CSC 370: Programming Assignment 0
Bayer Filter
100 pts

Using the Cimg library, complete the provided C++ program so that it loads an image, computes and displays a Bayer Filter version of it, then demosaics and displays the reconstructed image.

Cimg Library

- Download the CImg library http://cimg.eu/download.shtml
- http://cimg.eu/reference/group__cimg__overview.html provides on overview of how to write a program using CImg, a sample “hello world” program, and how to compile CImg code.
- When you include the CImg header file in your code (*#include "Cimg/CImg.h"*), make sure you provide the full relative path between your code and the header file.
- CImg can natively handle .bmp images. You can use an image editing program (like Paint) to save your images in different formats. If you want to load JPG's and PNG's or other formats in your program, download and install ImageMagick on your computer.
- If you're using relatively high resolution images (i.e. 4160x3120), feel free to downsample them so that your code will run faster for now.
- If you're compiling your code using MinGW and you're using a 64 bit computer you may encounter some issues. MinGW emulates a 32 bit system. If you see errors such as “_fseeki64 is not defined...” open Cimg.h, search for the following lines, and comment the if/else portion. So

```
    //! Version of 'fseek()' that supports >=64bits offsets everywhere (for Windows).
    inline int fseek(FILE *stream, cimg_long offset, int origin) {
#if cimg_OS==2
     return _fseeki64(stream,(__int64)offset,origin);
#else
     return std::fseek(stream,offset,origin);
#endif
    }

    //! Version of 'ftell()' that supports >=64bits offsets everywhere (for Windows).
    inline cimg_long ftell(FILE *stream) {
#if cimg_OS==2
     return (cimg_long)_ftelli64(stream);
#else
     return (cimg_long)std::ftell(stream);
#endif
    }
```

should be changed to:

```
  //! Version of 'fseek()' that supports >=64bits offsets everywhere (for Windows).
    inline int fseek(FILE *stream, cimg_long offset, int origin) {
```

```
//#if cimg_OS==2
//      return _fseeki64(stream,(__int64)offset,origin);
//#else
      return std::fseek(stream,offset,origin);
//#endif
    }

    //! Version of 'ftell()' that supports >=64bits offsets everywhere (for Windows).
    inline cimg_long ftell(FILE *stream) {
//#if cimg_OS==2
//      return (cimg_long)_ftelli64(stream);
//#else
      return (cimg_long)std::ftell(stream);
//#endif
    }
```

Assignment Specifications – some of these steps have already been completed in the provided code.

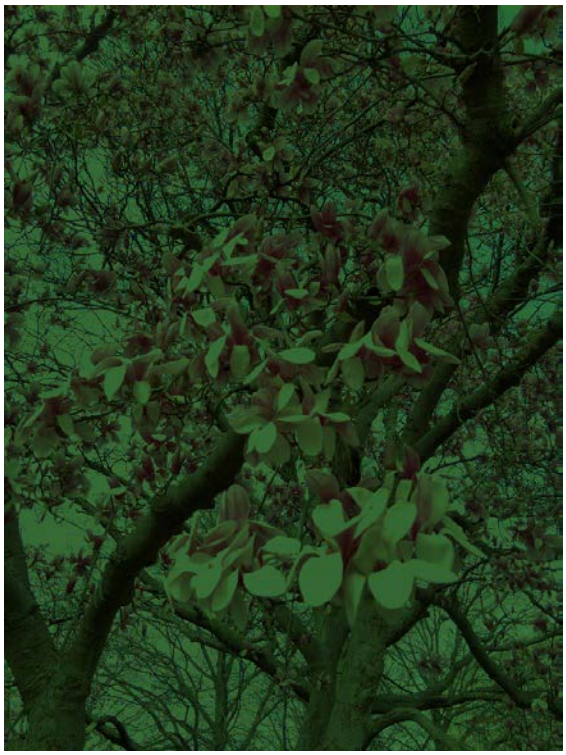
- Your code should load in a single image
- Create an empty image of the same size to create the Bayer Filter version of the image.
- Copy the red, green, and blue channels from the original image to the Bayer Filter image so that they follow the pattern shown below:

G	B	G	B	G	B	G
R	G	R	G	R	G	R
G	B	G	B	G	B	G
R	G	R	G	R	G	R
G	B	G	B	G	B	G
R	G	R	G	R	G	R

- Display and save the Bayer Filter image.
- Create an empty image of the same size as the original to reconstruct from the Bayer Filter.
- Demosaic the image.
- Display and save the Demosaiced image.

Example

Original image



Bayer filter image

Reconstructed image

Beware…
If there are missing/black pixels in your images at either stage (like below), that's an indication something has gone wrong.