

Automatic Generation of Multipath Algorithms in the Cellular Nonlinear Network

¹ Victor M. Preciado, ² Domingo Guinea and ² Rodrigo Montúfar.

² Instituto de Automatica Industrial- Spanish Council for Scientific Research
La Poveda (Arganda del Rey) - 28500 Madrid, Spain.

¹ Escuela de Ingenierías Industriales, Universidad de Extremadura.
Avda. de Elvas s/n - 28007 Badajoz, Spain.

ABSTRACT

The traditional image processing techniques require a lot of computational effort due to data on each pixel are computed in a sequential way and the path of information is an A/D converter. The delay accumulation create in this process is unacceptable in real time image processing because of the high information flow managed in the usual vision tasks (e.g. automatic industrial inspection, vision problems in robotics, pattern analysis, etc.).

Thus, the use of a massive parallel architecture working with analog signals avoids the previous problems. This is just the basis idea of Cellular Neural Network (CNN's): an array of analogic dynamic processors which cells interact directly within a finite local neighborhood . The local CNN connectivity allow its realization as VLSI chips that can operate at a very high speed and complexity. Nowadays CNN architectures implemented as VLSI chips shows the aptitude of extremely high speed compared with traditional digital image processing tools. The proliferation of more and more sophisticated CNN architectures, and the increasing effort to implant practical system based in CNN chips, make important the development of analog algorithm to perform complex image processing tasks dedicated to many different fields, i.e. industrial applications, robotic systems and pattern recognition.

The objective of this work is to generate a learning machine capable of find solutions for complex image processing task by CNN's. First a general machine for automatic analog algorithm design independent of the problem to solve is created, this is accomplished through an evolutionary strategy that is an extension of genetic programming. Second, this work introduces a suite of sub-mechanisms that increase the power of genetic programming and contribute to reduce the enormous space search for producing a plentiful search. Some concepts in this section are related with AI theory, in such a way that in this work we are in the intersection field of AI and Image Processing by CNN.

Keywords: Cellular Neural Network, Visual Processing, Parallel Computation, Real Time Processing, Automatic Templates Generation, Genetic Algorithms.

1. INTRODUCTION

The use of a massive parallel architecture working with analog signals avoids the computational effort that implie the tradicional image processing techniques due to data on each pixel are computed in a sequential way and the path of information is an A/D converter. This is just the basis idea of Cellular Neural Network (CNN's): an array of analogic dynamic processors which cells interact directly within a finite local neighborhood [1]. The local CNN connectivity allow its realization as VLSI chips that can operate at a very high speed and complexity[2]. Nowadays CNN architectures implemented as VLSI chips shows the aptitude of extremely high speed compared with traditional digital image processing tools. The proliferation of more and more sophisticated CNN architectures, and the increasing effort to implant practical system based in CNN chips, make important the development of analog algorithm to perform complex image processing tasks dedicated to many different fields, i.e. industrial applications[3], robotic systems and pattern recognition[4][5].

The objective of this work is to generate a learning machine capable of find solutions for complex image processing task by CNN's. First a general machine for automatic analog algorithm design independent of the problem to solve is created, this is accomplished through an evolutionary strategy that is an extension of genetic programming[6]. Second, this work

introduces a suite of sub-mechanisms that increase the power of genetic programming and contribute to reduce the enormous space search for producing a plentiful search. Some concepts in this section are related with AI theory, in such a way that in this work we are in the intersection field of AI and Image Processing by CNN.

The stages in the study of the automatic generation of CNN multi-template trees are the following:

1. General case, in which there isn't any constraint neither in the single templates nor in the tree shape.
2. Case of restrictions in a important set of templates with either the initial state image or the input image is pre-set by its design.
3. Case of introducing restrictions in the trees shape due to the use of a logic hierarchy in the templates order.

In previous works have been presented by several groups some CNN simulators with the feature of automatic single template generation [7][8][9], learning by examples using a Genetic Algorithm. In this work is presented a natural evolution of these works, in the way to automate the CNN image processing with a more complex structure than a single template.

2. MATHEMATICAL MODELLING OF THE CELLULAR NONLINEAR NETWORK.

Many computational problems can be reformulated as well-defined tasks where the signal values are placed on a regular 2-D grid, and the direct interactions between signal values are limited within a finite local neighborhood. This is just the basis idea of Cellular Neural Network (CNN's): An array of analogic dynamic processors which cells interact directly within a finite local neighborhood. The local CNN connectivity allow its realization as VLSI chips that can operate at a very high speed and complexity: 0.3TeraXPS performance for a 10x10mm² chip using a 2-μm technology in a robust implementation^{4,5}.

The dynamic of the array can be described by the following differential equations¹:

$$C \frac{dV_{xij}(t)}{dt} = -R_x \cdot V_{xij}(t) + \sum_{Cel(kl) \in N_r(ij)} A(ij; kl) V_{ykl}(t) + \sum_{Cel(kl) \in N_r(ij)} B(ij; kl) V_{ukl} + I_{ij} \quad (1)$$

$$V_{ykl} = \frac{1}{2} \cdot (|V_{xkl} + K| - |V_{xkl} - K|) \quad (2)$$

$$N_r(i, j) = \{C(k, l) | \max\{|k - i|, |l - j|\} \leq r\} \quad (3)$$

where i, j refers to a grid point associated with a cell on the 2-D grid, and k, l is a grid point in the neighborhood within a radius r of the cell i, j . Equation (1) describes a nonlinear dynamical system due to the equation (2) that includes in our system a piecewise linear function of saturation. In the equation (3) the concept of neighborhood is defined.

One processing element with a nonlinear template can be seen in fig. 1. The controlled current source connected by dotted lines represent the interactions within the neighborhood. Without these interactions, each processing element is just a low-pass filter with output saturation.

C is an input capacitor, R an input resistance and I an input bias current. $V_{ykl}(t)$ represents the neural activity, whereas $V_{yij}(t)$ is the output of the network and V_{uij} is a fixed external input to the network. B and A are connection matrices that respectively describe the input and feedback connectivity.

In many applications, the cloning templates A , B and the threshold current I are translation invariant. In this case, the dynamic of the array can be described by³:

$$C \frac{dV_{xij}(t)}{dt} = -R_x \cdot V_{xij}(t) + \sum_{Cel(kl) \in N_r(ij)} a_{kl} V_{ykl}(t) + \sum_{Cel(kl) \in N_r(ij)} b_{kl} V_{ukl} + I \quad (4)$$

when the template is space-invariant each cell is described by a simple identical cloning template defined by two $(2r+1) \times (2r+1)$ real matrices, as well as the constant term I . In this case every neuron has the same fixed synaptic weights, where N_r is the neighbourhood to which the synaptic connectivity of a neuron extends. So, the CNN will process local properties in the input image performing its convolution with a kernel defined by the cloning template. This feature makes the model very well adapted to image processing.

As the network will be devoted to this kind of tasks, it is convenient to represent equation (1) by the approximation of a difference equation of the form :

$$V_{xij}[n+1] = V_{xij}[n] + \frac{h}{C} \left(-\frac{1}{R} V_{xij}[n] + \sum_{C(k,l) \in N_r(i,j)} A(i,j;k,l) \cdot V_{ykl}[n] + \sum_{C(k,l) \in N_r(i,j)} B(i,j;k,l) \cdot V_{ukl} + I \right) \quad (5)$$

where h is the time step used to compute each iteration, A and B are the cloning templates.

3. MULTIPATH ALGORITHM NOTATION

In this work we are searching the automatic generation of an adequate structure, in the form of trees formed by templates for searching analog algorithms. So, the multi-template tree is presented like the fit way of representing the searched structure.

The notation used consists on symbols representing objects (which in our case are images) and on operators representing actions to be performed on objects (operators are templates in this work); so, templates are considered as operators relating images to produce as result a new image. An expression is a set of images and templates related coherently for performing complex image processing task (fig.1). A well-formed expression can itself be regarded as an object, and in turn can be related to others by an operator in the same way as an object can be related to others. In our case, the expressions are called trees.

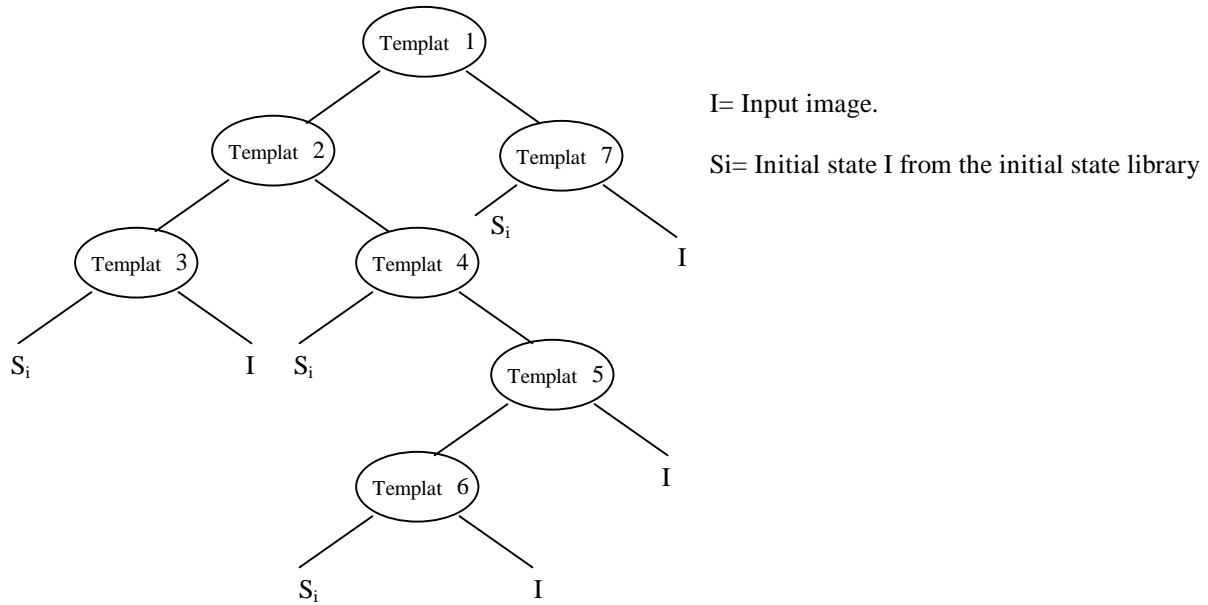


Fig. 1. Tree and node example.

The notation utilized is the *prefix* or Polish notation due to Lukasiewicz. According to this notation an operator O which performs an action on two objects x and y is represented $O x y$. In our case, O consist of a single template, x is an input

image and y is the initial state for the CNN differential equation to be solved. No parenthesis are needed and the principle operator for any term appear at the head of that term.

In the case of CNN analogic algorithms binary operator are the templates which need two external input images for its process to be performed (i.e. logic operators, CCD or objects extractor templates, etc.). Unary operators only need one image because either input image or initial state image are previously preset to a black, gray or white intensity level. In this way we can express the tree in fig.1 like (parenthesis are not needed, only used for making the expression clearer): $Tem.4(Tem.2(Tem.1(X, B), A), Tem.3(X, A))$.

The convenience of this notation can be seen in the random generation of trees representing analogic algorithms. The individuals in the initial random population and the offspring produced by each genetic operation must be all syntactically valid executable programs, we can assure this by checking that the next useful theorem, due to Rosenbloom is carried out in each new node of the tree.

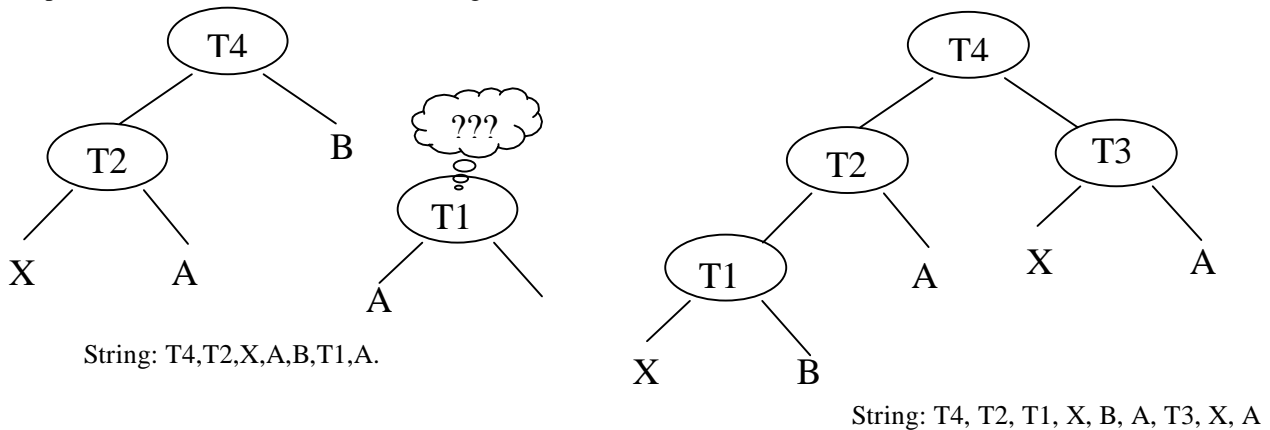
A sequence of symbols S in prefix notation is a well-formed expression if and only if:

- a. $rank(S) = -1$;
- b. $rank(\text{sub-expression on the left of } S) \geq 0$;

where rank is defined by:

- $rank(\text{binary operator}) = 1$;
- $rank(\text{unary operator}) = 0$;
- $rank(\text{constant}) = -1$;
- $rank(S1 \text{ concatenated with } S2) = rank(S1) + rank(S2)$.

This theorem enables us to recognize an coherent string from a non-coherent one. The concept of coherent string is important is process like the genetic growth where random process take place. In the following figure we can see both examples of coherent and non-coherent string.



The string $T4,T2,X,A,B,T1,A$ is an example of a non-coherent string and the string $T4, T2, T1, X, B, A, T3, X, A$ is a coherent one.

4. GENETIC PROGRAMING FOR SEARCHING A SOLUTION

Genetic algorithms is used to search the most adequate template sequence or multi-template tree for each case [10]. These algorithms are probabilistic search algorithms which simulates natural evolution and are very useful in combinatorial optimization.

In these algorithms the search space examined in any instant is called population. The population is formed by a collection of individuals that are represented by character strings. These individuals are represented by character strings and frequently named chromosomes, which are often referred to as chromosomes. The purpose of using a genetic algorithm is to find the

individual from the search space with the best “genetic material”. To do that the quality of an individual is measured with an evaluation function termed fitness function.

In definitive, the algorithm works first choosing the initial population of potential solution and defining the fitness function. Then, in every iteration, the individuals (parents) are selected to produce new individual (children) of the next generation (a new algorithm iteration) (children) by means of the combination of the their genetic material.(crossover operator). Then for every new individual a probability near to zero exists that it can “mutate”, i.e. suffer a small modification in their genetic material (mutation operator). Finally, some individuals are removed from the population according to a selection criterion in order to reduce the population to its initial size.

It interests to stand out the fact that the mutation is need to explore new states and helps the algorithm to avoid local minimum . Crossover should increase the average quality of the population. Chosen adequate crossover and mutation operators increase the probability that the genetic algorithm results in a near-optimal solution in a reasonable number of iterations.

3.1. Initial Generation and the Prefix Notation

Each individual codifies a multi-templates tree like a string in Polish notation in the method proposed. In the initial random population and the offspring produced by each genetic operation must be all syntactically valid executable programs, we can assure this by applying the Rosenbloom theorem. Although the individuals are strings in Polish notation, to simplify the following exposition a tree representation for individual is suppose.

The way we generate a tree is by fixing an upper and lower limit of operations that the tree can contain, and assign both binary operation (P_o) and image probability ($P_i=1-P_o$). These are the probabilities of a new node to be an operation or an image. In this way we add new nodes we the former probabilities and probe in each iteration if we exceed the upper limit or if we don't reach the lower one. The string utilized for encoding the tree consist of a list of templates and images expressed in the Polish notation previously mentioned. One feature of this strings is the variable length of these. So, The generation of the initial population is performed by linked lists in which each new terms is added based on the probability preset by the user of the program.

3.2. Evaluation of each Individual.

The function that produces the evaluation to minimize is an energy function proportional to the to the difference between pixels from the current output image ($I_{Current}$)and the desired one ($I_{Desired}$). This function is the following:

$$E(p) = \sum_{wholeimage} |Image_{Desired} - Image_{Current}(p)| \quad (1)$$

being t chromosome encoding the current tree, and $I_{Current}$ the output image of the highest template when the input image to process is connected in all the free branches. We shouldn't forget that what we want to do is to get a desired output image from a given input image by thus of the process performing by the multi-template tree.

3.3. Performing the Genetic Programming

In the Crossover operator the two parents participating in crossover are usually of different sizes and its branches are arbitrarily disposed. A crossover point is randomly chosen in the first parent and a crossover point is randomly chosen in the second parent. Then the subtree rooted at the crossover point of the first parent is deleted and replaced by the subtree from the second parent. Crossover is the predominant operation in genetic programming work and is performed with a high probability (say, 85% to 90%).

In the mutation operation defined by Koza[6], a individual is probabilistically selected from the population. A point is randomly chosen, the subtree rooted at that point is deleted, and a new subtree is grown there using the same random growth

process that was used to generate the initial population. This mutation operation is performed sparingly. The probability of the mutation is 1% during each generation of the run

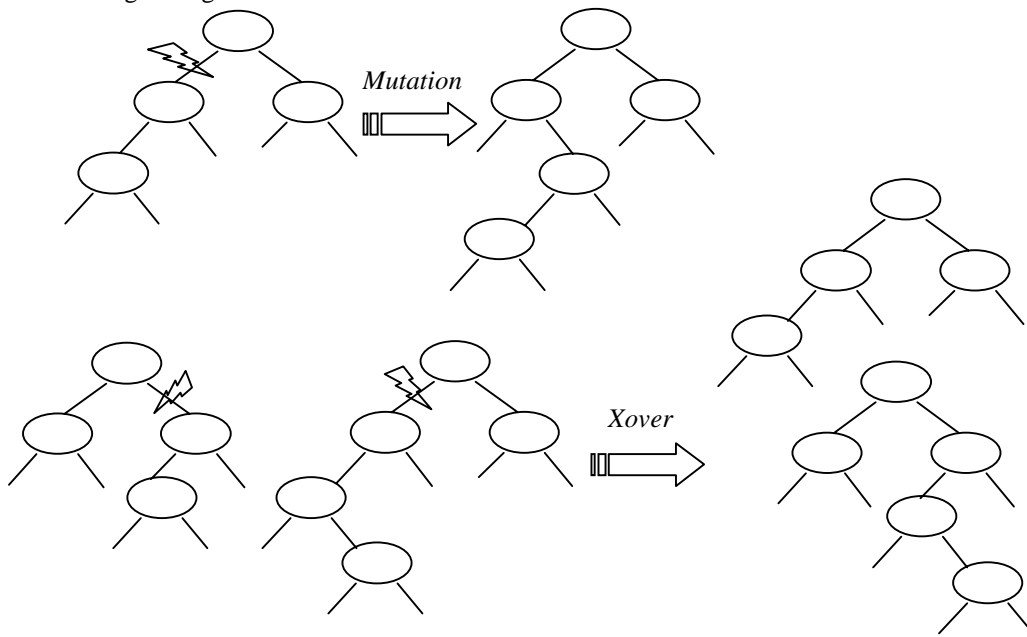


Fig.2. Mutation and Crossover example.

3.4. Heuristic Reduction of the Space of Solutions

Supposing that we are in the case of binary templates (unary template are a particular case of binary template with one of its input fixed ahead of time), we can affirm that in a tree formed by S templates there is $S+1$ free branches that can be filled by other terms (in general other trees). It can be demonstrated if we advise that the initial seed of a tree is an binary template (one template and two branches, fig.3.a), and the growth process is by adding new templates located in any of this branches (fig. 3.b and fig. 3.b show the possibilities in the growth process). Each new template cancel one of this free branches and produce two new ones. In conclusion, for each new template appear a new branch, remaining the initial difference between templates and branches. Going beyond, it can be seen that any tree of S templates can produce $S+1$ trees by adding one new operator, so the number of possible tree with S operators is a factorial function of S ($S!$).

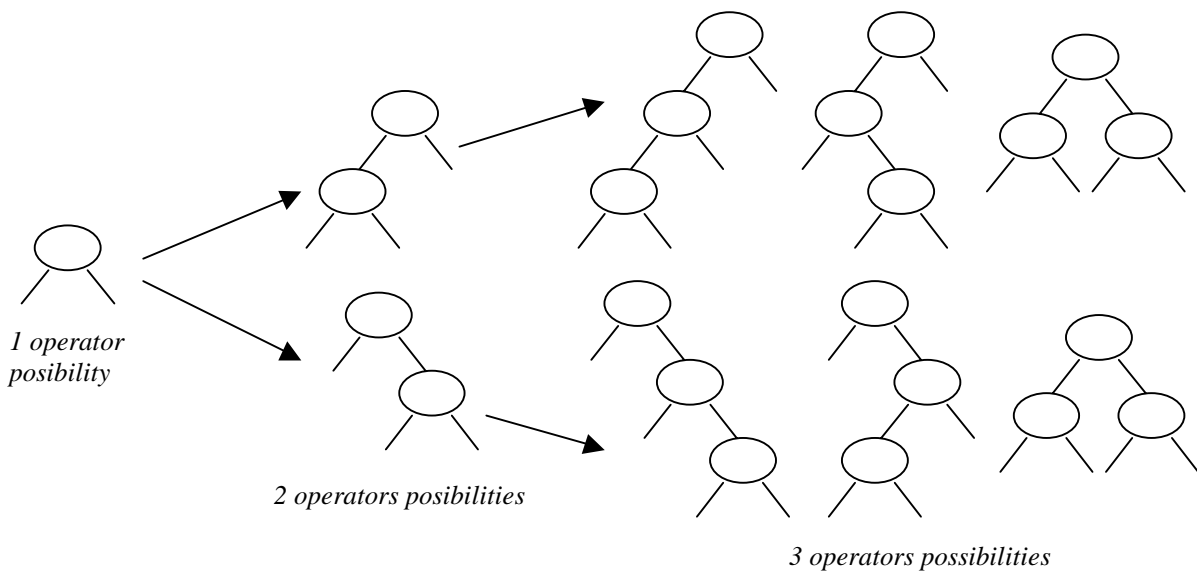


Fig.3 Tree growth process.

If we consider that the number of possible templates are N , extracted from a template library, we can conclude that the search space size are equal to the number of possible variations of S elements from a possible set of N . So, from a fixed tree formed by S nodes we have a search space of N^S elements. In conclusion, the search space size produced by the set of possible trees of S operator, considering a template library of N elements are (2), and if the set of possible trees are confined between a lower and an upper limits S_L and S_U the number of element in the search space are expressed in (3).

$$SIZE_{N_templates}^{S_Nodes} = N^S \cdot S! \quad (2) \quad \sum_{S=S_L}^{S_U} SIZE_{N_templates}^{S_Nodes} = \sum_{S=S_L}^{S_U} N^S \cdot S! \quad (3)$$

Like it was said before, the way to generate a new tree is by adding elements in a queue which represent the tree in Polish notation, taking into account the Rosenbloom's theorem for assuring the syntactic validity of the tree. With these premises, a tree formed by S templates is represented by a queue including S templates and $S+1$ input images. The probability of a tree of S binary operations is equal to the probability of obtaining the before queue, and it is $Po^S (1-Po)^{S+1}$.

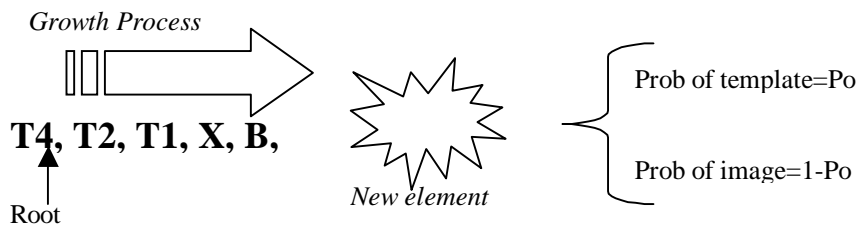


Fig.4 Probabilities implicated in the growth process.

The influence of Po (probability of a new element being an operation) in the size probability of the tree is represented in fig.4, being S the number of operations in the tree, $P(po,S)$ the probability of generate a tree of S operations being conditioned to a Po equal to po . Seeing the graphic it can be affirmed that in proportion to Po grows, the $P(po,S)$ decreases in the lower values of S and increases in the higher ones; it imply that when po increases, the probability of big trees is higher. This disposition is extreme when po is equal to 1, $P(1,S)$ is one for $S=\infty$ and zero otherwise.

In equation (6) and (7) we have seen the search space size produced by the set of possible trees of S operator, considering a template library of N elements. A way to reduce this size is by reducing the number N of elements in the template library. For this purpose 88 templates have been grouped in 19 sets, the elements of a same set have a similar behaviour. For not lose generality it will be used an annealing method after a previous search accomplished in the reduced space. It consist of carrying out a second search by an evolutive program starting in the former result. Concretely, the tree structure stays and the templates in each node are mutated, pertaining the new template to the same set.

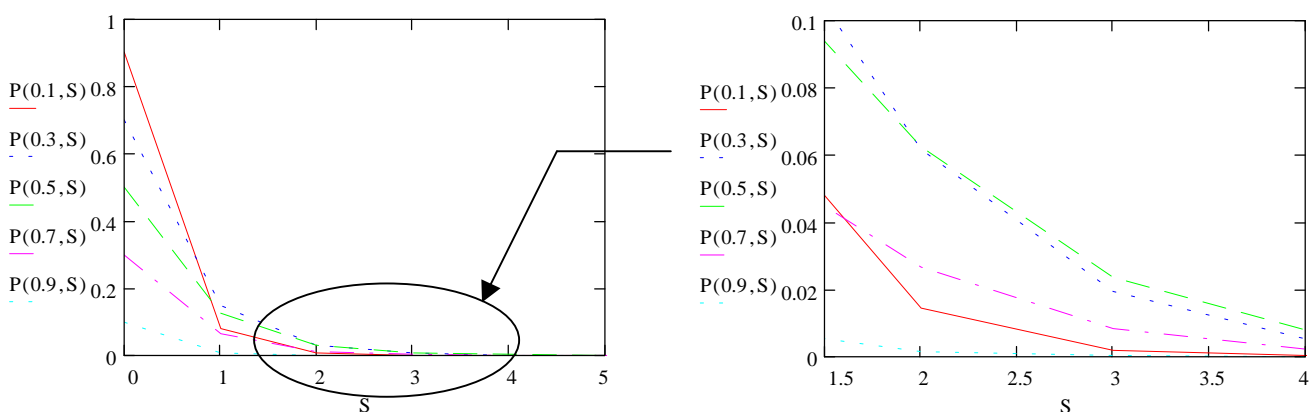


Fig. 4. Influence of Po in the Size of the Tree.

A second procedure to reduce the search space size is including an operation hierarchy in the random generation and mutation of the trees. It consist in classify the templates in three categories: a) Grey to Grey, b) Grey to Binary and c)

Binary to Binary templates in such a way that the tree starts with a Grey to Grey seed, and when a Grey to Binary or Binary to Binary template appears, the subtree that grows from this node are exclusively fashioned by Binary to Binary templates. So, the grey scale operations are isolated from binary ones being the grey to binary templates the interface between them.

- | | |
|------------------------------|-----------------------|
| 1. Homogeneous | 10. Lines eliminate |
| 2. Diffusion | 11. Points eliminate |
| 3. Average with Binarization | 12. Inverse CCD |
| 4. Smooth with Binarization | 13. CCD |
| 5. Border Detection | 14. Controlled CCD |
| 6. Concave/Convex. Border | 15. Morphing |
| 7. Diagonal Detection | 16. Halftoning |
| 8. Final Detection | 17. Binarization |
| 9. Propagation | 18. Colour Extraction |
| | 19. Logic Operations |

Fig.5 Families of templates implicated in the space reduction process.

5. EXPERIMENTAL RESULTS

In this section are going to be presented one application of the proposed algorithm for the automatic multipath algorithm design by means of genetic programming. The task to solve consists in the measure of the roughness of a figure by means of its concavities. To solve the problem one pair of input and desired images are given, and after some iterations a successful analog algorithm is provided.

In the following figure can be seen the screen of the software development tool programmed for solving the algorithm proposed in this work.



Fig. Software Development Tool Screen.

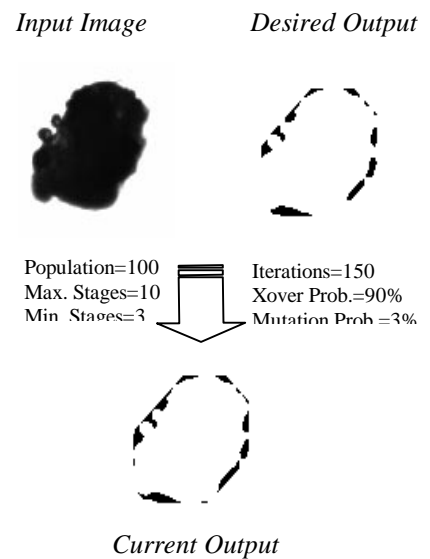


Fig. Inputs and output of the process

The idea in the solution of this task is the localization of the concave parts of the object. In the first step, a binarization process is performed in the input image. Next, the pixels located in the concave parts of the objects are carried into black, so with the XOR logic operator of these two images the concavities are located by the black pixels.

The templates implicated in this algorithm are the next:

THRESHOLD:

$$A := \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad B := \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I := 0$$

CONCAVE LOCATION FILLER:

$$A := \begin{pmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 2 & 0.5 \\ 0.5 & 0.5 & 0.5 \end{pmatrix} \quad B := \begin{pmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad I := 3.5$$

The XOR operation can be performance into the CNN-UM chip. This chip marry the analog processing capabilities of the CNN with local memories and logic to performance complex analog algorithms in a single chip.

In the next figure we can compare the just know algorithm to work the proposed task to the obtained by the software development tool programmed in this work.

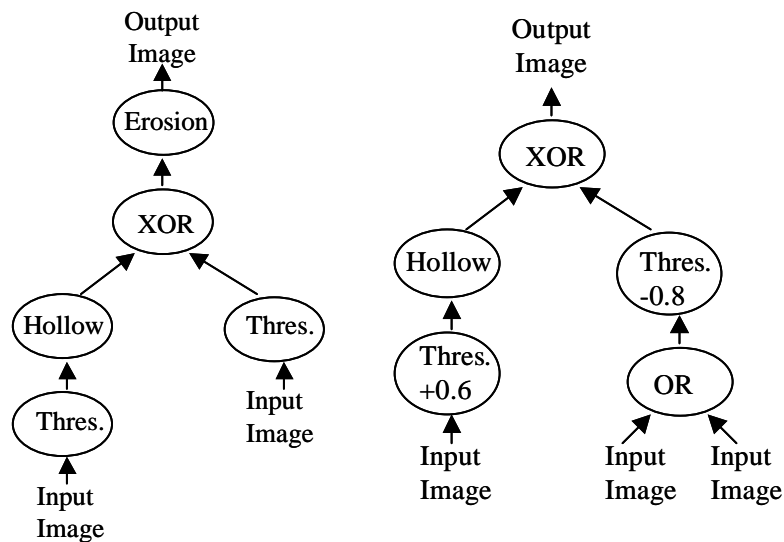


Fig. Algorithm for solving the proposed problem: In the left side we can observe the just known algorithm and in the right the one obtained by the proposed methodology.

There are differences between this algorithms: In the algorithm know there is a Erosion step that has faded in the obtained by the computer (this step only introduce a little improvement in the final appearance of the image), in the left branch under the XOR step the structure of the algorithm is the same but the value of the threshold and in the right branch change the value of the threshold and is introduced an absurd step that result in the input image.

6. CONCLUSION AND FURTHER RESEARCH

The template design task has been usually focused like an extrapolation of the traditional methods for digital image processing, or more recently, has been used complex mathematical techniques like mathematical morphology and PDE related methods. Every case, the success of the algorithm design rely on the designer experience. Automatic algorithm generation by GA's suppose a new methodology for this task, in this way, effort can be inverted in develop analogic algorithm for solving general vision tasks. The aim of this work is to introduce this methodology for the design of CNN's

Algorithms that allows the flexible implantation of industrial applications like industrial applications[3], robotic systems and pattern recognition[4][5].

Simple computer programs consist of one main program (called a result-producing branch). However, more complicated programs contain subroutines (also called automatically defined functions, ADFs, or function-defining branches), iterations (automatically defined iterations or ADIs), loops (automatically defined loops or ADLs), recursions (automatically defined recursions or ADRs), and memory of various dimensionality and size (automatically defined stores or ADSs). If an unexpert human user is trying to solve an engineering problem, he or she only have prespecify a reasonable fixed architectural arrangement for all programs in the population (i.e., the number and types of branches and number of arguments that each branch possesses). In further research genetic programming can then be used to provoke the evolution of the exact sequence of primitive work-performing steps in each branch.

The conclusion and further research can be summarized by the following:

1. Generic Tool for Algorithm Development by means of Genetic Programming.
2. Methods for reducing the enormous search space.
3. Industrial Applications.
4. *More Complicated Programs*: Subroutines, Iterations, Loops and Recursions.

ACKNOWLEDGMENTS

This work has been funded by the spanish CICYT proyect SIVA (Integrated System for Active Vision). We thank also our partners from CNM of Seville and the Institute of Optics.

REFERENCES

1. L. O. Chua, L. Yang. "Cellular Neural Networks: Theory". IEEE Trans. on Circuits and Systems. Vol. 35, No. 10. pp. 1257-1272. October 1988.
2. L. O. Chua, L. Yang. "Cellular Neural Networks: Applications". IEEE Trans. on Circuits and Systems. Vol. 35, No. 10. pp. 1273-1290. October 1988.
3. L. O. Chua, T. Roska. "The CNN Paradigm". IEEE Trans. on Circuits and Systems. I: Fundamental Theory and Applications. Vol. 40, No. 3. pp. 147-156. March 1993.
4. S. Espejo, R. Carmona, R. Domínguez-Castro, A. Rodríguez-Vazquez, "A VLSI oriented continuous-time CNN model", Int. Journ. Of Circuit Theory and Applications, vol. 24, pp. 341-356 (1996).
5. "Cellular neural network design for solving specific image-processing problems", Cellular Neural Networks, pp. 191-199, De. Wiley (1993)
6. J. M. Cruz and L. O. Chua, "Design of high speed high density CNN's in CMOS technology", Int. J. Circuit Theory and Applications, pp. 555-572, vol. 20, 1992.
7. P.Szolgay, I.Kispal and T.Kozek, "An Experimental System for Optical Detection of Layout Errors of Printed Circuits Boards Using Learned CNN Templates", Proceedings of the International Workshop on Cellular Neural Networks and their Applications (CNNA-92), pp. 203-209, Munich, 1992.
8. T.Sziranyi and M.Csapodi, "Texture classification and Segmentation by Cellular Neural Network using Genetic Learning", Computer Vision and Image Understanding, Vol. 71, No. 3, pp. 255-270, September 1998.

9. P.L.Venetianer, F.Werblin, T.Roska and L.O.Chua, "Analogic CNN Algorithms for some Image Compression and Restoration Tasks", *IEEE Transactions on Circuits and Systems*, Vol. 42, No. 5, 1995.
10. J. R. Koza: "Detailed Description of Genetic Programming". *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
11. R. Caponetto, M. Lavorgna, A. Martinez and L. Occhipinti, "Cellular Neural Network Simulator for Image Processing Applications", 1998 Fifth IEEE International Workshop on Cellular Neural Network and Their Applications Proceedings, pp.360-365. April 1998.
12. Martin Hanggi and George S. Mostchytz, "Stochastic and Hybrid Approaches Toward Robust Templates", 1998 Fifth IEEE International Workshop on Cellular Neural Network and Their Applications Proceedings, pp.366-371. April 1998.
13. V. Preciado, D. Guinea, J. Vicente, M.C.Garcia-Alegre and A.Ribeiro, "CNN's design by GA's", European Congress on Computational Methods in Applied Sciences and Engineering Proceedings, to be published in 2000.
14. J. M. Cruz and L. O. Chua, "A 16×16 Cellular Neural Network Chip: the First Complete Single-Chip Dynamic Computer Array with Distributed Memory and Grayscale I/O", *Analog Integrated Circuits and Signal Processing*, 15 (3), 1998, 227-238.
15. A. Paasio, A. Dawidziuk, K. Halonen and V. Porra, "Minimum size 0.5 μm CMOS Programmable 48×48 CNN Test Chip", *Proc. of the 1997 European Conference on Circuit Theory and Design*, 1997, 154-156.
16. S. Espejo, R. Domínguez-Castro, G. Liñan and A. Rodríguez-Vázquez, " 64×64 CNN Universal Chip with Analog and Digital I/O", *IEEE 1998 Int. Conf. On Electronic Circuits and Systems*, 1998