社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

DPマッチングによる接触文字分離

クリストファー マロン[†] 内田 誠一[††] 鈴木 昌和[†]

[†] 九州大学大学院数理学研究院 〒 812–8581 福岡市東区箱崎 6-10-1
[††] 九州大学大学院システム情報科学研究院 〒 819–0395 福岡市西区元岡 744
E-mail: †{malon,suzuki}@math.kyushu-u.ac.jp, ††uchida@is.kyushu-u.ac.jp

キーワード OCR、 セグメンテーション、接触文字、数式記号認識、 DP マッチング

Separation of Touching Characters Using DP Matching

Christopher MALON[†], Seiichi UCHIDA[††], and Masakazu SUZUKI[†]

[†] Faculty of Mathematics, Kyushu University
Hakozaki 6-10-1, Higashi-ku, Fukuoka-shi, 812–8581 Japan
[††] Faculty of Information Science and Electrical Engineering, Kyushu University
Motooka 744, Nishi-ku, Fukuoka-shi, 819–0395 Japan
E-mail: †{malon,suzuki}@math.kyushu-u.ac.jp, ††uchida@is.kyushu-u.ac.jp

Abstract Ideally, an OCR system would partition the set of connected black components on a page into subsets representing individual characters. However, this approach is inadequate if some component partially belongs to several touching characters. We present a DP matching-based method that cuts such a component apart, given a hypothetical classification for the leftmost part. Our method produces better quality cuts than well-known methods, particularly in mathematical expressions, where characters are often slanted and may touch in widely varying configurations. A good cut allows single-character recognition techniques to be applied to the cut part and the residual image, in order to judge whether the hypothetical classification was correct.

Key words Optical character recognition, segmentation, touching characters, mathematical symbol recognition, DP matching

1. Introduction

Single character recognition methods fundamentally rely upon segmentation into groups of connected components, representing individual characters. In practice, characters may be broken, or neighboring characters may touch, making the input to the single-character recognizer be more or less than a single character.

Printed mathematical documents are distinguished from typical literature by the variety of similar symbols, the need to distinguish identical letters appearing in different styles, the lack of a fixed vocabulary for mathematical expressions, and the two-dimensional arrangement of symbols on a page. In an early version of InftyReader, a mathematical OCR program available free of charge from the Suzuki Laboratory at Kyushu University [1], touching symbols accounted for 60% of character misrecognitions [2]. Some typical examples of touching patterns are shown in Figure 1.

Many researchers have recognized the importance of character segmentation; a survey appears in [3]. The goal of many of these algorithms is to cut the image in a straight vertical line from top to bottom, often with the feedback of a classifier to choose among possible lines. This process may be repeated recursively, as a "recursive segmentation and classification" process, until each character in the touching group is recognized [4]. Italic letters have been treated by similar approaches, after determining at the pitch angle and correcting for it [5].

The examples in Figure 1, however, show the inadequacy of these approaches for OCR of mathematical documents. In the images of 'S^q', 'e^s', and '$_k)$', the letters appear in different sizes, at different baselines. The images 'w^μ', '$f($', 'S^q',

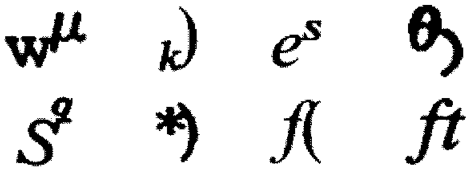

Fig. 1 Examples of touching characters in mathematical documents.

and '*)', 'f)', and 'ft' should not be cut vertically, by a segment of any length. No fully extended line in any direction could cut the 'f)', the 'ft', or the '*)' into clean specimens of the individual characters.

A couple of these authors have addressed segmentation of touching characters in mathematical expressions, in particular. A method of Okamoto *et al* [6], searches for valleys in the binary image convolved with the Gaussian kernel, and follows them to allow cuts in many directions. The method of Lee and Lee [7] may be closest to ours. They introduce "front features," which represent the first part of a letter's contour that is expected after an anchor point. Without guessing the initial letter, they attempt to match these features, for all reference images, to the touching image. DP matching is attempted against all reference images in parallel, with possible matches abandoned as their scores become too high. Our notions of *reduced frame* and variable-dimensional peripheral features, described below, take the front-matching idea of "front features" and make it less sensitive. Their system was tested on only 50 mathematical expressions, and the performance of the touching-character segmentation was not separately analyzed.

A seemingly effective algorithm is reported by [8], allowing cuts horizontally, vertically, and in both diagonal directions (though each of these possibilities must be considered separately). The segmentation is purely geometric, without initial recognition results, and proceeds to try various alternatives until a good recognition result is achieved. They utilize an unusual and powerful feature definition. The examples they picture are all breakable by lines; we are unsure how their method would perform on the characters of Figure 1, which must be broken not by a line but by a line segment.

We propose a fundamentally different strategy, making few assumptions about the geometry of touching characters. Rather than searching for unusual values of geometric features (extrema of vertical projections, minima of the Tsujimoto metric, or crossing counts), we begin by taking the entire image as input to a classifier that attempts to recognize the leftmost character. Whether the letter 'a' appears alone, or whether it is part of a group of touching letters such as 'as' or 'at,' this classifier is designed to return the result 'a.' We describe the construction of this classifier in Section 2.

After forming a hypothesis for the leftmost character, we cut the image, so that the remaining characters may be recognized. Performing this segmentation is the task of our *DP cutting* algorithm. DP cutting applies dynamic programming (DP) methods to separate a touching-character image (the *sample image*), using a prototype (the *reference image*) for the character believed to be on the left. It uses the geometry of the sample image to determine possible line segments (in many directions) where a cut is likely. Then, it decides among these possibilities by solving a DP matching problem that compares the alignment of the reference character's contour with the contour formed by breaking the sample character along a cut. We give details of this step in Section 3.

Thus, although our algorithm might be considered as "recursive segmentation and classification," it reverses the order of segmentation and classification, compared to the methods of previous such algorithms. The overall strategy may be summarized as follows:

```
For each connected component on the page:
(1) Classify the character using a traditional single-character recognizer $C$.
(a) If the score is good, accept the classification.
(b) Otherwise, suppose it to be a touching character.
    i. Use the touching-character classifier to produce a list $L$ of possible results for the left character.
    ii. For each candidate $x$ in the list $L$:
        A. Apply DP cutting to find the best segmentation of the sample image, assuming the left character to be $x$.
        B. Cutting at the proposed segmentation, classify the left part using the same classifier $C$ as above, calling the result $y$.
        C. If $y = x$, accept $x$ as the left character, and return to step 1 to recognize the residual image.
        D. Otherwise, continue with the next candidate in $L$.
```

Although we refer to the "left" and the "right" character, our notion of "leftmost" is significantly more general than in common algorithms, which suppose that one character falls completely to the left of another. Broken characters should also be treated in Step 1 (b), but describing that method is beyond the scope of this paper.

2. The touching-character classifier

The touching classifier aims to recognize the "leftmost"

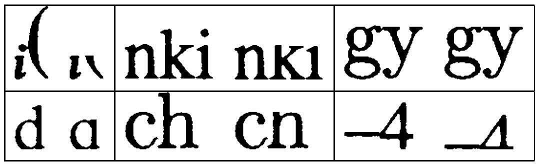
Fig. 2 Examples of cropping an image to its reduced frame.

character in an image consisting of several touching characters. The image may consist of more than one connected component. If so, the component with the longest perimeter (called the *target component*) is extracted, and the classifier attempts to break it into two pieces. For any break line, the remaining components are assigned either to the image that was broken off, or to the residual image, depending on which side of the line their centroids fall on.

First, the classifier finds an anchor point on the target point. This anchor point is the closest vertex to the lower left corner of the bounding box. If more than one vertex is equal in distance from the corner, the vertex farthest to the left is chosen. We use this definition of anchor, rather than searching for the leftmost point, because many letters and symbols have straight, vertical contours near the left side, making the anchor's position unstable.

From this point, a portion of the image called the *reduced frame* is found. The reduced frame is a strip bounded on three sides, extending infinitely far to the right. The left side of the reduced frame coincides with the left side of the bounding box. The top side and bottom side are defined by looking for the next y maximum, proceeding clockwise from the anchor point, and the next y minimum, proceeding counterclockwise from the anchor point, that satisfy the following three conditions, for a preset value k:

(1) It must be reached when x is increasing.

(2) It must be at a more extreme y-value than any point encountered previously while traveling from the anchor point.

(3) The local extremum must be stable (no more extreme point is encountered in the next k points along the contour).

Some examples of images and their reduced frames are shown in Figure 2. The point of the definition is that the reduced frame of a single letter A should typically be the same as the reduced frame for a group of symbols AB in which A is to the farthest left. It matters primarily that A begins farthest to the left; A need not fall entirely to the left of B in the touching image.

The classifier that we apply to the touching image before DP cutting uses a vector of *peripheral features*, captured in the reduced frame. The dimension of the vector may vary. Fix a grid size m. The reduced frame is divided into m equally sized horizontal strips. The number of color changes (black/white plus white/black) along each horizontal scanline is counted. Within the ith strip, let n_i be the mode value of the color change counts. On a scanline with n_i color changes, let $x_1, x_2, \ldots, x_{n_i}$ be their horizontal positions, measured from the left of the reduced frame. These values give run-length encodings of the scanlines. Let h be the height of the reduced frame. (The reduced frame has infinite width.) Average the vectors $(\frac{x_1}{h}, \frac{x_2}{h}, \ldots, \frac{x_{n_i}}{h})$ corresponding to each of the scanlines of the strip with exactly n_i color changes. The resulting vector is the ith contribution $\vec{v_i}$ to the peripheral feature vector. The complete peripheral feature vector is the concatenation $(\vec{v_1}, \ldots, \vec{v_m})$ of these feature vectors. Its dimension is $n_1 + n_2 + \cdots + n_m$. We call the m-tuple $(n_1, \ldots, n_m)$ the *dimension profile* of the feature vector.

The matching relation between two peripheral feature vectors $\vec{v} = (\vec{v_1}, \ldots, \vec{v_m})$ and $\vec{v'} = (\vec{v'_1}, \ldots, \vec{v'_m})$ involves a choice of *threshold vector* $\vec{\epsilon}$, with the same dimension profile as $\vec{v}$. We say that the vectors *match within* $\vec{\epsilon}$, and write $\vec{v} \lesssim_{\vec{\epsilon}} \vec{v'}$, if the following conditions are all satisfied:

(1) **Dimension compatibility**: For each i, $1 \leqq i \leqq m$, we have $n_i \leqq n'_i$.

(2) **Subprofile matching**: Write $\vec{v_i} = (x_{i,1}, \ldots, x_{i,n_i})$ and $\vec{v'_i} = (x'_{i,1}, \ldots, x'_{i,n'_i})$. For each i, and each j, $1 \leqq j < n_i$, we have

$$|x_{i,j} - x'_{i,j}| < \epsilon_{i,j}.$$

(3) **Boundary condition**: For each i, we have

$$x'_{i,n_i} - x_{i,n_i} > -\epsilon_{i,n_i}.$$

Typically, if an image XY is formed by adjoining a symbol Y touching the image X on the right, we expect the reduced frame of XY to match that of X, even if Y protrudes above or below the bounding box of X. Furthermore, we expect the feature vectors $\vec{v}$ of X and $\vec{v'}$ of XY to satisfy $\vec{v} \lesssim \vec{v'}$. (Refer again to the examples in Figure 2.) The "subprofile matching" in this situation occurs with strict equality; an inequality occurs in the "boundary condition" because the last run of black pixels has become longer.

In practice, we will not be comparing an image to its own left portion, but to idealized "reference images." From single character images in the training database, we record the dimension profiles (possibly several) with which each symbol appears. The (symbol, dimension profile) pairs are called *refined classes*. Within the training samples of each refined class A, we measure the median feature vector $\vec{m_A}$ and the standard deviation vector $\vec{\sigma_A}$.

This preparation allows us to construct a touching character classifier that assigns several candidate results to a touching image. We choose each threshold vector $\vec{\epsilon_A}$ to be a constant multiple of $\vec{\sigma_A}$. Given an input sample with feature vector $\vec{v}$, the classifier returns all classes A such that $\vec{m_A} \lesssim_{\vec{\epsilon_A}} \vec{v}$.

3. DP Cutting

The *DP cutting* algorithm is applied to a touching image against a reference image for each candidate refined class. In this experiment, these reference images are simply the training images with feature vectors closest to the medians. The touching image and reference images are each represented by polygonal appproximations. The vertices in each should be sampled at equally spaced intervals, after scaling both images to the same reduced frame height. The DP matching problem we construct attempts to align the chain code [9] of the reference image to the chain code of the touching image, broken by a cut. The cut emerges as part of the output of the DP matching problem.

A traditional DP matching problem can be visualized by a grid in which one axis represents the first sequence and another axis represents the second. A DP cutting problem can be visualized by two such grids—a "pre-cut" grid and a "post-cut" grid, with some edges joining the two grids.

A node of the DP cutting problem is a triple (i, j, k), where i is the index of a vertex of the reference image, numbered from the anchor point, j is the index of a vertex of the touching image, and k is 0 or 1 (indicating, respectively, the pre-cut or post-cut grid). Each node (i, j, k) has parents $(i-2, j-1, k)$, $(i-1, j-1, k)$, and $(i-1, j-2, k)$. Additionally, some nodes on the post-cut grid $(i, j, 1)$ have parents of the form $(i, j', 0)$, indicating a cut from vertex j' to j. At any node, the alignment cost is $\min(|r(i)-s(j)|, 8-|r(i)-s(j)|)$, where $r(i)$ and $s(j)$ are the chain codes at the corresponding vertices. The problem is to find the least expensive path from $(0, 0, 0)$ to $(m-1, n-1, 1)$, where m and n are the numbers of vertices in the two images. If $(i', j', 0)$ is the predecessor of $(i, j, 1)$ in this path, then the solution determines the cut (j', j).

The edges between the two grids are determined by the potential cuts. A pair of indices (j', j) of the contour of the touching image is a potential cut if:

(1) **Non-triviality**: We have $j' + 1 < j$.

(2) **Local minimum**: Let $d(k, k')$ denote the distance between the vertices at index k and index k'. We have $d(j'-1, j) \geqq d(j', j)$, $d(j'+1, j) > d(j', j)$, $d(j', j-1) \geqq d(j', j)$, and $d(j', j+1) > d(j', j)$.

(3) **Interior constraint**: The line segment joining the vertices at j' and j does not exit the polygon.

For each potential cut (j', j), there are edges from the pre-cut grid at $(i, j', 0)$ to the post-cut grid at $(i, j, 1)$, for every i.

4. Results

The touching characters we test are the 4,246 groups of two or more touching characters found in InftyCDB-1 [10]. They constitute 8,569 characters altogether.

To train the classifier, we use the 188,752 images of clean (non-touching, non-broken) single characters from InftyCDB-3-A [11], and measure dimension profiles and peripheral feature vectors using $m = 6$ horizontal strips. Rejecting classes with fewer than ten samples, 1662 refined classes are created from 363 different symbols appearing in InftyCDB-3-A. Within each refined class A, we measure the median $\vec{m_A}$ and standard deviation $\vec{\sigma_A}$ of the peripheral feature vectors.

The classifier assigns a sample with peripheral feature vector $\vec{x}$ to refined class A if $\vec{m_A} \lesssim_{\vec{\epsilon_A}} \vec{x}$, where $\vec{\epsilon_A} = 3\vec{\sigma_A}$. This condition may be satisfied for many classes A, each of which becomes a candidate recognition result; on average, there are 70 candidates. The correct result appears among these candidates for 83.1% of the samples in the test data (the *acceptance rate*). By comparison, the classifier yields a 88.2% acceptance rate on the 70,637 non-touching characters that appear in the InftyCDB-3-B database (described in [11]). Thus, our classifier is almost as likely to recognize the left character when it is touching as when it is alone.

In the strategy described in Section 1, our method would apply DP cutting against each of the candidate classes, and choose a final result based on whether a single-character classifier would recognize the cut result as the same candidate. For this experiment, we examine only the results of DP cutting against the true candidate. Particularly, we reject problems where the correct result does not appear in the candidate list. Sixteen randomly chosen results are shown in Figure 3.

To a human, each of the cuts appears satisfactory, except in examples 3 and 15. The problem in example 3 occurs before the DP cutting algorithm begins. We need the dot of the 'i' to be the reference image, and not the base of the 'i,' but our implementation assumes that if a reference character has multiple components, the biggest one (and only the biggest one) will be joined to another character. Example 15 fails because our implementation neglects to resample the vertices in the manner suggested in Section 3. Because the DP matching problem we have constructed allows the number of vertices in the contour outside of the jump to be stretched only by a factor of two, no reasonable cut can be produced in this example.

Though it is not apparent from Figure 3, the DP cutting al-

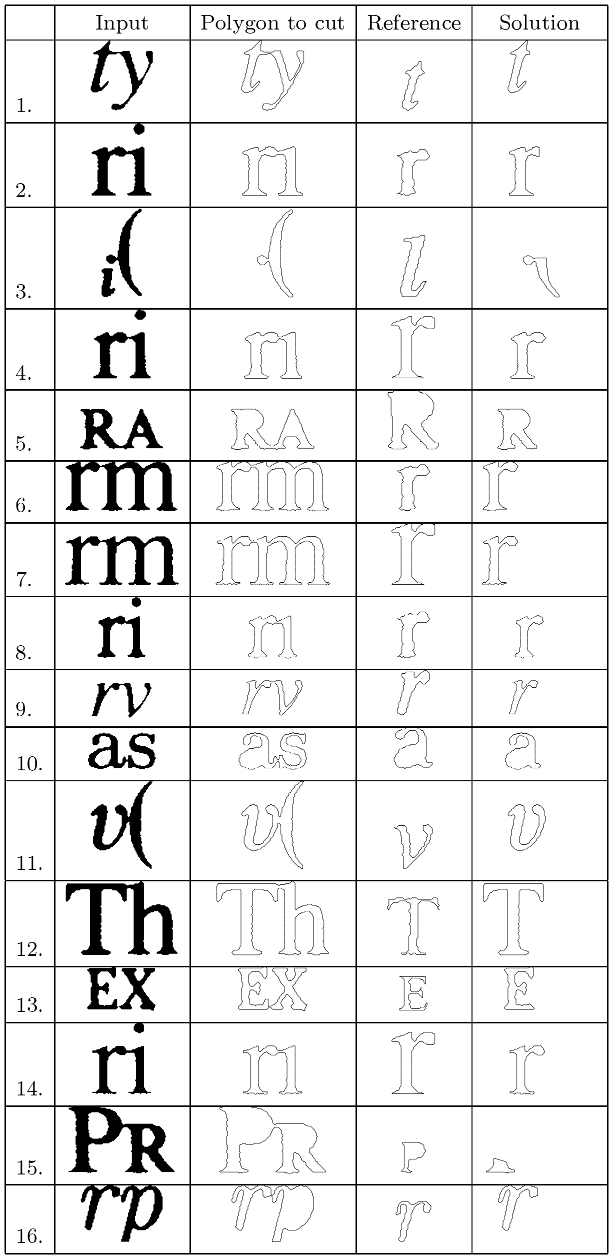

Fig. 3 Sixteen randomly chosen DP cutting problems, and their solutions.

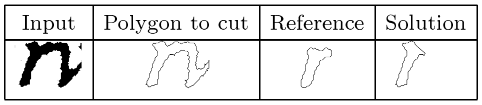

Fig. 4 The jumping problem.

gorithm sometimes fails to produce clean results, even when the correct polygons in the touching and reference images are targeted and the run lengths are properly normalized. These problems may be caused by poor reference image selection, or by a phenomenon we call the *jumping problem.*

The jumping problem occurs when the jump segment itself is so long that it should be aligned to part of the reference image. The DP cutting algorithm never allows reference vertices to be skipped, so such an alignment is impossible. Figure 4 shows an example in which italic r and v are stuck together. The touching image should be cut vertically, along a segment representing the right side of the tip of the 'r.' However, there is no progress along the reference image when this vertical segment is traversed. If the cut ends before the curve on the top of the 'r,' there will be a penalty, because the next node on the reference image would be upwards but the next node on the touching image would be leftward. Thus, the algorithm chooses a cut that skips the top curve of 'r,' and ends just before the vertical segment approaching the peak.

5. Summary

We have introduced two new algorithms in this paper. First, we have given a method that hypothesizes several recognition results for the leftmost character in a group of touching characters. Second, we have given a method for segmenting such a group into two pieces, when the ideal contour of the first piece is known. Compared to previous methods, our approach allows more kinds of interfaces between the characters and more positional relationships. At the cost of having to form a hypothesis for the recognition result, we gain freedom from strong geometric assumptions.

In future work, we intend to use the output of the DP cutting algorithm as input for a single-character recognizer. This will complete the implementation of the strategy described in Section 1 and allow us to quantitatively evaluate the accuracy of the DP cutting method.

Acknowledgments

The first author acknowledges the support of the Kyushu University Department of Mathematics 21st Century Center Of Excellence program.

文 献

[1] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori, "INFTY: an integrated OCR system for mathematical documents," DocEng '03: Proceedings of the 2003 ACM symposium on Document engineering, New York, NY, USA, pp.95–104, ACM Press, 2003.

[2] A. Nomura, K. Michishita, S. Uchida, and M. Suzuki, "Detection and segmentation of touching characters in mathematical expressions," Proceedings of the Seventh International Conference on Document Analysis and Recognition, Edinburgh, pp.126–130, IEEE Computer Society Press, 2003.

[3] Y. Lu, "Machine printed character segmentation: an overview," Pattern Recognition, vol.28, no.1, pp.68–80, 1995.

[4] R. Casey and G. Nagy, "Recursive segmentation and classification of composite patterns," Proceedings of the Sixth International Conference on Pattern Recognition, pp.1023–1026, 1982.

[5] S. Tsujimoto and H. Asada, "Major components of a complete text reading system," in Document image analysis, pp.298–314, IEEE Computer Society Press, Los Alamitos, CA, USA, 1995.

[6] M. Okamoto, S. Sakaguchi, and T. Suzuki, "Segmentation of touching characters in formulas," DAS '98: Selected Pa-

pers from the Third IAPR Workshop on Document Analysis Systems, London, UK, pp.151–156, Springer-Verlag, 1999.

[7] H. Lee and M. Lee, "Understanding mathematical expressions using procedure-oriented transformation," Pattern Recognition, vol.27, no.3, pp.447–457, 1994.

[8] U. Garain and B.B. Chaudhuri, "Segmentation of touching symbols for ocr of printed mathematical expressions: An approach based on multifactorial analysis," ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition, Washington, DC, USA, pp.177–181, IEEE Computer Society, 2005.

[9] S. Mori, H. Nishida, and H. Yamada, Optical Character Recognition, John Wiley & Sons, 1999.

[10] M. Suzuki, S. Uchida, and A. Nomura, "A ground-truthed mathematical character and symbol image database," ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR'05), Washington, DC, USA, pp.675–679, IEEE Computer Society, 2005.

[11] M. Suzuki, C. Malon, and S. Uchida, "Databases of mathematical documents," Research Reports on Information Science and Electrical Engineering of Kyushu University, vol.12, no.1, pp.1–8, 2007.