Heterogeneous and Cloud Computing

Homework #6

Max Points: 25

Due: Mon, Nov 3 2014 By 11:59 PM (EST)

Email-based help cut-off: 24 hours prior to due date

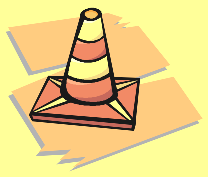
Objective: Develop an OpenCL image convolution program that can operate on different devices.

- Learning OpenCL operations on image data.

Submission: Once you have successfully tested your program and verified it meets all the formatting requirements, upload:

1. This document duly filled with necessary observations and inferences
2. The C++ source file(s) for this homework.

Grading Rubric:



This is a graduate course and consequently the expectations in this course are higher. Accordingly, the C++ program submitted for this homework must be operational in order to qualify for earning a full score.

NOTE: Program that do not compile, have methods longer than 25 lines, or just some skeleton code will be assigned zero score.

-5 points: The methods are not implemented concisely and efficiently by using good programming practices and suitable algorithms. Error conditions don't need to be handled and you can just throw an exception.

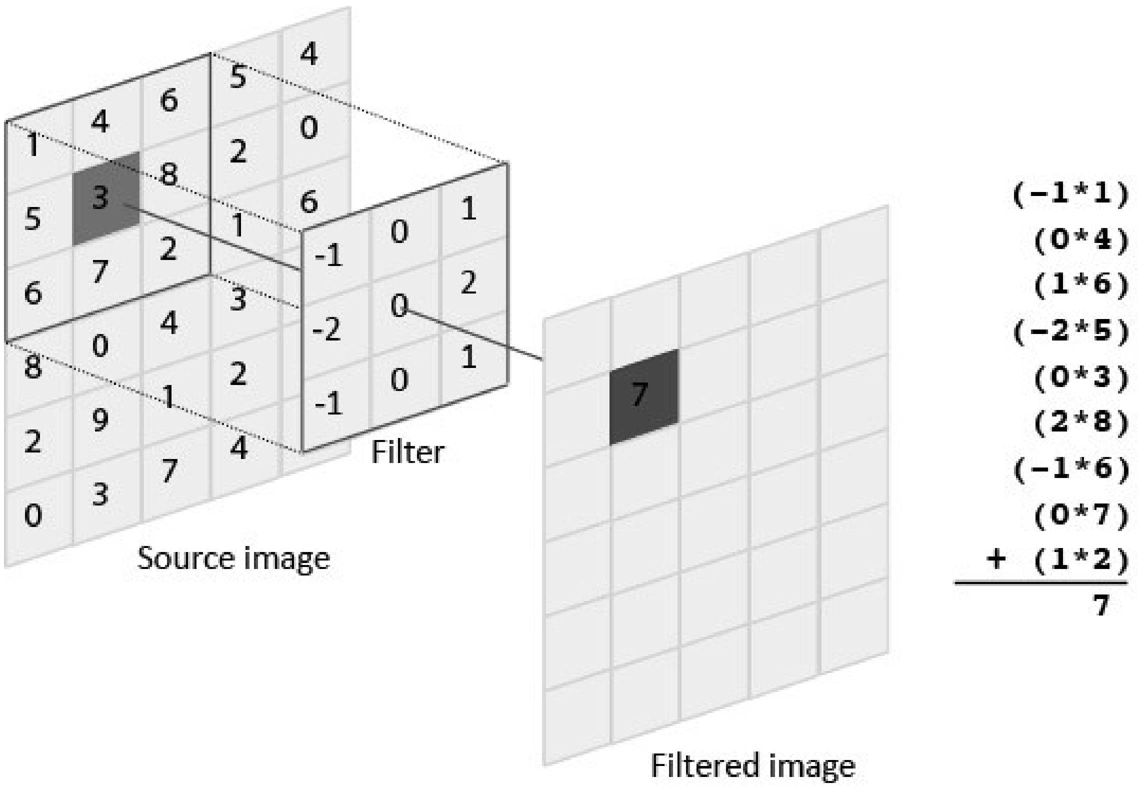
-1 point: For each style violations as reported by the C++ style checker (a slightly relaxed version from Google Inc. automatically downloaded by `Makefile` from Niihka)

-3 points: If the program does not include suitable comments at appropriate points in each method to elucidate flow of thought/logic in each method.

-1 point: For each warning message generated by g++ when compiling with `-Wall` (report all warnings) flag.

Image Convolution

In image processing, convolution involves using a small matrix to compute pixel values using sum-of-products of adjacent pixels. The operation is useful for blurring, sharpening, embossing, edge-detection, and more. This is accomplished by means of convolution between a given filter-matrix and an image. An example of convolution is illustrated in the image below:



Homework Program Requirement

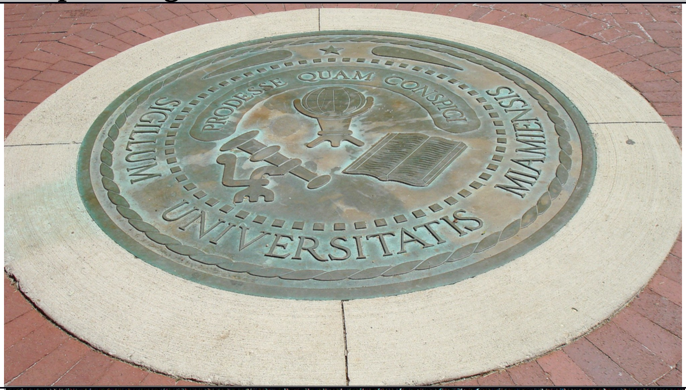
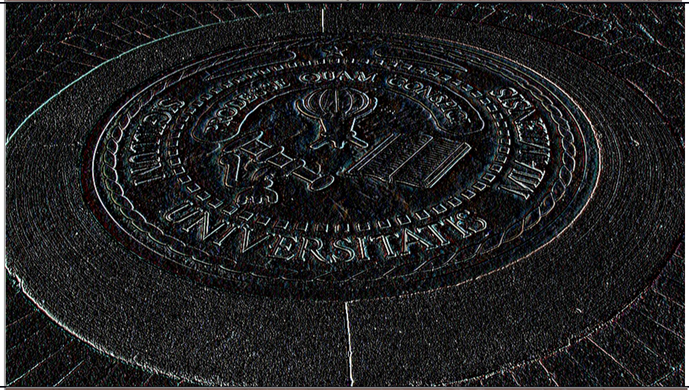
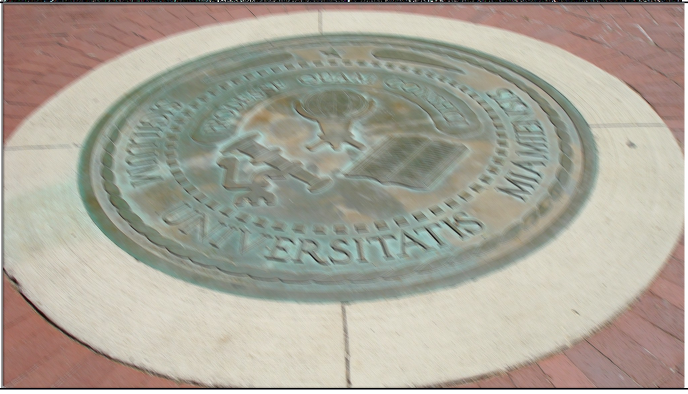
Develop a C++ program that uses OpenCL to perform image convolution (from data files) and writes the resulting image (to another image file) using the following command-line arguments:

1. The first command-line argument is one of the three strings "`CPU`", "`GPU`", or "`ACC`" indicating the OpenCL device to be used is a CPU, GPU, or Accelerator. Use the corresponding device on the first platform on which the device is available. Note: Not all Accelerators may support image format!
2. The second command-line argument is the input image file name. The input image will always be a bitmap image loadable using the supplied `BitmapImage` class.
3. The third command-line argument is the output imge file name. The output image will always be a bitmap image written using the supplied `BitmapImage` class.
4. The fourth command-line argument is the input convolution filter-matrix file name. Organization of the data in this file is simple and consists of 49 floating point numbers describing a 7x7 convolution filter-matrix. See supplied filter file(s) `copy.dat`, `edge.data`, and `blue.dat`.

Sample Outputs:

The sample outputs are shown as images to illustrate the operations performed by various convolution filter-matrices by running the different filters, instead of the edge filter, as show below:

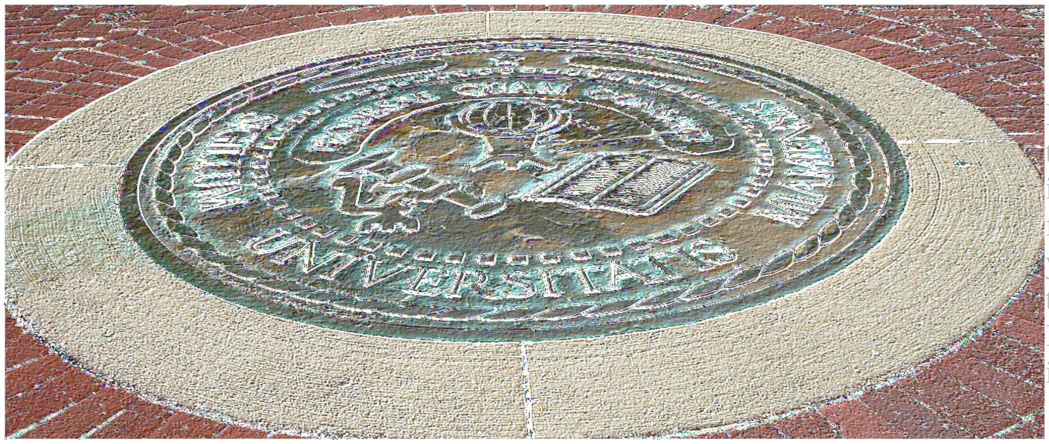
```
$ ./hw6 GPU miami_seal.bmp out.bmp edge.dat
$ eog out.bmp
```

Filter	**Output image**
`copy.dat`: Identical copy of source	
`edge.dat`: Simple edge detection filter.	
`blur.dat`: A blurring convolution filter	

Custom convolution matrix/filter

Once you have verified correct operation of your OpenCL program, create a custom convolution matrix that performs a custom convolution operation (its changes on the image must be clearly evident) on a given source image and include the details on the matrix below:

Emboss



Replace this text with the image resulting from applying your filter on the supplied Miami's seal image.

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	-2	-1	0	0	0
0	0	-1	1	1	0	0
0	0	0	1	2	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Help and questions

Use Niihka forums for soliciting clarifications regarding the homework, discussions, soliciting generic help, etc.

Submission

Once you have successfully tested your program and verified it meets all the requirements, upload:

1. This document saved as a PDF and duly filled with necessary information.
2. The C++ source file(s) for this homework.

Verify that your program meets all the requirements as stated in the grading rubric. No credit will be given for submitting skeleton code or programs that do not meet the base case. Ensure your C++ source files are named with the appropriate conventions. Ensure your solution meets the style guidelines by running it through the style checker. Upload each file individually. Do not upload zip/7zip/tar/gzip or any other archive file formats.