Une méthode générique pour la conception de moteurs de reconnaissance de symboles manuscrits en ligne[1,2]

A generic approach for desining on-line handwritten shapes recognizers

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

Manuscrit reçu le 19 mai 2004

Résumé et mots clés

Dans ce papier, nous présentons une approche générique pour le développement de moteurs de reconnaissance de symboles manuscrits en ligne. Cette approche permet de concevoir des systèmes de reconnaissance de types très variés correspondant à différents contextes des interfaces stylo, pouvant notamment fonctionner sur diverses classes de caractères ou symboles. Nous présentons en détail notre approche et faisons le lien avec d'une part les modèles de Markov hiérarchiques et d'autre part les réseaux bayésiens dynamiques. Nous évaluons ensuite les propriétés fondamentales de notre approche qui lui confèrent une grande flexibilité. Puis nous montrons que l'on peut, avec cette approche générique, concevoir aussi bien des systèmes omni-scripteur rivalisant avec les meilleurs systèmes actuels sur des caractères alphanumériques usuels, que des systèmes mono-scripteur pour des symboles graphiques quelconques, nécessitant très peu d'exemples d'apprentissage et peu gourmands en ressources machine.

Réseaux Bayésiens dynamiques, modèles de Markov, écriture en ligne, reconnaissance de symboles, reconnaissance de gestes graphiques, généricité

Abstract and key words

This paper presents a generic approach for designing on-line handwritten shapes recognizers. Our approach allows designing very different recognition engines that correspond to various needs in pen-based interfaces. In particular, it allows dealing with a wide class of symbols and characters. We present in detail our system and make the link between our models and more standard statistical models such as Hierarchical Hidden Markov Models and Dynamic Bayesian Networks. We then evaluate fundamental properties of our approach: learning from scratch any symbol, learning from very few training sample. We show experimentally that, using our approach, one can learn both a state-of-the-art writer-independent recognizer for alphanumeric characters, and a writer-dependent recognizer working with any two-dimensional shapes that learns a new symbol with a few training samples and requires very few machines resources.

Dynamic Bayesian Networks, Hidden Markov Models, On-line handwriting, Symbol recognition, Graphical gesture recognition, Genericity

---

[1] Ce travail a été réalisé en partie dans le cadre d'une collaboration avec France Télécom R&D (n°021BA40).

[2] Une version de démonstration de ce système, *SYMBOL*, est disponible à http://www-connex.lip6.fr dans la rubrique *DEMOS*.

1. Introduction

Nous présentons ici une approche générique de développement de systèmes de reconnaissances pour des signaux manuscrits en ligne. Bien que nous ne nous soyons pour le moment intéressés qu'à la reconnaissance de caractères isolés, notre approche est, par construction, aisée à étendre à la reconnaissance de mots car basés 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 complexe qui transforme un signal écrit en ligne en une représentation plus riche et plus compressée qu'une séquence temporelle de points ou de trames couramment utilisées en écrit. 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 et leurs paramètres 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 de 5 à 10 est suffisante en général.

Nous présentons tout d'abord au §2 les différents aspects de notre approche, le système de bas niveau et le système de haut niveau puis nous faisons le lien au §3 entre nos modèles et d'autres modèles utilisés en apprentissage statistique. Nous présentons ensuite au §4 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.

2. Description de l'approche

Nous présentons ici notre système, celui-ci est structuré en deux niveaux, que l'on peut voir comme un module de prétraitements et un module de reconnaissance proprement dit, nous donnerons une interprétation formelle plus fine au §3. 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.

2.1. Prétraitements et représentation de haut niveau

La représentation de haut niveau (*RHN*) d'un signal est un quadruplet $R = (N, S, D, B)$. N est la longueur de la représentation. La première composante, S, est une composante de formes. Il s'agit d'une séquence de N tracés élémentaires (ou strokes) $S = s_1, s_2, ..., s_N$ correspondant à la forme du signal, chacun des éléments de cette séquence, si, est une forme ou tracé élémentaire appartenant à un dictionnaire. Nous utilisons un dictionnaire Σ de 36 tracés élémentaires à partir desquels on peut représenter une grande variété de tracés en deux dimensions (voir figure 1). La seconde composante, D, est une composante de taille ou de durée. Il s'agit d'une séquence de N valeurs, correspondant aux durées respectives des tracés si de S. La troisième composante, B, est constituée des positions spatiales des strokes de S, nous utilisons ici les coordonnées des centres des tracés s_i de S, dans la boite englobante normalisée du tracé total (*i.e.* de largeur et de hauteur égales à 1).

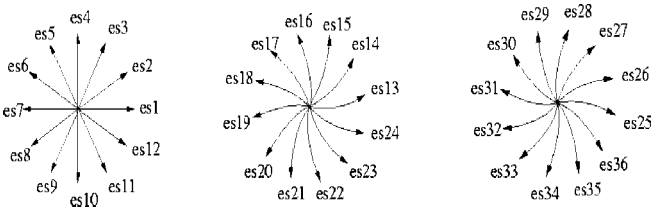

Figure 1. Le dictionnaire Σ des 36 modèles de tracés élémentaires utilisés dans le module de prétraitements pour décrire les signaux d'écriture manuscrite, 12 tracés droits de directions uniformément réparties entre 0 et 360°, 12 tracés convexes et 12 tracés concaves.

Divers travaux ont exploité la décomposition des signaux d'écriture en tracés élémentaires. Ce sont essentiellement des travaux sur des alphabets asiatiques, par nature plus graphiques [Cho99, Lee00, Liu01, Nak01]. Dans ces études, seuls des tracés élémentaires linéaires (*i.e.* des segments de droite) sont considérés. Dans notre cas, nous utilisons des modèles segmentaux pour implémenter des modèles de tracés élémentaires non linéaires, *i.e.* des morceaux de courbes décrits dans la Figure 1, permettant de représenter finement une plus grande variété de symboles et caractères. La *RHN* d'un signal manuscrit (séquence temporelle de points) est obtenue par programmation dynamique dans le système de bas niveau, un *MMC* ergodique à 36 états, un par tracé élémentaire. Un modèle de trajectoire segmental est utilisé dans chaque état pour implémenter une loi de probabilité sur des séquences [Art02]. Un *MMC* segmental est une extension d'un *MMC* standard dans lequel un état n'émet pas une série d'observations successives indépendantes les unes des autres mais une série d'observations dont la séquence obéit à une certaine dynamique linéaire ou non linéaire [Den94]. En effectuant, pour un signal manuscrit observé, une passe de programmation dynamique dans ce *MMC* ergodique, on récupère le chemin optimal, et donc la séquence de tracés élémentaires *S* la plus vraisemblable, *i.e.* représentant au mieux le tracé originel. Les durées *D* sont également récupérées à partir du chemin optimal dans ce *MMC*, en comptant la durée passée dans chacun des états du chemin optimal. Les boites englobantes, *B*, des tracés élémentaires de *S*, sont également calculées *a posteriori* à partir de la segmentation.

La Figure 2 illustre le calcul d'une *RHN* pour un tracé correspondant au chiffre 7. Le décodage de ce tracé dans le système de bas niveau conduit à une *RHN* de longueur $N = 3$ où $S = (es_1, es_9, es_1)$ et $D = (0.3, 0.6, 0.1)$. Le premier stroke à une taille correspondant à 30 % du tracé total, le second 60 % et le troisième 10 %. L'information spatiale, non représentée sur la figure, est constituée de la séquence des coordonnées des centres des trois tracés de S.

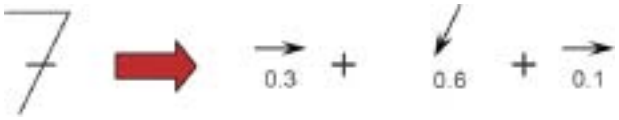

Figure 2. Exemple de RHN : Le signal à gauche est représenté par une séquence de 3 tracés élémentaires, es_1, es_9 et es_1, de durées respectives 0.3, 0.6 et 0.1. Les durées sont des durées relatives des tracés élémentaires par rapport à la durée totale du tracé qui vaut 1. Dans cet exemple, le premier tracé élémentaire représente 30% du tracé total et le second 60%.

2.2. Système de Reconnaissance

Le système de reconnaissance est constitué d'un modèle par caractère. Le modèle d'un caractère *c*, noté M_c, est un mélange de modèles de Markov travaillant sur des *RHN* (nous appelons un *MMC* de ce type un *MMC* de Haut Niveau ou *MMCHN*) . La probabilité d'observer un tracé manuscrit *X* correspondant au caractère *c* est calculée par la vraisemblance de la *RHN* du tracé par le modèle du caractère *c*, M_c :

$$P(X/c) = P(R/M_c) = \sum_{k=1}^{K_c} P(R/\lambda_{c,k}).P(\lambda_{c,k}) \qquad (1)$$

où *R* est le résultat du prétraitement de *X*, c'est-à-dire la *RHN* du signal *X* calculée par le système Markovien de bas niveau détaillé au §2.1. K_c est le nombre de *MMCHN* pour le modèle du caractère *c* (que nous appelons dans la suite la taille du modèle), $\lambda_{c,k}$ est le $k^{\text{ième}}$ *MMCHN* du caractère *c*. Nous notons ici $p(\lambda_{c,k})$ la probabilité *a priori* qu'un signal, ou plutôt sa *RHN*, soit produite par $\lambda_{c,k}$. Dans ce type de modèle de mélange, la variabilité dans les tracés d'un caractère est prise en compte par la multiplication de *MMCHN*.

La procédure de reconnaissance pour un signal manuscrit *X* est basée sur la probabilité d'observer la *RHN* de ce 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 est le caractère reconnu.

2.3. Apprentissage du système

L'apprentissage du système consiste à apprendre les modèles de caractères, la particularité de notre approche réside dans l'apprentissage complet de ces modèles (*i.e.* topologie et paramètres de modèles de Markov dans l'équation (1)) à partir de données d'apprentissage. Cela permet, comme nous le verrons dans la partie expérimentale, de concevoir des systèmes de reconnaissance pour des alphabets de caractères quelconques (symboles, caractères coréens, dessins etc) sans avoir à introduire manuellement (par exemple par essais successifs de topologie des modèles Markoviens) une information *a priori* dans le système. Cela signifie que l'apprentissage d'un système pour un alphabet de caractères peut être réalisé totalement automatiquement à partir d'une base d'apprentissage sans une phase de mise au point manuelle longue et coûteuse.

La conception du système de reconnaissance proprement dit, que nous appelons le système de haut niveau, 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. Nous décrirons en détail cette procédure ainsi que la loi de probabilité implémentée par un *MMC*NH au §2.4 et ne présentons ici que les idées principales. La Figure 3 illustre la construction d'un *MMCHN* à partir d'une *RHN* en reprenant l'exemple de la Figure 2. À partir d'un tracé, on calcule sa *RHN*, puis on construit un *MMC* gauche droite à partir de cette *RHN*. A partir d'une *RHN* $R = (N, S, D, B)$, on construit un *MMC* gauche-droite à *N* états, λ_R. Cette construction associe naturellement à chaque état e_i du *MMCHN* une forme élémentaire si, une durée d_i et une position

spatiale p_i. Les lois de probabilité associées aux états seront comme nous le verrons par la suite totalement définies à partir de cette association. Par exemple, pour le *MMCHN* de la Figure 3, la loi de probabilité d'émission associée au premier état, qui est définie sur Σ, ne donne de fortes probabilités que pour des tracés élémentaires proches du tracé « idéal » associé à cet état, es_1. Par ailleurs, nous utilisons tant que possible une stratégie de partage de paramètres, ainsi les lois de probabilité sur les tracés élémentaires associées sont identiques dans le premier et le troisième état dans l'exemple de la Figure 3.

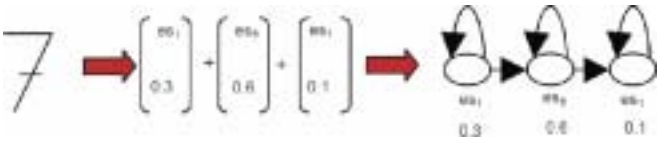

Figure 3. Principe de construction d'un MMCHN à partir d'un tracé (à gauche), via sa représentation de haut niveau (au centre). Le MMCHN a autant d'états qu'il y a de tracés élémentaires dans la RHN. Les paramètres du MMCHN sont déduits également de la RHN (voir texte) à partir de l'association des états du MMCHN et des composantes de la RHN.

Un *MMCHN* construit de cette façon à partir d'un exemple de tracé donne, par construction, de fortes vraisemblances pour les *RHN* proches de la *RHN* à partir de laquelle il a été déduit mais n'accepte pas beaucoup de variabilité autour de ce tracé originel. La variabilité dans les tracés d'un caractère n'est donc pas prise en compte par un *MMCHN* mais peut l'être par la multiplication de *MMCHN* dans un modèle de caractère. C'est pourquoi nous utilisons des modèles de caractères qui sont des modèles de mélange (*cf.* eq (1)).

L'approche classique pour apprendre un modèle tel que celui de l'équation (1) consisterait à fixer manuellement le nombre et la topologie des modèles de Markov du mélange, puis à apprendre les paramètres de ces modèles sur une base d'apprentissage à l'aide d'un critère du type Maximum de Vraisemblance. Cette approche nécessite d'une part une information *a priori* sur les caractères traités pour fixer raisonnablement la topologie des modèles, d'autre part un certain nombre d'essais expérimentaux (sur la topologie) pour arriver à des résultats performants. Afin d'apprendre complètement (topologie et paramètres) les modèles de caractères à partir de données d'apprentissage, nous avons exploité la procédure décrite précédemment (construction d'un *MMCHN* à partir d'une *RHN*) pour ramener l'apprentissage d'un modèle de caractère à un problème de sélection des *MMCHN* les plus représentatifs parmi tous les *MMCHN* construits à partir des *RHN* de tous les exemples d'apprentissage de ce caractère.

Ainsi, pour apprendre le modèle d'un caractère donné, on commence par calculer les *RHN* de tous les tracés d'apprentissage du caractère puis on construit tous les *MMCHN* correspondant. Ensuite, on réalise un partitionnement de l'ensemble de ces *MMCHN* à l'aide d'un algorithme de clustering du type K-Moyennes dans l'espace des modèles (*i.e.* en définissant une distance entre *MMCHN*) [Mar04-a]. Ensuite, on sélectionne un *MMCHN* par partition, celui qui représente au mieux chaque partition. Cette procédure, dont le seul paramètre est le nombre de partitions, K_c, a pour résultat l'ensemble des K_c *MMCHN* $(\lambda_{c,1},\lambda_{c,2},...,\lambda_{c,kc})$ du modèle du caractère de l'équation (1). Les probabilités *a priori* de chacune de ces composantes $p(\lambda_{c,k})$ sont ré-estimées ensuite avec un critère de Maximum de Vraisemblance.

Les *MMCHN* sélectionnés de cette façon peuvent être vus comme des modèles des différents allographes du caractère. L'avantage de cette méthode est que le nombre de modèles *MMCHN* d'un caractère, ainsi que leurs topologies et leurs paramètres sont déterminés automatiquement à partir des données d'apprentissage. À titre d'illustration, la figure 4 montre les 20 tracés du chiffre 7 de la base Unipen correspondant aux 20 *MMCHN* les plus significatifs, sélectionnés par la phase d'apprentissage sur tous les exemples de ce chiffre dans la base Unipen.

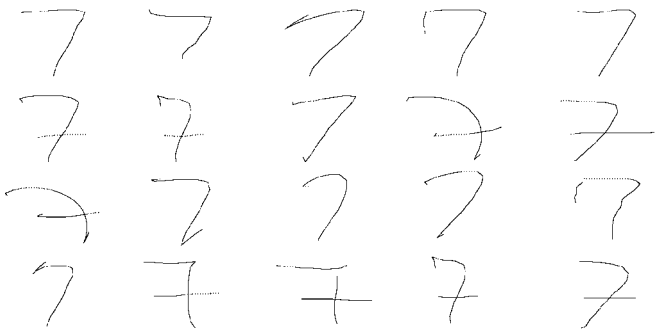

Figure 4. Tracés des exemples du chiffre 7 correspondant à 20 MMCHN sélectionnés par l'algorithme d'apprentissage sur la base Unipen.

Comme nous le verrons dans la suite, le système est défini à partir de quelques hyper-paramètres, la taille des modèles de caractères notamment ainsi que quelques hyper-paramètres à partir desquels sont définies les lois de probabilités implémentées par les modèles de caractères. Tous ces paramètres ont été appris une fois pour toutes avec un critère de type Maximum de Vraisemblance sur un sous-ensemble des données utilisées dans la partie expérimentale. Ils ne sont pas raffinés suivant les expériences pour coller à tel ou tel type de tâche de reconnaissance.

2.4. Formalisation des MMCHN

Nous décrivons maintenant plus en détail le calcul de la probabilité d'une *RHN* observée, de longueur T, $R' = (T, S', D', B')$ par un modèle *MMCHN* (de structure gauche droite) λ, construit à partir d'une *RHN* $R = (N, S, D, B)$. Cette vraisem-

blance est calculée par :

$$P(R'/\lambda) = \sum_{Q} P(R'/Q,\lambda).P(Q/\lambda) \quad (2)$$

où $Q = q_1^T = (q_1,q_2,...,q_T)$ est la segmentation de R' dans λ (*i.e.* une séquence d'états de λ). Pour intégrer explicitement les informations de forme, de durées et de relations spatiales, nous faisons une hypothèse simplificatrice d'indépendance des trois flux d'information conditionnellement à la segmentation. Ainsi, nous considérons séparément ces différents flux d'information en définissant la vraisemblance le long d'une segmentation par :

$$P(R'/q_1^T,\lambda) = P_{forme}(R'/q_1^T,\lambda).P_{durée}(R'/q_1^T,\lambda) .P_{spatial}(R'/q_1^T,\lambda) \quad (3)$$

où $P_{forme}(R'/q1T,\lambda)$ est la probabilité de la composante de forme S' du signal d'entrée, $P_{durée}(R'/q1T,\lambda)$ est la probabilité de la composante de durée D' du signal d'entrée, et $P_{spatial}(R'/q1T,\lambda)$ représente la probabilité des positions des différents strokes de S'.

Avant d'aller plus loin, rappelons que dans des *MMC* gauche droite comme ceux que nous utilisons, chaque état modélise une partie du tracé (*e.g.* le début, le milieu, la fin). Il est alors naturel de définir des lois de probabilités sur les segments de R' associés à chacun des états. Avec N le nombre d'états de λ, e_i son $i^{\text{ème}}$ état, et en exploitant la topologie gauche droite du *MMCHN* λ, notons b_i et l_i les instants où l'on entre et où l'on sort de e_i le long de la segmentation q_1^T, c'est-à-dire :

$$\begin{cases} b_1 = 1 \\ l_N = T \\ \forall i = 1 \dots N,\ \forall t \in [b_i,l_i],\ q_t = e_i \\ \forall i = 1 \dots N,\ b_{i+1} = l_i + 1 \end{cases} \quad (4)$$

Nous appelons segments les N sous-séquences de R' associées aux états $e_1,e_2,...,e_N$ suivant la segmentation $Q = q_1^T$. Par exemple pour l'information de durée, le premier segment seg_1^Q est la séquence $(d_1,...,d_{l1})$, le second segment seg_2^Q est $(d_{b2},...,d_{l2})$... et on calcule la probabilité de la composante de durée par :

$$P_{durée}(R'/q_1^T,\lambda) = P_{durée}(seg_1^Q seg_2^Q,\dots,seg_N^Q/q_1^T,\lambda) \quad (5)$$

Cette quantité peut être calculée par :

$$P_{durée}(R'/q_1^T,\lambda) = P_{durée}(seg_1^Q/e_1,\lambda) \prod_{i=2}^{N} P_{durée}(seg_i^Q/e_i,\ seg_1^Q seg_2^Q,\dots,seg_N^Q,\lambda) \quad (6)$$

Nous détaillons maintenant le calcul des trois quantités intervenant dans l'équation (3).

2.4.1. Composante de forme

La loi de probabilité dans un état ei associé à un tracé élémentaire s de Σ est définie par un calcul de similarités entre tracés élémentaires de Σ [Art02]. Cela permet, comme nous l'avons déjà dit, 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 associés à un même tracé élémentaire $s \in \Sigma$ (par exemple les 1er et le 3ème états du *MMCHN* de la Figure 3) partagent la même loi de probabilité d'émission, *i.e.* $\forall s' \in \Sigma$, $p(s'|e_i) = p(s'|e_j) = p(s'|s)$. Cette stratégie de partage permet de réduire de façon significative le nombre de paramètres libres du système puisque, quel que soit le nombre de *MMCHN* du système, il n'y a que 36 lois de probabilités d'émission sur Σ.

La probabilité de la composante de forme est calculée suivant :

$$P_{forme}(S'/q_1^T,\lambda) = \prod_{i=1}^{N} P_{forme}(s'_{b_i},s'_{b_i+1},\dots,s'_{l_i}/e_i,\lambda) \quad (7)$$

Nous utilisons pour la composante de forme l'hypothèse Markovienne classique d'indépendance des observations conditionnellement à la segmentation, et l'équation précédente se simplifie en :

$$P_{forme}(S'/q_1^T,\lambda) = \prod_{i=1}^{N} \prod_{t=b_i}^{j_i} P_{forme}(s'/e_i,\lambda) \quad (8)$$

2.4.2. Composante de durée

Pour calculer la vraisemblance des informations de durées nous exploitons la notion de segment. En effet, chaque état modélisant une partie du tracé il est naturel de définir des lois de probabilités sur les durées passées dans chaque état, c'est-à-dire sur la longueur totale passée dans un état. Par exemple, lorsque l'on calcule la vraisemblance d'un tracé avec le *MMCHN* de la Figure 3, on souhaite que la sous séquence des tracés élémentaires alignés sur le premier état (*i.e.* le segment associé au premier état) représente environ 30 % du tracé (c'est la durée du tracé « idéal »). En supposant une indépendance des durées passées dans les N états la probabilité de l'information de durée est calculée par :

$$P_{durée}(R'/q_1^T,\lambda) = P_{durée}(seg_1^Q/e_1,\lambda).P_{durée}(seg_2^Q/e_2,\lambda) \dots P_{durée}(seg_N^Q/e_N,\lambda) \quad (9)$$

où $P_{durée}(seg_i^Q/e_i,\lambda)$ représente la probabilité de la durée totale du $i^{\text{ème}}$ segment, suivant la segmentation Q, passée dans l'état e_i. Cette loi de probabilité est modélisée par une loi gaussienne centrée sur d_i (la durée du $i^{\text{ème}}$ stroke dans la *RHN* R) et calculée pour la durée totale passée dans l'état, c'est-à-dire $\sum_{t=b_i}^{l_i} d'_t$. La variance de cette loi, σ_d^2, est un hyper paramètre du système. Par exemple, dans le cas illustré par la Figure 5, la

vraisemblance de l'information de durée pour la segmentation Q donnée est calculée par :

$$P_{durée}(D'/Q,\lambda) = N(0.3,\sigma_d;0.29).N(0.6,\sigma_d;0.55) .N(0.1,\sigma_d;0.16) \quad (10)$$

où $N(0.3,\sigma_d;0.29)$ signifie la valeur d'une loi normale de moyenne 0.3 et d'écart-type σ_d calculée en 0.29. L'écart-type est partagé par toutes les lois de tous les *MMCHN*, de façon à limiter le nombre de paramètres effectifs du système.

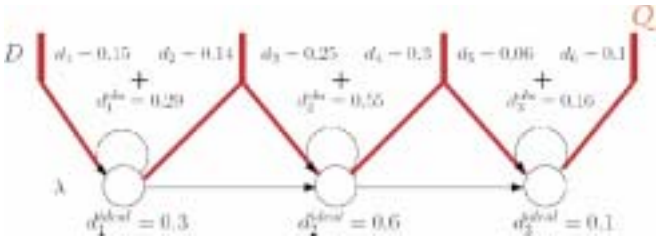

Figure 5. Illustration du calcul par segments de la probabilité de la composante de durées dans le MMCHN de la Figure 3. On prend un cas particulier. La RHN observée R', de longueur 6, a une composante de durée D' détaillée en haut de la figure (0.15, 0.14, 0.25, 0.3, 0.06, 0.1). La segmentation Q est illustrée par les traits épais : d'_1, d'_2 sont alignés sur e_1, d'_3, d'_4 alignés sur e_2 et d'_5, d'_6 sont alignés sur e_3. La vraisemblance de l'information de durée dans un état tient compte de la durée totale passée dans un (e.g. d'_1, d'_2 dans e_1) et de la durée idéalement passée dans cet état (e.g. d_1, la durée de s_1 dans la RHN R).

2.4.3. Composante spatiale

De même que pour la composante de durée, la vraisemblance liée à l'information spatiale fait intervenir la notion de segment. Ce qui nous intéresse c'est de caractériser la position du segment associé au premier état par rapport aux autres, pas de chaque stroke de S' les uns par rapport aux autres. Différentes modélisations de l'information spatiale sont possibles [Mar04-b], nous décrivons ici les modélisations que nous utilisons dans nos systèmes, une modélisation absolue et une modélisation relative. Ces deux informations étant complémentaires, nous utilisons une conjonction des deux de telle sorte que :

$$P_{spatial}(X/Q,\lambda) = P_{Spatial\ Relative}(X/Q,\lambda) .P_{Spatial\ Absolue}(X/Q,\lambda) \quad (11)$$

Nous décrivons le calcul de ces deux quantités dans la suite.

2.4.3.1. Information spatiale absolue

Utiliser une modélisation absolue de l'information spatiale est assez classique dans le traitement de l'écrit manuscrit [Kur99, Zhe03] et consiste à caractériser la position des différents constituants du caractère par rapport à un repère absolu, nous utilisons ici la boite englobante du signal manuscrit. En supposant une indépendance entre les positions des différents segments, la probabilité de l'information spatiale peut être calculée par :

$$P_{Spatial/Absolue}(X/Q,\lambda) = \prod_{i=1}^{N} P_{Absolu}(seg_i^Q/e_i,\lambda) \quad (12)$$

où seg_1^Q représente le segment associé à l'état e_i suivant la segmentation Q, c'est-à-dire la séquence des boites englobantes des tracés d'indices b_i à l_i, $(B_{b_i+1},\ldots,B_{l_i})$. Différents choix sont possibles pour la loi de probabilité $P_{Absolu}(seg_i^Q/e_i)$, nous avons choisi une modélisation simple où la loi de probabilité sur la position d'un segment associé à un état d'un *MMCHN* est modélisée par une loi normale sur la position du centre de gravité du segment :

$$P_{Absolu}(seg_i^Q/e_i,\lambda) = N\left(\mu_i,\Sigma_i;centre(seg_i^Q)\right) \quad (13)$$

où μ_i représente la position idéale (abscisse et ordonnée) du segment aligné avec le $i^{\text{ème}}$ état, Σ_i la variance autour de cette position idéale, et $centre(seg_i^Q)$ est le centre de gravité du segment seg_i^Q. Lors de la construction d'un *MMCHN* à partir d'une *RHN*, μ_i est choisi égal à la position du $i^{\text{ème}}$ tracé élémentaire. Afin de limiter le nombre de paramètres libres du système, toutes les matrice de covariance Σ_i sont égales à $\sigma^2_{spatial}.Id = \begin{bmatrix} \sigma^2_{spatial} & 0 \\ 0 & \sigma^2_{spatial} \end{bmatrix}$ où $\sigma_{spatial}$ est un hyper paramètre du système.
Afin de conférer plus de robustesse à cette modélisation, nous utilisons des coordonnées normalisées, c'est-à-dire que nous utilisons les coordonnées dans la boite englobante normalisée du signal, où l'abscisse comme l'ordonnée varient entre 0 et 1.

2.4.3.2. Information spatiale relative

Utiliser une information relative consiste à décrire les positions des différents segments les uns par rapport aux autres plutôt que par rapport à un repère absolu comme précédemment [Cho99]. Cette modélisation est plus fine dans le cas de caractères complexes écrits en plusieurs fois, avec des levés de stylo. Dans la suite, nous expliquons comment nous modélisons une relation spatiale élémentaire (*RSE*), c'est-à-dire comment caractériser la position d'un stroke (ou segment) par rapport à un autre stroke ou segment : est-il à gauche, à droite etc ? Avant cela, nous expliquons comment décrire toute l'information spatiale d'un tracé à partir des relations spatiales élémentaires existant entre ses constituants.

Description globale

Supposons que l'on dispose, pour une segmentation Q donnée, des *RSE* de tous les segments les uns par rapport aux autres. On pourrait, pour décrire le plus complètement l'information spatiale du tracé manuscrit, utiliser toutes ces *RSE* mais la modéli-

sation deviendrait délicate à intégrer dans une procédure de programmation dynamique. Nous avons choisi de ne conserver dans notre modélisation que les *RSE* d'un segment par rapport aux segments précédents (*i.e.* associés aux états précédents dans un *MMCHN* gauche droite). Ainsi :

$$P_{Spatial\ Relative}(X/Q,\lambda) = \prod_{i=2}^{N}\prod_{j=1}^{i-1} p_{spatial}\left(rse(seg_i^Q,seg_j^Q)/rse_{i,j}^{idéal},\lambda\right) \quad (14)$$

Où $rse(seg_i^Q,seg_j^Q)$ représente la relation spatiale élémentaire entre les segments associés, suivant la segmentation Q, aux états e_i et e_j, et $rse_{i,j}^{idéal}$ représente la relation spatiale élémentaire « idéale » entre ces deux segments. Lors de la construction d'un *MMCHN* à partir d'une *RHN*, cette relation spatiale élémentaire idéale est celle observée entre le $i^{\text{ème}}$ tracé et le $j^{\text{ème}}$ tracé de la *RHN*.

Relation spatiale élémentaire

On peut caractériser simplement la position d'un segment par rapport à un autre à l'aide d'attributs tels que la position horizontale, la position verticale, et la connexité. Ainsi, dans la figure 6 ci-dessous, on peut par exemple définir une *RSE* entre les deux tracés s_1 et s_2, caractérisant la position de s_2 par rapport à s_1, comme un triplet de trois attributs (*Position Horizontale*, *Position Verticale*, *Connexité*) prenant la valeur (« à droite », « au dessous », « ne touche pas »).

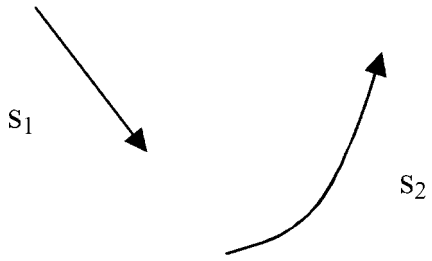

Figure 6. Caractérisation de la position d'un tracé par rapport à un autre. Ici, le tracé s_2 est à droite, en dessous et ne touche pas le tracé s_1. La RSE correspondante est donc un triplet du type (à droite, en dessous, ne touche pas).

Ce type de relation spatiale élémentaire, basée sur des attributs discrets, n'est pas assez robuste en général. Nous utilisons plutôt, pour la position horizontale comme pour la position verticale, 3 valeurs réelles dont le calcul est illustré par la figure 7. Soient b_1 et b_2 les boites englobantes des tracés s_1 et s_2, on calcule les proportions de la boite englobante de s_2 qui sont au dessus, alignées ou en dessous de la boite englobante de s_1.
La loi de probabilité $p_{spatial}\left(rse(seg_i^Q,seg_j^Q)/rse_{i,j}^{ideal},\lambda\right)$ est implémentée par :

$$p_{spatial}\left(rse(seg_i^Q,seg_j^Q)/rse_{i,j}^{ideal},\lambda\right) = N\left(ph_\lambda(seg_i^Q,seg_j^Q),Id;\ ph(seg_i^Q,seg_i^Q)\right). N(pv_\lambda(seg_i^Q,seg_j^Q),id;\ pv(seg_i^Q,seg_j^Q)) \quad (15)$$

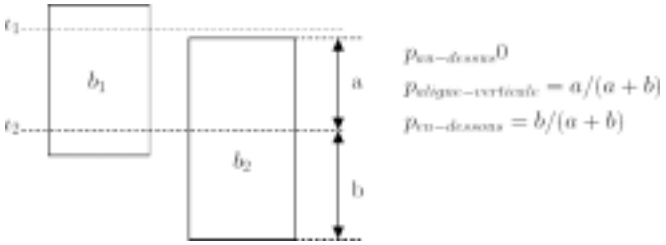

Figure 7. Relation spatiale élémentaire basée sur des attributs continus. Illustration du calcul du positionnement vertical du tracé s_2 par rapport au tracé s_1. À partir des boites englobantes de ces deux tracés, on calcule la proportion de la boite englobante de s_2 qui est plutôt au dessus de la boite englobante de s_1, ou alignée avec la boite englobante de s_1, ou en dessous de la boite englobante de s_1. Dans l'exemple, le positionnement vertical de s_2 par rapport au tracé s_1 est caractérisé par le vecteur de 3 valeurs $(0,a/(a+b),b(a+b))$.

où $N\left(ph_\lambda(eg_i^Q,seg_j^Q),Id;\ ph(seg_i^Q,seg_j^Q)\right)$ représente la valeur de la gaussienne centrée sur le vecteur idéal de positionnement horizontal $ph_\lambda(seg_i^Q,seg_j^Q)$ calculée pour le vecteur représentant le positionnement horizontal des segments associés aux états e_i et e_j suivant la segmentation Q. De même, $N\left(pv_\lambda(seg_i^Q,seg_j^Q),Id;\ pv(seg_i^Q,seg_j^Q)\right)$ représente la probabilité, implémentée par une loi gaussienne, du positionnement vertical observé.
Lors de la construction d'un *MMCHN* à partir d'une *RHN R*, les positionnements « idéaux » $ph_\lambda(seg_i^Q,seg_j^Q)$ et $pv_\lambda(seg_i^Q,seg_j^Q)$ sont ceux qui sont calculés entre les tracés de la *RHN R*.

3. Interprétation formelle du système

On peut interpréter formellement le système décrit jusqu'ici, notamment ses deux particularités qui sont l'architecture Markovienne à deux niveaux, et les modèles *MMCHN* travaillant sur trois flux d'informations simultanés. La vision que nous en avons donnée jusqu'ici consiste en un prétraitement complexe suivi de modèles de type *MMC* opérant sur trois flux d'informations indépendant conditionnellement à une segmentation. Nous faisons le lien ici avec d'une part les modèles de Markov hiérarchiques pour l'architecture à deux niveaux et avec d'autre part les réseaux bayésiens dynamiques pour le traitement de plusieurs types d'information (forme, durée et position).

3.1. Modèles de Markov hiérarchiques

Tout d'abord, on peut lier notre approche à des modèles de Markov hiérarchiques [Fine98]. Nous oublions ici, pour la clarté de la présentation, les informations de durées et les informations spatiales, une *RHN* est donc constituée uniquement d'une séquence de tracés élémentaires, calculée par le système de bas niveau.

Pour calculer la vraisemblance d'un signal observé X (*i.e.* une séquence temporelle de points) par un modèle *MMCHN* d'un caractère, notre approche consiste dans un premier temps à calculer la *RHN* du signal, obtenue par le système de bas niveau, puis à calculer la vraisemblance de cette *RHN* par le *MMCHN*. On calcule donc :

$$\begin{cases} P(X/\lambda) = P(N',S'/\lambda_{\text{Haut Niveau}}) \\ pour \\ (N',S') = \text{argmax}_{N,S}\ P(X/N,S,\lambda_{\text{Bas Niveau}}) \end{cases} \quad (16)$$

où $\lambda_{\text{Haut Niveau}}$ désigne le *MMCHN* et $\lambda_{\text{Bas Niveau}}$ désigne le système qui calcule la *RHN* du tracé. Il n'y a pas de loi de probabilité sur la longueur de la *RHN*, *N*, *i.e.* toutes les longueurs sont équiprobables.

Comparativement, un *MMC* hiérarchique très similaire, dont le second niveau serait proche du système de bas niveau, calculerait lui une probabilité du type :

$$\begin{aligned} P(X/\lambda_{\text{hiérarchique}}) &= \sum_{N',s'_1,s'_1,\ldots,s'_{N'}} P(X,N',S'/\lambda_{\text{hiérarchique}}) \quad (17) \\ &= \sum_{N',s'_1,s'_1,\ldots,s'_{N'}} P(X/N',S',\lambda_{BN}).P(N',S'/\lambda_{HN}) \end{aligned}$$

où λ_{HN} désigne le deuxième niveau du modèle hiérarchique, modèle opérant sur les séquences de tracés élémentaires (analogue au *MMCHN* précédent), et S' est une segmentation de X dans le premier niveau du modèle hiérarchique, λ_{BN}.

Par comparaison, un *MMCHN* conjugué au prétraitement calculé par le système de bas niveau, calcule donc une quantité proche de la précédente mais où la somme sur les *RHN* (N',S') est remplacée par son élément maximum. Notre système, équivalent à une cascade de modèles Markoviens, peut ainsi être vu comme une approximation peu coûteuse de modèles de Markov cachés hiérarchiques.

3.2. Réseaux Bayésien dynamiques

Une autre vison possible de nos modèles peut être obtenue en les formulant comme des réseaux bayésiens dynamiques (remarquons que les *MMC*s hiérarchiques peuvent également être exprimés de cette façon) [Murphy02]. Des travaux allant dans ce sens ont été proposés notamment dans [Cho99]. Commençons par représenter un état d'un modèle *MMCHN* comme un réseau Bayésien. La figure 8 montre un réseau bayésien représentant un état de ce type, en explicitant la dépendance de la forme du tracé, de sa durée et de sa position vis à vis de l'état, et l'indépendance conditionnelle de ces quantités, étant donné l'état.

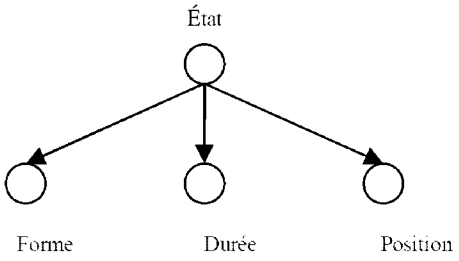

Figure 8. Modèle bayésien d'un état d'un MMCHN.

On peut raffiner cette modélisation en considérant par exemple le modèle d'un caractère constitué de deux tracés particuliers (*e.g.* comme un « y » tracé en deux traits rectilignes). La figure 9 montre un réseau bayésien correspondant à un modèle de caractère de ce type. On y voit les dépendances des deux états au noeud caractère ainsi que les indépendances conditionnelles des informations de forme de durée et de position étant donnés les états. Nous avons fait figurer également un noeud correspondant à la relation spatiale relative des deux tracés, dépendant de leurs positions respectives.

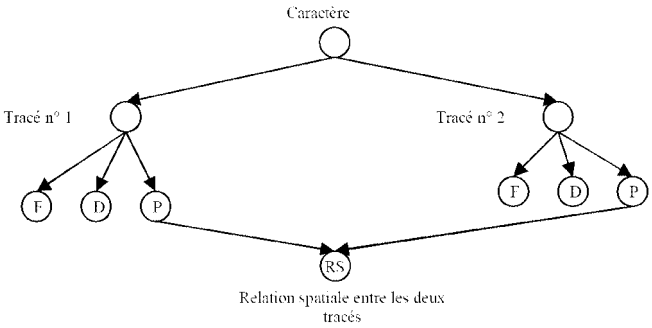

Figure 9. Réseau bayésien correspondant à un modèle de caractère composé de deux tracés, incluant un nœud modélisant la relation spatiale entre ces deux tracés.

Cependant, nous devons tenir compte dans notre modélisation qu'un caractère peut être représenté par un nombre variable de tracés élémentaires, ce qui n'est pas pris en compte par le modèle ci-dessus. Il faut donc pouvoir représenter la dynamique dans nos modèles et prendre en compte des représentations de longueurs variables. La figure 10 ci-dessous présente un *MMCHN* sous forme d'un réseau bayésien dynamique, nous avons omis ici la modélisation de l'information spatiale pour simplifier le schéma. Les significations des différentes quantités correspondant aux noeuds de la figure sont les suivants :

q_t : état à l'instant t
s_t : tracé élémentaire émis à l'instant t
d_t : Durée tracé élémentaire s_t
DT_t : durée totale passée dans l'état q_t
DR_t : durée restante dans l'état qt
F_t : indicateur de changement d'état ($F_t = 0$ si $q_t = q_{t+1}$, 1 sinon). L'introduction de ce type de nœud est inspirée de [Murphy02]

Ces quantités obéissent aux lois suivantes, en utilisant la topologie gauche droite des *MMCHN* considérés :

$q_t = q_{t-1} + 1$ *si* $F_{t-1} = 1$; $q_t = q_{t-1}$ *si* $F_{t-1} = 0$
s_t *est émis avec la loi de probabilité sur* Σ *associée à* q_t
d_t *est émis avec une loi de probabilité uniforme, i.e. il n'y a pas d'a priori sur la longueur d'un stroke*
$D_t = D_{t-1}$ *si* $F_{t-1} = 0$; *si* $F_{t-1} = 1$, *on utilise la loi de probabilité sur la durée à passer dans l'état* q_t
$DR_t = DR_{t-1} - d_t$ *si* $F_{t-1} = 0$; $DR_t = DT_{t-1} - d_t$ *si* $F_{t-1} = 1$
$F_t = 1$ *si* $DR_{t-1} = 0$, $F_t = 0$ *sinon*

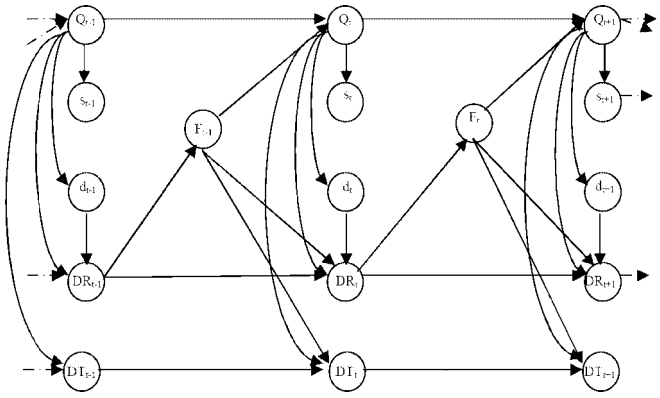

Figure 10. Représentation d'un MMCHN sous forme de réseau Bayésien dynamique (voir le texte pour la signification des différentes quantités).

Cette représentation des *MMCHN* sous forme de réseaux bayésiens dynamiques permet en théorie d'exploiter la théorie puissante développée pour ces modèles, pour réaliser l'inférence, l'apprentissage etc. En pratique les algorithmes développés pour ce type de modèles sont assez lourds et ne permettent pas encore d'apprendre efficacement la structure, comme nous l'avons fait pour les *MMCHN*. L'objet de cette discussion consiste donc essentiellement à montrer que nos modèles peuvent être reliés à des modèles plus classiques. Nous travaillons actuellement sur la façon d'exploiter cette formalisation.

4. Évaluations

Afin d'explorer le potentiel de notre approche, nous avons réalisé des expériences sur des données manuscrites variées. Nous étudions d'abord en détail les caractéristiques générales de notre approche, qui sont le résultat des idées fondamentales sous-jacentes de l'approche décrites dans le §3. Puis nous étudions la faisabilité de deux types de systèmes très différents. Les premiers sont des systèmes omni-scripteur pour des caractères alphanumériques usuels, ils doivent être capable de prendre en compte une variabilité inter-scripteurs importante, mais pour des caractères assez simples. Les seconds sont des systèmes 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, de notre système et de systèmes existants. Le lecteur intéressé pourra consulter [Mar04-a] pour plus de résultats expérimentaux.

4.1. Base de données

Nos expériences portent sur différentes bases de données que nous présentons maintenant.

4.1.1. Chiffres, caractères minuscules et majuscules (base UNIPEN)

La base UNIPEN [Guy94] 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 des 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 d'exemples total des chiffres (resp. des lettres majuscules et minuscules) est 15719 (resp. 22683 et 59691).

4.1.2 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. 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 (11-a) et (11-b) montrent différents styles d'écriture d'un même caractère et les figures (11-c), (11-d) et (11-e)) montrent des caractères distincts mais très semblables. Nos expériences portent sur un ensemble de 38 caractères coréens, ceux suffisamment représentés dans la base.

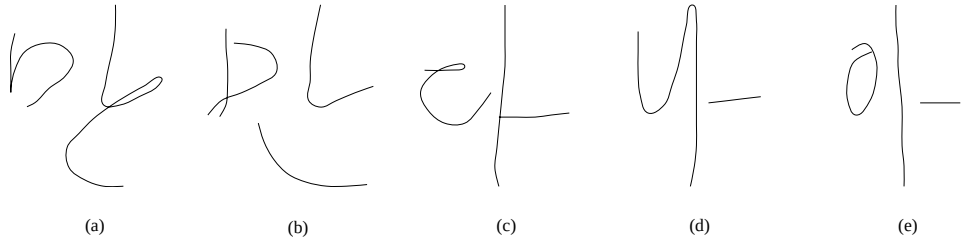

Figure 11. Variabilité et complexité de l'écriture coréenne : deux styles d'écriture différents pour un même caractère coréen ((a) et (b)) ; trois caractères très similaires (c), (d) et (e).

4.1.3 Symboles (base LIP6)

Cette base de symboles, collectée au LIP6, a pour but d'évaluer certaines caractéristiques du système plus précisément (voir figure 12). 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.

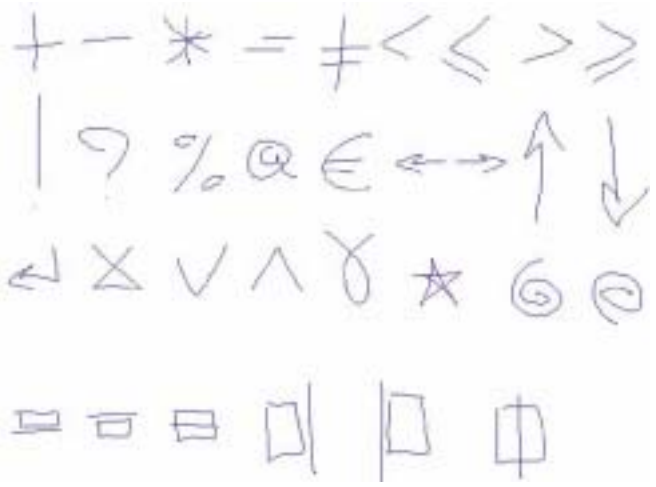

Figure 12. Exemples de tracés de la base de symboles collectée au LIP6.

4.2. Caractéristiques de l'approche

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

4.2.1. Variabilité dans les tracés

Les premières expériences concernent la prise en compte de la variabilité dans les tracés de caractères. Pour cela, nous avons testé notre système sur des caractères de complexités variables, 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 ainsi 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 19 caractères seulement, les autres étant trop peu représentés pour un même scripteur) pour les caractères coréens et de 3 scripteurs pour les symboles de la base LIP6.

Les résultats obtenus sont résumés dans la figure 13. 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. On remarque également que pour tous ces types de caractères, augmenter la taille des modèles permet d'atteindre des performances de l'ordre de 95 %.

Ces résultats démontrent la capacité de notre approche, basée sur des mélanges de modèles complètement appris à partir des données, à absorber la variabilité et la complexité des tracés de tout type de caractères. Par exemple, des résultats tout à fait raisonnables sont obtenus pour les caractères coréens alors que ceux-ci sont écrits avec ou sans ligatures, que les graphèmes les composant sont écrits en une ou plusieurs fois etc. On conclut également que 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.

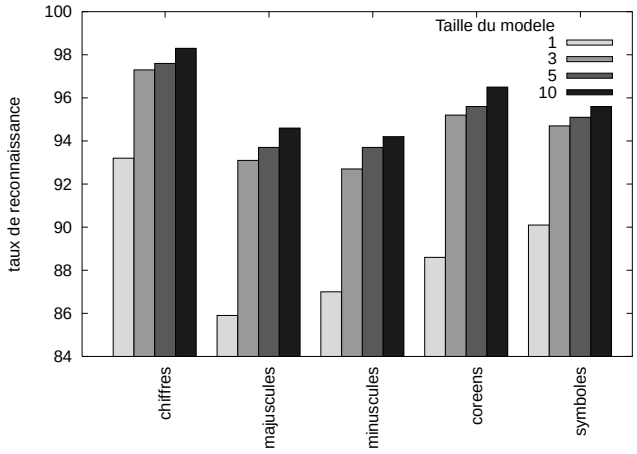

Figure 13. Prise en compte de la variabilité dans les tracés des caractères par l'augmentation de la taille des modèles de caractères.

4.2.2. Variabilité scripteur

Afin d'étudier la prise en compte de 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 14 montre les taux de reconnaissance obtenus en mode mono-scripteur et en mode multi-scripteur avec (10, 15 ou un peu plus de 200 scripteurs). Pour chaque expérience, 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 §4.3.2 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.

4.2.3. 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

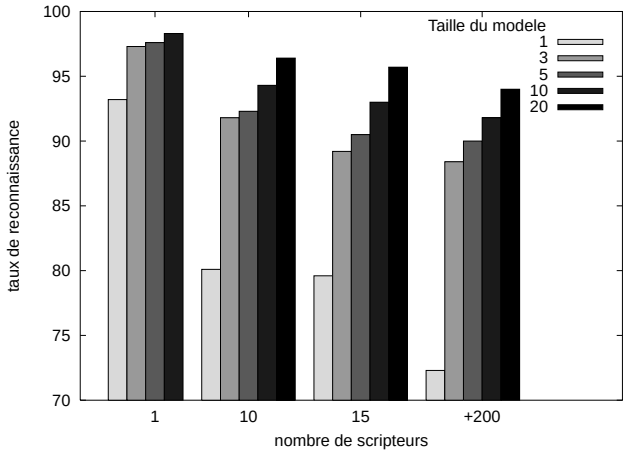

Figure 14. Prise en compte de la variabilité inter-scripteur par l'augmentation de la taille des modèles de caractères.

varier la taille de la base d'apprentissage et cela pour différentes tailles de modèles. Les expériences présentées ici ont été réalisées sur les caractères coréens de la base KAIST, en mode mono-scripteur, mais on obtient des résultats similaires sur toutes les bases de données. 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. La figure 15 montre les performances moyennes (sur l'ensemble de scripteurs) en fonction du nombre d'exemples d'apprentissage par caractère en apprentissage et de la taille des modèles de caractères. D'après ces résultats, on voit qu'avec 5 exemples en apprentissage seulement 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 %. Les résultats sont bien sur encore meilleurs avec des caractères plus simples comme des chiffres.

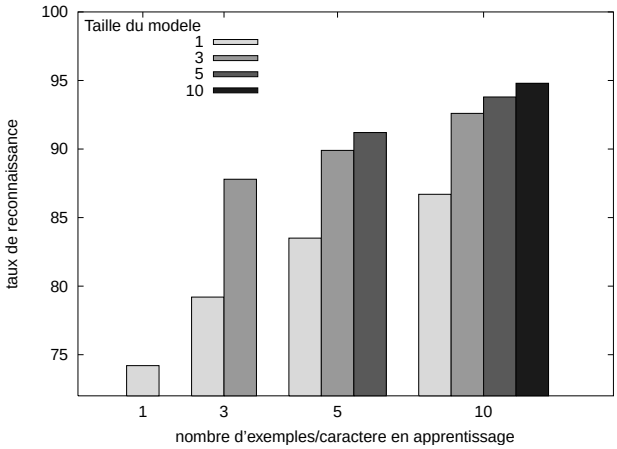

Figure 15. Performances en mode mono-scripteur en fonction du nombre d'exemples par caractère en apprentissage et de la taille des modèles de caractères pour la reconnaissance de caractères coréens.

4.3. Conception de systèmes

Les propriétés de notre approche que les expériences précédentes ont mises 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.

4.3.1. Systèmes mono-scripteur pour terminaux mobiles

Les résultats du §4.2 ont montré que notre approche permet de concevoir des systèmes de reconnaissance mono-scripteur ayant des taux de reconnaissance élevés (Figure 13), tout à fait acceptables du point de vue de l'acceptation par un utilisateur (de l'ordre de 95 %), et pour tout type de caractères. 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 (Figure 15). 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 [Lon99]. Nous verrons enfin au §4.4 que les ressources machine d'un système mono-scripteur (nécessitant une taille de modèle de l'ordre de 5) sont minimes et compatibles avec des terminaux légers.

4.3.2. Systèmes Multi et Omni-scripteur pour caractères alphanumériques

Nous étudions ici la faisabilité, avec notre approche, de systèmes multi ou omni-scripteur pour des caractères usuels. Nous évaluons nos systèmes 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 [Rat03]. 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 nos systèmes atteignent des performances comparables aux meilleurs classifieurs existants, de 2 à 3 % inférieurs aux meilleurs résultats rapportés, en multi scripteur et en omni-scripteur. Nous remarquons aussi que même avec une base de données de taille restreinte (5 %), notre système obtient déjà des bons résultats. Par ailleurs, notre approche, étant basée sur une modélisation markovienne, est extensible à la reconnaissance de mots, contrairement aux approches qui sont comparées ici. Enfin, la complexité et la performance de nos systèmes sont d'une certaine façon réglables via l'hyper-paramètre de l'apprentissage, la taille des modèles de caractères.

Tableau 1. Comparaison de notre approche et de travaux existants pour la conception de systèmes omni et multi scripteurs sur les signaux de la base Unipen. Les résultats sont donnés pour différentes tailles de la base d'apprentissage, PBA est le pourcentage de la base Unipen utilisé en apprentissage (5 %, 20 à 30 %, 50 à 60 %). Kc désigne la taille des modèles de caractères dans nos systèmes.

PBA	Système de reconnaissance	chiffres		majuscules		minuscules	
		multi	omni	multi	omni	multi	omni
5	*MMCHN*						
	K_c = 5	92.4	87.5	85.1	80.2	80.9	78.5
	K_c = 10	94.1	89.9	86.9	82.7	83.5	80.3
	K_c = 20	95.1	90.7	87.9	83.8	85.0	81.7
	K_c = 50	95.4	91.9	89.1	84.9	85.9	82.6
20-30	*MMCHN*						
	K_c = 5	92.3	90.3	84.7	85.6	81.7	80.3
	K_c = 10	95.1	91.9	86.9	87.8	84.1	82.8
	K_c = 20	96.4	93.4	88.6	89.1	86.2	84.5
	K_c = 50	97.1	94.9	89.2	89.5	87.9	85.9
	sn-tuple [Rat03]	98.7	97.1	93.7	91.8	90.6	87.8
	SDTW [Bahl01]	95.5		90.0		88.6	
	SVM [Bahl02]	96.0		92.4		87.9	
	HMM [Li98]	92.0					
	MLP [Lem02]		95.6				
	knn + SVM [Vuur00]			94.8			
50-60	*MMCHN*						
	K_c = 5	93.5	90.7	88.3	87.0	80.9	80.8
	K_c = 10	95.2	92.3	90.4	89.3	84.3	83.7
	K_c = 20	96.6	94.5	91.9	90.2	86.7	85.7
	K_c = 50	97.7	95.9	92.8	91.2	88.2	87.0
	À la 1-ppv	98.5	97.2	94.4	92.6		
	sn-tuple [Rat03]	98.8	98.1	95.1	94.0	92.0	
	Knn [Pre00]		98.8		96.6		96.3
	HMM Segmental [Art00]					79.5	69.8
	Bayesian Network [Cho99]		95.0				
	Fuzzy ARTMAP [Gom98]	82.4					
	HMM [Hu00]		96.8		93.6		85.9

4.4. 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.

4.4.1. 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.* [Bahl02, Vuur00], nécessitent une mémoire importante pour stocker les vecteurs supports. Dans [Bahl02], environ 100 vecteurs supports par classe sont nécessaires pour obtenir les performances du tableau, 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 [Vuur00], 21 Mo sont nécessaires pour le stockage des vecteurs supports. Le classifieur sn-tuple [Rat03] 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 a environ 19.3 Mo pour la classification des chiffres.

En comparaison, notre système nécessite moins de ressources mémoires. Pour donner une idée, le stockage de toutes les *RHN* des chiffres de la base UNIPEN (15719 exemples) tient en 8.2 Mo. Un système de reconnaissance des 26 lettres minuscules, avec des modèles de caractères de taille 20 (resp. 50) nécessite environ 500 Ko (resp. 1 Mo).

En ce qui concerne les modèles à base de prototypes, proche des nôtres, le nombre de prototypes nécessaires pour obtenir de bons résultats se situent généralement entre 40 et 50 prototypes par symboles. Par exemple, dans [Pre00], environ 47.6 prototypes par chiffre sont nécessaires pour obtenir un taux de 98.8 % de reconnaissance. Pour les lettres majuscules et minuscules, 44.5 et 42.9 prototypes par lettre sont nécessaires. Dans [Vuur00] environ 42.9 prototypes par lettre sont nécessaires pour obtenir un taux de 94.8 % de reconnaissance pour les lettres majuscules. Ces systèmes sont du même ordre de grandeur que le nôtre en ressource mémoire.

4.4.2. 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 immédiate 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 [Vuur00], nos systèmes sont plus rapides. Ceci est dû à la compacité des *RHN* utilisées dans notre système. En effet, un tracé en ligne d'une trentaine de points est typiquement représenté par une *RHN* de longueur 5. Or la complexité des algorithmes de programmation dynamique utilisés en reconnaissance est proportionnelle à la longueur de la description.

Tableau 2. Comparaison des vitesses de reconnaissance, en nombre de caractères par seconde (#CS) pour différents systèmes de reconnaissance de caractères. #C désigne le nombre de caractères traités par ces systèmes. K_c désigne la taille des modèles de caractères dans nos systèmes.

Système		#C	#CS	Machine
sn-tuple [Rat03]		10	70	300 MHz RS6000 workstation
		26	420	
SVM with GDTW kernel [Bahl02]		26	0.4	AMD Athlon 1200 MHz
Bayesian Network [Cho99]		10	84	IBM ThinkPad T23, 1.13 GHZ
Fuzzy ARTMAP [Gom98]		10	3.57	Pentium 120MHz
HMM [Hu00] SGI station		–	3.3	180 MHz
knn [Vuur00] avec post-traitement SVM		26	2.88	Pentium-II 400 MHz
MMCHN	K_c = 5	10	520	Pentium-III 500 MHz
		26	230	
	K_c= 50	10	74	
		26	20	

5. Conclusion

Nous avons présenté dans cet article une approche générique pour la conception de systèmes de reconnaissance de symboles et caractères graphiques en ligne. Cette approche est basée sur quelques idées simples qui lui confèrent des propriétés intéressantes du point de vue de la robustesse, de la prise en compte de la variabilité, et de la capacité d'apprentissage à partir de peu d'exemples. Par ailleurs, l'apprentissage des modèles, réalisé complètement et automatiquement à partir des données d'ap-

prentissage, permet de concevoir des systèmes traitant n'importe quel type de caractères (dans une limite de complexité raisonnable).

Nous avons réalisé un certain nombre d'expérimentations visant à valider de façon approfondie les caractéristiques de notre approche pour la reconnaissance de signaux écrits en ligne et à la comparer aux systèmes actuels. 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. Enfin, l'approche étant basée sur des modèles Markoviens, elle est en principe facilement extensible à la reconnaissance de mots ou de phrases.

6. Références

[Art00] T. ARTIÈRES and J.-M. MARCHAND and P. GALLINARI and B. DORIZZI, Stroke Level Modelling of On line Handwriting through Multi-modal Segmental Models, IWFHR, 2000.

[Art02] T. ARTIÈRES and P. GALLINARI, Stroke level HMMs for on-line handwriting recognition, International Workshop on Frontiers in Handwriting Recognition (IWFHR), 2002.

[Bahl01] C. BAHLMANN and H. BURKHARDT, Measuring HMM Similarity with the Bayes Probability of Error and its Application to Online Handwriting Recognition, International Conference on Document Analysis and Recognition (ICDAR), 2001.

[Bahl02] C. BAHLMANN, B. HAASDONK and H. BURKHARDT, On-line Handwriting Recognition with Support Vector Machines – A Kernel Approach, IWFHR, 2002.

[Cho99] S.J. CHO and J.H. KIM, Bayesian Network Modeling of Strokes and Their Relationships for On-line Handwriting Recognition, ICDAR, 1999.

[Den94] L. DENG, M. AKSAMOVIC, X. SUN, C.W. WU, Speech recognition using hidden Markov models with polynomial regression functions as nonstationary states, 1994, IEEE Trans. On Speech & Audio Proc., Vol. 2, n° 4.

[Fine98] S. FINE, Y. SINGER, N. TISHBY, The hierarchical Hidden Markov Model : analysis and applications, Machine Learning, 1998.

[Gom98] E. GOMEZ SANCHEZ, J.A. GAGO GONZALEZ and Y.A. DIMITRIADIS, Experimental study of a novel neuro-fuzzy system for on-line handwritten UNIPEN digit recognition, Pattern Recognition Letters, 1998, volume 19, pages 357-364.

[Kur99] K. KURODA, K. HARADA, M. HAGIWARA, Large scale on-line handwritten Chinese character recognition using successor method based on stochastic regular grammar, Pattern Recognition, Vol. 32, N° 8, 1999.

[Guy94] I. GUYON, L. SCHOMAKER and R. PLAMONDON, M. LIBERMAN and S. JANET, UNIPEN project of on-line data exchange and recognizer benrchmak, International Conference on Pattern Recognition (ICPR), volume 2, 1994.

[Hu00] J. HU, S.G. LIM, M.K. Brown, Writer independent on-line handwriting recognition using an HMM approach, Pattern Recognition, 2000, volume 33, number 1, pages 133-148.

[Lee00] J.J. LEE, J. KIM, J.H. KIM, Data-Driven Design of HMM Topology for Online Handwriting Recognition. IJPRAI, Vol. 15, N° 1, 2001.

[Lem00] A. LEMIEUX, C. GAGNE and M. PARIZEAU, Genetical Engineering of Handwriting Representations, IWFHR, 2002.

[Li98] X. LI, R. PLAMONDON and M. PARIZEAU, Model-based On-line Handwritten Digit Recognition, ICPR, 1998.

[Liu01] C.-L. LIU, I.-J. KIM, J.H. KIM, Model-based stroke extraction and matching for handwritten Chinese character recognition. Pattern Recognition, Vol. 34, N° 12, 2001.

[Lon99] A.C. LONG, J.A. LANDAY and L.A. ROWE, Implications for a Gesture Design Tool, Conference on Human factors in computing systems (CHI), 1999.

[Mar03] S. MARUKATAT, R. SICARD, T. ARTIÈRES and P. GALLINARI, A flexible recognition engine for complex on-line handwriting character recognition, ICDAR, 2003.

[Mar04-a] S. MARUKATAT, Une approche générique pour la reconnaissance de signaux écrits en ligne, Thèse de doctorat, LIP6, Université Paris 6, Juin 2004.

[Mar04-b] S. MARUKATAT and T. ARTIÈRES, Handling spatial information in on-line handwriting recognition, à paraître dans IWFHR'04.

[Mar04-c] S. MARUKATAT, T. ARTIÈRES, P. GALLINARI, A generic approach for on-line handwriting recognition, à paraître dans IWFHR'04.

[Murphy02] K.P. MURPHY, Dynamical Bayesian Networks : Representation, Inference and Learning, Ph. D, University of California, Berkeley, 2002.

[Nak01] M. NAKAI, N. AKIRA, H. SHIMODAIRA, S. SAGAYAMA, Substroke Approach to HMM-based On-line Kanji Handwriting Recognition, ICDAR, 2001.

[Pre00] L. PREVOST and M. MILGRAM, Modelizing character allographs in omni-scriptor frame : a new non-supervised clustering algorithm, Pattern Recognition, 2000, volume 21, pages 295-302.

[Rat03] Eugene H. RATZLAFF, Methods, Report and Survey for the Comparison of Diverse Isolated Character Recognition Results on the UNIPEN Database, ICDAR, 2003.

[Vuur00] L. VUURPIJL and L. SCHOMAKER, Two-Stage Character Classification : A Combined Approach of Clustering and support Vector Classifiers, IWFHR, 2000.

[Zhe03] J. ZHENG, X. DING, Y. WU, Z. LU, Spatio-temporal Unified Model for On-line Handwritten Chinese Character Recognition, ICDAR, 1999.

Patrick **Gallinari**

Patrick Gallinari is Professor at the University of Paris 6 (Université Pierre et Marie Curie, France). He is currently heading the Computer Science Lab (LIP6). He is in charge of a team at the Laboratoire d'Informatique of Paris 6 (LIP6), working on the digital learning area. The activities of this team are centered on the development of digital models of learning (markovian models, regression and classification trees, neural networks), from the theoretical point of view, as well as the applicative one (Speech Recognition, Handwritting Recognition, sequence processing). He is author or co-author of about a hundred publications, and was responsible from 1995 to 1997 in France of he Excellence European Network "Neuronet" concerning neural networks.

Thierry **Artières**

Thierry Artières is assistant Professor at the University of Paris 6 (Université Pierre et Marie Curie, France). He has worked on speech and speaker recognition, on-line and off-line handwriting recognition, pen-based interfaces, user modelling. His research activities concern statistical tools for signal modelling and classification. He is author or co-author of about 40 publications.

Sanparith **Marukatat**

Sanparith Marukatat defended his Ph.D in July 2004 at the University of Paris 6. He is now researcher at Nectec, Thaïland. He has worked on on-line handwriting recognition.