Universidade de Aveiro

Mestrado Integrado em Engenharia de Computadores e Telemática

Arquitectura de Computadores Avançada

Home group assignment 2: Semi-Global Matching stereo processing using CUDA

Academic Year 2015/2016 NL/JLA

1. Introduction

In this assignment gray scale images will be processed in order to determine the disparity image between two images of a stereo pair. The disparity image assigns each pixel with a value proportional to the distance between that pixel (generally in the left image) and the corresponding pixel in the right image of the stereo pair.

Each image will be modelled as an array of integers which values range from 0 to 255. The values in the image specify the pixel luminance, hence a value of 0 indicates a black pixel and a value of 255 indicates a white pixel. An image will be stored in memory as an array (or matrix) of integer values where each element of the array/matrix corresponds to a pixel in the image.

The left and right images, provided by a stereo pair, will be processed for determining the disparity image using a simplified Semi-Global Matching method [1] with Birchfield and Tomasi measure for pixel cost calculation. It can be assumed that both images have rectified epipolar geometry. This method finds the disparity image in 3 stages: first, it determines the cost of each pixel in the left image for different disparity values (up to a given disparity range) by comparing this image with the right image; then an aggregation of the costs is performed by minimizing an energy constraint on the path of disparities that lead to each point on several directions; finally, from the aggregated cost for each disparity, the disparity image may be determined by finding, for each pixel, the disparity that minimizes the cost.

2. Work description

The objective of this work is to start from the source code package **`aca_sgm.tgz`** (available at elearning), which includes a C implementation of the Semi-Global Matching method (adapted from code available at [2]) and the **`testDiffs`** tool to compare images, and develop improved versions of the Semi-Global-Matching method using the CUDA platform. Images may be of any size. The function **`sgmDevice()`** should encapsulate all the operations of preparation, execution and result retrieval of the CUDA kernel(s). The assignment should be tested using the **`nikola.ieeta.pt`** computer that includes GPUs with compute capability 1.3. The function **`sgmHost()`** and functions called from **`sgmHost()`** should not be changed.

You may develop (and compare) several versions of your code that use different functionalities of the CUDA device (global memory, shared memory, texture memory, etc.). If you do test the use of different CUDA memory resources, please deliver all developed versions and use an archive file with an additional suffix in its name (ex: **`ex1_`*`pn_nm1_nm2`*`_shared.tgz`**) for the different memory types that were used.

1. Develop a CUDA kernel that can be used to replace the **`determine_costs()`** function. Change the **`sgmDevice()`** function to compute the disparity image using the new kernel. Submit your source code file as "**`ex1_`*`pn_nm1_nm2`*`.tgz`**"[1].

2. Develop CUDA kernel(s) that can replace the **`iterate_direction_dirxpos()`** function and corresponding functions in the other directions. Change the **`sgmDevice()`** function (from task 1) to compute the disparity image using the new kernel. Submit your source code file as "**`ex2_`*`pn_nm1_nm2`*`.tgz`**".

3. Develop a CUDA kernel that can replace the **`inplace_sum_views()`** function. Change the **`sgmDevice()`** function (from task 2) to compute the disparity image using the new kernel. Submit your source code file as "**`ex3_`*`pn_nm1_nm2`*`.tgz`**".

4. Develop a CUDA kernel that can replace the **`create_disparity_view()`** function. Change the **`sgmDevice()`** function (from task 3) to compute the disparity image using the new kernel. Submit your source code file as "**`ex4_`*`pn_nm1_nm2`*`.tgz`**".

3. Important notes

Before executing your CUDA code directly in the GPU, you must always execute it using **`cuda-memcheck`** to check for errors.

The work should be performed, if possible, by a group of 2 students (groups of more than 2 students are not allowed). Each group must deliver:

- the source code of the developed program;
- a report that presents: a) the general architecture of the developed solutions; b) the main data structures and algorithms that have been used; c) the results that have been attained; d) basic instructions for compilation and execution of your program.

During the development of this assignment you should follow an ethical conduct that prohibits plagiarism, in any form, as well as the participation of external elements in the assignment development. Any initiative that, judged by the teaching team, might be considered as a plagiarism situation will have real consequences on the student(s) evaluation and may lead to disciplinary sanctions.

4. Due dates

- **January 18, 2016**

Submitting your work after this date will be penalized with 1 point for each day of delay.

Bibliography:

[1] Heiko Hirschmüller, *Stereo Processing by Semiglobal Matching and Mutual Information*, IEEE Trans. Pattern Analysis and Machine Intelligence, 30(2):328–341, 2008
[2] *Code to Semi Global Matching*. http://lunokhod.org/?p=1403
[3] *CUDA C Programming Guide*, PG-02829-001_v7.5, NVIDIA (available at elearning)
[4] *CUDA C BEST PRACTICES GUIDE*, DG-05603-001_v7.5, NVIDIA (available at elearning)

[1] "**`pn`**" is the practical class id (p1, p2, or p3), "**`nm1`**" and "**`nm2`**" are to be replaced by the numbers of the group students (e.g. "ex1_p1_32156_82345.tgz").