Automated system for text detection in individual video images

Yingzi Du
Chein-I Chang
University of Maryland Baltimore County
Remote Sensing Signal and Image Processing Laboratory
Department of Computer Science and Electrical Engineering
Baltimore, Maryland 21250
E-mail: {eliza_du, cchang}@umbc.edu

Paul D. Thouin
Department of Defense
Fort Meade, Maryland 20755

Abstract. *Text detection in video images is a challenging research problem because of the poor spatial resolution and complex background, which may contain a variety of colors. An automated system for text detection in video images is presented. It makes use of four modules to implement a series of processes to extract text regions from video images. The first module, called the multistage pulse code modulation (MPCM) module, is used to locate potential text regions in color video images. It converts a video image to a coded image, with each pixel encoded by a priority code ranging from 7 down to 0 in accordance with its priority, and further produces a binary thresholded image, which segments potential text regions from the background. The second module, called the text region detection module, applies a sequence of spatial filters to remove noisy regions and eliminate regions that are unlikely to contain text. The third module, called the text box finding module, merges text regions and produces boxes that are likely to contain text. Finally, the fourth module, called the optical character recognition (OCR) module, eliminates the text boxes that produce no OCR output. An extensive set of experiments is conducted and demonstrates that the proposed system is effective in detecting text in a wide variety of video images.* © 2003 SPIE and IS&T. [DOI: 10.1117/1.1584050]

1 Introduction

Information retrieval from video images has become an increasingly important research area in recent years. The rapid growth of digitized video collections is due to the widespread use of digital cameras and video recorders combined with inexpensive disk storage technology. Textual information contained in video frames can provide one of the most useful keys for successful indexing and retrieval of information. Keyword searches for scene text of interest within video images can provide additional capabilities to the search engines. Most existing algorithms for text detection were developed to process binary document images and do not perform well on the more complex video images.

In past years, many different methods have been developed for text detection in color document images by taking advantage of document characteristics.[1–4] For example, simple edge-based detection filters such as the Sobel edge detector have been proposed to detect text based on the fact that the text is brighter than the image background.[1–2] Some methods also make an assumption that the text and background in a local region have relatively uniform gray levels so that the contrast information can be used to extract text.[3–4] Unfortunately, these techniques are generally not applicable to the complex background found in most video images. Most recently, neural networks have been offered as an alternative method for detecting text in videos.[5–6] However, training networks and adjusting parameters increase the complexity of the implementation. Although some robust text detection and extraction methods developed for multiple frames may be also applicable to single frames,[7–9] their accuracy and precision will be reduced when they are applied directly to single-frame video images since the information provided by the multiple frames used in these methods is not available for this case.

In video images, text characters generally have much lower resolution and dimmer intensity than document characters. In addition, video text characters frequently have various colors, sizes, styles, and orientations within the same image. Furthermore, video backgrounds are generally much more complex than those of document images. A combination of this complex background and a large variety of low-quality characters causes text detection algorithms designed for processing document images to perform poorly on video images. There are two general types of text found within video images, scene text [e.g., Fig. 4(a), Fig. 8(d), Fig. 9(c), and Figs. 10(a)–10(b)] and superimposed text [e.g., Figs. 8(a)–8(c) and 8(e), Figs. 9(a)–9(b) and 9(d), and Figs. 10(c)–10(d)]. While the former is part

Paper JEI 02057 received May 7, 2002; revised manuscript received Sep. 20, 2002, and Dec. 11, 2002; accepted for publication Jan. 3, 2003. This paper is a revision of a paper presented at the SPIE conference on Document Recognition and Retrieval IX, January 2002, San Jose, Calif. The paper presented there appears (unrefereed) in SPIE Proceedings Vol. 4670.
1017-9909/2003/$15.00 © 2003 SPIE and IS&T.

of the original scene, the latter is a separate object that is overlaid on the original scene. Since they frequently possess different characteristics, a technique designed to detect one type of text may not be applicable for detecting the other type.

This paper presents a system that can detect both scene text and superimposed text within video images. It is made up of four processing modules: a multistage pulse code modulation (MPCM) module, a text region detection (TRD) module, a text box finding (TBF) module, and an optical character recognition (OCR) module. The MPCM module is designed to convert a color video image into a gray-scale image and further produce a binary thresholded image, which locates potential text regions. The MPCM coding scheme used in this module was initially developed for progressive image transmission[10] where each image pixel is encoded by a priority code word in accordance with its priority for reconstruction. It was discovered by our experiments that such a priority code provides a good measure for locating potential text regions. Then these regions are processed in the TRD module via several spatial filters designed to remove noisy regions and regions that do not contain text. The design of these spatial filters takes advantage of how text appears in images. Since a text region that contains characters almost always appears as a box, the purpose of the TBF module is to rectangularize the text regions detected by the TRD module and produce text boxes. In doing so, several text regions may be merged to form a single text box. Finally, the OCR module is used to process each of the text boxes and to eliminate those boxes that produce no OCR results. It should be noted that this OCR module is not used for character recognition, but is simply used to eliminate as many falsely identified text regions as possible. As a result, the output of the OCR module is a simple binary decision to determine whether a text box contains text. Each of these four modules is fully automated and can be operated without human intervention. In addition, each module can be upgraded and improved individually and separately without affecting the other three modules, despite the fact that these four modules are processed in sequence.

In order to evaluate our proposed system, a database obtained from the Language and Media Processing Laboratory at the University of Maryland, College Park, is used for our experiments and analysis of performance. The results have shown that our system can achieve an 85% precision rate and a recall rate as high as 92%.

The remainder of this paper is organized as follows: Section 2 introduces the idea of MPCM and details its implementation. Section 3 describes the system's architecture, including the spatial filters used to detect candidate regions of text. Section 4 presents experimental results. Section 5 draws some conclusions.

2 Multistage Pulse Code Modulation

Multistage pulse code modulation was first developed for progressive image transmission,[10] which could reconstruct images progressively. It is a multistage version of a commonly used coding scheme, pulse code modulation (PCM), and quantizes inputs in multiple stages, one quantization level at each stage. It stretches PCM in a progressive fashion so that each quantization level is implemented one stage at a time. The idea is to design a code that can prioritize the quantization levels in accordance with the significance of a particular level in image quality. When an image is reconstructed progressively, these code words provide priorities of quantization levels to be used in image reconstruction. The capacity of this method for progressive edge detection demonstrated in Ref. 10 offers a unique advantage over classical edge detection in detecting text in video images because edge changes are generally progressive and slow owing to their low resolution and complicated background.

2.1 *Overview of MPCM*

Suppose that an MPCM module has M stages with a given set of stage levels $\{\Delta_k\}_{k=1}^{M}$ where Δ_k is the quantization level used in stage k. Let $x(n)$ be the gray-level value of the n-th sample pixel that is currently being visited by MPCM. The idea of MPCM is to decompose $x(n)$ into a set of binary-valued stage components, $\{x_k(n)\}_{k=1}^{M}$, so that $x(n)$ can be approximated by $x(n) \approx \Sigma_{k=1}^{M} x_k(n)\Delta_k$. In this case, $x(n)$ can be represented by an M-tuple $[x_1(n), x_2(n), \ldots, x_M(n)]$ with the approximation error given by $\varepsilon_M(n) = x(n) - \Sigma_{k=1}^{M} x_k(n)\Delta_k$ with a bit-rate $\log_2 M$. If $\varepsilon_M(n) = 0$, $x(n)$ can be perfectly reconstructed by $\Sigma_{k=1}^{M} x_k(n)\Delta_k$. If $\varepsilon_M(n) > 0$, it is necessary to encode both $\varepsilon_M(n)$ and $\Sigma_{k=1}^{M} x_k(n)\Delta_k$ to achieve perfect reconstruction of $x(n)$. This is similar to the way that we represent a real number $a \in [0,1]$ by a binary expansion $a = \Sigma_{k=1}^{M} a_k 2^{-k} + \varepsilon_M$ with M precision and an approximation error ε_M where each stage represents one precision and the k'th stage level is specified by $\Delta_k = 2^{-k}$ with the binary coefficients $a_k \in \{0,1\}$. So, in order to reconstruct $x(n)$, we begin to approximate $x(n)$ by $x_1(n)\Delta_1$, then $x_1(n)\Delta_1 + x_2(n)\Delta_2$, etc., until we reach the last stage M, in which case $x(n)$ is approximated by $\Sigma_{k=1}^{M} x_k(n)\Delta_k$. Such a progressive approximation is carried by priority code words assigned to image pixels.

More specifically, in MPCM, the image pixels to be used for reconstruction in the first stage are those with the highest-priority code word, $c_1(n)$, assigned by $M-1$; they are followed by those pixels with the second highest-priority code word, $c_2(n)$, assigned by $M-2$, etc., until it reaches the last stage M where the pixels with the least-priority code word, $c_M(n)$, assigned by 0 are used to complete the reconstruction. This $\log_2 M$-bit priority coding is similar to so-called bit-plane coding which also prioritizes bit planes according to the significance of bits from the most significant to the least significant.

A key difference between bit-plane coding and MPCM is that the bit-plane coding does not use the correlation between two bit planes, whereas MPCM is a predictive coding scheme that takes advantage of previous higher priority code words to reduce reconstruction errors. In doing so, the MPCM module uses two types of predictors, referred to as an interpixel predictor, $\hat{p}$, and interstage quantizer, Q_k, to improve reconstruction. Let $\hat{x}_k(n)$ be the predicted stage component in stage k. The interpixel predictor $\hat{p}$ predicts the gray-level value of the current n-th sample pixel, $x(n)$, from the gray-level values of previous sample

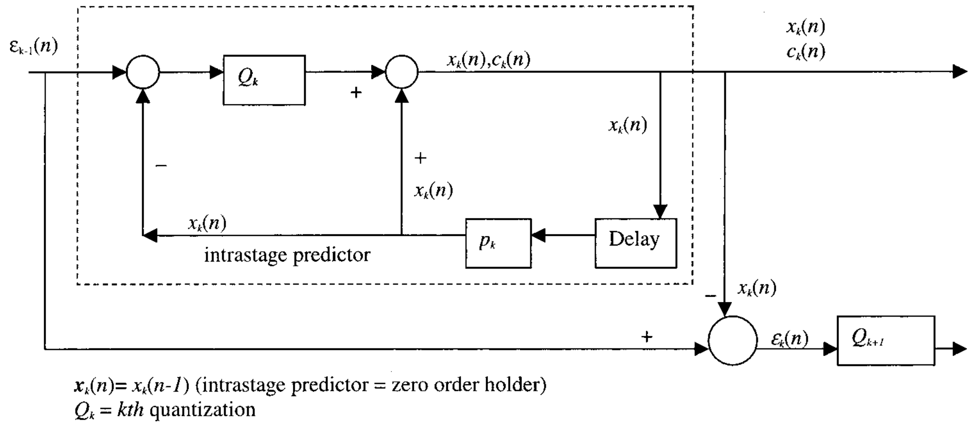


Fig. 1 Implementation of *k*-th stage MPCM.

pixels, $x(j)$ with $j<n$. In MPCM, only the immediate past sample pixel is used for $\hat{p}$, i.e., $\hat{x}(n)=\hat{p}[x(n-1)]$. The interstage quantizer, $Q_k[\varepsilon_{k-1}(n)]=\hat{x}_k(n)$, predicts the k'th stage component from $\hat{x}_k(n)$ based on $\varepsilon_k(n)=\varepsilon_{k-1}(n)-\Delta_k$ with $\varepsilon_0(n)=x(n)-\hat{x}(n)$. The key element of the MPCM module is the priority code $c(n)$ specially designed for $x(n)$ to store necessary information stage by stage to reconstruct $x(n)$ progressively, which can be described as follows:

At the kth stage of the MPCM, the kth interstage quantizer Q_k has two quantization levels, 0 and Δ_k, and three quantization intervals, $(-\infty, 0)$, $[0,\Delta_k)$, and $[\Delta_k,\infty)$, to predict $\hat{x}_k(n)$. It behaves like a soft limiter, that is,

$$Q_k[\varepsilon_{k-1}(n)]=\begin{cases} 0; \text{ if } \varepsilon_{k-1}(n)\in(-\infty,0) \\ \varepsilon_{k-1}(n); \text{ if } \varepsilon_{k-1}(n)\in[0,\Delta_k) \\ \Delta_k; \text{ if } \varepsilon_{k-1}(n)\in[\Delta_k,\infty) \end{cases}. \tag{1}$$

The details of the k-stage MPCM implementation are described the following section, and a block diagram of the encoding procedure of this 3-bit MPCM is depicted in Fig. 1.

2.2 *MPCM Encoding Process*

Assume that $x(n)$ is the n-th data sample pixel currently being visited. Let $\hat{x}(n)=\hat{p}[x(n-1)]$ and $\varepsilon_0(n)=x(n)-\hat{x}(n)$ be the initial prediction error at the initial stage resulting from the reconstruction of the previous sample pixel $x(n-1)$ via the predictor $\hat{p}$. Then for each stage k, $1\leqslant k\leqslant M$ we implement the following three-step procedure to produce a priority code $c(n)$ for $x(n)$.

MPCM encoding algorithm

Step 1:

If $\varepsilon_{k-1}(n)\geqslant\Delta_k$, then

(in this case, the input to the k-th stage quantizer Q_k exceeds the upper limit Δ_k)

$\hat{x}_k(n)=Q_k[\varepsilon_{k-1}(n)]=\Delta_k$

$\hat{x}_j(n)=Q_j[\varepsilon_{k-1}(n)]=0$ for all

$k<j\leqslant M$ and $c(n)=M-k$

(i.e., interstage interpolation for stages higher than k)

$\varepsilon_M(n)=\varepsilon_{k-1}(n)-\hat{x}_k(n)$

(i.e., prediction error of the n-th sample at stage k) and go to the next sample $x(n+1)$.
Step 2:

If $\varepsilon_{k-1}(n)<0$, then

(in this case, the input to the k-th stage quantizer Q_k falls below the lower limit 0)

$\hat{x}_k(n)=Q_k[\varepsilon_{k-1}(n)]=0$

$\hat{x}_j(n)=Q_j[\varepsilon_{k-1}(n)]=\Delta_j$ for all $k<j\leqslant M$
and $c(n)=M-k$

(i.e., interstage interpolation for stages higher than k)

$$\varepsilon_M(n)=\varepsilon_{k-1}(n)-\sum_{j=k+1}^{M}\hat{x}_k(j)=\varepsilon_{k-1}(n)-\sum_{j=k+1}^{M}\Delta_j$$

(i.e., prediction error of the n-th sample at stage k) and go to the next sample $x(n+1)$.
Step 3:

If $0\leqslant\varepsilon_{k-1}(n)<\Delta_k$, then

[in this case, the input lies between $[0,\Delta_k)$].

$\hat{x}_k(n)=Q_k[\varepsilon_{k-1}(n)]=\varepsilon_{k-1}(n)$ and $c(n)=0$.

$\varepsilon_k(n) = \varepsilon_{k-1}(n)$ (i.e., prediction error of the n-th sample at stage k)

$\varepsilon_k(n) = \varepsilon_{k-1}(n) - \Delta_k$

$k \leftarrow k+1$ and go to step 1.

Since the MPCM decoding algorithm is not required in our proposed text detection, there is no need to include its detailed implementation in this paper. So only the MPCM encoding algorithm is given here. For a complete implementation of the MPCM module, including encoding and decoding algorithms, see Refs. 10–11 for details. It should be also noted that the MPCM is executed pixel by pixel in a real-time manner and a very large scale integration (VLSI)-chip layout for real-time implementation of the MPCM was described in Ref. 11.

2.3 *Examples*

In order to illustrate how the MPCM module encodes an image, an example is provided in Fig. 2. It shows a stage-by-stage gray-level reconstruction of a one-dimensional gray-level value of pixels in one line of a video image plotted in Fig. 2(a). In Fig. 2(b), the first column consists of a sequence of progressive reconstructions of Fig. 2(a) from stage 1 to stage 8, whereas the second and third columns are the reconstruction errors and the plots of the priority code words generated. The eight-stage MPCM implemented for this example was one with $\Delta_k = 2^{8-k}$ and $\hat{x}(n) = \hat{p}[x(n-1)] = x(n-1)$. In order to shed more light on Fig. 2, in Table 1 we also provide a step-by-step procedure to encode the first ten sample pixels in Fig. 2 and tabulate the values of their priority code words and their associated reconstruction errors. The numerical values in Table 1 show how each pixel updates its gray-level value from the previous pixel to generate its priority code word where the initial condition $x(0)$ was set to 0.

3 System Architecture

In this section we propose the automatic text detection system diagrammed in Fig. 3 that consists of four main modules: the MPCM module, the text region detection module, the text box finding module, and the OCR module, where each module is responsible for a particular task in detection of text within video images. The MPCM module can encode a video image in both horizontal and vertical directions using the MPCM coding scheme and highlight suspected text within the image. However, since most text regions appear horizontal, only row-encoded MPCM images are used in this module. The TRD module takes advantage of the row-encoded MPCM images to remove areas unlikely to contain text and generates a low-resolution binarized image that segments suspected text regions. The TBF module creates rectangular boxes that surround the detected text regions. By means of such rectangularization it is very likely that several text regions may be merged into one text box. The final OCR module makes use of OCR results to eliminate falsely identified boxes that cannot be recognized as text characters by OCR. In what follows, each of the four modules is described in detail.

3.1 *MPCM Module*

The MPCM module converts a color video image into a gray-scale image via the HSI color model,[13] then encodes the resulting gray-scale image as a coded image with each pixel specified by a priority code word produced by MPCM. Prior to MPCM, a 3×3 low-pass window process is applied to the image to suppress noise. This is followed by an eight-stage MPCM as described in Sec. 2 with the k-th stage specified by $\Delta_k = 2^{8-k}$ where each image pixel will be assigned by a priority code word ranging from 0 to 7. The higher the code word number, the higher the priority. Since MPCM is a one-dimensional coding process, it can be carried out row by row in a one-dimensional fashion. As a result, a row-encoded MPCM image is generated by the MPCM module. The global mean of their priority code words is then calculated for the row-encoded MPCM image and used as a threshold value in the follow-up preprocessing module to segment potential regions that contain text. As an example, Figs. 4(a) and 4(b) show an original color video game show image and the row-encoded MPCM image, respectively. As we can see from Fig. 4(b), the row-encoded MPCM image tends to extract vertical line segments.

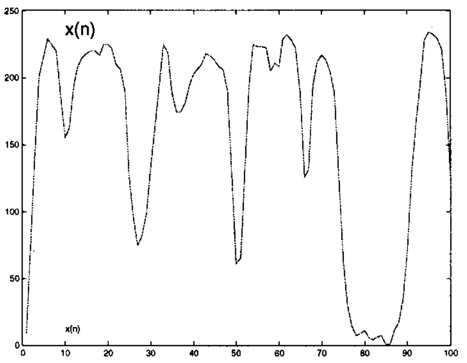
As indicated previously, the MPCM can detect text regions progressively by finding slow changes in edges instead of abrupt changes. Since the priority code yielded by MPCM detects edges progressively, the use of MPCM allows us to locate potential text regions in a slow-changing manner. This benefit cannot be gained by other existing edge-detection algorithms, which are primarily designed to detect something changing rapidly or abruptly. In order to demonstrate the merit of using our proposed MPCM, Fig. 4(c) shows an edge-detection map resulting from the Sobel edge detector that detected vertical changes in Fig. 4(a). As one can see, the progressive edge changing of "CAROLINE RHEA" was detected by the row-encoded MPCM image in Fig. 4(b) compared with the abrupt changes in "CAROLINE RHEA" detected by the Sobel edge detector. Such edge changes in a progressive manner provide valuable information in detecting potential text regions.

3.2 *Text Region Detection Module*

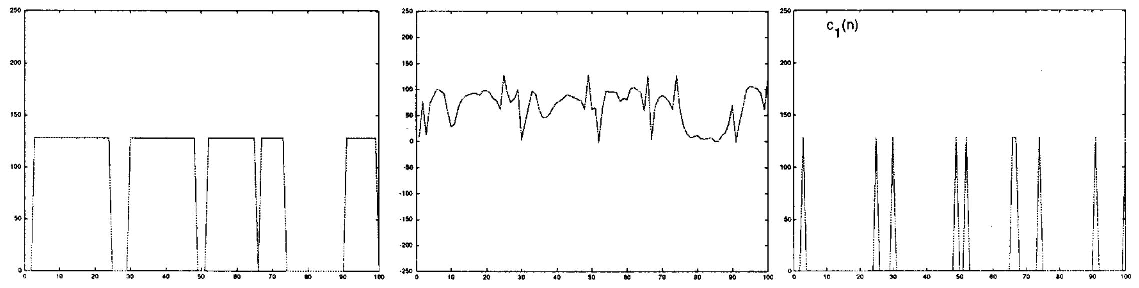
Upon completion of the MPCM module, the TRD module converts the row-encoded MPCM image to produce a single low-resolution binary image. Five filtering steps are included in this module.

1. *Thresholding*. This process converts the gray-scale row-encoded MPCM image into a binary image. It divides the row-encoded MPCM image into a set of nonoverlapping blocks. The block size is generally determined by the size of video images to be processed. The size can be defined by the smallest text block that a human being can recognize in the images. The size of the video image in Fig. 4(a) is 240×350 where the smallest recognizable text block is about 8×8.

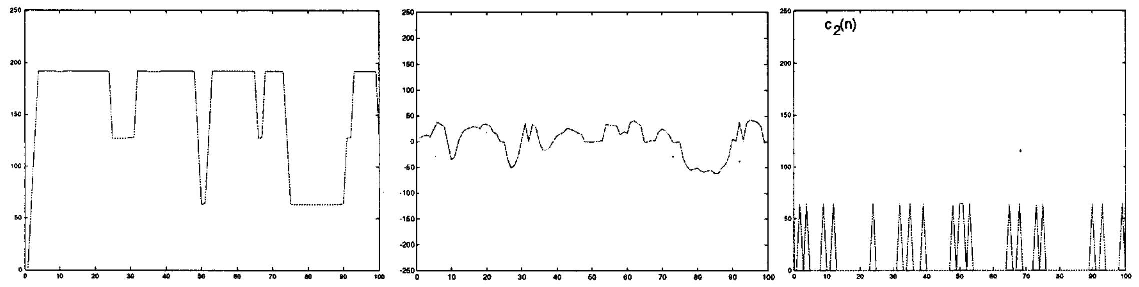
In order to properly threshold the row-coded MPCM blocks, we first calculate the global mean of the priority code words of the row-encoded MPCM image, denoted by μ_{row}. In analogy with the k-th interstage quantizer Q_k implemented in the MPCM module, we also make use of a soft limiter to bound the global mean μ_{row} from below and above by μ_{low} and μ_{upper}. The adjusted global



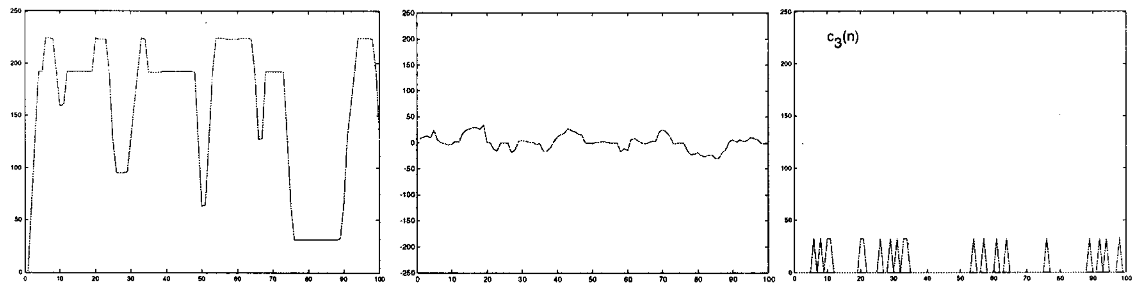
(a)



Stage 1



Stage 2



Stage 3

Fig. 2 Eight-stage reconstruction using a 3-bit MPCM. (a) A plot of 1-D gray-level values of pixels in a line of a video image. (b) Left to right: first column, progressive reconstruction of (a) from stage 1 to stage 8; second column, progressive reconstruction errors resulting from stage 1 to stage 8; third column; priority code words produced by each stage from 0 to 7.

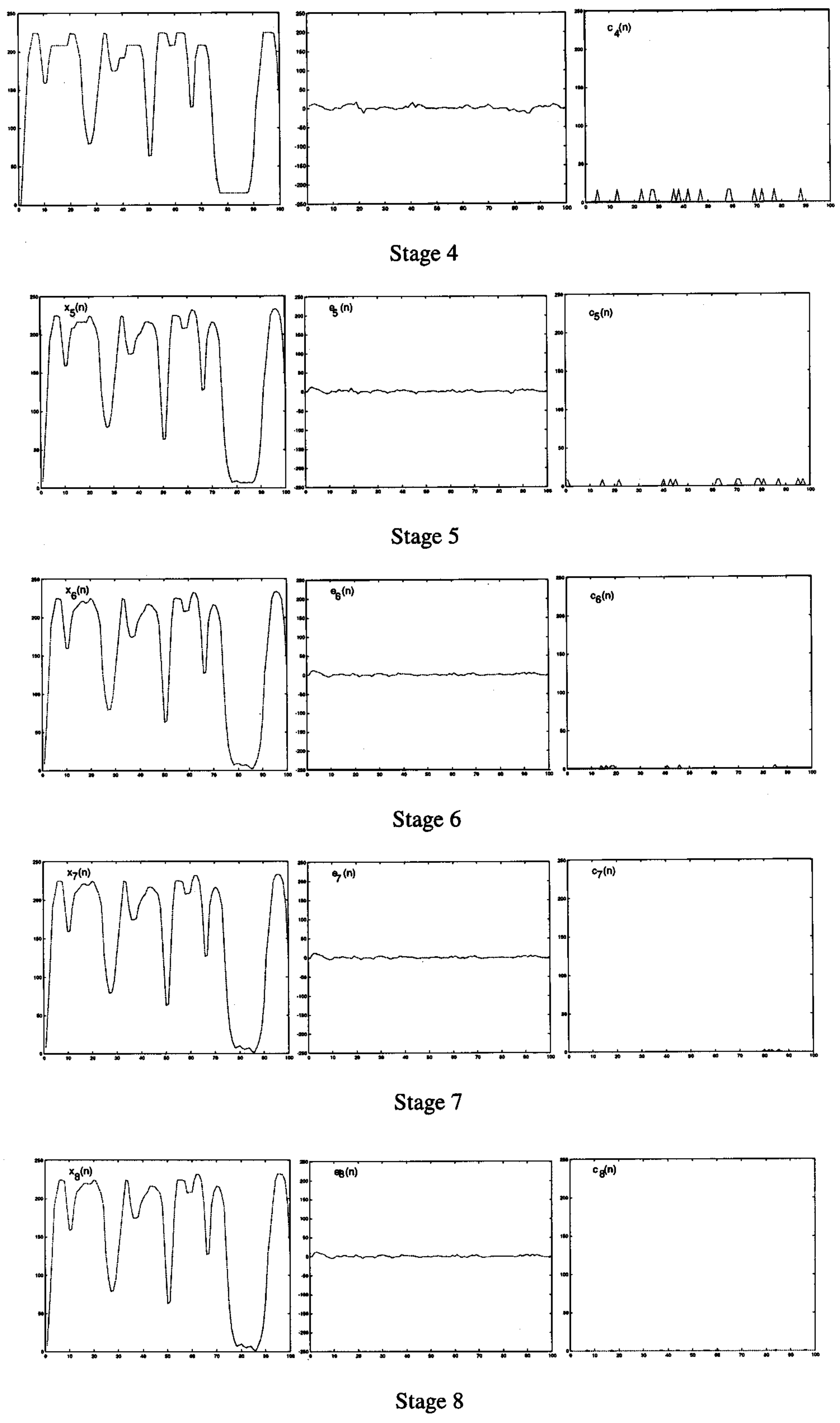


Fig. 2 Continued.

Table 1 A step-by-step procedure to encode the first ten sample pixels in Fig. 2.

$$\hat{x}(n)=x_1(n)+x_2(n)+\cdots+x_8(n)$$
$$e(n)=x(n)-\hat{x}(n)$$

MPCM Encoding Algorithm

Input		Reconstruction		Stage Components								Priority Code Word
n	$x(n)$	$\hat{x}(n)$	$\varepsilon(n)$	$x_1(n)$ 128	$x_2(n)$ 64	$x_3(n)$ 32	$x_4(n)$ 16	$x_5(n)$ 8	$x_6(n)$ 4	$x_7(n)$ 2	$x_8(n)$ 1	$c(n)$
0	0	0	0	0	0	0	0	0	0	0	0	0
1	8	8	0	0	0	0	0	8	0	0	0	3
2	74	64	10	0	64	0	0	0	0	0	0	6
3	140	128	12	128	0	0	0	0	0	0	0	7
4	201	192	9	128	64	0	0	0	0	0	0	6
5	215	208	7	128	64	0	16	0	0	0	0	4
6	229	224	5	128	64	32	0	0	0	0	0	5
7	224	224	0	128	64	32	0	0	0	0	0	0
8	220	223	−3	128	64	0	16	8	4	2	1	5
9	186	191	−5	128	0	32	16	8	4	2	1	6
10	155	159	−4	128	0	0	16	8	4	2	1	5

mean $\hat{\mu}_{\text{row}}$ is obtained by the following soft limiter:

$$\hat{\mu}_{\text{row}}=\begin{cases}\mu_{\text{low}}\,;\mu_{\text{row}}<\mu_{\text{low}}\\ \mu_{\text{row}}\,;\mu_{\text{low}}\leq\mu_{\text{row}}\leq\mu_{\text{upper}}\,.\\ \mu_{\text{upper}}\,;\mu_{\text{row}}>\mu_{\text{upper}}\end{cases}\tag{2}$$

For the experiments described in this paper, we have chosen $u_{\text{low}}=2.1$ and $u_{\text{upper}}=3$. Finally, the local mean of each block is calculated and compared against its corresponding adjusted global mean. If the local mean of a block in the row-encoded MPCM image is greater than a parameter λ times its respective adjusted global mean, $\hat{\mu}_{\text{row}}$, the block will be mapped to a pixel assigned by 1 and 0. As a result of such thresholding, a binary image is generated with a size that is only 1/64 of the original image size. The parameter λ was chosen empirically. For our experiments, we chose a value of $\lambda=1.2$. Figure 5(a) shows the binary image of Fig. 4(a) resulting from thresholding the row-encoded MPCM image in Fig. 4(b).

2. *Elimination of isolated blocks*. The following filter process is designed to eliminate isolated image blocks, which are unlikely to contain text. The filter is specified by

$$w_1=\begin{bmatrix}1&1&1\\1&0&1\\1&1&1\end{bmatrix}\tag{3}$$

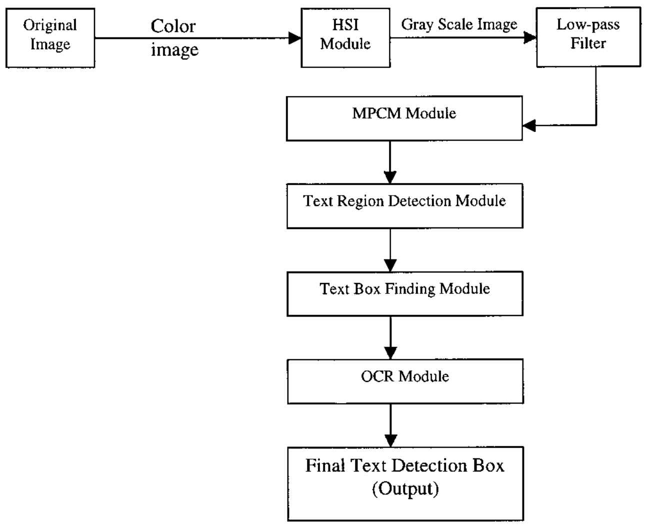


Fig. 3 Architecture of the proposed system.

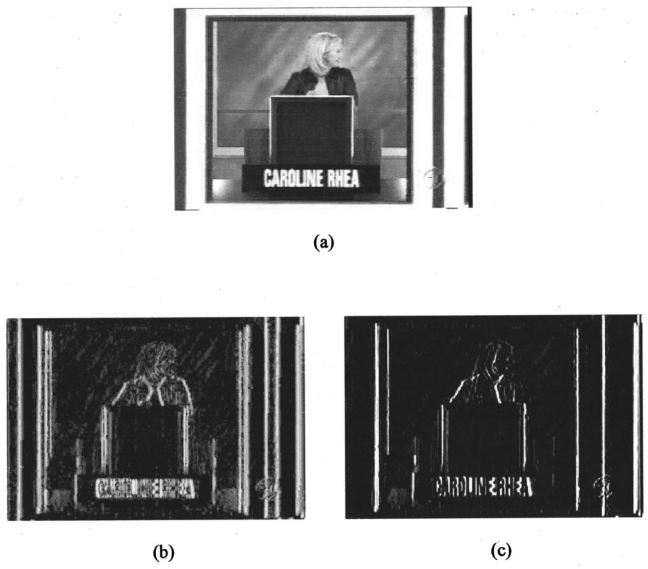


Fig. 4 Comparison of the row-encoded MPCM image with the vertical edges detected by a Sobel edge detector. (a) Original color video image. (b) Row-encoded MPCM image. (c) Vertical edges detected by a Sobel edge detector.

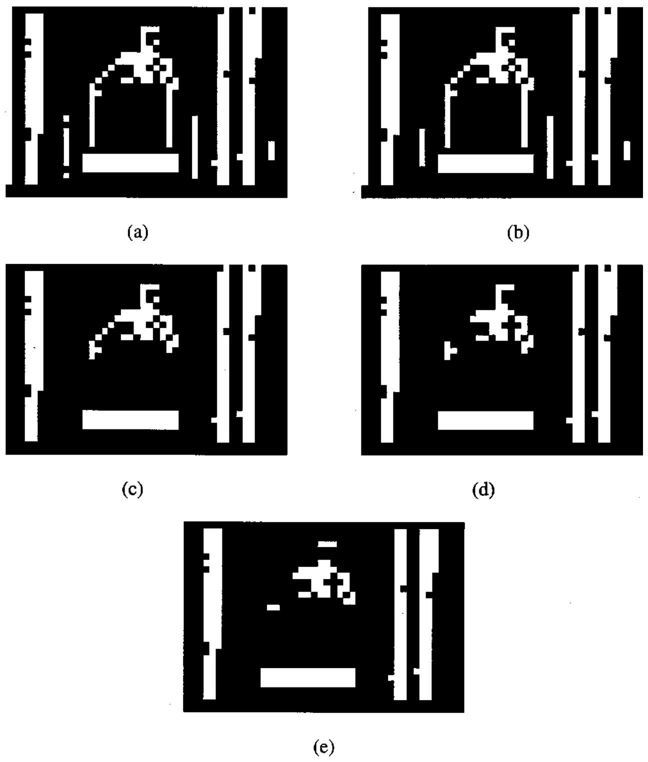


Fig. 5 Step-by-step implementation of the TRD module. (a) Result of thresholding the row-encoded MPCM image. (b) Elimination of two isolated blocks. (c) Elimination of six pixels aligned vertically with more than three blocks. (d) Elimination of five x-connected blocks. (e) Elimination of seven blocks with only one vertical connection.

and allows us to remove blocks that are isolated and do not have connected neighboring blocks. Figure 5(b) shows that two blocks near the lower-left side in Fig. 5(a) have been eliminated.

3. *Elimination of long vertical blocks*. Since text generally occurs horizontally, the following spatial filter removes vertical blocks that are more than three blocks long:

$$w_2 = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \tag{4}$$

This allows us to remove blocks that do not have adjacent blocks but are connected vertically by more than three blocks. The three-block length is also an empirical choice, but works well in our experiments. In Fig. 5(c) six such blocks in Fig. 5(b) have been removed.

4. *Elimination of diagonally connected blocks*. Blocks that are connected diagonally (i.e., 45, 135, 225, and 315 deg) are referred to as x-connected blocks. An x-connected block that does not have any "+" connected block (i.e., 0, 90, 180, and 270 deg) is not very likely to have text content. Therefore it should be removed. The following filter is designed for this purpose:

$$w_3 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}. \tag{5}$$

Figure 5(d) shows that five x-connected blocks in Fig. 5(c) have been removed.

5. *Elimination of weakly connected vertical blocks*. The following filters are designed to remove blocks that have only one vertical connection or are completely isolated:

$$w_4 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}, \quad w_5 = [1 \quad 0 \quad 1].$$

Figure 5(e) shows that seven such blocks in Fig. 5(d) have been removed.

3.3 *Text Box Finding (TBF) Module*

After completion of the TRD module, the text regions obtained can be considered to be candidate text regions, which are likely to contain text characters. Because a text box always occurs as a rectangle, this module expands a detected text region by filling in missing blocks to form a rectangular box. Figure 6(a) shows the result of filling eight missing blocks in the seven large text regions detected in Fig. 5(e). Since the image in Fig. 6(a) was shrunk from the original image by 1/64, we need to expand it back to the original size to identify original text regions as shown in Fig. 6(b). Because the expanded text boxes in Fig. 6(b) may create blocky effects, each expanded text box in the original image is further smoothed by including four pixels above, below, to the right, and to the left of each pixel in the text box. Figure 6(c) shows the resulting six rectangular boxes where the rightmost single block in Fig. 6(c) was a result of overlapping two expanded blocks produced by the two separate vertical blocks on the right in Fig. 6(b). In this case, this single block is counted as two separate blocks because each block was expanded separately before the blocks were connected in Fig. 6(c). The seven segments expanded by seven blocks in Fig. 6(b) are further matched with the original image in Fig. 4(a) to locate and identify their corresponding text boxes shown in Fig. 6(d). Finally, these seven text boxes in Fig. 6(d) are extracted directly from the original image and labeled in Fig. 6(e) for the OCR module.

3.4 *OCR Module*

The OCR module is included as a final process in order to eliminate many falsely identified text boxes. On most occasions the segmented image blocks are too small to be recognized by OCR. In this case, an interpolation is necessary prior to OCR processing. In our experiments, a spline interpolation was used. For each text box obtained by the TBF module, a cubic-splined interpolation by an expansion factor of 4 was performed to improve its resolution. However, a superior text enhancement technique such as the BSA algorithm[12–13] can also be used to achieve better results at the expense of additional computational complexity. The expanded-resolution text box is then input to a commercial character recognition engine to determine if any

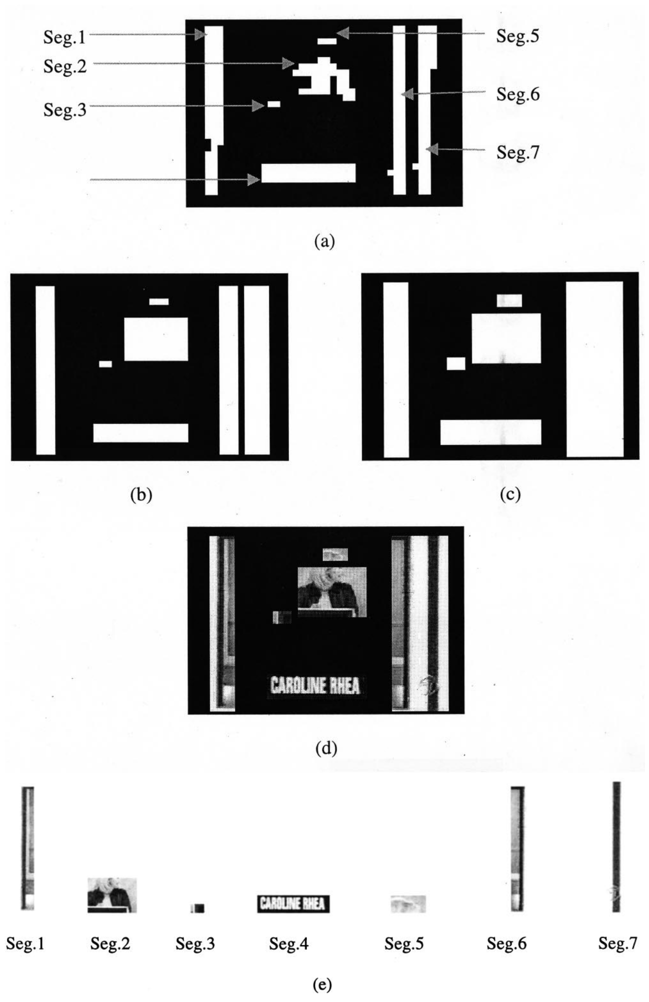


Fig. 6 Step-by-step implementation of the TBF module. (a) Text boxes obtained by rectangularizing text regions in Fig. 5(e). (b) Text boxes in (a) expanded back to the original image. (c) Seven text boxes in (b) smoothed by including four pixels above, below, to the right, and to the left of each pixel in the boxes to yield seven text boxes. (d) Seven segments in the original image that match the six text boxes identified in (c). (e) Seven segmented text boxes extracted from (d).

character within the text box can be recognized as a text character. It should be noted that commercial OCR engines are generally designed to recognize text in high-resolution, clean document images, and usually perform poorly on video images. The one used in our experiments was OmniPage Pro Version 10.0, which is the commercialized software produced by Scan Soft. It can recognize twelve different Latin-alphabet languages, including English, French, German, Italian, and Spanish. Unfortunately, it cannot recognize such languages as Chinese, Japanese, or Arabic. When non-Latin characters are input to a Latin OCR engine, the output typically contains gibberish Latin characters. The fact that any characters at all are recognized is used by the OCR module for detection. By taking advantage of this, we can find the blocks that contain characters in Chinese, Japanese, Arabic, or other languages that are not recognized by the software. The text boxes that produce no OCR results will be eliminated. After applying the

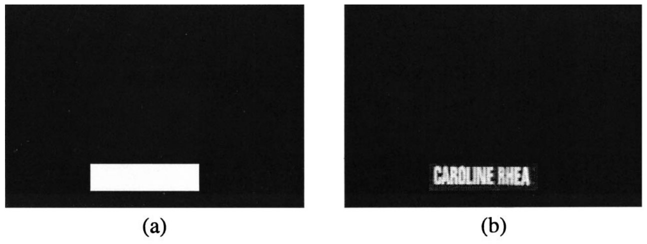


Fig. 7 Step-by-step implementation of the OCR module. (a) the text box recognized by the OCR. (b) the text within the box in Fig. 6(d) recognized by the OCR.

spline interpolation and OCR to the seven segments in Fig. 6(e), only one text box (Seg. 4) was recognized as a text box and is shown in Fig. 7(a). The other six segmented text boxes (Segs. 1 to 3 and Segs. 5 to 7) were thrown away because the OCR did not produce any output. Figure 7(b) shows the text within Seg. 4, which is ‘‘CAROLINE RHEA’’ produced by the OCR. If text detection is not used and the original image is expanded using cubic spline interpolation and then input to the OCR, the OCR will recognize it as an image containing no text and nothing will be output.

In summary, Figs. 4–7 show the step-by-step text detection process for a video frame. Thresholding of the row-encoded MPCM image results in the eleven connected regions shown in Fig. 5(a). Elimination of isolated blocks results in the removal of two blocks, leaving the nine blocks shown in Fig. 5(b). Long vertical regions are eliminated in Fig. 5(c), resulting in six components being eliminated. Elimination of the five x-connected boxes results in one connected region being split into two separate regions as shown in Fig. 5(d). The removal of seven weakly connected vertical blocks produces Fig. 5(e), which contains seven candidate text regions. These regions are converted to text boxes in Figs. 6(a) to 6(d) and further processed by the OCR module, where all but one box is eliminated. This text box is shown in Fig. 7. Another illustrative step-by-step experiment using a television commercial was described in Ref. 14.

4 Experiments

In order to evaluate the performance of our proposed text detection system, an extensive set of video images was used for experiments. These video images were obtained from the Language and Media Processing Laboratory at the University of Maryland, College Park. They were captured from commercial television broadcasts and contained ground truth, which marks and lists the bounding boxes for text regions within the video images. In addition to the bounding box markings, a subjective ‘‘image quality’’ score ranging from 1 (very poor) to 5 (excellent) was included for each region as well. A quality rating of 5 was given to a clear superimposed text with a simple background. A quality score of 4 was given to text regions with clear superimposed text and a slightly complex background. Quality scores of 3 were assigned to text regions containing complex superimposed text with a noisy background. Images with scene text and a complex background were rated as a quality score of 2. Text regions containing significantly blurred or distorted poor-quality text were assigned the lowest quality rating of 1. Figure 8 shows the results produced by our system for images selected from each of the five categories. For the experiments described in this section, only images with a quality score of 2 or higher were used. A set of 1170 video images was used to evaluate text detection performance in which a total of 4512 text regions were analyzed.

Two criteria that are commonly used to evaluate performance in information retrieval are the precision and recall rates, which are given in Ref. 16 by

$$\text{precision rate} = \frac{\text{No. of correctly detected text boxes}}{\text{No. of detected text boxes}} \tag{6}$$

and

$$\text{recall rate} = \frac{\text{No. of correctly detected text boxes}}{\text{No. of text boxes}}. \tag{7}$$

From a text detection viewpoint, the precision rate measures the percentage of the correctly detected text boxes within each video image as opposed to detected boxes, while the recall rate measures the percentage of correctly detected text boxes that actually contain text. Our system correctly detected 4150 of 4512 text regions for a precision rate of 85%. In addition, our system also detected 732 suspected text regions that were not actually text, which resulted in a recall rate of 92%. According to the ground truth provided, the bounding regions used in our experiments are horizontal and vertical. For slanted text, we extend the box that covers the slanted box horizontally and vertically, in which case the extended box may be larger than the original slanted text. In our database, there are only 43 slanted text regions out of 4512 text regions, which is less than 1% of the total cases. If the detected region is larger than the ground truth within an eight-pixel margin, we declare that our precision rate is 100% and recall is 100%. If the detected region is smaller than the ground truth within margins less than four pixels, we also declare that our precision rate and recall rate is 100%. Other than these two cases, the precision rate and recall rate are calculated according to the criteria used in Ref. 15 since the database used in our experiments was the same are used in those experiments.

The experimental results demonstrate the effectiveness of our text detection system on a diverse set of video images. Our proposed system seems to be language independent. In order to demonstrate this fact, four additional television news videos were evaluated. Figure 9(a) is the well-known image of Osama bin Laden with two Arabic texts appearing on the top right and left, which were successfully extracted. The size of this video image is 120×160 pixels and the smallest recognizable text block is about 4×4 pixels. Figure 9(b) is a Chinese television news video image in which four Chinese characters were correctly detected and identified. Figure 9(c) is a Russian television news video image in which the Russian text to the anchorman's right shoulder was extracted. Figure 9(d) shows a Japanese television news image in which Japanese characters were also successfully segmented from the image.

Finally, we compared our method with the method proposed by Peitikainen and Okun in Ref. 1 since it is the most

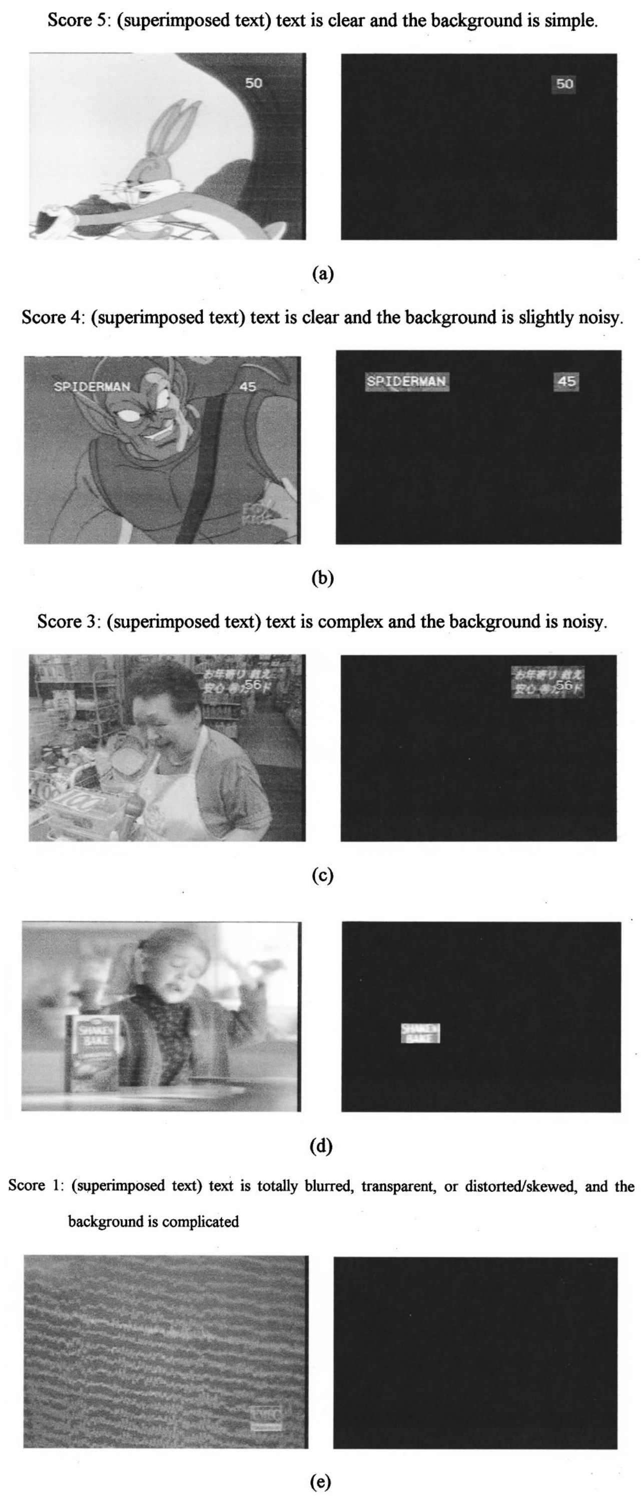
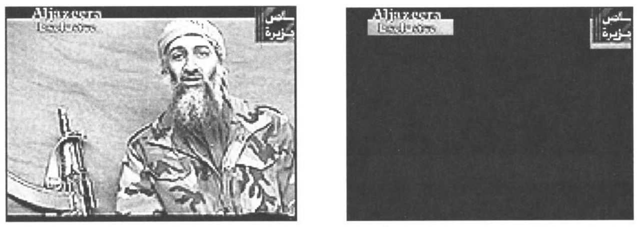
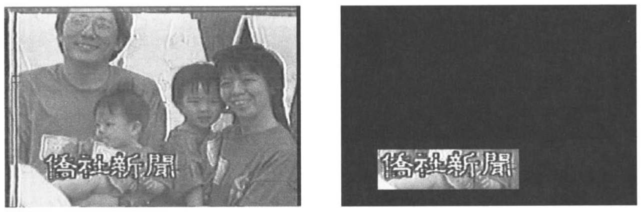

Score 5: (superimposed text) text is clear and the background is simple.
50
50
(a)
Score 4: (superimposed text) text is clear and the background is slightly noisy.
SPIDERMAN
45
SPIDERMAN
45
(b)
Score 3: (superimposed text) text is complex and the background is noisy.
(c)
SHAKE'N BAKE
(d)
Score 1: (superimposed text) text is totally blurred, transparent, or distorted/skewed, and the background is complicated
(e)

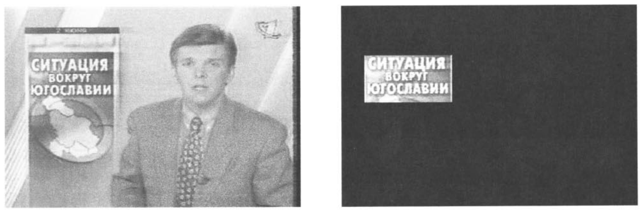
Fig. 8 Examples of image quality scores ranging from 5 to 1. The original video images are shown in the left column and the output text boxes with bounded images produced by our system are in the right column. (a) Score 5 (superimposed text): text is clear and the background is simple. (b) Score 4 (superimposed text): text is clear and the background is slightly noisy. (c) Score 3 (superimposed text): text is complex and the background is noisy. (d) Score 2 (scene text): text is blurred and small and the background is noisy. (e) Score 1 (superimposed text): text is totally blurred, transparent, or distorted and skewed, and the background is complicated.



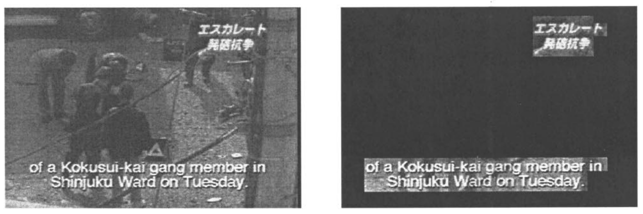
(a) Osama Bin Laden video image (superimposed text)



(b) Chinese TV news (superimposed text)



(c) Russian TV news (scene text)

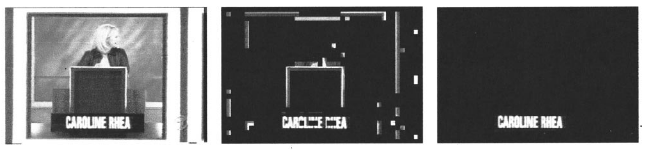


(d) Japanese TV news (superimposed text)

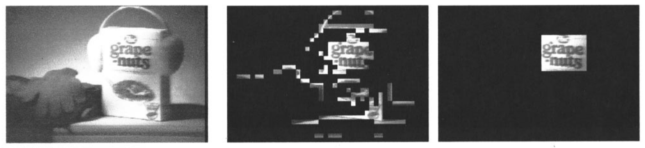
Fig. 9 Four video images with different text languages. (a) Osama bin Laden video image (superimposed text). (b) Chinese television news (superimposed text). (c) Russian television news (scene text). (d) Japanese television news (superimposed text).

recent result published in the open literature among Refs. 1–4 and also uses edge detection for text detection. The methods in Refs. 5–9 were not selected because the methods in Refs. 7–9 were developed from multiple video frames and the methods in Refs. 5–6 were developed using neural networks and it is difficult to repeat the results. Figure 10 shows the results produced by Peitikainen and Okun's method and our proposed system. The images in the first column are the original images; the images in the second column were produced by Peitikainen and Okun's method and those in the third column by our method. As can be seen, our proposed method performed significantly better than Peitikainen and Okun's method. It should be noted that these examples present only a small set of our experiments. Many more comparative experiments were also conducted but are not included in this paper.

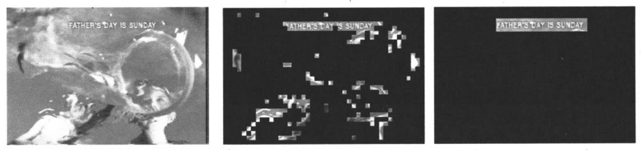
To conclude this section, one remark is worthwhile. According to the database provided by the University of Maryland, College Park, only two examples out of 1200



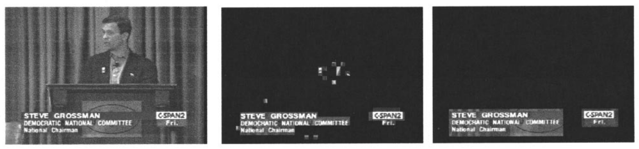
(a) Scene text



(b) Scene text



(c) Superimposed text



(d) Superimposed text

Fig. 10 Comparison of our method with Peitikainen and Okun's method. (a) and (b) Scene text; (c) and (d) superimposed text. The original video image is in the first column, the results produced by Peitikainen and Okun's method are in the second column, and results produced by our method are shown in the third column.

video images contain vertical text regions. Since vertical text does not occur frequently, our proposed system is primarily developed to detect horizontal text regions in video images. However, our system can be easily modified to detect vertical text regions by replacing the row-encoded MPCM images with the column-encoded MPCM images in the MPCM module. As an example, Fig. 11(a) shows a video image in which only five vertical scene text regions, Volume 1, Volume 2, Volume 3, Volume 4, and Volume 5 are visible. Figure 11(b) shows the results produced by Peitikainen and Okun's method and Fig. 11(c) shows the results obtained with our system using the column-encoded MPCM. As can be seen, Peitikainen and Okun's method did poorly in detecting these five vertical text characters compared with our column-encoded MPCM-based method, which was able to extract vertical text regions effectively,

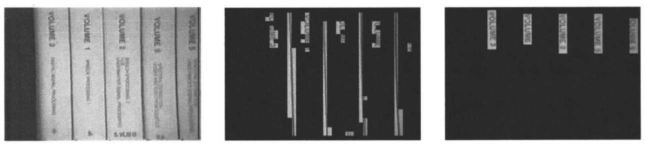


Fig. 11 (a) A video image with five vertical text regions. (b) Results produced by Peitikainen and Okun's method. (c) Results produced using our column-encoded MPCM image.

missing only two text characters, ''1'' in Volume 1 and ''V'' in Volume 5, because ''1'' was too small and ''V'' was too blurred. This experiment also demonstrates another merit of the use of MPCM in that it can be adapted to detect horizontal, slanted, or vertical text regions.

5 Conclusions

This paper describes an automated system for text detection in color video images. The system consists of four modules: the MPCM module, the text region detection module, the text box finding module, and the OCR module, each of which is new and designed to perform a specific task, particularly the MPCM module. It not only allows us to convert a color video image into a gray-scale image but also to locate regions that may contain text. This is critical to success in text enhancement and recognition. The MPCM module utilizes a priority code to rank each pixel based on its significance during progressive image transmission. It turns out that the priority associated with each pixel can be also used as an indication of a possible text character pixel. Through an extensive study of experiments, this MPCM module successfully demonstrated the capability to detect text regions in a large collection of video images. One advantage of our proposed system is that each module can be upgraded and improved separately and individually without affecting the performance of other modules. Although no restoration is discussed in this paper, the system can be expanded by including a text restoration module to improve text recognition in the text boxes detected.[17]

Acknowledgment

The authors would like to thank the U.S. Department of Defense for supporting their work through contract MDA-904-00-C2120. The authors would also like to thank Dr. D. Doermann of the Language and Media Processing Laboratory at the University of Maryland, College Park, for providing the database used for these experiments.

References

1. M. Peitikainen and O. Okun, ''Edge-based method for text detection from complex document images,'' in *Proc. Sixth International Conference on Document Analysis and Recognition*, pp. 286–291 (2001).
2. M. Kamel and A. Zhao, ''Extraction of binary character/graphics images from grayscale document images,'' *Comput. Vis. Graph. Image Process.* **55**(3), 203–217 (1993).
3. L. Agnihotri and N. Dimitrova, ''Text detection for video analysis,'' in *Proc. IEEE Workshop on Content-Based Access of Image and Video Libraries* (CBAIVL '99), pp. 109–113, Institute of Electrical and Electronics Engineers, New York (1999).
4. R. Lienhart and F. Stuber, ''Automatic text recognition in digital videos,'' in *Proc. ACM Multimedia Conf.*, pp. 11–20, Association for Computing Machinery, New York (1996).
5. H. Li, D. Doermann, and O. Kia, ''Automatic text detection and tracking in digital video,'' *IEEE Trans. Image Process.* **9**(1), 147–156 (2000).
6. 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 Workshop Neural Networks for Signal Processing X* **2**, pp. 634–641, Institute of Electrical and Electronics Engineers, New York (2000).
7. S. Antani, D. Crandall, and R. Kasturi, ''Robust extraction of text in video,'' in *Proc. 15th International Conference on Pattern Recognition* **1**, pp. 831–834 (2000).
8. 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*, pp. 618–620 (1998).
9. D. Crandall and R. Kasturi, ''Robust detection of stylized text events in digital video,'' in *Proc. Sixth International Conference on Document Analysis and Recognition*, pp. 865–869 (2001).
10. C.-I. Chang, Y. Cheng, J. Wang, M. L. G. Althouse, and M. L. Chang, ''Progressive edge extraction using multistage predictive coding,'' in *Proc. 1994 International Symposium on Speech, Image and Neural Networks*, pp. 57–60 (1994).
11. Y. Cheng, *Multistage Predictive Pulse Code Modulation* (MPCM), Department of Electrical Engineering, University of Maryland, Baltimore County, MD (1993).
12. P. Thouin and C.-I. Chang, ''An automated system for restoration of low-resolution document and text images,'' *J. Electron. Imaging* **10**(2), 535–547 (2001).
13. Y. Du, P. D. Thouin, and C.-I. Chang, ''Low resolution expansion of color text image using HSI approach,'' in *5th World Multiconference on Systems, Cybernetics and Informatics (SCI 2001) and 7th International Conference on Information Systems Analysis and Synthesis (ISAS 2001)*, pp. 295–300 (2001).
14. Y. Du, P. D. Thouin, and C.-I. Chang, ''A multistage predictive coding approach to unsupervised text detection in video images,'' *IS&T/SPIE's 14th Int. Symp. on Electronic Imaging: Science and Technology*, *Proc. SPIE* **4670**, (2002).
15. H. Li, ''Automatic processing and analysis of text in digital video,'' PhD Dissertation, Department of Computer Science, University of Maryland, College Park, MD (2000).
16. http://cslu.cse.ogi.edu/HLTsurvey/ch13node4.html
17. Y. Du, ''Text detection and restoration for color video images,'' PhD dissertation, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD (2003).

Yingzi Du received her BS and MS degrees in electrical engineering from Beijing University of Posts and Telecommunications in 1996 and 1999, respectively. She received her PhD from the University of Maryland, Baltimore County in 2003. Her research interests include documentation and text analysis, information retrieval and multispectral/hyperspectral image processing, and medical imaging. Dr. Du is a member of SPIE and IEEE and also a member of Phi Kappa Phi and Tau Beta Pi honor societies.

Chein-I Chang received his BS, MS, and MA degrees from Soochow University, the Institute of Mathematics at National Tsing Hua University, Hsinchu, Taiwan, and the State University of New York at Stony Brook, respectively, all in mathematics; he received MS and MSEE degrees from the University of Illinois at Urbana-Champaign, respectively, and a PhD in electrical engineering from the University of Maryland, College Park, in 1987. He was a visiting assistant professor from January to August 1987, assistant professor from 1987 to 1993, associate professor from 1993 to 2001 and since 2001 has been a professor in the Department of Computer Science and Electrical Engineering at the University of Maryland, Baltimore Country. Dr. Chang was a visiting specialist at the Institute of Information Engineering, National Cheng Kung University, Tainan, Taiwan, from 1994 to 1995. He has a patent for automatic pattern recognition and several pending patents on image processing techniques for hyperspectral imaging and detection of microcalcifications. He is currently the associate editor in the area of hyperspectral signal processing for the *IEEE Transactions on Geoscience and Remote Sensing* and is also on the editorial board of the *Journal of High Speed Networks.* In addition, Dr. Chang was the guest editor of a special issue of the lattest journal on telemedicine and applications. His research interests include automatic target recognition, multispectral and hyperspectral image processing, medical imaging, documentation and text analysis, information theory and coding, signal detection and estimation, and neural networks. Dr. Chang is a SPIE Fellow and a senior member of IEEE; he is also a member of Phi Kappa Phi and Eta Kappa Nu.

Paul D. Thouin received his BS degree in electrical engineering from the University of Michigan, Ann Arbor, in 1987. In 1993, he obtained his MSEE degree from George Washington University in Washington DC. He received his PhD in electrical engineering from the University of Maryland, Baltimore County in 2000. Dr. Thouin has been employed by the U.S. Department of Defense since 1987, where he is a senior engineer currently assigned to the Image Research Branch in the Research and Development Group. His research interests include image enhancement, statistical modeling, document analysis, pattern recognition, and multiframe video processing. Dr. Thouin is a SPIE member; a senior member of IEEE, and a member of Phi Kappa Phi.