2011 International Conference on Document Analysis and Recognition

Video Character Recognition Through Hierarchical Classification

[a]Palaiahnakote Shivakumara, [a]Trung Quy Phan, [b]Shijian Lu and [a]Chew Lim Tan
[a]School of Computing, National University of Singapore, Singapore
{shiva, phanquyt, tancl}@comp.nus.edu.sg
[b]Institute for Infocomm Research, Singapore, [b]slu@i2r.a-star.edu.sg

***Abstract*— We present a new video character recognition method based on hierarchical classification. In the first step, we propose a method for character segmentation of the text line detected by the text detection method. The segmentation algorithm uses dynamic programming to find least-cost paths in the gray domain to identify the spaces between characters. For the segmented characters, we get a Canny edge image as input for the character recognition step. We introduce hierarchical classification based on voting criteria with structural features to classify 62 character classes into different smaller classes. We divide the perimeter of a character into 8 segments according to 8 directions at the centroid. Then the shape of each segment is studied to recognize the characters based on distances between the centroid and end points, and distances between the midpoint and end points. Our experiments on 1462 characters of upper case, lower case and numerals shows that 10% samples per class for training is enough to obtain 94.5% recognition accuracy. The dataset is chosen from TRECVID database of 2005 and 2006.**

Keywords- ***Structural features, Hierarchical classification, Invariant features, Confusion matrix, Video character recognition.***

I. INTRODUCTION

Content based image retrieval (CBIR) and its extension to videos are research areas which have gained a lot of attention in the recent years. Various methods and techniques have been presented, which allow querying of big databases with multimedia contents (images, videos, etc.) using features extracted by low level image processing methods and distance functions which have been designed to resemble human vision perception as closely as possible. Nevertheless, query results returned by these systems do not always match the results desired by a human user. This is largely due to the lack of semantic information in these systems. Therefore, rather than analyzing the content of video frames when there is text, it is better to segment and recognize the text to obtain semantic information that can match the results desired by the human user. Hence, automatic segmentation and recognition of video text are essential for video annotation and retrieval systems [1-7].

Video text recognition is generally divided into four steps: detection, localization, extraction, and recognition. The detection step roughly identifies text regions and non-text regions. The localization step determines the accurate boundaries of the text lines. The extraction step removes background pixels in the text lines and the text pixels are retained for recognition [2]. There are several algorithms that are reported in the literature for accurate text detection and localization and they have achieved good accuracy even for scene text detection [8-11] and multi-oriented text detection. Therefore, in this work, we use the method that works for multi-oriented scene text reported in [11] to locate text lines with bounding boxes in video images. It is noticed from text extraction and recognition literature that most of the methods use text area or lines detected by the text detection methods for extraction. For recognition, they use OCR engines that are developed for recognizing characters printed on clear backgrounds. Applying these OCR engines directly on video text leads to poor recognition rates, typically from 0% to 45% [3]. This is because text characters in video can be of any grayscale values and are often embedded in frames with complex backgrounds. Traditional OCR engines are good for video frames with high contrast text and simple backgrounds but not for video frames with complex backgrounds. Therefore, character segmentation before extraction to increase the accuracy of video text recognition is performed in [2]. However, this work assumes that characters have uniform color because the objective is to segment graphics text, and not scene text in video. This work also uses a traditional OCR engine and thus is not good enough for extracting both graphics and scene text. Hence, segmentation and recognition steps are important and further research is required to meet the requirements of real time applications such as video event analysis and sports event analysis etc.

In the literature, there are also a few papers on the enhancement of text lines before passing them to OCR engines. Tang et al. [4] proposed a method for video caption detection and recognition based on fuzzy-clustering neural networks, which makes use of both spatial and temporal information. Wolf and Jolion [1] used gradient, morphological information and multiple frame integration for extraction and recognition for graphics text. Recently, a new approach for text detection and extraction from complex video scenes is proposed in [5] based on transient colors between graphics text and adjacent background pixel. Chen et al. [6] proposed a two-step method for text recognition. The first step uses edge information to localize the text. The second step uses features and machine learning to recognize the segmented text. Chen and Odobez proposed [3] using Monte Carlo sampling for text recognition. This method appears to be expensive as it uses probabilistic Bayesian classifier for selecting thresholds. It also requires a sequence of frames to achieve accuracy. Another method [7] for low resolution video character recognition based on holistic approach and a connected component analysis is proposed in. However, it requires a large number of training samples. In addition, there are methods which propose robust binarization algorithms to improve the recognition rate of video character recognition [12-14]. However, these methods focus on graphics text recognition and hence their error rates would be higher if there is scene text in the input images. Recently, Zhou et al. [14] developed a Canny-based binarization method for video text recognition, which achieved a reasonably good accuracy compared to the baseline thresholding methods. However, the assumption that Canny give fair edges in the paper restrict the accuracy of video character recognition.

Most of the previous methods focus on only graphics text and

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.35

IEEE computer society

high contrast text instead of considering both graphics and scene text in video images. The performance of these methods depends on the enhancement step and it is difficult to decide the number of frames used for enhancement. Most of the methods use traditional OCR engines to recognize the characters. Several methods use the text area detected by a detection algorithm for extraction and recognition. Therefore, we conclude that none of the existing methods have a perfect solution to the problem of character segmentation and recognition for both graphics text and scene text.

Hence, in this paper, we propose novel features for video character recognition through hierarchical classification. Prior to recognition, we propose a method for segmenting characters from text lines detected by our text detection method [11]. This segmentation method uses vertical and horizontal cuts to identify the gap between characters. The hierarchical classification is done based on a voting method, which gives 9 sub classes for 62 classes. We propose new properties for each segment given by 8 directional code of the edge character to study the shape of the segments and to obtain distinct features for each class of 62 characters. Finally, we evaluate the proposed method by varying the number of samples image per class for training and testing.

II. PROPOSED APPROACH

Since our intention is to recognize video characters in text lines, we use our text detection method proposed in [11] as it works for text lines in any direction and of different fonts, sizes, backgrounds and contrast. The proposed character recognition method consists of three steps. In section A, we present a method for segmenting characters from text lines detected by the text detection method. Hierarchical classification of 62 characters classes based on structural features is presented in Section B. Features based on the shapes of the characters are proposed for recognition of character in Section C.

A. Segmentation based on Cuts

The output of a text detection method is a collection of bounding boxes, each of which fully covers a text line. Before recognizing the characters, we perform segmentation to separate the individual characters from each other. Inspired by [15], we model the segmentation problem as finding the least-cost cuts from the top row to the bottom row of each bounding box. A cut is defined as a continuous path in which from each pixel, we can move in three directions: left, down and right. This problem is solved by dynamic programming as follows.

Let $f(x, y)$ be the grayscale region inside a bounding box, i.e. it contains one text line. Let $c(x, y, x', y')$ be the cost function between pixels (x, y) and (x', y'). Finally, let $d(x, y)$ be the cumulative distance of the least-cost path from a starting pixel (x_0, y_0) to (x, y).

Initialization:

$$d(x,y) = \begin{cases} 0, & x = x_0 \wedge y = y_0 \\ +\infty, & otherwise \end{cases} \quad (1)$$

Update rule:

$$d(x,y) = min \begin{cases} d(x-1,y-1) + c(x-1,y-1,x,y) \\ d(x,y-1) + c(x,y-1,x,y) \\ d(x+1,y-1) + c(x+1,y-1,x,y) \end{cases} \quad (2)$$

Where

$$c(x,y,x',y') = \begin{cases} |f(x,y) - f(x',y')|, & x = x' \\ k \times |f(x,y) - f(x',y')|, & otherwise \end{cases}$$

We assume that for text to be readable there should be some contrast between text region and background region. The cost function is thus set to the absolute difference between the two gray values to encourage the cut to say in the background region. If it makes a transition to the text region, i.e. it goes through the character, the corresponding cost will be high.

The complexity of this algorithm is $O(mn)$ where m and n are the width and the height of $f(x, y)$, respectively. The advantage of this algorithm is that it allows non-vertical cuts, which are useful for separating touching characters. However, for many cases, a vertical cut is sufficient and thus we set $k = 1.5$ to slightly penalize diagonal moves. The value of k is determined empirically.

Note that the above algorithm finds only one cut starting from pixel (x_0, y_0) ($y_0 = 1$ since we always start from the top row) to the bottom row. To segment all the characters, we run it multiple times and place x_0's every $n / 4$ pixels, which is half the character width estimated based on the height n. Sample segmentation results are shown in section III.A.

B. Hierarachical Classification based on Voting Method

In this work, we have collected a variety of sample character images segmented by the above method (Section A), which vary from 2 to 50 samples for 62 characters which includes 26 uppercase letters, 26 lowercase letters and 10 digits. As a result, we get 62 labeled classes for classification. We first resize the input image to 64×64 pixels (Figure 1(b)) and then compute its Canny edge image (Figure 1(c)). We have observed that the Canny edge image preserves the shape of the characters and resizing is done to standardize the characters of different font sizes. Large size of dataset, low contrast and background variation make video character recognition complex and challenging. In addition, there may be disconnections and background noise due to low contrast. Hence we propose new structural features that are robust to disconnections, noise, font, font size, rotation and scale. The structural features work based on two criteria that are (1) whether the centroid of the edge image falls on the edge itself or not and (2) the outlet in 8 directions from the centroid.

In order to ease the complexity of the recognition problem, we propose a voting method to divide 62 classes into progressively smaller classes in a hierarchical way. If more than 50% of the sample images in the class satisfy the criterion, say criterion 1, the method classifies the whole class into one group without testing the remaining images; otherwise it classifies it into another group (binary tree classification). For instance, the classification rate satisfying criterion 1 for each class of uppercase letters are reported in Table 1. For the uppercase letters A, E, F, G, H, J, K, L, N, P, Q, R, S, V, W, Y and Z the classification rate is more than 50% therefore those characters are classified into one group. For B, C, D, I, M, O, T, U and X the classification rate is less than 50% therefore those characters are classified into another group. Hence this feature is useful in classifying video characters as it is a robust and invariant feature. Note that due to space constraint, we have reported the classification rate in Table 1 only for criterion 1 for uppercase letters. In the same way, we have defined 6 more features for classification. Thus we have a binary tree of 7 levels. In the following figures, the left child shows the classification result when the condition is satisfied and the right child shows the classification result when it is not.

Feature 1 (F_1): This feature is criterion 1 which is tested on characters (Figure 1(d)) after removing small noisy components (Figure 1(c)). The method uses the voting criterion for feature 1 to classify 62 character classes into sub classes as shown in Figure 2.

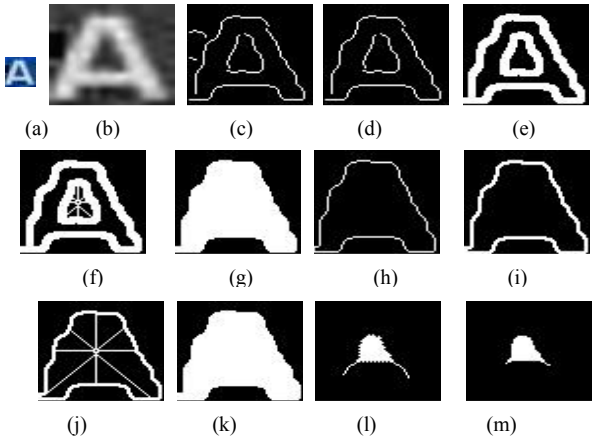

Figure 1. Features for hierarchical classification: (a) Input, (b) Resized, (c) Canny, (d) Filtered, (e) Dilated, (f) 8 directions, (g) Filled, (h) Perimeter, (i) Dilated, (j) 8 directions, (k) Filled, (l) Shrunk and (m) End points removed

Table 1. Classification rates of uppercase letter classes (in %).

A	B	C	D	E	F	G	H	I
98	49.4	2.2	15.2	94.7	97.2	90.4	93.3	3.1
J	**K**	**L**	**M**	**N**	**O**	**P**	**Q**	**R**
100	53.8	97.2	49.5	97.7	6.6	86.6	66.6	97.7
S	**T**	**U**	**V**	**W**	**X**	**Y**	**Z**	
90.4	32.6	15.5	77.7	100	40	72.5	100	

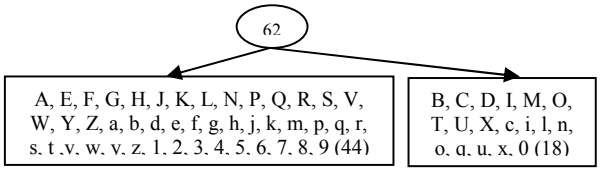

Figure 2. Feature 1 classifies the 62-character set into two subsets

Feature 2 (F_2): An interesting visual observation is to find outlets in 8 directions. From the centroid of the character edge image, if the method finds edge pixels in all 8 directions as shown in Figure 1(f) then we consider the character as having no outlets; otherwise, it has (criteria 2). For characters like J, V, h, v, C, M, U, c, u and n, we can expect an outlet as there is an open space while other characters do not have such spaces. This feature is used for binary classification at the second level as shown in Figure 3. In this way, we have proposed features to classify larger classes into smaller classes. Thus these features are robust and invariant in nature.

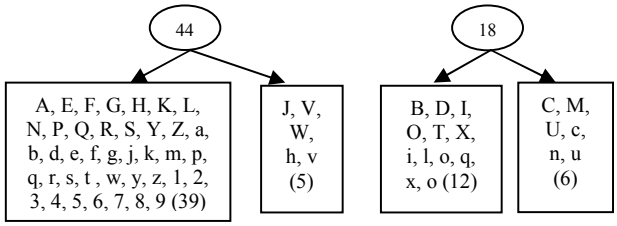

Figure 3. Feature 2 classifies the sets of 44 and 18 characters into two subsets each

Feature 3 (F_3): This feature works based on the perimeter of the character edge image. Before finding the perimeter, the method dilates the Canny edge image in Figure 1(d) as shown in Figure 1(e) and fills the gap as shown in Figure 1(g). For the filled character shown in Figure 1(g), the method finds perimeter as shown in Figure 1(h). The classification is then done as in Feature 2 by testing criterion 1 (Checking centroid falls on it or not). The result of classification is shown in Figure 4. Note that a double enclosure denotes end of classification. A double circle indicates that a single character has been identified, while a double rectangle indicates that the set will not undergo further classification.

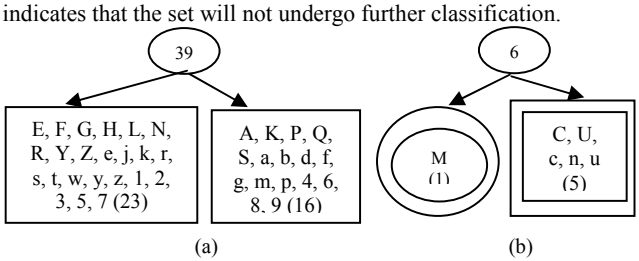

Figure 4 (a)-(b). Feature 3 classifies only two groups containing 39 and 6 characters of the previous level and for other two sets of 12 and 5 characters remain unchanged

Feature 4 (F_4): Before testing criterion 2 (outlet in 8 directions from the centroid), the method dilates the image in Figure 1(h) to get the image in Figure 1(i). It tests the criterion 2 as shown in Figure 1(j) for binary classification at third level. The classification results are shown in Figure 5(a)-(b).

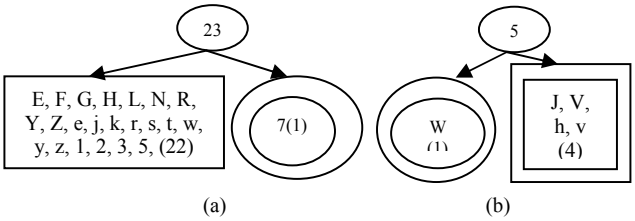

Figure 5 (a)-(b). Feature 4 classifies the sets of 23 characters and 5 characters into two subsets each. It does not classify other sets such as the sets of 16 and 12 characters.

Feature 5 (F_5): For the dilated edge image shown in Figure 1(i), the method again uses flood fill function to fill the character as shown in Figure 1(k). The method then uses the function shrink to shrink the image as shown in Figure 1(l). After that, criterion 1 is used for classification. The classification results are shown in Figure 6.

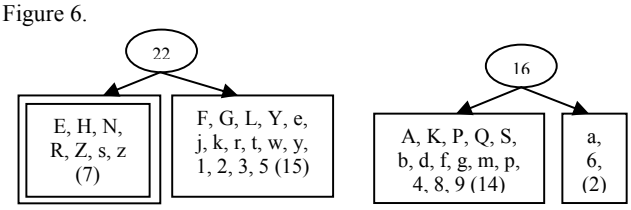

Figure 6. Feature 5 classifies the sets of 22 and 16 characters into two subsets each but other sets of 12 characters remains unchanged.

Feature 6 (F_6): For the result of the shrink function as shown in Figure 1(l), the method tests criterion 2 for classification. The results are shown in Figure 7.

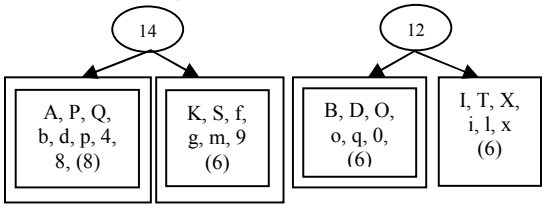

Figure 7. Feature 6 classifies the sets of 14 and 12 characters into two subsets each. The sets of 15 and 2 characters remain unchanged

Feature 7 (F_7): For shrunk characters shown in Figure 1(l), the method removes the two end points using the spur function as shown in Figure 1(m) and then tests criterion 1 for the result shown in Figure 1(m) for classification. The classification results are shown in Figure 8(a)-(c).

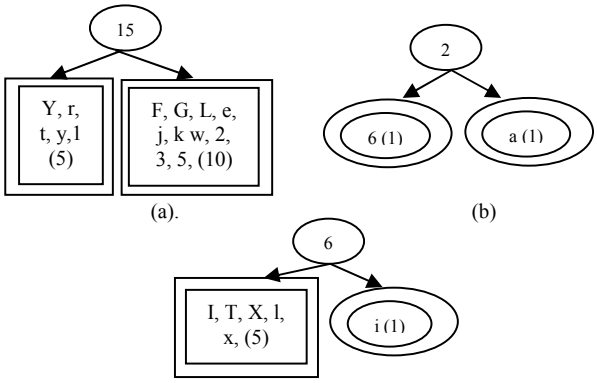

Figure 8 (a)-(c). Feature 7 classifies the sets of 15, 2 and 6 characters into two subsets each

The above classification results in 9 groups. Let G1 = {E, H, N, R, Z, s, z}, G2 = {Y, r, t ,y, 1}, G3 = {F, G, L, e, j, k, w, 2, 3, 5}, G4 = {A, P, Q, b, d, p, 4, 8}, G5 {K, S, f, g, m ,9}, G6 = {J, V, h, v}, G7 ={B, D, O, o, q, 0}, G8 = {I, T, X, l, x} and G9 = {C, U, c, n, u} be the groups obtained by the hierarchical classification. Note that at this stage it is already possible to classify W, 6, a, M, and i.

C. Structural Features for Recognition

This section presents new features based on the shape of each segment determined by 8-direction splitting. For these features, the input is the perimeter of the character edge character image. The method segments the perimeter into 8 subsegments according to 8 directions as show in Figure 1(j) from the centroid of the character edge image. For each segment, we propose new features to study its shape to find distinct features for each of the above 9 groups. The feature extraction process is illustrated in Figure 9 where C and M denote the centroid and the midpoint, respectively, of a segment. E1 and E2 denote the two end points of the segment.

The features are extracted based on the distance between the centroid and the end points, and the midpoint and the end points. Let (D1) be the distance between C and E1, (D2) be the distance between C and E2, D3 be the distance between C and M, D4 be the distance between M and E1 and D5 be the distance between M and E2 as illustrated in Figure 9.

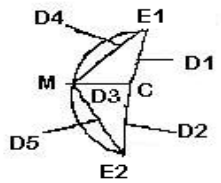

Figure 9. Structural feature extraction for each segment

We propose the following criteria to extract the properties to study the segment shape. If its centroid falls on the segment itself then we consider it as property St_1 else property Cur_1 ("St" and "Cur" stand for straight and cursive, respectively). If D3 < 1 then St_2 else Cur_2, If D1 ≤ D2 then St_3 else Cur_3, If D1 ≤ D3 then St_4 else Cur_4, If D2 ≤ D3 then St_5 else Cur_5, If D3 ≤ D4 then St_6 else Cur_6, If D3 ≤ D5 then St_7 else Cur_7, If D4 ≤ D5 then St_8 else Cur_8.

Based on observation and experiments, we arrive at a total of 16 distinct properties for classification in this work. The assignment of a property to a particular class is by observation. For example, consider property St_4 (D1 ≤ D3) in Table 2 for the representative of class E. This property enforces the condition that segments in images belonging to E class should have. Note that one would expect more segments that satisfy the property D1 ≥ D3 for class E but not that satisfies the property St_4 which is designed to discriminate among the other classes. Hence in Table 2 class E representative has a small value (almost 0). Similar properties for each of the other classes are used to calculate a representative value for that class.

A representative for a class is determined by averaging the number of segments that satisfy a given property. In other words, given a training set for a particular class, the average number of segments over all images that satisfy the property corresponding to that class is the representative number of the class. Thus, a particular class is represented by a single number. Given an unknown character to be recognized, it is given the label of the class whose representative number is closest to the number of segments in the unknown image. Representing a class of images by a single value has been studied earlier in scene category classification [16]. We have computed representatives for 10%, and 50% training samples per class for all 9 the groups. One such example for group 1 is shown in Table 2. Since we know class labels and their representative numbers, we compare the number of segments of an unknown image of the class with representative of that class first. If the group contains two or more same representative values but different properties. For instance, for class {E, H, s, z}, the representatives values are the same but the properties are different. In this context, the number of segments of the images in class z is compared with the representative of class z first and then the representatives of others classes. This criterion helps in classifying characters which have similar shapes and hence our method gives a good recognition rate for all classes. Due to space constraint, we have not given representative tables of other groups. In Table 2, SN denotes number of classes, NoI denotes the number of images in the class, P denotes property, S-10% and S-50% denote the number of samples considered for computing the average value (representative), R-50% and R-10% denote the corresponding representatives.

Table 2. Class representatives for G1 = {E, H, N, R, Z, s, z}

S. N	NoI	Cl	P	S-50%	S-10%	R-50%	R-10%
1	36	E	St-4	18	4	0.00	0.00
2	45	H	St-5	23	5	0.04	0.00
3	44	N	St-8	22	5	6.60	6.40
4	44	R	St-2	22	5	1.63	1.80
5	4	Z	St-6	2	1	13.0	12.0
6	27	s	Cur-6	14	3	0.00	0.00
7	4	z	Cur-7	2	1	0.50	0.00

III. EXPERIMENTAL RESULTS

To evaluate the proposed method, we have considered our own dataset as there is no benchmark dataset for video character recognition. We also noticed that this is the first work to make an attempt to develop video OCR without the help of current OCR

engines and enhancement techniques. We present sample segmentation results and recognition results below.

A. Segmentation Results

(a) (b)

(c)

Figure 10. Sample segmentation results. (a) Input image. (b) Least-cost non-vertical cuts. (c) Cuts that end at the same pixel on the last row are grouped together.

B. Recgonition Results

We evaluate the method by considering 10% and 50% images per class for training (computing the average value) and use the remaining images for testing. The confusion matrix for all the 9 groups is computed. Due to space constraint, in Table 3, we only give the confusion matrix of group 1 for 10% samples per class for training and 90% samples per class for testing. The results in graphic form for 10% and 50% samples per class for training are shown in Figure 11 where we have considered the average of diagonal elements of confusion matrices versus groups to know the performance of the method. It is observed from Figure 11 that there is no much difference in recognition rate for use of 10% and 50% samples per class (refer Table 2). Therefore, the proposed method gives a good recognition rate determined as the average of average of diagonal elements of confusion matrices of 9 groups as shown in Table 4 where NI is number of images used and RR is recognition rate.

Table 3. Confusion matrix for G1 for 10% training samples

	E	H	N	R	Z	s	z
E	93.7	0	0	0	0	6.2	0
H	2.5	90.0	0	0	0	7.5	0
N	0	0	94.8	0	2.5	2.5	0
R	10.2	0	0	89.7	0	0	0
Z	0	0	33.3	0	66.6	0	0
s	0	4.1	0	0	0	95.8	0
z	33.3	0	0	0	0	0	66.6

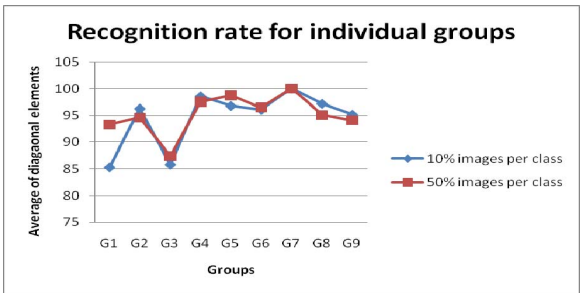

Figure 11. Recognition rate by varying training samples

Table 4. Recognition accuracy for various training sizes

50% samples per class for training		10% samples per class for training	
NI	RR	NI	RR
743	95.2%	174	94.5%

IV. Conclusion and Future Work

This paper proposes a new method for recognition of video characters through hierarchical classification. For recognition of text lines detected by a text detection method, we have proposed a segmentation algorithm which finds least-cost cuts by dynamic programming. Structural features that are invariant to geometrical transformation and robust to noise are proposed for classification and recognition. Voting criterion is adopted to classify the large number of classes into smaller groups based on structural features. In the future, we plan to extend our method for other languages.

Acknowledgment

This work is supported in part by A*STAR grant R252-000-402-305.

References

[1] C. Wolf and J. M Jolion, "Extraction and Recognition of artificial text in multimedia documents", Pattern Analysis and Applications 2003, pp 309-326.

[2] X. Hunag, H. Ma and H. Zhang, "A New Video Text Extraction Approach", In Proc. ICME 2009, pp 650-653.

[3] D. Chen and J. M. Odobez, "Video text recognition using sequential Monte Carlo and error voting methods", Pattern Recognition Letters, 2005, pp 1386-1403.

[4] X. Tang, X. Gao, J. Liu and H. Zhang, "A Spatial-Temporal Approach for Video Caption Detection and Recognition", IEEE Transactions on Neural Networks, 2002, pp 961-971.

[5] W. Kim and C. Kim, "A New Approach for Overlay Text Detection and Extraction from Complex Video Scene", IEEE Transactions on Image Processing, 2009, pp 401-411.

[6] D. Chen, J. M. Odobez and H. Bourland, "Text detection and Recognition in images and video frames", Pattern Recognition, 2004, pp 595-608.

[7] S. H. Lee and J. H. Kim, "Complementary combination of holistic and component analysis for recognition of low resolution video character images", Patten Recognition Letters, 2008, pp 383-391.

[8] D. Doermann, J. Liang and H. Li, "Progress in Camera-Based Document Image Analysis", In Proc. ICDAR 2003, pp 606-616.

[9] J. Zang and R. Kasturi, "Extraction of Text Objects in Video Documents: Recent Progress", In Proc. DAS 2008, pp 5-17.

[10] K. Jung, K.I. Kim and A.K. Jain, "Text information extraction in images and video: a survey", Pattern Recognition, 2004, pp. 977-997.

[11] P. Shivakumara, T, Q. Phan and C. L. Tan, "A Laplacian Approach to Multi-Oriented Text Detection in Video", IEEE Transactions on PAMI, 2011, pp 412-419.

[12] Z. Saidane and C. Garcia, "Robust Binarization for Video Text Recognition", In Proc. ICDAR 2007, pp 874-879.

[13] S. Bolan, L. Shijian and Chew Lim Tan. "Binarization of Historical Document Images Using the Local Maximum and Minimum". In Proc. DAS 2010, pp 159-165.

[14] Z. Zhou, L. Li and C. L. Tan, "Edge based Binarization for Video Text Images", In Proc. ICPR 2010, pp 133-136.

[15] J. Wang, J. Jean, "Segmentation of merged characters by neural networks and shortest-path", Pattern Recognition, 1994, pp 649-658.

[16] P.Shivakumara, Deepu Rajan and Suresh A. Sadanathan, "Classification of Images: Are Rule based Systems effective when Classes are fixed and known?", In Proc. ICPR 2008.