International Journal on Document Analysis and Recognition manuscript No.
(will be inserted by the editor)

Extraction of special effects caption text events from digital video

David Crandall*, Sameer Antani, Rangachar Kasturi**

Department of Computer Science and Engineering, The Pennsylvania State University, 202 Pond Laboratory, University Park, PA 16802

The date of receipt and acceptance will be inserted by the editor

Abstract. The popularity of digital video is increasing rapidly. To help users navigate libraries of video, algorithms that automatically index video based on content are needed. One approach is to extract text appearing in video, which often reflects a scene's semantic content. This is a difficult problem due to the unconstrained nature of general-purpose video. Text can have arbitrary color, size, and orientation. Backgrounds may be complex and changing.

Most work so far has made restrictive assumptions about the nature of text occurring in video. Such work is therefore not directly applicable to unconstrained, general-purpose video. Also, most work so far has focused only on detecting the spatial extent of text in individual video frames. But text occurring in video usually persists for several seconds. This constitutes a text event that should be entered only once in the video index. Therefore it is also necessary to determine the temporal extent of text events. This is a non-trivial problem because text may move, rotate, grow, shrink, or otherwise change over time. Such text effects are common in television programs and commercials but so far have received little attention in the literature.

This paper discusses detecting, binarizing, and tracking caption text in general-purpose MPEG-1 video. Solutions are proposed for each of these problems and compared with existing work found in the literature.

Key words: Text detection – Text tracking – Text binarization – Video indexing

1 Introduction

1.1 Motivation

As the popularity of digital video increases and quantities of video rise, automatic content-based video indexing will become an increasingly important problem. Interest in this area has fueled research into algorithms that automatically extract semantic features of video. Useful features include genre (sitcom, movie, sports event, etc.), filming location (indoor or outdoor, time of day, weather conditions, etc.), identity of important objects, identity of people (politicians, movie stars, sitcom characters, etc.), and human activity and interaction (running, laughing, talking, arguing, etc.).

Text appearing in a video sequence can also provide useful semantic information. Words have well-defined, unambiguous meanings. Text extracted from a video sequence provides natural, meaningful keywords that reflect the video's content.

Two types of text are found in video. *Caption text* is artificially superimposed on the video at the time of editing. Caption text usually underscores or summarizes the video's content. This makes caption text particularly useful for building keyword indexes. Figure 1 presents some examples of caption text. *Scene text* naturally occurs in the field of view of the camera during video capture. Scene text occurring on signs, banners, etc. may also give keywords that describe the content of a video sequence.

1.2 Video text extraction vs. document OCR

At first, extraction of text from video may seem to be a simple extension of existing optical character recognition (OCR) problems. In particular, OCR for document images has been studied extensively [32]. However, extraction of text from video presents unique challenges over OCR of document images. Such challenges include:

– **Lower resolution:** Video frames are typically captured at resolutions of 320×240 or 640×480 pixels,

* D. Crandall is now with Eastman Kodak Company, 1700 Dewey Avenue, Rochester, NY 14650-1816; e-mail: david.crandall@kodak.com
** S. Antani is now with the National Library of Medicine, 8600 Rockville Pike, Bethesda, MD 20894; e-mail: antani@nlm.nih.gov
Correspondence to: David Crandall

Fig. 1. Examples of caption text addressed in this work.

while document images are typically digitized at resolutions of 300 dots per inch or greater.
- **Unknown text color:** Text can have arbitrary and non-uniform color.
- **Unknown text size, position, orientation, layout:** Captions lack the structure usually associated with documents.
- **Unconstrained background:** The background can have colors similar to the text color. The background may include streaks that appear very similar to character strokes.
- **Color bleeding:** Lossy video compression may cause colors to run together.
- **Low contrast:** Low bit-rate video compression can cause loss of contrast between character strokes and the background.

The temporal nature of video also introduces a new dimension into the text extraction problem. Text in video usually persists for at least several seconds. Some text events remain unchanged during their lifetimes. Others, like movie credits, move in a simple, linear fashion. Still others, like scene text and stylized caption text, move and change in complex ways. Text can grow or shrink, or character spacing can increase or decrease. Text color can change over time. Text can rotate and change orientation. Text can morph from one font to another. Text strings can break apart or join together. Special effects or a moving camera can cause changing perspective.

It is possible to simplify the text extraction problem by making *a priori* assumptions about the type of video, or to extract only certain types of text. However, in a general-purpose video indexing application, it is important to be able to extract as much text as possible. Therefore text extraction systems must be applicable to general-purpose video data and should be able to handle as many types of text as possible.

1.3 Problem statement and scope

This paper discusses the extraction of unconstrained caption text from general-purpose video. In particular, it addresses extraction of types of text that have largely been ignored by the work in the literature to date. These types of caption text include moving text, rotating text, growing text, shrinking text, text of arbitrary orientation, and text of arbitrary size. In addition, our approach is capable of determining the temporal extent of each text event without the use of OCR. No other work in the literature to date has approached this problem. The focus of this work is on extraction of caption text, although much of the work could be applied to extracting scene text as well.

Text extraction from video can be divided into the following subproblems:

- **Text detection** involves locating regions in a video frame that contain text.
- **Text localization** groups text regions identified by the detection stage into text instances. The output of a good localization algorithm is a set of tight bounding boxes around all text instances.
- **Text tracking** involves following a text event as it moves or changes over time. Together, the detection, localization, and tracking modules determine the temporal and spatial locations and extents of text events.
- **Text binarization** involves separating text strokes from the background in a localized text region.[1] The output of binarization is a binary image, with pixels of text strokes marked as one binary level and background pixels marked as the other.
- **Text recognition** performs OCR on the binarized text image. The recognition problem is not discussed in this paper.

This paper discusses the text detection, tracking, and binarization problems, proposes new algorithmic solutions, and presents quantitative and qualitative results. Section 2 reviews the state-of-the-art in the text extraction problem. Section 3 presents a novel approach to these problems, with an emphasis on handling special-effects types of text that have been largely ignored in the literature to date. In Section 4 we present results of these algorithms on sample datasets. We also present results of a quantitative performance evaluation and comparison with several detection algorithms from the literature.

[1] In previous publications (e.g. [2,11]) we used the term *segmentation* to refer to the binarization problem. We used it in the context of segmenting individual text pixels from background pixels. Unfortunately this term is used inconsistently in the text extraction literature. Some authors (e.g. [5]) use this term to refer to the text *region* segmentation problem. Others (e.g. [19]) use it to refer to the *character* segmentation problem, in which individual characters are located. To avoid confusion, we will avoid the term *segmentation* in this paper.

Finally, conclusions are drawn and areas for future work are identified in Section 5.

2 Review of Prior Work

In this section we review past work relevant to the problem of extracting text from video. A literature survey in this area finds a significant amount of work in extracting text from images. Relatively fewer papers are found on extracting text from video frames, and only a handful consider the temporal nature of video. We include recent papers of all three types in our survey. In each case we summarize the approach and highlight any noteworthy contributions, assumptions, or limitations.

- Agnihotri and Dimitrova [1] detect and binarize horizontal white, yellow, and black caption text in video frames. After pre-processing, edge pixels are found using an edge detector with a fixed threshold. Frame regions with very high edge density are considered too noisy for text extraction and are discarded. Connected component analysis is performed on the edge pixels of remaining regions. Edge components are merged based on spatial heuristics to localize text regions. Binarization is performed by thresholding at the average pixel value of each localized text region.
- Antani *et al* [3] performs text detection in video using a decision fusion approach. Because different algorithms use different assumptions about the nature of text in video, an intelligent combination of the outputs of multiple detection algorithms was found to give better results than any individual algorithm.
- Chaddha [5] analyzes JPEG images in the frequency domain to detect blocks with text-like texture. For each block, the sum of a subset of DCT coefficients is computed and thresholded against a fixed constant. In earlier work, we modified and enhanced Chaddha's algorithm for use in video [6].
- Garcia and Apostolidis [9] locate and binarize horizontal text in color images. Edge pixel magnitudes and directions are determined in each plane. Text regions are selected by identifying areas with high edge density and high variance of edge orientation. Morphological operations remove singletons and non-horizontal regions. Localization is performed by connected component analysis. Candidate text regions are joined together and split apart using heuristics. Binarization is performed by clustering in HSV color space. It is assumed that after clustering, all text pixels will fall in a single cluster. That cluster is identified by choosing the cluster with the most periodic vertical profile.
- Gargi *et al* [10] describe an algorithm for locating horizontal text strings in video frames. Their method looks for horizontal streaks of similar color that may correspond to character strokes. Size and aspect ratio heuristics are applied to reduce false alarms.
- Jain and Yu [16] presents a method to locate text in pseudo-color images, color images, and video frames. Quantization and color clustering are performed. It is assumed that the largest color cluster is the background region and the other clusters represent text. Connected components of the foreground colors are grouped together into text lines using alignment and spacing heuristics. The example video images shown in the paper are relatively simple, yet the algorithm inexplicably misses several prominent text instances. The assumption that all background pixels are clustered together is often not true for unconstrained video.
- Jeong *et al* [17] apply neural networks to find text captions in Korean news broadcasts. Detection is performed on a hierarchy of sub-sampled images to allow for text of different sizes. Character spacing, text line spacing, horizontal alignment, and aspect ratio heuristics are applied in post-processing.
- LeBourgeois [19] localizes and binarizes text in grayscale images. After pre-processing, image gradients are smeared horizontally. Localization is performed by connected component analysis. Text lines are further segmented into individual characters by locating valleys in the horizontal and vertical projection profiles. It is assumed that the dominant portion of the image histogram is the background. Binarization is by global thresholding. A post-processing stage splits inadvertently connected characters. The text is assumed to have a fixed font size.
- Lee *et al* [21,20] use a combined detection and binarization approach. After a quantization step, the image is searched for horizontal and vertical streaks with uniform gray level. Connected streaks are merged to form character candidate regions. Each potential character is thresholded using the gray level of its boundary. Post-processing removes components with suspicious aspect ratios, contrast, and fill ratios.
- Li *et al* [22,23] address detection and tracking of moving text in video. A window of 16×16 pixels is passed over the image. The wavelet transform of the pixels under the window is taken, and its moments are passed as features into a neural network classifier. Blocks are grouped together to form horizontal text boxes. A multi-scale approach is used to detect text of different sizes. Tracking is accomplished using a pixel-wise matching scheme that minimizes least-squared-error. It is assumed that text moves with constant velocity and does not change in size or shape over time. A recent extension to this work [24] adds a post-processing step to correct tracking results when text grows or shrinks slightly from frame to frame. An edge detector is applied to find character strokes. A tight bounding rectangle is found around these characters and is used as the final tracking result. The authors report that this extension failed for text moving over complex backgrounds. To overcome this problem, the least-squared-error value computed during the neighborhood search is monitored. If a spike in the error occurs, it is assumed that the text is moving over a complex background, and the post-processing step is disabled.
- Lienhart [27,28,26] locates and recognizes text in video. A split-and-merge image segmentation technique is

applied to locate text in individual frames. Local color variance is used as the homogeneity criteria for segmentation. Segmented regions are chosen based on heuristics. Example detection results shown in the paper include many false alarms. This is mitigated by a custom OCR module that discards candidate regions that cannot be recognized. Temporal analysis is also used to remove false positives. For each candidate text region in a frame, the next frame is searched for a text candidate with identical position, size, color, and shape. If such an area is not found, the region is discarded as non-text. This approach assumes that text remains stationary.

- Mariano [29] performs simultaneous detection and binarization of horizontal text regions. Color clustering in L*a*b* space [37] is performed on individual scan lines. Clusters occurring in neighboring scan lines are analyzed to find regularly-spaced streaks corresponding to text strokes. It is assumed that text is precisely horizontal. The intense use of color clustering creates a high computational demand. An implementation of the algorithm obtained from the author required about 50 minutes to process a 1-second video clip on an SGI Octane workstation.
- Messelodi and Modena [30] present a system for binarizing text from book cover images. They use a simple global thresholding scheme at the tails of each side of the histogram. Their method considers binarization of oriented text.
- Ohya *et al* [35] use a combined detection/binarization stage and OCR to extract characters from scene images. Text is assumed to be either black or white. Regions of the image with bimodal histograms are assumed to be text regions. Local thresholding is performed on these regions. Shape and size heuristics are applied to reject non-text strokes. An OCR stage is used to validate detection by removing regions with low recognition confidence.
- Qi *et al* [38] extract captions from news video sequences. Horizontal and vertical edge maps of a video frame are computed using a Sobel operator. Alignment of edges is analyzed to find horizontally-oriented text instances. Sample results in the paper are quite noisy, suggesting that the algorithm is unsuitable for general-purpose video.
- Sato *et al* [39] performs simultaneous localization and binarization of caption text in video frames. Filters are applied to detect vertical, horizontal, and diagonal line elements. Edge pixels are grouped using aspect ratio and other heuristics. Text is assumed to be white, appear over a dark background, and have horizontal orientation. Final binarization is performed by thresholding at a fixed, pre-set threshold.
- Schaar-Mitrea *et al* [40] detect overlaid text and graphics in video frames. Blocks of size 4×4 pixels are examined. The number of block pixels having similar gray levels is counted. If this count is greater than a threshold, and if the dynamic range of the block is less than a threshold or greater than another threshold, the block is classified as text.
- Shim *et al* [41,42] propose a method to detect and binarize caption text in video. Regions with homogeneous intensity are identified, positive and negative images are formed by double thresholding, and heuristics are applied to eliminate non-text regions. Text is assumed to be either black or white. Binarization is performed by thresholding each character stroke individually using an adaptive threshold computed from a local histogram. A simple inter-frame analysis technique reduces false alarms. Candidate text regions in groups of five adjacent frames are considered. A candidate text region is discarded if regions of similar position, intensity, and shape do not appear in the other frames. Note that this approach incorrectly discards moving text regions.
- Shin *et al.* [43] perform detection and binarization in one step by classifying individual pixels using a support vector machine (SVM). Local grayscale pixel values are used as features. A hierarchical strategy is employed to handle text of various sizes.
- Winger *et al* [46] performs binarization of low-contrast scene text using a modification of Niblack's Multiple and Variable Thresholding scheme [34]. Correspondence with the author suggests that manual tuning of algorithm parameters is needed for each image.
- Wong [47] locate and binarize text in video frames. Text is detected in the luminance plane. A 1×21 pixel window is passed over the image, and the difference between the maximum and minimum gradients within the window is determined. Gradient zero-crossings are found and the mean and variance between zero-crossings are computed. Pixels under the window are marked as text if the gradient difference is high, the variance is low, and the mean is within a reasonable range. These text lines are merged together into localized text regions. For binarization, color clustering is performed until only two clusters remain. The two clusters are assumed to correspond with text pixels and background pixels. Our experimentation with color clustering has indicated that this assumption rarely holds due to complex backgrounds that contain colors similar to the text color.
- Wu *et al* [48] describe a scheme for finding and binarizing text in images. Texture segmentation is used to locate potential text regions. Edge detection is then applied to find candidate text strokes, which are merged to form text regions. Binarization is performed by smoothing the grayscale image and then thresholding at the valleys on either end of the grayscale histogram. This allows for both light text and dark text to be binarized. The algorithm does not determine whether the text is light or dark but instead generates two outputs, one for each case.

In summary, there has been a relatively large amount of work on text extraction from still images and individual video frames. Only a few papers have considered the temporal nature of video in the text extraction problem. Each text detection and binarization algorithm in the literature tends to make a unique set of assumptions about caption text. Common assumptions include restrictions

on text color, stroke size, background color, spatial text location, text orientation, and stroke color uniformity. Most text tracking algorithms so far assume that text remains rigid and exhibits simple, linear motion. Few papers in this field present objective, quantitative performance evaluations of proposed algorithms. Those papers that do report performance evaluations use different evaluation criteria and different datasets, making quantitative algorithm comparisons difficult.

3 Approach

In this section, we describe a novel approach for extracting text events from digital video. The approach consists of three steps: text detection and localization in individual frames, text binarization, and text tracking from frame to frame. We present two tracking algorithms: a very efficient tracker that assumes that text remains rigid with time, and a tracker that can follow text that shrinks or grows, rotates, or otherwise changes.

Our approach makes few assumptions about the nature of text in video. In particular, unlike all other text from video extraction work in the literature, our approach handles text of arbitrary orientation. No assumptions on text color or size are made. The algorithm is robust to complex, changing backgrounds. No explicit assumptions on language script are made, although scripts without separated characters are more challenging for the binarization algorithm (see Section 3.2). Although this approach was designed for caption text, it works well for high-contrast scene text also.

The remainder of this section describes the three steps of our approach in detail.

3.1 Text detection and localization

Text detection Detection is accomplished by analyzing local texture features and finding blocks of the image that have texture consistent with text. Texture analysis is performed in the block-wise Discrete Cosine Transform (DCT) domain. Texture is an attractive feature for this use because very small font sizes can be detected, even if the text is not easily readable. The DCT domain was used for text detection before in [5], but the application was constrained document images.

The basic text detection algorithm is accomplished as follows. The 8x8 block-wise DCT is performed on a video frame. For each block, an subset of the DCT coefficients is extracted. The sum of the absolute values of these coefficients is computed and regarded as a measure of the "text energy" of that block.

The subset of DCT coefficients that best correspond to the properties of text was determined empirically. Finding the optimal coefficients is non-trivial. An exhaustive search would require trying all combinations of between 1 and 64 coefficients, or $\sum_{i=1}^{64} \binom{64}{i} \approx 1.8 \times 10^{19}$ possibilities. An alternative is as follows. The average absolute value of each coefficient for both text and non-text

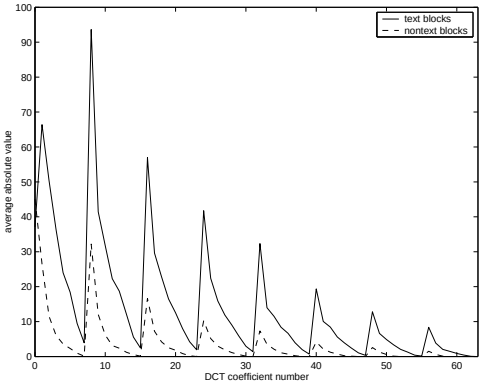


Fig. 2. Average DCT coefficient energy for text and non-text DCT blocks.

blocks is determined. Coefficients are sorted by the difference between text and non-text sums. Coefficients are then added one-by-one in sorted order until the optimal choice of coefficients is found. This procedure requires trying at most 64 combinations of coefficients.

We performed the optimization in this manner using 9,122,279 blocks (539,941 text blocks, 8,582,338 non-text blocks) from 9,329 frames of video from Dataset A described in Section 4.1. Figure 2 compares the average absolute value of each coefficient for text and non-text blocks. Using the procedure described above, the optimal coefficients were determined to be 1, 2, 3, 4, 5, 8, 9, 10, 11, 12, 16, 17, 18, 19, 24, 25, 26, 32, and 40, in row-major order.

The above coefficient choice optimization was performed for horizontal text. Because the 2-D DCT is separable, transposing the matrix of pixel values of a block corresponds with the transpose of the DCT coefficient matrix. It follows that vertical text can be detected by first taking the transpose of a block's DCT coefficient matrix, and then using the same coefficients listed above. We have observed that diagonal text has a combination of horizontal and vertical DCT text energy.

These observations motivate the following method for detecting text blocks. For each luminance DCT block, the horizontal text texture energy TTE_h is computed by summing the coefficients listed above. Similarly, the vertical text texture energy TTE_v is computed by transposing the DCT coefficient matrix, and then summing the above coefficients. Horizontal and vertical groups of three blocks are examined at once to encourage regions with high TTE_h to grow horizontally and blocks with high That is, the block at row i and column j in an image is marked as text if

$$\frac{TTE_h(i,j-1) + TTE_h(i,j) + TTE_h(i,j+1)}{3} + \frac{TTE_v(i-1,j) + TTE_v(i,j) + TTE_v(i+1,j)}{3} > T$$

where T is a threshold.

Next we consider how to choose the threshold T. A fixed threshold is undesirable, as we have found that the

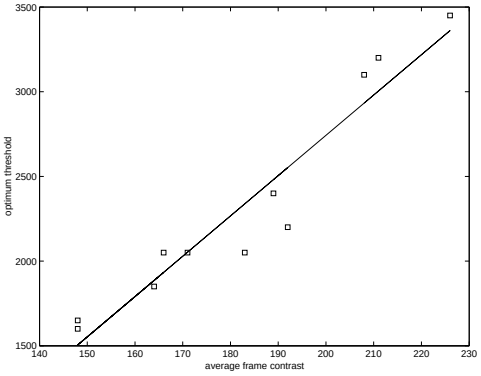


Fig. 3. Optimal threshold versus average video frame contrast.

optimal threshold varies widely from one video sequence to the next.

We have observed that the optimal threshold is fairly uniform across all frames of the same video sequence. Also, different video sequences of the same general type have similar optimal thresholds. This suggests that the optimal threshold depends on general characteristics of the video that could be computed or known *a priori*. For example, perhaps one threshold is best suited for news broadcasts, while another is better for commercials. The genre of video may be known *a priori*, or an algorithm could be used to automatically determine the genre (e.g. [14]). We also observed that low-level image features could also be used to predict optimal threshold. Specifically, we hypothesized that video contrast could be used.

This hypothesis was tested as follows. The optimal threshold for each video sequence in our dataset was determined by exhaustively trying all possible thresholds within a reasonable range (again according to the experimental protocol and evaluation criteria discussed in Section 4.1). The average contrast per frame for each video sequence was also computed.

Figure 3 plots the optimal threshold versus the average frame contrast. The figure indicates that there is a strong linear correlation between contrast and ideal threshold. The best-fit line that minimizes least-square-error was found to be about $T(c) = 23.8c - 2018.5$, where c is average contrast. This result suggests that it is possible to predict a good threshold based only on the general characteristics of a video sequence.

Note that this analysis was carried out on a relatively small dataset of 11 sequences and 11000 frames. More experimentation would be necessary to determine that this simple linear relationship holds for a larger dataset. Also, the optimal threshold may be better correlated with other features. For the work in this paper, however, we use only contrast to predict the threshold. The threshold for each sequence is computed using the $T(c)$ expression given above.

The use of an 8x8 block size implicitly limits the size of text that can be detected. This problem can be solved by analyzing subsampled versions of the frame. For example, text with strokes up to 16 pixels wide can be detected in a frame subsampled to half the original size.

Subsampling is incorporated into the algorithm as follows. The text block detection algorithm is applied to the image. Then, the image is subsampled to half its dimensions, and the 8×8 block classification algorithm is applied again. A block at this level corresponds to four blocks in the original image. For each block classified as text in the subsampled image, the corresponding four blocks in the original image are marked as text. Subsampling continues iteratively until a lower bound on the frame dimensions is reached. We have observed that proceeding to a resolution of 160×120 ensures that all text of reasonable size is found. This corresponds to two levels of subsampling for an original frame size of 320×240 and three levels for a frame size of 640×480. Note that subsampling can be performed efficiently in the DCT domain (see [8]).

Text localization Once individual blocks of a frame have been classified, we wish to group the blocks into text instances. This is done by finding minimum bounding rectangles around each text instance. In the case of non-horizontal text, the bounding rectangle should be oriented at the appropriate angle.

We use an iterative greedy algorithm for localizing text instances. First, connected component analysis is performed on the blocks detected as text. Orthogonal bounding rectangles are computed for each component. Then, the bounding rectangles are iteratively refined. Each iteration of the greedy algorithm attempts to increase the criterion

$$G = P_t \times (1 - P_{nt})$$

where P_t is the percentage of the detected text pixels that lie underneath the rectangle, and P_{nt} is the fraction of the rectangle's area covering non-text pixels. During each iteration, each rectangle is visited and one of the following actions is taken:

- Rectangle is left unchanged
- Increment or decrement rectangle height or width by one block
- Shift rectangle one block horizontally or vertically
- Rotate by 15 degrees clockwise or counter-clockwise

At the completion of each iteration, overlapping rectangles are merged together if doing so does not lower the value of G.

The iteration continues until convergence. Heuristics can then be applied to discard non-text regions based on rectangle dimensions. We discard rectangles whose length or width is less than 8 pixels.

3.2 Text binarization

Binarization is the process of separating character strokes from the background. That is, given a localized region

of a color video frame thought to contain text, binarization produces a binary image of the text. A text detection algorithm classifies video frame regions as text or background; a binarization algorithm classifies individual pixels as text or background.

Binarization is necessary to bridge the gap between localization and recognition. The eventual goal of most text extraction systems is to perform text recognition. Optical character recognition (OCR) in the context of document images has been extensively studied [32]. We would like to leverage existing OCR algorithms into the text-in-video extraction problem. However, most recognition algorithms expect images resembling documents, with black text strokes and white backgrounds. It is the responsibility of a binarization algorithm to convert the color text regions occurring in video frames to the simple binary images required by OCR.

A binarization step is usually employed when capturing document image data from a grayscale scanner. Thresholding based on histogram analysis is often used for this purpose, but histogram-based thresholding often fails for binarizing text from video frames. The variability of pixel colors within a single text stroke and unconstrained nature of the background can cause the same pixel values to be present in both foreground and background. In some cases, any histogram-based thresholding scheme will fail (see [6] for an example).

We have developed an algorithm that makes few assumptions about the text to be binarized, but produces clean binarizations even when text appears against complex backgrounds. Figure 4 illustrates the steps of the algorithm on a sample video frame. The following sections describe the binarization algorithm in detail.

Preprocessing Preprocessing is performed on each localized text box in a frame. First, if the localized text region is slanted at a non-horizontal orientation, a rotation transformation is applied to make it horizontal. A linear interpolation step is used to double the resolution.

Next, grayscale histogram stretching [12] is performed on each text region. This improves the contrast between text and background in low-contrast areas of the image.

If the text extraction system is run in rigid text mode, temporal averaging is performed during preprocessing. Temporal averaging is known to reduce random noise introduced during video digitization and compression. Temporal averaging of text regions also tends to smooth the background, increasing contrast and eliminating text-like strokes in the background (see [6] for an example). Caption text often remains stationary while the background behind it changes or moves. Or the text may move, causing the background behind the text to change. In either case, temporal averaging accomplishes background smoothing. Our rigid text tracking module (discussed in Section 3.3) is employed to resolve the text motion in each frame.

Temporal integration is accomplished as follows. For a localized text box in a given frame, the tracking module follows the text over the next 10 frames. The regions are averaged together and used for binarizing the first frame. Note that this assumes that the text remains rigid as it moves. If it does not remain rigid, temporal averaging blurs text strokes. To prevent this, the confidence of the text tracker is monitored. If the confidence falls below a threshold, it is likely that the text is not rigid, and temporal averaging is disabled for the remainder of the text instance. Figure 4(b) shows an example of the result of preprocessing.

More sophisticated resolution enhancement schemes could be used. For example, there has been some work in resolution enhancement using multi-frame integration of moving objects [44].

Logical level thresholding Some document analysis work has considered the problem of binarizing text occurring in very noisy document images. This problem shares similarities with our problem of extracting text occurring against complex backgrounds. Kamel and Zhao [18] evaluate seven binarization techniques on noisy bank check images. Their novel method, logical level thresholding, was shown to perform the best. The algorithm's strength over other binarization techniques is that it enforces restrictions on uniformity of stroke grayscale level, uniformity of stroke width, and bounds on stroke width. This leads to less noise in the binarized output.

We have adapted logical level thresholding to binarization of text in video frames. After the preprocessing steps described above, the text region is converted into the perceptually-uniform L*a*b* color space [37]. Logical level thresholding is then applied on the luminance plane. Logical level thresholding requires two parameters, the maximum stroke width W and the contrast threshold T. We observed that the algorithm's performance is relatively insensitive to the choice of parameters. We use $T = 5$, which is a good compromise between allowing binarization of low-contrast text and preventing noise. The choice of stroke width W is proportional to the size of the input video frame. For a frame resolution of $320n \times 240n$, $W = 10n$ works well. A stroke width of this size corresponds with a character that nearly fills the video frame.

Logical level thresholding requires that the text stroke color be darker than the background. In our application this is not an acceptable assumption because we wish to extract text of any color. We tried modifying the thresholding step to also permit light strokes, but this relaxes the restriction of stroke color consistency and created many false alarms. Instead, we assume that all text strokes within a localized text region are either lighter than or darker than the background. Logical level thresholding is then applied to both the original region and its inverse to produce two independent binarized outputs. The choice of correct polarity is delayed until a later step. Figure 4(c) shows an example of the result of logical level thresholding.

Character candidate filtering Connected component analysis is performed on both output images from the last step. These connected components are either characters

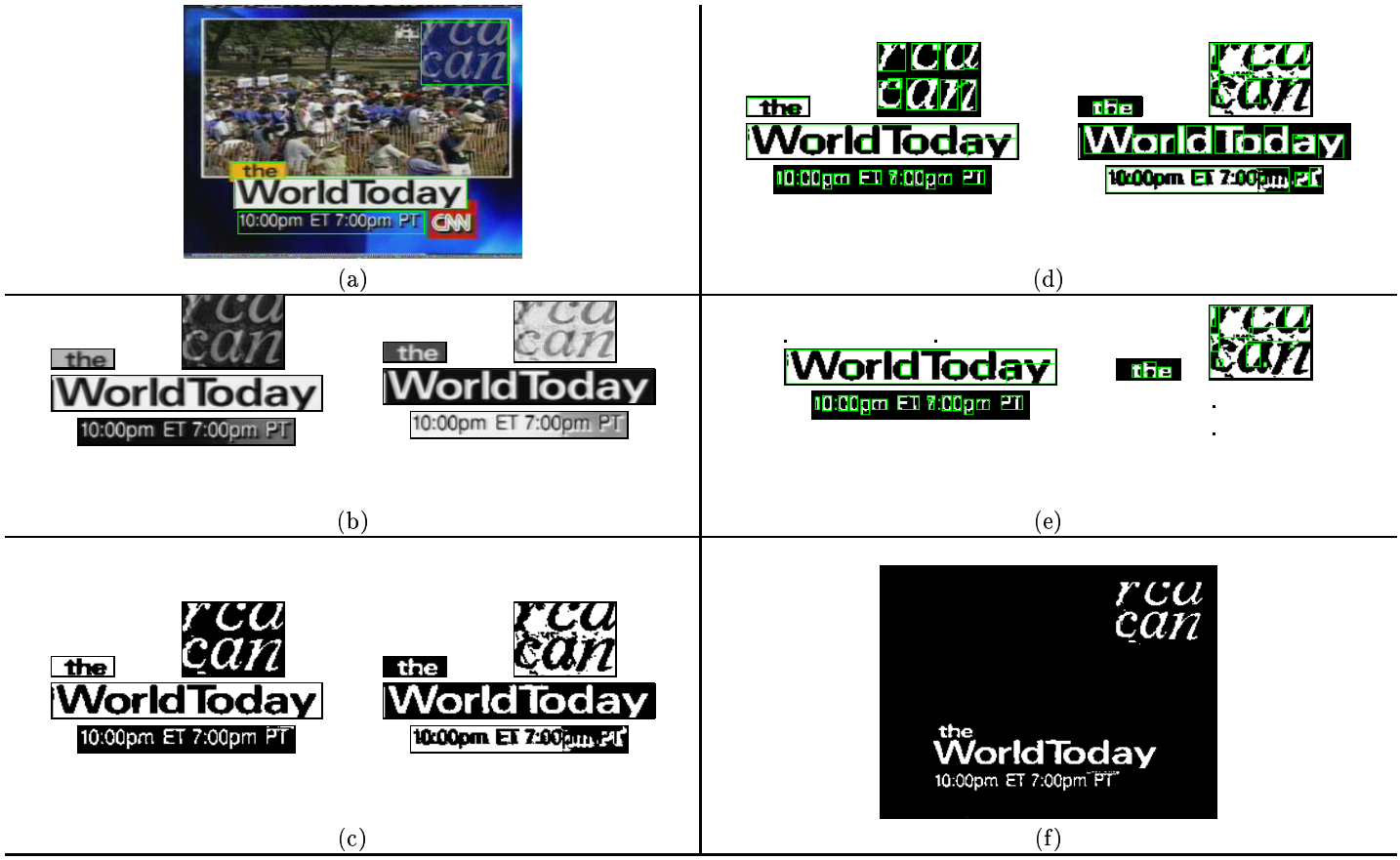


Fig. 4. Steps of the binarization algorithm: (a) video frame marked with localized text boxes; (b) preprocessing is applied on each text box, and the inverse of each is obtained; (c) logical level thresholding is applied to both polarities; (d) connected component analysis is performed and heuristics remove non-character components; (e) correct polarity for each text region is chosen; (f) final binarization result.

or noise. Heuristics are applied to preserve characters while removing noise.

Components having height less than 5 pixels or area less than 12 pixels are removed. Connected components this small are unlikely to be characters. Even if they are characters, it would probably not be possible for the OCR module to recognize them. Note that a minimum character width is not enforced because lowercase "l" characters are sometimes one pixel wide.

A component whose aspect ratio $\frac{width}{height}$ is very large or very small is discarded. These components are often horizontal or vertical lines, or other noise. We currently use the range $[0.1, 1.0]$ as acceptable aspect ratios.

A sample filtered result is shown in Figure 4(d).

Choice of binarization polarity As noted earlier, the logical level thresholding is applied on both the original localized text region and its inverse. In one of the polarities, the text is lighter than the background; in the other, it is darker. Logical level thresholding applied to the dark-text image will result in a binarization of the text. When applied to the light-text image, the algorithm will binarize the background. The correct binarization will have connected components with spacing, size, and alignment consistent with text characters. The incorrect binarization has irregular components of varying size and unusual alignment.

Our algorithm chooses the correct binarization by analyzing statistics about the connected components in each binarization polarity. A voting strategy is used. For each statistic, a vote is cast for the binarization that is more text-like. The binarization with the most votes is chosen as the final binarization output. The criteria are:

- **Height similarity:** Low standard deviation of connected component heights
- **Width similarity:** Low standard deviation of connected component widths
- **Spacing consistency:** Low standard deviation of horizontal distance between adjacent component centers
- **Horizontal alignment:** High number of pairs of components whose bottoms share roughly the same vertical scan line
- **Character-like aspect ratio:** Low difference between average component aspect ratio and 1.0
- **Clean spacing:** Low number of pixels that occur within the bounding box of more than one connected component

– **Periodicity of vertical projection:** The even spacing of text characters should cause the vertical projection to be roughly periodic. The more periodic of the two polarities is chosen using the method presented in [9].

Note that the number of votes for the winner is a confidence measure. A close vote may indicate that the localized region does not actually contain text. We have not exploited this confidence measure in our implementation.

For the ongoing example, Figure 4(e) shows the text boxes chosen by this voting process. The result of binarization for the example is shown in Figure 4(f).

3.3 Text tracking

This section describes two algorithms for tracking text over time. The first algorithm is a simple, very efficient tracker for text that remains rigid over time and that does not exhibit rotation or perspective distortion. It exploits the MPEG compression standard to optimize speed. The second algorithm allows for text to rotate or change with time.

Rigid text tracking Li and Doermann's work [23] represents the state-of-the-art in text tracking. However, their approach makes assumptions that we would like to avoid. First, it assumes that text moves with constant velocity in a linear trajectory. This assumption could be relaxed by using a more sophisticated trajectory model; however, even this would fail for random, erratic motion. Alternatively the predictive stage could be removed altogether and the size of the template search window could be increased. Unfortunately this could increase the computation cost prohibitively. A least-squared-error search for an $m \times n$ text region over a $w \times w$ pixel search window requires $m \times n \times w^2$ pixel comparisons. Therefore it becomes very expensive to increase the search window because the search operation is of order $O(w^2)$. A second limitation of their algorithm is that it compares all pixels within the localized text region, including background pixels. This can cause the algorithm to track the background instead of the text if the background changes or if text moves over backgrounds of different intensities.

We use the motion vectors of MPEG-compressed video to predict text motion with very little computational cost to the tracker. In effect, the computation cost has already been paid by the MPEG encoder. This idea was inspired by papers by Nakajima *et al* [33], who used motion vectors to detect moving objects in MPEG video, and by Pilu [36], who used them to detect camera motion. For details on the MPEG video standard, the reader is referred to [31].

It may seem trivial to apply MPEG motion vectors to the problem of tracking text in video. Unfortunately, MPEG motion vectors are usually too noisy for direct use in a tracker. This is explained by the following observation. Given a region of one frame, a tracker wishes to find the precise location of that region in the next frame. On the other hand, the goal of the MPEG encoder is to achieve minimal coding requirements in a minimum amount of time. MPEG encoders are willing to trade off motion vector accuracy for a decrease in the encoding time.

Figure 5 illustrates typical motion vectors found in MPEG video. Three consecutive P-frames of an MPEG encoded video are shown in (a). The video contains upward-scrolling text. Graphical representations of the macroblock boundaries and motion vectors found in the MPEG bit stream for these three frames are shown in (b). The grid indicates the macroblock boundaries. Macroblocks marked with an "X" are I-coded macroblocks. For macroblocks that are motion-compensated, the motion vectors are drawn from the macroblock to the center of the region used for motion compensation in the previous frame. Empty macroblocks have motion vector length zero. It is observed in this figure that many of the macroblocks corresponding to the text have motion vectors that accurately indicate the text's motion. However, some of the motion vectors point in random directions. In particular, we observe that macroblocks containing few edges tend to have incorrect motion vectors. Macroblocks containing strong edges tend to be most reliable.

Our algorithm deals with these issues as follows. Given a localized text region in one frame, search the next frame for all macroblocks whose motion vectors point back to any part of the text region. Extract the motion vectors from these macroblocks. Several constraints are then applied to the motion vectors to select only those that are likely to be reliable. Very small motion vectors (less than 2 pixels in magnitude) are noisy and therefore ignored. Motion vectors from relatively featureless macroblocks are also discarded, because they are not likely to be accurate. This is determined by applying a Sobel edge detector on each macroblock, and eliminating macroblocks that contain less than four edge pixels. An alternative would be to judge the "edginess" via the high frequency DCT coefficients.

The magnitude and direction of the remaining motion vectors are then clustered. It is assumed that the largest cluster corresponds to the approximate motion of the text block. The vectors in this cluster are then averaged to yield a single motion vector for the text region.

A text event cannot be tracked through an I-frame in this way, because I-frames do not contain motion vectors. Fortunately, I-frames are relatively rare in an MPEG stream (typically one every ten or twelve frames). Tracking through an I-frame is handled by averaging the motion vectors determined for the region in the neighboring frames.

We have found that tracking using the process above is generally quite accurate over short video sequences. However, any small errors made in the tracking from one frame to the next propagate through the entire lifetime of the text event.

We therefore employ a least-squared-error correspondence search around a very small neighborhood (4 pixels) of the location predicted by the MPEG motion vector analysis. Instead of comparing pixel gray levels directly,

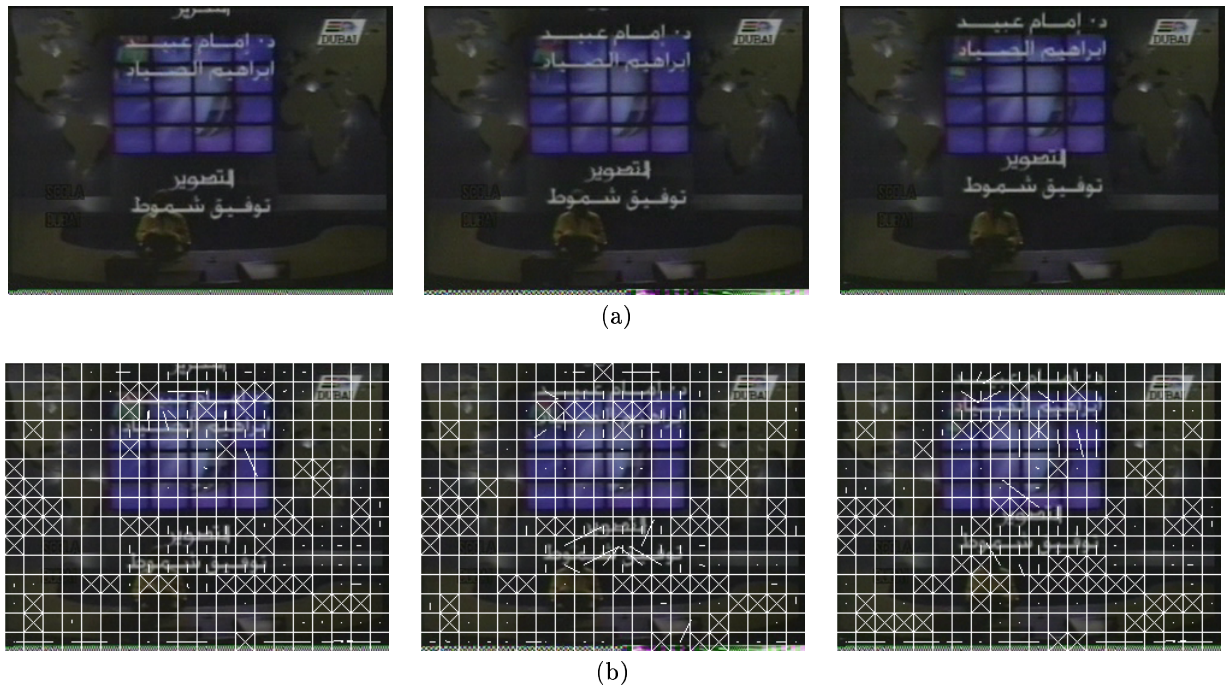


Fig. 5. MPEG motion vectors indicate object motion but are noisy. *(a):* Three consecutive P-frames in an MPEG-1 video. *(b):* The same three frames overlaid with graphical representations of the macroblock boundaries and motion vectors.

we perform the correspondence search only on edge pixels (pixels with high gradient). This encourages the algorithm to match only on the text pixels and not on the background pixels. Matching on edges implies that text can move across backgrounds of different colors without affecting tracking reliability.

MPEG encoders generally use a search window of 32 pixels in each direction during motion compensation searches [31]. This creates a large computation cost and accounts for the slow performance of MPEG encoding. The tracker algorithm is able to take advantage of this wide search window "for free." No assumptions on the trajectory of text are made. Assuming a 32 pixel search window during MPEG encoding, a text event would have to move at a speed greater than 32 pixels per frame in order for the tracking algorithm to fail. Text moving this fast is very rare, as it would travel from one edge of a 320×240 pixel video to the other edge in a third of a second.

Unfortunately, successful use of motion vectors is highly dependent on the quality of the MPEG motion vectors. It is possible (although rare) to encode a video using only I-frames, or to use a very small search window during motion compensation. To handle these cases, we also include a simple trajectory-based prediction similar to Li and Doermann's algorithm. A record of the current trajectory of the text region is kept. After performing the motion vector-based tracking approach described above, text motion is separately predicted using the past trajectory. A least-square-error search is performed around a neighborhood of the predicted location. The lowest error of this search is compared to the lowest error found during the motion vector-based search. Of these two choices, the location with the lowest error is chosen.

Text sometimes scrolls on or off the screen, such that in some frames only a portion of the text event is visible. We include special cases in the algorithm for handling this type of motion. Text exiting the frame is the easier case. The motion determination steps discussed above are applied only on the portion of the text event that is visible. If the computed motion indicates that the text event is exiting the frame, the tracked text box is clipped at the video frame boundary.

Text scrolling into the video is more difficult, because the spatial extent of the text event is not known. We would like the tracker to be able to automatically resize the tracking box as more text enters the frame. This is handled in the following way. The density of edge pixels occurring in the known text region is counted and used as a texture measure. When the tracker detects that the tracking box is moving from the edge of the frame towards the center, the number of edge pixels in the region near the edge is also counted. If the densities of edges of the two regions are comparable, the tracking box is expanded to accommodate the incoming text.

Tracking changing text The last section focused on tracking rigid text. However, caption text events can change. Text can grow, shrink, or rotate. In this section, we describe a method for tracking text that changes in these ways over time.

Instead of a stand-alone tracking algorithm, we propose tightly coupling the detection and tracking modules. The detection and localization algorithm identifies

text instances in each frame. It is the responsibility of the tracker to determine which text instances (if any) in adjacent frames correspond to the same text event.

Two text instances belong to the same text event if the *content* of the text is the same, regardless of changes in size, location, etc. It follows that although some characteristics of a text event may change over time, the basic shapes of the characters remain constant. This property can be exploited to determine whether two text boxes correspond to the same text event. Note that this comparison could be performed on recognized text after OCR has been performed. However, in many cases, a suitable OCR module may not be available, especially if the text language is unknown *a priori*. OCR algorithms also tend to incur a high computational cost. Our character shape comparison approach is a lightweight, language-invariant alternative.

We analyze two consecutive frames at a time. First, the text box localization algorithm described in Section 3.1 is applied to each frame. Oriented text instances are made horizontal by applying a rotation transformation. Our text binarization algorithm, described in Section 3.2 is next applied on each text instance.

Connected component analysis is performed on the binarized text to locate individual characters. The contour of each connected component is traversed. Each contour is parameterized as two 1-D functions $\theta(t)$ and $r(t)$,

$$\theta(t) = \tan\left(\frac{y(t) - y_0}{x(t) - x_0}\right)$$

$$r(t) = \sqrt{(x(t) - x_0)^2 + (y(t) - y_0)^2}$$

where $(x(t), y(t))$ is a point on the contour, and (x_0, y_0) is a reference point. To smooth out noise introduced by imprecise binarization, a low-pass filter is applied to both functions by convolving with a Gaussian.

The resulting functions $\theta_s(t)$ and $r_s(t)$ 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 the features (using the algorithm in [49]). The result is a set of points P for each localized text box, indicating the coordinates of each feature point with respect to the upper-left corner of the text box. The coordinates of P are then normalized by text rectangle height and width to give values between 0 and 1.

To decide whether two text boxes A and B in two adjacent frames belong to the same text event, the normalized feature point sets P_A and P_B are examined. For each point p_i in P_A, the point q_j in P_B having the smallest Euclidean distance from p_i is identified. The sum of the distances over all i is calculated. Then the process is repeated in the reverse direction. That is,

$$D(A, B) = \sum_i \min_j \left(\text{dist}\left(p_i, q_j\right)\right) + \sum_j \min_i \left(\text{dist}\left(p_i, q_j\right)\right)$$

where dist (r, s) is the Euclidean distance between points r and s. The text boxes are declared to belong to the same text event if $D(A, B)$ is below some threshold T_D.

Figure 6 illustrates the process of determining critical point features and comparing them between frames. In the left half of the figure, the images in (a) show two consecutive frames from a video sequence with growing text. The text boxes are binarized, as shown in (b). In (c), the contour of each character has been found, and critical points have been identified. The diagram in (d) shows the normalized feature points of the first frame (small squares) and the second frame (larger squares). Vectors are drawn between each feature point and its nearest neighbor in the other frame. It is observed that the lines between feature points are, in general, relatively short, causing a low value for the shape difference $D(A, B)$. This is expected because the text regions correspond to the same text event. Most of the longer vectors are caused by the non-character connected components introduced by imperfect binarization.

The right half of Figure 6 shows text in two adjacent frames that is not part of the same text event. The binarization and feature point extraction steps are presented in images (b) and (c). The normalized feature points and vectors are shown in (d). We observe that the vectors are longer and appear more random than those in the left side of the figure. This causes the $D(A, B)$ shape difference metric to be high, indicating that the two text boxes are from different text events.

4 Results and Discussion

4.1 Detection and Localization Results

Sample Results Figure 7 presents sample results of our detection and localization algorithms applied to 320 × 240 pixel MPEG-1 video frames captured from television channels. The images demonstrate the algorithm's effectiveness on both simple, horizontal caption text (first row) and more complex oriented text (third row). While the algorithm was designed for caption text, it can detect prominent scene text as well (second row, third column), although it misses more challenging scene text (third row, first column).

Performance Evaluation There have been few quantitative, comparative performance evaluations of text detection algorithms presented in the literature. In this section, we present the results of a quantitative evaluation of our detection algorithm and four others from the literature. We have presented a similar evaluation in [4].

Two datasets were used in the evaluation:

- **Dataset A** consisted of 15 MPEG-1 video sequences with 320 × 240 pixel resolution. There were a total of 10299 frames (about 175 megabytes of data). There were 156 caption text events and 144 scene text events in the video data. All text had horizontal orientation and most text events were stationary. The dataset contained a wide variety of video captured from television channels, including television commercials and news broadcasts (domestic and foreign). A wide variety of text fonts, colors, languages, and scripts were represented.

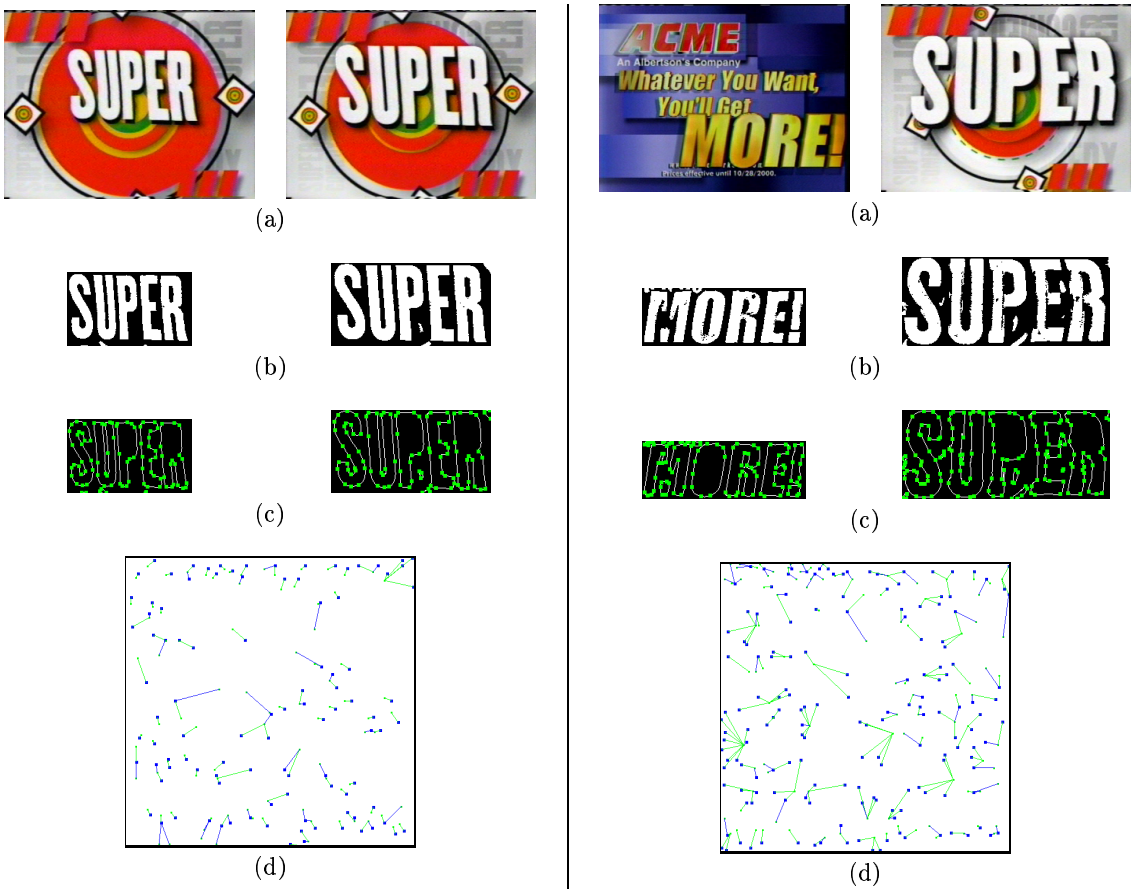


Fig. 6. Text feature point extraction and comparison for two consecutive video frames containing different text events.

Fig. 7. Examples of detected text of various types, sizes, and orientations.

– **Dataset B** consisted of 1 MPEG-1 video sequence with 320×240 pixel resolution. There were a total of 916 frames (about 26 megabytes of video data), and 25 caption text events. The dataset consisted of portions of commercials captured from various television channels. A wide variety of text sizes and colors was included in the dataset. All captions were in English. In addition to static text, text events undergoing rotation and size changes were included.

Video sequences were captured at 30 frames per second on an SGI workstation. The movies were converted to MPEG-1 using SGI's dmconvert software encoder. The compressed bit rate was 4.15 megabits per second. The MPEG group of pictures (GOP) size was 12 frames.

Datasets A and B were ground-truthed by hand using the ViPER tool [7]. In each frame, tight bounding rectangles were drawn around any text regions (regardless of whether the text could actually be read).

We desire an evaluation criteria that rewards algorithms for tightly localizing text events, while penalizing them for failing to detect text or for detecting only a portion of text. They should also be penalized for false alarms, or for loose localization of text. Further, the criteria should be objective and automatically computable by a program.

We perform a pixel-by-pixel match of the ground truth against the output of a localization algorithm. A detected pixel is counted as a *correct detect* if it is marked as text in the ground truth. A *false alarm* appears in the algorithm output but not the ground truth. A *missed detect* appears in the ground truth but not the algorithm output. To perform the evaluation, the number of correct detect, false alarm, and missed detect pixels are counted. The results are expressed as recall and precision, where:

$$Recall = \frac{correct\ detects}{correct\ detects + missed\ detects}$$

$$Precision = \frac{correct\ detects}{correct\ detects + false\ alarms}$$

A similar approach has been taken in evaluating algorithms in the document image domain in [25]. Note, however, that they were able to define the ground truth regions as contiguous black pixels at a given granularity (character, word, paragraph, etc.). The situation is more complex in the case of video because the concept of foreground is not as clear and because video frames lack the formal structure typical of documents. We therefore are forced to do pixel-wise comparisons of text bounding boxes instead.

The relative importance of recall and precision depends on the application. For this evaluation, we will assume that recall and precision are equally important. Therefore we will compare algorithms at the point where parameters have been adjusted such that recall and precision are equal.

Our algorithm was evaluated along with five other promising algorithms from the literature. These included a color stroke-based approach [10] (Algorithm 1), an intensity edge-based approach [19] (Algorithm 2), a color clustering approach [29] (Algorithm 3), and two texture-based approaches [40] (Algorithm 4) and [5,6] (Algorithm 5). These algorithms were chosen because they represent a cross-section of the different approaches to text detection in video. To emphasize the objectivity of the performance evaluation, we will refer to the algorithms by number instead of by author names.

Each algorithm requires one or more fixed parameters. The parameters were optimized on Dataset A by varying each parameter over a reasonable range. For each combination of parameter settings, the evaluation was performed on the full 10299 frames. The settings that gave the highest recall and precision under the constraint $recall = precision$ were declared optimal. The recall and precision obtained using these parameter values were used to represent the performance of the algorithm. Note that algorithms were neither penalized nor rewarded for missing or finding scene text.

Table 1 presents the results of the evaluation for Dataset A. It is observed that the new algorithm performed the best, followed very closely by Algorithm 5. This similarity is not surprising because both algorithms employ texture features.

Table 2 presents the evaluation results for Dataset B. Two sets of recall and precision statistics are given. The evaluation was first performed using the parameter values determined as optimal over Dataset A. These results are shown in the second and third columns of Table 2. The parameter values for each algorithm were then varied to find the optimal parameter sets for Dataset B. These results are shown in the fourth and fifth columns of the table. Note that Algorithm 3 was not included in these runs, because this dataset violated its assumption that all text is strictly horizontal.

It is observed that our algorithm gives by far the best performance on Dataset B, with an optimal precision and recall of 74%. Further, the results indicate that the optimal parameters for this algorithm on Dataset A are very close to optimal on Dataset B. This suggests that it is relatively insensitive to the value of its parameters. Algorithms 1, 2, and 4 exhibit optimal precision and recalls of around 49%, about 25 percentage points lower than those of our algorithm. The results also suggest that these algorithms are more sensitive to the values of their parameters. Algorithm 5 gives poor performance on this dataset. This is because much of the text in Dataset B is relatively large, violating the algorithm's maximum text size assumption.

We observe that our algorithm has shown the best performance on both datasets. It performs slightly better than other algorithms in the literature on a dataset containing mostly static, horizontal text. It performs significantly better than other algorithms on a dataset including non-horizontal text that rotates, changes size, and moves over time. This is an encouraging observation, because it demonstrates that it is possible to design text detection algorithms that make fewer assumptions about text in video while maintaining the accuracy typical of algorithms found in the literature.

The results of the quantitative performance evaluation indicate that the absolute performance figures of state-of-the-art detection and localization algorithms are

quite low. It is disappointing to see precision and recall values under 50%. This highlights the need for further research in designing more accurate algorithms that can detect text in general-purpose video. However, there are two caveats to our performance evaluation that should be kept in mind. First, our evaluation criteria is very strict. An algorithm must generate output that exactly matches the ground truth in order to achieve perfect precision and recall. In an actual application, such precision is probably not necessary. Second, our dataset is extremely challenging. The ground truth includes small, low-contrast text that is difficult for a human to read. Such text may not even be useful to an application.

Algorithm	Recall	Precision
Algorithm 1	29%	30%
Algorithm 2	33%	34%
Algorithm 3	40%	39%
Algorithm 4	37%	37%
Algorithm 5	46%	45%
Proposed algorithm	46%	48%

Table 1. Detection/localization algorithm performance for caption text on Dataset A. Dataset A contains mostly horizontal, static text events.

	Preset parameter set		Optimal parameter set	
Algorithm	Recall	Precision	Recall	Precision
Algorithm 1	37%	62%	46%	48%
Algorithm 2	25%	73%	49%	49%
Algorithm 4	37%	58%	47%	48%
Algorithm 5	36%	36%	36%	36%
Proposed algorithm	73%	75%	74%	74%

Table 2. Detection/localization algorithm performance for caption text on Dataset B. Dataset B includes text that moves, rotates, grows, and shrinks over time. Results are shown both for when the parameters were set to values found optimal for Dataset A, and when set to those found optimal for Dataset B.

4.2 Binarization Algorithm Results

Figure 8 presents results of the binarization algorithm on localized text boxes in sample video frames. The left side of the figure shows results from typical video frames. Note that the algorithm does a good job of binarizing some scene text, such as in the bottom image.

The right side of Figure 8 shows results of the binarization algorithm applied on very challenging video frames. These examples highlight some of the problems with our binarization algorithm. The top example shows the binarization of Arabic caption text. The output of our algorithm has given reasonable results for three of the text boxes. However, it failed to select the correct binarization polarity for the top text box. This can be explained by reviewing the polarity selection criteria described in Section 3.2. Many of the criteria assume that connected components correspond to text characters. This assumption is not valid for the Arabic script in this example. We conclude that our polarity selection gives accurate results only for scripts with separated characters. Alternative selection criteria could be devised to handle other scripts.

The second example on the right side of Figure 8 shows the algorithm applied to very small text. The average character size here is about 8 pixels high by 5 pixels wide, with a sub-pixel stroke width. The binarization is reasonable, but is probably not clean enough for accurate recognition.

The third row of images on the right side of Figure 8 shows binarization of low-contrast text. While the algorithm produces reasonable results, there is noise that could cause recognition to fail. Binarization of low-contrast text is another area that requires further research. Note that the shadow effect exhibited by the text in this example has not confused our binarization algorithm.

4.3 Tracking Results

Rigid Tracking Results In this section, we present results of running the rigid tracking algorithm on a variety of video sequences. Except for "poster.mpg", all video sequences have a 320×240 pixel frame size and a 30 fps frame rate. They were captured to MPEG-1 as described in Section 4.1. "Poster.mpg" was captured at 15 fps using a hand-held camera connected to a SunVideo hardware MPEG-1 compression card. Text regions were marked by hand in the first frame of each sequence, and the algorithm automatically tracked the regions for the remainder of the sequence.

Figure 9 shows the algorithm tracking caption text scrolling horizontally. Text is entering and exiting the frame, and the algorithm must determine the bounding boxes on the incoming text.

Although the tracking algorithm was designed for caption text, we have found that it works for quasi-rigid scene text events as well. Figure 10 demonstrates this. The algorithm tracks successfully despite some perspective distortion. The algorithm's robustness to erratic, fast motion is demonstrated in Figure 11. A tracker assuming a simple linear trajectory model would fail in this case. This example was chosen because it poses a significantly greater challenge to the tracker than the motion of typical caption text.

Evaluation of Non-rigid Text Tracking Experimentation was performed to investigate our non-rigid tracking method's accuracy. A dataset of 27 video sequences, each containing one caption text event, was captured from television commercials. The data was captured and ground-truthed in the same manner described in Section 4.1. There were

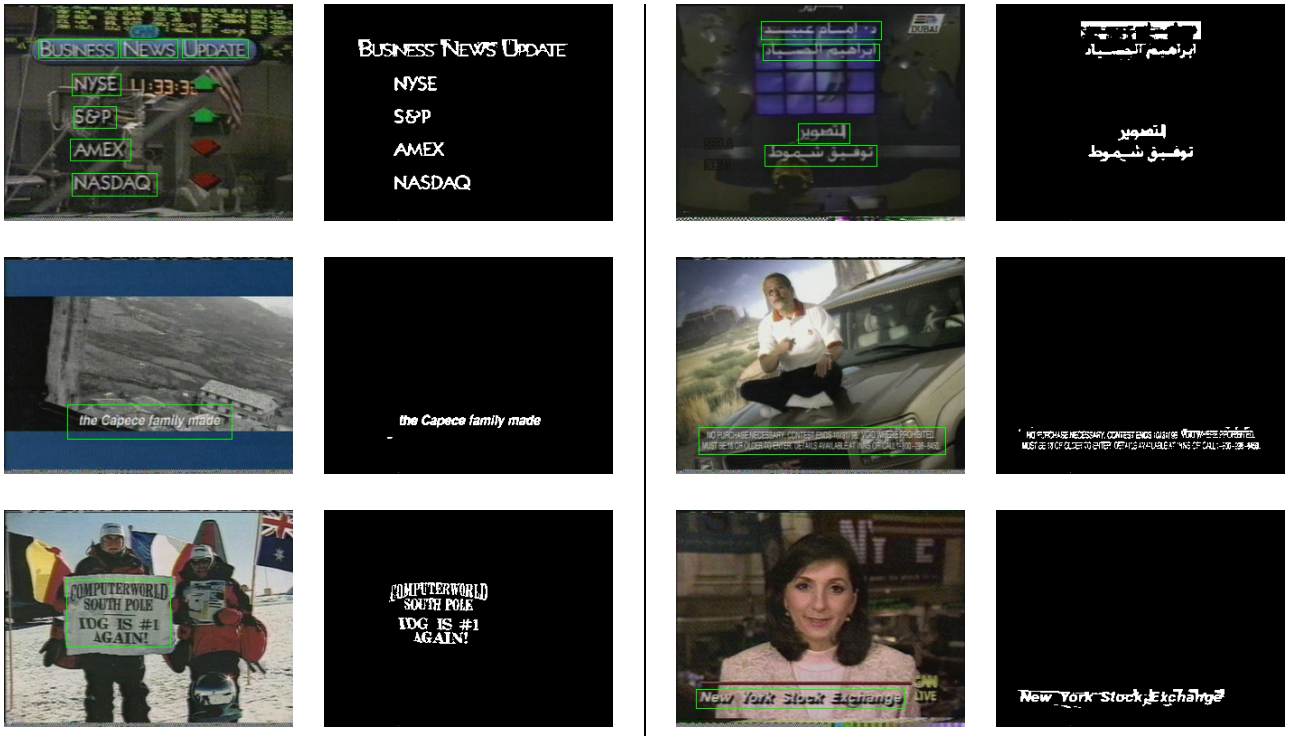


Fig. 8. Binarization results for sample video frames.

frame 1756 frame 1771 frame 1786 frame 1801

frame 1831 frame 1861 frame 1891 frame 1921

Fig. 9. Tracking algorithm applied to "scrolling7.mpg" video sequence with horizontally-scrolling text entering and exiting the frame.

a total of 1005 frames in the dataset. A variety of growing, shrinking, moving, and rotating text events were included. The 27 individual video sequences were combined into a single video sequence by appending together randomly-selected groups of adjacent frames of random lengths from the video sequences. The result was a single 1005 frame video sequence with 111 text events.

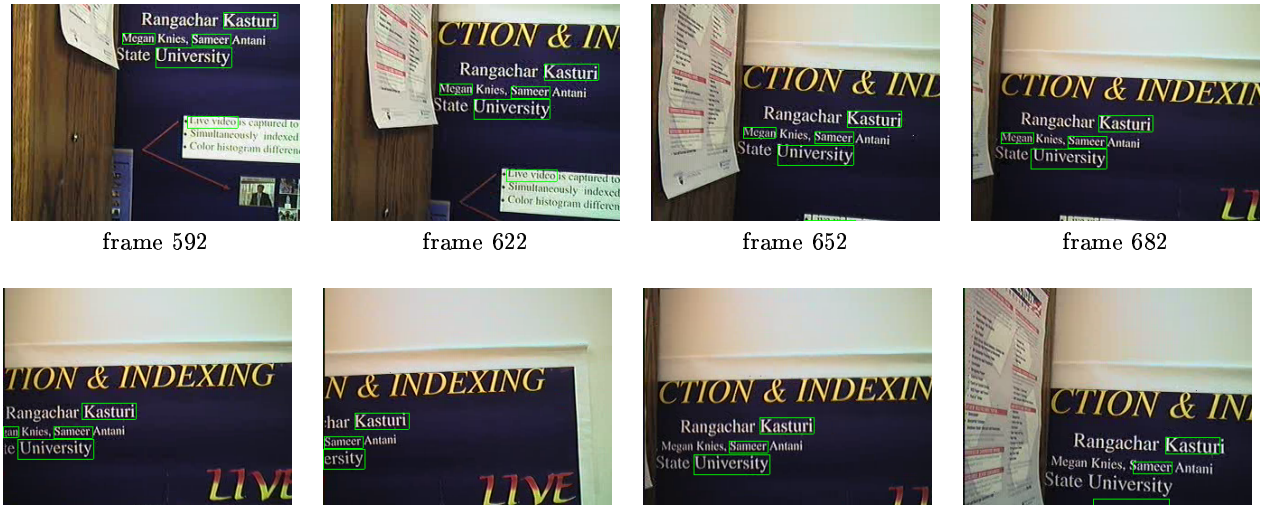
The evaluation was carried out as follows. The algorithm was run on the 1005-frame video sequence. For each pair of consecutive frames, the algorithm decided whether the text in the two frames belonged to the same text event. If the algorithm correctly determined that the text belonged to the same text event, a *correct detect* was recorded. If the algorithm incorrectly concluded that the two frames shared a text event, a *missed detect* was tallied. If the algorithm incorrectly concluded that the two frames had different text events, a *false alarm* was recorded. Precision and recall statistics were then computed using the definitions presented in Section 4.1.

Figure 12 presents the results of the experimentation as an ROC curve. It is observed from the ROC curve that very good precision and recall can be achieved. The optimal threshold value depends on the needs of the application. For example, for an application in which pre-

frame 1171 frame 1192 frame 1195 frame 1198

frame 1201 frame 1204 frame 1207 frame 1210

Fig. 10. Tracking algorithm applied to scene text in "foxsports.mpg" video sequence.



frame 592 frame 622 frame 652 frame 682

frame 712 frame 742 frame 772 frame 802

Fig. 11. Tracking algorithm applied to scene text with erratic motion in "poster.mpg" video sequence.

cision and recall are equally important, a threshold of $T_D = 0.55$ is optimal, at which precision and recall are both 97.5%. In a video indexing application, however, it is likely that a high recall would be more important than a high precision. This is because it is very important that all text events are entered at least once into the index, while duplicate entries are not harmful. It is observed from the ROC curve that it is possible to obtain a recall near 100% with a precision of 96% at $T_D = 400$.

Figure 13 shows sample qualitative results of the combined text detection, localization, and tracking steps. The figure shows eight frames from a commercial featuring rotating, shrinking, and growing text, and the boxes localized by our algorithm. The tracking algorithm concluded that the text boxes in frames 1 through 7 correspond to the same text event. Similarly the algorithm determined that frames 12 through 24 belong to a separate text event. Frame 10 confused the algorithm due to the overlapping text. The tracking algorithm concluded that frame 10 belonged to its own, one-frame text event.

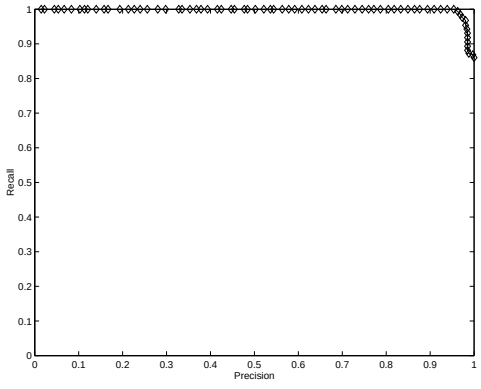


Fig. 12. ROC curve of tracker performance.

Figure 14 demonstrates the algorithm's effectiveness on text occurring against complex, unconstrained backgrounds. The tracking algorithm correctly identified the text in frame 1 as one event, and the text occurring in frames 2 through 6 as another text event.

5 Summary and Conclusions

This paper has discussed extraction of text from video. We have discussed several sub-problems of text extraction, including detection, localization, tracking, and binarization. We have proposed an algorithm that detects and localizes text of unconstrained size and orientation. We have presented two tracking algorithms: a rigid text tracker and a tracker for text that grows, shrinks, and rotates. Our binarization algorithm handles text of arbitrary color superimposed on complex color backgrounds. Few papers in the literature to date have addressed such a variety of text types.

Text extraction is still an open problem and much work in the area remains. Our evaluation of state-of-the-art detection and localization algorithms showed that no algorithm could achieve greater than 50% recall and precision simultaneously on our video dataset. For application in a video indexing system, algorithms with better accuracy are needed. Better features must be identified that can robustly distinguish between text and non-text regions. One possibility is to combine outputs of multiple detection algorithms to produce better output [3]. Another possibility is to perform a shape analysis of candidate text characters. For example, the frequency of corners and edges of the shapes in a region could be used to remove very simple shapes unlikely to be text characters.

Our binarization algorithm works well with most font sizes. However, there is some text in broadcast video with stroke width less than one pixel. Connected component labeling on small fonts gives inaccurate results, causing our binarization algorithms to fail. We are exploring an alternative approach for binarizing very small text based on topographical analysis [21].

We are trying to incorporate color features into our algorithms. We have tried using color clustering [15] to separate text strokes from the background. This approach worked once the parameters of the color clustering algorithm were manually adjusted for each text instance. However an automatic mechanism for setting these parameters will be necessary to make this a useful approach.

In addition to growing, shrinking, and rotating text, other types of special effects text can be found. For example, text can break into pieces, or morph between fonts, or undergo perspective distortion. The tracking algorithm presented in this paper works for some of these cases, but it could be extended to handle more types of stylized text.

The recognition problem has not been covered in this paper. Several researchers [13,45,48] have attempted recognition from images and video. However even with constraints on the video dataset and application-specific text dictionaries available *a priori*, recognition accuracy has been low. More research is needed to design OCR modules geared specifically for the unique challenges of text in video.

References

1. L. Agnihotri and N. Dimitrova. Text Detection for Video Analysis. In *Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries*, pages 109–113, 1999.
2. S. Antani, D. Crandall, and R. Kasturi. Robust extraction of text in video. In *Proc. Intl. Conf. on Pattern Recognition*, volume 3, pages 831–834, 2000.
3. 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.
4. S. Antani, D. Crandall, A. Narasimamurthy, Y. Mariano, and R. Kasturi. Evaluation of Methods for Extraction of Text from Video. In *IAPR Intl. Workshop on Document Analysis Systems*, pages 507–514, 2000.
5. N. Chaddha, R. Sharma, A. Agrawal, and A. Gupta. Text Segmentation in Mixed–Mode Images. In *28th Asilomar Conf. on Signals, Systems and Computers*, pages 1356–1361, October 1995.
6. David J. Crandall. Extraction of Unconstrained Caption Text from General-Purpose Video. Master's thesis, The Pennsylvania State University, University Park, PA 16802, USA, May 2001.
7. D. Doermann and D. Mihalcik. Tools and techniques for video performance evaluation. In *Proc. Intl. Conf. on Pattern Recognition*, pages 167–170, 2000.
8. R. Dugad and N. Ahuja. A Fast Scheme for Altering Resolution in the Compressed Domain. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 213–218, 1999.
9. C. Garcia and X. Apostolidis. Text detection and segmentation in complex color images. In *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, pages 2326–2329, 2000.
10. U. Gargi, S. Antani, and R. Kasturi. Indexing text events in digital video databases. In *Proc. Intl. Conf. on Pattern Recognition*, volume 1, pages 916–918, 1998.
11. U. Gargi, D. Crandall, S. Antani, T. Gandhi, R. Keener, and R. Kasturi. A system for automatic text detection in video. In *Intl. Conf. on Document Analysis and Recognition*, pages 29–32, 1999.
12. R.C. Gonzalez and R.E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, Inc., 1993.
13. O. Hori. A video text extraction method for character recognition. In *Intl. Conf. on Document Analysis and Recognition*, pages 25–28, 1999.
14. J. Huang, Z. Liu, Y. Wang, Y. Chen, and E.K. Wong. Integration of multimodal features for video classification based on hmm. In *Proc. IEEE Signal Processing Society Workshop on Multimedia Signal Processing*, 1999.
15. A.K. Jain. *Algorithms for clustering data*. Prentice Hall, Englewood Cliffs, NJ, 1988.

frame 1 frame 4 frame 6 frame 7

frame 10 frame 12 frame 17 frame 24

Fig. 13. Sample results of combined text detection and tracking on growing, shrinking, and rotating text.

frame 1 frame 2 frame 3

frame 4 frame 5 frame 6

Fig. 14. Sample results of combined text detection and tracking on growing text against an unconstrained background.

16. A.K. Jain and B. Yu. Automatic Text Location in Images and Video Frames. *Pattern Recognition*, 31(12):2055–2076, 1998.
17. K.Y. Jeong, K. Jung, E.Y. Kim, and H.J Kim. Neural network-based text location for news video indexing. In *Proc. IEEE Intl. Conf. on Image Processing*, pages 319–323, 1999.
18. 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.
19. F. LeBourgeois. Robust Multifont OCR System from Gray Level Images. In *Intl. Conf. on Document Analysis and Recognition*, volume 1, pages 1–5, 1997.
20. C.-M. Lee and A. Kankanhalli. Automatic Extraction of Characters in Complex Scene Images. *Intl. Journal of Pattern Recognition and Artificial Intelligence*, 9(1):67–82, February 1995.
21. 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.
22. H. Li and D. Doermann. Automatic text tracking in digital videos. In *IEEE Second Workshop on Multimedia Signal Processing*, pages 21–26, 1998.
23. H. Li, D. Doermann, and O. Kia. Text extraction and recognition in digital video. In *IAPR Intl. Workshop on Document Analysis Systems*, pages 119–128, 1998.

24. 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.
25. J. Liang, I.T. Phillips, and R.M. Haralick. Performance evaluation of document layout analysis algorithms on the uw data set. In *IS&T/SPIE Conference on Document Recognition*, volume 3027, pages 149–160, 1997.
26. R. Lienhart and F. Stuber. Automatic Text Recognition for Video Indexing. In *Proceedings of the ACM Intl. Multimedia Conf. & Exhibition*, pages 11–20, 1996.
27. R. Lienhart and F. Stuber. Automatic Text Recognition in Digital Videos. In *Proceedings of SPIE*, volume 2666, pages 180–188, 1996.
28. R. Lienhart and F. Stuber. Indexing and Retrieval of Digital Video Sequences based on Automatic Text Recognition. In *Proceedings of the ACM Intl. Multimedia Conf. & Exhibition*, pages 419–420, 1996.
29. V.Y. Mariano and R. Kasturi. Locating Uniform-Colored Text in Video Frames. In *Proc. Intl. Conf. on Pattern Recognition*, volume 4, pages 539–542, 2000.
30. 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.
31. Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. LeGall. *MPEG Video Compression Standard*. Digital Multimedia Standards Series. Chapman and Hall, 1997.
32. G. Nagy. Twenty Years of Document Image Analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, 2000.
33. 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.
34. W. Niblack. *An introduction to digital image processing*. Prentice-Hall International, 1986.
35. J. Ohya, A. Shio, and S. Akamatsu. Recognizing Characters in Scene Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:214–224, 1994.
36. M. Pilu. On Using Raw MPEG Motion Vectors to Determine Global Camera Motion. In *Proceedings of SPIE*, volume 3309, pages 448–459, 1998.
37. W.K. Pratt. *Digital Image Processing, 2nd Ed.* John Wiley & Sons, 1991.
38. W. Qi, L. Gu, H. Jiang, X.R. Chen, and H.J. Zhang. Integrating visual, audio and text analysis for news video. In *Proc. IEEE Intl. Conf. on Image Processing*, pages 520–523, 2000.
39. T. Sato, T Kanade, E.K. Hughes, and M.A. Smith. Video OCR for Digital News Archive. In *IEEE Intl. Workshop on Content–Based Access of Image and Video Databases CAIVD'98*, pages 52–60, January 1998.
40. 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.
41. J.C. Shim, C. Dorai, and R. Bolle. Automatic Text Extraction from Video for Content-Based Annotation and Retrieval. In *Proc. Intl. Conf. on Pattern Recognition*, pages 618–620, 1998.
42. J.C. Shim, C. Dorai, and R. Bolle. Automatic Text Extraction from Video for Content-Based Annotation and Retrieval. Technical Report RC21087(94340), IBM T.J. Watson Research Division, Yorktown Heights, NY, 1998.
43. C.S. Shin, K.I. Kim, M.H. Park, and H.J. Kim. Support vector machine-based text detection in digital video. In *Proc. IEEE Signal Processing Society Workshop*, pages 634–641, 2000.
44. H.C. Tom and H.K. Katsaggelos. Resolution enhancement of monochrome and color video using motion compensation. *IEEE Transactions on Image Processing*, 10(2):278–287, 2001.
45. Y. Watanabe, Y. Okada, K. Kaneji, and Y. Sakamoto. Retrieving related tv news reports and newspaper articles. *IEEE Intelligent Systems*, pages 40–44, September/October 1999.
46. 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.
47. E.K. Wong and M. Chen. A robust algorithm for text extraction in color video. In *Proc. IEEE Intl. Conf. on Multimedia and Expo*, pages 797–800, 2000.
48. V. Wu, R. Manmatha, and E.M. Riseman. Finding Text in Images. In *Second ACM Intl. Conf. on Digital Libraries*, 1997.
49. 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.