A Framework for Reliable Text Based Indexing of Video

R. Kasturi **S. Antani** **D. J. Crandall** **V. Y. Mariano**
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
kasturi@cse.psu.edu

Abstract

In this paper we describe our recent research efforts towards reliable and automatic generation of indices for use in content understanding of video. Following our earlier research in temporal shot segmentation of video, we have developed a comprehensive system framework for segmenting an unconstrained variety of text from general purpose broadcast video. In addition, the framework also contains a novel tracking and a binarization algorithm. Also developed to be a part of the above framework are other modules, viz. a novel scene text segmentation method, and a novel text segmentation method which extracts uniform colored text from still video frames. We have thoroughly evaluated the methods which form a part of our framework against a fairly large dataset. The framework applies a battery of methods for reliable localization and extraction of text regions. Towards this, we have developed methods for fusing the results from different methods. More recently, we have extended our interest to localizing and extracting stylized text from video and determining the lifetimes of the video text events. Results from the above research are presented in this paper.

1 Introduction

The use of digital video is becoming increasingly ubiquitous. Today, many homes are maintaining personal media archives. Large media archives are also being maintained by several organizations with an interest in commerce, entertainment, medical research, security, etc. Additionally, there has been a growing demand for image and video data in applications, due to the significant improvement in the processing technology, network subsystems and availability of large storage systems. Use of digital video involves capture, compression, archival, indexing, retrieval, querying, browsing, transmission, and viewing. While capture, compression and archival issues are being addressed by better hardware, transmission is the subject of recent advances in communication technologies and viewing is something we are all used to. Indexing, retrieval, querying and browsing, on the other hand, will require automated methods to understand the content of digital video. Content-based information retrieval from such digital video databases and media archives is a challenging problem and is rapidly gaining widespread research and commercial interest.

For retrieval purposes the video may be either annotated and indexed manually or be indexed using automated content description methods. Not only is manual indexing a challenging and cumbersome task, but also suffers from possibly incomplete, subjective and summarial content descriptions. The latter method can solve many of these problems. Several automated methods have been developed which attempt to access image and video data by content from media databases [1]. Providing semantic access to visually rich and temporally linear information is a challenging task. A popular approach to address this problem has been to temporally segment video into subsequences separated by shot changes, gradual transitions or special effects such as fade-ins and fade-outs [2, 3]. A story board of events that occurred in the video can thus be created by selecting a key frame from each (or significant) subsequence. The video can now be queried with visual queries that use color, texture or activity. This is a pseudo-semantic approach to video content description, wherein the human interpretation of color, texture and/or motion define the content. The next step in content-based indexing of digital video is to localize, extract, and recognize objects contained in it. An example of such an object is the visual text appearing in the video data.

1.1 Text as a Video Index

There is a considerable amount of text occurring in video that is a useful source of information. The presence of text in a scene, to some extent, naturally

describes its content. If this text information can be harnessed, it can be used along with the temporal segmentation methods to provide richer content-based access to the video data. The text in video frames can be classified broadly into two large categories - *caption* or *artificial, overlay* text and *scene* text. Caption text comprises of text strings that are generated by graphic titling machines and composited on the video frame during the editing stage of production. This text could also be graphical elements with text contained within them or graphic effects using text. It is placed intentionally by the program editor to provide information of the subject being discussed. Examples of such text are found as credit titles, ticker tape news, information in commercials, etc. Scene text, on the other hand, occurs naturally in the scene being imaged. The image of this text may be distorted by perspective projection, be subject to the illumination conditions of the scene, be susceptible to occlusion by other objects, suffer from motion blurring etc. It can also be on planar as well as non-planar surfaces such as the text on soft drink cans.

A number of research efforts are on to create video storyboards or abstracts in a digital library context. Such efforts naturally concentrate on artificial text since they deal with video from content creators or producers, whose structure and intent are well known [4, 5]. Artificial text can hence serve as a key to the visual content. In addition, any scene text is also likely to be highly correlated with the story being depicted. Thus, the detection and recognition of text from unconstrained, general-purpose video is an important research problem. An indexing system that seeks to comprehensively label or index video by detecting, localizing and recognizing text in the frame must handle both kinds of text in digital video.

In this paper, we describe our research in indexing video through reliable localization and extraction of text in video. We have developed a multi-threaded framework for this purpose [6, 7]. This framework applies a battery of text extraction methods, on MPEG-1 video and JPEG images, in order to add reliability in segmenting text from video. Some of these methods are novel methods developed by us, some contain enhancements made by us on algorithms published in the literature, while others are our implementations of original work by other authors. We have also developed a novel scene text extraction algorithm [8] and a novel algorithm for detection of uniform colored text from video [9]. In addition, the framework also contains a novel tracking and a binarization algorithm [10]. We have evaluated the methods which form a part of our framework against a fairly large dataset [11]. From our evaluation, we conclude that the results of the algorithms vary greatly. Common elements affecting the results are the size of text, the contrast between the text and the background, stroke width, the background image of which the text is a part, etc. Such a scenario points towards developing methods for combining the strengths of a variety of text extraction methods for achieving better results. We have developed methods for fusing the outputs of various text methods and are in the process of enhancing it further [12].

Thus far, the video text segmentation methods have used the assumption that the size of the text remains rigid. The methods also assume that the text blocks move in a predictable fashion. There is a large amount of text appearing in video that does not comply with these temporal assumptions. Moving text effects are often used with caption text to attract viewer attention. Such effects may cause text to change size, perspective, inter-character (word) distance, or color over time. Text strings may rotate or spin. All of these effects would cause text extraction systems to fail. A large amount of scene text also violates these assumptions. We call such text as *stylized* text. It is clear that in order to handle such a wide variety of scene and caption text, more sophisticated text segmentation and tracking algorithms are required. We are in the process of developing methods to address such text. In addition, a system to extract text from video must also determine the *lifetime* or extent of the text event over time. The lifetime of the text event will mark the first and last frame at which the text appeared. This will enable the system to index the video better. We are in the progress of developing methods for determining these.

The remainder of the paper is organized as follows. Section 2 highlights other attempts for extracting text from video. In Section 3, we describe the system framework. The text localization methods are described in Section 4. Tracking and binarization methods are described in Section 5. Section 6 presents the performance evaluation results. Finally, we conclude with Section 7.

2 Related Work

This section presents methods for extracting text from images and video that are published in the literature. There has been a growing interest in the development of methods for detecting, localizing and segmenting text from images and video. There has been relatively more work done on the detection and recognition of artificial text, but even here the literature is sparse on work that deals with video. More work has been done on the extraction of text strings from images and many of the schemes for artificial

text recognition in video are modifications of work originally done for static images.

2.1 Caption Text Extraction

Yeo [13] has proposed a method for detecting caption text that involves computing differences between *a priori* selected corresponding regions of consecutive frames. Changes in this region are assumed to be due to caption events, large shot changes having been filtered out. Sato et al [14] describe a system for performing OCR on video caption text in the context of a digital news archive. Text is localized by looking for clusters of edge pixels that satisfy aspect ratio and other criteria.to increase its resolution. Messelodi and Modena [15], extract text from book cover images. They use simple homogeneity properties to separate text from other image components and further correct their extraction through estimation of orientation and skew correction of text lines. Li and Doermann [16] extract text from digital video keyframes. They use the heuristic that the texture for text is different from the surrounding background to identify text regions. Wavelets are used for feature extraction and a Neural Network is used for decisions. They also present an algorithm for tracking moving text. The tracker assumes that text is mostly rigid and moves in a simple, linear manner. Wu et al [17] describe a scheme for finding text in images. They use texture segmentation to localize text, edge detection to detect character strokes and join strokes to form text regions. Chaddha et al [18] have developed a method to detect text from JPEG images. The sum of the absolute values of a set of DCT coefficients is computed. This measure reflects the high spatial frequency content of blocks containing text. Zhong et al [19] like Chaddha et al use the DC coefficients available in the MPEG I-frames. Those coefficients that highlight the vertical and horizontal frequencies are summed and then thresholded to detect text blocks.

Shim *et al* [20, 21] present a method to detect caption text from MPEG compressed video frames by identifying homogeneous regions in intensity images, forming positive and negative images by double thresholding and applying heuristics based on text characteristics for eliminating non-text regions. The text regions are validated using the temporal redundancy property of text in video. In [22, 23] methods for locating text in complex color images is presented. They have proposed a method for quantizing the color space using peaks in the histogram before performing segmentation. Their other method uses the heuristic of high horizontal spatial variance to localize text. Kim [24] also proposes text localization method for video images similar to the spatial variance method of Zhong [22]. Suen and Wang [25] propose a method for segmenting uniformly colored text from a color graphics background. They assume that all characters have the same color, an assumption that does not hold in general. Hase et al [26] propose an extraction algorithm for character strings. Their approach is directed towards binary document images. Lopresti and Zhou [27] use a similar method to segment text from images found on the Internet. Hauptmann and Smith [4] localize text in video using the heuristic that text regions consist of a large number of horizontal and vertical edges in spatial proximity. Lienhart and Stuber [28] describe a system for automatic text recognition in digital video that works on pre-title sequences, credit titles and closing sequences with title and credits. LeBourgeois [29] presents a system for multifont OCR from gray level images. The method is a modification of run length smearing to segment and recognize text. Mitrea and de With [30] propose a simple algorithm to classify video frame blocks into graphics or video based on the dynamic range and variation of gray levels within the block. Gargi et al [6] describe an algorithm for localizing text in a video frame. The method localizes bounding boxes of horizontal text strings, assuming that each character is composed of a number of segments and that the characters within the string are separated. Other approaches for detecting text in images and video are found in [31–36].

2.2 Scene Text Extraction

Ohya et al [37] describe a method to recognize characters in scene images. They use local gray level thresholding to segment the image and localize text regions by looking for high contrast, uniform gray level of a character, and uniform width. They also use the results of the OCR stage to improve their extraction result—if the Chinese OCR algorithm they use does not find a good enough match, they reject the character candidate region. There is work on the recognition of vehicle license plates [38, 39] from video which shares some of the characteristics of scene text. However, these approaches make restrictive assumptions on the placement, contrast or format of the license plate characters. Cui and Huang's approach [39] takes the advantage of using the information from multiple frames and also correcting for perspective projection distortion. Winger et al [40] discuss the segmentation and thresholding of characters from low-contrast scene images acquired from a hand-held camera. Their data set includes images with low contrast, poor and uneven illumination. Our implementation of the algorithm does not find it to perform well in general purpose video. Communications with the authors suggests that the parameters were fine tuned to individual images on a small dataset.

3 System Framework

A study of the literature reveals that no complete video text extraction system has been developed. Additionally, it is seen that no single algorithm is robust for detection of an unconstrained variety of text appearing in the video [11]. Most methods have been developed to extract text from complex color images and have been extended for application to video data. However, these methods do not take advantage of the temporal redundancy in video. Further study of the methods presented in the literature, presented in detail in Section 2, reveals that the methods assume that the text regions are in high contrast with the background, are composed of one consistent color or gray-level or form a major component in the image. In general, the use of a few rigid assumptions about the nature of text in video forms a weak heuristic. This study of the state of the art was the motivation behind the development of the framework for reliable extraction of text from video.

The video text extraction problem is divided into four main tasks, viz. detection, localization, tracking and binarization. The detection and localization tasks have been merged because there is a significant overlap between the detection and localization processes. A spatio-temporal algorithm fusion module has been proposed for aggregating the decisions of the multiple localization algorithms over multiple frames. The tracking stage is used for temporally validating the text localization. Also, the caption text and scene text segmentation tasks are separated. Extraction of scene text uses an algorithm developed by Gandhi [41] that uses the assumption that scene text lies on a plane in the 3-D world and that camera motion exists. Our approach is to use a battery of different methods employing a variety of heuristics for detecting, localizing and segmenting both caption and scene text. The system also takes advantage of the temporal nature of video and uses the fact that the text data lasts over several frames for providing robust text detection, specifically by performing algorithm fusion in a spatio-temporal manner. The system framework is described in the following section.

An object oriented approach has been adopted in the design of the software prototype. A multi-threaded design has been adopted to allow maximum flexibility. The reasoning stems from the desire that the detection and localization processing of new frames not stall because of a time consuming tracking or segmentation of older frames. The POSIX standard pthreads are used which are lightweight and portable between IRIX, Solaris and other flavors of UNIX. Figure 1 shows the design of the framework. The main components are:

- The **control thread** is the overall data and process control center. It creates a **Frame Queue** and spawn a **readframe** thread. This thread is also responsible for spawning the desired number of **detection-localizion threads**, one per algorithm, the **fusion thread**, the **tracking thread**, and the **binarization thread**.
- The **Frame Queue** contains all the information for both unprocessed and partially processed frames. It provides a data repository for all the threads in the system while maintaining the temporal seqentiality of the data.
- The **readframe thread** reads frames from video stream and adds it to the **Frame Queue**. It provides an abstraction to the data stream type. It presently can read in MPEG-1 bitstreams and JPEG images and can be extended to handle Motion JPEG compressed video.
- The **detection/localization threads** are the implementations of the detection and localization algorithms selected from the literature and the novel algorithms developed here.
- The **tracking thread** tracks a given bounding region over a set of frames.
- The **segmentation thread** binarizes a localized text instance to make it suitable for OCR.
- The spatio-temporal **algorithm fusion thread** fuses the results of various methods to result in a single text instance.
- The **output thread** uses the information in the **Frame Queue** to write output in one of multiple formats: a binarized image suitable for OCR, the original frame with localized text marked by a box, or a ViPER[1] compatible ASCII data file which lists the bounding boxes for each element, etc. ViPER (Video Performance Evaluation Resource) is a Java based tool developed at the laboratory for Language and Media Processing in Center for Automation Research at the University of Maryland for ground-truthing of video and evaluating the performance of content extraction methods.

4 Text Localization Algorithms

Of the methods seen in the literature, only those methods which we judged to be promising were selected. The selection was based on their applicability to general purpose video, use of features, ease of

[1] ViPER: `http://documents.cfar.umd.edu/LAMP/Media/Projects/ViPER`

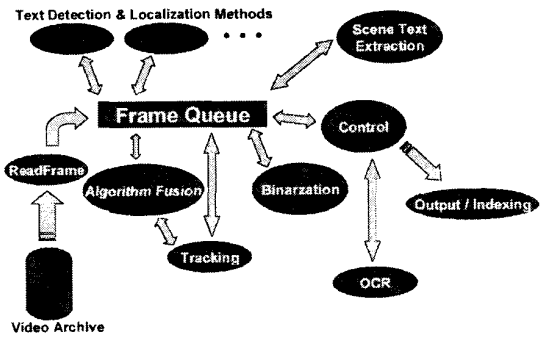


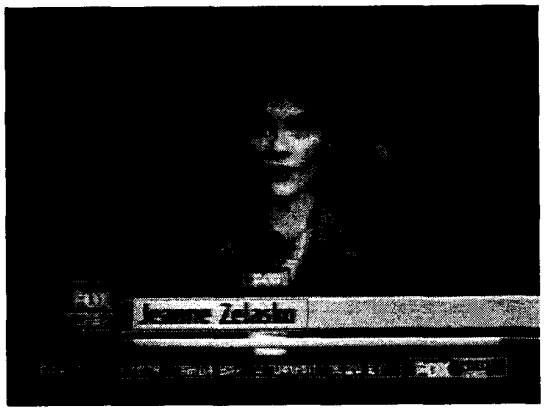
Figure 1: Text Localization/Extraction Framework

implementation and speed of detection. In addition to work done by others, we also include algorithms developed by us for evaluation. The algorithms chosen for evaluation are: **Method A** [6], **Method B** [29], **Method C**: based on initial idea published in [30], **Method D**: enhanced from initial idea published [18], and **Method E** [9]. Details on other methods can be found in the original publications cited above. We include details on the modifications here. Sample text localization results are presented in Figure 2.

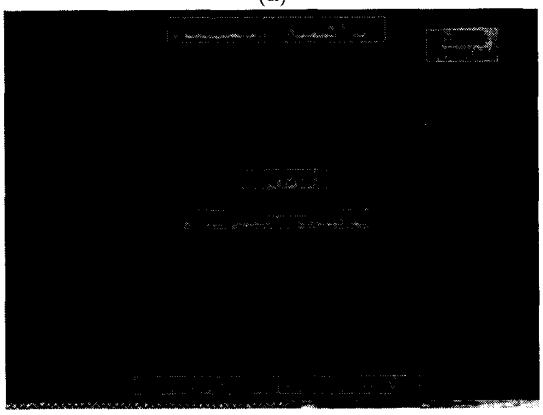
4.1 Modified Algorithm : Method C

A simple algorithm [30], originally proposed to classify video frame 4x4 pixel blocks into graphics or video based on the dynamic range and variation of intensity within the block. This method was developed to achieve higher compression for TV picture signals. The method operates on the premise that in the graphic regions in the frame, many adjacent pixels have the same luminance values or have regions of very high dynamic range. The dynamic range of a block is defined as the absolute difference between the maximum and the minimum intensity in a 4x4 block.

This method is modified to classify blocks as text or non-text. As with the original method the number of pixels in a 4x4 block that have similar gray levels is counted and the dynamic range is computed. If the number of gray level blocks is less than a parameter and the dynamic range of the block is either greater than or a distinct thresholds or is 0, the block is classified as a text block. This change in condition follows from the fact that text has a high number of edges. Thus, the number of pixels with similar intensity levels will be small and the dynamic range will be zero only on the character stroke which typically has near uniform intensity. Additionally, the modified method excludes the boundary regions from its operating space on the frame image.



(a)



(b)

Figure 2: Sample Text Localization Results

4.2 Modified Algorithm : Method D

This method [18] was originally proposed for classifying JPEG image blocks as text or non-text. It has been modified to work on MPEG-1 I-, B-, and P-frames and determined appropriate threshold empirically on our data. The method has been further refined to use an iterative thresholding scheme to reduce the number of false alarms. MPEG B- and P-frames need to be decompressed before Discrete Cosine Transform (DCT) can be reapplied to them for use with this method. For this the `fastct`[2] algorithm [42] was used.

The method uses texture energy to classify 8x8 blocks as text or non-text and works as follows. A subset of the 64 possible DCT coefficients produced during the MPEG encoding process is chosen. For each block, the sum of the absolute values of these coefficients is compared to a threshold to categorize it as text or non-text. Using these blocks as seed blocks, a series of decreasing thresholds is iteratively applied from high (150) to low (30) and the appearance of more and more text blocks as the threshold is lowered is observed. Blocks that are classified as

[2]fastct: `http://dmsun4.bath.ac.uk/dcts/fastdct.html`

text at a particular threshold are left in if they also have a 8-neighbor that was classified as text at the previous higher threshold. The motivation for doing this is that text regions usually have at least one of their component blocks detected at 150. So the text region can be enlarged by lowering the threshold without creating as many false positives. Any blocks with no neighbors on the left or right are removed. This is from the heuristic that the text is horizontal or if vertical, is fairly wide. Any other blocks which appear to be due to a sharp luminance change between two large homogeneous regions are discarded. This is done by computing the mean of average luminance given by the DC term of the DCT coefficients three blocks on either side of a target block. The mean energy of these blocks is also computed. If the average luminance of the three on the right is greater than that of the left by a certain threshold, and the energies of the blocks are below a threshold, we conclude that this block was found because of a sharp luminance cliff and it is discarded. Finally the aspect ratio constraint is used to filter out false alarms.

4.3 Novel Method : Method E

The algorithm visits every *Interval* rows in the image. *Interval* is set small enough to be able to detect very small text and large enough so as not to consume too much time. In our experiment we set $Interval = 3$. Given a row R on the image, we want to determine whether or not R passes through the middle of a text region.

Clustering in $L^*a^*b^*$ space: The pixels of R are transformed and clustered in the perceptually uniform $L^*a^*b^*$ color space using hierarchical clustering. The algorithm first assigns each pixel as a cluster and the distance of pairs of clusters are stored in an array. Two clusters A and B are merged if $||\mu_A - \mu_B||$ is minimum and for each pixel p in $A \cup B$, $||p - \mu_{A \cup B}|| < MaxClusterRadius$, where μ_Z is the mean $L^*a^*b^*$ vector of cluster Z and $||\cdot||$ is the weighted Euclidean norm. The weighted norm was used to achieve a slight invariance to lightness (weights: $L^* = 0.8, a^* = 1.1, b^* = 1.1$). In our experiments, we set $MaxClusterRadius = 10$ (ranges: $L^* = 0 \ldots 100$, $a^* = -97 \ldots 88$, $b^* = -100 \ldots 88$). Merging continues until no two clusters can be merged.

4.3.1 Determining bounding rows

Each cluster C is tested to see if it contains pixels belonging to text. Locating the bounding rows (top and bottom rows of text) is the first step (Fig. 3). The cluster points are marked back on row R to create streaks $S_i, i = 1 \ldots N_s (numberofstreaks)$ of pixels in the row R. Then all pixels in the entire image are examined and each pixel with a value within the range of values represented in the cluster are colored with a value of T. All other pixels are marked T'.

We now try to find out if there are bounding rows above and below R which may contain horizontal text. Given a pair of adjacent streaks S_i and S_{i+1}, we find R_a – the first row above R in which the segment covering S_i and S_{i+1} is colored T'. We also find R_b – the first row below R in which the segment covering under S_i and S_{i+1} is colored T'. The R_a of each pair of adjacent streaks is computed and collected in an alignment histogram H_a, where the bins are the rows of the image. H_b is computed in the same way by taking all the R_b's. We declare the existence of a bounding row B_a if at least 60% of the elements in H_a are contained in three or fewer adjacent histogram bins. B_b's existence is computed in the same way from H_b. If B_a and B_b exists, *height* is defined as their difference.

If the cluster C contains text pixels, then B_a and B_b would mark the text block's upper and lower row boundaries, and *height* would define its vertical dimension. Figure 3 illustrates the computation of B_a, B_b and *height*.

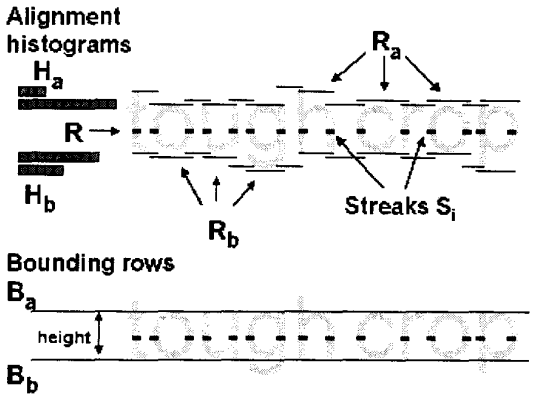


Figure 3: Computing the bounding rows. One of the color clusters in row R are marked as short streaks and pixels of the text "tough crop" lie within the range of values of the cluster. Each segment of the bounding top (R_a) and bottom (R_b) rows are shown separated for clarity even when they are actually on the same row.

Finding text blocks: We look for text blocks using heuristics on *height* and the lengths and gaps of the short streaks. Streaks longer than *height* are discarded and added to the gaps. Gaps longer than *height* are considered not part of a text block. The remaining regions are now smaller blocks with short streaks. If a block's width is greater than $1.5 * height$

and the number of short streaks inside is greater than 3, then it is considered a text block, otherwise it is discarded. Finally, the text block is expanded a few pixels to the left and right to ensure full coverage of the characters at the ends.

Figure 4 shows how the text block "For generations" is detected. The pixels of row R (passing through the middle of text) are clustered in color space. One of the color clusters is marked black. Pixels in the image having similar color as the black ones are marked white. On the left side of the image, the two alignment histograms H_a (above R) and H_b (below R) are used to mark the bright bounding rows B_a and B_b. The short streaks marked black and the *height* between the bounding rows are used to find the text block. The two black streaks on the right were not included in the text block because their gap from the other streaks is greater than *height*.

Figure 4: Analysis of a video frame and detected text block.

Fusing the detected text blocks: It was observed that other color clusters were caused by the presence of text. The characters' color "shadows" and the pixels in the transition from text foreground to background result in other detected text blocks which largely overlap with the foreground text block. All the detected text blocks are fused (set union) to come up with the final regions of text.

4.4 Localization of Stylized Text and Event Determination

It is observed that while the size, orientation, color, *etc.* of a text event may change over time, the basic shape of its characters remains constant. This property can be exploited to determine whether two text boxes correspond to the same text event. We analyze two consecutive frames at a time. First, the text box localization algorithm, Method D, described above is applied to each frame. Oriented text instances are made horizontal by applying a simple rotation transformation. A text binarization algorithm is next applied on each text instance. We used the binarization algorithm developed in our earlier work [10]. This algorithm is tailored for the special challenges of binarization of text in video frames, including low resolution, complex background, and unknown text color. Connected component analysis is performed on the binarized text to locate individual characters. The contour of each character is traversed and stored as a chain code. Each chain code is then parameterized as two 1-D functions $\theta(t)$ and $r(t)$, that represent the angle and distance of each point on the boundary from a reference point, respectively. A Gaussian filter is then applied to both functions,to smooth out any noise introduced by imprecise binarization. The resulting functions represent a signature of the shape of a given character. From this shape, feature points are extracted. We use the points of maximum curvature (critical points) as our features. Zhu and Chirlian's critical point detection algorithm [43] is used in our implementation.

The row and column coordinates of features points within each detected text rectangle are normalized. Text boxes within the two consecutive frames are then analyzed as follows. For each text box in the first frame, its normalized feature point locations are compared to the feature point locations of every text instance in the second frame. The feature point location error between each pair is computed, and the pair with the lowest error is chosen. If the lowest error is below a threshold, the two text instances are declared to belong to the same text event. Otherwise, the text event's lifetime is assumed to end with the current frame. Any text instances left unpaired in the second frame are assumed to be the start of a new text event.

5 Tracking and Binarization

5.1 Binarization Module

This section describes the binarization module of our system. The goal of the binarization module is to separate the pixels of a localized text region into categories of text and background. The output of the module is a binary image of the localized region suitable for input into an OCR system. For document images, simple thresholding is typically sufficient to convert a gray scale image into a binary image suitable for OCR. This technique assumes that the text and background colors are uniform (typically black on white) so that the image histograms are bimodal. In video, however, text often appears against complex, nonuniform backgrounds. The text color may also vary due to uneven illumination of scene text, antialiasing, or due to bleeding caused by video com-

pression. These problems are further compounded by the low resolution of video images, in which character strokes may be two pixel or less in width. Due to these factors, it was found that algorithms which rely on histogram bimodality [29, 17, 44, 45]) are generally unsuccessful for video images.

As with the detection and localization modules, the binarization module uses a number of different algorithms. After a preprocessing step, an initial binarization of the region is created. The binarization is then refined by examining other properties of the region, including stroke width, color, character size, character spacing, gray scale topography, and shapes. Following is a detailed description of each step.

- **Preprocessing**: Given a localized text box, the region is first pre-processed by stretching the gray scale contrast [29]. This allows binarization to succeed with low contrast text.

- **Binarization**: Logical level binarization algorithm proposed by Kamel and Zhao [46] has proven to be fairly successful for this step. The logical level algorithm was developed to extract character strokes from complex backgrounds in document images (for example, cash amounts from noisy check images). Upon experimentation, it was discovered that it also works well for extracting character strokes from gray scale images of video frames, provided that the text gray levels are darker than the the background. If the text is lighter than the background, the inverse of the frame must be taken before applying this algorithm.

  Unfortunately, determining whether the text is lighter than or darker than the background is a nontrivial problem. Other systems have handled this problem by making assumptions about the text color [47], by examining the pixels along the edge of the text box and assuming they are the background colors [37], and by examining both light and dark strokes and keeping those whose orientation and connected component size fit the characteristics typical of text strings. Methods described in [23, 17] perform poorly for text appearing against complex backgrounds. A variant of the method published in [15] is selected. The method performs logical level binarization on both the positive and the negative of the video frame, and the decision of determining the correct polarity is delayed until later. Connected components are then found in both binarized images.

  In general, logical level creates good binarizations of localized text. However, it can miss character strokes which have a very low contrast with the background, and it can include non-text pixels which exhibit some characteristics of character strokes. Color, size, spatial location, topography, and shape are used to refine the result of the binarization.

- **Color Clustering**: It is reasonable to assume that the characters of a text string are of uniform color. However, due to bleeding effects caused by low-resolution capture and compression, the actual pixel colors may vary significantly. Color clustering is used to allow for these effects. The text region is first quantized to reduce the color space, and then the complete-link algorithm [48] is used to cluster in the $l^*a^*b^*$ color space. The results of clustering may be used to refine the output of the logical level algorithm. Currently, components containing many different clusters are considered noise and removed [49]. Results of clustering could also be used to add or remove pixels from a given component.

- **Size filtering**: Connected components are filtered based on their size and aspect ratio. Very small components (with area less than about 12 pixels) are eliminated since they typically represent noise, or, if text, are too small to be recognized by an OCR system. Very large components and components with extreme aspect ratios are also removed.

- **Positive or negative image selection**: As mentioned earlier, it is not known *a priori* whether the text is lighter than or darker than the background in a gray scale video frame. Therefore the binarization and filtering steps were conducted on both the original video frame and its inverse. This approach has also been taken by [23, 17, 15]. These methods take the union of both image polarities (after applying heuristics to reduce noise) obtained as a result of binarization. Unfortunately, this results in too many false alarms. However, unlike the other systems using this approach, this method includes a localization module separate from the binarization module. It is reasonable to assume that all text in a localized bounding box is either darker than or lighter than the background. The binarized images are examined and a choice is made based on statistical information computed from the components of each image, such as the similarity in character height, character aspect ratio, vertical position, and horizontal spacing. The image with the more text-like features is then chosen.

- **Spatial location filtering**: Non-text components present in the binarization can be further reduced by introducing spatial constraints about character location. It is observed that most text in video is oriented horizontally, so components that are not located along horizontal lines with other components are eliminated. Components are clustered based on vertical position, and clusters with few components are then eliminated. This could be generalized to allow for non-horizontal text by, for example, using the Hough transform [50] or Messelodi and Modena's slope histogram method [15].

- **Topographical analysis**: While the logical level algorithm works well for most font sizes, it fails to capture the detail of very small fonts (with stroke widths near 1 pixel) effectively. For these fonts, we a topographical analysis algorithm [51] that operates on the gray scale image region is applied. Pixels that were chosen by the logical level algorithm and that correspond to a peak or ridge in the topography are used as the final binarization. This method serves to thin the binarization and produces cleaner output for small fonts.

- **Shape analysis**: It is observed that some common non-text objects satisfy the heuristics used by the detection and localization algorithms, such as uniformity of size, color, stroke width, spacing etc. Using the similarity of the shapes of connected component within a region serve as an indicator of text. It is noted that regions with nearly identical shapes are usually not text. However, the shapes of a given script are expected to be somewhat similar, so an area with very diverse shapes are also unlikely to have text. Currently these comparisons are based on simple statistics of the shapes, such as number and density of critical points along the contours. A simple contour-following algorithm is used to find the outline of each connected component, and this data is parameterized to polar coordinates. Zhu and Chirlian's algorithm [43] is used to locate the critical points of the contours.

5.1.1 Results

The results of binarization of localized text regions are shown in Figure 5.

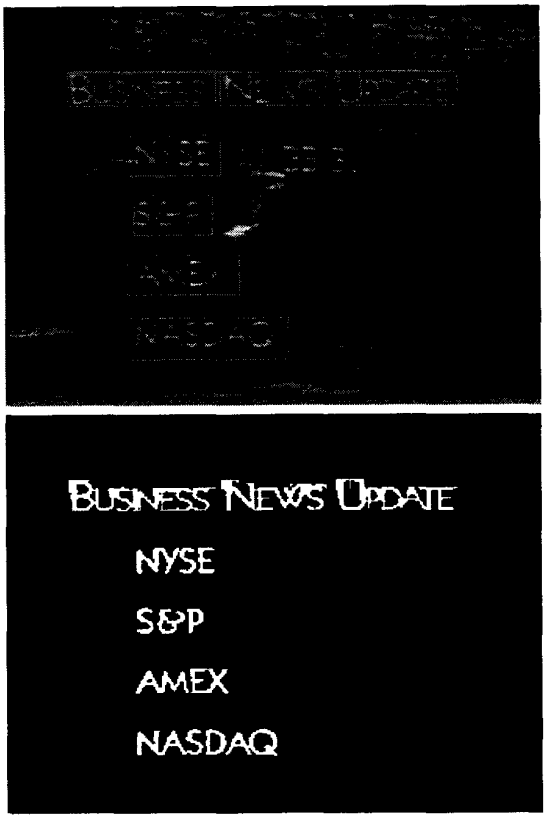


Figure 5: Sample Binarization Result

5.2 Text Tracking Module

Given the goal of automatically extracting text from video, it is not immediately apparent that a tracking component should be present in the system. However, it is needed as a verification of the localization algorithms. A detected text box that is changing in shape or that is moving (especially moving non-uniformly) may be eliminated as being due to noise unless its motion is verified by tracking. For scene text, zooming and rotation of the camera may alter the size and orientation of the text box. Erroneous discarding of localized regions can be avoided by using this knowledge. Finally. the intended application of this system may not always be to run in a completely automatic mode. In some scenarios, e.g., a ground truthing one, it may be desirable to have a human mark initial text boxes and have the system merely track it with time. The approach used for developing a tracker is based on methods described by Nakajima *et al* [52] and Pilu [53] with substantial modifications. Unlike the algorithm described by Li and Doermann [16] which uses only correlation within a search window (template matching) to track text, with the consequence that tracking is slower than localization, the adopted method also uses the motion vectors in the MPEG-1 bitstream.

The tracking algorithm assumes the availability of an initial horizontal bounding rectangular region to track. Multiple regions may also be specified. If the input is an MPEG bitstream, the method skips to the next available P-frame and extract the motion vectors for macroblocks that point to any

macroblock partially or completely within the initial box. Some macroblocks may be intra-coded in which case they cannot be used. Only those vectors that are 2 pixels or larger are considered. Next, spatial constraints are applied. This step eliminates stray motion vectors retaining only those that are similarly oriented. The motion vectors that point to the initial text box region are clustered and the largest cluster is assumed to belong to the new position of the moving text box. Further, flat vectors are deleted. This is achieved by computing edges using a kernel. Edgels[3] are statically thresholded and a count of remaining significant edgels is made. A macroblock is considered flat if it has less than 4 such significant edgels. In [52], the authors compute the absolute sum of the first 20 DCT coefficients and the last 60 coefficients and retain a block only if either of these is greater than an empirically determined threshold. This approach failed to provide satisfactory results.

Assuming that the text box undergoes rigid movements, the average motion vector for the region is computed. A correlation match is performed over a small neighborhood of the predicted text box region (the region in the P-frame). To ensure that the text is matched and not the background, the luminance gradients at each pixel (as computed by the Sobel operator) are compared against those in the initial text box. With subsequent P-frames an estimate of the block motion velocity is computed. The velocity is used to predict the position of the text box in the current P-frame. This region is used if it has a better correlation match than the region found using the motion vectors. If the correlation results in a high value then the neighborhood area is relaxed and the step is repeated. At the next I-frame, a predicted text box is obtained by averaging the motion from the last P-frame and the next P-frame, and then a correlation search is performed to find the exact position of the moving text box.

It is important to note that text may leave the frame or an initial text string may grow as more text belonging to the same string enters the frame. In the former case, the text box is resized appropriately. The latter situation is detected in [52] by looking for intra-coded macroblocks at the edge of the frame. If these are present, the authors hypothesize an object entering the frame. However, that paper deals with large, solid objects whereas we are tracking relatively small text regions with a possibly static background showing through. A different approach is adopted in this module. The number of edgels found along the edge of the frame is compared to the number found inside the text box. If these counts are comparable, text is assumed to be entering the frame, and the text box is grown to accommodate them. The tracker discards a block on complete failure of determining a suitable candidate region.

[3]Edgel is an edge pixel.

6 Performance Evaluation

Unlike the evaluation of automated methods for detection and localization of video events and objects contained within the imaged scene, the evaluation of text detection and localization methods presents interesting challenges. For example, when evaluating video shot change events [3], it is sufficient to detect at which frame a shot change (or other video transition event) occurred. In case of localization of vehicles, faces or other objects a tightly fitting bounding region is usually enough to perform a fair evaluation.

In case of automated text detection and localization methods, however, the degree of correctness is difficult to determine. This is because the the intent of text detection and localization is to recognize it for indexing, retrieval and other purposes. Also, humans tend to identify the text contained in the video as characters and words along a line, sentences, and paragraphs. Unfortunately, the algorithms that detect "text-like" regions within the video frame do not take this approach into consideration when applying the heuristics. They detect small regions that contain text and the size of the region (tightness of fit) is dependent on the size of the operating element used by the algorithm. For example, algorithms that operate on MPEG DC coefficients, will result in regions along 8x8 block boundaries, while those that use horizontal windows will have other boundaries. In order to obtain a commonality for evaluation, we evaluate the methods at the lowest common denomination, i.e. at the pixel level. Every pixel belonging to a text region, as detected by the algorithm as well as in the ground truth, is labeled as a such. All other pixels are labeled as non-text pixels. Evaluating the performance of the methods at the pixel level eliminates any issues related to the size operating elements of each method.

Unfortunately, the ground truth is usually marked by rectangular bounded regions which include the inter-character and sometimes inter-word non-text pixels. Also, non-text pixels surrounding the characters but within the ground-truth bounded region are considered as text pixels. Thus, if an algorithm is very accurate and detects the text but not the surrounding or inter-character pixels, it suffers a penalty for being very precise in the form of a low recall (higher missed detections). Conversely an algorithm which operates on large blocks actually detects the text correctly but has a looser region

boundary (due to operating block size) suffers the penalty in the form of low precision (higher false alarms). Thus, in a sense, the algorithms are being evaluated unfairly. It is necessary to allow a degree of subjectivity in evaluating these methods, which is to evaluate them based on their ability to detect each text event. We are in the process of developing such an evaluation method.

6.1 Test Data

Our test consists of 15 MPEG-1 video sequences totaling 10299 frames. The sequences were captured at 30 frames per second and encoded in MPEG-1 with a 352x240 frame size. The sequences are portions of news broadcasts and commercials from various countries. The test database is challenging due to the poor quality and low contrast of these broadcasts. Text appears in a variety of colors, sizes, fonts, and language scripts.

The ground truth was performed frame-by-frame by humans using the ViPER tool. Bounding text box size, position, and orientation angle were specified to pixel-level accuracy. All regions distinguishable as text by humans were included in the ground truth, including text too small or fuzzy to be actually read but nevertheless identifiable as characters. Closely spaced words lying along the same horizontal were considered to belong to the same text instance. Separate lines of text were kept separate. The ground truth contains a total of 133 temporally-unique caption text instances (36302167 ground-truth pixels) and 79 scene text instances (57532887 ground-truth pixels). There are 212 text events in total.

6.2 Evaluation criteria

The ground truth defines tightly-bound text boxes for each frame. A good detection/localization algorithm would (ideally) produce similarly tight boxes. To evaluate the accuracy and tightness of fit of an algorithm's output, the pixel areas of the text regions in the ground truth are matched with the detected text regions. The evaluation is a frame-by-frame, pixel-by-pixel comparison of algorithm output with the ground truth. In case of non-horizontal oriented scene text, All pixels within the oriented bounding region are considered. During evaluation, each pixel in the test database is placed into one of three categories:

- **Detection:** Pixels belonging to text regions in the ground truth and regions identified as text by the localization algorithm.
- **False Alarm:** Pixels identified by the detection algorithm but not belonging to text regions in the ground truth.
- **Missed Detection:** Pixels belonging to the text regions in the ground truth and not identified by the algorithm.

The performance of an algorithm is quantified by its recall and precision as defined in Equation 1. Note that this pixel-level evaluation is very strict. Most actual applications would not require such precise localization. However our pixel-level criteria provides an easily measurable basis by which the relative performances of algorithms may be compared.

$$Recall = \frac{detects}{detects + missed\ detects}$$
$$Precision = \frac{detects}{detects + false\ alarms} \quad (1)$$

6.3 Results and Discussion

This section presents the results of the performance evaluation of the selected text detection and localization algorithms. Most of the parameters for the methods were kept as described in the original publication. Only those parameters which were highly dependent on the dataset were tuned on a small subset of the test dataset (approx. 1000 frames).

Table 1 presents the caption text detection and localization performances, while Table 2 shows evaluation results for scene text, for the five algorithms on the entire test dataset. The table shows the raw numbers of total number of text pixels in the ground truth, the detected, false alarm, and missed detected pixels, along with computed recall and precision rates.

The results show that for caption text, overall Method D produces the highest precision rate of the individual algorithms, while the precisions of the other algorithms are comparably similar. Method E shows the highest recall. For scene text, Method D has the highest precision followed by Method E. Other methods have comparably similar results. Method E also has the highest recall for scene text. The test database contains some very challenging scene text instances. For applications in surveillance and navigation, detecting scene text would be important. In other applications, such as video indexing, detecting scene text may not be important or even useful. Therefore scene text and caption text were evaluated separately. All of the algorithms perform better for caption text than the scene text.

The recall and precision rates of the algorithms in our evaluation are relatively low and perhaps highlight the need for better text detection and localization algorithms. A solution to improving the precision and recall values of the methods is to apply algorithm fusion to combine the outputs of multiple existing algorithms to produce better outputs.

Algorithm	Text Pixels	Detects	FAs	MDs	Precision	Recall
Method A	36302167	14461593	62125359	21840574	39.84%	18.88%
Method B	36302167	14894707	45627542	21407460	41.03%	24.61%
Method C	36302167	22663915	156512965	13638252	62.43%	12.65%
Method D	36302167	26955906	119769022	9346261	74.25%	18.37%
Method E	36302167	17534049	35101417	18768118	48.30%	33.31%

Table 1: Overall Detection/Localization Performance : Caption Text

Algorithm	Text Pixels	Detects	FAs	MDs	Precision	Recall
Method A	57532887	10016556	66570396	47516331	17.41%	13.08%
Method B	57532887	7278171	53244078	50254716	12.65%	12.03%
Method C	57532887	27062384	152114496	30470503	47.04%	15.10%
Method D	57532887	22207563	124517365	35325324	38.60%	15.14%
Method E	57532887	13878758	38756708	43654129	24.12%	26.37%

Table 2: Overall Detection/Localization Performance : Scene Text

Method	Frames/sec.	Sec./frame
A	0.64	1.56
B	3.1	0.32
C	5.8	0.17
D	2.3	0.44
E	0.01	100

Table 3: Approximate algorithm running time.

Each algorithm uses an independent set of features and heuristics and so a fusing of outputs of multiple algorithms is likely to be beneficial.

6.4 Running time

Table 3 gives approximate running times for our implementation of each of the algorithms on an dual 270MHz. IP30 R12000 MIPS processor SGI Octane workstation. The times include overhead resulting from I/O and MPEG stream decompression. The times reported above can be improved since our implementations have not been fully optimized. Our implementation of the Method D operating on I, P and B frames was found to be the fastest (2.3 frames/sec.) as shown in the table. This method applied to I frames only clocked at 10.9 frames/sec. The increase in the processing time is because P and B frames need to be completely decompressed before Discrete Cosine Transform can be applied to them.

7 Conclusions

We have developed a system for extracting and segmenting an unconstrained variety of text from general purpose broadcast video. We have thoroughly evaluated the methods which form a part of our framework against a fairly large dataset. We have developed methods for fusing the results from different methods. More recently, we have extended our interest to localizing and extracting stylized text from video and determining the lifetimes of the video text events. We plan on developing methods for matching localized text regions based on shape and color properties. We also plan on developing methods for recognizing the localized text events to enable retrieval based on search strings.

References

[1] S. Antani, R. Kasturi, and R. Jain. Pattern Recognition Methods in Image and Video Databases: Past, Present and Future. In *Joint IAPR International Workshops SSPR and SPR*, number 1451 in Lecture Notes in Computer Science, pages 31–58, 1998.

[2] U. Gargi, R. Kasturi, and S. Antani. Performance characterization and comparison of video indexing algorithms. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 559–565, 1998.

[3] U. Gargi, R. Kasturi, and S. H. Strayer. Performance Characterization of Video-Shot-Change

Detection Methods. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):1–13, 2000.

[4] A. Hauptmann and M. Smith. Text, Speech, and Vision for Video Segmentation: The Informedia Project. In *AAAI Fall 1995 Symposium on Computational Models for Integrating Language and Vision*, 1995.

[5] M. A. Smith and T. Kanade. Video Skimming for Quick Browsing based on Audio and Image Characterization. Technical Report CMU-CS-95-186, Carnegie Mellon University, 1995.

[6] U. Gargi, S. Antani, and R. Kasturi. Indexing text events in digital video databases. In *Proc. International Conference on Pattern Recognition*, volume 1, pages 916–918, 1998.

[7] U. Gargi, D. Crandall, S. Antani, T. Gandhi, R. Keener, and R. Kasturi. A system for automatic text detection in video. In *International Conference on Document Analysis and Recognition*, pages 29–32, 1999.

[8] T. Gandhi, R. Kasturi, and S. Antani. Application of planar motion segmentation for scene text extraction. In *Proc. International Conference on Pattern Recognition*, volume 3, pages 445–449, 2000.

[9] V.Y. Mariano and R. Kasturi. Locating Uniform-Colored Text in Video Frames. In *Proc. International Conference on Pattern Recognition*, volume 4, pages 539–542, 2000.

[10] S. Antani, D. Crandall, and R. Kasturi. Robust extraction of text in video. In *Proc. International Conference on Pattern Recognition*, volume 3, pages 831–834, 2000.

[11] S. Antani, D. Crandall, A. Narasimamurthy, Y. Mariano, and R. Kasturi. Evaluation of Methods for Extraction of Text from Video. In *IAPR International Workshop on Document Analysis Systems*, pages 507–514, 2000.

[12] S. Antani, D. Crandall, V. Y. Mariano, A. Narasimhamurthy, and R. Kasturi. Reliable extraction of text in video. Technical Report CSE-00-022, Department of Computer Science and Engineering, The Pennsylvania State University, University Park, PA 16801, November 2000.

[13] B.-L. Yeo and B. Liu. Visual Content Highlighting Via Automatic Extraction of Embedded Captions on MPEG Compressed Video. In *SPIE/IS&T Symposium on Electronic Imaging Science and Technology: Digital Video Compression: Algorithms and Technologies*, volume 2668, pages 38–47, 1996.

[14] T. Sato, T Kanade, E.K. Hughes, and M.A. Smith. Video OCR for Digital News Archive. In *IEEE International Workshop on Content–Based Access of Image and Video Databases CAIVD'98*, pages 52–60, January 1998.

[15] S. Messelodi and C.M. Modena. Automatic Identification and Skew Estimation of Text Lines in Real Scene Images. *Pattern Recognition*, 32(5):791–810, May 1999.

[16] H. Li, D. Doermann, and O Kia. Automatic Text Detection and Tracking in Digital Video. *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[17] V. Wu, R. Manmatha, and E. Riseman. Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1224–1229, November 1999.

[18] N. Chaddha, R. Sharma, A. Agrawal, and A. Gupta. Text Segmentation in Mixed–Mode Images. In *28th Asilomar Conference on Signals, Systems and Computers*, pages 1356–1361, October 1995.

[19] Y. Zhong, H. Zhang, and A.K. Jain. Automatic Caption Localization in Compressed Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):385–392, 2000.

[20] Dimitrova, N. and Agnihotri, L. and Dorai, C. and Bolle, R. MPEG-7 *Videotext* description scheme for superimposed text in images and video. *Signal Processing: Image Communication*, 16:137–155, 2000.

[21] J.-C. Shim, C. Dorai, and R. Bolle. Automatic Text Extraction from Video for Content-Based Annotation and Retrieval. In *Proc. International Conference on Pattern Recognition*, pages 618–620, 1998.

[22] Y. Zhong, K. Karu, and A.K. Jain. Locating Text in Complex Color Images. *Pattern Recognition*, 28(10):1523–1536, October 1995.

[23] A.K. Jain and B. Yu. Automatic Text Location in Images and Video Frames. *Pattern Recognition*, 31(12):2055–2076, 1998.

[24] H.-K. Kim. Efficient Automatic Text Location method and Content–Based Indexing and

Structuring of Video Database. *Journal of Visual Communications and Image Representation*, 7(4):336–344, December 1996.

[25] H.-M. Suen and J.-F. Wang. Segmentation of Uniform-Coloured Text from Colour Graphics Background. *IEE Proceedings: Vision, Image and Signal Processing*, 144(6):317–322, December 1997.

[26] H. Hase, T. Shinokawa, M. Yoneda, M. Sakaiand, and H. Maruyama. Character String Extraction by Multi–stage Relaxation. In *International Conference on Document Analysis and Recognition*, volume 1, pages 298–302, 1997.

[27] J. Zhou, D. Lopresti, and Z. Lei. OCR for World Wide Web Images. In *Proceedings of SPIE Document Recognition IV*, volume 3027, pages 58–66, 1997.

[28] R. Lienhart and F. Stuber. Automatic Text Recognition in Digital Videos. In *Proceedings of SPIE*, volume 2666, pages 180–188, 1996.

[29] F. LeBourgeois. Robust Multifont OCR System from Gray Level Images. In *International Conference on Document Analysis and Recognition*, volume 1, pages 1–5, 1997.

[30] M.v.d. Schaar-Mitrea and P.H.N. de With. Compression of Mixed Video and Graphics Images for TV Systems. In *SPIE Visual Communications and Image Processing*, pages 213–221, 1998.

[31] B.T. Chun, Y. Bae, and T.-Y. Kim. Text extraction in videos using topographical features of characters. In *IEEE International Fuzzy Systems Conference*, volume 2, pages 1126 –1130, 1999.

[32] O. Hori. A video text extraction method for character recognition. In *International Conference on Document Analysis and Recognition*, pages 25–28, 1999.

[33] H. Kasuga, M. Okamoto, and H. Yamamoto. Extraction of characters from color documents. In *Proceedings of IS&T/SPIE Conference on Document Recognition and Retrieval VII*, volume 3967, pages 278–285, 2000.

[34] Y.-K. Lim, S.-H. Choi, and S.-W. Lee. Text extraction in mpeg compressed video for content-based indexing. In *Proc. International Conference on Pattern Recognition*, volume 4, pages 409–412, 2000.

[35] M. Sawaki, H. Hurase, and N. Hagita. Automatic acquisition of context-based images templates for degraded character recognition in scene images. In *Proc. International Conference on Pattern Recognition*, volume 4, pages 15–18, 2000.

[36] K. Sobottka, H. Bunke, and H. Kronenberg. Identification of text on colored book and journal covers. In *International Conference on Document Analysis and Recognition*, pages 57–62, 1999.

[37] J. Ohya, A. Shio, and S. Akamatsu. Recognizing Characters in Scene Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:214–224, 1994.

[38] P. Comelli, P. Ferragina, M. N. Granieri, and F. Stabile. Optical Recognition of Motor Vehicle License Plates. *IEEE Transactions on Vehicular Technology*, 44(4):790–799, November 1995.

[39] Y. Cui and Q. Huang. Character Extraction of License Plates from Video. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 502–507, 1997.

[40] L.L. Winger, M.E. Jernigan, and J.A. Robinson. Character Segmentation and Thresholding in Low-Contrast Scene Images. In *Proceedings of SPIE*, volume 2660, pages 286–296, 1996.

[41] T. Gandhi. *Image Sequence Analysis for Object Detection and Segmentation.* PhD thesis, The Pennsylvania State University, University Park, PA 16802, 2000.

[42] B.G. Sherlock and D.M. Munro. Algorithm 749: Fast Discrete Cosine Transform. *ACM Transactions on Mathematical Software*, 21(4):372–378, 1995.

[43] P. Zhu and P.M. Chirlian. On Critical-Point Detection of Digital Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):737–748, August 1995.

[44] J.N. Kapur, P.K. Sahoo, and A.K.C. Wong. A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram. *Computer Vision, Graphics, and Image Processing*, 29(3):273–285, March 1985.

[45] A.K.C. Wong and P.K. Shaoo. A Gray-Level Threshold Selection Method Based on Maximum Entropy Principle. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(4):866–871, July 1989.

[46] M. Kamel and A. Zhao. Extraction of Binary Character/Graphics Images from Grayscale Document Images. *Computer Vision, Graphics, and Image Processing*, 55(3):203–217, May 1993.

[47] J.-H. Jang and K.-S. Hong. Binarization of noisy gray-scale character images by thin line modeling. *Pattern Recognition*, 32(5):743–752, 1999.

[48] A.K. Jain. *Algorithms for clustering data.* Prentice Hall, Englewood Cliffs, NJ, 1988.

[49] S.D. Yanowitz and A.M. Bruckstein. A New Method for Image Segmentation. *Computer Vision, Graphics, and Image Processing*, 46(1):82–95, April 1989.

[50] L.A. Fletcher and R. Kasturi. Automated Text String Separation from Mixed Text/Graphics Images. Technical Report TR-86-001, Department of Electrical and Computer Engineering, Penn State University, University Park, PA, 1986.

[51] S.-W. Lee and Y.J. Kim. Direct Extraction of Topographical Features for Gray Scale Character Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):724–728, July 1995.

[52] Y. Nakajima, A. Yoneyama, H. Yanagihara, and M. Sugano. Moving Object Detection from MPEG Coded Data. In *Proceedings of SPIE*, volume 3309, pages 988–996, 1998.

[53] M. Pilu. On Using Raw MPEG Motion Vectors to Determine Global Camera Motion. In *Proceedings of SPIE*, volume 3309, pages 448–459, 1998.