UNIVERSITÀ
DEGLI STUDI
FIRENZE

Histogram Equalization

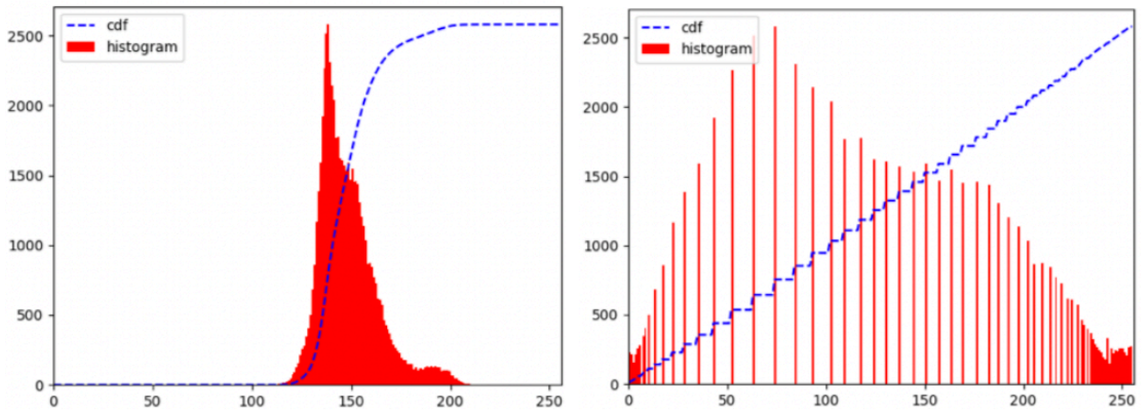
Andrea Simioni

UNIVERSITÀ DEGLI STUDI FIRENZE

Abstract

- Histogram equalization is a method used in Image Processing through which the contrast can be calibrated using the image histogram.
- An **image histogram** is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image.
- It plots the number of pixels for each tonal value.
- **Increases the global contrast:** especially when the usable data of the image is represented by close contrast values.
- Through this adjustment, the **intensities can be better distributed on the histogram**. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.
- **Invertible method**: if the histogram equalization function is known, then the original histogram can be recovered.

(a) Original Image (b) Equalized Image



(c) Original Histogram (d) Equalized Histogram

Main Goal

- Implement the method in two versions:
  - Sequential
  - Parallel
- Calculate the computational time of both.
- Evaluate the speed up.

Theoretical formulation

- The *probability* of an occurrence of a pixel x of level *k* in the image is:

$$p_x(k) = p(x = k) = \frac{n_k}{n} \qquad 0 \leq k < L$$

- *cumulative distribution function:*

$$cdf_x(k) = \sum_{l=0}^{k} p_x(l)$$

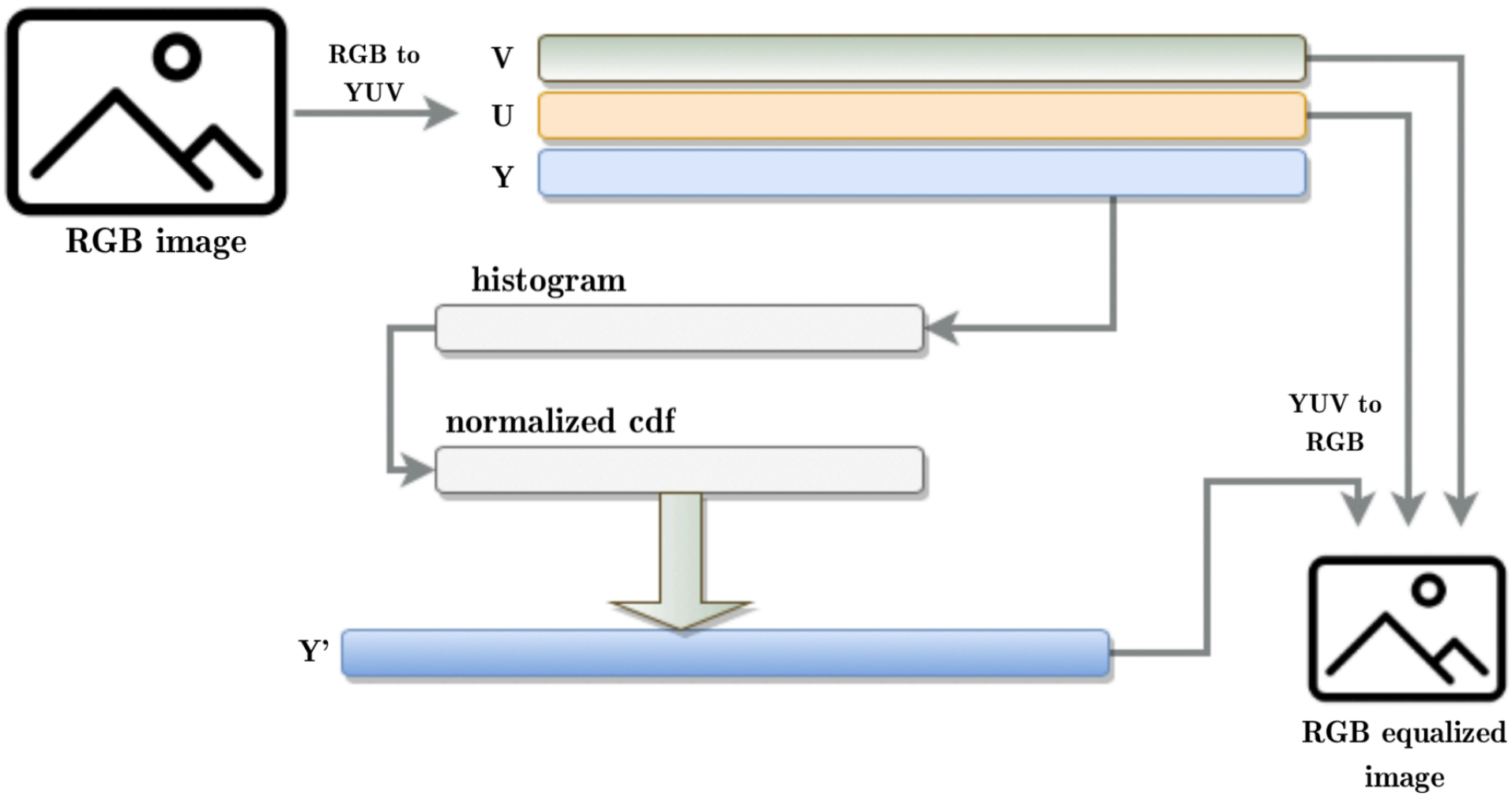
- *normalize* it such that the maximum value is 255 *:*

$$y(i,j) = h(x(i,j) = round\Big(\frac{cdf(x(i,j)) - cdf_{min}}{(M \cdot N) - cdf_{min}} \cdot (L-1)\Big)$$

Technologies

- Sequential Implementation: C++
- Parallel Implementation: OpenMP

Sequential



RGB⇒YUV

UNIVERSITÀ DEGLI STUDI FIRENZE

Parallel OpenMP

- OpenMP (Open Multi-Processing) is an API that provides C/C++/ Fortran the ability to run certain parts of the code in parallel .
- OpenMP version has been implemented reusing the sequential code and entrusting the **for loop** computation to an OMP specific structure:

```
#pragma omp parallel default(shared)
    {

#pragma omp for schedule(static)

        for (int i = 0; i < image.rows; i++) {

            for (int j = 0; j < image.cols; j++) {

                */ code here /*
            }
        }
    }
```

Tests

Sequential version:

- The testing phase has been performed using three images (Montebianco, Lavaredo, Forest), resizing their own size from 1000x1000 to 10000x10000 and calculating the time to complete the equalization procedure.

- Every result is calculated on the average of 5 runs.

Parallel versions:

- The analysis was not focused by varying only image size but also varying the number of threads used.

- 2, 4 and 8 threads were considered.

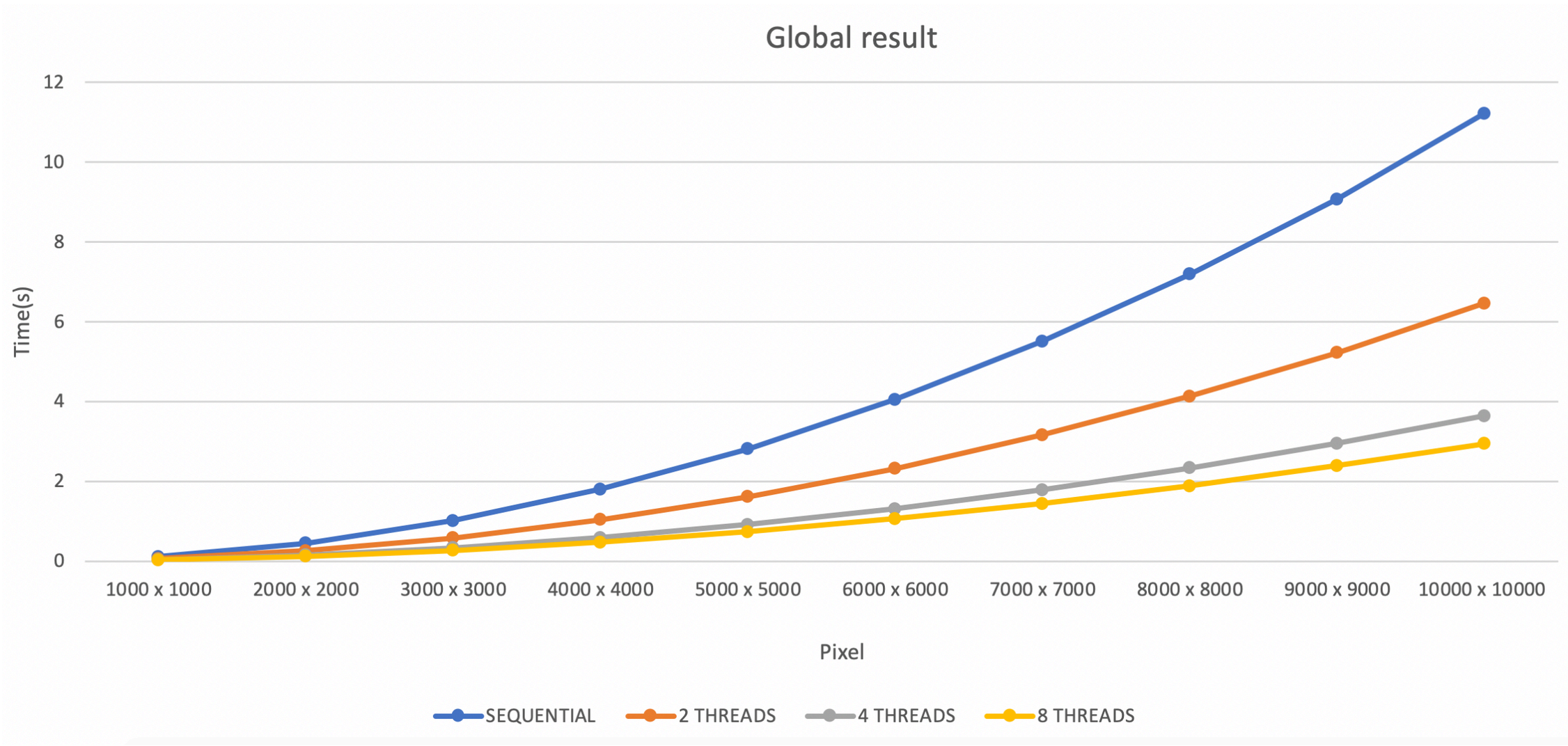
- Over the maximum number of threads chosen, computational times don't improve more significantly.

UNIVERSITÀ DEGLI STUDI FIRENZE

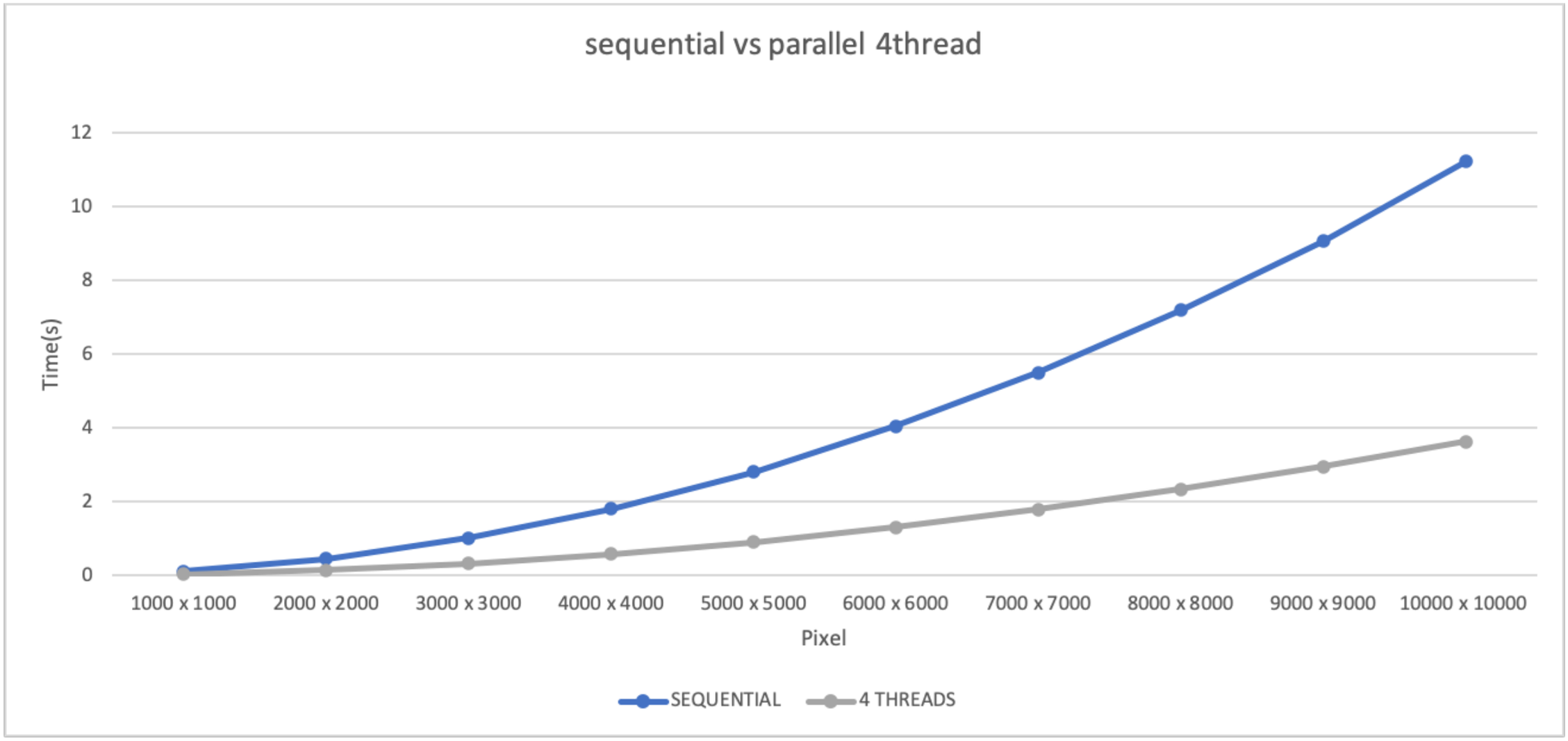
Results

Original image → Equalized image

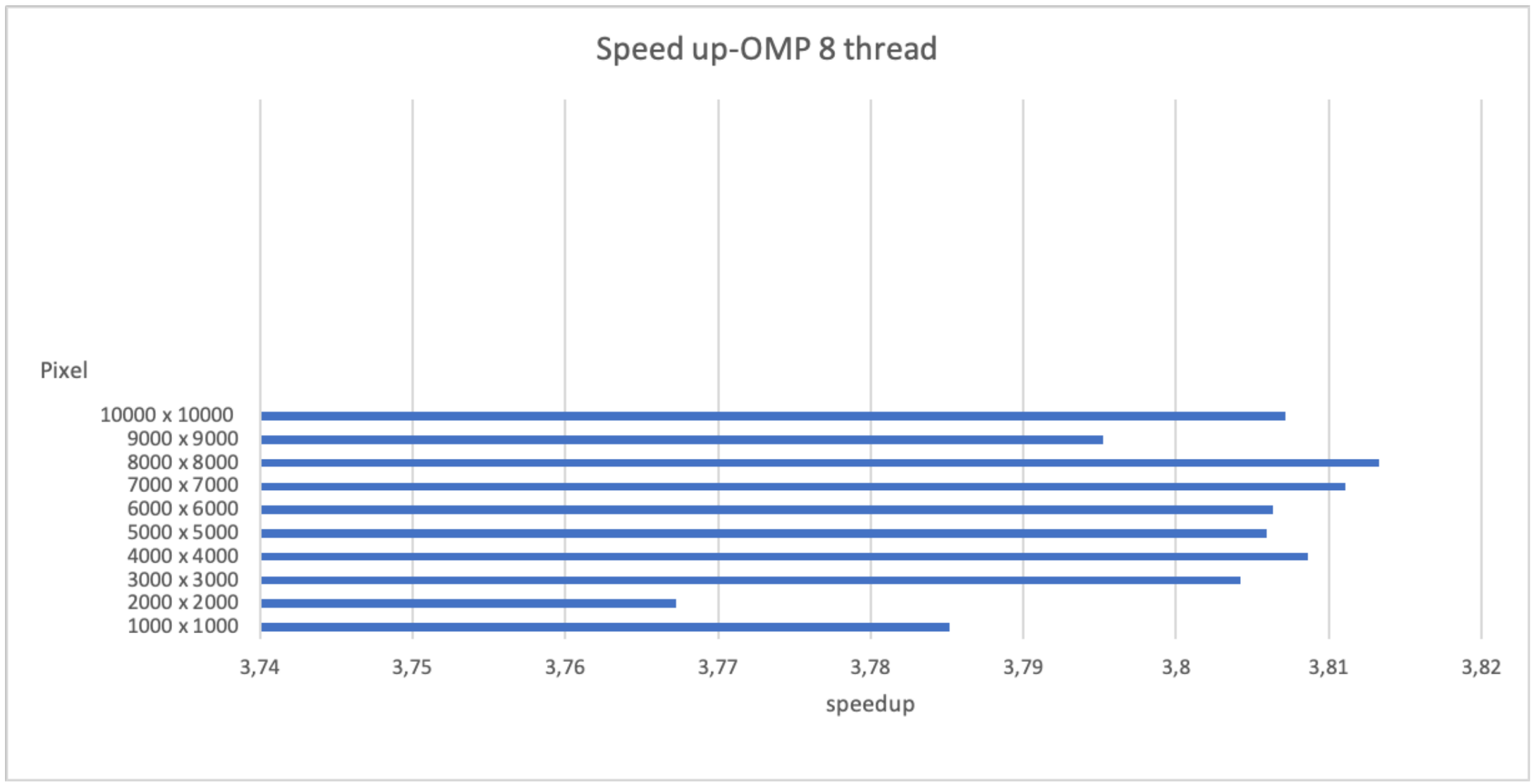
Results


Global result
12
10
8
6
4
2
0
Time(s)
1000 x 1000
2000 x 2000
3000 x 3000
4000 x 4000
5000 x 5000
6000 x 6000
7000 x 7000
8000 x 8000
9000 x 9000
10000 x 10000
Pixel
SEQUENTIAL
2 THREADS
4 THREADS
8 THREADS

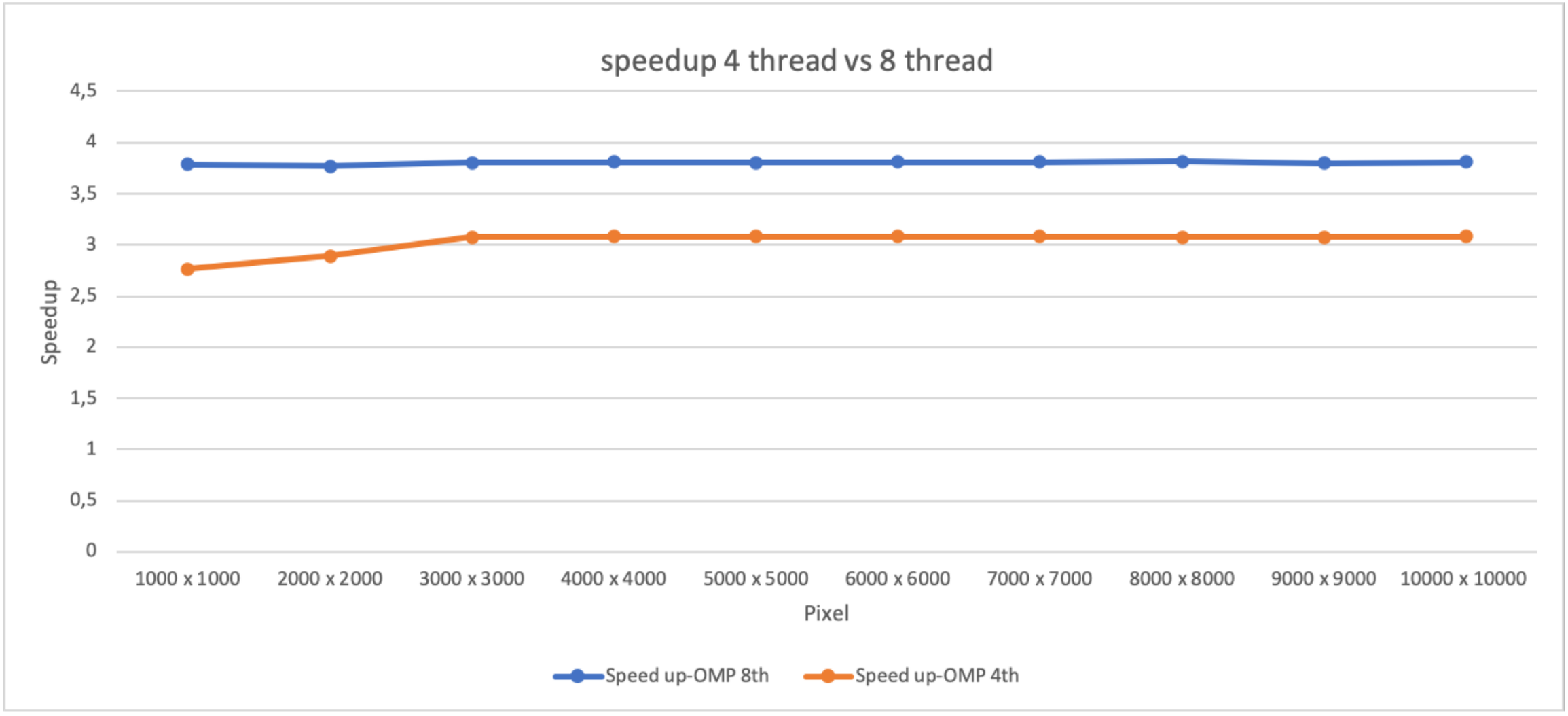
Results


sequential vs parallel 4thread
12
10
8
6
4
2
0
Time(s)
1000 x 1000
2000 x 2000
3000 x 3000
4000 x 4000
5000 x 5000
6000 x 6000
7000 x 7000
8000 x 8000
9000 x 9000
10000 x 10000
Pixel
SEQUENTIAL
4 THREADS

Results


Speed up-OMP 8 thread
Pixel
10000 x 10000
9000 x 9000
8000 x 8000
7000 x 7000
6000 x 6000
5000 x 5000
4000 x 4000
3000 x 3000
2000 x 2000
1000 x 1000
3,74
3,75
3,76
3,77
3,78
3,79
3,8
3,81
3,82
speedup

Results


speedup 4 thread vs 8 thread
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
Speedup
1000 x 1000
2000 x 2000
3000 x 3000
4000 x 4000
5000 x 5000
6000 x 6000
7000 x 7000
8000 x 8000
9000 x 9000
10000 x 10000
Pixel
Speed up-OMP 8th
Speed up-OMP 4th

UNIVERSITÀ
DEGLI STUDI
FIRENZE

Histogram Equalization

Andrea Simioni