RECONFIGURABLE ARCHITECTURE FOR MATHEMATICAL MORPHOLOGY USING GENETIC PROGRAMMING AND FPGAS

Emerson Carlos Pedrino, Osmar Ogashawara

Department of Computer Science
Federal University of São Carlos
Rod. Washington Luís, km235, SC, SP, Brazil
CEP: 13565905, Caixa Postal: 676
Tel: 55 16 33518232, Fax: 55 16 33518233
email: emerson@dc.ufscar.br,
osmaroga@ufscar.br

Valentin Obac Roda

Department of Electrical Engineering
University of São Paulo
Av. Trabalhador São-carlense, 400, SC, SP, Brazil
CEP: 13566590
Tel/Fax: 55 16 33739335
email: valentin@sel.eesc.usp.br

ABSTRACT

The task of designing manually morphological operators for a given application is not always a trivial one. Genetic programming is a branch of evolutionary computing and it is consolidating as a promising method for applications of digital image processing. The main objective of genetic programming is to discover how computers can learn to solve problems without being programmed for that. In the literature little has been found about the automatic morphological construction of operators using genetic programming. In this paper, it's presented an original architecture implemented in a FPGA for classical mathematical morphological (binary and gray level) operations that are generated automatically by a genetic programming approach. The possible applications for the system are: pattern recognition and emulation of simple filters, to name just a few. Practical examples using the developed system are presented.

1. INTRODUCTION

Morphological image processing is a nonlinear branch in image processing developed by Matheron and Serra in the 1960´s, based on the geometry and on the mathematical theory of order [1-2]. Morphological image processing has proved to be a powerful tool for binary and grayscale image computer vision processing tasks, such as edge detection, noise suppression, skeletonization, segmentation, pattern recognition and enhancement. Initial applications of morphological processing were biomedical and geological problems of image analysis. The basic operations in mathematical morphology are the dilation and the erosion, and these operations can be described by logical and arithmetic operators. Dilation and erosion morphological operators can be represented respectively by:

$$DnB(f)(x) = \{\sup[f(z)], z \in n(Bx)\}, \quad (1)$$

and

$$EnB(f)(x) = \{\inf[f(z)], z \in n(Bx)\}, \quad (2)$$

In Eq. (1), it's shown the dilation of image f at pixel x by the structuring element B of size n. In Eq. (2), the corresponding erosion operation is obtained by replacing the *sup* by *inf*. Some other basic computer vision operations such as edge detection, skeletons and noise elimination can be performed eroding or dilating objects in an algorithmic way.

The design of morphological procedures is not a trivial task in practice [3]. It is necessary to have some expert knowledge to properly select the structuring element and the morphological operators to solve a certain problem [4]. Genetic programming (GP) is the most popular technique for automatic programming nowadays and may provide a better context for the automatic generation of morphological procedures. GP is a branch of evolutionary computation and artificial intelligence, based on concepts of genetics and Darwin's principle of natural selection to genetically breed and evolve computer programs to solve problems [5]. In the literature, there are few applications of GP for the automatic construction of morphological operators. Thus, we propose a linear genetic programming approach for the automatic construction of morphological, arithmetic and logical operators, by means of a toolbox named *morph_gen* for the Matlab program, implemented in this work. The proposed toolbox can be used for the design of non linear filters, image segmentation and pattern recognition of binary and gray level images. In addition to toolbox, the instructions generated by the toolbox are transferred to a pipeline architecture developed in this work, which has been implemented on a FPGA. Some examples of applications are presented and the results are discussed and compared with other approaches.

978-1-4244-6534-7/10/$26.00 ©2010 IEEE

This article is organized as a brief review of the basic concepts of morphological operations and genetic programming, section 1; a description of the developed system, section 2; results and application examples are presented in section 3; section 4 presents the conclusions.

2. PROPOSED SYSTEM

The proposed algorithm developed in this paper for automatic construction of morphological operators uses a linear genetic programming approach that is a variant of the GP algorithm that acts on linear genomes [6]. The developed algorithm operates with two input images, an original image and an image containing only features of interest which should be extracted from the original image. The genetic procedure looks for operator sequences in the space of mathematical morphology (binary and gray-level) algorithms that allows extracting the features of interest from the original image. The operators are predefined procedures from a database that work with particular kinds of structuring elements having different shapes and sizes. It is also possible to include new operators in the data base when necessary. The program output is a linear structure containing the best individual of the final population. The output result from one operator is used as input to the subsequent operator and so on. The genetic algorithm parameters are supplied by the user using a graphical user interface (GUI). The main parameters are: tree depth, number of chromosomes, number of generations, crossover rate, mutation rate, reproduction rate and certain kinds of operators suited to a particular problem. It has been used for the problems the mean absolute error (MAE) as a fitness measure. The fitness function using MAE error was calculated as follows:

$$d(a,b) = \frac{1}{XY}\sum_{i}^{x}\sum_{j}^{y} | a(i,j) - b(i,j) | \qquad (3)$$

In Eq. (3), *a* is the resulting image evaluated by a particular chromosome (program) and *b* is the goal image. The chromosomes are encoded as variable binary chains. The genetic parameters and the images are supplied by the user; the initial population of programs is randomly generated. Since the chromosomes are encoded as binary chains, if the user has selected the instructions: *and* (AND logic), *sto* (STORE), *ero* (EROSION) and *cpl* (COMPLEMENT), the first operator will be coded as "00_2", the second as "01_2", the third as "10_2" and the last as "11_2". If the tree depth chosen was four, for example, the chromosome: "00011011_2" could be created. The evaluation of this chromosome will be: CPL(ERO(A AND A)), where A is an input image.

After evaluation of each chromosome in a generation, a fitness value is assigned to each one using Eq. (3). The selection method used for genetic operators was the tournament selection [7]. In crossover operation, morphological operators are randomly selected and exchanged between parents chromosomes. The mutation operation replaces a randomly selected instruction by another in the range of morphological algorithms space. The reproduction operator simply copies a single parent into the new generation according to its fitness value.

The block diagram of the developed architecture can be seen in Fig. 1. The opcodes (best chromosome) file ("program.mif") generated by the Matlab is transferred with the other project files containing the description of the architecture to the FPGA board by means of the USB interface from a PC. In this project the DE2 board from Altera was used to develop the video architecture that is based on 32-stage pipeline. The DE2 board contains a Cyclone II (2C35) FPGA, a NTSC/PAL TV decoder circuit and a VGA output circuit. A composite video supplied by a commercial video camera is de-interlaced and converted to 10 bit RGB data (640x480 pixels) through a video decoder stage. The RGB frames are processed through the pipeline stages and the results are converted to an analog format again through DA converter. Then, the processed images can be shown in a VGA monitor. A 27MHz oscillator was used as a clock source.

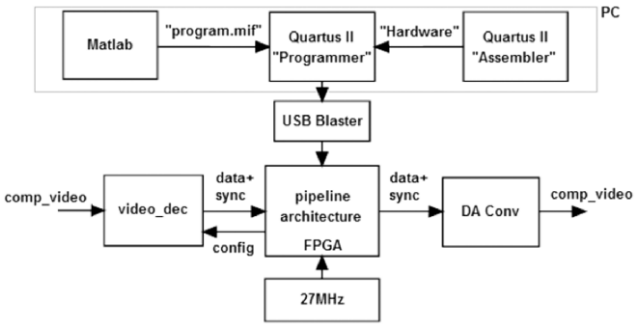

Fig. 1. Block diagram of the developed system. After FPGA programming, the composite video is de-interlaced by the *video_dec* and the 10-bit RGB data produced are processed by the pipeline architecture. The results are converted again to an analog format to be shown in a VGA monitor.

In Fig. 2 a block diagram of the pipeline stages is presented. The opcodes from *morph_gen* toolbox are loaded into the stages through a state machine named ROM that contains the original program (best chromosome of a certain kind of problem). The state machine ROM uses the bus *dat* and the bus *add* to distribute the data (instructions) to each processor that uses an add (address) in the architecture. For example, the P1 has the add=01h, the P2 has the add=02h and so on. The "program.mif"

contains a binary chain representing the chromosome generated by Matlab where each line corresponds to an instruction according to Table 1 that will be processed by each stage. After the reset process of the architecture, while the *end_add* pin is low, the state machine loads the instructions referring to a certain problem into the instruction register (IR) of each processor of the pipeline. When the load process ends, the *end_add* pin will be high and the state machine will indicate the time of video processing and this cycle will be repeated when the state machine reads a reset state again.

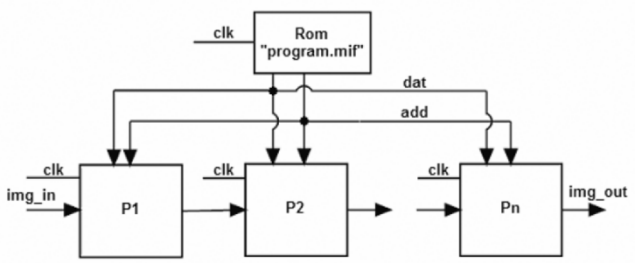

Fig. 2. Block diagram of the pipeline stages.

Table 1. Implemented instructions.

Opcode	Comment
nop	No operation. The input image is copied to the output image.
dil_q_3	The input image is dilated by a (3x3) square structuring element.
ero_q_3	The input image is eroded by a (3x3) square structuring element.
dil_c_3	The input image is dilated by a (3x3) circular structuring element.
ero_c_3	The input image is eroded by a (3x3) circular structuring element.
dil_h_3	The input image is dilated by a (1x3) horizontal structuring element.
ero_h_3	The input image is eroded by a (1x3) horizontal structuring element.
dil_v_3	The input image is dilated by a (3x1) vertical structuring element.
ero_v_3	The input image is eroded by a (3x1) vertical structuring element.
dil_dd_3	The input image is dilated by a (3x3) right diagonal structuring element.
ero_dd_3	The input image is eroded by a (3x3) right diagonal structuring element.
dil_de_3	The input image is dilated by a (3x3) left diagonal structuring element.
ero_de_3	The input image is eroded by a (3x3) left diagonal structuring element.
xor1	Exclusive-Or between the input image and a temporary image (previous result).
cpl	Logical complement of the input image.
sto1	Temporary storage of the input image.
and1	Logical AND between the input image and a temporary image (previous result).
or1	Logical OR between the input image and a temporary image (previous result).
ldi	Load of the original image.
cpl_cz	Complement of the input image (gray-scale).
add_cz	Arithmetic sum between the input image (gray-scale) and a temporary image (previous result).

Each stage stores two adjacent 640-pixel lines followed by a 3-pixel line to constitute a (3x3)-input to a morphological processor implemented in that stage. This can be accomplished through a shift register. This same structure is used by a previous stored result that is delayed by each stage, too. This result is stored in *img_temp* register. Fig. 3 shows the block diagram of a stage. The *Instr_dec* block in the processor decodes the instruction stored in IR register and apply an morphological, arithmetic or logical operation, according to Table 1, to input pixels *p1..p9* and/or *s2_2* (previous stored result from *n-1* stage). In *Instr_dec* block, the dilation and erosion operations are implemented according to equations 1 and 2, respectively. The instructions have been implemented using Verilog HDL.

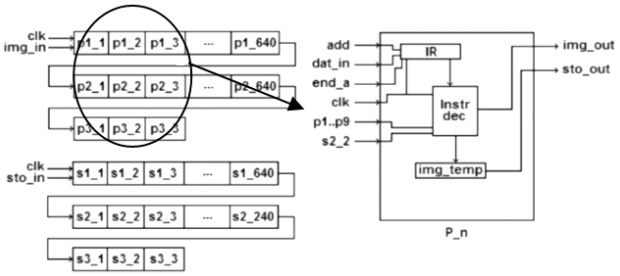

Fig. 3. Block diagram of a stage. The ellipse represents the input pixels of a stage.

3. RESULTS

In this section we present some results obtained by means of the proposed system. In Fig. 4 it is presented an input image and a training image containing features (heads) from a fragment of a music score to be extracted by the evolutionary system. The genetic procedure found the following best program to extract heads using the developed system: "dil_dd_3-> dil_de_3-> dil_dd_3-> dil_v_3-> dil_v_3-> dil_dd_3-> dil_v_3-> ero_q_3-> ero_v_3-> ero_q_3-> ero_c_3". The genetic parameters chosen for this task were: 50 generations, 50 chromosomes, tree of depth 12, crossover rate of 97%, mutation rate of 3% and reproduction rate of 10%. The MAE error found between the goal image and the obtained result was less than 1,1%. The training time was less than 12 min in Matlab and execution time was in real time in the proposed system.

In Fig. 5 there is an example of emulation of the Photoshop's Glowing Edge Filter (gray level) generated automatically for the following parameters: 51 generations, 70 chromosomes, tree of depth 9, crossover rate of 95%, mutation rate of 20% and reproduction rate of 10%. For this task was used an intensity image as input and the best program found was: "add_cz-> add_cz-> dil_c_3-> sto1-> cpl_cz-> dil_c_3-> dil_c_3-> add_cz-> sto1". The MAE error found was less than 6,2% compared to a Photoshop's result. The training time was less than 11 min in Matlab and

execution time was in real time by means of the developed hardware.

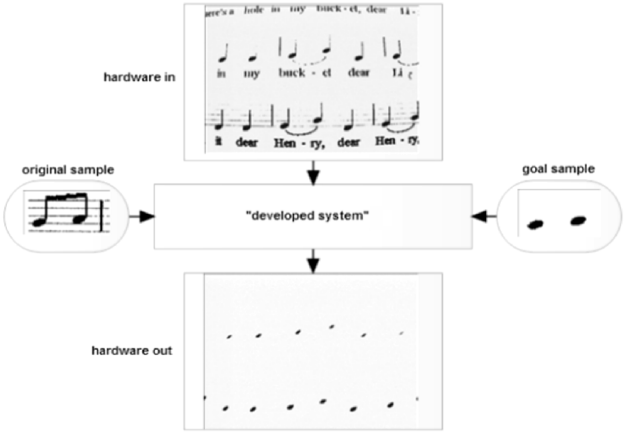

Fig. 4. Obtained result by developed hardware for head extraction in real time.

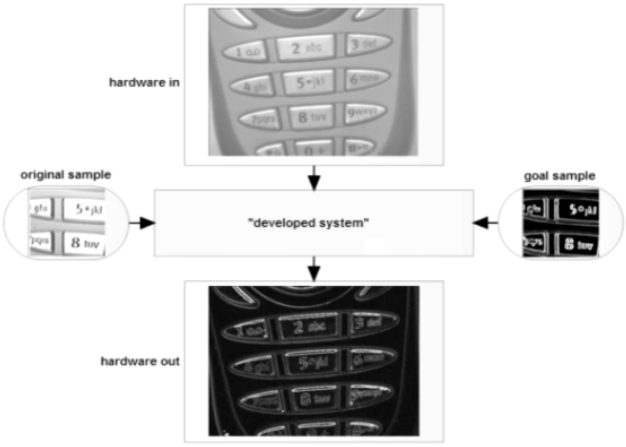

Fig. 5. Emulation of the Photoshop's Glowing Edge Filter (gray-level) in real time.

Table 2 summarizes the obtained results and Table 3 summarizes the FPGA used resources. The spatial resolution of the processed images is 640x480 pixels. The system is capable of processing images at 60 frames/s.

4. CONCLUSIONS

In this paper was presented the implementation of an architecture dedicated to classical mathematical morphology operations (binary and gray level) where the instructions are generated automatically by an approach based on genetic programming and the application solutions are stored in a FPGA. The implemented architecture is capable of processing morphological operations in real time. The spatial resolution of the images is 640x480 pixels and the processing rate is 60 frames/s. The implemented system has among the possible applications: pattern recognition and filter emulation, to name just a few. Examples were presented in order to demonstrate the performance of the system for fast image processing. In future is intended to implement the training genetic procedure in hardware.

Table 2. Summary of the results.

Parameters	**Head Extraction**	**Glowing Edge Filter**
Generations	50	51
Chromosomes	50	70
Tree depth	12	9
Cross. rate	97 %	95 %
Mut. rate	3 %	20 %
Repr. rate	10 %	10 %
MAE error	1.1 %	6.2 %
Training time (Matlab)	12 min.	11 min.
Execution time (Hardware)	60 frames/s	60 frames/s

Table 3 Summary of the FPGA device.

Device: Cyclone II EP2C35F672C6 / Application	**Pins**	**Memory Bits**	**LEs (Logic Elements)**
Head extraction	125 (26%)	134208 (28%)	2702 (8%)
Glowing Edge	125 (26%)	264944 (55%)	5672 (17%)

5. REFERENCES

[1] E. R. Dougherty, *An introduction to Morphological Image Processing*, SPIE, Bellingham, Washington, 1992.

[2] J. Serra, *Image Analysis and Mathematical Morphology*, Academic Press Inc, California, 1982.

[3] S. Marshall, N. R. Harvey, D. Greenhalgh, "Design of Morphological Filters using Genetic Algorithms," *EUSIPCO*, Tampere, Finland (2000).

[4] M. I. Quintana, R. Poli, E. Claridge, "Genetic programming for mathematical morphology algorithm design on binary images," *In M. Proceedings of the International Conference KBCS*, 161-171 (2002).

[5] J. Koza, *Genetic Programming*, MIT Press, 1992.

[6] M. Oltean, C. Grosan, M. Oltean, "Encoding Multiple Solutions in a Linear Genetic Programming Chromosome," *International Conference on Computational Science,* 1281-1288 (2004).

[7] D. Goldberg, K. Déb, "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," *In: Foundations of Genetic Algorithms* [edited by G. Rawlins]. Morgan-Kaufmann, 69-93, (1991).