PolyPublie
Polytechnique Montréal

POLYTECHNIQUE MONTRÉAL
UNIVERSITÉ D'INGÉNIERIE

Titre: Title:	Application d'un réseau de neurones ARTMAP à la reconnaissance des commandes gestuelles d'édition de documents braille
Auteur: Author:	Silvia Sabeva
Date:	1999
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Sabeva, S. (1999). Application d'un réseau de neurones ARTMAP à la reconnaissance des commandes gestuelles d'édition de documents braille [Master's thesis, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/8830/

Document en libre accès dans PolyPublie
Open Access document in PolyPublie

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/8830/
Directeurs de recherche: Advisors:	Jean-Jules Brault, & Réjean Plamondon
Programme: Program:	Unspecified

Ce fichier a été téléchargé à partir de PolyPublie, le dépôt institutionnel de Polytechnique Montréal
This file has been downloaded from PolyPublie, the institutional repository of Polytechnique Montréal

https://publications.polymtl.ca

UNIVERSITÉ DE MONTRÉAL

APPLICATION D'UN RÉSEAU DE NEURONES ARTMAP À LA RECONNAISSANCE DES COMMANDES GESTUELLES D'ÉDITION DE DOCUMENTS BRAILLE

SILVIA SABEVA
DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE ÉLECTRIQUE)
MARS 1999

© Silvia Sabeva, 1999.

National Library of Canada

Acquisitions and Bibliographic Services

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque nationale du Canada

Acquisitions et services bibliographiques

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

Our file Notre référence

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-42925-3

Canada

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

APPLICATION D'UN RÉSEAU DE NEURONES ARTMAP À LA RECONNAISSANCE DES COMMANDES GESTUELLES D'ÉDITION DE DOCUMENTS BRAILLE

présenté par : SABEVA Silvia

en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées

à été dûment accepté par le jury d'examen constitué de :

M. NERGUIZIAN Chahé, M.Ing., président

M. BRAULT Jean-Jules, Ph.D., membre et directeur de recherche

M. PLAMONDON Réjean, Ph.D., membre et codirecteur de recherche

M. PARIZEAU Marc, Ph.D., membre

REMERCIEMENTS

J'aimerais remercier tout d'abord mon directeur de maîtrise, le professeur Jean-Jules Brault, pour la confiance qu'il m'a accordée, pour ses judicieux conseils, son aide et la motivation qu'il a toujours su me transmettre tout au long de ce projet. Je voudrais de plus, exprimer ma reconnaissance à mon codirecteur, le professeur Réjean Plamondon, pour son optimisme et son dynamisme contagieux et pour l'aide financière qu'il m'a gentiment octroyée.

Je désire souligner à quel point j'ai pu apprécier l'enrichissante expertise apportée par M. Claude Richard, consultant de la compagnie Microsoft et Madame Chantale Perugino, consultante de la compagnie SoftZen, lors de la conception du cadre fonctionnel de ce projet.

Je tiens à remercier les membres du laboratoire Scribens qui m'ont accueillie dans leur groupe : Wacef Guerfali, pour sa bonne humeur et ses nombreux encouragements, Shahram Moïn pour son aide apportée lors de ma recherche bibliographique, et tous les autres dont Alessandro Zimmer, Octavian Urèche et Salim Djeziri pour leur aide ponctuel et la secrétaire Hélène Dallaire, pour son assistance lors de la rédaction et la mise en page de mon mémoire.

La réalisation de ce mémoire a été rendue possible grâce à l'appui financier du fonds FCAR, et de celui provenant des partenaires industriels suivants : CEDROM-SNI, Prosys-Tec Inc., Visuaide, Bibliothèque Jeann Cypihot, La Magnétothèque et l'Institut Nazareth et Louis Braille.

Je désire enfin remercier tout particulièrement mon époux, Ivo Tcholakov qui m'a apporté soutien, encouragements et réconfort tout au long de ces études de maîtrise.

Il me semble important de témoigner ma reconnaissance à M. Nerguizian - président du jury et M. Parizeau - membre du jury pour leur participation à ma soutenance.

RÉSUMÉ

Le travail décrit dans ce mémoire se situe dans le cadre d'un projet de reconnaissance de documents brailles pour l'édition et la mise en ligne de textes français sur le réseau Internet, projet développé conjointement par le laboratoire Scribens et divers intervenants extérieurs.

Nous voulons numériser des documents en braille pour assurer leur conservation, leur reproduction et leur diffusion sur le réseau Internet. Pour obtenir l'image d'une page braille, un scanner commercial est utilisé. Cette image représente l'entrée d'un module de reconnaissance optique de graphèmes brailles dont la sortie produit un texte ASCII correspondant au texte braille. Il est très probable qu'on retrouve des erreurs de reconnaissance, dues à plusieurs facteurs liés aux qualités des documents traités comme par exemple : l'âge du document, l'usure, la technique d'embossage utilisée et la qualité du support. L'intervention d'un humain est donc indispensable pour rectifier ces erreurs. Il faut donc avoir recours aux outils nécessaires à l'édition du texte brut, issu du reconnaisseur des caractères brailles.

Mentionnons l'importance du projet comme une façon de promouvoir la culture francophone car l'apparition des réseaux de communication globaux comme Internet met l'ensemble des cultures devant le défi d'assurer leur visibilité et leur survie dans ce nouvel environnement de stockage et de diffusion.

L'ordinateur utilisé dans notre projet est une ardoise électronique, appelée bloc-notes électronique ou papier électronique, utilisant un crayon comme seul dispositif d'entrée. Ce choix est justifié par le fait que l'utilisation d'un crayon devrait rendre l'interaction avec l'ordinateur naturelle, intuitive et rapide. La particularité de l'éditeur que nous proposons est qu'il fonctionne à l'aide de commandes gestuelles plutôt qu'avec des commandes entrées au clavier ou avec une souris.

La création d'un éditeur guidé par gestes nécessite la création d'un reconnaisseur des commandes gestuelles dont la tâche est d'associer une étiquette à une forme inconnue. Le choix d'un algorithme et son implantation est l'objectif de recherche principal du travail décrit ici.

Compte tenu des caractéristiques techniques limitatives, en termes d'espace mémoire et de vitesse de traitement de ces systèmes portables, l'éditeur de texte doit être compact, rapide d'exécution et surtout dans le contexte d'un éditeur qui se veut ergonomique, adaptable à l'usager. Nous devons donc envisager faire certains compromis lors du design de notre algorithme de reconnaissance de gestes.

Il existe plusieurs méthodes pour la classification des formes. On peut les diviser en deux groupes : les méthodes d'inspiration symbolique et les méthodes dites connexionnistes. Nous avons opté pour une approche connexionniste qui est particulièrement bien adaptée à la problématique d'association "forme – action".

Parmi les réseaux classificateurs supervisés qui existent, nous avons retenu le réseau ARTMAP car d'une part il s'adapte très rapidement et d'autre part, il permet d'ajouter relativement aisément de nouvelles commandes, avec laquelle l'usager serait éventuellement plus à l'aise que celles proposées. Dans le contexte d'un éditeur de texte basé sur des commandes gestuelles cette possibilité représente définitivement un atout important.

L'architecture de ARTMAP comporte deux modules ART1. Le réseau ART1 est dédié à la classification des images binaires. Nous l'avons adapté pour classer des gestes qui sont des signaux spatio-temporels. Il fallait donc trouver un prétraitement approprié. On a conçu un code dit isométrique par lequel on peut exploiter la simplicité du fonctionnement du réseau ART1. On montre que ce code est le seul qui permet d'obtenir

un comportement cohérent en fonction du paramètre de généralisation du réseau ART1. La définition du codage isométrique a permis la création d'un reconnaisseur de gestes rapide, fiable et adaptable qui est basé sur le réseau ARTMAP. Ce reconnaisseur est le noyau de l'éditeur gestuel adaptable.

La création de l'éditeur gestuel fonctionnel nécessite l'intégration de trois aspects soit la reconnaissance de commandes gestuelles, l'identification du contexte et l'exécution. Pour créer l'éditeur gestuel nous avons utilisé Word'97 de Microsoft et nous l'avons adapté aux nos besoins.

Une vérification expérimentale a déterminé l'impact des paramètres du reconnaisseur gestuel sur ses performances. Le reconnaisseur gestuel a deux paramètres : la longueur du signal d'entrée du réseau ARTMAP qui est déterminée par le nombre des segments (N) utilisés pour représenter les formes et les prototypes par le réseau ARTMAP, et le critère de vigilance (Ro) du réseau ARTMAP.

Pour tester nos algorithmes de reconnaissance, un ensemble de commandes gestuelles a été défini. Une recherche bibliographique dans la littérature scientifique et commerciale sur les commandes gestuelles existantes a été effectuée. Les conclusions qui en découlent se résument ainsi : il ne semble pas exister une définition précise sur les formes de gestes comparativement à celles existantes pour les lettres et les chiffes ; tout le monde s'entend pour dire que les gestes doivent être des formes simples et faciles à écrire.

On a créé un reconnaisseur de cet ensemble de formes et on a implanté les fonctionnalités (actions) pour en faire un éditeur gestuel fonctionnel. L'éditeur est adaptable puisque l'utilisateur a la possibilité d'ajouter des gestes à l'ensemble initial de gestes. L'ajout de nouveaux gestes s'effectue avec facilité, puisque la seule chose qui est alors exigée de l'usager, est qu'il dessine les nouveaux gestes.

Les tests d'évaluation des performances du reconnaisseur gestuel ont été effectués sur une banque de gestes, obtenus de quatorze scripteurs chacun fournissant trois fois quatorze formes, soit 588 formes. Le taux de reconnaissance des commandes gestuelles est 99.5%. La vérification expérimentale a permis de choisir le prétraitement par lequel on obtient les meilleurs résultats. Les analyses effectuées démontrent la robustesse de notre reconnaisseur de commandes gestuelles et sa tolérance aux variations des paramètres Ro et N.

ABSTRACT

The work described in this master thesis is part of a bigger project, which aims at the digitizing of Braille documents and publishing them on the Internet. The project was developed in the Scribens laboratory with the help of different external sponsors.

Our project aims at the numerisation of Braille documents in order to insure their conservation, reproduction and diffusion on the Internet. A commercial scanner is used to obtain an image of a Braille page. This page then becomes input for the Braille optical characters recognition module, which outputs the corresponding ASCII text. Taking into account the documents' age, their wearing, the obsolete printing technique and the quality of the support, it is realistic to assume that the resulting text will not be error prone. As a consequence tools must be used to edit the text produced by the Braille characters recognition module.

We should emphasize the importance of this project in terms of promoting the Francophone culture. The emergence of global communication nets creates a challenge for every culture to ensure its visibility and survival in this new environment for information stoking and diffusion - Internet.

The computer used in our project is a Windows-based pen tablet (Stylistic 1000). This choice accounts for the need of natural, intuitive and fast human-computer interaction. The specificity of the proposed text editor comes from the fact that it is guided by gestures instead of commands entered from the keyboard or the mouse.

The creation of a gesture based text editor requires the creation of a gesture recognizer, which is meant to associate a name with an unknown pattern. Choosing an algorithm and its implementation were the two main goals of the research described here. Designing

the editor involved compromising, due to the restrictive technical characteristics (memory and speed) of the used computer. Our goal is a reliable, compact, quick-responding and adaptable gesture guided editor.

Several pattern classification methods are described in the scientific literature. They can be broken down into two groups: symbolic methods and methods called *connectionist*. The solution exposed here is based on a connectionist approach, which seems to be especially well adapted to pattern-action association problem.

Among the existing supervised neural networks, we chose the ARTMAP neural network for its capability of fast adaptation and because it allows new gestures to be added in a relatively easy way. In the context of gesture editing this is an important feature. The ARTMAP is composed of two ART1 nets. ART1 was created as a binary images clustering network. We used it to classify gestures, which are, in fact, spatial signals. For this purpose we needed to find an appropriate preprocessing. We conceived a code called isometric, which makes it possible to take advantage of the operating simplicity of the ART1 net. The isometric codes specification became an important step in designing the proposed ARTMAP based gesture recognizer, which is the core of the aimed gesture guided editor.

Creating an operating gesture-based editor required the integration of three aspects: a gestural command recognition, context identification and command execution. Our editor was created by using Microsoft's Word'97, which had been adapted to our specific needs.

The goal of the testing was to determine the impact of recognizer's parameters on its performance. The recognizer has two parameters: the size of the ARTMAP input data space and the vigilance criterion of the net (Ro). The first parameter is specified by the number of segments (N) used to represent the forms and the prototypes in ARTMAP.

We defined a set of gestures and trained our recognizer with them. We conducted a research in the scientific and commercial literature concerning text-editing gestures and conclude the following: unlike the forms of digits or letters, the gestures' forms are not specified; on the other hand, there is a consensus that gestures' forms must be simple and easy to draw.

We also implemented all corresponding actions, because we were searching for an operating gesture-based editor. It is an adaptable editor in terms of user's ability to add new gestures to the initial gesture-set. Adding a new gesture (new form) is easy: we simply ask the user is to draw the new form.

The performance evaluation tests were carried out on a data base, taken from 14 future users. Every user wrote 14 gestures during three sessions, thus our data base totaled 588 forms. The computed gesture recognition rate is 99.5%. The evaluation tests helped as to choose the best preprocessing. The analysis showed that the created gesture recognizer is robust and tolerant to the variations of its parameters Ro and N.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ANNEXES

INTRODUCTION

L'interaction homme-ordinateur est un domaine de recherche très actif. Les ordinateurs sont devenus un outil de travail indispensable d'où l'intérêt de faciliter leur usage. Présentement les deux principaux médias d'entrée d'information à l'ordinateur sont le clavier et la souris. Cependant, on est témoin d'un développement intensif au niveau de la recherche universitaire et industrielle qui vise à explorer les deux nouvelles voies de communication : l'écriture et la parole. Dans le contexte de l'écriture, l'industrie informatique a mis sur le marché les bloc-notes électroniques ou ardoises électroniques qui ouvrent un domaine de recherche plein d'avenir et de défis : celui d'apprendre à l'ordinateur à "utiliser" ou plutôt à "comprendre" l'écriture manuscrite. Pour y parvenir, on utilise différentes techniques dont la reconnaissance de formes et l'analyse d'images.

Ce sont ces domaines qu'explorent les membres du laboratoire Scribens de l'École Polytechnique de Montréal. La recherche présentée ici a été effectuée dans ce laboratoire. Le groupe de recherche de Scribens a quatre grands champs d'activités :

- La modélisation du mouvement et de la perception ;
- La vérification des signatures ;
- Les applications utilisant un bloc-notes électronique ;
- La reconnaissance de l'écriture et l'analyse de documents.

Les travaux de recherche décrits dans ce mémoire sont axés sur le troisième et quatrième champs d'activités. Ils s'inscrivent dans un projet de reconnaissance de documents braille pour l'édition et la mise en ligne de textes sur le réseau Internet, projet développé conjointement par le laboratoire Scribens et divers intervenants extérieurs.

Le but de ce projet est de numériser des documents en braille pour assurer leur conservation, leur reproduction et leur diffusion sur le réseau Internet. Pour obtenir l'image d'une page braille, un scanner commercial est utilisé. Cette image représente

l'entrée d'un module de reconnaissance optique de graphèmes braille dont la sortie produit un texte ASCII correspondant au texte braille. Il est très probable qu'on retrouve des erreurs de reconnaissance, dues à plusieurs facteurs liés aux qualités des documents traités comme par exemple : l'âge du document, l'usure, la technique d'embossage utilisée et la qualité du support. L'intervention d'un humain est donc souvent indispensable pour rectifier ces erreurs.

L'ordinateur utilisé dans notre projet est une ardoise électronique, appelée bloc-notes électronique ou papier électronique, utilisant un crayon comme seul dispositif d'entrée. La particularité de l'éditeur que nous proposons est donc qu'il fonctionne à l'aide de commandes gestuelles plutôt qu'avec des commandes entrées au clavier ou avec une souris. Ce choix est justifié par le fait que l'utilisation d'un crayon devrait rendre l'interaction avec l'ordinateur naturelle, intuitive et rapide.

La création d'un éditeur guidé par gestes nécessite la création d'un module de reconnaissance de formes dont la tâche est d'associer une commande à une forme dessinée à l'écran. Le choix d'un algorithme et son implantation est le principal objectif de recherche du travail décrit dans ce mémoire.

Compte tenu des caractéristiques techniques limitatives, en termes d'espace mémoire et de vitesse de traitement de ces systèmes portables, l'éditeur de texte doit être compact, rapide d'exécution et surtout dans le contexte d'un éditeur qui se veut ergonomique, adaptable à l'usager. Nous devons donc envisager faire certains compromis lors du design de notre algorithme de reconnaissance de gestes.

Nos travaux de recherche ont donc été axés sur la création d'un éditeur gestuel, dont le noyau est le reconnaisseur des commandes gestuelles. La création de l'éditeur gestuel fonctionnel nécessite l'integration de trois aspects soit: la reconnaissance de commandes gestuelles, l'identification du contexte et l'exécution des commandes. Pour créer

l'éditeur gestuel nous avons utilisé le logiciel Word'97 de Microsoft et nous l'avons adapté à

nos besoins. Enfin, pour l'environnement de programmation et le système d'exploitation, nous avons retenu Microsoft Visual Basic, version 5.0 et Microsoft Windows 95.

Le mémoire se compose de cinq chapitres :

- Le premier chapitre présente les idées de base nécessaires à la compréhension de l'ouvrage : tout d'abord le contexte dans lequel ce projet a été développé, nous enchaînons ensuite avec la définition des caractéristiques des applications crayon. Finalement, nous présentons certaines explications portant sur les commandes gestuelles en général ainsi que sur les notions de base importantes de la reconnaissance des formes.
- Le deuxième chapitre décrit les démarches entreprises lors de la conception de l'éditeur gestuel. Plusieurs détails techniques sur ce projet y sont décrits et des explications portant sur les problèmes rencontrés, liées à la complexité de développement d'une application à crayon dans l'environnement Windows et de l'adaptation d'un éditeur de texte commercial aux besoins de l'entrée crayon.
- Le troisième chapitre résume une recherche bibliographique sur les commandes gestuelles existantes utilisées pour l'édition de texte. Les commandes gestuelles proposées par des chercheurs et celles proposées par des compagnies sont regroupées séparément. À la suite de cette revue, nous définissons un ensemble de gestes qui a été utilisé pour l'évaluation du reconnaisseur de gestes, proposé.
- Dans le quatrième chapitre, le recours à l'approche connexionniste pour la reconnaissance des commandes gestuelles et le choix du réseau ARTMAP sont justifiés. La nature des données et les différents prétraitements sont discutés à la lumière de notre approche. Ici on introduit le code Isométrique qui a permis la classification des signaux spatio-temporels par le réseau ART1 et on discute ses

avantages pour un réseau ARTMAP. On montre également comment l'adaptabilité de l'éditeur peut être réalisée grâce à cette approche.

- Le but du cinquième chapitre est l'évaluation du reconnaisseur gestuel sur une banque de gestes obtenue auprès d'utilisateurs potentiels. Nous avons conçu une application à l'aide de laquelle nous avons constitué notre base de gestes. La vérification expérimentale fut réalisée en changeant plusieurs paramètres du reconnaisseur des commandes gestuelles. De plus, deux types de prétraitement ont aussi été testés afin de choisir le meilleur.

CHAPITRE I
NOTIONS PRÉLIMINAIRES

L'objectif de ce premier chapitre est de présenter la problématique du projet et de définir certaines notions utilisées de façon à faciliter la compréhension de ce mémoire. Dans un premier temps, il sera question du contexte dans lequel ce projet a été développé pour, dans un deuxième temps, définir les caractéristiques d'une application crayon. Finalement des explications portant sur les commandes gestuelles en général ainsi que sur les notions de base dans la reconnaissance de formes seront présentées.

1.1 Problématique générale.

Le travail décrit dans ce mémoire est issu d'un projet de reconnaissance du braille pour l'édition et la mise en ligne de textes sur réseau Internet, projet développé conjointement par le laboratoire Scribens et divers intervenants extérieurs[1]. Le but recherché est de pouvoir numériser des documents en braille pour en assurer leur conservation, leur reproduction et leur diffusion sur le réseau Internet.

La matière première du projet est constituée de documents braille qui sont principalement localisés dans les bibliothèques destinées aux déficients visuels, comme celle de l'Institut Nazareth et Louis Braille. Il s'est avéré essentiel de bien connaître avant tout, les particularités de ce type de documents. Dans le but de trouver des réponses à nos nombreuses questions concernant des détails importants de ce projet deux visites furent effectuées à l'Institut Nazareth et Louis Braille. Le rapport technique (Gourrier, Sabeva, Brault, Plamondon, 1997) contient un compte rendu complet des caractéristiques reliées aux documents Braille provenant de la bibliothèque ainsi que des

[1] Cedrom-SNI, Prosys-Tec Inc., Visuaide, Bibliothèque Jeanne Cypihot, La Magnétothèque et Institut Nazareth et Louis Braille.

informations résultant de nombreuses discussions avec le personnel de l'Institut Nazareth et Louis Braille.

Dans ce projet, deux types de documents braille sont considérés:

- des vieux documents Braille n'existant pas sous forme ASCII ;
- des documents Braille plus récents, qui existent sous forme électronique quelque part mais pas à l'Institut Nazareth et Louis Braille (des livres achetés ailleurs), l'Institut est par conséquent intéressé à posséder des copies de sécurité, pour permettre la reproduction de ces livres en cas de perte.

La partie principale du projet consiste en la réalisation d'un système logiciel qui, à partir de l'image d'une page Braille obtenue à l'aide d'un scanner commercial, permettra d'obtenir le plus fidèlement possible l'équivalent ASCII du texte balayé.

La figure 1.1 décrit un tel système. Un scanner est requis pour fournir l'image d'une page en braille. Cette image est l'entrée du reconnaisseur du braille et à la sortie un texte ASCII est alors obtenu.

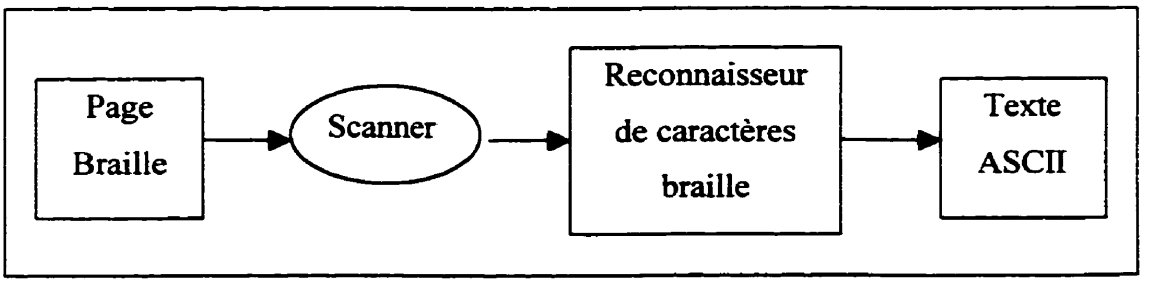


Figure 1.1 Le schéma bloc simple, représentant la partie reconnaissance du Braille du projet.

La reconnaissance optique du braille a été réalisée par un autre chercheur de l'équipe. Il est très probable que des erreurs de reconnaissance surviennent, dues à plusieurs facteurs liés aux qualités des documents traités (Gourrier, Sabeva, Brault, Plamondon, 1997) telles que: l'âge du document, l'usure, la technique d'embossage utilisée et la

qualité du support. Il est logique de supposer qu'il existe d'autres facteurs qui causent des erreurs dans le texte. Quelques facteurs sont nommés sur la figure 1.2, par exemple : papier foncé, papier tacheté, bruit lors de balayage, erreur survenu dans le module de reconnaissance de graphèmes Braille ou tout simplement des erreurs introduites au moment de l'impression de document – erreurs de français. L'intervention d'un humain est donc indispensable pour corriger ces erreurs.

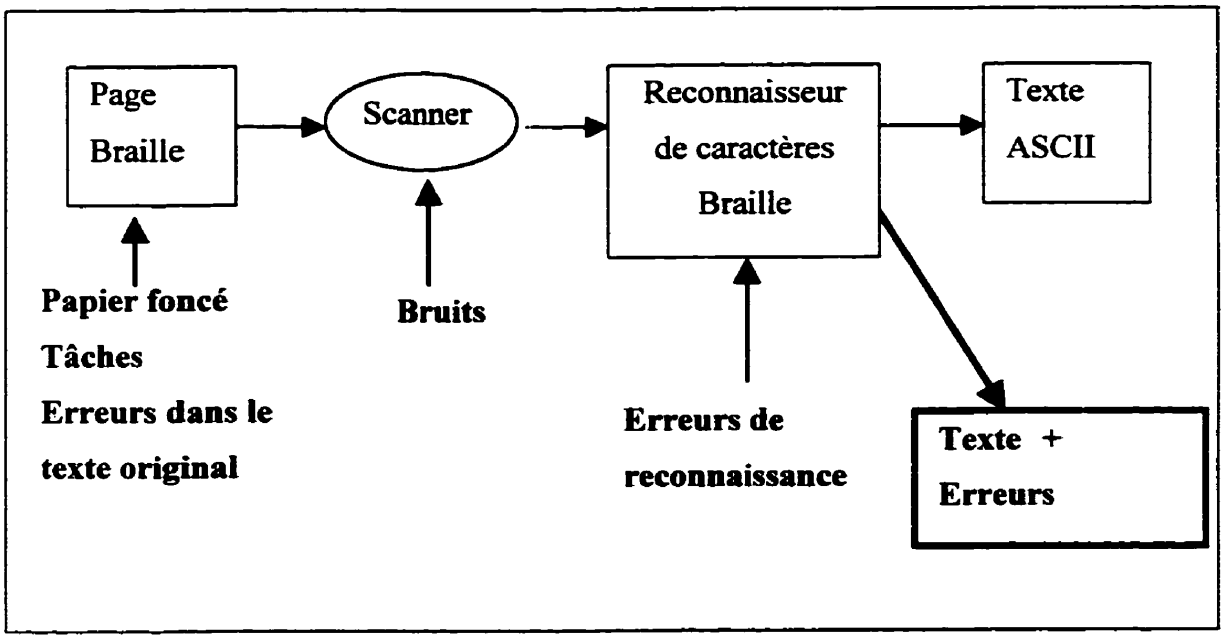


Figure 1.2 Une version plus détaillée de la figure 1.1.

Il faut donc offrir les outils nécessaires à l'édition du texte brut issu du reconnaisseur des caractères braille. La figure 1.3 contient le diagramme de flux de données décrivant globalement le projet. La chaîne de traitement commence par l'acquisition des images de pages braille, suivie par un module qui filtre les images pour produire ainsi l'entrée de module de reconnaissance de graphèmes braille la sortie duquel on obtient un texte ASCII correspondant au texte braille. Ce texte est affiché pour une vérification. Lors des corrections du texte l'entrée d'information se fait à l'aide d'un crayon.

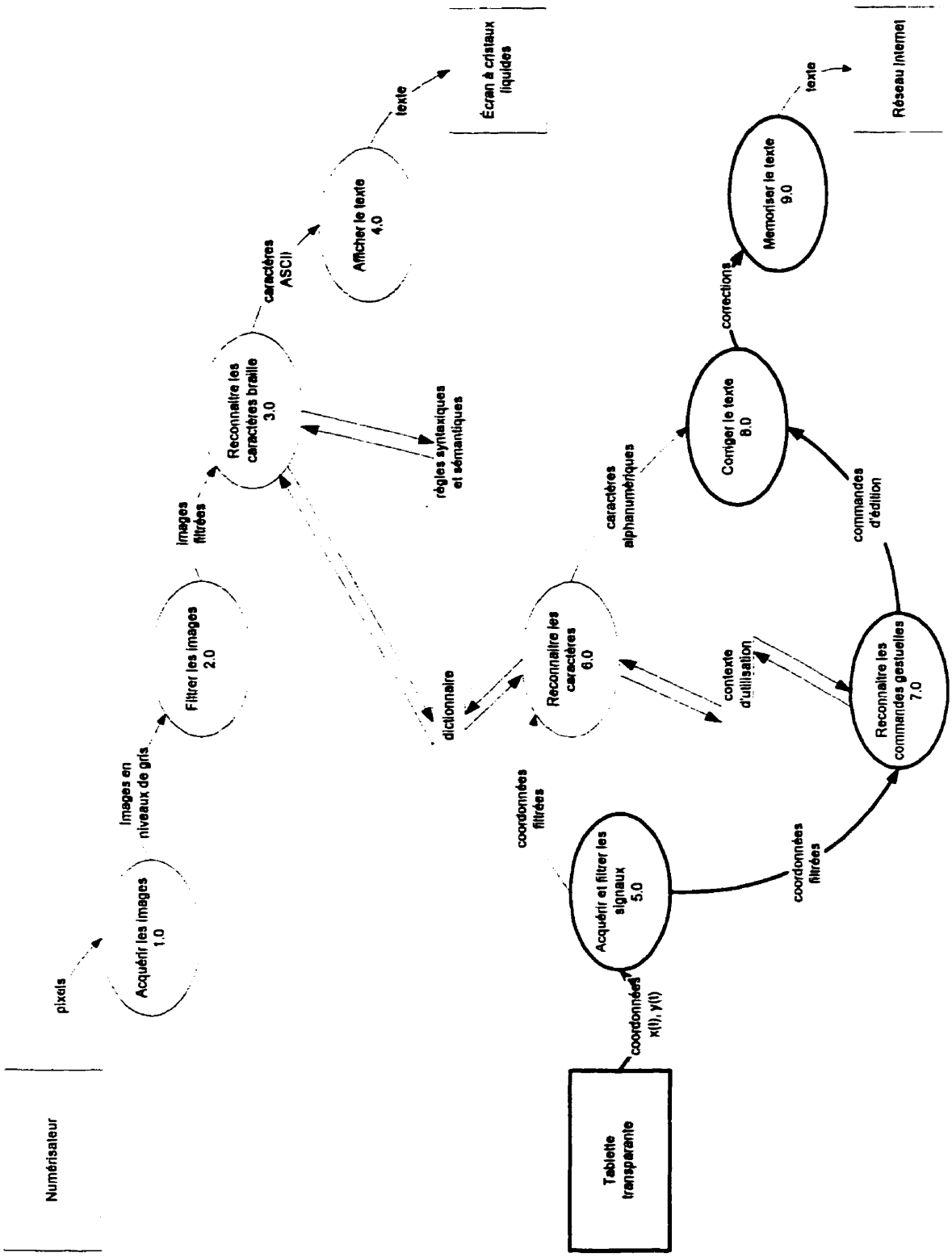


Figure 1.3 Diagramme de flux de données décrivant globalement le projet.

Les signaux acquis à l'aide d'une tablette transparente sont filtrés et transmis aux deux modules suivants :

- le module de reconnaissance de caractères;
- le module de reconnaissance de commandes gestuelles.

Après la rectification des erreurs, le texte prêt à être publié sur Internet est sauvegardé. Les modules et flèches en gras sur la figure 1.3 représentent la contribution de ce projet.

La configuration typique d'un système informatique comporte un clavier et une souris qui sont les outils principaux de la communication entre l'usager et la machine. Dans ce contexte, l'édition se fait à l'aide de logiciels spécialisés - logiciels de traitement de texte, qui contiennent tout d'abord une fenêtre qui visualise le texte et aussi des menus. Malgré les avantages que cette configuration offre au niveau de l'introduction du texte, elle n'est souvent pas assez efficace pour la tâche d'édition du texte, surtout parce que l'utilisateur doit constamment changer le point de son attention entre la place dédiée à l'affichage du texte, les menus et la manipulation de la souris et du clavier. C'est pourquoi les professionnels en édition – les correcteurs d'épreuves travaillent sur une copie papier d'un document en utilisant des signes de correction typographique. Cette technique a l'avantage d'être très rapide. Ceci explique pourquoi les bloc-notes électroniques peuvent s'avérer très appropriés à ce projet ainsi que l'utilisation des commandes gestuelles.

L'ordinateur utilisé dans notre projet est une ardoise électronique, aussi appelée bloc-notes électronique ou papier électronique, utilisant un crayon comme seul dispositif d'entrée. L'écran sensible de ce type d'ordinateur enregistre et affiche les traces du mouvement du crayon générant ainsi de l'encre dite électronique.

Il existe quatre types de données qui peuvent être entrées en utilisant un crayon (Schomaker, 1998) : entrée de texte, entrée de commandes, entrée d'objets graphiques et entrée de signatures. Dans ce mémoire on parlera de commandes gestuelles dans le cadre

d'édition de texte. Dans la littérature scientifique on retrouve deux tendances liées à la conception d'applications crayon. La première exploite la métaphore papier –crayon. Elle établit le parallèle entre l'écriture avec un crayon sur papier et l'écriture avec un crayon sur l'écran d'un bloc-notes électronique. Tandis que la deuxième met l'accent sur la différence entre le papier et l'ordinateur. Il semble que la deuxième tendance, qui est également la plus récente, est plus plausible. Notre expérience en utilisant le Stylistique 1000 fait croire que la deuxième tendance reflète mieux la réalité. En effet l'écriture sur papier diffère de l'écriture sur le papier électronique. La différence se trouve surtout dans l'interaction. Le papier est simplement un récepteur statique d'information, tandis que l'environnement ordinateur est très dynamique. Tout en sachant que la conception d'une application crayon doit commencer avec le développement d'un système d'exploitation spécialement conçu pour le crayon, ce qui en réalité déborde de cadre de ce travail, on a essayé d'utiliser les modules existant afin de créer un éditeur de texte qui soit guidé par nos gestes. Le but était de réunir les avantages de l'utilisation des ordinateurs et la commodité du crayon dans un environnement typique tel que Windows. Les objectifs de ce projet de maîtrise sont : la conception et la réalisation d'un éditeur gestuel de texte. Ce travail a deux grands volets : la conception du reconnaisseur de commandes gestuelles et la conception d'un cadre fonctionnel permettant l'édition gestuelle.

1.2 Applications crayon

Un des buts du projet est de concevoir une application crayon. Les applications crayon sont devenues une partie importante dans le domaine des communications homme-ordinateur, parce qu'elles offrent plusieurs avantages. Dans le passé, le crayon a parfois été utilisé comme un outil de pointage et de sélection précis. L'utilisation du crayon exige moins d'efforts mentaux et visuels, et réduit les possibilités d'erreurs tout en étant très facile à utiliser (Millen, 1993). Des scientifiques (Coleman, 1969, Suenaga et Nagura, 1980, Wolf et. Morrel-Samuels, 1987, Kankaanpaa, 1988, Kim,1988, Welbourn

et. Whitrow, 1990, Lipscomb,1991) et plusieurs compagnies (Slate corporation, 1991, Apple,1993, GO corporation,1996, Fujutsu Personal Systems,Inc, 1996) ont saisi l'importance des applications et des produits munis des fonctions exploitant la vraie nature du crayon.

1.2.1 Caractéristiques des applications crayon

Quatre caractéristiques se rattachent à une application crayon :

- L'utilisation du crayon pour pointer et sélectionner.
- L'émission et la manipulation d'encre électronique.
- L'utilisation du crayon pour activer des commandes.
- L'application des algorithmes de reconnaissance de caractères manuscrits, pour permettre l'entrée de texte à l'aide du crayon.

L'utilisation du crayon pour pointer et sélectionner est très semblable à l'utilisation de la souris. Par contre la génération, ainsi que la manipulation, de l'encre électronique est l'élément distinctif relié à l'utilisation propre du crayon. L'encre électronique représente la capacité qu'a un système informatique d'accepter des données graphiques et textuelles manuscrites et de les sauvegarder exactement comme elles sont écrites afin de permettre leur visualisation authentique. Certains voient l'encre électronique comme un nouveau type de données. Les traits du mouvement du crayon sont capturés (par une tablette digitalisante par exemple) et une représentation électronique est alors créée et émise. En même temps, cette représentation peut être sauvegardée. Au besoin, elle peut être visualisée, transmise et transformée si désiré. Dans de nombreuses applications l'encre électronique est transmise à un reconnaisseur de formes ou bien à un reconnaisseur de caractères et transformée en texte ou en objets graphiques. Il existe aussi des applications crayon appelées "notetaking applications" (Brault, Plamondon et Laframboise, 1995) dans lesquelles l'encre électronique est utilisée pour mémoriser des notes personnelles. La troisième caractéristique d'une application crayon est l'utilisation

du crayon pour l'activation des commandes. Le mécanisme est le suivant : à l'aide du crayon on écrit des formes prédéfinies appelées des gestes ou des commandes gestuelles dans le but d'exécuter une commande. Une commande gestuelle est en effet une forme dessinée par l'utilisateur d'un système informatique pour déclencher l'exécution d'une ou plusieurs actions. L'utilisation des commandes gestuelles est souvent associée à une communication homme-ordinateur plus naturelle, intuitive et rapide. Il faut mentionner qu'il existe aussi des problèmes liés à l'utilisation des commandes gestuelles. Ces problèmes se présentent sous deux formes particulières dont la première est la reconnaissance des gestes et la deuxième est le degré de consistance dans l'utilisation des gestes chez des usagers. Quelques recherches ont été effectuées dans le but de connaître ce dernier problème en détail. Les chercheurs Walf et Morrel-Samuels ont découvert entre autres en 1987, une grande consistance individuelle lors de définition de gestes par l'utilisateur.

La reconnaissance des caractères manuscrits est la quatrième caractéristique des applications crayon. Les milieux scientifique et industriel portent un grand intérêt à ce domaine, qui s'explique par la demande croissante de nouvelles méthodes d'entrée de texte dans de "petits" systèmes informatiques comme les ordinateurs portables, les blocs-notes électroniques, les organisateurs (agendas) personnels, etc. La performance des reconnaisseurs d'écriture manuscrite est très importante pour qu'un tel système soit accepté par les usagers.

1.3 La tâche d'édition

Comme on l'a déjà mentionné notre but est d'unir les avantages liés à l'utilisation de l'ordinateur (puissance, rapidité, services et outils disponibles) à la commodité du crayon. Les outils de traitement de l'information par ordinateur sont très développés, généralement ils ne sont pas des variants électroniques de leurs prototypes sur papier et ils sont plus performants, plus puissants et moins exigeants. Ces outils sont à présent,

couramment utilisés dans plusieurs domaines. Les outils de traitement de texte font partie de ses outils de traitement de l'information par ordinateur. Mais, l'édition de texte par ordinateur est en effet, différente de l'édition sur une page effectuée à l'aide d'un crayon. La démarche suivie pour créer l'éditeur gestuel dans ce projet diffère de l'approche classique. D'ordinaire lorsqu'il s'agit de créer une application crayon, il faut analyser tout d'abord l'exécution de la même tâche sur papier. Cette analyse sert à définir les fonctionnalités de l'application crayon. Cette approche est considérée à présent non adéquate (Schomaker, 1998). Voilà pourquoi lors de la conception nous avons suivi une voie différente. D'abord nous avons commencé par comprendre les particularités de notre projet. Le but ultime du projet est l'obtention d'équivalents ASCII des documents en Braille (figure 1.4). Partant du fait qu'on dispose d'un document qui a un contenu et une structure qui ne doivent pas être altérés, la tâche d'édition est restreinte car il ne s'agit pas de corriger des phrases, ni d'améliorer la syntaxe, ni de restructurer le texte. On peut donc déduire qu'il est interdit de changer l'ordre des entités textuelles (mots, phrases, paragraphes) de même que de remplacer des entités textuelles dans le but de préciser ou d'en changer le sens. Tout ce qui est permis c'est de corriger des fautes d'orthographe.

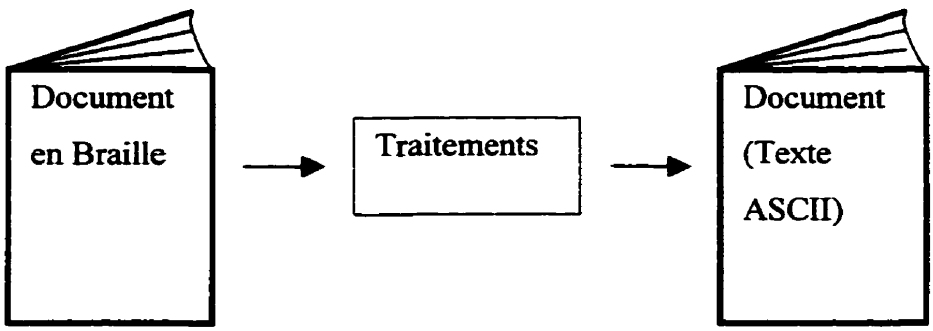


Figure 1.4 Le document-source et le document visé avec ce projet.

Les sources d'erreurs dans le texte ASCII ont été discutées dans la première section de ce chapitre. Mais quelle que soit la source des erreurs, dans le contexte d'édition il apparaît important de s'intéresser d'abord à bien les identifier pour pouvoir ensuite les

rectifier. Les correcteurs d'orthographe font partie des logiciels de traitement de texte depuis longtemps, par contre l'identification des fautes d'orthographe en-ligne est une nouvelle fonction, ainsi que la possibilité d'accéder rapidement à une liste de suggestions. L'identification des fautes d'orthographe exige beaucoup d'attention, de concentration et de connaissances de langue. C'est pourquoi la possibilité d'automatiser la tâche d'identification d'erreurs d'orthographe devient une fonctionnalité fortement appréciée dans ce présent contexte d'édition.

On ne dispose d'aucune information sur les erreurs qui surviennent dans le texte ASCII. On ne connaît pas exactement leur nature, ni leur fréquence. La recherche bibliographique effectuée par un de mes collèges n'a pas vraiment permis d'apporter d'éclaircissement précis sur ce sujet.

Une fois que les erreurs sont localisées, on utilise le crayon pour pointer, sélectionner et pour écrire du texte et des gestes. L'utilisation du crayon à la place d'une souris (pointer et sélectionner) permet d'éliminer les erreurs souvent engendrées par l'analyse du contexte et permet ainsi la création d'un éditeur de texte fiable.

1.3.1 Qualités visées de l'éditeur gestuel

Compte tenu des caractéristiques techniques limitatives (en termes d'espace mémoire et de vitesse de traitement) des systèmes portables (Stylistique 1000 de Fujitsu), l'éditeur de texte doit être compact, rapide et, dans le contexte d'un éditeur qui se veut ergonomique, adaptable à l'usager. Il faut donc accepter de faire certains compromis dans le design de l'algorithme de reconnaissance de gestes conçu pour ce projet.

1.4 Commandes gestuelles - notions

L'idée d'utiliser des commandes gestuelles a été développée et proposée en 1969 par Michel Coleman (Coleman, 1969). Il a proposé de combiner l'utilisation d'un crayon et d'une interface graphique pour former la base des systèmes contemporains de blocs-notes électroniques. Les deux problématiques principales de ces interfaces sont, d'une part qu'une commande gestuelle peut ne pas être reconnue correctement, et d'autre part, que l'utilisateur peut oublier la forme d'une commande. Une solution est de choisir des formes simples, faciles à dessiner, à apprendre et à comprendre ou d'offrir encore la possibilité d'adapter l'ensemble de commandes gestuelles aux besoins de l'utilisateur. Ainsi l'interaction d'un humain avec l'ordinateur se fera de façon naturelle, intuitive et efficace.

1.4.1 Définition

Définition 1 : Une commande gestuelle est (Schomaker, 1998) un symbole non alphanumérique qui après avoir été dessiné entraîne l'exécution d'une fonction donnée. Le mouvement élémentaire en écriture est une composante (Plamondon, Maarse, 1989) trait. Une composante correspond au tracé laissé par un crayon entre le moment où le crayon touche la surface sensible de l'écran dans notre cas, et le moment où il quitte cette même surface.
Définition 2 : Les commandes gestuelles sont des trajectoires du crayon qui sont limitées dans le temps et dans l'espace et qui ont des formes caractéristiques. En général, leurs formes diffèrent de celles utilisées pour les lettres et les chiffres.

1.4.2 Étapes de traitement

La figure 1.5 représente les trois étapes principales du traitement d'une commande gestuelle : l'acquisition de données, la reconnaissance du geste ou de la forme et

l'exécution de la commande. Le processus de traitement est déclenché lorsque l'utilisateur dessine et pose un geste. L'acquisition de données dans notre système se fait en utilisant le bloc-notes électronique Stylistic1000 de la compagnie Fujitsu. La deuxième étape consiste à reconnaître la forme dessinée. Il y a deux issus possibles :

- on associe la forme à un des groupes de formes connues ;
- on rejette la forme, ce qui signifie que la forme dessinée diffère considérablement des formes connues par le classificateur.

La troisième étape consiste à exécuter les actions (ou l'action) associées à la forme dessinée.

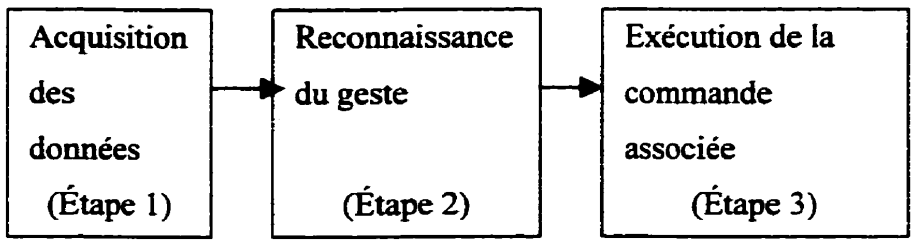


Figure 1.5 Les trois étapes de traitement d'une commande gestuelle.

1.4.3 La place de l'analyse du contexte dans le traitement d'un geste

Pour qu'une commande gestuelle soit exécutée on ne peut avoir recours seulement à la reconnaissance de sa forme. Cela n'est pas suffisant. Il faut aussi analyser le contexte et extraire les attributs de la commande gestuelle, ce qu'on appelle la portée de la commande. Les attributs d'un geste sont toutes les informations nécessaires pour son exécution. Le tableau 1.1 montre quelques exemples de commandes gestuelles utilisées pour éditer du texte. Par exemple, la commande "Joindre" implique qu'une fois que la forme est reconnue, il faut pouvoir déterminer quels sont les deux mots latéraux qui doivent être juxtaposés et connaître leur position. À partir de ces informations il devient possible alors d'exécuter le geste. Prenons un autre exemple, soit la commande "Effacer". Après la reconnaissance de la forme il faut identifier l'entité textuelle qui

devrait être effacée. Cette entité peut être une lettre, un mot, une expression, etc. L'identification peut donc devenir assez difficile à effectuer. Les attributs d'une commande gestuelle sont en fait une liste de données dont la longueur dépend de la nature de l'action qui lui a été associée. Cette liste pourrait aussi être vide. Dans ce cas, il s'agit d'action qui n'a pas d'attribut, comme par exemple la commande "Annuler la dernière action" (Undo).

Tableau 1.1 Exemples des commandes gestuelles.

Nom de fonction	Représentation graphique
Insérer	Les commandes gestuelles sont faciles à dessiner.
Disjoindre	Toutes les interfaces sont.....
Joindre	La conception d'un interface est une tâche....
Effacer	Toutes les suggestions et recommandations ne sont pas bien venues.
Sélectionner	Pour sélectionner on peut encercler ou souligner.

La construction de la liste d'attributs se fait par une analyse du contexte qui peut être vu comme une sous-étape du traitement d'un geste. Par conséquent, l'identification du contexte est aussi une source d'erreurs. Imaginons la situation suivante : on dessine une forme et la forme dessinée est correctement reconnue, mais la commande associée est exécutée sur un objet textuel erroné. Une façon d'éliminer les erreurs d'analyse du contexte est d'utiliser le crayon comme un outil de pointage (Ex. à l'aide du stylo positionner le curseur entre les deux mots qui doivent être unis) et de sélection (Ex. à

l'aide du stylo sélectionner le mot à effacer). De cette façon, le contexte est clairement défini par une technique simple et les ambiguïtés sont ainsi évitées.

1.4.4 Notre choix : commandes gestuelles d'un trait du crayon

Les commandes gestuelles sont définies comme étant des trajectoires qui ont des formes caractéristiques. En général, elles peuvent comporter plusieurs composantes, mais il est préférable (Sabeva, Brault et Plamondon,1997) de choisir des formes simples, d'une seule composante. Dans ce projet, des formes simples ont été choisies qui peuvent être dessinées d'un seul mouvement de crayon. Cette décision comporte plusieurs avantages. Le premier est que les gestes sont faciles à reproduire, donc l'éditeur sera facile à utiliser et ça affectera aussi la rapidité d'édition. Le deuxième avantage est que les formes simples sont plus faciles à reconnaître et que les algorithmes qui en découlent demandent moins de calculs, ce qui permet d'obtenir un outil de reconnaissance compact. Le troisième avantage est que la distinction entre les gestes et l'écriture manuscrite est facilitée par ce choix, ce qui revête un aspect important à considérer dans un éditeur de texte.

1.5 Reconnaissance de commandes gestuelles - notions

1.5.1 Stratégies de comparaison

Les commandes manuscrites sont en fait des formes graphiques. On peut considérer une forme comme une réalité qui a une structure observable. Par conséquent, on peut apprendre une forme en l'observant tout comme on peut aussi comparer deux formes. La comparaison entre une forme observée et une forme apprise mène au concept de la reconnaissance des formes. En comparant deux formes, on les analyse pour décider si elles se ressemblent, ce qui correspond au travail fait par un classificateur. Le classificateur permet de comparer plusieurs formes pour proposer un classement. Un

classificateur est un mécanisme qui associe à une observation une catégorie ou une classe. Ce procédé sous-entend que les classes sont bien définies au moment de la classification. La détermination des classes se fait en général pendant une phase d'apprentissage. En principe, il existent deux stratégies de comparaison de formes : la comparaison directe et la comparaison indirecte.

- La comparaison dite "directe" se définit comme suit : une comparaison qui est faite à un premier niveau de présentation, entre une observation et une forme apprise (prototype). Par exemple, dans le cas de la reconnaissance optique des caractères, il est possible de conserver une matrice de points qui représente un prototype et de la comparer à la matrice qui correspond à la forme à classifier.
- La comparaison dite "indirecte", permet l'analyse d'une représentation dans l'espace de caractéristiques. Il est évidemment délicat de définir ces caractéristiques. Cette approche permet d'envisager une comparaison non pas sur des informations "brutes", mais sur des caractéristiques pertinentes, c'est-à-dire généralisables à une classe donnée, et robustes pour le problème spécifique à l'étude.

1.5.2 Processus de reconnaissance

Le fonctionnement d'un système de reconnaissance de formes peut être décomposé en trois niveaux (figure 1.6), soit l'acquisition de données, le prétraitement, et l'analyse/décision. L'acquisition de données se fait en utilisant le bloc-notes électronique Stylistic1000. L'étape de prétraitement consiste à éliminer le bruit et à normaliser les données. Lors de l'étape d'analyse, on calcule un certain nombre de caractéristiques qui correspondent à des mesures de nature géométrique, topologique ou statistique. Finalement, le rôle de l'étape de décision ou de classement, est d'identifier ou de classer la forme en ayant recours à différentes stratégies de comparaison.

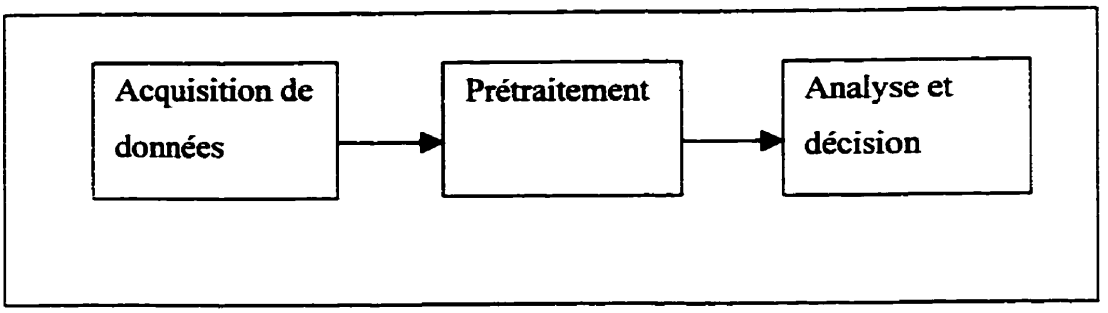


Figure 1.6 Schéma bloc du processus de reconnaissance.

1.5.3 Représentation interne des gestes

Une fois l'acquisition des données faite ,on dispose d'un signal d'entrée. Dans les applications crayon il s'agit d'une séquence de vecteurs : (X_k, Y_k). L'avantage principal des plates-formes qui utilisent le crayon est que deux représentations sont disponibles : la représentation statique ou image et la représentation dynamique (des coordonnées en fonction du temps). Si l'algorithme de reconnaissance fait usage uniquement d'information statique, on dira de cette reconnaissance qu'elle est hors-ligne. Les méthodes hors-ligne sont rarement utilisées dans les applications crayon, elles sont surtout utilisées pour le traitement d'image et ne seront pas abordées ici. Les méthodes de reconnaissance en-ligne sont celles qui sont utilisées dans le domaine de la reconnaissance d'écriture manuscrite. Dans le cas de gestes il est possible de représenter le geste spatio-temporel de la figure 1.7 de trois façons différentes,

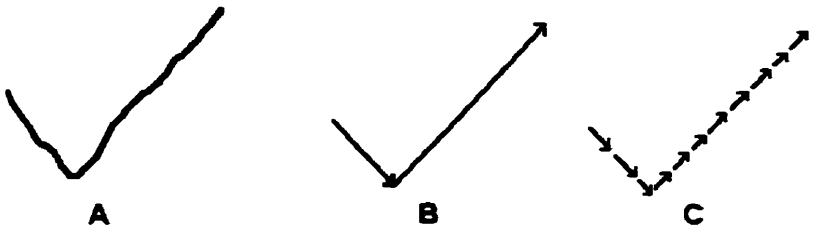


Figure 1.7 Trois façons de représenter un geste.

soit à l'aide de pixels (l'information spatiale est perdue), soit à l'aide de séquences extraites (comme B et C). Dans le cas B, une séquence de deux segments est codée en longueur Li, et en direction θi, où i=1,2,...nombre de segments. Par exemple, le geste de la figure 1.7-A sera présenté, par l'ensemble suivant : $\{(L_1,\theta_1), (L_2,\theta_2)\}$. Le problème avec cette représentation est qu'il faut détecter l'existence et la position de points de changement de direction. Ceci exige une analyse supplémentaire et une série de décisions arbitraires pouvant introduire des erreurs d'interprétation. Dans le cas C, une séquence de N segments de longueur égale est codée en direction θ(i), i=1 à N. Dans l'exemple de la figure 1.7-C, il s'agit de onze segments et le geste sera présenté par cette séquence : $\{\theta_1, \theta_2, \theta_3, \ldots, \theta_{11}\}$. Le seul arbitraire ici est de choisir un nombre N suffisamment grand pour bien représenter le geste. Dans ce qui suit, nous utilisons ce type de représentation.

Dans ce premier chapitre, la problématique générale du projet a été présentée et le travail a été situé comme étant une partie du projet de reconnaissance du braille pour l'édition et mise en ligne de texte sur réseau Internet.

CHAPITRE II
CADRE FONCTIONNEL DE L'ÉDITEUR

L'objectif de ce chapitre est de décrire les démarches entreprises lors de la conception de l'éditeur gestuel. Plusieurs détails techniques sur ce projet y sont décrits ainsi que des explications portant sur les problèmes rencontrés liés à la complexité d'adaptation d'une application de traitement de texte commerciale aux besoins de l'entrée crayon. Certaines informations qui viendront faciliter la compréhension du fonctionnement de base de chaque application à crayon.

Lors du travail sur la conception de l'éditeur gestuel on a rencontré des problèmes, qu'on n'a pas pu résoudre dans le cadre de ce projet. L'aide d'un professionnel était indispensable. Une étude à été faite par Chantal Perugino de SoftZen Inc. Elle a effectué des recherches sur le développement d'applications à crayon dans l'environnement Windows. Elle a également exploré les approches possibles pour répondre aux besoins particuliers de ce projet. Dans son rapport (Perugino, 1997) une approche a été suggérée et un cadre fonctionnel de départ a été développé pour démontrer la faisabilité de cette approche et mieux comprendre le fonctionnement des applications à crayon. Cette compréhension a ensuite permis la création d'un cadre fonctionnel qui diffère considérablement de celui proposé, mais qui fait usage des informations de ce rapport.

2.1 Analyse des besoins de l'édition gestuelle

L'analyse des besoins de l'édition gestuelle vise à déduire les fonctionnalités de base de l'éditeur guidé par des commandes gestuelles. La tâche de l'édition de texte consiste d'abord à afficher ce texte, deuxièmement à l'éditer à l'aide de gestes manuscrits et finalement, à le sauvegarder. La deuxième tâche, l'édition gestuelle, est la partie originale non supportée par les logiciels du traitement de texte. L'édition gestuelle est liée directement au processus de traitement des commandes gestuelles (voir la section

1.4.2 "Étapes de traitement" du premier chapitre). Le processus de traitement est déclenché par l'utilisateur à partir d'un geste. Cette possibilité d'écrire provient de la capacité d'afficher l'encre électronique, est la première fonctionnalité qui doit être supportée par notre éditeur. La deuxième étape du traitement est la reconnaissance du geste, qui consiste à traiter, selon un algorithme précis, les données provenant du crayon qui doivent être collectées puis sauvegardées. L'exécution du geste commence par la reconnaissance de la forme qui vient d'être tracée pour identifier l'action qui lui est associée (Ex. Effacer, Insérer, Annuler la dernière action, etc.). Cependant, l'exécution de la commande (ou de l'action) n'est lancée que lorsque ses attributs ont été déterminés.

L'utilisation du crayon comme souris (pour pointer et sélectionner) est un mécanisme "pont" qui constitue un moyen d'accéder à la structure hiérarchique du texte et de définir les attributs de la commande gestuelle. Pour exécuter un geste, il faut connaître la structure du texte et savoir comment altérer celui-ci.

Ce processus se répète avec chaque geste de l'utilisateur et se termine par la sauvegarde du texte. Ainsi, pour créer un éditeur gestuel, il faut être en mesure d'afficher, de manipuler et de sauvegarder l'encre électronique et le texte simultanément.

2.2 Plate-forme pour le développement d'applications à crayon et Types d'applications à crayon

Windows for Pen (WfP) est une plate-forme créée par Microsoft et mise sur le marché en 1991 pour faciliter le développement des applications à crayon. Il existe deux versions de 16 bits : la version 1.0 et la version 2.0. WfP fait partie de Windows'95. Le bloc-notes du projet Stylistic 1000, utilise Windows'95 et la version 2.0 des services pour crayon.

La figure 2.1 fait ressortir que les applications à crayon et les reconnaisseurs (de caractères manuscrits, de chiffres, de commandes gestuelles) existent séparément et que le système d'exploitation Windows assure la communication entre eux.

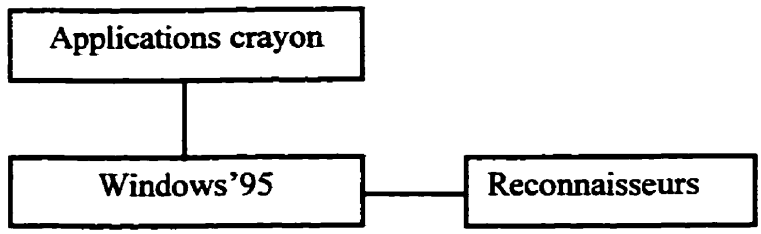


Figure 2.1 Les applications à crayon et les reconnaisseurs versus le système d'exploitation Windows.

WfP est un ensemble d'API[2], responsables de la collecte, de la visualisation et de la distribution de l'encre électronique lors de l'écriture. WfP fournit en principe deux types de services :

- Le système qui gère l'encre électronique (le nom de la librairie est PENWIN.DLL)
- Le système qui collecte, modifie et procède à la reconnaissance des données du crayon (le nom de la librairie est PKPD.DLL)

La figure 2.2 illustre de façon détaillée le principe de la figure 2.1. Le cheminement des données diffère selon le type d'application à crayon. Il existe deux types d'applications à crayon : des applications qui sont "conscientes" d'utiliser le crayon et des applications qui ne le sont pas. On désigne les premières comme étant les applications "orientées" crayon. Elles n'ont pas besoin de passer par l'interpréteur de messages crayon. Elles transmettent les messages crayon au reconnaisseur par défaut en utilisant des API de

[2] Interface de programmation d'application

WfP, tandis que les d'applications non "orientées" crayon passent par le système d'interprétation des messages du crayon.

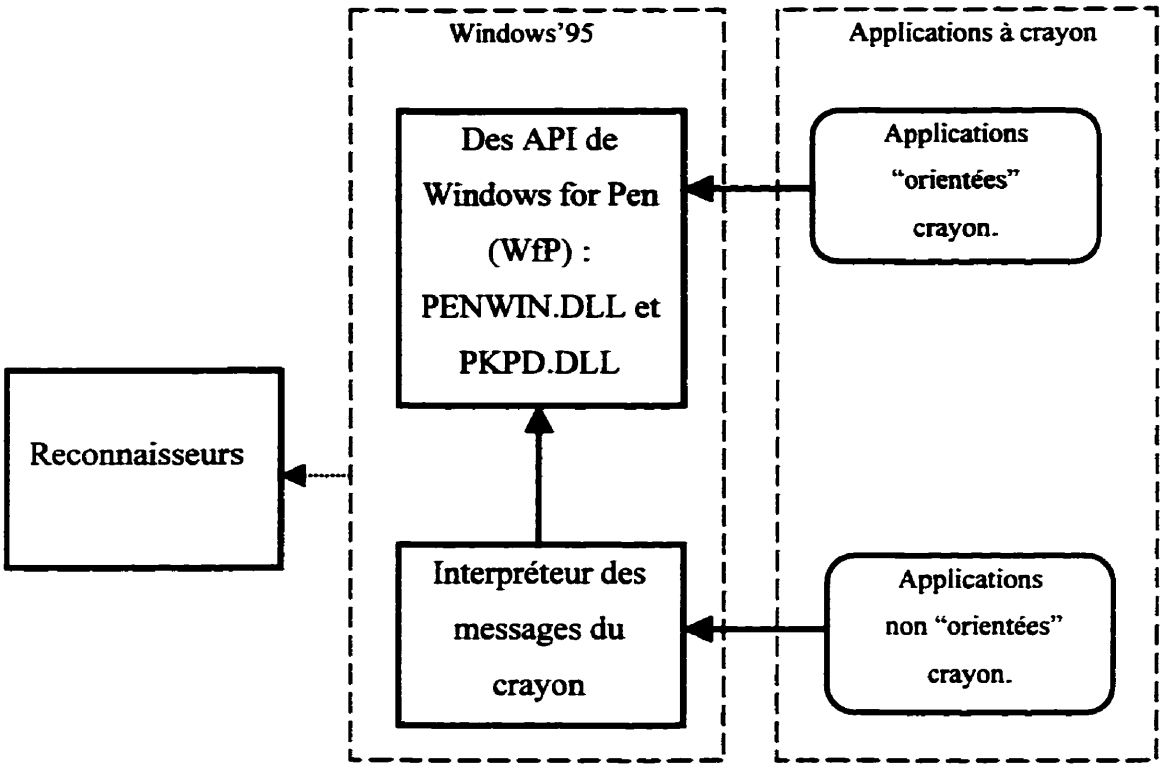


Figure 2.2 Schéma- bloc représentant les applications "orientées" crayon et non "orientées" crayon dans un environnement Windows.

L'interpréteur de messages crayon est une composante de Windows. Il intercepte les messages, détermine la provenance de ces messages : crayon, souris ou clavier. Si ces messages proviennent du crayon, ils sont transmis au reconnaisseur. Ce dernier interprète alors les tracés soit comme des caractères, soit comme des symboles, soit comme des gestes et renvoie ensuite, ses réponses à l'interpréteur. L'interpréteur transmet alors les messages Windows correspondants, c'est-à-dire des messages souris ou clavier, qui peuvent être traités par l'application non "orientées" crayon.

L'interpréteur de messages crayon ne peut transmettre des messages qu'à un seul reconnaisseur, celui installé comme le reconnaisseur du système.

L'exécuteur des commandes gestuelles utilisé dans le projet est Microsoft Word'97 - une application non "orientées" crayon.

2.3 La portée des reconnaisseurs

Selon leur portée il existe deux types de reconnaisseurs : global et local. Un reconnaisseur[3] est appelé global (ou reconnaisseur du système ou reconnaisseur par défaut) s'il est connu par le système d'exploitation – Windows'95. Le fonctionnement du reconnaisseur du système obéi à la philosophie du fonctionnement d'un système crayon tel que définie par Microsoft. De plus la création d'un reconnaisseur de système est aussi spécifiée. Pour bâtir un tel reconnaisseur, il faut utiliser les API de WfP ce qui revient à :

- Utiliser des objets prédéfinis comme le pointeur vers le contexte de reconnaissance (HRC), le pointeur vers le résultat de reconnaissance (HRCRESULT), etc. Ces objets sont utilisés pour stocker l'encre électronique.
- Exporter onze fonctions différentes comme : initialiser le reconnaisseur (ConfigRecognizer), ajouter des nouvelles données du crayon (AddPenInput), etc. C'est à l'aide de ces fonctions que l'accès au reconnaisseur est possible.
- Compiler le reconnaisseur comme une librairie de type DLL.

---

[3] La deuxième version des API pour crayon permet d'installer plusieurs reconnaisseurs et de les utiliser d'une façon sélective. Chaque reconnaisseur doit être spécialisé dans la reconnaissance d'un sous-ensemble de symboles particuliers au lieu d'essayer de traiter tous les types.

Le système se charge également de la détection de l'entrée crayon pour les applications non "orientées" crayon et de l'automatisation de la communication entre cette application et le reconnaisseur du système (voir la figure 2.1). Ainsi, on voit que dans un tel système tout est spécifié et étroitement lié au système d'exploitation Windows'95. La création d'un reconnaisseur du système doit obéir aux règles très rigides et son activation est automatisée.

Un reconnaisseur local est actif dans le cadre d'une application particulière. Sa création est guidée par le choix et les besoins du concepteur de l'application.

2.4 Solution du problème de cohabitation entre le reconnaisseur local et le reconnaisseur du système

Pour les besoins spécifiques du présent projet, nous avons créé un reconnaisseur de commandes gestuelles de type local. Ce reconnaisseur devait cohabiter avec le reconnaisseur du système, parce qu'on ne voulait pas éliminer ce dernier. Comme il a déjà été expliqué précédemment, le processus de traitement de l'encre électronique est entièrement pris en charge par WfP et plus précisément par son système d'interception des messages. La nature du fonctionnement de l'interpréteur de messages livré avec Windows'95 pour crayon porte à croire que l'interpréteur de messages est implanté sous forme d'un pilote de très bas niveau (Perugino, 1997). Le principe de l'interpréteur repose sur la détection du type de pointeur associé à la souris (sablier, pointeur de souris habituel, pointeur d'édition). Lorsque il s'agit d'un pointeur d'édition, l'interpréteur le remplace par un crayon. L'interception des messages crayon ainsi que l'affichage de l'encre sont alors pris en charge par différentes parties de WfP (Perugino, 1997). Ce mécanisme permet l'utilisation du crayon dans toutes les applications qui présentent une fenêtre d'édition. La détection du pointeur d'édition entraîne alors l'activation automatique des systèmes de génération d'encre et de reconnaissance du système (Perugino, 1997).

La figure 2.3 présente le cheminement normal des messages pour une application non "orientée" crayon. Les chiffres sont utilisés pour décrire la chronologie :

- 1 - l'interpréteur des messages de Windows détecte les messages provenant du crayon,
- 2 - les données du crayon sont redirigées vers le reconnaisseur du système qui interprète l'encre électronique comme des caractères, symboles ou gestes,
- 3 - le reconnaisseur envoie ses résultats à l'interpréteur des messages,
- 4 - l'interpréteur des messages communique les résultats à l'éditeur de texte sous forme de messages clavier ou souris.

Il faut souligner que tout ceci se fait automatiquement.

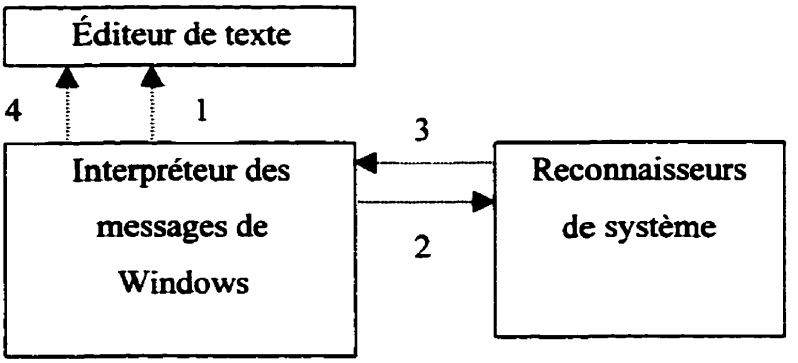


Figure 2.3 Schéma bloc représentant le cheminement des messages pour une application non "orientée" crayon.

La figure 2.4 inclut le système d'encrage qui n'est pas présenté à la figure 2.3. Le système d'encrage affiche le tracé du mouvement du crayon sur une surface réceptrice. L'encrage est pris en charge par des API de WfP et sa qualité est assurée par le producteur du matériel (Stylistic 1000). La figure 2.4 présente le cheminement normal des messages pour une application non "orientée" crayon. Cette figure contient le module de l'encrage. Les chiffres sont utilisés pour décrire la chronologie :

- 1 – l'interpréteur des messages de Windows détecte les messages provenant du crayon,

- 2 - l'interpréteur des messages de Windows active le système de l'encrage,
- 3 - le système de l'encrage génère encre électronique lors d'écriture,
- 4 - les données du crayon sont rédirigées vers le reconnaisseur du système,
- 5 - le reconnaisseur envoie le résultat à l'interpréteur des messages,
- 6 - l'interpréteur des messages communique le résultat à l'éditeur de texte sous forme de messages clavier ou souris.

Le système d'encrage est activé lors de la détection du pointeur d'édition.

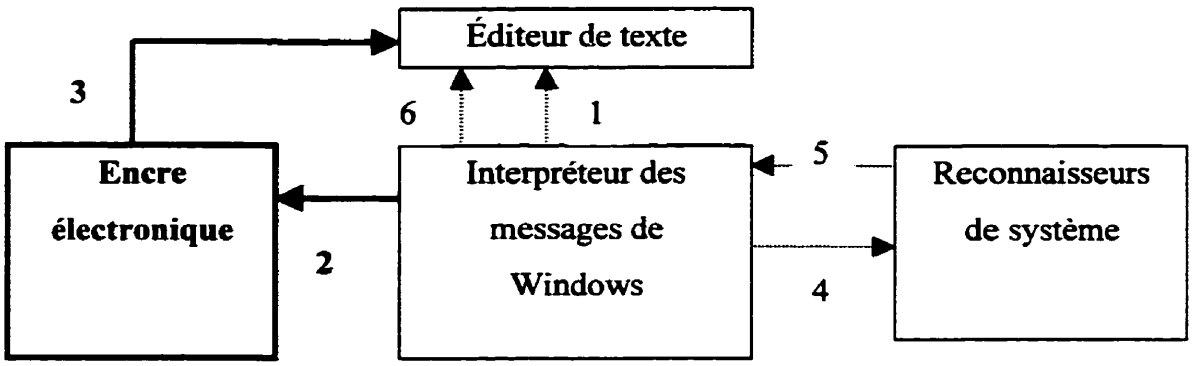


Figure 2.4 Schéma bloc représentant le cheminement normal des messages pour une application non "orientée" crayon, qui contient le module de l'encrage.

On peut constater que l'interpréteur des messages de Windows, l'encrage et le reconnaisseur forment un système fermé dans lequel il est très difficile de s'infiltrer. La figure 2.5 présente la séquence d'activation de différents modules qui participent à ce processus (figure 2.5 - les flèches non solides). Nous devons tester notre reconnaisseur de commandes gestuelles – reconnaisseur "maison", marqué en gras sur la figure 2.5. Lorsqu'on veut profiter des qualités de l'encrage de Stylistic 1000 (figure 2.5 - les flèches solides), un problème se pose : si on utilise l'encrage du système, c'est le reconnaisseur du système qui est appelé immédiatement après la visualisation et on perd ainsi le contrôle sur la reconnaissance. D'un autre côté, l'encrage est activé par l'interpréteur des messages de Windows. C'est pourquoi pour tester nos algorithmes de

reconnaissance, nous avons créé un système qui comprend notre intercepteur de messages et notre simulateur de l'encre électronique.

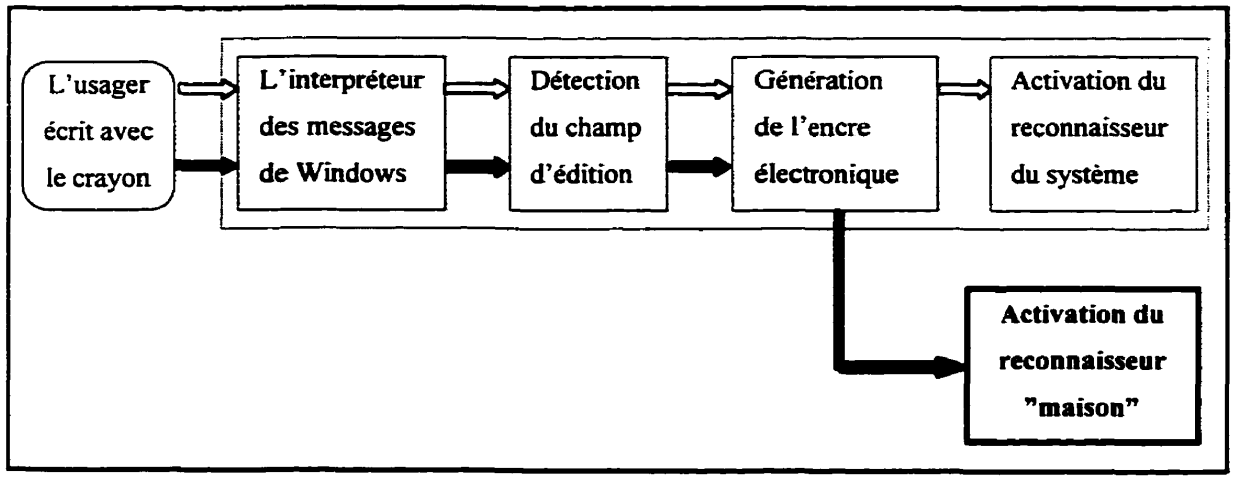


Figure 2.5 Schéma bloc illustrant l'activation du reconnaisseur du système.

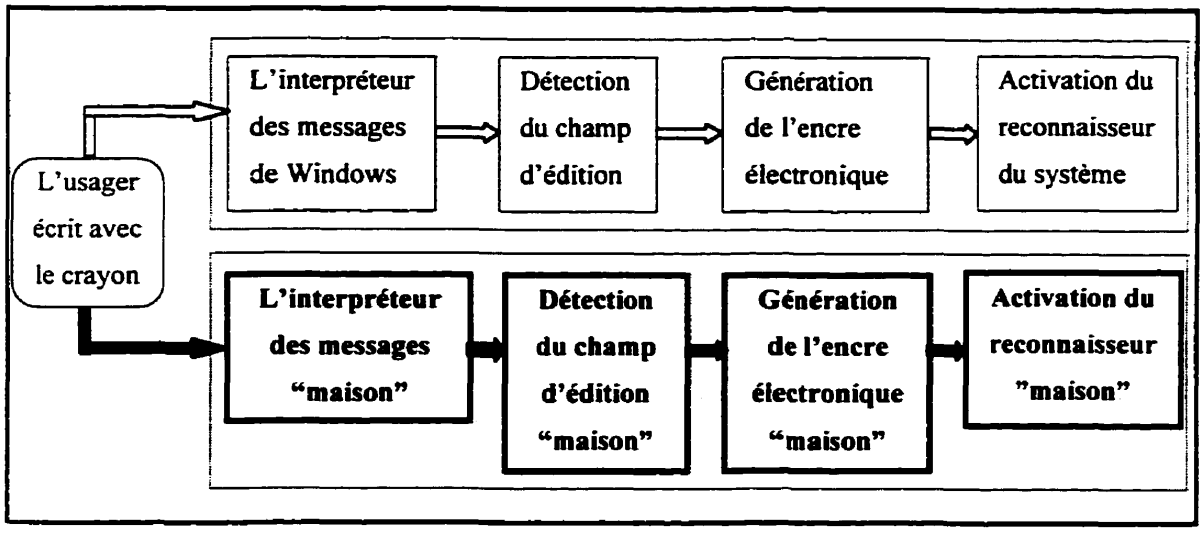


Figure 2.6 Schéma bloc de la solution au problème de cohabitation.

La figure 2.6 représente le schéma bloc de la solution apportée au problème de cohabitation de deux reconnaisseurs dans un bloc-notes électronique, utilisant WfP

(version 2.0). La chaîne commence par l'interpréteur des messages. Son rôle est de détecter l'utilisation du crayon pour l'édition du texte dans une application, non "orientée" crayon. Le crayon est alors perçu par une telle application comme une souris. La pointe du crayon correspond au bouton gauche de la souris et l'interrupteur[4] latérale du crayon correspond au bouton droit de la souris. L'intercepteur des messages "maison" doit détecter les trois situations suivantes (tableau 2.1 – première colonne): l'utilisateur commence à écrire, l'utilisateur écrit, l'utilisateur finit d'écrire. À chacune de ces situations correspond un événement souris : le bouton gauche de la souris est enfoncé, la souris bouge, le bouton gauche est relâché (tableau 2.1 – deuxième colonne). La détection de ces situations ou événements doit être suivie par des traitements appropriés, décrits dans la troisième colonne du tableau 2.1. On initialise les variables temporelles au début de chaque geste (rappelons qu'il s'agit de gestes d'une seule composante) et on active le processus de génération de l'encre électronique. Pendant l'écriture on collecte les données nécessaires pour nos algorithmes de reconnaissance. À la fin les données sont traitées par le reconnaisseur de commandes gestuelles "maison" et l'action associée est exécutée.

Tableau 2.1 Les situations qui devaient être détectées par notre intercepteur des messages

Situations	Événements souris	Traitements
L'utilisateur commence à écrire	Le bouton gauche de la souris est enfoncé	**A.** Initialisation des variables temporelles **B.** Début du processus de génération de l'encre électronique
L'utilisateur écrit	La souris bouge	Collecte des données nécessaires pour nos algorithmes de reconnaissance
L'utilisateur finit d'écrire	Le bouton gauche est relâché	**A.** Traitement des données par le reconnaisseur de commandes gestuelles "maison" **B.** Exécution de l'action associée à la commande gestuelle

[4] Le crayon du Stylistic 1000 est muni d'un interrupteur latérale.

2.5 Détails sur l'implantation de l'éditeur gestuel

La partie en gras de la figure 2.6 contient des modules requis dans la création de l'éditeur gestuel. Dans cette section, des explications seront donnés pour chacun de ces modules et sur leurs interactions.

2.5 1 La place de la technologie ActiveX

La création de l'éditeur gestuel est caractérisée par l'application d'une approche récente en programmation : la réutilisation de code existant. Cette stratégie de programmation été utilisée avec succès dans le développement de logiciels, et plusieurs concepts de réutilisation de code ont vu le jour. JavaBeans[5] de Sun et ActiveX (COM) de Microsoft sont les deux concepts les plus répandus. Dans le cadre de ce projet, on a utilisé la technologie ActiveX de Microsoft. La technologie ActiveX permet la réutilisation du code existant. Elle permet la coopération entre des composantes qui sont écrites en utilisant des langages différents de programmation orientés objet. Une composante ActiveX est une unité de code qui peut être exécutée dans des contextes différents grâce au fait que, lors de la création, son auteur a respecté certaines règles. Les composantes ActiveX sont des fournisseurs de services et/ou d'objets et elles peuvent collaborer dans le cadre d'une application. Il existe tout un marché de composantes ActiveX qui est très actif. La difficulté principale est de définir ce qu'on recherche et de le trouver. La recherche se fait généralement sur le Web.

---

[5] JavaBeans est le nom de la technologie qui permet la réutilisation du code existant développé par SUN.

2.5.2 Description des composantes

Les quatre modules en gras de la figure 2.6 correspondent aux quatre composantes ActiveX utilisées dans notre application. Le tableau 2.2 contient les noms et les extensions de toutes les composantes, utilisées.

Tableau 2.2 Les composantes ActiveX utilisées dans notre projet.

Bloc fonctionnel	Nom de la composante ActiveX	Extension
L'interpréteur des messages	Message Blaster	.OCX
Éditeur de texte	Microsoft Word'97	.EXE
Simulateur de l'encre électronique	KingS Screen	.OCX
Reconnaisseur des commandes gestuelles	Une composante que j'ai créée	.DLL

L'interpréteur des messages (Message Blaster) est un contrôle ActiveX. Un contrôle ActiveX est un élément qui a une interface standardisée et qui peut être ajouté dans la boite d'outils (ToolBox) dans l'environnement de développement de Visual Basic. En faisant ceci, le nouveau contrôle devient disponible dans l'application. Chaque contrôle a des propriétés et des méthodes qui constituent son interface. Les propriétés correspondent aux variables tandis que les méthodes correspondent aux procédures et aux fonctions. La seule façon d'utiliser un contrôle est d'exploiter ses propriétés et ses méthodes.

L'utilisation de l'interpréteur de messages est facile. Il faut spécifier quels sont les messages qui nous intéressent et de quelle fenêtre ils proviennent. La fenêtre cible dans notre application est l'éditeur de texte Word'97. Word'97 est une application non "orientée" crayon. Les messages provenant de la souris et du clavier sont ceux détectés

et traités par l'environnement Word. C'est pourquoi ceux qu'il faut surveiller sont les messages de la souris[6]. Parmi tous les messages de la souris (10 messages différents) il faut sélectionner ceux qui nous intéressent. Nous avons essayé d'utiliser notre intercepteur de messages pour guetter les mêmes messages que ceux de l'intercepteur de Windows. Mais cette approche n'a pas réussi, car l'intercepteur du Windows prenait le contrôle et activait le reconnaisseur du système avant même que l'intercepteur "maison" puisse s'en rendre compte. Pour cette raison, il été décidé de traiter des messages différents de ceux qui mènent vers l'activation du reconnaisseur par défaut. De cette manière, il devient possible de permettre à toutes les autres applications sur l'ordinateur Stylistic1000 de continuer d'utiliser le reconnaisseur du système. De plus au même moment dans notre application nous avons la liberté de tester et mettre au point nos propres outils de reconnaissance. Dans une application non "orientée" crayon, les messages traités par le reconnaisseur du système sont les messages provenant de la souris et plus précisément les messages provenant du bouton gauche de la souris. Notre interpréteur de messages intercepte des messages provenant du bouton droit de la souris (ce bouton existe également sur le crayon). Alors que si on écrit avec le crayon en poussant sur l'interrupteur latérale, on évite l'activation du reconnaisseur du système et on peut accueillir l'information nécessaire pour nos algorithmes de reconnaissance. On peut détecter quand la souris est sur le document édité et quand elle bouge à condition que le bouton droit demeure enfoncé. Les messages de la souris à observer sont par conséquent, les suivants : le bouton droit de la souris est enfoncé (RBUTTONDOWN), la souris est déplacée (MOUSEMOVE), le bouton droit de la souris est relâché (RBUTOONUP).

A l'aide du contrôle KingS Screen l'encre électronique est simulée. On l'utilise pour dessiner les commandes gestuelles et pour capturer l'information directement de l'écran de l'ordinateur. Il permet aussi de spécifier l'épaisseur et la couleur de notre crayon.

---

[6] Le bouton droit et le bouton gauche de la souris sont simulés par l'interrupteur latérale du crayon et la pointe du crayon.

Le reconnaisseur de commandes gestuelles sera expliqué en detail dans le quatrième chapitre.

L'exécuteur de commandes gestuelles – Word'97 est aussi une composante ActiveX. Cette composante ActiveX peut être lancée, contrôlée et fermée à partir de notre application Visual Basic. Le contrôle est possible parce qu'elle expose un ensemble riche d'objets et de méthodes. C'est en exécutant ses méthodes sur ses objets que l'exécuteur de commandes gestuelles a été réalisé. Le fonctionnement de notre application et l'interaction entre les composantes sont décrits à l'aide d'un algorithme présenté par la figure 2.7.

Dans ce chapitre les démarches entreprises lors de la conception de l'éditeur gestuel ont été décrites ainsi que plusieurs détails techniques sur ce projet. On a rencontré des problèmes liés à la complexité de développement d'une application à crayon dans l'environnement Windows et plus spécifiquement à l'adaptation d'un éditeur de texte commercial aux besoins de l'entrée crayon et on a présenté notre solution aux ces problèmes.

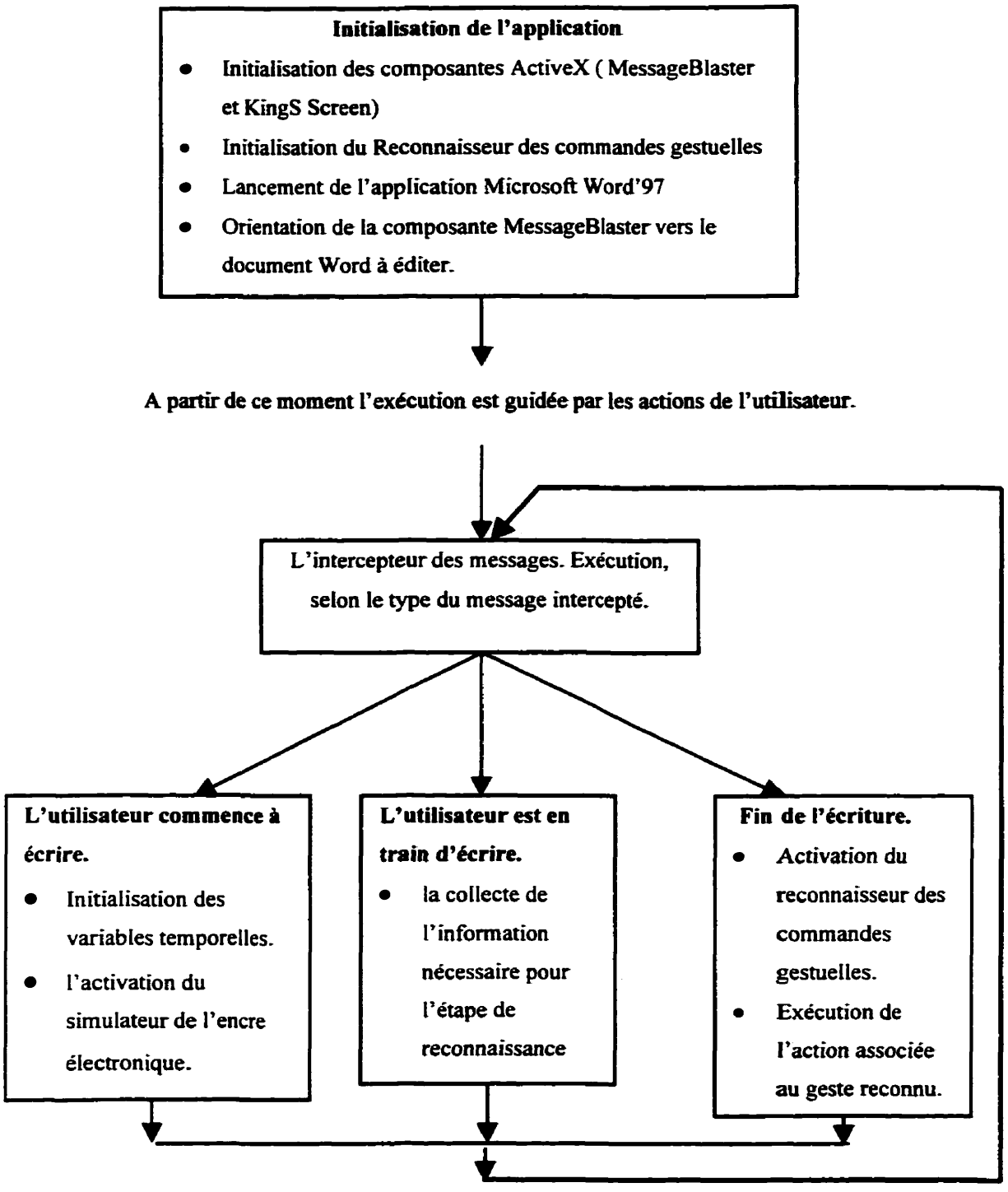


Figure 2.7 Interaction entre les blocs fonctionnels utilisés par notre éditeur gestuel.

CHAPITRE III
LE CHOIX DES GESTES

Ce chapitre présente l'ensemble des commandes gestuelles que nous avons choisi. Rappelons qu'il a été nécessaire de définir des gestes dans le but de permettre l'évaluation de nos algorithmes de reconnaissance ainsi que pour tester le fonctionnement de notre éditeur gestuel. Le lecteur trouvera ici les résultats d'une recherche bibliographique sur les commandes gestuelles existantes. Le nombre important de publications témoignent l'intérêt croissant tant au niveau de la recherche qu'au niveau des industries. Dans la section 3.1 on présentera un premier ensemble de commandes gestuelles qui sera référencé dans le chapitre suivant. Dans la section 3.2 on définit l'ensemble des gestes qui sont implantés dans la version finale de l'éditeur gestuel.

3.1 Revue des commandes gestuelles proposées par des chercheurs et des compagnies

Les commandes gestuelles sont perçues comme l'une des nouvelles techniques d'interaction homme-ordinateur. Ces dernières années, plusieurs prototypes de traitement de l'information par ordinateurs, guidés par des commandes gestuelles, ont été réalisés et analysés. Le champ d'application est vraiment vaste : Suenaga et Nagura (1980), Brault, Plamondon et Laframboise (1995) ont utilisé des commandes gestuelles pour éditer du texte manuscrit, Coleman (1969), Wolf et Morrel-Samuels (1987) Kankaanpaa (1988), Welbourn et Whitrow (1990), et Rubin (1991) pour l'édition de texte typographique , Kim (1988) a testé un tableur guidé par des gestes, Dimitriadis et Coronado (1995), Finkelstein (1991) ont créé un éditeur de formules mathématiques gestuel, Aimé-Desiré (1995) a édité des composants électroniques à l'aide de

commandes gestuelles, Welbourn et Whitrow (1990) ont fait un éditeur de diagrammes gestuel et Rubin (1991) a utilisé des gestes pour l'édition de musique.
Dans le rapport technique : "Commandes gestuelles utilisées pour l'édition de texte à l'aide d'un bloc-notes électroniques : État de l'art", de Sabeva, Brault et Plamondon (1997) sont résumés les résultats d'une recherche bibliographique sur les commandes gestuelles déjà utilisées dans le contexte d'édition de texte à partir de 1956 jusqu'à 1996 par des chercheurs et des compagnies. La figure 3.1 montre les 53 gestes répertoriés.

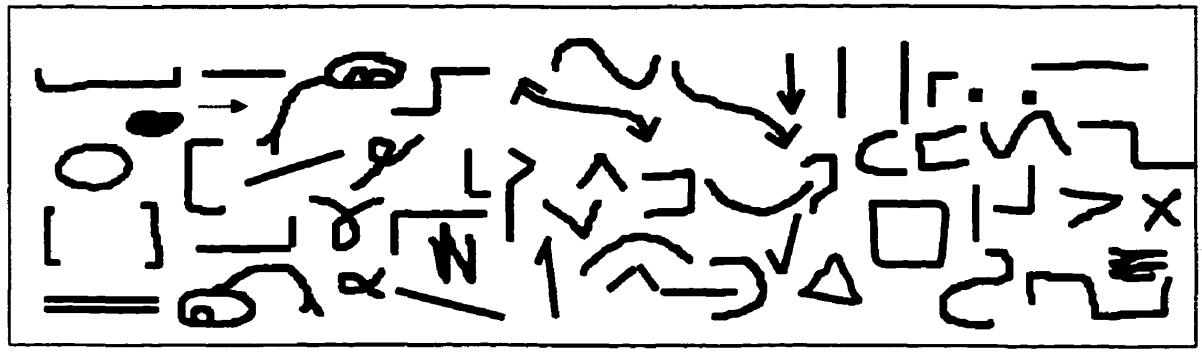


Figure 3.1 L'ensemble de 53 gestes répertoriés.

Les tableaux 3.1 à 3.5 présentent les résultats de la recherche bibliographique sous une autre forme. L'information dans les tableaux est organisée de la façon suivante :

- les chercheurs et les compagnies sont présentés en ordre chronologique ;
- les fonctions de base telles que sélectionner, effacer, insérer, etc., sont mises au début ;
- la comparaison des représentations graphiques choisies par différents chercheurs (compagnies) pour la même fonction est facilitée - il suffit d'observer une ligne.

3.1.1 Revue des commandes gestuelles proposées par des chercheurs

Cinq groupes de chercheurs ont proposé des ensembles de commandes gestuelles pour éditer du texte présenté en ordre chronologique dans le tableau 3.1. On retrouve Coleman (1969), Suenaga et Nagura (1980), Kankaanpaa (1988), Welbourn et Whitrow (1990), Brault, Plamondon et Laframboise (1995) ainsi que certains membres du laboratoire Scribens dont Sabeva, Brault et Plamondon (1998).

Tableau 3.1 Commandes gestuelles proposées par des chercheurs.

Nom de la fonction	**Coleman** *1969*	**Suenaga et Nagura** *1980*	**Kankaa-npaa** *1988*	**Welborn et Whitrow** *1990*	**Laframboise. Brault et Plamondon** *1995*	**Sabeva, Brault et Plamondon** 1998
1.Sélectionner					1.  2.  3. 	
2.Effacer	Je veux effacer le mot ~~concours~~.			Acad~~émie~~		
3.Inserer -un caractère -un mot -plusieurs mot - une ligne						
4. Echanger						
5.Lier						
6. Insérer espace						
7. Nouvelle ligne		1.nouvelle ligne ; 2.nouveau paragraphe :  3.nouvelle page :				
8.Lier deux lignes						
9.Copier (- copy)						
10.Déplacer (- move)						

Tableau 3.1 (suite)

Nom de la fonction	**Coleman** *1969*	**Suenaga e Nagura** *1980*	**Kankaan-paa** *1988*	**Welborn et Whitrow** *1990*	**Laframboise. Brault et Plamondon** *1995*	**Sabeva, Brault et Plamondon** *1998*
11.Confirmer						
12.Effacer la sélection	[/ /]					
13. Monter le texte						
14. Descendre le texte						
15. Effacer un bloc du texte	Si on veut effacer ce paragraphe il faut utiliser un geste.					
16. Positionne						
17.Demander aide						
18.Insertion avec un menu						
19.Changeme des attributs						

Les chercheurs essayaient de définir un ensemble de commandes gestuelles afin de couvrir les situations que leur éditeur de texte pouvait rencontrer. Les commandes gestuelles proposées récemment sont habituellement simples car le but est en réalité de trouver un ensemble minimal qui permette d'augmenter le taux de reconnaissance, de diminuer le temps de réponse et de réduire la quantité d'informations que l'utilisateur doit mémoriser. Quand on parle de gestes simples, habituellement on fait référence à un très petit nombre de levés de stylo. Il important de remarquer que les formes appartenant

à un même ensemble de commandes gestuelles soient faciles à distinguer (Wolf, Morrel-Samuels, 1987). L'environnement crayon regroupe tout ce qui est lié à l'utilisation du crayon. Les gestes écrits, comme l'écriture elle-même font partie de l'environnement crayon. L'écriture est le résultat du développement personnel et, par conséquent, est liée à la personnalité et est donc très variée. On trouve la même diversité dans l'écriture des commandes gestuelles. Cette diversité est observée à deux niveaux: les gens utilisent différents gestes pour la même fonction et les gens écrivent un geste différemment. Une façon d'accommoder un nombre plus grand d'utilisateurs est en définissant de multiple gestes pour la même fonction.

En observant le tableau 3.1 on remarque qu'il existe trois fonctions pour lesquelles tous les chercheurs ont défini des commandes gestuelles. Ce sont les fonctions : Sélectionner, Effacer et Insérer. Le tableau 3.2 montre leurs formes. Il y a 5 gestes différents pour Sélectionner et Effacer et deux gestes pour la fonction Insérer. On peut donc déduire que l'importance des gestes diffère. Il semble que les gestes qui correspondent aux fonctions fréquentes tentent d'être couverts par plusieurs formes.

Tableau 3.2 Les gestes pour les fonctions : Sélectionner, Effacer et Insérer.

Sélectionner					
Effacer					
Insérer					

Le tableau suivant (tableau 3.3) représente une statistique sur le nombre de fonctions et le nombre de gestes à reconnaître (différentes formes) par chercheur.

Tableau 3.3 Nombre de gestes et de fonctions par groupe de chercheurs.

	Coleman *1969*	Suenaga et Nagura *1980*	Kankaa-npaa *1988*	Welborn et Whitrow *1990*	Laframboise Brault et Plamondon *1995*	Sabeva, Brault et Plamondon *1998*
Nombre de gestes	**13**	**12**	**9**	**6**	**10**	**13**
Nombre de fonctions	**10**	**12**	**10**	**6**	**8**	**10**

Certains chercheurs ont défini différents gestes pour une même fonction selon les entités textuelles sur lesquelles la fonction est appliquée. Le tableau 3.4 présente un exemple de redondance dans la définition de la fonction insérer. On y voit onze gestes différents pour l'insertion des cinq objets textuels. Les commandes gestuelles sont les mêmes pour l'insertion d'un mot ou d'un caractère. Par contre, des formes différentes sont utilisées pour l'insertion d'un espace, du signe "nouvelle ligne" et du signe "nouveau paragraphe". En effet les trois derniers gestes d'insertion sont plutôt des actions de mise en page.

Tableau 3.4 Exemple de redondance dans la définition de la fonction "Insérer".

La fonction Insérer	Le geste
Insérer **un caractère**	
Insérer **un mot**	
Insérer **un espace**	
Insérer le signe "nouvelle ligne"	
Insérer le signe "nouveau paragraphe"	

Le tableau 3.1 démontre bien qu'un geste peut être utilisé pour des tâches (fonctions) différentes. Par exemple :

A. Un trait vertical est utilisé pour :

- insérer un espace par Welbourn et Whitrow (1990), Brault, Plamondon et Laframboise (1995), Sabeva, Brault et Plamondon (1998);
- nouvelle ligne par Coleman (1969);
- sélectionner par Welbourn et Whitrow (1990). Dans ce cas il fait partie d'un geste plus complexe;
- positionner par Suenaga et Nagura (1980).

B. Un trait horizontal est utilisé pour:

- effacer un mot par Coleman (1969), Suenaga et Nagura (1980), Kankaanpaa (1988);
- sélectionner par Brault, Plamondon et Laframboise (1995);
- lier deux mots par Welbourn et Whitrow (1990);
- positionner par Suenaga et Nagura (1980).

C. Les flèches peuvent être utilisées comme

- des gestes autonomes :
  - nouvelle ligne par), Welbourn et Whitrow (1990).
- partie de gestes plus complexes :
  - copier par Suenaga et Nagura (1980), Brault, Plamondon et Laframboise (1995);
  - déplacer par Suenaga et Nagura (1980), Brault, Plamondon et Laframboise (1995).

D. Le geste "parenthèse" est utilisé pour :

- sélectionner par Coleman (1969), Brault, Plamondon et Laframboise (1995). Dans ce cas il fait partie d'un geste plus complexe.
- nouvelle ligne par Sabeva, Brault et Plamondon (1998);
- lier deux lignes par Kankaanpaa (1988);

- effacer la sélection par Coleman (1969). Dans ce cas il fait partie d'un geste plus complexe.

Il est important de noter que la direction d'écriture d'un geste peut changer son interprétation. Chez Coleman (1969) la direction d'écriture du trait vertical a deux interprétations différentes : monter ou descendre le texte. Chaque concepteur de logiciel analyse les besoins, extrait les fonctionnalités que son application doit offrir aux utilisateurs. Après, pour chaque fonction, il choisit la réalisation adéquate - menu, commande alphanumérique ou commande gestuelle. De cette façon, il crée un groupe de fonctions pour lesquelles il définit un ensemble de commandes gestuelles. Les ensembles de commandes gestuelles diffèrent selon le domaine d'application. Concernant les formes de gestes, il ne semble pas exister une définition précise sur les formes de gestes comparativement à celles existantes pour les lettres et les chiffes ; tout le monde s'entend pour dire que les gestes sont des formes simples et faciles à écrire; il faut laisser le choix des formes à l'utilisateur.

3.1.2 Revue des commandes gestuelles proposées par des compagnies

L'avancement dans la technologie du micro-ordinateur a rendu possible un changement révolutionnaire dans l'utilisation des ordinateurs. Plusieurs fabricants ont mis sur le marché des ordinateurs qui utilisent le crayon comme seul dispositif d'entrée. Il faut distinguer trois types de compagnies reliées à la production de bloc-notes électroniques.

- Compagnies qui font la production du matériel.
- Celles qui font les systèmes d'exploitation conçus pour le support du crayon.
- Celles qui font les logiciels de reconnaissance de caractères, de commandes gestuelles, d'objets graphiques et de signatures.

Ces compagnies doivent collaborer pour assurer la qualité du produit final. Par exemple, les compagnies qui font la reconnaissance de gestes, travaillent sur un ensemble de commandes gestuelles, créé par une compagnie du deuxième niveau. Le bloc-notes électronique, utilisé dans le projet - Stylistic1000 est mis sur le marché par Fujitsu Personal Systems. Alors Fujitsu Personal Systems est le nom de la compagnie du premier niveau. Le système d'exploitation pour le support du crayon est Windows pour crayon de Microsoft. Microsoft est le nom de la compagnie du deuxième niveau et CIC est la compagnie qui a créé les reconnaisseurs du système. Stylistic 1000 possède un système de reconnaissance de caractères et un système de reconnaissance de commandes gestuelles. Le Tableau 3.5 contient une brève description des ensembles de commandes gestuelles implantées dans des systèmes commerciaux. Ils sont présentés en ordre de leur création. Ce sont les produits suivants :

- PenPoint, GO Corporation (1991);
- Pen computing, Microsoft - implanté par Fujitsu - Stylistic1000 (1991);
- PenScheduler, Slate corporation (1991);
- Newton, Apple (1993).

Tableau 3.5 Commandes gestuelles proposées par des compagnies.

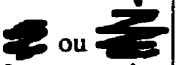
Nom de la fonction	**PenPoint** GO Corporation *1991*	**Windows for Pen** Microsoft *1991*	**Pen Scheduler** Slate Corporation *1991*	**Newton** Apple *1993*
1. Sélectionner	1.En utilisant le crayon comme une souris. 2. Le geste : [ ]	1. En utilisant le crayon comme une souris(en surlignant) **maman**	1. En surlignant : **maman** 2. Le geste : 	1.En utilisant un marqueur de sélection. 2. Le geste : 
2. Effacer	1. Un mot : -  -  2. Caractère : 	A.1.Selectionner 2  Le résultat est mis dans Clipboard. B. Backspace. pou effacer un caractère. C.1.Sélectionner. 2. **frigidaire** D.1 Un mot ~~saladier~~	A.1. Sélectionner 2. ou  B. En utilisant une boite d'édition. ~~Karaté~~ ou ~~box~~ 	1. texte ou   ou 2. une seule lettre 
3. Insérer ligne	1. 2. Ouvrir une tablette d'écriture.	Le geste : Ex. La marmotte s'est cachée. Résultat : La marmotte s'est cachée.		1. Une ligne  2. Plusieurs lignes  3. Interrompre la ligne courant est insère un espace pour plusieurs lignes. 
4. Insérer une tabulation		Le geste : Ex. Je vais à l'école Je vais à l'école		
5. Insérer un espace	Le geste : 	Le geste : 	Les gestes :  et	1.Pour une lettre :  2.Pour un mot : 

Tableau 3.5 (suite)

Nom de la fonction	**PenPoint** GO Corporation *1991*	**Windows for Pen** Microsoft *1991*	**Pen Scheduler** Slate Corporation *1991*	**Newton** Apple *1993*
7.Changement d'échelle (Zoom in, out)			1. Sélectionner 2. Dessiner un trait par haut ou par la bas.	
8.Navigation	La navigation en quatre directions est définie.			
9.Edition du texte		√	1. Sélectionner 2. Le geste : √ 3. Une tablette d'édition est ouverte.	Texte à éditer
10. Annuler la dernier action(Undo)				
11. Étendre la sélection				
12. Vérifier	√			
13.Copier	En utilisant le crayon comme une souris.	1.Sélectionner. 2.Le geste 3.La sélection est mise dans la mémoire temporelle	1.Sélectionner. 2.Le geste 3.La sélection est mise dans la mémoire temporelle	1.Sélectionner. 2.Double clic
14.Déplacer	En utilisant le crayon comme une souris.	1.Copier 2.	1.Sélectionner. 2,Appuyer. 3.Glisser à l'endroit désiré	1.Sélectionner. 2,Appuyer. 3.Glisser à l'endroit désiré..

Go Corporation est la première compagnie qui a développé un système d'exploitation, appelé PenPoint, dédié au crayon. Microsoft a créé son propre système d'exploitation - Windows pour crayon (Windows for Pen). Slate Corporation et Fujitsu développent leurs produits, PenScheduler et Stylistic 1000 respectivement, sur la base de Windows pour crayon de Microsoft. Tous ces produits ont une caractéristique spécifique, c'est qu'elles offrent la possibilité d'utiliser des commandes gestuelles. Présentement, tous les systèmes commerciaux utilisent des ensembles de commandes gestuelles basés sur le

choix du concepteur (utilise en ensemble de geste choisi par les concepteurs du systèmes). Certains offrent l'option d'utiliser le clavier virtuel (soft keyboard), mais pour tous, le crayon est le dispositif d'entrée principal. Stylistic 1000 possède un système de reconnaissance de caractères et un système de reconnaissance de commandes gestuelles. L'usager peut aussi choisir d'utiliser une commande gestuelle ou une lettre encerclée (voir le tableau 3.6).

Tableau 3.6 Liste de commandes qui peuvent être invoquées à l'aide d'une lettre encerclée.

	Copier
	Effacer (Delete)
	Menu
	Nouvelle ligne
	Insérer (Paste)
	Espace
	Tabulation
	Couper(Cut)
	Insérer
	Palette
	Lasso

Il est possible de définir d'autres commandes en utilisant les lettres qui restent et en les encerclant.

En observant le tableau 3.5, on s'aperçoit qu'un geste peut être utilisé pour des tâches (fonctions) différentes. Par exemple :

A. Un trait vertical est utilisé pour :

- naviguer dans la direction haut-bas (bas-haut) selon la direction de l'écriture du trait dans PenPoint (1991);
- changement d'échelle (zoom-in, zoom-out) selon la direction de l'écriture du trait dans PenScheduler (1991).

B. Un trait horizontal est utilisé pour:

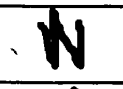
- effacer un mot dans PenScheduler (1991), Windows for Pen (1991);
- naviguer dans la direction gauche-droite (droite-gauche) selon la direction de l'écriture du trait dans PenPoint (1991);

C. Le geste ✓ (check mark) est utilisé pour:

- lancer une vérification d'orthographe dans PenPoint (1991);
- ouvrir une tablette d'édition dans Windows for Pen (1991).

Dans ces systèmes, on observe neuf différents gestes pour la fonction "effacer". Le tableau 3.7 présente leurs formes (première colonne) et les objets sur lesquels ils sont exécutés (deuxième colonne). Les "X" signifient que la compagnie correspondante (première ligne, en gras) utilise le geste et l'exécute sur l'objet respectif. On remarque que Pen computing (1991) de Microsoft a quatre gestes pour la fonction "effacer" parmi lesquels trois peuvent être utilisés pour effacer un mot. PenPoint (1991), PenScheduler (1991), Newton (1991) ont eux aussi défini trois gestes pour effacer.

Tableau 3.7 Les gestes utilisés pour la fonction "Effacer"

Geste pour effacer	Entité textuelle	**PenPoint** GO Corporation *1991*	**Windows for Pen** Microsoft *1991*	**Pen Scheduler** Slate Corporation *1991*	**Newton** Apple *1993*
	Mot	X		X	
	Mot	X			X
	Caractère	X			
	Mot		X	X	
	Caractère		X		
	Mot		X		
	Mot		X	X	
	Mot, Phrase				X
	Caractère				X

Dans tous les systèmes, la direction de l'écriture des gestes est très importante pour l'interpretation du geste. Dans les systèmes commerciaux il y a aussi une autre particularité : le crayon électronique est aussi utilisé comme souris pour sélectionner et positionner. Dans Windows pour crayon il y a un geste réservé à la fonction "Annuler la dernière action" (Undo). PenPoint (1991) utilise également des commandes gestuelles pour naviguer dans le document édité.

La recherche bibliographique et nos analyses ont permis de définir les critères suivants (ou guides) de design : gestes simples – un levé du crayon, si possible; formes distinctives dans le cadre de l'ensemble de commandes; formes intuitives et facile à se rappeler.

3.2 Sélection d'un ensemble de commandes gestuelles

L'interface du logiciel de traitement de texte Word'97 constitue le "visage " de l'éditeur de texte gestuel. Notre éditeur comporte une série de commandes qui se divise en deux classes : les commandes gestuelles et les commandes par pointage. Les dernières sont accessibles à partir des menus de Word'97. Nous ne nous arrêterons pas à ces fonctions, puisqu'elles sont généralement les mêmes chez les outils de traitement de texte standards. Les fonctions qui nous intéressent sont associées principalement au concept d'édition et peuvent être évoquées en dessinant une forme graphique à l'aide d'un crayon.

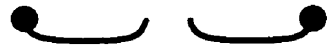
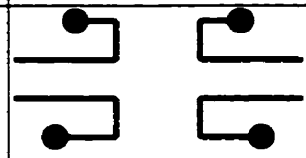
3.2.1 Définition des commandes manuscrites à implanter

Voici les huit fonctions de l'éditeur associées aux commandes gestuelles :

- Effacer. Cette fonction sert à effacer l'objet textuel sélectionné. La sélection se fait en utilisant le crayon comme une souris, qui est la même technique utilisée avec des outils de traitement de texte standards. De cette façon, on peut effacer un caractère, un mot, une phrase, un paragraphe, etc. Il faut premièrement sélectionner l'objet à effacer et dessiner ensuite le geste correspondant (Tableau 3.8).
- Insérer. Cette fonction permet l'insertion de texte à un endroit précis. La spécification de l'endroit se fait à l'aide d'une technique "souris". Le dessin de la commande est suivi par l'ouverture d'une palette d'édition. L'utilisateur écrit le texte à insérer en utilisant le crayon, le texte manuscrit est converti en texte ASCII par le reconnaisseur des caractères du système et affiché dans la palette. Une fois approuvé, il est inséré dans le document ouvert.

- Disjoindre. Cette commande gestuelle sépare en deux une entité textuelle considérée comme un mot. Il faut que la place de séparation soit indiquée à l'avance.
- Joindre. Cette commande lie deux mots en un seul. Le curseur doit être positionné entre les deux mots avant d'écrire le geste.
- Nouvelle ligne. Ce geste insère un signe "Nouvelle ligne" après la position du curseur en créant ainsi un nouveau paragraphe.
- Palette d'édition. Cette commande gestuelle permet l'édition manuscrite d'une portion de texte. Il faut d'abord sélectionner le texte à éditer et dessiner le geste après. Cette fonction permet à l'usager d'introduire un nouveau mot dans le dictionnaire, rappelons que dans le présent projet, il s'agit principalement d'éditer des vieux documents donc on peut s'attendre à un lexique particulier.
- Annuler la dernière action (Undo). Cette commande permet à l'usager d'annuler la dernière action exécutée. Il est possible de l'exécuter plusieurs fois et d'annuler une séquence d'actions d'édition.
- Image braille. Cette commande n'est pas associée à une action d'édition. Elle permet le lancement du module de reconnaissance de graphèmes braille. Elle est donc une commande "générale". Le processus d'édition habituellement commence par l'évocation de cette commande. Le résultat de son exécution est l'affichage du texte ASCII correspondant, et l'usager peut alors commencer l'édition gestuelle.

Tableau 3.8 Notre ensemble de commandes gestuelles.

Nom des fonctions	Signes	Remarques
Effacer		Il faut d'abord sélectionner l'entité textuelle à effacer.
Insérer		Il faut d'abord positionner le curseur.
Disjoindre		Il faut d'abord positionner le curseur.
Joindre		Le curseur doit être positionner entre les deux mots avant d'écrire le geste.
Nouvelle ligne		Il faut d'abord positionner le curseur.
Palette d'édition		Il faut d'abord sélectionner l'entité textuelle à éditer.
Annuler la dernière action		
Image braille		

Les commandes définies dans le tableau 3.8 doivent être reconnues par le reconnaisseur de commandes gestuelles. Notons de plus que notre but est également de permettre à l'utilisateur de choisir des formes avec lesquelles il sera plus à l'aise et ainsi créer un éditeur gestuel adaptable.

CHAPITRE IV
LE RÉSEAU ARTMAP POUR LA RECONNAISSANCE DES COMMANDES GESTUELLES

Dans ce chapitre notre solution au problème de reconnaissance de commandes gestuelles est décrite. Rappelons que le but recherché est de créer un reconnaisseur de formes fiable, compact, rapide et adaptable. Plusieurs méthodes de reconnaissance de formes sont alors à considérer. Elles peuvent être divisées en deux groupes : les méthodes d'inspiration symbolique et les méthodes dites connexionnistes. Le choix d'une méthode est normalement guidé par la nature du problème à résoudre.

La section 1 traite la justification du recours à l'approche connexionniste pour la reconnaissance des commandes gestuelles, alors qu'à la section 2 des explications sont données sur le réseau – ART1. La troisième section est entièrement consacrée à la nature des données et à leur prétraitement. Ici on introduira le code Isométrique qui a permis la classification de signaux spatio-temporels par le réseau ART1. Nous verrons aussi quels sont les avantages d'utiliser ce code lors de l'emploi du réseau ARTMAP. La quatrième section comporte la description du réseau ARTMAP : ses mécanismes particuliers et les algorithmes de simulation utilisés. La cinquième section explique la signification du mot "adaptable" dans le contexte de l'édition gestuelle et comment l'adaptabilité est réalisée dans notre projet. La dernière section contient un compte rendu du chapitre. Les expérimentations seront présentées dans le cinquième chapitre.

4.1 Propriétés intéressantes des machines neuronales pour la reconnaissance des commandes gestuelles

L'approche par réseaux de neurones artificiels (RNA), aussi appelée approche connexionniste a connu un développement remarquable au cours des dernières années, à tel point qu'elle est maintenant devenue une technique importante en reconnaissance de formes et dans d'autres domaines.

4.1.1 Machines neuronales (MN) – notions

Les réseaux de neurones sont essentiellement des processeurs distribués qui fonctionnent en parallèles. Ils ont l'habileté de mémoriser des connaissances sur le monde extérieur et de les rendre facilement accessibles. Ils ressemblent au cerveau en ce qui concerne ces deux aspects :

- Les connaissances sont acquises à partir d'un apprentissage;
- Les connexions entre des neurones sont utilisées pour stocker ces connaissances.

Donc l'unité fondamentale pour le traitement d'information est le neurone. Un neurone est en fait une unité de décision : il décide (de façon graduelle) si oui ou non le signal présenté à ses entrées est semblable (en fonction d'un seuil) à ce qu'il a déjà appris.

Chaque MN est caractérisée par son architecture et sa loi d'apprentissage. En générale, on identifie quatre classes d'architectures :

Selon le nombre de couches :

- Les MN d'une couche (single layer networks);
- Les MN qui ont des couches cachées (multilayer networks);

Selon la direction de propagation d'information :

- Les MN dans lesquelles l'information se propage de la couche d'entrée vers la couche de sortie, (Feedforward networks)

- Les MN qui comporte des rétroactions (Recurrent networks).

Les neurones peuvent aussi être complètement interconnectés ou partiellement interconnectés. Dans une MN d'une architecture spécifiée, les connaissances sont mémorisées dans les valeurs des paramètres de la machine (poids synaptiques et seuils). La représentation internes des connaissances dans un MN est un sujet très complexe. Cependant, il existe des règles de nature générale, comme par exemple :

- Les entrées semblables doivent mener vers des représentations internes semblables.
- Les entrées appartenant à des classes différentes, doivent avoir des représentations internes différentes.
- Une caractéristique importante doit être représentée à l'aide d'un plus grand nombre de neurones.

L'une des propriétés les plus intéressantes des MN est leur capacité d'apprendre. L'apprentissage est le processus qui change les paramètres d'un réseau de neurones. Ces changements se font selon des règles, appelées les lois d'apprentissage (Ex. la correction d'erreurs, l'apprentissage Hebbien, l'apprentissage compétitif, etc.). Il existe trois paradigmes d'apprentissage : *supervisé* lorsque le réseau se sert de la réponse attendue pour changer ses poids, *non-supervisé* lorsque le réseau change ses poids en utilisant comme seules informations les vecteurs qui lui sont soumis pour apprendre, et finalement, l'apprentissage par *renforcement* qui est basé sur un processus d'essai-erreurs et vise à maximiser un index de performance global.

Il est évidemment impossible de décrire en quelques lignes un domaine aussi vaste et dont les applications sont si variées. Le lecteur intéressé à obtenir des informations supplémentaires est invité à consulter les livres Hecht-Nielson (1990), Haykin (1994) et Hassoun (1995) ou bien des revues spécialisées ("Neural Networks", "IEEE Transactions on Neural Networks") ou encore des comptes rendus de conférences annuelles importantes ("International Conference on Neural Networks" (ICNN), "World Congress on Neural Networks" (WCNN)).

La puissance et l'efficacité des machines neuronales est le résultat direct de leur structure – massivement parallèle et distribuée ainsi que de leur capacité d'apprendre en fonction des données disponibles et de généraliser cette apprentissage à des données potentielles mais inconnues. Pendant l'apprentissage, les machines neuronales s'adaptent aux caractéristiques propres à chaque entrée (commande gestuelle dans ce cas-ci).

Ainsi, contrairement aux approches symboliques qui exigent un niveau de précision important, les machines neuronales acceptent des informations imprécises. Puisqu'une commande gestuelle possède une variabilité intrinsèque à l'exécution, il semble approprié de confier le traitement de ce type d'information à une approche connexionniste, capable d'analyser les données reliées au geste sans être ennuyée par les variations par rapport à une forme de base idéale.

Un autre avantage des MN est leur rapidité de traitement, parce que tous les neurones fonctionnent théoriquement[6] en parallèle. L'analyse d'une forme prend alors moins de temps que l'application d'une technique classique comme par exemple, la programmation dynamique.

Ce sont donc ces mêmes qualités (capacité d'apprentissage, tolérance à la variabilité, rapidité de traitement) qui ont poussé des chercheurs à utiliser des MN dans des domaines d'applications voisins à la reconnaissance des commandes gestuelles, comme par exemple la reconnaissance de l'écriture manuscrite (Dimitriadis et Coronado 1995) et la reconnaissance de la parole (Bengio, 1996).

---

[6] Les implantations parallèles des MN sont peu courantes. Il s'agit ici de simulations informatiques qui sont des implantations séquentielles de comportement parallèle.

4.1.2 Justification de l'emploi du réseau ARTMAP

Le choix d'une architecture particulière est difficile parce qu'il existe des nombreuses architectures neuronales. Rappelons encore qu'un des buts visé est que notre éditeur gestuel soit adaptable. Précisons la signification du mot adaptable dans notre contexte. Adaptable signifie qu'on veut permettre à l'utilisateur d'ajouter à l'ensemble initial des commandes gestuelles, des gestes (formes) avec lesquels il est plus à l'aise.

Notre ensemble de commandes gestuelles présenté au tableau 3.8 ainsi que la correspondance "forme–action". Ces associations ne représentent pas une fonction bijective[7] parce qu'à certaines actions correspondent plusieurs formes. Ce fait est illustré par la figure 4.1 où le nombre des formes à reconnaître est N et le nombre d'actions est M (généralement diffèrent de N). La figure 4.1 fait ressortir aussi un autre fait : une action peut être appelée en dessinant des formes différentes, donc plusieurs formes peuvent correspondre à une même action.

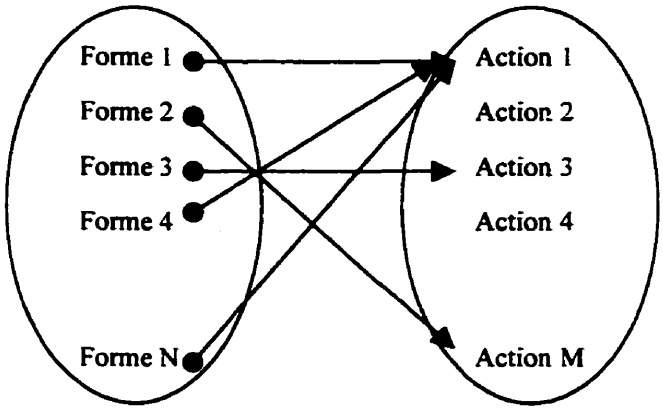


Figure 4.1 Correspondance "forme-action".

Dans notre système le nombre d'actions (M) qui peuvent être exécutées à partir d'un geste est fixé à huit: Effacer, Insérer, Disjoindre, Joindre, Nouvelle ligne, Palette d'édition, Annuler la dernière action, Image braille. Par contre le nombre de gestes à

reconnaître (N) peut varier ainsi que leurs formes, or, il faut que les ajouts puissent se faire en temps réel et sans corrompre les connaissances déjà acquises (apprentissage incrémental).

Selon Freeman (1991,) la majorité des MN (surtout les MLP - multilayer perceptron entraînés avec l'algorithme Back Propagation) ont de la difficulté à satisfaire cette dernière contrainte et voilà pourquoi typiquement l'utilisation des MN pour la reconnaissance des formes est précédée par la collecte de données. Ces données sont codées dans le système neuronal par les poids synaptiques pendant l'entraînement. Une fois l'entraînement fini, le système est prêt à être utilisé et aucune modification n'est plus permise. Ce scénario est applicable si la problématique peut être définie avec précision dès le début et si les données d'entraînement sont représentatives du problème à solutionner. Malheureusement ce scénario ne correspond pas toujours à la réalité, comme c'est le cas dans le contexte de ce travail. Supposons que le réseau est entraîné pour reconnaître un ensemble initial de formes et qu'on désire ajouter une autre commande gestuelle, donc une nouvelle forme à reconnaître. En général, si on veut ajouter une autre forme à l'ensemble appris, il faut recommencer l'entraînement du réseau (notons que souvent l'entraînement d'une MN exige beaucoup de temps) en lui présentant la nouvelle forme en plus de toutes les "anciennes" formes déjà apprises car autement, il est probable que le réseau oubliera les formes déjà apprises. Ceci arrive parce que la plupart des machines neuronales n'ont pas de mécanismes permettant d'ajouter des neurones pour distinguer les nouveautés de prototypes déjà appris. Nous venons de décrire ce que Stephen Grossberg et Gail Carpenter (Carpenter, Grossberg, 1987a) appelle le dilemme de la stabilité et de la plasticité. Ce dilemme comporte trois questions dont :

---

[7]À chaque action correspond une forme différente.

- ✓ Comment un réseau peut-il demeurer stable lorsqu'on lui présente des patrons d'entrée familiers mais être plastique lorsqu'on lui présente de nouveaux patrons d'entrée ?
- ✓ Comment le système peut-il changer entre le mode "plastique" et le mode "stable" ?
- ✓ Comment le système peut-il apprendre des nouvelles informations sans corrompre les connaissances déjà acquises ?

En réponse à ces questions Grossberg et Carpenter ont développé la théorie de résonance adaptative (ART).

4.1.3 La famille des réseaux ART

Au cours des années (à partir de 1987), Grossberg, Carpenter et leurs collègues ont créé une famille de machines neuronales : les architectures ART. Ces architectures présentent les avantages suivants :

- apprend rapidement après seulement quelques[8] présentations des patrons d'entrée,
- peuvent apprendre rapidement un nouveau patron d'entrée sans corrompre la mémoire existante (plasticité),
- permet d'ajuster la finesse de la reconnaissance.

Les architectures ART semblent être bien adaptées pour les problèmes de classification des formes dans des environnements réels. Elles offrent des avantages aux applications qui exigent un apprentissage continuel et autonome. Voilà, pourquoi on a décidé de créer notre reconnaisseur de commandes gestuelles en ayant recours à une architecture ART. Le tableau 4.1 contient une brève description des architectures ART existantes.

---

[8] On expliquera au chapitre 4, (Section 4.3.4 "L'influence du code Isométrique sur l'apprentissage de ART1") pourquoi dans notre cas une présentation est suffisante.

Tableau 4.1 La famille des architectures ART.

Nom de l'architecture ART	Description
ART1 (Carpenter, Grossberg, 1987a)	Réseau classificateur (clustering network). Apprentissage non supervisé Patrons d'entrée binaires.
ART2 (Carpenter, Grossberg, 1987b)	Réseau classificateur (clustering network) Apprentissage non supervisé Patrons d'entrée analogiques ou binaires
Fuzzy ART (Carpenter, Grossberg, Rosen, 1991)	Réseau classificateur (clustering network) Apprentissage non supervisé Incorporation des notions de la logique floue (fuzzy logic)
ARTMAP (Carpenter, Grossberg, 1991)	Réseau classificateur Apprentissage supervisé Agencement de deux unités ART pour former un réseau à apprentissage supervisé, patrons d'entrée binaires ou analogiques selon le type de réseaux ART qui le constitue ART1 ou ART2 respectivement
Fuzzy ARTMAP (Carpenter, Grossberg, Markuzon, Reynolds, Rosen, 1992)	Comme ARTMAP mais les ARTs sont remplacés par des fuzzyARTs.
Gaussian ARTMAP (Williamson, 1996)	Comme ARTMAP, mais la fonction de choix et la fonction de correspondance des modules ART sont définies comme des Gaussiennes et par conséquent, le réseau est plus performant et plus résistant au bruit

Comme nous avons besoin d'un classificateur à apprentissage supervisé nous avons choisi l'architecture ARTMAP. Tel qu'on peut voir au tableau 4.1, il existe deux variations de cette structure : ARTMAP "binaire" et ARTMAP "analogique". L'architecture ARTMAP binaire comporte deux modules ART1 tandis que l'architecture ARTMAP analogique est composée de deux modules ART2. Nous avons choisi d'utiliser l'architecture ARTMAP binaire parce que les réseaux ART2 semble être très

difficile à adaptér pour un problème donné à cause de leur sensibilité aux valeurs de leurs nombreux paramètres (Caudill, Butler, 1992). Selon Caudill et Butler "Changing certain parameter values as little as 1 % can have disastrous consequences for the network stability". Les réseaux ART2 sont aussi très lents alors que le but recherché ici est d'avoir un classificateur rapide. Ajoutons aussi que les réseaux ART1 et ARTMAP binaire ont été analysés au détails dans la littérature scientifique (Moore, 1988), (Georgiopoulos, Huang, Heileman, 1991), (Georgiopoulos, Huang, Heileman, 1992), (Georgiopoulos, Huang, Heileman, 1994).

4.1.4 Présentation de l'architecture ARTMAP

Le réseau ARTMAP binaire est composé de deux modules ART1, appelés - ARTa et ARTb, un module appelé Inter-ART et des contrôles qui gèrent l'apprentissage et le flux de données (figure 4.2). Pendant le processus d'apprentissage ARTa reçoit les patrons d'entrée et ARTb reçoit la classe qui aura à prédire. Les deux modules (ARTa et ARTb) sont liés par le réseau d'apprentissage associatif Inter-ART. Inter-ART est une mémoire associative, elle contient les connexions (les flèches sur la figure 4.1).

En principe, le réseau ARTMAP a deux modes de travail : le mode d'apprentissage et le mode de reconnaissance. Pendant l'apprentissage, on lui fournit deux signaux d'entrée : un pour le module ARTa qui est le patron à apprendre (une forme, dans ce contexte-ci) et l'autre, pour le module ARTb qui est le code correspondant à la catégorie (le code de l'action associée à la forme présentée à ARTa). On s'attend donc à ce que le réseau apprenne :

- les formes présentées ;
- les associations formes- actions.

Dans le mode reconnaissance, on fournit un seul signal : la forme à reconnaître qui est présentée au module ARTa. En réponse, le réseau émet l'action associée à la forme présentée (sortie de ARTb).

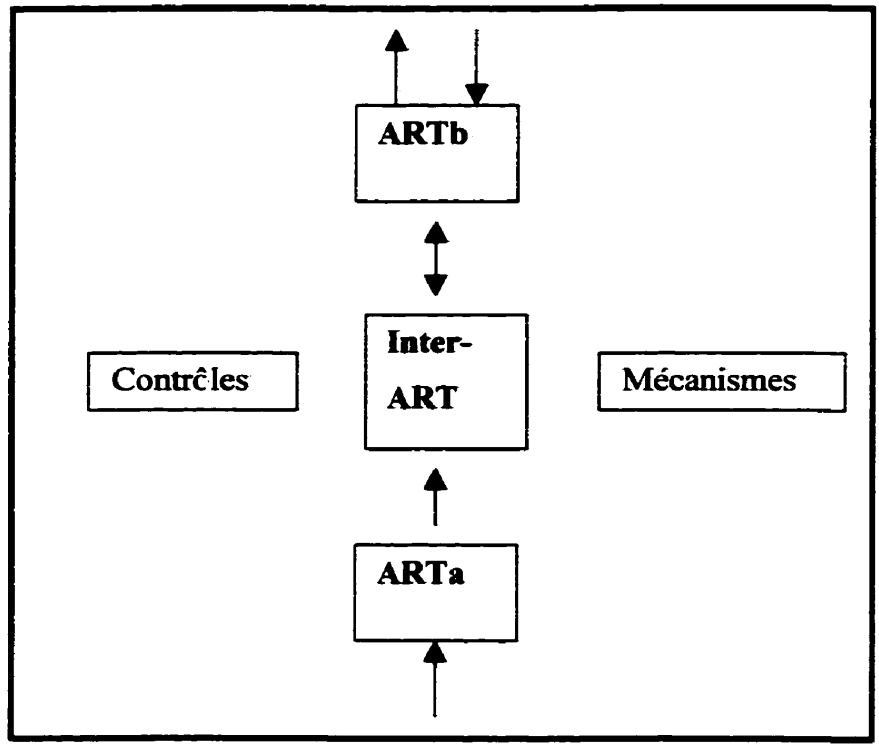


Figure 4.2 Le schéma bloc de l'architecture ARTMAP.

L'architecture ARTMAP est donc complexe. Le réseau est construit à partir de deux réseaux ART1 qui sont liés par le module Inter-ART. Dans la section qui suit, nous allons présenter l'architecture ART1. De cette façon, nous introduirons les notions importantes ainsi que le principe de la théorie de résonance adaptative. ART1 étant une architecture plus simple, donc plus facile à analyser, cela permettra par la suite, de comprendre plus aisément ARTMAP.

4.2 Le réseau ART1

4.2.1 Le principe de résonance adaptative

Le principe de résonance adaptative est l'idée de base qu'on retrouve dans toutes les architectures ART. Elle peut être expliquée en utilisant un réseau à deux couches (figure 4.3) : la couche d'entrée et la couche de prototypes. Le patron à classifier est présenté à la couche d'entrée, tandis que le résultat de la classification est émis par la couche des prototypes. Les deux couches sont complètement interconnectées.

Voilà une description simplifiée de la dynamique de fonctionnement. Chaque patron présenté à l'entrée stimule une activation qui se propage de bas en haut. De cette façon le patron d'entrée est présenté à chaque neurone de la deuxième couche après qu'il ait été modifié par les poids des connections. La deuxième couche essai de «deviner » quel est le patron d'entrée et fait une supposition (une hypothèse) qui se propage dans la direction : haut - bas. Donc les deux patrons, l'hypothèse et le patron d'entrée, se rencontrent à la première couche, s'ils sont assez semblables le patron est reconnu avec succès, sinon le réseau continue le processus en faisant une deuxième supposition.

Le degré de ressemblance demandé est indiqué par un paramètre du réseau appelé le critère de vigilance -ρ. Quand la première supposition faite par la couche de prototypes ne satisfait pas le critère de vigilance, ceci signifie que les deux signaux ne sont pas assez semblables, il est nécessaire d'avoir un mécanisme, permettant une autre hypothèse d'être testée. La couche de prototypes doit être libre d'émettre sa deuxième supposition. Pour cela, le premier gagnant doit rester inactif et laisser la compétition aux autres. Le mécanisme de réinitialisation assure un tel comportement du réseau. Finalement, le patron est associé à un prototype existant ou bien le réseau doit l'apprendre comme étant le premier exemplaire ou prototype d'une nouvelle classe. La couche de prototypes crée une hypothèse pour l'appartenance de chaque patron d'entrée

et envoie cette hypothèse à la première couche où l'hypothèse est validée. Si elle est correcte, le système entre en résonance pendant que les connections correspondantes se consolident mutuellement. Pour cette raison, la théorie est appelée "La Théorie de la Résonance", "Adaptative" parce que les poids sont modifiés continuellement pendant l'état de résonance.

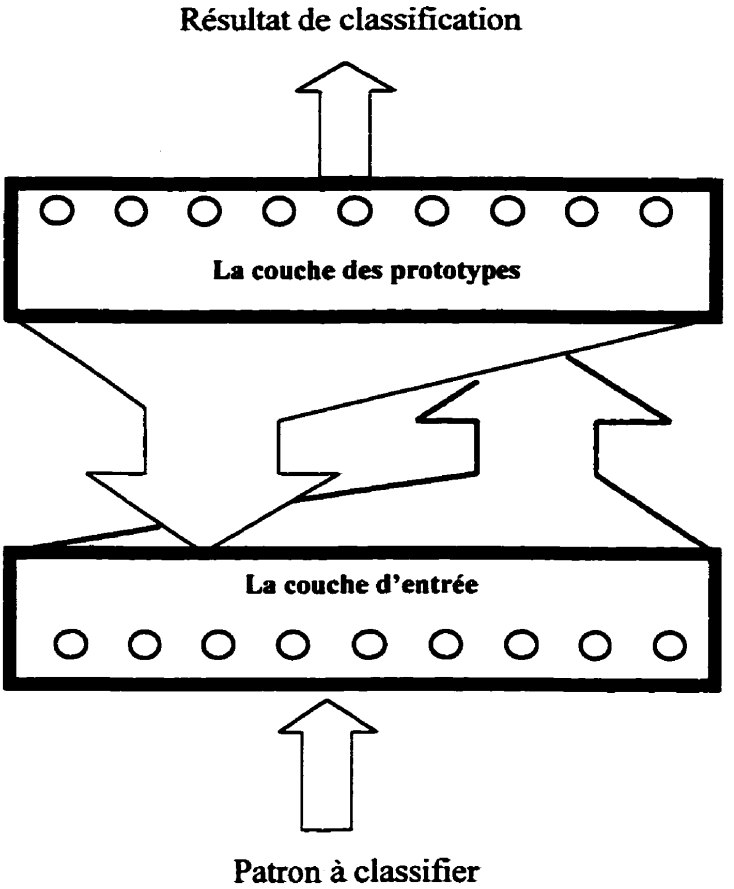


Figure 4.3 Le schéma bloc représentant les deux couches du réseau ART1.

Les réseaux ART sont donc des réseaux compétitifs basés sur les tests effectués à partir d'hypothèses. Leur fonctionnement est assuré par des mécanismes additionnels, comme par exemple, le système de réinitialisation qui désactive un prototype (une hypothèse) qui gagne la compétition, mais ne satisfait pas le critère de vigilance. Le critère de vigilance présente le niveau de confiance minimal du réseau pour qu'il accepte une hypothèse au lieu d'en chercher une autre plus appropriée.

Le critère de vigilance est un moyen de calibrer le mécanisme de sélection d'hypothèse. La valeur du critère de vigilance varie entre 0 et 1. Plus la valeur est grande, plus la ressemblance doit être grande, donc plus les détails doivent être pris en compte par le réseau. Une grande valeur du critère de vigilance résulte en un nombre important de prototypes à la deuxième couche. De l'autre côté, une petite valeur du critère de vigilance mène à un nombre restreint de prototypes à la deuxième couche. La valeur de ρ affecte la capacité du réseau à généraliser à partir des connaissances acquises. Une petite valeur permet au réseau de mieux généraliser qu'une grande valeur. Il est important de mentionner finalement, que ce réseau représente un groupe de patrons d'entrée appartenant à la même classe par un seul prototype, il peut donc également effectuer la compression des données.

Les généralités sur le fonctionnement des réseaux ART ont été présentées sans vraiment préciser l'architecture. Plus de détails seront fournis au lecteur dans ce qui suit.

4.2.2 L'architecture ART1

Le réseau ART1 a été introduit par Gail Carpenter et Stephen Grossberg en 1987 (Carpenter, Grossberg, 1987a). Leur but était de classifier des images binaires. Une restriction importante imposée par le réseau ART1 est la forme des patrons d'entrée : ils doivent être binaires et de longueur fixe. Le vecteur d'entrée (S comme source) est un vecteur binaire : $S \in \{1,0\}^N$ où N est la longueur du vecteur d'entrée.
ART1 est un réseau à deux couches : la couche d'entrée ou la couche des caractéristiques (F1) et la couche des prototypes ou la couche des catégories (F2). Sur la figure 4.4 la couche d'entrée est séparée en deux (F1a et F1b) dans le but d'illustrer le fait que la première couche a deux fonctions :

- Présenter le signal à classifier aussi long temps que nécessaire (F1a) ;

- Traiter les signaux (F1b). La couche F1b est aussi appelée couche interface parce qu'elle interface l'interaction entre F1a et F2.

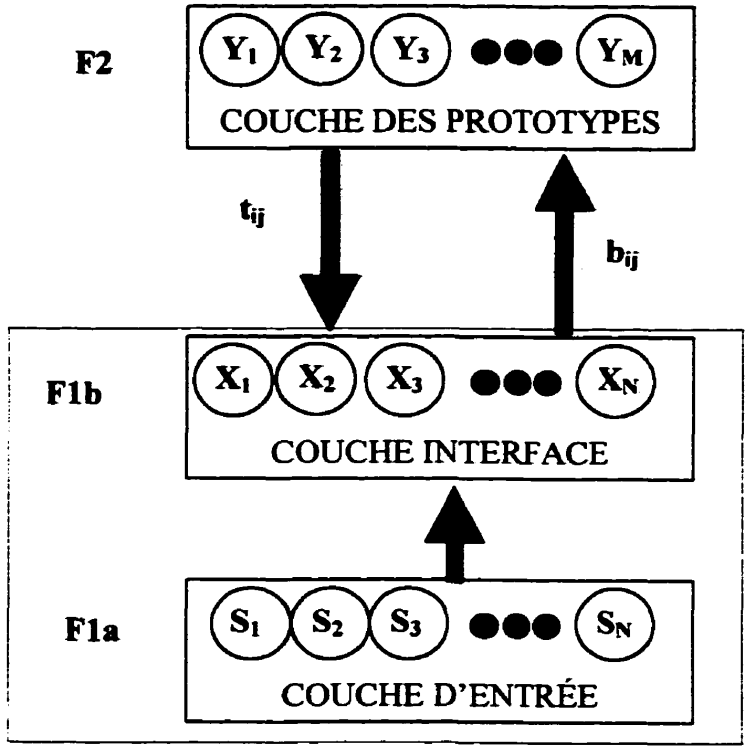


Figure 4.4 L'architecture générale du réseau ART1.

Pour comprendre l'algorithme décrit à la figure 4.5, il faut d'abord identifier les variables importantes. On désigne un neurone de la couche d'entrée (source) F1a par Si, ceux de la couche intermédiaire F1b par X_i, et ceux de la couche des classes F2 par Y_j. Chaque sortie des neurones de F1b est reliée via des connections ayant un poids b_{ij} (« b » pour Bottom-up) à tous les neurones de la couche F2. Chaque sortie des neurones de F2 est reliée à tous les neurones de F1b via des connections ayant un poids t_{ij} (« t » pour Top-down). Tous les neurones de la couche de sorties –F2 sont complètement interconnectés ce qui permet l'élection du neurone gagnant d'être suivie par la

désactivation des autres neurones. S est le vecteur source présenté au réseau. X_i et Y_j sont les activations des neurones des couches F1b et F2 respectivement :

$$X_i = \sum_j t_{ij} Y_j$$

$$Y_j = \sum_i b_{ji} X_i$$

N est la dimension de signal d'entrée et M, le nombre de neurones de la deuxième couche qui correspond au nombre de classes.

L'idée générale de ART1 est la suivante : à chaque étape de l'entraînement, on prend le patron d'entraînement S et on examine les prototypes existants qui lui ressemblent. Si on trouve un patron "assez similaire" Y_k (le test de similarité utilise le seul prédéfini - ρ) l'exemplaire S est ajouté vers la classe présentée par le prototype Y_k et le prototype Y_k est également modifié pour qu'il puisse mieux correspondre au signal S. Cette modification en réalité est un "ET" logique entre le prototype et le signal d'entrée qui change le prototype en diminuant le nombre de "1", donc le résultat est un sous-ensemble et non un moyen. Exemple : Y_k= 11111 00110 11111

S_j= 00001 11000 11100

Y_k ET S_j= 00001 00000 11100

Si un tel prototype Y_k n'existe pas, alors le patron d'entraînement S devient le prototype d'une nouvelle classe. La deuxième couche du réseau est vide au début, puis au fur et à mesure elle se remplit en mémorisant des informations importantes. Le regroupement ou la classification vise à mettre ensemble des données "semblables" et le réseau effectue la recherche des caractéristiques pertinentes de formation de groupes. La figure 4.5 présente le pseudocode de l'algorithme d'apprentissage de ART1. L'apprentissage rapide (Fast learning[9]) est considéré.

---

[9] On parle d'apprentissage rapide quand le signal d'entrée est présenté assez long temps pour que les poids des connections b_{ij} et t_{ij} puissent atteindre leurs valeurs limites.

Étape 0. Initialiser les paramètres:

Fixer le seuil de vigilance "ρ" (entre 0 et 1)

$Y_j = 1$

$t_{ji}(0) = 1$, $b_{ij}(0) = 1 / (1 + n)$

Étape 1. Pour chaque vecteur d'entraînement,

faire les Étapes 2 à 9.

Étape 2. Calculer la norme ||S||: $||S|| = \sum S_i$

soit le nombre de "1" dans le vecteur S.

Étape 3. $X = S$

Étape 4. Pour chaque neurone Yj qui est actif, calculer

$Y_j = \sum b_{ij} \wedge Xi$

Étape 5 Prendre le Y_j le plus grand.

Étape 6. Recalculer les activations X de F1b:

$X_i = X_i \wedge (t_{ji} \wedge Y_j)$

Étape 7. Calculer la norme de X: $||X|| = \sum X_i$

Étape 8. Faut-il une autre classe ?

Comparer avec le critère de vigilence:

Si ($||X|| / ||S||$) $< \rho$

Alors Y_j est désactivé

Retourner à l'Étape 4

Si ($||X|| / ||S||$) $<= \rho \rightarrow$

Passer à l'étape 9

Étape 9. Modifier les poids du noeud J

$b_{ij}(\text{nouveau}) = X_i / (1+||X_i||)$

$t_{ij}(\text{nouveau}) = X_i$

Figure 4.5 Pseudocode de l'algorithme d'entraînement du réseau ART1.

4.2.3 Caractéristiques du réseau ART1

La dynamique de l'architecture ART1 est décrite par un système d'équations différentielles dans l'article original de Carpenter et Grossberg de 1987. Dans le même article, les auteurs introduisent des hypothèses simplificatrices (Moore, 1989) grâce auxquelles le comportement du ART1 peut être analysé en termes d'un algorithme de classification discret. Barbara Moore dans son article "ART1 and Pattern Clustering" (1989) a analysé la capacité du réseau ART1 à classifier. Elle a rapporté ses observations sur son fonctionnement. Ses travaux revêtent une grande importance et résument bien les caractéristiques de ART1 qui sont parfois difficiles à déduire. Le lecteur intéressé est invité à consulter cet article, ici nous ne mentionnons que ce qui sera utile pour les discussions sur ART1 et ARTMAP.

1. *Le réseau ART1 est un réseau classificateur (clustering network) de vecteurs binaires, donc il regroupe automatiquement les vecteurs d'entrée dans des catégories. À chaque vecteur d'entrée, le réseau attribue une étiquette qui correspond à un groupe. Chaque groupe est représenté par un vecteur prototype. Le réseau ART1 est un classificateur incrémental. Les algorithmes incrémentaux peuvent traiter un nombre infini des données d'entrée. Les prototypes contiennent implicitement l'information sur chaque exemple vu. Par conséquent, ces algorithmes ne sont pas exigeants en termes de mémoire.* Cette qualité du ART1 et ARTMAP a été particulièrement appréciée compte tenu des contraintes techniques limitatives de l'ordinateur utilisé dans ce projet. Rappelons aussi qu'on vise un reconnaisseur de commandes gestuelles compact.
2. *ART1 utilise un algorithme d'apprentissage incrémental, différentes séquences de présentation des vecteurs d'apprentissage peuvent donc créer des prototypes différents. Autrement dit, l'ordre de présentation des données influence la formation des prototypes. Ceci est un désavantage des classificateurs incrémentaux.*

3. *Une fois l'apprentissage terminé, chaque vecteur d'entrée accède directement a son prototype.* Le réseau en mode de reconnaissance est très rapide. Ce qui répond á nos attentes, puisqu'on voulait que notre reconnaisseur soit rapide.
4. *Le réseau ART1 traite les "1" et les "0" d'une façon asymétrique.* Pour ART1 les "1" et "0" représentent respectivement la présence et l'absence de l'information. Rappelons que ART1 a été conçu pour traiter des images binaires où cette approche est raisonnable.
5. *La quantité de variations permises dans une catégorie donnée dépend de l'amplitude (l'amplitude d'un signal binaire est le nombre de "1"dans le signal) de son prototype. Plus cette amplitude est grande, plus de variations sont permises.*

4.3 Nature des données et prétraitement

Le projet porte sur l'utilisation de réseaux de neurones ARTMAP pour la reconnaissance des commandes gestuelles. L'acquisition des données se fait par l'intermédiaire de l'écran à cristaux liquides du bloc-notes électronique (Stylistic 1000). La fonction de l'écran est d'échantillonner la position du crayon en produisant la paire d'information : les coordonnées (X_i, Y_i) de la pointe du crayon sur sa surface. Donc une commande gestuelle (CG) est formée d'une séquence de cordonnées bidimensionnelles :

$$CG=\{(X_1,Y_1), (X_2,Y_2), (X_3,Y_3),\ldots(X_L,Y_L)\}$$

où L est le nombre d'échantillons qui composent la commande gestuelle. La fréquence d'échantillonnage est 100Hz et la résolution est 1/100 pouce. Un geste qui dure une seconde occupe donc 100 octets.

4.3.1 Exigences et contraintes dues à l'utilisation du réseau ART1

Une restriction importante imposée par le réseau ART1 est la forme des patrons d'entrée : ils doivent être binaires et de longueur fixe. Or, notre tâche consiste à traiter des signaux spatio-temporels. Le prétraitement qui doit être effectué est donc une étape importante. Les opérations de prétraitement préparent, modifient ou transforment les signaux acquis de l'écran pour faciliter leur traitement.

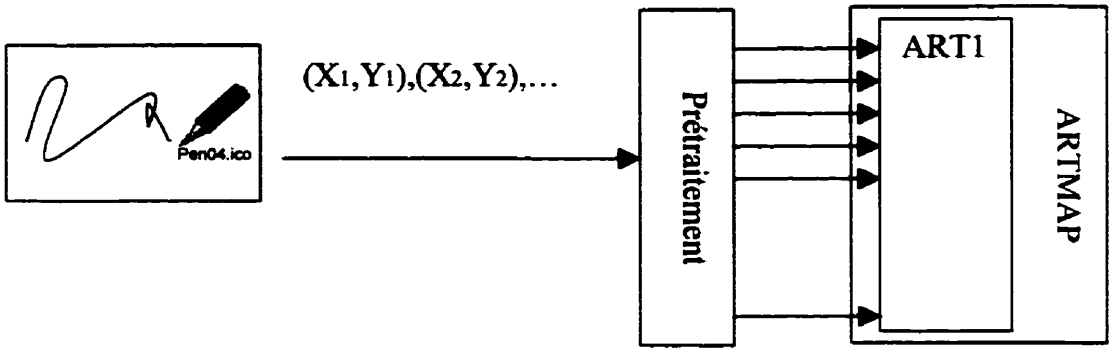


Figure 4.6 Le schéma bloc illustrant la nécessité de prétraitement.

Le problème qui nous amène à définir une séquence de prétraitement est illustré par la figure 4.6. Les gestes sont des séquences des coordonnées X et Y de points du tracé, donc, des séquences de longueurs variables. De l'autre côté, le réseau ART1 est le module dans l'architecture ARTMAP qui reçoit les signaux d'entrée et qui traite des signaux binaires de longueur fixe. Dans un premier temps il a fallu définir une série de transformations qui pour assurer l'adéquation désirée.

Nous avons en effet défini deux séries de transformations donc deux types de prétraitement. Notre but était de les utiliser lors de la vérification expérimentale et de choisir celui qui offrirait les meilleurs résultats. Pour ce faire ces deux types de prétraitement ont été identifies ainsi : type A et type B. Le prétraitement de type A

cherche à déterminer d'abord la direction dominante pour chaque segment du geste, cette direction est ensuite codée avec le code Isométrique et présentée au réseau ARTMAP. Les étapes du prétraitement de type A sont les suivantes:

Prétraitement de type A

- Rééchantillonnage équidistant du tracé (équivalent à un filtrage spatial qui élimine entre autres des points superposés).
- Calcul des directions de chaque partie et codage de la direction avec une des 8 directions dites de Freeman (résolution circulaire de $\pi/4$ – figure 4.17).
- Division du tracé en **N** parties de longueur égale et détermination de la direction dominante.
- Codage de la direction dominante ou moyenne avec une des 8 directions dites de Freeman (résolution circulaire de $\pi/4$) avec un code binaire quelconque.

De l'autre côté, le prétraitement de type B débute avec un rééchantillonnage équidistant du tracé, puis se poursuit avec l'extraction des (N+1) points, la direction entre chaque deux points est ensuite calculée et c'est la séquence de directions qui est codée en Freeman, avant d'être présentée au réseau ARTMAP. Les étapes de prétraitement de type B sont les suivantes :

Prétraitement de type B

- Rééchantillonnage équidistant du tracé.
- Division du tracé en **N** parties de longueur égale et calcul de la direction de chaque partie.
- Codage de cette direction avec une des 8 directions dites de Freeman (résolution circulaire de $\pi/4$).
- Codage de la direction (Freeman) avec un code binaire quelconque.

Le rééchantillonnage équidistant (RÉ) du tracé réduit l'effet des erreurs de discrétisation causées lors de l'acquisition des données et permet d'adoucir la courbe manuscrite. Il engendre une sorte de normalisation des données parce que le nombre de points après le rééchantillonnage équidistant est proportionnel à la longueur du tracé. La distance entre chaque paire de points après le rééchantillonnage est spécifiée par un paramètre – le rayon de rééchantillonnage (R). La figure 4.7 a) représente l'échantillonnage du tracé produit par le système, le rééchantillonnage équidistant du tracé (figure 4.7 b) ainsi qu'un agrandissement (figure 4.7 c) qui illustre la signification du rayon de rééchantillonnage.

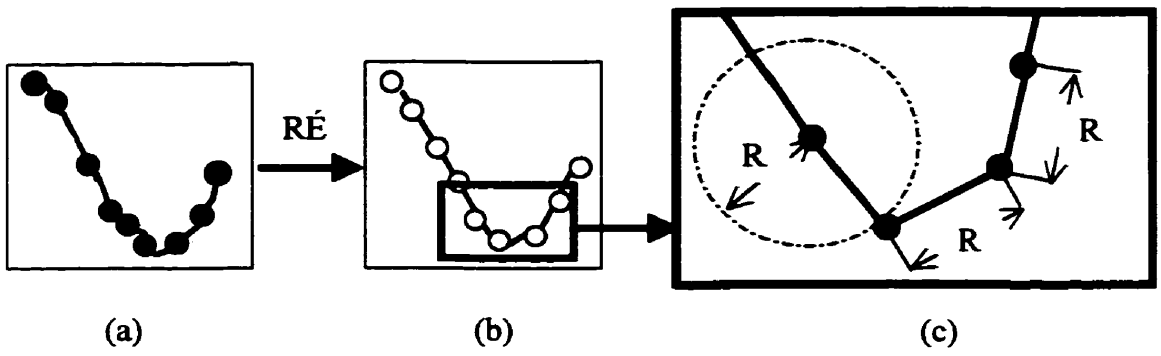


Figure 4.7 Rééchantillonnage équidistant du tracé.

Le rayon de rééchantillonage du tracé manuscrit (R) est le paramètre de notre reconnaisseur gestuel qui caractérise le prétraitement. Le but du rééchantillonnage équidistant dans l'espace est d'effectuer un filtrage spatial : la longueur du signal obtenu doit être proportionnelle à la longueur du tracé et non à sa durée. On veut donc disposer d'une quantité d'information suffisante mais non redondante (ce qui peut arriver si R est choisi trop petit). La valeur de ce paramètre dépend de la résolution de l'écran utilisé pour l'acquisition des données et de la fréquence d'échantillonnage du système.

Dans les deux types de prétraitement, le tracé est divisé en **N** segments de longueur égale. Il s'agit de transformer un signal de longueur variable en un signal de longueur

fixe. Pour ce faire il faut déterminer le nombre des segments de chaque partie ainsi que la direction dominante. Par conséquent, le signal d'entrée devient le suivant :

$$CG_{SD}=\{D_1, D_2, D_3, \ldots, D_N\}$$

CG, pour commande gestuelle et SD, pour séquence de directions. D_1 est le code de la direction dominante (moyenne) de la première partie, D_2 est le code de la direction dominante (moyenne) de la deuxième partie et N le nombre de segments. Ces directions sont par la suite codées à l'aide d'un code binaire spécifique et adapté à ART1.

Cette première série de transformations permet de s'assurer que les vecteurs d'entrée seront de longueurs fixes. Avant de poursuivre l'analyse, il faut rappeler que le réseau ART1 ne considère que les « 1 ». En fait, les « 0 » et les « 1 » codent l'absence et la présence d'information respectivement (section 4.2.3 Caractéristiques du réseau ART1).

- Pour l'application recherchée, il faudra que le nombre de « 1 » utilisés pour coder une direction soit le même, quelle que soit la direction car il n'y a pas a priori de direction plus importante qu'une autre.
- D'autre part, bien que la position d'un « 1 » dans le vecteur n'ait pas d'importance en soi, celle-ci a néanmoins une signification. Comme le réseau compare des vecteurs, deux vecteurs semblables doivent posséder des codes semblables.
- Finalement, dans un geste manuscrit, on doit tenir compte du fait que la différence maximale de direction entre deux gestes est de 180°. Étant donné que les 8 directions de Freeman sont utilisées, il faut avoir recours à un code qui soit capable de mesurer une différence maximale de 4 unités.

4.3.2 Le code isométrique

Afin que soient respectées toutes les contraintes mentionnées ci-haut, un codage dit isométrique (voir tableau 4.2) est proposé (Sabeva, Brault, Plamondon, 1998). Ce dernier semble plus approprié que les codes populaires BCD (Binary Coded Decimal) ou

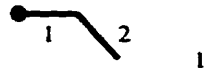
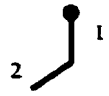
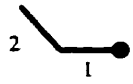
Gray dans le contexte d'utilisation du réseau ART1 parce que ce codage permet de garder les rapports de distances entre les directions.

Pour vérifier l'importance de cette étape du codage, six formes simples n'incluant aucune levée de crayon, ont été présentées séquentiellement au réseau ART1 (tableau 4.3). Le point noir indique le début du tracé. Le tracé de chaque forme est divisé en deux parties de longueur égale dont la direction est exprimée à l'aide d'une des huit directions de Freeman. Ces huit directions ont été codées à l'aide des codes BCD, Gray, et le code Isométrique proposé. Selon le cas, des séquences binaires différentes sont obtenues et utilisées à l'étape suivante pour entraîner le réseau et par conséquent, pour la création des classes.

Tableau 4.2 Trois codes utilisés pour le codage des directions de Freeman lors des tests.

Direction de Freeman	Code binaire (3 bits)	Code de Gray (4 bits)	Code Isométrique (8 bits)
0	000	0000	**00001111**
1	001	0001	**00011110**
2	010	0011	**00111100**
3	011	0111	**01111000**
4	100	1111	**11110000**
5	101	1110	**11100001**
6	110	1100	**11000011**
7	111	1000	**10000111**

Tableau 4.3 Formes simples utilisées pour tester la capacité du réseau ART1 à créer une représentation intérieure adéquate.

i	Formes à classifier (patrons, I_i)	Directions de Freeman (par unité de $\pi/4$)	
		Dir.1	Dir. 2
1		0	1
2		0	7
3		6	5
4		7	5
5		4	3
6		3	5

4.3.3 Justification intuitive du codage isométrique

La figure 4.8, démontre l'emplacement des six patrons d'entrée dans l'espace des caractéristiques (Direction 1, Direction 2). Il dénote 3 paires de formes, soit (1 et 2), (3 et 4), et (5 et 6). Les formes 3 et 4 sont les formes les plus semblables et sont représentées par les deux points les plus proches (I_3 et I_4), tandis que les formes 5 et 6 sont représentées par la paire de points les plus éloignés (I_5 et I_6). Naturellement, la

distance entre les points d'une paire est plus petite que la distance entre deux points qui appartiennent à des paires différentes.

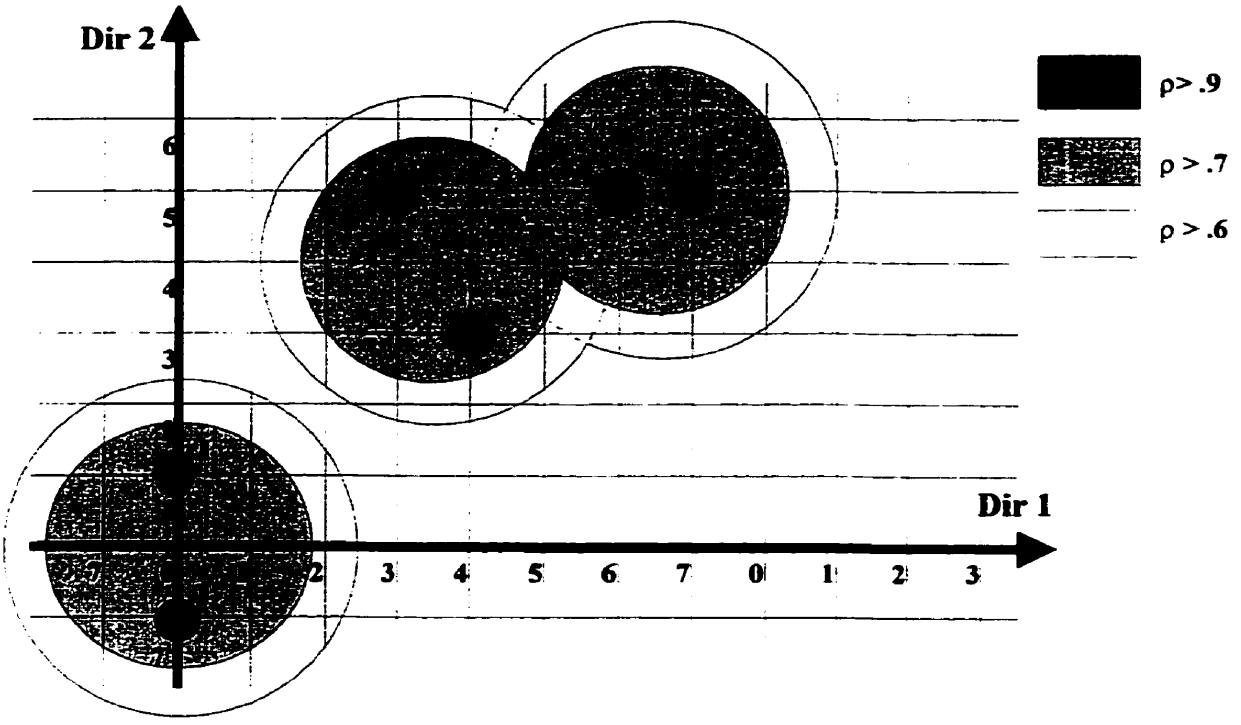


Figure 4.8 L'emplacement des six patrons d'entrée dans l'espace des caractéristiques (Direction 1, Direction 2).

Comme le réseau ART1 est un classificateur dont le critère de vigilance permet de contrôler la finesse des regroupements, il est possible de démontrer la régularité des regroupements en fonction des différents codages. Quand le critère de vigilance a une valeur inférieure à 0.6, le réseau est en mode de généralisation : il crée trois classes (représentées par les plus grands cercles de la figure 4.8). En augmentant la valeur du critère de vigilance, le réseau doit considérer plus de détails. Pour $\rho = 0.7$, le réseau ART1 a créé 4 classes en séparant les deux formes les plus éloignées (patrons 5 et 6). Si $\rho = 0.8$ la finesse est encore plus grande et les formes 1 et 2 sont placées dans les classes

différentes. Finalement, pour $\rho >= 0.9$ toutes les formes sont mises dans des classes distinctes.

La figure 4.9 montre ces mêmes résultats expérimentaux mais sous forme graphique en les comparent aussi avec ceux obtenus avec les deux autres codages mentionnés. Le réseau fonctionne d'une façon contre-intuitive si on utilise le code de Gray ou le code BCD. La généralisation se produit beaucoup trop tôt avec le code de Gray et se spécialise trop tard pour les codes BCD et Gray. De plus, l'utilisation de ces codes conduit le réseau à créer des prototypes différents pour des formes qui sont proches (ex. : Formes 1 et 2) et à mettre ensemble des formes très différentes (ex. : Formes 1 et 5).

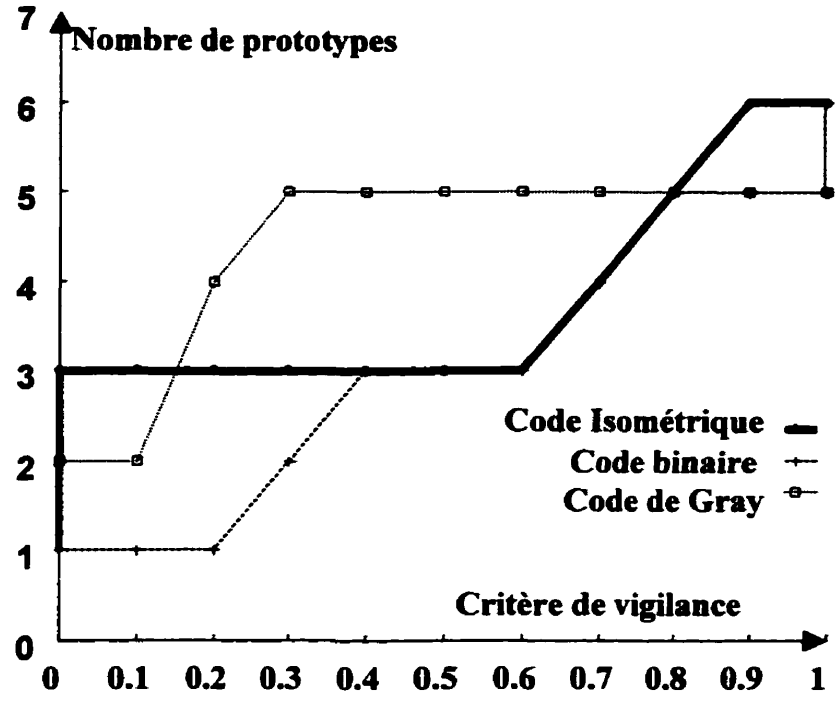


Figure 4.9 Le comportement du réseau ART1 représenté par la relation : nombre de prototypes créés en fonction du critère de vigilance.

4.3.3 Résultat de la classification des gestes avec ART1

L'objectif recherché à cette étape du projet est de classer les 13 commandes gestuelles illustrées à la figure 4.10. L'algorithme de classification utilisé ici, nécessite la détermination de trois paramètres : le rayon de rééchantillonnage R (non crucial), le nombre N de segments de longueur égale qui représentera le tracé de chaque geste manuscrit (on a fixé ce nombre à 10) et le critère de vigilance ρ du réseau ART1 qui varie entre 0 et 1.0. Chaque commande gestuelle a été représentée par une séquence de 10 directions et cette séquence a été codée en utilisant chacun des trois codes du tableau 4.2.

- Code Isométrique : Les 13 gestes montrés à la figure 4.10 ont été présentés au réseau dans l'ordre descendant. Le dendogramme permet de voir le processus de regroupement qu'accomplit le réseau en fonction du critère de vigilance. Ce processus se produit d'une façon logique et compréhensible pour le code isométrique. Le réseau regroupe dans une même classe, des formes qui sont semblables, et l'augmentation du critère de vigilance mène vers une spécialisation.
- Code BCD et code de Gray : il s'est avéré impossible de construire des dendogrammes dans ces deux cas à cause du manque de stabilité dans la formation des classes. En changeant la valeur de ρ, certaines formes se déplaçaient entre différents groupes. Par exemple, la forme I_6 a été groupée avec les formes I_3 et I_4 qui sont très différentes. Ou encore, les formes I_7, I_8 et I_{10} se sont regroupées à ρ=0.1 et sont restées ensemble jusqu'à ce que ρ=1.0. En fait, le réseau ART1 fonctionne d'une façon imprévisible, incompréhensible et instable si on utilise ces deux codes (BCD, Gray) pour représenter les données.

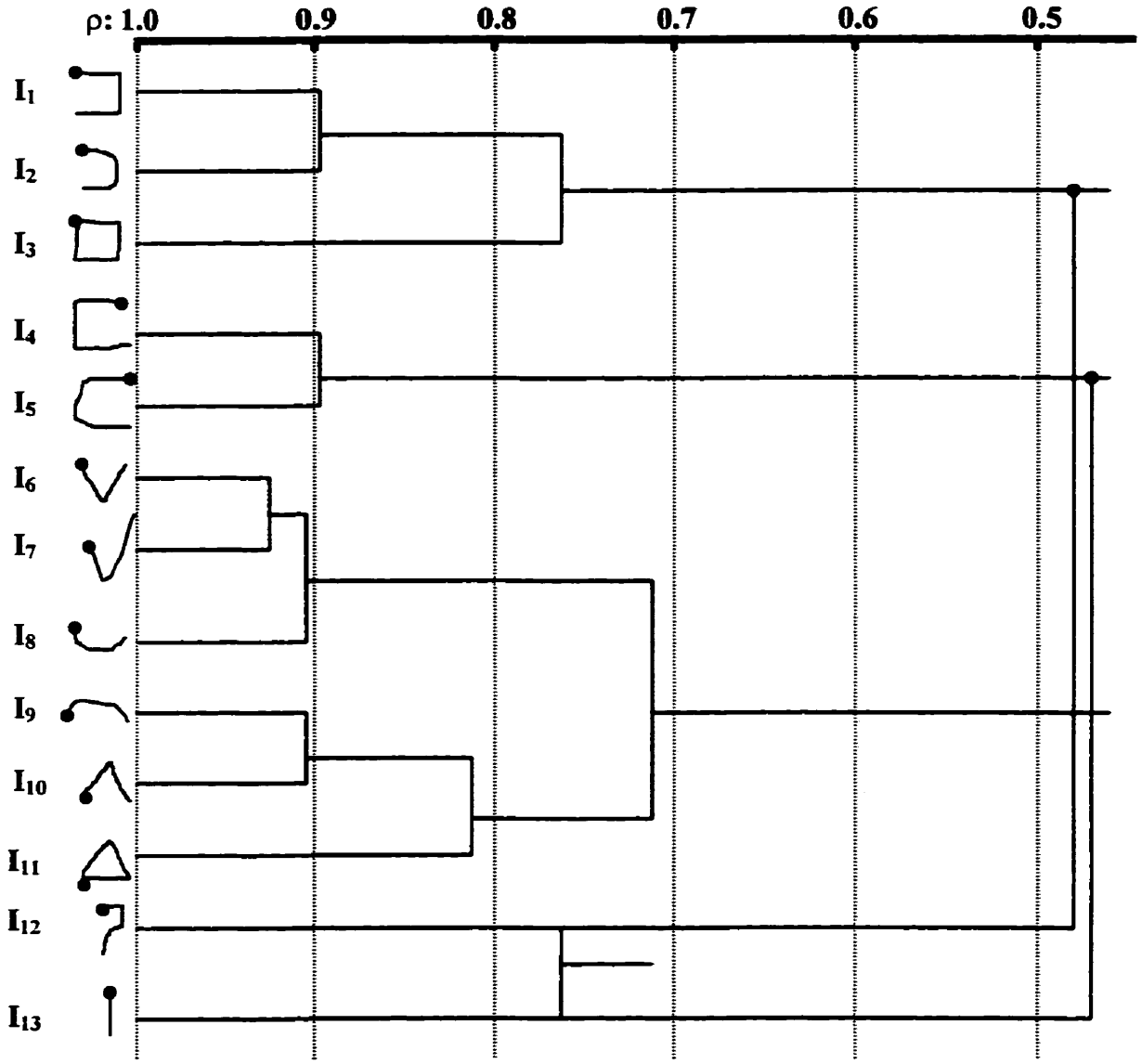


Figure 4.10 Dendogramme illustrant le processus de regroupement accompli par ART1 en fonction du critère de vigilance en utilisant le code isométrique.

4.3.4 L'influence du code Isométrique sur l'apprentissage de ART1

Plusieurs chercheurs ont analysé des architectures ART (Moore, 1988); (Georgiopoulos, Huang, Heileman, 1991); (Georgiopoulos, Huang, Heileman, 1992); (Georgiopoulos, Huang, Heileman, 1994). Les résultats publiés par l'équipe de Georgiopoulos portent sur les propriétés d'apprentissage du réseau ART1, liées à la diversité des patrons d'entrée. Ils prouvent que le nombre de présentations de l'ensemble d'apprentissage au réseau ART1 dépend uniquement du nombre de patrons qui diffèrent en amplitude[10] dans l'ensemble d'apprentissage[11]. Alors pour que le réseau soit stable il a besoin de M présentations de l'ensemble d'apprentissage s'il y a M groupes de patrons qui ont des amplitudes différentes dans cet ensemble. Il suffit d'observer le code pour constater que l'amplitude de tous les codes est 4; donc constante. Cette qualité demeure présente dans le cas de l'agencement des éléments codés de cette façon, seule l'amplitude changera, elle sera 4*Ne (Ne – nombre des éléments codés avec le code Isométrique). Alors, dans notre ensemble d'entraînement l'amplitude de tous les patrons d'entrée est constante. Par conséquent un seul cycle d'apprentissage suffira pour que le réseau stabilise ses catégories. Les tests effectués ont affirmé que le code Isométrique permet un apprentissage rapide.

4.3.5 Conclusion

À cette étape de ce projet, la nature des données et la description des opérations à effectuer pour préparer ces données aux traitements subséquents ont été définies. Une façon de représenter des séquences spatio-temporelles de tracés manuscrits permettant le

---

[10] L'amplitude d'un signal binaire est égale au nombre de "1".

[11] Cette confirmation s'applique sous certaines conditions qui sont accomplies dans ce projet.

classement par un classificateur connexionniste de type ART1 a été proposée. Il a été démontré que, quoique le codage isométrique proposé soit plus long que les codages conventionnels, la régularité de son comportement en fonction des critères de regroupement le rend plus apte à remplir sa fonction de classificateur automatique et que grâce à ce code, un seul cycle d'apprentissage suffit pour que le réseau stabilise ses catégories.

4.4 L'architecture ARTMAP – la base de notre reconnaisseur de commandes gestuelles

Dans cette section, l'architecture, le fonctionnement et les mécanismes propres au réseau ARTMAP sont présentés, de même que l'algorithme utilisé pour la simulation ainsi que tout ce qui est lié à son implantation: les valeurs de ses paramètres et la formation des signaux d'entrée. L'architecture du réseau ARTMAP est décrite en détail dans l'article de Carpenter, Grossberg (Carpenter, Grossberg, 1991).

4.4.1 ARTMAP- une architecture prédictive

Le schéma bloc du réseau ARTMAP a déjà été présenté à la figure 4.2. ARTMAP est composé de deux réseaux ART1 connectés à l'aide d'un module de mémoire associative Inter-Art et des contrôles qui régissent l'apprentissage. Dans le mode d'apprentissage, le patron à classifier Ia (input ARTa) est présenté à ARTa et le code de la classe correspondant Ib (input ARTb) est présenté à ARTb. Ce type d'architecture est une architecture prédictive parce qu'elle peut apprendre comment prédire un signal (de dimension Nb) à la sortie, tout en ayant un signal (de dimension Na) à l'entrée. Inter-ART lie les couches de prototypes de ARTa et ARTb.

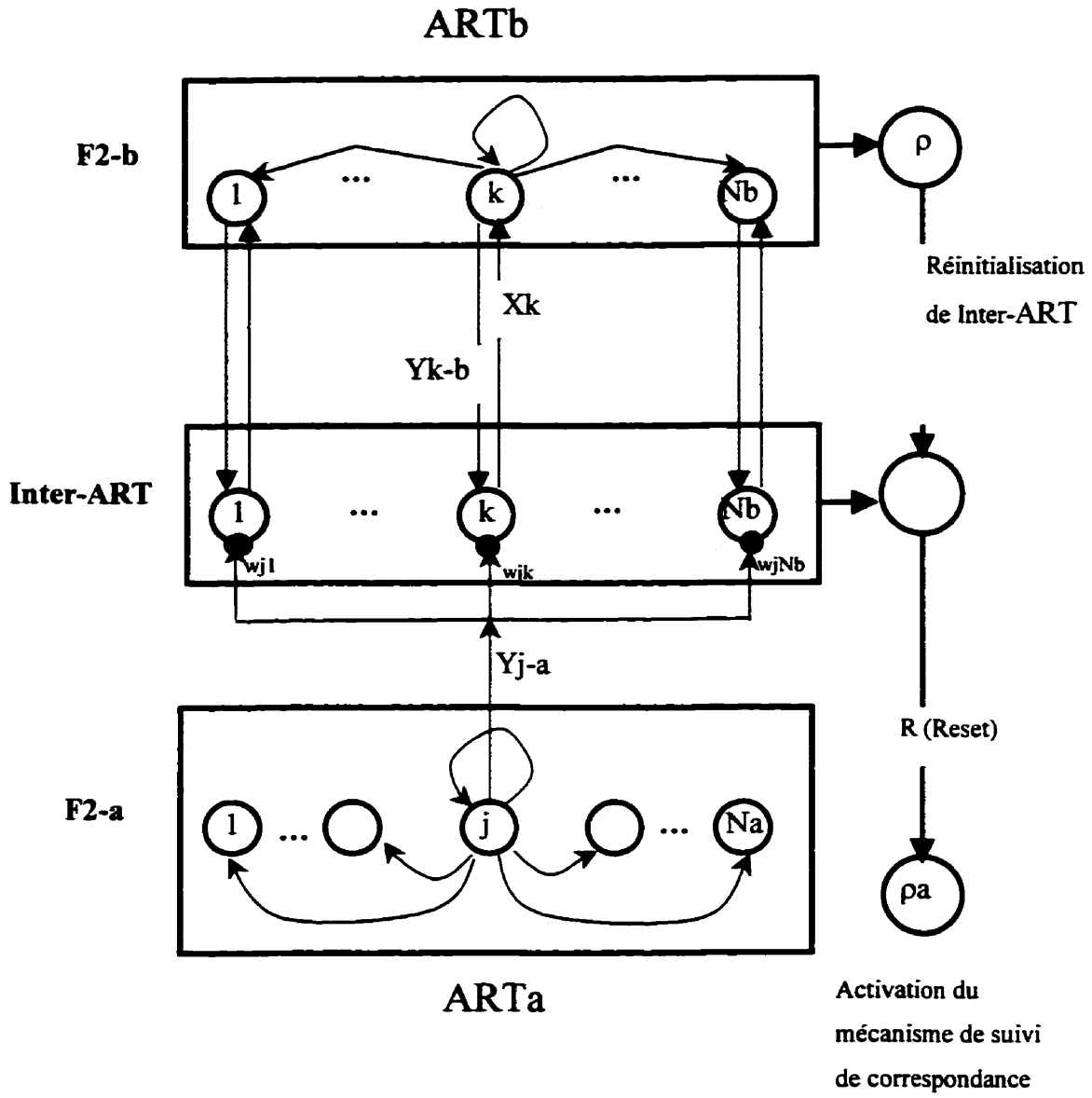


Figure 4.11 L'architecture du réseau ARTMAP.

La convention utilisée (figure 4.11) se résume ainsi :

- F2-a signifie la couche des prototypes du module ARTa ;
- F2-b signifie la couche des prototypes du module ARTb ;
- Inter-ART est la place où les signaux de F2a et F2b interagissent ;
- Les neurones de F2-a sont indexés de 1 à Na et les neurones de F2-b de 1 à Nb;

- J est l'index du neurone actif de F2-a ;
- K est l'index du neurone actif de F2-b ;
- Yj-a et Yk-b désignent des sorties de neurone j de F2-a et sortie de neurone k de F2-b.
- Les neurones d'Inter-ART sont indexés de 1 à Nb comme F2-b parce qu'ils correspondent au nombre de neurone de la couche F2-b.

La figure 4.11 illustre ses principales composantes. Le champ de correspondance est lié à F2b à travers des connexions bijectives, qui ne s'adaptent pas. Néanmoins, chaque neurone de F2a est lié à tous les neurones dans le champ de correspondance par des connexions adaptatives. Le champ de correspondance ou Inter-ART contrôle l'apprentissage associatif entre les catégories représentées dans ARTa et celles représentées dans ARTb. Donc il n'associe pas directement un patron d'entrée de ARTa à un patron d'entrée de ARTb.

Chaque module ART dans ARTMAP compresse les données en réponse de la séquence des patrons d'entrée (Ia) et (Ib). L'apprentissage associatif dans le champ de correspondance lie la paire de patrons à travers des codes compressés. Si un vecteur d'entrée Ia est associé à un vecteur d'entrée Ib, chaque vecteur d'entrée qui active la catégorie associée à Ia va prédire le catégorie du patron Ib.

Le processus d'apprentissage commence par un cycle de classification non supervisé qui se déroule dans chacun des modules ARTa et ARTb et se poursuit avec la vérification si l'association désirée existe ou non dans Inter-ART. En d'autres mots, on vérifie si la catégorie activée chez ARTa est connectée à la catégorie activée chez ARTb. Si cette connexion existe, le réseau ARTa apprend le patron d'entrée Ia et le réseau ARTb apprend le patron d'entrées –Ib. De l'autre côté, si l'association désirée n'existe pas (la catégorie active de ARTa n'est pas liée à la catégorie active de ARTb) le module Inter-

ART est alors réinitialisé. La réinitialisation déclenche un mécanisme propre au réseau ARTMAP, appelé le mécanisme de suivi de correspondance

Chaque neurone du champ de correspondance Inter-ART est lié à un neurone de la deuxième couche du module ARTb et la connexion est bi-directionnelle. Chaque neurone de la deuxième couche du module ARTa est lié aux tous les neurones de l'Inter-ART et les connexions sont adaptatives.

4.4.2 La réinitialisation et le mécanisme de suivi de correspondance

La mécanisme de réinitialisation et le mécanisme de suivi de correspondance assurent le fonctionnement du réseau ARTMAP. Dans les réseaux ARTMAP, il est possible d'associer différentes catégories à des patrons d'entrée similaires. Des patrons d'entrée très différents peuvent aussi appartenir à la même classe. Ces deux qualités, assez particulières, sont supportées par le système d'orientation du module Inter-ART. Le système d'orientation devient actif si le réseau ARTa fait une prédiction qui est incompatible avec le signal d'entrée du module ARTb. Cet événement - manque de correspondance - active le mécanisme du suivi de la correspondance (match tracking).

Pour l'expliquer, il faut se rappeler que chaque module ARTa et ARTb, étant un réseau ART1, est caractérisé par un paramètre – le critère de vigilance ρ. Le réseau ARTMAP a deux de ces paramètres: le critère de vigilance du ARTa (ρa) et le critère de vigilance du ARTb (ρb). Cependant il existe une différence entre ARTMAP et ART1 lié à la signification du critère de vigilance uniquement pour ARTa, la signification de ρb ne diffère pas de celle expliquée précédemment pour le réseau ART1.

Dans ART1, la valeur de ρ ne change pas en mode de fonctionnement. Par contre, dans ARTMAP, la valeur de base du critère de vigilance de ARTa - ρa est une valeur minimale du critère de vigilance. ARTMAP permet l'augmentation de cette valeur lors de l'apprentissage sous certaines conditions. La réinitialisation du champ de correspondance est la condition et le mécanisme de suivi de correspondance détermine la valeur de ρa. L'augmentation de ρa désactive la catégorie active de ARTa et permet le début d'une nouvelle étape de classification de ARTa avec une valeur du critère de vigilance plus grande, cette fois ci.

De là, on peut se poser la question suivante : "De combien doit-on augmenter ρa ?". On peut déduire qu'en augmentant d'une façon minimale ρa, le prototype actif pourra être désactivé et une nouvelle hypothèse pourra ainsi être testée. Le cycle d'ajustement de ρa dans le temps est le suivant :

- Au début de chaque présentation d'un patron d'entrée à ARTa : $\rho a = \rho a^*$.
- Le réseau ARTa entre en résonance et le neurone J de F2a est activé. Il existe deux possibilités :
    1. La catégorie prédite par le neurone J est la catégorie correcte.
    2. Il n'y a pas de correspondance entre la catégorie prédite par le neurone J et celle indiquée par ARTb. Dans ce cas, Inter_ART émet le signal de réinitialisation (Reset). Ce signal active le mécanisme de suivie de correspondance qui module la valeur du critère de vigilance ρa. Ce mécanisme assure qu'il n'y a pas de répétition des erreurs précédentes et tout se fait par des opérations locales.

4.4.3 Le codage complémentaire et l'impact du code Isométrique

Dans certains cas, la réinitialisation ne garantit pas le fonctionnement correct du réseau. Cette situation survient lors qu'un patron d'entrée Ia(i) est un sous-ensemble d'un autre patron d'entrée Ia(j) et que les deux appartiennent à différentes classes. Le problème est que l'apprentissage est trop lent, il faut souvent plusieurs "époques" pour que le réseau converge (Carpenter, Grossberg, 1991). Le codage complémentaire est la solution proposée par Carpenter et Grossberg (1991) pour résoudre ce problème. Le codage complémentaire consiste à présenter au réseau le patron d'entrée Ia mais aussi son complémentaire, soit la paire :

$$(I_a, I_a^c) = (a_1, a_2, a_3, ..., a_{Ma}, a_1^c, a_2^c, a_3^c, ..., a_{Ma}^c) \qquad a_i^c = 1 - a_i$$

La longueur du signal d'entrée de ARTa est ainsi doublée, mais l'amplitude des signaux d'entrée devient constante (Ma) pour tous les signaux.

Revenons au codage proposé à la section précédente, soit le code Isométrique. Il suffit d'observer le code pour s'apercevoir qu'il est impossible d'avoir un vecteur d'entrée qui soit un sous-ensemble d'un autre vecteur d'entrée. Par conséquent, il n'est pas nécessaire d'utiliser le codage complémentaire si on a codé le signal d'entrée à l'aide du code Isométrique. L'amplitude de tous les codes est égale à 4, donc constante et elle demeurera constante. Georgiopoulos et son équipe confirment (Georgiopoulos, Huang, Heileman, 1994) que le codage complémentaire n'est pas nécessaire si les patrons d'entrée ont le même nombre de "1".

On peut conclure que le code Isométrique qui a été conçu pour que les séquences puissent être traitées par le réseau ART1 s'avère également très approprié pour l'architecture ARTMAP binaire.

4.4.4 Algorithme d'apprentissage du réseau ARTMAP

L'algorithme de simulation présente sommairement des explications données par Carpenter et Grossberg (1991) sur le traitement effectué par ARTMAP lors d'entraînement. La figure 4.12 (pages 1 et 2) présente le pseudocode de cet algorithme. La section suivante décrit tous les détails de l'implantation de ARTMAP dans le cadre de ce projet.

Étape 0. Initialiser les paramètres:

- Initialiser ρa* à sa valeur de départ : ρa*=0.1
- Fixer la valeur de ρb=0.1
- Les couches des prototypes de ARTa et ARTb (F2a et F2b) sont vides et aucune association n'est faite dans Inter-ART.

Étape 1. Présentation de la première paire des vecteurs d'entraînement $(I_a^1, I_b^{1)})$.

- I_a^1 est traité par le module ARTa et le premier prototype est créé dans F2-a.
- I_b^1 est traité par le module ARTb et le premier prototype est créé dans F2-b.
- Création de l'association correspondante dans le champ de correspondance Inter-ART : le prototype de F2-a est lié au prototype de F2-b.

Étape 2. Pour chaque paire de vecteurs d'entraînement (I_a^p, I_b^p) où p est le numéro de la paire dans la séquence d'entraînement.

- Faire les Étapes 3 et 4.

Étape 3. Présentation de la paire des vecteurs d'entraînement suivants (I_a^p, I_b^p).

- I_a^p est traité par le module ARTa.
- I_b^p est traité par le module ARTb.

Figure 4.12 Pseudocode de l'algorithme d'entraînement du réseau ARTMAP.

Étape 4. Vérifier si le module ARTa a créé un nouveau prototype.

- Si OUI
  - Création de l'association correspondante dans le champ de correspondance Inter-ART : le prototype de F2-a est lié au prototype de F2-b.
  - Retourner à l'étape 3.
- Si NON
  - Vérifier si les neurones actifs de ARTa et ARTb ont été déjà associés l'un à l'autre .
    - Si OUI
      - Apprentissage : le neurone actif de ARTa modifie son prototype pour inclure le patron d'entrée I_a^p (voir ART1).
      - Retourner à l'étape 3.
    - Si NON Activation du mécanisme de suivi de correspondance :
      - $\rho_a^{nouveau} = \rho_a^{calculé} + 0.001$.
      - I_a^p est traité par le module ARTa avec la nouvelle valeur de ρa.
      - Retourner à l'étape 4.

Figure 4.12 (suite)

4.4.5 ARTMAP dans notre projet

Cette section apporte les précisions nécessaires reliées aux détails liés à l'emploi du réseau ARTMAP dans notre projet en spécifiant les signaux d'entrée pour les modules ARTa et ARTb respectivement, ainsi que les valeur des paramètres du reconnaisseur de commandes gestuelles.

Dans le but de tester le reconnaisseur de commandes gestuelles, un ensemble de gestes défini au chapitre 3 a été présenté (tableau 3.8). Comme il a déjà été expliqué, il existe deux modes de traitement : le mode d'apprentissage et le mode de classification. Lors de la classification, on présente le patron à classifier et on attend que le réseau propose un classement. Le signal d'entrée pour le reconnaisseur de commandes gestuelles est évidemment la forme que l'utilisateur de l'éditeur dessine sur l'écran de l'ordinateur (Stylistic 1000).

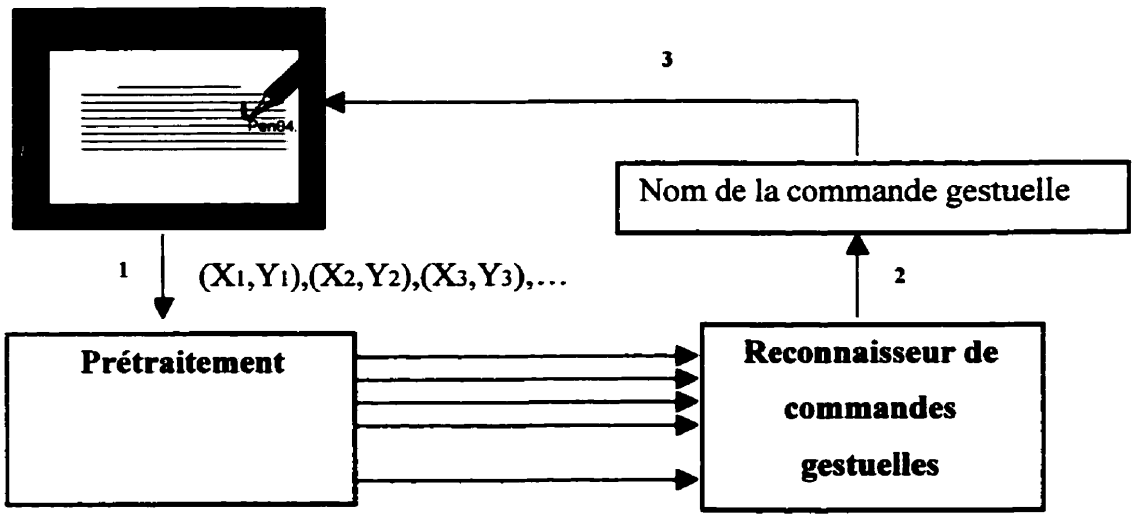


Figure 4.13 Le réseau ARTMAP fonctionne en mode de reconnaissance.

Le réseau traite ce signal : soit qu'il attribue une étiquette qui est le nom de la commande gestuelle (voir figure 4.13) ou soit qu'il affirme ne pas être en mesure de

classifier la forme. Il faut alors introduire à cette étape un nouveau paramètre du réseau ARTMAP – le critère de vigilance Ro. Sa valeur spécifie la correspondance minimale ou un seuil (entre la forme inconnue et le prototype qui l'a classifié) exigé pour que le réseau propose un classement. Si sa valeur est petite, le réseau aura moins de formes inconnues et fera plus d'erreurs. Une grande valeur conduira toutefois, à plus de rejets mais également à une plus grande précision. La valeur du critère de vigilance de notre reconnaisseur a été fixée à 0.5 ce qui signifie qu'un prototype classifie une forme donnée si plus que la moitié de ces bits sont identiques de ces (bits) de la forme à classifiée.

Dans ce projet, deux schémas d'apprentissage ont été utilises: l'apprentissage en-ligne et l'apprentissage hors-ligne. L'apprentissage hors-ligne (figure 4.14) est utilisé pour s'assurer que l'éditeur gestuel est fonctionnel, donc capable d'agir et exécuter l'ensemble de commandes gestuelles tel quel défini au deuxième chapitre, tandis que l'apprentissage en-ligne (figure 4.15) est utilisé pour adapter (personnaliser) les gestes aux préférences d'un utilisateur donné.

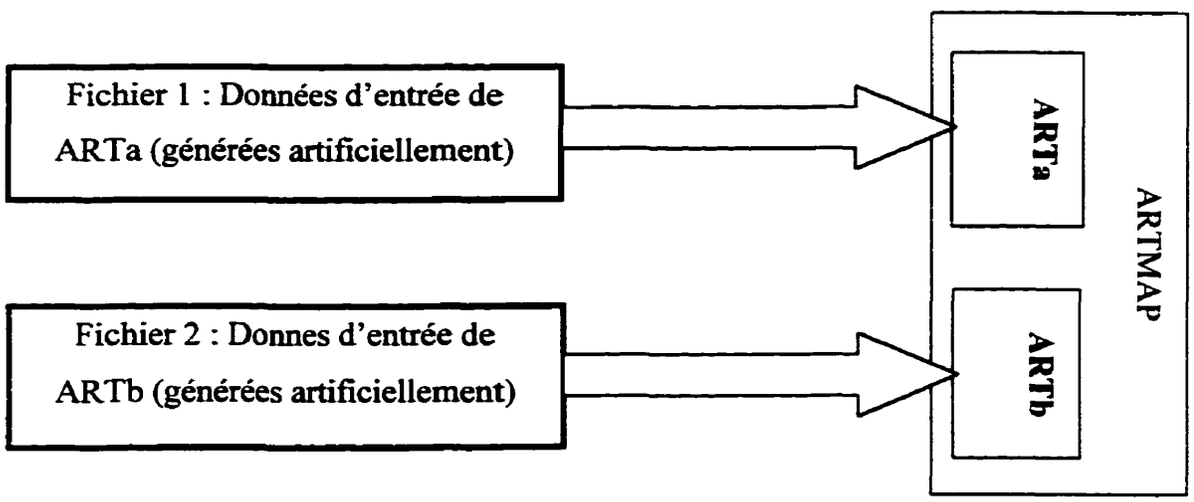


Figure 4.14 Apprentissage hors-ligne.

- Pendant l'apprentissage hors-ligne, la séquence des paires (I_a^p, I_b^p), p=1,2,3,…,P est présentée (P est le nombre de patrons d'entraînement, Ia – entrée de

module ARTa, Ib – entrée de module ARTb) à l'aide de deux fichiers :

- Le premier fichier contient les données nécessaires au module ARTa.
- Le deuxième fichier contient les données nécessaires au module ARTb.

Il s'agit de codes binaires dans les deux cas. La différence avec l'apprentissage en-ligne est que les données sont générées ou artificielles, donc le prétraitement ne fait pas partie du traitement. Le prétraitement est une étape d'analyse. La figure 4.14 illustre les deux modules du réseau ARTMAP – ARTa et ARTb qui reçoivent ses entrées de deux fichiers[12].

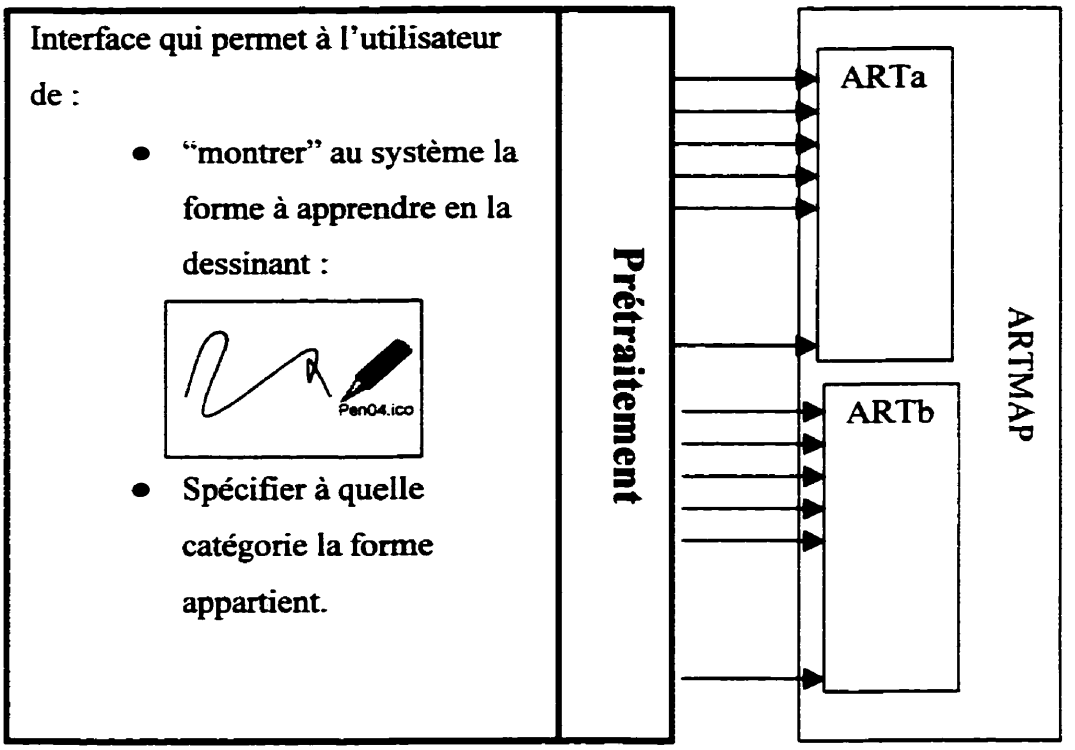


Figure 4.15 Apprentissage en-ligne.

➢ Pendant l'apprentissage en-ligne, la forme à apprendre est spécifiée à l'aide d'un exemple et la catégorie à laquelle elle appartient est choisie dans une liste des

[12] Inp-ARTa.txt et inp-ARTb.txt sont les noms de ces fichiers dans l'application.

catégories possibles. Dans ce cas, il nous faut un système qui inclut le prétraitement, ARTMAP reçoit donc deux codes binaires :

- Le code de la forme à apprendre ;
- Le code de la catégorie à laquelle la forme doit être associée.

4.4.5.1 Les données d'entrée du réseau ARTa

Lors de l'apprentissage hors-ligne, les commandes gestuelles sont présentées au réseau dans l'ordre spécifié par la première colonne du tableau 4.4. La deuxième colonne montre les formes des commandes gestuelles, la troisième colonne donne les codes de Freeman des 10 parties de chaque geste. Les directions de Freeman sont les suivantes : le code 1 est associé si la direction de mouvement est entre 2π et $\pi/4$, le code 2 est associé si la direction est entre $\pi/4$ et $\pi/2$, etc.

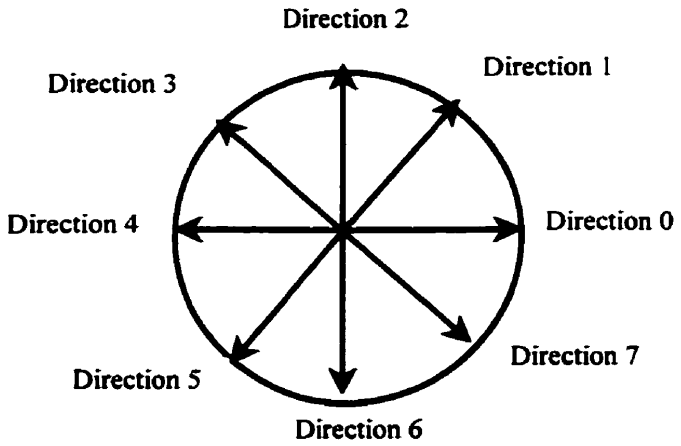
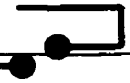
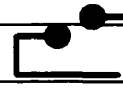
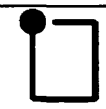
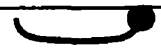


Figure 4.16 Directions de Freeman.

Les séquences de directions de Freeman sont codées à l'aide du code Isométrique pour construire la chaîne binaire qui constitue le patron d'entrée pour le module ARTa. Donc sa longueur est N*8 = 80 (N est le nombre de parties de longueur égale dans chaque geste). Les tableaux de l'annexe I contiennent les résultats obtenus lors de l'entraînement du réseau ARTMAP.

Tableau 4.4 Les codes qui sont utilisés pour coder les formes (les entrées du module ARTa).

#	Forme	Code (Directions de Freeman)										Nom de la commande
		1	2	3	4	5	6	7	8	9	10	
1		0	0	0	6	6	6	4	4	4	4	"Nouvelle ligne"
2		0	0	6	6	6	4	4	2	2	2	"Image braille"
3		0	0	0	2	2	2	4	4	4	4	"Nouvelle ligne"
4		4	4	4	6	6	6	0	0	0	0	"Nouvelle ligne"
5		4	4	4	2	2	2	0	0	0	0	"Nouvelle ligne"
6		6	6	6	6	0	0	0	0	0	0	"Disjoindre "
7		6	6	6	0	0	2	2	2	4	4	"Image braille"
8		7	7	2	2	7	7	2	2	7	7	"Effacer"
9		1	1	1	1	1	7	7	7	7	7	"Insérer"
10		3	3	3	3	3	5	5	5	5	5	"Insérer"
11		2	2	3	3	3	4	5	5	6	6	"Annuler la dernière action"
12		7	7	7	0	0	0	0	1	1	1	"Joindre"
13		7	7	7	7	1	1	1	1	1	1	"Palette d'édition"
14		5	5	5	4	4	4	4	3	3	3	"Joindre"

4.4.5.2 Les données d'entrée du réseau ARTb

Le tableau 4.5 présente les codes qui sont utilisés pour coder les catégories des formes, donc les entrées du module ARTb. On utilise un code binaire de huit bits dans lequel chaque mot de code a un seul "1" (rappelons que le nombre de catégories est huit). Pour choisir le code on s'est basé sur les caractéristiques du réseau ART1 comme classificateur de signaux binaires. La cinquième caractéristique citée à la section (4.2.3, Caractéristiques du réseau ART1) stipule que la variation permise dans une catégorie dépend de l'amplitude de son prototype. Ce fait a été considéré lorsqu'il s'agit de définir un code qui permet la convergence rapide du réseau ARTb. En choisissant un code qui a un seul "1", il était possible de réduire au minimum la variation permise dans une catégorie donnée. Le réseau est donc plus sensible aux variations et à la formation d'un nouveau prototype lors de la présentation d'un patron d'entrée inconnu. La nécessité de tester d'autres hypothèses est donc court–circuitée.

Tableau 4.5 Les codes qui sont utilisés pour coder les catégories des formes (les entrées du module ARTb).

#	Nom de fonction	Code binaire							
		1	2	3	4	4	5	7	8
1	Nouvelle ligne	1	0	0	0	0	0	0	0
2	Insérer	0	1	0	0	0	0	0	0
3	Joindre	0	0	1	0	0	0	0	0
4	Disjoindre	0	0	0	1	0	0	0	0
5	Palette d'édition	0	0	0	0	1	0	0	0
6	Effacer	0	0	0	0	0	1	0	0
7	Annuler la dernière action	0	0	0	0	0	0	1	0
8	Image braille	0	0	0	0	0	0	0	1

4.4.5.3 Les valeurs des paramètres du reconnaisseur ARTMAP

Les valeurs des paramètres de notre reconnaisseur de commandes gestuelles sont spécifiées au tableau 4.6. Certaines sont arbitraires comme par exemple, le nombre de segment de chaque forme. Nous avons fixé N à dix. Toutefois, si les formes sont plus complexes, il est possible d'augmenter ce nombre afin d'avoir une précision suffisante. La valeur de départ du critère de vigilance du module ARTa (ρa^*) est celle suggérées par (Georgiopoulos, Huang, Heileman, 1994). La valeur du critère de vigilance du module ARTb est 0.1 qui est la valeur recommandée par Carpenter et Grossberg (1991).

Tableau 4.6 Les valeurs des paramètres du reconnaisseur des commandes gestuelles.

Désignation	Signification	Valeur
P	Nombre de patrons d'entraînement (hors-ligne).	14
Na	Dimension des vecteurs d'entrée du ARTa.	80
Nb	Dimension des vecteurs d'entrée du ARTb.	8
N	Nombre des parties de longueur égales.	10
R	Rayon de rééchantillonnage équidistant.	15
ρa^*	Critère de vigilance de départ pour le réseau ARTa, mode d'apprentissage.	0.1
ρb	Critère de vigilance pour le réseau ARTb, mode d'apprentissage.	0.1
Ro	Critère de vigilance de ARTa, mode de reconnaissance.	0.5

4.5 L'éditeur gestuel adaptable

Une fois que le reconnaisseur des commandes gestuelles est créé, qu'il a été entraîné à classifier un ensemble prédéfini de gestes à partir des données générées et artificielles. Ceci a permis d'obtenir un reconnaisseur compact et rapide, il ne reste plus qu'à le rendre adaptable. Ce reconnaisseur basé sur le réseau ARTMAP peut être adapté relativement aisément aux besoins de chaque utilisateur. Dans le contexte d'édition gestuelle, nous voulons permettre aux utilisateurs de personnaliser l'ensemble de gestes existant. En tenant compte des caractéristiques du réseau ARTMAP en tant que classificateur incrémental (section 4.2.3 Caractéristiques du réseau ART1, voir la première caractéristique) il est possible d'ajouter un nouveau geste sans corrompre les connaissances déjà acquises. Pour atteindre cet objectif, il a été nécessaire de développer une application spécialisée.

4.5.1 L'application qui permet l'ajout des nouveaux gestes d'édition

L'analyse des besoins a été la première étape de la conception de cette application. Le but est d'ajouter une nouvelle commande gestuelle dans l'ensemble de commandes gestuelles qui sont connues par le reconnaisseur. Deux informations doivent être spécifiées, soit :

- la forme de la nouvelle commande gestuelle ;
- l'action à la quelle cette forme doit être associée.

Pour ajouter une nouvelle forme gestuelle, il faut évidement connaître les formes déjà connues par le reconnaisseur. Donc lors d'un ajout, il faut être en mesure de visualiser les formes connues pour que l'utilisateur puisse choisir une nouvelle forme. La deuxième question qu'on se pose est : "Comment l'utilisateur peut-il "montrer" au système sa forme ?". La façon la plus naturelle de le faire est de la dessiner. Pour spécifier alors l'action correspondante à cette nouvelle forme, il faut que l'utilisateur ait

la liste des actions disponibles et en choisisse une. En résumé, l'interface doit avoir un champ de visualisation de formes connues, un champ de dessin et un champ avec la liste des actions (fonctionnalités de l'éditeur gestuel qui puissent être convoquées à partir des gestes manuscrits) et bien sûr une façon de déclencher le processus d'ajout.

La figure 4.17 montre l'interface qui permet l'ajout des gestes. Cependant, les changements (ajouts) dans l'ensemble des gestes affectent le profil personnel de l'utilisateur. Ils sont mémorisés et à partir de ce moment, ils deviennent disponibles pour l'édition gestuelle. L'ajout d'un geste se fait en suivant les étapes illustrées par la figure 4.15 – "Apprentissage en-ligne", donc le prétraitement décrit en section 4.3.1 est exécuté et les codes binaires sont générés et présentés au réseau en suivant l'algorithme d'entraînement décrit par le Pseudocode de la figure 4.12. Bien sûr, avant d'entraîner le réseau avec le nouveau geste, le reconnaisseur doit être préparé. Les informations nécessaires sont extraites du profil personnel de l'usager pour que la nouvelle forme puisse être présentée.

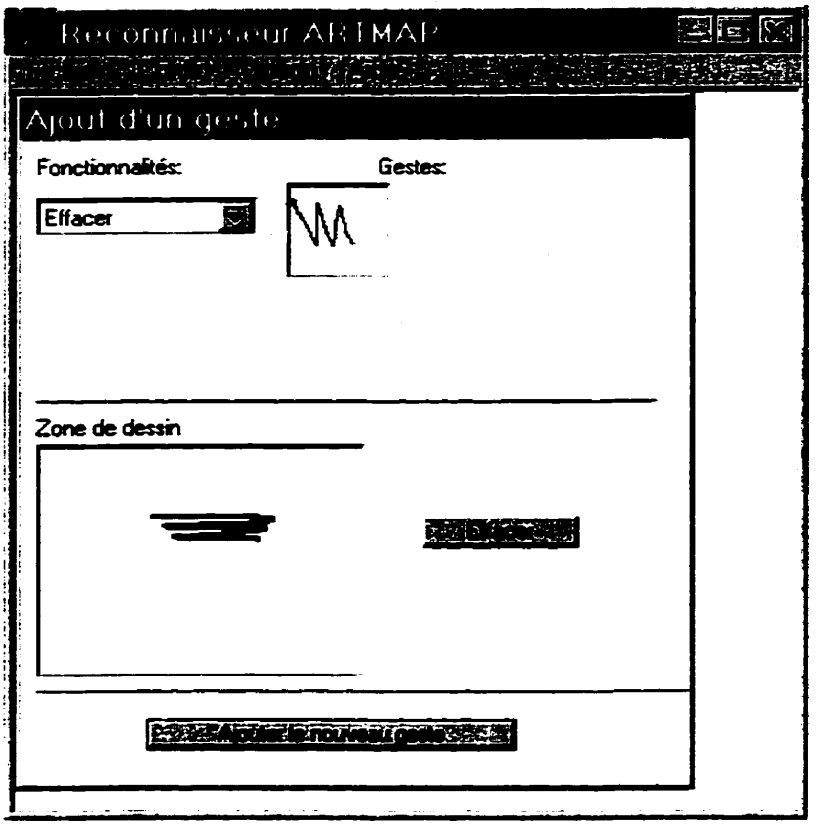


Figure 4.17 L'interface qui permet l'ajout des gestes.

4.5.2 D'autres caractéristiques de l'application

L'algorithme de reconnaissance de commandes gestuelles tient compte de la direction d'écriture de chaque tracé. Il est donc important de montrer à l'usager du système comment écrire les commandes gestuelles ainsi que de lui permettre de pratiquer et mémoriser les formes. Toutes ces considérations nous ont amenés à inclure dans cette même application des utilitaires supplémentaires. La figure 4.18 montre l'interface qui aide l'utilisateur à apprendre les associations "forme-action". L'usager peut choisir une fonctionnalité dans la liste de fonctionnalités et voir des images de formes qui sont

associées à la fonctionnalité choisie. En cliquant sur une image donnée, le système lui "montre" comment écrire la forme dans la zone de dessin en la dessinant à basse vitesse. Le moniteur gestuel peut être vu comme une façon d'augmenter la fiabilité (et par conséquent, la satisfaction de l'utilisateur) de notre système parce qu'un usager bien formé fera moins d'erreurs. Notre moniteur gestuel est une application différente de l'éditeur gestuel qui permet d'être utilisé comme un aide-mémoire, disponible à tout instant.

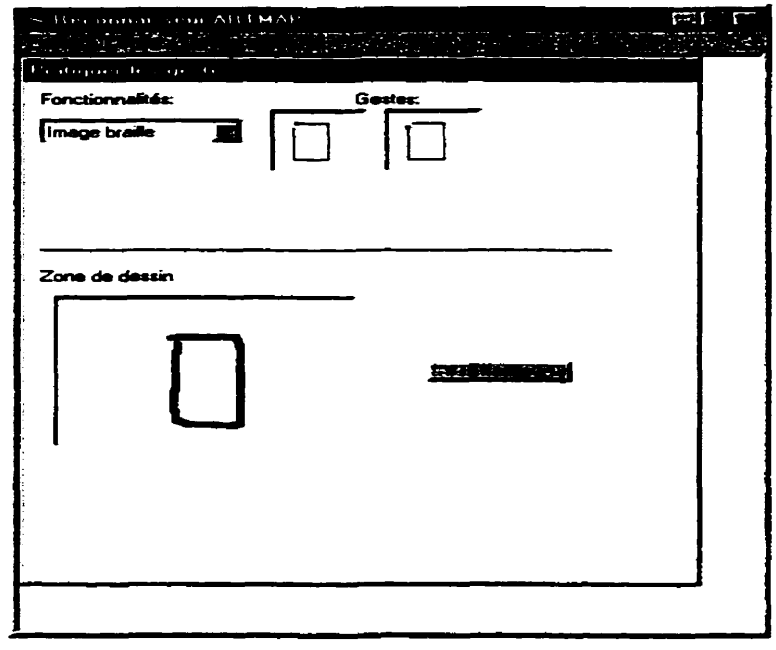


Figure 4.18 L'interface du moniteur gestuel.

De plus, la dextérité des usagers et le reconnaisseur de commandes gestuelles pourraient tous les deux être testés. La figure 4.19 montre cette interface. L'utilisateur dessine des formes dans la zone de dessin et appuie sur le bouton "Activer le reconnaisseur", le nom de la commande telle qu'elle a été reconnue est alors écrite dans le champ "Nom du geste". La possibilité d'utiliser le reconnaisseur de gestes séparément avant d'utiliser l'éditeur gestuel permet à l'usager de se familiariser progressivement avec les nouveautés. Elle représente aussi une façon d'augmenter la confiance de l'utilisateur dans l'interface. En effet, ce type de communication (écrire des commandes sur un écran) sera probablement une chose nouvelle pour la majorité des usagers.

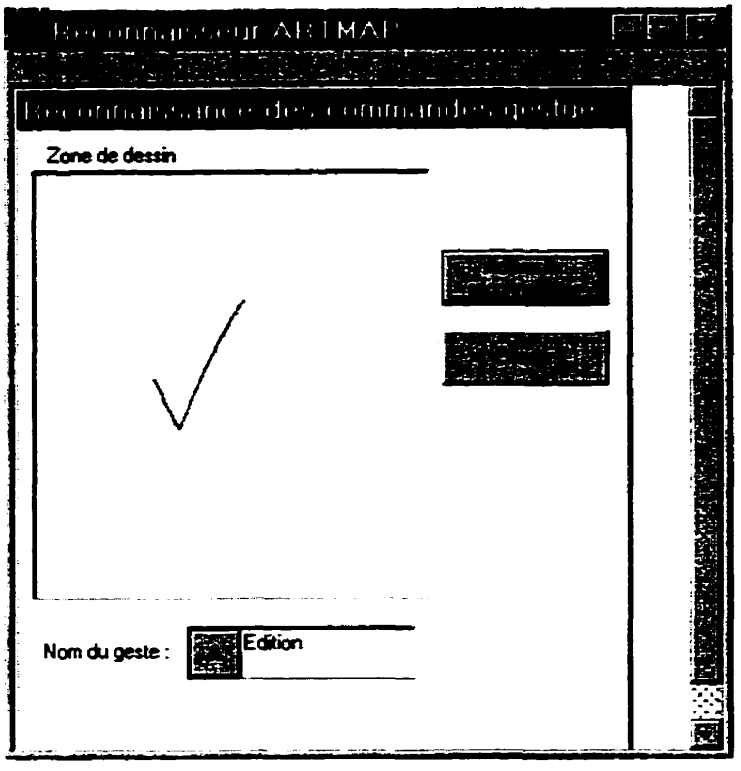


Figure 4.19 Interface qui permet de tester le reconnaisseur de gestes.

4.6 Conclusion

Ce chapitre a présenté un reconnaisseur de commandes gestuelles, basé sur une approche connexionniste : le réseau ARTMAP. Les algorithmes d'implantation, les détails techniques, les paramètres et leurs significations ainsi que leurs valeurs ont été précisés. On a aussi introduit un nouveau code – "Le Code Isométrique". Ce code a permis la classification des commandes gestuelles par un réseau ART1. Le même code a été utilisé à l'étape de prétraitement pour le classificateur supervisé ARTMAP. Encore cette fois, l'application de ce code nous a apporté des avantages : il n'a pas été nécessaire d'effectuer un codage complémentaire et une seule présentation de l'ensemble d'entraînement suffisait pour stabiliser l'apprentissage. Ce qui en définitive, réduit la

longueur des signaux d'entrée et assure la rapidité de la convergence du réseau ARTMAP.

Le cinquième chapitre donne les explications nécessaires reliées aux expérimentations. Une analyse des résultats obtenus sur ce reconnaisseur gestuel y est également présentée.

CHAPITRE V
VÉRIFICATION EXPÉRIMENTALE ET DISCUSSIONS

Ce chapitre est dédié à la présentation et à l'analyse des résultats de la vérification expérimentale. Le reconnaisseur de commandes gestuelles est l'élément clé d'un éditeur gestuel, sa fiabilité est donc une qualité très importante. Pour cette raison, il a fallu déterminer les valeurs des paramètres du reconnaisseur et l'évaluer. La base de données utilisée sera ensuite explicitée, pour enchaîner avec un compte rendu dans lequel des observations objectives seront fournis de même que des explications et des commentaires sur les résultats obtenus. La conclusion portera principalement sur les problèmes rencontrés.

5.1 Les paramètres du reconnaisseur gestuel

La vérification expérimentale vise à déterminer l'impact des paramètres du reconnaisseur gestuel sur ses performances. Il faut préciser qu'à nos yeux le prétraitement fait partie du reconnaisseur, rappelons qu'il a été spécialement conçu pour permettre la classification des séquences spatio-temporelles à l'aide d'un réseau ARTMAP. Les paramètres dont il fallait tenir compte sont les suivants:

- N - la longueur du signal d'entrée du réseau ARTMAP. N détermine le nombre de segments utilisés pour représenter les formes et les prototypes par le réseau ARTMAP.
- Ro – le critère de vigilance du réseau ARTMAP (en mode de reconnaissance).

On s'est intéressé de plus à savoir comment le taux de reconnaissance est affecté par ces deux paramètres.

$$TauxDeReconnaissance = f(N)$$

$$TauxDeReconnaissance = f(Ro)$$

Il restait finalement à tester deux types de prétraitement (prétraitement A et B) qui ont été présentés dans le quatrième chapitre (dans la section 4.3.1) pour n'en choisir qu'un seul, par lequel on obtiendra les meilleurs résultats.

$$TauxDeReconnaissance = f(A)$$
$$TauxDeReconnaissance = f(B)$$

5.1 La création de la base de données

La vérification expérimentale exige qu'on dispose d'une base de données. C'est en créant une application que nous avons pu constituer notre base de gestes. L'accessibilité et la possibilité de leur réutilisation future par d'autres applications sont deux caractéristiques importantes qu'on doit retrouver dans une base de données. Voilà pourquoi nous l'avons conçue sous forme de base de données ACCESS[13].

La base de données contient deux tableaux : "Gestes" et "Résultats de reconnaissance" (tableau 5.1). Le tableau "Gestes" contient les données brutes – données de l'acquisition ainsi que les informations sur la forme demandée et sur le scripteur. Le tableau "Résultats de reconnaissance" contient les identificateurs qui permettent de déterminer la performance du reconnaisseur en fonction des critères mentionnés au début du chapitre. Le tableau 5.1contient les liste des noms des colonnes pour nos deux tableaux.

[13] ACCESS fait partie de Microsoft Office'97

Tableau 5.1 Base de données ACCESS.

Noms de tableau	
"Gestes"	**"Résultats de reconnaissance"**
ID(numéro d'identification)	ID(numéro d'identification)
Numéro de scripteur	ID du tableau "Gestes",
Nom du scripteur	Code (du geste) lu
Numéro de séance	Code (du geste) reconnu
Code du geste	Nom (du geste) reconnu
Nombre de points	Ro (la valeur du critère de vigilance)
X1	N (la valeur du N)
Y1	Résultat ("True" ou "False")
X2	
Y2	
.	
.	
.	

5.1.1 Acquisition des données

La figure 5.1 montre la fenêtre d'interface utilisée pour l'acquisition des données. Le scripteur voit la forme à dessiner dans le coin supérieur gauche. Il est appelé à la dessiner dans la zone du coin inférieur droit. Cette disposition des éléments de l'interface ne permet pas au scripteur de copier la forme "modèle" en se servant de la colinéairité mais plutôt en reproduisant la forme comme il l'a conçoit. Si le geste dessiné le satisfait, il pèse sur le bouton "Enregistrer" et ajoute l'information dans la base de données.

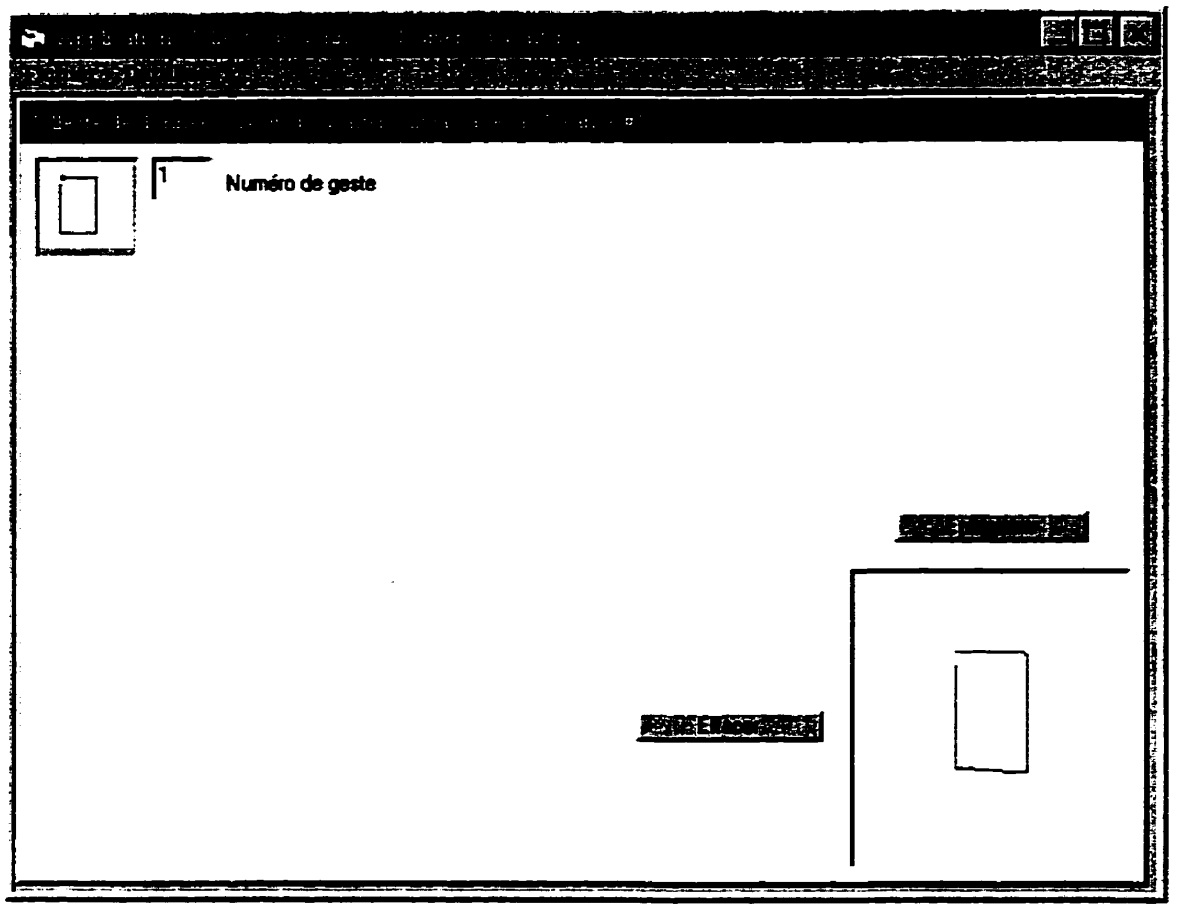


Figure 5.1 L'interface simple de l'application *Collecte de données*.

Quatorze scripteurs ont participé à l'expérience. Chaque scripteur a écrit les quatorze formes, trois fois. L'ordre de la présentation des formes différait pour chaque scripteur et à chaque séance. De cette façon, on a pu constituer une base de données qui contient 14*14*3 = 588 formes, donc 14*3=42 exemples de chaque forme. Cette base de données a servi aux analyses subséquentes.

On a pu remarquer que la dextérité des scripteurs s'améliore après chaque séance car ils se sentent de plus en plus à l'aise avec les formes. Cette expérience leur a servi d'entraînement. Un tel exercice sera très utile pour les futurs utilisateurs de l'éditeur

gestuel qui n'ont jamais travaillé avec un bloc-notes électronique. Le temps nécessaire pour effectuer ce test était de 3 minutes.

5.2 Résultats de la vérification expérimentale

Le reconnaisseur de commandes gestuelles est basé sur le réseau ARTMAP. Les tests ont permis de voir comment le réseau se comporte en fonction des différents paramètres. Pour faire cela, on a calculé le taux de reconnaissance (TdR) qui est en fait le nombre de gestes reconnus avec succès divisé par le nombre de gestes à classifier.

$$TdR = \frac{\sum_{1}^{NG} GRS}{\sum_{1}^{NG} GR}$$

où NG est le nombre de gestes dans la base des données, GRS est le geste reconnu avec succès et GR est le geste à reconnaître.

Les figures 5.2 et 5.3 présentent sous forme graphique les résultats obtenus. Les tableaux de l'annexe II contiennent les données utilisées pour réaliser les figures de ce chapitre. La figure 5.2 présente le taux de reconnaissance des commandes gestuelles en fonction de Ro et du nombre de segments lorsque le prétraitement de type A est utilisé. La figure 5.3 présente le taux de reconnaissance des commandes gestuelles en fonction de Ro et du nombre de segments lorsque le prétraitement de type B est utilisé.

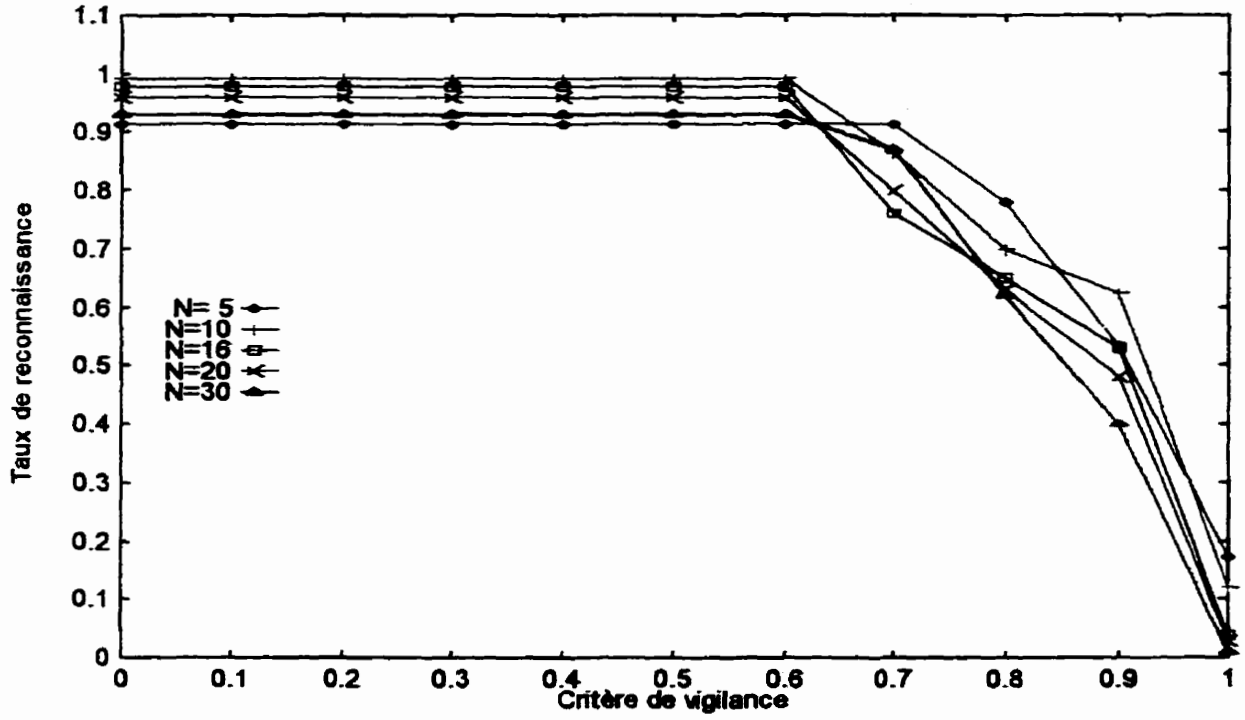


Figure 5.2 Taux de reconnaissance en fonction de N et Ro. Type de prétraitement : A.

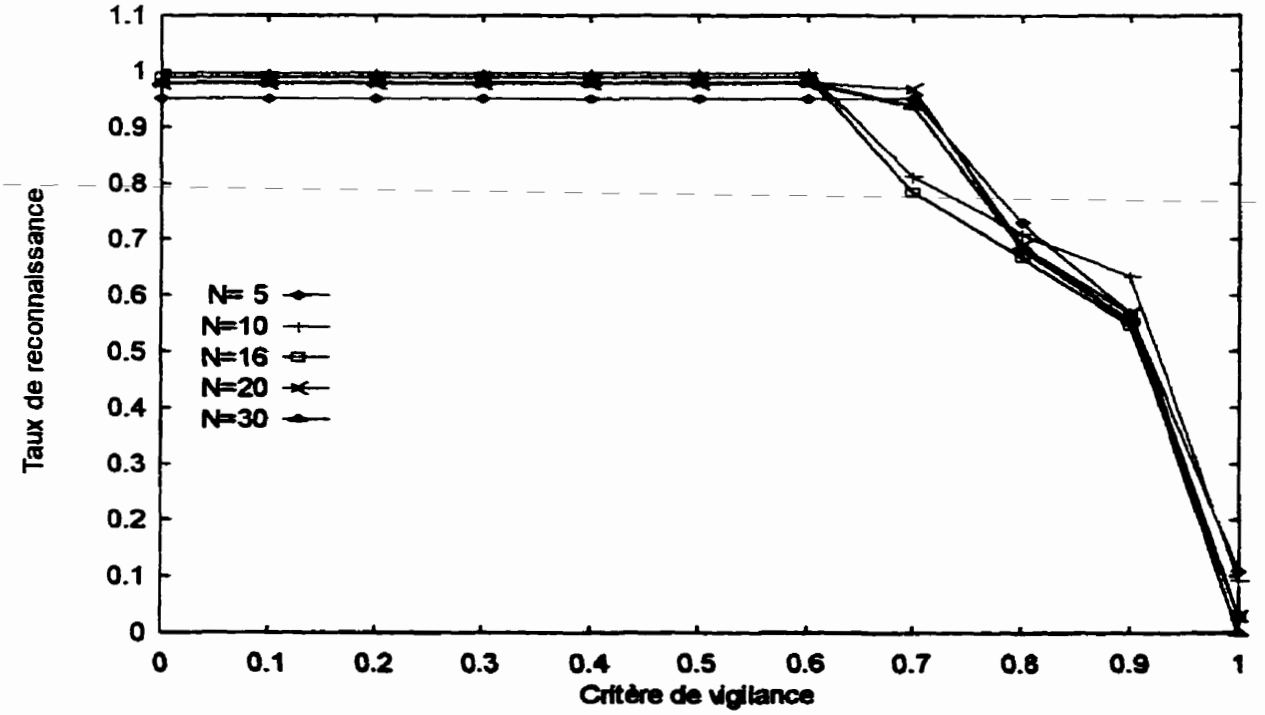


Figure 5.3 Taux de reconnaissance en fonction de N et Ro. Type de prétraitement : B.

En observant les deux groupes de graphiques (figures 5.2 et 5.3) on s'aperçoit que :

- le prétraitement B est plus performant;
- la différence de performance entre les deux types de prétraitement augmente avec N et la variation est plus grande si on utilise le prétraitement A

$$\Delta\%Rec_B < \Delta\%Rec_A;$$

- les graphiques sont plats jusqu'à Ro=0.6 et jusqu'à Ro=0.7 pour N=5;
- les meilleurs résultats sont obtenus pour N=10.

On constate qu'en utilisant le prétraitement de type B, les résultats obtenus sont meilleurs. Le prétraitement de type B a donc été retenu pour la version finale du reconnaisseur. A la lumière de nos tests, le prétraitement de type B semble être plus performant parce que la compilation du vecteur d'entrée pour le réseau ARTMAP dans ce cas se fait le plus tard possible. En faisant cela, on évite de prendre des décisions arbitraires pouvant introduire des erreurs de classification.

La différence dans la performance entre les deux types de prétraitement augmente avec l'augmentation de N. On remarque que le taux de reconnaissance varie moins en fonction de N à condition qu'on utilise le prétraitement de type B. Pour Ro=0.5, la variation entre la meilleure (N=10) et la pire (N=30) performance pour le prétraitement de type A est de 3.3 %, tandis que cette variation est de 0.2 % pour le type B. On peut donc déduire que la performance du reconnaisseur dépend moins du paramètre N si on utilise le prétraitement B.

Le réseau ARTMAP dans le cadre de notre reconnaisseur démontre une stabilité remarquable peu importe la valeur de N. Les plateaux (figures 5.2 et 5.3) sont présents pour toutes les valeurs de N et le plateau est plus large pour N=5 (jusqu'à Ro=0.7). Le taux de reconnaissance se dégrade lentement en augmentant Ro et les meilleurs résultats sont calculés pour N=10 dans les deux prétraitement considérés.

On obtient les meilleures performances pour N=10. Contrairement à ce qu'on pouvait envisager, l'augmentation de N ne mène pas nécessairement à l'augmentation du taux de reconnaissance. Un nombre plus grand de N peut être vu comme un nombre de points de contrôle plus grand sur la forme. Par conséquent, l'introduction de trop de points de contrôle sur les formes mène vers un système plus rigide, plus strict et qui aura tendance à faire plus de rejets. Sur un certain plan, l'augmentation de N est "semblable" à l'augmentation du critère de vigilance (Ro) du réseau ARTMAP.

Il existe néanmoins un nombre minimal de N (Nmin) qui permet une représentation interne adéquate pour un ensemble de formes donné. Selon nous, sa valeur est le maximum du nombre minimal de segments nécessaires pour représenter chaque forme "modèle".

$$N_{\min} = \underset{i=1}{\overset{NF}{Max}} \left(N_{Min}^{Fi} \right)$$

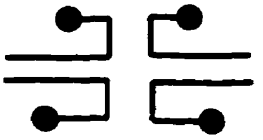
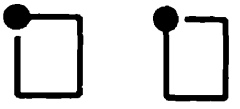
NF correspond au nombre des formes dans l'ensemble des commandes gestuelles (par exemple notre ensemble de commandes gestuelles comporte 14 formes différentes) et F_i désigne une forme donnée de cet ensemble. N_{Min}^{Fi} est le nombre minimale de segments pour avoir une représentation adéquate calculée pour une forme donnée de l'ensemble des gestes. N_{Min}^{Fi} est calculé ainsi :

$$N_{Min}^{Fi} = \sum_{i=1}^{CD+1} \left(\frac{l_i^+}{l_e} \mid l_i^+ \bmod l_e = 0, \forall l_i \right)$$

Dans cette formule, CD désigne le nombre de changements de direction. Dans un geste pour chaque segment (le nombre de segments est égale au nombre de changement de direction plus 1) on exprime sa longueur logique (l_i^+ pour i-ième segment) en longueur élémentaires l_e, d'une telle façon que ($l_i^+ \bmod l_e$)=0.

Le tableau qui suit aide à comprendre le sens des paramètres calculés sur l'ensemble des commandes gestuelles qui a servi à tester le réseau ARTMAP.

Tableau 5.2 Les calculs de Nmin sur l'ensemble de teste de commandes gestuelles.

Formes	**Nombre de changements de direction +1**	N_{Min}^{Fi}
	5	5
	2	2
	2	3
	3	5
	3	7
	2	8
	5	10
	4	10
		Max(2,3,5,7,10)=10

Considérant que Nmin est égale à dix, on peut conclure qu'il n'est pas possible de bien représenter nos formes en termes de prétraitement déjà défini si N<10.

En observant les figures 5.2 et 5.3, on s'aperçoit que les taux de reconnaissance les moins bons sont calculés pour N=5. Pour Ro=0.5, ils sont 91,5% pour le type A et 95.2 % pour le type B. On réalise aussi que pour toutes les autres valeurs de N (10, 16, 20,

30) on obtient des résultats très satisfaisants. L'explication, qui en découle est simple : pour N=5 les prototypes ne sont pas adéquatement représentés (ils sont déformés). La figure 5.4 montre un exemple de l'impact de la segmentation dans la présentation d'une forme.

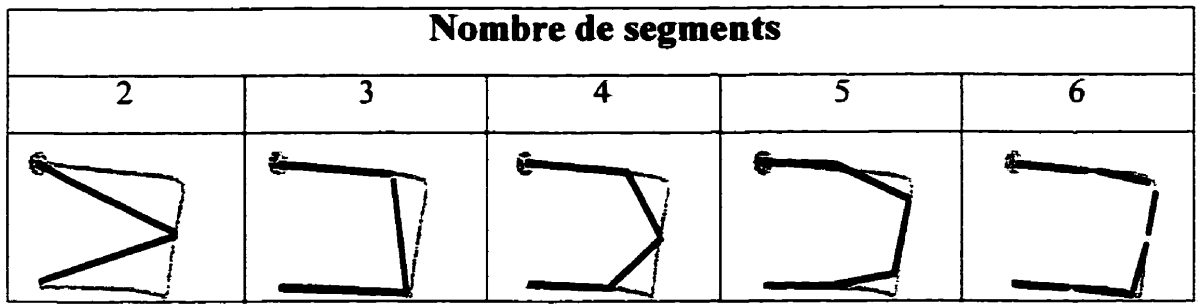
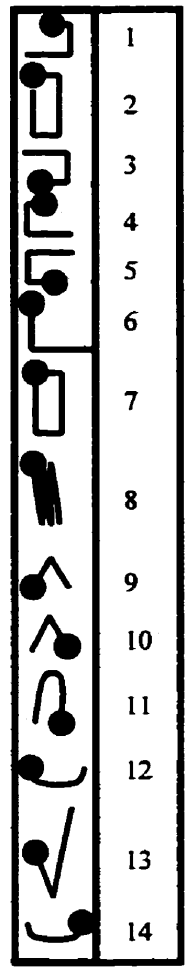


Figure 5.4 Un exemple de segmentation.

Il faut dire que le prétraitement de type B exige moins de calculs, cependant on a constaté que dans les deux cas (prétraitement A et prétraitement B) la reconnaissance d'une commande gestuelle est quasi-instantanée.

Une analyse plus détaillée du taux de reconnaissance en fonction de la forme testée et de N a produit les graphiques qu'on retrouve aux figures 5.5 (prétraitement A) et 5.6 (prétraitement B). Ro a été fixé à 0.5, pour l'ensemble des analyses. Les formes sont numérotées selon l'ordre de leur présentation au réseau (tableau 4.5). La figure 5.6 présente les formes de 8 à 10 seulement parce que le taux de reconnaissance pour les formes de 1 à 8 est 100% pour toutes les valeurs de N.

On obtient une fois de plus, le meilleur taux de reconnaissance à N=10, indépendamment du type de prétraitement, et les variations sont plus grandes pour le prétraitement de type A.
Le bar qui content les formes numérotées facilite la compréhension des figures 5.5 et 5.6.


1
2
3
4
5
6
7
8
9
10
11
12
13
14

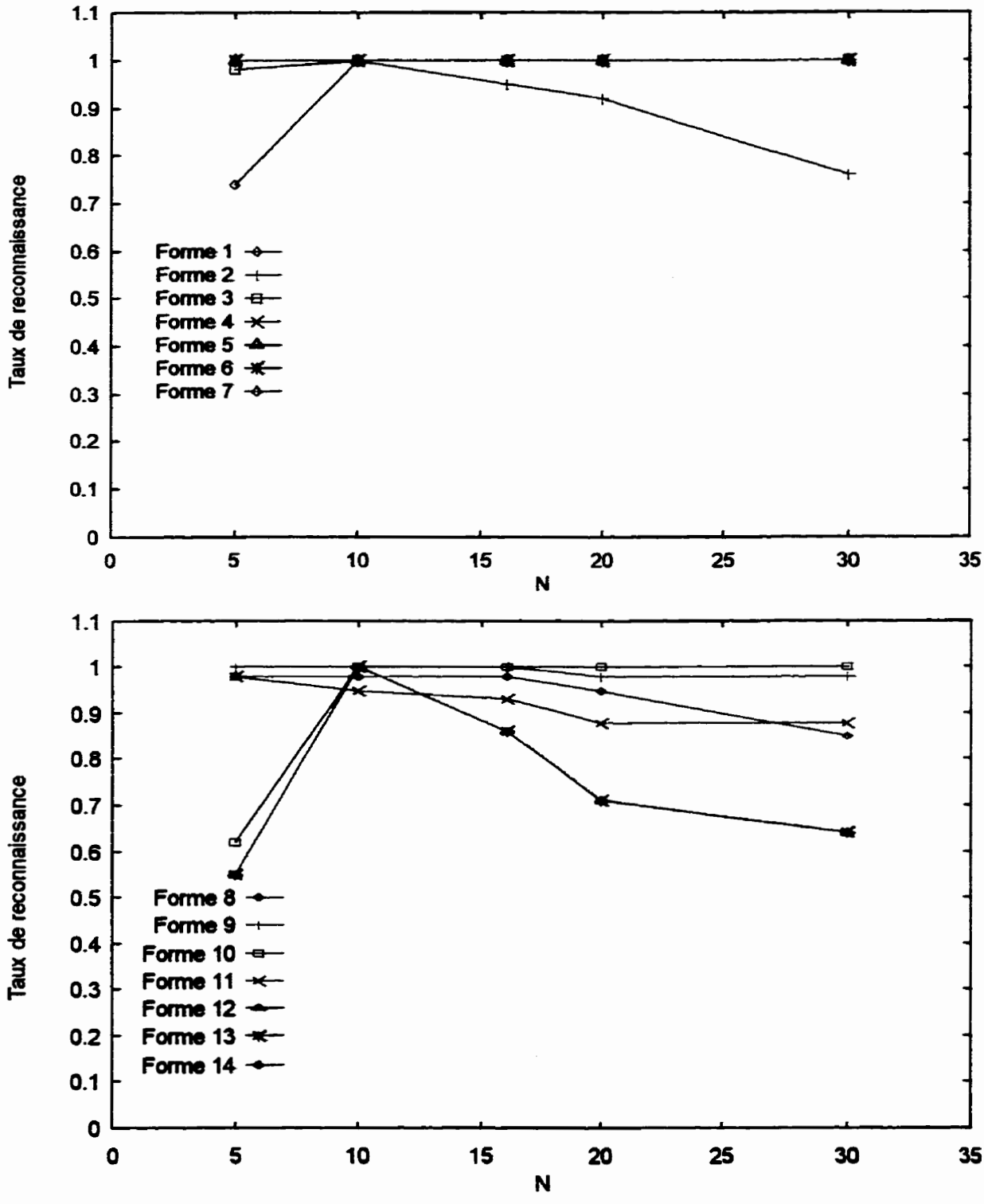


Figure 5.5 Taux de reconnaissance par forme. Prétraitement :A.

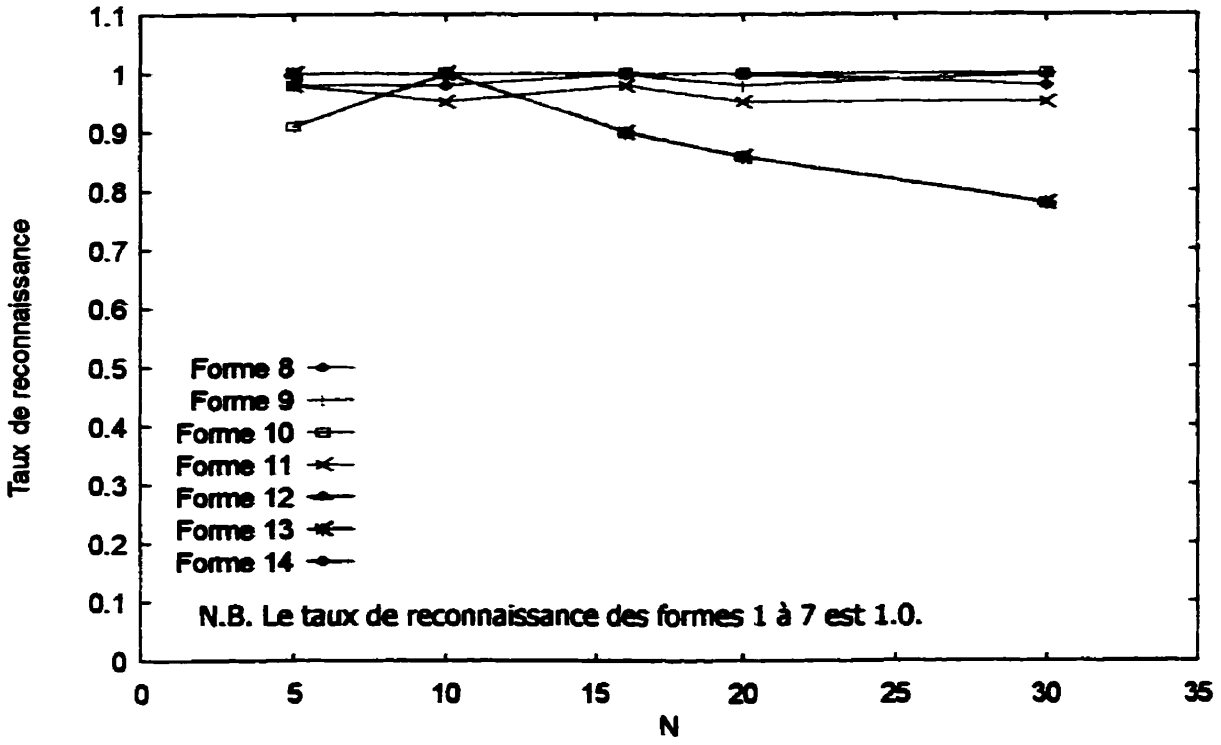


Figure 5.6 Taux de reconnaissance par forme. Prétraitement :B.

L'emploi du prétraitement B assure un meilleur fonctionnement de notre reconnaisseur. En analysant les graphiques, on voit que le prétraitement A présente plus de formes "problématiques" que le prétraitement B. On pense qu'il sera utile de séparer les analyses en deux: l'analyse des résultats obtenus pour N=5 et l'analyse des résultats obtenus pour N >5 parce que des sources d'erreurs sont différentes.

Pour N=5 et le prétraitement A utilisé, les formes mal reconnues sont les formes :

- "Nouvelle ligne" (#1);
- "Joindre"(#14);
- "Palette d'édition" (#13);
- "Insérer" (#10).

Pour N=5 et le prétraitement B utilisé, la seule forme mal reconnue est la forme "Insérer" (#10).

La source d'erreurs pour la forme numéro 10 provient du fait que le prototype correspondant n'a pas été représenté adéquatement : il est clair qu'il faut que N soit un nombre pair (tableau 5.2 troisième colonne). De plus, le nombre minimal nécessaire pour une représentation adéquate des gestes numéros 10 et 13 sont 7 et 8 (tableau 5.2 troisième colonne) nous ne disposons donc pas d'un nombre suffisant de segments pour représenter ces formes. La forme numéro 14 est la forme 12 définie pour les gauchers. Tous les scripteurs qui ont participé aux expériences, étaient des droitiers et ils ont eprouvé plus de difficultés à écrire certaines formes, dont entre autres la forme 14.

Passons maintenant à l'analyse des résultats obtenus pour N >5.
Pour N>5 et le prétraitement A utilisé, les formes mal reconnues sont les formes :

- "Palette d'édition" (#13);
- "Joindre"(#14);
- "Annuler la dernière action" (#11)
- "Effacer" (#8).

Pour N>5 et le prétraitement B utilisé, les formes mal reconnues sont les formes :

- "Palette d'édition" (#13);
- "Joindre"(#14);

On s'aperçoit qu'il y a des erreurs communes pour N >5 considérant les formes mal reconnues peu importe le prétraitement appliqué. Ces formes sont les formes 13 et 14. Il faut quand même dire que le prétraitement B fournit de meilleurs résultats. Nous avons effectué une analyse plus poussée. Nous avons visualisé des formes mal reconnues de la base de données. Nous avons donc constaté que la forme "Palette d'édition" (#13) est écrite de façon bien différente par les usagers. Au lieu de commencer par un trait oblique, ils commencent par un trait vertical. La solution de problème est simple on peut

ajouter un prototype qui représente cette façon d'écrire cette forme-ci. De cette façon on montre au réseau plus de variations de formes et on le laisse extraire les caractéristiques stables. Par exemple si on reprend une partie du tableau 4.5 concernant la forme "Palette d'édition" on pourrait ajouter la forme (13^{+}) qui commence par un trait vertical:

#	Forme	Code (Directions de Freeman)										Nom de la commande
		1	2	3	4	5	6	7	8	9	10	
13		7	7	7	7	1	1	1	1	1	1	"Palette d'édition"
13^{+}		6	6	6	6	1	1	1	1	1	1	"Palette d'édition

L'analyse de résultats de la vérification expérimentale, aide à formuler quelques critères de design supplémentaires envers un nouveau concepteur (ou utilisateur du système) qui veut ajouter des gestes :

- Les formes saccadées sont plus faciles à reconnaître que les formes comportant les changements doux de la direction.
- Mieux choisir les formes différentes.
- Les formes de l'ensemble doivent avoir une "complexité" semblable. Les formes "trop" simples (Ex. Un trait horizontal ou vertical) semblent pouvoir mieux généraliser et avoir une sphère d'influence plus grande que les formes "complexes".

5.3 Conclusion et discussion

Dans ce chapitre, on a présenté les résultats de la vérification expérimentale du reconnaisseur de commandes gestuelles. Les analyses effectuées sur les données de

notre expérience démontrent la robustesse de notre reconnaisseur de commandes gestuelles et sa tolérance aux deux paramètres Ro et N. Un éditeur basé sur les commandes gestuelles du tableau 4.5 serait donc performant puisque l'utilisation d'un quelconque contexte "éclairante" n'a pas été nécessaire (rappelons que grâce à la façon dont cet éditeur été conçu il n'y a pas d'erreurs d'identification du contexte, ni d'erreurs d'exécution des commandes).

Le reconnaisseur fonctionne bien pour l'ensemble des gestes conçus pour permettre son évaluation. On n'a pas testé la capacité du reconnaisseur à apprendre de nouveaux gestes. Notre expérience basée sur une méthode empirique est encourageante : on peut confirmer que le réseau possède le potentiel nécessaire requis pour acquérir de nouvelles connaissances.

La capacité du reconnaisseur à apprendre des nouveaux gestes, peu importe le geste, est très complexe à prédire. La complexité est surtout liée au fait que le réseau ARTMAP est difficile à prévoir. Son comportement ne peut pas être décrit à l'aide d'un modèle standard mathématique ou statistique. Il faudra d'abord trouver des réponses aux questions suivantes avant de s'attaquer à l'évaluation des capacités de post-apprentissage de ARTMAP.

- Est-ce que toutes les formes ou les classes doivent avoir une complexité "semblable"?
- Comment les nouvelles formes sont reliées à des formes déjà apprises dans le cadre de l'ensemble des gestes et dans le cadre des formes d'une catégorie?
- Comment se comporte le réseau en fonction du :
  - ✓ nombre de classes ou catégories à apprendre;
  - ✓ nombre de formes à apprendre;
  - ✓ taux de reconnaissance à atteindre;
  - ✓ nombre des segments ou la longueur du signal d'entrée, etc.?

Pour en connaître d'avantage, il faudra élaborer des nouvelles expériences qui permettront une évaluation systématique des différentes facettes de ARTMAP dans le contexte d'application actuelle de ce projet.

CONCLUSION

La reconnaissance des gestes fait partie d'un domaine de recherche actif depuis plusieurs années : la reconnaissance de formes. De nombreuses méthodes et techniques ont été utilisées pour arriver à mettre une étiquette sur une forme inconnue. Dans la littérature scientifique, on apprend qu'ils existent des systèmes qui offrent de bonnes performances.

L'analyse de plusieurs travaux récemment effectués dans ce domaine, nous a conduit à une avenue de recherche particulièrement intéressante voir attirante par sa popularité. Elle implique l'utilisation de réseaux de neurones artificiels, et parmi ces architectures neuronales, le choix des architectures de type ART nous semblait le plus approprié pour l'accomplissement de ce projet.

Les architectures ART sont reconnues pour être bien adaptées aux problèmes de classification des formes dans des environnements réels. Elles offrent des avantages aux applications qui exigent un apprentissage continuel et autonome. Le réseau ARTMAP constitue la base même du reconnaisseur de gestes que nous avons créé.

ARTMAP est un réseau à apprentissage supervisé. Pour les travaux qui ont été effectué dans le cadre de ce projet, nous avons eu recours à la version binaire de ARTMAP qui est construite à l'aide de deux modules ART1. Le réseau ART1 de son côté, a été originalement conçu pour faire de regroupements automatiques (clustering) d'images binaires. Nous l'avons adapté pour classer des gestes qui sont des signaux spatio-temporels en définissant un prétraitement approprié. On a conçu un nouveau code dit isométrique par lequel on peut exploiter la simplicité du fonctionnement du réseau ART1. On montre que ce code est le seul qui permet d'obtenir un comportement cohérent en fonction du paramètre de généralisation du réseau ART1. La définition d'une séquence de prétraitement et l'introduction de code isométrique ont permis la

classification des commandes gestuelles par un réseau ART1 et par la suite la reconnaissance des commandes gestuelles par un réseau ARTMAP.

L'application du code isométrique nous a apporté les avantages suivants lors d'entraînement de ARTMAP : il ne s'est pas avéré nécessaire qu'un codage complémentaire soit effectué, une seule présentation de l'ensemble d'entraînement a suffi. Ce qui en définitif, a permis de réduire la longueur des signaux d'entrée et a assuré la rapidité de la convergence du réseau ARTMAP.

Le prétraitement et la définition du code isométrique représentent les principaux apports au problème de la classification de séquences spatio-temporelles à l'aide d'une architecture ART binaire. Ils ont permis la création d'un reconnaisseur de gestes rapide, fiable et adaptable, basé sur le réseau ARTMAP.

Pour la poursuite de travaux, il serait particulièrement intéressant d'envisager á concevoir un reconnaisseur de commandes gestuelles multi-traits, qui serait basé sur une approche ART. Ce concept de création pourrait s'adapter aisément aux besoins de la reconnaissance de caractères ou des chiffres manuscrits.

BIBLIOGRAPHIE

AIMÉ-DESIRÉ, K.D. (1995). *Conception et réalisation d'un éditeur de composants électroniques sur ardoise.* Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

APPLE (1993). *Guide de l'utilisateur. Newton MessagePad.*

BELAID, A., BELAID, Y. (1984). *Méthodes structurelles pour la reconnaissance des formes.* Eyrolles, 184p.

BELAID, A., HATON, J.-P. (1984). A Syntactic Approach for Handwritten Mathematical Formula Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, January, 6, 1.

BENGIO, Y. (1996). *Neural Networks for Speech and Sequence Recognition.* International Thomson Computer Press, 167p.

BRAULT, J.J., PLAMONDON, R., LAFRAMBOISE, A. (1995). A Gesture Based Editor For Short Handwritten Messages. *Vision Interface '95*, 162-169.

CARPENTER, G., GROSSBERG, S. (1987a). A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine. *Computer Vision, Graphics, and Image Processing*, 37, 54-115.

CARPENTER, G., GROSSBERG, S. (1987b). ART2: Self Organization of Stable Category Recognition Codes For Analog Input Patterns. In M. Caudill and C. Butler (Eds.), *Proceedings of the IEEE International Conference on Neural Networks*, II, 737-746.

CARPENTER, G., GROSSBERG, S., ROSEN, D.B. (1991). Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System. *Neural Networks*, 4, 759-771.

CARPENTER, G., GROSSBERG, S. (1991). ARTMAP : Supervised Real-Time Learning and Classification of Nonstationary Data by a Self Organizing Neural Network. *Neural Networks*, 4.

CARPENTER, G., GROSSBERG, S., MARKUZON, N., REYNOLDS, J.H., ROSEN, D.B. (1992). Fuzzy ARTMAP : A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps. *IEEE Transactions on Neural Networks*, 3, No.5, September.

CARR, R.M. (1991). The Point of The Pen. GO's Vice President of Software Examines the New PenPoint Operating System. *BYTE*, February.

CAUDILL, M., BUTLER, C. (1992). *Understanding Neural Networks, Volume 2 : Advanced Networks*, IBM, 161-193.

COLEMAN, M.L. (1969). Text Editing on a Graphic Display Device Using Hand-Drawn Proofreader's Symbols. Pertinent Concepts in Computer Graphics, *Proc. Second Univ. Of Illinois Conf. Computer Graphics*, 282-290.

DIMITRIADIS, Y., CORONADO, J. (1995). Towards an Art Based Mathematical Editor that Uses On-line Handwritten Symbol Recognition. *Pattern Recognition*, 28, 6, 807-822.

DUDA, R.O., HART, P.E. (1972). *Pattern Classification and Scene Analysis*. A Wiley-Interscience Publication, 467p.

FAURE, C. (1996). Pen Based Human-Computer Interaction. *Handwriting and Drawing Research: Basic and Applied Issues, M. L. Simner, C.G. Leedham, A. J. W. M. Thomassen (Eds.)*, 373-385.

FAUSETT, L. (1994). *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*, Prentice Hall, 461p.

FINKELSTEIN, A. (1991). Reviewing and Correcting Specifications. *Proc. of the 4th. Annual Conference on Computers and the Writing Process*, 219-237.

FREEMAN, J.A., SKAPURA, D.M. (1991). *Neural Networks Algorithms, Applications and Programming Techniques*. Chapter 8 – Adaptive Resonance Theory, Addison-Wesley.

FUJUTSU PERSONAL SYSTEMS, INC. (1996). *User's Guide for Stylistic 1000*.

GEORGIOPOULOS, M., HUANG, J., HEILEMAN, G.L. (1991). Properties of Learning Related to Pattern Diversity in ART1. *Neural Networks*, 4, No.6, 751-757.

GEORGIOPOULOS, M., HUANG, J., HEILEMAN, G.L. (1992). The N-N-N Conjecture in ART1, *Neural Networks*, 5, 745-753.

GEORGIOPOULOS, M., HUANG, J., HEILEMAN, G.L. (1994). Properties of Learning ARTMAP. *Neural Networks*, 7, 495-506.

GOURRIER, É., SABEVA, S., BRAULT, J.J., PLAMONDON, R. (1997). Visites à l'Institut Nazareth et Louis Braille dans le cadre du projet Braille, Rapport technique interne,20 p., École Polytechnique de Montréal, Canada.

HASSOUN, M. H. (1995). *Fundamentals of Artificial Neural Networks*. MIT Press, 511p.

HAYKIN, S. (1994). *Neural Networks A Comprehensive Foundation*. Prentice Hall, 696p.

HECHT-NIELSEN, R. (1990). *Neurocomputing*. Addison-Wesley Publishing Company, 433p.

KANKAANPAA, A. (1988). FIDS - A Flat Panel Interactive Display System. *IEEE Computer Graphics & Applications*, 71-82, March.

KIM, J. (1988). On Line Gesture Recognition by Feature Analysis. *Vision Interface*, 51-54.

LAFRAMBOISE, A. (1993). *Conception et réalisation d'un éditeur d'écriture cursive à commandes gestuelles*. Mémoire de maîtrise, École Polytechnique de Montréal, Canada.

LIPSCOMB, J. (1991). A Trainable Gesture Recognizer. *Pattern Recognition*, 24, No.9, 895-907.

MICROSOFT (1995). *Pen Computing Quick Reference*. Microsoft Press

MILLEN, D.R. (1993). Pen Based User Interface. *AT&T Technical Journal*, May/June.

MOORE, B. (1988). ART1 and Pattern Clustering. In D.S. Touretzky & G.H. Sejnovski (Eds), *Proceedings of the 1988 Connectionist Summer School*, 174-185, San Mateo, CA, Morgan Kaufmann.

NILSSON, N.J. (1990). *Mathematical Foundations of Learning Machines*, 138p., Morgan Kaufmann Publishers, Inc.

PERUGINO, CH. (1997). Rapport – preuve de concept. Service de consultation SoftZEN Inc., 18p., décembre.

PLAMONDON, R., MAARSE, F.J. (1989). An Evaluation of Motor Models of Handwriting. *IEEE Transactions on Systems, Man, and Cybernetics*, 19, No. 5, September, 1060-1072.

RUBIN, D. (1991). Specifying Gestures by Examples. *Proc. Computer Graphics*, 25, 4, 329-337, July.

SABEVA, S., BRAULT, J.J., PLAMONDON, R. (1997). Commandes gestuelles utilisées pour l'édition de texte à l'aide d'un bloc-notes électroniques : État de l'art, Rapport technique interne, 20 p. École Polytechnique de Montréal, Canada.

SABEVA, S., BRAULT, J.J., PLAMONDON, R. (1998). Codage isométrique de tracés manuscrits pour la classification de séquences à l'aide d'un réseau ART1, *Premier Colloque International Francophone sur l'Écrit et le Document, CIFED'98* ,121-130.

SCHOMAKER, L. (1998). From Handwriting Analysis to Pen-Computer Applications. *Electronics & Communication Engineering Journal*, 93-102, June.

SLATE CORPORATION (1991). *Pen Scheduler basics*.

SUENAGA, Y. NAGURA M. (1980). A Facsimile Based Manuscript Layout and Editing System by Auxiliary Mark Recognition. *Proc 5th Int. Conf. Pattern Recognition*, 856-858, Dec.

WELBOURN, L.K., WHITROW R.J (1990). A Gesture Based Text and Diagram Editor. *Computer Processing of Handwriting*. Eds. Plamondon & C.G. Leedham, World Scientific Publishing Co., 221-234.

WILLIAMSON, J.R. (1996). Gaussian ARTMAP : A Neural Network for Fast Incremental Learning of Noisy Multidimensional Maps. *Neural Networks*, 9, No.5, 881-897.

WOLF, C.G., MORREL-SAMUELS, P. (1987). The Use of Hand-Drawn Gestures for Text Editing. *Proc. Int. J. Man-Machine Studies*, 27, 91-102.

ANNEXE I

La pressente annexe contient des informations qui facilitent la compréhension du quatrième chapitre.

Valeurs atteintes par le critère de vigilance (ρ_a) du module ARTa lors de l'entraînement du réseau ARTMAP avec des données artificielles. La valeur de départ de ρ_a est 0.1 telle que spécifiée dans le tableau 4.6. L'ordre de présentation des formes est celui montré dans le tableau 4.4.

# de la forme	La valeur atteinte par ρ_a
1	0.1
2	0.751
3	0.1
4	0.1
5	0.1
6	0.551
7	0.60
8	0.551
9	0.650
10	0.376
11	0.876
12	0.801
13	0.900
14	0.650

Les signaux d'entrées pour le module ARTa du réseau ARTMAP.

#	Les codes binaires de formes utilisés pour entraîner le réseau ARTPAP
1	000011110000111100001111100001111000011110000111110000111100001111000011110
2	000011110000111111000011110000111100001111000011110000001111000011110000111100
3	000011110000111100001110011110000111100001110011110000111100001111000011110000
4	111100001111000011110000110000111100001111000011000011110000111100001111000011111
5	111100001111000011110000000111100001111000011110000000111100001111000011110000111
6	110000111100001111000011110000110000111100001111000011110000111100001111000011111
7	110000111100001111000011000011110000111100111100001111000011110011110000111110000
8	100001111000011100111100001111001000011110000111001111000011110010000111100001111
9	000111100001111000011110000111100001111010000111100001111000011110000111100001111
10	011110000111100001111000011110000111100011100001111000011110000111100001111000001
11	001111000011110001111000011110000111100011110000111000011110000111000011110000011
12	100001111000011110000111000011110000111100001111000011110001111000011110000111110
13	100001111000011110000111100001110000111100001111000011110000111100001111000011110
14	111000011110000111100001111000011110000111100001111000001111000011110000111100001111000

Les prototypes binaires créés par le réseau lors de l'entraînement. Le réseau ARTMAP a créé 12 prototypes qui sont associés à 8 catégories (classes).

C- catégorie (ARTb)

A – amplitude (nombre de "1")

C	Les prototypes du module ARTa	A
1	000011110000111100001111000000000000000000000000111100001111000011110000111100000	28
2	000011110000111111000011110000111100001111110000111100000011110000111100001111100	40
1	111100001111000011110000000000000000000000000000000011110000111100001111000011111	28
3	111000011110000111100001111000011000011110000111100001111000011110000111100000111	40
2	111000011110000111100001100001111000011110011110000111100001111001111000011110000	40
4	100001111000011100111100001111001000011110000111001111000011110010000111100000111	40
5	000111100001111000011110000111100001111010000111100001111000011110000111100000111	40
5	011110000111100001111000011110000111100011110000111100001111000011110000111100001	40
6	001111000011110001111000011110000111100011110000111000011110000111000011110000011	40
7	100001111000011110000111000011110000111100001111000011110001111000011110000111110	40
8	100001111000011110000111100001111000111100001111000011110000111100001111000011110	40
7	111000011110000111100001111100001111000011110000111100000111100001111000011111000	40

Le tableau montre le lien entre les prototypes du module ARTa et les signaux d'entrées utilisés pour leur création.

Prototype (ARTa)	Vecteurs d'entrées
1	1, 3
2	2
3	4,5
4	6
5	7
6	8
7	9
8	10
9	11
10	12
11	13
12	14

ANNEXE II

La présente annexe contient différents tableaux utilisés pour réaliser les figures du chapitre 5.

Taux de reconnaissance (en %) des commandes gestuelles avec le prétraitement A.

Taux de reconnaissance : Direction Dominante (Type de prétraitement A)					
Ro (critère de vigilance)	Nombre de segments (paramètre N)				
	5	**10**	**16**	**20**	**30**
0.1	0.915	***0.993***	0.980	0.959	0.927
0.2	0.915	***0.993***	0.980	0.959	0.927
0.3	0.915	***0.993***	0.980	0.959	0.927
0.4	0.915	***0.993***	0.980	0.959	0.927
0.5	0.915	***0.993***	0.980	0.959	0.927
0.6	0.915	***0.993***	0.980	0.959	0.927
0.7	0.915	***0.867***	0.759	0.803	0.867
0.8	0.779	***0.701***	0.650	0.634	0.616
0.9	0.534	***0.624***	0.529	0.485	0.405
1.0	0.172	***0.122***	0.039	0.033	0.009

Taux de reconnaissance (en %) des commandes gestuelles avec le prétraitement B.

Taux de reconnaissance Direction du segment (Type de prétraitement B)					
Ro (critère de vigilance)	Nombre de segments (paramètre N)				
	5	**10**	**16**	**20**	**30**
0.1	0.952	***0.995***	0.991	0.985	0.980
0.2	0.952	***0.995***	0.991	0.985	0.980
0.3	0.952	***0.995***	0.991	0.985	0.980
0.4	0.952	***0.995***	0.991	0.985	0.980
0.5	0.952	***0.995***	0.991	0.985	0.980
0.6	0.952	***0.995***	0.991	0.985	0.980
0.7	0.952	***0.815***	0.787	0.980	0.937
0.8	0.731	***0.710***	0.670	0.688	0.687
0.9	0.566	***0.636***	0.548	0.575	0.556
1.0	0.111	***0.095***	0.030	0.027	0.002

Taux de reconnaissance (en %) en fonction du nombre de segments(N) calculé pour chaque forme et utilisant le prétraitement A, Ro=0.5.

# forme	N=5	N=10	N=16	N=20	N=30
1	0.738	1.00	1.00	1.00	1.00
2	1.00	1.00	0.952	0.929	0.762
3	1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00	1.00
7	1.00	1.00	1.00	1.00	1.00
8	0.976	0.976	0.976	0.952	0.857
9	1.00	1.00	1.00	0.976	0.976
10	0.619	1.00	1.00	1.00	1.00
11	0.976	0.952	0.929	0.881	0.881
12	0.548	1.00	0.857	0.714	0.643
13	1.00	1.00	1.00	1.00	0.952
14	0.976	1.00	1.00	0.976	0.952

Taux de reconnaissance (en %) en fonction du nombre de segments(N) calculé pour chaque forme et utilisant le prétraitement B, Ro=0.5.

# forme	N=5	N=10	N=16	N=20	N=30
1	1.00	1.00	1.00	1.00	1.00
2	1.00	1.00	1.00	1.00	1.00
3	1.00	1.00	1.00	1.00	1.00
4	1.00	1.00	1.00	1.00	1.00
5	1.00	1.00	1.00	1.00	1.00
6	1.00	1.00	1.00	1.00	1.00
7	1.00	1.00	1.00	1.00	1.00
8	0.976	0.976	1.00	1.00	0.976
9	1.00	1.00	1.00	0.976	1.00
10	0.905	1.00	1.00	1.00	1.00
11	0.976	0.952	0.976	0.952	0.952
12	1.00	1.00	0.904	0.857	0.786
13	1.00	1.00	1.00	1.00	1.00
14	0.976	1.00	1.00	1.00	1.00