NOVA SCHOOL OF SCIENCE & TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Concurrency and Parallelism 2022-23

Lab 07: CUDA Implementations of Image Manipulation

Hervé Paulino & Alex Davidson

April 2023

Abstract

In this lab class you will use CUDA programming to implement software for manipulating images.

1 Introduction

The CUDA parallel computing platform allows computing expensive computational tasks in parallel, using GPU-computing techniques. In this lab, we will use a GPU simulation container for implementing and running CUDA code.

2 Download code

- Clone the code for this weeks assignment by following this link.

  `https://classroom.github.com/a/InpYE8iw`

- Install Docker on your machine: `https://www.docker.com/`.
  - Docker is a containerisation application that allows you to produce ready-made environments for running code.
  - Can think of it as a virtual machine that is tailor-made for the application that you intend to run.
- You will now need to run a docker container for simulating a GPU-based environment, that will you allow you to progress with the rest of the assignment. The README.md file in the repository that you cloned contains instructions for solving and compiling this week's task

3 Example: Vector addition

There is already code for performing and timing vector addition of two vectors using CUDA, based on a size parameter that you can specify. You should study and run this code to see an example of how CUDA implementations are written.

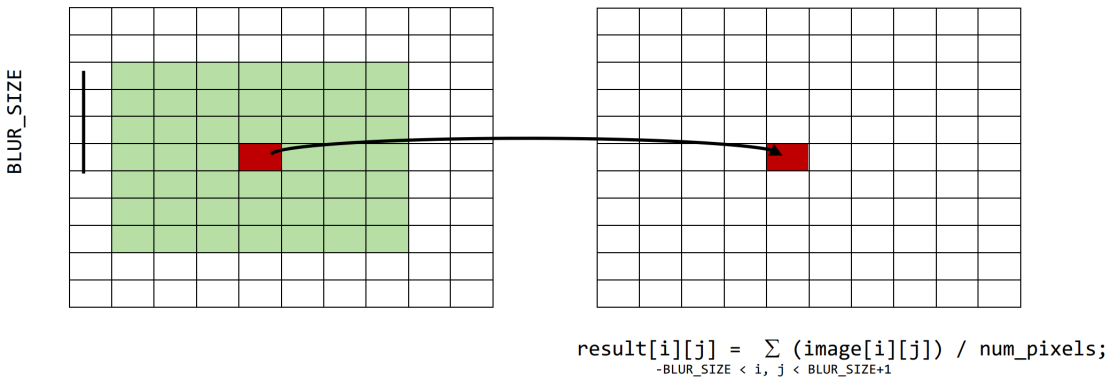
4 Task: Colour Image to Grayscale

Convert an RGB image into a grayscale image. For each pixel, the input is an RGB triple of float values that you must convert into a single float grayscale intensity value. A pseudo-code version of the algorithm is shown below:

```
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
```

5 Task: Image Blurring

Implement an efficient image blurring algorithm for an input image. Like the image convolution exercise, the image is represented as `RGB float` values. You will operate directly on the `RGB float` values, and use a $(2 \cdot \texttt{BLUR\_SIZE} + 1) \times (2 \cdot \texttt{BLUR\_SIZE} + 1)$' Box Filter to blur the original image, and to produce the blurred image.



To test your results against the supplied outputs, set `BLUR_SIZE` to 5.

6 Run on GPU cluster

If you complete the above tasks, we can attempt to run your solution on a dedicated GPU cluster.