27

Hardware Implementation of a Real-Time Image Segmentation Circuit based on Fuzzy Logic for Edge Detection Application

Angel Barriga
Instituto de Microelectrónica de Sevilla (CNM/CSIC)/Univ. Sevilla
Spain

1. Introduction

Edge detection algorithms in images make it possible to extract information from the image and reduce the amount of required stored information. An edge is defined as a sharp change in luminosity intensity between two adjacent pixels. Most edge detection techniques can be grouped into two categories: gradient based techniques and Laplacian based methods. Techniques based on gradient use the first derivative of the image and look for the maximum and the minimum of this derivative. Examples of this type of strategies are: the Canny method (Canny, 1986), Sobel method, Roberts method (Roberts, 1965), Prewitt method (Prewitt, 1970), etc. On the other hand the techniques based on Laplacian look for the cross by zero of the second derivative of the image. An example of this type of techniques is the zero-crossing method (Marr & Hildreth, 1980).

Normally edge extraction mechanisms are implemented by executing the corresponding software realisation on a processor. Nevertheless in applications that demand constrained response times (real time applications) the specific hardware implementation is required. The main drawback of implementing edge detection techniques in hardware is the high complexity of the existing algorithms. The process of edge detection in an image consists of a sequence of stages. Image segmentation is one step in the edge detection process. By means of the segmentation the image is divided in parts or objects that constitutes it. In the case of considering only one region the image is divided in object and background. The level at which this subdivision is made depends on the application. The segmentation will finish when all the objects of interest for the application have been detected.

The image segmentation algorithms are based generally on two basic properties of the image grey levels: discontinuity and similarity. Inside the first category the techniques tries to divide the image by means of the sharp changes on the grey level. In the second category there are applied thresholds techniques, growth of regions, and division and fusion techniques.

The simplest segmentation problem appears when the image is formed by only one object that has homogenous light intensity on a background with a different level of luminosity. In this case the image can be segmented in two regions using a technique based on a threshold parameter. Thresholding then becomes a simple but effective tool to separate objects from the background. Most of thresholding algorithms are initially meant for binary thresholding. This binary thresholding procedure may be extended to a multi-level one with the help of

multiple thresholds T_1, T_2,…,T_n to segment the image into n+1 regions (Liao et al., 2001), (Cao et al., 2002), (Oh & Kim, 2006). Multi-level thresholding based on a multi-dimensional histogram resembles the image segmentation algorithms based on pattern clustering.
Binary thresholding techniques classify the pixels of the image into two categories (black and white). This transformation is made to establish a distinction between the objects of the image and the background. This binary image is generated by comparing the values of the pixels with a threshold T. That is to say, any value lower than the threshold value is considered to be an object whereas values greater than the threshold belong to the background.

$$y_{ij} = \begin{cases} 0 & if \quad x_{ij} < T \\ L-1 & if \quad x_{ij} > T \end{cases} \tag{1}$$

where x_{ij} is a pixel of the original image and y_{ij} is the pixel corresponding to the binary image. In the case of a monochrome image in which the pixels are encoded with 8 bits the range of values adopted by the pixels corresponds to the range between 0 and 255 (L=256). It is usual to express the above mentioned range with normalized values between 0 and 1.

2. Thresholding techniques

A basic technique for threshold calculation is based on the frequency of grey level. In this case the threshold T is calculated by means of the following expression:

$$T = \sum_{i=1}^{L} p_i i \tag{2}$$

where i is the grey level, p_i represents the grey level frequency (also known as the probability of the grey level). For an image with n pixels and n_i pixels with the grey level i:

$$p_i = n_i / n \quad and \quad \sum_{i=1}^{L} p_i = 1 \tag{3}$$

Otsu's technique (Otsu, 1978) calculates the optimal threshold maximizing the variance between classes. For that it realizes an exhaustive search to evaluate the criterion of maximizing the variance between classes. One drawback of Otsu's method is the time required to select the value of the threshold.
In the case of two-level thresholding the pixels are classified into two classes: C_1, with gray levels [1, ..., t], and C_2, with gray levels [t+1, ..., L]. The distributions of probability of gray levels for the two classes are:

$$C_1 : \frac{p_1}{w_1(t)}, \ldots\ldots\ldots\ldots, \frac{p_t}{w_1(t)} \tag{4}$$

$$C_2 : \frac{p_{t+1}}{w_2(t)}, \frac{p_{t+2}}{w_2(t)}, \ldots\ldots\ldots\ldots, \frac{p_l}{w_2(t)} \tag{5}$$

where

$$w_1(t) = \sum_{i=1}^{t} p_i \quad \text{and} \quad w_2(t) = \sum_{i=t+1}^{L} p_i \tag{6}$$

The mean values for C_1 and C_2 classes are

$$\mu_1 = \sum_{i=1}^{t} \frac{ip_i}{w_1(t)} \quad \text{and} \quad \mu_2 = \sum_{i=t+1}^{l} \frac{ip_i}{w_2(t)} \tag{7}$$

Let μ_T be the average intensity of whole image, so that:

$$w_1\mu_1 + w_2\mu_2 = \mu_T \quad \text{and} \quad w_1 + w_2 = 1 \tag{8}$$

Using discriminant analysis the variance between classes can be defined as

$$\sigma^2{}_B = w_1(\mu_1 - \mu_T)^2 + w_2(\mu_2 - \mu_T)^2 \tag{9}$$

For a two-level thresholding the optimal threshold t^* is chosen so that $\sigma^2{}_B$ is maximum, ie

$$t^* = \max_t\{\sigma^2{}_B(t)\} \qquad 1 \le t \le L \tag{10}$$

Otsu's method can be easily applied to multiple thresholds. Assuming there are M-1 thresholds $\{t_1, t_2, ..., t_{M-1}\}$, which divide the image into M classes:

$$C_1 \text{ for } [1,...,t_1],\ C_2 \text{ for } [t_1+1,...,t_2],\ \ ...\ \ ,C_i \text{ for } [t_{i-1}+1,...,t_i],\ \ ...\ \text{ and } C_M \text{ for } [t_{M-1},...,L], \tag{11}$$

The optimal thresholds $t_1^*, t_2^*, ..., t_{M-1}^*$ are chosen to maximize $\sigma^2{}_B$:

$$\{t_1^*, t_2^*, ..., t_{M-1}^*\} = \max_{t_1, t_2, ..., t_{M-1}} \{\sigma^2{}_B(t_1, t_2, ..., t_{M-1})\} \qquad 1 \le t_1 < .... < t_{M-1} < L \tag{12}$$

$$\text{where } \sigma_B{}^2 = \sum_{k=1}^{M} w_k(\mu_k - \mu_T)^2 \tag{13}$$

$$\text{with } w_k = \sum_{i=C_k} p_i \text{ and } \mu_k = \sum_{i=C_k} \frac{ip_i}{w_k} \tag{14}$$

ω_k is known as zero-order cumulative moment of the k-th class C_k, and the numerator of the last expression is known as first-order cumulative moment of the k-th class C_k, ie

$$\mu(k) = \sum_{i=C_k} ip_i \tag{15}$$

Regardless of the number of classes that are considered during the thresholding process the sum of the cumulative probability functions of the M classes are equal to 1 and the mean of the image is equal to the sum of the means of the M classes weighted by their corresponding cumulative probabilities, ie

$$\sum_{k=1}^{M} w_k = 1 \quad \text{and} \quad \mu_T = \sum_{k=1}^{M} w_k\mu_k \tag{16}$$

Using (16) the variance between classes in equation (13) can be rewritten as follows

$$\sigma_B{}^2(t_1, t_2, ..., t_{M-1}) = \sum_{k=1}^{M} w_k \mu^2{}_k - \mu_T{}^2 \tag{17}$$

Since the second term in equation (17) depends on the choice of thresholds $\{t_1, t_2, ..., t_{M-1}\}$, the optimal thresholds $\{t_1^*, t_2^*, ..., t_{M-1}^*\}$ can be chosen maximizing a modified variance between classes $(\sigma_B')^2$, defined as the sum of the terms of the right side of equation (17). That is, the optimal threshold values $\{t_1^*, t_2^*, ..., t_{M-1}^*\}$ are chosen by

$$\{t_1^*, t_2^*, ..., t_{M-1}^*\} = \max_{t_1, t_2, ..., t_{M-1}} \{\sigma^2{}_B(t_1, t_2, t_{M-1})\} \tag{18}$$

$$\text{where } (\sigma_B^{'})^2 = \sum_{k=1}^{M} w_k \mu_k{}^2 \tag{19}$$

According to the criterion of expression (12) for σ_B^2 and equation (18) for $(\sigma_B')^2$, in order to find optimal thresholds, the search region for the maximum σ_B^2 and for the maximum $(\sigma_B')^2$ is $1<t_1<L-M+1$, $t_1+1<t_2<L-M+2$, …, $t_{M-1}+1< t_{M-1}<L-1$.

This exhaustive search involves $(L-M+1)^{M-1}$ possible combinations. Furthermore, equation (19) is simpler than (13) because it don't requires the subtractions.

In 1965 Zadeh proposed fuzzy logic as a reasoning mechanism that uses linguistic terms (Zadeh, 1965). Fuzzy logic is based on the fuzzy set theory in which an element can belong to several sets with different degrees of membership. This contrasts with the classic set theory in which an element either belongs or does not belong to a certain set. Thus a fuzzy set A is defined as

$$A = \{(x, \mu(x)) | x \in X\} \tag{20}$$

where x is an object of the set of objects X and $\mu(x)$ is the membership degree of element x to set A. In the classic set theory $\mu(x)$ takes values 0 or 1 whereas in the fuzzy set theory $\mu(x)$ belongs to the range of values between 0 and 1.

Techniques that apply fuzzy logic to threshold calculation are based mainly on three types of measures of fuzziness (Forero-Vargas & Rojas-Camacho, 2000): entropy, Kaufmann`s measure, and Yager's measure.

The technique based on entropy consists of minimizing the dispersion of the system. This way the pixels of the image are grouped into two classes corresponding to the objects and to the background. Huang and Wang (Huang & Wang, 1995) consider that the averages of the data corresponding to each class are μ_0 and μ_1. The membership function of each class is defined as:

$$u_x(x) = \begin{cases} \dfrac{1}{1 + \dfrac{|x - \mu_0|}{x_{\max} - x_{\min}}} & if \quad x < T \\ \dfrac{1}{1 + \dfrac{|x - \mu_1|}{x_{\max} - x_{\min}}} & if \quad x > T \end{cases} \tag{21}$$

The calculation of the threshold T is based on the entropy of a fuzzy set that is calculated using the function of Shannon:

$$H_f(x) = -x\log x - (1-x)\log(1-x) \tag{22}$$

The threshold will be that which minimizes the entropy of the data:

$$E(T) = \frac{1}{M}\sum_i H_f(\mu_x(i))h(i) \tag{23}$$

Kaufmann's measure of fuzziness is defined as (Kaufmann, 1975):

$$D(A) = \left[\sum_{x\in X}\left|\mu_A(x) - \mu_C(x)\right|^w\right]^{\frac{1}{w}} \tag{24}$$

This method is based on using the distance metric to set *A*. When *w*=1 Hamming's distance is used whereas if *w*=2 it is the Euclidean distance.
Yager's method (Yager, 1979) is based on the distance between a fuzzy set and its complementary, and basically entails minimizing the following function:

$$D_2(T) = \sqrt{\sum_i\left|\mu_x(i) - \mu_{\bar{x}}(i)\right|^2} \tag{25}$$

where $\mu_{\bar{x}}(i) = 1 - \mu_x(i)$.

(Barriga & Hussein, 2008) proposed a technique that, from a formal point of view, is based on calculating the average of the histogram of the image. One advantage of this technique is that the calculation mechanism improves the processing time since the image only needs to be processed once and the value of the threshold can be calculated directly. From the point of view of hardware implementation that enables low-cost circuit for fuzzy processing module as discussed in a later section
The fuzzy system receives the input pixel and generates an output that corresponds to the result of the fuzzy inference. Once the image has been read the output shows the value of threshold *T*. Basically the operation carried out by the fuzzy system is that of calculating the centre of gravity of the image histogram with the following expression:

$$T = \sum_{i=1}^{M}\sum_{j=1}^{R}\alpha_{ij}c_{ij} \Big/ \sum_{i=1}^{M}\sum_{j=1}^{R}\alpha_{ij} \tag{26}$$

where *T* is the threshold, *M* is the number of pixels of the image, *R* is the number of rules of the fuzzy system, *c* is the consequent of each rule and α is the activation degree of the rule.
In order to produce the fuzzy inference the universe of discourse of the histogram is divided into a set of *N* equally distributed membership functions. Figure 1 shows a partition example for *N*=9. Triangular membership functions have been used since they are easier for hardware implementation. These functions have an overlapping degree of 2 in order to limit the number of active rules. The membership functions of the consequent are singletons equally distributed in the universe of discourse of the histogram. The use of singleton-type membership functions for the consequent allows the application of simplified defuzzification methods such as the Fuzzy Mean. This defuzzification method can be interpreted as one in which each rule proposes a conclusion with a "strength" defined by its grade of activation. The overall action of several rules is obtained by calculating the average of the different conclusions weighted by their grades of activation. This type of processing,

based on active rules and a simplified defuzzification method, allows low cost and high speed hardware implementation.

a) b)

Fig. 1. Membership functions for *N*=9, a) antecedent, b) consequent.

The rule base of the system in figure 2 use the membership functions defined in figure 1. The knowledge base (membership functions and rule base) is common for any images, and the values can therefore be stored in a ROM memory.

if x is L1 then c is C1;
if x is L2 then c is C2;
if x is L3 then c is C3;
if x is L4 then c is C4;
if x is L5 then c is C5;
if x is L6 then c is C6;
if x is L7 then c is C7;
if x is L8 then c is C8;
if x is L9 then c is C9;

Fig. 2. Rulebase for *N*=9.

It is possible to optimize the expression shown in equation (26) if the system is normalized. In this case the sum extending to the rule base of the grades of activation of the consequent takes value 1:

$$\sum_{i=1}^{R} \alpha_{ij} = 1 \tag{27}$$

Then (26) transforms in:

$$T = \frac{1}{M} \sum_{i=1}^{M} \sum_{j=1}^{R} \alpha_{ij} c_{ij} \tag{28}$$

For each pixel the system makes the inference in agreement with the rule base of figure 2. The output of the system accumulates the result corresponding to the numerator of (28). The final output is generated with the last pixel of the image after division by *M*.

3. Image segmentation

The technique presented in (Barriga & Hussein, 2008) has the disadvantage that the rule base is predetermined and therefore the threshold does not fit to the characteristics of the image. It is a linear approximation. A mechanism to adjust the threshold to the

characteristics of the image is to perform a nonlinear approximation. Figure 3 shows some examples of knowledge bases that give place to non-linear approximations. The figure shows five fuzzy systems (figure 3a to figure 3e). For each system there have been represented the membership functions for antecedents, the output function and the result of segmentation using the threshold generated in each case. In all cases the membership functions of antecedents constitute a family of functions. This family consists of triangular functions with an overlapping degree of two. This structure is determined by the hardware implementation requirements of the system as we will discuss in a later section. It may be noted that the base and the position of the membership functions change from one system to another giving rise to a nonlinear behavior.
This approach allows to obtain thresholds adapted to the characteristics of the image or the requirements of the application. Table 1 shows the thresholds obtained in different images using the Otsu method, the grey level frequency method and usign the fuzzy systems of figures 3a to 3e.

4. Hardware implementation

4.1 Architecture description

The design goals of the fuzzy inference module (FIM) for calculating the threshold are: a low cost system and high processing speed. The architecture of the FIM circuit is based on the proposal described in (Baturone et al., 2000) shown in Figure 4. The module consists of three stages: fuzzifier, inference and defuzzifier. The inference mechanism is based on active rules. This allows to process only those rules that are active and avoids to analyze the whole rulebase. This way the processing time is reduced. For it the overlapping degree of the membership functions is limited. Another architecture feature is the use of singleton consequents. This allows to apply simplified defuzzification methods which supposes a reduction of hardware resources.
The first stage of the architecture corresponds to the fuzzificación stage. This stage receives the input data and generates for each input the pair (Label, membership degree) = (L, μ). MFC blocks (Membership Function Circuit) perform this task. There are several alternatives to the design of MFC blocks (Baturone et al., 2000). One solution is to design the block as an arithmetic circuit that interpolates the right output for each input. This solution gives place to a simple and fast circuit. However it has as counterpart that limits the type of membership functions to triangular and trapezoidal functions. A more flexible solution is based on the use of a memory. In this case the input acts as a pointer to a memory location. This memory location stores the output values. This allows to have membership functions of any form. The shape of the membership function has no restrictions other than the selected precision and has no influence on the computational load. As opposed to this advantage, in situations of high resolution, memory requirements can become very large since the number of rows in the antecedents memory depends exponentially on the number of bits of the input. In the case of N membership functions, with P bits of precision for the input, and J bits of precision for the membership degree, the size of the required memory is given by the equation (29).

$$T = N \cdot J \cdot 2^{P} \tag{29}$$

Since the overlapping degree of the membership functions is fixed, the number of output values of the fuzzification stage is limited. For example, in the case of limiting the

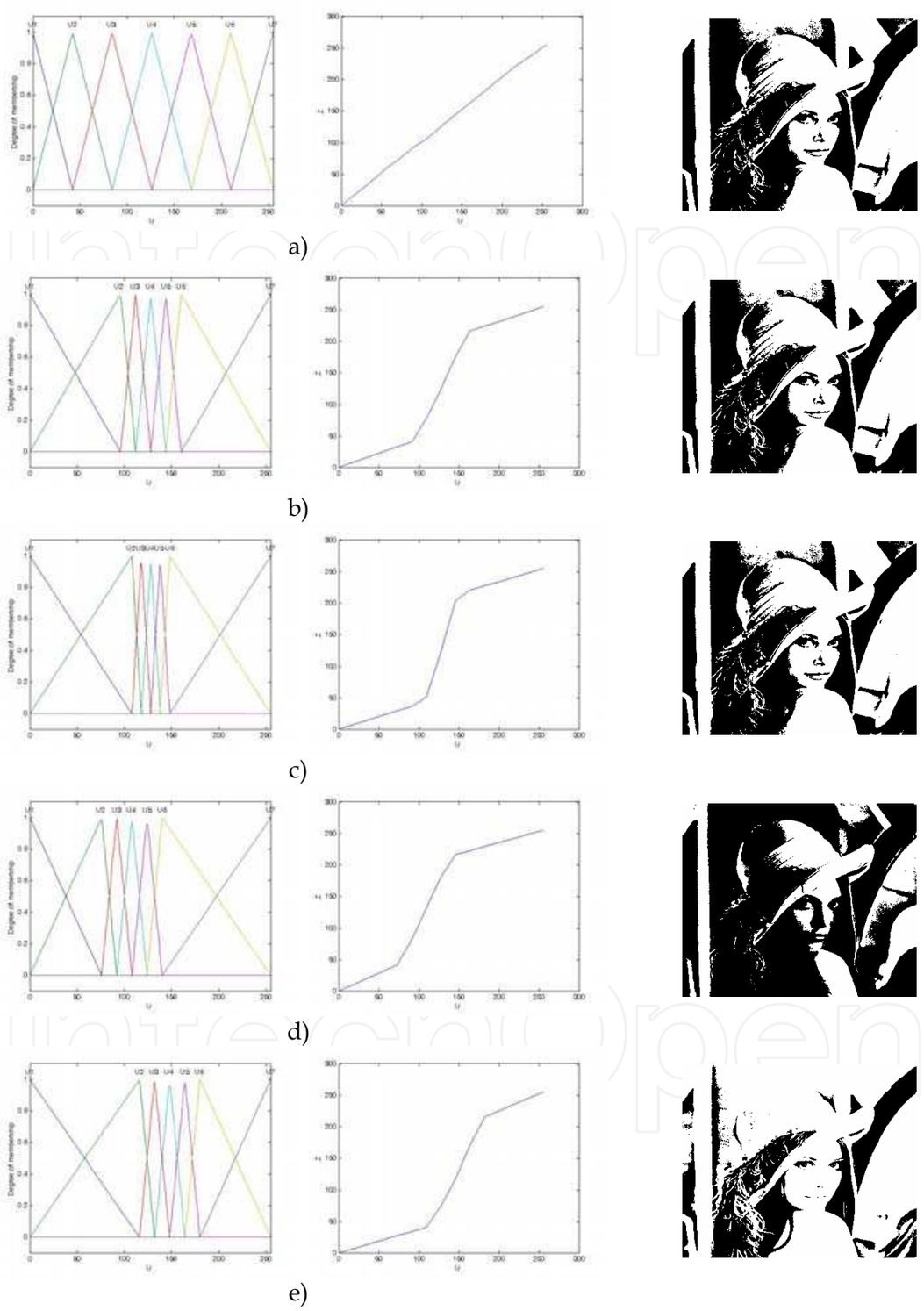


Fig. 3. Examples of fuzzy systems for image thresholding. For each subfigure there are: i) Membership functions for antecedent. ii) Output of the fuzzy system. iii) image segmentation sample.

	Otsu	Freq	Fuzz-a	Fuzz-b	Fuzz-c	Fuzz-d	Fuzz-e
Lena	116	123	126	124	126	152	96
Barbara	112	112	114	105	105	134	78
Cameraman	87	118	120	135	138	154	106
Peppers	101	104	106	103	105	123	80

Table 1. Thresholds obtained using different methods on sample images

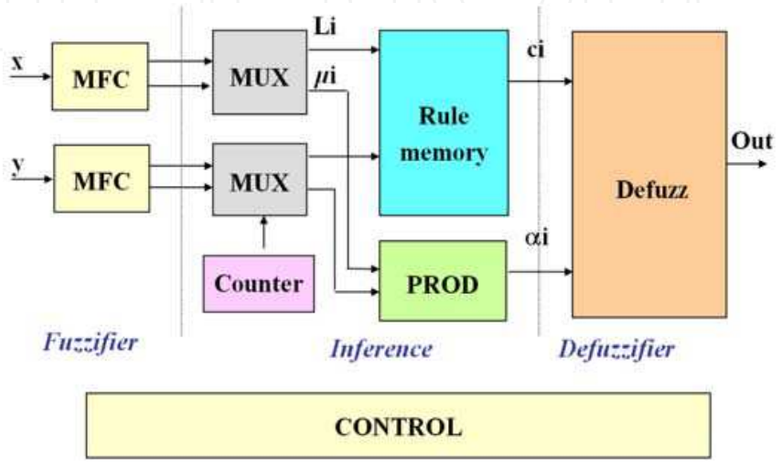


Fig. 4. Architecture of fuzzy inference module (FIM).

overlapping degree of the membership functions to 2, and in the case of a system of 2 inputs, only 4 couples of values (Label, degree) exist, i.e. only 4 rules are activated. Therefore the inference stage is constituted by the block that selects each one of the antecedents of the active rules. A set of multiplexers controlled by a counter allows to select sequentially the different combinations of antecedents of the active rules. In each counter cycle the membership degrees are processed through the conjunction operator to calculate the rule activation degree, while the labels of the antecedents address the memory position that contains its corresponding consequent. The output of the inference stage corresponds to the pair of values (Consequent, activation degree) = (c, α). for each rule.

The last stage performs the defuzzification. On having used singleton consequents, the defuzzification algorithm only requires operations on the rules. The hardware resources required for implementing the Fuzzy Mean defuzzification method are: a multiplier, two accumulators and a divisor. This defuzzification method corresponds to the following operation:

$$Out = \sum_r \alpha_i c_i \; / \sum_r \alpha_i \tag{30}$$

where the summations are extended to active rules, c_i is the consequent of each rule and α_i is the rule activation degree.

In the case of having normalized membership functions and applying the product as T-norm the denominator of the previous equation is 1. This means that a divisor is not needed and defuzzification operation is simplified according to the following expression:

$$Out = \sum_{r} \alpha_i c_i \tag{31}$$

4.2 Design and implementation

From the general characteristics of the FIM architecture it is possible to specify a set of simplification options that allow a reduction of hardware resources and increased parallelism (and thus the processing speed). Regarding the design of the different blocks of figure 4 and according to the knowledge base of the threshold system the memory requirements are: a) the MFC memory requires 256x10 bits; b) the rule memory requires 7x8 bits.

Figure 5 shows the system architecture. The FIM module receives input x corresponding to one pixel. MFC memory stores the data of the antecedent membership functions according to the scheme shown in figure 6a. Since the overlapping degree is fixed to 2, each row of memory only stores the value of a linguistic label and a membership degree (Label, degree)=(L, μ). The other label can be calculated increasing in a unit the stored value, since always the linguistic labels of both membership functions that are active are consecutive (L_2=L_1+*1*). While the other membership degree is calculated taking into account that the membership functions are normalized, by the operation μ_2=*1*-μ_1.

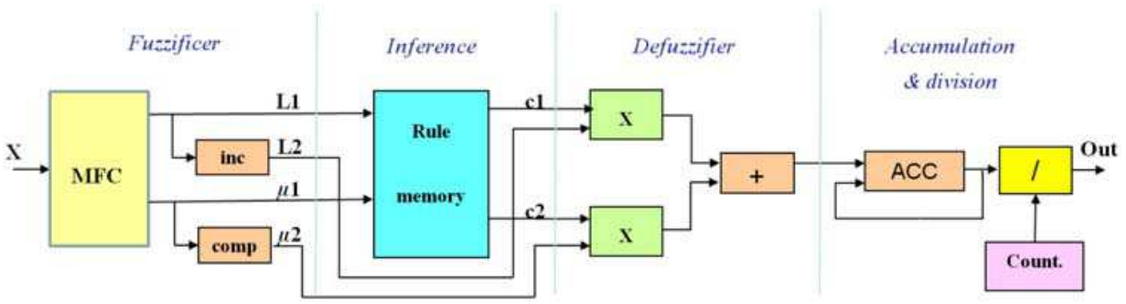


Fig. 5. FIM circuit for calculating the threshold

The rule memory is a dual-port memory. This way it is possible to access simultaneously to two active rules. This memory is addressed by the linguistic label that provides the MFC. This allows to eliminate the multiplexers and the counter of figure 4.

The defuzzification stage receives both the consequents (c_1 and c_2) and the activation degrees of the active rules (μ_1 and μ_2). The last stage makes the accumulation of the result generated by each pixel and the division by the number of pixels of the image. In agreement with the described FIM scheme it is possible to make an inference in each clock cycle.

In order to increase the operation speed of the system it is possible to process two pixels in parallel as shown in Figure 7. For it the blocks of higher cost (the MFC memory and the divisor) are shared by both inputs. The MFC memory is a dual-port memory. This allows to reduce by the half the time required to calculate the threshold.

The circuit of figure 7 has been implemented on a low cost FPGA Spartan3 device XC3S200 of Xilinx. The results of the required hardware resources on the Spartan3 FPGA circuit are shown in Table 2. The table shows the resources needed in the case of the circuit with and without the divisor. This division block is that of major cost of the system.

The circuit implemented on the Spartan3 FPGA operates at a frequency of 50MHz. In each clock cycle it allows to process two pixels. Thus the processing time of an SVGA image of

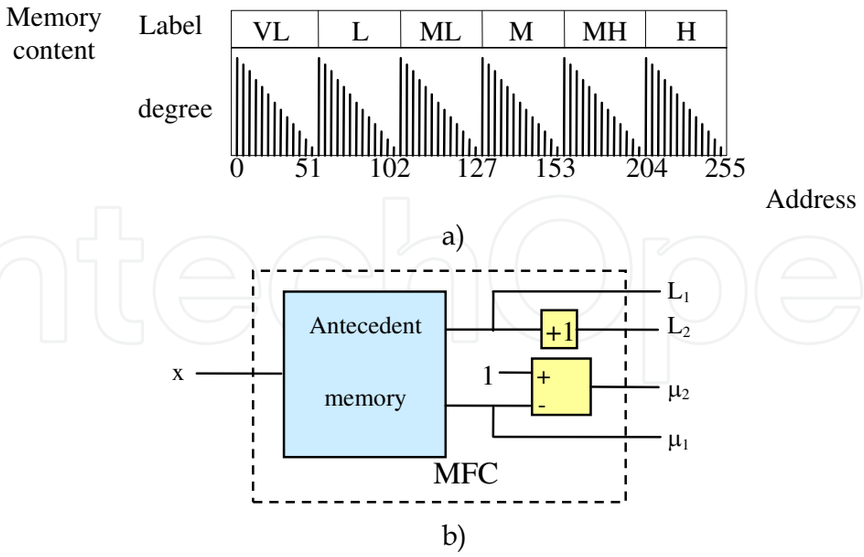


Fig. 6. a) Storage scheme in the antecedent memory. b) MFC circuit based on memory

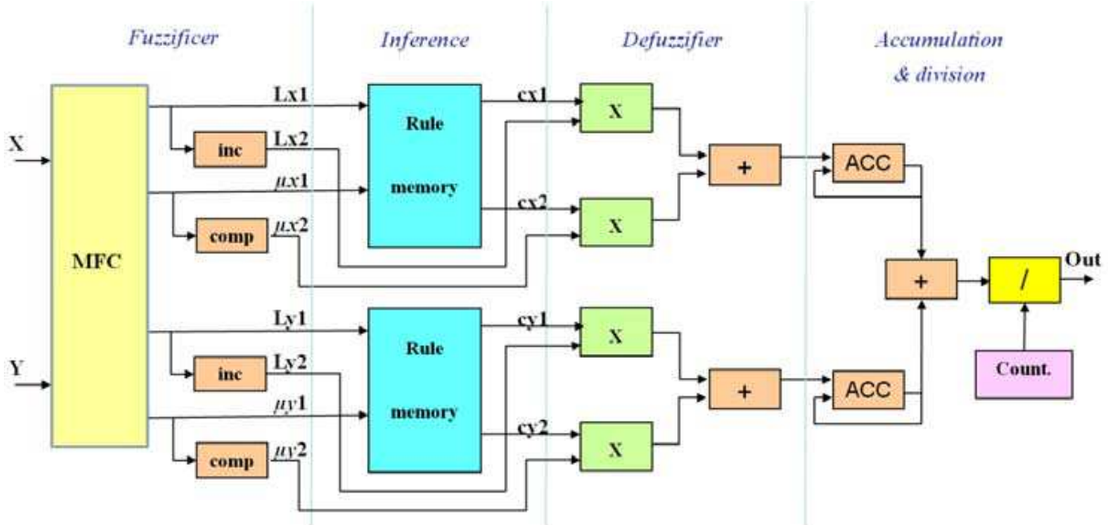


Fig. 7. Circuit that allows to process two pixels in parallel

Resources	without DIV	with DIV
slices	82	407
8x8 bit multiplier	4	4
Flip-flops	84	654
256x10 bit dual-port RAM	1	1
7x8 bit dual-port RAM	2	2

Table 2. Hardware resources on XC3S200 FPGA

800x600 pixels is 4.8 msec. This allows to make a processing of 208 frames per second. In the case of an HD image (1920x1080 pixels) it is possible to process 48 frames per second.

5. Edge detection

This section presents an application of image segmentation to edge detection. The method is applied to the luminosity of the image. An image is a bidimensional matrix of pixels whose values belong to certain range of values. In this section each pixel is codified with 8 bits, which gives rise to 256 possible values of grey tones. An image is therefore a function of two variables (dimensions) in the range from 0 to 255.
The process of edge detection in an image consists of the sequence of stages shown in figure 8. The first stage receives the input image and applies a filter to eliminate noise. The second step applies a threshold in order to classify the pixels of the image under two categories, black and white. The resulting image is a binary image. Finally, in the last stage the edges are detected.

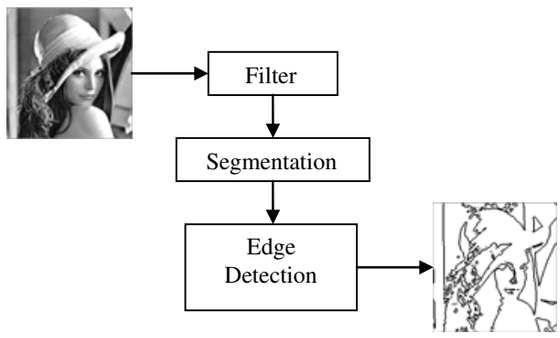


Fig. 8. Diagram flow for edge detection.

5.1 The filter stage

The filter stage makes it possible to improve details of edges in images and reduce or eliminate noise patterns. The aim of the filter step is to eliminate all those points that do not provide any type of information of interest. The noise corresponds to undesired information appearing in the image. It comes principally from the capture sensor (quantisation noise) and from the transmission of the image (fault in transmitting the information bits). Basically we consider two types of noise: Gaussian and impulsive (salt&peppers). Gaussian noise has its origin in differences of gains in the sensor, noise in digitalization, etc. Impulsive noise is characterized by arbitrary pixel values that are detectable because they are very different from their neighbours. A way to eliminate these types of noise is by means of a low pass filter, a filter which smoothens out the image replacing high and low values by average values.
The filter used in the proposed edge detection system is based on the bounded sum Lukasiewicz operator which is defined as:

$$BoundedSum(x,y) = \min(1, x+y) \tag{32}$$

The behaviour of the bounded-sum is shown in figure 9. It consists of a normalized function in the [0,1] range. An advantage of applying this operator lies in the simplicity of the hardware realisation.

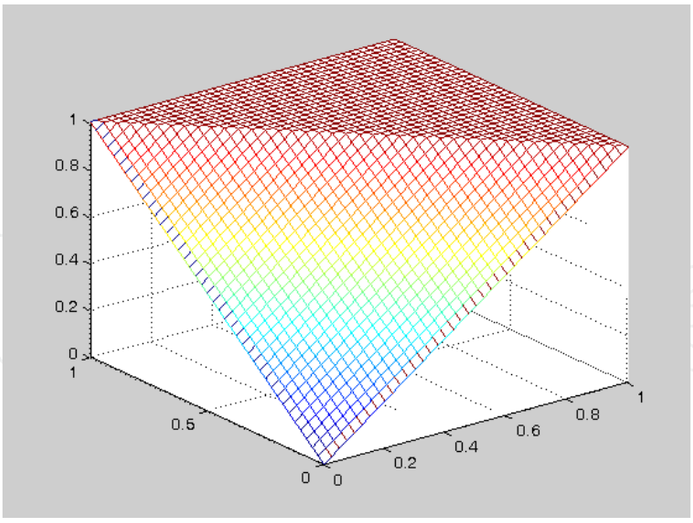
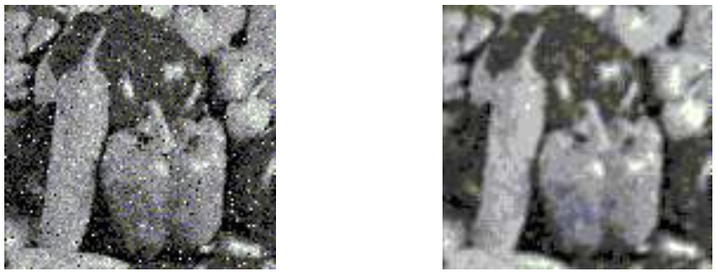


Fig. 9. Bounded sum graphical representation.

The Lukasiewicz bounded sum filter smoothens out the image and is suitable for both salt&peppers and Gaussian noise. Figure 10 shows the effect of applying this type of filter.



a) b)

Fig. 10. a) Input image with salt&peppers noise, b) Lukasiewicz's bounded sum filter output.

The filter has been applied using a mask based on a 3x3 array. For pixel x_{ij} the weighted mask is applied to obtain the new value y_{ij}, as is shown in the following expression:

$$y_{ij} = \min(1, \frac{1}{8} \sum_{k=-1}^{1} \sum_{l=-1}^{1} x_{i+k,j+l}) \tag{33}$$

5.2 The segmentation stage

Techniques based on thresholding an image allow pixels to de divided into two categories (black and white). This transformation is made to establish a distinction between the objects of the image and the background. This binary image is generated by comparing the values of the pixels with a threshold T. That is to say, any value lower than the threshold value is considered to be an object whereas values greater than the threshold belong to the background. In this stage there is applied the previously calculated threshold T in order to obtain the binary image.

5.3 Edge detection stage

The next step is the edge detection. The input image for the edge detection is a binary image in which pixels take value 0 (black) or 1 (white). In this case the edges appear when a change between black and white takes place between two consecutive pixels.

$$x_{edge} = \begin{cases} 0 & if \;\; x - y \neq 0 \\ 1 & if \;\; x - y = 0 \end{cases} \tag{34}$$

where x and y are consecutive pixels, and x_{edge} is the resulting pixel.
Edge generation consists of determining if each pixel has neighbours with different values. Since the image is binary every pixel is encoded with a bit (black=0 and white=1). This edge detection operation is obtained by calculating the *xor* logic operation between neighbouring pixels using a 3x3 mask. Figure 11 shows an example of applying the *xor* operator on the binary image.

a) b) c)

Fig. 11. a) Lena`s image, b) binary image, c) edge detection.

Using the 3x3 mask it is possible to refine the edge generation by detecting the orientation of the edges. To this end the four orientations shown in figure 12 can be considered. This enables calculation of the *xor* operation on 3 pixels. For a horizontal orientation we will therefore have

$$y_{i,j} = x_{i,j-1} \oplus x_{i,j} \oplus x_{i,j+1} \tag{35}$$

Whereas for an orientation of 45º it will be

$$y_{i,j} = x_{i+1,j-1} \oplus x_{i,j} \oplus x_{i-1,j+1} \tag{36}$$

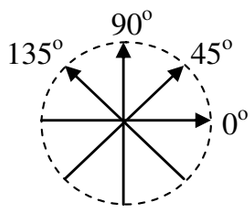


Fig. 12. Orientations for the edges generation

Figure 13 shows the results obtained when edge detection was carried out on a set of test images.

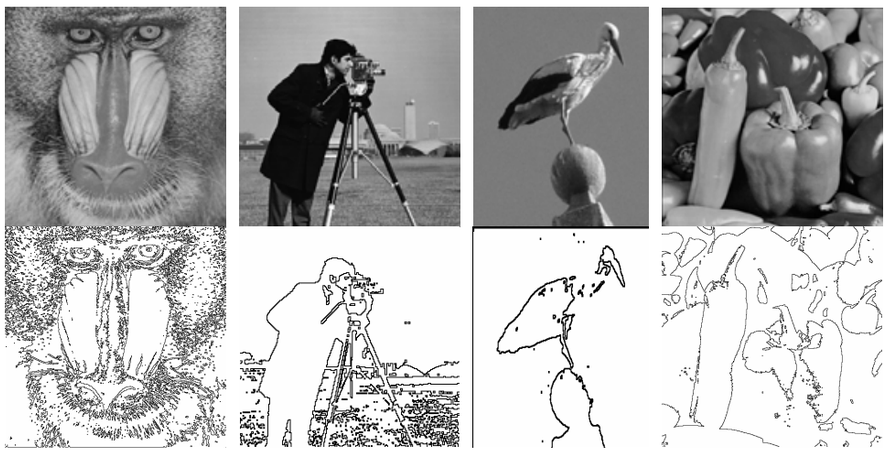


Fig. 13. Test images and edge detection results.

5.4 Hardware implementation

The edge detection circuit has been implemented on a low cost FPGA device of the Xilinx Spartan3 family. Figure 14 shows the block diagram for the system. The image is stored in a double port RAM memory. The data memory width is 32 bits. This makes it possible to read two words simultaneously.

In the first phase there is realized the calculation of the value of the threshold *T*. Later the edge detection circuit initiates its operation reading eight pixels from the memory in each clock cycle (2 words of 32 bits). The edge detection circuit is thus able to provide four parallel output data which are stored in the external memory. Each data corresponds to a pixel of the edge image. This image is binary, and only one bit is therefore needed to represent the value of the pixel (0 if edge or 1 if background). The new image of the edges is stored in the above mentioned memory.

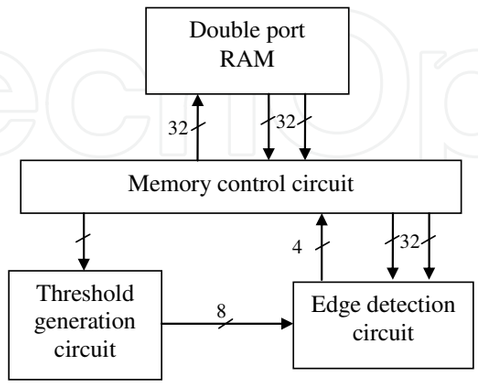


Fig. 14. Block diagram of the system.

The edge detection algorithm basically comprises three stages as shown in figure 8 (Hussein & Barriga, 2008, 2009). In the first stage the Lukasiewicz bounded-sum is performed. After

the filter stage a thresholding step is applied producing a black and white monochrome image. The value of the threshold is obtained by means of a fuzzy system that calculates the threshold related to the image.
In the third stage the edges of the image are obtained. For it the final value of each pixel is evaluated. Only those pixels that are around the target pixel are of interest (a 3x3 mask). Therefore if in the surroundings of a pixel the value is the same (all white or all black) this indicates no edge and the output value associates the above mentioned pixel with the background of the image. If a change is detected in any value of the surroundings of the pixel this indicates that the pixel at issue is in an edge, and it is therefore assigned the black value.
Figure 15 shows the system processing scheme. Pixels 1 to 9 correspond to the 3x3 mask that moves through the image. The Functional Unit (FU) processes the data stored in the mask registers.

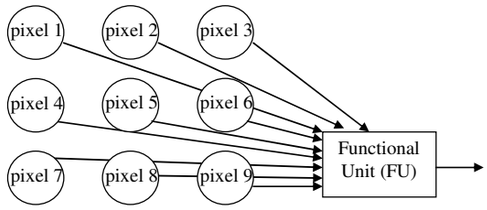
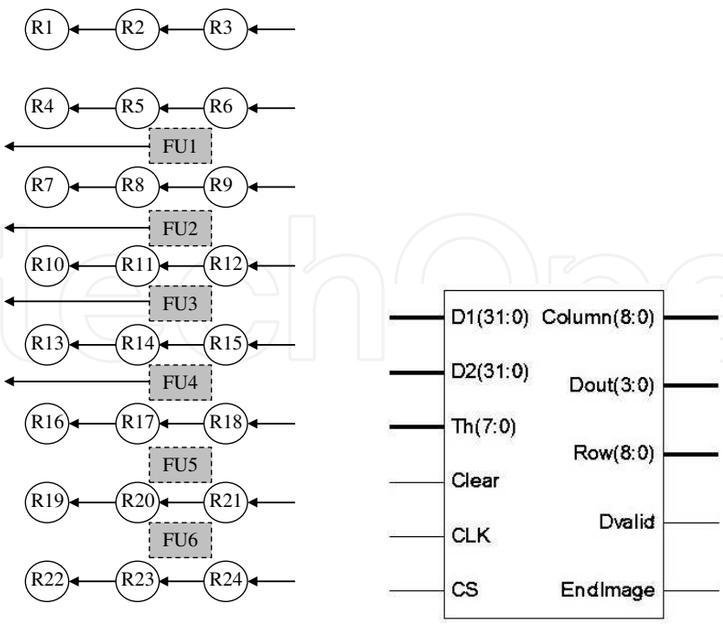


Fig. 15. System schema.

To improve image processing time the mask was spread to an 8x3 matrix as shown in figure 16a. Each Functional Unit (FU) operates on a 3x3 mask in agreement with the scheme shown in figure 15. The data are stored in the input registers (R3, R6, R9, …) and in each clock cycle they move to their interconnected neighbours registers. In the third clock cycle the mask registers contain the data of the corresponding image pixels. The functional units then operate with the mask data and generate the outputs. In each clock cycle the mask advances one column in the image. Pixels enter on the right and shift from one stage to another outgoing on the left hand side. It is a systolic architecture with linear topology and it allows several pixels to be processed in parallel.
Figure 16b shows the input/output ports in the symbol of the system. The system receives two input data of 32 bits (*D1* and *D2*). These data come from a double port memory that stores the image. The memory access time makes it possible to read 8 pixels (each of 8 bits) in a clock cycle. The circuit also receives the previously calculated threshold (*T*) as input data. The input control signals are the following: the clock (*CLK*), the synchronous clear (*Clear*), and chip select signal (*CS*). The circuit generates as output the 4 bits (*Dout*) corresponding to the output values of the processed pixels stored in R5, R8, R11 and R14. The address of the pixel stored in R5 is also generated by means of the buses *Row* and *Column*. The output control signals *Dvalid* and *EndImage* respectively indicate the validity of the outputs and the completion of the image processing.
The functional unit operates on the 3x3 mask and generates the output value corresponding to the centered element of the mask (pixel 5 in figure 15). A block diagram of a functional unit is shown in figure 17. The circuit consists of two pipeline stages so that the data has a latency of two clock cycles. The first stage is the image filter. Then threshold *T* is applied. The edge detector, in the output stage, operates on the binary mask (black and white image).



a) b)

Fig. 16. a) 8x3 architecture, b) symbol of the system.

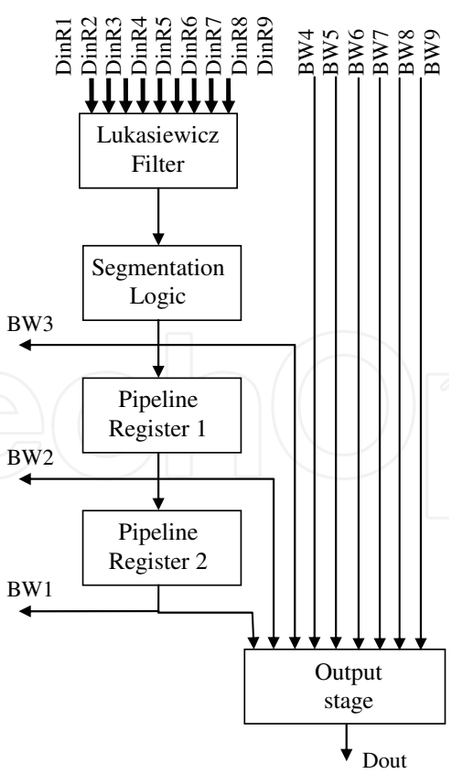
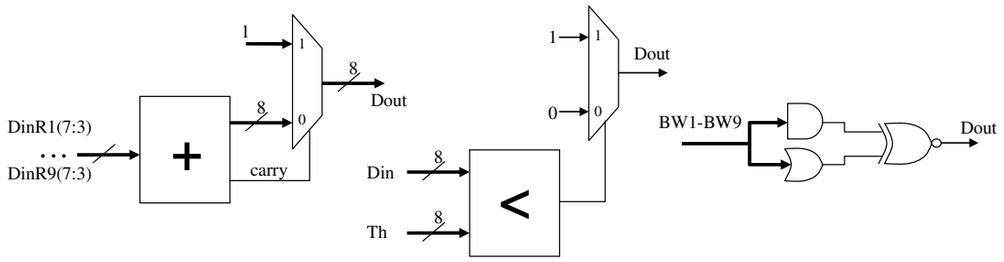


Fig. 17. Functional Unit (FU) circuit schematic.

Figure 18 shows the circuits corresponding to the different blocks of the functional unit (FU). As we can observe in figure 18a the filter based on Lukasiewicz's bounded sum receives the data stored in registers R1 to R9. These data are scaled by the factor 0,125 entailing division by 8, which signify a displacement of three places to the left. The sum of the data is compared (using the carry as control signal) with value 1. The segmentation circuit (figure 18b) compares the pixel with the threshold value. The output is a binary image (black and white) and only therefore requires one bit. Finally, the output stage receives a 3x3 binary image. It carries out the *xor* operation of the bits. If all the bits of the mask are equal the output pixel is in the background, whereas if some bit is different the output is an edge pixel.
The state machine that controls the system is shown in figure 19. This machine has four states. The mask moves through the image by columns. Whenever a row begins two clock



a) b) c)

Fig. 18. Lukasiewicz filter, b) Segmentation circuit, c) output stage.

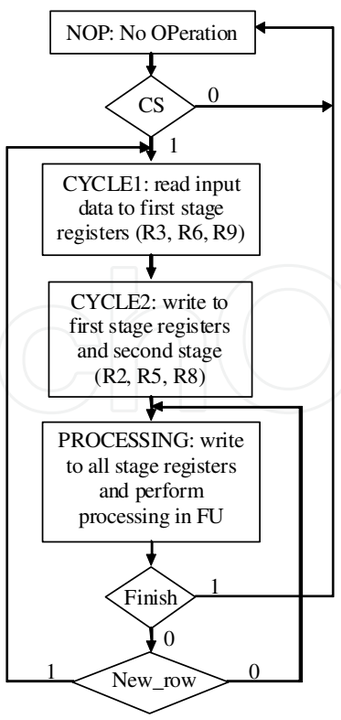


Fig. 19. FSM of the control unit of the edge detection system

cycles are needed to initialize the mask registers (CYCLE1 and CYCLE2 states). In the next cycle (PROCESSING state) the data is processed and the data of the following columns being processed in successive cycles.

Figure 20 shows the chronogram of the circuit. It can be observed that the operation of the system begins with the falling edge of signal *CS*. In the third clock cycle *Dvalid* signal take value 1, indicating a valid output. Input data are provided in each clock cycle. Once *Dvalid* has been activated the output data in the following cycles is also valid (since *Dvalid*=1).

The system has been implemented on an FPGA of the Spartan3 Xilinx family. The circuit for edge detection occupies an area of 318 slices. The resources needed for the full system (which includes the thresholding circuit and the edge detection circuit) occupies 735 slides which mean a 38% of the selected FPGA device. Regarding processing speed, the system required 7.2 msec to generate the edge image of a SVGA (800x600 pixels) using a 50 MHz clock cycle. This mean it is possible to process 132 frames per second. For a HD image (1920x1080 pixels) it is possible to process 32 frames per second.

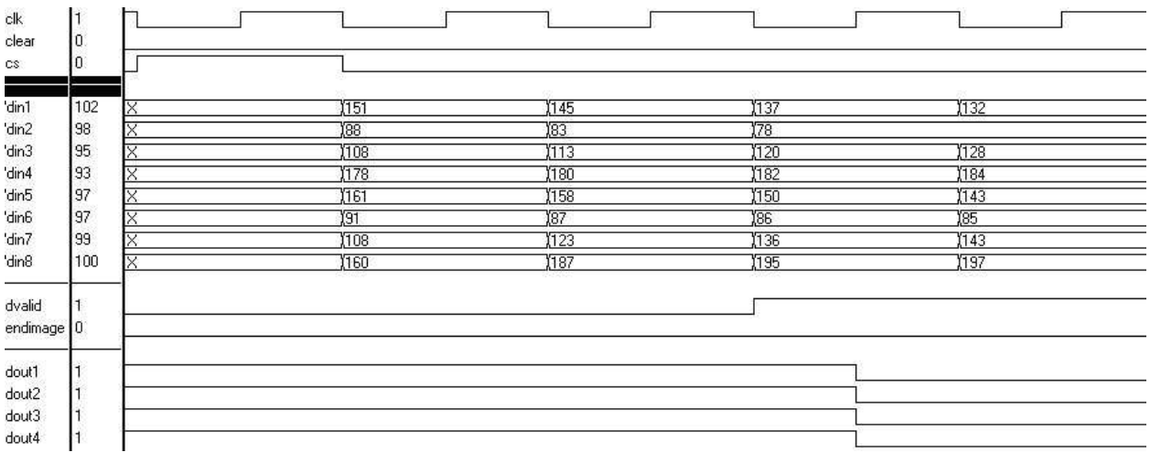


Fig. 20. Chronogram of the edge detection circuit

6. Acknowledgments

This work was supported in part by the European Community under the MOBY-DIC Project FP7-IST-248858 (www.mobydic-project.eu), by Spanish Ministerio de Ciencia y Tecnología under the Project TEC2008-04920, and by Junta de Andalucía under the Project P08-TIC-03674.

7. Conclusion

In this chapter there has been described a mechanism for binary image segmentation based on the application of fuzzy logic to calculate the threshold. The described thresholding method allows to adjust the threshold value to the characteristics of the image. The main advantage of this technique is that it allows very efficient hardware implementation in terms of cost and speed. This makes it especially suitable for applications which require real time processing. This technique has been applied for edge detection in images. The designed circuit has been implemented on an FPGA device.

8. References

Barriga, A. & Hussein, N.M. (2008). A Fuzzy Thresholding Circuit for Image Segmentation. International Conference on Knowledge-Based and Intelligent Information & Engineering Systems.

Baturone, I.; Barriga, A.; Sánchez-Solano, S.; Jiménez, C. J. & López, D. R. (2000). Microelectronic Design of Fuzzy Logic-Based Systems, CRC Press.

Canny, J. F. (1986). A computational approach to edge detection. IEEE Transaction of Pattern Analysis and Machine Intelligence, 679-698.

Cao, L.; Shi, Z.K. & Chenp E.K.W. (2002). Fast automatic multilevel thresholding method. In: Electronics Letters, 38 (16), 868-870.

Forero-Vargas, M.G. & Rojas-Camacho, O. (2000). New formulation in image thresholding using fuzzy logic. Portuguese Conference on Pattern Recognition, 117-124.

Huang, L.K. & Wang, M.J. (1995). Image thresholding by minimizing the measure of fuzziness. Pattern Recognition, 28, 41-51.

Hussein, N.M. & Barriga, A. (2008). Hardware Implementation of a Soft Computing Technique for Edge Detection. International Conference of Signal and Image Engineering (ICSIE 08), London (UK).

Hussein, N.M. & Barriga, A. (2009). High Speed Soft Computing based Circuit for Edges Detection in Images. In: Advances in Electrical Engineering and Computational Science. Gelman, L.; Balkan, N. & Ao, S.I., (Ed.). Springer.

Kaufmann, A. (1975). Introduction to the theory of fuzzy subsets. Academic Press.

Liao, P.S.; Chen, T.S. & Chung P.C. (2001). A Fast Algorithm for Multilevel Thresholding. In: Journal of Information Science and Engineering, 17, 713-727.

Marr, D. & Hildreth, E. (1980). Theory of Edge Detection. Proceedings of the Royal Society, London. 207, 187-217.

Oh, J.T. & Kim, W.H. (2006). EWFCM Algorithm and Region-Based Multi-level Thresholding, In: Lecture Notes in Artificial Intelligence, 4223, 864-873.

Otsu, N. (1978). A Threshold Selection Method from Gray Level Histogram. IEEE Trans. on Systems, Man and Cybernetics, 8, 62-66.

Prewitt, J.M.S. (1970). Object enhancement and extraction. In: Picture Processing and Psychophysics, A. Rosenfeld and B. S. Lipkin, (Ed.), Academic Press, New York, 75-149.

Roberts, L. G. (1965). Machine perception of three-dimensional solids. In: Optical and Electro-Optical Information Processing, J. T. Tippet et al., (Ed.), MIT Press, Cambridge, Massachusetts, 159-197.

Yager, R.R. (1979). On the measure of fuzziness and negation. Part 1: membership in the unit interval. Int. Journal of Genet. Syst., 5, 221-229.

Zadeh, L. A. (1965). Fuzzy sets, Information and Control, 8, 338-353.

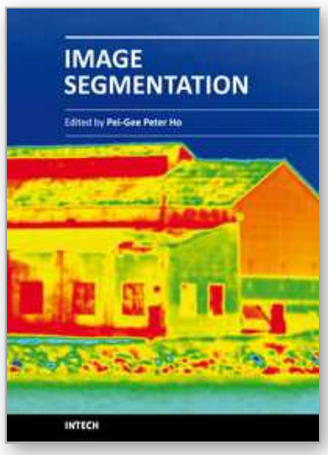


Image Segmentation
Edited by Dr. Pei-Gee Ho

ISBN 978-953-307-228-9
Hard cover, 538 pages
Publisher InTech
Published online 19, April, 2011
Published in print edition April, 2011

It was estimated that 80% of the information received by human is visual. Image processing is evolving fast and continually. During the past 10 years, there has been a significant research increase in image segmentation. To study a specific object in an image, its boundary can be highlighted by an image segmentation procedure. The objective of the image segmentation is to simplify the representation of pictures into meaningful information by partitioning into image regions. Image segmentation is a technique to locate certain objects or boundaries within an image. There are many algorithms and techniques have been developed to solve image segmentation problems, the research topics in this book such as level set, active contour, AR time series image modeling, Support Vector Machines, Pixon based image segmentations, region similarity metric based technique, statistical ANN and JSEG algorithm were written in details. This book brings together many different aspects of the current research on several fields associated to digital image segmentation. Four parts allowed gathering the 27 chapters around the following topics: Survey of Image Segmentation Algorithms, Image Segmentation methods, Image Segmentation Applications and Hardware Implementation. The readers will find the contents in this book enjoyable and get many helpful ideas and overviews on their own study.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen