RECONFIGURABLE ARCHITECTURE FOR BINARY IMAGES INVARIANT MOMENTS EXTRACTION

Guilherme H. R. Jorge , Valentin O. Roda
Departamento de Engenharia Elétrica/ EESC
Universidade de São Paulo, São Carlos, Brazil
email: guijorge@hotmail.com,
valentin@sel.eesc.usp.br

Juan Pablo Oliver†, Julio Perez Acle†, Sebastian Fernández†
†Universidad de la Republica
Inst. de Ingeniería Eléctrica, Montevideo, Uruguay
email: jpo[jpaipai][sebfer]@fing.edu.uy

ABSTRACT

Moments of the intensity function of a group of pixels have been used for the representation and recognition of objects in two dimensional images. Due to the high computational cost of evaluating the moments, the search for faster computing architectures is very important. This work presents a soft core architecture for the extraction of invariant moments from binary images, using high density logic programmable devices.

1. INTRODUCTION

Computers can distinguish shapes in their environment using images from video cameras and can take decisions based on pattern classes [1]. The complete recognition process requires the following steps: image acquisition, pre processing, attributes extraction and classification. In this work we will deal only with the image attributes extraction. The image attribute stage searches for image relevant features (characteristics) that can be used to characterize the objects present in the images. The image features used to characterize the objects in the scene can be numerical such as distances, area and volume or symbolic such as color or textures. The choice of the attributes set must take into account some properties such as: processing speed, class discrimination, small variations for each class and description completeness [2]. Transformation invariant moments are often used as attributes of objects because they are robust and capable of representing image properties invariant to rotation, translation and scale. In this work we present a novel approach to a co-processor architecture, dedicated to the extraction of invariant characteristics from binary images.

2. HU INVARIANT MOMENTS

The representation of objects using attributes unchangeable to geometric transformations is an important issue for pattern recognition. The choice of the best attributes, most of the time, is related with the performance of the pattern recognition system using the invariant attribute space [3]. Image moments are numerical values of the image pixel intensities considering the distances to a reference point or axis. Image moments are used to characterize a binary or gray level image considering it as a bi-dimensional density function distribution [4] [5].

The representation of all image information would use an infinite number of moment values. Therefore for the practical implementation of recognition systems using moments it is necessary to determine the order of the moments that can bring relevant information that characterize the images. The definition of regular geometric moments has the form of a projection of the f(x,y) function which represents the image to a polynomial function of the type $x^p y^q$. The (p, q) moment is defined according to equation (1).

$$m_{p,q} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q . f(x,y).dxdy \quad , \quad (1)$$

for (p,q = 0,1,2,....) were $m_{0,0}$ is the area of the region and $m_{0,1}$ e $m_{1,0}$ are the coordinates of the center of mass of the region. Central moments are moments centralized in regions and for digital images can be expressed as equation (2).

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y), \quad (2)$$

were $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$, are the coordinates of the center of mass normalized by the área.

The first to third order central moments, invariant to translation and scale, are obtained from equation (2) for values of p and q between 0 and 3.

1st order

$$\mu_{10} = \mu_{01} = 0$$

2nd order

$$\mu_{20} = m_{20} - \bar{x} m_{10}$$
$$\mu_{02} = m_{02} - \bar{y} m_{01}$$
$$\mu_{11} = m_{11} - \bar{y} m_{10}$$

3rd order

$$\mu_{12} = m_{12} - 2\bar{y}m_{11} - \bar{x}m_{02} + 2\bar{y}^2 m_{10}$$
$$\mu_{21} = m_{21} - 2\bar{x}m_{11} - \bar{y}m_{20} + 2\bar{x}^2 m_{01}$$
$$\mu_{30} = m_{30} - 3\bar{x}m_{20} + 2\bar{x}^2 m_{10}$$
$$\mu_{03} = m_{03} - 3\bar{y}m_{02} + 2\bar{y}^2 m_{01}$$

The combination of the 2nd and 3rd order moments results in attributes that are invariable to the image rotation. The central moments normalized by the area are determined by equation (3).

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}} \quad , \quad (3)$$

Were: $\gamma = \frac{p+q}{2} + 1$ for $(p+q = 2,3,4,....)$

A set of seven moments, invariant to translation, rotation and scale was determined by Hu [6]. Those moments, named ϕ_1 to ϕ_7, are commonly referenced in the literature as Hu invariant moments.

$$\phi_1 = \eta_{20} + \eta_{02} \qquad \phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2$$
$$\phi_3 = (\eta_{30} + 3\eta_{12})^2 + (3\eta_{21} + \eta_{03})^2$$
$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$
$$\phi_5 = (\eta_{30} + 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + 3(\eta_{21} + \eta_{03}) (\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$
$$\phi_6 = (\eta_{20} + \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$
$$\phi_7 = (\eta_{21} + \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + 3(\eta_{21} + \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

3. DEVELOPED ARCHITECTURE

The determination of the invariant moments comprises the following steps: area and perimeter calculation of the selected object in the image, central moments calculation and finally the determination of the invariant moments themselves. A novel hardware dedicated architecture, implemented in VHDL, for the determination of invariant moments was developed, a block diagram of this architecture is shown in figure 1.

The developed architecture comprises an image acquisition subsystem were an image acquired by a video camera is stored in a memory. The image acquisition system was constructed based on the system, previously developed by Pedrino [9].

The calculation methodology uses the vertical and horizontal projections of the object in the image. Initially the area and the perimeter of the object are calculated and those values are used for the central moments calculation. Once the central moments were determined, the invariant moments are calculated in a parallel form speeding up considerably the calculations in relation to traditional architectures.

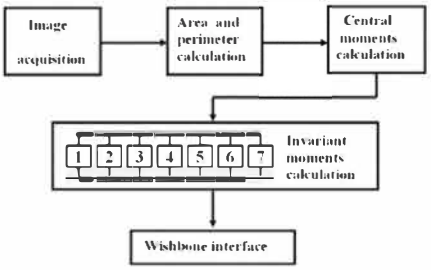

Fig.1. Block diagram of the architecture for invariant moments calculation

The interface to a PCI core is done through a wishbone interface, this Wishbone interface was created according to the wishbone standards, following the implementation available by Fernandez et al [10].

The system is distributed in nine modules, four of them used for floating point variables, three for the moment calculations and two for the Wishbone interface. A floating point synthesis package for VHDL based on the IEEE 754 standard was used for the calculations [11].

3.1. Area and X and Y average calculations.

To calculate the area of the object (in pixels), a *valid pixel counter* that generates a value of the amount of valid pixels is used. To calculate the central moments, the average of pixels in the X and Y directions are used as initial values. The procedure which calculates the area and the average of pixels in the X and Y directions is called *ARE*. The *ar* variable receives a new value at each clock cycle, updating the pixel sum coming from the memories with the actual pixel value. Simultaneously *mtpx* and *mtpy* are also calculated and updated at each clock cycle. *mtpx* stores the pixel values multiplied by the image column index, for an image of 512x512 pixels, while *mtpy* stores the pixel values multiplied by the image line index. When the procedure finds the end of the frame, by the *contlin* variable, it is closed

and the final values of *ar, mtpx* and *mtpy* are returned to the program. The final value of *mtpx* and *mtpy* correspond to the average values in the x and y directions respectively. A block diagram of the procedure to determine the area and the X and Y average values is illustrated in figure 2.

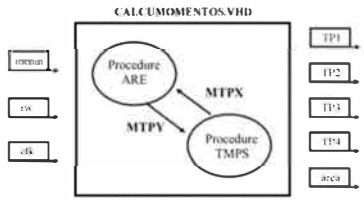

Fig. 2. Block diagram of the module for the area and the X and Y average calculations.

3.2. Central moments calculations

Once the area and the X and Y average calculations are completed, a new memory reading is performed in order to obtain the value of the image pixels in a procedure called TMPS. The values generated by the TMPS procedure are intermediate values that will be normalized by the area. The normalized values are obtained multiplying the sum of the image pixels by the pixel average values of the X and Y averages and the line and column indexes. When the procedure finds again the end of an image frame through the line counting variable *contlin*, the *tp1, tp2, tp3 e tp4* values are returned to the program to be used as inputs for the calculation of the central moments using the functions *n20, n02, n11, n30, n12, n21 and n03* present in the module called *INTERMED*.

3.3. Invariant moment calculations

Once the intermediate moments are calculated, the invariant moments determination is performed using sum and multiplications of the central moments. The invariant moments determination in performed in parallel, speeding the processing.

4. RESULTS

In order to validade the developed architecture, its results were compared with a Matlab implementation for the invariant moments calculation. The seven Hu invariant moments were determined using the MATLAB program for the objects shown in figure 3.

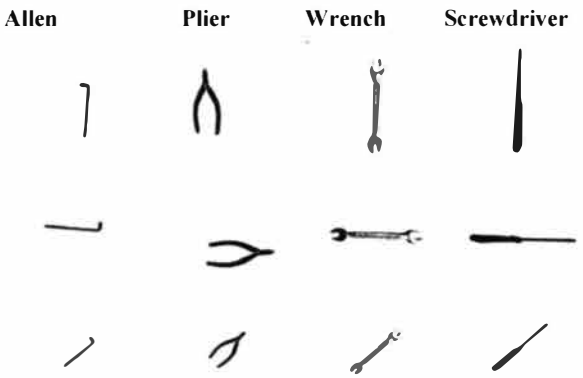

Fig. 3. Object images used to determine the Hu invariant moments.

Table 1 shows the Hu invariant moments calculated using the MATLAB program. It can be observed that using values with 4 decimal places it is possible to distinguish the objects. The invariant moments value could be fed to a classifier program to distinguish the objects.

Table 1. Hu invariant moments calculated using the MATLAB program.

Moment \Object	Allen	Plier	Wrench	Screwdriver
1st moment	0.0007	4.2600 to 4.2623	4.6538 to 4.6580	4.6270 to 4.6312
2nd moment	0.0000	8.5201 to 8.5245	9.3076 to 9.3159	9.2540 to 9.2625
3 rd moment	32.7408 to 34.3106	15.9496 to 15.9558	6.1502 to 6.1608	16.0871 to 16.0977
4 th moment	32.9791 to 33.3870	15.9496 to 15.9558	16.1502 to 16.1608	16.0871 to 16.0977
5 th moment	36.0437	31.8834 to 31.8957	32.2771 to 32.2976	32.1535 to 32.1742
6 th moment	16.8002 to 16.9562	20.2096 to 20.2181	20.8040 to 20.8187	20.7141 to 20.7289
7 th moment	36.0437	32.0694 to 32.0820	32.4768 to 32.4982	32.3481 to 32.3697

Previous results in order to validate the operation of the architecture were obtained using Altera´s Quartus II simulator software. Intermediate moment values were

inserted in the second software module and the result values were compared with the ones obtained by the Matlab´s moments determination program resulting in identical values.

An hardware implementation of the system was tested using a Stratix FPGA from Altera, the EP1S10F780C6. The FPGA device used has about 10.5K logic blocks and the NIOS board has expansion connectors which were used to read the output data trough a logic analyzer. The signals correspondent to the seven invariant moments calculations were collected by the logic analyzer in a bitwise form and converted to the floating point format to be compared with the results of the software simulation, resulting also in identical values.

Another interesting data obtained by the Quartus II software was the logical mapping of the chosen FPGA and a time estimative for the calculation, taking in account that the maximum time supported by the FPGA is 250Mhz.

It was possible to observe that the maximum estimated time between a pixel input and its output is 2771.546ns. Comparing the processing time for the invariant moments calculation in Matlab with the FPGA processing time there is a 33000% speed improvement which demonstrated that the dedicated hardware implementation is much more efficient. The improvement in processing time of the dedicated architecture over the Matlab´s implementation is because of the efficiency of the hardware versus software processing, the parallel computation of the invariant moments and the non optimized processing of Matlab, even considering that the dedicated hardware used is much simpler that the one used in the microcomputer that runs Matlab.

5. CONCLUSIONS

The calculation of invariant moments in reconfigurable logic is a challenge. Working with floating point routines in VHDL is something that has not been quite exploited, because VHDL has not support for floating point operations. The development of specific floating point libraries for VHDL widens the number of applications and opens the door for future developments. Although the development tools used for the reconfigurable devices were updated and the microcomputers used were of recent generation, system compilation and testing was slow and some times did not work properly because of its complexity. Nevertheless, the results obtained were good, considering the intrinsic difficulties. The speed improvement of the developed architecture in relation with a traditional microcomputer was a surprise. Even considering the large clock differences between the compared systems, the developed architecture was extremely faster for the dedicated calculations than the microcomputer. The speed increase validates the initial proposal that was one of the motivations to develop the work. Finally, we can consider that the work presented good results in face of the initial challenges, exploiting the new horizon of the use of complex mathematics in hardware in a new and functional manner.

Acknowledgements: The authors are grateful to CAPES to CNPQ and to DINACYT for the support throughout the work. The authors would like also to acknowledge the Electrical Engineering Department/EESC-USP and the Facultad de Ingenieria/Universidad de la Republica.

6. REFERENCES

[1] A.K. Jain, R.P.W. Duin and J. Mao, "Recognition: A review". IEEE Transactions on Pattern Analysis and Machine Intelligence 22, 1 (2000), 4–37.

[2] J.R. Parker (1994), Practical Computer Vision Using C, John Wiley & Sons, 1994.

[3] C. Yuceer and K. Oflazer, "A rotation, scaling and translation invariant pattern classification system", Pattern Recognition, 1993; 26(5): 687-710

[4] R.J. Prokop and A.P. Reeves, "A survey of moment-base techniques for unoccluded object representation and recognition", CVGIP: Graphical Models and Image Processing, 1992, vol. 54, no. 5, pp. 438-460.

[5] W.H. Wong, W.C. Siu and K.M. Lam, "Generation of moment invariants and their uses for character recognition.", Pattern Recognition Letters, 1995, vol. 16, no. 2, pp. 115-123.

[6] M. Hu, "Visual pattern recognition by moment invariants.", IEEE Transactions on Information Theory, 1962, vol. 8, no. 2, pp. 179-187.

[7] M. Brown and J. Rose, "FPGA and CPLD architectures: a tutorial", IEEE design and test of computers, 1996; vol. 13, no. 2, pp. 42-57.

[8] F.J. Bartos, "ASICs versus FPGAs", Control Engineering, 1995. , http://www.controleng.com/article/ CA607224.html, visited 09/03/2007.

[9] E.C. Pedrino "Morphological binary image real time pipeline architecture using Complex Programmable Logic Devices (in Portuguese)", MSc thesis in Electrical Engineering, University of Sao Paulo, Brazil, 2003.

[10] S. Fernandez, C. Mondueri and J.P. Oliver, "A platform for the development of the PCI bus (in spanish)", website http://mondueri.com/iiepci/, visited 07/03/2007.

[11] Floating-Point HDL Packages Home Page, website http://www.eda-stds.org/fphdl/, visited 07/03/2007.