Image histogram equalization using parallel processing in OpenCL

What is Image histogram equalization ?

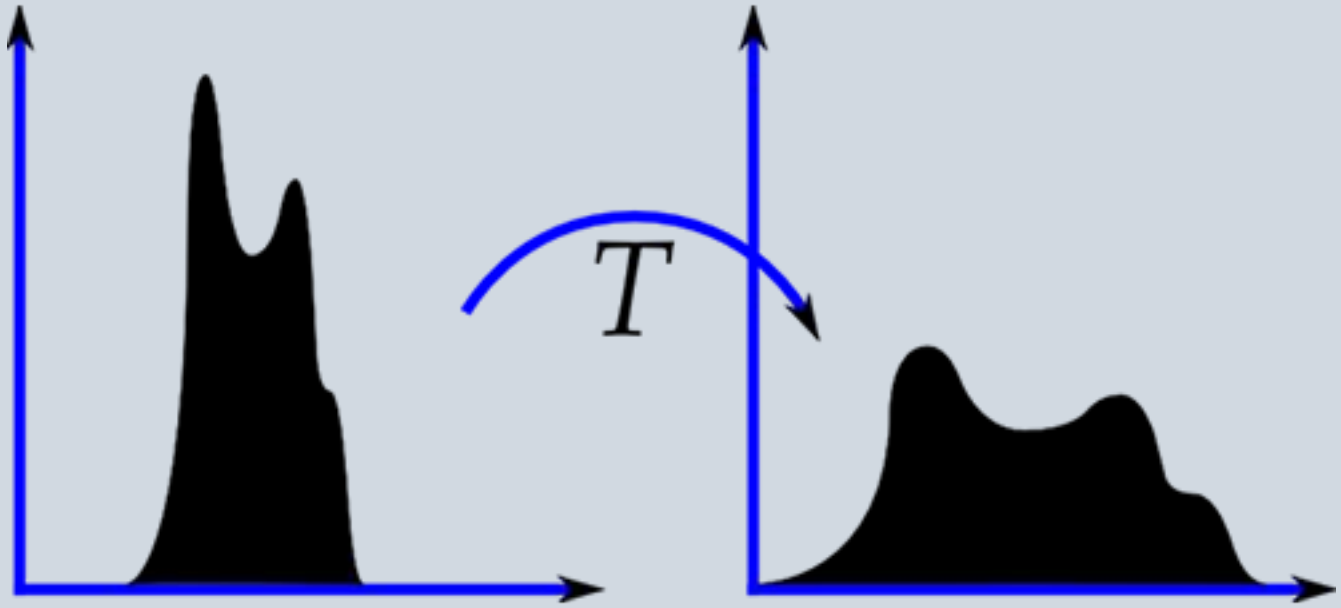
- The histogram equalization is an approach to enhance a given image. The approach is to design a transformation *T(.)* such that the gray values in the output is uniformly distributed in [0, 1] (Where 0 = Black and 1 = White).

- The concept of histogram equalization is to spread otherwise cluttered frequencies more evenly over the length of the histogram.

- A good histogram is that which covers all the possible values in the grayscale used. This type of histogram suggests that image has good contrast and the details in the image can be observed more easily.

- In histogram equalization we are trying to adjust the image contrast by applying a gray level transform which tries to flatten the resulting histogram.

- The following diagram shows the histogram equalization



Contrast correction using histogram equalization

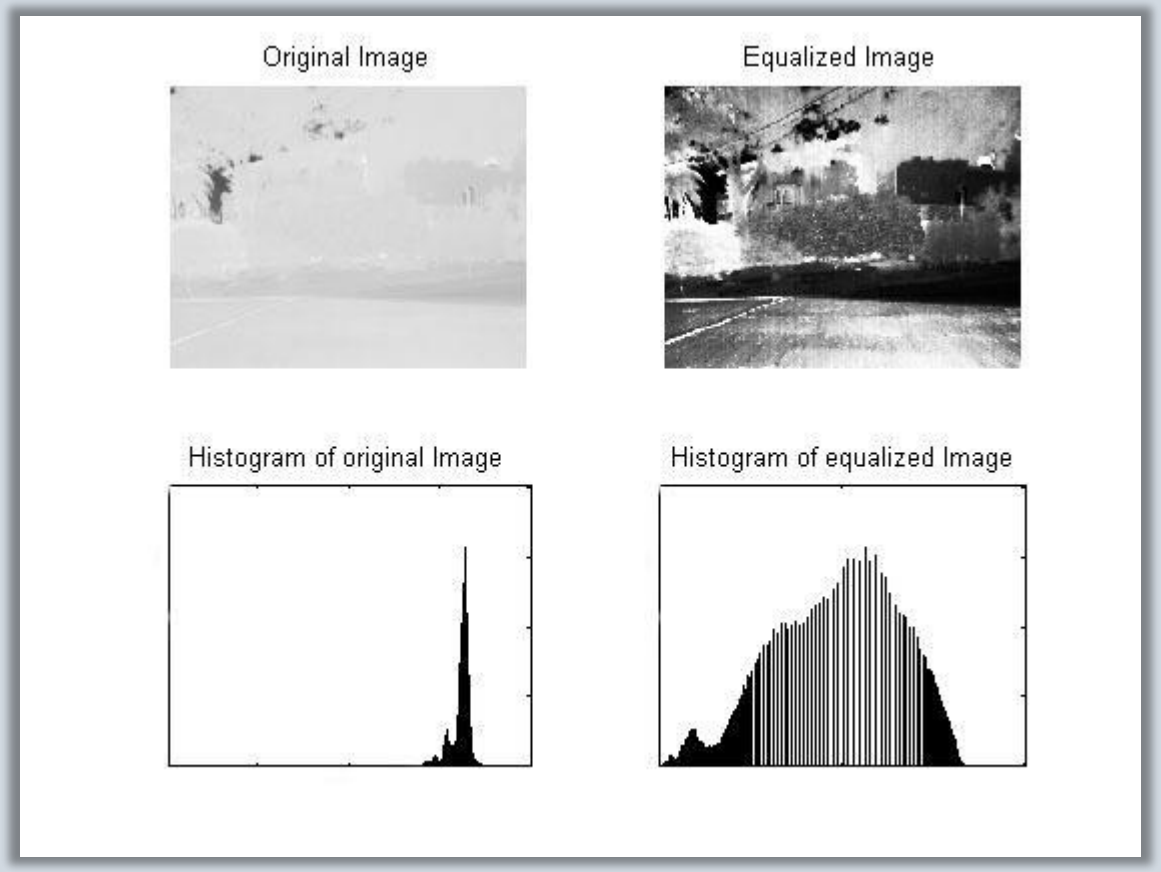
- Histogram equalization is a method in image processing of contrast adjustment using the image's histogram.

- This method usually increases the global contrast of many images, 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.

The image shows the effect of histogram equalization on highly exposed image. The high exposure caused image's histogram to towards the brighter tones. So many details in the image were not clear.

When the equalization is applied on the histogram , the resulting image has better details and histogram is more even than previous version.



Histogram equalization algorithm

- We use the method of linear stretching in equalizing the histogram.
- In this method we use the stretch operation which re-distributes values of an input map over a wider or narrower range of values in an output map.
- The input values of a map are re-scaled to output values in the output map. Input values are specified by the 'stretch from' values; the lower and upper 'stretch from' boundary values are included in the stretching.
- We can formulate

$$\frac{(\text{input value} - \text{Lowest input value})}{(\text{Highest input value} - \text{Lowest input value})} = \frac{(\text{Output value} - \text{Lowest Output Value})}{(\text{Highest Output value} - \text{Lowest Output value})}$$

- Here, Highest output value = 255 and lowest output value = 0
- So the formula for new value of any pixel in intensity matrix will be as

$$\text{Output value} = 255 \times \frac{(\text{Input Value} - \text{Minimum Pixel value})}{(\text{Maximum Pixel value} - \text{Minimum Pixel value})}$$

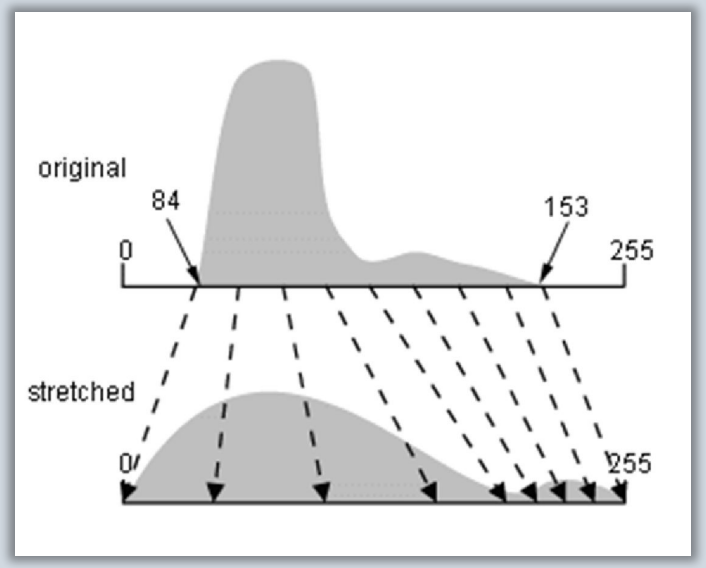


Figure : The figure shows the equalization of input image with lowest pixel value of 84 and highest pixel value 153, is mapped in the output domain with the lowest pixel value 0 and highest pixel value 255.

Parallel processing using OpenCL (GPU Computation)

- OpenCL™ (Open Computing Language) is open programming standard for general-purpose computations on heterogeneous systems.
- OpenCL™ allows programmers to easily target multi-core CPUs and GPUs.
- With OpenCL we can have many workers each executing a small piece of the work instead of a single worker doing all the job. The 1000 sums are executed at the same time, in parallel.
- After OpenCL came in action this parallel processing become more important and popular. For math and heavy calculation this technology would be a great combination of work.

Why OpenCL ?

- The OpenCL standard defines a set of data types, data structures, and functions that augment C and C++.

- **Portability :** Every vendor that provides OpenCL-compliant hardware also provides the tools that compile OpenCL code to run on the hardware. As long as all the devices are OpenCL-compliant, the functions will run. This is impossible with regular C/C++ programming, in which an executable can only target one device at a time.

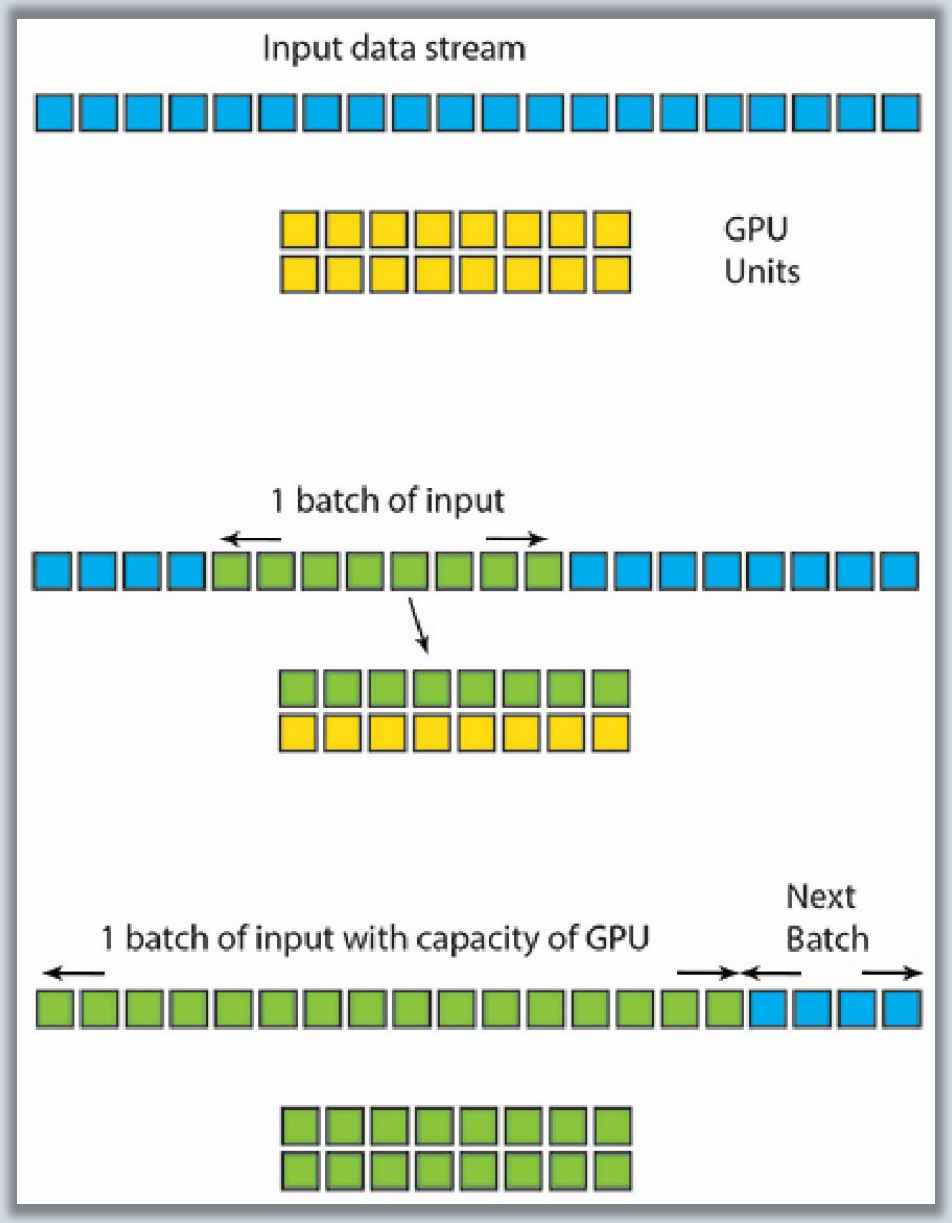
- **Parallel Programming:** it enables *parallel programming*. Parallel programming assigns computational tasks to *multiple* processing elements to be performed at the same time.

Parallel Implementation of algorithm

- To process the data in parallel manner, we first pass it onto the parallel processing unit.
- The passed data then, according to the capacity of parallel processor, is processed batch-by-batch.
- The size of the batch is same as the GPUs capacity of processing data at one time parallel, so the batches of data are processed sequentially and the data in every batch is then processed parallel manner.
- For one batch of data, the every work item on GPU, gets its own personal ID and then every work item is processed on one of the cores of GPU.

The figure shows how the data is passed on the GPU.

The input data stream is passed on the GPU batch-by-batch and every batch of data is called work group. Each work group has the data to be processed parallel they are called work items. The work group of the data is passed in sequential manner.



Results and timing comparison

Using profiling we get the following results for different data input

S. No.	Size of Input Matrix	Execution Timing on CPU (Parallel processing)(ms)	Execution Timing on GPU (Parallel Processing)(ms)	Execution time on CPU (sequentially Processing)(ms)
1.	100*100	0.005	0.069	0.85
2.	200*200	0.026	0.076	0.92
3.	300*300	0.072	0.178	1.98
4.	400*400	0.159	0.279	3.23
5.	500*500	0.248	0.351	4.81

One thing can be noted from this table that parallel processing on CPU is taking less time to execute the instructions, than GPU. But when very high amount of data is passed than time taken by GPU is less. We can see from the table as well that rate of growth of time consumption is higher in CPU.

The graph shows that for the less amount of data parallel processing on CPU takes lees time than on GPU, but the rate of growth in time consumption on CPU is higher on CPU. So at high amount of data, GPU performs much faster than CPU.

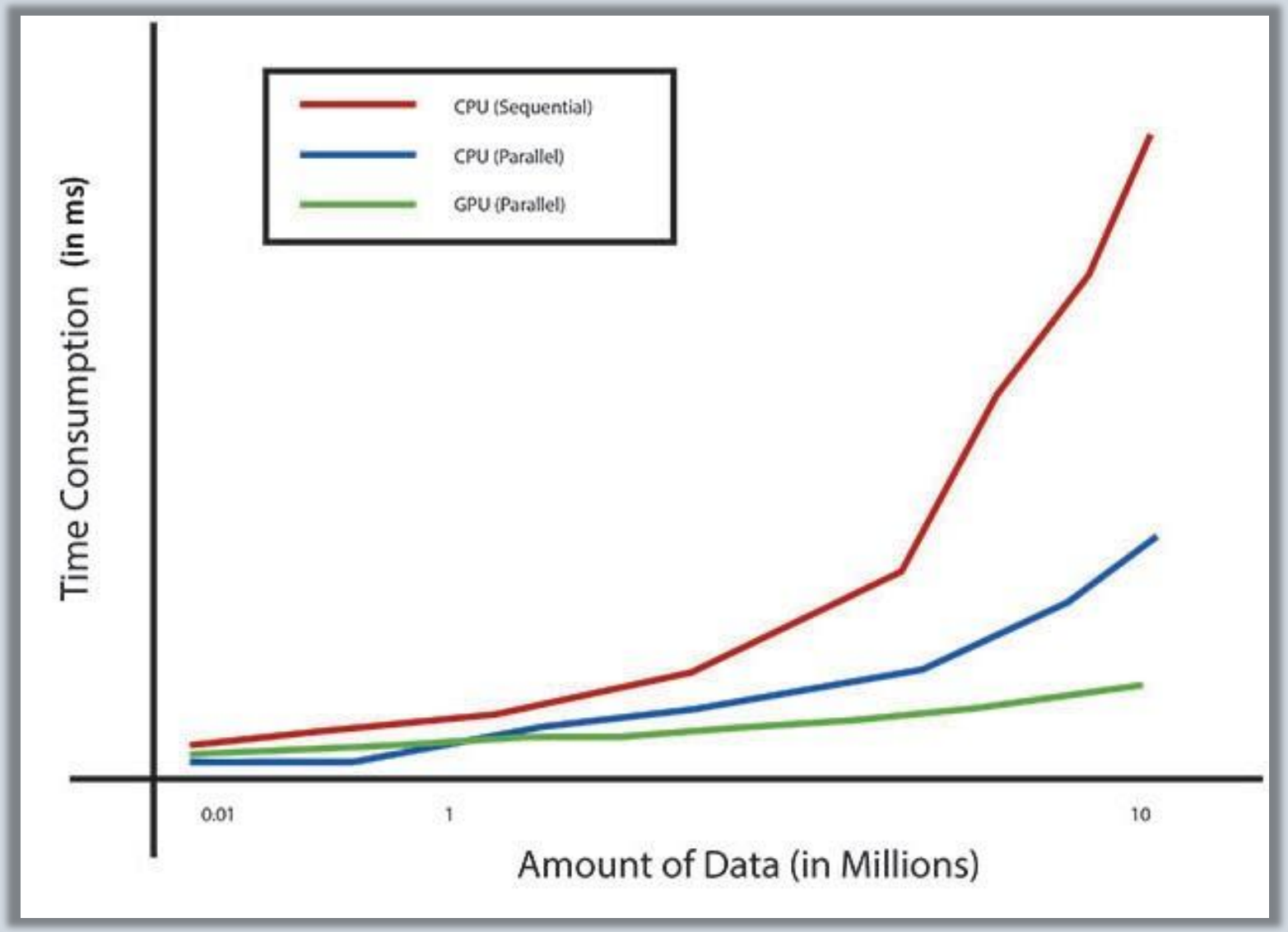


Fig. : graphical representation of time consumption on different processors and different amount of data.

Future Work

- This system can be included in different image editing software available in market that can make use of GPUs.
- As this algorithm automatically corrects the grayscale exposure. This can be used to make some automated systems for parallel processing multiple images, and by using the some part of GPU for every image, these one image's data can also be processed in parallel.
- This system can be used to increase the contrast in x-ray to adjust the exposure of bone structure. Also other medical reports that include images can be corrected.
- The system can be used in correcting images taken by spatial satellites, as due to different environmental situations, image exposure may not be good, and taking millions of high resolution picture again can cause time as well as economical loss. Applying this method to process in parallel can save time and money as well.
- Extending algorithms based on same concepts can also be implemented to make powerful tools to manipulate the colour images as well and can be extended to different image formats like RGB, Lab Colour Mode, HSL and CMYK