Curve Tool in Computational Photography

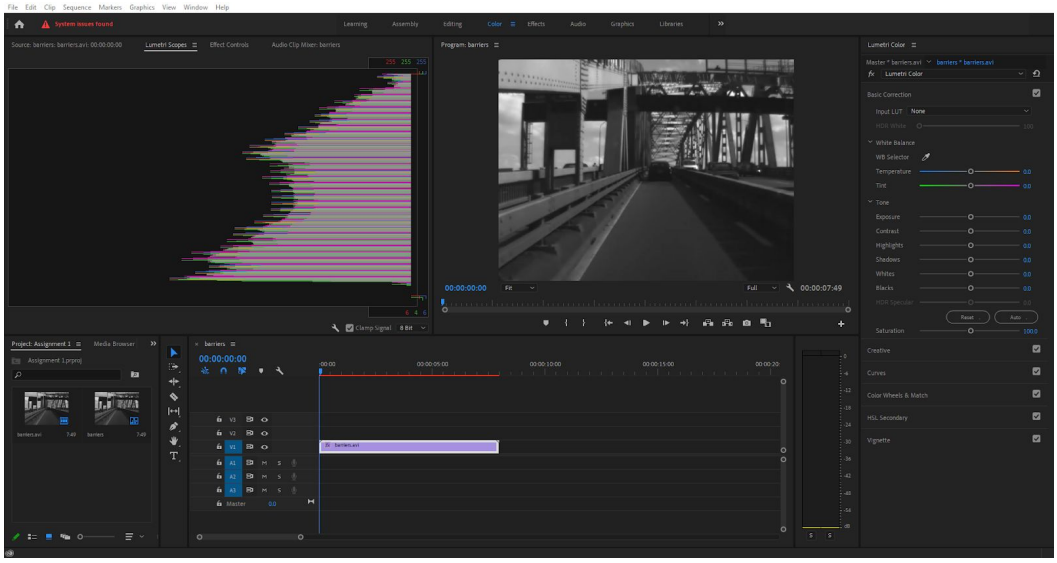
Prepared by
Aimee Valladares

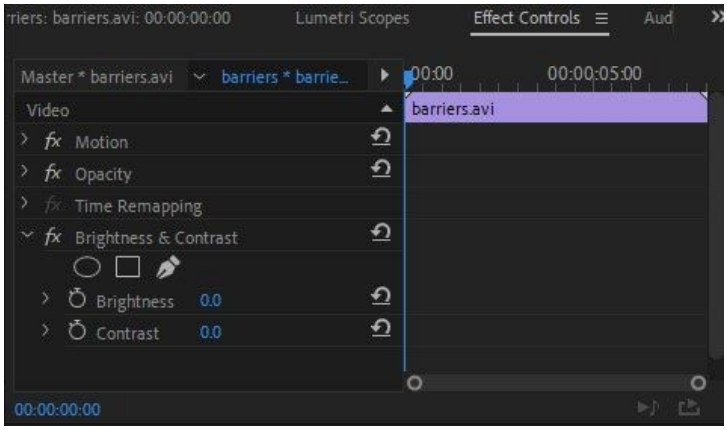
Computer Vision
CAP 4410

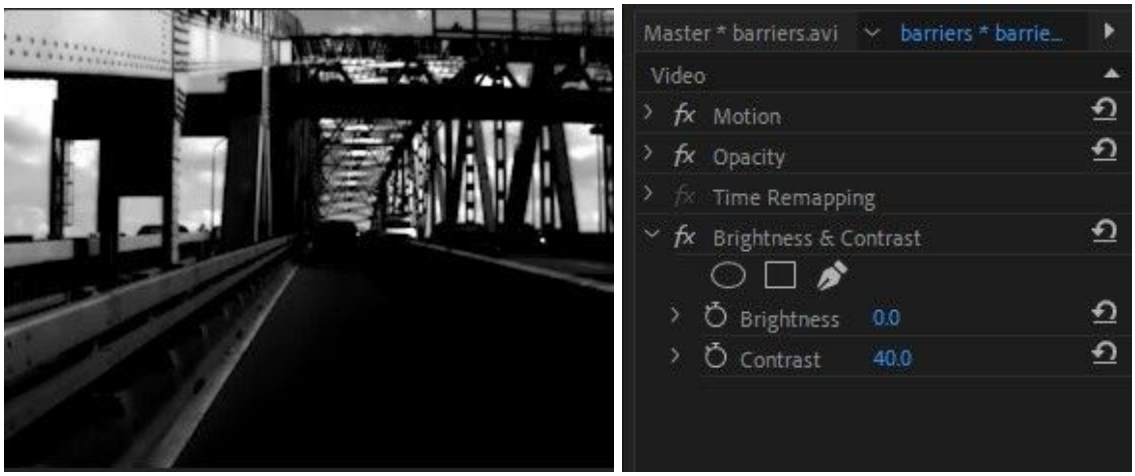
13 September 2020

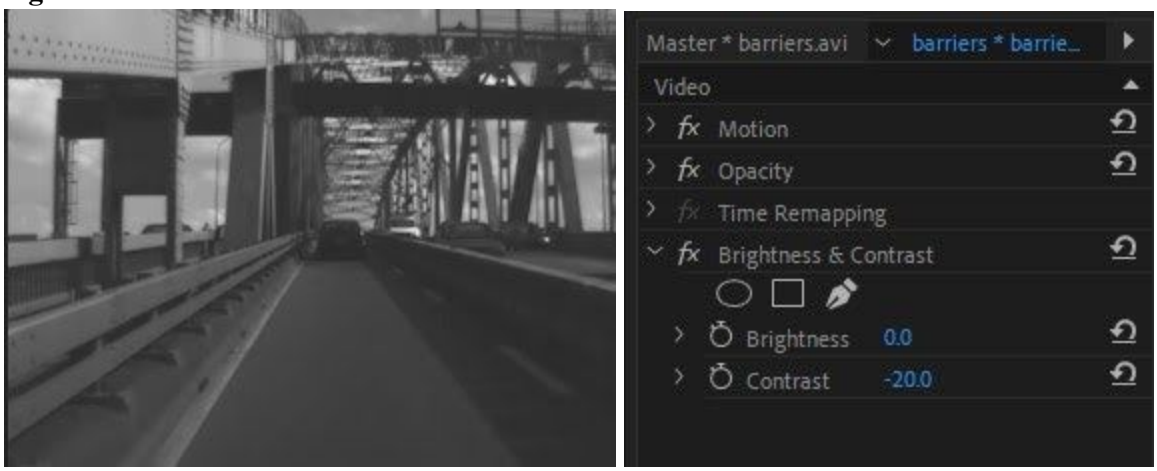
Table of Contents

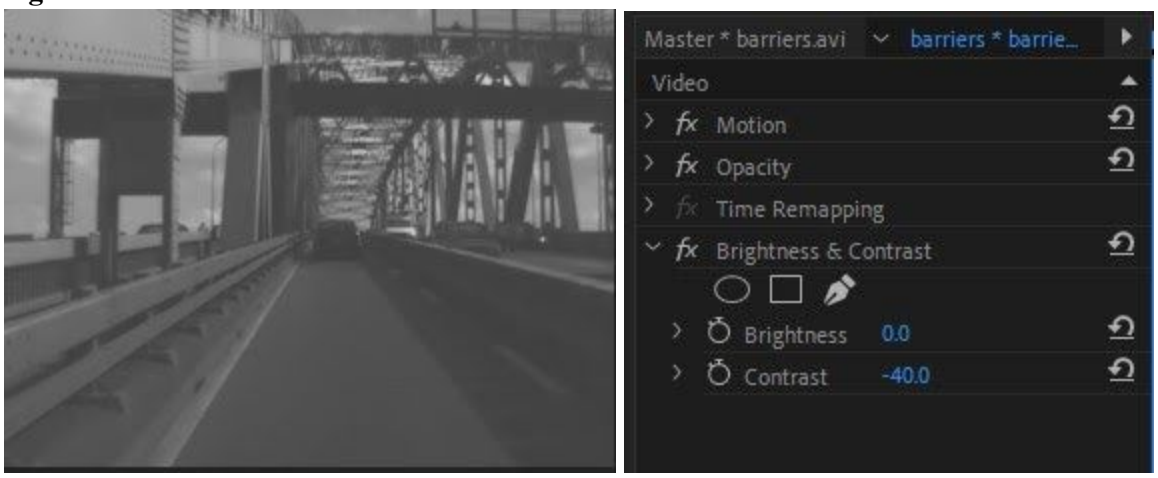
List of Figures

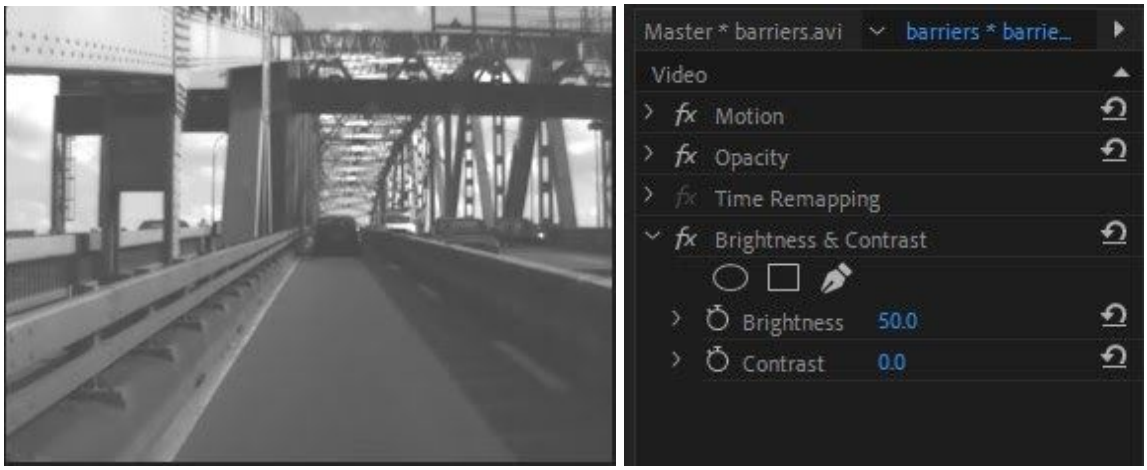


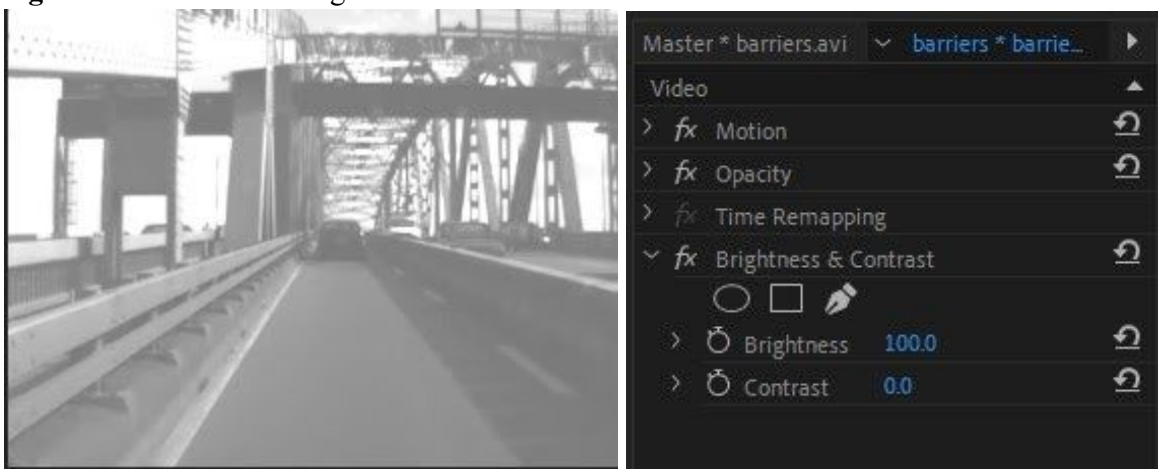


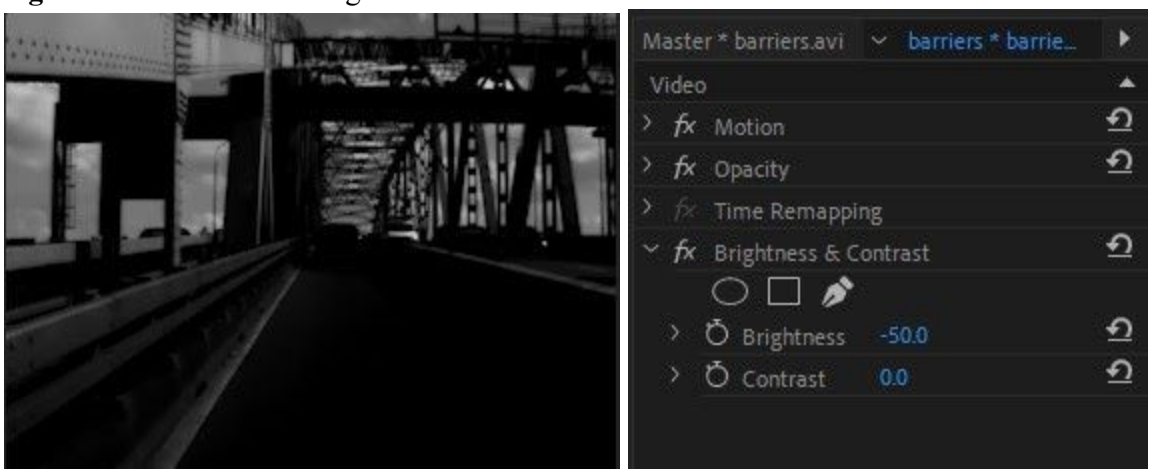


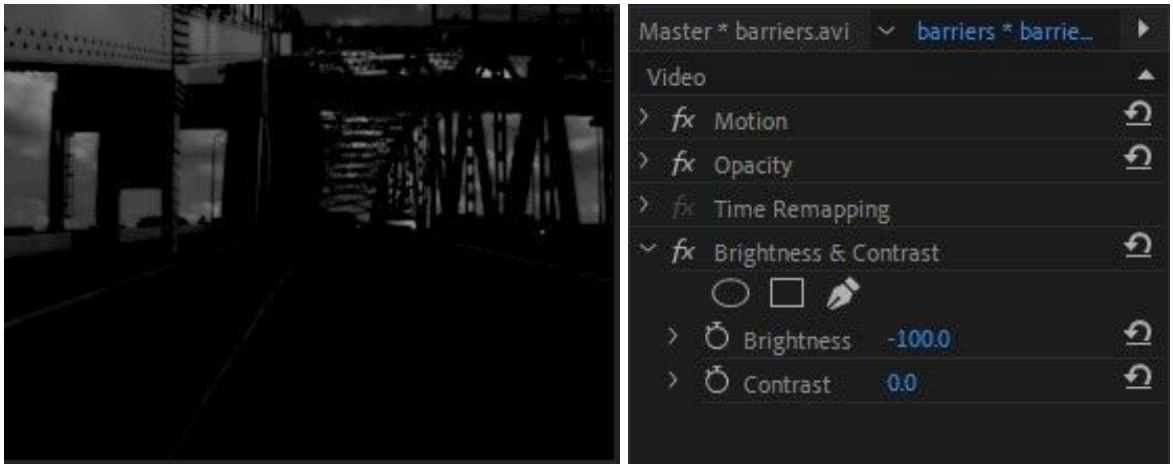


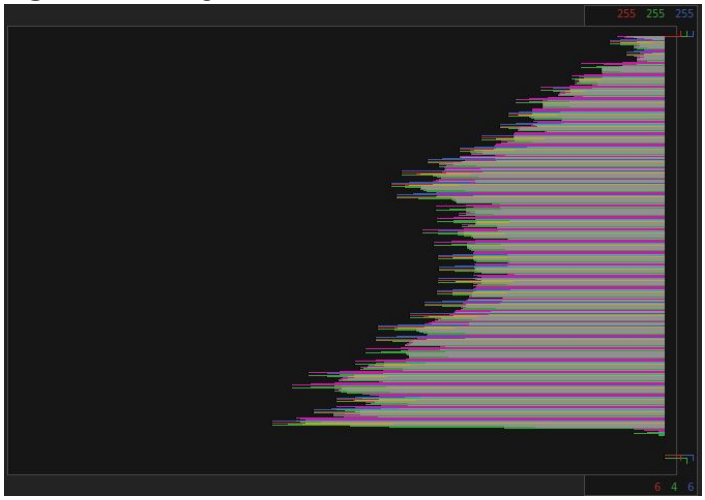


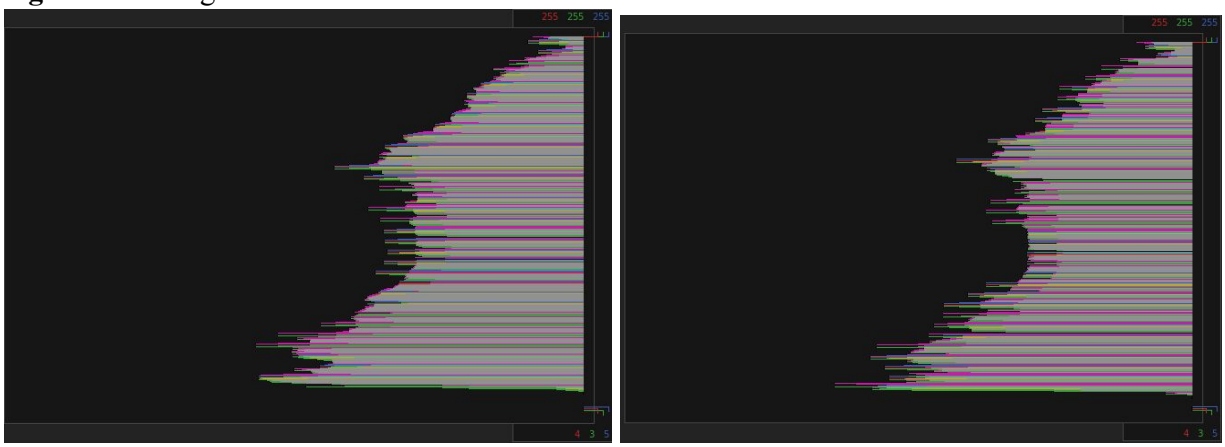


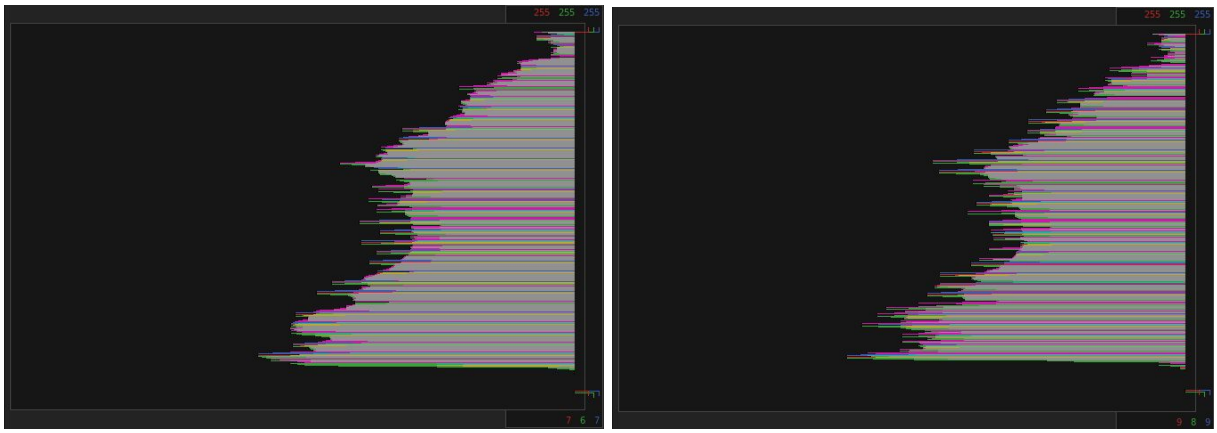


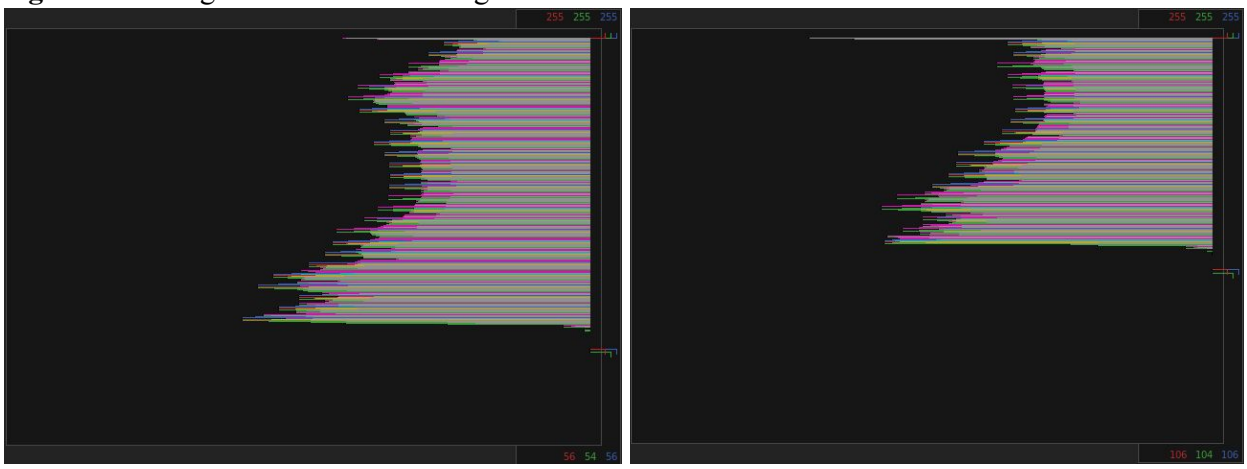


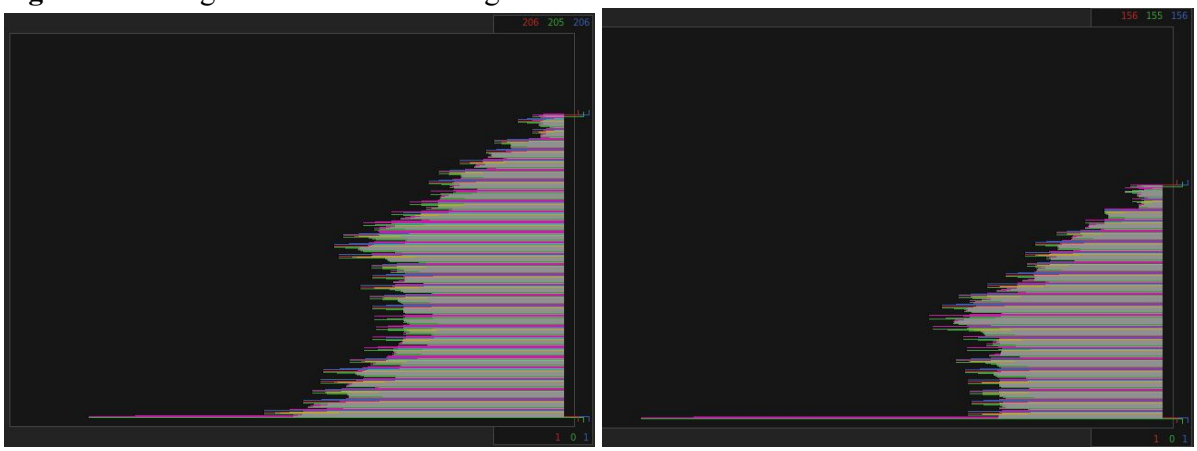












Curve Tool in Computational Photography

1.1 Introduction

Computational photography refers to computational image capture, processing, and manipulation techniques that are meant to enhance or extend the capabilities of digital photography. Computational photography has many applications like removing noise from an image, enhancing image features like saturation, and inpainting missing pixels [1]. An important component in computational photography are image histograms as they display the frequency of pixels in an image with a specific value. As a result, image histograms are useful tools in adjusting image features and understanding the data behind an image [2]. This project is meant to discuss another important tool that can adjust and enhance an image: the curve tool. A tool that is used in software like Adobe Photoshop and Photo Editor.

1.2 Project Description

The curve tool has the capability of adjusting and enhancing an image. In this project we are focusing on adjusting two image features: contrast and brightness. Contrast is the difference between maximum and minimum pixel intensity in an image [2]. Brightness is the amount of output by a source of light relative to the source we are comparing it to [3]. One process to change an image's brightness is to slide the histogram towards 0 (making the image darker) or towards 250 (making the image lighter). This is because the 0 value is a numerical representation of black and 250 is the numerical representation of white.

While a different process is used to change an image's contrast, this process is known as *histogram equalization*. Histogram equalization "quantifies the number of pixels for each intensity value considered" according to OpenCV documentation. Histogram equalization can adjust the contrast of an image by stretching out the intensity range [4]. The curve tool helps simplify these processes in software like Adobe Photoshop by acting on the histogram of the image.

The main objective of this is to implement a curve tool for video editing that controls the brightness or contrast of the video. This implementation will be done by creating an OpenCV project that will change an image's contrast or brightness. This program will illustrate the effects changing an image's contrast or brightness has on the original video histogram. These changes will be tested by comparing the results from the project with Premiere Pro.

1.3 Project Design

Before creating the OpenCV project in Visual Studio, it is important that the .avi video file is converted to an .mp4 file. This will be done through Premiere Pro as it is video editing software that will be used later for testing purposes. Once the video file is imported into a new project with a new sequence created in the bottom center panel. The video file can then be exported into the .mp4 file format.

The OpenCV code will comprise two C++ files. The first file will demonstrate the change in contrast of a video (this will be known as the ChangeContrast.cpp file). First, the file will delve into the OpenCV library and include the necessary files needed to run the program. The file will then read and load the video through its file path. Afterwards, an if conditional statement is used to help avoid any errors in running the program. Specifically, the conditional statement will create an exit or escape in case the video file could not be read or loaded. Then the windows are created to serve as a preview interface to help display the effect changing the video's contrast has on it. Once this is complete a while loop is used to help calculate the contrast change by having the file break the video into frames and call upon the convertTo function. The convertTo function allows the code to change the video's contrast by multiplying each pixel value with a constant that is either greater than one or is less than one but greater than zero. The constant represents the contrast adjustment value and it is inputted in the code as the alpha value [5]. Moreover, the while loop will have two if conditional statements to break the while loop. One if statement will stop the while loop once the video has ended while the other if statement will stop the while loop based on user input.

The second file will demonstrate the change in brightness of a video (this file will be known as the ChangeBrightness.cpp file). The code for this file is similar to the previous file but still contains a few significant differences. Like the first file, this fill will examine the OpenCV library and include the needed files, load the video, encompass a fail-safe conditional statement, create and open windows, and finally utilize a while loop to break the video into frames. However, while the code calls upon convertTo functions to calculate and alter the video's brightness, the calculation process is different. Specifically, the convertTo function changes the video's brightness by having the pixel values in the video either increase/decrease. In other words the contrast representing the brightness adjustment value will be either added to the video's pixel values (increasing brightness) or subtracted from the video's pixel values (decreasing brightness) [5]. Finally, the while loop will include two if conditional statements that can break the while loop.

The OpenCV project files can be seen in the section labelled "2.1 Project Code" under the Heading 2. OpenCV Project.

For testing purposes, Premiere Pro will be used to compare the results from the OpenCV project. This software will also help demonstrate the implementation of a curve tool for video editing and also illustrate how the video's image histograms' change. First, a project must be created with the test video imported. Once the video is imported create a new sequence in the bottom center panel. Then, in the top of the panel change the workspace to *Color*. In the Application panel, change to the *Lumetri Scopes* and right click on the panel to change the graphical representation to *Histogram*. Finally, in the bottom left-hand panel find the tab labelled *Effect*. Once this is done, there should be several folders. Under the folder labelled *Video Effects*, find a folder labelled *Color Correction*. Under this folder there should be several options, click the option labelled *Brightness & Contrast* and drag it onto the imported video file. Afterwards, in the panel containing the Lumetri Scopes switch to a tab labelled *Effect Controls*. Here is where the image's contrast and brightness can be changed [6]. Subsequently, the resulting workspace on Premiere Pro should be very similar to Figure 1.

Once the workspace is complete, the curve tool for the video's contrast and brightness can be changed with the histogram seen in the top left-hand side changing accordingly.

2. OpenCV Project

2.1 Program Code

The OpenCV Project was written in C++ with Visual Studio with the program consisting of two files. The program source code is shown below. The files are well-commented, with extended comments used to help distinguish between the two files and help explain the logic of the code [5] [7] [8] [9].

```
/* File Name: File 1 - ChangeContrast.cpp
 * Name: Aimee Valladares
 * Date of Submission: 9/15/2020
 * Purpose: Change a given video's contrast by either increasing or decreasing it
 */

//Include statements
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char** argv)
{
    // Reads the video file and loads it through file path
    VideoCapture cap("C:/Users/valla/Documents/Courses/Computer Vision/Assignments/barrier.mp4");

    // If statement meant to check for possible failures
    if (cap.isOpened() == false)
    {
        // If the video retrieval is not successful, exit the program
        cout << "Cannot open the video file" << endl;
        cin.get(); // Wait for user to press any key to exit
        return -1;
    }

    // Defining window names (variables) for the above videos
    String windowNameOriginalVideo = "Original Video";
    String windowNameContrastHigh20 = "Contrast Increased by 20";
    String windowNameContrastHigh40 = "Contrast Increased by 40";
    String windowNameContrastLow0_05 = "Contrast Decreased by 0.05";
    String windowNameContrastLow0_025 = "Contrast Decreased by 0.025";

    // Create and open windows for the above window names
    namedWindow(windowNameOriginalVideo, WINDOW_NORMAL);
    namedWindow(windowNameContrastHigh20, WINDOW_NORMAL);
    namedWindow(windowNameContrastHigh40, WINDOW_NORMAL);
    namedWindow(windowNameContrastLow0_05, WINDOW_NORMAL);
    namedWindow(windowNameContrastLow0_025, WINDOW_NORMAL);

    // A while loop to help calculate the change in contrast for the given video by
    // breaking it down to frames while also adding breaks to avoid errors like an
    // infinite loop.
    while (true)
```

```
    {
        // Read every new frame from the video
        Mat frame;
        bool bSuccess = cap.read(frame);

        // If statement to break the while loop once the video has ended
        if (bSuccess == false)
        {
            cout << "The video has ended" << endl;
            break;
        }

        // Calculate the contrast adjustment in the given video by using convertTo function
        // convertTo(Output image, Type of output image, alpha, beta)
        // Alpha - the contrast adjustment value
        Mat frameContrastHigh20;
        frame.convertTo(frameContrastHigh20, -1, 20, 0); // Increase the contrast by 20
        Mat frameContrastHigh40;
        frame.convertTo(frameContrastHigh40, -1, 40, 0); // Increase the contrast by 40
        Mat frameContrastLow0_05;
        frame.convertTo(frameContrastLow0_05, -1, 0.05, 0); // Decrease the contrast by 0.05
        Mat frameContrastLow0_025;
        frame.convertTo(frameContrastLow0_025, -1, 0.025, 0); // Decrease the contrast by 0.025

        //Show above frames inside the created windows.
        imshow(windowNameOriginalVideo, frame);
        imshow(windowNameContrastHigh20, frameContrastHigh20);
        imshow(windowNameContrastHigh40, frameContrastHigh40);
        imshow(windowNameContrastLow0_05, frameContrastLow0_05);
        imshow(windowNameContrastLow0_025, frameContrastLow0_025);

        // A break included based on user input and another way to stop the program from running
        // Break the while loop when 'Esc' key is pressed
        // Continue the loop when any other key is pressed or if any key is not pressed within 10
ms
        if (waitKey(10) == 27) //Wait for 10 ms until any key stroke
        {
            cout << "Esc key is pressed by user. Stopping the video" << endl;
            break;
        }
    }

    return 0;
}

/* File Name: File 2 - ChangeBrightness.cpp
 * Name: Aimee Valladares
 * Date of Submission: 9/15/2020
 * Purpose: Change a given video's brightness by either increasing or decreasing it
 */

//Include statements
#include "stdafx.h"
#include <opencv2/opencv.hpp>
#include "opencv2/imgcodecs.hpp"
#include "opencv2/highgui.hpp"
#include <iostream>

using namespace cv;
using namespace std;
```

```
int main(int argc, char** argv)
{
    // Reads the video file and loads it through file path
    VideoCapture cap("C:/Users/valla/Documents/Courses/Computer Vision/Assignments/barries.mp4");

    // If statement meant to check for possible failures
    if (cap.isOpened() == false)
    {
        // If the video retrieval is not successful, exit the program
        cout << "Cannot open the video file" << endl;
        cin.get(); // Wait for user to press any key to exit
        return -1;
    }

    // Defining window names (variable) for the above videos
    String windowNameOriginalVideo = "Original Video";
    String windowNameBrightnessHigh50 = "Brightness Increased by 50";
    String windowNameWithBrightnessHigh100 = "Brightness Increased by 100";
    String windowNameBrightnessLow50 = "Brightness Decreased by 50";
    String windowNameBrightnessLow100 = "Brightness Decreased by 100";

    // Create and open windows for the above window names
    namedWindow(windowNameOriginalVideo, WINDOW_NORMAL);
    namedWindow(windowNameBrightnessHigh50, WINDOW_NORMAL);
    namedWindow(windowNameWithBrightnessHigh100, WINDOW_NORMAL);
    namedWindow(windowNameBrightnessLow50, WINDOW_NORMAL);
    namedWindow(windowNameBrightnessLow100, WINDOW_NORMAL);

    // A while loop to help calculate the change in brightness for the given video by
    // breaking it down to frames while also adding breaks to avoid possible errors like
    // an infinite loop
    while (true)
    {
        //Read a new frame from the video
        Mat frame;
        bool bSuccess = cap.read(frame);

        // If statement to break the while loop once the video has ended
        if (bSuccess == false)
        {
            cout << "The video has ended" << endl;
            break;
        }

        // Calculate the brightness adjustment in the given video by using convertTo function
        // convertTo(Output image, Type of output image, alpha, beta)
        // Beta - the brightness adjustment value
        Mat frameBrighnessHigh50;
        frame.convertTo(frameBrighnessHigh50, -1, 1, 50); // Increase the brightness by 50
        Mat frameBrighnessHigh100;
        frame.convertTo(frameBrighnessHigh100, -1, 1, 100); // Increase the brightness by 100
        Mat frameBrighnessLow50;
        frame.convertTo(frameBrighnessLow50, -1, 1, -50); // Decrease the brightness by 50
        Mat frameBrighnessLow100;
        frame.convertTo(frameBrighnessLow100, -1, 1, -100); // Decrease the brightness by 100

        //Show above frames inside the created windows.
        imshow(windowNameOriginalVideo, frame);
        imshow(windowNameBrightnessHigh50, frameBrighnessHigh50);
        imshow(windowNameWithBrightnessHigh100, frameBrighnessHigh100);
        imshow(windowNameBrightnessLow50, frameBrighnessLow50);
```

```
        imshow(windowNameBrightnessLow100, frameBrighnessLow100);

        // A break included based on user input and another way to stop the program from running
        // Break the while loop when 'Esc' key is pressed
        // Continue the loop when any other key is pressed or if any key is not pressed within 10
ms
        if (waitKey(10) == 27) // Wait for 10 ms until any key stroke
        {
            cout << "Esc key is pressed by user. Stopping the video" << endl;
            break;
        }
    }

    return 0;
}
```

2.2 Test Plan

To properly demonstrate the OpenCV project code accuracy in changing the image features we will compare the project results with a control. The designated control is the use of Premiere Pro as it is a credible software with a curve tool in video editing that can adjust many different video features. For this project, we will focus on the curve tool meant to adjust a video's contrast and/or brightness and how these adjustments affect the video's histograms.

Figure 2 demonstrates the first frame of the video without any adjustments made to the image's contrast or brightness. Following the process of setting up the workspace and finding the adjustment feature previously stated in the "1.3 Program Design", apply the Brightness/Contrast effect and in the panel on the top left-hand corner the brightness and contrast can be changed as seen in Figure 3.

The first C++ file (known as the contrast file) will generate 6 different videos. The 3 videos in the top row demonstrate the contrast increase. In order from left to right, the program will generate the original video, the video with a contrast increase of 20, and the video with a contrast increase of 40. The 3 videos in the bottom row demonstrate the contrast decrease. In order from left to right, the program will generate the original video, the video with a contrast decrease of 0.05, and the video with a contrast decrease of 0.025.

The second C++ file (known as the brightness file) will also generate 6 different videos. The 3 videos in the top row demonstrate the brightness increase. In order from left to right, the program will generate the original video, the video with a brightness increase of 50, and the video with a brightness increase of 100. The 3 videos in the bottom row demonstrate the brightness decrease. In order from left to right, the program will generate the original video, the video with a brightness decrease of -50, and the video with a brightness decrease of -100.

To verify the result of the OpenCV project, we will compare the outputs of the two C++ files by making the same adjustments to the video in Premiere Pro.

To properly illustrate the implementation of the curve tool in video editing, we will adjust the brightness and contrast in the video by comparing the changes in the video's histogram.

Specifically, we will try to verify if the histogram will slide when adjusting the image's brightness as well as verify if histogram equalization occurs when adjusting the image's contrast.

2.3 Test Results

The OpenCV project was operational and was seemingly accurate in depicting the effects of changing an image's contrast or brightness as it produced similar results to Premiere Pro. This is illustrated by the differences seen between Figure 2 and Figures 4 – 11.

When comparing the original video seen in Figure 2 with the videos that depict changes in the video's contrast (Figures 4 - 7), changing a video's contrast affects the image's blurriness or clarity. This is because changing the contrast of a video changes the difference between the maximum and minimum pixel values essentially making the lighter parts lighter and the darker parts darker.

Whereas when comparing the original video seen in Figure 2 with the images that depict changes in brightness (Figures 8 - 11), the overall video changes. This is because the changes in brightness primarily affect the videos either maximum or minimum which are seen by a sliding histogram. Specifically, when the brightness increases the minimum for the histogram shifts toward 250. Whereas when the brightness decreases the maximum for the histogram shifts toward 0.

These changes are further emphasized by the implementation of the curve tool when video editing in Premiere Pro. In Figure 12, it demonstrates the histogram for the unedited video. However, in Figures 13 and 14, the histogram changes in response to increasing and decreasing the video's contrast. Figure 13 illustrates that the histogram will continue to "stretch" as the contrast increases. While Figure 14 illustrates that the histogram will continue to "squish" as the contrast decreases. These figures display histogram equalization.

Moreover, in Figures 15 and 16, the histogram slides in accordance to the video's brightness either increasing or decreasing. In Figure 15, the histogram will continually slide toward 250 (the numerical representation of white) as the brightness increases. This results in the overall video becoming lighter. Whereas, in Figure 16, the histogram will continually slide toward 0 (the numerical representation of black) as the brightness decreases.

2.4 Conclusion

The curve tool is an important tool for understanding how the pixel data and the image histogram changes when adjustments are made to an image's features. Consequently, in this project we learned that changing an image's contrast will change the image histogram as well by going through a process known as histogram equalization. Moreover, a similar change to an image's histogram will occur when changing an image's brightness. Though this process results in the histogram being slid towards 0 or 250.

In the future, I would like to make several improvements that would help improve the accuracy and efficiency of the OpenCV project. One possible improvement I would make to the OpenCV project is to be able to map the video's histogram to further demonstrate how the histogram changes when the video's contrast/brightness is changed. Moreover, another possible improvement I would make

to the OpenCV project is to have the program successfully run using the video in the original .AVI format. However, due to time constraints I was not able to get the program to properly implement the Fourcc video codecs as the video continued to be corrupted when the program tried to run[10]. Another possible improvement to the OpenCV project is to have more user input involved making the interface more user friendly. For example, the project could allow the user to input an image of her choosing like inputting any possible frame of the video, not just the first frame. Another instance is to allow the user to input the contrast or brightness adjustment value themselves.

References

[1] Abid, Muhammad. Lecture 1 Slides, CANVAS *Concise Computer Vision*.

[2] Abid, Muhammad. Lecture 2 Slides, CANVAS *Concise Computer Vision*.

[3] "Brightness and Contrast." *Tutorialspoint*, www.tutorialspoint.com/dip/brightness_and_contrast.htm.

[4] "3.15 Histogram Equalization." *OpenCV Tutorials*, pp. 263–269. OpenCV, 31 Dec. 2019, docs.opencv.org/2.4/opencv_tutorials.pdf.

[5] "2.5 Changing the contrast and brightness of an image!" *OpenCV Tutorials*, pp.152-156. OpenCV, 31 Dec. 2019, docs.opencv.org/2.4/opencv_tutorials.pdf.

[6] *Use and Customize Workspaces in Premiere Pro*, helpx.adobe.com/premiere-pro/using/workspaces.html.

[7] Fernando, Shermal Ruwantha. *Load & Display Image*, www.opencv-srf.com/2017/11/load-and-display-image.html.

[8] Fernando, Shermal Ruwantha. *Change Contrast*, https://www.opencv-srf.com/2018/02/change-contrast-of-images-and-videos.html

[9] Fernando, Shermal Ruwantha. *Change Brightness*, https://www.opencv-srf.com/2018/02/change-brightness-of-images-and-videos.html

[10] "What Is a FOURCC?" *Www.FOURCC.org - Video Codecs and Pixel Formats*, www.fourcc.org/fourcc.php.