GPU programming – Lab 6 18.11.22 – A.A. 2022/23

[GPU Teaching Kit – Accelerated Computing – Module 3]

Teacher: Corrado De Sio – corrado.desio@polito.it

Goal: to get familiar with simple image processing parallel algorithm and kernel optimization

Exercise 1: CUDA Image Color to Grayscale

Objective

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

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. (more info https://en.wikipedia.org/wiki/Netpbm)

```
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
```

Assignment

Write the code 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

Evaluate performance (global memory r/w, execution time…) and compare CPU (to be coded) and GPU solutions. Consider different input size (32x32, SD, HD, FHD, 4K).

Support Functions

img.h and img.cpp contains some useful functions for reading and writing PBM and PPM. You can extend them to support more formats, different depth or implement your own better version.

Exercise 2: Filters

Objective

In image processing, a kernel, convolution matrix, or mask is a small matrix used for blurring, sharpening, embossing, edge detection, and more. This is accomplished by doing a convolution between the kernel and an

image. [https://en.wikipedia.org/wiki/Kernel_(image_processing)]
The purpose of this lab is to implement an efficient image blurring algorithm for an input image.

Assignment
Write the code 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

Support different masks and different mask sizes.

Evaluate performance (global memory r/w, execution time…) and compare CPU (to be coded) and GPU solutions. Consider different input size (32x32, SD, HD, FHD, 4K) and different mask size.

Compare results with the previous exercise.