Universidade de Aveiro

Mestrado Integrado em Engenharia de Computadores e Telemática

Arquitectura de Computadores Avançada

Home group assignment 2:
Canny Edge Detector using CUDA

Academic Year 2014/2015 NL/JLA

1. Introduction

In this assignment gray scale images will be processed in order to detect the edges of the image. An 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 image will be processed for edge detection using the Canny Edge Detector [1]. This algorithm finds the edges in the image in 4 stages: first, a gaussian filter is applied to minimize noise effects; then the gradient of the resulting image is obtained; from the gradient, a "non-maximum suppression" approach determines the best candidates for edges among several neighbors and finally edges are traced using hysteresis.

2. Work description

The objective of this work is to start from the source code package **`aca_canny.tgz`** (available at elearning), which includes a C implementation of the Canny Edge Detector (adapted from code available at [2]), and develop improved versions of the Canny Detector using the CUDA platform. Images may be of any size. The function **`cannyDevice()`** 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 **`cannyHost()`** and functions called from **`cannyHost()`** 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 **`convolution()`** function. Change the **`cannyDevice()`** function to compute the Canny Edges of an image using the new kernel to determine the vertical and horizontal gradients. Submit your source code file as "**`ex1_`*`pn_nm1_nm2`*`.tgz`**"[1].

[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").

2. Develop a CUDA kernel that can replace the **`non_maximum_supression()`** function. Change the **`cannyDevice()`** function (from task 1) to compute the Canny Edges of an 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 **`first_edges()`** function. Change the **`cannyDevice()`** function (from task 2) to compute the Canny Edges of an 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 **`hysteresis_edges()`** function. Change the **`cannyDevice()`** function (from task 3) to compute the Canny Edges of an image using the new kernel. Submit your source code file as "**`ex4_pn_nm1_nm2.tgz`**".

5. Develop CUDA kernels that can replace the **`gaussian_filter()`** function. Change the **`cannyDevice()`** function (from task 2) to compute the Canny Edges of an image using the new kernel. Submit your source code file as "**`ex5_pn_nm1_nm2.tgz`**".

Tasks 1 and 2 are mandatory. Additionally, each group can choose to deliver either tasks 3 and 4 or, in alternative, task 5 only. That is, you have the option to deliver tasks 1, 2, 3 and 4 or tasks 1, 2 and 5.

3. Important notes

The work should be done, 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 13, 2015**

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

Bibliography:

[1] Canny, J., “A Computational Approach To Edge Detection”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.
[2] *Canny edge detector - Rosetta Code*. `http://rosettacode.org/wiki/Canny_edge_detector`
[3] *NVIDIA CUDA C Programming Guide*, PG-02829-001_v6.5, NVIDIA (available at elearning)
[4] *CUDA C BEST PRACTICES GUIDE*, DG-05603-001_v6.5, NVIDIA (available at elearning)