

On Using Cell Broadband Engine For Object Detection in ITS

Luciano Oliveira, Ricardo Britto and Urbano Nunes

Abstract—Trade-off between accuracy and computational cost is usually hard to achieve in order to build a real-time and accurate object recognition system. Many theoretically conceived, efficient computer vision applications are avoided to run on-the-fly for hardware limitations. For recognizing objects in images and extracting object information, a system usually owns object searching, object classification and object tracking modules. Nevertheless, each one of these modules may be high time demanding, requiring to be implemented in different computers in order to achieve efficient runtime. On the other hand, more multi-core processor computers come to provide powerful processing towards full parallel implementations inside cost effective systems. In this way, the main contribution of this paper is not only a survey of works under the main architecture pipeline of object recognition systems in the point of view of computational cost but also pointing some directions to implement those systems in the cell broadband engine, coming with Playstation 3™ game console. The main purpose is the implementation of a complete object detection system for Intelligent Transportation Systems.

I. INTRODUCTION

A typical system for object recognition in still images (or videos) is usually composed by three main modules: object searching, object recognition and object tracking. These modules are primarily conceived to be integrated, while information transfer among them should provide more confidence in the decision of what or where are the objects in the images. Yet, the first two modules are probably the most computationally expensive since, after the objects are hypothesized, a kalman filter based tracking system is almost costless.

For building a complete object detection system for real life applications (for instance, Intelligent Transportation Systems (ITS)), some project issues should be considered. The main one resides in obtaining a trade-off between a high performance object detection¹ and on-the-fly implementation. In ITS field, for instance, applications hold typically at least 15 frames per second (fps) in order to provide an effective timing system [1], [2]. This frame rate uses to decrease as increasing the complexity of the algorithms applied.

This work is supported by Fundação de Ciência e Tecnologia de Portugal (FCT), under Grant PTDC/EEA-ACR/72226/2006. Luciano Oliveira is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), programme of Ministry of Education of Brazil, scholarship nº BEX 4000-05-6. The authors are with Institute of Systems and Robotics, Department of Electrical and Computer Engineering, University of Coimbra, Coimbra, Portugal {lreboucas, rbritto, urbano}@isr.uc.pt

¹The word "detection" is used here meaning that the location of the objects is found in the images, while "recognition" means applying a classification algorithm to decide if that location contains an object or not.

Following these ideas, this paper brings twofold contributions: a survey of works that propose detection systems with the aim of being applied to on-the-fly applications and a discussion of some directions to improve system speed without a considerable loss in detection performance, using Cell Broadband Engine (Cell) processors, coming with Playstation 3™ (PS3). Also, a preliminary analysis linking object detection systems to Cell implementation is presented.

The paper is structured as follows: in Section II, some works that use the Cell to speed up their applications are discussed; Section III presents a brief survey about state-of-the-art computer vision methods and systems proposed in the literature. The objective of this short survey is to discuss the main object detection pipeline, linking it with a parallel implementation onto the Cell; in Section IV, some points regard to the cell architecture and how to parallelize application in it are analyzed. Yet, a preliminary study using the Cell is presented. Finally, Section V draws some conclusions and future works.

II. RELATED WORKS

Many researches have been doing towards using Cell processor to parallelize applications, mainly considering the cost and benefits of those platforms. In [3], authors implement a receptive field-based neural network onto a cluster of 3 PS3s. The neural network has the objective to recognize barstools in videos. With that hardware architecture, the processing gain is about 140 times against 62 times by using a Field Programmable Gate Array (FPGA) based solution. Actually, with just one PS3, the gain is about 60 times, opposing to the high cost characteristic of FPGA solutions. In [4], a discussion of a 3D computer tomography (CT) application on Cell is given. The Feldkamp algorithm to reconstruct the CT images is implemented onto a cluster of 2 Cell computers. The authors based their implementation on concurrent computation and communication, using Direct Access Memory (DMA) instructions. The runtime was decreased by a factor of approximately 45 times. [5] addresses the problem of Ribonucleic Acid (RNA) sequence alignment. The speedup achieved, using a single Cell platform, was of 4.5 times in comparison to a optimized implementation of the same algorithm on a Pentium 4. Another research involving the use of Cell in Biology can be found in [6]. The authors proposes an experimental evaluation of two bioinformatics applications and their processing gain under a Cell architecture. Some critical issues are discussed in order to implement the algorithms on the Cell and an experimental time analysis is given.

III. OBJECT DETECTION PIPELINE

A typical object detection pipeline is illustrated in Fig. 1. This pipeline is commonly used in order to improve the confidence of the hypothesized objects in still images. Among the modules, the first two are usually built by means of high time demanding algorithm. The tracking module is generally the least computational demanding module.

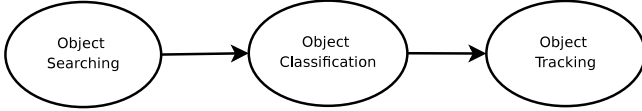


Fig. 1. **Object detection pipeline.** An object detection pipeline is typically composed by three modules: moving object segmentation, object classification and object tracking.

In the next sections, some details are given about the two most computational demanding modules and their characteristics which can be parallelized in a certain level of parallelization.

A. Object Searching

Finding objects in still images is a complex task, being accomplished in general under two main methods: brute force and moving object segmentation. In the first category, a normalized image window runs through various scales and positions over the input frame. The aim of this task is such that, in each position of the normalized window, features are extracted, feeding a trainable classifier, which will decide if the position contains an object or not. Figure 2 depicts the main idea of that pyramidal searching method. Concerning the effectiveness of finding an object, these techniques depend only on the performance of the classifier, which will be usually higher as more complex algorithms are used.

L+1 level

L level

L-1 level

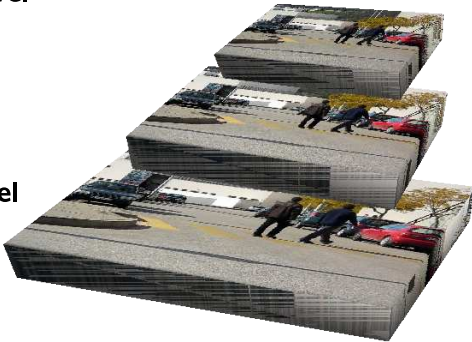


Fig. 2. **A pyramid object searching.** A normalized image window runs through all scales and positions over the input frame. The smaller is the scale of the input frame, the closer is the object.

To speed up pyramidal searches, [7] suggests a boosting method implemented over Haar-like features. There are other works which apply boosting methods over other kinds of feature extractors, like: Histogram of Oriented Gradients [8], Riemannian Manifold [9] and bag of scale invariant features

[10]. All these proposed methods use to run at between 10 and 20 fps, primarily depending on the number of objects in the images.

Under the perspective of the parallelization, the brute force methods rely on different computing parallelization strategies (onto PS3 platform): i) instruction level parallelization in the Synergistic Processing Unit (SPE); ii) image scale level parallelization, which each image scale should be sent to an SPE to be processed. The rationale behind the first one resides in speeding up the whole process, parallelizing as much as possible the way that each instruction of the algorithm is performed. The parallelization of the latter method, instead, should happen at the moment that each scale of the pyramidal (or part of one image; see Section IV-C) is sent to each SPE, making the parallelization to be accomplished in a higher level programming technique. Yet, if the size of each scaled image is longer than the Local Store (LS) (see Fig. 4) presented in each SPE, one should thinking about other strategies to break the processing in multiples clock cycles [3].

In moving object segmentation methods, the way that the image Region of Interest (ROI) is found usually relies on faster methods, since the main objective is to segmenting what is foreground and background based on hints of movement. Some works can be found in [11], [12], [1], [13]. The success of this type of method resides in a good hypothesis technique based on various clues. In other words, it is better to obtain false positives than miss detections, in view of the fact that in the object classification step a false positive can be corrected by a more reliable classifier method. The parallelization of these method depending on the algorithms used. Generally, optical flow and particle filter are used to estimate the difference between the foreground and background. In this way, an instruction level parallelization technique should present better results, since a scale level approach is not usually feasible.

B. Object Classification

Object classification can be accomplished by supervised or unsupervised approaches. In this section, the supervised deterministic approach is presented towards to analyze ways of parallelizing it. Several works utilize deterministic approaches in order to classify objects in images and some of them can be found in [14], [15], [16], [17], [18].

A supervised deterministic approach mostly relies on multiplying the input vector x to a weight matrix w , adding the result to a bias vector b , according to:

$$f(x) = \text{sign}(\langle w \cdot x \rangle + b) = \text{sign}\left(\sum_{i=1}^l w_i x_i + b\right) \quad (1)$$

If the $f(x)$ is greater than zero, then the input vector is considered an object, otherwise it is considered a non-object. The weight and bias vectors are achieved after the training stage.

This type of classification system fits the acceleration approach implemented by the Cell, since that architecture are

composed by inherently vector processors. The vectorization structure takes place by means of SIMD instructions, being able to multiple data in the same clock. The programming approach is performed by using C language but in a different manner, trying to explore the SIMD processing. Fig. 3 illustrates an example of a vector operation.

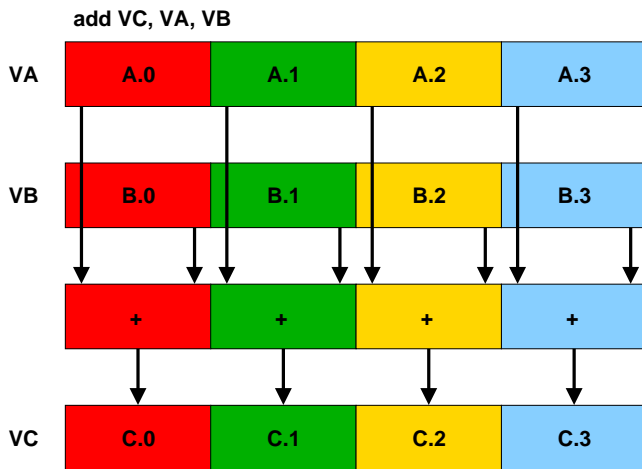


Fig. 3. **SIMD processing.** Adding two vectors VA and VB and sending the result to a third vector VC [19].

IV. EXPLORING PARALLELISM USING CELL

In 2000, Sony, Toshiba and IBM jointly began the development of an ambitious project of a new processor architecture, called Cell. This architecture is a powerful heterogeneous multi-core processor, developed to be used in many platforms, as high-definition televisions, supercomputers and game consoles, namely the Sony PS3. Nowadays, PS3 is used in many scientific computing projects due to presenting a suitable trade-off between cost and benefits.

This section presents the Cell architecture, as well as a preliminary study about the viability of using the Cell processor (using PS3) to image object detection in ITS.

A. Cell Architecture

Cell is a heterogeneous multi-core architecture, containing a dual-threaded processor unit, denominated Power Processing Element (PPE) and eight co-processors called Synergistic Processing Element [21]. Figure 4 depicts that architecture.

PPE is a 64-bit, two-way simultaneous multithreading which is compliant with the PowerPC 970 Architecture. PPE has one Power Processor Unit (PPU), 32 Kbytes of L1 cache, and 512 Kbytes of L2 cache. The PPU uses the PowerPC 970 instruction set with a SIMD engine called Vector Multimedia eXtension (VMX). SPE, where the real power of CELL processor lies on, consists of a Synergistic Processing Unit (SPU), 256 Kbytes of LS, and a Memory Flow Controller (MFC) that delivers powerful direct memory-access capabilities to the SPU using DMA. The SPEs possess a 128-bit vector register file and a range of SIMD instructions that can operate simultaneously on two double-precision values, four single-precision values, eight 16-bit integers or 16 8-bit characters. There are instructions pipelined which can

complete one vector operation in one clock cycle. The Cell components are connected via the Element Interconnection Bus (EIB). EIB has four unidirectional rings, two in each direction. Allied to these, there is a token-based mechanism which permits a internal speed of 204.8 Gbytes/s. The Cell is mainly a distributed memory processor, where each SPE has its private memory. There is an explicit control over data motion, where one can use mechanism like mailbox [21].

B. Playstation 3

PS3 is the cheapest Cell based platform in the market. It contains one Cell processor (with only six SPEs available for programming), 256 Mbytes of main memory, an NVIDIATM graphics card with 256 Mbytes of memory, and a gigabit Ethernet (GigE) network card. There is a virtualization layer which provides access to the PS3 resources, which means that one is not able to directly access the PS3 hardware under linux.

In order to use the full power of the CELL processor inside PS3, one must split programs into threads, assigning each one to an SPE. [3] suggests 5 levels of programming parallelization: (1) parallel execution of many Cells (cluster of Cells), (2) a parallelization of a program onto the 6 SPEs and 1 PPE, (3) concurrent communication of the SPEs via DMA, (4) instruction level parallelization in the SPE, and (5) SIMD parallelization using 128-bit data path.

C. Optimizing Object Detection Algorithms

Object detection algorithms appear like suitable ones for a CELL optimized execution. In the current state of our work, we identify two approaches to use the whole power of CELL processor in order to classify objects in images: fragmentation of the image in equal number of pieces (sending each piece to a different SPE) and a pyramidal searching. The first approach is illustrated in Fig. 5. The goal is to split the image into six pieces of the same size. After that, each SPE receives one of these pieces and executes the object detection algorithm over the image piece. At the same time, each image piece is processed by each SPE, improving the performance of the algorithm execution. The PPE has the role of fragmenting the original image, sending the resulted pieces to the SPEs.



Fig. 5. **Fragmentation Approach.** A single image is fragmented into same sized pieces. Each piece is then sent to one different SPE.

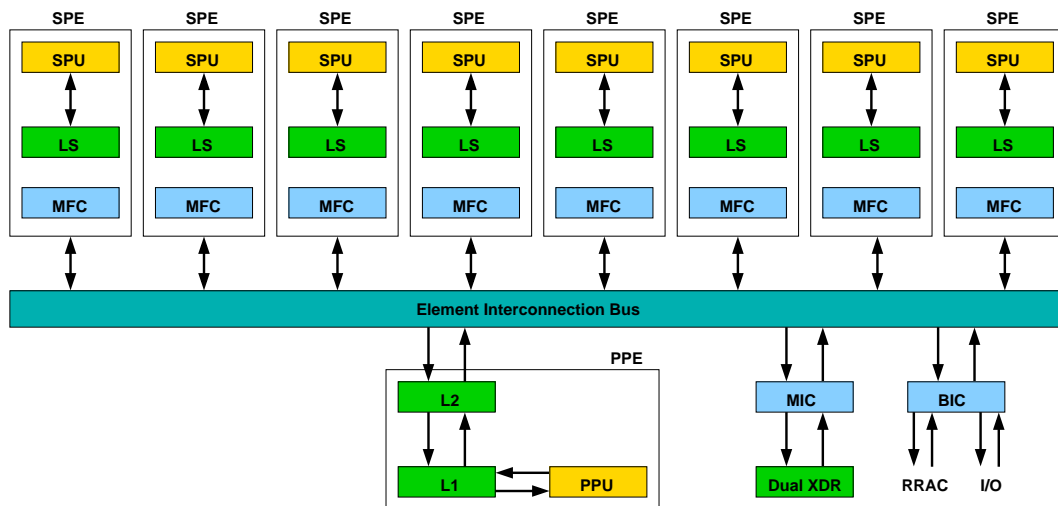


Fig. 4. **Cell Processor Design.** The architecture is composed by a dual-threaded processor called PPE and 8 vector processors, called SPE. Only 6 SPEs are programmable available under Linux.

The second approach (Fig. 6) uses the multi-resolution technique to create a pyramid with six stages, where each stage is the original image in a different resolution. Each SPE receives one pyramid level, performing the object detection algorithm on it. PPE is still in charge to receive each frame, scale it and send to each SPE.

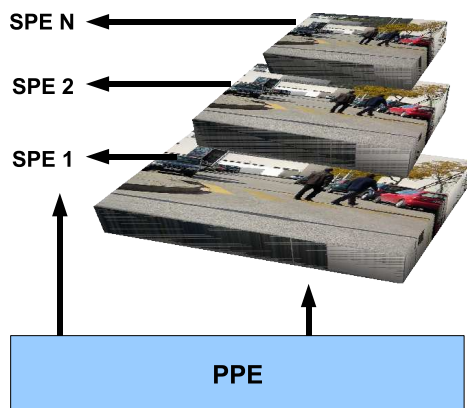


Fig. 6. **Multi-resolution Approach.** Each level of the pyramid is sent to an SPE. This approach is feasible as long as each scaled image can fit in the LS of each SPE. Otherwise, a multiple step technique should be implemented to process each image, considering multiple references to the LS.

V. CONCLUSION

The most powerful unit in Cell architectures is undoubtedly the SPEs. These processors are mainly vector-driven, and SIMD instructions are pervasively implemented for performance gain. On the other hand, within the pipeline of object detection, many points are found to be vectorized, being suitable to be implemented whether in a single Cell or in a cluster of Cells. Nowadays, PS3 game console is the one of

the cheapest computational system sold, with a great trade-off between cost and benefits. Following these ideas, the next step is to build a complete object detection system on a cluster of PS3.

REFERENCES

- [1] L. Llorca, M. Sotelo, L. Bergasa, P. Toro, J. Nuevo, M. Ocana and M. Garrido, "Combination of Feature Extraction Methods for SVM Pedestrian Detection", in *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, n. 2, 2006, pp 292–307.
- [2] G. Monteiro, C. Premebeda, P. Peixoto and U. Nunes, "Tracking and Classification of Dynamic Obstacles Using Laser Range Finder and Vision", in *Workshop on Safe Navigation in Open and Dynamic Environments, held at the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006, pp 213–219.
- [3] A. Felch, J. Nageswaran, A. Chandrashekar, J. Furlong, N. Dutt, R. Granger, A. Nicolau and A. Veidenbaumand, "Accelerating Brain Circuit Simulations of Object Recognition with CELL Processors", in *International Workshop on Innovative Architecture for Future generation Processors and Systems*, 2007, pp 33–42, 2007.
- [4] M. Sakamoto and M. Murase, "Parallel Implementation for 3-D CT Image Reconstruction on Cell Broadband Engine", in *IEEE International Conference on Multimedia and Expo*, 2007, pp 276–279.
- [5] A. Sarje and S. Aluru, "Parallel Biological Sequence Alignments on the Cell Broadband Engine", in *IEEE International Parallel and Distributed Processing Symposium*, 2008, pp 14–18, 2008.
- [6] V. Sachdeva, M. Kistler, E. Speight and T-H Tzeng, "Exploring the Viability of the Cell Broadband Engine for Bioinformatics Applications", in *IEEE International Parallel and Distributed Processing Symposium*, 2007, pp 1–8.
- [7] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp 511–518.
- [8] Q. Zhu, S. Avidan, M-C Yeh and K-T Cheng, "Fast Human Detection Using a Cascade of Histograms of Oriented Gradients", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2006, pp 1491–1498.
- [9] O. Tuzel, F. Porikli and P. Meer, "Human Detection via Classification on Riemannian Manifolds", in *IEEE Conference on Computer Vision and Pattern Recognition*, 2007, pp 1–8.
- [10] I. Laptev, "Improvements of Object Detection Using Boosted Histograms", in *British Machine Vision Conference (BMVC06)*, 2006, pp 949–958.
- [11] A. Talukder, S. Goldberg, L. Matthies and A. Ansar, "Real-time Detection of Moving Objects in a Dynamic Scene from Moving Robotic Vehicles", in *IEEE Conference on Intelligent Robots and Systems (IROS03)*, 2003, pp 1308–1313.

- [12] J. Boyoon and S. Gaurav, "Detecting Moving Objects using a Single Camera on a Mobile Robot in an Outdoor Environment", in *Conference on Intelligent Autonomous Systems*, 2004, pp 980–987.
- [13] K. Yamaguchi, T. Kato and Y. Ninomiya, "Vehicle Ego-Motion Estimation and Moving Object Detection using a Monocular Camera", in *IEEE Transactions on Intelligent Transportation Systems*, vol. 4, 2006, pp 610–613.
- [14] C. Papageorgiou and T. Poggio, "A Trainable System for Object Detection", *International Journal of Computer Vision*, vol. 38, 2005, pp 15–33.
- [15] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection", *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005, pp 886–893.
- [16] M. Szarvas, U. Sakai and J. Ogata, "Real-time Pedestrian Detection Using LIDAR and Convolutional Neural Networks", *IEEE International Symposium on Intelligent Vehicles*, 2006, pp 213–218.
- [17] L. Oliveira, G. Monteiro, P. Peixoto and U. Nunes, "Towards a Robust Vision-based Obstacle Perception with Classifier Fusion in Cybercars", *Computer Aided System Theory (Eurocast 2007), Lecture Notes in Computer Science (LNCS)*, Springer-Verlag, vol. 4739, 2007, pp 1089–1096.
- [18] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber and T. Poggio, "Robust Object Recognition with Cortex-like Mechanisms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, n. 3, 2007, pp. 411–426.
- [19] A. Buttari, P. Luszczek, J. Kurzak, J. Dongarra and G. Bosilca "A Rough Guide to Scientific Computing On the PlayStation 3", *Technical Report UT-CS-07-595*, 2007.
- [20] J. Kurzak, A. Buttari, P. Luszczek and J. Dongarra, "The PlayStation 3 for High-Performance Scientific Computing", in *Computing in Science & Engineering*, 2008, pp 84–87.
- [21] D. Pham, S. Asano, M. Bolliger, M.N. Day, H.P. Hofstee, C. Johns, J. Kahle, A. Kameyama, J. Keaty, Y. Masubuchi, M. Riley, D. Shippy, D. Stasiak, M. Suzuoki, M. Wang, J. Warnock, S. Weitzel, D. Wendel, T. Yamazaki and K.Yazawa, "The design and implementation of a first-generation CELL processor", *IEEE International Conference on Solid-State Circuits (Digest of Technical Reports)*, 2005, pp 184-592.