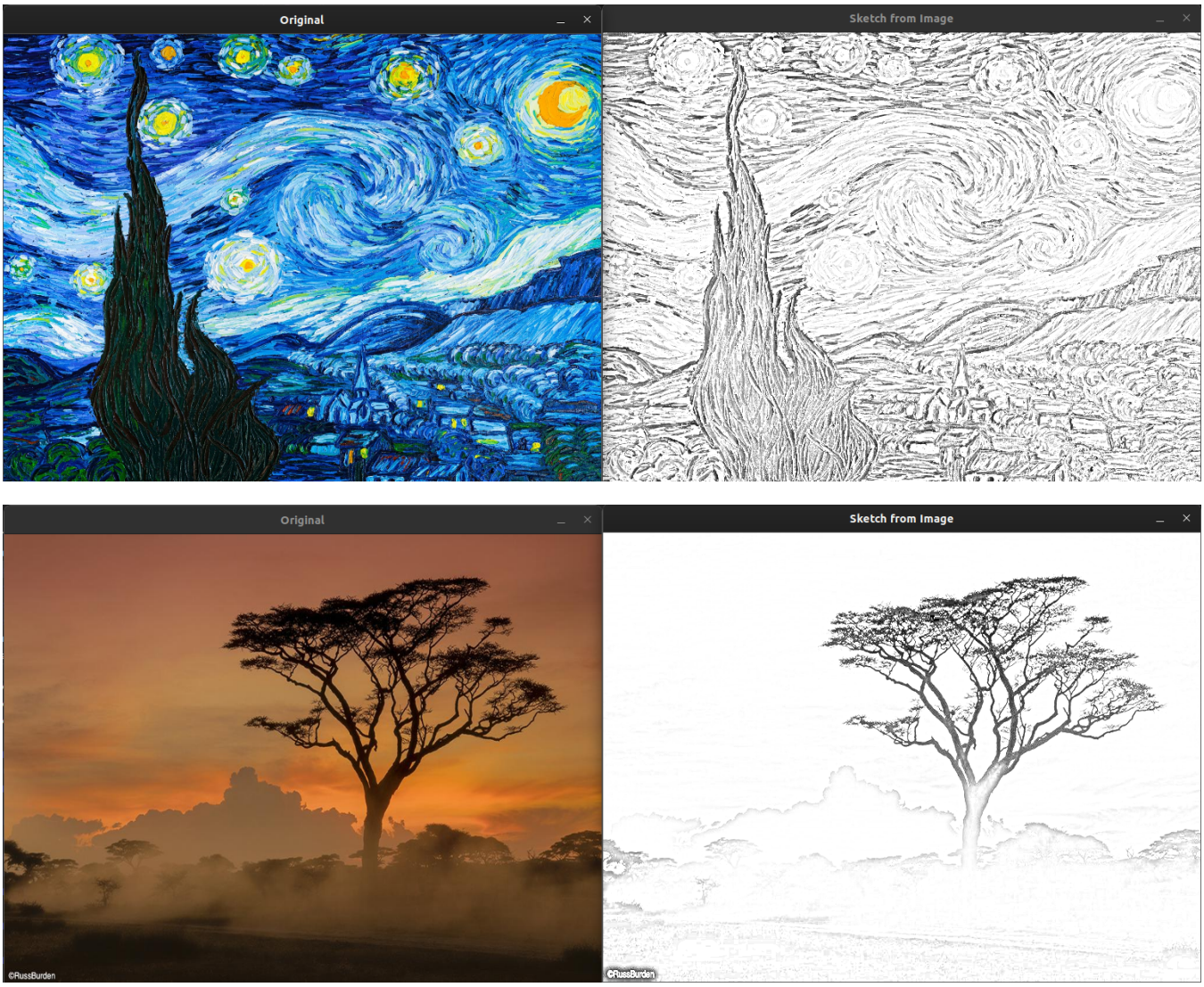
x = 255- x





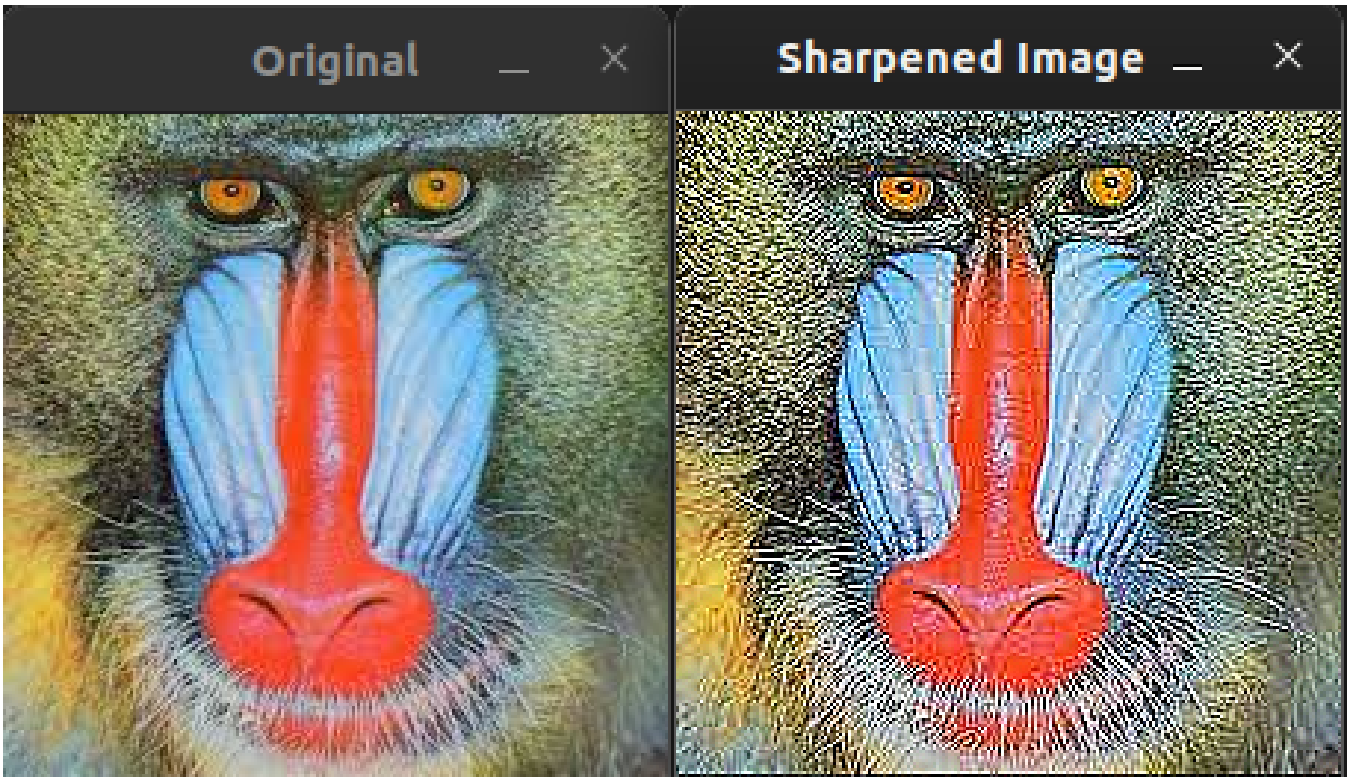
Extension 1: Sketch from Image

It is a custom implementation of a function that generates the sketch of an image. Builtin functions of OpenCV are also utilized within this function.



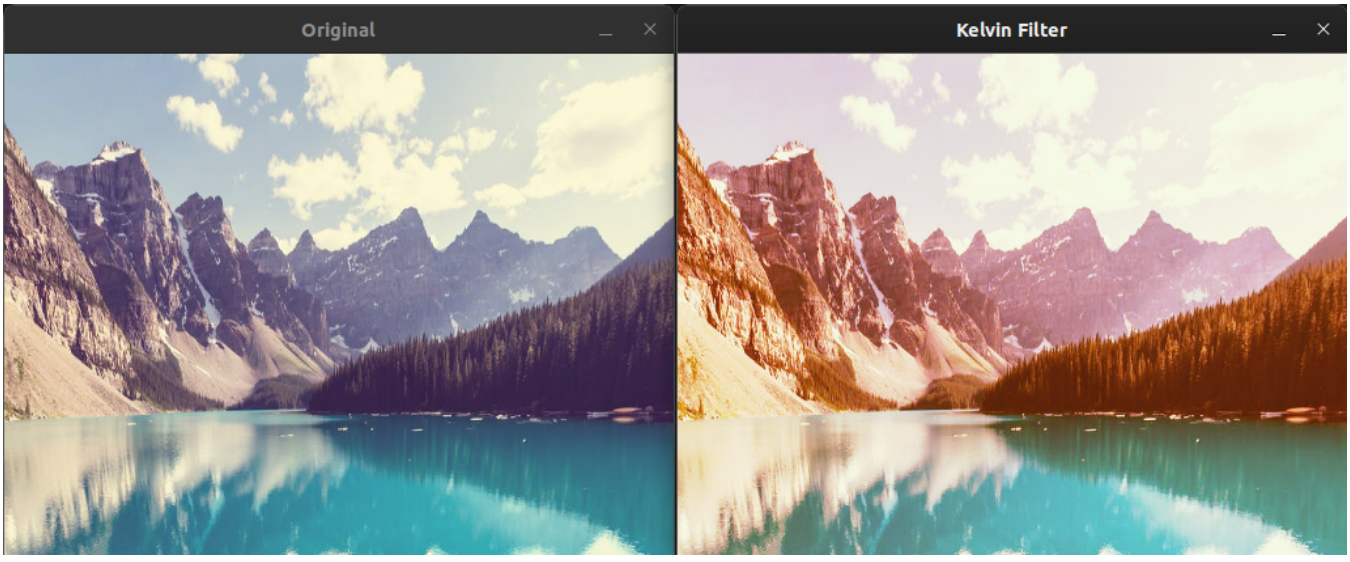
Extension 2: Sharpened Image

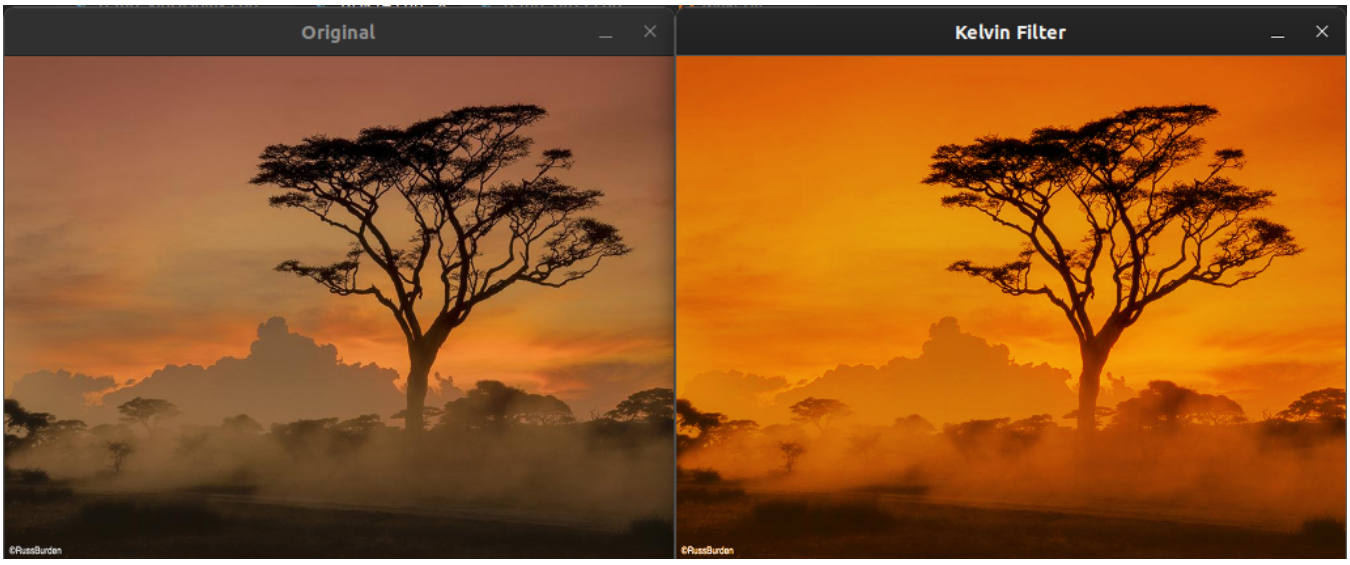
This function brings out the characteristics of the image by sharpening it. It uses the filter2D function of the OpenCV.



Extension 3: Kelvin Filtered Image

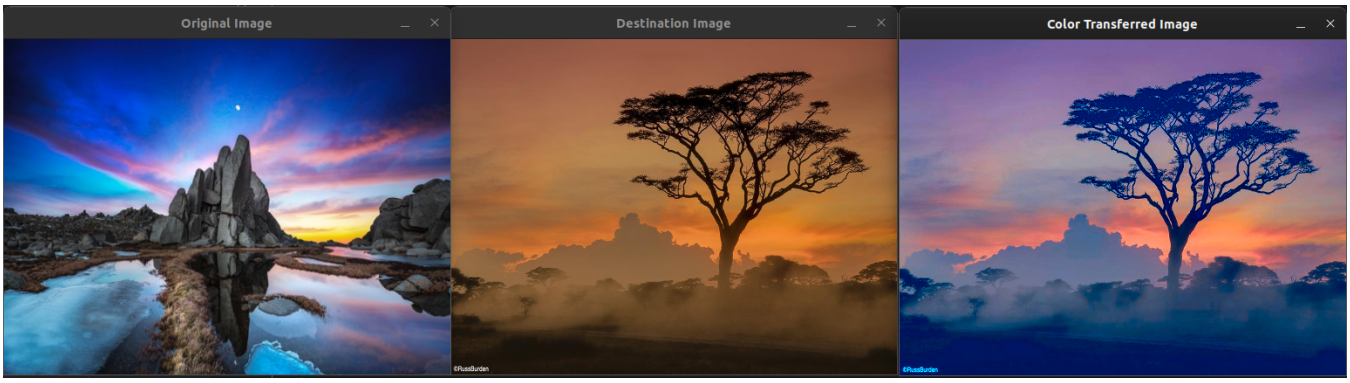
This is a custom implementation of the Kelvin filter which is also one of the popular Instagram filters!! This filter was a little challenging to implement and not straightforward like the previous ones. I got to discover the lookup tables and got to learn a lot about implementing and using them in different scenarios.





Extension 4: Color Transferred Image

For this extension, I have tried to take the characteristics of the first image and impose them on the characteristics of the second image resulting in the formation of the third image which looks like the colors of the first image have been brought to the second image and hence the name Color Transferred Image. This was a little challenging as well as all the operations being done in the LAB format which I was new to. It was a good learning process.



Reflection:

This is my first OpenCV and C++ project. I have worked with programming languages before but never have had created such an elaborate piece of code to accomplish tasks in a structured manner. Even though a little steep at first, I found the learning curve bending in my favour and have tried my best to write my code as elegantly and efficiently as possible. I was unfamiliar with almost all the things that I have accomplished in this project and now that I finish it, I find myself knowing the OpenCV package and C++ a little better making me more confident. Most of all, the extensions worked out great for me. I learnt about different image formats, different ways of manipulating images and also about lookup tables. It was a great way to learn on my own giving me the freedom to explore whatever I find interesting. Creating a structured code, debugging individual functions and then integrating everything with the video was overall a wonderful experience.

Acknowledgement:

Apart from googling various stuff, I would like to acknowledge Professor Maxwell's live coding class, the official documentation website of OpenCV, a lecture from the youtube channel Murtaza's workshop and the website: https://www.anishdubey.com from where I learnt a lot of basic stuff making me capable to complete this project. I would also like to acknowledge the contribution of my classmate Sumegha Singhania for being open to discussing various ideas and the fruitful brainstorming sessions.