TOWARDS ACCELERATING LOW-LEVEL VISION IN ROBOTICS

Gianna Rodrigues de Araújo[1]
Julio Cesar Paulino de melo[1]
José Alberto Nicolau de Oliveira[1]
Luiz Marcos Garcia Gonçalves[1]

[1]Universidade Federal do Rio Grande do Norte

ABSTRACT

This paper proposes a solution to speed up computer vision algorithms generally used in real-time robotics applications. The proposed solution integrates hardware and software components in a platform based on the Nios II processor. Hardware speed and parallelism are added to the flexibility and simple structure of event based programs simplifying the system architecture, making the processing faster.

1. INTRODUCTION

A challenge that still remains in the robotics area is how to construct a system to react in real time to visual stimulus. One of the main problems is related to the use of traditional computer vision algorithms that are known to be computationally expensive. That is, operations like image filtering or even image processing by mathematical morphology operators may take too much time in a sequential processor consequently delaying other possible activities of the whole system. A solution would be to use small images to compensate the time of the image processing techniques, but this may be useless for some applications as recognition.

In order to cut down most of the time spent in the image processing algorithms many people implement image processing algorithms in specific hardware architectures, using high parallelism hardware devices, thus obtaining good results as showed in the Sonic Architecture [2], the Multiplier Array for Video Processing [3] and other works that will be discussed further in this paper. These architectures are known to be expensive.

In this context we propose an approach that uses hardware processing and event based programming on a platform oriented architecture. The idea is to simplifying the vision based system while, at the same time, accelerating some parts of the used algorithms.

This paper is organized as follows. Next Section (2) discusses the works that are most related to ours. In Section 3, we introduce the concept of a platform based system that is the basis of our implementations. In Section 4, we discuss about computer vision algorithms and systems, and in Section 5 we talk about our system architecture. Experiments, results and a final discussion is presented in the remaining Sections.

2. RELATED WORK

Computer vision cover many problems of the Robotics area, so there are many implemented works that are related with our job, specially using hardware systems in order to accelerate the image operations.

The Sonic Architecture is a complex system that allows the user to configure hardware in real time providing faster image filtering, multiplying, accumulating and other operations [2]. The approach is similar to ours because a hardware/software system is developed to speed up the digital image processing. The work is not applied in computer vision, but it could, turning the image processing faster and saving time to other image processing methods.

On the other hand, Haynes and [3] implements a hardware only solution to speed up multiplication of images, called the “Multiplier Array for Video Processing”. A flexible bit array multiplier is implemented that can multiply 25 times faster than a software solution.

A more applicable work is proposed by Souza and Bianchi [4] that implement hardware processing techniques using the hardware description language VHDL. The main purpose of the work is to speed up the robot detection in robot soccer games. For that, some computer vision algorithms or image processing filters are implemented in VHDL thus improving the processing time in further steps of the soccer strategy.

The last two works [03, 04] were developed aiming to solve the same type of problem and are mainly implemented in software. The Sony Architecture, plus some other works [05, 06] are systems used to recognize and identify vehicle plate numbers. The main relation of these works with ours is the use of image processing algorithms to find certain data in an image. Of course the last cited works do not to deal with real time requirements. But, if a hardware device would be used to detect the location of the interest data, the image processing phase would be simpler.

3. PLATFORM BASED SYSTEMS

In the reconfigurable hardware research area, it is common to use Register Transfer Level (RTL) to design state machines and other constructions that are direct related with hardware. These kind of platform based systems add to hardware systems a way for them to be used in more complex systems while being controlled by a single processor.

In general, platform based systems do not have only specific hardware components in it. They have one or more processors and a source code that is used to control the main jobs of the specific hardware. This often provides more flexibility and configurability to systems that have many parameters that can be changed in time of execution. In this case, we can set those parameters via a software interface, which is better than a hardware one.

In fact, systems that are platform based are slower because they need to interact with processors via a shared bus interface. But, in many situations and cases, the flexibility of this kind of system may be needed, like in computer vision where there are many application specific software/hardware to be used. By using classic hardware design we would need to integrate all hardware systems via other hardware interfaces, while if using a platform based system we need to implement only the communication of the other system parameters via a common bus interface. This turns the whole system more scalable and reconfigurable, because all extern components are isolated of each other.

As said above, platform based systems must have a processor that controls all the communications between the components. In our implementations we used the Nios II processor developed by the Altera Company.

4. COMPUTER VISION ALGORITHMS

Computer Vision is relatively recent area of Computer Science that, in resume, applies image processing techniques to provide a way to extract important information from images or videos. Results of these operations are generally applied in other systems. Common computer vision techniques are developed aiming at providing support to bigger systems that need some information about images.

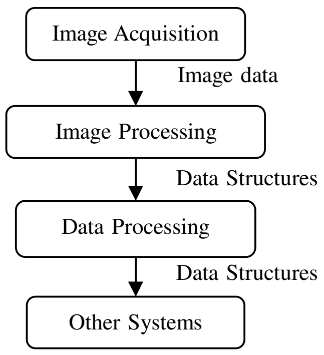

Figure 1 - Common structure of computer vision systems

The general structure of traditional computer vision algorithms is shown in Figure 2. First, in the image acquisition phase, images of the working environment are acquired and transformed into digital data. Next, the image is processed through some image processing techniques that extract data from the image to feed data structures that are supposed to be more simply to the application.

Then, resulting data is processed again through some other methods like neural networks, reinforcement learning, or some exact methods, no matter, etc. Finally the result is used in a specific block that does the real job of the system.

Some solutions as depicted above act in the image acquisition phase using hardware solutions to provide better images so the image processing phase, that, consequently may need less filtering than those required by many low-level vision algorithms.

Clearly, a big part of the computational delay is concerned to the image processing phase. For example, images with 640x480 pixels may take several milliseconds to be filtered or analyzed. This may be useless in some applications. For instance, consider a tracking system that need to follow an object in an image. Every time the system needs to check the whole image searching for the object. If it is found, the system can initialize the tracking job.

The problem is that the system needs to check the whole image in less than 33 ms, in order to be able to analyze at 30 fps, which is a common acquisition rate, until it finds the tracked object. And then use its position and velocity to optimize the searching area of the image. This slows down the whole system that could be performing some other job until the image does not contain the object.

5. EVENT BASED COMPUTER VISION ALGORITHMS

In Section 4 we discussed about the way that most vision systems are implemented and showed some problems with the traditional used vision architecture. On the one hand we can rely in this simple model. But, on the other hand, we can implement it with another point of view, which is less expensive and that does not block the system during the execution.

To better understand the implementation of this proposed approach, let us think about the keyboard event handling system that is shown in the **Figure 3**. The main processor is executing some other job when a certain interruption is activated warning it that a keyboard event arrived in the event queue. So, the processor calls a pre configured routine to handle this event. This routine can now get the event parameters and pass then to other applications that might be waiting for that kind of event.

In a similar approach, we can see the image processing system like an event source that can warn the processor about the occurrence of some kind of event. In this way, we can modularize part of the image processing block that was explained in Section 4 and reduce the computational cost of the whole computer vision system.

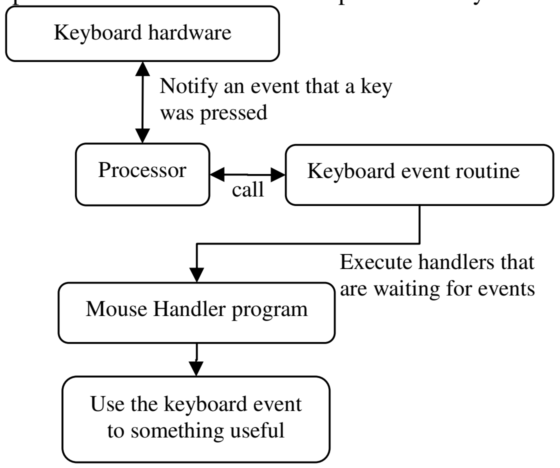
Keyboard hardware
Notify an event that a key was pressed
Processor
call
Keyboard event routine
Execute handlers that are waiting for events
Mouse Handler program
Use the keyboard event to something useful

Figure 2 - Common keyboard event handling

Figure 4 shows how the event system would become with a vision module that can process events in images and warn the processor that some events occurred. With our approach, the computer vision systems can become simpler and faster because a parallel hardware is processing the image in order to generate events, and only when such events occur the image processing methods are called.

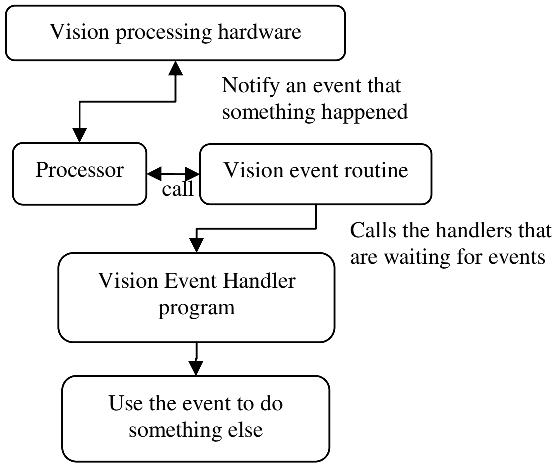
Vision processing hardware
Notify an event that something happened
Processor
call
Vision event routine
Calls the handlers that are waiting for events
Vision Event Handler program
Use the event to do something else

Figure 3 - Event based vision system.

A problem with this approach is the specialization of the system, that is, the definition of image events can change from one application to another. This means that what is considered an event to a tracking system, that has to follow a ball in an image, may not be considered to an automated robot that is programmed to avoid obstacles. This means that the system that is processing the images needs to be reconfigured based on the application that it is being used for. By using platform designed systems we found a good solution for this problem. Once we are talking about integrated hardware/software solutions, it is easy for the application to set some of the hardware parameters before starting the work making the system more flexible an applicable to many purposes, including thus both cases.

The implementation of an event module into a vision processing system can directly affect the model showed in Figure 2. This happens because the model on this figure has to block the processor during the time that the image processing block is working. With our approach, this does not happen because many of the processing is done in a parallel hardware module. This has lead us to another model that is shown in Figure 5.

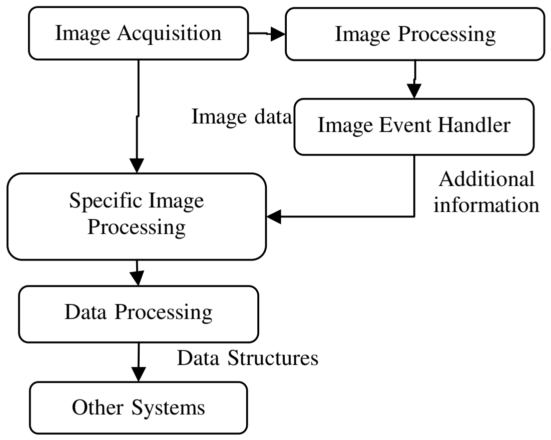

Figure 4 - Result of adding event based vision hardware

The hardware block is presented as the old image processing block. The difference is that it does not need to block the processor in order to execute. Another block that is connected with the image acquisition block, called Specific Image Processing, was added in order to turn the model more generic. This block is necessary in the case of new or experimental image processing algorithms that may not be implemented in the hardware module and need to execute. This may be the worst case of the system when it works similar to the first model, but our model is faster because it saves a lot of computing time in the hardware modules.

6. EXPERIMENTS AND RESULTS

In order to initially validate our idea, we have observed the behavior of the platform oriented systems over our architecture. We devised a general filtering system that is developed at the DSP Builder as shown in Figure 6.

We used the switches of the development platform and one of the buttons, respectively, to simulate the address given by the hardware system bus. By attributing a constant value to the registers reading and by using a multiplexer to verify their contents, we could test, in simulation, that our system behaves as expected.

We have further incorporated the system together with the Nios using the SOPC Builder, but we have had some problems while writing in the system bus. Apparently, the reading functions are working correctly however we could not write in some of the internal components.

Figure 7 shows the component integrated with the system bus. The bus inputs at the left and right represent the writing and reading interfaces, respectively.

7. CONCLUSION

We introduce a new idea to perform low level vision by joining the hardware parallelism and software in a platform generally used for other purposes based on the Nios II processor. The novelty is just the event handling block that was added to the traditional vision structure.

Low level vision is, in general, the bottleneck of robotics vision that the approach depicted here can help performing much faster than traditional algorithms. Our approach uses hardware parallelism when necessary and can be programmable in order to allow software operations to be done. By joining both hardware and software and adding the event handling block we could devise a useful architecture that has proven, in simulation, to be applicable in problems where real time is a requirement.

With the development of platform oriented systems we get a high degree of flexibility making it possible to develop solutions based on hardware and software at the same time. With this, we can specify a model that has several advantages over the traditional paradigms of computer vision.

The main advantage is for sure the use of a low expensive hardware. In fact, our work is still in the beginning because we have had to observe the behavior of these systems together with our platform. Besides, some simulation results could validate it. We are still finalizing the hardware implementation and resolving problems with the system bus in order to try a more complete implementation. This will be the basis for other works running at our Lab in the Robotics Vision or Graphics Processing subjects involving integrated hardware and software.

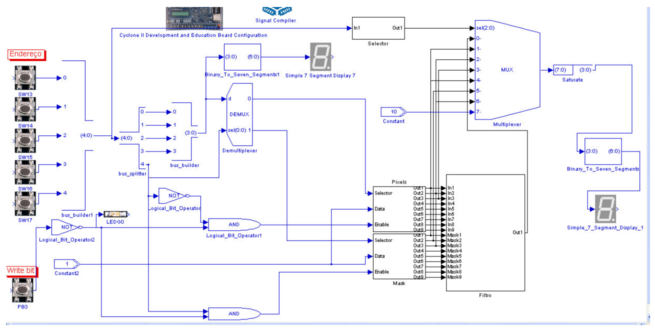

Figure 6 – General filter developed in the DSP Builder

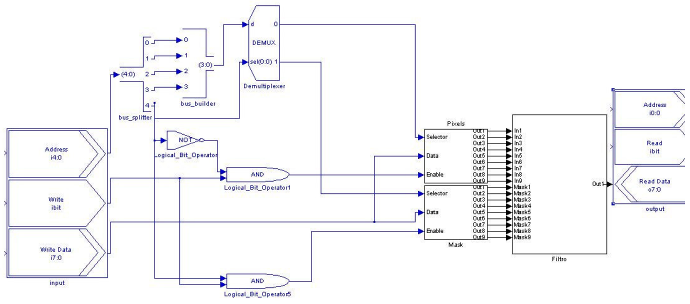

Figure 7 – Integration between system bus and developed component.

8. REFERENCES

[01] NIOS II Processor, altera website: http://www.altera.com/products/ip/processors/nios2/ni2-index.html.

[02] Haynes, S.D., Stone, J., Cheung, P. Y. K., Luk, W. Video image processing with the Sonic architecture. In: Computer , vol.33, no.4, pp.50-57, Apr 2000.

[03] Haynes, S.D., Cheung, P.Y.K., A Reconfigurable Multiplier Array For Video Image Processing Tasks, Suitable For Embedding In An FPGA Structure. In: FPGAs for Custom Computing Machines, 1998. Proceedings. IEEE Symposium on , vol., no., pp.226-234, 15-17 Apr 1998.

[04] Souza, P. V., Bianchi, R. A. C., Implementando Algoritmos de Visão Computacional em VHDL. In: Revista Pesquisa e Tecnologia FEI, São Bernardo do Campo, SP, v. 22, p. 2-8, 2002.

[05] Guingo, B. C., Rodrigues, R. J., Thomé, A. C. G., Técnicas de Segmentação de Imagens, Extração de Características e Reconhecimento de Caracteres de Placas de Veículos. In: VII Simpósio de Informática e II Mostra Regional de Software Acadêmico, 2002, Uruguaiana-RS.

[06] Campos, T.; Bampi, S.; Susin, A. Sistema de Identificação de Placas por Processamento Automático de imagens. Disponível em http://www.iberchip.org/VII/cdnav/pdf/49.pdf.