Segmentation Based Online Word Recognition: A Conditional Random Field Driven Beam Search Strategy

Arti Shivram[1], Bilan Zhu[2], Srirangaraj Setlur[1], Masaki Nakagawa[2] and Venu Govindaraju[1]
[1]Center for Unified Biometrics and Sensors, Department of Computer Science and Engineering, University at Buffalo, NY
[2]Department of Computer and Information Sciences, Tokyo University Agriculture and Technology, Tokyo, Japan
{ashivram, setlur, govind}@buffalo.edu, {zhubilan,nakagawa}@cc.tuat.ac.jp

***Abstract*— In this paper we undertake recognition of online unconstrained cursive handwritten English words. In contrast to popular dynamic programming or HMM-based approaches we propose a Conditional Random Field (CRF) driven beam search strategy applied in a combined segmentation-and-recognition framework. First, a candidate segmentation lattice is built using over-segmented primitives of the word patterns. Recognition is accomplished by synchronously matching lexicon words with nodes of the lattice. Probable search paths are evaluated by integrating character recognition scores with geometric and spatial characteristics of the handwritten segments in a CRF (conditional random field) model. To make computation efficient, we use beam search to prune the set of likely search paths. This overall system has been benchmarked on a new publicly available dataset - IBM_UB_1 as well as on the existing UNIPEN dataset for comparison.**

Keywords—recognition; trie-lexicon; beam search; Conditional Random Field; online; cursive; unconstrained handwriting (key words**).**

I. INTRODUCTION

The pervasiveness of mobile touch screen computing devices like tablets and smartphones has led to a push towards more fluid interaction with these electronics. A natural way of entering text is by writing. This has ushered the development of applications that seek to recognize electronic handwritten content. Google's recently launched 'Google-handwrite' feature facilitates handwritten search queries [1]; also, the German automotive manufacturer Audi equipped a few of their models with handwriting-supportive on-board computers to take in user instructions as an alternative to pressing buttons [1]. An important issue with such applications is that technology should be robust to recognizing handwriting with all the intrinsic stylistic issues and noise peculiar to a person's writing rather that requiring the user to modify his/her writing style to accommodate the system. Neither should it require person-specific, extensive initial training. Only then can it be looked at as a natural user experience. Hence, creating a writer-independent, real-time, online, cursive handwriting recognition system is necessary. This forms the main goal of the current research. The system we describe in this paper requires minimal pre-processing of data for stylistic normalizations and adopts a Conditional Random Field driven beam search algorithm that in turn, uses a trie-structure to efficiently search through lexicon entries. We build and test this system on a new publicly available dataset, namely, IBM_UB_1 [2]. This data was collected on the CrossPad™ device which was modeled as a notepad where users wrote on actual paper using a special pen. Thus, samples in this dataset reflect the variety and complexity involved in real life handwriting.

II. MOTIVATION

When dealing with automatic recognition of writing, a number of issues arise that vary in complexity and nature depending on the mode of data capture (online or offline), pattern of writing (discrete, cursive, mixed), representation of data to the recognizer (characters, sub-characters, word-level features etc.) and the underlying objective of the recognition task [3]. In the online mode, the main objective is to recognize the word as the user writes it [4] i.e., real-time recognition. This requires fast processing algorithms. With this objective in mind we have taken two design decisions – (1) we build our lexicon using a trie-structure that speeds up processing considerably in comparison to a flat-structure as shown in [5], and, (2) we use a beam search algorithm that prunes out lexicon paths that are not likely to yield good results. This is in contrast to past research approaches which rely on performing an exhaustive search of all words in the lexicon. Examples of such approaches include the HMM-based models in [6] and [7] as also in the work of Graves et al. [8] who use a recurrent neural network (RNN) to recognize unconstrained English words in sentences.

With regard to data representation, there are two broad approaches to recognize written words. One, which is often called the analytical approach, treats a word as a sequence of sub-units (such as characters, strokes etc.). It first breaks the word down to basic units, analyzes them, and ties them together [9]. For example, in the HMM based systems of Hu et al. [7] and Liwicki et al. [6] each character is modeled as a separate HMM and lexicon word HMMs are constructed by concatenating character HMMs. In contrast, the other approach treats a word as a whole indivisible unit and aims to recognize it using characteristics of the entire word. This is referred to as the holistic or word-based approach [9]. The main advantage of an analytical approach is that a large

1520-5363/13 $26.00 © 2013 IEEE
DOI 10.1109/ICDAR.2013.174

number of words can be modeled using a finite set of sub-units such as characters, and therefore, we opt for an over-segmentation based algorithm that divides each word into primitive segments (character or part of a character) that are later merged in an integrated segmentation-and-recognition framework. However, breaking apart a word into candidate character segments poses significant challenges. This is especially the case for words written either entirely in cursive or a mixed style [4]. Thus, in order to better identify characters in the word pattern, we use a model-driven (Conditional Random Field) approach that combines shape and geometric features of likely primitives to construct possible character segments and simultaneously recognize the word. To the best of our knowledge the only other CRF implementation for English word recognition pertains to offline handwriting [10]. Shetty et al.'s approach also significantly differs from ours in that they employ dynamic programming as opposed to our trie-lexicon based beam search. Moreover, our lexicon is over an order of magnitude larger in comparison (5000 vs. 300).

Following our motivation and design choices, we now explain the recognition system in detail below.

III. Preprocessing and Over-segmentation

Given a word, we first normalize it to a predetermined height and extract feature points using the method described in [11]. The feature points are so extracted that the overall shape of the word pattern as well as the pen-down and pen-up points of each stroke are retained while simultaneously down-sampling the number of online coordinate points substantially. Subsequently, we remove delayed strokes and detect the baseline and corpus line of the word using regression lines that approximate the local minima and maxima of the stroke trajectory as described in [5] (Figure 1). We then set the 'pen-down', 'pen-up' and the 'minima' and 'maxima' points as candidate segmentation points $P=\{p_1, p_2, p_3, \ldots, p_g\}$ which are used to over-segment the word into primitives.

(a) (b)

Figure 1: Preprocessing. (a) Original word (b) Preprocessed word – black dots denote feature points, red denote pen-down/up; blue and green denote maxima and minima respectively.

IV. Recognition System

A. Candidate lattice construction

The first module of the recognition system involves creating a candidate lattice for the word to be recognized from the over-segmented primitives. One or more consecutive primitive segments are combined to form a candidate (probable) character pattern. All possible sequence combinations of these candidate character pattern strings are represented in a lattice where each node in the lattice denotes a candidate character pattern and each edge denotes a segmentation point leading to the next pattern in the sequence. In order to skip ligatures, we define an edge such that each candidate character pattern that exists between segmentation points p_k and p_l (k, $l \in 1 \sim g$) is followed by candidate character patterns that start from segmentation point p_{l+1} (instead of segmentation point p_l). We also do not start a candidate character pattern from a pen-down point and do not end a candidate character pattern at a pen-up point. Figure 2 shows a segmentation candidate lattice (henceforth referred to as 'lattice') for a sample word.

Paths in the lattice are assumed to end at a terminal node. For each node in the lattice a vector of possible lengths to the terminal node is calculated. This is done by first setting the length of all candidate nodes with the final primitive segment as one and then working backwards one level at a time. Thus, for each preceding node, the length vector is calculated by adding one to the length of its succeeding nodes in the lattice sequence. This is shown in Figure 2, where the numbers in each node box refer to the possible lengths. This length vector is used in conjunction with the lexicon to prune out unlikely search paths (i.e., lexicon words with lengths different from that of the current lattice path) thereby improving both recognition accuracy and speed.

B. Search and Recognition

The second module involves matching sequences in the lattice with entries in the lexicon. For this purpose, the lexicon is set up as a trie [5] (Figure 3). In the lattice, the search space is expanded for each depth (or level) by synchronously matching every node at a particular depth to characters at a similar depth in the trie. This is done by restricting the search to only those word paths in the trie that have the same preceding path sequence as well as the same succeeding path lengths as that of the lattice node being expanded. Each search path ending at a matched node-character pair (probable word string) is evaluated according to a path evaluation criterion set by the CRF model detailed in the next section. Here, adopting a beam search, a pre-specified 'beam width' is used to prune the search paths at every depth level, i.e., for a beam width of 'n', at every level the paths with the top 'n' scores are retained. All other paths outside of this width are pruned away.

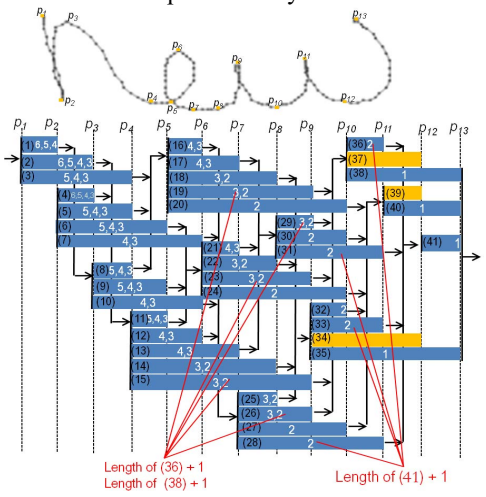

Figure 2: Segmentation Candidate Lattice

We walk through the above described character-synchronous search procedure using a synthetic example (Figures 2, 3 and 4). In Figure 4, d_1 - d_4 represent the depth levels in the search space. Starting with the root node, we observe that lattice nodes (1), (2) and (3) (Figure 2) are expanded synchronously at depth d_1 for possible character matches from the lexicon (Figure 3). At $N_{1\text{-}1}$ lattice node (1) can be matched to three possible characters – 'n', 'r' and 't'. The case is similar for $N_{1\text{-}2}$. At $N_{1\text{-}3}$ since the length vector for node (3) accommodates a length-to-terminal of 5, 4 or 3 and the lexicon paths from 't' do not match these possible lengths (they possess lengths of 2 or 6), we drop these paths. This forms the expansion for depth d_1 where we have eight pattern-character matches in total. At this stage, all eight paths are evaluated according to the path evaluation criterion and a few top scoring paths are selected while others are pruned out. The number of selected paths is called the beam width. In our synthetic example the beam width is two and for node (2), 'r' is selected whereas for node (3), 'n' is selected (highlighted in red in Figure 4). Similarly, at depth d_2 this process is repeated to expand the search. At depth d_2, following the lattice, we note that for node (2) the possible successor nodes are (11), (12), (13), (14) and (15) (which refer to $N_{2\text{-}1}$, $N_{2\text{-}2}$, $N_{2\text{-}3}$, $N_{2\text{-}4}$ and $N_{2\text{-}5}$ respectively) whereas for node (3), the possible successor nodes are (16), (17), (18), (19) and (20) (which refer to $N_{2\text{-}6}$, $N_{2\text{-}7}$, $N_{2\text{-}8}$, $N_{2\text{-}9}$ and $N_{2\text{-}10}$ respectively). For $N_{2\text{-}1}$, the two possible character categories are 'a' and 'o' and both are retained since paths from both satisfy the length requirement of node (11). At $N_{2\text{-}2}$, the two possible lexicon characters are 'a' and 'o'. Here, only one satisfies the length requirement of node (12) ('o' which has the length 3). Hence, the path from 'a' is dropped. Similarly we process the cases for $N_{2\text{-}3}$ up to $N_{2\text{-}10}$ for depth d_2. Finally, this expansion is repeated for depth d_3 and d_4. At d_4, the path ending at node (35) – 'w' pair is carried forward from d_3 because 'w' is a terminal node in the lexicon (as long as this path score falls within the beam width). This procedure yields two candidates – 'note' and 'new' for recognition.

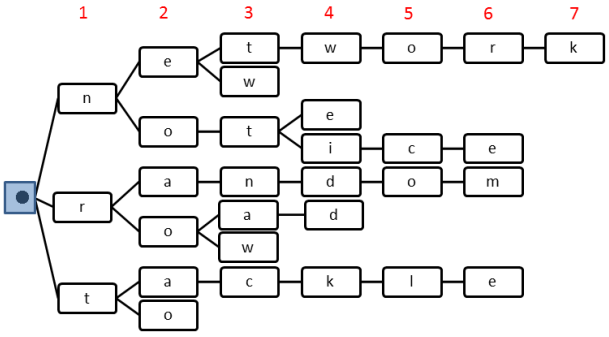

Figure 3: Lexicon Trie

It may be observed here that a particular candidate character pattern (lattice node) may appear in several search paths (possible word sequences). This would require matching of the same node with possible character matches from the trie for different path evaluations. Evaluating such node-character matches using a character recognizer requires extraction of feature points from the pattern primitives. In order to avoid redundant processing, once a node's features are extracted, we store them. These are then used in subsequent calls. Additionally, for a node and a character category, we store the recognition score at the first recognition instance and use this stored node-character pair score when called upon for subsequent evaluations. In our experiments, these steps greatly improved recognition speed (by about eight times).

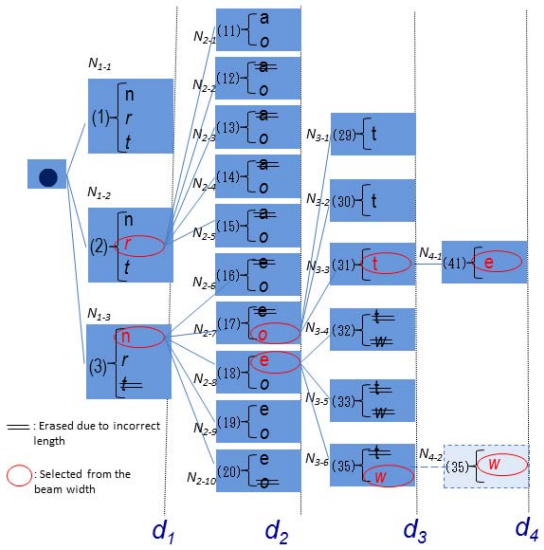

Figure 4: Character Synchronous Search

V. CRFs FOR WORD RECOGNITION

A. CRFs on Segmentations

Consider an input word pattern X which is over-segmented into a sequence of primitives using a set of candidate segmentation points $P=\{p_1, p_2, p_3, \ldots, p_g\}$. Relating this to the segmentation lattice described earlier, one can see that a start-to-end path on the lattice is analogous to a probable segmentation sequence S_i of X. Further, assume that Y refers to the label sequence of S_i.

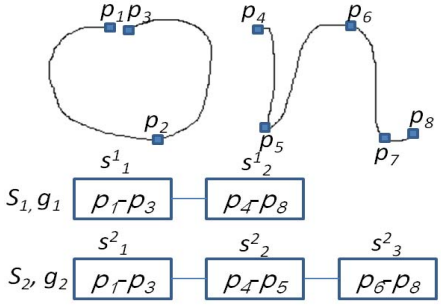

Figure 5: Neighborhood graphs of Probable Segmentations

Thus, for a given word, there are many probable segmentations $\{S_1, S_2, S_3, \ldots, S_m\}$, where each $S_i=\{s^i_1, s^i_2, s^i_3, \ldots, s^i_{n\_i}\}$ and s^i_j denotes a candidate character pattern between segmentation points p_k and p_l ($k, l \in 1\sim g$). For each S_i we construct a neighborhood graph g_i (Figure 5), such that each node denotes a candidate character pattern and each link

represents the relationship between neighboring candidate character patterns. Let (s^i_{j-1},s^i_j) denote a pair of neighboring candidate characters s^i_{j-1} and s^i_j. In our CRF framework, s^i_j and (s^i_{j-1},s^i_j) correspond to unary (single) and binary (pair-wise) cliques, respectively.

From the definition of CRF, $P(Y|S_i,X)$ can be approximated by an arbitrary real-valued energy function $E(X,S_i,Y,N,\Lambda)$ with clique set N and parameters Λ as:

$$P(Y \mid S_i, X) = \frac{\exp(-E(X, S_i, Y, N, \Lambda))}{Z(S_i, X)}$$
$$Z(S_i, X) = \sum_{(Y')} \exp(-E(X, S_i, Y', N, \Lambda)) \quad (1)$$

Since $Z(S_i,X)$, the normalization constant does not depend on Y, it may be ignored if we do not require strict probability values. Then, the problem of finding the best label Y^*, that involves maximizing $P(Y|S_i,X)$, becomes equivalent to minimizing the total graph energy:

$$Y^* = \text{argmax}_y\, P(Y \mid S_i, X) = \text{argmin}_y\, E(X, S_i, Y, N, \Lambda) \quad (2)$$

Utilizing only unary and binary cliques in our CRF model, the total energy function can be defined as

$$E(X, S_i, Y, N, \Lambda) = -\sum_{(s^i_{j-1}, s^i_j) \in S_i} \sum_{k=1}^{K} \lambda_k f^k_{(s_{j-1}, s_j)}(y_{j-1}, y_j) \quad (3)$$

where $f^k_{(s_{j-1},s_j)}(y_{j-1},y_j)$ are feature functions on a binary clique (s^i_{j-1},s^i_j), $\Lambda=\{\lambda_k\}$ are weighting parameters, y_{j-1} and y_j denote the labels of s^i_{j-1} and s^i_j, respectively. Without loss of generality, in eq. (3) we use only binary cliques to describe the feature functions as we assume that they (s^i_{j-1},s^i_j) subsume unary cliques s^i_j. Here, the total energy function is used to measure the plausibility of Y, and the smaller $E(X,S_i,Y,N,\Lambda)$ is, the larger will $P(Y|S_i,X)$ be.

In the word recognition task the focus now is on selecting the best path from all possible segmentations $\{S_1, S_2, S_3, \ldots, S_m\}$. Since all paths need not be of uniform length, we need to normalize the path scores. Therefore, we use the following path evaluation criterion to select the best path from all segmentations-cum-recognitions.

$$EC(S_i, Y, X) = \frac{E(X, S_i, Y, N, \Lambda)}{N_{(S_i, X)}} \quad (4)$$

Where $N_{(S_i,X)}$ denotes the number of binary cliques (length of word).

B. Parameter Learning

We apply the MCE criterion [12] optimized by stochastic gradient descent [13] to find the optimal parameter vector Λ by maximizing the difference between the evaluation criterion of the most confusing (S_i,Y) and that of the correct one:

$$L_{MCE}(\Lambda, X) = \sigma(\min(EC_f(S_i, Y, X)) - EC(S_t, Y_t, X))$$
$$\sigma(x) = (1 + e^{-x})^{-1} \quad (5)$$

where $EC(S_t,Y_t,X)$ and $EC_f(S_i,Y,X)$ are the evaluation criteria of the true path and of the most confusing path respectively.

C. Feature Functions

In CRF, feature functions are used to capture the node attributes (unary) and local dependencies between nodes (binary in our case). Further, they can be broadly classified into class-relevant/irrelevant on the basis of dependence/independence on the character class.

In our model we use two character recognition scores and five geometric feature functions. Of the two character recognition scores for each candidate pattern, one is given by a P2DBMN-MQDF classifier on direction histogram features [14] and another is given by an MRF classifier [11]. Among the geometric features are three unary features that capture the character structure (such as the number of down strokes), size (height, width) and its relative position in the word (distance from the center line). The binary feature functions measure overlap and positional differences between adjacent character patterns. Though the latter is to be calculated for all possible pairs of characters, we simplify the process by clustering the characters into four super-classes according to the mean vector of their unary position features. Each of these geometric features are extracted as feature vectors and transformed to log-likelihood scores using quadratic discriminant function (QDF) classifiers. The feature functions are summarized in Table 1.

TABLE I. SUMMARY OF FEATURE FUNCTIONS

	Type	Features	Classifier
f_1	Unary class-rel.	Character shape	P2DBMN-MQDF
f_2	Unary class-rel.	Character structure	MRF
f_3	Unary class-rel.	Unary geometric (Down stroke number and inner gap)	QDF
f_4	Unary class-rel.	Unary geometric (Character size)	QDF
f_5	Unary class-rel.	Unary geometric (Single-character position)	QDF
f_6	Binary class-irrel.	Binary geometric (Pair-character position)	QDF
f_7	Binary class-rel.	Binary geometric (horizontal overlap)	QDF

VI. EXPERIMENTATION

In order to build and empirically validate our model we concurrently utilized two publicly available datasets – IBM_UB_1 and UNIPEN [15]. We chose this concurrent approach in order to overcome the absence of true character segmentation points for words in the IBM_UB_1 dataset. Further, this also helped us benchmark against an already existing and widely adopted handwriting dataset (UNIPEN).

IBM_UB_1 consists of a twin-folio structure where each author-specific file consists of a summary-query pair. The summary is a full page of unconstrained cursive writing whereas the query page consists of approximately 25 handwritten words appearing in the summary counterpart. For this research we used only the data from the query pages which is at the word-level granularity.

Within the UNIPEN dataset we found that a subset of samples in the train_r01_v07 folder were shared across two categories – isolated characters (Benchmark #3) and isolated words (Benchmark #6). Specifically, the "art" folder with 6

writers, “cea” with 6 writers, “ceb” with 4 writers and, “kai” with 28 writers. By mapping coordinate information from character to word categories for these samples, we were able to obtain words samples with true character-level segmentation labels. Using this data (14,691 characters, 2,127 words, 44 writers), we trained our CRF-based framework to obtain initial (a) weighting parameters and (b) QDF classifiers of the geometric feature functions. With this initialized model and a lexicon of a single word (the ground truth word) we were able to deduce the approximate segmentation for each word in the IBM_UB_1 dataset. The creation of character-level segmentation points for words in IBM_UB_1 represents the first phase of experimentation.

In the second phase, we retrained and rebuilt our model using the updated IBM_UB_1 data. For this we selected four pages at random from 20 writers as testing data and used the remainder for training. Due to the data-intensive nature of CRF training, a majority of the data was used for training. The training set consisted of 61,105 words and 355,895 characters across 62 categories (digit, uppercase and lowercase Latin alphabet), while the test set consisted of 1,795 words and 10,987 characters. Further, we used the above retrained model for testing on the UNIPEN subset too to provide results for comparison (Table 2). For both experiments the beam band is set at 1000.

TABLE II. RECOGNITION RESULTS

Dataset	Lexicon Size (words)	Test Set Size (words)	Recognition Rate
IBM_UB_1	5000	1795	78.72%
UNIPEN	2127	2127	92%

Though we provide recognition rates for the UNIPEN dataset, it must be noted that it consists primarily of laboratory samples and is not reflective of free, unconstrained handwriting from the field. The only other relatively close dataset to IBM_UB_1 is the IAM-OnDB dataset. Here too, since the dataset consists of sentences and not individual words we are unable to perform appropriate benchmark tests. Extant research using IAM-OnDB has relied upon language models for enhanced recognition performance [6, 8]. Nevertheless, our model – without the advantage of language models – compares favorably (Table 3). A second issue to note is that in [8], which uses RNN with a language model, the metric used for assessing recognition is similar to string matching using Levenshtein distance. This measure allows for more differences in characters within a word and thus is more forgiving than a simple binary correct/incorrect metric. Our results are presented using the more stringent correct/incorrect metric.

TABLE III. ONLINE UNCONSTRAINED ENGLISH WORD RECOGNITION COMPARATIVE RESULTS

Dataset	Data Type	Model	Recognition Rate
IBM_UB_1	Word Level	CRF Beam	78.72%
IAM OnDB	Sentence Level	HMM + LM	70.8%
IAM OnDB	Sentence Level	RNN + LM	79%

Recognition results with different combinations of feature functions are summarized below (Table 4). As evident, the P2DBMN-MQDF and MRF based features and search length restrictions contribute significantly to overall accuracy.

TABLE IV. EXPERIMENT RESULTS

Method / Perfor.	*Without f_1*	*Without f_2*	*Without f_3-f_7*	*Without length restrictions*	*Using all feature functions*
Word rec. rate	71.92%	72.53%	77.05%	72.70%	78.72%

VII. CONCLUSION

We explored a CRF-driven beam search method to recognize unconstrained cursive online English words. Combining a trie-lexicon with a character-synchronous lattice search algorithm, we achieve recognition rates that compare favorably to the current state of the art. Our contribution is two-fold: (a) application of a beam search strategy to enable efficient processing of the search space, and, (b) merging beam search with a CRF model that combines both feature probability scores and character recognition scores to improve performance. This, we believe, is a promising avenue for future research.

REFERENCES

[1] K. Shabanova. “Do Androids dream of handwriting recognition?,” http://www.manufacturing.net/articles/2012/10/do-androids-dream-of-handwriting-recognition.

[2] U. a. B. Center for Unified Biometrics and Sensors, “IBM-UB Online and Offline Multi-lingual Handwriting Data Set,” 2012.

[3] S. Connell, and A. K. Jain, “Online handwriting recognition using multiple pattern class models,” unpublished, 2000.

[4] C. C. Tappert, and S.-H. Cha, *English language handwriting recognition interfaces*: San Francisco: Morgan Kaufmann, 2007.

[5] S. Jaeger, S. Manke, J. Reichert, and A. Waibel, “Online handwriting recognition: the NPen++ recognizer,” *International Journal on Document Analysis and Recognition,* vol. 3, no. 3, pp. 169-180, 2001.

[6] M. Liwicki, and H. Bunke, “HMM-based on-line recognition of handwritten whiteboard notes,”*Frontiers in Handwriting Recognition, Proc 10th Int. Workshop on,* pp. 595-599, 2006.

[7] J. Hu, M. K. Brown, and W. Turin, “HMM based online handwriting recognition,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 18, no. 10, pp. 1039-1045, 1996.

[8] A. Graves, S. Fernández, M. Liwicki, H. Bunke, and J. Schmidhuber, “Unconstrained online handwriting recognition with recurrent neural networks,” *Adv. in Neural Information Processing Systems,* vol. 20, pp. 1-8, 2008.

[9] S. Madhvanath, and V. Govindaraju, “The role of holistic paradigms in handwritten word recognition,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on,* vol. 23, no. 2, pp. 149-164, 2001.

[10] S. Shetty, H. Srinivasan, and S. Srihari, “Handwritten word recognition using conditional random fields.” *Proc. 9th Int. Conf. Document Analysis and Recognition,* pp. 1098-1102, 2007.

[11] B. Zhu, and M. Nakagawa, “A MRF Model with parameter optimization by CRF for on-line recognition of handwritten Japanese characters,” *Proc. Doc. Recognition and Retrieval XVIII*, pp. 1-10, 2011.

[12] B.-H. Juang and S. Katagiri, “Discriminative learning for minimum error classification,” *IEEE Trans. Signal Processing,* 40(12), pp. 3043-3054, 1992.

[13] H. Robbins and S. Monro, “A stochastic approximation method,” *Ann. Math. Stat. 22*, pp. 400-407, 1951.

[14] C.-L. Liu and X.-D. Zhou, “Online Japanese Character Recognition Using Trajectory-based Normalization and Direction Feature Extraction,” *Frontiers in Handwriting Recognition, 10th Int. Workshop on,* pp.217-222, 2006.

[15] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, “UNIPEN project of on-line data exchange and recognizer benchmarks.” *Pattern Recognition, Computer Vision & Image Processing., Proc. of the 12th IAPR Int. Conf. on,* vol.2., pp. 29-33, 1994.