AdaBoost Learning for Detecting and Reading Text in City Scenes.

X. Chen and A.L. Yuille
Dept. Statistics
University of Los Angeles
Los Angeles, CA 90095.
yuille@stat.ucla.edu

Preliminary version. To appear as oral presentation in Computer Vision and Pattern Recognition 2004. Not for distribution.

Abstract

This paper gives an algorithm for detecting and reading text in natural images. The algorithm is intended for use by blind and visually impaired subjects walking through city scenes. We first obtain a dataset of city images taken by blind and normally sighted subjects. From this dataset, we manually label and extract the text regions. Next we perform statistical analysis of the text regions to determine which image features are reliable indicators of text and have low entropy (i.e. feature response is similar for all text images). We obtain weak classifiers by using joint probabilities for feature responses on and off text. These weak classifiers are used as input to an AdaBoost machine learning algorithm to train a strong classifier. In practice, we trained a cascade with 4 strong classifiers containg 79 features. An adaptive binarization and extension algorithm is applied to those regions selected by the cascade classifier. An commercial OCR software is used to read the text or reject it as a non-text region. The overall algorithm has a success rate of over 90% (evaluated by complete detection and reading of the text) on the test set and the unread text is typically small and distant from the viewer.

1. Introduction

This paper presents an algorithm for detecting and reading text in city scenes. This text includes stereotypical forms – such as street signs, hospital signs, and bus numbers – as well as more variable forms such as shop signs, house numbers, and billboards. Our database of city images were taken in ZZZZ partly by normally sighted viewers and partly by blind volunteers who were accompanied by sighted guides (for safety reasons) using automatic camera settings and little practical knowledge of where the text was located in the image. The databases have been labelled to enable us to train part of our algorithm and to evaluate the algorithm performance.

The first, and most important, component of the algorithm is a strong classifier which is trained by the AdaBoost learning algorithm [4],[19],[20] on labelled data. AdaBoost requires specifying a set of features from which to build the strong classifier. This paper selects this feature set guided by the principle of *informative features* (the feature set used in [19] is not suitable for this problem). We calculate joint probability distributions of these feature responses *on* and *off* text, so weak classifiers can be obtained as log-likelihood ratio tests. The strong classifier is applied to sub-regions of the image (at multiple scale) and outputs text candidate regions. In this application, there are typically between 2-5 false positives in images of 2,048 x 1,536 pixels. The second component is an extension and binarization [12] algorithm that acts on the text region candidates. The extension and binarization algorithm takes the text regions as inputs, extends these regions, so as to include text that the strong classifier did not detect, and binarizes them (ideally, so that the text is white and the background is black). The third component is an OCR software program which acts on the binarized regions (the OCR software gave far worse performance when applied directly to the image). The OCR software either determines that the regions are text, and reads them, or rejects the region as text.

The performance is as follows: (I) Speed. The current algorithm runs in under 9 seconds on images of size 2,048 by 1,536. The biggest delay is when we apply the strong classifier to the image in the detection stage. By using multiscale techniques, it should be possible to reduce the run time. (II) Quality of Results. We are able to detect text of *almost all form* with false negative rate of 2.8 %. We are able to read the detected text correctlty at 93.0 % (correctness is measured per complete word and not per letter). We incorrectly read non-text as text for 10 % of cases. But only 1 % remains incorrectly read after we prune out text which does not form coherent words. (Many of the remaining errors correspond to outputting ”111” due to vertical structures in the image.)

2 Previous Work

There has been recent succesful work on detecting text in images. Some has concentrated on detecting individual letters [1], [6],[7]. More relevant work is reported in [23], [10], , [22] [8], [9]. In particular, Lucas *et al* [10] report on performance analysis of text detection algorithms on a standardized database. It is hard to do a direct comparison to these papers. None of these methods use AdaBoost learning and the details of the algorithms evaluated by Lucas *et al* are not given. The performance we report in this paper is better than those reported in Lucas *et al*, but the datasets are different and more precise comparison on the same datasets are needed. We will be making our dataset available for testing.

3 The Datasets

We used two image datasets with one used for *training* the AdaBoost learning algorithm and the other used for *testing* it.

The training dataset was 162 images of which 41 of them were taken by scientists from XXXX (name witheld for confidentiality) and the rest taken by blind volunteers under the supervision of scientists from XXXX.

The test dataset of 117 images was taken entirely by blind volunteers. Briefly, the blind volunteers were equipped with a Nikon camera mounted on the shoulder or the stomach. They walked round the streets of YYYY taking photographs. Two observers from XXXX accompanied the volunteers to assure their safety but took no part in taking the photographs. The camera was set to the default automatic setting for focus and contrast gain control.

From the dataset construction, see figure (1), we noted that: (I) Blind volunteers could keep the camera approximately horizontal. (II) They could hold the camera fairly steady so there was very little blur. (III) The automatic contrast gain control of the cameras was almost always sufficient to allow the images to have good contrast.

Figure 1: Example images in the training dataset taken by blind volunteers (top two panels) and by scientists from XXXX (bottom two panels). The blind volunteers are, of course, poor at centering the signs and in keeping the camera horizontally alligned.

4. Selection of Features for AdaBoost

The AdaBoost algorithm is a method for combining a set of weak classifiers to make a strong classifier. The weak classifiers correspond to image features. Typically a large set of features are specified in advance and the algorithm selects which ones to use and how to combine them.

The problem is that the choice of feature set is critical to the success and transparency of the algorithm. The set of features used for face detection by Viola and Jones [19] consists of a subset of Haar basis functions. But there was no rationale for this choice of feature set apart from computational efficiency. Also there are important differences

between text and face stimuli because the spatial variation per pixel of text images is far greater than for faces. Facial features, such as eyes, are in approximately the same spatial position for any face and have similar appearance. But the positions of letters in text is varied and the shapes of letters differ. For example, PCA analysis of text, see figure (2), has far more non-zero eigenvalues than for faces (where Pentland reported that 15 eigenfaces capture over ninety percent of the variance [14]).

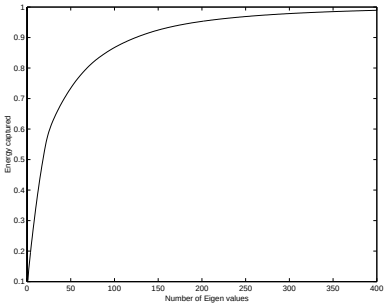

Figure 2: PCA on our dataset of text images (40 x 20 pixels). Observe that about 150 components are required to get 90 percent of the variance. Faces require only 15 components to achieve this variance [14].

Ideally, we should select *informative features* which give similar results on all text regions, and hence have low entropy, and which are also good for discriminating between text and non-text. Statistical analysis of the dataset of training text images shows that there are many statistical regularities.

For example, we allign samples from our text dataset (precise allignment is unnecessary) and analyze the response of the modulus of the x and y derivative filters at each pixel. The means of the derivatives have an obvious pattern, see figure (3), where the derivatives are small in the background regions above and below the text. The x derivatives tend to be large in the central (i.e. text) region while the y derivatives are large at the top and bottom of the text and small in the central region. But the variances of the x derivatives are very large within the central region (because letters have different shapes and positions). However, the y derivatives tend to have low variance, and hence low entropy.

Our first set of features are based on these observations. By averaging over regions we obtain features which have lower entropy. Based on the observation in figure (3), we designed block patterns inside the sub-window, corresponding to horizontal and vertical derivative. We also designed three symmetrical block patterns, see figure (4), which are chosen so that there is (usually) a text element within each sub-window. This gives features based on block based mean and STD of intensity and modulus of x and y derivative filters.

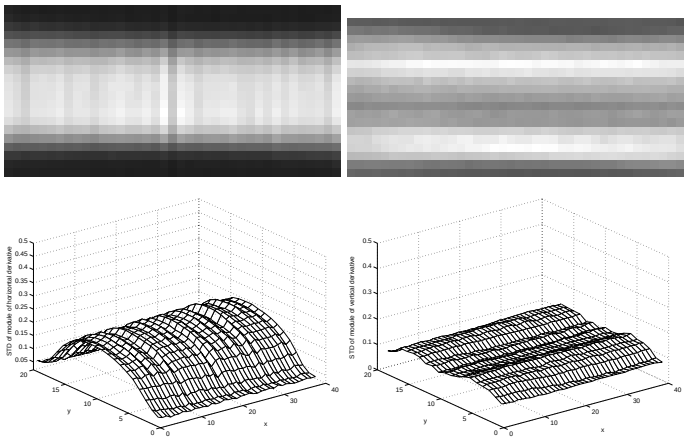

Figure 3: The means of the moduli of the x (left top) and y (right top) derivative filters have this pattern. Observe that the average is different directly above/below the text compared to the response on the text. The y derivative is small everywhere. The x derivatives tend to have large variance (bottom left) and the y derivatives have small variance (bottom right).

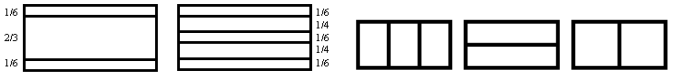

Figure 4: Block patterns. Features which compute properties averaged within these regions will typically have low entropy, because the fluctuations shown in the previous figure have been averaged out.

We build weak classifiers from these features by computing probability distributions. Formally, a good feature $f(I)$ will determine two probability distributions $P(\vec{f}(I)|\text{text})$ and $P(\vec{f}(I)|\text{non}-\text{text})$. We can obtain a weak classifier by using the log-likelihood ratio test. This is made easier if we can find tests for which $P(\vec{f}(I)|\text{text})$ is strongly peaked (i.e. has low-entropy because it gives similar results for every image of text) provided this peak occurs at a place where $P(\vec{f}(I)|\text{non}-\text{text})$ is small. Such tests are computationally cheap to implement because they only involved checking the value of $\vec{f}(I)$ within a small range.

We also have a second class of features which are more complicated. These include tests based on the histograms of the intensity, gradient direction, and intensity gradient. In ideal text images, we would be able to classify pixels as text or background directly from the intensity histogram which should have two peaks corresponding to text and background mean intensity. But, in practice, the histograms typically only have a single peak, see figure (5)

(top right). But by getting a joint histogram on the intensity and the intensity derivative, see figure (5) (bottom left), we are able to estimate the text and background mean intensities. These joint histograms are useful tests for distinguishing between text and non-text.

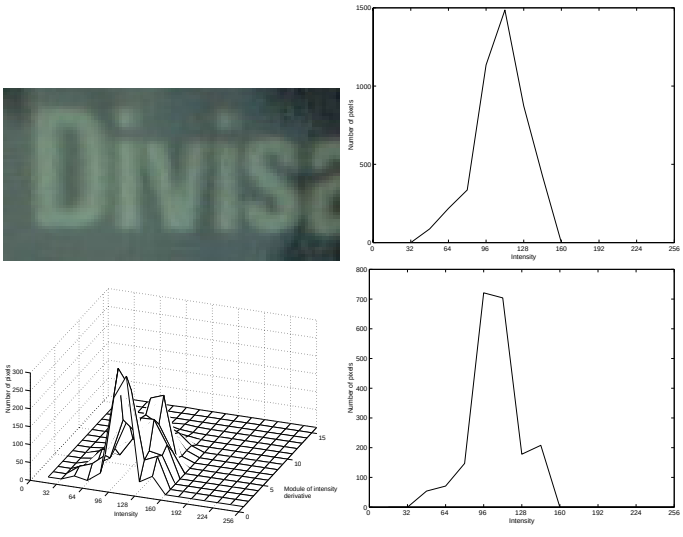

Figure 5: Original image (top left) has intensity histogram (top right) with only a single peak. But the joint histograms of intensity and intensity gradient shows two peaks (bottom left) and shown in profile (bottom right). The intensity histogram is contaminated by edge pixels which have high intensity gradient and intensity values which are intermediate between the background and foreground mean intensity. The intensity gradient information helps remove this contamination.

Our third, and final, class of features based on performing edge detection, by intensity gradient thresholding, followed by edge linking. These features are more computationally expensive than the previous tests, so we only use them later in the AdaBoost cascade, see next section. Such features count the number of extended edges in the image. These are also properties with low entropy, since there will typically be a fixed number of long edges whatever the letters in the text region.

In summary, we had: (i) 79 first class features including 4 intensity mean features, 12 intensity standard deviation features, 24 derivative features, (ii) 14 second class features (histograms), and (iii) 25 third class features (based on edge linking).

Ideally, we would learn a joint distributions $P(\vec{f}(I)|\text{text})$ and $P(\vec{f}(I)|\text{non}-\text{text})$ for all features f. In practice, this is impossible because of the dimensionally of the feature set and because we do not know which set of features should be chosen. We would need an immense amount of training data.

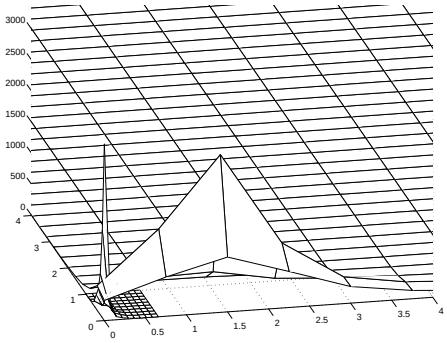

Figure 6: Joint histograms of the first features that AdaBoost selected.

Instead, we use both single features and joint distributions for pairs of features, followed by log-likelihood ratio tests, as our weak classifiers. See figure (6. These are then combined together by standard AdaBoost techniques. It is worth to note that all weak classifiers selected by AdaBoost are from joint distributions, indicating that it is more "discriminant" and making the learning process less greedy.

The result of this feature selection approach is that our final strong classifier, see next section, uses far fewer filter's than Viola and Jones' face detection classifier [19]. This helps the transparency of the system.

5. AdaBoost

The AdaBoost algorithm [4] has been shown to be arguably the most effective method for detecting target objects in images [19]. Its performance on detecting faces [19] compares favourably with other successful algorithms for detecting faces [3, 15, 17, 21, 24] and for detecting text [8],[16],[9],[1].

The standard AdaBoost algorithm learns a "strong classifier" $H_{\text{Ada}}(\mathbf{I})$ by combining a set of T "weak classifiers" $\{h_t(\mathbf{I})\}$ using a set of weights $\{\alpha_t\}$:

$$H_{\text{Ada}}(\mathbf{I}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{I})).$$

The selection of features and weights are learned through supervised training off-line [4]. Formally, AdaBoost uses a set of input data $\{\mathbf{I}_i, y_i : i = 1, .., N\}$ where $\mathbf{I}_i$ is the input, in this case image windows described below, and y_i is the classification where $y_i = 1$ indicates text, $y_i = -1$ is not-text. The algorithm uses a set of *weak classifiers* denoted by $\{h_\mu(.)\}$. These weak classifiers correspond to a decision of text or non-text based on simple tests of visual cues (see next paragraph). These weak classifiers are only required to make the correct classifications slightly over

half the time. The AdaBoost algorithm proceeds by defining a set of weights $D_t(i)$ on the samples. At $t = 1$, the samples are equally weighted so $D_1(i) = 1/N$. The update rule consists of three stages. Firstly, update the weights by $D_{t+1}(i) = D_t(i)e^{-y_i\alpha_t h_t(\mathbf{I}_i)}/Z_t[\alpha_t, h_t]$, where Z_t is a normalization factor chosen so that $\sum_{i=1}^{N} D_{t+1}(i) = 1$. The algorithm selects the $\alpha_t, h_t(.)$ that minimize $Z_t[\alpha_t, h_t(.)]$. Then the process repeats and outputs a *strong classifier* $H_t(\mathbf{I}) = \text{sign}(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{I}))$. It can be shown that this classifier will converge to the optimal classifier as the number of classifiers increases [4].

AdaBoost requires a set of classified data with image windows labelled manually as being text or non-text. We performed this labelling for the training dataset and and divided each text window into several overlapping text segments with fixed width-to-height ratio 2:1. This lead to a total of 7,132 text segments which were used as positive examples, see figure (7). The negative examples were obtained by a bootstrap process similar to Drucker et al [2]. First we selected negative examples by randomly sampling from windows in the image dataset. After training with these samples, we applied the AdaBoost algorithm to classify all windows in the training images (at a range of sizes). Those misclassified as text were then used as negative examples for retraining AdaBoost. The image regions most easily confused with text were vegetation, repetitive structures such as railings or building facades, and some chance patterns.

The previous section described the weak classifiers we used for training AdaBoost.

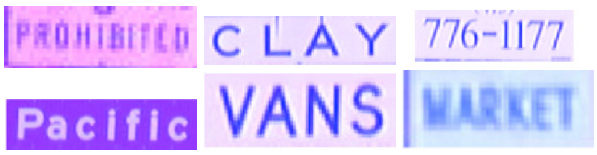

Figure 7: Positive examples used for training AdaBoost. Observer the low quality of some of the examples.

We used standard AdaBoost training methods to learn the strong classifier [4] [5] combined with Viola and Jones' cascade approach which uses asymmetric weighting [19]. The cascade approach enables the algorithm to rule out most of the image as text locations with a few tests (so we do not have to apply all the tests everywhere in the image). This makes the algorithm extremely fast when applied to the test dataset and yields order of magnitude speed-up over standard AdaBoost [19]. Our algorithm had a total of 4 cascade layers. Each layer has 2, 10, 30, 50 tests respectively. The overall algorithm uses 92 different feature tests. The first three layers of the cascade only use mean, STD and module of derivative features, since they can be easily calculated from integral images[19]. Computation intensive features, histogram and edge linking, involve all pixels inside the sub-window. So we only let them be selected in the last layer.

In the test stage, we applied the AdaBoost strong classifier $H(\mathbf{I})$ to windows of the input images at a range of scales. There was a total of 14 different window sizes, ranging from 20 by 10 to 212 by 106, with a scaling factor of 1.2. Each window was classified by the algorithm as text or non-text. There was often overlap between windows classified as text. We merged these regions by taking the union of the text windows. (Much of the run time of our algorithm is due to applying the strong classifier to the image windows at these different scale. A multiscale approach should speed this up significantly).

In our test stage, AdaBoost gave very high performance with low false positives and false negatives (in agreement with previous work on faces [19]). When applied to over 20,000,000 image windows, taken from 35 images, the total number of false positives was just over 118 and the number of false negatives was 27. By alterering the threshold we could reduce the number of false negatives to 5 but at the price of raising the number of false positives, see table (1). We decided to keep not to alter the threshold so as to keep the number of false positives down to an average of 4 per image (almost all of which will be eliminated at the reading stage).

Object	Thresh	False Pos.	False Neg.	Images	Subwindows
Text	0.00	118	27	35	20,183,316
Text	-0.05	1879	5	35	20,183,316

Table 1: Performance of AdaBoost at different thresholds. Observe the excellent overall performance and the trade-off between false positives and false negatives.

We illustrate these results by showing the windows that AdaBoost classifies as text for typical images in the test dataset, see figure (8).

6 Extension and Binarization

Our next stage produces binarized text regions to be used as inputs to the OCR reading stage. (It is possible to run OCR directly on intensity images but we obtain substancially worse performance if we do so). In addition to binarization, we must extend the text regions found by the AdaBoost strong classifiers because these regions sometimes miss letters or digits at the start and end of the text.

We start by appling adaptive binarization [12] to the text regions detected by the AdaBoost strong classifier. This is followed by a connected component algorithm [13] which detects letter and digit candidates and enables us to estimate

Figure 8: Results of AdaBoost on typical test images (taken by blind subjects). The boxes display areas that AdaBoost classifies as text. Observe that text is detected at a range of scales and at non-horizontal orientations. Note the small number of false positives. The boxes will be expanded and binarized in the next processing stage.

their size and the spacing between them. These estimates are used to *extend* the search for text into regions directly to the left, right, above and below of the regions detected by AdaBoost. Binarization is then applied in these extended text regions.

More precisely, we use Niblack's adaptive binarization algorithm [12] which was reported by Wolf [22] to be the most succesful binarization algorithm (jointly with Yanowitz-Bruckstein's method [25]). This requires selecting both a threshold for the local sub-windows of the text region and an appropriate size for the sub-window. The sub-windows have vertical and horizontal lengths in the range 3-11 pixels. At each point in the text region, the sub-window is chosen to be the smallest possible subwindows whose variance is above a fixed threshold. Then the binarization is performed by selecting a threshold $T(x, y)$ based on the mean $\mu(x, y)$ and standard deviation $s(x, y)$ within the sub-window:

$$T(x, y) = ks(x, y) + m(x, y)$$

The value k is used to adjust how much the foreground object edges that are taken as a part of the objects. (For example: $k = \pm 0.2$ to check for cases where the foreground is brighter or darker than the text.)

We show results for the extension and binarization algorithms in figure (9) using the text regions shown in figure (8).

7 Text Reading

We applied commercial OCR software to the extended text regions (produced by AdaBoost followed by extension and binarization). This was used both to read the text and to discard false positive text regions.

Overall, the AdaBoost strong classifier (plus extension/binarization) detected 97.2 % of the visible text in our test dataset (text that could be detected by a normally sighted viewer). See figure (10) for typical examples of the text that AdaBoost fails to detect. Most of these errors correspond to text which is blurred or badly shadowed. Others occur because we do not train AdaBoost to detection vertical text or individual letters. (Our training examples were horizontal segments usually containing two or three letters/digits).

For the 286 extended text regions correctly detected by the AdaBoost strong classifier (plus extension/binarization), we obtained a correct reading rate of 93.0 % (proportion of words correctly read). This required a preprocessing stage to scale the text region. The 7 % errors are caused by small text areas. See figure (11) for examples of text that we can read successfully and figure (12) for text that we cannot read.

Figure 9: Extension (left column) and binarization of the text regions shown in figure (8). Note that our OCR software is not yet able to read Chinese or Spanish so we treat these as non-text.

Figure 10: Examples of different text that we fail to detect by the AdaBoost strong classifier. Some are blurred, badly shaded, or have highly non-standard font. Others are not detected because we did not train AdaBoost to detect individual letters/digits or vertical text.

Figure 11: Examples of different text that can be correctly detected and read. First row: Road signs and street numbers. Second and third rows: Commercial and Informational signs. Fourth row: bus signs and bus stops. Fifth row: House numbers and Phone Numbers.

Figure 12: Examples of text that we can detect by the AdaBoost strong classifier but cannot read correctly. These correspond to small text and blurred text. But improvements in our binarization process might make some of them readable.

The OCR algorithm will also sometimes misclassify the false positive text regions found by AdaBoost and classify them as text. See the example in figure (13). This occurred for about 10 % of false positive text regions, but often the text read made no grammatical sense and can be removed (though this requires an additional stage after the OCR software). The most common remaining error are text string like "111" or "Ill" which correspond to vertical edges in the image caused, for example, by iron railings.

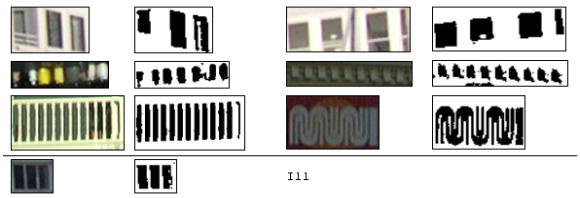

Figure 13: Examples of OCR output on non-text. Only the bottom window is incorrectly read as text, "Ill",

The results presented here use the ABBYY Fine Reader software. Other OCR software we tested that gives almost identical performance includes TOcr and Readiris Pro 8.

8 Summary

This paper used the AdaBoost algorithm to learn a strong classifier for detecting text in unconstrained city scenes. The key element was the choice of the feature set which

was selected to have features with low entropy of the positive training examples (so that they gave similar responses for any text input). In addition, we used log-likelihood ratio tests on the joint probability distributions of pairs of features. The resulting system is small, compared to that used by Viola and Jones for face detection [19], and required only 92 filters and 4 layers of cascade.

The resulting strong classifier was very effective on our dataset taken by blind users. This database, with ground truth, will be made available to other researchers. The detection rates resulted in only a small number (2-5) false positive rates in images of size 2,048 x 1,536.

To demonstrate the effectiveness of our approach, we used it as a front end to a system which included an extension and binarization algorithm followed by a commercial OCR system. The resulting performance was highly effective.

The algorithm currently runs at 9 seconds on a 2,048 x 1,536 image (which compares favourably to speeds for AdaBoost classifiers for faces when the size of the image is taken into account). We anticipate that multi-scale processing will enable us to significantly reduce the algorithm speed.

Our future work involves developing alternative text reading software. Although current OCR algorithms are, as we have shown, very effective they remain a black box and we cannot modify them or improve them. Instead we will continue developing our reading algorithms based on deformable templates [6],[1]. These algorithms have the additional advantage that they use generative models [18] and can be applied directly to the image intensity without requiring binarization.

References

[1] S. Belongie, J. Malik, and J. Puzicha. "Matching shapes", In *Proc. of the IEEE Intl. Conf. on Computer Vision*, 2001.

[2] H. Drucker, R. Schapire, and P. Simard. "Boosting Performance in Neural Networks". *International Journal of Pattern Recognition and Artificial Intelligence*. Vol. 7, no. 4, pp 705-719. 1993.

[3] F. Fleuret, and D. Geman. "Coarse-to-Fine Face Detection." *I nternational Journal of Computer Vision*. 2000.

[4] Y. Freund and R. Schapire, "Experiments with a new boosting algorithm", Proc. of the Thirteeth Int. Conf. on Machine Learning, 148–156 (1996).

[5] J. Friedman, T. Hastie and R. Tibshirani. "Additive logistic regression: a statistical view of boosting", Dept. of Statistics, Stanford University Technical Report. 1998.

[6] S. Gold, A. Rangarajan, C.-P. Lu, S. Pappu, and E. Mjolsness. New algorithms for 2D and 3D point matching: pose estimation and correspondence. Pattern Recognition, 31(8), 1998.

[7] M. Revow, G.K.I. Williams and G.E. Hinton, "Using generative models for handwritten digit recognition", *IEEE Trans . PAMI*, 18, pp. 592–606, 1996.

[8] A.K. Jain and B. Tu. "Automatic Text Localization in Images and Video Frames". *Pattern Recognition*. 31(12), pp 2055-2076. 1998.

[9] Huiping Li, David Doermann and Omid Kia. "Automatic Text Detection and Tracking in Digital Video".*IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[10] S. M. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong and R. Young. "ICDAR 2003 Robust Reading Competitions", In *7th International Conference on Document Analysis and Recognition - ICDAR2003*, 2003.

[11] J. Malik, S. Belongie, T. Leung and J. Shi, "Contour and Texture Analysis for Image Segmentation." IJCV 43 (1):7-27, June 2001.

[12] W. Niblack. **An Introduction to Digital Image Processing.** pp. 115-116, Prentice Hall, 1986.

[13] T. Pavlidis. **Structural pattern Recognition.** Springer-Verlag, Berlin-Heidlesberg-New York. 1977.

[14] M. Turk and A. Pentland. "Eigenfaces for recognition". *Journal of Cognitive Neuro Science*, vol. 3, pp. 71-86, 1991.

[15] H. Rowley, S. Baluja, and T. Kanade. "Neural network-based face detection". In *IEEE Patt. Anal. Mach. Intell.*. Vol. 20, pp 22-38. 1998.

[16] T. Sato, T. Kanade, E. Hughes, and M. Smith. "Video OCR for Digital News Archives". IEEE International Workshop on Content Based Access of Image and Video Databases. 1998.

[17] H. Schniederman and T. Kanade. "A Statistical method for 3D object detection applied to faces and cars". In *Computer Vision and Pattern Recognition*. 2000.

[18] Z.W. Tu and S.C. Zhu, "Image segmentation by Data Driven Markov chain Monte Carlo", *IEEE Trans. PAMI*, vol 24, no 5. May, 2002.

[19] P. Viola and M. Jones. "Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade". In *Proceedings NIPS01*. 2001.

[20] P. Viola, M. Jones and D. Snow. "Detecting Pedestrians using Patterns of Motion and Appearance". In *International Conference on Computer Vision*. 2003.

[21] M. Weber, W. Einhuser, M. Welling, P. Perona "Viewpoint-Invariant Learning and Detection of Human Heads". In *Proc. 4th IEEE Int. Conf. Automatic Face and gesture Recognition.*. 2000.

[22] C. Wolf and J-M. Jolion, "Extraction and Recognition of Artificial Text in Multimedia Documents", http://rfv.insa-lyon.fr/ wolf/papers/tr-rfv-2002-01.pdf.

[23] V. Wu, R. Manmatha, and E. M. Riseman. ”Finding text in images”, In *Proc. of the 2nd ACM Conf. on References Digital Libraries*, pages 3C12, 1997.

[24] Ming-Hsuan Yang, N. Ahuja, D. Kriegman. ”Face Detection Using Mixtures of Linear Subspaces”. In *Proc. 4th IEEE Int. Conf. Automatic Face and gesture Recognition.*. 2000.

[25] S. D: Yanowitz and A. M. Bruckstein.” A New Method for Image Segmentation”. *Computer Vision, Graphics, and Image Processing* CVGIP, Vol.46, no. 1, pp. 82-95, 1989.