Multimedia Systems 10: 344–355 (2005)
Digital Object Identifier (DOI) 10.1007/s00530-004-0159-y

Caption analysis and recognition for building video indexing systems

Fu Chang[1], **Guey-Ching Chen**[1], **Chin-Chin Lin**[1,2], **Wen-Hsiung Lin**[1]

[1] Institute of Information Science, Academia Sinica, Taipei, Taiwan
[2] Department of Electrical Engineering, National Taipei University of Technology, Taipei, Taiwan
(e-mail: {fchang,ching64,erikson,bowler}@iis.sinica.edu.tw

Published online: 2 February 2005 – © Springer-Verlag 2005

Abstract. In this paper, we propose several methods for analyzing and recognizing Chinese video captions, which constitute a very useful information source for video content. Image binarization, performed by combining a global threshold method and a window-based method, is used to obtain clearer images of characters, and a caption-tracking scheme is used to locate caption regions and detect caption changes. The separation of characters from possibly complex backgrounds is achieved by using size and color constraints and by cross examination of multiframe images. To segment individual characters, we use a dynamic split-and-merge strategy. Finally, we propose a character recognition process using a prototype classification method, supplemented by a disambiguation process using support vector machines, to improve recognition outcomes. This is followed by a postprocess that integrates multiple recognition results. The overall accuracy rate for the entire process applied to test video films is 94.11%.

Keywords: Background removal – Caption tracking – Character recognition – Support vector machines – Prototype classification

1 Introduction

The rapid increase in the use of digital videos in recent years has raised the need for an archival storage system. Such a system, in turn, requires an effective indexing technique for retrieving and browsing video content [2]. Since video captions are rich sources of information, caption-based retrieval has become a popular focus for research into video content retrieval.

The aim of this paper is to provide systematic methods for analyzing and recognizing video captions. We limit ourselves to finding horizontal captions of a single color that stay the same throughout a number of successive frames. The video captions, moreover, are in Chinese, and most characters in the captions are either light with dark borders or dark with light borders. Chinese characters are unique in three ways. First, there are thousands of commonly used characters in Chinese, compared to the 26 uppercase and 26 lowercase letters in English. Second, Chinese characters are much more complicated than English letters. Third, many Chinese characters are composed of more than one component.

To deal with these complexities, we must solve the following problems. First, we must locate text regions in video frames. Second, to separate text from background, we must first binarize video images, that is, turn colored pixels into black and white pixels. Third, because captions extend across frames, we need to detect caption changes within successive frames. Fourth, as video captions are often embedded in complicated backgrounds, we need to solve the background-foreground separation problem. Fifth, character segmentation is crucial for satisfactory character recognition. Sixth, we require a classifier that recognizes each individual character obtained in the segmentation process. Finally, after character recognition, we need a postprocessing method to select the best possible recognition results from successive frames that contain the same text. In general, Chinese characters require delicate treatment in the image binarization, character segmentation, and recognition processes. However, Chinese captions are easier to locate than Roman (and other Western) language captions because Chinese characters are often more complicated than Roman (and other Western) letters.

The remainder of this paper is organized as follows. In Sect. 2, we discuss related works and the main concepts of our proposed methods. In Sect. 3, we propose a caption-tracking scheme for acquiring spatial and temporal information from video frames. Section 4 introduces a method for removing backgrounds, and Sect. 5 describes character segmentation. This is followed, in Sect. 6, by a description of character recognition. In Sect. 7, we present a postprocessing scheme that integrates multiple recognition results. Section 8 contains the end-to-end performance of the entire caption analysis and recognition process. Finally, in Sect. 9, we present our conclusions and discuss four major types of error in our results.

2 Background

In this section, we discuss related works for analyzing and recognizing video captions and present our main ideas for solving the seven major problems listed in the introduction. For an overview of caption analysis and recognition, readers can refer to the survey papers of [18] and [8]. Other references can be found on the following Web site: `http://www.videoanalysis.org/Research_ Topics/Text_ Localization__Text_Segmen/ text_localization__text_segmen.html`.

There are a number of ways to find text in video frames, including those that classify textual and nontextual regions by their textural features [12,17,20], by connected components [1,21,10,28], and by the spatial distribution of object edges [14,27,31]. Kuwano et al. [14] propose a method to identify text in individual video frames that overcomes some shortcomings found in the method proposed by [27]. In this method, they first calculate color gradients between neighboring pixels to obtain edges and then examine adjacent edges with opposite gradient signs to obtain edge-pairs. Text regions can be detected by checking the spatial properties of edge-pairs. Meanwhile, Wu et al. [31] use Gaussian filters to extract character strokes from text regions and then group these strokes into rectangular bounding boxes. This approach is appropriate for high-resolution scanned images, but not for video images that are often in low resolution. The method of [14], on the other hand, is suitable when a significant contrast exists between characters and their background. We focus on this aspect because dark Chinese characters usually have light fringes, and vice versa. Furthermore, we use the edge detection method proposed by [14] to locate text regions because of its lower processing cost and high accuracy rate.

To transform color video images into binary images, some works combine locally adaptive thresholding and global thresholding to obtain better binary results [11,**?**]. These works, however, do not have a *quantitative* basis for evaluating their performance. Our solution to the binarization problem combines the use of the global thresholding method proposed by Otsu [24] and a window-based thresholding method proposed by one of the authors of this paper [4,5]. To evaluate the performance of our hybrid thresholding method, we derive two classifiers from our machine learning method. One classifier uses as training samples the caption characters binarized using Otsu's method, and the other uses the characters binarized using our hybrid method. We apply each classifier to a set of test samples that has been binarized using the corresponding thresholding method, that is, either Otsu's method or the hybrid method. Recognition accuracy constitutes the quantitative basis for evaluating both Otsu's and our binarization methods.

For detecting caption changes, Sato et al. [26] propose a temporal segmentation method, while Lin et al. [21] propose a method that compares character contours across successive frames. Li et al. [17] compare intensity and texture across successive frames, and Lienhart and Wernicke [20] compare vertical and horizontal projection profiles. The last two approaches track stationary as well as moving captions. Our approach, however, is limited to stationary captions. For this purpose, we improve on the method in [21] by using images derived from both character contours and edge pairs, since these two features are well preserved by the aforementioned sharp contrast between characters and their fringes.

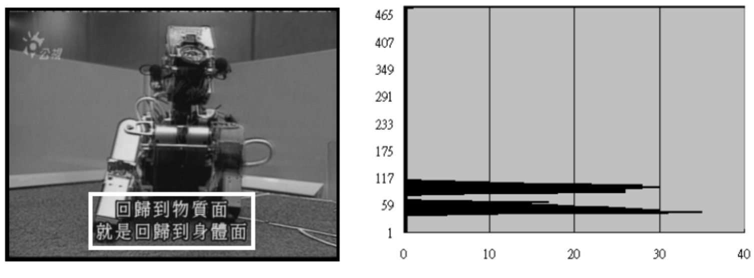
For background removal, some methods use spatial characteristics to filter nontext regions [14,30,31], while others remove nontext regions by means of the temporal redundancy of captions [9,26]. Although some approaches combine spatial characteristics and temporal redundancy to remove nontextual objects, they have difficulty dealing with nontextual objects that are similar to textual objects in spatial characteristics [21, 28]. Lienhart and Wernicke [20] combine text color and temporal redundancy to remove nontextual objects. In this paper, we employ a three-stage process that gradually removes background pixels via spatial characteristics, color constraints, and multiframe information. The accuracy rate increases dramatically, from 30.27% to 92.85%, as we proceed from the first to the last stage.

For character segmentation, Lu [22] describes algorithms for character segmentation, some of which rely solely on pixel projection profiles. These algorithms achieve good segmentation results for Roman letters. Lee et al. [15] use recognition results to select the most suitable segmented points on grayscale images. Sato et al. [26] first segment characters by way of a vertical projection profile and then use a character recognizer to refine the segmented candidates. For Chinese characters that are composed of an indefinite number of components, we propose an algorithm that performs dynamic split and merge operations based on certain parameters derived from the textlines containing those characters. Although this method does not require any recognition process, it achieves a very high accuracy rate of 97.87%.

Most works on video captions do not provide a classification method for recognizing caption characters. Many rely on a commercial recognizer to conduct recognition [19,20,26, 31]. To build a classifier based on machine learning methods using caption characters as part of the training data, we propose a two-stage method that combines a prototype classification method with the support vector machine (SVM) method [29]. SVM is an effective classification method, but it is extremely slow in the training process when the number of class types (character types, in our case) is large. The two-stage method rectifies this by decomposing the original problem into two subproblems. Experimental results show that this method achieves accuracy rates on test data comparable to the SVM and the nearest neighbor (NN) method [7] but has a much lower training cost than SVM and runs faster than both SVM and NN in the testing process.

There are two methods for character recognition postprocessing: selection of the best possible recognition results from successive frames [23] and recognition outcome refinement via a dictionary [26]. The first method works for our application, but the second is not so easily adapted to the Chinese language without first solving the parsing problem, since Chinese words are not separated by white spaces like words in Western languages. For the first method, a recognition score is assumed to exist. In our adaptation, we use ranks rather than the recognition scores of candidates.

As well as providing solutions to the stated problems, this paper provides a systematic solution for detecting, segmenting, and recognizing video captions. Few works in the literature present such a complete solution. To the best of our knowledge, the only exceptions are Sato et al. [26] and Lin et al. [21]. Sato et al. focus on the Roman alphabet, so its solutions for character segmentation and recognition are not suitable for Chinese characters. Although Lin et al.'s work focuses on Chinese captions, it lacks a systematic method for character segmentation. Also, its character recognizer is restricted



a b

Fig. 1. a An image containing a caption region. **b** The number of edge-pairs is plotted on the x-axis, and the vertical position of each row is plotted on the y-axis

to characters of a single font. Our method has a multiple-font recognizer constructed from machine learning techniques and has a 94.11% recognition rate obtained from 26 test films containing 114,590 characters, compared to Lin et al.'s 84.1% obtained from 6 test films containing 7,818 characters.

3 Caption tracking

Caption tracking is comprised of three stages. The first locates text regions within single video frames using spatial edge information, the second uses a binarization method to classify pixels within a text region as either foreground or background, and the third stage uses previously obtained binarized images and edge information to detect caption changes across successive frames.

3.1 First stage: Locating caption regions

We choose the edge-detection method proposed by [14] to locate caption regions. This method calculates color gradients between neighboring pixels. If the gradient is larger than a threshold T ($T = 140$), it is regarded as an edge. When two adjacent edges with opposite gradient signs (i.e., one has a positive and the other a negative gradient) are found within a certain distance D, they form an edge-pair. Parameter D is set at (frame width) $\times$ (20/352). The factor 20/352 stems from the requirement that if the frame width is 352, D must be 20. When a horizontal scan line contains at least M edge-pairs, it is said to be an M-line. M is set at (frame height) $\times$(6/240). When more than N ($N = 10$) contiguous M-lines exist, an area consisting of these M-lines is regarded as a caption region.

Figure 1a is an image captured from one of our test video frames. The area of dense edge-pairs is marked with a white rectangle. The number of edge-pairs per scan line is shown in Fig. 1b.

To evaluate the method adopted for locating captions, we use 100 frames from each video as test data, totaling 2,600 frames. There are exactly 1,792 text regions in these test frames. After implementing an edge detection method, we successfully detect 1,791 text regions, missing only one. As our proposed method only detects the vertical range of caption regions, we consider the detection successful when the detected range completely covers the vertical range of a caption region but does not cover more than 130% of this range. The number of false alarms, that is, the noncaption regions misidentified as caption regions, is 197. The recall rate, calculated by (correctly identified items) / (correctly identified items + missed items), is 99.94%. The precision rate, calculated by (correctly identified items) / (correctly identified items + false alarms), is 90.09%. At this point, we use the recall rate rather than the precision rate because if we miss caption regions at this stage, we will miss them at all later stages. However, we can still filter out false alarms at later stages. To make a performance comparison, we implement the method of [21] designed for finding Chinese caption regions in videos. Using exactly the same parameter values specified by them and adopting our performance metric we measure their recall rate at 84.65% and precision rate at 65.28%.

3.2 Second stage: Image binarization

Our character segmentation and recognition processes use binary images as input. To fulfill this requirement, we transform color images first into grayscale images and then into binary images. For typical video characters, such as Fig. 2a, the intensity of white pixels that are physically closer to background pixels degrades to a greater extent because of decayed signals. Pixels with a degraded intensity complicate binarization because a global threshold method is not sufficient for classifying them. A high global threshold tends to make a character appear broken (Fig. 2b), and a low global threshold makes it appear blurry (Fig. 2c). To solve this binarization problem, we propose a binarization method [4,5] that combines a global threshold method with a window-based method.

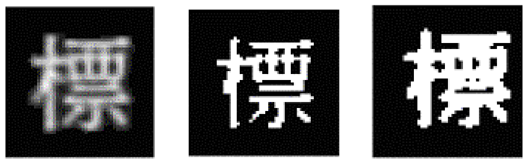

a b c

Fig. 2a–c. A typical video character affected by intensity degradation

A global threshold T is obtained using Otsu's binarization method [24]. Normally, T assumes a value between 100 and 150. For a pixel p to which the window-based binarization method is *not* applied, we classify it as black if $g(p) < T$ or as white if $g(p) > T$. The window-based binarization method is applied to pixels whose grayscales fall within a range between S and T. S is determined as follows: If the population size of pixels whose grayscales fall below T is Ψ, then the population size of pixels whose grayscales fall between S and T is one quarter of Ψ. Normally, S assumes a value between 90 and

Table 1. Recognition accuracy rates using Otsu's binarization method and our method

Otsu's method	Our method
92.74%	95.66%

120. To slightly extend the application range, we apply the window-based binarization method to all pixels that are within a 5×5 neighborhood of any pixel p such that $S \leq g(p) \leq T$.

For each pixel p to which the window-based binarization method is applied, let $W_p(n)$ be the $n \times n$ window centered at p. Let *Mini*(p) = min$\{g(q)$: $q \in W_p(n)\}$ and *Maxi*(p) = max$\{g(q)$: $q \in W_p(n)\}$. We then use $t(p)$ = (*Maxi*(p)-*Mini*(p)) / 2 as the local threshold. Pixel p is classified as white if $g(p) \geq t(p)$ or black if $g(p) < t(p)$.

The window-based binarization method is based on finding the correct window size so that the window centered at each pixel has a good mixture of background and foreground pixels. For this purpose, it is useful to estimate the stroke width of each character and set the window size slightly larger than that width. To estimate the stroke width, one can employ Hadamard multiresolution analysis [4,5]. Hadamard kernels of various scales are applied to each character pixel horizontally and vertically to obtain the strength of each character pixel. The stroke width of a character is then the Hadamard scale at which the sum of the strengths reaches the maximum value. The window size of each character is then set at $W + 1$, where W is the stroke width determined by Hadamard multiresolution analysis.

As noted, the window size is set so that it contains both character and background pixels. Since most Chinese video characters have sharp outlines, window-based analysis should improve the quality of binarized images. For example, Fig. 3a shows an original grayscale image, Fig. 3b is the binary result using Otsu's binarization method, and Fig. 3c is the binary result using our method.

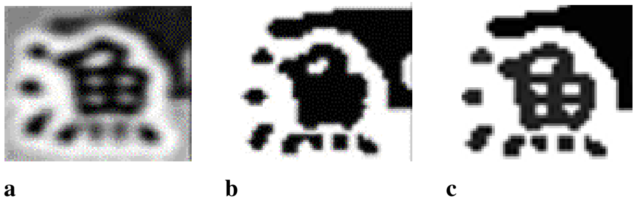


Fig. 3. a Grayscale image of a character. **b** Otsu's global threshold method result. **c** Our method's result

We evaluate the results of Otsu's method and our method by measuring the accuracy rate of character recognizers applied to their binarized output. The test data are comprised of well-segmented character images collected from the same image sources, but binarized using the two different methods. To make a fair comparison, we evaluate each binarization method with a recognizer whose training data are binarized using the corresponding method. The test data consist of 29,267 well-segmented character images. The recognition accuracy results are listed in Table 1.

3.3 Third stage: Caption change detection

To detect caption change, we need to find collections of successive video frames that contain the same text. Here, we compare three methods for caption change detection. The first method uses binarized images as the basis for comparison. It examines pixels in two successive frames that have the same coordinates but differ in binary values; that is, one of them is white and the other is black. These pixels are called *opposite-valued* pixels.

In Fig. 4, we display binarized captions in three successive frames. Captions in Figs. 4a and b contain different texts. The proportion of opposite-valued pixels in these two different-text frames is 50%. Captions in Figs. 4b and c contain the same text. The proportion of opposite-valued pixels in these two same-text frames is 30%.

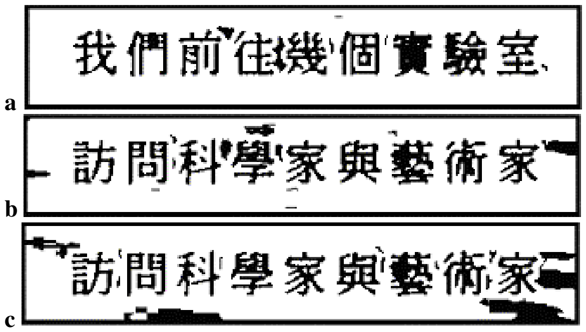


Fig. 4a–c. Binarized captions in three successive frames, where **a** and **b** contain different texts, and **b** and **c** contain the same text

The second method, proposed by Lin et al. [21], compares images using the contours of connected components as foreground pixels; the images are called contour images (Fig. 5). For these images, the proportion of opposite-valued pixels in different-text frames (Figs. 5a and b) is 65%, while the proportion of those pixels in same-text frames (Figs. 5b and c) is 25%.

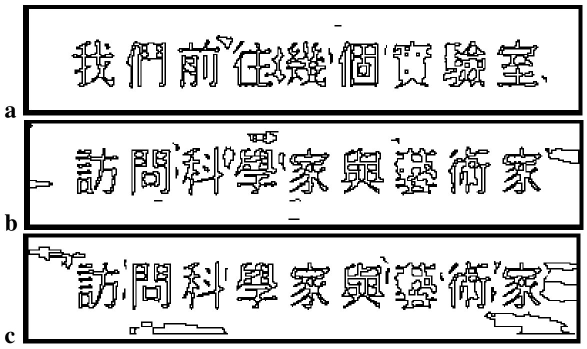


Fig. 5a–c. Contour pixels are used as foreground pixels in the images

Recall that an edge-pair is a pair of edge points of opposite gradient signs. In the third method, we first form images using edge-pairs as foreground pixels; the images are called edge-pair images. We then combine contour images with edge-pair images through an AND operation (Fig. 6). In the combined images, the proportion of opposite-valued pixels in different-text frames (Figs. 6a and b) is 62%, while the proportion of

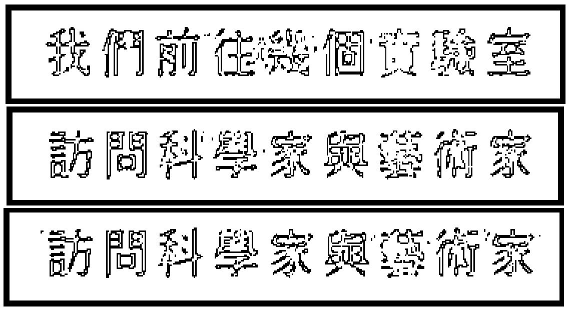


Fig. 6a–c. A pixel that is both a contour pixel and a point in an edge-pair constitutes a foreground pixel in the images

those pixels in same-text frames (Figs. 6b and c) decreases dramatically to 7%.

To evaluate these three methods, we use 26 MPEG files of 640×480 resolution as test data, each of which lasts 5 to 7 min, making approximately 150 min in total. There are 2,672 true caption changes in the test videos. When the proportion of opposite-valued pixels exceeds 25%, the two successive frames are judged to contain different texts. The results are summarized in Table 2. Because of these results, we choose the last method for detecting caption changes. This method detects 2,624 caption changes and 108 false alarms. There are also 48 missed changes.

4 Background removal

In the binarized caption region shown in Fig. 7b, many background pixels have the same binary value as foreground pixels. The objective of background removal is to eliminate such pixels.

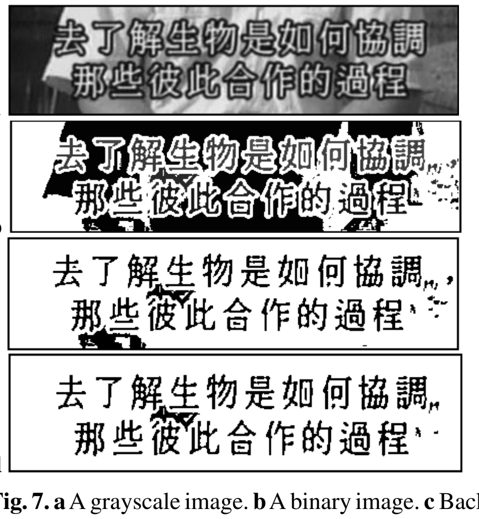


Fig. 7. a A grayscale image. **b** A binary image. **c** Background removal using the size and height-to-width ratio constraint. **d** Background removal using the size, height-to-width ratio, and color constraint

The first two stages of background removal are applied to caption regions in single frames. For each frame, we record all connected components (hereafter referred to as components for short) found in binarized caption regions (Chang et al. [6]). In the first stage, when a component C has height H_c and width W_c, and the video frame has height H_f and width W_f, we filter out C if (i) $H_c > 1/7 \times H_f$ or $W_c > 1/7 \times W_f$, (ii) $H_c < 4$ and $W_c < 4$, or (iii) $H_c/W_c > 10$ or $H_c/W_c < 0.1$. The remaining components are shown in Fig. 7c.

In the second stage, we first compute three standard channel values: average R-channel, average G-channel, and average B-channel, where each is the average of all the component pixels. We also compute the average channel values of each component and filter out a component if any of its average channel values differs from the corresponding standard channel value by a threshold T_c (T_c = 30). The remaining components at the end of this stage are called *characterlike* components and are shown in Fig. 7d.

The third stage works on multiple frames containing the same caption. To reduce the processing time, we only examine caption regions in five samples that are evenly spaced temporally. These regions are called *sampled regions*.

We first derive the *standard region* from the sampled frames. A pixel in the standard region is black if and only if at least four corresponding pixels (i.e., pixels having the same coordinates) in the sampled regions fall within a characterlike component. Figures 8a–e show the five sampled regions. Figure 8f shows the derived standard region.

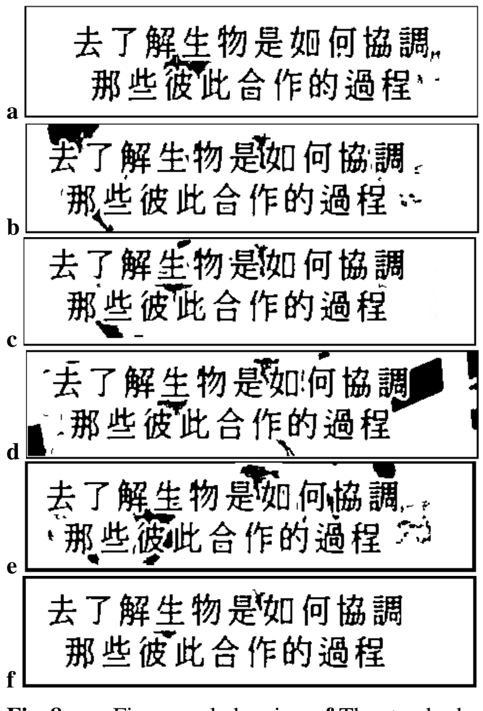


Fig. 8. a–**e** Five sampled regions. **f** The standard region derived from them

To further refine the sampled regions, we use the standard region as a reference region. Any component in the sampled region that overlaps significantly with foreground pixels in the standard region is regarded as a genuine component. The details are as follows. For each component C in a sampled region, we examine the subset C' of C that consists of pixels whose corresponding pixels in the standard region are black. If C' contains more than 60% of the pixels in C, we retain C;

Table 2. Results of our caption change detection method

	Hits	Misses	False alarms	Precision rate	Recall rate
Binary image	1,031	1,641	2,846	26.59%	38.59%
Contour image	2,052	620	739	73.52%	76.80%
Contour image AND edge-pair image	2,624	48	108	96.05%	98.20%

Table 3. Accuracy rates of character recognition at each stage of background removal

Test images obtained	Recognition accuracy
At the end of the 1st stage	30.27%
At the end of the 2nd stage	89.78%
At the end of the 3rd stage	92.85%

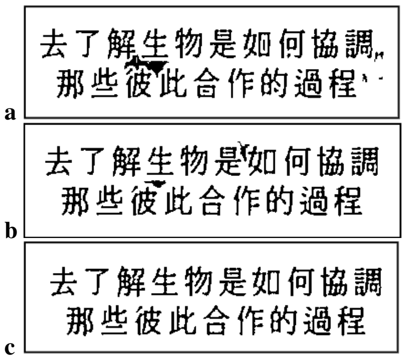


Fig. 9. a A sampled region. **b** The standard region derived from sampled regions. **c** The refined region derived from the sampled and standard regions

otherwise, we remove C from this region. After the refinement process shown in Fig. 9, the refined region has a better image quality than the standard region.

To evaluate the effects of the three stages of background removal, we prepare a set of frames that contain 2,624 different captions and 124,000 characters as input. Each caption exists within five frames, and there are 13,120 frames in total. At the end of each stage we collect the output images as test data, which gives us three sets of test images. The recognition accuracy rates for these three sets are listed in Table 3.

5 Character segmentation

If a Chinese character has several components whose bounding boxes overlap, it can be treated as a single box (Fig. 10a). However, about 15% of Chinese characters remain composite in that their component boxes do not intersect with each other (Fig. 10b). If two composite characters are neighbors, we need to decide which component boxes should be combined as a character box. Occasionally, the touching of characters, or partial characters, causes their bounding boxes to become unusually large. Thus, we also need to divide these boxes into correct parts.

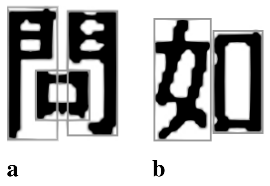


Fig. 10. a A composite character comprised of three components whose bounding boxes overlap. **b** A composite character comprised of two components whose bounding boxes do not overlap

5.1 Textline construction

A useful clue for character segmentation is that characters of the same textline have similar heights and widths (Fig. 11), so we can use the standard size of characters as the guideline for combining or dividing boxes. First, we construct a textline and then infer the standard width of the characters from that textline.

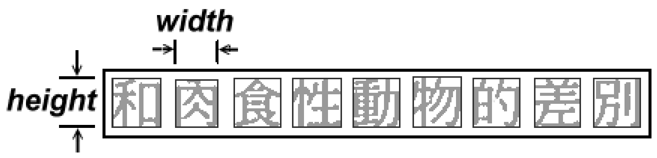


Fig. 11. Characters in the same textline that display similar characteristics

To start textline construction, we first randomly choose a component as the starting point and expand horizontally to merge adjacent components that are properly aligned along the top and bottom margins. After this process, two or more textlines may still overlap (Fig. 12a); however, we only select textlines that do not overlap with other textlines (Fig. 12b).

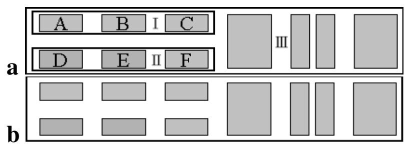


Fig. 12. a Textline I is composed of components A, B, and C, and textline II is composed of D, E, and F. **b** Textline III contains textlines I and II, so only textline III is retained

After textline construction, we further estimate the common character height of each textline. We first merge two component boxes into one if their vertical projections overlap and let these boxes cast votes to boxes of similar height. The common character height is the height of the box that receives the maximum number of votes. We assume that the common character height is the *standard character width* of a textline because the height-to-width ratio of a typical Chinese video character is close to 1.

5.2 Character segmentation

We first project all component boxes in a textline onto the horizontal axis and form the nonzero intervals on the projection profile (Fig. 13). Working from left to right, we merge and split these intervals. Components in the merged intervals are regarded as constituents of a character.

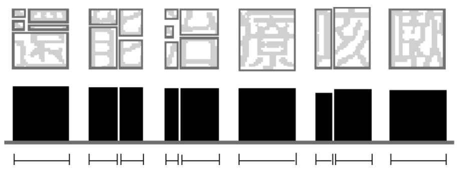


Fig. 13. Nonzero intervals in the projection profile; each interval contains at least one component

Because punctuation marks can cause problems in this process, we eliminate them before we start character segmentation. As shown in Fig. 14, a comma and the left part of a character (你) might combine into a proper, though false, character box. Punctuation marks are easily identified by their heights, widths, cross-counts, and the standard gap in the textline. The standard gap G in a textline is defined as the most common gap size found in the textline. The cross-count of a character box is obtained as follows. Our procedure works from left to right on each scan line. If two adjacent pixels in a scan line L_i have different binary values, we increase $\mathrm{C}(L_i)$ by one [$\mathrm{C}(L_i) = 0$ initially]. The cross-count of a character box is then defined as $\sum_{i=1}^{n} \mathrm{C}(L_i)/n$, where n is the number of scan lines found in the box. Character box B is then identified as a punctuation mark if it conforms to all of the following rules. (1) The height and width of B are less than half the common character height of the textline. (2) The gaps between B and the boxes to its left and right are larger than $0.6\times G$. (3) The average value of the cross-count for the character is less than four.

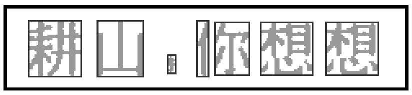


Fig. 14. A textline with Chinese characters and a punctuation mark

When punctuation marks have been eliminated, we perform character segmentation. For interval J, we must decide whether to divide J, to leave J as is, or to join J to neighboring intervals. To decide this, we consider three cases. Let the standard character width of this textline be denoted as W.

Case 1. Length $\mathrm{l}(J)$ of J exceeds $1.2\times W$.
We look for a point P at which to divide J. P is the point between the positions $0.8\times W$ and $1.2\times W$ measured from the left margin of J, at which the projection profile has the minimum value (Fig. 15). When J is divided, we reset J to be the first section of the original interval. The second section is considered an untreated interval, and the procedure stops.

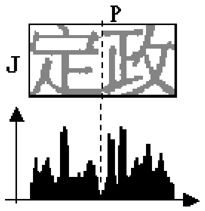


Fig. 15. Two touching characters are divided at a point P that lies at the minimum value of the projection file

Case 2. Length $\mathrm{l}(J)$ falls below $0.8\times W$.

Subcase 2.1. If the adjacent interval of J is identified as a punctuation mark, we stop the procedure.

Subcase 2.2. If not, we extend J by joining adjacent intervals to J, until $\mathrm{l}(J)$ exceeds $0.8\times W$. If $\mathrm{l}(J)$ still falls below $0.8\times W$ and a punctuation mark is encountered, then we stop the procedure. If $\mathrm{l}(J)$ falls below $0.8\times W$, but the adjacent interval K has a length exceeding $1.2\times W$, we cut K at a point P between the positions $0.8\times W$ and $1.2\times W$, measured from the left margin of J (Fig. 16). When K is divided, we extend J up to point P. The remaining section of K is now an untreated interval, and the procedure stops.

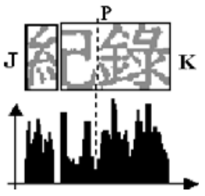


Fig. 16. A partial character touches a full character. The latter is cut at point P, which lies at the minimal value of the projection file

Case 3. Length $\mathrm{l}(J)$ falls between $0.8\times W$ and $1.2\times W$.

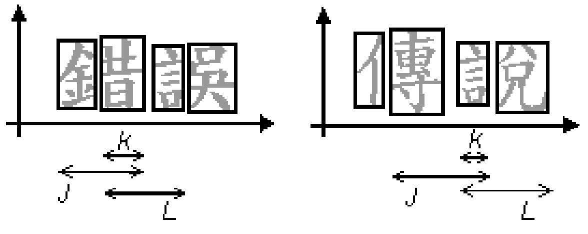
Subcase 3.1. If a further extension of J results in a length exceeding $1.2\times W$ or encounters a punctuation mark, we stop the procedure.

Subcase 3.2. If not, we extend J up to a point before its length exceeds $1.2\times W$, or before we encounter a punctuation mark. We then let k be the rightmost interval of J and denote the maximal extension of k as L. If $\mathrm{l}(L) < 0.8\times W$, we decide that J is a proper interval and stop the procedure; otherwise, we have to decide whether k belongs to J or to L by comparing the space to the left of k with the space to its right. If the former space is smaller, we decide that J is a true character; otherwise, we decide that L is a true character (Fig. 17). If we reach the end of the procedure and there are still untreated intervals in the textline, we restart the procedure, this time using the leftmost untreated interval as J.

We use caption regions output from the background removal process (Sect. 3) to test the performance of our segmentation method. There are 13,100 regions and 114,590 characters in the output. The results are listed in Table 4.

6 Character recognition

For character recognition, we employ a two-stage classification method for pattern recognition within a large set of



a b

Fig. 17. a The space to the *left* of k is smaller than the space to the *right* of k; k is attributed to J. **b** The space to the *left* of k is larger than the space to the *right* of k; k is attributed to L

Table 4. Character segmentation performance

Number of true characters (N_t)	114,590
Number of segmented characters (N_s)	115,512
Number of correctly segmented characters (N_c)	112,151
Recall rate (N_c/N_t)	97.87%
Precision rate (N_c/N_s)	97.09%

class types. At the first stage, we match each unknown character against a set of prototypes. At the second stage, an SVM method is used to work on the top-k classes that have been selected in the prototype-matching process. The prototypes used for the first stage are derived from an offline training process that constructs prototypes from training samples. This training process has two advantages. First, it produces a small set of prototypes compared to the set of training samples. Second, if SVM is used solely for classifying an object into one of N classes, it has to solve $N(N-1)/2$ binary classification problems in the training phase. The prototype learning process dramatically reduces the number of binary classification problems to be solved, thereby making it possible to use SVM for large-scale character recognition. The core of our two-stage classification method is, therefore, the learning mechanism that constructs prototypes in the offline training process.

6.1 The prototype-construction problem and its solution

We want to construct a set of prototypes, or standard patterns, from training samples to serve as the basis for classification. When the prototypes are specified, to classify a character **c** in the testing process is to find a prototype **p** nearest to **c**. The character **c** is then assigned the class type of **p**.

Prototypes are derived from training samples. We assume that a set of samples is given and labeled with their class type. Each training sample is a vector in n-dimensional Euclidean space. In this space, let *dist*(**x**, **y**) be the square of the Euclidean distance between two vectors **x** = $(x_1, x_2, \ldots, x_n)$ and **y** = $(y_1, y_2, \ldots, y_n)$. A prototype can be any vector in the same space. For a set P of prototypes, where **p** is a prototype in P, the *attraction domain* of **p** is the collection of all samples **s** for which **p** is the nearest prototype, i.e., dist(**s**, **p**) $<$ dist(**s**, **q**) for all other prototypes **q** in P. For a set of prototypes to serve as a *solution* for the prototype-construction problem, we require that the attraction domain of each prototype be *homogeneous*, namely, it contains only same class-type samples. This condition reflects the requirement that each prototype serves as the representative of its neighboring samples.

The learning algorithm we propose for solving the prototype-construction problem dynamically alters both the number and the location of prototypes. For this reason, it is called a *dynamic algorithm* (DA), which is stated as follows:

1. Initiation: for each class type C, the initial C-prototype is the statistical average of all C-samples.
2. Absorption: for each sample **s**, find its nearest prototype **p**. Let *type*(**x**) be the class type of **x**, where **x** is either a sample or a prototype. If *type*(**s**) = *type*(**p**) and *dist*(**s**, **p**) $<$ *dist*(**s**, **q**) for all other prototypes **q**, then **s** is *absorbed*; otherwise, it is *unabsorbed*.
3. New prototype construction: for each class type C, let the number of C-prototypes be *num*(C). If there are unabsorbed C-samples, construct *num*(C)+1 C-prototypes; otherwise, C-prototypes remain the same as in the previous iteration.
4. Process termination: if there are still unabsorbed samples, go to step 2; otherwise, stop the whole process.

In step 3, the construction of new C-prototypes is performed as follows. First, we select a sample from the unabsorbed C-samples. Then, we use the selected sample and existing C-prototypes as *seeds* and employ the K-means clustering method to form new C-prototypes. To select an unabsorbed C-sample, we focus on a set U consisting of unabsorbed C-samples that are *not* themselves C-prototypes. We let each sample in U cast a vote to its nearest sample in U and select the sample in U that gains the highest number of votes. To construct new C-prototypes, we apply the K-means method to group all C-samples according to the following procedure. The K-means method uses the seeds as initial cluster centers. It then assigns each sample to the cluster whose center is nearest and resets each cluster center as the statistical average of all elements in that cluster. This procedure continues until all cluster centers no longer change. The final cluster centers are then assigned as new prototypes.

DA terminates within a finite number of iterations, where the number of iterations is the number of times step 3 is executed. This is because the total sum of the distances between samples and nearest prototypes of the same class types decreases by at least a constant number in each iteration. A sample thus remains unabsorbed for only a finite number of iterations.

6.2 Disambiguation

Although the prototype-matching method achieves very high accuracy rates for k nearest prototypes when $k > 1$, there is a noticeable gap between the top-k and top-1 accuracy rates. The disambiguation procedure bridges this gap. There are some requisites for the disambiguation. In the training process, we must determine which class types can be mistaken for another in the prototype classification. These types are always paired and are thus referred to as *confusing pairs*. For these pairs, we have to specify *reassessing schemes* using an SVM method. We use these schemes in the testing process to reassess the top-k candidates for each object.

Recall that, in the prototype construction, we must determine the nearest prototype for each training sample **s**. At the end of the construction, we find k ($k = 3$ in our application) nearest prototypes for each **s**. The *class types* of these k prototypes are referred to as *candidates* of **s**. We collect the pairs (C_i, C_j), where C_i and C_j are, respectively, the class types of the ith and jth nearest prototype of **s** for $1 \leq i, j \leq k$.

For each confusing pair (A, B) and its training samples, we use SVM to create a reassessing scheme. The purpose of the SVM is to provide *decision functions* for classifying objects into class A or class B, where the parameters and support vectors that appear in the decision function are derived from an optimization problem using training samples of A and B as components. Details are given in [29]. For handwritten character recognition, we adopt the dual formulation of the optimization problem using a first-degree polynomial as the kernel function. In [29], comparisons of SVM and other methods for classifying UPS handwritten numerals are given, and SVM is shown to perform competitively.

After completing the offline process by determining the reassessing scheme for each confusing pair, we can address the online process. Suppose that an object O is given and its first k candidates are already found. We apply reassessing schemes to all confusing pairs found within the top-k candidates of O. When the confusing pair is (A, B) and the unknown object is classified as A, then A scores one unit. When all the confusing pairs in the candidate list are reassessed, we reorder the involved candidates. The candidate with the highest mark is ranked first, the candidate with the second highest mark is ranked second, and so on. If two candidates receive the same score, their relative positions remain the same as before. We then rearrange the involved candidates according to their assigned ranks.

6.3 Experimental results

To build a classifier for multiple-font video characters, we collect 147,118 character images as *training* samples derived from two major sources: (1) 84,000 computer-generated characters compressed using a JPEG algorithm to simulate blurred video characters and (2) 63,118 characters segmented from video frames (we remove poorly segmented ones). There are 2,973 class types in our collected samples. We call this set of class types the 2,973-class. Each character image is normalized to a bitmap of 64×64 pixels and represented as a vector. Each vector component takes as its value the number of black pixels found within a 4×4 cell. Since there are 256 (16×16) nonoverlapping cells within a 64×64 bitmap, the dimension of each vector is 256. From this large set of training samples we also extract a smaller set of training samples that consists of 340 class types. We call this set the 340-class. Each class type contains the same number of samples as the full training data. In order to test the accuracy of our method and to compare it with alternative methods, we prepare 29,263 character images as test data for the 2,973-class. These images are obtained from video frames and are not part of the training samples. Within this set of images, there are 18,569 samples whose class types fall in the 340-class. The 18,569 samples are thus used as test data for the 340-class. All these samples are well-segmented character images. They serve to test the performance of classification methods only. The end-to-end performance of the whole caption analysis and recognition process is given in Sect. 8.

As shown in Table 5 for the 340-class data and the 2,973-class data, the DA prototype construction process produces 1,033 prototypes and 8,484 prototypes, respectively. The proportions of prototypes to training samples are 2.2% and 5.8%, respectively. For disambiguation, we use the top-3 candidates to form confusing pairs. In so doing, we produce 8,782 pairs and 76,514 pairs, respectively. The ratio of confusing pairs to the total number of character pairs is 15% and 1.7%, respectively. These results show that our method effectively reduces the large set of training data to a small set of prototypes and reduces the even larger set of all possible pairs of class types into a tiny set of confusing pairs.

In addition to the DA and the two-stage classification methods, we also measure three other classification methods using the same training and test data. These are 1-NN [7], one-against-one SVM [13], and DAGSVM [25]. The 1-NN method regards all training samples as prototypes and classifies a test sample as the type of its nearest prototype. Both the one-against-one SVM and DAGSVM solve $N(N$-1)/2 binary classification problems in the training phase, where N is the number of class types. In the testing phase, the one-against-one technique conducts $N(N$-1)/2 classifications for each test sample. DAGSVM employs a directed acyclic graph that has $N(N$-1)/2 nodes and N leaves. The number of classifications for each test sample reduces to N-1 in DAGSVM. The accuracy rates of 1-NN, DA, one-against-one, DAGSVM, and the two-stage methods are listed in Table 5. The missing items are the accuracy rates of the two SVM methods in the 2,973-class data because it is too costly to complete their training phase. From the results we can see that all methods have comparable accuracy rates. In all our SVM experiments, we use LIBSVM [3] as it provides SVM tools for solving binary classification problems.

In Table 6, we list the time consumption (in seconds) and the number of support vectors of all methods. The computing environment is an Intel Pentium IV CPU, 2.4 GHz with 256 MB RAM. The training time, testing time, and number of support vectors for the SVM methods in the 2,973-class data are extrapolated from those obtained in the 340-class data. As shown in Table 6, the time needed to complete the training phase of SVM in the 2,973-class is estimated at 2,483,760 s, or 690 h, while that of the two-stage method takes only 36,600 s, or 10 h. The results in Tables 5 and 6 show that the two-stage method not only achieves comparable accuracy rates to the other three methods but is also more effective in computation than all of them (with the exception of 1-NN training, which takes no time).

7 Multiframe integration

To illustrate our proposed method, we examine the recognition results for the five sampled caption regions shown in Fig. 18. There are five recognition results, each corresponding to one sampled region. Correctly recognized characters are placed in white boxes and misrecognized characters in gray boxes. Recognition results that do not correspond to true characters are denoted by "&" or "&&."

Table 5. Training and testing results of the classification methods

	1-NN		DA		1-against-1 SVM		DAGSVM		Two-stage	
	Acc	# Pr	Acc	# Pr	Acc	# CP	Acc	# CP	Acc	# CP
340 class types	**99.04%**	47,537	98.44%	**1,033**	99.03%	57,630	99.03%	57,630	98.76%	**8,782**
2,973 class types	**97.91%**	147,118	97.05%	**8,484**		4,417,878		4,417,878	97.87%	**76,514**

(Acc: accuracy rates; #Pr: number of prototypes; # CP: number of confusing pairs)

Table 6. Training time, testing time, and number of support vectors

	1-NN	1-against-1 SVM			DAGSVM	Two-stage		
	testing	Training		Testing	Testing	Training		Testing
	Time	Time	# SVs	Time	Time	Time (DA+SVM)	# SVs	Time (DA+SVM)
340 class types	1,163	32,400	1,193,310	1,521	51	**5,640 (600+5,040)**	**209,702**	**24 (23+1)**
2,973 class types	3,960	2,483,760	9.1×10^7	116,599	446	**36,600 (18,480+18,120)**	**1,745,709**	**284 (278+6)**

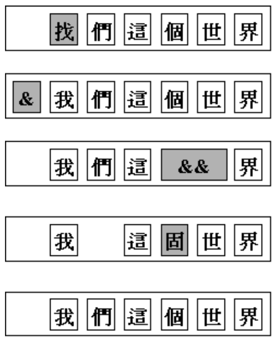


Fig. 18. Recognized results of five sampled frames that contain the same caption

We register characters by their bounding boxes and then align bounding boxes A and B on two successive frames if the size of A∩B is at least 90% the size of A and of B. At the first stage, we group aligned bounding boxes that contain the same character. Those groups that contain more than three elements are assumed to correspond to true characters and are retained. The remaining groups are assumed to correspond to spurious characters and are eliminated. As shown in Fig. 19, eight groups, G1–G8, are obtained from results in Fig. 18. Groups G2–G7 are retained because they contain at least four elements; G1 and G8 are eliminated since they contain only one element.

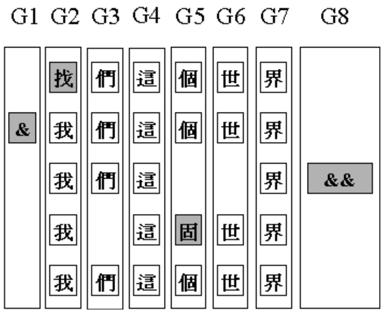


Fig. 19. An example of grouping characters based on location and size

Table 7. Scores assigned to candidates

Rank 1	Rank 2	Rank 3
5	3	2

At the second stage, we redetermine candidates for each retained group based on the candidates obtained from sampled regions. Let us take G2 in Fig. 20 as an example. This group has five elements, each corresponding to a sampled region. For each element, we consider its top-3 recognition candidates, C1–C3. We want to derive candidates for G2 from all top-3 candidates in the five frames.

Although 15 top-3 candidates are found, there are only 4 class types. We want to assign each candidate a score and sum up the scores contributed by candidates of the same class type. The score assignment for each candidate is made according to its rank, as listed in Table 7. The rationale for this assignment is that a class type serving twice as a rank 2 candidate should score higher than a class type that serves only once as a rank 1 candidate. Similarly, a class type serving twice as a rank 3 candidate should score higher than a class type that serves only once as a rank-2 candidate.

The class type with the highest score is ranked first, the class type with the second highest score is ranked second, etc. The outcome of candidate redetermination for group G2 is shown in N1 through N3 in Fig. 20.

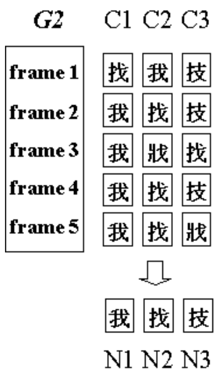


Fig. 20. Candidates for G2 are derived from the top-3 candidates for the five elements of G2

Table 8. Accuracy rates before and after multiframe integration

Candidates	Before multiframe integration	After multiframe integration
Top-1	93.33%	94.11%
Top-2	93.63%	94.50%
Top-3	93.75%	94.71%

8 End-to-end performance of caption analysis and recognition

Our test data are derived from a set of 26 videos collected from TV programs on the Public Television Service station in Taiwan. Each lasts from 5 to 7 min. The videos are in MPEG2 format and have a resolution of 640×480 pixels. The end-to-end process being tested combines our proposed methods for locating caption regions, binarization, caption change detection, background removal, character segmentation, character recognition, and multiframe integration. Our process selects 13,100 individual frames, as well as 114,590 characters from the test data. At the end of multiframe integration, we are left with 2,624 integrated regions and 22,778 characters. Since multiframe integration removes spurious characters, it improves recognition accuracy rates. The results are shown in Table 8.

Note that multiframe integration improves the top-1 accuracy rate by 0.78%. The time consumption of each process is shown in Table 9, where the time is measured in terms of seconds per unit of work item. If a process works on a single frame, then a unit of work item is one frame. If it works on multiple frames, a unit of work item is a set of successive frames over which the same caption text resides. The computing environment is a PC with a Pentium III 1.0-GHz CPU with 128 MB RAM. We spend approximately 1,560 min of processing time to detect and recognize texts in the 26 videos, which last 156 min and contain 13,100 frames and 114,590 characters.

Table 9. Time consumption of each process per unit of work item

Process	Unit	Run time (s)
Locating caption regions	Single frame	0.32
Binarization	Single frame	0.19
Caption change detection	Single frame	0.076
Background removal – single frame	Single frame	0.072
Background removal – multiframe	Multiple frames	0.39
Character segmentation	Single frame	0.072
Character recognition	Single frame	0.5
Multiframe integration	Multiple frames	0.193

9 Conclusion

In this paper, we have proposed various techniques for building video indexing systems using captions as sources of information. We summarize these techniques as follows. (1) Locating caption regions: our method extracts caption regions using edge pairs extracted from video frames. (2) Binarization: our method combines a global thresholding method with a window-based method to binarize video images. (3) Caption change detection: our method tracks caption changes using contours and edge-pairs. (4) Background removal: our method uses size and color constraints, as well as integrated information from multiple frames, to filter out background components in video frames. (5) Character segmentation: we use a dynamical split-and-merge method to segment characters. (6) Character recognition: we use a two-stage classification method that combines a prototype classification technique and an SVM technique. (7) Multiframe integration: our method integrates multiple recognition results to determine final output candidates.

When we combine all these techniques to analyze and recognize Chinese video captions, we obtain 94.11% as the top-1 accuracy rate and 94.71% as the top-3 accuracy rate from test data consisting of 26 videos. These figures represent the end-to-end performance of the combined processes.

Finally, we identify four major types of errors that occur in our end-to-end results. The first occurs when binarized characters connect to background objects, so that they fail to be segmented (Fig. 21). This connection occurs when a “leak” exists in the character fringe, causing the characters to become part of a large background object. This character is thus removed in the background removal process. The second type of error occurs when noises are included in segmented characters in the character segmentation process (Fig. 22). The third

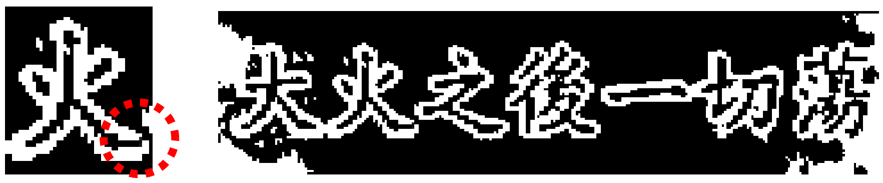


Fig. 21. *Left panel*: The character connects to its background at the *encircled area*. *Right panel*: Context in which this character appears

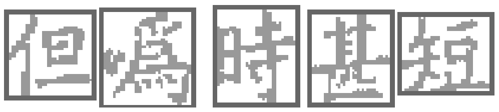


Fig. 22. The second character in this caption contains a background noise

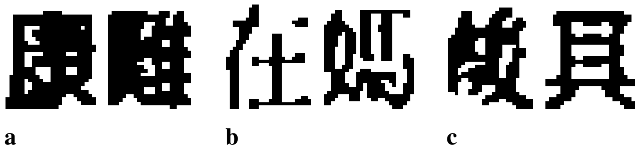


Fig. 23. a Blurry characters. **b** Disconnected characters. **c** Distorted characters

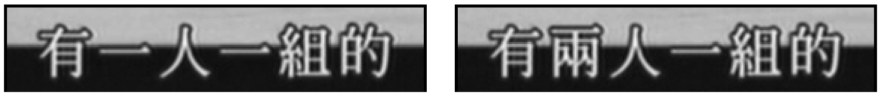


Fig. 24. Two different caption texts differ in only one character (the second character in each text)

error occurs when the binarization process produces blurry, disconnected, or distorted characters (Fig. 23), causing misrecognition of these characters. The fourth type of error occurs in caption change detection. This happens when there is very little change between different captions (Fig. 24). This type of error causes further errors in multiframe integration.

References

1. Antani S, Crandall D, Kasturi R (2000) Robust extraction of text in video. In: Proceedings of the IEEE international conference on pattern recognition, 1:831–834
2. Aslandogan YA, Yu CT (1999) Techniques and systems for image and video retrieval. IEEE Trans Knowl Data Eng 11:56–63
3. Chang CC, Lin CJ (2001b) LIBSVM – A library for support vector machines. `http://www.csie.edu.tw/~cjlin/libsvm/`
4. Chang F (2001) Retrieving information from document images: problems and solutions. Int J Doc Anal Recog 4:46–55
5. Chang F, Liang KH, Tan TM, Hwang WL (1999) Binarization of document images using Hadamard multiresolution analysis. In: 5th international conference on document analysis and recognition, Bangalore, India
6. Chang F, Chen CJ, Lu CJ (2004) A linear-time component-labeling algorithm using contour tracing technique. Comput Vis Image Understand 93:206–220
7. Dasarathy BV (1991) NN concepts and techniques, nearest neighbor (NN) norms: NN pattern classification techniques. IEEE Press, New York, pp 1–30
8. Doermann D, Liang J, Li H (2003) Progress in camera-based document image analysis. In: Proceedings of the IEEE international conference on document analysis and recognition, pp 606–616
9. Hua XS, Yin P, Zhang HJ (2002) Efficient video text recognition using multiple frame integration. In: Proceedings of the IEEE international conference on image processing, 2:397–400
10. Jain AK, Yu B (1998) Automatic text location in images and video frames. In: Proceedings of the IEEE international conference on pattern recognition, 2:1497–1499
11. Kamada H, Fujimoto K (1999) High-speed, High-accuracy binarization method for recognizing text in images of low spatial resolutions. In Proceedings of the 5th international conference on document analysis and recognition, pp 139–142
12. Kim EY, Kim KI, Jung K, Kim HJ (2000) A video indexing system using character recognition. In: Proceedings of the international conference on consumer electronics, pp 358–359
13. Knerr S, Personnaz L, and Dreyfus G (1990) Single-layer learning revisited: a stepwise procedure for building and training a neural network. In: Neurocomputing: algorithms, architectures and applications. Springer, Berlin Heidelberg New York
14. Kuwano H, Taniguchi Y, Arai H, Mori M, Kurakake S, Kojima H (2000) Telop-on-demand: video structuring and retrieval based on text recognition. In: Proceedings of the IEEE international conference on multimedia and expo, 2:759–762
15. Lee SW, Lee DJ, Park HS (1996) A new methodology for gray-scale character segmentation and recognition. IEEE Trans Pattern Anal Mach Intell 18:1045–1050
16. Li H, Doermann D (1999) Text enhancement in digital video using multiple frame integration. ACM Multimedia 1:19–22
17. Li H, Doermann D, Kia O (2000) Automatic text detection and tracking in digital video. IEEE Trans Image Process 9:147–156
18. Lienhart R (2003) Video OCR: a survey and practitioner's guide. Kluwer, Dordrecht
19. Lienhart R, Effelsberg W (2000) Automatic text segmentation and text recognition for video indexing. Multimedia Syst 8:69–81
20. Lienhart R, Wernicke A (2002) Localizing and segmenting text in images and videos. IEEE Trans Circuits Syst Video Technol 12:256–268
21. Lin CJ, Liu CC, Chen HH (2001) A simple method for Chinese video OCR and its application to question answering. Int J Comput Linguist Chinese Lang Process 6:11–30
22. Lu Y (1995) Machine printed character segmentation – an overview. Pattern Recog 28:67–80
23. Mita T, Hori O (2001) Improvement of video text recognition by character selection. In: Proceedings of the IEEE international conference on document analysis and recognition, pp 1089–1093
24. Otsu N (1979) A threshold selection method from gray-scale histograms. IEEE Trans Syst Man Cybern 1:62–66
25. Platt JC, Cristianini N, Shawe-Taylor J (2000) Large margin DAG's for multiclass classification. In: Advances in neural information processing systems. MIT Press, Cambridge, MA, pp 547–553
26. Sato T, Kanade T, Hughes EK, Smith MA, Satoh S (1999) Video OCR: indexing digital news libraries by recognition of superimposed captions. Multimedia Syst 7:385–395
27. Smith MA, Kanade T (1997) Video skimming and characterization through the combination of image and language understanding techniques. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Puerto Rico, pp 775–781
28. Shim JC, Dorai C, Bolle R (1998) Automatic text extraction from video for content-based annotation and retrieval. In: Proceedings of the international conference on pattern recognition, 1:16–20
29. Vapnik V (1995) The nature of statistical learning theory. Springer, Berlin Heidelberg New York
30. Wong EK, Chen M (2000) A robust algorithm for text extraction in color video. In: IEEE international conference on multimedia and expo, 2:797–800
31. Wu V, Manmatha R, Riseman EM (1999) TextFinder: an automatic system to detect and recognize text in images. IEEE Trans Pattern Anal Mach Intell 21:1224–1229