Fusion probabiliste d'informations appliquée à la reconnaissance dynamique de textes

Probabilistic fusion for on-line hand-printed text recognition

L. Oudot L. Prevost M. Milgram

Université Pierre & Marie Curie
Laboratoire des Instruments & Systèmes d'Ile de France
Groupe Perception, Automatique & Réseaux Connexionnistes
4 Place Jussieu, Paris Cedex 75252, Case 164

loic.oudot@lis.jussieu.fr

Résumé

Nous présentons dans cet article un nouveau moteur de reconnaissance de textes en écriture scripte dynamique non contraints. Les sources d'informations extraites des données d'acquisition sont riches et se situent à des niveaux d'abstraction différents. Nous disposons ainsi d'informations géométriques, morphologiques et lexicales qu'ils convient de fusionner efficacement. Une méthode de fusion robuste aussi bien mathématiquement que pratiquement, consiste à utiliser le formalisme Bayésien pour gérer les probabilités et les règles de fusions associés. Le système présenté ici est basé sur plusieurs détecteurs probabilistes qui serviront à l'expert de segmentation pour générer un treillis d'hypothèses de segmentation. Ce dernier sera exploré par un moteur guidé par un lexique français de 200 000 formes.

Mots Clef

Ecriture dynamique scripte, fusion d'informations, probabilité, exploration lexicale.

Abstract

We present in this article a new text recognition engine of non constrained on-line hand-printed text. The informations extracted from the acquired data are rich and located at different levels of abstraction. We have topological, morphological and lexical informations and have to merge these appropriately. A robust method of fusion as well mathematically as practically, consists in using the probabilities and the associated rules of fusions (Bayesian formalism). We present here a system based on several probabilistic detectors. The expert of segmentation uses them to generate a set of segmentations. This latter will be explored by an engine guided by a French lexicon of 200 000 forms.

Keywords

On-line script handwriting recognition, data fusion, probabilities, lexical exploration.

1 Introduction

Le développement rapide des assistants personnels et des ordinateurs portables dépourvus de clavier et de souris nécessite la mise en place de nouveaux outils de saisie. Le stylo apparaît aujourd'hui comme un très bon moyen de navigation et d'interaction entre l'Homme et la machine et s'étend aujourd'hui sur tous ces nouveaux ordinateurs. Mais l'utilisation faite sur ces derniers du stylo est encore extrêmement limitée, seules les fonctions de pointage sont disponibles. L'utilisateur doit encore saisir un texte à l'aide d'un clavier virtuel. Les systèmes de reconnaissances d'écriture commercialisés actuellement sont encore très limités (cf.*graffiti* `http://www.palm.com/products/input/`) et très contraignants pour l'utilisateur.

La plupart des systèmes de reconnaissance de l'écriture en cours de recherche sont centrés sur l'écriture cursive non contrainte, c'est-à-dire l'écriture naturelle. Ces systèmes utilisent généralement une approche analytique qui repose sur une segmentation du mot à reconnaître [6, 8]. L'entité étant alors le graphème. L'intérêt de ce découpage est de n'avoir que des modèles de graphèmes ou de lettres. Ces modèles sont limités (26 modèles pour les minuscules par exemple) et permettent de modéliser tous les mots quelque soit la taille du lexique utilisé.

Les deux principales méthodes qui prédomine pour la classification des graphèmes, sont les réseaux neuronaux [5] et les modèles de Markov cachés [12]. Ils sont aujourd'hui très bien connus du point de vue théorique et ils ont fait leurs preuves d'efficacité dans la pratiques. D'autres méthodes de reconnaissance de formes ont été utilisées

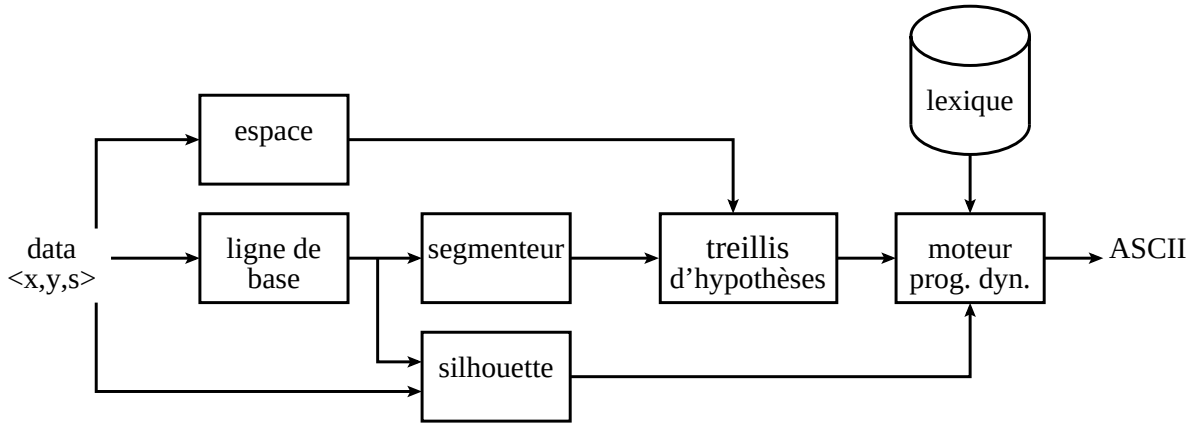

FIG. 1 – Structure générale du système de lecture.

comme les classifieurs k-ppv [18], la théorie du floue [2, 15] et les modèles statistiques [11]. La tendance actuelle repose sur la fusion de plusieurs classifieurs différents [1, 4] ou sur la fusion d'un classifieur en-ligne et d'un classifieur hors-ligne [13, 14].

Les performances de ces systèmes ne sont pas pour l'instant suffisantes. En choisissant l'écriture scripte nous diminuons le problème de la segmentation. Nous proposons donc un système de reconnaissance traitant l'écriture scripte dynamique "libre" (Majuscules, minuscules, chiffres et ponctuations), la seule contrainte pour l'utilisateur est de séparer chaque lettre par un lever de stylo.

L'acquisition des textes est effectuée sur une tablette graphique pourvue d'un écran à cristaux liquides qui permet un retour visuel direct sous le stylo (émulation de l'encre réelle : encre électronique). De plus notre système est omni-scripteur et n'implique aucun apprentissage préalable de la part de l'utilisateur. Le moteur de reconnaissance utilise un lexique de 200 000 formes couvrant un large champ de la langue française.

La suite de cet article suit le plan suivant : dans la seconde partie nous détaillerons la structure générale de notre système pour nous concentrer dans la troisième partie, sur les différents détecteurs probabilistes utilisés. Le segmenteur sera détaillé dans la quatrième partie et enfin nous présenterons le moteur de fusion ainsi que les résultats dans la cinquième partie.

2 Système de lecture

2.1 Base de données

Notre système est entraîné et évalué à partir de deux bases équilibrées, l'une d'apprentissage, l'autre de test, qui contiennent chacune 45 textes, 31 scripteurs pour la base d'apprentissage et 25 pour la base de test, dont 18 scripteurs en commun. Au total il y a 90 textes et 40 scripteurs différents. Un étiquetage manuel de la base a été réalisé permettant les mesures de performances du système de lecture.

2.2 Structure du système

La structure générale est schématisée figure 1. Notre système de lecture se présente sous la forme d'une série d'experts de pré-traitement et de codage qui permettent d'extraire les informations de type topologique et géométrique du texte d'entrée. Ces experts, au nombre de trois, traitent respectivement la détection des lignes d'assises du texte, la caractérisation de la silhouette (qui représente la forme globale d'un mot en détectant les lettres ascendantes et descendantes) et la détection des espaces inter-mots (le blanc séparant deux mots). Ces experts fournissent des informations probabilistes au niveau *stroke* (plus petite entité qui correspond au trait séparé par deux levés de stylo) et non au niveau lettre car nous ne disposons pas encore de la segmentation du texte.

Comme tout système de lecture, le problème de la segmentation du texte en lettres et en mots est une tâche difficile et la contrainte de l'écriture scripte ne la simplifie pas beaucoup. En effet, un stroke ne peut pas contenir plusieurs lettres mais une lettre peut être composée de plusieurs strokes, l'écriture scripte est isolée, mais les lettres ne sont pas isolables. De plus l'écriture scripte entraîne une difficulté dans la différenciation des espaces séparant deux lettres d'un même mot (espace intra-mot) de ceux séparant deux mots (espace inter-mot). Notre expert neuronal de segmentation est entraîné à fournir le type d'espace caractérisant chaque blanc compris entre deux strokes consécutifs (*arc inter-stroke*) à partir des données de ces derniers et des informations issues du détecteur de lignes de base.

Ces probabilités sur la segmentations vont nous permettre de générer des hypothèses de segmentation du texte en lettres et en mots. Toutes les hypothèses ne sont évidemment pas explorées car le temps de calcul deviendrait prohibitif, seules les hypothèses les plus probables sont conservées, selon un modèle rigoureux.

Chaque lettre correspondant à une hypothèse de segmentation est envoyée au classifieur de caractère dynamique pour fournir un mot hypothèse. Chaque mot est corrigé par un lexique qui renvoie une liste de mots candidats or-

thographiquement justes. Une probabilité est calculée pour chaque mot candidat en fusionnant les informations issues du segmenteur, du classifieur et de la silhouette. Tous les mots candidats de toutes les hypothèses possibles constituent un nouveau treillis de mots que le moteur final va utiliser pour trouver la solution qui maximise la probabilité et fournir ainsi le texte ASCII.

3 Détecteurs

Ces experts donnent de manière probabiliste des informations sur la topologie et la géométrie des mots et des lettres. On définit deux lignes d'assises, la *ligne de base* qui correspondent à l'appui des lettres et la *ligne de corps* qui relie le sommet des petites lettres. L'expert de détection de lignes d'assises doit donner des informations sur l'emplacement de ces lignes les plus précises possibles car la caractérisation de la silhouette d'un mot ainsi que la segmentation du texte s'appuient sur ces résultats.
Nous allons maintenant détailler les trois experts de prétraitement.

3.1 Expert détecteur de lignes d'assises

La recherche des lignes de base et de corps d'un texte doit être la plus précise possible et il existe plusieurs techniques de détection [9, 10]. L'approximation de celle-ci par une ligne droite n'est pas suffisante car l'écriture manuscrite, même guidée par les lignes d'une feuille, présente de nombreuses oscillations. Ce phénomène est d'ailleurs accentué par la parallaxe due à l'utilisation d'un tablette graphique.

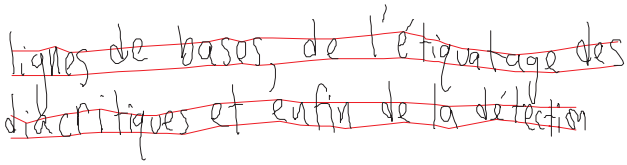

FIG. 2 – Exemple de détection des lignes de bases.

Pour réaliser cette tâche, cet expert recherche les strokes représentant des lettres médianes (*a, c, e, i, n...*). La hauteur moyenne des lettres médianes permet de sélectionner tous les strokes ayant cette hauteur. Un filtrage est effectué sur ces derniers pour supprimer les strokes parasites (accent, ponctuation etc...) afin de garder au final les lettres médianes. Les lignes de bases brisées (figure 2) sont obtenues en reliant les strokes restants. On relie le bas des boîtes englobantes (plus petit rectangle contenant un stroke) de ces derniers pour obtenir la ligne de base et le haut des boîtes pour obtenir la ligne de corps.
Il est difficile de juger la pertinence de cet expert, car nous ne disposons pas des lignes de bases réelles. Pour comparaison, nous avons utilisé les étiquettes de la base de données pour sélectionner les caractères médians et appliqué la même construction des lignes de bases. Cette comparaison nous a montré que l'expert donne des résultats extrêmement proches de ce dernier, les différences sont minimes et ne devraient entraîner que peu d'erreur.

3.2 Caractérisation de la silhouette

Le rôle de cet expert est de déterminer la forme, c'est-à-dire, la silhouette générale des mots en utilisant les lignes de bases et d'extraire ainsi des informations globales [15]. Cet expert estime pour chaque stroke s_i deux probabilités correspondants à la présence d'une hampe $p(h|s_i)$ et d'un jambage $p(j|s_i)$. De plus, à partir de ces deux informations, on attribue une classe d'activation Act_i parmi quatre :
– *médiane (m)* : stroke ne contenant ni hampe ni jambage ;
– *hampe (h)* : stroke comportant une hampe et pas de jambage ;
– *jambage (j)* : stroke possédant un jambage et pas de hampe ;
– *f* : stroke avec une hampe et un jambage.

Estimation des probabilités des hampes et jambages. L'estimation de ces probabilités est basée sur l'utilisation des lignes de bases calculées par l'expert précédent (figure 3). La position du bas d'une boîte englobante d'un stroke comparée à la position de la ligne de base permet d'obtenir la probabilité d'existence d'un jambage $p(j|s_i)$. De même, la comparaison entre le haut de la boîte englobante et la ligne de corps nous donne la probabilité d'avoir une hampe $p(h|s_i)$. Cette probabilité est proportionnelle à la hauteur du dépassement et est comprise entre 0 pour un stroke ne possédant ni hampe ni jambage et 1 pour le stroke ayant le plus grand dépassement dans la ligne de texte traité.

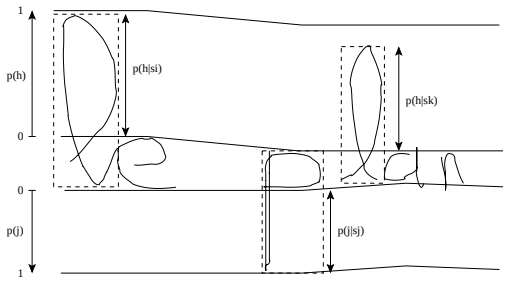

FIG. 3 – Estimation des probabilités d'existence des hampes et jambages.

Le cas des lignes de texte sans hampe ou jambage est également pris en compte. Si la plus haute hampe est inférieure à la moitié de la hauteur des lettres médianes, la ligne est considérée comme ne possédant pas de hampe, de même pour les jambages. Une ligne ne possédant ni hampe ni jambage est une ligne composée soit uniquement de lettres médianes, soit uniquement de majuscules et de chiffres.

Calcul de la classe d'activation. Le calcul de la classe d'activation est laissé à la charge d'un réseau de neurones de type MLP. Deux bases de données d'activation ont été extraites pour l'apprentissage et les tests du réseau.
Nous disposons en entrée des deux probabilités $p(h|s_i)$ et $p(j|s_i)$ et en sortie les quatre classes d'activation. La structure du réseau est donc un 2-X-4, avec la taille de la couche cachée à déterminer. Le problème du réseau est qu'il discrimine une classe parmi toutes les autres, on va donc trou-

ver des frontières entre les classes qui vont nous amener à une mauvaise généralisation de la classification. De manière à retrouver de bonnes frontières, on transforme ce problème 4 classes en 2 problèmes 2 classes. Le réseau va discriminer des doublets de classes ; $\{m, h\}$ contre $\{j, f\}$ et $\{m, j\}$ contre $\{h, f\}$. Pendant l'apprentissage, on active les deux sorties dont les doublets contiennent la classe désirée. Pour l'utilisation de ce réseau, l'intersection des deux doublets les plus actifs correspond à la classe d'activation. Par exemple, si les deux sorties actives du réseau sont $\{m, j\}$ et $\{m, h\}$ la classe d'activation correspondante est $Act_i = \{m, j\} \bigcap \{m, h\} = m$.
Nous pouvons introduire la notion de rejet dans le cas ou l'intersection est vide ($\{m, j\} \bigcap \{h, f\} = \emptyset$), c'est-à-dire, qu'aucune classe d'activation ne correspond. Dans le cas où il n'y a qu'une sortie active, l'intersection est un doublet ($\{h, f\} \bigcap \{h, f\} = \{h, f\}$), nous ne pouvons prendre de décision définitive, et les deux activations possibles sont conservées ($Act_i = \{h, f\}$). L'apprentissage du réseau n'a pas permis, lors des tests, d'avoir ces configurations de sorties. Il répond systématiquement par deux sorties actives avec une classe d'activation commune.
Les meilleurs résultats ont été obtenus avec une couche cachée de 3 cellules. Nous obtenons avec le réseau 2-3-4, un taux de bonne activation de 94,7 % sur la base de test. La matrice de confusion est donnée sur le tableau 1.

	Réel			
	m	h	j	f
m	98,1	1,7	0,2	0
h	11,5	88,3	0	0,2
j	18	1,8	78,7	1,5
f	14,1	41,3	0	44,6

TAB. 1 – Matrice de confusion.

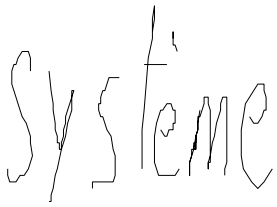

FIG. 4 – Exemple d'écriture, le 'y' est activé comme médiane.

On observe une confusion entre les *médianes* et les *jambages* (18 %) due au style de l'écriture comme on peut l'observer sur la figure 4. Ces confusions apparaissent lorsque les hampes et les jambages sont peu marqués. La confusion *f-h* de 41,3 % s'explique simplement par l'existence de la lettre '*f*' écrite sans jambage typique de l'écriture scripte. L'étiquetage de la base de textes BA_{texte} fournit toujours une classe d'activation f pour les 'f' du texte qu'ils soient écrit avec ou sans jambage. De même pour les 'z' qui peuvent ne pas contenir de jambage dans l'écriture scripte et qui sont tous étiquetés comme en possédant un (j). Cette confusion n'est donc pas représentative car il s'agit d'un manque de précision de l'étiquetage des bases de textes qui n'influencera pas les résultats du système par la suite. Le taux de bonne activation est donc légèrement supérieur à 94.7 %.
La classe d'activation étant obtenue à partir des probabilités d'existence des dépassements issues elles même de l'estimation des lignes de bases, ces résultats remettent en cause le bon calcul des lignes de bases de l'expert précédent. Un apprentissage a également été effectué en utilisant les lignes de bases réelles et le taux obtenu est à peine meilleur (95,2 %). Ces résultats montrent donc finalement que l'estimation des lignes de bases est correcte. Les confusions viennent des caractéristiques de l'écriture des scripteurs eux-mêmes (figure 4), les lettres à hampes et jambages ont la même hauteur que les lettres médianes. Ceci milite en faveur d'un système bouclé où toute décision peut-être remise en cause.

3.3 Détecteur d'espace

Cet expert calcule pour chaque arc inter-stroke a_i une probabilité $p(e|a_i)$ d'être un espace inter-mot. La répartition des distances séparant les bords de deux boîtes englobantes peut être modélisée par un mélange de deux gaussiennes (figure 5). Le premier mode correspond aux arcs séparant des strokes d'un même mot (espace intra-mot) et le second représente les arcs séparant deux mots (espace inter-mot).

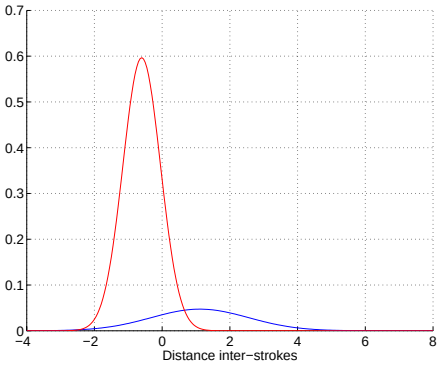

FIG. 5 – Répartition des distances inter-strokes.

La première approche utilise la classification Bayésienne pour estimer les deux gaussiennes sur la base d'apprentissage et minimiser la probabilité d'erreur lors de la classification. Cette méthode donne un taux de bonnes classifications de 94 %. Pour améliorer ce résultat, nous utilisons un réseau de neurones pour déterminer un meilleur seuil de décision. Après apprentissage de ce réseau (qui revient à faire une minimisation par les moindres carrés), le taux de bonnes classifications est de 95,4 %. Pour avoir une probabilité en sortie de ce réseau, la cellule de sortie utilise une fonction de transfert linéaire saturée entre 0 et 1. La matrice de confusion de cet expert est donnée tableau 2.
Nous avons remarqué qu'un correcteur lexical est plus à

	Réel	
	intra-mot	inter-mot
intra-mot	13968	195
inter-mot	586	2049

TAB. 2 – Matrice de confusion du détecteur d'espace (nombre d'espaces).

même de détecter des erreurs de sous-segmentation (omission d'un inter-mot) que de sur-segmentation (insertion d'un inter-mot). Une analyse de notre lexique a montré que la probabilité qu'une partie d'un mot soit également un mot réel est d'environ 0.6. Dans le même temps, la probabilité que la fusion de deux mots existants soit également un mot du lexique n'est que de 0.02. Ces statistiques dépendent beaucoup de la longueur des mots et sont données ici à titre indicatif, mais montrent très clairement le rapport qui existe entre ces deux types d'erreur. On remarque d'ailleurs la très bonne reconnaissance des intra-mots de cet expert qui va être d'une très grande utilité pour le fonctionnement du lexique.

4 Segmentation

Les levés de stylo (espaces inter-strokes) tout comme les strokes, contiennent une grande quantité d'informations. Comme [7] on utilise un réseau de neurones pour classifier ces arcs inter-strokes. A partir des données d'entrée, on construit un graphe non orienté $G = (S, A)$ avec un ensemble de sommets constitués des strokes $S = \{s_1, s_2, \ldots, s_n\}$ et des arêtes $A = \{(s_1, s_2), (s_2, s_3), \ldots, (s_{n-1}, s_n)\}$ correspondant aux arcs reliant deux strokes consécutifs. On distingue cinq classes d'arcs (fig 6) qui ont été définies comme ci-dessous :

- Arc intra-lettre (A_1) : Arc séparant deux strokes appartenant à la même lettre ;
- Arc inter-lettre & intra-mot (A_2) : Arc entre deux strokes d'un même mot mais de lettres différentes ;
- Arc inter-mot (A_3) : Arc séparant deux mots d'une même ligne ;
- Arc diacritique (A_4) : Arc pointant vers un stroke accent ou diacritique ;
- Arc ponctuation (A_5) : Arc pointant vers une ponctuation.

On considère le stroke, "*barre de t*", comme faisant partie de la lettre et non comme diacritique. Ce codage permet d'avoir, pour un texte donné, un unique étiquetage des arcs. Les paramètres servant à caractériser les arcs sont issus des coordonnées de deux strokes consécutifs s_n et s_{n+1}, de leurs boîtes englobantes, de leurs positions par rapport aux lignes de bases et du lever de stylo. Le lever de stylo est défini comme le segment joignant le dernier point de s_n au premier point de s_{n+1} (fig 7). L'idéal étant de trouver des paramètres invariants aux scripteurs (paramètres omni-scripteurs) [3]. Les 32 paramètres sont les suivants :

- **(1, 2)** $d_{Gx},\ d_{Gy}$: Projections horizontale et verticale de la distance entre les centres de gravité d_G des strokes ;

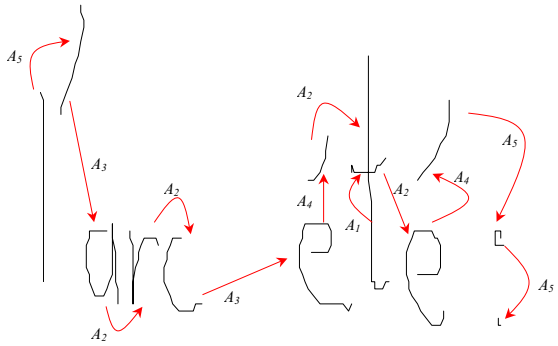

FIG. 6 – Exemple d'étiquetage d'arc d'un texte.

- **(3)** d_{min} : Plus petite distance séparant les deux strokes. La plus grande distance d_{max} ne fait pas directement partie des paramètres ;
- **(4, 5)** $d_{xmin},\ d_{ymin}$: Projections horizontale et verticale de la distance minimum d_{min} ;
- **(6, 7)** $d_x,\ d_y$: Projections horizontale et verticale du lever de stylo ;
- **(8)** Θ : Angle du lever de stylo par rapport à l'horizontale ;
- **(9, 10)** $d_{Gmin},\ d_{Gmax}$: Paramètres qui caractérisent la distance des centres de gravité par rapport aux distances d_{min} et d_{max} : $d_{Gmin} = \sqrt[3]{d_G^2 d_{min}}$ et $d_{Gmax} = \sqrt[3]{d_G^2 d_{max}}$;
- **(11)** r : Rapport d_{min}/d_{max} ;
- **(12)** $larg$: Largeur du recouvrement normalisée par la largeur des deux boîtes englobantes fusionnées ;
- **(13)** Δ : Valeur absolue de la différence de largeur de deux boîtes ;
- **(14, 15)** $Nb_1,\ Nb_2$: Nombre de points des deux strokes ;
- **(16, 17)** $S_1,\ S_2$: Surfaces des boîtes englobantes ;
- **(18, 19)** $P_1,\ P_2$: Périmètres des boîtes englobantes ;
- **(20, 21)** $L_1,\ L_2$: Longueurs des deux strokes ;
- **(22, 23)** $M_{1x},\ M_{1y}$: Projection horizontale et verticale de l'axe principal d'inertie du stroke s_n ;
- **(24, 25)** $M_{2x},\ M_{2y}$: Projections horizontale et verticale de l'axe principal d'inertie pour le stroke s_{n+1} ;
- **(26, 27)** $h_1,\ h_2$: Hauteurs du centre de gravité des strokes par rapport à la ligne de base ;
- **(28, 29)** $h_3,\ h_4$: Distance du bas de la boite englobante par rapport à la ligne de base ;
- **(30, 31)** $h_5,\ h_6$: Distance du haut de la boite englobante par rapport à la ligne de corps ;
- **(32)** h : Hauteur du centre du lever de stylo par rapport à la ligne de base.

4.1 Base de données

Pour la classification, deux bases d'arcs ont été formées à partir des textes étiquetés. Une base d'apprentissage comprenant 16642 arcs et une base de test de 17020 arcs. La répartition de chacune des classes est donnée tableau 3.
On constate que les arcs A_4 et A_5 sont très faiblement représentés, la ponctuation étant sous-représentée par rapport

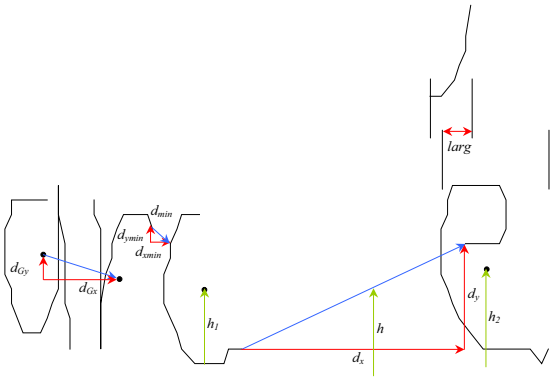

FIG. 7 – Paramètres de distance.

Base	Arcs	A_1	A_2	A_3	A_4	A_5
App.	16642	1806	10982	2296	1153	405
	%	10.9	66.0	13.8	6.9	2.4
Test	17020	1814	11515	2301	1038	353
	%	10.7	67.6	13.5	6.1	2.1

TAB. 3 – Répartition des types d'arc.

aux lettres. De plus, nos textes sont en français, augmentant de manière non négligeable la classe A_4 qui englobe en plus des diacritiques (point sur les i et j) les accents qui n'existent pas en langue anglaise. Cette base étant déséquilibrée au niveau de la répartition des classes, elle est difficile à apprendre : un classifieur répondant toujours A_2 aurait un taux de classification de 67.6 % et un classifieur ne travaillant que sur les 4 premières classes peut déjà atteindre 97.9 %.

4.2 Structure du segmenteur

Trois classifieurs ont été étudiés et comparés pour réaliser cette tâche de segmentation. Le premier de type probabiliste minimise l'erreur de classification (Bayésien), le second utilise un classifieur de type k-ppv et enfin le dernier utilise un réseau de neurones de type MLP. Ce dernier à été retenus par sa capacité à très bien généraliser l'étape de la segmentation sur la base de test. Ce classifieur utilise en entrée les 32 paramètres qui ont été préalablement normalisés (centrés et réduits).

Ce classifieur possède 32 neurones d'entrée, 20 neurones sur la couche cachée et 5 en sortie pour les 5 types d'arcs. Plusieurs apprentissages ont été nécessaires pour obtenir la taille de la couche cachée de manière optimale. Un apprentissage de 200 itérations par une descente du gradient total du deuxième ordre par l'algorithme de Levenberg-Marquardt et cross-validation donne un taux de bonnes segmentation sur la base d'apprentissage de 92.2 % (tableau 4). La figure 8 compare le taux de généralisation en fonction des n meilleurs résultats (top_n) sur la base de test des trois classifieurs testés.

Les meilleures performances sont atteintes par le classifieur *k-ppv* mais son utilisation est assez difficile car il utilise une grande base de données de référence et les temps de calculs sont assez longs. Nous avons donc décidé de retenir la version neuronale du classifieur. Très peu encombrant en place mémoire et très rapide (100 fois plus rapide que le *k-ppv*), il a su le mieux généraliser le problème de la segmentation. En analysant l'évolution du taux de reconnaissance en fonction des n meilleurs résultats, on constate qu'en conservant les deux meilleurs résultats de segmentation (top_2) le classifieur à MLP dépasse le classifieur *k-ppv*. Ainsi le classifieur à MLP présente un taux de reconnaissance de 99.1 % en top_2.

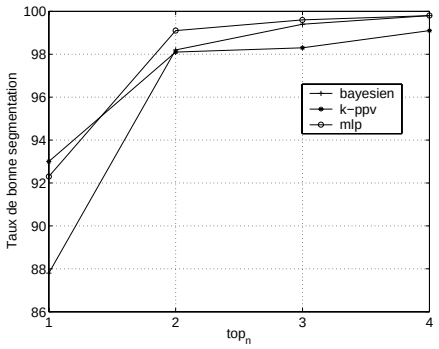

FIG. 8 – Taux de bonnes segmentation.

	Réel				
	A_1	A_2	A_3	A_4	A_5
A_1	84.8	14.7	0.1	0.3	0.1
A_2	0.7	98.0	1.1	0.1	0.1
A_3	0	27.4	72.4	0.1	1.1
A_4	1.1	0.8	0.1	97.5	0.5
A_5	2.6	22.5	6.0	7.1	61.8

TAB. 4 – Matrice de confusion du MLP.

La classe A_2 étant très fortement représentée, elle a tendance a jouer un rôle d'attracteur pour les autres classes. Les sorties y_i du réseau de neurones sont comprises entre [-1,1] et sont indépendantes les unes des autres. Pour avoir des probabilités d'appartenance aux 5 classes d'arcs, on utilise la règle du *softmax* en sortie :

$$p_i = \frac{e^{-\alpha y_i}}{\sum_j e^{-\alpha y_j}}$$

de cette manière on respecte bien les règles imposées par les probabilités, les sorties $p_i \in [0,1]$ et $\sum p_i = 1$.

5 Moteur

Le coeur de notre système de lecture est basé sur la fusion d'informations de tous les experts vu précédemment. Ce moteur débute par la génération d'un treillis d'hypothèses de segmentation. On calcul pour chacune de ces hypothèses une probabilité à partir des différents experts et du classifieur de caractères. Une recherche par programmation dy-

namique de ce treillis permet de retrouver la meilleur transcription possible de la ligne.

5.1 Génération du treillis

D'après le comportement du segmenteur (figure 8) on doit conserver plusieurs classes pour chaque arc pour obtenir un meilleur taux de bonne segmentation et ainsi construire un treillis hypothèse de segmentation. Pour des raisons évidentes de temps de calcul, il est impossible de tester toutes les segmentations possibles. Notre générateur de treillis va donc utiliser les probabilités de sortie du segmenteur ainsi que sa matrice de confusion pour réduire le nombre de classes possibles attribuées aux arcs. De plus, la fusion du segmenteur et du détecteur d'espace permet d'isoler quelques espaces inter-mot sûrs et ainsi de fixer des points d'ancrages dans la ligne.

Les règles actuelles de génération du treillis, encore en phase de recherche, sont très simples et ne fonctionnent pas dans le cadre omni-scripteur. Un arc A est un point d'ancrage si la probabilité de la classe A_3 du segmenteur est supérieur à 0.7 et la probabilité du détecteur d'espace supérieur à 0.7. Les classes attribuées aux arcs sont définies pas le top_1 du segmenteur et suivent les règles suivantes :

$$\begin{array}{lll} A = \{A_1, A_2\} & , & top_1 = A_1 \\ A = \{A_2\} & , & top_1 = A_2 \ \& \ p_{top_1} \geq 0.9 \\ A = \{A_2, A_3\} & , & top_1 = A_2 \ \& \ p_{top_1} < 0.9 \\ A = \{A_3, A_2\} & , & top_1 = A_3 \\ A = \{A_4, A_5, A_2\}, & & top_1 = A_4 \\ A = \{A_5, A_4, A_2\}, & & top_1 = A_5 \end{array}$$

Ces règles sont déterminées à partir de la matrice de confusion du segmenteur (tableau 4) et de la probabilité *a priori* d'apparition des différentes classes (tableau 3). Il faut générer un maximum de singletons A_2 pour diminuer la combinatoire. Un seuil de 0.9 sur la probabilité d'appartenance à cette classe s'est avéré dans un premier temps très efficace. Les autres classes sont triées par ordre de probabilité décroissante.

5.2 Recherche lexicale

Chaque hypothèse de segmentation regroupe donc notre flux de strokes en caractères et en mots. Toutes les lettres ainsi formés sont envoyés à un classifieur de caractères qui va renvoyer les probabilités d'appartenance aux 62 lettres possibles. Le classifieur est de type *k-ppv* et utilise une base de prototypes qui ont été déterminés par un algorithme de clustering original décrit dans [16] et testé sur les symboles de la base de donnée UNIPEN corpus *Train R01-V07*.

On peut donc maintenant générer une première hypothèse de phrase, en conservant pour chacun des caractères classifiés, le caractère le plus probable et qui correspond à la même classe d'activation que celle fournit par le détecteur de silhouette.

Chaque mot de cette hypothèse est corrigé à l'aide du lexique. Comme toutes les hypothèses de segmentation du treillis vont être explorées et que la "bonne" segmentation en lettres et en caractères devrait apparaître, il est inutile de faire une recherche lexicale basée sur la distance d'édition en autorisant un nombre de lettres différents. De cette manière, la recherche lexicale est très rapide, puisqu'il s'agit simplement d'organiser le lexique en fonction de la longueur des mots. Notre recherche lexicale retourne une liste de mots candidats qui comporte au plus n_{err} lettres de différence. Ce paramètre n_{err} est déduit du nombre de caractères n_{car} du mot d'origine et suit les relations suivantes :

$$\begin{cases} n_{err} = 1 & , \quad n_{car} \leq 2 \\ n_{err} = 2 & , \quad n_{car} = 3 \\ n_{err} = n_{car}/2, & n_{car} > 3 \end{cases}$$

La probabilité de chaque mot candidat est calculée en fusionnant les probabilités des différents experts. Nous utilisons la transformation classique du produit de probabilités en somme de logarithmes pour la fusion. Ainsi la probabilité d'un mot est la somme des probabilités suivantes :

– La probabilité de la segmentation $p_{segment}$ utilise directement la sortie du segmenteur p_{arc} après transformation par *softmax* :

$$p_{segment} = \sum_{arc} \log p_{arc}$$

– La probabilité du classifieur $p_{classif}$ issue des lettres du mot candidat. La probabilité p^i_{car} de la i^{eme} lettre, correspond à la probabilité renvoyée par le classifieur de caractères pour la lettre proposée par le lexique :

$$p_{classif} = \sum_{i} \log p^i_{car}$$

– La probabilité de la silhouette p_{silh} issue des probabilités des hampes et des jambages données par l'expert détaillé au §3.2. Ainsi, un caractère à hampe se voit attribuer la probabilité $p_h = p(h|s_i)$ et un caractère sans hampe la probabilité $p_h = 1 - p(h|s_i)$. De la même manière pour les caractère à jambage, avec toutefois une légère modification pour les lettres f et z qui peuvent s'écrire avec ou sans jambage et qui se voit alors attribuer la plus grande probabilité : $p_j = \max(p(j|s_i), 1 - p(j - s_i))$.

$$p_{silh} = \sum_{i} \log p^i_h + \sum_{i} \log p^i_j$$

Finalement la probabilité d'un mot est définie par :

$$p_{mot} = p_{segment} + p_{classif} + p_{silh}$$

Cette fusion d'informations permet de tenir compte de toutes les sources mises à notre disposition ainsi que de leur importance. Les mots qui ne respectent pas exactement la silhouette déduite du contexte peuvent être conservés si le résultat de la classification en lettres est bon. Inversement, un mot bien reconnu par le classifieur mais qui ne respecte pas la silhouette se verra attribuer une probabilité plus faible qu'un mot proche de la silhouette. Ceci

est très important, car nous somme dans un contexte omni-scripteur, et les caractéristiques de l'écriture peuvent entraîner de nombreux *artefact* aussi bien du point de vue géométrique que du point de vue morphologique.

5.3 Exploration du treillis

Les mots candidats de chaque hypothèse de segmentation se voient attribuer une probabilité p_{mot} définit plus haut. Tous ces mots sont regroupés dans un seul nouveau treillis de mots qui est constitué d'une liste de mots avec les positions dans la phrase et les probabilités associées.
La particularité de notre exploration du treillis est de ne pas simplement choisir la meilleure hypothèse de segmentation, mais de fournir la meilleure segmentation possible en combinant plusieurs hypothèses. Pour cela, la recherche par programmation dynamique s'effectue dans tout le treillis de mots.

5.4 Résultats

Ce moteur étant encore au stade de recherche, les résultats présentés ici sont donnés à titre indicatif.
Les premiers résultats concernent la recherche lexicale ainsi que l'exploration du treillis de mot en fixant artificiellement la segmentation réelle du texte (il n'y a pas de génération de treillis d'hypothèses de segmentation). Sur 40 textes exploitables de la base de test, on obtient un taux d'erreur sur les mots de 21.5 %. Sur ces 21.5 % d'erreur, plus de 8 % sont sur les mots de moins de 3 lettres, ceci s'explique par l'utilisation du lexique qui ne peut pas faire la différence sur les mots courts (par exemple entre *le* et *la* ou entre *l'* et *d'*). De nouvelles recherches effectuées en parallèle sur l'adaptation au scripteur [17] et l'auto-similarité ont montré que l'on pouvait diminuer grandement ce taux d'erreur. La répartition des erreurs en fonction de la longueur des mots (figure 9) montre clairement que l'efficacité du lexique dépend de la longueur des mots.

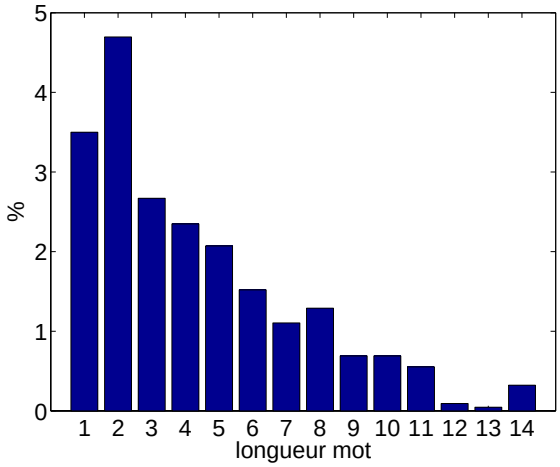

FIG. 9 – Répartition des erreurs de reconnaissance mot.

Pour la partie segmentation et génération du treillis, les résultats ne sont valables que sur 3 scripteurs différents (5 textes). L'expert de génération du treillis de segmentation émet en moyenne 110 hypothèses par ligne. Le moteur d'exploration du treillis retrouve systématiquement la bonne segmentation. Ces très bon résultats sont à généraliser sur le reste de la base de données.

Conclusions & Perspectives

Ce moteur de lecture automatique de texte s'avère très prometteur, les premiers résultats sont très encourageants et indiquent que l'on peut encore les améliorer. L'introduction de points d'ancrages plus robustes et plus nombreux ainsi que l'utilisation complète des informations contenus dans la matrice de confusion du segmenteur nous ont permis de conserver les bon résultats du segmenteur multi-scripteur présenté ici tout en ayant un système omni-scripteur. La modification du calcul de probabilité des mots de la fusion d'informations et l'introduction de connaissances supplémentaires dans le système par l'intermédiaire de nouveaux experts (comme un détecteur de diacritiques et de ponctuations) permettrais également d'augmenter le taux de reconnaissance global. Les recherches parallèles effectuées sur l'adaptation au scripteur de ce système de lecture ont permis une augmentation de plus de 35 % du taux de reconnaissance.

Références

[1] F. Alimoğlu and E. Alpaydin. Methods of combining multiple clasifiers based on different representations for pen-based handwritten digit recognition. *5th Turkish Artificial Intelligence and Artificial Neural Networks Symposium, TAINN*, 1, 1996.

[2] E. Anquetil. *Modélisation et reconnaissance par logique floue : Application à la lecture en-ligne de l'écriture manuscrite omin-scripteur*. PhD thesis, Université de Rennes I, 1997.

[3] H. S.M. Beigi. Processing, modeling and parameter estimation of the dynamic on-line handwriting signal. *Proceedings World Congress on Automation*, pages 27–30, 1996.

[4] A. Bellili, M. Gilloux, and P. Gallinari. An hybrid MLP-SVM handwritten digit recognizer. *International Conference on Document Analysis and Recognition, ICDAR01*, pages 28–31, 2001.

[5] C. Choisy and A. Belaïd. Coupling of a local vision by markov field and a global vision by neural network for the recognition of handwritten words. *International Conference on Document Analysis and Recognition, ICDAR'03*, 2 :849–853, 2003.

[6] L. Duneau. *Etude et réalisation d'un système adaptatif pour la reconnaissance en ligne de mots manuscrits*. PhD thesis, Université Technologique de Compiègne, 1994.

[7] P. D. Gader, M. Mohamed, and J.-H. Chiang. Handwritten word recognition with character and inter-

character neural networks. *IEEE Transactions on Systems, Man, and Cybernetics*, 27(1) :158–164, 1997.

[8] A. Henning and N. Sherkat. Cursive script recognition using wildcards and multiple experts. *Pattern Analysis & Applications*, 4 :51–60, 2001.

[9] A. Henning, N. Sherkat, and R.J. Whitrow. Zone-estimation for multiple lines of handwriting using approximating spline functions. *5th International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pages 325–328, 1996.

[10] O. Lavialle, X. Molines, F. Angella, and P. Baylou. Réseau de contours actifs et systèmes particulaire : application au redressement de textes incurvés. *GRETSI'01*, 2001.

[11] M. Leroux. *Reconnaissance de textes manuscrits à vocabulaire limité avec application à la lecture automatique des chèques*. PhD thesis, Université de Rouen, 1991.

[12] M. Morita, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Unsupervised feature selection using multi-objective genetic algorithms for handwritten word recognition. *International Conference on Document Analysis and Recognition, ICDAR'03*, 2 :666–670, 2003.

[13] L. Oudot and L. Prevost. Techniques de coopération pour la reconnaissance d'écriture en contexte. *Extraction des Connaissances et Apprentissage, ECA'02*, 2002.

[14] E. Poisson, C. Viard-gaudin, and P.-M. Lllican. Combinaison de réseaux de neurones à convolution pour la reconnaissance de caractères manuscrits en-ligne. *Colloque International Francophone sur l'Ecrit et le Document, CIFED'02*, pages 315–324, 2002.

[15] R. K. Powalka, N. Sherkat, and R. J. Whitrow. Word shape analysis for a hybrid recognition system. *Pattern Recognition*, 30(3) :421–445, 1997.

[16] L. Prevost and M. Milgram. Modelizing character allographs in omni-scriptor frame : a new non-supervised algorithm. *Pattern Recognition Letters*, 21(4) :295–302, 2000.

[17] L. Prevost, L. Oudot, A. Moises, and M. Milgram. Reconnaissance de textes manuscrits dynamiques : Mise en œuvre de concepts perceptifs pour l'adaptation à l'utilisateur. *soumis à RFIA*, 2004.

[18] V. Vuori, J. Laaksonen, and E. Oja. On-line adaptation in recognition of handwritten alphanumeric charaters. *International Conference on Document Analysis and Recognition, ICDAR'99*, pages 792–795, 1999.