TextFinder: An Automatic System To Detect And Recognize Text In Images [*]

Victor Wu, R. Manmatha, Edward M. Riseman

Multimedia Indexing And Retrieval Group

Computer Science Department

University of Massachusetts, Amherst, MA 01003-4610

Email:{vwu,manmatha}@cs.umass.edu

June 10, 1999

[*]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, in part by the National Science Foundation under grant number IRI-9619117 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.

TextFinder: An Automatic System To Detect And Recognize Text In Images

Victor Wu, R. Manmatha, Edward M. Riseman

Abstract

There are many applications in which the automatic detection and recognition of text embedded in images is useful. These applications include digital libraries, multimedia systems, information retrieval systems, and geographical information systems (GIS). When machine generated text is printed against clean backgrounds, it can be converted to a computer readable form (ASCII) using current optical character recognition (OCR) technology. However, text often is printed against shaded or textured backgrounds, or is embedded in images. Examples include maps, advertisements, photographs, videos and stock certificates. Current document segmentation and recognition technologies cannot handle these situations effectively.

In this paper, a four-step system to automatically detect and extract text in images is proposed. First, a texture segmentation scheme is used to focus attention on regions where text 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 rectangular boxes surrounding the corresponding text strings. To detect text over a wide range of font sizes, the above steps are first applied to a pyra-

mid of images generated from the input image, and then the text boxes formed at each resolution level of the pyramid are fused within the image at the original resolution level. Third, text is extracted by cleaning up the background and binarizing the detected text strings, then, better bounding boxes are generated by using the binarized text as strokes. Finally, text is then cleaned and binarized from these new boxes. If the extracted text is of an OCR-recognizable font, it is passed through a commercial OCR engine for recognition.

The system is stable, robust, and works well on images (with or without structured layouts) from a wide variety of sources, including digitized video frames, photographs, newspapers, advertisements, stock certificates, and personal checks. Color images are converted into gray scale images before the algorithm is carried out. All parameters remain the same for all the experiments presented. We also describe a methodology for automatically evaluating such systems and validate it with a manual evaluation technique.

Keywords — text reading system, character recognition, multimedia indexing, text detection, texture segmentation, filters, hierarchical processing, binarization, background removal, connected-components analysis.

1 Introduction

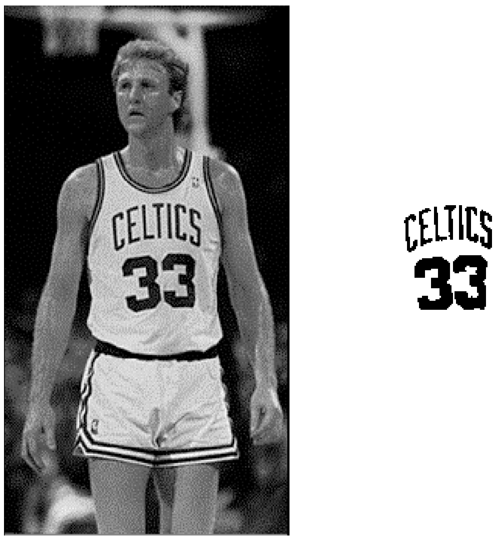
Most of the information available today is either on paper or in the form of still photographs and videos. To build digital libraries, this large volume of information needs to be digitized into images and the text converted to ASCII for storage, retrieval, and easy manipulation. However, current OCR technology [2, 11] 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. 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 non-text items. However, most such schemes require clean binary input [4, 20, 21, 22]; some assume specific document layouts such as newspapers [9] and technical journals [12]; others utilize domain-specific knowledge such as mail address blocks [16] or configurations of chess games [1]. Thus there is a need for systems which extract and recognize text from general backgrounds.

In this paper, a new end-to-end system is proposed which automatically extracts and recognizes text in images. The system takes greyscale images as input[1]. 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.

Systems which automatically extract and recognize text from images with general backgrounds are also useful in the following situations:

1. Text found in images or videos can be used to annotate and index those mate-

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



(a) (b)

Figure 1: An example. (a) An input image; (b) Extracted text before the Character Recognition module.

rials. 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(a, b). In contrast, image indexing based on image content, such as the shape of an object, is difficult and computationally expensive to apply.

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.

2 Prior Work

OCR technology 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 the 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* ([5, 15]). Unfortunately, global thresholding is usually not possible for complicated images, as noted by many researchers ([15], [19]). Consequently, current OCR systems work poorly in these cases.

One solution to the global thresholding problem is to use different thresholds for different local regions (*adaptive thresholding*) [9]. Trier and Taxt [19] 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 [20, 22] to smooth the image first, then, horizontal and vertical projection profiles [21] are commonly used to cut the page into smaller blocks such as columns and paragraphs [12, 17, 21]. 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 [4, 14]. The third category of document segmentation methods treat text as a type of texture and hence use texture segmentation algorithms to detect text [6, 7]. Some work has been done to detect text using color information [25].

Smith and Kanade [18] developed a simple technique to detect text in video images. Although fast, the technique is not robust. Etemad et al [3] used a neural net to classify the output of wavelets into text and non-text regions. The neural net requires a rich set of training examples to work effectively. 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 [6] in principle can be applied to greyscale images, it was only used on binary document images, and in addition, the binarization problem was not addressed. Other systems utilize domain-specific knowledge such as mail address blocks [16] or configurations of chess games [1].

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 or non-structured layouts, and textured or hatched backgrounds. 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.

3 System Overview

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 certain frequency and orientation information; (2) Text shows **spatial cohesion** — characters of the same text string (a word, or words in the same sentence on the same line) 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

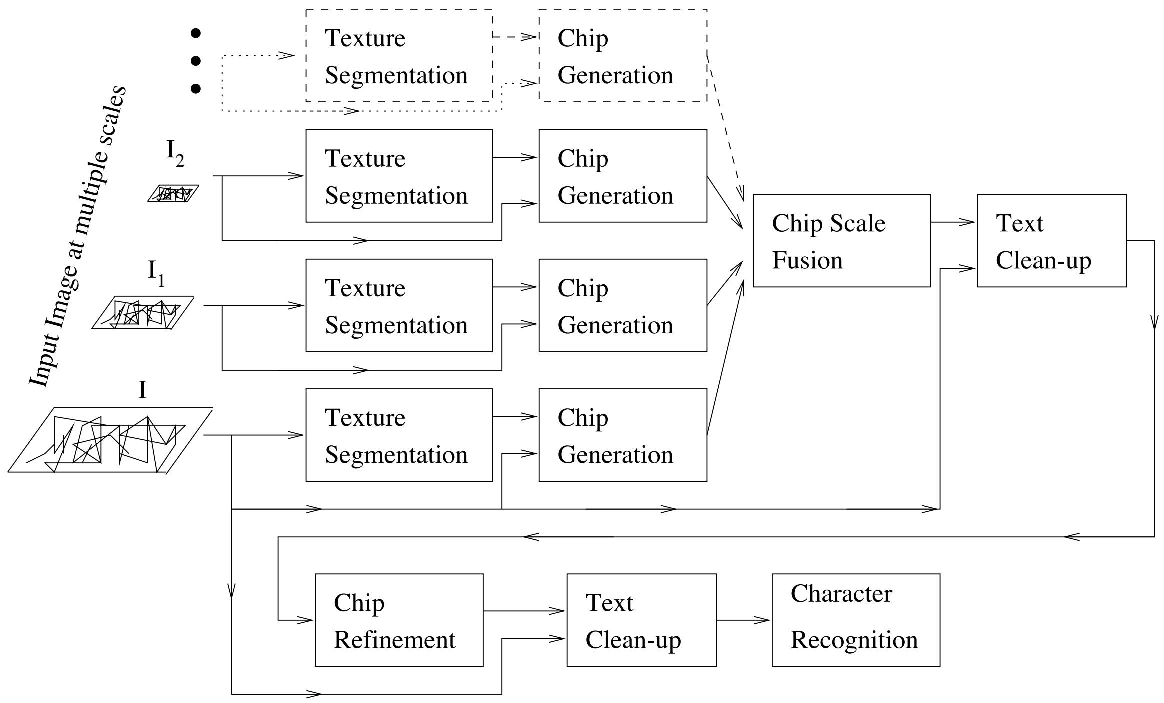


Figure 2: The top level components of the text detection and extraction system. The pyramid of the input image is shown as I, I_1, I_2 ....

the system, therefore, uses **Texture Segmentation** (Figure 2) to segment the text (Section 4). The texture segmentation scheme used is 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 5) 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 spatial cohesion of the text. 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 the locations of the hypothesized (detected) text strings.

The text detection procedures just outlined work well for text over a certain range of font sizes. To detect text whose font size varies significantly, the input image is processed at different resolutions (Section 6). The output chip boxes generated at each resolution level are then 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 2).

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, as described in [23], is used to find 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 than those used 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.

Figure 2 depicts the system described above. Experimental results have shown that the system works well with both machine generated fonts and some script fonts.

Generally the system is not sensitive to the font sizes. 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 9.

4 The Texture Segmentation Module

As stated in section 3 text can be treated as a specific texture, and therefore a natural way to detect text is texture segmentation. A standard approach to texture segmentation is to first filter the image using a bank of linear filters, such as Gaussian derivatives [10] or Gabor functions [6] 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 transformation $tanh(\alpha t)$ where $\alpha = 0.25$. A local energy estimate is computed using the outputs of the non-linear transformation. 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 set of feature vectors is clustered using a K means algorithm (with K = 3).

Since text generally has a stronger response to the filters, while background areas with little intensity variation have little response (i.e. have energy close to zero), the following cluster labeling scheme is used. The cluster whose center is closest to the $(min_1, min_2, \ldots, min_n)$ where min_i is the minimum value for the i^{th} feature, is labeled as background. The cluster whose center is furthest away from the background cluster center is labeled as text.

Figure 3(a) shows a portion of an original input image — a Stouffer's advertisement scanned at 300dpi. Only part of the original input image is shown to fit in the page. 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 3(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 labeled as text pixels.

As shown in Figure 3(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 3(c) shows the result of this operation carried out on the text regions shown in 3(b).

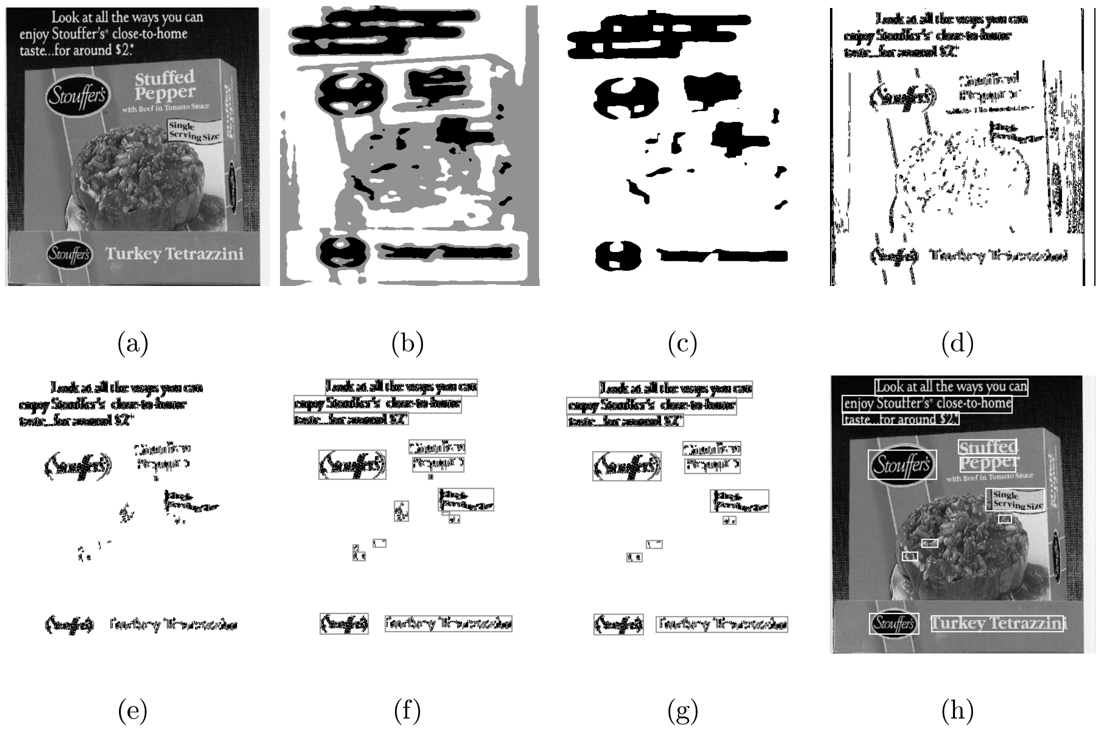


Figure 3: An example of Texture Segmentation and Chip Generation at the half resolution level. (a) Portion of an input image; (b) Output of the clustering stage. White regions are labeled as "text" regions; (c) The Text regions after the morphological closure operation; (d) Strokes produced by performing the Stroke Generation procedure on a; (e) Filtered strokes; (f) Chips (marked by the grey boxes) produced by applying Stroke Aggregation on strokes in e. (g) Chips after the Chip Filtering and Extension processes; (h) Chips mapped to the input image.

5 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 3(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, then a single intensity threshold might not separate both text strings from the background.

Therefore, heuristics need to be employed to refine the segmentation results. Our experiments have shown that the segmentation process usually finds text regions while excluding most non-text areas. These regions can be used to direct further processing. Since text is intended to be readable, there is usually significant contrast between text and background, contrast can be utilized. 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[2].

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, and strokes are connected to form chips corresponding to text strings. The rectangular bounding boxes of the chips are used to indicate the locations of the hypothesized (detected) text strings.

Conceptually, Chip Generation consists of the following main steps which are applied in the order given:

[2]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.

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.

5.1 Stroke Generation

The edges of characters can be expected to have significant contrast in order to be readable. A standard technique is to convolve the input image with a second-order Gaussian derivative in the horizontal direction, and then thresholded to find the significant edges. Connected components computation then groups edges into **strokes**.

Empirically, $\sigma = 1$ was found to be a reasonable choice, with the threshold value set to 10 for all the experiments. Figure 3(d) shows the strokes for Figure 3(a).

5.2 Stroke Filtering

As one can see in Figure 3(d), strokes are found at regular intervals in regions where text is present. However, non-text strokes will also be extracted where there are significant horizontal intensity changes in a scene.

Therefore, the purpose of Stroke Filtering is to eliminate the false positive strokes by using heuristics which take into account the fact that neighboring characters/words 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. These heuristics can be described using **connectability** which is defined as:

Definition 1 *Strokes A and B are* **connectable** *if they are of similar height and horizontally aligned, and there is a* **path** *between A and B, where a path is a horizontal sequence of consecutive pixels in the segmented region which connects A and B by 4-neighbor adjacency.*

Reasonable assumptions are made for text string formation. 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.
    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.

Sometimes words were extracted as one connected component by the stroke generation module, especially when the the contrast is poor. If there is only one such word in the neighborhood, it will be incorrectly removed by condition 2. Therefore, in practice, strokes are also retained if they are sufficiently wide with aspect ratios implying text strings. We have chosen widths of at least 20 pixels, and aspect ratio (width/height) of at least 0.9.

Figure 3(e) shows the result of applying this procedure to the strokes in figure 3(d). Notice that most of the text is still present while more of the background has been eliminated.

5.3 Stroke Aggregation

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 to generate chips that correspond to text strings. In addition, 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 across a range of text sources, the spacing between the characters and words of a text string is usually less than twice 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 twice the height of the taller stroke.

Figure 3(f) shows the result of applying the Chip Generation procedure to the strokes in figure 3(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.

5.4 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 patterns in the image, such as vertical lines and window frames.

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 unacceptable small 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 (width/height)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, $ratio_{\tau} = 0.9$ is used here, attempting to cover that case to some extent.

5.5 Chip Extension

It is to be expected that some of the strokes may 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.

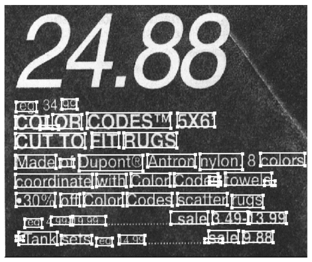
Fortunately, this fragmentation problem can usually be corrected. Notice that the chips corresponding to the same text stroke are still horizontally aligned and of similar height. Thus, by treating the chips as strokes, the Stroke Aggregation procedure can be applied again to aggregate the chips into larger chips and capture more complete words in the extended chips.

Figure 3(g) shows the result of applying the Chip Filtering and Extension steps to the chips in Figure 3(f). The rectangular chip bounding boxes are mapped back onto the input image to indicate detected text as shown in Figure 3(h).

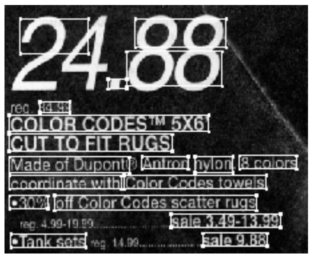
6 A Solution to the Scale Problem

The three frequency channels used in the segmentation process work well to cover text over a certain range of font sizes, but 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., and when text font size gets too large, its frequency falls outside the three channels selected in section 4.

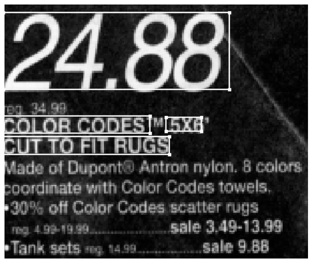
We propose a pyramid approach to the scale problem. A pyramid of input images and each image is processed using the standard channels ($\sigma = 1, \sqrt{2}, 2$) as described in the previous sections (see Figure 2). The original image is at the bottom of the pyramid, and the image at each next level is one-half the resolution in both dimensions. Text of smaller font sizes can be detected using the images lower in the pyramid as shown in Figure 4(a) while text of large font sizes is found using images higher in the pyramid


24.88
reg. 34.99
COLOR CODES™ 5X6'
CUT TO FIT RUGS
Made of Dupont® Antron nylon. 8 colors
coordinate with Color Codes towels.
•30% off Color Codes scatter rugs
reg. 4.99-19.99 sale 3.49-13.99
•Tank sets reg. 14.99 sale 9.88

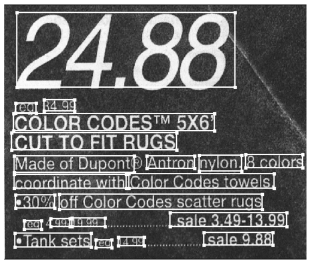
(a)


24.88
reg. 34.99
COLOR CODES™ 5X6'
CUT TO FIT RUGS
Made of Dupont® Antron nylon. 8 colors
coordinate with Color Codes towels.
•30% off Color Codes scatter rugs
reg. 4.99-19.99 sale 3.49-13.99
•Tank sets reg. 14.99 sale 9.88

(b)


24.88
reg. 34.99
COLOR CODES™ 5X6'
CUT TO FIT RUGS
Made of Dupont® Antron nylon. 8 colors
coordinate with Color Codes towels.
•30% off Color Codes scatter rugs
reg. 4.99-19.99 sale 3.49-13.99
•Tank sets reg. 14.99 sale 9.88

(c)


24.88
reg. 34.99
COLOR CODES™ 5X6'
CUT TO FIT RUGS
Made of Dupont® Antron nylon. 8 colors
coordinate with Color Codes towels.
•30% off Color Codes scatter rugs
reg. 4.99-19.99 sale 3.49-13.99
•Tank sets reg. 14.99 sale 9.88

(d)

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

as shown in Figure 4(c). The bounding boxes of detected text regions at each level are mapped back to the original input image (bottom level).

7 Text on Complex Backgrounds

Text strings may be printed against complex image backgrounds (see for example Figure 5(a)), but current OCR systems require text on 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

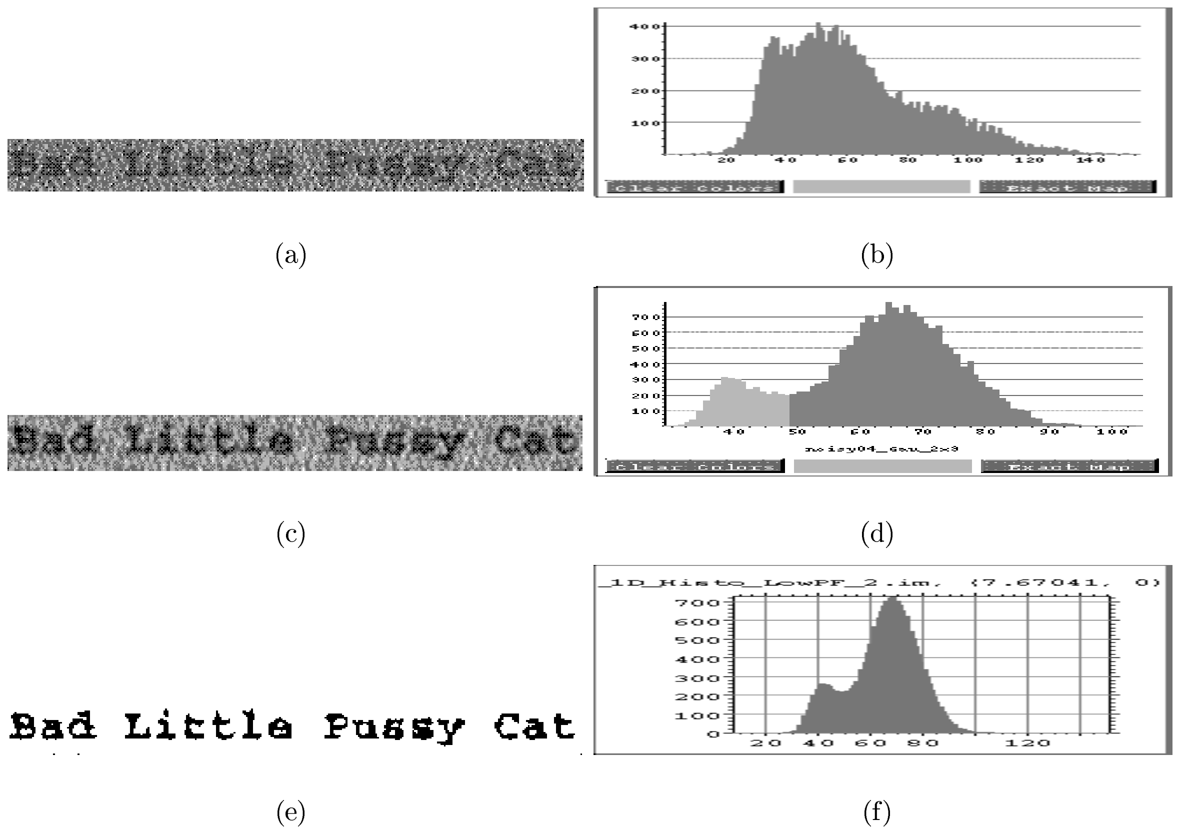


Figure 5: The Text Clean-up process. (a) Original text chip; (b) Histogram of a; (c) Smoothed version of a; (d) Histogram of c; (e) The binarization result by thresholding c using a value in the valley of f; (f) Smoothed version of d.

which current OCR systems can handle. Local thresholding is a good way to perform 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 algorithm for background removal and binarization is described in [23]. Here, we use Figure 5 to briefly demonstrate how the algorithm works: First, the text chip (Figure 5(a)) generated by the system is smoothed to produce the chip shown as Figure 5(c). Next, the intensity histogram of the smoothed chip is computed as shown in Figure 5(d). 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 5(d) to produce the histogram in Figure 5(f). The histogram smoothing eliminates the small noise peaks in Figure 5(d). The text in Figure 5(c) can then be automatically extracted by picking a threshold at the first valley counted from the left side of the histogram in Figure 5(f). The resulting thresholded output is shown in Figure 5(e). All the characters but one "s" have been successfully recognized using an commercial 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 the Experiments section). More discussion can be found in [23].

8 The Chip Refinement

Experiments show that the text detection phase is able to locate text strings in regular fonts, and even from some 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 Figures 6 and 7. These non-text items are not desirable since they may hinder the performance of an OCR system.

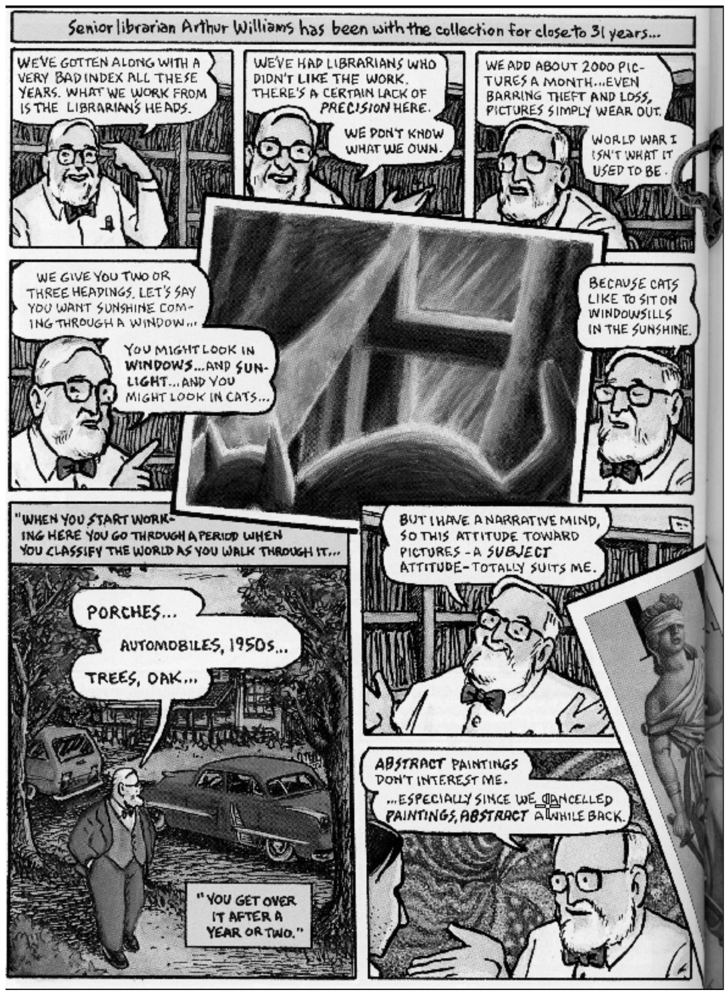

Senior librarian Arthur Williams has been with the collection for close to 31 years...
WE'VE GOTTEN ALONG WITH A VERY BAD INDEX ALL THESE YEARS. WHAT WE WORK FROM IS THE LIBRARIAN'S HEADS.
WE'VE HAD LIBRARIANS WHO DIDN'T LIKE THE WORK. THERE'S A CERTAIN LACK OF PRECISION HERE.
WE DON'T KNOW WHAT WE OWN.
WE ADD ABOUT 2000 PICTURES A MONTH...EVEN BARRING THEFT AND LOSS, PICTURES SIMPLY WEAR OUT.
WORLD WAR I ISN'T WHAT IT USED TO BE.
WE GIVE YOU TWO OR THREE HEADINGS. LET'S SAY YOU WANT SUNSHINE COMING THROUGH A WINDOW...
YOU MIGHT LOOK IN WINDOWS...AND SUNLIGHT...AND YOU MIGHT LOOK IN CATS...
BECAUSE CATS LIKE TO SIT ON WINDOWSILLS IN THE SUNSHINE.
"WHEN YOU START WORKING HERE YOU GO THROUGH A PERIOD WHEN YOU CLASSIFY THE WORLD AS YOU WALK THROUGH IT...
PORCHES...
AUTOMOBILES, 1950s...
TREES, OAK...
"YOU GET OVER IT AFTER A YEAR OR TWO."
BUT I HAVE A NARRATIVE MIND, SO THIS ATTITUDE TOWARD PICTURES - A SUBJECT ATTITUDE - TOTALLY SUITS ME.
ABSTRACT PAINTINGS DON'T INTEREST ME.
...ESPECIALLY SINCE WE CANCELLED PAINTINGS, ABSTRACT A WHILE BACK.

Figure 6: An input image from the New Yorker magazine.

However, by comparing the extracted items (characters and non-character objects) at this stage with strokes after the Stroke Generation step, we observe the following:

(1) the clean-up procedure in general is able to extract most characters without attaching to characters nearby and non-text items (Figure 7);

(2) the extracted characters are usually complete as opposed to the vertical connected edges of the characters generated by the Stroke Generation step (section 5.1).

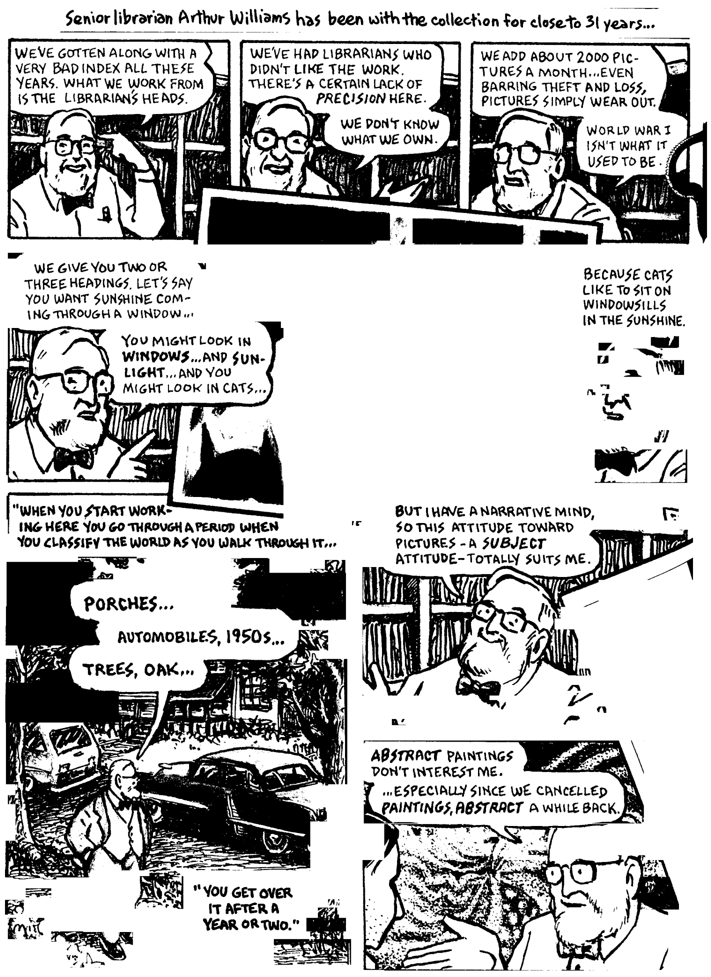

Senior librarian Arthur Williams has been with the collection for close to 31 years...
WE'VE GOTTEN ALONG WITH A VERY BAD INDEX ALL THESE YEARS. WHAT WE WORK FROM IS THE LIBRARIAN'S HEADS.
WE'VE HAD LIBRARIANS WHO DIDN'T LIKE THE WORK. THERE'S A CERTAIN LACK OF PRECISION HERE.
WE DON'T KNOW WHAT WE OWN.
WE ADD ABOUT 2000 PICTURES A MONTH...EVEN BARRING THEFT AND LOSS, PICTURES SIMPLY WEAR OUT.
WORLD WAR I ISN'T WHAT IT USED TO BE.
WE GIVE YOU TWO OR THREE HEADINGS. LET'S SAY YOU WANT SUNSHINE COMING THROUGH A WINDOW...
YOU MIGHT LOOK IN WINDOWS...AND SUNLIGHT...AND YOU MIGHT LOOK IN CATS...
BECAUSE CATS LIKE TO SIT ON WINDOWSILLS IN THE SUNSHINE.
"WHEN YOU START WORKING HERE YOU GO THROUGH A PERIOD WHEN YOU CLASSIFY THE WORLD AS YOU WALK THROUGH IT...
PORCHES...
AUTOMOBILES, 1950S...
TREES, OAK...
"YOU GET OVER IT AFTER A YEAR OR TWO."
BUT I HAVE A NARRATIVE MIND, SO THIS ATTITUDE TOWARD PICTURES - A SUBJECT ATTITUDE - TOTALLY SUITS ME.
ABSTRACT PAINTINGS DON'T INTEREST ME.
...ESPECIALLY SINCE WE CANCELLED PAINTINGS, ABSTRACT A WHILE BACK.

Figure 7: The binarization result of Figure 6 before the refinement step.

In other words, compared with the strokes generated at the Storke Generation step, the extracted items at this stage are more “text-like” if they belong to text strings. Thus, it can be expected that these items comply more with the heuristics used in the early Chip Generation phase.

Therefore, by treating the extracted items as “strokes”, the Stroke Filtering procedure (section 5.2) with tighter constraints can be used here to remove more non-text items. Then, the Stroke Aggregation process (section 5.3) is used again to generate

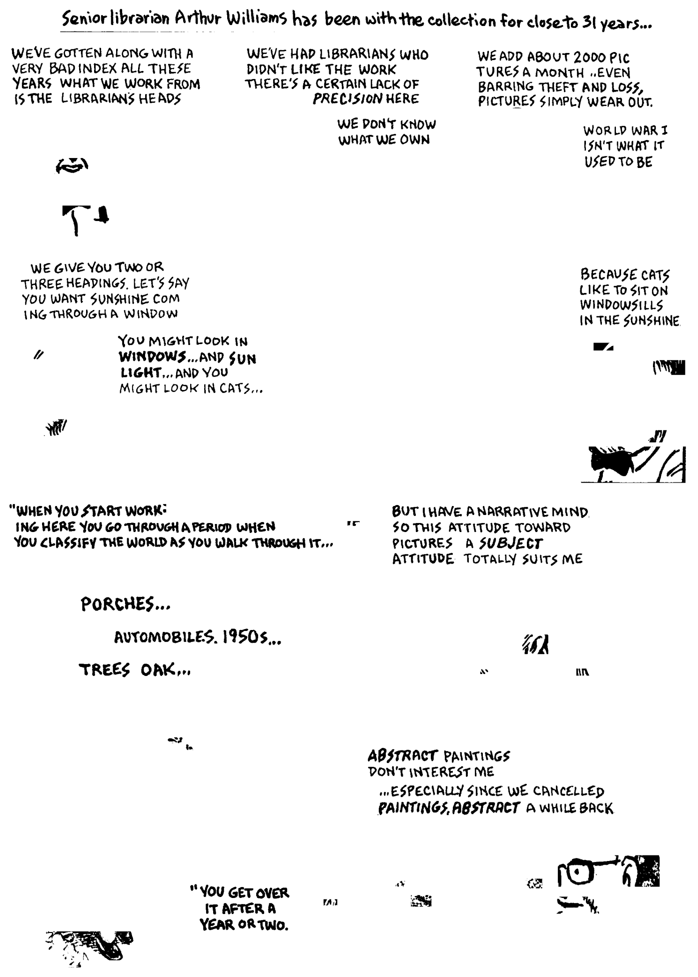

Senior librarian Arthur Williams has been with the collection for close to 31 years...
WE'VE GOTTEN ALONG WITH A VERY BAD INDEX ALL THESE YEARS WHAT WE WORK FROM IS THE LIBRARIAN'S HEADS
WE'VE HAD LIBRARIANS WHO DIDN'T LIKE THE WORK THERE'S A CERTAIN LACK OF PRECISION HERE
WE ADD ABOUT 2000 PICTURES A MONTH ..EVEN BARRING THEFT AND LOSS, PICTURES SIMPLY WEAR OUT.
WE DON'T KNOW WHAT WE OWN
WORLD WAR I ISN'T WHAT IT USED TO BE
WE GIVE YOU TWO OR THREE HEADINGS. LET'S SAY YOU WANT SUNSHINE COMING THROUGH A WINDOW
BECAUSE CATS LIKE TO SIT ON WINDOWSILLS IN THE SUNSHINE
YOU MIGHT LOOK IN WINDOWS...AND SUN LIGHT...AND YOU MIGHT LOOK IN CATS...
"WHEN YOU START WORKING HERE YOU GO THROUGH A PERIOD WHEN YOU CLASSIFY THE WORLD AS YOU WALK THROUGH IT...
BUT I HAVE A NARRATIVE MIND SO THIS ATTITUDE TOWARD PICTURES A SUBJECT ATTITUDE TOTALLY SUITS ME
PORCHES...
AUTOMOBILES. 1950S...
TREES OAK...
ABSTRACT PAINTINGS DON'T INTEREST ME
...ESPECIALLY SINCE WE CANCELLED PAINTINGS, ABSTRACT A WHILE BACK
"YOU GET OVER IT AFTER A YEAR OR TWO.

Figure 8: The binarization result of Figure 6 after the refinement step.

a new set of (hopefully better) chips, followed by the Text Clean-up process. 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 more restricted constraint for connectability of two similar strokes is used at this stage. For example, this constraint requires that the gap between them must be no more than the height of the shorter stroke as opposed to twice the distance used in

Color	Greyscale	Size (W×H)	
		min	max
27	21	186×349	3391×2486

Table 1: Characteristics of the image data base.

the earlier Chip Generation stage. Higher percentage is also used to determine if two strokes are horizontally aligned.

Text does not usually overlap with other text. Therefore, a text chip should not contain other characters which does not belong to this chips. However, chips formed by large non-text items, such as the rectangular frames in the cartoon images, do contain text in it. To remove these non-text chips at this stage, an extra filtering constraint is used: *a chip after the stroke aggregation procedure is removed if it contains reasonable number of strokes which do not belong to this chip.*

An example is given in Figure 8 which shows the binarization result of Figure 6 after this refinement phase.

9 Experiments

Ads	Photo / Video	Check	Envelop	Map	Cartoon	Invoice	Other
20	8	3	1	1	5	2	8

Table 2: Characteristics of the image type.

	Newspaper	Magazine	Web	Flyer	Other
count	4	10	4	13	17

Table 3: Characteristics of the image source.

There are 48 test images in our database, and more will be added over time. Tables

1, 2, 3 and 4 show the characteristics of the test images in the database. Some of the test images were downloaded from the internet (the web images), 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, personal checks and envelopes. Some of the images have regular page layouts, others do not.

For the images scanned by us, a resolution of 300 dpi (dots per inch) was used. This is the standard resolution required, for example, by the Caere OCR engine (WordScan) that was used. It should be pointed out that 300 dpi 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 images from outside sources, such as those downloaded from the Internet.

Color documents were scanned as color images, and then converted into greyscale using John Bradley's image displaying/processing system called XV. More than half of the images currently in the data base were originally in color. The image sizes varied from 186×349 (a thumbnail image from the web) to 3391×2486 (roughly a full 8.5×11 square inch document scanned at 300 dpi). The height of the text varied from 6 pixels to 197 pixels.

Eight images in the set were used to set and test the parameters. These parameters were applied to all the images. The results (discussed later in this section) show the robustness of the system.

9.1 Evaluation Methodology

As discussed above the collection consists of many different kinds of images. The text collection has a variety of characters in different types of fonts. For example, some of the characters are machine fonts. Most of the characters in the Stouffer's advertisement (Figure 10a are of this kind. Some of the characters are handwritten using block printing. The cartoon in Figure 6 consists of such characters. Decorative fonts are often used for logos. For example, the Stouffer's logo is printed using such a decorative font (Figure 10a.

There has been little work on techniques to automatically evaluate text detection on such diverse document collections. Although two recent papers provide results on detection accuracy and false alarms, the methodology has either been restricted to specific classes of images or is not extendable for automatic evaluation on large collections of diverse images. Smith and Kanade [18] evaluate their algorithm for detecting caption text in video images. They, however, do not provide any details on whether their evaluation methods were manual (counting characters) or automatic, and if automatic as to how it was done. Jain and Yu [?] report results on a variety of images. They state that "We compute the accuracy for advertisement images by manually counting the number of correctly located characters. The accuracies for other images are subjectively computed based on the number of correctly located important text in the image".

Jain and Yu approach is reasonable, but clearly not practical for large collections.

Note that the advertisement images they show are much simpler than ours. There are a number of deficiencies with Jain and Yu's approach to evaluating other (non-aadvertisement) images. First, it is not clear as to what text is "important" (and who decides what is important). For example, they show an image of a weather map from a television program in which their algorithm misses all of the town names and the temperatures, but were not counted as errors. Clearly in the context of a weather map, town names and temperatures are important pieces of information. Second, it is not clear what they mean by correctly located text regions. Are these regions characters, words, units larger than words, or sentences? Their technique for evaluation does not allow for comparisons between different systems.

We propose a technique for automatic evaluation for text detection in images. First, ground truth is created by manually constructing minimum bounding boxes around each character in the image. The output produced by the text detection algorithm is compared against the ground truth. If the output of the text detection algorithm completely covers the box created for ground truth, then the character is regarded as being detected, otherwise it is regarded as not being detected. Areas of the image that are detected but do not contain text are regarded as false alarms. The percentage of this area as a function of the total area of the image is reported as the false alarm rate. We validate this approach to automatic evalution by visually inspecting and finding out which characters have been detected and showing that we get similar results using both the automatic evaluation and visual inspection. We now discuss all these aspects in greater detail below.

9.1.1 Creating Ground Truth

The ground truth is created by manually outlining “minimum bounding” boxes for each character in the images. Boxes are also manually drawn around each word. If the character or word belongs to a machine font (as determined by visual inspection) it is tagged as such. Certain criteria are followed when the boxes are created. Some of these criteria are meant to facilitiate automatic evaluation. Others are motivated by the possible applications of the text detection scheme described here.

1. Tiny characters are omitted. Such small characters often show up poorly, since they often are not well formed, sometimes aliased, hard to detect, and OCR systems have difficulty recognizing them. The criteria we use to consider whether a character is tiny are context sensitive. If the majority of the characters in a word are greater than 8 pixels in height, then all the characters in that word are tagged. So for example, consider the word “buy”. Since “b” has an ascender and “y” a descender, both these characters will have greater height than “u”. If both “b” and “y” are greater than 8 pixels in height, then the character “u” is also tagged even if it is not 8 pixels in height.

2. Characters which touch each other are omitted. When characters touch each other, it is difficult to delineate where one character ends and another one begins. This poses problems for the automatic evaluation technique and hence it was

	min	max	mean	std. dev.
Height	6	197	25.9	13.9
Width	2	240	17.6	11.8

Table 4: Font sizes in pixels.

decided to eliminate such characters altogether. Note that the text detection system described in this paper actually detects many such characters.

3. The text detection system described here is intended to find characters which are roughly horizontal. Thus, only characters whose skew angle is less than roughly 30 degrees are included.

4. Blurred or low-contrast characters are omitted. Often it is hard for a human being to read such text and it is, therefore, unreasonable to expect a machine to detect them. For the same reason, characters which have poor contrast are omitted.

5. Characters which are significantly slanted in depth are also omitted.

There are 22030 characters (4806 words) in this dataset from from 48 images for the testing. Table 4 shows the variations in font size of the selected text. We intend to make this dataset publicly available in the future so that other researchers can publish comparative results.

9.1.2 Detection Accuracy

The percentage of the total number of characters that have been detected will be used as a measure of the detection accuracy of the sytem. A character is considered to be detected if it is completely covered by a generated text box. Here, we use this manual approach of visual inspection to validate our automatic evalution scheme. Table 5 shows the results obtained by manually counting the characters.

As shown in Table 5, 93.5% of the 22030 characters (90.2% of the 4806 words) were successfully detected (A word is detected if and only if all of its characters are detected). Considering the content complexity of the test images, these detection rates can be considered very high in comparison with other work in the field. Table 5 also shows that the system does well in detecting text whose font height is more than ten pixels, but as expected it does not work well for very small characters.

Since we have the "minimum" bounding boxes for the reference characters and words (the ground truth), the area of the reference boxes covered by the generated text boxes may be easily computed. Thus, the area of the covered reference boxes is used here as a measurement [3]. If the reference bounding boxes are accurate, a character should be considered as being detected if its reference bounding box is completely covered by one of the generated text boxes.

However, this criterion is accurate only if the bounding boxes are minimal — making them any smaller will lose part of the characters. Unfortunately, it is often hard to

[3]Note that the borders of the boxes are not considered.

tell where the borders of a character should be in a grayscale image. The consequence is that a manually drawn "minimum" bounding boxes might not be minimal. Thus, the all-or-nothing criterion tends to underestimate the true performance of the system, as shown in the last column of Table 6. Therefore, for the automatic measurement, a character is considered detected if at least a certain percentage, τ, of its the area of its bounding box is covered by the generated boxes. Table 6 shows the system performance using this criterion. For each threshold $\tau >= 80\%$, $\tau >= 90\%$ and $\tau >= 100\%$ (complete coverage), the detection rates in terms of the percentage of detected characters over the total number of the characters are shown.

Comparing Tables 5 and 6, it is clear that the performance as reported by the automatic evaluation system is close (but slightly lower) than that obtained using visual inspection. Thus, the detection rates obtained using the automatic evaluation schemes are conservative estimates. Although the performance at 80% coverage is slightly better than at 100% coverage, we shall again be conservative and assume that the performance of the algorithm is best estimated by using the figures for 100% coverage.

Height (pixels)	# of chars. of	# of chars Detected	
<= 10	705	393	55.7%
11 – 20	7571	6890	91.0%
> 20	13754	13310	96.8%
>= 6	22030	20593	93.5%

Table 5: Detection rates by visual inspection.

Height (pixels)	# of chars. of	Detection rates by char coverage		
		$\tau >= 80\%$	$\tau >= 90\%$	$\tau >= 100\%$
<= 10	705	55.5%	55.2%	55.2%
11 – 20	7571	90.5%	90.4%	90.0%
> 20	13754	96.5%	96.3%	95.2%
>= 6	22030	93.2%	93.0%	92.1%

Table 6: Detection rates using the automatic evaluation scheme.

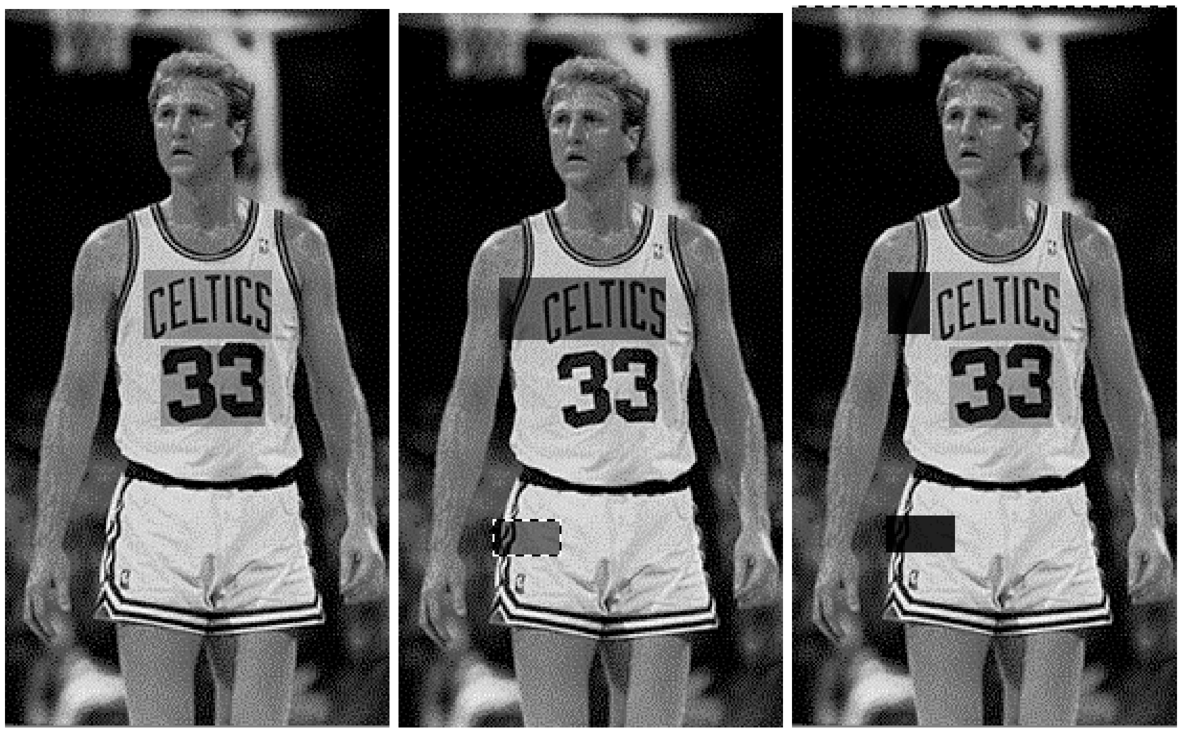
Both tables also show that the performance for tiny characters ≤ 10 pixels is poor. Larger characters (11 pixels or greater in height) have detection rates above 90%.

9.2 False Alarm

Another important issue for evaluating a text detection system is the false alarm rate. A high detection rate is meaningful only if the false alarm is also reasonably low at the same time. For example, one can just put a box on a whole image to guarantee a 100% detection rate, but this would be of little use.

The false alarm rate could be measured by counting the number of boxes which do not contain text. There are a couple of problems with this approach. First, it is often the case that only parts of the generated boxes do not belong to text. Second, the time wasted in processing many tiny false positive regions might be much less than that in processing one big false positive region. Thus, it makes more sense to measure false positives in terms of the area of the parts of the minimum bounding boxes which do not overlap with the reference text region (i.e. the ground truth).

A pixel in a generated text chip is a false alarm if it is not inside any of the pre-



(a) (b) (c)

Figure 9: Identifying false alarm. (a) Input image with pre-drawn text regions shown in the grey rectangular areas. (b) Hypothetically generated text regions shown in the grey rectangular areas. (c) False alarm (dark) and pre-drawn (light) rectangular regions.

drawn boxes. Figure 9 illustrates false alarm pixels. The false alarm rate is then defined as the total area covered by the false alarm pixels expressed as a percentage of the image size. For the set of 48 images, the false alarm rate was only 5.3%. The area covered by the false alarm pixels as a proportion of the area covered by the text chips is 30.1%.

9.3 Clean-up Accuracy

The accuracy of the clean-up algorithm was also measured both visually and more objectively using an OCR. A character is successfully cleaned up by visual inspection if its binarized result is clearly recognizable by a person. The objective measurement is described in section 9.4. Table 7 shows the clean-up rates judged by visual inspection. Column 4 of Table 7 shows the number of characters which have been successfully cleaned-up and binarized while column 5 shows the number of cleaned up characters as a percentage of the number of detected characters. The last column shows the number cleaned-up as a percentage of the number of characters in the ground truth. The table shows that the clean-up and binarization algorithm is robust for text which is at least eleven pixels in height. Overall, 96.7% of the detected characters are successfully cleaned and binarized.

Height (pixels)	# of Chars of N and C types	# of Chars Detected	# of chars Cleaned		
			Count	% over Detected	% over Column 2
<= 10	705	393	164	41.7%	23.3%
11 – 20	7571	6890	6550	95.1%	86.5%
> 20	13754	13310	13198	99.2%	96.0%
>= 6	22030	20593	19912	96.7%	90.4%

Table 7: Clean-up and binarization rates using visual inspection.

More discussion on the clean-up algorithm may be found in [24, 23].

9.4 OCR Testing

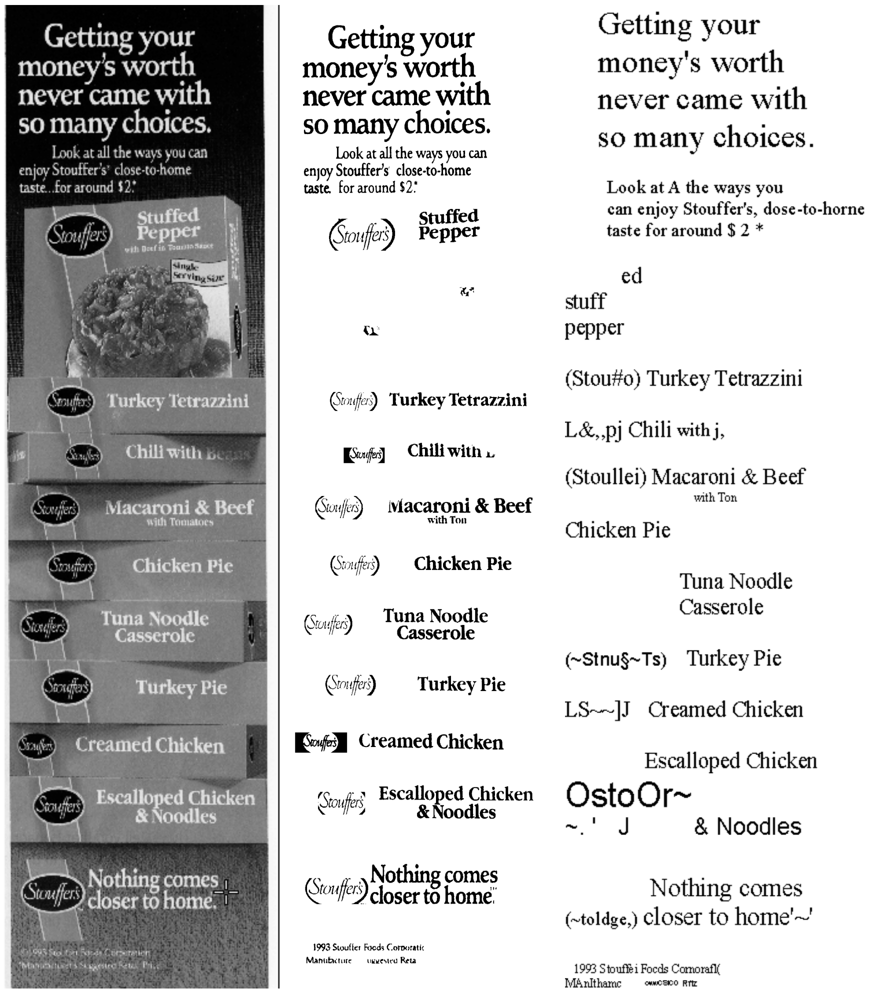
Caere's OmniPage Pro 8.0 for Windows95 was applied for character recognition. For this experiment, a binary image is formed using all the cleaned-up text chips for each input image. Then these binary images are manually fed to the OCR system for recognition. This test also provides an objective evaluation of the text clean-up and binarization algorithm.

	Total in database	Extracted	Recognized		
			Count	% over Extracted	% over Total
Char	18688	16664	15656	94.0%	83.8%
Word	4042	3304	2926	88.6%	72.4%

Table 8: OCR test results.

In Table 8, the column "Total" shows the total number of extracted characters (words) which appear to be of machine printed fonts in all the 48 test images. Only the machine printed characters are evaluated since the OCR engine cannot reliably recognize text of other fonts. The "Recognized" columns shows the number and the percentage of characters (words) in these images which are correctly recognized by the OCR engine. As shown in Table 8, 94.0% of the characters and 88.6% of the words are correctly recognized by the OCR engine. We'd like to point out that for many of the input images, applying the OCR engine directly without the clean-up procedure yields very poor results if not failing completely. This shows the importance of having a robust binarization and clean-up algorithm.

Figure 10(a) is the full original image of an advertisement for Stouffer's which


Getting your money's worth never came with so many choices.
Look at all the ways you can enjoy Stouffer's close-to-home taste...for around $2.*
Stuffed Pepper
Turkey Tetrazzini
Chili with Beans
Macaroni & Beef with Tomatoes
Chicken Pie
Tuna Noodle Casserole
Turkey Pie
Creamed Chicken
Escalloped Chicken & Noodles
Nothing comes closer to home.

```
Getting your
money's worth
never came with
so many choices.

Look at A the ways you
can enjoy Stouffer's, dose-to-horne
taste for around $ 2 *

   ed
stuff
pepper

(Stou#o) Turkey Tetrazzini

L&,,pj Chili with j,

(Stoullei) Macaroni & Beef
                with Ton
Chicken Pie

             Tuna Noodle
             Casserole

(~Stnu§~Ts)   Turkey Pie

LS~~]J   Creamed Chicken

          Escalloped Chicken
OstoOr~
~. '  J        & Noodles

           Nothing comes
(~toldge,) closer to home'~'

1993 Stouffei Foods Comoraf\(
MAnlthamc
```

(a) (b) (c)

Figure 10: Example 1. (a) Original image; (b) Extracted text; (c) The OCR result using Caere's OmniPage Pro 8.0 on b.

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 feel for the overall performance of the system by showing whole images. In this presentation 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.

All the characters under "Stuffed Pepper" were missed due to their poor contrast and small size. 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 much the text of machine-printed fonts as shown in Figure 10 (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 many of the OCR errors occurred.

9.5 Speed

Convolution is extensively used in this system, especially in the texture segmentation phase. As a result, most of the time is spent on the texture segmentation phase. This situation is especially serious in the cases when input images are large. For example, computing the nine energy features of an image of size 3391×2486 takes about 2 minutes on our Pentium Pro PC with 200 MHz CPU and 128M memory running Linux. Clustering 3391×2486 nine dimensional features also takes about 2 minutes.

To speed up the system, only one instead of three channels of the second-order Gaussian derivative filters may be used for the texture segmentation phase. This reduces the feature space from nine dimensions to three dimensions. In our experiment, the frequency channel corresponding to $\sigma = \sqrt{2}$ worked well.

Table 9 shows the user time needed for processing images of different sizes, using

System	Time in seconds		
	Image Size (W $\times$ H)		
	3391×2486	1512×1517	320×240
1-channel	558	143	3
3-channel	1399	291	10

Table 9: Processing time in seconds needed for images of different sizes.

both the 1 channel and the 3 channel texture segmentation modules on a Pentium Pro PC with 200 MHz CPU and 128M memory running Linux. Each entry in the table was measured for an image at that size (i.e., they are not averages).

It should be pointed out that there is still a lot of room to reduce the processing time. For example, the multi-resolution texture segmentation is a parallel process, but it was only implemented as a serial process. Furthermore, the system is implemented in C++, and minimal programming effort was spent on making the system run faster. For example, the system was implemented as modules, and intermediate results were also generated to make debugging and analyzing easier. The extra I/O involved in this process takes a significant amount of time.

It is important to point out that both the detection rates and false alarm rate of the 1-channel system are almost identical to those of the 3-channel system, as shown in Table 10. However, the 1-channel system reduces the processing time by more than 50%.

System	Detection Rates by Char Count			False Alarm
	$\tau >= 80\%$	$\tau >= 90\%$	$\tau = 100\%$	w.r.t image size
1-channel	93.5%	93.2%	92.4%	5.6%
3-channel	93.2%	93.0%	92.1%	5.3%

Table 10: Comparison of the overall performance of the 1-channel and 3-channel systems.

10 Discussion

There are systems [25, 8, ?] which utilize color information to detect text in color images. The system described in this paper does not use color. It handles color images by first converting them to grayscale images. However, 27 of the 48 images evaluated were originally in color, and as the results show the system does quite well on these images. To understand why the system does well without utilizing color information, Nevatia [13] pointed out in his work on color edge detection that in general color edges are also intensity edges. Our system starts by extracting significant intensity edges (strokes). Thus, "strokes" which were originally in color are usually still present in the converted greyscale image. It is possible that some of the characters may be printed in colors such that there is little or no contrast with the background in greyscale. The system would have problems with such characters. However, we believe (and the results bear us out) that such characters are not common, but in such cases, use of one or more color bands might be effective.

It would be desirable to compare the result with other published results in other domains such as maps, engineering drawings and mail pieces. Unfortunately, without access to these databases and agreements on how the measurements should be done, it is difficult to make any meaningful comparisons. However, we believe this kind of comparison is important. In an effort to encourage improved community evaluation, we will make our test database available to anyone who may be interested.

This system is not sensitive to image resolution. It works particularly well in extracting text from textured and/or hatched background. However, it tends to have problem extracting very small text (font height less than 10 pixels) or text with poor contrast. There are a couple of reasons for this. To begin with, it is very hard to extract strokes needed by the successive steps in these situations. When text has poor contrast, it has low energy as well, so the texture segmentation phase may misclassify it as background. We are working on solving these problems. One possibility is to use adaptive thresholding algorithm, such as Kamel and Zhao's Logical Level thresholding algorithm [9], to extract strokes in the stroke generation step.

11 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 first uses a text segmentation procedure to focus attention on regions where text may occur, and then a Chip Generation module is used to find actual text strings within these regions. Reasonable heuristics on text strings, such as height similarity, spacing and horizontal alignment are used in this module. Multi-scale processing is used to account for significant variation in font sizes. Detected text strings are cleaned up and binarized before actual character recognition begins.

In our experiments, 48 images from a wide variety of sources such as newspapers, magazines, printed advertisements, photographs, and checks have been tested on the system. They are greyscale images with structured and non-structured layouts and a wide range of font styles (including certain script and hand-written fonts) and sizes. In practice, we have found that font sizes do not affect the performance of the system. Text overlapping background texture patterns are also successfully extracted.

There are 22030 characters in the test images that are perceivable to one of the authors. Over 93% of these characters have been successfully detected by the system. More than 96% of the detected characters are successfully cleaned up and binarized. Out of some 166604 characters and 3304 words of extracted text which are of OCR-readable fonts, 94% of the characters and 88% of the words are successfully recognized by a commercial OCR system. On average, the false positive rate is about 30% with respect to the area of text regions, or less than 5.3% of the image size.

The system is stable and robust, with all the system parameters remaining the same throughout all the experiments.

12 Acknowledgments

We would like to thank Bruce Croft and CIIR for supporting this work. Adam Jenkins helped port the code to a standalone version while Jonathan Lim provided system support. We would also like to thank Allen Hanson, and Yong-Qing Cheng for their constructive comments and suggestions.

A Appendix: Notes on Implementation

In an attempt to help readers who may be interested in implementing this system, a high level description of the modules of the system is presented in this section, together with user adjustable parameters. Readers should refer to Figure 2 for a block diagram of the system.

A.1 Texture Segmentation

Given an input greyscale image, it is segmented as follows:

1. Filter the image with $G_{xx}(x, y, \sigma)$, $G_{yy}(x, y, \sigma)$ and $G_{xy}(x, y, \sigma)$ where $G(x, y, \sigma)$ is the Gaussian filter with standard deviation σ. In order to speed up the process, it suffices to use only $\sigma = \sqrt{2}$.

2. Transform each of the output images I_i in the previous step by T:

   $T(I_i(x, y)) = (tanh(\alpha I_i(x, y)))^2$

   where α is set to 0.25.

3. For each output image T_i in the previous step, compute the local energy for each pixel by summing up the values in a square window centered at that pixel. The window width w is set to the Gaussian kernel size plus 5.

4. Normalize each output image in the previous step so it has 0 mean and unit standard deviation. Each output image represents the values of one feature.

5. Cluster the feature vectors and segment the input image as described in section 4.

6. Apply morphological closure operation on the "text" segments. This is done by dilating the image four times with the following kernel

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

   and then eroding the output with the same kernel four times. The output is used in the Chip Generation phase (Stroke Filtering and Aggregation steps).

There is no need for a user to adjust any parameters in this module.

A.2 Chip Generation Module

The Chip Generation module consists of Stroke Generation, Stroke Filtering, Stroke Aggregation, Chip Filtering and Chip Extension submodules executed in the order listed.

A.2.1 Stroke Generation

- Convolve the input image with the Gaussian $G(x, y, \sigma)$, where $\sigma = 1.0$. Denote the output as S;
- Convolve S with the kernel of

  $$\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

  Denote the output as D;
- Threshold D. A pixel (x, y) is set to foreground if and only if $|D(x, y)| \geq \tau$ where τ is an user adjustable parameter. $\tau = 10$ is a reasonable choice.
- Compute connected components of the thresholded image. Each component is called a "stroke".

It should be pointed out that other global or adaptive thresholding methods, such as gradient magnitude thresholding, may also be used for this purpose. In this case, the threshold value can be set relative to the maximum gradient magnitude. For example, a pixel is set to foreground if its gradient magnitude is at least 10% of the maximum gradient magnitude.

A.2.2 Stroke Filtering and Aggregation

- Remove the strokes if less than τ_{seg} percent of their foreground pixels coincide with the foreground pixels of texture segmentation. Denote the remaining strokes as S.

  τ_{seg} is a user parameter. $\tau_{seg} = 50$ works well in the experiments.

- Compute the equivalent classes of S of the following relation R:

  - For all $s \in S$, $(s, s) \in R$;
  - Let $s_1, s_2 \in S$, $s_1 \neq s_2$. Then, $(s_1, s_2) \in R$ if and only if s_1 and s_2 are connectable and there is a path between them with length less than s_d times the height of the taller stroke;
  - If $(a, b) \in R$ and $(b, c) \in R$, then $(a, c) \in R$.

- Compute the chips. Each equivalent class containing more than one stroke forms a chip. The minimal bounding box is the smallest rectangular box which contains all the foreground pixels of the strokes in the class.

s_d is a user parameter which should be selected to correspond to the normal space between two characters in a word. In most cases, this space is less than twice the height of a character, so $s_d = 3$ was used.

The other two user parameters are the height ratio , h_r, of two strokes with similar height, and the overlap, y_{ov}, of the Y-projections of two horizontally aligned strokes. See section 5.2 for the definitions of these parameters.

A.2.3 Chip Filtering and Extension

- Remove the chips whose height is less than c_h, or whose width is less than c_w, or whose minimum aspect ratio (width/height) is less than c_{ar}.
- Extend the remaining chips as described in section 5.5.

c_h, c_w and c_{ar} are user parameters. 7, 15 and 0.9 were used respectively in the experiments.

A.3 Chip Scale Fusion

- Project the chips from any resolution to the original resolution by scaling them appropriately.
- Combine the magnified chips with the chips at the original resolution level.
- Remove the redundant chips as described in section **??**.

The last step can be ignored if the chips are cleaned and binarized in descending order of size.

A.4 Text Clean-up

- Binarize the chips using the algorithm described in [23].
- Compute the connected components of the binarized region.
- Store the connected components.

A.5 Chip Refinement

The Chip Refinement module is essentially the same as the Chip Generation module with the following exceptions:

- The strokes are now the connected components generated using the Text Clean-up algorithm.
- The texture segmentation result is no longer needed in stroke filtering and aggregation (equivalently, the text region comprises the whole input image).
- Chip extension was not necessary.

Since this step removes false positive chips while retaining/improving the right ones each time it is applied, it can be applied more than once to remove more false positives.

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 Technologies. *Proceedings of The IEEE*, 80(7):1066–1078, July 1992.

[3] K. Etemad, D. Doermann, and R. Chellapa. Multiscale Segmentation of Unstructured Document Pages using Soft Decision Integration. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 19(1):92–96, Jan. 1997.

[4] 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.

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

[6] Anil K. Jain and Sushil Bhattacharjee. Text Segmentation Using Gabor Filters for Automatic Document Processing. *Machine Vision and Applications*, 5:169–184, 1992.

[7] Anil K. Jain and Bin Yu. Automatic Text Location in Images and Video Frames. *Pattern Recognition*, 31(12):2055–2076, 1998.

[8] Anil K. Jain and Yu Zhong. Page Segmentation Using Texture Analysis. *Pattern Recognition*, 29(5):743–770, 1996.

[9] Daniel Lopresti Jiangying Zhou and Tolga Tasdizen. Extracting Text from WWW Images. *Proc. of SPIE'98 Document Recognition V*, pages 130–138, Jan. 1998.

[10] 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.

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

[12] 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.

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

[14] Ramakant Nevatia. A Color Edge Detector and Its Use in Scene Segmentation. *IEEE Transactions on System, Man, and Cybernetics*, SMC-7(No. 11):820–826, Nov. 1977.

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

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

[17] 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.

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

[19] M. A. Smith and T. Kanade. Video Skimming and Characterization Through the Combination of Image and Language Understanding Techniques. *Proc. of the IEEE CVPR '97*, pages 775–781, June 17 - 19 1997.

[20] 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.

[21] 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.

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

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

[24] Victor Wu and R. Manmatha. Document Image Clean-up and Binarization. *Proc. of SPIE'98 Document Recognition V*, pages 263–273, January 1998.

[25] Victor Wu, R. Manmatha, and Edward. M. Riseman. Finding Text In Images. *Proc. of the 2nd intl. conf. on Digital Libraries. Philadaphia, PA*, pages 1–10, July 1997.

[26] Y. Zhong, K. Karu, and Anil K. Jain. Locating Text in Complex Color Images. *Pattern Recognition*, 28(10):1523–1536, October 1995.