Distributed Parallel Method for Efficient Fractal Image Encoding

Akhilesh Kumar
Faculty of Engineering and Technology
Shri Shankaracharya Technical Campus Bhilai, India

G R Sinha
Faculty of Engineering and Technology
Shri Shankaracharya Technical Campus Bhilai, India

Vikas Dilliwar
Chhattisgarh Institute of Technology
Rajnandgaon, India

ABSTRACT
Fractal image compression reduces amount of redundancy to great extent using suitable affine transformations. This requires less number of bits to encode the same image. However the process of encoding requires enormous computational processing to generate required fractal codes. A distributed parallel method is proposed to reduce computational time by portioning and distributing the input image among different computing nodes, as each computing node performs encoding and block matching individually; which results in significant reduction in processing time to generate fractal codes. This paper presents a review of different parallel algorithms and architecture that has been applied to enhance the speedup also can be used for fractal image encoding task.

General Terms
Parallel Fractal Image Compression, Distributed Parallel Architecture, Mapping Procedure.

Keywords
Fractal image compression, CN's (Computing Nodes), Iterated Function System (IFS), Domain offset, Speedup, Load balancing.

1. INTRODUCTION
The process of representing an image into digital form requires large storage space. Several methods are available in existing literatures which exploit the redundancy in an image and produce encoded image that requires fewer bits than the original image. However, these methods result in a very low compression ratio. The compression ratio of an image can be improved by using fractal image encoding method which exploits inter image redundancy of an image but fractal encoding process is computationally intensive but the decoding phase is faster and can be decoded at any level of resolution. Barsney [1] suggested collage theorem which characterizes an IFS (Iterated Function System) then Jacquin [2] discovered an algorithm which partitioned an image into range and domain blocks; where the size of the range block is smaller than the domain; and the process is to search the best matching domain block from the range block which requires high computation resources. Numerous parallel algorithms and architecture are existing to speedup Fractal image encoding but the most appropriate papers highlighting the parallel implementation of fractal image compression is Min, Palazzari and Cao respectively [4, 8, 11]. A parallel architecture is developed and an algorithm is implemented to minimize the communication overhead and processing time.

This paper is organized as follows: Section 2 presents the literature review of the papers related to the parallel fractal image compression. Section 3 presents the introduction of parallel fractal image compression. Section 4 discusses challenges in parallelizing fractal encoding. In Section 5 suggests parallel distributed architecture and encoding algorithm which may reduce the domain search and processing, resulting in high speedup with satisfactory PSNR. The results are reported in Section 6. Section 7 highlights the conclusions related to parallel fractal image compression.

2. LITERATURE REVIEW
The sequential fractal image encoding procedure requires very high computational time and hence a parallel architecture and efficient algorithm are required to enhance fractal encoding. There have been different parallel algorithms and architectures presented to increase the efficiency of fractal image encoding in existing literature. Few of the prominent research contributions are reported here in terms their findings and limitations.

Xue et al. (1994) proposed an implementation to reduce the encoding complexity by parallel algorithm on the SPHINX machine using multi-SIMD quad pyramid and reduction parameter reduced the complexity from O (n4) to O (n2). This obtained a speedup of O (n2) and gains 25 % in efficiency in encoding procedure [4]. Jackson et al. (1995) introduced a parallel scheme for hypercube multiprocessor to increase speedup of encoding process. The method uses Fishers [3] quad tree model to partition the image into range blocks, then they are compared to the domain blocks. The method solves the load imbalance problem by applying task to the node when they are idle; instead the proposed architecture is not scalable as there is only 64 KB memory available for each processing element [5]. Nge et al. (1997) speeds up fractal encoding procedures by applying dynamic load distribution among 12 computing nodes and gains speedup to 1.5 times which is better than the static load allocation; the encoding process takes 119.86 seconds which makes it slower [6].

Chow et al. (1997) proposed a hardware based pipeline architecture using multiple digital signal processor by utilizing Hurd and Barsney algorithm, the parallel execution of domain search result in 519 seconds. This pipeline architecture after completion of one job the first card will have to pause for a short time while waiting for the 2nd card to finish also it requires an additional file management program that after completion of one job stores the compressed data, which creates overhead [7]. Palazzari et al. (1999) presented a massively parallel processing on SIMD machines that uses High Level Parallelism (HPL) to minimize communication overhead, as in HLP the speed is equal to the number of processors, the architecture has peak power of 100Gflops as there are 2048 processors connected in 3-dimensional toroidal topology which compressed the image in 11.2 seconds, although the system is not cost effective because of a very large number of processors [8]. Cao et al.

(2010) proposed Jacquin [2] fractal coding based parallel algorithm using an OpenMP program model which calculates variance between each range block with all transformed domain blocks, and image sliced are fed into the individual node of multi-core processor hence the speedup result in four times of the sequential algorithm. As the model can only operate on two times the number of cores [9]. Fang et al. (2011) created a client server model using four computing nodes that produce the compression ratio of about 15.944 seconds, while maintaining the PSNR (picture signal to noise ratio) of 33.235 and the speedup seen on two and four computing nodes 6.17 and 9.25 respectively [10].

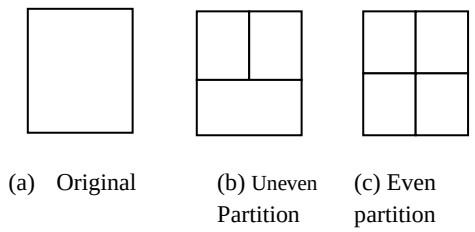
Wakatani et al. (2011) achieved improvement in performance of fractal image coding algorithm using Graphic Processing Unit (GPU) on compute-unified-device architecture (CUDA) and to increase the effectiveness of a parallel program, the index vector method is used. The algorithm, achieves the speedup of about 128.35 on a given GPU but the memory access cost of GPU is high [11]. Wakatani et al. (2012) reduced the memory access cost of the GPU by adding more threads for each thread block, as the result the algorithm achieves speedup of about 162.54 on a given GPU [12].

3. PARALLEL FRACTAL IMAGE COMPRESSION

As we know that the images have certain geometry through which we can compress them this is why fractal image compression technique utilizes this geometry to compress an image. The objective is to compress these types of geometry using parallel computing. The fractal image compression can be defined over the image I, and to find a function $\mathbf{X_i(I) = I}$ that represent the image I in a smaller space, for that I is subsequently divided into two regions, i.e. range RI and domain DI and RI, $\mathbf{D_I \rightarrow R_I}$ such that after the certain affine transformation over the subset DI and contractive mapping Xi between RI and DI yields a smaller space than I, that may represent an original image using lesser bits [13]. The mapping function takes longer time to execute hence parallel approach can solve this problem as mapping of subsets is an individual process, and can be paralleled for every range and domain blocks [14]. Fisher [2] discussed multiple processing elements (PE) that can reduce the encoding time. To parallelize fractal compression it is necessary to find out the sequential and parallel part and feeding parallel part to the parallel machine may result in compression with less encoding time.

3.1 Partitioning

For a given finite image space I, we need to find the partition function p (n) such that it can equally divide I to be processed by the parallel machine. As shown in Figure 1, the partition scheme divides the original image into subsets of equal size or otherwise distribution will be uneven which leads to inconsistency.



(a) Original (b) Uneven Partition (c) Even partition

Fig 1: Partition Scheme

3.2 Load Balancing

For 'n' tasks available to compress, a mechanism is determined which dynamically balances the load among multiple CN's (Computing Node). As shown in Figure 2, the task of unequal size is distributed to three CN's, but the allocation is not equal on all CN's. P1 has two tasks; P2 and P3 have one task equally. This can be seen that after completion of task 2 and task 4, the CN, P2 and P3 will remain idle, which can lead to load imbalance problem.

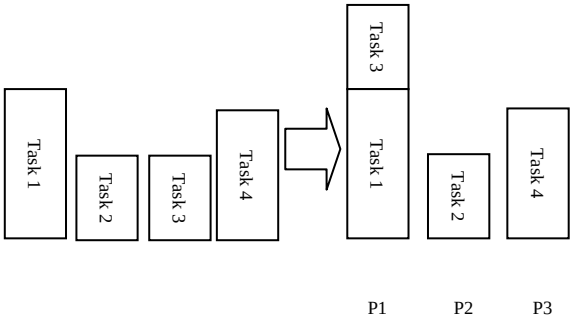


Fig 2: Unequal Load distribution.

3.3 Contractive Mapping

For a given subsets, the domain is geometrically transformed (domain D_I of an image over the Range R_I), which is smaller than the domain blocks and contractive mapping $X_i: D_I \rightarrow R_I$ is applied for each domain blocks. The minimum difference is measured against the corresponding range block at each position, and record those transformations until we get desired similarity as shown in Figure 3, the process of contractive mapping ensures against the corresponding range block at each position, and record those transformations until we get desired similarity as shown in Figure 2, the process of contractive mapping ensures IFS (Iterated Function System) [1] to converge to a fixed point of an IFS of an image.

4. ENCODING CHALLENGES

Parallel processing is used to solve complex problems such as geographical image processing and as the problem size grows, there is a tremendous reduction in computational processing. The primary objective of parallelism is to increase speedup. The major factors diminishing the image encoding speedup are: start up and consolidation cost; communication cost and load imbalance. Serial portion of an operation may affect the overall execution time which leads to lower speedup. Execution is divided into two parts: serial execution and parallel execution. In serial execution, different operation

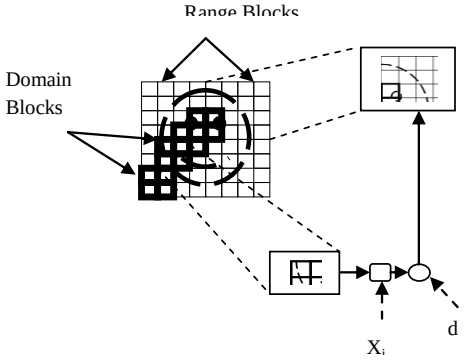


Fig 3: Mapping Procedure.

can be performed for fractal image compression as shown in Figure 4, each operations are separated and executed by multiple CN's. The operation is performed in serial fashion and next operation has to wait until first is completed [15].

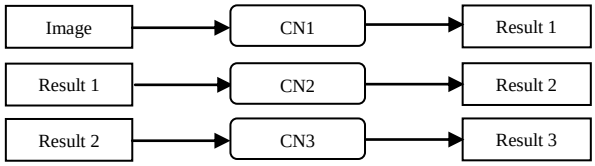


Fig 4: Serial Execution.

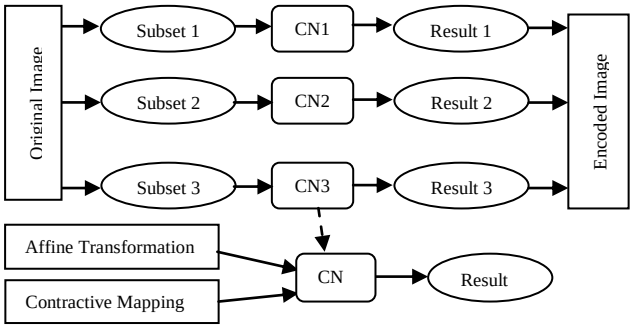


Fig 5: Parallel Execution.

Figure 5 shows the parallel execution in which an image is partitioned into subsets of small defined size images using various partitioning schemes. The subsets are fed into individual CN's, as each CN's performs its operation individually which produces individual results of each subset of an image. The intermediate are consolidated to generate the fractal code. The CN performs an operation of partitioning, affine transformation, contractive mapping and code book generation individually without waiting for other CN [16].

5. PROPOSED WORK

The problem of encoding is solved by partitioning and distributing to multiple CN's. For an image of size N x N, if there are n task available after partitioning the original image, then to solve it using $\propto_n$ processors it would take $\propto_n$. t_n time, where t_n is the time to perform operations. If there are Oi operations to be executed in parallel, then the computation time with $\propto_n$ processors $O_i/\propto_n$. A distributed computing architecture is proposed to reduce the encoding time. For an image I of size N X N, we must find a partition scheme which uniformly subdivide the subsets of I i.e. range $R_I = r$, $(r = N/4)$ and domain block $D_I=R_I/f$ where f is a domain offset $r < f < r$ and $f \neq r$. There are multiple CN's $\propto=1,2,3\ldots.n$, and each performs mapping over a subset R_I and D_I individually, and find a mapping function $X_i(I) = I$ using contractive mapping of a range bock to domain block and achieve a f smaller than the I, the individual result generated will be consolidated into codebook. Total number of range blocks $N_r = (N/4)^2$; total number of domain blocks $N_d = (R_I/f)^2$; total number of Domain to Range comparison $= N_r . N_d$; partitioning $Time = T_{part}$; domain Pool Formation $Time = T_{DP}$; communication $Time = T_{comm}$; mapping $Time = T_{mapp}$; code book Generation $Time = T_{code}$; other overhead $= T_{other}$ and total computation $time = T_{total}$. There are *eight* isometric transformations are required to map domain block to similar range blocks, which made total of $8 \times N_r \times N_d$ domain to range comparison.

$$T_{mapp} = ((8. N_r . N_d). T_x)/ \alpha_n \quad (5.1)$$

$$T_{total} = T_{part} + T_{DP} + T_{comm} + T_{mapp} + T_{code} + T_{other} \quad (5.2)$$

The proposed distributed parallel architecture is shown in Figure 6. The architecture takes original square image partition the image according to the parallel algorithm. The master creates a task and send these to multiple computing node, as each task contains subset of range and domain pool, the CN's performs search for range from the domain pool (DP) and perform mapping operation until perfect matched domain block is obtained and store the transformation only, each CN's perform this operation individually and the result of each node is consolidated.

The algorithm for proposed Parallel Encoding is as follows:

Begin:

Input: I = Read, NxN square grey scale image. (N = 256x256)

Step 1: Image I is divided 16x16 non- overlapping range blocks $[R_I]$, divide image I, $R_I/fxR_I/f$ where (f = 0.25, 0.5, 1, 2, 4) overlapped domain blocks (D_I) and distribute it to all PEs.

Step 2: Perform following for each D_I block on PEs simultaneously.

Step 3: Take a D_j from domain pool.

Step 4: Perform $X_j : D_j \rightarrow R_i$

Step 5: If $Dj \neq Ri$ then Repeat steps 4 to 5 until best match found

End;

6. RESULT AND DISCUSSION

The parallel Fractal image compression algorithm has been implemented on Intel Pentium Dual core CPU, 1.6 GHz, 1GB RAM personal computer as a server and Pentium D-CPU, 2.8GHz, 512MB RAM personal computers as a node using Java RMI (Remote Method Invocation) with Linux platform. The result was evaluated in terms of encoding time, percentage time saving and maximum speed-up by using variable domain size. The result is obtained for encoding of 256 x256 images on 128 CN's. Choosing a fixed range block of size 16×16 and varying domain block size $N_d = R_I /f$ (f= 0.25, 0.5, 1, 2, 4). We obtain the domain blocks 64x64, 32x32, 16x16, 8x8 and 4x4. The performance was compared of sequential execution time over parallel execution using 128 CN's and size of the domain blocks are 64x64, 32x32, 16x16, 8x8 and 4x4. The result can be seen in Figure 7. As we decrease the domain size and increase the CN's, execution time decreases and as shown in Table 1 we have obtained 430.21 sec. using 64 CN which is comparatively lesser than the lager domain block size. The evaluated speedup using varying domain size is shown in Figure 8 which shows maximum speed-up of 25.9 using 64 CN, which gets decreases as we increase the number of nodes. The lower speedup shows that, we distribute the task only up to 64 CN's and beyond that no improvement will be measured. Table 2 and 3 shows how speedup and percentage time saving varies

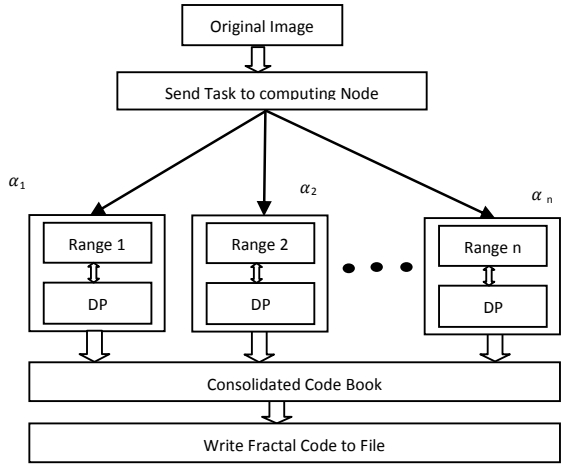


Fig 6: Distributed Parallel Architecture

by decreasing the domain block size and increasing the number of CN's.

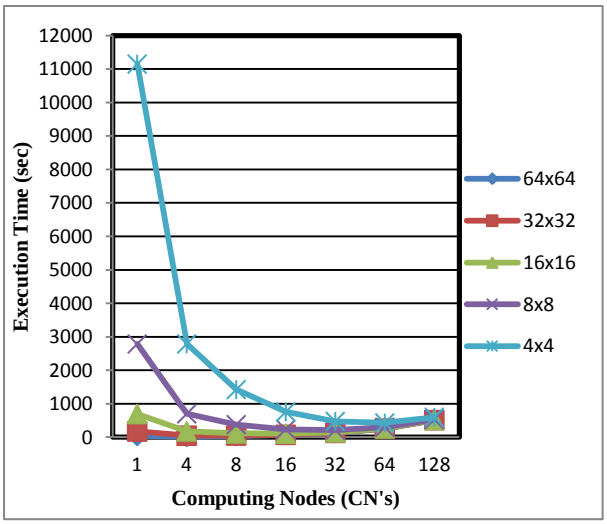


Fig 7: Execution Time

As execution time reduces, the percentage time saving increases and the maximum of 96.60 % time saving is obtained for 64 CN's as shown in Figure 9. It can be observed that for domain size 64x64 and after 8 CN no improvement is noted, also for domain size 32x32 and in 64 and 128 CN's no improvement in percentage time saving is seen.

We have evaluated average PSNR (dB) for variable domain size blocks, and as shown in Figure 10 the PSNR increasing as we decrease the domain size, which shows at smaller domain blocks good quality of image can be constructed.

Table 1: Execution Time

Domain Block Size	64x64	32x32	16x16	8x8	4x4
CN's	Execution Time (Sec.)				
1	43.55	174.21	696.83	2787.33	11149.31
4	14.89	47.55	178.21	700.83	2791.33
8	37.44	53.78	119.10	380.42	1425.66
16	66.72	74.89	107.55	238.21	760.83
32	129.3	133.44	149.78	215.10	476.42
64	256.6	258.72	266.89	299.55	430.21
128	512.3	513.36	517.44	533.78	599.10

Table 2: Speedup

Domain Block Size	64x64	32x32	16x16	8x8	4x4
CN's	Time Saving (%)				
4	65.82	72.70	74.43	74.86	74.96
8	14.02	69.13	82.91	86.35	87.21
16	0.00	57.01	84.57	91.45	93.18
32	0.00	23.40	78.51	92.28	95.73
64	0.00	0.00	61.70	89.25	96.14
128	0.00	0.00	25.74	80.85	94.63

Table 3: Percentage Time Saving

Domain Block Size	64x64	32x32	16x16	8x8	4x4
CN's	Speedup				
4	2.9	3.7	3.9	4.0	4.0
8	1.2	3.2	5.9	7.3	7.8
16	0.7	2.3	6.5	11.7	14.7
32	0.3	1.3	4.7	13.0	23.4
64	0.2	0.7	2.6	9.3	25.9
128	0.1	0.3	1.3	5.2	18.6

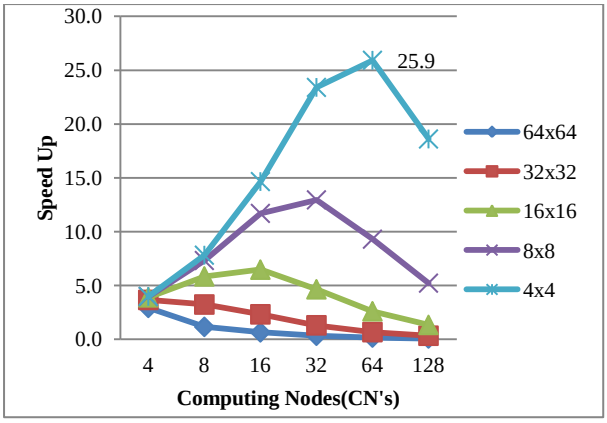


Fig 8: Speedup

7. CONCLUSIONS

Variable domain blocks were used to compare with the fixed size range block. The result shows that if the proposed architecture is implemented on cluster of 64 CN's, it may result significant improvement in performance. The results obtained were compared as against the sequential encoding time and saved 96.60 % time. The speedup shows that the parallel architecture is 25 times better than the sequential scheme. Future work will focus on parallel fractal encoding using distributed parallel architecture along with evaluation of the effectiveness of the parallel algorithm.

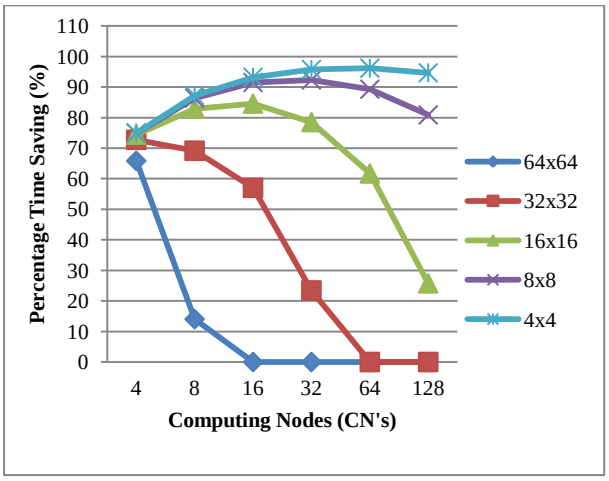


Fig 9: Percentage Time Saving

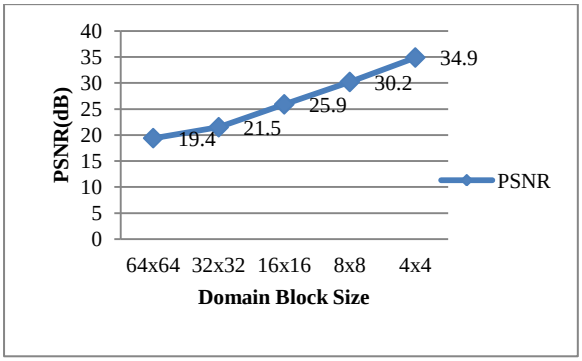


Fig 10: Average PSNR for the domain blocks

8. REFERENCES

[1] M. Barnsley Fractals Everywhere. Academic Press Inc., San Diego, 1988.

[2] A. Jacquin, "Image coding based on a fractal theory of iterated contractive image transformations," IEEE Transaction on Image Processing, Vol.1, No.1 pp. 18-30, January 1992.

[3] Y. Fisher Fractal Image Compression: Theory and Application, Springer Verlag, New York, 1999.

[4] Mitt Xue, Timothy Hansott, and Alain Merigot, "A Massively Parallel Implementation of Fractal Image Compression", proceeding of IEEE international conference on image processing, November 13-16, 1994, Austin, Texas.

[5] Jackson, D.J. and Blom T. "A parallel fractal image compression algorithm for hypercube multiprocessors", proceeding of IEEE 27th International Southeastern Synopsium on System Theory, pp. 274-278, March 12-14, 1995, Starkville, *Mississippi*.

[6] Toh Guan Nge and Wong Kin Keong, "Parallel implementation of fractal image compression", proceedings of IEEE International Symposium on Consumer Electronics, pp. 169-172, 1997.

[7] S.K Chow, M. Gillies and S.L. Chan "Parallel Implementation of Fractal Image Compression Using Multiple Digital Signal Processors" Vol. 1751 of Lecture Notes on Computer Science, pp.714-721. Springer-verlog, 1997.

[8] Paolo Palazzari, Moreno Coli and Guglielmo Lulli, "Massively parallel processing approach to fractal image Compression with near-optimal coefficient quantization", Journal of Systems Architecture: the EUROMICRO Journal - Special issue on parallel image processing (PIP), Vol. 45, No. 10, pp. 765-779, Elsevier , April 1999.

[9] Hua Cao, Xi-jin Gu , "OpenMP Parallelization of Jacquin Fractal Image Encoding" , IEEE International Conference on E-Product E-Service and E-Entertainment (ICEEE) pp.1-4, Nov 7-9, 2010, Henan.

[10] Yan Fang, Hang Cheng, Meiqing Wang, "Parallel Implementation of Fractal Image Compression in Web Service Environment", proceeding of IEEE 10th International Symposium on Distributed Computing and Applications to Business, Engineering and Science (DCABES), pp. 59-64,October 14-17, 2011, Wuxi.

[11] Wakatani A., Mede M. and Tanaka K.," GPGPU implementation of adaptive fractal image coding algorithm using index vectors", IEEE International Conference on Industrial Informatics (INDIN), pp. 316 – 321, July 26-29, 2011, Caparica, Lisbon.

[12] Wakatani A., "Improvement of adaptive fractal image coding on GPUs", IEEE International Conference on Consumer Electronics (ICCE) pp. 255-256, Jan 13-16 2012, Las Vegas, Nevada.

[13] G.R.Sinha, Ravindra Ramteke and Vikas Dilliwar, "Implementation of dimension fractal image segmentation using MATLAB", International Jounal. of Engg. Research & Indu. Appls. (IJERIA), Vol. 2, No. I, pp. 221-226, 2009.

[14] Bhagwati Charan Patel and G.R.Sinha, **"**Early detection of breast cancer using self similar fractal method", International Journal of Computer Application, New York USA, Vol. 10, No. 4, pp. 39-43, November 2010.

[15] Bheshaj Kumar, Kavita Thakur, G. R. Sinha, Bhagwati Charan Patel and Siddhartha Choubey, "Parallel implementation for fast and efficient image compression in spatial domain", 3rd International Conference on Machine Learning and Computing(ICMLC 2011), Vol. 4, pp. 378-381, February 26-28, 2011, Singapore, (IEEE Catalog Number: CFP1127J-PRT, ISBN: 978-1-4244-9252-7).

[16] Bheshaj Kumar, Kavita Thakur and G R Sinha, "A new hybrid JPEG symbol reduction image Compression technique", The International Journal of Multimedia & Its Applications (IJMA) Vol.4, No.3, pp. 81-92, June 2012.