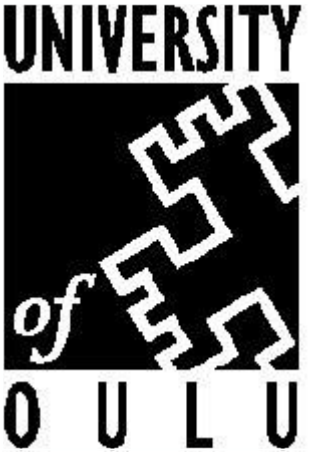


CONTROL ENGINEERING LABORATORY
Infotech Oulu and Department of Process Engineering

Linguistic equations and their hardware realisation in image analysis

Pasqualino Axel Donnini

Report A No 8, June 1999

University of Oulu
Control Engineering Laboratory
Report A No 8, June 1999

Linguistic equations and their hardware realisation in image analysis

Pasqualino Axel Donnini

Abstract: In this paper, a new application for the Linguistic Equation method is presented. Developed mainly for industrial process control system, this method has been successfully applied to the processing of digital information. In particular in the field of image analysis a noise suppression algorithm has been developed. In order to evaluate its performances this filtering algorithm has been tested with some images. The results are presented both for allow the comparison with other filtering methods, and highlight the main advantages and future developments of this kind of processing approach. Then, a general hardware system for Linguistic Equation applications is presented, with particular regard to the noise reduction application. This one seems to be a very useful application of the Linguistic Hardware System, since in the resulting hardware plan a lot of simplifications are allowed.

Keywords: image processing, Linguistic Equations, noise reduction, hardware architecture

ISBN 951-42-5314-0
ISSN 1238-9390
ISBN 951-42-7503-9 (PDF)

University of Oulu
Control Engineering Laboratory
Linnanmaa
FIN-90570 Oulu

1. Introduction

There are a lot of variables and factors that have an influence on the control of an industrial process. Since these variables act in many different ways on the output variable, one of the biggest difficulties is just the way we can manage all these things in order to control the process without using too complex and expensive algorithms.

With the Linguistic Equation approach, we try to find the way to work with linear control rules, namely linguistic equations, since they do not give computational problems. Particular care is for the choice of the correct coefficient of each addend of the given equations. References [1, 2] give more details of the approach.

Normally the linguistic equations are extracted from the experimental data, both with the expert knowledge and the simulation results, in order to reproduce the behaviour of the output process in a working point. The final aim is to find linear equations and in particular the right coefficients for the equations, in order to put all the significant variables at the same level during the computation of the output. The choice of the right amplitude and sign of those coefficients makes it possible. In fact, in this case a variable that has a bigger effect on the output has also a bigger coefficient.

In order to use the Linguistic Equation approach with the image processing application, and in particular for the noise cancellation with image filtering, we have to use a little bit different approach than in the normal method used for the process control. In this case, we have to reproduce the original image at the output, that is a non-noisy image. Our current input, however, is a noisy image that could be very different with respect to the original training input image. The equations that we have to use should be structured in a particular way with which we can recognise the edges and details of the image and at the same time clean the noisy pixels. The cleaning operation is made by correcting the noisy pixel value with regard to the values of the neighbouring pixels.

The data are also used in a different way than in the normal Linguistic Equation method. If we only use data to find the correlation and the direction of the changes among all the variables, we could obtain a wrong result. This is due to the fact that in the tuning step we are used to change the system in a way that when the input is similar to the tuning data, the output is similar to the correct one. In this case we normally have noisy data as input, and of course we should reproduce the original data, but at the same time take off the noisy pixels by recognising and extracting all the edges and details from the noisy image.

This work has been developed and designed in the Control Engineering Laboratory, University of Oulu, during 15.01.99-30.06.99, in co-operation with the Department of Electrical Engineering, University of Rome "Tor Vergata". The co-operation includes exchange of researchers as well as in the organisation of common conferences and workshops. I have been studying automatic industrial processes and control systems as well as digital electronics at my own University and this has been a good basic knowledge to develop this kind of application.

The project has been financed by Technology Development Centre in Finland.

2. Image analysis problem

One of the most important problems in the digital processing, and especially in the image processing, is how we can remove the noise from a corrupted image in order to preserve and obtain the correct information, without degrading the image structure and the quality of the details. Several non-linear filtering techniques could be used [6], and in these years some good results have been obtained using fuzzy theory and some fuzzy inference based algorithms (see for example [3], [4], [5]). Fuzzy logic has in fact demonstrated to be very well suited to address the uncertainty, which characterises the process of extracting information from image data. This is the starting point of developing of our theory and filtering algorithm based on the Linguistic Equations approach.

The grey level images (from 0 to 255 different grey levels) are analysed within 5x5 window (Fig. 1). We can recognise here 24 input variables, and 1 output variable, that is the central pixel of the window. Since this is not a time depending process, the input data is obtained using an imaginary time axis, by moving the window on the image, and taking every time the values of the pixels like the new values for the variables. In the tuning step we use the original non-noisy image, but afterwards, during the normal computational use the input image will be a noisy image.

We have adopted two different types of test images: the well known 256x256 "Lena" pictures, corrupted with different kind of noise to obtain the input samples; a 256x256 SAR image (SAR, Synthetic Aperture Radar), in order to check the performances of the system for the developing of real time applications.

All the tuning steps are performed with the adaptation of the output membership definition at the input data. This is obtained by fitting the function to all the points **(x, y),** where **y** is the output value for the central pixel in the non-noisy image and **x** is the corresponding linguistic value, that is the result of the computation of the Linguistic Equation. This is the way, how we can use together the effect of the Linguistic Equations and the input data in order to optimise the output membership definitions.

3. Image analysis using Linguistic Equations approach

3.1 Form of Linguistic Equations

To show which kind of Linguistic Equation we have to use in our application, we can examine the case of the identification of a horizontal edge in the input window of Fig.1, which has to be preserved in the output image.

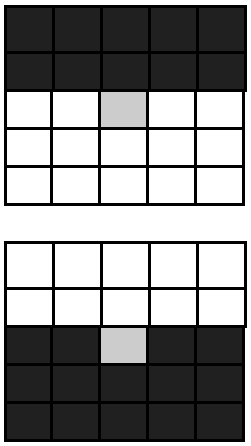
The corresponding Linguistic Equation should be a linear combination of the input and output variables, with all the coefficients –1 or +1. The output coefficient default value will be +1, while for the input coefficients we choose the value in relation to their position in the window, i.e. for the horizontal edge, if this is over or below to the edge line that we have to identify. Let's consider the window in the Fig. 2. It is a window configuration for recognising horizontal edges. The input values are now converted to the corresponding linguistic values, and we are going to compute the value for this linguistic equation for every input window configurations. All the variables should have values in the interval [-2,2] (negative values are for dark grey levels, and positive for the bright ones), and the coefficients of the equation related with the two upper rows are –1 and the others +1. Therefore the resulting equation will have big positive (negative) values only when the most of the variables in the upper part of the edge will be dark (bright), and those in the lower part will be bright (dark). Every other input configuration will give lower absolute values.

Another important coefficient configuration is when they are all fixed to 1. We need to use this rule for identify all the uniform areas in the input noisy image, where there are no edges or details. In those cases the noise can be removed just by giving to the central pixel the mean value of all neighbouring variables.

For a better understanding of this approach we show how the variables are distributed in the window (the central output pixel is marked in grey), and the practical case of two rules, with the correspondent linguistic equation. The first one is able to identify horizontal edges, while the second one is used for noise cancellation (Fig. 2 and Fig. 3).

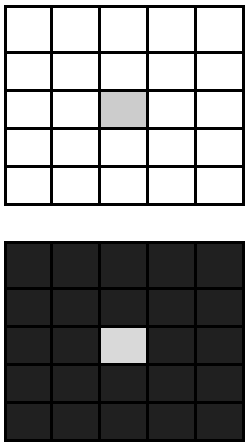
X1	X2	X3	X4	X5
X6	X7	X8	X9	X10
X11	X12	X25	X13	X14
X15	X16	X17	X18	X19
X20	X21	X22	X23	X24

Figure 1. 5x5 window used in image analysis.



$$(-1)\cdot X1+(-1)\cdot X2-\ldots+(-1)\cdot X10+(1)\cdot X11+\ldots+(1)\cdot X24=(1)\cdot X25 \quad Xi\in[-2..2] \quad i=1..25$$

Figure 2. Linguistic equation for identification of horizontal edges.



$$(1)\cdot X1+(1)\cdot X2-\ldots+(1)\cdot X10+(1)\cdot X11+\ldots+(1)\cdot X24=(1)\cdot X25 \quad Xi\in[-2..2] \quad i=1..25$$

Figure 3. Linguistic equation for uniform area noise cleaning.

Usually in the Linguistic Equations methods, the equations are extracted from the experimental data, in according with the expert knowledge. This means that there should be a preliminary step for the implementation of this operation. In this particular application we choose from the beginning the shape of the linguistic equations (the coefficient's vectors), that should be the same for all the input images, and this leads to a big reduction in the complexity of the structure of our system.

3.2 Obtaining Linguistic Equations

If we compare the results calculated for the whole set of Linguistic Equations used for the identification of the edges (Fig. 2), we see that equations with a big output value (positive or negative) are only those, which have a coefficient vector that better matches its disposition with the current input window. Thus the value of the central pixel should depend only on those rules (and the linguistic equations). In the next part we'll call those window configurations "winners".

There can be many ways to compute the linguistic value of the central pixel. Every "winner" equation can provide a probable linguistic value for the output. This means the mean (or median) value only of those pixels belonging to the bright (or dark) side of the window (in the case an edge is identified, this divide the window in two sides) at which the central pixel should belong. The final value for the output is a weighed combination of all these probable values.

The main problem is now how to find the best way to characterise the weight coefficients. Note that those coefficients could also be used for choose from the set of the filtering linguistic equations which ones are the "winners".

Actually the coefficients are chosen in this way:

- if **T(k)** is the value which results from the evaluation of the k-th equation, and **M** is the input's noisy value for the central pixel, the coefficient **c(k)** will be **c(k)=T(k)/M**. In this way we want to put in evidence how much an equation, and in particular the result of his evaluation, tries to change the value of the noisy central pixel. A rule is one of the "winners" if **c(k)>1**.

We need also to consider the case in which there are no "winner rules." When this happens it means that no one of the rules match the shape with the actual input window. Therefore we also have to use the noisy value of the central pixel for the actual input window in the computation of the output. We are in a situation in which no particular edge configuration or noisy pixels have been found, and we don't want to change too much the value of the original input value (even if it is noisy).

Actually in this cases we assign to the output pixel the mean value among its original noisy input value and the output evaluation value of the Linguistic Equation that matches best his shape with the input window.

For example, suppose that we have only two rules to filter the image (Fig. 2 and Fig. 3). If the actual noisy window configuration is the one showed in Fig.4a, since there is an evident horizontal edge, the resulting values for the Linguistic equations are: 1.64 and 0.21, and it means that the first equation match better than the second the input pixel configuration. The weight coefficients are:

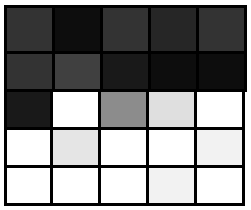
$c(1)=1.64/0.5=3.28 > 1 \rightarrow$ "winner"

$c(2)=0.21/0.5=0.42 < 1$

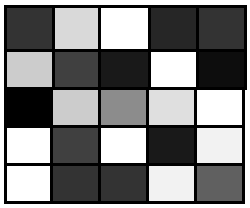
This means that only the first result (1.64) is used to compute the output value for the central pixel, and his probable value refers to a bright pixel, like we expected.

In the second example, shown in Fig. 4b, there is not any "winner" equation since both of the weight coefficients c(1) and c(2) are less than one:

$c(1)=0.304/0.5=0.61 < 1$
$c(2)=-0.129/0.5=-0.26 < 1$



$[X_1,X_2..X_{25}]$=[-1.6, -1.9,-1.6,-1.7,-1.6,-1.6,-1.5,-1.8,-1.9,-1.9,-1.8,
+2,+1.3,+1.9,+2,+1.5,+1.9,+2,+1.8,+2,+1.9,+1.9,+1.8,+2,+0.5]
Figure 4a. Noisy input configuration and linguistic variables vector.



$[X_1,X_2..X_{25}]$=[-1.6,+1.8,+1.9,-1.7,-1.6,-0.8,-1.5,-1.8,+2,-1.9,-2,
+0.8,+0.8,+1.7,+2 , -1.5, +1.9,-2,+1.8,+2,-1.9,-1.9,+1.8,-1.4,+0.5]
Figure 4b. Noisy input configuration and linguistic variables vector.

Then the output central pixel value is (0.304 + 0.5)/2=0.42, and it means that the value of the central pixel is more or less the same of the noisy image, since no particular window configuration is been detected.

4. Comparison between fuzzy filtering and Linguistic Equation Approach

The first advantage that we can find in our method with respect to the traditional fuzzy filtering is the very small number of rules.

For example, for a 5x5 window:

- in the fuzzy filtering, in order to recognise all the edges, we have to evaluate 25 rules for the white (or bright) output pixel case, and 25 rules for the black (or dark) case [4, 10].
- in the linguistic filtering the number of rules is reduced to 15 (Fig. 5), both for the case of white and black output pixel. The only difference between these cases is the sign that results from the evaluation of each equation

This means that the computational step, in which we evaluate all the equations, will be very simple.

Another important and very powerful topic is that this number of rules could be theoretically once more reduced. This is possible because we identify edges by checking which configuration of rules fit better in the actual input window. This is done just with 8 principal rules (2 for horizontal edges, 2 for vertical edges, 4 for the oblique ones), i.e. the windows in the first two columns of Fig. 5, and one for the mean cleaning. Obviously, bigger is the number of rules, bigger is the probability that we can better identify every configuration of edges, but since most often the input images are noisy, the edges are not sharp at all and we cannot be completely sure about any window's pixel configuration.

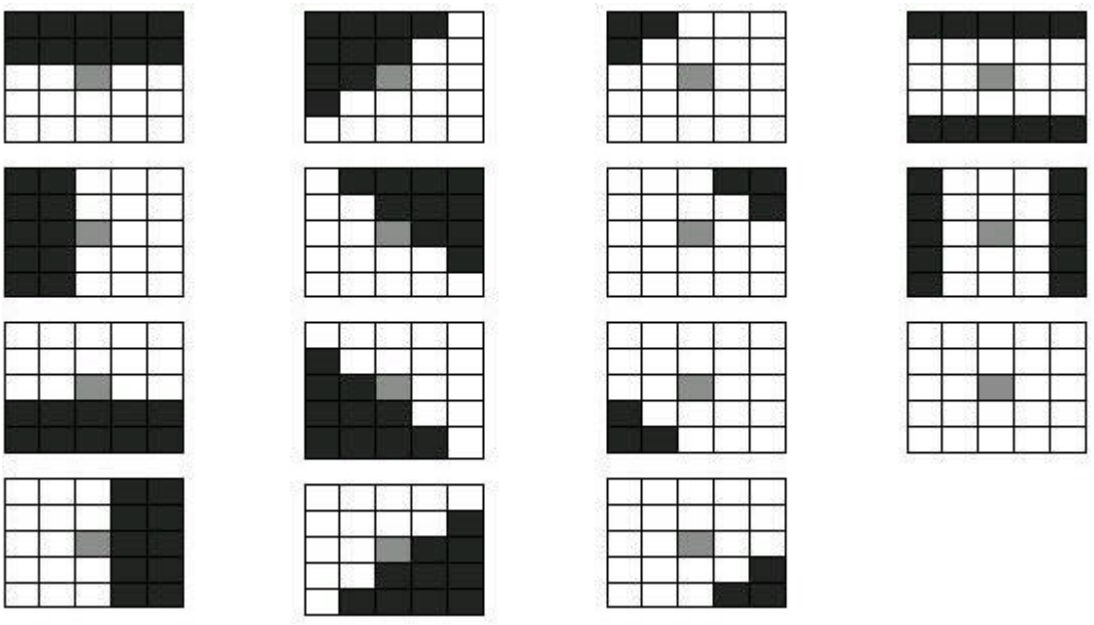


Figure 5. Filtering window configurations.

Another advantage is the reduced computational complexity, due to the simple way, with which we can evaluate the Linguistic Equations. Now the final output conversion block (delinguistification or defuzzification) presents a very low complexity, respect to the fuzzy systems, where in the most of the cases, this is just the more complex and time consuming part of the whole system.

Example:
Let's analyse a 50x50 Lena picture, added with a uniform noise (maximum amplitude 32).
We filter this image with all the 15 rules:

- 1 for the identification of the noise in the uniform areas,
- 2 for horizontal edges,
- 2 for the vertical edges,
- 4 + 4 rules for the different oblique edges,
- 2 for the identification of vertical and horizontal band inside the window

If we compare our result with the original image we obtain a reduction of the MSE from 310.54 to 120.13 (gain 61.3%).

Filtering now the same noisy image just with the first 9 rules the reduction of MSE is 121.1 (gain 61%), that is a little bit less than the first result, but we have also reduced by 1/3 the time of evaluation of the linguistic equations.

We must underline that this behaviour is strongly dependent on the input image, and we ca not exclude some improvement in the results by adding new rules. For example with a 100x100 SAR image, the 9 rules gain is 15.9%, while the 15 rules gain is 16.7%. See Fig. 10 to have a visual evaluation of these results.

A lot of tests with different kind of additive noises are been done to check the behaviour of this linguistic approach.

- uniform noise (max amplitude 32 and 48)
- spot noise
- gaussian noise (max amplitude 96 and 64 with variance 48 and 32)

We have compared these results with those from the fuzzy filtering with the MSE evaluation.

$$MSE = [\ \Sigma\ (y(n)-s(n))^2\ ] / N^2$$

The behaviour of our system is quite good if we compare the value of the MSE with the noisy one, but since we are dealing with images, we also have to use a visual evaluation. In this case, even if the edges are well identified, there is a kind of difficulty to get into focus.

The most important result is that the behaviour of our system is good, independently from the type of the noise, and this is a very interesting feature. Many times we do not know which kind of noise is added on our image, especially when we have just the noisy one.

The results about the processing of the Lena image are shown in the Table 1.

Before showing the results of SAR image filtering we need to explain something about this particular case. Here we have no possibility to tune the system with a non-noisy image, since we have only the noisy one (Fig. 9b). For this reason at the beginning we were tuning the system by the wavelet filtered Sar image (Fig. 9a), that seems to be the best results obtained with complex filtering operations. The only way to evaluate how good is the result of the LE filtering (Fig. 9c), is to compare the details that has been pointed out from the noisy image, between the two filtered images. MSE is not a good indicator (we compare the LE filtered image with the wavelet image, that is just another filtered image, not the original one!), anyway the MSE value in this case is 17%.

If we use a visual comparison, we can see that in the wavelet images in some areas, the noise cleaning seems to be very efficient. Looking to the noisy image it seems that there are some missing details, otherwise, the LE filtered image provides a good performance in preserving details, that is the main important thing.

It is also possible to tune the system with the noisy image. The results of this kind of filtering are showed in Fig. 11 (the MSE between this image and the wavelet one is about 10%).

Table 1. Linguistic Equation filtering results with different kind of additive noises.

NOISE TYPE	GAIN (%)	MSE noisy image	MSE LE filtered image
Uniform (max amplitude 32)	61.7	333.7	127.8
Uniform (max amplitude 48)	73.8	768.0	201.2
Salt and Pepper (frequency 0.1)	89.9	1882.9	190.2
Gaussian (max amplitude 64, variance 32)	72.0	762.8	213.8
Gaussian (max amplitude 96, variance 48)	79.7	1624.7	329.66

5. Hardware realisation of the Linguistic Equation approach

5.1 Basic system functions

To perform the control algorithm the hardware system should consequently proceed in three different phases:

STEP 1: Analyse the experimental input data (stored in a memory), and extract the values of min, max, mean, meanH, meanL to obtain the initial membership definition for the variables.

STEP 2: Extract by linear regression methods the structure of the linguistic equations, that is try to find the sign and the amplitude of the variation of each variable with respect to the others. Then we have to tune the output membership definition.

Since these two steps are executed only once at the beginning, the problem is to check which kind of advantages we got if these parts are developed by hardware. This means, can we implement only the hardware linear regression circuit, and then choose the final linguistic equation structure by software, or can we design a full hardware system.

If we look at this problem from an image analysis point of view, the solution is very simple, because in this case the linguistic equations are fixed, and we have only to think how to design the optimisation step for the output membership definition. Again, in this case, the hardware block for the evaluation of the linguistic equations is a very simple circuit. We have to evaluate linear equations where all the coefficients for the linguistic variables are +1 or –1.

STEP 3: the system is now ready for the evaluation of the inputs. In this step we have to transform the input vector (that is the input window) in linguistic values, evaluate the linguistic equations (as for the STEP 2), and return as the result the output, converted from a linguistic value to a real value.

Finally it must be done a consideration on how we extract the experimental data for all the input variables from the original image. Since the data are obtained shifting the 5x5 window on the image, most of the image's pixels belong to every set of train data for the variables. Consequently these train data values are practically the same for all of them.

A remarkable advantage that can be derived from this observation is the radical reduction of the process of construction of the membership definition. In fact it will be sufficient just to calculate such single function for one variable, and the result could be generalised for all. This is an important advantage for the hardware design.

From our tests it turns out that the system seems not to feel the effect of these particular changes.

5.2 Hardware configuration

Once we have defined the basic functions of this hardware approach for the implementation of a Linguistic Equation system, we should now explore every part in more detail. Since we are oriented to an image processing application, every single block will be also studied from this point of view.

The first step we have to implement is how to extract membership definitions from the input data. Usually this operation is done independently for all the variables, but in this particular case we can just analyse one variable and generalise the results.

We have to compute 5 particular values:

- **min** : minimum input value
- **max** : maximum input value
- **mean** : mean input value
- **meanL** : mean value of all the input values between **min** and **mean**
- **meanH** : mean value of all the input values between **mean** and **max**

In this case we have to analyse the input values twice, first time to obtain **min**, **max** and **mean** values, and another time to obtain **meanL** and **meanH.**

These particular points are now used to define the coefficients of the two membership definition's parabola, one for positive and the other for negative linguistic values, that we are going to use both in the tuning step and the computational step.

At this point we have two different choices:

- we can use a mixed hardware-software system, in which the role of the software part is to tune the system with the input data, obtaining the linguistic equations and optimising the shape (i.e. the coefficients) of the output membership functions. After that the hardware part will take from the software system the input and output membership definition's coefficients as well as the linguistic equations coefficients, and perform the computational step
- Both tuning and computational steps will be done in a hardware environment. In this case we have to arrange hardware regression circuits to extract the relations between input data in order to extract the coefficients of the linguistic equations, and to optimise the output membership definitions

The tuning data can be stored in an external memory, since both the hardware and software parts are going to use these values.

Since in every case the computational part will be done with a hardware implementation, we should now explain this part in more detail, that is the one's pointed out in STEP 3.

This part should take input data from the real world and convert them in linguistic values, compute the linguistic values, and perform another conversion to interface our system with the real world. We have to perform the linguistification and delinguistification operations, i.e. transform a crisp input value in a linguistic one, similar to the fuzzification step in the fuzzy systems. After this we have to transform a linguistic

value (our output) in a more comprehensible way for the external world, that is our controlled system, similar to the defuzzification. Between these two operations there is the computation of the linguistic equation. Since in the image processing application we have to compute more than one linguistic equation, and use all their results to define the actual output, we need also one circuit to perform this operation.

Note that all these parts will be used also during the first tuning operations, with the difference that in this case we are going to compute the linguistic equations in order to fit the output membership definition to the input data.

The principal blocks of the computational part for the HW system are:

- Linguistification circuit
  - comparator
  - circuit for the extraction of the radix for a 2^{nd} order polynomial
  - saturation circuit
- Delinguistification circuit
  - saturation circuit
  - adders and multipliers
  - comparator
- Circuit to perform a linear combination
  - adders and multipliers
- Memories

Fig. 6 shows a general scheme for this computational part.
The tuning phase, described previously in the STEP 2, is compound basically with two blocks:

- Circuits for the extraction of **min**, **max** and **mean** value from the experimental input data
- Circuit to perform a linear regression

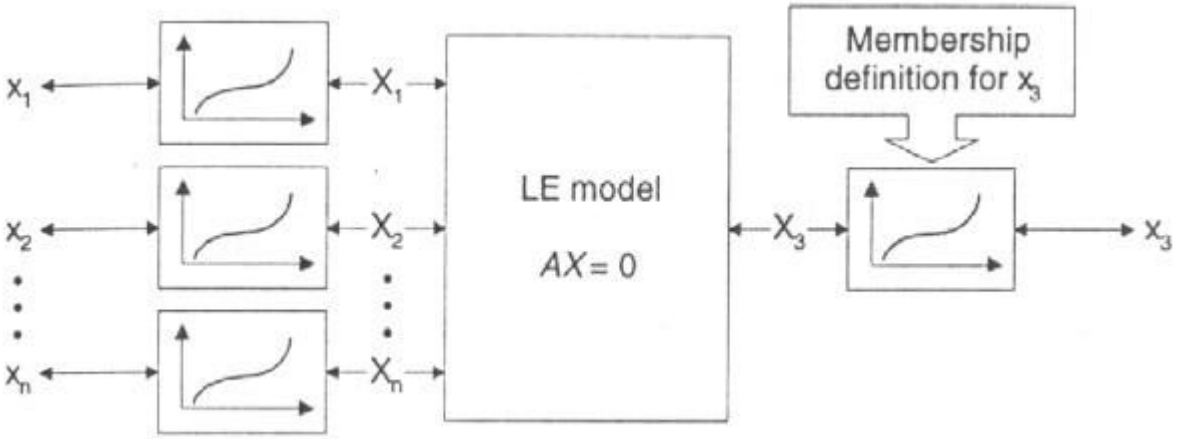


Figure 6. A computational part of the Linguistic Equation system.

More in detail we can now explore every single block.

Circuit to perform a linear combination

This is the block with which we evaluate the linguistic equations, and usually is implemented with adders and multipliers. It receives linguistic input values, and the coefficient of every linguistic equation we need to evaluate.

We should now study more in depth this block for the image processing application. For this reason we have to divide this evaluation in two principal steps. In the first we compute every equation, i.e. by doing a linear combination between coefficients and linguistic values for the input variables and finding at the same time the maximum and minimum evaluation value. In the second step we assign the linguistic value at the central pixel, i.e. the output of our system.

Step I:

Let's suppose that we have to evaluate n = 1…N different equations. During this operation we should multiply every input value by a coefficient that could be +1 or −1, so we can simplify these operations by using add/subtract circuit controlled by coefficients, instead of more heavy multipliers, that helps to reduce the final complexity of the circuit. Finally, to obtain the right linguistic result for that operation we should normalise it.

To simplify more the circuit we can do other considerations. We can change the shape of the coefficient vectors, by just using 0 or −1 values, and normalising the result by a simple mean operation on the variables with non zero coefficient. This means that for each rule we are going to analyse only the window area the central pixel should belong to. Testing on this kind of methods have proved to be very efficient.

Taking coefficient vectors with only +1 or 0 the final configuration will use only add??ers, with an inverter on the output.

Once we have computed the output T(n), we should also compute the coefficient c(n).

$c(n) = T(n)/M$ where M is the linguistic input value of the central pixel

If this value is bigger than 1 we recognise that the actual input pixel configuration fits quite good with the shape of the n-th equation, i.e. the n-th equation is a "winner" one, and we should compute the value $T(n)\cdot c(n)$ to update the value of a register S1, while the value of c(n) is going to update a second register S2, see Fig. 7c.

At the same time an independent circuit will compute the value of the minimum and maximum T(n).

Step II:

After the evaluation of every linguistic equation, we should still evaluate the right value of the system's output.

If the value of S2 is nonzero, the output value will be S1/S2, that is a weight combination of all the results of the "winners" linguistic equation.

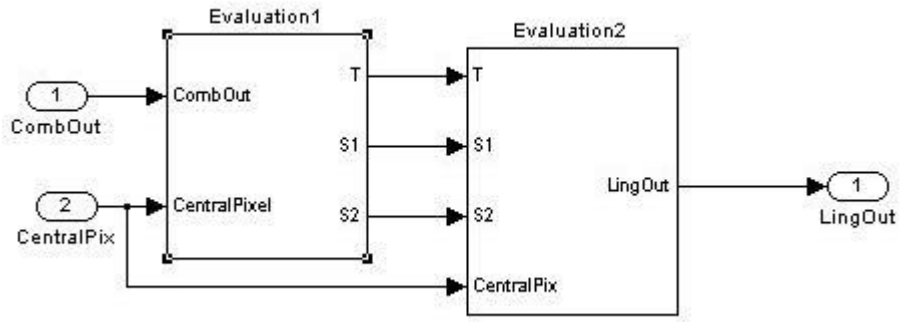
A special situation is when S2 is zero and the actual input window can not belong to any of the configurations represented by the linguistic equations. In this case, we have to try to leave the input value for the central pixel without any particular changes. We assign the output value by computing the mean value between the input central pixel and T, where T is the value of T(n) maximum or T(n) minimum with the bigger amplitude. The mean operation is implemented with an adder and a shift register, but in order to found the value of T we have to implement two absolute operators, and a comparator.

We can implement also these operations by looking at the sum T(n)max+T(n)min. Assuming that T(n)min is always negative and T(n)max is always positive, since we are working with linguistic value, and there should be at least one equation that fits a little bit with the actual input configuration, and one that is very far from that, just looking at the sign of the sum we can choose which has the biggest absolute value.
The schematics in Fig. 7 show more in detail this block.

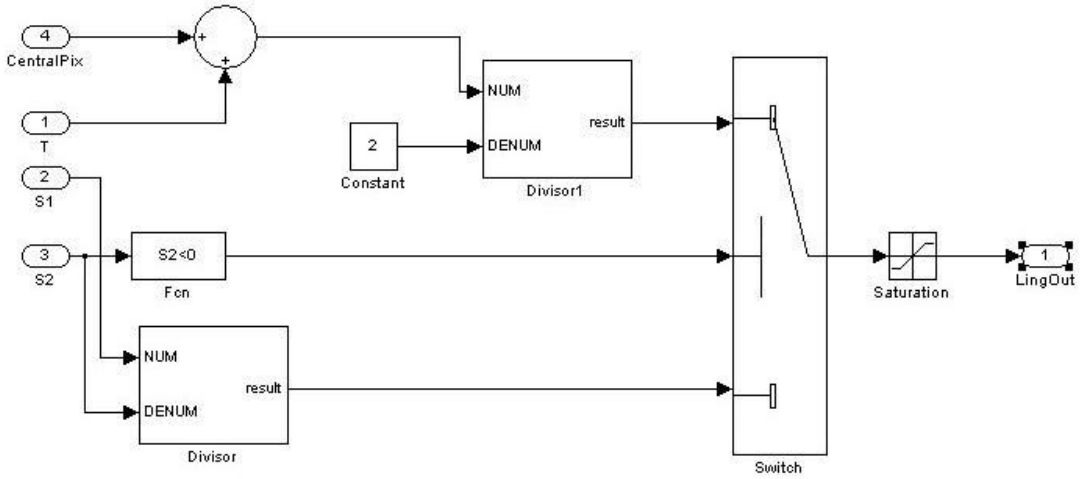
Circuit for the extraction of the radix for a 2^{nd} order polynomial:
The inputs of the linguistification block are the real input value, and the coefficients of the input membership definition. The heaviest part of that circuit is the transformation of the input values in linguistic ones, that means founding the radix for an 2^{nd} order equation $y=a \cdot x^2+b \cdot x+c$.

This operation should be done simultaneously for all the inputs, i.e. in our case 25 input values representing the grey level of the actual input window configuration, and with the same membership definition coefficients.

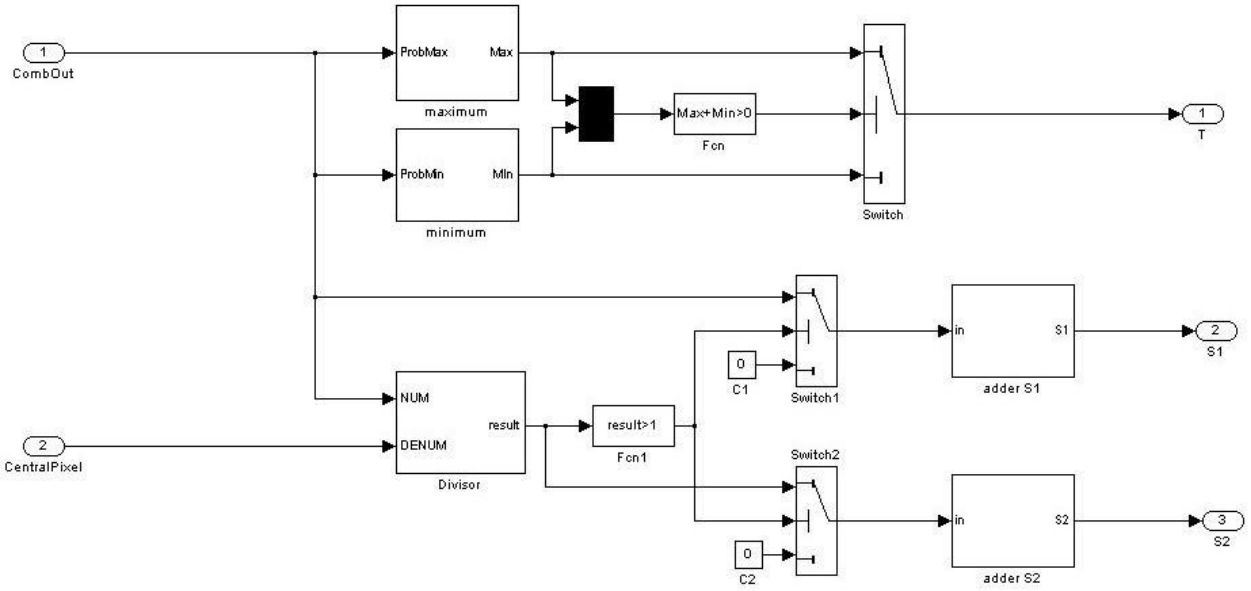
The critical part is how to perform the division and the square root circuits. There are a lot of different solutions for both the circuits (see for example [7 - 9] for the basic theory), and the problem seems to be just how we are going to choose the best schematics. This is driven mostly from which kind of performance we need, and which kind of area, time and power constrains we have in our system.



7a). General scheme for the output evaluation blocks



7b). Output evaluation block (Evaluation 2)



7c). Max-min circuit and evaluation block for the "winner" equations (Evaluation 1)

Figure 7. Schematic for the evaluation of the Linguistic Equation.

We can also try to simplify our problem by doing some approximations, using lower precision in the computation of that kind of results. For this reason we need to check what happens to the performance of the system, and in some cases it seems to be enough strong to allow that kind of changes.

A different kind of approach is to use a look-up table to represent our membership definitions. In this case we can perform high resolution data processing, but we need to use more memory than in the other solution, especially for that application with a lot of input variables, like in our case. On the other hand, since all the input variables have the same kind of membership definition (or membership function), this kind of approach seems to be very useful.

Delinguistification circuit

The input of this block is the linguistic value for the output and the coefficients for the optimised output membership definition. We have to evaluate the solution of an II order equation, and it is the inverse operation respect to the one executed in the linguistic block.

Also in this case we can implement the circuit using LUT.

In order to define better the hardware learning approach, we can briefly describe his basic blocks.

min, **max** and **mean** circuits:

There are not particular problems in this kind of circuit, but since we need the membership definition's coefficient for the parabola, we need also to compute these values from the particular points.

We can now describe briefly how to compute these values.
Given the co-ordinates of the points [(0,y0) (-1,y1) (-2,y2)] e [(0,y0) (1,y'1) (2,y'2)], i.e. **min, meanL, mean** and **mean, meanH, max,** we have to compute the coefficients of the parabola fitting in that points:

We obtain two parabola $y=a\cdot x^2+b\cdot x+c$ and $y=a'\cdot x^2+b'\cdot x+c$ where:

$a=(-2\cdot y1+y2+y0)/2$ $b=(y2-4\cdot y1+3\cdot y0)/2$ $c=y0$ with $x\in[-2,0]$

$a'=(-2\cdot y'1+y'2+y0)/2$ $b'=(-y'2+4\cdot y'1-3\cdot y'0)/2$ $c=y0$ with $x\in[0,2]$

These 5 coefficients [a b c a' b'] will be used in the evaluation of the linguistic input values for the system.

To compute these coefficients we need adders, multipliers and dividers. In this particular case the division is not a problem since the divider is a power of two, and we can perform the operation by a shift register.

Circuit to perform a linear regression

We need this block during both the initial learning and optimisation step. We have to perform a linear regression when we need to extract the linguistic equation from the experimental data, and a 2^{nd} order regression when we need to tune the output membership function with the input data, and the results of the evaluation of the linguistic equations.

Note that once we found the regression coefficients for all the input variables, we still need to choose the correct shape of the linguistic equation. In order to use this system in an image processing environment, we don't need at all this type of learning step. This is a big simplification for our final hardware system.

Usually the best way to perform arithmetic operations seems to be using a floating-point number representation, in order to obtain higher precision in the final result, even if the hardware complexity will increase. In this case we can also choose a fixed

point representation, since all the computations are made with linguistic values, i.e. numbers that belong to the interval [–2…2]. We have just to decide the level of the precision we need, that is which is the best number of digits we want to use for our representation.

To obtain a low complexity circuit we can decide to implement a fixed-point arithmetic circuitry.

6. Conclusion and further research

Linguistic Equation Approach is a very powerful and compact method for the representation and the control of conventional expert systems. This approach has been developed mainly for the control of industrial processes, but because of the particular structure of this control system, that uses appropriate linear relations between the system variables, we have been trying to apply the theory of the Linguistic Equations also in the process of digital information.

In particular a noise suppression algorithm has been developed, as an application in the field of image processing. We have illustrated the effectiveness of the proposed approach by performing a series of tests on images dependent with different kind and levels of noise, allowing to solve properly the problems connected with the managing of large set of control rules characterising that kind of applications. Good results could also be obtained in those cases in which we can't use a non-noisy image as a tuning and optimising tool, or when we haven not enough information about the type of the noise is affecting our images. This is very important especially in some kind of application where a real time information processing is needed, i.e. extracting information from satellite images (SAR).

A schematic about the possible Linguistic Equation architecture of a hardware system has been developed, with particular regard to the image processing application. In this case we can obtain a low complexity circuit configuration, and this is one of the reasons of choosing image analysis as a first example of hardware realisation of Linguistic Equations. This is a good characteristic, and means that this methodology can be used successfully in all that applications where we have to handle big quantity of data, and a lot of input variables. This is valid also in cases they are not directly joined with an industrial process, where the main problem is just that we have to use simple computation structures. Low complexity and low computational time are also required. This is a fundamental topic, especially thinking to the software and hardware implementation of real-time applications.

Actually only some blocks of the schematic have been simulated on a software environment and we are working on the tuning and optimisation of each part of the circuit, in order to test the system on a real application. In order to improve the filtering results, we are going to plan some tests with different kind of window shapes and weigh vectors.

Figure 8a. Original 256x256 Lena image.

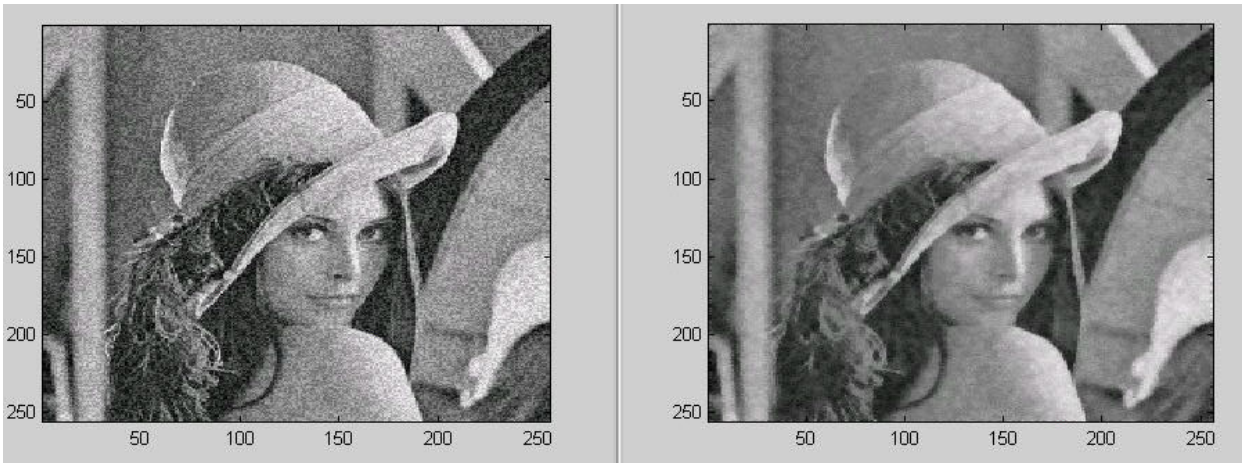


Figure 8b. Comparison between uniform noise [-32,+32] and the LE filtered image.

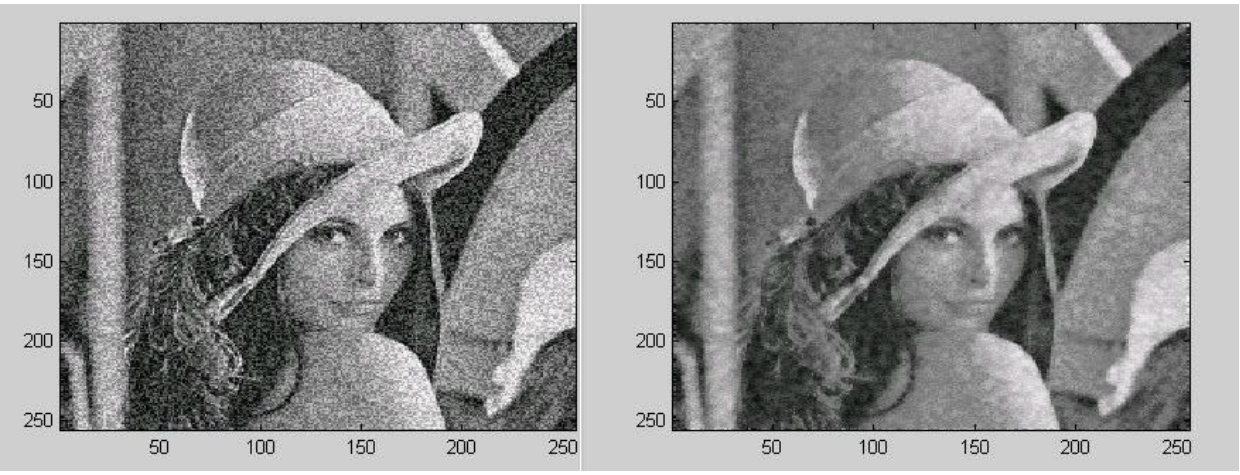


Figure 8c. Comparison between uniform noise [-48,+48] and the LE filtered image.

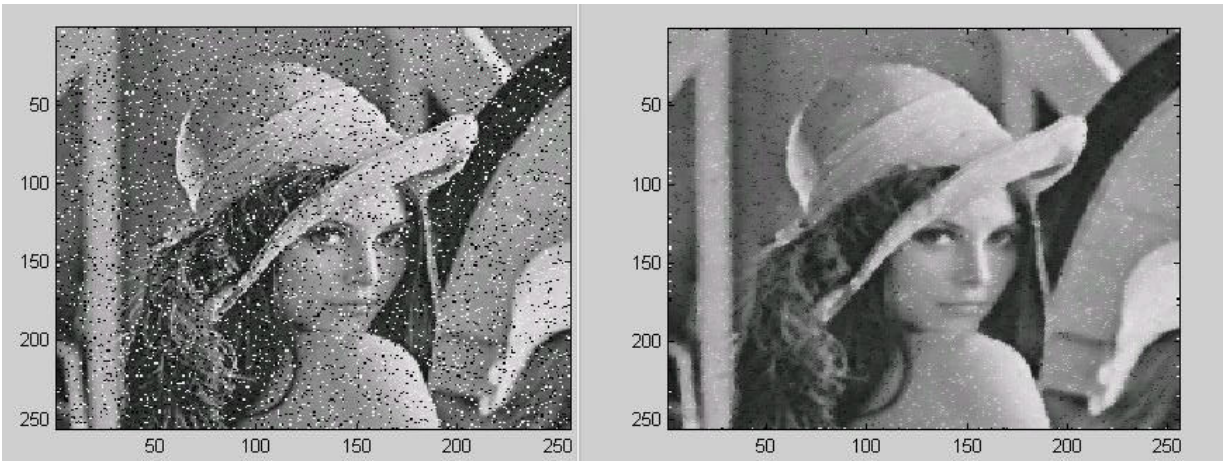


Figure 8d. Comparison between Salt&Pepper noisy image and the LE filtered image.

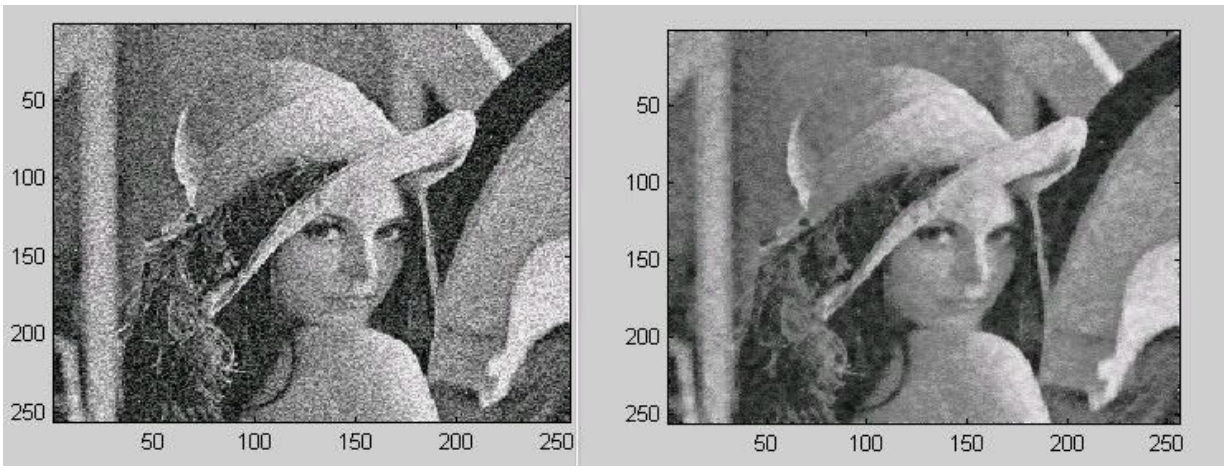


Figure 8e. Comparison between Gaussian noisy (variance 32) image and the LE filtered image.

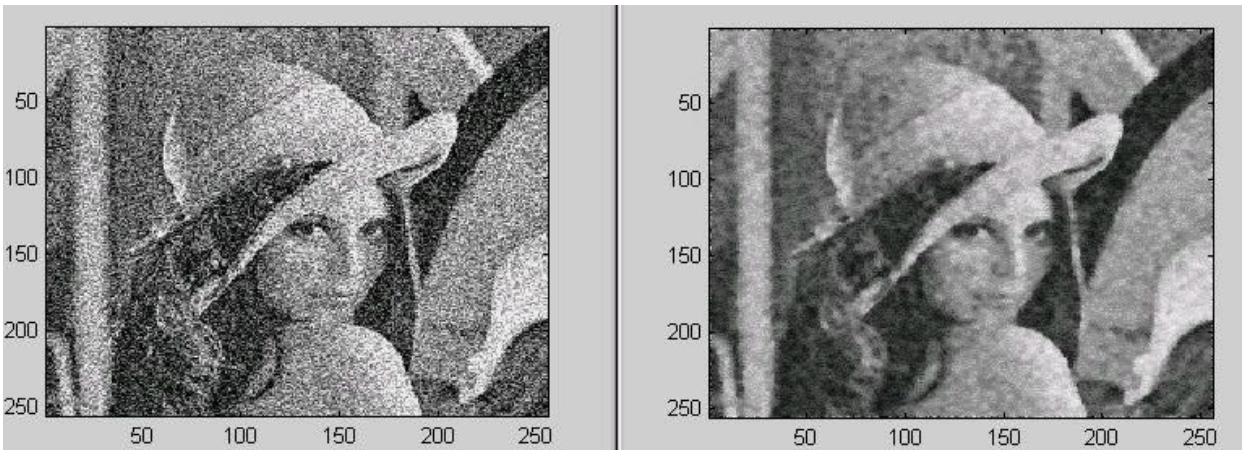
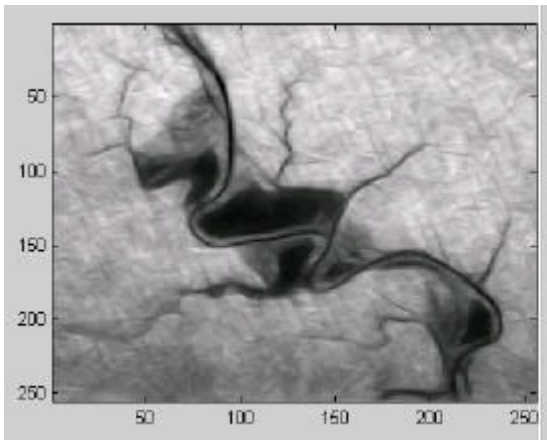
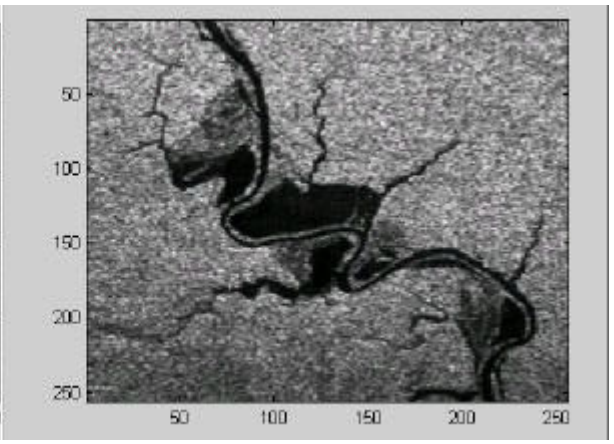


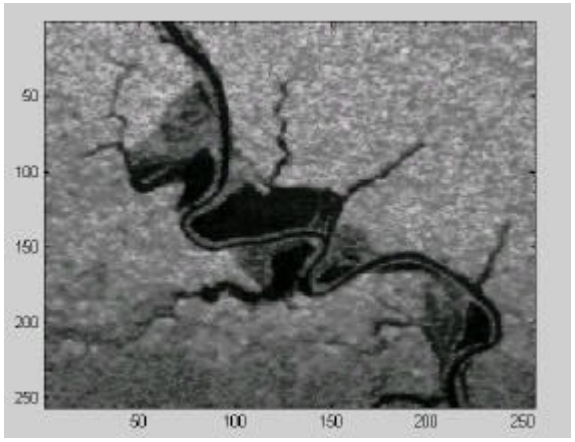
Figure 8f. Comparison between Gaussian noisy (variance 48) image and the LE filtered image.



a) Wavelet filtered SAR image



b) Original noisy SAR image



c) LE filtered SAR image

Figure 9. SAR images.

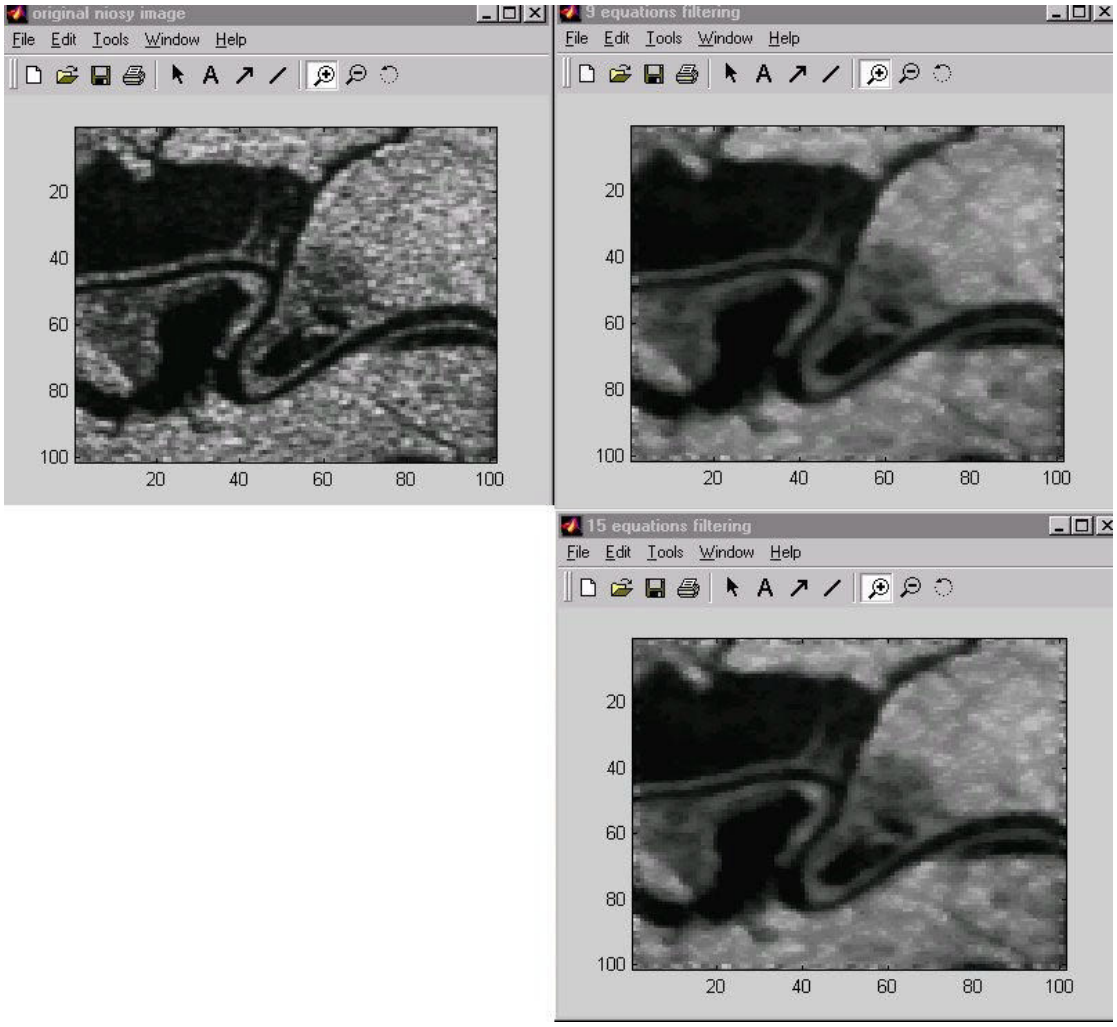


Figure 10. Comparison between LE filtering of a 50x50 SAR image with 15 rules or 9 rules.

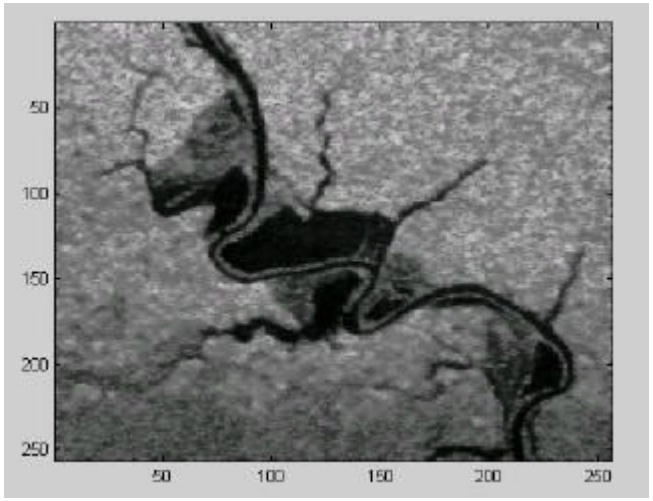


Figure 11. LE filtered SAR image by a system tuned on the noisy SAR image.

Literature

[1] E.K. Juuso, 1996. Linguistic Equations in System Development for Computational Intelligence. Proceedings of the 4th European Congress on Intelligent Techniques and soft Computing EUFIT'96, September 2-5,1996, Aachen, Germany, Vol. 1, pp.1127-1131.

[2] E.K. Juuso, K. Leiviskä, 1992. Adaptive Expert Systems for Metallurgical Processes. Expert Systems in Mineral and Metal processing, Proceedings of the IFAC Workshop, Espoo, Finland, August 26-28,1991, IFAC Workshop Series, 1992, Number 2, Oxford, UK, Pergamon, pp.119-124.

[3] F. Russo, 1995. Fuzzy Systems in Instrumentation: Fuzzy Signal Processing. Instrumentation and Measurement Technology Conference, IMTC/95. Proceedings. Integrating Intelligent Instrumentation and Control, IEEE , 1995, pp.735-740.

[4] F. Russo, G. Ramponi,1995. A Noise Smoother using Cascaded FIRE Filters. Fuzzy Systems. International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems and The Second International Fuzzy Engineering Symposium., Proceedings of 1995 IEEE International Conference, Vol.1 , 1995 , pp.351 –358.

[5] F. Russo, G. Ramponi, 1997. Noise Cancellation Using Nonlinear Fuzzy Filters, Proceedings of the IEEE Instrumentation and Measurement Technology Conference IMTC97, Ottawa, Canada, May 19-21, pp.772-777.

[6] I. Pitas, A.N. Venetsanopoulos, 1990. Nonlinear Digital Filters: Principles and Applications, Kluwer.

[7] P.Soderquist, M.Leeser. Division and Square Root: Choosing the right Implementation, IEEE Micro Volume: 17 4,pp.56-66.

[8] H.R Srinivas, K.K Parhi. High-Speed VLSI Arithmetic Processor Architectures using Hybrid Number Representation. Computer Design: VLSI in Computers and Processors, 1991. ICCD '91. Proceedings, 1991 IEEE International Conference on , pp. 564-571.

[9] K.Hwang, Computer Arithmetic Principles, Architecture and Design, John Wiley & Sons, 1979.

[10] M. Salmeri, M.Re, G.C. Cardarilli, R. Lojacono, A. Salsano, 1999. SAR Images Filtering: a Fuzzy Approach, Proceedings of TOOLMET'99 Symposium-Tool Environments and Development Methods for Intelligent Systems, 15-16 April,1999, Oulu (Finland), pp.176-181.