IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)
Volume 4, Issue 2, Ver. IV (Mar-Apr. 2014), PP 22-25
e-ISSN: 2319 – 4200, p-ISSN No. : 2319 – 4197
www.iosrjournals.org

Edge Detection through Artificial Neural Networks

P. S. K. Rohit Varma, Prathik S.M., R. Rohit
IV year, B.E.Electronics and Communication departmentS. D. M. college of Engineering and Technology Dharwad, KarnatakaIndia

Abstract:*In the field of Image processing there are many techniques to detect edges. Some of the methods are 'First derivative method' ,'Second derivative method', Sobel, Laplacian of Guassian(LoG) and these methods are used for image segmentation and object identification. In this paper, a novel idea on how to detect edges using MATLAB has been described. The edges have been detected using fuzzy logic and image processing tool boxes. A novel method of detecting edges through Artificial Neural Networks(ANN) has been proposed. Experimentation is performed on gray scale image in MATLAB using First derivative, second derivative and Sobel operator.*

Keywords:*Edge detection, first and second derivative, Fuzzy logic, Artificial Neural Network*

I. Introduction

Edge detection is an important field in image processing. It can be used in many applications such as segmentation, registration, feature extraction, and identification of objects in a scene. An effective edge detector reduces a large amount of data but still keeps most of the important feature of the image. Edge detection refers to the process of locating sharp discontinuities in an image. These discontinuities originate from different scene features such as discontinuities in depth, discontinuities in surface orientation, and changes in material properties and variations in scene illumination.

Artificial Neural Network is a network which is trained to produce an output of the required program automatically. It is formed from the basis of human neuron reactions. The ability of a human to give immediate reflexes has been the basic force for the development of ANN.

There are many methods by which an ANN can be trained. We used Levenberg-Marquardt backpropogation. The function for the same is trainlm.Trainlmis a network training function that updates weight and bias values according to Levenberg-Marquardt optimization. The reason for selecting this training method is, because trainlm is often the fastest backpropogation algorithm in the toolbox, and is highly recommended as a first-choice supervised algorithm, although it does require more memory than other algorithms.

1) Edge detection methods

i) First derivative principle

An edge in a continuous domain edge segment F(x,y) can be detected by forming the continuous one-dimensional gradient G(x,y). If the gradient is sufficiently large above some threshold value an edge is deemed present.

The classical methods for edge detection are based on the first derivative and second derivative principle. The operators which are used to carry out the differentiation operation are called gradient operators.

If we go to the discrete domain in terms of row $G_R(j,k)$ and column gradient $G_C(j,k)$. The spatial gradient amplitude is given by:

$$G(j,k)=[[G_R(j,k)]^2 + [G_C(j,k)]^2]^{1/2}$$

For computational efficiency, the gradient amplitude is sometimes approximated by the magnitude combination

$$G(j,k)=|G_R(j,k)| + |G_C(j,k)|$$

Using newton's forward and backward difference methods for calculating the derivatives of the pixels we get,

The row gradient as

$$G_R(j,k)=F(j,k)-F(j,k-1)$$

and the column gradient is

$$G_c(j,k)=F(j,k)-F(j+1,k)$$

If we apply first derivative method again at the output obtained from differentiation then we get second derivative results.

ii) Second Derivative Principle

$G(j,k)=\delta^2/\delta x^2+\delta^2/\delta y^2$ for continuous function but as we are computing for discrete function the formulae is

$$G_R(j,k)=F(j,k)-F(j+1,k)-F(j-1,k)+F(j,k);$$
$$G_c(j,k)=F(j,k)-F(j,k-1)-F(j,k+1)+F(j,k);$$

$$G(j,k)=[[G_R(j,k)]^2+[G_C(j,k)]^2]^{1/2}$$

iii) Sobel Edge Detection Method
Gradient based edge detection methods, such as Sobel, have used two 2-D linear filters to process vertical edges and horizontal edges separately to approximate first-order derivative of pixel values of the image.

1	2	1
0	0	0
-1	-2	-1

1	0	-1
2	0	-2
1	0	-1

Fig 1: sobel row and column gradient

II. Methodology

A) In Matlab
i) Algorithm
Step 1: select an image of grayscale format.
Step 2: Sobel operator matrix is defined.
Step 3: the operator is applied to every pixel in the image and a new matrix is formed.
Step 4: the matrix obtained is the edge detected image of the given matrix.
Edge detection of an image is white line as edges of the image on the black background. Generally grayscale image lies between 0-255 where 0 represents black moving toward white. From the experience of the tested images in this study, it is found that the best result to be achieved at the range black from zero to 126 gray values and from 126 to 255 meaning is trace as white**..**
The edge detected matrix has a wide range of pixel values from 0 to 255. In order to increase the accuracy we assign a threshold value. Pixel values above the threshold are made white and below are made black. Now the matrix contains elements only of 0 and 255.

Step 5: After this, the resulting Edge detected image is passed through a network that was previously trained with the specified input output.

Next time, any input of any size or of any pixels can be sent into the network and the edge detected image can be automatically obtained.

B) In Neural Network
You can create a standard network that uses trainlm with feedforwardnet or cascadeforwardnet.
To prepare a custom network to be trained with trainlm,

i. Set net.trainFcn to 'trainlm'. This sets net.trainParam to trainlm's default parameters.
ii. Set net.trainParam properties to desired values.

In either case, calling train with the resulting network trains the network with trainlm.
Training stops when any of these conditions occur:
- The maximum number of epochs (repetitions) is reached.
- The maximum amount of time has been exceeded.
- Performance has been minimized to the goal.
- The performance gradient falls below mingrad.
- mu exceeds mu_max.

- Validation performance has increased more than max_fail times since the last time it decreased (when using validation).

C) Levenberg-Marquardt Algorithm

The Levenberg-Marquardt algorithm is a very simple, but robust, method for approximating a function. Basically, it consists in solving the equation:

$(J^tJ + \lambda I)\delta = J^tE$

Where J is the Jacobian matrix for the system, λ is the Levenberg's damping factor, δ is the weight update vector that we want to find and E is the error vector containing the output errors for each input vector used on training the network. The δ tell us by how much we should change our network weights to achieve a (possibly) better solution. The J^tJ matrix can also be known as the approximated Hessian. The λ damping factor is adjusted at each iteration, and guides the optimization process. If reduction of E is rapid, a smaller value can be used, bringing the algorithm closer to the Gauss–Newton algorithm, whereas if an iteration gives insufficient reduction in the residual, λ can be increased, giving a step closer to the gradient descent direction.

The Jacobian is a matrix of all first-order partial derivatives of a vector-valued function. In the neural network case, it is a N-by-W matrix, where N is the number of entries in our training set and W is the total number of parameters (weights + biases) of our network.

$$J = \begin{bmatrix} \frac{\partial F(x_1, w)}{\partial w_1} & \cdots & \frac{\partial F(x_1, w)}{\partial w_W} \\ \vdots & \ddots & \vdots \\ \frac{\partial F(x_N, w)}{\partial w_1} & \cdots & \frac{\partial F(x_N, w)}{\partial w_W} \end{bmatrix}.$$

Where $F(x_i, w)$ is the network function evaluated for the i-th input vector of the training set using the weight vector w and w_j is the j-th element of the weight vector w of the network.
Each learning iteration (epoch) will consist of the following basic steps:

i. Compute the Jacobian (by using finite differences or the chain rule)
ii. Compute the error gradient
$g = J^tE$
iii. Approximate the Hessian using the cross product Jacobian (eq. 3)
$H = J^tJ$
iv. Solve $(H + \lambda I)\delta = g$ to find δ
v. Update the network weights w using δ
vi. Recalculate the sum of squared errors using the updated weights
vii. If the sum of squared errors has not decreased,
viii. Discard the new weights, increase λ using v and go to step 4.
ix. Else decrease λ using v and stop.

III. Results

The results we obtained after performing the edge detection technique are as shown below

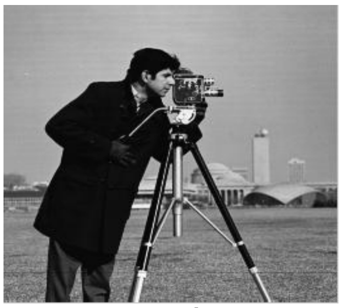
Fig 1: grayscale image

The edge detected image obtained from first derivative method is not very accurate but if a noisy image is given as an input then the output obtained is better than other techniques.

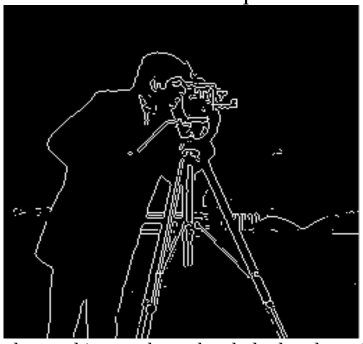
Fig 2: Edge detected image through sobel edge detection method

The image obtained from sobel edge detection technique is very accurate but when a noisy image is given as an input then the edges can’t be detected accurately.

These results are fed into a network to train using the trainlm function. After training the network, a completely different input is forced into the network and the results are as shown below.

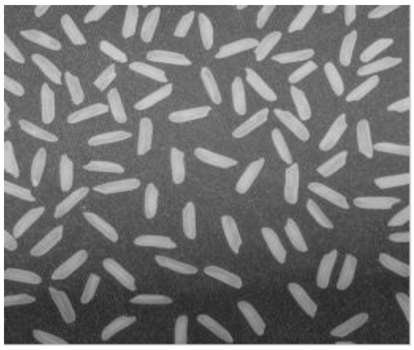
Fig 3:input to the network

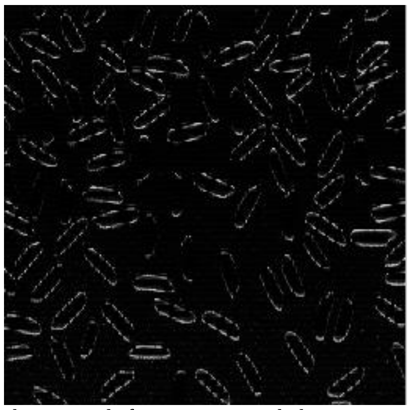
Fig 4:output from the network generated after training it with the cameraman images(input and output)

References

[1]. ZahariDoychev, edge detection and feature extraction
[2]. Mohamed A. El-Sayed, A New Algorithm Based Entropic Threshold for Edge Detection in Images
[3]. You-yiZheng, Ji-laiRao, Lei Wu, “Edge detection methods in digital image processing, “Computer Science and Education (ICCSE), 2010 5th International Conference”, page 471 – 473 on Aug. 2010 ,Ferdinand van der Heijden, Edge and Line Feature Extraction Based on Covariance Models
[4]. Gonzalez woods and Eddins, Digital image processing
[5]. Mike Boldischar and Cha PoaMoua, Edge Detection and Feature Extraction in Automated Fingerprint Identification Systems
[6]. André Aichert, Feature extraction techniques
[7]. O. Folorunso and O. R. Vincent, A Descriptive Algorithm for Sobel Image Edge Detection
[8]. Neural network toolbox in MATLAB.