Reconnaissance en ligne de caractères alphanumériques manuscrits

Wee Wang Loy

THESE

Présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE
DE GRENOBLE

Pour obtenir

LE TITRE DE DOCTEUR—INGENIEUR
SPECIALITE : AUTOMATIQUE

Par

Wee Wang LOY

RECONNAISSANCE EN LIGNE DE CARACTERES ALPHANUMERIQUES MANUSCRITS

Soutenue le 2 Octobre 1981 devant la commission d'Examen

JURY

Monsieur	*R. PERRET,*	*Président*
Messieurs	*L. BOLLIET*	*Examinateurs*
	A. CHEHIKIAN	
	J.P. HATON	
	I.D. LANDAU	

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année Universitaire
1980-1981

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

PROFESSEURS DES UNIVERSITES

MM	ANCEAU François	Informatique fondamentale et appliquée
	BESSON Jean	Chimie Minérale
	BLIMAN Samuël	Electronique
	BLOCH Daniel	Physique du Solide - Cristallographie
	BOIS Philippe	Mécanique
	BONNETAIN Lucien	Génie Chimique
	BONNIER Etienne	Métallurgie
	BOUVARD Maurice	Génie Mécanique
	BRISSONNEAU Pierre	Physique des Matériaux
	BUYLE-BODIN Maurice	Electronique
	CHARTIER Germain	Electronique
	CHENEVIER Pierre	Electronique
	CHERADAME Hervé	Chimie Physique Macromoléculaire
Mme	CHERUY Arlette	Automatique
MM	CHIAVERINA Jean	Biologie, biochimie, agronomie
	COHEN Joseph	Electronique
	COUMES André	Electronique
	DURAND Francis	Métallurgie
	DURAND Jean-Louis	Physique Nucléaire et Corpusculaire
	FELICI Noël	Electrotechnique
	FOULARD Claude	Automatique
	GUYOT Pierre	Métallurgie Physique
	IVANES Marcel	Electrotechnique
	JOUBERT Jean-Claude	Physique du Solide - Cristallographie
Mme	JOURDAIN Geneviève	Traitement du Signal
MM	LACOUME Jean-Louis	Géophysique - Traitement du Signal
	LANCIA Roland	Electronique - Automatique
	LESIEUR Marcel	Mécanique
	LESPINARD Georges	Mécanique
	LONGEQUEUE Jean-Pierre	Physique Nucléaire Corpusculaire
	MOREAU René	Mécanique
	MORET Roger	Physique Nucléaire Corpusculaire
	PARIAUD Jean-Charles	Chimie-Physique
	PAUTHENET René	Physique du Solide - Cristallographie
	PERRET René	Automatique
	PERRET Robert	Electrotechnique
	PIAU Jean-Michel	Mécanique
	POLOUJADOFF Michel	Electrotechnique
	POUPOT Christian	Electronique - Automatique
	RAMEAU Jean-Jacques	Electrochimie - Corrosion
	ROBERT André	Chimie appliquée et des matériaux
	ROBERT François	Analyse numérique
	SABONNADIERE Jean-Claude	Electrotechnique
Mme	SAUCIER Gabrielle	Informatique fondamentale et appliquée

.../...

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

PROFESSEURS DES UNIVERSITES

MmeSCHLENKER Claire	Physique du Solide - Cristallographie
MM SCHLENKER Michel	Physique du Solide
SOHM Jean-Claude	Chimie Physique
TRAYNARD Philippe	Chimie - Physique
VEILLON Gérard	Informatique fondamentale et appliquée
ZADWORNY François	Electronique

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

M FRUCHART Robert	Directeur de Recherche
MM ANSARA Ibrahim	Maître de Recherche
CARRE René	Maître de Recherche
DAVID René	Maître de Recherche
DRIOLE Jean	Maître de Recherche
KAMARINOS Georges	Maître de Recherche
KLEITZ Michel	Maître de Recherche
LANDAU Ioan-Doré	Maître de Recherche
MERMET Jean	Maître de Recherche
MUNIER Jacques	Maître de Recherche

Personnalités habilitées à diriger des travaux de recherche (Décision du Conseil Scientifique)

E.N.S.E.E.G.

MM ALLIBERT Michel	
BERNARD Claude	
CAILLET Marcel	
MmeCHATILLON Catherine	
MM COULON Michel	
HAMMOU Abdelkader	
JOUD Jean-Charles	
RAVAINE Denis	
SAINFORT	C.E.N.G.
SARRAZIN Pierre	
SOUQUET Jean-Louis	
TOUZAIN Philippe	
URBAIN Georges	Laboratoire des Ultra-Réfractaires ODEILLO

E.N.S.M.S.E.

MM BISCONDI Michel	
BOOS Jean-Yves	
GUILHOT Bernard	
KOBILANSKI André	
LALAUZE René	
LANCELOT Francis	
LE COZE Jean	
LESBATS Pierre	
SOUSTELLE Michel	
THEVENOT François	

.../...

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

THOMAS Gérard
TRAN MINH Canh
DRIVER Julian
RIEU Jean

E.N.S.E.R.G.

MM BOREL Joseph
CHEHIKIAN Alain
VIKTOROVITCH Pierre

E.N.S.I.E.G.

MM BORNARD Guy
DESCHIZEAUX Pierre
GLANGEAUD François
JAUSSAUD Pierre
Mme JOURDAIN Geneviève
MM LEJEUNE Gérard
PERARD Jacques

E.N.S.H.G.

M DELHAYE Jean-Marc

E.N.S.I.M.A.G.

MM COURTIN Jacques
LATOMBE Jean-Claude
LUCAS Michel
VERDILLON André

*
* *
*

à Geok Lian,

REMERCIEMENTS.

==========

Cette thèse a pu être réalisée, grâce à l'appui financier du gouvernement français et l'aide technique et financière de la Société OPTION.

Je tiens à remercier sincèrement:

Monsieur le professeur R. PERRET, Directeur du Laboratoire d'Automatique de Grenoble, de m'avoir accueilli dans son laboratoire et pour l'honneur qu'il me fait en présidant ce jury.

Monsieur L. BOLLIET, Professeur à l'Université des Sciences Sociales de Grenoble et Monsieur A. CHEHIKIAN, Professeur à l'Université Scientifique et Médicale de Grenoble d'avoir accepté de siéger dans ce jury.

Monsieur J. P. HATON, Professeur à l'Université de Nancy I d'avoir accepté d'examiner mes travaux et de participer au Jury.

Monsieur I. D. LANDAU, Maître de Recherche au C.N.R.S., mon Directeur de thèse, pour les conseils, l'aide et les encouragements qu'il m'a prodigués tout au long de ce travail.

Je remercie également :

Mon collègue Monsieur Luc DUGARD pour son aide et qui a bien voulu relire le manuscrit de ma thèse.

Monsieur C. BEVIER, Chef de Service Calcul de notre Ecole pour m'avoir aidé à résoudre les problèmes softwares et hardwares durant ce travail.

° °
0

TABLE DES MATIERES.

Pages

CHAPITRE III.

CHAPITRE IV.

CHAPITRE V.

APPENDICES.

APPENDICE A. *L'idendification des paramètres avec un algorithme récursif*

APPENDICE B. *L'équation récursive de l'inverse de la matrice de covariances*

APPENDICE C. *L'analyse en composantes principales*

APPENDICE D. *Les caractères de base de données monoscripteur*

APPENDICE E. *Les automates finis*

APPENDICE F. *Les caractères de base de données multiscripteurs*

APPENDICE G. *An ON-LINE procedure for recognition of handwritten alphanumeric characters*

APPENDICE H. *Reconnaissance EN LIGNE de caractères alphanumériques manuscrits ultilisant des approximations polygonales lineaires et curvilignes*

BIBLIOGRAPHIE.

° °

0

CHAPITRE I.

I. INTRODUCTION.

1.1. PRESENTATION DU PROBLEME.

La reconnaissance de caractères a été un sujet de recherche attentive pendant une vingtaine d'années. Les progrès et les efforts considérables se sont manifestés par un grand nombre de publications, de brevets et de livres dans ce domaine. Aujourd'hui, les systèmes de reconnaissance ont évolué des méthodes primitives qui peuvent seulement reconnaître les caractères imprimés jusqu'aux méthodes sophistiquées qui permettent la reconnaissance de caractères et de symboles multifontes et manuscrits.

La reconnaissance de caractères manuscrits est un problème difficile, mais très intéressant et stimulant. Pour savoir jusqu'à quel point l'homme est capable de reconnaître les caractères manuscrits, Neisser et Al [23] ont fait une étude sur ce problème. Ils demandent à neuf personnes d'identifier les caractères alphanumériques majuscules manuscrits et ils trouvent un taux de reconnaissance de 94,9 % à 96,5 % selon les individus. Le score avec une meilleure supposition est de 96,8 %. D'après cette étude, on peut voir que même l'homme, qui est supposé posséder le meilleur système de lecteur optique (les yeux) ne peut atteindre qu'un taux de 96 % en moyenne. Ce résultats peut nous servir comme référence quand on développe un système de reconnaissance.

Les recherches dans le domaine de la reconnaissance de caractères incluent :

. La définition des caractères manuscrits standards.

. La reconnaissance de caractères Hors Ligne (Système optique).

. La reconnaissance de caractères En Ligne (Tablette graphique).

La recherche des standards pour les caractères manuscrits est nécessaire si nous voulons accroître le taux de reconnaissance. Comme nous venons de voir, nous ne pouvons pas espérer un taux meilleurs que 97 % avec les caractères écrits sans contraintes. A part la détermination des contraintes appropriées à imposer, il est nécessaire de modifier ou de redéfinir les formes normales de certains doublets (ex. 5 et S, U et V, etc...), pour supprimer les ambiguités. Trois pays (le Canada, les Etats Unis et le Japon) ont défini leurs propres standards pour les caractères manuscrits [34]. Malheureusement, les trois standards ne sont pas tout à fait les mêmes et leur usage est encore très limité.

Le schéma bloc d'un système typique de reconnaissance est illustré figure 1.1. Mis à part les matériels utilisés, la différence entre un système de reconnaissance en ligne et un système hors ligne est la structure des données. Dans un système en ligne, les données sont acquises au fur et à mesure de l'écriture, elles sont donc fonction du temps et de la séquence des tracés. Alors que dans un système hors ligne, elles sont sous formes de matrices. En dehors de ceci, les approches sont identiques.

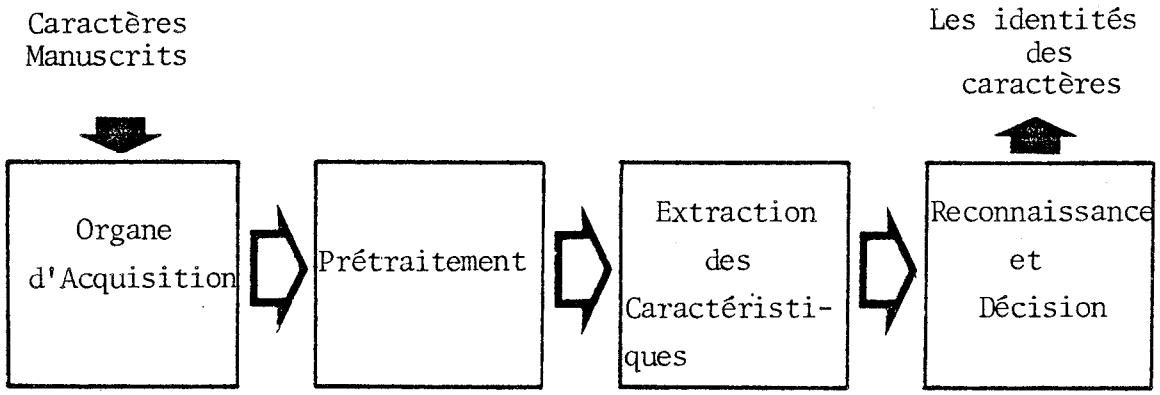


FIGURE 1.1. Schéma bloc d'un système de reconnaissance.

Les caractères manuscrits sont acquis et digitalisés par l'organe d'acquisistion. Le caractère isolé représenté par une matrice ou une chaîne de coordonnées est prétraité (ex. l'élimination des bruits d'acquisition, lissage et normalisation de la taille) pour faciliter la procédure d'extraction des caractéristiques. La reconnaissance de caractères est faite normalement en comparant les caractéristiques extraites avec celles acquises pendant l'apprentissage.

A cause de la disponibilité des informations temporelles dans un système en ligne, le prétraitement d'un caractère peut se faire plus facilement que dans un système hors ligne. D'autre part, le rapport signal/bruit est aussi plus élevé bien que cela dépende aussi de la taille du caractère écrit et la précision de la tablette employée.

1.2. LES DIFFERENTES APPROCHES.

Diverses procédures ont été proposées dans la littérature pour résoudre le problème de la reconnaissance de caractères. On peut les grouper en trois approches principales, à savoir :

. l'approche statistique [1, 3, 4, 8, 15, 16, 24, 29, 33]

. l'approche géométrique (ou par test) [12, 14, 22, 25, 27]

. l'approche structurelle (ou syntaxique) [2, 5, 6, 30, 35]

1.2.1. *l'approche statistique.*

Dans une approche statistique, on décrit un caractère entré par un vecteur de caractéristiques de dimension n, n est normalement beaucoup plus petit que le nombre total de données au départ.

On classifie ensuite le vecteur de caractéristiques suivant une règle de décision (Classifieur de Bayes par exemple).

Les paramètres du vecteur de caractéristiques sont généralement obtenus par une procédure de traitement classique. Par exemple, les coefficients de Fourier [1, 13]. Un choix est souvent nécessaire pour sélectionner les paramètres les plus significatifs pour réduire la dimension du vecteur.

L'aspect théorique de l'approche statistique est très développé en ce qui concerne les règles de décisions, les algorithmes d'apprentissage et le choix des paramètres. Le succès d'une telle approche dépend principalement de l'étape d'extraction des caractéristiques.

1.2.2. l'approche géométrique.

Dans une approche géométrique, on applique une série de tests portant sur des aspects distinctifs du caractère et on compare les réponses avec celles obtenues par apprentissage.

Cette approche est très spécifique à un problème particulier. Les tests à appliquer sont des tests particuliers à un problème précis et ils sont généralement désignés à l'avance. Dans le cas où le nombre de classes à identifier est grand, il est très difficile d'utiliser cette approche. D'ailleurs, l'ordre d'exécution des tests peut être critique dans certains cas.

1.2.3. l'approche syntaxique.

La plupart des travaux dans la reconnaissance de formes pendant les années passées concerne principalement l'approche sta-

tistique ou l'approche décision-théorique. Mais dans certains problèmes de reconnaissance, les informations structurelles qui décrivent chaque forme sont importantes et la reconnaissance implique non seulement la capacité d'attacher une forme à une certaine classe mais aussi la capacité de décrire l'aspect d'une forme pour qu'elle ne soit pas appropriée à une autre classe. Pour ce genre de problème, une approche syntaxique est nécessaire et convient mieux. Dans la reconnaissance de caractères manuscrits sans contraintes par exemple, il serait très difficile, sinon impossible, d'appliquer une méthode statistique.

La méthode syntaxique et la méthode statistique présentent deux aspects différents. Premièrement, la forme est décrite par une phrase d'un langage et non pas un vecteur de n paramètres. Deuxièmement, la classification d'une forme est faite par un 'accepteur syntaxique'.

La méthode syntaxique utilise les primitives pour décrire les caractéristiques locales et les règles de productions pour décrire les caractéristiques globales. Mais le choix des primitives et la construction des règles de production sont des problèmes très difficiles en général. Si les primitives sont très simples, on aura besoin d'une grammaire sophistiquée et vice-versa. De plus, il n'existe pas encore de moyen automatique pour construire la grammaire ('l'inférence grammaticale') d'une manière efficace [11]. Ce qui implique que l'on doit construire les règles à la main ou interactivement avec l'ordinateur. Pour un problème complexe, cette approche nécessiterait donc un travail manuel énorme !.

1.3. LA RECONNAISSANCE EN LIGNE.

Certaines personnes peuvent poser la question : A quoi sert la reconnaissance en ligne de caractères alors que la vitesse de frap-

pe pour une dactylo qualifiée est cinq fois plus grande que la vitesse normale de l'écriture. Evidemment, les systèmes de reconnaissance en ligne de caractères ne sont pas destinés à remplacer les machines à écrire. Mais pour la plupart des gens qui n'ont pas l'habitude de se servir de la machine à écrire, ils frappent beaucoup moins vite - de l'ordre de 60 caractères par minute pour les alphabets majuscules. Cette vitesse est inférieure à celle de l'écriture d'après l'étude de Devoe [7]. Dans ce cas là comme dans beaucoup d'autres, un système de reconnaissance en ligne de caractère se justifie.

En réalité, la reconnaissance en ligne de caractères est un sujet de grande importance pratique. Vu le progrès récent de la technologie, la tablette graphique - un élément essentiel de système en ligne - est devenue aujourd'hui un instrument précis et peu cher. Avec les softwares appropriés, beaucoup d'applications comme la conception assistée par ordinateur, l'entrée directe des formules mathématiques, logiques et chimiques sous formes canoniques, etc..., peuvent être réalisés. Quelques systèmes de ce genre sont en application actuellement, mais leurs performances demandent encore de grandes améliorations. Un effort important pour le développement de softwares, particulièrement celui de reconnaissance de caractère est donc nécessaire.

Le problème de la reconnaissance des caractères dans un environnement interactif a ses caractéristiques spécifiques concernant le type d'information disponible, le taux d'erreur tolérable, l'aspect de l'apprentissage et la complexité du matériel.

Dans la reconnaissance en ligne des caractères, à la différence du système optique, le temps de calcul n'est pas un problème crucial car la vitesse de reconnaissance de caractères n'a pas besoin d'excéder la vitesse d'écriture. Normalement, une vitesse de l'ordre de 0,5 sec suffit.

Les informations comme le nombre de traits et la séquence du tracé qui ne sont pas disponibles dans les systèmes optiques peuvent être incorporés dans le système en ligne. L'usage de ces informations peut aider à réduire la complexité de l'algorithme de reconnaissance et probablement le taux d'erreur.

La caractéristique 'Feed back' qui est associée avec l'environnement interactif peut être formalisée et incluse dans le système. Cette possibilité d'apprendre en ligne est très importante car le taux de reconnaissance s'améliore une fois que le système a corrigé ('Update') ses modèles pour mieux s'adapter à l'écriture de l'utilisateur ou inversement, l'utilisateur a appris la façon d'écrire pour que ses caractères puissent être reconnus.

1.4. NOS PROPOSITIONS.

Avant de donner nos propositions, nous allons rappeler l'état actuel de recherche dans le domaine de reconnaissance de caractères. En fait, les systèmes de reconnaissance de caractères imprimés ou manuscrits avec contraintes sont disponibles et sont en applications actuellement. Mais la reconnaissance de caractères manuscrits SANS contraintes est encore un sujet de recherche, ce problème est particulièrement difficile et il n'y a pas encore de propositions performantes. En comparaison, la recherche sur ce problème est plus attentive pour les systèmes optiques que pour les systèmes en ligne.

Ce travail est initialement proposé par la société OPTION. Le but est de transformer la tablette x - y à une tablette 'intelligente' qui permet de reconnaître les caractères écrits sur la tablette pour la compression de données à la transmission. Vu l'état actuel de recherche, nous pouvons constater que ce travail est encore sur le stade de recherche.

Nous avons divisé le travail en deux étapes.

La première étape est une étape d'initialisation et de compréhension. Nous nous limitons donc dans un environnement plus simple. Monoscripteur et Multiscripteur Avec contraintes. Ceci nous permet d'une part d'étudier les diverses approches, la possibilité d'introduire les méthodes utilisées en analyse de système (les algorithmes d'identification, d'adaptation et le traitement en ligne), et d'autre part, de réaliser un algorithme de reconnaissance simple que l'on peut implanter aisément sur un microprocesseur. Pour cela, nous avons proposé une solution utilisant l'approche statistique. Nous présenterons un résumé de cette approche dans la partie 1.4.1.

Dans la deuxième étape, nous étudions un problème plus compliqué - la reconnaissance de caractères Multiscripteurs SANS contraintes. Nous avons proposé une approche différente - une méthode syntaxique. Un résumé de cette approche est donné dans la partie 1.4.2.

1.4.1. Proposition N° 1.

Le schéma bloc du système de reconnaissance est illustré dans la figure 1.2.

Après le prétraitement, les caractères entrés sont segmentés en polygones linéaires par une méthode de segmentation. Un vecteur de caractéristiques est formé en utilisant les paramètres des segments du polygone. Chaque segment a deux paramètres : l'orientation du segment (θ) et la distance (d) du segment de l'origine des axes de références (Voir figure 1.3).

La classification est faite en 2 étapes :

La première en fonction du nombre de segments.

La seconde en fonction des règles de décisions statistiques.

Dans un premier temps, nous construisons le classifieur avec une base de données. Nous calculons la moyenne et la matrice de covariances du vecteur de caractéristiques pour chaque catégorie de caractères.

Avec ces types de classifieurs, il nous est possible d'une part, d'introduire de nouveaux caractères de manière récursive et d'autre part, d'améliorer la définition de chaque modèle du classifieur avec de nouveaux échantillons en utilisant l'algorithme d'apprentissage récursif développé.

1.4.2. Proposition N° 2.

Le schéma bloc du système de reconnaissance est illustré dans la figure 1.5.

Après le prétraitement, nous traçons le caractère entré sur le plan θ - S (θ - l'intégrale de courbure, S - l'axe curviligne) et nous segmentons les courbes $\theta(S)$ en une série de segments de droites. Chaque segment de droite est classé dans une classe de primitives (exemple : droite, coin, courbe à courbure positives, etc...). Nous enlevons ensuite les caractéristiques globales de ces primitives. La description du caractère entré est complétée en attachant à chaque caractéristique globale un attribut de 'position' (voir figure 1.4).

Pour reconnaître l'identité d'un caractère, nous procédons de la manière suivante : Nous cherchons dans le dictionnaire, l'existence de la même expression (Caractéristiques globales + Attributs).

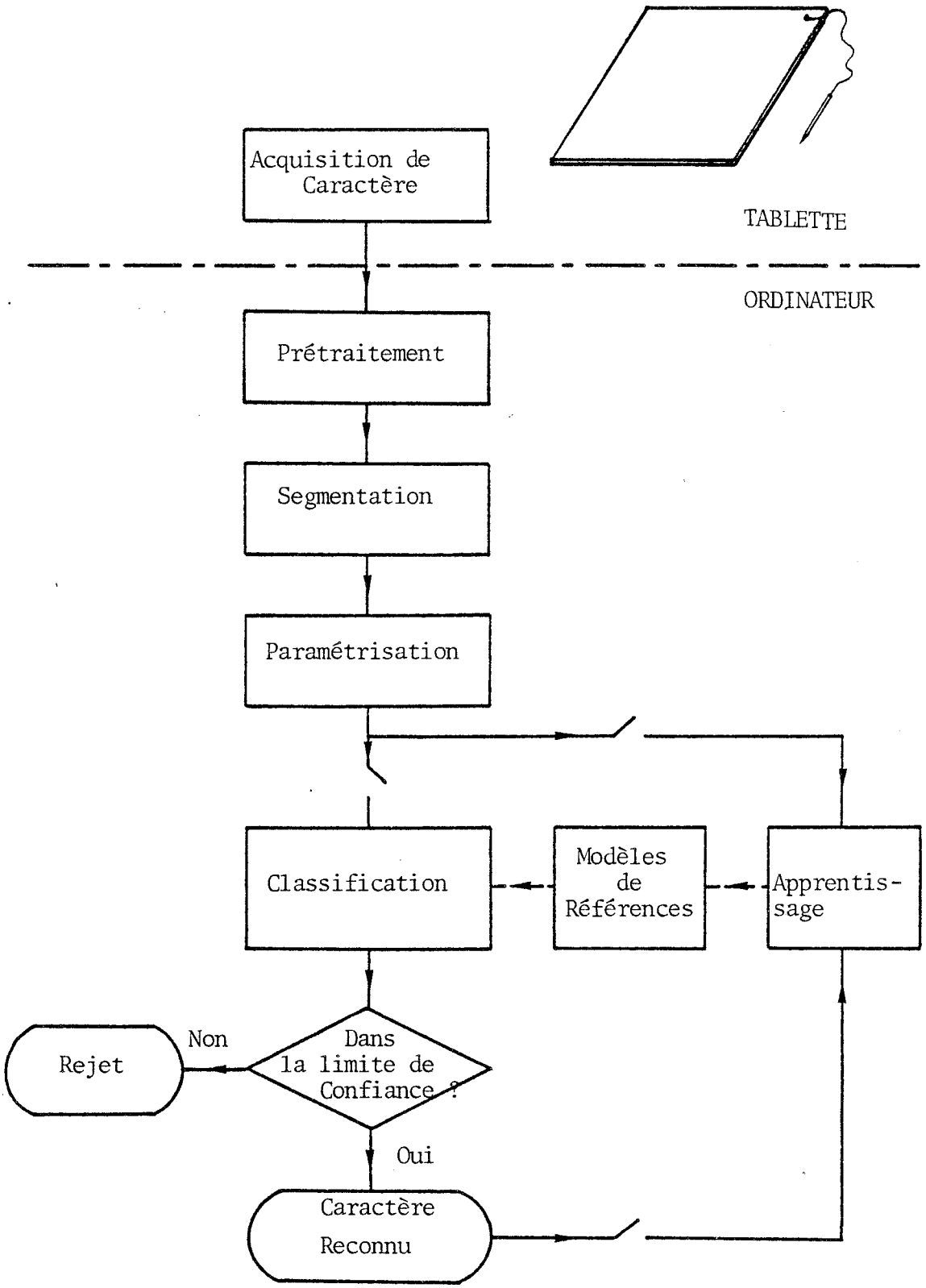


FIGURE 1.2. Schéma bloc du Système de Reconnaissance (Proposition N° 1).

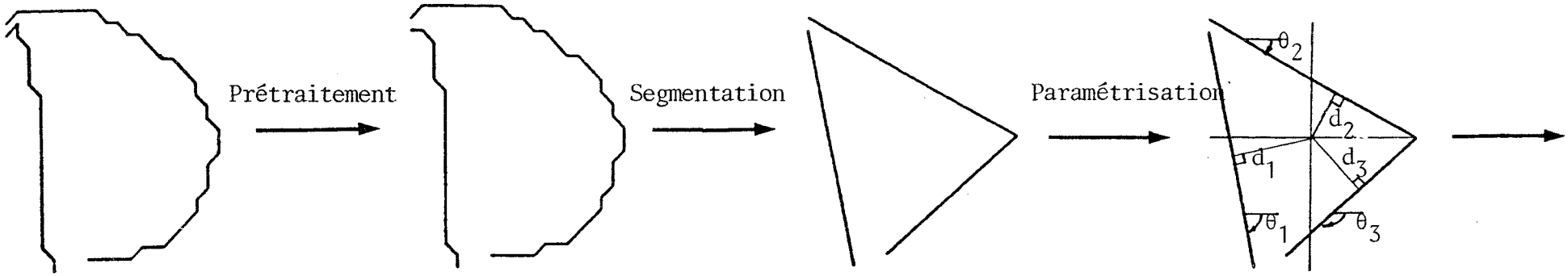


FIGURE 1.3. Procédure d'extraction de caractéristiques. (Proposition N° 1).

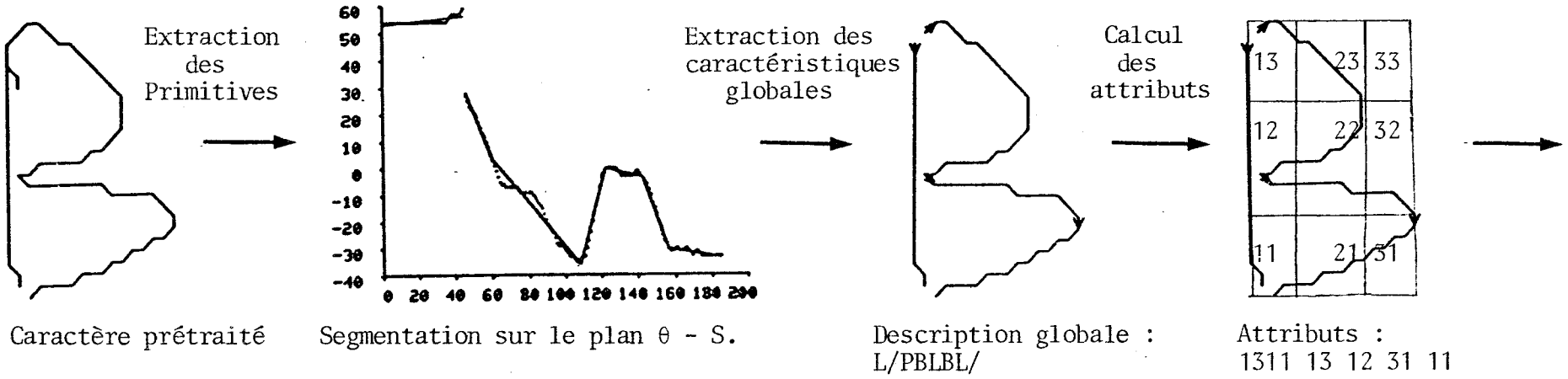


FIGURE 1.4. Procédure d'extraction de caractéristiques (Proposition N° 2).

S'il en existe, le caractère est classifié. Dans le cas contraire, si la description globale n'existe pas dans le dictionnaire nous rejetons le caractère entré ; sinon, nous calculons la distance entre l'expression du caractère entré avec celle du dictionnaire, le caractère sera reconnu ou rejeté selon cette distance.

L'apprentissage d'un nouveau caractère peut se faire avec un nombre très réduit d'échantillons pour commencer. Si la performance n'est pas satisfaisante, on peut introduire des nouveaux échantillons au fur et à mesure pour élargir les modèles de références.

Les deux programmes sont écrits en FORTRAN sur l'ordinateur NORD 10 S. Le premier programme est réécrit en langage Assembleur d'INTEL 8086 et testé sur le système de développement [46].

Cette thèse est organisé de la façon suivante :

- Dans le Chapitre II, nous présentons la méthode d'acquisition des données et les algortihmes de prétraitements. Nous discutons en détail la structure des données, le filtrage et le lissage effectués au niveau de la tablette et au niveau de l'ordinateur.

- Le Chapitre III est consacré à l'étude d'une méthode statistique pour la reconnaissance de caractère manuscrits Monoscripteur et Multiscripteur AVEC contraintes. Ce chapitre résume l'article "An on line procédure for recognition of handwritten alphanumeric characters" [17] et l'article "Reconnaissance en ligne de caractères alphanumériques manuscrits utilisant des approximations polygonales linéaires et curvilignes" [18].

- Dans le Chapitre IV, nous étudions une méthode syntaxique

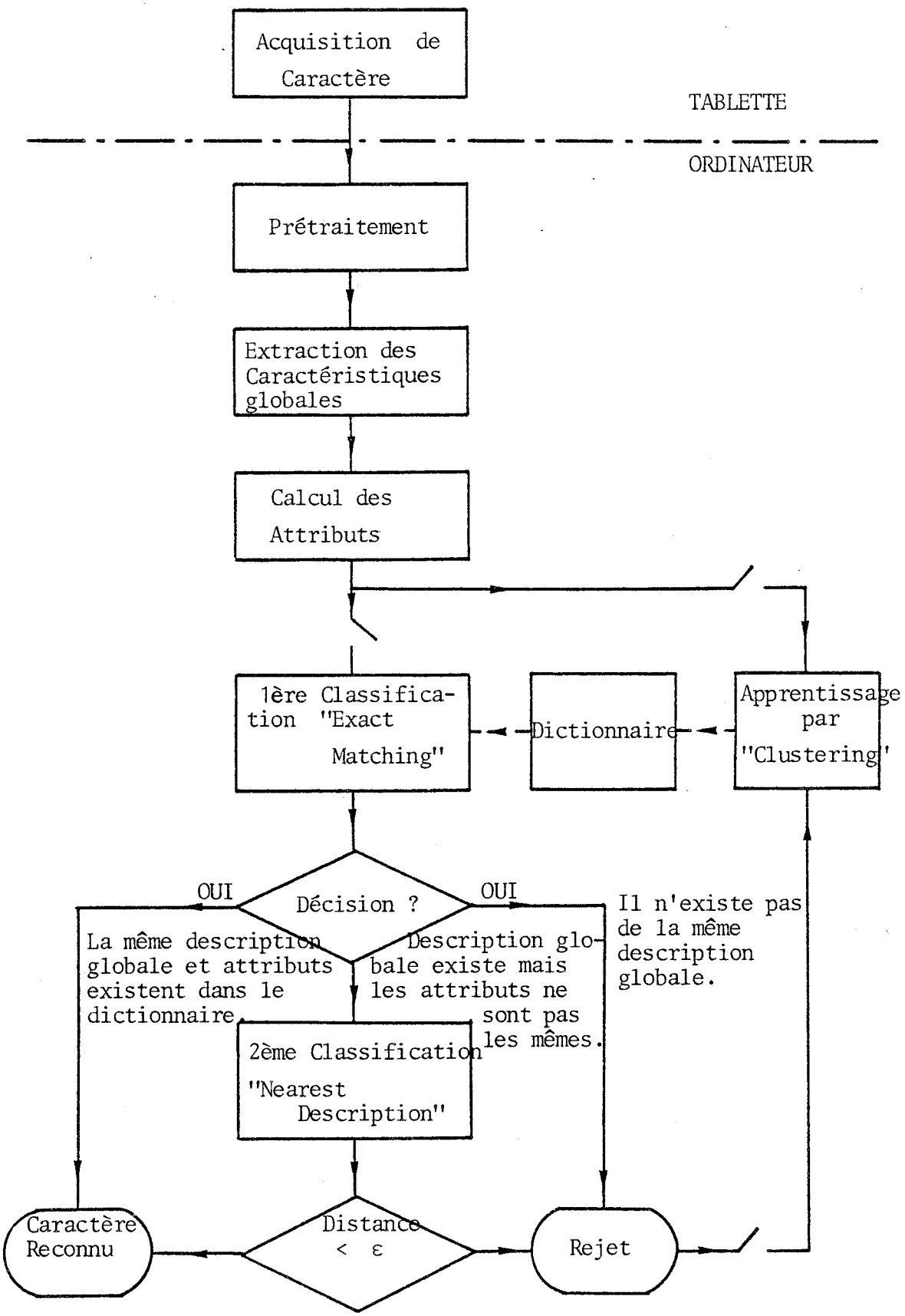


FIGURE 1.5. Schéma bloc du Système de Reconnaissance (Proposition N° 2).

pour la reconnaissance de caractères manuscrits Multi-scripteur SANS contraintes. Ce chapitre résume l'article "On line recognition of unconstrained alphanumeric characters" [19].

Nous terminons cette étude en faisant le point sur les deux méthodes proposées et en donnant quelques perspectives pour des recherches futures.

° ° 0

CHAPITRE II.

II. ACQUISITION ET PRETRAITEMENT D'UN CARACTERE.

2.1. ACQUISITION D'UN CARACTERE.

L'acquisition d'un caractère se fait en 2 étapes. Dans la première étape, le microprocesseur de la tablette échantillonne et traite les informations concernant le caractère écrit et il range à titre provisoire ces données traitées dans la mémoire tampon de la tablette. Dans la deuxième étape, la tablette émet les données stockées vers l'ordinateur quand celui-ci est prêt à les recevoir.

Dans notre système de reconnaissance, nous utilisons la tablette x - y de la société OPTION. La tablette a une surface active de taille 10 x 14 cm et une précision d'environ 0.2 mm [50]. A cause de l'utilisation d'une tablette comme terminal d'entrée, il est nécessaire d'assurer l'efficacité de transmission afin de réduire la charge de l'ordinateur sur l'acquisition de ces données. Ceci est accompli en incluant les fonctions de bases telles que le lissage et le filtrage préliminaires, la compression de données et l'isolement de symboles dans la tablette.

2.1.1. Lissage et filtrage préliminaires.

Quand la plume du stylo touche la surface active de la tablette, les signaux représentant la position de la plume sont traités et transformés en coordonnées. Un lissage préliminaire est nécessaire à cause de la présence de parasites. Ceci est accompli en remplaçant les coordonnées par leurs propres moyennes mobiles.

Les nouvelles coordonnées peuvent être redondantes (c'est à dire Ayant les mêmes valeurs que leurs précédents coordonnées quand la vitesse de tracé est trop lente par rapport à la vitesse d'échan-

tillonnage du microprocesseur). Pour éliminer celles qui sont redondantes, nous échantillonnons une nouvelle fois les coordonnées. Dans ce cas, un nouveau point sera retenu si la distance du point précédente est supérieure ou égale à une unité.

2.1.2. *Compression de données.*

Au point de vue de l'efficacité de la transmission de données, il est beaucoup plus efficace d'utiliser les codes de Freeman [10] pour représenter un trait que d'utiliser les coordonnées directement. Les codes de Freeman sont illustrés dans la figure 2.1. Un trait représenté par une chaîne de codes de Freeman a la forme suivante.

$$x_o, y_o, c_1, c_2, \ldots\ldots, c_n$$

d'où x_o, y_o —— les coordonnées du premier point,

et c_i —— le code de Freeman ;

$c_i = \{0, 1, 2, 3, 4, 5, 6, 7\}$.

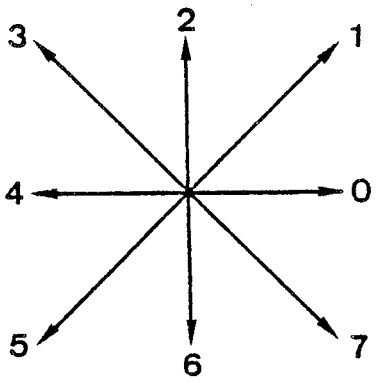


FIGURE 2.1. Les codes de Freeman.

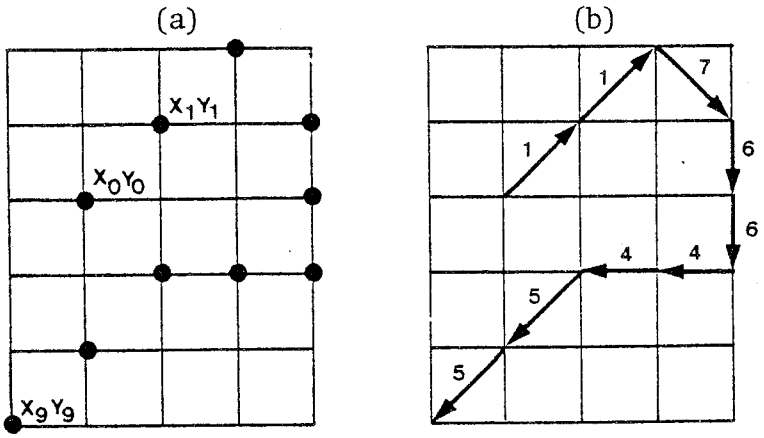


FIGURE 2.2. (a) Un trait digitalisé,
(b) La représentation d'un trait par une chaîne de codes de Freeman.

La figure 2.2. montre la représentation d'un trait par une chaîne de codes de Freeman. On remarque que pour un trait de n points, cette représentation n'a besoin que de n + 1 mots pour le décrire, tandis que si on utilise les coordonnées, il aurait fallu 2 n mots. Par exemple dans la figure 2.2., la longueur de la chaîne de Freeman est de 11 alors que la longueur originale du trait est de 20.

2.1.3. Isolement de caractères.

L'isolement de caractères est lui-même une espèce de reconnaissance, si nous n'imposons aucune contrainte sur le scripteur, il peut devenir un problème plus complexe que celui que nous entreprenons. Une de ces deux contraintes peut être imposée de la façon suivante.

i) - Contrainte temporelle. Un caractère est considéré comme terminé dans le temps lorsqu'il s'est écoulé ΔT seconde depuis le moment où le stylo n'est plus en contact avec la surface active de la tablette. Le scripteur est obligé d'attendre un temps supérieur à ΔT avant d'écrire un autre caractère. Evidemment, si un caractère a plusieurs traits, chaque levée de plume entre 2 traits successifs doit être inférieur à ΔT.

ii) - Contrainte spatiale. Le scripteur doit écrire ses caractères dans une grille imposée.

Dans cette expérience, nous imposons la contrainte temporelle. ΔT dans notre cas est fixé à 300 ms. Au bout des 300 ms, une chaîne de codes signifiant la fin d'un caractère sera introduite dans la mémoire tampon.

2.1.4. Les sorties de la tablette.

Les trois fonctions de bases :

. Lissage et filtrage préliminaires.

. Compression de données.

. Isolement de caractères.

sont actuellement incluses dans la tablette x - y. En général, à la sortie de la tablette x - y, nous avons les messages suivants pour chaque caractère acquis :

$x_{11}, y_{11}, C_{11}, C_{12}, \ldots\ldots\ldots, C_{1n_1}, CR$

$x_{21}, y_{21}, C_{21}, C_{22}, \ldots\ldots\ldots, C_{2n_2}, CR$

$\vdots$

$x_{m1}, y_{m1}, C_{m1}, C_{m2}, \ldots\ldots\ldots\ldots, C_{mn_m}, CR$

L, L, CR

d'où :

$x_{i1}\ y_{i1}$: Les coordonnées du 1er point du $i^{ème}$ trait.

C_{ij} : Code de Freeman.

CR : Retour chariot.

L : Code spécial qui signifie la fin d'un caractère.

2.1.5. La transmission de données sur l'ordinateur.

En réalité, on peut envoyer des messages au fur et à mesure que l'on échantillonne si l'ordinateur sur lequel on travail est en temps réel. Mais dans le cas où l'ordinateur travaille en temps partagé, il faut prendre la précaution de stocker à titre provisoire ces messages dans la mémoire tampon de la tablette et ne les transmettent que si l'ordinateur est prêt à les recevoir, si non, on risque de perdre des données.

2.2. PRETRAITEMENT.

Bien que les données soient filtrées par le filtre temps réel à l'intérieur de la tablette, il est toujours nécessaire de les retraiter afin de supprimer les bruits introduits lors des opérations telles que la levée de plume, l'abaissement de plume et le maintien de la plume sur la surface active. Il est clair que ce genre de bruits ne peut pas être éliminé en temps réel. Nous montrons dans figure 2.3. quelques caractères 'originaux' (caractères non traités) pour illustrer ce type de bruits.

La procédure de prétraitement contient deux étapes. La première concerne le lissage de l'enregistrement de la plume et la seconde, l'élimination éventuelle de petites 'bavures' au début et à la fin de chaque trait.

2.2.1. Lissage d'un trait.

Dans notre système, chaque trait d'un caractère est codé par une chaîne de Freeman. Nous rappelons qu'une chaîne de Freeman bien lissée ne doit pas avoir deux codes voisins dont l'écart de leur valeurs est supérieur à 1 (Modulo 8), à l'exception des endroits

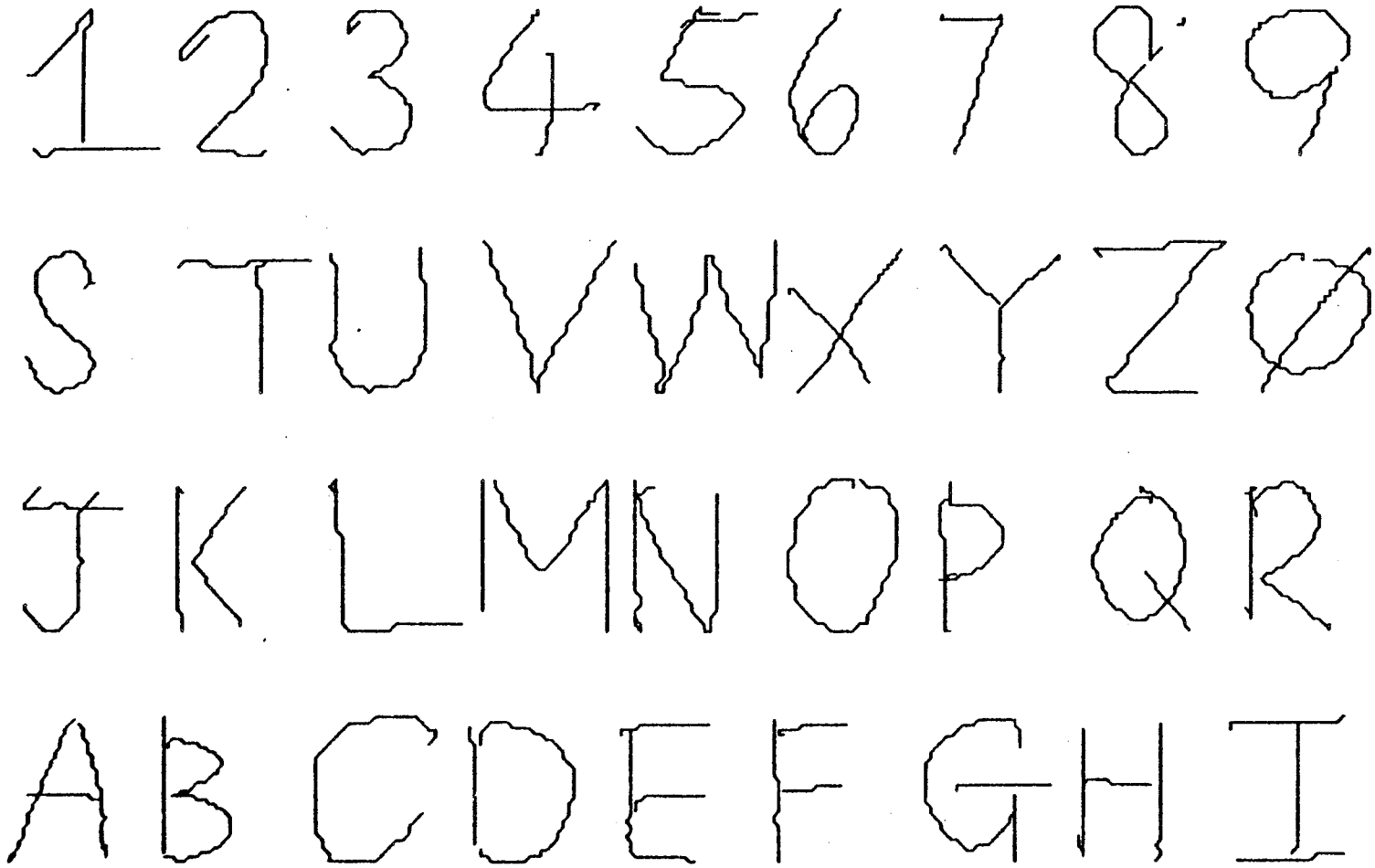


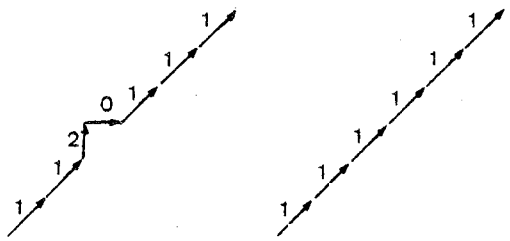
FIGURE 2.3. Les caractères originaux (avant prétraitement).

où un changement de la direction générale existe (par exemple, un coin). Si ce critère n'est pas satisfait, le lissage est nécessaire.

Le lissage doit être à la fois 'local' et 'global'. Par local, nous voulons dire que le lissage se fait en examinant successivement le long de la chaîne, l'écart entre deux codes voisins. Si l'écart est plus grand que 1, nous remplaçons ou supprimons le code en question. Et par global, nous examinons plusieurs codes en même temps pour déterminer la tendance générale. S'il y a un changement, nous conservons au mieux les codes en question. Nous allons considérer quelques exemples pour éclairer notre idée.

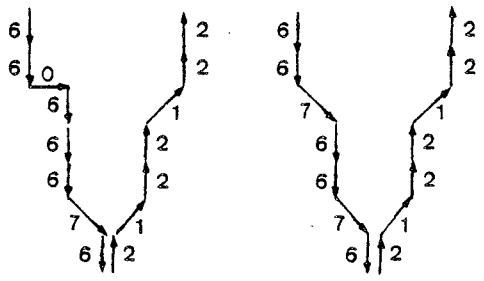
Exemple 2.1.

Nous voyons que les codes 20 doivent être considérés comme des bruits et nous devons les remplacer par un code équivalent à 1.



Exemple 2.2.

De même que l'exemple précédent, les codes 06 doivent être remplacés par 7. Cependant, les codes 62 doivent être conservés car il y a un changement de la tendance générale. La direction générale change de 67 à 12.



En réalité, il est très difficile de formuler un algorithme qui fait à la fois le lissage local et global. Un préalable pour un lissage global est de pouvoir localiser les endroits où il y a des changements de tendances générales. C'est en effet un problème de segmentation qui est considéré comme trop complexe pour être un algorithme de prétraitement.

Pour arriver à faire le lissage d'une manière analytique, nous avons formulé la procédure suivante :

i) - Nous commençons par $i = 1$.

ii) - Si le couple C_i, C_{i+1} est irrégulier[1], nous allons en (iii), sinon, nous allons en (V).

iii) - Le couple irrégulier représente soit un bruit ou un changement de tendance du trait. Pour les distinguer, il faut faire une sorte d'analyse globale. Nous examinons donc en supplément les 2 codes voisins (C_{i-1} et C_{i+2}) du couple en question les relations suivantes :

Soit $C_{i-1}\ C_i\ C_{i+1}\ C_{i+2}$ la chaîne de Freeman à examiner

et ΔXc_j (ΔYc_j) le déplacement en X(Y) du code C_j.

Condition 1 :

$$\Delta Xc_{i-1} = -\Delta Xc_{i+2}$$
$$\Delta Xc_i = -\Delta Xc_{i+1}$$

1. *Un couple de codes voisins $C_1\ C_2$ est irrégulier si $|C_1 - C_2| > 1$ (Modulo 8). Plus précisément si $2 \leq |C_1 - C_2| \leq 6$, sinon, il est régulier.*

Condition 2 :

$$\Delta Yc_{i-1} = - \Delta Yc_{i+2}$$

$$\Delta Yc_{i} = - \Delta Yc_{i+1}$$

Si l'une de ces deux conditions est satisfaite, cela signifie qu'il y a un changement de tendance, nous allons donc en (V), si aucune de ces conditions n'est satisfaite, le couple $C_i\ C_{i+1}$ est donc un bruit, nous allons en (iV).

iV) - Nous calculons :

$$\Delta X = \Delta X\ c_i + \Delta X\ c_{i+1}$$

$$\Delta Y = \Delta Y\ c_i + \Delta Y\ c_{i+1}$$

Si ΔX et ΔY sont zéros, nous diminuons i et nous allons en (ii) sinon, nous avons le tableau suivant où nous pouvons obtenir le(s) code(s) corrigé(s).

ΔY \ ΔX	- 2	- 1	0	1	2
- 2	*	*	66	*	*
- 1	*	5	6	7	*
0	44	4	*	0	00
1	*	3	2	1	*
2	*	*	22	*	*

* - les combinaisons impossibles.

Si nous obtenons 2 codes, nous allons en (V), sinon, nous allons en (ii).

v) - Nous augmentons i = i+1 et nous allons en (ii).

Nous reprenons l'exemple 2.1. La chaîne de Freeman est 1120111 dont le couple 20 est irrégulier.

Nous calculons les déplacements en x et en Y, nous avons :

Code 1	$\Delta X_1 = 1$	$\Delta Y_1 = 1$
Code 2	$\Delta X_2 = 0$	$\Delta Y_2 = 1$
Code 0	$\Delta X_0 = 1$	$\Delta Y_0 = 0$
Code 1	$\Delta X_1 = 1$	$\Delta Y_1 = 1$

Les deux conditions ne sont pas satisfaites, nous calculons donc :

$$\Delta X = \Delta X_2 + \Delta X_0 = 1$$

$$\Delta Y = \Delta Y_2 + \Delta Y_0 = 1$$

Avec le tableau prédédent, nous remplaçons le couple 20 par le code 1.

Dans l'exemple 2.2., la chaîne de Freeman est 660666762122122. Le premier couple irrégulier est 06, avec la même démarche, nous le remplaçons éventuellement par le code 7. Le deuxième couple irrégulier est 62. Ce couple satisfait la condition 2. Donc, nous ne le changeons pas. En réalité, ce couple marque le changement de tendance.

Nous montrons dans la figure 2.4. les caractères de figure 2.3. après lissage.

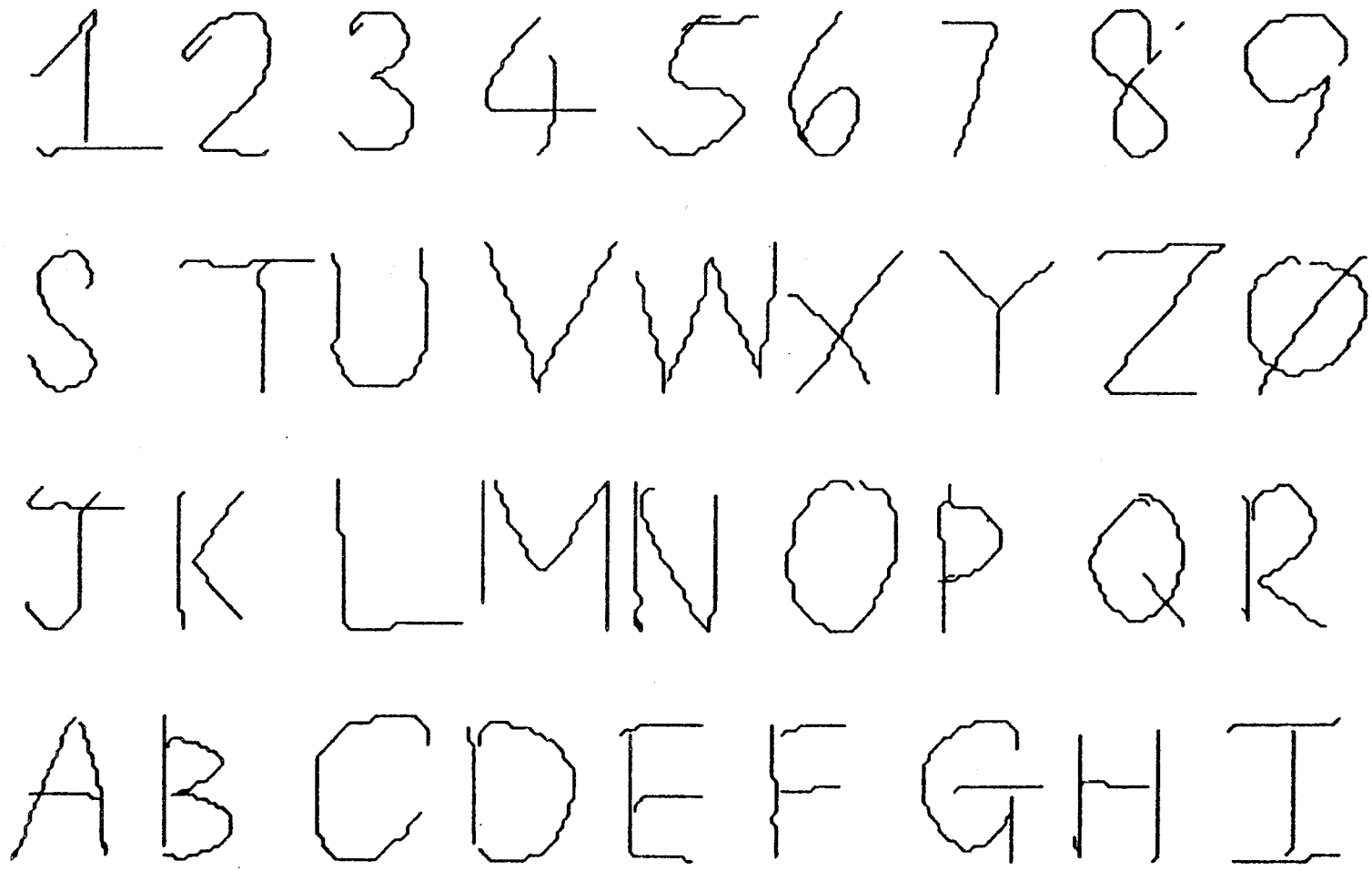


FIGURE 2.4. Les caractères après lissage.

2.2.2. Elimination de Bavures.

Les bavures qui se trouvent au début et à la fin de traits peuvent introduire des ennuis au niveau de la segmentation et la classification. Car dans certains cas, l'algorithme de segmentation les considère comme des segments caractéristiques. Nous aurons dans ces cas des descriptions de caractères très différentes de leurs propres modèles de références. D'autre part, les bavures modifieraient sensiblement les paramètres des vecteurs de caractéristiques ou les attributs des descriptions. En conséquence, ces caractères ne seront pas correctement reconnus au niveau classifieur.

En réalité, nous ne pouvons, qu'après avoir reconnu le caractère, dire si oui ou non un segment au début ou à la fin d'un trait du caractère est une bavure, sauf dans les cas très évidents où la taille de la bavure est très petite par rapport à la taille du caractère. Heureusement, il présente dans la plupart des cas, les caractéristiques intéressantes suivantes :

i) - La dimension d'une bavure est petite par rapport à la dimension du caractère. Un rapport de 1/5 serait approprié.

ii) - La direction générale d'une bavure s'oppose souvent à la direction générale du 'vrai' début d'un trait.

D'ailleurs, nous pouvons constater ces deux propriétés dans les figures 2.3. et 2.4.

Tenant compte de ces deux caractéristiques, nous développons la procédure d'élimination de Bavures de la manière suivante :

i) - On fixe une fenêtre de taille égale à 1/4 la dimension du caractère, à chaque extrèmité du trait à la

fois.

ii) - On calcule la direction générale de la partie du trait qui se trouve dans la fenêtre.

iii) - On commence par un code dans la fenêtre qui est le plus loin de l'extrêmité. On compare la direction de ce code avec la direction générale calculée en (ii). Si ces deux directions s'opposent, on supprime le reste des codes dans la fenêtre, sinon, on continue la comparaison avec le code suivant jusqu'à ce que l'on se trouve à l'extrêmité du trait.

Nous allons considérer un exemple pour mieux expliquer cette méthode.

Exemple 2.3.

La taille du caractère est 14 x 20. Pour calculer la taille de la fenêtre, nous divisons 20 par 4 et nous obtenons la dimension de la fenêtre qui est égale à 5 x 5.

Fenêtre 1.

La direction générale est de Nord à Sud, c'est-à-dire, $\Delta Y = -1$. La partie du tracé à traiter dans l'ordre de traitement est 66666221.

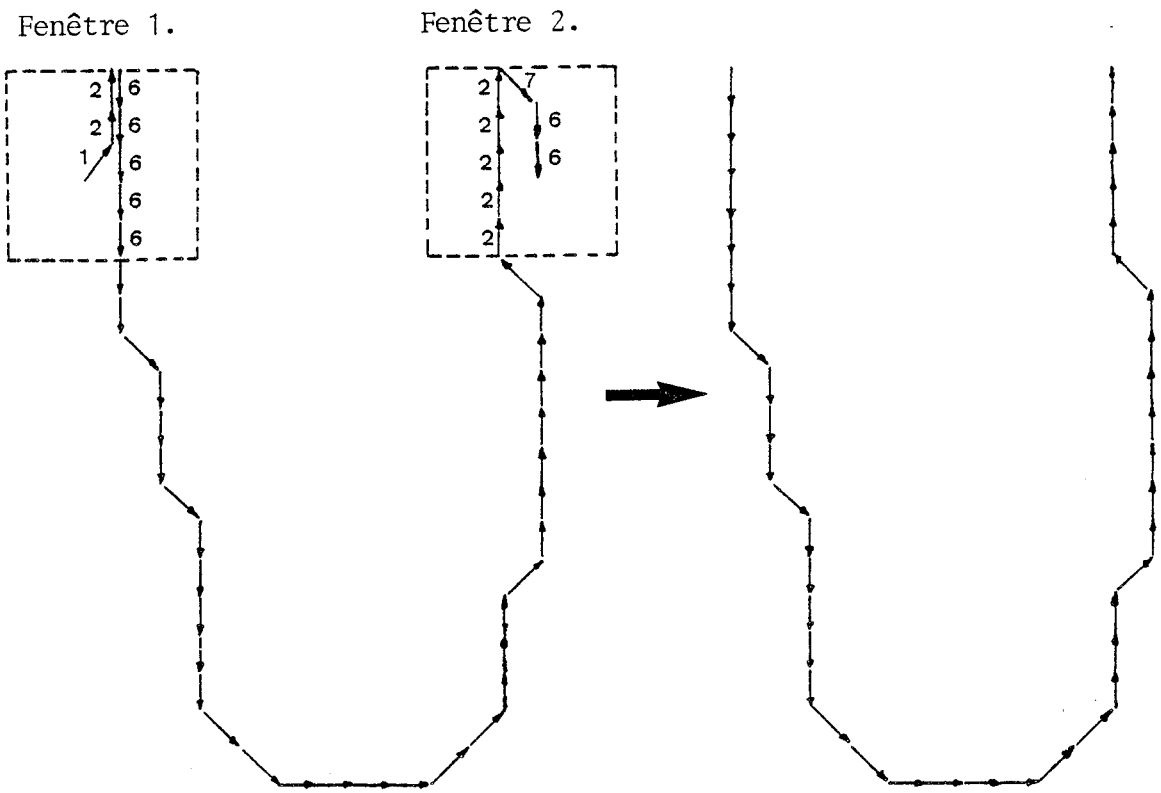
La direction du premier code 6 est la même que la direction générale, on continue donc au suivant. Comme les quatre suivants sont les mêmes que le premier, on arrive donc au sixième code 2. La direction du code 2 est + 1. Dans ce cas là, ces deux directions

s'opposent, on arrête donc le traitement et on supprime la partie 221 qui est considérée comme une bavure.

Fenêtre 2.

La direction générale est de Sud à Nord, donc $\Delta Y = 1$.
La partie à traiter dans l'ordre est 22222766.

Les 5 premiers codes 22222 ont la même direction que la direction générale. Mais quand on arrive au sixième code 7, les directions s'opposent. On supprime donc la partie 766.



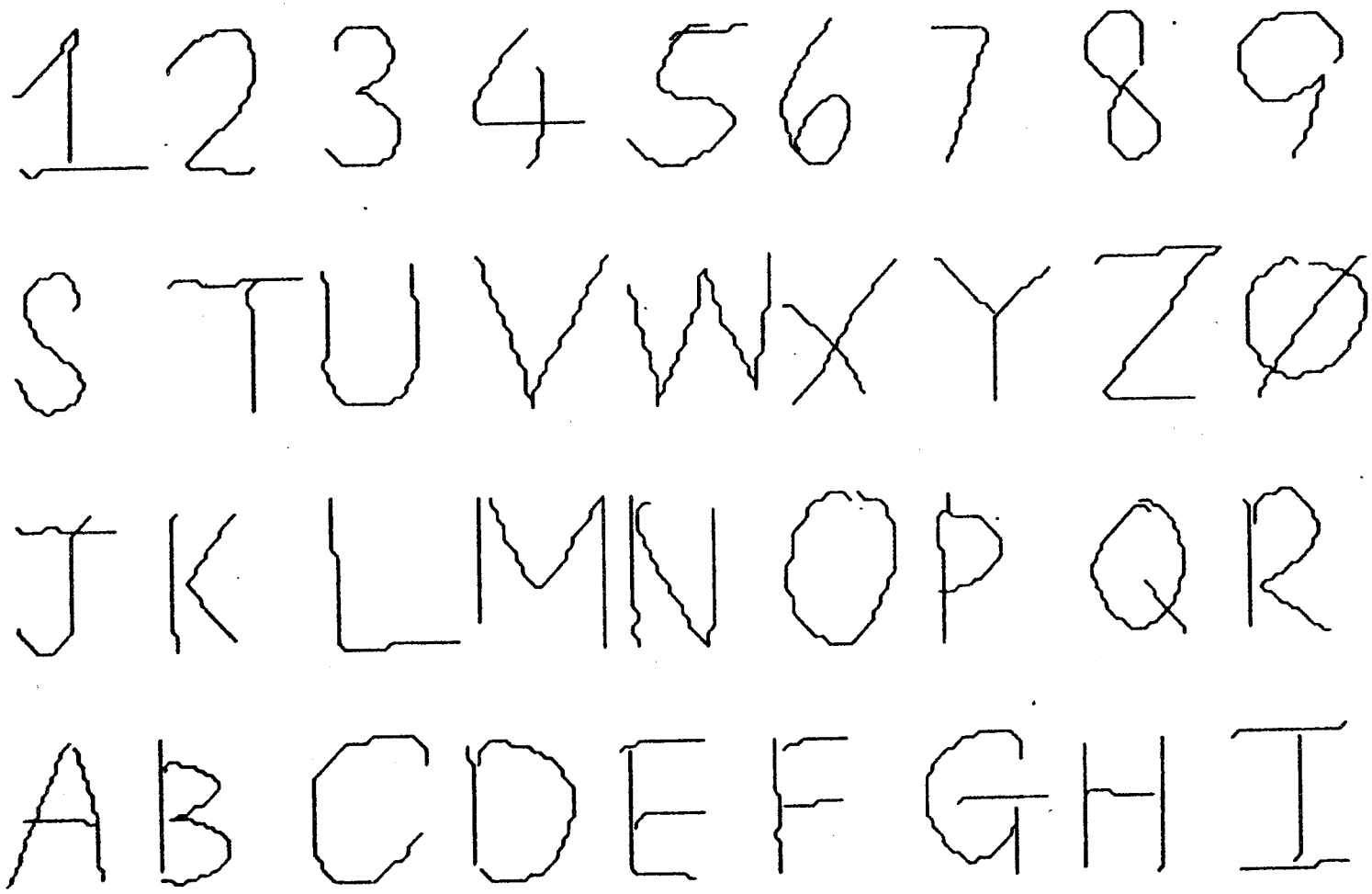


FIGURE 2.5. Les caractères après l'élimination de bavures.

Nous montrons dans la figure 2.5. les exemples des caractères de la figure 2.4. après l'élimination des bavures.

2.3. CONCLUSION.

Nous avons présenté dans ce chapitre la méthode d'acquisition, le filtrage et le lissage de données aux niveaux de la tablette et de l'ordinateur.

Nous voulons pour terminer ce chapitre faire quelques remarques sur l'utilisation de codes de Freeman dans notre système.

. L'aspect temporel du tracé (c'est-à-dire, l'aspect vitesse du tracé) n'est pas conservé dans la représentation de Freeman. Nous considérons ce type d'information peu important dans notre système. Mais dans certains cas de reconnaissances, par exemple, la reconnaissance de signatures, ce type d'information est très utile. Dans ces cas, il ne faut pas utiliser la représentation de Freeman.

. Un trait représenté par une chaîne de Freeman peut être traité "analytiquement' tant au niveau prétraitement qu'au niveau segmentation (nous allons montrer dans la partie 3.2). En effet, nous trouvons le prétraitement plus efficace par la voie analytique. Par la voie normale (lissage par moyenne mobile par exemple), nous serions obligé de travaillé avec des nombres réels ; d'autre part, le caractère après le traitement subirait de distorsion.

° ° °

CHAPITRE III.

III. PROPOSITION N° 1.

3.1. INTRODUCTION.

Le choix des caractéristiques distinctives est le centre du problème de la reconnaissance. Les caractéristiques à choisir dépendent des formes à reconnaître et il n'y a pas de règle générale pour cela. Dans la reconnaissance de caractères, l'une des deux classes de caractéristiques suivantes est souvent utilisée.

. Les coefficients d'une expansion ou transformation. Par exemple, les coefficients de la transformation de Fourier [3, 13, 29].

. Les primitives qui décrivent d'une façon explicite les structures locales d'un caractère. Par exemple, un segment de droite, un coin, etc..., [2, 5].

La première classe de caractéristiques est assez facile à extraire mais sa puissance descriptive est limitée. Par exemple, il serait difficile d'utiliser cette classe de caractéristiques pour reconnaître les caractères écrits sans contraintes. D'autre part, il a fallu un temps de calcul généralement long pour effectuer les transformations ou expansions nécessaires.

La deuxième classe de caractéristiques est de plus en plus utilisée par les chercheurs dans ce domaine, mais le choix des primitives reste toujours un problème difficile à résoudre. Dans la plupart des cas, il faut faire un compromis entre la difficulté d'extraction et la complexité du classifieur.

Avant d'expliquer notre première proposition, nous voulons d'abord définir ce que l'on attend dans cette partie d'étude. Notre

but est de réaliser un système de reconnaissance de caractère à base microprocesseur qui satisfait les critères attendus d'un système interactif. Dans un premier temps, nous allons traiter seulement les caractères alphanumériques manuscrits d'un seul scripteur et plusieurs scripteurs avec contraintes (Monoscripteur sans contraintes et Multiscripteur avec contraintes).

Pour implanter notre algorithme sur un système micro-ordinateur sans avoir trop de difficultés, il faut que l'algorithme soit de nature simple. Ce qui exclut donc toutes les approches utilisant la première classe de caractéristiques. Mais pour que le système ait les aspects interactifs, c'est-à-dire, les possibilités d'apprendre récursivement des caractères nouveaux et de modifier les modèles de références en cours de reconnaissance, il serait plus facile d'employer un classifieur statistique.

Il est possible à priori de décomposer un caractère en une chaîne de segments de droite ou/et de courbes. Dans le cas du monoscripteur, le nombre, le type (droite ou courbe) et la séquence de segments représentent un caractère sont en général invariables pour les caractères de la même identité mais distincts pour les caractères ayant des identités différentes. On peut donc considérer une telle représentation comme des caractéristiques distinctives de caractères alphanumériques dans la reconnaissance en ligne. Le problème d'extraction de caractéristiques est devenu ici un problème de segmentation et d'identification de paramètres.

Avec la procédure de segmentation que nous présenterons en détail dans la section suivante, nous décrivons un caractère entré par une série de segments de droites. Nous trouvons que si les paramètres des segments de droites sont bien normalisés et si la segmentation est régulière, nous pouvons utiliser ces paramètres comme des caractéristiques représentatives du caractère et un classifieur

de Bayes suffit pour identifier les caractères alphanumériques manuscrits.

En réalité, une étude théorique a été conduite pour vérifier nos hypothèses concernant la distribution des paramètres et la séparation entre les différentes classes. Notre approche est aussi justifiée d'après cette étude.

Ce chapitre est organisé de la façon suivante :

Dans la partie 3.2., nous présentons l'algorithme de segmentation que nous avons développé. Nous discutons les diverses méthodes de paramétrisations et le choix d'une meilleure représentation dans la partie 3.3. La partie 3.4. est consacrée aux algorithmes de classification et de l'apprentissage. Les algorithmes récursifs sont également représentés. Dans la partie 3.5., nous vérifions les hypothèses que nous faisons et nous présentons les résultats expérimentaux sur le taux de reconnaissance. Nous terminons ce chapitre en donnant une extension à cette méthode (la partie 3.6.) et en faisant le point sur cette méthode (la partie 3.7.).

3.2. SEGMENTATION D'UN CARACTERE.

Notre idée est de construire le vecteur de caractéristiques directement à partir du résultat de la segmentation. Ceci implique qu'un caractère segmenté utilisant un algorithme de segmentation doit possèder les propriétés suivantes :

- Le nombre de segments d'un caractère segmenté est le minimum nécessaire.

- Le nombre de segments est régulier pour les caractères de la même identité.

En réalité, il n'est pas possible de satisfaire ces deux critères parfaitement. Nous pouvons seulement essayer de réaliser un algorithme de segmentation qui les satisfait au mieux.

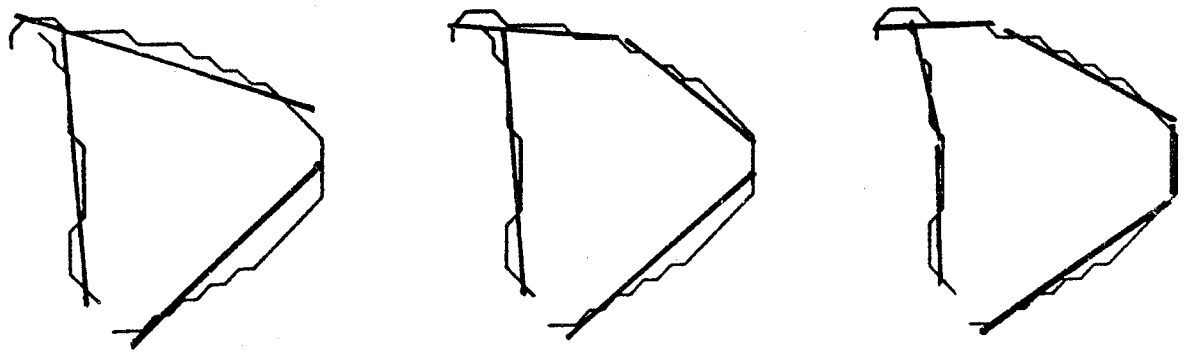
Les algorithmes de segmentations proposés dans la littérature [26, 28, 32] ne sont pas adaptés à notre problème. Dans tous ces algorithmes d'approximations, on cherche à minimiser un certain critère ou à satisfaire un certain seuil. Donc, le nombre de segments que l'on obtient dépend du critère ou du seuil que l'on fixe au départ. Nous donnons quelques exemples de segmentations obtenues avec les algorithmes de Pavlidis [26], de Ramer [28] et de Sklansky [32] utilisant différentes valeurs de critères ou seuils dans la figure 3.1. pour illustrer ce problème. A part la variation sur le nombre de segments, un autre problème est que les sommets du polygone qui représentent le caractère segmenté n'ont pas de signification particulière, ils ne peuvent donc pas être utilisés comme des caractéristiques du caractère.

En examinant les échantillons des 36 caractères alphanumériques, nous remarquons que si une segmentation est faite avec les points tournants[1], le caractère segmenté possèdera les propriétés indiquées précédemment. Nous montrons dans la figure 3.2. quelques exemples des caractères segmentés suivant cette idée. De cette figure, nous pouvons en effet constater que les caractères ainsi segmentés possèdent ces propriétés. Nous allons présenter dans la suite cet algorithme de segmentation.

La procédure commence par une recherche des points tournants. Une segmentation initiale est obtenue en découpant les traits en segments déterminés par ces points (Comprendre le premier et le dernier point de chaque trait). La recherche des points tournants peut être accomplie très facilement car les traits sont représentés par les chaînes de Freeman. Nous allons considérer un exemple pour montrer cette démarche.

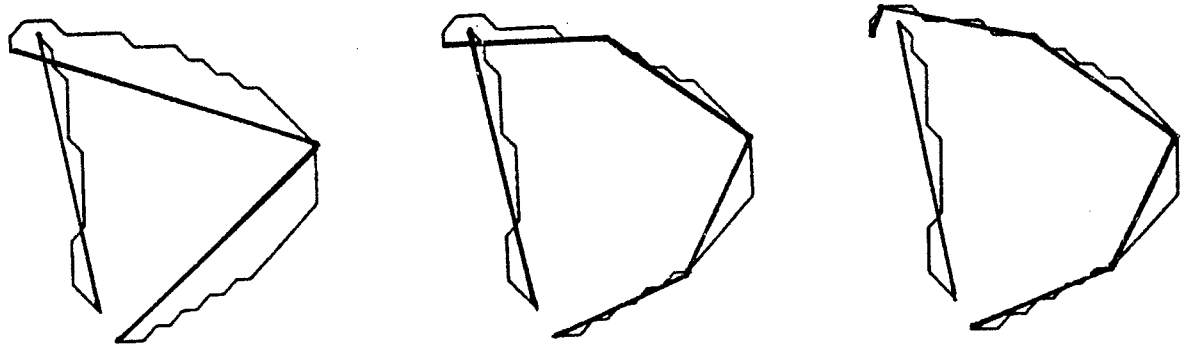
1. *Un point tournant est un point sur une courbe où les signes des tangentes à gauche et à droite de ce point se trouvent inversés.*

E = 30 E = 15 E = 10



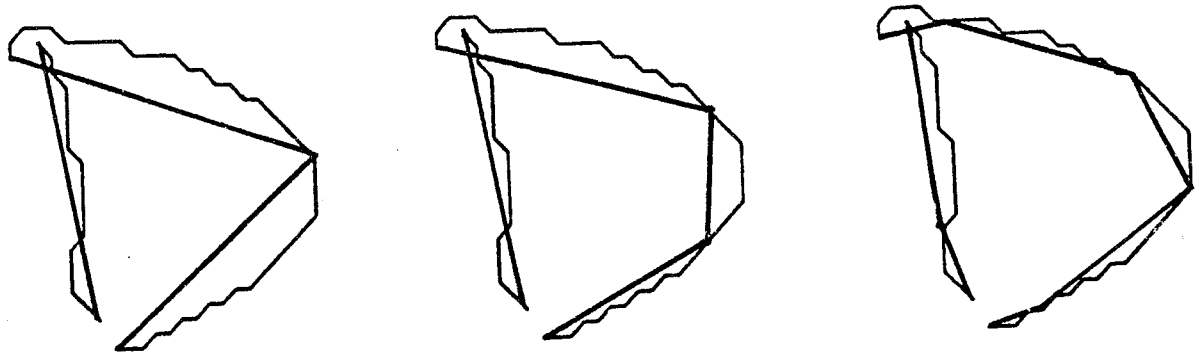
(a) Segmentation par l'algorithme de PAVLIDIS.

S = 4 S = 3 S = 2



(b) Segmentation par l'algorithme de RAMER.

S = 3 S = 2 S = 1



(c) Segmentation par l'algorithme de SKLANSKY.

FIGURE 3.1. Segmentation de caractères.

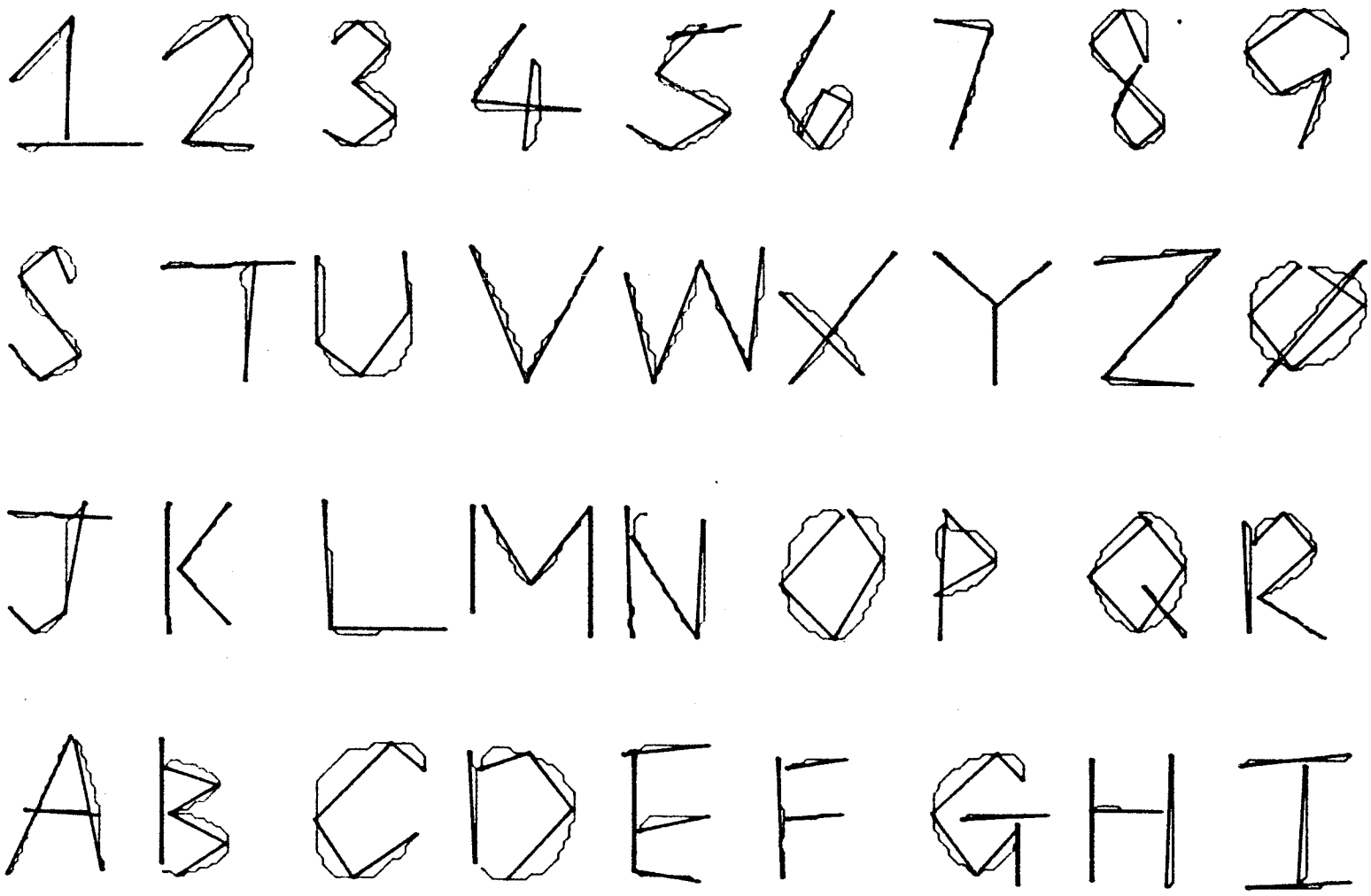
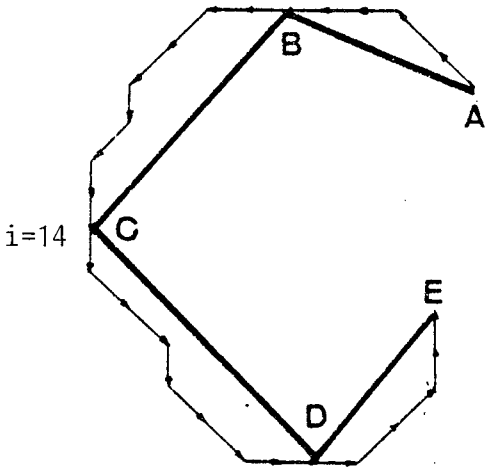


FIGURE 3.2. Les caractères segmentés.

Dans la figure, il y a 3 points tournants : B, C et D. La méthode pour déterminer B, C et D est la suivante :

i) - On commence par le 1er code, i = 1. On stocke ΔX_{sx} et ΔY_{sy}, SX = SY = 1.



ii) - On augmente i, i = i + 1, si i > n, on arrête la procédure.

iii) - Si $\Delta X_i = 0$, on va en (iv)
sinon et si $\Delta X_i = -\Delta X_{sx}$, on va en (v)
sinon, on met SX = i et $\Delta X_{sx} = \Delta X_i$ et on va en (iv)

iv) - Si $\Delta Y_i = 0$, on va en (ii)
sinon et si $\Delta Y_i = -\Delta Y_{sx}$, on va en (v)
sinon, on met SY = i et $\Delta Y_{sy} = \Delta Y_i$ et on va en (ii)

v) - Il y a un point tournant, l'index de point tournant est :

j = (SX ou SY + i)/2

On met $\Delta X_{sx} = \Delta X_i$

$\Delta Y_{sy} = \Delta Y_i$

et $SX = SY = i$ et on va en (ii)

Il est facile avec cet exemple de vérifier les relations suivantes :

Pour $i = 14$, $\Delta X_i = 0$, $\Delta Y_i = -1$, $\Delta X_{sx} = -1$, $\Delta Y_{sy} = -1$,

$SX = 11$ et $SY = 14$

Pour $i = 15$, $\Delta X_i = 1$ $\Delta Y_i = -1$ $\Delta X_{sx} = -1$, $\Delta Y_{sy} = -1$,

on voit que $\Delta X_i = -\Delta X_{sx}$, il y a donc un point tournant, l'index du point tournant C est donné par

$j = (11 + 15)/2 = 13$.

La procédure de segmentation que nous veons d'expliquer a les deux défauts suivants :

. Pour certains caractères, cet algorithme ne peut pas donner le nombre de segments nécessaires. Ceci est dû au fait que nous segmentons les caractères par les points tournants, or, pour certains caractères, ces formes particulières ne présentent pas de point tournant, nous trouvons donc moins de segments que nécessaire. Par exemple, pour le caractère L, en principe, il n'existe pas sur le contour de ce caractère de point tournant. Comme résultat, ce caractère sera représenté par un seul segment de droite qui lie le premier et le dernier point du caractère, alors que la segmentation correcte doit avoir deux segments avec le sommet en bas à gauche du caractère.

Les autres caractères que nous rencontrons, posant le même problème sont les caractères U, E et Y.

. Cet algorithme donne parfois plus de segments que nous devons habituellement obtenir en présence de bruits. Par exemple, les segments voisins qui sont presque parallèles l'un à l'autre.

Nous montrons dans la figure 3.3. quelques exemples de caractères après cette segmentation initiale.

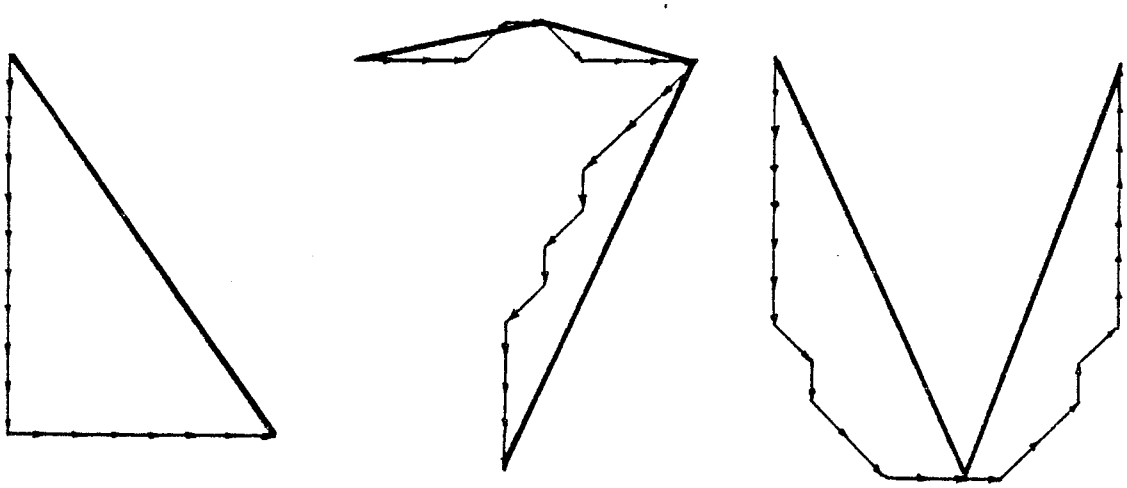


FIGURE 3.3. Les exemples des caractères après la segmentation initiale (—— : droite d'approximation après la segmentation initiale).

Nous allons corriger ces deux défauts par les opérations suivantes :

. Une seconde recherche est conduite, mais cette fois-ci ;

elle est au niveau de chaque segment. On cherche le point (sur le trait segmenté) qui est le plus éloigné du segment de droite d'approximation. La distance de ce point à la droite d'approximation est comparée avec un seuil. Si la distance est supérieure au seuil, le segment sera coupé en deux en utilisant ce point comme un nouveau point tournant. Afin de simplifier les calculs des distances, nous calculons, selon la pente de la droite d'approximation, la distance verticale ou la distance horizontale pour remplacer la distance normale. Un exemple est montré (figure 3.4).

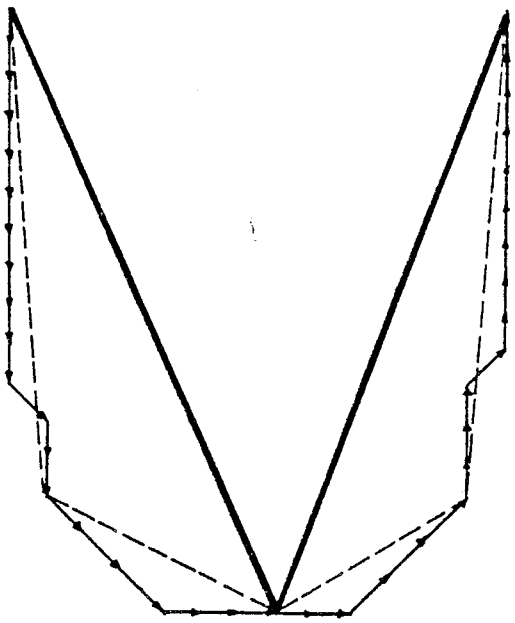


FIGURE 3.4.
La segmentation d'un caractère après la 1ère correction.
——— : droite d'approximation après la Segmentation initiale
---- : droite d'approximation après la 1ère correction.

. La deuxième façon est une opération de vérification. La segmentation obtenue après la première opération est vérifiée dans cette étape afin d'éliminer les points tournants rédondants. L'élimination est effectuée si l'une des deux conditions ci-après est satisfaite :

i) - l'angle obtus entre deux segments de droites d'approximations voisins est plus grand que 160 °.

ii) - la longueur totale des deux segments voisins est plus petite que la hauteur du caractère et ses pentes se trouvent dans le même quadrant.

3.3. PARAMETRISATION D'UN CARACTERE SEGMENTE.

Par la procédure de segmentation, un caractère est représenté par une chaîne de droites. L'examen visuel sur les caractères alphanumériques dans cette représentation (cf. figure 3.2. par exemple) nous donne l'idée que si le nombre, la séquence et les paramètres de segments de droites sont utilisés pour former un vecteur de caractéristiques particulier pour chaque caractère, il est possible de les séparer. Le problème maintenant est donc de choisir les paramètres appropriés.

Pour simplifier, nous allons limiter le nombre de paramètres à utiliser pour représenter un segment de droite à deux. Il y a généralement deux types d'équations de droites :

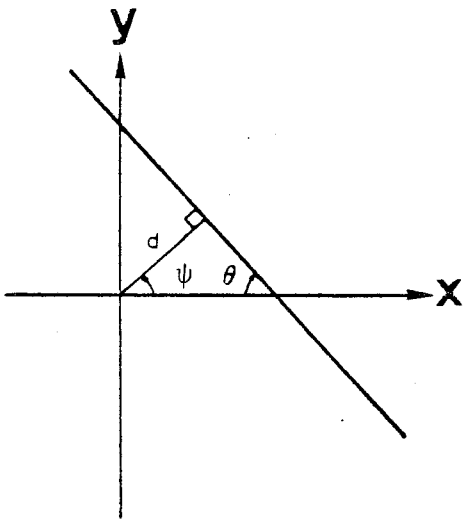
1er type : $y = ax + b$.

. Il est connu que l'on ne peut pas utiliser les coefficients a et b comme paramètres car ils ne sont pas des variables normalisées, d'autre part, il est très difficile de tenir compte des discontinuités présentés dans les distributions des paramètres.

2e type : $x \cos\psi + y \sin\psi = d$

ou $x \sin\theta + y \cos\theta = d$

Les définitions de ψ, θ et d sont indiquées dans la figure.



Les coefficients de ce type d'équations sont des coefficients normalisés. ψ et θ varient de $-\pi/2$ à $\pi/2$ et d est borné. Mais le problème de discontinuité existe. Nous considérons un exemple pour illustrer ce problème.

Exemple 3.1. (cf. figure 3.5).

Considérons 4 droites l_1, l_2, l_3 et l_4. Physiquement, les deux droites l_1 et l_2 sont très proches l'une de l'autre et les deux droites l_3 et l_4 également (sur le plan x - y). Mais nous voyons que si nous utilisons l'équation $x \sin\theta + y \cos\theta = d$ pour décrire ces droites, la distribution des paramètres θ et d ne met pas en évidence cette situation. Sur le plan θ - d (cf. figure 3.5.e), la distance entre les droites l_1 et l_2 est en effet très loin et les droites l_3 et l_4 aussi.

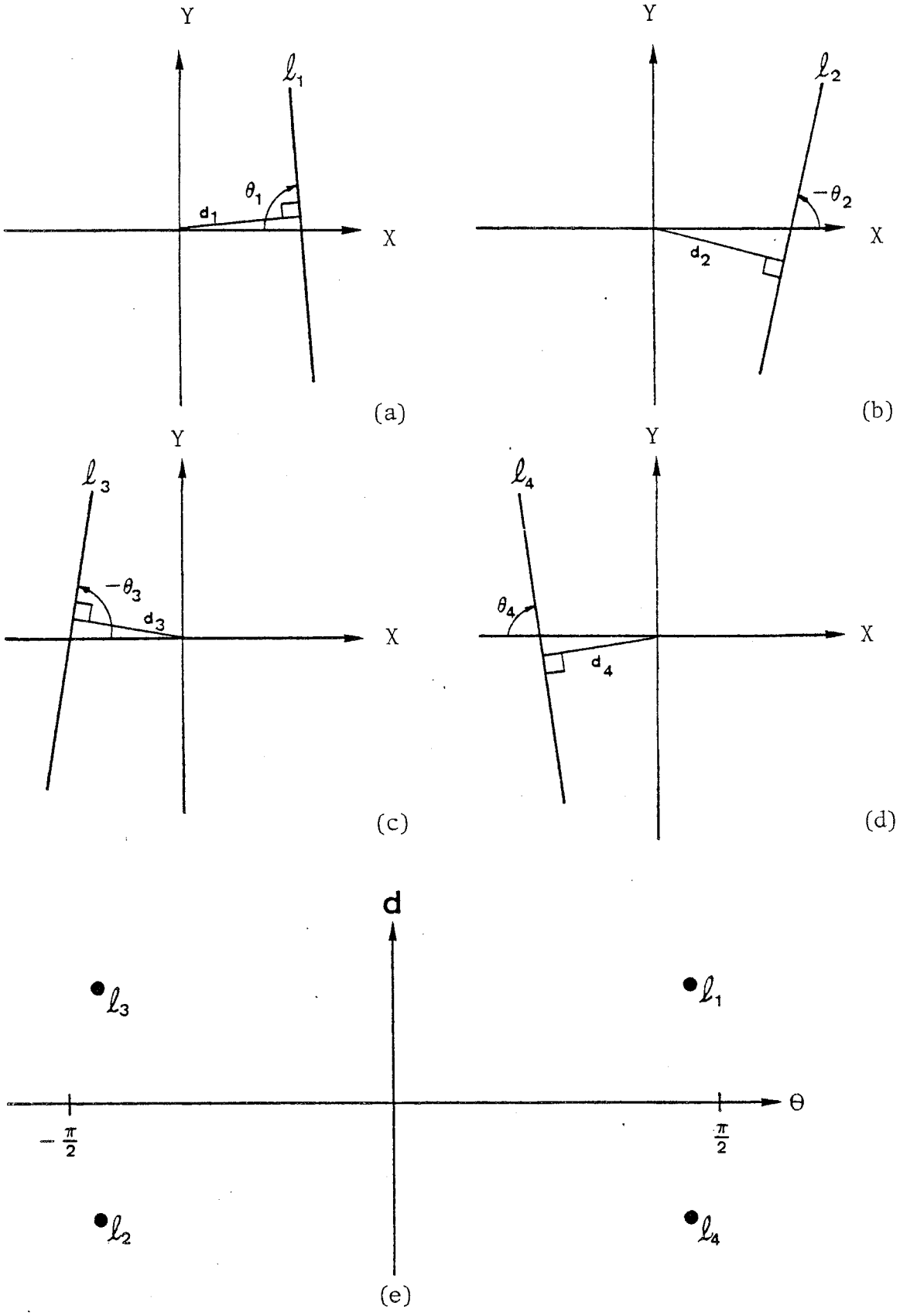


FIGURE 3.5. La distribution de θ et d.

En réalité, ce problème de discontinuité ne peut pas être supprimé, il existe dans tous les types d'équations. Mais il est possible de transformer une équation pour que la discontinuité apparaisse dans les différents cas. Dans l'exemple précédent, la discontinuité arrive quand nous avons un groupe de droites proches de la verticale mais si nous remplaçons θ par $\psi = 90 - \theta$, dans ce cas, la discontinuité arrivera pour des droites proches de l'horizontale.

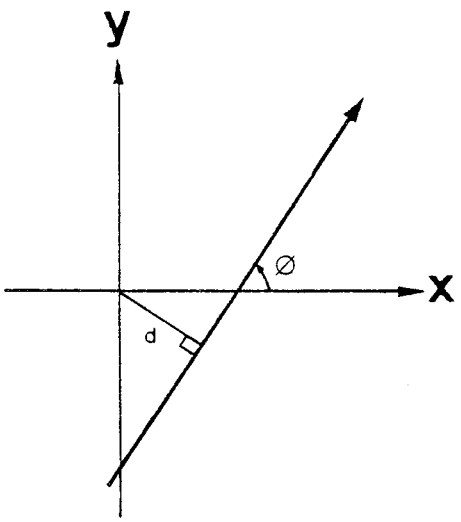
Nous voulons que les deux paramètres qui représentent une droite d'approximation comportent aussi l'information sur le sens du tracé. D'ailleurs, nous voulons que la discontinuité soit limitée au cas où les droites se trouvent pratiquement à l'horizontale avec comme direction de droite orientée vers la gauche.

L'équation de droite est donnée par :

$$x \cos\psi + y \sin\psi = - d$$

$$\psi = \phi - \pi/2$$

ϕ est l'angle entre le segment de droite et l'axe horizontal, ϕ est positive si le segment est tracé de bas en haut et vice-versa. d est la distance verticale du segment de l'origine des axes de références.



Nous traçons dans la figure 3.b la distribution de ψ et d de quelques segments de droites significatifs pour montrer les propriétés de cette description. Nous pouvons constater qu'avec

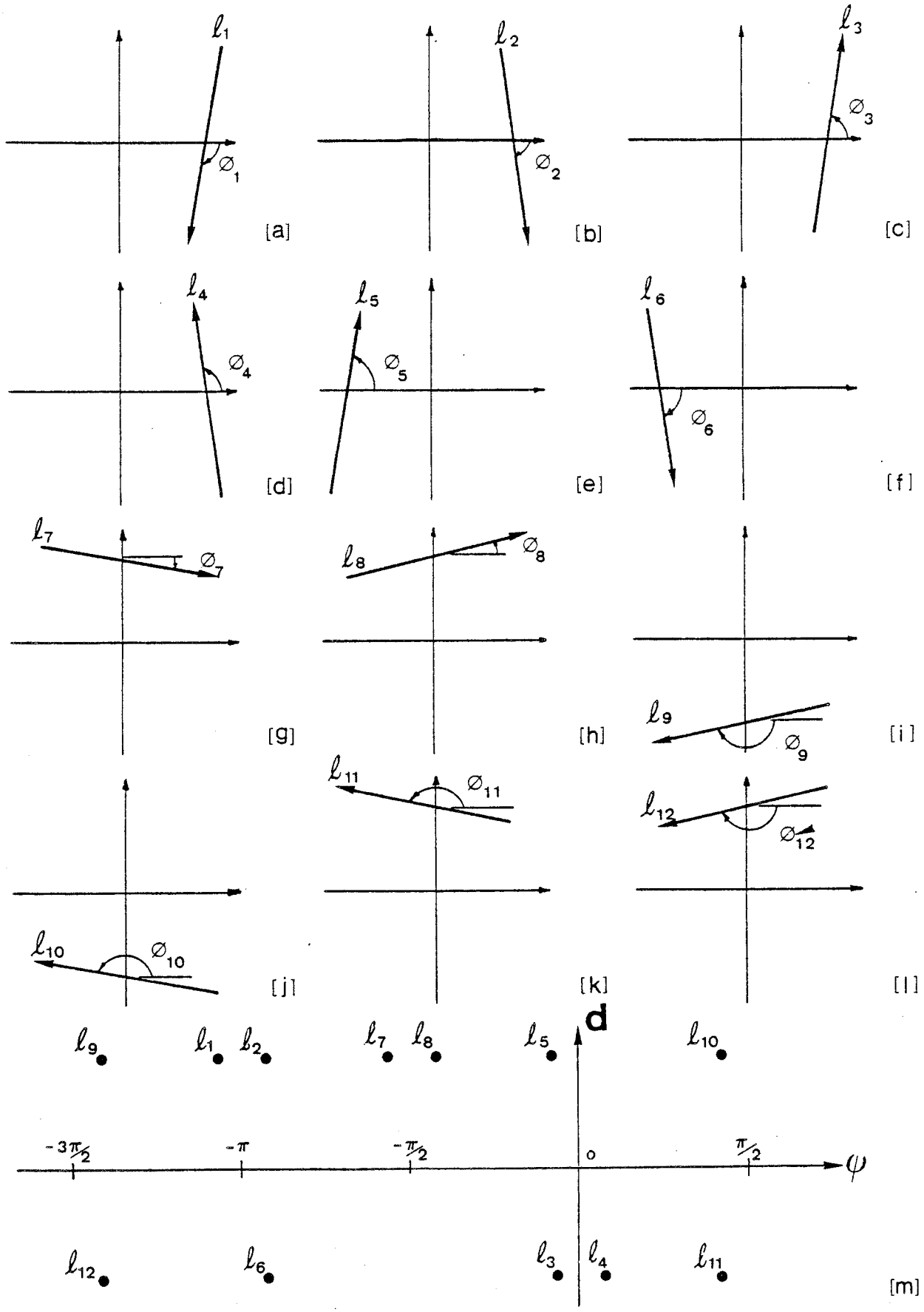


FIGURE 3.6. La distribution de ψ et d ($\psi = \phi - \pi/2$)

cette description seule le groupe de droites avec $|\phi| \simeq \pi$ (l_9 l_{10} et l_{11} l_{12}) connaît des discontinuités sur le plan ψ - d. En réalité, nous rencontrons rarement ce type de droites d'approximation dans les caractères alphanumériques segmentés que nous considérons.

Le choix des types de paramètres est une étape très importante de l'extraction de caractéristiques. Surtout dans le cas ou le classifieur est de nature statistique. Car pour calculer les moyennes et la matrice de covariance, il faut que les paramètres des vecteurs de caractèristiques soient normalisés et que la distribution des paramètres soient normale.

θ et d de chaque droite d'approximation peuvent être déterminés approximativement en utilisant les points tournants ou mieux encore avec un algorithme récursif d'identification (cf. Appendice A). Une fois que les paramètres de chaque segment du caractère segmenté sont définis, nous construisons le vecteur de caractéristiques comme suit :

$$X^T = (\psi_1, d_1, \psi_2, d_2, \ldots\ldots, \psi_n, d_n) \text{ ---------- } (1)$$

Pour savoir si notre choix est bon et si la description avec un vecteur de ce type est suffisante, une étude détaillée a été conduite. Les résultats de cette étude nous montrent que le vecteur de caractéristiques que nous considérons est sufficant pour séparer tous les 36 caractères alphanumériques dans le cas d'un seul scripteur et dans le cas d'un nombre limité de façon d'écrire (Multiscripteur avec contraintes). Nous discuterons plus en détail sur ce point dans la section 3.5. suivante.

3.4. CLASSIFICATION ET APPRENTISSAGE.

La classification comporte deux étapes. La première classifie le caractère inconnu dans une classe par la dimension du vecteur de caractéristiques. La seconde identifie la catégorie qui représente le caractère inconnu en utilisant une fonction discriminante. Pour la plupart des scripteurs, le nombre de segments de droites nécessaire pour approximer un caractère alphanumérique varie de 1 à 6. Donc, souvent 6 classes seulement sont nécessaires.

L'examen bibliographique sur la reconnaissance de formes [36, 40] indique que l'on peut supposer que la distribution des vecteurs de caractéristiques est multivariable normale si les paramètres des vecteurs de caractéristiques sont proprement normalisés. Nous pouvons utiliser le critère suivant pour classifier nos caractères.

$$P(X) = \frac{1}{(2\pi)^{n/2}\,|\sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\underline{X} - \underline{M})^T \underline{\sigma}^{-1}(\underline{X} - M)\right\} \quad (2)$$

$\underline{M}$: vecteur moyen

$\underline{\sigma}$: la matrice de covariances

Le vecteur moyen M et la matrice de covariances peuvent être estimés à partir des échantillons en utilisant les équations suivantes :

$$\underline{M}_N = \frac{1}{N} \sum_{j=1}^{N} \underline{X}_j \quad (3)$$

$$\underline{\sigma}_N = \frac{1}{N} \sum_{j=1}^{N} (\underline{X}_j - M_N)(\underline{X}_j - M_N)^T \quad \text{---------- (4)}$$

L'une des possibilités qu'il faut exploiter dans un système de reconnaissance interactive est que chaque caractère reconnu correctement est utilisé pour remettre à jour les valeurs de $\underline{M}$ et $\underline{\sigma}$. La mise à jour de ces valeurs peut être effectuée récursivement. D'ailleurs, $\underline{\sigma}^{-1}$ qui se trouve dans le critère peut être mise à jour directement sans passer par l'inversion de la matrice $\underline{\sigma}$.

Les formes récursives des équations (3) et (4) sont données par :

$$\underline{M}_{N+1} = \frac{1}{N+1} (N \underline{M}_N + \underline{X}_{N+1}) \quad \text{---------- (5)}$$

$$\underline{\sigma}_{N+1} = \frac{N}{N+1} \underline{\sigma}_N + \frac{1}{N} Z_{N+1} Z_{N+1}^T \quad \text{---------- (6)}$$

et $$\underline{\sigma}^{-1}_{N+1} = \frac{N+1}{N} \left(\underline{\sigma}_N^{-1} - \frac{\underline{\sigma}_N^{-1} Z_{N+1} Z_{N+1}^T \underline{\sigma}_N^{-1}}{\frac{N^2}{N+1} + Z_{N+1}^T \underline{\sigma}_N^{-1} Z_{N+1}}\right) \quad \text{---------- (7)}$$

ou $$Z_{N+1} = (\underline{X}_{N+1} - M_{N+1})$$

Le calcul de l'équation (6) et de l'équation (7) est donné dans l'appendice B.

A vrai dire, avec une analyse des valeurs de $\underline{\sigma}$ générées par un nombre de scripteurs, il est possible d'initialiser une valeur appropriée pour $\underline{\sigma}^{-1}$ et puis d'utiliser ces équations récursives.

pour l'apprentissage.

Le classement dans une catégorie peut se faire avec l'une des conditions discriminantes suivantes :

i) - Condition Discriminante 1 (C.D.1). L'équation (2) est utilisée avec $\underline{\sigma}$ simplifiée à une matrice de bloc diagonale de dimension 2 x 2. C'est-à-dire, nous supposons qu'il n'y a pas de corrélation parmi les segments de droites d'approximations d'un caractères. La fonction discriminante est donnée par :

$$D_1 (x, i, j) = \frac{1}{(2\pi)^{n/2} |\underline{\sigma}_{i,j}|^{1/2}} \exp \{- \frac{1}{2} (\underline{X} - \underline{M}_{i,j})^T \underline{\sigma}_{i,j}^{-1} (\underline{X} - M_{i,j})\} \text{ -- (8)}$$

$\underline{M}_{i,j}$: moyen du vecteur de caractéristiques de la $j^{ème}$ catégorie, $i^{ème}$ classe.

$\underline{\sigma}_{i,j}$: matrice de covariances de la $j^{ème}$ catégorie, $i^{ème}$ classe.

ii) - Condition Discriminante 2 (C.D.2). Nous utilisons seulement l'exposant du terme exponentiel de l'équation (8). C'est-à-dire :

$$D_2 (x, i, j) = (\underline{X} - \underline{M}_{i,j})^T \underline{\sigma}_{i,j}^{-1} (\underline{X} - \underline{M}_{i,j}) \text{ -------- (9)}$$

iii) - Condition Discriminante 3 (C.D.3). La même équation que C.D.2. sauf que $\underline{\sigma}$ est à nouveau réduite à une matrice diago-

nale. Cette condition est souvent appelée la distance normalisée.

La décision est accomplie par une comparaison de valeurs données par la fonction discriminante D(x, i, j) de toutes les catégories dans la $i^{ème}$ classe déterminée auparavent utilisant la dimension du vecteur de caractéristiques $\underline{X}$. Si la fonction discriminant D(x, i, k) a une valeur maximum (pour C.D.1) ou une valeur minimum (pour C.D.2 et C.D.3) de D(x, i, j) pour toutes les catégories j de la $i^{ème}$ classe, alors le caractère entrée est classifié avec l'identité de la $K^{ème}$ catégorie de $i^{ème}$ classe.

3.5. VERIFICATION DES HYPOTHESES ET RESULTATS.

3.5.1. Vérification des hypothèses.

L'expérience est conduite sur une base de données créée par un sujet. On lui a demandé d'écrire 100 échantillons pour chacun de 26 lettres de l'alphabet. Donc 2600 échantillons de caractères alphabétiques sont disponibles. La taille moyenne est d'environ 5 x 7 mm. Aucune contrainte particulière n'a été imposée sur sa façon d'écrire durant la génération des échantillons.

Chaque échantillon est d'abord classé dans une classe déterminée par la dimension de son vecteur de caractéristiques. Et puis nous effectuons l'analyse en composantes principales séparément pour chaque classe d'échantillons. Cette analyse doit être effectuée séparément car il est nécessaire d'avoir toujours le même nombre de variables pour chaque échantillon.

Les résultats montrent que le type de vecteur de caractéristiques considérés dans notre approche sont appropriés et suffisants. En fait, il est possible de séparer tous les caractères dans

un espace de dimension 4 maximum (les 4 axes principaux). De plus, sur les plans principaux, les échantillons de la même identité tombent tous dans un elliposoïde (cf. Appendice C et [43]). Il nous semble donc que notre hypothèse concernant la distribution des paramètres de vecteur de caractères est justifiée.

Cette expérience est très importante car elle nous aide à comprendre les points suivants :

- Le vecteur de caractéristiques proposé est approprié, au moins pour le cas du monoscripteur.

- Les fonctions discriminantes sélectionnées sont propres pour notre système.

3.5.2. *Taux de reconnaissance*.

Nous avons étudié dans cette expérience la performance du système entier sous diverses conditions d'apprentissage et avec les trois types de classifieurs. Deux bases de données sont créées afin que toutes nos techniques de reconnaissance puissent être comparées sur la même suite de caractères.

3.5.2.1. *Les descriptions des bases de données Monoscripteur*.

Les deux bases de données sont créées par le même sujet. La taille moyenne des caractères est de 5 x 7 mm. Dans la première base de données, le sujet écrit 24 échantillons pour chacun de 36 caractères alphanumériques. A cause du fait que certains caractères peuvent avoir des formes et des séquences de tracé différentes (par exemple, 1 peut s'écrire comme 1 ou 1 et le caractère T peut

s'écrire avec la barre horizontale d'abord au lieu de la barre verticale, etc...), on demande donc au sujet d'épuiser les variations habituelles de son écriture. 33 variations avec 24 échantillons chacune sont ainsi stockées. Donc, un total de 1656 caractères avec 24 échantillons pour chacune de 69 manières d'écrire les 36 caractères alphanumériques sont disponibles comme base d'apprentissage (abrév. Base - Lo - Ap).

La deuxième base de données est une base de test (abrév. Base - Lo - Te) qui contient 1000 échantillons, il y a environ 27 échantillons pour chacun de 36 caractères alphanumériques. La contrainte qui a été imposée sur le scripteur est que seulement les 69 manières sont permises. Les exemples des échantillons sont données dans l'Appendice D.

3.5.2.2. *Les essais*.

Un ensemble provenant de la base d'apprentissage et qui contient N échantillons pour chacune de 69 manières d'écrire est utilisé pendant la phase d'apprentissage. Pour étudier l'effet du nombre d'échantillons d'apprentissage sur le taux de reconnaissance, les essais suivants ont été effectuées pour N varie de 2 à 20.

i) - Pour chaque valeur de N, les 3 conditions discriminantes sont testées, d'abord avec la base d'apprentissage puis avec la base d'essai. Les résultats obtenus sont résumés respectivement dans la figure 3.7. et la figure 3.8.

ii) - Pour chaque valeur de N, les 3 conditions discriminantes sont testées avec l'apprentissage en ligne sur la base d'essai. C'est-à-dire, chaque fois qu'un caractère est correctement reconnu, le vecteur moyen M

et la matrice de covariances $\underline{\sigma}$ de modèle correspondant seront mis à jour (cf. Figure 3.9).

iii) - La base d'apprentissage est modifiée pour inclure un mauvais échantillons entre l'intervalle 12 et 14 ou plus clairement, le 13ème échantillon est modifié pour que sa séquence de tracé soit différente des autres échantillons. Le même essai que (i) est effectué sur la base d'apprentissage. Les résultats sont résumés dans la figure 3.10.

Notons que dans les cas où N est petit, il est nécessaire d'initialiser la matrice de covariances. Nous mettons les variances aux mêmes ordres de grandeur que ceux produits par le multiscripteur.

Nous avons quelques remarques suivantes sur les résultats obtenus :

. Dans ce système, si nous voulons avoir un taux de reconnaissance acceptable, le nombre d'échantillons ne doit pas être inférieur à 4.

. De figure 3.7. Nous voyons que la condition discriminante 3 (C.D.3) donne des meilleurs taux de reconnaissance que C.D.1 et C.D.2. Mais ce classifieur est moins sûr car il est plus susceptible aux bruits (un mauvais échantillon) comme illustré figure 3.10.

. La supériorité de C.D.1 ou C.D.2 sur C.D.3 est montrée plus clairement dans la figure 3.8. et la figure 3.9.

. Une amélioration assez nette sur le taux de reconnaissance est obtenue dans le cas de l'apprentissage en ligne.

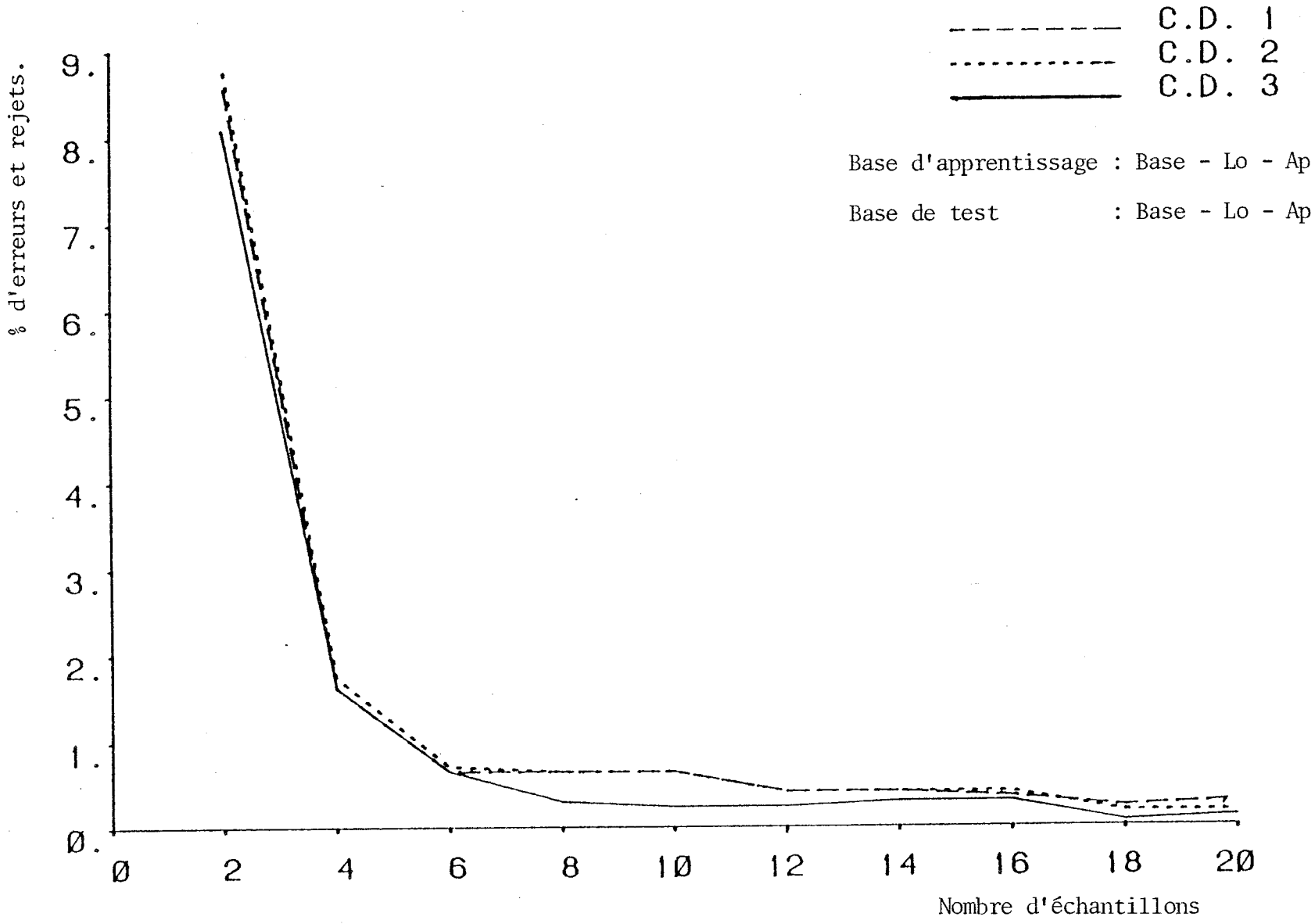


FIGURE 3.7. Taux d'erreurs et rejets en fonction du nombre d'échantillons sur la base de données d'apprentissage.

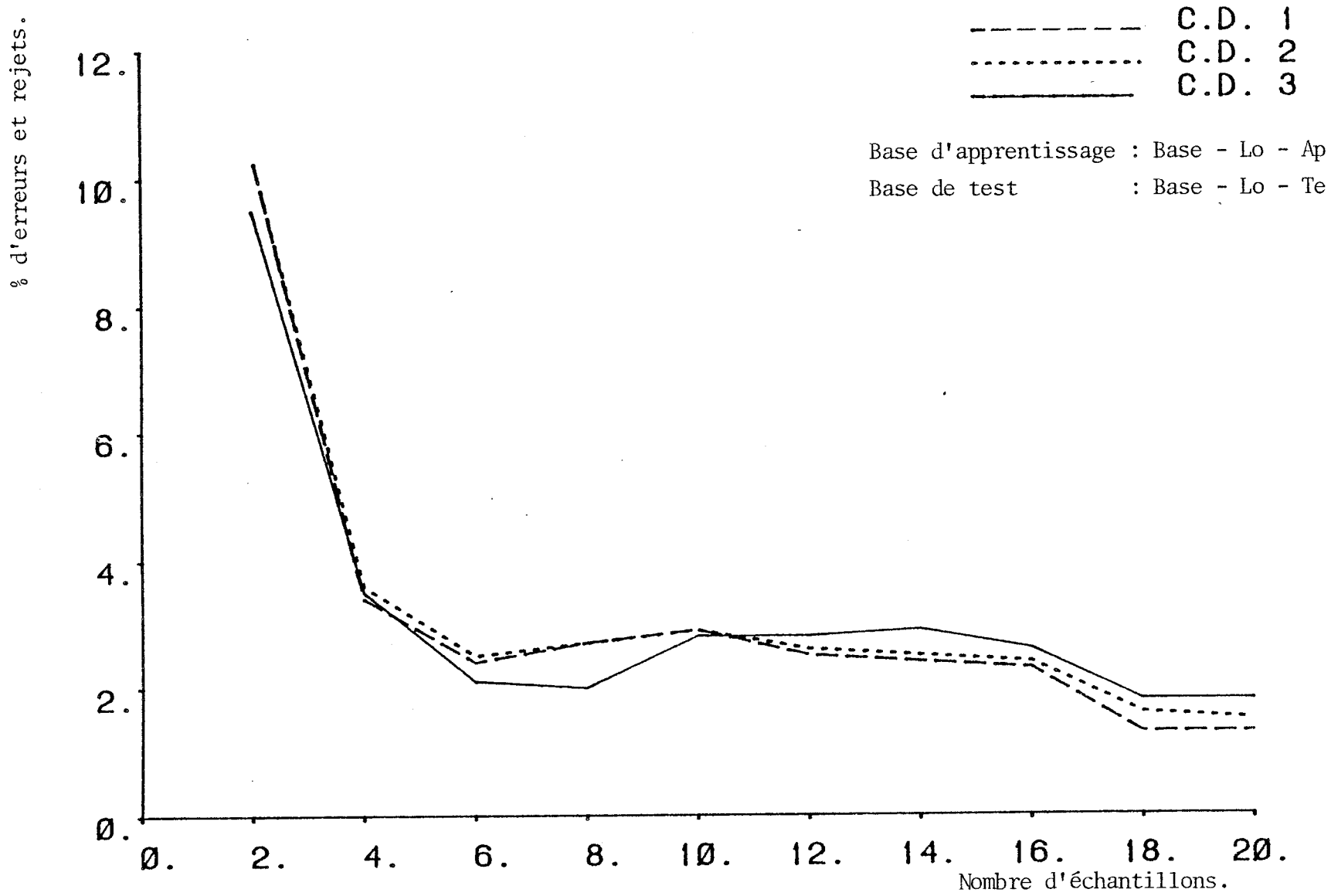


FIGURE 3.8. Taux d'erreurs et rejets en fontion du nombre d'échantillons sur la base de données de test. — Sans l'apprentissage en ligne.

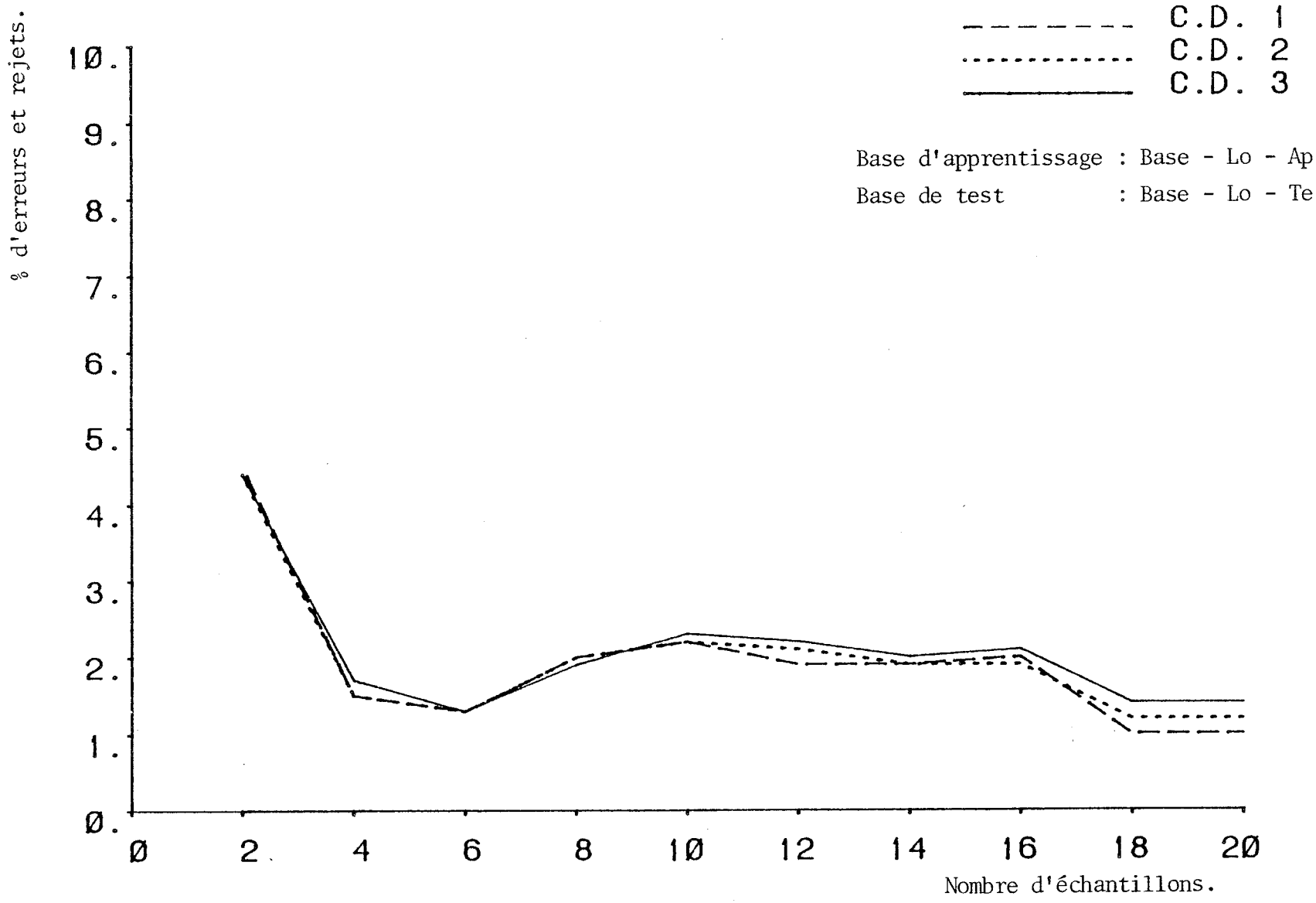


FIGURE 3.9. Taux d'erreurs et rejets en fonction du nombre d'échantillons sur la base de données de test. —— AVEC l'apprentissage en ligne.

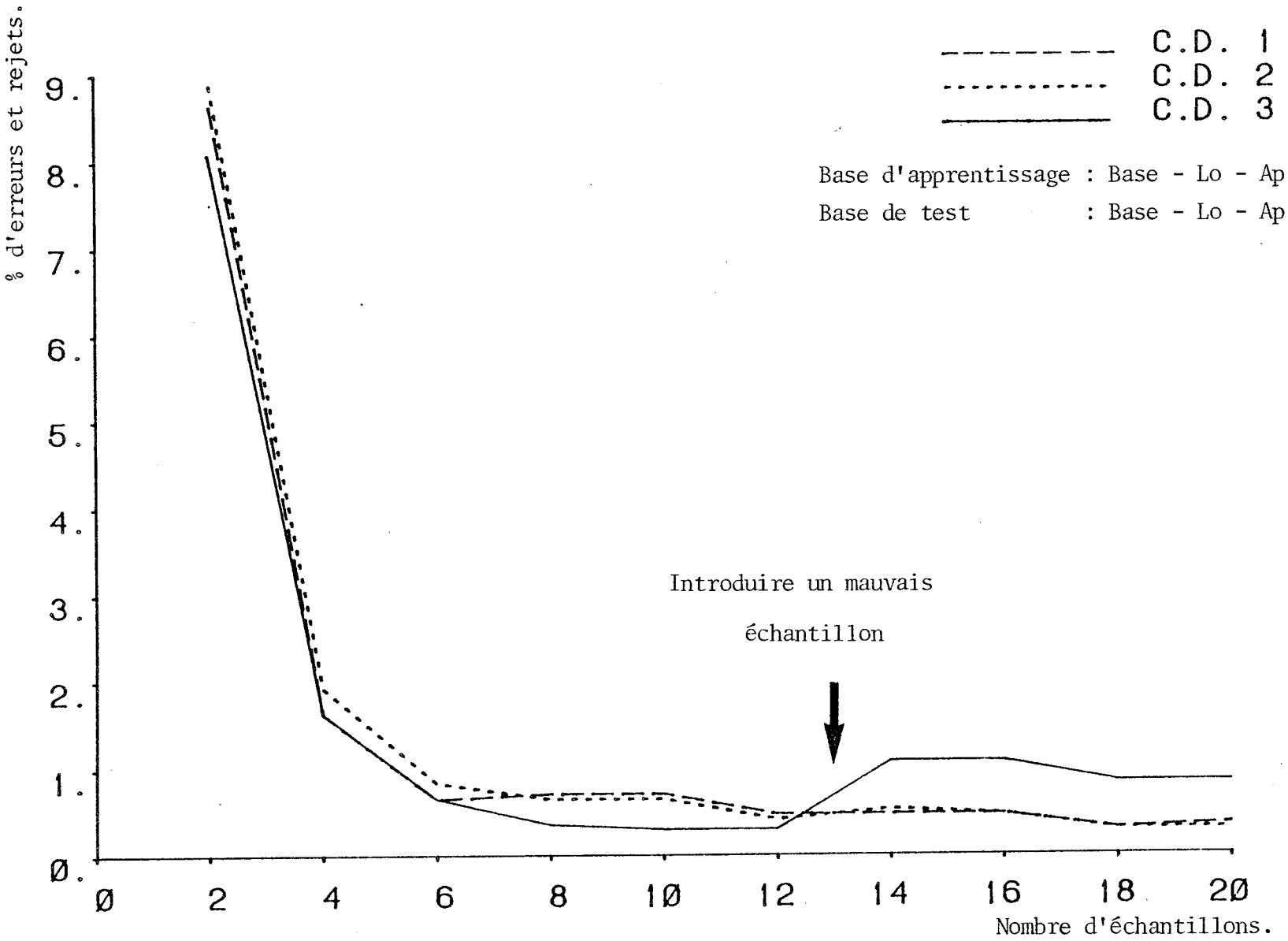


FIGURE 3.10. Taux d'erreurs et rejets en fonction du nombre d'échantillons sur la base de données d'apprentissage. —— L'effet d'un mauvais échantillon.

. Le taux de reconnaissance est de 99 % sur la base d'apprentissage et 98 % sur la base d'essai. Ces résultats semblent comparables à ceux cités dans le tableau IV de Suen et Al [34]. Mais dans l'absence d'une base de données commune, il n'est pas possible de faire une comparaison précise.

. Les performances des classifieurs C.D.1 et C.D.2 sont comparables. C.D.2. est préférable car le calcul est beaucoup plus simple que C.D.1.

3.6. EXTENSION POSSIBLE.

Nous venons de présenter dans les sections précédentes une méthode pour la reconnaissance de caractères manuscrits. Dans cette méthode, les caractères sont représentés par des polygones linéaires. Cette approche est satisfaisante pour la reconnaissance de caractères alphanumériques mais ne peut pas généraliser à une base plus large de caractères (les caractères alphanumériques plus les symboles). Dans ce dernier cas, la puissance descriptive du vecteur de caractéristiques proposé n'est pas suffisante.

Le problème de cette méthode est que l'on ne distingue pas un trait linéaire d'un trait cursif, on le représente par un ou plusieurs segments de droites. Par exemple, la représentation du caractère 0 sera exactement la même que le symbole ◇ . Donc, il est nécessaire d'identifier la nature de chaque segment de trait si on veut rendre le système plus versatile. D'autre part, comme le nombre de segments est utilisé pour effectuer le premier classement, il faut donc stocker toutes les segmentations possibles. Ce n'est pas pratique dans le cas multiscripteur puisque le nombre de segmentations possibles peut être très important pour certains caractères.

Pour résoudre ce problème, nous allons incorporer une description globale du caractère. Une description initiale peut être obtenue très facilement à partir de caractère segmenté en utilisant le signe de l'angle entre deux segments voisins. Un trait est segmenté à l'endroit où il y a un changement de signe. Un exemple de caractère après la segmentation globale est illustré dans la figure 3.11.

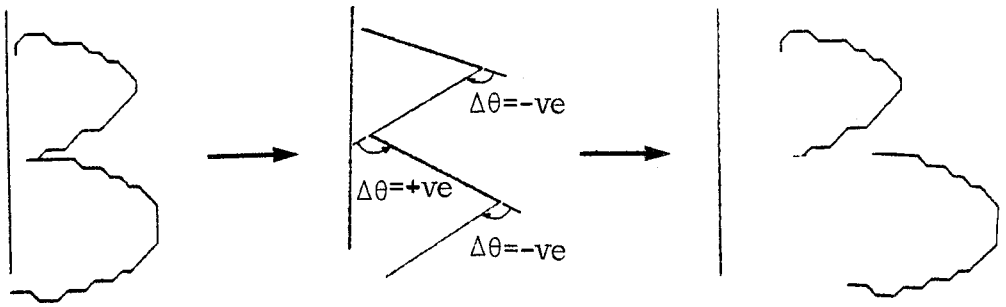


FIGURE 3.11. La segmentation globale de caractère B → une droite suivie par deux courbes positives (dans le sens des aiguilles d'une montre).

3.6.1. Définition des paramètres pour l'identification de droites et courbes.

Il est nécessaire d'identifier la nature de chaque segment pour minimiser l'erreur sur la description globale. Pour identifier la nature d'un segment, nous caractérisons chaque segment par les trois paramètres suivants (cf. Figure 3.12) :

i) - $\Delta\varepsilon^2 = (\varepsilon_1^2 - \varepsilon_2^2)/N$

ε_1^2 (ε_2^2) est l'erreur d'approximation au sens des moindres carrés par une courbe parabolique (une droite). N est le nombre de points du segment.

ii) - ΔS = la surface entre la droite d'approximation et le segment original divisé par N.

iii) - $\Delta\theta = (\theta_1 - \theta_2)/N$

θ_1 (θ_2) est l'angle de la tangente au début (à la fin) du segment.

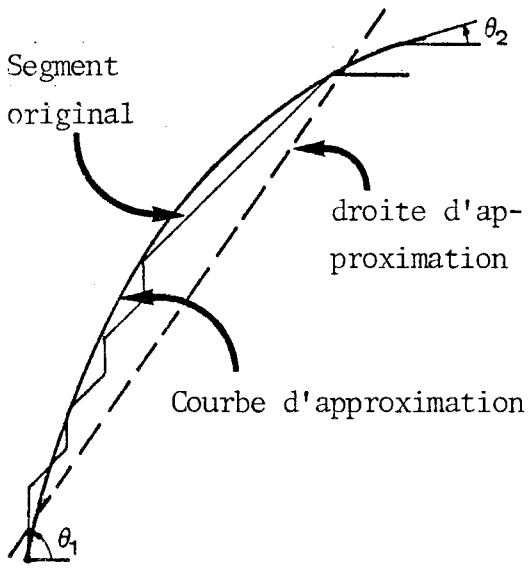


FIGURE 3.12.
Les approximations d'un segment.

3.6.2. Détermination des seuils pour la séparation de droites et courbes.

Pour déterminer les seuils approprés, nous avons stocké 20 échantillons pour chacun de quatre caractères choisis : A, B, G et R. Les caractères sont segmentés et les trois paramètres qui représentent chaque segment sont calculés. Il y a au total 160 segments de droites et 200 segments de courbes.

Nous avons voulu savoir parmi ces trois paramètres, qu'elle est l'information utile pour la classification de courbes et de droites que porte chacun de ces paramètres. La méthode utilisée ici est l'analyse en composantes principales. Les sorties graphiques de cette analyse nous ont montré que le paramètre $\Delta\varepsilon^2$ a moins de signification par rapport aux deux autres, ce paramètre est donc abandonné. Il nous est donc resté deux paramètres. Pour mieux voir la distribution des données, nous les avons tracées sur le plan $\Delta\theta$ - ΔS en échelle logarithme (cf. Figure 3.13). Après un certain nombre d'itérations, nous avons trouvé la surface de séparation suivant :

Si $\Delta\theta < 2.5$

et $\Delta S < .13$, alors le segment est une droite

sinon c'est une courbe.

Nous avons une erreur de l'ordre de 2 % ou plus précisément 1,8 % d'erreur qui provient d'une mauvaise décision concernant les courbes et 0,1 % qui provient des droites prises comme des courbes. La plupart des erreurs proviennent du fait que les segments définis à priori comme des courbes sont trop plats même à l'oeil.

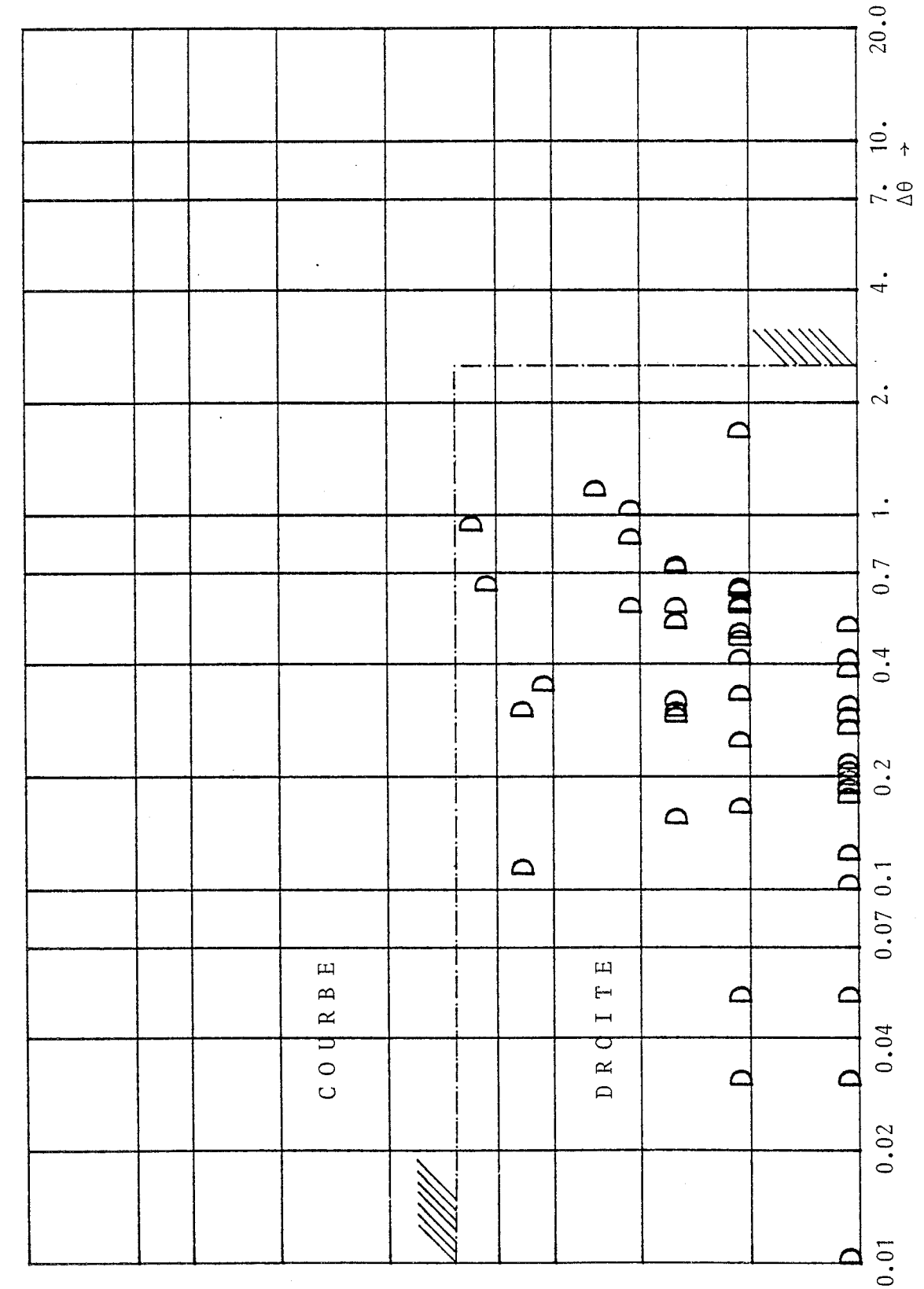


FIGURE 3.13.a. La distribution de données sur le plan ΔS - Δθ. Caractère A. (D - Segment de droite ; C - Segment de courbe).

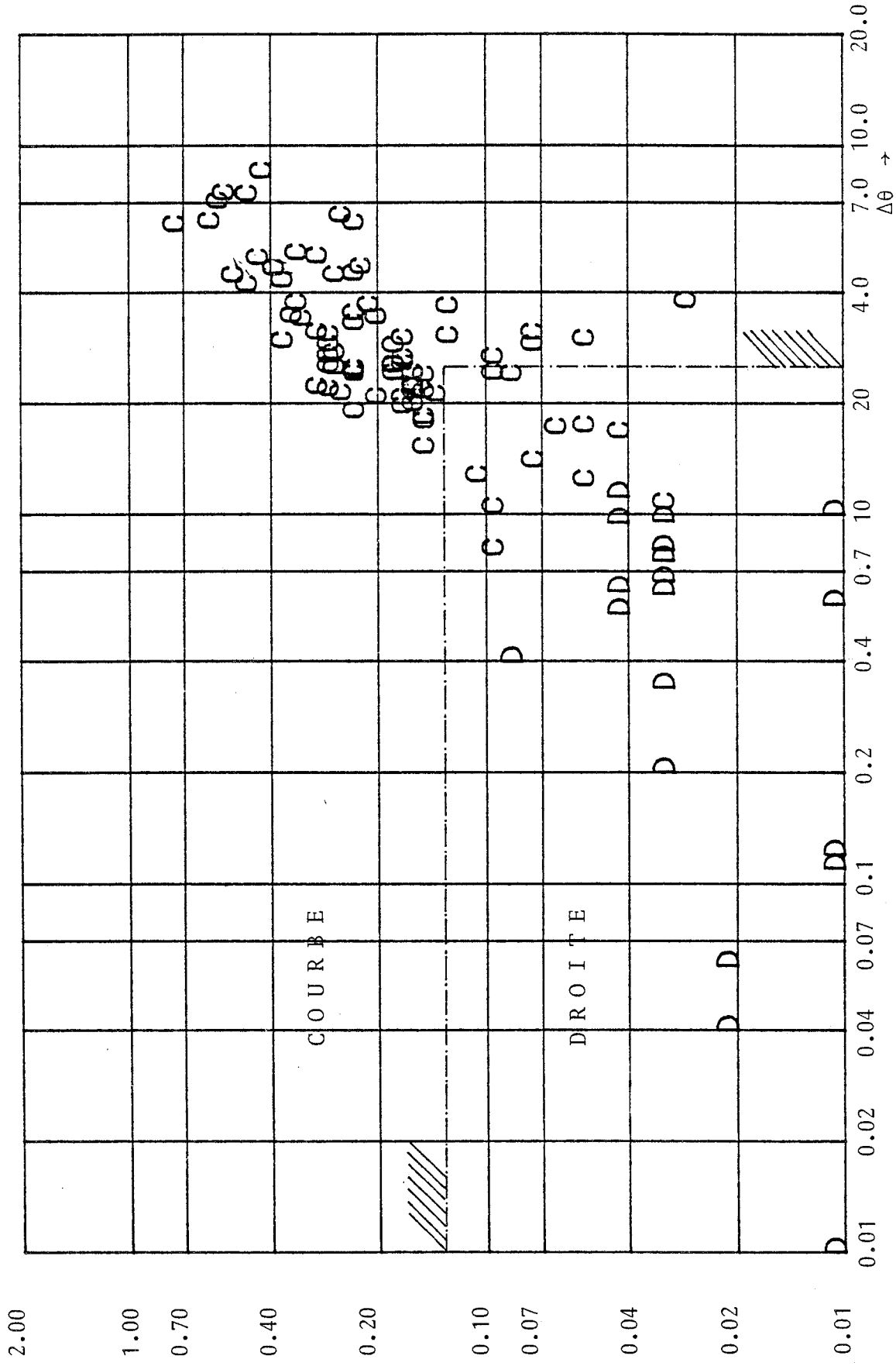


FIGURE 3.13.b. La distribution de données sur le plan ΔS - Δθ. Caractère B. (D - Segment de droite ; C - Segment de courbe).

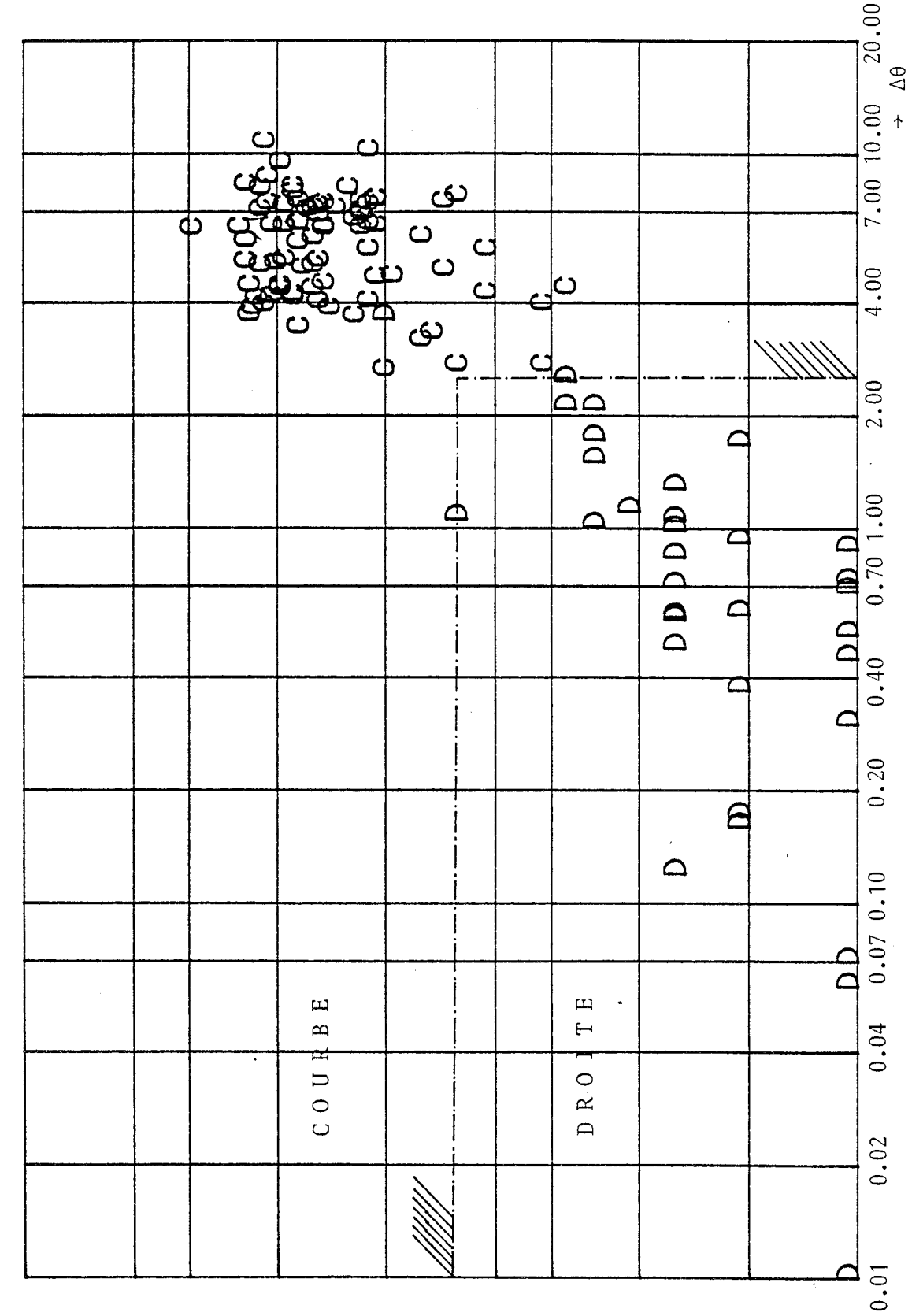


FIGURE 3.13.c. La distribution de données sur le plan ΔS - Δθ. Caractère G. (D - Segment de droite ; C - Segment de courbe).

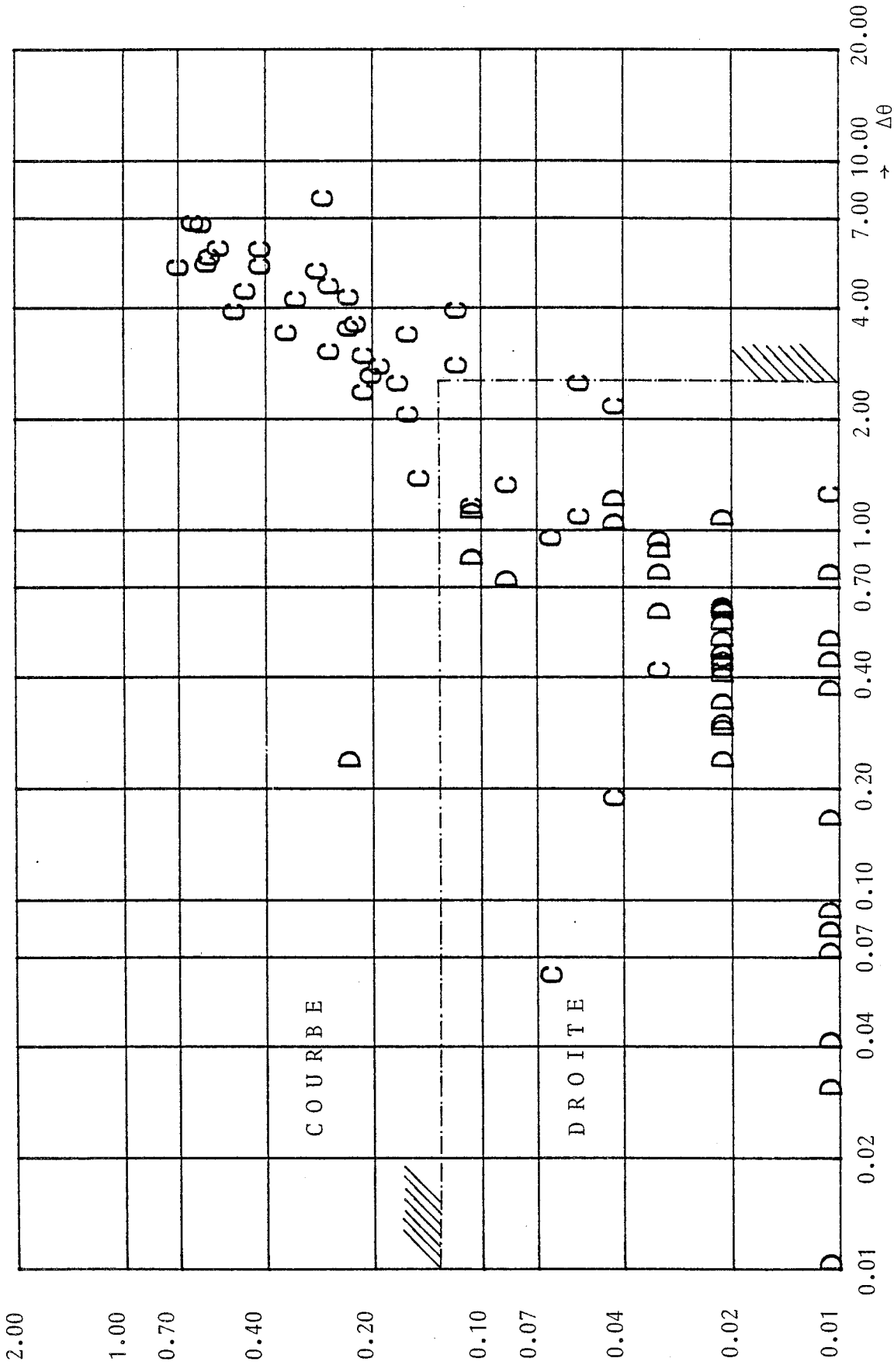


FIGURE 3.13.d. La distribution de données sur le plan ΔS - Δθ. Caractère R. (D - Segment de droite ; C - Segment de courbe).

3.6.3. La description globale de caractère.

Une fois que chaque segment d'un caractère est identifié, nous obtiendrons la description globale finale du caractère. Cette description peut être considérée comme une information supplémentaire. Au lieu de faire un classement par le nombre de segments dans la première étape du classifieur, nous classons le caractère entré par ses caractéristiques globales. Un classement typique est illustré dans le tableau 3.1. Il y a deux avantages en incluant cette nouvelle information :

i) - Un classement plus fin dans le classifieur qui permet éventuellement la classification des autres caractères ou symboles.

ii) - Les erreurs de segmentation linaire (par des droites) peut être corrigées avec cette description globale.

TABLEAU 3.1.

CLASSIFICATION PAR LES CARACTERISTIQUES GLOBALES.

Classe	Catégories	Classe	Catégories
DD	L, T, V, X	D+ +	B
DDD	A, F, H, I, K, N,	-	C, 0, 6
	Y, Z, 1, 4, 7	- +	S, 8
DDDD	E, W, M	- D	Q, 9, Ø
D+	D, J, P	- DD	G
D+D	R, 5	+ D	2
D-D	U	+ +	3

3.7. CONCLUSION.

Nous venons de présenter dans les sections précédentes une méthode de reconnaissance de caractères alphanumériques manuscrits dans un environnement monoscripteur et multiscripteur avec contraintes. Pour extraire les caractéristiques du caractère entrée, nous proposons une méthode de segmentation qui transforme le caractère entrée en polygone linéaire. Contrairement aux autres méthodes de segmentations, cette méthode segmente un caractère selon le nombre de points tournants sur le contour du caractère et non pas par rapport à un critère défini à priori. Par conséquence, nous obtenons, après l'étape de segmentation, la représentation directement utilisable par le classifieur. Mais en raison du type de classifieur utilisé dans cette méthode, le résultat de segmentation est très important. Une erreur de segmentation entraînera une mauvaise décision ou un cas de rejet. Pour limiter ce genre d'erreur, nous avons mis beaucoup d'attention sur l'étape de prétraitement et l'étape de segmentation. Le choix du type de paramètres est aussi une étape importante dans notre système. Pour cela, nous avons étudié plusieurs types de paramètres et nous avons choisi le type de paramètres qui convient le mieux. Un bon choix limitera l'erreur de classification.

Le programme complet a environ 500 instructions FORTRAN, avec les données du classifieur, il occupe à peu près 20 K octects de l'espace mémoire. La vitesse d'exécution sur le NORD 10/S est de l'ordre de 100 ms par caractère.

° ° °

CHAPITRE IV.

IV. PROPOSITION N°2

4.1. INTRODUCTION

La méthode que nous avons présentée dans le chapitre précédent a été dévelopée dans le cadre de la reconnaissance de caractères manuscrits monoscripteur et multiscripteur AVEC contraintes. Cet approche donne des résultats assez satisfaisants dans cet environnement, mais on ne peut pas le généraliser pour le cas du multiscripteur SANS contraintes. Dans ce dernier cas, la méthode de segmentation nous donne souvent trop de variations sur le nombre et la séquence de segments et les problèmes suivants se manifestent :

. Au niveau de l'apprentissage, il faut grouper les échantillons de même type (c'est-à-dire, les échantillons qui ont la même séquence et le même nombre de segments) pour que l'on puisse calculer les moyennes et la matrice de covariances de chaque groupe de façon indépendante. Ce genre de groupement est toujours possible mais on risque d'avoir un classement peu efficace, surtout dans le cas d'apprentissage récursif.

. Au niveau classifieur, le nombre de catégorie risque d'être trop important. Ceci amenera donc à une taille importante de mémoires, un temps de calcul plus long et un taux de substitution plus élevé.

. Il faut une base de données très grande pour définir les moyennes et les matrices de covariances.

En effet une étude préliminaire utilisant l'analyse en composantes principales nous a montré que le risque de recouvement est très probable [43]. Pour résoudre tous ces problèmes, nous allons entreprendre une nouvelle méthode.

Dans le cas multiscripteur SANS contraintes, on peut dire qu'il n'existe pas de caractéristiques invariantes. La seule caractéristique qui varie peu par rapport aux autres est la nature (ou le type) de chaque trait qui constitue un caractère ; qu'un trait s'agisse d'une droite, d'une courbe à courbure positive ou négative, d'une courbe fermée, d'une droite suivie par un coin et une courbe, etc... Nous appelons cette description qualitative les Caractéristiques Globales du caractère. Ce genre de description, qui est très proche de celui utilisé par l'homme, a été employé par des nombreux chercheurs dans ce domaine. Nous allons citer quelques méthodes proposées afin de les comparer avec la notre.

Berthod et al [5] et Belaïd [44] utilisent la même méthode de segmentation : la segmentation angulaire. Après la segmentation, le caractère entré est représenté par des segments de droite. L'angle entre chaque paire de segments voisins et la longeur de chaque segment sont traités comme des primitives. Ils extraient ensuite de ces primitives, à l'aide d'un automate fini, la description globale du caractère.

Ali et al [2] emploient une méthode de segmentation plus compliquée. Un caractère entré est transformé en polygone en utilisant un algorithme de segmentation de type "Split and Merge" [26], le caractère segmenté est traité ensuite par un 'Parseur syntaxique' où la sortie est la description globale du caractère considéré.

On remarque que les démarches sont similaires. On commence par une segmentation de caractère sur le plan x-y pour fabriquer les primitives et ou extrait ensuite de ces primitives la description globale.

Etant donné que nos caractères sont codés par des chaines de Freeman, nous abordons le problème de façon différente. En fait,

si on trace les codes de Freeman traités en fonction de la longeur (c'est-à-dire, sur le plan Θ - S), on distinguera assez facilement les traits droites des traits cursifs car seuls les précédents deviennent des droites horizontales sur ce plan. D'ailleurs, on peut y distinguer les deux types de courbes, les courbes de courbure positives et négatives. Pour décrire les courbes Θ - S d'une manière explicite,nous les segmentons en une série de segments de droites. En fonction de sa longeur et sa pente, chaque segment sera classé comme étant une primitive de droite, de coin, de courbe à courbure positive ou négative. La description du caractère par des primitives est une description préliminaire. Elle doit être simplifiée et unifiée pour réduire le nombre de variances. Ceci est faite par l'intermédiaire d'un automate fini.

Nous allons voir que seule la description globale ne suffit pas pour distinguer un caractère d'un autre. Il faut en plus attacher à chaque caractéristique globale des attributs appropriés. Après avoir essayé plusieurs types d'attributs, nous avons choisi les attributs de "positions" qui semblent être les plus répandus sur ce problème.

Notre démarche est la suivante : Après le prétraitement du caractère entré, nous enlevons les caractéristiques globales directement du plan Θ - S , un automate accepte et transforme ces caractéristiques à une forme finale. Nous complétons la description du caractère en attachant à chaque caractéristique les attributs de position. Au niveau classifieur, le caractère entré sera classé si on trouve la même description globale dans le dictionnaire et la distance entre les attributs correspondants est plus petite qu'un certain seuil.

Ce chapitre est organisé de la façon suivante. Dans la partie 4.2 et 4.3, nous traiterons la méthode pour la présentation de caractère en plan Θ - S, les algorithmes de segmentation et la définition des primitives. L'automate qui conduit les primitives à la

description globale et la définition des attributs de position seront élaborées dans la partie 4.4. Dans la partie 4.5 nous présenterons les algorithmes d'apprentissage et le classifieur. Nous conclurons ensuite ce chapitre par la présentation des résultats obtenus (partie 4.6) et les discussions de la méthode proposée (partie 4.7).

4. 2. REPRESENTATION D'UN CARACTERE EN PLAN Θ - S

Nous rappelons que dans notre système chaque caractère entrée est codé par une ou plusieurs chaînes de Freeman, chacune représente le tracé d'un trait. Si on prend la tangente (Θ) de chaque point d'un trait et que l'on trace Θ en fonction de la distance le long du trait (S), les segments de droites du trait deviennent des lignes droites horizontales et les segments d'arcs deviennent des lignes penchées dont les pentes mesurent les courbures des arcs. Nous pouvons actuellement estimer Θ à partir des codes de Freeman directement mais nous préférons appliquer la méthode approximative de Mckee et Aggarwal [20] qui nécessite moins de calculs et qui évite surtout les calculs en mode réel.

La méthode contient les trois opérations suivantes :

. Compensation,

. Extension,

. Lissage.

Un code de Freeman après avoir subi ces trois opérations s'appelle : Un code de Freeman lissé étendu compensé (Smoothed Compensated Extended Freeman Code : SCEFC). Nous allons présenter très briévement ces trois opérations dans les sections suivantes.

4.2.1. Code de Freeman Compensé (Compensated Freeman Code CFC)

La longeur de code de Freeman n'est pas unique. La longeur d'un code de valeur impaire est $\sqrt{2}$ fois plus longue qu'un code de valeur paire (voir figure 2.1). Il faut donc tenir compte de cette différence de longeur quand on trace les codes de Freeman en fonction de S. La solution moyenne est de prolonger la chaîne de Freeman. C'est-à-dire, qu'on double les codes de valeur paire et qu'on triple les codes de valeur impaire pour compenser la différence de longeur. Par exemple, si la chaîne de Freeman originale est : 223455, la chaîne de Freeman compensée sera 222233344555555.

Cette opération présente un inconvénient : les chaînes deviennent en générale 2,5 fois plus longues et en conséquence on aura 2,5 fois plus de données à traiter. D'autre part, il y a une erreur de l'ordre de 6 % sur la longeur de code impaire.

4.2.2. Code de Freeman Compensé Etendu (Compensated Extended Freeman Code CEFC)

Les codes de Freeman présentent des discontinuités qu'il faut supprimer. Par exemple, quand les valeurs de codes changent de 0 à 7, le changement réel est - 1 et non -7. Pour s'assurer de la continuité, nous modifions les codes par la règle suivante :

Soit $e_i = f_i$

e_i = Code de Freeman Compensé Etendu

f_i = Code de Freeman Compensé

Pour $2 \leqslant i \leqslant N$, on calcule E_i telle que

$$| e_{i-i} - (f_i + 8\, E_i) | \leqslant 4$$

et on calcule e_i pour $i > 1$ par

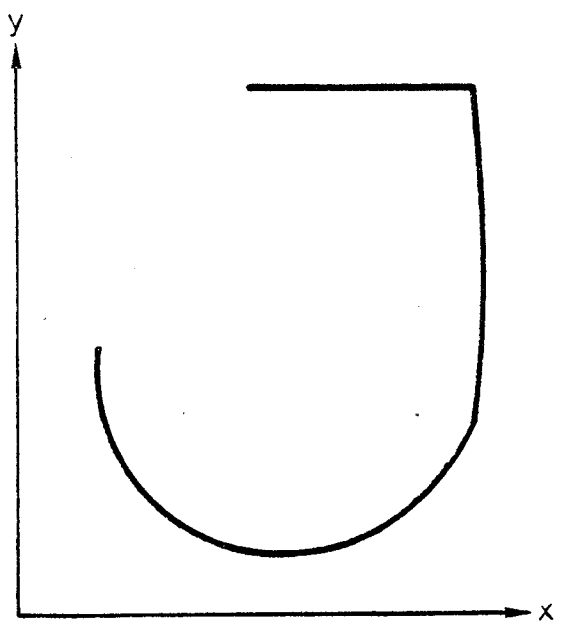
$$e_i = f_i + 8\ E_i$$

4.2.3. Code de Freeman Lissé Compensé Etendu (Smoothed Compensated Extended Freeman Code SCEFC)

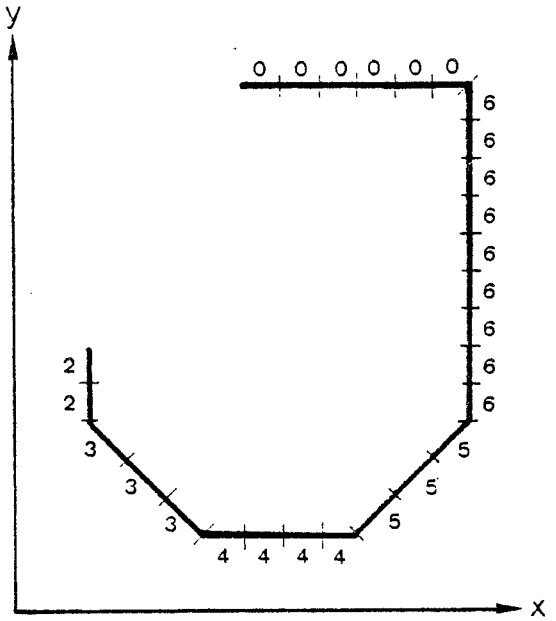
Un CEFC ne prend que 8 valeurs de 0 à 7, il ne représente donc pas une bonne estimation de Θ. Un moyen simple pour l'estimer est de calculer les sommes mobiles, c'est-à-dire, on remplace chaque CEFC e_i par $\sum_{j=i-k}^{i+k} e_j$. k est une constante. Cette opération est en effet une sorte de lissage. Pour avoir un bon lissage, il faut bien choisir la valeur de k qui dépend du niveau de bruits et de la dimension des caractères. Par exemple, nous prenons k = 4 si la dimension du caractère est environ 32 x 32 unités.

Un exemple de traitements est décrit dans la figure 4.1. Dans les figures 4.1.a et 4.1.b, nous montrons un trait original et le trait digitalisé représenté par une chaîne de Freeman sur le plan x - y. Les codes de Freeman après la première opération sont tracés dans la figure 4.1.c, notons qu'il existe une discontinuité. Dans la figure 4.1.d, cette discontinuité est supprimée avec l'opération d'extension. La représentation finale (figure 4.1.e) est obtenue avec l'ordre de lissage K = 4. Dans cette figure, nous avons une ligne horizontale suivie par une droite penchée. Cette courbe signifie que nous avons un segment de droite suivi par un coin, un autre segment de droite et une courbe de courbure positive sur le plan x - y. Ce qui correspond exactement à ce que nous avons au départ (figure 4.1.a).

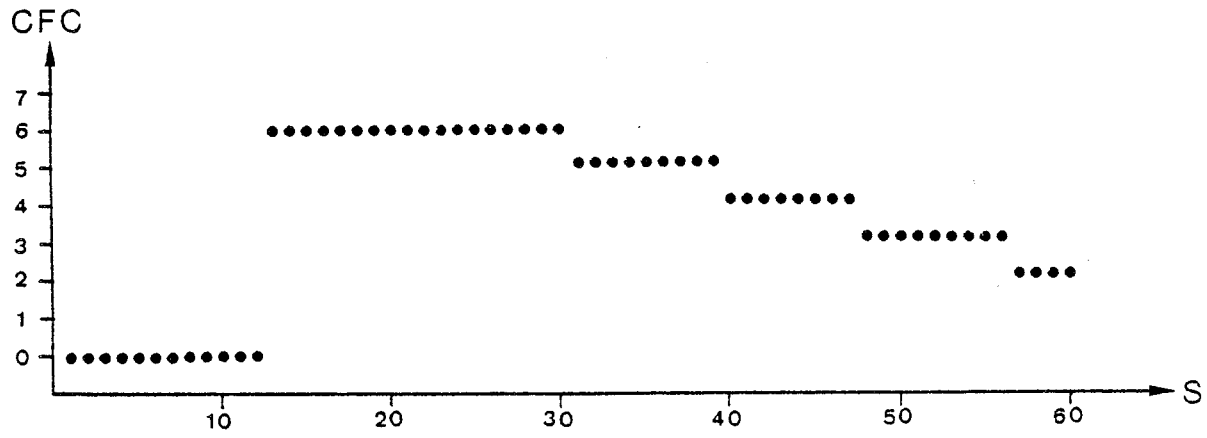
A part la simplicité des opérations et les calculs en mode entière, cette représentation donne aussi une image claire de la nature de trait qui facilite plus tard l'étape d'extraction des caractéristiques.


y
x

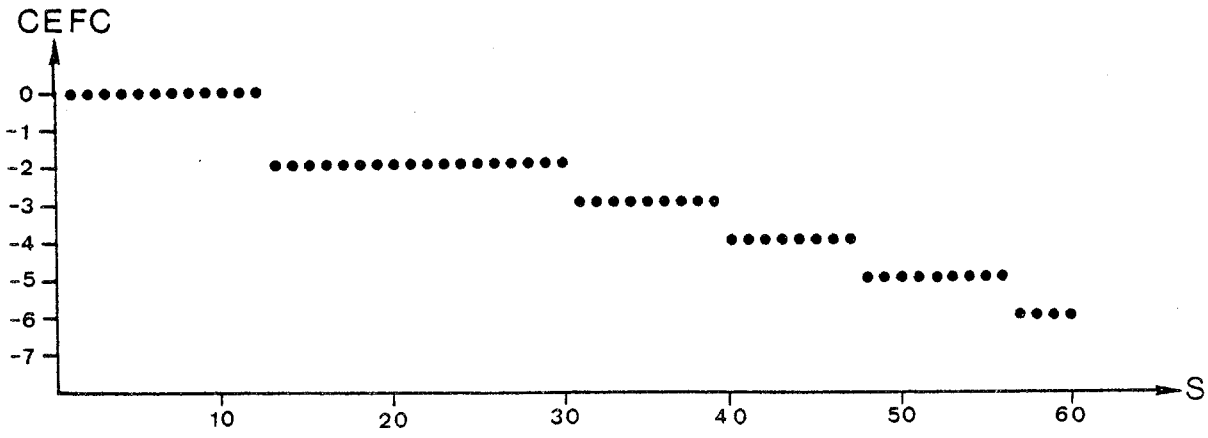
(a)


y
x
0 0 0 0 0 0
6 6 6 6 6 6 6 6 6 6
2 2
3 3 3
4 4 4 4
5 5 5 5

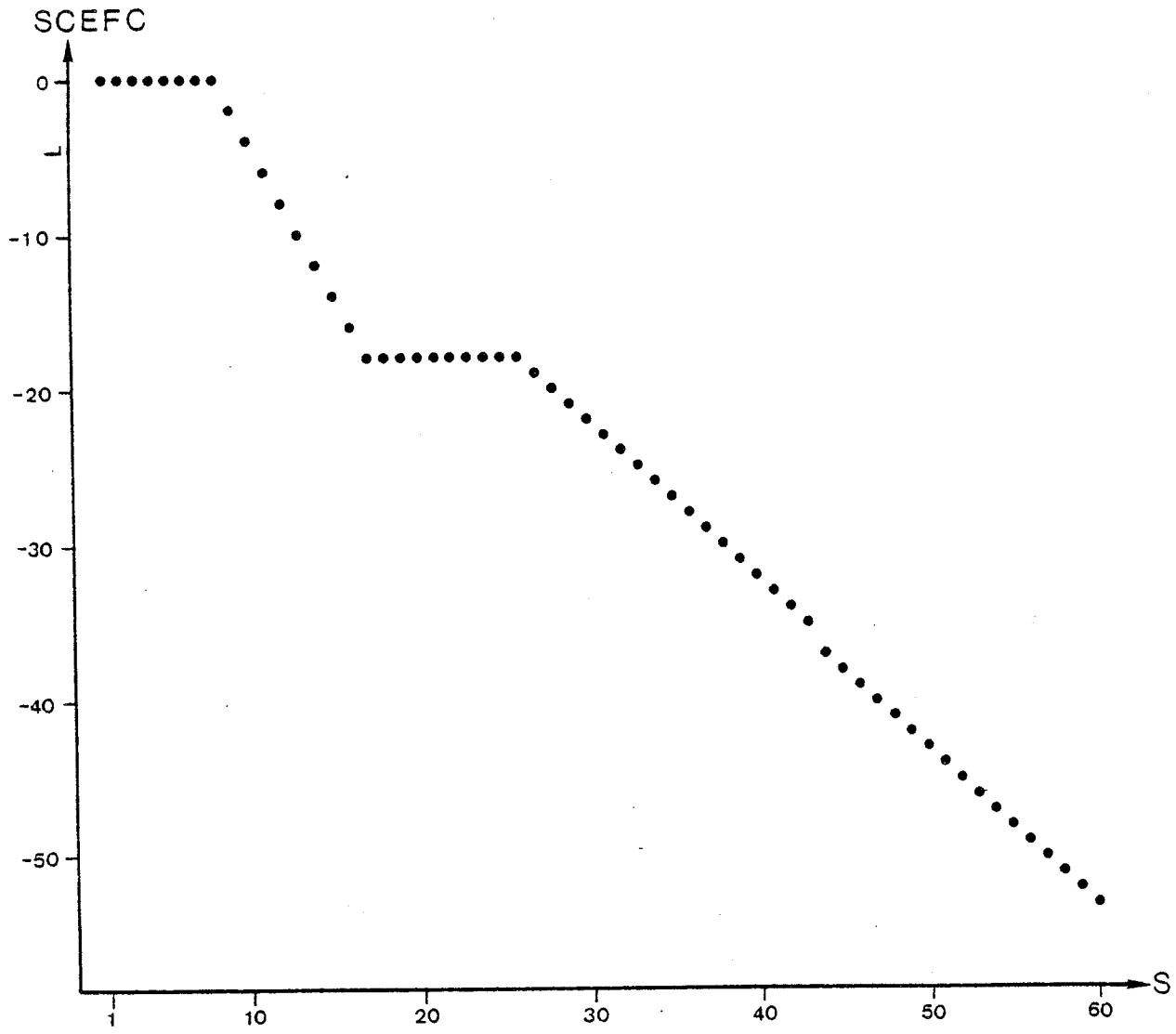
(b)


CFC
7
6
5
4
3
2
1
0
10
20
30
40
50
60
S

(c)


CEFC
0
-1
-2
-3
-4
-5
-6
-7
10
20
30
40
50
60
S

(d)



(e)

FIGURE 4.1.(a) Le trait original, (b) le trait digitalisé, (c) Code de Freeman compensé, (d) Code de Freeman compensé étendu, (e) Code de Freeman lissé compensé étendu.

4.3. DECOMPOSITION EN PRIMITIVES

Nous avons vu que les courbes Θ - S expriment d'une manière assez évidente les caractéristiques du caractère écrit. Mais, pour pouvoir décrire le caractère utilisant ces courbes, il faut d'abord segmenter ces courbes en segments de droite et classifier ensuite chaque segment en primitives. Nos problèmes seront donc les suivants.

- Trouver un algorithme de segmentation qui s'adapte le mieux à nos données.

- Trouver les seuils appropriés pour classer les segments en primitives.

4.3.1. Segmentation de courbes Θ - S

Le problème de segmentation pour le cas de caractères alphanumériques a été discuté dans la partie 3.2. Dans cette partie, nous allons étudier la segmentation d'une courbe en général. Plusieurs méthodes ont été proposées dans la littérature, notamment l'algorithme "Split and merge" de Pavlidis et al [26], l'algorithme itérative de Ramer [28] et l'algorithme "rapide" de Slansky [32]. Pour en choisir un approprié, nous avons essayé ces trois algorithmes sur nos données et les remarques suivantes ont été notées.

- L'algorithme de Pavlidis donne une segmentation très satisfaisante mais le temps de calcul est trop long. Il peut être réduit par une bonne initialisation de "break points". Mais nous n'avons pas trouvé un algorithme simple capable de nous donner cette bonne initialisation. Une possibilité est donc d'utiliser l'algorithme de Sklansky comme l'algorithme d'initialisation. Mais dans ce cas, nous aurons alors un algorithme de segmentation trop complexe pour notre application.

. L'algorithme de Ramer est un algorithme dont le principe est simple mais pas très efficace. Du fait de la nature itéractive de cet algorithme, on fait souvent plusieurs passages sur les mêmes données avant d'arriver à la segmentation finale. En présence de bruits, la segmentation que l'on obtient n'est pas toujours optimale. Mais il est relativement facile de choisir un seuil convenable pour obtenir une segmentation assez bonne.

. L'algorithme de Sklansky est plus récent. L'avantage de cet algorithme par rapport aux algorithmes précédents est que l'on ne fait qu'une fois le passage des données, il est donc beaucoup plus rapide. Mais sur nos données, à cause de la variation du niveau de bruit, il est difficile de choisir un seuil approprié.

Nous avons retenu l'algorithme de Ramer, nous l'avons trouvé plus approprié pour notre application que les deux autres. Les raisons sont les suivantes :

. Un seuil convenable peut être choisi assez facilement et il peut être formulé comme une fonction de la dimension.

. Le temps de calcul est acceptable bien qu'il soit plus long que l'algorithme de Sklansky.

. Les segmentations obtenues sont assez bonnes. On peut bien sûr espérer des meilleurs résultats avec l'algorithme de Pavlidis mais on augmentera alors la complexité. Ces légères améliorations à notre avis ne justifient pas l'emploi.

Cet algorithme cherche essentiellement à produire d'une

manière itéractive les polygones avec un nombre petit - mais pas minimum - de vertex sur la courbe à segmenter. La distance de ce courbe de polygone est utilisée comme un critère (D) à satisfaire. Ce critère dépend de l'ordre de lissage K (qui dépend principalement de la dimension du critère). Pour K = 4, nous trouvons qu'avec D fixé à 5 (unité en SCEFC), les segmentations que l'on obtient sont assez satisfaisantes. Nous montrons dans la figure 4.2. un exemple de segmentation. Dans cet exemple, le caractère B à 2 traits, nous appliquons consécutivement pour chaque trait représenté sur le plan Θ - S l'algorithme de segmentation de Ramer et nous obtenons 9 segments de droites avec les valeurs de D et K fixées à 5 et 4 respectivement.

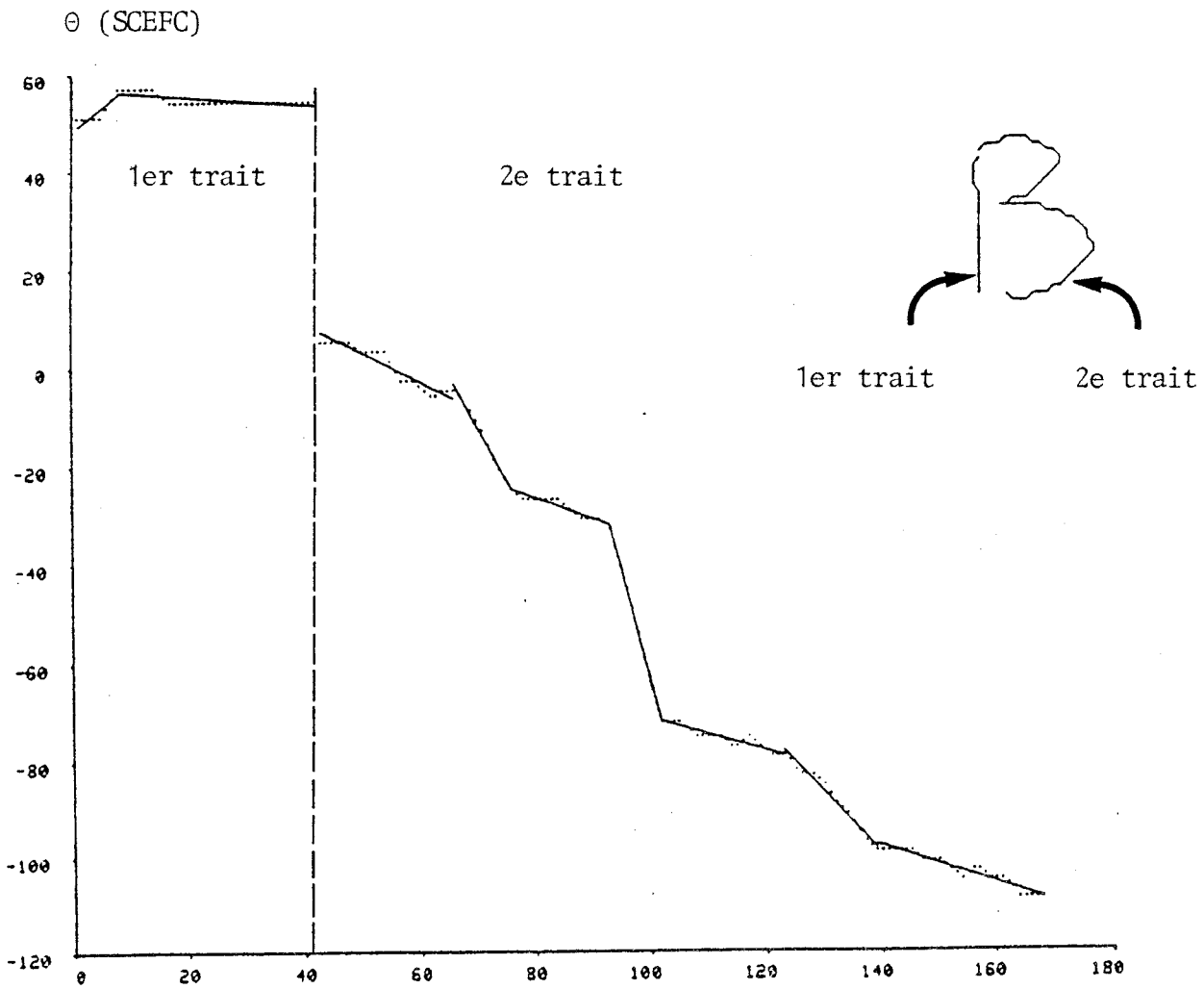


FIGURE 4.2. Segmentation des courbes Θ - S

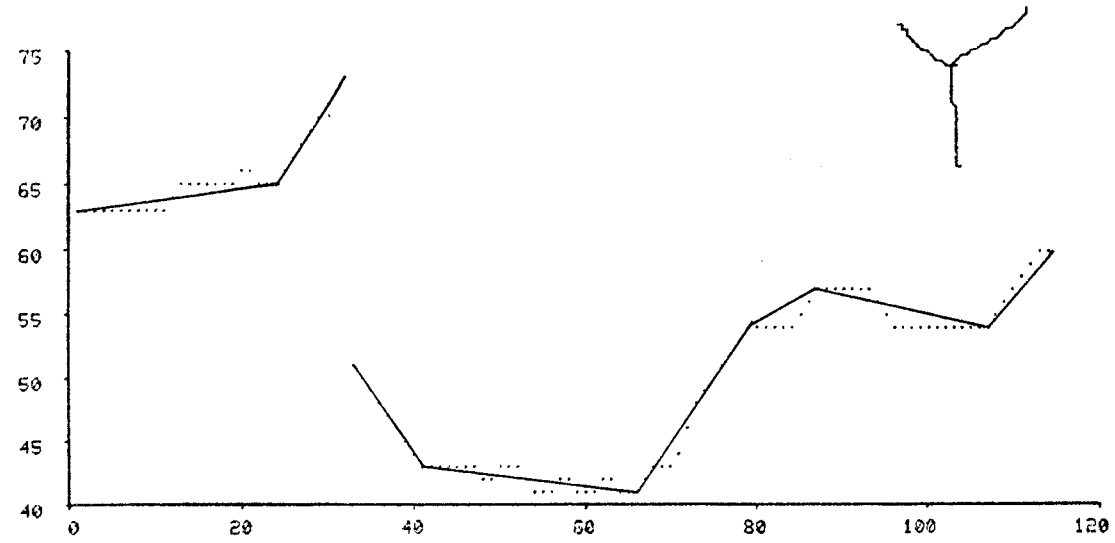
En général, nous obtenons plus de segments que nous le souhaitons, avec cet algorithme de segmentation. Nous pouvons diminuer le nombre de segments en utilisant une valeur plus grande de D, mais nous risquons alors de perdre des détails. La valeur de D a été choisie itéractivement avec une base de données qui contient tous les 36 caractères alphanumériques. Nous faisons varier la valeur de D de 3 à 7 et nous examinons les segmentations obtenues pour chaque valeur de D. Nous trouvons qu'à partir de 6, nous commençons à perdre des détails importants. Dans le cas contraire, si D est trop petit, nous obtiendrons plus de segments que nécessaire. Dans ces deux cas extrèmes il serait très difficile d'en extraire les caractéristiques globales.

En fait, la valeur de D dépend principalement de la valeur de K (l'ordre de lissage), et K dépend de la dimension du caractère écrit et du niveau du bruit. Nous avons en réalité fixé K comme une valeur proportionnelle à la dimension de caractère (car il n'est pas possible de savoir le niveau de bruit à priori). Pour la base de caractères que nous utilisons, la dimension de chacun est d'environ 30 x 30, donc K = 4 et nous avons trouvé que D = 5 donne un meilleur compromis.

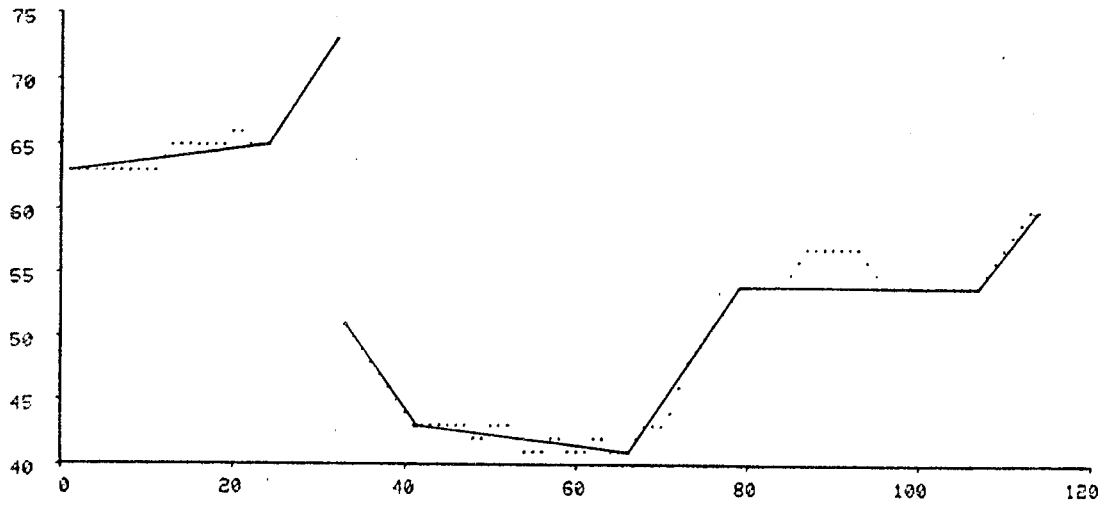
Un exemple sur l'effet de critère de Ramer D est illustré dans la figure 4.3.

4.3.2. Classification de segments en primitives

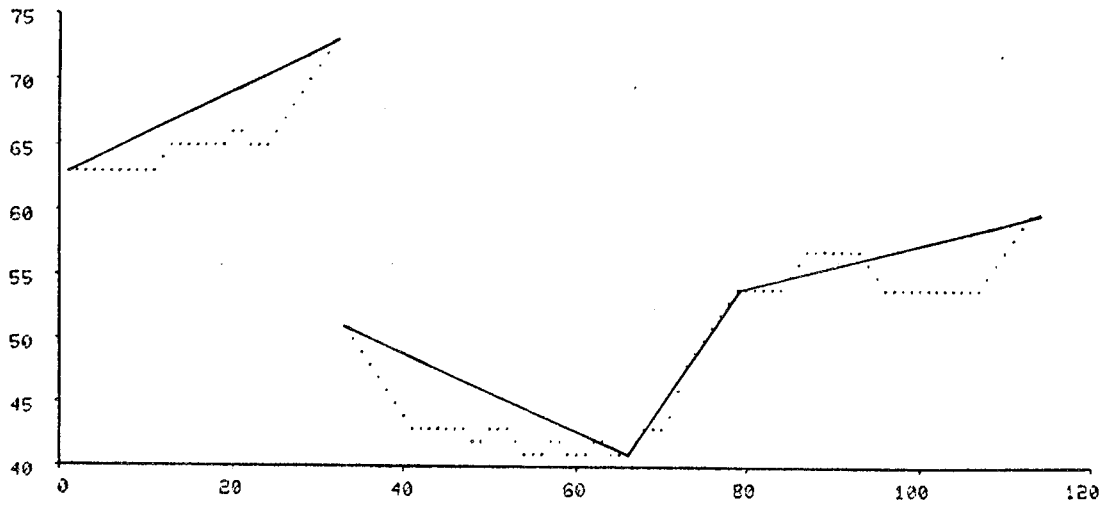
Nous voulons utiliser les résultats de segmentation, c'est-à-dire, les différents segments de droites sur le plan Θ - S, pour décrire d'une manière explicite les caractéristiques du caractère écrit. Nous disposons de 3 paramètres pour chaque segment de droite, à savoir : la pente (P), le changement de l'angle ($\Delta\Theta$) et la longeur (N).



(a)



(b)



(c)

FIGURE 4.3. Choix de critère de Ramer D
(a) D = 3, (b) D = 5, (c) D = 6.

Nous allons les utiliser pour classer chaque segment dans les 6 catégories suivantes :

i) Un segment de courbe de courbure positive c^+

ii) Un segment de courbe de courbure négative c^-

iii) Un segment de droite d

iv) Un "break" de courbure positive b^+

v) Un "break" de courbure négative b^-

vi) Un segment provient du bruit au début ou à la fin du trait n

Nous ajoutons une dernière catégorie (f) qui représente simplement une levée de plume pour délimiter le trait. Nous les appelons les primitives du caractère écrit.

Nous allons classer ces segments par une méthode de seuil. Une question se pose naturellement : comment choisir les seuils convenables ? Il faut tout d'abord bien mesurer ces trois paramètres et puis étudier ces distributions afin de choisir une meilleure surface de séparation.

4.3.2.1. Estimation des paramètres P, $\Delta\Theta$ et N

Nous rappelons que l'algorithme de segmentation de Ramer ne localise que les vertex (ou "break points"), et les segments de droites qui joignent ces vertex sont approximatifs et très sensibles aux bruits. Nous ne pouvons donc pas calculer S et $\Delta\Theta$ directement (voir figure 4.4.).

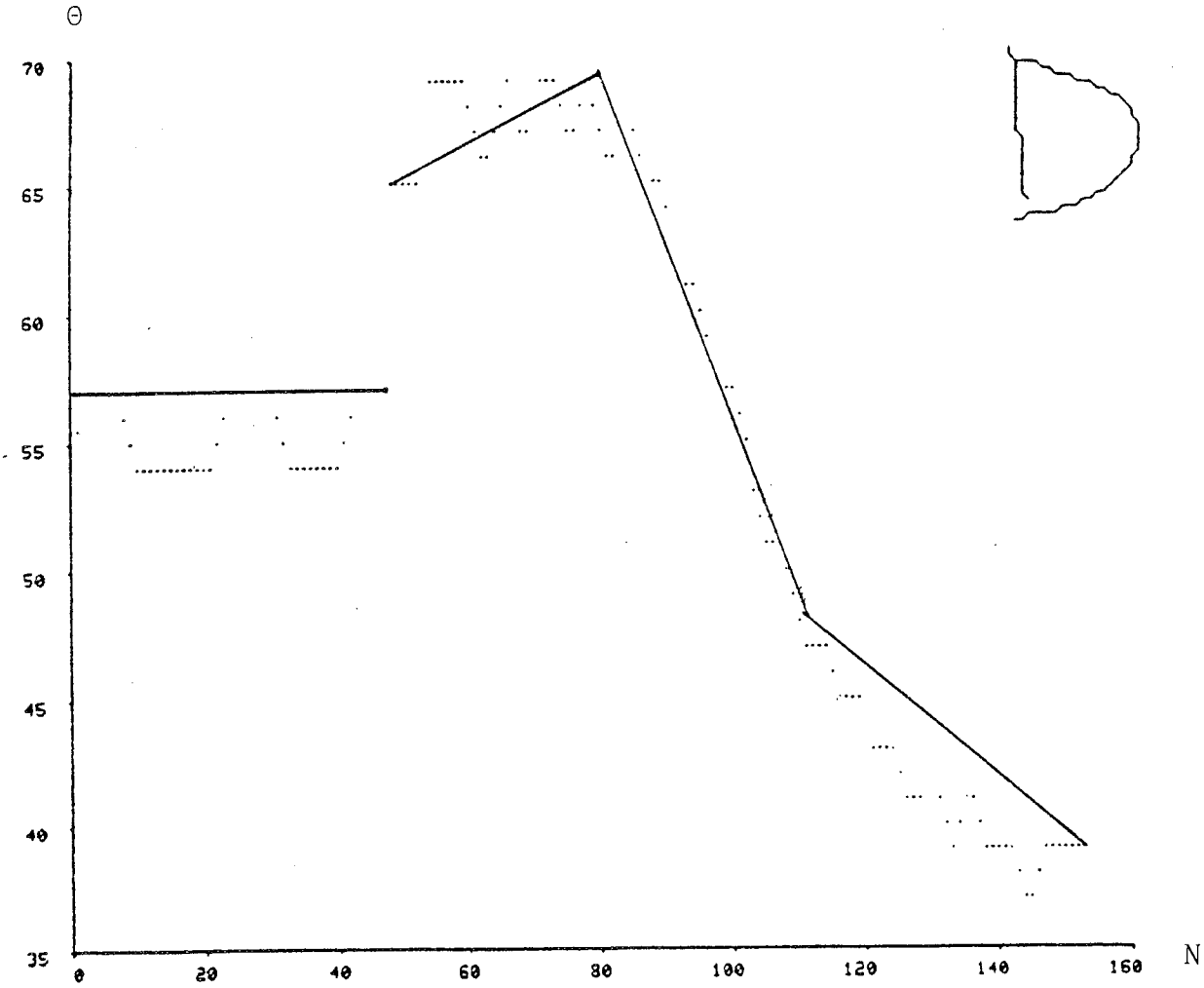


FIGURE 4.4. Segmentation initiale d'un caractère représenté en Θ - N

En général, le niveau de bruit est assez élevé, il est nécessaire d'utiliser la méthode des moindres carrées pour déterminer la meilleure droite. Cette méthode nous donnera la meilleure estimation de P. Une précaution importante à souligner est que pour les segments éloignés de l'origine des axes de références, on risque de perdre en précision sur les valeurs estimées. Pour avoir la même précision pour chaque segment, on doit donc déplacer l'origine des axes de références au début ou au milieu des segments à traiter.

Etant donné que les segments sont estimés indépendamment l'un

et l'autre sans la contrainte de continuité. Ceci pose un problème pour l'estimation de $\Delta\Theta$ dans le cas où le trait est segmenté en plusieurs segments. Car il y a deux valeurs possibles de Θ à chaque vertex (voir par exemple le 2e trait du caractère D dans figure 4.5).

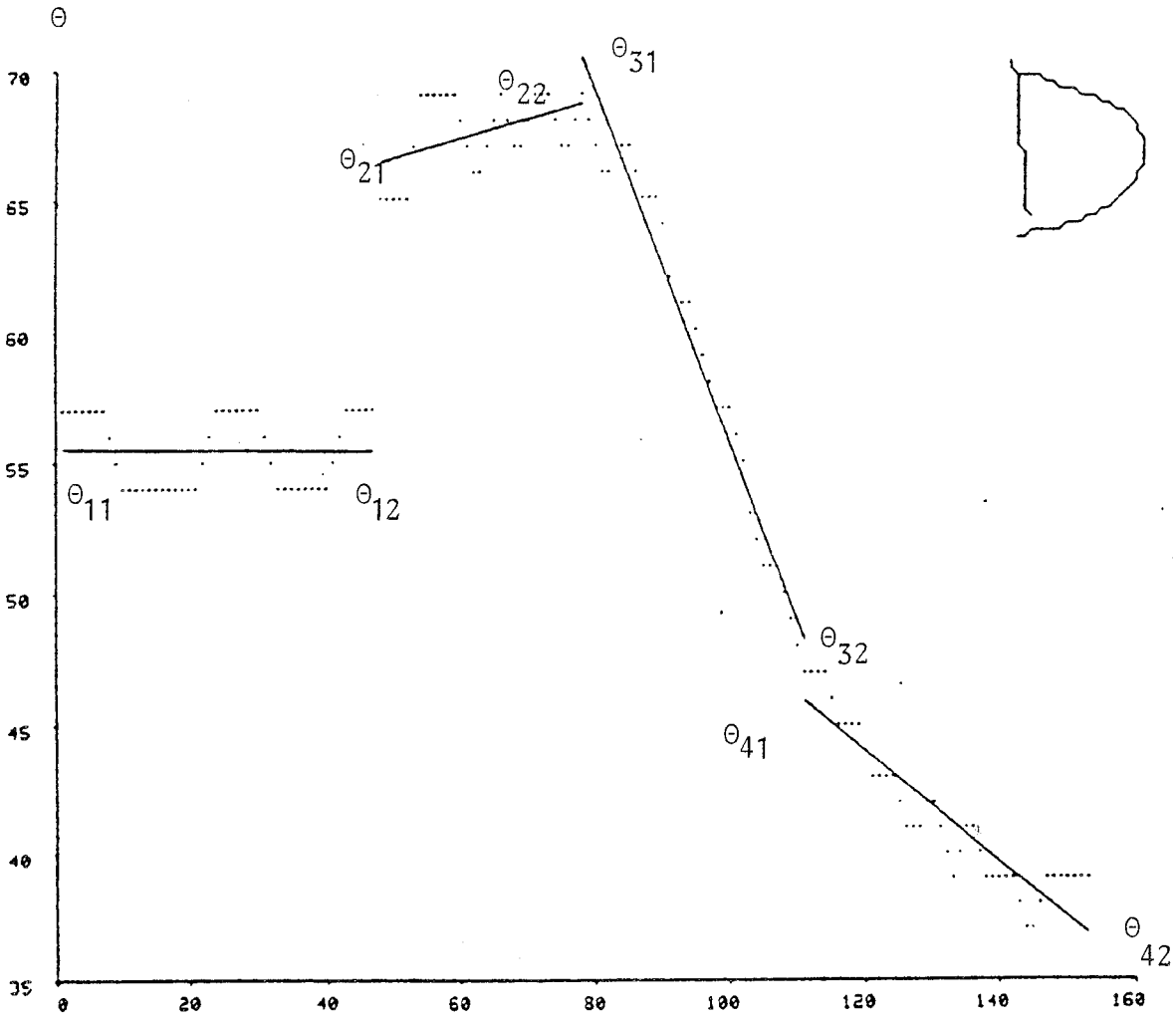


FIGURE 4.5. Approximation de courbes utilisant la méthode de moindres carrés.

Il est évident que ce serait mieux si nous pouvions avoir une espèce de continuité dans notre représentation. Cet objectif peut être atteint assez facilement car nous disposons actuellement de l'équation de chaque segment de droite, il suffit donc de remplacer les vertex par des intersections comme sur la figure 4.6. Remarquons que cette nouvelle segmentation est beaucoup plus précise que la segmentation originale de figure 4.4., en conséquence, les valeurs de P, $\Delta\Theta$ et N que nous allons obtenir seront meilleures.

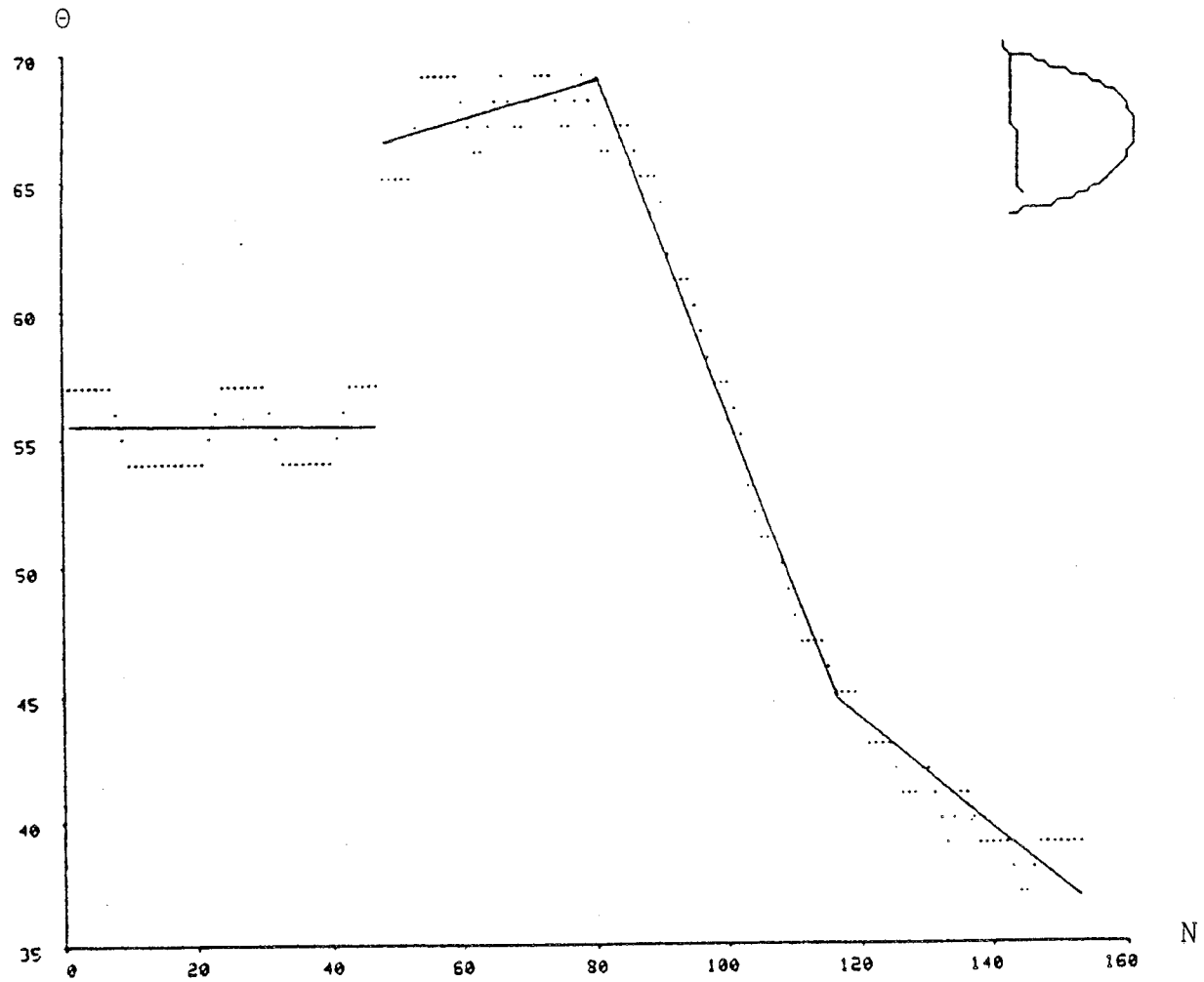


FIGURE 4.6. Segmentation initiale d'un caractère représenté en Θ - N.

4.3.2.2. Distribution des paramètres P, ΔΘ et N

Il est nécessaire d'avoir une idée sur la distribution des paramètres P, ΔΘ et N. P représente actuellement la courbure et c'est un paramètre normalisé. Par contre, N et ΔΘ ne le sont pas, ils dépendent de la dimension du caractère. Donc, si on utilise ces deux derniers comme seuils, il faut qu'ils soient fonction de la dimension ou plus précisément, en fonction de l'ordre de lissage k (voir section 4.2.3.).

Dans figure 4.7 et figure 4.8, nous voyons la distribution de | ΔΘ | en fonction de | P | et la distribution de | P | en fonction de N pour les caractères de la base : Base-Lo-Ap.

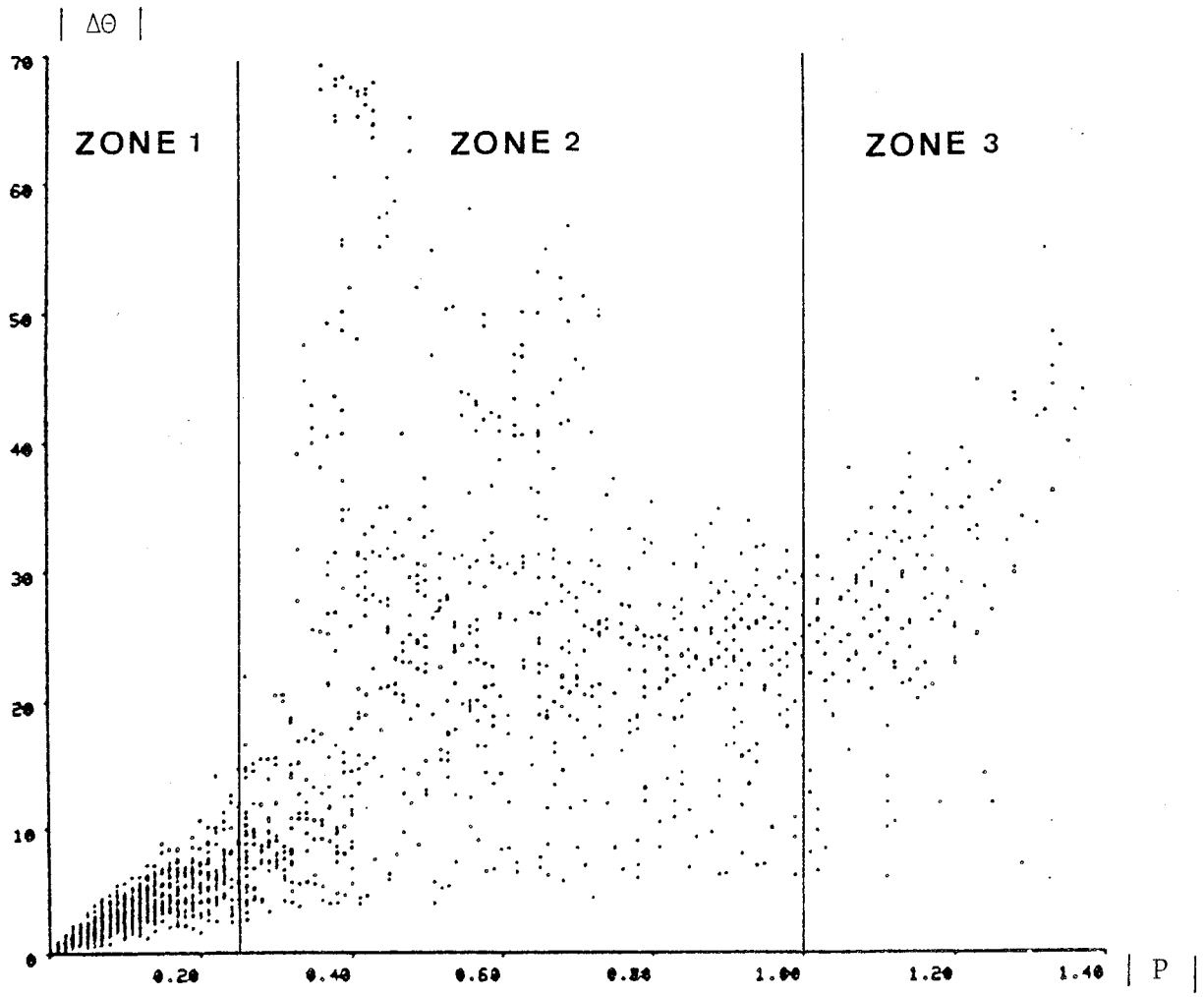


FIGURE 4.7 La distribution de | ΔΘ | en fonction de | P |

Il est clair que l'on ne peut pas déterminer des seuils appropriés à partir de ces figures. Malgré cela, on peut voir l'existence de quelques zones privilégiées. Par exemple, ZONE 1 : zone de segment droites (d), ZONE 2 : zone de segment de courbes (c^- et c^+) et ZONE 3 : zone de breaks (b^- et b^+).

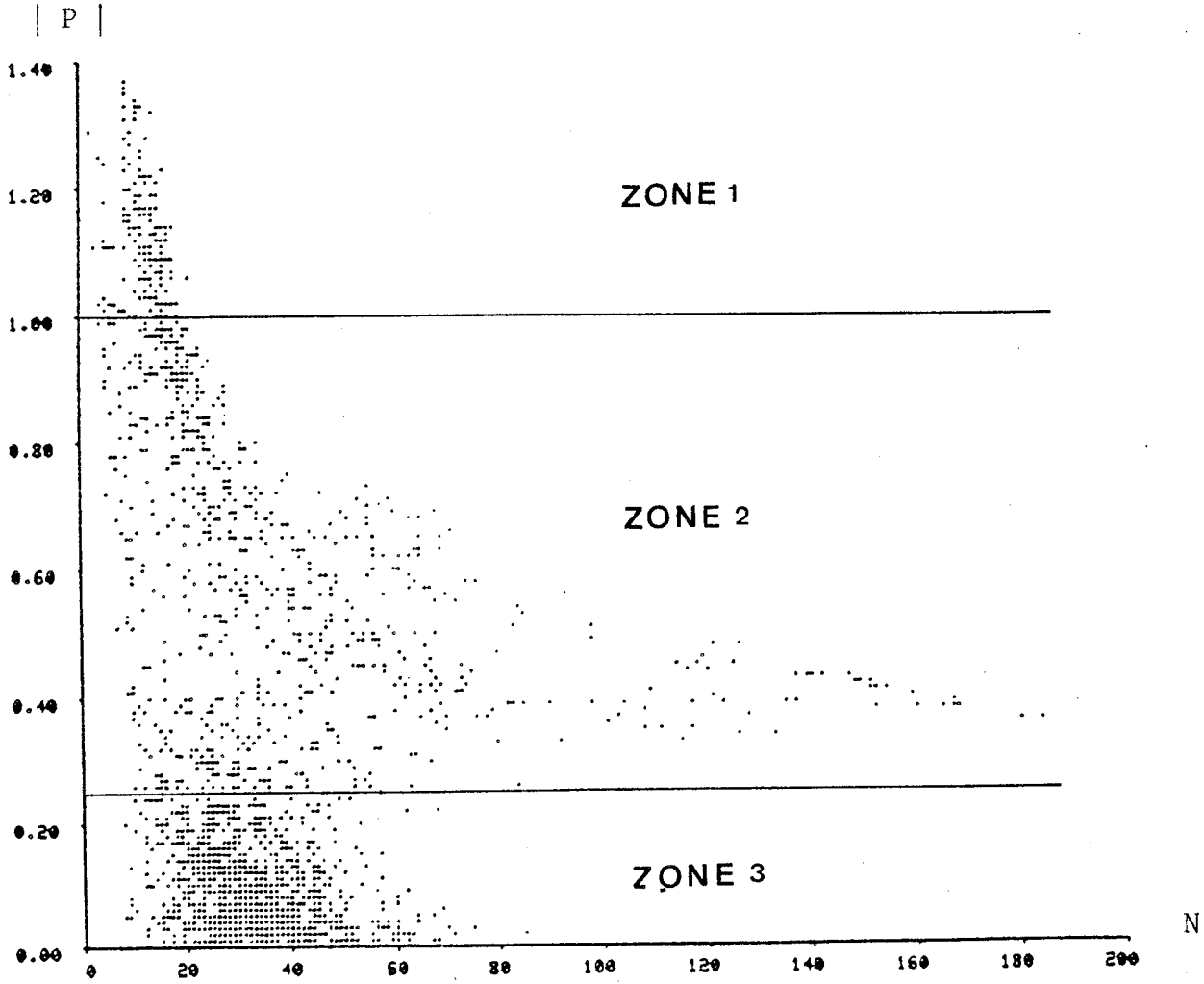


FIGURE 4.8 La distribution de | P | en fonction de N

Il n'existe pas de séparations nettes dans ces données. Pour pouvoir bien choisir les limites de séparation, il nous faut la connaissance à priori de la nature de chaque donnée, car sans cette

connaissance, il est impossible de dire si un choix est approprié ou non. Par contre, il n'y a pas de critères qui nous permettent de dire si un segment est une droite ou une courbe, même pour un esprit humain, c'est souvent une décision subjective.

4.3.2.3. Choix des limites de séparation

Il est difficile de choisir les seuils appropriés à partir de la figure 4.7 ou figure 4.8 car nous ne connaissons pas la signification de chaque point dans ces figures. Pour aboutir à un choix raisonable, nous avons utilisé une base de données plus réduite qui contient 36 échantillons, soit un échantillon pour chacun de 36 caractères alphanumériques. Après la segmentation de caractère en plan Θ - S, nous donnons à chaque segment un nom qui semble d'être le plus approprié par rapport à sa forme originale. Nous traçons en suite chaque segment sur le plan $|P|$ - N (voir figure 4.9).

Nous pouvons constater que les classes sont pratiquement linéairement séparables sur cette figure et d'ailleurs, les limites peuvent être choisies assez facilement. Les limites choisies sont montrées dans la figure 4.10 et superposées sur la figure 4.9.

En résumé, un segment i en plan Θ - S est classifié comme :

b^- si $\quad \Delta\Theta \leqslant -4K - 2$ et $P > 0{,}95$

c^- si $\quad -4K - 2 < \Delta\Theta < -K - 1$ et $0{,}25 < P < 0{,}95$

d si $\quad -K - 1 \leqslant \Delta\Theta \leqslant K + 1$ ou $-0{,}25 \leqslant P \leqslant 0{,}25$

c^+ si $\quad k + 1 < \Delta\Theta < 4K + 2$ et $-0{,}95 < P < 0{,}25$

b^+ si $\quad \Delta\Theta \geqslant 4K + 2$ et $P \leqslant -0{,}95$

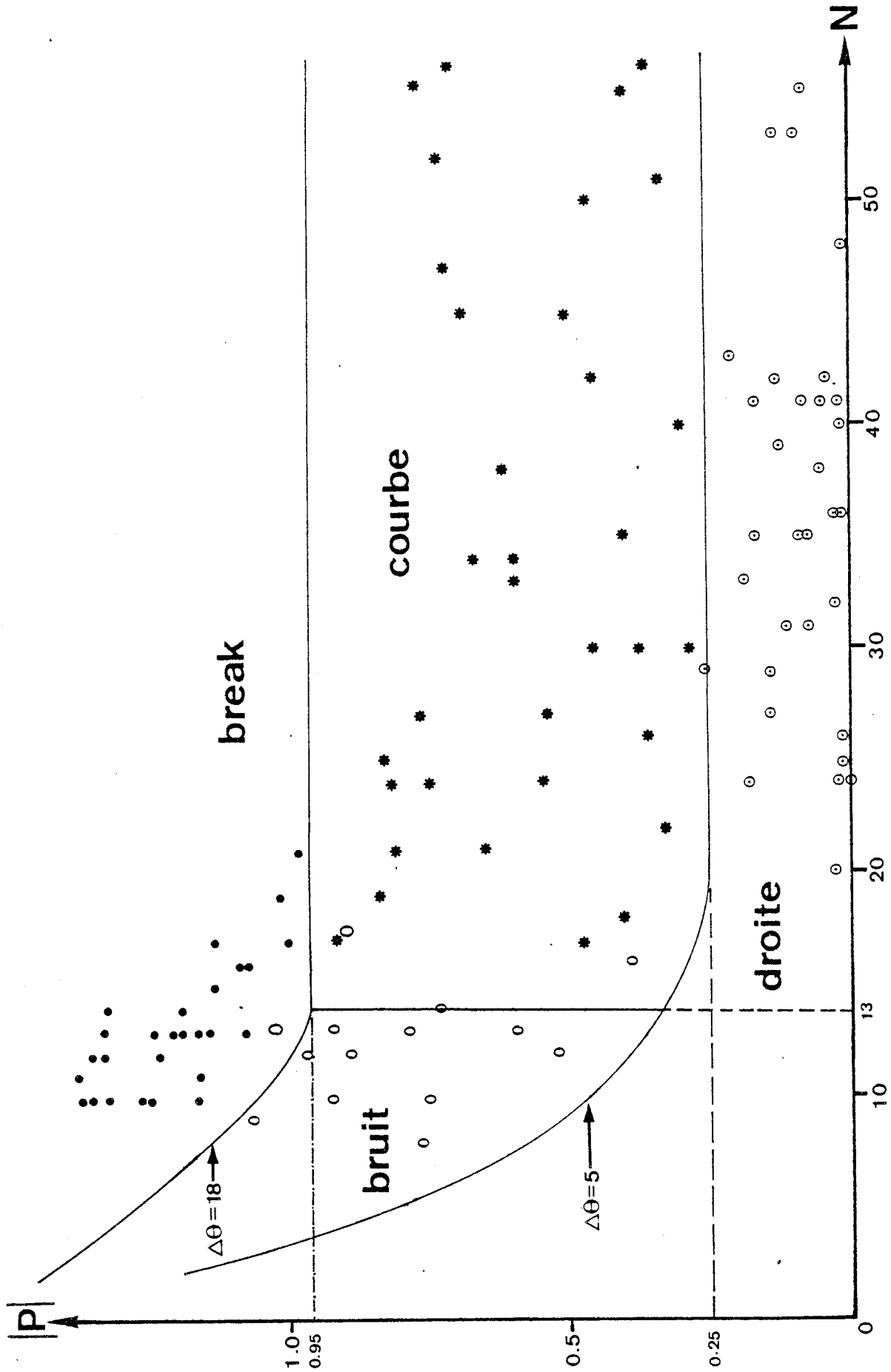


FIGURE 4.9 La distribution de paramètres et les limites de séparations superposées.

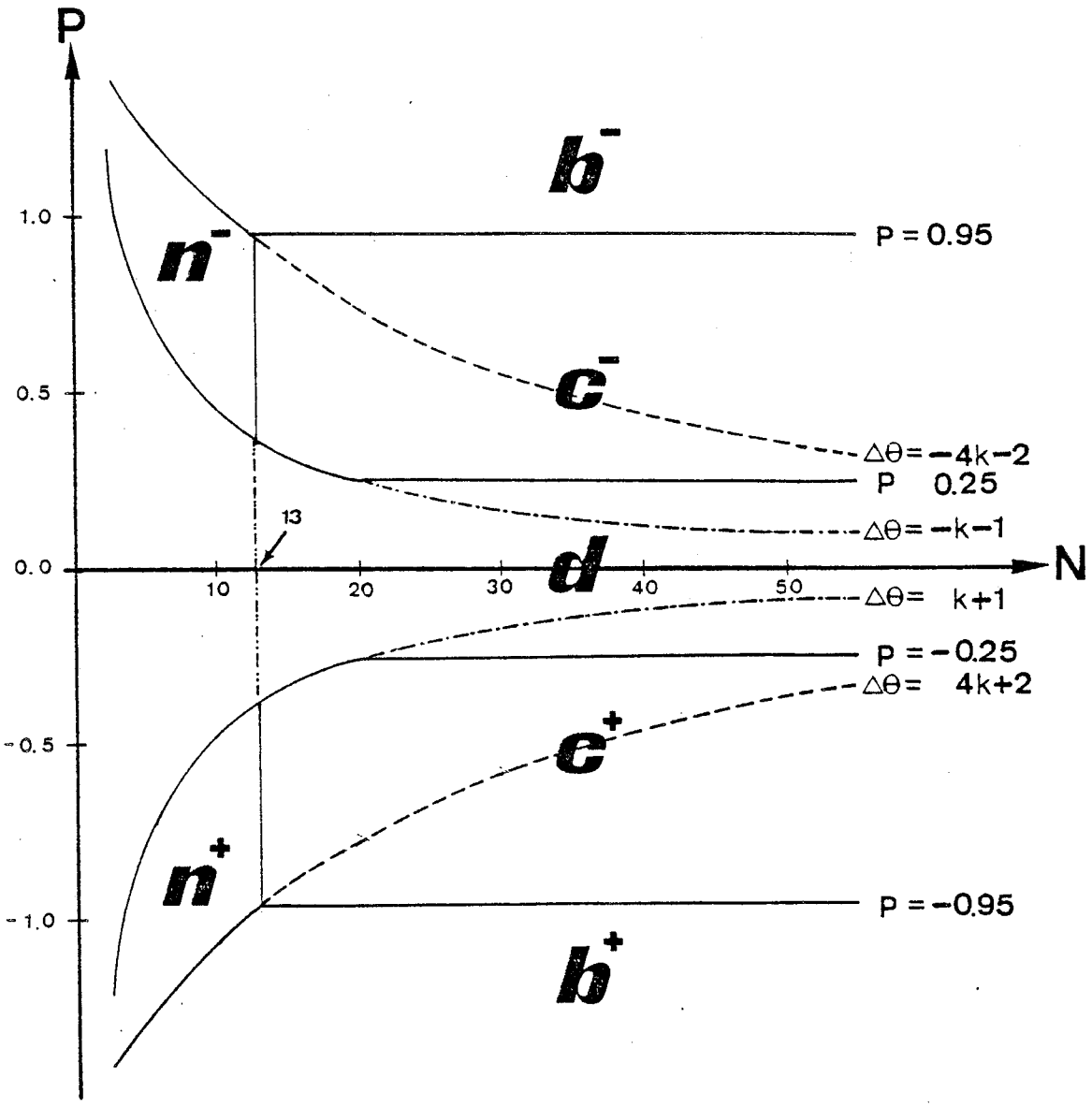


FIGURE 4.10 Choix des limites des séparations.

n si segment i est le premier ou dernier segment d'un trait classé précédemment comme c^- ou c^+ et N < 15.

Avec ces limites déterminées, nous decrivons donc à cette étape les caractères entrés par des chaînes de primitives.

Par exemple, la description du caractère B de la figure 4.2 est :

$$\underbrace{n\ d\ f}_{\text{1er trait}}\ \underbrace{c^+\ c^+\ c^+\ b^+\ c^+\ c^+\ c^+\ f}_{\text{2e trait}}$$

et le caractère D (voir figure 4.6.) est décrit :

$$\underbrace{d\ f}_{\text{1er trait}}\ \underbrace{d\ c^+\ d\ f}_{\text{2e trait}}$$

4.4. DESCRIPTION GLOBALE ET ATTRIBUTS

Nous avons deux problèmes à résoudre à ce niveau. Le premier est de réduire le nombre de variations des descriptions primitives. Le deuxième est de chercher les attributs appropriés qui rendront les descriptions finales des caractères distinctes.

4.4.1. La description globale

Pour transformer une chaîne de primitives en une chaîne de caractéristiques globales, nous utilisons un automate fini. Nous donnons dans l'Appendice E les détails de l'automate construit. A la sortie de l'automate, nous avons une description globale de carac-

tère qui comprend les caractéristiques globales suivantes :

L : Une droite
P : Une courbe de courbure +ve
N : Une courbe de courbure -ve
B : Un coin
D : Un point
C : Une courbe fermée
/ : Une levée de plume.

Par exemple, les descriptions globales des caractères B et D de l'exemple précédent seront :

B : L/ PBP/
D : L/ P/

Nous montrons dans le tableau 4.1. les exemples des descriptions globales et les caractères correspondants pour la base : Base-Lo-Ap. Nous voyons que la description globale seule ne suffit pas pour differencier tous les caractères. Il faut une description plus détaillée.

4.4.2. Les attributs

Prenons un exemple pour montrer ce qui manque dans notre description à ce niveau. La description globale L/L/L/ dans tableau 4.1 décrit soit le caractère A, F, H ou I. Pour différencier ces 4 caractères. Il est nécessaire d'ajouter plus d'informations sur la description globale. Dans ce cas par exemple, nous pouvons différencier les 4 caractères par la pente de chaque droite. C'est-à-dire, utilisant les attributs d'orientations suivants :

TABLEAU 4.1 Description globale et caractères

DESCRIPTION GLOBALE	CARACTERES	DESCRIPTION GLOBALE	CARACTERES
C/L/	Ø, Q	NBLBL/	8
N/L/	4, Ø, Q	CBL/	9
LBL/L/	1, 4, 5, 7, J	L/L/L/	A, F, H, I
P/L/	1, 5, 7, J	L/PBP/	B
PBL/	2	L/LBLBP/	B
PBP/	3	L/PBLBL/	B
LBLBP/	3	L/LBLBLBL/	B
LBNP/	3	L/P/	D, J, P
LBLBLBL/	3, 8, W	L/LBL/L/	E
PBLBL/	3	L/N/L/	E
LBP/L/	5	N/L/L/	E, G
LBL/P/	5, D, J	L/LBL/	J, K, N, P, Y
LBLBL/L/	5	L/N/	K, N, Y
NBL/L/	5	L/LBLBL/	M, R
N/	6, C, L, O, U	L/NBL/	M
LBL/	7, L, V	C/	O
LBLBL/	7, S, U, Z	L/PBL/	R
NP/	8, S	L/PN/	R
LBP/	8, S	NBL/	S
NBP/	8, S	L/L/	T, X

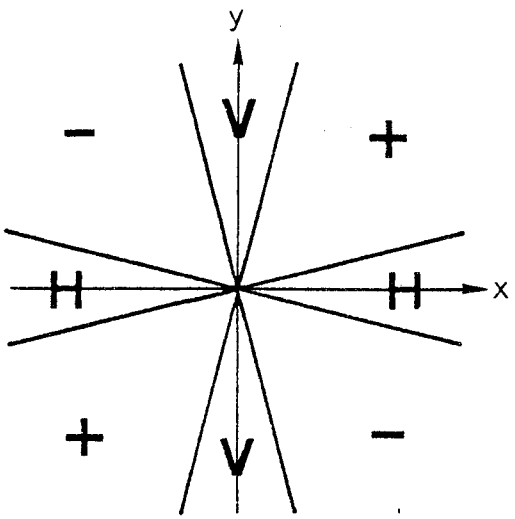


FIGURE 4.11.

Les attributs d'orientation.

V	:	Verticale ;		|pente| >	75°
H	:	Horizontale ;		|pente| <	15°
-	:	Négative ;	- 75°	pente <	- 15°
+	:	Positive ;	15°	pente <	75°

Nous avons donc les descriptions complètes suivantes :

A : L(+) / L(-) / L(H) /
F : L(H) / L(H) / L(V) /
H : L(V) / L(H) / L(V) /
I : L(H) / L(V) / L(H) /

Ce type d'attributs n'est pas suffisant pour différencier tous les 36 caractères. Par exemple nous avons la même description suivante : L(V) / P(V) /, pour les caractères P et D.

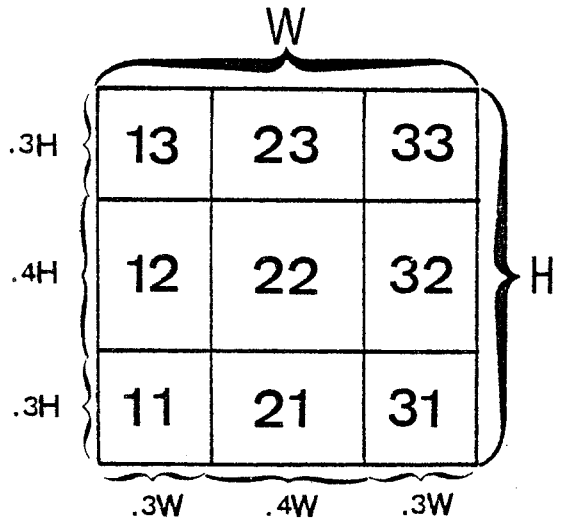


FIGURE 4.12.

Les attributs de Position.

Après un certain nombre d'essais, nous avons trouvé les attributs suivants, qui conviennent mieux à nos problèmes.

Nous divisons la case qui couvre le caractère (après normalisation) en 9 zones (voir Figure 4.12) et nous attachons à la description globale du caractère une phrase d'attributs qui marquent la zone de départ et d'arrivée de chaque caractéristique.

Exemple.

La description complète de caractère B est :

L / PBP / 13 11 13 12 11

Cette description peut être traduite en paroles comme suit :

Ce caractère a une droite qui part de la zone 13 jusqu'à la zone 11. Une courbe positive qui part de la zone 13, termine par une rupture à la zone 12 suivie par une autre courbe positive qui se termine à la zone 11.

Nous montrons dans le tableau 4.2. les exemples des descriptions complètes du caractère.

TABLEAU 4.2 Les exemples des descriptions complètes de caractères.

S/N°	CARACTERE	DESCRIPTION COMPLETE	
		Description globale	Attributs
1	0	C/ C/	1212 1313
2	1	LBL/ L/L/ NBL/	113331 12232321 123321
3	2	PN/ PBL/	231131 121131
4	3	PBP/ PNP/	231211 23333211
5	4	N/L/ LBL/L/	23313231 2311323231
6	5	P/L/ LBP/L/	13111333 1312112333
7	6	N/ N/	2311 3311
8	7	P/L/ LBL/L/	13211232 1333211232
9	8	PN/ NP/	123313 232123
10	9	CBP/ NBLBP/	333311 33223311

Tableau 4.2 (suite)

11	A	LBLBL/L/ LBP/L/	231123311232 1311311232
12	B	LBPBP/ L/PBP/	12112211 1311132221
13	C	LBN/ N/	233331 3331
14	D	LBP/ L/P/	232111 23221311
15	E	N/L/L/ L/L/L/L/	13311232 1333 1311133312321131
16	F	LBP/L/ L/L/L/	1311331232 131113331232
17	G	NBL/ N/L/	232232 33322232
18	H	L/L/L/ LBP/L/	131133311232 1311223331
19	I	L/L/L/ LBNPBL	232113331131 1333221131
20	J	P/L/ L/P/	23111333 13332311
21	K	L/LBL/ LBL/L/	1311331231 1311332231
22	L	N/ LBL/	1331 231131
23	M	LBLBNBL/ LBLBLBLBL/	1311133331 131113213331

Tableau 4.2. (suite)

24	N	LBLBLBL/ LBLBN/	1311133133 13111333
25	O	C/L/ N/L/	23233311 12223311
26	P	LBP/ L/P/	131122 13111312
27	Q	N/L/ C/L/	13222231 23232231
28	R	LBPN/ L/PBL/	12112231 1311131231
29	S	NP/ LBNP/	333211 23332211
30	T	L/L/ L/L/	23211333 13332321
31	U	N/ NBL/	1333 133331
32	V	LBL/ L/L/	132133 13213321
33	W	LBLBLBL/ NBLBL/	1311233133 12232133
34	X	L/L/ L/L/	33111331 13313311
35	Y	L/L/ NBL/	13223311 133321
36	Z	LBLBL/ LBN/	13331131 132331

4.5. APPRENTISSAGE ET CLASSIFICATION

4.5.1. Apprentissage

Après l'étape d'extraction de caractéristiques, le caractère inconnu est décrit par deux "phrases" de symbols. La première décrit les caractéristiques globales du caractère et la seconde comporte les informations spatiales de chaque caractéristique. Il s'agit alors, pour le classifieur, de construire les modèles de références et la règle de décision.

Nous voulons que notre système de reconnaissance soit capable de construire ses modèles automatiquement sans faire intervenir l'utilisateur. Ceci implique que nous devons construire notre classifieur sous une forme de dictionnaire. Nous construisons notre dictionnaire de la manière suivante :

i) Pendant l'apprentissage, le scripteur écrit plusieurs échantillons du caractère à apprendre sur la tablette où on lit dans la base de données les échantillons concernés.

ii) Nous rassemblons les caractères qui ont la même description globale en groupes et nous calculons la distance entre les phrases des attributs.

iii) Nous construisons à partir de la "matrice de distance" les phrases modèles et les phrases complémentaires de chaque groupe qui sont précisément les modèles de références.

Pour montrer plus clairement la démarche, nous allons considérer un exemple :

Exemple

Après l'étape d'extraction de caractéristiques, nous avons les descriptions suivantes pour 80 échantillons du caractère "3" de la base de données multiscripteur (nous la préciserons en détail dans la section suivante) :

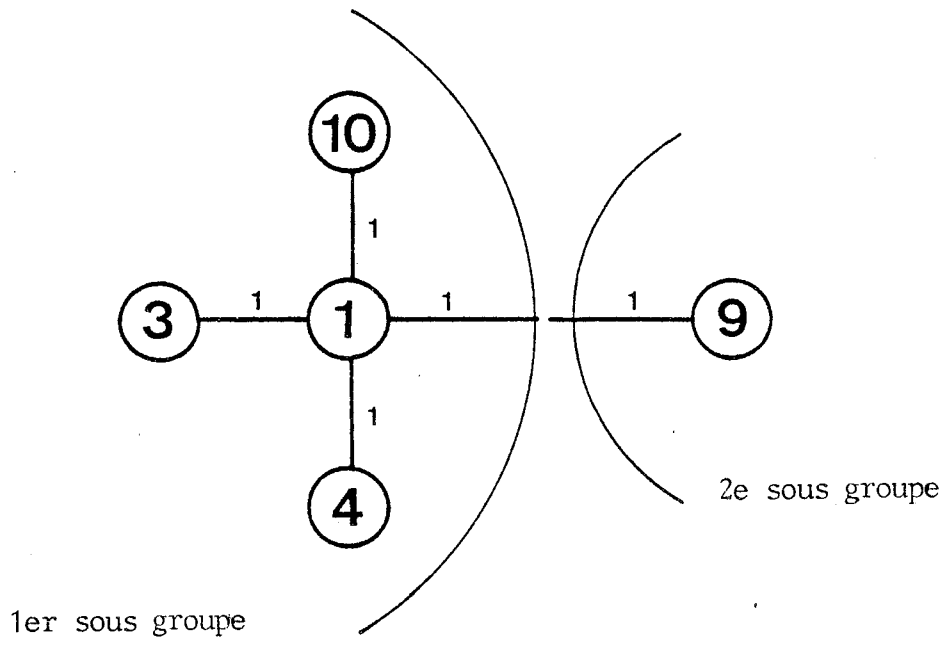
S/N°	Nb d'échantillons	description globale	attributs
1	10	PBP/	132221
2	2	PNP/	13223111
3	1	PBP/	122221
4	26	PBP/	132211
5	8	LBLBP/	13332211
6	3	LBLBP/	13232221
7	1	PNP/	23333211
8	4	PNP/	23323211
9	14	PBP/	131211
10	3	PBP/	131221
11	5	LBLBP/	13332221
12	3	PBLBL/	13123111

Les échantillons 1, 3, 4, 9 et 10 constituent le premier groupe, ces échantillons ont la même description globale PBP/.

La matrice de distance de ce groupe est :

		1	3	4	9	10
1	(132221)	0	1	1	2	1
3	(122221)	1	0	2	3	2
4	(132211)	1	2	0	1	2
9	(131211)	2	3	1	0	1
10	(131221)	1	2	2	1	0

Nous voyons que la première phrase (132221) est la phrase qui a le plus de voisins de distance 1. géométriquement, nous avons la distribution suivante :



Nous divisons ce groupe dans deux sous groupes. Le premier sous groupe se rassemble autour de 1. c'est-à-dire, les phrases 1, 3, 4 et 10. Le deuxième sous groupe contient simplement la phrase 9. Nous avons donc les modèles suivants pour ce groupe.

1er sous groupe : 1 3 2 2 2 1 - 2 1 - 1 -

2e sous groupe : 1 3 1 2 1 1 - - - - - -

La deuxième phrase est une phrase complémentaire qui mémorise les variations possibles. (- signifie qu'il n'y a pas de changement).

Nous continuons la même démarche pour les autres groupes et nous obtenons les modèles de références de ces 80 échantillons du caractère "3" suivants :

Caractère "3"	PBP/	13 22 21	-21-1-
		13 12 11	------
	PNP/	23 33 32 11	---2----
		13 22 31 11	--------
	LBLBP/	13 33 22 11	------2-
		13 23 22 21	--------
	PBLBL/	13 12 31 11	--------

L'apprentissage peut se faire de manière récursive dans ce système, nous prenons avec les nouveaux échantillons, les modèles de références du caractère concerné et nous suivons exactement la même démarche.

Etant donné que l'apprentissage est fait séquentiellement, un caractère à la fois, il risque d'y avoir des recouvrements entre les caractères. Donc, après avoir appris tous les caractères alpha-

numériques, nous devons corriger le classifieur pour enlever les modèles ambigus. La correction est faite de la même manière que la procédure d'apprentissage, nous calculons les distances entre les classes et dans les cas ambigus, nous supprimons les modèles qui apparaissent peu statistiquement.

4.5.2. *Classification*

Nous pouvons actuellement construire notre classifieur sous forme d'arbes de décision mais nous préférons garder la forme montrée dans l'exemple précédent. La raison est que sous cette forme, nous pouvons facilement mettre à jour le classifieur. C'est-à-dire, nous pouvons avoir l'apprentissage récursif.

Pour classifier un caractère, nous procédons de la façon suivante :

i) Nous comparons la description globale du caractère entré avec celles du dictionnaire.

ii) Si nous ne trouvons pas la même description globale dans le dictionnaire, nous rejetons le caractère.

iii) Sinon, nous comparons les attributs, si nous trouvons les mêmes attributs ou la distance entre les attributs sont plus petites qu'un certain seuil. Alors, nous classifions le caractère entré avec l'identité du modèle dans le dictionnaire qui satisfait cette condition. Si la distance est plus grande que le seuil, nous rejetons alors le caractère entré.

4.6. RESULTATS EXPERIMENTAUX

Pour bien évaluer les performances de notre algorithme, nous avons utilisé plusieurs bases de données en diverse combinaisons. Nous allons d'abord définir la structure de chaque base de données et nous donnerons ensuite les résultats obtenus :

4.6.1. Les bases de données

Essentiellement, il y a deux types de bases de données : la base de données monoscripteur et la base de données multiscripteur. Nous avons déjà défini le premier type dans la section 3.5.2.1. précédente. La base de données multiscripteur est préparée par la société OPTION, elle est créée par 50 scripteurs différents choisis au hasard. Il est demandé à chaque scripteur d'écrire 4 échantillons pour chacun de 36 caractères alphanumériques. Donc, un total de 7 200 caractères étaient disponibles. Les scripteurs ont dû suivre quelques conventions : par exemple : entre les caractères 1 et I, et entre la lettre O et le chiffre 0. A part cela, les scripteurs étaient totalement libres d'écrire suivant leurs habitudes sans aucune contrainte. Nous avons découpé cette base de données en 4 parties et leurs compositions sont détaillées dans le tableau 4.3. Nous montrons aussi une partie de cette base de données dans l'Appendice F.

4.6.2. Le taux de reconnaissance

Les résultats d'essais sont résumés dans les tableaux 4.4 à 4.10. Nous avons les commentaires suivants.

Monoscripteur - Alphanumérique (Cf. tableau 4.4)

i) Nous avons obtenu un taux de reconnaissance très élevé.

TABLEAU 4.3 Les compositions des bases de données.

$S/_{N°}$	NOM DE LA BASE	COMPOSITION DE LA BASE
1	Base-Mu-Ap-Ch (Base-Multiscripteur-Apprentissage-Chiffres)	Multiscripteur 50 scripteurs différents 2 échantillons par chiffre/scripteur 1000 échantillons au total Ussage : Apprentissage
2	Base-Mu-Te-Ch (Base-Multiscripteur-Test-Chiffres)	La même composition que la base Base-Mu-Ap-Ch susmentionnée Ussage : Test
3	Base-Mu-Ap-Le (Base-Multiscripteur-Apprentissage-Lettres)	Multiscripteur 50 scripteurs différents 2 échantillons par lettre/scripteur 2600 échantillons au total Ussage : Apprentissage
4	Base-Mu-Te-Le (Base-Multiscripteur-Test-Lettres-	La même composition que la base Base-Mu-Ap-Le susmentionnée Ussage : Test
5	Base-Lo-Ap (Base-Monoscripteur-Apprentissage)	Monoscripteur 24 échantillons pour chacune de 69 manières d'écrire les 36 caractères $A/_N$. 24 x 69 = 1656 échantillons au total Ussage : Apprentissage
6	Base-Lo-Te (Base-Monoscripteur-Test)	Monoscripteur $\sim$ 27 échantillons par caractère $A/_N$. 1000 échantillons au total Ussage : Test

TABLEAU 4.4 Résultats sur les bases de données - Alphanumériques - Monoscripteur

S/N°	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Lo-Ap	Base-Lo-Ap	16/1656	-	0,97	99,03
2	Base-Lo-Ap	Base-Lo-Te	12/1000	6/1000	1,80	98,20
3	Base-Lo-Ap	Base-Lo-Ap + Base-Lo-Te	28/2656	6/2656	1,28	98,72
4	Base-Lo-Te	Base-Lo-Te	6/1000	4/1000	1,00	99,00
5	Base-Lo-Te	Base-Lo-Ap	51/1656	63/1656	6,88	93,12
6	Base-Lo-AP + Base-Lo-Te	Base-Lo-Ap + Base-Lo-Te	25/2656	3/2656	1,05	98,95

TABLEAU 4.5 Résultats sur les bases de données - Chiffres - Multiscripteur

(Avant la correction de classifieur)

S/N°	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Mu-Ap-Ch	Base-Mu-Ap-Ch	12/1000	-	1,20	98,80
2	Base-Mu-Ap-Ch	Base-Mu-Te-Ch	37/1000	73/1000	11,00	89,00
3	Base-Mu-Ap-Ch	Base-Mu-Ap-Ch + Base-Mu-Te-Ch	49/2000	73/2000	6,10	93,90
4	Base-Mu-Ap-Ch + Base-Mu-Te-Ch	Base-Mu-Ap-Ch + Base-Mu-Te-Ch	23/2000	1/2000	1,20	98,80

TABLEAU 4.6 Résultats sur les bases de données - Alphabets - Multiscripteur

(Avant la correction de classifier)

S/N°	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Mu-Ap-Le	Base-Mu-Ap-Le	253/2600	10/2600	10,16	89,84
2	Base-Mu-Ap-Le	Base-Mu-Te-Le	367/2600	161/2600	20,31	79,69
3	Base-Mu-Ap-Le	Base-Mu-Ap-Le + Base-Mu-Te-Le	620/5200	171/5200	15,21	84,79
4	Base-Mu-Ap-Le + Base-Mu-Te-Le	Base-Mu-Ap-Le + Base-Mu-Te-Le	487/5200	20/5200	9,75	90,25

TABLEAU 4.7 Résultats sur les bases de données - Chiffres - Multiscripteur

(Après la correction du classifieur)

$S/_{N°}$	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Mu-Ap-Ch	Base-Mu-Ap-Ch	$5/_{1000}$	-	0,05	99,50
2	Base-Mu-Ap-Ch	Base-Mu-Te-Ch	$33/_{1000}$	$73/_{1000}$	10,60	89,40
3	Base-Mu-Ap-Ch	Base-Mu-Ap-Ch + Base-Mu-Te-Ch	$38/_{2000}$	$73/_{2000}$	5,55	94,45
4	Base-Mu-Ap-Ch + Base-Mu-Te-Ch	Base-Mu-Ap-Ch + Base-Mu-Te-Ch	$11/_{2000}$	$1/_{2000}$	0,60	99,40

TABLEAU 4.8 Résultats sur les bases de données - Alphabets - Multiscripteur

(Après la correction du classifieur)

S/N°	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Mu-Ap-Le	Base-Mu-Ap-Le	129/2600	10/2600	5,35	94,65
2	Base-Mu-Ap-Le	Base-Mu-Te-Le	272/2600	187/2600	17,65	82,35
3	Base-Mu-Ap-Le	Base-Mu-Ap-Le + Base-Mu-Te-Le	401/5200	197/5200	11,50	88,50
4	Base-Mu-Ap-Le + Base-Mu-Te-Le	Base-Mu-Ap-Le + Base-Mu-Te-Le	218/5200	20/5200	4,58	95,42

TABLEAU 4.9 Résultats sur les bases de données - Alphanumériques - Multiscripteur

(Avant la correction de classifieur)

S/N°	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	480/3600	10/3600	13,61	86,39
2	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	Base-Mu-Te-Ch + Base-Mu-Te-Le	735/3600	226/3600	26,69	73,31
3	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	Base-Mu-Ap-Ch + Base-Mu-Ap-Le + Base-Mu-Te-Ch + Base-Mu-Te-Le	1215/7200	236/7200	20,15	79,85
4	Base-Mu-Ap-Ch + Base-Mu-Ap-Le + Base-Mu-Te-Ch + Base-Mu-Te-Le	-"-	965/7200	21/7200	13,69	86,31

TABLEAU 4.10 Résultats sur les bases de données - Alphanumériques - Multiscripteur

(Après la correction de classifieur)

S/N°	BASE D'APPRENTISSAGE	BASE DE TEST	Nb DE SUBSTI-TUTIONS	Nb DE REJETS	ERREUR TOTALE %	TAUX DE RECON-NAISSANCE
1	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	Base-Mu-Ap-Ch + Base-Mu-Ap-Ch	154/3600	10/3600	4,56	95,44
2	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	Base-Mu-Te-Ch + Base-Mu-Te-Le	391/3600	228/3600	17,19	82,81
3	Base-Mu-Ap-Ch + Base-Mu-Ap-Le	Base-Mu-Ap-Ch + Base-Mu-Ap-Le + Base-Mu-Te-Ch + Base-Mu-Te-Le	545/7200	238/7200	10,88	89,12
4	Base-Mu-Ap-Ch + Base-Mu-Ap-Le + Base-Mu-Te-Ch + Base-Mu-Te-Le	Base-Mu-Ap-Ch + Base-Mu-Ap-Le + Base-Mu-Te-Ch + Base-Mu-Te-Le	347/7200	21/7200	5,11	94,89

99.03 % avec la base d'apprentissage et 98.20 % avec la base d'essai. Donc, en gros, un taux de 98.72 % dans le cas monoscripteur. Ceci est légèrement plus élevé que celui obtenu avec la première méthode (98,60 %).

ii) Les résultats sont moins bons quand la base destinée aux essais (Base-Lo-Te) est employée pour l'apprentissage. Ceci montre que le contenu de la base d'apprentissage est très important. Il faut que la base de données soit très représentative, autrement dit, la base doit contenir toutes les variations possibles des caractères du scripteur.

Multiscripteur - Chiffres (Cf. tableau 4.5 et 4.7)

i) Notons qu'avant la correction de classifieur (Cf. tableau 4.5), le taux de reconnaissance était de 93,9 %, mais après la vérification de classifieur, le taux s'est élevé à 94,5 % (Cf tableau 4.7).

ii) Quand tous les données sont utilisées (2000 échantillons) pendant l'apprentissage, nous avons un taux de reconnaissance de 99,40 %, ce taux est sûrement suffisant pour une application précise.

Multiscripteur -lettres (Cf. tableaux 4.6 et 4.8)

i) Nous avons obtenu un taux moyen de 88,5 %, ce taux n'est pas encore suffisant pour une application industrielle.

ii) Notons qu'après la correction du classifieur, le taux augmente en moyenne de 5 %. Cette augmentation est très élevée et elle montre clairement l'intérêt du "clustering".

Multiscripteur - Alphanumériques (Cf. Tableaux 4.9 et 4.10)

i) Les taux de reconnaissance après la correction de classifieur est en moyenne 10 % meilleur que ceux avant la correction.

ii) Nous pouvons dire que le taux de reconnaissance de cet algorithme pour les caractères alphanumériques écrits sans contraintes est d'environ 89 %.

4.7. CONCLUSION

Nous venons de présenter une méthode de reconnaissance de caractères alphanumériques manuscrits dans un environnement multiscripteur SANS contraintes. Nous avons utilisé une approche syntaxique qui est la plus appropriée à notre problème. Pour raison de conserver l'aspect intéactive de notre système de reconnaissance, nous employons un classifieur de type dictionnaire. Car il est plus facile de mettre à jour un classifieur de ce type.

Afin de réduire la taille du dictionnaire, nous proposons une méthode de groupement des expressions régulières particulières à notre système. La même procédure est utilisée pour examiner le dictionnaire afin de supprimer les cas ambigus.

Le programme complet a environ 900 instructions FORTRAN. avec les données du classifieur, il occupe à peu près 35K octets de l'espace mémoire. La vitesse de reconnaissance est de l'ordre de 300 ms par caractère. Cette vitesse est certainement suffisante pour une application en ligne. Notons que nous avons testé notre algorithme sur l'ordinateur NORD-10/S qui travaille en temps partagé, le temps réel de reconnaissance devrait être plus court.

o o

o

CHAPITRE V.

V. CONCLUSION.

Dans cette thèse, nous avons proposé deux méthodes différentes pour la reconnaissance en ligne de caractères alphanumériques manuscrits.

La première est une méthode statistique relativement simple qui a comme objectif de reconnaître les caractères générés par un seul scripteur ou par plusieurs scripteurs avec contraintes. La simplicité de cette méthode nous permet l'implantation sur un micro-ordinateur (INTEL 8086) . La performance de cette méthode est assez satisfaisante, nous avons obtenu un taux de reconnaissance de l'ordre de 99 % sur la base d'apprentissage et 98 % sur la base de test. En gros, un taux de réussite de 98,6 % dans le cas monoscripteur.

La deuxième méthode a pour but de reconnaître les caractères dans le cas multiscripteur SANS contraintes. Dans le cas monoscripteur, la performance de cette méthode est légèrement supérieure à celle de la première méthode. Dans le cas multiscripteur sans contraintes, nous avons un taux de reconnaissance de 89.12 %. Nous ne pouvons certainement pas considérer une telle performance comme très satisfaisante. Mais vu le nombre (50), les différentes habitudes (gaucher et droitier) de scripteurs d'une part et le nombre très petit d'échantillons d'apprentissages (2 échantillons par caractère par scripteur) d'autre part ; nous considérons cette performance plutôt bonne et encourageante car il s'agit d'un résultat initial, nous n'avons pas encore cherché à optimiser les différentes étapes de cet algorithme.

Nous voulons faire quelques dernières remarques concernant nos deux propositions dans la suite.

Proposition N° 1.

i) - Cet algorithme satisfait les besoins que nous pensons très importants d'un système de reconnaissance en ligne de caractères. Ce sont les suivants :

a) - L'apprentissage en ligne : la possibilité d'apprendre les caractères nouveaux sans être obligé de modifier manuellement les données du classifieur ou le programme.

b) - La possibilité pour un utilisateur d'entraîner le système pourqu'il soit plus adapté à sa façon décrire.

ii) - Malgré sa simplicité, cette méthode peut reconnaître tous les 36 caractères alphanumériques d'une manière satisfaisante.

iii) - On peut avoir un problème dans le cas où la longueur de segment est importante. Par exemple, entre les caractères X et Y, si on écrit le caractère Y de la manière suivante.  , on ne pourra plus les différencier car leurs vecteurs de caractéristiques seront les mêmes. Ceci indique que cette méthode a certaines limitations et on ne peut pas l'appliquer dans un cas plus général. C'est-à-dire, en dehors des caractères alphanumériques majuscules. Si on veut l'appliquer dans ce dernier cas, il faut plus de deux paramètres pour représenter chaque segment des caractères segmentés.

Proposition N° 2.

i) - Cet algorithme satisfait lui aussi les besoins importants d'un système de reconnaissance en ligne de caractères.

ii) - Cet algorithme peut être étendu pour reconnaître les autres types de caractères, par exemple : les symboles FORTRAN.

iii) - Pour pouvoir évaluer notre approche et estimer le taux de reconnaissance préliminaire, nous avons simplifié la procédure dans diverses étapes, à savoir :

a) - L'étape de segmentation et le choix de critère de segmentation.

b) - Le choix de seuils pour la classification de primitives.

c) - La construction de l'automate.

Nous pensons que la performance de notre système peut être améliorée si nous faisons une étude plus complète dans ces trois étapes.

iv) - La description globale de caractère est insensible à l'orientation du caractère. Donc, si nous orientons la case qui couvre le caractère, les attributs seront eux aussi insensibles à l'orientation du caractère. En conséquence, nous aurons une description complète qui ne dépendra pas de l'orientation du caractère.

Du point de vue de la complexité du programme, la deuxième méthode est plus complexe que la première (le deuxième programme est presque deux fois plus long que le premier). Donc plus chère. Mais les instructions utilisées dans ces deux algorithmes sont des instructions de bases, par conséquence, on peut facilement les implanter sur un système à base microprocesseur. En fait, les deux approches sont complémentaires et leur utilisation dépend du problème précis que l'on veut résoudre.

Dans la reconnaissance en ligne de caractères, il n'existe malheureusement pas de base de données commune (comme celle de IEEE data base pour les lectures optiques) dont nous pouvons nous servir pour évaluer notre algorithme de reconnaissance et pour comparer nos résultats avec ceux des autres auteurs. En fait, tant que l'on n'utilise pas la même base de données et la même méthode d'essai, on ne pourra jamais comparer d'une façon exacte les performances de deux algorithmes de reconnaissance différents. Néanmoins, nous trouvons que dans les conditions semblables, nos résultats sont comparables avec ceux des autres auteurs, particulièrement ceux résumés dans le tableau I et tableau IV de Suen et Al [34].

Nous pensons que pour un système de reconnaissance en ligne de caractères, il est plus important d'avoir une bonne performance dans le cas monoscripteur car le système ne peut avoir qu'un utilisateur à la fois. D'autre part, nous pouvons pour ce genre de système accepter un taux de reconnaissance raisonnable (~ 95 %) au début et ensuite améliorer sa performance au fur et à mesure de reconnaissance en utilisant les caractéristiques interactives du système. Nous l'avons d'ailleurs montré dans nos essais.

Nous pensons enfin que pour la reconnaissance de caractères manuscrits multiscripteurs, si nous ne posons aucune contrainte sur la façon d'écrire, nous ne pouvons jamais atteindre un taux de reconnaissance satisfaisant !.

° ° 0

APPENDICES.

APPENDICE A : *L'identification des paramètres avec un algorithme récursif.*

Une droite est décrite par :

$$y_{i,j} = a_i \, x_{i,j} + b_i + w_{i,j} \quad \text{(A.1)}$$

où a_i et b_i sont les paramètres inconnus qui caractérisent la $i^{ème}$ droite. $x_{i,j}$ et $y_{i,j}$ sont les $j^{ème}$ coordonnées du $i^{ème}$ segment à paramètriser.

$w_{i,j}$ représente le bruit.

Nous pouvons approximer le segment de droite par le modèle ajustable suivant :

$$\hat{y}_{i,j} = \hat{a}_{i,j} \, x_{i,j} + \hat{b}_{i,j} \quad \text{(A.2)}$$

où $\hat{a}_{i,j}$ et $\hat{b}_{i,j}$ sont les paramètres estimés.

Pour identifier les paramètres $\hat{a}_{i,j}$ et $\hat{b}_{i,j}$, nous utilisons une technique d'identification récursive [39] qui nécessite peu de calcul :

On définit le vecteur de paramètres ajustables par

$$\hat{P}_{i,j}^T = [\, \hat{a}_{i,j}, \hat{b}_{i,j} \,] \quad \text{(A.3)}$$

et l'erreur de prédiction par :

$$\hat{\varepsilon}_{i,j} = Y_{i,j} - \hat{Y}_{i,j} \quad \text{(A.4)}$$

Alors, l'algorithme d'adaptation est donné par :

$$\hat{P}_{i,j+1} = \hat{P}_{i,j} + \frac{F_{i,j}\, Z_{i,j}}{1 + Z_{i,j}^T\, F_{i,j}\, Z_{i,j}}\ \hat{\varepsilon}_{i,j} \qquad \text{(A.5)}$$

où $Z_{i,j}$ est défini comme

$$Z_{i,j}^T = [\, x_{i,j},\ 1\,] \qquad \text{(A.6)}$$

et la matrice de gain $F_{i,j}$ est :

$$F_{i,j+1} = F_{i,j} - \frac{F_{i,j}\, Z_{i,j}\, Z_{i,j}^T\, F_{i,j}}{\frac{1}{\lambda} + Z_{i,j}^T\, F_{i,j}\, Z_{i,j}} \quad ;\ 0 < \lambda < 2 \qquad \text{(A.7)}$$

Les valeurs φ et d de l'équation $x \cos\varphi + y \sin\varphi = -d$ sont calculées à la fin d'adaptation en utilisant les dernières valeurs estimées de $\hat{a}$ et $\hat{b}$.

APPENDICE B *Obtention de l'équation récursive de l'inverse de la matrice de covariances* $\underline{J}^{-1}$

La matrice de covariances pour (N+1) échantillons est donnée par :

$$\underline{J}_{N+1} = \frac{1}{N+1} \sum_{J=1}^{N+1} (\underline{X}_j - \underline{M}_{N+1}) (\underline{X}_j - \underline{M}_{N+1})^T \ldots\ldots\ldots \quad (B.1)$$

Par expansion, l'équation (B.1) devient :

$$\underline{J}_{N+1} = \frac{1}{N+1} \sum_{j=1}^{N} (\underline{X}_j - \underline{M}_{N+1}) (\underline{X}_j - \underline{M}_{N+1})^T + \frac{1}{N+1} (\underline{X}_{N+1} - \underline{M}_{N+1}) (\underline{X}_{N+1} - \underline{M}_{N+1})^T \ldots \quad (B.2)$$

Le vecteur de moyenne pour (N+1) échantillons est donnée par :

$$\underline{M}_{N+1} = \frac{1}{N+1} (N\underline{M}_N + \underline{X}_{N+1}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (B.3)$$

En introduisant l'équation (B.3) dans le premier terme de l'équation (B.2), nous obtenons :

$$\underline{J}_{N+1} = \frac{1}{N+1} \sum_{j=1}^{N} (\underline{X}_j - \underline{M}_N - \frac{1}{N+1} (\underline{X}_{N+1} - \underline{M}_N)) (\underline{X}_j - \underline{M}_N - \frac{1}{N+1} (\underline{X}_{N+1} - \underline{M}_N))^T + \frac{1}{N+1} (\underline{X}_{N+1} - \underline{M}_{N+1}) (\underline{X}_{N+1} - \underline{M}_{N+1})^T \ldots\ldots\ldots \quad (B.4)$$

En simplifiant le premier terme de l'équation (B.4) nous avons :

$$\underline{J}_{N+1} = \frac{1}{N+1} \left(N\underline{J}_N + \frac{1}{N} (\underline{X}_{N+1} - \underline{M}_{N+1}) (\underline{X}_{N+1} - \underline{M}_{N+1}) \right)^T$$

$$+ \frac{1}{N+1} (\underline{X}_{N+1} - \underline{M}_{N+1}) (\underline{X}_{N+1} - \underline{M}_{N+1})^T$$

$$= \frac{N}{N+1} \underline{J}_N + \frac{1}{N} (\underline{X}_{N+1} - \underline{M}_{N+1}) (\underline{X}_{N+1} - \underline{M}_{N+1})^T \quad \ldots\ldots\ldots \quad (B.5)$$

Posons :

$$\underline{Z}_{N+1} = (\underline{X}_{N+1} - \underline{M}_{N+1})$$

$$\lambda_1 = \frac{N}{N+1}$$

$$\lambda_2 = \frac{1}{N}$$

L'équation (B.5) devient :

$$\underline{J}_{N+1} = \lambda_1 \underline{J}_N + \lambda_2 \underline{Z} \underline{Z}^T \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (B.6)$$

Par le lemme d'inversion de Matrice [39], nous obtenons finalement l'inverse de $\underline{J}_{N+1}$ qui est donnée par :

$$\underline{J}_{N+1}^{-1} = \frac{1}{\lambda_1}\left(\underline{J}_N^{-1} - \frac{\underline{J}_N^{-1}\,\underline{Z}_{N+1}\,\underline{Z}_{N+1}\,\underline{J}_N^{-1}}{\frac{\lambda_1}{\lambda_2} + \underline{Z}_{N+1}^{T}\,\underline{J}_N\,\underline{Z}_{N+1}}\right)$$

ou

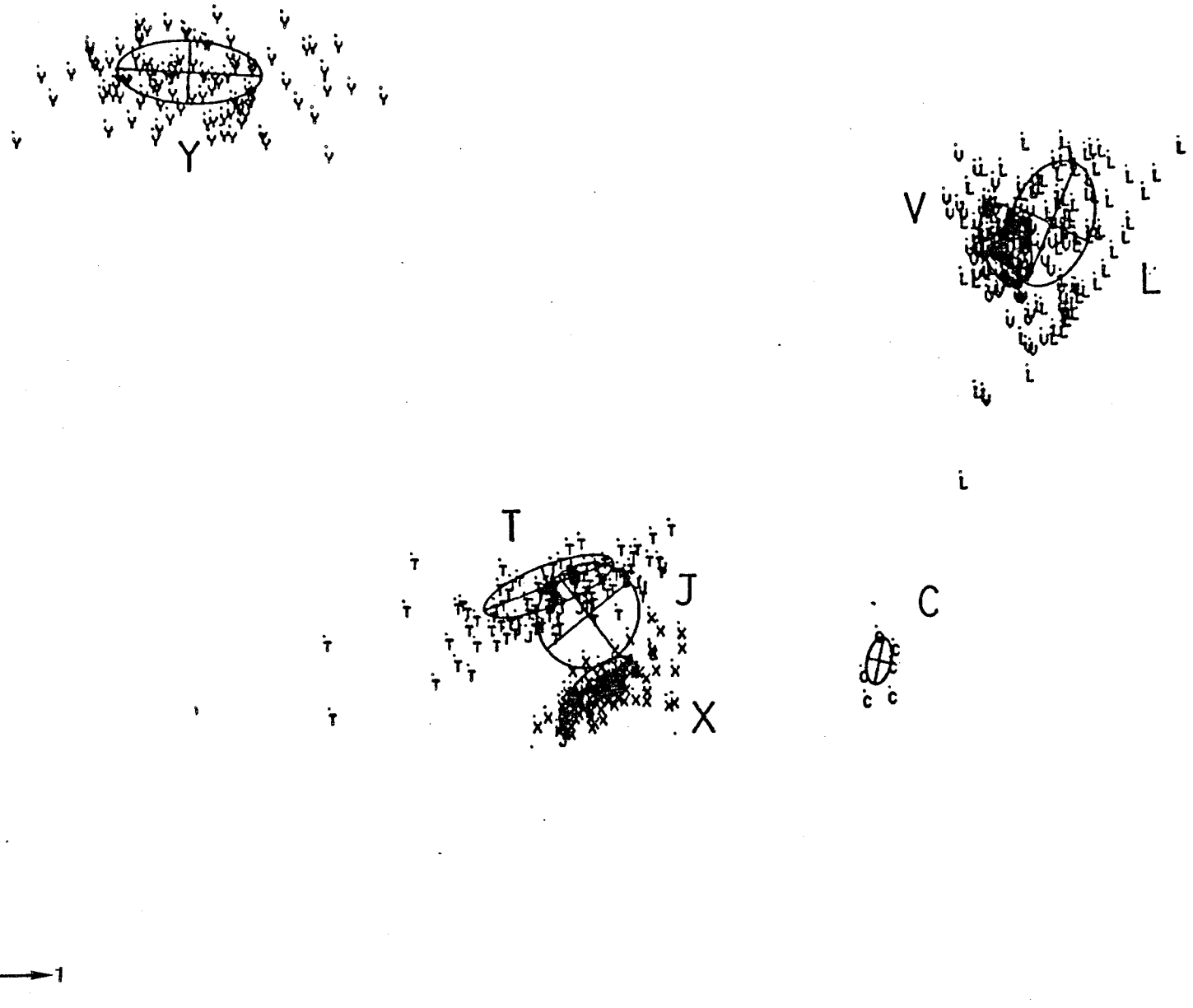
$$\underline{J}_{N+1}^{-1} = \frac{N+1}{N}\left(\underline{J}_N^{-1} - \frac{\underline{J}_N^{-1}\,\underline{Z}_{N+1}\,\underline{Z}_{N+1}^{T}\,\underline{J}_N^{-1}}{\frac{N^2}{N+1} + \underline{Z}_{N+1}^{T}\,\underline{J}_N\,\underline{Z}_{N+1}}\right) \quad \cdots\cdots \quad (B.7)$$

APPENDICE C-1 :

L'analyse en composantes principales

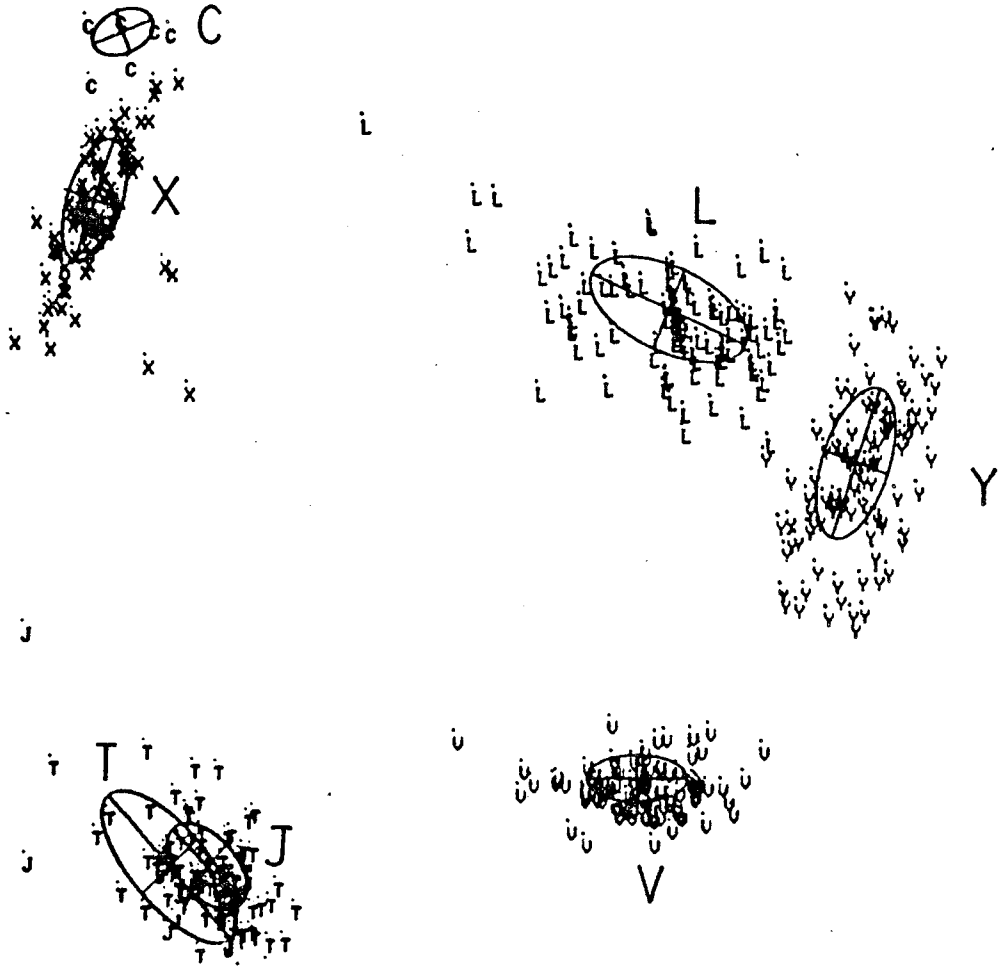
Lettre à 2 segments

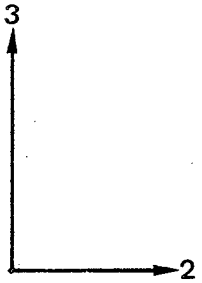
Plan de projection 1 et 2



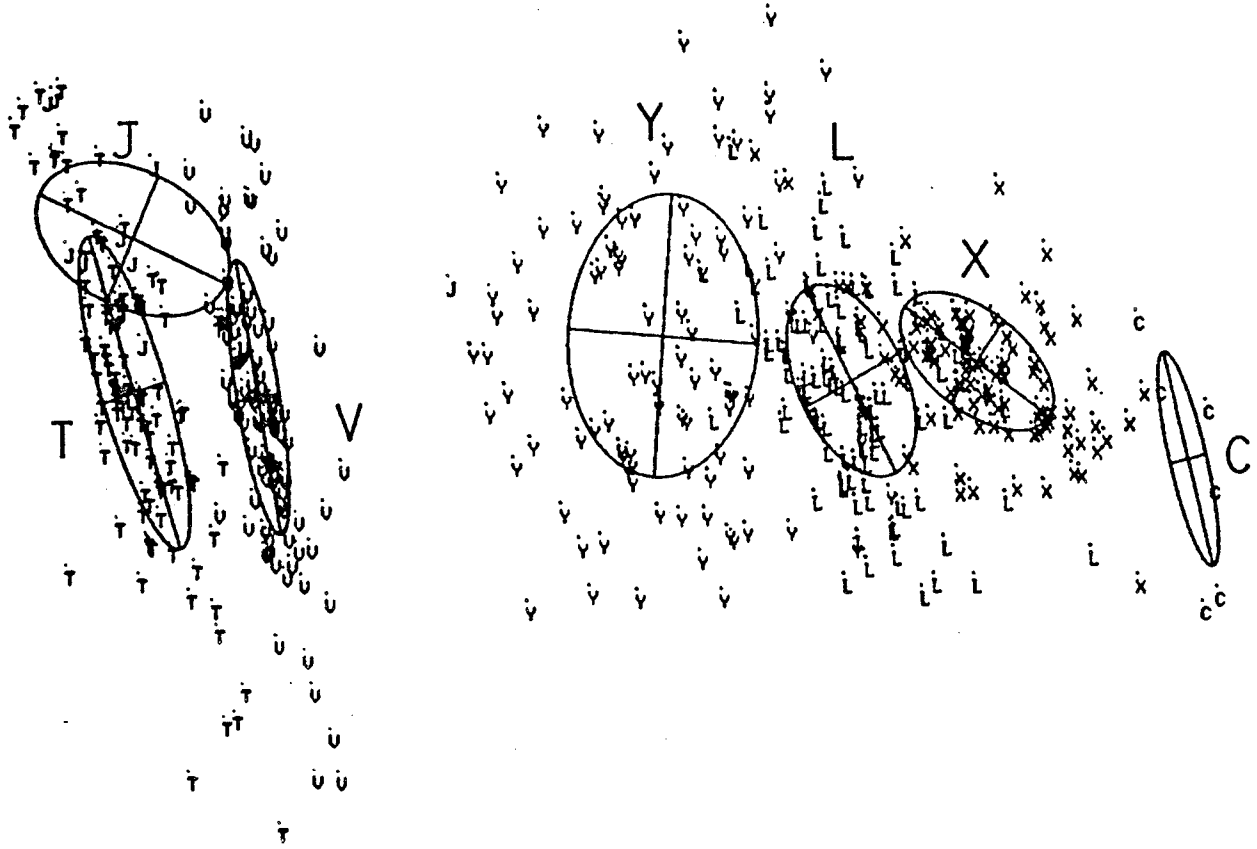
Appendice C-1 (suite)
Lettres à 2
segments

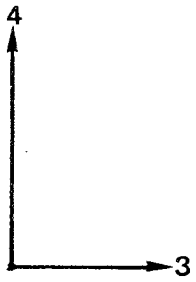
Plan de projection
2 et 3





Appendice C-1 (suite)
Lettres à 2 segments
Plan de projection
3 et 4



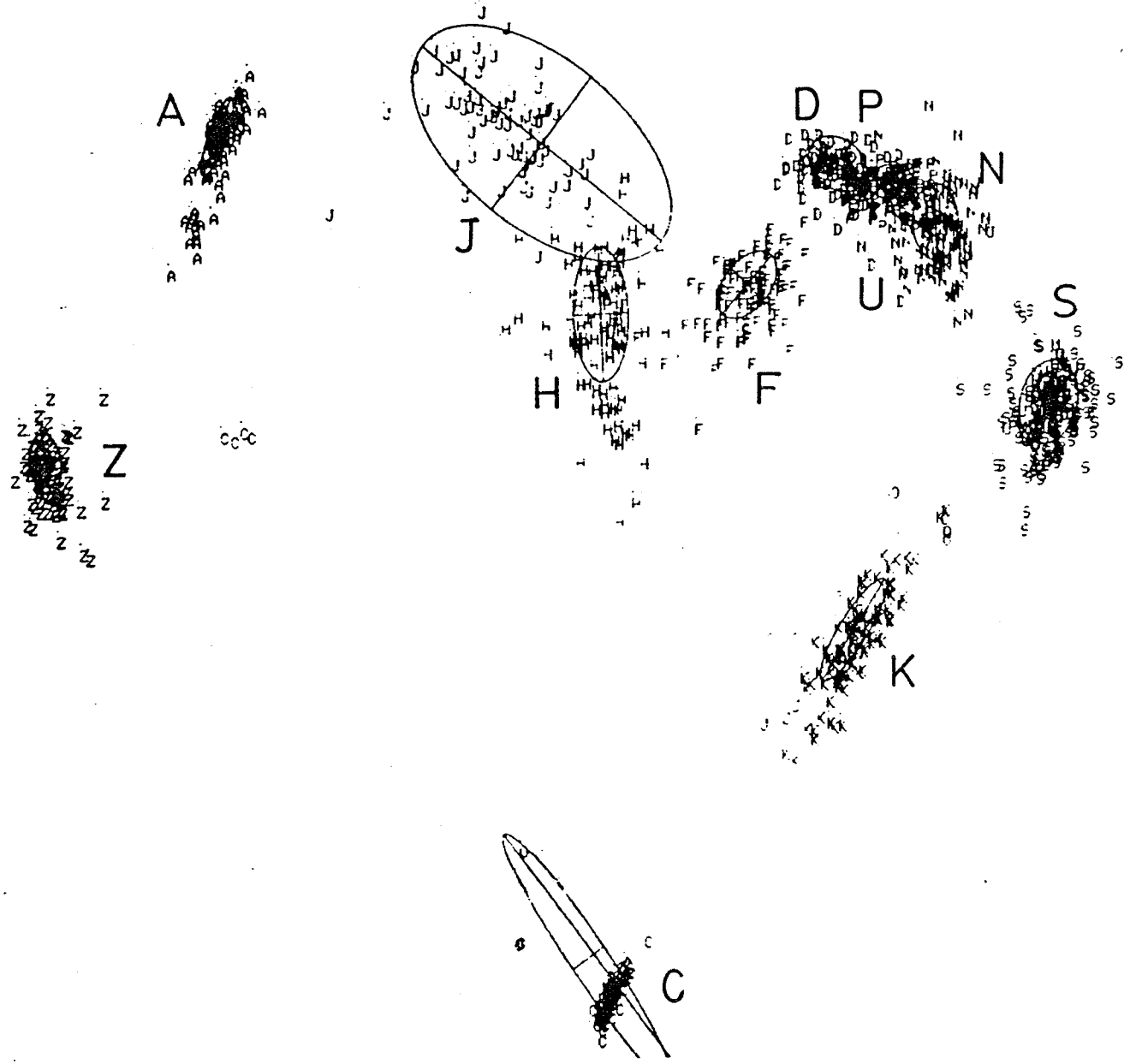


APPENDICE C-2 :

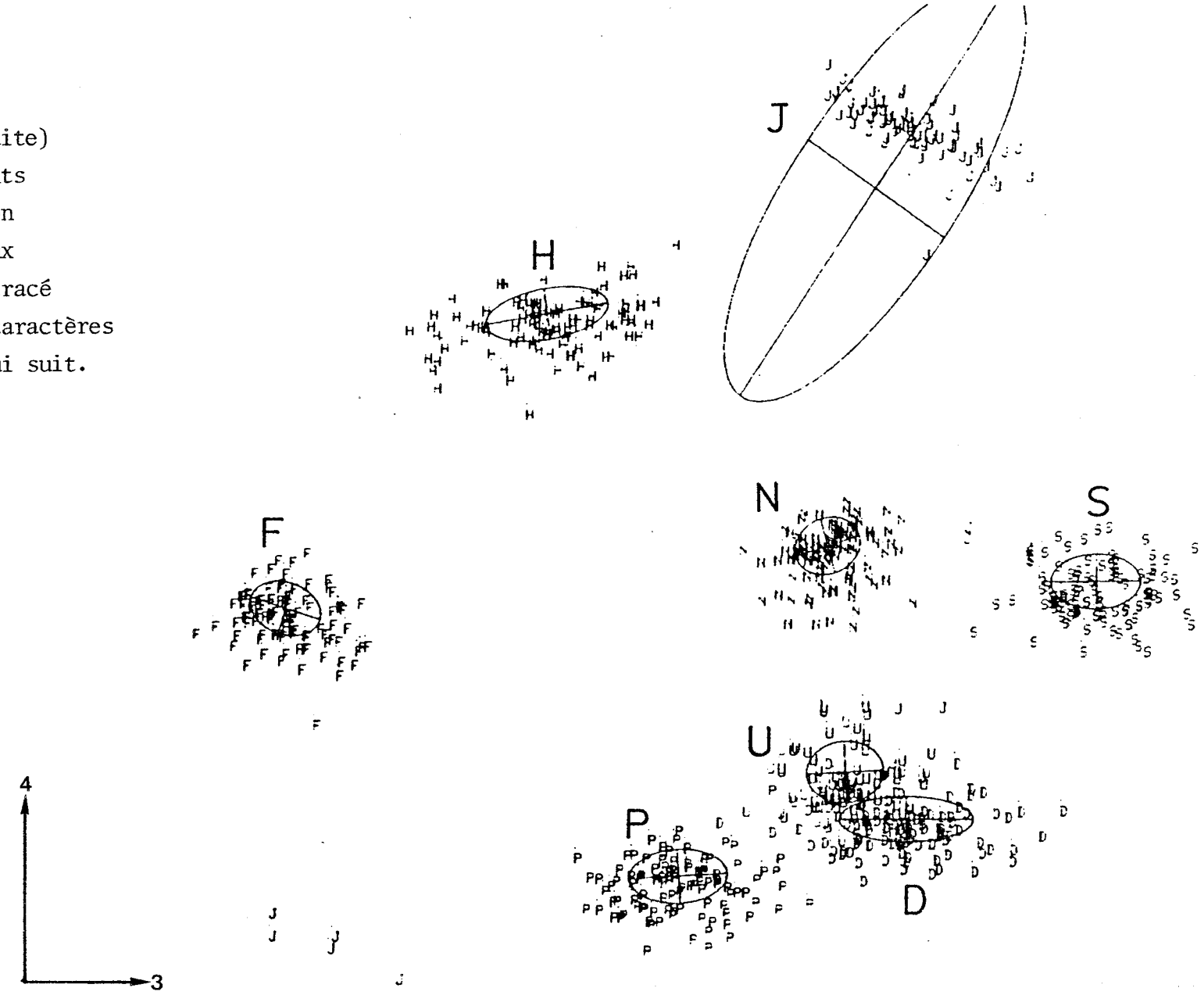
L'analyse en composantes principales

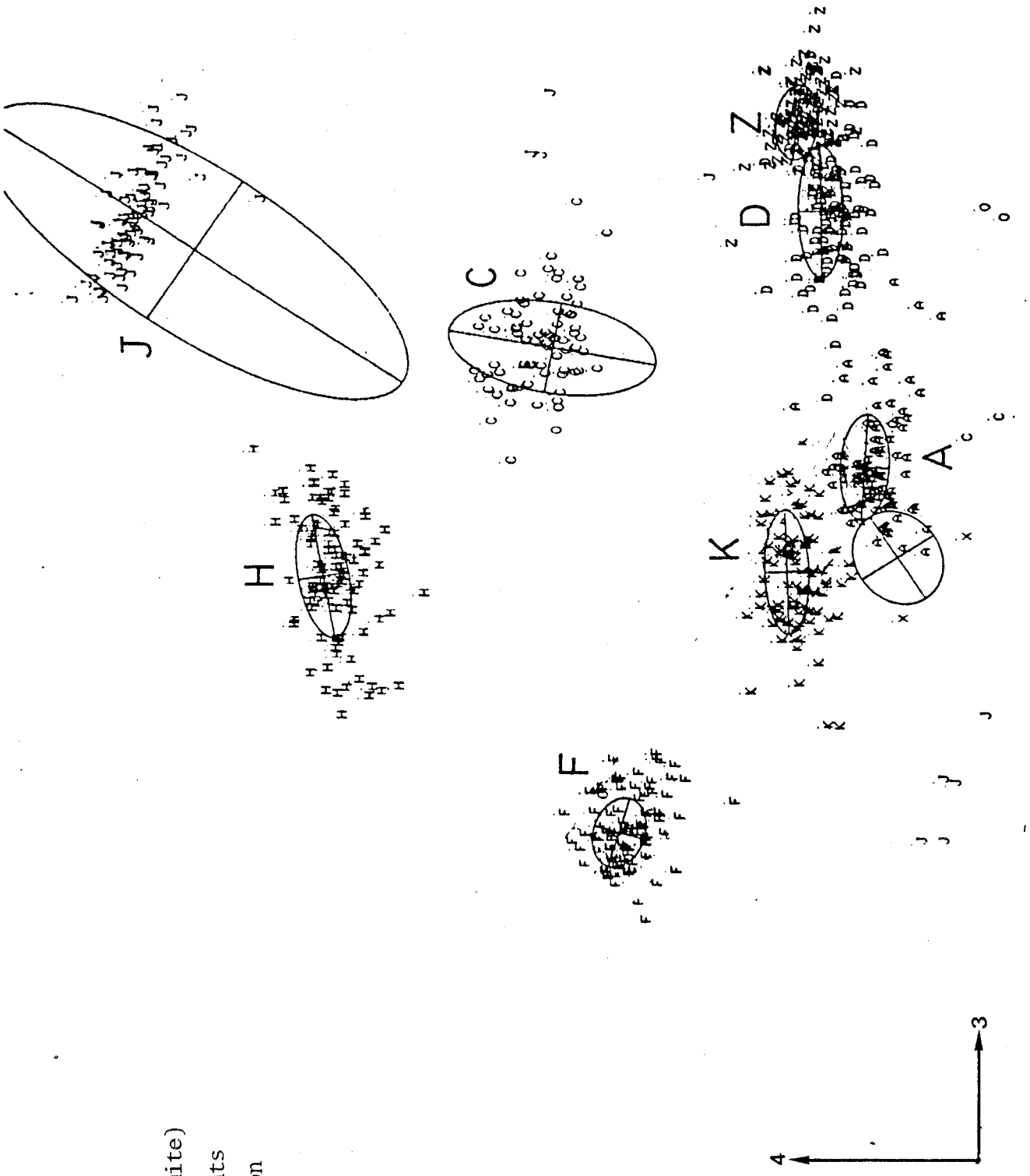
Lettre à 3 segments

Plan de projection 1 et 2



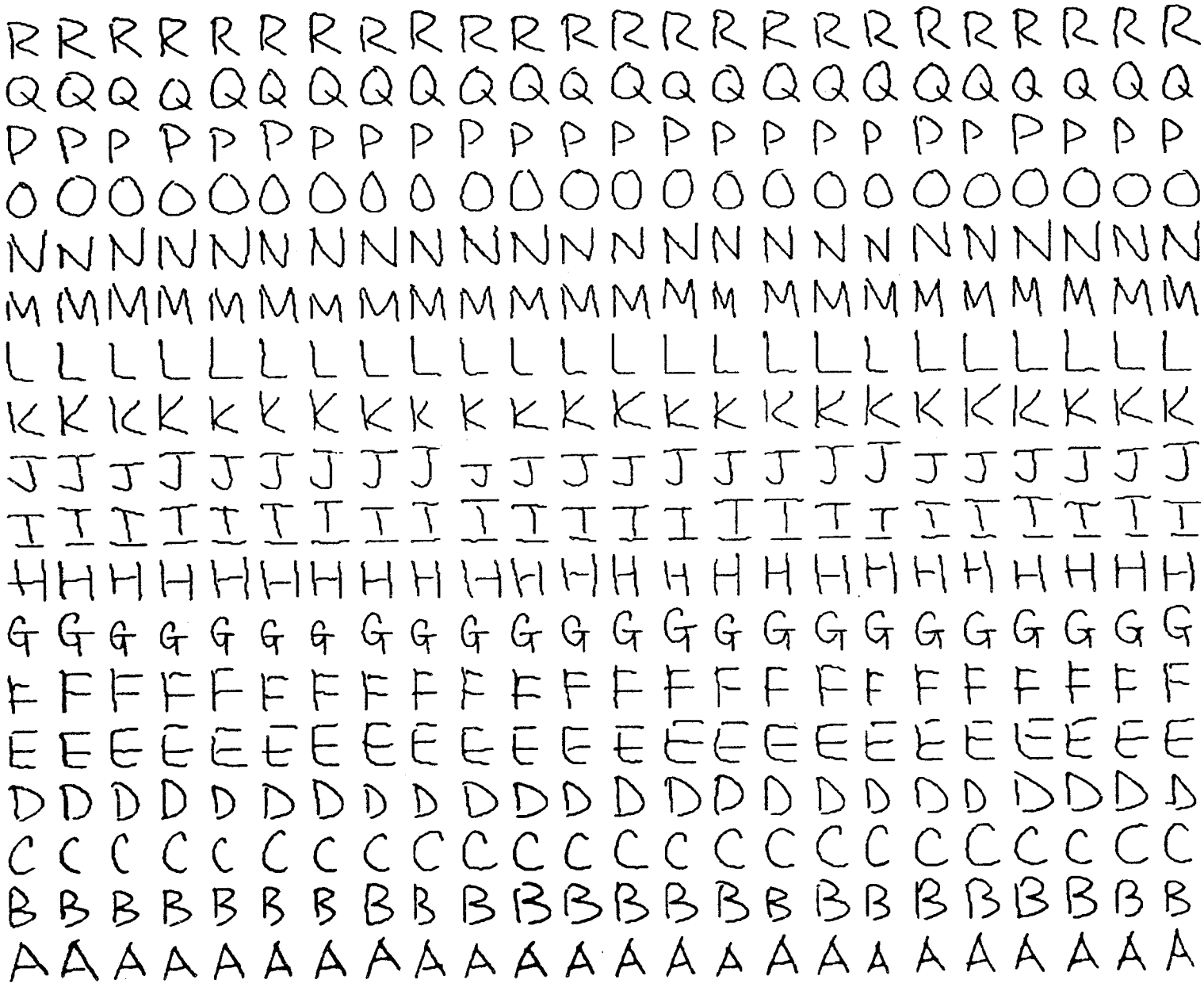
Appendice C-2 (suite)
Lettre à 3 segments
Plan de projection
3 et 4. Pour mieux
voir nous avons tracé
quelques autres caractères
dans la figure qui suit.



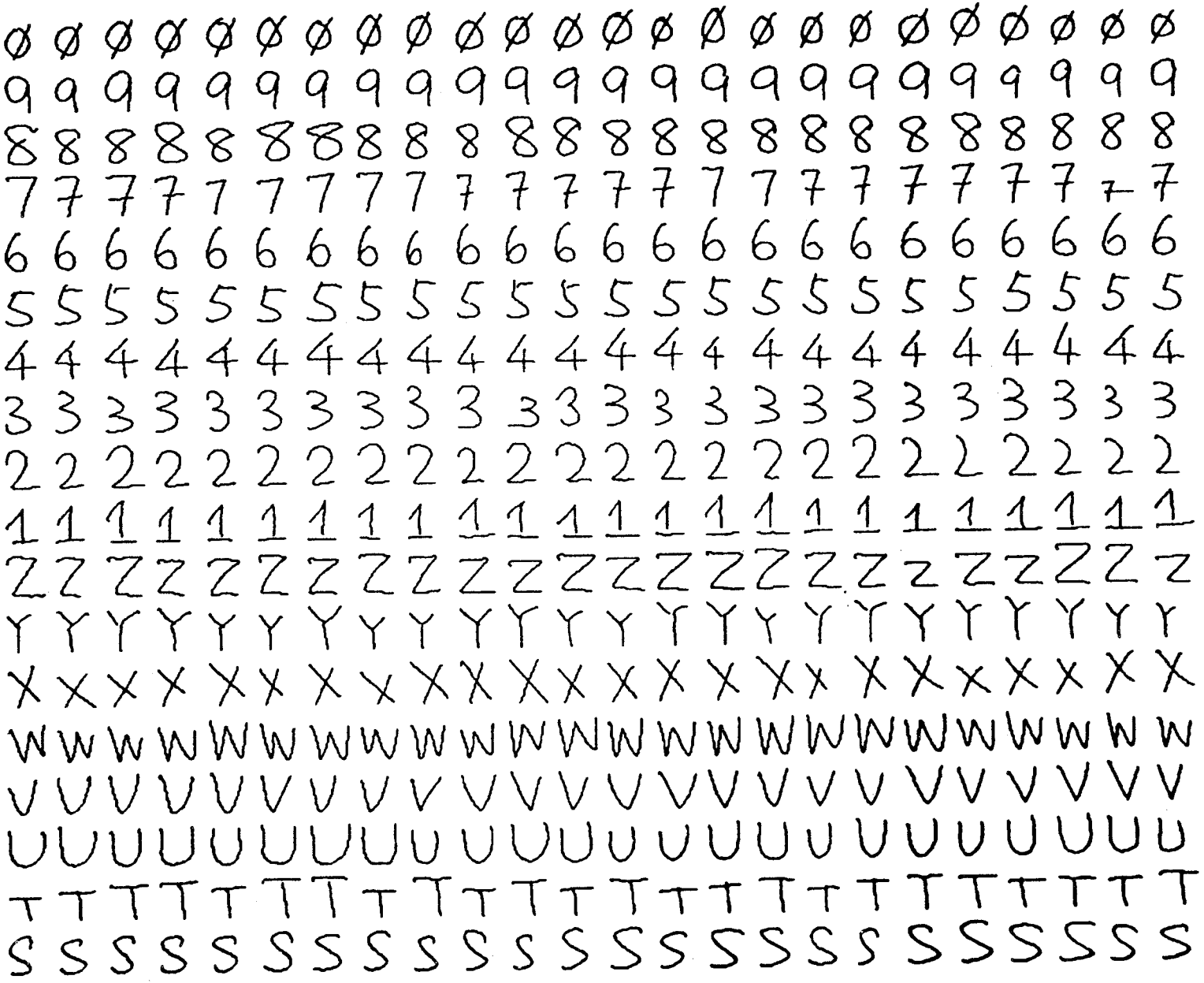


Appendice C-2 (suite)
Lettre à 3 segments
Plan de projection
3 et 4.

Appendice D
Les caractères
de base de données
monoscripteur :
Base-Lo-Te



Appendice D
(suite)
Les caractères de
base de données
monoscripteur
Base-Lo-Te



APPENDICE E *Les Automates finis*

DEFINITION :

S - L'état initial

B à G - les états intermédiaires

L, P, N, B, D, C et / - les caractéristiques globales (produit de transition)

d, b^-, b^+, c^+, c^-, n, f - les primitives

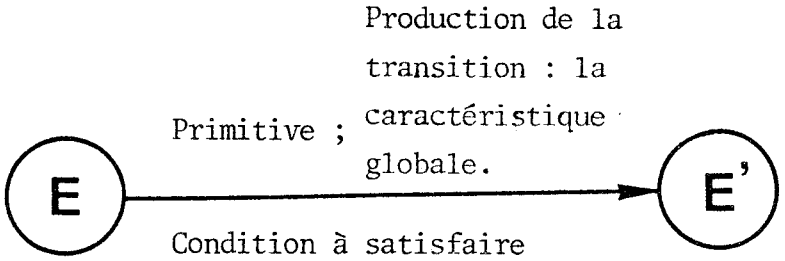
Θ_1 - somme de tous les $\Delta\Theta_s$ des primitives précédents qui ne font pas parties d'une caractéristique globale déjà extraite.

Θ_2 - $\Delta\Theta$ du primitive courant

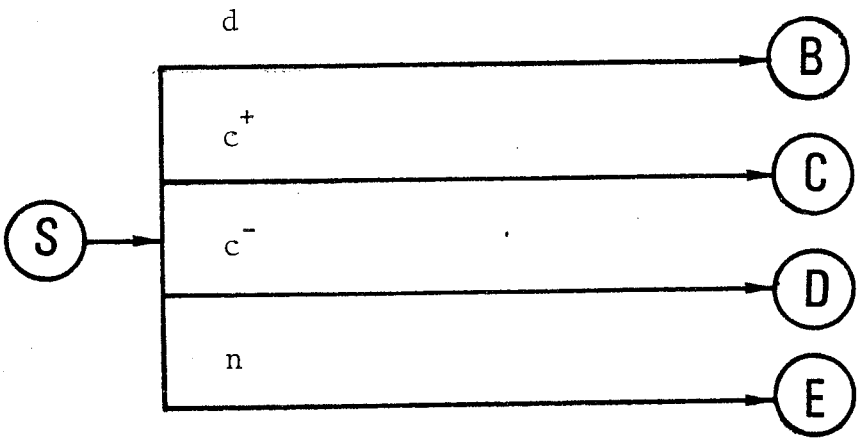
Θ_3 - $\Delta\Theta$ du primitive précédent

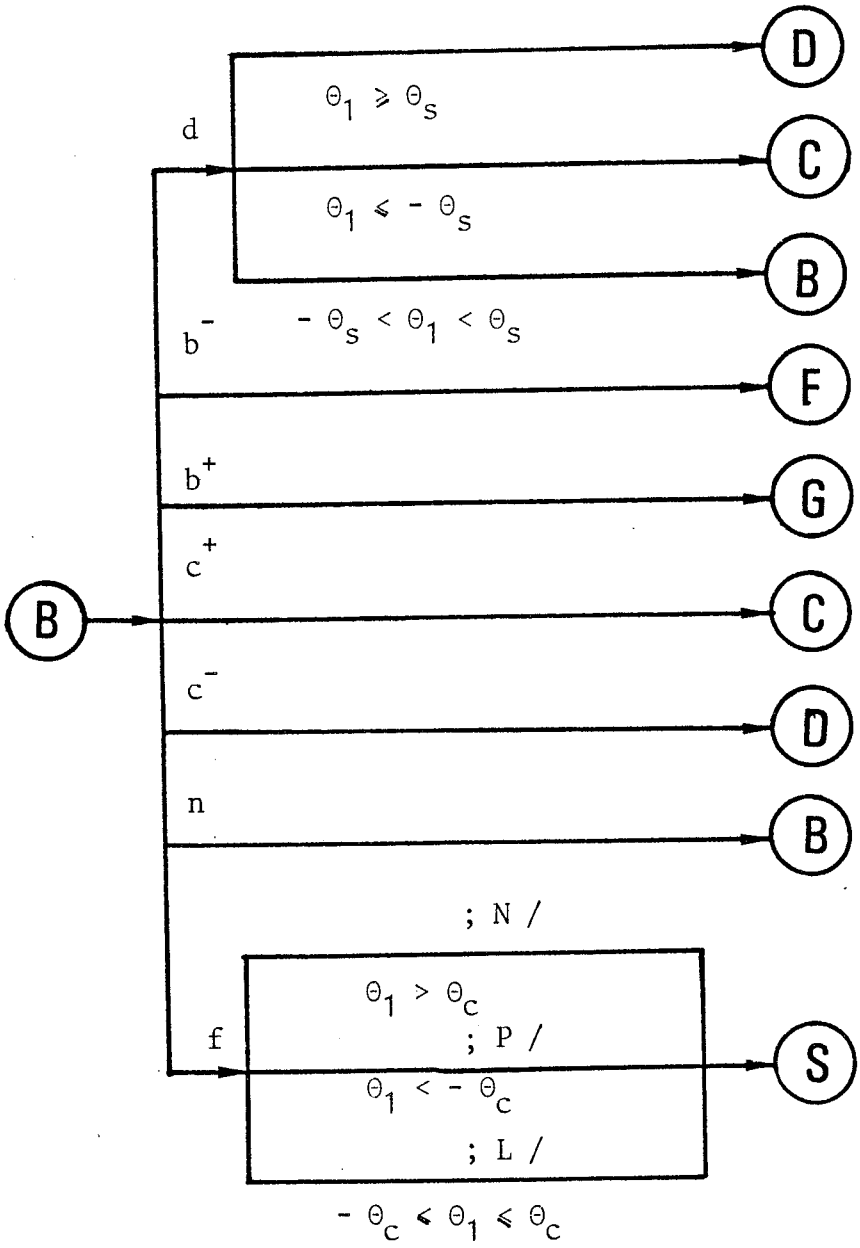
Θ_b, Θ_c, Θ_s - les seuils de break, de courbe et de droite.

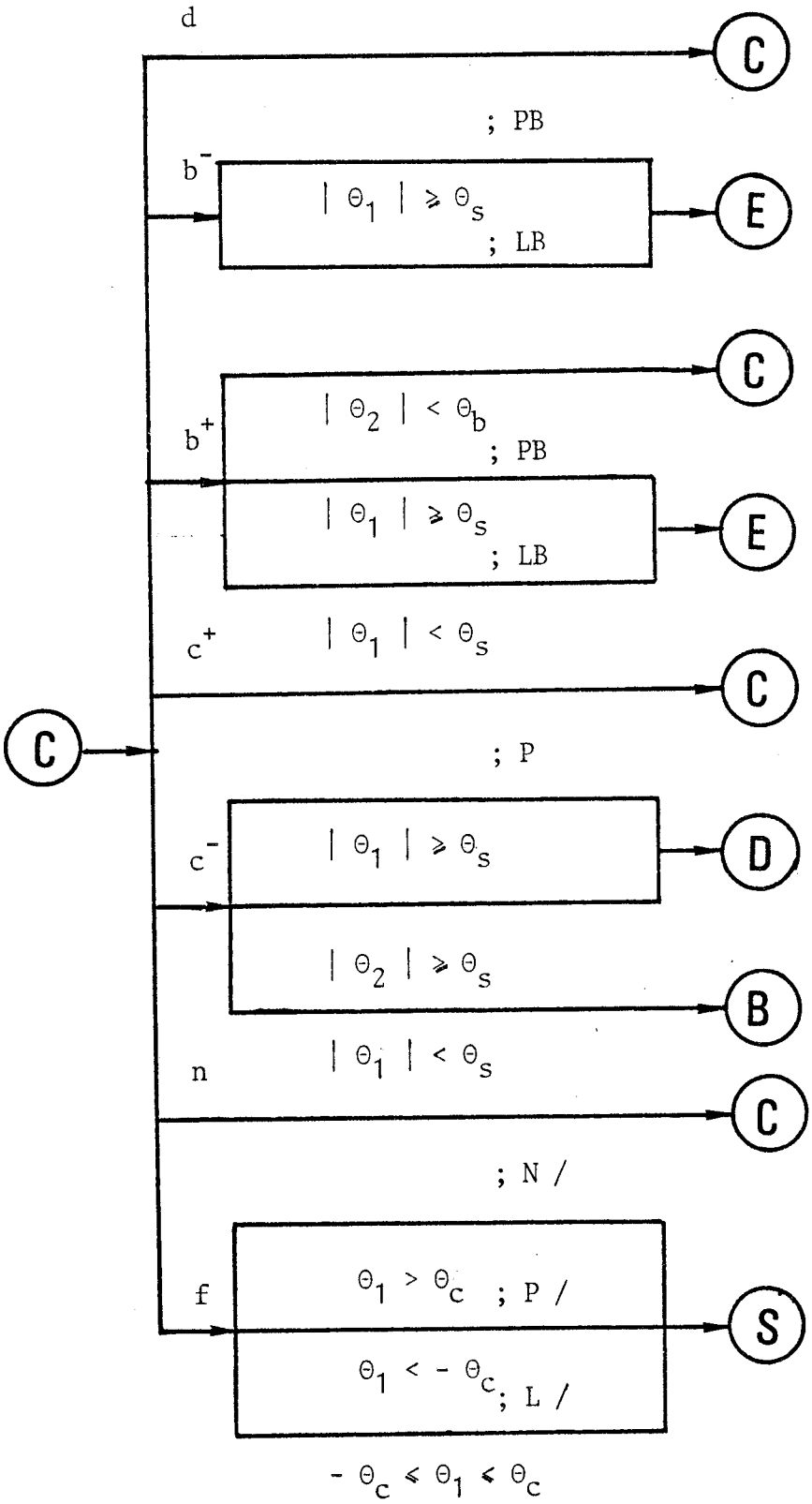
CONVENTION :

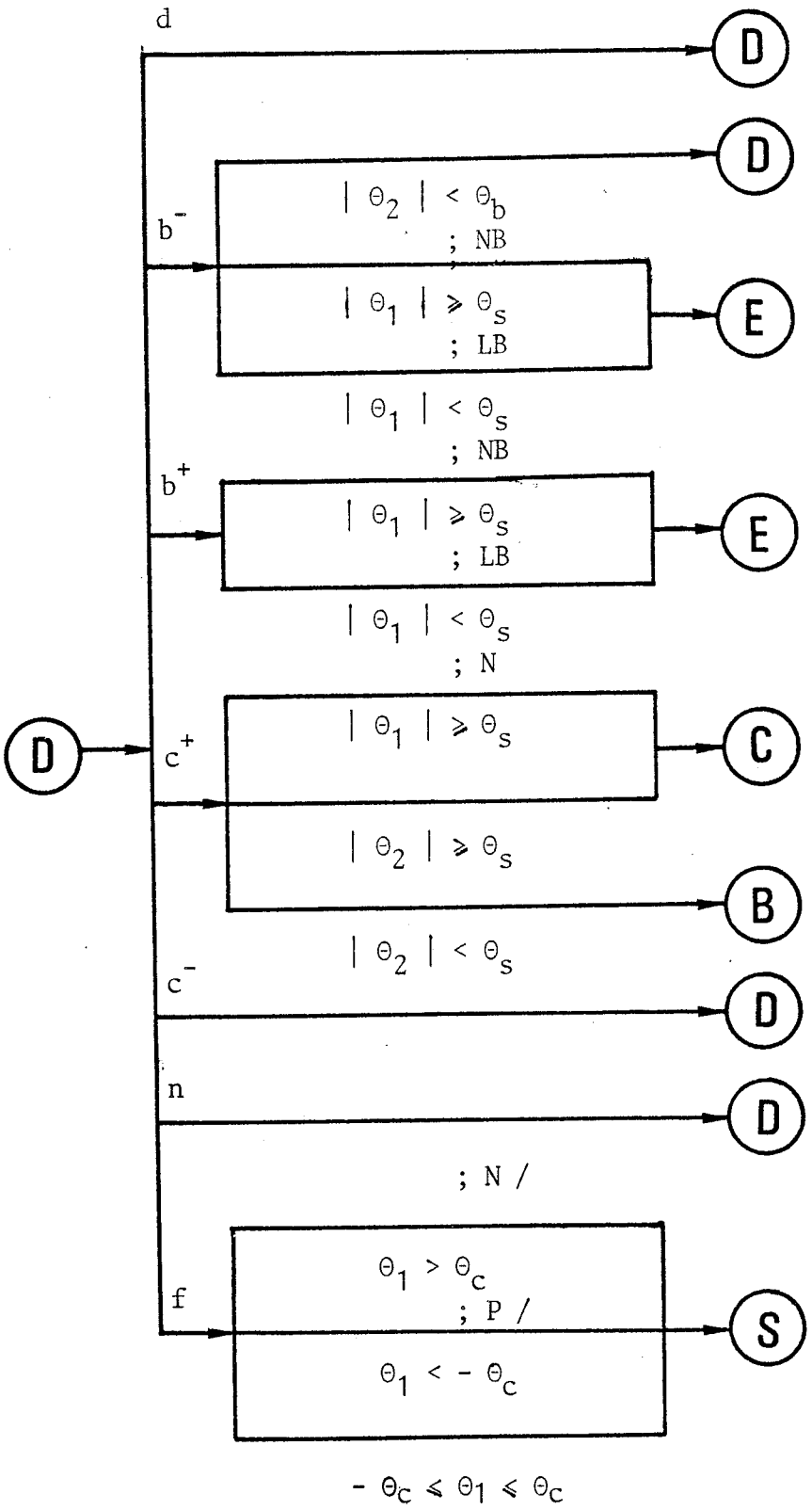


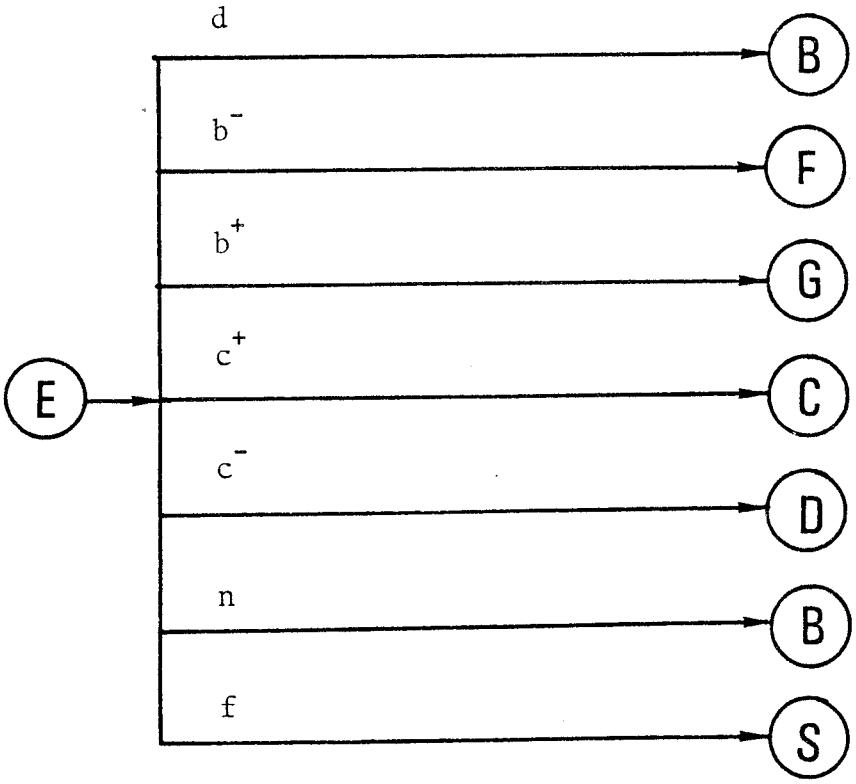
l'état précédent l'état suivant

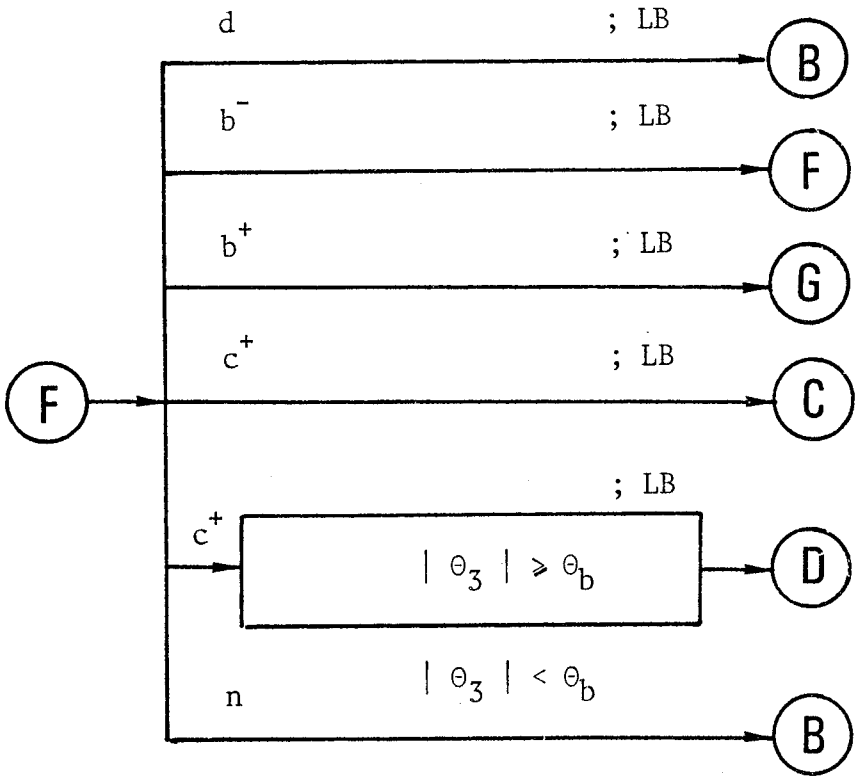

S
d
B
c+
C
c−
D
n
E

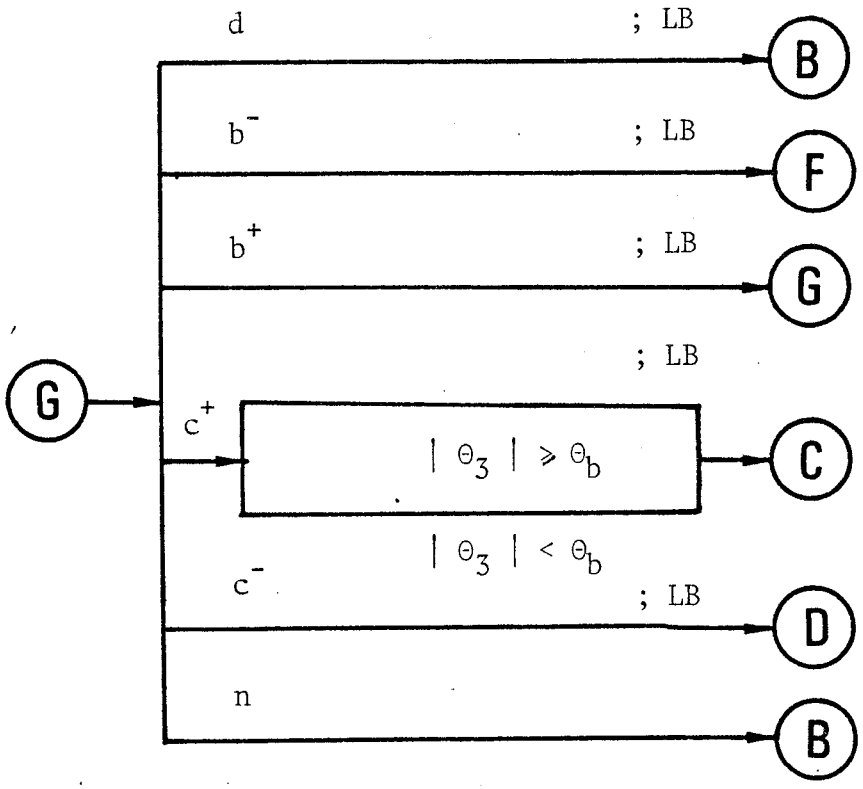

B
d
D
Θ1 ≥ Θs
C
Θ1 ≤ − Θs
B
b−
− Θs < Θ1 < Θs
F
b+
G
c+
C
c−
D
n
B
; N /
f
Θ1 > Θc
; P /
S
Θ1 < − Θc
; L /
− Θc ≤ Θ1 ≤ Θc


C
d
; PB
b⁻
| θ1 | ≥ θs
; LB
E
C
b⁺
| θ2 | < θb
; PB
| θ1 | ≥ θs
; LB
E
c⁺
| θ1 | < θs
C
C
; P
c⁻
| θ1 | ≥ θs
D
| θ2 | ≥ θs
B
n
| θ1 | < θs
C
; N /
f
θ1 > θc ; P /
θ1 < - θc ; L /
S
- θc ≤ θ1 < θc


d
D
D
$|\Theta_2| < \Theta_b$
; NB
b^-
$|\Theta_1| \geqslant \Theta_s$
; LB
E
$|\Theta_1| < \Theta_s$
; NB
b^+
$|\Theta_1| \geqslant \Theta_s$
; LB
E
$|\Theta_1| < \Theta_s$
; N
D
c^+
$|\Theta_1| \geqslant \Theta_s$
C
$|\Theta_2| \geqslant \Theta_s$
B
$|\Theta_2| < \Theta_s$
c^-
D
n
D
; N /
f
$\Theta_1 > \Theta_c$
; P /
S
$\Theta_1 < -\Theta_c$
$-\Theta_c \leqslant \Theta_1 \leqslant \Theta_c$


E
d
B
b⁻
F
b⁺
G
c⁺
C
c⁻
D
n
B
f
S


F
d
; LB
B
b⁻
; LB
F
b⁺
; LB
G
c⁺
; LB
C
; LB
c⁺
| θ3 | ≥ θb
D
| θ3 | < θb
n
B

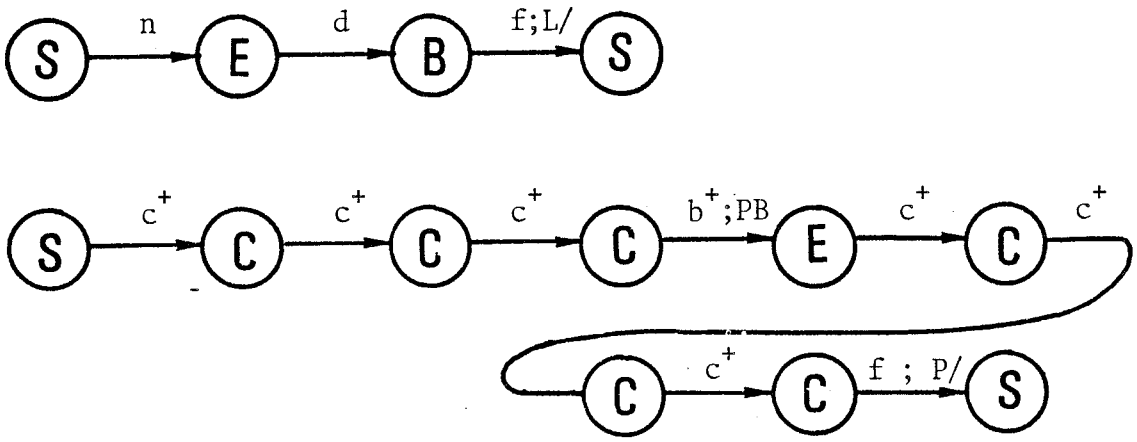


Exemple

Prenons un exemple précédent, le caractère de la figure 4.2 a comme primitives :

n d f c^+ c^+ c^+ b^+ c^+ c^+ c^+ f

Le diagramme de transition est le suivant :



La description globale est donnée comme L/PBP/ d'après cet automate.

Appendice F
Les caractères de base de données multi-scripteur Base-Mu-Te-Le et Base-Mu-Te-Ch

Appendice F
(suite)
Les caractères de
base de données
multiscripteur
Base-Mu-Te-Le
et Base-Mu-Te-Ch

APPENDICE G :

An ON-LINE Procedure for Recognition of Handwritten Alpha-Numeric Characters

W.W. LOY and I.D. LANDAU

5th International Conference on Pattern Recognition, pp.712-714, Dec. 1980, Miami USA.

Note Interne LAG 80

APPENDICE H :

Reconnaissance EN LIGNE de caractères Alphanumériques Manuscrits Utilisant des Approximations Polygonales Lineaires et Curvilignes.

W.W. LOY et I.D. LANDAU

3ème Congrès Reconnaissance des Formes et Intelligence Artificielle, pp.133-143, Sept. 1981, Nancy FRANCE

Note Interne LAG 81

BIBLIOGRAPHIE.

B I B L I O G R A P H I E.

Les références bibliographiques sont groupées sous trois rubriques :

A - *Articles de congrès ou journaux.*

B - *Livres.*

C - *Thèses et Rapports techniques.*

Nous suivons l'ordre alphabétique du nom du premier auteur de chaque références dans chacune de ces trois rubriques.

° I °

A - ARTICLES DE CONGRES OU JOURNAUX.

[1] AHLGREN R. C., RYAN H. F. AND SWONGER C. W.
A character recognition application of an iterative procedure for feature selection.
IEEE Trans. Comput., (short notes), Vol. C - 20, pp. 1067 - 1075, Sept. 1971.

[2] ALI F. AND PAVLIDIS T.
Syntactic recognition of handwritten numerals.
IEEETrans. Syst., Man, and Cybern., Vol. SMC - 7, pp. 537 - 541, July 1977.

[3] ARAKAWA H., MASUDA I. AND ODAKA K.
On line recognition of handwritten characters.
The Trans. of IECE of Japan, Vol. E - 59, N° 11, pp. 809 - 816, Nov. 1976.

[4] BAKIS R., HERBST N. M. AND NAGY G.
An experimental study of machine recognition of hand-printed numerals.
IEEETrans. Syst., Sci., and Cybern., Vol. SSC - 4, pp. 119 - 132, July 1968.

[5] BERTHOD M. AND MAROY J. P.
Learning in syntactic recognition of symbols drawn on a graphic tablet.
Computer Graphics and Image Processing 9, pp. 166 - 182, 1979.

[6] CHINNUSWAMY P.
Recognition of handprinted Tamil characters.
Pattern Recognition, Vol. 12, pp. 141 - 152, 1980.

[7] DEVOE D. B.
Alternatives to handprinting in the manual entry of data
IEEE Trans. on Human Factors in Electronics, Vol. 8, N° 1, pp. 21 - 32, March 1967.

[8] DRISCOLL R. J. AND GORDON J. A.
Real-time reading of handwritten symbols and applications.
Digital Processing of Signals in Communications, pp. 293 - 302, 6 - 9 Sept. 1977. LONDON.

[9] EHRICH R. W. AND KOEHLER K. J.
Experiments in the Contextual Recognition of Cursive Script.
IEEE Trans. Comput., Vol. C - 24, N° 2, Feb. 1975.

[10] FREEMAN H.
On the encoding of arbitary geometric configurations.
IRE Trans. Electronic Computers, Vol. EC - 10, pp. 260 - 268, June 1961.

[11] FU K. S. AND BOOTH T. L.
Grammatical Inference : Introduction and Survey Part I and Part II.
IEEE Trans. Syst., Man, and Cybern.
Vol. SMC - 5, N° 1, pp. 95 - 111, Jan. 1975 (Part I).
Vol. SMC - 5, N° 4, pp. 409 - 423, July 1975 (Part II).

[12] GAILLAT G.
Nouveaux regards sur la reconnaissance des caractères.
Revue Technique THOMSON-CSF, Vol. 9, pp. 529 - 545, Sept. 1977.

[13] GRANLUND G. H.
Fourier preprocessing for hand print character recognition.
IEEE Trans. Comput., (short notes), Vol. C - 21, pp. 195 - 201, Feb. 1972.

[14] GRONER G. F.
Real-time recognition of handprinted text.
AFIPS, Proc. Fall Joint Computer Conference, Vol. 29, pp. 591 - 601, Nov. 1966.

[15] HUSSAIN A. B. S., TOUSSAINT G. T. AND DONALDSON R. W.
Results obtained using a simple character recognition procedure on Munson's handprinted data.
IEEE Trans. Comput., (short notes), Vol. C - 21, pp. 201 - 205, Feb. 1972.

[16] KNOLL A. L.
Experiments with "Characteristic Loci" for recognition of handprinted characters.
IEEE Trans. Comput., (short notes), Vol. C - 18, pp. 366 - 372, Apr. 1969.

[17] LOY W. W. AND LANDAU I. D.
An on-line procedure for recognition of handwritten alphanumeric characters.
Proc. 5th International Conference on Pattern Recognition, pp. 712 -714, Dec. 1980, Miami, USA.

[18] LOY W. W. AND LANDAU I. D.
Reconnaissance en ligne de caractères alphanumériques manuscrits utilisant des approximations polygonales linéaires et curvilignes.
3ème Congrès AFCET "Reconnaissance des Formes et Intelligence Artificielle, Sept. 1981, NANCY - France.

[19] LOY W. W. AND LANDAU I. D.
On ligne Recognition of Unconstrained Alphanumeric Characters.
Note Interne - LAG Juin 81.

[20] McKEE J. W. AND AGGARWAL J. K.
Computer recognition of partial views of curved objects.
IEEE Trans. Comput., Vol. C - 26, pp. 790 - 800, Aug. 1977.

[21] MERMELSTEIN P. AND EDEN M.
Experiments on Computer Recognition of Connected Hand written words.
Information and Control 7, pp. 255 - 270,, 1964.

[22] MILLER G. M.
On line recognition of hand-generated symbols.
AFIPS, Proc. Fall Joint Computer Conference, Vol. 32, pp. 399 - 412, Nov. 1969.

[23] NEISSER U. AND WEENE P.
A note on human recognition of handprinted characters.
Information and Control 3, pp. 191 - 196, 1960.

[24] NIEMANN H.
Classification of characters by man and by machine.
Pattern Recognition, Vol. 9, pp. 173 - 179, 1977.

[25] PAVLIDIS T. AND ALI F.
Computer recognition of handwritten numerals by polygonal approximations.
IEEE Trans. Comput., Vol. SMC - 5, pp. 610 - 614, Nov. 1975.

[26] PAVLIDIS T. AND HOROWITZ S. L.
Segmentation of plane curves.
IEEE Trans. Comput., Vol. C - 23, pp. 860 - 870, Aug. 1974.

[27] POWERS V. M.
Pen direction sequences in character recognition.
Pattern Recognition, Vol. 5, pp. 291 - 302, 1973.

[28] RAMER U.
An iterative procedure for the polygonal approximation of plane curves.
Computer Graphics and Image Processing 1, pp. 244 - 256, 1972.

[29] RAVAUD A. M.
Système de reconnaissance automatique des chiffres manuscrits utilisant la transformation de Walsh-Hadamard.
Automatisme, pp. 212 - 218, Juillet-Août 1978.

[30] SINHA R. M. K. AND MAHABALA H. N.
Machine recognition of Devanagari script.
IEEE Trans. Syst., Man and Cybern, (correspondance), Vol. SMC - 9, pp. 435 - 441, Aug. 1979.

[31] SKLANSKY J.
Image segmentation and feature extraction.
IEEE Trans. Syst., Man and Cybern, Vol. SMC - 8, pp. 237 - 247, Apr. 1978.

[32] SKLANSKY J. AND GONZALEZ V.
Fast polygonal approximation of digitized curves.
Pattern Recognition, Vol. 12, pp. 327 - 331, 1980.

[33] SPINRAD R. J.
Machine recognition of hand printing.
Information and Control 8, pp. 124 - 142, 1965.

[34] SUEN C. Y., BERTHOD M. AND MORI S.
Automatic Recognition of Handprinted Characters.
The State of the Art.
Proceeding of the IEEE, Vol. 68, N° 4, Apr. 1980.

[35] YAMAMOTO K. AND MORI S.
Recognition of handprinted characters by an outermost point method.
Pattern Recognition, Vol. 12, pp. 229 - 236, 1980.

B - *LIVRES.*

[36] DUDA R. O. AND HART P. E.
Pattern Classification and Scene Analysis.
New-York : Wiley, 1973.

[37] FU K. S. ED.
Syntactic Pattern Recognition Applications.
Springer-Verlag Berlin Heidelberg 1977.
Springer Series in Electrophysics - Vol. 14.

[38] FU K. S.
Syntactic methods in pattern recognition.
New-York : Academic, Mathematics in science and engineering
Vol. 112.

[39] LANDAU I. D.
Adaptive Control, The Model Reference Approach.
New-York : Dekker, 1979.
Control and system theory - Vol. 8.

[40] MEISEL W. S.
Computer-Oriented Approaches to Pattern Recognition.
New-York : Academic Press, 1972.
Mathematics in science and engineering - Vol. 83.

[41] NEWMAN W. M. AND SPROULL R. F.
Principles of Interactives Computer Graphics.
Mcgraw Hill Computer Science Series, 1973.

[42] PAVLIDIS T.
Structural Pattern Recognition.
Springer-Verlag Berlin Heidelberg 1977.
Springer Series in Electrophysics - Vol. 1.

C - *THESES ET RAPPORTS TECHNIQUES.*

[43] ADAM G. ET GENTRIC A.
Extraction de Caractéristiques par Analyse en Composantes Principales.
Projet 3ème Année, ENSIMAG et LAG, 1980.

[44] BELAID A.
Reconnaissance Structurelle de Caractères Manuscrits et de Formules Mathématiques.
Thèse de Doctorat de 3e Cycle, Université de Nancy I, Oct. 1979.

[45] BOUVIER G.
Système d'Acquisition et de Reconnaissance de Caractères Alphanumériques.
Thèse de Doctorat de 3e Cycle. Institut National Polytechnique de Grenoble, Mars 1977.

[46] CABRERA S.
Système de Reconnaissance en Ligne de Caractères Manuscrits - Réalisation Micro-informatique.
Rapport DEA, ENSIEG - LAG, 1981.

[47] CHEHIKIAN A.
Conception et Réalisation d'un Automate de Reconnaissance de Caractères Alphanumériques.
Thèse d'Etat. USM-INP Grenoble, Juin 1977.

[48] HASSAN H. F.
Procédé de Reconnaissance Structurelle des Caractères Alphanumériques Multifontes Utilisant des Niveaux de Décision Multiples.
Thèse d'Etat. USM-INP Grenoble, Mai 1981.

[49] KINDER A.
Procédé de Reconnaissance Syntaxique des Caractères Alphanumériques Manuscrits.
Thèse de Docteur-Ingénieur, INP Grenoble, Janvier 1981.

[50] OPTION.
Tablette X - Y, Notice technique, Société Option, Meylan, France. Mai 1978.

[51] WEAVER A. C.
On-line character recognition.
University of Illinois, department of computer science Urbana, Illinois U.S.A.
Report No. UI UCDCS - R - 74 - 660, Aug. 1974.