Parallel Computer Architecture
Separable 2D Convolution

Prof. Naga Kandasamy
ECE Department, Drexel University

May 20, 2019

The problem, worth 15 points, is due June 10, 2019, by 5:00 pm via BBLearn. You may work on this problem in a team of up to two people. One submission per group will suffice. Please submit original work.

Convolution filtering is used in a wide range of image processing tasks, including smoothing, sharpening, and edge detection. This assignment asks you to implement a 2D separable convolution filter on the GPU.

A *convolution filter* is a scalar product of the filter weights with the input pixels within a window surrounding each of the corresponding output pixels. More specifically, given a vector s and a convolution kernel k of size n, the convolution $r(i)$ of the i^{th} element of the vector is given by

$$r(i) = \sum_{n} s(i-n)k(n).$$

The elements at the boundaries, that is, elements that are "outside" the vector s are treated as if they had the value zero. Convolution can be easily extended into two dimensions by adding indices for the second dimension. For example, given a 2D matrix s and a convolution filter k of dimension $n \times m$, where n is the width and m is the height of the filter, the convolution of element (i, j) in s is given by

$$\sum_{n}\sum_{m} s(i-n, j-m)k(n, m).$$

The above operation should be quite familiar to you, having discussed 1D convolution in lecture. A 2D convolution filter requires $n \times m$ multiplications for each output element. *Separable filters* are a special type of filter that can be expressed as the composition of two 1D filters, one on the rows of the matrix and one on the columns of the matrix. So, a separable filter can be divided into two consecutive 1D convolution operations on the data (row-wise and column-wise), requiring only $n + m$ multiplications for each output element.

The program provided to you accepts the dimensions—number of rows and number of columns—of the input matrix as command-line parameters. It creates a random Gaussian kernel of specified width as well as an input matrix consisting of random floating-point values. A CPU implementation of separable convolution generates a reference solution which is compared with your GPU program's output.

Edit the `compute_on_device()` function within the file `separable_convolution.cu` to complete the functionality of 2D separable convolution on the GPU. To achieve this functionality, you may add multiple kernels to the `separable_convolution_kernel.cu` file.

This assignment will be graded on the following parameters:

- **(5 points)** You may just use GPU global memory to get the program working correctly without any performance-related optimizations.
- **(10 points)** Improve the performance of your code using the GPU's memory hierarchy (shared memory, and/or texture memory, and/or constant memory) efficiently to maximize the speedup over the serial version. Other optimizations may include loop unrolling, appropriately sizing the thread granularity, etc.

Submit the files needed to run your code as a single zip file via BBLearn. Provide a short report describing how you designed your GPU code, using code or pseudocode if that helps the discussion, and the amount of speedup obtained over the serial version for input-matrix sizes of 2048×2048, 4096×4096, and 8192×8192 elements. Also, characterize the sensitivity of the GPU kernels to thread-block size in terms of the execution time. Ignore the overhead due to CPU/GPU communication when reporting the performance.