International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 17 (2015) pp 37651-37657

A Significant Approach for Implementing Parallel Image Processing using MATLAB with Java Threads and Its Implementation in Segmentation using Otsu's Method in Multi Core Environment

Sanjay Saxena
PhD Scholar, School of Biomedical Engineering Indian Institute of Technology (BHU) Varanasi
ssaxena.rs.bme11@itbhu.ac.in

Dr. Shiru Sharma
Assistant Professor, School of Biomedical Engineering Indian Institute of Technology (BHU) Varanasi
shiru.bme@itbhu.ac.in

Dr.Neeraj Sharma
Associate Professor, School of Biomedical Engineering Indian Institute of Technology (BHU) Varanasi
neeraj.bme@itbhu.ac.in

Abstract
This paper present a considerable approach for implementing parallel image processing using MATLAB with Java. Parallel Computing has become an essential pillar for implementing numerous applications of image processing. As its computation is repeatedly done on matrix of pixels so it needs several resources and processing execution time. For efficient implementation of parallel image processing there have been developed several tools and techniques till now such as CUDA (Computed Unified Devise Architecture), GPU (Graphics Processing Unit) , MATLAB, Hadoop, OpenCV, OpenCL, Java and many more. Main purpose of this paper is to show application of merging of MATLAB with java for implementing image processing techniques parallely. In this paper implementation of Otsu's segmentation of the several images including Dicom images using parallel computing toolbox of MATLAB with the concept multithreading in java has been done and got significant results in terms of speedup which can be the good alternative and efficient approach for implementing parallel image processing. Main purpose of implementing MATLAB with java as MATLAB plays a very significant role in image processing techniques since it is having a toolbox containing variety of image processing functions which expand the ability of its numeric computing environment. It also consists of parallel computing toolbox for implementing explicit multithreading and supportable environment of GPU and CUDA. However, Programming Environment of MATLAB relies on Java for several jobs or tasks. Java Virtual Machine (JVM) permits an application to have several threads which runs concurrently.

Keywords: Image Processing, Parallel Computing, Java, MATLAB, Multithreading, Parallel Image Processing.

I. Introduction

Now days, High performance computing is the need of several image processing problems for fast and efficient results [1-13]. There are several tools; techniques and libraries are available for solving the same problem like GPU, CUDA, MATLAB, OpenCL, OpenCV, Java, Hadoop [14]. MATLAB plays a very vital role for developing algorithms of image processing as it consists of Image Processing Toolbox containing numerous image processing functions. It also consist the system for implementing parallel computing explicitly which is called as Parallel Computing Toolbox (PCT) and Implicit multithreading which is based on built-in multithreading in functions of MATLAB. PCT of MATLAB generally uses heavyweight processes instead of using light weight threads. Multithreading using java can be the significant solution for implementing parallel computing using light weight threads [15]. Threads are used for implementing parallel computing that exploit the capabilities of easily available multi-core technology. It also provides a method to perform jobs which are autonomous from each other. Basically this paper presents the role of implementing parallel image processing using concepts of parallelization of MATLAB with the multithreading concepts of Java by implementing Otsu's method for image segmentation.

Parallel image processing provides an efficient way to solve compute problem of image processing that requires large time to execute or processing of the huge amount of information, in reasonable time [16]. It founds to be only the effective solution to gain the required processing rate for handling high-speed image processing applications. Till now, there have been developed several researches based on image processing based on parallel computing. They presented the requirement of parallel computing in image processing, its implementation techniques and results [1-25]. In [14] we effectively explained the significance of parallel image processing with several benefits and limitations. That paper also describes the different tools and techniques of implementation of parallel image processing.

This paper is prepared as follows: In the next section we present a brief overview of the concepts of Multithreading in Java. Third part of this paper is having concise introduction of the parallel computing in MATLAB with its applications in image processing. Then we present a significant technique

for implementing parallel image processing using java threads with MATLAB in the next section. Section V discuss about Otsu's method for Thresholding for image segmentation. Approach of this method i.e. implementation part is discussed in section VI. Experimental set up and results are given in section VII. As a final point, we discussed and conclude the paper in section VIII.

II. Multithreading in Java with its Applications in Image Processing

Java is the object oriented, platform independent and multithreaded programming environment. A multithreaded program is that program which contains several sections that can run parallelly and each section can handle diverse task at the same time making most favorable use of the accessible resources especially when our computer has numerous CPUs. Multithreading enables us to write a program in a way where several actions can proceed concurrently in the identical program [16]. A thread is a smallest unit of work. These 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. Life cycles of threads are given in the following figure given in [16].

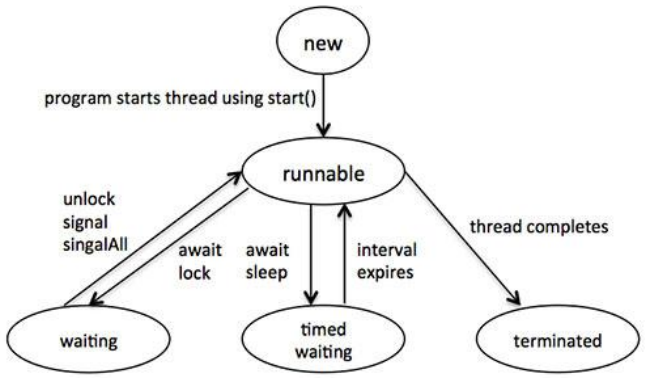


Figure 1: Life Cycle of Thread [16]

Above mentioned stages are briefly described as below:
New: In this stage a new thread starts its living cycle in the new state. It remains in the same state till the program starts the thread. We can also refer it as a born thread.
Runnable: After the first stage i.e. when newly born thread is in progress, the thread becomes runnable. A thread in this state is considered to be executing its job.
Waiting: Occasionally, a thread transitions to the waiting situation while the thread waits for an additional thread to perform a task. A thread transitions back to the runnable state only when one more thread signals the waiting thread to persist executing.
Timed waiting: A runnable thread can go through the timed waiting state for a particular interval of time. A thread in this situation transition reverses to the runnable state while that time interval expires or once the event it is waiting for occurs.
Terminated: A runnable thread enters the terminated state while it executes its job or otherwise terminates.

We can also set the priority of java threads as per requirement which helps operating system to decide the order in which that will execute.
Threads can be created by two ways by implementing runnable interface or by extending thread class [16]. Main advantage of implementing multithreading using java is that Threads share the similar address space. It doesn't block the user because threads are independent and we can perform multiple operations at same time. Another important thing is that communication between threads is normally inexpensive. Threads are independent so it doesn't affect other threads if exception occurs in a single thread [15].
There are several studies based on parallel image processing using java threads have been implemented like Devrim Akgün proposed Parallel Image Filter on Multi - Core Computer using Java Threads and got significant results[25], Alda Kika et al. describes applications of multithreaded image processing in solitary core and multi core CPU via Java [26] and many more.

III. Parallel Computing in MATLAB with its Applications in Image Processing

MATLAB provides a very important toolbox i.e. Parallel Computing Toolbox (PCT) which provides us to resolve computationally and data concentrated tasks/jobs/problems using multicore processors, GPUs, and computer clusters [27]. It also provides us high level programming constructs like parallel for-loops, special array types, and parallelized numerical algorithms by which we can parallelize several MATLAB applications without CUDA or MPI programming.
Key Features of PCT of MATLAB are given below [27-28]

1. It provides us parallel for loops i.e. parfor to execute parallel algorithms on several processors.
2. It support for CUDA enabled NVIDIA GPUs.
3. It provides complete utilization of multicore processors on the desktop through workers that execute locally.
4. Grid support and Computer cluster (with MATLAB Distributed Computing Server) are supported by PCT.
5. We can develop interactive and batch execution of parallel applications.
6. Distributed arrays and single program multiple data (spmd) can use for huge dataset handling and data parallel algorithms.

We can efficiently use PCT to execute applications on a multicore computer with local workers available in the toolbox as we have already discussed above, which obtain benefits of GPUs and can be scale up to a cluster (with MATLAB Distributed Computing Server) [27-28]. Basic structure of PCT is given below [27-28]

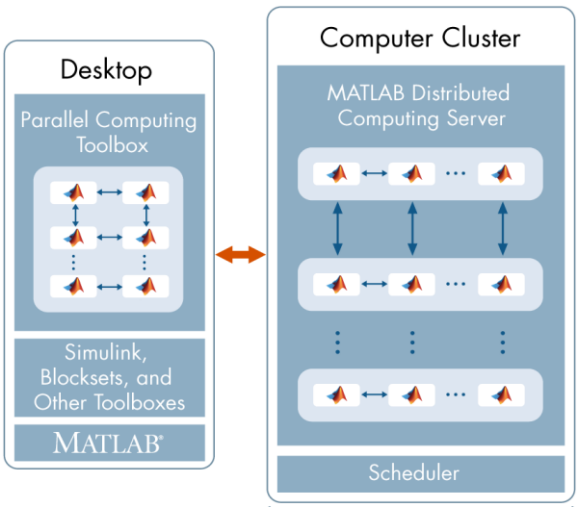


Figure 2: Basic Structure of PCT

In our previous research works we have implemented several image processing application using PCT of MATLAB and got tremendous results[4][12][13][14][29-33] like Complex noise reduction, region wise processing of an image, segmentation of abdominal image and many more. It provides efficient ways to solve big problem in easiest way.

IV. A New Approach: MATLAB with Java For Implementing Parallel Image Processing Applications

In 2014 Yair altman has given his tremendous contribution in "Undocumented secret of MATLAB java programming" and in "Accelerating MATLAB performance". He described that MATLAB does not give the users straight access to threads exclusive of the PCT (Parallel Computing Toolbox). He explained it by giving some expensive computations or I/O to be run in the background without freezing the main application. Instead, in MATLAB there is either implicit multiprocessing which relies on built-in threading support in some MATLAB functions, or explicit multiprocessing using PCT (note: PCT workers use heavyweight processes, not lightweight threads). So the solitary and important way to achieve really multi-threading in MATLAB is through Java, MEX, or .Net, or by spawning peripheral standalone processes. Note that we do not hoard any CPU cycles by running jobs/tasks in parallel. In general balance, we really enlarge the amount of CPU processing, because of the multi-threading overhead. However, in the enormous majority of cases we are extra concerned in the responsivity of MATLAB's chief processing thread (it is known as the Main Thread, MATLAB Thread, or simply MT) than in dropping the computer's whole energy utilization. In that cases, offloading effort to asynchronous Java, C++ or .Net threads could eradicate bottlenecks from MATLAB's main thread, by achieving considerable speedup. Java (.Net/C++) threads are extremely valuable when they can run exclusively autonomously from MATLAB's main thread. MATLAB uses Java for plentiful tasks, including data-processing algorithms, networking, and graphical user-interface (GUI). Actuality, under the cover, even MATLAB timers utilizes Java threads for their internal triggering system. To employ Java, MATLAB launches its individual devoted JVM (Java Virtual Machine) once it starts (except it's started through the *-nojvm* startup option). Once in progress, Java can be straightforwardly used within MATLAB as a accepted extension of the MATLAB. So there are several potential benefits of Java multithreading for MATLAB users. For this implementation researchers are assumed to be familiar with how to program Java code and one thing is more how to compile Java classes.

So what we see that if we use efficiently and fully utilize all the resources of MATLAB with java than we can get significant result. In this paper we have tested this technique for implementing segmentation based using Thresholding with the variety of Images.

V. Image Segmentation based on Otsu's Method

Image Segmentation is one of the most important and fundamental tasks for efficient image analysis and understanding [34-38]. There are several techniques are available for image segmentation such as different types of thresholding, region growing, region split and merging, edge based segmentation, feature based segmentation and many more[39-40]. Though, there are so many researches are still going on. One of the most important method of image segmentation is Otsu's Thresholding method [41-42].

Basically it works directly on the gray level of histogram and it selects the threshold to minimize the intraclass variance of the thresholded black and white pixels.

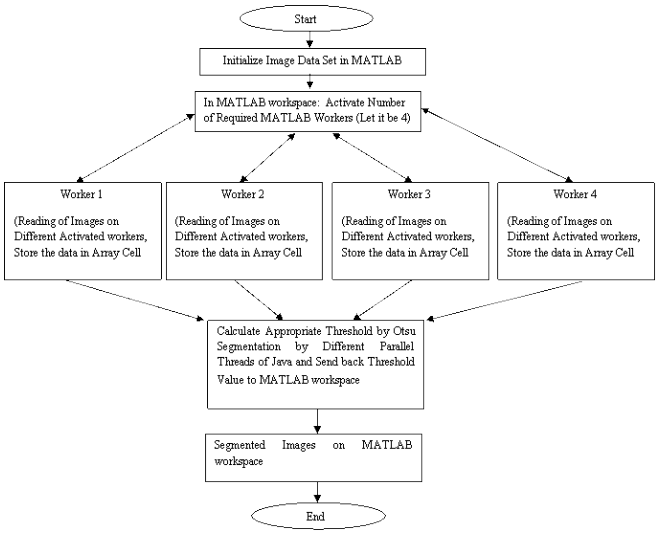
There are some assumptions of Otsu Thresholding method given as below

- It considers Histogram (and the image) are bimodal.
- There is no use of spatial coherence, nor any other notion of object structure.
- It assumes stationary statistics, but can be changed to be locally adaptive.
- It also assumes identical illumination, so the bimodal brightness performance arises from object appearance differences only.

It is based on a very simple idea: Determine the threshold that minimizes the weighted within-class variance which turns out to be the same as maximizing the between-class variance. Detailed description of Otsu's based image segmentation is given in [41].

VI. Implementation

To show the efficient working of MATLAB with java threads in image processing we have chosen Otsu segmentation in environment of MATLAB and java. Flow chart of the implemented method is shown in the following flow chart.



Flowchart of parallel image segmentation using MATLAB by creating threads in Java:

As we know that Java classes can be easily import into the MATLAB workspace [43] by adding the path statically or dynamically. Basic approach of our implementation is given below

1. Firstly created a java class with different threads for implementing Otsu's segmentation.
2. Initiate MATLAB work space.
3. After that initialized workers of MATLAB using PCT.
4. Equally divide set of images present in image datasets and load images on different workers for fast executions.
5. Then called existing multithreaded java class.
6. Thresholds calculated and send back to MATLAB workspace where the segmented images are obtained.

We measured the performance of the work in terms of speed up. In general speedup is the ratio of time consumed in parallel execution and time consumed in sequential execution [4][12][13][14].

VII. Experimental Setup And Results

We have performed total nine experiments with three image data set on three distinguished multicore environment given below and found significant speedup. First image data set consist 12 grayscale of size 256 X 256, Second image dataset consists 12 RGB images of size 512 X 512 and Third data set 12 Dicom images of size 512 X 512. .

Experimental Setup 1
Hardware Con`uration:
Processor: Intel Core i7 -3770 CPU @340 GHz
RAM (Random Access Memory): 4 GB
Hard Disk Drive: 320 GB

Software Environment:
System Type: 64 Bit operating System, x – 64 - based processor.
Development Tools: MATLAB, Intel Compiler 9.1, jdk (java development kit) version 1.6.0_21

Parallelization Scheme in MATLAB:
Multithreaded Parallelism and Explicit Parallelism

Images:
12 gray scale, 12 RGB and 12 Dicom images

Results: Speed up is the average of all images present in the data set
Obtained Speed up (For Gray Scale Images): 245%
Obtained Speed up (For Indexed Images): 223.5%
Obtained Speed up (For Dicom Images): 189%

Experimental Setup 2
Hardware Configuration:
Processor: Intel Core i3 – 2350M Processor 2.30 GHz
RAM (Random Access Memory) : 3 GB
Hard Disk Drive: 320 GB

Software Environment:
System Type: 32 Bit operating System.
Development Tools: MATLAB, Intel Compiler , jdk(java development kit) version 1.6.0_21

Parallelization Scheme in MATLAB:
Multithreaded Parallelism and Explicit Parallelism

Images:
12 gray scale, 12 RGB and 12 Dicom images

Results: Speed up is the average of all images present in the data set
Obtained Speed up (For Gray Scale Images): 225.5%
Obtained Speed up (For Indexed Images): 197%
Obtained Speed up (For Dicom Images): 191.06%

Experimental Setup 3
Hardware Configuration:
Processor: Intel Core 2 duo T 6400 CPU @2.00 GHz
RAM (Random Access Memory) : 2 GB
Hard Disk Drive: 250 GB

Software Environment:
System Type: 32 Bit operating System, x 64 based processor.
Development Tools: MATLAB, Intel Compiler , jdk (java development kit) version 1.6.0_21

Parallelization Scheme in MATLAB:
Multithreaded Parallelism and Explicit Parallelism

Images:
12 gray scale, 12 RGB and 12 Dicom images

Results: Speed up is the average of all images present in the data set
Obtained Speed up (For Gray Scale Images): 228.2%

Obtained Speed up (For Indexed Images): 190.05%
Obtained Speed up (For Dicom Images): 186%

Following are the some random examples of gray scale, indexed and Dicom images with their segmented region using Otsu's Thresholding method by this proposed method.

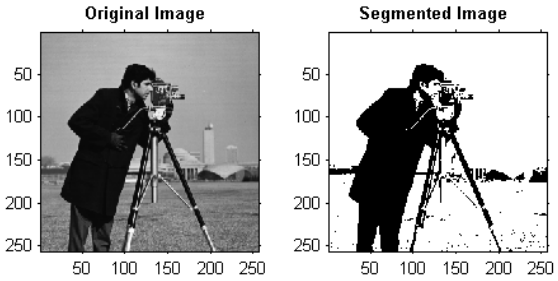


Figure 3: Segmentation on Gray Scale Image

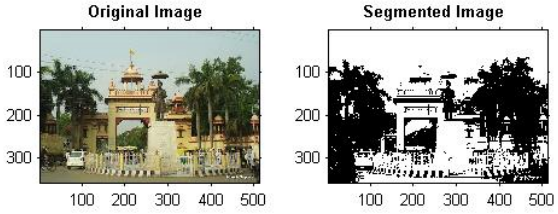


Figure 4: Segmentation on Indexed Image

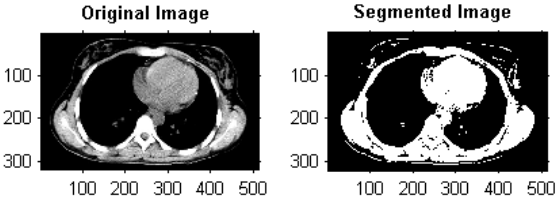


Figure 5: Segmentation on Dicom Image

VIII. Discussion & Conclusion

This paper proposed a considerable approach using MATLAB with java threads for implementing image segmentation parallely. MATLAB is required for implementing different kind of simulation and processing. It also consists of very effective PCT (parallel computing toolbox) for implementing explicit parallelism. Though, it is also having some kind of execution problem as it use heavy weight processes instead of light weight threads. This work provides the efficient way by creating multiple threads in java and call them appropriately to solve existing problem. We have chosen Otsu's method for image segmentation for its effectiveness, constancy and its ease of computation. It is well performed threshold process and time consumption is very lesser as compare to other thresholding techniques. We have implemented total nine experiments with three image data set on three distinguished multicore environment and found significant speedup. Each image data set consists 12 gray scale, 12 RGB and 12 Dicom images. Main purpose of this paper is to reduce the execution time of the techniques of image processing which take larger time to execute and this results shows this implementation as significant approach with efficiently using all available compute resources. As per future work it can be implemented more effectively by using GPUs and we can also execute different kind of other image processing algorithms used for image analysis and image understanding. This technique can also be used effectively in the field of medical imaging to give fast and promising results which will be useful for radiologists for faultless treatment planning.

References

[1] Thomas Bräunl, "Tutorial in Data Parallel Image Processing" in Australian Journal of Intelligent Information Processing Systems (AJIIPS), vol. 6, no. 3, pp. 164–174, 2001.

[2] Eric Olmedo, Jorge de la Calleja, Antonio Benitez, and Ma. Auxilio Medina, "Point to point processing of digital images using parallel computing" International Journal of Computer Science Issues, Vol. 9, Issue 3, pp. 1-10, May 2012.

[3] Preeti kaur, "Implementation of image processing algorithms on the parallel platform using MATLAB" International Journal of Computer Science & Engineering Technology, Vol. 4, No. 06, pp. 696-706, Jun 2013.

[4] Sanjay Saxena, Neeraj Sharma and Shiru Sharma, "Image Processing Tasks using Parallel Computing in Multi core Architecture and its Applications in Medical Imaging" International Journal of Advanced Research in Computer and Communication Engineering, Vol. 2, Issue 4, pp. 1896-1900, April 2013.

[5] J. Fung and S. Mann, "Using graphics devices in reverse: GPU – based image processing and computer vision", IEEE International Conference on Multimedia and Expo. IEEE, pp. 9-12, June 2008.

[6] Teng-Yi Huang, Yu-Wei Tang, Shiun-Ying Ju , "Accelerating image registration of MRI by GPU-based parallel computation" in Magnetic Resonance Imaging , pp. 712–716, 2011.

[7] JeffreyM. Squyres, Andrew Lumsdaine and Robert L. Stevenson, "A cluster-based parallel image processing toolkit" in IS&T Conference on Image and Video Processing, San Jose, CA, pp. 5–10, Feb 1995.

[8] Chouchene Marwa, Bahri Haythem, Sayadi Fatma Ezahra & Atri Mohamed, "Image Processing Application on Graphics processors", International Journal of Image Processing (IJIP), Volume 8, Issue 3, pp. 66-72, 2014.

[9] K.Manjunathachari and Dr.K.Satyaprasad "Modeling and Simulation of Parallel Processing Architecture for Image Processing" Journal of Theoretical and Applied Information Technology, pp 1 -11, 2005.

[10] Fahim Ahmed M., "Parallel Implementation of K-Means on Multi-Core Processors" in Computer Science and Telecommunications No.1, 2014.

[11] Zhiyi Yang, Yating Zhu, Yong Pu, "Parallel Image Processing Based on CUDA" in International Conference on Computer Science and Software Engineering, 2008.

[12] Sanjay Saxena, Neeraj Sharma, Shiru Sharma, "Region wise processing of an image using multithreading in multicore environment & Its application in medical imaging " in International Journal of computer Engineering & Technology, Vol 4, Issue 4, pp 20 -30, 2013.

[13] Sanjay Saxena, Neeraj Sharma and Shiru Sharma. "Multithreaded Approach for Registration of Medical Images using Mutual Information in Multicore Environment and its Applications in Medical Imaging" International Journal of Computer Applications vol. 113(3), pp. 23-32, March 2015.

[14] Sanjay Saxena, Neeraj Sharma and Shiru Sharma, "Parallel Image Processing Techniques, Benefits and Limitations" in Press. om

[15] Yair Altman, "Explicit Mutithreading in MATLAB", Undocumented MATLAB, 2014. http://undocumentedmatlab.com/blog/explicit-multi-threading-in-matlab-part1.

[16] "Concepts of Java Multithreading" Tutorial Points, Simply Java Learning. http://www.tutorialspoint.com/java/java_multithreading.htm.

[17] In Kyu Park, Nitin Singhal, Man Hee Lee, , Sungdae Cho, and Chris W. Kim, "Design and Performance Evaluation of Image Processing Algorithms on GPUs", IEEE Transactions On Parallel And Distributed Systems, VOL. 22, NO. 1, January 2011.

[18] Cristina Nicolescu and Pieter Jonker, "A data and task parallel image processing environment" in Parallel Computing Vol. 2, pp. 945–965, 2002.

[19] Roberto R. Osorio, Cesar Daz-Resco and Javier D. Bruguera, "Highly Parallel Image Processing on a Massively Parallel Processor Array". www.ac.usc.es/system/files/Jornadas09.pdf.

[20] Piotr Wendykier, "High Performance Java Software for Image Processing" PhD Thesis, 2003.

[21] Jan Lemeire, Yan Zhao, Peter Schelkens, "Towards Fully User Transparent Task and Data Parallel Image Processing".

[22] Dr. Christos Bouganis, "An introductory lecture to the project – Parallel Image Processing".

[23] Muneto Yamamoto and Kunihiko Kaneko, "Parallel Image Database Processing With Mapreduce And Performance Evaluation in Pseudo Distributed Mode" in International Journal of Electronic Commerce Studies, Vol.3, No.2, pp.211-228, 2012.

[24] Emerson C Pedrino and Marcio M Fernandes, "Automatic Generation of Custom Parallel Processors for Morphological Image Processing" in International Seminar on Computer Architecture and High Performance Computing, 2014.

[25] Devrim Akgün, "Performance Evaluations for Parallel Image Filter on Multi - Core Computer using Java Threads" in International Journal of Computer Applications, Volume 74– No.11, pp. 13-19, July 2013.

[26] Alda kika, Silvana Greca, "Multithreading Image Processing in Single-core and Multi-core CPU using Java" , International Journal of Advanced Computer Science and Applications, Vol. 4, No. 9, pp. 165-169, 2013.

[27] "Perform Parallel Computations on multicore computers, GPUs, and computer clusters", Mathworks, http://in.mathworks.com/products/parallel-computing/?s_tid=hp_fp_list.

[28] "Parallel Computing Toolbox, User's Guide" by Mathworks.

[29] Sanjay Saxena, Shiru Sharma, Neeraj Sharma, "Parallel Computation of Mutual Information in Multicore Environment and Its Applications in Medical Imaging" IEEE database in MEDCOM, International Conference on Medical Imaging, m Health & Emerging Communications System, Noida, India, 2014.

[30] Sanjay Saxena, Shiru Sharma, Neeraj Sharma , "Image Registration Techniques Using Parallel Computing in Multicore Environment and Its Applications in Medical Imaging" IEEE International Conference on Computer and Communication Technology, ICCCT 2014, MNNIT (Moti Lal Nehru, National Institute of Technology), Allahabad, September 2014.

[31] Sanjay Saxena, Shiru Sharma, Neeraj Sharma, "Parallel Approaches of Image Processing Techniques and Its implementation in Abdominal Image Segmentation" in Conference on Present Scenario and Future Trends in Biomedical Engineering and HealthCare Technology, School of Biomedical Engineering, Indian Institute of Technology (Banaras Hindu University), Varanasi, India, October 2014.

[32] Sanjay Saxena, Neeraj Sharma, L. M. Aggarawal, Shiru Sharma, "Parallel Processing of Images in Multi Core Environment: It's Applications in Medical Imaging" in Annual Conference of Association of Medical Physicists of India, AMPINC – CON, KGMC Lucknow 2014.

[33] Sanjay Saxena, Neeraj Sharma, Shiru Sharma, "An Intelligent System for Segmenting an Abdominal Image in Multicore Architecture ", in IEEE International Conference and Expo on Emerging Technologies for a smarter world at New York,

USA, Held in New York, United States of America, 2013.

[34] Yu-Hsiang Wang, "Tutorial: Image Segmentation", Graduate Institute of Communication Engineering National Taiwan University, Taipei, Taiwan. Available: http://disp.ee.ntu.edu.tw/meeting/%E6%98%B1%E7%B F%94/Segmentation%20tutorial.pdf

[35] Prabhjot Kaur, A.K. Soni, Anjana Gosain, "Image segmentation of noisy digital images using extended fuzzy C–means clustering algorithm" in International Journal of Computer Applications in Technology, Inderscience Publishers, vol. 47, issue 2-3, 2013.

[36] Ricardo Pérez-Aguila, "Automatic Segmentation and Classification of Computed Tomography Brain Images: An Approach Using One-Dimensional Kohonen Networks" in IAENG International Journal of Computer Science, vol. 37, 2010.

[37] Jian-de Zhang, Jin-gui Lu, Hong-liang Li, "Hybrid particle swarm optimisation algorithm for image segmentation" in International Journal of Modelling, Identification and Control , Inderscience Publishers , Volume 14, Issue 4, 2011.

[38] Siddhartha Bhattacharyya, Ujjwal Maulik , Paramartha Dutta, "A Pruning Algorithm for Efficient Image Segmentation with Neighborhood Neural Networks" in IAENG International Journal of Computer Science, vol. 35:2, 2008.

[39] Rohan Kandwal, Ashok Kumar and Sanjay Bhargava, "Review: Existing Image Segmentation Techniques" in International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 4, pp. 153-156, April 2014.

[40] V.Vijaya Kumar, A. Nagaraja Rao, U.S.N.Raju, and B.Eswara Reddy, "Pipeline Implementation of New Segmentation Based on Cognate Neighborhood Approach" in IAENG International Journal of Computer Science, 35:1, Feb 2008.

[41] "Notes of Automatic Thresholding" in http://www.math.tau.ac.il/~turkel/notes/otsu.pdf.

[42] Miss Hetal J. Vala, Prof. Astha Baxi, "A Review on Otsu Image Segmentation Algorithm" in International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 2, pp. 387-389, February 2013.

[43] "Bringing Java Classes into MATLAB Workspace" by Mathworks. http://in.mathworks.com/help/matlab/matlab_external/bringing-java-classes-and-methods-into-matlab-workspace.html.