READ ME

I. Description

This program is image convolution tool which can be used to read a image and implement convolution on this image with different convolution kernels. There are two main kinds of kernels in this program, i.e. basic kernels and Gabor kernels. In the basic kernel mode, the program simply reads a kernel file and an image file and convolve the image with this kernel. In the Gabor kernel mode, the program reads parameters and generate a Gabor kernel according to these parameters. Then the program convolve the input image with the generated kernel.

The program includes 6 files, which are 'my.h', 'filt.cpp', ‘Makefile’, ‘new1.filt’, ‘new2.filt’ and 'README.pdf'.

To compile on linux, use the Makefile that is included in the folder of this program.

Author: Siqi Wu
Email: siqiw@clemson.edu
Data: 10/15/2015

2. Data Structure And Methods

1). Object-Oriented Data Structure
In this program, a class is built for image. The class includes the basic information and some manipulation functions of the image.

Class Name: MyImage
Basic Information:

```
char *filename;
int    height;
```

```
int    width;
int    channel;
int    ochannel;
int    oheight;
int    owidth;
unsigned char *data;
unsigned char *originaldata;
```

Manipulation Functions:

```
void        SetFilename(char *newname){ filename = newname;}
void        SetOriginalData ();
void        ImageRead();
void        ImageWrite(char *);
unsigned char* ImageGetData(){ return data; }
unsigned char* GetOriginalData(){ return originaldata; }
int         GetHeight(){ return height; }
int         GetOHeight(){ return height; }
int         GetWidth(){ return width; }
int         GetOWidth(){ return owidth; }
int         GetChannel(){ return channel; }
int         GetOChannel(){return ochannel;}
char*       GetFilename(){ return filename; }
void        SetHeight( int seth ){ height = seth; }
void        SetWidth( int setw ){ width = setw; }
void        SetData( unsigned char *newdata){data = newdata;}
void        SetOHeight( int seth ){ oheight = seth; }
void        SetOWidth( int setw ){ owidth = setw; }
void        SetOChannel( int setc ){ ochannel = setc; }
void        SetChannel( int setc ){ channel = setc; }
void        ImageDisplay();
```

2). Image and Kernel Data Structure

1. Kernel Structure

In this program, kernels are stored using a 2D array of float. The size of the array is num*num(where num is the number of kernel size).

2. Convolution Image Structure

The convolution function reads the input image as a 1D array of unsigned character. Inside the function, the images is converted to three separate 2D arrays storing Red, Green and Blue channel information.

3). Methods

1. Normalization Method

In the convolution function, input image is first converted to floating point numbers between 0 and 1 when stored separately in three 2D arrays. After convolution, the image is normalized to 0~255 using clamping. Each pixel in Red, Green and Blue channel is scaled back to the range of the original channel. For example, a final pixel value of the red channel is result[m] = (middle[m]-minr)*(omaxr-ominr)/(maxr-minr). Where middle[m] is the result of convolution. minr and maxr are the minimum value and maximum value of the convolution result in Red channel. ominr and omaxr are the minimum and maximum value of the original Red channel.

2. Boundary Mechanism

Redefine convolution to produce zero when the kernel falls off of the boundary. If the kernel extends beyond the source image when centered on a sample I(x, y) then the output sample is set to zero. Adjust the scaler according to the kernel.

4) New Filters

There are two new filters in this package.

new1.filt :

0	2	-4
2	0	2
-4	2	0

new2.filt:

```
-4    2    0
 2    0    2
 0    2   -4
```

3.Introduction of Functions

1. Class Functions in file 'my.h'

```
    void     SetFilename(char *newname){ filename = newname;}
    void     SetOriginalData ();
    void     ImageRead();
    void     ImageWrite(char *);
unsigned char* ImageGetData(){ return data; }
unsigned char* GetOriginalData(){ return originaldata; }
    int      GetHeight(){ return height; }
    int      GetWidth(){ return width; }
    int      GetChannel(){ return channel; }
    char*    GetFilename(){ return filename; }
    void     SetHeight( int seth ){ height = seth; }
    void     SetWidth( int setw ){ width = setw; }
    void     SetData( unsigned char *newdata){data = newdata;}
    void     SetChannel( int setc ){ channel = setc; }
    void     ImageDisplay();
```

2. Function in file 'oiioview.cpp'

```
RenderScene()
float ** kernel(char* file)
float ** garbor(float theta, float sigma, float period)
unsigned char * convolve(MyImage image, float** kernel, int num)
void handlekey(unsigned char key, int x, int y)
```

4.User Instructions

1). Command Line Arguments

This program can be used in 2 modes. The first one is to convolve the input image with basic filter kernel. The second one is to convolve the input image with Gabor filter kernel generated by parameters. For each mode, user can write the displaying image if a write file name is specified.
Mode One:
Usage: ./filt [kernelname][filename]([writename])
Mode Two:
Usage: ./filt [filename]([writename])[-g][theta][sigma][period]

2). Keyboard Manipulation
I. Press 'c' or 'C' to convolve the image with specified kernel.
II. Press 'r' or 'R' to display the original image.
III. Press 'w' or 'W' to write the displaying image as the file name of the second argument.
IV.Press 'q' or 'Q' to quit the program.

5. Experiment Result

1). Filter Image square.png Using bell9.filt

The image on the left is the original image and the one on the right is the result. The image is blurred after convolution.

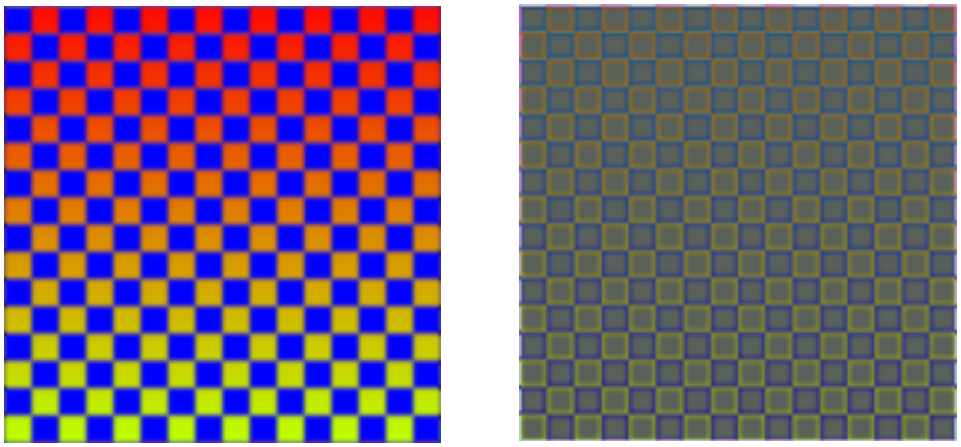
2). Filter Image square.png Using box9.filt

The image on the left is the original image and the one on the right is the result. The image is blurred after convolution. Compared to the result of bell9.filt, this image is more blurry.

3). Filter Image square.png Using tent9.filt

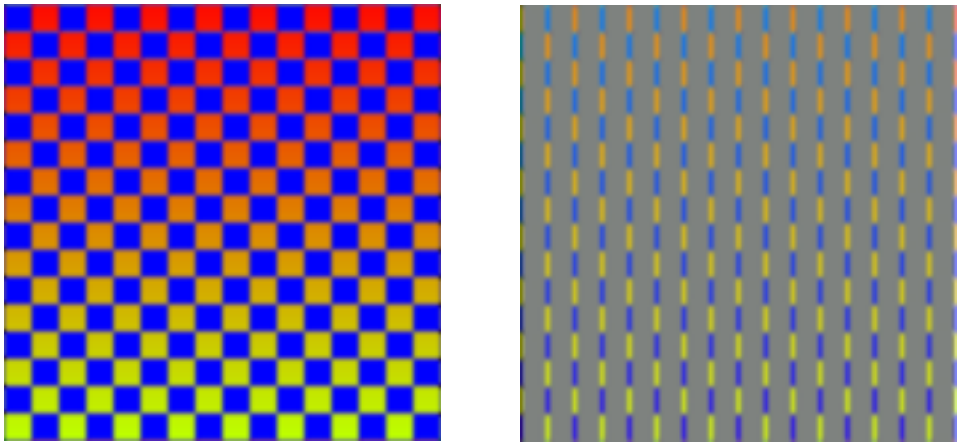
The image on the left is the original image and the one on the right is the result. The image is blurred after convolution. This image is less blurry compared to the result of box9.filt and slightly more blurry than the result of bell9.filt.

4). Filter Image checkers.png Using hp.filt



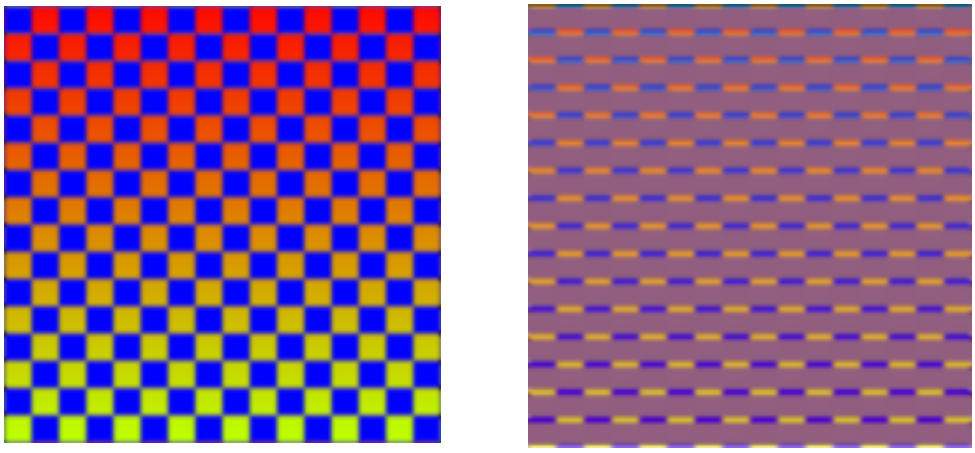
The image on the left is the original image and the one on the right is the result. The result highlights the edges of the original image.

5). Filter Image checkers.png Using sobol-horiz.filt



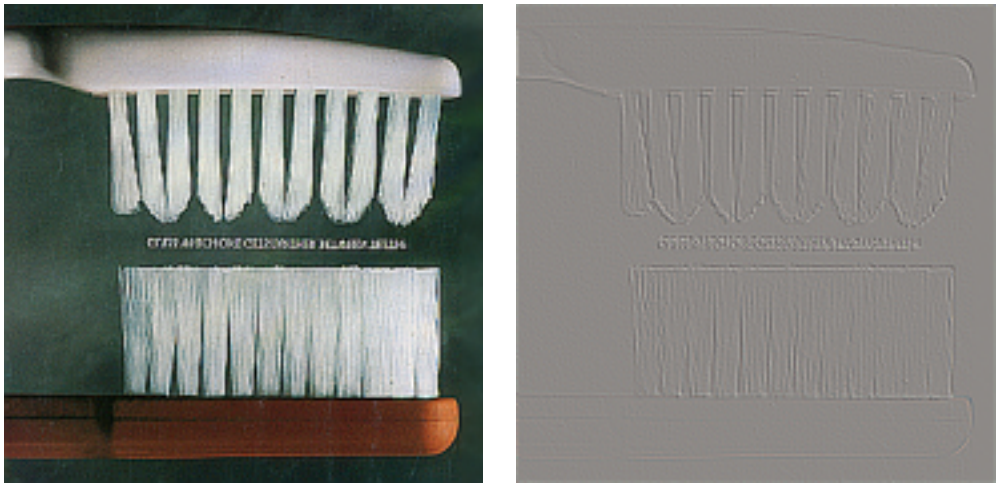
The image on the left is the original image and the one on the right is the result. The result detects the horizontal transitions of the original image.

6). Filter Image checkers.png Using sobol-vert.filt



The image on the left is the original image and the one on the right is the result. The result detects the vertical transitions of the original image.

7). Filter Image brushes.png Using new1.filt

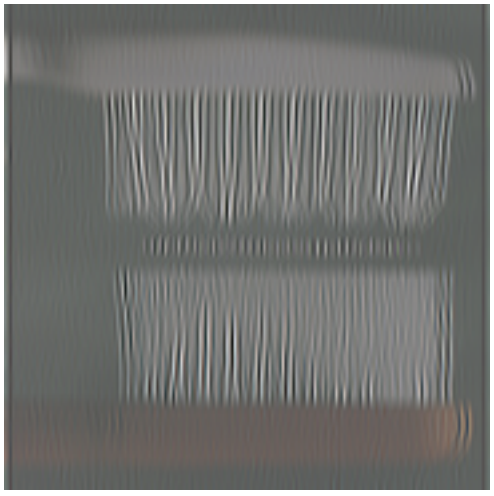


The image on the left is the original image and the one on the right is the result. The result detects the edges of the image. Edges with a direction of top right to bottom left are more obvious than others.

8). Filter Image brushes.png Using new2.filt


The image on the left is the original image and the one on the right is the result. The result detects the edges of the image. Edges with a direction of top left to bottom right are more obvious than others.

9). Filter Image brushes.png Using Gabor theta=0, sigma=4, period=4

The image on the left is the original image and the one on the right is the result. The result highlights the edges of the original image and also blurred the image, creating triple lines on vertical edges. The kernel focuses more on vertical edges.

10). Filter Image brushes.png Using Gabor theta=45, sigma=4, period=4


The image on the left is the original image and the one on the right is the result. The result only highlights the edges of the original image, creating triple lines on edges with a 45 degree angel. The kernel focuses more on edges with a 45 degree angel.

11). Filter Image brushes.png Using Gabor theta=0, sigma=4, period=8


The image on the left is the original image and the one on the right is the result. The result highlights the edges of the original image. Because the period is larger

than that in (9), so here the edges are stronger than that in (9). And the triple lines disappear. The kernel focuses more on vertical edges.

12). Filter Image brushes.png Using Gabor theta=45, sigma=4, period=8

The image on the left is the original image and the one on the right is the result. The result blurred the image and highlights the edges especially for the edges with a 45 degree angel.