SPEEDING UP EXECUTION TIME OF A SMART WHEELCHAIR COMMAND TECHNIQUE USING PARALLEL COMPUTING

Agnès Ghorbel, Mohamed Jallouli and Nader Ben Amor

Computer and Embedded Systems: CES Laboratory
ENIS, Université de Sfax, Tunisie.

ABSTRACT

An Human Machine Interface for wheelchair control based on facial expressions is presented. The interface is implemented on embedded system architecture based on ARM processor rather than personal computer, like usual wheelchair command implementation, to reduce energy consumption while maintaining similar computing performance. The command technique code is complex but offers inherent parallelism. To reduce processing time, two parallelism levels are exploited. The first one is instruction parallelism which is applied to a dual core architecture using OpenMP directives. The second level is data parallelism in which an SIMD specific unit is exploited. In both parallelization techniques, a minimum of initial code re-manipulation is required. As processing unit, we choose the Pandaboard-ES platform that includes a dual-core ARM9 and a set of control interfaces. The obtained preliminary experiments demonstrate the effectiveness of this parallelization and conduct to a 40% reduction of processing time against a conventional x86 CPU.

Index Terms— HMI, embedded system, dual-core, parallelism.

1. INTRODUCTION

The deployment of Electric Powered Wheelchairs (EPWs) has been increased rapidly for better quality of life for handicapped and elderly people over the last 20 years [1]. However, as many disabled people have problems handling current assistive robots and rehabilitation devices that use traditional manual command such as joystick and/or keyboard, a need of more advanced hands-free HMI is required. Wherefore, alternative HMI methods for wheelchair operation have been studied like head gesture [2], voice recognition [3], tongue movement [4] and biopotential signals interface [5, 6]. Biopotential signals such as Electrooculargraphic (EOG), electromyographic (EMG) and electroencephalographic (EEG) are also deployed to control such kind of smart service robot [7–9]. In the wheelchairs field, to ameliorate the mobility of disabled persons, a need for systems that ensure users safe driving conditions with a fast response (notably in dangerous situations) is required. For this purpose, two conditions must be met: 1) the use of an optimized and efficient control algorithm and 2) the use of a very high performance computing units. The existing implementation of EPW command mostly relied on Personal Computer (PC) platform or on embedded platform. Many of researchers implement their wheelchair command techniques on PC [2, 7] since it can adequately process complex and richness HMI applications and provide an easy programming framework. While, despite the continuous improvement of semi-conductor technology and complex computation resources (DSP, FPGA, multicors architecture), the relevant studies focusing on embedded implementations of wheelchair command are too simple. In [3, 10], the microprocessors and DSP are used to carry out wheelchair control operations. But the major drawbacks of such these architectures are: the lack of design portability and sometimes the necessity to add one CPU to the DSP (graphical interface, OS support...). In [11, 12], the author proposed an FPGA realization of wheelchair command based respectively on EEG signals and Fuzzy Logic Controller. However, the design of a FPGA is very complex since it requires custom complex hardware accelerators.

In the literature and in our knowledges, few studies address the problem of executing complex control techniques of smart wheelchair on embedded systems. In this paper, we present a complex command technique, based on image processing algorithms, developed on embedded target since PC has a lack of autonomy and consumes more energy. Moreover, image processing algorithms require complex data processing across complex environments. Signal/Image processing applications have different parallelism types like data and thread parallelism. An application is generally composed of multiple tasks which may be simultaneously performed on different cores. Consequently, it is worthwhile to explore software optimizations and parallelism that can be easily implemented on popular and low cost embedded multi-processor platforms like Zynq, Tegra 2, etc. Our contribution thus is twofold: 1) we propose a novel HMI for people with hands impairments based on human face. Human face has a large range of facial expressions (emotion, gaze tracking, mouth movements, eye movements, etc.) that can be conceivably used for people interacting with a computer. This intelligent module developed for the autonomous guid-

978-0-9928626-3-3/15/$31.00 ©2015 IEEE

ance of the chair to improve the mobility and guarantee a safe functioning relies on determining the orientation of the face and the eye blinks using image processing algorithms. 2) we implement this algorithm on an embedded multi-core architecture based on SoC design with moderate consumption of electrical energy by exploiting both instruction-level parallelism and data-level parallelism to reduce processing time.

The reminder of this paper is organized as follows. Section 2 explains the concept of the HMI used to interact with the handicapped and details about the software implementation are presented in section 3. In sections 4 and 5, we present the HMI parallelization techniques. Section 6 reports the experimental results, which were obtained from implementing the proposed algorithm on the target multiprocessor architecture. Finally, section 7 draws some conclusions.

2. HUMAN MACHINE INTERFACE

This paper proposes a bi-modal HMI for hands-free control of an EPW by using face and eyes. The human machine interface is in charge for reading characteristics, classifying selected face and eyes movements and mapping the classified movement patterns into four wheelchair control commands: "GO", "Turn Right", "Turn Left" and "Stop", that emulates the four direction of the joystick control. The user is sitting in front of the camera which is equipped with a light source to make the detection robust and effective in real environment (illuminations and crowded problems). Figure 1 describes the process to recognize user's intention.

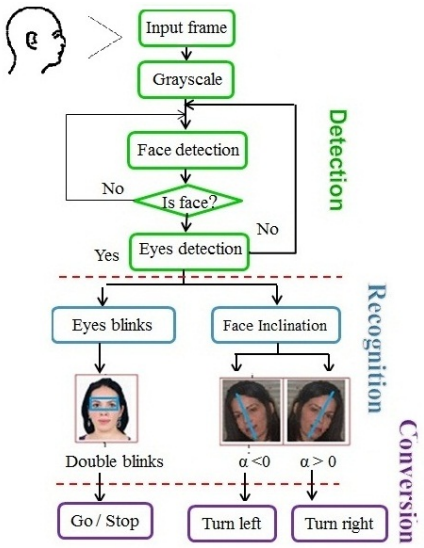

Fig. 1. Process to recognize user's intention.

The process is conducted in three steps: detection, recognition system and conversion. More details about the process are presented in [13]. The proposed system can differentiate between the voluntary and involuntary behaviour to preclude potential accidents, when the user naturally rotates his head to look around during the navigation process. In addition, the suggested mechanism demands a minimal motion from the user, which makes it more convenient and suitable for persons with severe disabilities compared to conventional methods.

3. SOFTWARE IMPLEMENTATION DETAILS

Face and eyes detections are the significant steps for our command algorithm: determination of face inclination and eyes blinks. The detection algorithms are performed based on the Viola-Jones procedure [14].

3.1. Viola and Jones algorithm

Figure 2 demonstrates the viola and Jones principle.

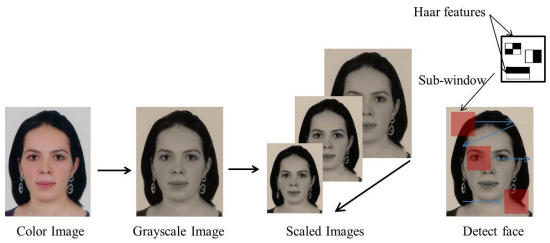

Fig. 2. Viola and Jones procedure.

The image is initially loaded, gray-scaled and scanned with a sub window. In each sub window within the image, a cascade classifier is running to detect the presence of face/eye or not. This procedure is repeated for different image scales until the sub window size surpasses the scaled image size.

3.2. Implementation using OpenCV Library

The software prototype was developed in C++ using OpenCV Library [15]. Figure 3 presents the call graph of the whole algorithm where different functions are called.

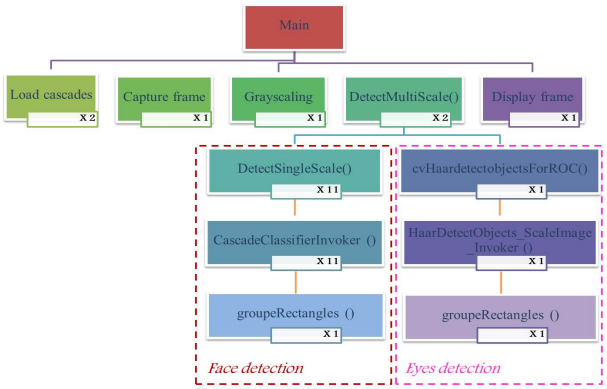

Fig. 3. The call graph.

As shown in Fig. 3, the bulk of the algorithm is based on detectMultiScale () function in the CascadeClassifier class in the objdetect module of OpenCV. This function was called twice ("x" means the call number of the functions in the whole algorithm): one for face detection which use Local Binary Patterns (LBP) cascade classifiers and the other for eyes detection which use HAAR cascade classifiers.

For face detection, in a loop which scales the image several times by the growing factor, the detectSingleScale() function is invoked on the scaled image. This function scans all areas in the image and performs the cascaded classifiers on each possible sub-window. Once all faces were found, the groupRectangles() function was invoked to group all similar face rectangles found.

For eyes detection, all detection stuff is done within cvHaardetectobjectsForROC () function which can be found in Haar class in objdetect module of OpenCV. In this function, a loop that computes the re-scaled versions of the original image with a re-size factor is done. Within this loop, the HaarDetectObjects_ScaleImage_Invoker () is called to start the eye detection. After detection on all scales has been done, a grouping of similar rectangles (groupRectangles() function) starts to fuse the multiple detections.

4. OVERVIEW OF MULTIPROCESSOR ARCHITECTURE

Parallel computing and multi-core architectures became essential to attaining great performance in actual and future computing systems. The parallel memory architecture targeted in this work is a Symmetric Multi-Processing (SMP) architecture. SMP implies multiprocessor hardware description in which two or more identical CPU are linked to a single shared memory and are controlled by a single operating system (OS) instance which deals with all processors equally.

In parallel programming, the easiest way to implement a parallel application is exploiting multithreading. It's still more gratifying to note that nearly all programming libraries, that habitually offer multithreading facilities, give a mechanism to automatically run multiple threads in different processors. Three user friendly programming models are available nowadays: POSIX threads (Pthreads), Open Multi-Processing (OpenMP) and the Threading Building Blocks (TBB).

5. PARALLELIZATION OF THE WHEELCHAIR HMI APPLICATION

To reduce processing time, two parallelism levels are exploited. The first one is instruction (threads) level parallelism using a multi-threading library and the second level is data parallelism in which an SIMD specific unit is exploited in the target architecture.

To determine which parts of the code must be optimized and parallelized, we have made a profiling of our algorithm. Through profiling, supercomputing functions and instructions within these functions which are usually performed can be identified. Figure 4 illustrates the profiling results of a $640 * 480$ image made on the Pandaboard-ES.

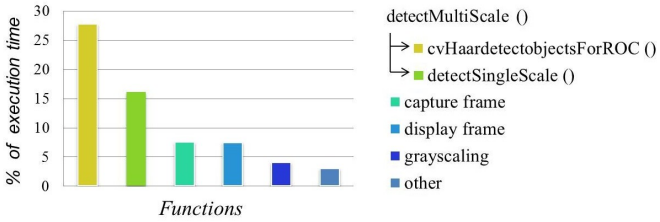

Fig. 4. The profiling results of a $640 * 480$ image resolution.

The obtained profiling results illustrate that there are two great candidates functions for multi-threading optimization:

- For face detection: the detectMultiScale() and within it detectSingleScale().
- For eyes detection: the detectMultiScale() and within it cvHaardetectobjectsForROC() .

5.1. Instructions-level parallelism

5.1.1. Parallelizing the "scanning" loop:

As mentioned before, the detectSingleScale() and HaarDetectObjects_ScaleImage_Invoker () functions perform the sub-windows scanning in the image. This parallelization was possible since each sub-window in the image may be independently treated by the classifier. Therefore, by creating multiple copies of the classifier, it's quite easy to parallelize the scanning. In fact, each thread is allocated a set of sub-windows of the same lines to treat. This procedure may be regarded as embarrassingly parallel , since no shared memory or conflicts happen between threads.

The parallel design architecture is shown in figure 5.

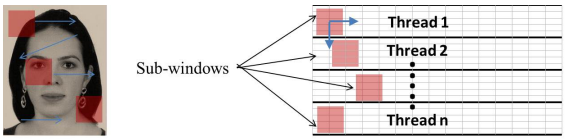

Fig. 5. The parallelization architecture in the scanning loop.

5.1.2. Parallelizing the "factor" loop:

Parallelism may also be carried out on a coarser scale in detectMultiScale() and cvHaardetectobjectForROC() functions where the image resizing / scaling is performed. Most of the

code in these functions is in a wide loop that resizes, in each iteration, the image by a designed factor. As there are no data changed in the image and no data dependency for the calculations of different scaled images, a possible optimization is the parallelization of the loop factor. A potential solution could be a pool of threads with factors attributed to each thread. Each thread should, thereby, resize/scale the image according to its particular factor and transfer the image information to corresponding functions to perform scaling. The growing-factor is determined according to equation (1):

$$growing\_factor = growing\_factor * scalefactor \quad (1)$$

where the scalefactor is one of parameters defined in detectMultiScale() and cvHaardetectobjectForROC() functions.

Figure 6 presents the distribution scheme, that assigns each factor to a different thread as the factor increase.

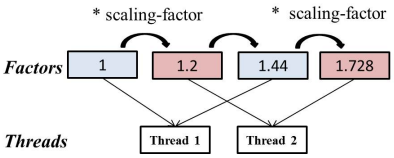

Fig. 6. The parallelization architecture for the factor loop.

5.2. Data-level parallelism

In this level, with a single instruction at a given moment, the CPU processes multiple data points simultaneously. This is a way of getting the processor doing more work with fewer instructions. Eventually this means that the program code is increasingly compact and often functions faster.

Nearly all image processing algorithms utilize grayscale images as input one. But nearly all hardware video sources supply pictures in RGB formats. Thus grayscale conversion is a highly popular transformation. To this end, in this paper, we have used Advanced SIMD instructions to get significant performance gain of RGB to GRAY conversion.

6. EXPERIMENTAL RESULTS

The experiments were conducted to test the benefit of the performance of parallel solutions discussed in section 5.

6.1. Pandaboard-ES overview

The mobile platform on which the treatment was applied is the TI OMAP4 SoC. The PandaBoard-ES is a low cost and an open mobile software development platform enabling easy, rapid and extremely extensible development. It is based on the OMAP4460 SoC that highlights a dual-core ARM Cortex-A9 MPCore based on SMP clocked at 1.2GHz.

6.2. Software specification

The operating system where the Pandaboard-ES is running is the Ubuntu 12.04. The prototype is implemented using C++ language. The profiling tool used to measure the average processing time in each function is the valgrind. The OpenMP library, ported to a number of different platforms, was used to perform the thread creation and synchronization.

6.3. Results

The effectiveness of the proposed HMI and detection results are presented in [13].

Since, the Pandaboard-ES has two Cortex A9 CPUs, we can run our functions, which are candidates for multi-threading optimization, in two threads. Figure 7 shows the performance results when the treatment is made in parallel.

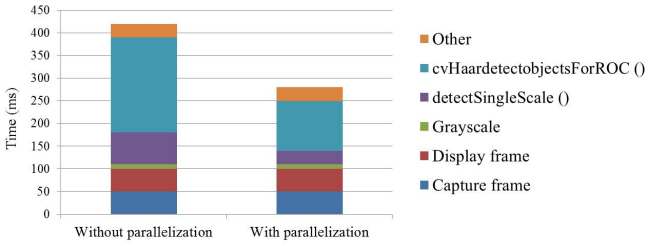

Fig. 7. The performance results of the parallel implementation.

It was shown an interesting speed-up (about 1.54 times faster than the initial sequential implementation).

The ARM Cortex A9 has a SIMD engine called NEON used to accelerate media applications. As grayscaling has SIMD features, we decide to accelerate its execution on the NEON engine. SIMD technology, as mentioned in section 5, can process multiple data with a single instruction call, conserving time for further computations. For that, we have implemented an assembler "NEON version" to accelerate color conversion and we have treat eight pixels at one time. The findings results give a great performance gain but we assessed to go further and perform parallel calculation "Parallel NEON version" using two threads.

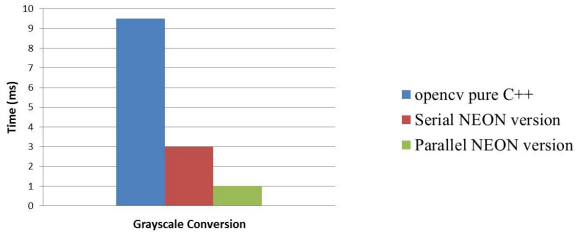

Fig. 8. Neon optimization.

We can see through Fig. 8 that Parallel NEON version runs 3 times faster than Serial NEON version and 9 times faster than purely C++ implementation from OpenCV (cvtColor() function).

Table 1 summarizes all obtained results. A reduction of 40% is obtained versus the mono core execution time.

Table 1. Summarized results.

Device	OMAP 4460: Pandaboard-ES
Timing results (ms)	
1 core: 1.2 GHz	475
2 cores: each 1.2 GHz	421
2 cores + OpenMP + Neon	273
Gain	
Acceleration factor	1.739

Table 2 indicates that the obtained time for the embedded system implementation is not sufficiently close to that obtained on PC running an Intel core 2 Duo T5800 CPU. However, with the embedded platform, we have the possibility to 1) improve timing results, 2) exceed those obtained on PC and 3) satisfy the energy consumption criterion by combining the two Cortex A9 with the accelerated engines available on the SoC such as the DSP or the GPU.

Table 2. Time/Power comparison.

Measures	PC	Embedded System
Time (ms)	150	273
Thermal Design Power (Watt) [16, 17]	35	0.6

7. CONCLUSION

The paper describes the architecture of an HMI for the control of an EPW based on face inclinations and eye blinking. The implementation on dual core processor exploits the use of threads for achieving greater efficiency and the use of SIMD engine called NEON to accelerate image color conversion. The proposed approach has proven to be effective since a speed up of 1.73x is obtained. We are planning to perform further studies on maximizing the power of ARMs with DSP/GPU accelerator since the achieved reduction in time is not sufficient yet.

REFERENCES

[1] Nihei M., Fujie M.G.: Proposal for a New Development Methodology for Assistive Technology Based on a Psychological Model of Elderly People. In: Assistive Technologies. USA, (2012)

[2] Jia P., Hu H., Lu T., Yuan K.: Head gesture recognition for hands-free control of an intelligent wheelchair. Int. J. Ind. Rob. vol. 34, pp. 60-68, (2007)

[3] Wallam F., Asif M.: Dynamic finger movement tracking and voice commands based smart wheelchair. In: Int. J. of Computer and Electrical Engineering, (2011)

[4] Kim J., Park H., Bruce J., Sutton E., Rowles D., Pucci D., Holbrook J., Minocha J., Nardone B., West D., Laumann A., Roth E., Jones M., Veledar E., Ghovanloo M.: The Tongue Enables Computer and Wheelchair Control for People with Spinal Cord Injury. In: Science Translational Medicine, Vol. 5, Issue 213, (2013)

[5] Wei L., Hu H.: A hybrid human-machine interface for hands-free control of an intelligent wheelchair. In: Int. J. of Mechatron. Autom. vol. 1. pp. 97-111, (2011)

[6] Maskeliunas R., Simutis R.: Multimodal wheelchair control for the paralyzed people. Electron. Electr. Eng. vol. 111, pp. 81-84, (2011)

[7] Ben Taher F., Ben Amor N., Jallouli M.: EEG control of an electric wheelchair for disabled persons. In: Int. conf. on Individual and Collective Behaviors in Robotics (ICBR), pp. 27-32, Sousse, 15-17 Dec (2013)

[8] Santana A., Yang C.: Robotic control using physiological EMG and EEG signals. In: Advances in Autonome. Robots. pp. 449-450, (2012)

[9] Brando A.S., Felix L.B., Bastos-Filho T.F., Sarcinelli-Filho M.: Controlling devices using biological signals. In: Int. J. Advanced. Robotic. Systems. pp. 22-33, (2011)

[10] Lian J. Hu., Qing H. G., Hong S., Gui X. Ch.: Research of Electric Wheelchair Control System Based on DSP. In: Applied Mechanics and Materials. pp. 1122-1126, (2011)

[11] Jzau S. L., Sun M. H.: An FPGA-Based Brain-Computer Interface for Wireless Electric Wheelchairs. In: Applied Mechanics and Materials. vol. 284-287, pp. 1616-1621, January (2013)

[12] Islam Md. Sh., Bhuyan M. S., Sawal H. Md. Ali.: FPGA Realization of a Fuzzy Based Wheelchair Controller. In: Research J. of Applied Sciences. vol. 8, Issue. 9, pp. 442-448, (2013)

[13] Ghorbel A., Ben Amor N., Jallouli M.: An embedded real-time hands free control of an electrical wheelchair. In: IEEE Visual Communications and Image Processing (IEEE VCIP) Conference. pp. 56-59, December (2014)

[14] Viola P., Jones M.: Robust Real-Time Face Detection. In: Int. J. of Computer Vision. pp. 137-154, (2004)

[15] Krejcar O.: Handicapped People Virtual Keyboard Controlled by Head Motion Detection. Symposium on Smart Environments at workplaces, (2010)

[16] http://ark.intel.com/products/35581

[17] http://www.notebookcheck.net/OMAP-4460-Notebook-Processor.83630.0.html