International Journal of Advance Research in Science and Engineering
Volume No.07, Special Issue No.01, February 2018
www.ijarse.com
IJARSE
ISSN: 2319-8354

OPTICAL CHARACTER RECOGNITION USING MODIFIED GABOR FILTER

Sandhya Balakrishnan P K[1], Dr.L.Pavithira[2]

[1]*Research Scholar, Department of Computer Science,*
CMS College of Science and Commerce, Coimbatore.
[2]*Associate Professor, Department of Computer Science,*
CMS College of Science and Commerce, Coimbatore.

ABSTRACT

Optical character recognition (OCR) is a very active field for research and development, and has become one of the most successful applications of automatic pattern recognition. Feature extraction is a key step in the process of OCR, which in fact is a deciding factor of the accuracy of the system. This paper proposes a novel and robust technique for feature extraction using Modified Gabor Filters, to be employed in the OCR. The use of 2D Gabor filters is investigated and features are extracted using these filters. The technique generally extracts fifty features based on global texture analysis and can be further extended to increase the number of features if necessary. The proposed algorithm using neural network classification provides a high accuracy rate in recognition of characters. The proposed approach is implemented and tested on isolated character database consisting of English characters, digits and keyboard special characters.

Keywords: OCR, Gabor Filters, Neural network, pattern recognition.

I. INTRODUCTION

In the past few years, optical character recognition (OCR) systems have been developed maturely in both academic research and industrial applications. They are also successfully applied to vision-based inspection and handwriting recognition for input interface. It has been proven that high recognition rates can be achieved in specific application scenarios using some standard and well-studied methods such as neural network, support vector machine (SVM), etc. However, since their primary focus is usually fast or real-time recognition in the fairly controlled environments, two major issues related to the system's adaptivity are not fully investigated in the literature.

For the development of fast online processing techniques which give the results with high recognition rates, the price to pay is usually the considerable amount of training time associated with the algorithms. Most recognition systems perform one time training, and use it for multiple recognition tasks [1], [2]. Thus, it is worth spending the long training time to achieve the high recognition rates. However, if different training data have to be adopted for different kind of applications, the long training time required for the recognition algorithms clearly

poses a serious problem. In these cases, it is highly desirable to have a fast training approach for the OCR technique to cope with various types of source for recognition.

The paper is divided into five sections thus: Introduction, Pre-processing, Modified gabor filter feature extraction, Back propagation Neural network classification and conclusion.

II. PREPROCESSING

The process of enhancing the image, which should be used for further processing, is called preprocessing. Preprocessing is the major step in handwriting recognition system. Noise in a document image is due to poorly photocopied pages. Median Filtering [3], Wiener Filtering method [4] and morphological operations can be performed to remove noise [5]. Median filters are used to replace the intensity of the character image [6], Where as Gaussian filters can be used to smoothing the image [7]. The initial images which we tend to use contain some information which is not necessary for the next step which is feature extraction. The images which are scanned may also contain noises.

The scanned images not only have noises which are inbuilt within it, but also the noise may be during the scanning of that image. So the noises and the unwanted information should be removed from the image.

Preprocessing is not the single step rather it contains sequences of steps. They are

- Binarization
- Normalization
- Sampling
- Denoising
- Thinning

A. Binarization

The conversion of the grayscale image to black and white is called binarization. Binary images are also called as Bi-level or two-level. First, the original RGB image should be converted to grayscale and then the image is converted to black and white image. Most of the OCR packages work on the binarized images.

The conversion is possible because of the threshold values and the values which are higher than the threshold are white and the values which are lower than this threshold are black. Otsu's method is used to perform threshold based on cluster i.e. from gray level image to binary image.

The threshold value 0.5 yields better for all type of images. The laser printer, fax machines can handle the binarized images.

B. Normalization

The process of changing the intensity value of the pixel to the range of [0,1] is called normalization in image processing. The conversion of various dimension images into fixed dimensions is also called as normalization. Normalization is used in digital signal processing [8]. The matrix values of the image can be normalized along the column and row using the normc and normr commands in Matlab. Further, complications during feature extraction are removed if normalization is done in the earlier stages.

C. Sampling

Discretization of analog signal is called sampling. The smallest element which is the result of discretization of the space is called pixel. The process of selecting the subset of individuals from the large sample of population and examining those samples, can generalize the results to the whole population.

D. Denoising

Digital images are prone to a variety of types of noise. There are several ways that noise can be introduced into an image. For example:

• If there is scanned image then there will be the noise.

• During the electronic transmission of image, the noise can be introduced.

E. Thinning

Thinning is a pre-process which results in single pixel width image to recognize the handwritten character easily. It is applied repeatedly leaving only pixel-wide linear representations of the image characters [1].

III. FEATURE EXTRACTION

Gabor filters have been used extensively in image processing, texture analysis for their excellent properties: frequency and orientation representation of Gabor filters are similar to those of the human visual system, and they have been found to be particularly appropriate for texture representation and discrimination.

A Gabor Filter is a linear filter whose impulse response is defined by a harmonic function multiplied by a Gaussian function.

$$h(x, y) = g(x, y)\, s(x, y)$$

Where s(x, y) is a complex sinusoid, known as carrier and g(x, y) is a Gaussian shaped function, known as envelope. The Gabor filters are self similar, i.e. all filters can be generated from one mother wavelet by dilation and rotation.

Thus the 2-D Gabor filter with the response in spatial domain is given by Eq. (1) and in spatial-frequency domain is given by Eq.(2). Since Gaussian Function is a complex function so on convolving Gabor Filter with input image the output obtained can be used in various ways. Two of ways of manipulating the output of Gabor Filter to extract features are described below.

$$h(x, y; \lambda, \phi, \sigma_x, \sigma_y) = \frac{1}{2\pi\,\sigma_x\,\sigma_y} \exp\left\{-\frac{1}{2}\left[\frac{R_1^2}{\sigma_x^2} + \frac{R_2^2}{\sigma_y^2}\right]\right\} \times \exp\left[i \cdot \frac{2\pi R_1}{\lambda}\right] \quad (1)$$

Where

$$R_1 = x\cos\phi + y\sin\phi,$$
$$R_2 = -x\sin\phi + y\cos\phi.$$

$$h(u, v; \lambda, \phi, \sigma_x, \sigma_y) = C \exp\left\{-2\pi^2\left(\sigma_x^2\left(F_1 - \frac{1}{\lambda}\right)^2 + \sigma_y^2 (F_2)^2\right)\right\}. \quad (2)$$

Where

$$F_1 = u\cos\phi + c\sin\phi\ ,$$
$$F_2 = -u\sin\phi + v\cos\phi.$$
$$C = const.$$

The other form of 2-D Gabor Filter in terms of frequency can be represented as:

$$h_{x,y,\theta,f} = e^{-\frac{1}{2}\left(\frac{x'^2}{\sigma_x^2}+\frac{y'^2}{\sigma_y^2}\right)}.e^{i2\pi fx} \quad (3)$$

Where σ_x and σ_y explain the spatial spread and are the standard deviations of the Gaussian envelope along x and y directions. x' and y' are the x and y co-ordinates in the rotated rectangular co-ordinate system given as:

$$x' = x\cos\theta + y\sin\theta$$
$$y' = y\cos\theta - x\sin\theta$$

Any combination of θ and f, involves two filters, one corresponding to sine function and other corresponding to cosine function in exponential term in Eq. (3). The cosine filter, also known as the real part of the filter function, is an even-symmetric filter and acts like a low pass filter, while the sine part being odd-symmetric acts like a high pass filter.

Gabor filters having Spatial frequency (f = 0.0625, 0.125, 0.25, 0.5, 1.0) and orientation (θ =nπ/6) where n varies in the range 0 to 6, have been used in our work.

IV. CLASSIFICATION

The back propagation algorithm trains a given feed-forward multilayer neural network for a given set of input patterns with known classifications. When each entry of the sample set is presented to the network, the network examines its output response to the sample input pattern. Back propagation used to train neural networks, used along with an optimization routine such as gradient descent. Gradient descent requires access to the gradient of the loss function with respect to all the weights in the network to perform a weight update, in order to minimize the loss function. Back propagation computes these gradients in a systematic way.

Rojas [2005] claimed that BP algorithm could be broken down to four main steps. After choosing the weights of the network randomly, the back propagation algorithm is used to compute the necessary corrections. The algorithm can be decomposed in the following four steps:

i) Feed-forward computation
ii) Back propagation to the output layer
iii) Back propagation to the hidden layer
iv) Weight updates

The algorithm is stopped when the value of the error function has become sufficiently small.

The back propagation algorithm cycles through two distinct passes, a forward pass followed by a backward pass through the layers of the network. The algorithm alternates between these passes several times as it scans the training data.

Forward Pass: Computation of outputs of all the neurons in the network

- The algorithm starts with the first hidden layer using as input values the independent variables of a case from the training data set.
- The neuron outputs are computed for all neurons in the first hidden layer by performing the relevant sum and activation function evaluations.
- These outputs are the inputs for neurons in the second hidden layer. Again the relevant sum and activation function calculations are performed to compute the outputs of second layer neurons.

Backward pass: Propagation of error and adjustment of weights

- This phase begins with the computation of error at each neuron in the output layer. A popular error function is the squared difference between ok the output of node k and yk the target value for that node.
- The target value is just 1 for the output node corresponding to the class of the exemplar and zero for other output nodes.
- The new value of the weight wjk of the connection from node j to node k is given by: $wnewjk = woldjk + \eta oj\delta k$. Here η is an important tuning parameter that is chosen by trial and error by repeated runs on the training data. Typical values for η are in the range 0.1 to 0.9.
- The backward propagation of weight adjustments along these lines continues until we reach the input layer.
- At this time we have a new set of weights on which we can make a new forward pass when presented with a training data observation

BP-NN Algorithm steps

1. Let N be a neural network with e connections.

2. Below, $x, x_1, x_2, \ldots$ will denote vectors in $\mathbb{R}^m$, y, $y^1, y_1, y_2, \ldots$ vectors in $\mathbb{R}^n$, and vectors $w, w_1, w_2, \ldots$ in $\mathbb{R}^e$.

   These are called *inputs*, *outputs* and *weights* respectively.

3. The neural network corresponds to a function $y = f_N(w, x)$ which, given a weight *w*, maps an input *z* to an

   output *y*.

4. The backpropagation algorithm takes as input a sequence of *training examples* $(x_1, y_1), \ldots, (x_p, y_p)$ and produces a sequence of weights $w_0, w_1, \ldots, w_p$ starting from some initial weight W_0, usually chosen at random.

5. These weights are computed in turn: first compute w_i using only (x_i, y_i, w_{i-1}) for $i = 1, \ldots, p$. The output of the backpropagation algorithm is then w_p, giving us a new function $x \to f_N(w_p, x)$. The computation is the same in each step, hence only the case $i = 1$ is described.

6. Calculating w_1 from (x_1, y_1, w_0) is done by considering a variable weight w and applying gradient descent to the function $w \to E(f_N(w, x_1), y_1)$ to find a local minimum, starting at $w = w_0$. This makes w_1 the minimizing weight found by gradient descent.

V. IMPLEMENTATION AND EXPERIMENTS

The proposed optical character recognition system with fast training neural network has been tested on the images captured in the practical applications. Figure shows some typical examples of the collected image dataset. It includes standard characters, dotted font, variable spacing, tilted capture, non uniform illumination, contaminated source, and curved printing. Since our neural network training and recognition is single character based, basic image processing techniques need to be performed on the source image for individual character extraction.

Number of Images	Zoning Feature	Modified Gabor filter
30	79	89
60	83	91
90	86	94
120	84	92
150	82	95

Table I. Recognition rate comparison

Table I tabulates the overall recognition rate using the proposed technique for different types of input images.

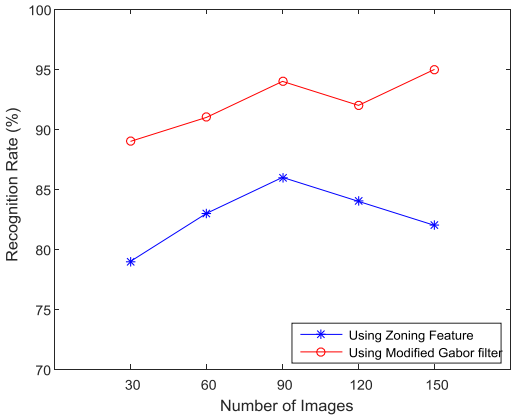

Figure 1. Comparison of recognition rate for zoning feature vs Modified gabor filter

In figure.1, shows that the recognition rate of the proposed modified gabor filter with back propagation neural network. The performance is measured by varying the number of sample images and compares the proposed method with our previous work zoning feature extraction with ANFIS classification.

V. CONCLUSION

In this work, we have presented an optical character recognition system with fast training neural network. A preprocessing stage is used to partition the training data to smaller subsets based on the modified gabor filter. The neural network training time can be reduced dramatically at the cost of longer recognition time. In our experiment, the recognition rate of our technique is higher compared to the conventional neural network approach. Thus, it is suitable for the applications with limited training time constraint.

REFERENCES

[1] M. Ganis, C. Wilson, and J. Blue, "Neural network-based systems for handprint ocr applications," IEEE Transactions on Image Processing, vol. 7, no. 8, pp. 1097 –1112, aug 1998.

[2] L.-O. Fedorovici, E. Voisan, F. Dragan, and D. Iercan, "Improved neural network ocr based on preprocessed blob classes," in 2010 International Joint Conference on Computational Cybernetics and Technical Informatics, May 2010, pp. 559 –564.

[3] Jagadeesh Kumar R, Prabhakar R and Suresh R.M, "Off-line Cursive Handwritten Tamil Characters Recognition", International Conference on Security Technology, page(s): 159 – 164, 2008.

[4] Anil.K.Jain and Torfinn Taxt, "Feature extraction methods for character recognition-A Survey," Pattern Recognition, vol. 29, no. 4, pp. 641 - 662, 1996.

[5] Shanthi N and Duraiswami K, "A Novel SVM -based Handwritten Tamil character recognition system", springer, Pattern Analysis & Applications,Vol-13, No. 2, 173-180,2010.

[6] Sutha J and RamaRaj N, "Neural network based offline Tamil handwritten character recognition System", International Conference on Conference on Computational Intelligence and Multimedia Vol : 2, page(s): 446 – 450, 2007.

[7] L. R. B. Schomaker, H. L. Teulings, .A handwriting recognition system based on the properties and architectures of the human motor system., Proceedings of the IWFHR, CENPARMI, Concordia, Montreal, 1990, pp 195-211.

[8] Normalization http://en.wikipedia.org/wiki/Normalization_(image_processing)