International Journal of Pure and Applied Mathematics
Volume 117 No. 15 2017, 117-139
ISSN: 1311-8080 (printed version); ISSN: 1314-3395 (on-line version)
url: http://www.ijpam.eu
Special Issue

ijpam.eu

An Improved Shared Memory Parallelization of Image Segmentation Algorithms in Multicore Architecture

[1]Priya P. Sajan and [2]S.S. Kumar
[1]Department of Computer Applications,
Noorul Islam University,
India.
priyasajans@gmail.com
[2]Electronics and Instrumentation,
Noorul Islam University,
India.
kumar_s_s@hotmail.com

Abstract

Shared memory parallelism is an emerging domain of interest as it focuses in nurturing the computational swiftness. This line of orientation induces processing efficiency so that computing processors could undertake complicated scientific procedures within less time, say seconds or minutes with extreme accuracy. Digital image segmentation is one such real life matter of subject which is possessing huge computational complexity that readily calls for adapting shared memory multicore parallelization to resolve immense processing concerns. This could be attained with OpenMP by exploiting architectural parallelism and could be easily implemented to methods which possess structured parallelism. This research article focuses on the parallelization and implementation of three such image segmentation methods - GVF Active Contour Snake, MAP-ML and JF-Cut whose figuring complexity could be lowered by applying appropriate OpenMP directives. Examination, evaluation and comparison of execution time of these algorithms were deeply analysed with various image types and is successfully deployed in multicore systems. OpenMP remains area of interest for multicore systems as it quickens the computational speed by enabling incremental parallelization of existing sequential methods without much restructuring.

Key Words:Open MP, GVF active contour snake, MAP-ML, JF-Cut.

1. Introduction

Computational speed could be quickened by stimulating the multicore processors which constitutes the profound underlying technique for shared memory parallel processing. Computer vendors more and more rely on shared memory architecture to exploit the hardware processing power. This trend is experiencing increase in velocity due to high demand in computational requirements. Current multicore technology demands development and implementation of parallel processing applications with optimal usage of memory. OpenMP (Open Multi-Processing) is emerging as an interesting and promising shared memory parallel programming methodology for scientific computations as it performs parallelization of existing applications with comparatively lesser modification and minimal implementation cost. In this paper we evaluate the processing performance and efficiency of three memory bound image segmentation methods: GVF Active Contour Snake, MAP-ML and JF-Cut by iteratively tuning to parallelized C version with *omp* directives to reach maximum threshold. To be more precise, OpenMP allows incremental formulation for constructing the parallel version of an existing sequential algorithm that exhibits structured parallelism in task, instruction and data level with promising accuracy.

2. Related Works

To briefly summarize the existing work, enormous studies has been made in parallelizing of GVF Snake, MAP-ML and JF-Cut methods on distributed parallel architecture. Rigo, Juan and Julio parallelized segmentation of medical images with deformable models using GPU with CUDA and OpenMP [1]. In [2] Philips, Watson and Wynein conducted hybrid image classification and parameter selection with shared memory concept. Segmentation of skin cancer images was conducted by Mahmoud and Jumaily [3] with GVF Snake. Pallippuram and Bhuiyan [4] made a comprehensive comparison on various GPU programming models. Application of OpenMP and its various strategies is well described by Chapman, Jost and Van Der Pas in [5]. In [6], He Z and Kuester made an attempt to parallelize GVF using OpenGL in GPU environment. Zheng and Zhang [7] tried to parallelize GVF using OpenGL in GPU environment. Novel segmentation algorithm was implemented on many core CPU for accelerating the simulation of brain tumor proliferation [8]. W. Kim and C. Kim [9] explained salient energy model for driving the active contours. Probabilistic model based image segmentation and objective evaluation of segmentation algorithms are done in [11] and [12]. Evaluation and impact of parallelizing image segmentation methods has been studied in depth in [13] and [14]. In [15], a parallel graph cut approach for large scale image and video is well mentioned. Grig computing in structured grids for cache efficient graph cut methods are explained in [16] by Jamriska and in [17] by Jiu and SUn.

Yet these scientific implementations results in promising output, their mode of implementations, complexity in programming and adaptations to the existing environment are not feasible in terms of execution time and cost factor. Designing the parallel versions of segmentation methods with the referentially cited papersare comparatively difficult with limited instruction set in distributed parallelization mode. All optimizations are done on hardware and compiler structure to maximize parallel execution in GPU so that more number of processing hardware is required resulting to be less cost effective. This paper is structured by specifying the details of existing methods with their pros and cons and the mode by which their performance is accelerated by parallelizing with OpenMP. Performance analysis is described with execution time, OpenMP constructs and algorithm efficiency analysis to determine which of the methods viz GVF active Contour Snake, MAP-ML and JF-Cut exhibits more parallelism with less processing time without compromising the accuracy.

3. Existing System

GVF Active Contour Snake

Gradient Vector Flow is a deformable model that detect shape with boundary concavities with extenal force field called Active Contour Snake. Itadapts model topology for splitting or merging parts during object deformation. GVF provides an advancement to the traditional parameterized snake X(s) = (X(s), Y(s)), s $\in$[0,1], that minimizes the energy functional. Pixel allocation starts with the calculation of a field of external forces which is derived from Gaussian Diffusion function so that the contour regions remain connected. This extends the image capture range so that snakes can find objects far away from snake's initial position. This diffusion forces with the usual internal forces yields a powerful and accurate segmented model for images with irregular shape.

The principal fall behind GVF active contour snake is that it suffers from huge and complex arithmetic calculations which eventually raises the entire process execution time.

MAP-ML Method

Maximum-a-Posteriori(MAP) Maximum-Likelihood (ML) constitute probabilistic model for image segmentation with iterative optimization scheme by clustering relevant regions as well. For a given image P, 4 dimensional vector is used to represent features of every pixel $I(p) = (I_L(p), I_a(p), I_b(p), I_t(p))^T$where $I_L(p)$, $I_a(p)$and $I_b(p)$ are the components of pixel in rgb colour space and $I_t(p)$ denotes texture feature. Pixels belonging to the region of interest should possess some similar features and is represented by a vector $\Phi(i)$, where $\Phi(i) = (\bar{I}_L(i), \bar{I}_a(i), \bar{I}_b(i), \bar{I}_t(i))^T$. If P and Φ are known optimal label configuration is segmented that maximize posterior possibility of label configuration. From the label configuration, MAP is modelled with Markov Random Field(MRF) and ML with Gaussian Model.

MAP estimation detects the edges of the image to be segmented and ML refines those segmented pixels.

Computational complexity of this method depends on number of pixels in image, attributes required denote a pixel belonging to a region of interest and approximation of total number of iterations for the estimation of MAP-ML region.

JF-Cut Method

Jump Flooding is a graph cut scheme which adapts Push Re-label method using Breadth First Search(BFS) scheme for image segmentation followed by a global re-labelling. This is performed by computing the lower bound of node's distance from source to sink. Different active nodes can be pushed or relabelled simultaneously there by exhibiting high degree of parallelism. BFS is performed staring from the source node to check if there a path exists to destination. If yes, push relabel continues else min cut is found and partitions the graph. Then Nearest Seed Point(NSP) of current point is replaced with nearest one of all its neighboring points.

Assigning a label and grouping of unstructured blocks are time consuming complicated process. While detecting the convergent regions in an image which is to be segmented, relabelling has to be continued by partitioning the entire graph into two parts. Implementation of this concept needs nested iterations thereby intensely take up the execution clock time.

Relatively unspecified studies had been laid down with the parallel implementation of image segmentation in shared memory multicore systems with OpenMP. This research paper eyes on parallelizing three such image segmentation methods - GVF Active Contour Snake, MAP-ML and JF-Cut with OpenMP as it is highly adapted to multicore processing architecture, easy to parallelize with less modification to existing code, better cost effectiveness and easily accessible in multiple platforms. Parallelization and execution time comparison of these methods were experimented and evaluated in sequential and parallel mode, analysedperformance with multiple flavours of OpenMP constructs and finally assessed the algorithm performance efficiency.

4. Proposed System with Open MP Programming Model

OpenMP remains of interest for multicore CPUs as it accomplishes fork-join model of parallelism using threads, with the number of threads being equal to the number of underlying processors. Parallelism with OpenMP is specified through compiler directives which are embedded in C/C++ source code so that sequential methods could be incrementally parallelised without major restructuring. It is rich with run time library routines and environment variables

which enables the programmer to give instruction to the compiler regarding how to multi thread and parallelize a specific block of code and the manner by which to distribute the task, instructions or data in a balanced and impartial way among the multiple co-processors.

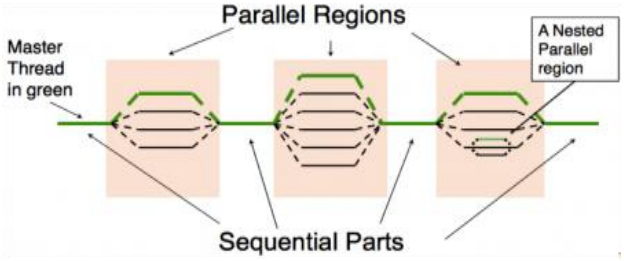


Figure 1: OpenMP Fork-Join Model

All OpenMP programs begin as single *initialmasterthread* which executes sequentially until the first parallel construct is encountered. The *master* then creates a team of co-processing parallel threads often termed as team threads which executes simultaneously in multicore systems. After completion of process in parallel construct, the co-processing threads synchronize to join back together and terminate leaving behind only the *initialmaster* thread. If any of the single forked thread commences its execution, it will resume with its terminal state and wait until the entire co-processing threads completes the whole process.

With parallelized iterative tuning of GVF Active Contour Snake, MAP-ML and JF-Cut, the user is able to control each data point of problem space in source code and to decompose them in flexible way. For this we adopted OpenMP language extension in appropriate manner so as to get maximum throughput and finally measured algorithms performance efficiency.

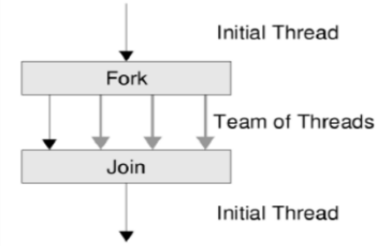


Figure 2: OpenMP Thread Execution with Fork-Join Model

Parallel GVF Active Contour Snake Algorithm with OpenMP

Gray level scale is calculated for each pixel position in the image and are then partitioned depending on their intensity value. In the proposed segmented area, initial seed points are calculated from pixel map based on the binarized gradient value.

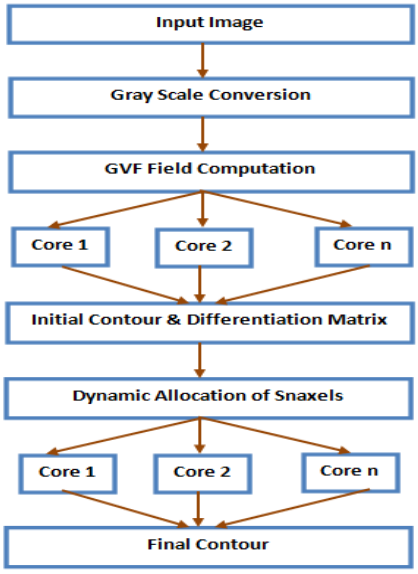


Figure 3: Proposed Parallel Active Contour Snake Flow Chart

```
#pragma omp parallel shared(n)
{       for(i=0;i<=n;i++)
        #pragma omp master
            i)  compute length of contour region
            ii) compute the rgb values to gray scale

        #pragma omp for private
            i)  calculate each point of contour region

        #pragma omp for nowait()
            i)  boundary check with pixel allocation for GVF field computation

        #pragma omp for atomic nowait()
            i)  initial contour identification
            ii) dynamic allocation of snaxels

        #pragma omp for ordered()
            i)  final contour extraction
}
```

Figure 4: Proposed Parallel Active Contour Snake Pseudocode

For calculating the gradient value of the input image, the RGB value is converted into grayscale which will be stored in an array for further processing. Threshold value for each pixel will be calculated based on the internal energy values. This will be a weighted sum which determines the gradient value for each pixel position. The pixels will then be classified depending on this gradient

intensity value. This is a time consuming process because the intensity value of each pixel in the gradient field has to be individually processed and calculated. This could be parallelized with level constructs of OpenMP. Flow chart and pseudocode of proposed parallel GVF Snake method is described.

The task of gray scale parallelization is attained by *#pragma omp parallel* according to the number of threads set created. *Master* construct defines and initializes the contour region and is distributed to compiler generated threads. Data within *master* region process simultaneously without awaiting the result generated by co-processing threads. Instruction for calculating the gradient value and its corresponding intensity is paralleized with iterative *for* loops. The first *for* loop shares the parameters for contour generation in *private* manner and the second *for* loop as *nowait.* Practically it may requires additional computation after the *master* construct been initialized. So paralleization of outer loop is the better choice. O*mp* directive *private* is also used here because it directs the compiler to make variables private so that multiple copies of the same variable will not execute again and again. By adopting this mechanism, it is seen that the execution time could be reduced to one eighth when compared with the same sequential mode and when executed in an 8 core system.

Parallel MAP-ML Method with OpenMP

MAP-ML uses iterative optimization scheme for image segmentation by clustering the relevant regions. MAP is modeled with Markov Random Field(MRF) and ML is achieved with Gaussian model. MAP estimation detects the edges of the image to be segmented and ML refines those segmented pixels.

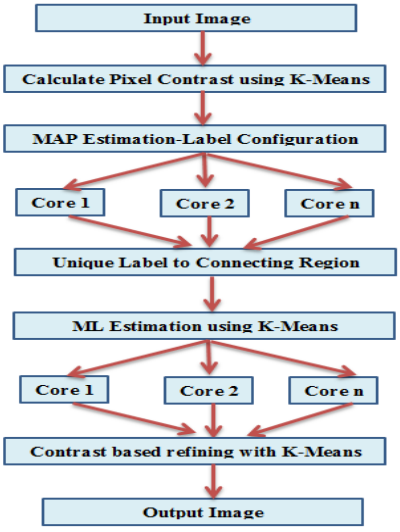


Figure 5: Proposed Parallel MAP-ML Flow Chart

```
#pragma omp parallel shared(n), private(i)
{       for(i=0;i<=n;i++)
        #pragma omp for
          1. calculate pixel contrast
          2. iterative optimization for MAP estimation
        #pragma omp parallel
          1. relabelling of connecting region
        #pragma omp sections
        {  #pragma omp section
            1. dynamic clustering of pixels
           #pragma omp section
            1.  pixel refinement with K-means clustering
        }
        #pragma omp for nowait
          1.  contrast based refining with K-means
        #pragma omp ordered()
          1. final segmented region
}
```

Figure 6: Proposed Parallel MAP-ML Pseudocode

Edge detection using MRF could be shared and distributed among multiple cores through *omp parallel* directives. This task results in unique labelling of connecting regions where segmented pixels are refined and clustered with K-Means clustering method which possess stuctured parallelism.

Number of pixels in the refinement area will be shared and accessed parallely for MAP estimation. Data read for re-labelling and dynamic clustering of pixels is attained by parallelising K-Means clustering using *ompparallel* directive and is shared among co-processing threads. The sections construct enables different threads to carry out different works by task parallelism. *Nowait* clause will override the implicit barrier for the work sharing process in contrast based refining phase through iterative loops. Output will be produced in an ordered construct so that execution of structured block within the parallel loop will be in sequential way to avoid data race within the associated block.

Parallel JF-Cut Method with OpenMP

JF-Cut uses push re-label method with BFS(Breadth First Search) scheme for relabeling and computing lower bound of node's distance from source to sink. Different active nodes can be pushed or relabeled simultaneously. This exhibitsdata and task parallelism. Starting from the source, performs BFS to check whether a path exists**.** Ifno, min cut found and partitions the graph andif yes, push re-label continues. Then NSP(Nearest Seed Point) of current point is replaced with nearest one of all its neighbouring points.Process begins by initialising all seed blocks that contains nodes.Then perform jump flooding to find NSP using BFS.

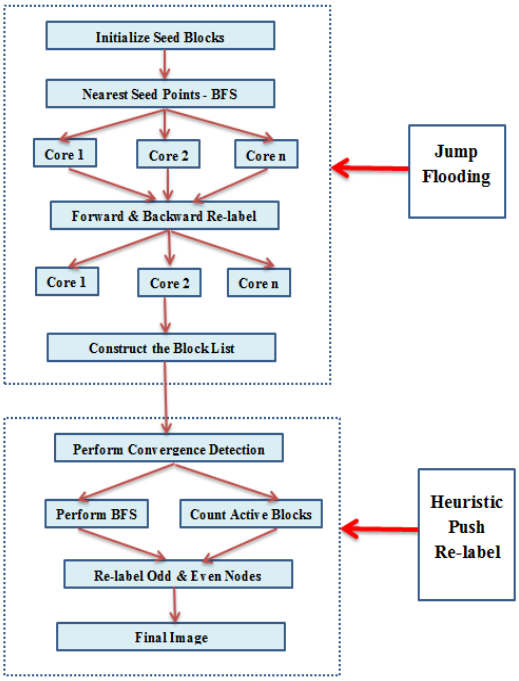


Figure 7: Proposed Parallel JF-Cut Flow Chart

```
//Jump Flooding
#pragma omp parallel shared(seed nodes)
{       #pragma omp for each seed points
          1. scan primitive on distance histogram with BFS
          2. perform scan primitive on distance histogram for forward and backward
             relabel
        #pragma omp for nowait()
          1. construct block list
}
  //Heuristic Push Re-label
#pragma omp parallel
{         1. perform JF BFS to build block list
          1. repeat until detection failed or no more nodes marked
        #pragma omp critical
          1. count active blocks
          2. re-label active nodes- odd and even
        #pragma omp ordered
          1. final segmented image
}
```

Figure 8: Proposed Parallel JF-Cut Pseudocode

This task could be parallelized among multiple cores evenly. Also perform the scan primitive to compute distance histogram from the constructed block list. Then compute convergence detection through push and relabel operations inside the block list in a repeatitive manner. This synchronized operations enables to adapt data and instruction parallelism which inturn improves propogation speed of information..

After initializing the number of seed blocks, parallelization is applied with *omp for* to find NSP using BFS. Followed by this, relabelling of seed points is made through push and relabel operations. This task when parallelized override the construct block list.

Hence using *nowait* clause once a thread finishes the work of relabelling seeds, it will come with an associated resultant block list. Entire seed block list is futher finalised by repeatedly checking whether the nodes being marked or not. After getting the active count, the *critical* construct ensures no attempt is made to update same active blocks simultaneously and result will be generated using *ordered* construct.

5. Experimental Results

The results have been analysed using images downloaded from internet within the range of 64×64 to 2048×2048 pixels in size. Experimental environment is Windows10 64-bit Intel(R) Core(TM) i7-3770 CPU @ 3.40GHz and 8GB RAM. Investigations were conducted on these large data sets to analyze the practicality and efficiency of parallelizing GVF Active Contour Snake, MAP-ML and JF-Cut with OpenMP.

The processing time of the sequential version of these digital image segmentation methods which is implemented in Matlab, Java and C language is presented for the reference. Indepth quantitative analysis has been made to evaluate the sequential as well as parallel performance efficiency of the selected image segmentation methods.

Execution Time Based Analysis

Processing time for the proposed parallel version of GVF Active Contour Snake, MAP-ML and JF-Cut methods were executed and analysed both in sequentialas well as parallel mode in Windows dual core, quad core and 8 core systems.

From the theoretical studies it was revealed that the result will improve as the number of processing cores increases.But in practical scenario, there comes a point where the result becomes stagnant even if the number of processing cores increases indicating that the processors and complier thread communication has reached the maximum level. The experiment was conducted on various image types.

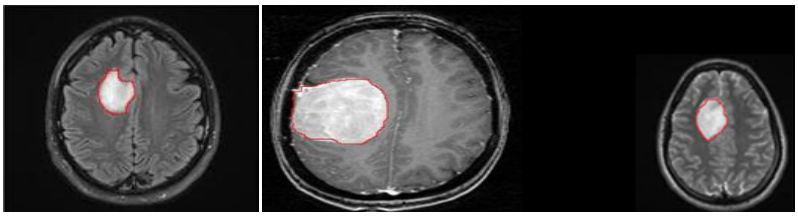


Figure 9:527kb Figure 10: 590kb Figure 11: 600kb

Figure 12: 820kb Figure 13: 4mb Figure 14: 6mb

a) Sequential and Parallel Execution Time Analysis

Execution time comparison chart for analysing the processing performance and efficiency ofeach of GVF Active Contour Snake, MAP-ML and JF-Cut methods are described with corresponding tables and graphical charts.

1. GVF Active Contour Snake

Table 1: GVF Active Contour Snake Time Comparison

Image Size	**Sequential (in secs)**			**Parallel (in secs)**	***Speed - Ratio Analysis with C and OpenMP***
	Matlab	***Java***	***C***	***OpenMP***	
527 KB	4.95	4.56	4.38	1.11	3.94
590 KB	5.02	4.79	4.60	1.19	3.8655
600 KB	5.82	5.42	5.13	1.31	3.9160
820 KB	6.91	6.33	6.08	1.85	3.2864
4 MB	13.09	12.19	10.56	4.92	2.1463
6 MB	25.32	22.39	21.04	9.80	2.1469

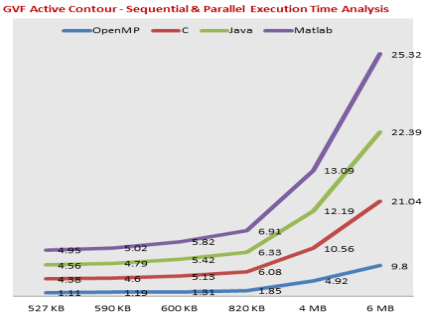


Figure 15: GVF Active Contour Time Chart

2. MAP-ML

Table 2: MAP-ML Time Comparison

Image Size	**Sequential (in secs)**			**Parallel (in secs)**	*Speed - Ratio Analysis with C and OpenMP*
	Matlab	***Java***	***C***	***OpenMP***	
527 KB	5.49	5.02	4.87	1.48	3.290
590 KB	5.86	5.28	5.06	1.88	2.691
600 KB	6.83	6.72	5.44	2.33	2.334
820 KB	7.60	7.11	6.89	3.52	1.957
4 MB	13.25	12.37	11.39	6.64	1.715
6 MB	27.69	24.26	23.63	11.37	2.078

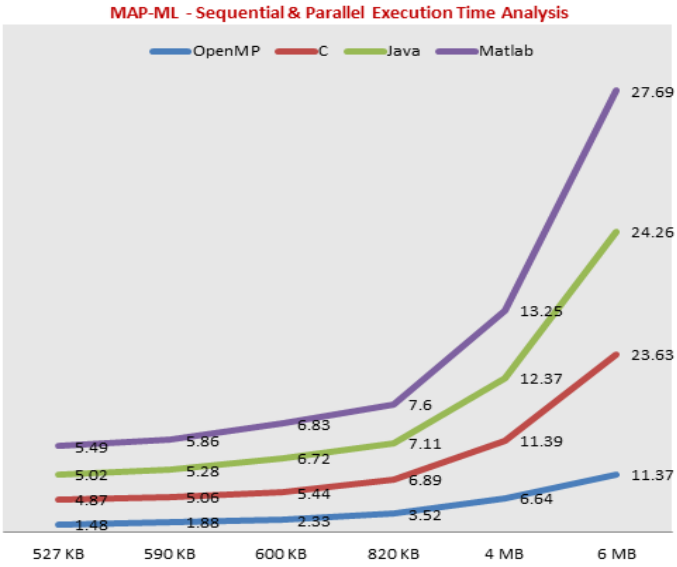


Fig16: MAP-ML Time Chart

3. JF-Cut Method

Table 3: JF-Cut Time Comparison

Image Size	**Sequential (in secs)**			**Parallel (in secs)**	*Speed - Ratio Analysis with C and OpenMP*
	Matlab	***Java***	***C***	***OpenMP***	
527 KB	5.97	5.82	4.84	1.25	3.872
590 KB	6.76	6.26	5.37	1.57	3.420
600 KB	7.07	6.77	5.93	1.64	3.615
820 KB	7.68	7.18	6.32	2.30	2.747
4 MB	12.75	12.03	11.41	5.83	1.957
6 MB	27.25	24.63	22.52	10.07	2.236

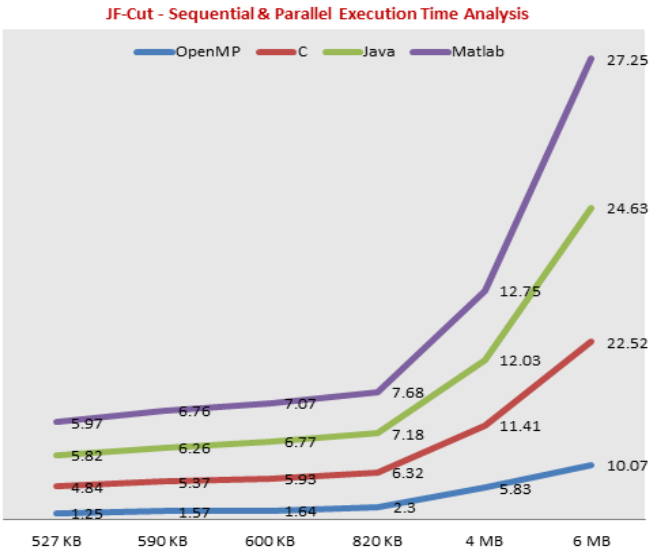


Figure 17: JF-Cut Time Chart

Table1, *Table2* and *Table3* describes the sequential (Matlab, Java and C) and parallel execution time analysis (OpenMP) measured in seconds for various types of images.Also, the tables describes the best computation time by speed-ratio analysis with C and OpenM

b) Algorithm Efficiency Analysis

Efficieny of GVF Active Contour, MAP-ML and JF-Cut methods were analysed in terms of their total execution time and measured in seconds.

From the result obtained, it is lighted that of the three image segmentation methods considered, GVF Active Contour Snake exhibits more parallelism followed by JF-Cut and MAP-ML methods.

To get more precision on these results further indepth experiments were conducted on multiple layers of OpenMP constructs.

Table 4: Parallelization with OpenMP

	Parallelization with OpenMP (in sec)		
Image Size	***GVF Active Contour Snake***	***MAP-ML***	***JF-CUT***
527 KB	1.11	1.48	1.25
590 KB	1.19	1.88	1.57
600 KB	1.31	2.33	1.64
820 KB	1.85	3.52	2.30
4 MB	4.92	6.64	5.83
6 MB	9.80	11.37	10.07

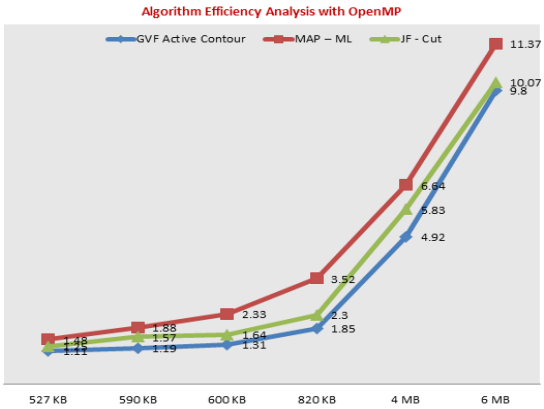


Figure 18: Algorithm Efficiency Analysis with OpenMP

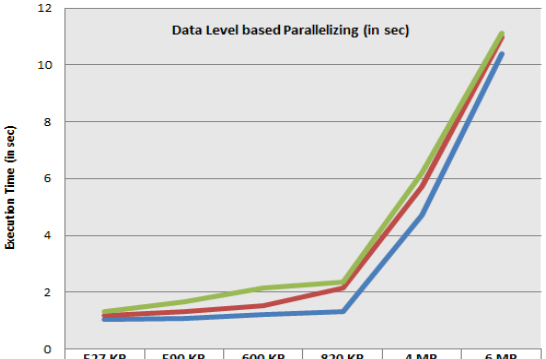
c) OpenMP Performance Analysis

Shared memory parallelism is attained with OpenMP by orchestrating the action of threads in multiple clauses viz level constructs, work-sharing constructs, synchronization constructs and data property constructs.

1. Level Constructs

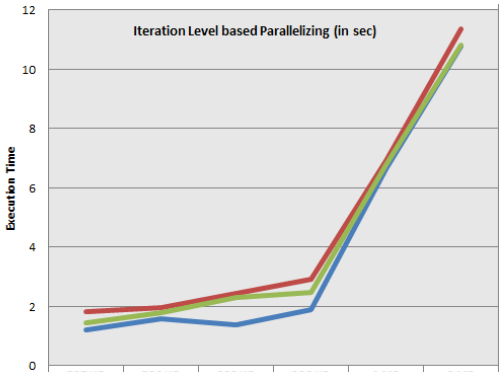
With level constructs, same computations are performed on large set of data by adopting parallelism in data level, iterations level, instruction level and task level.

GVF Active Contour Snake method exhibits more parallelism in data level and iterations level for pixel allocation and computation. Meanwhile JF-Cut possess better parallelism in instruction level as well as task level for image segmentation with multiple instructions of jump flooding for calculating NSP using BFS method and re-labelling of pixels for convergence detection.



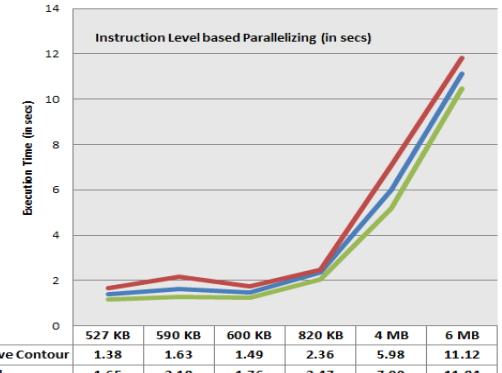
	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.05	1.08	1.2	1.31	4.71	10.39
MAP - ML	1.19	1.32	1.52	2.16	5.7	10.97
JF - Cut	1.32	1.68	2.15	2.35	6.19	11.13

Figure 19: Data Level Constructs



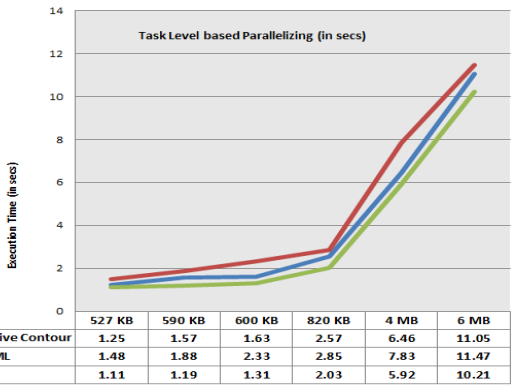
	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.22	1.58	1.37	1.89	6.65	10.78
MAP - ML	1.83	1.97	2.43	2.93	6.91	11.37
JF - Cut	1.44	1.8	2.29	2.48	6.8	10.82

Figure 20:Iteration Level Constructs



	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.38	1.63	1.49	2.36	5.98	11.12
MAP - ML	1.65	2.18	1.76	2.47	7.09	11.84
JF - Cut	1.16	1.28	1.23	2.04	5.18	10.46

Figure 21: Instruction Level Constructs



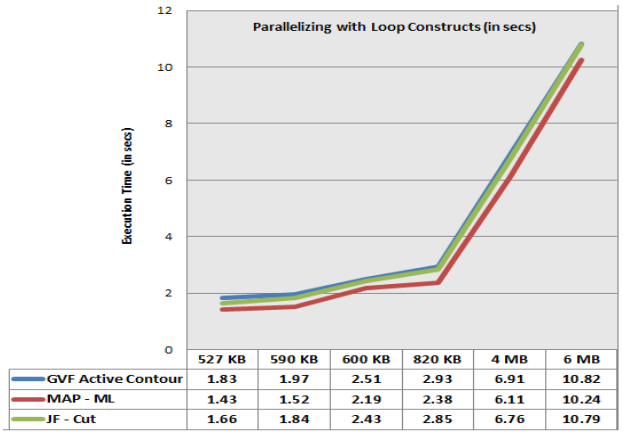
	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.25	1.57	1.63	2.57	6.46	11.05
MAP - ML	1.48	1.88	2.33	2.85	7.83	11.47
JF - Cut	1.11	1.19	1.31	2.03	5.92	10.21

Figure 22:Task Level Constructs

2. Work Sharing Constructs

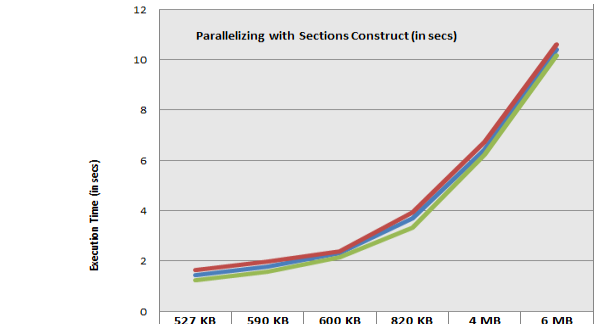
Computations to be performed will be distributed among threads with work sahring constructs. Loop constructs causes iterations of the loop to be immediately executed in parallel and is confronted by MAP-ML method with multiple iterations in single loop for calculating the boundaries of resultant image by unique labelling with graph-cut. With Sections construct threads have variable quantum of tasks to perform and take variable time for completion. For jump flooding and push re-labelling, each thread executes one block at a time and each code block will be executed exactly once.

Single construct is associated with structured code block and is executed by a single thread alone. JF-Cut out performs with single construct for building a constructive block list of re-labelled nodes by invoking exactly a single thread and therby increases data output consistency.



	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.83	1.97	2.51	2.93	6.91	10.82
MAP - ML	1.43	1.52	2.19	2.38	6.11	10.24
JF - Cut	1.66	1.84	2.43	2.85	6.76	10.79

Figure 23: Loop Constructs



	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.43	1.79	2.31	3.68	6.41	10.39
MAP - ML	1.65	1.97	2.39	3.93	6.74	10.61
JF - Cut	1.23	1.58	2.14	3.34	6.22	10.16

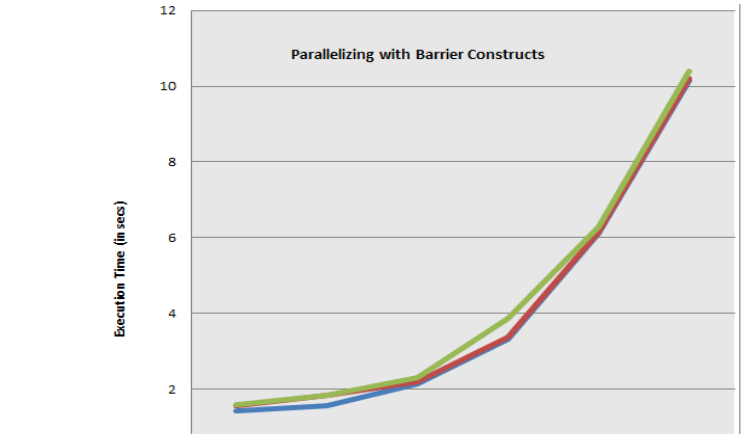
Figure 24: Sections Constructs

	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.12	1.19	1.56	1.8	5.58	9.77
MAP - ML	1.15	1.21	1.65	1.86	5.85	9.8
JF - Cut	1.08	1.09	1.48	1.28	5.27	9.19

Figure 25: Single Constructs

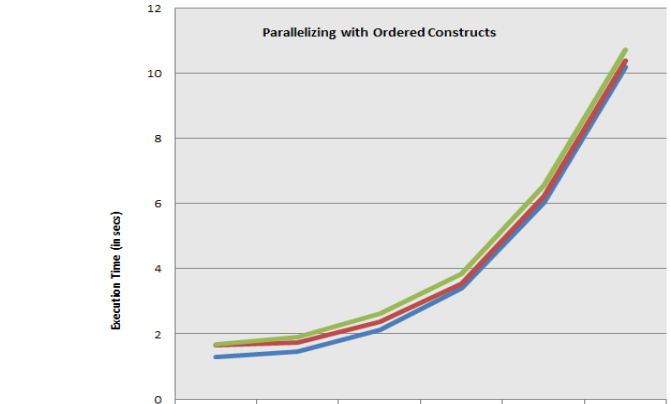
3. Synchronization Constructs

Access to shared data by multiple co-processing threads is associated with synchronization constructs of OpenMP. With orderly accessing of pixels, no pixels could wait longer with barrier constructs and is well suited with GVF Active Contour method. Also with this method.



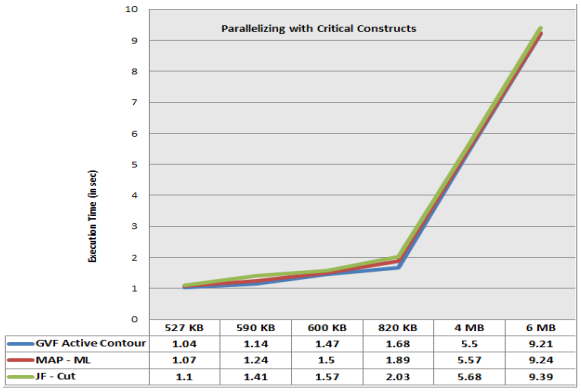
	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.43	1.58	2.13	3.33	6.13	10.16
MAP - ML	1.56	1.83	2.19	3.39	6.16	10.22
JF - Cut	1.6	1.85	2.32	3.86	6.28	10.41

Figure 26: Barrier Constructs



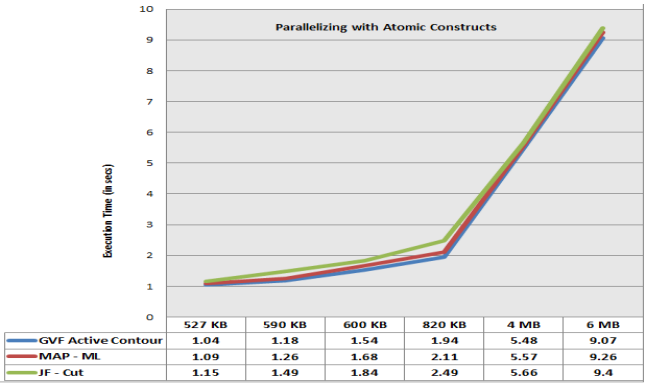
	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.29	1.45	2.13	3.4	6.04	10.18
MAP - ML	1.66	1.74	2.37	3.54	6.22	10.37
JF - Cut	1.68	1.91	2.62	3.83	6.56	10.71

Figure 27: Ordered Constructs



	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.04	1.14	1.47	1.68	5.5	9.21
MAP - ML	1.07	1.24	1.5	1.89	5.57	9.24
JF - Cut	1.1	1.41	1.57	2.03	5.68	9.39

Figure 28: Critical Constructs



	527 KB	590 KB	600 KB	820 KB	4 MB	6 MB
GVF Active Contour	1.04	1.18	1.54	1.94	5.48	9.07
MAP - ML	1.09	1.26	1.68	2.11	5.57	9.26
JF - Cut	1.15	1.49	1.84	2.49	5.66	9.4

Figure 29: Atomic Constructs

snaxel allocation is performed in an sequential ordered manner bounded inside a parallel loop by single and nested loop parallelization. GVF also produce better results with critical construct where multiple threads do not update same shared data simultaneously with GVF field computation for max and min pixel calculation and atomic constructs with dynamic allocation of snaxels where range of application is strictly restricted.

4. Data Property Constructs

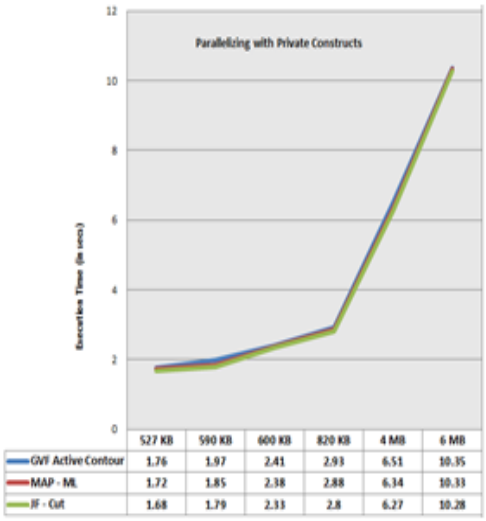


Figure 30: Private Constructs

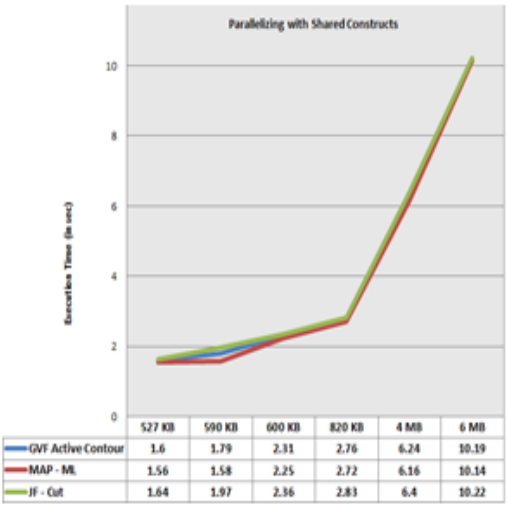


Figure 31: Shared Constructs

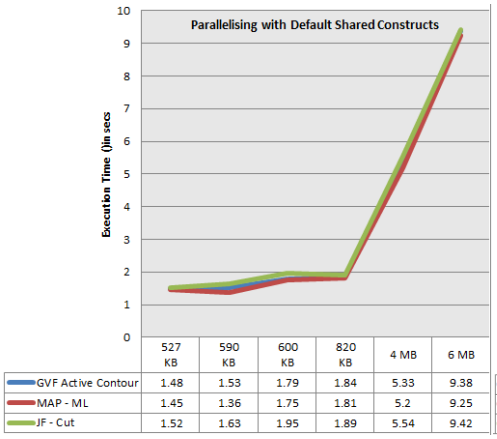


Figure 32: Default Shared Constructs

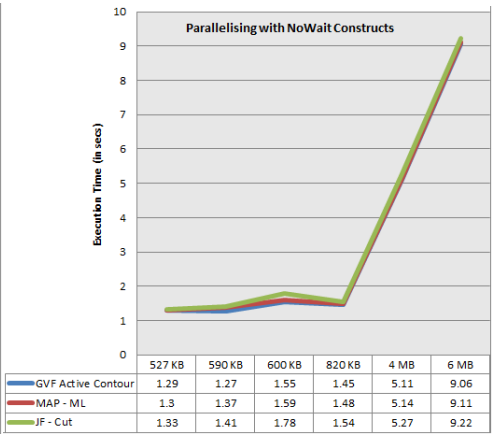


Figure 33: Nowait Constructs

Private clause replicates each thread in the team of threads by having exclusive access so that changes constituted to the data by one processing thread are not perceptible to other co-processing threads. This feature well suits for forward and backward labelling of pixels and to count active blocks of pixels push re-

labelling. In MAP-ML, pixels within a region will be sharing homogenous features of intensity and texture that enables this method most suited and well appropriate to maximize parallelism with *shared* and *default shared* OpenMP constructs. By using *nowait* clause parallelism nature of JF-Cut will be fine tuned for calculating the differentiation matrix and evolution of contour map with pixels having greater values at image edges.

6. Conclusion and Future Scope

Optimization of algorithms that exhibits parallelism will improves processing efficiency of methods and has high impact on compiler architecture. Hence it must be ensured that translation of OpenMP does not excessively affect traditional optimization ability of compiler. Challenges faced with optimization of GVF Active Contour Snake, MAP-ML and JF-Cut was that the application of *omp* constructs in different mode yields various interesting results. Recent platforms may provide features for achieving additional level parallelism that may interact with OpenMP and may improves translation of OpenMP constructs which may always make this topic alive in scientific research.

References

[1] Alvarado R., Tapia J.J., Rolón J.C., Medical image segmentation with deformable models on graphics processing units, The Journal of Supercomputing 68 (1) (2014), 339-364.

[2] Phillips R.D., Watson L.T., Wynne R.H., Hybrid image classification and parameter selection using a shared memory parallel algorithm, Computers & Geosciences 33 (7) (2007), 875-897.

[3] Mahmoud M.K.A., Al-Jumaily A., Segmentation of skin cancer images based on gradient vector flow (GVF) snake, IEEE International Conference on Mechatronics and Automation (ICMA) (2011), 216-220.

[4] Pallipuram V. K., Bhuiyan M., Smith M. C., A comparative study of GPU programming models and architectures using neural networks, The Journal of Supercomputing 61 (3) (2012), 673-718.

[5] Chapman B., Jost G., Van Der Pas R., Using OpenMP: portable shared memory parallel programming, MIT press 10 (2008).

[6] He, Z., Kuester, F., GPU-based active contour segmentation using gradient vector flow, Advances in Visual Computing (2006), 191-201.

[7] Zheng Z., Zhang, R., A GPU-accelerated GVF snake algorithm, IEEE Workshop on Digital Media and Digital Content Management (DMDCM) (2011), 233-236.

[8] Karantasis K.I., Polychronopoulos E.D., Panourgias K.T., Ekaterinaris J.A., Accelerating the simulation of brain tumor proliferation with many-core GPUs, Journal of Computational Science 3 (5) (2012), 306-313.

[9] Kim W., Kim C., Active contours driven by the salient edge energy model, IEEE Transactions on Image Processing 22 (4) (2013), 1667-1673.

[10] Chen S., Cao L., Wang Y., Liu J., Tang X., Image segmentation by MAP-ML estimations, IEEE Transactions on Image Processing 19 (9) (2010), 2254-2264.

[11] Mrudula Karande, Kshirsagar D.B., Probabilistic Model Based Image Segmentation, International Journal of Multimedia and its Applications (IJMA) 6 (2) (2014).

[12] Unnikrishnan R., Pantofaru C., Hebert M., Toward objective evaluation of image segmentation algorithms, IEEE transactions on pattern analysis and machine intelligence 29 (6) (2007), 929-944.

[13] Shen J., Fang J., Sips H., Varbanescu A.L., An application-centric evaluation of Open CL on multi-core CPUs, Parallel Computing 39 (12) (2013), 834-850.

[14] Wan J., Liu Y., Hybrid MPI-OpenMP Parallelization of Image Reconstruction, JSW 8 (3) (2013), 687-693.

[15] Peng Y., Chen L., Ou Yang F.X., Chen W., Yong J.H., JF-cut: A parallel graph cut approach for large-scale image and video, IEEE Transactions on Image Processing 24 (2) (2015), 655-666.

[16] Jamriška O., Sýkora D., Hornung A., Cache-efficient graph cuts on structured grids, IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012), 3673-3680.

[17] Liu J., Sun J., Parallel graph-cuts by adaptive bottom-up merging, In IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2010), 2181-2188.

[18] RAJESH, M. "A SYSTEMATIC REVIEW OF CLOUD SECURITY CHALLENGES IN HIGHER EDUCATION." The Online Journal of Distance Education and e- Learning 5.4 (2017): 1.

[19] Rajesh, M., and J. M. Gnanasekar. "Protected Routing in Wireless Sensor Networks: A study on Aimed at Circulation." Computer Engineering and Intelligent Systems 6.8: 24-26.

[20] Rajesh, M., and J. M. Gnanasekar. "Congestion control in heterogeneous WANET using FRCC." Journal of Chemical and Pharmaceutical Sciences ISSN 974 (2015): 2115.

[21] Rajesh, M., and J. M. Gnanasekar. "Hop-by-hop Channel-Alert Routing to Congestion Control in Wireless Sensor Networks." Control Theory and Informatics 5.4 (2015): 1-11.

[22] Rajesh, M., and J. M. Gnanasekar. "Multiple-Client Information Administration via Forceful Database Prototype Design (FDPD)." IJRESTS 1.1 (2015): 1-6.

[23] Rajesh, M. "Control Plan transmit to Congestion Control for AdHoc Networks." Universal Journal of Management & Information Technology (UJMIT) 1 (2016): 8-11.

[24] Rajesh, M., and J. M. Gnanasekar. "Consistently neighbor detection for MANET." Communication and Electronics Systems (ICCES), International Conference on. IEEE, 2016.