A generic approach for on-line handwriting recognition

Sanparith Marukatat and Thierry Artières and Patrick Gallinari
LIP6, Université Paris 6
8, rue du Capitaine Scott, 75015
Paris, France
{Sanparith.Marukatat,Thierry.Artieres,Patrick.Gallinari}@lip6.fr

Abstract

We present here thorough experimental studies of a generic approach for developing on-line handwriting recognition systems. The basic principles of our approach lead to generic intrinsic properties that we investigate. These properties allow building various recognition engines corresponding to different needs in pen-based interaction.

Keywords: Generic Online Handwriting Recognition

1 Introduction

We present in this paper thorough experimental studies of a generic approach for developing on-line handwriting recognition (on-line HWR) systems that has been partially presented in [1, 11]. This paper focuses on an extensive experimental study of our system which is is very similar to the one presented in previous work, hence we do not present it in detail here for place reasons. Note that, although we deal with isolated character recognition only, our approach, being based on Hidden Markov Models (HMM) character models, may be further extended to word recognition.

Our approach is based on three basic ideas: the use of a "high level" representation of on-line handwriting, the deterministic building of an HMM character model from a "high level" representation and a massive parameter sharing strategy. As we will show, these three ideas together lead to intrinsic properties such as the ability to learn a character model from a few (possibly one) samples, the ability to learn any "simple" graphical character (Korean characters, latin characters, digits, symbols etc), the possibility to build very light single-writer systems -running fast and requiring few memory- for mobile devices as well as writer independent systems performing similarly to state of the art systems.

In the following, we first present briefly our approach and

its main principles. Then we investigate experimentally the intrinsic properties of the approach, e.g. its ability to handle variability and to learn any kind of graphical character from very few training data. Then we discuss how our work may be used to build both a recogniton engine adapted to pen-based interaction for small mobile devices and a writer-independent recognition engine for standard alphabtetic characters. We also provide some details on ressource requirements, memory size and recognition speed and show that our approach compares well with respect to existing systems.

2 A generic approach for building on-line handwriting recognition engines

In this section, we briefly present our approach for building on-line handwriting recognition systems, more details can be found in [1, 11]. We review the basic principles first, then we explain how to compute a SLR then how to build a character model.

2.1 Principles

We detail a little bit more the three main ideas of our approach. First, we use what we call a *Stroke-Level Representation* or *SLR* (see [1, 11]) of on-line handwriting signals, this may be viewed as a complex preprocessing. This representation is richer, and more compact, than classical sequence of low-level feature vectors. The second idea consists in deriving automatically, from a given SLR, an HMM working directly on SLR –we call such HMM *Stroke Level HMM* (*SLHMM*). Combining these two ideas, each training sample may be transformed into a SLHMM so that the training procedure consists in selecting a set of SLHMM. This selection may then be viewed as a choice of HMM models for allographs of the character. The third idea consists in a massive parameter sharing strategy that allows the above building of a SLHMM from a SLR to be efficient. Parameter sharing concerns emission probabilities involved in SLHMM, there are at the end only few free parameters to define all emission probability laws of all character HMM models.
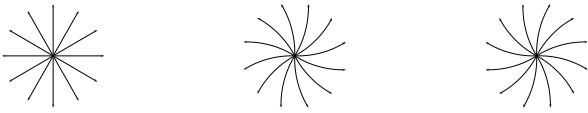
Figure 1. The set of 36 elementary strokes (noted Σ hereafter) used to describe the shape of handwriting signals.

A recognition engine consists in a cascade of two HMM systems. The first system is used to preprocess an input signal (that has been smoothed, resized and spatially resampled) and to compute its SLR (§2.2). This SLR is then input to the second system (§2.3) which is the real recognition engine and consists in Markovian character models.

2.2 SLR extraction

Basically a SLR $\mathcal{R}$ is a sequential representation of the original signal, it consists in three components (S, D, RS) with S the shape component, D the size component and RS the spatial component. The sequence S is composed of N elementary strokes, i.e. $S = s_1, \ldots, s_N$ where s_i is one of the 36 elementary strokes of a disctionary Σ of stroke shapes (see figure 1). The sequence D is also composed of N elements, i.e. $D = d_1, \ldots, d_N$ where d_i is the relative size of the stroke s_i. The last component $RS = rs_1, \ldots, rs_N$ is the spatial relations between each stroke s_i and its predecessing strokes.

To obtain these three components, we consider an HMM whose states corerspond to the elementary strokes in Σ. The state emission probability distributions are implemented through trajectory models. A Viterbi algorithm is used to segment the original signal into a sequence of elementary strokes in this segmental HMM. The two other components are then extracted from this optimal segmentation.

2.3 Recognition Engine

The design of the recognition engine is based on the idea that an HMM operating on SLR (a SLHMM) may be completely defined (emission probabilities and topology) from a given SLR. Based on a SLR $\mathcal{R} = (S, D, RS)$ with S a sequence of N elementary strokes, we build a Left-Right HMM $\lambda_{\mathcal{R}}$ with N states. Each state of $\lambda_{\mathcal{R}}$ is associated with an elementary shape s_i, a duration d_i and spatial relation rs_i.

As in [11], the parameters of emission probability laws, defined over Σ, are shared among all state in all SLHMM of the system. For example, if two states e_i and e_j (not necessarily in the same SLHMM) are associated to the same elementary shape s of Σ, then $\forall s' \in \Sigma$, $p(s'|e_i) = p(s'|e_j) = p(s'|s)$. This parameter sharing strategy is also applied to the duration and the spatial component modeling. This strategy allows reducing significantly the number of free parameters

in the system. Such an heuristic building of SLHMM allows defining, from a SLR R, a SLHMM $\lambda_{\mathcal{R}}$ that handles small variability around the SLR it is based on. This means that $\lambda_{\mathcal{R}}$ gives high likelihoods for SLR very close to $\mathcal{R}$ only. Therefore the modeling ability of such SLHMM is quite limited. To build character models able to handle much more variability (including allographs) we consider character models that are mixture of SLHMM. The model for character c is a mixture of SLHMM, denoted by M_c, and the probability of observing a SLR $\mathcal{R}'$ is given by :

$$p(\mathcal{R}'|M_c) = \sum_{k=1}^{K_c} p(\mathcal{R}'|\lambda_{c,k})\, p(\lambda_{c,k}) \qquad (1)$$

where K_c is the number of SLHMM in the mixture (it is called the model size in the following), $\lambda_{c,k}$ is the kth SLHMM of the character c and $p(\lambda_{c,k})$ is its prior. Hence, such models may take into account the variability (intra and inter character as well as intra and inter writer) by increasing the size of character models. This number is an hyperparameter that allows to easily choose the tradeoff between the modeling power of character models and the recognition system complexity. For writer-independent tasks, model size of about 20 to 50 are required to reach good performances whereas for writer-dependent systems a model size equal to 5 is most often enough. The training procedure for a character model consists in selecting a set of SLHMM (among the set of SLHMM built from SLR of training samples) which best represent this character. This is done with a K-means like clustering algorithm in SLHMM space, using a distance between SLHMM.

3 Experiments

We have conducted many experiments in order to deeply investigate the properties of our approach. We first present the databases used in our experiments, then we investigate basic properties of our approach. Next, we discuss the feasibility of building two very different recognition engines with our approach: a writer-independent system for classical latin character recognition and a writer-dependent system for small mobile devices able to learn fastly any graphical character. The main problem for the writer-independent system is variability handling while the problems for the writer-dependent system are limited training set size and handling of any graphical character. At last, we compare our system to other recognition engines in terms of computational ressources required.

3.1 Databases

3.1.1 Digits, Lowercase and Uppercase Letters (UNIPEN database)

The UNIPEN database [6] is a standard benchmark to evaluate on-line HWR system. In this work, we use part of the
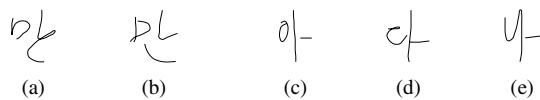
IEEE COMPUTER SOCIETY

Figure 2. Variability and complexity of Korean characters: two different writing styles for the same character (a) and (b); three characters with similar writing styles (c), (d) and (e).

Figure 3. Miscellaneous symbols of LIP6 database.

(a)

(b)

Figure 4. Recognition results for varibility in the drawing (a) and inter-writer variability (b).

database (version R01/V06) containing isolated digits, uppercase and lowercase letters (directories 1a, 1b and 1c). This part of the database contains signals from more than 200 writers with 15719 digits (resp. 22683 uppercase and 59691 lowercase letters) in the whole.

### 3.1.2 Digits and Korean characters (KAIST database)

The KAIST database[1] contains on-line handwriting signals from Korean highschool students. In this work, we use part of the database correspnding to isolated digits and Korean characters from about 20 writers.

The recognition of these Korean characters is difficult since a character is usually written with several pen lifts, with or without ligatures and each natural stroke (the drawing between two pen lifts) may be quite complex. In addition, the spatial disposition of strokes may be very important to distinguish some characters. Figure 2 illustrates such a difficulty: the figures (a) and (b) show two writing styles of the same character while figures (c), (d) and (e) show three similar writings for three different characters. In our experiments, we use 19 distinct characters, those characters for which enough samples are available for testing in writer-dependent context.

3.1.3 Miscellaneous Symbols (LIP6 database)

We also considered a set of miscellaneous symbols, written by people in our laboratory, as shown in figure 3. These symbols are used to evaluate more deeply of our system properties (with respect to spatial information handling) since some of these symbols share the same shape and can be distinguished by the spatial positions of their components only. There are 32 symbols. It should be noted that some of these symbols are not standard; a consequence is that the drawings for these latter symbols are rather variable, e.g. strokes are not always written in the same order. There are 60 samples for each symbol written by three writers.

---

[1] http://ai.kaist.ac.kr/
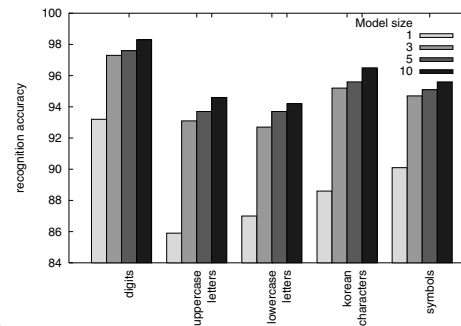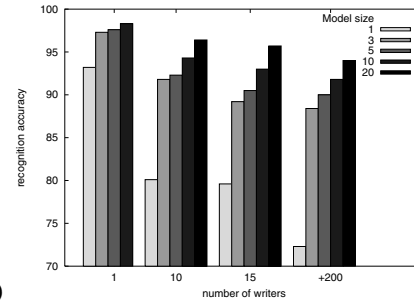
3.2 Intrinsic properties of our approach

The objective of this subsection is to investigate the behavior of our systems in different settings concerning the writing style variability, the inter-writer variability and the training set size.

3.2.1 Variability in the drawing

These first experiments investigate writing complexity handling through character recognition results for character of various complexities in single writer mode only. As the database for each writer is limited, we considered two-fold cross-validation for each writer and averaged results. In these experiments, we consider only characters and writers for whom we have at least 10 samples. This represents 14 writers for digits (from UNIPEN and KAIST database), 5 writers for lowercase letters, 7 writers for uppercase letters, 5 writers for Korean characters and 3 writers for misc. symbols. The results are compiled in figure 4 (a). It is not straightforward to compare all these accuracies since these do not correspond to a same number of writers and characters. The main point is the systematic increasing of accuracies when the model size increases, and that in any case recognition rate of 95% may be achieved, whatever the characters.

3.2.2 Writer variability

In order to study the behavior of our approach with writer variability, we conducted experiments with the digits from the UNIPEN database with different numbers of writers. Figure 4 (b) shows recognition rates for single writer experiments as well as for writer-dependent experiment with 10, 15 and more than 200 writers. For each experiment, results are given for different character model sizes. One may see that for a given model size, recognition rates naturally decreases when the number of writers increases (the recognition task become more difficult). But here again increasing model size allows taking into account the increased variability and reaching high accuracies (up to 94%), even with more than 200 writers.

These results suggest that the model size is a simple mecanism for controling the recognition system quality. Adapting the system complexity as a function of desired accuracy may also be done by choosing adapted model size. As we will see in the following (§3.3), recognition accuracy may still be improved by using a larger system.

3.2.3 Training set size

In this subsection, we are concerned with the performance of systems with limited training set size. Since a character model is based on SLHMM that are built from training samples, one can build a new character model from one training sample. To highlight this property, we have conducted experiments in single writer mode, using simple characters (digits) as well as complex characters (Korean characters) from the KAIST database, by varying the training set size and for various model sizes. Again, to avoid the bias due to the training set size, we performed cross-validations and averaged results.

Figures 5 (a) and (b) show average accuracies as a function of the training set size per character and for a few maximum model sizes. From these results, one may see that accuracies of 75% to 84% are reached with only one training sample per character and that higher recognition rates (up to 95%) may be reached with only 10 training samples per character.

3.3 Design of recognition systems

The above experiments have shown some main properties of our approach which may be useful in the design of recognition systems adapted to different contexts. We show now how our approach may be used to build two very different engines.

3.3.1 Pen-based interacttion for small mobile devices

The results in the previous sections have shown that our approach allows building single-writer recognition systems with high accuracies (figure 4 (a)) for various graphical characters. Furthermorte, high performances were reached using only very limited training material. Together, these two comments mean that a user could easily use the recognition engine
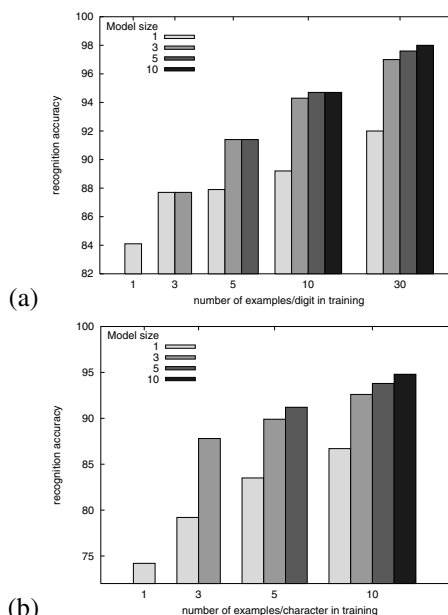

(a)

(b)

Figure 5. Recognition accuracies in single writer experiments for various model sizes and training set sizes: for digits (a) and Korean characters (b).

to recognize his own symbols but also abreviations or command gestures, the only requirement is to provide a few training samples per character or symbol to learn. This makes the personalization of the recognition engine very easy and powerful, which is an essentiel property for pen-based computer as pointed out by computer-Human Interaction (CHI) studies [10]. Finally, we will see in the following (§3.4) that the computational ressources required for a single-writer recognition engine are very limited. For a 26 charaters recognizer, the system requires a few hundred Kilo bytes (model size about 5) and allows recognizing a few hundred characters per second on a standard pentium III 500 MHz machine. Therefore this kind of system could be reasonnably implement on small mobile devices which limited computational ressources.

3.3.2 Writer-independent system for standard characters

In this subsection we show that the same approach may be used to build a writer-independent system for standard characters. In order to compare with other recognition systems, we used signals from the UNIPEN database. As usual, the writer-independent context is splitted in two sub-tasks: multi-writer and writer-independent recognition. Table 1 compares recognition accuracies of our systems with various model sizes to some existing recognition systems. Results are given for different sizes of the training set size, given as a percentage of the whole database. From these results, one may see that our approach allows to reach (with big model size) comparable recognition rates to best systems, for multi as well as

writer-independent recognition tasks. Moreover, in this table we also present a recognition result named 1-nn where we use a 1-nn like classifier with the likelihood (equation 1) as distance. This scheme corresponds to a bigger system but allows to reach even higher accuracies. We may also remark that with very limited training set (i.e. 5% of database), we can already obtain good results.

3.4 Complexity issues

All the results up to now demonstrate the ability of our approach to perform similarly to state of the art systems for standard benchmarks but also the ability to develop very different recognition engines able for example to learn quickly. In this section, we show that our systems exhibit another important property –they are economic in terms of memory requirement as well as in terms of computational complexity. We compare in the following our systems to existing systems cited above in the discussion and tables.

3.4.1 Memory requirement

SVM based systems, e.g. [4, 15], require a large amount of memory to store support vectors. In [4], about 100 support vectors per character are needed to reach results of Table 1; according to the authors, this corresponds to 17.5 Megabytes (Mb) for 26 lowercase letters and 6.7 Mb for 10 digits. Other systems, such as [15] requires 21 Mb to store all support vectors for 26 uppercase letters. The sn-tuple [13] is in theory even more memory demanding. Nevertheless, by limiting the storage to the really observed tuple in the training set, the author reduced the memory requirement to about 19.3 Mb for digit recognition task. In comparison, our systems require much less memory. To give an idea, the storage of *all* SLR corresponding to the digit of the UNIPEN database (15719 examples) requires less than 8.2 Mb (the size of a SLHMM being roughly the same as a SLR). A recognition engine for 26 letters requires about 130 Kb for a model size equal to 5, about 500 Kb for a model size equal to 20 and 1 Mb for a model size equal to 50.

3.4.2 Recognition speed

The table 2 compares recognition speed (number of recognized characters per second) for some systems cited in the table 1 (Machines used for tests are detailed). One may see here that our systems are really fast comparing to most other systems especially prototype-based system such as [15].

4 Conclusions

We provided various experimental results in order to investigate the properties of our approach for on-line HWR.

We have shown that our approach is efficient to model various graphical characters, allows taking into account the variability in the drawing as well as inter-writer variability. We also shown that learning may be done with very limited training samples. These properties allow developing recognition engines adapted to pen-based interaction with strong personalization features as well as writer-independent systems for standard characters competing with state of the art recognition systems.

References

[1] T. Artières and P. Gallinari. Stroke level hmms for on-line handwriting recognition. In *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2002.

[2] T. Artières, J-M. Marchand, P. Gallinari, and B. Dorizzi. Stroke level modelling of on line handwriting through multi-modal segmental models. In *IWFHR*, 2000.

[3] C. Bahlmann and H. Burkhardt. Measuring hmm similarity with the bayes probability of error and its application to online handwriting recognition. In *International Conference on Document Analysis and Recognition (ICDAR)*, 2001.

[4] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines – a kernel approch. In *IWFHR*, 2002.

[5] S. J. Cho and J. H. Kim. Bayesian network modeling of strokes and their relationships for on-line handwriting recognition. In *ICDAR*, 2001.

[6] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benrchmak. In *International Conference on Pattern Recognition (ICPR)*, volume 2, 1994.

[7] J. Hu, S. G. Lim, and M. K. Brown. Writer independent on-line handwriting recognition using an hmm approach. *Pattern Recognition*, 33(1):133–148, 2000.

[8] A. Lemieux, C. Gagné, and M. Parizeau. Genetical engineering of handwriting representations. In *IWFHR*, 2002.

[9] X. Li, R. Plamondon, and M. Parizeau. Model-based on-line handwritten digit recognition. In *ICPR*, 1998.

[10] A. C. Long, J. A. Landay, and L. A. Rowe. Implications for a gesture design tool. In *Conference on Human factors in computing systems (CHI)*, 1999.

[11] S. Marukatat, R. Sicard, T. Artières, and P. Gallinari. A flexible recognition engine for complex on-line handwriting character recognition. In *ICDAR*, 2003.

[12] L. Prevost and M. Milgram. Modelizing character allographs in omni-scriptor frame: a new non-supervised clustering algorithm. *Pattern Recognition*, 21:295–302, 2000.

[13] Eugene H. Ratzlaff. Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database. In *ICDAR*, 2003.

[14] E. Gómez Sánchez, J.A. Gago González, and Y.A. Dimitriadis. Experimental study of a novel neuro-fuzzy system for on-line handwritten unipen digit recognition. *Pattern Recognition Letters*, 19:357–364, 1998.

[15] L. Vuurpijl and L. Schomaker. Two-stage character classification: A combined approach of clustering and support vector classifiers. In *IWFHR*, 2000.

IEEE
COMPUTER
SOCIETY

% UNIPEN in training set	Recognition systems	digits		uppercase		lowercase	
		multi	omni	multi	omni	multi	omni
5	SLHMM						
	model size = 5	92.4	87.5	85.1	80.2	80.9	78.5
	model size = 10	94.1	89.9	86.9	82.7	83.5	80.3
	model size = 20	95.1	90.7	87.9	83.8	85.0	81.7
	model size = 50	95.4	91.9	89.1	84.9	85.9	82.6
20-30	SLHMM						
	model size = 5	92.3	90.3	84.7	85.6	81.7	80.3
	model size = 10	95.1	91.9	86.9	87.8	84.1	82.8
	model size = 20	96.4	93.4	88.6	89.1	86.2	84.5
	model size = 50	97.1	94.9	89.2	89.5	87.9	85.9
	sn-tuple [13]	98.7	97.1	93.7	91.8	90.6	87.8
	SDTW [3]	95.5		90.0		88.6	
	SVM [4]	96.0		92.4		87.9	
	HMM [9]	92.0					
	MLP [8]		95.6				
	knn + SVM [15]			94.8			
50-60	SLHMM						
	model size = 5	93.5	90.7	88.3	87.0	80.9	80.8
	model size = 10	95.2	92.3	90.4	89.3	84.3	83.7
	model size = 20	96.6	94.5	91.9	90.2	86.7	85.7
	model size = 50	97.7	95.9	92.8	91.2	88.2	87.0
	1-nn like	98.5	97.2	94.4	92.6	90.7	88.7
	sn-tuple [13]	98.8	98.1	95.1	94.0	92.0	
	knn [12]		98.8		96.6		96.3
	HMM Segmental [2]					79.5	69.8
	Bayesian Network [5]		95.0				
	Fuzzy ARTMAP [14]	82.4					
	HMM [7]		96.8		93.6		85.9

Table 1. Recognition results of recognition systems for characters from UNIPEN database.

Recognition systems	Number of characters	Recognition speed (#car/sec)	Machine
sn-tuple [13]	10	70	300 MHz RS6000 workstation
	26	420	
SVM with GDTW kernel [4]	26	0.4	AMD Athlon 1200 MHz
Bayesian Network [5]	10	84	IBM ThinkPad T23, 1.13 GHZ
Fuzzy ARTMAP [14]	10	3.57	Pentium 120MHz
HMM [7]	-	3.3	180 MHz SGI station
knn [15] with SVM in post-processing	26	2.88	Pentium-II 400 MHz
our system : model size = 5	10	520	Pentium-III 500 MHz
	26	230	
model size = 50	10	74	
	26	20	

Table 2. Comparaison of recognition speed (given as number of recognized charater per second) between different recognition systems.