Building a compact online MRF recognizer for large character set by structured dictionary representation and vector quantization technique

Bilan Zhu*, Masaki Nakagawa

Department of Computer and Information Sciences, Tokyo University of Agriculture and Technology, 2-24-16 Naka-cho Koganei-shi, Tokyo 184-0012, Japan

Abstract

This paper describes a method for building a compact online Markov random field (MRF) recognizer for large handwritten Japanese character set using structured dictionary representation and vector quantization (VQ) technique. The method splits character patterns into radicals, whose models by MRF are shared by different character classes such that a character model is constructed from the constituent radical models. Many distinct radicals are shared by many character classes with the result that the storage space of model dictionary can be saved. Moreover, in order to further compress the parameters, VQ technique to cluster parameter sequences of the mean vectors and covariance matrixes for MRF unary features and binary features as well as the transition probabilities of each state into groups was employed. By sharing a common parameter sequence for each group, the dictionary of the MRF recognizer can be greatly compressed without recognition accuracy loss.

1. Introduction

Online handwritten character recognition has been receiving increased attention due to the development and proliferation of pen-based or touch-based input devices such as tablet computers, smart phones, electronic whiteboards and digital pens (e.g., Anoto pen). Realizing online handwritten character recognition with high performance is vital, especially for applications such as the natural input of text on smart phones, to provide a satisfactory user experience.

Although online handwritten character classifiers with high recognition accuracy have been reported [1–7], the demand for speeding up recognition and small memory consumption is very high for portable devices as well as desk-top applications for which handwriting recognition is incorporated as one of the modules. Performance of these relatively small devices requires a handwriting recognition system as small as possible and recognition speed as fast as possible while maintaining high accuracy. Even for a desktop PC with relatively high device performance, recognition speed within 0.5 s per page and memory consumption within 10 MB are required in actual applications since those applications need more memory consumption on top of the memory consumption for a character recognizer. Therefore, it is required to refine the recognition scheme to reduce the system memory consumption in order to leave more memory space to applications.

Chinese, Japanese or Korean languages have thousands of different character categories and their large character sets create problems not only in recognition rate but also in recognition speed and memory consumption.

Hidden Markov models (HMMs) have been applied widely to online handwriting recognition [8–14]. HMMs probabilistically treat a sequence of feature vectors in writing or position order, so that they can only use the neighborhood relationships between the successively adjacent feature vectors in writing or position order (the so-called one-dimensional neighborhood relationships) but the two-dimensional neighborhood relationships, such as those among spatially neighboring feature vectors are not explicitly expressed. For one-dimensional neighborhood relationships, HMMs only use the state transition probabilities and unary features, but binary features are not used well. Moreover, the neighborhood relationships among more than two neighboring feature vectors, such as ternary features, cannot be used. Although some HMMs apply binary features, they only merge the binary features into the unary features and use a vector of larger dimension, which limits recognition accuracy.

The MRF model is described using an undirected graph in which a set of random variables have a Markov property, and MRFs can be used to effectively integrate information among neighboring feature vectors, such as binary and ternary features, and two-dimensional...
neighborhood relationships [15]. Therefore, MRFs have been effec-
tively applied to stroke-analysis-based structural offline character
recognition [16,17]. They have also been widely and successfully
applied to image processing [18,19]. However, MRFs had not been
applied to online character recognition until our reports [6,7].
Although current online handwritten character recognition tends
to use HMMs, MRFs have more degrees of freedom than HMMs for
explicitly expressing relations among multiple feature vectors (note
that HMMs can be viewed as specific cases of MRFs).

Since online character patterns contain temporal information on
pen movements, structural methods that discard temporal
information and only apply structural information can realize
stroke-order independence. However, it is computationally expen-
sive since the neighborhood relationships must be examined
in two dimensions. Although the method introducing temporal
information is very sensitive to stroke order variations, it is
efficient in recognition speed, and combining it with an un-
structural method can deal with the stroke-order variations
[3,20]. Even for the one-dimensional neighborhood relationships
applying MRFs instead of HMMs to integrate the information of
binary features between the successively adjacent feature vectors
in writing or position order can improve performance. We have
proposed a robust online handwritten Japanese character recogni-
tion method using a MRF model [6,7] that introduces temporal
information into one-dimensional neighborhood relationships and
has shown MRFs outperform HMMs.

Although it realizes high recognition performance, the MRF
recognizer requires about 20 MB memory consumption due to
the larger character set of 7097 character categories within
which 2965 Kanji characters of the JIS level-1 standard and 3390
Kanji characters of the level-2 standard as well as Kana (phonetic
characters), English alphabets and so on are included. After
combining the MRF recognizer with an un-structural recognizer
which employs the modified quadratic discriminant function
(MQDF) and linguistic context processing at the string recognition
step, the recognition system would require larger memory con-
sumption (about 65 MB) because of the large memory size for the
MQDF dictionary and the linguistic context dictionary. Therefore, it
is required to compress the memory size as much as possible
while maintaining high accuracy.

To compress the memory size for a large character set, we
can apply two methods: structured dictionary representations

In structured dictionary representations for Chinese, Japanese
or Korean character recognition, strokes/substrokes structure
representations [11–14] and radical-based structure representa-
tions [21–29] can be used. Nakai et al. [11–14] presented a
structured HMM-based online Japanese recognizer, where sub-
stroke HMMs were used to construct character models by
concatenating them. However, only sharing the common para-
meters of substrokes limits the compression effectiveness.

Chinese ideograms (ideographic character shapes), which are
called Kanji in Japanese, are formed in systematic and hierarchical
structure. Radicals are commonly used as the basic semantic or
phonetic units to construct the ideograms. Each Chinese ideogram
consists of one or several radicals. With a reasonable set of small
number of radicals, we are able to represent all the Chinese
ideograms. Using radical models instead of holistic character
models can largely reduce the number of models. Hereafter, we
use “character” instead of “ideogram” when it is clear to denote
the shape of the character.

In segmentation-free HMM-based English word recognition
[8–10], each letter is modeled as a HMM and word HMMs are
constructed by concatenating letter HMMs during recognition.
This exploits the advantages that one-dimensional structural
methods (one-dimensional neighborhood relationships) such as
HMMs may offer the ability to concatenate character HMMs to
construct word HMMs based on the provided lexicon of words. If
each radical of Japanese characters is considered as an English
letter and each Japanese character is treated like an English word,
then the English word recognition method can be applied to
recognize Japanese characters. However, the radicals in Japanese
characters are different from the English letters largely, where
the radicals with the same type from different Japanese characters
may have different sizes, vertical–horizontal (vh-) ratios and posi-
tions with the result that they have very different covariance
matrices among MRF models. It results in low recognition perfor-
mance if we merge the radical MRF models with different covariance
matrices.

Some structured radical-based and template-based Japanese
online recognizers [21,22] were presented, but they were so
simple without statistical nature so that their recognition rates
were poorer in comparison with the state of the art recognition
methods for online handwriting recognition. In recent statistical
and structural methods such as HMMs and MRFs, however, it is
the important issue to share statistic features of common radicals
among different characters, i.e., to share means and covariance
matrices of features in the same radical, which appears in different
character categories with different size, vh-ratio and position.

Ma et al. has presented a new radical-based approach for online
handwritten Chinese character recognition that uses statistical
classification of radicals, over-segmentation of character patterns
into candidate radicals, and lexicon-driven recognition of charac-
ters [23–26]. Another radical-based method has also been applied
to offline Chinese character recognitions [27,28] where 10 or 12
structure types were used to split Chinese characters into radicals.
A radical-based online Chinese handwriting HMM recognizer has
been also presented [29]. However, they did not present two very
important problems of how to split characters into radicals and
how to construct character models from radicals.

As for VQ technique [30] for compressing a large-set character
recognizer, Long et al. constructed a compact offline MQDF
Chinese recognizer by VQ technique [31] and Du et al. [32] and
Wang et al. [33,34] also constructed compact online multiple
prototype based Chinese recognizers by VQ techniques. They
used the statistical methods (un-structural methods) to represent
the holistic character shape as a feature vector, and then split
successive parameter elements of feature vectors into shorter
subsequences (usually one or two elements), and clustered them
into groups, and then by sharing a common parameter subse-
quence for each group, the dictionary of the recognizer was greatly
compressed. Also in our online MRF recognizer that is a structural
method, there are many similar sequences in unary features and
binary features among characters or radicals. Therefore, we can
apply the same method to the parameters of each mean vector,
those of each covariance matrix for unary features and binary
features, as well as those of the transition probabilities of each
state, and cluster the parameter sequences into groups, with each
group sharing a common parameter sequence. It can result in
more effective compression.

In this paper, a method for building a compact online MRF
recognizer for large Japanese character set is presented, where the
radical models are shared by different characters, and a character
model is constructed from the constituent radical MRF models. It
was investigated how to split characters into radicals and how to
normalize the sizes and positions of common radicals from
different characters. Moreover, VQ technique was employed to
cluster the parameter sequences of the mean vectors and the
covariance matrices as well as the state transition probabilities for
MRF into groups. By sharing a common parameter sequence for
each group, the dictionary of the MRF recognizer can be greatly
compressed without recognition accuracy loss.
This paper is an extension of a preliminary report made in the conference paper [35]. We described background and objective in more details and extended technical description. The main contribution is made by adding extensive experimental results through expansion of the number of character classes from 2965 to 4218, applying several new approaches and methods: two approaches for how to split characters into radicals, two methods for how to normalize the sizes and positions of the common radicals from different character classes, and a method for constructing multiple MRF models for each radical class to improve recognition performance.

The rest of this paper is organized as follows: Section 2 gives an overview of our online handwritten character recognition method by unstructured MRF dictionary. Section 3 presents the construction of our structured MRF dictionary and its usage for recognition. Section 4 describes the dictionary compression by VQ. Section 5 presents the experimental results and Section 6 draws our conclusion.

2. Overview of recognition system by unstructured MRF dictionary

An input pattern is linearly normalized by converting the pen-tip trace pattern to a standard size, preserving the vh-ratio. After normalization, feature points are extracted using the method developed by Ramer [36]. First, the start and end points of every stroke are picked up as feature points. Then, the most distant point from the straight line between adjacent feature points is selected as a feature point if the distance to the straight line is greater than a threshold value. This selection is done recursively until no more feature points are selected. This feature point extracting process is shown in Fig. 1(a).

The extracted feature points represent the structure of a pattern. They are effective and more efficient to process compared with processing all the pen-tip points, as was done in previous studies [8–10].

We set feature points from an input pattern as sites \( S = \{s_1, s_2, s_3, \ldots, s_l\} \) and states of a character class \( C \) as labels \( L = \{l_1, l_2, l_3, \ldots, l_j\} \). The method recognizes the input pattern by assigning labels to the sites to make the matching between the input pattern and each \( C \) such as \( F = \{s_1 = l_1, s_2 = l_1, s_3 = l_3, \ldots, s_9 = l_8, s_{10} = l_9\} \), as shown in Fig. 1(b). The notation \( F \) is a configuration and denotes a mapping from \( S \) to \( L \).

The coordinates of feature points are employed as unary features and the differences in coordinates between the neighboring feature points are employed as binary features. A MRF model is used to match the feature points with the states of each character class and obtain a similarity for each one. The character class with the largest similarity is then selected as the recognition result [6,7].

The neighborhood system of MRF model is set according to the successive adjacent feature points in writing order. A linear-chain MRF for each \( C \) is then defined, as shown in Fig. 1(c), where each label has a state and each state has three transitions.

Therefore, the energy function is as follows:

\[
E(\mathbf{O}, \mathbf{F} | C) = E(\mathbf{O} | F, C) + E(F | C) \\
= \sum_{i=1}^{l} \left[ - \log P(O_i | l_i, C) - \log P(O_{i+1} | l_i, l_{i-1}, C) - \log P(l_i | l_{i-1}, C) \right] 
\]

(1)

where \( l_i \) is the label of a \( C \) assigned to \( s_i \), \( O_i \) is the unary feature vector extracted from site \( s_i \), and \( O_{i+1} \) is the binary feature vector extracted from the combination of \( s_i \) and \( s_{i-1} \).

The smaller the energy function in (1) becomes, the larger the similarity between the input pattern and a \( C \).

Each \( C \) has a linear-chain MRF, and the system uses the Viterbi search to match feature points of the input pattern with states for the MRF model of each \( C \) and to find the matching path with the smallest energy in (1) for each \( C \).

The unary feature vector \( O_i \) comprises \( X \) and \( Y \) coordinates of \( s_i \). The binary feature vector \( O_{i+1} \) has two elements \((dx: \text{X coordinate of } s_i - \text{X coordinate of } s_{i-1}, dy: \text{Y coordinate of } s_i - \text{Y coordinate of } s_{i-1})\).

Gaussian functions are used to estimate \( P(O_i | l_i, C) \) and \( P(O_{i+1} | l_i, l_{i-1}, C) \). \( P(l_i | l_{i-1}, C) \) is estimated as follows:

\[
P(l_i | l_{i-1}, C) = \frac{\text{Number of transitions from } l_{i-1} \text{ to } l_i}{\text{Number of sites assigned } l_{i-1}} \\
P(l_i | l_0, C) = \frac{\text{Number of } s_1 \text{ assigned } l_i}{\text{Number of } s_1}
\]

(2)

To train the MRF of each \( C \), we first initialize the feature points of an arbitrary character pattern among the training patterns of the \( C \) as states of the MRF, set each unary feature vector of each feature point as the mean of the Gaussian function for each single-state, and set each binary feature vector between two adjacent feature points as the mean of the Gaussian function for each pair-state, and initialize the variances of those Gaussian functions and the state transition probabilities as 1. Then we use the Viterbi algorithm or the Baum–Welch algorithm to train up the parameters of the MRF (the means and variances of Gaussian functions and the state transition probabilities). We repeat the training until the optimal parameters are obtained.
3. Construction of structured MRF dictionary and its usage for recognition

3.1. Splitting character MRF models

Initially, a MRF recognizer at the character level (unstructured MRF dictionary) is created, where the character MRF models are trained up from the training patterns. From each character MRF model, it is possible to obtain the mean vectors of unary features.

![Mean coordinates of character models](image)

Fig. 2(a) and (b) show the mean coordinates of two character models. Each small rectangle shows a feature point. All the strokes are concatenated into a stroke for each character to achieve stroke-number independence.

From this character level MRFs, structured MRF models which are composed of radical MRF models are created. In order to make the latter, an interface to construct structured MRF dictionary as shown in Fig. 2 was created. By a mouse right click at a feature point, we can cut the character model at the feature point and, by a mouse left click at a feature point, we can concatenate the two radicals beside the feature point. We can split each character model into several parts by these operations. Then, for each part, we can register it as a new radical to the radical dictionary by cutting it out and normalizing it (normalizing the mean vectors and the covariance matrices) to the radical dictionary size. We can also search similar radicals from the registered radical dictionary and then select a similar radical index to register it. Fig. 2 shows the split character models for characters [xy] and [yz].

![Split character models](image)

When registering radicals for constructing structured MRF dictionary, two dictionaries: a dictionary for radical MRF models, and a dictionary for character radical information are created. The former stores radical MRF models while the latter stores the radical information for each character class such as what radicals it has and the radical indexes pointed to.

After all character MRF models are split and registered, all training patterns are matched with character MRF models, and then each training pattern can be cut into radicals at the places where character MRF models are split. From the bounding boxes of extracted radical training patterns at the same position of the same character class, the mean bounding box is obtained by averaging those bounding boxes as shown in Fig. 2.

![Bounding boxes of extracted radicals](image)

We need to investigate how to split characters into radicals here because it affects the character recognition performance largely. Combination of radicals can be categorized into nine types in [25,26], 10 types in [27] and 12 types in [28]. We consider thirteen types in our system as shown in Fig. 3(a) where three types (ULR, LRD and LUD) are newly proposed. For the surrounding (SUR) type, the surrounding strokes are always divided by the middle radicals as shown in Fig. 3(b) so that it is difficult to split.

![Structure types of Chinese characters](image)

Fig. 3. Structure types of Chinese characters.

Table 1: Examples of thirteen structure types.

<table>
<thead>
<tr>
<th>Type</th>
<th>SE</th>
<th>LR</th>
<th>UD</th>
<th>UL</th>
<th>UR</th>
<th>LD</th>
<th>ULD</th>
<th>LUR</th>
<th>LRD</th>
<th>LUD</th>
<th>LMR</th>
<th>LMD</th>
<th>UMD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td>白本</td>
<td>漢</td>
<td>台北</td>
<td>匯</td>
<td>路</td>
<td>天</td>
<td>市</td>
<td>市</td>
<td>市</td>
<td>市</td>
<td>市</td>
<td>市</td>
<td>市</td>
</tr>
</tbody>
</table>
them into radicals. Therefore, we treat the characters of this type as single-element type. Some examples of thirteen structure types are shown in Table 1.

To split Chinese characters into radicals, we consider two approaches: the coarse level approach (coarse_approach) to split them according to the structure types and the fine level approach (fine_approach) to split divided structures furthermore into smaller radicals. For example, the character “火” can be split into R1 and R2 by the coarse_approach as shown in Fig. 4(a), and it can also be split into R1, R2, R3, and R4 by the fine_approach as shown in Fig. 4(b). The fine_approach splits characters into a more number of smaller radicals and shares the radicals with a more number of character classes resulting in a more compact recognizer. According to this approach, however, the small radicals with the same type from different character classes such as R3 in Fig. 4(b) and R3 in Fig. 4(c) may have different shapes being affected by the neighboring radicals, so that it results in unstable radical structures. The problem will cause low recognition rate. On the other hand, the coarse_approach splits characters according to the defined structure types without applying decomposition recursively, so that the radical shapes are not affected easily by the neighboring radicals and the variations in shared radicals may be smaller resulting in stable radical structures although compaction effect is less.

As shown in Fig. 2, we can see that the same radical such as [火] from different characters may have different sizes, vh-ratios and positions. It is required to normalize them, which can be done by the following two methods:

Method 1: normalize each radical pattern according to its bounding box (independent normalization).
Method 2: normalize each radical pattern according to the mean bounding box of the same radical in training patterns (mean-based normalization).

In the subsequent sections, we present them in relation with their usage in character recognition. Depending on which normalization is employed, not just model description but also the recognition method is affected.

3.2. Independent normalization and its usage for recognition

This section presents the independent normalization in relation to its usage for recognition. We describe them in a single section since they are tightly related.
3.2.1. Independent normalization method

Each radical pattern is normalized by its bounding box of the radical pattern as shown in Fig. 5, where the patterns with thick strokes show the input radical patterns while the patterns with thin strokes show the mean coordinates of the feature points of character models. Each radical pattern is cut out, its bounding box is transformed into a unit rectangle and each feature point is reflected by this transformation and then the transformed radical pattern is used to train the radical MRF models.

Namely, the coordinate \((x, y)\) of each feature point of each radical pattern is transformed to a new coordinate \((x', y')\) as follows:

\[
\begin{align*}
    x' &= (x - l_{rb}) \frac{N}{w_{rb}} \\
    y' &= (y - t_{rb}) \frac{N}{h_{rb}}
\end{align*}
\]

where \(l_{rb}, t_{rb}, w_{rb}\) and \(h_{rb}\) are the leftmost position, the top position, the width and the height of the bounding box of the radical pattern, respectively, and \(N\) is the normalization size.

3.2.2. Employment of independent normalization for recognition

When an input character pattern is recognized, we need to obtain the bounding box of each radical pattern to transform and match it with the radical sequence of each character class. Namely, we need to decide the start and end feature points of a radical pattern to calculate the bounding box. Without obtaining recognition result, however, it is difficult to know the start point and end point of each radical pattern. Therefore, we apply an over-segmentation-based method to recognize. This is feasible for English words but is very difficult for Japanese characters. Ma et al. presented an over-segmentation method for over-segmenting characters into candidate radicals [23-26], but it was very troublesome for over-segmenting the special radicals (UL, UR, LD, ULD, and LUR). In this research, our objective is to investigate the recognition accuracy for the over-segmentation-based method so that we take a simple over-segmentation, where we hypothetically segment radicals at every feature point, which may result in large processing time but keep the recognition accuracy. If the recognition accuracy is higher than the other method, we consider to speed it up.

As shown in Fig. 6, each character pattern is over-segmented into primitives at feature points \(P = \{p_1, p_2, p_3, \ldots, p_g\}\). A segmentation candidate lattice is created from the over-segmented primitives. One or more consecutive primitive segments are combined to form a candidate (probable) radical pattern. All possible sequential combinations of these candidate radical patterns are represented in a lattice where each node in the lattice denotes a radical candidate pattern and each edge denotes a link leading to the next pattern in the sequence. We define an edge such that each candidate radical pattern existing from the segmentation point \(p_k\) to \(p_l\) \((k, l = [1 \sim g])\) is followed by candidate radical patterns starting from the segmentation point \(p_{l+1}\) (instead of segmentation point \(p_l\)).

A radical sequence of a character class \(C\) or an input character pattern is denoted as \(R = \{R_1, R_2, R_3, \ldots, R_n\}\) in writing order as shown in Fig. 6. A radical-synchronous beam search strategy is used to match node sequences in the lattice with the radical sequence of each character class \(C\). All search paths are evaluated according to the path evaluation criterion, and then the best path is selected as the matching result with \(C\) and the matching score of the best path is selected as the similarity with the character class \(C\). Then, the character class with the largest similarity is selected as recognition result. The following path evaluation criterion is used:

\[
E_C = \frac{\sum_{i=1}^{N_C} (\lambda_1 S_{\text{MRF}}/N_f + \lambda_2 S_{\text{size}} + \lambda_3 S_{\text{pos}} + \lambda_4 S_{\text{binary}})}{N_C}
\]

where \(E_C\) stands for the evaluation criterion for an input character pattern and a character class \(C\). \(N_C, S_{\text{MRF}}, S_{\text{size}}, S_{\text{pos}}, S_{\text{binary}}\) are the number of radicals in \(C\), the score of the MRF recognizer for a radical \(R_i\), the number of the feature points of \(R_i\), the score of size evaluation for \(R_i\), the score of position evaluation for \(R_i\), and the score of binary feature evaluation between \(R_i\) and \(R_{i-1}\), respectively, and \(\lambda_h (h = 1 \sim 4)\) are the weighting parameters.

When normalizing radical patterns by the bounding boxes of the radical patterns, the size and position information are lost. Therefore, it is required to evaluate the size and position scores for each radical and the binary feature scores between neighboring radicals.

In order to evaluate the radical size, a feature vector comprising the height and width of the bounding box of each radical pattern is extracted. The feature vector for the radical position comprises the leftmost position and the top position of the bounding box of each radical pattern. The feature vector for the binary feature between neighboring radicals has two elements measured from the bounding boxes of two adjacent radical patterns \(R_i\) and \(R_{i-1}\): the leftmost position of the bounding box of \(R_i\) – the leftmost position of that of \(R_{i-1}\), and the top position of that of \(R_i\) – the top position of that of \(R_{i-1}\).

3.3. Mean-based normalization and its usage for recognition

In this section, we present the mean-based normalization in relation to its usage for recognition. Again, they are tightly related so that we describe them in a single section.

3.3.1. Mean-based normalization method

After training the character MRF models, we can cut out radical models, and normalize each radical model to a standard size (the normalization size) by the corresponding mean bounding box averaged from those of the same radical patterns as shown in Fig. 7. When
cutting out and normalizing each model, its values of mean vectors and covariance matrices are transformed as follows:

\[
\begin{align*}
m'_1 &= (m_1 - l_{mb}) \frac{N}{w_{mb}} \\
m'_2 &= (m_2 - t_{mb}) \frac{N}{h_{mb}} \\
E'_{11} &= E_{11} \frac{N^2}{w_{mb}^2} \\
E'_{22} &= E_{22} \frac{N^2}{h_{mb}^2} \\
E'_{12} &= E_{12} \frac{N^2}{w_{mb}h_{mb}}
\end{align*}
\]

where \( m_1 \) and \( m_2 \) denote two elements of each mean vector, respectively, \( m'_1 \) and \( m'_2 \) denote the transformed values, respectively, \( l_{mb}, t_{mb}, w_{mb} \) and \( h_{mb} \) are the leftmost position, the top position, the width and the height of the corresponding mean bounding box, respectively, \( E_{ij} \) denotes an element of the covariance matrixes, \( E'_{ij} \) is the transformed element and \( N \) is the normalization size. In our MRF model, each unary feature has two elements and each binary feature also has two elements, so that each mean vector has two elements and each covariance matrix has four elements where \( E_{12} = E_{21} \), and only three elements need to be saved.

We can consider that the normalized models of the same radical from different characters are similar after normalization so that we can integrate them by one model through the method described below. The transformation in (5) is a linear transformation so that if we train a radical model from the transformed radical patterns appearing in the same character class by the following linear transformation, we can obtain the same radical model as that cut out and normalized from the character MRF model.

\[
\begin{align*}
x' &= (x - l_{mb}) \frac{N}{w_{mb}} \\
y' &= (y - t_{mb}) \frac{N}{h_{mb}}
\end{align*}
\]  

where the coordinate \((x, y)\) of each feature point of each radical pattern is transformed to a new coordinate \((x', y')\).

We can obtain the integrated radical model by training all transformed patterns of the same radical from different characters. It is also possible to reflect the integrated radical model back to the character models according to the mean bounding box by transforming the values of mean vectors and covariance matrices of each integrated radical model as follows:

\[
\begin{align*}
m''_1 &= m'_1 \frac{w_{mb}}{N} + l_{mb} \\
m''_2 &= m'_2 \frac{h_{mb}}{N} + t_{mb} \\
E''_{11} &= E'_{11} \frac{w_{mb}^2}{N^2} \\
E''_{22} &= E'_{22} \frac{h_{mb}^2}{N^2} \\
E''_{12} &= E'_{12} \frac{w_{mb}h_{mb}}{N^2}
\end{align*}
\]

where \( m'_1 \) and \( m'_2 \) denote two elements of each mean vector, respectively, \( m''_1 \) and \( m''_2 \) denote the transformed elements, respectively, \( E''_{ij} \) denotes an element of the covariance matrixes and \( E'_{ij} \) is the transformed value.
According to this basic idea, we have designed the normalization method by the mean bounding box as follows:

Firstly, the radical patterns are cut out and normalized by the mean bounding box of the radical as shown in Fig. 8, where the patterns with thick strokes show the input radical patterns and the patterns with thin strokes show the character models. When cutting out and normalizing them, the coordinate \((x, y)\) of each feature point of each radical pattern is transformed to a new coordinate \((x', y')\) by the transformation in (6). Finally the normalized radical patterns of the same radical from different characters are used to train up the radical MRF model (the integrated radical model).

3.3.2. Employment of mean-based normalization for recognition

Each trained radical model is considered as integrated from its patterns appearing in many character patterns. Therefore, when recognizing, the radical MRF models can be set into the character models according to their mean bounding boxes by the transformation in (7). Then we use the character models to recognize input patterns. In this method, the size and position information are not lost unlike the independent normalization method. Therefore, it is not required to evaluate the size and position scores for each radical and the binary feature scores between neighboring radicals.

3.3.3. Clustering mean bounding boxes

Each radical MRF may come from various radical patterns in many character classes having very different sizes and positions. It is considered that different sizes will bring different variances, which may result in low recognition performance if too many and various radical patterns are merged into a radical class. Therefore, a method is proposed to solve this problem. The \(k\)-mean method is used to cluster the mean bounding boxes (width and height) of the same radical from different character classes into groups for each radical in the radical dictionary and split each radical into multiple radical classes by registering the centers of groups as new radicals to the radical dictionary. After the \(k\)-means clustering, some clusters are combined again if the distances among their centers are smaller than a threshold. Eventually, more radical classes are obtained. Then, the radical MRF models are trained by training...
patterns. Fig. 9 gives an example for clustering the mean bounding boxes. Suppose three thousand character classes \(\{C_1, C_2, C_3, ..., C_{3000}\}\) which are construed from one hundred radicals \(\{R_1, R_2, R_3, ..., R_{100}\}\). \(C_1, C_2\) and \(C_3\) include \(R_1\). We cluster the mean bounding box of \(R_1\) of \(C_1\), that of \(C_2\) and that of \(C_3\) into two groups and split \(R_1\) into two radical classes \(R'_1, R'_2\), where \(C_1\) and \(C_2\) point to \(R'_1\), while \(C_3\) points to \(R'_2\). After applying this clustering for all radical classes, we obtain a set of new radical classes \(\{R'_1, R'_2, R'_3, ..., R'_{1000}\}\).

### 3.4. Clustering radical models

Even in each radical class multiply split by the bounding box, there may be several different variations of radical patterns such as different stroke-orders. Moreover, input radical patterns may be heavily distorted depending on writers. It may result in low recognition performance. To solve the problem, radical classes are further split and the radical training patterns of each radical are clustered into groups and split them into multiple groups. A MRF model for each group is created.

Again, Fig. 9 shows an example of clustering radical training patterns of each radical. After clustering the mean bounding boxes of the same radicals, each of the radical classes \(\{R'_1, R'_2, R'_3, ..., R'_{1000}\}\) only has a single MRF model. The radical training patterns of each radical are clustered to split a single MRF model into multiple MRF models.

When clustering, it is necessary to calculate the distance between each radical pattern and each group center. To do so, a MRF model from each radical pattern of the center is created by setting the feature points of the radical pattern as states of the MRF, setting each unary feature vector of each feature point as the mean of the Gaussian function for each single-state, and setting each binary feature vector between two adjacent feature points as the mean of the Gaussian function for each pair-state, and setting the variances of those Gaussian functions and the state transition probabilities as 1. Then, the Viterbi algorithm is used to match each radical pattern with the MRF model of each group center and a similarity is obtained and it is used as the distance.

When recognizing, all MRF models are tried for each radical class and the MRF model with the best result is selected. Training patterns are used to optimize the number of the subclasses of each radical class, by changing it at every one step to cluster radical training patterns to obtain the best recognition rate.

### 4. Dictionary compression by VQ

We propose a dictionary compression method inspired by the construction methods for compact recognizers using VQ technique that applied statistical methods to represent the holistic character shape as a feature vector [31–34]. In our online MRF recognizer that is a structural method, there are many similar unary features and binary features among character models or radical models. Since we employ a linear-chain MRF for the MRF model, each label has a state and each state has three transitions. Then, we need to store two parameter elements for each mean vector, and three parameter elements for each covariance matrix of both of unary features and binary features as shown in Fig. 10. Each state has three transition probabilities so that it has three parameter elements. We consider the parameters of each mean vector, those of each covariance matrix and those of three transition probabilities of each state as a sequence. Then we cluster the parameter sequences into groups, with each group sharing a common parameter sequence that is the center of the group. Therefore, it is required only to store the indexes of the groups and the center parameters of the groups, so that it can realize more effective compression.

### 5. Experiments

We made the compact recognizer for 4218 Kanji characters (all the 2965 Kanji characters of the JIS level-1 standard and 1253 Kanji characters for naming in the JIS level-2 standard) evaluated. This extended set is commonly used for daily life. The character recognizer was trained up by using an online Japanese handwriting database called Nakayosi [37]. On the other hand, the performance test was made on an online Japanese handwriting database called Kuchibue [37]. Table 2 shows the details of the

<table>
<thead>
<tr>
<th></th>
<th>Nakayosi_1</th>
<th>Kuchibue_d</th>
</tr>
</thead>
<tbody>
<tr>
<td>#writers</td>
<td>163</td>
<td>120</td>
</tr>
<tr>
<td>#characters/each writer</td>
<td>11,962</td>
<td>10,403</td>
</tr>
<tr>
<td>Kanji/Kana/Symbol/alpha numerals</td>
<td>5643/5068/1085/166</td>
<td>5799/3723/816/65</td>
</tr>
<tr>
<td>#character categories/each writer</td>
<td>4438</td>
<td>3356</td>
</tr>
<tr>
<td>Kanji/Kana/Symbol/alpha numerals</td>
<td>4058/169/149/62</td>
<td>2976/169/146/62</td>
</tr>
<tr>
<td>#average category characters</td>
<td>2.3</td>
<td>3.6</td>
</tr>
<tr>
<td>Kanji/Kana/Symbol/alpha numerals</td>
<td>1.4/22.0/5.5/1.0</td>
<td>1.9/30.0/7.4/2.7</td>
</tr>
</tbody>
</table>
databases. Each character class (character category) has a different number of sample patterns, and Kana and symbol have more patterns (see Table 2). To maintain balance, 100 patterns were selected randomly from each character class of the Kuchibue database and the same number of sample patterns was used for each character class to evaluate the performance. The experiments were implemented on an Intel(R) Xeon(R) CPU W5590 at 3.36 GHz and 3.36 GHz (2 processors) with 12 GB memory.

We first compared the performance of three models: the original MRF without structured dictionary representation, the structured MRF (Str-L_N) which uses the independent normalization according to its bounding box described in Section 3.2, and the structured MRF (Str-M_N) which uses the mean-based normalization according to the mean bounding box described in Section 3.3. We also compared the two approaches to split characters into radicals (coarse_approach and fine_approach) for Str-L_N and Str-M_N.

For Str-L_N, we compared the two methods: the method which evaluates the scores of radical geometric features (size, position between neighboring radicals) for the path evaluation criterion in (4) (With-geo), that which does not evaluate the scores of radical geometric features (Without-geo). For each character class, we trained quadratic discriminant function (QDF) classifiers for unary features (sizes and positions) of radicals and binary features between two neighboring radicals from training patterns to evaluate the scores of radical geometric features. The geometric feature vectors are transformed to log-likelihood scores using QDF classifiers. After all the training patterns are matched with character MRF models, and are cut at the places where character MRF models are split, we can train the three QDF models (size, position, binary feature) for each character class. The weighting parameters \( \lambda_h \) \((h = 1, 2, \ldots, 4)\) in (4) are trained using the training patterns. We change each weighting parameter at every 0.01 step to obtain the best recognition rate.

For Str-M_N, we clustered the mean bounding boxes (width and height) of the same radical from different character classes into groups, and tried the number of clusters from 1 to 9 at every 1 step \((c_1, c_2, \ldots, c_9)\) and from 10 to 70 at every 10 step \((c_{10}, \ldots, c_{70})\) for all radicals. Table 3 shows the results of the original MRF model and Table 4 shows the results of the structured models where \( C_r \) is the character recognition rate, speed is the average time to recognize a character, \( c_1, c_2, \ldots, c_{10}, \ldots, c_{70} \) denote the clustering number for clustering the mean bounding boxes of the radicals from 1 to 9 and from 10 to 70.

From these results, we can see that the coarse_approach outperforms the fine_approach, because the coarse_approach brings stable radical structures although it has larger dictionary sizes.

Str-L_N largely degrades character recognition accuracy, and evaluating the scores of radical geometric features can improve the accuracy. Str-L_N normalizes radical patterns by the bounding boxes of the radical patterns, so that it loses the size and position information, and the lost information cannot be saved even after evaluating the scores of radical geometric features. Moreover, Str-L_N needs to hypothetically segment radicals at every feature point and results in very larger processing time.

Str-M_N by the coarse_approach yields better recognition accuracy than Str-L_N by the coarse_approach, and clustering the mean bounding boxes improves recognition accuracy to a large extent, and it also compresses the memory space from the original MRF. As for the speed of Str-M_N, no loss was observed.

The methods for Str-L_N and fine_approach were omitted for the next evaluations because of their very low recognition accuracies.

### Table 3

Performance of the original MRF model.

<table>
<thead>
<tr>
<th>Method</th>
<th>Cr (%)</th>
<th>Memory (MB)</th>
<th>Speed (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original MRF</td>
<td>96.25</td>
<td>12.00</td>
<td>254.0</td>
</tr>
</tbody>
</table>

### Table 4

Performance of structured models.

<table>
<thead>
<tr>
<th>Method</th>
<th>Fine_approach</th>
<th>Coarse_approach</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cr (%)</td>
<td>Memory (MB)</td>
</tr>
<tr>
<td>Str-L_N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>With-geo</td>
<td>93.92</td>
<td>0.85</td>
</tr>
<tr>
<td>Without-geo</td>
<td>89.81</td>
<td>0.85</td>
</tr>
<tr>
<td>Str-M_N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c1</td>
<td>91.07</td>
<td>0.85</td>
</tr>
<tr>
<td>c2</td>
<td>91.57</td>
<td>1.40</td>
</tr>
<tr>
<td>c3</td>
<td>91.74</td>
<td>1.75</td>
</tr>
<tr>
<td>c4</td>
<td>91.88</td>
<td>2.06</td>
</tr>
<tr>
<td>c5</td>
<td>92.16</td>
<td>2.27</td>
</tr>
<tr>
<td>c6</td>
<td>92.09</td>
<td>2.48</td>
</tr>
<tr>
<td>c7</td>
<td>92.16</td>
<td>2.67</td>
</tr>
<tr>
<td>c8</td>
<td>92.26</td>
<td>2.76</td>
</tr>
<tr>
<td>c9</td>
<td>92.40</td>
<td>2.88</td>
</tr>
<tr>
<td>c10</td>
<td>92.15</td>
<td>3.01</td>
</tr>
<tr>
<td>c20</td>
<td>92.51</td>
<td>3.59</td>
</tr>
<tr>
<td>c30</td>
<td>92.57</td>
<td>3.85</td>
</tr>
<tr>
<td>c40</td>
<td>92.78</td>
<td>4.02</td>
</tr>
<tr>
<td>c50</td>
<td>92.40</td>
<td>4.02</td>
</tr>
<tr>
<td>c60</td>
<td>92.46</td>
<td>4.20</td>
</tr>
<tr>
<td>c70</td>
<td>92.50</td>
<td>4.22</td>
</tr>
</tbody>
</table>
and the method for clustering radical models described in Section 3.4 was evaluated. By this method, several MRF models for each radical class (Multiple-MRFs) were constructed, while a single MRF model for each radical class (Single-MRF) was also constructed for comparison. Table 5 shows the results.

From these results, we can see that the method of Multiple-MRFs achieves better character recognition accuracy, although it consumed larger memory space and longer recognition time compared to Single-MRF. Compared with the original MRF, the character recognition rate of Multiple-MRFs is slightly higher, its memory size is reduced but its speed is three times slower, which has little problem for usual applications.

Then, the performance of the dictionary compression method by VQ was compared. The parameter sequences can be clustered into 255 groups, and each group can be saved in 1 byte data (VQ1). The parameter sequences can also be clustered into the optimal number of groups, and each group can be saved in 2 byte data (VQ2). Moreover, after constructing the structured MRF, the VQ method was applied to the structured dictionary to further compress the dictionary. For the original MRF, 4 bytes were used to store each parameter of the state transition probabilities and the covariance matrices, and 2 bytes were used to store each parameter of the mean vectors. Table 6 shows the results.

From these results, it can be seen that VQ2 achieves better character recognition accuracy, although it consumed slightly more memory space compared to VQ1. Combining the Str-M_N and Single-MRF with VQ2 yields better recognition accuracy and smaller memory space than VQ1. Combining the Str-M_N and Multiple-MRFs with VQ2 achieves the best recognition accuracy and smaller memory space although it consumed more recognition time compared to VQ2. As for the speed, little loss was observed from VQ.

6. Conclusion

This paper presented a method for building a compact online MRF recognizer for large handwritten Japanese character set using structured dictionary representation and VQ technique. The radical models were shared by different characters, and a character model was constructed from the constituent radical MRF models. Moreover, VQ technique was employed to further compress the memory space without losing recognition accuracy. We applied our method to 4218 Kanji characters (a commonly used character set for daily life), investigated and compared two approaches for how to split characters into radicals, and two methods for how to normalize the sizes and positions of the same radical from different character classes. We also presented a method that constructs multiple MRF models for each radical class to improve recognition accuracy and showed its effect. As the result, we have produced several recognizers each of which has its advantage. By structuring MRFs, introducing the effective normalization and combining with the VQ technique, we have produced a recognizer which keeps recognition accuracy while reducing the whole memory size to 16% but entailing tripled processing time. When the requirement for processing time is tight, we can choose other combinations such as VQ2 for the original MRF, but the tripled processing time has little problem for usual applications on a usual CPU, so that structuring MRFs and combining with VQ provide large benefit.

Conflict of interest statement

None declared.

References


Table 6

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>O (%)</td>
</tr>
<tr>
<td>Original MRF</td>
<td>96.25</td>
</tr>
<tr>
<td>VQ1 for original MRF</td>
<td>95.30</td>
</tr>
<tr>
<td>VQ2 for original MRF</td>
<td>95.60</td>
</tr>
<tr>
<td>Str-M_N + c8 + Single-MRF + VQ2</td>
<td>95.80</td>
</tr>
<tr>
<td>Str-M_N + c8 + Multiple-MRFs + VQ2</td>
<td>96.50</td>
</tr>
</tbody>
</table>
Bilan Zhu was born on 13 December 1974 in China. She received B.Sc. and M.Sc. degrees from Tokyo University of Agriculture and Technology (TUAT) in 2003 and 2004, respectively. In 2007, she received Ph.D. degree in Engineering from TUAT. From April 2007, she has been working at Tokyo University of Agriculture and Technology. Currently, she is an Assistant Professor of Media Interaction in Department of Computer and Information Sciences. She is working on on-line, off-line handwriting character recognition, documents processing and context analysis of Japanese, Chinese and English text.

Masaki Nakagawa was born on 31 October 1954 in Japan. He received B.Sc. and M.Sc. degrees from the University of Tokyo in 1977 and 1979, respectively. During the academic year 1977/78, he followed Computer Science course at Essex University in England, and received M.Sc. with distinction in Computer Studies in July 1979. He received Ph.D. in Information Science from the University of Tokyo in December 1988. From April 1979, he has been working at Tokyo University of Agriculture and Technology. Currently, he is a Professor of Media Interaction in Department of Computer and Information Sciences.