System Prototyping dedicated to Neural Network Real-Time Image Processing

Rolf F. Molz[1], Paulo M. Engel[1], Fernando G. Moraes[2], Lionel Torres[3], Michel Robert[3]

[1]*UFRGS – Instituto de Informática*
Av. Bento Gonçalves, 9500 – Bloco IV
Caixa Postal 15064
91501-971 - Porto Alegre – Brazil
{rolf,engel}@inf.ufrgs.br

[2]*PUCRS – Faculdade de Informática*
Av. Ipiranga, 6681 - Prédio 30
90619-900 - Porto Alegre - Brazil
moraes@inf.pucrs.br

[3]*LIRMM –Université Montpellier II*
161, rue Ada
34392 Montpellier - Cedex 5 - France
{torres,robert}@lirmm.fr

Abstract

Pattern localization and classification are CPU time intensive being normally implemented in software, however with lower performance than custom implementations. Custom implementation in hardware (ASIC) allows real-time processing, having higher cost and time-to-market than software implementation. We present an alternative that represents a good trade-off between performance and cost. This paper presents initially the state-of-the-art in this field, analyzing the performance and implementation of each work. After we propose a system for localization and classification of shapes using reconfigurable devices (FPGA) and a signal processor (DSP) available in a flexible codesign platform. The system will be described using C and VHDL languages, for the software and hardware parts respectively, and has been implemented in an APTIX prototyping platform.

1. INTRODUCTION

Vision has long fascinated researchers from a broad range of disciplines, such as psychology, neural science, computer science, and engineering. Vision has been defined as a process of recognition of objects of interest, and it deals with image understanding. Artificial Neural Networks (ANN) have been used to model the human vision system. They are biological inspired, containing a large number of simple processing elements, with an execution model analogous to biological neurons. The natural neurons provide a computing architecture that is radically different from computers that are widely used today: they are massively parallel systems.

This paper proposes an image processing prototyping system based on an APTIX platform [1], aiming the integration of a Digital Signal Processor (DSP) with programmable devices (FPGA). The system comprises preprocessing, feature extraction and classification tasks. These tasks will be employed to find and classify shapes for an input image. These shapes can be traffic signals, landmarks for robotics, car license, and so on.

The DSP processor is used when sequential processing is required or floating point arithmetic is needed. The programmable device is used for parallel and bit level operations. This partition improve the performance of the system when compared to pure DSP implementations, reduce the time to market and cost when compared to ASIC implementation. So, this hardware/software system represents a good trade-off between performance and cost.

This system is being implemented in a codesign platform (APTIX [1] board). This board has two FPGA modules (10k100 - Altera), one DSP core module (D950 - ST Microelectronics), and two memories modules (256Kbytes – 4 bits each one).

This paper is organized as follows. Section 2 presents the background in object localization and classification, describing the implementation (software, mixed, ASIC or FPGA) and performance of each system. Next, in Section 3, we describe the system structure, detailing the tasks that must be accomplished. Section 4 presents preliminary results of the hardware implementation. Finally we present our conclusion and future work.

2. BACKGROUND

In this section, we describe some systems dealing with object localization and classification. The relevant works published in the literature are presented in [2].

In [11] the objects must be in pre-defined positions and is completely implemented in software. A hardware implementation using the PAPRICA ASIC connected to a host is also presented in [11]. The reported results do not allow real time processing (2.5 frames/sec). The work [10] assumes a fixed distance between the vehicle and the image sensor. This restriction allows the use a proportional rule in order to accelerate the detection of desired patterns in the vehicle license. This system has a mixed implementation, with the software part running in a PC, and the hardware part running in a FPGA. The system can be used in real-time applications (30 frames/sec). The work [9] uses ANN for classification, however the system is implemented in software, resulting in a poor performance (10 sec for localization and classification). Finally, a similar work is presented in [3], aiming object localization and classification, however it was implemented in software (10-15 frames/sec). These works do not include object occlusion.

Our objective is to develop a portable system for image processing, having light weight and low power consumption. The performance for real time processing is obtained implementing the parallel operations in hardware, using FPGAs. This hardware/software implementation is a full stand-alone system, able to execute all required tasks for shape localization and classification. This system can be implemented in a dedicated ASIC, characterizing a system-on-a-chip for image processing.

To complete our system, we can connect it to a CIS circuit (CMOS Image Sensor) in order to include the image acquisition task. Finally, the learning phase can be implemented in software, increasing the flexibility of the system.

3. PROPOSED SYSTEM

In this section, the tasks for shape localization and classification are described. Figure 1 presents the main tasks executed by our system.

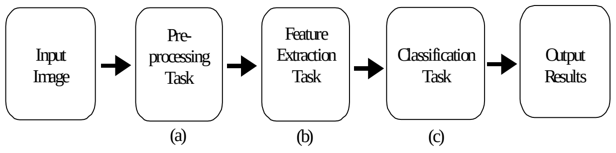


Figure 1 – System General Vision

The preprocessing task (Figure 1a) accomplishes the localization process. We can find in the literature some methods for geometrical shapes (objects) localization: texture analysis [12], fuzzy logic [13], local features [14], global features of the desired shapes (polygon approximation [15] and dominants points detection [16]) and mathematical morphology [17]. In this work we used the *global features of the desired shapes* method.

The first task to obtain the *global features of the desired shapes* is the segmentation or line extraction process. In [18] several methods for line extraction in images are presented, and a comparison between their performance is done. The works [18] and [19] relate that the Canny method presents the best results among several algorithms, however it is CPU time consuming to consider real-time processing. An alternative approach, [20], presents a method suited for hardware implementation. This algorithm performs *line extraction* and *segmentation* using the gradient and direction of each pixel into the source image.

The system to implement such method is depicted in Figure 2, as a sequence of independent tasks.

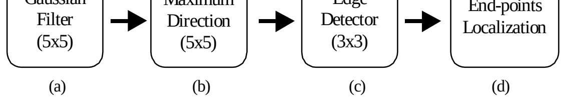


Figure 2 – Preprocessing tasks implemented in hardware

The segmentation process starts by applying a Gaussian Filter over the input image (Figure 2a). This filter acts as a smoothing function in the input image. In this work we used a 5x5 window. Equation 1 describes this filter. A fixed-point constraint is applied to this filter, since it will be implemented in hardware.

$$g'(i,j) = \sum_{p=-m}^{m} \sum_{q=-n}^{n} w(p,q) * g(i-p, j-q) \qquad (1)$$

Where: w (p, q) = weighting coefficients;
g (i-p, j-q) = gray value of the (i-p, j-q) pixel;
g' (i, j) = new gray value of the (i, j) pixel;
m, n = width and height window.

Next, each pixel direction is obtained, using a 5x5 window (Figure 2b). Each pixel can have one of four different directions: horizontal, vertical, positive and negative slope. The third step applies a threshold value over the image containing the pixel direction, using a 3x3 window (Figure 2c). The resulting image contains the edges of the original image, with its directions.

Figure 3 illustrates these three steps. Figure 3a presents the original image, Figure 3b the image after the Gaussian filter, Figure 3c the image with the pixel directions and in the Figure 3d the image with the edges obtained from Figure 3c.

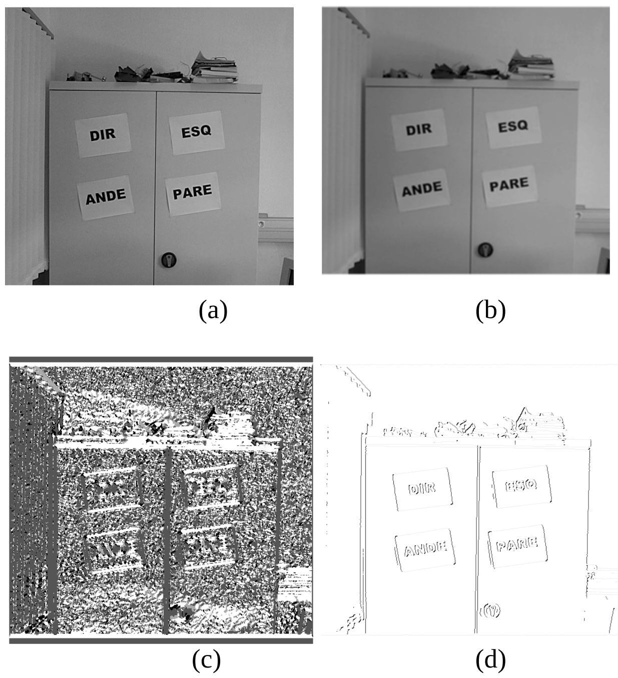


Figure 3 – Segmentation process

The segmentation process (steps a, b and c in Figure 2) does not localize the lines. The algorithm only detects the existence of the lines, with some dimension and some angle inside the image. To obtain the exact localization of these lines two approaches can be considered:

1) **Vertices localization**. Pre-defined convolution masks are applied in the image to obtain the vertices localization. This process calculates the straight angles in this image. The convolution masks works only for straight angles or vertices with zero near values. This method can not be used, since the image acquisition process can introduce some noise, making difficult to find

straight angles. Others techniques for this method can be used, such as [21], [22] and [23].

2) **Lines end-points localization**. This approach is used in [24]. This method can be easily implemented in hardware, and has a good performance. However, a good segmentation process is necessary for the end-points localization algorithm. This algorithm is the task (d) in Figure 2. The output of this algorithm gives the initial position, final position, angle and length for each line in the input image. Figure 4 illustrates the image obtained by this algorithm.

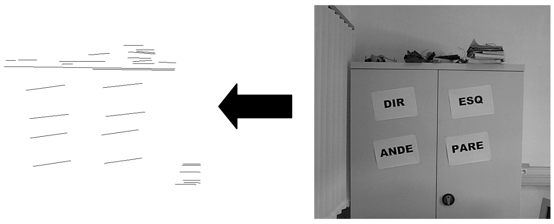


Figure 4 – Lines obtained using the end-point localization algorithm

The next task in the system is the feature extraction (Figure 1b). Selection of a feature extraction method is probably the most important factor in achieving high recognition performance. Given the large number of feature extraction methods reported in the literature [29], a newcomer to the field is faced with the following question: w*hich feature extraction method is the best for a given application?*

For our application, we work with two methods:

1) Angle correction and viewport mapping (50 x 50pixels) [25]. Next, the averages in the viewport columns are calculated. These average values are applied in the Classification Process (Figure 1c).

2) In the future, this feature extraction methodology will be replaced by moment-invariant [28]. This technique defines a set of seven moment-invariant functions that are invariant to translacional, scale, and rotational differences in input patterns. The main disadvantage of the moment-invariant technique is that there is no guarantee that seven functions represents a complete set of descriptors. However, for practical applications the set of seven invariants are adequate to distinguish between input patterns [26].

Finally, a MLP (Multi-Layer Perceptron) Artificial Neural Network (ANN) with supervised learning accomplishes the classification task (Figure 1c). The method to correct errors is the backpropagation algorithm.

4. Implementation Analysis and Results

4.1 Hardware Prototyping Environment

The APTIX environment [1] is a new generation of prototyping platforms. The hardware part of the system is essentially a board organized as female support matrix allowing to connect virtually any integrated circuit, as FPGAs, Microcontrollers or DSPs, Analog/Digital converters, etc. A MP3 Aptix board is used with 2 Flex 10K100 modules (Altera, 100,000 equivalent gates), 1 ST18952 module (DSP core - ST Microelectronics), and 2 256Kbytes/4bits memories blocks. The Altera FPGAs uses two clock frequencies, 16 Mhz or 25 Mhz. Figure 5 shows the Aptix prototyping system.

The block diagram presented in Figure 6 corresponds to the hardware implementation of our system. This diagram shows one part implemented in FPGA and other part implemented in DSP. The FPGA part contain four developed cores: Interface Core, ADM Core [16], Lines Core and Neural Core.

The system is organized as a pipeline, to increase the final throughput of the system. Between each pipeline stage (task), buffers are inserted. Between the Gaussian filter and direction computation a buffer with 5 image lines is required. So, the direction computation starts after the Gaussian Filter has processed 5 lines. Between the edge detection and direction computation a buffer with 3 image lines is inserted, corresponding to edge detection window. The edge detection writes its results in the FPGA external memory. The end-point localization reads the memory, sending each processed line to the DSP.

The DSP processes the tasks corresponding to shapes extraction and feature extraction. We consider in the future implement the shapes extraction in hardware. The feature extraction is done in the DSP, since a huge amount of mathematical operations are executed. The work [29] present some methods for feature extraction. These feature values are sent to the classification block implemented in hardware.

ANN accomplishes the last system task, classification. This ANN is implemented in hardware (FPGA) assuring a great output performance due to the parallel operations [27].

The complete system consumes 4600 (≈ 92100 gates) Logic Elements (10K100 Altera has 4992 Logic Elements). The timing tool has fixed the maximum operation frequency as 9,2MHz for the 10K100gc503-4 FPGA device. Using the 10K100gc503-3 device we obtained 11,2MHz. Because the image size used is 256x256 (65536 pixels/image) and the bottleneck is the ADM Core with 51 clock cycles, the system has a throughput of 2,7 frames/sec in a 10K100gc503-4 FPGA. Using faster devices, Virtex (from Xilix) or APEX (from Altera) the operation frequency can easily be enhanced, allowing real-time processing.

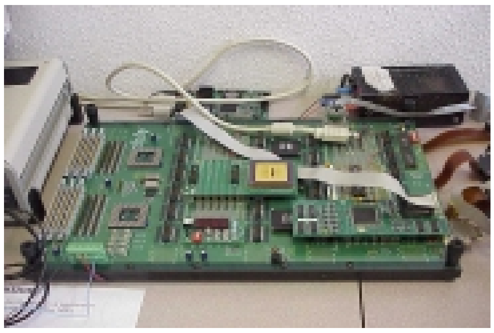


Figure 5 - Prototyping System

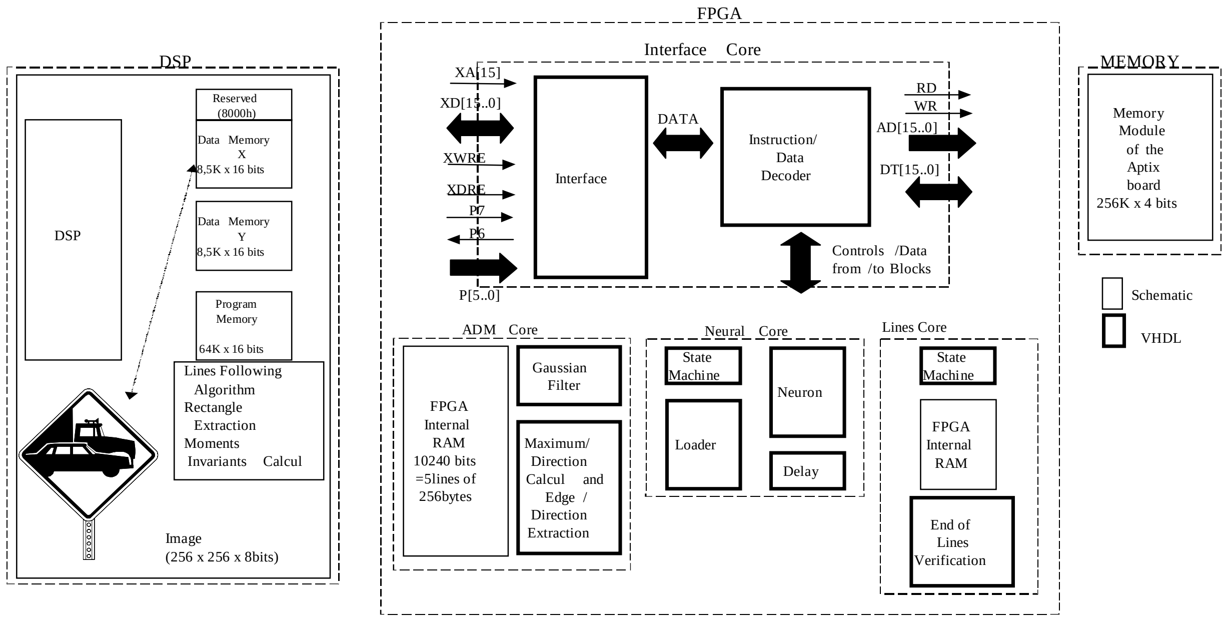
4.2 Implementation Comparison

With this implementation we can make some comparisons. The employed cost function to measure performance in the following tables is the number of clock cycles, since it represents the real speed-up of the FPGA over the other devices.

The classification block was implemented in 4 different devices: FPGA, two DSP processors and a standard microprocessor. Table 2 presents the comparative results for the neuron propagation without the activation function. The simulations were executed with two different configurations: fixed- and floating-point weights.

Table 3 shows the results obtained for the activation function. The simulations were executed with three different configurations: table with floating-point and fixed-point weights, and tangent function.

Finally, Table 4 shows the results obtained for the gaussian filter. The simulations were executed with two different configurations: fixed- and floating- point values.



DSP
Reserved (8000h)
Data Memory X 8,5K x 16 bits
Data Memory Y 8,5K x 16 bits
Program Memory 64K x 16 bits
DSP
Lines Following Algorithm Rectangle Extraction Moments Invariants Calcul
Image (256 x 256 x 8bits)
FPGA
Interface Core
XA[15]
XD[15..0]
XWRE
XDRE
P7
P6
P[5..0]
Interface
DATA
Instruction/ Data Decoder
RD
WR
AD[15..0]
DT[15..0]
Controls /Data from /to Blocks
ADM Core
FPGA Internal RAM 10240 bits =5lines of 256bytes
Gaussian Filter
Maximum/ Direction Calcul and Edge / Direction Extraction
Neural Core
State Machine
Neuron
Loader
Delay
Lines Core
State Machine
FPGA Internal RAM
End of Lines Verification
MEMORY
Memory Module of the Aptix board 256K x 4 bits
Schematic
VHDL

Figure 6 – System blocks diagram

Table 2 – Results comparative for the propagation neuron

Description	FPGA (25MHz) 40ns	DSP D950 (*) (143MHz) 7ns	DSP TMS320C6201 (*) (200MHz) 5ns	Pentium MMX (*) (233MHz) ≈4.3ns
Fixed-point weights	200ns (5 cycles)	7,7µs (1040 cycles)	360ns (72 cycles)	1.06µs (246 cycles)
Floating-point weights	Non Implemented	53,3µs (7195 cycles)	12,5µs (2514 cycles)	1.9µs (441 cycles)

*: The program was compiled from C-code.

Table 3 – Results comparative for the activation function

Description	FPGA (25MHz) 40ns	DSP D950 (*) (143MHz) 7ns	DSP TMS320C6201 (*) (200MHz) 5ns	Pentium MMX (*) (233MHz) ≈4.3ns
Table (Floating-point values)	Non Implemented	27,5µs (3709 cycles)	6,5µs (1307 cycles)	3.31µs (769 cycles)
Table (Fixed-point values)	40ns (1 cycle)	1,5µs (207 cycles)	320ns (64 cycles)	169ns (39 cycles)
Tanh function	Non Implemented	96,6µs (13039 cycles)	24µs (4812 cycles)	5.3µs (1232 cycles)

*: The program was compiled from C-code.

Table 4 – Results comparative for the gaussian filter

Description	FPGA (25MHz) 40ns	DSP D950 (*) (143MHz) 7ns	DSP TMS320C6201 (*) (200MHz) 5ns	Pentium MMX (*) (233MHz) ≈4.3ns
Fixed-point values	2.04µs (51 cycles)	26µs (3512 cycles)	8,3µs (1662 cycles)	6,1µs (1418 cycles)
Floating-point values	Non Implemented	180µs (24344 cycles)	41µs (8196 cycles)	12µs (2790 cycles)

*: The program was compiled from C-code.

5. CONCLUSIONS

This paper discussed the implementation of a complete system for shapes localization and classification. We presented preliminary results concerning the classification block, and the gaussian filter implemented in programmable devices (FPGAs) of a codesign platform.

The classification block has a performance 14 times (72 DSP cycles divided by 5 FPGA cycles) greater than the DSP TMS320C6201 and the gaussian filter block has a performance 32 times (1662 DSP cycles divided by 51 FPGA cycles) greater than the DSP TMS320C6201, considering the number of cycles. Despite of this frequency, the system is faster than DSP processors, since parallel operations are executed in this hardware block.

We expect a best performance in Virtex (Xilinx) or APEX (Altera) FPGAs, because the maximum operation frequency is technology-dependent. The complete system is now being implemented.

All presented algorithms to solve this problem are fast and suitable to hardware implementation, presenting small error when compared to software implementations. However, occlusion shapes are not considered by our system. This feature needs other processing methods, such as [14], [15], [16], and [24].

The proposed work can be a future prototype platform for image processing, allowing implementing and testing several algorithms. Also, each developed algorithm in VHDL can be inserted in library, composing in this way a rich environment to implement customized image processing systems.

ACKNOWLEDGEMENTS

The author Rolf Fredi Molz would like to gratefully acknowledge the support of CAPES Capes/Cofecub project number 245/98-Brazil. The author Fernando G. Moraes would like to gratefully acknowledge the support of CNPq project number 522939/96-1-Brazil.

REFERENCES

[1] APTIX Corporation. www.aptix.com

[2] MOLZ, R. F.; ENGEL, P.M. ; MORAES, F.G; TORRES, L.; ROBERT, M. “Design of a Classification System for Rectangular Shapes Using a Co-Design Environment”, XIII Symposium on Integrated Circuits and System Design (SBCCI2000), Manaus, Brasil, September 2000, will be presented

[3] GAVRILA, D.M. ; PHILOMIN, V. "Real-Time Object Detection for Smart Vehicules". International Conference on Computer Vision (ICCV99). Vol. 1. Corfu, Greece, 20-25 september, 1999.

[4] SIM, H.C.; NG. G.S. "Recognition of Partially Ocluded Objects with Back-Propagation Neural Network". Intern. Journal of Pattern Recognition and Artificial Intelligence, vol. 12, n°. 5, pp. 645-660, 1998.

[5] TSANG, K.M.; TO, F.W. "Recognition of Partially Occluded Objects using an Orthogonal Complex AR Model Approach". Intern. Journal of Pattern Recognition and Artificial Intelligence, vol. 13, n°. 1, pp. 85-107, 1999.

[6] FORESTI, G.L. "Object Recognition and Tracking for Remote Video Surveillance". IEEE Trans. On Circuits and Systems for Video Technology, vol. 9, n° 7, pp. 1045-1062, october 1999.

[7] LOWE, David G. "Object Recognition from Local Scale-Invariant Features". Proc. Of the International Conference on Computer Vision, Corfu, Greece, 20 - 25 September, 1999.

[8] RUCKLIDGE, William. "Efficiently Locating Objects using the Hausdorff Dsitance". International Journal of Computer Vision vol. 24, n° 3, pp. 251-270, 1997.

[9] JORGENSEN, T.M.; CHRISTENSEN, S.S.; ANDERSEN, A.W. "Detecting danger labels with RAM-based neural networks". Pattern Recognition Letters, vol. 17, pp. 399-412, 1996.

[10] MINGO, Ferran L. "Configurable Computing for Real-Time Vision". Tése de Doutorado, Universitat Autônoma de Barcelona - UAB, pp. 180, Bellaterra, september, 1998.

[11] ADORNI, G.; GORI, M.; MORDONINI, M. "Just-in-time Landmarks Recognition". Real-Time Imaging, vol. 5, pp. 95-107, 1999.

[12] LIBERMANN, F. "Classificação de Imagens Digitais por Textura usando Redes Neurais". Dissertação de Mestrado – Instituto de Informática – UFRGS/ Brasil, 86 pp. 1997.

[13] HOEPPNER, F. "Fuzzy Shell Clustring Algorithms in Image Processing: Fuzzy C-Rectangular and 2-Rectangular Shells". IEEE Transactions on Fuzzy Systems, vol. 5, n. 4, pp. 599- 613, November 1997.

[14] LOWE, David G. "Object Recognition from Local Scale-Invariant Features". Proc. Of the International Conference on Conputer Vision. Sept. 1999.

[15] YANG, Fan. "Traitement automatique d'images de visages algorithmes et architecture". Thèse de Doctorat – U.F.R. Sciences et Techniques -Université de Bourgogne /France, 181pp., Juin 1998.

[16] TSANG, K.M.; TO, F.W. "Recognition of Partially Occluded Objects using an Orthogonal Complex AR Model Approach". Intern. Journal of Pattern Recognition and Artificial Intelligence, vol. 13, n°. 1, pp. 85-107, 1999.

[17] FORESTI, G.L. "Object Recognition and Tracking for Remote Video Surveillance". IEEE Trans. On Circuits and Systems for Video Technology, vol. 9, n° 7, oct. 1999.

[18] HEATH, M.; SARKAR, S.; SANOCKI, T.; BOWYER, K. "Comparison of Edge Detectors". Computer Vision and Image Understanding, vol. 69, n. 1, pp. 38-54, January, 1998.

[19] PARKER, J.R. "Algorithms for Image Processing and Computer Vision". John Wiley & Sons, Inc. New York. 1997.

[20] ALZAHRANI, F.M.; CHEN, T. "A Real-Time Edge Detector: Algorithm and VLSI Architecture". Real-Time Imaging, vol. 3, pp. 363-378, 1997.

[21] MOKHTARIAN, F.; SUOMELA, R. "Robust Image Corner Detection Through Curvature Scale Space". IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, n. 12, pp. 1376-1381, Dec. 1998.

[22] CHABAT, F.; YANG, G.Z.; HANSELL, D.M. "A corner orientation detector". Image and Vision Computing, vol. 17, pp. 761-769, 1999.

[23] FREEMAN, H. "On the encoding of arbitrary geometric configurations". IRE Trans. Electron. Comput, vol. 26, pp. 297-303. 1977.

[24] COCQUEREZ, J.; PHILIPP, S.; et al. "Analyse d'images: filtrage et segmentation". Masson, Paris, 1995.

[25] FOLEY, James D. "Computer Graphics : Principles and Practice". Second Edition in C. 1995.

[26] KULKARNI, Arun D. "Artificial Neural Networks for Image Understanding". Van Nostrand Reinhold, USA, 1994.

[27] MOLZ, R. F.; ENGEL, P.M. ; MORAES, F.G. “Uso de um ambiente codesign para a implementação de redes neurais”, IV Congresso Brasileiro de Redes Neurais, p. 13-18, São José dos Campos, Brasil, July 1999.

[28] HU, M.K. "Visual pattern recognition by moment invariants". IRE Transactions on Information Theory IT-8:28-32, 1962.

[29] TRIER, D.; JAIN, A.K.; TAXT, T. "Feature Extraction Methods for Character Recognition – A Survey". Pattern Recognition, vol. 29, n 4, pp. 641-662, 1996.