All exercises must be done by yourself. You may discuss questions and potential solutions with your classmates, but you may not look at their code. If in doubt, ask the instructor.

Acknowledge all sources you found useful.

Your code should compute the correct results.

Partial credit is available, so attempt all exercises.

Submit your answers as a PDF file.

Exercise 1

Take the 2D convolution program provided in Assignment 1 and rewrite it in CUDA. The command-line interface should remain the same, and you will need to write a `Makefile` so that all of the programs below can be compiled by invoking `make` *`program`* where *`program`* is the binary name given in each question.

1. Convert the `convolve` function to a CUDA kernel and use standard CUDA memory copies to transfer the images, filters, and outputs to/from the CPU. Call this `2dconv_gpu.cu`, which should compile to the binary `2dconv_gpu`.

2. Convert the `2dconv_gpu` to use CUDA Unified Memory, call this `2dconv_gpu-um`. In this version, you will use `cudaMallocManaged` to allocate all data shared between the CPU and GPU, and remove all explicit memory copies. Compare the performance of this version to `2dconv_gpu`.

3. *(4xx-level only, extra credit for 2xx)* Modify `2dconv_gpu` from Q1 above to convert all read-only data on the GPU to use CUDA texture objects. Essentially, after allocating the data in GPU memory, you will need to bind those arrays to CUDA texture objects and modify the kernel to read from these textures instead of using loads/stores. **IMPORTANT**: You should setup textures to return 0 when reading beyond the border so that no conditionals checking for border pixels remain in the kernel. Call this `2dconv_gpu-tex`.

4. *(Extra credit for everybody)* Modify either `2dconv_gpu` or `2dconv_gpu-tex` so that filter coefficients (i.e. the 3x3 matrix or the 5x5 matrix) are stored in CUDA constant memory. Call this `2dconv-gpu-const`.

END.

1 Answers

Answer of exercise 1