Computação em Larga Escala

(ano letivo 2024'25)

Practical Assignment 3

Canny Edge Detector with CUDA Programming

Objective

This assignment involves processing grayscale images to detect their edges. An image will be represented as an array of integers, with values ranging from 0 to 255. These values denote pixel luminance: 0 represents a black pixel, and 255 represents a white pixel. Images will be stored in memory as an array (or matrix) of integer values, where each element corresponds to a pixel.

Edge detection will be performed using the Canny Edge Detector [1]. This algorithm proceeds in four stages: first, a Gaussian filter is applied to reduce noise; next, the gradient of the filtered image is calculated; subsequently, a "non-maximum suppression" step identifies the strongest edge candidates among neighboring pixels; and finally, edges are traced using hysteresis.

This work aims to improve the Canny Edge Detector by developing a CUDA-based version. This development will start from the provided source code package `cuda-canny.tar.gz` (available on elearning), which includes a C implementation of the Canny Edge Detector (adapted from code available at [2]).

Requirements

The `cannyDevice()` function, in the file `canny-device.cu`, should encapsulate all operations related to preparing, executing, and retrieving results from the CUDA kernel(s). The assignment must be tested on the `banana.ua.pt` computer, which has a GPU with compute capability 7.5. The `cannyHost()` function and functions called from `cannyHost()` should remain unchanged.

1. Develop a CUDA kernel to replace the `convolution()` function. Modify the `cannyDevice()` function to compute the Canny Edges of an image using this new kernel to determine the vertical and horizontal gradients.
2. Develop a CUDA kernel to replace the `non_maximum_suppression()` function. Modify the `cannyDevice()` function to compute the Canny Edges of an image using this new kernel.
3. Develop a CUDA kernel to replace the `first_edges()` function. Modify the `cannyDevice()` function to compute the Canny Edges of an image using this new kernel.
4. Develop a CUDA kernel to replace the `hysteresis_edges()` function. Modify the `cannyDevice()` function to compute the Canny Edges of an image using this new kernel.
5. Develop CUDA kernels to replace the `gaussian_filter()` function. Modify the `cannyDevice()` function to compute the Canny Edges of an image using these new kernels.

Grading

- Each task is worth 4 points.

Important Notes

Each group has access to the computer located at `banana.ua.pt`. To access this computer you must be connected to the UA network via eduroam or vpn. The login credentials for each group is `cleNN` whe `NN` is the group number, e.g, cle01, cle03, cle14. The password will be provided to each group upon request in class or by email.

The project source code include an `image` directory with 8 test images. These images will be used for evaluating the submited code.

Deliverable

The source code should be in the `banana.ua.pt` group home directory in the day of the delivery

Ensure that your directory includes:

- **All source files** .
- A **README.md** file with setup instructions, usage details, and a summary of your approach.
- **Performance analysis** results, including execution time comparisons and speedup calculations.

Deadline

June 5, at midnight.

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