Robust Detection of Stylized Text Events in Digital Video

David Crandall Rangachar Kasturi
Department of Computer Science & Engineering
The Pennsylvania State University
University Park, PA 16802
{crandall, kasturi}@cse.psu.edu

Abstract

Automatic content-based video indexing is an important research problem. One approach is to extract text appearing in video as an indication of a scene's semantic content. Most work so far has focused only on detecting the spatial extent of text instances 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 necessary to determine the temporal extent of text events by combining the results of text detection on individual frames over time. This is a non-trivial problem because a text event may move, rotate, grow, shrink, or otherwise change throughout its lifetime. Such text effects are common in television programs and commercials to attract viewer attention, but have so far been ignored in the literature. In this paper, we present a method for detecting and tracking moving, changing caption text events in MPEG–1 compressed video.

1. Introduction

As the popularity of digital video grows, quantities of available video are rapidly increasing. Efficient navigation of such vast collections of digital video requires automatic indexing based on *content.* Algorithms are needed that automatically extract semantic information from video.

One approach is to extract text occurring in video. Text in video often reflects semantic content to some extent. For example, captions in television newscasts summarize relevant names, locations, and times pertaining to a news story. Movies and television programs include credits that indicate names of characters and actors. Text in commercials indicates company name and product information.

Extraction of text from video frames is more challenging than that from document images. Color video frames have much lower resolution than document images. Lossy video compression techniques (such as MPEG) cause color bleeding, blocking artifacts, and loss of contrast. Text stroke color is not known *a priori*, and the background is often complex and changing.

In addition, the temporal nature of video must be considered. Text in video usually persists for several seconds, creating a *text event.* During its lifetime, the text may remain stationary or it may move. It may remain rigid, or it may shrink, grow, rotate, or change perspective. Characters may break apart or join together. As long as the content of the text string remains the same from frame to frame, it is still part of the same text event.

While a significant amount of work is found in the literature on extracting text from images, there has been relatively little work on text extraction from video. Most approaches simply apply text extraction on individual video frames without considering the temporal dimension. Different algorithms exploit different features to localize text. Popular approaches include examining edge features [1, 12], stroke features [5], color features [7, 10], and texture features [14]. At Penn State, we have developed a system that combines the output of multiple text detection algorithms using a decision fusion module for more robust detection results [6].

Some work has considered the temporal nature of video to some extent. Shim *et al* [13] identify regions with homogeneous intensity, obtain positive and negative images by double thresholding, and apply heuristics to eliminate non-text regions. Inter-frame analysis is performed to reduce sporadic false-alarms and missed-detects caused by noisy individual frames. It is assumed that text remains stationary over time. Lienhart [9] segments individual video frames using properties of local color histograms and chooses regions corresponding to text based on heuristics. Temporal analysis is used to refine detection results. When a potential text region is detected in a frame, the next frame is searched for a region similar in size, color, and shape. If such an area is not found, the region is discarded as non-text. This approach assumes text remains rigid and moves in a simple, linear manner. Li and Doermann [8] use a neu-

0-7695-1263-1/01/$10.00 © 2001 IEEE

ral network trained on texture features to find text regions. They also propose a text tracker for following moving text. The tracker is based on a simple least-square-error correspondence search around a small neighborhood. In postprocessing, tracking results are refined by analyzing character edges to handle text that changes size slowly from frame to frame.

We observe the following about the text extraction from video work found in the literature to date. Extracting stationary, horizontal text has been well-studied. However, detection of stylized text that rotates or changes size has been ignored.

In this paper, we consider the problem of identifying caption text events in MPEG-1 digital video. In Section 2, we describe an efficient algorithm for detecting and localizing caption text of arbitrary size and orientation in video frames. In Section 3, an algorithm for tracking text is proposed. The text tracker groups text located in individual frames into text events that persist over time. The tracker allows text to rotate, shrink, or grow, as long as the text content remains the same.

2. Caption text detection and localization in video frames

We are not aware of any work in the video domain that has considered text that is not horizontally-aligned. To detect stylized text events, like rotating captions, we require an algorithm that can detect text of arbitrary orientation.

2.1. A DCT-based text detection algorithm

We use the blockwise two-dimensional discrete cosine transform (DCT) as a texture measure to detect text in video. This is attractive for use in video because the MPEG compression standard employs 8 × 8 pixel blockwise DCT. DCT features have been successfully applied to text region detection in document images [4]. In a recent performance evaluation of text detection algorithms, we found that a DCT-based method for detecting constrained text in video frames gave the best performance [3]. The method described here is inspired by these DCT algorithms.

The strong edge features and unique texture of text create a unique signature in the frequency domain. This causes text blocks to have high coefficient values for certain frequencies. The text texture energy of a block can be computed by summing selected coefficients within the DCT coefficient matrix. We also observe that text of arbitrary orientation has a combination of horizontal and vertical text texture energy. The sum of a text region's horizontal and vertical energy components gives the overall text texture energy.

These observations motivate the following method for detecting text blocks. For each luminance plane DCT block of a frame, the horizontal text texture energy TTE_h is computed by summing the selected coefficients found to be optimal in [4]. Similarly, the vertical text texture energy TTE_v is computed by transposing the DCT coefficient matrix, and then summing the 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 TTE_v to grow vertically. 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.

The fixed block size causes text with large stroke width to be missed. To handle text of arbitrary size, we employ a hierarchical subsampling process. After applying text detection on an original frame, it is subsampled in half. Text detection is again applied. For each text block detected at this level in the hierarchy, the corresponding four blocks are marked as text in the original image. In our implementation, we continue this procedure until a resolution of less than 160 × 120 has been reached.

2.2. Text box localization

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

We propose an iterative greedy algorithm for localizing text instances. First, connected component analysis is performed for the blocks marked 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 fraction of detected text pixels that lie underneath a rectangle, and P_{nt} is the fraction of the rectangle's area that covers non-text pixels. During each iteration, each rectangle is visited, and one of the following actions is taken:

- Rectangle is left unchanged
- Rectangle height or width is incremented or decremented by one block

- Shift rectangle one block horizontally or vertically
- Rotate rectangle by 15 degrees clockwise or counterclockwise

At the end of each iteration, overlapping rectangles are merged 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. In our implementation, rectangles with length or width less than 8 pixels are discarded. Very little text in video is smaller than this size, and even if present, it is unlikely that an OCR module could recognize them accurately.

2.3. Results

Figure 1 presents sample results of the algorithm applied to 320 × 240 pixel MPEG-1 video frames captured from television. The superimposed rectangles indicate the results of text localization. The results demonstrate the algorithm's effectiveness on a variety of text with different sizes, orientations, and language scripts. Note that image (c) includes some text missed by the algorithm. This text is less than 8 pixels tall and thus was discarded by our size heuristic.

The algorithm requires a fixed threshold as a parameter. However, our algorithm automatically adjusts this threshold based on the overall contrast of the video. We have found that this makes the algorithm not very sensitive to the choice of threshold. For example, all of the in Figure 1 were obtained using the same input threshold value.

An advantage to this algorithm is its low computation cost. Since DCT coefficients are available in the I-frames of MPEG video, the block classification step involves trivial addition and comparison operations. The MPEG video file need not be fully decompressed for the algorithm to be applied. The rectangle determination process typically converges in few iterations.

In some applications, it may be sufficient to perform text detection on just I-frames, since they occur relatively frequently in the MPEG stream. To process predictive (P- and B-) frames, it is necessary to reconstruct the DCT coefficients using motion compensation. For speed, it is possible to perform motion vector compensation in the frequency domain [11].

3. Text Event Determination

After text detection is performed on individual video frames, we wish to group text instances into events that persist over time. That is, given text detected in a frame, it is necessary to determine if the same text occurs in the next frame, regardless of whether it has rotated or changed size. If so, both text rectangles are part of the same text event. If not, the current frame represents the end of the text event's lifetime.

3.1. Our approach

It is observed that while the size, orientation, color, and other features of a text event may change over time, the basic shapes of its characters remain constant. This property can be exploited to determine whether two localized text boxes correspond to the same text event.

We analyze two consecutive frames at a time. First, the text box localization algorithm described in Section 2 is applied to each frame. Oriented text instances are made horizontal by applying a rotation transformation.

A text binarization algorithm is next applied to each text instance. We use the binarization algorithm developed in earlier work [2]. This algorithm is designed to handle text of arbitrary and unknown color, so that no assumptions about text color need to be made. Next, connected component analysis is performed on the binarized text to locate individual characters. The contour of each character is traversed. Each contour is then parameterized as two 1-D functions $\theta(t)$ and $r(t)$, representing the angle and distance of each point on the contour from a reference point. A Gaussian filter is applied to smooth these functions.

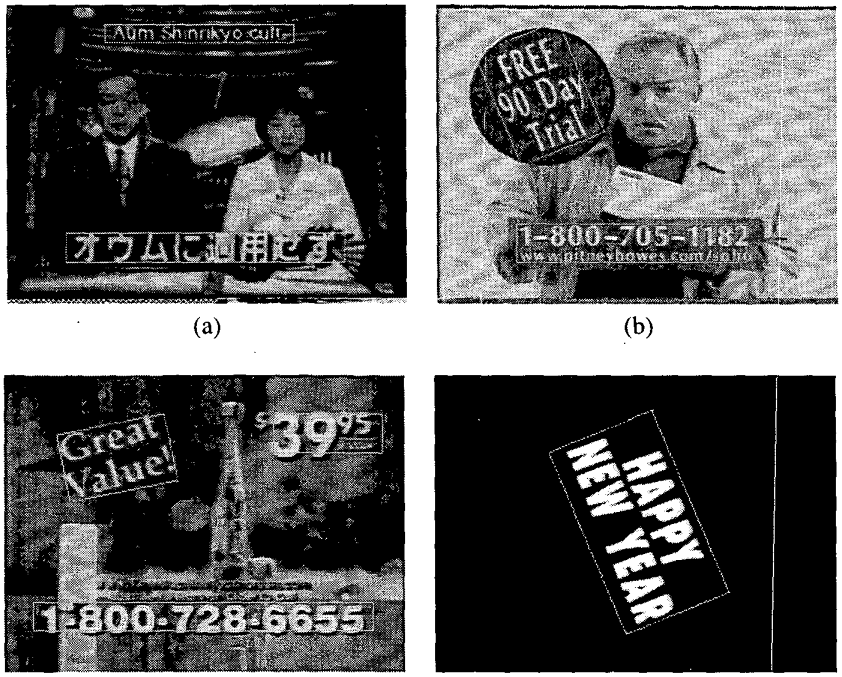
The resulting functions represent a signature of the shape of a given character. From this shape, feature points are extracted. We use the points of maximum curvature (critical points) as our features. Zhu & Chirlian's critical point detection algorithm [15] was used in our implementation.

The row and column coordinates of features points within each detected text rectangle are normalized to be between 0 and 1. To decide whether two text boxes A and B in two adjacent frames belong to the same text event, we employ the following distance measure:

$$\begin{aligned} D(A,B) &= \sum_i \min_j (\mathrm{dist}\,(p_i,q_j)) + \\ &\quad \sum_j \min_i (\mathrm{dist}\,(p_i,q_j)) \end{aligned}$$

where p_i is a normalized feature point in box A, q_j is a normalized feature point in box B, and dist (r,s) is the Euclidean distance between points r and s.

Text boxes within the two consecutive frames are then analyzed as follows. For each pair of text boxes in the two frames, the distance $D(A,B)$ is computed, and the pair with the lowest error is chosen. If the lowest error is below a threshold T_D, 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.



(a) (b) (c) (d)

Figure 1. Examples of detected and localized text in video frames.

3.2. Results

Experimentation was performed to investigate the proposed method's accuracy. A dataset of 1005 frames containing 111 text events was captured from television commercials. A variety of growing, shrinking, moving, and rotating text events were included.

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 or not. The outcome of each trial was classified into one of three categories: *correct detect* if the algorithm correctly stated that the text instances were the same, *missed detect* if the algorithm incorrectly stated that they were the same, or *false alarm* if the algorithm incorrectly concluded that they were different. Precision and recall statistics were then computed:

$$\begin{aligned} Recall &= \frac{correct\ detects}{correct\ detects + missed\ detects} \\ Precision &= \frac{correct\ detects}{correct\ detects + false\ alarms} \end{aligned}$$

Figure 2 presents the results of the experimentation as an ROC curve. Each point on the curve shows the precision and recall achieved for one value of threshold T_D. 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 particular application. For example, if precision and recall are equally important, a threshold of $T_D = 0.55$ is optimal, at which point 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, since missing index entries are more harmful than duplicate entries. In this case, a recall of 100% is possible with a precision of 96% at $T_D = 0.40$.

4. Conclusion

We describe algorithms for detecting, localizing, and tracking caption text in digital video. Detection and localization is performed on each individual video frame. Then, tracking is performed to identify text events that persist over time. Unlike other algorithms described in the literature so far, our methods are designed to handle stylized text that moves, rotates, and changes size over time.

Many opportunities for future work in this area exist:

- The algorithms presented here work well for large,

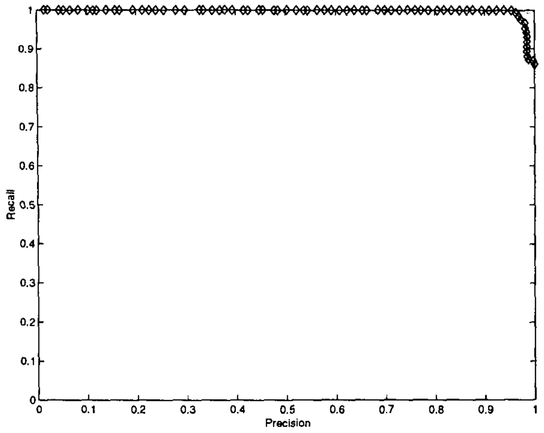


Figure 2. ROC curve of tracker performance.

high-contrast caption text events. Ongoing work includes extending the algorithms to handle small text and text appearing in low-bitrate video data.

- Currently, we are performing the binarization algorithm on each frame individually. It may be possible to obtain a higher-quality, higher-resolution binarization by integrating multiple frames of moving, growing text.
- We have not yet investigated the recognition stage of the video extraction problem. Note that it in some applications, recognition need not be necessary [5].
- While this paper has focused on caption text, the ideas presented here could also be applied to extracting scene text occurring in video. Tracking scene text is challenging due to lighting and perspective changes. We are investigating the idea of using our texture and shape-based algorithms to track scene text events.

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. International Conference on Pattern Recognition*, volume 3, pages 831–834, 2000.

[3] 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.

[4] 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.

[5] 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.

[6] 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.

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

[8] 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.

[9] R. Lienhart and F. Stuber. Automatic Text Recognition for Video Indexing. In *Proceedings of the ACM International Multimedia Conference & Exhibition*, pages 11-20, 1996.

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

[11] N. Merhav and V. Bhaskaran. Fast algorithms for dct-domain image down-sampling and for inverse motion compensation. *IEEE Transactions on Circuits and Systems for Video Technology*, 7:468–476, 1997.

[12] T. Sato, T. Kanade, E. Hughes, and M. 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.

[13] J. 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.

[14] V. Wu, R. Manmatha, and E. Riseman. Finding Text in Images. In *Second ACM International Conference on Digital Libraries*, 1997.

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