Bruno PADUA^2, Denis LIMA^2, Marcio FERNANDES^2,Ali ABUASSAL^1,Gianluca TEMPESTI^1, Emerson PEDRINO^{1,2}

Department of Electronic Engineering, The University of York, York, UK (1),Department of Computer Science, Federal University of Sao Carlos, Sao Carlos, Brazil (2)

doi:10.15199/48.2020.04.02

Hybrid Implementation of Evolutionary Algorithms in FPGAs for Automatic Generation of Morphological Image Filters

Abstract. *This paper proposes a novel hybrid software/hardware system to automatically create filters for image processing based on genetic algorithms and mathematical morphology. Experimental results show that the hybrid system, implemented using a combination of a NIOS-II processor and a custom hardware accelerator in an Altera FPGA device, is able to generate solutions that are equivalent to the software version in terms of quality in approximately one third of the time.*

Streszczenie. *W artykule zaproponowano nowe hybrydowe oprogramowanie do automatycznego tworzenia filtrów grafiki bazujących na algorytmach genetycznych i morfologii matematycznej. Eksperymenty wykazały że proponowany system wykorzystujący procesor NIOS-II i Altera FPGA jest w stanie generować rozwiązanie niemal trzy razy szybciej niż dotychczas stosowane systemy.****(Wykorzystanie hybrydowego algorytmu bazującego na algorytmach genetycznych do automatycznego generowania filtrów grafiki.)***

Keywords: Image processing, Genetic algorithms, Reconfigurable architectures, FPGA, Morphological operations.
Słowa kluczowe: przetwarzanie obrazu, algorytmy genetyczne, FPGA, operacje morfologiczne

Introduction

Evolutionary Algorithms (EAs), inspired by biological evolution [1] [2] [3], were created as optimization tools for engineering problems. Several papers in the literature deal with the use of EAs for Digital Image Processing (DIP), both in the software and hardware domains [4] [5] [6] [7]. In this context, given a set of training images, an EA aims to find an individual that represents a sequence of linear and non-linear operators used in DIP, with the objective of extracting some characteristics from the images.

Using traditional methods, the construction of image filters for a given task is non-trivial in practice, even for a domain specialist [8] [9]. Thus, EAs can be used to automatically find a sequence of image operators for the solution of a practical problem involving DIP. Because the intelligent training of such filters is usually a costly process in computational terms, some designers of hybrid systems implement one phase of the EA in software (Training Phase) and another in hardware (Calculation of Fitness Phase) exploiting FPGA (Field Programmable Gate Array) platforms [10].

This article presents a novel approach and implementation of a hybrid FPGA architecture for the automatic construction of morphological image filters, by means of a Genetic Algorithm (GA). This approach is an optimized alternative to a Matlab software implementation. Experimental results have indicated the practical feasibility of the proposed hybrid approach.

The remainder of this paper is organized as follows: section Genetic Algorithm and section Mathematical Morphology. Sections Software algorithm and Hardware algorithm are concerned with the development and implementation of GAs in software and hardware, respectively. The experimental results for the proposed approach are presented in section 0.9, before a final conclusions drew in section 0.10.

GENETIC ALGORITHMS

Genetic algorithms, originally developed by Holland [1], draw inspiration from biological evolution, using mutation and crossover operators to search for optimal or near-optimal solutions in a wide variety of tasks.

Initially, a random population of individuals (chromosomes) is generated, where each individual represents a candidate solution to the given problem. Each chromosome consists of genes, commonly represented by a vector of binary numbers (integers or real). The vector encodes information about the individual, such as the general characteristics of the decoded individual (phenotype) or the presence or absence of specific characteristics (genes).

Each individual is tested through a fitness function, which calculates how apt it is to solve the problem. The fitness determines the probability of the individual to reproduce itself by means of crossover and mutation operations, generating offspring for the next generations. This process implies that, eventually, the fittest individual should represent an optimal or near-optimal solution for the problem under consideration.

The most significant challenge for a genetic algorithm to work satisfactorily is the efficient coding of the individuals that will represent the given problem [2]. Another important consideration is that the fitness function should be specific to each type of problem, and this is crucial for the appropriate selection of individuals each generation.

MATHEMATICAL MORPHOLOGY FOR DIGITAL IMAGE PROCESSING

According to Gonzalez & Woods [4], an image can be defined as a two-dimensional function $f(x, y)$, where x and y are spatial coordinates of an image point, and the amplitude f of a point defines its intensity or gray level.

In digital image processing, filters are used to correct, smooth, and/or highlight details, or even extract features from an image. Mathematical Morphology (MM) is a DIP approach that defines operations which can be used to extract shape characteristics.

The main idea of MM [11] [12] is to exploit Set Theory to process images. Information regarding image geometry and topology can then be extracted by means of geometrical transformation operations, using a known set called Structuring Element (SE).

Two morphological operations are fundamental in this context: *dilation* and *erosion*. Dilation is defined by Equation 1, where A and B are two sets of Z^2, with A being the original image, B the structuring element, and $\varnothing$ an empty set [4].

$$A \oplus B = \{x | (B)_x \cap A \neq \varnothing\} \quad (1)$$

Thus, *dilation* can be defined as the reflection of B around its origin, combined with a translation of x. Therefore, dilation is the set of all x's displacements, in which the reflection of B and A overlap on at least one non-zero element.

Erosion, on the other hand, is defined by Equation 2, where A and B are two sets of images, where A is the original image, B the structuring element, and $\varnothing$ is an empty set [4].

(2) $$A \Theta B = \{x | (B)_x \subseteq A\}$$

Thus, erosion is the set of all x's displacements such that the translation of B by x is contained in A. By means of these basic operators, it is possible to perform more complex operations on images, such as: construction of filters, segmentation or pattern recognition [13].

Other types of operations that can be applied to images are the so-called *logical operations*. These are unique to binary images; and consist of applying operations AND, OR, NOT, XOR, and NOT to each image point.

SOFTWARE ALGORITHM IMPLEMENTATION

To accelerate the development of the proposed algorithm, it was initially implemented using MATLAB and its digital image processing toolbox. The algorithm uses two morphological operators (erosion and dilation), four logical operators (AND, OR, NOT, XOR), a NOP operator and a memory operator (STO - Store), as shown in Table 1.

Table 1. Operators used by the proposed algorithm and some structuring elements used by the procedure.

INDEX	OPERATION	STRUCTURING ELEMENTS
0	- - - - - - -	[0 0 0;1 1 1;0 0 0]
1	EROSION	[1 1 1;1 1 1;1 1 1]
2	DILATATION	[0 1 0;1 1 1;0 1 0]
3	NOP	[1 0 0;0 1 0;0 0 1]
4	AND	[0 0 1;0 1 0;1 0 0]
5	OR	[0 1 0;0 1 0;0 1 0]
6	XOR	[1 0 1;0 1 0;1 0 1]
7	NOT	- - - - - - -
8	STO	- - - - - - -

The system has a graphical interface to define parameters such as number of generations, population size, initial chromosome size, maximum chromosome size, crossover rate and mutation rate, in addition to the choice of which operators and structuring elements will be used by the designer.

The proposed algorithm automatically creates an image filter based on the training and test sets containing the original and desired images; (which are also parameters for the procedure). A block diagram of the system is shown in Fig. 1, followed by additional details regarding the five steps involved (A to E).

0.1 Generating an initial population

The creation of each individual in the initial population (Fig. 1.A) is done randomly using integer values of f_i and e_i, which are respectively the values for the operation index and structuring elements according to Table 1. These are stored in a vector composed of genes in Fig. 2, according to a parameter defining the initial chromosome size.

Once the genes are generated, they are stored in a MATLAB cell data structure, which represents a chromosome corresponding to an individual. Once all individuals are generated, they are stored as the initial population.

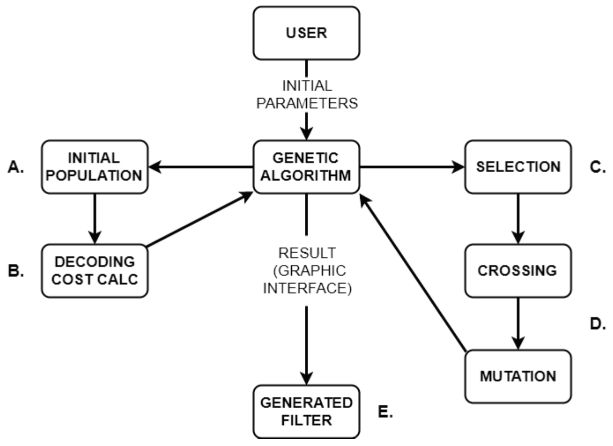


Fig. 1. Block diagram representation of the proposed solution based on a genetic algorithm.

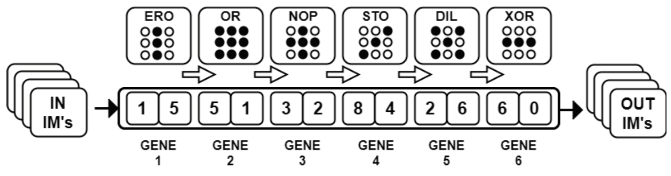


Fig. 2. Chromosome format adopted by the proposed system. Table 1 gives the used indexes.

0.2 Decoding and evaluating an individual

After a new population is generated (the first and any subsequent one), it is necessary to decode each individual and evaluate them. The evaluation is based on the error of the generated filter compared to the desired result (Fig. 1.B).

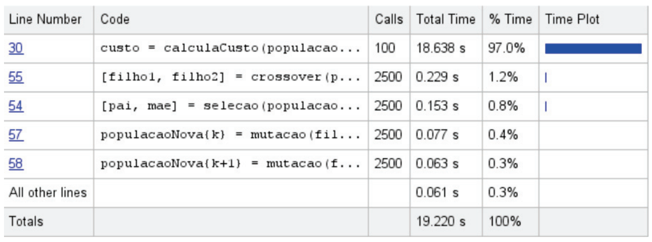
The sequence of operations decoded from the chromosome (Fig. 2), is applied to the input image, generating an output image. Based on output image and the desired image, the error is calculated by the appropriate fitness function. The calculated error is subsequently stored in an error vector, in which each position is associated to a given individual from the population.

The fitness function used by the proposed software system is defined by Equation 3, where the parameter *desired* corresponds to the desired image, and the parameter *result* corresponds to the output image after the application of the generated filter.

(3) $$error = \sum_{i=0}^{N} \sum_{j=0}^{M} |desired_{i,j} - result_{i,j}|$$

0.3 Selection

In the implemented GA, elitism and tournament selection techniques were used (Fig. 1.C). Elitism consists in keeping one or more of the k best individuals of each generation in the next population. Selection by tournament is used to select the individuals who will be parents in the operations of crossover and mutation of the GA. In this method, a number *n* of individuals are randomly chosen from the population, and from those, the two best ones are chosen, based on their corresponding errors.



Line Number	Code	Calls	Total Time	% Time	Time Plot
30	`custo = calculaCusto(populacao...`	100	18.638 s	97.0%	▬
55	`[filho1, filho2] = crossover(p...`	2500	0.229 s	1.2%	|
54	`[pai, mae] = selecao(populacao...`	2500	0.153 s	0.8%	|
57	`populacaoNova(k) = mutacao(fil...`	2500	0.077 s	0.4%	
58	`populacaoNova(k+1) = mutacao(f...`	2500	0.063 s	0.3%	
All other lines			0.061 s	0.3%	
Totals			19.220 s	100%	

Fig. 3. MATLAB profiler tool showing the time spent in the costing function of the proposed genetic algorithm.

0.4 Crossover and Mutation

In order to generate new individuals exhibiting genetic diversity, two operators can be used by a GA: *crossover* and *mutation* (Fig. 1.D).

The implemented GA uses two-point crossover: a value between 0 and 1 is drawn, and if the value drawn is less than or equal to the crossover rate, two new children are generated.

In the mutation, a number between 0 and 1 is also drawn, and if this is less than or equal to the mutation rate, the value of a gene position is changed to a random integer value in the corresponding domain. This value must be contained in the group of possible indexes, either for operations or for structuring elements, according to Table 1. This process is repeated for all the chromosome genes.

0.5 Filter generation

After the GA executes a predetermined number of generations, the algorithm then decodes the result and displays it through its graphical interface (Fig. 1.E).

HARDWARE ALGORITHM IMPLEMENTATION

As shown in Fig.3, by using the MATLAB profiler tool, it is possible to demonstrate that the main performance bottleneck of the genetic algorithm implemented in software is the calculation of the cost function, which accounts for 97% of the total processing time. This finding justifies the proposed hardware implementation for this phase, establishing a hybrid overall solution for the problem.

Based on the GA implementation in software and corresponding results, an architecture was implemented for the automatic creation of image filters using the Altera's Quartus II development tools.

The proposed architecture consists of two main parts:

i) a custom processor implemented in Verilog for the application of image filters and calculation of the error between the result and the desired image.

ii) a Nios II processor for the implementation of the genetic algorithm in C language.

A block diagram of the prototype architecture is presented in Fig. 4.

0.6 Image processor in Verilog

Considering that the main performance bottleneck for the proposed GA is the calculation of the error between the result generated by the image filter application and the desired image, a custom processing unit was created in Verilog to perform this task (Fig. 4.A).

A state machine was created to perform the tasks of a common processor: fetch of each instruction of the generated chromosome, decode, and execution.

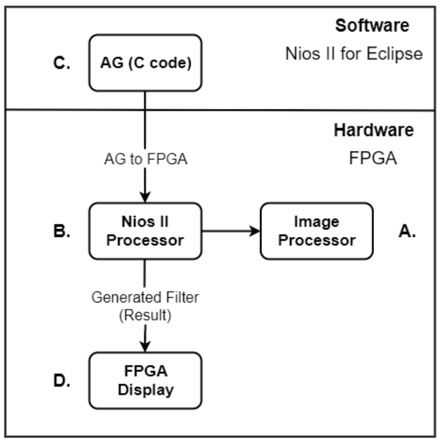


Fig. 4. Block diagram representation of the proposed hybrid architecture.

The original and desired images are then stored in files and loaded into memory. After applying the image filter generated by the GA, the processor calculates the cost of the process. The fitness function used by the processor is a modification of Equation 3, given by Equation 4.

$$error = \sum_{i=0}^{N}\sum_{j=0}^{M} desired_{i,j}\ XOR\ result_{i,j} \tag{4}$$

It is also worth noting that the operations on images implemented in Verilog are the same as those of Table 1 used by the GA implemented in MATLAB.

0.7 Nios II processor

Since the implementation of a complete GA in hardware can be very complex (e.g, for the generation of random individuals) and does not provide significant speedup compared to a much simpler software implementation, the proposed system uses a Nios II processor to implement the GA using the C language (Fig. 4.B), as already mentioned.

Nios II is a 32-bit RISC processor, designed to be synthesized in an FPGA. Its advantage is the ability to run programs in C / C++ language through Nios II for Eclipse, and the flexibility of communication with other system components.

0.8 GA implemented in C

The GA executed in the Nios II processor (Fig. 4.C), is a simplified version adapted from the one implemented in MATLAB.

The main difference between the two implementations is the cost calculation: while in MATLAB the application of image filters exploits tools presented in the toolbox functions, in the C version the calculation is fully performed by the custom processor running in the FPGA hardware.

The communication between the GA and the image processor is handled by the Nios II processor bus, using library functions specially created for this purpose.

0.9 GA results visualization

The GA implemented in C also includes a function for decoding the generated result. Using the Nios II library functions as well, the result is displayed directly on the seven-segment FPGA displays.

Experimental Results

0.10 Software tests

To benchmark the performance of our approach, we have set the GA parameters to the same values used in some previous papers found in the literature [7] [8].

A set of training and test images pairs was artificially generated using MATLAB, producing random images with features such as circles, disks, stars, and squares. Sets of images in three different resolutions were also created, where the set for each feature consists of 60 pairs of training images, and 40 pairs of test images. Selected operations (erosion, dilation, AND, OR, NOT, XOR, NOP, and STO) were used along with 3x3 structuring elements.

A population of 50 individuals was used, and the algorithm was run for 200 generations, with a crossover rate of 0.9, and mutation rate of 0.2. Also, the initial size of the chromosome was 4, and the maximum size was set to 20.

Initially, the software-only implementation was benchmarked. Table 2 presents a comparison of the obtained results against those obtained by Quintana [7], where a similar methodology was employed for the GA.

The GA software implementation presents advantages for all of the tested features, i.e, the characteristics of interest (squares, disks, circles and stars), and different images sizes, as shown in Table 2.

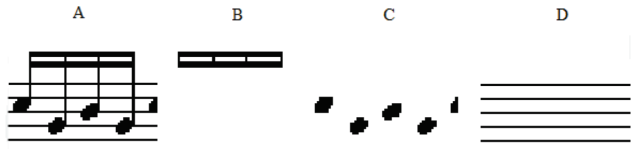


Fig. 5. Characteristics of interest in the test image set: (A) Original Image; (B) Beams; (C) Noteheads; (D) Staffs.

In the tests performed in software by Quintana [7], the set formed by musical notes was the one demanding the longest processing time for the algorithm evolution phase.

Due to this characteristic, a similar test set was created to compare the GA in software against the hardware implementation proposed in this work.

An image set with 64x64 resolution was created, consisting of three images with three different characteristics: beams, note heads, and staffs, referring to musical notes in a score. In this set, the features of interest for each image were manually removed using editing tools, as shown in Fig. 5. Other GA parameters were chosen based on other papers found in the literature [7] [8].

Once again, the operations employed by the GA were erosion and dilation, and structuring elements of size 3x3, as defined in Table 1. The values for crossover and mutation were also maintained, at 0.9 and 0.2 respectively, for a population of 50 individuals, and the GA algorithm was set to run for 100 generations. The chromosome size was fixed to 16 genes.

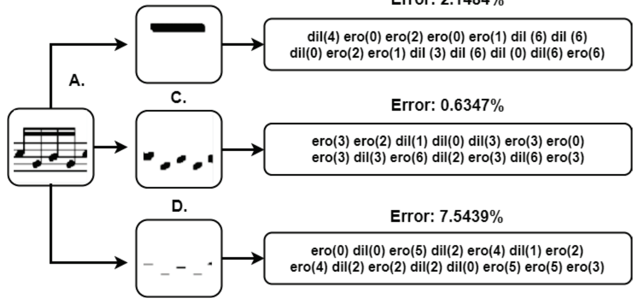

B.
Error: 2.1484%
dil(4) ero(0) ero(2) ero(0) ero(1) dil (6) dil (6) dil(0) ero(2) ero(1) dil (3) dil (6) dil (0) dil(6) ero(6)
A.
C.
Error: 0.6347%
ero(3) ero(2) dil(1) dil(0) dil(3) ero(3) ero(0) ero(3) dil(3) ero(6) dil(2) ero(3) dil(6) ero(3)
D.
Error: 7.5439%
ero(0) dil(0) ero(5) dil(2) ero(4) dil(1) ero(2) ero(4) dil(2) ero(2) dil(2) dil(0) ero(5) ero(5) ero(3)

Fig. 6. Best Filters in Hardware: (A) Original Image; (B),(C) and (D): Desired Images.

Table 2. Results comparison: proposed system x Quintana's system

		Image Pair Resolution					
	Stats	100x75		300x225		640x480	
		(1)	(2)	(1)	(2)	(1)	(2)
SQUARE	MEAN	0.870	0.954	0.864	0.979	0.843	0.992
	SD	0.087	0.028	0.079	0.014	0.076	0.005
	Min	0.761	0.885	0.781	0.934	0.748	0.976
	Max	0.944	0.984	0.933	0.990	0.930	0.996
DISKS	MEAN	0.868	0.895	0.807	0.955	0.870	0.989
	SD	0.074	0.021	0.118	0.016	0.059	0.003
	Min	0.746	0.868	0.642	0.917	0.811	0.983
	Max	0.947	0.945	0.935	0.984	0.935	0.994
CIRCLES	MEAN	0.960	0.992	0.900	0.996	0.896	0.998
	SD	0.043	0.002	0.047	0.001	0.048	0.001
	Min	0.862	0.986	0.850	0.994	0.845	0.997
	Max	0.948	0.994	0.944	0.996	0.939	0.999
STARS	MEAN	0.922	0.997	0.921	0.999	0.857	1.000
	SD	0.028	0.001	0.025	0.001	0.105	0.001
	Min	0.893	0.996	0.895	0.998	0.641	0.999
	Max	0.951	0.998	0.948	0.999	0.943	1.000

(1) QUINTANA (2) Current Paper.

The filters generated by the algorithm in hardware were again used in MATLAB only for the display and comparison of the results presented in this paper. Fig. 6 shows the best filters obtained in hardware, including the characteristic of interest, the error obtained and the chromosome representing the generated filter.

In Fig. 7 the best filters that obtained in software is shown, including characteristics of interest, the error obtained and the chromosome representing the generated filter.

Finally, a time comparison was made between the time

Table 3. Performance comparison: hybrid system x software.

Implementation	Task	Time(s)
Hybrid	Beans	2
Software	Beans	6
Hybrid	Noteheads	2
Software	Noteheads	6
Hybrid	Staffs	2
Software	Staffs	6

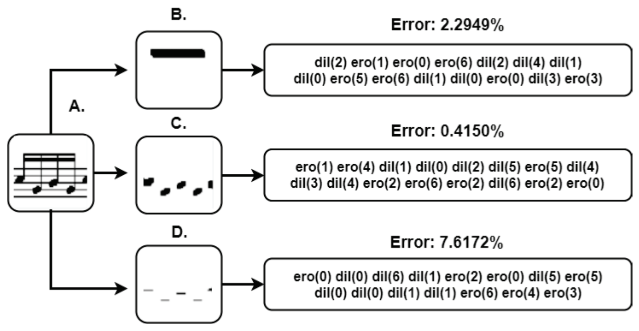

B.
Error: 2.2949%
dil(2) ero(1) ero(0) ero(6) dil(2) dil(4) dil(1)
dil(0) ero(5) ero(6) dil(1) dil(0) ero(0) dil(3) ero(3)
A.
C.
Error: 0.4150%
ero(1) ero(4) dil(1) dil(0) dil(2) dil(5) ero(5) dil(4)
dil(3) dil(4) ero(2) ero(6) ero(2) dil(6) ero(2) ero(0)
D.
Error: 7.6172%
ero(0) dil(0) dil(6) dil(1) ero(2) ero(0) dil(5) ero(5)
dil(0) dil(0) dil(1) dil(1) ero(6) ero(4) ero(3)

Fig. 7. Best filters obtained in Software (A) Original Image; (B),(C) and (D): Desired Images.

spent in the GA evolution to generate the desired image filters between the software version in MATLAB, and the hybrid architecture created. As seen in Table 3, the GA implemented in software takes about six seconds to generate the result, whereas in hardware it takes about two seconds, i.e., an overall reduction of two thirds of the processing time.

CONCLUSIONS

The use of GAs has proved to be a valuable technique for problems in several knowledge areas. Among these areas, digital image processing can benefit from the use of GAs for the creation of filters to improve the information obtained from images, and also for pattern recognition. The manual construction of such filters is not a trivial task, which has motivated the development of this work.

In this paper, a hybrid system was proposed in order to automatically create filters for image processing based on genetic algorithms and mathematical morphology. Initially, the whole scheme was implemented in software, using the MATLAB tool. However, the cost function of the GA algorithm proved to be very expensive in terms of processing time, leading to the development of a hardware module, implemented in FPGA.

The resulting solution can be characterized as a hybrid system, with software modules running on an embedded NIOS-II processor, and the cost module running on a hardware custom processor.

Experimental results have shown that the results of the two approaches (software and hybrid) are generally equivalent in terms of quality, but in terms of performance the hybrid system proved to be 3 times faster than the all-software solution. Thus, the results indicates the feasibility of using a hybrid solution to automatically generate filters for DIP tasks using genetic algorithms and mathematical morphology.

Acknowledgements

This work was supported by funding from FAPESP (Grant 2017/26421-3).

***Authors**: Bruno Padua, Prof. Emerson Pedrino, M. Sc. Denis Lima, Prof. Marcio Fernandes, Computer Science Department, Federal University of Sao Carlos, SP., Brazil, email: emerson@dc.ufscar.br. M. Sc. Ali Abuassal, Prof. Gianluca Tempesti, Department of Electronic Engineering, The University of York, York, UK.*

REFERENCES

[1] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, re-issued by MIT Press, 1992.
[2] M. Mitchell, *An Introduction to Genetic Algorithms*. Cambridge, MA, USA: MIT Press, 1999.
[3] J. R. Koza, *Genetic programming: On the programming of computers by natural selection*. Cambridge, Mass.: MIT Press, 1992.
[4] R. E. W. R. C. Gonzalez, *Digital Image Processing*. Upper Saddle River, New Jersey, Prentice Hal, 2002.
[5] O. M. Filho and H. V. Neto, *Processamento Digital de Imagens*. Rio de Janeiro: Brasport, 1999.
[6] J. Wang, "Design and implementation of a virtual reconfigurable architecture for different applications of intrinsic evolvable hardware," *IET Computers & Digital Techniques*, vol. 2, pp. 386–400(14), September 2008. [Online]. Available: https://digital-library.theiet.org/content/journals/10.1049/iet-cdt_20070124
[7] M. I. Quintana, R. Poli, and E. Claridge, "Morphological algorithm design for binary images using genetic programming," *Genetic Programming and Evolvable Machines*, vol. 7, no. 1, pp. 81–102, Mar 2006. [Online]. Available: https://doi.org/10.1007/s10710-006-7012-3
[8] I. Yoda, K. Yamamoto, and H. Yamada, "Automatic acquisition of hierarchical mathematical morphology procedures by genetic algorithms," *Image and Vision Computing*, vol. 17, pp. 749–760, 08 1999.
[9] E. C. Pedrino, J. H. Saito, and V. O. Roda, "Architecture for binary mathematical morphology reconfigurable by genetic programming," in *2010 VI Southern Programmable Logic Conference (SPL)*, March 2010, pp. 93–98.
[10] M. Almeida and E. Pedrino, "Hybrid evolvable hardware for automatic generation of image filters," *Integrated Computer-Aided Engineering*, vol. 25, pp. 1–15, 01 2018.
[11] J. Serra, *Image Analysis and Mathematical Morphology*. Orlando, FL, USA: Academic Press, Inc., 1983.
[12] J. Facon, *Morfologia Matemática: Teoria e Exemplos*. Editora Universitária da Pontifícia Universidade Católica do Paraná, 1996.
[13] S. G. M. Hossain, C. A. Nelson, and P. Dasgupta, "Hardware design and testing of modred: A modular self-reconfigurable robot system," in *Advances in Reconfigurable Mechanisms and Robots I*, J. S. Dai, M. Zoppi, and X. Kong, Eds. London: Springer London, 2012, pp. 515–523.