A New Architecture for Content based Image Retrieval with Graphical Processing Unit

BommarahyNarsing
Asst Professor
Department of ECE
Scient institute of engg and technology,Hyderabad,Telangana ,India

Abstract

CBIR is the method of searching the digital images from an image database. "Content-based" means that the search analyzes the contents of the image rather than the metadata such as colours, shapes, textures, or any other information that can be derived from the image itself. The GPU is a powerful graphics engine and a highly parallel programmable processor having better efficiency and high speed that overshadows CPU. It is used in high performance computing system. The implementation of GPU can be done with CUDA C. Due to its highly parallel structure it is used in a number of real time applications like image processing, computational fluid mechanics, medical imaging etc. Graphical Processors Units (GPU) is more common in most image processing applications due to multithread execution of algorithms, programmability and low cost. In this paper, we are explaining the parallel implementation of CBIR with GPU. We have shown various stages of CBIR with GPU results into better performance as well as speed ups. We have given a review of various techniques that can be practised for high performance CBIR stages with Graphics Processing Units.

I. INTRODUCTION

GPU is Graphics Processing Unit used for high performance parallel computing. It is a processor that over the past few years has evolved from a fixed-function special-purpose processor into a full-fledged parallel programmable processor with additional fixed-function special-purpose functionality. More than ever, the programmable aspects of the processor have taken centre stage [1]. GPU computing is the use of GPU together with CPU (Central Processing Unit) to accelerate general-purpose scientific and engineering applications. The GPU is a processor with ample computational resources. GPU computing is the use of a GPU together with a CPU to accelerate general-purpose scientific and engineering applications [1]. Fig.1 shows the architecture of GPU where CPU sends tasks and data to GPU whereas GPU performs computations on data and sends back results to CPU. The CPU controls all the computations. So, in context of GPU computing:

• GPU code is called DEVICE code.

• CPU code is called HOST code

The reason why the variance of the image scores became small is in its calculation process. In the image processor, the gender score of a user is calculated as the mean of the highest object scores extracted from each image. Figure 5 shows a distribution of "male-person" label scores. Though a distribution of each object probability scores centres not at 0.5, highest score selections and the averaging of them leads to a mid-range value, in this case 0.5.

Our intuition behind the introduction of α was to provide a reliability ratio parameter of the text processor and the image processor. But as a matter of fact, this parameter also worked to calibrate the scale difference between the two probability scores. From this observation, a function that includes a reliability parameter and a calibration parameter separately can be considered as an alternative to the proposed function. Using this kind of function will provide further insights about combining a text processing and image processing.

We compared two consolidation methods, computing the average of all scores and computing the average of the highest scores for 30 object scores. We applied the two method to the training data of the user annotation data, and tested them on the evaluation data. Results show that the accuracy of former method is 60.11. That of the latter is 65.42. The reason the latter method is superior to the former one is probably attributable to noise reduction effects of ignoring low scores Features can be extracted in parallel from the images with GPU using various techniques. Feature extraction can be performed by using GPU based KLT feature tracker and GPU based SIFT feature extraction algorithm. The KLT tracking algorithm computes displacement of features or interest points between consecutive video frames when the image brightness constancy constraint is satisfied and image motion is fairly small[12]. In its parallel implementation the GPU distributes its various steps in fragment programs

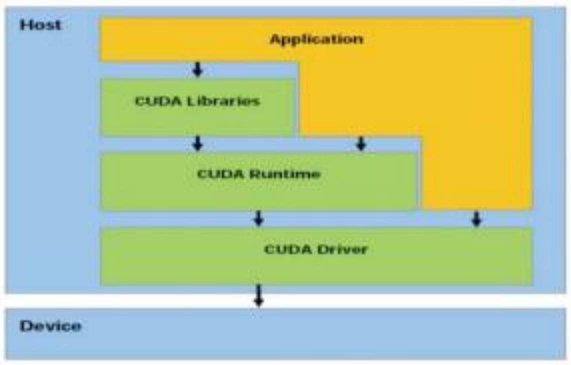

Fig.1: GPU Architecture

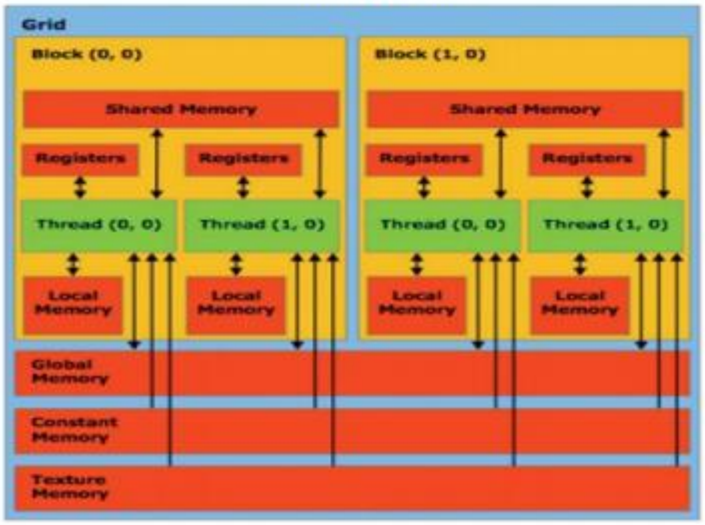

Fig1:Cudu architecture

To estimate the optimal value of α, we conducted a preliminary experiment of the combined method with training data. We first prepared text and image probability scores. The text score is obtained by executing five-fold cross validation of the text processor for training data. We used the probability score derived in section 5.1 as the image score. The accuracies were, respectively, 86.23 and 65.42. Next, the combined formula was applied to these probability scores with moving α from 0 to 1. Figure 3 shows the correlation between accuracy and α. To obtain the α value of the peak, we executed polynomial fitting to a part of the correlation curve where α is 0.1–0.4. By differentiating this function, we calculated the α value of the peak as equal to 0.244 indicated by the arrow in Figure 3. The accuracy reaches 86.73% at the peak, which is 0.50 pt higher than that of the text processor.

the screen and further assembled to develop a complete image or a picture [1].The Gpu graphics pipeline undergoes various steps like vertex operations, Primitive assembly, Rasterization, Fragment operations and Composition [1].From this pipeline structure, Gpu have become further programmable. Purpose GPU is quite suitable for computing intensive data parallel. In Jun. 2007 NVIDIA released CUDA and in Dec 2008, Khronos Group released OpenCL1. In Aug. 2009, AMD launched ATI Stream SDK v2.0 Beta which supported X86 processor. Open CL is an open standard for many processors [2]. Modern GPUs contain hundreds of processing units, capable of achieving up to 1 TFLOPS for single-precision (SP) arithmetic, and over 80 GFLOPS for double-precision (DP) calculations .Recent High Performance Computing (HPC) optimized GPUs contain up to 4GB of on board memory and 100GB/sec [3].

Due to its parallel architecture and high performance of floating point and memory operations GPU is well suited for many same scientific and engineering applications that occupy HPC clusters, leading to their incorporation as HPC accelerator[3].So, they can reduce space, power, and cooling demands, and reduce the number of operating system images that must be managed relative to traditional CPU-only clusters of similar aggregate computational capability [3].GPU has its amazing computational capabilities and functionalities that extends its applications to the field of non-graphics computations and such a type of GPU is known as General Purpose GPU [4] Due to their cost performance and evolution speed they are becoming significant[4].

A.CUDA

CUDA is Compute Unified Design Architecture. It was introduced by NVIDIA in November 2006. It allows software developers to access GPUs through standard programming language, ‘C. For CUDA (C with NVIDIA extensions and certain restrictions).A compiled CUDA program can execute on any number of processor cores. CUDA C is the language specifically designed to provide general purpose computing on GPU.CUDA C adds the global qualifier to standard C. This mechanism alerts the compiler that a function should be compiled to run on a device instead of the host. In this simple example, nvcc gives the function kernel ( ) to the compiler that handles device code, and it feeds main ( ) to the host compiler. It provides this language integration so that device function calls look very much like host function calls [5].

Fig.2 shows the architecture of CUDA which is composed of Grids and Blocks. In the grid there are multiple threads of the main kernel running parallel and each grid composes a number of blocks that contains the threads and the shared memory [5].The NVIDIA CUDA technology (developer.download.nvidia.com) is a fundamentally new computing architecture that enables the GPU to solve complex computational problems.

CONTENT BASED IMAGE RETRIEVAL AND GPU

Content Based Image Retrieval is based upon retrieving an image by comparing a query image with all the images in the database. Fig.3 shows the certain steps that are followed to fetch an image from the database. Graphical Processors Units (GPU) is more common in most image processing applications due to multithread execution of algorithms, programmability and low cost. CBIR along with GPU finds its applications in medical field [7].Medical imaging produces great amounts of data which, once investigated and classified, might be extremely valuable for future diagnoses. The image retrieval in medical applications (IRMA, (irma-project.org) approach aims at providing a framework for medical CBIR applications including interfaces to PACS and hospital information systems (HIS) [6] where IRMA exactly addresses the Kilo- to Terabyte sized datasets challenge in medical image management and data mining. The main disadvantage of working with Giga- to Terabyte volume data is the runtime performance. This can be achieved by two methods [7]

FEATURE EXTRACTION WITH GPU

In content-based image retrieval, images are automatically stored at index in feature database which describe the content of the image. The features are extracted and stored as feature vectors. The feature vectors of both the query image and the image database are compared and thus the required image is retrieved. Features may be colour, shape, contour, etc. [10] Q be the query image, T be the image database and t be the threshold distance. So, the distance between both the feature vectors is given as,

D (Feature (Q), Feature (T)) $\leq t$........ (1)

Where frame performs tracking using the image pyramids of multi-resolution and intensity. SIFT is scale invariant feature transform algorithm [11] which performs extraction of interest points invariant to translation, rotation, scaling and illumination changes in images. It constructs a Gaussian scale-space pyramid from the input image and also calculates the gradients and difference-of-Gaussian (DOG) images at these scales. Interest points are detected at the local extremes within the DOG scale space. With GPU, the construction of the Gaussian scale space pyramid is accelerated by using fragment programs for separable Gaussian convolution. These implementations are 1020 times faster than the corresponding optimized CPU counterparts and enable real-time processing of high resolution video [12]. We can also use the improved version of parallel SIFT algorithm that provides better performance on multi core platforms [13] and takes care of following:

1. Load Balancing.
2. Reducing Synchronization Overhead.
3. Removing False Sharing.
4. Applying Thread Affinity.

GPU-based KLT implementation tracks about a thousand features in real-time at 30 Hz on 1024x768 resolution video which is a 20 times improvement over the CPU. It works on both ATI and NVIDIA graphics cards. The GPU-based SIFT implementation works on NVIDIA cards and extracts about 800 features from 640x480 video at 10Hz which is approximately10 times faster than an optimized CPU implementation [14]. There is another application of feature detection where eye blink detector works on very low contrast images acquired under near-infrared illumination with GPU[15].Eye blinks are detected inside regions of interest that are aligned with the subjects eyes at initialization. Alignment is maintained through time by tracking SIFT feature points that are used to estimate the affine transformation between the initial face pose and the pose in subsequent frames. Eye blink detection obviously implies prior detection of the eyes in the image of the subjects face. Here also GPU based implementation of SIFT is used for tracking as provided in the library Open NVIDIA [16] openvidia.sourceforge.net.

INDEXING WITH GPU

In Content-based Image Retrieval (CBIR) systems, accurately ranking images is of great relevance, since users are interested in the returned images placed at the first positions, which usually are the most relevant ones. In general, CBIR systems consider only pair wise image analysis, that is, compute similarity measures considering only pairs of images, ignoring the rich information encoded in the relations among several images. On the other hand, the user perception usually considers the query specification and responses in a given context.

In CBIR the similar images are collected by considering its visual properties and ranked in decreasing order of similarity, according to a given image descriptor. An image content descriptor is characterized by [38]:
1. An extraction algorithm that encodes image features into feature vectors;
2. Similarity measure used to compare two images

The similarity between two images is computed as a function of the distance of their feature vectors. In CBIR , the relationships among images, encoded in ranked lists and distances among images, can be used for extracting contextual information[38].There is a ranking method where ranking aggregation is done which uses the reranking methods to combine CBIR descriptors. The post-processing of the distance/ similarity scores, by taking into account the contextual information available in relationships among images in a given collection. The methods require no user intervention, training or labelled data, operating on an absolutely unsupervised way. After that the outputs of re-ranking and rank aggregation methods, are combined aiming at further improving the effectiveness of CBIR results. Finally, for efficient performance the ranking approach is implemented on GPU [38].

CONCLUSION

We have given a review of different techniques for various stages in CBIR that can be performed along with the Gpus.GPU enhance the speed as well as the performance of the method adopted for image retrieval. Graphical Processors Units (GPU) is more common in most image processing applications due to multithread execution of algorithms, programmability and low cost

REFERENCES

[1] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, J. C. Phillips, “Gpu computing”, Proceedings of the IEEE 96 (5) (2008) 879–899,2008.

[2] H. Zhu, Y. Cao, Z. Zhou, M. Gong, “Parallel multi-temporal remote sensing image change detection on gpu”, in: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, IEEE, pp. 1898–1904, 2012.

[3] V. V. Kindratenko, J. J. Enos, G. Shi, M. T. Showerman, G. W. Arnold, J. E. Stone, J. C. Phillips,W.-m. Hwu, “Gpu clusters for high-performance computing”, in: Cluster Computing and Workshops, 2009. CLUSTER’09.IEEE International Conference on, IEEE, pp. 1–8, 2009.

[4] H. Takizawa, H. Kobayashi, “ Hierarchical parallel processing of large scale data clustering on a pc cluster with gpu co-processing”, The Journal of Supercomputing 36 (3) 219–234, 2006.

[5] J. Sanders, E. Kandrot, “CUDA by example: an introduction to general purpose GPU programming”, AddisonWesley Professional, 2010.

[6] M. O. Guld, C. Thies, B. Fischer, T. M. Lehmann, “A generic concept for the implementation of medical image retrieval systems”, international journal of medical informatics 76 (2) 252–259, 2007. 372

[7] I. Scholl, T. Aach, T. M. Deserno, T. Kuhlen, “Challenges of medical image processing”, Computer scienceResearch and development 26 (1-2)5–13, 2011.

[8] M. Strengert, M. Magall´on, D. Weiskopf, S. Guthe, T. Ertl, “Hierarchical visualization and compression of large volume datasets using gpu clusters”, in: Proceedings of the 5th Eurographics conference on Parallel Graphics and Visualization, Eurographics Association, pp. 41–48,2004.

[9] P. V. Coveney, “Scientific grid computing”, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 363 (1833) (2005) 1707–1713, 2005.

[10] R. S. Choras, “Image feature extraction techniques and their applications for cbir and biometrics systems”, international journal of biology and biomedical engineering 1 (1) 6–16, 2007.

[11] D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, International journal of computer vision 60 (2) 91–110, 2004.

[12] S. N. Sinha, J.-M.Frahm, M. Pollefeys, Y. Genc, “Feature tracking and matching in video using programmable graphics hardware”, Machine Vision and Applications 22 (1) 207–217, 2011.

[13] Q. Zhang, Y. Chen, Y. Zhang, Y. Xu, “Sift implementation and optimization for multi-core systems”, in: Parallel and Distributed Processing, 2008. IPDPS 2008.IEEE International Symposium on, IEEE, pp. 1–8, 2008.

[14] S. N. Sinha, J.-M.Frahm, M. Pollefeys, Y. Genc, “Gpu-based video feature tracking and matching”, in: EDGE,

Workshop on Edge Computing Using New Commodity Architectures, Vol. 278, p. 4321, 2006.

[15] M. Lalonde, D. Byrns, L. Gagnon, N. Teasdale, D. Laurendeau, "Real-time eye blink detection with gpu-based sift tracking", in: Computer and Robot Vision, CRV'07. Fourth Canadian Conference on, IEEE, 2007, pp. 481–487, 2007.

[16] J. Fung, S. Mann, "Openvidia: parallel gpu computer vision", in: Proceedings of the 13th annual ACM international conference on Multimedia, ACM, pp. 849–852, 2005.

[17] Y. Freund, R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting", in: Computational learning theory, Springer, pp. 23–37,1995

[18] H. Ghorayeb, B. Steux, C. Laurgeau, "Boosted algorithms for visual object detection on graphics processing units", in: Computer Vision–ACCV 2006, Springer, pp. 254–263,2006.

[19] Y. Poornima, I. Salem, P. Hiremath, I. Gulbarga, "Scalable learning based framework for pruning cbir system using visual art image" , International Journal of Engineering 2 (8).

[20] S. Heymann, K. Muller, A. Smolic, B. Frohlich, T. Wiegand, "Sift implementation and optimization for generalpurposegpu", in: Proceedings of the international conference in Central Europe on computer graphics, visualization and computer vision, 7, p. 144,2007.

[21] J. Lokoˇc, T. Groˇsup, T. Skopal, "Image exploration using online feature extraction and reranking", in: Proceedings of the 2nd ACM International Conference on Multimedia Retrieval, ACM, p. 66, 2012.

[22] D. Ramona, "Matlab medical images classification on graphics processors".

[23] H. Bay, T. Tuytelaars, L. Van Gool, "Surf: Speeded up robust features", in: Computer Vision–ECCV 2006, Springer, pp. 404–417, 2006.

[24] F. Schweiger, G. Schroth, R. Huitl, Y. Latif, E. Steinbach, "Speeded-up surf: Design of an efficient multiscale feature detector".

[25] A. Shahbahrami, T. A. Pham, K. Bertels, "Parallel implementation of gray level co-occurrence matrices and haralick texture features on cell architecture", The Journal of Supercomputing 59 (3) 1455–1477, 2012.

[26] H. Heidari, A. Chalechale, A. A. Mohammadabadi, "Parallel implementation of color based image retrieval using cuda on the gpu", International Journal of Information Technology and Computer Science (IJITCS) 6 (1) 33,2013.

[27] H. Jang, A. Park, K. Jung, "Neural network implementation usingcuda and openmp", in: Computing: Techniques and Applications, 2008. DICTA'08. Digital Image, IEEE, pp. 155–16, 2008.

[28] P. Piro, S. Anthoine, E. Debreuve, M. Barlaud, "Sparse multiscale patches for image processing", in: Emerging Trends in Visual Computing, Springer, pp. 284–304, 2009.

[29] A. Chariot, R. Keriven, "Gpu-boosted online image matching", in: Pattern Recognition, 2008. ICPR 2008.19th International Conference on, IEEE, pp. 1–4, 2008.

[30] C. Zhang, H. Li, Q. Guo, J. Jia, I.-F.Shen, "Fast active tabu search and its application to image retrieval", in: IJCAI, Vol. 9,pp. 1333–1338, 2009.

[31] K. Yadav, A. Mittal, M. Ansari, V. Vishwarup, "Parallel implementation of similarity measures on gpu architecture using cuda", Indian Journal of Computer Science and Engineering 3.

[32] I. K. Park, N. Singhal, M. H. Lee, S. Cho, C. W. Kim, "Design and performance evaluation of image processing algorithms on gpus", Parallel and Distributed Systems, IEEE Transactions on 22 (1) 91–104,2011.

[33] V. Garcia, E. Debreuve, M. Barlaud, "Fast k nearest neighbor search using gpu", in: Computer Vision and Pattern Recognition Workshops, 2008. CVPRW'08.IEEE Computer Society Conference on, IEEE, pp. 1–6, 2008.

[34] R. J. Barrientos, J. I. G´omez, C. Tenllado, M. P. Matias, M. Marin, “knn query processing in metric spaces using gpus”, in: Euro-Par 2011 Parallel Processing, Springer, pp. 380–39 2011.