A new edge detection method
Based on threshold binarization

H. Mehrara¹, E. Zahedinejad², A. Pourmohammad³
¹Young Researchers club, Science and Research Branch, Islamic Azad University, Tehran, Iran.
Email: h.mehrara@ieee.org
²Department of mathematics science, Isfahan university of technology, Isfahan, Iran
³Electrical Engineering Department, Amirkabir University of Technology, 424 Hafez Avenue, Tehran 15914, Iran.
Email: elahezahed61@gmail.com, pourmohammad@aut.ac.ir

Abstract—The most fundamental features of a digital image are the edges of objects. The edge detection method's reliability and accuracy will affect directly on the comprehension machine vision system. Several edge detectors have been developed in the past decades as Sobel and Canny. Although no single edge detector has been developed satisfactorily enough for all applications. In this paper, a new edge detection method is proposed based on the threshold binarization using multiple methods as BP_MLP neural network and Boolean algebra. Based on this method, the edge patterns of binary images are classified into 16 possible types of visual patterns. Experimentally results demonstrate improving in the computations mass and mathematical complexity in comparison with the traditional edge detection methods.

Index Terms—Image processing, Neural Networks, Edge detection, Boolean Algebra, Image binarization.

I. INTRODUCTION

In image processing and computer vision, edge detection is a process which attempts to capture the significant properties of objects in the image. These properties include discontinuities in the photometrical, geometrical and physical characteristic of objects. Such information give rise to variations in the grey level image; the most commonly used variations are discontinuities (step edges), local extrema (lines edges), and 2D features formed where at least two edges meet (junctions) [1]. The purpose of edge detection is to localize these variations and to identify the physical phenomena which produce them.

Up to now, many edge detection techniques, such as first derivative algorithm, second derivative algorithm, template matching, edge fitting, and statistical approaches, have been proposed, and several commercial systems are on the market. As the validity, efficiency and possibility of the completion of subsequent processing stages rely on edge detection, it must be efficient and reliable. To fulfill this requirement, edge detection provides all significant information about the image. For this purpose, image derivatives are computed but image derivatives are sensitive to various sources of noise, i.e., electronic, semantic, and discretization/quantification effects. To regularize the differentiation, the image must be smoothed. However, there are undesirable effects associated with smoothing, i.e., loss of information and displacement of prominent structures in the image plane. Furthermore, the properties of commonly-used differentiation operators are different and therefore they generate different edges.

It is difficult to design a general edge detection algorithm which performs well in many contexts and captures the requirements of subsequent processing stages. Consequently, over the history of digital image processing a variety of edge detectors have been devised which differ in their purpose and their mathematical and algorithmic properties. We have proposed a heuristic approach which detects edges of an image most efficiently. The key features of our approach which differentiate us from others are the use of image content simulated with different methods: Articial Neural Network (NN), Boolean Functions (BF) and Matrix Manipulation (MO) for edge detection of application-specific image. The proposed techniques can be extended for color images as well. This paper describes the characteristics of edges, the properties and the methodology of proposed edge detections. From section 2 to section 5, we will introduce related work and new edge detection methods, and summarized its results. Finally, this paper analyzed proposed methods advantages and disadvantages.

II. BACKGROUND

There are many methods for edge detection, but most of them can be grouped into three categories: search-based, zero-crossing based and threshold base. The search-based method detects edges by first computing a measure of edge strength, usually a first-order derivative expression such as the gradient magnitude, and then searching for local directional maxima of the gradient magnitude using a computed estimate of the local orientation of the edge, usually the gradient direction. Canny formulated an optimal edge detector based on derivatives of the Gaussian, Deriche introduced a fast recursive implementation of Canny's edge detector, Lanser and Eckstein improved the isotropy of Deriche's recursive filter, and Jahne et al. provide a nonlinear optimization strategy for edge detectors with optimal isotropy.

Edge detection based on second-order difference (zero crossings) was strongly influenced by biological vision.
The pioneering work is described by Marr and Hildreth. More recent work towards unified frameworks for neighborhood operators can be found in Koen-derink and van Doom and Danielsson et al. The zero-crossing based method searches for zero crossings in a second-order derivative expression computed from the image in order to find edges, usually the zero-crossings of the Laplacian or the zero-crossings of a non-linear differential expression. As a pre-processing step to edge detection, a smoothing stage, typically Gaussian smoothing, is almost always applied.

Raman Maini and J. S. Sobel had given their efforts to evaluate the performance of the Prewitt edge detector for noisy image. They had demonstrated the limitations of the Prewitt edge detector works. Both the gradient and derivative based operations are simple and easy to be implemented. But the main disadvantages are the thick and partially connected borders of regions and their unsteadiness with respect to the noise.

Types of edge detectors based on thresholding can be grouped into two classes: (a) local techniques, which use operators on local image neighborhoods and (b) global techniques, which use global information and filtering methods to extract edge information [2]. Both methods have their advantages and disadvantages on various types of images. Nearly all detectors utilize thresholding of the image for edge detection. Each pixel in the image is compared with this threshold value. If the pixel's intensity is higher than the threshold value, the pixel will set to White in the output image. If it is less than the threshold, it will set to Black. The efficient selection of single threshold value is the most important and the most difficult process in edge detection technique. Edge detectors based on local techniques, use local feature for selecting threshold value. Similarly, edge detectors based on global techniques, use global feature for selecting threshold value. An image contains variations at different levels, use of a single global threshold over the whole image gives poor results. But local threshold method also detects false edges due to noise. The global threshold value depends on the presence of noise in the image.

More recently there have been several papers published on the use of neural networks for edge detection. Paik, Brailean, and Katsaggelos considered multi-state ADALINES for edge detection. Wang [3] proposed algorithm used bit plane slicing to binarize gray level image, an optimized neural network using 3x3 sliding window is trained to extract edge of any bit plane and finally are weighted to detect edges accurately by the parallel model. Terry & He [4, 5] proposed algorithm used NN by introducing primitive and constrained pattern of each edge map to train the network, also Basturk [6] exert cellular NN for this purpose.

Bilal Ahmad & choi[7] & Vemis [8] employed mean operator on 3x3 pre-defined window to threshold locally which Boolean function exert on windows to extract output edges. Jiang [9] mapped grayscale or multibit image to a set of several morphological image by using polynomials and a fusion of individual edge map on each binary images. Our approach differs in the form of the data and the network structure. Our approach differs in the form of the data and the network structure; it makes use of feedforward networks, trained by backpropagation. The input and output patterns and the preparation of training data for edges also differ.

III. OUR WORK

This paper is base on unique algorithm which is implemented by neural network, Boolean algebra and matrix operation. Neural network can be a useful tool for edge detection, since a neural network edge detector is a nonlinear filter. An edge detector base neural network can be trained with backpropagation multi layer perceptron(BP-MLP) using relatively few training patterns. The most difficult part of any neural network training problem is to define the proper training set. A simple method is recommended for the edge detection training problem. Boolean functions (BF) are fast operators and are also well-defined which can substitute NN approach. In three subsections we will describe fundamental and implementation of this new method by NN, BF and MO sequentially.

A. Neural Network Algorithm

1) Identify the Algorithm: The suggested algorithm is shown in Fig. 1. To detects edges in a grey level image, we firstly binarize the image by modified Otsu's method threshold value. Researchers have proposed m number of techniques to improve selecting thresholds or to provide some criteria for optimal selection for threshold value. No local threshold will solve this problem. Otsu [10] suggested minimized group variances for the probable distribution of gray value as optimal criteria. Different from Otsu's suggestion, Kittler and Illingworth used a mixture of two Gaussian distributions, by adjusting the proportions of two distributions, to approximate histograms. As we will show in Fig.7, some thresholding method (e.g. mean value, median value…) contain so much detail that computer edge recognizer does not require them and sometime will lid identifying edge pattern wrongly.

Best threshold value which concludes better output is Otsu's method empirically. Base on histogram, Otsu selects a single gray level (index in Fig 2.) for clipping intensity but this method loses some detail up and down of this level. In modified Otsu, the gray levels which are under the index, will stretch by a scaled gamma function up to above the index, and whose are above the index, will stretch down[Fig . 2]. This reduces the contrast in an image, but the shrieked histogram is capable to reconstruct eliminated pixels under and above the index. By applying a factor used to modify the gray levels within the specified range, a new image produces. The slope may be positive, negative, or zero which negative and zero slope will cause all values in the given range mapped to the same gray level value. Finally a binary image will be generated.

Binary image disintegrate to 2x2 windows and generate a set of image pattern and then we classified...
edge patterns in binary images into 16 categories, as shown in 3.a and we start to train the neural network on these patterns. In Fig. 3, the blank elements in each 2x2 window indicate white (pixels value: 1) in binary images, whereas the dark elements indicate black (pixels value: 0). To extract edges from a binary image, we use a four pixel window in output pattern that keep all pattern except black-black-black-black (binary code: 0000) and white-white-white-white (1111). These patterns do not include intensity variation and have no edge points and neural network returning white-white-white-white (Whitewash) for both. We can also reduce noisy pattern by Whitewashing. Noisy patterns contain one black pixel that replaced using whitewashing (see Table 1). Thinning procedure also done by whitewashing one black pixel of (1000), (0100), (0010), (0001) that makes these patterns diagonal. In Fig.3.b output patterns are presented. After training the network, it can recognize the input pattern as a most similar pattern in edge pattern trained set.

![Fig. 1. Proposed algorithm](image1)

![Fig. 2. Histogram shrink by $\alpha$ constant](image2)

![Fig. 3. All possible type of input patterns and output of neural network](image3)

2) Designed neural network: The choice of images for our algorithm has some special characteristics because binary images are the input to our algorithm. Sigmoid function was used so that output extrema from a network node are 0/1. Training can be accomplished by preparing a dataset in the following manner:

Take an image object to be learned and slide it from point to point across all locations of a window which will be the input window to the pattern detection network. We use 2x2 windows because all other windows will reduce detail and include more training set (2 WindowSize) but this is efficiently simple and accurate. In our system, pixel values are range from 0 to 1, usually 0 is representing black. If the detection window which needs to be learned is 2x2 pixels, than it has $2^2$ patterns to train on. As neural networks select small training samples, so training is easy. The network structure for this example could be 4x12x4 layer: 4 inputs, 12 hidden and 4 outputs (see Fig. 4). Training will converge in less than 6 epoch using Levenberg-Marquardt backpropagation (more memory efficient) or in less than a minute by using conjugate gradient with momentum and adaptive learning rate function minimization. The network is trained on 16 pre-defined edge patterns (see Table 1). All training are done by using back propagation learning rule with a learning rate $\eta$ and a momentum $\mu$ (in our experiment $\eta=0.01$ and $\mu=0.9$). It is evidently observed that the application of momentum can effectively prevent the training progress from local minima, although the selection of the momentum value is a trial and error procedure.

![Fig. 4. Input pattern and proposed MLP neural network layers](image4)

<table>
<thead>
<tr>
<th>TABLE 1. Input and output neural network patterns</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Decimal code</strong></td>
</tr>
<tr>
<td>------------------</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
<tr>
<td>11</td>
</tr>
<tr>
<td>12</td>
</tr>
<tr>
<td>13</td>
</tr>
<tr>
<td>14</td>
</tr>
<tr>
<td>15</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

When the whole binary image is scanned by the a four-pixel window, we obtain the edges. Note that we process the image four pixels by four pixels, and the windows are not overlapped i.e. for covering all pixels we use this process 4 times; initially we start from position $(x_1y_1, x_2y_2, x_3y_2, x_3y_3)$ that all obtained images are finally multiplied pixel-by-pixel. This reduces the computation time dynamically. We also notice that the two margin rows and two margin columns around the image cannot be processed and a kind of padding required. But, in practice, the effect of margin pixels on the whole image can be ignored.

B. Boolean Function Algorithm

Proposed method also can be implemented by Boolean algebra which is faster than NN approach with the same output detected edges, but latter is more preferred because of its artificial intelligence (AI). In Table 2, input columns represented sliding window elements and second which capitalize, demonstrated output pattern. Each output pixel is obtained by truth table (Table 2) using these equations:


\[ X_1 Y_j = x_1 y_j + (x_2 y_1 \cdot x_3 y_2 \cdot x_2 y_1 \cdot x_2 y_2) + (x_2 y_1 \cdot x_3 y_2 \cdot x_2 y_1) \]

\[ X_2 Y_j = x_1 y_2 + (x_2 y_1 \cdot x_3 y_2 \cdot x_3 y_1 \cdot x_2 y_2) + (x_2 y_1 \cdot x_3 y_2 \cdot x_2 y_1) \]

\[ X_2 Y_j = x_2 y_1 + (x_1 y_2 \cdot x_1 y_2 \cdot x_2 y_1 \cdot x_2 y_2) + (x_2 y_1 \cdot x_3 y_2 \cdot x_2 y_1) \]

\[ X_2 Y_j = x_2 y_2 + (x_1 y_2 \cdot x_1 y_2 \cdot x_2 y_1 \cdot x_2 y_2) + (x_2 y_1 \cdot x_3 y_2 \cdot x_2 y_1) \]

**C. Matrix Operation Algorithm**

Due to the importance of accurate and fast edge detection for image processing applications, it is necessary to continue researching effective edge detection methods. In this section, we show that former simulations for the edge detection method will not be unique and new algorithm can give same result but faster. Using matrix operation, selected window that all possible type of its visual pattern shown in Fig.5, converted to edge pattern. In summary, this algorithm is executed as follows:

- Get Input Image.
- Step 1: Select basic threshold by Otsu’s method (Tb)
- Step 2: Initialize histogram interfering parameter: \( \alpha \)
- Step 3: Stretch graylevel [min Tb] to [min Tb+Tb*\( \alpha \)]
- Step 4: Stretch graylevel [Tb max] to [Tb-Tb*\( \alpha \) max]
- Step 5: Threshold new enhanced image by Tb
- Step 6: Apply 2x2 windows to image
- Step 7: Measure summation of 2x2 for classifying edges
- Step 8: If \( \text{sum} = 0 \) then whitewash it
  - If \( \text{sum} = 1 \) then thin edge
  - If \( \text{sum} = 3 \) then reduce noise
- Output Image.

**IV. RESULTS AND DISCUSSIONS**

In this section, the proposed method is applied to some representative images. The performance of proposed method is compared with Canny, Roberts, Prewitt and Sobel method. The following detections executed on MatLab and Fig.6 shows that proposed method has good output performance. Roberts’s operator has less continuity on the contour of the images. Canny operator has distortion on the contour of the image.

Sobel operator has better performance, but the some part of image edge is incomplete. The experiment has been executed on an Intel core2 Duo 2.5GHz processor and 3GB RAM computers. The detection results show that all the well-known Canny, Roberts, Prewitt and Sobel edges have somewhat distortion on the binary image; while the new methods has better visual performance such as the detail of the characters and improving in the computations mass and mathematical complexity in comparison with the traditional edge detection methods is obvious.

We demonstrate the use of the NN algorithm by showing the effects of the parameters. Fig.8 shows a collection of enhanced images using this algorithm. As \( \alpha \) increases, details of hidden objects will increase and by decreasing it to zero and less than zero, basic Otsu thresholding returned. This also demonstrates the importance of selecting \( \alpha \) correctly. If \( \alpha \) is set too high, then the important edges are lost. From Fig.9, it can be seen that the best two images occur at \( \alpha = 0.4 \) which are the parameters used for the Cameraman image in Fig.6. This indicates that the best balance would occur somewhere in the range 0 < \( \alpha \) < 1.
CONCLUSION

In this paper, a new edge detection method is proposed based on the threshold binarization using multiple methods as BP_MLP neural network and Boolean algebra. Experimental results show that designed neural network simply converges because of its small training sets. According to using Boolean functions and matrix operations, the computation time decreases. Also results confirm that our approach is superior to traditional edge detection methods as Sobel and Canny in computation time and complexity. It solves the problem of convergence difficulty if the BP neural network is directly used for edge detection. A Modification of proposed method using BA and MO results lower processing time. Simulations results show that MO is the fastest edge detection method. Choosing the best threshold value for binarization results more accurate edge detection. Our future work will concentrate on choosing suitable local thresholding value to hold important image details.

Fig. 6. Edge map: Original image, canny edge, Sobel edge, Roberts’s edge, Perwit edge, new detector

Fig. 7. Binary image: Global threshold (128), Median 3x3 thresholds, Mean 3x3 thresholds, Otsu’s method

Fig. 8. Edge map by default parameter

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>Sobel</th>
<th>Roberts</th>
<th>canny</th>
<th>Log</th>
<th>BP N/N</th>
<th>BA</th>
<th>MO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>1.3165</td>
<td>1.2471</td>
<td>4.5021</td>
<td>1.5006</td>
<td>1.0313</td>
<td>1.0246</td>
<td>0.0744</td>
</tr>
<tr>
<td>Min</td>
<td>0.2149</td>
<td>0.1982</td>
<td>0.4051</td>
<td>0.2150</td>
<td>0.9376</td>
<td>0.8348</td>
<td>0.0260</td>
</tr>
<tr>
<td>Avg. (in 40 epoch)</td>
<td>0.7768</td>
<td>0.7308</td>
<td>2.4594</td>
<td>0.8726</td>
<td>0.9819</td>
<td>0.9159</td>
<td>0.0285</td>
</tr>
</tbody>
</table>

TABLE 3. Time Comparison

REFERENCES


