Extracting Text From Greyscale Images

Victor Wu
R. Manmatha

CMPSCI TR95-88

October, 1995

This work is supported by the Center for Intelligent Information Retrieval and by NRaD grant number N66001-94-D-6054. This work is also partially supported by the Vision Lab, Computer Science Department,University of Massachusetts at Amherst.

Extracting Text From Greyscale Images

Victor Wu
R. Manmatha

Computer Science Department
Lederle Graduate Research Center
University of Massachusetts
Amherst, MA 01003-4601
U.S.A.
vwu@cs.umass.edu
manmatha@cs.umass.edu
Tel: (413)545-0528
Fax: (413)545-1249

October, 1995

Abstract

A suite of algorithms is presented to detect text in images. The algorithms are shown to work on images with a wide variety of backgrounds. The algorithms assume that text is a special kind of texture characterized by a frequency distribution different from the background. An algorithm to remove the background and binarize the detected text so that it can be processed by an OCR is also presented. Experimental results are shown for a number of images.

Keywords — text detection and extraction, feature clustering, background removal, image smoothing.

1 Introduction

Current OCR and page segmentation schemes are largely restricted to finding text against good clean backgrounds ([2], [6], [1], [4], [3]). However, text is often found in images or against shaded or textured backgrounds such as maps, financial documents and engineering drawings. In addition, text found in images or videos can be used to annotate and index those materials since **image indexing** based on image content is difficult to do. Thus, a system which automatically detects and recognizes text in such contexts is highly desirable.

A suite of algorithms is presented to detect text in images with a variety of backgrounds. The detected text is then separated from the background and then binarized. The text detection and background removal algorithms work with machine generated fonts and script. Machine generated fonts which are readable by an OCR system are then passed to a commercial OCR for reading.

The algorithms have been tested on a variety of images from different sources, including newspapers, magazines, printed advertisement, photos, checks, invoices. Furthermore, the algorithms have been tested on images which have the following characteristics:

1. binary or greyscale
2. little restriction on character size
3. little restrictions on font
4. the text may overlap background textures (hatched or shaded background), drawings, and other non-text contents

This paper is organized as follows: section 2 gives a brief review of previous work. The overview of our system is presented in section 3 and 4 while detailed discussion on the techniques is provided in section 5. Although some results of experiments appears in the early sections, most of them are presented and explained in section 6. The text chip binarization algorithm is described in 7. Lastly, we conclude and summarize our work in section 8.

2 Previous Work

A few systems have been developed in the past to detect text in images in specific contexts. These include engineering drawings, maps, bar codes and newspapers, and envelopes [8].

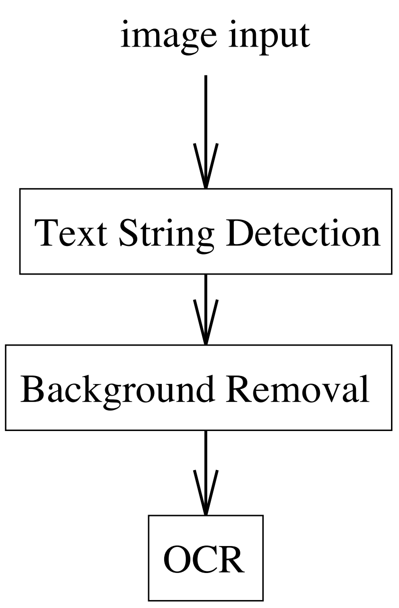

Figure 1: Three phases of the system

Many people have developed algorithms to do text extraction. However, most of them have severe limitations on the types of images. One group of the algorithms can only work with binary images [4, 5, 1]. The ones which can handle greyscale images either require that the images has few variation in intensity [6], require prior knowledge of the character stroke width [3], or require some prior information about the distribution of gray levels for text and non-text [7]. Only one algorithm that we know of has addressed the issue of text strings at any orientation [2].

3 System Overview

This section provides an overview of the main components of our system. Details on the techniques used are provided in section 5.

The solution to the text extraction problem is divided into three phases (Figure 1): in the first phase, regions in the image which are likely to correspond to text strings are identified (**text detection**). Rectangular bounding boxes are used to represent the regions (called *chips*). This phase is crucial since it focuses subsequent processing to those restricted regions (**focus of attention**). In the next phase, the detected text regions are binarize by setting pixels of the text as foreground (e.g. 1 or ON) and the rest of the pixels as background (e.g. 0 or OFF). This phase will be called **Background Removal**. Lastly, the binarized

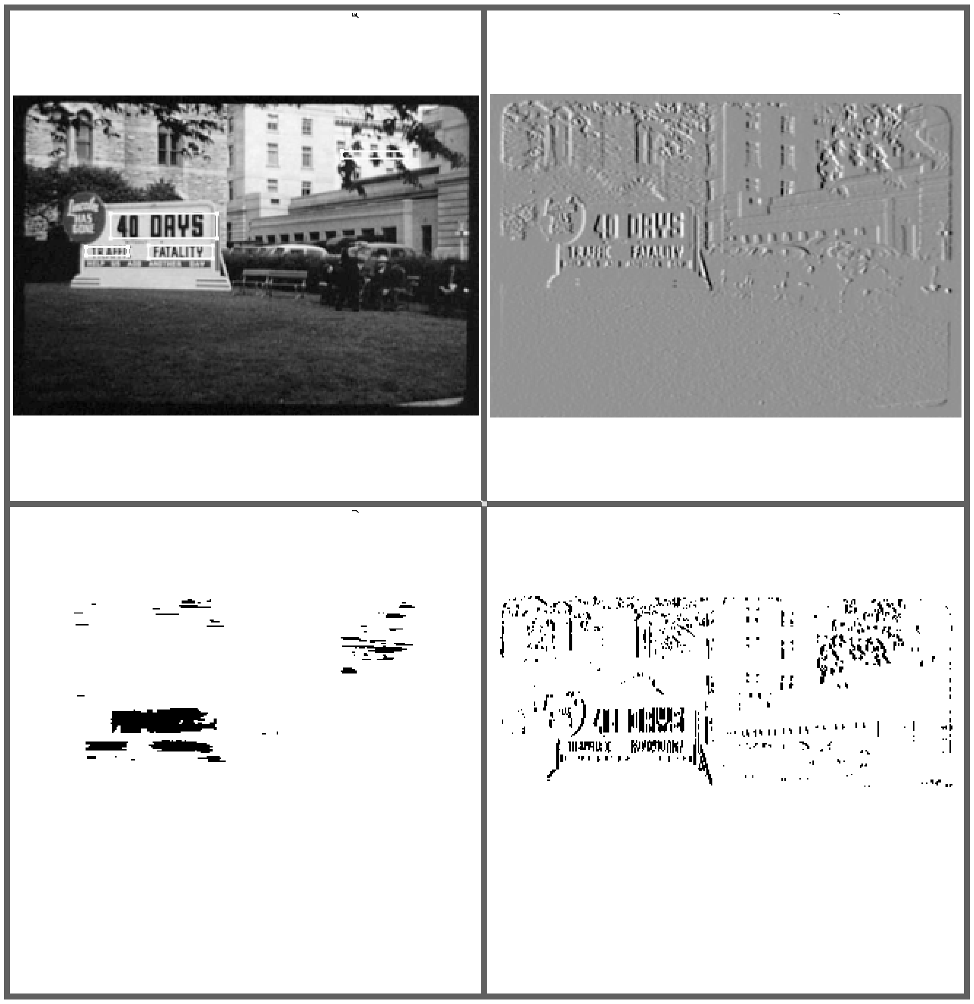

Figure 2: Clockwise from upper left: (a) original image & detected text boxes; (b) first order derivative along the X direction; (c) binarization of b using threshold = 50; (d) horizontal smoothing of c.

text chips are fed into a commercial **OCR** system for character recognition.

4 Text String Detection

The *strokes* of text strings correspond to certain frequencies. Intuitively, the wider the spacing between characters, the lower the corresponding frequency. There are two essential conditions that make the text strings **noticeable** by a human being: **the distinct contrast between the characters and background, and the more or less regular patterns of appearances of the character strokes, such as similar heights, orientation and spacing**. The latter can be interpreted as frequency and outline shape.

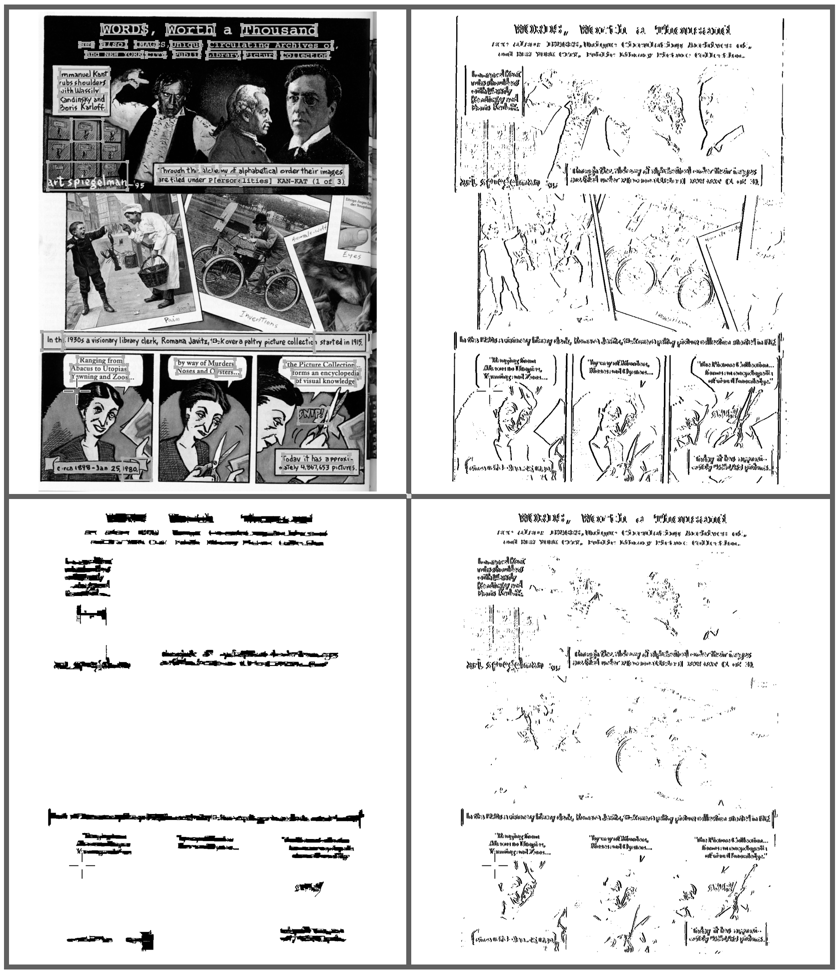

Figure 3: Clockwise starting from upper left: (a) original image with rectangle boxes representing the identified text regions; (b) output of K-means clustering with $K = 4$; (c) result of filtering of b; (d) identified text regions

The word "noticeable" is used here instead of "readable" because one does NOT need to be able to read each individual characters to tell where the text strings are. Imagine looking at a comic page of a newspaper a few feet away. Even though one may not be able to recognize the characters individually, one can probably still detect where the text is.

Thus text has two distinct characteristics. It may be viewed as a distinctive kind of texture characterized by a certain frequency content. In addition, text shows spatial cohesion — characters are joined to form words, words to form sentences. Rarely are characters found by themselves.

The above observation inspires the key ideas embedded in the two different methods described here for text string detection. Although the two methods are different, they both

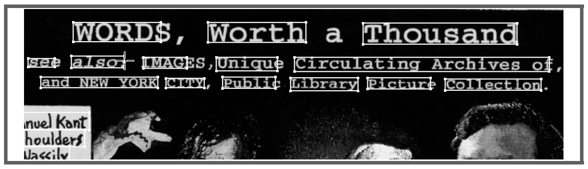
WORD$, Worth a Thousand
see also IMAGES, Unique Circulating Archives of,
and NEW YORK CITY, Public Library Picture Collection.
nuel Kant
houlders

Figure 4: Magnified part of Figure 3

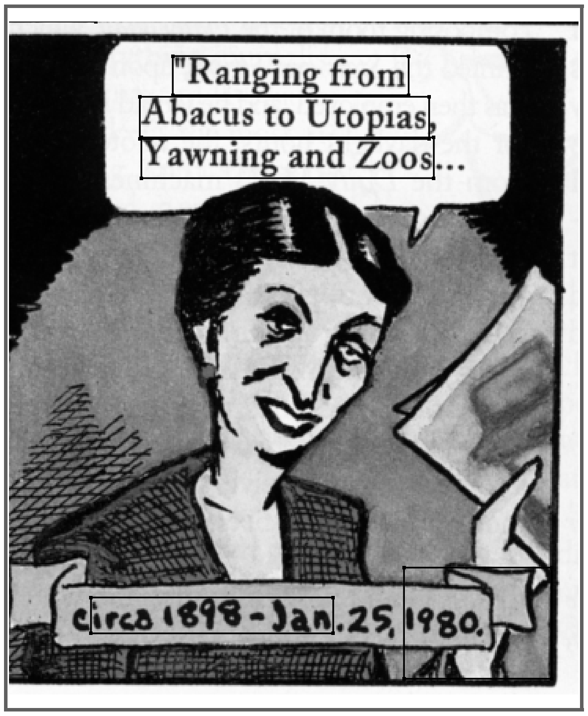

Figure 5: Magnified part of Figure 3

utilize the contrast and the inherent frequency of text strings as the crucial cues for text region identification.

4.1 A Simple Text Detector

The following is a simple but surprisingly effective algorithm for text string detection. It is also an efficient linear time algorithm. Here are the main steps:

1. Reduce the image by Gaussian filtering and then sub-sampling. The amount of reduction depends on the actual size of the text. (In some cases, no reduction is required).

2. Filter the image with a 1-D Gabor filter along the X-direction (Figure 2 (b)).

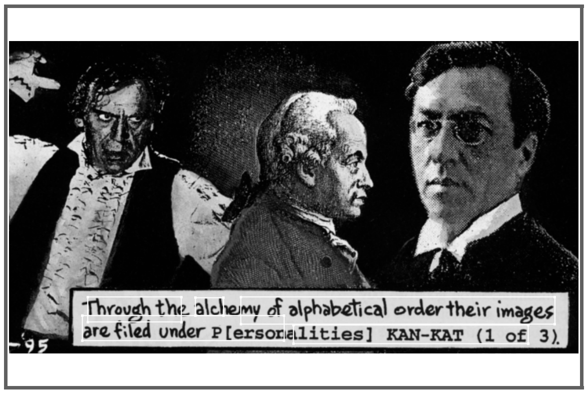
Through the alchemy of alphabetical order their images
are filed under P[ersonalities] KAN-KAT (1 of 3).

Figure 6: Magnified part of Figure 3

3. Select a proper threshold to generate a binary image in which only the pixels at which the derivative value exceeds the threshold are set to 1 (significant contrast) (Figure 2 (c))

   This step reflects the idea of finding significant contrast as noted in the observation above.

4. Smooth the binary image horizontally to form larger connected regions (Figure 2 (d)).

5. Find connected components.

6. Compute the minimum rectangular bounding boxes of the connected regions.

7. Select boxes which satisfy certain constraints such as minimum/maximum height/width, orientation, etc, and hence are most likely to correspond to text. (Figure 2 (a)).

4.2 Feature Clustering Based Text Detector

There are several weaknesses of the method in section 4.1, the most critical of which is that it is rather sensitive to the value of the threshold in step 3. It also has a tendency to fragment large-sized text (**scale problem**).

In this section, another approach is proposed for text region detection which is quite different from the one presented in section 4.1. The main steps are the following:

1. Smooth the input image with Gaussian masks of N different standard deviations (σ). For example, $N = 3$ was used in all the examples. Save each output corresponding to each σ in an image.

2. Compute the first order derivative along X direction on each of the outputs of step 1.

3. Form a N dimensional **feature vector** at each pixel by using the outputs of the derivatives from step 2 .

4. Classify the feature vectors into K classes (e.g. $K = 4$) using a minimum distance clustering algorithm.

5. Select the pixels with features belonging to text strings. In the case when $K = 4$, take the two clusters which are the furtherest away from the background cluster (the cluster which contains feature $\{0, 0, 0, ..., 0\}$).

   The output now is a binary image in which pixels which might correspond to text have value 1 and the rest of the pixels have value 0. (Figure 3 (b)).

6. Filter **the connected components** (**tokens**) in the output of step 5 to remove the tokens which are not likely to be part of a text string (Figure 3 (c)).

7. Connect the tokens which are likely to be part of the same text string to form larger tokens which covers more of the region of the text strings. Then, filter out tokens which are not likely a part of a text region (Figure 3 (d)).

8. Compute the horizontal minimum bounding boxes for each of the remaining tokens and output them as identified text regions (Figure 3 (a)).

Steps 1 – 3 basically compute the feature vector corresponding to each pixel in the input image. Again, the first derivative computation captures the contrast at each pixel location. The Gaussian masks are used not only to smooth out non-text high frequency details of the image, but also to measure the frequency information. It should also be noted that each feature is normalized to have zero mean and unit standard deviation. This is to prevent one feature from being dominant in the clustering process.

Step 4 is discussed in detail in section 5.2. The new filtering steps 6 – 8 are discussed in detail in section 5.3.

Figure 4-5 show some regions of Figure3 , enlarged to make the result more readable. Notice that there are a large amount of false positives in the middle section in Figure 3 (b), but then all of those are filtered out in the end (Figure 3 (d)).

5 Further Discussion

In this section the techniques used in the algorithms proposed in the previous section are described in detail.

5.1 Image Smoothing

Many functions have been proposed in signal/image processing literature for smoothing purpose, of which the most often used is perhaps the Gaussian function. The 1D version of the Gaussian is defined as

$$G(x, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

where σ is called the standard deviation. It is clear that $G(x)$ is symmetric. It is also straight forward to show that $G(x)$ is a low pass filter and the larger the σ_x is, the lower the cutoff frequency. The first derivative of an image, I, is computed by convolving I with a 1×3 array $\{1, 0, -1\}$.

5.2 Feature Space Clustering

A K-means clustering algorithm is used to cluster the feature vectors [9]. This algorithm is based on the minimization of the sum of squared distances from all vectors in a cluster to the *cluster center*. The number of desired clusters (classes), K, has to be provided for this algorithm, together with the initial cluster centers for each of the clusters. Vectors are then iteratively distributed among the K cluster domains. New cluster centers are computed from these results, such that the sum of the squared distances from all points in the cluster is minimized.

A practical upper bound on the iterations is chosen for convergence since K-means does not really converge. 100000 is used as the upper bound in our system.

We use the version of K-means implemented in KHOROS[1]. In our case, K is selected to reflect how different the text strings look in terms of contrast. For example, if the text strings have roughly the same contrast (which in many cases is true), $K = 4$ is used: one for the background, one for the other non-text content which includes "noise" and two for the text

[1]KHOROS is the visual programming system invented by The Department of Electrical and Computer Engineering, University of New Mexico

(one counts for positive contrast and one for negative contrast). If the documents contains text with two significantly different contrasts, then use $K = 6$ and select the two classes which are furthest away from the background class (the one which contain the feature vector $\{0, 0, \ldots, 0\}$) as candidates for the group of text which has the highest contrast; select the two classes which are the furthest away from the background class in the remaining classes as the candidates for the group of text which has less contrast.

When the number of feature vectors (which equals the number of pixels in the input image in our case) and the dimension of the feature vector get large, K-means becomes unpractical since it take too long to run. In this case, we randomly select a small percentage (e.g 20%) of the total vectors to run the clustering algorithm, then use computed cluster centers to classify all the feature vectors into classes.

More information on feature classification is available in [9].

5.3 Forming the Text Regions

Feature vector clustering generates both background pixels and foreground pixels. The foreground pixels can further be grouped as tokens such that each token represent one connected component (**strokes**). In reality, many strokes are misclassified ones (**false positive**) as shown in Figure 3 (b). Also, the strokes have to be grouped together to represent the text regions. Thus, we need methods to

(a) filter out those false positives;

(b) group the strokes to form regions corresponding to the text strings in the image;

Furthermore, the regions should be tight enough so that the background removal algorithm proposed in section 7 can be applied.

Recall that text strings correspond to certain frequency and have regular bounding boxes. Also, the heights of strokes are somewhat uniform for each text string. This observation is used to filter the tokens as follows:

Given the clustering output (the binary image which contains the possible strokes (tokens)):

1. Compute the population distribution of tokens according to their heights. More specifically, for each height h, compute $P(h, \gamma)$ which is the number of tokens which have

heights between $[h-\gamma, h+\gamma]$. γ is a parameter that reflects the fact that not all strokes in a text string are necessarily of the same height. $\gamma = 10$ in all the experiments.

2. Compute the max size of the m most popular gaps on row y, denoted $G(m, y)$.

   A gap is defined as any segment of connected OFF pixels between two ON pixels in the same row and the size of a gap is the number of pixels in the gap. Thus, $G(m, y)$ reflects the fact that the text strings normally have higher frequencies than false positives do.

   m is set to 4 in our experiments.

3. Discard tokens if the population of its height (h), $P(h, \gamma)$, is less than a preset threshold τ_h. $\tau_h = 15$ was used in the experiments.

   This procedure helps filter out the long vertical lines (since they are usually few in number) as shown in Figure 3 (b) and (c).

4. Discard tokens which do not have tokens near by.

   More specifically, given a token of height h, with lowest pixel at$(x0, y0)$ and highest pixel at $(x1, y1)$ (hence $h = y1 - y0 + 1$), scan each row, y, between $y0$ and $y1$. Suppose (x_l, y) is the left most ON pixel in row y in the token and (x_r, y) is the right most one, look left from (x_l, y) (and right from (x_r, y)) to see if there is another ON pixel within distance $G(m, y) + \beta$. If there is one on either side or both, mark that row as "good". If less than $\theta \times h$ rows in the token are good, discard the token.

   In the experiments, $\beta = 3$ and $\theta = 50\%$.

The above filtering step remove many false positive strokes (Figure 3 (b) and (c)). The next thing to do is to connect the remaining strokes to form bigger regions. This is done in the following way: Given a token A, look on both of its sides to see if there is another token nearby in the same way as in step 4, but look only as far as the height of the token (i.e., if the height is 20, then look as far as 20 pixels away. We refer to this as the **reaching distance** of the token). If there is another token B within the reaching distance, and A is also within the reaching distance of B, connect A and B if they satisfy one of the following:

1. $(y_{A,hi} \leq y_{B,hi}) \wedge (y_{A,low} \geq y_{B,low})$

2. $(y_{A,hi} \geq y_{B,hi}) \wedge (y_{A,low} \leq y_{B,low})$

3. $(height_{comm} \geq h_{thr} \times height_A) \vee (height_{comm} \geq h_{thr} \times height_B)$

where h_{thr} is a preset threshold value and $height_{comm}$ the number of pairs of pixels, each from one of the tokens, such that they have the same y coordinates (size of overlapping segment if we project both A and B onto the Y-axis). $y_{A,hi}$ is the largest y coordinates of all the pixels in token A. Similarly, $y_{A,low}$ is the lowest y coordinates of all the pixels in token A. All the rest of the variables bear the obvious meaning.

The above connecting operation can be applied repeatedly to form longer regions.

Lastly, we exam each of the region by its width, height and number of gaps averaged by it height to discard the regions which are too narrow (e.g., less than 5 pixels) in either direction or too low (< 8) of the average number of gaps.

6 Experiments

In our experiment, the images come from a variety of sources. Some are down-loaded from internet; some are scanned at 300dpi (dots per inch) resolution (Figures 7, 11, 12); The images scanned are from very different sources: the image in Figure 3 is scanned from the New Yorker magazine; the one in Figure 11 is from a school newspaper; the one in Figure 12 is a personal cheque; the one in Figure 7 is from a magazine advertisement.

Figure 7 demonstrates another interesting property of our algorithm: It captures text which varies significantly in size. Our algorithm also has some tolerance for text which is not horizontally aligned.

Figure 9 shows the result on a image scanned from a (seriously degraded) Fedex invoice. Even though the image is of such poor quality, our algorithm can still pick out a large portion of the text. Figure 10 show zoomed-in portions of Figure 9 (a).

Figure 8 shows the results of applying both text detectors on the same image. It also shows another interesting property of our algorithms: it detects text written in script fonts , such as the trademark "Coca Cola".

The second text detector failed to completely identify the text located at the lower left part of the image. The reason is that the text string violated the assumption that it should be straight.

7 Background Removal

Figure 14 shows an example of text printed against a shaded background. Such examples are not uncommon. Sometimes shaded backgrounds are used in currency or other applications to prevent copying. OCR's cannot handle such backgrounds. They usually require that the background be removed and the text binarized.

Text in shaded backgrounds still needs to be read by humans. Thus the text needs to be printed in a way which distinguishes shading from text. This is often done by using fine lines or lots of small dots (high frequency) for the shading. The high frequency ensures that copying is difficult. Compared to the shading, the text is of much lower frequency. Thus a technique to remove the background can be constructed by using filters to separate the shading and the text.

In general, it is difficult to find a global threshold to binarize text (i.e., there is usually no single threshold which can be applied to the entire image to separate the text from non-text), as noted by many researchers ([6], [8]).

However, the binarization algorithm proposed here is applied to the text chips detected in the previous sections. Thus, it is a local method and therefore likely to succeed.

The following algorithm for background removal and binarization is proposed:

1. Smooth the text chip identified by the text detectors proposed above. The smoothing operation affects the background much more than the text. Another way of looking at this is to notice that the smoothing blends the white and black in the background into grey while leaving the text black (see second row of Figure 14).

2. Compute the intensity histogram of the smoothed chip.

3. The histogram is jagged and needs to be smoothed to allow the valley to be detected. This is done by smoothing the histogram using a low-pass filter (see third row at right in Figure 14).

4. The darkest area in the smoothed image is the text.Therefore, a threshold is picked at the first valley closest to the dark side of the histogram. The thresholded image is shown on bottom left in Figure 14. This can be successfully recognized by an OCR.

The method is applicable to a wide variety of backgrounds and also successful in removing noise. If the text in the chip is brighter than the rest of the pixel, the pixel values may be inverted and then the above algorithm may be applied to select the text.

Figure 15 shows some results of the background removal process.

7.1 OCR

The output of the background removal phase can be feed to an OCR system for character recognition.

8 Conclusion

Automated document analysis has become an very important research topic and gained much attention from researchers working on areas such as information retrieval, multimedia, computer vision and pattern recognition. Images normally contains enormous amount of data, thus it is highly desirable to develop automatic methods for extracting text from images so that the text can be stored in symbolic form which requires minimal space and make the future access of the text much easier. In this paper, two methods for automatic text region detection are proposed. They are applicable to a variety of types of images. An algorithm for background removal is also proposed. Standard computer vision and pattern recognition techniques such as feature clustering, image convolution and smoothing, intensity histogram analysis and high level information (tokens) processing, are applied in a novel way so that the algorithms performs impressively well, as shown in the experiment results.

One property of the approach which is worth noting is that it detects and extracts text with complicated or textured backgrounds. This expands the domain of it application greatly and distinguishes itself from most of the published methods.

The methods also detects text which varies significantly in size in the same image. No parameter adjusting is needed.

It should also be noted that our methods works for text in different fonts, including some script fonts.

Acknowledgements

We thank Jonathan Lim for useful discussions and system support. We also thank Yongqing Cheng, Bruce Croft, Ed Riseman and Al Hanson for helpful comments, critics and support.

References

[1] Lloyd Alan Fletcher and Rangachar Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 10(6):910–918, Nov. 1988.

[2] Anil K. Jain and Sushil Bhattachariee. Text segmentation using gabor filters for automatic document processing. *Machine Vision and Applications*, 5, 1992.

[3] Mohamed Kamel and Aiguo Zhao. Extraction of binary character/graphics images from grayscale document images. *Computer Vision, Graphics and Image Processing*, 55(3):203–217, May. 1993.

[4] Su Liang and M. Ahmadi. A morphological approach to text string extraction from regular periodic overlapping text/background images. *Computer Vision, Graphics and Image Processing*, 56(5):402–413, Sept. 1994.

[5] Huizhu Luo and Its'hak Dinstein. Using directional mathematical morphology for separation of character strings from text/graphics image. IAPR International Workshop on Structural and Syntactic Pattern Recognition, Naharia, Israel, Oct 1994.

[6] Lawrence O'Gorman. Binarization and multithresholding of document images using connectivity. *Computer Vision, Graphics and Image Processing*, 56(6):494–506, Nov. 1994.

[7] Torfinn Taxt, Patrick J. Flynn, and Anil K. Jain. Segmentation of document images. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 11(12):1322–1329, Dec. 1989.

[8] Øivind Due Trier and Torfinn Taxt. Evaluation of binarization methods for document images. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 17(3):312–315, March 1995.

[9] Tzay Y. Young and Thomas W. Calvert. *Classification, Estimation And Pattern Recognition.* American Elsevier Publishing Company, Inc, 1974.

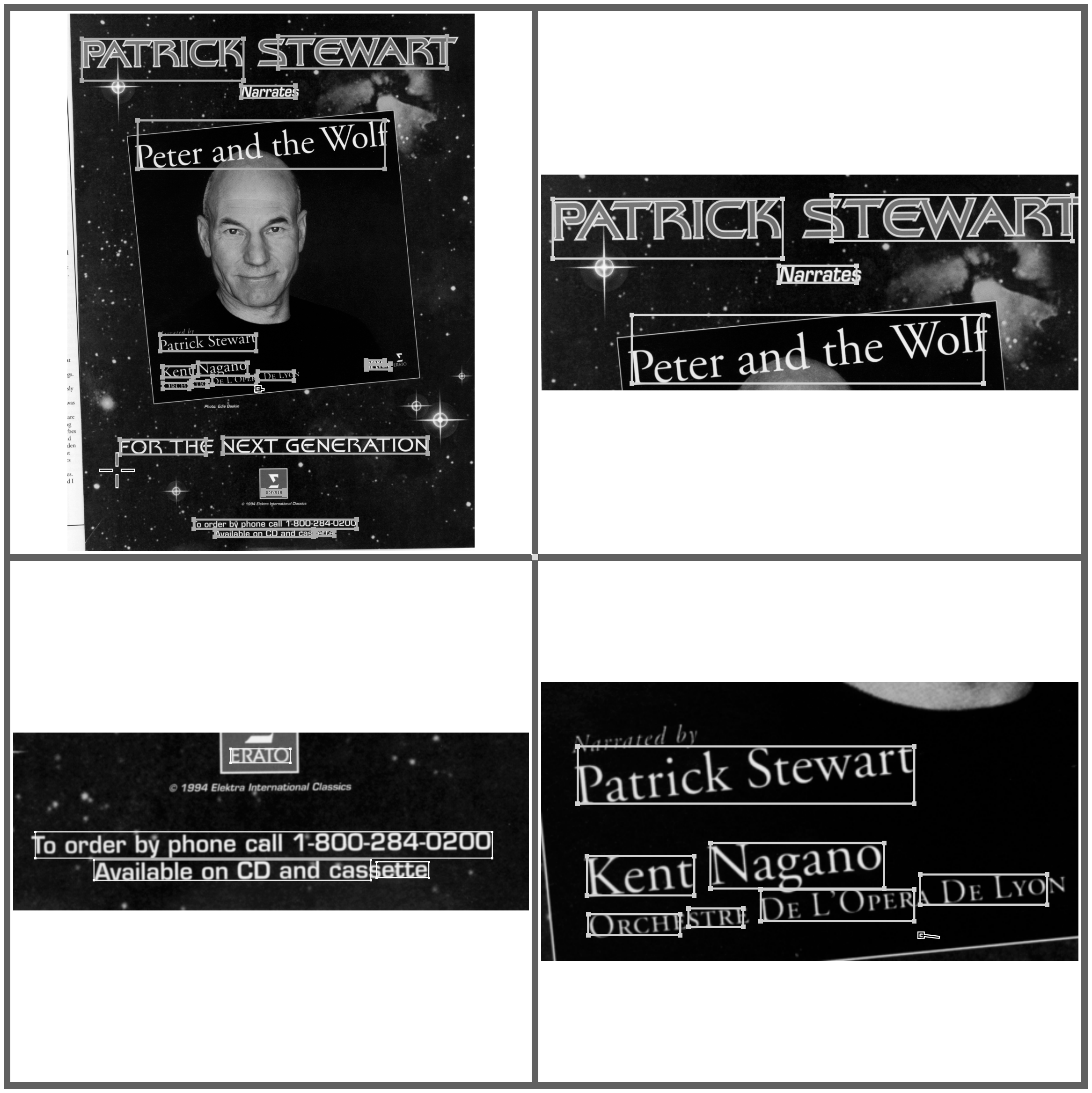
PATRICK STEWART
Narrates
Peter and the Wolf
Narrated by
Patrick Stewart
Kent Nagano
Orchestre de l'Opera de Lyon
FOR THE NEXT GENERATION
ERATO
© 1994 Elektra International Classics
To order by phone call 1-800-284-0200
Available on CD and cassette

Figure 7: Upper left window shows the original image with rectangle boxes representing the identified text regions; the rest of the windows show magnified parts of the upper left window. The second text detector is used.

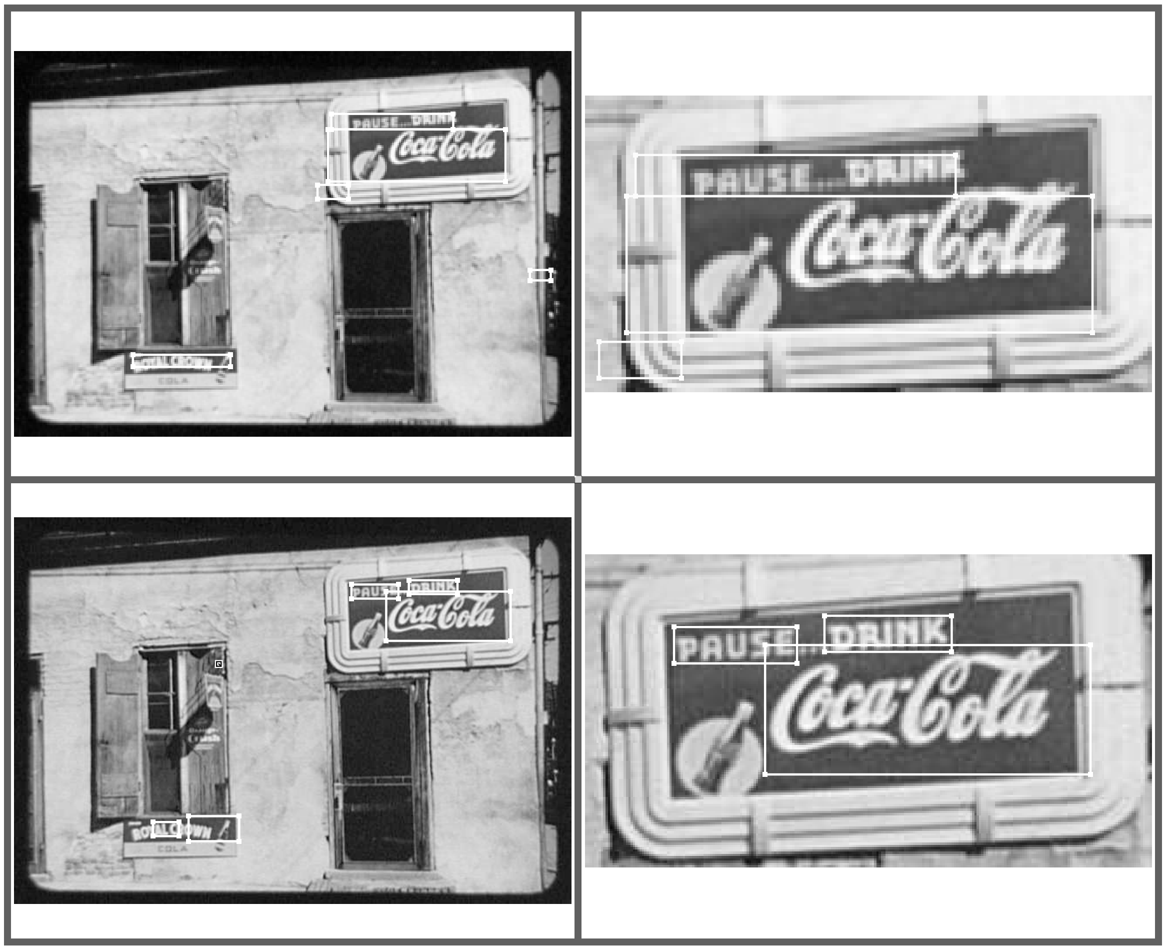

Figure 8: Upper left: original image with rectangle text boxes identified by the simple text detector; Lower left: text boxes identified by the text detector using K-means algorithm. On the right are a small region of what is shown on its left.

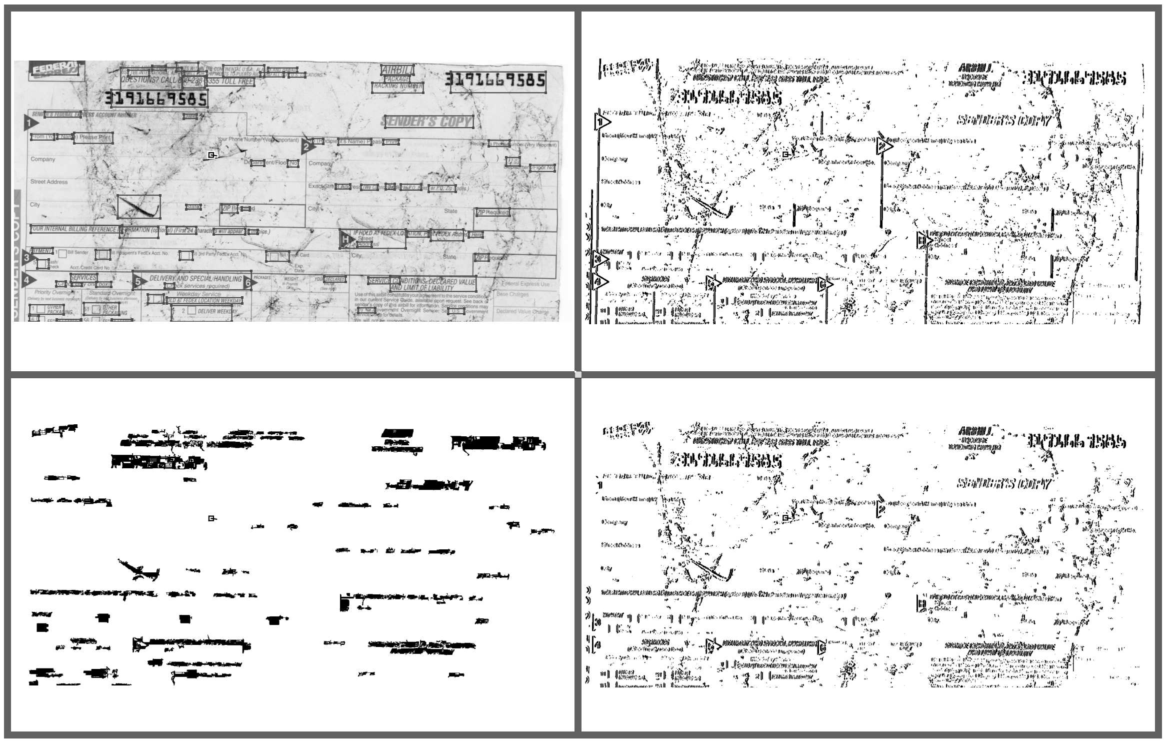

Figure 9: Clockwise starting from upper left: (a) original image with rectangle boxes representing the identified text regions; (b) output of K-means clustering with $K = 4$; (c) result of filtering of b; (d) identified text regions

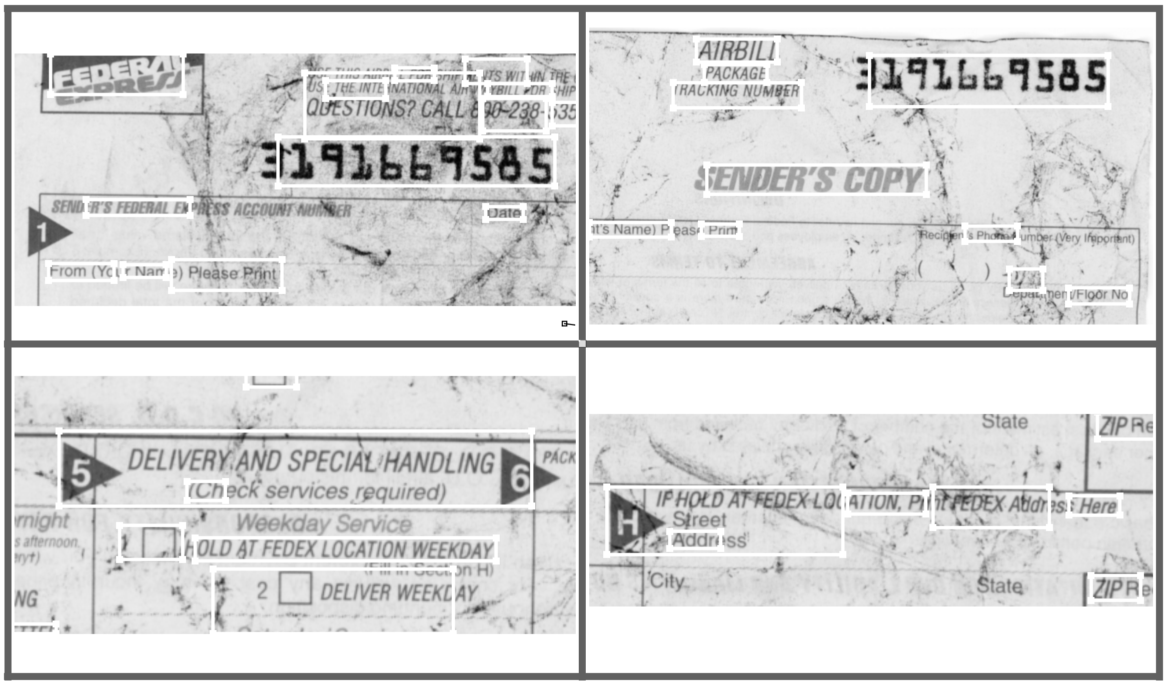

Figure 10: Parts of Figure 9

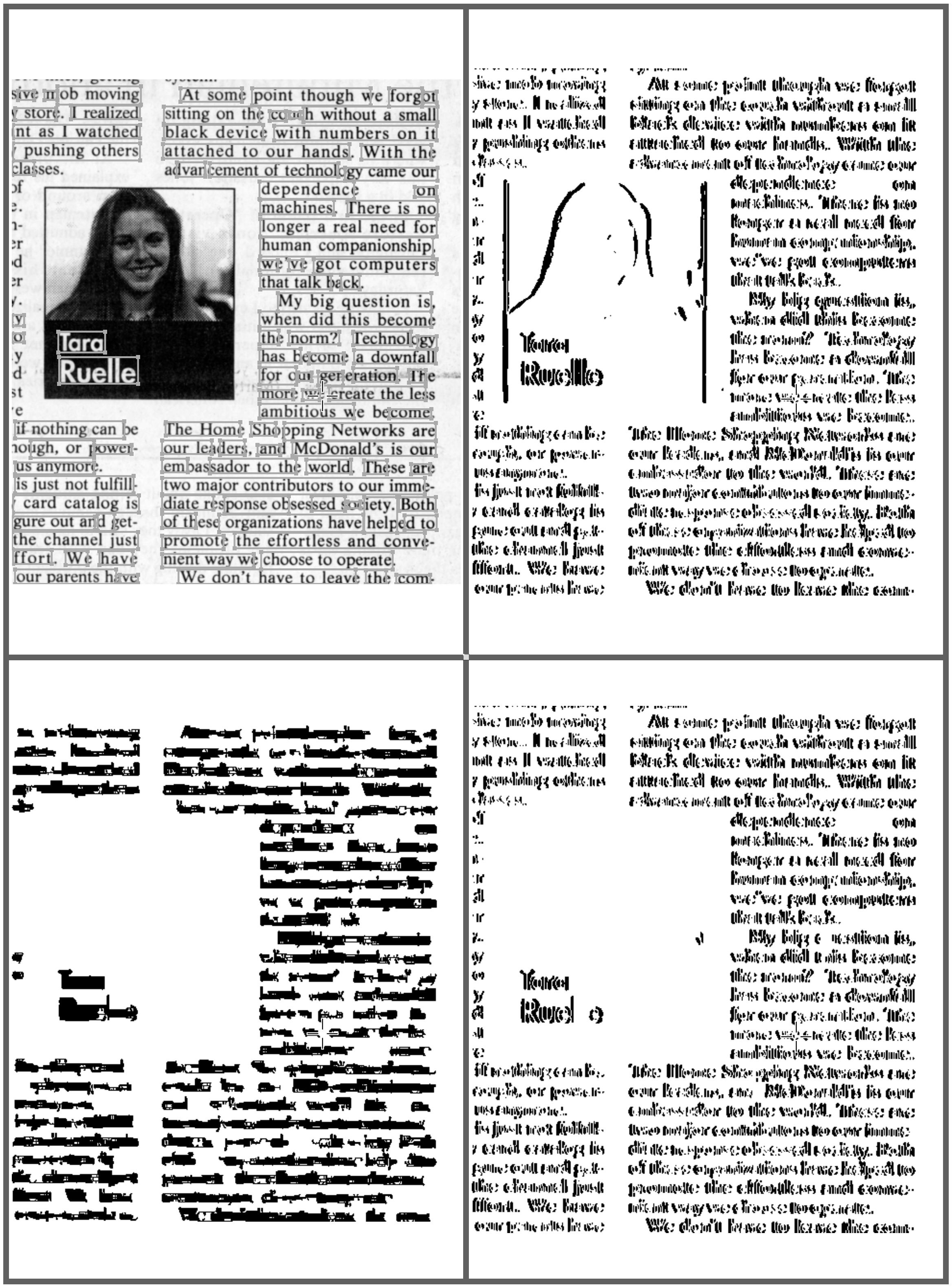

Figure 11: Clockwise starting from upper left: (a) original image with rectangle boxes representing the identified text regions; (b) output of K-means clustering with $K = 4$; (c) result of filtering of b; (d) identified text regions

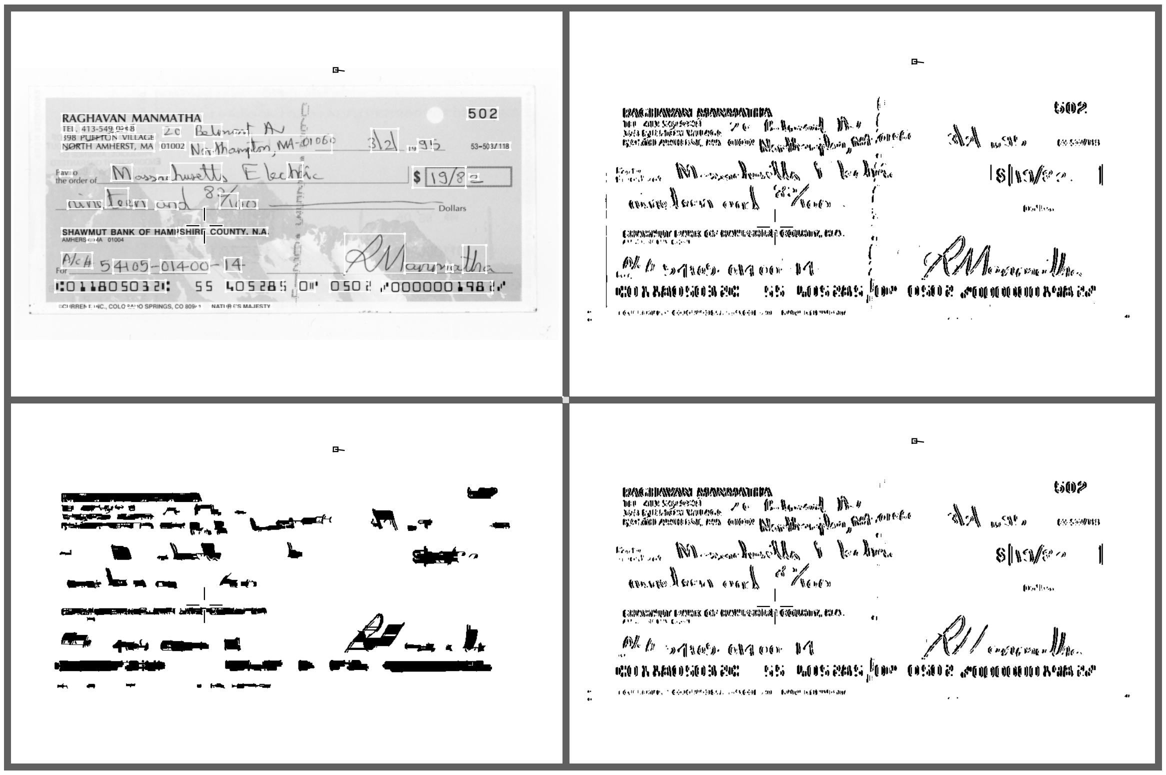

Figure 12: Clockwise starting from upper left: (a) original image with rectangle boxes representing the identified text regions; (b) output of K-means clustering with $K = 4$; (c) result of filtering of b; (d) identified text regions

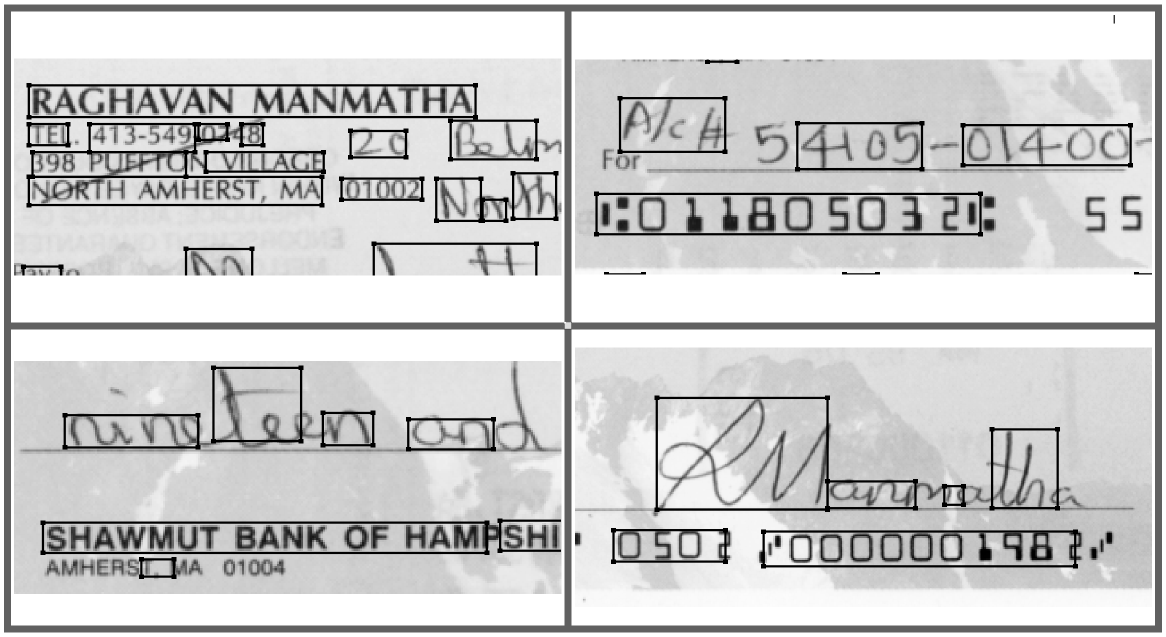

Figure 13: Parts of Figure 12 (a)

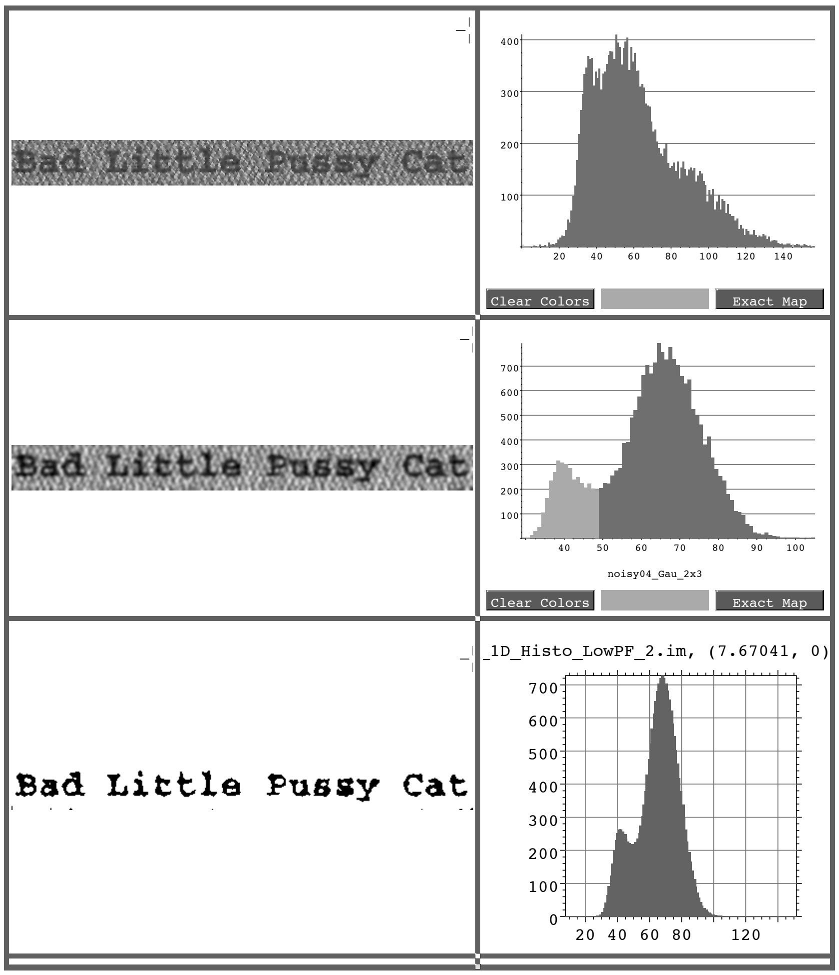

Figure 14: Top to Bottom: Left (1) original text chip (2) smoothed version of 1 (3) final binarization result; Right: plots of histograms of the images on the left

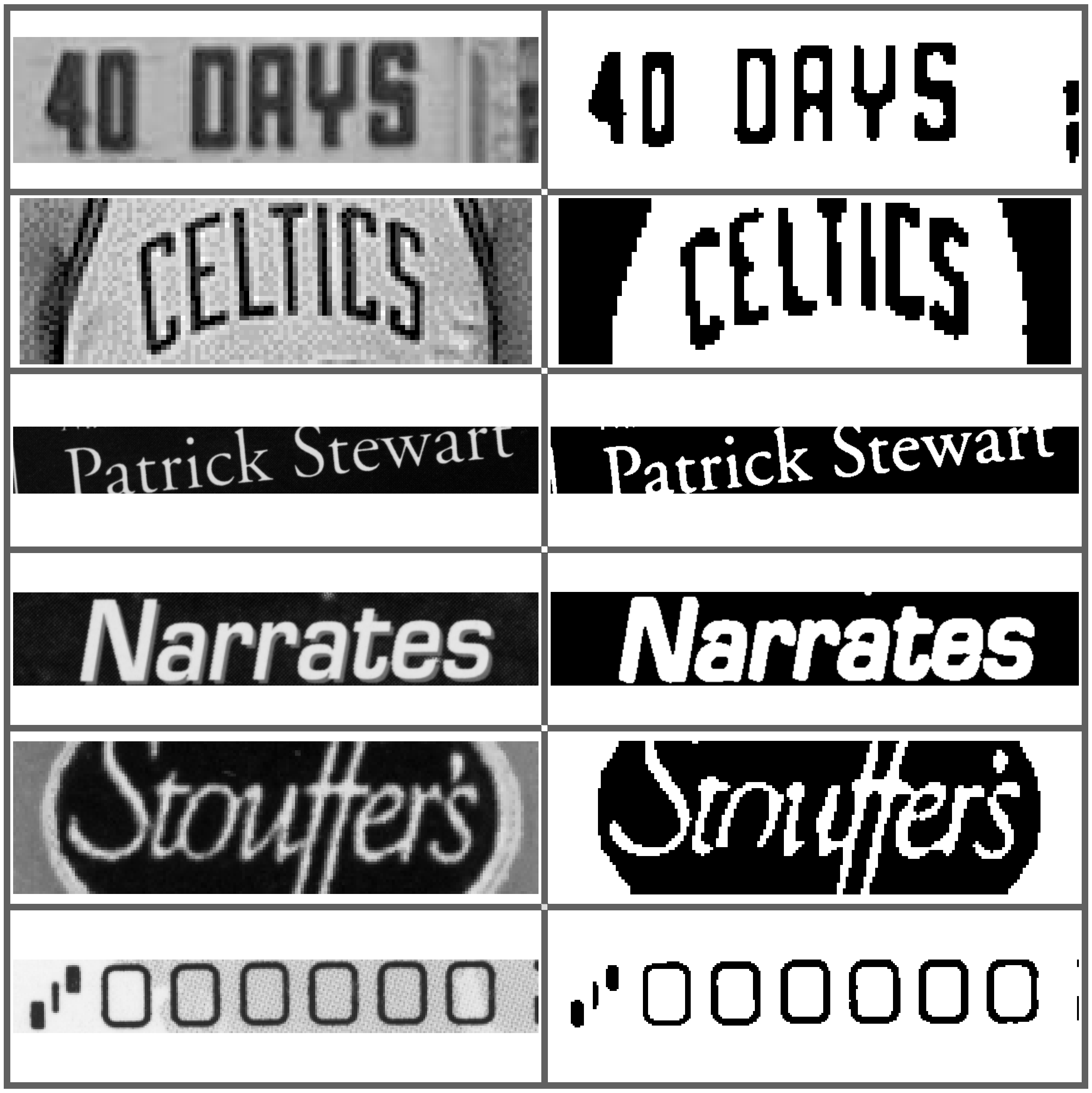

Figure 15: On the left are text chips generated by the text detectors; on the right are the corresponding results of the background removal algorithm.