High-Performance Scientific Computing (MATH-GA 2011/ CSCI-GA 2945)

Homework Set 4

Out: October 4, 2012 · Due: October 10, 2012

Problem 1: Blur an image with OpenCL

In this problem, we're going to make an image "blurry" by replacing each pixel's value with a weighted average of its neighboring pixels. The weighting of the neighbors is given by this picture (which we'll call the *kernel*[1], not to be confused with the OpenCL kernel). You can find it in the PPM repository[2] as `gaussian-kernel.ppm`:

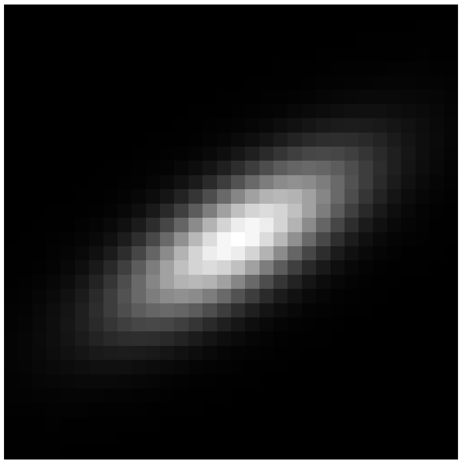

Figure: The blurring "kernel", an image which will provide the weights for our weighted average.

This image was made by `make-gaussian-kernel.c` in the same repository, which you can feel free to play with.

If you're on the hunt for fancy words, this whole process of 'compute weighted average of neighbors for each pixel' is called a 'convolution[3]'. We let

- $f(i, j)$ be the pixel value of the unblurry image at position (i, j) with $(0, 0)$ being the upper left,
- W and H be the width and height of the unblurry image,
- k be the width of the kernel in pixels (which we'll assume is even), and
- $K(i, j)$ be the value of the red channel of the kernel image such that $K(0, 0)$ represents the center. We'll also let $l = k/2$.

With these definitions we'll be computing the 'convolved' image g as

$$g(i,j) = \frac{1}{\bar{K}} \sum_{m,n=-l}^{l-1} K(m,n) f(i-m, j-n) \qquad (l-1 \le i < W-l, l-1 \le j < H-l)$$

[1] `https://en.wikipedia.org/wiki/Integral_kernel`
[2] `https://github.com/hpc12/hw3-ppm`
[3] `https://en.wikipedia.org/wiki/Convolution`

where

$$\bar{K} := \sum_{m,n=-l}^{l-1} K(m,n).$$

a) Load `gaussian-kernel.ppm` into memory. We will only be using information from the red channel.

   Precompute $\bar{K}$ for the kernel you load in (on the host). For `gaussian-kernel.ppm` it should come out to 48424.

b) Read a file name from the command line and read the image given by that file name. For the rest of this problem, you may assume that both width and height are divisible by 16. If you do assume this, make sure you check for it and print an error if it's not satisfied.

   If you need an image to test with, just use one that you like (a family photo, say, or one of the Mandelbrot images from last time) and convert it to PPM by writing

   ```
   convert my-treasured-family-memory.jpeg test-image.ppm
   ```

   `convert` is preinstalled in the virtual machine. You may also pass the option `-geometry WxH!` to `convert` if you need to change the size of the image. (The exclamation mark is important.)

c) Write an OpenCL kernel that directly implements the convolution formula above. Here and in the following parts, make sure to implement the formula using `floats` by converting all values to that type. Convert back to `unsigned char` (the image channel data type) on output. Do this for each channel of your image, and write the resulting image to `blurry.ppm`.

   Do not yet worry about the boundaries. In other words, only consider pixels of your image that are at least l pixels away from the boundary.

   Use workgroups of size 16×16.

d) Write an OpenCL kernel that loads the kernel K into local memory before it starts work. Make sure to use synchronization as needed. Again, carry out the convolution for each channel and write the resulting image to `blurry-local.ppm`.

   Use workgroups of size 16×16.

e) Change the kernel so that it does the right thing at the boundaries, too. Specifically, the average should not take into account pixels "outside" the picture. Note that you must also adjust $\bar{K}$ for these corner cases. Again, carry out the convolution for each channel and write the resulting image to `blurry-local-boundary.ppm`.

   Use workgroups of size 16×16.

f) Time the execution of all of the above kernels. Make sure you wait for the compute device at all the right spots. Print performance data in terms of pixels/s for all kernels.

g) Once again make sure you free/release all your buffers, command queues, host memory, and whatever other resources you've used.

   Also make sure that you check for error returns on all functions that can fail, including `malloc`, the `OpenCL` interface functions and the image reading/allocation functions.

Turn in a main C file `convolution.c` along with kernel files `convolution.cl`, `convolution-local.cl`, `convolution-local-boundaries.cl`. Make sure that `convolution.c` exercises all parts of this assignment when compiled and run.

Also, please make sure to *not* check any large image files into git. (You'll kill my server if you do. I'm not kidding.)