

# Parallel Implementation of Texture based Medical Image Retrieval in Compressed Domain using CUDA

Kuldeep Yadav

Department of CSE,  
College of Engineering Roorkee,  
Roorkee-247667, Uttarakhand, India

Avi Srivastava

Department of CSE,  
College of Engineering  
Roorkee, Roorkee-247667,  
Uttarakhand, India

M.A Ansari

Department of Electrical,  
GBU, Greater Noida,  
Uttar Pradesh, India

## ABSTRACT

In huge databases, Image processing takes more time for execution on a single core processor because of slow single thread algorithms. Graphics Processing Unit (GPU) is more popular now-a-days due to their speed, programmability, low cost and more inbuilt execution cores in it. Most of the researchers started work to use GPUs as a processing unit with a single core computer system to speedup execution of algorithms. The main goal of this research work is to parallelize the process of content based image retrieval through texture and that in compressed domain making whole process much faster than normal. In this paper, parallel implementation is focused on the well known Euclidean Distance approach for texture based image retrieval systems, since it is one of the most fundamental and important problems in the field of computer vision and content based image retrieval (CBIR) and for compressed images we have taken standard JPEG format. Our work employs extensive usage of highly multithreaded architecture of multi-cored GPU. An efficient use of shared memory is required to optimize parallel reduction in Compute Unified Device Architecture (CUDA). Experimental results show that parallel implementation achieved an average speed up of 30 x over the serial implementation when running on a GPU named GeForce 9500 GT having 32 cores. Texture based retrieval method of CBIR is also evaluated using Recall, Precision, F-measure, True Negative rate, and Accuracy evaluation measures.

## General Terms

Content Based Image retrieval, GPU, Parallel Computing.

## Keywords

Texture Based Image Retrieval; CUDA; GPU; Parallelization.

## 1. INTRODUCTION

Graphical Processing Units (GPUs) have been proved its importance in terms of performance as hardware for computer graphics [1]. Many researchers have already been applied GPUs to implement many algorithms in various areas such as image processing, computational geometry, and scientific computation, as well as computer graphics [2-7]. GPUs play important role to speedup processing of database images matching algorithms because it has more inbuilt execution cores. The parallel implementation of image analysis algorithms using GPU encounters two problems. First, the programmer should master of the fundamentals of GPU and CUDA [8]. CUDA platform is used to implement the parallel implementation of algorithms.

Second, in a job it needs much process cooperation between CPU and GPU.

Presented approach of parallelization is based on the first most important phase of Image retrieval process named texture based image retrieval. Texture based image retrieval is used to distinguish a specific image or similar image, from a database of hundreds of images. Image database cannot be stored in uncompressed format because of limitation of space so generally these images of database are in compressed format. There are two types of retrieval process of images from the database: uncompressed domain and compressed domain. Uncompressed domain [9-14] retrieval process first decompresses the image in database for matching while compressed domain methods [15-21] match them in compressed format. The overhead cost of decompression cost too much of clock cycle so compressed format retrieval process has gained considerable amount of attention for research. Compressed domain retrieval process has only shortcoming of increasing data acquisition time and that too is negligible.

Parallel implementations on GPUs have been applied to various numerical problems [22-25] to reduce the computation time without sacrificing the degree of accuracy. Fast CBIR is one of the important problems in the field of computer vision. The decompression of images and their high computation cost are the main drawbacks of slow implementations of uncompressed CBIR systems. Computational cost reduction approaches of CBIR were proposed in [26] by Emmanuel et al. recently.

In the following sections, we present a detailed description of the proposed methodology as well as experimental results that demonstrate the efficiency of the proposed methodology.

## 2. INTRODUCTION TO NVIDIA CUDA ARCHITECTURE

CUDA™ is a general purpose parallel computing architecture introduced by NVIDIA. It contains the CUDA Instruction Set Architecture (ISA) and parallel compute engine in the GPU. The CUDA architecture is programmed using C language, which can then be run with great performance on a CUDA enabled processor [27]. CUDA-enabled GPUs have hundreds of cores that can collectively run thousands of computing threads. Each core has shared resources, including registers and memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus [28]. Thread hierarchy, shared memories and barrier

synchronization are the three key abstractions of CUDA. A kernel can be executed by a one dimensional or two dimensional grids of multiple equally-shaped thread blocks. A thread block is a 3, 2 or 1-dimensional group of threads. Threads within a block can cooperate among themselves by sharing data through some shared memory and synchronizing their execution to coordinate memory accesses. Threads in different blocks cannot cooperate and each block can execute in any order relative to other blocks. The number of threads per block is therefore restricted by the limited memory resources of a processor core.

CUDA kernel function is a fundamental building block of CUDA programs. When launching a CUDA kernel function, a developer specifies how many copies of it to run. We call each of these copies a task. Because of the hardware support of the GPU, each of these tasks can be small, and the developer can queue hundreds of thousands of them for execution at once. These tasks are organized in a two-level hierarchy, block and grid. Small sets of tightly coupled tasks are grouped into blocks. In a given execution of a CUDA kernel function, all blocks contain the same number of tasks. The tasks in a block run concurrently and can easily communicate with each other, which enables useful optimizations such as those of the section "Shared Memory". GPU's hardware keeps multiple blocks in flight at once, with no guarantees about their relative execution order. As a result, synchronization between blocks is difficult. The set of all blocks run during the execution of a CUDA kernel function is called a grid.

### 3. TEXTURE BASED SIMILARITY FUNCTION IN JPEG

We used a method called the pyramid-structured wavelet transform for texture classification. Its name comes from the fact that it recursively decomposes sub signals in the low frequency channels. It is mostly significant for textures with dominant frequency channels. For this reason, it is mostly suitable for signals consisting of components with information concentrated in lower frequency channels.

Using the pyramid-structured wavelet transform [11], the texture image is decomposed into four sub images, in low-low, low-high, high-low and high-high sub-bands. At this point, the energy level of each sub-band is calculated using eq. 1. This is first level decomposition. As in our case we have taken image of size 512x512 giving DC coefficients of size 64x64 and minimum size for sub bands can be 16x16 so we apply decomposition procedure two times and energy level of the sub bands are calculated. The energy level values are stored for using in Euclidean distance algorithm.

$$E = \frac{1}{MN} \sum_{i=1}^M \sum_{j=1}^N |x(m, n)| \quad (1)$$

After calculation of energy level, difference between values of query and database images are found and arrange in decreasing order using Euclidean distance algorithm defined in eq. 2.

$$d^2(x, y) = \sum_{i=1}^k (x_i - y_i)^2 \quad (2)$$

## 4. EVALUATION MEASURES

The method of texture based image retrieval is evaluated using the six evaluation measures: Precision, Recall, F-measure, True negative rate, (Negative Rate Metric) NRM and accuracy.

- Precision:

$$\text{Precision} = \frac{TP}{TP + FP} \quad (3)$$

- Recall:

$$\text{Recall} = \frac{TP}{TP + FN} \quad (4)$$

- F-Measure:

$$F - \text{Measure} = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (5)$$

- True Negative Rate:

$$\text{True Negative Rate} = \frac{TN}{TN + FP} \quad (6)$$

- Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (7)$$

- NRM:

$$NRM = \frac{NR_{FN} + NR_{FP}}{2} \quad (8)$$

$$\text{Where } NR_{FN} = \frac{FN}{FN + TP} \text{ and } NR_{FP} = \frac{FP}{FP + TN}$$

## 5. IMPLEMENTATION

In this work, the implementation of proposed approach is based on the two set of experiments. In the first set of experiment, proposed algorithm is implemented in C language and in second set, parallel implementation is done using CUDA. The following section dictates the detailed description of the parallel implementation of the algorithm.

### 5.1 Parallel Implementation

In CUDA, it is assumed that both host and device maintain their own DRAM. Host memory is allocated using malloc and device memory is allocated using cudaMalloc. CUDA threads are assigned a unique thread ID that identifies its location within the thread, block and grid. This provides a natural way to invoke computation across the image, by using the thread IDs for

addressing. The parallel implementation of algorithm of CBIR is shown in the form of pseudo code [11] shown in algorithm 1.

Algorithm 1: Parallel Implementation of Texture based image retrieval's Energy level calculation algorithm

Step1. Decompose image in four sub-bands.  
Step2. Parallely calculate Energy (eq. 1) for each band using four threads.  
Step3. Select least energy sub-band and apply step2 one more time.

Algorithm 2: Parallel Implementation of Texture based image retrieval's Euclidean distance calculation algorithm

Step1. Select energy set of query image.  
Step 2. Select one energy set of image in database.  
Step2. Parallely calculate Euclidean distance (eq. 2) and save it in Euclidean vector.  
Step3. If more image in database  
    Goto step 2  
    Else  
        Goto step 4  
Step4. Arrange Euclidean vector in decreasing order of its magnitude.

## 6. HARDWARE SPECIFICATIONS

All the experiments are carried out using the hardware specifications of GPU: GeForce 9500 GT, 1 MB DDR2, No of Processors = 4, No of core =32, RAM 1 GB, Frequency 1.35 GHz, DDR2 and CPU: Intel Core 2 Duo, 2.66 GHZ, No of cores available =2, No of thread=1, No of thread/core=1, Physical Memory =2 GB, DDR2

## 7. RESULTS AND DISCUSSIONS

For the testing of texture based retrieval approach of CBIR, we collected a data set of MRI, CT-scan and X-ray to form database of images in compressed format of JPEG. The results of texture based retrieval approach are shown in fig. 4 that demonstrates the efficiency of this approach. On the basis of visual observation, texture based retrieval method of CBIR manages to find images similar to query image in database but with a drawback of a lot of time consumption. To make faster the method, we parallelized it on CUDA and achieved an average speed up of 30 x(approx) over the serial implementation when running on a GPU. The comparison of serial implementation over parallel is shown in table 1. Table 1 also shows that execution time depends on the image resolution.

Further, the performance of method is evaluated using Precision, Recall, F-measure, True Negative Rate, NRM and Accuracy

measures, which show the effectiveness of method shown in table 2. Fig.2 shows the graph of execution time of GPU in seconds. Fig.3 shows the graph of speedup. Fig. 1 shows the graph of execution time of CPU in seconds. Output images of Texture based retrieval approach is shown in fig. 4

Table 1: Execution time serial over parallel implementation

Resolution (a X a)	Serial	Parallel	Speed-Up	Speed-Up Average
512	5.323	0.186559	28.532424	27.939524
256	3.264	0.119356	27.346625	
512	4.756	0.153482	30.987394	29.559863
256	3.001	0.106674	28.132333	
512	4.978	0.175967	28.289322	28.371278
256	2.954	0.103819	28.453234	
512	5.121	0.175989	29.098344	29.000237
256	3.442	0.119091	28.902130	
512	5.332	0.180891	29.476376	28.859399
256	3.991	0.141312	28.242423	
Average Speed-Up				28.746060

Table 2: Evaluation Measures

Image	Precision	Recall	F-Measure	TNR	NMR	Accuracy
1	75	75	75	75	25	75
2	75	75	75	75	25	75
3	75	75	75	75	25	75
4	50	50	50	50	50	50
5	100	100	100	50	00	100

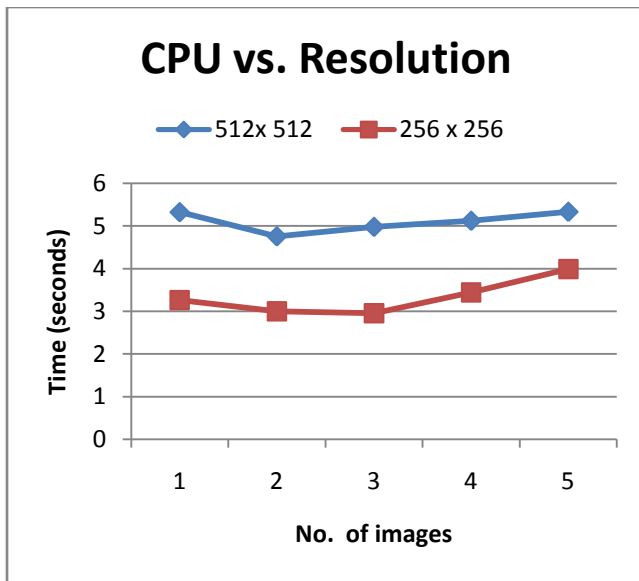


Fig1. Execution time in CPU vs. Resolution

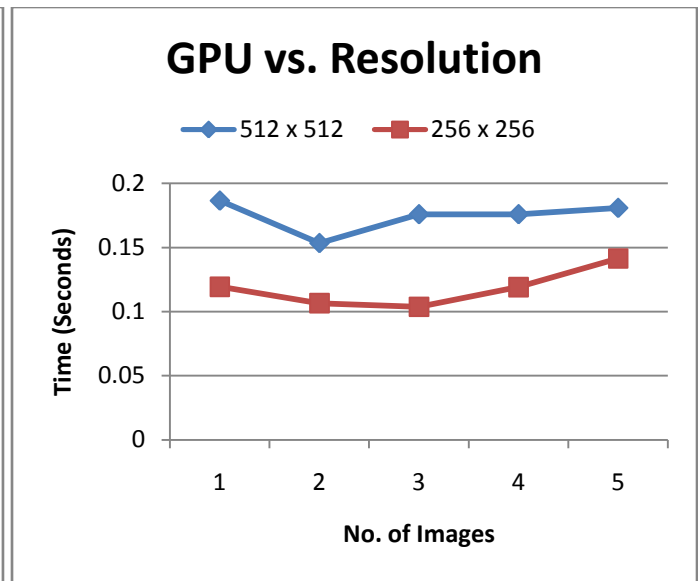


Fig2. Execution time in GPU vs. Resolution

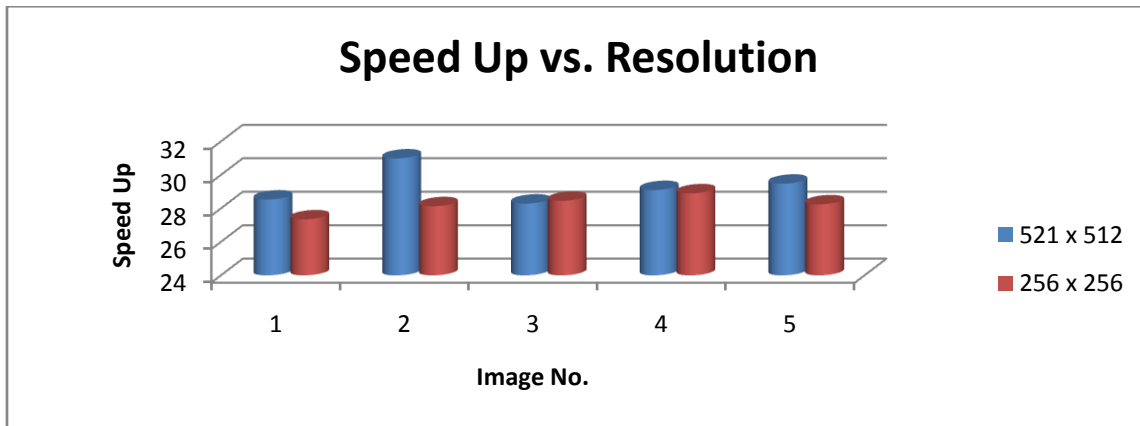
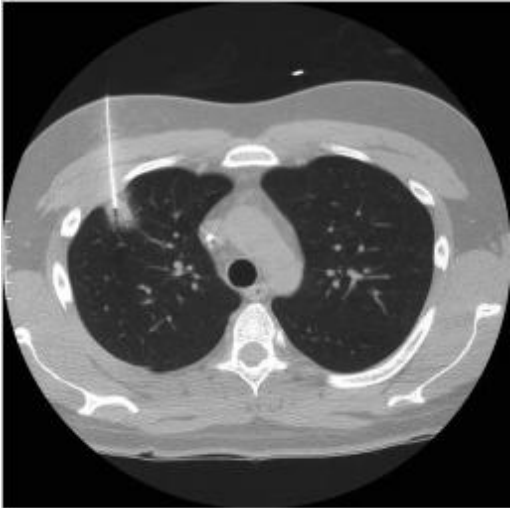



Fig3. Speed up vs. Resolution

Image Number	Query Image	CBIR results
1.		

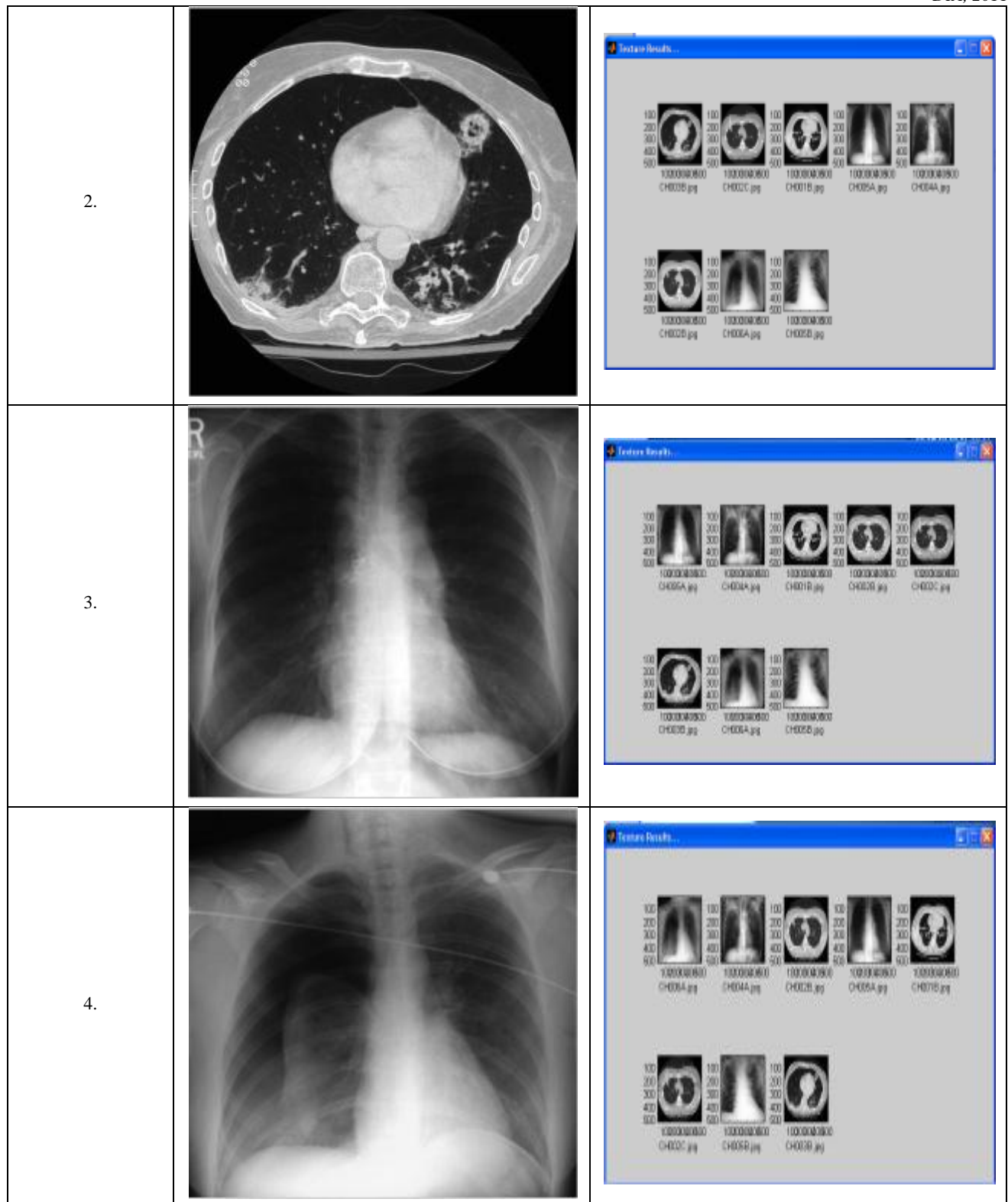


Fig. 4: Output images of CBIR

## 8. CONCLUSION

In this research work, a well known texture based image retrieval algorithm of CBIR has been parallelized and analyzed with evaluation measures. The method is evaluated using

Precision, Recall, F-measure, True Negative Rate, NRM and Accuracy measures. The implementation of CBIR algorithm on the graphics device is promising with large image database. However, texture based retrieval method produces images which

are not similar in vision but they are not in top three results which is considerable when compared with speed up of approx 30x.

CUDA itself has been shown to be an excellent framework to accelerate computational problems in image processing, numerical solving techniques and Image Processing areas.

## 9. REFERENCES

- [1] Fernando, R and Kilgard, M. J. The Cg tutorial the definitive guide to programmable real-time graphics. Addison-Wesley, 2003.
- [2] Moravanszky, Linear algebra on the GPU, in: W.F. Engel (Ed.), Shader X 2, Wordware Publishing, Texas, 2003.
- [3] Manocha, Interactive geometric & scientific computations using graphics hardware, SIGGRAPH 2003.
- [4] Moreland, K. and Angel E. "The FFT on a GPU". In Proceedings of SIGGRAPH Conference on Graphics Hardware, 112-119, 2003.
- [5] Mairal, J., Keriven, R. and Chariot, A. "Fast and efficient dense variational Stereo on GPU". In Proceedings of International Symposium on 3D Data Processing, Visualization, and Transmission, 97-704, 2006.
- [6] Yang, R. and Welch, G. "Fast image segmentation and smoothing using commodity graphics hardware". Journal of Graphics Tools, Vol. 17, (4), 91-100, 2002.
- [7] Fung, J. and Man, "OpenVIDIA: Parallel GPU computer vision". In Proceedings of ACM International Conference on Multimedia, 849-852, 2005.
- [8] Jang, H., Park, A. and Jung, K. "Neural network implementation using CUDA and OpenMP". In Proceeding of Computing: Techniques and Applications, (DICTA), IEEE, 155 – 161, 2008.
- [9] Th. Gevers. "Image segmentation and matching of color-texture objects". IEEE Trans. on Multimedia, 4(4), 2002.
- [10] R. Jain, R. Kasturi, and B. G. Schunck, Machine Vision, McGraw Hill International Editions, 1995.
- [11] Rami Al-Tayeche and Ahmed Khalil, "CBIR: Content Based Image Retrieval" Department of Systems and Computer Engineering Faculty of Engineering Carleton University" Tech. Rep. April 4, 2003.
- [12] Hemant d. Tagare, c. Carl jaffe, james duncan," Medical Image Database Retrieval", 1/21/97.
- [13] Grosky WI. "Iconic Indexing Using Generalized Pattern Matching Techniques". Computer Vision, Graphics, and ImageProcessing, 1986. 35:383–403.
- [14] Chang SK. "Picture Indexing and Abstraction Techniques for Pictorial Databases". IEEE Transactions on Pattern Analysis and Machine Intelligence, 1984. 6(4).
- [15] Padmashri Suresh ,RMD.Sundaram Aravindhan Arumugam, "Feature Extraction in Compressed Domain for Content Based Image Retrieval", International Conference on Advanced Computer Theory and Engineering, 10.11.09
- [16] M. Hatzigiorgaki and A. N. Skodras, "Compressed Domain Image Retrieval: A Comparative Study of Similarity Metrics", Visual Communications and Image Processing 2003, Touradj Ebrahimi, Thomas Sikora, Editors, Proceedings of SPIE Vol. 5150 (2003).
- [17] B. Furht, P. Saksobhavitvat, "Fast Content-Based Multimedia Retrieval Technique Using Compressed Data," Proc. SPIE Vol. 3527, pp. 561-571, 1998
- [18] W.B. Pennebaker and J.L. Mitchell, "JPEG Still Image Data Compression Standard," Van Nostrand Reinhold, NY, 1993.
- [19] B.S. Manjunath, J.-R. Ohm, V.V. Vasudevan, and A. Yamada, "Color and Texture Descriptors," IEEE Trans. Circuits and Systems for Video Technology, Vol. 11, No. 6, pp.703-715, June 2001.
- [20] V. Castelli and L.D. Bergman (Editors), Image Databases: Search and Retrieval of Digital Imagery, J. Wiley & Sons, NY, 2002
- [21] Chen, J.Y., Bouman, C.A., and Allebach, J.P., "Fast image database search using tree structured VQ," Proc. Int. Conf. on Image Processing, USA, Vol.2, pp. 827-830, October 1997.
- [22] Owens, J. D. Luebke, D., Govindaraju, N., Harris, M., Kruger, J., Lefohn, A. E. and Purcell, T. J. "A survey of general-purpose computation on graphics hardware". In proceeding of Eurographics, State of the Art Reports, 21–51, 2005.
- [23] Larsen, E. S., McAllister, D. "Fast Matrix Multiplies using Graphics Hardware". In Proceeding of International Conference for High Performance Computing and Communications, 159-168, 2001.
- [24] Trendall C. and Stewart, A. J. "General calculations using graphics hardware with applications to interactive caustics". Rendering Techniques 2000: 11th Eurographics Workshop on Rendering, 287-298, 2000.
- [25] Li, Wei, Wei, Xiaoming, A. and Kaufman, "Implementing lattice boltzmann computation on graphics hardware". In proceeding of the International Conference for High Performance Computing and Communications , 2001.
- [26] M. Emmanuel, D.R. Ramesh Babu, Jayashree Jagdale, Pravin Game and G.P. Potdar, "Parallel Approach for Content Based Medical Image Retrieval System", Journal of Computer Science 6 (11): 1258-1262, 2010.
- [27] NVIDIA CUDA Programming Guide Version 2.0, available at [www.nvidia.com/object/cuda\\_develop.html](http://www.nvidia.com/object/cuda_develop.html).
- [28] NVIDIA Corporation: NVIDIA CUDA programming guide. Jan 2007, available at [http://developer.download.nvidia.com/compute/cuda/2\\_0/docs/NVIDIA\\_CUDA\\_Programming\\_Guide\\_2.0.pdf](http://developer.download.nvidia.com/compute/cuda/2_0/docs/NVIDIA_CUDA_Programming_Guide_2.0.pdf)