A Segmentation-free Approach to OCR

C. H. Chen and J. L. DeCurtins

Information, Telecommunications, and Automation Division
SRI International, Menlo Park, California 94025

ITAD-3339-PA-92-128

Abstract

One of the most successful applications of computer vision is in the area of OCR (optical character recognition). In the last ten years many commercial OCR systems have been introduced to the market, and all have claimed read rates above 99%. When confronted with degraded images of text, however, the performance of these systems deteriorates severely. This occurs because all these systems rely on a segmentation step that is prone to error in the presence of image noise and printing artifacts. We present a novel OCR approach that overcomes this problem by eliminating the segmentation step altogether. This approach is based on the concept and techniques of occluded object recognition. To achieve high efficiency as well as robustness, we incorporate the notions of indexing and voting, and tailor them to the problem of OCR. Preliminary experimental results are given.

1: Introduction

Document analysis is an application of computer vision that shows great commercial potential and has, consequently, received a great deal of attention in recent years. The most critical step in a document analysis system is the automated recognition of characters in text images. This is conventionally referred to as optical character recognition (OCR).

Our experience with some of these commercial OCR machines and, in particular, with U.S. Postal Service mail-sorting OCR machines, has shown that their performance deteriorates severely when they are presented with degraded images. An analysis of the subsequent read errors revealed a fundamental problem in their basic approach: they all use a segment-then-classify scheme to recognize characters. In this approach, a text image is segmented into isolated blocks that supposedly contain individual characters. Each unknown character is then classified into one of the reference characters, using various classification techniques. The fundamental problem with this approach is that the segmentation step is very sensitive to image noise and printing artifacts, and any errors in segmentation almost inevitably cause errors or failures in classification.

Our OCR approach overcomes this segmentation problem by eliminating the segmentation step altogether. This segmentation-free approach is based on the concept of occluded object recognition, in which objects are recognized and *then* segmented out from the image. In applying the concept of occluded object recognition to the problem of OCR, we treat characters as touching or occluded objects that are subject to special constraints on their poses, i.e., they are juxtaposed with little or no freedom in rotation. Based on these characteristics, we combine two very powerful techniques used in occluded object recognition—indexing and voting (pose clustering)—and tailor them to the problem of OCR. This results in a segmentation-free OCR approach that is both highly efficient and robust. We note that recently some techniques have been proposed (e.g., [1]) for hand-written OCR that conceptually are also segmentation-free, although these techniques are quite different from ours.

2: Conventional OCR

Most conventional OCR techniques use a segment-then-classify approach to read text. First, an image of a line of text is segmented into blocks that supposedly contain individual characters. This step is called character segmentation. The unknown character in each segmented block is then classified into one of the reference characters, based on features extracted from the block. It is interesting to note that this segment-then-classify OCR approach represents a paradigm of the divide-and-conquer problem-solving strategy. To make OCR size independent, there is usually a normalization step between character segmentation and classification. This is done by scaling the width and height of each segmented block to the corresponding standard size of the reference characters. Here, we present a brief review of various aspects of the approach and note the fundamental problem with the approach.

0-8186-2840-5/92 $03.00 © 1992 IEEE

2.1: Character segmentation

The basic idea of character segmentation is to dissect the image of a line of text into locations between characters, i.e., character breaks. In the simplest case, one would expect that text is printed with a sufficient gap between adjacent characters, and therefore character breaks will be indicated by uninterrupted vertical white space. In reality, however, character breaks are frequently distorted by the compound effect of fonts, type sizes, printing quality, binarization, and imaging artifacts. These distortions pose serious problems for character segmentation. For example, in the case of dot matrix printing or insufficient inks, characters tend to be fragmental, resulting in oversegmentation. While in the case of ink smudging, serif fonts, and ligatures, many adjacent characters may be touching each other, resulting in undersegmentation (see Figure 1).

These segmentation problems can be divided into two classes: (1) merging of characters, and (2) fragmentation of characters. Common reasons for merging characters include:

- Underthreshold binarization.
- Ink smudging. When the ink on pages bleeds, characters that were not intended to be joined may touch.
- Background patterns. For example, underlines on a form connect the characters.
- Serifs, proportional font spacing, and ligatures. Some pairs of letters, such as "ry," "ri," "oo," and "rn," are actually touching in many fonts. Other characters touch because of ligatures in the font (for example, "fi," "fl," "ffi," "ffl," or "ft").

(A) Oversegmentation

(B) Undersegmentation

Figure 1. Character Recognition Errors Caused by Incorrect Segmentation

Common reasons for character fragmentation include:

- Overthreshold binarization.
- Poor-quality printing.
- Thin strokes.
- Dot matrix or ink jet printing. If the dots are too small and too sparse, they may not be connected to form strokes, resulting in a broken character.

To handle these segmentation problems, a combination of the following techniques is used in many OCR systems:

Separation by Valley of Vertical Projection: A more sophisticated technique than the search for uninterrupted vertical white space between characters is the projection of character pixels along the vertical direction and the detection of valleys in the projection's histogram.

Separation by Connected White Path: Very often in italic or kerned fonts, characters overlap vertically but do not touch. These characters can be segmented by finding a connected path of white pixels separating the characters from top to bottom. This technique is equivalent to segmenting characters according to connected components. It is, however, very sensitive to noise or artifacts in the image.

Fixed-Pitch Segmentation: Nearly all typewriter fonts, dot matrix and daisy wheel printer fonts, and a large number of laser printer fonts are fixed pitch. Since all characters in these fonts are the same width, if the pitch can be determined, the characters can be easily separated by breaks at regular intervals.

Cut and Test: This technique dissects the character image at several candidate locations and evaluates the result of the segmented pieces. The candidate locations are determined by considering factors such as the average character width, the peaks of the lower contour, and the valleys of the upper contour of the character image. This technique, which requires feedback from the evaluation step, requires a great deal of processing time.

Remerging of Fragmented Character Pieces: In many cases, the character image will be oversegmented by using any combination of the above techniques, especially when the characters in the image are fragmented. Consequently, it is necessary to remerge the fragments. However, techniques for putting these split pieces back together usually involve many ad hoc heuristics that are both unreliable and time consuming.

2.2: Features and classification

Recognition of characters, as in the case of object recognition, must rely on features that describe the shapes of character archtypes. The task of selecting the types of features to be used for OCR is similar to that for object recognition. The features should be insensitive to noise, easy to extract, and provide sufficient discriminatory power for distinguishing among different character archtypes. The

classification method for an OCR system is generally tied to the type of features used. These features can be divided into two categories: global and geometric.

2.2.1: Global features

In general, a global feature is obtained by mapping the pixels of a character image into a feature vector according to some transformation function. Usually, this transformation function is designed to bring out some particular shape characteristic that is useful for classification. For example, projections of the character image along the horizontal, vertical, and diagonal axes have been used in the recognition of kanji characters [2]. Other types of transformations include the Fourier transform of the character contour, and moments of character pixels in the image [3].

2.2.2: Geometric features

Geometric features extract various shape primitives from a character. The major advantage of using geometric features is that they allow compact font and size-independent representation of characters. Geometric features may be derived from the contour or the skeleton of the character, or from the character image itself. Some examples are lines and curves from contours, horizontal and vertical strokes from skeletons, and cavities and holes from the original character image.

2.3: Problems

A fundamental problem with the segment-then-classify OCR approach is the total dependency of the classification step on the segmentation step. The major result of bad segmentation is that (in the case of oversegmentation) features may be missing, or, (in the case of undersegmentation) extraneous features (from the adjacent characters) may be included in the block. As a result, the structure of the character is destroyed, and the character cannot be recognized by any of the existing OCR classification techniques.

3: Segmentation-free OCR

The segmentation-free approach to OCR presented in this paper is derived from the concept of occluded object recognition. In techniques based on this approach, objects are modeled as a set of geometric features, such as lines, arcs, and corners. To recognize objects specified by a model, these techniques search the image for the maximal subset of object features that have a spatial relationship consistent with the model features. When such a subset of consistent features is found, the object is recognized (more precisely, an instance of the model object is found in the image), and only then is the object segmented out from the image. The recognition step continues in this fashion until all the objects are recognized and segmented out from the image. Because these techniques only look for subsets of features, they work well when part of the object is obscured.

Two important notions—*indexing and voting*—have been recently introduced to attack the problem of occluded object recognition. The basic idea of indexing is to speed up the image-to-model feature-matching process by setting up an indexing mechanism for all the model features. One example that uses this indexing idea is the geometric hashing technique [4]. The concept of voting is to perform recognition by looking for consensus among features detected in an image. The pose clustering technique [5], also referred to as generalized Hough transform, is an example. Because an object is recognized only if it obtains the consensus of image features, recognition by voting is a very robust technique in the sense that it can tolerate missing as well as spurious features.

The basis of our segmentation-free OCR approach, presented below, is an integration of the notions of indexing and voting.

3.1: Overview of the approach

Without loss of generality, we assume that the task is to recognize characters within some portion of a line of text as shown at the top of Figure 2. Akin to the problem of occluded object recognition, the recognition of characters also involves determining the positions of the recognized character within the line of text. Unless stated otherwise in the following discussion, we will use the term *position* to refer to the location on the horizontal axis.

We model each character by a set of geometric features. Each geometric feature is described by a set of attributes (such as its vertical and horizontal position, and stroke length). The attributes depend on the types of features that are used. A key component in this approach is the pre-organized indexing-and-voting mechanism, as shown in Figure 2. This indexing-and-voting mechanism is organized according to the geometric features of the reference characters so that an image feature of an unknown character can be rapidly indexed to its matching model features. This mechanism is constructed offline so as to maximize the efficiency in the online character recognition process.

The recognition of characters proceeds, as illustrated in Figure 2, with the following steps:

Feature Extraction: Geometric features are extracted from the character image.

Indexing: Deriving hash keys from the attributes that specify each image feature, we match reference characters to image characters by matching hash keys. Each hit, i.e., a key into an index associated with a particular reference character, indicates a possibility (hypothesis) that a reference character appears in the image at a position relative to the detected image feature.

Voting: For each hit, we compute the implied horizontal position of the reference character on the line, based on the position of the image feature. A vote is then cast for that character at that position. Note that an image feature may

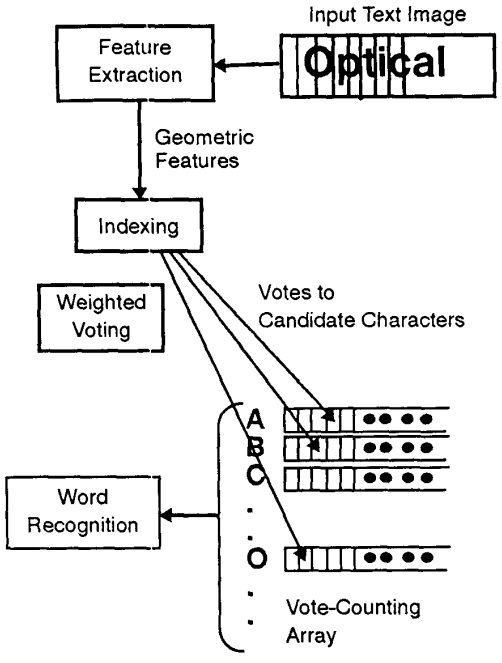


Figure 2. Segmentation-Free Approach

cast votes for more than one reference character. These votes are tallied in the vote-counting array (as shown later in Figure 6).

Character/Word Determination: The collection of votes is thresholded to determine the character with the highest probability at each image position corresponding to an entry in the vote-counting array. Confidence in each recognized character is based on the number and nature of the features found in the image.

In this approach, no character segmentation is necessary because the features are extracted from the entire text line. The method is insensitive to occlusion, touching characters, and noise as long as enough of each character is visible and a robust set of feature detectors is used. The four major steps outlined above are discussed in detail below.

3.2: Feature extraction

Because no single type of feature is appropriate for all images and types of printing, we are creating a library of features to be applied. In our initial trials, two types were used, *mask* and *contour*.

The mask feature is essentially a template in which the entries in a predefined mask (an array of values) are compared, pixel by pixel, to the corresponding values in an image region. A score is tallied as the image pixels either match or don't match the expectations encoded in the

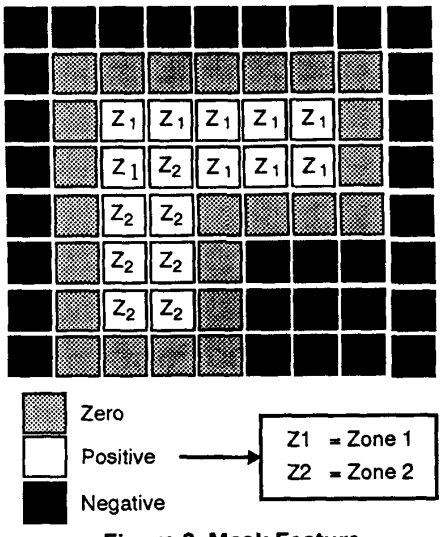


Figure 3. Mask Feature

mask. The mask entries are of three classes: positive, negative, and zero. If a mask entry is positive and the corresponding image pixel is a pen stroke, that positive value is added to the feature score. If a mask entry is negative and the corresponding image pixel is a pen stroke, that negative value is added to the feature score (thus diminishing the score). A mask entry of 0 indicates a "don't care" pixel, and thus the score is unaffected by the status of the image pixel. These values are shown in Figure 3.

We have augmented this standard template feature by the introduction of zones within the positive regions of the mask. A zone is an identified subset of the positive mask entries. When a mask feature is being scored over an image region, each zone within the feature must attain some minimum number of matches (i.e., image pen pixels). This technique implies that a high-scoring detected feature has evidence throughout its area of interest. Figure 3 is a typical zone layout.

The second type of feature we have used is an edge contour, which is a second-order Bezier curve fit to smooth portions of the outer and inner contours of pen pixel regions within the image. An example of such a fit (including the "control points" that define the curve) is shown in Figure 4.

3.3: Indexing

To construct an effective indexing scheme, we have to decide what attributes of the features are to be used as the indexing keys. These attributes should provide discriminative power and be reliable. For each attribute used as an indexing key, an index table, which essentially is a quanti-

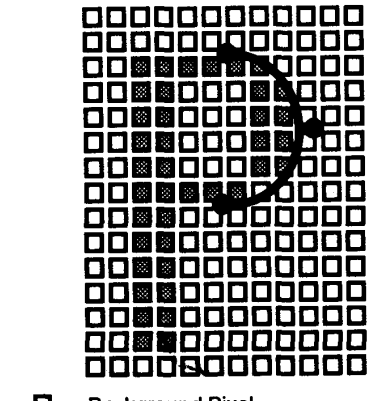
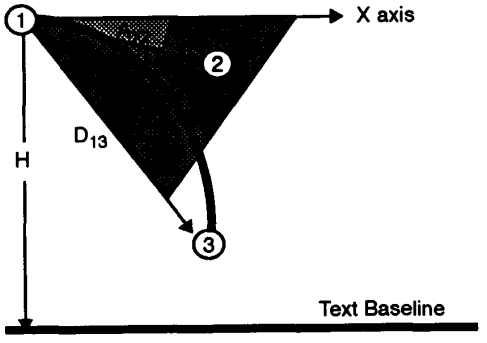


Figure 4. Contour Feature



H = Height above Baseline
D_{12} = Distance from Pt 1 to Pt 2
D_{13} = Distance from Pt 1 to Pt 3
A_{X2} = Angle from X axis to Pt 2
A_{X3} = Angle from X axis to Pt 3

Figure 5. Contour Feature Indexing

zation of the space of the attribute value, is constructed. The quantization resolution depends upon the sensitivity of the attribute.

In our initial experiments we have used the height above baseline as the single index for the mask features. For contour features, we use the height above baseline and four additional attributes as shown in Figure 5. Two of these four are the distances between control point 1 and the remaining control points. The other two attributes are the angles between the control point lines and the horizontal axis.

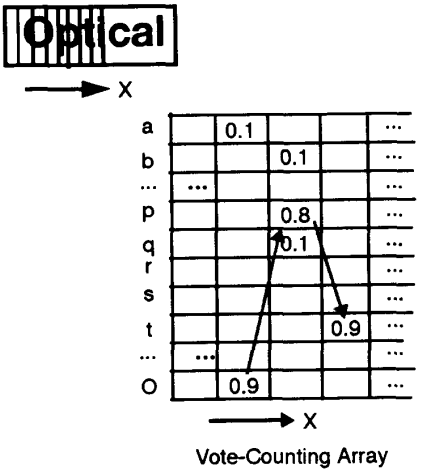


Figure 6. Voting by Indexed Features

3.4: Voting

In the current implementation, a matched image feature casts exactly one vote to each of its reference characters. For more robust recognition, however, the vote to a reference character should be weighted according to the utility of the feature with respect to the reference character (see Figure 6. In other words, the confidence of a recognized character depends on the features of the character that have been matched, not just the number of features recognized. For example, an image of "rn" can be recognized as two characters "r" and "n" or confused as one character "m." The confidence on "m" should be lower because an important feature, the Y junction in the middle section of "m," is usually missing in the image. To quantify the importance of a feature or a feature cluster (a set of closely related features) to a character, we plan to apply the statistically robust definition of feature utility developed by Chen and Mulgaonkar [6]. By using this type of utility measure, we can compute a confidence value that properly takes into account features that are seen as well as those that are missed.

3.5: Character/word determination

The simplest form of character determination from the vote accumulation array is to choose the reference character that receives locally the highest percentage of votes. However, there could be ambiguity that cannot be reliably resolved by this simple technique. We are currently investigating several techniques for improving the reliability of this process, e.g., utilizing character co-occurrence statistics, using lexicon, and eventually integrating character determination with word recognition.

4: Experimental results

To apply our segmentation-free scheme, we must first create the feature indexing and voting mechanism. This is done during a training phase. Features are detected within text images for which the ground truth (i.e., the identity and location of each character) is known. The indices of each detected feature are computed and the identity of the character that gave rise to the set of indices is noted in the corresponding location of the index table.

These computed indices are quantizations of one or more attributes of a particular feature. As noted above, we have used one attribute for our 'mask' feature and five for our 'contour' feature. There is a tradeoff in the choice of quantization size for a given attribute. Too fine a resolution can generate a very large index space in which features detected during OCR may not 'hit' in the neighborhood of a feature encountered during the training phase. Too coarse a resolution causes different characters to be bunched (perhaps incorrectly) into the same region of the index space. For our initial trials, we chose a coarse quantization.

The height attribute of the mask and contour features were quantized to one-eighth of the possible range of values (i.e., twice the height of an uppercase letter in the text line). The distances between control points (for contour features) were quantized to the same resolution as height. The angles (also for contour features) were quantized to 30 degrees.

Note that a given feature (resulting in a given set of indices) may imply more than one character. This occurs when the same feature attributes are found on more than one character. An example of this is the curved top of a lowercase "e." If a mask feature detector is used to find this, it will likely find the tops of "c," "g," "o," and others as well.

In addition to noting the identity of the character associated with the feature, its horizontal location (relative to the feature location) is also stored. Thus, when this feature is detected during OCR operations, the resulting indices will retrieve both the corresponding character and its location. These locations are mapped into the vote-counting array.

In Figure 7 we show an initial result obtained after a very small training phase using only contour features. The boxes below the text image indicate the locations of the vote-counter slots. The horizontal size of the counter slots was chosen (somewhat arbitrarily) to be one quarter the height of the line. The number in each box indicates the maximum vote tallied for any one character in that slot. In some cases, more than one character received the maximum. These maximum-vote characters are listed below each box. We applied a simple threshold of two votes and displayed (above the boxes) those characters that received

Figure 7. Initial Voting Results

at least this many votes. Nearly all characters are correctly identified. Note that the vote for the second 'p' in "Mississippi" was split between two accumulator slots.

5: Conclusion

We have discussed a new, segmentation-free approach to optical character recognition. Our initial results look promising and, in contrast to conventional OCR, we expect further results to verify the following advantages of this approach:

- *Higher read rate.* Because our approach gets around the error-prone segmentation step, we expect to improve the OCR read rate by as much as is lost in the segmentation step.
- *Faster Execution.* A conventional OCR system has four major processing steps: (1) segmentation, (2) normalization, (3) feature extraction, and (4) classification. Our segmentation-free approach has only two major processing steps: (1) feature extraction, and (2) indexing and voting. Assuming that the processing time for feature extraction is about equal, our approach has saved the time required for segmentation and normalization. In addition, the indexing and voting scheme we have described involves only very simple computation, saving the expense of the classification techniques used by conventional OCR. This combination of better read rate and faster execution is possible because much of the essential information for recognition, i.e., attributes of the features of the reference characters, has been built into the indexing and voting mechanism offline. Therefore, we gain maximal efficiency in the online recognition process.

Our future plans include:

- Continued development of feature detectors.
- The use of weighted voting, i.e., analysis of feature utility.
- Extensive comparative testing using UNLV and NIST images.
- Extension to handwritten character recognition, where the segmentation problem is even greater.

6: References

[1] S. Edelman, T. Flash, and S. Ullman, 1990. "Reading Cursive Handwriting by Alignment of Letter Prototypes," *Int. J. of Computer Vision*, pp. 303-331.

[2] S. Mori, K. Yamamoto, and M. Yasuda, 1984. "Research on Machine Recognition of Handprinted Characters," *IEEE Trans. on PAMI*. Vol. 6, No. 4, pp. 386-405.

[3] N.D. Tucker and F.C. Evans, 1974. "A Two-step Strategy for Character Recognition using Geometrical Moments," *Proc. 2nd Int. Joint Conf. Pattern Recognition*, pp. 223-225.

[4] Y. Lamdan and H.J. Wolfson, 1988. "Geometric Hashing: a general and efficient model-based recognition scheme," *2nd Int. Conf. on Computer Vision*, pp. 238-249.

[5] G. Stockman, 1985. "Object Recognition and Localization via Pose Clustering," *Computer Vision, Graphics, and Image Processing*, 40, pp. 361-403.

[6] C-H. Chen and P.G. Mulgaonkar, 1990. "Feature-Utility Measures for Automatic Vision Programming," *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 1840-1845.