Real Time Hardware Vision System Design

Pedro Cobos Arribas
Departamento Sistemas Electrónicos y de Control
Universidad Politécnica de Madrid.
E.U.I.T.Telecomunicación, Km 7, Ctra Valencia, 28031 (Madrid)
España

Abstract: This paper describes an electronic system designed for real-time vision applications. The system will allow the development of low-complexity tasks, such as obstacle detection, at very high frequencies. This makes it useful for robotic applications and control systems in real time (up to 25 images per second).
Its design has taken into account low-cost and flexible modular solutions that can be applied in research laboratories or used with engineering students in order to make real time image process practices.

Key-Words: Vision, Real time processing, optical flow, robot systems, Hardware systems, FPGA.

1 Introduction

This paper describes the materialization of the hardware section of my PhD [1]. The objective of this work was the development of hardware solutions to be able to calculate –in real time– the optical flow of real images. The second objective was to generate their optical invariants by building the components of a system that could be used as a vision module in a robot. These objectives will be reached through the different tests that will be carried out in the next stage.
These tests will include the calculation of the time to impact from the data provided by the optical flow, the log-polar transformation of input images, and the calculation of the optical flow.
Alongside these experiments, a methodology will be developed in a vision laboratory. This will measure the theoretical values of the algorithm simulation by comparing in real time the values obtained by applying the images to the hardware system.
The reason for the development of this system is the lack of one that proves the validity of the algorithms without having to develop a different testing system for each case, as it is happening at the moment. This item will be described in next section.

1.1 Vision System

The different blocks of the active vision system can be seen in Fig. 1.
The actions required to move the camera to the desired region are calculated from the data

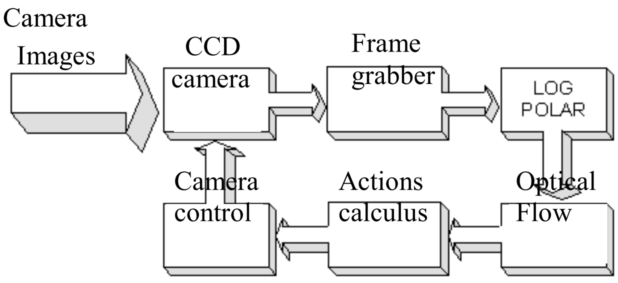

Fig. 1 Blocks diagram of the vision system. The frame grabber is not necessary at the moment since we are using CMOS digital cameras.

compress the data, and finally the extraction of the optical flow which contains relevant information for the system.

2 Previous work

In [2-5], [6], some implementations of these blocks are described. The structure of all of them is the same (Fig. 2), where an easy interconnection and real time operation are provided. A little hardware block can do its work quickly than a larger one and many of these blocks can operate in a parallel mode if the intermediate results are accessible. The input and output memories, that can be accessed at the same time from the source and the target systems, are the more important parts of this hardware structure. The other element is always a FPGA (or several), that holds the vision algorithm and the memories addressing task. Depending on the algorithm, the complexity of the FPGA design will

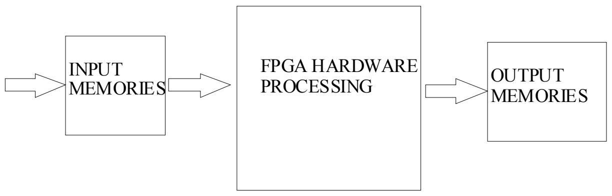

Fig. 2 Hardware data processing structure

change from very hard constraints for the optical flow calculation to a little effort in image transformation algorithms, as in the log-polar

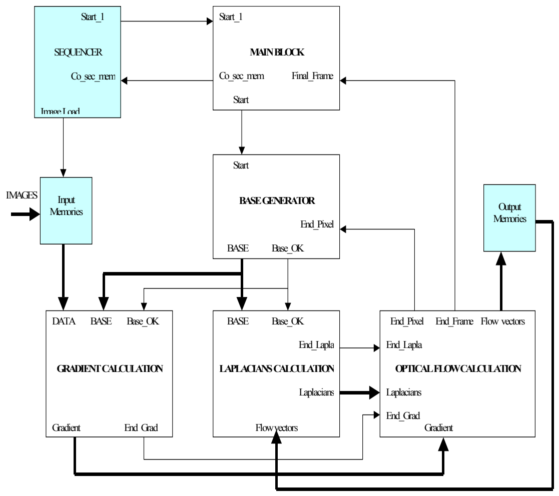

Fig. 3 Blocks diagram of a FPGA implementation of a differential optical flow extraction algorithm.

implementation. At Fig.3 an implementation of the first type is shown and at Fig.4 another of the second type cited. Only at a first look the complexity required can be noted. All the original algorithms, [7-10], have served to develop the hardware approaches.

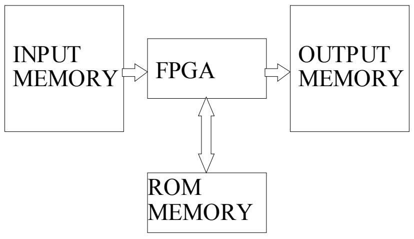

Fig. 4 Blocks diagram of the Log-polar FPGA implementation

Nevertheless, the system structure is the same and if required, several systems could operate at a parallel way.

3 Hardware vision system description

All the described processing takes place in the hardware vision system based on FPGA, represented in Figs. 5, 6, 7 and which will be described next.

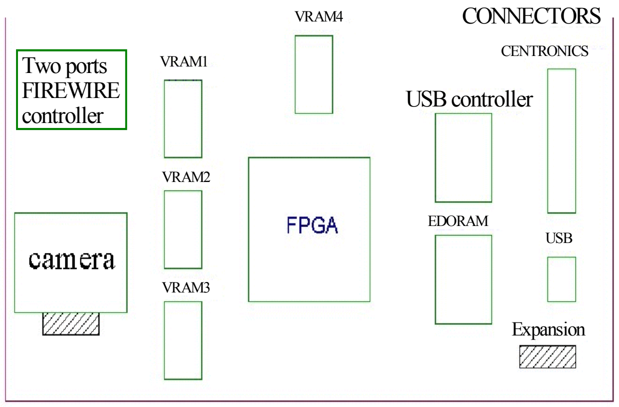

Fig. 5 Components of the hardware vision system

The system components can be modified by using the programmable logic circuit capabilities. For instance, you can choose whether or not to use the above mentioned transformation, which has a higher resolution in the attention region (fovea) and a lower resolution in the peripheral region, as in biological vision systems. You can also choose to store only two images, perform an internal buffer to do an image filtering, etc. Therefore, previous tasks can easily be repeated, as well as new ones, [11-13]. For these reasons we focussed greatly on the flexibility of the system. Also, we have intended to reduce as much as possible the system dimensions so that it can be assembled on real systems. Therefore, we have designed the printed circuit assembly with six faces (two planes for ground and power connections and four for routing signals) and a fabrication class of 6.
The vision system has the following input options:
- A low-cost, monochrome, digital camera (OV7120) [14] with a 640 x 480-pixel resolution, which can be directly connected to the vision board using a 32-pin flat cable female connector.
- An IEEE-1394 camera, which can be connected using a (Firewire) input connector. This is for applications with more features and higher costs. This option allows the interweaving of the vision

system in applications that require the processing of images that reach the camera at high speeds.
- Another identical block output to break down complex processing into several phases. As a result, each system runs in parallel parts of the operations, and a complex algorithm can be completed using

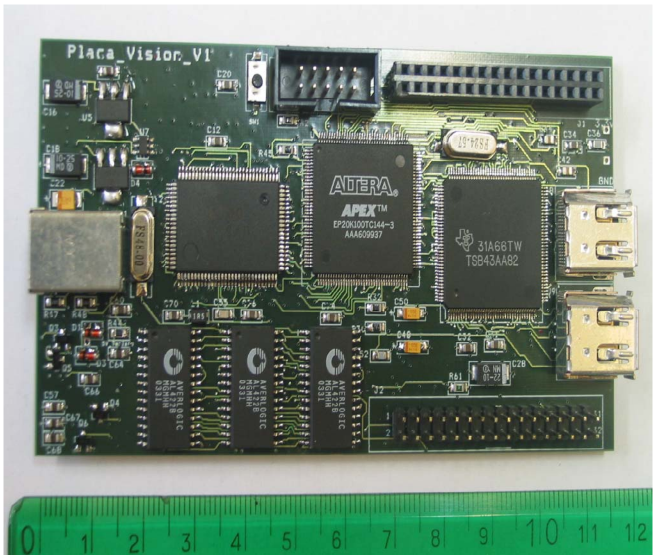

Fig.6 Upper view of the vision system.

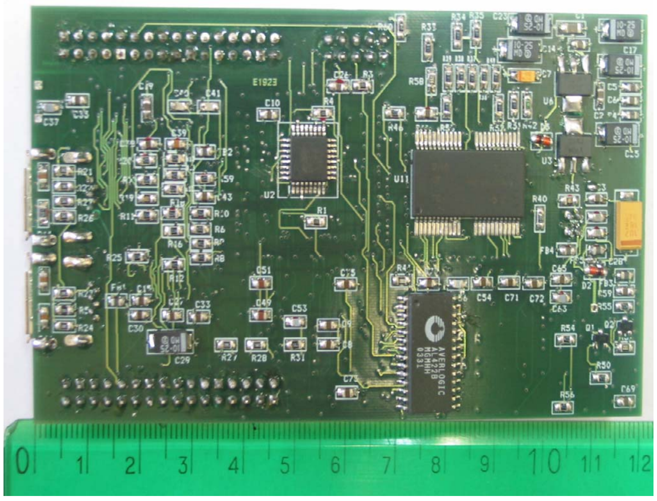

Fig.7 Lower view of the vision system. In order to reduce the dimensions of the printed circuit board, the components have been placed on both sides of the board.

simple blocks, with each block hosted on a vision board.
The above input options can be connected directly to the FPGA. However, they are normally connected to several memories (typically three input memories) in order to store consecutive image information. These memories are frame buffers, that is, memories used to store a full image, of sequential access, especially the AL422 [15], with two stand-alone input/output ports.
This is so that they can be written and read at different speeds. They have a capacity of 3Mbits and therefore the images can be easily stored in different formats.
This information is processed by an FPGA manufactured by ALTERA [16], (model 20k100) which addresses the input memories, performs the required calculations and downloads the results on a dual-port memory. It has the same features as the input memories and is accessible to other systems. The system can be adapted to multiple situations since it is possible to reconfigure the input pins in different ways.

The processing outputs can be obtained from several connectors:
- A flat cable male connector, which is the male version of the female input connector, to be able to connect several systems in parallel so that the system outputs to the connector or the output memory. It is configured with a signal organization identical to the one in the system input.
- A USB connector to monitor the processing output from a PC. To do this, it is necessary to have a specific controller (OV511+) from the same manufacturer as that of the cameras.
- A Firewire connector to read the results from this format. The communications with this standard are performed by a Texas Instruments integrated circuit –TSB3A82– which controls the three levels of the IEE1394 protocol.
Alongside these input/output connections, another 10-pin connector is required for the programming of the FPGA from a PC, through a cable connected to its parallel port. This programming can be stored in a non-volatile configuration memory, included in the system, so that there is no need to re-programme every time the power is shut down.

The system power supply (5V, 3.3V y 2.5V) can be obtained from either the USB connector, Firewire connector or via a connection to a 5V external power supply source.
In Figs 8 and 9, you can see the layout of a camera connected to the vision system, and of two systems operating from a camera. These are examples of the system operational modes.

These can deal with complex issues, by breaking down the tasks so that they can be performed in parallel in several systems with the same structure: the output of the first is the input of the second, and so on, until the last one, where the global result of the processing could be read by a host processor.

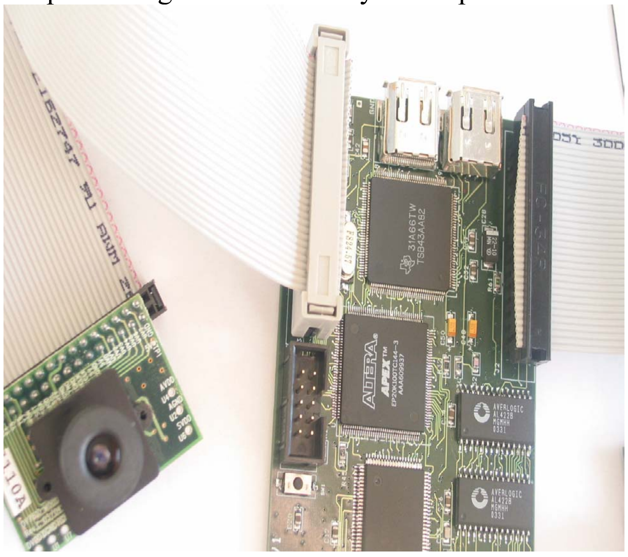

Fig. 8 Detail of the interconnection between the camera and the vision system. In reduced size applications, it can be installed directly onto the board connector.

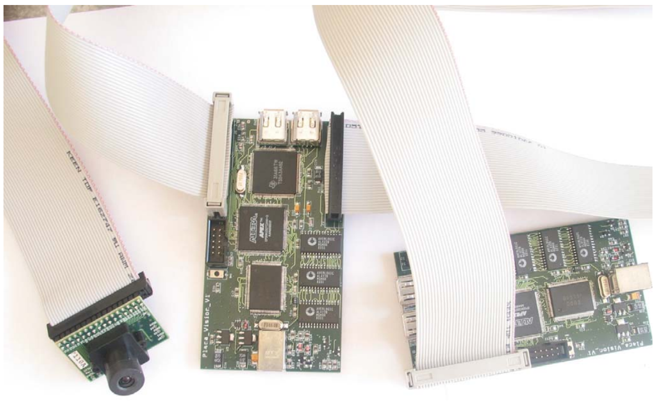

Fig. 9 Detail of the interconnection between two vision systems and a camera. By using flat cable connectors, you can install the modules required to implement the vision algorithm.

4 Conclusion and future developments

The original contribution of this work is the building of a modular hardware architecture, which allows the interconnection of the different modules, whose outputs can be monitored and whose parameters can be programmed by a globally controlled system.

We considered it was necessary to create the correct modular functioning of the vision system components and also the possibility of being controlled and monitored by a higher-level hierarchy system. Even if we were incapable of obtaining results in real time, this system could track and adjust each components operation. Therefore, all the modules have a dual-port output memory, which can be read by the adjacent module and by other hardware that requires the information it generates. Normally this is the PC or workstation which, from the visual information, can calculate new positions for the camera or generate the required changes to adapt to a new situation of the system.

For this last objective, the modules must have their control/reconfiguration inputs accessible.

For example the number of iterations and the size of the window (for the algorithms calculating the optical flow through gradient and correlation techniques respectively). If there has been a reconfiguration, it will be necessary to wait for a period of time before obtaining the correct results. Likewise, when the system starts working, there will be a latency period in which the images are loaded into the input memories and all the blocks operating in parallel have their output data ready.

In my opinion, which has followers and detractors among researchers in the vision field –as proved by the critiques or good words received in the different international conferences I have attended–, the hardware implementations of vision algorithms will be very important in real applications. This is due to the possibility of having large complex ASIC, and larger and faster FPGA, together with the gradual sophistication and increase of the capacity of the all-purpose processors or the DSP processors. The research done intends to cover an area of the important developments in hardware for artificial vision.

Therefore, this research does not intend to be a new application for artificial vision, but a hardware architecture that allows the implementation of vision algorithms by trying to generate vision systems that include and leverage different hardware elements, as real-time special solutions. These will complement the typically programmed systems.

During this research, we have found that many of the techniques based on the calculation of the optical flow, like the ones used here, are very limited in real-time applications, where it is necessary to reduce the dependency on the restrictions derived from the projection process at the image level. However, by reaching the processing speeds indicated and using elements accessible to any researcher (such as the FPGA) the work will be relevant. This is especially true in robotic applications where the vision is used by the robots to avoid obstacles, follow paths or to perform any other required task. Another application where vision systems could be key is in the development of help systems for driving vehicles to avoid collisions with other vehicles or other types of road traffic accidents.

The future development of this investigation can be summarized in three points:

- Inclusion of the developed vision module in a robotic head with motors to control the camera(s) with three freedom degrees of movement, as well as to generate saccadic movements, nystagmus and other biological behaviour to be able to study in detail the effects on the images. These movements set the camera on the desired region, so that more information can be obtained from the scene, and the volume of data required for the processing can be reduced.
- Use of the vision module on motorized vehicles so that their movement can be controlled according to the visual information received.
- Development of other solutions that include digital signal processors (DSP) and programmable logic devices in order to create other vision applications. This new hardware system would perform functions that require intensive-data processing with one or several FPGAs, and mathematical functions, or data-processing, with DSP processors. For example, the LSP [17] system could be used since it allows faster performance implementations by being able to be connected directly to the applications developed in Matlab or Simulink, which are currently in common use.

Many of the future results depend greatly on the electronic technology evolution, which in coming years will be able to provide integration levels, which were inconceivable in recent times, including microprocessors with a high-level parallelism, and with high-capacity and highspeed FPGAs. We believe that with this fast calculation potential, vision algorithms and dedicated hardware will be developed and improved so that tasks that are apparently simple (such as obtaining optical flow to be able to generate the structure of an image from movement data) stop being an extremely complex issue [18], which vision scientists have been investigating in the last 25 years.

We hope that the hardware solutions provided in this paper are helpful in our scientific effort to find an answer to a problem that nature solved a long time ago.

References:

[1] Cobos, Pedro. “Arquitectura hardware para la extracción de invariantes ópticos, a partir del flujo óptico, en tiempo real“. Tesis Doctoral, Universidad Politécnica de Madrid, 2001.

[2] Cobos P., Monasterio F., “Fpga implementation of the Horn & Schunk Optical Flow Algorithm for Motion Detection in real time Images”. *Dcis`98 Proceedings, XIII Design of circuits and integrated systems conference*, pp: 616-621. (1998).

[3] Cobos P., Monasterio F., “Fpga implementation of a Log-polar Algorithm for real time Applications”. *Dcis`99 Proceedings, XIV Design of circuits and integrated systems conference*, pp: 63-68. (1999).

[4] Cobos P. , Monasterio F., "FPGA implementation of Camus correlation Optical Flow Algorithm for real time images." *Vision Interface Proceedings VI2001, 14th International Conference on Vision Interface, S.S. Beauchemin, F. Nouboud and G. Roth, Editors Canada*, pp: 7_9 (2001).

[5] Cobos P, Monasterio F. "FPGA implementation of Santos_Victor optical flow algorithm for real_time image processing: a useful attempt," VLSI Circuits and Systems, José Fco. López, Juan A. Montiel_Nelson, Dimitris Pavlidis, Editors, Proceedings of SPIE Vol. 5117 (2003) (23_32).

[6] http://www.sec.upm.es/pcobos/

[7] Barron, J.L., Fleet, D.J., Beauchemin, S. S., “Systems and Experiment Performance of optical flow techniques“. *International Journal of Computer Vision*, Kluwer Academic Publishers. Vol. 12, Nº 1, pp: 43-77. (1994).

[8] Horn, B., and Schunck, P., “Determining Optical Flow”. *Artificial Intelligence,* Vol 17, pp: 185-203. (1981).
[9] Camus,T., “Real-Time Optical Flow”. *PhD Thesis*, (1994).
[10] Camus, T., “Calculating time-to collision with real-time Optical Flow”. *SPIE Visual Communications and IMAGE Processing*. Vol 2308, pp 661-670. (1994).
[11] Cobos, P., Monasterio F., "FPGA Based Development Vision System". *Dcis`2001 Proceedings, XVI Design of circuits and integrated systems conference*, pp: 322-326. (2001).
[12] Cobos, P., Monasterio F., "FPGA Board For Real Time Vision Development Systems". *Proceedings of the ICCDCS 2002, 4th IEEE International Caracas Conference on Devices, Circuits and Systems*. 0-7803-7381-2/02/$17.00 © 2002 IEEE T021 pp:1-6 (2002).
[13] Cobos, P., "Real Time Hardware Vision System Applications: Optical Flow and Time to Contact Detector Units". *Proceedings of the ICCDCS 2004, 5th IEEE International Caracas Conference on Devices, Circuits and Systems*. 0-7803-8777-5 © 2004 IEEE , pp:281-288 (2004).
[14] www.ovt.com
[15] www.averlogic.com
[16] www.altera.com
[17] LSP Signal Processing. http://www.signal-lsp.com (2005).
[18] Chellapa, R., Quian, G, and Srinivasan, S. "Structure from motion: sparse versus dense correspondence methods". IEEE Proceedings 1999 International Conference on Image Processing, Vol 2, pp:492_499, 1999.

APPENDIX: ALTERA´S APEX DEVICE FAMILY

The capacity of APEX™, the programmable logic device family from ALTERA, ranges from 30,000 to 1.5 million system gates and uses technologies of 0.22-µm, 0.18-µm and 0.15-µm. Its architecture, shown in Fig. 10, is suited for solutions SOPC (system-on-a-programmable-chip), allowing designers to use it easily in a broad range of applications, which combine logic and memory needs, as is the case with the present vision system.
The device used in this application belongs to the 20k device family, more specifically the 20k100 TC144-3, with a power supply of 2.5V and an equivalent capacity of 263,000 system gates,

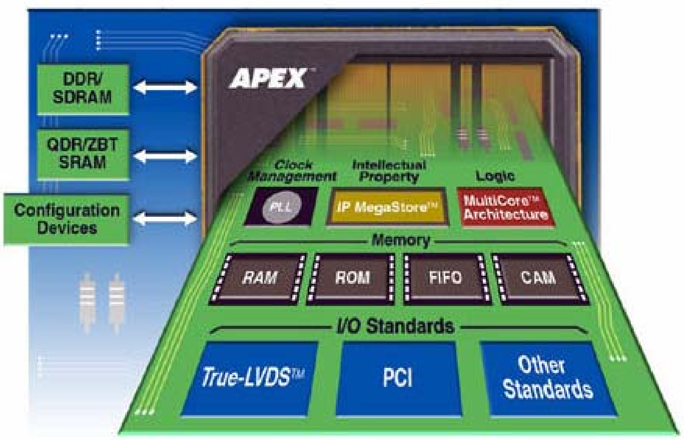

Fig. 10 Apex 20k device family: Solutions to build full systems in a programmable chip.

together with 101 input/output pins. The system can use IPs designed by Altera or by third-parties, as well as different types of memories, such as ROM, RAM, dual-port and FIFO.
Its CPLD-type architecture is based on fine grain logic blocks which can be grouped to build more complex ones. The interconnection resources are predefined on the chip so that the compilations are more efficient. Furthermore, since the configuration is based on RAM technology –to take up less space– it is necessary to include in the system a FLASH memory module, EPC2, so that the system can keep a stable configuration once the application has been debugged.
In order to be able to programme the FPGA, the free broadcast environment, “Max + plus2”, and the programming cable “Bite-blasterMv” manufactured by ALTERA are used.