Finding Text In Images [1]

Victor Wu, R. Manmatha, Edward M. Riseman
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610
{vwu, manmatha}@cs.umass.edu
January, 1997

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 readable 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 recognition technology 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 same 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]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.

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, 15] is largely restricted to finding text printed against clean, uniform 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 ([23]) 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, 12, 13, 24, 25, 26]; some assume specific document layouts such as newspapers [11] and technical journals [16]; others utilize domain-specific knowledge such as mail address blocks [7, 19] 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[2]. 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 OCR-readable fonts.

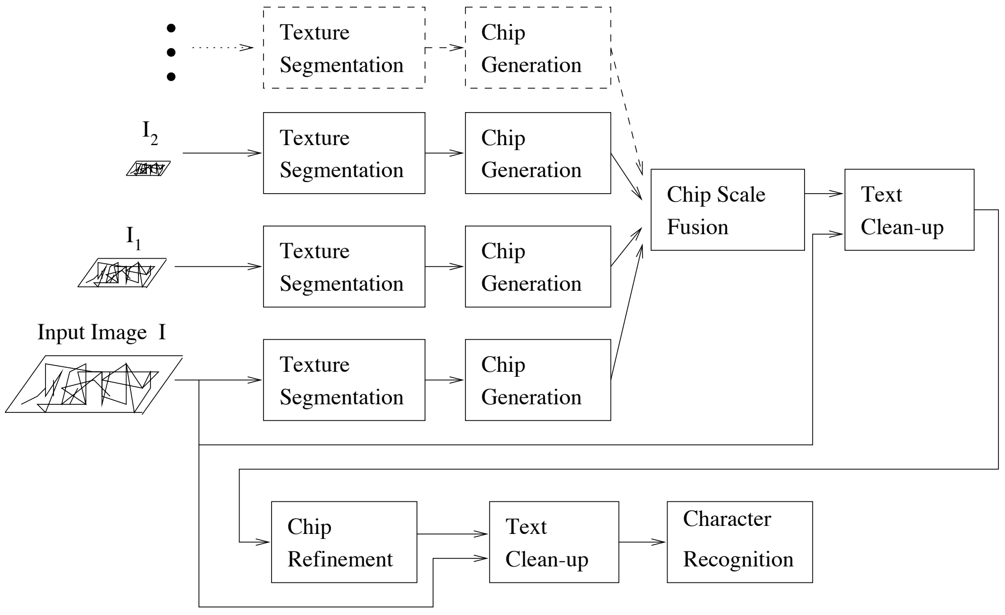
1.1 Motivation

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

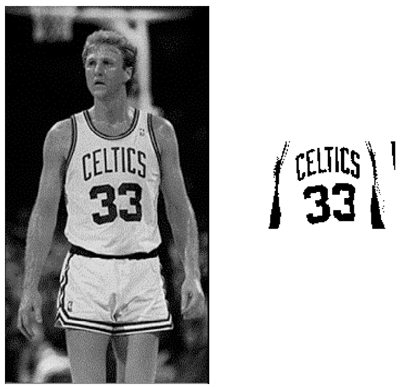
1. Text found in images or videos can be used to annotate and index those materials. For example, video sequences of events such as a basketball game can be annotated and indexed by extracting a player's number, name and the name of the team that appears on the player's uniform (Figure 1(b, c)). In contrast, 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.

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



(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.

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 ([6]) 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 has been used to convert text in scanned paper documents into ASCII symbols. A typical OCR system does this by first binarizing the document using a global threshold to separate text from the background. The binarized text is then segmented into individual characters, and lastly these segmented characters are recognized and converted. A comprehensive historical overview of OCR research and development is given by Mori, Suen and Yamamoto [15] . The techniques used in a commercial OCR system are discussed in detail in Bokser's paper [2].

However, text is often printed against shaded or hatched backgrounds in documents such as photographs, maps, financial documents, engineering drawings, and commercial advertisements. Furthermore, documents are often scanned in greyscale or color to preserve details of the graphics and pictures which often exist along with the text. In these cases the document images need to be segmented to identify the text regions (*segmentation*), and then cleaned up until the text stands out. As noted by many researchers ([18], [21]), global thresholding is usually not possible for these complicated images. Thus, more sophisticated page segmentation and binarization schemes are needed before OCR technology can be reliably applied .

Many document segmentation methods have been proposed in the literature. Some are top-down approaches, some are bottom-up schemes, and others are based on texture segmentation schemes in computer vision. In top-down approaches, a page is segmented by splitting large components into smaller components. For example, a page may be divided into column(s) of text blocks, then each block may be split into paragraphs, each paragraph may be split into text lines, and so on. Classic top-down techniques are based on the run length smoothing (RLS) algorithm ([24],[26]), which examines white (background/OFF) space between two black (foreground/ON) pixels in the same line (either horizontal or vertical) and sets it to black if the length is less than a preset threshold. Depending on the threshold, this operation can merge a character, a word, a text line and even multiple text lines into an unrecognizable blob. Then, horizontal and vertical projection profiles [25] are commonly used to cut the page into smaller blocks. A projection profile is a histogram of the number of ON pixels accumulated along parallel sample lines in an image. These sample lines can be of any orientation. For a document which consists of paragraphs of horizontal text lines printed in multiple columns, the horizontal projection profile will have alternating plateaus and valleys. The width of a plateau equals the height of the corresponding text line while the width of a valley equals the corresponding interline spacing. The vertical projection profile, however, will have wide valleys corresponding to the spacing between columns. Pavlidis and Zhou [20] use run-length-smoothing and projection profiles to examine both foreground and background for segmentation. Recursive projection profile cuts are used by Nagy et al [16], Wang and Srihari [25] to segment a page into blocks corresponding to headlines and paragraphs.

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. For example, O'Gorman's docstrum method [17] uses bottom-up k-nearest-neighbors clustering to group characters into text lines, and text lines into structural blocks. It starts out by finding k nearest neighbors for each connected component, where $k = 4$, or 5 are normally used. Then, two histograms, one for the angles and the other for the distances of all the connections, are compiled. Since most of the connections will be made between characters on the same text line, the peak angle will indicate the skew of the page and the peak distance will indicate inter-character spacing. Using these estimates, characters of the same orientation are grouped into words, and words into text lines. Fletcher and Kasturi [4] use a Hough transform, and the heuristic that characters in a text string of any orientation are collinear, to group characters into words and phrases.

The third category of document segmentation methods treat text as a type of texture and hence use texture segmentation algorithms to detect text. Jain and Bhattacharjee [8], for example, segment text using the standard multi-channel filtering technique for texture segmentation. In this case, the input image is filtered by a bank of Gabor filters. Then features of local energy estimates at each pixel are computed over each filtered image. These features are organized into feature vectors and then classified using an unsupervised clustering algorithm into a fixed number of clusters. One of the clusters corresponds to text.

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 Jain and Bhattacharjee's texture segmentation scheme can in principle be applied to greyscale images, it was only used on binary document images, and it is not shown whether text printed against shaded/textured backgrounds can be segmented. Furthermore, they did not address the binarization problem at all.

Many methods have been proposed to binarize greyscale images so that the pixels belonging to the objects of interest (e.g., text and/or graphics) are set to one intensity while the rest are set to another different intensity. Sometimes, global thresholds are used. Many methods has been proposed to find these global thresholds [5, 22]. O'Gorman [18] proposed a method which handles greyscale images with few variations in intensity. However, it is often the case that global thresholds do not exist, due to the nonuniformities within foreground and background regions. Different thresholds have to be used for different local regions (*adaptive thresholding*). Techniques for adaptive thresholding can be found in Kamel and Zhao's paper [11]. Trier and Taxt [21] report an evaluation of eleven local adaptive thresholding schemes.

The point we want to make here is that although a considerable amount of work has been done in 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, or text printed over shaded/textured backgrounds. These kinds of documents are often found in commercial advertisements, stock certificates, photographs and maps. The system presented in this paper is our contribution to filling the gap in this area of research and development, and to constructing a completely 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 a **Texture Segmentation** procedure to segment the text (Section 2). This algorithm is based on the standard multi-channel filtering techniques in texture segmentation [8, 14]. It should be pointed out that the standard 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 use the segmented regions as the focus of attention, and then apply a set of appropriate heuristics to find text strings within 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 sizes vary 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** in Section 4). 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 binarize the corresponding region in the input image so that the text stands out. A simple, effective histogram-based algorithm is also proposed which finds the threshold value automatically for each text region. It also cleans up the text. This algorithm is used for the **Text Clean-up** module in the system (Section 5).

Non-text items might survive the previous processing and occur in the binarized output. Thus, a **Chip Refinement** (Section 6) 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 Text 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. In principle, the font sizes do not matter. The system is also stable and robust—all the system parameters remain the same for all of the test 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. Overall, the algorithms have been tested on images which have the following characteristics:

1. binary and greyscale images.
2. structured and non-structured page layout.
3. wide range of character sizes (font heights vary from 10 to 160 pixels, or even higher).
4. large range of font styles including certain script and handwritten fonts.
5. text overlapping background textures, e.g., hatched or shaded background, drawings, and other non-text content.

A detailed description of the experiments is presented in section 8.

It is also worth noting that the segmentation algorithm can segment text at any orientation, although currently the rest of the algorithms handle only horizontal text strings.

2 The Texture Segmentation Module

As stated in section 1.3, in principle text can be treated as a specific texture. Thus, one natural way to detect text is by using texture segmentation techniques. A standard approach to texture segmentation is to first filter the image using a bank of linear filters 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 ([9],[14]).

Gaussian derivatives have often been used for texture segmentation ([14]). A symmetric 2D Gaussian function is given by

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} exp\{-\frac{x^2+y^2}{2\sigma^2}\} \tag{1}$$

The second-order derivatives of $G(x,y,\sigma)$ are:

$$G_{xx} = \frac{\partial^2 G(x,y,\sigma)}{\partial x \partial x}, \qquad G_{xy} = \frac{\partial^2 G(x,y,\sigma)}{\partial x \partial y}, \qquad G_{yy} = \frac{\partial^2 G(x,y,\sigma)}{\partial y \partial y} \tag{2}$$

These Gaussian derivatives are bandpass filters. Furthermore, G_{xx} provides information about the horizontal orientation, G_{xy} about the two diagonal directions, and G_{yy} about the vertical orientation. In other words, G_{xx}, G_{xy}, and G_{yy} can be used to approximate Gabor filters with low center frequency at orientations $\theta = 0, \pi/4, \pi/2, 3\pi/4$. It has been reported that text does fall into the low frequency channels ([8]). Another attractive property of these Gaussian derivatives is that they are separable filters which reduces the cost of computation significantly.

It should be pointed out that the frequency- and orientation-selective Gabor filters have often been used for texture segmentation. Details can be found in [3] and [9].

2.1 Segmenting Text

By treating text as a distinctive texture, we propose a text segmentation algorithm which uses second-order Gaussian derivatives G_{xx}, G_{xy} and G_{yy} as filters. This is similar to the approach proposed by Jain and Bhattacharjee ([8]) except that Gabor filters were used in their algorithm. Another distinction of this algorithm is that a simple calculation can be used to identify the "text" class, so that no training is needed. Lastly, a morphological closure operation is applied to the text class to fill the holes in the text regions.

Again, it should be pointed out that texture segmentation alone is not sufficient to solve the text detection and extraction problem, especially if the input images have complicated texture patterns. However, the segmentation may be used as a means of focusing attention for further processing. The details are discussed in section 3.

The following is an abstract description of the algorithm used in the texture segmentation module of the system, followed by a detailed discussion of each step.

1. Filter the input image by bands of second-order Gaussian derivative filters of varying σ to obtain n filtered images. Each band consists of three filters G_{xx}, G_{xy} and G_{yy}.

2. Apply the non-linear transformation, $tanh(\alpha t)$, to the filtered images.

3. For each image generated by step 2, compute the corresponding *feature image* which consists of a local energy estimate over a w by w window centered at each pixel.

4. Cluster the feature vectors corresponding to each pixel using the K-means clustering algorithm [10] where K is the number of clusters expected in the data.

5. Compute a segmented image. A pixel is set to ON if and only if its feature vector belongs to the cluster identified as "text".

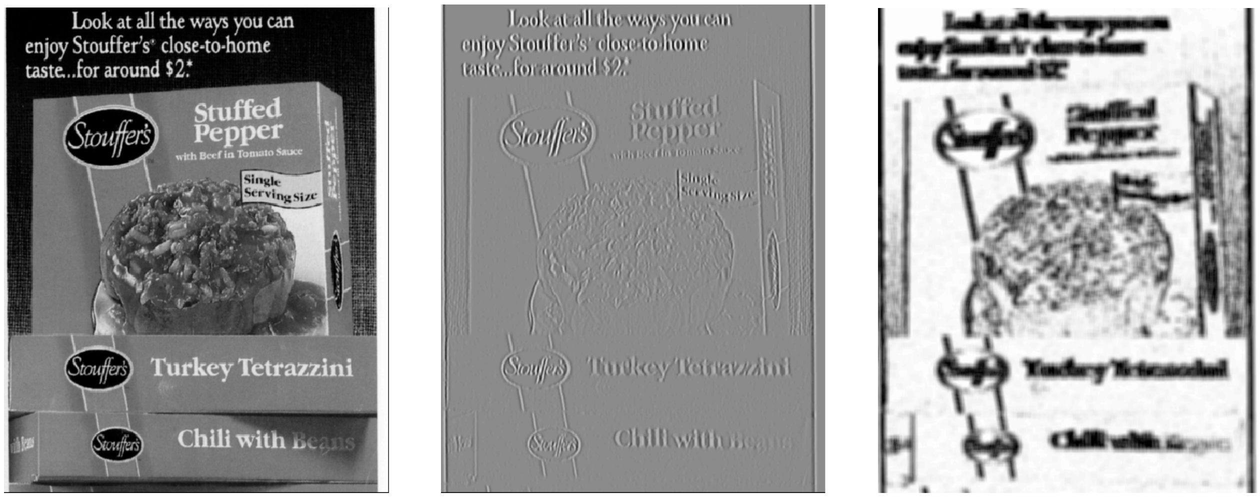
6. Apply the morphological closing operation (discussed later) to the "text" region.

Experiments have shown that using $\sigma \in \{1, \sqrt{2}, 2\}$ (i.e., 3 bands, 9 filters) works well for segmenting text whose font heights are less than 40 pixels. Note that the σ's are spread half an octave apart. The justification for using frequency bands which are half octaves apart comes from studies of biological visual systems . The σ's were also chosen so that the corresponding bands mostly overlap the bandwidth used by Jain and Bhattacharjee [8] for text segmentation.

The filtering operation is performed by first convolving the image with the Gaussian mask, and then convolving the result with one of the masks below respectively:

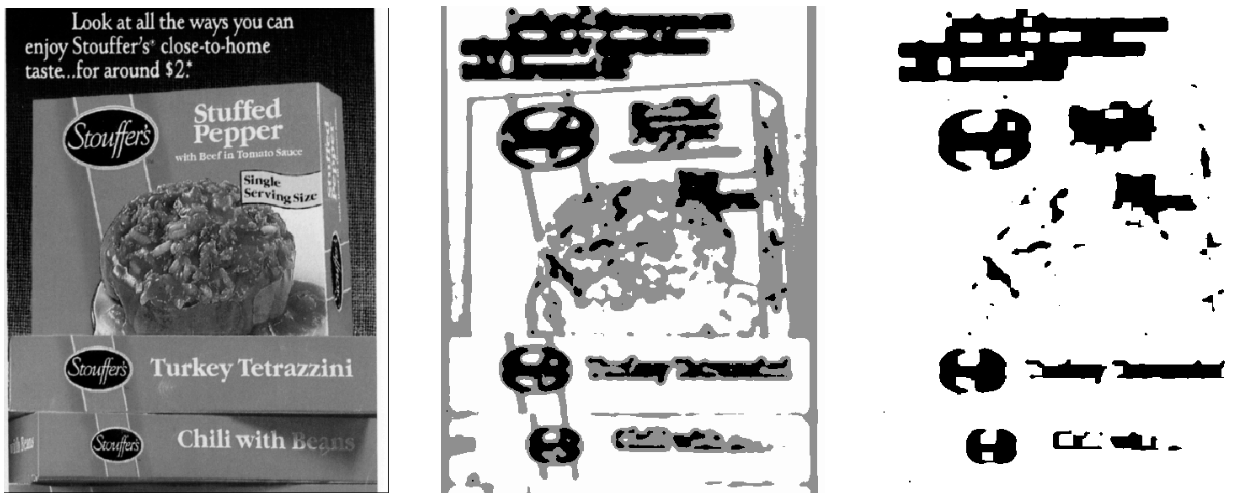
$$[-1, 2, -1], \quad \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

Figure 2(a) shows a portion of an original input image. This will be a running example for the rest of the paper. Only a 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



(a) (b) (c)

Figure 2: Texture feature computation. **(a)** Portion of an input image. **(b)** Filtered output of (a) using $G_{xx}, \sigma = 1$. **(c)** Normalized average energy of each pixel in (b)



(a) (b) (c)

Figure 3: Results of Texture Segmentation. **(a)** Portion of an input image. **(b)** Clustered output of (a). Dark is labeled as "text". **(c)** Text cluster after the morphological closure operation.

background, text printed on Stouffer boxes, Stouffer's trademarks (in script), and a picture of the food. Figure 2(b) is a filtered output of the input image using G_{xx} where $\sigma = 1$. The dark/light intensities indicate strong responses while grey indicates weak response.

The energy averaged over a $w \times w$ window W_{xy} centered at each pixel (x, y) is used as a feature for that pixel (see Jain and Bhattacharjee [8]). More specifically, the k^{th} feature image $e_k(x, y)$, corresponding to the k^{th} filtered image, denoted as $g_k(x, y)$, is computed using

$$e_k(x,y) = \frac{1}{w^2} \sum_{(a,b)\in W_{xy}} (tanh(\alpha g_k(a,b)))^2, \qquad k = 1, 2, ..., n$$

where n is the number of filters ($n = 9$ in this case). The size of the window W_{xy} is chosen to be the same size as the corresponding Gaussian filter mask. α is set to 0.25 through out our experiments.

The values in the 9 feature (energy) images corresponding to a given pixel form a feature vector for that pixel. To prevent a feature with higher numerical value from dominating other features, each feature is normalized to have zero mean and unit standard deviation. The normalized feature of Figure 2(b) is shown in Figure 2(c) (the darker the pixel, the higher its normalized feature value). Notice that the text is prominent in this figure.

The next step is to use the K-means clustering algorithm to cluster the feature vectors. It should be noted that selecting the number of clusters (value of K) is a non-trivial problem. Empirically, it was found that three clusters ($K = 3$) worked well with all the test images. The clustering algorithm is known to be computationally expensive, hence, in order to reduce the cost of computation for large images, a small amount of pixels were picked randomly and their feature vectors were clustered. Then, a minimum distance classification was carried out over the entire feature vector set using the cluster centers of the sampled feature vectors. It was observed that 8000 sample pixels were sufficient for the larger images in the test set — no noticeable degradation of the clustering output occurred.

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 3(b) shows the pixel labels after the clustering of the feature vectors of the test image shown in Figure 3(a). The dark areas correspond to pixels labeled as text, and white areas correspond to pixels labeled as background. The other cluster is marked grey. Grey pixels have some energy, but probably not enough to be 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, the following morphological closure operation on the text regions is proposed: *dilate the binary image corresponding to the text cluster with a* 3×3 *mask four* *times, then erode its output with the same mask four times.* The output of this operation is called **segmented text regions**. Figure 3(c) shows the result of this operation carried out on the text regions shown in Figure 3(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 must be expected to pass the filters and be misclassified as text, as shown in Figure 3. 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 as one region, one intensity threshold might not separate both 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 to find 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[3].

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]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.

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 covers text strings more completely.

A detailed discussion of these steps is presented in the following corresponding subsections.

3.1 Stroke Generation

The purpose of this step is to produce strokes, which are connected edges with significant contrast. As stated before, text must have significant contrast in order to be readable. In other words, the edges of characters can be expected to have significant contrast. Based on this assumption, the following simple edge-detection procedure is used to produce edges, followed by connected-components computation to group edges into **strokes**:

1. Convolve the original input image with a second-order Gaussian derivative G_{xx}.

2. Threshold the image generated by the last step.

3. Generate connected components with each connected component of ON pixels forming a distinct stroke.

It is desirable to use a large σ for the Gaussian filter to smooth out the undesired details and noise in the image which usually have little to do with text. However, if σ is too large, the image will be blurred too much so that edges of some of the text may be suppressed. Empirically, $\sigma = 1$ was found to be a reasonable choice.

The thresholding is carried as the following: for each pixel, set it to ON if the absolute value of the pixel is greater than τ where τ is a positive threshold. Otherwise, set it to OFF. It is clear that the threshold τ should be small so that most edges of all the characters are to be found. The consequence of this strategy is that edges of non-text items are found as well (see Figure 4(b)). However, most of those false positive strokes will be eliminated by the processes that follow. Experiments have shown that $\tau = 10$ worked well for all text images. Figure 4(b) shows the strokes of Figure 4(a).

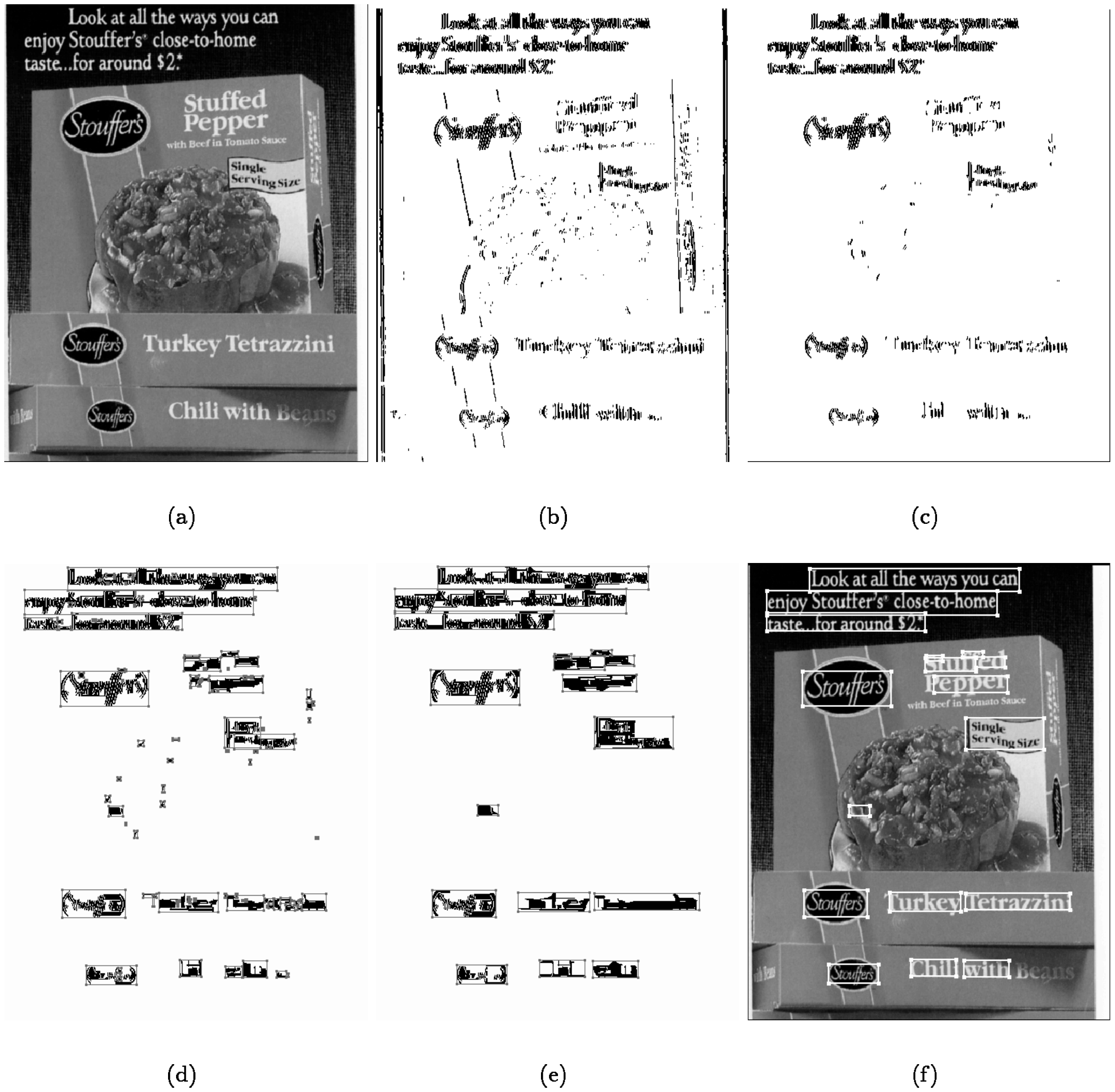
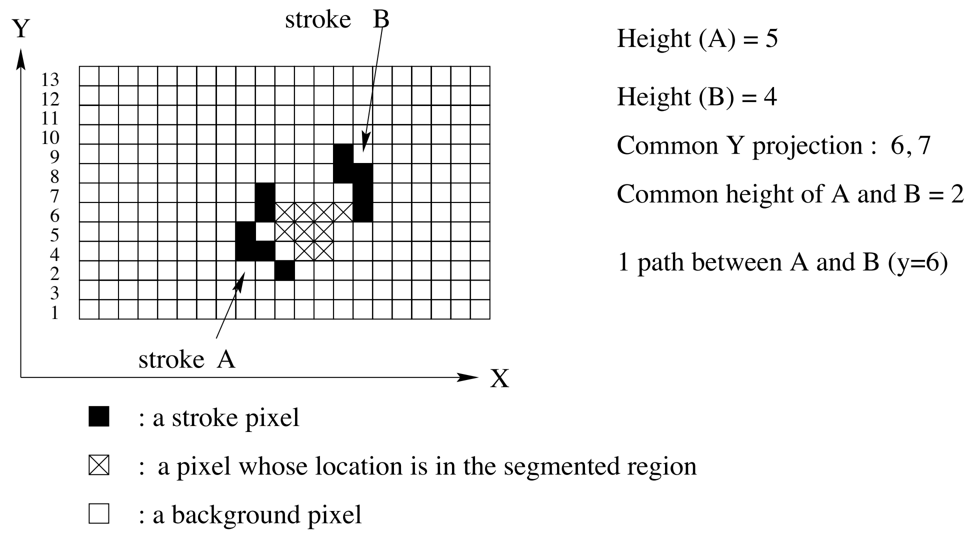


Figure 4: Results of Chip Generation. **(a)** Portion of an input image. **(b)** Strokes produced by performing the Stroke Generation procedure on (a). **(c)** Strokes after applying the Stroke Filtering step on (b). **(d)** Chips produced by applying Stroke Aggregation on strokes in (c). **(e)** Chips after the Chip Filtering and Extension processes. **(f)** Chips in (e) mapped to the input image.

3.2 Stroke Filtering

As one can clearly see in figure 4(b), there are strokes appearing 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.

The purpose of Stroke Filtering is to eliminate the false positive strokes by using heuristics which



Y
stroke B
13
12
11
10
9
8
7
6
5
4
2
3
1
stroke A
X
Height (A) = 5
Height (B) = 4
Common Y projection : 6, 7
Common height of A and B = 2
1 path between A and B (y=6)
■ : a stroke pixel
⊠ : a pixel whose location is in the segmented region
□ : a background pixel

Figure 5: Example of the stroke definitions. White pixels form background while dark pixels are part of the strokes. The pixels belong to the segmented region are marked using crosses.

take into account the fact that neighboring 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 the following definitions:

Definition 1 *Two strokes are of* **similar height** *if the height of the shorter stroke is no less than h_τ percent of the height of the taller stroke , where h_τ is a threshold.*

It is easy to see that the smaller h_τ is, the more height variation is allowed. Thus, this parameter controls the degree of height variation of characters in a text string that is acceptable. However, if h_τ is too small, long line segments near true text strokes, and/or strokes belonging to a significantly different text string or font might be grouped as part of the text string. This is not desirable.

Since it is usually the case that the heights of adjacent characters in the same text string do not vary more than twice the shorter height, $h_\tau = 40\%$ has been found to be effective for all test images. As an example, Figure 5 shows two strokes which are of similar height since the height of stroke B (4) is more than 40% of the height of stroke A (5).

An intuitive way to describe the horizontal alignment of characters is to project them onto the Y-axis and check how much the projections overlap. If the characters are aligned, their projections mostly overlap. Thus, the following definition is used to measure the alignment:

Definition 2 *Two strokes are* **horizontally aligned** *if the common portion of the projection of the strokes on the y-axis is no less than o_τ percent of the height of the shorter stroke, where o_τ is* *a threshold.*

The threshold o_τ dictates how much the text string can be off the horizontal orientation, or bend, or both. The smaller the value, the more the string can be non-horizontal or bend, but also prone to including strokes which are not related to the same text string. Experimental results have shown that $o_\tau = 50\%$ works well for all the test images. Thus, the strokes in Figure 5 are horizontally aligned since the common Y-projection of the strokes is 2 which is 50% of the height of stroke B (4).

As stated before, the focus is on finding text strings, not just characters standing alone. Thus, a text stroke should have similar strokes nearby which belong to the same text string. The question is to decide whether any two given strokes are likely to be from the same text string using reasonable heuristics. As explained in the definitions above, strokes belonging to the same text string can be expected to be of similar height and horizontally aligned. Also, since text is expected to occur in the segmented regions, the strokes belonging to the same text string can be expected to overlap with the same segmented region. These concepts can be expressed using **connectability** which is defined as:

Definition 3 *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).*

By definition 3, the two strokes in Figure 5 are also connectable since they are of similar height, horizontally aligned, and there is a path (at $y = 6$) between them.

Given the above definitions, the criterion used for stroke filtering can be simply stated as the following:

- 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 that the strokes are expected to overlap the segmented regions. In the ideal case, text strokes should be completely contained within the segmented text region. But the segmentation is often not perfect (e.g., the segmented regions may have inner holes and text strokes going through those holes will have pixels which do not belong to the segmented regions). Hence one cannot expect total overlap, or too many correct text strokes would be eliminated. 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.

Figure 4(c) shows the result of applying this procedure on the stroke in figure 4(b). Notice that most of the text is still present while some of the background has been eliminated.

3.3 Stroke Aggregation

An important function of 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 of the same text string are expected to be of similar height, horizontally aligned, and they are expected to occur in the segmented regions, 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. Since the width of a character and the spacing between adjacent characters in a text string are related to the heights of the characters, 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 shorter stroke.

Figure 4(d) shows the result of applying the Chip Generation procedure on the strokes in figure 4(c). Notice that most 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 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 a single character, the width of a chip is also used to filter out non-text chips. Lastly, for horizontally aligned text strings, their aspect ratios (width/height) are usually large. These constraints are incorporated in the following algorithm for filtering:

- compute the minimum bounding box for each chip

- compute the width, height and aspect ratio of each box
- a chip is eliminated if one of the following is true
  - the width of its box is less than cw_{τ}
  - the height of its box is less than ch_{τ}
  - 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. Thus, $ch_{\tau} = 1.1$ is used, attempting to cover that case to some extent.

3.5 Chip Extension

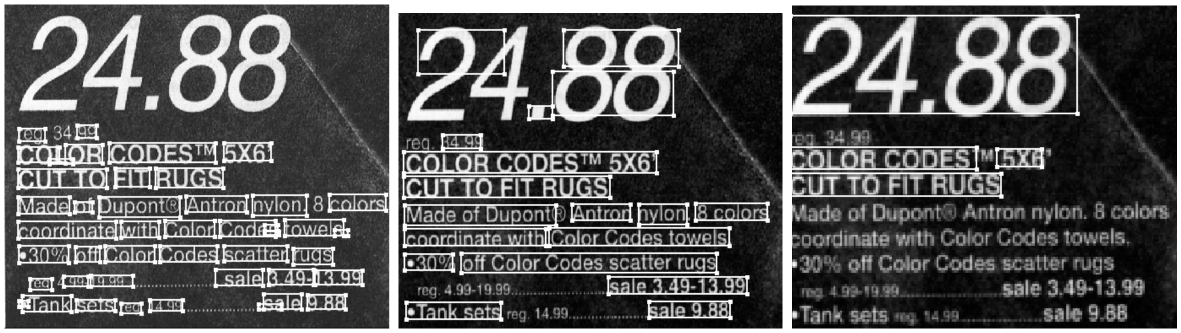
It is expected that some strokes only cover fragments of the corresponding characters. Therefore, these strokes might violate the constraints used for stroke filtering, and hence be eliminated. Consequently, some of the chips generated so far may only cover part of the corresponding text strings.

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 4(e) shows the result of applying the Chip Filtering and Extension steps on the chips in Figure 4(d). The rectangular chip bounding boxes are mapped back onto the input image to indicate detected text as shown in Figure 4(f).

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**). Figure 6 gives an example of the scale problem. Intuitively, the larger the font size of the text, the lower the frequency it possesses. Thus, when the text font size gets too large, its frequency falls outside the three channels selected in section 2.


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) (b) (c)

Figure 6: The scale problem and its solution. The text detections are carried out using the standard three frequency channels in all three cases with the same input image at different resolutions. **(a)** Input image at its full resolution. **(b)** The input image is that of (a) reduced by half in both dimensions. **(c)** The input image is that of (b) reduced by half in both dimensions.

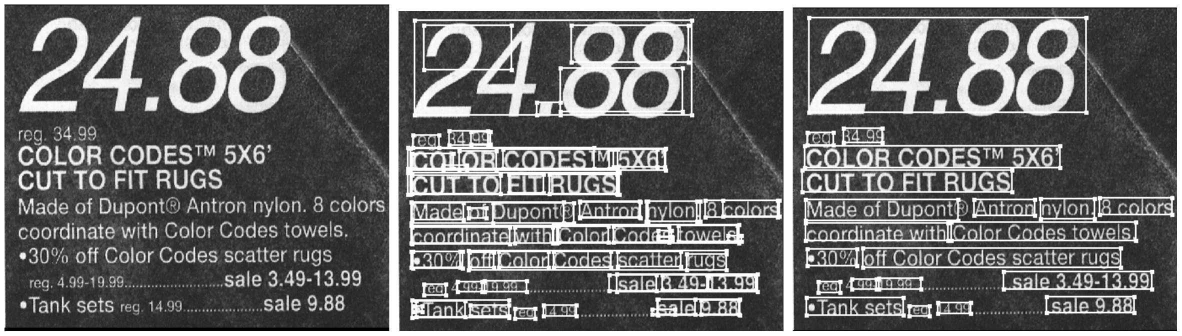
One approach to the scale problem is to process the original image at multiple bands of frequency channels. For example: band 1 covers the standard channels of second order Gaussian derivatives with $\sigma = 1, \sqrt{2}, 2$; band 2 for $\sigma = 2, 2\sqrt{2}, 4$, and so on. However, as the σ values increase, so do the corresponding kernel sizes, and thus the cost of the computation increases dramatically.

Instead, a pyramid approach is proposed in this paper: 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). 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 at the 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 6(a) while text of large font sizes is found using images higher in the pyramid as shown in Figure 6(c). The bounding boxes of detected text regions at each level are mapped back to the original input image (bottom level) as shown in Figure 7(b).

4.1 Chip Scale Fusion

Some text or part of it responds to more than one band of the frequency channels, and hence forms overlapping chips at different levels (see Figure 6). For example, text with large fonts may partially respond to higher frequency channels, hence forming fragmented boxes which covers parts of the string at a lower level. It also strongly respond to some lower frequency channels which are more compatible, hence forming a more complete box to cover the entire string at higher level(s). After the chips produced at different levels are mapped back onto the original image (bottom level), the


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) (b) (c)

Figure 7: Fusion of text detection at different level. (a) Original input image. (b) Chips generated at all three levels. (c) Scale-redundant chips are removed.

fragmented chips, which are called the **scale-redundant** chips, will be covered in full or in part by larger boxes, as shown in figure 7(b). Thus, it is desirable to eliminate the scale-redundant chips and keep the chips which overlap more with the text strings. We call this process **chip scale fusion**.

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. Experiments have shown that this is particularly the case when more than 85% of chip B overlaps with chip A. Another situation is that only an insignificant portion of B overlaps with A. In this case, if A is significantly larger than B, B is also likely a scale-redundant chip, especially when at least 50% of chip B overlaps with chip A and chip A is at least 10 times bigger than B (empirical finding). For example, the smaller boxes may cover small fragments of a text string of a large font size and some other local details which are not part of the text string. 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

As an example, Figure 7(c) is the result of the above scale fusion procedure applied to the chips in figure 7(b).

5 Text on Complex Backgrounds

The previous sections have described a system which detects text in images and puts boxes around detected text strings in the input image. If the input image is binary and the text is printed on a clean background, the corresponding region enclosed by the box can be directly fed to an OCR system for actual character recognition. However, documents are often scanned in greyscale rather than in binary to preserve information other than text, such as picture inserts and graphics. For example, all the images used so far have more than two levels of tonal variation. In addition, it is not uncommon for text to be printed against a shaded or hatched background. For example, shaded, very high frequency backgrounds are used in currencies and stock certificates to prevent copying. This is often done by using fine lines or many small dots (high frequency) for the shading.

Unfortunately, current OCR systems cannot handle such images as explained in section 1.2. Furthermore, in general the OCR recognition accuracy depends heavily on how well text is segmented into individual characters. Thus, the chips generated by the text detection phase of the system, have to be binarized and the shaded background removed.

Our goal here is to find a simple, robust binarization algorithm which is applicable to the text chips detected as described in the previous sections. By taking advantage of the localization of text strings achieved by the segmentation and chip generation procedures, it is reasonable to assume that a threshold exists for that local area. This is based on the observation that normally all the characters in a single string are likely to have similar intensities which are distinguishable from the non-text content of that area. This is a local adaptive approach, and experiments show that this binarization approach robustly extracts text from a wide variety of images.

More specifically, the following algorithm for background removal and binarization is proposed:

1. smooth the text chip generated by the system.

2. compute the intensity histogram of the smoothed chip.

3. smooth the histogram using a low-pass filter.

4. pick a threshold at the first valley counted from the left side of the histogram.

Histograms have been used in global thresholding algorithms proposed in the literature. The important distinction of this algorithm is that it is applied locally instead of over the whole image. The smoothing operation of step 1 affects the background more than the text because the text normally is of lower frequency than the shading. Thus it cleans up the background. Another way of looking at it is to notice that the smoothing blends the background into grey while leaving the text black (see the second row in Figure 8).

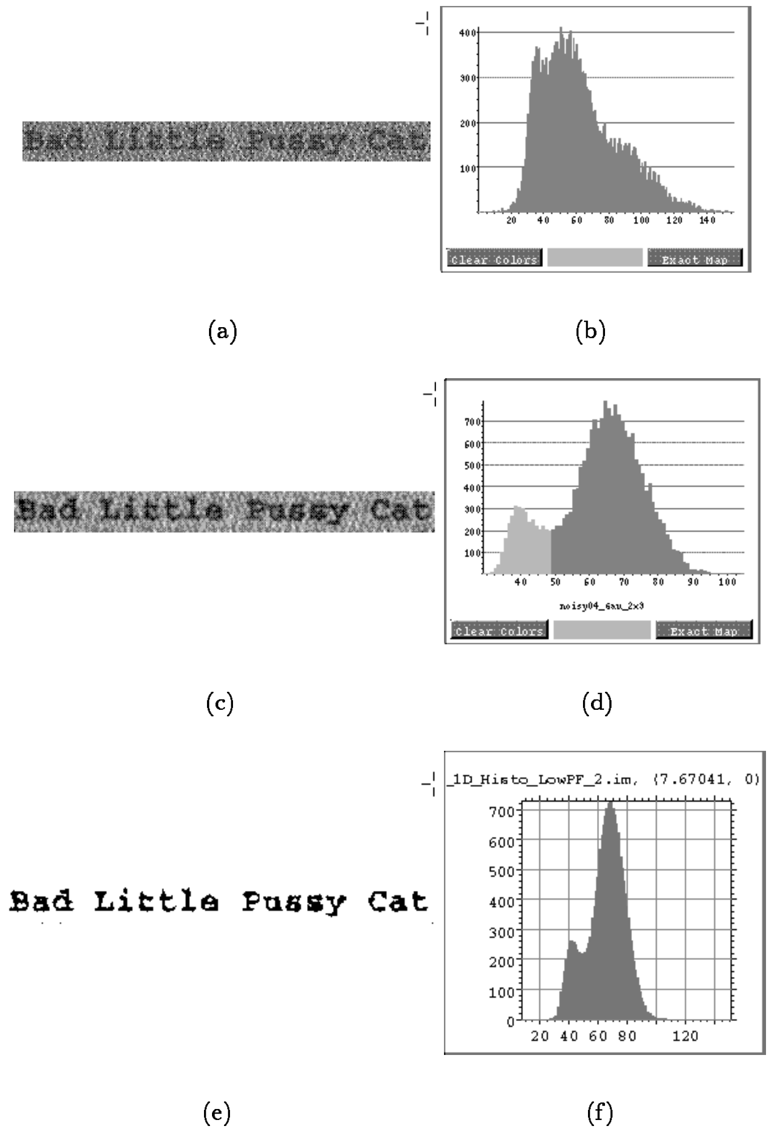
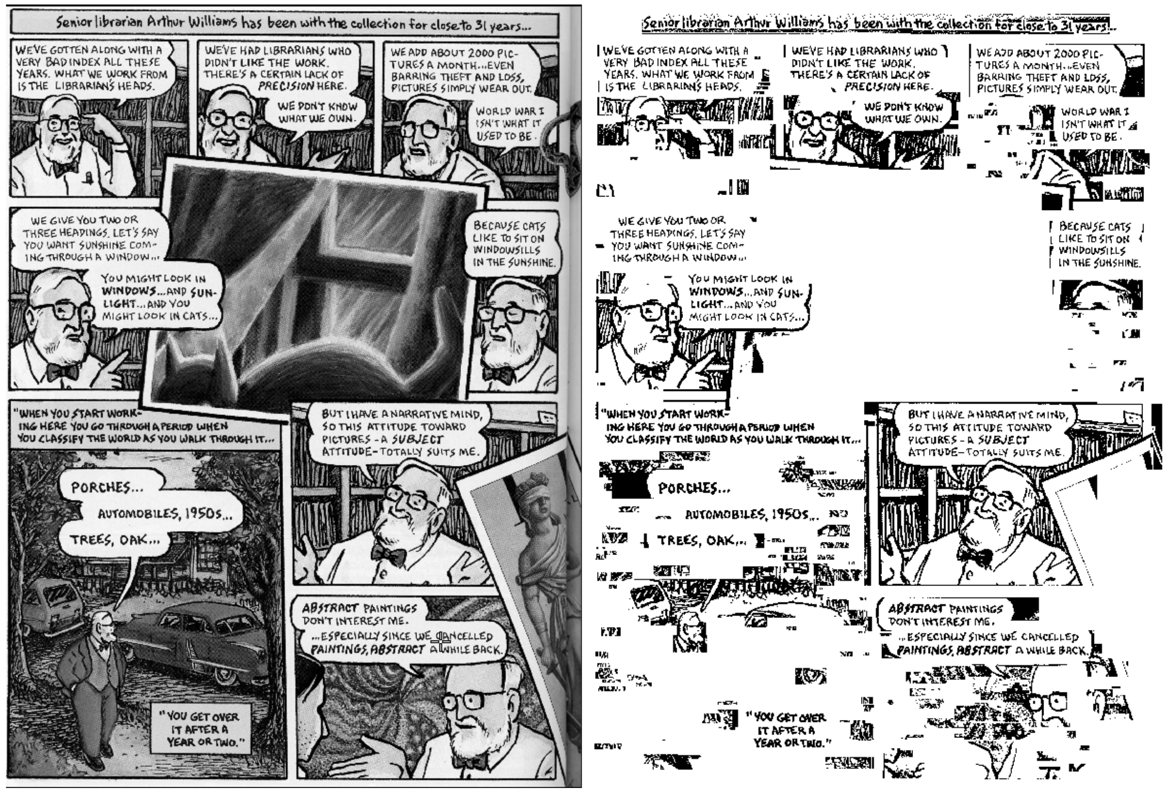


Figure 8: The Text Clean-up process. **(a)** Original text chip. **(b)** Histogram of (a). **(c)** Smoothed version of (a). **(d)** Histogram of (c). **(e)** The binarization result by thresholding (c) using a value in the valley of (f). **(f)** Smoothed version of (d).

The histogram generated by step 2 is often jagged, hence it needs to be smoothed to allow the valley to be detected (see Figure 8(d)). Text is normally the darkest item in the detected chips. Therefore, a threshold is picked at the first valley closest to the dark side of the histogram. To extract text against darker background, one simply reverses the intensity value of the chip before applying this algorithm. 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.

The thresholded image is shown at bottom left in Figure 8. This has been successfully recognized


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 LIBRARIANS' 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.

(a) (b)

Figure 9: First binarization result. (a) Original image scanned from New Yorker magazine. (b) Result after the first Text Clean-up process

by an OCR system.

6 Chip 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. An example is shown in Figure 9(b). This is produced by mapping the extracted items onto the original page. 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. Tighter constraints can be used here because of two reasons. First, the clean-up procedure is able to extract most characters without attaching to characters nearby or non-text items as shown in Figure 9(b). Second, 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 Generation step (section 3.1). Thus, it can be expected that the correct text strokes (characters) 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 more restricted constraint for connectability of two similar strokes is used at this stage. This constraint requires that the gap between two adjacent strokes 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.

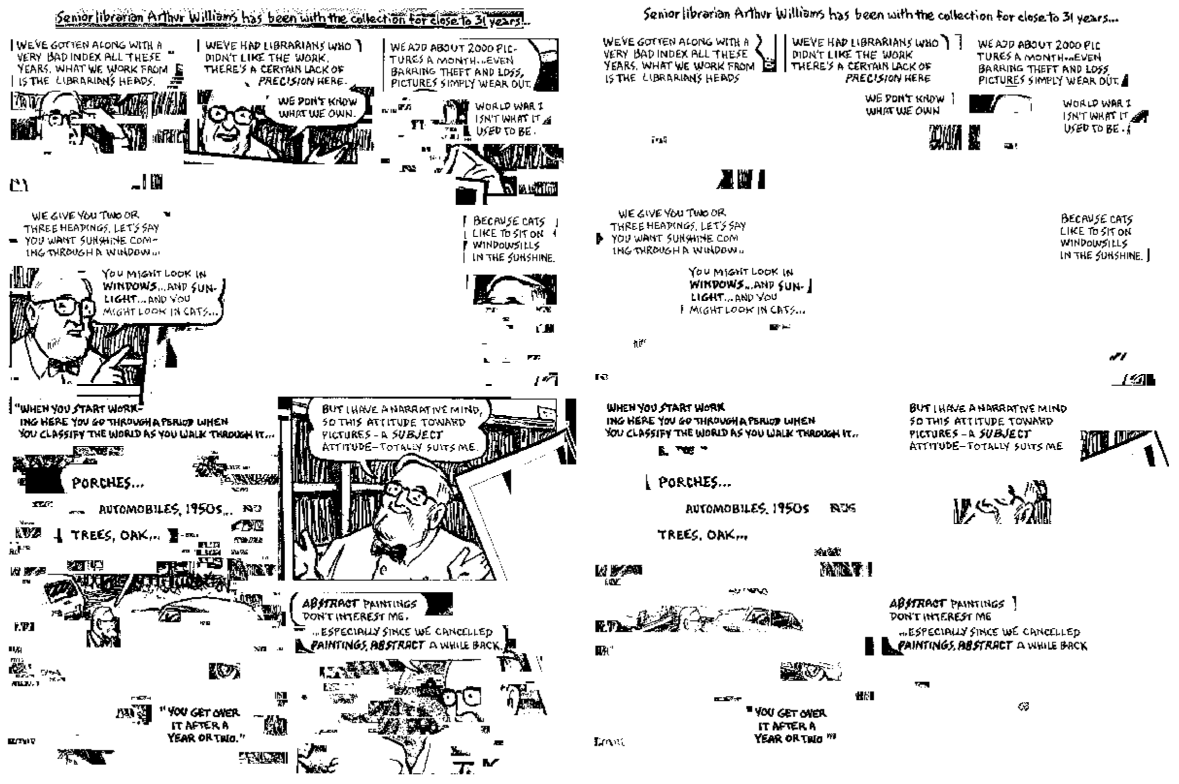
An example is given in Figure 10 which shows the binarization results before and after this refinement phase, where Figure 10(a) is the same picture as that in Figure 9(b) . A magnified portion of the image and its binarization results are shown in Figure 11.

7 Character Recognition

The cleanly extracted text strings produced after the Text Refinement phase are still in the image format, thus a character recognition process is needed to convert the text strings into the ASCII format. 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.

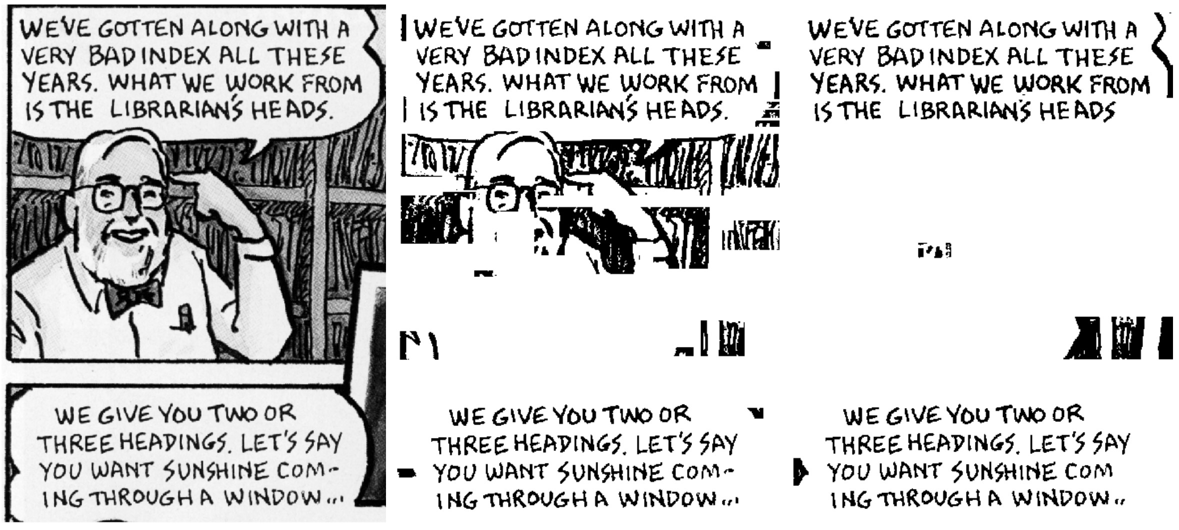
A good way of doing such experiment is to integrate the OCR system into the text extraction system so that the binarized text strings can be processed automatically when they are generated, one string at a time. Unfortunately, the OCR package that we have cannot be used in this way. Thus, for the experiments, a binary image is formed using all the cleaned-up text chips for each input image. Then these binary images are fed to the OCR system for recognition manually.

The OCR results are presented in the experiments section.


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 SUNLIGHT...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."

(a) Before (b) After

Figure 10: Comparison of binarization results before and after the Chip Refinement phase.


WE'VE GOTTEN ALONG WITH A VERY BAD INDEX ALL THESE YEARS. WHAT WE WORK FROM IS THE LIBRARIAN'S HEADS.
WE GIVE YOU TWO OR THREE HEADINGS. LET'S SAY YOU WANT SUNSHINE COMING THROUGH A WINDOW...

(a) (b) (c)

Figure 11: Magnified portions of Fig. 9 and Fig. 10.

File Name	Total Char	Detected Char	Cleaned Char	Total Words	Detected Words	Cleaned Words	Split Words
ads01	167	154	154	38	33	33	5
ads02	221	204	204	48	43	43	2
ads03	206	205	183	45	44	37	1
ads04	722	702	659	151	145	132	4
ads05	273	250	238	68	64	61	1
ads06	377	227	227	62	29	29	18
ads07	131	131	111	27	27	23	0
ads08	132	83	83	23	14	14	4
ads09	29	29	26	4	4	3	0
ads10	259	251	247	61	53	53	3
ads11	402	369	358	71	63	61	2
ads12	295	294	292	64	63	63	0
ads13	417	417	385	150	150	142	0
ads14	2143	1852	1810	424	373	256	5
ads15	394	394	380	88	87	84	1
ads16	487	329	157	104	67	34	13
ads17	298	295	270	56	55	41	0
ads18	458	456	356	96	95	68	1
che05	273	273	273	47	47	47	0
che06	273	272	273	47	46	47	1
che07	273	273	273	47	47	47	0
cpn01	742	736	712	152	148	147	2
ctn01	2063	1945	1945	417	387	386	0
ctn02	543	540	540	101	96	96	2
ctn03	300	296	296	64	61	61	3
ctn04	813	813	812	186	186	185	0
ctn05	928	928	928	196	196	196	0
env01	160	156	156	28	27	27	1
inv01	1355	1293	1188	285	264	233	7
inv02	579	578	500	133	132	113	0
mag01	817	817	812	166	166	163	0
mag02	91	87	87	19	18	18	0
npr01	124	124	124	26	26	26	0
npr02	772	768	767	173	169	168	0
pho01	11	5	4	2	0	0	1
pho02	7	6	5	2	1	1	0
pho03	2	0	0	1	0	0	0
pho04	9	9	9	2	2	2	0
pho05	39	8	0	8	0	0	5
pic08	28	28	13	6	6	3	0
pic02	21	21	19	7	7	2	0
pic04	31	26	17	6	5	3	1
pic05	12	5	0	5	3	0	0
pic06	31	31	6	5	5	1	0
pic07	33	30	20	9	7	2	1
pic01	1217	1215	1177	250	249	244	1
txt01	2009	2008	2008	234	233	233	0
txt02	968	960	960	199	192	192	2

Table 1: Results for the text extraction

8 Experiments

The system has been tested using 48 images. Some of the test images were downloaded from the Internet (file names "pho02-04" in table 1), some from the Library of Congress (file names "pic02-06" and "pic08" in table 1), and the rest of them 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, etc. 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.

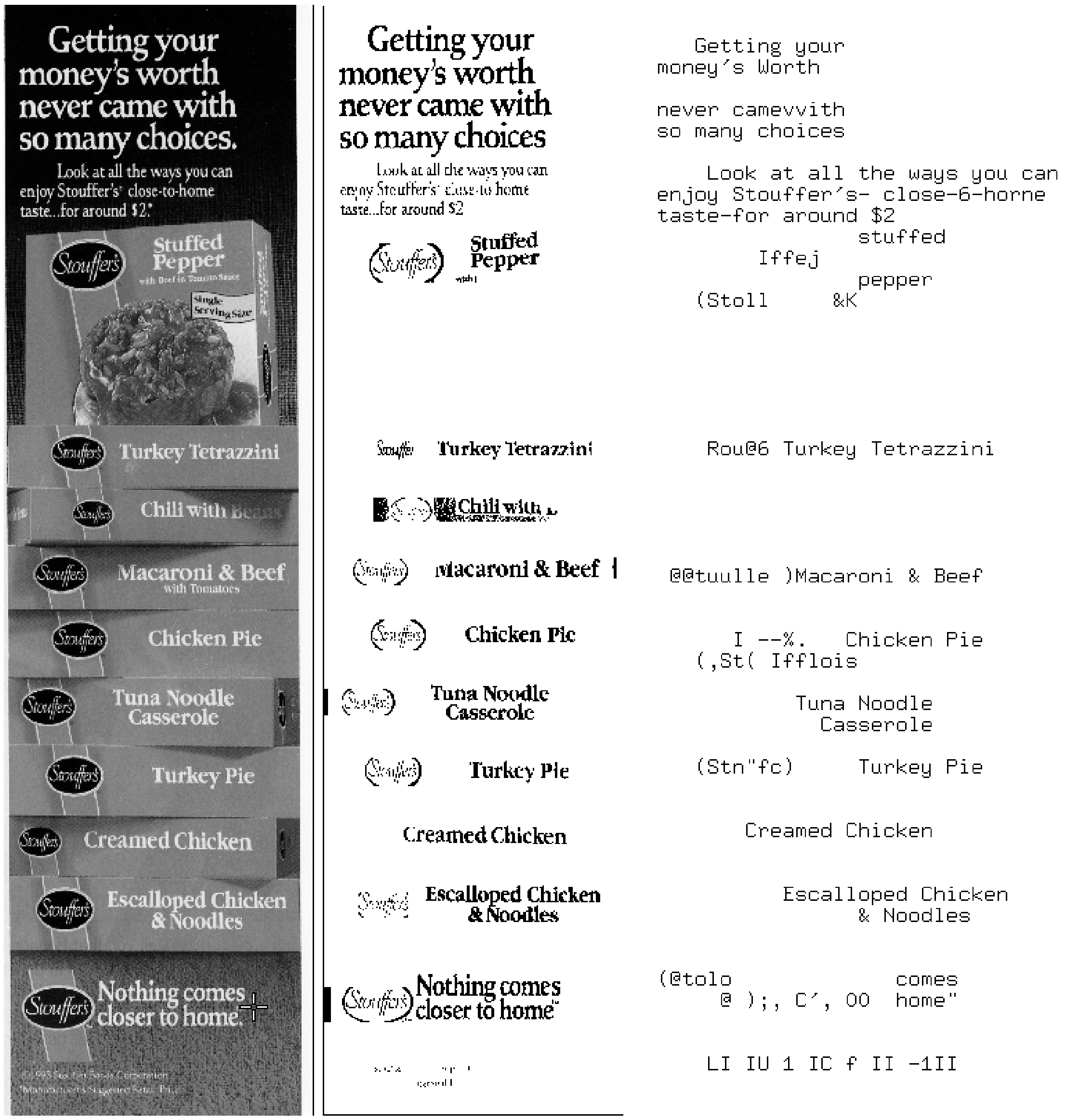
8.1 Text Detection and Clean-up

Table 1 demonstrates the performance of the system up to the Chip Refinement process. The Total Char (Words) column shows the number of characters (words) in each image as perceived by one of the authors — this is the ground truth. The Detected Char (Words) shows how many characters (words) are completely enclosed by the text boxes produced after the Chip Scale Fusion step. The Cleaned Char (Words) shows the number of characters (words) which are clearly readable by a person after the Chip Refinement and Text Clean-up processes. Note that only the text strings with skew angles less than roughly 30 degrees are counted, since the OCR engine cannot handle text which is skewed seriously. Split Words shows how many words are split (i.e., no single bounding box covers the word completely).

As shown in the table, there are a total of 21820 characters and 4406 words in the testing 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. A word is successfully cleaned only if all its characters are clearly recognizable by a person. These results are listed in Table 2.

	Total	Detected	Percent	Normalized Percent	Clean-up Percent
Char	21820	20788	95%	89%	91%
Word	4406	4139	93%	85%	86%

Table 2: Summary of the text extraction performance over 48 images.


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
Chicken Pie
Tuna Noodle Casserole
Turkey Pie
Creamed Chicken
Escalloped Chicken & Noodles
Nothing comes closer to home.

```
  Getting your
money's Worth

never camevvith
so many choices

   Look at all the ways you can
enjoy Stouffer's- close-6-horne
taste-for around $2
                stuffed
        Iffej
                pepper
   (Stoll     &K

   Rou@6 Turkey Tetrazzini

 @@tuulle )Macaroni & Beef

      I --%.   Chicken Pie
   (,St( Ifflois
              Tuna Noodle
                Casserole
   (Stn"fc)     Turkey Pie

          Creamed Chicken

             Escalloped Chicken
                 & Noodles

(@tolo               comes
     @ );, C', 00   home"

   LI IU 1 IC f II -1II
```

(a) (b) (c)

Figure 12: Example 1. **(a)** Original image (ads11). **(b)** Extracted text. **(c)** The OCR result using Caere's WordScan Plus 4.0 on (b)

File Name	OCRable Char	OCRed Char	Print Words	OCRed Words	Light Dark
ads01	81	77	20	17	L
ads02	162	152	41	35	L
ads03	184	61	42	6	L
ads04	652	392	129	46	D
ads05	210	148	58	29	D
ads06	162	147	25	20	D
ads07	111	98	23	18	D
ads08	68	61	9	6	D
ads09	24	21	3	2	D
ads10	134	75	27	18	L
ads11	147	123	50	45	L
ads12	211	173	42	31	D
ads13	425	379	120	105	D
ads14	1471	1237	316	270	D
ads15	212	183	40	34	D
ads16	108	66	27	15	L
ads17	224	114	36	19	L
ads18	258	47	57	11	D
che05	207	166	45	31	D
cpn01	608	568	135	120	D
ctn01	1064	990	288	260	D
ctn02	104	102	19	18	L
ctn05	102	102	25	25	D
env01	156	152	27	24	D
inv01	1188	680	233	73	D
inv02	481	454	108	98	D
mag01	812	762	163	137	D
mag02	84	83	17	16	D
npr01	132	130	27	26	D
npr02	756	729	166	156	D
pho04	9	0	2	0	D
pic02	11	4	2	1	D
pic01	1177	1009	244	202	D
txt01	2008	2003	223	218	D
txt02	960	940	192	182	D

Table 3: OCR result using Caere's WordScan Plus 4.0 for MicroSoft Windows95

8.2 OCR Testing

Figure 12(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 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 3). This is because the words are actually blurred, hence the region has very low energy. For the same reason, the two words under "Macaroni & Beef" and those on the bottom of the input image are not found. 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 12 (c). It made mistakes on the Stouffer's trademarks since they are in script. It failed to recognize "Chili with" mainly because of the noise around there. 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.

Table 3 shows the results of applying Caere's WordScan Plus 4.0 on some of the cleaned text from the test images. The OCRable Char column shows the number of characters (cleaned) that appear to be of machine printed fonts in the corresponding image, while the Print Words column shows the number of words from these characters. The OCRed Char column shows the number of characters which are correctly recognized by the OCR engine, and the OCRed Words column shows the number words which are correctly recognized. A "L" ("D") in the Light Dark column means the text is lighter (darker) than the background in the original image. Notice that only the machine printed characters are counted so that the OCR engine can be applied.

As shown in the table, there were 35 images used in this experiment which contains 14703 printed characters and 2981 printed words. 12428 (84%) of the characters, 2314 (77%) of the words are correctly recognized. The normalized percentages are 78% and 72% respectively. These results are summarized in the following table:

	Total OCRable	Total OCRed	Percent	Normalized Percent
Char	14703	12428	84%	78%
Word	2981	2314	77%	72%

Table 4: Summary of the OCR performance over 35 cleaned-up images.

9 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 is used to direct 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 original resolution of the input image. Third, an algorithm which cleans up the background and binarizes the detected text strings 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 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 video frames, 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. 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 characters 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.

10 Acknowledgements

We would like to thank Bruce Croft and CIIR for supporting this work. We would also like to thank Allen Hanson, Jonathan Lim and Yong-Qing Cheng for their constructive comments and suggestions. Jonathan Lim also provided system support.

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] Dennis Dunn, William E. Higgins, and Joseph Wakeley. Texture Segmentation Using 2-D Gabor Elementary Functions. *IEEE Transactions on Pattern Analysis And Machine Intelligence*, 16(2):130–149, Feb. 1994.

[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] Osamu Hori and Akio Okasaki. High Quality Vectorization Based on a Generic Object Model. *Structured Document Image Analysis*, pages 325–339, 1992.

[7] Anil K. Jain and Sushil K. Bhattachariee. Address Block Location On Envelopes Using Gabor Filters. *Pattern Recognition*, 25(12), 1992.

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

[9] Anil K. Jain and Farshid Farrokhnia. Unsupervised Texture Segmentation Using Gabor Filters. *Pattern Recognition*, 24(12), 1991.

[10] Richard A. Johnson and Dean W. Wichern. *Applied Multivariate Statistical Analysis.* Prentice-Hall, Inc, 1982.

[11] 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.

[12] Su Liang and M. Ahmadi. A Morphological Approach to Text String Extraction from Regular Periodic Overlapping Text/Background Images. *Computer Vision, Graphics and Image Processing*, 56(5):402–413, Sept. 1994.

[13] Huizhu Luo and Its'hak Dinstein. Using Directional Mathematical Morphology for Separation of Character Strings from Text/Graphics Image. IAPR International Workshop on Structural and Syntactic Pattern Recognition, Naharia, Israel, Oct 1994.

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

[15] 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.

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

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

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

[19] 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.

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

[21] 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.

[22] W. H. Tsai. Moment-Preserving Thresholding: A New Approach. *Computer Vision, Graphics, and Image Processing*, 29(3):377–393, Mar. 1985.

[23] S. Tsujimoto and H. Asada. Major Components of a Complete Text Reading System. *Proceedings of The IEEE*, 80(7):1133–1149, July 1992.

[24] 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.

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

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