Lab 5

Comp 11 - Summer Session — Instagram Filter

5.1 Description

In this lab we will implement a function to apply a color filter to images. Color filters are used in programs such as Photoshop or Instagram to post process images to get different effects.

Our objectives are the following:

- First, understand the format in which an image is stored (read the color tutorial below).
- A function which loads an image is provided below.
- Your job will be to write a function that writes a new file and applies a grayscale filter to the image.

5.2 Color Tutorial

There are many different image file formats, such as .png, jpg, or .bmp that you may have encountered. We are going to work with what is known as a .ppm image.

5.2.1 Pixel

Images are made of individual pixels. An image that is 4 pixels wide, and 4 pixels high, has 16 total pixels (width * height).

Individual pixels (on computers) have Red, Green, and Blue color components that make up the final color that is output to your screen. Sometimes we abbreviate this color scheme as RGB.

Each color component(the red, green, and blue) has an intensity associated with it. This intensity value ranges from a value of 0-255. The higher the value, the more of that color for example.

So as a reminder, a pixel has three components of (R,G,B). A red pixel would have values of (255,0,0), noting that there is a 0 in the green position, and a 0 in the blue position, and a maximum red intensity. This means that 1 pixel will be fully red.

5.2.2 PPM Format

A .ppm images holds a series of values for each pixel. Here is what the raw data looks like. Note: If you open it up in a text editor, you will also see additional information regarding the files

```
1  // Here are the raw pixels with color values. We can notice some
       red pixels.
2   0   0   0     0   0   0     0   0   0    255   0  0
3   0   0   0     0   0   0     0   0   0      0   0   0
4   0   0   0     0   0   0     0   0   0      0   0   0
5 255   0  0     0   0   0     0   0   0      0   0   0
6
7 // Here each pixel is labeled.
8 // The first pixel corresponds to R0,G0,B0.
9 // The second pixel corresponds to R1, G1,B1.
10 // The third R2,G2,B3, and so on.
11  R0 G0    B0     R1   G1   B1     R2   G2   B2    R3   G3   B3
12  R4 G4    B4     R5   G5   B5     R6   G6   B6    R7   G7   B7
13  R8 G8    B8     R9   G9   B9     R10 G10 B10   R11 G11 B11
14  R12 G12 B12    R13 G13 B13    R14 G14 B14   R15 G15 B15
15
16  // Note that at the top of the .ppm image, there are
17  // two lines of 'header' information that describe the file
18  // format.
```

Listing 5.1: Raw pixels

5.3 Files

You will use the following code to help get you started.

- Typing this code in will give you some practice using the file commands.
- You will have to download this file to your local directory: `www.mshah.io/comp/11/labs/lab5/image.ppm`
- The part you have to fill in, is again figuring out how to get pixels output to a file.
- Note, that if you want them to be gray, a gray pixel is the average of the R,G,B components, divided by 3. This value then gets put into each of R,G,B from which the value was computed.

```cpp
#include <iostream>
#include <fstream>

// Desc:
// Input:
// Output:
void loadImage(int data[], std::string fileName){
  std::ifstream myImageFile(fileName.c_str());

  // This is the header information for the .ppm file.
  // It stores the format, and dimension so we know what
  // our input is.
  std::string format;
  int width,height;

  myImageFile >> format;
  myImageFile >> width;
  myImageFile >> height;
  // Why do we iterate from width*height*3?
  // This is the number of pixels we have.
  // And each iteration we get 3 values
  // from our file.
  // Convince yourself that incrementing by 3 is okay
  // Also convince yourself, incrementing by 1, and only
  // calling myImageFile >> data[i] would also be okay.
  for(int i =0; i < 640*480*3; i=i+3){
    // put data from myImageFile into array
    myImageFile >> data[i];
    myImageFile >> data[i+1];
    myImageFile >> data[i+2];
  }

  // Do not forget to close
  myImageFile.close();
}


// Desc: you write
// Input: you write
// Output: you write
void makeGrayScale(int pixelData[],std::string outputFileName){
  // Output a file here
  // We use c_str() member function to get the correct format.
  std::ofstream outFile(outputFileName.c_str());
  // This is some special header information for the PPM image
  // Do not remove these 2 lines.
  outFile << "P3\n";
  outFile << "480 640\n";

  // Write your code here
  // This is where you need to figure out how to manipulate
  // pixels and make them gray.
  //
    // TODO: hmmm
    //
    //

```

```
58    // Do not forget to close your file
59    outFile.close();
60  }
61
62
63  int main(){
64    // Our program will handle exactly
65    // 640x480 size images.
66    // Why do I multiply by 3 here?
67    // 640x480 is the number of pixels, but we have 3 components (R,G
        ,B)
68    // So this gives us all of the colors we need.
69    int pixelData[640*480*3];
70
71    loadImage(pixelData, "image.ppm");
72    makeGrayScale(pixelData,"gray.ppm");
73
74
75    return 0;
76  }
```

Listing 5.2: Starter Code

5.4 Output

When we run your program, the expectation is that we will get an image called gray.ppm that has applied a gray scale filter.

- You can use the program GIMP (Free) to preview .ppm files if you are not able to view them on your machine.
- You can try the command line on Mac or Unix: "convert gray.ppm gray.png" (This will give you a .png file instead, which you can likely preview)
- An alternative is to use a free online convertor, although this is slower way to iterate on `https://www.coolutils.com/online/PPM-to-JPG#`

5.5 Submission

```
1  provide comp11 lab5 your_file.cpp README
```

Listing 5.3: Submit Assignment

5.6 Going Further

Did you enjoy this lab? Want to try out some additional commands to go further?

- Try outputting only the red colors from the image.

- Try outputting only the blue colors from the image.
- Try outputting only the green colors from the image.
- Try making the image black and white. How can you make use of your grayscale filter, and the fact the pixel intensity goes from 0-255 to determine if a pixel is black or white?