Text Extraction, Enhancement and OCR in Digital Video

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

[1] Language and Media Processing Laboratory
Institute for Advanced Computer Studies, University of Maryland
College Park, MD 20742-3275
{huiping,doermann}@cfar.umd.edu
[2] Advanced Network Technologies Division
National Institute of Standards and Technology
Gaithersburg, MD 20899
omid.kia@nist.gov

Abstract. In this paper we address the problem of text extraction, enhancement and recognition in digital video. Compared with optical character recognition (OCR) from document images, text extraction and recognition in digital video presents several new challenges. First, the text in video is often embedded in complex backgrounds, making text extraction and separation difficult. Second, image data contained in video frames is often digitized and/or subsampled at a much lower resolution than is typical for document images. As a result, most commercial OCR software can not recognize text extracted from video. We have implemented a hybrid wavelet/neural network segmenter to extract text regions and use a two stage enhancement scheme prior to recognition. First, we use Shannon interpolation to raise the image resolution, and second we postprocess the block with normal/inverse text classification and adaptive thresholding. Experimental results show that our text extraction scheme can extract both scene text and graphical text robustly and reasonable OCR results are achieved after enhancement.

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. Text can provide important supplemental index information in video sequences. Examples may include sports scores, product names, scene locations, speaker names, movie credits, program introductions and special announcements. If text can be extracted and recognized robustly, we may, for example, submit queries such as *"Bruce Willis"* and retrieve a list of all movies featuring him, or "stock news" to retrieve relevant financial reports.

Text extraction and recognition from digital video presents several challenges. First, the text is usually embedded in complex backgrounds, making extraction

S.-W. Lee and Y. Nakano (Eds.): DAS'98, LNCS 1655, pp. 363–377, 1999.

and recognition difficult. Second, the video image is usually digitized or subsampled at an extremely low resolution and as a result, text can not be recognized by most commercial OCR software. For document images, 300 *dpi* is commonplace and normal characters (12 points) occupy an area as large as 40×40 pixels. Video frames are often digitized at 352×240 pixels with text rendered as small as 10×10 pixels, resulting in no output from OCR software, even though text is clearly human readable.

1.1 Related Work

Some work on the extraction of text from road signs [1], license plates [2], library books [3], *WWW* images [4] and isolated video frames [5] has been reported in the literature. The methods can be broadly classified into two types: connected component (*CC*) based and texture based. Scene images and video frames are usually recorded in multivalued (gray-scale or colored) form. For *CC* based approaches, color clustering [6,7] or binarization [8,9] are usually used to decompose the multivalued image into several elementary images in which all the pixels share the same color or intensity value. Connected components are then extracted from each decomposed image and heuristic restrictions on component size, number of aligned components and line orientation are used to identify text lines. The second approach is texture-based and uses well-known texture analysis methods such as Gabor filtering [10], Gaussian filtering [11] or spatial variance [12] to locate text regions. In [10], Jain describes a method of separating text and image areas based on a group of multichannel Gabor filters.

Previous work on text enhancement has focused primarily on binary document images with black pixels representing text and white pixels representing background. Hobby presents a method to enhance degraded document images via bitmap clustering and averaging for better display quality and recognition accuracy [13]. OCR accuracy is improved from 6% to 38% for documents with varying quality. Liang [14] addresses the problem of document image restoration using morphological filters and achieves nearly 80% OCR accuracy for subtractive and additive noise images.

Work on text recognition in scene images and digital video is reported in [5,11,15] and [16]. Wu and Manmatha describe a text extraction and recognition system and achieve 84% correct OCR rates based only on "*OCRable*" text [11]. Lienhart describes a text recognition system in digital video and achieves a recognition result of nearly 80% [5]. Shim and Dorai present a text extraction system in video sequences. The output of the system is OCR-ready bitmaps but text recognition problem is not addressed [15]. Zhou and Lopresti describe their work on text extraction and recognition from *WWW* images [4,16]. None of these systems, however, perform text enhancement and all rely on the text having "substantial" resolution.

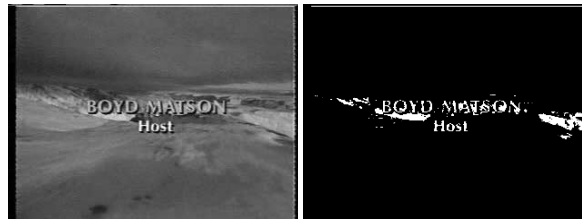
Our goal is to develop an algorithm to detect both scene text and graphical text in video and to use text enhancement to achieve reasonable OCR accuracy. We use a hybrid wavelet/neural network segmenter to detect text regions.

After text detection, Shannon interpolation is used to increase the image resolution. Postprocessing including normal/inverse text classification and adaptive thresholding are applied to generate OCRable text.

The rest of the paper is organized as follows. In Section 2 we address our text detection scheme in detail. Text enhancement is described in Section 3 and postprocessing is described in Section 4. Experimental results are presented in Section 5 and finally a brief discussion is given in Section 6.

2 Text Detection

Text in digital video is typically overlayed on complex backgrounds. As a result, methods based on connected component analysis usually fail since the text often touches graphical objects after binarization or color segmentation. Figure 1a shows a typical video frame and Figure 1b is the binarized version using an ideal threshold. The binarized image shows connectedness between character components and the background.



(a) (b)

Fig. 1. (a) A video frame, (b) binarization by manually picking up an ideal threshold.

Our approach uses a small window (typically 16×16) to scan the image and classify each window text or non-text. Two important issues we need to consider are the feature extraction and the choice of classifier.

There are types of classifiers which are commonly used. The first is the traditional classifier such as linear discriminant analysis (LDA), maximum likelihood, k-nearest neighbors. The other is the neural network based classifier. In video frames, natural scenes like the leaves of a tree or grass in a field have textures similar to text. As a result, text and nontext often overlap in the feature space. Since the text and nontext are not linearly separable, a neural network classifier's adaptive learning offers an attractive and computationally efficient alternative for classification.

Our feature extraction 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. The feature extraction and selection scheme is described in the next section.

2.1 Feature Extraction and Selection

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 down-sampling and have the ability to detect edges during the high-pass filtering. The low-pass filter creates successive approximations to the image while the detailed signal provides a feature-rich representation of textual content [17]. This is easily seen in the image decomposition shown in Figure 2 where 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.

(a) (b)

Fig. 2. The single-level wavelet decomposition of a video frame: (a) Original image, (b) The first level decomposed images.

We use the mean and the second and third-order central moments as features. For an $N \times N$ subblock I we calculate the mean (m), the second-order (μ_2) and third-order (μ_3) central moments as:

$$\begin{aligned} M(I) &= \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} I(i,j) \\ \mu_2(I) &= \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I(i,j) - M(I))^2 \\ \mu_3(I) &= \frac{1}{N^2} \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (I(i,j) - M(I))^3 \end{aligned} \tag{1}$$

All of features are computed on the decomposed subband images. Since the original window size is 16×16, the maximum decomposition level we could choose is four, with only one pixel left for each subband image in the fourth level. Therefore, the features described in Equation 1 are calculated only in the subblocks of the first three levels. There are 36 features corresponding to

each 16 × 16 window. We conduct feature saliency analysis to reduce feature set since a larger feature set requires more training samples and time.

We collected 1000 text blocks and 1000 nontext blocks and split them training and testing sets. Each feature was trained on the training set and then applied to classify the testing data. We use the Bayes error rate P_e to analyze the saliency of features since it determines 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 labeled samples from the various classes. This is typically done by estimating a posteriori probability of each class for each sample, then assigning each sample to the class with the *MAP* (maximum a posteriori) probability. The percentage of samples misclassified by applying the *MAP* decision rule to the posteriori estimates is taken as an estimate of the Bayes error rate. Finally, eight features which have the lowest Bayes error rate are selected from original 36 features and are fed to the neural network.

2.2 Training the Neural Network

After selecting the features, we train the neural network. The neural network consists of 8 input, 12 hidden and 1 output node. 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 handle this problem we use a *bootstrap* method recommended by Sung and Poggio [18] to train the neural network. The idea is that the training samples are collected in part during training rather than before training:

1. Create an initial set of training samples which includes a complete set of text samples and a partial set 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 Step 2 and 3 until the accuracy converges.

2.3 Classification

After training, we use the neural network to classify each block as text or nontext. We can view the output of the neural network as a mapping function which maps each feature set into a real value between 0 and 1. Figure 3a shows the distribution of the neural network outputs. A threshold of 0.5 is a reasonable choice to indicate whether the window contains text or not.

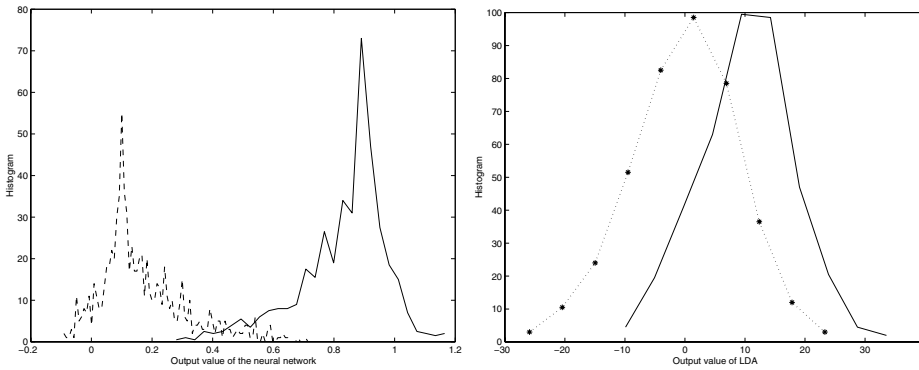
In order to compare the performance of different classifiers, we used *LDA* [19] to conduct the classification on the same data. Suppose we have C classes with class *means* M_i, $i = 1, 2, ...C$. Then the within-class scatter matrix S_w and between-class scatter matrix S_b can be defined as

$$S_w = \sum_{i=1}^{C} \sum_{j=1}^{N_i} (X_{ij} - M_i)(X_{ij} - M_i)^T$$
$$S_b = \sum_{i=1}^{C} (M_i - M)(M_i - M)^T \tag{2}$$

where X_{ij} is the jth sample vector in class i, N_i is the number of samples in class i and M is the grand *mean* of all sample vectors. The projection matrix P is chosen to maximize $\frac{det(S_b)}{det(S_w)}$. It has been shown that the ratio is maximized when the columns of P are the eigenvectors of $T = S_w^{-1} S_b$ associated with the largest eigenvalues. In our case P consists of one eigenvector corresponding to the largest eigenvalue of T since the first eigenvalue $(4.8317e-04)$ is much bigger than the second largest one (4.5114e-18). Correspondingly, each 16×16 window is mapped to one value:

$$Y = P^T X \tag{3}$$

Figure 3a is the neural network result and 3b is the *LDA* result. In both cases the solid line represents text and the dash line represents nontext. We can see the data in Figure 3a is more easily separable than in 3b, which suggests the neural network has a better performance than *LDA*.



(a) (b)

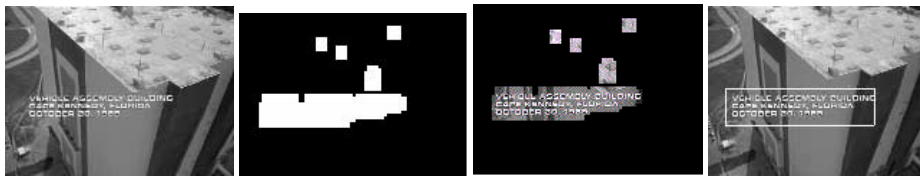
Fig. 3. The comparison of the output distribution when we use trained neural network and LDA for the testing text and nontext blocks. In both cases the solid line represents text and the dash line represents nontext. (a) The distribution of neural network output, (b) The distribution of *LDA* output.

2.4 Text Detection

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 window step,

the fewer the number of windows to be processed but the less refined the result. Considering the trade-off between the precision and speed, we move the window 4 pixels at a time.

If a single 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 as nontext. The result of classification is a label map of the original image. Figure 4(a) is a video frame and Figure 4(b) is the classified label map corresponding to Figure 4(a). Figure 4(c) shows the extracted text regions. We can see all of the text is labeled correctly, but there are some small isolated areas which are incorrectly labeled as text. We use size constraints between blocks to filter out these areas. The bounding box of the text area is generated by a connected component analysis of the text windows. Figure 4(d) is the result after we filter out the non-text areas and generate the bounding box.



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

Fig. 4. (a) Original frame, (b) The classified label map, (c) The segmented text area corresponding to (b), (d) The segmented text area after postprocessing and bounding box generation.

3 Text Resolution Enhancement

Video images are limited in spatial resolution. In typical document images, 300 *dpi* is common with characters occupying an area as large as 40×40 pixels. In *MPEG-1* each frame is digitized and subsampled to 352×240 pixels with text rendered as small as 10×10 pixels. Text in this resolution is readable by humans but may not be sufficient for computers to recognize. Image resolution enhancement is necessary to improve the OCR results.

In digital video, a lowpass filter is often applied to filter the high resolution image before subsampling in order to eliminate aliasing. Suppose the original high resolution image is I_0 and the available low resolution image is I. We can model I as the image obtained from lowpass filtering a high resolution image I_0 followed by downsampling (Figure 5a). Since the high frequency information has been lost, it is impossible to recover the original image I_0 from I. If we assume that no information about the high frequency components is available, the optimal interpolation scheme which yields the least mean squared error is upsampling the image I followed by passing the resulting image through a low-

pass filter (Figure 5b). It is easy to show that the resulting image I' has the same frequency content as I'_0.

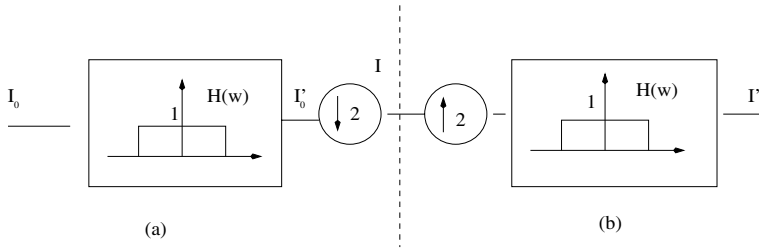


Fig. 5. Problem models. (a) Model the available low resolution image I as the image obtained from lowpass filtering a higher resolution image I_0 followed by downsampling. (b) Assuming that we have no information about the higher frequency component, the optimal interpolation scheme is upsampling followed by lowpass filtering.

The scheme can be implemented as an extension of the Nyquist sampling theorem where a sampled image is a weighted sum of delayed *Sinc* functions (Equation 4). Inter-sample values are then the sum of the *Sinc* functions at their non-zero crossings. This process is computationally intensive but we can pursue a frequency-based approach. The dual of the weighted *Sinc* functions can be performed by Fast Fourier Transforms (FFT) and matrix masking. We increase the resolution of our images by copying each pixel to neighboring pixels in the amount of the desired increase in resolution. We then take the two dimensional FFT. The resulting matrix is then multiplied by a mask matrix which zeros in the high frequency components. The number of low frequency components that are preserved is equal to the size of the original image. This in effect is a low pass filter where high frequency components introduced by copying pixels to neighboring pixels are removed. An inverse two dimensional FFT renders the image in the higher resolution. Figure 6b shows the result of the increase in resolution of Figure 6a by a factor of four. This task does not require a lot of processing and can be easily encoded in hardware.

$$i(X,Y) = \sum_{x=1}^{N}\sum_{y=1}^{M} i(x,y)sinc(X-x, Y-y), \text{ for } X \in \Re(1,N) \text{ and } Y \in \Re(1,M) \tag{4}$$

4 Postprocessing

4.1 Identification of Inverse Text

The classification of normal text and inverse text is necessary since typical OCR software can only handle binary images with black pixels representing text and

(a) (b)

Fig. 6. (a) Anti-aliased image showing blurred edges, (b) Zoomed image by Shannon's interpolation.

white pixels representing background. The scheme we use is simple: First, we calculate the global thresholding value **Th**. The background is typically defined as the maximum part of the histogram of the image, so we can make our decision by comparing the threshold **Th** and the background value **Bg** which corresponds to the maximum part of the histogram. Specifically, if we find **Th** > **Bg**, then it is normal text, otherwise, it is inverse text.

4.2 Adaptive Thresholding

Due to the large variations in font size and contrast with the background, adaptive thresholding techniques are usually required to binarize text blocks. We use modified lit20's adaptive thresholding method [20] to threshold text images. The basic idea of lit20's method is to vary a threshold over the image, based on the local mean m and local standard deviation d computed in a small neighborhood of each pixel. A threshold for each pixel is computed from $T = d + k * s$, where k is a user defined parameter. The result of this method is not good for background containing light textures.

We incorporate a simple classification scheme into lit20's algorithm to improve both the binarization result and speed. Considering that text usually has significant contrast with the background and therefore has a larger standard deviation d than background areas, we can set all pixels in a window as background if the standard deviation $d < th$. We collected over 600 text blocks to determine the proper value for th. As a result, both the binarization result and speed are improved (Figure 7).

5 Experimental Results

5.1 Evaluation of Text Detection

We collected two sets of data for experiments. The first set of data included 500 key frames selected automatically from *MPEG* video using key frame detection algorithm [21]. The second data set included 75 frames selected manually from

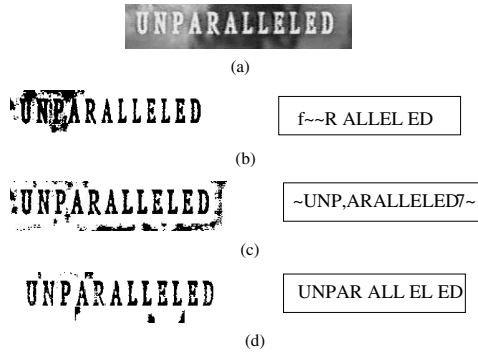


Fig. 7. (a) A gray-scale text block, (b) Global Thresholding and OCR result, (c) lit20's method and OCR result, (d) Our method and OCR result.

cable TV by using *miro VIDEO DC30 plus* video capture board. The samples include both scene text and graphic text with multiple font sizes.

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

Evaluation of Text Event Detection Text event detection checks if a video frame contains text or not and is useful in video indexing and retrieval as well as video classification. We output 1 if a frame contains text and 0 if the frame does not contain text. In 500 video frames, 151 of them contain text and the remaining 349 frames do not contain text (Table 1).

	text	nontext
text (151)	133 (88%)	18(12%)
nontext (349)	81 (23%)	268 (77%)

Table 1. The confusion matrix for text event detection

We use two metrics (*precision* and *recall*) commonly used in information retrieval (IR) to evaluate text event detection:

$$\begin{aligned} precision &= \frac{\text{correctly detected text frames}}{\text{totally returned text frames}} \\ recall &= \frac{\text{correctly detected text frames}}{\text{total text frames in data set}} \end{aligned} \tag{5}$$

Therefore, the *recall* is $\frac{133}{151} = 88\%$ and the *precision* is $\frac{133}{133+81} = 62\%$. *Precision* is relatively low in part because the number of non-text frames is more than that of text frames.

Evaluation of Text Block Detection The second data set is used to evaluate the extraction of text blocks. A text block may contain one or more text lines which are close to each other. There are a total of 153 text blocks in the 75 frames. 142 (93%) of them were correctly detected by our algorithm (Figure 8) and 11 (7%) of them were missed. Errors occur primarily because of low resolution or small text block size. On the other hand, 14 non-text blocks are misclassified as text blocks. Further training, domain-specific training, or attempting OCR will overcome these problems.

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

Fig. 8. Correct results. (a) Reverse text with large font size, (b) Text with low resolution (bottom) and scene text (text on T-shirt of athlete), (c) Text with different font sizes, (d) Text with different font styles.

5.2 Evaluation of Text Enhancement

In this section we describe the experimental setup for enhancement based on OCR. It should be noted that we do not intend to implement our OCR system since there exists many commercial OCR packages. The OCR software we used in our experiment is Xerox TextBridge Pro98. After binarization of text image, we manually feed the image to TextBridge Pro98 for recognition.

Evaluation Metrics The evaluation software we use for OCR is *ope* provided by the University of Washington [22]. *Ope* performs string matching between the ground truth data and the recognized data by counting the amount of character substitutions, insertions, and deletions. The outputs of *ope* have the following parameters:
T : character number of ground truth;
O : character number from OCR output;
R : correctly recognized character number;
D : number of deletion operation;
I : number of insertion operation ;
S : number of substitution operation;

We use the following metrics to evaluate the OCR results:

- *Accuracy* (or *Recall*). Accuracy can be defined as the ratio of correctly recognized character number **R** to character number of ground truth **T** ($accuracy = \frac{R}{T}$).
- *Precision*. Precision is defined as the ratio of correctly recognized character number **R** to character number from OCR output **O** ($Precision = \frac{R}{O}$).
- *Cost*. Cost is defined as the weighted sum of number of deletion (**D**), insertion (**I**) and substitution (**S**) ($cost = D + I + S$). Here we suppose each operation has the same weight.

Experiments In order to eliminate the effect text detection algorithm may introduce, we collected 45 correctly detected text blocks from our text detection experiments for enhancement experiments. We generated another 45 text blocks by Shannon interpolation to magnify the image in the factor of two. In order to observe how our interpolation scheme can improve the OCR results, we generated another group of text blocks by simply copying each pixels in the image to four pixels (Zero order hold interpolation). Both the interpolation schemes increase the resolution by a factor of two. The same text separation scheme mentioned above is applied to all the original text blocks and the interpolated text blocks.

	No Enhancement	Zero order hold Interpolation	Shannon Interpolation
# of Total Blocks	45	45	45
# of Blocks Having OCR Output	13(29%)	36(80%)	45(100%)
# of Ground Truth Characters (T)	1452	1452	1452
# of OCR Output Symbols (O)	308	1032	1341
# of Matched Symbols (R)	188	489	970
# of Deletion Operation (D)	1164	651	241
# of Insertion Operation (I)	20	231	111
# of Substitution Operation (S)	100	312	269
Accuracy $\frac{R}{T}$	13%	34%	66.8%
Precision $\frac{R}{O}$	61%	47%	72%
Cost $D + I + S$	1284	1194	621

Table 2. The Performance evaluation of OCR result with no enhancement, zero order hold and Shannon interpolation (interpolation factor is 2).

As shown in Table 2, only 13(29%) text blocks have OCR output before enhancement, but a significant improvement is achieved even with Zero order hold interpolation (36(80%)). All 45 text blocks have OCR output for Shannon interpolation. The difference here tells us the resolution of most text blocks is beyond the machine recognition capability if no resolution enhancement is performed.

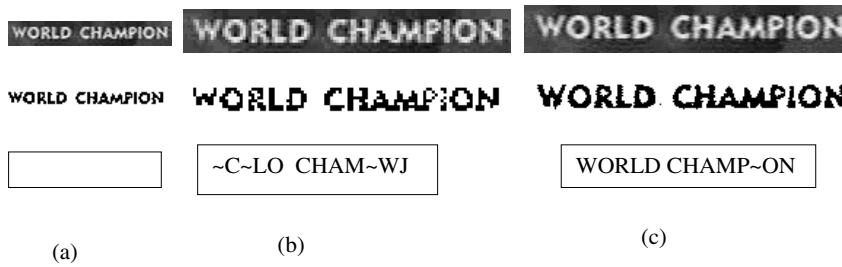


Fig. 9. Comparison of OCR results. (a) No enhancement, (b) Zero order hold interpolation, (c) Shannon interpolation.

There are a total of 1452 characters in the 45 text blocks. Before enhancement, only 188(13%) are correctly OCRed. This is insufficient for any successful indexing operation. For Zero order hold interpolation, the OCR accuracy rate rises to 34% and we observe 66.8% correct rate for Shannon interpolation. We also can see Shannon interpolation has the best precision (72%). The precision of no interpolation is better than Zero order hold interpolation because there are few character outputs in the case of no interpolation. The *Cost* of Shannon interpolation is much smaller than Zero order hold interpolation or no interpolation. Figure 9 shows an example of these three cases. We can see there is no OCR output for original image (Figure 9a), and although Zero order hold interpolation has OCR output, there are considerable recognition errors (Figure 9b). Figure 9c shows the recognition result for Shannon interpolation.

In order to investigate the relation between the interpolation factor and OCR accuracy, we interpolate the image by factor of 4 and 8, respectively for Shannon interpolation. The output of *ope* is shown in Table 3. We can see an increase in image resolution does not necessarily improve OCR accuracy. OCR accuracy at factor 4 is slightly better than that at factor 2. However, the performance begins to decrease at factor 8. Note that the average character sizes are 40×40 for interpolated image at factor 4, which corresponds to 300 *dpi* in typical document images. Consequently, character sizes at factor 2 and 8 correspond to 200 *dpi* and 600 *dpi* respectively. To most commercial OCR software, documents scanned with 300 *dpi* achieve the best OCR results [23].

Although these results are far from perfect, we do not discriminate whether text blocks are OCRable or not (Figure 10). It is very encouraging that we observed over 500% improvement.

6 Discussion and Conclusions

We have presented a hybrid wavelet/neural network method to detect both graphical and scene text in digital video and a text enhancement scheme to improve OCR accuracy in digital video. The text detection algorithm can detect 93% of text blocks. An overall 67% OCR accuracy rate is achieved after enhancement compared with only 13% accuracy rate with no enhancement.

Interpolation Factor	2	4	8
# of Ground Truth Characters (T)	1452	1452	1452
# of OCR Output Symbols (O)	1341	1402	1336
# of Matched Symbols (R)	970	977	892
# of Deletion Operation (D)	241	239	248
# of Insertion Operation (I)	111	120	105
# of Substitution Operation (S)	269	250	279
Accuracy $\frac{R}{T}$	66.8%	67.3%	61.4%
Precision $\frac{R}{O}$	72%	70%	67%
Cost $D + I + S$	621	609	632

Table 3. The Performance evaluation of OCR results with different interpolation factors

Fig. 10. Even with enhancement, there is still no OCR output due to poor image quality.

The experiments on text recognition raise several interesting challenges in text-based indexing of digital video. First, we expect to find missing or incorrectly segmented characters and only partial OCR results. As a result, exact matches between words will not always be possible. *Approximate word matching* instead of *exact word matching* is a possible aid. Second, text in digital video is usually very terse and may lack semantic breadth. *Wordnet* can be used to extend semantic connections. Our next goal is to build a text-based indexing system in digital video database.

References

1. 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. 364
2. 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. 364
3. 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, 1998. 364
4. J. Zhou, D. Lopresti, and T. Tasdizen. Finding text in color images. In *Proceedings of SPIE, Document Recognition V,* pages 130-140, 1998. 364

5. R. Lienhart and F. Stuber. Automatic text recognition in digital videos. In *Proceedings of ACM Multimedia,* pages 11-20, 1996. 364
6. A.K. Jain and B. Yu. Automatic text location in images and video frames. In *Proceedings of ICPR,* pages 1497-1499, 1998. 364
7. 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. 364
8. 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. 364
9. J. Ohya, A. Shio, and S. Akamatsu. Recognizing characters in scene images. *IEEE Trans. PAMI,* 16:214 - 220, 1994. 364
10. A.K. Jain and S. Bhattacharjee. Text segmentation using Gabor niters for automatic document processing. *Machine Vision and Applications,* 5:169 - 184, 1992. 364
11. V. Wu, R. Manmatha, and E.M. Riseman. Automatic text detection and recognition. pages 707-712. 5 1997. 364
12. Y. Zhong, K. Karu, and A.K. Jain. Locating text in complex color images. *Pattern Recognition,* 28:1523-1236, 1995. 364
13. John D. Hobby and Tin K. Ho. Enhancing degraded document images via bitmap clustering and averaging. In *ICDAR'97: Fourth International Conference on Document Analysis and Recogntion,* pages 394-400, August 1997. 364
14. J. Liang and R.M. Haralick. Document image restoration using binary morphological filters. In *SPIE Vol. 2660,* 1996. 364
15. 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. 364
16. J. Zhou and D. Lopresti. Ocr for world wide web images. In *Proceedings of SPIE, Document Recognition IV,* pages 58-66, 1997. 364
17. S. G. Mallat. A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. PAMI,* 11:674-693, 1989. 366
18. 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. 367
19. K. Fukunaga. *Introduction to Statistical Pattern Recognition.* Academic Press, New York, 1990. 367
20. Niblack W. In *An introduction to image processing,* pages 115-116, Englewood Cliffs, N.J.:Prentice Hall, 1986. 371
21. V. Kobia, D.S. Doermann, and K.I. Lin. Archiving, indexing, and retrieval of video in the compressed domain. In *Proc. of the SPIE Conference on Multimedia Storage and Archiving Systems,* volume 2916, pages 78-89, 1996. 371
22. S. Chen. OCR performance evaluation software - user's manual. In *The University of Washington Database.* 373
23. T. Kanungo, G.A. Marton, and O. Bulbul. Omnipage vs. sakhr: Paired model evaluation of two arable ocr products. In *Proc. of the SPIE Conference on Document Recognition and Retrieval (VI),* volume 3651, 1999. 375