COMPARISON OF PERFORMANCE BETWEEN CPU AND GPU ON POINT-TO-POINT IMAGE PROCESSING OPERATIONS

KUMAR UTKARSH, ADITYA BHARDWAJ, ANVESH VERMA, VIPUL GUPTA, SIDDHANTH IYER, MADHAV GINORIA, DEEPANSHU SEHGAL

CSE DEPARTMENT
BENNETT UNIVERSITY, Plot Nos 8-11,
TechZone 2, Greater Noida,
Uttar Pradesh 201310

SURVEY PAPER

Abstract

A very particular specialized area of Digital Signal Processing is Image Processing that makes use of various different operations both mathematical and algebraic like matrix operation, matrix transpose, derivative, convolution and methods like Fourier Transform. Functions such as these need higher processing capabilities as compared to operations carried out by computers on a daily basis. This is on the grounds that, attributable to larger picture measures and highly complex computations, CPUs can be disappointing and lacking since they require Serial Processing indefinitely. To solve this problem GPUs are used since their processing power and speed is greater than that of CPUs as a result of their Parallel Processing/Computation abilities.

A platform and a program that is fit for parallel computation and is furthermore a programming model, CUDA was created by NVIDIA and utilized in making GPUs which were likewise produced by them. Developers use it as a platform to generate parallel processing applications that are advanced in terms of computational needs.

This paper executes different image processing algorithms on both CPU and GPU. It showcases an approach to the purpose of point processing of digital images by making use of parallel computing, especially for grayscale, darkening, brightening, thresholding and change of contrast i.e. RGB to Grayscale filter. The point to point method is responsible for transforming every pixel in the image parallelly in contrast with the dominant sequential approach. This particular approach makes use of CUDA like a parallel programming platform on a GPU so as to take full advantage of the cores and use them to their maximum potential.

In this paper, processing power of some common Image Processing methods are going to be compared with each other on OpenCV's inbuilt CPU thereto of functions in GPU that make use of CUDA. Initial results indicate that CUDA secures better outcomes in the majority of the used filters. The only exception being when it comes to the negative filters that operate on images at lower resolutions where OpenCV landed better results, however making use of images with high resolution, performance of CUDA is much better than that of OpenCV.

1. Introduction

Signals refers to a mathematical function that signifies the change in a variable depending on one or more variables. Lots of physical phenomena can be termed as signals, for example, variation in capacitor voltage over time at an RLC circuit or the voice of a human being that attenuates over time or the temperature of a room at a specific spot etc. As mentioned earlier, signals are functions that have been mathematically defined having 1 or more independent or dependent variables. Even audio signals can be represented mathematically in terms of amplitude as a function with respect to time. The concept i.e. DIGITAL SIGNAL PROCESSING originated in the 1960s and 1970s. During this period computers were still considered a luxury; applications of Digital Signal Processing were curtailed to only a few critical sections.

These sections were:

- Sonar and Radar
- Oil Exploration in water and land
- Medical Imaging of body
- Discovery of Space and celestial bodies

The revolution of PCs from 1980s to 1990s, applications of Digital Signal Processing broadened and were not just for government projects but also were made available to the general public and therefore the Digital Signal Processing developed with the work and help of the businesses that saw profit during this newly conceived concept, and it started gaining popularity at a really fast pace.

Since images are composed of 2D signals, these signals are to be processed to be deployed into different applications, this directed way to a new subdomain which was to be explored, Image Processing. Image processing is a particular type of applied research area that spans from expert photography to various other fields such as meteorology, astronomy, medical imaging, computer vision, etc. Its aim is to enhance the information obtained from pictures so as to subsequently perform other jobs like classification based on images, feature extraction etc. Image processing is an expansive and highly time-consuming task, for instance when it comes to point-to-point processing, a grayscale picture of 1024×1024 pixels, requires a CPU to make a surplus of a million operations, and in the case of a colored image the number of operations are cross multiplied by the number of channels.

During the developmental stages of Digital Signal Processing, even the cameras improved and the quality of images and their sizes increased in concurrence with it, because of which processing of pictures started to consume more computational power and system RAM. Parallel processing was considered as an effective and quick path in light of the structure of images.

Parallel computing using CUDA was the backbone of many works. As of late, video cards or GPUs have become devices for registering a lot of data parallelly. NVIDIA built up the CUDA design that is liable for gathering cores of GPU in a vector that can be modified to lessen computational time over a lot of information. Utilizing GPU to parallelize jobs began numerous years back, for instance, Fung and Mann in 2004 proposed another architecture utilizing different GPUs for image processing and PC vision; they recorded critical acceleration over a CPU based implementation. In 2006 Farrugia et al built up the GPUCV library to accelerate the time required to process images by utilizing GPUs; they saw that computing time with GPUCV changes from 18 to 1.2 times quicker as compared to local OpenCV function. The introduction of CUDA resulted in design-based research of algorithms being the most essential job and the concept of operating a GPU was “forgotten” when it came to processing tasks. In the year 2007 Sham et Al suggested a proficient strategy to compute mutual information between pictures to improve the effectiveness of the calculation by twenty-five times when contrasted with a standard CPU-based usage. In the year 2008 Fung and Mann made use of CUDAs architecture to lend a hand in “converting pictures into numbers''. They got a significant acceleration from 9.8 until multiple times then CPU usage to almost about 21 times. Then in the subsequent year of 2008 Zhiyi et al was able to improve processing times for a few filters like edge detection, decode algorithms, DCT encode and histogram equalization; they achieved acceleration up to 200 times as compared to a mere 8 times in contrast to a CPU. In 2009, Tarabalka et al, utilized GPUs for real-time computation of large quantities of data registered by a hyperspectral image; they observed that their GPU based design ran faster by about 100 times.

This paper outlines a particular method, making use of point-to-point processing of pictures by making use of parallel processing in order to reduce computational delays. The said approach has been tried upon pictures of various different resolutions, using brightening, grayscale, thresholding, contrast of images, and darkening. The outcomes indicate that the filters that have been implemented in CUDA are quicker than C implementations and OpenCV functions.

2. Some Important Definitions

- Digital Image Processing - It is the computation of pictures in PCs via algorithms. The operation algorithms are implemented on matrices which contain pixels. To add reference, it is a sub-field of Signal Processing.
- Parallel Processing - It comprises breaking down and running program jobs on different microprocessors at once, thereby reducing computational delay. Parallel processing comes extremely handy in large matrix operations which takes greater computation time.
- CUDA - *Compute Unified Device Architecture.* It is a type of platform developed by NVIDIA for the purpose of parallel processing. It allows software programs to perform calculations by making use of both the CPU and GPU by sharing the processing load with the GPU. CUDA-enabled programs can

achieve significant increases in performance.

- OpenCV - OpenCV refers to machine learning software library and an open source computer vision. This library contains a surplus of 2500 algorithms which are optimized, which includes an extensive arrangement of both best in class and great PC vision and AI and ML algorithms.
- GPU - GPU stands for *Graphics Processing Unit*. It is a programmable processor which provides the fastest graphics processing. It is a stand-alone card plugged into the PCI Express (PCIe) bus which performs parallel operations.
- Grayscale - Grayscale image is a particular type of image in which the value of every pixel is a singular sample indicating only an amount of light; i.e., it consists of only data of intensity. These images are a kind of gray monochrome or black-and-white, composed exclusively of color shades of gray.
- Thresholding - Thresholding is a concept of digital image processing which is used for segmentation of images. It is generally used to make binary pictures by utilizing grayscale pictures as a kind of point of reference. It basically replaces every pixel in an image with a black pixel provided the image intensity I is less than some fixed constant T (i.e. $I<T$), or a white pixel if the intensity of image is greater than that constant.
- Histogram Equalization - It is an instrument which is utilized for the portrayal of the distribution in intensity in a picture. Equalization of histogram is a technique that enhances images that have bad contrast. Overall Histogram Equalization is about mapping image distribution wherein one image distribution is mapped to another one which is wider and uniform in terms of distribution of intensity values.
- Edge Detection - Edges are clearly the locations where the intensity of image changes sharply. Detection method of edge determines points of intensity change. Edge Detection algorithm is crucial in applications like lane detection, image segmentation.

3. Paper Discussion

A. Summary of the Published Papers

This survey paper incorporates studies from multiple published papers. The objective of the refereed research papers sums up to improvement in the quality of pictorial information for better processing and human interpretation of image data for representation, transmission and storage purposes. All of the referenced papers consisted of experimented implementations and comparison between the results based on CPUs and GPUs.

The papers give a brief on different modelling and simulation techniques used in parallel and distributed systems. Some papers discuss how image processing grosses much more time to perform convolution infiltration of image on CPU, since the computational need of image filtration is enormous in contrast to GPU which accelerates the processing. By comparison and analysis of implementations the papers state that GPU is an appropriate way for processing high scale load of data parallelism of high-density computing. The result of a paper shows that the execution on GPU can get a speedup of about 61% as compared with the filtering implementation on CPU. Others show significant progress in memory utilization as well as time utilization.

B. First Paper

In the first paper, OpenCV built in CPU and GPU functions were used for performance comparison. For this author performed different image processing operations like grayscaling, thresholding, histogram equalization, edge detection using OpenCV & CUDA libraries. They generated these time comparison graphs of different operations.

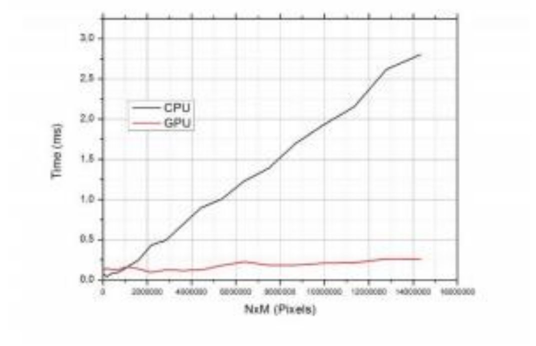


Fig. 17. The figure given above shows the comparison of built-in CPU and GPU functions that thresholds the image. While thresholding images, Otsu's method was used.

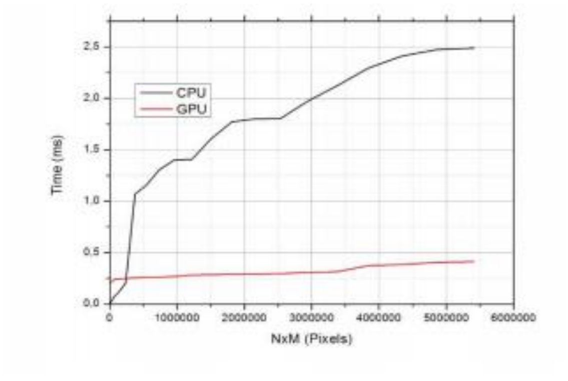
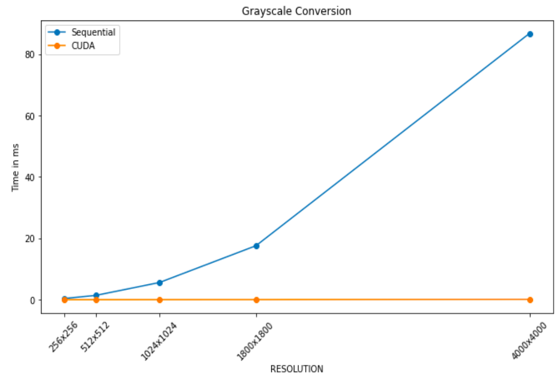


Fig. 18. The figure given above shows the comparison of built-in CPU and GPU functions that equalize the histogram of image.

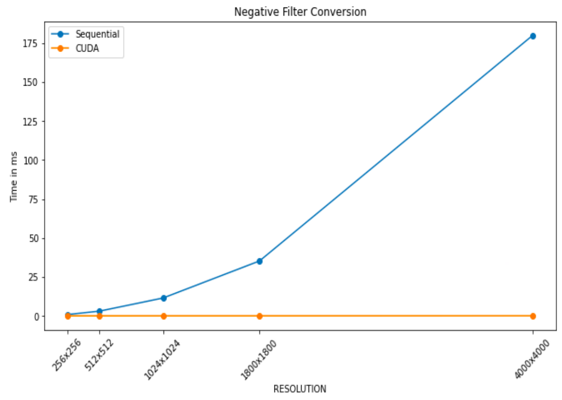
C. Second Paper

In the second paper authors compared OpenCV and CUDA performances based on different image filters. We implemented this paper on the same image with different resolutions to track the exponential speedup of parallel processing. We performed sequential and parallel processing operations on various image filters like greyscaling, negative filter, thresholding, brightening filter, darkening filter, inverse sinusoidal contrast and hyperbolic tangent contrast using OpenCV and CUDA libraries like Numba, PyCuda and CuPy. We obtained performance graphs with significant processing time difference between parallel and sequential image processing.

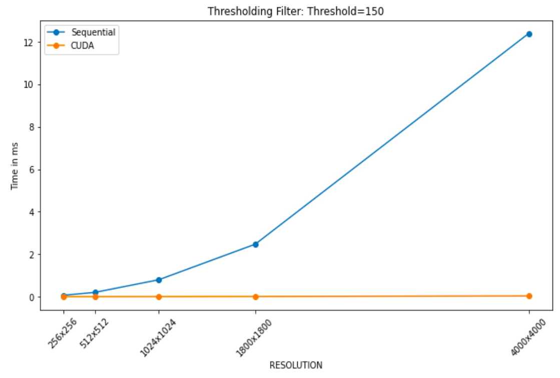
```
Average Speedup: 951x
```



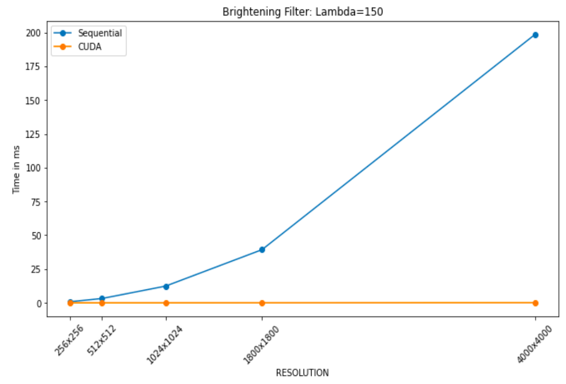
```
Average Speedup: 5884x
```



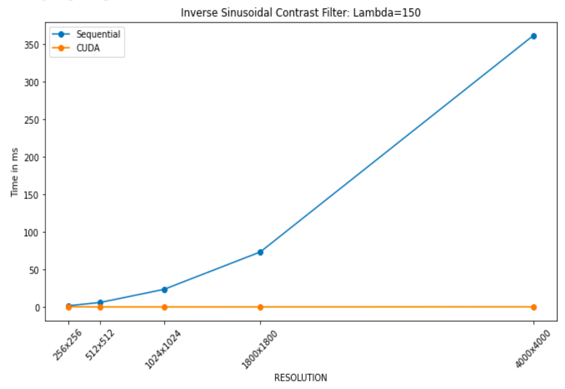
```
Average Speedup: 374x
```



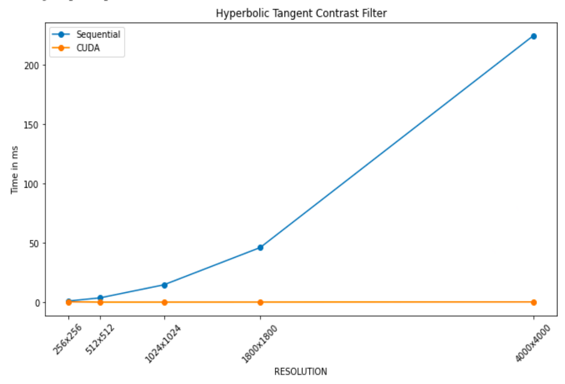
```
Average Speedup: 2629x
```



```
Average Speedup: 969x
```



```
Average Speedup: 623x
```



D. Third Paper

Third paper was on implementation of image enhancement algorithms using CUDA. In this paper the author implemented different filters like grayscaling, negative filter, brightening filter, darkening filter, low pass filter and high pass filter on CPU and GPU with variation in the number of threads per block and generated table of analysis to compare the performance results based on processing time.

Table 2 Negative Filter

No. of blocks	No. of Threads/block	Time (ms)
27000000	1	21.03
54000	500	29.02
27000	1000	44.32

Fig. 6 Output Negative Image

Table 4 Darkening Filter

Nu\. of blocks	No. of Threads/block	Time (ms)
27000000	1	36.71
54000	500	53.78
27000	1000	56.39

Fig. 8 Output Darkened Image

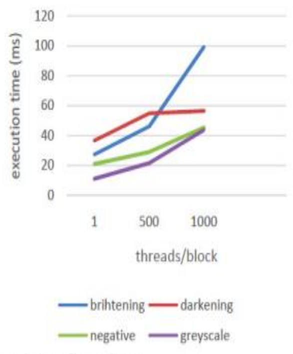


Fig. 9 Performance Analysis for Point to Point image filters on CUDA

Table 3 Brightening Filter

No. of blocks	No. of Threads/block	Time (ms)
27000000	1	27.36
54000	500	46.14
27000	1000	99.39

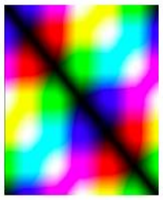


Fig. 7 Output Brightened Image

Table 1 RGB to Greyscale

No. of blocks	No. of Threads/block	Time (ms)
27000000	1	11.08
54000	500	21.61
27000	1000	42.86

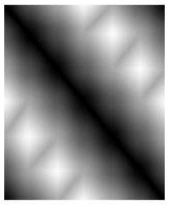


Fig 5 Output Greyscale Image

4. Conclusion

Through these papers we tried to compile and summarize results of various research papers on point to point processing of digital images by making the use of parallel computing and compared the results with sequential processing to justify advantages of parallel computing over sequential. For this we compared the time spent at some commonly used image processing operations using OpenCV's built-in CPU and GPU functions. Measurement results proved that GPU functions had better performance results because they run in parallel especially when image size increases. Parallelism and GPU significantly reduces the run time of algorithms of image processing. Further modules using CUDA were implemented for running in GPU instead of OpenCV's built in GPU functions and again we compared the results with OpenCV functions. The comparison results were

1. CUDA increases the performance power of some tasks that require millions of operations.
2. It is possible to get better results with OpenCV at times code is optimized and in parallel processing the performance gain may be not very significant.

However, from the comparison result we can conclude that CUDA obtained better results in most cases than OpenCV. We also implemented this section and found CUDA library was 165-5265 times faster than OpenCV functions which makes this difference very significant in some cases. Coming to image processing algorithms using CUDA, by observing and analyzing the experimental result we can conclude following results

1. In the applications which involve high inter-thread communication, we found that with increase in number of threads per block we obtain faster results.
2. In the applications that are parallel in nature or do not require inter-thread communication, we obtain better results with relatively lower number of threads per block.
3. Also, it was observed that the product of number of thread/blocks and number of blocks while implementation of image filters should be equal to number of elements to be processed.

5. References

1. **http://ijcsi.org/papers/IJCSI-9-3-3-1-10.pdf**
2. **'Performance Comparison Between OpenCV Built in CPU and GPU Functions on Image Processing Operations' by INTERNATIONAL JOURNAL of ENGINEERING SCIENCE AND APPLICATION Hangun and Eyecioglu, Vol.1, No.2, 2017**
3. **https://www.ijraset.com/fileserve.php?FID=6250**
4. **https://www.serialsjournals.com/abstract/99810_26-priya_nagargoje..pdf**

5. Parallel Processing of image in multi-core system IJCTA, 10(8), 2017, pp. 239-248 ISSN: 0974-5572 Link: https://www.serialsjournals.com/abstract/99810_26-priya_nagargoje..pdf

6. Implementation of Image Enhancement Algorithms and Recursive Ray Tracing using CUDA Proceedings of International Conference on Communication, Computing and Virtualization(ICCV) 2016 Link: https://www.sciencedirect.com/science/article/pii/S1877050916001976

7. http://www.ijiet.org/papers/106-T039.pdf

8. Mr. Diptarup Saha, Mr. Karan Darji, Narendra Patel, Darshak Thakore. "Implementation of Image Enhancement Algorithms and Recursive Ray Tracing using CUDA", Procedia Computer Science, 2016 Link: https://reader.elsevier.com/reader/sd

/pii/S1877050916001976?token=8DF16AB4F64EA246FF5D3D81EB95668673761C22E26ADABADA2B116F9BA23377BC027FF44C5D5C4045CC3B7386B8B3BB

9. https://www.ijert.org/research/using-opencv-over-matlab-for-implementing-image-processingapplication-on-cuda-gpu-to-achieve-better-execution-speedup-IJERTV6IS040455.pdf

10. https://www.ijert.org/research/different-optimization-strategies-and-performance-evaluation-ofreduction-on-multicore-cuda-architecture-IJERTV3IS041897.pdf

11. https://www.ijert.org/research/performance-optimization-of-highly-computational-tasks-usingcuda-IJERTV2IS2577.pdf

12. https://www.ijert.org/research/an-approach-to-optimize-the-speed-of-data-intenseapplications-with-parallel-computing-IJERTCONV2IS13136.pdf