FEATURE EXTRACTION USING CONTOUR PROJECTION

ROBERTO J. RODRIGUES
AEP/NCE - Núcleo de Computação Eletrônica
UFRJ, Caixa Postal 2324, Ilha do Fundão, Rio de Janeiro, RJ, Brasil
EUGÊNIO SILVA
IME - Instituto Militar de Engenharia, DE9 - Departamento de Engenharia de Sistemas,
Praça Gen. Tibúrcio, 80, Praia Vermelha, Rio de Janeiro, RJ, Brasil
ANTONIO C. G. THOMÉ
AEP/NCE- Núcleo de Computação Eletrônica/
UFRJ, Caixa Postal 2324, Ilha do Fundão, Rio de Janeiro, RJ, Brasil

ABSTRACT

This paper describes the set of evaluation tests performed over the feature extraction technique proposed by the authors in [11]. As explained in [11], the technique was mainly developed to deal with the problem of cursive character recognition. It is based on projection of the image contour over the edges of a surrounding regular polygon placed around each character. The surrounding polygon can be of any number of sides, from triangle up to the circle. The feature vector is formed by the perpendicular distance taken from each edge of the polygon up to the contour of the image. The evaluation under this investigation compares the proposed approach using two different polygons: square and hexagon, and five different number of projection lines taken over each edge, against two versions of bitmap coding, one taken with the original image and other after its thinning. The power of discrimination of each case is examined through the usage of a MLP model of neural network. Despite its ability to discriminate the patterns, the proposed method is also good in dealing with scale, rotation and translation problems.

Keywords: Feature Extraction, Character Recognition, Pattern Recognition, Neural Networks.

INTRODUCTION

Cursive character recognition is still an open research area and a very hard and complex task due to the individual characteristics related to the cursive written and the substantial amount of parameters that must be considered. The performance of an automatic recognition system depends deeply on the quality of the original image and its digitalization. To minimize the effects due to a poor quality of the data, it is very common to use some kind of image compensation technique. These techniques include contour enhancement, line and underline removal, noise removal, and others. The most common problems related to the quality and difficulties of a text-based image processing are noise, distortion, translation, rotation, style variation, scale, texture and trace.

The recognition process of manuscript texts, that includes both types: cursive and typed material, normally takes several steps that goes from an ordinary analysis of the individual characters up to the utilization of lexical or contextual information. These last procedures are used to validate the text as a whole. In general, the very first steps include digitalization, capture the target, extraction and segmentation of the image into individual characters. From those, capture and segmentation are fundamental for the entire process and, in general, they are very complex and difficult to implement. Recognition comes after segmentation and normally it is done on the basis of individual character's image.

The target capture may be relatively easy task if the target position is previously known or very tough if the target position not only is unknown but also is hard to be found due to the image background and noise. Segmentation, on the other hand, deals with problems like background noise, slanted, underlined and connected characters.

Feature extraction is another important phase; some of the known techniques are based on the digits themselves through some kind of image processing, such as thinning or skeleton algorithms. There are procedures that generate features vectors, based on the direction of the segment lines, ending points, intersection points and cycles. A validation procedure is commonly used after recognition and may be based on some kind of contextual libraries.

Figure 1: General diagram

One of the currently most used segmentation approaches is based on the extraction of lines and points for a later contour analysis. However, this method shows itself to be too complex and requires extremely interdependent parameterization. The relevance of such technique remains on the fact that it is very tolerant to distortion and style variation.

The investigation described in this paper has its main focus on the task of feature extraction and coding for the problem of cursive character recognition. Such step, as showed in the figure 1, takes part in the recognition system that has been built in the UFRJ's Computational Intelligence Laboratory.

The characters to be recognized are gotten from a manually filled form. The written style is free but in typed format and the target information is restricted to be placed inside a preprinted grid. However, it is true that many writers do not follow the grid and, in general one can find several parts of the characters out of the expected place.

Form recognition applications can be classified as an off-line system, at least in the general case, and because of this there is no strong pressure on the system response time. Despite this characteristic, it is desired that any recognition algorithm may be as much accurate as possible within a reasonable response time and computational effort. It is then fundamental to take part in the definition of some of the form particularities or at least to pass to the system as much information or previous knowledge as possible, such as: the colors used in the form, mainly those used for the background and the grid lines, the quality of the printed material, the features of the scanner used as the capture device and the CCD configuration of the scanner. It is a mistake to assume that a good recognition algorithm can be sufficient to guarantee the success of the recognition process. If the target location and extraction do not work well, then all the process will not do a good job too.

CAPTURE AND SEGMENTATION

The capture device used in this experiment was a simple 300 dpi scanner (commercial default resolution) that was able to digitalize the source image with a resolution of 200 dpi using color mode. The scanned image was then compressed with the LZW algorithm and saved in TIFF format.

All interesting fields in the form (figure 2) needed to be part of a previously constructed database, from where the program used to store and retrieve the information from the form was able to get the precise location of each one. After the extraction of those fields, the segmentation process is then started.

Figure 2: Typical form

The segmentation method used in this investigation is based on the use of profile projection histograms, as described in [1]. Profile projection histogram represents a structure that stores the result of the image projection over one of the two existing dimensions. Each dimension assigns a vector where it is stored the number of pixels that present energy above a predefined threshold (usually established from the background color). A set of successive refinements is then applied on the obtained vector, up to a satisfactory segmentation is achieved.

The approach can be seen as a sequence of 3 distinct stages: it starts with an image quality compensation, followed by an initial segmentation and then a successive set of refinements.

Image quality compensation step is used to improve the quality of the scanned image, reducing or enhancing details as noises and contrasts.

Initial segmentation is the step where the image is initially segmented and is based on the projection histogram [1]. The algorithm is very fast and all disconnected characters are successfully segmented within this stage. Dots and lines can be removed using the data retrieved from the horizontal histogram vector. The last product generated by the initial segmentation is the identification of those possible connected character images (figure 3).

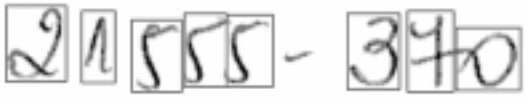

Figure 3: Highlighted digits segmented from a extracted field

The refinement stage may have many successive steps but it applies as a first step the same histogram strategy used in the initial stage. The difference relies only on the usage of a new set of refinement parameters. This procedure allows segmenting the characters weakly connected. The following stage steps include the utilization of other segmentation techniques. The results achieved with the conducted experiments showed to be possible to get an accuracy around 95.20% using only the first level of refinement.

FEATURE EXTRACTION AND CODING

The implementation of a character recognition procedure through neural networks requires the construction of input vectors taken from each source character image. Pattern discriminative ability and dimensionality reductions are two relevant problems to work with at this point.

The feature extraction method as described in [11] represents a novel approach derivated from the ordinary contour detection type methods. The feature vectors in the proposed method are based on the computation of a set of distances from the image contour up to a reference polygon [11].

The reference polygon has to be regular but can have any number of sides. It must be placed surrounding the image as in the figure 4. Using the provided set of formulas, Equations (1, 2 3 and 4), one can build polygons of any number of sides.

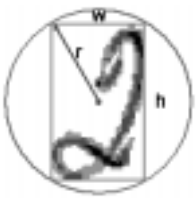

Figure 4a: circumscriptive circle around the image

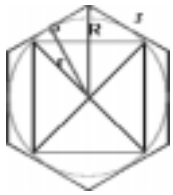

Figure 4b: a circumscriptive polygon around the circle

$$r = \frac{1}{2}\sqrt{h^2 + w^2} \quad (1)$$

$$R = \frac{r}{\cos\left(\frac{\pi}{n}\right)} \quad (2)$$

$$R = \frac{1}{2}\sqrt{h^2 + w^2} \bullet \cos(\pi/n)^{-1} \quad (3)$$

$$s = \sqrt{h^2 + w^2} \bullet \tan(\pi/n) \quad (4)$$

Where,
r = circumscriptive circle radius
(one half of the image diagonal)
h = height of the image
w = width of the image
n = number of sides of the polygon
R = circumscriptive circle radius
(referenced to the hexagon), side of the hexagon
s = size of the polygon edge.

The basic process takes the distance from each polygon side to the image contour and stores them into a vector that also contains the number of sides and the number of measured points in each side. As seen in figure 4, the number of extracted features is not the same for different polygons, despite the desired size of the side or the desired number of features. The computed and used number of features is always close to the desired one but may not be equal. This happens because of the difference in terms of the real size of the edges for the chosen polygons. The difference in fact is always small and not relevant for the problem.

Using the trick of reconstructing the original image from the extracted feature vector (based on a reverse algorithm) (figure 7) we were able to realize that the fidelity was not quite good for those odd sided polygons as it is for the even ones.

It was then decided to concentrate the investigation over the square (figure 5) and the hexagon (figure 6) and then, compare the performance of those against two types of bitmap matrix (figure 8): a full bitmap matrix and a thinned version of it. The thinning algorithm was taken from [10].

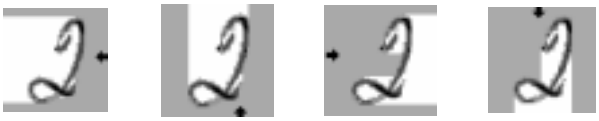

Figure 5: sequence of square scanning procedure

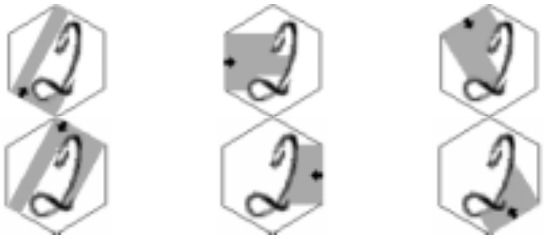

Figure 6: sequence of hexagon scanning procedure

Figure 7: final result reconstructed.

Figure 8: Bitmap (16x16 pixels); Original and thinned.

As said before, the input vector has a predefined dimension and the features are the distances computed in number of pixels and taken from each side of the polygon to the contour of the image.

This method not only proved to provide good discrimination power but also showed to mitigate the problem related to scale, rotation and translation of the source image. Once the bitmap is initially centralized inside a square before being surrounded by the desired polygon, the translation problem is automatic eliminated. The application of an interpolation scheme while taking the points for the contour distance solves the scale solution problem, and the rotation issue can be tackled just performing a sequence of right or left shift rotate on the coding vector.

EVALUATION ENVIRONMENT

The database used to evaluate the proposed technique was obtained from prepared forms (figure 2) that were filled up by several persons inside campus. In order to improve the scenario diversity it was chosen both male and female persons with different scholarship level. In each form the custom was asked to provide several numeric and alphanumeric fields like name, address, age, telephone number, zip code and others, in a free style cursive written. The forms were then scanned using a color scanner and the images were taken with a resolution of 200 dpi. After the capture and segmentation processes, each character image was converted from the gray scale to a binary scale [0, 1] and stored as a LZW Tiff image format.

The experiments related to this paper were restricted to the use of numeric characters only. From the total amount of existing numbers it was noticed that a small number of them were not correctly segmented or presented problems of texture or trace, and were taken out of the context. The total amount of data finally available for the experiments is shown in the table 1. As can be seen on the table, the amount of each number is not homogeneous.

Table 1 – Individual digits amount distribution

DIGIT	1	2	3	4
0	16,00%	681	544	137
1	15,90%	680	544	136
2	18,00%	770	616	154
3	9,39%	401	320	81
4	7,54%	322	257	65
5	7,68%	328	262	66
6	6,68%	285	228	57
7	7,50%	320	256	64

8	5,65%	241	192	49
9	5,65%	241	192	49
Total	100%	4269	3411	858
		100%	80%	20%

1 – Percentage value from the total number of patterns.
2 – Number of patterns per class
3 – Training patterns per class (80% from column **2**)
4 – Test patterns per class (20% from column **2**)

As mentioned before, the major evaluation objective was to find out if the proposed approach is able or not to improve the discrimination performance. In order to tackle such goal it was chosen a MLP neural network model. The comparative analyses considered the bitmap matrix coding (full and thinned) and the ones obtained from the usage of the two previously describe polygons: square and hexagon.

The bitmap matrix was obtained converting the original image from gray to binary scale and then rescaling the image to fit in 16 by 16 pixels. The thinned image was obtained through the application of a thinning algorithm from [10] (figure 8).

The polygonal features vectors were mounted through a succession of extractions using 5 different dimensions for each type of polygon: 32, 64, 128, 256 and 512 features. The vectors were built through the program showed in the figure 9.

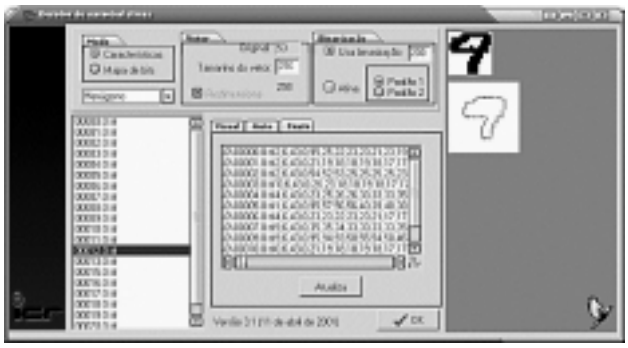

Figure 9: Feature extractor API

The vector dimension for the square polygon, is exactly the ones listed above, however, for the hexagon this is not true. In the hexagon case, the dimension value was increased to match the real proportion. For instance: 256 / 4 = 64 (square) but we have 256 / 6 = 42.666 and so, the closest value will be: 258 / 6 = 43. The difference is always small and as mentioned before, it does not interfere on the final results.

The format of the output file generated by the extraction process is of the ASCII type as shown in figure 11. This files is the input for the MLP neural model, and each register has a layout as shown in figure 10. The MLP neural model was created using the MATLAB Neural Networks Tool Box [12].

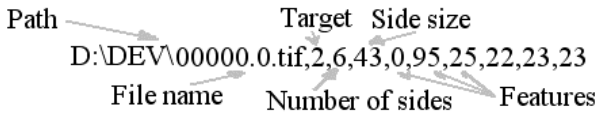

Figure 10: detail of one line in the features file.

EVALUATION BY NEURAL NETWORKS

The type of neural network model chosen to evaluate the discrimination capability of the proposed feature extraction technique was a single MPL – Multi Layer Perceptron Network, trained with the backpropagation algorithm. The network architecture has one variable size hidden layer and one output layer with 10 neurons (table 2). The network was trained to learn all 10 digits. The size of the hidden layer was empirically chosen based on the size of the input feature vector. The activation function was implemented using a simple summation of the inputs adjusted by theirs corresponding weights. For the propagation function it was decided to use the sigmoid. The summary of the obtained results is shown in table 3.

Table 2 - Network output scheme. The output vector was chosen to be orthogonal to facilitate the MLP training process.

Character	Output
0	0000000001
1	0000000010
…	…
9	1000000000

Data Preprocessing

The data set was split into two distinct sets: a training set and a test set, according to the amount showed in table 1. Except the bitmap matrix values (full and thinned), all neural network input data (training and test) were normalized by the maximum value, once this value represents the distance between two opposite sides of the polygon used.

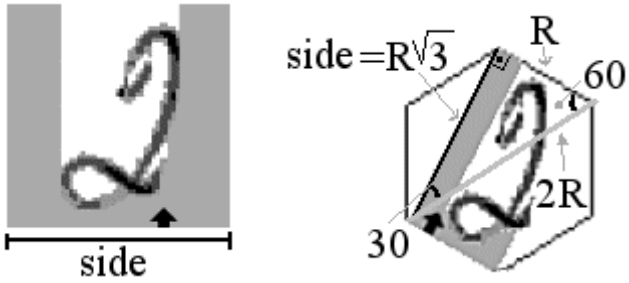

Figure 11: Side representation for the normalization factor

Another procedure adopted for the data set preparation was to eliminate all those columns with zero variance. This very simple procedure eliminated some variables from the input space and so, reducing its dimensionality. For instance: square side 128 ⇔ 116; hexagon side 136 ⇔ 128.

As said before, the neural network environment used in this experiment was the Neural Networks Toolbox from MATLAB version 5.02. The MLP training was done using the "traingdx" option with moment and adaptive learning rate parameters.

Once the major goal is a comparative analysis, it was chosen to define a simple output selection criterion. In this way, the neural network output was modified according to the strategy "Winner Takes All", where the largest value among all 10 outputs was changed to 1 (one) and the remaining ones were changed to 0 (zero). This procedure leads to a zero rejection for the neural network response.

The training was executed with 3 different epochs sizes: 500, 900 and 1000. The obtained results for the 500 case epochs are shown in table 3 below.

Table 3 – Network configuration with 500 epochs. The representation A-B-C means A= # of input, B= # of hidden layer neurons and C= # of output neurons. S=Square,H=hexagon, Bitmap-T=bitmap thinned matrix

Method	Network configuration	Epochs	% error	% corrects
S-32	24 – 17 – 10	500	51.63	48.37
S-64	56 – 23 – 10	500	15.97	84.03
S-128	116 – 47 – 10	500	8.39	91.61
S-256	240 – 87 – 10	500	7.23	92.77
S-512	484 – 173 – 10	500	7.11	92.89
H-32	37 – 17 – 10	500	58.16	41.84
H-64	63 – 23 – 10	500	38.58	61.42
H-128	136 – 47 – 10	500	15.27	84.73
H-256	256 – 87 – 10	500	9.67	90.33
H-512	520 – 173 – 10	500	9.79	90.21
Bitmap	223 – 85 – 10	*378	11.42	88.58
Bitmap-T	252 – 85 – 10	*360	11.07	88.93

Table 4 – Network configuration with 900 epochs.

Method	Network configuration	Epochs	% error	% correct
S-32	24 – 17 – 10	900	45.80	54.20
S-64	56 – 23 – 10	900	10.02	89.98
S-128	116 – 47 – 10	900	6.76	93.24
S-256	240 – 87 – 10	900	6.53	93.47
S-512	484 – 173 – 10	900	5.36	94.64
S-32	37 – 17 – 10	900	53.26	46.74
S-64	63 – 23 – 10	900	32.75	67.25
S-128	136 – 47 – 10	900	12.00	88.00
S-256	256 – 87 – 10	900	7.69	92.31
S-512	520 – 173 – 10	900	7.11	92.89
Bitmap	223 – 85 – 10	378	11.42	88.58
Bitmap-T	252 – 85 – 10	360	11.07	88.93

Table 5 – Network configuration with 1000 epochs.

Method	Network configuration	Epochs	% error	% correct
S-32	24 – 17 – 10	1000	45.92	54.08
S-64	56 – 23 – 10	1000	10.96	89.04
S-128	116 – 47 – 10	1000	5.59	94.41
S-256	240 – 87 – 10	**943	5.94	94.06
S-512	484 – 173 – 10	1000	5.24	94.76
S-32	37 – 17 – 10	1000	51.75	48.25
S-64	63 – 23 – 10	1000	33.33	66.67
S-128	136 – 47 – 10	1000	16.90	83.10
S-256	256 – 87 – 10	1000	6.88	93.12
S-512	520 – 173 – 10	1000	6.99	93.01
Bitmap	223 – 85 – 10	**345	10.96	89.04
Bitmap-T	252 – 85 – 10	**364	12.47	87..53

It can be noticed that square polygon with 512 features provided the best result in all cases above – 500, 900 and 1000 epochs. Notice that for some networks (*/**) the training converged to the minimum desired error before *500/**1000 epochs.

From the tables, one can observe that the results provided by the square model with features dimension of size 128 and 256 were not far away from the ones obtained with the 512 model. Taking into consideration the computational effort generated by the different models and the results accuracy, one can reach to the conclusion that a 128 square may be the best option.

As said at the beginning of this paper, it was realized many different experiments varying: feature extraction method, input vector size and network architecture size. Table 5 shows details for one of those experiments.

Table 6 – Detailed results for one experiment

Method	square 128
Number of Inputs	116
Number of Outputs	10
Number of Layers	2
Neurons in the hidden layer	47
Training function	traingdx
Learning rate	0.30
Momentum	0.50
Epochs (predicted)	1000
Epochs (done)	1000
MSE	0.0109507
Correct Classifications	810
Incorrect Classifications	48
Rejections	0
Correct classifications %	94.41
Incorrect classifications %	5.59
Rejections %	0.00

Table 7 - Confusion matrix for the Square 128

135	0	0	0	1	0	2	0	4	0
1	135	3	1	5	0	0	3	0	2
0	0	149	4	0	0	0	1	2	1
0	0	1	73	0	0	0	0	0	1
0	1	0	0	56	0	0	0	0	1
0	0	0	1	1	65	0	0	1	0
0	0	0	0	0	0	55	0	0	0
0	0	0	1	1	0	0	58	0	0
1	0	0	0	0	0	0	1	42	2
0	0	1	1	1	1	0	1	0	42

From the confusion matrix it is easy to observe that the accuracy provided by the square 128 was quite good. The mistakes are few and mainly concentrated on number one that sometimes were recognized as 2, 4, 7 and 9.

CONCLUSIONS

As showed in the paper, the proposed approach, manly the one based on the square polygon, proved to provide good discrimination power to be used with neural network models. Comparing the results obtained here with those described in [11, we may conclude that square method works better than all other polygons already evaluated: circle using radius, circle using diameter, hexagon and octagon, and also superior to the bitmap approach using or not thinning process. [4 and 11]

Future Works

The ongoing research is now expanding the database by the inclusion of alphabetic symbols and the comparison of the method with other approaches found in the literature.

REFERENCES AND BIBLIOGRAPHY

[1] Rodrigues, R. J.; Thomé, A. C. G.; “Cursive character recognition – a character segmentation method using projection profile-based technique”, The 4th World Multiconference on Systemics, Cybernetics and Informatics SCI 2000 and The 6th International Conference on Information Systems, Analysis and Synthesis ISAS 2000 – Orlando, USA – August 2000

[2] Bishop, C. M.; “Neural Networks for Pattern Recognition”, Oxford University Press, 1995, pp.

[3] Duda, R. O; Hart, P. E. “Pattern Classification and Scene Analysis”, John Wiley & Sons, 1973, pp.

[4] Kupac, Gizelle Vianna, Rodrigues, R. J.; Thomé, A. C. G.; “Extração de características para reconhecimento de dígitos cursivos”, VIth Brazilian Symposium on Neural Networks - SBRN2000, Rio de Janeiro, november 2000.

[5] Srikantan, G.; Lam; S. W.; Srihari, S, N; Gradient-Based Contour Encoding For Character Recognition, Pattern Recognition, Vol. 29, No. 7, pp. 1147-1160, 1996

[6] Suen, C.Y.; Berthold, M.; Mori, S.; Automatic Recognition of Handprinted Characters – The State of The Art,Proceedings of the IEEE, Vol. 68, No. 4, April 1980

[7] Trier, O. D.; Jain, A.K.; Taxt, T.; “Feature Extraction Methods for Character Recognition – A Survey”; Pattern Recognition, Vol. 29, No. 4, pp. 641-662, 1996

[8] Verschueren, W.; Schaeken, B.; Cotret, Y. R.; Hermanne, A.; Structural Recognition of Handwritten Numerals, CH2046-1/84/0000/0760$01.00@1984 IEEE

[9] Yang, L.; Prasad, R.; “Online Recognition of Handwritten Characters Using Differential Angles and Structural Descriptors”, Pattern Recognition Letters, no 14, Dec-93, North-Holland, pp: 1019-1024.

[10] Pavilidis. Theodosios, Algorithms for Graphics and Image Processing, Computer Science Press, Inc., 1982

[11] Character Feature Extraction Using Polygonal Projection Sweep (Contour Detection), accepted for presentation at IWANN2001. Granade - Spain, June 13 a 15.

[12] Demuth, Howard; Beale, Mark; "Neural Network Toolbox User’s Guide Version 3.0", COPYRIGHT 1992 - 1997 by The MathWorks, Inc.