CPSC 4040/6040
Computer Graphics Images

Assigned: Sept. 29, 2015
Due: 11:59pm, Thurs., Oct. 15, 2015

Programming Assignment 4 — Spatial Convolution *(Grading: 0–10 points)*

Problem Description

In this homework, you are asked to write and experiment with a program that will filter an image using spatial convolution. The program should be called `filt`.

In the zipped directory `convolve.zip` you will find subdirectories `images/` and `filters/`. `images/` contains .png images that you can use for your experiments. `filters/` contains text files with the file name suffix `.filt` that describe convolution kernels that you can use to filter your images. The format of the convolution kernel files is as follows (all values except the kernel size, N, are to be read as floats):

line 1: [kernel size N]
line 2: the first row of weights (N numbers)
...
line $N+1$: the last row of kernel-size weights

All kernel files describe square filters (of size $N \times N$). For example, a 3×3 unit box convolution kernel would be formatted:

```
3
1 1 1
1 1 1
1 1 1
```

The weights are to be read and stored in a convenient data structure (a 2D array is easiest).

Basic Requirements

The `filt` program should work as follows. It should first load and display an input image from an image file. When the user presses the 'c' key, it should then compute the convolution of the currently displayed image with the specified filter. In your convolution, pixel channels should be separately convolved. Repeated presses of the 'c' key will repeatedly apply the convolution. If the user presses the 'r' key, it should reload the original image. If the user presses the 'w' key, it should write the currently displayed image to an optionally specified output file.

The command line for `filt` should be as follows:

```
filt filter.filt in.png [out.png]
```

Where the `[ ]`'s indicate an optional parameter. Parameters have the following meanings:

1. `in.png`: The input image file name
2. `filter.filt`: The filter file name
3. `out.png`: The output image file name. If present, the filtered image is saved to this file.

Note that, when you convolve you will need to divide by the sum of the weights or "rescale" the kernel values. The scale factor to use is the maximum *magnitude* of either (a) the sum of the positive weights or (b) the sum of the negative weights. By using the magnitude, scale factors will always be a positive number. When you divide by this scale factor, you will guarantee that the sum of the weights is between $[-1, 1]$. Still, in this case it is possible to produce negative values for pixels. You will have to handle this via a normalization. *Your README should state which normalization method you've implemented.*

Your will find that your filtering code may be simpler if you store each channel of the image in memory as floating point numbers between 0 and 1. As you compute each output pixel value, it can be converted directly back to pixmap format and stored in the output pixmap.

Your code should implement a boundary mechanism of your choice. *Clearly state which boundary mechanism you chose to implement in your README.* You can either restrict the kernel (effectively skipping pixels that fall outside of the image), pad with zero, use tiling, or reflection. Other options are presented on pg. 128 of Hunt. If you skip pixels that are outside of the image instead of padding, make sure that you correctly scale the resulting kernel values.

There is no need to support filtering of four channel images, but if you choose to then I recommend filtering the alpha mask exactly as you would for any other color channel.

Experimentation

Do at least the following experiments with your program when you are done. Do a brief write-up of each experiment (put the text in your README and include attached images) explaining what you noticed about each one. You can use the remaining filters and images for additional testing.

1. Filter the image `squares.png` using the filters `bell9.filt`, `box9.filt`, and `tent9.filt`.
2. Filter the image `checkers.png` using the filters `hp.filt`, `sobol-horiz.filt`, and `sobol-vert.filt`
3. Define two of your own convolution kernels and try them on `brushes.png`. Please submit these kernels with your submission as new .filt files.

Code Documentation Please see Programming Assignment 01 for the expected requirements. Your code will be evaluated based on code structure, commenting, and the inclusion of a README and build script.

Make sure that your README describes the results of experimentation and that you include your two new .filt files in the submission.

Advanced Extension for 6040 students (worth 3 points)

A Gabor filter is one in which the filter kernel weights are determined by

$$g(x, y; \theta, \sigma, T) = \exp(-\frac{\hat{x}^2 + \hat{y}^2}{2\sigma^2}) \cos(\frac{2\pi\hat{x}}{T})$$

where

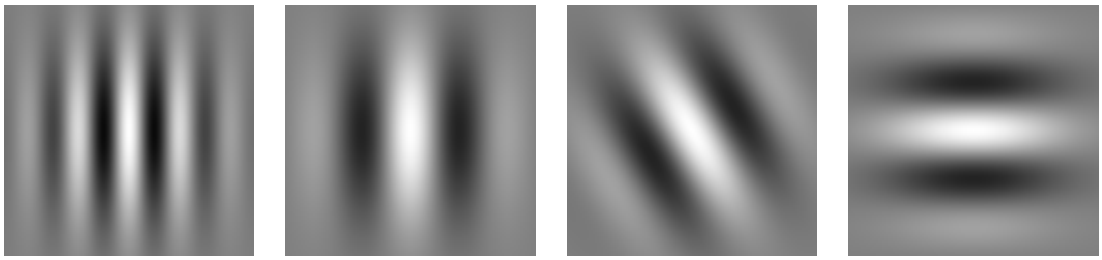
$$\hat{x} = x\cos\theta + y\sin\theta$$

and

$$\hat{y} = -x\sin\theta + y\cos\theta$$

Here (x, y) are distances measured from the kernel center, θ is an angular orientation, σ is the standard deviation of the Gaussian curve, and T is the period of the cosine.

The following images will give you an idea of what the Gabor kernels look like (white is weight of +1, black is weight of -1, mid-grey is 0):



$(0, 50, 50)$ $(0, 50, 100)$ $(30, 50, 100)$ $(90, 50, 100)$

For the advanced extension, allow the command line option `-g theta sigma period`. If this command line option is present, instead of reading the filter kernel from a file, build a kernel using the Gabor equations, with a square kernel of dimensions $(2\sigma + 1) \times (2\sigma + 1)$. Set the center of the kernel to be $(x, y) = (0, 0)$. In addition to the basic requirement, in your writeup, show examples of any one image convolved with Gabor filters of two different angles $\theta = 0, 45$ degrees, with a fixed value of $\sigma = 4$, and two different periods $T = 4, 8$.

Submission

(Please read all of these instructions carefully.)

Please write the program in C or C++, and I recommend using OpenGL and GLUT graphics routines for the display. Use OIIO for image input and output. Please take extra care to make sure that your homework compiles on the School of Computing's Ubuntu Linux cluster—testing on your own home machine, even if it runs Ubuntu, may not be sufficient. Remember:

both a working build script and README must be provided

Consequently, to receive *any credit whatsoever* this code must compile on the cluster, even if it does not accomplish all of the tasks of the assignment. The grader will give a zero to any code that does not compile.

Submit using the handin procedure outline at `https://handin.cs.clemson.edu/`. You are welcome to use the commandline interface, but the web interface is sufficient. The assignment number is `pa04`.

Finally, since we are using multiple files, please only submit a single file which has aggregated everything. This file should be named `[username]_pa04.tgz` where `[username]` is your Clemson id (please remove the brackets []). For example, mine would be `levinej_pa04.tgz`. Please make sure your build script is at the top level directory.

Rubric for Grading (Programming Assignment 4 — Spatial Convolution)

4040 students will be graded based on the following requirements:

I. +1 File I/O and Display

Code correctly reads, writes, and displays image data and reads in the filter file. Pressing 'r' resets to the input image and 'w' writes to an output image (if an output filename is specified).

II. +3 Convolution

Code correctly implements the convolution operator. Reflection is properly implemented. Code applies the convolution to the displayed image and properly sums the values per pixel. Scale factors are appropriately applied. Pressing 'c' convolves the currently displayed image (and pressing it repeatedly causes repeated applications).

III. +2 Boundary Handling

Which method of boundary handling is fully documented. Implementation is correct for the chosen method. Scaling factors are adjusted near the boundary, where appropriate.

IV. +2 Clarity

Does the code have good structure? Is the code commented and is a README provided? Is a working build script provided?

V. +2 Experimentation

README includes writeup of experiments, sample images, and two new filters are submitted.

6040 students will receive 7 points for completing the above requirements, scaling the above by 70%. To achieve 10 points they must also complete:

I. +3 Gabor filtering

Code correctly implements the -g flag and allows the user to vary the θ, σ, and period T. Kernel of the correct size and values are built and 'c' convolves with this filter. Writeup documents a couple of example images using this filter

Going above and beyond these requirements may result in extra credit at the discretion of the instructor and grader. Please note any extra features you implement in the README.