A Parallel Architecture for Multiple-Face Detection Technique Using AdaBoost Algorithm and Haar Cascade Keywords: Parallel architecture Multiple-face detection AdaBoost algorithm H...

Conference Paper · December 2013

2 authors, including:

Hadi Santoso
Institut Sains dan Bisnis Atma Luhur Pangkalpinang, Indonesia

14 PUBLICATIONS   **7** CITATIONS

Some of the authors of this publication are also working on these related projects:

Project   K-Means Algorithm on Home Industry View project

A Parallel Architecture for Multiple-Face Detection Technique Using AdaBoost Algorithm and Haar Cascade

Hadi Santoso*, Reza Pulungan**

*Department of Information System, Faculty of Information Technology, STMIK Atma Luhur, Pangkalpinang, Bangka
Belitung Island Province, Indonesia, E-mail: hadimkom@gmail.com, hds4n@yahoo.co.id
**Jurusan Ilmu Komputer dan Elektronika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada,
FMIPA UGM Gedung Selatan, Sekip Unit III, Yogyakarta, Indonesia, E-mail: pulungan@ugm.ac.id

Keywords:	ABSTRACT
Parallel architecture Multiple-face detection AdaBoost algorithm Haar cascade	Face detection is a very important biometric application in the field of image analysis and computer vision. The basic face detection method is AdaBoost algorithm with a cascading Haar-like feature classifiers based on the framework proposed by Viola and Jones. Real-time multiple-face detection, for instance on CCTVs with high resolution, is a computation-intensive procedure. If the procedure is performed sequentially, an optimal real-time performance will not be achieved. In this paper we propose an architectural design for a parallel and multiple-face detection technique based on Viola and Jones' framework. To do this systematically, we look at the problem from 4 points of view, namely: data processing taxonomy, parallel memory architecture, the model of parallel programming, as well as the design of parallel program. We also build a prototype of the proposed parallel technique and conduct a series of experiments to investigate the gained acceleration.

Corresponding Author:

Hadi Santoso
Departement of Information System, Faculty of Information Technology, STMIK Atma Luhur, Jalan Raya Sungailiat,
Selindung Lama-Pangkalpinang, Bangka Belitung Island Province, Indonesia.
E-mail: hadimkom@gmail.com, hds4n@yahoo.co.id

1. INTRODUCTION

Face detection is a process of identifying all regions of an image that contain a face regardless of position, orientation, and environmental conditions in the image [1]. Face detection is used in a variety of applications, such as control and security applications, and identification systems. Face detection algorithms have been widely researched and improved dramatically both in terms of performance and speed. Viola and Jones in [2] proposed a framework for face detection based on AdaBoost learning algorithm using Haar-like features. Typically face detection algorithms require a high-performance computation.

The development of microprocessors has been known to more or less follow Moore's law [3]. Increasing computation performance takes place rapidly with the growth of microprocessor. However, this increase is hardly able to catch up with the rapid advancement in image and video technology, including camera resolution, and in the computational requirements in many emerging real-time applications.

Parallelism was born from the approach used by system designers to implement concurrent processing concepts. This technique improves processing speed by multiplying the number of hardware modules that operate simultaneously, accompanied by forming multiple processes working simultaneously on hardware modules. Parallel processing and multi-core architectures have become key to achieving high performance in future computing systems [4]. In multi-core system standards, the level of parallelization is fixed and the performance can be improved only with certain architectural features, such as SIMD units or exploitation of cache effects [5]. In this paper we examine the approach by focusing on the parallel processing and multi-core architecture that work on desktops or notebook computers.

Our contribution. Our contribution is threefold: (1) we propose an architectural design for a parallel and multiple-face detection technique based on Viola-Jones framework, by looking at the problem from 4 point of views: data processing taxonomy, parallel memory architecture, the model of parallel programming, and

the design of parallel program; (2) we build a prototype implementing the proposed algorithm; and (3) we conduct a series of experiments to demonstrate the acceleration of multi-face detection process parallelism.

Previous work. Viola-Jones method is the first design for a rapid object detection technique for real-time applications. Detector based on AdaBoost algorithm and Haar-like features is able to detect 15 frames per second in QVGA 1II 700MHz Pentium processor [2]. Deng and Pei in [6] proposed a multi-view face detection method, which combines skin color information and AdaBoost. Their method was tested with a set of 50 high-resolution images (640×480) on a 2.4GHz Pentium 4 processor. Detection rate reached more than 95% and the processing time is approximately 400 milliseconds per image [6]. A faster variant based on AdaBoost was proposed in [7]. The author showed that approximately 134,736 sub-windows are processed by the primitive search method; while, after optimization, the method they proposed could narrow down the number of processed sub-windows to only 38,532.

Another way to improve the performance is to run the face detection algorithm in multi-processor or multi-core systems. Open Source Computer Vision Library (OpenCV) provides many optimized image-processing algorithms using multi-threading technique. OpenCV can be optimized by using the single instruction multiple data (SIMD) technique or other special instructions as shown in [8]. A study conducted by Puppala in [9] used Very Long Instruction Word-SIMD (VLIW-SIMD) processor architecture to accelerate object detection algorithms in embedded applications, and resulted in a detection rate of 8.33 fps. Chen in [10] showed how to analyze parallelism at the thread level, and how to choose the right one to take advantage of existing 4-core and 8-core processors. By optimization and parallelization, face detection with AdaBoost algorithm utilizing multi-core processors achieved result that is 7 times faster than the serial version. Shekhar and Varaganti [11] presented a method to efficiently parallelize face detection method, which can be developed on any object detection algorithms for Symmetric Multi-Processing (SMP) architecture. This research indicated that a well-designed parallel face detection algorithm would result in a performance gain 2 times faster than the dual-core systems [11].

A parallel variant of Viola-Jones method has been developed on a single GPU + CPU desktop system with a high bandwidth to achieve GPU thread parallelism [12]. The authors reported that the system, which is running on a NVIDIA GeForce GTX260 graphic card, could achieve a speed of 12 fps and detection rate of 92%. The performance of an implementation of Viola-Jones face detection method using NVIDIA and AMD GPUs showed speed-ups that reach up to 5.193 until 35.08 times (average 16.91) and, respectively, 5.85 until 32.641 times (average 17.535) [13]. Most researches focus on the use of hardware architectures to improve performance. Our approach is to focus on achieving better performance on multi-core systems, in the hope that the proposed design can be easily applied to the SMP systems without any additional hardware.

Organization of the paper. This paper is divided into 4 sections. In Section 2, we introduce the face detection method of Viola and Jones. Section 3 presents the design of the parallel architecture for the algorithm. In Section 5, we discuss implementation issues and several experiments we conducted. Section 5 concludes the paper.

2. RESEARCH METHOD
2.1. Face Detection with Viola-Jones Method

Face detection by Viola-Jones method has been widely used by many researchers in a variety of applications because of its high degree of accuracy and speed. Broadly, Viola-Jones method consists in 4 four steps, as depicted by Figure 1.
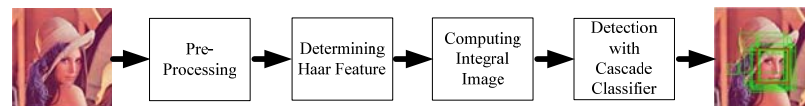

Figure 1. Face detection process with Viola-Jones method

Pre-processing. At this stage, an image, as an input, undergoes scaling and gray-scaling processes:
   a. Scaling is the process of resizing a digital image. This is performed so that all digital images have the same size. To perform scaling in our implementation, we take advantage of the package display object and we reduce the size of digital images by using the interpolation method.
   b. Gray-scaling is the process of converting a full-colour digital image into a two-colour image once it has been through the process of scaling.

Determining Haar-like features. After scaling and gray-scaling in the first stage, Viola-Jones algorithms will attempt to systematically detect faces in the image and at the same time determine their positions. This is

carried out by looking for features that have high degrees of differentiation in all possible rectangular regions (called sub-windows) in the image. For this purpose, the algorithm uses a selection of Haar-like features. A Haar-like feature is a contrasting pattern in several (usually from 2 to 4) adjacent rectangular regions in an image; cf. Figure 2. The pixel intensities in each region are summed and the difference between these sums is calculated. This difference determines the Haar-like feature found in the corresponding sub-window formed by the adjacent rectangular regions. Figure 2 depicts four Haar-like features used in this study, while Figure 3 shows how these features roughly correspond with the contrasting shades of faces in an image.

Figure 2. A selection of Haar-like features

Figure 3. The correspondence of Haar-like features and contrast in image

Computing integral image. Integral image was first introduced in [14] and it is used to quickly compute and find Haar-like features in an image [15]. Integral image at location (x,y) is defined as the pixel intensity at the rectangular region from the top left (0,0) down to point (x,y) [15], namely:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i(x',y')$$

where $ii(x,y)$ is the integral value of all image pixels in a rectangular region from (0,0) to (x,y), while $i(x,y)$ is the pixel value at location (x,y). The computation of integral image can be performed quickly and efficiently for all points in the image once and for all. Once the integral image of all points is obtained, the pixel intensity of any sub-window in the image can be computed using only at most four memory references. For instance, cf. Figure 4, the integral image value at point 4 ($ii(4)$) is the sum of sub-windows A + B + C + D. Similarly, the integral image value at point 3 ($ii(3)$) is the sum of sub-windows A + C. Therefore, the integral image value of sub-window D can be obtained simply by computing ($ii(1) + ii(4)$) - ($ii(2) + ii(3)$), which only requires four memory references.
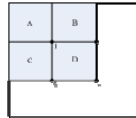
Figure 4. Using integral image to find the pixel intensities in any sub-window [2]

Detection with cascading classifiers. A series of classifiers (namely Haar-like features, each with increasing importance) is applied to each sub-window. In the first step of the classification, each sub-window will be classified using a particular feature. If the result of the feature does not meet the desired criteria, the result was rejected. The algorithm then moves on to the next sub-window and calculates the value of feature again. If the result obtained fulfills the desired threshold, then the algorithm proceeds with the next steps, i.e. the next features in the series. A sub-window is considered containing a face, if it passes through all steps of the classification. Figure 5 depicts an example of the face detection result. By combining the classification in a cascading structure, the speed of detection can be greatly improved, as sub-windows that contain insufficient number of Haar-like features are discarded rapidly.

Figure 5. Face detection results

2.2. The Design of Parallel Architecture

Data processing taxonomy. Based on Michael J. Flynn's data processing taxonomy [16], the designed parallel architecture employs Single Instruction Multiple Data (SIMD) model. This model allows a single

instruction to operate on multiple data elements, which results in reduced code size and improved performance. SIMD operations also reduce the total number of instructions and allow more efficient movement of data, which leads to increased energy efficiency [16].
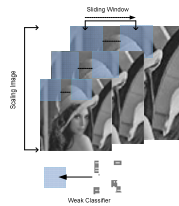

Figure 6. Sliding windows in Viola and Jones method

 Three sources of possible parallel data processing can be identified in the Viola-Jones method, namely:
1. The computation of integral image.
2. The processing of sub-windows: As show in Figure 6, face detection method of Viola and Jones uses the principle of the sliding window. The method performs an exhaustive search of all possible sub-windows and then determines whether a set of Haar-like features is found in each sub-window. Input image may contain tens of thousands of sub-windows. Each sub-window can be processed in parallel because they are independent of each other.
3. The processing of classifiers: each classifier in the AdaBoost algorithm can be computed in parallel since they are also independent of each other. In fact, this is desirable, since most of the classifiers are weak and therefore will be discarded.

Parallel memory architecture. The proposed parallel architecture employs a shared memory architecture [4]. In shared memory architecture, multiple CPUs using a shared memory resource, where all CPUs and the memory are connected via a single bus. All processors can then access all memory as a global address space. The main reason for selecting this architecture is because we wish to develop a parallel architecture that is especially suitable for multi-core environments, instead of computer clusters.

Model of parallel programming. In software approach of parallel programming, the best and simplest way to implement parallelism is to use multithreading. It is even more encouraging to note that almost all modern programming libraries, which usually provide multithreading facilities, provide a mechanism to automatically execute different threads in different processor cores.

Three widely used programming models are available nowadays: OpenMP, POSIX threads (Pthreads) and the Message Passing Interface (MPI). OpenMP is a set of Application Programming Interfaces, which is used to develop parallel applications on shared-memory platforms [17]. OpenMP developers just use compiler commands to parallelize existing sequential programs and need not worry about synchronization and communication, as calculation tasks (e.g.: command loop) annotated with compiler's parallel commands are automatically allocated to thread [18].

Design of Parallel Program. Increasing the efficiency of and optimizing program code for modern parallel architectures is a time consuming task and tends to produce mistakes. Designing parallel programs requires many iterations of code transformations, tuning and performance analysis, and in many cases, has to be redone at each different architecture [19].
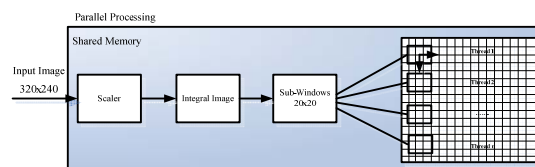

Figure 7. The parallelization in the proposed method

In our case, we design the parallel multiple-face detection method based on Viola-Jones' by making use of an existing serial implementation, which is based on OpenCV. In particular, we decide to exploit and parallelize the last two of the three sources of possible parallel data processing mentioned above. This is done by locating sections of the source code implementing the corresponding sources in the serial implementation, and parallelizing them using available compiler commands. Each sub-window within an image can be independently processed by a classifier. By duplicating the classifier, it is relatively simple to parallelize the

sweeping, where a thread is assigned a set of sub-windows of the same rows to process. The architectural design of the parallelization is shown in Figure 7.

3. RESULTS AND ANALYSIS

To demonstrate the feasibility of the proposed parallel architecture, we perform a series of experiments on a notebook with Intel® Core™ 2 duo, dual-core processor running at 2.00 GHz, 2 GB of memory, and a desktop computer with Intel® Core™ 2, quad-core processor running at 2.33 GHz, 2 GB of memory. Both systems run Microsoft Windows 7 Operating System.

In the experiments, we use ten 320×240 pixel images; each depicting human faces of varying numbers. We then record the computation times (in seconds) of both serial and parallel implementations of the Viola-Jones method run on the two different machine configurations. The experimental result is shown in Table 1.

Table 1. Result of experiments with serial and parallel multiple-face detection

No.	Images	Number of Faces	Serial (second)		Parallel (second)		Acceleration	
			2 core	4 core	2 core	4 core	2 core	4 core
1	DBHADI_1	1	2.1	1.0	0.10	0.01	21	100
2.	DBHADI_2	2	3.2	1.5	0.22	0.02	14.5	75
3	DBHADI_3	3	3.2	1.5	0.22	0.02	14.5	75
4	DBHADI_4	3	4.3	2.2	0.43	0.03	10	73.3
5	DBHADI_5	4	5.6	3.3	0.56	0.05	10	66
6	DBHADI_6	7	7.3	5.7	0.67	0.06	10.8	95
7	DBHADI_7	8	8.1	6.2	0.72	0.07	11.2	88.5
8	DBHADI_8	10	9.7	7.1	0.84	0.08	11.5	88.7
9	DBHADI_9	28	10.4	8.2	0.92	0.09	11.3	91.1
10	DBHADI_10	31	11.2	9.6	1.00	0.10	11.2	96

Table 1 indicates that the parallel face detection systems on a 2-core processor has the performance improvement ranging from 10 seconds to 21 seconds when processing a 320×240 image. On a 4-core processor, on the other hand, the performance improvement ranges from 66 seconds to 100 seconds. We observe that more faces in the input image results in slower performance gain. This is to be expected, since each face must pass through more classifiers and therefore takes more time to process. Figure 8 shows the result of the proposed face detection system. The red squares represent the detected face on the image.

Figure 8. Experimental result of the multiple-face detection systems

4. CONCLUSION

We have laid out our proposal of an architectural design for a parallel and multiple-face detection technique based on Viola and Jones framework in this paper. We have also built a prototype of the proposed parallel technique and conducted a series of experiments to investigate the gained acceleration. The proposed approach has proven to be effective. Since input images may contain tens of thousands of sub-windows and the classifier in the AdaBoost algorithm must compute these one by one, the performance gain from parallelizing the algorithm by replicating the classifier is enormous. The experiment we have conducted validates this. Performance gains of up to around 21 and 100 times can be obtained on 2-core and respectively 4-core systems.

We are planning to perform further studies on tuning the prototypical parallel implementation and adding more features and improvements. We are also interested in studying the energy consumption of the proposed parallel method, as face detection technique is increasingly used in mobile devices equipped with multi-core processors.

REFERENCES

[1] T. Theocharides, N. Vijaykrishnan, and M. J. Irwin, "A Parallel Architecture For Hardware Face Detection", University of Cyprus Penn State University," 2006.

[2] P. Viola and M. Jones, "Robust Real-time Object Detection," *Second International Workshop On Statistical And Computional Theories OF Vision – Modeling, Learning,Computing, And Sampling,* Vancouver, Canada, 2001.

[3] G. E. Moore and L. Fellow, "Cramming More Components onto Integrated Circuits," *Proceeding Of The IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[4] C.-H. Chiang, C.-H. Kao, G.-R. Li, and B.-C. C. Lai, "Multi-level parallelism analysis of face detection on a shared memory multi-core system," *Proceedings of 2011 International Symposium on VLSI Design, Automation and Test*, vol. 1, pp. 1–4, Apr. 2011.

[5] M. Schmidt, D. Fey, and M. Reichenbach, "Parallel Embedded Computing Architectures," *Embedded System Institute, Friedrich-Alexander-University Erlangen-Nuremberg Germany*, 2005.

[6] P. Deng and M. Pei, "Multi-View Face Detection Based on AdaBoost and Skin Color," *2008 First International Conference on Intelligent Networks and Intelligent Systems*, pp. 457–460, Nov. 2008.

[7] L. Lang and W. Gu, "Improved Face Detection Algorithm Based on Adaboost," *2009 International Conference on Electronic Computer Technology*, pp. 183–186, Feb. 2009.

[8] S.-K. Chen, T.-J. Lin, and C.-W. Liu, "Parallel object detection on multicore platforms," *2009 IEEE Workshop on Signal Processing Systems*, pp. 075–080, Oct. 2009.

[9] V. G. Puppala, "VLIW – SIMD Processor based Scalable Architecure for Parallel Classifier Node Computing," no. 1, pp. 1496–1502, 2012.

[10] Y.-K. Chen, "Parallelization of AdaBoost algorithm on multi-core processors," *2008 IEEE Workshop on Signal Processing Systems*, no. 978, pp. 275–280, Oct. 2008.

[11] T. C. Deepak Shekhar and K. Varaganti, "Parallelization of Face Detection Engine," *2010 39th International Conference on Parallel Processing Workshops*, pp. 113–117, Sep. 2010.

[12] Gaowei and Cheming, "The face detection system based on GPU+CPU desktop cluster," *2011 International Conference on Multimedia Technology*, pp. 3735–3738, Jul. 2011.

[13] H. Jia, Y. Zhang, W. Wang, and J. Xu, "Accelerating Viola-Jones Facce Detection Algorithm on GPUs," *2012 IEEE 14th International Conference on High Performance Computing and Communication & 2012 IEEE 9th International Conference on Embedded Software and Systems*, pp. 396–403, Jun. 2012.

[14] F. C. Crow, "Summed-area tables for texture mapping," *ACM SIGGRAPH Computer Graphics*, vol. 18, no. 3, pp. 207–212, Jul. 1984.

[15] C. Zhang and Z. Zhang, "A Survey of Recent Advances in Face Detection," no. June, 2010.

[16] M. J. Flynn, "Some Computer Organizations and Their Effectiveness," vol. c, no. 9, pp. 948–960, 1972.

[17] H. Xiao, T. Isshiki, D. Li, H. Kunieda, Y. Nakase, and S. Kimura, "Optimized Communication and Synchronization for Embedded Multiprocessors Using ASIP Methodology," *IPSJ Transactions on System LSI Design Methodology*, vol. 5, no. i, pp. 118–132, 2012.

[18] H. Zhou, X. Wang, and Y.-H. Guo, "A Parallel Computing Component for Linux Cluster with Threads Binding Supports," *2010 International Conference on Computational Intelligence and Software Engineering*, pp. 1–4, Dec. 2010.

[19] H. Jordan, P. Thoman, J. J. Durillo, S. Pellegrini, P. Gschwandtner, T. Fahringer, and H. Moritsch, "A multi-objective auto-tuning framework for parallel codes," *2012 International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–12, Nov. 2012.

BIBLIOGRAPHY OF AUTHORS

Hadi Santoso received his bachelor's degree in 2000 from STMIK Budi Luhur, Indonesia; his master's degree in 2011 from Universitas Budi Luhur, Indonesia; Since 2002 he has been a lecturer and a researcher at the department of Information System, STMIK Atma Luhur, Indonesia. His research interests lie in the field of computer vision, image processing, and database. He is currently studying Computer Science at the Faculty of Mathematics and Natural Sciences's doctoral program, Universitas Gadjah Mada, Indonesia.

Reza Pulungan received his bachelor's degree in 1999 from Universitas Gadjah Mada, Indonesia; his master's degree in 2004 from the University of Twente, the Netherlands; and his doctoral degree in computer science in 2009 from Saarland University, Germany. Since 2009 he has been a lecturer and a researcher at the department of Computer Science and Electronics, Universitas Gadjah Mada. His research interests lie in the field of stochastic processes, especially Markov processes and phase-type distributions. He is also interested in modeling and analysis of networked systems.