COMBINAISON D'ALGORITHMES POUR LA RECONNAISSANCE DES CHIFFRES ET DES LETTRES BATONS DANS UN ENVIRONNEMENT MULTISCRIPTEUR D'ECRITURE COURANTE MIXTE

H. Oulhadj, J. Lemoine, E. Petit, H. Wehbi
Laboratoire d'Etude et de **R**echerche en **I**nstrumentation, **S**ignaux et **S**ystèmes
Université Paris XII – Val de Marne
UFR de Sciences et Technologie
61, Av. du Général de Gaulle
94010 CRETEIL CEDEX – France
Tél. : 01.45.17.14.97 – Télécopie : 01.45.17.14.92 – Email : OULHADJ@univ-paris12.fr

RESUME
La reconnaissance automatique de l'écriture naturelle est une opération particulièrement complexe et ardue. A ce jour, il n'existe aucun algorithme capable de traiter de façon fiable l'intégralité des tracés qui la constituent : un texte peut contenir à la fois des chiffres, des lettres minuscules, des majuscules et des caractères bâtons. Par conséquent, il est capital de savoir extraire et séparer ces différentes entités afin de les traiter grâce aux algorithmes spécifiques à chacune des catégories d'écriture qu'ils constituent. Ce papier décrit un système en-ligne capable de réaliser cette opération grâce à la segmentation-reconnaissance des chiffres et des lettres bâtons dans un environnement multiscripteur d'écriture mixte. Pour minimiser les risques d'erreur de localisation de ces tracés, la segmentation et la reconnaissance ne sont pas séparées mais coopèrent en interagissant dans une opération unique à mesure que le scripteur écrit. Une fois les caractères alphanumériques sont segmentés et reconnus, les parties cursives restantes peuvent être facilement localisées et par suite traitées grâce aux algorithmes spécifiques à ce type de tracé.

MOTS CLES :
Ecriture mixte, caractères alphanumériques, segmentation, coopération, reconnaissance en-ligne multiscripteur, combinaison d'algorithmes, classifieur.

ABSTRACT
The automatic recognition of natural handwriting is a particulary complex and arduous operation. Today, there is no algorithm able to recognize successfully all the characters that can be met in it : a text usually contains both numbers, baton characters, lowercase and capital letters. Therefore, we have to know how to extract and separate all these tracings in order to process them with specific algorithms to their categories handwriting. This paper describes an on-line system wich provides a segmentation module able to separate alphanumerical characters from lowercase letters in natural handwriting environment. In order to reduce the segmentation errors, the recognition is not separated from the segmentation but cooperate togother in a single operation as the writer writes. Once alphanumerical characters are segmented and recognised, cursive parts can be easily localised and as result processed with appropriate algorithms for them.

KEYWORDS
Mixed handwriting, alphanumerical characters, segmentation, cooperation, multiscripter on-line recognition, algorithms combination, classifier.

1 INTRODUCTION
La reconnaissance automatique de l'écriture manuscrite représente un vaste champ d'étude dont les premiers travaux datent de plusieurs dizaines d'années [4, 8, 14, 16]. Aujourd'hui, malgré les nombreuses années d'investigation consacrées au sujet, il n'existe toujours pas de système fiable capable de traiter l'écriture naturelle dans sa globalité. En effet, les résultats publiés montrent que les taux de reconnaissance obtenus sont restreints à des domaines d'application bien limités (adresses postales, chèques bancaires) ou à des catégories d'écriture très contraintes ne représentant qu'un aspect particulier de l'écriture courante et spontanée [12, 15, 23, 24]. Le travail présenté ici entre dans le cadre d'un programme d'étude visant à terme l'élimination de la majorité des contraintes imposées afin d'aborder l'écriture naturelle dans sa globalité [18, 19, 25]. Nous présentons dans cette article les parties réservées à la segmentation et à la reconnaissance en-ligne des chiffres et des lettres bâtons dans un environnement multiscripteur d'écriture mixte. Cette étape préalable, qui est non triviale par suite des difficultés sous-jacentes, s'avère tout à fait indispensable avant d'aborder la reconnaissance du mot dans sa globalité. En effet, une fois les caractères bâtons sont segmentés et

reconnus, les parties cursives restantes peuvent être facilement localisées et par suite traitées grâce aux algorithmes spécifiques à ce type de tracé. L'architecture générale du système est décrite dans la figure 1.

- en mode continu : le système ne fait qu'apprendre des caractères sous le contrôle d'un expert,
- en mode alterné : le système, en phase de reconnaissance, peut manquer un caractère, l'utilisateur peut alors formuler une demande d'apprentissage de ce caractère.

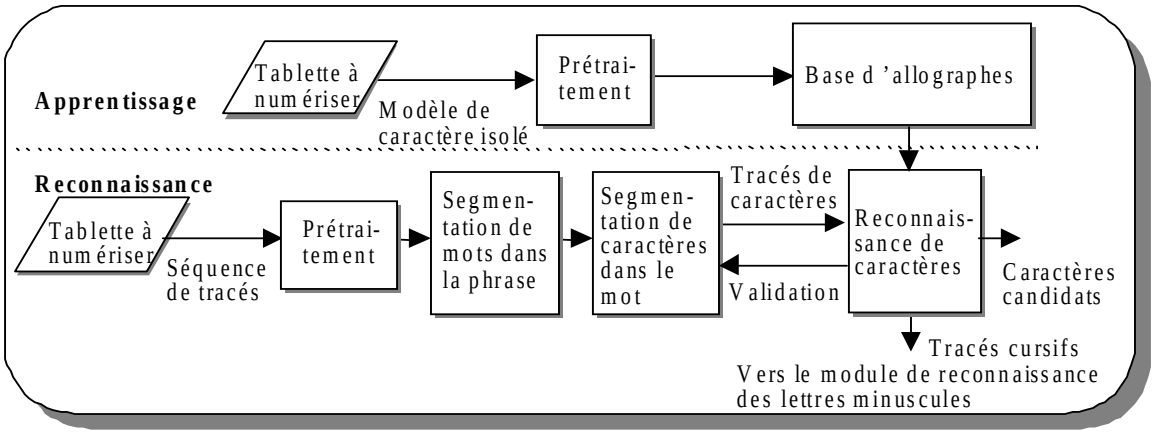

Fig. 1 : Synoptique général du système

Schématiquement, cinq parties fonctionnelles peuvent être distinguées : acquisition, prétraitement, apprentissage, segmentation et reconnaissance. Le fonctionnement détaillé de chacune de celles-ci est décrit dans les paragraphes qui suivent ici.

2 ACQUISITION / PRETRAITEMENT

L'acquisition en-ligne des tracés est faite à l'aide d'une tablette à numériser (OCE GRAPHICS - OG6461). Ces tracés sont en général bruités. Leur déformations peuvent avoir plusieurs origines possibles [2, 18, 20]. Les plus fréquentes sont dues aux tremblements de la main ainsi qu'au bruit de discrétisation généré par l'unité d'acquisition. Le rôle du prétraitement consiste alors à lisser le tracé d'entrée et à réduire le nombre de points échantillonnés grâce à des filtrages en distance et angulaire [3, 18, 25].

3 APPRENTISSAGE

Fonctionnellement, deux phases d'apprentissage peuvent être distinguées. La première permet d'initialiser le système par un expert, la seconde d'enrichir les bases obtenues afin d'améliorer la reconnaissance et assurer un fonctionnement multiscripteur. On parle alors d'apprentissage d'initialisation et d'apprentissage d'enrichissement (ou permanent). L'apprentissage d'initialisation porte sur des modèles de caractères isolés supposés parfaits. Une fois le tracé du caractère d'entrée est filtré, il est codé puis enregistré dans la base d'allographes appropriée (une base par méthode de codage). A l'issue de l'apprentissage d'initialisation, les bases obtenues peuvent être enrichies par un expert ou un scripteur quelconque (base multiscripteur) en les complétant par de nouvelles représentations de lettres plus ou moins dégradées. Ces mises à jour, qualifiées d'apprentissage permanent, peuvent être effectuées de deux façon différentes :

4 SEGMENTATION

4.1 Principe général

Deux niveaux de segmentation sont distingués. Le premier permet d'isoler les mots dans la phrase à mesure que le scripteur écrit [18, 25], le second de localiser à leur tour les caractères dans chacun des mots isolés [25]. Chaque caractère segmenté est immédiatement validé par la reconnaissance. Les parties de mot non reconnues sont alors considérées comme étant formées de lettres cursives et envoyées au module spécialisé dans la reconnaissance de ce type de tracé [25].

4.2 Segmentation des mots dans la phrase

La majorité des études consacrées à la segmentation des mots dans la phrase reposent sur l'hypothèse d'existence d'un espace inter-mots toujours supérieur à l'espace inter-lettres ou à la largeur moyenne d'une lettre [1, 9, 13, 21] . Les erreurs de segmentation générées sont dans la plupart des cas difficiles à éviter car l'espace inter-lettres, nécessaire à l'établissement de l'espace inter-mots, est évalué avant même que les lettres ne soient reconnues. Pour réduire les erreurs ainsi générées, nous introduisons un pas d'espacement adaptatif fondé essentiellement sur la connaissance des extrema des tracés indépendamment des lettres qu'ils peuvent représenter. Ce pas est calculé comme suit :

$$PAS = \frac{\sum_{i=1}^{n} d_{\min_i} + \sum_{j=1}^{m} d_{\max_j}}{n+m}$$

où : d_{min} = distance entre deux minima successifs
d_{max} = distance entre deux maxima successifs
n = nombre d'intervalles (ou distances) formés par les minima
m = nombre d'intervalles formés par les extrema

Nous tolérons sur ce pas une marge d'erreur (Ecart) égale à l'écart type des distances entre maxima (d_{max}) et entre minima (d_{min}) :

$$Ecart = \sqrt{\frac{\sum_{i=1}^{n}\left(PAS - d_{\min i}\right)^2 + \sum_{j=1}^{m}\left(PAS - d_{\max j}\right)^2}{n+m}}$$

Compte tenu des quantités PAS, DIT (distance entre deux tracés successifs) et de la probabilité d'erreur d'appartenance de deux tracés à un même mot, trois situations peuvent se présenter :

- *DIT < PAS => Perr = 1/3 => regroupement des tracés*
- *PAS ≤ DIT ≤ PAS+Ecart => Perr = 2/3 => regroupement des tracés*
- *PAS+Ecart < DIT => Perr = 3/3 => séparation des tracés*

Ces probabilités d'erreur ne représentent que des indications permettant de guider la reconnaissance. En cas d'échec de celle-ci, des opérations de regroupement ou de séparation des tracés peuvent être envisagées :

- *Perr = 1/3 => séparation des tracés*
- *Perr = 2/3 => séparation des tracés*
- *Perr = 3/3 => regroupement des tracés*

Telle que décrite ici, cette méthode de segmentation peut être sujette à des erreurs ayant pour origine l'inclinaison des lettres dans le mot ou la présence des signes diacritiques mal tracés [18, 25]. Nous traitons ce problème en évaluant l'espace inter-mots essentiellement dans la zone moyenne d'écriture et en éliminant complètement les signes diacritiques de l'opération de segmentation une fois que ces derniers sont identifiés et localisés [25].

Tant que i ≤ n faire
- encadrer T et S_i de deux rectangles R_1 et R_2
- si largeur $R_1 \cap R_2 \geq$ 1/4 Min [largeur R_1, largeur R_2]
  alors T = T ∪ S_i
  sinon si hauteur $R_1 \geq$ 60% (largeur zone moyenne d'écriture + largeur zone supérieure)
  alors valider la segmentation de T par la reconnaissance
  finsi
  T = S_i
  finsi
- i=i+1

fin tant que fin

On peut remarquer que la segmentation de chaque caractère est immédiatement validée par la reconnaissance. Cependant, des erreurs de sur-segmentation peuvent être générées lorsque des caractères formés de segments séparés, se chevauchant faiblement ou complètement pas, sont rencontrés. On y remédie en parachevant le processus de segmentation-reconnaissance avec la deuxième passe. Celle-ci consiste à invoquer une base de règles d'association de caractères et/ou de pseudo-caractères (composantes d'un caractère) afin d'identifier les caractères sur-segmentés (exemples : I + 3 => B, V + V => W, \ + / => V). A l'issue de chaque passe, seuls les caractères reconnus avec un score supérieur à un seuil S_{max} fixé expérimentalement sont retenus. Les parties de mot non reconnues sont alors considérées comme étant formées de lettres cursives et envoyées au module spécialisé dans la reconnaissance de ce type de tracé.

5 RECONNAISSANCE DES CARACTERES SEGMENTES

5.1 Concepts généraux

La figure 2 illustre le fonctionnement général de la reconnaissance.

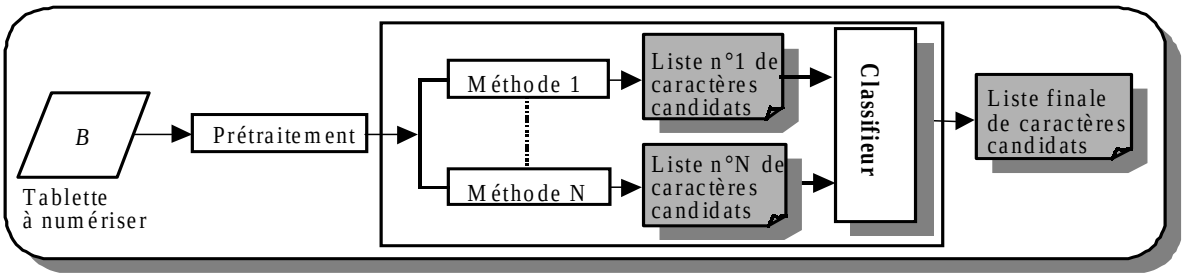

Fig. 2 : Principe de reconnaissance des caractères

4.3 Segmentation des chiffres et des lettres bâtons

La segmentation des caractères est activée automatiquement dès que le tracé d'un mot est isolé. L'algorithme s'exécute en deux passes. La première procède comme suit :

- Soient : T un tracé comportant au moins une séquence S; n le nombre total de séquences S dans le tracé T; S_i la ième séquence du tracé
- T = S_1
- i = 2

L'identification des caractères repose sur la combinaison de plusieurs algorithmes de reconnaissance. L'objectif visé est l'amélioration des taux de reconnaissance en tirant avantage des forces de chacun des algorithmes tout en éludant leurs faiblesses. Ces algorithmes sont complètement indépendants et fonctionnent sans aucune interférence. Ils peuvent s'exécuter en parallèle si le système le permet ou séquentiellement sur un système monotâche. A l'issue de la phase de reconnaissance, ils fournissent chacun une liste de caractères candidats. Les

différentes listes obtenues sont alors regroupées dans un ensemble de conflits traité par un organe de décision (classifieur) qui doit parvenir à un accord et proposer une solution finale. A l'heure actuelle, deux algorithmes de reconnaissance sont combinés. Le premier exploite la dynamique des tracés (ou ordre d'écriture), le second leur aspect visuel ou géométrique. On parle alors de méthode de reconnaissance dynamique et de méthode de reconnaissance statique. Ces deux méthodes sont détaillées dans la partie qui suit.

5.2 Méthode dynamique

5-2-1 Codage

Avec la méthode de reconnaissance dynamique, la principale caractéristique d'une lettre est le mouvement effectué durant son tracé. Ce mouvement (ou ordre d'écriture) est décrit par la suite ordonnée des déplacements de la plume sur la surface d'écriture. Pour cela, deux techniques de codage directionnel sont utilisées: le codage relatif (ou différentiel) et le codage absolu [7, 25]. Le codage différentiel code une direction du tracé par rapport à la précédente. Ce dernier présente la caractéristique d'être invariant par translation mais peut générer dans certains cas des ambiguïtés. Par exemple, il est difficile de discriminer les chiffres 6 et 9 en se basant essentiellement sur le codage différentiel. Pour y remédier on complète la représentation des tracés par le codage absolu. Ce dernier comporte 8 codes directionnels obtenus grâce à un échantillonnage angulaire de pas $\pi/4$. Il porte essentiellement sur le premier segment d'un caractère dont le code varie avec l'orientation du tracé. Une première primitive, notée P1, est ainsi obtenue. Celle-ci comporte deux composantes : la première correspond au codage absolu du premier segment d'un caractère, la seconde au codage différentiel permettant de décrire le reste du tracé. Cependant, ces informations sont encore insuffisantes pour discriminer certains caractères, telles les lettre F et I ou D et P, qui peuvent posséder une même primitive P1. Pour y remédier, nous introduisons une deuxième primitive P2 dont le rôle est de lever ce type d'ambiguïté. Cette primitive comporte deux composantes : $P2_{C1}$ et $P2_{C2}$. La composante $P2_{C1}$ code la répartition spatiale des séquences (suites de points séparées par des levés de plume) d'un caractère dans le cadre délimitant son tracé. Ce cadre est divisé en trois zones. $P2_{C1}$ indique la ou les zones occupées par chaque séquence. La figure 3 illustre un exemple de levé d'ambiguïté à l'aide de $P2_{C1}$.

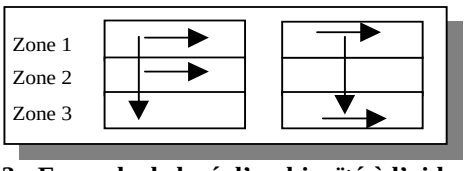

Fig. 3 : Exemple de levé d'ambiguïté à l'aide de la composante $P2_{C1}$

Si les trois séquences composant F et I sont tracées dans le même ordre, P1 fournit un codage identique pour les deux lettres. En revanche, $P2_{C1}$ indique une différence au niveau de la séquence S3 (S3=2 pour F et S3=3 pour I). Cette première composante de P2 est aussi insuffisante à elle seule pour lever l'ensemble des ambiguïtés rencontrées. Par exemple, les lettres D et P ne comportent qu'une seule séquence, couvrant les trois zones, lorsque leurs tracés sont continus. On introduit alors la deuxième composante $P2_{C2}$ pour les discriminer. La figure 4 illustre le principe de résolution de ce type d'ambiguïté grâce à la composante $P2_{C2}$. Cette dernière prend en compte la position du levé et du baissé de plume : pour la lettre P, le levé de plume est dans la zone 2 alors que pour D, il est dans la zone 3.

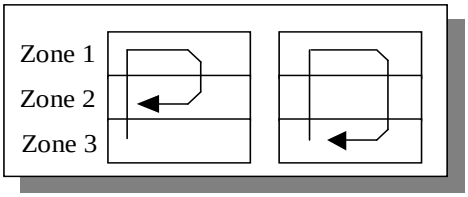

Fig. 4 : Exemple de levé d'ambiguïté avec la composante $P2_{C2}$

5-2-2 Reconnaissance

L'identification d'un caractère inconnu s'effectue en le comparant aux modèles de référence (obtenus par apprentissage) grâce à un algorithme de reconnaissance par programmation dynamique [3, 22]. Cette opération peut se décrire comme une optimisation locale de la distance cumulée entre le caractère inconnu et le prototype considéré. Notons *d(i, j)* la distance entre les éléments (les codes de la primitive considérée (P1 ou P2)) d'indices i et j du caractère inconnu et du prototype activé, *D(i, j)* la distance minimale cumulée de *(0, 0)* à *(i, j)*. L'algorithme de comparaison s'exprime alors ainsi :

$$D(0,0) = d(0,0)$$

$$D(i,j) = Min\left\{\begin{array}{c} D(i,j-1)+d(i,j) \\ D(i-1,j-1)+\frac{1}{2}d(i,j) \\ D(i-1,j)+d(i,j) \end{array}\right\}$$

Cet algorithme invoque séparément les primitives P1 et P2 des tracés comparés. Compte tenu du nombre total d'éléments composant ces primitives (m_{P1} *et* m_{P2} *pour le* caractère inconnu, n_{P1} *et* n_{P2} pour le prototype activé), deux distances $D(m_{P1}, n_{P1})$ et $D(m_{P2}, n_{P2})$ sont calculées. En les combinant, on obtient une distance finale *Dop* calculée comme suit :

$Dop = \alpha.D(m_{P1}, n_{P1}) + \beta.D(m_{P2}, n_{P2})$, *avec* $\alpha=1$ *et* $\beta=2$ *(valeurs fixeés expérimentalement)*

A l'issue de ce calcul, l'algorithme de reconnaissance propose une liste formée des 12 meilleurs caractères candidats, compte tenu des 12 plus faibles distances *Dop* calculées. Cette liste se présente sous la forme d'une suite de couples formés chacun d'un caractère candidat et de la distance *Dop* qui lui est associée.

5-3 Méthode statique

5-3-1 Codage

Il s'applique à des tracés filtrés essentiellement en distance. Il consiste à subdiviser le rectangle encadrant un tracé en 36 cellules de mêmes tailles et à lui faire

correspondre une matrice de codage M[6,6]. On affecte le code 9 aux cellules comportant au moins un point du tracé (cellules pleines). Les codes des autres cellules sont déterminés en comptabilisant pour chaque cellule vide le nombre de cellules adjacentes affectées du code 9. Le voisinage considéré étant 8-connexes, la valeur de ces codes varie de 0 à 8. Ceci indique que lorsque le tracé d'une lettre modèle se déforme, la probabilité de trouver un point dans une cellule initialement vide augmente avec le nombre de cellules pleines qui l'entourent. Ce moyen de codage très simple permet d'appréhender les éventuelles migrations d'un point d'une cellule vers ses voisines lorsque l'écriture d'un scripteur se dégrade. La figure 5 illustre un exemple de codage de la lettre E

9	9	9	9	9	9
9	5	3	3	3	2
9	5	3	3	3	2
9	9	9	9	9	9
9	7	6	6	6	4
9	9	9	9	9	9

Fig. 5 : Exemple de codage de la lettre E

5-3-2 Reconnaissance

Elle consiste à comparer la matrice de codage du caractère inconnu à celles des modèles de référence (obtenus par apprentissage) grâce à un balayage systématique de la base d'apprentissage. Notons *I(6, 6)* la matrice de codage du caractère inconnu et *P(6, 6)* celle du prototype activé.

Le score *S* de reconnaissance se calcule alors comme suit :

$$S_1 = \sum_{i=0}^{6}\sum_{j=0}^{6} P(i,j) \;;\quad S_2 = \sum_{i=0}^{6}\sum_{j=0}^{6} (9 - I(i,j)) \;;$$
$$I(i,j)=9 \qquad P(i,j)=9$$

$$S = Max(\alpha S_1 - \beta S_2, 0)$$

Les termes S_1 et S_2 mesurent respectivement les degrés de correspondance et de dissemblance entre le prototype sélectionné et le caractère inconnu. Les coefficients α et β sont ajustés expérimentalement (α=4; β=9). A l'issue de cette opération de comparaison, une liste formée des 12 meilleurs caractères candidats, assortis de leur score de reconnaissance, est proposée.

6 NORMALISATION DES RESULTATS

Les scores fournis par les deux approches ne sont pas homogènes. En effet, ils ne représentent pas les mêmes grandeurs et n'ont pas les mêmes échelles. Par conséquent, la combinaison de ces deux approches ne peut avoir lieu que si l'on parvient à transformer ces mesures en des valeurs de même type. L'opération de normalisation consiste alors à les convertir en pourcentages mesurant le degré de ressemblance entre le caractère inconnu et les prototypes de référence. Notons $S_{b1}(i)$ et $S_{b2}(i)$ les scores brutes des caractères candidats de rang i dans les deux listes. Leurs scores normalisés $S_{n1}(i)$ et $S_{n2}(i)$ sont alors calculés comme suit :

$$S_{n1}(i) = (\underset{j}{Max}\ (S_{b1}(j)) - S_{b1}(i)) \times \frac{100}{\underset{j}{Max}\ (S_{b1}(j))}$$

$$S_{n2}(i) = \frac{S_{b2}(i) \times 100}{\alpha \times \beta \times NCP}$$

$\underset{j}{Max}(S_{b1}(j))$ représente la valeur maximale des scores associés aux candidats de la première liste et $\alpha.\beta$.NCP celle des candidats de la deuxième liste. NCP étant le nombre de cellules pleines du caractère inconnu, on atteint la valeur $\alpha.\beta$.NCP lorsqu'il y a correspondance totale entre le caractère inconnu et le prototype sélectionné [25].

7 CLASSIFIEUR

Il consiste à parvenir à un consensus en fournissant une liste finale composée des 12 meilleurs caractères candidats dans les deux listes. On y arrive en fusionnant les deux listes après avoir recalculé l'ensemble des scores de reconnaissance en combinant trois paramètres : le score normalisé, le rang et la fréquence d'apparition de chaque caractère dans les deux listes. Notons : $[C_1(i), S_1(i)]$ le couple *[candidat, score associé]* de rang i dans la liste L_1 et $[C_2(i), S_2(i)]$ celui de la liste L_2. Les couples $[C_c(k), S_c(k)]$ de la liste de fusion sont alors calculés selon l'algorithme suivant :

Soient : $i=0,\ k=0,\ n=11$ *et* $j \in \{0,1,2,\ldots,11\}$
Tant que $i \le 11$
faire

$$\left[C_c(k), S_c(k)\right] = \begin{cases} \left[C_1(i), \frac{1}{2} \times \left(S_1(i) \times \frac{n-i}{n} + S_2(i) \times \frac{n-i}{n} \right)\right] & si \quad \exists j \ / \ C_1(i) = C_2(j) \\ \left[C_1(i), \frac{1}{2} \times S_1(i) \times \frac{n-i}{n}\right] & si \quad C_1(i) \notin L_2 \end{cases}$$

$$k = k+1 \ \ et \ \ \left[C_c(k), S_c(k)\right] = \left[C_2(i), \frac{1}{2} \times S_2(i) \times \frac{n-i}{n}\right] \quad si \quad C_2(i) \notin L_1$$

$i = i+1$
$k = k+1$
Fin faire
Fin Tant que

A l'issue de ce calcul, on parachève la liste de fusion en classant les caractères candidats par ordre de vraisemblance décroissant, compte tenu des scores réalisés. Finalement, une liste finale formée des 12 meilleurs caractères candidats est proposée.

8 MISE EN OEUVRE ET RESULTATS

Après une courte phase d'apprentissage sur une base de laboratoire composée de 287 tracés de 4 scripteurs différents, le système est soumis à 2 séries de tests. 5 scripteurs dont l'écriture est inconnue ont participé aux essais. La première évaluation porte sur des tracés de caractères isolés. Pour chaque scripteur, l'évaluation consiste à écrire 3 fois la liste des 10 chiffres et 26 lettres bâtons. Les taux de reconnaissance moyens obtenus sont affichés ici :

Méthode de reconnaissance	En 1ère position	Autre position	Absent de la liste
Dynamique	73.39%	18,81%	07,80%
Statique	71,10%	20,64%	08,26%
Classifieur	87,61%	09,64%	02,75%

Malgré le nombre réduit des algorithmes mis en combinaison, ces résultats montrent que le classifieur induit une amélioration des taux de reconnaissance de l'ordre de 15%. Ceci confirme que lorsque des algorithmes basés sur des méthodologies ou des primitives différentes sont complémentaires, une fonction de combinaison peut être efficace et augmenter le taux de bonne classification. Toutefois, parvenir à un consensus dans une combinaison de plusieurs algorithmes est un problème difficile dont la résolution peut nécessiter, entre autre, une normalisation des scores de reconnaissance. Cette opération très délicate représente aujourd'hui le souci majeur de notre recherche car il n'existe pas de solution générale à ce problème. En effet, les solutions proposées dans la littérature sont encore dans leurs débuts et dépendent fortement du type de résultat fourni par les algorithmes mis en combinaison [5, 6, 11, 17].

La deuxième évaluation porte sur la segmentation et la reconnaissance de caractères dans l'écriture mixte. Pour chaque scripteur le texte écrit est : ***« Ceci Est UN Test et n'a pas de Rapport AVEC l'Ecriture Ordinaire. Merci ».*** Compte tenu du nombre total des caractères bâtons présents dans le texte, d'une part, et de celui des minuscules d'autre part, la moyenne arithmétique et l'écart type des résultats obtenus sont affichés ici:

Résultats (candidat en tête de la liste finale)	Moyenne (%)	Ecart type (%)
Caractères segmentés et reconnus	81.87	5.15
Caractères manqués (non détectés)	18.13	5.15
Confusion (minuscules → majuscules)	15.50	7.77

Ces premiers résultats, obtenus dans un environnement multiscripteur d'écriture mixte, sont très encourageants. Toutefois, pour espérer atteindre des scores plus élevées, deux difficultés doivent être surmontées : réduire le nombre de caractère manqués par le système et traiter les confusions introduites par les lettres minuscules prises pour des majuscules. Le premier problème peut être traité en augmentant le nombre d'algorithmes mis en combinaison. Ceci doit permettre d'améliorer la reconnaissance et par suite de mieux contrôler l'opération de segmentation sous-jacente. Le second problème peut être pris en charge en faisant interagir la segmentation et la reconnaissance des caractères bâtons avec celles des lettres cursives. Pour ce faire, plusieurs solutions peuvent être envisagées. La plus simple consiste à doubler systématiquement la reconnaissance des lettres bâtons par celle des lettres cursives. Un module de décision placé en aval doit permettre d'optimiser les résultats en sélectionnant le meilleur candidat compte tenu des scores de reconnaissance obtenus. Enfin, l'introduction de la logique floue [10] doit permettre d'améliorer les performances globales du système en retardant la décision finale afin de tenir compte de l'ensemble des informations en provenance des différents agents impliqués dans le processus de reconnaissance.

Références

[1] **E. BROCKLEHURST,** *The NPL electronic paper project , Technical report DITC 133/138,* National Physic Laboratory, Teddington, Middlesex, TW11OLW, UK., 1988

[2] **V. Bouletreau, N. Vincent, R. Sabourin, H. Emptoz,** *Ecriture et signature : même comportement face à la reconnaissance, ,* Actes du 1er Coll. Inter. Franc. sur l'Ecrit et le Doc., Québec, Canada, mai 1998, pp. 316-324

[3] **C. CHEN,** *Modèle cognitif et reconnaissance des formes. Une représentation des connaissances morphologiques : Application à la reconnaissance de l'écriture cursive de mots manuscrits,* Thèse de docteur d'université, Univ. Paris XII – Val de Marne, 1986

[4] **R. De Possel,** *La lecture automatique intensive, automatisme n° 9, sept. 1962*

[5] **A. Ennaji, Y. Lecourtier,** *Classification et coopération en reconnaissance de caractères imprimés multifontes,* Actes 9ème Cong. Recon. des Formes et Int. Artif., Paris, pp. 665-670, 1994

[6] **F. Firama, M. Shridhar,** *Handwritten numerical recognition based on multiple algorithms,* Pattern Recognition, 24, 10, pp. 969-983, 1991.

[7] **H. Freeman,** *On the quantization of line drawing data,* IEEE Trans. Syst. Sci. and Cybernetics, 1969, pp. 70-79

[8] **L. S. Frishkopf, L. D. Harmon,** *Machine reading of cursive script words,* Information Theory Symp., London 1960, C. CHERRY ed. Washington : Butterworths, 1961, pp. 300-316

[9] **V. Govindaraju, S. N. Srihari,** *Separating handwritten text from overlaping non-textual contours,* Proc. 2nd Intern. Workshop on Frontiers in Handwriting Recog., Bonas, France, sept. 1991, pp. 111-119

[10] **J.F. Hébert M. Parizeau et N. Ghazzali**, *Une représentation floue régionale pour la reconnaissance en-ligne de chiffres manuscrits,* Actes du 1er Coll. Inter. Franc. sur l'Ecrit et le Doc., Québec, Canada, mai 1998, pp. 394-403

[11]**T. K. ho, J. J. Hull, N. Srihari,** *Combinaison of structural classifieurs,* in Proc. IAPR Workshop Statystic. and Structural Pattern Recog., june 1990, pp. 123-137

[12]**E. Lecolinet,** *Segmentation d'images de mots manuscrits : Application à la lecture de chaînes de caractères alphanumériques et à la lecture de l'écriture cursive,* Thèse de docteur d'université, Univ. Paris VI, 1990

[13]**M. Leroux,** *Reconnaissance de textes manuscrits à vocabulaire limité, avec application à la lecture automatique des chèques,* Thèse de docteur d'université, Univ. De Rouen, nov. 1991

[14]**N. Lindgreen**, Cursive script recognition, IEEE Spect., Vol. 2, n° 5, 1965, pp. 104-116

[15]**G. Lorette, Y. Lecourtier,** *Reconnaissance et interprétation de textes manuscrits hors-ligne : un problème d'analyse de scène ?,* Coll. Nat. Sur l'Ecrit et le Docum., Nancy 1992, pp. 109-139

[16]**G. M. Miller,** *Real time classification of script handwriting words,* IFIP Congress, 1971, Amesterdam North-Holland Publishing Company, 1972, pp. 218-223

[17]**C. Nadal, R. Legault, C. Y. Suen**, *Complementary algorithms for the recognition of totally unconstrained handwritten numerals,* in Proc. 10 th Int. Conf. Pattern Recog., vol. A, June 1990, pp. 434-449

[18]**H. Oulhadj, L. Bennacer, h. Wehbi, J. Lemoine, E. Petit,** *Système de reconnaissance hiérachique d'écriture à architecture modulaire,* 9ème Cong. AFCET-RFIA, Paris, janv. 1994, pp. 451-460

[19]**H. Oulhadj, J. Lemoine, L. Bennacer, E. Petit,** *Reconnaissance en-ligne d'écriture cursive dégradée : une méthode structurelle par double balayage et recherche de points d'ancrage,* Actes du 1er Coll. Inter. Franc. sur l'Ecrit et le Doc., Québec, Canada, mai 1998, pp. 209-217

[20]**R. Plamondon , G. Lorette, G. Poulin,** *On the intrinsic theoritical and pratical difficulties of designing an automatic sigature verification system,* Proc. 3rd Int. Symp. On Handwriting and Computer Applic., Montreal, 1987, pp. 154-156

[21]**J. C. Salome M. Leroux, H. Oiry, A. Saad,** *Retrievial of script information appering on bank cheques for automatic reading purpose,* Proc. SPIE VCIP, Lausanne, 1990, pp. 1652-1662.

[22]**C. C. Tappert,** *Cursive script recognition by elastic matching,* IBM J. Research and Develop. Vol. 26, n° 6, nov. 1982, pp. 765-771

[23]**C. C. Tappert,** *Adaptative on-line handwriting recognition,* Proc. of the 7th ICPR, Montréal, Canada, July 1984, pp. 1004-1007

[24]**C. C. Tappert, C. Y. Suen T. Wakahara**, *The state of the art in on-line handwriting recognition,* IEEE Trans. Patt. Analys. and Mach. Intel., PAMI-12, 8, 1990, pp. 787-807.

[25]**H. Wehbi,** *Segmentation et reconnaissance en-ligne des mots d'écriture manuscrite mixte,* thèse de docteur d'université, Univ. Paris XII – Val de marne, 1996