Finding Text In Images [1]

Victor Wu, R. Manmatha, Edward M. Riseman[2]
Center For Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610
{vwu, manmatha}@cs.umass.edu

Abstract

There are many applications in which the automatic detection and recognition of text embedded in images is useful. These applications include multimedia systems, digital libraries, and Geographical Information Systems. When machine generated text is printed against clean backgrounds, it can be converted to a computer readble form (ASCII) using current Optical Character Recognition (OCR) technology. However, text is often printed against shaded or textured backgrounds or is embedded in images. Examples include maps, advertisements, photographs, videos and stock certificates. Current OCR and other document segmentation and recognition technologies cannot handle these situations well.

In this paper, a system that automatically detects and extracts text in images is proposed. This system consists of four phases. First, by treating text as a distinctive texture, a texture segmentation scheme is used to focus attention on regions where it may occur. Second, strokes are extracted from the segmented text regions. Using reasonable heuristics on text strings, such as height similarity, spacing and alignment, the extracted strokes are then processed to form tight rectangular bounding boxes around the corresponding text strings. To detect text over a wide range of font sizes, the above steps are first applied to a pyramid of images generated from the input image, and then the boxes formed at each resolution of the pyramid are fused at the original resolution. Third, an algorithm which cleans up the background and binarizes the detected text is applied to extract the text from the regions enclosed by the bounding boxes in the input image . Finally, text bounding boxes are refined (re-generated) by using the extracted items as strokes. These new boxes usually bound text strings better. The clean-up and binarization process is then carried out on the regions in the input image bounded by the boxes to extract cleaner text. The extracted text can then be passed through a commercial OCR engine for recognition if the text is of an OCR-recognizable font. Experimental results show that the algorithms work well on images from a wide variety of sources, including newspapers, magazines, printed advertisements, photographs, digitized video frames, and checks. The system is also stable and robust—the system parameters work for all the experiments.

Keywords — text reading system, character recognition, multimedia indexing, digital libraries, text detection, text extraction, texture segmentation, filters, focus of attention, hierarchical processing, binarization, histogram-based thresholding, background removal, morphological processing, connected-components analysis.

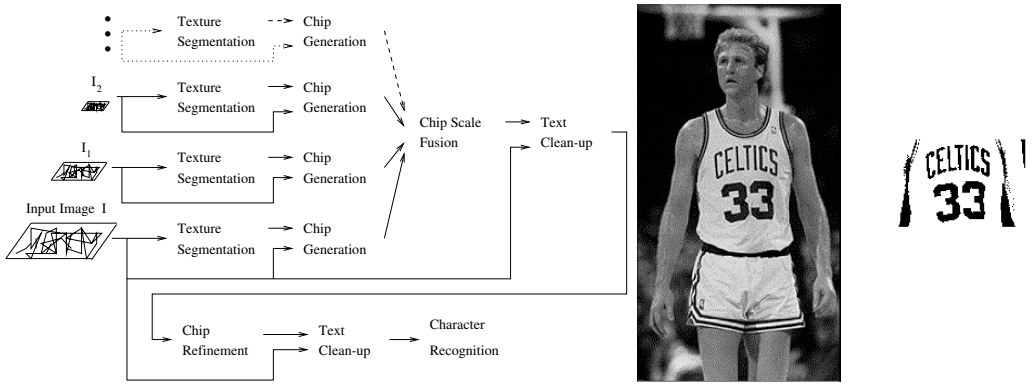
1 Introduction

Most of the information available today is either on paper or in the form of still photographs and videos. To build digital libraries, this large volume of information needs to be digitized into images and the text converted to ASCII for storage, retrieval, and easy manipulation. Current OCR technology [2, 8] is largely restricted to finding text printed against clean backgrounds, and cannot handle text printed against shaded or textured backgrounds, and/or embedded in images. There is thus a need for systems which extract and recognize text from general backgrounds.

More sophisticated text reading systems usually employ document analysis (page segmentation) schemes to identify text regions before applying OCR so that the OCR engine does not spend time trying to interpret

[1]This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, in part by the United States Patent and Trademark Office and Defense Advanced Research Projects Agency/ITO under ARPA order number D468, issued by ESC/AXS contract number F19628-95-C-0235 and in part by NSF Multimedia CDA-9502639. Any opinions, findings and conclusions or recommendations expressed in this material are the author(s) and do not necessarily reflect those of the sponsors.

[2]E. M. Riseman is with the Computer Vision laboratory.



(a) (b) (c)

Figure 1: The system, an input, and extracted text. (a) The top level components of the text detection and extraction system. The pyramid of the input image is shown as I, I_1, I_2 ...; (b) An example input image; (c) Output of the system result before the Character Recognition module.

non-text items. However, most such schemes require clean binary input [3, 15, 16, 17]; some assume specific document layouts such as newspapers [6] and technical journals [9]; others utilize domain-specific knowledge such as mail address blocks [12] or configurations of chess games [1].

In this paper, a new end-to-end system is proposed which automatically extracts and recognizes text in images. The system takes both greyscale and binary images as input[3]. It detects text strings in the image and puts rectangular bounding boxes around them. These bounded regions in the input images are then cleaned up and binarized so that the text stands out. The extracted text can then be recognized by a commercial OCR system, if the text is of an OCR-readable font.

1.1 Motivation

There are many situations where it would be useful to detect and read text embedded in images. A few examples are listed below:

1. Text found in images or videos can be used to annotate and index those materials. For example, video sequences of events such as a basketball game can be annotated and indexed by extracting a player's number, name and the name of the team that appear on the player's uniform (Figure 1(b, c)). In contrast, image indexing based on image content, such as the shape of an object, is difficult and computationally expensive to do.

2. Systems which automatically register stock certificates and other financial documents by reading specific text information in the documents are in demand. This is because manual registration of the large volume of documents generated by daily trading requires tremendous manpower.

3. Maps need to be stored electronically in building a Geographical Information System (GIS). One approach is to scan the maps first and then extract the lines, text, and symbols. The lines are then stored in a vector representation and the text and symbols in symbolic forms. The electronic representation of a map makes updating, scaling, and retrieval much easier.

1.2 Prior Work

OCR technology ([2],[8]) has been used to convert the text in scanned paper documents into ASCII symbols. However, current commercial OCR systems do not work well if text is printed against shaded or hatched backgrounds, often found in documents such as photographs, maps, monetary documents, engineering drawings and commercial advertisements. Furthermore, these documents are usually scanned in greyscale or color to preserve details of the graphics and pictures which often exist along with the text. For current OCR systems, these scanned images need to be binarized before actual character segmentation and recognition can be done. A typical OCR system does the binarization to separate text from the background by *global thresholding* ([4, 11]). Unfortunately, global thresholding is usually not possible for complicated images, as noted by many researchers ([11], [14]). Consequently, current OCR systems work poorly in these cases.

[3] A binary image can be processed by first scaling it so that its intensity ranges from 0 to 255

One solution to the global thresholding problem is to use different thresholds for different local regions (*adaptive thresholding*) [6]. Trier and Taxt [14] report an evaluation of eleven local adaptive thresholding schemes.

Many document segmentation methods have been proposed in the literature. Some of these methods are top-down approaches, some are bottom-up schemes, and others are based on texture segmentation schemes in computer vision. Classic top-down techniques are based on the run length smoothing (RLS) algorithm [15, 17] to smooth the image first, then, horizontal and vertical projection profiles [16] are commonly used to cut the page into smaller blocks such as columns and paragraphs [9, 13, 16]. Bottom-up methods work by grouping small components (starting with pixels as connected components) into successively larger components until all blocks are found on the page [3, 10]. The third category of document segmentation methods treat text as a type of texture and hence use texture segmentation algorithms to detect text [5].

Many of these algorithms have limitations on their use. The top-down and bottom-up approaches require the input image to be binary. The projection profile based schemes work if the page has a Manhattan layout: that is, there is only one skew angle and the page can be segmented by horizontal and vertical cuts. Although the texture segmentation scheme in [5] can in principle be applied to greyscale images, it was only used on binary document images, and in addition, the binarization problem was not addressed. In summary, although a considerable amount of work has been done on different aspects of document analysis and understanding, few working systems have been reported that can read text from document pages with both structured and non-structured layouts. The system presented in this paper is our contribution to filling the gap in this area of research and development, and to constructing a complete automatic text reading system.

1.3 Our Approach

The goal here is to build an end-to-end automatic text extraction system which accepts a wide range of images as input, detects text in the input images, and then binarizes and cleans up the detected text so that it can be fed into a commercial OCR for character recognition.

The system takes advantage of the distinctive characteristics of text which make it stand out from other image material. For example, by looking at the comic page of a newspaper a few feet away, one can probably tell quickly where the text is without actually recognizing individual characters. Intuitively, text has the following distinguishing characteristics: (1) Text possesses a certain frequency and orientation information; (2) Text shows **spatial cohesion** — characters of the same text string are of similar heights, orientation and spacing.

The first characteristic suggests that text may be treated as a distinctive texture, and thus be segmented out using texture segmentation techniques. The first phase of the system, therefore, uses (**Texture Segmentation**(Figure 1(a))) to segment the text (Section 2). This algorithm is based on the standard multi-channel filtering techniques in texture segmentation [5, 7]. It should be pointed out that texture segmentation schemes are not sufficient for text detection and extraction if images more complicated than clean newspaper scans have to be dealt with. Nevertheless, the segmentation result can be used as a focus of attention for further processing called **Chip Generation** (section 3) of the system.

The basic idea for chip generation is to apply a set of appropriate heuristics to find text strings within/near the segmented regions. The heuristics are designed to reflect the second characteristic of text as listed above. The algorithm uses a bottom-up approach: significant edges form **strokes**; strokes are connected to form **chips** (regions) corresponding to text strings. The rectangular bounding boxes of the chips are used to indicate where the hypothesized (detected) text strings are.

The above text detection procedures work well for text over a certain range of font sizes. To detect text whose font size varies significantly, the notion of scale is used: a pyramid of images is created in which the original image is at the bottom level. An image at a higher level is obtained by reducing the image at the level below it by half in each dimension. Each image in the pyramid is then fed into the text detection subsystem separately. The output chip boxes are mapped back onto the original image and redundant boxes are eliminated (**Chip Scale Fusion**). As an example, to find text of fonts up to 160 pixels in height, a hierarchy of three levels is required (Figure 1(a)).

It will be shown that for each chip, a single threshold suffices to clean up and binarize the corresponding region in the input image so that the text stands out. A simple, effective histogram-based algorithm is proposed which finds the threshold value automatically for each text region. This algorithm is used for the **Text Clean-up** module in the system.

Non-text items might survive the previous processing and occur in the binarized output. Thus, a **Chip Refinement** phase is used in the system to filter them out. This is done by treating the extracted items (text and non-text) as strokes to re-generate chips using the same algorithms (with stronger constraints) in

the Chip Generation phase. The chips produced this time usually enclose the text strings better. The Chip Clean-up process is then applied to the new chips to obtain better binarization results since less irrelevant area (noise) is involved.

Figure 1(a) depicts the system described above. Experimental results have shown that the system works well with both machine generated fonts and some script fonts. Practically speaking, the font sizes do not matter. The system is also stable and robust—all the system parameters remain the same for all of the text images from a wide variety of sources including newspapers, magazines, printed advertisement, photographs, and checks. Notice that some of these documents have structured layout, some do not, and the system works well in either case. A detailed description of the experiments is presented in section 7.

2 The Texture Segmentation Module

As stated in section 1.3 text can be treated as a specific texture. Thus, one natural way to detect text is by using texture segmentation. A standard approach to texture segmentation is to first filter the image using a bank of linear filters, such as Gaussian derivatives ([7] or Gabor functions [5] followed by some non-linear transformation such as half-wave rectification, full-wave rectification, or a hyperbolic function $tanh(\alpha t)$. Then features are computed to form a feature vector for each pixel from the filtered images. These feature vectors are then classified to segment the textures into different classes.

In this paper, 9 filters are used to segment the texture. The filters are the 3 second order derivatives of Gaussians at three different scales $\sigma = (1, \sqrt{2}, 2)$. Each filter output is passed through the non-linear function $tanh(\alpha t)$ where $\alpha = 0.25$. A local energy estimate is computed using the outputs of the non-linearity. The result consists of 9 images where each pixel in one of these images represents the local energy due to a given filter. At each pixel, a feature vector can be constructed consisting of the energy estimates from the 9 images for that location. The "image" of feature vectors is clustered using a K means algorithm (with K = 3). One of the clusters is labelled as text.

Since text generally 'has a stronger response to the filters, while background areas with little intensity variation have nearly no response (i.e. have close to zero energy), the following cluster labeling scheme is used. The cluster whose center is closest to the origin of the feature vector space, $(0, 0, \ldots, 0)$, is labeled as background. The cluster whose center is furthest away from the background cluster center is labeled as text.

Figure 2(a) shows a portion of an original input image. Only part of the original input image is shown so that the details are more noticeable. This is a Stouffer's advertisement scanned at 300dpi. There is text on a clean dark background, text printed on Stouffer boxes, Stouffer's trademarks (in script), and a picture of the food. Figure 2(b) shows the pixel labels after the texture segmentation step. The dark area corresponds to the segmented "text" regions, and the white area corresponds to the background. The grey area is where the pixels have some energy, but not enough to be text pixels.

As shown in Figure 2(b), the text regions may be broken or have holes. Thus, as the last step of the segmentation phase of the system, a morphological closure operation is carried out on the segmented text regions. Figure 2(c) shows the result of this operation carried out on the text regions shown in 2(b).

3 The Chip Generation Phase

In practice, text may occur in images with complex backgrounds and texture patterns, such as foliage, windows, grass etc. In other words, some non-text patterns may pass the filters and initially be misclassified as text, as shown in Figure 2(c). Furthermore, segmentation accuracy at texture boundaries is a well-known and difficult problem in texture segmentation. Consequently, it is often the case that text regions are connected to other regions which do not correspond to text, or one text string might be connected to another text string of a different size or intensity. This might cause problems for later processing. For example, if two text strings with significantly different intensity levels are joined into one region, one intensity threshold might not separate both text strings from the background.

Therefore, heuristics need to be employed to refine the segmentation results. Generally speaking, the segmentation process usually finds text regions while excluding most of the non-text ones (experimental finding). These regions can be used to direct further processing (**focus of attention**). Furthermore, since text is intended to be readable, there is usually a significant contrast between it and the background; thus contrast can be utilized finding text. Also, it is usually the case that characters in the same word/phrase/sentence are of the same font and have similar heights and inter-character spaces (unless it is in some kind of decorative font style). Finally, it is obvious that characters in a horizontal text string are horizontally aligned[4].

[4] In this paper, the focus will be on finding horizontal, linear text strings only. The issue of finding text strings of any orientation will be addressed in future work.

The basic idea for the **Chip Generation** phase is to use the segmented regions as the focus of attention, and then apply a set of appropriate constraints to find text strings within the segmented regions. The algorithm uses a bottom-up approach: significant edges form strokes; strokes are connected to form chips corresponding to text strings. The rectangular bounding boxes of the chips are used to indicate where the hypothesized (detected) text strings are. Conceptually, Chip Generation consists of the following main steps which are applied in the order given: (1)**Stroke Generation** — strokes are generated from significant edges; (2)**Stroke Filtering** — strokes which are unlikely to belong to any horizontal text string are eliminated; (3)**Stroke Aggregation** — strokes which are likely to belong to the same text string are connected to form chips; (4)**Chip Filtering** — chips which are unlikely to correspond to horizontal text strings are eliminated; (5)**Chip Extension** — filtered chip are treated as strokes and aggregated again to form chips which cover the text strings more completely. A detailed discussion of these steps is presented in the following corresponding subsections.

3.1 Stroke Generation

Text must have significant contrast in order to be readable. Thus, the edges of characters can be expected to have significant contrast. Therefore, the input image is first convolved with a second-order Gaussian derivative in the horizontal direction and then thresholded to find the significant edges. Connected components computation is then emplyed to group edges into **strokes**.

Empirically, $\sigma = 1$ was found to be a reasonable choice. Also, the threshold value was set to 10 for all the experiments. Figure 2(d) shows the strokes for Figure 2(a).

3.2 Stroke Filtering

As one can clearly see in Figure 2(d), strokes are found at regular intervals in the regions where text is present. However, non-text strokes will also be extracted where there are significant horizontal intensity changes in a scene.

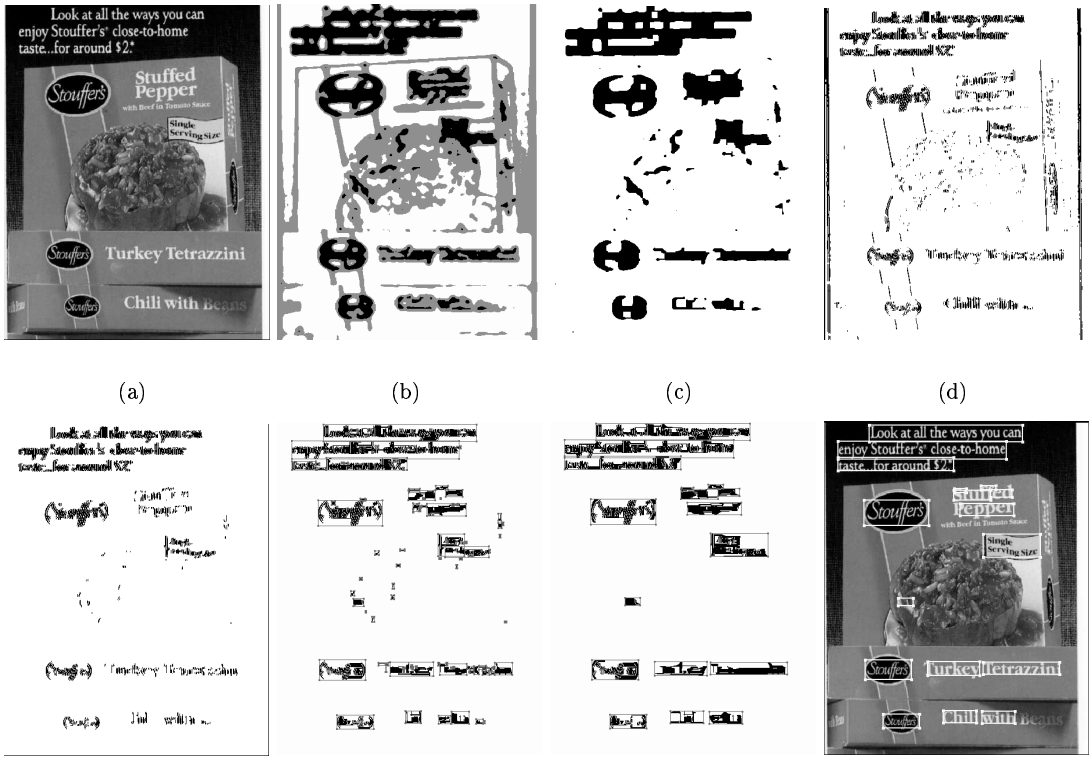


Figure 2: Results of Texture Segmentation and Chip Generation. (a) Portion of an input image; (b) Output of the clustering stage. Dark regions are labeled as "text" regions; (c) The Text regions after the morphological closure operation; (d) Strokes produced by performing the Stroke Generation procedure on a; (e) Filtered strokes; (f) Chips produced by applying Stroke Aggregation on strokes in e; (g) Chips after the Chip Filtering and Extension processes; (h) Chips in g mapped to the input image.

The purpose of Stroke Filtering is to eliminate the false positive strokes by using heuristics which take into account the fact that neigboring characters in the same text string usually have similar heights and are horizontally aligned. It is reasonable to assume that the similarity of character heights causes the heights of the corresponding stokes to be similar. Furthermore, since the focus is on finding text strings, a text stroke should have similar strokes nearby which belong to the same text string. These heuristics can be described using **connectability** which is defined as:

Definition 1 *Let A and B be strokes. A and B are* **connectable** *if they are of similar height and horizontally aligned, and there is a* **path** *between A and B (a horizontal sequence of consecutive pixels in the segmented region which connects A and B).*

Here, two strokes are considered to be of similar height if the height of a shorter stroke is at least 40% of the height of a taller one. To determine the horizontal alignment, strokes are projected onto the Y-axis. If the overlap of the projections of two strokes is at least 50% of the shorter stroke, they are considered to be horizontally aligned.

Given the above definition, the criterion used for stroke filtering can be simply stated as follows:

- a stroke is eliminated if one of the following conditions are true: (1) it does not sufficiently overlap with the segmented text regions produced by the texture segmentation phase; (2) it has no connectable stroke.

Condition 1 says the strokes are expected to overlap the segmented regions. Since the text segmentation is often not perfect, one cannot expect total overlap. A minimum of 30% overlap rate worked well for all the test images. Condition 2 says that if there is no path that leads to some connectable stroke(s), it is probably an isolated stroke or line which does not belong to any text string.

Figure 2(e) shows the result of applying this procedure to the strokes in figure 2(d). Notice that most of the text is still present while more of the background has been eliminated.

3.3 Stroke Aggregation

An important function for the Chip Generation phase is to generate chips that correspond to text strings. This is done by aggregating strokes belonging to the same text string.

Since characters which belongs to the same text string are expected to be of similar height and horizontally aligned, the concept of connectability can be used to aggregate the strokes. In addition, it is clear that strokes corresponding to the same text string should be close to each other. The width of a character and the spacing between adjacent characters in a text string are related to the heights of the characters. Thus, it is reasonable to measure the spacing between adjacent strokes as a function of the heights of the strokes. By empirical observation, the spacing between the characters and words of a text string is usually less than 3 times the height of the tallest character, and so is the width of a character in most fonts. Therefore, for all of the experiments, the following criterion is used to generate chips:

- two strokes, A and B, are connected if they are connectable and there is a path between A and B whose length is less than 3 times the height of the taller stroke.

Figure 2(f) shows the result of applying the Chip Generation procedure to the strokes in figure 2(e). Notice that most of the isolated strokes are connected into chips which partially or completely cover text strings. The chips are shown with their bounding boxes to make it easier to see.

3.4 Chip Filtering

Some non-text strokes may also pass the Stroke Filtering process, and therefore form false positive chips requiring further filtering. This might happen, for example, when there are periodically occurring lines or patterns in the image.

Text strings are expected to have a certain height in order to be reliably recognized by an OCR system. Thus, one choice is to filter the chips by their heights. Furthermore, since we are interested in text strings, not just isolated characters, the width of a chip is also used to filter out text. Lastly, for horizontally aligned text strings, their aspect ratio (width/height) is usually large. Therefore, chips are filtered using the following constraints on their minimum bounding boxes:

- a chip is eliminated if the width of its box is less than cw_τ; or the height of its box is less than ch_τ or the aspect ratio of its box is larger than $ratio_\tau$

It is usually difficult even for a human to read the text when its height is less than 7 pixels, thus 7 has been used for ch_τ for the experiments. A horizontal text string is usually longer horizontally, hence setting cw_τ to at least twice the minimum height seems reasonable. Thus, in all of our experiments, $cw_\tau = 15$ and $ch_\tau = 7$ were used. Normally, the width of a text string should be larger than its height. But in some fonts, the height of a character is larger than its width. Therefore, $ch_\tau = 1.1$ is used here, attempting to cover that case to some extent.

3.5 Chip Extension

It is expected that some strokes only cover fragments of the corresponding characters. Therefore, these strokes might violate the constraints used for stroke filtering, and hence be eliminated. Consequently, some of the chips generated so far may only cover part of the corresponding text strings.

Fortunately, this fragmentation problem can usually be corrected. Notice that the chips corresponding to the same text stroke are still horizontally aligned and of similar height. Thus, by treating the chips as strokes, the Stroke Aggregation procedure can be applied again to aggregate the chips into larger chips. This is exactly what the Chip Extension step does. As a result, more words are completely covered by the extended chips.

Figure 2(g) shows the result of applying the Chip Filtering and Extension steps to the chips in Figure 2(f). The rectangular chip bounding boxes are mapped back onto the input image to indicate detected text as shown in Figure 2(h).

4 A Solution to the Scale Problem

The three frequency channels used in the segmentation process work well to cover text over a certain range of font sizes. Text from larger font sizes is either missed or fragmented. This is called the (**scale problem**). Intuitively, the larger the font size of the text, the lower the frequency it possesses. Thus, when text font size gets too large, its frequency falls outside the three channels selected in section 2.

We propose a pyramid approach to the scale problem: form a pyramid of input images and process each image in the pyramid using the standard channels ($\sigma = 1, \sqrt{2}, 2$) as described in the previous sections (see Figure 1(a)). At the bottom of the pyramid is the original image; the image at each level (other than the bottom) is obtained by reducing the image one level below by half in both dimensions. Text of smaller font sizes can be detected using the images lower in the pyramid as shown in Figure 3(b) while text of large font sizes is found using images higher in the pyramid as shown in Figure 3(d). The bounding boxes of detected text regions at each level are mapped back to the original input image (bottom level) as shown in Figure 3(e).

4.1 Chip Scale Fusion

Sometimes a text string or a part of it responds to more than one band of the frequency channels, and hence forms overlapping chips at different levels (Figure 3(e)). For example, text with large fonts may partially responds to higher frequency channels, hence forming fragmented boxes which covers parts of the string at a lower level. At the same time, it strongly responds at some lower frequency channels which are more compatible, hence forming a more complete box to cover the entire string at higher level(s) (Figure 3(c,d)). After the chips produced at different levels are mapped back onto the original image (bottom level), the fragmented chips will be covered in full or in part by larger boxes, as shown in Figure 3(e). Thus, it is desirable to eliminate these scale-redundant chips and keep the chips which overlap more with the text strings.

When most of a chip B overlaps with another chip A, it is likely that the text covered by B is also covered by the other chip A, especially when more than 85% of chip B overlaps with chip A (empirical finding). Another situation is that only a small portion of B overlaps with A. In this case, if A is significantly larger than B, B is also likely a scale-redundant chip. It is especially the case if at least 50% of chip B overlaps with chip A and chip A is at least 10 times bigger than B (empirical finding). Thus, the following straightforward procedure is used for the scale fusion for all of the experiments:

- for any pair of chips A and B assuming $Area(A) \geq Area(B)$, chip B is eliminated if one of the following holds: (1) 85% of B is covered by A; (2) 50% of B is covered by A and the area of B is less than 10% of the area of A.

Figure 3(f) is the result of the above scale fusion procedure applied to chips in Figure 3(e). Notice that text from a large range of font sizes is detected.

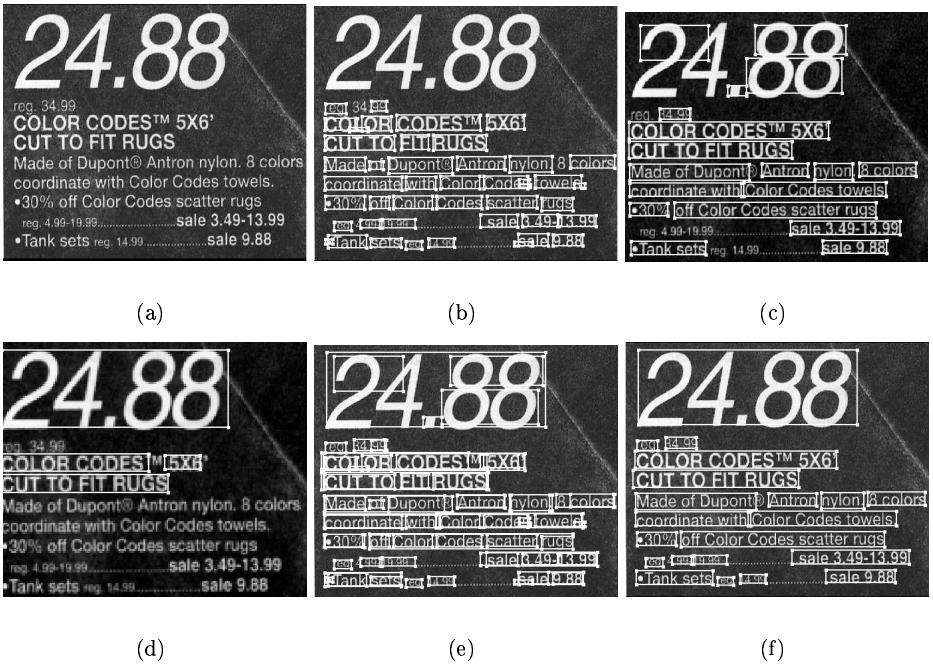


Figure 3: The scale problem and its solution. (a) Original input image; (b) Chips generated for the input image at its full resolution; (c) Chips generated at half the resolution of the input image; (d) Chips generated at $\frac{1}{4}$ of the resolution of the input image; (e) Chips generated at all three levels a, b, and c, mapped back to a; (f) Scale-redundant chips are removed.

5 Text on Complex Backgrounds

The previous sections describe a system which detects text in images and puts boxes around detected text strings in the input image. The text strings may be printed against complex image backgrounds (see for example Figure 4a). Current OCR systems cannot handle such text because they require text to be printed against a clean background. In addition, OCR systems require that the text must be binarized before they can process it.

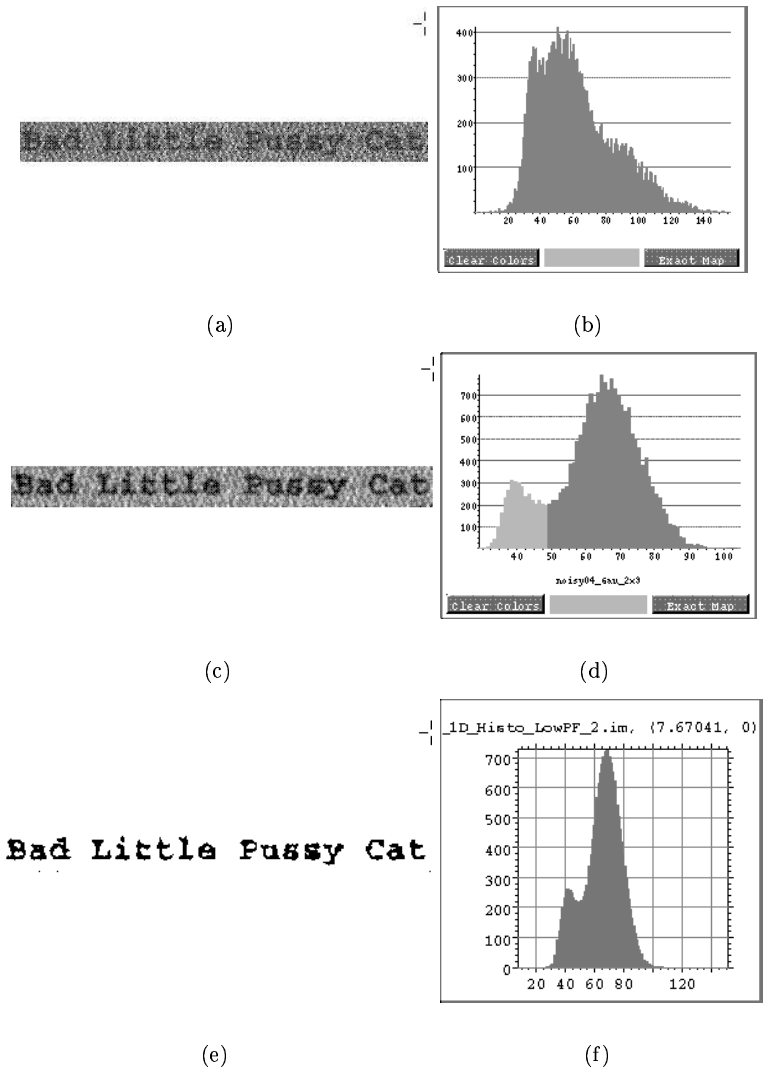
Our goal here is to find a simple, robust algorithm which will remove the background while preserving the text, eliminate noise and also produce a binary image which current OCR systems can handle. Local thresholding is a good way to do background and noise removal while simultaneously binarizing the image. The text chips produced usually contain text strings whose characters all roughly have the same intensity. These chips are, therefore, good candidates for local thresholding.

The following algorithm for background removal and binarization is proposed in this paper. Smooth the text chip generated by the system. In Figure 4, the text chip in Figure 4(a) is smoothed to produce Figure 4(c). The smoothing preserves the low frequency text while eliminating some of the high frequency shading and noise. Another way of looking at it is to notice that the smoothing blends the background into grey while leaving the text black (see Figure 4(c)).

Text is usually either darker than the background or lighter than it. Without loss of generality we assume it is darker. Compute the intensity histogram of the smoothed chip(Figure 4d). The black text corresponds to the portion of the histogram to the left of the major valley. The valley can be automatically detected by first smoothing the histogram in Figure 4(d) to produce the histogram in Figure 4(f). The histogram smoothing eliminates the small noise peaks in Figure 4d. The text in Figure 4(c) can be automatically detected by picking a threshold at the first valley counted from the left side of the histogram in Figure 4(f). The resulting thresholded output is shown in Figure 4(e) and has been successfully recognized using an OCR engine.

Since the current system does not know whether dark text or light text is in a text chip, one output is produced for each case for all the text chips.

This clean up and binarization procedure has been successfully used on many images (see for example the Expperiments section). Although simple, it is highly effective and we are not aware of such a procedure having been used before.



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

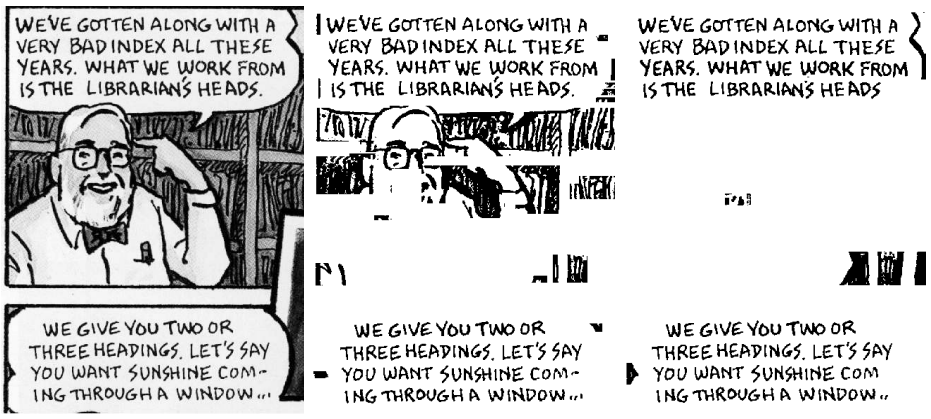
Figure 4: The Text Clean-up process. (a) Original text chip; (b) Histogram of a; (c) Smoothed version of a; (d) Histogram of c; (e) The binarization result by thresholding c using a value in the valley of f; (f) Smoothed version of d.

6 The Text Refinement

Experiments show that the text detection phase is able to locate text strings in regular fonts, and some even from script fonts or trademarks. However, sometimes non-text items are identified as text as well. In addition, the bounding boxes of the chips sometimes do not tightly surround the text strings. The consequence of these problems is that non-text items may occur in the binarized image, produced by mapping the extracted items onto the original page. An example is shown in Figure 5(a,b). These non-text items are not desirable since they may hinder the performance of an OCR system.

However, by treating the extracted items as strokes, the Stroke Filtering process (section 3.2) can be applied here to eliminate the non-text items, since tighter constraints can be used at this time. This is because (1) the clean-up procedure is able to extract most characters without attaching to characters nearby and non-text items (Figure 5(b)), and (2) the strokes at this stage are composed of mostly complete or almost complete characters as opposed to the vertical connected edges of the characters generated by the Stroke Genersation step (section 3.1). Thus, it can be expected that the correct text strokes comply more with the heuristics used in the early Chip Generation phase.

Therefore, the Stroke Filtering procedure (section 3.2) with tighter constraints is used here to remove more non-text items. Then, the Stroke Aggregation process (section 3.3) is used again to generate a new set of probably better chips. This is followed by the Text Clean-up process to extract the text from the input image regions corresponding to the chips. The binarization result is usually better since the tighter the chip bounds the text, the less irrelevant image area (noise) is included, and hence the better the clean-up process works.



(a) (b) (c)

Figure 5: Binarization results before and after the Chip Refinement step. (a) Magnified portion of an input image; (b) result before the refinement; (c) result after the refinement.

A more restricted constraint for connectability of two similar strokes is used at this stage. This constraint requires that the gap between them must be no more than twice the height of the shorter stroke as opposed to three times used in the earlier Chip Generation stage. Also, an extra constraint which requires that the vertical distance between the bottoms or the tops of the strokes be small is used to determine if two strokes are horizontally aligned. In all of the experiments, no more than 10 pixels were allowed for this distance.

An example is given in Figure 5 which shows the binarization results before and after this refinement phase. Figure 5(a) is a portion of a scanned cartoon page in the New Yorker magazine (The full image is not shown due to the space limit). The binarization result before the refinement is shown in 5(b) and the one after the refinement is shown in 5(c). Figure 6 shows an example of the refinement and binarization result on a whole image. A portion of it has been magnified and shown in Figure 5.

7 Experiments

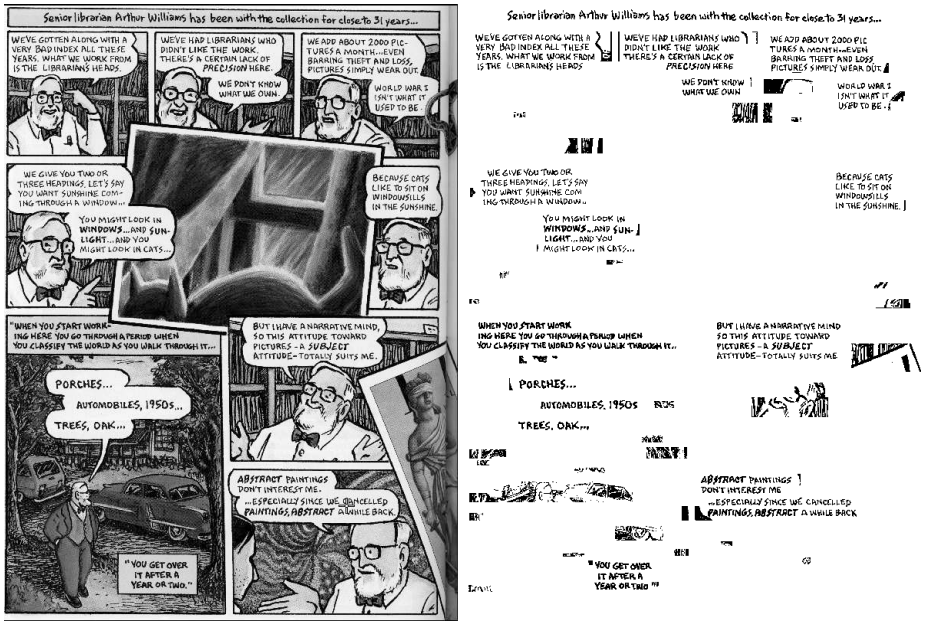
The system has been tested over 48 images. Some of the test images were downloaded from the Internet, some from the Library of Congress, and others were locally scanned documents. These test images came from a wide variety of sources: digitized video frames, photographs, newspapers, advertisements in magazines or sales flyers, and personal checks. Some of the images have regular page layouts, others do not. It should be pointed out that all the system parameters remain the same throughout the whole set of test images, showing the robustness of the system.

For the images scanned by us, a resolution of 300dpi (dots per inch) was used. This is the standard resolution required, for example, by the Caere OCR engine that was used. It should be pointed out that 300dpi resolution is not required by our system. In fact, no assumptions are made about the resolution of the input images, since such information is normally not available for the images from outside sources, such as those downloaded from the Internet.

7.1 Text Detection and Clean-up

This experiment demonstrates the performance of the system up to the Text Filtering process. Characters and words (as perceived by one of the authors) were counted in each image — this is the ground truth. The total numbers over the whole test set are shown in the "Total Perceived" column in Table 1. Then, detected characters and words which are completely enclosed by the text boxes produced after the Chip Scale Fusion step were counted for each image. The total numbers of detected characters and words summed up over the whole test set are are shown in the "Total Detected" column of Table 1. Finally, characters and words which are clearly readable by a person after the Chip Refinement and Text Clean-up steps (final extracted text) are counted for each image. Note that only the text which is horizontally aligned is counted (skew angle of the text string is less than roughly 30 degrees). These extracted characters (words) are called cleaned-up characters (words)[5].

[5]Due to space limitations, the table of the above results itemized for each test image is not included in this paper



(a) (b)

Figure 6: Binarization results after the Chip Refinement step. (a) An input image; (b) Binarization result after the refinement.

As shown in Table 1, there are a total of 21820 characters and 4406 words in the 48 test images. Of these, 20788 (95%) characters and 4139 (93%) words are detected. If the percentages are computed for each image and then averaged over the whole set of images, the normalized percentages (by weighting all the images equally) are 89% for detected characters and 85% for detected words. 91% of the detected characters and 86% of the detected words are successfully cleaned[6]. These results are listed in Table 1.

	Total Perceived	Total Detected	Total Clean-up	Total OCRable	Total OCRed
Char	21820	20788 (95%)	91%	14703	12428 (84%)
Word	4406	4139 (93%)	86%	2981	2314 (77%)

Table 1: Summary of the system's performance. 48 images were used for detection and clean-up. Out of these, 35 binarized images were used for the OCR process.

7.2 OCR Testing

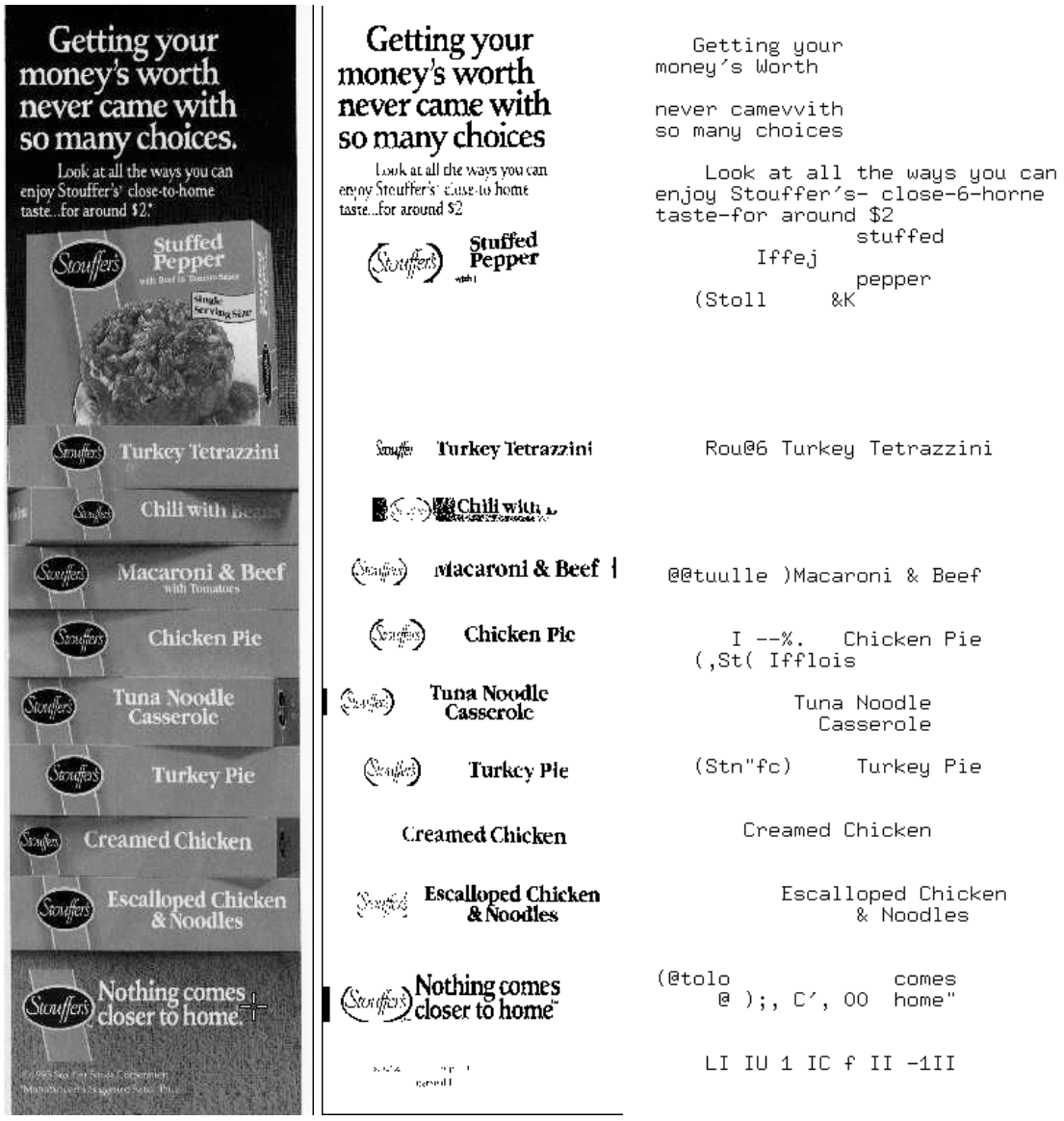
It shoud be pointed out that we do not intend to invent our own OCR system at this point. Instead, Caere's WordScan Plus 4.0 for Windows was used to do character recognition. For this experiment, a binary image is formed using all the cleaned-up text chips for each input image. Then these binary images are manually fed to the OCR system for recognition.

35 images were reconstructed by mapping the extracted text back to the corresponding input images used in the previous experiment. In Table 1, the column "Total OCRable" shows the total number of extracted characters (words) (shown in the Total Clean-up column) that appear to be of machine printed fonts in the corresponding images (Note that only the machine printed characters are counted so that the OCR engine can be applied). The "Total OCRed" column shows the number of characters (words) in these images which are correctly recognized by the OCR engine.

As shown in the table, there are 14703 characters and 2981 printed words which are OCRable in these images. 12428 (84%) of the characters and 2314 (77%) of the words are correctly recognized by the OCR engine. The normalized percentages are 78% and 72% respectively.

Figure 7(a) is the original image of file ads11. This is an image of an advertisement for Stouffer's, which has no structured layout. The final binarization result is shown in the middle. The corresponding OCR output is shown on the right. This example is intended to provide a feeling of the overall performance of the system

[6] A word is successfully cleaned only if all its characters are clearly recognizable by a person.



(a) (b) (c)

Figure 7: Example 1. (a) Original image (ads11); (b) Extracted text; (c) The OCR result using Caere's WordScan Plus 4.0 on b.

by showing whole images. The drawback is that some fine details are lost due to the scaling of the images to fit the page. For example, the words of the smaller fonts and the word Stouffer's in script appear to be fragmented, although actually they are not.

The words under "Stuffed Pepper" were not found because there is little response to the texture segmentation process in that region (see Figure 2). This is because the words are actually blurred, hence the region has very low energy. Notice that most of the texture in the picture of the food is filtered out, showing the robustness of the system.

The OCR engine correctly recognized most of the text of machine-printed fonts as shown in Figure 7 (c). It made mistakes on the Stouffer's trademarks since they are in script. It should be pointed out that the clean-up output looks fine to a person in the places where the rest of the OCR errors occurred.

8 Conclusion

Current OCR and other document segmentation and recognition technologies do not work well for documents with text printed against shaded or textured backgrounds or those with non-structured layouts. In contrast, we have proposed a text extraction system which works well for normal documents as well as documents

described in the above situations.

The system proposed is composed of the following steps. First, a texture segmentation module which directs attention to where the text is likely to occur. Second, strokes are extracted from the segmented text regions. Using reasonable heuristics on text strings such as height similarity, spacing and alignment, the extracted strokes are then processed to form rectangular bounding boxes around the corresponding hypothesized (detected) text strings. To detect text over a wide range of font sizes, the above steps are first applied to a pyramid of images generated from the input image, and then the boxes formed at each resolution level of the pyramid are fused at the image in the original resolution level. Third, an algorithm which cleans up the background and binarizes the detected text is applied to extract the text from the regions enclosed by the bounding boxes in the input image. Finally, the extracted items are treated as strokes, and the more restrictive heuristics are used to generated better boxes for the text strings while eliminating most of the false positive boxes. The clean-up process is then applied again to extract text which can then be processed by an OCR module for recognition.

48 images from a wide variety of sources such as newspapers, magazines, printed advertisement, photographs, and checks have been tested on the system. They are greyscale images with structured and non-structure layouts and a wide range of font styles (including certain script and hand-written fonts) and sizes (practically, the font sizes do not matter for the system). Some text has overlapping background texture patterns in the images.

There are 21820 characters and 4406 words in the test images (perceivable to one of the authors). 95% of the characters and 93% of the words have been successfully extracted by the system. Out of some 14703 characters and 2981 words of extracted text which are of OCR-readable fonts, 84% of the character and 77% of the words are successfully recognized by a commercial OCR system.

The system is stable and robust — all the system parameters remain the same through out all the experiments.

References

[1] H. S. Baird and K. Thompson. Reading Chess. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(6):552–559, 1990.

[2] Mindy Bokser. Omnidocument Technoligies. *Proceedings of The IEEE*, 80(7):1066–1078, July 1992.

[3] 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.

[4] C. A. Glasbey. An Analysis of Histogram-Based Thresholding Algorithms. *CVGIP: Graphical Models and Image Processing*, 55(6):532–537, Nov. 1993.

[5] Anil K. Jain and Sushil Bhattacharjee. Text Segmentation Using Gabor Filters for Automatic Document Processing. *Machine Vision and Applications*, 5, 1992.

[6] 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.

[7] Jitendra Malik and Pietro Perona. Preattentive texture discrimination with early vision mechanisms. *J. Opt. Soc. Am.*, 7(5):923–932, May 1990.

[8] S. Mori, C. Y. Suen, and K. Yamamoto. Historical Review of OCR Research and Development. *Proceedings of The IEEE*, 80(7):1029–1058, July 1992.

[9] G. Nagy, S. Seth, and M. Viswanathan. A Prototype Document Image Analysis System for Technical Journals. *Computer*, pages 10–22, July 1992.

[10] Lawrence O'Gorman. The Document Spectrum for Page Layout Analysis. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15(11):1162–1173, Nov. 1993.

[11] Lawrence O'Gorman. Binarization and Multithresholding of Document Images Using Connectivity. *Computer Vision, Graphics and Image Processing*, 56(6):494–506, Nov. 1994.

[12] Paul W. Palumbo, Sargur N. Srihari, Jung Soh, Ramalingam Sridhar, and Victor Demjanenko. Postal Address Block Location in Real Time. *Computer*, pages 34–42, July 1992.

[13] Theo Pavlidis and Jiangying Zhou. Page Segmentation and Classification. *CVGIP: Graphical Models and Image Processing*, 54(6):484–496, Nov. 1992.

[14] 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.

[15] F. M. Wahl, K. Y. Wong, and R. G. Casey. Block Segmentation and Text Extraction in Mixed Text/Image Documents. *Computer Graphics and Image Processing*, 20:375–390, 1982.

[16] D. Wang and S. N. Srihari. Classification of Newspaper Image Blocks Using Texture Analysis. *Computer Vision, Graphics and Image Processing*, 47:327–352, 1989.

[17] K. Y. Wong, R. G. Casey, and F. M. Wahl. Document Analysis System. *IBM Journal Res. Dev.*, 26(6):647–656, 1982.