International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE) Vol 2, Issue 3, Mar 2015

GPU Accelerated Image Processing using Bilateral Filter

[1]Sonal D. Nikhade,[2]Mayuri L. Kale,[3]Krutika Y. Bhagwatkar,[4]Anjali B. walke,[5]Dhiraj K. Thote
[1][2][3][4] Student, Department of Electronics and Telecommunication, YCCE, Nagpur, India
[5] Assistant Professor, Department of Electronics and Telecommunication, YCCE, Nagpur, India
[1]sonal.nikhade10@gmail.com,[2]mayurikale23@gmail.com,[3]kriti.bhagwatkar@gmail.com,[4]anjaliwalke11@gmail.com
,[5]dhira.net@rediffmail.com

Abstract: **The future of computation is the GPU, i.e. Graphical Processing Unit. They are developing into great parallel computing units with the promise that the graphics cards have shown in the field of image processing and the computational capability that these GPUs possess. Compute Unified Device Architecture i.e. CUDA is NVIDIA's parallel computing architecture. It enables dramatic increase in computing performance, by completely harnessing the power of the GPU. Initially we generate a MATLAB code for image processing using bilateral filter, and then we interface GPU with CPU using CUDA and create the related code of image processing for GPU for this filter. By changing the values of distinct parameters of the images, we have obtained variations in the output images and their corresponding computational time on CPU as well as on GPU. Further these computational times are compared and it is observed that the time taken by the GPU is 70 to 80% less than that of CPU.**

Keywords: **Image processing, bilateral filter, GPU, CUDA**

I. INTRODUCTION

Image Processing:

Image processing is any form of signal processing in which an image, such as a photograph or video frame, is taken as input and after processing the output may be either an image or a set of characteristics or parameters related to the image. Mostly, the image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it. It is the application of signal processing techniques to the domain of images — two-dimensional signals such as photographs or videos. Image processing typically involves filtering an image using various types of filters. Sometimes, images are often corrupted by random variations in illumination, intensity, or may have poor contrast and can't be used and processed directly for particular software or systems. So the filtering of the images is required. Basically, filtering transforms pixel intensity values to reveal certain image characteristics such as

–**Enhancement:** improves contrast
–**Smoothing**: remove noises
–**Template matching**: detects known patterns

There are different types of filters available for image processing such as linear filters or non-linear filters, time-invariant or time-variant, also known as shift invariance, causal or non-causal, analog or digital, discrete-time (sampled) or continuous-time, passive or active, infinite impulse response (IIR) or finite impulse response (FIR) type of discrete-time or digital filter. Since we are using only bilateral filter for processing, therefore we are going to discuss about bilateral filter which was introduced by Tomasi et al in 1998. It is a non-linear filter. It is used to preserve edges of an image and to reduce noise for smoothing the images as shown in figure 1 and 2.

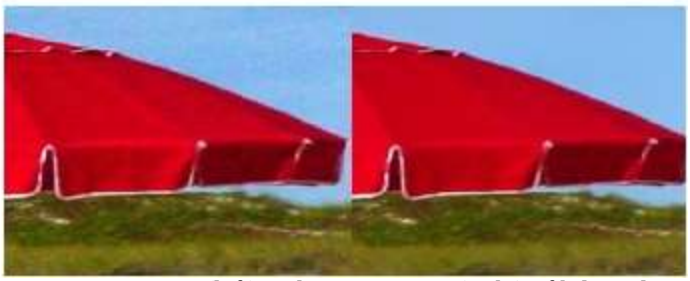

Fig.1: Input image (left) and output image (right) of bilateral filter

All Rights Reserved © 2015 IJERECE

Fig.2: The Bilateral Filter 8-Bit Source Image (left), Linear-Intensity Bilateral (middle), Logarithmic-Intensity Bilateral (right)

In bilateral filter, at a given location the value of the filtered image is a function of the input image values in a small neighborhood of the same location. In particular, Gaussian low-pass filtering computes a weighted average of pixel values in the neighborhood, in which, the weights decrease with distance from the neighborhood center, therefore, appropriate to average them together. The noise values that corrupt these nearby pixels are mutually less correlated than the signal values, so noise is averaged away while signal is preserved.

1) GPU (Graphical Processing Unit):

The primary goal of our project is to maximize computation speed of image processing. Performance can be accelerated by exploiting parallelism. The best example of this Parallel processing architecture is GPU which stands for graphical processing unit also referred as visual processing unit. It is basically a specialized electronic circuit which rapidly manipulates and alters memory to accelerate the image processing. GPU is used to calculate 3D functions as this type of calculations are very heavy on CPU to compute, thus increasing computational speed.

GPU are primarily designed for graphical purposes but now it is evolved into computing, precision and performance which lead into emergence of so called GPGPU. It stands for general purpose GPU and is fundamentally a software concept. CUDA architecture is used to achieve GPGPU with GPU architecture. Architecturally GPU consist of hundreds of cores that can handle thousands of software threads simultaneously while CPU composed of few threads with lots of cache and handle process sequentially. GPU supports data parallelism; in contrast of CPU that supports task parallelism. Also GPU has higher computational and memory bandwidth capabilities than CPU. GPUs are not used alone but are used along with CPU.

Fig.3: Comparison between CPU and GPU

The applications of GPU are in embedded systems, mobile phones, personal computers, work stations and play stations. In GPU accelerated computing, the cluster of GPU and CPU is used. Here sequential part of application is run on CPU and computationally intensive part accelerated by GPU. This delivers best value of system performance, price and power.

2) CUDA (Compute Unified Device Architecture):

CUDA is a programming interface of NVIDIA's GPU architecture which is featured in the GPU cards, for general purpose computing. This architecture is a set of library functions that can be coded as extension of C/C++ programming languages. CUDA provides great parallel computational power to the programmer which is provided by NVIDIA's graphics cards. An executable code is generated by compiler for CUDA device which is seen by CPU as a multi-core co-processor. GPGPU does not imply any memory restrictions on CUDA device. All the memory which is available on the device can be accessed without any restriction by CUDA. For different types of memories, there is variation in access time. This gives advantage to programmers to fully use the parallel computing power of the processor for general purpose computation. CUDA which is well suited for highly parallel algorithms, provide 128 cores which can communicate and exchange information with each other. To optimize the performance of algorithm to run on GPU, large numbers of threads are required. CUDA have thousands of threads executing in parallel which are going to execute the same code or function which is called as kernel. Each thread has its own ID, and based on its ID, it will determine to work on which pieces of data. A collection of threads is called a block which runs on a multiprocessor at a given time with multiple blocks assigned to a single multiprocessor and time-shared execution. A number of blocks are generated on a device with a single execution. A collection of all the blocks in a

All Rights Reserved © 2015 IJERCSE

single execution of code is called a grid. Like threads, each block is given a unique ID that can be accessed within the thread during its execution. Resources are divided equally amongst all the threads of all blocks executing on a single multiprocessor.

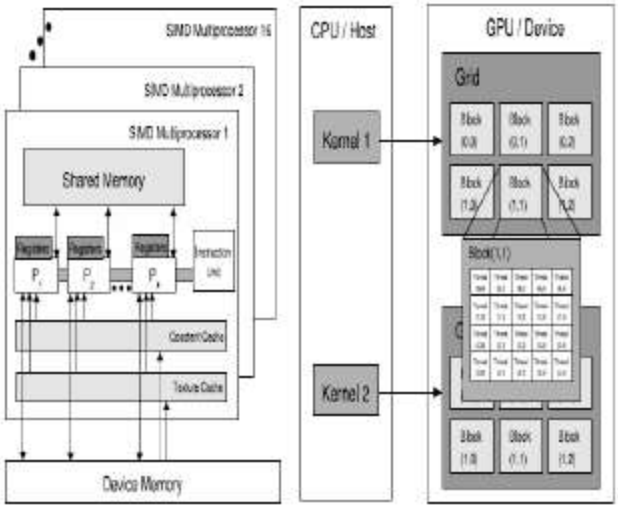

Fig .4: CUDA Hardware Interface and Programming Model

I. INTERFACING OF GPU WITH CPU

Fig.5a: PCI slots of CPU for interfacing with GPU

Fig.5b: Graphics card interfaced to PCI slot of motherboard

II. PROPOSED MODEL:

i. Block diagram:

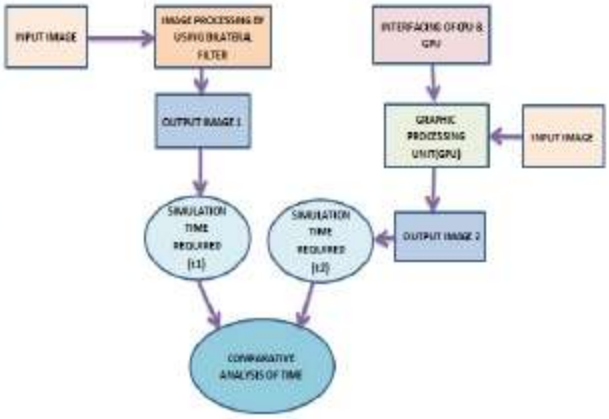

Fig.6: Block diagram of Proposed Model.

- Consider an input image.
- This image is initially processed by using bilateral filter.
- This processed image is the first output image.
- The time required for processing the input image (simulation time) using CPU without GPU is noted down as t1.
- Now again consider the same input image and process it by interfacing CPU with GPU.
- The second output image is obtained.
- Note the time required to obtain this output image (simulation time) as t2.
- Compare and analyze the two timings i.e. time t1 and time t2.

ii. Flowchart:

All Rights Reserved © 2015 IJERCSE

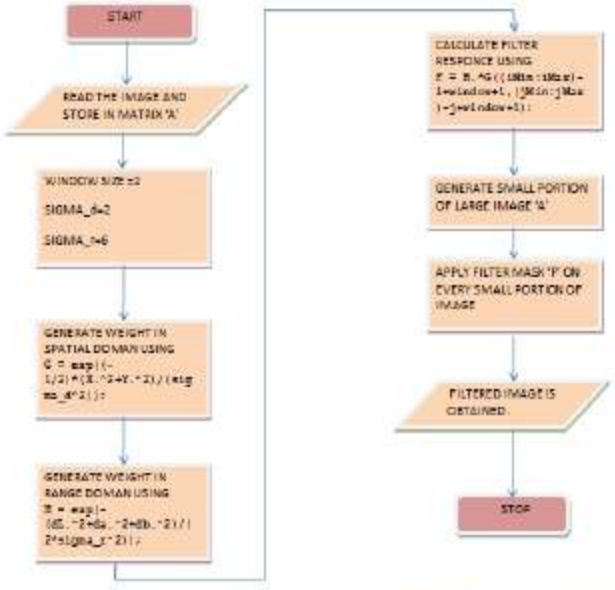

Fig.8: Flowchart of Bilateral Filter for MATLAB

- Initially the input image which is in matrix form, is read and stored in variable A.
- Here the window size is taken as 2 and the values of σ_d (sigma_d) and σ_r (sigma_r) are taken as 2 and 6 respectively and these values can be varied.
- Now using the following formulae, weights are generated in spatial domain as well as in frequency domain.

  G = exp((-1/2)*(X.^2+Y.^2)/(sigma_d^2))

H = exp(-(dL.^2+da.^2+db.^2)/(2*sigma_r^2))

- By using another formula, filter response is calculated as

F = H.*G((iMin:iMax)-i+window+1,(jMin:jMax)-j+window+1)

- The whole image is then divided in small portions and the filter mask is applied to each small portion.
- The filtered image is thus obtained and its simulation time is noted.

III. SIMULATED RESULT:

The input image is processed on CPU without GPU and the output images are obtained as follows:

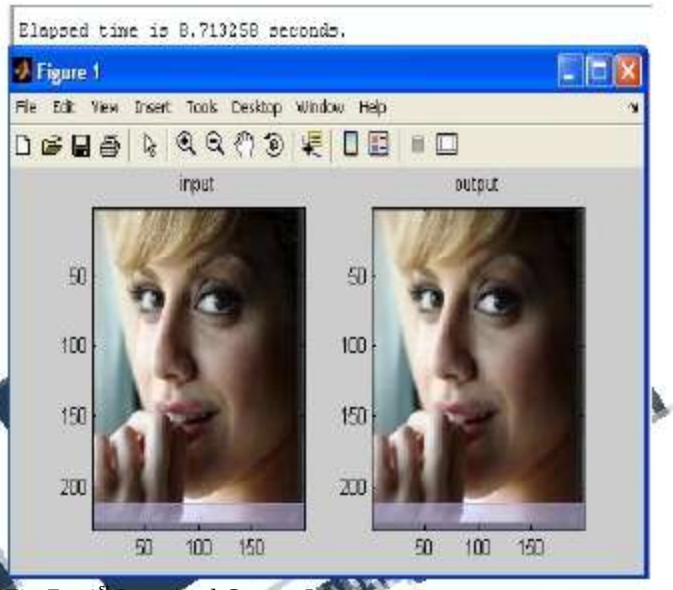

Fig.7a: 1st Input and Output Image

As we can see in the output image of fig (7a), the quality of object is improved and background is smoothened.

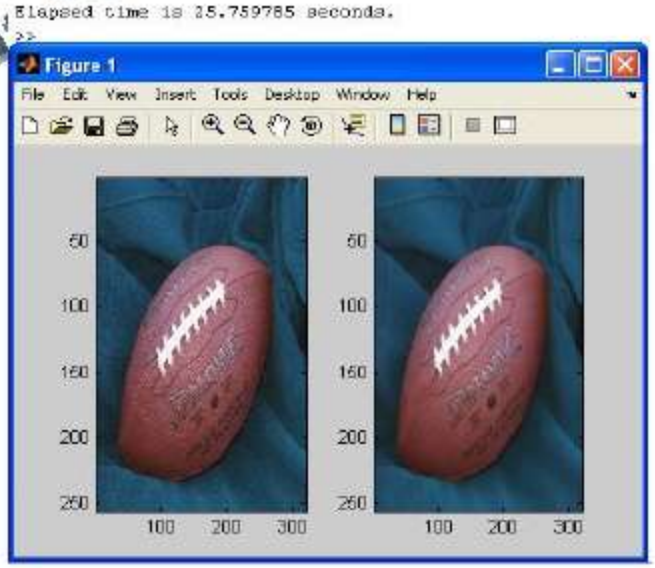

Fig.7b: 2nd Input and Output Image

The output image of fig. (7b) is smoothened and its edges get sharpened. Also, the details are enhanced.

All Rights Reserved © 2015 IJERCSE

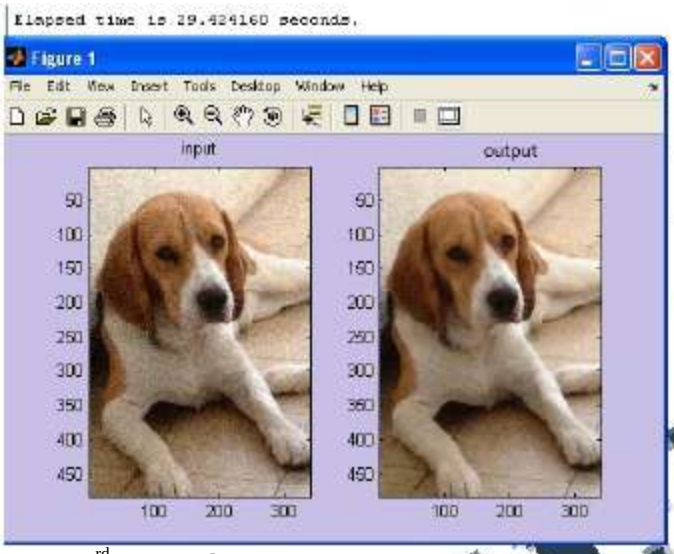

Fig.7c: 3rd Input and Output Image

From the top right corner of output image of fig (7c), we can observe that the background is smoothened and the edges of object are sharpened.

On CPU, with following specifications, we have run MATLAB code of bilateral filter and got the following results which are shown in the tabular form for above images:
Operating system: Microsoft windows XP professional
Processor: Intel(R), Pentium(R), 4 CPU 1.40GHz
Memory: 254 MB Ram
Page file:150 MB used, 2090 MB available

Table: The elapsed time of processed images on above CPU without GPU for distinct values of sigma_d and sigma_r (filter parameters):

Sr. No.	IMAGE RESOLUTION	CPU ELAPSED TIME	
		Sigma_d=2 sigma_r=6	Sigma_d=4 sigma_r=6
1.	199 x 230	8.713 sec	8.011 sec
2.	256 x 320	25.759 sec	15.942 sec
3.	339 x 483	29.424 sec	25.462 sec

The image of books(fig. 7d) of resolution 427 x 285 shown below, run on CPU with the specification descried above and GPU with following specification and their simulated times are compared in table:
Product name: Nvidia GT 520
Cuda core: 48 cores each of 1.3GHz
Memory space: 2GB RAM on graphics card
Bus support: PCI-E 2.0×16

IMAGE RESOLUTION	ELASPED TIME	
	ON CPU	ON GPU
427 x 285	28.861 sec	3.05sec

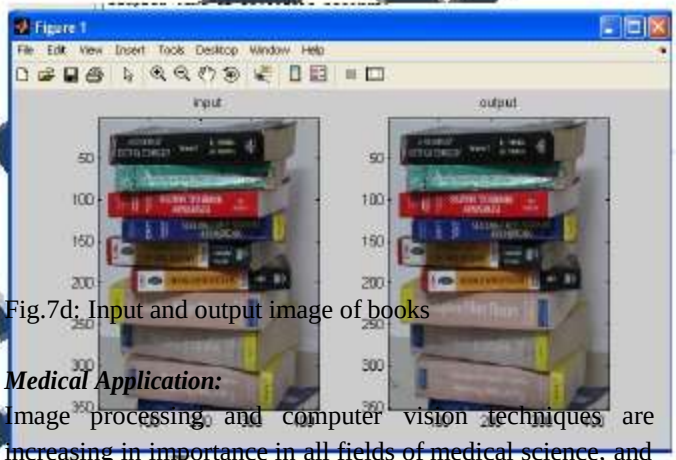

Fig.7d: Input and output image of books

Medical Application:
Image processing and computer vision techniques are increasing in importance in all fields of medical science, and are especially applicable to modern ophthalmology.

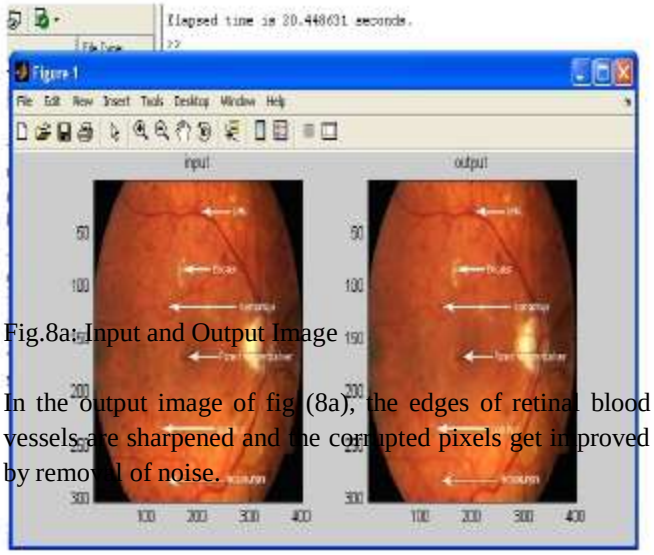

Fig.8a: Input and Output Image

In the output image of fig (8a), the edges of retinal blood vessels are sharpened and the corrupted pixels get improved by removal of noise.

All Rights Reserved © 2015 IJERCSE

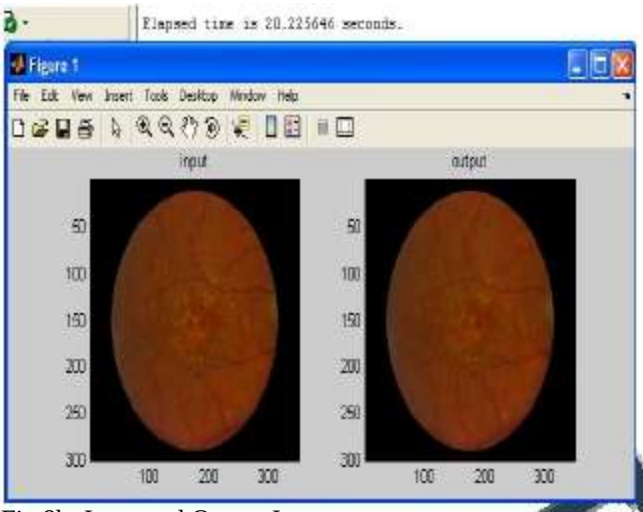

Fig.8b: Input and Output Image

In the output image of fig (8b), posterisation has occurred.

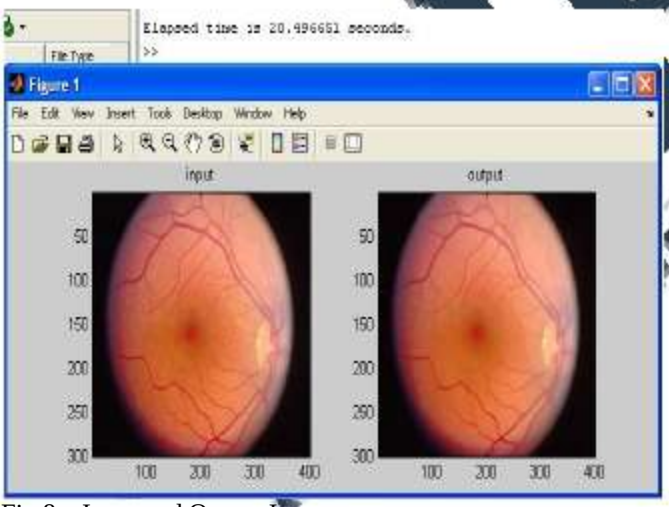

Fig.8c: Input and Output Image

In the output image of fig (8c), the main retinal blood vessels are highlighted with thorough smoothening and ramp-edge.

CONCLUSION:

Thus, from the results, we can conclude that after applying bilateral filtering on different images, the output images are smoothened by noise removal and preserving the edges. Also, for the images of different resolutions and for distinct filter parameters such as window size, sigma_d, sigma_r, etc, the simulation time changes accordingly. For higher resolution images, the computational speed of CPU is less and therefore the simulation time required is more. From the literature survey, we had said that computational time reduces and thus after interfacing GPU with CPU using CUDA architecture, the simulation speed is boosted which is about 70 to 80 % of original speed thus reducing the elapsed time of processing.

REFERENCES:

- Giuseppe Palma, Marco Comerci , and Bruno Alfano. Institutute of Biostuctures and Bioimaging, National Research Council of Italy. Salvatore Cuomo, Pasquale De Michele and Francesco Piccialli. Department of mathematics and applications, University of Naples Federico II, Italy. Pasquale Borelli, Department of advanced biomedical sciences, University of Naples Federico II, Italy. ―3-D Non-Local Means denoising via multi-GPU.‖ *Published by IEEE* (2013).
- Stephen W. Keckler, William J. Dally, Brucek Khailany, Michael Garland and David Glasco. NVIDIA. ―GPUs and the future of parallel computing‖. *Published by IEEE computer society* (2013).
- Jayshree Ghorpade, Jitendra Parande, Madhura Kulkarni, Amit Bawaskar. ―GPGPU processing in CUDA architecture.‖ (2013)
- Fernandes Palhano, Xavier de Fontes, Guillermo Andrade Barroso, Pierre Hellier. INRIA Centre de Recherche Rennes Bretagne Atlantique. ―Real time ultrasound image denoising‖. *Published in "Journal of Real-time image processing"* (2010).
- Bart Goossens, Hiep Luong, Jan Aelterman, Aleksandra Pizurica, and Wilfried Philips. Ghent University, ―A GPU accelerated real-time NLMeans algorithm for denoising color video sequences‖. *TELIN-IPI-IBBT, St. Pietersnieuwstraat* 41, 9000 Ghent, Belgium.(2009)
- Ben Wiess, Shell & Slate Software Corp. ―Fast Median and Bilateral Filtering‖. (2008)
- Aaron Lefohn, Joe M. Kniss, Robert Strzodka, Shubhabrata Sengupta, and John D. Owens. – Glift: Generic, efficient, random-access GPU data structures.‖ *ACM Transactions on Graphics*, 25(1):60–99, January 2006.
- David A. Bader and Kamesh Madduri. ―Parallel algorithms for evaluating centrality indices in real-world networks.‖ *In ICPP '06: Proceedings of the*

All Rights Reserved © 2015 IJERCSE

2006 International Conference on Parallel Processing, pages 539–550, Washington, DC, USA, 2006. IEEE Computer Society.

- John D. Owens, Shubhabrata Sengupta, and Daniel Horn. ―Assessment of Graphic Processing Units (GPUs) for Department of Defense (DoD) Digital Signal Processing (DSP) Applications.‖ *Technical Report ECE-CE-2005-3, Department of Electrical and Computer Engineering*, University of California, Davis, October 2005.
- Zhe Fan, Feng Qiu, Arie Kaufman, and Suzanne Yoakum-Stover. ―GPU cluster for high performance computing.‖ *In SC '04: Proceedings of the 2004 ACM/IEEE conference on Supercomputing, page 47*, Washington, DC, USA, 2004. IEEE Computer Society.
- Wen Wu and Pheng Ann Heng. ―A hybrid condensed finite element model with GPU acceleration for interactive 3D soft tissue cutting‖ *Research Articles. Comput. Animat. Virtual Worlds*, 15(3-4):219–227, 2004.
- Jens Kruger and Rudiger Westermann. ―Linear algebra operators for GPU implementation of numerical algorithms.‖ *ACM Transactions on Graphics (TOG)*, 22(3):908–916, 2003.
- P. J. Narayanan.‖ Single Source Shortest Path Problem on Processor Array.‖ *In Proceedings of the Fourth IEEE Symposium on the Frontiers of Massively Parallel Computing*, pages 553–556, 1992

All Rights Reserved © 2015 IJERCSE