An AdaBoost Approach to Detecting and Extracting Texts from Natural Scene Images

Chin-Shyurng Fahn and Chia-Wei Liu

Department of Computer Science and Information Engineering
National Taiwan University of Science and Technology, Taipei 10607, Taiwan
E-mail: csfahn@csie.ntust.edu.tw

Abstract- In this paper, we present a new connected-component-based text detection and extraction method for natural scene images using AdaBoost techniques. First, we utilize the Canny operator and devise a two-phase two-scan labeling connected components algorithm to precisely find out candidate character blocks. Subsequently, several fundamental filtering rules are derived from the characteristics of texts to screen non-text blocks. Reducing the number of candidate character blocks can speed up the efficiency of the text classifier and improve the accuracy. Then we distinguish text blocks from the remaining candidate blocks using the strong classifier that is trained by an AdaBoost algorithm. In the sequel, we group the detected characters into words. Compared with other machine learning algorithms, the algorithm has an advantage of facilitating the speed of convergence during the training. Thus, we can update training samples to deal with comprehensive circumstances but do not spend much computational cost. Finally, we adopt a binarization method with an adaptive threshold to extract text regions. Even in an unbalanced illuminant environment, we can still extract texts successfully. Experimental results reveal that the text recall and precision rates are both more than 95% and the system efficiency of execution is also satisfactory.

Keywords: AdaBoost algorithm, connected component labeling, text detection, text extraction, natural scene image.

1. Introduction

In recent years, it is easy to take a picture anywhere and anytime owing to the popularization of digital cameras and cellular phones. Therefore, the image containing characters is not restricted to being obtained through a scanner. The existing text detection methods can be grouped into two classes typically. The first one is *connected-component-based* (CC-based) methods and the second one is *texture-based* methods. The CC-based methods can extract texts efficiently, but they will encounter difficulties when texts touch with themselves or other graphical objects.

Characters in videos, documents, and images possess quite rich information. The text information extraction (TIE) techniques are widely applied to the license plate detection, business card analysis, and video subtitle location, and so forth. Characters in an image will have the following general characteristics [1]:

1) Geometry: The arrangements of text lines are vertical and horizontal in the majority. Characters in the same text line are usually aggregated with a uniform clearance and their sizes are regular.

2) Color: Characters in the same text line often have the same or similar color. The CC-based methods are also set up on this assumption.

3) Edge: Characters within a title in most of natural scenes commonly have a strong contrast to the background in order to read conveniently.

The bulk of the existing text detection methods are implemented according to the character characteristics mentioned above. But such characteristics will be varied by the influence of environments or people. Hence, we want to develop an adaptive text detection and extraction method, which can keep certain accuracy along with the change of scenes. Fig. 1 shows the system flow chart of our proposed text detection and extraction method.

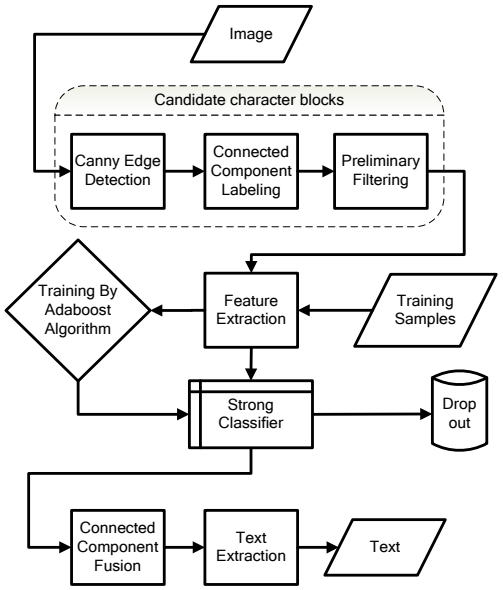

Fig. 1 The system flow chart.

2. Text Detection

In this section, we first indicate how to obtain candidate character blocks by Canny edge detection and connected component (CC) labeling. Then we will conduct a preliminary filter to further screen the candidate character blocks.

2.1. Canny edge detection

In order to label CCs more precisely, it essentially requires a good edge detection result. Too many useless edge pixels will make the CC-based methods be confused. The Canny edge detection method is known as the optimal edge detector. Through a careful evaluation, we adopt the edge detection algorithm proposed by Canny [2].

2.2. Connected component labeling

To obtain the CC properties of an image, such as the number of CCs and the size of each CC, we develop a simple and fast CC labeling algorithm. [3]. This algorithm provides much less execution time than the conventional labeling algorithms do [4]-[6], particularly than the recursive methods.

2.3. Character block filtering

The outcome of the CC labeling is the primary text detection result. In order to decrease the computational time of the text classifier later and improve the recognition rate, we have to screen some candidate blocks which do not contain characters obviously. The following depicts the CC properties and filtering rules where B_W represents the width of a CC block, B_H represents its height, H indicates the height of the original image, W indicates its width, and N_E symbolizes the number of edge pixels in a CC. The candidate character block satisfying with any condition of the filtering rules listed in Table 1 will be discarded, where S_i means the scaling factor of a CC block.

Considering feature extraction later, the block size used for executing the wavelet transform is 16 by 16. So in the first rule, we remove the block whose height is smaller than 16. In the second and third rules, we want to filter out too large blocks and too thin blocks. In the fourth rule, we take account of whether the number of edge pixels is enough to form a closed area. Because the characteristic of strokes, the edge pixels in a character block will constitute a closed outline. Therefore, while the condition of $B_W / B_H > 1.5$ meets, it shows that the associated block contains more than one letter at least. If the number of edge pixels is smaller than the circumference of the block, it shows that these edge pixels are certainly unable to form the closed area. While the condition of $B_H / B_W > 1$ satisfies, it may be the letter L or I, or the digit 1, and so forth. In our experiments, the preliminary filter can take away about 50 percents of blocks for most of natural scene images and only costs a little computational time. It contributes to the reduction of computational time for candidate blocks classification afterwards.

If the width of a candidate character block is greater than or equal to its height, we split it into several blocks in the horizontal direction to handle them separately. Fig. 2(a) graphically shows the above manipulation. Conversely, on the condition of $B_H > B_W$, we extend the block into a square as Fig. 2(b) illustrates.

Table 1 The Conditions of Filtering Rules

Condition	Meaning
$B_H < 16$	The block height is too small.
$B_W / B_H > 1.5$ and $N_E < 2(B_H + B_W)$, or $B_H / B_W > 1$ and $N_E < 1.5(B_H + B_W)$	The number of edge pixels is too few.
$B_W > W/2$ or $B_H > H/2$	The block size is too large.
$B_H \times B_W > \frac{10}{n} \times \sum_{i=1}^{n} S_i B_H \times S_i B_W$, or $B_H \times B_W < \frac{1}{10n} \times \sum_{i=1}^{n} S_i B_H \times S_i B_W$	The block size related to all valid blocks is too large or too small.

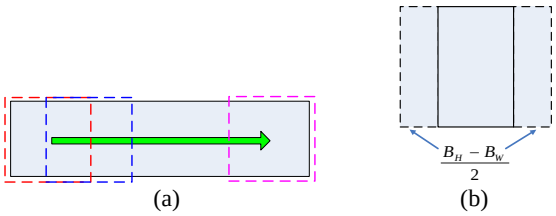

Fig. 2 Illustration of manipulating a candidate character block; (a) partition of the block if $B_W \geq B_H$; (b) extension of the block if $B_H > B_W$.

3. Text Verification and Extraction

After candidate character blocks have been screened, we will first proceed with feature extraction based on texture analysis. The text verification of the candidate blocks are then conducted by an AdaBoost approach. Finally, CC fusion is carried out to further achieve text extraction from natural scene images.

3.1. Feature extraction

We adopt the discrete wavelet transform (DWT) to extract the features of characters for texture analysis and description. Since its advantage is to provide not only the relation between time and frequency domains but also the relation between time and intensity domains. According to this, we perform the 2D-Haar (1-Level) DWT on the candidate blocks and subsequently extract features from four Haar wavelet sub-bands. First, we calculate the histogram of wavelet coefficients, where z_k means the intensity of a wavelet coefficient and $p(z_k)$ stands for the probability density function of z_k. In the following expression, M and N are the height and width of a wavelet sub-band, respectively, and $w(i, j)$ is the wavelet coefficient. Herein, we take a wavelet sub-band as an example depicted below:

The first kind of features is the mean of wavelet coefficients:

$$Mean = \frac{1}{M \times N} \times \sum_{i=1}^{M} \sum_{j=1}^{N} w(i, j) \quad (1)$$

The second kind of features is the contrast of wavelet coefficients:

$$Contrast = 1 - \left[1/(1+\sigma^2) \right] \quad (2)$$

where $\sigma^2 = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} [w(i, j) - \mu]^2$

The third kind of features is the entropy of wavelet coefficients, which is based on the information theory:

$$Entropy = -\sum_{i=1}^{M} \sum_{j=1}^{N} w(i, j) \log w(i, j) \quad (3)$$

The fourth kind of features is the energy of wavelet coefficients:

$$Energy = \sum_{i=1}^{M} \sum_{j=1}^{N} w(i, j)^2 \quad (4)$$

And the fifth kind of features is the third moment of the intensities of wavelet coefficients, which is used for a measurement of histogram skewness:

$$The\ third\ moment = \sum_{k=0}^{255} (z_k - \mu)^3 p(z_k) \quad (5)$$

Besides the above five kinds of features derived from the wavelet coefficients directly, some other features that express the information of two relative coefficients for all wavelet sub-bands are employed. Considering the strokes of a character, we can extract the features in form of a co-occurrence matrix whose element can be represented as $C(m, n \mid d, \theta)$, where m and n indicate two wavelet coefficients, d stands for the distance, and θ means the angle between m and n, respectively. In the experiments, we find four co-occurrence matrices with $\theta = 0°, 45°, 90°,$ and $135°$ and d = 1 for each wavelet sub-band. Furthermore, we extract three kinds of features from a co-occurrence matrix, which are the correlation, inverse difference moment, and inertia stated as follows:

$$Correlation = \frac{\sum_m \sum_n (m - \mu_m)(n - \mu_n) C(m, n \mid d, \theta)}{\sigma_x \sigma_y \sum_m \sum_n C(m, n \mid d, \theta)} \quad (6)$$

where $\mu_m = \frac{1}{N} \sum_{j=1}^{N} w(m, j)$, $\mu_n = \frac{1}{M} \sum_{i=1}^{M} w(i, n)$,

$$\sigma_x = \sqrt{\frac{1}{M} \sum_m \sigma_m^2} \ , \quad \sigma_y = \sqrt{\frac{1}{N} \sum_n \sigma_n^2}$$

$\sigma_m^2 = \frac{1}{N} \sum_{j=1}^{N} [w(m, j) - \mu_m]^2$, and $\sigma_n^2 = \frac{1}{M} \sum_{i=1}^{M} [w(i, n) - \mu_n]^2$

$$Inverse\ difference\ moment = \frac{\sum_m \sum_n \frac{C(m, n \mid d, \theta)}{1 + (m-n)^2}}{\sum_m \sum_n C(m, n \mid d, \theta)} \quad (7)$$

$$and\ Inertia = \frac{\sum_m \sum_n (m-n)^2 C(m, n \mid d, \theta)}{\sum_m \sum_n C(m, n \mid d, \theta)} \quad (8)$$

Hence, each wavelet sub-band will have twelve features extracted from the four co-occurrence matrices. Adding the five kinds of features mentioned above, there are seventeen features for a sub-band. Finally, we have a sixty eight dimensional feature vector to describe the texture of a character block. Such a feature vector will be applied to train the AdaBoost strong classifier that is used for verifying the candidate character blocks later.

3.2. Classification strategies

Boosting is a general method for enhancing the performance and extending the capabilities of a learning scheme.

3.2.1. The AdaBoost algorithm.

The AdaBoost (adaptive boosting) algorithm is a kind of boosting algorithms which were proposed by Freund and Schapire [7]. Fig. 3 is a generalized version of the AdaBoost algorithm for binary classification problems.

Given: $S = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$ with $x_i \in \mathcal{X}$ and $y_i \in \{-1, +1\}$.

Initialize the distribution: $D_1^{(i)} = 1/l$, $i = 1, 2, ..., l$.

For t = 1,2,…,T:

Train the weak learner using the distribution $D_t^{(i)}$, $i = 1, 2, ..., l$.

Get the weak hypothesis $g_t : \mathcal{X} \to R$.

Update:

$D_{t+1}^{(i)} = D_t^{(i)} \exp(-\alpha_t y_i g_t(x_i)) / Z_t$, $i = 1, 2, ..., l$,

where Z_t is a normalization factor ($D_{t+1}^{(i)}$ is still a distribution)

and $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$ with $\varepsilon_t = \sum_{i=1}^{l} D_t^{(i)} [y_i \neq g_t(x_i)]$.

Output the final hypothesis: $G(\mathcal{X}) = sign\left(\sum_{t=1}^{T} \alpha_t G_t(\mathcal{X}) \right)$.

Fig. 3 A generalized version of the AdaBoost algorithm.

In such an AdaBoost algorithm, we select a weak classifier g_t which has the lowest classification error with respect to the distribution D_t at each of iterations. And then update the distribution of training samples and the weight of the weak classifier according to the *weighted training error rate* ε_t and *weighting factor* α_t until reaching the number of specified iterations. However, the prerequisite is $\varepsilon_t < 1/2$; otherwise, the iteration will terminate. All the Adaboost-based techniques can be regarded as a greedy optimization method for minimizing the exponential error function as follows [8].

$$E_t = \sum_{i=1}^{l} \exp(-y_i g_t(x_i)) \quad (9)$$

The upper bound of the training error for the strong classifier is $\prod_t Z_t$, where Z_t is expressed below.

$$Z_t = \sum_{i=1}^{l} D_t^{(i)} \exp(-y_i g_t(x_i)) \quad (10)$$

As a result, minimizing $\prod_t Z_t$ is almost identical to minimizing the overall classification error, because training samples usually account for a significant part of the whole samples. The purpose of the AdaBoost algorithm is to improve the existing weak learning algorithms L. The boosting algorithm looks for several weak classifiers g_t through conducting L repeatedly, and combines these weak classifiers to be a strong classifier G in accordance with certain rules. Consequently, the performance of G is much better than that of any weak classifier in the classification function space.

3.2.2. Classifier realization.

The weak classifier is the core of an AdaBoost algorithm. Each weak classifier produces the answer "yes" or "no" for a particular feature. There are various algorithms for predicting continuous variables or categorical variables from a group of continuous predictions/categorical factor effects. The classification and regression tree (CART) algorithm was proposed by Breiman et al. [9]. Here, we adopt the CART as the structure of the weak classifier trained with the AdaBoost algorithm.

We utilize the following two rules for the construction of a CART node. Given a training sample set $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_l, y_l)\}$, where each x_i belongs to an instance space $\mathcal{X} \in R^n$ (each vector with dimensionality n; $x_i = (x_{i_1}, x_{i_2}, ..., x_{i_n})$) and each label y_i belongs to a finite label space $Y \in \{-1, +1\}$.

Rule 1. For each feature (all dimensions), find out a threshold which separates the sample set *S* with a minimal error.

Rule 2. Select the *j*-th feature with the minimal error and build a CART node.

(a) Set up the branch condition: $\xi_j > threshold_j$.

(b) Arrange the branches that are connected with leaves to perform respective classification.

3.3. Connected component fusion

Before text extraction, we have to group and extend the character blocks verified by the AdaBoost strong classifier into complete word regions. Because valid character regions sometimes lose letters resulting from the verification, it is necessary to remedy the misclassification blocks by clustering character regions into word ones. What follows is the remedial method called CC fusion process.

Starting from an arbitrary connected component CC_A, the search region is extended to twice the height of the CC in the horizontal direction (both left and right). If there is a connected component CC_B satisfying with either of the following conditions, the two CCs are regarded as the same cluster.

1) The top of CC_B is between the tops of $CC_A \pm \left(B_{H_A}/7\right)$, and the bottom of CC_B is between the bottom of $CC_A - B_{H_A}$ and the bottom of $CC_A + \left(B_{H_A}/2\right)$.

2) The bottom of CC_B is between the bottoms of $CC_A \pm \left(B_{H_A}/7\right)$, and the top of CC_B is between the top of $CC_A - \left(B_{H_A}/2\right)$ and the top of $CC_A + B_{H_A}$.

The CC fusion process stops until each valid CC has been clustered. We rescue the misclassified CC of a character in the search region of each cluster and set it to be valid. Finally, the new boundary of each cluster is found and the cluster which consists of only one CC is discarded because of characters in an image often in form of text lines. However, if the distribution of characters in an image is sparse or too many misclassification blocks are closely located to each other, the CC fusion process will obtain a poor result.

Although it is possible to directly extract texts from gray images, we acquire a worse performance substantially than from binary images. Therefore, we apply an improved version of Niblack's adaptive binarization method which was proposed by Sauvola et al. [10] to extract word regions. Sauvola's method is required to suitably determine a dynamic threshold T_r for each pixel from intensity statistics within a local mask of size *r* as Eq. (11) states.

$$T_r(x, y) = \mu_r(x, y) \times [1 + k(1 - \sigma_r(x, y)/R)] \quad (11)$$

where $\mu_r(x, y)$ and $\sigma_r(x, y)$ stand for the mean and standard deviation (STD) of the pixel intensities within the local mask, respectively. Additionally, *R* is the possible range of STD, whose purpose is normalization, and *k* is an empirical constant.

4. Experimental Results and Discussion

The developing experimental environment includes Borland C++ Builder 6, MATLAB 7.2, and Microsoft Windows XP; the personal computer is equipped with Pentium IV 3.2 GHz CPU and 1 GB RAM.

4.1. Training samples

Our training samples are gathered from the competition dataset provided by the International Conference on Document Analysis and Recognition in 2006. The competition dataset mainly comprises the covers of books, natural scenes, various kinds of articles, signboards, and so on. Besides, there are some samples collected from Internet. We use the GML AdaBoost Matlab Toolbox which is provided by the Graphics and Media Laboratory of the Computer Science Department at Moscow State University [11] and Matlab 7.2 for training/constructing our strong classifier. The positive samples and negative samples are marked manually from CC labeling images by users. We have 3,000 training samples, including equal number of text samples and non-text samples. Fig. 4 illustrates some of our training samples.

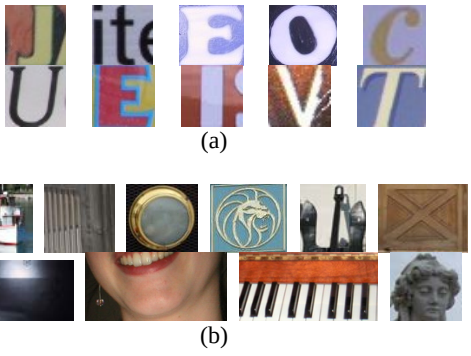

Fig. 4 Part of training samples: (a) positive ones; (b) negative ones.

4.2. Error estimation of different models

In the field of statistics, cross-validation is employed to estimate a generalization error based on *re-sampling*. The generalization errors of various models serve as

choosing the best one among them. Cross-validation is to partition a set of samples into many subsets such that performance analysis is initially executed on a single subset of samples. The remaining subsets of samples used to validate the hypothesis are called validation (or test) data. There are several common types of cross-validation; for example, holdout cross-validation, *K*-fold cross-validation, and leave-one-out cross-validation. In the experiments, we utilize 5-fold cross-validation to estimate the accuracy of different system models. Attempting to find out the optimal numbers of splits and training iterations is to prevent from the occurrence of over-fitting.

Figs. 5 and 6 show the error rates resulting from varied training models with different numbers of CART splits, feature types, and versions of AdaBoost algorithms. From the model estimation, we can see that the best performance is achieved while the Gentle AdaBoost algorithm uses wavelet features and 16 CART splits. As for the number of CART splits greater than 16, it does not attain an obvious improvement in the classification accuracy. Moreover, it becomes very time-consuming while training and verification. Thus, we adopt the Gentle AdaBoost algorithm employing wavelet features and 16 CART splits as our classification model for a tradeoff between accuracy and efficiency. The training time on the condition of 3,000 samples, 16 CART splits, 68 wavelet features, and 200 iterations of the Gentle AdaBoost algorithm takes 5 minutes approximately.

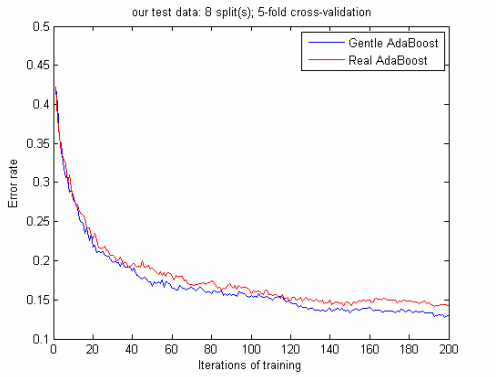

Fig. 5 The error rate resulting from the training model with gray level features for 8 splits and 5-fold cross-validation.

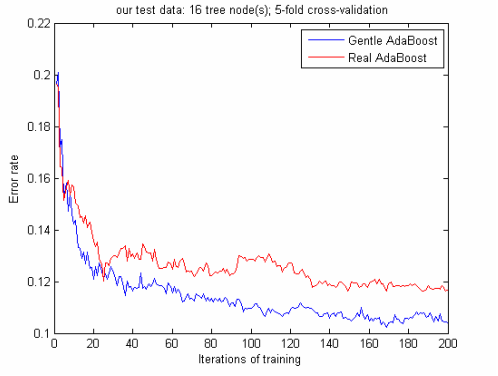

Fig. 6 The error rate resulting from the training model with wavelet features for 16 splits and 5-fold cross-validation.

4.3. System evaluation and data analysis

Before the exhibition of experimental results formally, we first explain our notations and definitions listed in Table 2 which will be used to measure our system performance subsequently.

Table 2 Notations and Definitions Used for the Measurement of System Performance

Notation	Definition
Precision rate	$\frac{True\ positive}{True\ positive\ +\ False\ positive}$
Recall rate	$\frac{True\ positive}{True\ positive\ +\ False\ negative}$
True positive	*Result* ∩ *Ground Truth*
False positive	*Result* ∩ $\overline{Ground\ Truth}$
False negative	$\overline{Result}$ ∩ *Ground Truth*

We apply both precision and recall rates to measure the performance of our system; generally speaking, the precision rate is inversely related to the recall rate, where the true positive is the number of texts that are correctly classified as texts, the false positive is the number of non-texts that are incorrectly classified as texts, and the false negative is the number of texts that are incorrectly classified as non-texts.

Our testing image set has 100 pictures altogether, each of which is retrieved from Internet or taken by ourselves. The contents of the image set are signboards mainly. The following demonstrates the classification accuracy of the AdaBoost strong classifier and analyzes the experimental results according to the above definitions. The basic unit is a block of 16×16 pixels, from which 256 gray level features and 68 wavelet features are extracted, respectively. The total number of CCs is 36,612, including 1,938 text blocks and 34,674 non-text ones. The accuracy of classification is recorded in Table 3.

Table 3 The Overall Performance of Our System

Set	Feature	Precision rate	Recall rate
Training	wavelet	98.25%	99.14%
	gray level	96.89%	97.73%
Testing	wavelet	95.27%	95.95%
	gray level	90.56%	92.54%

Note that the meanings of the precision and recall rates for the training and testing sets are slightly different. The cardinality of the training set is the number of training samples. The precision and recall rates of the training set mean the classification accuracy of the AdaBoost strong classifier when the validation is in progress directly. And the cardinality of the testing set is the number of CCs totally. We compute the numbers of true positives, false positives, and false negatives from the final classification results of the AdaBoost strong classifier. Table 4 shows the execution time for each step on an average. It is easily seen that the step of edge detection using the Canny operator takes most of the execution time for processing an image.

Table 4 Average Execution Time for Processing an Image of 640×480 Pixels

Step	Edge detection	CC labeling	Preliminary filtering	Recognition time	CC fusion	Text extraction
Time (ms)	270	30	< 10	see Table 5	< 10	35

The recognition time for identifying a character block of 16×16 pixels is revealed in Table 5 which includes the duration of block resizing, wavelet transform (for taking wavelet features), feature extraction, and text identification by the AdaBoost strong classifier. Compared to adopt raw gray level features, the recognition time needs six more times when the wavelet features derived from the wavelet transform are employed.

Table 5 Average Recognition Time for Identifying a Character Block of 16×16 Pixels

Feature	Wavelet	Gray level
Recognition time (ms/block)	1.257928	0.208485

The total execution time for an image of 640×480 pixels depends on the number of candidate character blocks; in general, it is less than 1 second spent in the processing steps from edge detection to text extraction. Fig. 7 demonstrates some examples of text detection and extraction results from the final strong classifier using the Gentle AdaBoost algorithm with wavelet features and 16 CART splits in 200 iterations.

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

Fig. 7 Some examples of text detection and extraction results: (a) & (b) the original images; (c) & (d) the extracted texts from (a) & (b), respectively.

5. Conclusions and Future Works

In this paper, we have presented a robust text detection and extraction method based on CC labeling and an AdaBoost algorithm. Combining with the Canny edge detection and CC labeling, we can find out candidate character blocks precisely even in a noisy image. As the experimental outcomes show, our proposed method achieves a satisfactory result in most of different natural scenes and the computational time is also appreciatively.

Compared to Chen and Yuille's texture-based AdaBoost text detection method [12], we manipulate an image of 2,048×1,536 pixels by taking 2 seconds approximately, while their system needs about 3 seconds. Some future works are worth investigating to attain better performance. We can choose other feature extraction techniques; for example, multi-channel Gabor transform coefficients, or we can mix up distinct types of features.

Acknowledgement

The authors are thankful for this work supported in part by the National Science Council of Taiwan under Grant NSC95-2218-E-011-009.

References

[1] K. Jung, K. I. Kim, and A. K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition*, vol. 37, no. 5, pp. 977-997, 2004.

[2] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, 1996.

[3] C. S. Fahn, K. L. Lin, and K. Y. Chu, "An efficient two-phase two-scan labeling connected components algorithm," Submitted to *IEEE Transactions on Image Processing*, 2008.

[4] K. Suzuki, I. Horiba, and N. Sugie, "Linear-time connected-component labeling based on sequential local operations," *Computer Vision and Image Understanding*, vol. 89, no.1, pp. 1-23, 2003.

[5] F. Chang, C. J. Chen, and C. J. Lu, "A linear-time component-labeling algorithm using contour tracing technique," C*omputer Vision and Image Understanding*, vol. 93, no. 2, pp. 206-220, 2004.

[6] L. F. He, Y. Y. Chao, and K. Suzuki, "A run-based two-scan labeling algorithm," *IEEE Transactions on Image Processing*, vol. 17, no. 5, pp. 749-756, 2008.

[7] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning*, pp. 148-156, Bari, Italy, 1996.

[8] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: A statistical view of boosting," *The Annals of Statistics*, vol. 28, no. 2, pp. 337-407, 2000.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Chapman and Hall, New York, USA, 1984.

[10] J. Sauvola, T. Seppanen, S. Haapakoski, and M. Pietikainen, "Adaptive document binarization," in *Proceedings of the 4th International Conference on Document Analysis and Recognition*, Ulm, Germany, vol. 1, pp. 147-152, 1997.

[11] A. Vezhnevets, *GML Adaboost Matlab Toolbox*, Graphics and Media Laboratory, Computer Science Department, Moscow State University, Moscow, Russian Federation, http://research.graphicon.ru/.

[12] X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, vol. 2, pp. 366-373, 2004.