On-line Handwritten Japanese Characters Recognition Using A MRF Model with Parameter Optimization by CRF

Bilan Zhu and Masaki Nakagawa
Department of Computer and Information Science,
Tokyo University of Agriculture and Technology,
Tokyo 184-8588, Japan
{zhubilan, nakagawa}@cc.tuat.ac.jp

***Abstract*— This paper describes a Markov random field (MRF) model with weighting parameters optimized by conditional random field (CRF) for on-line recognition of handwritten Japanese characters. It also presents updated evaluation using a large testing set. The model extracts feature points along the pen-tip trace from pen-down to pen-up and sets each feature point from an input pattern as a site and each state from a character class as a label. It employs the coordinates of feature points as unary features and the differences in coordinates between the neighboring feature points as binary features. The weighting parameters are estimated by CRF or the minimum classification error (MCE) method. In experiments using the TUAT Kuchibue database, the method achieved a character recognition rate of 92.77%, which is higher than the previous model's rate, and the method of estimating the weighting parameters using CRF was more accurate than using MCE.**

Keywords-On-line recognition; Markov random field; character recognition

I. INTRODUCTION

Efforts to improve on-line handwritten character recognition are continuing to yield higher recognition rates and remove constraints on writing text.

Hidden Markov model (HMM) matches pen-points of an input pattern with states for character classes probabilistically [1, 2]. However, the information between the neighboring pen-points such as binary or triple features have not been used well; only unary features have been employed with the consequence being limited recognition accuracy.

MRFs can effectively integrate the information between neighboring pen-points such as binary features and triple features [3] and they have been successfully applied to off-line handwritten character recognition [4] and on-line stroke classification [5]. However, MRFs have not been applied to on-line handwritten character recognition; current on-line handwritten character recognition tend to use HMM-based models (note that HMMs can be viewed as a specific case of MRFs).

Cho et al [6] propose a Bayesian network (BN) based framework for on-line handwriting recognition. BNs share similarities with MRFs. BNs are directional acyclic graphs and model the relationships between the neighboring pen-points as conditional probability distributions, while MRFs are undirected graphs and model the relationships between the neighboring pen-points as probability distributions of binary or triple features.

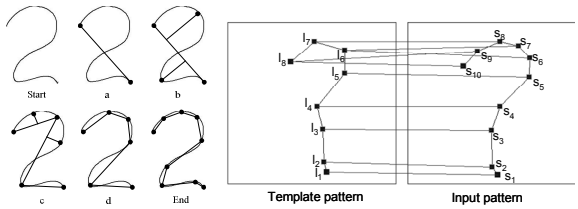
Introducing weighting parameters to MRFs and optimizing them based on CRFs [7] or MCE [8] may bring even higher recognition accuracy; CRF has been successfully applied to on-line string and off-line word recognition [9, 10].

In this paper, we present an MRF model with weighting parameters optimized by CRFs for on-line recognition of handwritten Japanese characters. The model effectively integrates unary and binary features and introduces adjustable weighting parameters to the MRFs, which are optimized according to CRF. The proposed method extracts feature points along the pen-tip trace from pen-down to pen-up and matches those feature points with states for character classes probabilistically based on this model. Experimental results on the TUAT Kuchibue database [11] demonstrate the superiority of our method.

The rest of this paper is organized as follows: Section 2 gives an overview of our on-line handwritten character recognition system. Section 3 constructs a character recognition MRF model, and Section 4 introduces weighting parameters and methods to optimize them. Section 5 presents the experimental results, and Section 6 is our conclusion.

II. RECOGNITION SYSTEM OVERVIEW

We normalize an input pattern linearly by converting the pen-tip trace pattern to a standard size, preserving the horizontal-vertical ratio.



(a) (b)

Fig. 1. Feature points extraction and Labeling.

After the normalization, we extract feature points using the method by Ramner [12]. 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. The feature point extracting process is shown in Fig. 1(a).

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.127

The extracted feature points stand for the structure of a pattern. They are effective and more efficient to process in comparison with processing all the pen-tip points, as is done in [1, 2].

Then we use a MRF model to match the feature points with the states of each character class and obtain a similarity for each character class. We then select the character class with the largest similarity as the recognition result.

III. MRF FOR CHARACTER RECOGNITION

A. Maximum a Posteriori Probability

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

The feature vectors of the feature points from the input pattern constitute the observation set $\boldsymbol{O}$. In statistical or Bayesian paradigms, the decision-making by the character recognizer is based on the concept of the maximum a posteriori (MAP) probability:

$$P(C \mid \mathbf{O}) = \frac{P(C)P(\mathbf{O} \mid C)}{P(\mathbf{O})} \tag{1}$$

where $P(C)$ is the a priori probability that the given pattern belongs to a character class C, $P(\boldsymbol{O}|C)$ is the likelihood function of the observation set $\boldsymbol{O}$ for a class C. $P(\boldsymbol{O})$ is the probability of the observation set $\boldsymbol{O}$, and $P(C|\boldsymbol{O})$ is the probability that the input pattern belongs to a class C for the observation set $\boldsymbol{O}$.

The decision is as follows:

$$\begin{aligned} C^* &= \arg\max_C P(C \mid \mathbf{O}) \\ &= \arg\max_C [P(C)P(\mathbf{O} \mid C)] \end{aligned} \tag{2}$$

where C^* is the estimated character class. If $P(C)$ is set to be constant, and the MAP estimation becomes the maximum likelihood (ML) estimation. The problem in (2) is how to estimate $P(\boldsymbol{O}|C)$. We can express $P(\boldsymbol{O}|C)$ as

$$P(\mathbf{O} \mid C) = \sum_{all\ \mathbf{F}} P(\mathbf{O}, \mathbf{F} \mid C) \tag{3}$$

Here, $\boldsymbol{F} = \{s_1 = l_i, s_2 = l_j, \cdots, s_I = l_k \mid l_i, l_j, l_k \in L\}$ is the matching from the sites S of the input pattern to the labels $\boldsymbol{L}$ of a character class C.

$$P(\mathbf{O}, \mathbf{F} \mid C) = P(\mathbf{F} \mid C)P(\mathbf{O} \mid \mathbf{F}, C) \tag{4}$$

The amount of direct computation required by Eq. (3) is intractable. For HMM, there are two solutions: the forward-backward (Baum-Welch) algorithm and the Viterbi algorithm. To perform this computation task, we consider only the best matching, as in the case of the Viterbi algorithm. That is,

$$P(\mathbf{O} \mid C) \approx P(\mathbf{O}, \mathbf{F}_{best} \mid C) \tag{5}$$

where

$$P(\mathbf{O}, \mathbf{F}_{best} \mid C) = \arg\max_{\mathbf{F}} P(\mathbf{O}, \mathbf{F} \mid C) \tag{6}$$

Therefore, the problem of recognition is to obtain $P(\boldsymbol{F}_{\text{best}}|C)P(\boldsymbol{O}|\boldsymbol{F}_{\text{best}},C)$ and the best match.

B. Markov Random Field Models

Calculating the probability $P(\boldsymbol{F}|C)$ is intractable because the interactions between the variables are global. To make it tractable, MRFs constrain the interdependence of labels by assuming that they only depend on the labels of the neighboring sites. This is described as Markovianity and can be depicted by the neighborhood system [3]. The neighborhood system N_i denotes the neighbors of a site s_i that satisfies $s_j \in N_i \Leftrightarrow s_i \in N_j$, $s_i \notin N_i$. A label interacts with only the neighboring labels. A clique c is defined as a subset of sites that are all mutual neighbors according to the neighborhood system.

The Hammersley-Clifford theorem establishes the equivalence between the Markov random field and the Gibbs random field [3],

$$P(\mathbf{F} \mid C) = \frac{1}{Z}\exp(-E(\mathbf{F} \mid C)) \tag{7}$$

where

$$E(\mathbf{F} \mid C) = \sum_c V_c^F(\mathbf{F} \mid C) \tag{8}$$

is called the prior energy function and $V^F{}_c(\boldsymbol{F}|C)$ is called the prior clique potential function defined on the corresponding clique c, and

$$Z = \sum_{\mathbf{F}} \exp(-E(\mathbf{F} \mid C)) \tag{9}$$

is the normalization factor called the partition function.

Taking $P(\boldsymbol{O}|\boldsymbol{F},C)$ into consideration, we obtain

$$\begin{aligned} &P(\mathbf{F} \mid C)P(\mathbf{O} \mid \mathbf{F}, C) \\ &= \frac{1}{Z}\exp(-(E(\mathbf{O} \mid \mathbf{F}, C) + E(\mathbf{F} \mid C))) \end{aligned} \tag{10}$$

which is called global likelihood energy function, and

$$E(\mathbf{O} \mid \mathbf{F}, C) = \sum_c V_c^O(\mathbf{O} \mid \mathbf{F}, C) \tag{11}$$

where $V^O{}_c(\boldsymbol{O}|\boldsymbol{F},C)$ is called the likelihood clique potential function.

For simplicity, we consider only single-site cliques c1=$\{s_i\}$ and pair-site cliques c2=$\{s_i, s_j\}$. From Eq. (8) and Eq. (11), we get

$$\begin{aligned} &E(\mathbf{F} \mid C) + E(\mathbf{O} \mid \mathbf{F}, C) \\ &= \sum_c \left[V_c^O(\mathbf{O} \mid \mathbf{F}, C) + V_c^F(\mathbf{F} \mid C)\right] \\ &= \sum_{s_i \in c1} \left[V_1^O(O_{s_i} \mid l_{s_i}, C) + V_1^F(l_{s_i} \mid C)\right] \\ &+ \sum_{\{s_i, s_j\} \in c2} \left[V_2^O(O_{s_i s_j} \mid l_{s_i}, l_{s_j}, C) + V_2^F(l_{s_i}, l_{s_j} \mid C)\right] \end{aligned} \tag{12}$$

where l_{s_i} is the label of a class C assigned to s_i, O_{s_i} is the unary feature vector extracted from site s_i, and $O_{s_i s_j}$ is the binary feature vector extracted from the combination of s_i and s_j.

The likelihood clique potentials describe the statistical information about the observations given the labels and the prior clique potentials encode the prior information about the neighboring labels.

In the MAP framework, maximizing the a posteriori probability in Eq. (2) is equivalent to minimizing the energy function in Eq. (12).

C. Decoding Strategy

We defined the cliques as follows:

Single-site: c1 = $\{s_1, s_2, s_3, \cdots, s_{10}, \cdots\}$

Pair-site: c2= $\{\{s_1, s_2\}, \{s_2,s_3\}, \{s_3,s_4\}, \{s_4,s_5\}, \cdots\cdots, \{s_9,s_{10}\}, \cdots\}$

The neighborhood system is according to the successive adjacent feature points in writing order. We define a linear-chain MRF for each character class, as shown in Fig. 2, where each label has a state and each state has three transitions.

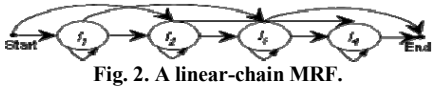


Fig. 2. A linear-chain MRF.

The energy function is as follows:

$$E(\mathbf{O}\mid\mathbf{F},C)=\sum_{i=1}^{I}\left[V_1^O(O_{s_i}\mid l_{s_i},C)+V_2^O(O_{s_is_{i-1}}\mid l_{s_i},l_{s_{i-1}},C)\right] \tag{13}$$

$$E(\mathbf{F}\mid C)=\sum_{i=1}^{I}\left[V_1^F(l_{s_i}\mid C)+V_2^F(l_{s_i},l_{s_{i-1}}\mid C)\right]$$

where I is the number of feature points.

We derive the likelihood clique potentials from the negative logarithm of the conditional probabilities.

$$V_1^O(O_{s_i}\mid l_{s_i},C)=-\log P(O_{s_i}\mid l_{s_i},C) \tag{14}$$

$$V_2^O(O_{s_is_{i-1}}\mid l_{s_i},l_{s_{i-1}},C)=-\log P(O_{s_is_{i-1}}\mid l_{s_i},l_{s_{i-1}},C)$$

where $P(O_{s_1s_0}\mid l_{s_1},l_{s_0},C)$ is set as 1.

We use a linear-chain MRF in Fig. 2 so that the state transition probability can be used to derive the prior energy function instead of the prior clique potential:

$$E(\mathbf{F}\mid C)=\sum_{i=1}^{I}-\log P(l_{s_i}\mid l_{s_{i-1}},C) \tag{15}$$

where $P(l_{s_i}\mid l_{s_{i-1}},C)$ is the state transition probability.

Therefore, the energy function is as follows:

$$E(\mathbf{O},\mathbf{F}\mid C)=E(\mathbf{O}\mid\mathbf{F},C)+E(\mathbf{F}\mid C) \tag{16}$$

$$=\sum_{i=1}^{I}\left[-\log P(O_{s_i}\mid l_{s_i},C)-\log P(O_{s_is_{i-1}}\mid l_{s_i},l_{s_{i-1}},C)-\log P(l_{s_i}\mid l_{s_{i-1}},C)\right]$$

The smaller the energy function in Eq. (16) becomes, the larger will be the similarity between the input pattern and a character class C.

Each character class 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 of each character class and to find the matching path with the smallest energy in Eq. (16) for each character class.

The unary feature vector O_{s_i} comprises X and Y coordinates of s_i. The binary feature vector $O_{s_is_{i-1}}$ has two elements (dx: X coordinate of s_i - X coordinate of s_{i-1}, dy: Y coordinate of s_i - Y coordinate of s_{i-1}), or an element ($\tan^{-1}(dy/dx)$).

Gaussian functions are used to estimate $P(O_{s_i}\mid l_{s_i},C)$ and $P(O_{s_is_{i-1}}\mid l_{s_i},l_{s_{i-1}},C)\cdot\ P(l_{s_i}\mid l_{s_{i-1}},C)$ is estimated as follows:

$$P(l_{s_i}\mid l_{s_{i-1}},C)=\frac{\text{Number of transitions from } l_{s_{i-1}} \text{ to } l_{s_i}}{\text{Number of sites assigned } l_{s_{i-1}}} \tag{17}$$

$$P(l_{s_1}\mid l_{s_0},C)=\frac{\text{Number of } s_1 \text{ assigned } l_{s_1}}{\text{Number of } s_1}$$

To train the MRF of each character class, we first initialize the feature points of an arbitrary character pattern among the training patterns of the character class 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 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.

IV. Optimization of Weighting Parameter

For Eq. (16), we can introduce weighting parameters ($\boldsymbol{\lambda}$=λ_1, λ_2, λ_3) to adjust the values of the unary features, binary features, and state transition probabilities as follows:

$$E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}\mid C)=\sum_{i=1}^{I}\begin{bmatrix}-\lambda_1\log P(O_{s_i}\mid l_{s_i},C)\\-\lambda_2\log P(O_{s_is_{i-1}}\mid l_{s_i},l_{s_{i-1}},C)\\-\lambda_3\log P(l_{s_i}\mid l_{s_{i-1}},C)\end{bmatrix} \tag{18}$$

The weighting parameters can be optimized based on CRF or MCE. Different weighting parameters can be applied to different character classes. We can also adjust more parameters such as the means and the variances of Gaussian functions and the state transition probabilities of the MRFs. In doing so, however, more training patterns must be prepared. The training patterns that we have are not enough to adjust more parameters to obtain a higher recognition rate. Therefore, we only introduce the three common weighting parameters for all the character classes to adjust the values of the unary features, binary features, and state transition probabilities.

According to the CRF model, the posterior probability of a character class C is given by:

$$P(C\mid\mathbf{O})=\frac{\sum_{\mathbf{F}_c}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_c,C))}{\sum_{C_i}\sum_{\mathbf{F}_{C_i}}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_{C_i},C_i))} \tag{19}$$

$$=\frac{\sum_{\mathbf{F}_c}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_c\mid C)-E(C))}{\sum_{C_i}\sum_{\mathbf{F}_{C_i}}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_{C_i}\mid C_i)-E(C_i))}$$

$$=\frac{\sum_{\mathbf{F}_c}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_c\mid C))\exp(-E(C))}{\sum_{C_i}\sum_{\mathbf{F}_{C_i}}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_{C_i}\mid C_i))\exp(-E(C_i))}$$

where $\boldsymbol{F}_C$ is a matching of a character class C. We set $P(C)$ to be constant so that $E(C)$ = -log P(C) is also a constant and the posterior probability is:

$$P(C\mid\mathbf{O})=\frac{\sum_{\mathbf{F}_c}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_c\mid C))}{\sum_{C_i}\sum_{\mathbf{F}_{C_i}}\exp(-E(\boldsymbol{\lambda},\mathbf{O},\mathbf{F}_{C_i}\mid C_i))} \tag{20}$$

We can optimize the parameter vector $\boldsymbol{\lambda}$ by minimizing the following negative log-likelihood (NLL) loss function [15] using stochastic gradient descent [16].

$$L_{NLL}(\boldsymbol{\lambda}, \mathbf{O}) = -\log P(C \mid \mathbf{O}) \tag{21}$$

where C is the correct character class of $\boldsymbol{O}$.

We can also apply the MCE criterion [8] optimized by stochastic gradient descent [16] to find the optimal parameter vector $\boldsymbol{\lambda}$ by minimizing the following difference between the scores of the most confusing character class and that of the correct one:

$$\begin{aligned} &L_{MCE}(\boldsymbol{\lambda}, \mathbf{O}) = \sigma(\max(Score_{incorrect}) - Score_{correct}) \\ &\sigma(x) = (1 + e^{-x})^{-1} \\ &Score_{correct} = \text{score of the correct character class} \\ &Score_{incorrect} = \text{scores of incorrect character class} \end{aligned} \tag{22}$$

where the score for the input pattern and the character class C_i is as follows:

$$Score_{C_i} = -\log\left(\sum_{\mathbf{F}_{C_i}} \exp\left(-E(\boldsymbol{\lambda}, \mathbf{O}, \mathbf{F}_{C_i} \mid C_i)\right)\right) \tag{23}$$

Each character class has an MRF with weighting parameters and the system uses the Viterbi search to match the feature points of the input pattern with the states of the MRF for each character class and to find the matching path with the smallest $E(\boldsymbol{\lambda}, \boldsymbol{O}, \boldsymbol{F}|C)$ in Eq. (18) for each character class.

V. EXPERIMENTS

To evaluate the character recognition model, we trained the character recognizer of the MRFs and the weighting parameters by using an on-line Japanese handwriting database called Nakayosi [11]. The performance test used an on-line Japanese handwriting database called Kuchibue [11]. Table 1 shows the details of the databases. Each character class (character category) has a different number of sample patterns, and kana and symbol have more patterns (see Table 1). To maintain balance, we selected 100 patterns at random from each character class of the Kuchibue database and used the same number of sample patterns for each character class to evaluate the performance. The experiments were implemented on an Intel(R) Core(TM)2 Duo CPU 2.66 GHz with 1.99 GB memory.

Table 1. Statistics of character pattern databases.

		Nakayosi_t	Kuchibue_d
#writers		163	120
#characters /each writer	Total	11,962	10,403
	Kanji/Kana/ Symbol/alpha numerals	5,643/5,068/ 1,085/166	5,799/3,723/ 816/65
#character categories /each writer	Total	3,356	4,438
	Kanji/Kana/ Symbol/alpha numerals	2976/169/ 146/62	4058/169 149/62
#average category characters	Total	3.6	2.3
	Kanji/Kana/ Symbol/alpha numerals	1.9/30.0/ 7.4/2.7	1.4/22.0 5.5/1.0

VI. COMPARISON OF MRFS AND HMMS

First, we compared MRFs and HMMs. To ensure a fair comparison, the MRFs and HMMs used the same databases, the same training method, and the same features. For the HMMs, we merged the binary features into the unary features and used a vector of larger dimension for each single-site, because the HMMs do not consider the binary features for each pair-site and only use the unary features for each single-site. We defined an HMM for each character class, in a manner similar to the linear-chain MRF shown in Fig. 2, where each label had a state and each state had three transitions.

We extracted the following features from each single-site s_i and each pair-site $\{s_i, s_j\}$:

- x: X coordinates of s_i
- y: Y coordinates of s_i
- dx: X coordinate of s_i - Y coordinate of s_{i-1}
- dy: Y coordinate of s_i - Y coordinate of s_{i-1}
- dir: $\tan^{-1}(dy/dx)$

The HMMs evaluated the similarity between the input pattern and a character class C by using Eq. (24) below, whereas the MRFs evaluated it by using Eq. (16).

$$\begin{aligned} E(\mathbf{O}, \mathbf{F} \mid C) &= E(\mathbf{O} \mid \mathbf{F}, C) + E(\mathbf{F} \mid C) \\ &= \sum_{i=1}^{I} \left[-\log P(O_{s_i} \mid l_{s_i}, C) - \log P(l_{s_i} \mid l_{s_{i-1}}, C)\right] \end{aligned} \tag{24}$$

where O_{s_i} is the unary feature vector extracted from a site s_i and has four elements (x, y, dx, dy), three elements (x, y, dir), or only two elements (x, y). Since HMMs always tend to use the direction features dir we also tested their performance. For the MRFs, we tried two types of features. The first type was (x, y) for the unary features and (dx, dy) for the binary features. The second type was (x, y) for the unary features and (dir) for the binary features.

We test the performance of recognizing kanji of Chinese origin with 1,000 categories, hiragana (a subset of kana) with 46 categories and lowercase alphabet with 26 categories. We used the Viterbi algorithm and the Baum-Welch algorithm to train the models. Table 2 shows the results.

Table 2. Results of MRFs and HMMs (%).

Performance	Method	MRFs		HMMs		
		x,y,dx,dy	x,y,dir	x,y,dx,dy	x,y,dir	x,y
kanji	Viterbi	97.44	95.36	96.69	93.25	94.58
	Baum-Welch	97.37	95.32	96.69	93.20	94.64
hira-gana	Viterbi	95.36	91.30	93.86	90.45	90.08
	Baum-Welch	95.30	91.39	93.80	90.45	90.80
alphabet	Viterbi	92.61	88.65	89.84	89.23	87.23
	Baum-Welch	93.15	89.23	90.23	89.65	87.26

MRFs and HMMs took about the same recognition times and training times. The average character recognition time was 0.0029 ms when using features (x, y, dx, dy), 0.0027 ms when using (x, y, dir), and 0.0022 ms when using (x, y). The average training time of an iteration for the Viterbi algorithm is about 16 s whereas it is about 51 s for the Baum-Welch algorithm. These results lead us to the following observations:

(1) MRFs had higer recognition accuracy than HMMs except in the case of alphabet recognition with features (x, y, dir). Therefore, we can conclude that the MRFs are more effective than HMMs as a result of their integrating information between neighboring pen-points such as binary features.

(2) More features resulted in higher recognition accuracy except in the case of kanji recognition with HMMs and features (x, y, dir) and the case of hiragana recognition with HMMs and features (x, y, dir) trained by the Baum-Welch algorithm.

(3) The accuracies of the Viterbi algorithm and the Baum-Welch algorithm were comparable.

A. Comparison of Models with Parameter Optimization

Next, we compared the performance of four recognition models: MRFs with weighting parameters optimized by CRF or by MCE, MRFs without weighting parameters, and the model presented in [15] that uses a Structured Character Pattern Representation (SCPR) dictionary and Linear-time Elastic Matching (LTM). We test the performance for all character categories of the Kuchibue database. We used the Viterbi algorithm to train the MRF models and used unary features (*x, y*) and binary features (*dx, dy*) for the MRFs. LTM extracted the same feature points from on-line patterns as the MRFs and learned several prototypes using a learning vector quantization (LVQ) method for each character class. It then matches those feature points from the input pattern with those of each prototype of each character class. LTM does not consider the distributions for each feature points and only uses the unary features (*x, y, dir*) to calculate the distances between matched pairs of feature points of the input pattern and each prototype, and then sum those distances to evaluate the similarity between the input pattern and each prototype.

We use character recognition rate C_r, average character recognition time $T_{av\_rec\_t}$, and memory consumption to evaluate the performance of character recognition. Table 3 shows the results. For reference, the trained weights gotten by CRF are as follows:

$$(\lambda_1, \lambda_2, \lambda_3) = (0.28, 0.48, 0.94).$$

From the weighting parameters, we can see that the weighting parameter λ_3 for state transition probabilities is the highest and the weighting parameter λ_1 for unary features is the lowest.

Table 3. Comparison of recognition models.

Performance	Method	MRF with weighting parameters		MRF	LTM
		CRF	MCE		
Test	*Cr* (%)	92.77	92.53	92.30	89.67
	$T_{av\_rec\_t}$(s)	0.003	0.003	0.003	0.002
	memory	12MB	12MB	12MB	149KB

From the results, we can see that the MRF model remarkably improved the character recognition accuracy, although it consumed slightly more processing time and larger memory space compared with LTM. Introducing the adjustable weighting parameters to the MRF model yielded better recognition accuracy than not using them, and the CRF method for estimating the weighting parameters was more accurate than the MCE method.

B. Analysis of Misrecognitions

Figure 3 shows some examples of misrecognition produced by the proposed model. For each example, the upper line is the written character and the lower line is the recognition result followed by the correct result (ground-truth). These recognition errors are due to similar characters. To avoid them, we need to improve the character recognition accuracy. Exploiting linguistic context can dramatically reduce such misrecognitions.

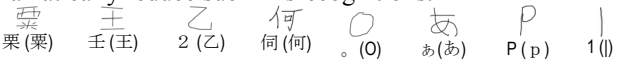


Fig. 3. Examples of recognition errors. The character below each character pattern is the recognition result, followed by the ground-truth.

VII. Conclusion

We presented a method of on-line handwritten Japanese character recognition using MRFs with weighting parameters optimized on the basis of CRFs. The method effectively integrates unary features and binary features, uses adjustable weighting parameters, and optimizes them. Experimental results demonstrated the superiority of our method.

Improving recognition performance is the aim of our future work. This can be achieved by incorporating more effective unary and binary features and exploiting better weighting parameters. Speeding up recognition is another goal.

Acknowledgment

This work was supported in part by an R&D fund for development of pen & paper based user interaction by the Japan Science and Technology Agency.

References

[1] M. Liwicki and H. Bunke, "HMM-based On-line Recognition of Handwritten Whiteboard Notes," Proc. 10th Int'l Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 595-599, 2006.

[2] Y. Katayama, S. Uchida and H. Sakoe, "HMM for On-Line Handwriting Recognition by Selective Use of Pen-Coordinate Feature and Pen-Direction Feature (in Japanese)," IEICE Trans. Information and Systems, Vol. J91-D(8), pp. 2112-2120, 2008.

[3] S. Z. Li, Markov Random Field Modeling in Image Analysis, Springer, Tokyo, 2001.

[4] J. Zeng and Z.-Q. Liu, "Markov Random Fields for Handwritten Chinese Character Recognition," Proc. Eighth Int'l Conf. Document Analysis and Recognition, Seoul, pp. 101–105, 2005.

[5] X.D. Zhou and C.L. Liu, "Text/non-text Ink Stroke Classification in Japanese Handwriting Based on Markov Random Fields," Proceedings of the Ninth International Conference on Document Analysis and Recognition, Curitiba, Brazil, pp. 377-381, 2007.

[6] S.J. Cho, J.H. Kim, "Bayesian Network Modeling of Strokes and their Relationships for On-line Handwriting Recognition," Pattern Recognition, 37, pp. 253-264, 2004.

[7] J. Lafferty, A. McCallum, and F. Pereira, "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data," Proc 18th ICML, pp. 282-289, 2001.

[8] B.-H. Juang and S. Katagiri, "Discriminative Learning for Minimum Error Classification," IEEE Trans. Signal Processing, 40(12), pp. 3043-3054, 1992.

[9] S. Shetty, H. Srinivasan, and S. Srihari, "Handwritten Word Recognition Using Conditional Random Fields," Proc. 9th ICDAR, pp. 1098-1102, 2007.

[10] X.D. Zhou, C.L. Liu, and M. Nakagawa, "Online Handwritten Japanese Character String Recognition Using Conditional Random Fields," Proceedings of the Tenth International Conference on Document Analysis and Recognition, Barcelona, Spain, 2009.

[11] M. Nakagawa and K. Matsumoto, "Collection of On-line Handwritten Japanese Character Pattern Databases and their Analysis," Int. J. Document Analysis and Recognition, 7(1), pp. 69-81, 2004.

[12] U. Ramer, "An Iterative Procedure for the Polygonal Approximation of Plan Closed Curves," Computer Graphics and Image Processing, vol. 1, pp. 244-256, 1972.

[13] Y. LeCun, S. Chopra, R. Hadsell, R. Marc'Aurelio, and F. Huang, A Tutorial on Energy-Based Learning. In: G. Bakir et al. (Eds.), Predicting Structured Data, MIT Press, 2007.

[14] H. Robbins and S. Monro, "A Stochastic Approximation Method," Ann. Math. Stat. 22, pp. 400-407, 1951.

[15] A. Kitadai and M. Nakagawa, "A Learning Algorithm for Structured Character Pattern Representation used in On-line Recognition of Handwritten Japanese Characters," Proc. 8th International Workshop on Frontiers in Handwriting Recognition (IWFHR), Niagara-on-the Lake (Canada), pp. 163-168, 2002.