Évaluation d'une approche générique pour la reconnaissance de l'écriture manuscrite en-ligne

Sanparith Marukatat, Thierry Artières, Patrick Gallinari

Évaluation d'une approche générique pour la reconnaissance de l'écriture manuscrite en-ligne[1]

Sanparith Marukatat – Thierry Artières – Patrick Gallinari

LIP6, Université Paris 6
8 rue du Capitaine Scott, 75015 Paris

{Sanparith.Marukatat,Thierry.Artieres,Patrick.Gallinari}@lip6.fr

Résumé : *Dans ce papier, nous présentons une étude expérimentale approfondie d'une approche générique de développement de systèmes de reconnaissance de l'écriture manuscrite en ligne. Cette approche permet de concevoir des systèmes de reconnaissance de types variés correspondant à différents besoins des interfaces stylo. Nous évaluons en détail les propriétés de notre approche qui lui confèrent une grande flexibilité. Nous montrons ensuite que l'on peut concevoir aussi bien des systèmes omni-scripteur rivalisant avec les meilleurs systèmes actuels sur la base UNIPEN, et des systèmes mono-scripteur nécessitant très peu d'exemples d'apprentissage et peu gourmands en ressources machine.*
Abstract : *In this paper, we present an experimental study of a generic approach for developping on-line handwriting recognition systems. This approach allows building several types of handwriting recognition systems according to different needs arrised in pen-based interfaces. We evaluate different properties which give flexibility to our approach. We show that this approach may be used to build either writer-independent systems competing with current state-of-art systems on the UNIPEN database, or light single-writer systems able to learn with limited training samples.*

Mots-clés : Reconnaissance de l'écriture manuscrite en-ligne, Système flexible.
Keywords : On-line Handwriting Recognition System, Flexible System.

1 Introduction

Nous présentons dans ce papier une étude expérimentale approfondie d'une approche générique de développement de systèmes de reconnaissances présentée partiellement dans [ART 02, MAR 03]. Bien que nous ne nous soyons pour le moment intéressés uniquement à la reconnaissance de caractères isolés, notre approche est en principe aisée à étendre à la reconnaissance de mots car basée sur des modèles de Markov cachés comme modèles de caractères. Notre approche repose sur quelques idées relativement simples et qui conditionnent ses propriétés de flexibilité.
Tout d'abord, nous utilisons un prétraitement qui transforme un signal écrit en ligne en une représentation plus riche et plus compressée qu'une séquence temporelle de trames couramment utilisée. Par opposition à cette représentation de bas niveau, nous appelons notre représentation une représentation de "haut niveau" (ou RHN). C'est sur cette représentation que travaillent les modèles de caractères.

La seconde idée consiste à construire à partir d'une RHN, un modèle de Markov caché (MMC) travaillant sur des RHN. Ce MMC est appelé un MMC de Haut Niveau (MMCHN). Un MMCHN construit de cette façon donne naturellement de fortes vraisemblances pour les RHN proches de la RHN à partir de laquelle il a été déduit. Partant de cette idée, l'apprentissage du modèle d'un caractère est vu comme la sélection de quelques MMCHN parmi tous les MMCHN construits à partir des exemples d'apprentissage du caractère. Les MMCHN sélectionnés peuvent être vus comme des modèles correspondant aux différents allographes du caractère. L'avantage de cette méthode est que le nombre de modèles MMCHN d'un caractère, ainsi que leurs topologies sont déterminés automatiquement à partir des données d'apprentissage.

La troisième idée consiste à partager massivement les paramètres des lois de probabilités d'émission des MMCHN. Cette idée est exploitée en définissant, en plus de sa topologie, les lois de probabilité d'un MMCHN à partir d'une RHN.

Un MMCHN possède une capacité de modélisation limitée, ces modèles sont conçus de façon à bien modéliser les signaux proches des signaux utilisés pour les construire. La variabilité (intra-caractère et inter-scripteur) est prise en compte à travers le nombre de MMCHN d'un modèle de caractère. Ce nombre, -nous parlerons de la taille d'un modèle- est un hyper-paramètre du système extrêmement facile à définir et qui permet de régler la finesse de modélisation souhaitée. Pour un système omni-scripteur, la taille doit être élevée (il faut entre 20 et 50 MMCHN par caractère). Pour un système mono-scripteur, une taille égale à 5 est suffisante en général.

Après avoir brièvement présenté les composantes de notre approche, nous présentons un certain nombre de résultats expérimentaux obtenus dans des contextes variés, tant par la complexité et la nature des caractères, que par le mode d'exploitation (mono, multi et omni scripteur) ou la taille de la base d'apprentissage.

[1]Une partie des travaux présentés dans cet article a été réalisée dans le cadre d'une collaboration avec France Télécom R&D (convention 021BA40).

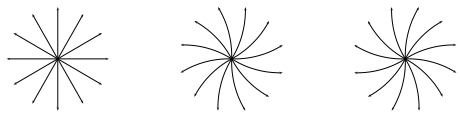


FIG. 1 – Le dictionnaire Σ des 36 modèles de tracés élémentaires utilisés dans le module de prétraitements pour décrire les signaux d'écriture manuscrite.

2 Architecture du Système

Nous présentons ici brièvement notre système, plus de détails peuvent être trouvés dans [ART 02, MAR 03]. Le système est séparé en deux niveaux, que l'on peut voir comme un module de prétraitements et un module de reconnaissance proprement dit. L'entrée du système est un signal manuscrit en-ligne (une séquence temporelle de points) acquis avec un stylo électronique ou une tablette à digitaliser. Ce signal, après avoir subi quelques prétraitements simples (lissage, normalisation en taille et ré-échantillonnage spatial), est traité par le premier module, un système markovien (§2.1). Ce module produit en résultat une représentation appelée *Représentation de Haut Niveau* ou *RHN*. Il s'agit d'une représentation séquentielle du signal d'entrée mais de plus haut niveau et plus compacte. Cette représentation est transmise en entrée du second module (§2.2). Ce système de haut niveau est également un système markovien, avec un modèle par caractère constitué de quelques MMCHN.

2.1 Prétraitements et représentation de haut niveau

La représentation de haut niveau d'un signal est une représentation séquentielle constituée de trois composantes. Une RHN est ainsi un triplet $\mathcal{R} = (S, D, RS)$. La première composante, S, est une composante de formes. Il s'agit d'une séquence de N tracés élémentaires $S = s_1, s_2, ..., s_N$ correspondant à la forme du signal, chacun des éléments de cette séquence, s_i, est une forme élémentaire. Nous utilisons un dictionnaire Σ de 36 tracés élémentaires à partir desquelles on peut représenter une grande variété de symboles écrits (voir 1). La seconde composante, D, est une composante de taille. Il s'agit d'une séquence de N durées, correspondant aux tailles respectives des tracés s_i de S. La troisième composante, RS, est une composante encodant les relations spatiales entre les différents tracés de S. La RHN d'un tracé est calculée par un système Markovien opérant sur la séquence temporelle de points représentant le signal d'entrée.

2.2 Système de Reconnaissance

La conception du système de reconnaissance repose sur l'idée que l'on peut totalement définir un MMC travaillant sur des RHN, à partir d'une RHN. La construction d'un MMCHN à partir d'une RHN est assez simple et intuitive. A partir d'une RHN $\mathcal{R} = (S, D, RS)$ de longueur N, on construit un MMCHN $\lambda_{\mathcal{R}}$ gauche-droite de N états, à chaque état e_i sont associés une forme élémentaire s_i, une durée d_i et des relations spatiales par rapport aux états précédents rs_i. Les lois de probabilités d'émission des états (définies sur Σ), les lois de durées etc, sont déterminées à partir de la RHN, de façon analogue à [MAR 03]. Nous avons choisi de partager un grand nombre des paramètres, notamment pour les lois d'émission, entre tous les états de tous les MMCHN du système. Ainsi, deux états e_i et e_j de deux MMCHN associés à un même tracé élémentaire, $s \in \Sigma$, partagent la même loi de probabilité d'émission, *i.e.* $\forall s' \in \Sigma$, $p(s'|e_i) = p(s'|e_j) = p(s'|s)$. On utilise un partage de paramètres du même type pour les composantes de durées et de relations spatiales. Cette stratégie permet de réduire de façon significative le nombre de paramètres libres du système. Notons ici que cette stratégie permet de construire de façon automatique des MMCHN (topologie comprise) à partir des signaux de la base d'apprentissage.

Le module de reconnaissance est constitué d'un modèle par caractère. Le modèle d'un caractère c est un mélange de MMCHN, noté M_c, et la probabilité d'observer une RHN $\mathcal{R}'$ est donnée par :

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

avec K_c la *taille du modèle*, *i.e.* le nombre de MMCHN pour le modèle du caractère c, $\lambda_{c,k}$ le $k$ème MMCHN du caractère c et $p(\lambda_{c,k})$ sa probabilité a priori.

L'apprentissage d'un modèle de caractère consiste à sélectionner des MMCHN représentatifs parmi tous les MMCHN construits à partir des RHN de tous les exemples d'apprentissage du caractère. Pour cela, nous réalisons un partitionnement de ces MMCHN puis on sélectionne un MMCHN par partition, celui qui représente au mieux chaque partition. Ce partitionnement est basé sur une distance entre MMCHN qui est définie en fonction de la divergence de Kullback-Leibler entre MMCHN, calculée sur l'ensemble des RHN de la base d'apprentissage. Les probabilités a priori de chaque composante $p(\lambda_{c,k})$ sont ré-estimées ensuite.

La procédure de reconnaissance est basée sur la probabilité d'observer la RHN d'un signal d'entrée par les différents modèles de caractère. En supposant que les différents caractères sont équiprobables, le caractère dont le modèle maximise la vraisemblance (équation (1)) est le caractère reconnu.

3 Évaluations

Nous avons réalisé des expériences sur des données variées. Nous étudions d'abord en détail les caractéristiques de notre approche, puis nous étudions la faisabilité de deux types de systèmes très différents. Le premier système est un système omni-scripteur pour des caractères alphanumériques usuels, il doit être capable de prendre en compte une variabilité inter-scripteurs importante. Le second est un système mono-scripteur permettant d'apprendre à partir de très peu d'exemples et consommant peu de ressources machine. Nous terminons par une comparaison, en termes de ressources machine nécessaires entre notre système et des systèmes existants.

3.1 Bases de données

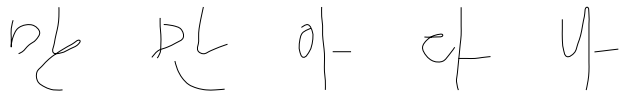
Nous présentons maintenant les bases de données utilisées dans nos expériences.

3.1.1 Chiffres, caractères minuscules et majuscules (base UNIPEN)

La base UNIPEN est la base de référence pour l'élaboration et la comparaison de systèmes de reconnaissance d'écriture. Nous utilisons ici la partie de cette base contenant des caractères isolés, lettres minuscules et majuscules, chiffres. Cette base contient les tracés de plus de 200 scripteurs. La difficulté de cette base est due principalement au nombre de scripteurs et donc aux nombreux allographes qu'ils emploient. Dans ces expériences, nous disposons de la version R01/V06, le nombre total d'exemples de chiffres (resp. de lettres majuscules et minuscules) est de 15719 (resp. 22683 et 59691).

3.1.2 Chiffres et caractères coréens (base KAIST)

La base KAIST contient des signaux d'écriture en-ligne écrits par des collégiens et des lycéens coréens. Nous n'avons travaillé que sur une partie de cette base contenant des chiffres et des caractères coréens provenant d'une vingtaine de scripteurs.



(a) (b) (c) (d) (e)

FIG. 2 – Variabilité et complexité de l'écriture coréenne : deux styles d'écriture différents pour un même caractère coréen ((a) et (b)) ; trois caractères très similaires (c), (d) et (e).

La reconnaissance des caractères coréens est difficile car d'une part les strokes naturels (entre deux levées de stylo) peuvent être assez complexes, d'autre part ces caractères sont écrits avec plusieurs levées de stylo ou avec des ligatures. De plus, les emplacements relatifs des composants d'un caractère les uns par rapport aux autres sont essentiels à la définition du caractère. Pour illustrer ces difficultés, les figures (2 (a) et (b)) montrent différents styles d'écriture d'un même caractère et les figures (2 (c), (d) et (e)) montrent des caractères distincts mais très semblables. Nos expériences portent sur un ensemble de 19 caractères coréens.

3.1.3 Symboles (base LIP6)

Cette base de symboles, collectée au LIP6, a pour but d'évaluer certaines caractéristiques du système plus précisément (voir figure 3). Notamment, certains symboles partagent les mêmes traits et ne se différencient que par les positions relatives des différents traits. Nous utilisons 32 symboles au total. Il faut noter ici que ces symboles n'étant pas tous usuels, différents scripteurs ne les écrivent pas avec le même ordre de tracé (par exemple pour les différentes branches de l'étoile), et en réalité un même scripteur n'écrit pas systématiquement dans le même ordre les composants d'un même symbole. Trois scripteurs ont écrit les symboles de cette base. Chaque scripteur a écrit 20 exemples pour chaque symbole.

3.2 Caractéristiques de l'approche

L'objet des expériences suivantes est de montrer le comportement de nos systèmes vis à vis de certaines conditions

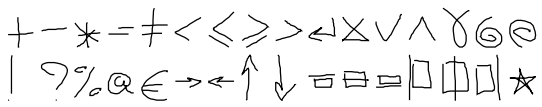
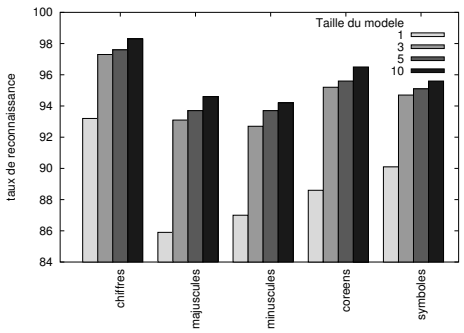
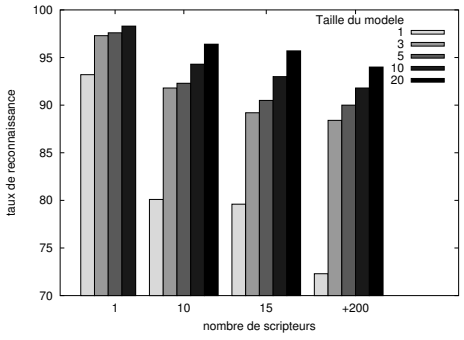


FIG. 3 – Les symboles de la base collectée au LIP6.



(a)



(b)

FIG. 4 – Prise en compte de la variabilité des signaux par l'augmentation de la taille du modèle : (a) variabilité dans les tracés de caractères et (b) variabilité inter-scripteur.

expérimentales, variabilité dans les tracés, variabilité inter-scripteur, taille de la base d'apprentissage, en faisant des expériences ciblées dans lesquelles une seule de ces conditions varie.

3.2.1 Variabilité dans les tracés

Les premières expériences concernent la capacité à apprendre n'importe quel type de caractère. Pour cela, nous avons testé notre système sur différents types de caractère, en mode mono-scripteur. Nous réalisons deux validations croisées pour chaque scripteur et fournissons des résultats moyens. Nous avons réalisé ces expériences sur des signaux de scripteurs ayant écrit au moins 10 exemples pour chaque caractère. Les expériences sont donc réalisées avec des signaux de 14 scripteurs pour les chiffres (provenant de la base UNIPEN et de la base KAIST), de 5 scripteurs pour les lettres minuscules, de 7 scripteurs pour les majuscules, de 5 scripteurs pour les caractères coréens et de 3 scripteurs pour les symboles de la base LIP6.
Les résultats obtenus sont résumés dans la figure 4 (a). Il n'est pas immédiat de comparer les performances sur les différentes types de caractères, puisqu'il ne s'agit pas des mêmes scripteurs, que les caractères sont plus ou moins

similaires suivant les bases de données et que le nombre d'exemples d'apprentissage par caractère varie suivant les bases (au minimum il y a 5 exemples d'apprentissage par caractère). Néanmoins, on remarque l'augmentation systématique de la performance lorsque la taille des modèles de caractères croit, permettant d'atteindre de l'ordre de 95% de reconnaissance pour tous ces types de caractère. La taille des modèles, un hyper-paramètre de l'apprentissage, apparaît comme un moyen simple de régler la qualité du système obtenu. Cela signifie qu'il est aisé d'adapter la complexité du système en fonction de la performance souhaitée et des caractères traités. Rappelons ici que les modèles sont totalement appris à partir des données (topologie comprise) ce qui montre que notre approche peut être utilisée pour n'importe quel type de caractère graphique sans introduction de connaissance a priori.

3.2.2 Variabilité inter-scripteur

Afin d'étudier la robustesse à la variabilité inter-scripteurs, nous avons comparé les performances en reconnaissance de chiffres sur la base UNIPEN en faisant varier le nombre de scripteurs. La figure 4 (b) montre les taux de reconnaissance obtenus en mode mono-scripteur et en mode multi-scripteur avec 10, 15 ou plus de 200 scripteurs. Pour chaque cas, nous donnons des résultats obtenus pour différentes tailles de modèle. On voit qu'en augmentant le nombre de scripteurs, donc la variabilité dans les signaux, la tâche de reconnaissance devient plus difficile et on obtient de moins bons taux de reconnaissance, à taille de modèle égale. Là encore, l'augmentation de la taille du modèle permet de prendre en compte la variabilité accrue et d'obtenir, même avec un grand nombre de scripteurs, un taux de reconnaissance de l'ordre de 94%. Bien entendu, ce taux reste inférieur aux taux obtenus en mono-scripteur. Cependant, avec l'augmentation de la taille des modèles, on diminue l'écart entre les performances obtenues en mono-scripteur et en mode multi-scripteur. Nous verrons par ailleurs au §3.3 que l'on peut obtenir des performances de l'ordre de 98% sur cette base en apprenant sur tous les scripteurs de la base et avec une taille de modèle égale à 50.

3.2.3 Taille de la base d'apprentissage

L'architecture de notre système permet également d'apprendre des modèles de caractères à partir de très peu d'exemples d'apprentissage. Pour mettre en évidence cette propriété, nous avons étudié les performances obtenues par notre système en faisant varier la taille de la base d'apprentissage et cela pour différentes tailles de modèles. Les expériences présentées ici ont été réalisées sur deux bases de données, des caractères simples (chiffres de la base KAIST), et des caractères complexes (caractères coréens), en mode mono-scripteur. Pour éviter le biais introduit par le choix des données d'apprentissage (en quantité limitée) les résultats fournis sont moyennés sur plusieurs expériences correspondant à plusieurs tirages aléatoires des exemples pris en apprentissage. Les figures 5 (a) et (b) montrent les performances moyennes (sur l'ensemble de scripteurs) en fonction du nombre d'exemples d'apprentissage par caractère en apprentissage ainsi qu'en fonction de la taille maximale du modèle. D'après ces résultats, on voit qu'avec 5 exemples

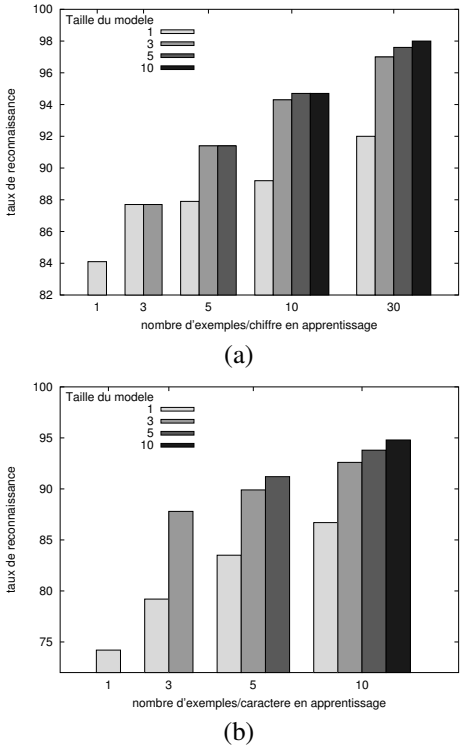


FIG. 5 – Performances en mode mono-scripteur en fonction du nombre d'exemples par caractère en apprentissage et de la taille des modèles de caractères : pour les chiffres (a) et pour les caractères coréens (b).

d'apprentissage on obtient de l'ordre de 90% de reconnaissance, et avec 10 exemples d'apprentissage par caractère on atteint des taux de reconnaissance de l'ordre de 95%. Même si les caractères coréens sont plus complexes que les chiffres, un peu plus d'exemples d'apprentissage et une taille de modèle légèrement supérieure suffisent à obtenir des taux de reconnaissance équivalents. Notons également ici qu'un seul exemple en apprentissage permet déjà d'obtenir de taux de reconnaissance non négligeable.

3.3 Conception de systèmes

Les propriétés de notre approche que les expériences précédentes ont mis en évidence permettent de développer des systèmes très différents, correspondant à des contextes d'utilisation variés. Nous discutons par exemple dans la suite de la conception de systèmes mono-scripteurs adaptés à des terminaux mobiles légers et de la conception d'un système plus lourd, fonctionnant en mode omni-scripteur pour des caractères alphanumériques usuels.

3.3.1 Systèmes mono-scripteur pour terminaux mobiles

Les résultats précédents ont montré que notre approche permet de concevoir des systèmes de reconnaissance mono-scripteur ayant des taux de reconnaissance élevés (figure 4 (a)), tout à fait acceptables d'un point de vue utilisateur (de l'ordre de 95%). Par ailleurs ces taux de reconnaissance sont obtenus avec une quantité très limitée de données d'apprentissage et pour tous les types de caractères envisagés. Cela signifie d'une part qu'un utilisateur peut utiliser un système de ce type sans avoir à passer par une étape fastidieuse de

collection de données d'apprentissage, d'autre part que l'utilisateur peut lui-même définir des symboles, abréviations, ou gestes de commande qui lui sont propres. Or cet aspect de personnalisation et d'appropriation de l'interface par l'utilisateur est considéré, du point de vue des usages, comme une propriété essentielle conditionnant l'acceptation des interface stylo pour les objets mobiles [LON 99]. Nous verrons enfin au §3.4 que les ressources machine d'un système mono-scripteur (nécessitant une taille de modèle de l'ordre de 5) sont très minimes et parfaitement compatibles avec des terminaux légers.

3.3.2 Systèmes Multi et Omni-scripteur pour les caractères alphanumériques

Nous étudions ici la faisabilité, avec notre approche, d'un système multi ou omni-scripteur pour des caractères usuels. Nous évaluons notre système sur la base UNIPEN. Le tableau 1 montre les résultats de ces évaluations en comparaison avec les résultats d'autres systèmes (notamment ceux rapportés dans [RAT 03]). Ces résultats sont ordonnés selon le pourcentage de la base totale utilisée en apprentissage (5%, 20% et 50%). D'après ces résultats, on voit que notre système atteint des performances comparables aux meilleurs classifieurs existants en multi scripteur et en omni-scripteur. Nous remarquons aussi que même avec une base de données de taille restreinte (*i.e.* 5%), notre système obtient déjà des bons résultats.
Par ailleurs, notre approche est, contrairement aux approches qui sont comparées ici, extensible à la reconnaissance de mots et est d'une certaine façon réglable via l'hyperparamètre de l'apprentissage, la taille des modèles de caractères.

3.4 Implémentation

Les résultats expérimentaux montrent que nos systèmes atteignent des performances au niveau des systèmes qui sont à l'état de l'art aujourd'hui. Nous montrons maintenant que ces systèmes sont assez économes en ressources machine, processeur et mémoire.

3.4.1 Taille mémoire

Nous comparons ici certains des systèmes dont les performances sont rapportées dans le tableau 1, du point de vue de la taille mémoire nécessaire, en nombre de paramètres. Les systèmes à base de MVS, *e.g.* [BAH 02, VUU 00], nécessitent une mémoire importante pour stocker les vecteurs supports. Dans [BAH 02] environ 100 vecteurs supports par classe sont nécessaire pour obtenir les performances du tableau 1, ce qui nécessite, selon les auteurs, environ 17.5 Mo pour la classification des 26 lettres minuscules (et environ 6.7 Mo pour les 10 chiffres). Dans [VUU 00], 21 Mo sont nécessaires pour le stockage des vecteurs supports. Le classifieur sn-tuple [RAT 03] est en théorie encore plus gourmand. Toutefois, en se limitant aux tuples réellement observés, l'auteur de ces travaux a ramené la taille mémoire nécessaire à environ 19.3 Mo pour la classification des chiffres [RAT 03].
En comparaison, notre système nécessite moins de ressources mémoires. Un système de reconnaissance de 26 lettres, avec des modèles de caractères de taille 5 (resp. 50)

%UNIPEN en apprentissage	Système	chiffres		majuscules		minuscules	
		multi	omni	multi	omni	multi	omni
5	MMCHN						
	taille de modèle=5	92.4	87.5	85.1	80.2	80.9	78.5
	taille de modèle=10	94.1	89.9	86.9	82.7	83.5	80.3
	taille de modèle=20	95.1	90.7	87.9	83.8	85.0	81.7
	taille de modèle=50	95.4	91.9	89.1	84.9	85.9	82.6
20-30	MMCHN						
	taille de modèle=5	92.3	90.3	84.7	85.6	81.7	80.3
	taille de modèle=10	95.1	91.9	86.9	87.8	84.1	82.8
	taille de modèle=20	96.4	93.4	88.6	89.1	86.2	84.5
	taille de modèle=50	97.1	94.9	89.2	89.5	87.9	85.9
	sn-tuple [RAT 03]	98.7	97.1	93.7	91.8	90.6	87.8
	MMC [BAH 01]	95.5		90.0		88.6	
	MVS [BAH 02]	96.0		92.4		87.9	
	MMC [LI 98]	92.0					
	PMC [LEM 02]		95.6				
50-60	MMCHN						
	taille de modèle=5	93.5	90.7	88.3	87.0	80.9	80.8
	taille de modèle=10	95.2	92.3	90.4	89.3	84.3	83.7
	taille de modèle=20	96.6	94.5	91.9	90.2	86.7	85.7
	taille de modèle=50	97.7	95.9	92.8	91.2	88.2	87.0
	sn-tuple [RAT 03]	98.8	98.1	95.1	94.0	92.0	
	k-ppv [PRE 00]		98.8		96.6		96.3
	MMC Segmental [ART 00]					79.5	69.8
	RB [CHO 99]		95.0				
	Réseau ART [SAN 98]	82.4					

TAB. 1 – Évaluation de notre approche pour les signaux de la base Unipen, en comparaison avec d'autres systèmes.

référence	nombre de caractères	temps de reconnaissance	machine
sn-tuple [RAT 03]	10 (chiffres) 26 (lettres)	420 420	300 MHz RS6000 workstation
MVS noyau alignement temporel [BAH 02]		0.4	AMD Athlon 1200 MHz
RB [CHO 99]		84	IBM ThinkPad T23, 1.13 GHZ
Réseau ART et logique floue [SAN 98]		3.57	Pentium 120MHz
MMC [HU 00]		3.3	180 MHz SGI station
k-ppv [VUU 00] avec MVS post-traitement		2.88	Pentium-II 400 MHz
MMCHN avec une taille de modèle = 50	10 (chiffres) 26 (lettres)	74 20	Pentium-III 500 MHz
MMCHN avec une taille de modèle = 5	10 (chiffres) 26 (lettres)	720 230	Pentium-III 500 MHz

TAB. 2 – Comparaison des vitesses de reconnaissance de différents systèmes de reconnaissance de caractères.

nécessite environ 130 Ko (resp. 1 Mo) de mémoire pour stocker le système (les MMCHN ainsi que les lois de probabilité).

3.4.2 Vitesse de reconnaissance

En ce qui concerne la rapidité de reconnaissance, liée à la complexité algorithmique de la procédure de reconnaissance, la comparaison entre les différents systèmes n'est pas triviale car les évaluations ont été faites sur différentes machines. Le tableau 2 rapporte les vitesses de reconnaissance (en nombre de caractères par seconde) pour certains des systèmes cités dans le tableau 1. On voit qu'en comparaison avec d'autres systèmes, surtout ceux à base de prototypes *e.g.* [VUU 00], nos systèmes sont plus rapides. Notons ici que le calcul d'une RHN est de l'ordre de la milliseconde, ce qui est négligeable par rapport au temps de reconnaissance.

4 Conclusions

Nous avons réalisé un certain nombre d'expérimentations visant à étudier de façon approfondie les caractéristiques de notre approche pour la reconnaissance de signaux écrits en ligne et à la comparer aux systèmes actuels. Notre approche est basée sur quelques idées simples et possède un certain nombre de bonnes propriétés. Nous avons exploré sa capacité à traiter n'importe quel type de primitive graphique simple, sa capacité à prendre en compte automatiquement la variabilité dans les signaux, et la possibilité d'apprendre un caractère, ou symbole à partir de très peu de données d'apprentissage. Ces propriétés font qu'il est possible de développer aussi bien des systèmes mono-scripteur avec lesquels l'utilisateur peut lui-même définir ses propres caractères ou symboles, que des systèmes omni-scripteurs au niveau des meilleurs systèmes actuels. Par ailleurs, les systèmes développés se situent très bien du point de vue des ressources machine nécessaires, à la fois en mémoire et en processeur. Par exemple, dans un contexte mono-scripteur, un système léger (e.g. avec une taille du modèle égale à 5) suffit à donner de bons résultats. Ce système, nécessitant moins de 130 Ko, peut reconnaître jusqu'à 230 caractères par seconde. Pour un contexte omni-scripteur, un système plus lourd (e.g. avec une taille de modèle égale à 50) est nécessaire. Ce type de système nécessite environ 1 Mo de mémoire et peut reconnaître jusqu'à 20 caractères par seconde.

Références

[ART 00] ARTIÈRES T., MARCHAND J.-M., GALLINARI P., DORIZZI B., Stroke Level Modelling of On line Handwriting through Multi-modal Segmental Models, *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2000.

[ART 02] ARTIÈRES T., GALLINARI P., Stroke level HMMs for on-line handwriting recognition, *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2002.

[BAH 01] BAHLMANN C., BURKHARDT H., Measuring HMM Similarity with the Bayes Probability of Error and its Application to Online Handwriting Recognition, *Proc. of the 6th ICDAR*, 2001, pp. 406–411.

[BAH 02] BAHLMANN C., HAASDONK B., BURKHARDT H., On-line Handwriting Recognition with Support Vector Machines – A Kernel Approch, *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, 2002, pp. 49-54.

[CHO 99] CHO S. J., KIM J. H., Bayesian Network Modeling of Strokes and Their Relationships for On-line Handwriting Recognition, *International Conference on Document Analysis and Recognition (ICDAR)*, 1999.

[HU 00] HU J., LIM S. G., BROWN M. K., Writer independent on-line handwriting recognition using an HMM approach, *Pattern Recognition*, vol. 33, n° 1, 2000, pp. 133-148.

[LEM 02] LEMIEUX A., GAGNÉ C., PARIZEAU M., Genetical Engineering of Handwriting Representations, *International Workshop on Frontiers of Handwriting Recognition (IWFHR) 2002*, Niagara-on-the-Lake, ON, Canada, 2002, pp. 145–150.

[LI 98] LI X., PLAMONDON R., PARIZEAU M., Model-based On-line Handwritten Digit Recognition, *14th Int. Conf. Pattern Recognition*, Brisbane, Australia, August 1998, pp. 1134-1136.

[LON 99] LONG A. C., LANDAY J. A., ROWE L. A., Implications for a Gesture Design Tool, *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI)*, 1999.

[MAR 03] MARUKATAT S., SICARD R., ARTIÈRES T., GALLINARI P., A flexible recognition engine for complex on-line handwriting character recognition, *International Conference on Document Analysis and Recognition (ICDAR)*, 2003.

[PRE 00] PREVOST L., MILGRAM M., Modelizing character allographs in omni-scriptor frame : a new non-supervised clustering algorithm, *Pattern Recognition*, vol. 21, 2000, pp. 295-302.

[RAT 03] RATZLAFF E. H., Methods, Report and Survey for the Comparison of Diverse Isolated Character Recognition Results on the UNIPEN Database, *International Conference on Document Analysis and Recognition (ICDAR)*, 2003.

[SAN 98] SÁNCHEZ E. G., GONZÁLEZ J. G., DIMITRIADIS Y., Experimental study of a novel neuro-fuzzy system for on-line handwritten UNIPEN digit recognition, *Pattern Recognition Letters*, vol. 19, 1998, pp. 357-364.

[VUU 00] VUURPIJL L., SCHOMAKER L., Two-Stage Character Classification : A Combined Approach of Clustering and support Vector Classifiers, SCHOMAKER L., VUURPIJL L., Eds., *International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, Amsterdam, 2000, pp. 423-432.