NOVA SCHOOL OF SCIENCE & TECHNOLOGY
COMPUTER SCIENCE DEPARTMENT

Concurrency and Parallelism 2023-24
Iterative Histogram Equalization

Hervé Paulino

April 14, 2024

This is the project assignment for the Concurrency and Parallelism course, edition 2023-24. The deadline for submitting your project (both code and report) is

May 31, 2024, by 23:59

Submissions require the GIT repository URL and the commit ID of your project's code and report. Ensure the commit ID timestamp precedes the deadline. The submission will be done via a form to be available weeks before the deadline.

If you wish, you can submit the outcome of the initial phase of the project (Stage 1) by May 13. This submission is informal and can be done through email. The aim is for you to receive feedback and assessment of your work by that time.

1 Introduction

Histogram equalization is a technique used in image processing to enhance the contrast of an image by redistributing pixel intensities. It works by transforming the intensity values of an image so that the histogram of the output image is approximately uniform. This means that the intensities of the pixels are spread out over the entire range of possible values, making the image more visually appealing. The process of histogram equalization involves the following steps:

1. Compute the histogram of the input image, which represents the distribution of pixel intensities.
2. Compute the cumulative distribution function (CDF) of the histogram.
3. Normalize the CDF to scale the intensities to the range $[0, 255]$.
4. Map each pixel intensity of the input image to its corresponding normalized intensity using the CDF.
5. Generate the equalized image using the mapped intensities.

Iterative histogram equalization is an enhanced version of the standard histogram equalization technique. In standard histogram equalization, the histogram of the entire image is computed and used to transform the image's pixel intensities in a single step. Iterative histogram equalization, on the other hand, involves a iterative process to refine the histogram equalization. Here's how iterative histogram equalization typically works:

Initialization: Begin with an initial estimate of the histogram equalization transformation.

Iterative Process: Compute the Histogram Equalization of the current image until a convergence criteria are met (e.g., the difference between consecutive iterations falls below a certain threshold) or a predefined number of iterations is met.

Final Transformation: Apply the final estimated histogram equalization transformation to the input image to obtain the enhanced image.

Iterative histogram equalization offers several advantages over standard histogram equalization. It allows for finer control over the transformation process and can lead to better results, especially in cases where the histogram is not initially well-distributed or when there are large variations in contrast across different regions of the image. Additionally, iterative approaches can help mitigate the risk of over-amplifying noise present in the image. However, it may require more computational resources and time compared to the standard method.

2 Codebase

You have access to a C++ sequential implementation of a program designed to perform iterative histogram equalization on an image for a predetermined number of iterations. The code is available from the following GIT repository:

`https://classroom.github.com/a/EH4Oowbe`

The program accepts three arguments: the path of the file storing the image to be processed, the desired number of iterations for the iterative histogram equalization process, and the path of the file to store storing the generated image. More details on the project's `README` file.

A description of the project's structure follows:

cmake helper files for the project's CMake configuration.

dataset set of images for you to test. Your performance analysis should include more images than the ones featured here.

include project header files.

libwb header and source files for the WB library that offers some utility functions to handle images.

report template for the project's report. You must place your final report in this folder.

scripts folder for you to place scripts that allow to reproduce your experiments. More details in Section 3.3.

src project source files.

test project test files that make use of the Google Test framework (`https://github.com/google/googletest`)-

3 Assignment

Your assignment is divided into 2 stages, whose description follows.

3.1 Stage 1

In the first stage, your assignment is to parallelize the given sequential implementation. This involves scrutinizing the code to pinpoint areas of high computational load (hot-spots) or bottlenecks and restructuring them to exploit parallel processing capabilities. For that purpose, you may use OpenMP and/or other parallel programming libraries, if approved by myself. Here's the suggested strategy:

1. Thoroughly examine the original source code to ensure a comprehensive understanding. Feel free to discuss the code with colleagues and myself. Piazza is a good place to have such discussions.
2. Use a profiler to pinpoint potential hot-spots and bottlenecks, and identify the code suitable for parallelization.
3. Address any code dependencies that could impede the correctness or performance of parallelization.
4. Implement the parallelization. **Suggestion**: refactor your code to have all functionalities parallelized in their own function. It will help profiling and the development of stage 2.
5. Conduct tests to verify the correctness of the parallelization approach. You may use the Google Test framework for this purpose; an example is provided in the `test` folder.
6. Assess the impact of parallelization on performance.
7. Repeat steps 2 to 5 as needed.

3.2 Stage 2

Implement a GPU-accelerated version of the functionalities (functions) you have parallelized in Stage 1. For that purpose, you may use NVIDIA's CUDA parallel programming platform and the CUB pattern library (`https://docs.nvidia.com/cuda/cub/`)

To have multiple versions of the same function specialized for different parallelization strategies or hardware, I suggest the use of C++ templates and/or C++ compile-time expression evaluation (namely, the `if constexpr` construct). You have examples of both in the codebase (see files `include/template_example.h` and `test/template_test.cpp`).

To develop and test your solution, you will have access to the department's cluster. However, if you have an NVIDIA GPU you may do all the development and experiments in your own computer.

3.3 Performance Analysis

You are tasked with assessing your solutions for both Stage 1 and Stage 2. Regarding Stage 1, you are required to examine the performance enhancements achieved through each parallelization choice, contrasting the performance of our solution with the sequential version. Additionally, it is necessary to provide a scalability analysis, demonstrating how our solution adapts to increasing problem sizes and the number of workers.

Concerning Stage 2, you must compare the performance of the GPU-accelerated version against the most optimal configuration (wrt. the number of workers) achieved in Stage 1. You

may want to explore how these results fluctuate with the image size under processing and evaluate the effectiveness of combining or alternating the usage of CPU and GPU versions for the parallelized functionalities.

In order for me to assess the outcomes of your experiments, you need to place scripts in the `scripts` folder. These scripts, written in any language you prefer, should enable the replication of experiments and extraction of results. No need to generate charts, numeric values are just fine.

3.4 Project Report

The report must include the results of your profiling, which candidates were selected for parallelization and why, the parallelization decisions, the results of your experiments, and your conclusions.

Take note that your experiments must be as reproducible as possible. Therefore, you must present your evaluation setup (hardware and software) and evaluation methodology.

The report shall be written **strictly** following the format provided in the `report` folder. The report length is **limited to four pages** of text and graphics, plus **one extra page** for:

Section	Description
bibliography	Remember to add **all the documents and web-sites** relevant to your work to the bibliography section... and remember that you must cite them in the text wherever they were relevant;
acknowledgments	Please identify (number and name) your colleagues that were somehow helpful to your group while developing the project... also explain why/how they helped you;
individual contribution(s)	Please present your group working methodology and list clearly: i) how the work was divided between the group members, and ii) the relative contribution of each member (in percentage). For example, *A did whatever (20%), B did something else (30%), and C did a lot of stuff (50%).* If the group members cannot agree in the terms/contents for this section, add one (identified) separate statement for each group member.
comments, critics, and suggestions	All your comments, critics and suggestions are very welcome. All your feedback is very welcome (of course this section **will have no impact on your grade**).

3.5 Grading Criteria

Group grading ($\mathcal{G}$) remains consistent across all group members and is heavily influenced by my assessment of the group's performance, primarily based on the written report. Additional relevant criteria include ease of compilation and execution, code cleanliness and readability, result accuracy, execution time, scalability, among others.

Individual grading ($\mathcal{S}$) may vary among group members and is determined by both the group's reported work division and my evaluation of each individual's contribution, evident from the written report. Additional criteria may include the quantity and quality of commits in the GIT repository, and the significance of the commit messages.

As defined in CLIP, the individual lab grade will be calculated as a weighted averaged following the formula:

$$0.6 \cdot \mathcal{G} + 0.4 \cdot \mathcal{S}$$

Both the group members and the professor may ask for an individual or collective (group) presentation and discussion of the project. As a result of such a presentation and discussion, both the group grade ($\mathcal{G}$) and the individual grade ($\mathcal{S}$) may be revised upwards or downwards.