LAMP-TR-028
CAR-TR-900
CS-TR-3962

December 1998

Automatic Text Detection and Tracking in Digital Video

Huiping Li
David Doermann
Omid Kia

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **DEC 1998**	2. REPORT TYPE	3. DATES COVERED **00-12-1998 to 00-12-1998**
4. TITLE AND SUBTITLE **Automatic Text Detection and Tracking in Digital Video**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Language and Media Processing Laboratory,Institute for Advanced Computer Studies,University of Maryland,College Park,MD,20742-3275**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR’S ACRONYM(S)
		11. SPONSOR/MONITOR’S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES **The original document contains color images.**		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**40**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

Automatic Text Detection and Tracking in Digital Video

[1]Huiping Li, [1]David Doermann and [2]Omid Kia

[1]Language and Media Processing Laboratory
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742
(huiping,doermann)@cfar.umd.edu

[2]Mathematical and Computational Sciences Division
National Institute of Standards and Technology
Gaithersburg, MD 20899

Abstract

Text which either appears in a scene or is graphically added to video can provide an important supplemental source of index information as well as clues for decoding the video's structure and for classification. In this paper we present algorithms for detecting and tracking text components that appear within digital video frames. Our system implements a scale-space feature extractor that feeds an artificial neural processor to extract textual regions and track their movement over time. The extracted regions can then be used as input to an appropriate Optical Character Recognition system which produces indexible keywords.

Keywords: Text Detection, Text Tracking, Video Indexing, Digital Libraries, Neural Network, Wavelet

The support of this effort by the Department of Defense under contract MDA 9049-6C-1250 is gratefully acknowledged.

1 Introduction

The increasing availability of online digital imagery and video has rekindled interest in the problems of how to index multimedia information sources automatically and how to browse and manipulate them efficiently. Traditionally, images and video sequences have been manually annotated with a small number of keyword descriptors after visual inspection by a human reviewer. Unfortunately, this process can be very time-consuming, and such delays, although perhaps acceptable for archival applications, may inhibit the ability to perform near-real-time filtering and retrieval.

For some media, the problem of indexing is better understood and has been more thoroughly addressed. For example, there has been tremendous success in the automatic conversion of hard-copy documents via optical character recognition (OCR) technology [20, 22, 35, 41] and the transcription of speech via voice recognition (VR) technology [24, 43]. In both cases, the output, although typically less than perfect, is an ASCII text representation which can be indexed with traditional information retrieval techniques. Unfortunately, when dealing with visual information, we do not always have a linguistic representation of content. We do find, however, that some *information-rich* video sources such as newscasts, commercials, movies and sporting events contain meaningful content in the form of voice, closed-caption text and/or text in the image. These sources often complement each other, and the ability to use them can provide essential supplemental information useful for indexing and retrieval.

In this paper, we address only the problem of detecting and tracking text in digital video. Although sound and closed captions provide index information on the spoken content, basic annotational information often appears only in the image text. For example, sports scores, product names, scene locations, speaker names, movie credits, program introductions and special announcements often appear and supplement or summarize the visual content. These types of annotations are usually rendered in high contrast with respect to the background, are of readable quality, and use keywords that facilitate indexing. Specific searches for a particular actor or reference to a particular story can easily be realized if there exists access to this textual content.

New standards in video coding are extending the scope of video to streaming multimedia. The efforts of the Motion Picture Experts Group (MPEG), part of the joint International Organization for Standards (ISO) and International Engineering Consortium, are producing standards which are object-based. Within these standards video can be encoded as a static background with moving foreground composed of various objects, ultimately allowing annotation of textual content as an object and making it easier to extract and use as an indexing tool. Unfortunately, the standards are not yet widely implemented and are not expected to be accepted as mainstream for some years. Nevertheless, the extraction of textual content rendered as part of the video and text embedded in the scene is extremely valuable for indexing.

2 Background

Before we embark on the general subject of text extraction it is useful to consider some specific aspects of the problem. We first discuss synthetic and natural sources of textual content within video frames and then review some related work and present our approach.

2.1 Scene Text and Graphic Text

At a high level, text can be divided into two classes, *scene* text and *graphic* text. Scene text appears within the scene which is then captured by the recording device. It is an integral part of the image and can be considered a sample of the real world. Examples of scene text include street signs, billboards, text on trucks and writing on shirts. Although valuable, the appearance of such text is typically incidental to the scene content, and only useful in applications such as navigation, surveillance or reading text appearing on known objects, rather than general indexing and retrieval. One exception is in constrained domains, where text or symbols may be used to identify players or vehicles. Scene text is often difficult to detect and extract since it may appear in a virtually unlimited number of poses, sizes, shapes and colors.

Graphic text, on the other hand, is text that is mechanically added to video frames to supplement the visual and audio content, and is often more structured and closely related to the subject than scene text is. Examples of graphic text include headlines, keyword summaries, time and location stamps, names of people and sports scores. The descriptors are typically predictable, have simple styles, and are produced with the intent of being read by the viewer.

Graphic text has a number of functions which differ between domains. In commercials, text appears to reinforce the vital information such as the product name, claims made, or in some cases, to provide disclaimers. In sporting events, text is used to identify specific players, provide game information, or relay statistics. In newscasts, graphic text can be used to either identify key features of the scene, such as location (White House lawn) or speaker (Bill and Hillary), to provide a synopsis of the topic (Blizzard 97), or to provide a visual summary of statistical information. In movies and television shows, text provides production and acting credits, and in other cases captions or language translations. For the most part, research in this area has focused on the identification of graphic text.

2.2 Related Work

In related domains there has been work on the extraction of text from road signs [32, 44], license plates [10, 29, 31], library books [19], WWW images [59, 60], scene images [30, 42, 57, 58] and isolated video frames [33, 36, 53]. The methods can be broadly classified into two types. The first is connected component (CC) based. Unlike binary document images, scene images and video frames are usually multivalued (gray-scale or colored), and therefore multivalued image decomposition is required before the connected components can be extracted. The general idea of the CC-based methods can be described as: In a multivalued image I with pixel values $\mu \in \{0, 1, ..., U-1\}$, where U is an integer greater than 1, I can be decomposed into a set of elementary images $I' = \{I_i\}$ with each I_i having the same pixel value. Under the assumption that text is represented with a uniform color (or intensity), connected components are extracted for each elementary image I_i. Some heuristic restrictions on component size, number of aligned components and line orientation are used to identify text lines. If I is a color image, color clustering techniques are required [26, 28, 59, 60], whereas for gray-scale images, a binarization process is typically used [30, 42]. Depending on the complexity of the color clustering or binarization method used, CC-based methods can locate text quickly, but have difficulties when text is embedded in complex backgrounds or touches other text or graphical objects.

The second approach is texture-based and uses well-known texture analysis methods such as Gabor filtering [25], Gaussian filtering [56] or spatial variance [58] to locate text regions. In [25] Jain describes a method of separating text and image areas based on a group of multichannel Gabor filters. Wu and Manmatha present a text extraction system based on Gaussian filtering [56] and treating text as a distinctive texture. After filtering, each pixel in the original image will be represented by a feature vector that consists of the energy calculated from the filtered images. Clustering methods, such as K-means, are then used to cluster these feature vectors.

For text detection in digital video, some authors use interframe analysis to add missing characters or to delete incorrectly identified regions. Shim detects text in five consecutive frames and then examines the similarity of text regions in terms of their positions, intensities and shapes [53]. In [36] Lienhart extracts text in one frame and then checks if the region corresponds to text in the following frame by using simple region matching. Five frames are used to filter out incorrectly identified text regions. Neither work addresses the problem of tracking text to find temporal correspondence of detected text in digital video. For example, in Lienhart's system, all the text in video frames is saved in a database after recognition.

The literature on object tracking is extensive and includes face tracking [4, 14, 21], human body tracking [18, 49, 55], vehicle tracking [16], medical imaging [2, 52] and agricultural automation [23, 47]. A detailed survey of tracking algorithms for non-rigid motion can be found in [1], but there appears to be very little appropriate literature on the tracking of text.

2.3 Our Approach

Text detection and tracking in digital video can be viewed as a multi-target detection and tracking problem since multiple text blocks can appear at any time and can move in different directions within the field of view. Unlike other typical tracking problems with a relatively static background and moving objects, we can have four combinations of static and moving text and background if we treat all non-text as background. It would be very difficult, if not impossible, to detect and track text simply by motion analysis. Although in some special cases such as news or movie credits, we can make use of motion information to segment text, in general, motion information alone is not sufficient. A text detection scheme is required in conjunction with tracking.

A very simplistic solution to text tracking would be to perform the text detection frame by frame and then match the corresponding text blocks between consecutive frames. Considering that there may be close to 30 frames per second, the text detection phase in the absence of context would be prohibitive. We can, however, make use of the fact that text remains in the scene for a number of consecutive frames to reduce the complexity by performing the text detection process periodically and focusing on the tracking process (Figure 1).

Our text detection scheme is based on the observation that text regions typically have different texture properties than the surrounding areas. This texture has similar frequency and orientation information, making wavelets a reasonable candidate for representation. We use a hybrid wavelet/neural network segmenter (Figure 2) to segment text regions. To facilitate the detection of various text sizes, we use a pyramid of images generated from original image by halving the resolution at each level. The extracted text regions are hypothesized at each level

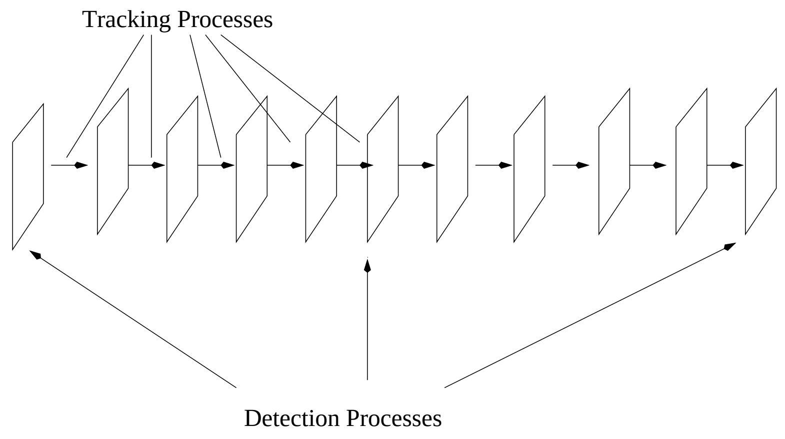


Figure 1: The scheme for text detection and tracking in digital video.

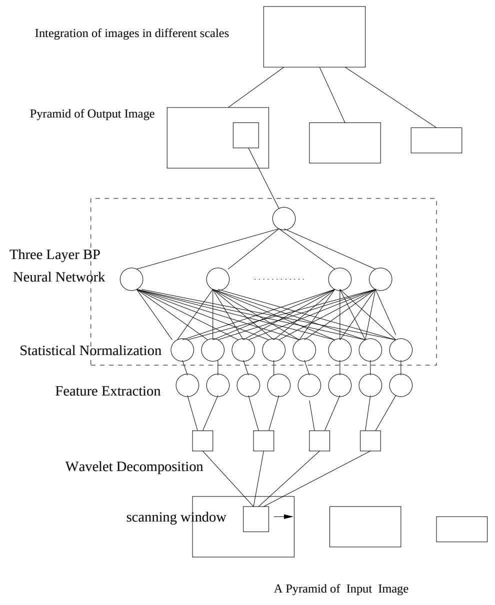
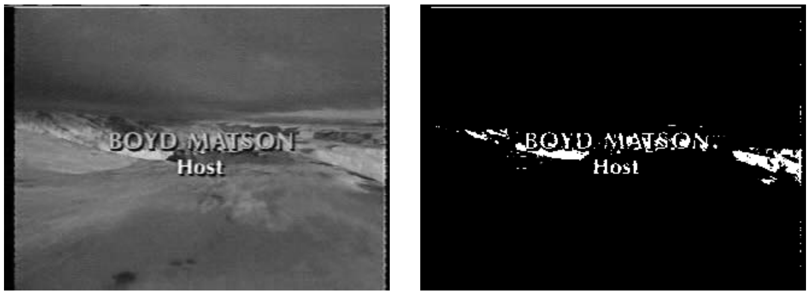


Figure 2: The architecture for detecting text in video frames.

and then extrapolated to the original scale.

Once the text is detected, a multi-resolution SSD (Sum of Squared Differences) based tracking module is started to track the detected text. We make use of the text contour to stabilize the tracking so text can be tracked more robustly.

The remainder of this paper is organized as follows. We describe our text detection scheme in Section 3 and our text tracking scheme in Section 4. Experimental results are presented in Section 5 and in Section 6 we discuss some immediate applications of our work.



(a) (b)

Figure 3: (a) A video frame, (b) binarization by manually picking best thresholds.

3 Text Detection in Video Frames

Compared with other text detection problems, text detection in digital video has several new challenges. First, text in digital video is usually embedded in a complex background, which makes it difficult for the Connected Component (CC)-based methods. Figure 3a shows a typical video frame and Figure 3b is the binarized version using a good threshold. The binarized image shows connectedness between character components and the background. Second, text in digital video is rendered at low resolution. For document images, scans of 300 dots per inch are common, which results in "normal" characters occupying an area as large as 50×50 pixels. Video frames are usually limited in spatial resolution to as low as 352×240 with a character size of approximately 10×10 pixels. The effect of low resolution is two-fold: First, the text is often aliased so color clustering may result in scattered text. Second, text components tend to touch each other, making detection difficult.

In some video frames, natural scenes like the leaves of a tree or grass in a field have textures similar to text. In feature space, text and nontext often overlap. To illustrate this point, we collected 500 text blocks and 500 nontext blocks and used Linear Discriminant Analysis (LDA) [17] to project them to a new subspace where the between-class scatter is maximized and the within-class scatter is minimized, which benefits the classification. The projection result (Figure 4) shows that text and nontext overlap in the feature space. Therefore, we think that using supervised learning to classify text and nontext will be more effective than the unsupervised clustering techniques used in [25, 56]. An artificial neural network is a natural

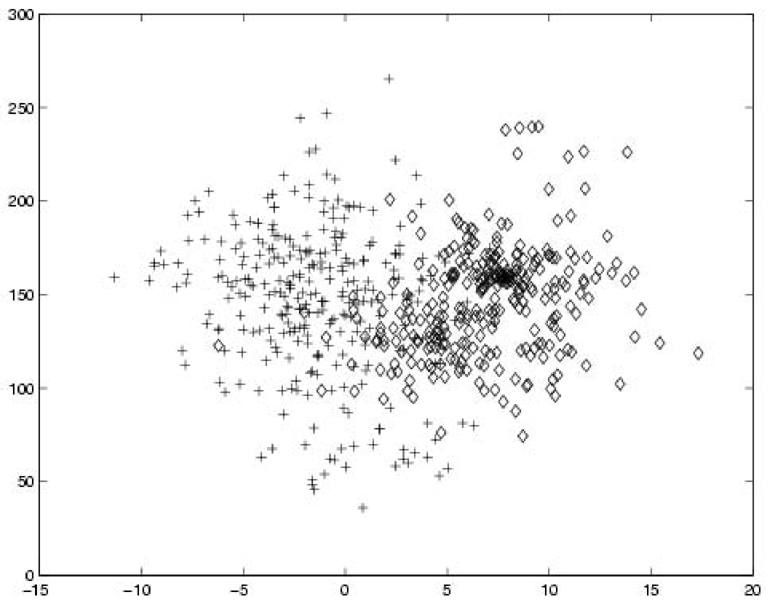


Figure 4: The distribution of text (◇) and nontext (+) in the subspace after LDA projection. Text and nontext overlap.

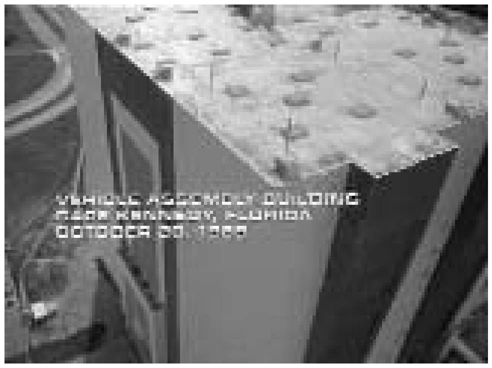
choice as a classifier because of its ability to learn. Theoretically, a three-layer neural network can approximate any nonlinear function after training. The success of neural networks in related problems [8, 11, 13, 39, 48] provides us with further motivation to rely on a neural network as a classifier to identify text regions.

Our methodology uses a small window (typically 16×16) to scan the image and classify each window as text or non-text using the neural network (Figure 2). We will address the following issues related to the problem:

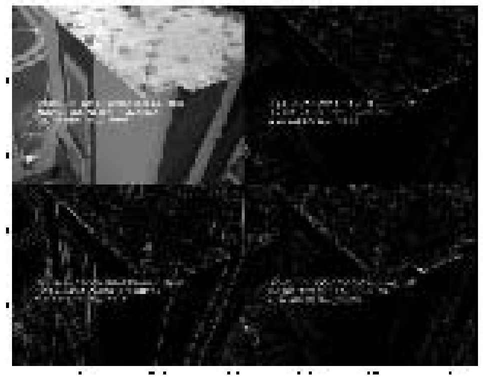
- Scale-space decomposition of the image,
- Feature extraction and selection,
- Classification using the neural network,
- Text identification.

3.1 Scale Space Decomposition

Analysis of scale space provides a method of identifying the spatial frequency content in local regions within the image. We use wavelets to decompose the image because they provide successive approximations to the image by downsampling and have the ability to detect edges



(a)



(b)

Figure 5: Single-level wavelet decomposition of a video frame: (a) Original image, (b) Decomposed images.

during the high-pass filtering. The low-pass filter creates successive approximations to the image while the detailed signal provides a feature-rich representation in terms of textual content [38]. This is easily seen in the image decomposition shown in Figure 5 where 5a is the original image and 5b is its first-level wavelet decomposition. Note that the text region shows high activity in the three high-frequency subbands (HL, LH, HH). As a result of their local nature, only wavelets which are located on or near the edge yield large wavelet coefficients, making text regions detectable in the high frequency subbands. High frequency components are important for recognition. Jones and Palmer have shown that wavelets can closely represent the response profiles of neurons in the striate cortex [27].

Depending on the application, the choice of suitable wavelets and the corresponding decomposition level are based on different criteria. In our case, Haar wavelets were chosen since they are adequate for local detection of line segments, which characterize text regions. In addition to their simplicity, image decomposition can be implemented with simple mask operation, i.e., weighted sums and comparisons, making Haar wavelets computationally efficient. Furthermore, little or no improvement was observed when using other types of wavelets, based on our experiments.

The scaling and wavelet functions of Haar wavelets can be written as

$$\phi(x) = \sum_{k \in Z} p_k \phi(2x - k) = \phi(2x) + \phi(2x - 1) \tag{1}$$

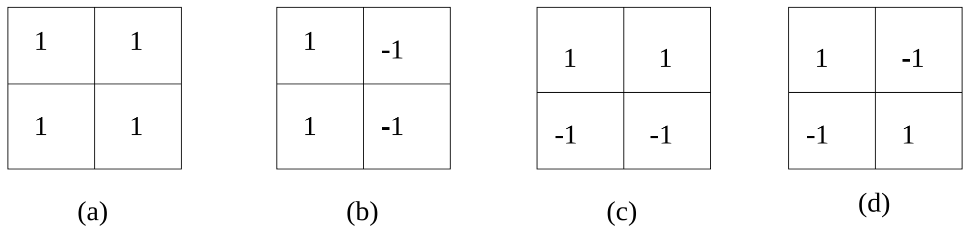


Figure 6: Haar masks. (a) Lowest frequencies, (b) Vertical high frequencies, (c) Horizontal high frequencies, (d) High frequencies in horizontal and vertical directions.

$$W_H(x) = \sum_{k \in Z} q_k \phi(2x - k) = \phi(2x) - \phi(2x - 1) \tag{2}$$

respectively, with

$$\phi(x) = \left\{ \begin{array}{ll} 1 & \text{for } 0 \le x < 1 \\ 0 & \text{otherwise} \end{array} \right\} \tag{3}$$

The two-scale sequences p_k in Equation 1 have non-zero values $p_0 = p_1 = 1$ and zero values for all other p_j. q_k in Equation 2 is zero except for $q_0 = 1$ and $q_1 = -1$.

For an image $I(x, y)$ represented as

$$I(x,y) = \begin{bmatrix} i_{0,0} & i_{0,1} & \cdots & i_{0,2N-1} \\ i_{1,0} & i_{1,1} & \cdots & i_{1,2N-1} \\ \vdots & \vdots & \vdots & \vdots \\ i_{2N-1,0} & i_{2N-1,1} & \cdots & i_{2N-1,2N-1} \end{bmatrix}_{2N \times 2N} \tag{4}$$

we can use Mallat's algorithm [37] to obtain the two-dimensional Haar wavelet transform of $I(x, y)$:

$$LL_{x,y} = \frac{1}{4} \sum_{k_1,k_2=0}^{1} p_{k_1} p_{k_2} i_{k_1+2x,k_2+2y} = \frac{1}{4}(i_{2x,2y} + i_{2x,2y+1} + i_{2x+1,2y} + i_{2x+1,2y+1}) \tag{5}$$

$$LH_{x,y} = \frac{1}{4} \sum_{k_1,k_2=0}^{1} p_{k_1} q_{k_2} i_{k_1+2x,k_2+2y} = \frac{1}{4}(i_{2x,2y} - i_{2x,2y+1} + i_{2x+1,2y} - i_{2x+1,2y+1}) \tag{6}$$

$$HL_{x,y} = \frac{1}{4} \sum_{k_1,k_2=0}^{1} q_{k_1} p_{k_2} i_{k_1+2x,k_2+2y} = \frac{1}{4}(i_{2x,2y} + i_{2x,2y+1} - i_{2x+1,2y} - i_{2x+1,2y+1}) \tag{7}$$

$$HH_{x,y} = \frac{1}{4} \sum_{k_1,k_2=0}^{1} q_{k_1} q_{k_2} i_{k_1+2x,k_2+2y} = \frac{1}{4}(i_{2x,2y} - i_{2x,2y+1} - i_{2x+1,2y} + i_{2x+1,2y+1}) \tag{8}$$

The image decomposition process is equavalent to the convolution of the image $I(x, y)$ with the kernels in Figure 6. The computational efficiency is obvious.

3.2 Feature Extraction and Selection

The task of feature extraction and selection cannot be solved in closed form. Our methods of feature extration and selection are explained in this section.

3.2.1 Feature Extraction

For our proposes we extract features from the wavelet decomposition of the image since (as shown in Figure 5) text has a different texture from background in subbands and is detectable. We use as features the mean and the second- and third-order central moments. For an $N \times N$ subblock I we calculate the mean, the second- and third-order central moments of the subblock which can be written as

$$E(I) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(i,j) \tag{9}$$

$$\mu_2(I) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I(i,j) - M(I))^2 \tag{10}$$

$$\mu_3(I) = \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I(i,j) - M(I))^3 \tag{11}$$

All the features are computed on decomposed subband images. Since the original window size is 16×16, the maximum decomposition level we can choose is 4, and only one pixel is left for each subband image on the fourth level.

3.2.2 Feature Saliency

Feature Saliency is defined as a measure of a feature's ability to impact classification. A benchmark to compare feature saliency is a probability of error criterion, denoted by P_e, which computes the probability of error when only a single feature is used. Another saliency metric for features measures the MLP outputs, sampled over an appropriate range of allowable input values [50]. Empirical results indicate that this metric provides similar rankings to P_e [48].

We use Bayes error rate as a measure of saliency. The Bayes error rate can be computed (or estimated) to determine whether or not the feature will yield adequate separation between the classes. In most practical cases, the Bayes error rate is estimated using a finite set of labelled samples from the various classes. This is typically done by estimating the posterior probability of each class for each sample, then assigning each sample to the class with the MAP (maximum

Decomposition Level	Subband Image	P_e E(I)	P_e $\mu_2(I)$	P_e $\mu_3(I)$
1	LL	0.3600	0.3000	0.3325
	HL	0.4519	0.2256	0.2400
	LH	0.4281	0.2256	0.2344
	HH	0.3969	0.2387	0.2356
2	LL	0.3600	0.3750	0.3612
	HL	0.4169	0.2131	0.2087
	LH	0.4050	0.2481	0.2538
	HH	0.3769	0.1987	0.1950
3	LL	0.3600	0.4669	0.4613
	HL	0.4038	0.3750	0.3912
	LH	0.4113	0.3025	0.2963
	HH	0.3556	0.3069	0.2919

Table 1: Using Bayes error P_e to measure feature saliency at different decomposition scales.

a posteriori) probability estimate. The percentage of samples misclassified by applying the MAP decision rule to the posterior estimates is taken as an estimate of the Bayes error rate.

A Java-based interface was implemented to collect text samples and nontext samples. The interface can allow the user to load images and then use simple mouse operations to label text regions and nontext regions to collect samples. We collect 1000 text blocks and 1000 nontext blocks and used 600 samples from each class for training and the remaining samples for testing.

Table 1 lists the P_e of each feature. Three conclusions can be drawn from Table 1. First, the features in the third-level decompostion have larger P_e values than features in the first two levels. The explanation is that the subband image size in the third level is 2×2, and therefore provides less information. Second, on all levels, the P_e values of the mean are generally larger than those of the second- and third-order moments. Finally, for the second- and third order moments, P_e in the high-frequency subband image (HL,LH, HH) is smaller than in the low-frequency subband image (LL) on the same decomposition level. This implies that we should use the second- and third-order moments calculated from the high-frequency subband images on the first two decomposition levels.

In order to verify the above conclusions, we tested the data on other wavelets. We ignore the third level here since it is unlikely the third level can provide additional discrimination. Table 2 is the result when we use Daubechies-2 wavelets and Table 3 is the result when we use Coiflets-3 wavelets. Although there exist small variations, we can still draw the same conclusions as in

Level Decomposition	SubImage Subband	Expectation P_e	Variance P_e	3rd order Moment P_e
	LL	0.3600	0.4181	0.4644
1	HL	0.4519	0.4137	0.4369
	LH	0.4281	0.3588	0.3669
	HH	0.3962	0.3350	0.3337
	LL	0.3600	0.4537	0.4694
2	HL	0.3706	0.4344	0.4550
	LH	0.3950	0.3725	0.3769
	HH	0.3975	0.3825	0.3700

Table 2: The Bayes error P_e when we use Daubechies-2 wavelets.

Level Decomposition	SubImage Subband	Expectation P_e	Variance P_e	3rd order Moment P_e
	LL	0.3600	0.4513	0.5019
1	HL	0.4519	0.4219	0.4300
	LH	0.4281	0.3350	0.3225
	HH	0.3969	0.3700	0.4087
	LL	0.3600	0.3925	0.4281
2	HL	0.4206	0.3506	0.3550
	LH	0.3931	0.3488	0.3600
	HH	0.3794	0.2169	0.2225

Table 3: The Bayes error P_e when we use Coiflets-3 wavelets.

the Haar wavelet case. By comparing these results with Table 1, we can see that the Haar wavelets outperform the Daubechies and Coiflets wavelets for this problem.

3.2.3 Feature Selection

The criterion for choosing features is to keep the feature set as small as possible where the features maximize the classification result. One obvious reason for a reduced feature set is the efficiency of training with smaller numbers of parameters. Some authors have addressed the impact on required training samples of the dimensionality of the input features [3, 9, 15]. The dimensionality curve shows that the number of training samples required grows exponentially with the number of features.

The saliency analysis in the last section can help us choose a feature set. We rank the twelve features in the order of increasing Bayes error P_e as shown in Table 4. We use μ_{2i}^{j}

feature	μ^2_{3HH}	μ^2_{2HH}	μ^2_{3HL}	μ^2_{2HL}	μ^1_{2HL}	μ^1_{2LH}	μ^1_{3LH}	μ^1_{3HH}	μ^1_{2HH}	μ^1_{3HL}	μ^2_{2LH}	μ^2_{3LH}
P_e	0.195	0.198	0.208	0.213	0.225	0.225	0.234	0.235	0.238	0.240	0.248	0.253
rank	1	2	3	4	5	6	7	8	9	10	11	12

Table 4: Ranking the features in the order of increasing P_e.

and μ^j_{3i} to denote the second- and third-order moments respectively, with j representing the decomposition level and i representing the subband.

Our scheme for selecting features is to use the first feature (μ^2_{3HH}) in Table 4 to train and classify and then calculate the corresponding classification error. We then add another feature and use the first two features (μ^2_{3HH} and μ^2_{2HH}) to repeat the process. At each step one more feature is added until all twelve features are used.

One limitation of the Bayes method is its inability to deal with high-dimensional data. We therefore use a neural network instead of the Bayes classifier used in the analysis of saliency. By changing the number of input features, the parameters of the neural network (the numbers of input and hidden-layer nodes) also need to change. For each group of inputs, we adjust the number of hidden-layer nodes, so the minimal classification error can be reached. Figure 7 illustrates the relation between the classification error and the number of features used. Although the best classification result is achieved when all twelve features are used, we have dropped the last four features after considering the balance between accuracy and processing time (classification is conducted for each 16×16 window in each video frame and accounts for the principal processing time for text detection). The first eight features in Table 4 are selected for classification and the corresponding configuration of the neural network consists of three layers with 8 input nodes, 12 hidden nodes and 1 output node.

3.3 Training the Neural Network

After selecting the features, we re-train the neural network. The samples we collected in feature analysis may be not enough to cover the feature space extensively. Although it is easy to get representative samples of text, it is more difficult to get representative samples of non-text since non-text spans a vast space. To deal with this we use a *bootstrap* method recommended by Sung and Poggio [54] to re-train. The idea is that the training samples are collected during training instead of before training as follows:

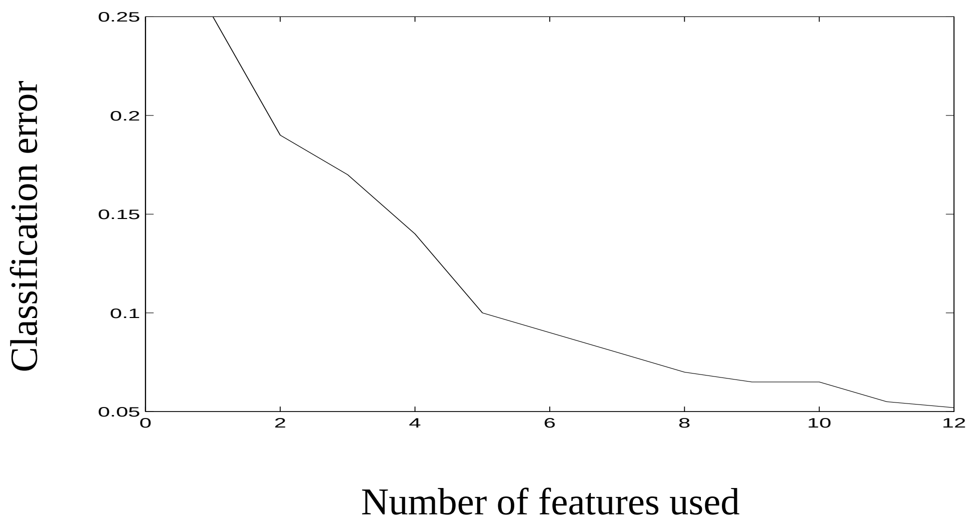


Figure 7: Test set classification error as the number of features is increased.

1. Create an initial set of training samples which includes a complete set of text samples and a portion of non-text samples.
2. Train the network on these samples.
3. Run the system on a video frame which contains no text and add image blocks which the network incorrectly classifies as text to the non-text sample set.
4. Repeat steps 2 and 3 until the accuracy converges.

After training, the neural network maps each block into a real value between 0 and 1 for nontext and text respectively. Figure 8a shows the distribution of the neural network outputs when testing on text and nontext blocks after training. A threshold of 0.5 is a reasonable choice to indicate whether the window contains text or not. We can see that the text and nontext classes are well separated when using the neural network as a classifier.

3.4 Text Identification

After training the neural network, we use a 16×16 window to scan the video frame to classify each window as text or nontext. The larger the step by which window moves, the smaller the number of windows to be processed but the less refined the result. For a video frame of size 352×240, the system needs to classify 75825 windows if the window moves 1 pixel at a time; this takes more than 4 seconds on a Sun Ultra I. On the other hand, for a step size of 16 pixels, where the windows don't overlap, a text line may be split between the windows and may not be

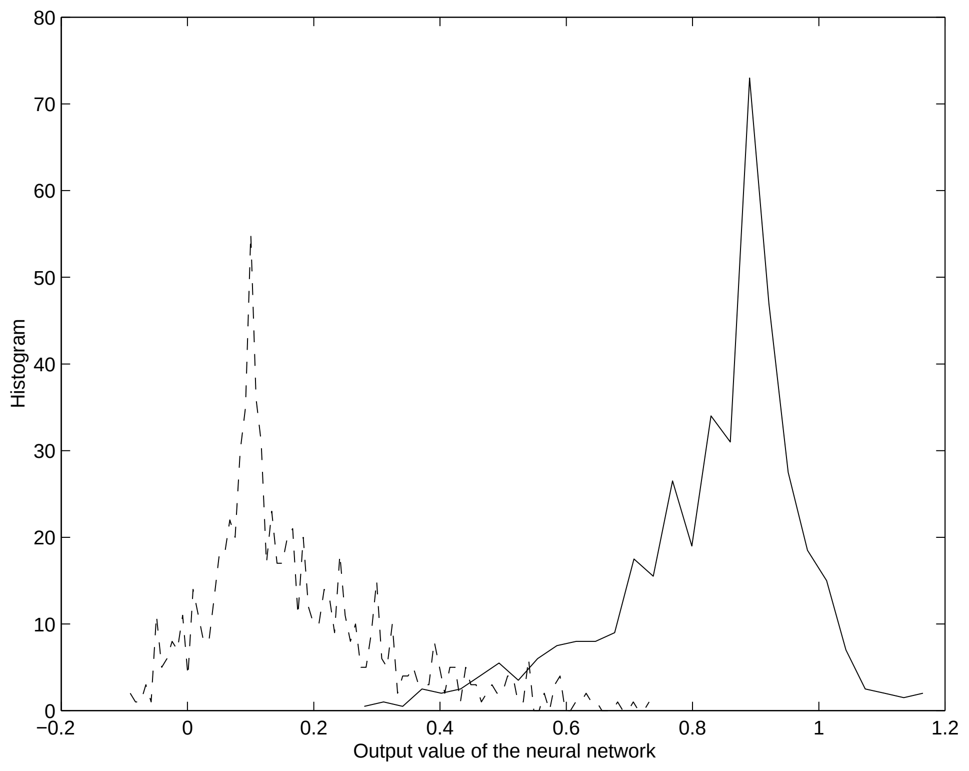


Figure 8: The output distribution when we use the trained neural network on the testing text and nontext blocks. (The solid line represents text and the dashed line represents nontext.)

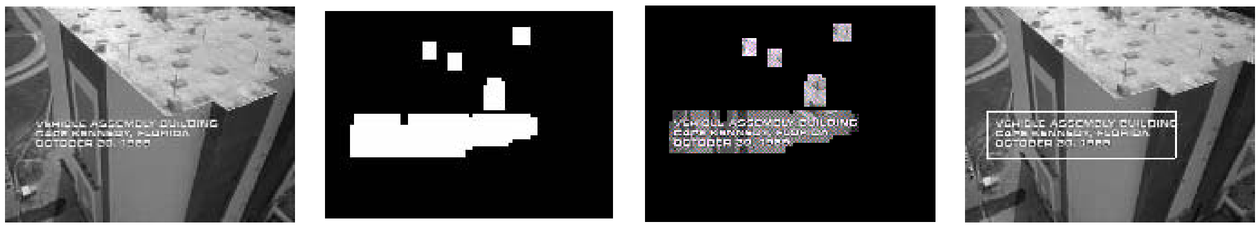


Figure 9: (a) Original frame, (b) The classified label map, (c) The segmented text area corresponding to (b), (d) The segmented text area after post-processing and bounding box generation.

detected. Considering the trade-off between precision and speed, we move the window 4 pixels at a time.

If a window is classified as text, all the pixels in this window are labeled as text. Those pixels which are not covered by any text window are labeled 0. The result of the classification is a label map of the original image. Figure 9(a) is a video frame and Figure 9(b) is the classified label map corresponding to Figure 9(a). Figure 9(c) shows the extracted text regions. We can see that all of the text is labelled correctly, but there are some small isolated areas which are incorrectly labeled as text. We use size consistency between blocks to filter out these areas. The bounding box of the text area is generated by connected component analysis of the text windows. Figure 9(d) is the result after we filter out the non-text areas and generate the bounding box.

3.5 Result Integration

As depicted in Figure 2 we use a three-level pyramid approach to detect text with a wide range of font sizes. After the detection step is applied at each level, the bounding boxes are mapped back to the original input image. If the bounding boxes at one level do not overlap with the boxes at other levels, we simply merge them into the original image. In some cases, however, text strings or parts of text strings appear at multiple levels so the boxes detected at different levels overlap (Figure 10(a)). In this case we merge them by forming a bounding box which contains both of them. Figure 10(b) is the integration result corresponding to Figure 10(a).

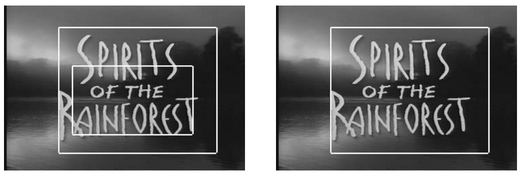


Figure 10: (a) The text regions detected at different levels, (b) Integrated result.

3.6 Postprocessing

The result of our text detection algorithm is a set of text blocks, which may contain multiple text lines if the text is spatially compact (Figure 11a). For the text detection task, we can keep the result in the form of blocks since it will benefit image resolution enhancement [34] and OCR (multiple text lines can provide more context information). However, these blocks are not typically good candidates for tracking because of their size. Furthermore, different text lines in the same text block may move at different speeds, or some text may remain static while other text moves. For these reasons, we need to divide the text blocks into more compact text elements, typically a line or a word.

We use the projection profile to extract text elements from text blocks. To avoid the difficulties of normal text and reverse text, we use the projection profile of the Canny edge map of the image instead of the image itself. Figure 11 is the set of generated text elements corresponding to the detection result in Figure 11a. Blocks which do not have a horizontal orientation will be tracked as a whole.

4 Text Tracking

4.1 Problem Overview

Once text is detected, the tracking process is started. Text motion in digital video is of three types: static; simple linear, rigid motion (for example, scrolling movie credits); and complex nonlinear, nonrigid motion (for example, scene text zooming in and out, or rotation caused by camera motion). For the first two cases, a simple image matching technique may track the text well. For the third case, a more robust technique is required. Our goal is to design a scheme to

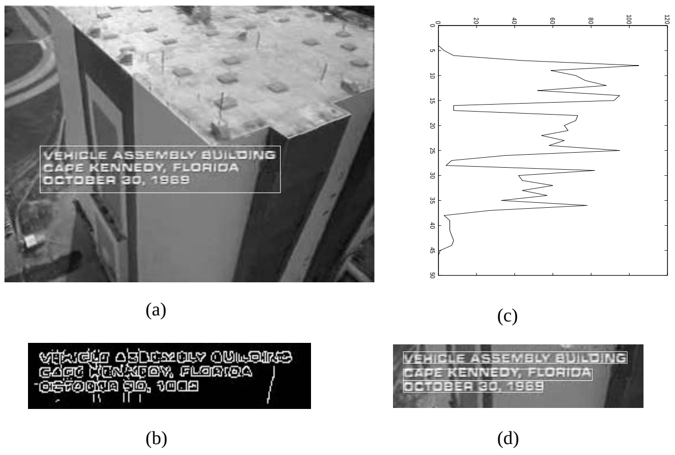


Figure 11: (a) The result of text detection, (b) Canny edge map of detected text block, (c) Horizontal projection profile of (b), (d) Generated text lines.

efficiently track text with both simple and complex motion.

An arbitrary motion can encompass a wide range of possible motion transformations. A general paradigm for estimating motion parameters would be extremely difficult, if not impossible, to develop. In the literature researchers have proposed various approaches to deal with different motion transformations, given certain restrictions imposed on the object's behavior. Robust, reliable visual tracking of an object in a complex environment will require the integration of several different visual modules, each using a different criterion and each employing different assumptions about the incoming images. The modules will be selected so that they complement each other — if one module fails, the other one can come to its aid.

We can treat a text line as a closed set S in the plane, where the bounding box of the text line corresponds to the boundary of S. The boundary B and the interior I of S provide complementary information: B determines the shape (and size) of S and I determines its content (intensity or texture). We believe that the integration of tracking modules based on image content and shape will make the tracking processing more powerful and stable.

We use a general SSD (Sum of Squared Differences) module to measure the similarity of image content (corresponding to the interior I) and then make use of information about the text contour to stabilize the position of the text block (corresponding to the boundary B). In the rest of this section we will give the details of the implementation.

4.2 SSD (Sum of Squared Differences) Based Image Matching

As either the camera or the text moves, the pattern of image intensities changes in complex ways. We can describe these changes as image motion:

$$I(x, y, t+\tau) = I(x - \xi(x, y, t, \tau), y - \eta(x, y, t, \tau)) \tag{12}$$

A subsequent image taken at time $t+\tau$ can be obtained by moving every point in the image taken at time t by an amount $\delta = (\xi, \eta)$, represented by an affine transform:

$$\delta = D\vec{x} + \vec{d} \tag{13}$$

where D is a 2×2 deformation matrix and $\vec{d}$ is a 2×1 displacement vector.

If we choose minimization of the SSD (Sum of Squared Differences) as a matching metric, then the basic tracking algorithm can be described as follows: Given a reference block in image I, and a search region W in image J, determine the six parameters of the deformation matrix D and displacement vector $\vec{d}$, which minimize

$$\epsilon = \int\int_W [J(D\vec{x} + \vec{d}) - I(\vec{x})]^2 d\vec{x} \tag{14}$$

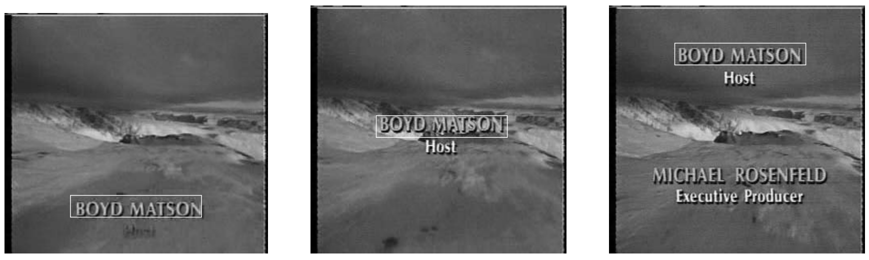
The solution to Equation (14) can be very complex. The quality of the estimate depends on many factors such as the size of the feature window, the texturedness of the image within it, the amount of camera motion between frames, etc. Under the assumption of small inter-frame motion, we can simplify the equation by setting the deformation matrix D to the *identity matrix*. Equation (14) then simplifies to

$$\epsilon = \int\int_W [J(\vec{x} + \vec{d}) - I(\vec{x})]^2 d\vec{x} \tag{15}$$

The model described by Equation (15) is a *pure translational* model. The search space W is within some range of the predicted location. We use a simple prediction technique to locate W more accurately. Suppose $\vec{x}_n, \vec{x}_{n-1}$ are the text positions in the current and past frames; then

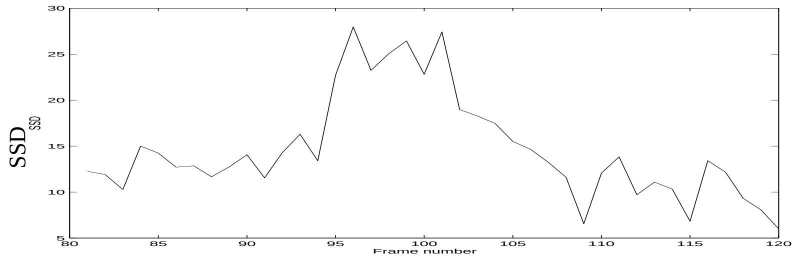
$$\vec{x}_{n+1} = \vec{x}_n + (\vec{x}_n - \vec{x}_{n-1}) \tag{16}$$

In essence this is a second-order linear predictor. Although we could use more precise and robust models such as high-order adaptive linear prediction or Kalman filtering, this will add to the computational complexity.



Frame 81 Frame 100 Frame 116

(a)



Frame Number

(b)

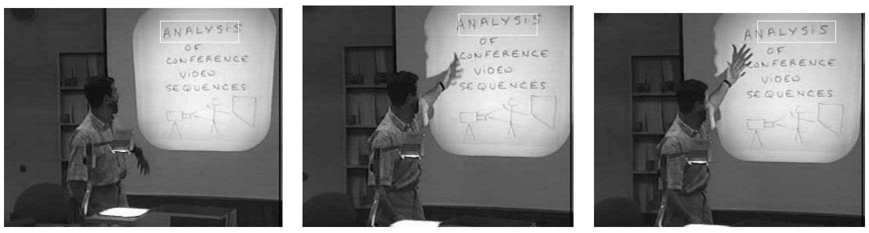
Figure 12: Using an SSD-based model to track a movie credit. The initial position is specified manually. (a) Tracking result, (b) Graph of SSD in consecutive frames. SSD is obviously higher in frame 95–105 since the text line moves onto a complex background.

The pure translational model tracks well in the cases of simple motion or static text. Figure 12a shows an example of tracking movie credits. Although the text line being tracked moves from a clean background to a complex background (in the middle of the frame), the text line is correctly tracked. Figure 12b shows the graph of the SSD.

Evidently, the pure translational model is not adequate to handle scale, rotation and perpective distortions [21, 51]. While it is computationally efficient, over a long sequence the error will accumulate to the point that the tracker loses the target [51]. Figure 13 shows the tracking result when we track a text line in a conference video sequence. Both camera and scene change and the text moves in a complex way including zooming, rotation, etc. Although the SSD is small enough in consecutive frames (Figure 13b), the tracker gradually loses the target after several frames.

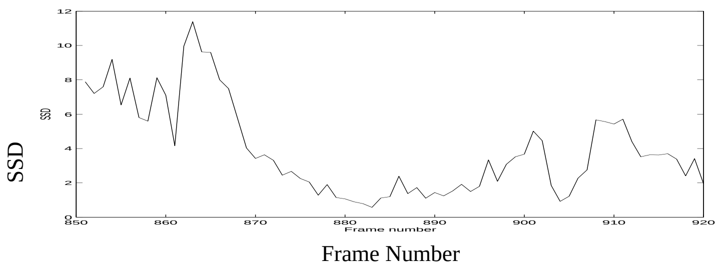
4.3 Contour-Based Text Stabilization

When text undergoes complex motion, the SSD-based pure translational model does not track it well. There exist more elaborate tracking models including parametrized models for articulation



Frame 850 Frame 860 Frame 870

(a)



(b)

Figure 13: SSD-based methods will fail when there are distortions. In this conference video, there exist scale change and rotation.

[7, 45], nonrigid deformations [6, 46] and linear image subspaces [5, 40]. However, the resulting algorithms rely on nonlinear optimization techniques which require from several seconds to several minutes per frame to compute. Instead, we use the text contour to stabilize the tracking process.

The stabilization process can be implemented efficiently in the following way:

1. To matched text position $s = (x1, y1, x2, y2)$, generate a slightly larger text block $s_0 = (x1 - \delta, y1 - \delta, x2 + \delta, y2 + \delta)$. The real text position will be included within s_0.
2. Generate the edge map of s_0 by calculating the Canny edges. We use the edge map instead of thresholding the image to avoid the difficulties of identifying normal text and inverse text.
3. Apply a horizontal smearing process so the edge map can be grouped to form a text block.
4. Extract connected components and their positions $s' = (x1', y1', x2', y2')$ to represent the refined text position.

Figure 14a shows the initial matching position and Figure 14e shows the refined position.

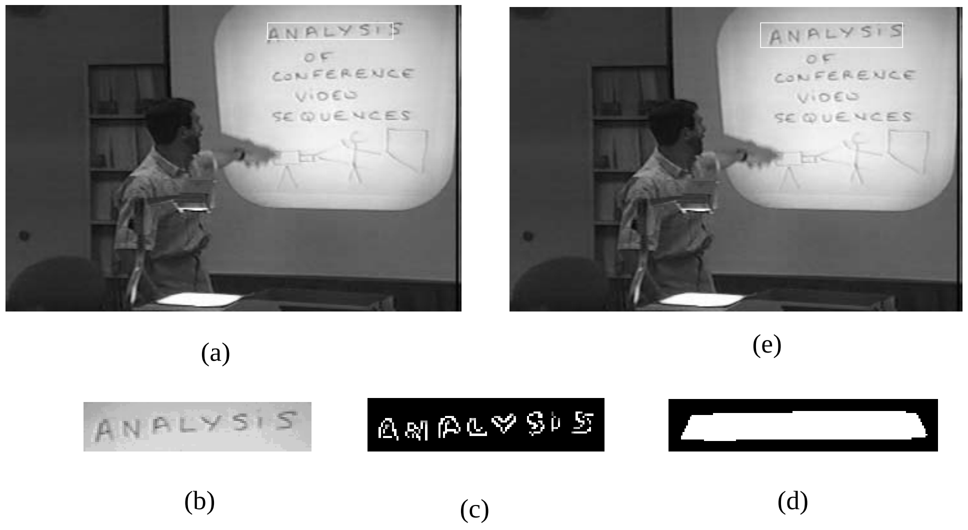


Figure 14: Text position refinement based on contour stabilization. (a) Initial matched text position, (b) an enlarged text block, (c) edge map, (d) smeared image, (e) refined position.

The scheme described above works well when text is moving on a relatively clean background. However, when text moves over a complex background, problems may occur. Over a complex background, text may touch other graphical objects. Since our contour extraction process operates on the slightly enlarged box (as is necessary since we must leave extra space for contour extraction), the text contour will become larger even though the text is in linear, rigid motion.

When the text moves on a complex background, the SSD between two consecutive frames becomes larger since the pixel values in the background change considerably (Figure 12b). In this case, the contour is harder to extract and contour stabilization cannot achieve a satisfactory result. In this case we need to stop the stabilization process and depend only on the SSD model. Once the text moves out of the complex background (SSD becomes smaller again), the stabilization can be started again.

4.4 Using Multi-Resolution Matching to Reduce Complexity

Our SSD-based module is region-based and its computational cost is considerable when we track a large text line. We perform matching from coarse to fine in a hierarchical fashion on a Gaussian image pyramid. For a frame I_t of size $w \times h$, a Gaussian pyramid G_t^l is formed by combining several reduced-resolution Gaussian images of frame I_t, where t is the frame number and $l = \{0, 1, 2, ..., N\}$ represents the level in the pyramid. The size of the frame at level i is

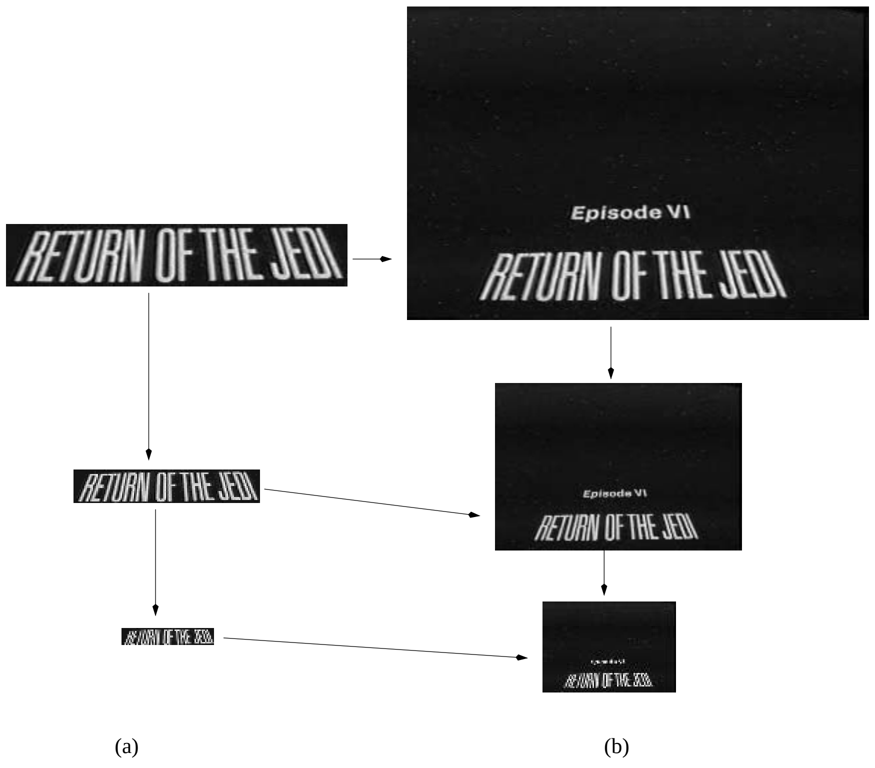


Figure 15: Multiple resolution based image matching. (a) Text block pyramid in Frame 650, (b) Image pyramid formed with Frame 651. The matching process is conducted between images of the corresponding scale.

$\frac{w}{2^i} \times \frac{h}{2^i}$.

The matching is conducted starting at the coarsest resolution (level N). Each level contributes to determine the position of the matching on the next level. The search for the minimum SSD starts on level N over a window size $S = (2s+1) \times (2s+1)$. Suppose the matching point found is $P^N(x, y)$. Then at level $N-1$, the search for the minimum SSD will be conducted around pixel $P^{N-1}(2x, 2y)$ over the same window size $S = (2s+1) \times (2s+1)$. This process continues until the finest resolution (level 0) is reached. Although at each level, the maximum displacement supported by the search is s, which is much smaller than what is required in a one-step search, the displacement is doubled after each level. The total displacement reached at the finest level is $s \times 2^N$. Figure 15 illustrates such a process. The level of the pyramid depends on the size of the text block. If the text line is small enough, no pyramid will be formed and matching will be conducted directly at the original image scale.

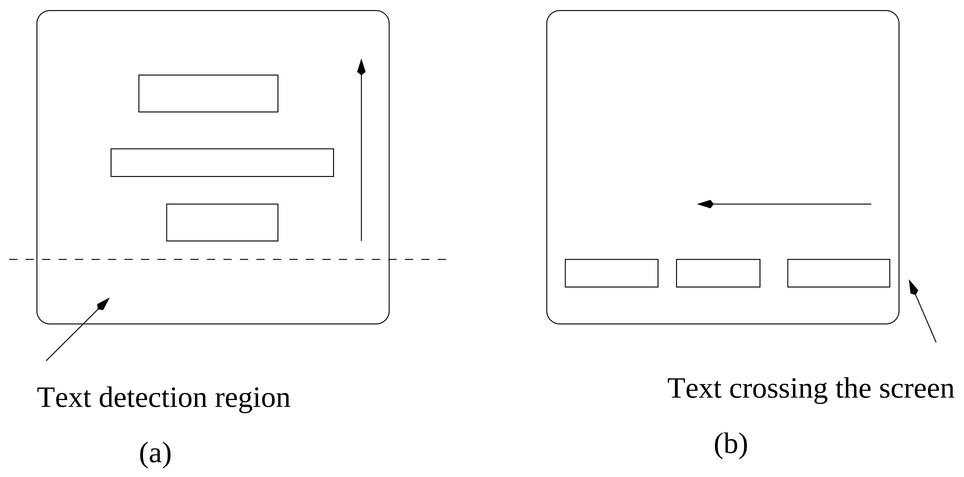


Figure 16: Making use of temporal tendency.

4.5 Temporal Tendency

Although the goal of our system is to track text in the general case, the temporal tendency of text motion can help facilitate the tracking process. If we find, for example, the text lines are scrolling up (or down), we can deduce that new text lines will appear at the bottom (or top) of the video frame. We can then restrict the costly text detection process to a relatively small region (Figure 16a).

When text crosses the frame horizontally, analysis of temporal tendency is necessary to track text correctly. This often happens, for example, in newscasts when announcements cross the bottom of the TV screen. We have no way to track the whole text line in this case since the virtual text line is much bigger than the frame. In this case, we separate the text line into words and track the words. If we find that all the words in a text line are moving horizontally in the same direction and new words continue to appear on the same line, we can draw the conclusion that the text is crossing, and we only need to monitor the small areas where new text words may appear (Figure 16b).

5 Implementation and Experiments

We have implemented three tools based on the methods described above: A text detection tool (*TextDetect*), a text tracking tool (*TextTracker*), and a text detection and tracking tool (*TextDT*). All the programs are written in C and run on a Sun workstation with operating system *Solaris 2.5*. *TextDetect* detects text regions in single video frame. *TextTracker* tracks a text region whose initial position is specified. *TextDT* combines the two modules and conducts

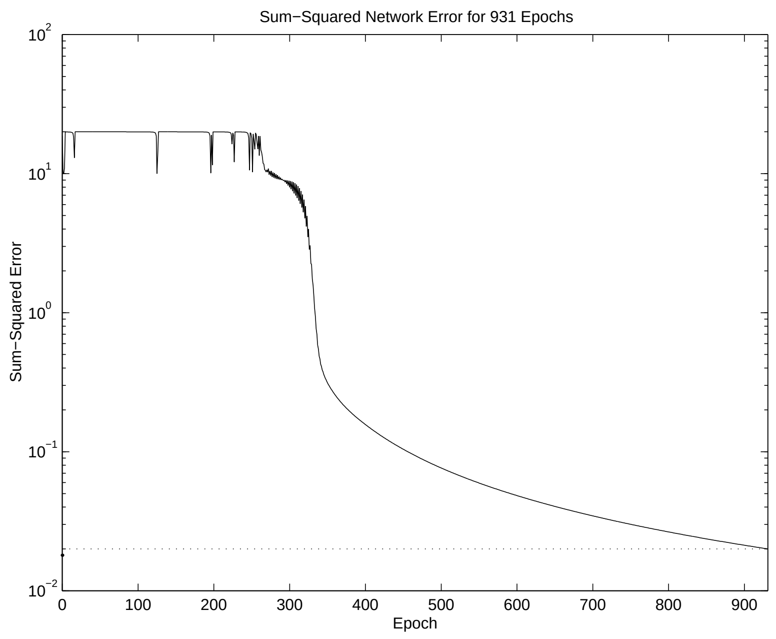


Figure 17: The relation between SSE and training time.

detection and tracking in a sequence of frames. We evaluated the performance of these tools in the following experiments.

5.1 Text Detection

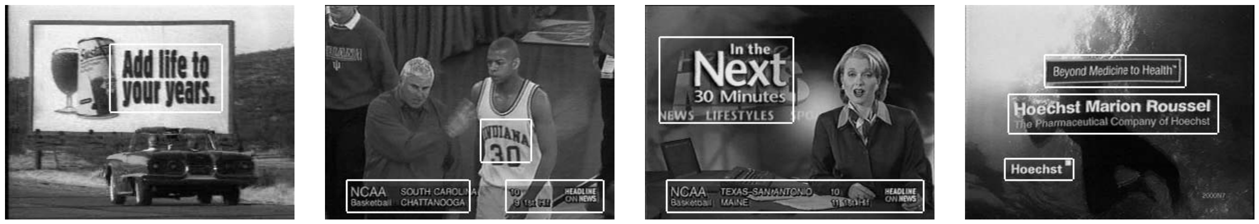
We collected 200 video frames and split them into a training set (50 frames) and a testing set (150 frames). The training samples (text blocks and non-text blocks) were collected from video frames in the training set using the *bootstrap* method described above. The training time depends on the required Sum-Squared Error (SSE). Figure 17 shows that the smaller SSE, the longer the training of the neural network.

The detection procedure requires about 1 second on a Sun Workstation Ultra to process a 352×240 frame with unoptimized code. Classification (including feature extraction) takes 0.5 seconds; postprocessing and image input and output take another 0.5 seconds.

The text collected includes both scene text and graphic text with multiple font sizes. We evaluated our text detection system on the block level and did not segment text blocks into words and characters. A text block may contain one or more text lines which are close to each other. As shown in Table 5, there are a total of 283 text blocks in the 150 frames. 261 (92.4%) of them were correctly detected by our algorithm (Figure 18) and 22 (7.6%) of them were missed. Errors occur primarily because of low resolution (Figure 19(a), the text block at

Frame Number	Text Line	Detected	Missed	False Detection
150	283	261 (92.4 %)	22(7.6%)	23

Table 5: Text detection result


Add life to your years.
INDIANA 30
NCAA Basketball SOUTH CAROLINA CHATTANOOGA
HEADLINE NEWS
In the Next 30 Minutes
NEWS LIFESTYLES
NCAA Basketball TEXAS-SAN ANTONIO MAINE
Beyond Medicine to Health
Hoechst Marion Roussel
The Pharmaceutical Company of Hoechst
Hoechst

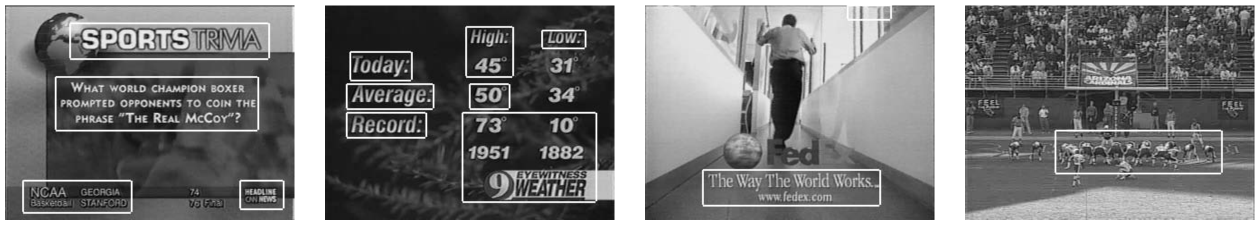
(a) (b) (c) (d)

Figure 18: Part of the identification results. (a) Reverse text with large font size, (b) Text has low resolution (bottom) or is scene text (text on T-shirt of athlete, (c) Text with different font size, (d) Text with different font style.

the bottom of the frame) or small text block size (Figure 19(b), text strings “31” and, “34”).

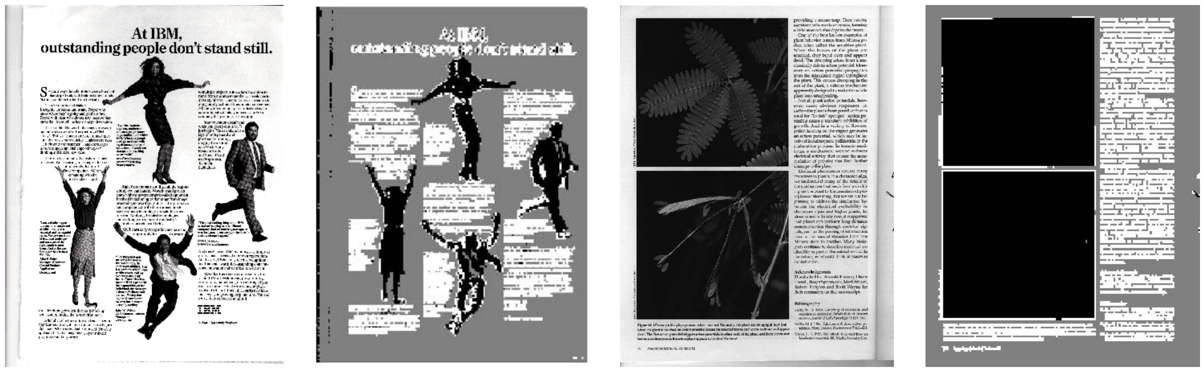
On the other hand, 23 non-text blocks were misclassified as text blocks. Figures 19(c) and (d) are two examples. Further training, domain-specific training, or attempting OCR should overcome these problems.

With slight modifications, *TextDetect* could be used in other detection or segmentation tasks. One direct application is to document image segmentation. We change the number of nodes in the output layer to 3, which corresponds to three classes (text, background and image).



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

Figure 19: Examples of misdetection: (a) A text block at the bottom of the frame is not detected due to low resolution, (b) Text strings “31” and, “34” are not detected due to the small block size, (c) At the top of the frame a non-text block is misclassified, (d) In the middle of the frame a non-text block is misclassified.



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

Figure 20: After slight modification, *TextDetect* can be used to segment document images (white: text, black: picture, gray: backround). (a) A document image scanned from a magazine cover, (b) Segmentation result of (a), (c) A document image in the University of Washington database, (d) Segmentation result of (c).

The same feature extraction scheme is used and the number of hidden layer nodes is adjusted to maximize the classification result. We chose a training set from the gray-scale documents in University of Washington database and tested the system on the rest of the documents in the same database and on scanned magazine covers. Figure 20 shows that the segmentor works well even when the layout of the document is complex.

5.2 Text Tracking

We collected ten video sequences for experiments on text tracking from a wide variety of video sources, including movie credits, TV programs, football games, news and conference videos, etc. A description of the video sequences, which included static text, simple motion, and complex motion, is given in in Table 6.

We conducted our experiments in two steps. First, we studied the performance of text tracking by manually specifying the initial position of the text block, to filter out the possible effects of text detection.

Unfortunately, quantitative evaluation of tracking accuracy is not easy because of the lack of ground truth data. We output the tracking results in the form of MPEG video which can be viewed at website *http://documents.cfar.umd.edu/LAMP/Media/Projects/TextTrack/*. Some of the tracking results are shown in Figure 21.

Our experiments show that the tracker can work well when the text is in simple (rigid, linear) motion (Figure 21c) or when the motion is complex but the background is clean (Figures 21a

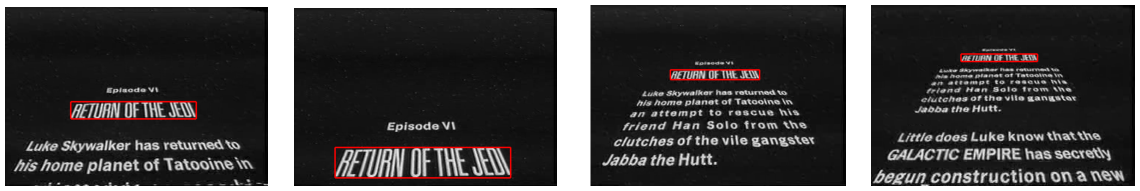
Video ID	Video Type	Frame Number	Text Line	Motion Description
1	Movie Credit	2600	29	Complex
2	Movie Credit	2600	87	Scrolling
3	News	800	8	Static
4	Sports	200	1	Complex
5	Sports	200	2	Complex
6	Conference	800	4	Complex
7	Conference	800	1	Complex
8	Scene	228	7	Crossing
9	TV program	1200	1	Zooming in
10	Commercial	300	1	Crossing

Table 6: Video sources used in experiments

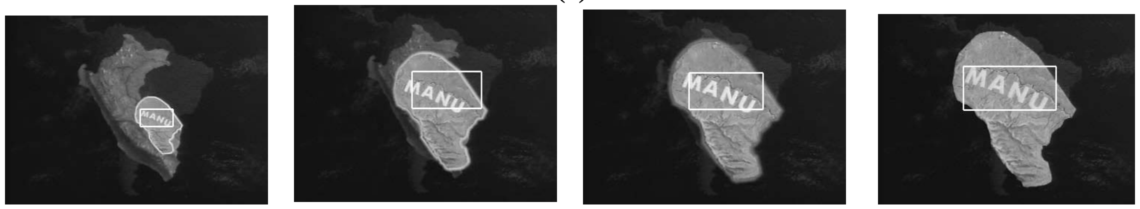
and e). When the text moves arbitrarily on a complex background, the tracker may be confused in some frames but will adjust its position once the text moves to a relatively clean background. In Figure 21b, the initial position specified touches another object (the edge of the map). When the text size increases, the tracking position deviates for some frames but adjusts when the text grows large enough to be separated from the map. Another example tracks the number on an athlete's jersey in a football game. The task is complicated by the athlete's running, jumping, and rotating, as well as by camera motion (Figure 21d). The tracker loses part of the target in some frames but then adjusts to the correct position.

The average tracking time for one text block is about 0.2 second per frame. If the text line is large enough, we can use multi-resolution matching to reduce this. By tracking text we can get text blocks as well as temporal correspondences of the blocks. If we perform detection frame by frame, extra time is required to find the correspondences of blocks between consecutive frames.

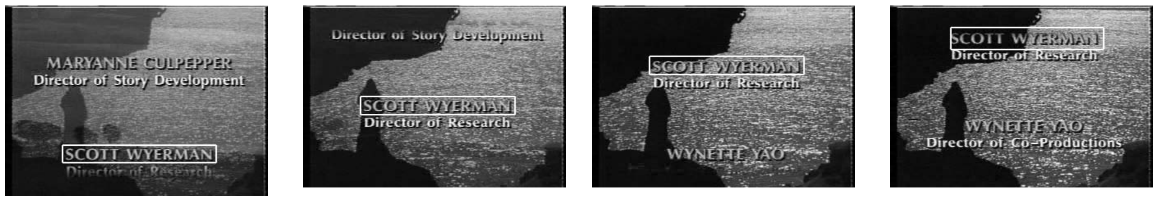
One application of our text tracking tool *TextTracker* is to ground truthing of video data; it is being used for this purpose in the Video Processing Evaluation Resource (ViPER) [12] project in our lab. One of the functions *ViPER* provides is a mechanism to create ground truth data (text, faces, etc) and perform some quantitative evaluations. According to some accounts it may cost as much as $40,000 to ground truth a 120-minute video manually. With the tracking algorithm, we reduce the cost significantly. The user can specify the initial box and then track the rest of the frames. After tracking, the user can check and adjust the text postion if necessary.


Episode VI
RETURN OF THE JEDI
Luke Skywalker has returned to his home planet of Tatooine in
an attempt to rescue his friend Han Solo from the clutches of the vile gangster Jabba the Hutt.
Little does Luke know that the GALACTIC EMPIRE has secretly begun construction on a new

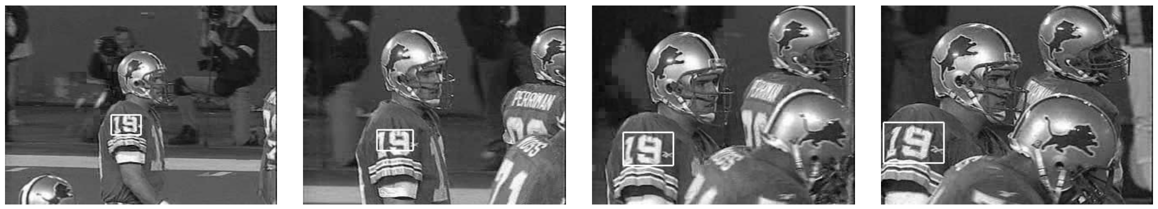
(a)



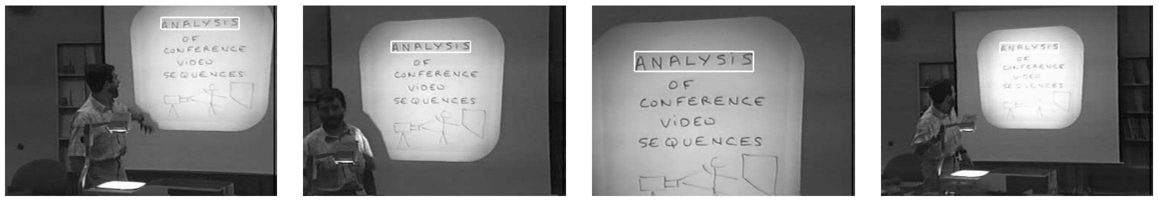
(b)



(c)

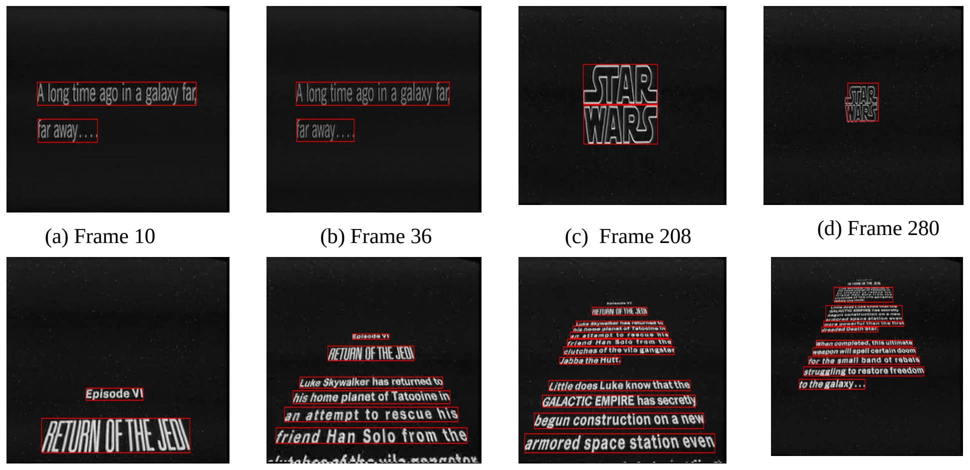


(d)



(e)

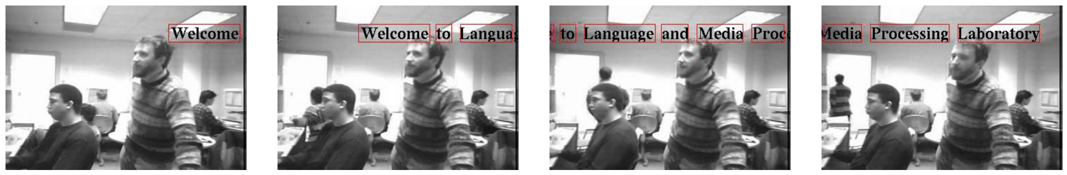
Figure 21: Demonstration of *TextTracker*'s performance on various video sources. The initial position of the text block is manually specified. (a) Tracking of movie credits: Text zooming with displacement. (b) Zooming scene text. (c) Text scrolling up across varying background. (d) Tracking text on a football athlete's jersey. (e) Tracking text in a conference room scene. All of these and other tracking results can be found as *MPEG* video at http://documents.cfar.umd.edu/LAMP/Media/Projects/TextTrack/


A long time ago in a galaxy far,
far away....
STAR WARS
Episode VI
RETURN OF THE JEDI
Luke Skywalker has returned to
his home planet of Tatooine in
an attempt to rescue his
friend Han Solo from the
clutches of the vile gangster
Jabba the Hutt.
Little does Luke know that the
GALACTIC EMPIRE has secretly
begun construction on a new
armored space station even

(a) Frame 10 (b) Frame 36 (c) Frame 208 (d) Frame 280

(e) Frame 652 (f) Frame 974 (g) Frame 1380 (h) Frame 2000

Figure 22: Text detection and tracking in the movie *Star Wars*.



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

Figure 23: "Welcome to the Language and Media Processing Laboratory". (a) Frame 30, (b) frame 60, (c) frame 114, (d) frame 170.

In the second part of our experiments we combined the text detection and the tracking modules. The input to *TextDT* is a sequence of images and the output is a sequence of detected and tracked text regions represented as an *MPEG* video or a ASCII file which records the information for all the text blocks.

Figure 22 shows a tracking result for the movie *Star Wars*. There are 2600 frames in the sequence, which includes static, zooming, and scrolling text. Figure 23 shows tracking results for a transverse text line. We detect it as horizontal scrolling by the temporal tendency analysis described in Section 4, and we thus divide the line into words and track them. All these results can be viewed at website http://documents.cfar.umd.edu/LAMP/Media/Projects/TextTrack/.

The processing time changes considerably with the number of text lines per frame. Tracking movie credits takes more time than other video types since there are more text lines per frame in movie credits. For example, for the movie *Star Wars*, it takes about 1 second to track one frame (the average number of text lines in a frame is 5), while it takes only 0.17 seconds to track text in a football game (only one text line is moving in all of the frames). But as we indicated above, we can detect the text as well as the temporal correspondences of the text blocks.

There are several limitations to our system. First, text tracking is started only when text is detected. If the text detection module fails, the system will miss the text. Second, our tracker uses SSD-based image matching to approximate the position and then uses the text contour to refine the position. Therefore, the system can only be used to track text. In addition, since we use speed prediction to predict the position of the text, the text's acceleration is limited. The tracker has difficulties when text moves too abruptly or keeps moving on a complex background. This happens especially in sports video. For example, when tracking the name of an athlete on a jersey, the text may occlude quickly because of the athlete's jumping and rotating.

6 Applications and Discussion

In this section we will discuss several applications that involve using the detected and tracked text for video indexing.

6.1 OCR in Digital Video

For text-based video indexing, OCR is required to convert the text from image format to plain text. One stumbling block for OCR is that, as mentioned above, the text in digital video is usually at low resolution. As a result, it is beyond the limits of most commercial OCR software. Figure 24a shows a text block extracted from a video frame. Even when we manually pick the best threshold to binarize the image (Figure 24b), there is still no output from the OCR software[1], even though the text is clearly readable.

Text in digital video is limited in spatial resolution with a font size of approximately 10×10 pixels. For typical document images that commercial OCR software works on, scans of 300 dots per inch are common, translating to characters occupying an area as large as 50×50 pixels. This motivates us to perform text resolution enhancement to achieve reasonable OCR results using commercial OCR software so that text-based indexing and retrieval is possible.

One way to improve text resolution is to use multiple frames. This type of technique needs some sort of movement within the frame and the objects contained in it, and usually it requires

[1]We have tested Caere Omnipage 8.0, Xerox TextBridge Pro98 and Xerox ScanWorx.

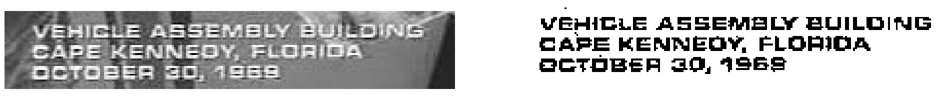


Figure 24: (a) A text block with a font size of approximately 6-7 pixels, (b) Binarization by manually picking the best threshold. There is no output from commercial OCR software although the text is clearly readable.

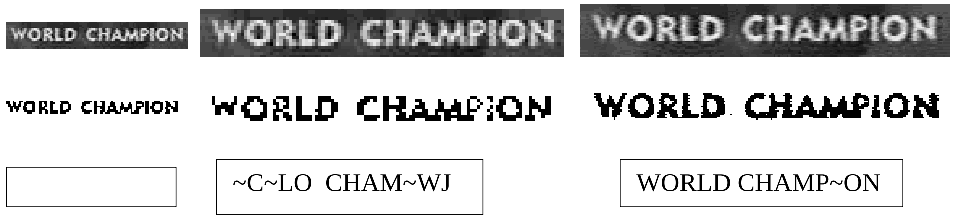


Figure 25: Comparison of OCR results. (a)No enhancement, (b)Zero-order interpolation, (c)Shannon interpolation.

estimation of the movement to sub-pixel accuracy. If the text is static or moves in integer pixel steps, it will not have significant influence. We therefore use image interpolation to improve the resolution directly.

Our experiments have given encouraging results. We chose 45 text blocks extracted from video frames and used Zero-order hold interpolation and Shannon interpolation to increase the image resolution [34]. The same binarization scheme and OCR software (Xerox TextBridge Pro98) was used to test these three groups of images. The OCR accuracy for the original images was 13%. This rises to 34% for Zero-order hold interpolation and to 66.8% for Shannon interpolation (Figure 25). A detailed description of this topic is beyond the scope of this paper and can be found in [34].

6.2 Text-based Indexing and Retrieval in Digital Video

Our ultimate goal is to build a text-based indexing system for digital video. Unlike most retrieval problems, which deal with clean, full text, video indexing based on automatically extracted text has several interesting problems. First, we expect missing or incorrect characters or even words because of poor OCR performance. As a result, exact matches between words will not be possible. We need to use approximate word matching instead of exact word matching. For example, if the user submits "house" as a query, the word "hose" in the database will be considered as a match.

The second problem is that text in digital videos is usually very terse and may lack semantic breadth. Methods that deal with semantic indexing, such as Latent Semantic Indexing (LSI), need extensive training data that has similar characteristics. Intuitively, constructing a semantic dictionary might be a useful approach, but a great deal of work would be required, making this impractical. If we consider only specific topics, such as news, finance, or sports, it should be possible for us to build a semantic dictionary related to each specific topic. For example, if a user submits "Financial" as a query, video frames containing "Stock" can be returned if we put this into the semantic dictionary. We are actively investigating this topic and will report the research results in the future.

7 Conclusions

We have presented a system for detecting and tracking text in digital video automatically. A hybrid wavelet/neural network based method is used to detect text regions. The tracking module uses SSD-based image matching to find an initial position, followed by contour-based stabilization to refine the matched position. The system can detect graphical text and scene text with different font sizes and can track text that undergoes complex motions.

We have also discussed the OCR and indexing problems in video images. Our current focus is on making use of recognized text to build a text-based video indexing and retrieval system. The experiments we have conducted suggest that text enhancement is necessary for reasonable OCR results. Steps should be taken to deal with the poor OCR results and extend their semantic breadth.

References

[1] J.K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Nonrigid motion analysis: Articulated and elastic motion. *Computer Vision and Image Understanding*, 70:142–156, 1998.

[2] E. Bardinet, L. Cohen, and N. Ayache. Tracking medical 3D data with a deformable parametric model. In *Proceedings of ECCV*, pages 317–328, 1996.

[3] E.B Baum and D. Haussler. What size net gives valid generalization? *Neural Computation*, 1:151–160, 1989.

[4] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *Proceedings of CVPR*, pages 232–237, 1998.

[5] M. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proceedings of ECCV*, pages 329–342, 1996.

[6] M. Black and Y. Yacoob. Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion. In *Proceedings of ICCV*, pages 374–381, 1995.

[7] C. Bregler. Learning and recognizing human dynamics in video sequences. In *Proceedings of CVPR*, pages 568–574, 1997.

[8] R. Chellappa, B.S. Manjunath, and T. Simchony. Texture segmentation with neural networks. In *Neural Networks in Signal Processing*, pages 37 – 61. Prentice Hall, 1992.

[9] T.M Cover. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE Trans. Electronic Computers*, 14:326–334, 1965.

[10] Y. Dai, N. Zheng, X. Zhang, and G. Xuan. Automatic recognition of province name on the license plate of moving vehicle. In *Proceedings of ICPR*, pages 927–929, 1988.

[11] D. Dekruger and B.R. Hunt. Image processing and neural networks for recognition of cartographic area features. *Pattern Recognition*, 27:461–483, 1994.

[12] D.S. Doermann. Video performance evaluation. Technical report, University of Maryland, College Park, 1998.

[13] K. Etemad, D. S. Doermann, and R. Chellappa. Multiscale document page segmentation using soft decision integration. *IEEE Trans. PAMI*, 19:92–96, 1997.

[14] P. Fieguth and D. Terzopoulos. Color-based tracking of heads and other mobile objects at video frame rates. In *Proceedings of CVPR*, pages 21–27, 1997.

[15] D.H Foley. Considerations of sample and feature size. *IEEE Trans. Information Theory*, 18:618–626, 1972.

[16] T. Frank, M. Haag, H. Kollnig, and H.H. Nagel. Tracking of occluded vehicles in traffic scenes. In *Proceedings of ECCV*, pages 485–494, 1996.

[17] K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, New York, 1990.

[18] D. Gavrila and L. Davis. Tracking humans in action: A 3D model-based approach. In *Proceedings of Image Understanding Workshop*, pages 737–746, 1996.

[19] T. Gotoh, T. Toriu, S. Sasaki, and M. Yoshida. A flexible vision-based algorithm for a book sorting system. *IEEE Trans. PAMI*, 10:393–399, 1988.

[20] I. Guyon. Applications of neural networks to character recognition. *International Journal of Pattern Rocgnition and Artificial Intelligence*, 5:353–382, 1991.

[21] G.D. Hager and P.N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Trans. PAMI*, 20:1025–1039, 1998.

[22] S. Harmalkar and R.M.K. Sinha. Integrating word level knowledge in text recognition. In *Proceedings of ICPR*, pages 758–760, 1990.

[23] R.C. Harrell, D.C. Slaughter, and P.D. Adsit. A fruit-tracking system for robotic harvesting. *Machine Vision and Applications*, 2:69–80, 1989.

[24] J. Hernando. Voice signal processing and representation techniques for speech recognition in noisy environments. *Signal Processing*, 36:393, 1994.

[25] A.K. Jain and S. Bhattacharjee. Text segmentation using Gabor filters for automatic document processing. *Machine Vision and Applications*, 5:169 – 184, 1992.

[26] A.K. Jain and B. Yu. Automatic text location in images and video frames. In *Proceedings of ICPR*, pages 1497–1499, 1998.

[27] J. P. Jones and L.A. Palmer. The two-dimensional spatial structure of simple receptive fields in the striate cortex of cats. *Journal of Neurophysiology*, 58:1187–1211, 1987.

[28] Hae-Kwang Kim. Efficient automatic text location method and content-based indexing and structuring of video database. *Journal of Visual Communication and Image Representation*, 7:336–344, 1996.

[29] S.K. Kim, D.W. Kim, and H.J. Kim. A recognition of vehicle license plate using a genetic algorithm based segmentation. In *Proceedings of ICIP*, pages 661–664, 1996.

[30] C-M. Lee and A. Kankanhalli. Automatic extraction of characters in complex scene images. *International Journal of Pattern Rocgnition and Artificial Intelligence*, 9:67–82, 1995.

[31] E. Lee, P. K. Kim, and H. J. Kim. Automatic recognition of a car license plate using color image processing. In *Proceedings of ICIP*, pages 301–305, 1994.

[32] P. Letellier, M. Nadler, and J.F. Abramatic. The telesign project. *Proceedings of the IEEE*, 73:813–827, 1985.

[33] H. Li and D.S. Doermann. Automatic identification of text in digital video key frames. In *Proceedings of ICPR*, pages 129–132, 1998.

[34] H. Li, D.S. Doermann, and O. Kia. Text extraction and recognition in digital video. In *Proceedings of DAS*, pages 119–128, 1998.

[35] T.F. Li and S.S. Yu. Handprinted Chinese character recognition using probability distribution feature. *International Journal of Pattern Rocgnition and Artificial Intelligence*, 8:1241–1258, 1994.

[36] R. Lienhart and F. Stuber. Automatic text recognition in digital videos. In *Proceedings of ACM Multimedia*, pages 11–20, 1996.

[37] S. G. Mallat. Multiresolution approximations and wavelet orthonormal bases of $l^2(r)$. *Trans. Amer. Math. Soc.*, 315:69–87, 1989.

[38] S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. PAMI*, 11:674–693, 1989.

[39] B.S. Manjunath and R. Chellappa. A unified approach to boundary perception: edges, textures and illusory contours. *IEEE Trans. Neural Networks*, 4:96 – 108, 1993.

[40] H. Murase and S. Nayar. Visual learning and recognition of 3-D objects from appearance. In *Proceedings of ICCV*, pages 5–24, 1995.

[41] M. A. O'Hair and M. Kabrisky. Recognizing whole words as single symbols. In *Proceedings of ICDAR*, pages 350–358, 1991.

[42] J. Ohya, A. Shio, and S. Akamatsu. Recognizing characters in scene images. *IEEE Trans. PAMI*, 16:214 – 220, 1994.

[43] Y. Pan, J. Wu, S. Tamura, and K. Okazaki. Neural network vowel-recognition jointly using voice features and mouth shape image. *Pattern Recognition*, 24:921–927, 1991.

[44] G. Piccioli, E. De Micheli, P. Parodi, and M. Campani. Robust method for road sign detection and recognition. *Image and Vision Computing*, 14:209–254, 1996.

[45] J. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Proceedings of ECCV*, pages 35–46, 1994.

[46] J. Rehg and A. Witkin. Visual tracking with deformation models. In *Proceedings of IEEE Int'l Conf. on Robotics and Automation*, pages 844–850, 1991.

[47] D. Reynard, A. Wildenberg, A. Blake, and J. Marchant. Learning dynamics of complex motions from image sequences. In *Proceedings of ECCV*, pages 357–368, 1996.

[48] S.K. Rogers, J.M. Colombi, C.E. Martin, and J.C. Gainey. Neural networks for automatic target recognition. *Neural Networks*, 8:1153–1184, 1995.

[49] K. Rohr. Towards model-based recognition of human movements in image sequences. *CVGIP: Image Understanding*, 59:94–115, 1994.

[50] D.W. Ruck, S.K. Rogers, and M. Kabrisky. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2:40–48, 1990.

[51] J. Shi and C. Tomasi. Good features to track. In *Proceedings of CVPR*, pages 593–600, 1994.

[52] P. Shi, G.Robinson, T.Constable, A. Sinusas, and J. Duncan. A model-based integrated approach to track myocardial deformation using displacement and velocity constraints. In *Proceedings of ICCV*, pages 687–692, 1995.

[53] J. Shim, C. Dorai, and R. Bolle. Automatic text extraction from video for content-based annotation and retrieval. In *Proceedings of ICPR*, pages 618–620, 1998.

[54] K. Sung and T. Poggio. Example-based learning for view-based human face detection. Technical report, MIT, A.I. Memo 1521, CBCL Paper 112, 1994.

[55] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Trans. PAMI*, 19:780–785, 1997.

[56] V. Wu, R. Manmatha, and E.M. Riseman. Automatic text detection and recognition. In *Proceedings of Image Understanding Workshop*, pages 707–712, 1997.

[57] B. Yu, A.K. Jain, and M. Modiuddin. Address block location on complex mail pieces. In *Proceedings of ICDAR*, pages 897–901, 1997.

[58] Y. Zhong, K. Karu, and A.K. Jain. Locating text in complex color images. *Pattern Recognition*, 28:1523–1236, 1995.

[59] J. Zhou and D. Lopresti. Extracting text from WWW images. In *Proceedings of ICDAR*, pages 248–252, 1997.

[60] J. Zhou, D. Lopresti, and T. Tasdizen. Finding text in color images. In *Proceedings of SPIE, Document Recognition V*, pages 130–140, 1998.