

Edge Detection Tecnique By Using Neuro Fuzzy Sysytem

Vandana Mittal*, Mrs. Shalini Batra**

Deptt. of Computer Sci. & Engg., Thapar Universty, Patiala

Abstract-Most commonly used methods of edge detection are Canny and Sobel. An Edge detection technique using neuro fuzzy system is proposed in this paper. The use of Neural Networks for edge detection is in its infancy. First, the distance measures between the vector in 4 directions and the six edge vectors for each pixel are taken as input pattern and fed into input layer of the self-organizing neural network. By classifying the type of edge through this network, the thick edge image is obtained. After classifying, it has been utilize the competitive rule to thin the thick edge image in order to get the fine edge image. In the end, the stain edges are removed from the edge image, and the final optimal edge image is got. It has been compared the edge images got from our method with that from Canny's one and Sobel's one in experiments. The experimental results show that the effect of this method is better to other two methods.

Keyword: Image Processing, Fuzzy logic, Edge Detection

I. INTRODUCTION

Since edge detection is in the forefront of image processing for object detection, it is crucial to have a good understanding of edge detection methods. Edge detection is one of the most commonly used operations in the image analysis.

An edge is defined by a discontinuity in gray level values. In other words, an edge is the boundary between an object and the background. The shape of edges in images depends on many parameters: The geometrical and optical properties of the object, the illumination conditions, and the noise level in the images. Edges include the most important information in the image, and can provide the information of the object's position.[1] Edge detection is an important link in computer vision and other image processing, used in feature detection and texture analysis. Most previous edge

detection techniques used first-order derivative operators such as the Sobel edge operator,[2] the Prewitt edge operator and the Robert edge operator. The Laplacian operator is a second-order derivative operator for functions of two-dimension operators and is used to detect edges at the locations of the zero crossing. However, these points of zero crossing aren't certainly the edge points and can only be determined to be edge points by further detection. Another gradient operator is the Canny operator that is used to determine a class of optimal filters for different types of edges,[3] e.g., step edges or ridge edges. A major problem in Canny's work is that a trade-off is emerged between detection and localization: as the scale parameter increases, the detection accuracy increases but the localization accuracy decreases. In order to set the appropriate

value for the scale parameter, the noise energy must be known. However, it is not an easy task to locally measure the noise energy because both noise and signal affect the local measure. Many researchers have proposed the statistical or stochastic methods of boundary detection, but these methods don't always perform well since they lack powerful two-dimensional structure knowledge and employ only a single formula to treat different edge patterns.[4] Due to the effects of noise and other factors, all the edge detection methods above may lead a result not satisfied, such as false edge or missing edges, for many complicated actual images.

In recent years, an increasing number of researches have been involved in research of the subjects of fuzzy logic and neural network in the hope of combining the superiority of fuzzy logic and neural network to achieve a more useful tool for fuzzy information processing.[5-8] In this paper, a new edge detection method of combining fuzzy logic and neural network is presented. To verify the method proposed in this paper, we compare it with Canny method and Sobel method in the term of images which have no noise and noise respectively. The experimental results show that the effect of our method is superior to that of Canny method and Sobel method.

II. PIXEL EDGE CLASSIFICATION AND NEURAL NETWORK

Figure1 shows 3x3 the neighborhood of pixels about the center pixel p5 as well as the four edge directions which may appear. The bi-directional grey level summed magnitude of differences between p5 and its neighbors are designated respectively by d1, d2, d3 and d4 for directions 1, 2, 3 and 4, are shown in Figure1 and are calculated by,

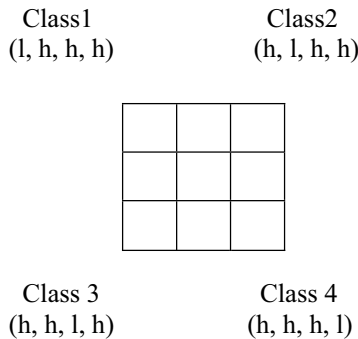
$$\begin{aligned} d1 &= |p1-p5| + |p9-p5| \\ d2 &= |p2-p5| + |p8-p5| \\ d3 &= |p3-p5| + |p7-p5| \\ d4 &= |p4-p5| + |p6-p5| \end{aligned} \tag{1}$$

For each pixel in an input image that is not on the outer boundary of the image, we define its four-dimensional feature.

P1	P2	P3
P4	P5	P6
P7	P8	P9

Figure 1. Pixels and directions in neighborhood 3x3

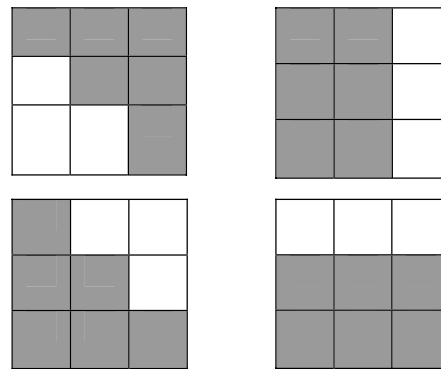
Each edge class has a single feature vector of four directional grey level summed magnitudes of differences as far as the low and high values are concerned. According to the feature vector of pixel, we classify pixels into 6 classes (four edge classes, a background class and a speckle edge class). Four typical neighborhood situations are shown in Figure2. Among them the grey level summed magnitudes of differences of class 1 are ‘l’ in direction 1 and ‘h’ in directions 2, 3, 4. The summed magnitudes of differences of class 2 are ‘l’ in direction 2 and ‘h’ in directions 1, 3, 4; the grey level summed magnitudes of differences of class 3 are ‘l’ in direction 3 and ‘h’ in directions 1, 2, 4; the grey level summed magnitudes of differences of class 4 are ‘l’ in direction 4 and ‘h’ in directions 1, 2, 3; the background class is for the pixel whose neighborhood has low grey level summed magnitude of differences in the four directions; the speckle edge class is used for pixels on whose neighborhood the change magnitudes in all directions are high.



Given a pixel, any neighborhood has a situation that determines a feature vector such as $x = (3, 35, 26, 4)$, of magnitudes of differences in each of the four directions shown in Figure1. We construct 6 prototype vectors C_0, \dots, C_5 to be the respective centers of the 6 classes (four edge classes, one background class and one speckle edge class). These centers or prototypes for the respective classes have component values ‘l’ and ‘h’ that represent low and high grey level summed magnitude of differences in the directions indicated. The parameters l and h are set by the user depending on the image region contrasts and the noise sensitivity desired. For example, l could be set to a gray level difference of 5 and h set to a value from 30 to 40. These low and high values decide the prototype vectors C_0, \dots, C_5 .

A. The Self-Organization Competitive Neural Network

Self-organization competitive neural network is a neural network which conducts the network training by no teacher’s guiding and has the function of self-organization. Through training itself the network classifies the input pattern automatically. The simple working process of self-organization competitive neural network is that after inputting the pattern vector, the network lets the neurons of output layer start competition according to a rule, and when a neuron wonned network lets the connective weights structure to be updated along the direction which can make the winning neuron more sensitive to this pattern. When the network input again this pattern or similar pattern, this neuron wins easier. At the same time, the other neurons are restrained and aren’t sensitive to this pattern, thus they have difficult to win. When there are other pattern inputting, these neurons take part in the hopeful competition again. These characteristics of self-organization competitive neural network make it have high application value in pattern classification. The neural network proposed in this paper is a self-organization competitive neural network (shown as Figure3), consisted of two layers of neuron(input layer and competitive layer) and a set of connective weight. There are six neurons in input layer,



indicating respectively the distance measures between the feature vector of pixel and the six edge prototype vectors. (u_0 indicates the distance measure from feature vector to background, u_1 to class 1 edge, u_2 to class 2 edge, u_3 to class 3 edge, u_4 to class 4 edge, u_5 to speckle edge.) There are six neurons in the competitive layer which correspond respectively to the six edge classes. (C_0 indicates background, C_1 class 1 edge, C_2 class 2 edge, C_3 class 3 edge, C_4 class 4 edge, C_5 speckle edge.)

Due to the Euler distance between the feature vector of the pixel and the edge prototype vectors in different images is differ in thousands ways and the near distance in a image may be the far distance in another image, so only upon providing more training samples can we make the network recognize the edge pixels in different type images. In fact, the ‘far’ or ‘near’ distance is a fuzzy concept in the given patterns. For this, we propose a ‘measure’ concept, and the Euler distance between the feature vector of pixel and the edge prototype vector is fuzzified. If the measure approaches 1, it indicates the near

distance; if the measure approaches 0, it indicates the far distance.

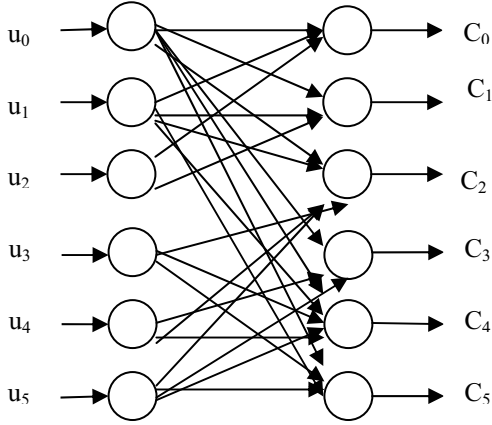


Figure 3. The competitive neural network

$$\underline{U}_i = 1 - \|x - c_i\| / W \quad (2)$$

where, w is the biggest Euler distance obtained from computing in the image, and the quality of edge detection depends mainly on the parameters l , h and w .

Feeding the preprocessed distance measures into the above self-organization competitive neural network, the network will recognize automatically which edge type the pixel belongs to. Suppose that the connectional weights of the network are $\{W_{ij}\}$, $i = 1, 2, \dots, 6$, $j = 1, 2, \dots, 6$, and the restrictive condition is

$$\sum_k W_{ij} = 1 \quad (3)$$

and suppose $P_k = (p_1, p_2, \dots, p_6)$ is one of the studying patterns, then the input value of the neurons of the competitive layer is,

$$S_j = \sum_i W_{ij} p_i \quad i=1 \dots 6, j=1 \dots 6 \quad (4)$$

According to the rule of ‘the winner is the king’, the neuron corresponding to the maximum value in S_j ($j = 1, 2, \dots, 6$) is considered as the ‘winner’ and its output status is set as 1, while the output status of all the other neurons are set as 0, then the connectional weights connected with the winning neuron are updated by the following equation and the other connectional weights keep unchangeable.

$$W_{ij} = W_{ij} + \Delta W_{ij} \\ \Delta W_{ij} = \eta * ((p_i^k) / m - W_{ij}) \quad (5)$$

where, η is the studying coefficient, and $0 < \eta < 1$, is the number of elements with value ‘1’ in the study pattern $P_k = (p_1, p_2, \dots, p_6)$.

B. The Competitive Rules

After recognizing the edge type of each pixel by the network, a competitive selection is conducted for each edge pixel according to its assigned class. Only the edge pixels that are first classified as edge pixels and then win in the edge competition, or speckle edge pixels, are mapped as the edge pixels in the new output map, and all other pixels are mapped as background.

Once a pixel is classified as an edge class, it will compare with the two adjacent edge pixels along the edge direction. For these three pixels, only the one with the largest grey level summed magnitude of difference is saved as a white edge, the others are saved as black background, thus the thick edge image got from the self-organization competitive neural network is thinned. These competitive rules are as follows:

If x is class 0 (background), then change pixel to black.

If x is class 1 (edge), then compare this pixel with d_3 value of neighborhood pixels in direction 3.

If it wins, then change it to white (edge), else change to black.

If x is class 2 (edge), then compare this pixel with d_4 value of neighborhood pixels in direction 4.

If it wins, then change it to white (edge), else change to black.

If x is class 3 (edge), then compare this pixel with d_1 value of neighborhood pixels in direction 1.

If it wins, then change it to white (edge), else change to black.

If x is class 4 (edge), then compare this pixel with d_2 value of neighborhood pixels in direction 2.

If it wins, then change it to white (edge), else change to black.

If x is class 5 (speckle edge), then change pixel to white (edge).

III. THE REALIZATION OF THE ALGORITHM OF EDGE DETECTION

Because the speckle edge mapped as a white edge pixel is not always an edge, we implement a despeckler that removes isolated single and double edge pixels from the edges after the edge competition have been done. Before the edge detection is done we pretreated image by a smoothing operator. There are three steps in the edge detection algorithm. First, the feature vector $x = (d_1, d_2, d_3, d_4)$ of each pixel and the distance measures between it and the six prototype vectors ($C_0, C_1, C_2, C_3, C_4, C_5$) are computed, and the distance measures are fed into the self-organization competitive neural network for edge classification, and the thick edge image is got. Afterwards the edge image obtained is thinned with competitive rules and at last the speckles are removed to get the final edge image.

A. Fuzzy Classification

I: set parameters l and h .

II: for each pixel in the image,

Compute and save the bi-directional grey level summed magnitudes of differences,

construct the feature vector x and compute the distance measures between it and the six prototype vectors and feed the distance measures into the neural network,

the type of the edge of pixel is recognized by the network automatically and the thick edge image is obtained.

B. Competition Of Edge Strengthening

I: for each pixel in the image,
 if it is edge class, then apply appropriate competition rule and record pixel value,
 if it is background class, then change the pixel as black,
 if it is speckle edge class, then change the pixel as black.

C. Despeckling

I: for each pixel in the image,
 If it is isolated single or double speckle, then change the pixel as black

IV. RESULTS

The source image used for simulating experiment is shown in Figure4 (a). All of our experimental results were obtained by detecting every pixel in the image using a neighborhood. The appropriate parameters l and h must be provided to achieve good results. From the experimental results, we think that the detective result is the best when $l = 5$ and $h = 40$. The smaller is the h value, the more sensitive edges and the more noise are produced; whereas the larger is the l value, the more false edges are produced. We also use Canny method to detect the edges in the source image. In Canny method, we found that a smaller threshold T gives more details and noises, and a smaller also gives more details but no noise. So T is the most sensitive. Through experiments, we found that the effect of Canny edge detection method is best when $T = 0.04$ and $= 0.6$. $33 \times \sigma \sigma$ Figure4 (b), Figure4 (c) and Figure4 (d) show respectively the best results obtained with Canny edge detection method, our fuzzy neural network edge detection method and Sobel edge detection method. It is can be seen from these figures that the edge image obtained with our method detected many details which are not detected by Canny method and Sobel method, thus having the best effect. The edge image obtained with Sobel method lacks lots of details and its effect is the worst. The effect of Canny method is between our method and Sobel method .

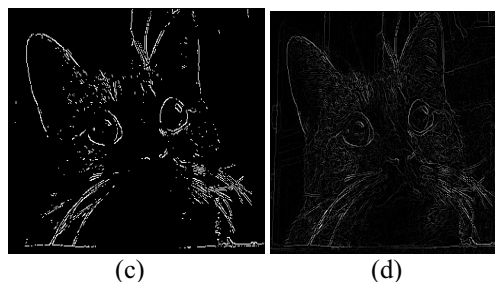
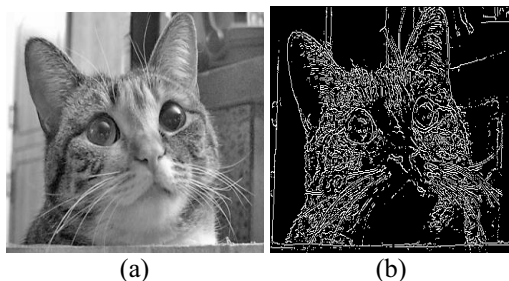


Figure 4: source image without noise and its stimulating results

CONCLUSIONS

We define the gray level summed magnitude of differences in four directions on 3×3 neighborhood of each pixel as a feature vector, construct six edge vectors and take the distance measures between the feature vector and the prototype vectors as inputs and feed them into the self-organization competitive neural network to recognize whether the pixel is the edge, speckle or background to get the thick edge image . The thin edge image is obtained by using the competitive rule in the thick edge image. From the experimental results, it can be seen that the edge detection method proposed in this paper is superior to Canny method and Sobel method, this advantage is more prominent under the noisy condition and the robusticity is better.

REFERENCES

- [1] RongWang, Li-qun gao, Shu Yang, Yan-Chun Liu: An Edge detection method by combining fuzzy logic and neural networks.7(2005) 4539-4543
- [2] Lily Rui, Liang, Carl G., Looney.: Competitive Fuzzy Edge Detection. Applied Soft Computing, 3 (2003) 123-137.
- [3] J. Canny.: A Computational Approach to Edge Detection. IEEE Trans, Pattern Anal. Mach. Intell. 8 (6) (1986) 679-687.
- [4] W. Deng, S.S. Iyengar.: A New Probabilistic Relaxation Scheme and Its Application to Edge Detection. IEEE Trans, Pattern Anal. Mach. Intell. 18 (4) (1996) 432-437
- [5] H.S. Wong, L. Guan.: A Neural Learning Approach for Adaptive Image Restoration Using a Fuzzy Model-based Network Architecture. IEEE Trans, Neural Network 12 (3) (2001) 516-531.
- [6] C.G. Looney.: Nonlinear Rule-based Convolution for Refocusing. Real Time Imaging, 6 (2000) 29-37.
- [7] H. Maturino-Lozoya, D. Munoz-Rodriguez, F. Jaimes-Romero, H. Tawfik.: Handoff Algorithms Based on Fuzzy Classifiers. IEEE Trans, Vehicular Technol, 49 (6) (2000) 2286-2294.