Program 3

Due Nov 4, 2019 by 1:15pm **Points** 20 **Submitting** a file upload **File Types** zip

In this assignment, you will use OpenCV to modify an image similar to a "green screen" technique. Pixels of a selected color will be replaced with pixels from a second image. For further information on the general idea see: **http://en.wikipedia.org/wiki/Chroma_key.** **(http://en.wikipedia.org/wiki/Chroma_key)**

Install OpenCV

These instructions are for Windows 10 and Visual Studio 2017/2019.

1. Go to: **https://sourceforge.net/projects/opencvlibrary/files/opencv-win/** **(https://sourceforge.net/projects/opencvlibrary/files/opencv-win/)**
2. Download: opencv-4.1.2-vc14_vc15.exe (the big green button)
3. Execute the downloaded file. Extracting to your desktop is ok, but (probably) move the folder to be C:\opencv
4. Do one of the following:
   a. Update the **system** PATH variable (not the user variable) to include C:\opencv\build\x64\vc15\bin (ideal, but must be admin)
   b. Copy the dll files in C:\ opencv\build\x64\vc15\bin to your project directory. This should be in the same directory as the executable (x64/Debug). Note that there may be multiple Debug folders. Make sure you copy it to the correctly location. (You may not be able to do this until you have compiled a program and generated an executable.)
5. Create a Visual C++ 2017/2019 project.
6. Change your configuration to compile for x64 (not x86). In my Visual Studio, this is an option on the drop down on the standard toolbar. If you don't have this option, then click the "Configuration Manager" option. Under Active solution platform, click "New". Under "Type of select the new platform", click "x64" and then "OK".
7. Download the **Project Property Sheet** **(https://canvas.uw.edu/courses/1332532/files/59112615/download?download_frd=1)** and put it in your project folder.
8. Go to View -> Other Windows -> Property Manager. (May need to turn on expert settings.)
9. Click the Add Existing Project Property Sheet icon in Property Manager and add the downloaded property sheet.
10. Download the **sample code** **(https://canvas.uw.edu/courses/1332532/files/59123212/download?download_frd=1)** and images and put them in your project folder.
11. Add the code to your project, compile and run. You can use this **test3.jpg** **(https://canvas.uw.edu/courses/1332532/files/58262733/download?download_frd=1)** .

Compute a color histogram

1. Read two images from the disk. (Use the default directory, which is the same directory that the source code is in.) The images should be called “foreground.jpg” and “background.jpg”. Here are two examples: **foreground.jpg** (https://canvas.uw.edu/courses/1332532/files/58262731/download?download_frd=1) , **background.jpg** (https://canvas.uw.edu/courses/1332532/files/58262732/download?download_frd=1)

2. Use a **color histogram** (https://en.wikipedia.org/wiki/Color_histogram) to find the most common color in the foreground image. The histogram should be a three-dimensional matrix of integers. (Note: don’t use the OpenCV methods for creating histograms.)

```
// create an array of the histogram dimensions
// size is a constant - the # of buckets in each dimension
int dims[] = {size, size, size};
// create 3D histogram of integers initialized to zero
Mat hist(3, dims, CV_32S, Scalar::all(0));
```

To create the histogram, loop through the foreground image and assign each pixel to a histogram bucket. That bucket should be incremented by one. To decide which bucket to increment, you divide the color value by (256/size):

```
int bucketSize = 256/size;
int r = red / bucketSize;
int g = green / bucketSize;
int b = blue / bucketSize;
```

I have found that size = 4 often works well. Use this value for this assignment.

3. Find the histogram bin with the most “votes” by looping over all three dimensions. If the bin with the most votes is [r, g, b], then the most common color is approximately:

```
int cRed = r * bucketSize + bucketSize/2;
int cGreen = g * bucketSize + bucketSize/2;
int cBlue = b * bucketSize + bucketSize/2;
```

Create the overlay output

1. Replace every pixel in the foreground image that is close to the most common color (no more than bucketSize away in all three color bands) with the corresponding pixel from the background image (same row and column, unless the background image is too small). If the background image is too small, start over from the start of the background image. (This can be accomplished by taking the foreground row modulo the number of rows in the background image and similarly for columns.)
2. Display the resulting image on the screen and save it to “overlay.jpg”. Here is **my result** (https://canvas.uw.edu/courses/1332532/files/58262736/download?download_frd=1) . Note that large images are not shrunk by default by OpenCV. Use code similar to this to resize a window:

```
    namedWindow("output", WINDOW_NORMAL);
    resizeWindow("output", output.cols / 8, output.rows / 8);
    imshow("output", output);
```

Use OpenCV methods

Modify the background image by flipping it horizontally, converting it to greyscale, blurring it, and detecting edges (each of these is a call to an OpenCV method – flip, cvtColor, GaussianBlur, Canny). Documentation on the methods can be found on the OpenCV site: **https://docs.opencv.org/master/ (https://docs.opencv.org/master/)** . Display the image to the screen and write the result as “output.jpg”. Here is **my output (https://canvas.uw.edu/courses/1332532/files/58262734/download?download_frd=1)** .

- For cvtColor, use COLOR_BGR2GRAY as the color code to convert from color to grayscale.
- For blurring, use Size(7, 7) and sigmaX = sigmaY = 2.0. (Note that Size is an OpenCV type.)
- For edge detection (the Canny method), use threshold1 = 20 and threshold2 = 60.
- For these methods, you will need to #include <opencv2/imgproc.hpp>

Explore OpenCV methods

Examine the OpenCV documentation some and find three additional operations to apply to the (original) image to create an interesting output. Display the image to the screen and write the result as "myoutput.jpg".