Recognition of Multi-Oriented Touching Characters in Graphical Documents

Abstract

Touching characters are major problem of achieving higher recognition rate in Optical Character Recognition (OCR). Present OCR systems do not perform well when adjacent characters touch. If characters are touched in graphical documents (e.g. map) then such touching string recognition is more difficult because in such documents touching characters appear in multi-oriented direction. In this paper, we present a scheme towards the recognition of English two-character multi-oriented touching strings. When two or more characters touch, they generate a big cavity region at the background portion and we used this background information in our scheme. To handle the background information, convex hull is used. In this scheme, at first, a set of initial segmentation points is predicted based on the concave residues of the convex hull of the touching characters. Next, based on the initial points, we select some candidate segmentation lines. Finally the recognition confidence of two sub-images of a touching string, obtained from each candidate segmentation line, is computed. The candidate segmentation line from which we get optimum confidence is the actual segmentation line and the corresponding characters in favour of which the two segmentation parts show optimum confidence is the recognition result of the touching string. To compute the recognition confidence, SVM classifier is used. The features used in the SVM are invariant to character orientation. Circular ring and convex hull ring based approach has been used along with angular information of the contour pixels of the character to make the feature rotation invariant. From the experiment we obtained encouraging result.

1. Introduction

Optical character recognition (OCR) systems available commercially are not capable of handling different ranges of images containing multi-sized and multi-oriented characters. For printed text, the main difficulty arises from the severely merged or degraded characters. Incorrect segmentation of merged characters is still one of the main causes for recognition errors. As segmentation errors induce recognition errors, the performance of segmentation is crucial for the whole OCR process. In graphical documents, such as map, text characters are often in different scale to give importance to specific locations in some regions. Also the characters frequently appear in different orientations other than the usual horizontal directions. They are printed thus to annotate graphical objects such as river, road etc. The interpretation of graphical documents does not only require the recognition of graphical parts but the detection and recognition of text that may appear in multi-oriented fashion. If the characters, aligned in multi-oriented direction are joined due to degradation, it is difficult for segmentation and recognition. We show a part of a map in Fig.1 to illustrate the problem. It can be seen that, in the words "Avranches" and "Couesnon" some of the characters are touched (for example, "an", "ch" of the word "Avranches" and "es" and "no" of the word "Couesnon"). It may be noted that orientations of "es" and "no" in the word "Couesnon" are not same and thus it is very difficult for their recognition.

Fig.1. Example of a map shows orientation of different characters.

Although there is some published work to recognize the touching characters of normal horizontal direction

[10], to the best of our knowledge, there is no work to recognize touching characters of arbitrary orientation. To handle multi-oriented touching string of map documents, in this paper, we propose a scheme to recognize two-character touching patterns of arbitrary orientations. When two or more characters touch, they generate a big cavity region at the background portion and we used this background information in our method. To handle the background information convex hull is used. In the proposed scheme, at first, a set of initial segmentation points is predicted based on the concave residues of the convex hull of the touching characters. Next, based on the initial set, we select some candidate segmentation lines by considering the position of the points with respective to the residue surface level and comparing the size of the two segmented parts at the segmentation line. Finally based on the candidate segmentation lines, the recognition confidence of two sub-images, obtained from the touching string is computed. The candidate segmentation line from which we get optimum confidence is the actual segmentation line and the corresponding characters in favour of which the two segmentation parts show optimum confidence is the recognition result of the touching string. To compute the confidence, SVM classifier is used and based on the recognition result of the two segmented parts, the optimum confidence is calculated. For recognition purpose, the features used are mainly based on the angular information of the external and internal contour pixels of the characters. We compute frequencies of different angles obtained from the contour pixels of a character and these frequencies are used for recognition. It may be noted that such frequency of a character will be similar even if the character is rotated. Circular ring and convex hull ring based approach has been used to divide a character into several zones and zone wise angular histogram is computed to get higher dimensional feature for better performance.

The rest of the paper is organized as follows. In Section 2 we describe briefly the related work on touching character segmentation and multi-oriented character recognition approaches. In Section 3, we explain our feature extraction procedure for multi-oriented string handling. Recognition confidence computation is discussed in Section 4. The experimental results are discussed in Section 5. Finally conclusion is given in Section 6.

2. Related work

For isolated character recognition at different scale and rotation, there exist some pieces of work. The method proposed by Adam et al. [1] used Fourier-Mellin Transform for text character recognition from engineering drawings. The image is convolved by a set of filters and then the system tries to locate the pixel for which the response is pre-specified. Hase et al. [2] used parametric eigen-space for rotated character recognition. Xie and Kobayashi [3] proposed a system for multi-oriented English numeral recognition based on angular patterns. Pal et al. [6] used a feature based on angular information of the contour points of the characters. Recently, Roy et. al. [8] proposed an approach for multi-oriented isolated character recognition from graphical documents.

Although there exist some work towards the recognition of multi-oriented isolated character, but to the best of our knowledge there is no work towards the recognition of multi-oriented touching characters and this is the first work for the recognition of multi-oriented touching strings.

Touching character recognition methods can be divided into two classes: segmentation based approach and holistic approach [10].

In segmentation based recognition domain, at first touching string is segmented into characters and then segmented characters are passed for recognition. Character segmentation positions can be located from the valleys of the projection profile [14, 15]. Arica and Yarman-Vural [18] define a cost function using the information extracted from both grayscale image and binary image, and use a dynamic programming algorithm to find the shortest path. Chen and Wang [17] extracted feature points from the thinned image of both foreground and background of the image and then constructed several segmentation paths from this information.

In holistic method, the algorithms use the word as a single entity, and try to recognize it using features of the word. Rocha and Pavlidis [12] propose a segmentation-free approach based on feature graph and match the sub-graphs of features with predefined character prototypes. Lee and Kim [13] have used cascaded neural network to use the spatial relationship of connected characters.

3. Feature extraction

To make the system rotation invariant, the features are mainly based on angular information of the external and internal contour pixels of the characters. Circular ring and convex hull ring have been used to divide a character into several zones [8]. Details of the feature extraction are as follows.

3.1. Global feature based on angular information

For an input image internal and external contour pixels are computed and they are used to determine the angular information feature of the image.

Given a sequence of consecutive contour pixels $V_1 \ldots V_i \ldots V_n$, of length n (n>7), the angular information of the pixel V_i is calculated from the orientation of vector pairs V_{i-k}, V_i and V_i, V_{i+k}. For better accuracy, we take the average of 3 orientations for each pixel, considering k=1, 2 and 3. From each pixel we will get two angles (one from background side and other from foreground side) and the angle corresponding to background side is considered here. The angles obtained from all the contour pixels of a character are grouped into 8 bins corresponding to eight angular intervals of 45 degree (337.5 degree to 22.5 degree as bin no. 1, 22.5 to 67.5 as bin no. 2 and so on). For a character, frequency of the angles of 8 bins will be similar even if the character is rotated at any angle in any direction. For illustration, see Fig.2. Here, we compute the histogram of the angles corresponding to 8 angular information of two rotated shapes of the character "M". From the figure it can be noted that angle histogram of two characters is similar although the character has different rotations.

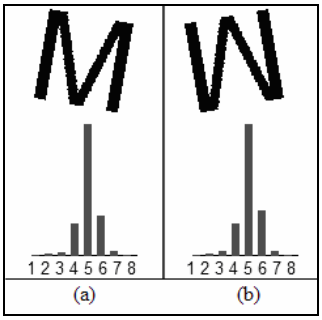

Fig.2. Input images of the character "M" in 2 different rotations and their angle histogram of contour pixels are shown. The numbers 1-8 represent 8 angular bins.

We may feed this global angular histogram of the characters into the classifier for recognition but to get higher accuracy, we divide a character into several zones and zone-wise angular information is computed and hence higher dimensional features are found. Circular ring and convex hull ring have been used for this purpose.

3.2. Higher dimensional feature detection using circular ring and convex hull

Circular ring based division: A set of four circular rings is considered here and they are defined as the concentric circles considering centre as the centre of minimum enclosing circle (*MEC*) of the character and the minimum enclosing circle as the outer ring. The radii of the rings are chosen such that, the area of each ring is equal. Let R_1 be the radius of *MEC* of the character, then the radii (outer to inner) of these four rings are R_1, R_2, R_3 and R_4, respectively. Where, $\prod*R_4^2 = \prod*R_3^2 - \prod*R_4^2 = \prod*R_2^2 - \prod*R_3^2 = \prod*R_1^2 - \prod*R_2^2$. See Fig.3(a), where four rings are shown on the character 'E'. These circular rings divide the *MEC* of a character into four zones.

Convex hull based division: Convex hull rings are computed from the convex hull boundary. We compute 4 convex hull rings and we consider the outermost convex hull ring (say C_1) as the convex hull itself. Other 3 convex hull rings are similar in shape and computed from C_1 by reducing its size. The 2nd ring can be visualized by zooming out C_1 with $R_1 - R_2$ pixels inside. Other 2 rings are computed similarly. Four convex hull rings are shown in Fig.3(b) on the character 'W'.

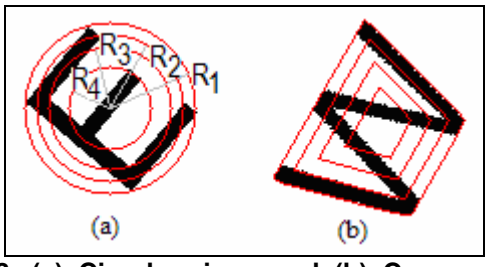

Fig.3. (a) Circular rings and (b) Convex hull rings.

We have used this convex-hull representation since it has some interesting properties. These properties help us to get higher dimensional feature. Let us describe such parameters in the following.

Residue surface level (*RSL*): The RSL is obtained by joining two endpoints of the open face of the residue.

Residue depth: It is the normal distance of the farthest point of the residue from RSL.

Residue area: The area of a residue is defined by the number of points inside the residue.

Residue border pixels: The border pixels of each residue are defined as the contour pixels of the residue excluding the RSL pixels.

Mid-Point of RSL: This is defined as the mid-point of RSL.

See Fig.4 where residue and its different parameters are shown.

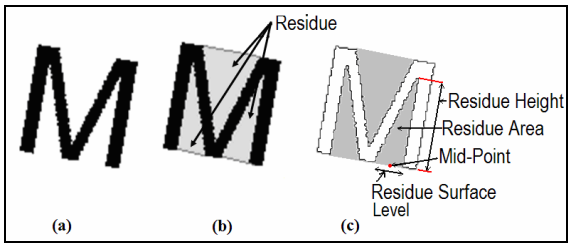

Fig.4.(a) Image of the character "M". (b) Three concave residua from the convex hull of "M". (c) Different parameters of convex hull.

Reference point and reference line detection: If we compute information of angular histogram on the character portions in each of the ring, then we will get 32 (4 rings × 8 angular information) dimensional feature. To get more local feature for higher accuracy, we have divided each ring into few segments. To do such segments, we need a reference line (which should be invariant to character rotation) from a character. The reference line is detected based on the background part of the character using convex hull property. The mid-point of *RSL* of the largest (in area) residue of a character is found and the line obtained by joining this mid-point and the centre of *MEC* of the character is the reference line of the character. If there are two or more largest (in area) residue then we check the height of these largest residue and the residue having largest height is selected. If the heights of the largest residue are same, then we select the residue having maximum *RSL*. The mid-point of the *RSL* of the selected residue and centre of *MEC* of the character is the reference line. The mid-point of *RSL* is the reference point. If no residue is selected by above, we consider the farthest contour point (P_f) of the character from the centre of gravity (CG) of the character and the line obtained by joining P_f and the CG is the reference line. Here, P_f is the reference point.

Initially, we used principal component analysis to get a reference line and we obtained lower accuracy. Hence we have computed reference line as discussed above.

A reference line can segment each ring into two parts and if we compute the feature on each of the segment, then we will get 64 dimensional features (4 rings × 2 segments × 8 angular information). If we take another reference line perpendicular to this reference line, then each ring will be divided into 4 segments and as a result, we will get 128 dimensional features (4 rings × 4 segments × 8 angular information). See Fig.5, where two reference lines PP´ and QQ´ are shown. To get 256 dimensional features, we consider 2 more reference lines which are angular bisectors of these lines.

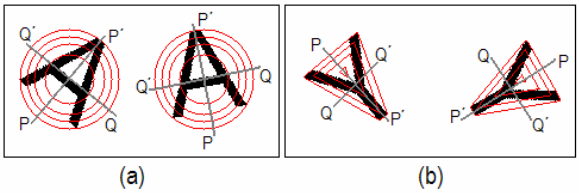

Fig.5. Reference lines PP´ and QQ´ are shown with (a) circular ring division in character 'A' (b) convex hull ring division in character 'Y'.

To get the different segments sequentially we consider the segment that starts from the reference point as segment number 1 (say S_1). To get 256 dimensional features each ring will be divided into 8 block segments. Starting from S_1 if we move anti-clockwise then the segments obtained from outer ring block (R_1-R_2) are designated as 1st, 2nd...8th. Similarly, from the (R_2-R_3) ring block, we will get 9th, 10th.....16th segment. Other segments are obtained in similar way.

To get size independent features we normalize them. For normalization we divide the angular information in each segment by the total number of contour pixels. Hence, we get these feature values between 0 and 1.

4. Recognition Confidence Computation

Recognition confidence computation of a touching string is done based on the segmented pair obtained by a segmentation line. Details of the segmentation line extraction method are discussed below. Before, going into details, we will describe segmentation zones, computation of initial segmentation points which are needed for detecting candidate segmentation lines.

4.1. Computation of segmentation zones

For a touching character, at first stroke-width (S_w) is calculated. The stroke width S_w is nothing but the statistical mode of the black run lengths of the character. For a character, S_w is calculated as follows. The character is, at first, scanned row-wise (horizontally), column-wise (vertically) and then in two diagonal directions (45º and 135º). If n different runs of lengths r_1, r_2..r_n with frequencies f_1, f_2..f_n, respectively are obtained by the scanning from the character, then value of S_w will be r_i if $f_i = \max(f_j)$, j = 1, 2,..n.

For an input image of a touching character, we compute the convex hull to find the concave residues,

as they will be used to determine the probable segmentation points. These residues are regarded as the segmentation zones. In our process, we do not consider all residues. Only those residues having depth > S_w are considered. The residues found from convex hull of a touching character are shown in Fig.6.

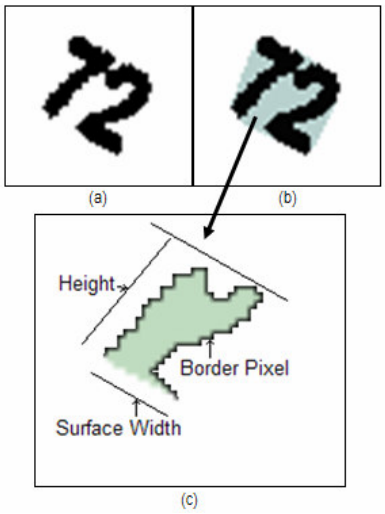

Fig.6.(a) Image of the touching character "72". (b) Concave residues from the convex hull of "72". (c) A convex hull with its different parameters is shown on a zoomed version of a residue found from (b).

4.2. Computation of initial segmentation points

We apply a polygonal approximation method, due to Douglas–Peucker [11,16] to the residue border contour pixels. This is done to segment the residue curves into a list of lines to find initial segmentation points. The line of RSL is regarded as the initial rough estimation of the poly-line. Using this crude initial guess, the other vertices are approximated using a tolerance threshold $\epsilon > 0.5*S_w$. After approximation, we will have the end-points of these line-segments which are treated here as key-points. The advantage of using polygonal approximation is that, it helps us to provide the key-points which are at the corner of edges in that segmentation zone. These key points information is necessary for touching character segmentation because, usually when the characters touch, they form a corner at the touching region and such points are initial segmentation points.

Further, we exclude the key-points which are very close to residue surface level. This is because we assume segmentation points are close to deepest point from that residue surface width. The remaining key-points are the initial segmentation points. In Fig.7, we have shown the initial segmentation points for the image Fig.6(a).

Fig.7. Image shows initial segmentation points found from concave residues after using polygonal approximation.

4.3. Computation of candidate segmentation lines

Once we get an initial segmentation point (S_i), we compute another point (S_j) through which we will divide this whole image. This will be done by joining a straight line from S_i to S_j. S_j is selected as follows.

We know the direction of the residue surface level (RSL) with the horizontal axis. The perpendicular angle will give us a clue to the direction of the segmentation line. Let the perpendicular angle with the direction of RSL be θ. We draw a line from the point S_i at an angle θ with the RSL until it passes through the object pixels. Let the last object pixel on this line be S_c. The line from S_i to S_c is considered as segmentation line. This segmentation line may not give the best segmentation always and to get better segmentation, the point S_c is tuned to have better segmentation point. This tuning is done by considering some neighbour contour points from point S_c. Tracing the contour upto stroke-width length in clockwise and anti-clockwise starting from S_c, we get these neighbour points. From, the neighbour pixels, a point is chosen which has the minimum distance from S_i. This point is S_j. The line obtained by joining S_j and S_i is the probable segmentation line. So, for every initial segmentation point, we can have its respective segmentation line. But, to reduce the computation complexity, we remove some segmentation lines which divide the image in two very un-equal size. The segmentation lines which segment the touching component into two parts of similar sizes are considered for future consideration. Size similarity is done by finding the minimum enclosing rectangle (*MER*) of the resultant sub-images. Let the larger sides of the two *MER* are L_1 and L_2. If $\max(L_1, L_2) > 1.5*\min(L_1, L_2)$, then we ignore such segmentation lines. The remaining segmentation lines are considered as candidate lines. Candidate segmentation lines of Fig.7 have been shown in Fig.8. From this set of candidate segmentation lines, we will

choose the best line using recognition confidence obtained by SVM.

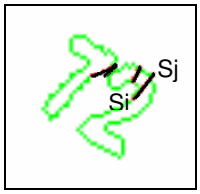

Fig.8. Image shows the candidate segmentation lines formed by selected segmentation points.

4.4. Selection of best segmentation line

We will select the best segmentation line by recognition confidence, computed by Support Vector Machine. This is done as follows. For each segmentation line, we will have two sub-images, which are formed by dividing the two-character touching string by the segmentation line. Two sub-images are tested for their recognition confidence by SVM, which is detailed as follows. Based on the recognition result, our SVM generates a value (between 0 and 1) for each character and this value is the confidence for that character. We add the confidence of these two sub-images to get the total confidence. When a line separates a touching character at the proper location, we generally get highest confidence value from the sub-images of the string. This segmentation line will be invariant to rotation and size. So, for each segmentation lines, we compute the confidence value of two segmented parts and use the joint confidence value as the cost function to the corresponding segmentation line. A rejection threshold function is included to achieve good performance. If the joint confidence value is less than TH_{rej}, we do not consider that touching component. In our method, the value of TH_{rej} is considered as 0.50 and the value is obtained from experimental result. The segmentation line, for which we get the highest confidence result, gives us the final segmentation line. In Fig.9 we have shown the final segmentation result of the touching character of Fig.6(a). To get an idea about the number of segmentation line for each touching component, we computed total number of segmentation line in our dataset. We noted that on an average, we got 2.81 segmentation line for a touching string.

4.5. SVM classifier

The Support Vector Machine (SVM) is defined for two-class problem and it looks for the optimal hyper-plane which maximizes the distance, the margin, between the nearest examples of both classes, named support vectors (SVs). Given a training database of M data: {xml m=1,...,M}, the linear SVM classifier is then defined as: $f(x)=\sum_j \alpha_j x_j \cdot x + b$

Where, $\{x_j\}$ is the set of support vectors and the parameters α_j and b have been determined by solving a quadratic problem [5]. The linear SVM can be extended to a non-linear classifier by replacing the inner product between the input vector x and the SVs x_j, to a kernel function k defined as: $k(x,y)=\phi(x)\cdot\phi(y)$. This kernel function should satisfy the Mercer's Condition [5]. Some examples of kernel functions are polynomial kernels $(x\cdot y)^p$ and Gaussian kernels exp(-||x-y||2/c), here c is a real number. We use Gaussian kernel for our experiment.

Fig.9. Image shows final segmentation line.

5. Result and discussions

5.1. Data collection

For the experiment of present work, we considered real data from map, newspaper, magazines etc. We used a flatbed scanner for digitization. Digitized images are in grey tone with 300 dpi. We have used a histogram based global binarization algorithm [4] to convert them into two-tone (0 and 1) images. We also smoothed the image to get better result [19]. An automatic method [7] has been used to extract text characters from the scanned document. From this text dataset, we manually select the touching character components which are formed by 2 characters touching. In our experiment, we tested our scheme on 1200 touching characters. It contains touching string of different font, size and orientation.

Except the touching dataset, we have a dataset of isolated characters. Total data size of this isolated dataset is 26,482 (8,250 from graphical document and 18,232 from normal text). This dataset is also of different font, size and orientation. We use this data to train our SVM. Both uppercase and lowercase letters were considered for our experiment, so we should have 62 classes (26 for uppercase, 26 for lowercase and 10

for digit). But because of shape similarity of some characters/digits, here we have 40 classes. We are considering arbitrarily rotation (any angle up to 360 degrees) so, some of the characters like “p” and “d” are considered same since, we will get the character “p” if we rotate the character “d” 180 degrees. Hence, we have 40 classes instead of 62 classes.

5.2. Segmentation result

For experiment of touching components, two-character touching of multi-oriented directions are considered. To check whether a touching component is segmented correctly or not, we draw a line which segments the touching string into individual characters. By viewing the results on the computer’s display we check the line segmentation results manually.

From the experiment, we noticed that out of 1200 touching characters, there was no error on 1054 components. The number of rejected component is 84. Hence, segmentation accuracy is 94.44%. In Fig.10, we have shown some touching images with their segmentation results. Fig.11. shows some wrong segmentation of touching characters from our method.

Fig.10. Few images showing segmentation lines.

From the experiment, we noted that most of the segmentation errors are for the following two cases.
(a) When a touching string can be segmented in more than two ways to get the valid segmented characters. For example, see the first example of Fig.11. Here the touching string was formed from the characters ‘r’ and ‘m’. But this touching string can be visualized as ‘n’ and ‘n’ also, and our system segmented this string into ‘n’ and ‘n’ instead of ‘r’ and ‘m’, which we consider as erroneous.
(b) When there is a multiple touching instead of single touching, our method could not segment it properly. For example, see the 2nd example of Fig.11. Here, the characters ‘t’ and ‘n’ are touching in two positions and our system segmented this string into ‘b’ and ‘l’. We also considered it as wrong segmentation. Since our method is based on convex hull, when touching is made in two or more positions, then we can not find any segmentation line in the touching cavity region. Hence we get erroneous results.

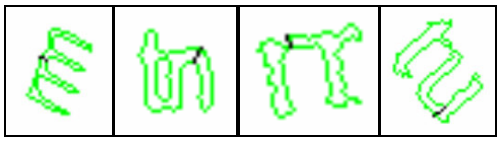

Fig.11. Images showing wrong segmentation.

5.3. Recognition result

For recognition accuracy computation of 2-touching character string, we have considered only the good segmented characters. It is noticed that, the accuracy obtained by SVM is 97.6% for touching characters. The errors were generated mainly from similar-shaped characters pairs ‘f’, ‘r’; ‘t’, ‘l’ etc.

5.4. Comparison of segmentation results

To the best of our knowledge, this is the first work on multi-oriented touching strings. Hence we can not compare the results. However to get an idea about the segmentation result of the existing system on normal touching of horizontal direction, we compare the results in Table 1. It has been noted that, though the accuracy is less compared to that of result obtained by [10, 17], but we think for multi-oriented environment it is a good performance.

Table 1: Comparison of results.

Method	Touching Mode	Segmentation Accuracy
FP-based system [10]	Single Direction (Horizontal)	98.3%
Chen and Wang [17]	Single Direction (Horizontal)	96.0%
Our System	Multi-Direction (Multi-Orientation)	94.44%

5.5. Drawback of the proposed system

As we discussed in Section 5.2, if there are more than one touching in a touching string our method will not work properly. Another drawback of our method is that it will not work if the characters are broken and that broken part could not be joined through the pre-processing, we used.

6. Conclusion

In this paper, we have proposed a scheme towards recognition of multi-oriented touching characters. Based on the experiment on 2000 two-character touching data in multi-oriented direction, we obtained 94.44% segmentation accuracy and 97.6% recognition accuracy. To the best of our knowledge, this is the first work towards multi-oriented touching strings. In future, we plan to extend our work to handle n-character touching strings in multi-oriented direction.

References

[1] S. Adam, J. M. Ogier, C. Carlon, R. Mullot, J. Labiche and J. Gardes,"Symbol and character recognition: application to engineering drawing", IJDAR,(3), pp. 89-101, 2000.

[2] H.Hase, T.Shinokawa, M.Yoneda and C. Y.Suen, "Recognition of Rotated Characters by Eigen-space", Proc. 7th ICDAR, pp. 731-735, 2003.

[3] Q.Xie and A. Kobayashi, "A construction of pattern recognition system invariant of translation, scale-change and rotation transformation of pattern", Trans. of the Society of Instrument and Control Engineers (27), pp.1167-1174, 1991.

[4] F. Kimura and M. Shridhar, "Handwritten numeral recognition based on multiple algorithms", Pattern Recognition, vol. 24, pp. 969-983, 1991.

[5] V. Vapnik, "The Nature of Statistical Learning Theory", *Springer Verlang,* 1995.

[6] U. Pal, F. Kimura, K. Roy, T. Pal: Recognition of English Multi-oriented Characters. ICPR (2) pp. 873-876, 2006.

[7] P. P. Roy, E. Vazquez, J. Lladós, R. Baldrich, U. Pal, "A system to segment text and symbols from color maps", GREC, pp. 245-256, 2007.

[8] P. P. Roy, J. Lladós, U. Pal, "Multi-Orientrd Character recognition from Graphical documents", ICCR, Mandya, India, pp. 30-35, Apr. 2008.

[9] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation", IEEE Trans. Pattern Anal. Machine Intell., vol. 18, pp. 690–706, Jul. 1996.

[10] J. Song, Z. Li, M. R. Lyu, Shijie Cai, "Recognition of Merged Characters Based on Forepart Prediction, Necessity-Sufficiency Matching, and Character-Adaptive Masking", IEEE Trans. Syst., Man, Cybern. B, vol. 35, no. 1, 2005.

[11] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature", The Canadian Cartographer, vol. 10, no. 2, pp. 112-122, 1973.

[12] J. Rocha and T. Pavlidis, "Character recognition without segmentation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 17, no. 9, pp. 903–909, Sep. 1995.

[13] S.W.Lee and S.Y.Kim, "Integrated segmentation and recognition of handwritten numerals with cascade neural network," IEEE Trans. Syst., Man, Cybern. C, Appl. Rev., vol. 29, no. 2, pp. 285–290, May 1999.

[14] Y. Lu, "On the segmentation of touching characters," ICDAR, Tsukuba City, Japan, Oct. 1993, pp. 440–443.

[15] S. Liang, M. Ahmadi, and M. Shridhar, "Segmentation of touching characters in printed document recognition," ICDAR, Tsukuba City, Japan, Oct. 1993, pp. 569–572.

[16] J. Hershberger and J. Snoeyink, "Speeding Up the Douglas-Peucker Line-Simplification Algorithm", Proceedings of the 5th International Symposium on Spatial Data Handling, vol. 1, pp. 134-143, 1992.

[17] Y.K.Chen and J.F.Wang, "Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis," IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 11, pp. 1304–1317, Nov. 2000.

[18] N. Arica and F. T. Yarman-Vural, "Optical character recognition for cursive handwriting," IEEE Trans. Pattern Anal. Mach. Intell., vol. 24, no. 6, pp. 801–813, Jun. 2002.

[19] K. Roy, U. Pal and B. B. Chaudhuri, "Address block location and pin code recognition for Indian postal automations", In Proc. National Workshop on Computer Vision Graphics and Image Processing, pp. 5-9, 2004.