1

Video OCR: A Survey and Practitioner's Guide

Rainer Lienhart
Intel Labs, Intel Corporation
Santa Clara, California, USA
Rainer.Lienhart@intel.com

1 INTRODUCTION

This survey strives to present the core concepts underlying the different *texture-based* approaches to automatic detection, segmentation and recognition of visual text occurrences in complex images and videos. It emphasizes the different approaches to attack the many issues in this space. For each kind of approach only a few representative references are given. This survey does not try to give an exhaustive listing of all relevant work, but to help practitioners and engineers new in the field to get a thorough overview of the state-of-the-art principles, methods, and systems in Video OCR. Hereto, the approaches of the various researchers are broken up into its constituents and presented as a design choice in a hypothetical image and video OCR system.

Sometimes video text detection algorithms are classified by whether the originally proposed detection algorithm was designed to operate on uncompressed or compressed video streams. In this survey, we do not make this distinction, since almost all texture-based detection algorithms can be applied to the compressed as well as to the uncompressed domain. The issue of compressed versus uncompressed processing is orthogonal to the task of Video OCR. It only determines the space in which to perform the text texture detection. Text segmentation is usually performed in the uncompressed domain.

From a bird's eye view, the task of detecting, segmenting and recognizing text visually appearing in complex images and/or video seems to be well defined. However, many design decisions have to be taken based on the overall goal.

Typical design choices are:

- *What kind of text occurrences should be considered?*
  Based on its origin there exist two different kinds of text in videos and images [13][14]. ***Scene text*** is text that was recorded as part of scene such as street names, shop names, and text on T-shirts. It mostly appears accidentally and is seldom intended. Due to its

incidental and the thus resulting unlimited variety of its appearance, it is hard to detect, extract and recognize. It can appear with any slant, tilt, in any lighting and upon straight or wavy surfaces. It may also be partially occluded.

In contrast, the appearance of ***overlay text*** is carefully directed. It is often an important carrier of information and herewith suitable for indexing and retrieval. For instance, embedded captions in TV programs represent a highly condensed form of key information on the content of the video [30]; in commercials, the product and company name are often part of the text shown. Here, the product name is often scene text but used like artificial text. Most research work concentrates on artificial text occurrences, where it is implicitly implied that text lies in a plane roughly perpendicular to the optical axis of the camera. Only little work can be found on scene text [20][21][3][4].

A different classification scheme of text occurrences is based on the constraints in the placement of planar text in the 3D space. Typical classes with increasing degree of freedom are

(1) horizontal overlay text in the plane parallel to the camera plane ($\theta = 0, \varphi = 0, \gamma = 0$) (see Figure 1(a)),

(2) planar overlay text in the plane parallel to the camera plane ($\theta = any, \varphi = 0, \gamma = 0$) (see Figure 1(b)),

(3) Unconstrained planar 3D text ($\theta = any, \varphi = any, \gamma = any$) (see Figure 1(c)), and

(4) Unconstrained 3D text ($\theta = any, \varphi = any, \gamma = any$, text on any surface).

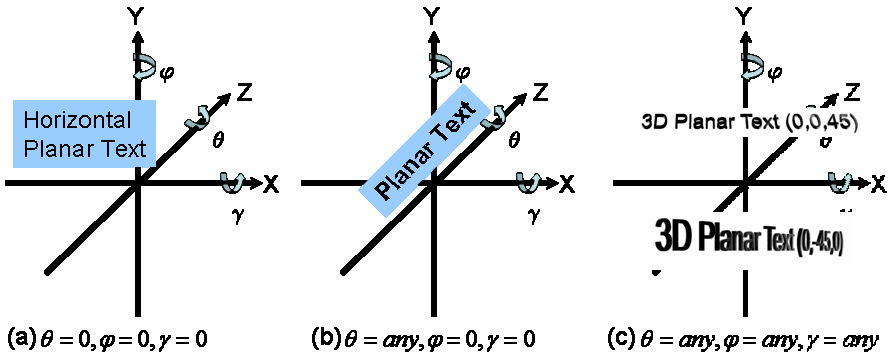

Figure 1: Different degrees of freedom in the placement of planar text in the 3D space.

- *With what font attributes?*
  Text occurrences can differ significantly in font size, type, style, and colour. Some research work has been tailored to very specific domains with limited variations in these attributes. For instance, the Video OCR module in the Informedia project is tailored to CNN Headline News video encoded in MPEG-1 [24][25]. It explicitly

exploits the domain knowledge about the tight restrictions in font attributes such as font types and font sizes. Other Video OCR systems avoid any attribute restrictions [16][29]. Text can be of any size, type, style and colour.

- *In what kind of media data?*
  Should the underlying text detection, segmentation and recognition approach be image-based (i.e., treating a video as a set of independent images) or should it exploit the fact that the same text line occurs in videos for some time and that, therefore, the multiple instances of the same text line can be utilized to achieve better detection, segmentation and recognition performance.

- *How will the output of the Video OCR system be used?*
  Different usages have different levels of tolerance against errors. For instance, if the Video OCR output is only used for image/video indexing based on the transcribed text, pixel errors in the localization and segmentation steps as well as recognition errors can be tolerated and compensated. If, however, the output is used for object-based video encoding, the system must minimize the errors in pixel classification. The system in [16], for example, was explicitly designed to label each pixel in a video as whether it belongs to text or not. This information was used to encode text occurrences then as a high-fidelity foreground MPEG-4 video objects (VOPs) at low frame per second (fps) rate, while re-colouring the pixels behind the text pixels with colours efficient for compression. A gain of about 1.5dB in PSNR at low bit rates were reported (see Figure 2).

  Other usage scenarios are the visual removal of text from videos and the automatic translation of detected text from one language into another language.

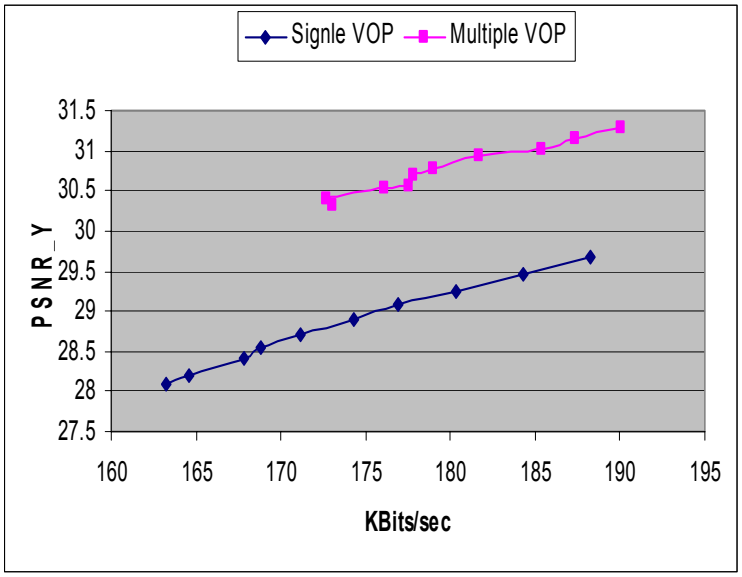

Figure 2: PSNR comparison of same video encoded as a single VOP MPEG-4 video and a multiple VOP MPEG-4 video with one additional VOP for each detected text line.

This paper focuses on texture-based Video OCR algorithms. It does not address the many connected-component-based approaches such as [13][14][15][26]. This choice was purely made to present the vast research in this field in a more structured way. In no sense it should be understood as a judgement of the connected-component-based approach. In fact, connected-component based approaches are working surprisingly well and successful in practice and can compete in performance with the best texture-based approaches.

The remainder of this paper is organized at follows. In Second 2 we address texture-based text detection -- the task of finding the locations of text occurrences in images and videos. Starting with general observations about text, a set of suitable texture features are listed in Subsection 2.1. Then, Subsection 2.2 details how to use the texture features to achieve image-based text detection. For video specialized extensions exist to further improve text detection performance. They are explained in Subsection 2.3. An overview of common performance measures and the performance of existing systems is given in Section 2.4. Section 3 addresses text segmentation – the task of preparing bitmaps of localized text occurrences for optical character recognition (OCR). It is subdivided into three subsections. Subsection 3.1 and 3.2 discuss pre-processing steps helping to improve text segmentation performance. The former subsection focuses on approaches in the image domain, while the latter investigates the unique possibilities with videos. Finally Subsection 3.3 introduces the segmentation algorithms. An overview of common performance measures and the performance of existing text segmentation systems is given in Subsection 3.4. Section 4 concludes the paper with a summary and outlook.

2 Detection

Text detection is the task of finding the locations of text occurrences in images and videos. Dependent on the subsequent task the circumscribing shapes of the text locations either comprise whole text columns or individual text lines. For planar text occurrences in the plane parallel to the camera plane the circumscribing shape is a rectangle, while for scene text it usually is a rotated parallelogram ignoring the foreshortening under fully perspective projection.

Text detection has many applications. It is the prerequisite for text segmentation. It, however, can also be used to rectify documents captured with still image or wearable cameras for improved readability. In most cases without rectification the text would exhibit significant perspective distortions.

2.1 Text Features

2.1.1 General Observations

Humans can quickly identify text regions without having to search for individual characters. Even text too far to be legible can easily be identified as such. This is due to the stationary pattern text lines and text columns exhibit at different scales.

In Roman languages text regions consist of text lines of the same orientation with roughly the same spacing in between. Each text line is composed of characters of approximately the same size, placed next to each other. A text line contrasting with the background shows a large intensity variation vertically to the writing direction as well as horizontally at its upper and lower boundaries.

The mainstream of overlay text in Roman languages is characterized by the following features [13][14]. Only a few exceptions may be observed in practice:

- Characters are in the foreground. They are never partially occluded.
- Characters are monochrome.
- Characters are rigid. They do not change their shape, size or orientation from frame to frame.
- Characters have size restrictions. A letter is not as large as the whole frame. Nor are letters smaller than a certain number of pixels as they would otherwise be illegible to viewers.
- Characters are mostly upright.
- Characters are either stationary or linearly moving. Moving characters also have a dominant translation direction: horizontally from right to left or vertically from bottom to top.
- Characters contrast with their background since artificial text is designed to be read easily.
- The same characters appear in multiple consecutive frames.
- Characters appear in clusters at a limited distance aligned to a virtual line. Most of the time the orientation of this virtual lines is horizontal since that is the natural writing direction.

Most of these features also hold for ***non-Roman languages***, but some need to be adapted to the characteristics of the particular language system. For instance, the minimal readable font size of Roman languages is about 7 to 8 pt. In contrast, Chinese characters due to their complex structure require at least twice the size. In Roman languages meaningful words are built from multiple characters. Therefore, a semantically meaningful text line should be composed of at least three or more characters. In Chinese, however, each character has a meaning voiding this constraint. Roman languages are most readable with justified characters. Justified text lines in turn result in homogenous stroke densities that can easily be detected. Chinese characters, in contrast, have a fixed block size letting its spatial stroke density vary significantly. Every character occupies the same space. At the same time the number of strokes can vary from 1 to 20 [2]. Some texture-based features might therefore not be applicable to Chinese character detection.

In this survey we will concentrate on texture-based approaches for Roman languages. Other language system as well as its dual approach, text detection and text segmentation based on connected component analysis, will not be addressed here.

All approaches should keep the following general challenges in mind:

- The contrast of text in complex backgrounds may vary in different areas of the image. Complex background usually requires strong contrast to make text still readable, while for simple background even a small contrast is sufficient [2].
- The colour of text is not uniform due to colour bleeding, noise, compression artefacts, and applied anti-aliasing. Colour homogeneity should therefore not be strictly assumed [14][18].

2.1.2 Texture-based Features

Text exhibits unique features at many scales. Researchers have developed many statistical features based on the local neighbourhood to capture certain texture aspects of text. Some features operate at different text scales and are designed to identify individual text lines, while others measure certain attributes of text paragraphs. In this subsection, the most important features are listed. None of them will uniquely identify text regions. Each individual feature will still confuse text with non-text areas, but models one or several important aspects of text versus non-text regions. A society of features will complement each other and allow identifying text unambiguously.

Gray Levels of Raw Pixels

Shin et al. suggest the use of grey levels of raw pixels as features. The input feature vector size is reduced by taking only a structured subset of all pixels in a neighbourhood. For instance, they suggest the use of a star pattern mask as shown in Figure 3 [27].

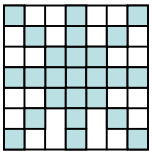

Figure 3: Star-like pixel pattern

Local Variance

The observed local variance in text regions depends on the scale. For small and medium text medium values are expected, since text in such areas undergoes aliasing at the boundaries. Very high variance region indicate single sharp edges and not text. In [3] a circular disk filter S of radius 3 is applied to measure local variance V:

$$V = S \bullet \left(I - S \bullet I\right)^2 .$$

S is the area mask of the local neighbourhood and I the input image.

Local Edge Strength

Characters consist of strokes. Text regions thus have a high density of edges. The local edge strength *E* is defined as the average edge magnitude in a neighbourhood:

$$E = S \bullet \left| D \bullet I \right| .$$

I is the input image, *D* an edge filter (e.g., gradient or Sobel filter), and *S* some averaging filter (e.g., box, binomial, or Gaussian filter). In [2] and [3] a Sobel filter is applied, followed by a circular disk filter of radius 6. The local edge strength responds to text of any orientation.

If only horizontal in plane text should be detected, it is favourable to consider primarily only the horizontal edge strength:

$$E_h = S \bullet \left| D_x \bullet I \right| ,$$

where D_x is some horizontal edges detector. In [32] the horizontal edge strength is directly derived from DCT-encoded JPEG-images and MPEG-based I-frames by means of the sum of the absolute amplitude of the horizontal harmonics in each DCT block(i,j):

$$E_h(i,j) = \sum_{v=v1}^{v2} \left| c_{ov}(i,j) \right|$$

$c_{ov}(i,j)$ are the horizontal harmonics of 8x8 DCT block (*i,j*). The boundaries *v1* and *v2* have to be chosen according to the character size. [32] uses 2 and 6 for *v1* and *v2*, respectively. The DCT coefficients capture the spatial periodicity and directionality in a local block and are therefore a short cut to edge detection. Such a compressed domain edge detector, however, covers only a small part of the many resolutions of a frame posing a problem to scale-independent text extraction. This is especially true for high resolution videos such as HDTV video sequences.

Cai et al. suggest using an adaptive edge strength threshold [2]. They observed that for text embedded in simple background low contrast suffices to render text readable, and that this can also be observed in practice. However, for text embedded in complex-background a high-contrast is always required and used. In a first step a low threshold is applied to the edge strength map. The threshold is selected to accommodate for low-contrast text in simple background. Based on a sliding window, the number of edge-free rows is counted. A high count suggests simple background and no threshold adjustments, while higher counts suggest choosing a higher adaptive threshold in that area to remove more edge pixels. One might argue that a more efficient continuous classifier can be build by using machine learning algorithms.

Edge Density

Text density is usually evaluated by opening/closing operations applied to binarized edge maps. In [2] specific filters are designed, however, it is not clear why they should perform better than standard opening/closing operations. In general, the optimization criterion would be to learn a filter or morphological operation that keeps text regions of certain edge density, while removing non-text regions based on their diverging edge density.

Symmetric Edge Distribution

In areas of clearly readable text one expects – besides high local edge strength – to find edges of all angles and that in most cases an edge of a certain angle is accompanied by an edge in the opposite direction [15]. Clearly visible and readable text should have an edge on both sides of a stroke. Thus

$$1-\sum_{\theta=0}^{\pi}\left(A(\theta)-A(\theta+\pi)\right)^2$$

is measure of symmetry using local edge angle histograms [4]. $A(\theta)$ is the total magnitude of edges in direction θ. This feature is scale invariant. Figure 4 shows an example taken from [3].

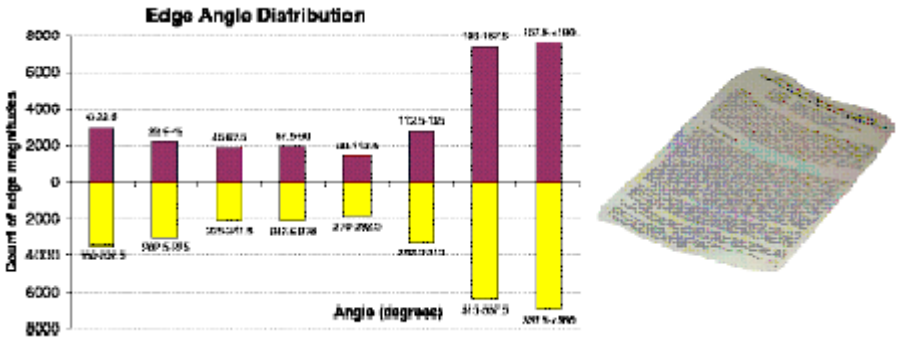

Figure 4: Histogram of edge angle values between 0˚ and 360˚ for the text shown on the right (Figure taken from [3]).

Edge Angle Distribution

For text regions we expect edge angles to be well distributed, i.e., almost all edge angles will occur. An appropriate measure is:

$$EAD=\sum_{\theta=0}^{2\pi}\left(A(\theta)-\overline{A}\right)$$

$\overline{A}$ represents the average magnitude over all directions. The *EAD* measure has its lowest value for homogenous edge distributions and will increase for skewed ones. Unlike most other features this feature allows to distinguish straight ramps, canals, or ridges from text [3]. In other words, at the appropriate scale text areas are isotropic. Alternatively, this attribute could be measured by Jaehne's Inertia tensor [10].

Wavelets

Wavelet decomposition naturally captures directional frequency content at different scales. Li et al. suggest using the mean, second order (variance) and third-order central moments of the LH, HL, and HH component of the first three levels of each 16x16 window [9].

Derivatives

In [16] the gradient image of the RGB input image $I = (I_r, I_g, I_b)$ is used to calculate the complex-values edge orientation image *E*:

$$E(x,y) = \left(\sum_{c \in \{r,g,b\}} \left| \frac{I_c(x,y)}{dx} \right| + \left| \frac{I_c(x,y)}{dy} \right| \cdot i \right).$$

E maps all edge orientations between 0˚ and 90˚, and thus distinguishes only between horizontal, diagonal and vertical orientations.

2.2 Detection

The most common and generic form of feature-based text detection is based on a ***fixed scale and fixed position text classifier*** on some feature image *F*. A feature image *F* is a multi-band image where each band can be one of the features described in subsection 2.1 computed at a given scale from the input image *I*. Given a *W*x*H* window region in a multi-band feature image *F*, a fixed size fixed position text detector classifies the window as containing text if and only if text of a given size is completely contained in the window. Often the window height is chosen to be one or two pixels larger than the largest targeted font height, and the width is chosen based on the width of the shortest possible, but semantically still meaningful word. For instance, in [16] a window of 20x10 was used.

Many different supervised machine learning techniques have been used to train a fixed scale fixed position text classifier such as Decision Trees, Neural Networks, complex Neural Networks, Boosting, Support Vector Machines, GMs, and handcrafted methods. An important design consideration at this stage is the amount of scale and location independence that should be trained into the fixed size fixed position classifier. Common choices for scale independence range from ± 10% to ±50% of some reference font size, while for position independence ±1 to ±*W**10% pixels are common.

Location independence is achieved by sliding the *W*x*H* window pixel by pixel over the whole feature image and recording the probability of having text at that location in a scale-dependent saliency map (see Figure 5, single row). ***Scale independence*** is achieved by applying the fixed scale detection scheme to rescaled input images of different resolution [9][16][29]. Alternatively the features instead of the image can be rescaled to achieve a multi-scale search [17][28].

As one can observe from the forth column in Figure 5, where confidence in text locations is encoded by brightness, text locations stick out as correct hits at multiple scales, while false alarms appear less consistent over multiple scales. Similar results have been observed by Rowley et al. for their neural network-based face detector [23] and by Laurent Itti in his work on models of saliency-based visual attention [5].

In order to recover initial text bounding boxes, the response images at the various scales must be integrated into a consistent text detection result. Different approaches are used for **scale integration**. Examples are:

- Extract and refine initial text boxes at each scale from its associated saliency map in parallel before integrating them into the final

detection result. Each scale might also take into account the response of nearby scales (3).

- Extract and refine initial text boxes sequentially -- from the saliency maps at lower scales to the saliency maps at higher scales. Remove all regions in the higher scale response maps, which have already been detected at lower scales.
- Project the confidence of being text back to the original scale of the input image and extract and refine initial text boxes from the scale-integrated saliency map. Figure 5 column 5 gives an example [16].

There are two principle ways of extracting initial text boxes: bottom-up and top-down approaches. ***Bottom-up approaches*** are region growing algorithms. Starting with seed pixels of highest text probability, text regions are grown iteratively. While this works well for Roman languages due to their low-variance stroke density property, it might cause problems for Chinese characters due to their large variance in stroke density [2]. ***Top-down approaches*** split images regions alternately in horizontal and vertical directions based on texture features [2]. Sometimes both approaches are used simultaneously. For instance in [16] a bottom-up approach is used to find text columns, while a top-down approach is used to partition these text columns into individual text lines.

The overall multi-scale search procedure is summarized inFigure 5. Note that the raw scale and scale independent saliency maps are often smoothed by some morphological operations such as opening and closing.

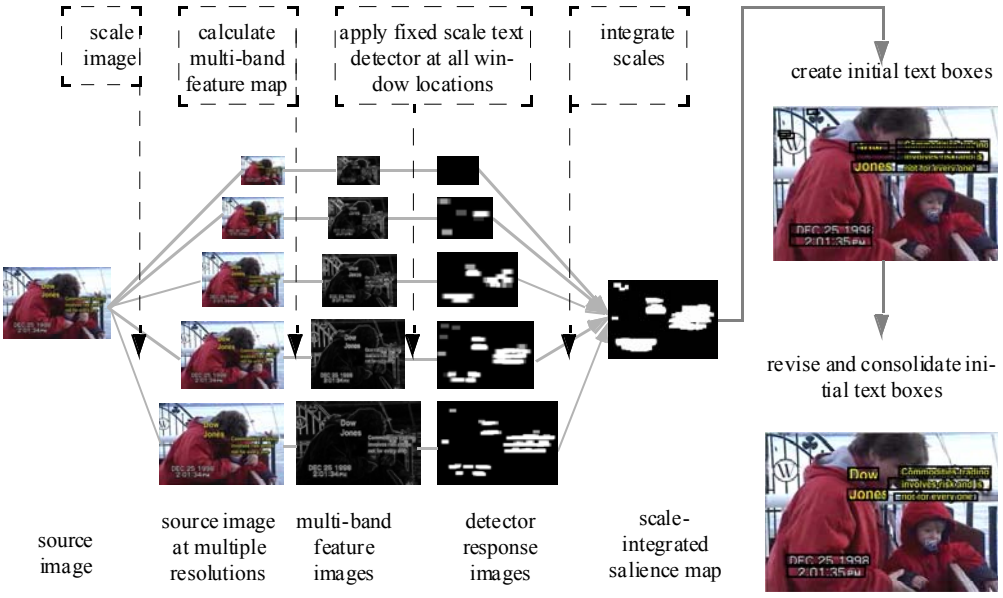

Figure 5: Scale and position independent text localization

2.3 Exploiting Temporal Redundancy

Videos differ from images by temporal redundancy. Each text line appears over several contiguous frames. This temporal redundancy can be exploited to

- increase the chance of localizing text since the same text may appear under varying conditions from frame to frame,
- remove false text alarms in individual frames since they are usually not stable throughout time,
- interpolate the locations of 'accidentally' missed text lines in individual frames, and
- enhance text segmentation by bitmap/stroke integration over time.

Early approaches used tracking primarily to remove false alarms. Therefore, potential text lines or text stroke segments were only tracked over a few frames (e.g., 5 frames) [13][26]. Dependent on whether the tracking was successful or not, a text candidate box or text stroke region was either preserved or discarded. Short term tracking also put fewer requirements on the quality of the tracking module.

More recent approaches summarize text boxes and character strokes of the same content in contiguous frames into a single text object. A text object describes a text line over time by its text bitmaps or connected-components, their sizes and their positions in the various frames as well as their temporal range of occurrence.

Text objects are extracted in a two-stage process in order to reduce computational complexity: In stage 1, a **video** is **monitored** at a coarse temporal resolution (see Figure 6 and [9][16]). For instance, the image-based text localizer of subsection 2.2 is only applied to every second (i.e., every 30th and 25th frame in NTSC and PAL, respectively). The maximum possible step size is given by the assumed minimum temporal duration of text line occurrences. It is known from vision research that humans need between 2 and 3 seconds to process a complex scene. Thus, it is safe to assume that text appears clearly for at least one second.

If text is detected, the second stage of **text tracking** will be entered. In this stage text lines found in the monitoring stage are tracked backwards and forwards in time up to their first and last frame of occurrence. We will restrict our description to forward tracking only since backward tracking is identical to forward tracking except in the direction you go through the video. Also the tracking description will be biased towards the feature based approach, although most can be directly applied to the stroke-based text detection approaches, too.

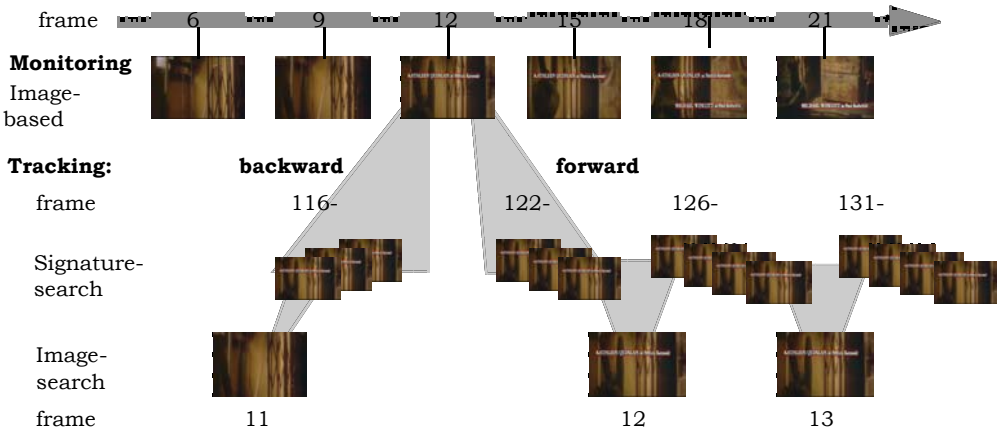

Figure 6: Relationship between video monitoring and text tracking stage

A fast text tracker takes the text line in the current video frame, calculates a characteristic signature, which allows discrimination of this text line from text lines with other contents, and searches in the next video frame for a region of the same dimension, which best matches the reference signature. If the best match exceeds a minimal required similarity, the text line is declared to be found and added to the text object. If the best match does not exceed a minimal required similarity, a signature-based drop-out is declared. The size of the search radius depends on the maximal assumed velocity of text. Heuristically text needs at least 2 seconds to move from left to right in the video. Given the frame size and the playback rate of the video this translates directly to the search radius in pixels. In principle, the search space can be narrowed down by predicting the location of text in the next frame based on the information contained in the text object so far.

The signature-based text line search cannot detect a text line fading out slowly since the search is based on the signature of the text line in the previous frame and not on a fixed master/prototype signature. The frame to frame changes are likely to be too small to be detectable. Further, the signature-based text line search can track zooming in or zooming out text only over a very short period of time. To overcome these limitations, the signature-based search is replaced every x-th frame by the image-based text localizer in order to re-calibrate locations and sizes of the text lines.

Often continuous detection and tracking of text objects is not possible due to imperfection in the video signal such as high noise, limited bandwidth, text occlusion, and compression artefacts. Therefore tracking should be terminated only if for a certain number of contiguous frames no corresponding text line could be found. For this, two thresholds $\max_{DropOut}^{signature-based}$ and $\max_{DropOut}^{image-based}$ are used. Whenever a text object cannot be extended to the next frame, the drop-out counter of the respective localization technique is incremented. The respective counter is reset to zero whenever the search succeeds. The tracking process is finished as soon as one of both counters exceeds its threshold.

Post-Processing
In order to prepare a text object for text segmentation, it must be trimmed down to the part which has been detected with high confidence: the first and last frame in which the image-based text localizer detected the text line. Text objects with a high drop-out rate and/or short duration (e.g., less than a second) should be discarded. The first condition rests on our observation that text lines are usually visible for at least one second. The second condition removes text objects resulting from unstable tracking which cannot be handled by subsequent processing. Unstable tracking is usually caused by strong compression artefacts or non-text objects.

Finally, a few attributes should be determined for each text object:

- **Text colour**: Assuming that the text colour of the same text line does not change over the course of time, a text object's colour is determined as the median of the text colours per frame.
- **Text position**: The position of a text line might be static in one or both coordinates. If static, all text bounding boxes are replaced by the median text bounding box. The median text bounding box is the box whose left/right/top/bottom border is the median over all left/ right/top/bottom borders. If the position is only fixed in one direction such as the x or y axes, the left and right or the top and bottom are replaced by the median value, respectively. Temporally changing coordinate components may be smoothed by linear regression over time.

Figure 7 shows the result of text tracking of located text lines for a sample sequence. All text lines except 'Dow' could be successfully tracked. The line 'Dow' is missed due to its partially difficult background such as the iron gate and face border. The iron gate's edge pattern is very similar to text in general. It also contains individual characters, thus confusing the image-based text localization system, which in turn renders tracking impossible.

Figure 7: Example of text tracking of located text lines. All text lines except 'Dow' could be successfully tracked. The line 'Dow' is missed during text localization due to its difficult background (iron gate and face border).

2.4 Experimental Results

Two different kinds of performance measure have been used by the researchers in the field:

- Pixel-based performance measures and
- Text box-based performance measures.

Both performance measures require ground truth knowledge, i.e., precise knowledge about the text positions in each image/frame. Such ground truth knowledge usually has to be created by hand.

Pixel-based performance numbers calculate the hit rate, false hit rate and miss rate based on the percentage of pixels the ground truth and the detected text bounding boxes have in common:

$$hitrate_{pixel-based} = \frac{100}{|G|} \cdot \sum_{g \in G} \frac{1}{|g|} \cdot \max_{a \in A} \{|a \cap g|\}$$

$$missrate_{pixel-based} = 100 - hitrate_{pixel-based}$$

$$falsehits_{pixel-based} = \frac{100}{|G|} \cdot \sum_{g \in G} \left(\left| \arg\max_{a \in A} \{|a \cap g|\} \right| - \max_{a \in A} \{|a \cap g|\} \right) \Big/ |g|$$

where $A = \{a_1, \ldots, a_N\}$ and $G = \{g_1, \ldots, g_M\}$ are the sets of pixel sets representing the automatically created text boxes and the ground truth text boxes of size $N = |A|$ and $M = |G|$, respectively. $|a|$ and $|g|$ denote the number of pixels in each text box, and $a \cap g$ the set of joint pixels in a and g.

In contrast, the ***text box-based performance numbers*** refer to the number of detected boxes that match with the ground truth. An automatically created text bounding box A is regarded as matching a ground truth text bounding box G if and only if the two boxes overlapped by at least *x*%. Typical values for *x* are 80% or 90%:

$$hitrate_{box-based} = \frac{100}{M} \cdot \sum_{g \in G} \max_{a \in A} \{\delta(a, g)\}$$

$$missrate_{box-based} = 100 - hitrate_{box-based}$$

$$falsehits_{box-based} = \frac{100}{M} \cdot \left(N - \sum_{a \in A} \max_{g \in G} \{\delta(a, g)\} \right),$$

where

$$\delta(a,b) = \begin{cases} \min\left(\frac{|a \cap g|}{|a|}, \frac{|a \cap g|}{|g|} \right) & if \ \min\left(\frac{|a \cap g|}{|a|}, \frac{|a \cap g|}{|g|} \right) \geq 0.8 \\ 0 & else \end{cases}.$$

Alternatively, often recall and precision values are reported:

$$recall = \frac{hits}{hits + missed} \;,\; precision = \frac{hits}{hits + falsealarms}$$

The most important text detection approaches and their reported performances numbers are listed and compared in Table 1.

Table 1: Comparison of text detection approaches; *H*=hit rate; F_1=false alarm rate with respect to text regions; F_2=false alarm rate with respect to patches tested; *F*=false alarm rate with unknown basis; *P*/*R*=precision/recall values.

Work	Scope				Do-main	Per-formance	Comments
	image	exploit video	captions	Scene text	Compressed / Uncompressed		
Cai'02 [2]	x		x		U	*H*: 98.2% *F*: 6.5%	Detection of horizontal text in English and Chinese
Jeong'99 [12]	x		x		U	*H*: 92.2% *F*: 5.1%	Neural Network (NN)-based text detection for news video; English and Chinese
Li'00 [9]	x	x	x	(x)	U	*R*: 92.8% *P*: 91.0%	Tracking system is sensitive to complex background; multi-scale search
Lienhart'02 [16]	x	x	x	(x)	U	*H*: 94.7% F_1: 18%	Complete NN-based system; multi-scale search
Mariano'00 [19]	x		x		U	*H*: 94% *F*: 39%	Designed for horizontal, uniformed coloured text
[Ohya'94] [21]	x		x	x	U	*H*: 95.0%	Detection, Segmentation and Recognition are tightly integrated into each other; focus on upright scene text
Sato1999 [24]	x	x	x		U	*H*: 98.6%	Complete innovative system for CNN Headline News; designed for very small font sizes
Shim'98 [26]	x	x	x	x	U	*H*: 98.8%	Designed for horizontal text only; similar to [14]
[Shin] [27]	x		x		U	*H*: 94.5% *F*: 4.2%	Uses SVM on raw pixel inputs; multi-scale search
Wu'99 [29]	x		x	x	U	*H*: 93.5%	Complete system for video, newspapers, ads, photos, etc.; multi-scale search
Zhong'99 [31]	x	x	x		C	*H*: 96% F_1: 6.07%	Very fast pre-filter for text detection
Zhong'00 [32]	x		x		C	*H*: 99.1% F_1: 36% F_2:: 1.58%	Very fast pre-filter for text detection

Commonly reported sources of text misses are due to weak text contrast with the background, large spacing between the characters, or too large fonts. Non-text regions with multiple vertical structures often result in false alarms.

3 Segmentation

Text segmentation is the task of preparing the bitmaps of localized text occurrences for optical character recognition (OCR). Often standard commercial OCR software packages, which are optimized for scanned documents, are used for recognition due to their high level of maturity.

Text segmentation is commonly performed in two steps: In a first step, the image quality is enhanced in the still image and/or video domain, before in a second step a binary image is derived from the visually enhanced image by means of standard binarization algorithms [21][22].

3.1 Enhancements in the Image Domain

Resolution Enhancement

The low resolution of video (typically 72 ppi) is a major source of problems in text segmentation and text recognition. Individual characters in MPEG-I encoded videos often have a height of less than 11 pixels. Although such text occurrences are still recognizable for humans, it challenges today's standard OCR systems due to anti-aliasing, spatial sampling and compression artefacts [18][15][24]. Today's OCR systems have been designed to recognize text in documents, which were scanned at a resolution of at least 200dpi to 300dpi resulting in a minimal text height of at least 40 pixels. In order to obtain good results with standard OCR systems it is necessary to enhance the resolution of segmented text lines.

A common pre-processing step is to obtain higher resolution text bitmaps by sub-pixel accurate rescaling of the original text bitmaps to a fixed target height, while preserving the aspect ratio. Typical values for the target height range from 40 to 100 pixels, and cubic interpolation or better up-sampling filters are used for rescaling. Fixing a target height is computationally efficient, because text with a larger height neither improves segmentation nor OCR performance [9][16][24]. In addition, the fixed target height effectively normalizes the stroke widths to a narrow range for Roman characters, which in turn can be used later for additional refinement operations.

Character Stroke Enhancement

Sato et al. propose to use 4 directional stroke filters of 0°, +45°, -45°, and 90° trained by fixed English fonts. These filters calculate the probability of each pixel being on a text stroke of that direction. By integrating the four filter results an enhance text stroke bitmap is formed (see Figure 8 taken from [24])

Figure 8: "Result of character extraction filters: (a) 0° (b) 90° (c)-45° (d) 45° (e) Integration of four filters (f) Binary image" (Figure taken from [24]).

3.2 Enhancements in the Video Domain

Temporal Integration

Text objects in videos consist of many bitmaps of the same text line in contiguous frames. This redundancy can be exploited in the following way to remove the complex background surrounding characters: Suppose the bitmaps of a text object are piled up over time such that the characters are aligned perfectly with each other. Looking through a specific pixel in time, one may notice that pixels belonging to text vary only slightly, while background pixels often change tremendously through time. Since a text line's location is static due to its alignment its pixels are not supposed to change. In contrast, background pixels are very likely to change due to motion in the background or motion of the text line (see Figure 9(a)).

A temporal maximum/minimum operator applied to all or a subset of perfectly aligned greyscale bitmaps of a text object for normal/inverse text is generally capable to separate text pixels from background pixels. This temporal maximum/minimum operation was first proposed by Sato et al. for static text [25], but can also applied to moving text if the text segmentation system supports sub-pixel accurate text line alignment [16]. An alternative approach to the min/max operation is to calculate a pixel's temporal mean and variance and reject pixels with large standard deviations or a few outliers.

Sub-pixel Accurate Text Alignment

Two similar proposals have been developed by Li [9] and Lienhart [16]. The latter approach, though, is more robust since it exploits the estimated text colour during tracking and, therefore, does not have problems with complex background as reported by [9].

The sub-pixel accurate text alignment is achieved as follows: In a first step, the bounding boxes of detected text locations are slightly increased to

ensure that text is always 100% contained in the enlarged bounding boxes (see Figure 10). Let $B_1(x,y),...,B_N(x,y)$ denote the N bitmaps of the enlarged bounding boxes of a text object and $B^r(x,y)$ the representative bitmap, which is to be derived and initialized to $B^r(x,y) = B_1(x,y)$. Then, for each bitmap $B_i(x,y), i \in \{2,...,N\}$, the algorithm searches for the best displacement vector $\left(dx_i^{opt}, dy_i^{opt}\right)$, which minimizes the difference between $B^r(x,y)$ and $B_i(x,y)$ with respect to pixels having text colour, i.e.,

$$(dx_i^{opt}, dy_i^{opt}) = \arg\min \sum_{(x,y) \in B^r \cap B^r(x,y) \subseteq textColor} | B^r(x,y) - B_i(x+dx, y+dy) |$$

A pixel is defined to have text colour if and only if it does not differ more than a certain amount from the greyscale text colour estimated for the text object.

At each iteration, $B^r(x,y)$ is updated to

$$B^r(x,y) = op(B^r(x,y), B_i(x+dx_i^{opt}, y+dy_i^{opt})),$$

where *op=max* for normal text and *op=min* for inverse text. Figure 9(b) shows an example of the min/max operation.

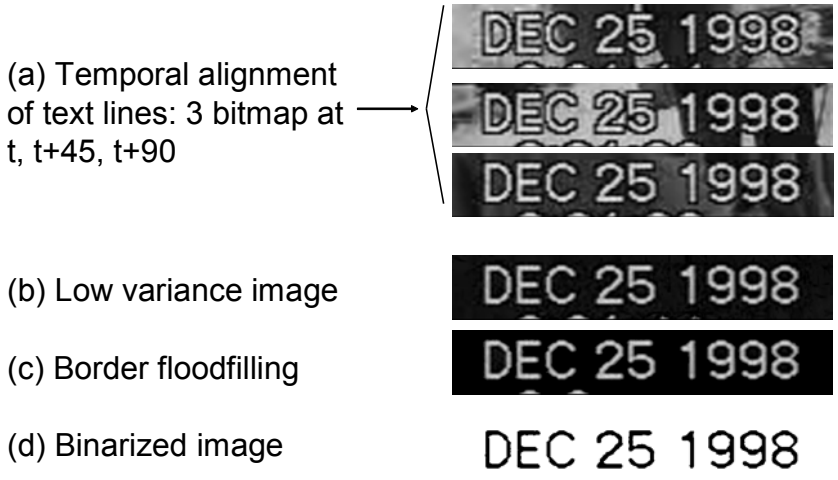

Figure 9: Example of the various text segmentation steps

3.3 Segmentation

Different segmentation techniques have been used for text segmentation. Sometimes several of them are combined to achieve better and more reliable segmentation results.

Seedfilling from Border Pixels

Text occurrences are supposed to have enough contrast with their background in order to be easily readable. This feature can be exploited to remove large parts of the complex background. The basic idea is to increase the text bounding boxes such that no text pixels fall onto the border and then to take each pixel on the boundary of the text bounding box as a seed to a virtual seedfill procedure, which is tolerant to small colour changes. Pixels which differ not more than $threshold_{seedfill}$ from the seed will be regarded as pixels of the same colour as the seed. In theory the virtual seedfill procedure should never remove character pixels since the pixels on the boundary do not belong to text and text contrasts with its background. We attributed the seedfill procedure with "virtual" since the fill operation is only committed after the seedfill procedure has been applied to all pixels on the border line in order avoid side effects between different seeds [16].

In practise, however, text segmentation sometimes has to deal with low contrast, which may cause the seedfill algorithm to leak into a character. A stop criterion may be defined based on the expected stroke thickness. Regions which over a large extent comply with the stroke thickness range of characters in one dimension should not be deleted.

Not all background pixels are eliminated by this procedure, since the sizes of the regions filled by the seed-fill algorithm are limited by the maximum allowed colour difference between a pixel and its border pixel seed. In addition, some regions are not connected to the border such as the interior of closed stroke characters 'o' and 'p'. Therefore, a hypothetical 8-neighborhood seedfill procedure with $threshold_{seedfill}$ is applied to each non-background pixel in order to determine the dimension of the region that can hypothetically be filled. Background regions should be smaller then text character regions. Therefore, all hypothetical regions violating the typical range of width and height values for characters are deleted.

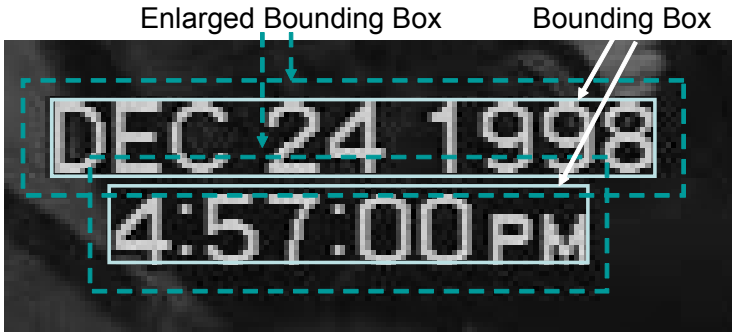

Figure 10: Relationship between bounding boxes and their enlarged counterparts.

Thresholding

The simplest form of thresholding rests on a single, global threshold. Many different variants of global thresholding have been designed – ranging from

bi-level schemes to tri-level schemes. More sophisticated variants also exploit the estimated text colour [16][24][29].

For text on complex background a global threshold may not be appropriate since background pixel can have similar greyscale values as the text, or it be brighter and darker than the text at different locations. In these cases an adaptive threshold should be applied. Commonly used adaptive binarization algorithms are derivatives of Otsu's [22] and Ohya's work [21].

3.4 Experimental Results

For text segmentation no generally accepted performance measure has emerged in the literature. The three most common performance measures are:

- ***Manual visual inspection***: Correctness is determined by manual visual inspection of all created binary bitmaps.
- ***OCR Accuracy***: Segmentation performance is evaluated indirectly by means of the resulting OCR error rate with a given OCR engine making the results dependent on the OCR engine and its peculiarities.
- **Probability of Error**: The probability of error measure requires pixel maps of the ground truth data, which in most cases is very hard to provide. The probability of error (PE) is defined as follows [6]:

  $$PE = P(O)P(B \mid O) + P(B)P(O \mid B),$$

  where P(B|O) and P(O|B) are the probability of error in classifying a text/background pixel as background/text pixel, P(O) and P(B) are the a priori probabilities of text/background pixels in the test images.

Table 2 reports the important text segmentation approaches and their performances numbers. Only OCR accuracy in reported for comparability.

Work	Analysis scope		Performance	Comments
	Image/video	Captions/ Scene text		
Li'99	I, V	C, S	RC: 88%	Address overlay text and scene text with two separate approaches; resolution and video enhancement
Lienhart'02 [16]	I, V	C, (S)	RC: 69.9%	Uses standard OCR software; resolution and video enhancement; difficult and large test set
Lienhart'96 [14]	I, V	C, (S)	RC: 80.3%	Connected-component based approach; simple self-trained OCR engine
Lienhart'00 [15]	I, V	C, (S)	RC: 60.7%	Connected-component based approach; standard OCR engine
Lopresti'00	I	C	RC: 87.9	Developed for text in web images

[18]				
Ohya'94 [21]	I	C,S	RC: 34.3%	Detection, Segmentation and Recognition are tightly integrated into each other
Sato'99 [24]	I, V	C	RC: 83.5%	Integrated domain specific (CNN Headline News) text segmentation and recognition approach; achieved further recognition improvements by using domain-specific dictionaries; resolution and video enhancement
Wu'99 [29]	I	C, (S)	RC: 83.8% RW: 72.4%	Uses standard OCR software; segmentation is based on global threshold per text region

Table 2: Comparison of text segmentation approaches; RC=character recognition rate, RW= word recognition rate

4 CONCLUSION

Text localization and text segmentation in complex images and video have reached a high level of maturity. In this survey we focused on texture-based approaches for artificial text occurrences. The different core concepts underlying the different detection and segmentation schemes were presented together with guidelines for practitioners in video processing. Future research in Video OCR will focus more on scene text as well as on further improvements of the algorithms for localization and segmentation of artificial text occurrences.

5 REFERENCES

[1] Lalitha Agnihotri and Nevenka Dimitrova. Text Dection for Video Analysis. *IEEE Workshop on Content-Based Access of Image and Video Libraries*, 22 June 1999, Fort Collins, Colorado, 1999.

[2] Min Cai, Jiqiang Song, and Michael R. Lyu. A New Approach for Video Text Detection. *IEEE International Conference on Image Processing*, pp. 117-120, 2002.

[3] P. Clark and M. Mirmehdi. Finding Text Regions Using Localised Measures. Proceedings of the *11th British Machine Vision Conference*, pp. 675-684, BMVA Press, September 2000.

[4] P. Clark and M. Mirmehdi. Estimating the orientation and recovery of text planes in a single image. Proceedings of the *12th British Machine Vision Conference*, pp. 421-430, BMVA Press, September 2001.

[5] L. Itti, C. Koch, and E. Niebur. A Model of Saliency-Based Visual Attention for Rapid Scene Analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(11):1254-1259, 1998.

[6] S. U. Lee, S. Y. Chung, and R. H. Park. A Comparative Performance Study of Several Global Thresholding Techniques for Segmentation. *Computer Vision, Graphics, and Image Processing*, Vol. 51, pp. 171-190, 1990.

[7] Huiping Li, O. Kia and David Doermann. Text Enhancement in Digital Videos. *Proc. SPIE Vol. 3651: Document Recognition and Retrieval VI*, p. 2-9, 1999.

[8] Huiping Li and David Doermann. Superresolution-Based Enhancement of Text in Digital Video. *15th Pattern Recognition Conference*, Vol.1, pp. 847-850, 2000.

[9] Li, D. Doermann and O. Kia. Automatic Text Detection and Tracking in Digital Video. *IEEE Transactions on Image Processing*. Vol. 9, No. 1, pp. 147-156, Jan. 2000.

[10] Bernd Jaehne. Digital Image Processing. Springer-Verlag Berlin Heidelberg, 1995.

[11] Anil K. Jain and Bin Yu. Automatic Text Localication in Images and Video Frames. *Pattern Recognition*, 31(12), pp. 2055-2076, Dec. 1998.

[12] Ki-Young Jeong, Keechul Jung, Eun Yi Kim, and Hang Joon Kim. Neural Network-based Text Location for News Video Indexing. *IEEE International Conference on Image Processing*, Vol. 3, pp. 319-323, 1999.

[13] Rainer Lienhart and Frank Stuber. Automatic Text Recognition in Digital Videos. *Proc. SPIE 2666: Image and Video Processing IV*, pp. 180-188, 1996.

[14] Rainer Lienhart. Automatic Text Recognition for Video Indexing. *Proc. ACM Multimedia 96*, Bosten, MA, pp. 11-20, Nov. 1996.

[15] Rainer Lienhart and Wolfgang Effelsberg. Automatic Text Segmentation and Text Recognition for Video Indexing. *ACM/Springer Multimedia Systems*, Vol. 8, pp. 69-81, Jan. 2000.

[16] Rainer Lienhart and Axel Wernicke. Localizing and Segmenting Text in Images, Videos and Web Pages. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.12, No. 4, pp. 256 -268, April 2002.

[17] Rainer Lienhart and Jochen Maydt. An Extended Set of Haar-like Features for Rapid Object Detection. *IEEE International Conference on Image Processing*, Vol. 1, pp. 900-903, Sep. 2002.

[18] Daniel Loprestie and JiangYing Zhou. Locating and Recognizing Text in WWW Images. *Information Retrieval*, Kluwer Academic Publishers, pp. 177-206, 2000.

[19] Vladimir Y. Mariano and Rangachar Kasturi. Locating Uniform-Colored Text in Video Frames. *15th Int. Conf. on Pattern Recognition*, Vol.4, pp. 539-542, 2000.

[20] G. Myers, R. Bolles, Q.-T. Luong, and J. Herson. Recognition of Text in 3-D Scenes. *4th Symposium on Document Image Understanding Technology*, Columbia, Maryland, pp. 23-25, April 2001.

[21] Jun Ohya, Akio Shio, and Shigeru Akamatsu. Recognizing Characters in Scene Images. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2, Febr. 1994.

[22] N. Otsu. A Threshold Selection Method From Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, pp. 62-66, 1979.

[23] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural Network-Based Face Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 1, pp. 23-38, January 1998.

[24] T. Sato, T. Kanade, E. Hughes, M. Smith. Video OCR for Digital News Archives. *IEEE Workshop on Content-Based Access of Image and Video Databases*, Bombay, India, January, pp. 52-60, 1998.

[25] T. Sato, T. Kanade, E. K. Huges, M. A. Smith, and S. Satoh. Video OCR: Indexing Digital News Libraries by Recognition of Superimposed Caption. *ACM Multimedia Systems*, Vol. 7, No. 5, pp. 385-395, 1999.

[26] Jae-Chang Shim, Chitra Dorai, and Ruud Bolle. Automatic Text Extraction from Video for Content-based Annotation and Retrieval. *IBM Technical Report*, RC21087, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, January 1998.

[27] C.S. Shin, K.I. Kim, M.H. Park, H.J. Kim. Support Vector Machine-based Text Detection in Digital Video. *Proceedings of the IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing X*, Vol. 2, pp. 634-641, 2000.

[28] Paul Viola and Michael J. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE Computer Vision and Pattern Recognition*, Vol. 1, pp. 511-518, 2001.

[29] V. Wu, R. Manmatha, E.M. Riseman. Textfinder: An Automatic System to Detect and Recognize Text in Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 21, Issue 11, pp. 1224-1229, Nov. 1999.

[30] Boon-Lock Yeo and Bede Liu. Visual Content Highlighting via Automatic Extraction of Embedded Captions on MPEG Compressed Video. in Digital Video Compression: Algorithms and Technologies, Proc. SPIE 2668-07 (1996).

[31] Yu Zhong, Hongjiang Zhang, and A.K. Jain. Automatic Caption Localization in Compressed Videos. *IEEE International Conference on Image Processing*, Vol. 2, pp. 96-100, 1999.

[32] Yu Zhong, Hongjiang Zhang, and A.K. Jain. Automatic Caption Localization in Compressed Videos. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, Issue 4, pp. 385-392, April 2000.