Histogram Equalization - Final Project Assignment

Distributed and parallel systems and algorithms

Pa3cio Bulić

December 2022

1 A digital image and its histogram

Histogram equalization is an image processing technique that transforms an image in such a way that the histogram of the resultant image is equally distributed, enhancing the image's contrast. An equalized histogram means that probabilities of all grey levels are (almost) equal. In other words, histogram equalization makes an image using all colours in equal proportion.

This method usually increases the global contrast of images, especially when close contrast values represent the data in the image. 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.

A digital image is a binary representation of a two-dimensional image. The digital representation is an array of picture elements called pixels. Each of these pixels in monochromatic images has a numerical value representing a grey level. For example, in an 8-bit monochromatic image, each pixel has an integer value from the interval $[0, 255]$. The pixel value 0 represents the black colour, and the pixel value 255 represents the white colour. An image histogram is a type of histogram that offers a graphical representation of the tonal distribution of the grey values in a digital image. By viewing the image's histogram, we can analyze the frequency of appearance of the different grey levels contained in the image.

Consider a grayscale image X of size $N \times M$ pixels. Each pixel has a grey level $l \in [0, L-1]$. Let n_l be the number of occurrences of grey level l. The **image histogram** H is an array of length L:

$$H = \left\{n_0, n_1, n_2, \ldots, n_{(L-1)}\right\}$$

The **probability** of an occurrence of a pixel of level l in the image of size $N \times M$ pixels is

$$p(l) = \frac{n_l}{N \times M}$$

The **normalized histogram** contains the probabilities of the appearance of the different grey levels:

$$H_{norm} = \{p(0), p(1), p(2), \ldots, p(L-1)\}$$

The **cumulative distribution function** gives the probability that the random variable X is less than or equal to a certain number x and is defined as:

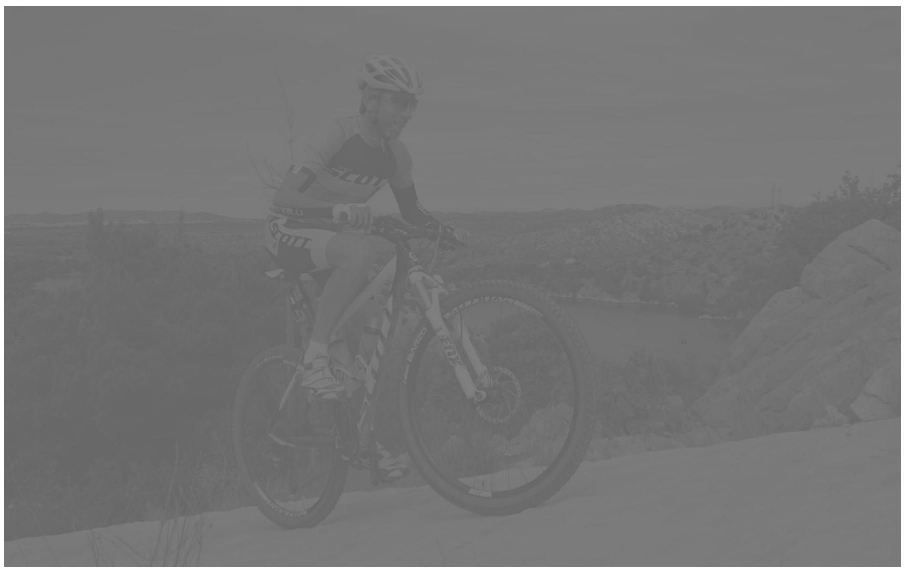
$$cdf(x) = p(X < x) = \int_0^x p(x)$$

and in the discrete space:

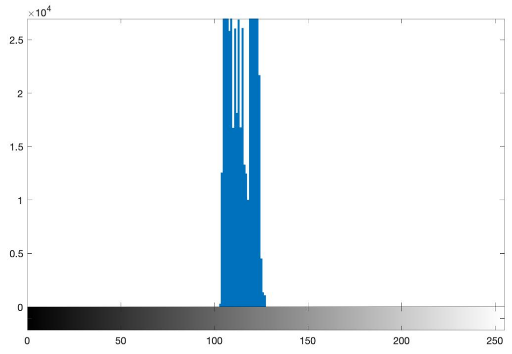
$$cdf(x) = p(X < x) = \sum_{x_i=0}^{x} p(x_i)$$

The cumulative distribution function corresponding to n_l is:

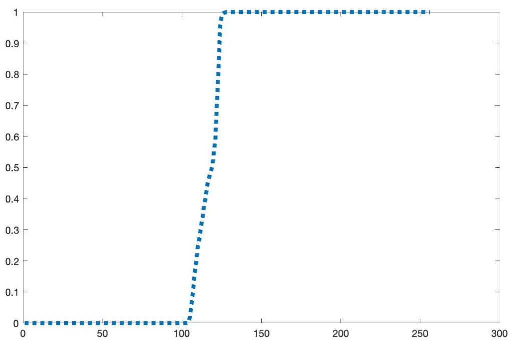
$$cdf(l) = p(X < l) = \sum_{i=0}^{l} p(l)$$



(a) Before Histogram Equalization.



(b) Histogram.



(c) Cumulative histogram.

Figure 1: A cyclist and its histograms.

or without normalization:

$$cdf(n_l) = p(n < n_l) = \sum_{i=0}^{l} n_l$$

The non-normalized $cdf(n_l)$ is used to count how many pixels in an image have a grayscale colour (level) smaller than l. It is easy to see that

$$0 \leq cdf(n_l) \leq N \times M$$

The cumulative histogram CDF contains all $cdf(n_l)$:

$$CDF = \left\{cdf(n_0), cdf(n_1), cdf(n_2), \ldots, cdf(n_{(L-1)})\right\}$$

Figure 1a depicts an image of a cyclist. The image has bad contrast. Its histogram is presented in Figure 1b. The histogram shows us that the image contains only a fraction of the entire range of grey levels. There are 256 grey levels in this case, and the image only includes values 105-125. Therefore this image has low contrast. Figure 1c shows us the cumulative histogram of the image in Figure 1a. The cumulative histogram is not linear across the input range. In the following sections, we describe the histogram equalization method that yields an image which uses all grey levels and has improved contrast.

2 Histogram equalization

An image histogram that covers all possible values across the grayscale range suggests that the image has good contrast and that details in the image are clearly visible. Our goal is to create a transformation to produce a new image with a flat histogram. Such an image would have a linearized cumulative distribution histogram across the value range. Suppose the image Y is obtained from the image X with the histogram equalization. Our goal is to find a **mapping function** $y = f(x)$ which

1. maps the gray level x in the image X to the gray level y in the image Y,
2. is such that the histogram of the image Y is flat (all grey levels are equally distributed).

2.1 The mathematical background of a mapping function [1]

Let $p(x)$ be the continuous histogram of image X and let $p(y)$ be the continuous histogram of image Y. Figure 2 shows an arbitrary mapping function $y = f(x)$. The mapping function should map the grey values in the same range but also flatten the histogram of the output image Y, i.e.:

1. if $p(x)$ is high for some grey level x, then map this grey level to a wider range, and
2. if $p(x)$ is low for some gray level x, then map this gray level to a narrower range.

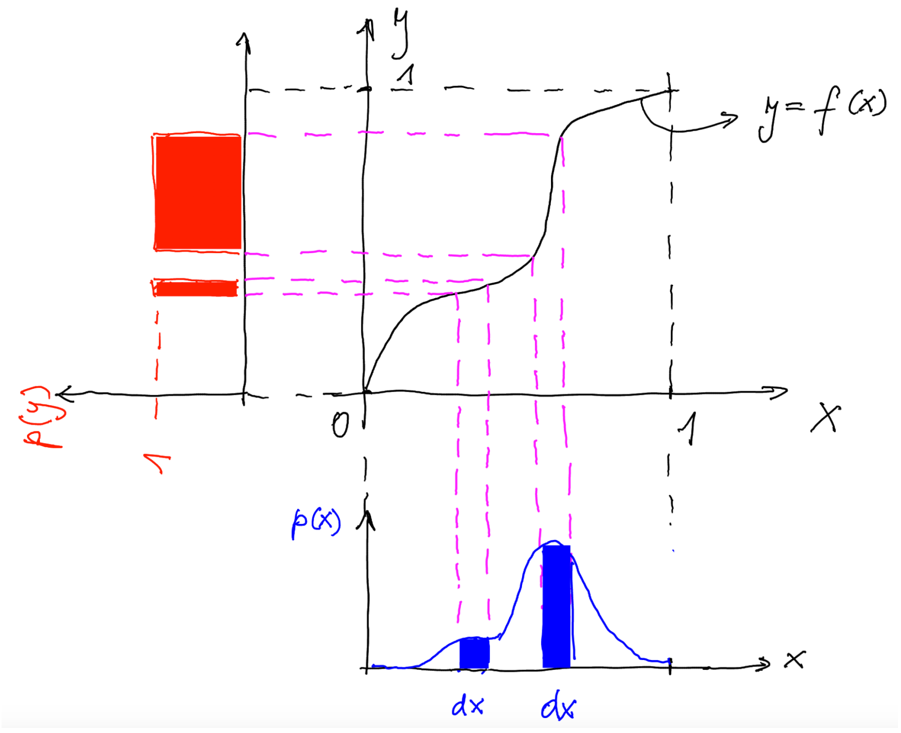


Figure 2: A mapping function.

Hence, the following must hold:

$$p(y) \cdot dy = p(x) \cdot dx$$

The above equation ensures that the transformation function results in an image in which the grey levels are mapped to a wider range dy if $p(y) < p(x)$ and the $p(x)$ is high, and vice versa. We want the histogram $p(y)$ to be flat, so $p(y)$ needs to be constant. This means that $p(y)$ needs to be 1 so that the cumulative probability of the output image is 1 ($\int_0^1 1 \cdot dy = 1$), hence

$$dy = p(x) \cdot dx$$

[1]The reader can skip this subsection if they are not interested in math

$$\int_0^y y = \int_0^x p(x) \cdot dx$$

$$y = \int_0^x p(x) \cdot dx$$

The new value for a grey level comes from the cumulative probability of that grey level. Thus, **the mapping function is the cumulative probability**. Note that the above formula is for the continuous domain.

The new grey level is a cumulative probability function of the old grey level. It can be represented in the discrete domain as follows:

$$l_{new} = (L-1) \cdot \sum_{i=0}^{l} p(l)$$

In the following subsection, we present the equalization algorithm and the mapping function for the non-normalized cumulative distribution.

2.2 Equalization algorithm

To equalize the histogram and linearize the cumulative distribution histogram, we will use the **histogram equalization algorithm**. The steps for implementing the histogram equalization algorithm are as follows:

1. Create the histogram for the input grayscale image.
2. Calculate the cumulative distribution histogram.
3. Calculate the new grey-level values through the general histogram equalization formula.
4. Assign new values for each grey value in the image.

The general histogram equalization formula (a mapping function) is:

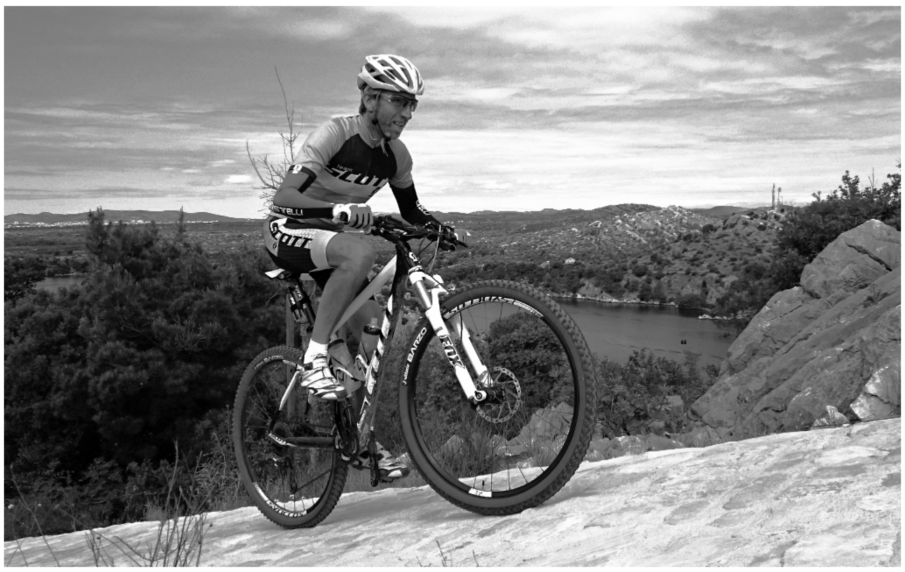
$$l_{new} = \left\lfloor \left(\frac{cdf(l) - cdf_{min}}{(N \times M) - cdf_{min}} (L-1) \right) \right\rfloor$$

where

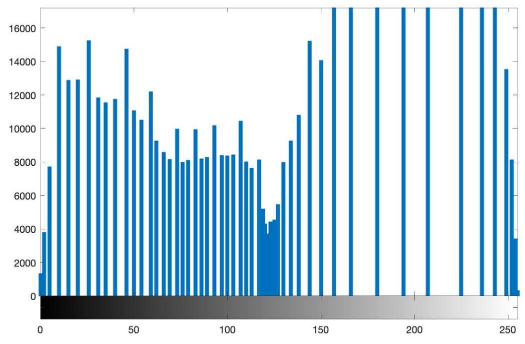
$$cdf(l) = \sum_{i=0}^{l} n_l$$

is the non-normalized cumulative distribution function of the original grey-level value l, cdf_{min} is the minimal cdf value in the cumulative histogram CDF, $(N \times M)$ is the dimension of the input image and L is the number of gray levels.

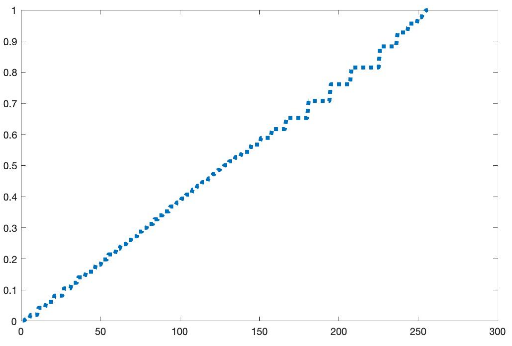
In Figure 3a, we can see an image of a cyclist after histogram equalization. We can see that the image's contrast has been improved. The original histogram has been stretched across the full range of grey values, as shown in the equalized histogram in Figure 3b. The cumulative histogram is now almost linear across the input range. Note that the histogram is not perfectly flat because of the approximation we did for the discrete domain. However, the formula works perfectly in the continuous domain. Although the histogram is not perfectly flat, histogram equalization increases the contrast because the resultant image uses all available grey levels. Another example of histogram equalization can be seen in Figure 4.



(a) After Histogram Equalization.

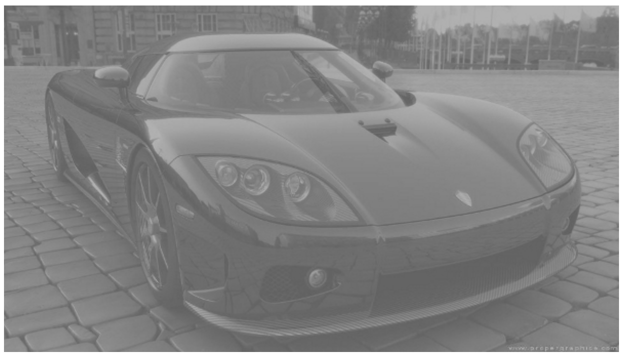


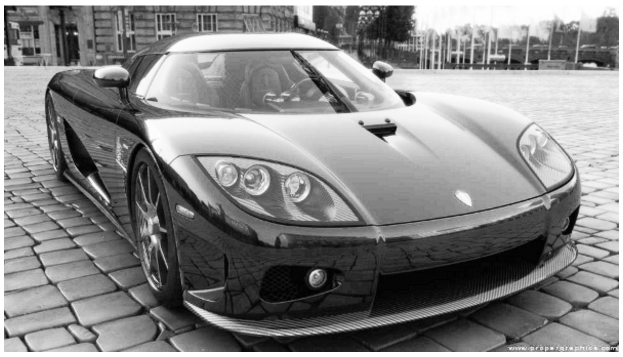
(b) Histogram.

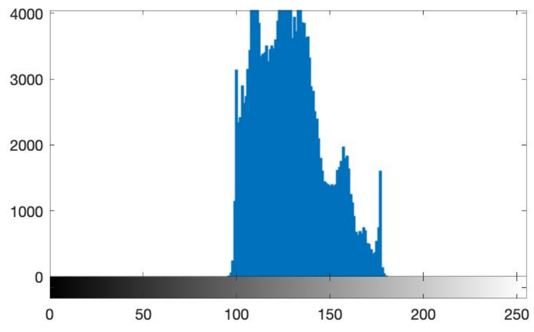


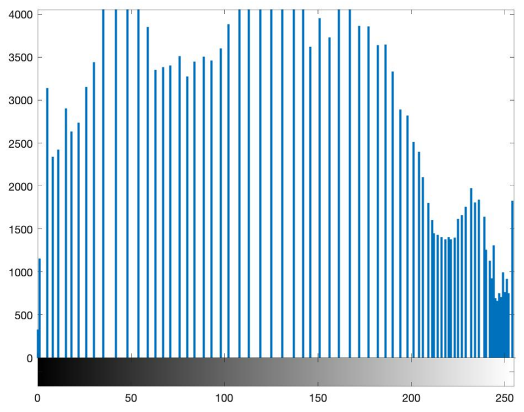
(c) Cumulative histogram.

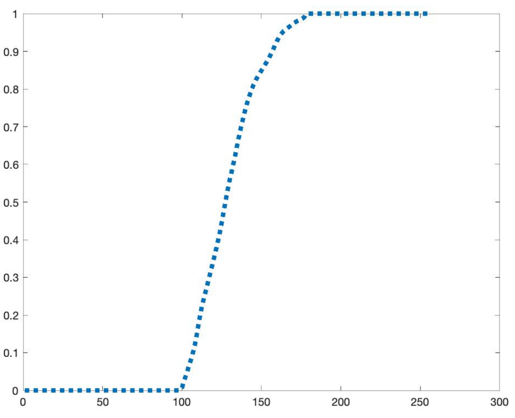
Figure 3: A cyclist and its histogram after equilization.


(a) Before Histogram Equalization.


(b) After Histogram Equalization.


(c) Histogram.


(d) Histogram after equilization.


(e) Cumulative histogram.

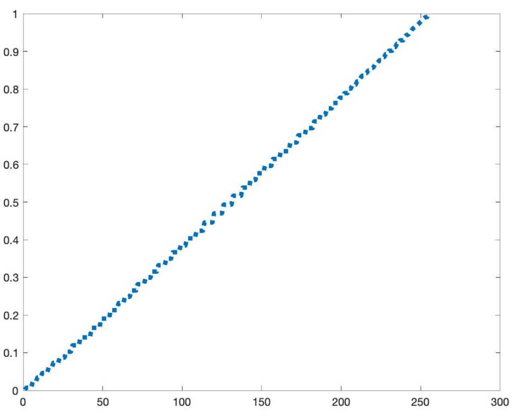

(f) Cumulative histogram after equalization.

Figure 4: A car before and after histogram equalization.

3 Final project assignment

The final project assignment is a compulsory part of the course and must be done by every student. **Due date for the assignment is 27/1/2022.** You must hand in the source code and a short report by this date, or you will not be able to complete the course. A form will open on the course web page to hand in your assignments. After that, you will have to defend your work during the oral exam.

3.1 Programming assignment

We described the histogram equalization technique for improving image contrast in the previous sections. Your task is to implement a parallel algorithm using CUDA based on the sequential code from Spletna učilnica. For simplicity, limit yourself to 8-bit grayscale images. Your algorithm should be able to process images of arbitrary dimensions. You will have to write multiple CUDA kernels to perform each of the three steps needed to equalize an image histogram:

1. **Compute the image histogram**
   To compute the histogram in parallel using CUDA you will have to employ atomic operations. An atomic operation guarantees that only a single thread can access a variable in memory. CUDA provides atomic operations to deal with race conditions (see "CUDA by Example", chapter 9). Furthermore, you should try and use shared memory to improve the performance of your algorithm.

2. **Compute the cumulative distribution of the histogram**
   Parallel computation of cumulative distribution is possible, even though the code given on Spletna Učilnica is strictly sequential. You can refer to the technical documentation given in `https://developer.nvidia.com/gpugems/gpugems3/part-vi-gpu-computing/chapter-39-parallel-prefix-sum-scan-cuda` for more details on how to perform this step efficiently using CUDA.

3. **Transform the original image using the scaled cumulative distribution as the transformation function**
   This step is straightforward to parallelize since you can perform the transformation for each pixel independently of the others. The exception is the problem of finding the minimum value in the cumulative distribution (hint: reduction algorithm from the lectures).

All of the above steps must be efficiently parallelized using CUDA to be able to achieve the highest grade for the project. Your code will be tested for plagiarism. If found guilty, the students involved will fail the course. For a passing grade, you should implement in CUDA at least the first step (histogram computation) using the atomics and the last step (which is straightforward). Consult your teacher if in doubt!

3.2 Report

After completing the programming part of the assignment, you will need to document your work in a short report. Your report should include the following:

1. **Table of contents**

2. **Introduction**
   Give a short description of the image equalization algorithm.

3. **Parallel implementation of histogram equalization in CUDA**
   For each of the steps of the histogram equalization algorithm, describe your parallel CUDA implementation. Then, explain why you chose the described approach and comment on possible drawbacks. **You should write down which steps did you manage to implement in CUDA!**

4. **Experiment**
   In this section, you should provide details about your experimental setup. Your task is to measure the execution time of both CUDA and sequential implementation. First, give the hardware specifications of the computer on which you ran the experiments. Next, provide details (dimensions) about the input images used in the experiment (at least 10 images of different sizes). Finally, describe your methodology for measuring execution time (you should repeat each measurement at least 10 times and compute the average).

5. **Results and discussion**
   Provide a few examples of histogram equalization on images. Compare the execution times of your CUDA implementation and the sequential implementation (chart, table) and compute the speed-ups. Establish the point (size of image) where CUDA becomes faster than sequential code and comment on the findings. Include a caption for each figure, chart, table, etc., in your report. Remember to include units in your results.
6. **Conclusion**
   Summarize your findings and comment on any possible improvements to your code.

You can write the report in Slovene or English language. Please check your spelling and grammar before you hand in the report!