CENG305 hw 1 – Histogram equalization with Parallel Processing Using Pthreads

Due: November 8, 2024

In this assignment, you will implement a parallel image processing algorithm and learn foundational techniques for handling data synchronization to prevent race conditions. You will take necessary precautions to protect critical sections and ensure data consistency when multiple threads access shared data.

You will utilize multi-threaded parallelism with the **pthread** library to achieve efficient image processing. Image processing tools offer a wide range of algorithms for various applications, and a simple yet effective technique is Histogram Equalization. This algorithm re-adjusts the contrast of an image by modifying its histogram, enhancing the overall visual clarity and bringing out finer details. As you implement this algorithm, you will also gain experience in **managing concurrent threads** and **securing shared resources for correct and reliable outcomes.**

I have written the sequential code for you. Your task is to implement a parallel version of the algorithm that must produces the same output.

To compile it: **gcc hw1_yourid.c -lm -lpthread -o hw1**

The program takes 2 inputs from console: the first for the input image and second for the number of threads. Your program should perform a parallel multi-threaded **Histogram equalization** algorithm on the input image and then store the results into a new image. Your multi-threaded algorithm should give the exactly the same results as your sequential algorithm. To check the correctness of your parallel algorithm you must compare each pixel of the output image of multi-threaded algorithm with the output of the sequential algorithm.

You need to measure the elapsed times for the histogram equalization function (exclude the reading and writing times of the images), and in your report put a table that compare timings of your algorithms with varying number of threads.

What is required?

1. Report
    - A report that includes your names and surnames and appropriate title and small description of your homework.
    - Explain the multi-threaded algorithm and any specific considerations for handling the pixels of image.
    - Very short pseudocode of your **multithreaded** algorithm with at most **12 lines**.
    - A Table for elapsed times by using 1, 2, 3 and 4 threads, if your PC has more cores than 4 you can increase the number of threads. For accurate timing please **take average of at least 3 tests.**
2. Code
    - Source code named **hw1_yourId.c** written in C programming language in Linux environment.
    - Create varying number of threads
        - Assign sections of the image to threads for making the algorithm faster.
        - Measure the elapsed time for multi-threaded algorithm and print it to the screen.

    - The output should be saved as a **multithread_output.jpg** file.
    - A Makefile (20 pts) description for compiling the source codes.
    - You must use pthreads library (o.w. -100 pts).

- In your multithreaded code, you must take necessary precautions for data consistency for the race condition in the shared variables if it is necessary.
- You must keep the given global variables as they are and you must use them in your multi-threaded algorithm.

3. Submit your source codes (including hw1_yourId.c) and **Makefile** together with your report (**PDF format**) in a zip file named **hw1_yourid_yourNameYourSurname.zip** into AYBUZEM.

Example Table Layout:

	Elapsed Time (ms)
Sequential	505
Number of Threads = 1	506
Number of Threads = 2	289
Number of Threads = 3	151
Number of Threads = 4	172