Nisha Soni, Garima Mathur, Mahendra Kumar / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 4, Jul-Aug 2013, pp.785-790

A Matlab Based High Speed Face Recognition System Using Som Neural Networks

Nisha Soni[1], Garima Mathur[2], Mahendra Kumar[3]
Department of Electronics[1, 2, 3]
JEC Kukas[1, 2], Mewar University Gangrar[3]
Jaipur[1, 2], Chittorgarh[3], India

Abstract

Face recognition (FR) is a challenging issue due to variations in pose, illumination, and expression. The search results for most of the existing FR methods are satisfactory but still included irrelevant images for the target image. We have introduced a new technique uses an image-based approach towards artificial presents a new technique for human face recognition. This intelligence by removing redundant data from face images through image compression using Sobel Edge Detection (SED) and comparing this with the two-dimensional discrete cosine transform (2D-DCT) method for the better speed and efficiency. SED which is a popular edge detection method is considered in this work. There exists a function, edge.m which is in the image toolbox. In the edge function, the Sobel method uses the derivative approximation to find edges. Therefore, it returns edges at those points where the gradient of the considered image is maximum. The developed algorithm for the face recognition system formulates an image-based approach, using the Two-Dimensional Discrete Cosine Transform (2D-DCT) or SED for image compression and the Self-Organizing Map (SOM) Neural Network for recognition purposed, simulated in MATLAB.

Key Words-- *Face recognition; discrete cosine transform (DCT); sobel edge detection (SED); SOM network.*

I. INTRODUCTION

Using computers to do image processing has two objectives: First, create more suitable images for people to observe and identify. Second, we wish that computers can automatically recognize and understand images. The edge of an image is the most basic features of the image. It contains a wealth of internal information of the image. Edge detection is the process of localizing pixel intensity transitions. The edge detection has been used by object recognition, target tracking, segmentation, and etc. Therefore, the edge detection is one of the most important parts of image processing. The current image edge detection methods are mainly differential operator technique and high-pass filtration. Among these methods, the most primitive of the differential and gradient edge detection methods are complex and the effects are not satisfactory. The widely used operators such as Sobel, Prewitt, Roberts and Laplacian are sensitive to noises and their anti-noise performances are poor. The Log and Canny edge detection operators which have been proposed use Gaussian function to smooth or do convolution to the original image, but the computations are very large. This paper mainly used the Sobel operator method to do edge detection processing on the images It has been proved that the effect by using this method to do edge detection is very good and its anti-noise performance is very strong too accuracy. The Sobel edge detector uses two masks, one vertical and one horizontal. These masks are generally used 3×3 matrices. Especially, the matrices which have 3×3 dimensions are used in MATLAB (see, edge.m). The masks of the Sobel edge detection are extended to 5×5 dimensions [1], are constructed in this work. A matlab function, called as Sobel 5×5, is developed by using these new matrices. Matlab, which is a product of The Mathworks Company, contains has a lot of toolboxes. One of these toolboxes is image toolbox which has many functions and algorithms [2]. Edge function which contains several detection methods (Sobel, Prewitt, Roberts, Canny, etc) is used by the user.

The image set, which consist of 8 images (256×256), is used to test Sobel3×3 and Sobel5×5 edge detectors in Matlab.

II. THE PRINCIPLE OF EDGE DETECTION

In digital image, the so-called edge is a collection of the pixels whose gray value has a step or roof change, and it also refers to the part where the brightness of the image local area changes significantly. The gray profile in this region can generally be seen as a step. That is, in a small buffer area, a gray value rapidly changes to another whose gray value is largely different with it. Edge widely exists between objects and backgrounds, objects and objects, primitives and primitives. The edge of an object is reflected in the discontinuity of the gray. Therefore, the general method of edge detection is to study the changes of a single image pixel in a gray area, use the variation of the edge neighbouring first order or second-order to detect the edge. This method is used to refer as local operator edge

detection method. Edge detection is mainly the measurement, detection and location of the changes in image gray. Image edge is the most basic features of the image. When we observe the objects, the clearest part we see firstly is edge and line. According to the composition of the edge and line, we can know the object structure. Therefore, edge extraction is an important technique in graphics processing and feature extraction. The basic idea of edge detection is as follows: First, use edge enhancement operator to highlight the local edge of the image. Then, define the pixel "edge strength" and set the threshold to extract the edge point set. However, because of the noise and the blurring image, the edge detected may not be continuous. So, edge detection includes two contents. First is using edge operator to extract the edge point set. Second is removing some of the edge points from the edge point set, filling it with some another and linking the obtained edge point set into lines.

III. AN EDGE DETECTION SOBEL FILTER DESIGN

Compared to other edge operator, Sobel has two main advantages: 1) since the introduction of the average factor, it has some smoothing effect to the random noise of the image. 2) Because it is the differential of two rows or two columns, so the element of the edge on both sides has been enhanced, so that the edge seems thick and bright.

Most edge detection methods work on the assumption that the edge occurs where there is a discontinuity in the intensity function or a very steep intensity gradient in the image. Using this assumption, if one take the derivative of the intensity value across the image and find points where the derivative is maximum, then the edge could be located. The gradient is a vector, whose components measure how rapid pixel value are changing with distance in the *x* and *y* direction. Thus, the components of the gradient may be found using the following approximation:

$$\frac{\partial f(x,y)}{\partial x} = \Delta x = \frac{f(x+dx,y) - f(x,y)}{dx} \quad (2.1)$$

$$\frac{\partial f(x,y)}{\partial x} = \Delta y = \frac{f(x,y+dy) - f(x,y)}{dy} \quad (2.2)$$

where *dx* and *dy* measure distance along the *x* and *y* directions respectively. In discrete images, one can consider *dx* and *dy* in terms of numbers of pixel between two points. *dx* = *dy* = 1 *(*pixel spacing) is the point at which pixel coordinates are(*i*,*j*) thus,

$$\Delta x = f(i+1,j) - f(i,j) \quad (2.3)$$

$$\Delta y = f(i,j+1) - f(i,j) \quad (2.4)$$

In order to detect the presence of a gradient discontinuity, one could calculate the change in the gradient at (*i, j)* .This can be done by finding the following magnitude measure

$$M = \sqrt{\Delta x^2 + \Delta y^2} \quad (2.5)$$

and the gradient direction is given by

$$\theta = \tan^{-1}\frac{\Delta y}{\Delta x} \quad (2.6)$$

A. METHOD OF FILTER DESIGN

There are many methods of detecting edges; the majority of different methods may be grouped into these two categories:

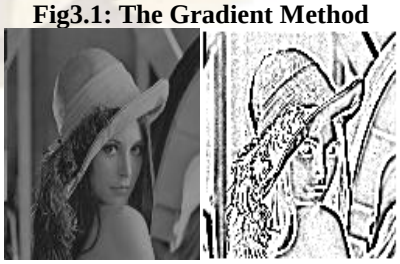
i. **Gradient:** The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. For example Roberts, Prewitt, Sobel where detected features have very sharp edges. (see Figure 1)

ii. **Laplacian:** The Laplacian method searches for zero crossings in the second derivative of the image to find edges e.g. Marr-Hildreth, Laplacian of Gaussian *etc*. An edge has one dimensional shape of a ramp and calculating the derivative of the image can highlight its location (see Figure 2).

Edges may be viewpoint dependent: these are edges that may change as the viewpoint changes and typically reflect the geometry of the scene which in turn reflects the properties of the viewed objects such as surface markings and surface shape. A typical edge might be the border between a block of red colour and a block of yellow, in contrast. However, what happens when one looks at the pixels of that image is that all visible portion of one edge are compacted.

Input Image Output Edges

Fig3.1: The Gradient Method



Input Image Output Edges

Fig 3.2: The Laplacian Method

The Sobel operator is a discrete differentiation operator which computes an approximation of the gradient of the image intensity function (Sobel, 1968). The different operators in

eq. (2.3) and (2.4) correspond to convolving the image with the following masks

$$\Delta x = \begin{bmatrix} -1 & 1 \\ 0 & 0 \end{bmatrix}, \quad \Delta y = \begin{bmatrix} -1 & 0 \\ 1 & 0 \end{bmatrix} \qquad (3.1)$$

When this is done, then

i. The top left-hand corner of the appropriate mask is super-imposed over each pixel of the image in turn

ii. A value is calculated for Δx or Δy by using the mask coefficients in a weighted sum of the value of pixels *i*, *j* and its neighbours

iii. These masks are referred to as convolution masks or sometimes convolution kernels. Instead of finding approximate gradient components along the *x* and *y* directions**,** approximation of the gradient components could be done along directions at 45° and 135°to the axes respectively.

iv. An advantage of using a larger mask size is that the errors due to the effects of noise are reduced by local averaging within the neighbourhood of the mask. An advantage of using a mask of odd size is that the operators are centred and can therefore provide an estimate that is based on a centre pixel *(i,j)*. The Sobel edge operator masks are given as

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad (3.2)$$

$$\Delta y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \qquad (3.3)$$

The operator calculates the gradient of the image intensity at every point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. Therefore the result shows how "abruptly" or "smoothly" the image changes at that point and therefore how likely it is that part of the image represents an edge and how that the edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation. The gradient of a two-variable function (the image intensity function) at each image point is a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each point of image, the gradient vector points to the direction of largest possible increases the intensity and the length of the gradient vector corresponds to the rate of change in that direction which results in Sobel operator at any image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values. The algorithm for developing the Sobel model for edge detection is given below.

PSEDO CODES

Input: A Sample Image
Output: Detected Edges
Step 1: Accept the input image
Step 2: Apply mask G_x, G_y to the input image
Step 3: Apply Sobel edge detection algorithm and the gradient
Step 4: Masks manipulation of G_x, G_y separately on the input image
Step 5: Results combined to find the absolute magnitude of the gradient

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad (3.4)$$

Step 6: the absolute magnitude is the output edges

Fig 3.3: Detected Image

IV. SELF-ORGANIZING MAPS

A. *OVERVIEW*

The self-organizing map also known as a Kohonen Map is a well-known artificial neural network. It is an unsupervised learning process, which learns the distribution of a set of patterns without any class information. It has the property of topology preservation. There is a competition among the neurons to be activated or fired. The result is that only one neuron that wins the competition is fired and is called the "winner"[5]. A SOM network identifies a winning neuron using the same procedure as employed by a competitive layer. However, instead of updating only the winning neuron, all neurons within a certain neighborhood of the winning neuron are updated using the Kohonen Rule. The Kohonen rule allows the weights of a neuron to learn an input vector, and because of this it is useful in recognition applications. Hence, in this system, a SOM is employed to classify DCT-based vectors into groups to identify if the subject in the input image is "present" or "not present" in the image database [3].

B. *NETWORK ARCHITECTURE*

SOMs can be one-dimensional, two-dimensional or multi-dimensional maps. The number of input connections in a SOM network depends on the number of attributes to be used in the classification [4].

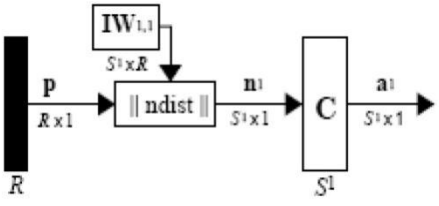


Fig. 4 Architecture of a simple SOM

The input vector p shown in Fig. 4 is the row of pixels of the input compressed image. The ||dist|| box accepts the input vector p and the input weight matrix $IW^{1,1}$, which produces a vector having S^1 elements. The elements are the negative of the distances between the input vector and vectors $_iIW^{1,1}$ formed from the rows of the input weight matrix. The ||dist|| box computes the net input n^1 of a competitive layer by finding the Euclidean distance between input vector p and the weight vectors. The competitive transfer function C accepts a net input vector for a layer and returns neuron outputs of 0 for all neurons except for the winner, the neuron associated with the most positive element of net input n^1. The winner's output is 1. The neuron whose weight vector is closest to the input vector has the least negative net input and, therefore, wins the competition to output a 1. Thus the competitive transfer function C produces a 1 for output element a^1_i corresponding to i^*, the "winner". All other output elements in a^1 are 0[6].

$$n^1 = - \ ||IW_{11} - p|| \quad (4.1)$$

$$a^1 = \text{compet}(n^1) \quad (4.2)$$

Thus, when a vector p is presented, the weights of the winning neuron and its close neighbours move toward p. Consequently, after many presentations, neighbouring neurons learn vectors similar to each other[6]. Hence, the SOM network learns to categorize the input vectors it sees.

The SOM network used here contains N nodes ordered in a two-dimensional lattice structure. In these cases, each node has 2 or 4 neighboring nodes, respectively. Typically, a SOM has a life cycle of three phases: the learning phase, the training phase and the testing phase.

C. *UNSUPERVISED LEARNING*

During the learning phase, the neuron with weights closest to the input data vector is declared as the winner. Then weights of all of the neurons in the neighborhood of the winning neuron are adjusted by an amount inversely proportional to the Euclidean distance. It clusters and classifies the data set based on the set of attributes used. The learning algorithm is summarized as follows [4]:

1. Initialization: Choose random values for the initial weight vectors $w_j(0)$, the weight vectors being different for j = 1,2,...,l where l is the total number of neurons.

$$w_i = [w_{i1}, w_{i2}, ..., w_{il}]^T \in \Re^n \quad (4.3)$$

2. Sampling: Draw a sample x from the input space with a certain probability.

$$x=[x_1,x_2,...,x_l]^T \in \Re^n \quad (4.4)$$

3. Similarity Matching: Find the best matching (winning) neuron i(x) at time t, $0 < t \leq n$ by using the minimum distance Euclidean criterion:

$$i(x)=\arg\min_j \left\| x(n) - w_j \right|_{j=1,2...,l} \quad (4.5)$$

4. Updating: Adjust the synaptic weight vector of all neurons by using the update formula:

$$w_j(n+1)=w_j(n)+\eta(n)h_{j,i(x)}(n)(x(n)-w_j(n)) \quad (4.6)$$

where $\eta(n)$ is the learning rate parameter, and $h_{j,i}x(n)$ is the neighborhood function centered around the winning neuron i(x). Both $\eta(n)$ and $h_{j,i}x(n)$ are varied dynamically during learning for best results.

5. Continue with step 2 until no noticeable changes in the feature map are observed.

Training images are mapped into a lower dimension using the SOM network and the weight matrix of each image stored in the training database. During recognition trained images are reconstructed using weight matrices and recognition is through untrained test images using Euclidean distance as the similarity measure. Training and testing for our system was performed using the MATLAB Neural Network Toolbox.

D. *TRAINING*

During the training phase, labeled Sobel-vectors are presented to the SOM one at a time. For each node, the number of "wins" is recorded along with the label of the input sample. The weight vectors for the nodes are updated as described in the learning phase. By the end of this stage, each node of the SOM has two recorded values: the total number of winning times for subject present in image database, and the total number of winning times for subject not present in image database [2].

E) *TESTING*

During the testing phase, each input vector is compared with all nodes of the SOM, and the best match is found based on minimum Euclidean distance, as given in (4.5)[4]. The final output of the system based on its recognition, displays if the test image is "present" or "not present" in the image database.

V. EXPERIMENTAL RESULTS

A. IMAGE DATABASE

A face image database was created for the purpose of benchmarking the face recognition system. The image database is divided into two subsets, for separate training and testing purposes. During SOM training, 30 images were used, containing six subjects and each subject having 5 images with different facial expressions. Fig. 5 shows the training and testing image database constructed.

(a)

(b)

Fig. 5.1 Image database for training: (a) Image database for training.(b) Untrained image for testing

The face recognition system presented in this paper was developed, trained, and tested using MATLAB™ 7.5. The computer was a **Windows** 8 with a 2.40GHz Intel(R) core(TM) i3-3110M processor and 2 GB of RAM.

B. *VALIDATION of TECHNIQUE*

The pre-processed grayscale images of size 8 × 8 pixels are reshaped in MATLAB to form a 64 × 1 array with 64 rows and 1 column for each image. This technique is performed on all 5 test images to form the input data for testing the recognition system. Similarly, the image database for training uses 30 images and forms a matrix of 64 × 30 with 64 rows and 30 columns. The input vectors defined for the SOM are distributed over a 2D-input space varying over [0 255], which represents intensity levels of the gray scale pixels. These are used to train the SOM with dimensions [64 2], where 64 minimum and 64 maximum values of the pixel intensities are represented for each image sample. The resulting SOM created with these parameters is a single-layer feed forward SOM map with 128 weights and a competitive transfer function. The weight function of this network is the negative of the Euclidean distance[3]. As many as 5 test images are used with the image database for performing the experiments. Training and testing sets were used without any overlapping. Fig. 5.3 shows the result of the face recognition system simulated in MATLAB using the image database and test input image shown in Fig. 5.1.

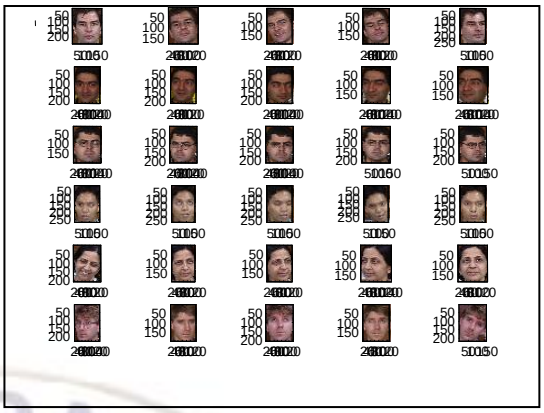
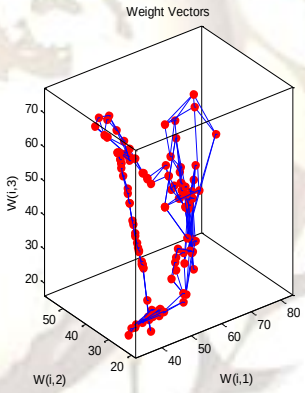
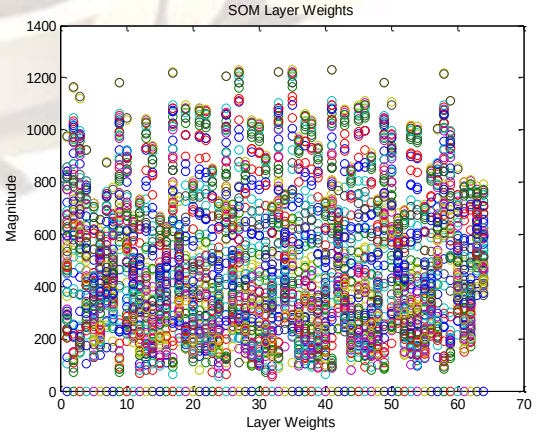


Fig.5.2 Training & testing image database

Fig. 5.3 Result of face recognition system. (a) Untrained input image for testing. (b) Best match image of subject found in training database





(b)

Fig.5.4 (a) Weight vector graph (b) SOM layer weight graph

Table 5.1: Comparison for 5 images B/W DCT & SED based face recognition system

Time	NO.of epoch	No.of Image	DCT	SED
Training time	200	1	10.2594	Unmatched
Execution time			27.9148	
Training time		2	10.1553	10.1383
Execution time			36.0948	25.0202
Training time		3	10.3623	9.9243
Execution time			26.3857	29.4493
Training time		4	10.3039	10.0291
Execution time			36.8931	29.0859
Training time		5	10.2635	10.0649
Execution time			24.2630	12.0557
Recognition rate			100%	83.33%

This table shows that training and execution time through SED is less then DCT but the recognition rate provided by DCT is 100%.

Table 5.2: Comparison for one image at different epoch B/W DCT & SED based face recognition system

Image	**Epoch**	**Method ⇨ Time ⇩**	**DCT**	**Sobel**
S01_01	100	Training time	5.9940	4.6029
		Execution time	115.7445	14.0523
S01_01	500	Training time	24.8952	24.4737
		Execution time	43.3684	43.3329
S01_01	1000	Training time	46.2868	45.6684
		Execution time	59.6358	68.7433

This table shows that SOBEL is faster then DCT.

VI. CONCLUSION

This paper met all objectives by the successful design of an efficient high-speed face recognition system. The SED in MATLAB was successful and all face images were successfully compressed to the desired size and quality and Self-Organizing Maps (SOM's) which proved to be highly accurate for recognizing a variety of face images with different facial expressions under uniform light conditions with light backgrounds. Hence as a conclusion, the SED and the SOM neural network are the heart for the design and implementation, which are the final algorithms used for the design of an efficient high-speed face recognition system.

References

[1] D. Kumar, C.S. Rai, and S. Kumar, "Face Recognition using Self- Organizing Map and Principal Component Analysis" in Proc. on Neural Networks and Brain, ICNNB 2005, Vol. 3, Oct 2005, pp. 1469-1473.

[2] Y. Zi Lu and Z. You Wei, "Facial Expression Recognition Based on Wavelet Transform and MLP Neural Network", in Proc. 7th International Conference on Signal Processing, ICSP 2004, Vol. 2, Aug 2004, pp. 1340-1343.

[3] AYBAR, E., "Topolojik Kenar _slecleri", Anadolu Üniversitesi, Fen Bilimleri Enstitüsü, Ph.D. thesis, **2003**.

[4] Image Toolbox (for use with Matlab) User's Guide, The MathWorks Inc., **2000**.

[5] J. Nagi, "Design of an Efficient High-speed Face Recognition System",Department of Electrical and Electronics Engineering, College of Engineering, Universiti Tenaga Nasional, March 2007.

[6] A. Abdallah, M. Abou El-Nasr, and A. Lynn Abbott, "A New Face Detection Technique using 2D DCT and Self Organizing Feature Map" in Proc. of World Academy of Science, Engineering and Technology,Vol. 21, May 2007, pp. 15-19.

About auther

Nisha Soni was born on 7th November 1984 at Bhanpura, madhyapradesh (India). She has completed Bachelor Engineering Degree in Electronics & Communication from M.L.V. Textile & Engineering College , Bhilwara, University of Rajasthan, in 2007. She has registered for M. Tech (Full Time) in Digital Communication branch at Electronics Engineering Department, under the guidance of Mrs. Garima Mathur in JEC, kukas(Jaipur) under Rajasthan Technical University (Kota) .

.