Bulletin of the *Transilvania* University of Braşov
Series I: Engineering Sciences • Vol. 4 (53) No. 1 - 2011

A FUZZY LOGIC BASED METHOD FOR EDGE DETECTION

C. SULIMAN[1] C. BOLDIŞOR[1] R. BĂZĂVAN[2] F. MOLDOVEANU[1]

Abstract: *In this paper we present a method for detecting edges in grayscale images. The method is based on the use of a fuzzy classifier. The difference between our method and other similar methods is the use of a morphological operation to thin the obtained edges. Our entire algorithm was implemented in the development/simulation environment called Matlab. The experiments have shown promising results, we have obtained the desired thickness of the edges.*

Key words: *fuzzy logic, fuzzy classifier, computer vision, edge detection, morphological skeleton.*

1. Introduction

In computer vision applications, the difference between the objects from the scene and the rest of the scene is an extremely important task. Thus, in order to extract the contour of an object we must have full information on the edges of the image. Therefore, edge detection is an essential operation in image processing. An extremely important property of any edge detection method is its ability to extract precise and good oriented image edges. The performance of edge detection methods has always been subjective since each user seeks to obtain certain results from an image.

In the literature there are numerous methods used to detect edges, some of them leading to the appearance of some of the classical edge detectors like the Roberts edge detector [6], the Sobel [6] or the Prewitt edge detector [6]. More recently, among conventional edge detection techniques emerged some new methods based on fuzzy logic [1], [4], [5].

In the second chapter of the paper we present the six classes to which an image pixel can belong to. In the third chapter we present the fuzzy classifier architecture that we have used. In chapter four we mention the fuzzy rules used to carry out the competition between the pixels of a mask. Also, we present here the results from the fuzzy edge detection and the results after applying the morphological skeleton [2], [3] operation over the output image. The paper ends with some conclusions.

2. Pixel Classification

As a first step before applying the fuzzy logic based techniques for edge detection, the image should be pre-processed. Since images usually contain noise, this should be removed. For the noise removal we have chosen a 3 x 3 Gaussian filter.

[1] Dept. of Automatics, *Transilvania* University of Braşov.
[2] Dept. of Telecommunications and Information Technology, *Politehnica* University of Bucureşti.

We consider the case in which we use a 3 x 3 mask, with the central pixel situated at coordinates (i, j). Since the edge detection method is applied to grayscale images, the pixel position is represented as a scalar $p_{i,j}$. The representation of such a mask can be seen in the next figure:

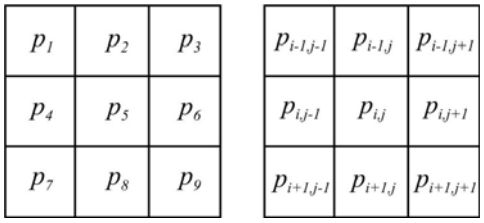


Fig. 1. *3 x 3 mask with the central pixel at* $p_{i,j}$

An edge may appear in many directions. In Figure 2 are presented the four cases in which an edge may appear.

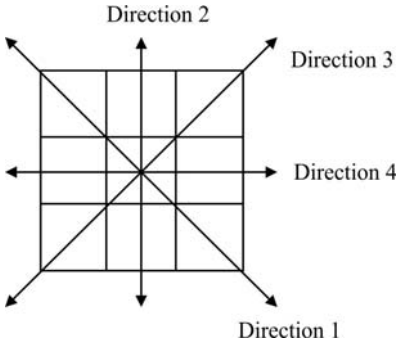


Fig. 2. *The four directions in which an edge may appear*

For the four direction, denoted by d_1, d_2, d_3, d_4, it is necessary to calculate the sum of the differences of the bidirectional amplitudes between the p_5 pixel and its neighbours. These will be computed as follows:

$$d_1 = |p_1 - p_5| + |p_9 - p_5|,$$
$$d_2 = |p_2 - p_5| + |p_8 - p_5|,$$
$$d_3 = |p_3 - p_5| + |p_7 - p_5|,$$
$$d_4 = |p_4 - p_5| + |p_6 - p_5|.$$

For each pixel in the input image that is not at the periphery, we must form a vector $x = (d_1, d_2, d_3, d_4)$ that contains the four previously calculated distances.

The next step is to divide the input image pixels into classes. For this purpose we defined six classes: four classes for edges, a background class and a class for noisy edges. For the four classes that define the edges we have used four typical situations (see Figure 3).

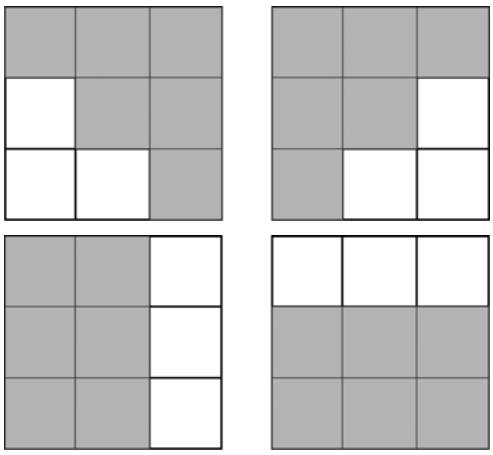


Fig. 3. *Typical situations for the four classes of edges*

Each of the four situations described above corresponds to a single vector of amplitudes. The amplitudes will be related only to the minimum and maximum values that they may have. The appropriate class for the image background will correspond to any pixel in whose neighbourhood the amplitude difference in all four directions is small. The last class, in which an edge is regarded as containing noise, the amplitude change in the vicinity of a pixel in all four directions is considered to be high. Examples of cases where a pixel is part of the last class can be seen in Figure 4c and d.

At this point it is necessary to build six prototype vectors, which will be noted with c_0, c_1,…,c_5. These vectors are the centres of the six classes defined previously. The

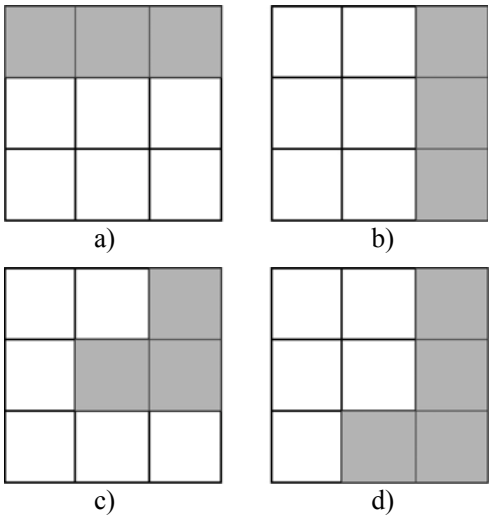


Fig. 4. *Typical situations for the four classes of edges*

vectors will contain the "min" and "max" attributes. These attributes correspond to the amplitude in the four directions. The attributes "min" and "max" will be defined by the user depending on the degree of sensitivity needed in the application. The six vectors are given below:

$c_0 = (\min, \min, \min, \min)$ - background,

$c_1 = (\min, \max, \max, \max)$ - edge,

$c_2 = (\max, \min, \max, \max)$ - edge,

$c_3 = (\max, \max, \min, \max)$ - edge,

$c_4 = (\max, \max, \max, \min)$ - edge,

$c_5 = (\max, \max, \max, \max)$ - noisy edge.

The six vectors are central points for the six sets of membership functions corresponding to the six classes. The membership functions are represented by symmetrical triangular functions and are determined by the central vector c_i and a parameter, w, with which you can change the base of the surface. The parameter w is set by the user.

After reading the input image, each pixel must be classified as belonging to one class, otherwise it will be considered as belonging to the background, and its colour will be changed to black.

3. The Fuzzy Classifier Architecture

A fuzzy classifier is a system that accepts as inputs vectors containing attributes or fuzzy truths that allow fuzzy attributes to belong to different membership functions. The output of the classifier is represented by fuzzy truths for the membership functions corresponding to the input vector. The class attributed to an input vector is the one whose truth value, given by the membership functions, is the greatest. For an input vector to belong to only one class, the highest truth value must beat by far the following truth value. The architecture of a fuzzy classifier for only two classes is presented in Figure 5. In this type of architecture each node of the hidden layer is represented by a triangular type function centred on a prototype $\boldsymbol{c}_i$. These membership functions are presented in the following relations:

$$\mu_0(x) = \max\left\{0, 1 - \frac{\|\mathbf{x} - \mathbf{c}_0\|}{w}\right\} \text{ - for class 0,}$$

$$\mu_1(x) = \max\left\{0, 1 - \frac{\|\mathbf{x} - \mathbf{c}_1\|}{w}\right\} \text{ - for class 1,}$$

$$\mu_2(x) = \max\left\{0, 1 - \frac{\|\mathbf{x} - \mathbf{c}_2\|}{w}\right\} \text{ - for class 2,}$$

$$\mu_3(x) = \max\left\{0, 1 - \frac{\|\mathbf{x} - \mathbf{c}_3\|}{w}\right\} \text{ - for class 3,}$$

$$\mu_4(x) = \max\left\{0, 1 - \frac{\|\mathbf{x} - \mathbf{c}_4\|}{w}\right\} \text{ - for class 4,}$$

$$\mu_5(x) = \max\left\{0, 1 - \frac{\|\mathbf{x} - \mathbf{c}_5\|}{w}\right\} \text{ - for class 5,}$$

where $\boldsymbol{x}$ is the input vector for a pixel, $c_0 \ldots c_5$ are the five prototypes of classes defined above, and w is the parameter used to modify the radius of the membership functions.

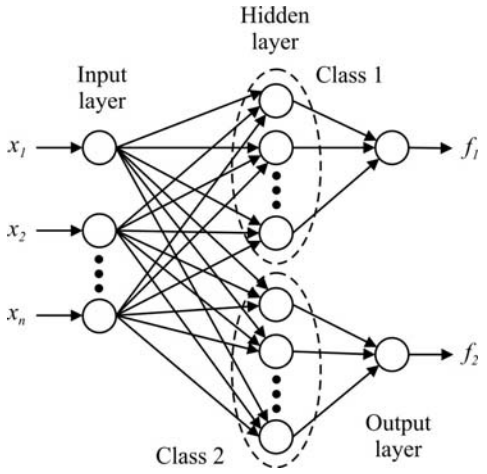


Fig. 5. *The fuzzy classifier architecture for two classes*

In the fuzzy classifier architecture presented in Figure 5 the output is represented by a single node for each class. This node summarizes all the values that reach him from the hidden layer. When the membership function of a node in the hidden layer has a high value, the node in the output layer, where the summing takes place, will also have a high value. This way the class will be given by the output node with the highest value.

This type of architecture leads to undesirable results in the case of an edge detector. By using this type of architecture one can obtain thick edges, or in an image processing application thin edges are needed.

To overcome the problems that may appear in the case of using the above architecture we will use a architecture in which in the hidden layer is no longer necessary to use multiple nodes for each class, it is sufficient to use a single node. This way results a two level architecture (see Figure 6).

The next step in edge detection is to apply a set of rules to from image pixels that have already been classified.

In the output image, pixels that have already been classified as components of an edge or those which are part of a noisy edge will have the value 1. This value corresponds to the colour white in the binary images. The pixels that will be classified as belonging to the background will have in the output image the value 0 (black). As a result of this process we will have a binary image.

4. The Fuzzy Rule Set

Before a pixel from the input image is to be changed to white or black in the output image, it must win the competition with all the pixels in its neighbourhood. The rules that carry out the competition between pixels are presented below:

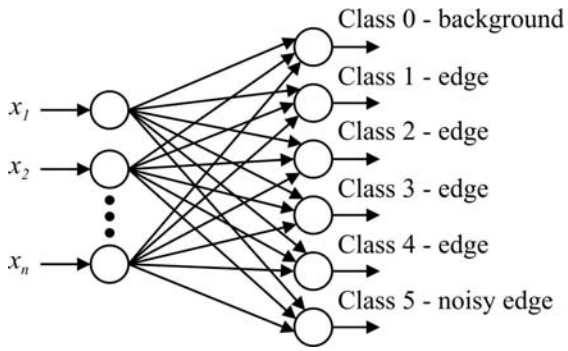


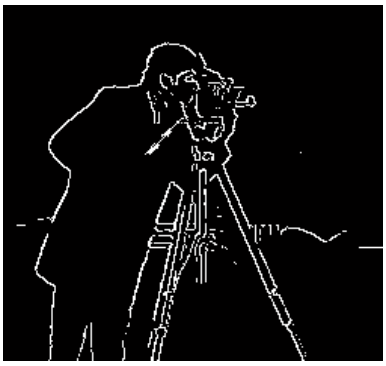
Fig. 6. *The fuzzy classifier architecture used in the fuzzy edge detection process*

- IF ***x*** belongs to class 0 THEN change pixel to black;
- IF ***x*** belongs to class 1 THEN carry out the competition between d_3 and the neighbouring pixels in direction 3;
  - o If d_3 wins the competition THEN change pixel to white OTHERWISE change pixel to black;
- IF ***x*** belongs to class 2 THEN carry out the competition between d_4 and the neighbouring pixels in direction 4;
  - o If d_4 wins the competition THEN change pixel to white OTHERWISE change pixel to black;
- IF ***x*** belongs to class 3 THEN carry out the competition between d_1 and the neighbouring pixels in direction 1;
  - o If d_1 wins the competition THEN change pixel to white OTHERWISE change pixel to black;
- IF ***x*** belongs to class 4 THEN carry out the competition between d_2 and the neighbouring pixels in direction 2;
  - o If d_2 wins the competition THEN change pixel to white OTHERWISE change pixel to black;
- IF ***x*** belongs to class 5 THEN change pixel to white.

After applying the method presented so far we have obtained edges close to the desired thickness. The image used in our experiments is a representative one for the image processing domain. We used a photo of a cameraman (see Figure 7a). The thickness of the obtained edges is of 2 pixels (see Figure 7b). However, for the method to give the best results, the obtained edges should have a thickness of 1 pixel.

To solve this problem and to be able to obtain edges of the desired thickness we have chosen to apply a morphological operation, in particular the method of morphological skeletonization. This method reduces all objects in the image to simple lines without changing the essential structure of the image.

a)



b)

Fig. 7. *Fuzzy edge detection: a) original image; b) image containing detected edges*

After applying the morphological skeleton method, the problem with the thick edges was removed. The result can be seen in Figure 8.

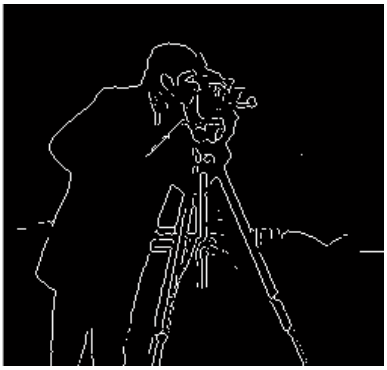


Fig. 8. *Edges after morphological skeletonization*

Edges now have a thickness of 1 pixel, that is the desired thickness. The structure of the image after applying the skeleton was not affected.

5. Conclusions

In this paper we have presented a fuzzy based method for edge detection. The method alone was not able to detect edges of the desired thickness, but with the help of the morphological skeleton we were able to extract edges with the thickness of 1 pixel.

Our entire algorithm was developed in the environment called Matlab. One direct improvement of this method is to implement an edge linking algorithm so that one can obtain continuous edges. This method can also be combined with the CPDA (C*hord-to-*P*oint* A*ccumulation* T*echnique*) and can be used in the process on corner extraction.

Acknowledgement

This paper is supported by the Sectoral Operational Programme Human Resources Development (SOP HRD), financed from the European Social Fund and by the Romanian Government under the contract number POSDRU/6/1.5/S/6.

References

1. Alshennawy, A.A., Aly, A.A.: *Edge Detection in Digital Images Using Fuzzy Logic Technique*. In: World Academy of Science, Engineering and Technology **51** (2009), p. 178-186.
2. Gonzalez, R.C., Woods, R.E.: *Digital Image Processing*. Upper Saddle River, New Jersey, USA. Prentice Hall, 2002.
3. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: *Digital Image Processing Using Matlab*. Upper Saddle River, New Jersey, USA. Prentice Hall, 2003.
4. Liang, L.R., Looney, C.G.: *Competitive Fuzzy Edge Detection*. In: Applied Soft Computing **3** (2003), p. 123-137.
5. Liang, L.R., Basallo, E., Looney, C.G.: *Image Edge Detection with Fuzzy Classifier*. In: Proc. of the 14th International Conference on Computer Applications in Industry and Engineering, Las Vegas, U.S.A., 2001, p. 279-283.
6. Senthilkumaran, N., Rajesh, R.: *Edge Detection Techniques for Image Segmentation - A Survey of Soft Computing Approaches*. In: International Journal of Recent Trends in Engineering **1** (2009) No. 2, p. 250-254.