Finding Text in Color Images*

Jiangying Zhou[a] Daniel Lopresti[b] Tolga Tasdizen[c]

[a]Panasonic Information and Networking Technologies Laboratory
Two Research Way, Princeton, NJ 08540

[b]Bell Laboratories, Lucent Technologies Inc.
700 Mountain Ave, Room 2C-552, Murray Hill, NJ 07974

[c]Division of Engineering, Box D, Brown University, Providence, RI 02912

ABSTRACT

In this paper, we consider the problem of locating and extracting text from World Wide Web images. A previous algorithm based on color clustering and connected components analysis works well as long as the color of each character is relatively uniform and the typography is fairly simple. It breaks down quickly, however, when these assumptions are violated. In this paper, we describe more robust techniques for dealing with this challenging problem. We present an improved color clustering algorithm that measures similarity based on both RGB and spatial proximity. Layout analysis is also incorporated to handle more complex typography. These changes significantly enhance the performance of our text detection procedure.

Keywords: text detection, information retrieval, GIF images, World Wide Web.

1. INTRODUCTION

The World Wide Web has evolved into a vast repository of truly multimedia information. Plain text, graphics, images, and audio and video data have all become integral parts of WWW pages. Hence, the issue of developing effective methods for searching such diverse media is becoming increasingly important.

Some progress has already been made in this area. There are, for example, various well-known search engines for retrieving Web pages based on text extracted from HTML. Research is also ongoing on developing retrieval methods for non-textual data, such as images, audio, and video.

Much work is still needed, however. Currently, even access to textual information is incomplete. This is because a significant amount of the text on a given Web page can be embedded in images and completely absent from the HTML.[1] Examples of this include banners, icons, headlines, *etc.*, which may contain the most valuable indexing information for the page in question. In extreme cases, all of the text on a page might be present soley in image format (see, for example, `http://www.nytimes.com`). Existing commercial search engines are limited to indexing the raw ASCII text they find in the HTML – they cannot recover image text. This situation presents a new and exciting challenge for the field of document analysis.

Locating text in color images has been addressed under different contexts in the literature.[2–5] Recently, the issue of extracting text from WWW images has attracted the attention of several groups of researchers.[6,7] A paper by Zhou and Lopresti reviews the work done in the area up to that point in time.[8]

In this paper, we describe a method for locating and extracting text from color GIF images, the predominant format used on the World Wide Web. A previous algorithm based on color clustering and connected components analysis works well as long as the color of each character is relatively uniform and the typography is fairly simple. It breaks down quickly, however, when these assumptions are violated. Our focus here is on several new techniques we have developed for dealing with these problems. An improved color clustering algorithm measures similarity based on both RGB and spatial proximity. Layout analysis has also been incorporated to handle more complex typography. These changes significantly enhance the performance of our text detection procedure.

* Presented at *Document Recognition V (IS&T/SPIE Electronic Imaging)*, San Jose, CA, January 1998.

Correspondence concerning this paper can be addressed to the first author. E-mail addresses are: jz@research.panasonic.com (Jiangying Zhou), dlopresti@bell-labs.com (Daniel Lopresti), tt@lems.brown.edu (Tolga Tasdizen).

The rest of the paper is organized as follows: Section 2 gives an overview of the problem. In Section 3, we describe in detail our approach for detecting text in GIF images. Experimental results are presented in Section 4. Finally, we conclude the paper in Section 5.

2. TEXT IN GIF IMAGES

Text in GIF images has certain distinguishing characteristics. One key feature is the presence of multiple colors. Typically, scanned documents are bi-tonal or gray-level, while Web images make liberal use of color.

Image text also tends to be more diverse in shape and appearance. Attracting attention is a primary goal in Web graphic design. A large number of different typographic effects have been developed for this purpose. For example, text can be rendered using textures, shadows, 3-D embossing, *etc.* (see Figures 9 and 10). Text may also be overlayed on complex backgrounds and made so nearly transparent that it would be impossible to locate much less recognize using existing approaches. These techniques make the image more interesting to look at, but much harder to analyze for its textual content.

Even in cases where the text is presented in a uniform color, there still may be variations at the pixel level due to anti-aliasing, lossy compression, *etc.* The former is the process of reducing the jagged appearance of sharp edges by blending boundary pixels with the background; the relatively low spatial resolution of computer monitors (72 dpi) often makes anti-aliasing desirable. On the other hand, the application of block-oriented, lossy compression schemes such as JPEG can lead to visible distortion in the vicinity of characters. Neither of these effects arises in traditional scanned documents.

The 3-D color space also brings significant complications to the processing of GIF images. In order to analyze the shapes of objects in an image, one must first group pixels belonging to the same object together. For bi-tonal images, such groupings can be achieved by well-developed methods such as connected components analysis. These procedures are not readily applicable in color space since the notion of a "connected component" cannot be defined in the same way.

3. DESCRIPTION OF THE METHOD

An underlying assumption in our current text detection algorithm is that each character is rendered in a *homogeneous* color. This covers GIF images in which the text is composed of a single color, or a simple, uniform texture. It excludes, for example, instances where a character is rendered with its top half one color and its bottom half another, different color.

Our approach is to divide the problem into three stages: color clustering, character detection, and layout analysis. Color clustering groups together multiple colors that are part of the same text and reduces the overall number of colors present in the image. Character detection identifies connected components in each color group and classifies them as "character" or "non-character." In the layout analysis stage, we check to see which of the detected characters are consistent with possible text layouts and eliminate those that do not meet the requirements.

3.1. Color Clustering

The goal of color clustering is to identify pixels from the same piece of text and group them into a single cluster. In our previous work, we developed a global clustering procedure which relies only on the RGB similarity of colors in the image.[8] The approach we describe here uses a hierarchical method which combines both the RGB similarity and the local spatial proximity of the colors.

3.1.1. Clustering in RGB space

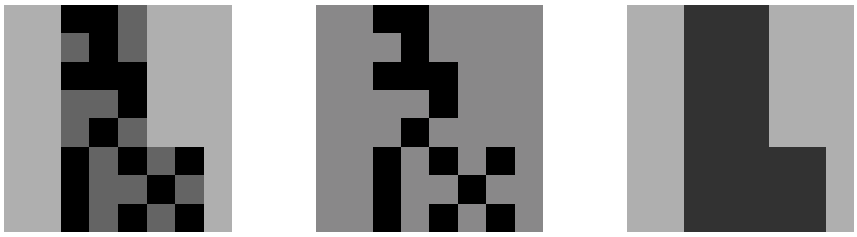
The RGB clustering method we employ is based on the Euclidean minimum-spanning-tree (MST) technique. We view each color as a point in a three dimensional RGB space. For a Web image in GIF format, there are at most 256 unique colors; hence, the number of nodes in an MST is no more than 256. The distance between two colors is computed as the Euclidean distance of their RGB elements. Once the MST is constructed, we compute the average length of the edges. Edges whose lengths are larger than the average are then removed from the MST. The MST thus trimmed may contain several disjoint sub-trees, each of which corresponds to a color cluster.

This MST-based clustering works well when the color of each character is clearly distinguishable from the background. The method does not do well at handling text involving large numbers of colors.

(a)

(b)

Figure 1. (a) Anti-aliased text in a GIF image. (b) Same image rendered using random colors.



(a) (b) (c)

Figure 2. (a) A letter 'L'. (b) Clustering based on RGB distance. (c) Clustering based on combined distance.

As mentioned earlier, text in Web images is often anti-aliased. This can introduce numerous border pixels with similar but not quite identical colors. Figure 1(a) is one such example. There are approximately 230 colors in the original version this image, of which 130 are different hues of blue used for the text. This effect is more apparent in Figure 1(b), where a different random color has been assigned to each of the original colors in Figure 1(a). (By necessity, the images reproduced in this paper are black-and-white. Copies of the original color images can be obtained by contacting the authors.)

In the clustering process, this gradual change of color produces a chain-like MST linking colors from the foreground (dark blue) to the background (white). The MST-based clustering method may not properly break such a chain-like tree structure, resulting in either a failure to separate the background from the foreground text, or fragmentation of the characters.

When dealing with textures, the MST-based approach may fail because it measures similarity using only RGB components. Figure 2 illustrates the problem. The letter 'L' in the image is rendered with a "checkerboard-like" texture. The actual color pattern is replicated in Figure 3. The color values in RGB space are: $A = (0, 0, 0)$, $B = (100, 100, 100)$, and $C = (175, 175, 175)$. To a human observer, it is obvious that colors A and B together form the letter 'L' and that color C is the background. However, in terms of RGB distance, color B is closer to the background color C (RGB distance $= 75\sqrt{3}$) than to color A (RGB distance $= 100\sqrt{3}$). Thus, in a color clustering scheme using pure RGB distance, colors B and C will be clustered together.

3.1.2. Clustering based on combined RGB/spatial distance

To address the issues just raised, we observe that colors that are spatially close to each other tend to belong to the same object. In order to improve our clustering performance, we incorporate a *local* clustering process prior to application of the MST method. This new procedure takes the spatial distribution of colors into consideration when grouping similar colors.

C	C	A	A	B	C	C	C
C	C	B	A	B	C	C	C
C	C	A	A	A	C	C	C
C	C	B	B	A	C	C	C
C	C	B	A	B	C	C	C
C	C	A	B	A	B	A	C
C	C	A	B	B	A	B	C
C	C	A	B	A	B	A	C

Figure 3. Symbolic color map for the letter 'L' of Figure 2(a).

First, we introduce a measure for the "spatial proximity" of colors: let $d_p(X)$ denote the distance between a given pixel p and the closest pixel with color X. For example, if we let p be the pixel at the fourth column and first row in Figure 3 and X be color B, then $d_p(B) = 1$. The closest pixel of color B is at the fifth column and first row, one pixel to the right of p. If we let X be color C, then $d_p(C) = 2$ since the closest pixels of color C are at the second and fifth columns of the first row. Denote by N an $m \times m$ neighborhood in the image, and by $c(p)$ the color of pixel p. Let set $\mathcal{P}_N(X)$ represent all pixels of color X in the neighborhood N:

$$\mathcal{P}_N(X) = \{p \in N \mid c(p) = X\}$$

Then the spatial distance from color X to color Y in the neighborhood of N is calculated as follows:

$$D^s_N(X,Y) = \frac{1}{\#\mathcal{P}_N(X)} \sum_{p \in \mathcal{P}_N(X)} d_p(Y) \tag{1}$$

where $\#\mathcal{P}_N(X)$ is the number of elements of $\mathcal{P}_N(X)$. Equation 1 is the average of the distances from every pixel of color X to the nearest pixel of color Y.

The distance defined in Equation 1 is not symmetric, *i.e.*, $D^s_N(X,Y)$ and $D^s_N(Y,X)$ are not necessarily equal. However, if pixels of two colors X and Y are from the same pattern and thus spatially interwoven, both $D^s_N(X,Y)$ and $D^s_N(Y,X)$ will be small. To make the distance symmetric, we define:

$$\overline{D}^s_N(X,Y) = \frac{D^s_N(X,Y) + D^s_N(Y,X)}{2}$$

Let $D^c(X,Y)$ be the RGB distance between colors X and Y. The combined RGB/spatial distance between colors X and Y in neighborhood N is defined as the following function of their RGB and spatial distances:

$$D_N(X,Y) = D^c(X,Y) \times \overline{D}^s_N(X,Y) \tag{2}$$

Continuing the example from Figure 3:

$$\begin{array}{lll} D^s_N(A,B) = 1.00 & D^s_N(B,A) = 1.00 & D^s_N(A,C) = 1.46 \\ D^s_N(C,A) = 1.77 & D^s_N(B,C) = 1.59 & D^s_N(C,B) = 2.12 \end{array}$$

$$\begin{array}{lll} \overline{D}^s_N(A,B) = 1.0 & \overline{D}^s_N(B,C) = 1.86 & \overline{D}^s_N(A,C) = 1.62 \\ D_N(A,B) = 100\sqrt{3} & D_N(B,C) = 140\sqrt{3} & D_N(A,C) = 122\sqrt{3} \end{array}$$

We observe that the new, combined distance between A and B is smaller than the combined distances between either A and C or B and C. If we were to cluster the colors in Figure 3 using the distance defined in Equation 2, colors A and B would be grouped together (see Figure 2(c)).

In applying the local clustering scheme to actual images, we first divide the image into non-overlapping $m \times m$ blocks and process each block independently. The parameter m is chosen such that each region is roughly bi-tonal ($m = 8$ in the current implementation). This allows us to make the assumption that there will likely be only two

(a)

(b)

Figure 4. Examples of characters touching in GIF images.

primary clusters for pixels in the region: one corresponding to the foreground color and the other to the background color. The clustering algorithm is a bottom-up approach that outputs one, two, or three color clusters depending on the input block. The three-cluster situation may arise in cases where the region contains a foreground object (*e.g.*, a text fragment), its shadow, and the background.

The clustering process first uses a well-known nearest-neighbor technique to group the pixels in a block into three clusters. It then decides either to combine the clusters further or to stop clustering. This decision is based on comparing the remaining distances with a fixed threshold. If all the colors are extremely similar and distributed evenly throughout the block, the algorithm will output only one cluster.

3.2. Character Detection

Once the color space has been clustered, the next step is to find connected components belonging to each cluster and identify those which correspond to text. Here we use a classification process based on geometric features extracted from the connected components. These include size, aspect ratio, the presence of holes and branches, *etc.*[8]

To reduce the complexity of the analysis, the classification process assumes that a text component roughly corresponds to a single character. Unfortunately, the connected components that result from color clustering sometimes contain more than one character. Multi-character components may arise when text is underlined or characters touch (*e.g.*, due to anti-aliasing), or when the clustering process fails for some other reason. Figure 4 shows two examples where the single-character-component assumption is violated.

Our remedy is to introduce a segmentation step into the classification process. Notice that the segmentation problem here is different from traditional document analysis: in the latter case, the text is already identified and the objective of the segmentation is to isolate individual characters, whereas the goal here is to decide whether or not a connected component is even text to begin with. Indiscriminately breaking components into smaller pieces will result in an excessive number of false alarms since non-text objects can look like text when broken up.

The classification process is divided into three phases. In the first, we select candidate text-like components based on features that are relatively invariant to touching. These include size, stroke width, and white-to-black-area ratio. In the second phase, we single out those components from the first phase that have an "elongated" shape (*i.e.*, components whose width/height ratio exceeds a given threshold) and apply the segmentation procedure.

The segmentation process first determines the dominant color for each component – this is the color used by the majority of the pixels in the component. The dominant color of a character usually appears in the interior of its structure, while other colors are present around its edges. We use this knowledge to aid the segmentation process.

Two criteria are used for segmentation. The first is an analysis of the color distribution along scanlines. Currently, we assume that text is roughly horizontal. The segmentation computes a histogram of foreground pixels at each

Figure 5. Results of the character detection stage.

column and identifies potential break-points. A column position is designated as a potential break-point if it contains only a small number of foreground pixels, of which few are the dominant color.

The second criterion is an examination of the color correlation between adjacent scanlines. The segmentation procedure checks the consistency between pixels at adjacent columns. We expect columns that fall between characters to exhibit a low color correlation. Consider, for example, Figure 4(b); if we take any two adjacent columns within the letter 's' or the letter 'o', we see that the number of rows whose pixel-pairs are either both foreground or both background is greater than the number of rows where only one pixel is foreground. On the other hand, at the right edge of the 's', the number of rows whose adjacent pixels have the same type is much smaller than the number of rows whose adjacent pixels have different types. This is where we place break-points. To avoid over-segmentation, we further restrict that the minimum distance between break-points be half the height of the connected component.

Finally, all of the components from the above two phases are subjected to an additional filtering process. This makes use of more sophisticated features such as the number of holes and the number of upward/downward ends to further weed out non-text components.

Figure 5 shows the character detection results for the GIF image from Figure 1(a). Note that almost all of the pixels corresponding to characters have been identified, along with a small number of false alarms.

3.3. Layout Analysis

The layout analysis we use can be viewed as a form of post-processing. Character detection based solely on geometric features cannot be made 100% reliable. A wide range of non-text objects may resemble characters and thus be erroneously classified. The post-processing step attempts to eliminate these false alarms by using additional heuristics based on layout criteria typical of text.

In post-processing, we consider each color cluster independently. We first group characters to find words or text lines. Characters that are close in the horizontal direction and whose top and bottom positions are roughly aligned are said to belong to the same word. A measure of "goodness," which we call the *saliency* of the text, is then calculated for each word. The saliency of a word is computed as follows:

$$S = \frac{c}{1 + \sigma_h/\bar{h} + \sigma_b/\bar{h}} \tag{3}$$

where σ_h and σ_b are the standard deviations of the heights and the baselines of the characters in the word, $\bar{h}$ is the average height, and c is the *character factor* which is 0, 0.5, or 1 for a word containing one, two, or more than two characters, respectively.

Saliency is a simple heuristic measure that reflects the degree of height and positional alignment of the characters in a word. Clearly, the saliency measure gives higher scores for words containing larger numbers of characters and having more regular typography.

Post-processing eliminates those words whose saliency measures are lower than a given threshold. In addition, we apply a set of procedures to handle certain special cases:

Nested components Occasionally, the pixels inside the holes of characters such as the letters 'o' and 'b' satisfy the criteria for being characters themselves. This results in false words detected inside of actual words. An example of this can be seen in Figure 5, where parts of the white background are classified as characters. These false alarms tend to align fairly well and can often be grouped into words. However, they usually have lower saliency measures than the words they lie within because their alignment is not quite as good. Therefore, we eliminate all words that lie within another word whose saliency score is higher.

Table 1. Test database statistics (482 GIF images).

	Minimum	Maximum	Average
Number of characters	2	496	34
Number of colors	2	256	61
Width (pixels)	20	667	285
Height (pixels)	8	799	75

Shadow text Text in GIF images sometimes includes a shadow effect. This may result in the detection of partially overlapping words, one corresponding to the actual word, the other to its shadow (see Figure 6). We note that a word's shadow usually contains fewer pixels than the word itself, and can be eliminated on this basis.

Figure 6. An example of text with a shadow effect.

Figure 7 shows the final output of the algorithm for the input image from Figure 1(a). Observe that the false alarms seen in Figure 5 have been eliminated.

Figure 7. Final output of the algorithm for the example from Figure 1(a).

4. EXPERIMENTAL RESULTS

We tested our text extraction algorithm on 482 GIF images chosen from real WWW pages. All of these images contained some amount of text. Table 1 summarizes several statistics for the test database.

Our current method for evaluating performance is to manually transcribe the text from each image before and after running our algorithm. In either case, the decision as to whether a given character is "readable" is subjective. Generally, if enough pixels are missed (or added) that a character's shape is visibly affected, the character is regarded as unreadable (see, for example, the letter 'S' in "HOTELS" in Figure 7). Although in many cases a human could still read the text in question, OCR processes tend to be much more sensitive; hence, we err on the side of being conservative. We then apply an approximate string matching procedure to count the number of correctly detected characters. False alarms are currently ignored.

The average character detection rate for the set of 482 GIF's was found to be 68.3%. While this is still rather low, it is a significant improvement over our earlier results (a 47% average detection rate for a similar database). The new algorithm not only improved overall performance, it also reduced the percentage of "catastrophic" failures (see the chart on the left in Figure 8). Figure 9 shows some of the sample images and their corresponding detection results.

The test images used in the experiment above were selected randomly from WWW pages without regard to several of the assumptions made earlier in the paper. As a consequence, some of them contain non-homogeneously-colored text and/or non-horizontal layouts. In order to see how well our algorithm performs when these premises hold, we manually inspected the test set and separated it into two groups: the first consisted of 315 images which met the

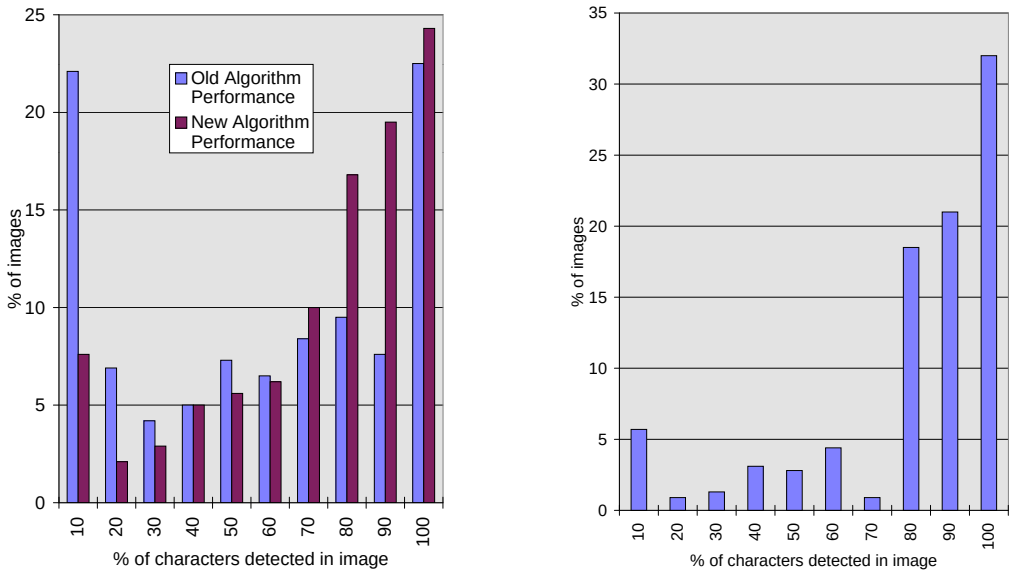


Figure 8. Performance of the text detection algorithm.

assumptions, and the second the remaining 167 images. The chart on the right side of Figure 8 shows the results when running our algorithm on the images from the first group. The average character detection rate in this case was 78.8%.

Figure 10 presents some difficult examples from the second group. These include non-linear text layout, embossed text, non-homogeneously-colored text, and stylized text (a logo). We break these into three distinct cases:

1. Skewed or wrapped text. A few instances of severe skew angles or wrapped text were found in the test set. Most images in this category contained text skewed just slightly off the horizontal.
2. Non-homogeneously-colored text. Low detection rates for text in this class demonstrate the importance of the homogeneity assumption.
3. Extremely small text. Some text in Web GIF images is very small (3-5 pixels tall). This results in miss-detections since the algorithm uses a size threshold to filter out noise in the image.

Table 2 shows a performance break-down for the images in each of these classes.

Table 2. Performance on various difficult test cases.

	Number of Images	Average Character Detection Rate
Case 1 (skewed)	18	70.5%
Case 2 (non-homogeneous)	12	36.9%
Case 3 (extremely small)	21	44.3%

5. CONCLUSIONS

In this paper, we have described a method for locating and extracting text from color GIF images. Our technique employs a color clustering algorithm that attempts to group colors from the same object together based on both RGB

and spatial proximity. Geometric as well text layout features are then used to distinguish text-like from non-text-like components. Experimental results show that our method is promising.

Clearly, there is also much room for improvement. For example, more sophisticated measures of text "goodness" need to be developed. The current algorithm uses a measure based solely on character alignment (the variation in the heights of components). Since text detection is intended to be followed by recognition, a measure that reflects the quality of the character images would be useful.

In order to recover broken or missing characters, an iterative process could be introduced into the algorithm. After completing the final stage of text detection, a refined clustering procedure might be applied to areas where text has been located to ensure the integrity of the characters.

Finally, better methods are needed for performance evaluation. Ideally, OCR would be run on the results of text detection and the performance of the entire end-to-end system measured. However, this is in itself a challenging problem because of the wide range of fonts used for WWW text and its highly stylized layout. We are currently exploring evaluation criteria that operate at the image level with the hopes of developing a more robust scheme for quantifying the performance of text detection algorithms.

ACKNOWLEDGEMENTS

The trademarks depicted in this paper are the properties of their respective owners.

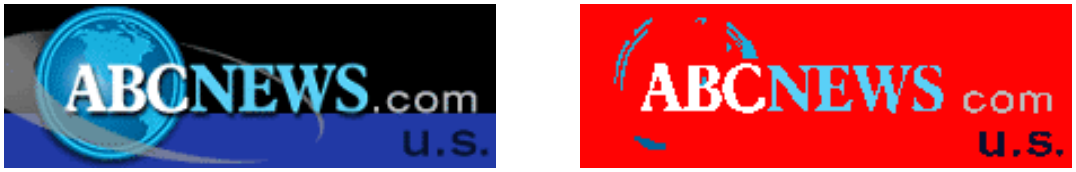
REFERENCES

1. D. Lopresti and J. Y. Zhou, "Document analysis and the World Wide Web," in *Proceedings of the Workshop on Document Analysis Systems*, J. Hull and S. Taylor, eds., pp. 417–424, (Marven, Pennsylvania), October 1996.
2. L. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **10**, pp. 910–918, November 1988.
3. V. Wu, R. Manmatha, and E. Riseman, "Finding text in images," in *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3–12, (Philadelphia, Pennsylvania, USA), July 1997.
4. B. Yu, A. Jain, and M. Mohiuddin, "Address block location on complex mail pieces," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, pp. 897–901, (Ulm, Germany), August 1997.
5. Y. Ariki and T. Teranishi, "Indexing and classification of TV news articles based on telop recognition," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, pp. 422–427, (Ulm, Germany), August 1997.
6. J. Y. Zhou, D. Lopresti, and Z. Lei, "OCR for World Wide Web images," in *Proceedings of the IS&T/SPIE International Symposium on Electronic Imaging*, pp. 58–65, (San Jose, California), Febuary 1997.
7. Y. Zhong, K. Karu, and A. Jain, "Locating text in complex color images," *Pattern Recognition* **28**(10), pp. 1523–1535, 1995.
8. J. Zhou and D. Lopresti, "Extracting text from WWW images," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, pp. 248–252, (Ulm, Germany), August 1997.

(a)

(b)

(c)



(d)

Figure 9. Examples of GIF images and the results of text detection.

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

Figure 10. Images the current algorithm cannot handle well.