Available online at www.sciencedirect.com

SCIENCE DIRECT®

Pattern Recognition 38 (2005) 1961 – 1975

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

A novel adaptive morphological approach for degraded character image segmentation

Shigueo Nomura[a,*], Keiji Yamanaka[b], Osamu Katai[a], Hiroshi Kawakami[a], Takayuki Shiose[a]

[a]*Department of Systems Science, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan*
[b]*Faculty of Electrical Engineering, Federal University of Uberlândia, Uberlândia-MG 38400-902, Brasil*

Received 30 July 2004; accepted 7 January 2005

Abstract

This work proposes a novel adaptive approach for character segmentation and feature vector extraction from seriously degraded images. An algorithm based on the histogram automatically detects fragments and merges these fragments before segmenting the fragmented characters. A morphological thickening algorithm automatically locates reference lines for separating the overlapped characters. A morphological thinning algorithm and the segmentation cost calculation automatically determine the baseline for segmenting the connected characters. Basically, our approach can detect fragmented, overlapped, or connected character and adaptively apply for one of three algorithms without manual fine-tuning. Seriously degraded images as license plate images taken from real world are used in the experiments to evaluate the robustness, the flexibility and the effectiveness of our approach. The system approach output data as feature vectors keep useful information more accurately to be used as input data in an automatic pattern recognition system.
© 2005 Pattern Recognition Society. Published by Elsevier Ltd. All rights reserved.

Keywords: Mathematical morphology; Adaptive segmentation; Feature extraction; Fragmented characters; Overlapped characters; Connected characters; Degraded images; Pattern recognition

1. Introduction

The automation of character segmentation and features extraction from images acquired in real time, such as license plate images, is quite a complex problem because of the quality degradation. Character segmentation carries out an important role in a pattern recognition system and it is responsible to split the word (set of digits and letters) image into its component characters. If these characters could be accurately detected, segmented, and recognized, they would be a valuable source for recognizing or retrieving words.

1.1. Related work

In recent years, many character segmentation techniques have been developed for texts in printed documents [1–4]. Essentially, those techniques are based on heuristics where the text contains horizontal lines and the characters have proportional sizes with uniformly well-separated characters. An example of character segmentation technique [3] based on vertical projection applied to a machine printed text is shown in Fig. 1. Also, there are recognition-based segmentation algorithms that require a recognizer, which is used to validate the segmentation process. If the character is recognized then the segmentation is accepted, otherwise segmentation is

* Corresponding author. Tel.: +81 75 753 3592; fax: +81 75 753 5042.
E-mail addresses: shigueo@sys.i.kyoto-u.ac.jp (S. Nomura), keiji@ufu.br (K. Yamanaka), katai@i.kyoto-u.ac.jp (O. Katai), kawakami@i.kyoto-u.ac.jp (H. Kawakami), shiose@i.kyoto-u.ac.jp (T. Shiose).

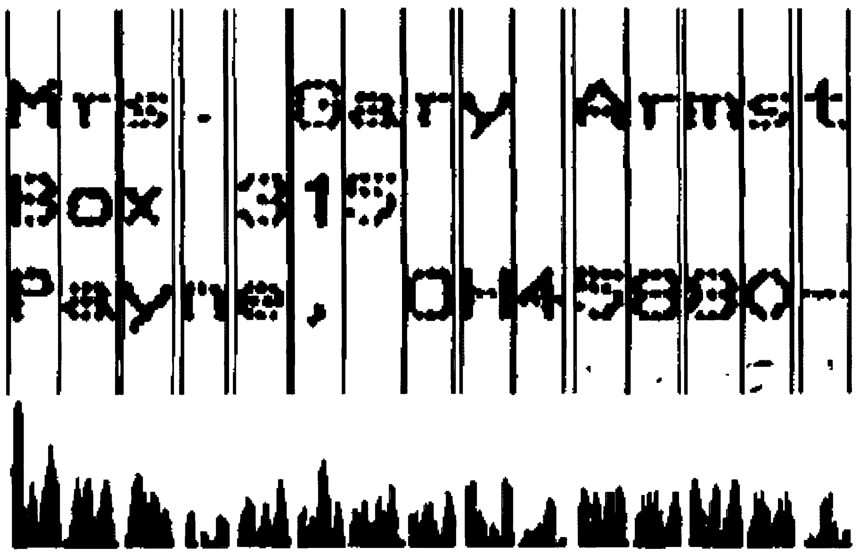


Fig. 1. Example of character segmentation process based on histogram for machine printed words [3].

re-applied [5,6]. These recognition-based segmentation algorithms are not only time consuming, but their outputs are heavily depending on the character-recognizer process.

So, the existing techniques and algorithms for segmenting characters have not been effective when applied to real-world degraded images. In other words, they have not been able to simultaneously treat the character problems and adaptively segment the fragmented, overlapped or connected characters to accurately extract their features from these degraded images.

1.2. Our approach

In this paper, a new adaptive segmentation and extraction approach is proposed based on mathematical morphology (MM) operators in conjunction with heuristics that determine the potential segmentation points. The license plate digitized images, that represent the degraded images, are acquired from real scenes of moving vehicles taken by cameras installed at the streets [7] in Uberlândia city. Fig. 2 shows a sample of these digitized images.

The images are often acquired under dubious light and focusing conditions, and are often badly degraded images that include serious problems such as fragmented, overlapped or connected characters in their corresponding binary images. Examples of these problems are shown in Fig. 3. In the binary image of Fig. 3(a), its corresponding histogram (vertical projection obtained by counting the number of shape pixels in each column of image matrix) provides only one segment (instead of three) because the three letters are overlapped. Fig. 3(b) shows the binary image with connected characters that cannot be segmented using only its histogram data. Finally, the binary image in Fig. 3(c) presents fragmented characters, which result in unnecessary segments for the second and the last character when a natural segmentation is performed using the corresponding histogram.

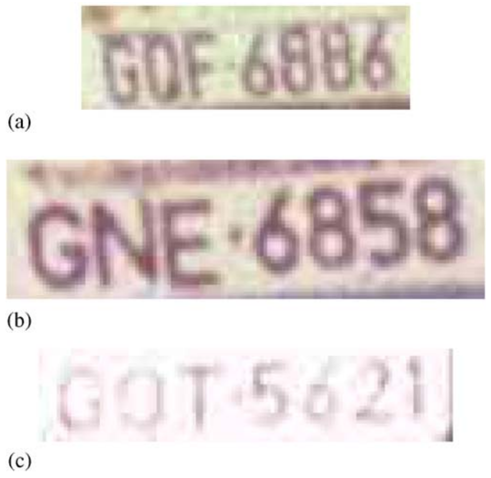


Fig. 2. Sample of the original degraded images used in this work.

Moreover, the degraded images used in this work have presented themselves with size and inclination variations of the characters. Thus, an adaptive system is required that can automatically solve all the problems mentioned above in real time. Among various techniques that have been proposed for image segmentation, the morphological segmentation technique is considered promising because it relies on MM operations that are very attractive for dealing with object-oriented criteria such as size and shape.

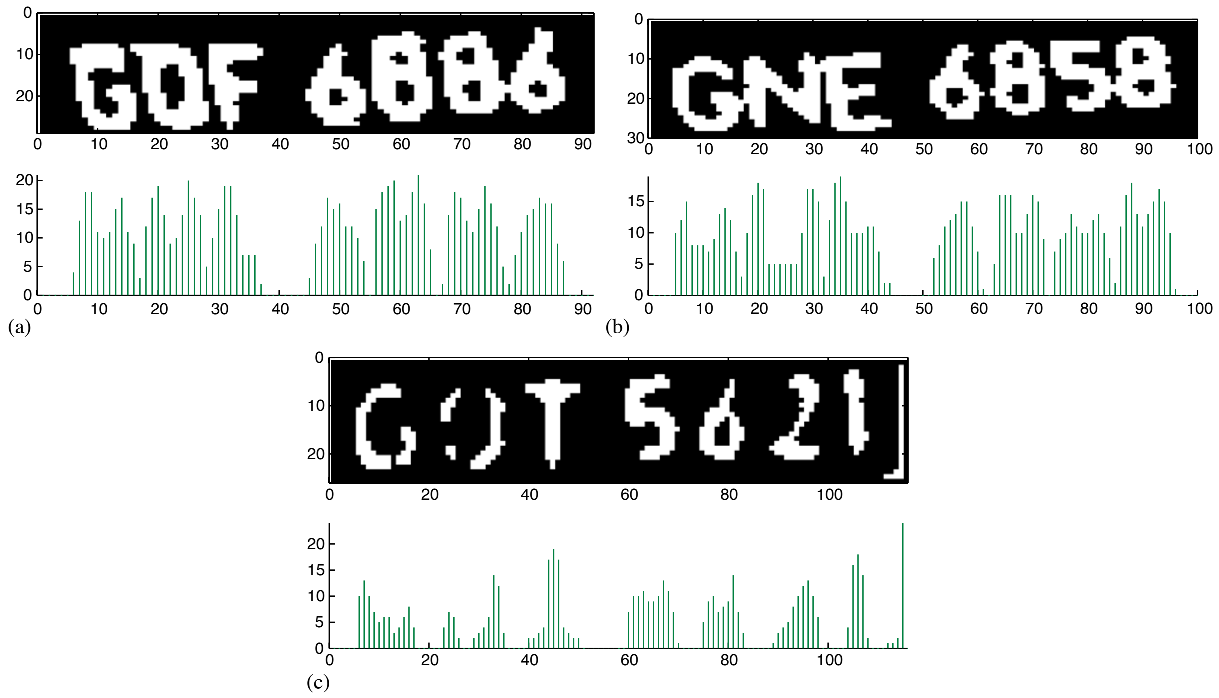


Fig. 3. Binary images and their corresponding histograms with (a) overlapped, (b) connected and (c) fragmented character problems.

Also, in this paper, character position identification and segmentation followed by extraction process constitute another contribution of the proposed approach. The simple segmentation of characters is insufficient for systems [8] to recognize or to retrieve codes of license plates; it is necessary to obtain the position of each character (feature vector) in the code. Using the position information, it is possible to make up for some unrecognizable character problems caused by poor input image quality. The following geometric information applies to license plate images:

(a) A set of letters occupies the first region (almost half of a plate) and a set of digits occupies the second region on each plate.
(b) All the plates have three letters and four digits that constitute their codes.

The paper is structured as follows. Developed algorithms for the proposed approach using MM operations are presented in Section 2. The steps of the proposed approach based system model to obtain binary images, to remove noise, and to perform the adaptive segmentation of characters are described in detail in Section 3. Section 4 discusses and evaluates the segmentation results with the proposed approach by using the geometric information about license plate images, and by executing a pattern recognition system based on artificial neural networks. Finally, the work is concluded in Section 5.

2. Elements of mathematical morphology

MM is a versatile and powerful image analysis technique for extracting useful image components with respect to representation and description of regions, such as boundaries, size, area, shape, or connectivity in segmentation-oriented image processing [9]. Morphological operators are widely used in image processing and analysis [10,11].

The language of MM [10,12] is a set theory that represents objects in an image. In this work, a new morphological approach was applied to support the shape (character) segmentation and extraction from degraded images.

Moreover, MM works better than other techniques, such as artificial neural networks when the task is to provide segmentation, because:

(a) MM does not require complex and heavy mathematical calculations.
(b) MM avoids the training stage.
(c) MM does not need training parameters.
(d) MM provides more reliable results.

However, most of existing morphological approaches employ a top-down technique [13,14]. In this case, if two shapes have ambiguous boundaries due to low contrast, they may be considered to belong to the same shape in the sense of a homogeneity or similarity measure. Thus, sometimes the top-down technique tends to imprecisely segment the shapes. This tendency may cause a serious problem in perceptual applications such as realistic automatic recognition of degraded images, where semantic object shapes should be defined accurately.

In this work, morphological image processing is used to achieve the following tasks:

(a) Location of separating lines in overlapped characters.
(b) Determination of the baselines for crossing points in connected characters.
(c) Location of the isolated noises to remove them from binary images. The images can subsequently be processed to precisely perform enhancement, edge detection, thinning, thickening, and segmentation of their objects (shapes).

MM operations [12] for dilation, erosion and "hit-or-miss" transform of objects are used for implementing thickening, thinning and pruning algorithms. Also, crossing point detection has been developed and segmentation cost, that is similar to splitting cost in handwritten words [15], has been calculated to increase the accuracy of segmentation process.

2.1. Thinning algorithm

A good thinning algorithm should preserve the connectivity and remove pixels from a shape in an image until the width of the skeleton to unity [16,17]. The MM thinning is considered convenient and it is used to reveal crossing points from connected characters in this work.

The thinning [18] of a binary image X by a structuring element B is denoted by $X \times B$ and defined as the set difference between X and the hit-or-miss transform of X by B that is denoted by $X * B$:

$$X \times B = X - (X * B). \quad (1)$$

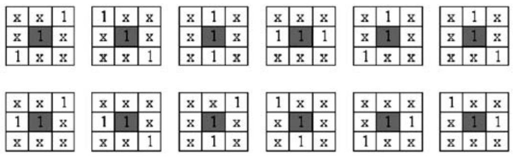


Fig. 4. Structuring elements for detecting interconnection pixel. Where x represents the pixel 0 or 1, 1 represents the pixel from the 8-connected object.

2.2. Thickening algorithm

The thickening algorithm is the morphological dual of the thinning algorithm.

The thickening algorithm works by adding background pixels to the exterior of shapes until doing so would result in previously unconnected objects being 8-connected. In this work, it is used to determine boundaries between shapes in binary images.

The thickening [18] of a binary image X by a structuring element B is denoted by $X \times B$ and defined as the union of X and the hit-or-miss transform of X by B:

$$X \otimes B = X \cup (X * B). \quad (2)$$

2.3. Pruning algorithm

The pruning algorithm is an essential complement to the thickening algorithm, and is used to clean up parasitic components from 8-connected objects. The solution is based on suppressing a parasitic branch by successively eliminating its end point [12].

Particularly in this work, there are situations in which a branch must not be suppressed. For this reason, an original pruning algorithm was developed to obtain the necessary "image without spurs" as reference data for segmentation.

The 12 structuring elements in Fig. 4 have been created to detect a pixel of interconnection. A pixel of interconnection

Table 1
A formal statement of the pruning algorithm

X	The binary image with 8-connected objects
$p(r, s)$	The value of a pixel from X
(r, s)	The coordinates of $p(r, s)$
(x, y)	The coordinates of a pixel from the branch
(nx, ny)	The coordinates of the neighbor of a pixel
k	The neighbor counter
S	The matrix of 12 structuring elements
V	The vector for a matrix of a pixel with its eight neighbors
$GetV(p(x, y))$	The function to obtain the vector V of $p(x, y)$
$Relev(S, V)$	The boolean function to verify the relevance of $p(x, y)$. If $p(x, y)$ is relevant, then the output is true
$elim$	The boolean flag to indicate the elimination or not
$SaveN$	The function to save the coordinates of a new neighbor
$LoadN$	The function to load the coordinates of an older neighbor

Table 2
The pruning algorithm

```
Initialize $k = 0$
Initialize $S$
For each pixel of $X$
  If $p(r, s) > 0$
    Compute $elim$ = true
    Initialize $(x, y) = (r, s)$
    Execute $V = GetV(p(x, y))$
    While $elim$
      If $k > 0$ (there is a neighbor to analyze)
        Execute $(x, y) = LoadN$
        Compute $k = k - 1$
        Execute $V = GetV(p(x, y))$
      Else
        Compute $elim$ = false
      If $p(x, y)$ is not $Relev(S, V)$
        Compute $p(x, y) = 0$
        For each neighbor of $p(x, y)$
          If $p(nx, ny) > 0$
            Execute $SaveN$
            Compute $elim$ = true
End of algorithm
```

must belong to the 8-connected object, that is, it must not be an end point.

The definition of a relevant pixel is used to eliminate a parasitic component. A pixel is considered relevant if it is in accordance with one of the two following conditions; that is, it must not be removed.

(1) The pixel belongs to the boundary of the image.
(2) It is an interconnection pixel.

Furthermore, all eight neighbors of each non-relevant pixel (non-zero) are analyzed, after which it is possible to maintain the relevant pixels so that only a parasitic branch is eliminated.

Table 1 presents a formal statement of the pruning algorithm and Table 2 presents the pruning algorithm.

3. The proposed system model and its description

The steps in Fig. 5 can be described in the following way:

3.1. Step 1—binary image obtaining

Binary images are obtained using new adaptive methods [7,19] applied to color license plate images. Fig. 6 displays a binary image obtained from a license plate image.

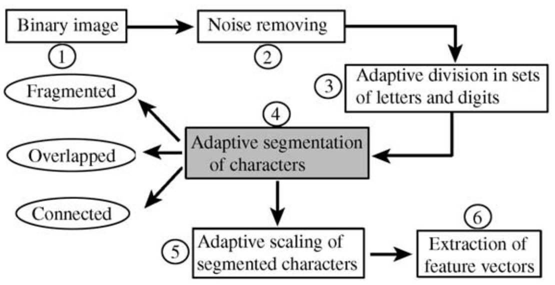
3.2. Step 2—isolated noise removing

The isolated noise in Fig. 6 is removed. Noise is located by applying the thickening and pruning algorithms to the binary image in Fig. 6. The boundaries (gray lines) in Fig. 7 locate noise in the image of Fig. 6, while the image of Fig. 8 is the cleaned image after removing the noise.

3.3. Step 3—adaptive division of codes

Geometric information about code arrangements on plates is used for adaptive division of these codes into two sets: letters and digits. A vertical histogram of the binary image is used to locate that division line.

The initial location (coordinate y) of the division line is estimated as the midway point of the total width.


Binary image
1
Noise removing
2
Adaptive division in sets of letters and digits
3
Fragmented
4
Adaptive segmentation of characters
Overlapped
Connected
5
Adaptive scaling of segmented characters
6
Extraction of feature vectors
Fig. 5. The architecture of the proposed system to adaptively segment characters in degraded license plate images.

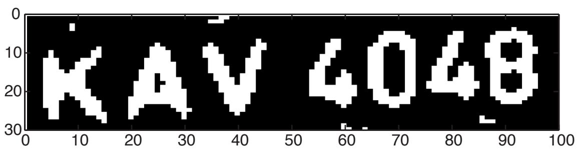


Fig. 6. Noisy binary image obtained from a degraded license plate image.

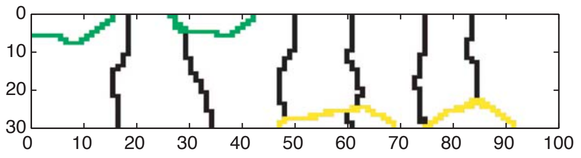


Fig. 7. Noise located after applying the thinning algorithm.

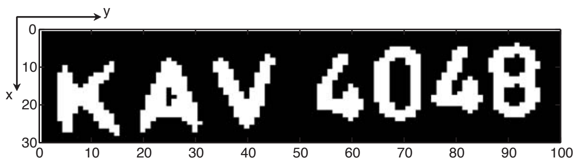


Fig. 8. Binary image without noise.

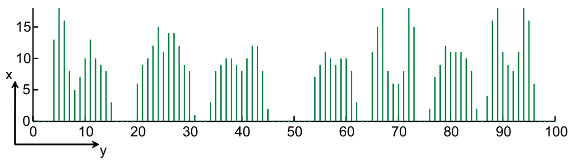


Fig. 9. Vertical histogram that results from the image in Fig. 8, the division line coincided with the midway point of the width (100).

Table 3
A formal statement of the algorithm to find the line that divides the code into two sets

Img	The binary image
w	The value in number of columns for the total width of *Img*
V	The vector for the vertical histogram of *Img*
p	The position or index to indicate the division line

Table 4
The algorithm to find the line that divides the code into two sets

```
Get $Img$, $V$, $w$
Estimate $p = w/2$
While $Vp$ not null and $p > 0$
    Compute $p = p - 1$ (shift to the left)
If $p < w/3$ or $p = 0$
    Discard the image
End of algorithm
```

The column in the image is shifted by one position to the left until a coordinate y is found in the vertical histogram whose corresponding coordinate x is null. The left shift is necessary because the set of letters always occupy a smaller space than the set of digits in the analyzed images. In case of the histogram in Fig. 9 that results from the image in Fig. 8, the division line coincided with the midway point of the total width (100) of the binary image. If the division line's position falls within a range size smaller than a third of the image's total width, then this image is discarded. Table 3 presents a formal statement of the algorithm to find the position that divides the code into two sets and Table 4 presents the corresponding algorithm.

3.4. Step 4—adaptive segmentation of characters

This is the main step in executing the adaptive segmentation of characters in the proposed approach. The following operations are carried out to segment the characters.

3.4.1. Searching for natural segment points

A natural segment is a space clearly limited by columns whose values are null at its initial and final coordinates. The natural segment is determined directly based on information in those columns from the binary image's vertical histogram.

Table 5
A formal statement of the algorithm to obtain natural segments

Img	The binary image
ord	The order of the character on the number plate
V	The vector for the vertical histogram of *Img*
ic	The initial coordinate of a natural segment from *Img*
InitC(*V*)	The function to obtain *ic* based on *V*
fc	The final coordinate of a natural segment from *Img*
FinC(*V*)	The function to obtain *fc* based on *V*
Save(*ic*,*fc*,*ord*)	The procedure to save the coordinates of the natural segment and its order in a matrix

Table 6
The algorithm to obtain natural segments

```
Initialize $ord = 0$
Get $Img$, $V$
While is not the end of $V$
    Execute $ic = InitC(V)$
    Execute $fc = FinC(V)$
    Compute $ord = ord + 1$
    Execute $Save(ic,fc,ord)$
End of algorithm
```

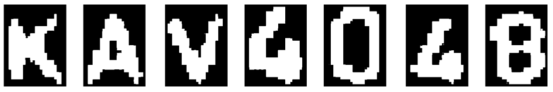


Fig. 10. Characters naturally segmented by using the vertical histogram in Fig. 9.

Table 5 presents a formal statement of the algorithm to obtain natural segments and Table 6 presents the algorithm to obtain the natural segments.

For example, the seven segments in Fig. 10 can be found by using the vertical histogram in Fig. 9.

3.4.2. Merging fragments that belong to the same character

Prior knowledge of the maximum quantity of segments for each set (letters or digits) is needed to decide whether the merging is necessary. In the present work, we have designated 3 to the set of letters and 4 to the set of digits based on geometric information on available data. If the quantity of fragments from an analyzed set is larger than the limited quantity of segments, then the merging must be performed. The merging corresponds to the execution of the function to shift the indexes of fragments if the standard deviation calculated by Eq. (6) is larger than the tolerance value (equal to 1 in this work). The segments are considered to be fragments during the merging process. Table 7 presents a formal statement of the merging process and Table 8 presents the merging process.

Table 7
A formal statement of the merging process

j	The index of each fragment
Nf	The number of fragments
MaxF	The maximum quantity of segments for each set
S	The matrix of initial (1) and final (2) coordinates for the fragment *j*
M	The vector of coordinates for medium points corresponding to the fragments
k	The index of each distance
D	The vector of distances between two consecutive medium points
AD	The average distance for all the fragments
SD	The vector of results by calculating the standard deviation between each distance and the average distance
Max(*SD*)	The function to return the largest element in *SD* and its corresponding index
maxSD	The value of largest element
indMax	The value of index for *maxSD*
Shift(*indMax*)	The function to shift the indexes of fragments after *indMax*. The effect of this function is the merging of fragments

Table 8
The merging process

```
Get $Nf$
While $Nf > MaxF$
  For each fragment $j$
    Calculate the medium point $M_j$
  For each two consecutive medium points
    Calculate the distance $D_k$
  Calculate the average distance $AD$
  For each distance $k$
    Calculate the standard deviation $SD_k$
  Execute $(maxSD, indMax) = Max(SD)$
  If $maxSD > 1$
    Execute $Shift(indMax)$
  Compute $Nf = Nf - 1$
End of algorithm
```

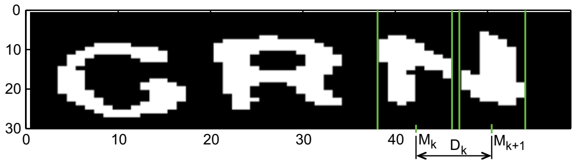


Fig. 11. Set of letters with fragments.

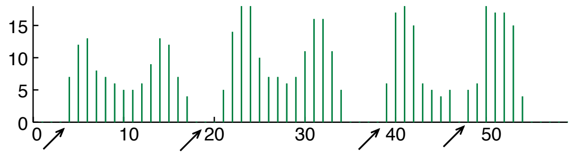


Fig. 12. Vertical histogram to indicate the initial index of a natural segment or a fragment in the image of Fig. 11.

3.4.2.1. *Mathematical operations for the merging process*

(a) Determination of the medium point for each fragment:

$$M_j = \frac{S_{j1} + S_{j2}}{2}. \tag{3}$$

(b) Calculation of the distance between two consecutive points:

$$D_k = M_{k+1} - M_k. \tag{4}$$

(c) Calculation of the average distance among all the distances calculated on task *b*:

$$AD = \frac{\sum_{k=1}^{Nf-1} D_k}{Nf - 1}. \tag{5}$$

(d) Calculation of the standard deviation between each distance and the average distance:

$$SD_k = \left(\frac{1}{Nf-1}\sum_{k=1}^{Nf-1}(D_k - AD)^2\right)^{1/2}. \tag{6}$$

In the merging process, the distance between two medium points from the fragments was used instead of the width of each character. Using the width is inconvenient because this width depends on the character type. For example, the digit 1, or the letter I is narrower than other characters.

The arrows on the histogram in Fig. 12 indicate the initial index of each natural segment or each fragment. This histogram originated from the image in Fig. 11. We note in Fig. 12 that the fourth arrow indicates the initial index for the second fragment (a fragment that corresponds to the third letter in the image of Fig. 13). The previously presented algorithm detects the index of the red line between two fragments of the third letter in Fig. 13. Then, the fragments are merged and the result of the merging process is verified in Fig. 14.

3.4.3. *Locating fixed segments*

The condition is that a fixed segment must contain only one character from the license plate. The fixed segment is a

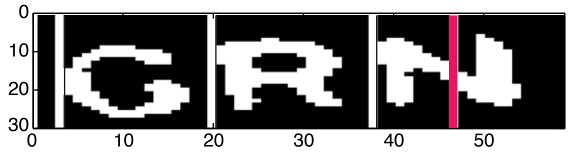


Fig. 13. Red segmentation line between the fragments before the merging process.

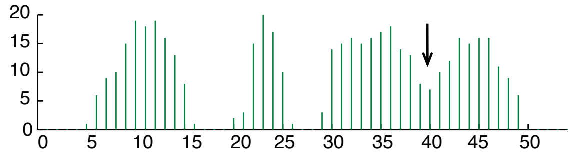


Fig. 16. Vertical histogram to locate the problem of connected characters in Fig. 15.

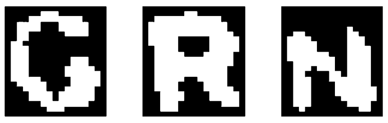


Fig. 14. Segmented characters after the merging process.

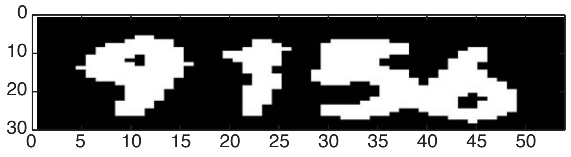


Fig. 15. Set of digits with connected characters.

zone delimited by two vertical lines and considered "fixed" because its width is equal for all the characters. The vertical histogram in Fig. 16 is used as a base for locating the fixed segments. The valleys presented in that histogram are important references to locate the fixed segments.

The main objective of "the algorithm to locate fixed segments" (Table 10) is to adaptively determine the best distribution of segmentation lines based on the calculation of segmentation costs for each distribution. The distribution includes the optimal width of fixed segments and the optimal position of each segmentation line to reach the minimum cost. Therefore, an iteration of the algorithm shifts the segmentation lines by keeping the width of fixed segments and saves the corresponding positions for the minimum cost of these conditions. The range of shifting depends on the total width of the image. Another iteration increases the width of each fixed segment and saves the width for the minimum cost of these conditions. Table 9 presents a formal statement of the algorithm to locate fixed segments and Table 10 presents the corresponding algorithm.

3.4.3.1. Mathematical operations for locating fixed segments

(a) Determination of the maximum width of a fixed segment:

$$maxW = \frac{imgW}{nFixS}, \tag{7}$$

Table 9
A formal statement of the algorithm to locate fixed segments

initC	The column that corresponds to the division line (Tables 3 and 4)
maxW	The maximum width of a fixed segment. It is determined by Eq. (7)
segCost	The segmentation cost of a column that corresponds to a segmentation line
inInd	The index column for the first segmentation line
w	The width of a fixed segment
Vect	The column vectors (positions) of segmentation lines in an image
nFixS	The number of fixed segments
IndVect (*inInd*, *w*, *nFixS*)	The function to obtain *Vect* according to the *inInd*, *w*, and *nFixS*
TotCost(*Vect*)	The function to calculate *segCost* for each element of *Vect*, and to obtain the sum of all the costs
totC	The value of the sum of all the costs
MatSC	The matrix to save *inInd* and *totC*
SaveIC (*inInd*, *totC*, *MatSC*)	The procedure to save the reference position of segmentation lines and the corresponding total cost into *MatSC*
MinCostS (*MatSC*, *inInd*, *minC*)	The procedure to obtain the minimum total cost (*minC*) and its corresponding inInd from *MatSC*
MatWC	The matrix that contains the values of *w*, *inInd*, *minC*
SaveWC (*w*, *inInd*, *minC*, *MatWC*)	The procedure to save the values of *w*, *inInd*, and *minC* in to *MatWC*
MinCostW (*MatWC*, *w*, *inInd*)	The procedure to obtain the minimum total cost and its corresponding values of *w* and *inInd* from *MatWC*

Table 10
The algorithm to locate fixed segments

```
Initialize $initC$, $nFixS$
Calculate $maxW$
For $w =$ initial width to $maxW$
    Compute $inInd = initC$
    For each position of segmentation lines by $imgW$
        Execute $Vect = IndVect(inInd, w, nFixS)$
        Execute $totC = TotCost(Vect)$
        Execute $SaveIC(inInd, totC, MatSC)$
        Shift the value of $inInd$
    Execute $MinCostS(MatSC, inInd, minC)$
    Execute $SaveWC(w, inInd, minC, MatWC)$
Execute $MinCostW(MatWC, w, inInd)$
Execute $Vect = IndVect(inInd, w, nFixS)$
End of algorithm
```

where *imgW* is the total width of the image; *nFixS* is three for the set of letters, and four for the set of digits.

(b) Calculation of the segmentation cost of a column:

$$segCost_c = \sum_{l=1}^{lMax} p(l, c) \tag{8}$$

where c is the column index in the image; l is the row index in the image; $p(l, c)$ is the value of a pixel with coordinates (l, c); *lMax* is the total quantity of rows in the image.

The segmentation cost of a column is the number of pixels that a vertical segmentation line picks up from an object. In this case, it is calculated for each segmentation line corresponding to fixed segments whose widths varies from the initial width (nine columns for the set of digits and 11 for the set of letters) by *maxW*.

For example, *segCost* of the second line on the image in Fig. 17. By visual inspection, we can verify that the vertical lines on the image in Fig. 17 hit a greater quantity of object pixels than the vertical lines on the image in Fig. 18. Consequently, the total cost of segmenting the image in Fig. 18 is lower than the total cost for the one in Fig. 17. Thus, the distribution of segmentation lines in Fig. 18 is considered to be the optimal one.

We can verify the optimal distribution of segmentation lines after applying the algorithm to locate the fixed segments in the image of Fig. 18. is larger than *segCost* of the corresponding second line on the image in Fig. 18.

3.4.4. Separating overlapped characters

The overlapped characters (the three letters) as in Fig. 19 cannot be divided by using only the vertical histogram presented in Fig. 20. Noting that the columns indicated by the arrows are not nulls, it is not possible to obtain the natural segments using information from the vertical histogram. The idea is to locate the lines to separate the overlapped characters by using the thickening algorithm presented in Section 2. The arrows on the image in Fig. 21 indicate the

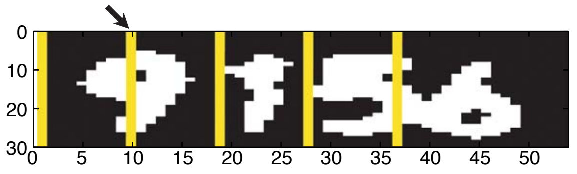


Fig. 17. Distribution of segmentation lines before analyzing the segmentation cost.

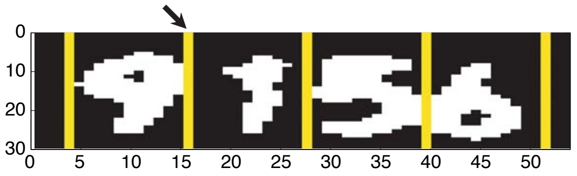


Fig. 18. Optimal distribution of segmentation lines after analyzing the segmentation cost.

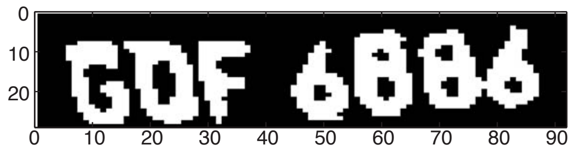


Fig. 19. Image with three overlapped characters.

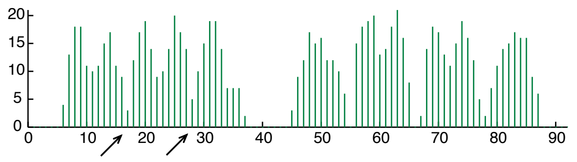


Fig. 20. Vertical histogram for the image with overlapped characters in Fig. 19.

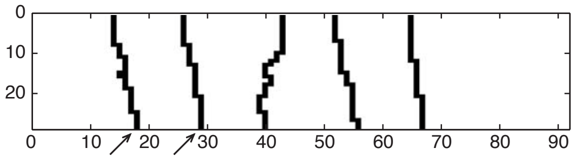


Fig. 21. Separating lines after the thickening.

separating lines performed by applying the thickening algorithm to the image in Fig. 19. Table 11 presents a formal statement of the algorithm to separate overlapped characters and Table 12 presents the corresponding algorithm.

3.4.4.1. Mathematical operations for separating overlapped characters

(a) Calculation of the average column for the pixels from the separating line:

$$avCol = \frac{\sum_{c=1}^{nP} VectC(c)}{nP}, \tag{9}$$

Table 11
A formal statement of the algorithm to separate overlapped characters

ImgB	The binary image that contains the overlapped characters
ImgS	The image after applying the thickening algorithm
GetVC(*ImgS*)	The function to obtain the vector of coordinates for a separating line of *ImgS*
VectC	The vector that contains the columns for the pixels from a separating line
avCol	The average column for the pixels from the separating line
VectF	The vector of columns that correspond to the segmentation lines of fixed segments
VectSD	The vector of values of standard deviation between each element of *VectF* and *avCol*
MinSD(*VectSD*)	The function to obtain the minimum value from *VectSD*
mSD	The minimum standard deviation value
tol	The tolerance value for the standard deviation in the number of columns to include the separating line. In the present work, this value is 2
MatC (*ImgB*,*VectC*)	The function to extract the character from *ImgB* by using *VectC* as reference data
MC	The matrix of pixels for a separated character

Table 12
The algorithm to separate overlapped characters

```
Get $ImgB$, $ImgS$
Initialize $VectF$
For each separating line
    Execute $VectC = GetVC(ImgS)$
    Calculate $avCol$
    Calculate $VectSD$
    Execute $mSD = MinSD(VectSD)$
If $mSD < tol$
        Execute $MC = MatC(ImgB, VectC)$
End of algorithm
```

where nP is the number of pixels in the separating line; c is the index for each column.

(b) Determination of the vector for values of standard deviation between each element of *VectF* and *avCol*.

$$VectSD_e = \frac{\|VectF_e - avCol\|}{2}, \tag{10}$$

where e is the index for each element from *VectF*.

3.4.5. Segmenting connected characters

The image in Fig.15 presents two connected characters. In this case, the vertical histogram in Fig. 16 cannot provide the exact column (in the range close to the arrow) for natural segmentation as the natural segmentation is disabled.

To solve the problem of connected characters, we have performed the following actions:

(a) Locating crossing points: The crossing points [20] are located in skeletons of characters and are used as reference points to segment the connected characters. In this work, the thinning algorithm presented in Section 2 obtains the skeleton of the connected characters. A set of 12 structuring

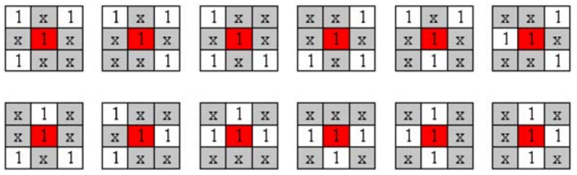


Fig. 22. Structuring elements for locating the crossing points. Where x represents the pixel 0 or 1; 1 represents the pixel from the skeleton.

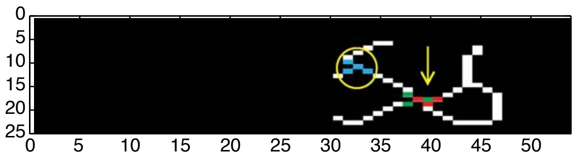


Fig. 23. Crossing points in a thinned image.

elements (presented in Fig. 22) has been provided to locate a crossing point on the skeleton line.

The skeleton of the connected characters in the image of Fig. 15 is presented in Fig. 23.

(b) Discarding unnecessary crossing points: Unnecessary crossing points, like the one marked by a circle in Fig. 23, must be discarded during the process of segmenting the connected characters.

We then consider three criteria.

First—The image in Fig. 24 results by applying the thickening algorithm to the image in Fig. 15. That thickened image contains the first reference, that is, the vertical line that connects the upper edge as indicated in Fig. 24. The vertical line is due to the space between the first and second connected characters. Its corresponding column is a reference for crossing points close to it. If the difference or distance between columns is smaller than a tolerance value (in this

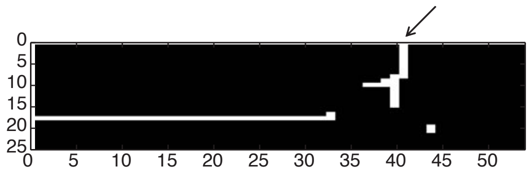


Fig. 24. The first reference for selecting the crossing points.

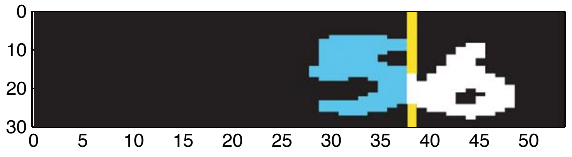


Fig. 25. Segmentation line before finding the cost.

Table 13
A formal statement for the second criterion

Img	The binary image that contains the connected characters
VectCP	The vector of columns for the crossing points
col	The column of *Img* corresponding to a segmentation line
Cost(*col*)	The function to calculate the cost for *col*
valC1	The cost value for a segmentation line
valC2	The cost value for the next one
k	The index of a column from *Img*
bestC	The optimal column with respect to the second criterion. This is the new value of *col*

work we adopted the value 2), then the analyzed crossing point is accepted.

Second—The calculation of the segmentation line cost is the second criterion to select the best crossing point; that is, the segmentation line whose cost is minimum. We can verify an inappropriate segmentation line in Fig. 25.

The process for calculating the cost of a segmentation line uses the same Eq. (8) that was presented in "locating fixed segments." Then, we can obtain the segmentation line whose cost is minimum after calculating the costs for all the columns that correspond to the considered range in an image. Table 13 presents a formal statement for the second criterion and Table 14 presents the algorithm for this criterion.

In Fig. 25, we have a segmentation line that was located without calculating the cost of segmentation. In Fig. 26, we can verify the best segmentation line that was obtained after considering the second criterion. Consequently, we can appropriately segment the connected characters.

Third—Segmentation lines of fixed segments are used as reference data for the third criterion. The column of a crossing point is compared with the column of the nearest segmentation line. If the standard deviation between the values of two columns is greater than a tolerance value (in this work, 1), then the corresponding crossing point is discarded. Table 15 presents a formal statement for the third criterion and Table 16 presents the algorithm for this criterion.

Table 14
The algorithm for the second criterion

```
Get $Img$
For each $col$ from $VectCP$
  Execute $valC1 = Cost(col)$
  Initialize $k = col + 1$
  Repeat
    Execute $valC2 = Cost(k)$
    If $valC1 >= valC2$
      $valC1 = valC2$
      $k = k + 1$
  until $valC1 < valC2$
  $bestC = k - 1$
End of algorithm
```

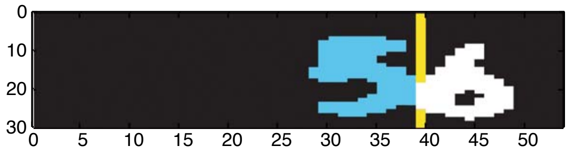


Fig. 26. After locating the best segmentation line.

Table 15
A formal statement for the third criterion

Img	The binary image that contains the connected characters
VectF	The vector of indexes for the fixed segments
VectCP	The vector of columns for the crossing points
col	The elements of *VectCP*
MinSD(*VectF*,*col*)	The function to calculate the standard deviation between *col* and each element of *VectF*, and to provide the minimum value of this standard deviation with *col*
minV	The minimum value of the standard deviation. This is the output value from *MinSD*
tol	The tolerance value for the standard deviation

Table 16
The algorithm for the third criterion

```
Get $Img$, $VectF$, $VectCP$
For each $col$
  Execute $minV = MinSD(VectF, col)$
  If $minV > tol$
    Delete $col$ from $VectCP$
End of algorithm
```

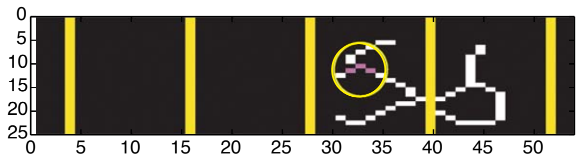


Fig. 27. A crossing point to be discarded.

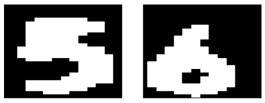


Fig. 28. Segmentation of the connected characters.

The last crossing point marked by a circle in Fig. 27 is discarded to obtain the segmented characters in Fig. 28. Table 17 presents a formal statement of the algorithm to segment connected characters and Table 18 presents the corresponding algorithm.

3.5. Step 5—adaptive scaling of the segmented characters

A simple adaptive scaling of segmented characters is processed considering the proportionality between the height and width of each character. In this work, the size of each character matrix has been normalized to 20 rows and 15 columns. Fig. 29 displays the before and after images; that is, the effects of applying the adaptive scaling.

Table 18
The algorithm to segment the connected characters

```
Get $Img$, $ImgSK$, $ImgT$
Initialize $SE$
Execute $VectCP = LocCP(ImgSK, SE)$
Execute $VeCT = GetCT(ImgT)$
Execute $Criter1(VectCP, VeCT)$
Execute $Criter2(VectCP, Img)$
Execute $Criter3(VectF, VectCP)$
Execute $MVC = MatC(Img, VectCP)$
End of algorithm
```

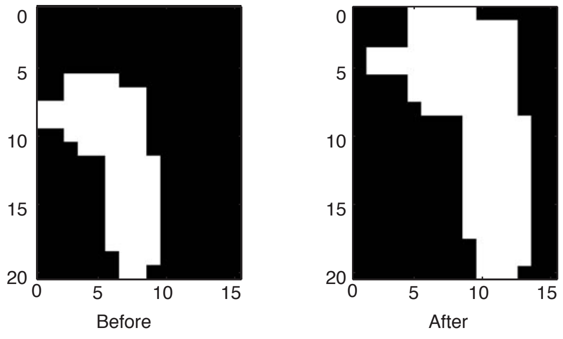


Fig. 29. Character images before and after the adaptive scaling.

3.6. Step 6—extraction of feature vectors

Each character matrix is converted to a vector with 300 elements. Its representation can be binary or bipolar depending on the pattern recognition system.

4. Experimental results

A test set of 1189 binary images was processed to evaluate the proposed approach. The criterion used to evaluate

Table 17
A formal statement to segment the connected characters

Img	The binary image that contains the connected characters
VectCP	The vector of columns for the crossing points
ImgSK	The skeleton of *Img* after applying the thinning algorithm
SE	The matrix of 12 structuring elements to locate the crossing points
LocCP(*ImgSK*,*SE*)	The function to obtain the coordinates for the crossing points in *ImgSK* by using *SE*
ImgT	The thickened image of *Img* using the thickening algorithm
VeCT	The vector of columns corresponding to the vertical lines in *ImgT*
GetCT	The function to obtain *VeCT* from *ImgT*
Criter1(*VectCP*, *VeCT*)	The procedure for filtering *VectCP* by using *VeCT* as a reference. This is based on the first criterion
Criter2(*VectCP*, *Img*)	The procedure for shifting the elements of *VectCP* according to the calculation of segmentation costs for *Img*
Criter3(*VectF*, *VectCP*)	The procedure for the discarding the elements of *VectCP* according to the calculation of standard deviations between *VectF* and *VectCP*
MatC(*Img*,*VectCP*)	The function to extract the characters from *Img* by using *VectCP* as reference data
MVC	The vector matrix for the segmented characters

Table 19
Quantity of characters per category

Image quantity	Overlapped characters	Connected characters	Fragmented characters
1005	153	276	20

as correct segmented character is based on the geometric information as described in Section 1.2. If the contents ofthe feature vector match the corresponding characters by respecting the position in the plate code then the segmentation is considered as valid. Also, the segmentation results are verified by a recognition system [8]. However, recognition is not affecting the decision making of the segmentation process. A set of 1005 images from the test set was correctly segmented and the feature vectors extracted to be used by the posterior recognition system. Table 19 presents the quantity of overlapped, connected, and fragmented characters in the set of 1005 images. Examples of the automatic segmentation using the proposed approach are depicted in Fig. 30 where the binary image of each plate is obtained using the new adaptive methods [7,19] and the segmented characters are normalized to 20 rows and 15 columns.

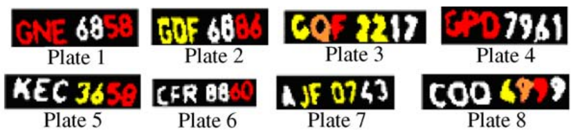


Fig. 31. Degraded images for segmentation and extraction test.

Three categories of characters for the performance analysis are presented in Fig. 31: isolated (white), overlapped (yellow) and connected (red). The orange color indicates a character that overlaps to the previous character, and connects to the posterior character. The results of the proposed approach applied to eight degraded images in Fig. 31 are represented by the corresponding images of feature vectors in Fig. 32.

Fig. 34 shows some results after applying the merging process to the degraded images in Fig. 33. The fragmented characters are represented by the blue color in Fig. 33.

A set of 184 images from the total quantity (1189) was considered difficult to segment by the proposed approach. Examples of this set are the plates 1 and 2 shown in Fig. 35. An excessive noise can be seen on the upper or lower side of the images.

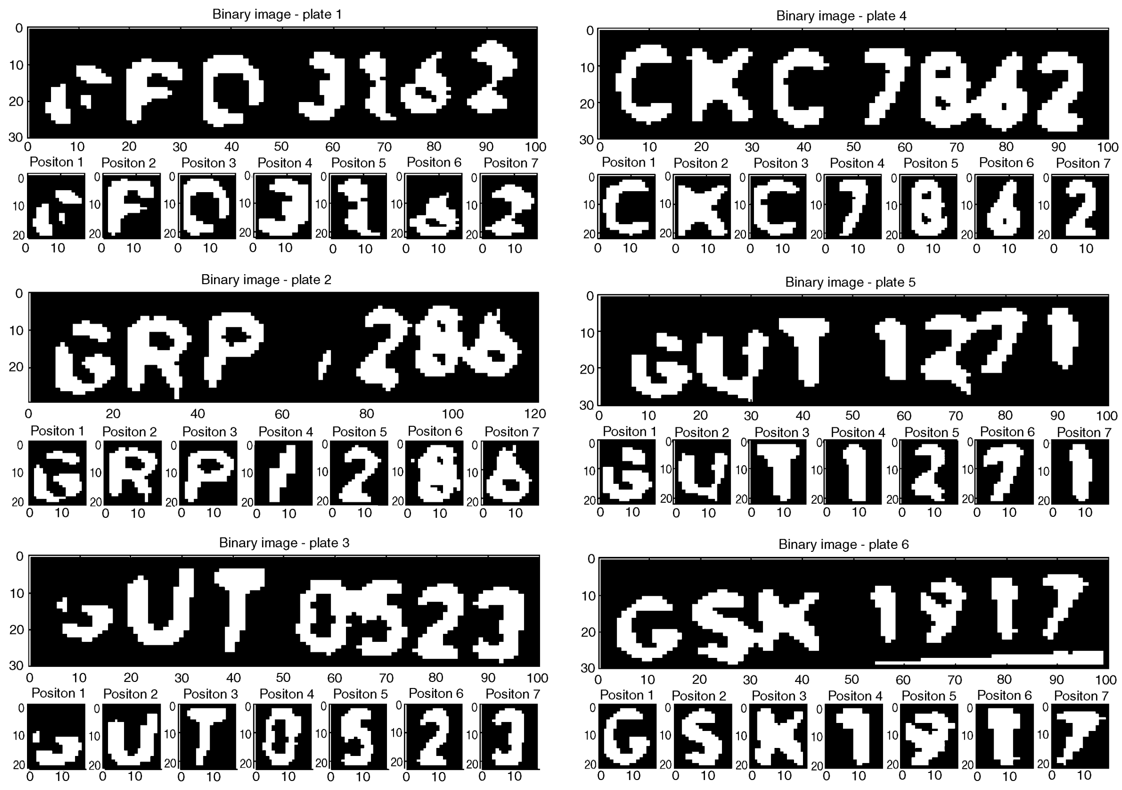


Fig. 30. Examples of segmentation results from degraded images.

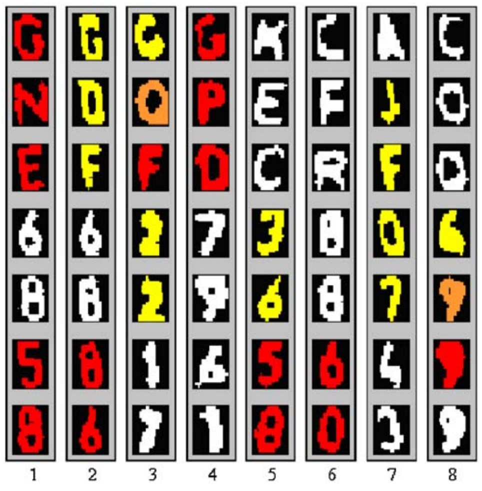


Fig. 32. Feature vectors of the extracted characters.

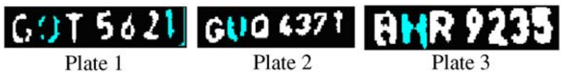


Fig. 33. Degraded images with fragmented characters.

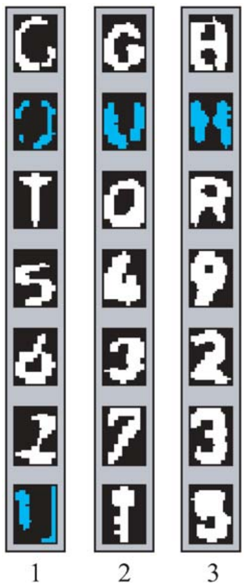


Fig. 34. Extracted feature vectors after the merging process.

5. Conclusion

In this paper, we have presented a new adaptive approach for segmenting characters and extracting feature vectors from degraded images taking advantage of mathematical morphology potential. A simple application system based on the proposed approach has successfully performed the automatic segmentation and extraction of isolated, fragmented, overlapped or connected characters from the set of 1005 real plate images processed during the experiments. Furthermore, the proposed approach performs well even if the characters have inclination problems or have different sizes. Moreover, the system can insert each extracted character in a standard matrix (20×15 in this work) by an appropriate scaling process, in addition to keeping the position of each character in the plate during the entire process. Also, a pattern recognition experiment based on artificial neural networks techniques showed that high recognition rate is reached by using the feature vectors extracted by the proposed approach. Thus, the novel adaptive morphological approach can be considered as a promising alternative method for accurately segmenting characters and extracting feature vectors from degraded images.

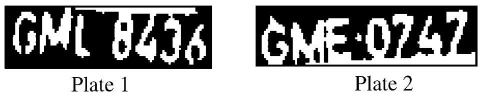


Fig. 35. Degraded images whose characters are difficult for segmenting.

References

[1] T. Taxt, P.J. Flynn, A.K. Jain, Segmentation of document images, IEEE Trans. Pattern Anal. Machine Intell. 11 (12) (1989) 1322–1329.

[2] Y. Lu, Machine printed character segmentation—an overview, Pattern Recognition 28 (1) (1995) 67–80.

[3] C.L. Tan, W. Huang, Z. Yu, Y. Xu, Image document text retrieval without OCR, IEEE Trans. Pattern Anal. Machine Intell. 24 (6) (2002) 838–844.

[4] M.C. Jung, Y.C. Shin, S.N. Srihari, Machine printed character segmentation method using side profiles, IEEE SMC '99 Conference Proceedings, vol. 6, No. 10, 1999, pp. 863–867.

[5] N. Arica, F. Yarman-Vural, A new scheme for off-line handwritten connected digit recognition, Proceedings of the International Conference on Pattern Recognition, vol. 1, 1998, pp. 127–129.

[6] S. Lee, S. Kim, Integrated segmentation and recognition of handwritten numerals with cascade neural network, IEEE Trans. Systems Man Cybern. 29 (2) (1999) 285–290.

[7] S. Nomura, K. Yamanaka, O. Katai, New adaptive methods applied to printed word image binarization, Proceedings of the Fourth IASTED International Conference Signal and Image Processing, August 12–14, 2002, pp. 288–293.

[8] S. Nomura, New methods for image binarization and character segmentation applied to an automatic recognition system of number plates, M.S. Thesis, Faculty of Electrical Engineering, Federal University of Uberlândia, Uberlândia City, Brazil, 2002 (in Portuguese).

[9] H. Gao, W. Siu, C. Hou, Improved techniques for automatic image segmentation, IEEE Trans. Circuits Systems Video Technol. 11 (12) (2001) 1273–1280.
[10] J. Serra, Image Analysis and Mathematical Morphology, vol. 1, Academic Press, London, 1982.
[11] J. Serra, Image Analysis and Mathematical Morphology, Theoretical Advances, vol. 2, Academic Press, London, 1988.
[12] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley Publishing Company, Reading, MA, 1993.
[13] H.S. Park, J.B. Ra, Efficient image segmentation preserving semantic object shapes, IEICE Trans. Fund. E82-A (6) (1999) 879–886.
[14] P. Salembier, M. Pardas, Hierarchical morphological segmentation for image sequence coding, IEEE Trans. Image Process. 3 (5) (1994) 639–651.
[15] F. Kimura, S. Tsuruoka, Y. Miyake, M. Shridhar, A lexicon directed algorithm for recognition of unconstrained handwritten words, IEICE Trans. Inform. Systems E77-D (7) (1994) 785–793.
[16] A. Elnagar, R. Alhajj, Segmentation of connected handwritten numeral strings, Pattern Recognition 36 (2003) 625–634.
[17] B. Jang, R. Chin, One-pass parallel thinning: analysis, properties, and quantitative evaluation, IEEE PAMI 14 (11) (1992) 1129–1140.
[18] P. Soille, Morphological Image Analysis: Principles and Applications, Springer, Berlin, 1999.
[19] S. Nomura, K. Yamanaka, O. Katai, H. Kawakami, A new method for degraded color image binarization based on adaptive lightning on grayscale versions, IEICE Trans. Inform. Systems E 87-D (4) (2004) 1012–1020.
[20] L.S. Oliveira, E. Lethelier, F. Bortolozzi, R. Sabourin, A new approach to segment handwritten digits, Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition, 2000, pp. 577–582.

About the Author—SHIGUEO NOMURA (SM'03) received his B.E. degree in 1992 and M.Sc. degree in 2002 from the Faculty of Electrical Engineering at the Federal University of Uberlândia, Uberlândia, Brazil. He won a research scholarship from the Japanese Government (Monbukagakusho) in April 2002. He is presently a doctorate student at the laboratory on Theory of Symbiotic Systems from the Department of Systems Science, Graduate School of Informatics, Kyoto University. His research interests include image processing, computer vision, pattern recognition, and neuro-computing.

About the Author—KEIJI YAMANAKA received his B.E. degree in 1980 and M.E. degree in 1992 in Electrical Engineering from the Federal University of Uberlândia, Brazil and Ph.D. degree in 1999, in Electrical and Computer Engineering from Nagoya Institute of Technology, Nagoya, Japan. Now, he is a professor at the Faculty of Electrical Engineering of the Federal University of Uberlândia His research interests include neuro-computing, computational intelligence and pattern recognition.

About the Author—OSAMU KATAI received his B.E., M.E. and Ph.D. degrees in 1969, 1971 and 1979, respectively, from the Faculty of Precision Engineering, Kyoto University. He has been with Kyoto University from 1974. Now, he is a professor at the Department of Systems Science, Graduate School of Informatics. From 1980 to 1981, he had been a visiting researcher at INRIA (National Research Institute for Information Science and Automation), France. His current research interests are on the harmonious symbiosis of artificial systems with natural systems including ecological design, ecological interface design, universal design, etc.

About the Author—HIROSHI KAWAKAMI received his B.E., M.E. and Ph.D. degrees in 1987, 1989, and 1994, respectively, from the Faculty of Precision Engineering, Kyoto University. From 1989, he was an instructor at the Department of Information Technology, Faculty of Engineering, Okayama University. From 1998, he has been an associate Professor, Department of Systems Science, Graduate School of Informatics, Kyoto University. His current research interests are on the ecological and emergent system design, co-operative synthesis method and knowledge engineering.

About the Author—TAKAYUKI SHIOSE received his B.E., M.E. and Ph.D. degrees in 1996, 1998 and 2003, respectively, from the Faculty of Precision Engineering, Kyoto University. He was engaged in Research Fellow, the Japan Society for the Promotion of Science from 1998 to 2000 and Research Associate at Graduate School of Science and Technology, Kobe University from 2000 to 2002. Now, he is a research associate at Graduate School of Informatics, Kyoto University and a visiting researcher at ATR Network Informatics Labs. His current research interests are on the assistant systems for a proficient-skill transfer from the viewpoint of ecological psycholoy.