Computational Analysis of Canny & Binary Fuzzy Rough Set Model Based on Triangle Modulus Edge Detectors

Raúl Pardo
Dept. de Sistemas Informáticos
E. Superior de Ingenieria Informática de Albacete
Universidad de Castilla - La Mancha
Campus Universitario. 02071-Albacete, Spain
Email: RaulPardo@dsi.uclm.es

Fernando L. Pelayo
Dept. de Sistemas Informáticos
E. Superior de Ingenieria Informática de Albacete
Universidad de Castilla - La Mancha
Campus Universitario. 02071-Albacete, Spain
Email: FernandoL.Pelayo@uclm.es

***Abstract*—In this work we present a formal comparative over the computational complexity of two edge detectors, in one hand the Canny Edges Detector and on the other an edges detector based on Rough Sets Theory which has been proved as very efficient from the results point of view, so we are now interested in their performances, i.e., we have developed an analysis from the computational point of view.**

To develop such study we have used ROSA Analyser tool which generates the Labelled Transition System, LTS, corresponding to a process specified in the Markovian Process Algebra ROSA. ROSA Analyser takes as input the syntactical representation of the process and once its syntactical structure has been properly layered represented -internally- by the tool, it applies the Operational Semantics of ROSA, so producing the corresponding LTS, which shows all the possible behaviours of the system we are interested in.

A clear advantage, also from the computational point of view, of the encoder using Rough Sets Theory has been found.

I. Introduction

Edge detection becomes a key tool for image processing, in fact, the areas of object classification, both features detection and features extraction and some others are necessarily supported on successful edges detection processes. Also in the field of image compression, edge detectors are very important since they allow to significatively reduce the amount of data necessary to describe an image since the human eye is specially sensitive to edges and some remainder information is "interpolated" by the brain, and also a clear advantage to process digital pictures is founded in medical, engineering and some other fields. These techniques identify some pixels (usually lines) in which a change appears, i.e. image features change sharply or in general, some kind of discontinuity appear.

They are mainly focused on analysing the values which encode an image and when they are over or below some thresholds they are taken as edges, but in order to be more accurate they run, the most times over different domains of representation of the former images. These new domains can be frequencies, chrominance and luminance, levels of gray, red-green-blue, and so on, even more some times they use more than one set to describe an image, as it is the case of one of the edge detectors we have analysed.

In 1986 John Canny developed a a theoretical system in which determines the best way to get the edges of a picture by means of establishing three constrains in which his theory was based. Then, taking this constrains into account he reached the conclusion that the optimal detector has a simple approximate implementation in which edges are marked at maxima in gradient magnitude of a Gaussian-smoothed image.

Apart from this characteristics Canny defined an algorithm in which the edge detection is made by means of four steps. Nowadays the Canny Edge Detector is a main reference in this field, in fact, many of modern edge detector algorithms are based in the very same constrains and so they search the edges based in the gradient magnitude.

Rough sets theory which was firstly defined by Z. I. Pawlak in 1982, have been successfully applied to artificial intelligence, image processing, and a lot processes which include a grade of uncertainty in their formulation and/or data characterization. Rough sets become a formal approximation to a fuzzy set by means of a couple of sets called upper approximation set (negative region) and lower approximation set (positive region).

Some algorithms based on Binary Fuzzy Rough Sets and Triangle Modulus Functions do compute the edges mainly by calculating the "difference" between these upper and lower approximations sets, sometimes also including the original data on its formulae.

The Markovian Process Algebra ROSA is a formalism able to capture non-deterministic, probabilistic and temporal behaviours of a process [3]. Moreover, it provides a very straightforward and clear way to specify processes. In fact, it has been already used to specify and to analyse some cognitive processes as memorization, see [4].

It is well known the *state explosion* problem which raise the computational cost of generating the LTS to exponential, mainly due to this fact Process Algebras are not as used as its analyzing capabilities would indicate. In an initial attempt

to get a more practical built of the LTS, our tool is able to detect syntactically identical states and a little bit more, i.e., commutability and so on. Thus it shouldn't appear many instances of almost syntactically identical nodes. At this stage of development, ROSA Analyser is able to show significative differences on the computational costs of the edge detectors analysed.

This paper is structured as follows, next section roughly describes how ROSA Analyser works, sections 3 and 4 describes both Canny and Rough Sets Edges Detector algorithms respectively and their corresponding specifications in ROSA; Section 5 shows the LTS generated by the tool over them, so allowing an easy comparison over its computational costs. Conclusion and Future Work section, ends the paper.

II. ROSA Analyser tool

ROSA Analyser has been developed in JAVA, since having to deal with complex data structures. In particular ROSA Analyser works on two data structures, Syntax and Semantics trees, the last one corresponds with the LTS.

As input of the LTS generator, we have a ROSA syntactical expression, this input data is still not optimized to perform a semantics analysis, because of this, in the first step of the LTS generator, ROSA Analyser will generate a Syntax tree. This step involves the generation of a binary tree in which the higher syntactical priority the elements have, the closer to the top they are located. This is a key step since not only the pattern matching of semantics rules must be done, but also the conformance with the conditions of the upper side of the rules must be checked to determine whether a rule must be applied or not.

Once the Syntax tree has been built, the next step, is generating the LTS which has to contain rather complex information, in order to support it a semantics tree data structure has been also defined. In addition, it has been defined a core class for selecting (once checked) the set of rules that can be applied to every new generated process.

Our final goal regarding the tool is to provide a ROSA Analyser as usable as possible, so that we are working on an heuristics for only unfolding the "more promising branches" this leads us to define a metrics over the set of ROSA processes and so we need to guarantee that two semantically identical processes have to be also syntactically identical, so that, some fist steps in this line has been done in this initial development stage. So far, ROSA Analyser provides an output where can be seen the nodes and the transitions of the LTS.

ROSA Analyser is also able to export the LTS to pdf, svg, jpg and png; using graphviz, so providing a graphical view of the Semantics tree.

III. Canny Edges Detector

The Canny Edge Detector Algorithm, was considered the best algorithm to find the edges in a digital pictures in 80's [1]. This algorithm is based on find abrupt changes in the gradient value of the picture, since this values correspond with the picture edges. In order to reach the best results in his edge detector three performs criteria was defined:

1) *Good Detection*. The points marked as edges must have low probability of not being an edge. The mathematical representation of this condition is:

$$SNR = \frac{A\left|\int_{-W}^{0} f(x)dx\right|}{n_0\sqrt{\int_{-W}^{W} f^2(x)dx}} \tag{1}$$

2) *Good localization*. The points which has been marked as edges has to be as near as possible to a true edge, reaching a high accuracy level in the edge detection, the formal specification of this property involves the location of the point marked as an edge and the distance between this point and the true edge:

$$Localization = \frac{A|f(0)|}{n_0\sqrt{\int_{-W}^{W} f^2(x)dx}} \tag{2}$$

$$Distance = \pi\left(\frac{\int_{-\infty}^{\infty} f^2(x)dx}{\int_{-\infty}^{\infty} f'^2(x)dx}\right)^{1/2} \tag{3}$$

3) *Only one respond to a single edge*. This criteria establish that the algorithm does not mark as an edge the pixel where has been found the edge and the surrounding pixels, because it could be a consequence of having another edge within its neighbourhood.

An extended justification of the above criteria and its mathematical representations can be found in [1]. To achieve the best values for this three criteria, its product had to be optimized and four steps were defined by means of this optimization.

Step 1 - Smooth picture

The first step in this algorithm is to smooth the picture, due to the fact that noise is the focus of many errors during the edge detector process, based on 1 Canny reached the conclusions that the best way to smooth the picture is by means of a Gaussian filtering so it can be done using a mask which follows the function:

$$G(x,y) = ke^{-\frac{(x+y)^2}{2\sigma^2}} \tag{4}$$

The Gaussian filters are characterized by having a high value in the central pixels and this value decreases as we move away this pixel, the strength of this decreasing is determined by the parameter σ.

Several studies shows that a good filter to get the best values in the edge detector is:

$$\frac{1}{115}\begin{pmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{pmatrix} \tag{5}$$

Step 2 - Gradient Computation

The second step in the algorithm involves the calculation of the picture gradient. Once the picture has been smoothed, it have reached an optimal conditions to get both the module and the angle gradient. As it is well known the gradient operator is based in the first derivative and, for a picture will be calculated with respect x and y, in this way, for each pixel it has to be computed:

$$||\nabla G|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} \tag{6}$$

$$\theta = tan^{-1}\left(\frac{\partial f}{\partial x}/\frac{\partial f}{\partial y}\right) \tag{7}$$

The partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$ are represented by the matrices G_x and G_y respectively. So that in order to implement the calculation of G_x and G_y, the Sobel operator becomes a very good option according to Canny's studies. An example of a mask to apply a Sobel operator for getting $\partial f/\partial x$ or what is the same G_x, is:

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{8}$$

As can be seen in 8 the Sobel operator so defined, increments the values of horizontal components. It happens analogically with G_y by incrementing the vertical values:

$$\begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{9}$$

Once the Sobel operator has been applied both with respect to x and to y, the module and the angle of the gradient can be calculated, and as a result of this, we get the matrix M with the gradient module value of each pixel and A with the gradient angle value. It is important to point out that the angle values are discretized, so that it only could take angle values equal to 0, 45, 90 and 135 degrees for each pixel, depending on the interval in which the value are located $[0 - 22.5]$ are changed to 0, $[22.5 - 67.5]$ to 45, $[67.5 - 112.5]$ to 90 and finally in the range $[112.5 - 157.5]$ to 135.

Step 3 - Non maximum suppression

By means of this step the third criteria defined is reached. In this step the gradient module values of the pixel which are non maximum are changed to 0, in order to do this, the value for the pixel is compared with its nearest neighbours and if the pixel has not the higher value in this pixels set it is established as 0. Nevertheless, it gets a thinner picture edges, because only the maximum gradient module values will be checked to determine if they are true edges.

Step 4 - Hysteresis

In this step there will be established two thresholds and two conditions to mark a pixel as an edge or not. The need to define two thresholds appears due to the fact that one threshold having a high value, many edges was not detected and, on the other hand, if the threshold has a very low value, it was marked as a edge pixels which was not an edge. Because of this was decided to take two threshold, and the conditions to mark a pixel as an edge are:

1) If the gradient module value is higher than the upper threshold, the pixel is marked as an edge.
2) If the gradient module value is lower than the upper threshold but it is higher than the lower threshold and there is a path from this pixel to another pixel established as an edge (the path it is determined by the gradient angle value) this pixel is marked as an edge.

A. Canny Edge Detector ROSA specification

In order to be able to make a formal analysis about performance in the Canny Edge Detector algorithm, we have specified it with ROSA Process Algebra. The strategy followed to specify the algorithm is to split the whole process in several subprocess, so as to reach a clear and more precise view of the algorithm behaviour.

The processes in which the whole algorithm has been split, do not match with the steps involved in the theoretical algorithm definition; It is because the algorithm specification take sometimes a group of steps as a single encapsulated action, since the complexity of each one is not enough to take more than one action, i.e. there is no variability on the execution of that group.

Deleting noise

The process with which is specified the picture smoothing is the process DEL_NOISE, the process behaviour is pure sequential. The actions involved in this process are:

- $Read$. This action represents the reading of the picture.
- $GaussM$. This action involves the task which apply the Gaussian filter in the read picture.
- $GradSmt$. This action is used as a synchronization point, where the Sobel operator is initialized, so that to execute these processes concurrently.

Finally, the DEL_NOISE process is specified as follows:

$$DEL\_NOISE \equiv < Read, \alpha > . < GaussM, \beta > . \\ < GradSmt, \infty > \tag{10}$$

Gradient

Gradient process $GRAD$, see theoretical Canny Edge Detector description, is formed of many subprocesses, the build of G_x and G_y, get the gradient module and the gradient angle. For this reason the specification will be split in several processes.

Firstly, it initializes the Sobel operator, this is represented as the action $SobelCM$, then in order to get G_x and G_y the process must synchronize with the previous process (DEL_NOISE) by means of $GradSmt$ action, since the Sobel operator must be applied in the smoothed picture. Once the Sobel operator is initialized and the picture is smoothed, it is ready to provide both G_x and G_y, this is represented in the ROSA specification by the $GxGy$ action.

Once reached this point the process will start to calculate the module gradient as process MOD and the angle gradient as process ANG and finally ends the algorithm with the hysteresis process $HYST$, therefore the ROSA specification of the Canny Edges Detector is:

$$GRAD \equiv < SobelCM, \gamma > . < GradSmt, \infty > .$$
$$< GxGy, \delta > (MOD||_{\{EndGrad\}} ANG).HYST \quad (11)$$

In order to fully specify the edge detector we will define more detailed the processes MOD, ANG and $HYST$.

Gradient Module

The gradient module process MOD is formed of three actions:

- $ModVal$. This action represents the calculation of the gradient module, to do this operation 6 will be used for each pixel.
- $NonMax$. In the theoretical edge detector description, this action involves a whole step, but from the point of view of ROSA specification, this constitutes a single action which changes the module value of the pixel when it is not a maximum value.
- $EndGrad$. The processes MOD and ANG are considered to be executed at the same time. They synchronize in this action.

The ROSA specification for this process is:

$$MOD \equiv < ModVal, \theta > . < NonMax, \lambda > .$$
$$< EndGrad, \infty > \quad (12)$$

Gradient Angle

This gradient angle process ANG has three actions as the previous process, but the tasks to do in each action are a bit different:

- $ArctgGxGy$. This actions represents the calculation of the angle value for each pixel by means of 7.
- $Discret$. By this action the angle values are discretized as it is defined in the theoretical description.
- $EndGrad$. As was said in the previous process, by means of this action the processes MOD and ANG are synchronized, because they are executed concurrently.

The ROSA specification for this process is:

$$ANG \equiv < ArctgGxGy, \rho > . < Discret, \mu > .$$
$$< EndGrad, \infty > \quad (13)$$

Hysteresis

The hysteresis process $HYST$ represents the hysteresis step defined in the theoretical algorithm, because of it is only an operation which mark a pixel as an edge depending on two conditions, there is not a need to split in more actions or processes, in this way the process is specified with ROSA as follows:

$$HYST \equiv < Hyst, \phi > \quad (14)$$

Full specification for Canny Edge Detector process

Once defined all these processes, the whole ROSA process for the Canny Edge Detector is:

$$CANNY \equiv DEL\_NOISE||_{\{GradSmt\}} GRAD \quad (15)$$

In order to get a better view for the above process we can write:

$$CANNY \equiv DEL\_NOISE||_{\{GradSmt\}} < SobelCM, \gamma > .$$
$$< GradSmt, \infty > . < GxGy, \delta > .(MOD||_{\{EndGrad\}} ANG).$$
$$HYST \quad (16)$$

Analysis using ROSA Analyser

Now we are going to show how ROSA Analyser tool runs over Canny edge detector specification. For the sake of space we are using capital letters for processes, and lower case letters for actions.

According to the previous specification, the following table establishes the equivalence between the process defined and the corresponding input for the tool.

ROSA Syntax	Tool Syntax
MOD	M
ANG	E
$HYST$	H

TABLE I
EQUIVALENCE BETWEEN PROCESSES

Similarly, table II relates defined actions in ROSA to the actions used in the tool.

ROSA Syntax	Tool Syntax
$Read$	r
$GaussM$	a
$GradSmt$	g
$SobelCM$	s
$EndGrad$	z
$GxGy$	b
$ModVal$	x
$NonMax$	n
$ArctgGxGy$	m
$Discret$	d
$Hyst$	h

TABLE II
EQUIVALENCE BETWEEN ACTIONS

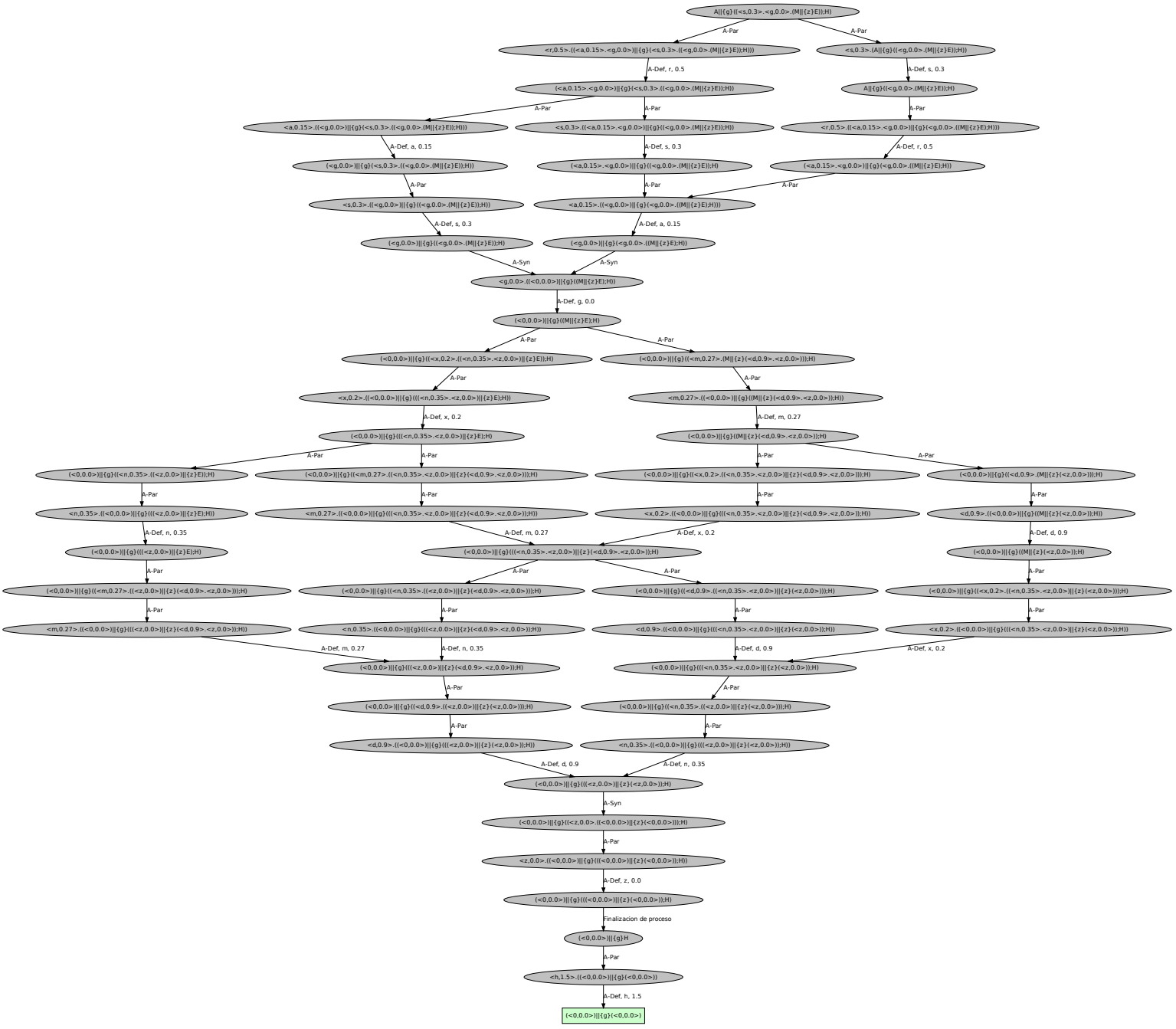

Fig. 1. LTS for Canny Edge Detector Algorithm (http://raulpardo.files.wordpress.com/2012/05/cannysemanticstrees.jpg)

In addition, there is another constrain regarding temporal parameters. ROSA analyser works directly with real numbers in order to make proper calculations with true quantitative results, Since we lack of accurate data about these actions temporal costs, these data have been chosen randomly within what the authors consider reasonable in each case.

Therefore, the ROSA specification for being taken by the tool as input is:

$$M \equiv < x, 0.5 > . < n, 0.4 > .z$$

$$E \equiv < m, 0.57 > . < d, 0.9 > .z$$

$$H \equiv < h, 0.1 >$$

$$CANNY \equiv < r, 0.5 > . < a, 0.15 > .g||\{g\}((< s, 0.3 > .g. < b, 0.3 > .(M||\{z\}E)); H) \quad (17)$$

Finally, once (17) has been introduced in the tool, it builds the whole LTS for the Canny Edge Detector, in which we can see all of the possible behaviours of this algorithm, figure 1.

IV. Binary Fuzzy Rough Sets Edges Detector

The Binary Fuzzy Rough Sets Edges Detector, BFRSED see [2] is based on representing images by means of both sets *upper approximation* and *lower approximation* from which the image detection algorithm is able to detect edges trough quite elementary operations sometimes also involving the original (at most) normalized image. In fact, it is built a binary fuzzy rough set model based on triangles modules defined over a partition of fuzzy sets over the axis of the image. These triangle modules and what is called their *residuation implications* are used to define the *negative region*, upper approximation set, and the *positive region*, lower approximation set.

This BFRSED provides its best results when running over smooth gray level changed images.

The way of working is basically the one presented below:

1) Normalization of gray in which the original gray values of the image with $M \times N$ pixels is normalized so obtaining a representation of the image as $R : X \times Y \rightarrow [0, 1]$, which is considered as a fuzzy relation over the Cartesian Product $X \times Y$.
2) From this representation of the image R, there will be computed the $M + N$ fuzzy sets defined over the coordinates of the image, M fuzzy sets over X, as $|X| = M$ (F_X^i where $i \in \{1, \ldots, M\}$), and N fuzzy sets over Y, as $|Y| = N$ (F_Y^j where $j \in \{1, \ldots, N\}$).
3) From the *triangle modulus* function $T : [0, 1] \times [0, 1] \rightarrow [0, 1]$ defined over R, can be computed the upper approximation set over each $(x, y) \in X \times Y$ as $\overline{R}(x, y) = \vee_{i \in X, j \in Y} T(F_X^x(i) \wedge F_Y^y(j), R(i, j))$
4) The Residuation Implication from T is $\theta_T(a, b) = sup\{c \in [0, 1].T(a, c) \leq b\} a, b \in [0, 1]$ and from it can be computed the lower approximation set over each $(x, y) \in X \times Y$ as $\underline{R}(x, y) = \wedge_{i \in X, j \in Y} \theta_T(F_X^x(i) \wedge F_Y^y(j), R(i, j))$
5) Once the binary relations R, $\overline{R}$ and $\underline{R}$ have been computed, it is time to define three edges detection algorithms:
   - **Algorithm 1**: $B_1 = \overline{R} - R$
   - **Algorithm 2**: $B_2 = R - \underline{R}$
   - **Algorithm 3**: $B_3 = \overline{R} - \underline{R}$

A. Binary Fuzzy Rough Sets Edges Detector ROSA specification

In order to develop a formal analysis over BFRSED, we have specified it by using ROSA. In the same way of the previous specification this algorithm will be specified in split subprocesses, making clearer this specification.

Calculations

This process $CALC$ must be executed for all algorithms, because it involves the initialization of the upper and lower approximations fuzzy sets with which this algorithm works. The actions executed during this process are:

- $GrayNorm$. This action represents the normalization in which the picture gray values are set in [0,1].
- $FuzzySetsGenerat$. This action represents the fuzzy sets building process.
- $TriangleModuleComput$. In which it is based.
- $ResiduationImpliComp$. Which captures a cumulative density function on the previous Triangle Modulus.
- $\overline{R}$ $\underline{R}$ R. These actions are used to process synchronizations, in fact, depending on the algorithm chosen (three different options) three different pair of them will be used.

Therefore, as this process is mainly sequential, ROSA specification for this process is:

$$< GrayNorm, \alpha > . < R, \infty > . < FuzzySetsGenerat, \beta > . < TriangleModuleComput, \gamma > . < \overline{R}, \infty > . < ResiduationImpliComp, \mu > . < \underline{R}, \infty > \quad (18)$$

Algorithm 1

The specification of these three algorithms calculating edges according to the upper and lower approximation sets are quite simple since they all mainly apply a kind of substraction over the values given for each pixel by a pair of relations belonging to the set $\{R, \overline{R}, \underline{R}\}$; Therefore, process Calculations only have to synchronize by means of R and $\overline{R}$ and then to calculate the sets subtraction, i.e. action B_1. In particular, the process $ALGO_1$ is specified as follows:

$$ALGO_1 \equiv R.\overline{R}. < B_1, \rho > \quad (19)$$

Algorithm 2

In the same fashion that algorithm 1 has been specified, this synchronizes by means of R and $\underline{R}$, to calculate the sets subtraction, action B_2. The process $ALGO_2$ is:

$$ALGO_2 \equiv R.\underline{R}. < B_2, \phi > \quad (20)$$

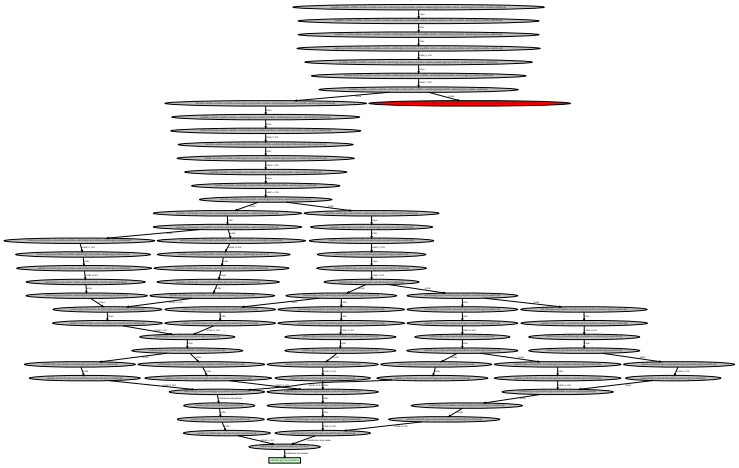

Fig. 2. LTS for Rough Sets Edge Detector Algorithms (http://raulpardo.files.wordpress.com/2012/05/fuzzysets.jpg)

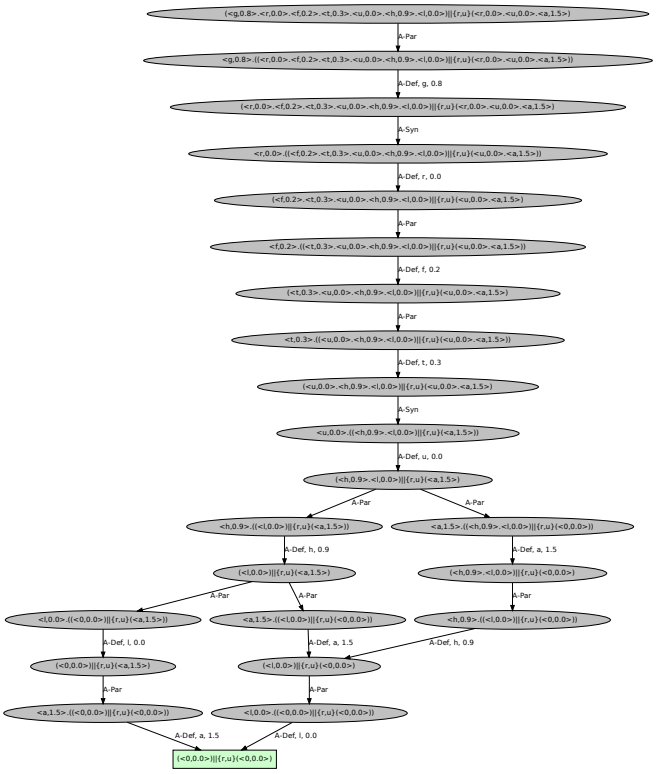

Fig. 3. LTS for Rough Sets Edge Detector Algorithm 1 (http://raulpardo.files.wordpress.com/2012/05/fuzzysetsalgorithma.jpg)

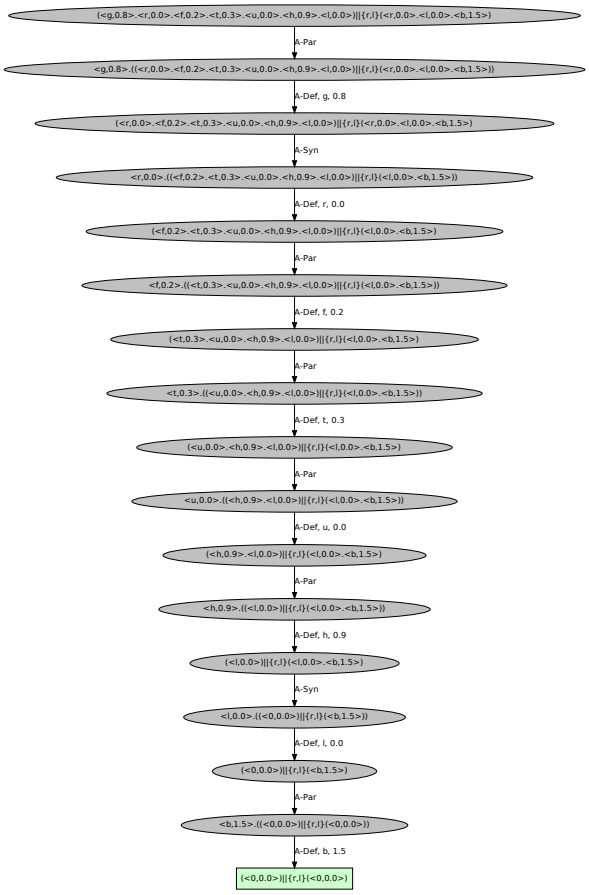

Fig. 4. LTS for Rough Sets Edge Detector Algorithm 2 (http://raulpardo.files.wordpress.com/2012/05/fuzzysetsalgorithmb.jpg)

Algorithm 3

Finally the third algorithm is defined by synchronizing with the upper and the lower approximations fuzzy sets and then it estimates the edges by the corresponding substraction; Its specification in ROSA is:

$$ALGO_3 \equiv \overline{R}.\underline{R}. < B_3, \varphi > \tag{21}$$

Analysis using ROSA Analyser

According to the input constrains for ROSA Analyser, in this algorithm also has been made two equivalence tables for processes and actions, respectively (tables III and IV).

ROSA Syntax	Tool Syntax
$CALC$	C
$ALGO_1$	A
$ALGO_2$	B
$ALGO_3$	D

TABLE III
EQUIVALENCE BETWEEN PROCESSES

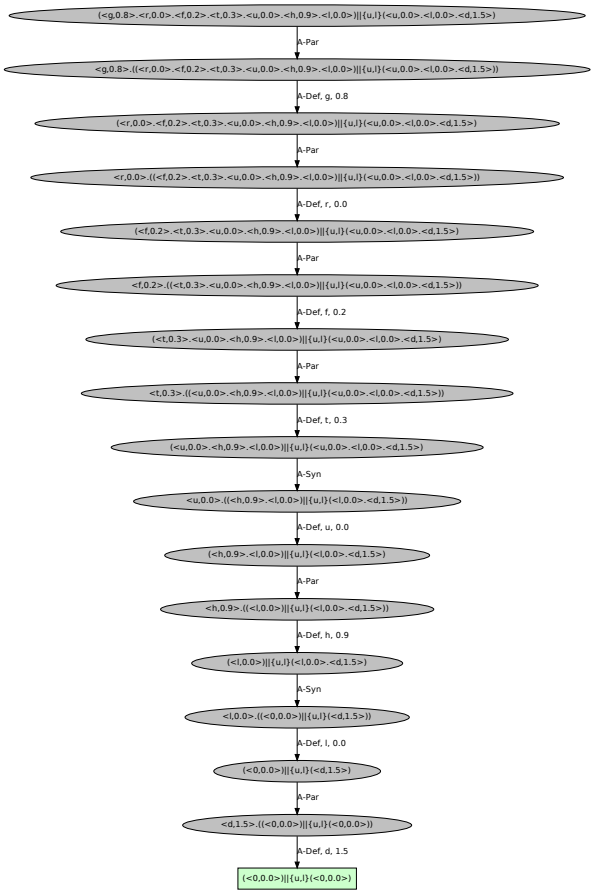

Fig. 5. LTS for Rough Sets Edge Detector Algorithm 3 (http://raulpardo.files.wordpress.com/2012/05/fuzzysetsalgorithmd.jpg)

ROSA Syntax	Tool Syntax
$GrayNorm$	g
$FuzzySetsGenerat$	f
$TriangleModuleComput$	t
$ResiduationImpliComp$	h
R	r
$\overline{R}$	u
$\underline{R}$	l
B_1	a
B_2	b
B_3	d

TABLE IV
EQUIVALENCE BETWEEN ACTIONS

Furthermore, with regards to the temporal parameter constrains in the same way that Canny Edge Detector, the temporal behaviour has been randomly established according to the computational cost by which the actions are characterized.

Finally, the specification for Binary Fuzzy Rough Sets Edge Detector to be "computed" by ROSA Analyser Tool is:

$$C \equiv < g, 0.8 > .r. < f, 0.2 > . < t, 0.3 > .u. < h, 0.9 > .l$$
$$A \equiv r.u. < a, 1.5 >$$
$$B \equiv r.l. < b, 1.5 >$$
$$D \equiv u.l. < d, 1.5 >$$

$$BFRSMBTM \equiv C||\{r,u,l\}A||\{r,u,l\}B||\{r,u,l\}D \quad (22)$$

Figure 2 shows the Labelled Transition System generated by the Tool with the Binary Fuzzy Rough Set Model Based on Triangle Modulus Edge Detector, $BFRSMBTM$, as INPUT, i.e. once run over (22)

But, this is the specification of the computation of all the three algorithms for edge detection from the upper and lower approximations here described.

In order to develop a fair comparison between Canny Edges Detector and the one we have described, it should be made taking just one of the three algoritms. For this sake the specification of every algorithm has served as input for the tool, i.e.:

$$Algorithm_1 \equiv C||\{r,u\}A \quad (23)$$

Similarly can be defined the remainder cases

$$Algorithm_2 \equiv C||\{r,l\}B \quad (24)$$
$$Algorithm_3 \equiv C||\{u,l\}D \quad (25)$$

Figures 3-5 show the computational complexity of the three algorithms implementing $BFRSMBTM$

V. Conclusion and Future Work

We have specified two of the best edges detectors by means of the Markovian Process Algebra ROSA. From these specifications ROSA Analyser Tool have generated their corresponding Labelled Transitions Systems so showing a very detailed information about the behaviour of both kind of algorithms. This analysis ends up as a victory of the Binary Fuzzy Rough Set model based on Triangle Modulus over the Canny algorithm, not only as the most accurate for identifying edges, but also as the cheapest algorithm from the computational point of view.

We plan to develop a set of statistical measures to properly estimate the temporal cost of the actions invloved, so being to more accurately computing this advantage. We also plan to go further on the problem of digital image processing analysis, taking a view over the use of wavelets and others transforms.

Acknowledgements

This work has been partially supported by projects TIN2009-14312-C02-02 & CGL2010-20787-C02-02

References

[1] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, nov. 1986.

[2] Wang Dan and Wu Meng-da. Binary fuzzy rough set model based on triangle modulus and its application to image processing. In *2009. ICCI '09. 8th IEEE International Conference on Cognitive Informatics*, pages 249–255, june 2009.

[3] F. L. Pelayo, M. L. Pelayo, and J. G. Guirao. Generating the syntactic and semantics graphs for a markovian process algebra. *Journal of Computational and Applied Mathematics*, 204:38–47, 2007.

[4] Maria L. Pelayo, Fernando L. Pelayo, Fernando Cuartero, Valentin Valero, Gregorio Diaz, and Elena Nieto. Does rosa provide a good view of the memorizing process? In *Proceedings of the 6th IEEE International Conference on Cognitive Informatics*, COGINF '07, pages 273–283, Washington, DC, USA, 2007. IEEE Computer Society.