Info7072a Lab 2

CUDA Image Color to Grayscale

(based on the GPU Teaching Kit -- Accelerated Computing)

Objective

The purpose of this lab is to convert an RGB image into a gray scale image. The input is an RGB triple of float values and the student will convert that triple to a single float grayscale intensity value. A pseudo-code version of the algorithm is shown bellow:

```
for ii from 0 to height do
   for jj from 0 to width do
      idx = ii * width + jj
      # here channels is 3
      r = input[3*idx]
      g = input[3*idx + 1]
      b = input[3*idx + 2]
      grayImage[idx] = (0.21*r + 0.71*g + 0.07*b)
   end
end
```

Prerequisites

Before starting this lab, make sure that:

- You have completed lab1

Image Format

For people who are developing on their own system, the input image is stored in PPM P6 format while the output grayscale image is stored in PPM P5 format. Students can create their own input images by exporting their image into PPM images. The easiest way to create image is via external tools. On Unix, bmptoppm converts BMP images to PPM images.

As the wblib is not working for us at this time, we can use netpbm lib to read an write the image files fro this lab. There is a compiled version of the netpbm lib at the lab directory. You can use the include files and in the netpbm directory and link with libnetpbm.a (just add libnetpbm.a to the nvcc compiler command)

Instructions

Edit the code in the code tab to perform the following:

- allocate device memory
- copy host memory to device
- initialize thread block and kernel grid dimensions
- invoke CUDA kernel
- copy results from device to host
- deallocate device memory

Instructions about where to place each part of the code is demarcated by the //@@ comment lines.

Local Setup Instructions

The most recent version of the template for this lab (source code and instructions) is at ~wagner/ci853/labs in the local machine.

You have to make a copy of each lan directory to your home account and work with your copy. The professor will give you instructions as to where/how to hand in your final solution.

The executable generated as a result of compiling your lab solution must run using the following command:

./ImageColorToGrayscale -i **<**input.ppm**>** -o **<**output.pbm**>**

where:

<input.ppm> is the input dataset, and <output.pbm> is an optional path to store the results. Some datasets are already generated in the lab directory.

Questions

1. How many floating operations are being performed in your color conversion kernel? EXPLAIN.
2. Which format would be more efficient for color conversion: a 2D matrix where each entry is an RGB value or a 3D matrix where each slice in the Z axis representes a color. I.e. is it better to have color interleaved in this application? can you name an application where the oposite is true?
3. How many global memory reads are being performed by your kernel? EXPLAIN.
4. How many global memory writes are being performed by your kernel? EXPLAIN.
5. Describe what possible optimizations can be implemented to your kernel to achieve a performance speedup.
6. Name three applications for color conversion.

Code Template

The code template at the lab directory is suggested as a starting point for students. The code handles the import and export as well as the checking of the solution. Students are expected to insert their code is the sections demarcated with //@@. Students expected to leave the other code unchanged. The template uses a wblib (a file header included by the template). At this point the original wblib is not functional, so the professor is providing a “work-arround” version of the lib.

Modified by W.Zola/UFPR