

FPGA Solution for Low Cost Applications of Real-Time Automated Visual Inspection (RT-AVI) Systems

Real-time requirements set an important aspect to consider during the design of most automated systems for visual inspection tasks. Usually, it is necessary to use specialised hardware to satisfy the above requirements. During the development stage it is normal to use expensive hardware that provides more and higher functionality than the necessary in the final application. This paper presents a hardware solution for low cost applications of Real-Time Automated Visual Inspection (RT-AVI) systems. This includes a new methodology for development of hardware cores implemented on programmable logic devices (FPGA). The proposed solution have been applied to different industrial inspection systems. In particular, the architecture has been implemented for SMARTMEC AVI system within a RENAULT VI crankshaft production line and also reused for the development of the SIVAFRUT AVI system for the quality inspection of preserved orange segments.

INTRODUCTION

To implant RT-AVI systems on industrial environments it is necessary, first, to satisfy requirements of real time, robustness and reliability, and second, requirements of economic profitability. A strong commitment exists among both.

The final price of one application and therefore their economic feasibility will depend on the development costs, and the cost of the image acquisition and pre-processing hardware. The different nature of the working environments and activities make very useful the use of commercial software (i.e. Matrox Image Library) [1] supporting hardware for real time during the development tasks. But, in most cases, once the application has been developed the general performances of the real time hardware used are better than the ones needed in the final application. Expensive hardware supports complex operations and architectures, but perhaps we only need a pipeline with a few very simple stages.

In this paper, we present a hardware solution which reduces the final cost of a RT-AVI system. This is based on three usual components: the frame grabber, the image processing board and the visualisation module. Also we present its application for different industrial solutions.

Also, we present a brief description of hardware modules and how they have been implemented. The flexibility of the solution and the cores allow the replacement of subsystems with little effort and to reuse the same processing cores for different products and applications.

SYSTEM DESCRIPTION

Fig. 1 shows a schematic view of the components for the proposed low cost hardware solution. The host used for the system is a personal computer (Pentium

III-500MHz). The different modules are attached to the internal PC system bus (PCI)[3]. Video images are acquired with a low cost Matrox Meteor [1] frame grabber that allows to digitalise analogue video on images of 512x512 pixels and 8 bits per pixel. The image processing module includes a configurable processing element that consist in the XC4036 FPGA of Xilinx [2]. This programmable device is configurable from the PCI bus using a Windows95-based visual programming environment. This advanced software tool allows to build a hardware pipeline to implement any particular image processing solution. For that purpose, we have implemented a library of generic image processing algorithms (LUTs, convolutions, arithmetic operations, etc.). The hardware cores allow the pre-processing of images in continuous video mode. To accelerate the algorithms, we use two memory banks of 256Kx32bits. The configurable processing element is communicated with the host using a FIFO buffer attached to a PCI interface. Finally the intermediate results can be visualised with the standard video board.

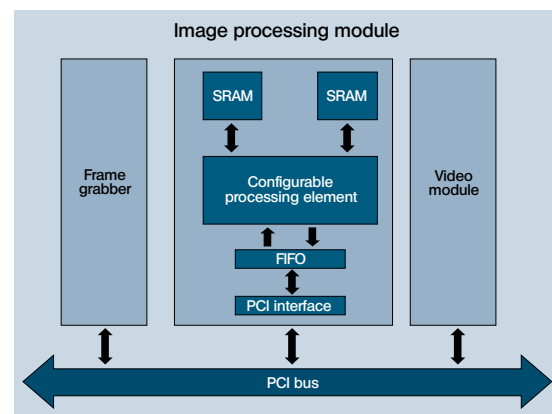


Figure 1. PCI-based low cost solution.

REQUIREMENTS

During the definition of the image processing module, the functional and non-functional hardware requirements, concerned to processing elements (cores), were described. Like in software applications, the specification of a functional model for the hardware architecture can be obtained from two categories: specification of functions and specification of behaviours.

The specification of functions describes the structural aspect of an architecture in terms of inputs, outputs, activities, internal data, physical structural of them, and data flow relationships between them. The specification of behaviours describes how a hardware architecture behaves in terms of events, inputs, states, conditions, and state transitions.

The analysis of the non-functional requirements allowed us to identify requirements that are referred to issues like how easy is to synchronise and reuse the cores, and how expensive is the development. In this way, these issues were considered from the very beginning.

Following, an abstract of main system requirements is presented.

A. Functional Requirements.

The basic functional requirements or services that the cores should provide to their end users are summarised within the following functional modules:

- Image acquisition set-up. This module should allow to the user to configure pixel resolution, image width and height.
- Monopixel operators. They allows to obtain all the basic images processes related with the use of look-up tables (binarization, negative, threshold, masks, scaled, etc) .
- Convolutions (2x2 and 3x3 pixels). This linear operator, computed as a weighted-sum of the pixel values in a Moore neighbourhood, is very used for many image processing operations (laplacian, roberts, sobel and high-pass filters).
- Template operator. It is an extension of the previous one in order to make possible dilatation and erosion of images.
- Histogram module. This core provides the number of occurrences of every pixel values appear in an

image. It allows to enhance images using histogram equalisation.

B. Non-Functional Requirements

Main non-functional requirements are the following:

- Maintainability and adaptability. The global system should perform different kind of tasks required by computer vision applications. For this reason, the design should be adapted to different cases with minimum effort.
- System safety. The system must be robust, because it should be usable on industrial environments.

$$t_p = f^{-1}.m.n \quad [2]$$

Hardware malfunction should be avoided.

- Scalability. To adapt the system to a specific application it should be modular and easy to use.
- Time to market. It is of strategic importance to develop the product fast enough to acquire a good position in the market. This will allow to get earlier profit and to pay off the investments made in the whole system.

HARDWARE MODULES.

We have been implemented a group of hardware cores from their functional and non-functional requirements as described in the previous section. Each one has been developed for a concrete image processing algorithm. These cores have been synthesised from a high level description language (VHDL) and using an EDA tool (Foundation of Xilinx). The architecture of the cores depends on the algorithm that is implemented for each one. At first, two types of architectures have been developed: monopixel and multipixel. The cores related with the use of look-up tables and histogram handling have been implemented based on monopixel architecture, while convolutions and temple operators have been designed from multipixel architectures.

A. Monopixel architecture

The architecture to implement monopixel algorithms consists in a three-stages pipeline, such it is shown in Fig. 2. Each pixel of the digitised image is sent to the FIFO input. The core, implemented on the FPGA, reads the data from the input FIFO, it processes them and

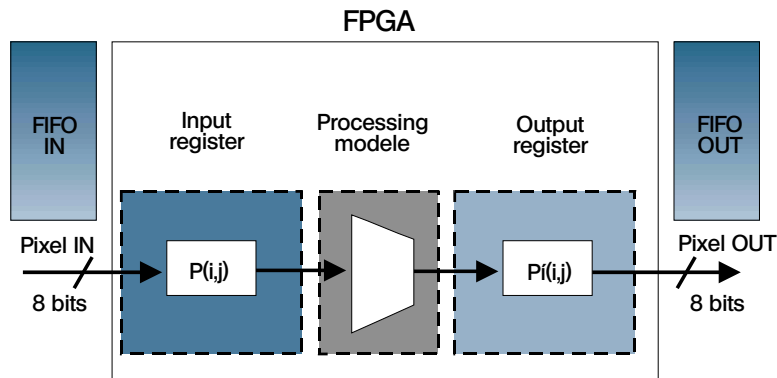


Figure 2. Monopixel architecture.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3
4 entity M_Calculo_Umbral is
5     port (
6         Pixel_IN: in STD_LOGIC_VECTOR (7 downto 0);
7         Pixel_Bin: out STD_LOGIC_VECTOR (7 downto 0)
8     );
9 end M_Calculo_Umbral;
10
11 architecture M_Calculo_Umbral_arch of M_Calculo_Umbral is
12 begin
13
14     process (pixel_in)
15     begin
16         if pixel_in > "11001000" then --200
17             Pixel_Bin <= "11111111";
18         else
19             Pixel_Bin <= "00000000";
20         end if;
21     end process;
22 end M_Calculo_Umbral_arch;

```

Figure 3. VHDL code for the threshold HW module.

writes the results on the output FIFO. The input register, the output register and the processing module set the processing core.

Each pixel $p(i,j)$ is acquired in sequential way and stored at the input register. After a first upward clock edge, the stored pixel inside the eight-bit flip-flop type D,

$$t_{pp} = f^{-1} \quad [1]$$

is processed by the combinational logic. The delay of this module is related with the delay due to the combinational logic and to its connections. This process is repeated for the following clock flanks, until the $m \times n$ pixels are processed. So, the processing time per pixel (t_{pp}) will be the following:

Where f is the clock frequency for the combinational

circuit. Then, for a image with $m \times n$ pixels the processing time (t_p) will be the following:

The work frequency comes fixed by the number of gates used inside the FPGA and the complexity of the design. For the XC4036 FPGA, we have obtained typical clock frequencies around 40 MHz. For this frequency the processing time will be:

$$t_p = \frac{256.256}{40.10^6} = 1,63 \text{ ms} \quad [1]$$

Using this architecture can be configured easily different algorithms concerning the use of look-up tables, such as the threshold operation. Figure 3 shows the VHDL code used for this purpose.

B. Multipixel architectures

To obtain a pixel of one image processed by means of convolution and template operators we need to store, as minimum, a number of pixels equal to the size of the kernel (2x2 or 3x3). We propose two types of architectures to implement this kind of algorithms, due to the necessity to store a certain number of pixels for their later process: architecture for line-processing and architecture for mask-processing.

In the architecture for line- processing it is necessary to previously store (k-1) lines of the image, where k is the mask dimension. Once this is concluded, pixels corresponding to the k line are read. As the pixels are read they are processed by the combinational logic and we obtain a pixel-value for the output image.

Figure 4 shows this architecture for a 2x2 convolution.

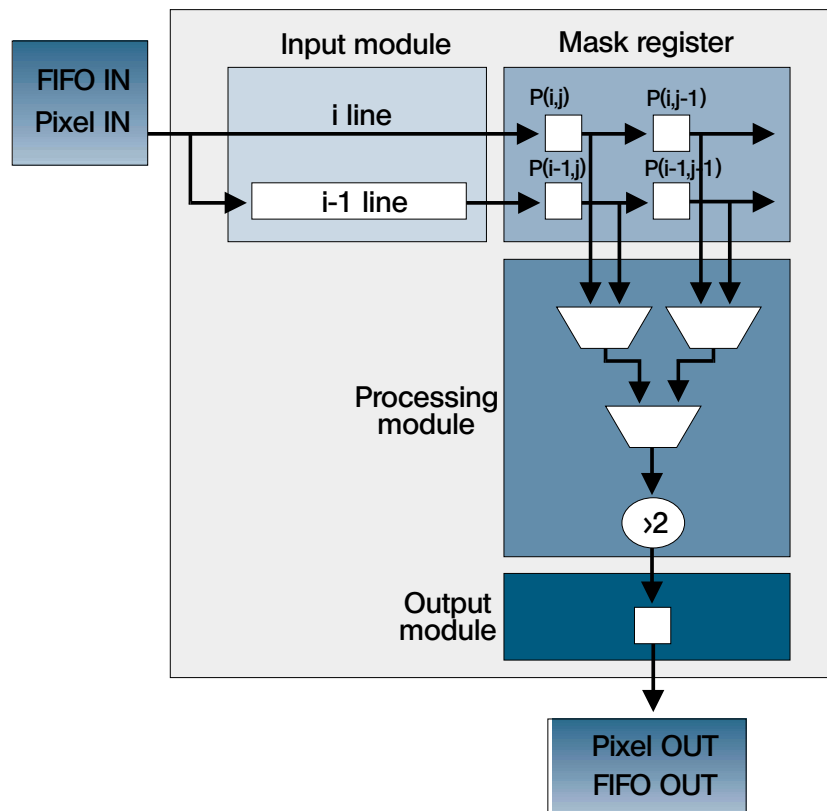


Figure 4. Multipixel architecture for 2x2 convolution.

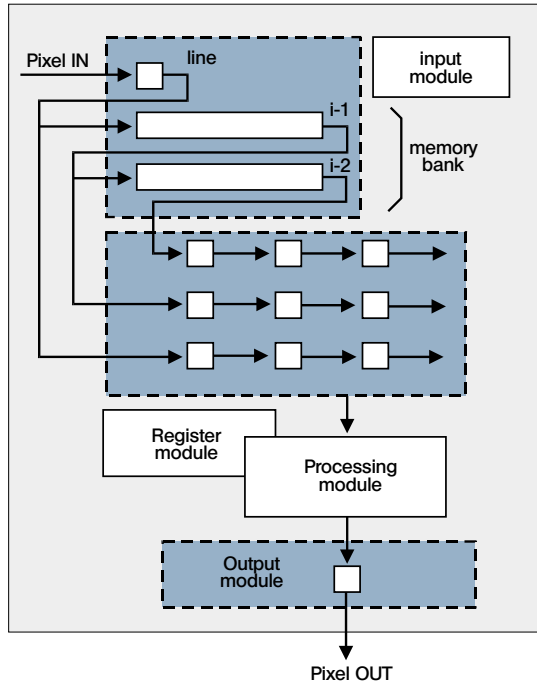


Figure 5. Multipixel architecture for 3x3 convolution.

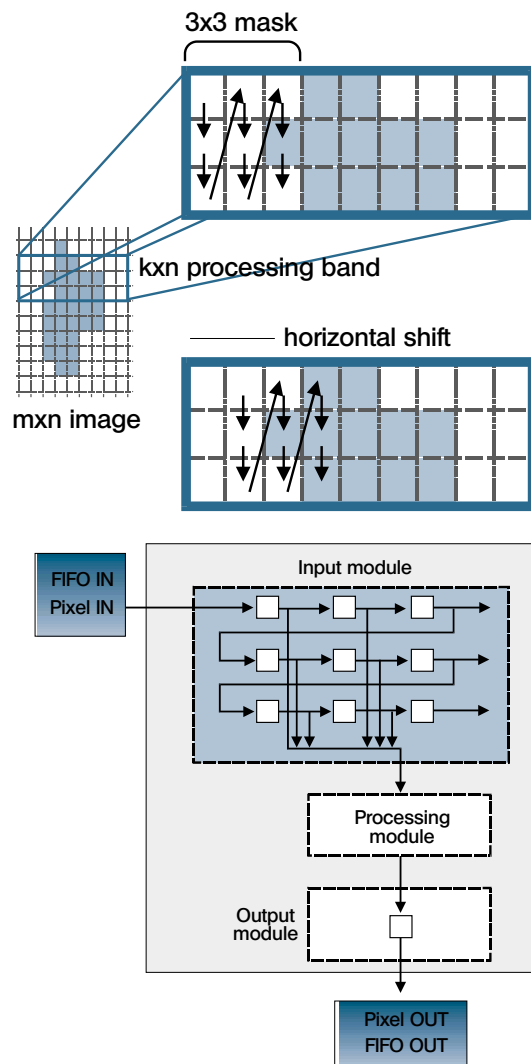


Figure 6. Zig-zag solution for 3x3 convolution.

The input module registers the $i-1$ line in the SRAM. Two pixels of the following line (i), given by the input FIFO, are located in the mask register. At the same time, two pixels from the $i-1$ file are transferred to the mask register. With the 2x2 pixels in the mask register, the processing module can obtain the output pixel using combinational logic. This solution can be used to configure a pipeline of several stages. It is only necessary to connect the output of the processing module with PIXEL IN line.

The implementation for a 3x3 convolution (Fig. 5) requires a bigger number of registration elements for the mask, as well as for storing the first two image lines ($i-1, i-2$).

In this solution the synchronization among modules is more complex than in the 2x2 convolution. This complexity sets up the necessity of using a bigger number of auxiliary elements for control and communication purposes. As the length of clock lines increases, the propagation time becomes significant and therefore it is necessary to reduce the frequency of the synchronization clock. This fact reduces processing speed.

A possible strategy to increase the clock frequency is to reduce the complexity of the design, increasing the number of memory access. In this case, we have to balance the problematic of increasing the processing rate diminishing the complexity of the design on one hand, and diminishing the processing rate increasing the number of accesses to memory, on the other. A solution based on this idea is shown in the Figure 6. For each $k \times n$ processing band all the pixels of this are acquired in zig-zag way. Then, to process a band of $3 \times n$ pixels we have to acquire the first nine pixels in zig-zag way. Once we have transferred these pixels, we move the mask on pixel to the right. Then it is possible to process the nine following pixels again. Following this method, the complexity of the design is simplified in front of the solution previously shown, although we increase the number of accesses to memory. Although this is a worse solution when only one convolution is needed, it allows to chain more than one convolution, diminishing the complexity of the auxiliary logic, and making therefore suitable its use inside a FPGA.

Table 1 summarises the processing times and the best internal clock frequency for some of the developing hardware cores:

Algorithm	Processing time (ms/pixel)	Clock Frequency (MHz)
Threshold	0,250	50
Negative	0,287	50
Area	0,287	50
2x2 convolution	0,178	50
2x2 convolution (zig-zag)	0,178	60
3x3 convolution (zig-zag)	0,539	37

Table 1. Processing times and clock frequency.

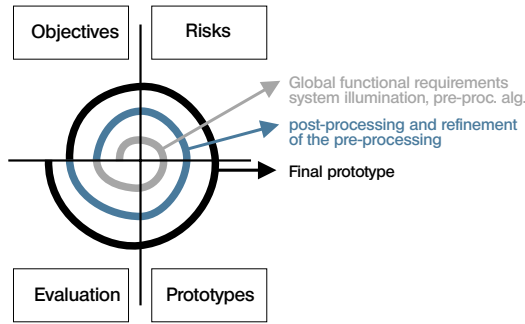


Figure 7. Boehm spiral model for AVI systems.

BUILDING NEW APPLICATIONS

The new methodology we propose for the development of real-time automated visual inspection (RT-AVI) systems is based on our own experience in the design of this kind of systems.

Usually, for the development of this type of applications we follow the model proposed by Boehm [4], as shown in figure 7. This model describes an iterative process for development of software where the planning of objectives, identification of risks, development of prototypes or products and evaluation of results are part of each iteration. Therefore, this model has allowed us to identify a set of development cycles. Each of them has served to deal with a set of development risks, reducing the errors in the selection of the platforms, in the design and implementation.

For the development of AVI systems we normally complete three cycles. At first, the global functional requirements (this include real-time requirements) of the application are specified. Next, the objectives of the first cycle of the spiral are established. These usually consist on finding the appropriate system of illumination for the application and to establish suitable algorithms for pre-processing step. Starting from these objectives and analysing the risks of the application we define a first prototype that allows to evaluate if the established objectives have been reached. In this first cycle we usually use, as development tool, the MIL library from Matrox. These libraries are very flexible and allow us to test a great quantity of solutions with a relatively light effort.

The second cycle of the Boehm spiral model is devoted to the post-processing step, with the help of neural networks. During this cycle, the pre-processing algorithms are also refined considering the results from the second prototype. In this way, when the second cycle is finished, the suitable pre-processing algorithms and the rejection strategies are well defined.

The third cycle of the spiral is devoted to define the

final prototype that will be installed within the production line. In this stage, it is necessary to guarantee that it fulfil the condition of temporary answer of the system. For this purpose, the developed application is optimised over a hardware platform that facilitates the required performance regarding real time. For many of the applications that we have developed, we have used the boards from Matrox. Main problem we have encountered with the use of this type of devices is its relative high cost, mainly when the benefits of real time are very strong. Most of the applications on which we have worked are for the manufacturer industry, where automated visual inspection systems compete for 100% inspection of the production. In this type of industries, the AVI substitutes low qualified workers and the profitability of the system is based on obtaining a very cheap product (typically around 18.000 euros/AVI platform). In many occasions, this price is very difficult to reach due fundamentally to the hardware cost that facilitates good real time performances. For this reason the use of a low cost hardware solution like the one proposed becomes of great interest.

Table 2 shows the overall costs for two AVI applications solved with two different solutions (traditional and new solution). The first of the applications (SMARTMEC -see illustr. 2-)[6][7] it is an RT-AVI system for a RENAULT V.I. crankshafts production line, while the second (SIVAFRUT -see illustr. 1-)[5] consists on the development of an RT-AVI system for the inspection of preserved orange segments.

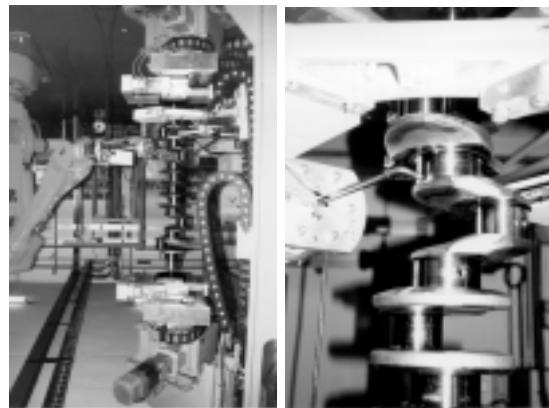


Illustration 1. SMARTMEC RT-AVI System.

CONCLUSIONS

A methodology for AVI-RT systems design based on the use of a low cost hardware architecture has been presented. It has been reused for the development of two automated visual inspection systems for 100% product inspection for the manufacturing industry. This

	SIVAFRUT		SMARTMEC	
	Traditional Solution	New Solution	Traditional Solution	New Solution
Software for development	2.152	2.152	2.152	2.152
Hardware Platform	12.158	3.616	24.306	5.100
Total Cost(Euros)	14.310	5.768	26.458	7.252

Table 2. Costs of different solutions.



Illustration 2. SIVAFRUT RT-AVI System.

has allowed to guarantee its success. The adaptability and the flexibility of the hardware solution allows the replacement of hardware cores with little effort, which assures that both the methodology and the hardware solution meet the adaptability requirements.

In this way, we devote some time to study the methodology used in the development of automated visual inspection applications and the hardware solutions for real-time applications. The result was a generic requirements specification that allows us to obtain a generic design. We have focused our work in the development of generic cores for real-time applications. The extra time for definition invested to develop the first system, has allowed to significantly reduce the development time for the second one ■

Andrés Iborra received the Ph.D. degree in Industrial Engineering from the Technical University of Madrid, Spain in 1993. He has worked as research engineer in the field of instrumentation systems, for nuclear power plants, in the company Westinghouse-Nuclear Equipment S.A. (ENWESA) during one year (1993-1994). In this company, there is occupied the position or R&D Managing since 1995, participating as project leader in different works for the industry, the most of them focused in mechatronics and robotics applications for hazardous environments. He is currently Professor in the Dept. of Electronics Engineering at the Technical University of Cartagena. His current research interests include computer vision software, machine vision and robotics since 1993.

Carlos Fernández received the Ph.D. degree in Industrial Engineering from the Technical University of Madrid, Spain in 1997. He is currently Professor in the Dept. of Information Technologies and Communications at the Technical University of Cartagena. He has been working as research engineer in the field of robotics and computer vision at the Universidad Politécnica de Madrid from 1989 to 2000. His current research interests include computer vision software, machine vision and robotics.

Bárbara Álvarez received the Ph.D. degree in engineering from the Universidad Politécnica de Madrid (UPM), Spain in 1997. She is presently Professor in the Dept. of Information Technologies and Communications at the Technical University of Cartagena (UPCT). She has been working as research engineer in the field of real-time systems and software architecture at the Universidad Politécnica de Madrid from 1993 to 1997. Her current

research is in the field of codesign of heterogeneous systems for industrial applications and mechatronics systems at the Universidad Politécnica de Cartagena (UPCT) since March 1998.

Jose Fernandez received the Ph.D. degree in engineering from the Universidad Politécnica de Madrid, Spain in 1989. Nowadays He is Chair Professor of Automatic Control Systems at the Department of Automatic Control and Electronic Engineering, Universidad Politécnica de Cartagena (UPCT). He has been working as research engineer in the field of Automatic Control and Electronic Engineering at the Universidad de Murcia from 1970 to 1999. He is the manager of the department of R+D of the Commerce, Industry and Navigation Official Chamber of Cartagena, where he is developing at the present time the project FARO, which is dedicated to transfer GPS techniques to taxis and public transport. Nowadays, he is working also in the field of mechatronic engineering applied to industrial systems.

ACKNOWLEDGEMENTS

The authors want to acknowledge the DG XII of the Commission of the European Communities for SMARTMEC Project's financing within the BRITE Euram Programme (BRPR-CT97-0372), as well as the Spanish government for CICYT's support for SIVAFRUT project's (TAP-1FD97-1606-C02-01).

REFERENCES

- [1] <http://www.matrox.com>.
- [2] <http://www.xilinx.com>.
- [3] Shanley T., Anderson D. "PCI System Architecture". MindShare, Addison-Wesley, 1995.
- [4] Boehm B., "A spiral Model of Software Development and Enhancement", IEEE Computer, vol. 21, n°5, 1988.
- [5] Navarro, P.J. "Hardware architecture for computer vision algorithms", UPCT Internal Report, Cartagena, November 2000.
- [6] A. Iborra, B. Alvarez, C. Jimenez, J.M. Fernandez-Meroño, C. Fernández, and J. Suardiaz "Automated Visual Inspection (AVI) System for crankshaft production processes", PHOTOMEC'99/ETE'99, Liege, November 1999.
- [7] C. Fernández, J. Suardiaz, A. Iborra, J.M. Fernandez-Meroño. "On-line automated visual inspection for quality control within the automobile industry". Proceedings of the QCAV99- 5th International Conference on Quality Control by Artificial Vision, Trois-Rivieres (Canada). ISBN 0-9680833-2-3.