Research Journal of Applied Sciences, Engineering and Technology 12(2): 223-238, 2016
DOI:10.19026/rjaset.12.2324
ISSN:2040-7459; e-ISSN: 2040-7467

Submitted: July 2, 2015 Accepted: August 15, 2015 Published: January 20, 2016

Research Article

Parallel Image Processing Techniques, Benefits and Limitations

Sanjay Saxena,Shiru Sharma and Neeraj Sharma
School of Biomedical Engineering, Indian Institute of Technology, Banaras Hindu University, Varanasi, India

Abstract: The aim of digital image processing is to improve the quality of image and subsequently to perform features extraction and classification. It is effectively used in computer vision, medical imaging, meteorology, astronomy, remote sensing and other related field. The main problem is that it is generally time consuming process; Parallel Computing provides an efficient and convenient way to address this issue. Main purpose of this review is to provide the comparative study of the existing contributions of implementing parallel image processing applications with their benefits and limitations. Another important aspect of this study is to provide the brief introduction of parallel computing and currently available parallel architecture, tools and techniques used for implementing parallel image processing. The aim is to discuss the problems encountered to implement parallel computing in various image processing applications. In this research we also tried to describe the role of parallel image processing in the field of medical imaging.

Keywords: GPU (Graphic Processing Unit), high performance computing, image processing, medical imaging, parallel computing

INTRODUCTION

Now days, Image processing plays a very essential role in numerous fields for example optics, computer science, mathematics, surface physics and visual psychophysics in case of computer vision its applications include remote sensing, feature extraction, meteorology, face detection, finger-print detection, optical sorting, astronomy, argument reality, microscope imaging, lane departure warning system (Basavaprasad and Ravi, 2014). Sometimes it takes so much time to execute several application for example point to point processing of a gray scale image of size 1024 X 1024 requires a CPU to make more than one million operations, for color image it is multiplied by number of channels and in the processing of images with high resolution (Olmedo *et al*., 2012).

In current years, parallel processing has become a significant tool for implementing high speed computing. For implementing this in image processing, several research and contributions have been done till now using several tool likes GPU (Graphical Processing Unit), CUDA (Computed Unified Device Architecture), Java, Hadoop and OpenCV and MATLAB (2014) and many more. However, it is very important to find most suitable technique of parallel computing for a particular application of image processing. In this review we have tried to overcome this difficulty by analyzing numerous algorithms and contributions by different scientists and researchers. This study summarizes existing parallel image processing techniques and tools implemented by different scientists and researchers. In this study we have also mentioned the benefits and limitations of the existing contributions, architectures and tools of parallel computing.

Image processing: Image is the two dimensional distributions of tiny image points called as pixels. It can be considered as a function of two real variables, for example, f(x,y) with f as the amplitude (e.g., brightness) of the image at position (x,y) (Saxena *et al*., 2013a). Image Processing is the process of enhancing and manipulation with an image in order to extraction of meaningful information (Olmedo *et al*., 2012). Image processing has become a useful research area that goes from professional photography to several different fields such as Astronomy, Computerized photography (e.g., photoshop), Space image processing (e.g., Hubble space telescope images, interplanetary probe images) Medical/Biological image processing (e.g., interpretation of X-ray images, blood/cellular microscope images, CT Scan, PET Scan), Automatic character recognition (zip code, license plate recognition), Finger print/face/iris recognition, Remote sensing: aerial and satellite image interpretations, Reconnaissance, Industrial applications (e.g., product inspection/sorting) (Kaur, 2013; Dougherty, 2009; Kamboj and Rani, 2013; Drakos, 2014;Aoki and Nagao, 1999). Thereare several techniques used in

Corresponding Author: Sanjay Saxena, School of Biomedical Engineering, Indian Institute of Technology, Banaras Hindu University, Varanasi, India

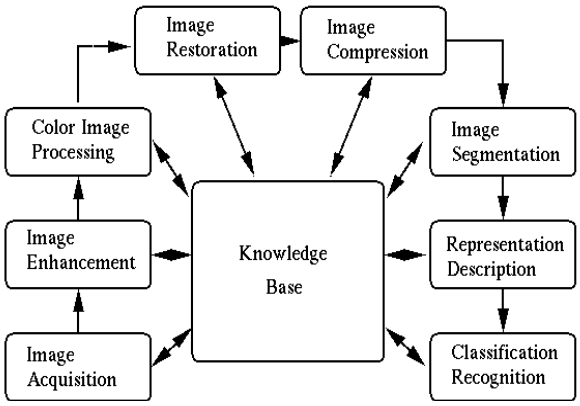

Fig. 1: Basic flow diagram of the steps of image processing (Bräunl, 2001)

image processing to enhance and extract information from images. Several Authorscategorized image processing into three groups (Soviany, 2003). Low level image processing, Intermediate Level Image Processing, High Level Image Processing(Aashburner and Friston, 2005).

Low level image processing: It usually converts image data into image data. Example Contrast Enhancement, Noise Reduction, Filter Transformations, Calculations of features of input images like contours, histogram etc.

Intermediate level image processing: These are more complex operations which derive abstractions from the image pixels like region labeling and object tracking.

High level image processing: This is knowledge based processing which concerns the interpretation of the information extracted from the intermediate level processing for example Pattern Recognition, Object Classification etc.(Soviany, 2003). Basic flow diagram of the different steps of Image Processing given as following (Drakos, 2014) in which different steps of Image Processing such as Image Acqutition, Image Preprocessing etc are in Fig. 1.

In this section we have seen about the different basic steps of image processing now we are going to give the brief description of parallel computing and its importance in image processing.

PARALLEL COMPUTING AND ITS ENVIRONMENT

Parallel computing or processing is the process of simultaneous uses of various compute resources to solve a computational job/task/work (Saxena *et al*., 2013b). Main principle of parallel computing is to divide a task in such a way that the task executes in minimum time with maximum efficiency. To implement parallel computing there can be several kind of parallel machine like a cluster of computers which is having multiple PCs combined together with an elevated speed network; a shared memory multiprocessor by connecting multiple processors to a single memory system, a Chip Multi-Processor (CMP) contains multiple processors (called cores) on a single chip (Saxena *et al*., 2013c; Fung and Mann, 2008; Edelman *et al*., 2006; Barney, 2014; Huang *et al*., 2011). There are several application of high performance or parallel computing in various fields describes in Barney (2014). There are several application area of parallel computing image processing, Atmosphere, Earth, Environment, Applied Physics, Nuclear, condensed matter Computer Science, Mathematics, Electrical Engineering and Many more discussed in Barney (2014)and Slabaugh *et al*. (2010).

Basic concepts of parallel computing: Barney (2014) gave the basic terminology which are generally used in parallel computing.

Node: It is an individual "computer in a box". Typically it is comprised of numerous CPUs/Cores/Processors, network interfaces, memory, etc. These are networked simultaneously to encompass a supercomputer.

CPU/Processor/Core: Previously, a Central Processing Unit was a particular execution part for a computer. After that multiple CPUs were included into a node. After that individual CPUs were subdivided into numerous cores, each being an exclusive effecting unit.

Task: This is a logically distinct section of computational effort. This is normally a program or set of commands which is executed by a core/processor. A parallel program, that consists of numerous tasks running on many processors.

Pipelining: It is the breaking of a task or job into steps performed by dissimilar processing units, in which inputs streaming through like an assembly stripe.

Shared Memory: As per hardware point of view, it is just like a computer architecture in which all cores/processors have straight access to regular physical memory. For programming point of view, it is a model in which concurrent tasks are having the simmiler picture of memory and it can directly address and access the similar logical memory locations in spite of the place where physical memory really exists.

Symmetric Multi-Processor (SMP): It is a hardware architecture in which several processors share a solitary address space and having capability to access all resources; shared memory computing.

Distributed memory: For hardware point of view, it is just like a network based memory access used for

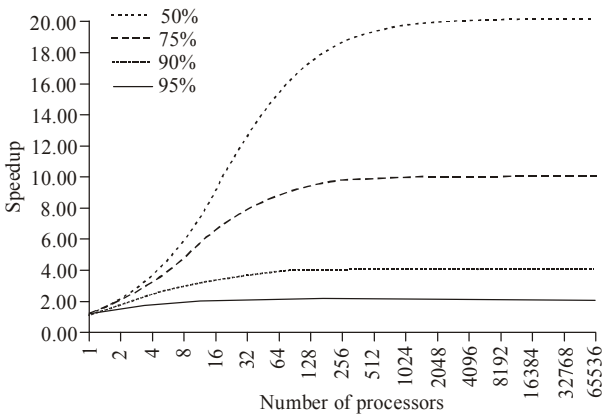

Fig. 2: Amdahl's law

physical memory. For programming terms, tasks can just rationally see local machine memory and have to use communications to access memory built in other machines where added tasks are executing.

Communications: Parallel jobs classically need to swap data. It can be accomplished by many ways, like via a shared memory bus otherwise over a network, though the real event of data exchange is normally referred to as communications in spite of the method employed.

Synchronization: It is defined as the synchronization of parallel jobs in genuine time, very frequently linked with communications. Habitually implemented by establishing a coordination point within an application where a job may not carry on further until a further tasks reach the same or logically comparable point. Synchronization generally consists of waiting by at least one job and can therefore cause a concurrent application's execution time to increase.

Granularity: In terms of parallel computing, It is a qualitative measure of the proportion of computation to communication.

Parallel overhead: That is the amount of time necessary to coordinate parallel jobs, as contrast to doing useful work.

Scalability: It refers to a parallel system's which is having ability to divulge a balanced increase in parallel speedup with the addition of additional compute resources.

Performance measures: It is having a set of metrics that is used for quantifying the quality of an algorithm (Navarro*et al*., 2014). If we talk about the quality of sequential algorithms it is usually evaluated in terms of time and space (Rajaraman and Siva Ram Murthy, 2006; Dougherty, 2009). However for the quality of the parallel algorithm it depends on the parallel architecture and the number of processors employed. Here we are going to describe the metrics and measures for analyzing the performance of the parallel computing system described by several authors.

Parallel run time: This is the time taken by a program which is executed on an n-processor parallel computer. When n = 1, T(1) denotes the sequential run time of the program in single processor (Rajaraman and Siva Ram Murthy, 2006).

Speedup (Navarro*et al*., 2014): This is the important measures of parallel computing. Basically it measures how much faster a parallel algorithm runs with respect to the best sequential one. For a problem of size *n*, the expression for speedup is:

$$Sp = T(n,1)/T(n,p) \quad (1)$$

where, $Ts(n,1)$ is the time of the best sequential algorithm (i.e., $Ts(n,1) \leq T(n,1)$) and $T(n,p)$ is the time of the parallel algorithm with p processors, both solving the same problem. Navarro*et al*. (2014) described when speedup is linear, when it is super linear and the different models of speedup like fixed time speedup, fixed size and scaled speed up. Now we are going to discuss speedup performance laws ie Amdahl's law, Gustafson's Law, Sun and Ni's law this is also known as laws of speedup.

Amdahl's law: It is often used to forecast the theoretical highest speed up using numerous processors. According to this Law "The speedup of a program using numerous processors in parallel computing is restricted by the sequential portion of the program". For example, for a program if 95% of that can be parallelized, then the theoretical greatest speedup using parallel computing would be 20 times as shown in the following Fig. 2, it doesn't matter the number of processors are used.

Gustafson's Law: This law says that increase of the problem size for larger machines can retain scalability with respect to the number of processors (Zhou *et al*., 2012; Rajaraman and Siva Ram Murthy, 2006).

Sun and Ni's Law: This one is referred to as a memory bound model. It turns out that when the speedup is computed by the problem size limited by the available memory in n-processor system, it leads to a generalization of Amdahl's and Gustafson's law (Zhou *et al*., 2012; Rajaraman and Siva Ram Murthy, 2006).

Efficiency: It measures how the processors are efficiently used in a parallel program. It can be

expressed by the following equations (Zhou *et al*., 2012; Rajaraman and Siva Ram Murthy, 2006):

$$Ep = \frac{Sp}{p} = \frac{Ts(n,1)}{pT(n,p)} \leq 1 \quad (2)$$

where, Sp is the speedup, Ep is the efficiency of an algorithm with p processors and $Ts(n,1)$ is the time of the best sequential algorithm (i.e., $Ts(n,1) \leq T(n,1)$) and $T(n,p)$ is the time of the parallel algorithm with p processors, both solving the same problem. These above are the chief performance measures of a parallel program used by different researchers apart from these there are other several other performance matrices are like Navarro*et al*. (2014) described Work and Span, Flops, Performance per Watt and Memory Bandwidth as important metrics of measurement of parallel program.

Need of parallel computing in image processing: There are several studies have been till now which describe the requirements of parallel computing in image processing. As we have already discussed that processing of a gray scale image of size 1024 X 1024 requires a CPU to make more than one million operations for color image it multiplied by number of channels (Olmedo *et al*., 2012). So efficiently implementation of parallel computing can reduce the processing time.

Several techniques of image processing requires parallel computing described by different researchers like working with Images with High resolution in Fung and Mann (2008) authors described that Images of size 10000 X 10000 pixels requires sufficient computing power to perform operations with in time. Akgün (2013) described about the performance evaluations for parallel Image filter on multi-core architectures using Java Threads in this authors have developed image convolution filters, Basic image processing techniques like contrast enhancement, brightness improvement also need high computation power as these are having several time consuming steps, Alda Kika and Greca (2013) also discussed applications of image processing using java threads and many more researchers have been done researches in this field till now. In the following comparison of benefits and limitations we have also discussed our previous developed approaches (Saxena *et al*., 2013a).

In the next section we are providing a benefits and limitations of the above given techniques by different authors.

EXISTING LITERATURES FOR IMPLEMENTING PARALLEL IMAGE PROCESSING, BENEFITS AND LIMITATIONS

Following Table 1 illustrates the benefits/advantages and Comments/improvement area of the different contributions giving different researches.

According to Table 1 we can see that till now there have been developed numerous approach for implementing parallel image processing using GPU, CUDA, Hadoop, OpenCV, OpenCL and many more. Some of them are very useful and informative. However, there are some other methods also implemented by different researchers which are not described in Table 1.

Now we are going to give the brief comparison of different parallel implemented image processing algorithms GPU and CUDA in terms of time

Table 1: Analysis of different parallel implementation of image processing algorithms

Implemented image processing concepts	Tools/Techniques used by different researchers for implementing parallel image processing	Benefits	Comments	References
Image convolution filters	Multithreading using Java on multi core computer	Good speed up and parallel efficiency with suitable multithreading. Ability to execute in multi core computer	This method needs more data to test with different performance measuring parameters.	Akgün (2013)
Contrast enhancement, brightness improvement and steganography	Multithreading using Java on multi core computer	Efficient use of multithreading using Java. Good communication between threads.	High level image processing needs to be done for validation.	Kika and Greca(2013)
Different Image Processing application	Parallel Virtual Machine(PVM) MATLAB using Mex Files external interface API and C language on Linux OS	Not required comprehensive skills to write a parallel program.	In this the software has been tested to work with double matrices. Data redistribution and data dependency analysis need to be done.	Manjunathachari and SatyaPrasad (2005)
RGB to Gray conversion, image morphology, integral of image	CUDA Programming with GPU	Speed up is good.	Need to be tested variety of images with large dimension.	Marwa *et al*. (2014)

Table 1: Continue

Survey on CUDA	This paper reviews all different techniques of parallel image processing and gives brief introduction of CUDA.	It gives brief introduction of GPU based parallel image processing and Brief Introduction of CUDA. In this contribution critical analysis has been done with different parallel algorithm implemented sequentially and parallelly.	More algorithms need to be evaluated.	Kaur and Nishi (2010)
Geometrical transformation, image De noising, Edge detection	GPU with CUDA and OpenGL	Significant speed UP in most of the algorithms, worked on most time consuming steps	Variety of images is not tested. High level image processing needs to be done.	Mahmoudi *et al*. (2009)
Bilinear interpolation, watershed segmentation and volume rendering	MATLAB with Mex Files	Efficient use of MATLAB with comparison with C.	Good programming skills needed, now MATLAB pool can be used.	Bister *et al*. (2007)
Graphic rendering for 3D polygon model	GPU with CUDA	Results are good.	Need to test variety of rendering data.	Ji-Hoon*et al*. (2014)
Retinal Blood extraction using Kirsch's template	MATLAB with parallel computing toolbox	Proficient use of different core of a system	Specific agenda has been done	Roy (2013)
Gaussian, median, average and motion filters	Digital signal processor	Different parameters like time, MSE, PSNR, Overlap Factor is calculated	Lots of data need to be tested with variety of images	Iqbal and Raghuwanshi (2014)
Basic image processing concepts like sobel filters and many more.	Image processing and parallel computing toolbox in MATLAB with GPU and CUDA	Efficiently use of the tools of MATLAB	Needs different types of data to be tested.	Georgantzoglou *et al*. (2014)
Face detection algorithm	CUDA integrated with Hadoop distributed network	Throughput is good.	Need to show more results	Malakar and Vydyanathan (2013)
K Means clustering	Parallel computing toolbox in MATLAB	Speed UP is good working with CPUs	Need to implement this algorithm with GPU to get significant results other measuring parameters need to be calculated.	Ahmed (2014)
Medical Imaging: Parameter optimization for lung texture using SVM, Content based medical image indexing, 3D directional wavelet analysis	Cluster of heterogeneous computing nodes using Hadoop	Optimization performed better. High level image processing algorithms are tested.	Need to comparison of speedup obtained.	Markonis *et al*. (2012)
Morphological image processing	Device Cyclone II EP2C35F672C6, Image Processing Libraries of MATLAB	It used highly parallel configurable architecture. Able to execute a morphological operation in all of the image pixels in a single cycle.	The work has been done only on black and white images.	Pedrino and Fernandes (2014)
Sobel edge detection,	Hadoop	Speed Up is good.	Needs more different measuring parameters to be calculated.	Fernandez and Kumar (2009)
Medical image processing, texture feature calculations involving the correlation of the data with the Gabor and Gaussian filters, FFT	MATLAB, Intel Parallel Studio XE 2013 software development kit, OpenMP for multithreading and Intel Math Kernel Library	Efficient use of Multicore with tremendous speed up.	More measuring parameters needs to be tested.	Low (2013)

Table 1: Continue

Medical image processing, image filtering algorithm with FFT	Parallel Computing toolbox, MATLAB, GPU, CUDA	Efficient use of PCT of MATLAB using GPU and CUDA.	More measuring parameters for performance evaluation needs to be tested.	Pan (2013)
Average filter	Network of Sun SPARC Station 5 workstations connected with standard (10Mbit/s) Ethernet and Parallel Image Processing toolkit with MPI	Tested dynamic load balancing algorithm and significant speed up	Needs to form cluster. Other parameters of parallel computing need to be tested.	Squyres *et al*. (1995a)
Large scale mosaic and stereo and stereo image correlations.	Message Passing Interfaces(MPI)	Execution time is drastically changed by utilizing image correlation quality.	Cluster of computer needed. Now, it can be done by GPUs. It needs more types of data to be tested.	Klimeck *et al*. (2003)
Video database processed, extraction of features from video images	MapReduce on Hadoop	Tremendous extraction of features.	Variety of data should be tested.	Yamamoto and Kaneko (2012)
Point operators, local operators, ditherning, smoothing, edge detection, morphological operators & image segmentation	Parallaxsis and it can be used in data parallel system	Several frequently used image processing algorithm is implemented with good results.	In current scenario numerous tools for implementing parallel image processing are available.	Bräunl(2001)
Hyperspectral imaging	Multi cluster system	Remarkable speed of all algorithms	Efficient use of GPUs can enhance this implementation.	Fangbin *et al*. (2011).
Face recognition	IMAP-board	Easy and efficient way of implementation.	Big set of data is requisite to test.	Fatemi *et al*. (2004)
Extraction of digital slices from 3D images.	Computer-Aided Parallelization tool, A pre compiler containing C++ code	Several PCs are connected efficiently for good results.	Variety of data is required.	Gennart and Hersch (1999)
Astronomical image processing	Hadoop	Performance is good for classifications.	Needs more different measuring parameters to be calculated.	Wiley *et al*. (2010)
Basic operations on image enhancement	MATLAB	Multicore computer is fully utilized	Other high level image processing algorithm need to be tested.	Kaur (2013)
Grayscale, brightening, darkening, thresholding and contrast change	CUDA as Programming tool Implemented on GPU.	Significant speed up with high resolution images	Not so good performance for images with low resolution	Olmedo *et al*. (2012)
Restoration, deconvolution, frequency domain	Multithreading using Java, ImageJ, Parallel Colt	Variety of data has been tested with good results.	Computational proficiency needed	Wendykier (2003).
Mosaicing of images for preclinical research.	ITK library, Implementation is done by using C and C++ languages, Cluster of 8 quad cores.	In this image processing tasks are wrapped into objects which are passed to the parallel engine. The engine is able to exploit data and task parallelism when executing the tasks on multicores, clusters and/or GPUs	High level image processing functions need to be implemented with variety of data sets.	Lemeire *et al*. (2009)
JPEG Image encoding and binary image processing, edge detection, hole filing	Massively parallel Processor arrays	Efficient Implementation of hole filling and edge detection.	Needs to be tested variety of images.	Osorio *et al*. (2009)
Nuclei detection on hematoxilin eosin (HE) stained colon tissue sample images	CUDA as Programming tool Implemented on GPU	Good speed UP with GPU over CPU	Other measuring parameters of parallel computing need to be calculated.	Reményi *et al*. (2011)

Table 1: Continue

Histogram computation, Removing clouds and calculation of DCT	CUDA as Programming tool Implemented on GPU	Direct comparison with sequential and parallel versions Significant speed up	Variety of images needs to be tested.	Yang *et al*. (2008)
Image Processing applications: Average filtering	Parallel Image Processing Toolkit	Significant speed up for large images. It is portable to different programming environments. Parallelization up to greatest extent.	It needs cluster of workstations to implement this research.	Squyres *et al*. (1996)
Point operators, Arithmetic and logic operations, Local neighborhood operators, Global operators	C using MPI-Panda library	Data and Task Parallelization has been done. Code can be easily used by parallel libraries.	High level image processing algorithm need to be tested	Nicolescu and Jonker (2002)
Colour to Black & White, Edge Detection, Convolution Masks	Handel-C(C Like Language) and DK	Make efficient use of parallellism	Diverse applications of image processing needs to be implemented.	Bouganis (2014)
Image Processing Toolbox (PIP)	Message Passing Modal is designed using MPI standard, Cluster of Workstation	Several Tested Parameters are having significant result.	This research needs Cluster of Workstations to implement.	Squyresy *et al*. (1998)
SIFT Keypoint Matching, Histogram Matching and Arithmatic Intensity Analysis	CUDA, GPU and Open CL	Varity of Data is tested with significant result.	High level image processing algorithm need to be tested.	Connors (2013)
Calculation of Mutual Information	Using Parallel Computing toolbox of MATLAB	Significant results using Multi Core computer	Registration needs to be done.	Saxena *et al*. (2014a).
Deblurring, Matrix factorization, and tomography	SIMD Parallel Processors, GPU	Good results obtained.	Variety of data needs to be tested.	Brand and Chen (2011)
Motion, Edge, Line Detection, Optical flow based tracking.	Open CV Library	Performance is significant.	Variety of data needs to betested.	Gregori (2012)
Binarization, Copy, Transpose, Blur, sobel, erosion, dilation, gradient, sum, max/min, histogram, mean	CUDA with GPU	Software as well as Hardware Efficiency Tested with significant result.	Needs more high level image processing to be done.	Nugteren *et al*. (2011)
Segmentation by Region Growing, Global Thresholding, Noise Reduction and Histogram Equalization	Parallel Computing toolbox in MATLAB with Multi Core Computer	Speed UP is significant	Higher level image processing algorithms need to be tested.	Saxena *et al*. (2013b)
Segmentation of Abdominal Image	Parallel Computing toolbox in MATLAB with Multi Core Computer	Speed UP is good and Intelligent Method to utilize cores	Other parameters need to be calculated	Saxena *et al*. (2013c)
Cellular Image Segmentation and Calculation of Statistical Features.	Parallel Computing toolbox in MATLAB with Multi Core Computer	Region wise utilization of cores.	Different variety of images are required to Test.	Saxena *et al*. (2013a)
Sobel Edge Detection and Homomorphic Filtering	CUDA implemented on GPU	Remarkable Speed UP	More complex algorithm of image processing is not evaluated	Zhang *et al*. (2010)
Segmentation using Region Growing	CUDA enabled GPU	Significant Speed UP	Variety of images need to be tested with different size.	Happ *et al*. (2012)

consumption in CPUs as well as in GPUs. Kaur (2013) have been already done this study. In the following Table 2 we are just adding study of some more algorithms, which will be very helpful for the researchers to study different image processing algorithms in terms of speed up.

Table 2: Analysis of the Comparison of Execution Time of different image processing algorithms in CPU and GPU

Image processing algorithms	Reference	Platform used	Size	CPU (Executing time) in ms	GPU (Executing time)
Brightening image transformation	Olmedo *et al*. (2012)	AMD Phenom II Quad-core to 3.2 GHz, 12 GB of RAM, Operating System: 64-bit Linux Fedora 14 GPU: GeForce 430 GT video card with 96 cores and 1 GB of RAM DDR3 is used	256×256	8.847117	0.19903
			512×512	36.12278	0.761405
			1024×1024	142.6773	2.995606
			1800×1400	342.4271	7.20223
			4000×3000	1610.854	33.97197
Darkening image transformation	Olmedo *et al*. (2012)	Same as above	256×256	9.5512192	0.2016992
			512×512	38.731661	0.7718848
			1024×1024	151.256079	3.0336608
			1800×1400	336.17552	7.2744512
			4000×3000	1719.52881	34.4515743
Inverse sinusoidal contrast transformation	Olmedo *et al*. (2012)	Same as above	256×256	11.26008	0.203088
			512×512	46.3111264	0.776816
			1024×1024	184.731943	3.057152
			1800×1400	448.714719	7.3071904
			4000×3000	2040.16223	34.6397216
Hyperbolic tangent contrast transformation	Olmedo *et al*. (2012)	Same as above	256×256	5.681072	0.1872896
			512×512	22.7657601	0.7213312
			1024×1024	91.6525903	2.823536
			1800×1400	219.788937	6.8108832
			4000×3000	1054.94413	32.2680701
Sine contrast transformation execution times	Olmedo *et al*. (2012)	Same as above	256×256	12.5076096	0.2077856
			512×512	51.3341637	0.7901344
			1024×1024	205.06483	3.1164224
			1800×1400	500.958685	7.4421792
			4000×3000	2273.02695	35.2699835
Linear feature extraction	Park *et al*. (2011)	CPU: Q9450 with 42.56 GFLOPS GPU: NVIDIA G92 (GeForce 9800 GTX) with 128 Cores and 512 MB Video Memory, GTX 280 for Speed UP Test	512×512	109	54.77
			1024×768	422	166.63
			1280×1024	610	250.42
			1200×1800	1250	471.88
			2278×1712	2375	1018.98
JPEG2000 encoding (DWT	Park *et al*. (2011)	Same as above	512×512	31.85	7.84
			1024×768	150.60	20.72
			1280×1024	164.50	31.17
			1200×1800	264.93	51.52
			2278×1712	471.66	91.08
			3024×2089	754.95	142.45
JPEG2000 encoding (Tier-1)	Park *et al*. (2011)	Same as above	512×512	94	205
			1024×768	234	390
			1280×1024	328	484
			1200×1800	891	735
			2278×1712	1640	1468
			3024×2089	1500	2062
Cartoon style NPR	Park *et al*. (2011)	Same as above	512×512	4594	49.31
			1024×768	14594	149.66
			1280×1024	18688	243.35
			1200×1800	47688	406.53
			2278×1712	93891	741.42
Oily style NPR	Park *et al*. (2011)	Same as above	512×512	7172	87.77

Table 2: Continue

Image processing algorithms	Reference	Platform used	Size	CPU (Executing time) in ms	GPU (Executing time)
			1024×768	15609	226.22
			1280×1024	35313	334.83
			1200×1800	49047	589.78
			2278×1712	94406	1107.12
Multiview stereo matching	Park *et al*. (2011)	Same as above	Temple ring (47 Images)	18422	340
Corners and edge detection	Squyres *et al*. (1995b)	OS: Ubuntu 11.04 CPU: Dual Core 6600, 2.40 GHz, Mem: 2 GB GPU: GeForce GTX 280, 240 CUDA cores, Memory: 1GB GPU: Tesla C1060, 240 CUDA cores, Memory: 4GB	2048×2048	4006	1240
Content authentication	Lin *et al*. (2011).	CPU Intel Xeon 5520 (2.26GHz) RAM 12GB DDR3 (1333MHz) GPU Architecture Tesla C1060 OS Centos 5.3 (64 bit) V2.3 CUDA	1024×1024	28877.66	903.83
Binarize	Nugteren *et al*. (2011)	Intel Core-i7 930, GPU: Geforce GTX470 GPU with 448 CUDA cores	2048×2048	106	0.34
Copy	Same as above	Same as above	2048×2048	99	0.34
Transpose	Same as above	Same as above	2048×2048	88	0.50
Blur	Same as above	Same as above	2048×2048	208	0.74
Sobel	Same as above	Same as above	2048×2048	230	1.21
Erode	Same as above	Same as above	2048×2048	151	0.65
Dilate	Same as above	Same as above	2048×2048	445	1.67
Gradient	Same as above	Same as above	2048×2048	432	1.45
Sum	Same as above	Same as above	2048×2048	37	0.28
Max	Same as above	Same as above	2048×2048	41	0.57
Min	Same as above	Same as above	2048×2048	41	0.56
Histogram	Same as above	Same as above	2048×2048	213	0.47

PARALLEL ARCHITECTURE, TOOLS AND TECHNIQUES AVAILABLE FOR IMPLEMENTING PARALLEL IMAGE PROCESSING

In previous section we have seen that several authors have implemented different techniques or algorithms parallally using different architectures and tools like MATLAB, CUDA, Hadoop and Many more. Now we are going to give brief description of these techniques with their advantages and disadvantages (Hadoop Advantages and Disadvantages, 2015).

GPU (Graphical Processing Unit): It is a graphical processing unit. A CPU contains few cores while GPU contains thousands of cores. As it is shown in the following Fig. 3.

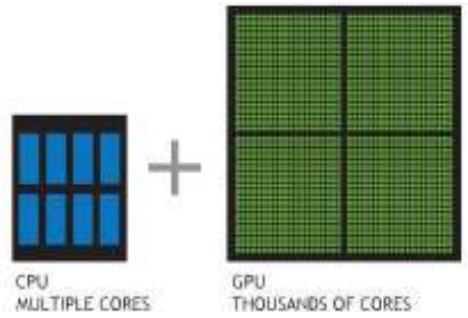

Fig. 3: Architecture of CPU and GPU [NVIDIA's Article]

It is also known as Visual Processing Unit (VPU). GPU has hundreds of cores while newest CPU's contain 4 or 8. At present a major challenge in image processing is that several applications of it need high computational power to attain high precision and real-time performance which is not easy to achieve by using CPU. Every NVIDIA GPU has 8 to 240 parallel cores, each core are having four units named floating point unit, logic unit (for add, sub, mul, madd), move and compare unit, branch unit. Cores in GPU are managed by Thread manager which can manage 12,000+ threads per core. GPU has been developed into a very bendable and controlling processor, which can be implemented by using high level languages. GPU supports 32-bit and 64-bit floating point IEEE-754 precision and offers lots of GFLOPS (Applications 2014).Srinivasan (2009) 8 series GPU deliver 25 to 200+ GFLOPS on compiled parallel C applications which are available in laptops, desktops and clusters. It is noticed that GPU parallelism is doubling every year. GPU provide high computational density (uses 100s of ALUs) and memory bandwidth (100+GB/s) (Nickolls, 2007). Where GPU executes kernel code and CPU executes serial code in the program. This reduces the execution time of the program. In this way while doing calculations by GPU, CPU time cycles can be used for other high priority tasks (Kaur and Nishi, 2010).

Advantages/Benefits: Benefits of using GPUs given in (Kaur and Nishi, 2010; Tariq, 2011) are following:

- It condensed power consumption
- GPUs are genuinely programmable and hold up high precision that is 32 bit floating point throughout the pipeline
- It provides portability, programmability, flexibility
- In GPU computing model CPU and GPU work together in a heterogeneous co- processing computing model

Limitations/Drawbacks: In Kaur and Nishi (2010) and Ruetsch and Oster (2008) there are some drawbacks are given as following:

- Gaining this speedup requires that algorithms are coded to reflect the GPU architecture and programming for the GPU differs significantly from traditional CPUs. In particular, incorporating GPU acceleration into pre-existing codes is more difficult than just moving from one CPU family to another; a GPU-savvy programmer will need to dive into the code and make significant changes to critical components.
- Incorporating GPU hardware into systems adds expense in terms of power consumption, heat production and cost. Some job mixes may be served more economically by systems that maximize the number of CPUs that can be brought to bear.

CUDA (Computed Unified Device Architecture): It is scalable parallel programming model and a software environment specifically used for parallel computing (Inam, 1994). CUDA is a parallel programming standard which is released in NVIDIA (2007). Generally, it is used to develop software that are used for graphics processors and is used to build up a diversity of general purpose applications for GPUs that are tremendously parallel and run on hundreds of GPU's processors or cores. It uses a language that is very analogous to C language and has a high learning curve. It has some extensions to that language to use the GPU-specific features that include new API calls and some new type qualifiers that apply to functions and variables. It has some definite functions, which is called as kernels. It can be a function or a full program invoked by the Central Processing Unit. It also provides common memory and synchronization among threads. It is supported only on NVIDIA's GPUs based on Tesla architecture. The graphics cards that support CUDA are GeForce 8-series, Quadro and Tesla (Kaur and Nishi, 2010; Inam, 2010). Heterogeneous architecture of CUDA is given in Inam (2010) and Saxena *et al*. (2014b). Working details of CUDA is given in Kaur and Nishi (2010).

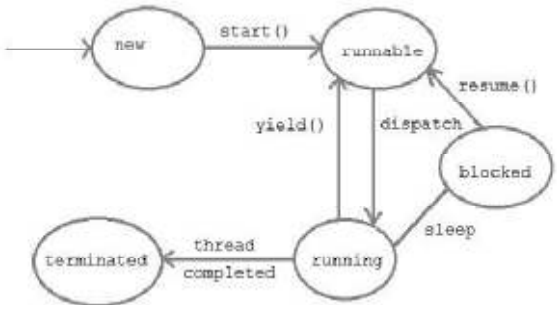

Fig. 4: Life cycle of thread

Advantages/Benefits (Kaur and Nishi, 2010):

- It is specifically designed to run for non graphic purposes.
- Its software development kit includes libraries, various debugging, profiling and compiling tools.
- In this programming task is simple and easy as kernel calls are written in C-like language.
- Provides faster downloads and read backs to and from the GPU.
- It exposes a fast shared memory region (up to 48 KB per Multi-Processor).

Limitations/Drawbacks (Kaur and Nishi, 2010):

- It is constrained to NVIDIA GPU's only.
- It runs its host code through a C++ compiler so it doesn't support the full C standard.
- Texture rendering is not supported in it.

Multithreading using Java: A thread is a dispatchable unit of work. Threads are light-weight processes within a process. A process is a collection of one or more threads and associated system resources. Java supports thread-based multitasking. Multithreading is the conceptual programming concepts when a program (process) is divided into two or more sub programs that can be implemented at the same time. A multithreaded program is having two or more parts that can run concurrently. Each part of such program is called thread. A thread is a dispatchable unit of work. Threads are light-weight processes within a process. A process is a collection of one or more threads and associated system resources. Java supports thread-based multitasking (Chapter Multithreaded Programming, 2015; Jain, 2015). Life cycle of thread is shown in the following Fig. 4.

In java we can construct single-thread as well as multi-thread application with it. A multi-threaded program in java has many entry and exit points, which are run concurrently with the main () method. Imageprocessing Applications can be implemented using single thread approach and multithreading approach by different contributors. In [Article] the

multithreading approach the shared memory in which the threads operate is the matrix of the image pixels. It can be used the Java packages to grab the pixel matrix of the image that has to be processed. Then different threads manipulate different parts of the matrix depending on the algorithm. The work task and the part of the matrix that each thread has to manipulate are determined by the main thread. The time that is necessary to manipulate the entire matrix either by a single thread or by all the threads is registered.

Advantages/Benefits: Benefits of using Multithreading are given below defined in Jain (2015):

- Threads share the same address space
- Generally Context-switching among threads is usually economical
- It is found that communication between threads is normally inexpensive
- It improved performance and concurrency

Limitations/Drawbacks (Java Tutorialsand Projects,2015):

- Mostly multithreaded programs are not easy to write. Only experienced programmers should undertake coding for these types of applications.
- It is much harder to replicate an error in a multithreaded or multicontexted application than it is to do so in a single-threaded, single-contexted application. As a result, it is more difficult, in the former case, to identify and verify root causes when errors occur.
- In this numerous threads can hinder with every other at the time of sharing hardware resources of hardware caches or Translation Look a side Buffers (TLBs).
- Execution times for a solo thread are not enhanced but can be degraded; still when only a single thread is executing. This is done because of slower frequencies and/or extra pipeline stages which are required to accommodate hardware containing thread switching.
- Support especially of hardware for multithreading is more observable to software, that requires more changes to application programs and OS (operating systems) than multiprocessing.

Hadoop: It is an open source software project that enables the parallel processing of huge data sets among clusters of commodity servers. Hadoop is specially designed to scale up from a single server to thousands/several of machines, with a very high degree of fault lenience. Rather than relying on high-end hardware, the resiliency of these clusters comes from the software's ability to observe and handle failures at the application layer (Hadoop Introduction, 2015).

Advantages/Benefits (Hadoop Introduction, 2015; big data concept):

- New nodes can be added as needed and added without needing to change data formats, how data is loaded, how jobs are written, or the applications on top.
- Hadoop brings extremely concurrent computing to commodity servers. The result is a generous decrease in the cost per terabyte of storage, which in turn makes it reasonable to model all your data.
- It is schema-less and can soak up any type of data, structured or not, from any numeral of sources. Data from numerous sources can be connected and aggregated in random ways enabling deeper analyses than any one structure can provide. When we lose a node, the system redirects effort to another position of the data and continues processing without missing a fright hit.

Limitations/Drawbacks (Hadoop Introduction, 2015):

- Hadoop Map-reduce and HDFS are rough in manner because the software under active development.
- Programming Model is very restrictive.
- Joins of multiple data sets are tricky and slow.
- Cluster management is hard.
- Still single master which requires care and may limit scaling.
- It is not fit for small data.
- It is having potential instability issues.

Parallel computing tool box in MATLAB (MATLAB Intro):MATLAB is extensively used for developing/prototyping algorithms. It is having several toolbox as image processing, signal processing, neural network toolbox and many more. Matlab 2010a onwards finally enables the "Parallel Computation Toolbox" for student use. By using this we can solve computationally and data-intensive tasks using multicore processors, GPUs and clusters of computer. We can parallelize applications of MATLAB without using CUDA or MPI programming. It contains High level constructs for example parallel for-loops, particular array types and parallelized arithmetical algorithms. The toolbox lets us use the full processing power of multicore desktops by executing applications on workers (MATLAB computational engines) that run locally. Without changing the code, we can execute the same applications on a computer cluster or a grid computing service (MATLAB Distributed Computing Server™). We can run parallel applications interactively or in batch. Built-in Parallel Computing

Support in MathWorks Products MATLAB Distributed Computing Server for Amazon EC2-Early Adopter Program (Mathworks, Parallel Computing Toolbox, 2015).

Advantages/Benefits:

- It contains Parallel for-loops (parfor) that is used for running task/job parallel algorithms on numerous cores/processors.
- It provides support for CUDA and enabled NVIDIA GPUs
- It uses fully utilization of multi core processors on the desktop using workers that run locally
- Computer cluster and grid support is provided by MATLAB Distributed Computing Server.
- Interactive and batch execution of parallel tasks/applications/jobs can be done by MATLAB.
- Distributed arrays and (SPMD) single program multiple data build for big dataset handling and data parallel algorithms.

Drawbacks/Limitations (article of MATLAB):

- Due to high level nature of MATLAB, it uses a lot of system resources.
- MATLAB is built on Java and Java is built upon C. So when we run a MATLAB program, our computer is busy trying to interpret all that MATLAB code. That consumes extra time.

OpenCL: Open Computing Language (OpenCL) is an open and royalty free parallel computing API designed to enable GPUs and other co processors (Article of Open CL) It is a standard for large scale parallel processing, it can help image processing but it is very low level and is designed for simplify the way to take advantage of many cpu cores and GPU stream processors.

Advantages/Benefits [Guide Open CL][Intro of Open CL]:

- Cross vendors software portability
- It Provides substantial acceleration in parallel programming.

Disadvantages/Limitations [Guide Open CL][Intro of Open CL]:

- It is not trouble-free to be trained.

OpenCV: OpenCV is an Image Processing library created by Intel and maintained by Willow Garage(Smith,2014). OpenCV is a library for computer vision, includes a lot of generic image processing routines and high level functions to support face recognition etc. It is available for C, C++ and Python. Several algorithms of image processing can be easily implemented by using this.

Advantages/Benefits:

- Easy to use and Install
- Open Source and Free

Disadvantages/Limitations:

- It is not easy to become skilled in Open CV (2014)

Parallel image processing in medical imaging: Kadah *et al*. (2011), Xu and Thulasiraman(2011), Schweiger (2011), Kim *et al*. (2010), Eklund *et al*. (2011a), Tward*et al*. (2011), D'Amore *et al*. (2011),Thiyagalingam*et al*. (2011)and Kadah *et al*. (2011).

Parallel Image Processing plays a very vital role in Medical Imaging. It is a rapidly growing interest in parallel computation application in various medical imaging applications. This inclination is estimated to carry on as more sophisticated and challenging medical imaging and high-order data visualization problems. Till now there have been done several research of parallel image processing in different medical image modalities like MRI, CT, PET, X-Ray, Ultrasound and Optical tomography as processing of these images requires numerous image processing algorithms like diffeomorphic mapping, image denoising, image reconstruction, motion estimation, deformable registration and modeling. Kadah *et al*. (2011) summarizes various parallel implementation of image processing techniques like an accelerated algorithm for brain fiber tracking, a new 3D deformable registration algorithm for mapping brain datasets, low computational efficiency of the conventional active shape model (ACM) algorithm and exploitation of the potential acceleration achieved when ACM is implemented on a parallel computation architecture, investigation of the potential of parallel computation in accelerating the image algebraic reconstruction techniques, a GPU-accelerated finite element solver for the computation of light transport in scattering media, investigation of the different throughput-oriented architectures can benefit Compressed Sensing (CS) MRI reconstruction algorithm and what levels of acceleration are feasible on different modern platforms, implementation of a four-dimensional denoising algorithm on a GPU, an accelerated automated process for creating complete patient specific pediatric dosimetry phantoms from a tiny set of segmented organs in a child's CT scan, solution of nonlinear Partial Differential Equations (PDEs) of diffusion/advection type, fundamental most problems in

image analysis, mapping of an enhanced motion estimation algorithm to novel GPU architectures, Eklund *et al*. (2011b) it is shown that how the computational power of cost-efficient GPUs can be used to speed up random permutation tests, Parallel computing in Radiotherapy planning. In spite of these several studies based on parallel medical image registration has been done till now.

CONCLUSION

As we have discussed above that parallel computing is having very important significance in several image processing techniques like edge detection, histogram equalization, noise removal, image registration, image segmentation, feature extraction, different optimization techniques and many more. In the field of medical imaging it also play a significant role. In current years a broad variety of approaches have been proposed for parallel image processing having their benefits and limitations. The present review provides the brief introduction of image processing techniques, different tools and techniques of computing parallel image processing with their respective features and limitations as mentioned in Table 1. Parallel Architectures, tools and techniques of parallel image processing is also discussed in this review with their advantages and limitations as in Table 2. It can be used in different applications of image processing on the basis of its appropriateness, performance, computational cost on the basis of time, applicability. As discussed parallel implementation of Image processing find to be great area of interests by different researchers because of its performance, suitability and availability. We saw that some techniques find to be limited applications and needs more computational knowledge. However, their performance can be improved by implementing them intelligently for example integration of the concepts of java's multithreading with MATLAB can give the significant results. Finally, we have also discussed applications of parallel image processing in medical imaging in this review and highly preferred to employ parallel image processing in various techniques of medical imaging for fast and efficient results for treatment planning.

REFERENCES

Aashburner, J. and K. Friston, 2005. Unified segmentation. Neuroimage, 26(3): 839-851.

Ahmed, M.F., 2014. Parallel implementation of K-means on multi-core processors. Comput. Sci. Telecommun., 41(1): 52-60.

Akgün, D., 2013. Performance evaluations for parallel image filter on multi-core computer using java threads. Int. J. Comput. Appl., 74(11): 13-19.

Aoki, S. and T. Nagao, 1999. Automatic construction of tree-structural image transformations using genetic programming. Proceeding of the Conference on Image Analysis and Processing.Venice, pp: 136-141.

Barney, B., 2014. Introduction to Parallel Computing. Retrieved from: https://computing.llnl.gov/tutorials/parallel_comp/.

Basavaprasad, B. and M. Ravi, 2014. Study on the importance of image processing and its applications. Int. J. Res. Eng. Technol., 3: 155-160.

Bister, M., C.S. Yap, K.H. Ng and C.H. Tok, 2007. Increasing the speed of medical image processing in MATLAB. Biomed. Imaging Interv. J., 3(1): e9.

Bouganis, C., 2014. Parallel Image Processing: An Introductory Lecture to the Project. Retrieved from: https://www.google.com.pk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CBsQFjAAahUKEwiimvqMybnHAhVBuhoKHWBQBBs&url=http%3A%2F%2Fcas.ee.ic.ac.uk%2Fpeople%2Fccb98%2Fteaching%2FHandelC%2FProjDocPresentation.pdf&ei=ScjWVeLRCsH0auCgkdgB&usg=AFQ.

Brand, M. and D. Chen, 2011. Parallel quadratic programming for image processing. Proceeding of the 18th IEEE International Conference on Image Processing (ICIP). Brussels, pp: 2261-2264.

Bräunl, T., 2001. Tutorial in data parallel image processing. Aust. J. Intell. Inform. Process. Syst., 6(3): 164-174.

Chapter Multithreaded Programming, 2015. Retrieved from: http://www.buyya.com/java/Chapter14.pdf.

Connors, D., 2013. Exploring Computer Vision and Image Processing Algorithms in Teaching Parallel Programming. Retrieved from: https://www.google.com.pk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CCAQFjAAahUKEwj07_Tg0LnHAhVC1BoKHRuwCcU&url=http%3A%2F%2Fgrid.cs.gsu.edu%2F~tcpp%2Fcurriculum%2Fsites%2Fdefault%2Ffiles%2FTeaching%2520Parallel%2520Programming%2520Usin.

D'Amore, L., D. Casaburi, L. Marcellino and A. Murli, 2011. Numerical solution of diffusion models in biomedical imaging on multicore processors. Int. J. Biomed. Imaging, DOI: 10.1155/2011/680765.

Dougherty, G., 2009. Digital Image Processing for Medical Applications. Cambridge University Press, Cambridge, pp: 462, ISBN: 1139476297.

Drakos, N., 2014. Computer Based Learning Unit. University of Leeds and Ross Moore, Mathematics Department, Macquarie University, Sydney. Retrieved from: http://fourier.eng.hmc.edu/e161/lectures/introduction/index.html.

Edelman, A., P. Husbands and S. Leibman, 2006. Interactive supercomputing's star-P platform: Parallel MATLAB and MPI homework classroom study on high level language productivity. HPEC. Retrieved from: http://www.ijcaonline.org/archives/volume113/number3/19807-1598.

Eklund, A., M. Andersson and H. Knutsson, 2011a. True 4D image denoising on the GPU. Int. J. Biomed. Imaging, DOI: org/10.1155/2011/952819.

Eklund, A.,M. Andersson and H. Knutsson, 2011b. Fast random permutation tests enable objective evaluation of methods for single-subject fMRI analysis. Int. J. Biomed. Imaging, DOI: org/10.1155/2011/627947.

Fangbin, L., J.S. Frank and A. Plaza, 2011. Parallel hyperspectral image processing on distributed multicluster systems. J. Appl. Remote Sens., 5: 1-14.

Fatemi, H., H. Corporaal, T. Basten, P. Jonker and R. Kleihorst, 2004. Implementing Face Recognition Using a Parallel Image Processing Environment Based on Algorithmic Skeletons. Retrieved from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.10.7759

Fernandez, B.A. and S. Kumar, 2009. Distributed Image Processing Using Hadoop Map Reduce Frame Work, Retrieved from: http://search.iiit.ac.in/cloud/presentations/26.pdf.

Fung, J. and S. Mann, 2008. Using graphics devices in reverse: GPU-based image processing and computer vision. Proceeding of the IEEE International Conference on Multimedia and Expo. Hannover, pp: 9-12.

Gennart, B. and R.D. Hersch, 1999. Computer-aided synthesis of parallel image processing applications. Proceeding of the SPIE International Symposium on Optical Science, Engineering and Instrumentation. Denver, Colorado, 3817: 48-61.

Georgantzoglou, A., S. Joakim da and J. Rajesh, 2014. Image Processing with MATLAB and GPU. DOI: org/10.5772/58300

Gregori, E., 2012. Introduction to Computer Vision using Open CV. Proceeding of the Embedded Systems Conference in San JoseBerkeley Design Technology, Inc.Oakland, California USA.

Hadoop Advantages and Disadvantages, 2015. Retrieved from: http://www.j2eebrain.com/java-J2ee-hadoop-advantages-and-disadvantages.html.

Hadoop Introduction, 2015.IBM. Retrieved from: http://www-01.ibm.com/software/data/infosphere/hadoop/.

Happ, P.N., R.Q. Feitosa, C. Bentes and R. Farias, 2012. A parallel image segmentation algorithm on gpus. Proceeding of the 4th GEOBIA. Rio de Janeiro-Brazil, pp: 580.

Huang, T.Y., Y.W. Tang and S.Y. Ju, 2011. Accelerating image registration of MRI by GPU-based parallel computation. Magn. Reson. Imaging, 29(5): 712-716.

Inam, R., 1994. An Introduction to GPGPU Programming-CUDA Architecture. Retrieved from: http://www.es.mdh.se/pdf_publications/1994.pdf.

Inam, R., 2010. Algorithm for multi-core graphics processors. M.A. Thesis, Chalmers University of Technology, Göteborg.

Iqbal, M. and S. Raghuwanshi, 2014. Analysis of digital image processing with parallel with overlap segment technique. Int. J. Comput. Sci. Netw. Secur., 14(6): 52.

Jain, A., 2015. Java Notes-multithreading. Retrieved from:http://www.niecdelhi.ac.in/uploads/Notes/btech/5sem/cse/Java%20Notes%201%20-%20Multithreading.pdf.

Java Tutorials and Projects, 2015. Retrieved from: http://javatutorialandprojects.blogspot.in/2012/09/advantages-and-disadvantages-of-threads.html%20om.

Ji-Hoon, K., A. Syung-Og, K. Shin-Jin, K. Seok-Hun and K. Soo-Kyun, 2014. Fast 3D graphics rendering technique with CUDA parallel processing. Int. J. Multimed. Ubiquit. Eng., 9(1): 199-208.

Kadah, Y.M., K.Z. Abd-Elmoniem and A.A. Farag, 2011. Parallel computation in medical imaging applications. Int. J. Biomed. Imaging, 2011: 2, Doi: org/10.1155/2011/840181.

Kamboj, P. and V. Rani, 2013. Brief study of various noise model and filtering techniques. J. Global Res. Comput. Sci., 4(4): 166-171.

Kaur, P., 2013. Implementation of image processing algorithms on the parallel platform using matlab. Int. J. Comput. Sci. Eng. Technol., 4(6): 696-706.

Kaur, P. and Nishi, 2010. A survey on CUDA. Int. J. Comput. Sci. Inform. Technol., 5: 2210-2214.

Kika, A. and S. Greca, 2013. Multithreading image processing in single-core and multi-core CPU using Java. Int. J. Adv. Comput. Sci. Appl., 4(9): 165-169.

Kim, D., D.T. Joshua, S. Mikhail, R.H. Clifton, M. Armando and D. Pradeep, 2010. High-Performance 3D compressive sensing MRI reconstruction. Proceeding of the 32nd Annual International Conference of the IEEE EMBS Buenos Aires, Argentina.

Klimeck, G., F. Yafuso, M. McAuley, R. Deen, G. Yagi, E.M. DeJong and A.C. Thomas, 2003. Near Real-time Parallel Image Processing using Cluster Computers.Space Mission Challenges for Information Technology.

Lemeire, J., Y. Zhao, P. Schelkens, S. De Backer, F. Cornelissen *et al*., 2009. Towards fully user transparent task and data parallel image processing. Proceeding of Workshop on Parallel and Distributed Computing in Image Processing, Video Processing and Multimedia.

Lin, C., L. Zhao and J. Yang, 2011. A high performance image authentication algorithm on GPU with CUDA. Int. J. Intell. Syst. Appl., 2: 52-59.

Low, J., 2013. Medical image processing on intel parallel frameworks. M.Sc. Thesis, High Performance Computing.

Mahmoudi, A., S., P. Manneback, F. Lecron, M. Benjelloun and S. Mahmoudi, 2009. Computing - parallel image processing on GPU with Cuda and OpenGL. Complex HPC meating Lisbon, Retrieved from: http://docs.oracle.com/cd/E13203_01/tuxedo/tux71/html/pgthr5.htm.

Malakar, R. and N. Vydyanathan, 2013. A CUDA-enabled hadoop cluster for fast distributed image processing. Proceeding of the National Conference on Parallel Computing Tecnologies, pp: 1-5.

Manjunathachari, K. and K. SatyaPrasad, 2005. Modeling and simulation of parallel processing architecture for image processing. J. Theor. Appl. Inform. Technol., 3(1): 1-11.

Markonis, D., R. Schaer, I. Eggel, H. Müllerand A. Depeursinge, 2012. Using mapreduce for large-scale medical image analysis. Proceeding of the IEEE 2nd International Conference on Healthcare Informatics, Imaging and Systems Biology (HISB, 2012). San Diego, CA, pp: 1.

Marwa, C., B. Haythem, S. Fatma Ezahra and A. Mohamed, 2014. Image processing application on graphics processors. Int. J. Image Process., 8(3): 66-72.

Mathworks, Parallel Computing Toolbox, 2015. Retrieved from: http://www.mathworks.in/products/datasheets/pdf/parallel-computing-toolbox.pdf

Navarro,C.A., N. Hitschfeld-Kahler and L. Mateu, 2014. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. Commun. Comput. Phys., 15(2): 285-329.

Nickolls, J., 2007. GPU Parallel Computing Architecture and CUDA Programming Model. Hot chips 2007: NVIDIA GPU parallel computing architecture, NVIDIA Corporation

Nicolescu, C. and P. Jonker, 2002. A data and task parallel image processing environment. Parallel Comput., 2: 945-965.

Nugteren, C., H. Corporaal and B. Mesman, 2011. Skeleton-based automatic parallelization of image processing algorithms for GPUs. Proceeding of the International Conference on Embedded Computer Systems (SAMOS). Samos, pp: 25-32.

NVIDIA, 2007. Retrieved from: http://www.nvidia.com/object/what-is-gpu-computing.html.

Olmedo, E., J. Calleja, A. Benitez and M.A. Medina, 2012. Point to point processing of digital images using parallel computing. Int. J. Comput. Sci. Issues, 9(3): 1-10.

Osorio, R.R., C. Daz-Resco and J.D. Bruguera, 2009. Highly Parallel Image Processing on a Massively Parallel Processor Array. Retrieved from: www.ac.usc.es/system/files/Jornadas09.pdf.

Open CV, 2014. Retrieved from: http://www.aishack.in/2010/02/why-opencv/.

OpenCV and MATLAB, 2014. Retrieved from: http://opencv-users.1802565.n2.nabble.com/OpenCv-vs-Matlab-td2426918.html.

Pan, Z., 2013. High performance computing image analysis for radiotherapy planning. M.Sc. Thesis University of Edinburg.

Park, I.K., N. Singhal, M. Hee Lee, S.Choand C.W.Kim, 2011. Design and performance evaluation of image processing algorithms on GPUs.IEEE T. Parall. Distr., 22(1): 91-104.

Pedrino, E.C. and M.M. Fernandes, 2014. Automatic generation of custom parallel processors for morphological image processing. Proceeding of the IEEE 26th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD, 2014). Jussieu, pp: 176-181.

Rajaraman, V. and C. Siva Ram Murthy, 2006. Parallel Computers-architecture and Programming. Prentice-Hall of India, New Delhi.

Reményi, A., S. Szénási, I. Bándi, Z. Vámossy, G. Valcz *et al*., 2011. Parallel biomedical image processing with GPGPUs in cancer research. Proceeding of the 3rd IEEE International Symposium on Logistics and Industrial Informatics (LINDI). Budapest, pp: 245-248.

Roy, F., 2013. Compiling an Image Processing GUI and Accelerating it Using a GPU.

Ruetsch, G. and B. Oster, 2008. Getting Started with CUDA. nVision 08: The World of Visual Computing, NVIDIA Corporation.

Saxena, S., N. Sharma and S. Sharma, 2013a. Image processing tasks using parallel computing in multi core architecture and its applications in medical imaging. Int. J. Adv. Res. Comput. Commun. Eng., 2(4): 1896-1900.

Saxena, S., N. Sharma and S. Sharma, 2013b. An intelligent system for segmenting an abdominal image in multicore architecture. Proceeding of the 10th International Conference and Expo on Emerging Technologies for a Smarter World (CEWIT). Melville, NY, pp: 1-6.

Saxena, S., N. Sharma and S. Sharma, 2013c. Region wise processing of an image using multithreading in multicore environment & Its application in medical imaging. Int. J. Comput. Eng. Technol., 4(4): 20-30.

Saxena, S., S. Sharma and N. Sharma, 2014a. Parallel computation of mutual information and its applications in medical image registration. Proceeding of the IEEE Xplore Database, International Conference MEDCOM held at Noida, India.

Saxena, S., S. Sharma and N. Sharma, 2014b. Image registration using parallel computing in multicore environment and its applications in medical imaging: An overview. Proceeding of the International Conference on Computer and Communication Technology (ICCCT, 2014). Allahabad, pp: 97-104.

Schweiger, M., 2011. GPU-accelerated finite element method for modelling light transport in diffuse optical tomography. Int. J. Biomed. Imaging, 2011: 11, Doi: org/10.1155/2011/403892.

Slabaugh, G., R. Boyes and X. Yang, 2010. Multicore image processing with openMP. IEEE Signal Proc. Mag., 27(2): 1-9.

Smith, M., 2014. Introduction to OpenCV. MATLAB and OpenCV. Retrieved from: http://blog.fixational.Com/post/19177752599/opencv-vs-MATLAB.

Soviany, C., 2003. Embedding data and task parallelism in image processing applications. Ph.D. Thesis, Technische Universiteit Delft.

Squyres, J.M., A. Lumsdaine and R.L. Stevenson, 1995a. A cluster-based parallel image processing toolkit. Proceeding of the IS&T Conference on Image and Video Processing.San Jose, CA, pp: 5-10.

Squyres, J.M., A. Lumsdaine and R.L. Stevenson, 1995b. A cluster-based parallel image processing toolkit. Proceeding of the Society of Photo-Optical Instrumentation Engineers, pp: 228-239.

Squyres, J.M., B.C. McCandless, A. Lumsdaine and R.L. Stevenson, 1996. Parallel and Distributed Algorithms for High-Speed Image Processing. Proceeding of 6th Annual Dual-Use Technologies and Applications Conference.

Squyresy, J.M., A. Lumsdainey and R.L. Stevenson, 1998. A toolkit for parallel image processing. Proceeding of the SPIE Conference on Parallel and Distributed Methods for Image Processing, pp: 69-80.

Srinivasan, B.V., 2009. Graphical Processor and Cuda. Slides Adapted from CMSC828E Spring Lectures.

Tariq, S., 2011. An Introduction to GPU Computing and CUDA Architecture. NVIDIA Corporation. Retrieved from: http://on-demand,gputechconf.com/gtc-express/2011/presentations/GTCExprcssSarahTariq_June2011.pdf. (Accessed on: October, 2014)

Thiyagalingam, J., D. Goodman, J.A. Schnabel, A. Trefethen and V. Grau, 2011. On the usage of GPUs for efficient motion estimation in medical image sequences. Int. J. Biomed. Imaging, DOI: org/10.1155/2011/137604.

Tward,D.J., C. Ceritoglu, A. Kolasny, G. Sturgeon, W.P. Segars, M.I. Miller and J.T. Ratnanather, 2011. Patient specific dosimetry phantoms using multichannel LDDMM of the whole body. Int. J. Biomed. Imaging, DOI: org/10.1155/2011/481064.

Wendykier, P., 2003. High performance java software for image processing. Ph.D. Thesis, Adam Mickiewicz University.

Wiley, K., A. Connolly, S. Krugho, G. Je, M. Balazinska *et al*., 2010. ASP Conference Series.

Xu, M. and P. Thulasiraman, 2011. Mapping iterative medical imaging algorithm on cell accelerator. Int. J. Biomed. Imaging, 2011: 11, Doi: org/10.1155/2011/843924.

Yamamoto, M. and K. Kaneko, 2012. Parallel image database processing with mapreduce and performance evaluation in pseudo distributed mode. Int. J. Electron. Comm. Stud., 3(2): 211-228.

Yang, Z., Y. Zhu and Y. Pu, 2008. Parallel image processing based on CUDA. Proceeding of the International Conference on Computer Science and Software Engineering.

Zhang, N., Y.S. Chen and W. Jian-Li, 2010. Image parallel processing based on GPU. Proceeding of the 2nd International Conference on Advanced Computer Control (ICACC). Shenyang, pp: 367-370.

Zhou, L., H. Wang and W. Wang, 2012. Parallel implementation of classification algorithms based on cloud computing environment. Indonesian J. Electr. Eng., 10(5): 1087-1092.