ASIAN JOURNAL OF PHARMACEUTICAL AND CLINICAL RESEARCH

INNOVARE ACADEMIC SCIENCES
Knowledge to Innovation

Special Issue (April)

Online - 2455-3891
Print - 0974-2441

Full Proceeding Paper

CENTRAL PROCESSING UNIT-GRAPHICS PROCESSING UNIT COMPUTING SCHEME FOR MULTI-OBJECT TRACKING IN SURVEILLANCE

ANKUSH RAI*, JAGADEESH KANNAN R

School of Computing Science and Engineering, VIT University, Chennai, Tamil Nadu, India. Email: ankushressci@gmail.com

Received: 13 December 2016, Revised and Accepted: 03 April 2017

ABSTRACT

This research work presents a novel central processing unit-graphics processing unit (CPU-GPU) computing scheme for multiple object tracking during a surveillance operation. This facilitates nonlinear computational jobs to avail completion of computation in minimal processing time for tracking function. The work is divided into two essential objectives. First is to dynamically divide the processing operations into parallel units, and second is to reduce the communication between CPU-GPU processing units.

Keywords: Parallel computing, Visual surveillance, Graphics processing unit, Multi-core.

 DOI: http://dx.doi.org/10.22159/ajpcr.2017.v10s1.19651

INTRODUCTION

The visual surveillance scenario usually involves a varied process such as environment modeling, motion detection and its estimation, classification of objects and its tracking in real time; but such jobs are hefty in nature for the present computing hardware devices. Thus, it requires the presence of advanced servers or heavily modified devices to accomplish such computational tasks [1,2]. There are various relevant extensive investigations have already been made toward the surveillance or tracking based applications [3-7]. Moreover, such techniques are majorly comprised pre-trained knowledge of the scenes, where the objects geometrical interpretation or motion behavior is predefined to the algorithms in a manner [8,9]. Recent years have witnessed the success of other robust techniques for detection and tracking of people. This allowed the interest to be focused over higher and rich understanding of the scene in terms of computations or pixel orientations which indeed is a large and daunting task and can only be achieved with high powered processing units. In particular, this is an active focal point when there are several multi-networked camera units are used for security system; where three-dimension interpretation of the environment is built with the help of parallel behavior of assembled multiple hardware devices.

In the past decade, there was high rise in the processing units from central processing units-graphics processing units (CPUs-GPUs) where more and more cache units are binded with the multi-core processors [10-13]. Such techniques are extensively being used for computer vision tasks and image processing by the researchers [14-21]. Moreover, GPUs have played a vital role in this context. Thus, the developers worldwide have picked GPUs as the load balancing device for programing their applications [22]. The current generation of GPUs has many core processors and can accomplish the sequential task in a minimal execution time at the get go itself. However, the parallel implementation of the bulky programs over the GPUs requires redesigning the algorithm; such that the threading and pipelining of the code can be accomplished for parallel data computations [23-27]. Therefore, in this study, we proposed an algorithm to accomplish the same effectively for multiple object tracking.

THE MODEL

A parallel execution of a computational program is divided into two phases such as parallelism phase, computation phase, and interaction phase between CPU and GPU operations. The operations for object recognition and tracking algorithm are achieved through cited literature [28-30]. Thus, the total execution time (E_T) for the execution of a program parallely over CPU-GPU can be represented mathematically in form of the following equation as:

$$E_T = E_P + E_{Co} + E_C$$

$$E_T = \left(c + j\sqrt{\log_2 n}\right) t_f + n.\theta.t_c + t_c(n)$$

Where, E_P is the execution time for the division of task in paralleling sequence, E_{Co} is the execution time for the communication between CPU-GPU, E_C is the computational time over the processors. Furthermore, c is the number of cycles, n is the number of processors, t_f is the average time to execute a flop by the processor, t_c is the time for the load balancing of the CPU-GPU communication for the division and fetching of jobs, θ represents communication to communication ratio (from CPU to GPU), and t_c is the time taken for the computational operation over a GPU processor [31,32].

Now, that we need to reduce the time taken for the parallelism of the jobs and communication overheads for the CPU-GPU processing. Therefore, the objectives are respectively divided into two parts such as:

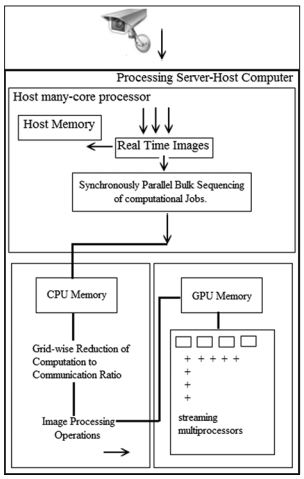

Fig. 1: Workflow operation of the proposed graphics processing unit schemes

1. Synchronously parallel bulk sequencing of computational jobs.
2. Grid-wise reduction of computation to communication ratio (Fig. 1).

Since parallelization of the jobs is differed by the architecture of the system, thus a lot of libraries had been already built upon it. However, to reduce the complexity of GPU based parallelizing process, we outlined our work upon the novel fussing of page ranking method for the effective memory utilization to parallelize the number of processing jobs. This algorithm replaces the recursive rounds of input/output with one step. This is summed in the following algorithm.

Algorithm: Synchronously parallel bulk sequencing algorithm

Input: N number of algorithm or parts of algorithm to be executed (vertex and edge), n number of processors, l = cost of synchronization, g = bandwidth.

Output: S_{cc} super step which consists of communication step between CPU-GPU, computation steps of parallelizing and the synchronization step.

Step 1: Evaluate and initialize a partition matrix:

$$\sum_{i=1}^{cc}\rho_{V,E}=\begin{cases}1 \text{ if } g\geq\dfrac{\sum_{i=0}^{c} w}{\sum_{j=1}^{c'} l}\\ 0 \text{ if } g>\dfrac{\sum_{i=0}^{c} w}{\sum_{j=1}^{c'} l}\end{cases}$$

Where, V and E are the vertex and edge of the graph of the memory page, w is the local computation in process. In addition, the c is the number of iteration in computation and c' in the number of communication overheads.

Step 2: Compute the page rank of the vertex:

$$\varphi_V=\frac{n-E}{|V|}+\sum_{i=1}^{cc}\rho_{V,E}\varphi_E+l*g$$

Step 3: Calculate the super step by:

$$S_{cc}=\sum_{i=1}^{cc}\rho_{V,E}*\varphi_V$$

Step 4: End process.

This reduces the variant of the memory page and maps the parallelization of computational jobs in one go. In addition, it emulates the optimized mapping of programing model over GPU framework. The other methods usually have one component per vertex, but the proposed algorithm uses the single balancing equation for parallelization depending on the allowable bandwidth in synchronous with the computational workload that to one iteration based on the page rank equations. This reduces the memory mapping and thus prioritizes the jobs based on page ranking. Irrespective of the shuffling the algorithm eschews the hashing table for effective memory utilization with respect to the amount of the jobs required for balancing the workload (Fig. 2).

CONCLUSION

Fig. 3 shows in this study, we have successful showed the effectiveness of the current synchronously parallel bulk sequencing algorithm to reduce

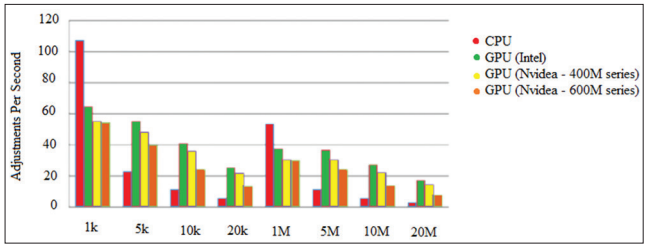

Fig. 2: The adjustments rate per second as per the proposed super step by central processing unit and different graphics processing unit devices at different operational level of the memory usage

Fig. 3: The illustration of the test frames for multiple-object detection using the proposed load balancing scheme with under 530 iterations of the input training patterns under a feasible time period with 0.21 delay with that of the real-time video

the time of operation for the input and the output cycles. This is a new affective approach for the others jobs to collaborate and algorithms for parallel job divisions, while the available GPU devices are ensured to avoid keeping the devices unnecessarily idle or wait for the another job to complete its execution as the parallelization and allocation of the memory is dynamic in nature with the proposed algorithm which suits it in a many-core processing environment. The result will allow other multi-camera based computer operations to be addressed in the future.

REFERENCES

1. Hu W, Tan T, Wang L, Maybank S. A survey on visual surveillance of object motion and behaviors. IEEE Trans Syst Man Cybern C 2004;34:334-52.
2. Velastin SA, Remagnino P. Intelligent Distributed Video Surveillance Systems. London, UK: IET Digital Library; 2006.
3. Collins RT, Lipton AJ, Kanade T. Introduction to the special section on video surveillance. IEEE Trans Pattern Anal Mach Intell 2000;22:745-6.
4. Howarth RJ, Buxton H. Conceptual descriptions from monitoring and watching image sequences. Image Vis Comput 2000;18(2):105-35.
5. Hu W, Xie D, Tan T. A hierarchical self-organizing approach for learning the patterns of motion trajectories. IEEE Trans Neural Netw 2004;15(1):135-44.
6. Tian Y, Tan TN, Sun HZ. A novel robust algorithm for real-time object tracking. Acta Automat Sin 2002;28:851-3.
7. Wu Y, Liu Q, Huang TS. An Adaptive Self-Organizing Color Segmentation Algorithm with Application to Robust Real-Time Human Hand Localization. In: Proceedings of 4th Asian Conference on Computer Vision, Taipei, Taiwan, 8-11, January; 2000. p. 1106-11.
8. Howarth RJ, Buxton H. Analogical representation of space and time. Image Vis Comput 1992;10(7):467-78.
9. Brand M, Kettnaker V. Discovery and segmentation of activities in video. IEEE Trans Pattern Anal Mach Intell 2000;22(8):844-51.
10. Garcia-Rodriguez J, Garcia-Chamizo JM. Surveillance and human-computer interaction applications of self-growing models. Appl Soft

Comput 2011;11(7):4413-43.
11. Nageswaran JM, Dutt N, Krichmar JL, Nicolau A, Veidenbaum A. Efficient Simulation of Large-Scale Spiking Neural Networks Using CUDA Graphics Processors. In: Proceedings of the 2009 International Joint Conference on Neural Networks, Atlanta, GA, USA, 14-9 June, 2009; p. 3201-8.
12. Nasse F, Thurau C, Fink GA. Face Detection Using GPU-Based Convolutional Neural Networks. In: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns. Berlin, Heidelberg, Germany: Springer-Verlag; 2009. p. 83-90.
13. Uetz R, Behnke S. Large-Scale Object Recognition with CUDA-Accelerated Hierarchical Neural Networks. In: Proceedings of 2009 IEEE International Conference on Intelligent Computing and Intelligent Systems. Vol. 1. Shanghai, China, 20-2 November, 2009; p. 536-41.
14. Che S, Boyer M, Meng J, Tarjan D, Sheaffer JW, Skadron K. A performance study of general-purpose applications on graphics processors using CUDA. J Parallel Distrib Comput 2008;68(10):1370-80.
15. Jang H, Park A, Jung K. Neural Network Implementation Using CUDA and OpenMP. In: Proceedings of the 2008 Digital Image Computing: Techniques and Applications, Canberra, ACT, Australia, 1-3 December; 2008. p. 155-61.
16. Kim J, Hwangbo M, Kanade T. Realtime Affine-Photometric KLT Feature Tracker on GPU in CUDA Framework. In: Proceedings of IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops), Kyoto, Japan, 27 September, 4 October; 2009. p. 886-93.
17. Oh S, Jung K. View-point insensitive human pose recognition using neural network and CUDA. World Acad Sci Eng Technol 2009;60:723-6.
18. Schwarz M, Stamminger M. Fast GPU-based adaptive tessellation with CUDA. Comput Graph Forum 2009;28(2):365-74.
19. Simek V, Asn RR. GPU Acceleration of 2D-DWT Image Compression in MATLAB with CUDA. In: Proceedings of the 2008 2nd UKSIM European Symposium on Computer Modeling and Simulation, Liverpool, UK, 8-10 September; 2008. p. 274-7.
20. Stone SS, Haldar JP, Tsao SC, Hwu WM, Sutton BP, Liang ZP. Accelerating advanced MRI reconstructions on GPUs. J Parallel Distrib Comput 2008;68(10):1307-18.
21. Hwu WW. GPU Computing Gems Emerald Edition. 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2011.
22. Garcia-Rodriguez J, Angelopoulou A, García-Chamizo JM, Psarrou A, Orts-Escolano S, Morell-Gimenez V. Fast Autonomous Growing Neural Gas. In: Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN), San Jose, CA, USA, 31 July, 5 August; 2011. p. 725-32.
23. Nickolls J, Dally WJ. The GPU computing era. IEEE Micro 2010;30:56-69.
24. Satish N, Harris M, Garland M. Designing Efficient Sorting Algorithms for Manycore GPUs. In: Proceedings of IEEE International Symposium on Parallel and Distributed Processing, Rome, Italy, 23-29 May; 2009. p. 1-10.
25. CUDA Programming Guide. Version 5.0, 2013. Available from: http://www.docs.nvidia.com/cuda/cuda-c-programming-guide. [Last accessed on 2013 Jun 12].
26. Kirk DB, Hwu WW. Programming Massively Parallel Processors: A Hands-on Approach. 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 2010.
27. Rai A. Attribute based level adaptive thresholding algorithm for object extraction. J Adv Robot 2015;1(2):64-8.
28. Rai A. Attribute based level adaptive thresholding algorithm (ABLATA) for image compression and transmission. J Math Comput Sci 2014;12:211-8.
29. Rai A. An introduction of smart self-learning shell programming interface. J Adv Shell Program 2015;1(2):3-6.
30. Rai A. Dynamic data flow based spatial sorting method for GPUs: Software based autonomous parallelization. Recent Trends Parallel Comput 2014;1(1):15-8.
31. Rai A. Dynamic pagination for efficient memory management over distributed computational architecture for swarm robotics. J Adv Shell Program 2014;1(2):1-4.
32. Rai A. Parallelizing mutations for genetic algorithm. Recent Trends Parallel Comput 2015;1(3):7-9.