Model based text detection in images and videos: a learning approach*

Christian Wolf Jean-Michel Jolion

Technical Report LIRIS RR-2004-13

LIRIS INSA de Lyon
Bât. Jules Verne 20, Avenue Albert Einstein
Villeurbanne, 69621 cedex, France
Tel.: +33 4 72 43 60 89, Fax.: +33 4 72 43 80 97
Email: {christian.wolf,jean-michel.jolion}@rfv.insa-lyon.fr

Abstract

Existing methods for text detection in images are simple: most of them are based on texture estimation or edge detection followed by an accumulation of these characteristics. Geometrical constraints are enforced by most of the methods. However, it is done in a morphological post-processing step only. It is obvious, that a weak detection is very difficult — up to impossible — to correct in a post-processing step. We propose a text model which takes into account the geometrical constraints directly in the detection phase: a first coarse detection calculates a text "probability" image. After wards, for each pixel we calculate geometrical properties of the eventual surrounding text rectangle. These features are added to the features of the first step and fed into a support vector machine classifier.

Keywords

Text detection, recognition, OCR, semantic indexing, content based video retrieval

1 Introduction

The existing OCR technology and document page segmentation algorithms were developed for scanned paper documents, an application to natural image taken with a camera or video sequences is hardly possible. Therefore, robust reading from these media needs to resort to specific text detection and extraction algorithms.

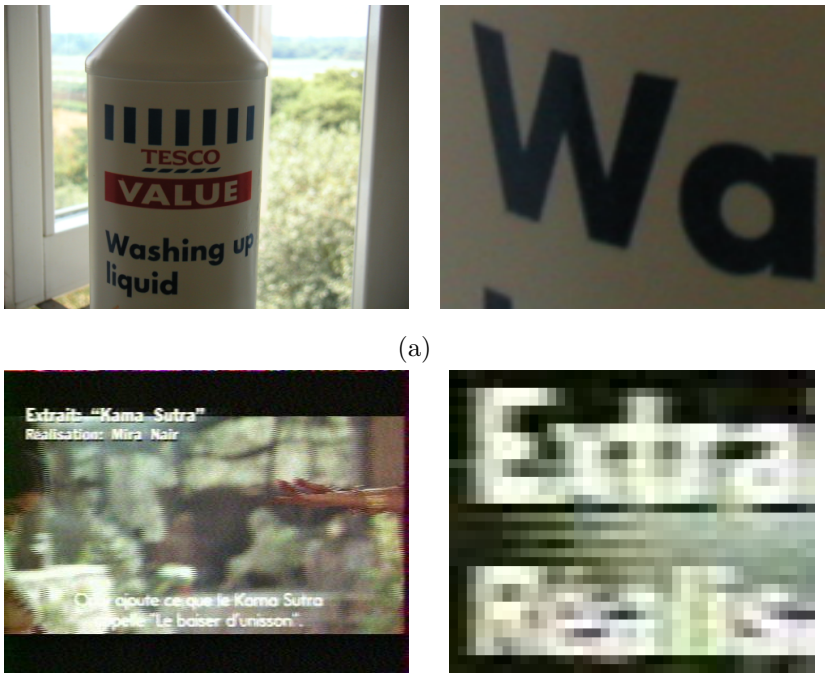
Text detection and extraction from images and video sequences is a relatively young research topic. The first algorithms had been developed for complex scanned paper documents, for instance colored journals. Then, the potential of text detection for semantic video indexing was discovered and algorithms working on videos were proposed. These algorithms were mostly conceived for artificial text, i.e. text which has been overlaid over the image by an operator after it has been taken by a camera. This kind of text is often considered as easier to detect and more useful for indexing purposes as scene text, i.e. text which is present in the scene when the image or video is shot.

By definition, camera based document analysis targets scene text, which is considered as being harder to detect and to process. However, the distinction between artificial text and scene text has been made from a conceptual point of view. From a signal processing point of view, the two types of text are not necessarily very different, as figure 1 illustrates. Figure 1a shows an image with scene text taken with a digital camera[1] and a zoom into the text area. The high resolution of the image (1600×1200 pixels) results in a high visual quality of the characters, which may be segmented very well. On the other hand, figure 1b shows an image taken from a frame of an MPEG 1 video sequence with overlaid artificial text. The low resolution results in a very bad quality of the characters, which cannot be segmented.

Thus, one of the most limiting factors for camera based document analysis algorithms is the image resolution and therefore the size of the text. In this article we propose a method for the extraction of low quality and low resolution text from images and videos. For this purpose we resort to signal features which

*The work presented in this article has been conceived in the framework of two industrial contracts with France Télécom in the framework of the projects ECAV I and ECAV II.

[1]The image has been used in the ICDAR 2003 robust reading competition.



(a)

(b)

Figure 1: Example images and zooms into the text area: (a) artificial text; (b) scene text.

can be robustly detected for text of very small size: contrast and word or phrase geometry as opposed to character geometry.

This article is organized as follows: section 2 gives an overview of the state of the art of text detection and extraction. Section 3 describes the general framework of text detection in video sequences. Section 4 treats the problem of text detection in still images or video frames. We model the signal and the geometric properties of text, whose parameters are learned from training data. Section 5 presents the experimental results obtained on a database of still images and video sequences containing artificial and scene text. New evaluation measures are introduced in order to evaluate the detection performance of our algorithm. Finally, section 6 gives a conclusion.

2 Previous work

The existing work on text detection can be classified according different criteria. The cited methods are classified according to the type of algorithms they employ. However, a historical point of view is taken into account.

Detection through segmentation and spatial grouping

The first text detection algorithms, introduced by the document processing community for the extraction of text from colored journal images and web pages, segment characters before grouping them to words and lines. Jain et al. [14] perform a color space reduction followed by color segmentation and spatial regrouping to detect text. Although processing of touching characters is considered by the authors, the segmentation phase presents major problems in the case of low quality documents, especially video sequences. A similar approach, which gives impressive results on text with large fonts, has been presented by Lienhart [24]. A segmentation algorithm and regrouping algorithm are combined with a filter detecting high local contrast, which results in a method which is more adapted to text of low quality. Still, the author cannot demonstrate the reliability of his algorithm in the case of small text. False alarms are removed by texture analysis, and tracking is performed on character level, which might pose considerable problems in the case of text as presented in figure 1b. Similar methods working on color clustering or thresholding followed by a regrouping of components have been presented by Lee and Kankanhalli [21], by Zhou and Lopresti [46] and by Sobottka et al. [35]. Hase et al. cluster the components and allow for spatial arrangements which follow a quadratic function [11].

Scanline processing

Some methods are scanline based, i.e. they proceed line by line during the classification phase. Mariano and Kasturi perform a color clustering of the pixels of each scan line in order to find the pixels of the text cluster [27]. Histograms of line segments of uniform color are computed and compared across lines to form rectangles. Wong and Chen calculate gradient measures for each line and cut the lines to segments of similar gray value [43]. Adjacent scanlines are merged using a statistical similarity criterion.

Detection in maps and charts

Methods for text extraction from very graphical documents, e.g. maps, followed similar patterns as the ones developed by the document processing community. Tan et al. use a hierarchical processing of the connected components in the image to find text as regrouped components [36]. Brès and Eglin [2] binarize the map and glide a rectangular map across the image. Inside each window, measures as the number of vertical segments, spacing, regularity etc. are calculated and used for the decision whether a pixel contains text or not.

The methods based on segmentation work fine for high resolution images as newspapers and journals but fail in the case of low resolution video, where characters are touching and the font size is very small. New methods developed by the image and video processing community based on edge detection or texture analysis were soon introduced when the attention focused to video.

Edge based detection

The video indexing system introduced by Sato et al. [32] combines closed caption extraction with superimposed caption (artificial text) extraction. The text extraction algorithm is based on the fact that text consists of strokes with high contrast. It searches for vertical edges which are grouped into rectangles. The authors recognized the necessity to improve the quality of the text before passing an OCR step. Consequently, they perform an interpolation of the detected text rectangles before integrating multiple frames into a single enhanced image by taking the minimum/maximum value for each pixel. They also introduced an OCR step based on a correlation measure. A similar method using edge detection and edge clustering has been proposed by Agnihotri and Dimitrova [1]. Wu, Manmatha and Riseman [44] combine the search for vertical edges with a texture filter to detect text. Unfortunately, these binary edge clustering techniques are sensitive to the binarization step of the edge detectors. A similar approach has been developed by Myers et al. [29]. However, the authors concentrate on the correction of perspective distortions of scene text after the detection. Therefore, the text must be large so that baselines and vanishing points can be found. Since the detection of these features is not always possible, assumptions on the imaging geometry need to be made.

LeBourgeois [20] moves the binarization step after the clustering by calculating a measure of accumulated gradients instead of edges. The coarse detection step of our work is based on a slightly modified variant of this filter, but our detection technique also uses higher level features based on a robust estimation of the geometry of the coarsely detected features. In his work, LeBourgeois proposes an OCR algorithm which uses statistics on the projections of the gray values to recognize the characters.

A couple of methods use mathematical morphology in the detection step. Hori dilates Sobel edges into text regions [12]. The emphasis of his work is set on binarization and removal of complex backgrounds. Hasan and Karam dilate edges detected with a morphological detector [10]. The main drawback of their algorithm are the strong assumptions on the geometry and the size of the text.

In the previously introduced methods, the features calculated for the discrimination between text and non-text pixels are of very basic nature. This is due to the fact that in a neighborhood of small size, text does not have a very distinctive signature. Most people use edge density and related features, as high frequency components of wavelet decompositions etc. Very sophisticated texture features are of limited use since the texture of text is very irregular, especially in case of short text. Sin et al. detect the text using features calculated on the autocorrelation function of a scanline [34]. However, they exploit the fact that text in their application (billboard detection) is very long and enclosed by a rectangular frame (e.g. the panel of the billboard). Furthermore, several scanlines of a text rectangle are concatenated, which creates problems due to the non-alignment of the phases of the Fourier-transforms of the different scanlines. The method cannot be applied to short text.

Detection through learning methods

Various methods based on learning have also been presented. Li and Doermann use a Haar wavelet for feature extraction [22]. By gliding a fixed size window across the image, they feed the wavelet coefficients

to a MLP type neural network in order to classify each pixel as "text" or "non-text". Clark and Mirmehdi also leave the classification to a neural network fed with various features, as the histogram variance, edge density etc. [7]. Similarly, Wernike and Lienhart extract overall and directional edge strength as features and use them as input for a neural network classifier [40]. In a more recent paper, the same authors change the features to a 20×10 map of color edges, which is fed directly to the neural network [25]. Jung directly uses the gray values of the pixels as input for a neural network to classify regions whether they contain text or not [16]. Kim et al. also use the gray values only as features, but feed them to a support vector machine for classification [17]. Tang et al. use unsupervised learning to detect text [37]. However, their features are calculated from the differences between adjacent frames of the same shot, resulting in a detection of appearing and disappearing text. Text which is present from the beginning of a shot to the end (a type of text frequently encountered for locations in news casts) is missed. Chen et al. [6] introduce a two step process of fast candidate text line detection by edge detection and morphological processing and text rectangle identification by SVM learning. The features fed to the SVM are an edge distance map of the scaled text box. The drawback of the method is the simple coarse detection step. Text on complex background is likely to touch the background, thus cannot be segmented and therefore not be verified by the more complex identification step.

As usual, learning methods depend on the quality of the training data used to train the systems and on the features which are fed into the learning machine. However, in video, text appears in various sizes, fonts, styles etc., which makes it very difficult to train a generalizing system. The features which have been presented in the previous work are very basic (gray values, normalized gray values, distances to the edge map, etc.), it must be feared that they do not generalize well across the set of possible text types.

Detection in compressed data

Methods working directly in the compressed domain have also been introduced. Zhong, Zhang and Jain extract features from the DCT coefficients of MPEG compressed video streams [45]. Detection is based on the search of horizontal intensity variations followed by a morphological clean up step. Randall and Kasturi [8] compute horizontal and vertical texture energy from the DCT coefficients. Unfortunately they only obtain good results for large text. Gu [9] also uses the DCT coefficients to detect static non-moving text and removes false alarms by checking the motion vector information. One of the problems of these methods is the large number of existing video formats (MPEG 1&2, MPEG 4, Real Video, wavelet based methods etc.) and the high rate of innovation in the video coding domain, which makes methods working on compressed data quickly obsolete.

Although the research domain is very young, there exist already a high number of contributions. Only few of them present complete methods beginning from the detection of the text until the last binarization step before passing the results to an OCR. In most cases the model behind the detection algorithms is very simple: edge detection, regrouping into rectangles.

3 The extraction framework

One of the major design goals of our method was the ability to extract text from still images as well as video sequences. Therefore, in the case of video sequences, spatio-temporal approaches to text detection were not an option, which treat the 3D input stream as such and detect the text directly in the 3D space. Instead, a tracking method treats the input stream as a temporal sequence of images and detects text on a frame by frame basis. The detected text objects are tracked across several frames in order to create the text appearance, which is a result of the detection *and* the tracking process. In this article, we concentrate on static, non moving text. A generalization of our work to simple, linear movement has been done by Marquis et al. [28].

Apart from the detection itself, the additional temporal component in video sequences also leads to consequences and open questions concerning the usage of the detected appearance. The detected text forms a 3D object with a temporal dimension, which cannot be recognized by an OCR software directly, at least not by existing OCR technology. Several possible techniques may be considered:

- Choose a single frame out of all possible frames of the appearance. This is a simple technique, which has the disadvantage of losing the possible additional temporal information in the appearance.
- Use the 3D text object as it is and develop a new recognition algorithm which exploits the temporal information directly in order to increase recognition performance.

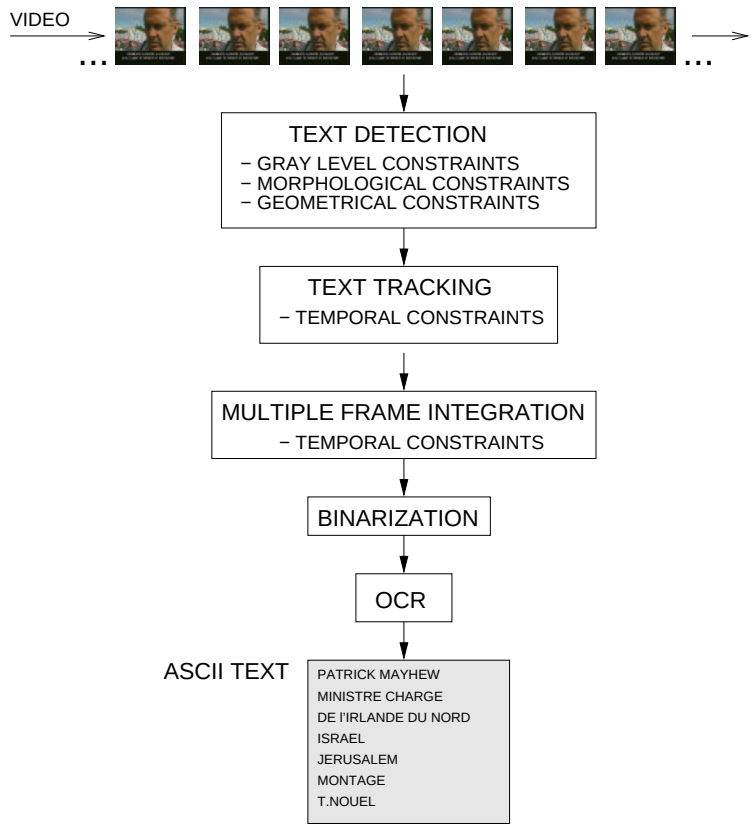


Figure 2: The scheme of our system.

- Integrate the 3D appearance into a single 2D image, exploiting the temporal information in order to "clean up" the image and to create a single image of better quality.
- Apply the recognition algorithm to *each* of the frames in order to create a set of recognized text strings. Apply symbolic statistics to the set in order to create a single string with the most probable contents.

The first solution — choosing the text rectangle from a single frame for recognition — fails due do the miserable quality of most text taken from videos. At least some processing is necessary to enhance the image quality.

The second solution is beyond the scope of this work. The development of a new OCR technology needs a tremendous amount of engineering experience in order to find the necessary heuristics which allow these systems to obtain their excellent recognition performance. Instead, we apply commercial software, which delivers excellent results on scanned printed or faxed documents.

Unfortunately, commercial OCR software is not adapted to the type of data, which is why we chose the third solution: We integrate the text appearance into a single image of better quality, which is closer to the type of data expected by commercial OCR software.

Our research team also successfully worked on the fourth way to use the text appearance. An algorithm which uses new theoretical research on statistical processing of character strings done by our team is given in [15].

A global scheme of our proposed system for text extraction from video sequences is presented in figure 2. As already stated, the detection algorithm for still images is applied to each frame of the sequence separately. The detected text rectangles are passed to a tracking step, which finds corresponding rectangles of the same text appearance in different frames. From several frames of an appearance, a single enhanced image is generated and binarized, i.e. segmented into characters and background, before passing it to a standard commercial OCR software.

Text in videos has gray level properties (e.g. high contrast in given directions), morphological properties (spatial distribution, shape), geometrical properties (length, ratio height/length etc.) and temporal properties (stability). Our method makes use of these properties, starting from the signal and going se-

quentially to the more domain dependent properties. The final step (the character segmentation) results in a set of binary boxes containing text which need to be recognized by a classical commercial OCR system.

The global scheme of our system including the tracking, enhancement and binarization algorithms has already been published in detail in [42]. In this work, we concentrate on the algorithm for text detection in images or video frames, which will be described in detail in the next section.

4 Detection in still images

The heart of the extraction system is the detection algorithm for still images. A common issue in object detection is the problem of finding a decision function for each pixel deciding whether it is part of the object or not. The pre-attentive nature of the state of the art of computer vision tends to produce algorithms whose results are not binary, but continuous. This is a very common problem in computer vision, which is for example also encountered during edge detection algorithms. Gradient based edge detection algorithms produce continuous outputs with smaller responses to noise and light background texture and larger responses for edges, which (hopefully) correspond to object boundaries. The problem of automatically determining a threshold which separates these two cases is far from being resolved.

The same is true in the case of text detection, the decision function needs to be thresholded. In [42] we proposed a method which assumes that there is text in a frame or image and calculates the optimal threshold using Otsu's method based on discriminant analysis [31]. In this article we propose a different approach, which learns text features from training data and therefore delegates the problem of finding a threshold to the learning machine, which needs to learn a decision border in the feature space. The use of machine learning allows us to benefit from several advantages:

- We increase the precision of the detection algorithm by learning the characteristics of text.
- We are able to use more complex text models, which would be very difficult to derive analytically or to verify by heuristics.
- The discovery of support vector machine (SVM) learning and it's ability to generalize even in high dimensional spaces opens the door to complex decision functions and feature models.

The distinction of our work to other methods based on SVM learning lies in the choice of features. The approaches [16] and [6] feed very simple features into the SVM, namely directly the gray values in a local window around the pixel or an edge distance map. In these works, the scientists delegated the difficult task of feature design to the learning machine. It is well known that implicit feature extraction does not give the same results as wisely done manual feature design — finally, only the scientist knows what he or she needs to detect, as opposed to the learning machine.

Our text model contains the following hypotheses:

Text contains a high amount of *vertical strokes*.
Text contains a *baseline*, i.e. its border area forms a regular rectangle.

Indeed, one of the main properties of text is its geometry: The presence of a rectangular bounding box and/or the presence of a baseline[2]. On the other hand, the texture properties of text tend to get more and more unreliable, as the detected text gets shorter and shorter.

Until now, existing text detection features have been very simple: gradient or edge densities, texture features based on filtering (Gabor, derivatives of a Gaussian etc.) or high frequency components of wavelet decompositions. While it is true, that higher level features as geometrical constraints etc. are enforced by most of the existing methods, they are employed at a second stage for verification of a segmented rectangle only. Very often, mathematical morphology is used to clean up noise and also to enforce geometrical constraints, but once again this happens after the classification of the pixels — whether they are text or non-text — has been done. As a consequence, the weakness of these approaches is the hard (up to impossible) improvement of weak classification decisions. A very badly segmented image, where the text box touches a complex background in many places, or where a text box and the background make a single convex region, are impossible to correct.

[2]In document analysis, the baseline is traditionally the lower boundary of the text area. In our case, when we refer to "baseline", we mean the upper and/or the lower boundary line of the text, since the presence of these two lines is a necessary condition for text.

A logical step is to include the geometrical features directly into the decision process for each pixel. Unfortunately, this is a chicken-egg problem: in order to estimate geometrical constraints, we first need to detect text. Consequently, we adopted a two step approach:

- Perform a coarse detection to emphasize text candidates without taking into account geometrical features. This detection is based on the detection of areas containing a high density of vertical strokes.
- For each pixel, calculate geometrical features of its neighborhood based on the detection results from step 1. Use these features together with the features calculated in step 1 and perform a new refined detection.

4.1 Coarse detection - stroke density

The first coarse detection phase reuses the detection algorithm we presented in [42], which is a modified version LeBourgeois's algorithm [20]. It detects the text with a measure of accumulated gradients:

$$\boldsymbol{A}(x,y) = \left[\sum_{i=-\lfloor S/2 \rfloor}^{\lfloor S/2 \rfloor} \left(\frac{\partial \boldsymbol{I}}{\partial x}(x+i,y) \right)^2 \right]^{\frac{1}{2}} \tag{1}$$

where $\boldsymbol{A}$ is the filtered image and $\boldsymbol{I}$ is the input gray value image. The parameters of this filter are the implementation of the partial derivative and the size S of the accumulation window. We chose the horizontal version of the Sobel operator as gradient measure, which obtained the best results in our experiments[3]. The size of the accumulation window depends on the size of the characters and the minimum length of words to detect. Since the results are not very sensitive to this parameter, we set it to a fixed value $S = 13$.

4.2 Refinement - geometrical features

In this second step we detect the text baseline as the boundary of a high density area in the image which is the result of the first, rough text detection. Detecting the baseline explicitly, i.e. by using a Hough transform or similar techniques, is inherently difficult due to the irregularity of the boundary and the possibility of very short text. Furthermore, the boundaries between the text region and the non-text background may be fuzzy, especially if the text orientation does not totally coincide with the text filter direction (e.g. if a text with 15° slope needs to be detected with a horizontal filter). Therefore, the direct detection of lines and contours is rather difficult. Instead we detect, at each pixel, the height of the eventually associated text rectangle, and check for differences in these heights across a neighborhood. In text areas, the text height should remain approximately constant. Using the height instead of the vertical position of the rectangle is more robust, especially to rotations of the text.

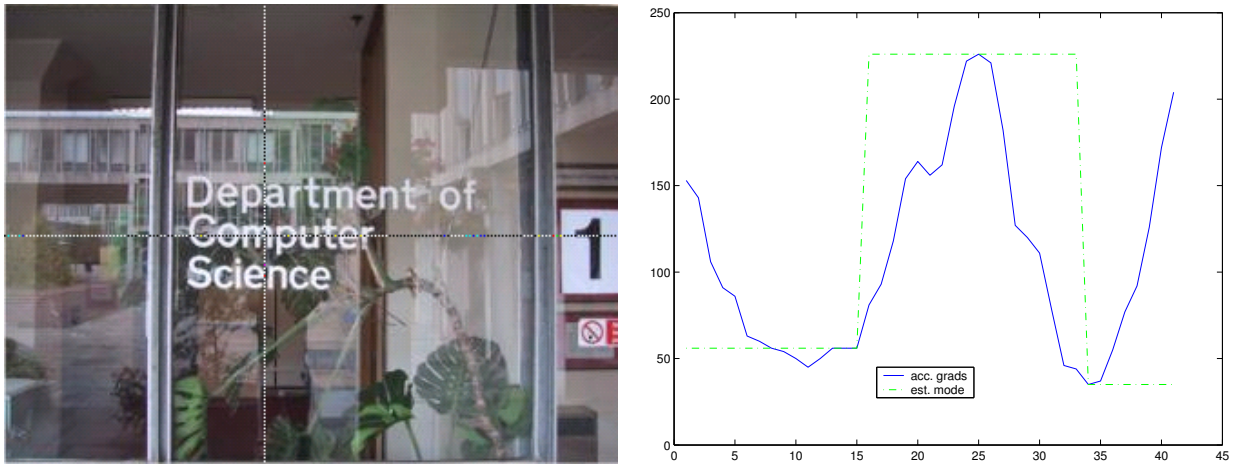
The text rectangle height is computed from the vertical profile of the first filter response, i.e. at each pixel a vertical interval of T pixels centered on the given pixel is considered. In this interval, we search for a mode (peak) containing the center pixel. Figure 3a shows an example image whose rectangle height is estimated at the pixel position indicated by the cross hair. Figure 3b shows the vertical interval of accumulated horizontal gradients with a visible peak in the center region of the interval (the center value corresponds to the examined pixel). The dashed-dotted line shows the estimated borders of the mode.

4.2.1 Text height estimation

The peak may be more or less flat, depending on the orientation of the text. For the horizontal filter, horizontal text creates a rectangular shaped peak, whereas slightly rotated text creates a flatter, more trapezoidal peak. The areas of the interval which are outside of the mode, are undefined.

The detection of peaks has already been tackled in the framework of edge detection. In contrast to classical step edges, "roof" edges or "ridge" edges are peak shaped functions. For instance, Ziou presents a solution for the detection of roof edges [47] which is derived from Canny's criteria of localization quality and detection quality [5]. These solutions are based on linear filtering with infinite impulse response followed by a maximum search in the filter response. In our case, we are not so much interested in the localization of the maximum of the peak, which may be anywhere in the text rectangle, but in the localization of the peak borders. The peak may either be situated in a flat environment, if the respective

[3]Since we integrate the gradient magnitude, we are not subject to the localization criteria as it was defined by Canny.



(a) (b)

Figure 3: Searching the mode in an interval (parameter $T = 41$): (a) the original image; (b) the vertical interval of gradients and the estimated mode.

rectangle lies in an unstructured background, or neighbor other peaks, if the text is part of several lines. Therefore, a simple model of the mode as a roof function with added white noise is not possible.

Instead of trying to find the closed form of an optimal filter kernel for a linear filter, we posed the peak detection as optimization problem over the space of possible peak borders. We exploit various properties of the situation at hand, based on the following assumptions:

- A high filter response inside the mode.
- A high contrast to the rest of the profile, i.e. the difference between the maximum of the mode and the values at the borders should be high.
- The size of the mode, which corresponds to the height of the text rectangle, needs to be as small as possible, in order to avoid the detection of multiple neighboring text rectangles.

Given an interval of N values $G_1 \dots G_N$ where the center value $G_{N/2}$ corresponds to the pixel to evaluate, the following values are possible for the mode borders a and b: $a \in [1, N/2-1]$, $b \in [N/2+1, N]$. The border values are estimated maximizing the following criterion:

$$(a,b) = \arg\max_{a,b} \left[\alpha_1 \left(1 - \frac{width}{N}\right) + \alpha_2 \frac{height}{\max_i(G_i) - \min_i(G_i)} + \alpha_3 \frac{\mu(G_i)}{\max_i(G_i)} \right]$$

where $width = b - a + 1$ is the width of the mode (i.e. height of the text rectangle),

$$height = \left[\max_{i \in [a+1, b-1]} (G_i) \right] - \frac{1}{2}(G_a + G_b)$$

is the height of the mode (i.e. the contrast of the text rectangle to its neighborhood), $\mu(G_i),\ i \in [a,b]$ is the mean of the mode values and the $\alpha_j,\ j \in 1..3$ are weights. The criterion searches for a mode with large height and mean but small width. The weights have been experimentally set to the following values:

$$\alpha_1 = 0.25 \quad \alpha_2 = 1.0 \quad \alpha_3 = 0.5$$

These weights emphasize the height criterion and still allow the width criterion to avoid the detection of multiple modes.

Note, that the three mode properties — height, width and mean — are combined additively instead of multiplicatively, as the criteria proposed by Canny. Multiplication favors configurations where all three properties are high. However, in our case we wanted to put the emphasis on high mode height, which is the main characteristic of text modes. The other two properties, width and mean, have been added to further increase the performance and to restrict the choice of borders to a single peak. This choice to combine the mode properties may be compared to the combination of multiple experts. In the case where some of the experts are weakly trained, i.e. less reliable than others, the sum rule of classifier combination is more powerful than the product rule [18].

(a)

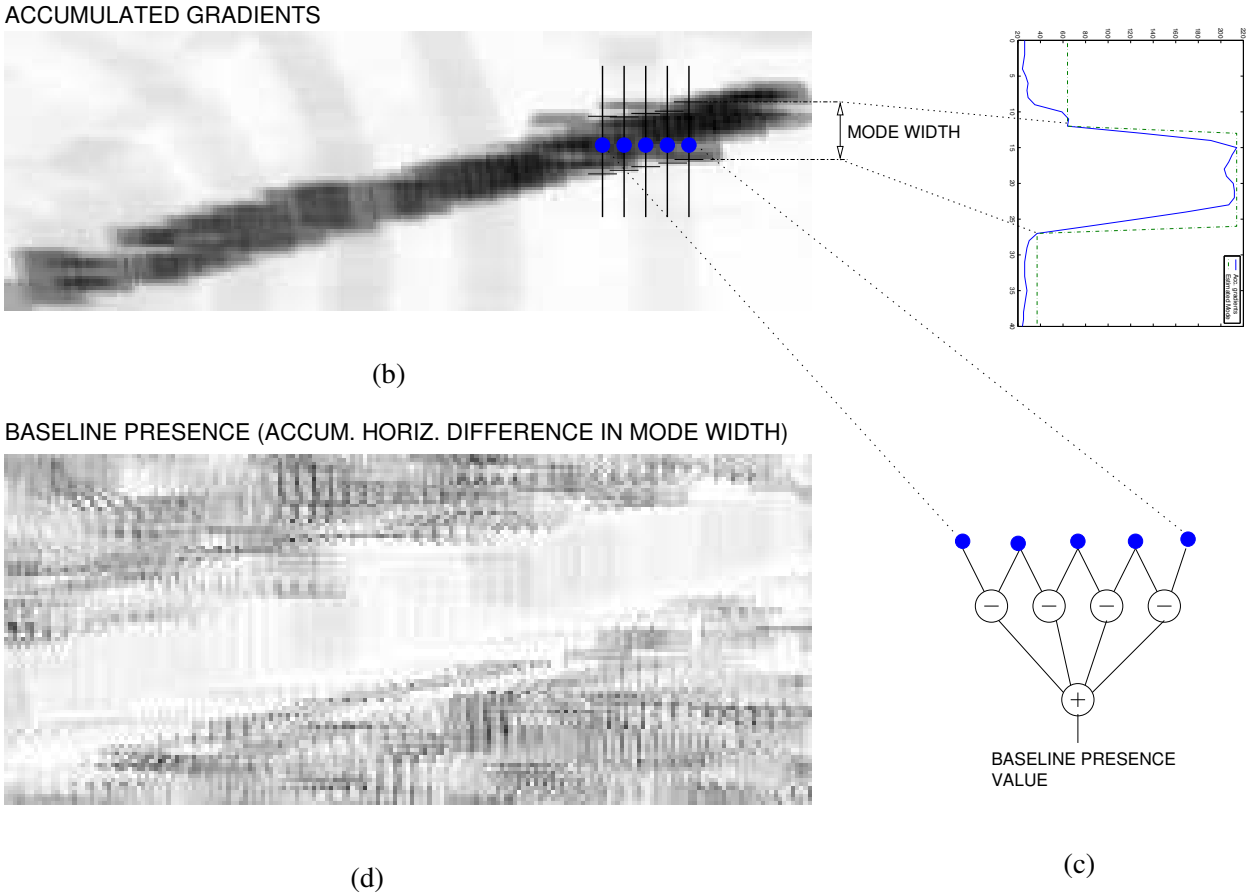


Figure 4: Combining the mode features of neighboring pixels: (a) the original image; (b) estimating the mode at different pixels; (c) calculating the accumulated difference of mode widths; (d) the baseline presence image.

# Values	Feature
1	Horizontally accumulated first derivative $\frac{\partial I}{\partial x}$.
1	The width of the detected mode.
1	The height of the detected mode.
1	The difference of the heights of the mode to the left and to the right border.
1	The mean of the gradient values in the mode.
1	The standard deviation of the gradient values in the mode.
1	The baseline presence (accumulated differences of the mode widths across several modes in a horizontal neighborhood).
7	Total per orientation
28	Total

Table 1: The contents of a feature vector.

4.2.2 Text height regularity

Once the mode is estimated, we can extract a number of features which we already used for its detection: width, height, mean, etc. Combining the properties of the modes of several neighboring pixels, we are able to extract features on a larger spatial neighborhood, which is schematically shown in Figure 4. Figure 4b shows the accumulated gradients of an example image. As already mentioned, around each pixel we want to classify, a vertical interval of accumulated gradients is considered and the mode in this interval is estimated. Then the variability of the mode width across horizontally neighboring pixels is verified (figure 4c) by calculating the difference in mode width between neighbors and accumulating this difference across a horizontal window of size S_w, where S_w is a parameter which depends on the text size:

$$\Delta W(x,y) = \sum_{i=-\lfloor S_w/2 \rfloor}^{\lfloor S_w/2 \rfloor} |W(x+i-1,y) - W(x+i,y)| \tag{2}$$

where $\Delta W(x,y)$ is the accumulated difference in mode width (which we call baseline presence) at pixel (x,y) and $W(x,y)$ is the mode width at pixel (x,y).

Figure 5 shows an example image and the resulting feature images, where the feature values are scaled for each feature independently into the interval $[0,255]$ in order to be able to display them as gray values, and the negative image is displayed. Note, that the mode width is approximately uniform in the text areas, which results in low accumulated mode width differences in this area, displayed as close to white in the respective feature image.

The final seven feature values corresponding to a single orientation are listed in table 1. The mode estimation features are not rotation invariant, although they are robust to slight changes up to 25^o. Therefore, we calculate the features for the four principal orientations of the image (horizontal, vertical, right diagonal, left diagonal) resulting in a $7 \times 4 = 28$ dimensional feature vector.

4.2.3 Reducing the computational complexity

For speed reasons, the classification is done on every 16^{th} pixel only, i.e. on every 4^{th} pixel in x direction and every 4^{th} pixel in y direction. The classification decisions for the other pixels are bi-linearly interpolated.

The most complex step during the feature calculation phase is the mode estimation. It is possible to perform this calculation only on pixels which are evaluated later in the classification phase, which reduces the complexity tremendously. However, the mode properties are reused to calculate the baseline presence feature (equation 2), so the calculation of this feature needs to be changed since the mode properties of the immediate neighbors of a pixel are not available anymore. Instead, the differences in mode width between the nearest horizontally neighbors with available mode properties are accumulated:

$$\Delta W(x,y) = \sum_{i=-\lfloor S_w/2 \rfloor}^{\lfloor S_w/2 \rfloor} |W(x+4(i-1),y) - W(x+4i,y)|$$

where of course the length parameter S_w needs to be adapted to the new situation. Figure 6 shows an example image and the baseline presence feature image with feature calculation on every pixel (6b) and with feature calculation on every 16^{th} pixel and bi-linear interpolation of the other pixels (6c). As can be seen, no significant difference can be noted between the two feature images.

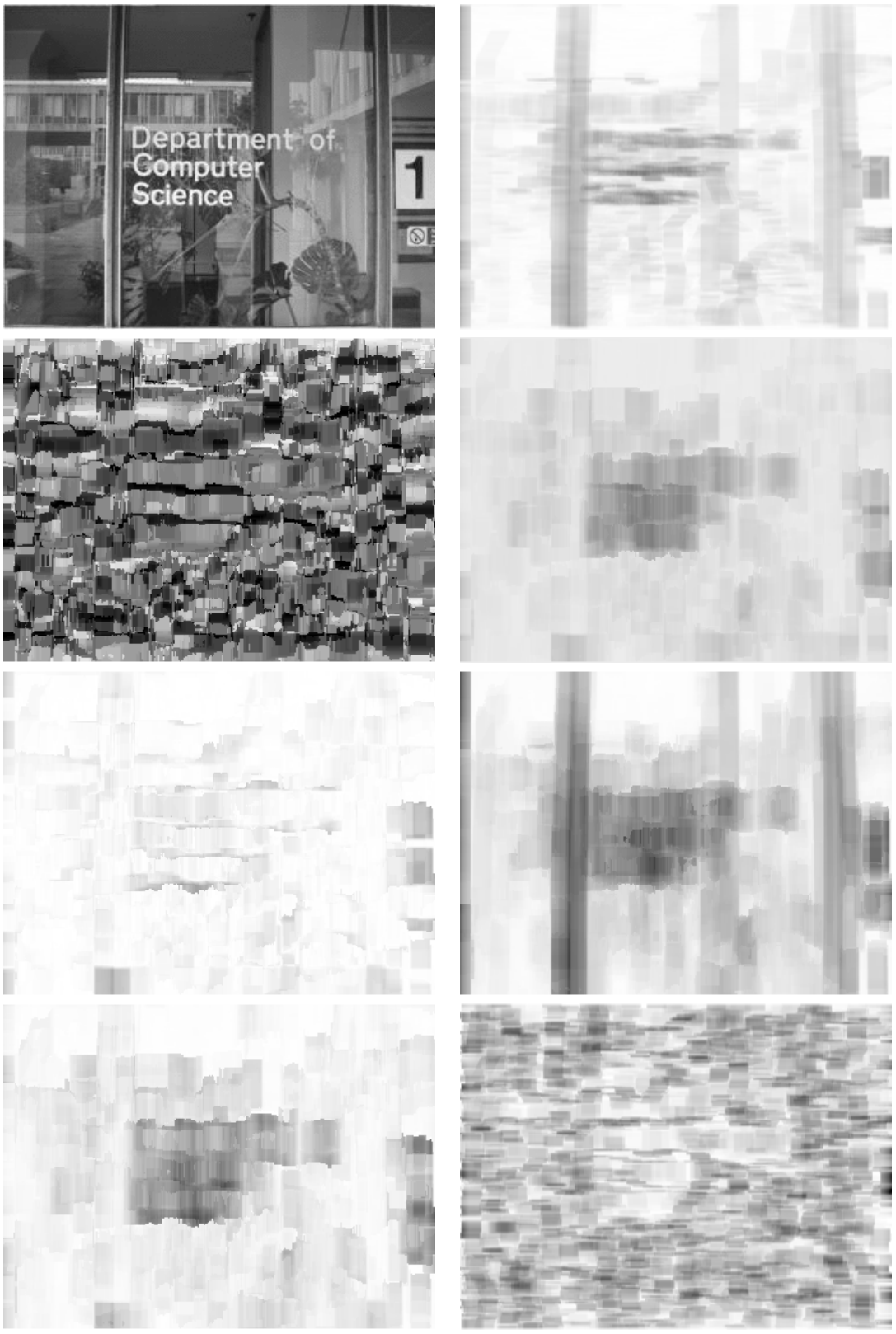
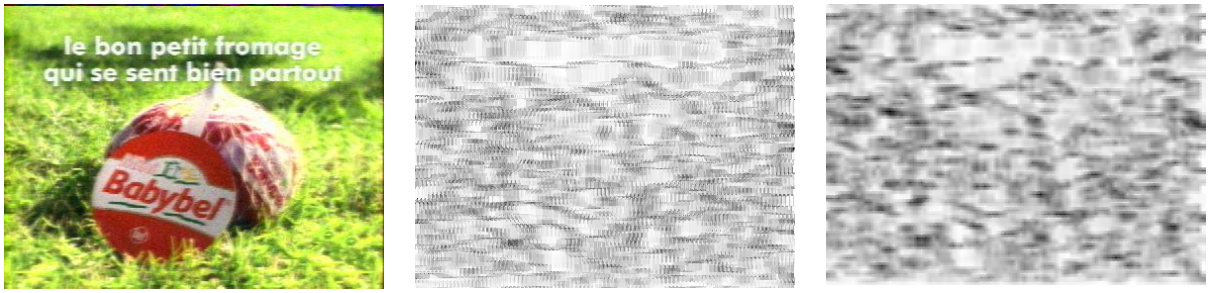


Figure 5: From left to right and top to bottom: The original image, the accumulated gradients (darker→ better), mode width, mode height (darker→ better), difference of mode heights (brighter→ better), mean (darker→ better), standard deviation (darker→ better), baseline presence (brighter→ better).



(a) (b) (c)

Figure 6: (a) example image; (b) the baseline presence with feature calculation at every pixel; (c) the baseline presence with feature calculation at every 16^{th} pixel only.

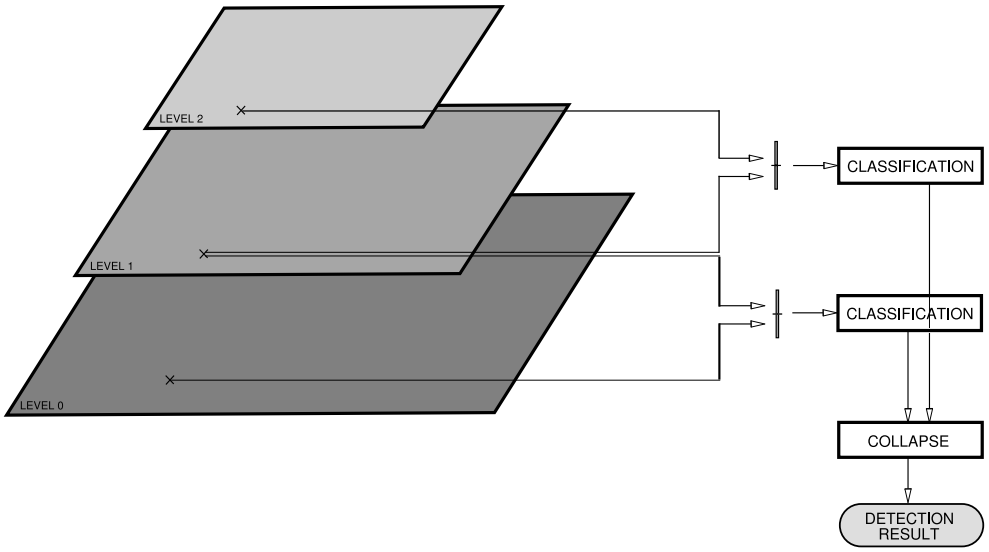


Figure 7: The hierarchical structure of the algorithm.

4.3 From classification to detection

The feature vectors of a training set are fed into a learning machine in order to learn the differences between the features of text and non-text, as we will explain in section 4.4. In this section we describe how the classification of the pixels is carried out on images and how it is translated into detection results.

In order to be able to detect text of various sizes in images of various sizes, the detection algorithm is performed in a hierarchical framework. Each input image forms the base of a Gaussian pyramid, whose height is determined by the image size. The classical detection approaches apply a classification step at each level of the pyramid and collapse the pyramid by combining these intermediate results (see for instance [40]).

The combination of the results of different levels is a difficult problem. Boolean functions as *AND* and *OR* have drawbacks: The *OR* function creates a response for each response on one of the levels of the pyramid, therefore large images with high pyramids tend to create many responses and many false alarms. The *AND* function only responds if the text is detected at all levels, and therefore eliminates the advantages of a hierarchical solution. We decided to partly delegate this problem to the learning machine by creating feature vectors which contain features taken from two levels. Figure 7 illustrates the principle: a Gaussian pyramid is built for the image as already stated above. As done in the classical approaches, classification is done at each level of the pyramid for each pixel of the image. However, each feature vector is doubled: it contains the features calculated on the level itself as well as the features calculated for the central parent pixel. Therefore the dimensions of the feature vectors are doubled from 28 to 56 dimensions.

Since a single feature vector now contains the features from two levels, the classification decisions are more robust to changes of text size. Text, which is normally detected at a certain level l, will also be

detected on a lower level $l-1$ with a higher probability, since the features for level l are included in the feature vector for level $l-1$.

Of course, the fact that we create vectors with features from two different levels does not solve the problem of finding a way to collapse the hierarchical decisions into a flat result. We chose a simple OR function for the combination of the results of the different levels. However, we do not combine the results on pixel level but on rectangle level. Hence, the detection of text on one level of the pyramid involves the following steps:

- Calculation of the features on each pixel of the level, using the neighborhood of the pixel itself as well as the neighborhood of the central parent pixel.
- Classification of the feature vector, i.e. of the pixel, using the learned model.
- Post processing of the decision image and extraction of text rectangles as bounding boxes of connected components. We perform the morphological and geometrical post-processing already described in [42].

The hierarchical detection and post-processing results in a list of detected rectangles per pyramid level. The final list of rectangles consists of the union of all rectangles, where each rectangle is projected to the base of the pyramid in order to normalize their sizes and positions. Overlapping and touching rectangles are merged according to fixed rules using the amount of overlap area (see [42] for details).

4.4 Learning

The feature vectors given in the previous section have been designed to distinguish single pixels between text and non-text. It is the task of learning machines to learn this distinction from training data, i.e. from a set of positive and negative examples.

From the large pool of existing learning machines, we chose support vector machines (SVM) for the task of classification between the two classes. Lately, they received tremendous attention in the learning community and have been applied successfully to a large class of pattern recognition problems, where they performed better then competing techniques, e.g. artificial neural networks.

A major advantage of SVMs is their smaller sensitivity to the number of dimensions of the feature space (the curse of dimensionality), which hurts the performance of neural networks. Indeed, whereas a reduction of the dimensionality of the feature space with tools as e.g. the principal components analysis is crucial when "traditional" learning techniques are employed, learning with SVM often does not require this reduction.

Support Vector Machine learning has been introduced by Vapnik to tackle the problem of learning with small data sets. The interesting point which distinguishes SVM learning from classical neural network learning is the fact, that SVMs minimize the generalization error using a principle called the structural risk minimization. A detailed introduction would be too long for this article, the interested reader is referred to [4] for a tutorial on SVM learning or Vapnik's books for a detailed treatment of the theory [38][39].

In this work, we conducted ν-SVM learning[33] instead of classical $C-$SVM learning. In ν-SVM learning, the classical training parameter C, which depends on the dimension of the problem and the size of the training set, is replaced by a a new normalized parameter $\nu \in [0,1]$. This parameter is independent of the size of the data set, which allows to estimate it on a smaller training set before training the classifier on a large training set.

4.4.1 Reducing the complexity

Support vector machines are currently significantly slower than learning machines with a similar generalization performance. The complexity of the classification process is proportional to the dimension of the problem and the number of support vectors in the model. Since the dimension of the problem cannot always be changed easily, the key to the reduction of the computational complexity is a reduction of the number of support vectors.

Different methods exist in literature for this purpose. Burges proposes the minimization of the quadratic error of the initial hyper plane and a new one with a — fixed — lower amount of vectors which are not necessarily data vectors [3]. The method has the disadvantage of being difficult to implement. Osuna and Girosi propose the usage of SVM regression to approximate the original hyper plane with a new function with less support vectors [30]. We used this approach for the reduction of our model. The algorithm to reduce the model can be outlined as follows:

1. Train the full classification model with a given kernel and parameters.

2. Run epsilon support vector machine regression (SVMR) on the hyper plane evaluated at the support vectors, i.e. $(\boldsymbol{s_i}, f(\boldsymbol{s_i}))$. This results in a new hyper plane with fewer support vectors.

The parameters of the SVMR algorithm are C (the sum of the slack-variables as in C-SVM classification) and the ϵ for Vapnik's ϵ-insensitive cost function defined as:

$$|x|_\epsilon = \begin{cases} 0 & \text{if } |x| < \epsilon \\ |x| - \epsilon & \text{else} \end{cases}$$

The two parameters define the accuracy of the approximation and therefore the performance of the reduced classifier and the number of support vectors.

4.5 The training algorithm

The learning machine as described above needs a training set with positive and negative samples in order to learn to model. In the case of object detection problems, and as a special case text detection problems, the positive examples are not hard to determine. On the other hand, which samples shall be chosen as negative samples? Practically speaking, the size of the training set is limited since the complexity of the training algorithm grows non-linearly with the number of the training samples. Hence, the negative samples need to be chosen wisely in order to represent the class of non-text as closely and completely as possible.

To tackle this problem, we employed the well known bootstrapping approach for the selection of the training set, i.e. the negative samples depend on the positive ones and are chosen in an iterative algorithm. as follows:

1. The initial training set consists of the positive training sample set T_P and $\frac{1}{K}|T_P|$ randomly chosen vectors from a large set of negative samples T_N, where K is the number of bootstrap iterations.

2. Training is performed on the training set.

3. The returned model is applied to the rest of the negative samples and the correctly classified samples are removed from this set. From the remaining set, the $\frac{1}{K}|T_P|$ samples with the smallest distance to the separating hyperplane are added to the training set.

4. Go to step 2 until the number of iterations has been performed.

During the training phase, we perform another iterative algorithm: We employ n-fold cross validation in order to estimate the generalization error of the model we obtained by the learning process. The two iterative processes, bootstrapping and N-fold cross validation, are combined into a single training algorithm as follows:

1. Create two sets of training samples: a set T_P of positive samples and a large set T_N of negative samples.

2. Shuffle the two sets and partition each set into N distinctive and non-zero subsets of sizes $\frac{1}{N}|T_P|$ and $\frac{1}{N}|T_N|$, respectively, where N is the parameter from the N-fold cross validation.

3. $i = 1$.

4. For each of the two sets, choose subset i.

5. Train the SVM on the two sets using the iterative bootstrapping method described above.

6. Test the model on the samples not chosen in step 4 and calculate the error.

7. $i = i + 1$.

8. Go to step 4 until the number of iterations is reached (i.e. i=N).

At each iteration, $\frac{N-1}{N}(|T_P| + |T_N|)$ samples are used for training and $\frac{1}{N}(|T_P| + |T_N|)$ samples are used for testing. The final generalization error is computed as the mean of the errors computed in step 6 over all iterations.

5 Experimental Results

To estimate the performance of our system on still images, we used a database containing 384 ground-truthed images in format CIF (384×288 pixels), among which 192 contain text and 192 do not contain text. The former image set contains a mixture of 50% scene text and 50% artificial text. To test the system on video sequences, we carried out exhaustive evaluations using a video database containing 60.000 frames in 4 different MPEG 1 videos with a resolution of 384×288 pixels. The videos provided by INA[4] contain 323 appearances of artificial text from the French television channels TF1, France 3, Arte, M6 and Canal+. They mainly contain news casts and commercials.

As already stated, the proposed method consists of a classification step and post-processing step (morphological and geometrical post-processing and combination of the levels of the pyramid). Hence, the experiments and the evaluation of the detection algorithm need to be conducted on two different levels:

- An evaluation on pixel level, i.e. feature vector level, which evaluates the discrimination performance of the features and the performance of the classifier. We also based the choice of the learning parameter of the learning machine and the choice of the kernel and its parameters on the results of this evaluation, in order to keep the choice independent of the parameters of the post-processing step.
- An evaluation on text rectangle level.

The experiments performed on the two levels of evaluation are described in the next two sub sections.

5.1 Classification performance

The estimation of the classification error with 5-fold cross validation helped us to determine various choices:

- The choice of the kernel;
- The kernel parameter;
- The learning parameter ν;
- The reduction parameters C and ϵ for the reduction of the number of support vectors;

For speed reasons, we carried out the evaluation on two training sets of different sizes. For the selection of the kernel, its parameters and the training parameters, we chose for each feature type 3,000 vectors corresponding to text pixels (positive examples) and 100,000 feature vectors corresponding to non-text pixels (negative examples). In this step, we determined an optimal polynomial kernel of degree 6 and a learning parameter of $\nu = 0.37$. Once the optimal parameters were selected, we trained the system with 50,000 positive feature vectors and 200,000 negative samples. All learning was done with combined boot strapping (3 iterations) and 5-fold cross validation as described in section 4.5, i.e. not all negative samples were actually used in the final training set.

Table 2 shows the effect of the reduction of the number of support vectors. The full, unreduced trained SVM model contains 22,100 support vectors, which results in very high classification complexity. The classification of an image of size 384×288 pixels takes around 15 minutes on a Pentium III-700 (if only every 4^{th} pixel in each dimension, i.e. every 16^{th} pixel is classified!). However, by applying the technique described in section 4.4.1, a large reduction of the number of support vectors can be achieved without a significant loss in classification performance. A reduction from 22,100 support vectors to 806 support vectors decreases the classification performance only by 0.5 percentage points (see table 2a). The tradeoff between the number of support vectors and the classification performance can be controlled conveniently by tuning the parameter ϵ of the model reduction algorithm. More information on the run-time complexity of the classification algorithm for different models with numbers of support vectors is given in sub section 5.4.

The classification performance figures given above have been achieved calculating the features for each pixel. However, as explained in sub section 4.2.3 and shown in figure 6, the calculation may be accelerated by calculating the features on each 4^{th} pixel in each dimension only. The classification results for this calculation mode are given in table 2b. The final model we used throughout this work is the one reduced with parameters $C = 100$ and $\epsilon = 1.0$ resulting in 610 support vectors.

Finally, figure 8 shows three result images and the text detection results without post processing. The result images are calculated on the original scale of the image (as opposed to the feature vectors, which are calculated on two scales).

[4]The *Institut National de l'Audiovisuel* (INA) is the French national institute in charge of the archive if the public television broadcasts. See http://www.ina.fr

C	ϵ	Recall	Precision	H.Mean	#SVs
no reduction		81.0	96.0	**87.9**	22100
1000	0.01	80.5	95.9	87.5	2581
	0.1	80.3	95.8	87.4	806
	1	75.5	96.6	84.8	139
10000	0.01	80.5	95.9	87.5	2581
	0.1	80.3	95.8	87.4	806
	1	75.5	96.6	84.8	139

(a)

C	ϵ	Recall	Precision	H.Mean	#SVs
no reduction		84.0	93.2	88.4	19058
100	1.0	**78.3**	**93.0**	**85.0**	**610**
100	1.5	69.3	93.7	79.7	236

(b)

Table 2: The effect of the reduction of the number of support vectors on the classification performance: (a) feature calculation for each pixel; (b) partial calculation of the features only (each 16^{th} pixel).

5.2 Detection performance in still images

In contrast to an evaluation on pixel level, for object detection systems (and as a special case, text detection systems), the notion of "the object has been detected" is not well-defined. The question cannot be answered with a simple "yes" or "no", since objects may be partially detected. Therefore, the familiar precision/recall measures need to be changed to incorporate the quality of the detection.

5.2.1 Evaluation measures

Unfortunately, until now there is no widely used evaluation scheme which is recognized by the scientists of the domain. We therefore used different evaluation schemes, among which are schemes proposed by other researchers and our own techniques which address some short comings of the already existing ones:

The ICDAR measure A simple but effective evaluation scheme has been used to evaluate the systems participating at the text locating competition in the framework of the 7^{th} International Conference on Document Analysis and Recognition (ICDAR) 2003 [26]. The two measures, recall and precision, are changed slightly in order to take into account the amount of overlap between the rectangles: If a rectangle is matched perfectly by another rectangle in the opposing list, then the match functions evaluate to 1, else they evaluate to a value < 1 proportional to the overlap area.

The ICDAR evaluation scheme has several drawbacks:

- Only one-to-one matches are considered. However, in reality sometimes one ground truth rectangle may correspond to several text rectangles or vice-versa.
- The amount of overlap between two rectangles is not a perceptively valid measure of detection quality (see figure 12).
- The inclusion of overlap information into the evaluation measures leaves room for ambiguity: a recall of 50% could mean that 50% of the ground truth rectangles have been matched perfectly, or that all ground truth rectangles have been found but only with an overlap of 50%, or anything in between these two extremes.

The CRISP measure We developed an evaluation scheme which addresses these problems. Inspired by the method presented in [23], it takes into account one-to-one as well as many-to-one and one-to-many matches. However, the algorithm aims at an exact determination — controlled by thresholds — whether a ground truth rectangle has been detected or not.

One-to-one matches need to fulfill an additional geometrical constrained in order to be validated: the differences of the left (respectively right) coordinates of the rectangles need to be smaller than a threshold which depends on the width of the ground truth rectangle. This constraint, which does not depend on the overlap information, avoids the stituation that a rectangle is considered as being detected, although a significant part of its width is missing.

(a) (b)

Figure 8: Detection without post-processing: (a) original images; (b) classification results.

These adapted precision and recall measures provide an intuitive impression on how many rectangles have been detected correctly and how many false alarms have been produced. Please note that text which is only partly detected and therefore not matched against a ground truth rectangle will decrease the precision measure, in contrast to the ICDAR evaluation scheme.

Details on these text detection evaluation schemes and other evaluation methods together with their advantages and disadvantages are given in [41].

5.2.2 Generality

As for information retrieval (IR) tasks, the measured performance of an object detection algorithm highly depends on the test database. It is obvious that the nature of the images determines the performance of the algorithm. As an example we could think of the text type (artificial text or scene text), its size, the image quality, noise, fonts and styles, compression artifacts etc. On the other hand, the nature of the images is not the only variable which determines the influence of the test database on the detection performance. The structure of the data, i.e. the ratio between the relevant data and the irrelevant data, is a major factor which influences the results. In [13], Huijsmans et al. call attention to this fact and adapt the well known precision/recall graphs in order to link them to the notion of generality for an IR system, which is defined as follows:

$$\text{Generality}_{IR} = \frac{\text{number of relevant items in the database}}{\text{number of items in the database}}$$

Very large databases with low generality, i.e. much irrelevant clutter compared to the relevant material, produce results with lower precision than databases with higher generality. However, unlike IR tasks, text detection algorithms do not work with items (images, videos or documents). Instead, images (or videos) are used as input, and text rectangles are retrieved. Nevertheless, a notion of generality can be defined as the amount of text which is present in the images of the database. We define it to be

Dataset	#†	G††	Eval. scheme	Recall	Precision	H. mean
Figure 9a:	6	6.63	ICDAR	48.8	49.7	49.3
Text			CRISP	52.8	47.3	49.9
Figures 9a+9b:	12	3.5	ICDAR	49.1	37.4	42.5
Text + no text			CRISP	63.8	47.2	54.3
Artificial text +	144	1.49	ICDAR	54.8	23.2	32.6
no text			CRISP	59.7	23.9	34.2
Artificial text +	384	1.84	ICDAR	45.1	21.7	29.3
scene text + no text			CRISP	47.5	21.5	29.6

†Number of images
††Generality

Table 3: The detection results of the learning based method for different datasets and different evaluation schemes.

$$\text{Generality} = \frac{\text{Number of text rectangles in the database}}{\text{Number of images in the database}} \tag{3}$$

Another difference to IR systems is the lack of a result set window, because all detected items are returned to the user. Therefore, the generality of the database does influence precision, but *not* recall. Thus, the influence of the database structure on the system performance can be shown with simple two-dimensional precision/generality graphs.

Finally, a decision needed to be made concerning the generality level of the database when result tables or graphs are displayed which contain a fixed level a generality. In other words, we needed to decide how many images with zero ground truth (no text present) should be included in the database. We concluded, that a mixture of 50% images with text and 50% images without text should be a reasonable level. We should keep in mind that this amount is not representative for realistic video streams. However, a larger part of non-text images would introduce a higher bias into the detection system. The detection results for realistic videos are given in sub section 5.3.

5.2.3 Results

Figure 9 shows the performance of the detection system on 6 images containing text and 6 images not containing any text. The images have been chosen randomly from our image database, which is further illustrated by the first two segments of table 3 which give the performance measures of the system applied to these images. The lower two segments of table 3 give the performance figures for the whole dataset, which are comparable to the ones for the 12 images. The shown examples illustrate the good detection performance of the system.

The dependence of CRISP precision on the generality of the dataset is given in figures 10a and 10b for, respectively, the dataset containing artificial text only and the dataset containing both types of text. We remark that the precision/generality curves are very flat, which illustrates an excellent behaviour when applied to sources of low generality, as e.g. video sequences.

Figure 11 illustrates the dependence of the CRISP performance measure on the internal thresholds used by this evaluation method. In order to determine whether a rectangle has been detected or not, the amount of detected rectangle surface is thresholded. In other words, if a rectangle has been detected partially only, then it's surface recall is < 1. If the surface recall is below a threshold, the detection is rejected. Similarily, if the detected rectangle is bigger than the ground truth rectangle, then the surface precision is < 1. Figure 11 presents the CRISP recall and precision, i.e. measures on rectangle level, w.r.t. to these thresholds.

As can be seen in figure 11a, the evaluated detection performance only slightly depends on the surface recall thresholds. This may be explained by the fact, that most text is detected with a slightly larger rectangle than the retangle given in the ground truth.

On the other hand, the detection performance significantly depends on the surface precision thresholds. When a surface precision of 100% per rectangle is enforced, the performance drops to zero. Unfortunately, as explained before, the surface precision per rectangle alone is not a perceptively valid measure capable of deciding whether the detection performance is sufficiently precise. As an example, figure 12 shows two different detection results with a precision value of only 77.1%. This relatively low value is astonishing

(a)

(b)

Figure 9: Some detection examples: (a) images with text; (b) images without text.

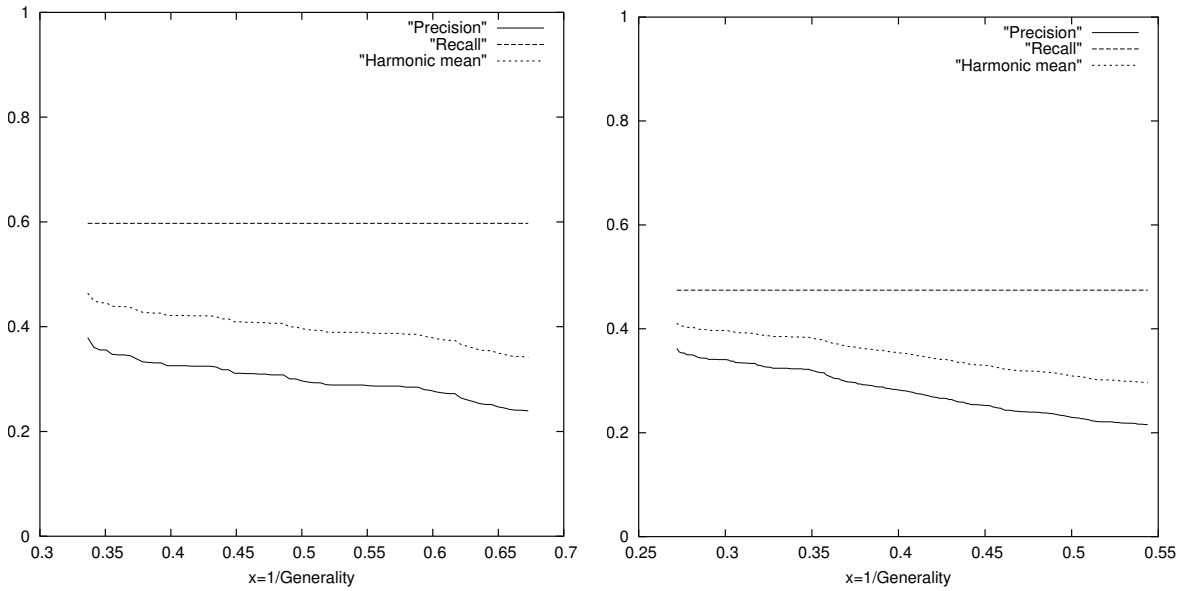


Figure 10: Precision for different generalities. Reciprocal generality is displayed on the x-axis: (a) artificial text + no text; (b) artificial text + scene text + no text.

	Video files				
Category	#1	#2	#3	#4	Total
Classified as text	92	77	55	60	284
Total in ground truth	110	85	63	64	322
Recall (%)	83.6	90.6	87.3	93.8	**88.2**
Positives	161	121	106	125	513
Total detected	209	138	172	165	684
Precision (%)	77.0	87.7	61.6	75.8	**75.0**
Harmonic mean (%)	80.2	89.1	72.3	83.8	**81.1**
Generality	0.80	1.04	0.85	0.70	0.85
Ratio text frames	0.39	0.32	0.31	0.40	0.34

Table 4: The detection results for video sequences given on text object level.

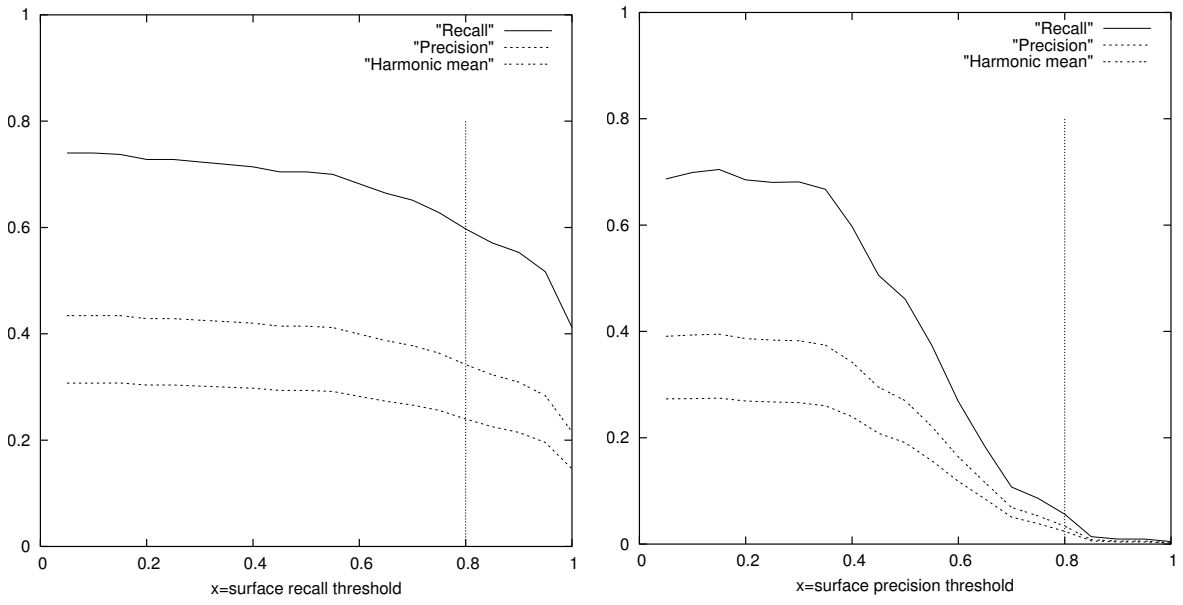
given the detection in figure 12a, which is perceived as rather precise. For this reason we set the "standard" threshold throughout this work to a surface precision of 0.4, a rather small value. However, we added an additional constraint through an additional threshold on the differences of the x-coordinates of the ground truth rectangle and the detected rectangle.

5.3 Detection performance in video sequences

Table 4 shows the results for the detection in video sequences. We achieve an overall detection rate of 88.2% of the text appearing in the video. The remaining 11.8% of missing text are mostly special cases, which are very difficult to treat, or text with very weak contrast. Comparing these results with the ones of our previous system [42], we remark a small drop in recall (93.5% → 88.2%), which can be explained by the change from a system based on heuristics to a system based on learning. However, detection precision is significantly higher (34.4% → 75.0%).

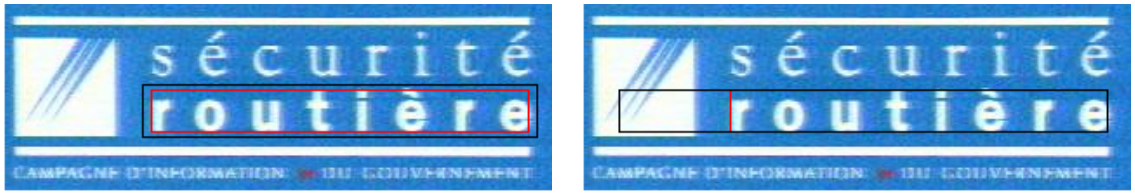
5.4 Execution time

The execution time of the algorithm implemented in C++ on a Pentium III with 700 Mhz running under Linux is shown in table 5. The execution time has been measured for an input image in CIF format, i.e. of size 384×288 pixels. As already mentioned, the classification largely depends on the number of support



(a) (b)

Figure 11: The system performance evaluated with the CRISP method for different evaluation thresholds (the vertical bars indicate the threshold chosen throughout this work): (a) changing the thresholds related to surface recall; (b) changing the thresholds related to surface precision.



(a) (b)

Figure 12: The ground truth rectangle and detected rectangles for an example image. Surface precision and recall for figures (a) and (b) are equivalent.

vectors, which in turn depends on the parameters of the regression algorithm used to approximate the decision function. The classification time can be reduced from 8 minutes and 49 seconds with the full model down to 3 seconds with a very reduced model. The final model we chose is the one with 610 support vectors. The times and performances given in table 5 have been achieved by calculating the mode feature properties only on the pixels which are classified.

6 Conclusion

In this article we proposed a method to detect and process text in images and videos. We propose an integral approach for the detection in video sequences, beginning with the localization of the text in single frames, tracking, multiple frame enhancement and the binarization of the text boxes before they are passed to a commercial OCR.

The detection method exploits several properties of text. Among these are geometrical properties, which are fed into the detection algorithm at an early stage, in contrast to other existing algorithms which enforce the geometrical constraints in a post processing phase.

The perspectives of our work on detection are further improvements of the features, e.g. normalization to the overall contrast in the image, integration of geometrical texture analysis [19] and an improvement of the run-time complexity.

A further perspective of our work is the generalization of the method to a larger class of text. Until now, we did not restrict ourselves to artificial text. However, generally oriented scene text is not yet fully supported. Although the features and the classification algorithm itself have been designed for multiple

Model	Full	reduced	reduced
# Support vectors	19,058	**610**	236
Feature calculation (sec)	1	**1**	1
Classification (sec)	528	**18**	3
Total	529	**19**	4
Classification Recall (%)	84.0	**78.3**	69.3
Classification Precision (%)	93.2	**93.0**	93.7
Classification H. mean (%)	88.4	**85.0**	79.7

Table 5: Execution time on a Pentium III with 700 Mhz for different SVM model sizes. The final model is the one with 610 support vectors.

orientations, the post-processing steps need to be adapted. For the detection of distorted scene text, a module for the detection and the correction of text orientation and skew needs to be conceived.

As already mentioned, our work already has been extended to linear movement of the movie casting type [28]. A further extension to more complex movement might be envisioned.

The text detection and extraction technology itself seems to have reached a certain maturity. We believe that the future will show that the integration of different methods (e.g. structural and statistical methods) will further boost the detection performance, especially in cases where the type of data is not known beforehand. For instance, in a hierarchical framework, structural methods on lower levels might confirm detections done on higher levels by texture or edge based methods.

References

[1] L. Agnihotri and N. Dimitrova. Text detection for video analysis. In *IEEE Workshop on Content-based Access of Image and Video Libraries*, pages 109–113, 1999.

[2] S. Brès and V. Eglin. Extraction de textes courts dans les images et les documents à graphisme complexe. In *Colloque International Francophone sur l'Ecrit et le Document, Lyon*, pages 31–40, 2000.

[3] C.J.C. Burges. Simplified support vector decision rules. In *Proceedings of the 13th International Conference on Machine Learning*, pages 71–77, 1996.

[4] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] J.F. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[6] D. Chen, J.M. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. *Pattern Recognition*, 37(3):595–608, 2004.

[7] P. Clark and M. Mirmehdi. Combining Statistical Measures to Find Image Text Regions. In *Proceedings of the International Conference on Pattern Recognition*, pages 450–453, 2000.

[8] D. Crandall and R. Kasturi. Robust Detection of Stylized Text Events in Digital Video. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 865–869, 2001.

[9] L. Gu. Text Detection and Extraction in MPEG Video Sequences. In *Proceedings of the International Workshop on Content-Based Multimedia Indexing*, pages 233–240, 2001.

[10] Y.M.Y. Hasan and L.J. Karam. Morphological text extraction from images. *IEEE Transactions on Image Processing*, 9(11):1978–1983, 2000.

[11] H. Hase, T. Shinokawa, M. Yoneda, M. Sakai, and H. Maruyama. Character string extraction from a color document. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 75–78, 1999.

[12] O. Hori. A video text extraction method for character recognition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 25–28, 1999.

[13] N. Huijsmans and N. Sebe. Extended Performance Graphs for Cluster Retrieval. In *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, volume 1, pages 26–32, 2001.

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

[15] J.M. Jolion. The deviation of a set of strings. *Pattern Analysis and Applications*, 2004. (to appear).

[16] K. Jung. Neural network-based text location in color images. *Pattern Recognition Letters*, 22(14):1503–1515, 2001.

[17] K.I. Kim, K. Jung, S.H. Park, and H.J. Kim. Support vector machine-based text detection in digital video. *Pattern Recognition*, 34(2):527–529, 2001.

[18] J. Kittler, M. Hatef, and J. Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[19] P. Kruizinga and N. Petkov. Nonlinear operator for oriented texture. *IEEE Transactions on Image Processing*, 8(10):1395–1407, 1999.

[20] F. LeBourgeois. Robust Multifont OCR System from Gray Level Images. In *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 1–5, 1997.

[21] C.M. Lee and A. Kankanhalli. Automatic extraction of characters in complex scene images. *International Journal of Pattern Recognition and Artificial Intelligence*, 9(1):67–82, 1995.

[22] H. Li and D. Doermann. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[23] J. Liang, I.T. Phillips, and R.M. Haralick. Performance evaluation of document layout analysis algorithms on the UW data set. In *Document Recognition IV, Proceedings of the SPIE*, pages 149–160, 1997.

[24] R. Lienhart. Automatic Text Recognition for Video Indexing. In *Proceedings of the ACM Multimedia 96, Boston*, pages 11–20, 1996.

[25] R. Lienhart and A. Wernike. Localizing and Segmenting Text in Images, Videos and Web Pages. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[26] S.M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 robust reading competitions. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, volume 2, pages 682–687, 2003.

[27] V.Y. Mariano and R. Kasturi. Locating uniform-colored text in video frames. In *Proceedings of the International Conference on Pattern Recognition*, volume 4, pages 539–542, 2000.

[28] D. Marquis and S. Brès. Suivi et amélioration de textes issus de génériques vidéos. In *Journées d´Études et d´Echanges "Compression et Réprésentation des Signaux Audiovisuels"*, pages 179–182, 2003.

[29] G. Myers, R. Bolles, Q.T. Luong, and J. Herson. Recognition of Text in 3D Scenes. In *Fourth Symposium on Document Image Understanding Technology, Maryland*, pages 85–99, 2001.

[30] E. Osuna and F. Girosi. Reducing the run-time complexity in support vector machines. In C.Burges B. Schölkopf and A. Smola, editors, *Advances in Kernel Methods: Support Vector Learning*, pages 271–284. MIT Press, Cambridge, MA, 1999.

[31] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

[32] T. Sato, T. Kanade, E.K. Hughes, M.A. Smith, and S. Satoh. Video OCR: Indexing digtal news libraries by recognition of superimposed captions. *ACM Multimedia Systems: Special Issue on Video Libraries*, 7(5):385–395, 1999.

[33] B. Schölkopf, A. Smola, R.C. Williamson, and P.L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

[34] B.K. Sin, S.K. Kim, and B.J. Cho. Locating characters in scene images using frequency features. In *Proceedings of the International Conference on Pattern Recognition*, volume 3, pages 489–492, 2002.

[35] K. Sobottka, H. Bunke, and H. Kronenberg. Identification of text on colored book and journal covers. In *Proceedings of the 5th International Conference on Document Analysis and Recognition*, pages 57–62, 1999.

[36] C.L. Tan and P.O. NG. Text extraction using pyramid. *Pattern Recognition*, 31(1):63–72, 1998.

[37] X. Tang, X. Gao, J. Liu, and H. Zhang. A Spatial-Temporal Approach for Video Caption Detection and Recognition. *IEEE Transactions on Neural Networks*, 13(4):961–971, 2002.

[38] V.N. Vapnik. *Statistical Learning Theory.* John Wiley & Sons, Inc., 1998.

[39] V.N. Vapnik. *The Nature of Statistical Learning Theory.* Springer Verlag, 2nd edition, 2000.

[40] A. Wernike and R. Lienhart. On the segmentation of text in videos. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) 2000*, pages 1511–1514, 2000.

[41] C. Wolf. *Text detection in images taken from video sequences for semantic indexing.* PhD thesis, INSA de Lyon, 20, Av. Albert Einstein, 69621 Villeurbanne cedex, France, 2003.

[42] C. Wolf and J.-M. Jolion. Extraction and recognition of artificial text in multimedia documents. *Pattern Analysis and Applications*, 2004. (to appear).

[43] E.K. Wong and M. Chen. A new robust algorithm for video text extraction. *Pattern Recognition*, 36(6):1397–1406, 2003.

[44] V. Wu, R. Manmatha, and E.M. Riseman. Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1224–1229, 1999.

[45] Y. Zhong, H. Zhang, and A.K. Jain. Automatic Caption Localization in Compressed Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):385–392, 2000.

[46] J. Zhou and D. Lopresti. Extracting text from www images. In *Proceedings of the 4th International Conference on Document Analysis and Recognition*, pages 248–252, 1997.

[47] D. Ziou. Line detection using an optimal IIR filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(6):465–478, 1991.

Christian WOLF received his Master's degree (the Austrian Dipl. Ing.) in computer science in 2000 from the Vienna University of Technology and his PhD in computer science in 2003 from the National Institute for Applied Science (INSA) of Lyon (France). He is currently with the Lyon Research Center for Images and Intelligent Information Systems, France. His research interests include image and video indexing and contents extraction from multimedia documents. (more details at http://rfv.insa-lyon.fr/~wolf)

Jean-Michel JOLION received the *Diplôme d'ingénieur* in 1984, and the PhD in 1987, both in Computer Science from the National Institute for Applied Science (INSA) of Lyon (France). From 1987 to 1988, he was a staff member of the Computer Vision Laboratory, University of Maryland (USA). From 1988 to 1994 he was appointed as an Associate Professor at the University Lyon 1 (France). From 1994, he has been with INSA and the Lyon Research Center for Images and Intelligent Information Systems where he is currently Professor of computer science. His research interests belong to the computer vision and pattern recognition fields. He has studied robust statistics applied to clustering, multiresolution and pyramid techniques, graphs based representations ... mainly applied to the image retrieval domain. Professor Jolion is a member of IAPR and IEEE (M'90). (more details at http://rfv.insa-lyon.fr/~jolion)