UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À CHICOUTIMI
COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN INGÉNIERIE

par
Stéphane LOISELLE

Exploration de réseaux de neurones à décharges dans un contexte de reconnaissance de parole

Hiver 2004

Bibliothèque
Paul-Émile-Boulet

UQAC
Université du Québec
à Chicoutimi

Mise en garde/Advice

Afin de rendre accessible au plus grand nombre le résultat des travaux de recherche menés par ses étudiants gradués et dans l'esprit des règles qui régissent le dépôt et la diffusion des mémoires et thèses produits dans cette Institution, **l'Université du Québec à Chicoutimi (UQAC)** est fière de rendre accessible une version complète et gratuite de cette œuvre.

Motivated by a desire to make the results of its graduate students' research accessible to all, and in accordance with the rules governing the acceptation and diffusion of dissertations and theses in this Institution, the **Université du Québec à Chicoutimi (UQAC)** is proud to make a complete version of this work available at no cost to the reader.

L'auteur conserve néanmoins la propriété du droit d'auteur qui protège ce mémoire ou cette thèse. Ni le mémoire ou la thèse ni des extraits substantiels de ceux-ci ne peuvent être imprimés ou autrement reproduits sans son autorisation.

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

RÉSUMÉ

De plus en plus de recherches s'effectuent pour développer des systèmes de reconnaissance en se basant sur l'approche connexionniste. En grande partie, on tente de résoudre de cette manière les problèmes que l'on rencontre avec les approches statistiques plus conventionnelles (reconnaissance difficile dans des conditions bruités, quantité de données nécessaire pour l'apprentissage, ...).

L'objectif de cette recherche est de tester la reconnaissance d'un signal vocal à l'aide d'un prototype inspiré par un excellent système de reconnaissance vocale qui a évolué depuis des milliers d'années. On fait évidemment allusion à celui de l'être humain qui est non seulement capable de comprendre rapidement les paroles prononcées par un individu dans des conditions adverses, mais aussi de deviner son sexe, son âge ainsi que son accent.

Des réseaux de neurones ont déjà été développés pour effectuer la reconnaissance vocale, avec plus ou moins de succès. Cependant, nous ne commençons qu'à avoir une idée du potentiel offert par ces outils. Pour cette raison, nous poursuivons l'exploration des réseaux de neurones à décharges, car ils devraient nous permettre d'effectuer la reconnaissance vocale indépendamment du locuteur, sans avoir à exécuter une longue période d'apprentissage qui nécessite une importante quantité de données. De plus, certains types de réseaux à décharges se prêtent bien à la programmation événementielle. Nous avons donc tenté de conserver cette orientation pour obtenir une exécution plus efficace lors de la reconnaissance.

La revue de certains articles jugés intéressants ou qui nous ont inspirés pour ce travail compose la première partie de cette recherche. Aussi, nous donnons une brève description des éléments théoriques amenés par ces articles ou ceux particulièrement intéressants rencontrés à travers le cheminement pédagogique de cette maîtrise. Parmi

les domaines qui se rattachent à notre sujet, ceux qui nous viennent le plus rapidement en tête sont les domaines de l'intelligence artificielle (plus précisément les domaines portant sur les réseaux de neurones), des systèmes en temps réel et des processus aléatoires.

Dans ce mémoire, on a exploré deux façons différentes d'effectuer la reconnaissance vocale à l'aide de neurones à décharges.

Dans une première partie, un réseau composé d'oscillateurs a été utilisé avec un cepstrogramme. Malheureusement, il semble que ce type d'entrée ne soit pas compatible avec l'architecture présentée. De plus, l'implémentation des neurones oscillatoires dans un langage de programmation orienté événements, comme *Rodin*, risque à première vue d'être difficile.

D'un autre côté, le prototype qui a été conçu en s'inspirant du fonctionnement du système auditif possède un potentiel intéressant. En effet, à l'aide du codage par ordre de rang et en modélisant le fonctionnement de l'oreille interne par un banc de filtre et des détecteurs de seuils, nous avons effectué efficacement la reconnaissance vocale avec un vocabulaire limité. Les résultats du prototype se comparent à ceux obtenus avec des chaînes de Markov et des coefficients cepstraux lorsqu'on limite le nombre de prononciations à l'apprentissage. De plus, l'approche développée est simple et peut facilement s'implémenter dans un langage de programmation orienté événements. Cet avantage va éventuellement nous permettre une exécution plus efficace du système de reconnaissance vocale.

D'après les tests présentés dans ce mémoire, les efforts futurs devraient se concentrer sur le prototype qui utilise le codage par ordre de rang. En effet, ce prototype présente déjà un grand potentiel pour la reconnaissance vocale.

REMERCIEMENTS

Je tiens d'abord à remercier Jean Rouat, sans qui l'opportunité de cette maîtrise n'aurait pas existée, Luc Morin et le personnel de l'Université du Québec à Chicoutimi ainsi que celui de l'Université de Sherbrooke qui ont rendu ce projet possible.

De l'autre côté de l'Atlantique, je remercie Simon Thorpe et les responsables de l'université Paul Sabatier de Toulouse qui ont permis la réalisation d'une collaboration Chicoutimi/Sherbrooke et Toulouse. Merci à Daniel Pressnitzer et au sympatique groupe du CERCO pour m'avoir acceuilli chaleureusement lors de mon séjour et pour leur aide.

Pour leur aide et leur soutient, je souhaite remercier Hassan Ezzaidi, Ramin Pichevar, Steeve Larouche et Rachid Moussaoui.

Enfin, je remercie ma famille, mes proches pour leur soutien et leur amitié et particulièrement Robert Loiselle pour ses commentaires et corrections.

TABLE DES MATIÈRES

1 INTRODUCTION 1

1.1 Problématique 1

1.2 Objectif 2

1.3 Méthodologie 2

1.3.1 Oscillateurs 3

1.3.2 Code par ordre de rang 3

1.3.3 *Rodin* 3

1.3.4 Tests de référence 4

I RECHERCHE BIBLIOGRAPHIQUE 5

2 THÉORIE 6

2.1 Anatomie et physiologie 6

2.1.1 Production de la parole 6

2.1.2 Système auditif 8

2.1.3 Système visuel 11

2.2 Reconnaissance vocale 13

2.2.1 Coefficients cepstraux 13

2.2.2 Chaîne de Markov cachée 14

2.3 Réseau de neurones 16

2.3.1 Neurones biologiques 16

2.3.2 Codes par cadence 17

2.3.3 Codes par impulsions 18

2.3.4 Modèles de neurones artificiels 22

3 REVUE **25**

3.1 Réseaux de neurones pour la reconnaissance vocale 25

3.1.1 *SOM* 26

3.1.2 Oscillateur 27

3.1.3 Neurones à impulsions 31

3.1.4 Détecteurs de seuils 34

3.2 Cartes auditives 35

3.3 *Time-sliced paradigm* 38

3.4 Code par ordre de rang 40

II OSCILLATEURS **41**

4 CEPSTROGRAMME ET OSCILLATEURS **42**

4.1 Structure du réseau 42

4.1.1 Implémentation 44

4.2 Représentation du signal 45

4.2.1 Implémentation 46

4.3 Expérimentation 46

4.3.1 Simulation 46

4.3.2 Résultats 47

4.3.3 Discussion 50

III CODE PAR ORDRE DE RANG **52**

5 CODE PAR ORDRE DE RANG **53**

5.1 Banc de filtres cochléaires avec seuils multiples 53
5.1.1 Traitement 54
5.1.2 Création des modèles 55
5.1.3 Reconnaissance 55
5.1.4 Résultats 57
5.2 Compression 58
5.2.1 Traitement 59
5.2.2 Création des modèles et reconnaissance 59
5.2.3 Résultats 62
5.3 Regroupement par seuil 63
5.3.1 Traitement 63
5.3.2 Modèles 65
5.3.3 Reconnaissance 68
5.3.4 Résultats 68
5.4 Regroupement par seuil pour les voyelles 69
5.4.1 Traitement 70
5.4.2 Modèles 71
5.4.3 Reconnaissance 71
5.4.4 Résultats 71
5.5 Modèle binaire 73
5.5.1 Traitement 73
5.5.2 Modèles 73
5.5.3 Reconnaissance 74
5.5.4 Résultats 76
5.6 Influence du nombre de canaux et d'impulsions 76

5.6.1 Modèles 76
5.6.2 Reconnaissance 77
5.6.3 Résultats 80
5.6.4 Modèles binaires 81
5.6.5 Résultats 82
5.6.6 Stratégie des seuils séparés 84
5.6.7 Stratégie des seuils séparés avec modèles binaires 85
5.7 Influence du nombre d'impulsions pour la reconnaissance 87
5.7.1 Résultats 88
5.7.2 Modèles binaires 90
5.8 Conclusion 90

IV *RODIN* 93

6 MODULE *RODIN* 94

6.1 Programmation 95
6.1.1 Concurrence 95
6.1.2 Temps réel 96
6.1.3 Parallélisme 96
6.1.4 Programmation orientée événements 96
6.2 *Rodin* 97
6.3 Expérience 100
6.4 Conclusion 103

V TESTS DE RÉFÉRENCE 105

7 *HMM* ET COEFFICIENTS CEPSTRAUX 106

7.1 Implémentation 106
7.2 Modèles généraux 107
7.2.1 Discussion 108
7.3 Modèles à deux locuteurs 108
7.3.1 Discussion 108
7.4 Modèles avec une seule prononciation 109
7.4.1 Discussion 109
7.5 Reconnaissance des chiffres 111
7.5.1 *HMM* 111
7.5.2 Prototype 114
7.5.3 Discussion 115

VI DISCUSSION 119

8 BILAN 120

9 CONCLUSION 122

VII ANNEXES 124

A PREMIÈRE EXPLORATION 125

A.1 Code par ordre de rang 125
A.1.1 Objectif 125
A.1.2 Expérience 125
A.1.3 Conclusion 127
A.1.4 Modèle 129
A.1.5 Exemple 129

A.1.6 Reconnaissance 129

B PROTOTYPE **133**

B.1 Codage par ordre de rang pour la reconnaissance vocale 133

B.1.1 Installation 133

B.1.2 Fichiers 133

B.1.3 Interface 134

B.1.4 Options 136

B.1.5 Paramètres 136

B.1.6 Fonctions 137

BIBLIOGRAPHIE **139**

LISTE DES FIGURES

2.1 Conduit vocal. 7

2.2 Position des deux plus bas formants des voyelles a, e, i, o et u. [8] . . . 8

2.3 Système auditif périphérique. 9

2.4 Réponse de la membrane basilaire selon la fréquence de l'excitation. [14] 10

2.5 Réponse d'un banc de filtres. 11

2.6 Système visuel. 12

2.7 Chaîne de Markov cachée à cinq états. 15

2.8 Neurone biologique. [12] 16

2.9 Code par cadence utilisant la densité des impulsions. [12] 18

2.10 Vague de potentiels d'action générée par une entrée donnée, appliquée à une population de huit neurones. 20

2.11 Circuit d'un neurone, avec inhibition et entrées pondérées, sensible à une séquence temporelle. 21

2.12 Code par le temps de la première impulsion et code par corrélation et synchronie. 21

2.13 Évolution d'un seuil dynamique. [12] 22

3.1 Segmentation d'une image par *LEGION*. [28] 28

3.2 Système de reconnaissance vocale qui utilise le réseau *STAN*. [13] . . . 31

3.3 (a) Séparation du signal de deux locuteurs avec représentation par carte auditive. (b) Activité des oscillateurs et synchronisation du réseau permettant la séparation des signaux. [19] 37

3.4 Architecture du réseau pour la séparation de locuteurs. 38

3.5 Fonctionnement du *Time spliced paradigm*. [1] 39

4.1 Architecture du réseau avec oscillateurs imaginée pour la reconnaissance vocale. 43

4.2 Couche d'oscillateurs pour la segmentation de la représentation. 44

4.3 Exemple de cepstrogramme de la première prononciation du chiffre un par le locuteur *slo*. 45

4.4 Cepstrogramme de la première prononciation du chiffre un par le locuteur *slo*. 48

4.5 Cepstrogramme de la deuxième prononciation du chiffre un par le locuteur *slo*. 48

4.6 Valeur initiale pour chaque oscillateur de la première couche du réseau de neurones. 49

4.7 Segmentation de la première prononciation du chiffre un par le locuteur *slo*. 49

4.8 Segmentation de la deuxième prononciation du chiffre un par le locuteur *slo*. 50

4.9 Segmentation de la première prononciation du chiffre un par le locuteur *slo* avec le réseau modifié. 51

5.1 Traitement avec banc de filtres cochléaires et seuils multiples. 54

5.2 Traitement avec compression. 59

5.3 Traitement avec regroupement par seuil. 64

5.4 Représentation du regroupement par seuil des impulsions pour la prononciation *locfmtr1a*. 64

5.5 Paramètres du traitement avec *sna*. 79

5.6 Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions. 80

5.7 Paramètres du traitement avec *sna* et des modèles binaires. 81

5.8 Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions pour des modèles binaires. 83

5.9 Paramètres du traitement avec *sna* et séparation des seuils. 84

5.10 Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions pour des modèles avec séparation des seuils. 85

5.11 Paramètres du traitement avec *sna* et séparation des seuils avec modèles binaires. 86

5.12 Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions pour des modèles binaires avec séparation des seuils. 87

5.13 Paramètres du traitement avec *sna*. 88

5.14 Taux de reconnaissance en fonction du nombre d'impulsions pour la reconnaissance (60 impulsions pour le modèle). 89

5.15 Taux de reconnaissance en fonction du nombre d'impulsions pour la reconnaissance (40 impulsions pour le modèle). 89

5.16 Paramètres du traitement avec *sna* pour des modèles binaires. 90

5.17 Taux de reconnaissance en fonction du nombre d'impulsions pour la reconnaissance des modèles binaires. 91

6.1 Types de programmation. 95

6.2 Exemple de programmation orientée événements. 97

6.3 Exemple de programmation séquentielle (à gauche) et parallèle (à droite). 98

6.4 Traitement et module *Rodin*. 100

6.5 Composant *Rodin* pour la détection des seuils. 100

6.6 Traitement du signal *locfjlaa5.wav* par le prototype en *MATLAB* . . . 102

7.1 Modèle de chaîne de Markov pour les voyelles. 107

7.2 Paramètres du traitement avec *sna* pour la reconnaissance des chiffres. 114

A.1 Première prononciation du chiffre 1 par la locutrice *mtr*. 126

A.2 Filtrage du signal par un banc de filtre à 24 canaux. 126

A.3 Valeur absolue des signaux pour les 24 canaux. 127

A.4 Somme cumulée et détection du seuil pour les 24 canaux. 127

B.1 Interface. 135

LISTE DES TABLEAUX

5.1 Compilation des dix séquences d'impulsions produites par les dix prononciations du chiffre un par la locutrice *mtr* 56

5.2 Résultats de la reconnaissance des chiffres de un à quatre avec banc de filtres cochléaires à seuils multiples 58

5.3 Modèles pour la reconnaissance des chiffres de un à quatre. 60

5.4 Résultats de la reconnaissance avec compression des chiffres de un à quatre 62

5.5 Modèle à seuils regroupés pour la prononciation *locfmtr1a* 65

5.6 Modèles pour la reconnaissance avec seuils groupés. 66

5.7 Résultats de la reconnaissance avec le regroupement par seuil pour les chiffres de un à quatre 69

5.8 Résultat de la reconnaissance pour seuils regroupés avec les voyelles et les locuteurs utilisés dans la création des modèles (données d'apprentissage) 71

5.9 Résultat de la reconnaissance pour seuils regroupés avec les voyelles et les locuteurs exclus de la création des modèles 72

5.10 Modèle binaire à seuils regroupés pour la prononciation *locfmtr1a* . . . 74

5.11 Résultat de la reconnaissance pour les locuteurs utilisés dans la création des modèles binaires des voyelles (données d'apprentissage) 75

5.12 Résultat de la reconnaissance pour les locutrices non utilisées dans la création des modèles binaires des voyelles 75

5.13 Liste des modèles sélectionnés 78

6.1 Ports d'entrées des composants *Rodin* 101

6.2 Événement pour les ports d'entrées des composants *Rodin* 102

6.3 Événements générés par les ports de sorties des composants *Rodin* . . . 103

7.1 Résultat de la reconnaissance pour la création des modèles avec une seule prononciation des voyelles 110

7.2 Liste des modèles sélectionnés pour les chiffres 111

7.3 Résultat de la reconnaissance avec les *HMM*s pour les chiffres 112

7.4 Résultat de la reconnaissance avec notre prototype pour les chiffres . . 116

A.1 Représentation des impulsions générées par les dix prononciations du chiffre un par la locutrice *mtr*. 128

A.2 Représentation des impulsions générées par les dix prononciations du chiffre deux par la locutrice *mtr*. 128

A.3 Modèles pour les chiffres un à quatre pour les locuteurs *hgi*, *mtr*, *rlo* et *slo* 130

A.4 Reconnaissance pour les chiffres un à quatre avec des modèles particuliers aux locuteurs. 130

A.5 Modèles généraux pour les chiffres un à quatre 131

A.6 Reconnaissance pour les chiffres un à quatre avec des modèles généraux. 131

CHAPITRE 1

INTRODUCTION

1.1 Problématique

Comme elle l'était déjà pour nos ancêtres, la communication est très importante pour nous. Elle est primordiale autant pour le travail que pour les relations interpersonnelles. De nos jours, avec le développement de la technologie, on se sert de plus en plus de l'informatique. En plus de communiquer avec les gens, nous devons maintenant communiquer avec des machines. Pour ce faire, plusieurs moyens de communication ont été développés. Maintenant, avec le développement des techniques en reconnaissance vocale, la parole peut être utilisée pour parler aux ordinateurs. Il est évident que ce moyen de communication est beaucoup plus naturel que l'utilisation du clavier.

Cependant, les techniques actuelles qui nous permettent la reconnaissance vocale sont encore loin d'être parfaits. Pour avoir une reconnaissance efficace avec de tels outils, il faut « entraîner » le système à reconnaître les caractéristiques vocales de l'utilisateur (locuteur). Il faut également éviter le plus possible les sources de bruits qui pourraient corrompre le signal reçu par le système et ainsi empêcher ou induire en erreur la reconnaissance. Bien entendu, ces systèmes demandent aussi de bonnes ressources matérielles pour pouvoir fonctionner.

1.2 Objectif

L'objectif de cette recherche est de tester la reconnaissance d'un signal vocal à l'aide d'un prototype inspiré par un excellent système de reconnaissance vocale qui a évolué depuis des milliers d'années. On fait évidemment allusion à celui de l'être humain qui est non seulement capable de comprendre rapidement les paroles prononcées par un individu dans des conditions adverses, mais aussi de deviner son sexe, son âge ainsi que son accent.

Depuis déjà quelques années, on crée des systèmes informatiques inspirés du comportement de notre système nerveux pour effectuer certaines tâches facilement réalisables par l'homme, mais difficiles pour la machine. L'un de ces systèmes bio-inspirés et prometteurs est le réseau de neurones à décharges.

Des réseaux de neurones (à décharges et formels) ont déjà été développés pour effectuer la reconnaissance vocale, avec plus ou moins de succès. Cependant, nous ne commençons qu'à avoir une idée du potentiel offert par ces outils. Pour cette raison, nous souhaitons poursuivre l'exploration des réseaux de neurones à décharges, car ils devraient nous permettre d'effectuer la reconnaissance vocale indépendamment du locuteur, sans avoir à exécuter une longue période d'apprentissage qui nécessite une importante quantité de données. De plus, certains types de réseaux à décharges se prêtent bien à la programmation événementielle. Nous allons donc tenter de conserver cette orientation pour obtenir une exécution plus efficace lors de la reconnaissance.

1.3 Méthodologie

La revue de certains articles jugés intéressants ou qui nous ont inspirés pour ce travail compose la première partie de cette recherche. Aussi, nous donnons une brève description des éléments théoriques amenés par ces articles ou ceux particulièrement intéressants rencontrés à travers le cheminement pédagogique de cette maîtrise. Parmi

les domaines qui se rattachent à notre sujet, ceux qui nous viennent le plus rapidement en tête sont les domaines de l'intelligence artificielle (plus précisément les domaines portant sur les réseaux de neurones), des systèmes en temps réel et des processus aléatoires.

1.3.1 Oscillateurs

Pour la partie pratique, nous avons d'abord vérifié si un réseau de neurones développé par un étudiant au doctorat (Ramin Pichevar) pour la séparation de sources pouvait être modifié pour la reconnaissance vocale.

1.3.2 Code par ordre de rang

D'un autre côté, nous avons envisagé une coopération avec une équipe de Toulouse qui combine l'expertise de nos deux groupes (les réseaux de neurones à décharges pour l'équipe de Toulouse et l'analyse de scène auditive pour l'équipe de l'UQAC/Université de Sherbrooke). Ce séjour au sein du laboratoire toulousain nous a permis d'acquérir des connaissances et de tester une approche bien différente basée sur le codage par ordre de rang. À notre connaissance, nous sommes parmi les premiers (Toulouse, UQAC/Université de Sherbrooke) à vouloir appliquer cette approche à la reconnaissance vocale.

1.3.3 *Rodin*

De brefs tests ont aussi été réalisés, pour implémenter en temps réel une partie du fonctionnement de notre prototype, inspiré du codage par ordre de rang, avec le langage de programmation *Rodin*.

1.3.4 Tests de référence

Aussi, nous testerons l'efficacité de notre prototype en comparant les résultats obtenus avec ceux d'un système de reconnaissance plus standard.

Enfin, ce travail se termine avec un bref retour sur les éléments importants et les conclusions qui s'en dégagent.

PREMIÈRE PARTIE

RECHERCHE BIBLIOGRAPHIQUE

CHAPITRE 2

THÉORIE

2.1 Anatomie et physiologie

Il faut situer le sujet du mémoire dans le contexte d'une approche bio-inspirée. Pour cette raison, nous commençons par introduire les éléments nécessaires à la compréhension de la production de la parole et du système auditif humain, grâce à un rapide survol des éléments importants. En effet, une partie de l'approche repose sur ces notions. Ce bref résumé sur les systèmes vocal et auditif a été inspiré du livre de Brian C.J. Moore [14], de l'article de J.M. Ferrandez et associés [8] et de l'article de R. Staloff [22].

D'un autre côté, une approche inspirée du système visuel [27, 25, 26, 5, 17] a influencé une bonne partie de ce projet. Pour mettre en contexte le lecteur, une courte section contient un bref résumé du fonctionnement de la rétine.

2.1.1 Production de la parole

Pour produire le signal vocal, nous utilisons notre système respiratoire. Lors de l'expiration, la cage thoracique s'affaisse, ce qui force l'air de nos poumons à passer par la trachée jusqu'au larynx, qui contient les cordes vocales (figure 2.1). De ce point, l'air peut être expulsé par la cavité buccale (du pharynx aux lèvres) ou par la cavité nasale.

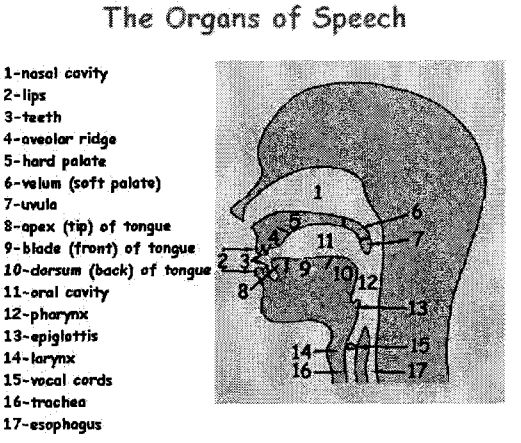


Figure 2.1 – Conduit vocal.

L'air expulsé des poumons peut arriver au conduit vocal sous forme de vibrations (phonèmes voisés) ou sous forme de souffle (phonèmes non-voisés). La fréquence fondamentale correspond à la période de ces vibrations. Il est évident que le flux d'air passant par le conduit vocal est affecté par plusieurs facteurs anatomiques : la position de la langue, du velum, des lèvres... Ces facteurs créent un résonateur dont les pôles de sa fonction de transfert correspondent à la résonance naturelle du conduit vocal. Ces pôles caractéristiques sont appelés les formants. En général, il nous est possible de classer des paroles voisées en utilisant les deux plus bas formants (figure 2.2).

Les voyelles sont des sons voisés à états stables qui produisent des formes d'onde quasi-périodique à formants fixes. La position exacte de l'espace des formants varie avec l'âge, le sexe, le langage et plusieurs autres facteurs physiologiques et psychologiques du locuteur, mais il est possible de donner une position générale.

Il est à noter que le même mot prononcé par des personnes différentes possède des formes d'onde différentes (inter-variabilité). Dans le même ordre d'idées, le même mot prononcé par la même personne à des moments différents va aussi présenter un degré de variation au niveau de son contenu spectral (intra-variabilité).

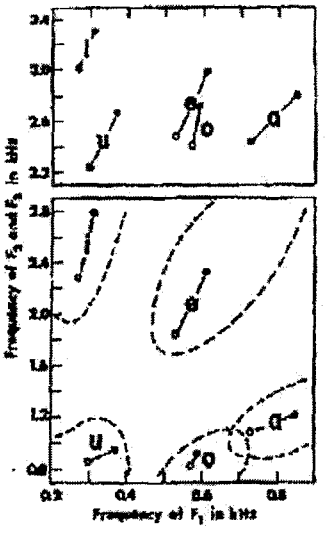


Figure 2.2 – Position des deux plus bas formants des voyelles a, e, i, o et u. [8]

2.1.2 Système auditif

Le système auditif humain (figure 2.3) peut être séparé en trois parties : l'oreille externe, l'oreille moyenne et l'oreille interne. L'oreille externe, composée du pavillon et du conduit auditif, est importante au niveau de notre habilité à localiser les sons. Cette partie du système auditif modifie particulièrement les hautes fréquences.

La fonction majeure de l'oreille moyenne est d'assurer un transfert efficace des vibrations de l'air aux fluides de la cochlée. Cette partie du système auditif commence avec la membrane du tympan qui vibre sous l'effet des ondes sonores acheminées par le conduit auditif. Cette vibration est ensuite transmise par les osselets à la cochlée à travers la fenêtre ovale qui est une ouverture dans la paroi osseuse recouverte d'une membrane.

Du point de vue de l'audition, la cochlée est la partie la plus importante. La cochlée est une structure en forme de spirale qui compose l'oreille interne. Cette structure à paroi osseuse, remplie de fluides presque incompressibles, est séparée sur le sens

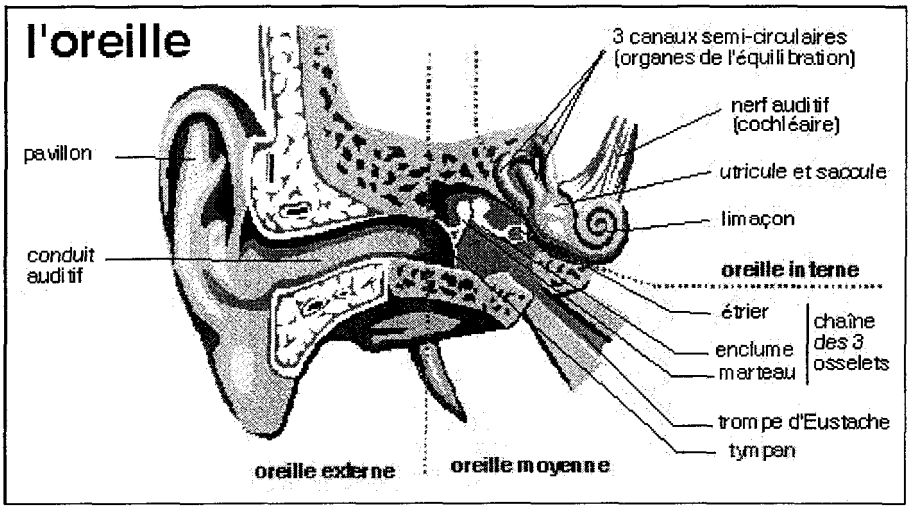


Figure 2.3 – Système auditif périphérique.

de sa largeur par deux membranes (de Reissner et basilaire). Lorsque la fenêtre ovale est activée par un son, une différence de pression est appliquée à travers la membrane basilaire. Cette différence de pression force un mouvement au niveau de la membrane. Les patrons des vibrations que l'on y trouve dépendent de la fréquence de la stimulation (figure 2.4). Les hautes fréquences produisent un maximum de déplacement près de la fenêtre ovale alors que les basses fréquences ont un patron de vibration qui se propage tout le long de la membrane basilaire, mais qui atteint un maximum juste avant la fin de la membrane. En fait, on peut comparer le comportement de la membrane basilaire à un banc de filtres à facteur de qualité pratiquement constant. Enfin, le mouvement dans la membrane basilaire déplace les cellules ciliées internes de la cochlée qui génèrent alors des potentiels d'action. De cette façon, les cellules ciliées traduisent les mouvements mécaniques en activités neuronales.

Il existe une forme de compression au niveau de la réponse de la membrane basilaire. Pour des sons de bas et de moyen niveaux, le mécanisme physiologique amplifie la réponse de la membrane. Cette amplification pour un niveau très bas est presque constante. Cependant, avec l'augmentation du niveau du son, elle est réduite progres-

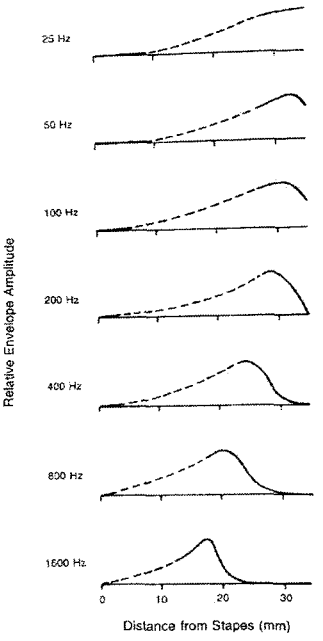


Figure 2.4 – Réponse de la membrane basilaire selon la fréquence de l'excitation. [14]

sivement. Pour des sons de niveau suffisamment haut, le mécanisme devient incapable de contribuer à l'amplification de la réponse de la membrane et la réponse devient linéaire.

Plusieurs chercheurs ont remarqué que le système auditif périphérique se comporte comme s'il contenait un banc de filtres passe-bande (figure 2.5) avec recouvrement. Chaque location de la membrane basilaire répond à une plage limitée de fréquences. On peut alors affirmer que chaque point correspond à un filtre qui possède une fréquence centrale caractéristique. On nomme largeur de bande critique, la largeur de bande à laquelle la sensibilité au signal cesse de croître. On peut approximer la forme des filtres auditifs par des filtres passe bande rectangulaires qui correspondent à la largeur de bande critique. En fait, on doit considérer que la bande passante critique à un sommet bombé et des bords en pente. Plusieurs estimations récentes calculent la largeur de bande rectangulaire équivalente du filtre auditif (ERB).

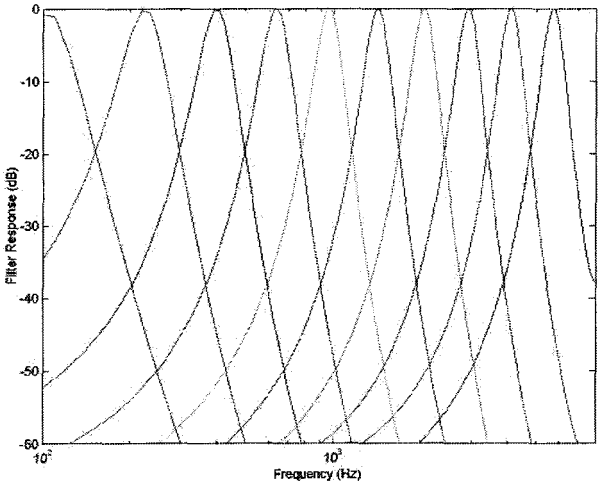


Figure 2.5 – Réponse d'un banc de filtres (ERBfilterBank de 100 à 16 000 Hz avec 10 canaux).

2.1.3 Système visuel

Les yeux sont les composants visibles du système visuel (figure 2.6). Leurs différentes parties permettent, entre autres, de faire la mise au point et de contrôler l'intensité lumineuse. Les traitements qui nous intéressent sont effectués au niveau de la rétine. Celle-ci est une mince surface d'environ 0,5 mm d'épaisseur située au fond de chaque oeil qui couvre presque 75 % de sa surface. La rétine constitue la partie sensible de la vision qui transforme l'image lumineuse focalisée par l'oeil en un signal de potentiels d'action.

La rétine est constituée de neurones et de cellules de soutien qui sont innervées par des vaisseaux sanguins. On y retrouve six couches successives de neurones différents, des neurones sensibles à la lumière aux neurones dont les axones forment le nerf optique. La concentration et la nature de ces différents neurones varient selon la position sur la rétine.

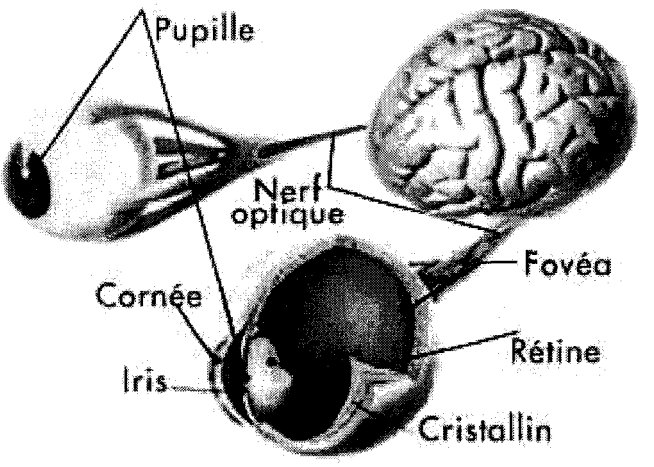


Figure 2.6 – Système visuel.

La première couche de cellules convertit l'énergie lumineuse en un signal neurochimique. Cette activité est diffusée, grâce à leurs synapses, à d'autres cellules, de telle sorte que l'activité des cellules sur la surface rétinienne forme une image des contrastes lumineux. Il nous est d'ailleurs possible de définir une carte rétinotopique qui va correspondre à la transformation spatiale de l'image par l'arrangement spatial de la grille de neurone. Certains types de cellules sont sensibles à des contrastes spatiaux alors que d'autres sont sensibles à des variations temporelles. Elles vont ainsi transformer temporellement l'image des contrastes lumineux qui vont être propagés vers les autres couches. Ces cellules sont sensibles à des contrastes de lumière à plusieurs tailles et certaines cellules répondent de façon maximale lorsque le signal correspond à patron précis. Finalement, la couche la plus interne produit les impulsions neuronales qui constitueront le signal impulsionnel.

Finalement, des expériences ont montré que si on présente assez rapidement une image à un sujet, celle-ci va se projeter au fond de l'oeil en une image distordue et inverse, activer les photo-récepteurs, puis tout le réseau rétinien, pour enfin être transformée en de multiples canaux par des cellules particulières. Chacune de ces cellules

peut alors être caractérisée par une sensibilité maximale à un canal particulier et par une réponse temporelle. L'image que nous percevons est alors entièrement codée en un train d'impulsions entre 20 à 40 ms. On en déduit que les neurones doivent fournir une réponse en intégrant seulement l'information d'une impulsion par synapse [17].

2.2 Reconnaissance vocale

Les systèmes de reconnaissance vocale ont beaucoup évolués depuis leur début et on nous offre maintenant des outils assez efficaces. Les chercheurs tentent maintenant d'améliorer ces systèmes pour obtenir un outil de reconnaissance vocale qui fonctionne indépendamment du locuteur, qui contient un vocabulaire important, qui effectue la reconnaissance en continue (au lieu d'une reconnaissance par mots isolés) et qui offre de bonnes performances même dans un environnement adverse (présence de bruits).

2.2.1 Coefficients cepstraux

Les éléments présentés sur les coefficients cepstraux sont tirés en partie de l'article de Joseph W. Picone [20].

L'utilisation des coefficients cepstraux constitue une technique de traitement de signaux homomorphiques[1] très populaire dans le domaine de la reconnaissance vocale. Ces systèmes sont intéressants pour les approches visant principalement à modéliser le conduit vocal, car ils nous offrent une méthode pour séparer l'excitation du signal provenant des cordes vocales de la forme du conduit vocal. Dans le modèle acoustique linéaire de la production de la parole, le spectre composé du signal, mesuré par exemple par une transformée de Fourier, consiste au filtrage de l'excitation des cordes vocales par un filtre linéaire variable au niveau du temps qui représente le conduit vocal.

[1] Les systèmes homomorphiques sont une classe de systèmes non-linéaires qui obéissent à un principe généralisé de superposition.

Pour séparer $g(n)$, l'excitation du signal, de $v(n)$, la réponse impulsionnelle du conduit vocal, on effectue un traitement qu'on nomme une déconvolution.

$$s(n) = g(n) * v(n) \tag{2.1}$$

En représentant cette équation dans le domaine fréquentiel :

$$S(f) = G(f) \bullet V(f) \tag{2.2}$$

Ensuite, on utilise la propriété du logarithme :

$$\log(S(f)) = \log(G(f) \bullet V(f)) \tag{2.3}$$

$$\log(S(f)) = \log(G(f)) + \log(V(f)) \tag{2.4}$$

Alors, en théorie, l'excitation des cordes vocales ainsi que la représentation du conduit vocal s'additionnent dans le domaine logarithmique. On peut ensuite utiliser des traitements de signaux conventionnels pour les séparer.

À l'intérieur de la représentation qu'on obtient, les termes d'ordre inférieur correspondent à la corrélation à court terme du signal (forme du conduit vocal), alors que les maximums locaux des termes d'ordre supérieur démontrent la corrélation à long terme, ou la périodicité de la forme d'onde (information sur l'excitation). Pour la reconnaissance vocale, on n'utilise généralement que les premiers termes ($n < 20$).

Il est à noter que les coefficients cepstraux risquent d'être sensibles à certains types de bruits et à des distorsions du signal. Ce risque proviendrait du fait qu'on utilise un opérateur non linéaire pour les calculer (logarithme).

2.2.2 Chaîne de Markov cachée

Le type de système de reconnaissance le plus souvent utilisé se base sur des chaînes de Markov cachées [11] (figure 2.7).

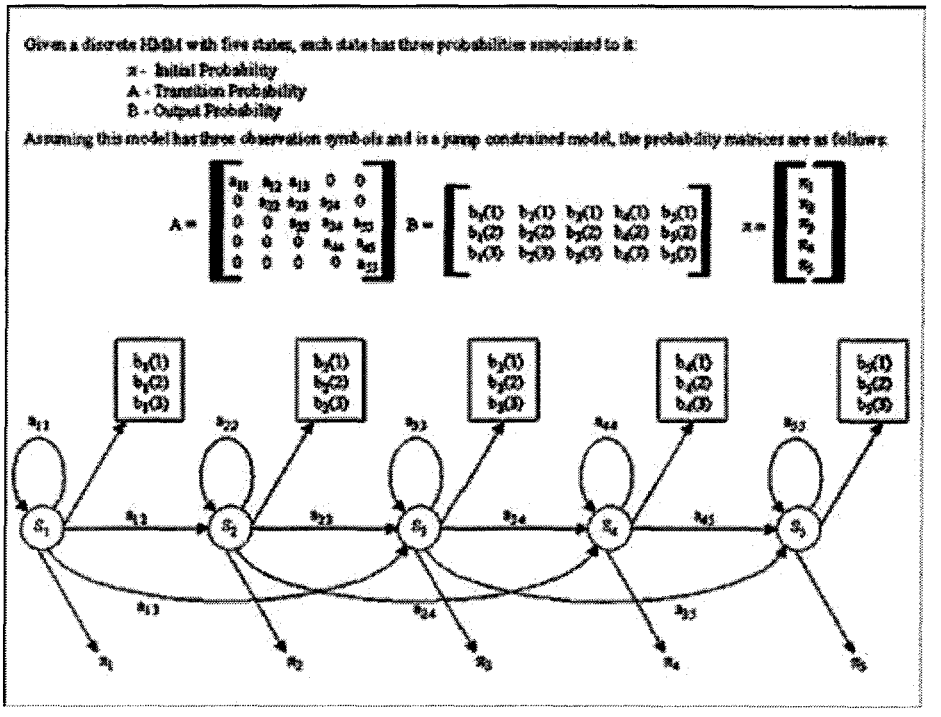


Figure 2.7 – Chaîne de Markov cachée à cinq états.

Dans ce cas, le modèle de mots ou de phonèmes est représenté par un modèle de Markov. Ce type de modèle est un automate probabiliste d'états finis qui se déplace d'état en état une fois par unité de temps. À chaque instant t, un état actif j lui est associé et va générer une observation O_t de densité de probabilité $b_j(O_t)$. Des modèles acoustiques à quatre ou cinq états avec des densités d'observations multi-gaussiennes peuvent représenter les mots et générer des coefficients cepstraux à chaque instant t. De plus, la transition d'un état i vers l'état j peut aussi obéir à une loi de probabilité comme une probabilité discrète a_{ij}. Les probabilités des modèles pour chaque mot ou phonème sont apprises lors de l'apprentissage. Enfin, la probabilité finale est calculée en multipliant toutes les probabilités b_j des vecteurs que nous avons obtenus et les a_{ij} correspondants.

2.3 Réseau de neurones

Ce n'est que récemment que les modèles de neurones à impulsions, inspirés des neurones biologiques, sont utilisés dans le domaine de la reconnaissance vocale. On suppose que la principale raison vient du fait que le signal audio est loin d'un signal impulsionnel et que sa conversion n'est pas évidente.

Les réseaux de neurones constituent un vaste domaine. Comme l'indique le titre de ce mémoire, nous avons restreint notre recherche aux neurones à décharges et à ses dérivés susceptibles d'être utilisés. Les éléments théoriques de cette section sont inspirés du livre de W. Maass [12].

2.3.1 Neurones biologiques

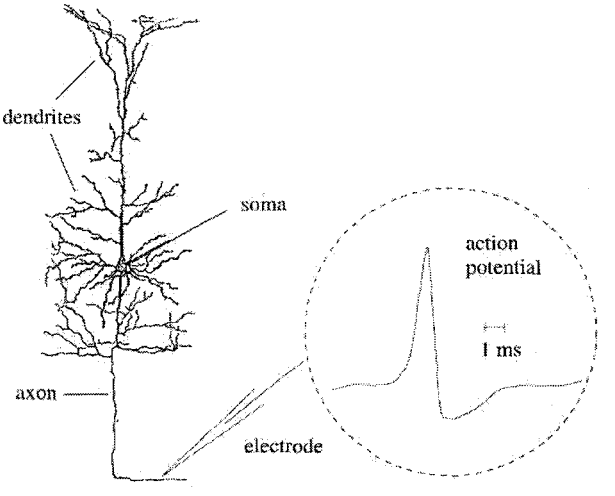


Figure 2.8 – Neurone biologique. [12]

Le neurone biologique (figure 2.8) possède en général trois parties : l'arbre dendritique, le noyau et l'axone. Brièvement, les signaux provenant des autres neurones arrivent à l'arbre dendritique et sont transmis au corps cellulaire et à l'axone. La zone

de transition entre le corps cellulaire et l'axone est d'un grand intérêt. En effet, c'est à cet endroit que l'essentiel du traitement non linéaire est effectué. Si l'excitation des entrées est suffisante, un signal de sortie est émis et propagé à travers l'axone et ses branches aux autres neurones. La jonction qui relie une branche de l'axone et la dendrite d'un neurone récepteur se nomme synapse. D'ailleurs, un neurone transmetteur est communément appelé un neurone présynaptique et un neurone postsynaptique fait référence à un neurone récepteur.

En observant la différence de potentiel de l'axone, le signal observé est une séquence de courtes impulsions appelées potentiel d'action. En général, les impulsions produites à partir d'un même stimulus présentent un grand degré de variabilité. On sait depuis un certain temps que les neurones utilisent ces potentiels d'action pour transmettre des signaux sur de longues distances. Comme les impulsions d'un neurone sont semblables, la forme de l'impulsion renferme peu d'information. Les chercheurs s'accordent pour dire que la nature binaire de ces potentiels d'action signifie qu'ils codent l'information par leur présence ou absence et non pas par leur taille ou par leur forme. De plus, le nombre et la chronologie de ces impulsions constituent de bonnes sources d'information.

La biologie de l'intégration neuronale des impulsions, spécialement au niveau des dendrites, est aussi importante que le codage neuronal. Ce calcul local aux arbres dendritiques est accompli par la conductance de la membrane active.

2.3.2 Codes par cadence

La cadence par le compte des impulsions est essentiellement le calcul du nombre d'impulsions dans un intervalle de temps T divisé par la grandeur de cet intervalle.

La cadence par la densité des impulsions fonctionne pour des stimulus stationnaires ou dépendants du temps. Dans ce cas, l'expérimentateur enregistre les impulsions

générées par les neurones en simulant ceux-ci à l'aide d'une séquence d'entrées. Cette séquence est répétée plusieurs fois et le résultat est placé dans un histogramme. Pour chaque intervalle de temps court avant, pendant et après la simulation, l'expérimentateur compte le nombre de fois qu'une impulsion est produite et les additionne sur toutes les répétitions de l'expérience. Ensuite, le nombre d'occurrences des impulsions est divisé par le nombre de répétitions et par l'intervalle de l'histogramme pour obtenir la densité pour cet intervalle (figure 2.9).

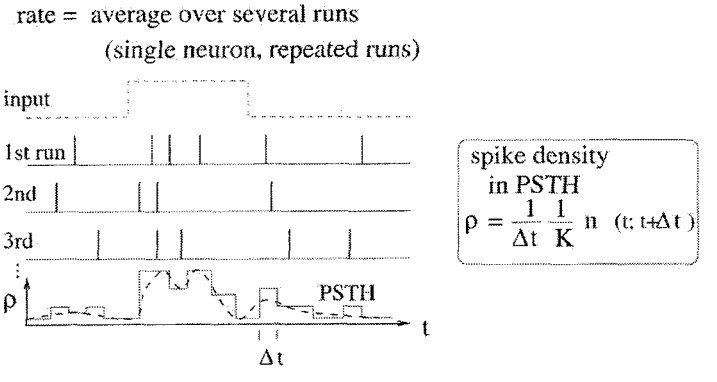


Figure 2.9 – Code par cadence utilisant la densité des impulsions. [12]

La cadence par l'activité d'une population est justifiée, car souvent plusieurs neurones possèdent des propriétés et des réponses aux stimulus similaires. Il est alors possible de définir la cadence des impulsions à l'aide d'un ensemble (une population) de neurones.

2.3.3 Codes par impulsions

Le temps de la première impulsion est une méthode de codage par impulsions. Dans la version la plus pure de cette méthode, uniquement la première impulsion de chaque neurone compte. Toutes les impulsions subséquentes sont jugées inutiles. De la

même façon, on peut assumer que chaque neurone n'émet qu'une impulsion par saccade et qu'elles sont par la suite inhibées. Il est évident que dans un tel cas, l'information est située dans la chronologie et non dans le nombre des impulsions.

Le codage par ordre de rang (*Rank Order Coding*) est une stratégie inspirée du système visuel [27, 17] au niveau duquel de nombreux types de codes auraient de la difficulté à opérer à la vitesse impressionnante que l'on retrouve pour traiter certaines information.

Pour ce type de code, l'information est distribuée à travers une grande population de neurones et elle est représentée par le temps de décharge relatif des impulsions dans une seule vague de potentiels d'action.

En général, le taux de décharge d'un neurone est une fonction monotone de l'intensité de l'entrée (plus le neurone est excité, plus il décharge). Nous pouvons alors affirmer que la latence des décharges d'un neurone, comme son taux de décharge, vont refléter l'intensité de l'entrée. Il est alors possible de dépendre de l'ordre spécifique dans lequel les premières impulsions ont été générées à travers la population. En effet, lorsqu'on présente un patron d'entrées à une population de neurones, la première impulsion de la vague de potentiels d'action générés est produite par le neurone le plus excité (figure 2.10).

La quantité d'information qui peut être transmise par ce type de code augmente en fonction du nombre de neurones qui constituent la population. Pour un nombre relativement grand de neurones, la puissance de transmission d'un tel code peut satisfaire les besoins de n'importe quelle tâche visuelle [27].

Le fait d'utiliser l'ordre relatif et non pas les latences exactes procure des avantages : la stratégie est plus simple à implémenter, le code procure au système une certaine invariance au niveau des changements d'intensité des stimulus ou des contrastes

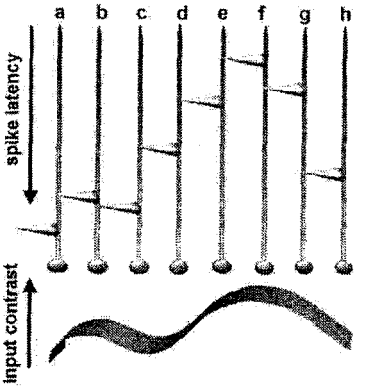


Figure 2.10 – Vague de potentiels d'action générée par une entrée donnée, appliquée à une population de huit neurones, qui produit la séquence *f*, *g*, *e*, *d*, *h*, *b*, *c* et *a*. [27]

et l'information est disponible dès que le premier neurone décharge. Cependant, le système est incapable de faire des jugements précis à propos de la valeur de l'intensité des entrées. Par contre, on sait que le système visuel humain fonctionne mieux en comparant directement les caractéristiques des stimulus.

Pour ce code, les neurones doivent donc être sensibles à la structure temporelle des impulsions qu'ils reçoivent en entrées. Ils doivent répondrent de façon sélective à une séquence particulière d'activation de leurs entrées. Les patrons des impulsions d'entrées sont alors représentés dans le domaine temporel et un neurone donne une importance maximale à la première impulsion qu'il reçoit. L'influence des impulsions suivantes sur le neurone diminue progressivement.

Le niveau d'activité résultant pour le neurone est le produit de ses poids synaptiques distribués à ses entrées et de la fonction de désensibilisation croissante. À l'aide d'un seuil approprié, le neurone (figure 2.11) peut alors être sensible à une séquence particulière de ses entrées.

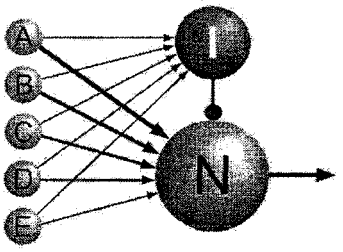


Figure 2.11 – Circuit d'un neurone, avec inhibition (I) et entrées pondérées (A, B, C, D et E), sensible à la séquence temporelle : A, B, C, D et E. [27]

La corrélation et la synchronie peuvent aussi jouer un rôle dans le codage. Il est aussi possible d'utiliser d'autres neurones comme signal de référence pour un code impulsionnel. Par exemple, la synchronisation entre deux ou plusieurs neurones (figure 2.12) aurait une signification particulière. D'un autre côté, on pourrait utiliser non seulement la synchronisation, mais aussi des patrons spatio-temporels précis des impulsions. Il est à noter que la corrélation des neurones auditifs est dépendante du stimulus et pourrait fournir de l'information au-delà de la cadence des décharges.

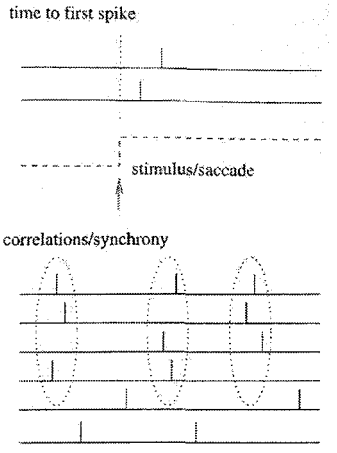


Figure 2.12 – Code par le temps de la première impulsion (en haut), code par corrélation et synchronie (en bas). [12]

2.3.4 Modèles de neurones artificiels

Le neurone avec seuil à décharges est un modèle simple. Avec un niveau d'abstraction élevé, on peut considérer le neurone comme une unité homogène qui génère des impulsions lorsque son excitation totale est suffisamment importante.

Le modèle simple de neurone à décharge est décrit par une variable (le potentiel interne) et il peut recevoir des signaux de ses neurones présynaptiques. Si cette variable augmente par l'effet de ses entrées et atteint un seuil défini, le neurone décharge. Immédiatement après cette décharge, la variable qui représente le potentiel interne est souvent réinitialisée en additionnant une valeur négative. De cette façon, on obtient une période réfractaire à la suite de chaque décharge.

Un seuil dynamique (figure 2.13) peut aussi être utilisé. Dans ce cas, le seuil augmente après chaque décharge et reprend lentement sa valeur originale pendant les périodes où il n'y a pas de décharges.

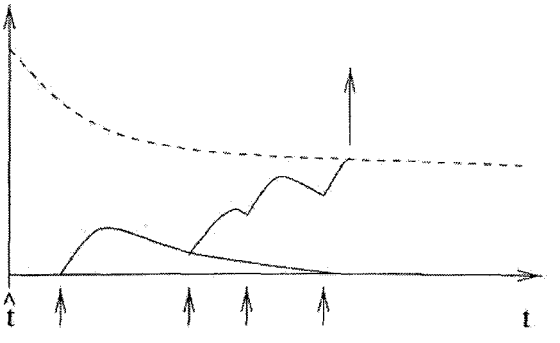


Figure 2.13 – Évolution d'un seuil dynamique. [12]

Le modèle d'impulsion. L'état interne du modèle d'un neurone est décrit par une seule variable, le potentiel de la membrane. Lorsque ce potentiel augmente et dépasse la valeur d'un seuil, une impulsion est générée. Le résultat des activités au niveau des

ions crée un pic escarpé du potentiel suivi d'un long potentiel négatif. Il faut remarquer que lorsque le potentiel d'action augmente ou diminue rapidement, aucune émission d'impulsion n'est possible.

Le modèle à intégration et décharge constitue un exemple important de la classe de modèle avec seuil à décharge. Le comportement du neurone est défini par un intégrateur à fuite avec seuil au niveau du potentiel de la membrane. Lorsque la valeur du potentiel de la membrane dépasse la valeur du seuil, le neurone décharge. Suivant la production d'une impulsion, le potentiel est réinitialisé à une nouvelle valeur.

Une période réfractaire est souvent utilisée pour empêcher la production d'une seconde impulsion trop près de la précédente. Dans ce cas, on donne une valeur au potentiel de la membrane qui sera conservée pendant un certain temps. Ensuite, on redémarre l'intégration avec une valeur initiale.

Des oscillateurs hautement non linéaires peuvent définir des neurones à décharges. Ils présentent alors un comportement collectif qui peut montrer d'importantes caractéristiques au niveau des calculs.

La synchronisation est un comportement que l'on peut obtenir avec un modèle de neurone à intégration et décharges en utilisant des liens synaptiques excitateurs sans délais, ou des liens inhibiteurs avec délais. Ce phénomène peut être utilisé pour définir des fragments de données. Ces fragments sont alors composés de neurones qui déchargent en même temps et les différents fragments possèdent un temps de décharge qui leur est propre. Il est aussi possible de décrire ce comportement comme une segmentation temporelle avec laquelle on sépare les données grâce au retard de phase entre l'excitation des différents groupes. Cette séparation est caractérisée par un petit nombre comme limite de segments, une limitation inhérente au comportement d'oscillateur non linéaire.

La majorité des systèmes biologiques qui présentent une synchronisation peuvent être décrits comme des oscillateurs liés, où l'état parfaitement synchronisé n'est qu'un attracteur dynamique parmi d'autres. De plus, il est possible de diviser les modèles d'oscillateurs liés en deux types, oscillateurs liés par la phase et oscillateurs liés par les impulsions.

CHAPITRE 3

REVUE

À travers ce chapitre, on survole brièvement certains types de réseaux de neurones qui effectuent la reconnaissance vocale. Aussi, on présente différents types d'entrées et des stratégies (ou méthodes) qui peuvent s'appliquer aux réseaux.

3.1 Réseaux de neurones pour la reconnaissance vocale

L'approche la plus utilisée pour effectuer la reconnaissance vocale est basée sur les modèles de Markov cachés[1] [1]. D'un autre côté, la technique connexionniste est une autre approche fortement supportée (basée sur les réseaux de neurones artificiels). Comme elle semble être l'approche la plus naturelle pour mimer le parallélisme massif inhérent au traitement effectué par le cerveau humain, il y a de fortes motivations au niveau des recherches en réseaux de neurones artificiels pour la reconnaissance vocale automatique. Un avantage important des réseaux de neurones sur les autres méthodes, est qu'ils peuvent généraliser à partir de différentes données d'entrées et extraire les caractéristiques distinctives du signal complexe qu'est la parole.

Cependant, l'approche connexionniste est encore considérée comme une jeune école de pensée dans le domaine de la reconnaissance vocale.

[1] *Hidden Markov models, HMM*

3.1.1 *SOM*

Les réseaux de Kohonen, ou *SOM*[2], algorithmes de réseaux de neurones artificiels les plus populaires dans la catégorie d'apprentissage non supervisé, sont des réseaux structurés non hiérarchiques d'unités simples interconnectées latéralement. Habituellement, on utilise une structure en deux dimensions dans les applications et celle-ci détermine le voisinage pour chaque unité. À l'intérieur de cette structure, tous les neurones agissent comme unités d'entrée et reçoivent tous des entrées de dimension identique en parallèle. De plus, chaque entrée externe d'un neurone est pondérée par une valeur réelle (un poids). Par les connections latérales, les neurones reçoivent aussi un signal de rétroaction interne. Pour effectuer un apprentissage avec ce type de réseau, on cherche à donner au réseau le bon ensemble de poids qui est trouvé en fonction du type d'entrée donné et des résultats espérés.

Des expériences ont montré qu'il était possible de créer des cartes pour reconnaître des phonèmes (cartes phonotopiques). N. Aroud et N. Elloze [2] ont poursuivi cette démarche en créant un système coopératif basé sur l'association des différents algorithmes d'apprentissage supervisés ainsi que non supervisés hérités des *SOM*. Le système étudié résultant de cette étude présente un taux de reconnaissance, pour un groupe de phonèmes, supérieur à d'autres systèmes utilisant des *SOM*.

Ferrandez J.M. et collaborateurs [8] ont aussi créé un système de reconnaissance vocale en se basant sur un *SOM*. Cette fois, le système consiste en un module d'extraction des paramètres qui alimente un module de reconnaissance. Ces modules sont inspirés du système biologique.

Le module d'extraction paramétrique comprend un modèle de la cochlée basé sur un banc de filtre gammatone, qui fournit l'analyse en fréquence et la réponse temporelle, ainsi qu'un mécanisme pour le traitement de la transduction neuronale qui

[2] *Self-organizing maps*

inclut l'adaptation, la compression et la rectification. On y retrouve une étape pour l'intégration temporelle qui met l'emphase sur les composants statiques et un module d'extraction des composants qui utilisent une stratégie spatio-temporelle pour obtenir une robustesse et une indépendance du niveau de l'approche temporelle et l'estimation de l'énergie des méthodes spatiales.

Le module de reconnaissance consiste en un *SOM* à délais temporels qui groupe et classifie de façon automatique les différentes combinaisons des composants.

Des voyelles ont été utilisées pour tester la reconnaissance pour leur spectre statique. Elles possèdent toutes la même fréquence fondamentale et on ne conserve que les deux premiers formants pour effectuer les tests. Les chercheurs sont satisfaits du taux de reconnaissance obtenu avec leur système pour les voyelles. Par contre, le taux de reconnaissance diminue de façon importante avec les autres types de phonèmes.

Même s'ils sont populaires, les *SOM* ne sont pas les seuls types de réseaux de neurones utilisés dans le domaine de la reconnaissance vocale. En effet, plusieurs travaux se basent sur des neurones oscillatoires et d'autres types de neurones à impulsions.

3.1.2 Oscillateur

LEGION[3] est un outil développé par D. Wang et D. Terman [28] utilisé dans la conception d'un autre réseau de neurones dont l'objectif est la séparation de locuteurs [18]. La base de cet outil est dérivée d'une hypothèse où l'on assume que les neurones de haut niveau dans le cerveau deviennent de plus en plus sélectifs pour éventuellement en arriver à ce qu'un neurone représente un objet. Aussi, on fait mention de liens entre les différentes caractéristiques de l'objet par corrélation temporelle. À travers cette corrélation, l'objet est représenté par la corrélation temporelle de l'activité d'impulsions des cellules qui encodent les différentes caractéristiques de l'objet.

[3] *Locally excitatory globally inhibitory oscillator networks*

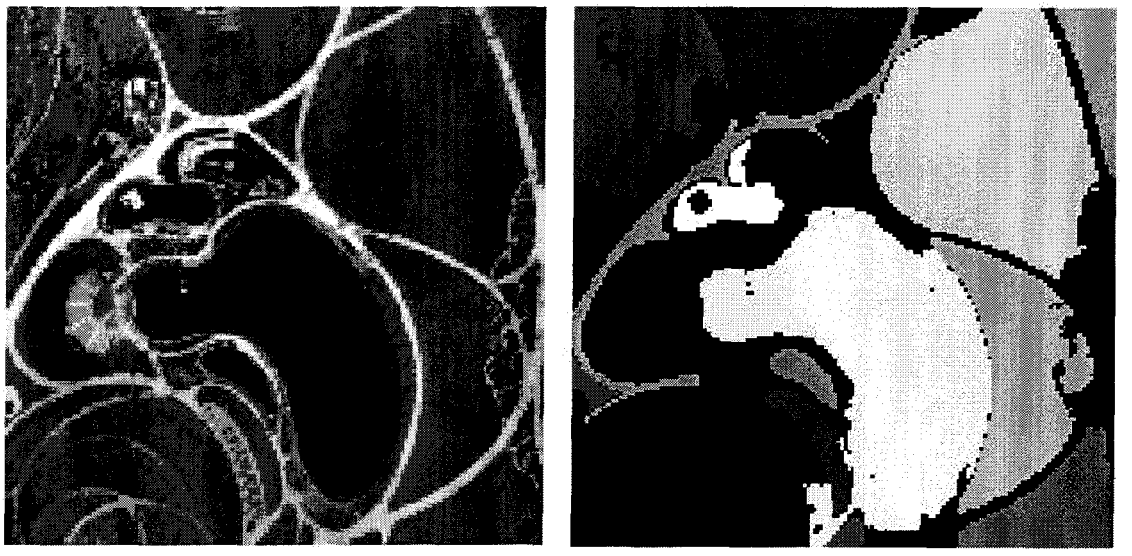


Figure 3.1 – Segmentation d'une image par *LEGION*. [28]

Pour effectuer la segmentation d'une image, le système *LEGION* utilise une forme spéciale de corrélation temporelle, la corrélation oscillatoire. En effet, on utilise dans ce système des neurones oscillatoires comme unités de base. Chaque oscillateur de *LEGION* est modélisé comme un oscillateur à relaxation standard. L'excitation locale est implémentée par des liens positifs entre les oscillateurs voisins et l'inhibition globale est réalisée à l'aide d'un inhibiteur global. *LEGION* présente le mécanisme de *gating* sélectif dans lequel des oscillateurs stimulés par les mêmes patrons tendent à se synchroniser par l'excitation locale et des groupes d'oscillateurs stimulés par différents patrons tendent à se désynchroniser par l'inhibition globale. Au cours du traitement, un oscillateur excitable peut devenir oscillatoire s'il reçoit des autres oscillateurs des liens suffisamment importants. Alors, un oscillateur qui possède un grand potentiel peut diriger l'activation d'un bloc d'oscillateurs correspondant à un même objet. Cependant, les liens ne sont pas suffisants, ils doivent aussi posséder un certain degré de synchronisation.

Enfin, les auteurs présentent des résultats intéressants au niveau de la segmentation d'images obtenue avec leur système (figure 3.1). Aussi, les caractéristiques des oscillateurs de *LEGION* ont servi au travail sur les scènes auditives suivant.

R. Pichevar et J. Rouat utilisent une représentation AM^4 des signaux d'un banc de filtres cochléaires en combinaison avec un masque qui est dérivé d'un réseau de neurones pour effectuer la ségrégation au niveau des locuteurs dans un signal [18].

Pour obtenir la représentation du son, on filtre le signal à l'aide d'un banc de filtres cochléaires qui imite une partie du comportement de la cochlée. Par la suite, on utilise un algorithme pour extraire les caractéristiques et le corrélogramme normalisé est calculé. On segmente ensuite la représentation en se basant sur la cohérence du taux de décharge des neurones, les neurones qui possèdent une même phase faisant partie d'un même groupe. Pour ce faire, on utilise un réseau de neurones localement connectés et une approche par corrélation oscillatoire.

L'image présentée au réseau a deux dimensions : la fréquence (représentée par les canaux du banc de filtres) et le temps. Cette image est appliquée à une carte de neurones chaotiques où chaque neurone est connecté avec ses voisins par des délais. Au début de la simulation, ces neurones fonctionnent de façon indépendante, mais à travers la simulation, des connections sont créées entre des groupes de neurones au comportement similaire et des régions sont ainsi formées.

D'un autre côté, la couche de sortie du réseau possède une seule dimension et chaque neurone traite la sortie d'un canal du banc de filtres. Ce vecteur de neurones inspirés du modèle de *LEGION* est utilisé avec un contrôleur global, ce contrôleur sert à briser la symétrie entre les différentes régions. Le réseau crée un masque de façon non supervisée qui prend en considération l'information mutuelle des canaux du banc de filtres. Enfin, ce masque permet ensuite d'augmenter l'importance des canaux les moins corrompus par du bruit.

[4] *Amplitude modulation*

R. Pichevar et J. Rouat ont observé qu'il était difficile d'obtenir une synchronisation au niveau du réseau à oscillation. D'un autre côté, les neurones chaotiques sont moins compliqués en terme de calcul et peuvent présenter un comportement périodique.

L'article de Y. Yamaguchi et associés [10] propose aussi d'utiliser des neurones oscillatoires. Cette fois, les auteurs présentent un réseau capable d'effectuer la reconnaissance vocale indépendamment du locuteur. L'architecture de ce réseau est constituée de quatre couches. La première couche correspond au système sensoriel périphérique qui définit l'enveloppe du spectre. Les trois autres couches correspondent au cortex auditif. Pour générer une cohérence dynamique, des neurones oscillatoires sont utilisés pour les deuxième et troisième couches. Ces couches agissent respectivement comme extracteurs de paramètres et comme assembleur de caractéristiques. D'abord les caractéristiques des formants sont codées sur un espace tonotopique et on identifie ensuite trois formants comme caractéristiques majeures d'une voyelle en accord avec les statistiques pour les voyelles japonaises.

L'entrée de la couche périphérique consiste en enveloppes de spectre dérivées par analyses de prédiction linéaire. Ces entrées sont ensuite traitées par un banc de filtres à 32 canaux qui est simulé en transformant l'enveloppe du spectre en l'échelle *mel* pour obtenir des caractéristiques similaires à celles de la perception auditive humaine. En l'absence de stimulus au niveau de la couche périphérique, les oscillateurs des couches supérieures sont dans un état de repos uniforme. Lorsqu'un signal est reçu à la couche périphérique, des groupes synchronisés locaux, qui correspondent à des caractéristiques locales, émergent suite à la coopération entre les oscillations de la deuxième couche de façon autonome. Par l'interaction entre les deuxième et troisième couches ainsi que les interactions de longue portée dans la troisième couche, les groupes sont reliés de façon autonome pour générer un groupe global synchronisé. Enfin, la dernière couche évalue l'activité synchronisée des trois domaines des formants de la troisième couche et sélectionne la catégorie de la voyelle.

Les résultats obtenus par Y. Yamaguchi et associés suggèrent que la reconnaissance de voyelles, indépendamment du locuteur et dans un environnement bruité, peut être réalisée à l'aide d'une génération de cohérences dynamiques à travers les oscillateurs du réseau de neurones.

3.1.3 Neurones à impulsions

Dans l'article de D. Mercier et R. Séguier [13], on utilise des coefficients cepstraux, le logarithme de l'énergie et leur valeur delta. Comme la quantification vectorielle fonctionne sur ces paramètres, cette méthode est utilisée.

Pour générer facilement des impulsions à partir d'un signal multidimensionnel qui évolue continuellement au cours du temps, on propose de faire une quantification vectorielle statique à partir du signal à différents moments. Cette méthode permet d'associer la forme du prototype à la forme statique.

La procédure utilisée comporte quelques étapes : l'apprentissage dans laquelle on trouve la définition des prototypes statiques et l'exploitation où l'on fait l'identification du prototype qui ressemble le plus au signal à chaque intervalle et où l'on émet une impulsion à la sortie correspondante au prototype.

Le principe du réseau *STAN*[5] (figure 3.2) utilisé est de coder des événements discrets avec deux degrés de liberté (amplitude et temps) avec des nombres complexes qui sont aussi bivalents (amplitude et phase).

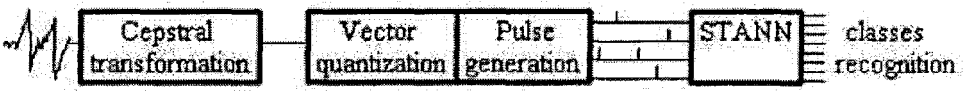


Figure 3.2 – Système de reconnaissance vocale qui utilise le réseau *STAN*. [13]

[5]Spatio-temporal artificial neuron

Les tests sont effectués avec la base de données *Tulips* pour trois problèmes classiques en reconnaissance de parole. D'abord, ils ont testé la reconnaissance des chiffres avec un seul locuteur, c'est-à-dire que l'apprentissage ainsi que la reconnaissance sont effectués avec le même locuteur. Ensuite, les tests se sont étendus sur plusieurs locuteurs pour la reconnaissance et l'apprentissage. Pour le dernier test, on utilise des locuteurs différents pour la reconnaissance et l'apprentissage.

En conclusion à leurs résultats, les auteurs perçoivent un potentiel intéressant pour l'utilisation de réseaux de neurones à impulsions pour la reconnaissance vocale.

Toujours pour la reconnaissance vocale, l'objectif de l'article de M. Bodruzzan et associés [4] est d'extraire les informations véhiculées par le signal, spécialement l'information qui n'est pas directement reconnaissable, par exemple la parole. Comme ce signal varie dans le temps, une méthode jointe temps-fréquence, telle une représentation spectrogramme, est appliquée en entrée pour un réseau de neurones impulsionnels liés. Le réseau présente en sortie une série temporelle de caractéristiques qui correspondent au nombre de neurones qui ont déchargés à chaque fois que la représentation en entrée est appliquée.

Le spectrogramme de la parole est une représentation temps-intensité du spectre à court terme. Le signal est découpé en segments fenêtrés (généralement par une fenêtre de Hanning) qui se recouvrent et dont on calcule la transformée de Fourier à court terme. Cette technique nous permet alors d'obtenir du signal une représentation en deux dimensions.

Le réseau qui est utilisé se base sur le modèle de neurone de Eckhom. Les neurones possèdent donc une entrée pour le stimulus et un petit champ récepteur lié aux cellules voisines. Les impulsions sont générées par une fonction échelon et un seuil interne à fuite. La cadence de décharge de chaque neurone dépend de son entrée. Par contre, grâce au champ récepteur lié, des cellules proches qui ont une excitation similaire se synchronisent.

Des tests avec différents mots et différents locuteurs ont été réalisés ; comme conclusion, les auteurs affirment que le système peut assez bien classifier les mots des différents locuteurs.

Certains travaux, comme ceux de J. Stork et associés [15, 24], accordent un intérêt particulier sur les liens entres les neurones qui composent le réseau de neurones. J. Stork et associés n'ont aucun doute sur le fait que le cerveau utilise la chronologie exacte des potentiels d'action pour coder l'information. Cette idée est aussi supportée par des découvertes au niveau de la dynamique à court terme des synapses des neurones biologiques.

Dans cet article, le modèle de cochlée passive de Lyon est utilisé pour sa simplicité et parce qu'il produit directement la probabilité des impulsions. On tente de décrire pour ce modèle la transformation du son par le système auditif humain en probabilités de décharges similaires à la représentation dans le nerf auditif.

L'oreille externe ainsi que l'oreille moyenne sont représentées par un filtre de pré-emphase. La cochlée est divisée en un large nombre de canaux, chacun décrit par deux filtres linéaires qui modélisent la vague de pression passante dans la cochlée et le déplacement de la membrane basilaire. Les cellules ciliées internes, modélisées par des *half-wave rectifiers*, convertissent le déplacement mécanique en cadence de décharge qui détecte l'énergie du signal. Quatre étapes de gains automatiques contrôlent ensuite la quantité de compression et de masquage des sorties. Le résultat de ce traitement est un cochléogramme qui présente les différents canaux à travers le temps.

Un réseau de neurones totalement connecté avec *feed-forward* et synapses dynamiques est utilisé. Le modèle de neurone sélectionné est réalisé à partir d'intégrateur à décharges avec fuite et période réfractaire absolue.

Les premiers tests de reconnaissance ont été effectués avec des sinusoïdes comme stimuli. J. Stork et associés ont ensuite testé le réseau avec des phonèmes manuellement

séparés. Comme il y a un grand degré de variabilité au niveau de la durée d'un même phonème, il a été décidé de comparer des phonèmes de même durés. Cependant, les phonèmes sont trop stationnaires et ils ont été changés pour des segments plus longs.

Enfin, le modèle conçu par les auteurs peut apprendre les transitions dans le signal de parole, mais il ne fonctionne pas avec un signal stationnaire.

3.1.4 Détecteurs de seuils

Comme on a pu le constater, les modèles de neurones peuvent prendre plusieurs formes. Dans les travaux ci-dessous, le comportement des neurones du système auditif est simulé grâce à des détecteurs de seuils.

S. Sandhu et O. Ghitza [21] proposent d'utiliser une autre sorte de coefficients avec les chaînes de Markov cachées. Ces coefficients sont calculés à partir d'histogrammes normalisés. Ces histogrammes sont obtenus en trois étapes. D'abord, on filtre le signal par un banc de filtres pour simuler la membrane basilaire. Ensuite, on détecte, pour chaque sortie de filtre, les dépassements de seuils. Cette étape simule les cellules ciliées internes. Enfin, on accumule les intervalles des dépassements de seuils dans un histogramme.

Les chercheurs ont expérimenté cette méthode d'extraction de caractéristiques spectrales avec un système de reconnaissance vocale en continu sur la base de données *TIMIT*. Ils ont noté une amélioration pour la reconnaissance lorsque le signal était de qualité téléphonique (contre l'approche traditionnelle des coefficients cepstraux). Par contre, c'était l'inverse lorsque la reconnaissance était effectuée sur les signaux propres.

S. Gagné et D. Nonnon ont aussi envisagé un projet en se basant sur une représentation du signal sous forme d'histogrammes [6, 16]. Le but de leur projet, dirigé par Jean Rouat, était d'élaborer un système capable de séparer automatiquement des locuteurs présents dans un même signal de parole. Ainsi, ils ont vérifié si une telle séparation

était possible par reconstitution du signal à la sortie d'un banc de filtres cochléaires. Évidemment, la difficulté qu'ils avaient résidait dans le choix des canaux à synthétiser pour chacun des locuteurs ainsi que dans l'exclusion des canaux bruités.

La conclusion de ce projet nous laisse croire que ce type de représentation peut offrir de bonnes possibilités dans le domaine de la reconnaissance vocale.

3.2 Cartes auditives

L'architecture et le modèle de neurone utilisés pour créer un réseau sont évidemment des éléments importants. D'un autre côté, il ne faut pas oublier le type de paramètre qui sera appliqué en entrée à ce réseau. En effet, ces éléments doivent être bien adaptés pour pouvoir obtenir de bons résultats lors de la reconnaissance vocale. Les types d'entrées qui semblent le plus intéressants, pour les réseaux de neurones, sont les paramètres perceptifs.

S. Kempf et associés [7] affirment que les caractéristiques sur lesquelles sont basées l'analyse ainsi que la reconnaissance effectuée par le système auditif humain restent en grande partie inconnues. Cependant, elles peuvent être indiquées par des cartes topologiques organisées. En effet, on retrouve des cartes organisées en fréquences à presque tous les niveaux du système auditif.

L'article introduit un modèle qui utilise des cartes primaires de paramètres, provenant des signaux sonores, attribuées à un niveau plutôt bas du système auditif. Ils impliquent qu'à ce niveau, les paramètres reliés aux déplacements de la membrane basilaire dans une section centrée à une location cochléaire précise doivent être représentés de façon similaire dans une zone où les neurones sont syntonisés à cette fréquence particulière.

Le réseau conçu consiste en un *SOM* et se base sur l'hypothèse qu'un certain type de neurone dans un plan avec une fréquence caractéristique donnée, se spécialise dans un patron d'activité instantané particulier. Ce patron émerge de la partition correspondant au niveau de la membrane basilaire. Aussi, on affirme dans cette hypothèse que les neurones adjacents sont syntonisés à des patrons similaires.

Les entrées fournies au réseau sont les résultats d'un traitement effectué en deux phases qui approximent la transformation de la pression du son au cour du temps sur la membrane basilaire en patron de déplacements mécaniques spatiotemporelles. La seconde transformation est une transformation mécanique à électrique effectuée par les cellules ciliées internes.

Les patrons de déplacements spatiotemporels à travers la membrane basilaire sont approximés par un banc de filtre gammatone. De plus, les déplacements continus de cette membrane contrôlent les changements de potentiel des neurones ciliés internes. Il faut noter que ces opérations introduisent une compression non linéaire dans la transformée du signal.

Les entrées utilisées pour tester le système sont des signaux de fréquence pure. Par la membrane basilaire, ils présentent des déplacements dotés de deux paramètres inhérents et mutuellement indépendants : l'amplitude instantanée et la phase. Lorsque ces patrons sont cartographiés dans un plan, le nombre de dimensions du réseau lié correspond au nombre d'axes spatiaux. Cette carte représente les caractéristiques du signal obtenu après la transduction des cellules ciliées (échantillon instantané du déplacement de la membrane et transformation d'Hilbert).

Une autre technique inspirée du système biologique est proposée dans l'article de R. Pichevar et J. Rouat [19] pour solutionner le problème d'analyse des scènes auditives. Les auteurs proposent une architecture neuronale capable de séparer des couples de voyelles en utilisant des cartes *cochléotopiques/AMtopiques*. On fait aussi remarquer que pour certains types de sons, des cartes *cochleotopiques/spectrotopiques*

devraient être utilisées. La différence entre ces deux types de cartes, est que la détection de l'enveloppe est effectuée pour les *CAM*[6], mais pas pour les *CSM*[7].

Le réseau utilisé est bio-inspiré et imite le comportement fonctionnel des neurones biologiques. Brièvement, ces neurones produisent une impulsion lorsque leur potentiel interne franchit un seuil préalablement défini. Cette décharge entraîne une remise à zéro du potentiel. Le réseau présenté code l'information dans la phase des décharges des neurones. En effet, la dynamique des neurones est gouvernée par des oscillateurs à relaxation.

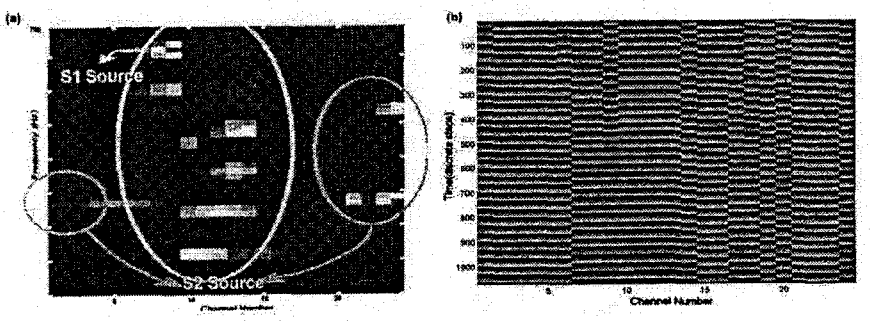


Figure 3.3 – (a) Séparation du signal de deux locuteurs avec représentation par carte auditive. (b) Activité des oscillateurs et synchronisation du réseau permettant la séparation des signaux. [19]

Pour obtenir les représentations des signaux sonores, on filtre d'abord le signal avec un banc de filtres cochléaires (vingt-quatre canaux pour l'exemple de la figure 3.3). Ensuite, seulement dans le cas des *CAM*, on utilise une fenêtre de Hamming pour calculer la *STFT*[8]. Une opération supplémentaire est effectuée pour en augmenter la résolution avant de calculer le logarithme de l'amplitude. La matrice obtenue a comme dimensions : le nombre de canaux du banc de filtres et le nombre d'échantillons de la *FFT*.

[6] *Cochleotopic/AMtopic map*

[7] *Cochleotopic/Spectrotopic map*

[8] *Short time Fourier transform*

Les cartes sont appliquées à la première couche du réseau qui est constituée d'oscillateurs possédant un voisinage immédiat (quatre voisins) et qui est de dimension identique à l'entrée. La couche supérieure comprend des neurones connectés aux neurones d'un seul canal et permet de regrouper les canaux susceptibles d'appartenir à la même source (figure 3.4).

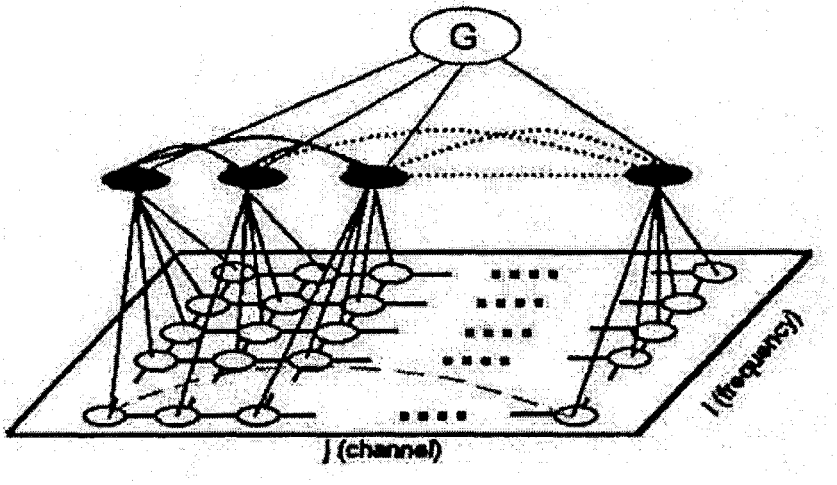


Figure 3.4 – Architecture du réseau pour la séparation de locuteurs (deux couches et contrôleur global). [19]

R. Pichevar et J. Rouat sont encouragés par les résultats obtenus dans la ségrégation des sources et mettent en valeur l'absence de détecteur de fréquence fondamentale ainsi que le fonctionnement non supervisé de leur réseau.

3.3 *Time-sliced paradigm*

Pour éviter des prétraitements coûteux en temps et des algorithmes de segmentation automatique plus ou moins fiables pour les signaux de parole, J. Aoe [1] a créé un réseau de neurones conçu pour reconnaître le signal de parole comme il vient. Ce réseau est rapide et n'utilise qu'une petite quantité de ressources.

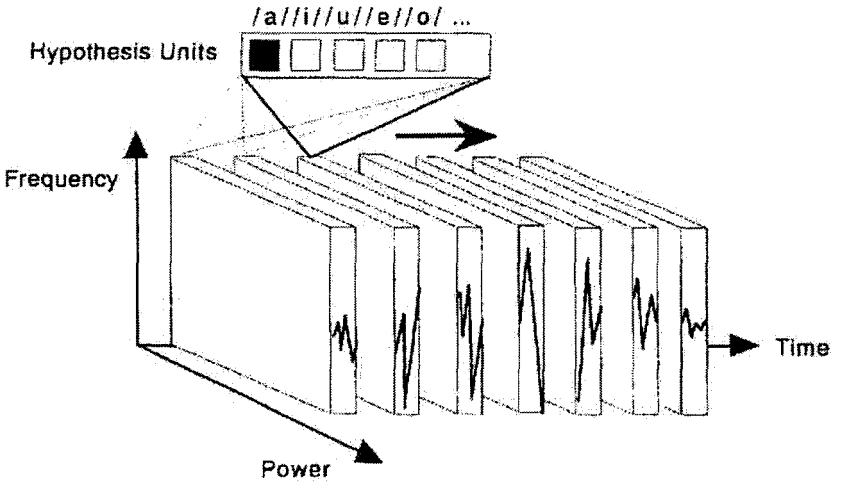


Figure 3.5 – Fonctionnement du *Time spliced paradigm.* [1]

Le réseau de neurones utilise une méthode appelée coupe de temps. L'objectif de cette méthode est de prendre plusieurs tranches du signal comme il arrive, de traiter le flux continu de tranches et d'effectuer immédiatement la reconnaissance. Le signal est généralement obtenu en utilisant une fenêtre (Hamming) et on effectue une FFT[9] sur cette fenêtre pour obtenir un vecteur qui contient les composants spectraux du signal.

Le réseau produit une hypothèse de reconnaissance pour chaque tranche de temps (figure 3.5). Alors l'hypothèse pour un mot est donc une séquence de caractères dont le nombre dépend de la longueur du signal. Il est à noter qu'il est nécessaire de traiter la séquence de caractères pour en conserver l'essentiel.

Le réseau ainsi développé présente, selon les auteurs, une bonne reconnaissance pour les quelques phonèmes japonais qui ont été testés.

[9] *Fast Fourier transform*

3.4 Code par ordre de rang

Une équipe du Centre de Recherche Cerveau et Cognition de Toulouse a fait la conception d'un réseau de neurones pour la reconnaissance d'images. Pour ce simulateur, les chercheurs se sont inspirés du système visuel biologique [25]. Le réseau est composé de neurones à intégrations et décharges qui peuvent être vus comme un modèle simplifié des vrais neurones biologiques. De plus, ce type de neurones permet la programmation par événements du réseau [27].

Le fonctionnement du réseau est basé sur le *Rank order coding* [26, 5]. En se basant sur des expériences effectuées au niveau du système visuel, ils affirment que pour certaines étapes de traitements effectués par le cerveau, les neurones ont rarement le temps de décharger plus d'une fois avant que l'étape suivante doive produire une réponse. La première vague d'impulsions doit alors contenir une bonne partie de l'information sur le stimulus. De cette façon, l'information pour le réseau n'est pas codée par la fréquence des décharges ni par la chronologie précise de ces décharges, mais par l'ordre dans lequel les neurones ont déchargé. Ainsi, les sorties des neurones dépendent de l'ordre de décharge relatif des neurones qui leur sont afférents. Un mécanisme d'inhibition pour le neurone accorde plus d'importance à la cellule qui a déchargé le plus tôt (poids important) et de moins en moins d'importance est attachée aux impulsions suivantes.

En effectuant des simulations avec le réseau pour la reconnaissance d'images, il a souvent été remarqué qu'avec seulement un pourcent de la propagation (1% des neurones qui déchargent), la reconnaissance était possible. En ajoutant à cela le fait que la théorie du codage par ordre de rang ainsi que son algorithme est simple à implémenter, on obtient un simulateur doté d'une grande vitesse d'exécution.

DEUXIÈME PARTIE

OSCILLATEURS

CHAPITRE 4

CEPSTROGRAMME ET OSCILLATEURS

La première approche adoptée est inspirée des réseaux d'oscillateurs présentés à la sous-section 3.1.2 dont l'utilisation pour la segmentation d'une représentation de la parole offre un certain potentiel pour la séparation des sources sonores. Dans le même ordre d'idées, nous souhaitons vérifier si la reconnaissance vocale peut s'effectuer avec un système similaire.

4.1 Structure du réseau

L'architecture imaginée (figure 4.1) comprend un tableau d'oscillateurs reliés (uniquement par les voisins immédiats) comme couche d'entrée. De façon similaire au fonctionnement du réseau *LEGION*, chaque valeur de la représentation est appliquée au neurone correspondant sur cette couche. D'ailleurs, l'importance du lien entre deux oscillateurs voisins est déterminée par la similarité de leur entrée. Deux oscillateurs seront liés par un lien important s'ils possèdent des valeurs semblables en entrées. Dans le cas contraire, un lien faible reliera les oscillateurs. Lors de la simulation, la valeur du lien est déterminée à l'initialisation du réseau et elle demeure constante.

Au cours de la simulation, les oscillateurs qui possèdent des liens suffisamment importants vont se synchroniser. Ces groupes vont alors représenter un caractère important de la représentation. Pour séparer les différents groupes, on utilise un contrôleur

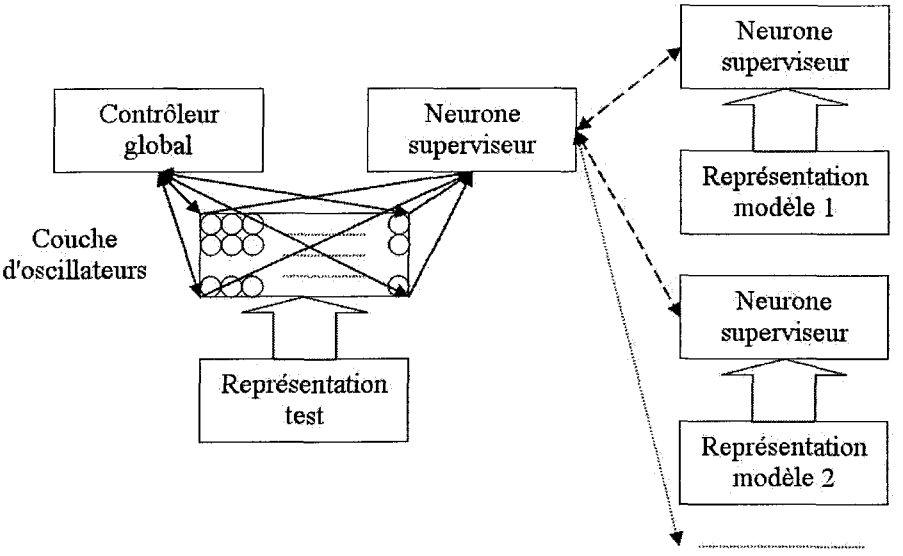


Figure 4.1 – Architecture du réseau avec oscillateurs imaginée pour la reconnaissance vocale.

global. Celui-ci injecte une inhibition à toutes les cellules lorsque l'activité du réseau dépasse un seuil déterminé. Les groupes sont de cette façon séparés par la phase de leurs oscillations. Évidemment, ce composant est connecté avec toutes les cellules d'entrées et les sorties de celles-ci sont toutes reliées au contrôleur global.

Enfin, un neurone superviseur (ou un groupe de neurones) recevra les sorties de la couche d'entrée; il sera ainsi en mesure de comparer en parallèle le patron d'activité généré par celle-ci à celui des modèles. En effet, ce neurone superviseur sera connecté avec d'autres neurones semblables qui recevront l'activité générée par une représentation « modèle ». Les liens entre le neurone « test » et les neurones « modèles » se modifieront au cours de la simulation, de telle sorte que les modèles les plus similaires à la représentation testée possèderont les liens les plus importants. On peut alors imaginer que le modèle sera sélectionné avec plus de précision à travers la simulation et qu'il ne sera pas nécessaire d'attendre la fin du mot pour en faire la sélection.

4.1.1 Implémentation

Dans cette partie, la couche d'entrée du réseau qui effectue la segmentation de la représentation est constituée de neurones oscillateurs. Le modèle de neurones utilisé est celui que l'on retrouve dans le travail de Ramin Pichevar [18], inspiré de *LEGION* [28].

Ces neurones peuvent se connecter à quatre autres cellules. Il existe aussi un lien pour la valeur d'entrée de la cellule, un autre pour l'inhibition provenant du contrôleur global et deux entrées avec lesquelles on peut spécifier les variables initiales. Enfin, chaque neurone possède une sortie qui indique la valeur de son potentiel d'action.

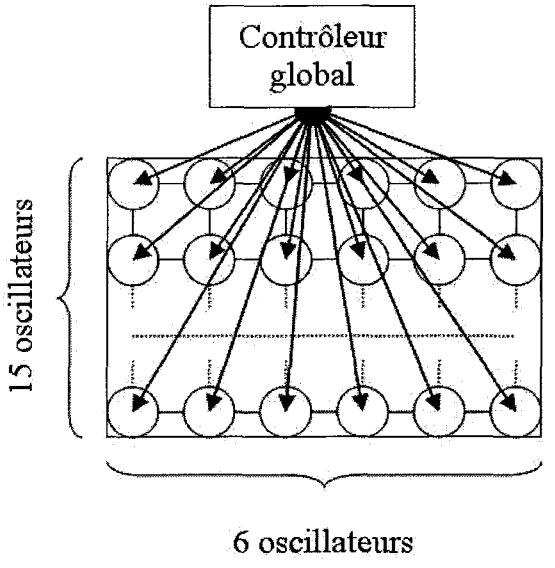


Figure 4.2 – Couche d'oscillateurs pour la segmentation de la représentation.

Pour former la couche d'entrée du réseau, on construit un tableau de 15 cellules par 6 (figure 4.2) où chaque neurone est connecté à ses quatre voisins immédiats (s'ils existent). On a décidé de conserver un nombre limité de cellules pour garder un temps de calcul relativement court. Néanmoins, la portion de l'image utilisée avec ces 90 cellules semble suffisante pour les premiers tests.

4.2 Représentation du signal

La représentation sélectionnée comme entrée au réseau décrit plus haut est un cepstrogramme (figure 4.3). En effet, les coefficients cepstraux sont bien connus dans le domaine de la reconnaissance vocale et ils ont souvent été utilisés [2, 11]. Pour construire le cepstrogramme, on calcule un nombre déterminé de coefficients cepstraux (première dimension) à partir des tranches du signal (le temps devient la deuxième dimension). On observe alors l'évolution des coefficients cepstraux à travers le temps.

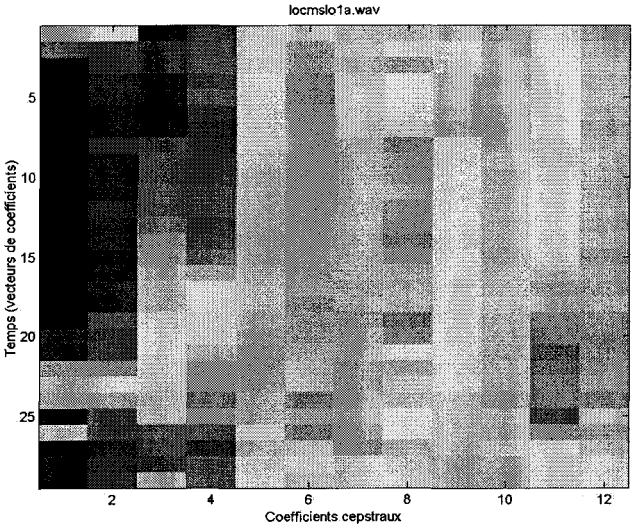


Figure 4.3 – Exemple de cepstrogramme de la première prononciation du chiffre un par le locuteur *slo*.

La reconnaissance s'effectuera d'abord sur des chiffres isolés (zéro à neuf) prononcés par des locuteurs et des locutrices. Il est évident qu'un mécanisme devra être mis en place pour tenir compte des longueurs différentes des prononciations et éventuellement pour permettre une reconnaissance continue. Cependant, nous souhaitons avoir une idée du potentiel de cette approche avant de nous pencher sur ce problème.

4.2.1 Implémentation

Pour obtenir la représentation du signal (chiffres de zéro à neuf prononcés par les locuteurs et les locutrices d'une petite base de données [11] enregistrés en format *wav* à 16 kHz), on génère un vecteur de 6 coefficients cepstraux (calculés avec les fonctions du *VOICEBOX* de *MATLAB*) pour chaque fenêtre de 256 échantillons du signal (avec recouvrement de 128 échantillons).

4.3 Expérimentation

Le premier test consiste à appliquer, en entrée au réseau, une portion des valeurs du cepstrogramme qui correspond aux dimensions, en neurones oscillateurs, de la première couche. Comme l'information importante des coefficients cepstraux se situe dans les premiers coefficients, on utilise un nombre limité de neurones pour cette dimension.

4.3.1 Simulation

Pour effectuer les tests de cette partie, nous utilisons *SIMULINK* du logiciel *MATLAB*. Aussi, ces tests ont été effectués sur un *Pentium 4 Mobile 2 GHz* avec *MATLAB R12*.

D'abord le modèle des neurones est compilé avec *RTW* pour une exécution plus rapide en *MATLAB*. Ensuite, on relie les neurones entre eux et avec le contrôleur global selon l'architecture décrite précédemment. Avant de démarrer la simulation, on détermine les valeurs des variables initiales pour tous les neurones pour obtenir des valeurs les plus différentes possibles (entre zéro et un) et on calcule les coefficients cepstraux pour obtenir la représentation du chiffre. Comme nous sommes limités par les dimensions de la couche d'entrée, on prend une portion de la représentation de

dimensions correspondantes et chaque valeur du cepstrogramme devient l'entrée d'un neurone.

Pour les tests, on sélectionne deux mille itérations (le réseau s'est généralement stabilisé dans cette période) et on peut observer les oscillations de tous les neurones grâce aux différents outils de visualisation de *SIMULINK* ou en les chargeant dans l'environnement de *MATLAB*. Au fur et à mesure que la simulation se déroule, les oscillations (caractérisées par leurs variables initiales) vont se synchroniser et former des groupes. Les différents groupes ainsi formés vont être séparés par l'activité inhibitrice du contrôleur global.

4.3.2 Résultats

À travers les représentations du chiffre un utilisées, on remarque certaines « barres » caractéristiques formées par des coefficients qui présentent peu de variations sur une période de temps (figures 4.4 et 4.5). Cependant, le réseau n'effectue pas la segmentation dans le sens des barres ainsi formées.

En effet, suite à de nombreuses itérations de la simulation, des oscillateurs qui possèdent des valeurs similaires en entrées se regroupent, mais ces groupes semblent être choisis aléatoirement. Il est connu que les valeurs initiales d'un système d'oscillateurs, comme celui présenté dans cette partie, ont un impact important sur la simulation. Pour cette raison, nous avons effectué plusieurs manipulations avec la même représentation d'entrée, mais avec des valeurs initiales générées différemment. Finalement, même en distribuant manuellement les valeurs initiales de façon à ce qu'ils présentent le moins de similitudes avec leurs voisins (figure 4.6), les segmentations obtenues ne caractérisent pas le chiffre (figures 4.7 et 4.8).

Pour favoriser la segmentation des « barres » verticales, nous avons éliminé les liaisons horizontales entre oscillateurs voisins. Dans ce test, les groupes d'oscillateurs

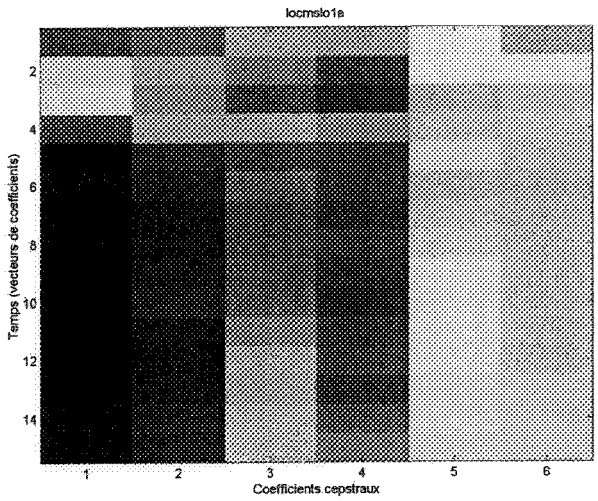


Figure 4.4 – Cepstrogramme de la première prononciation du chiffre un par le locuteur *slo*.

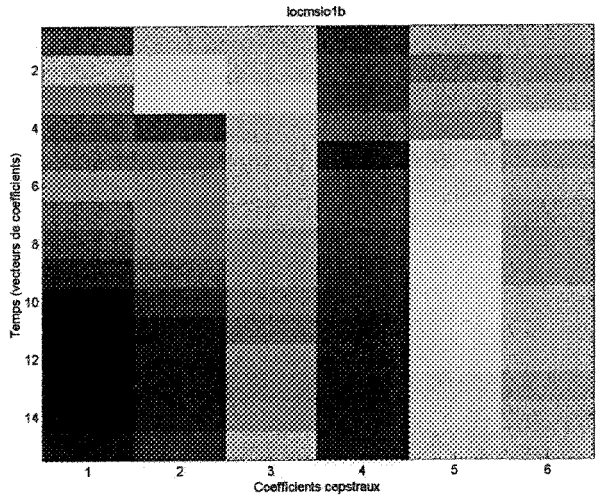


Figure 4.5 – Cepstrogramme de la deuxième prononciation du chiffre un par le locuteur *slo*.

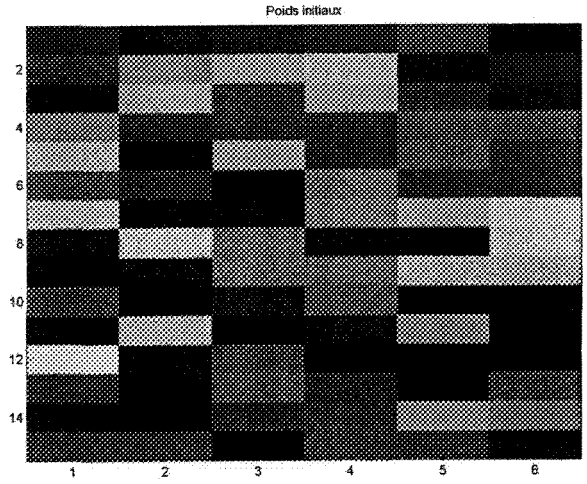


Figure 4.6 – Valeur initiale pour chaque oscillateur de la première couche du réseau de neurones.

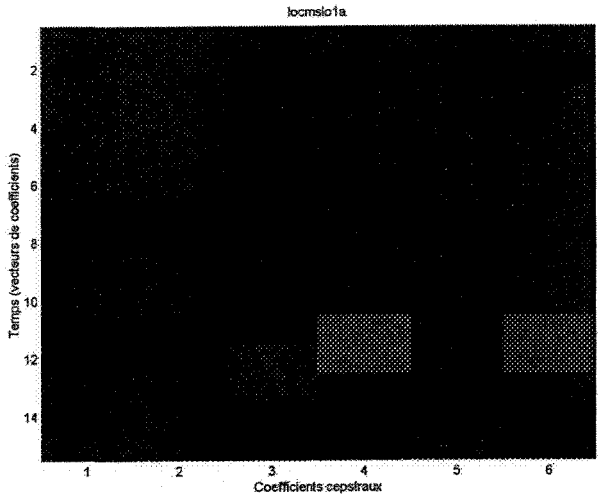


Figure 4.7 – Segmentation de la première prononciation du chiffre un par le locuteur *slo*.

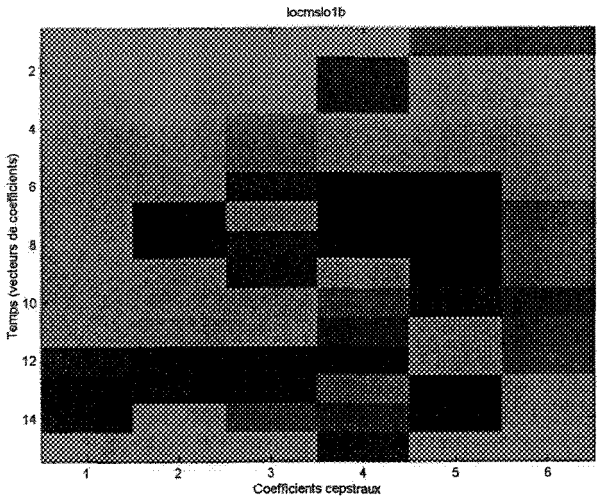


Figure 4.8 – Segmentation de la deuxième prononciation du chiffre un par le locuteur *slo*.

qui se synchronisent et qui survivent à l'inhibition du contrôleur global devrait alors représenter des « barres » plus ou moins longues.

Après avoir effectué des simulations avec le réseau modifié, les segmentations obtenues présentent bien quelques « barres », mais elles sont courtes (deux ou trois neurones) et ne caractérisent pas le chiffre prononcé (figure 4.9).

4.3.3 Discussion

Même si les coefficients cepstraux peuvent être utilisés efficacement pour la reconnaissance vocale, ils ne semblent pas constituer une bonne représentation pour notre approche. En effet, les grandes variations des coefficients cepstraux permettent difficilement un regroupement caractéristique du mot au niveau de la couche d'entrée. Cependant, il est probable que l'architecture du réseau imaginé dans cette partie puisse fonctionner efficacement avec des paramètres perceptifs comme représentations des si-

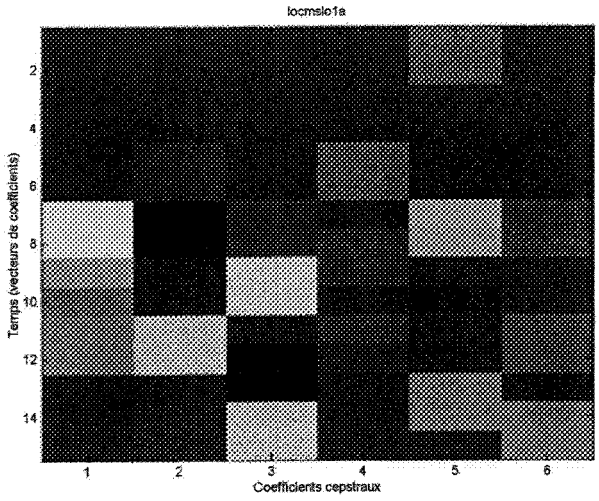


Figure 4.9 – Segmentation de la première prononciation du chiffre un par le locuteur *slo* avec le réseau modifié.

gnaux (travaux cités à la section 3.2). Enfin, le problème, souligné précédemment dans cette partie, concernant les longueurs différentes des mots n'a pas été abordé. Pour le résoudre, la stratégie qui utilise des tranches de temps pour générer des hypothèses (section 3.3) est une solution envisageable.

Il est à noter que les oscillateurs utilisés possèdent des paramètres qui ne sont pas toujours évidents à ajuster. De plus, ce type de neurone demande une quantité importante de calculs et un réseau qui possède un grand nombre de neurones devient difficile à simuler. Un algorithme est présenté dans l'article de D. Wang et D. Terman [28] pour réduire le temps de traitement tout en conservant les propriétés essentielles des oscillateurs à relaxation. Cependant, il n'a pas été implémenté dans le cadre de ce mémoire. Suite à cette partie, une nouvelle approche a été adoptée.

TROISIÈME PARTIE

CODE PAR ORDRE DE RANG

CHAPITRE 5

CODE PAR ORDRE DE RANG

À travers cette partie du projet, nous avons adopté une autre approche. Celle-ci est basée sur la stratégie utilisée par un simulateur développé à Toulouse (voir la section 3.4 à la page 40) soit le codage par ordre de rang (*rank order coding).* Il est à noter que cette partie de la recherche s'est effectuée au sein du laboratoire de l'équipe de Toulouse.

Nous nous sommes d'abord familiarisés avec le codage par ordre de rang, inspirée par le système visuel pour effectuer efficacement la reconnaissance de formes, pour vérifier s'il existe une façon de l'appliquer au niveau de la reconnaissance vocale. En s'inspirant des ouvrages précédents (voir la partie I), nous avons opté pour un modèle simple à banc de filtres et neurones à décharges pour commencer.

Un rapide test (annexe A) nous a donné un bref aperçu du potentiel de notre stratégie et semble indiquer qu'il existe une information caractéristique au mot dans l'ordre relatif des générations d'impulsions que nous obtenons. Ces résultats nous ont encouragés à poursuivre dans cette direction.

5.1 Banc de filtres cochléaires avec seuils multiples

Les sources d'inspiration pour ce test sont les travaux de Steeve Gagné, David Nonnon, Oded Ghitza et G.V. Kiran [6, 16, 21, 9]. Dans ces ouvrages, on filtre le

signal à l'aide d'un banc de filtres cochléaires et ensuite, on note l'intersection de l'enveloppe du signal à la sortie de chaque canal avec des seuils (ce qui est vu comme la génération d'une impulsion). Au contraire de ces chercheurs, nous n'allons pas utiliser les intervalles entre les impulsions, mais leur ordre relatif.

5.1.1 Traitement

Nous tentons d'imiter quelques étapes effectuées par le système auditif humain en espérant trouver une représentation fonctionnelle des chiffres de zéro à quatre, prononcés par cinq locuteurs *mpe*, *rlo*, *slo*, *hgi* et *mtr*, enregistrées en format *wav* à 16 kHz, pour notre approche. Nous filtrons le signal par un banc de filtres *gammatone* de 24 canaux (100 Hz à 8 kHz) implémenté dans *MATLAB* (*auditory toolbox* [23]) et nous utilisons la rectification sur les sorties des canaux (figure 5.1). Pour ce test, les impulsions sont générées lorsque l'amplitude du signal dépasse certains seuils : de $0,02$ à $0,16$ avec un pas de $0,02$. Pour le moment, nous ne permettons qu'une seule impulsion par seuil par canal et lorsqu'un seuil est dépassé, nous considérons que c'est le canal qui a généré l'impulsion. On doit aussi noter que la normalisation n'est pas utilisée pour ce test, ainsi que pour les tests suivants.

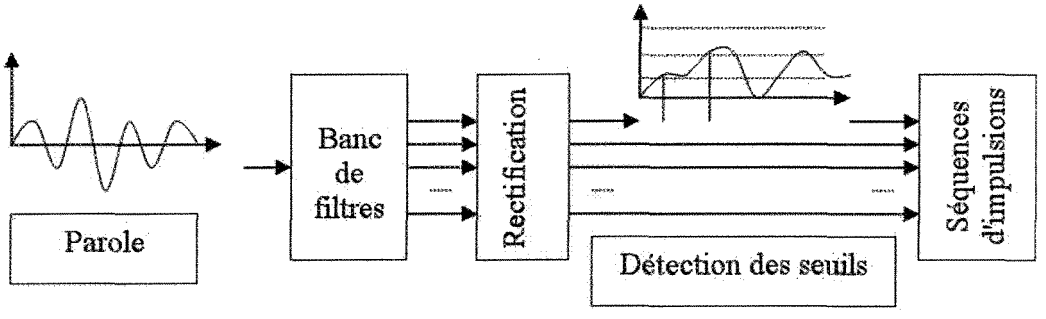


Figure 5.1 – Traitement avec banc de filtres cochléaires et seuils multiples.

5.1.2 Création des modèles

D'abord, nous avons créé un modèle particulier pour les chiffres un à quatre, pour les locutrices *hgi* et *mtr* et pour les locuteurs *rlo* et *slo*, pour un total de 16 modèles. Nous avons jugé important de représenter de façon similaire les locuteurs et les locutrices parce que le sexe du locuteur est un facteur important au niveau de la reconnaissance vocale (influence de la fréquence fondamentale). Les modèles sont conçus en utilisant les vingt premiers canaux les plus susceptibles d'être activés en premier d'après les impulsions générées par les dix prononciations du locuteur ou de la locutrice.

Par exemple, d'après la représentation des impulsions des dix prononciations du chiffre un de la locutrice *mtr* (tableau 5.1), le modèle est 12, 13, 15, 16, 17, 15, 14, 12, 13, etc. Pour générer cette séquence, on sélectionne pour chaque rang (colonne) le canal (ligne) qui produit le plus souvent une impulsion (la plus grande valeur de chaque colonne). Si l'on retrouve, pour un rang, la plus grande valeur dans deux canaux ou plus, les numéros de tous ces canaux sont inscrits dans la séquence. Dans ce cas, on n'accorde aucune importance à l'ordre dans lequel on inscrit les canaux. Cette méthode est utilisée pour sa rapidité et sa simplicité, mais elle n'est évidemment pas optimisée.

Enfin, on accorde un poids à chaque canal en fonction du modèle. Le canal qui devrait franchir le seuil en premier possède le plus grand poids et on va en décroissant pour les canaux suivants.

5.1.3 Reconnaissance

Lors de la reconnaissance, nous comparons à tour de rôle l'ordre des impulsions dans la séquence générée par le traitement du signal avec celui de chaque modèle et le mot reconnu est celui qui possède la plus grande similitude. La similitude entre le signal et le modèle est calculée avec l'équation 5.1. Si l'ordre dans lequel les canaux de

TABLEAU 5.1 – Compilation des dix séquences d'impulsions (chacune caractérisée par leur position dans la séquence et le canal qui l'a générée) produites par les dix prononciations du chiffre un par la locutrice *mtr*

	Rang																			
Canaux	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																				
3																				
4																				
5																				
6					1															
7																				
8																				
9																				
10																				
11	2							1	1	1							1	1		
12	5	2	1	1	1	2	1	3			2	1	1	1		1				1
13		4	1	1	1	2		1	2		2		1		2		1			
14	2	1	2	2	1	2			1	2		1					1		1	
15	1	2	3	1	2				1	2	1	1	1	1				1		1
16			2	3	1	1	3		1	2		1	1	1	1	1		1	1	
17			1	1	2	1	3	2	2		2		2	1	1	3	3	1	1	1
18		1		1	1	1		2	1	1	2		2	2			1	2		2
19							2				1	2	1	1		4			4	1
20							1	1				2		2	2			1	2	
21						1			1	1		1	1	1	2		1	1		
22										1		1			2	1	2			3
23																		2	1	1
24																				

la simulation qui franchissent leur seuil est semblable à celui représenté par le modèle, la similitude va être élevée. Lorsqu'une impulsion est détectée dans un canal, le poids (k) du modèle est modifié en fonction de l'ordre dans lequel cette impulsion a été générée (on utilise une inhibition qui augmente de façon croissante avec le nombre d'impulsions). Enfin, on fait la somme de chaque poids, influencé par l'inhibition, pour obtenir la similitude du signal avec le modèle. Dans ce test, le nombre d'impulsions n est 20 et l'inhibition utilisée est de $0,9$.

$$Similitude = \sum_{k=n}^{1} k \times Inhibition^{Rang del'impulsion} \tag{5.1}$$

On remarque que seuls les premiers canaux à décharger ont un véritable impact sur le pourcentage de vraisemblance. Il est donc possible de faire la reconnaissance avec un nombre limité d'impulsions.

5.1.4 Résultats

Pour tester notre approche, nous avons effectué la reconnaissance sur 200 fichiers (dix prononciations des chiffres un à quatre par les locuteurs *hgi*, *mtr*, *mpe*, *rlo* et *slo*). On doit noter que le locuteur *mpe* n'est pas utilisé dans la conception des modèles et que la sélection des locuteurs et des locutrices s'est faite de façon aléatoire.

Le taux de reconnaissance est obtenu par le rapport des prononciations dont le modèle qui possède la plus grande similitude est bien un modèle du chiffre prononcé sur le nombre total des prononciations du chiffre. Les taux de reconnaissance obtenus (tableau 5.2) sont en général au-dessus de 80 % pour les chiffres de un à quatre. On note cependant une faiblesse pour la reconnaissance du chiffre trois. En effet, pour la locutrice *mtr* et le locuteur *rlo*, moins de la moitié des prononciations ont été reconnues. Les modèles du chiffre quatre présentent les plus grandes confusions. Pour les autres locuteurs, le taux de reconnaissance est supérieur, mais loin du taux de reconnaissance obtenu pour les autres chiffres. Pour le locuteur qui n'est pas utilisé dans la création des modèles (*mpe*), il présente un minimum de 80 % pour la reconnaissance des quatre chiffres.

Le filtrage et la rectification semblent permettre à notre approche de représenter les mots de façon caractéristique. Cependant, le nombre de mots est très limité et déjà on remarque un problème important de confusion entre les chiffres trois et quatre. Comme l'information se retrouve principalement dans les canaux supérieurs, en distribuant cette information sur l'ensemble des canaux, nous devrions obtenir de meilleurs résultats.

TABLEAU 5.2 – Résultats de la reconnaissance des chiffres de un à quatre avec banc de filtres cochléaires à seuils multiples

	Modèles				%	
Locuteurs	1	2	3	4	83,5	
hgi1	10				100	
mtr1	10				100	
rlo1	10				100	98
slo1	10				100	
mpe1	9		1		90	
hgi2		10			100	
mtr2		8		2	80	
rlo2		10			100	96
slo2		10			100	
mpe2		10			100	
hgi3	4		6		60	
mtr3	2		4	4	40	
rlo3	1		3	6	30	54
slo3			6	4	60	
mpe3		2	8		80	
hgi4		1	1	8	80	
mtr4			1	9	90	
rlo4			1	9	90	86
slo4		1		9	90	
mpe4		2		8	80	

5.2 Compression

Pour mieux distribuer les impulsions à travers les canaux du banc de filtres, nous utilisons une forme de compression sur le signal rectifié (racine carrée). Cette étape est inspirée d'un phénomène similaire que l'on retrouve au niveau du système auditif. Aussi, pour tenter d'améliorer la séparation des modèles de ces deux chiffres, nous avons diminué le nombre de seuils qui sont plus espacés et nous prenons chaque seuil dans un canal comme un canal différent.

5.2.1 Traitement

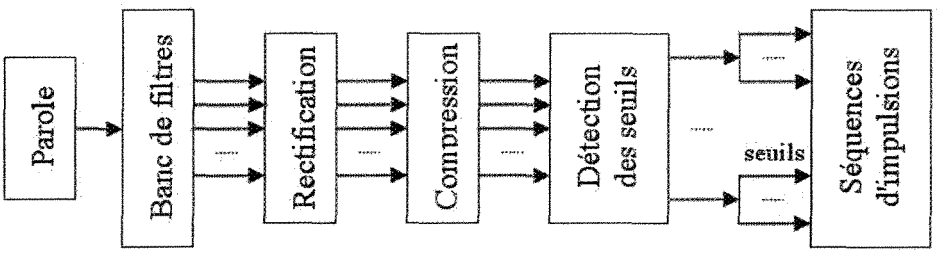


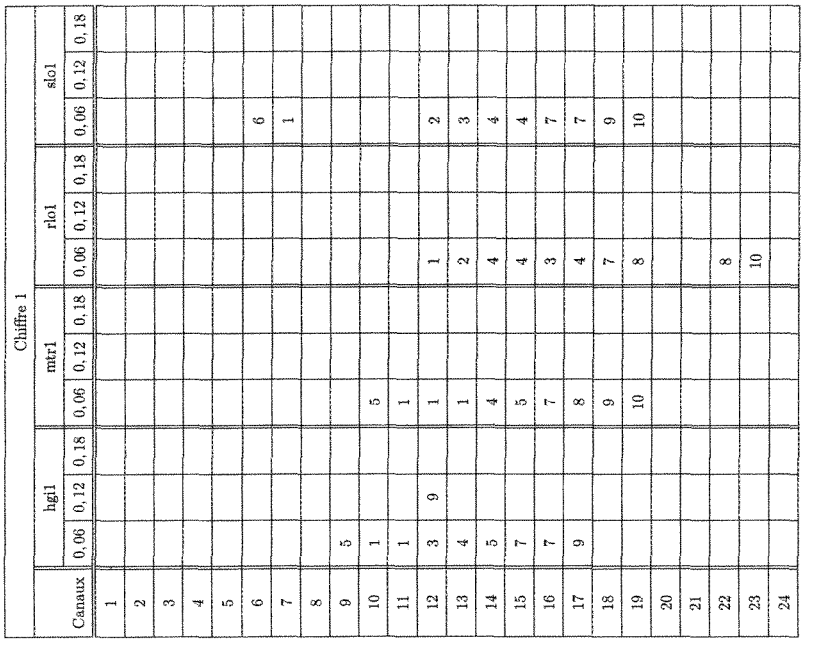
Figure 5.2 – Traitement avec compression.

Comme pour le test précédent, nous filtrons le signal par un banc de filtres de type *gammatone* de 24 canaux (100 Hz à 8 kHz) implémenté dans *MATLAB*. La sortie de chaque filtre est ensuite rectifiée. Pour ce test, nous ajoutons une compression, les impulsions sont générées lorsque la racine carrée de l'amplitude du signal dépasse certains seuils (figure 5.2). Ces seuils agissent comme des neurones qui possèdent des niveaux d'excitation différents, mais ils ne peuvent générer qu'une impulsion. D'ailleurs, quelques brèves expérimentations ont aussi été effectuées pour trouver des valeurs intéressantes comme seuils. Nous avons sélectionné des valeurs qui n'était pas trop sensibles et qui étaient suffisamment espacées pour éviter qu'un canal génère plusieurs fois de suite des impulsions. De plus, chaque seuil dans un canal est maintenant interprété comme un canal différent. Les seuils pour ce test sont $0,06$; $0,12$ et $0,18$. Les signaux utilisés restent les chiffres de zéro à quatre prononcés par cinq locuteurs *mpe*, *rlo*, *slo*, *hgi* et *mtr*, enregistrées en format *wav* à 16 kHz.

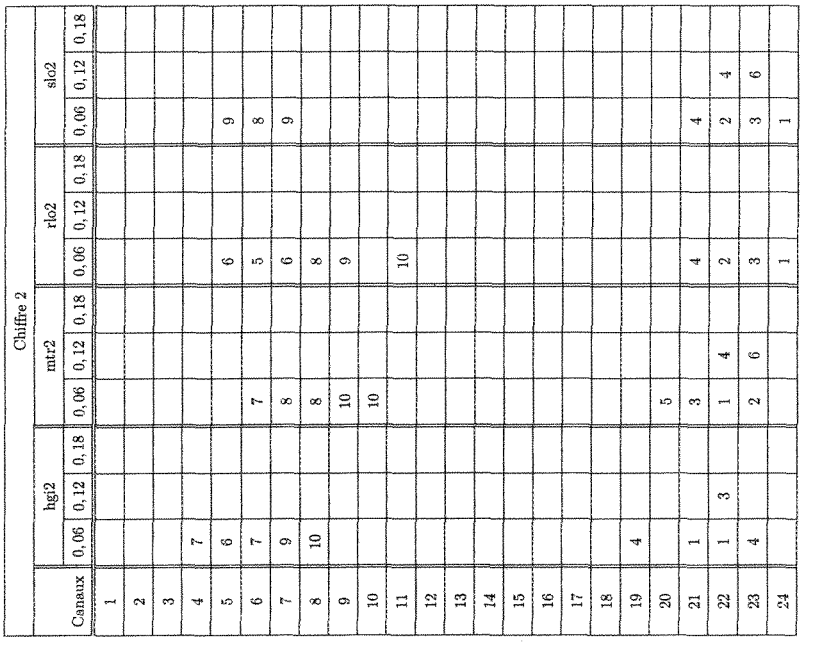
5.2.2 Création des modèles et reconnaissance

La création des modèles ainsi que la reconnaissance s'effectuent de la même façon que pour le test précédent (section 5.1) sauf qu'on n'utilise que les dix premières impulsions des séquences générées par le traitement des signaux. Il arrive quelques fois que plus d'un canal possède la même probabilité de décharger à un moment précis.

TABLEAU 5.3 – Modèles pour la reconnaissance des chiffres de un à quatre.



Chiffre 1																							
	hgi1			mtr1			rlo1			slo1													
Canaux	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18											
1																							
2																							
3																							
4																							
5																							
6										6													
7										1													
8																							
9	5																						
10	1			5																			
11	1			1																			
12	3	9		1			1			2													
13	4			1			2			3													
14	5			4			4			4													
15	7			5			4			4													
16	7			7			3			7													
17	9			8			4			7													
18				9			7			9													
19				10			8			10													
20																							
21																							
22							8																
23							10																
24																							



Chiffre 2																							
	hgi2			mtr2			rlo2			slo2													
Canaux	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18											
1																							
2																							
3																							
4	7																						
5	6						6			9													
6	7			7			5			8													
7	9			8			6			9													
8	10			8			8																
9				10			9																
10				10																			
11							10																
12																							
13																							
14																							
15																							
16																							
17																							
18																							
19	4																						
20				5																			
21	1			3			4			4													
22	1	3		1	4		2			2	4												
23	4			2	6		3			3	6												
24							1			1													

Chiffre 3																							
		hgi3			mtr3			rlo3			slo3												
Canaux	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18		
1																							
2																							
3													2										
4	1			4			1																
5	1			1			2									2							
6	3			2			4						1										
7				3			5						2	8									
8	4			5			6						4										
9	4			5									5										
10	4			7			7						6										
11	7			8			7						6										
12	8			9			9						9										
13	9						10						9										
14	10			9																			
15																							
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							

Chiffre 4																							
		hgi4			mtr4			rlo4			slo4												
Canaux	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18											
1																							
2	2																						
3	2			8																			
4	2			6						3													
5	1			2						3													
6	2	10		1			1			1													
7	6	10		2	10		2			2													
8	7			4			2	7		3													
9	8			5			4			6													
10	9			7			5			7													
11	10			8			6			9													
12				10			8			9													
13							8			8													
14							8																
15																							
16																							
17																							
18																							
19																							
20																							
21																							
22																							
23																							
24																							

Pour cette raison, des canaux sont représentés dans le tableau 5.3 par le même rang. Dans ce cas, un canal est choisi aléatoirement pour le rang qui porte confusion et les autres occupent les rangs suivants. Les modèles obtenus des représentations des chiffres trois et quatre se ressemblent beaucoup et on remarque que les deuxièmes et troisièmes seuils des canaux n'apparaissent que rarement dans les premiers canaux à décharger.

5.2.3 Résultats

TABLEAU 5.4 – Résultats de la reconnaissance avec compression des chiffres de un à quatre

	Modèles				%	
Locuteurs	1	2	3	4	80	
hgi1	10				100	
mtr1	10				100	
rlo1	10				100	96
slo1	8	1		1	80	
mpe1	10				100	
hgi2		10			100	
mtr2		9		1	90	
rlo2		10			100	98
slo2		10			100	
mpe2		10			100	
hgi3			8	2	80	
mtr3			9	1	90	
rlo3	2		8		80	78
slo3			8	2	80	
mpe3	2	1	6	1	60	
hgi4			3	7	70	
mtr4			3	7	70	
rlo4			2	8	80	48
slo4	2		6	2	20	
mpe4			10		0	

Du côté des résultats de la reconnaissance (tableau 5.4), le problème de confusion se rattache maintenant au chiffre quatre (confondu avec le chiffre trois). Par contre, il est plus isolé (locuteurs *mpe* et *slo*). Sinon, les taux de reconnaissance sont légèrement inférieurs au test précédent. Le nombre d'impulsions qui a été réduit pour la création des modèles et à la reconnaissance doit être en partie responsable de la baisse du taux de reconnaissance. Quant à la confusion entre les chiffres trois et quatre, elle est probablement dû aux sons t et k au début des mots. Comme nous n'utilisons que la première impulsion de chaque canal, il est fort possible que la comparaison ne se fasse qu'avec l'information provenant du début de la prononciation. Cette information doit donc être insuffisante pour notre approche.

Pour les tests qui sont présentés dans ce mémoire et dont le traitement contient la compression, la racine carrée est utilisée. D'autres opérateurs, tel le logarithme, peuvent aussi effectuer ce type d'opération, mais ils n'ont pas été testés.

5.3 Regroupement par seuil

Dans la section précédente, on souligne que les deuxièmes et troisièmes seuils des canaux n'apparaissent que rarement dans les premiers canaux à décharger. L'objectif de cette sous-section est de vérifier si les seuils pris séparément caractérisent d'avantage les chiffres. En conservant l'information sur la valeur du seuil qui génère l'impulsion, nous devrions observer une augmentation du taux de reconnaissance. De plus, en spécifiant le nombre d'impulsions pour chaque seuil, on assure un rôle aux seuils à valeur élevée.

5.3.1 Traitement

Effectué dans *MATLAB*, le traitement commence par le filtrage du signal ; les chiffres un à quatre sont prononcés par cinq locuteurs *mpe*, *rlo*, *slo*, *hgi* et *mtr*, enregistrés en format *wav* à 16 kHz par un banc de filtres de type *gammatone* de 24 canaux

(100 Hz à 8 kHz). Ensuite, nous ajoutons une compression en calculant la racine carrée des sorties rectifiées du banc de filtres. Lorsque l'amplitude du signal résultant dépasse les valeurs $0,06$; $0,12$ et $0,18$, une impulsion est générée (figure 5.3). À la différence des tests précédents, les impulsions sont regroupées selon la valeur des seuils (figure 5.4). Ainsi, comme on utilise trois seuils pour ce test, il y a trois groupes.

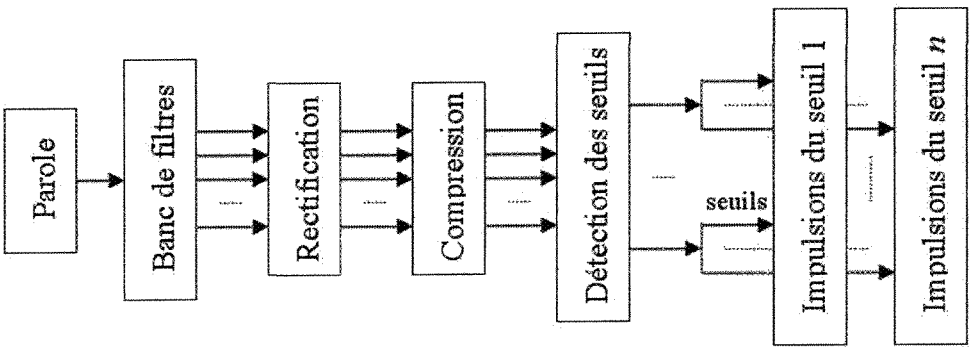


Figure 5.3 – Traitement avec regroupement par seuil.

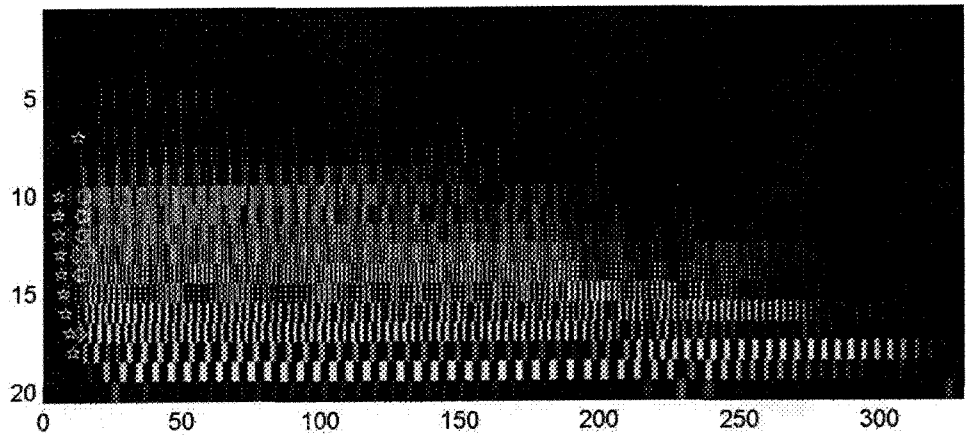


Figure 5.4 – Représentation du regroupement par seuil des impulsions pour la prononciation *locfmtr1a*, 10 impulsions pour le seuil à $0,06$ (étoiles), 5 impulsions pour le seuil à $0,12$ (losanges) et 5 impulsions pour le seuil à $0,18$ (carrés).

Par exemple, pour le signal *locfmtr1a*[1], les dix premiers canaux à générer une impulsion au premier seuil (0,06) sont les canaux : 11, 12, 10, 13, 14, 15, 16, 17, 18 et 7. Les cinq premiers canaux à décharger pour le second seuil (0,12) sont : 13, 14, 17, 11 et 12. Enfin, on obtient la séquence : 14, 11, 10, 12 et 17 pour les cinq premiers canaux qui dépassent la valeur du troisième seuil (0,18). Le modèle de cette seule prononciation serait alors représenté par les poids indiqués au tableau 5.5.

TABLEAU 5.5 – Modèle à seuils regroupés pour la prononciation *locfmtr1a*

Seuils	Canaux																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0,06							1			8	10	9	7	6	5	4	3	2		
0,12											2	1	5	4			3			
0,18										3	4	2	1	5						

5.3.2 Modèles

Pour chaque chiffre et pour chacun des quatre locuteurs : *rlo*, *slo*, *hgi* et *mtr*, on crée maintenant un modèle séparé en trois parties (tableau 5.6). Pour cette expérience, le partie du modèle pour le seuil le plus bas (0,06) contient la valeur des dix premiers canaux les plus susceptibles de générer en premier une impulsion et pour les deux autres seuils (0,12 et 0,18), les parties ne contiennent que les cinq premiers canaux. Cette configuration est choisie puisque pour les seuils les plus élevés, nous sommes limités au niveau du nombre d'impulsions.

[1]première prononciation du chiffre « 1 » par la locutrice *mtr*

TABLEAU 5.6 – Modèles pour la reconnaissance avec seuils groupés.

Chiffre 1																							
		hgi1			mtr1			rlo1			slo1												
Canaux	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18											
1																							
2																							
3																							
4																							
5										6													
6													6										
7													1										
8																							
9	5																						
10	1			5																			
11	2			1																			
12	3	2		1	5	1	1			2													
13	4	1	2	1	1	5	2			3	1	1											
14	5	2	5	4	1	2	4			4	4	3											
15	7	4	1	5	3	2	4			4	1												
16	8	5	2	7	3	2	3	1		7													
17	9			8		5	4	2	1	7	5	5											
18	10			9			7	2	2	9	1	2											
19			2	10			8	4	4	10		3											
20																							
21																							
22							9		4														
23							9																
24								5	3														

Chiffre 2																							
		hgi2			mtr2			rlo2			slo2												
Canaux	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18											
1																							
2																							
3																							
4	5																						
5	5						6			7													
6	7			5			5			5	4												
7	7			6	3		6			5													
8	9			6			8			7													
9	9			8			9																
10				8																			
11				10			10																
12																							
13																							
14																							
15																							
16					5																		
17		4			5					9													
18		2	3		4	4				9	5	5											
19	4	2	1		5	1		4	2	9		1											
20			4	4		3		2	1	3		1											
21	1	4	2	3	5		4	3	2	3	3	3											
22	1	1		1	1	2	2	4	5	1	1												
23	3			2	2	4	3	1	4	1	2	3											
24							1		5														

Chiffre 3

Canaux	hgi3			mtr3			rlo3			slo3		
	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18
1												
2												
3							2					
4	1			4			1					
5	1			1	3		2			2		
6	3			2	1		4			1		
7	6			3			5			2	1	
8	4			6			6			4		
9	4			4			7			5		
10	4			7	4		7			6		
11	8			7	2		9			7		
12	9			9			10			8		
13	10			10	5					9		
14				10	5					10		
15		4				3						
16		3	3		5	3			5			
17		2	3			1		1	2		4	
18		1	1			2		2	1		1	3
19			5			3		3	3		3	1
20								4	4		4	1
21		4	2					5			3	
22		4										
23											4	5
24						3						

Chiffre 4

Canaux	hgi4			mtr4			rlo4			slo4		
	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18	0,06	0,12	0,18
1	2											
2												
3	2			9								
4	2			6						3		
5	1			2						3		
6	2	1		1	2		1			1	1	
7	6	2		2	1		2			2	1	
8	7	3		4	3		2	1		3		
9	8			5			4			6		
10	9			7			5			7		
11	10			8			6					
12				10			7			8		
13							8			8		
14							8			10		
15							8			10		
16			5		5	4						
17		5	5		4	3		5	2			
18			3			1		2	1		3	1
19			4		5	2		2	2		3	2
20		4	1			4			4		5	3
21		5	2			4						
22												3
23								2	4			5
24												

5.3.3 Reconnaissance

Lors de la reconnaissance, nous utilisons la même stratégie que pour les tests précédents, mais une similitude partielle est calculée pour les trois parties du modèle. Une première similitude est obtenue en comparant les premières impulsions de la simulation générées par le premier seuil $(0,06)$ de tous les canaux avec les premières parties des modèles correspondant à ce seuil. Ensuite, on procède de la même façon pour calculer les similitudes partielles au niveau des deux autres seuils. La similitude complète entre les modèles et le signal est la somme pondérée des similitudes partielles pour tous les seuils. La pondération des similitudes est calculée pour ce test par le rapport de la dimension de la partie pour le seuil sur la dimension totale du modèle $(0,5$ (dix impulsions sur un total de vingt) pour le seuil $0,06$ et $0,25$ pour les seuils $0,12$ et $0,18)$.

5.3.4 Résultats

La reconnaissance des chiffres par cette méthode (tableau 5.7) est avantageuse par rapport aux expériences précédentes. En effet, le taux de reconnaissance général dépasse 90 %, une augmentation de plus de 5 %. Il existe toujours une petite faiblesse pour la reconnaissance du chiffre trois (80 %), mais le chiffre quatre est bien reconnu. Pour la reconnaissance des chiffres du locuteur exclu de la production des modèles (*mpe*), le taux de reconnaissance est égal ou supérieur à 80 %, sauf pour le chiffre quatre (60 %).

Nous expliquons l'augmentation du taux de reconnaissance par l'ajout de l'information de la valeur du seuil qui a généré l'impulsion lorsque nous effectuons la comparaison séparément pour chaque seuil. Cependant, cette méthode demande plus de calcul puisqu'il y a autant de comparaisons que de seuils et la conception des modèles est plus compliquée. De plus, il existe toujours une confusion non négligeable entre les modèles du chiffre trois et ceux du chiffre quatre.

TABLEAU 5.7 – Résultats de la reconnaissance avec le regroupement par seuil pour les chiffres de un à quatre

	Modèles				%	
Locuteurs	1	2	3	4	90,5	
hgi1	10				100	
mtr1	10				100	
rlo1	10				100	100
slo1	10				100	
mpe1	10				100	
hgi2		10			100	
mtr2		9		1	90	
rlo2		9		1	90	96
slo2		10			100	
mpe2		10			100	
hgi3			10		100	
mtr3			7	3	70	
rlo3	1		9		90	80
slo3			6	4	60	
mpe3		2	8		80	
hgi4				10	100	
mtr4				10	100	
rlo4			1	9	90	86
slo4			2	8	80	
mpe4			4	6	60	

5.4 Regroupement par seuil pour les voyelles

Comme nous limitons la génération d'impulsions à une seule impulsion par canal, nous limitons la création des modèles ainsi que la reconnaissance au début du mot. Pour des mots courts, cette approche ne devrait pas causer de problème. Cependant, pour des mots plus longs, il est fort possible qu'elle nous limite beaucoup. C'est probablement le cas pour les chiffres trois et quatre où le t et le q au début des mots possèdent des représentations suffisamment similaires pour causer l'importante confusion que nous avons observée. Une solution serait d'implémenter un mécanisme pour que les canaux puissent générer plus d'une impulsion de façon cohérente. Une période réfractaire suite

à une décharge pourrait être utilisée pour chaque canal ; la stratégie de découper le signal en tranche de temps présentée dans la section 3.3 pourrait également être suivie. Cependant, ces solutions risquent d'être complexes et nous préférons continuer notre étude avec des tests simples pour rapidement avoir une idée du potentiel de notre approche.

Pour poursuivre, nous avons donc décidé de tester la méthode utilisée au test précédent (section 5.3) avec des prononciations de voyelles. Ces signaux sont en général plus courts que les chiffres et l'information extraite par notre approche devrait être plus représentative pour ainsi produire une meilleure reconnaissance. De plus, en utilisant des voyelles isolées (a, e, i, o et u), nous évitons le problème complexe rattaché à la reconnaissance des phonèmes. Des voyelles ont déjà été utilisées avec succès dans bon nombre de travaux touchant à la reconnaissance vocale à l'aide de réseaux de neurones [8, 10, 19]. Cependant, comme il s'agit de parole voisée, la fréquence fondamentale risque d'être bien présente à travers le traitement.

5.4.1 Traitement

Les voyelles a, e, i, o et u sont prononcées dix fois par cinq locuteurs : *elo*, *mpe*, *rlo*, *slo* et *str* et cinq locutrices : *fpe*, *hgi*, *jla*, *lau* et *mtr*. Les cinq locuteurs des tests précédents sont utilisés et cinq autres ont été rajoutés. Comme pour les tests précédents, les signaux sont enregistrés en format *wav* à 16 kHz.

Le traitement, que nous effectuons dans *MATLAB*, commence par le filtrage du signal par un banc de filtres de type *gammatone* de 24 canaux (100 Hz à 8 kHz). Ensuite, nous ajoutons une compression en calculant la racine carrée des sorties rectifiées du banc de filtres. Lorsque l'amplitude du signal résultant dépasse les valeurs $0,06$, $0,12$ et $0,18$, une impulsion est générée. Pour ce test, les impulsions sont rassemblées en trois groupes, selon la valeur des seuils.

5.4.2 Modèles

Un modèle séparé en trois parties (une partie par seuil) est créé pour chaque mot et pour chacun des quatre locuteurs : *hgi*, *mtr*, *rlo* et *slo*. La partie du modèle pour le seuil le plus bas $(0,06)$ contient la valeur des dix premiers canaux les plus susceptibles de générer en premier une impulsion et pour les deux autres seuils $(0,12$ et $0,18)$, les parties ne contiennent que les cinq premiers canaux.

5.4.3 Reconnaissance

Comme pour effectuer la reconnaissance à la section précédente, on calcule une similitude partielle par seuil (les impulsions générées par chaque seuil sont comparées à la section correspondante du modèle) et on obtient la similitude complète en effectuant la somme des similitudes partielles pondérées.

5.4.4 Résultats

TABLEAU 5.8 – Résultat de la reconnaissance pour seuils regroupés avec les voyelles et les locuteurs utilisés dans la création des modèles (données d'apprentissage)

Modèles												%
Locutrices	a	e	i	o	u	Locuteurs	a	e	i	o	u	93,5
hgia	10					rloa	9			1		95
mtra	10					sloa	9		1			
hgie	1	7		2		rloe		10				85
mtre	1	9				sloe		8		2		
hgii			10			rloi			10			97,5
mtri			9		1	sloi			10			
hgio	1			9		rloo				10		95
mtro				10		sloo			1	9		
hgiu					10	rlou			1		9	95
mtru			1		9	slou					10	

TABLEAU 5.9 – Résultat de la reconnaissance pour seuils regroupés avec les voyelles et les locuteurs exclus de la création des modèles

Modèles												%
Locutrices	a	e	i	o	u	Locuteurs	a	e	i	o	u	87
fpea	10					eloa	10					
jlaa	9	1				mpea	10					96,7
laua	9	1				stra	10					
fpee	2	6		2		eloe		8		2		
jlae		10				mpee		7		3		80
laue		10				stre		7		3		
fpei			9		1	eloi			9		1	
jlai			10			mpei			4		6	78,3
laui			6		4	stri			9		1	
fpeo	5			5		eloo		1		9		
jlao				10		mpeo		2		8		85
lauo				10		stro		1		9		
fpeu					10	elou			1		9	
jlau		1			9	mpeu					10	95
lauu			1		9	stru					10	

Les résultats de la reconnaissance des voyelles en regroupant les seuils sont très encourageants. Pour les cinq voyelles, on obtient une reconnaissance supérieure à 85 % pour les locuteurs utilisés dans la création des modèles (tableau 5.8) et pour les autres (tableau 5.9), une reconnaissance supérieure ou égale à 80 % (excepté pour la voyelle « o » qui présente une reconnaissance de 78 %). Les confusions sont surtout entre « e » et « o » ainsi que « i » et « u ». En observant la distribution du premier formant dans les fréquences pour les voyelles (figure 2.2 à la page 8), on remarque que les espaces des voyelles mal reconnues se chevauchent. Le « u » recouvre presque en totalité le « i » et le « o » se retrouve dans l'espace du « e ». De son côté, le « a » est pratiquement seul à sa position. Il est logique d'imaginer que le système se base en partie sur la position des formants, puisque c'est à ces endroits que l'on retrouve la plus grande amplitude dans le spectre du signal.

5.5 Modèle binaire

En observant les modèles obtenus dans l'expérience précédente, on remarque qu'ils caractérisent de façon assez particulière les différentes voyelles. De plus, on sait que l'utilisation des séquences temporelles des impulsions générées par notre approche avec ces modèles produit, à quelques exceptions près, un taux de reconnaissance supérieur à 80 %. Pour cette expérience, nous sommes intéressés à savoir s'il est possible d'effectuer la reconnaissance vocale avec notre approche en éliminant l'inhibition (l'information temporelle de la séquence). On simplifierait alors, de façon importante, la reconnaissance.

5.5.1 Traitement

Le traitement qui est appliqué aux différentes prononciations des voyelles isolées (a, e, i, o et u) par nos locuteurs et locutrices est le même que celui décrit au test précédent. Le signal est filtré par un banc de filtres, chaque sortie des filtres est par la suite rectifiée et compressée en utilisant la racine carrée. Lorsque l'amplitude du signal dépasse les seuils définis, une impulsion est générée. La stratégie des seuils regroupés est toujours utilisée, alors on regroupe ces impulsions par la valeur du seuil qui les a générées.

5.5.2 Modèles

Avec les mêmes éléments qu'en 5.4, les modèles sont convertis en modèles binaires pour la reconnaissance, les canaux activés ne sont plus pondérés en fonction de l'ordre des impulsions dans la séquence (« 1 » pour les canaux activés en premier et « 0 » pour les autres).

En reprenant l'exemple avec le signal *locfmtr1a* dont les dix premiers canaux à générer une impulsion au premier seuil $(0,06)$ sont les canaux : 11, 12, 10, 13, 14, 15,

16, 17, 18 et 7. Les cinq premiers canaux à décharger pour le second seuil (0, 12) sont : 13, 14, 17, 11 et 12. Enfin, on obtient la séquence : 14, 11, 10, 12 et 17 pour les cinq premiers canaux qui dépassent la valeur du troisième seuil (0, 18). Le modèle binaire de cette prononciation est représenté au tableau 5.10.

TABLEAU 5.10 – Modèle binaire à seuils regroupés pour la prononciation *locfmtr1a*

Seuils	Canaux																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0, 06							1			1	1	1	1	1	1	1	1	1		
0, 12											1	1	1	1			1			
0, 18										1	1	1	1	1						

5.5.3 Reconnaissance

Lors de la reconnaissance, les similitudes partielles (on compare les impulsions générées par chaque seuil avec la section correspondante du modèle) ne tiennent plus compte de l'ordre de génération des impulsions. On détermine maintenant quels sont les n premiers canaux activés par la prononciation traitée et on compte le nombre de ces canaux qui sont aussi activés dans le modèle d'une voyelle. Par exemple, la prononciation serait exactement représentée par un modèle si les n premiers canaux activés pour chaque seuil étaient les mêmes. Ensuite, on obtient la similitude complète en effectuant la somme des similitudes partielles qui peut être pondérée. Enfin, une prononciation est correctement reconnue si le modèle le plus similaire est bien un modèle de la voyelle. Dans le cas ou plusieurs modèles possèdent la plus grande similarité, la prononciation est correctement reconnue si et seulement si la confusion se limite aux modèles de la bonne voyelle.

TABLEAU 5.11 – Résultat de la reconnaissance pour les locuteurs utilisés dans la création des modèles binaires des voyelles (données d'apprentissage)

Modèles														%
Locutrices	a	e	i	o	u	?	Locuteurs	a	e	i	o	u	?	85
hgia	10						rloa	8			2			92,5
mtra	9					1	sloa	10						
hgie		7			1	2	rloe		6			1	3	65
mtre	1	8				1	sloe		5		1		4	
hgii			10				rloi			7			3	87,5
mtri			8			2	sloi			10				
hgio				10			rloo		1		8		1	95
mtro				10			sloo				10			
hgiu					10		rlou			1		8	1	85
mtru					6	4	slou					10		

TABLEAU 5.12 – Résultat de la reconnaissance pour les locutrices non utilisées dans la création des modèles binaires des voyelles

Modèles														%
Locutrices	a	e	i	o	u	?	Locuteurs	a	e	i	o	u	?	83,3
fpea	10						eloa	10						95
jlaa	9					1	mpea	10						
laua	9	1					stra	9					1	
fpee		4		1		5	eloe		5				5	66,7
jlae		10					mpee		7		1		1	
laue		10					stre		4		2		4	
fpei			10				eloi			9			1	81,7
jlai			10				mpei			9			1	
laui			10				stri			1			9	
fpeo				7		3	eloo				10			90
jlao				10			mpeo				8		2	
lauo				10			stro				9		1	
fpeu					10		elou					9	1	83,3
jlau					8	2	mpeu					8	2	
lauu			1		5	4	stru					10		

5.5.4 Résultats

Suite aux résultats de l'expérience, il semble que l'information temporelle dans les séquences d'impulsions ne soit pas essentielle pour effectuer la reconnaissance vocale avec notre approche. On note cependant une baisse au niveau du taux de reconnaissance sur l'ensemble des locuteurs (tableaux 5.11 et 5.12), mais le taux de reconnaissance global reste supérieur à 80 %. Cette diminution est principalement attribuable à la confusion que l'on retrouve lors de la reconnaissance. En effet, comme on n'utilise pas l'information temporelle, des séquence d'impulsions provenant des même canaux, mais dans un ordre différent, sont perçues de façon identiques par notre approche et il arrive souvent que plus d'un modèle soit sélectionné comme reconnu (taux de reconnaissance égal pour deux ou plusieurs modèles).

5.6 Influence du nombre de canaux et d'impulsions

À travers cette section, nous analysons les résultats d'expériences qui touchent le nombre de canaux du banc de filtres ainsi que sur le nombre d'impulsions qui sont utilisées lors du traitement, ceci afin d'avoir une idée de l'influence que peuvent avoir ces facteurs sur le taux de reconnaissance de notre prototype (voir l'annexe B à la page 133).

5.6.1 Modèles

À la différence des autres tests où les modèles étaient créés en utilisant dix ou plusieurs prononciations, les modèles des cinq voyelles (a, e, i, o et u) pour la reconnaissance de cette section vont être créés à partir de la prononciation (à travers les dix prononciations pour chacun des dix locuteurs) qui possède les plus hauts taux de reconnaissance global (rapport du nombre de prononciations représentées par un modèle du bon mot sur le nombre total de prononciations). Cette méthode n'est évidemment

pas optimisée, mais elle est simple et de plus, l'option de charger des modèles généraux n'existe pas dans la version du prototype que l'on utilise.

Pour sélectionner les modèles des tests suivants, nous filtrons le signal par un banc de filtres de 24 canaux (100 Hz à 8 kHz) implémenté dans *MATLAB*. La sortie de chaque filtre est ensuite rectifiée et les impulsions sont générées lorsque la racine carrée de l'amplitude du signal dépasse certains seuils (0,06, 0,12 et 0,18). On utilise l'information temporelle des séquences d'impulsions grâce à une inhibition avec une modulation de 0,9, mais l'information particulière de la valeur du seuil qui a généré l'impulsion n'apparaît ni dans les modèles ni lors de la reconnaissance. Les signaux utilisés restent les voyelles : a, e, i, o et u, prononcées par cinq locuteurs : *elo*, *mpe*, *rlo*, *slo* et *str* et cinq locutrices : *fpe*, *hgi*, *jla*, *mtr* et *lau*, enregistrées en format *wav* à 16 kHz. Après avoir obtenu la séquence des vingt premières impulsions qui caractérise chaque signal, on calcule le taux de reconnaissance global en effectuant la moyenne des taux de reconnaissance (rapport du nombre de prononciations représentées par un modèle du bon mot sur le nombre total de prononciations) pour chacune des prononciations. Par la suite, on sélectionne la prononciation pour les locutrices et pour les locuteurs qui possèdent les plus hauts taux de reconnaissance. Dix modèles (un modèle pour les locuteurs et un autre pour les locutrices pour les cinq voyelles) sont donc sélectionnés dans les expériences suivantes (tableau 5.13).

5.6.2 Reconnaissance

Pour le premier test, les seuils sont groupés (les impulsions ne sont pas caractérisées par la valeur du canal qui les a générées) et on fait varier le nombre de canaux du banc de filtres ainsi que le nombre d'impulsions pour la reconnaissance et l'apprentissage de vingt à cent par bonds de vingt. La reconnaissance s'effectue, pour ce test et les tests suivants, sur l'ensemble des prononciations (dix locuteurs qui prononcent dix fois les cinq voyelles pour un total de cinq cents prononciations).

TABLEAU 5.13 – Liste des modèles sélectionnés

Voyelle	Modèle	Taux de reconnaissance
a	locfjlaa5	83, 2
	locmmpea2	87, 3
e	locffpee8	83, 1
	locmstre7	83, 6
i	locfmtri1	84, 1
	locmsloi0	88, 2
o	locfhgio8	91, 6
	locmsloo8	91, 2
u	locfmtru2	87, 1
	locmstru6	90, 6

Pour déterminer la performance des différentes combinaisons des paramètres, on se base sur la moyenne des taux de reconnaissance pour chaque voyelle. Il est à noter que pour la reconnaissance, on utilise la similitude relative. En effet, il arrive souvent (surtout pour un petit nombre de canaux) que le nombre d'impulsions produites par le traitement du signal soit inférieur à celui désiré. Dans le cas où le nombre d'impulsions de la séquence du signal traité est différent du nombre d'impulsions dans la séquence du modèle, on calcule la similitude en utilisant, pour les deux séquences, le plus petit nombre d'impulsions.

Nous avons effectué le traitement ainsi que la reconnaissance avec le prototype de l'annexe B à la page 133. On ajoute un effet de fondu (*fade*) sur les cents premiers et les cents derniers échantillons du signal pour éviter les discontinuités. Les signaux des voyelles ont été séparés de façon à ce que le contenu ne soit pas affecté par cette opération et en conservant un nombre limité d'échantillons, ce paramètre n'influence pas les résultats des tests. Un vecteur peut ensuite être utilisé pour spécifier le nombre

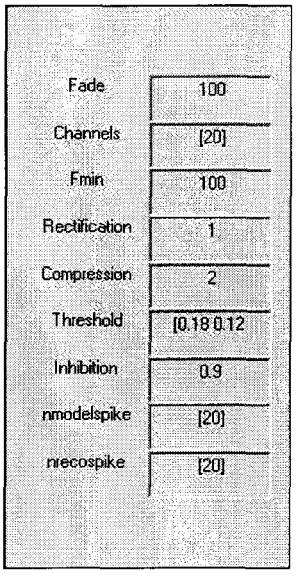


Figure 5.5 – Paramètres du traitement avec *sna*.

de canaux du banc de filtres (*Channels*). Dans ce cas, si on utilise *Automatch*, les traitements sont faits pour chaque élément du vecteur et les résultats de la reconnaissance sont ajoutés à la file dans le fichier sélectionné. Les paramètres : fréquence minimale du banc de filtres (*Fmin*), rectification et compression sont ensuite spécifiés. On utilise un vecteur pour donner la valeur des trois seuils du traitement (en ordre décroissant dans *Threshold*). Enfin, la modulation de l'inhibition pour la reconnaissance, le nombre d'impulsions des modèles (*nmodelspike*) et du signal à reconnaître (*nrecospike*) sont spécifiés. Lorsque les paramètres des tests sont fournis (figure 5.5), on sélectionne la liste de nos modèles (*Load model*), des fichiers à reconnaître (*Load reco*) et on effectue le traitement (*Automatch*). Les similitudes entre chaque signal (lignes) et chaque modèle (colonnes) sont ensuite générées dans le fichier spécifié à la suite des caractéristiques du test. Il ne reste plus qu'à trouver le modèle le plus similaire pour chaque signal et à calculer le taux de reconnaissance pour chaque voyelle.

5.6.3 Résultats

En général, le taux de reconnaissance augmente avec le nombre d'impulsions utilisées à la reconnaissance et à l'apprentissage (figure 5.6). Cependant, il diminue lorsque le nombre de canaux du banc de filtres augmente. Il est à noter que pour un petit nombre de canaux, il est difficile de générer un grand nombre d'impulsions. Ce fait expliquerait le plateau atteint par certaines courbes. Pour d'autres, la diminution est probablement attribuable au fait que notre approche demande que le modèle possède un certain nombre d'impulsions pour caractériser le signal, mais qu'avec une augmentation trop importante du nombre d'impulsions ou de canaux, le modèle finit par devenir trop spécifique à la prononciation qui l'a générée. Enfin, le taux de reconnaissance maximal est atteint avec un petit nombre de canaux et un nombre plus important d'impulsions (presque 90 %).

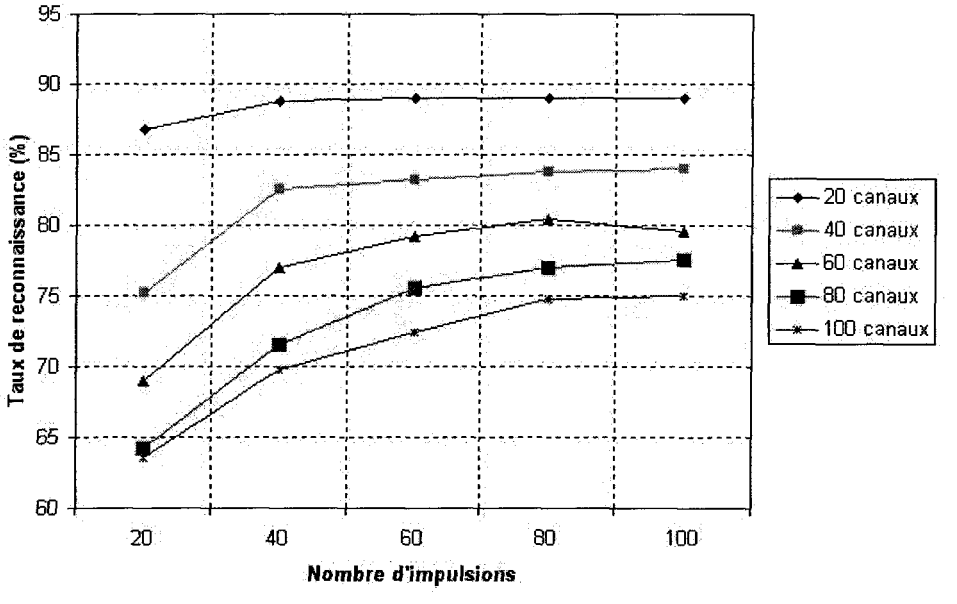


Figure 5.6 – Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions.

Au niveau de la reconnaissance des voyelles, les taux de reconnaissance global du « i » et du « o » sont élevés (aux alentours de 90 %). Le « a » ainsi que le « u » sont reconnus à 75 % et « e » est faiblement reconnu (56 %). En comparant les résultats obtenus avec ceux des expériences ultérieures, on remarque que ce sont sensiblement les mêmes voyelles qui sont bien et moins bien reconnues.

La sélection des modèles constitue un facteur important qui influence les taux de reconnaissance. En effet, nous avons utilisé un ensemble particulier de paramètres lors de la sélection des modèles et si ces paramètres varient, les modèles seront représentés différemment. Nous n'avons pas approfondi la méthode de sélection des modèles pour ces tests, cette étude pourra faire parti d'un travail parallèle.

5.6.4 Modèles binaires

Les modèles binaires sont ensuite testés de façon similaire. Les manipulations restent les mêmes, excepté que la création des modèles et la reconnaissance s'effectuent avec la stratégie décrite à la section 5.5 à la page 73.

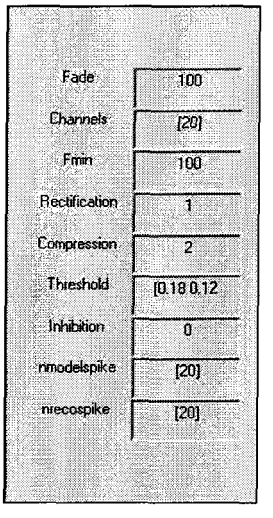


Figure 5.7 – Paramètres du traitement avec *sna* et des modèles binaires.

Nous testons encore l'influence du nombre de canaux du banc de filtres ainsi que le nombre d'impulsions des séquences pour la reconnaissance et l'apprentissage en les faisant varier de 20 à 100 par bonds de 20 (figure 5.7). Les dix modèles restent les mêmes (tableau 5.13) et on utilise toujours l'ensemble des prononciations pour effectuer la reconnaissance.

5.6.5 Résultats

En observant les résultats de ce test (figure 5.8), on remarque que l'utilisation des modèles binaires a un impact important sur le taux de reconnaissance. En effet, le taux de reconnaissance augmente maintenant en fonction du nombre d'impulsions jusqu'à ce que celui-ci atteigne une valeur semblable au nombre de canaux du banc de filtres. Par la suite, le taux de reconnaissance diminue. Cette diminution provient en partie du fait qu'on n'utilise plus l'information temporelle des séquences. Notre approche demande qu'un modèle soit spécifié par une séquence assez importante d'impulsions, mais en augmentant le nombre d'impulsions sans spécifier l'ordre de génération, on se trouve à caractériser un plus grand nombre de séquences de façon identique. Par exemple, dans un cas tout à fait hypothétique, si la séquence ne contient qu'une impulsion qui est générée par un banc de filtres à 20 canaux qui ne contiennent qu'un seuil, on peut créer 20 modèles qui sont caractérisés par un seul canal. Dans le cas ou la séquence contient deux impulsions, 190 modèles englobent maintenant deux séquences temporelles différentes (le modèle constitué des impulsions des premier et deuxième canaux caractérise les séquences 1-2 et 2-1). De plus, en augmentant le nombre d'impulsions, le modèle devient de plus en plus spécifique à la prononciation qui a généré le modèle.

Cette fois, l'augmentation du nombre de canaux du banc de filtres semble aider la reconnaissance. En effet, les meilleurs taux de reconnaissance sont obtenus avec un nombre élevé de canaux. Comme un plus grand nombre de canaux nous permet un plus grand nombre de séquences différentes, on imagine que cette diversité aide la

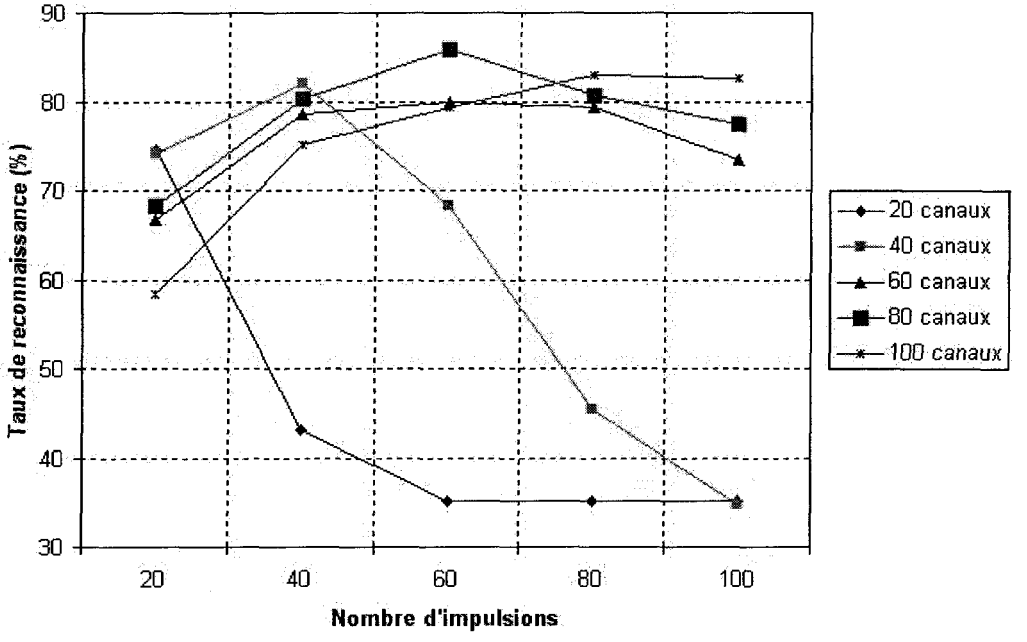


Figure 5.8 – Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions pour des modèles binaires.

reconnaissance. Cependant, comme pour le nombre d'impulsions, l'augmentation du nombre de canaux finit par nuire à la reconnaissance lorsque les modèles deviennent trop spécifiques à la prononciation qui les a générés.

Les voyelles qui possèdent les meilleurs et les pires taux de reconnaissance, quoique plus faibles qu'au test précédent, restent néanmoins les mêmes. Aussi, les taux de reconnaissance des voyelles produits par ce test présentent de plus grandes variations aux paramètres que ceux obtenus dans le test précédent. Ces observations sont probablement le résultat des confusions entre les modèles, qui sont évidemment plus nombreuses lorsqu'on utilise des modèles binaires.

5.6.6 Stratégie des seuils séparés

La stratégie des seuils séparés a par la suite été testée de la même façon que pour les tests précédents (figure 5.9). Pour ce test, on utilise l'information temporelle des séquences d'impulsions pour la création des modèles. Par contre, on sépare les impulsions par la valeur du seuil qui l'a générée. Pour conserver une similitude au niveau du nombre d'impulsions utilisées, nous avons remplacé les valeurs vingt, quarante, soixante, quatre-vingts et cent par cinq cinq dix, dix dix vingt, quinze quinze trente, vingt vingt quarante et vingt-cinq vingt-cinq cinquante. Le premier chiffre correspond au nombre d'impulsions pour le seuil à 0, 18, le second pour le seuil à 0, 12 et le dernier chiffre pour le seuil à 0, 6. Les variations du nombre de canaux du banc de filtres et les prononciations utilisées pour créer les modèles restent inchangées. Enfin, le taux de reconnaissance se calcule toujours en utilisant l'ensemble des prononciations.

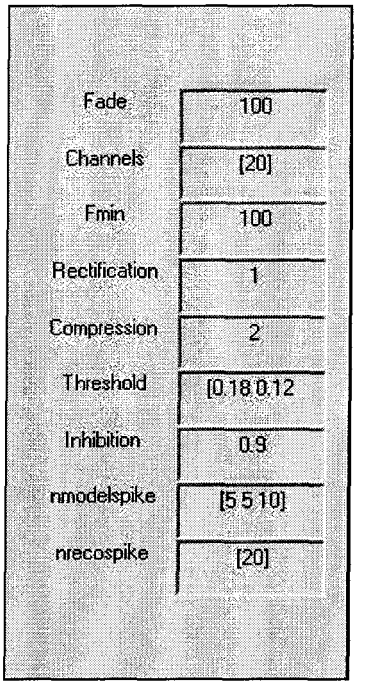


Figure 5.9 – Paramètres du traitement avec *sna* et séparation des seuils.

Les résultats obtenus avec ce test (figure 5.10) montrent les mêmes tendances que celles du premier test de cette sous-section. En général, le taux de reconnaissance diminue avec le nombre de canaux et augmente en fonction du nombre d'impulsions.

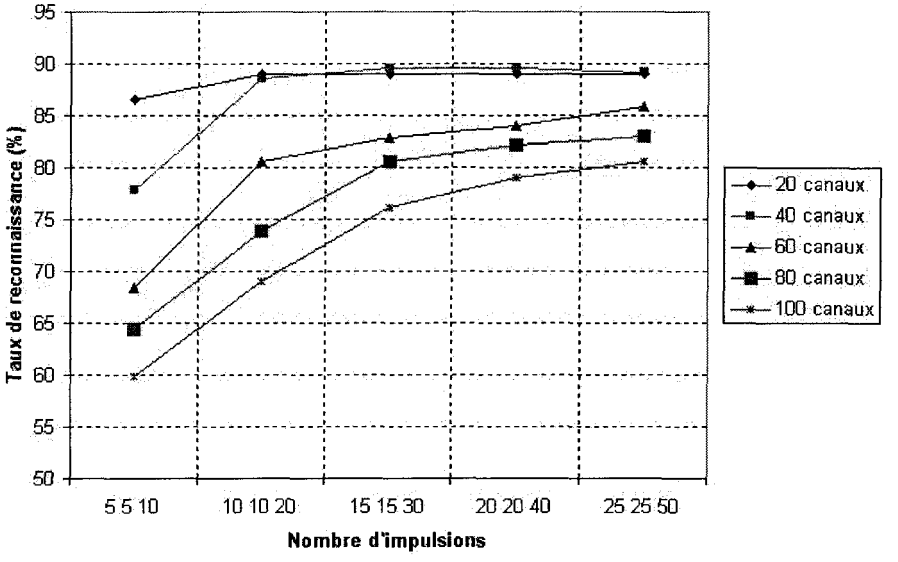


Figure 5.10 – Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions pour des modèles avec séparation des seuils.

Les taux de reconnaissance sont légèrement supérieurs à ceux obtenus précédemment. La voyelle « a » est pour ce test la voyelle la mieux reconnue (88 %), suivie par la voyelle « o » (85 %). La voyelle « e » reste au dernier rang avec un taux de reconnaissance de 68,8 %.

5.6.7 Stratégie des seuils séparés avec modèles binaires

Pour le test suivant, on combine la stratégie des seuils séparés avec les modèles binaires (figure 5.11). On élimine l'importance accordée à l'ordre des impulsions dans les séquences générées et l'inhibition utilisée à la reconnaissance en donnant la valeur « 0 » au paramètre *inhibition*. Autrement, ce test reste identique au test précédent.

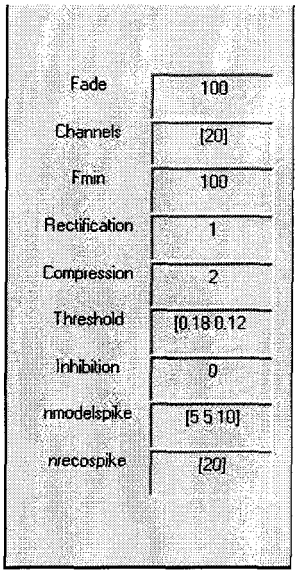


Figure 5.11 – Paramètres du traitement avec *sna* et séparation des seuils avec modèles binaires.

Comme pour le test qui utilise des modèles binaires, le taux de reconnaissance maximal est atteint lorsque le nombre de canaux du banc de filtres est similaire au nombre d'impulsions et le taux de reconnaissance (figure 5.12) augmente en fonction du nombre d'impulsions.

Au niveau du taux de reconnaissance général, on note une légère diminution à comparer au test précédent. Cependant, on note que l'utilisation des modèles à seuils séparés augmente le taux de reconnaissance. Pour ce qui est des voyelles, les tendances restent les mêmes que pour le test précédent.

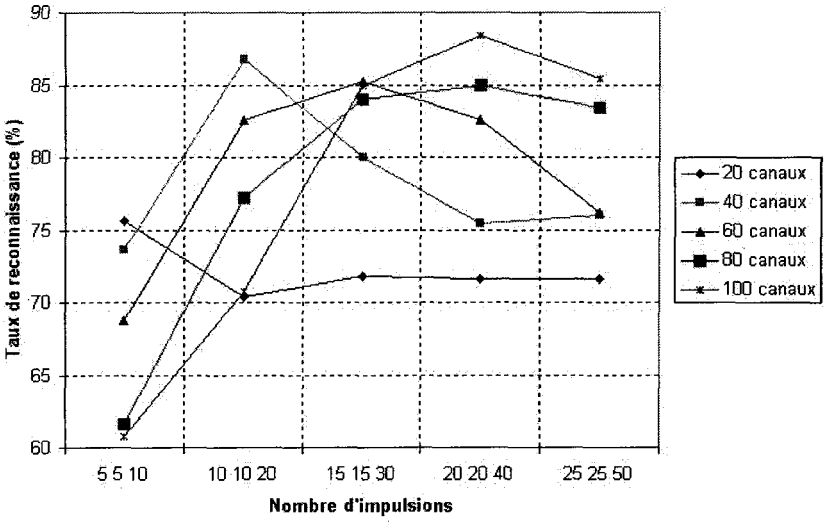


Figure 5.12 – Taux de reconnaissance en fonction du nombre de canaux du banc de filtres et du nombre d'impulsions pour des modèles binaires avec séparation des seuils.

5.7 Influence du nombre d'impulsions pour la reconnaissance

Nous poursuivons l'étude de notre prototype avec des tests qui ont pour objet de vérifier l'influence du nombre d'impulsions utilisées pour la reconnaissance, indépendamment du nombre d'impulsions pour la création des modèles.

Les tests de cette sous-section se font de manière identique à ceux réalisés à la sous-section précédente. Cependant, on fait varier le nombre d'impulsions pour la création des modèles (*nmodelspike*) indépendamment du nombre d'impulsions utilisées à la reconnaissance (*nrecospike*). Pour un banc de filtres de 20, 40, 60 et 80 canaux, on fixe le nombre d'impulsions pour la création des modèles (un test avec 40 impulsions et un autre avec 60 impulsions) et on calcule le taux de reconnaissance globale avec des séquences d'impulsions pour la reconnaissance variant de 10 à 100 par bonds de 10 (figure 5.13).

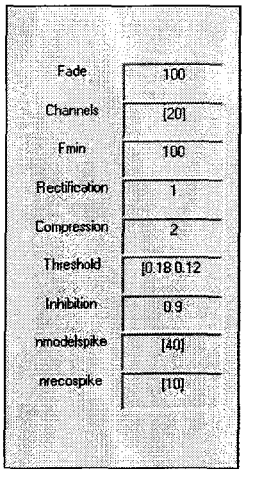


Figure 5.13 – Paramètres du traitement avec *sna*.

5.7.1 Résultats

La figure 5.14 présente les variations du taux de reconnaissance global pour différents nombres de canaux pour le banc de filtre en fonction des variations du nombre d'impulsions utilisées à la reconnaissance. Comme pour la figure 5.6, on remarque que le taux de reconnaissance diminue avec l'augmentation du nombre de canaux du banc de filtres, mais qu'il augmente en fonction du nombre d'impulsions jusqu'à atteindre un plateau. On explique ce plateau par le nombre maximum d'impulsions qu'il est possible de générer par le traitement avec les paramètres sélectionnés.

En augmentant le nombre d'impulsions utilisées pour créer les modèles, on remarque peu de changements au niveau des taux de reconnaissance globaux (figure 5.15). Le taux de reconnaissance pour un nombre élevé de canaux est légèrement supérieur. Sinon, le meilleur taux de reconnaissance est obtenu avec un nombre limité de canaux pour le banc de filtres et un nombre important d'impulsions dans les séquences pour la reconnaissance et ce indépendamment du nombre d'impulsions utilisées à la création des modèles. Ces tests sont évidemment limités, mais on imagine que ce dernier paramètre n'a pas un gros impact au niveau du taux de reconnaissance global.

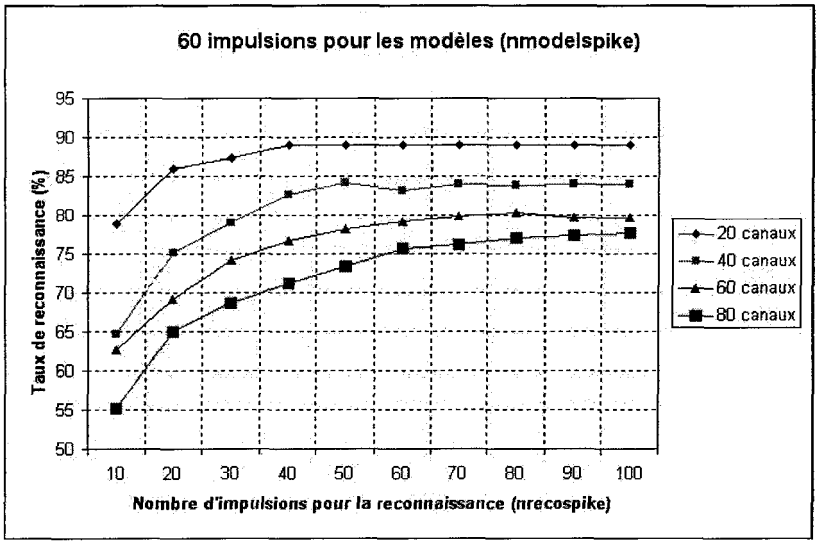


Figure 5.14 – Taux de reconnaissance en fonction du nombre d'impulsions pour la reconnaissance (60 impulsions pour le modèle).

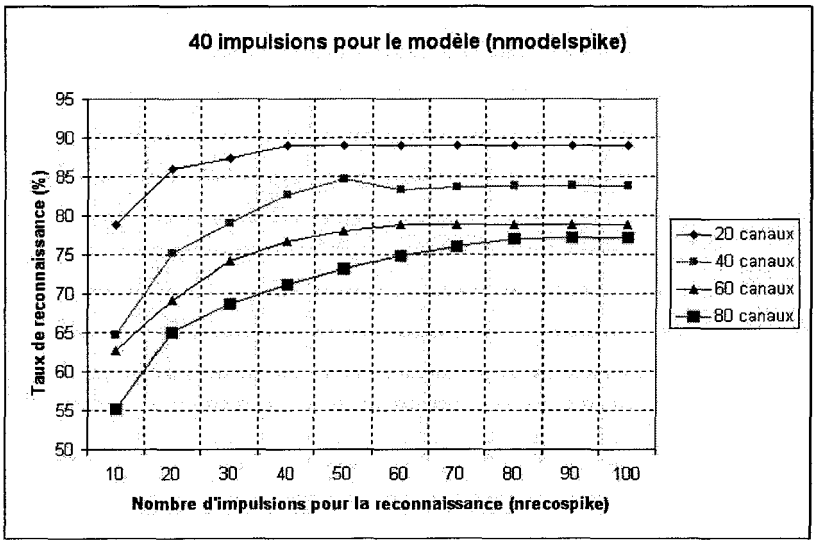


Figure 5.15 – Taux de reconnaissance en fonction du nombre d'impulsions pour la reconnaissance (40 impulsions pour le modèle).

5.7.2 Modèles binaires

Les modèles binaires sont ensuite utilisés pour poursuivre les tests de cette section (figure 5.16). Les 60 premières impulsions générées par le traitement forment le modèle et on continue à faire varier le nombre d'impulsions des séquences utilisées pour la reconnaissance.

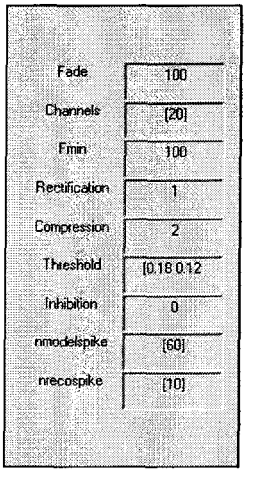


Figure 5.16 – Paramètres du traitement avec *sna* pour des modèles binaires.

Les résultats de ce dernier test (figure 5.17) présentent des similitudes avec ceux de la figure 5.8. En effet, le taux de reconnaissance maximal est généralement atteint lorsque le nombre de canaux du banc de filtres est près du nombre d'impulsions utilisées à la reconnaissance. De plus, il augmente en fonction du nombre d'impulsions utilisées à la reconnaissance pour atteindre un sommet et diminue ensuite pour enfin former un plateau.

5.8 Conclusion

On a démontré dans cette partie que le codage par ordre de rang peut être utilisé pour effectuer la reconnaissance vocale avec un vocabulaire limité.

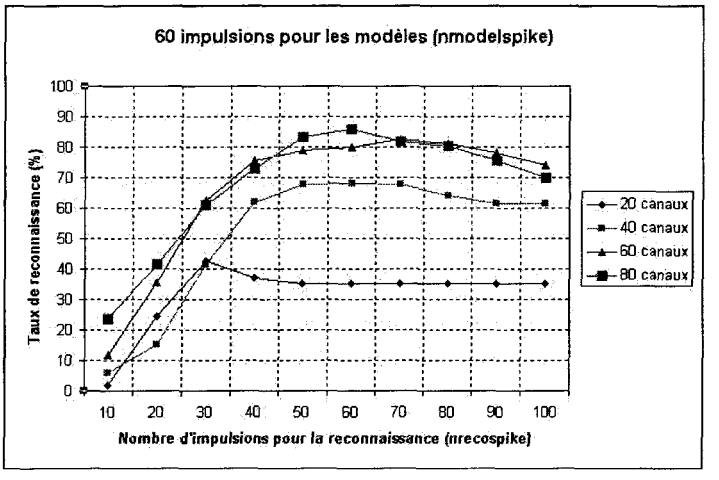


Figure 5.17 – Taux de reconnaissance en fonction du nombre d'impulsions pour la reconnaissance des modèles binaires.

En jumelant le codage par ordre de rang avec un traitement inspiré du système auditif (banc de filtres et canaux à seuils multiples), on obtient des taux de reconnaissance pouvant dépasser 90 %.

Les meilleurs résultats sont obtenus lorsqu'on utilise la compression au niveau des canaux pour distribuer l'information. De plus, le nombre de canaux du banc de filtres doit être limité, mais le nombre d'impulsions utilisées à l'apprentissage et à la création des modèles doit être élevé pour une meilleure reconnaissance. Aussi, la stratégie du regroupement par seuil ajoute l'information sur la valeur du seuil qui a généré l'impulsion et améliore ainsi la reconnaissance.

D'un autre côté, les tests effectués avec les modèles binaires présentent des diminutions au niveau du taux de reconnaissance. En effet, lorsqu'on ne tient pas compte de l'aspect temporel des séquences d'impulsions, le prototype est moins efficace.

Comme l'approche présentée est simple, nous poursuivons en explorant une façon d'implémenter notre prototype plus efficacement.

Enfin, des tests de reconnaissance sont effectués, plus loin dans ce mémoire, avec une approche plus conventionnelle, pour mieux situer la performance de notre prototype.

QUATRIÈME PARTIE

RODIN

CHAPITRE 6

MODULE RODIN

Pour effectuer la reconnaissance vocale en temps réel, il faut que les opérations du système s'exécutent rapidement. Cependant, la reconnaissance vocale est souvent une tâche complexe qui demande un temps de calcul et des ressources importants pour un ordinateur. Plusieurs stratégies et méthodes ont d'ailleurs été développées pour simplifier les opérations que demande un tel système. Par contre, il ne faut pas oublier que l'outil de développement du système de reconnaissance vocale influence directement le temps de traitement.

MATLAB est simple à utiliser, mais ne génère pas les codes les plus rapides. Pour une amélioration éventuelle du temps de traitement de notre système, nous explorons brièvement, le langage de programmation *Rodin*. On développe dans cette partie un module qui effectue une partie du travail de notre prototype (annexe B). Au contraire de *MATLAB*, *Rodin* nous permet, entre autres, la programmation orientée événements et en temps réel.

Avant de passer à l'expérience, nous souhaitons survoler différentes stratégies de programmation et expliquer rapidement le fonctionnement de *Rodin*.

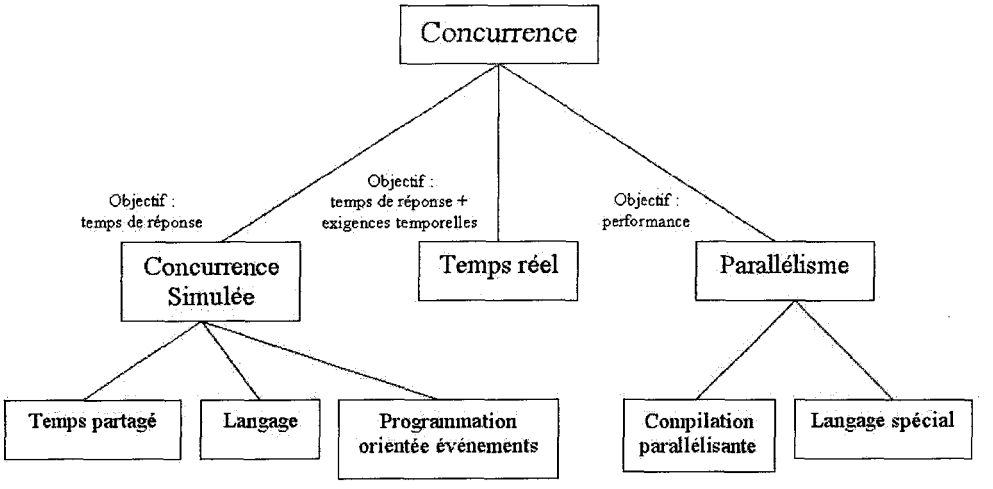


Figure 6.1 – Types de programmation.

6.1 Programmation

Il existe plusieurs stratégies de programmation (figure 6.1). Certaines ont comme objectif de garantir un temps de réponse, de satisfaire des exigences temporelles ou simplement de générer un code plus performant.

6.1.1 Concurrence

Un système concurrent est un système capable d'exécuter plusieurs tâches simultanément du point de vue de l'utilisateur. Le système d'exploitation Windows est un exemple de système concurrent. Ce type de système peut utiliser un algorithme de temps partagé (*fifo*[1], à priorités, etc.) pour pouvoir répondre aux commandes de l'utilisateur plus ou moins rapidement tout en effectuant les tâches critiques pour le bon fonctionnement de l'ordinateur.

[1] *First in, first out*

6.1.2 Temps réel

À tort, il arrive que le terme temps réel soit utilisé pour nommer des systèmes concurrents. De cette manière, on veut illustrer la rapidité du système. De telles erreurs sont compréhensibles, car un système en temps réel est un système concurrent qui possède en plus les caractéristiques suivantes :

- capacité d'effectuer chaque tâche dans un temps précis ;
- haut degré de fiabilité.

On peut alors définir un système en temps réel comme un système qui doit répondre à des stimulus de manière prévisible fonctionnellement et temporellement pour garantir un temps de réponse et assurer la fiabilité. À l'intérieur de tels algorithmes, on doit trouver le moyen de synchroniser l'échelle de temps de la simulation au temps réel.

6.1.3 Parallélisme

Pour une meilleure performance, il arrive qu'on veuille exécuter un programme simultanément sur deux ou sur plusieurs machines. On parle alors de parallélisme. Cependant, le parallélisme n'est pas avantageux dans tout les cas. En effet, il faut que le programme dont on veut partager l'exécution sur un groupe de machines soit décomposable. Le gain en performance sera alors proportionnel à la fraction du programme que l'on peut décomposer.

6.1.4 Programmation orientée événements

Dans la programmation orientée événements, l'exécution d'une fonction ou d'une séquence d'instructions est provoquée par un événement. Ces événements peuvent être produits par des fonctions ou par des ports de communication. En général, ces programmes sont divisés en deux ensembles, les composants et les signaux. Aussi, pour ce type de programmation, on utilise habituellement une boucle de contrôle. À l'intérieur de cette boucle, on doit lire un événement, le traiter et passer au pas suivant. Ce qui

est intéressant avec cette programmation, c'est qu'en général, une diminution de la grandeur du pas de temps (le temps entre deux pas de la boucle) a peu d'influence sur le temps de calcul, puisque le nombre d'événements reste constant, tout en nous donnant une meilleure précision.

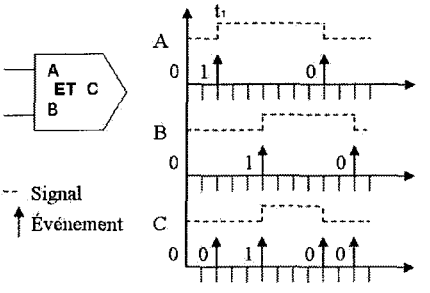


Figure 6.2 – Exemple de programmation orientée événements.

La figure 6.2 illustre un composant ET qui reçoit des signaux logiques par les ports A et B. Pour cet exemple, le temps est discret, séparé en intervalles de temps constants. Lorsqu'un événement (un nouveau signal logique arrive sur un ou les deux ports d'entrées) a lieu (par exemple la valeur du port A qui devient 1 au temps t_1), le composant effectue le calcul (dans ce cas, un ET logique avec la valeur 1 pour le port A et la valeur 0 pour le port B) et la sortie est modifiée, produisant ainsi un nouvel événement (la valeur 0 pour le port de sortie C).

6.2 *Rodin*

Rodin est un langage de programmation parallèle qui supporte le modèle de la programmation orientée événements. Pour *Rodin*, au contraire d'une exécution séquentielle, les instructions pendant un même intervalle de temps se font simultanément. Par exemple, pour échanger deux valeurs ($A \rightarrow B$ et $B \rightarrow A$) dans un programme séquentiel, nous devons utiliser une variable temporaire supplémentaire. Dans le cas d'un programme *Rodin*, la valeur est échangée pour l'intervalle suivant (figure 6.3).

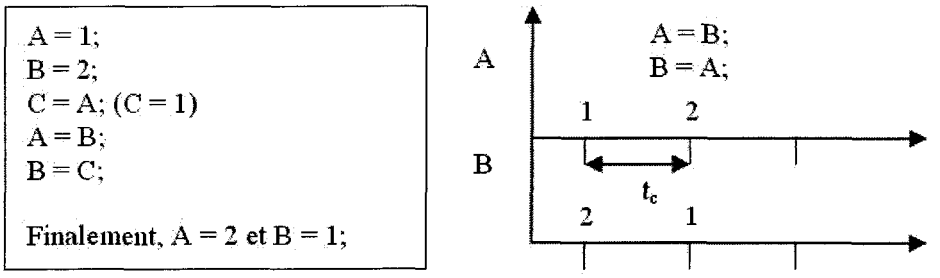


Figure 6.3 – Exemple de programmation séquentielle (à gauche) et parallèle (à droite).

Les éléments de base d'un circuit *Rodin* sont des composants interreliés. Chaque composant *Rodin* peut posséder n ports d'entrée et n ports de sortie. De plus, une priorité est associée à chaque port. Dans le cas d'événements simultanés, l'opération ayant la priorité la plus basse est sélectionnée en premier. Les connexions qui relient les composants peuvent être vues comme un lien de communication par variable partagée.

Une approche par composants offre l'avantage de pouvoir facilement s'implémenter comme système parallèle. On sépare les différents composants sur les machines et seulement les signaux qui les relient doivent être propagés. On limite ainsi les communications pour obtenir un système performant.

Dans *Rodin*, grâce à la programmation orientée événements, seul les composants qui reçoivent des événements doivent mettre à jour leurs sorties. Lorsque les fonctions ont été évaluées, les événements de sortie qui ont été générés se retrouvent dans une liste. Ensuite, ces événements sont propagés vers les entrées. Chaque événement esclave possède une priorité et la boucle, lorsqu'elle prend un événement et le traite, commence par la plus basse priorité (le plus haut numéro) et poursuit jusqu'à la plus haute (le plus bas numéro). De ce fait, si deux événements contradictoires (avec priorités différentes) se produisent simultanément, la plus haute priorité va écraser la plus basse. On comprend alors que ces événements doivent être conservés par priorité. Dans *Rodin*, c'est en utilisant une liste de priorités qu'on le fait.

Le cycle d'exécution d'un programme *Rodin* consiste en quatre opérations :

- lecture des événements des ports externes et copie de ces événements dans la liste des événements courants ;
- appel du pilote pour transmettre la valeur pour chaque événement qui arrive sur un port de sortie externe au programme ;
- exécution de la fonction pour chaque événement ;
- placer dans une liste d'événements de sortie les événements provenant de l'exécution de la fonction de sortie.

La durée des cycles est définie dans les paramètres de la simulation. Cependant, en ce moment, seulement les intervalles constants sont supportés par *Rodin.*

Pour la synchronisation en temps réel, à chaque fin de pas de la boucle, on vérifie si l'on est rendu au temps du pas suivant. Dans le cas contraire, on doit attendre le reste du temps. Pour la synchronisation de l'échelle du temps, il est aussi possible d'utiliser des chronomètres à l'intérieur de notre composant. Lorsque le temps du chronomètre s'est écoulé, sa fonction est exécutée et on peut redémarrer le chronomètre.

Dans un autre mode, on n'effectue pas la synchronisation entre l'échelle de temps *Rodin* et l'échelle de temps réel. Comme pour le mode en temps réel, on peut contrôler le temps avec un intervalle défini : on traite les événements présents pour ensuite passer au pas suivant ; on vérifie alors la présence de nouveaux événements et ainsi de suite. Ou bien, on peut aussi utiliser les minuteurs des composants. Cette fois, les événements des composants sont déclenchés par les minuteurs lors de leur retour à zéro. On peut alors trier en ordre croissant les minuteurs (ceux qui se déclenchent plus tôt en premier). On traite le ou les événements au temps du premier minuteur et si aucun nouvel événement n'est généré entre ce temps et celui du prochain minuteur, on saute au déclenchement du prochain minuteur. En temps réel, il faut synchroniser une attente si les minuteurs ne déclenchent pas en des intervalles successifs pour conserver un temps constant.

6.3 Expérience

Pour l'expérience, nous utilisons *Rodin* (*RIDE 0 4 2*) pour développer un module qui sert à détecter les seuils et générer la séquence d'impulsions. Aussi, ce test est effectué sur un *Pentium 4 Mobile 2 GHz.*

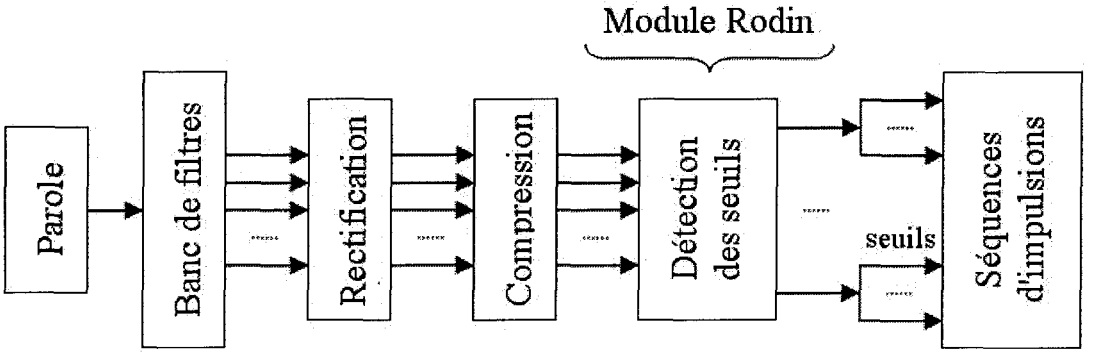


Figure 6.4 – Traitement et module *Rodin.*

D'abord, on effectue le traitement (figure 6.4) sur le signal avec notre prototype développé en *MATLAB* (filtrage, rectification et compression). Ces signaux proviennent d'une petite base de données [11] et ils sont enregistrées en format *wav* à 16 kHz. Ensuite, on crée un composant (le détecteur de seuil représenté à la figure 6.5) pour chacun des canaux du banc de filtres. Grâce au fichier d'événements d'entrée (*fichier.evi*) de *Rodin*, les sorties du banc de filtres deviennent les événements d'entrées des composants.

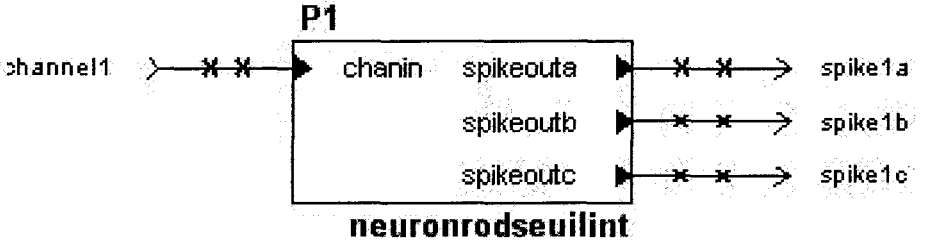


Figure 6.5 – Composant *Rodin* pour la détection des seuils.

TABLEAU 6.1 – Ports d'entrées des composants *Rodin*

```
; ------------------- Initialization -------------------
; Initial value of each port
; Format :  input_number  initial_value

[initials_values]
1 0
2 0
3 0
4 0
5 0
6 0
7 0
8 0
9 0
10 0
11 0
12 0
13 0
14 0
15 0
16 0
17 0
18 0
19 0
20 0
```

Par exemple, pour un traitement avec un banc de filtres de 20 canaux, on crée 20 composants avec chacun un port d'entrée. Ces ports d'entrées sont numérotés et ils possèdent une valeur initiale (tableau 6.1). On peut par la suite donner la liste des événements qui est constituée du signal échantillonné et traité. De cette façon, un événement active le composant à chaque pas de la simulation. De plus, la résolution de la simulation en *Rodin* peut être spécifiée. Il est à noter que pour l'opération simple qu'effectuent nos composants, les entrées sont codées en entiers (on multiplie le signal traité par 10 000 et on arrondit vers le bas). Le tableau 6.2 présente un exemple de fichier d'événements. La première colonne affiche le type de donnée (*e* pour entier). La deuxième colonne contient le numéro du port où il y a un événement. Les dernières colonnes contiennent respectivement le moment de l'événement ainsi que sa valeur.

Lorsque le composant reçoit un événement, il vérifie si la valeur d'entrée (*chanin*) est supérieure ou égale à la valeur des seuils (on utilise trois seuils dans le composant : 600, 1200 et 1800). Si le seuil n'a pas déjà été franchi au court de la simulation, un événement (« 1 ») est produit à la sortie correspondante au seuil (*spikeouta* pour 600, *spikeoutb* pour 1200 et *spikeoutc* pour 1800).

TABLEAU 6.2 – Événement pour les ports d'entrées des composants *Rodin*

```
; ------------------- Event Table --------------------
; Format :   type   input_number  time(ms)   value

[events]
e 1 1 0
e 1 2 0
e 1 3 0
e 1 4 0
...
e 2 1 0
e 2 2 0
e 2 3 0
e 2 4 0
...
e 20 1 0
e 20 2 0
e 20 3 0
e 20 4 0
...
```

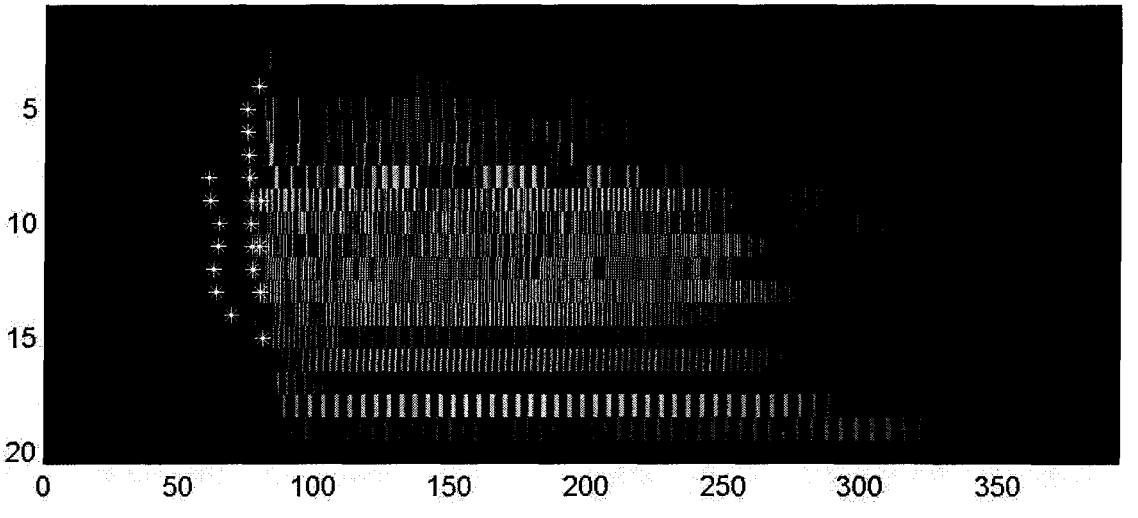


Figure 6.6 – Traitement du signal *locfjlaa5.wav* par le prototype en *MATLAB*

Nous avons testé le module *Rodin* avec le signal *locfjlaa5.wav* qui a été traité (figure 6.6). Pour ce test, nous avons 60 sorties (20 composants avec 3 sorties chacun) et le tableau 6.3 présente les 20 premiers événements à être générés. La première colonne indique que la valeur de l'événement est un entier. Les colonnes suivantes affichent le numéro du port de sortie qui génère un événement (la valeur 1) ainsi que le moment de cet événement. En prenant le numéro du composant qui a produit ces événements, on obtient la séquence des canaux qui ont « généré les impulsions » : 8, 9, 12, 13, 11, 10, 14, 5, 6, 7, 8 (2^e seuil), 9 (2^e seuil), 10 (2^e seuil), 11 (2^e seuil), 12 (2^e seuil), 4, 11 (3^e seuil), 13 (2^e seuil), 9 (3^e seuil) et 15. Cette séquence correspond à celle que l'on obtiendrait en utilisant le prototype développé en *MATLAB*.

TABLEAU 6.3 – Événements générés par les ports de sorties des composants *Rodin*

```
[events]
e 22 972 1
e 25 976 1
e 34 1003 1
e 37 1014 1
e 31 1024 1
e 28 1032 1
e 40 1108 1
e 13 1197 1
e 16 1198 1
e 19 1200 1
e 23 1211 1
e 26 1215 1
e 29 1220 1
e 32 1226 1
e 35 1233 1
e 10 1266 1
e 33 1269 1
e 38 1278 1
e 27 1279 1
e 43 1292 1
```

6.4 Conclusion

Une fois familiarisé avec les particularités de *Rodin*, il est non seulement rapide de développer des composants, mais il est aussi facile de les modifier. En effet, l'approche

par composants adoptée par *Rodin* nous permet de changer par exemple le type de « neurone » utilisé dans notre module sans avoir à le refaire. En respectant les entrées et les sorties, il nous est ainsi possible de passer des détecteurs de seuils à ceux par sommes cumulées et même à des intégrateurs à décharges. Les structures hiérarchiques nous permettent aussi de créer des réseaux de neurones plus importants de façon simple.

D'ailleurs, notre approche se prête bien à la programmation orientée événements. En effet, simuler des détecteurs de seuils et générer une séquence d'événements comme impulsion est facilement réalisé en *Rodin*. Ce langage de programmation présente certainement un grand potentiel pour l'implémentation de notre approche.

D'un autre côté, les neurones oscillatoires, utilisés au chapitre 4, vont demander plus d'efforts à l'implémentation par événements. D'abord, on présente une image statique en entrée au réseau. Cette image devra être présentée plusieurs fois pour générer les événements pour entraîner le calcul des oscillations. L'utilisation des chronomètres internes pourrait aussi être envisageable pour résoudre ce problème. Enfin, les équations caractéristiques des oscillations vont demander des calculs importants.

CINQUIÈME PARTIE

TESTS DE RÉFÉRENCE

CHAPITRE 7

HMM ET COEFFICIENTS CEPSTRAUX

Pour avoir une idée plus précise de l'efficacité du prototype qui a été développé, nous souhaitons comparer nos résultats avec ceux obtenus grâce à un système de reconnaissance plus conventionnel. En nous inspirant d'un projet sur la reconnaissance de la parole pour la commande vocale des équations mathématiques [11], nous utilisons des modèles de Markov cachés et des coefficients cepstraux (voir la section 2.2 à la page 13) pour effectuer la reconnaissance vocale.

Les premiers tests sont effectués avec l'ensemble des prononciations des voyelles a, e, i, o et u des cinq locuteurs : *elo*, *mpe*, *rlo*, *slo* et *str* et des cinq locutrices : *fpe*, *hgi*, *jla*, *lau* et *mtr*. Par la suite, on teste la reconnaissance des chiffres de façon similaire avec neuf locuteurs (la locutrice *jla* n'est pas disponible pour les chiffres).

7.1 Implémentation

Chaque modèle consiste en une chaîne de Markov cachée gaussienne (implémentée dans *MATLAB* [3]) de cinq états dont les transitions de retour sur un même état, vers l'état directement à droite ou vers le second état à droite sont permises (figure 7.1). À chaque passage d'un état à un autre, un vecteur de douze coefficients cepstraux (par fenêtre de 512 échantillons avec recouvrement de 256 échantillons) est calculé avec les fonctions du *VOICEBOX* de *MATLAB*. Le nombre de vecteurs que nous obtenons

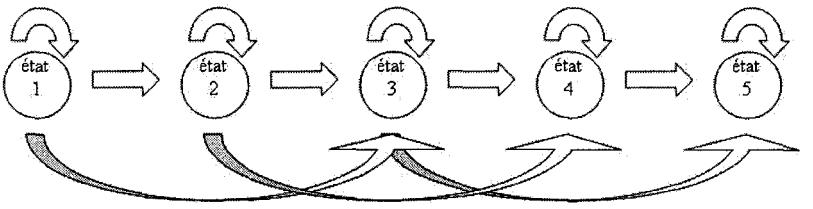


Figure 7.1 – Modèle de chaîne de Markov pour les voyelles.

varie de trente à cent, dépendamment de la longueur du mot. Par la suite, on utilise ces vecteurs pour entraîner les modèles et pour réaliser la reconnaissance.

7.2 Modèles généraux

Dans un premier temps, nous construisons un modèle pour les locuteurs et un modèle pour les locutrices pour chaque voyelle. Ces modèles sont créés en utilisant les dix prononciations des cinq locuteurs ou des cinq locutrices. On extrait tour à tour les coefficients cepstraux des prononciations pour effectuer l'entraînement de la chaîne de Markov. Pour ne pas accorder plus d'importance à un locuteur, un soin particulier est pris dans l'ordre de sélection de la prononciation à traiter : on effectue la rotation du locuteur après chaque traitement (on traite la première prononciation de chaque locuteur dans la liste des locuteurs pour l'apprentissage, on passe à la seconde et ainsi de suite jusqu'à la dixième).

Ensuite, on trouve pour l'ensemble des prononciations (tous les locuteurs et toutes les voyelles) la vraisemblance avec chaque modèle. Enfin, le taux de reconnaissance est obtenu pour chaque locuteur pour chaque voyelle en effectuant le rapport du nombre de prononciations dont la plus grande similitude appartient à un modèle de la voyelle prononcée sur le nombre total de prononciations de la voyelle par ce locuteur. Il est à noter qu'on ne fait pas la distinction entre un modèle pour les locuteurs et celui pour les locutrices ; le modèle est accepté s'il est bien celui de la voyelle prononcée.

7.2.1 Discussion

La reconnaissance obtenue pour ce test est parfaite. En effet, chaque prononciation pour tous les locuteurs est bien reconnue. De plus, les prononciations sont correctement classées selon le sexe du locuteur. Comme la reconnaissance s'effectue sur les prononciations qui servent à la création des modèles, il est normal d'observer un taux de reconnaissance aussi élevé. En limitant le nombre de locuteurs et de locutrices utilisés lors de l'apprentissage, une diminution du taux de reconnaissance devrait être observée.

7.3 Modèles à deux locuteurs

Dans ce second test, l'entraînement du modèle pour les locuteurs, ainsi que celui pour les locutrices, s'effectuent avec toutes les prononciations de seulement deux locuteurs (*rlo* et *slo* pour les locuteurs, *hgi* et *mtr* pour les locutrices). Néanmoins, le type de modèle utilisé, le traitement effectué pour obtenir les coefficients cepstraux et le calcul des similitudes restent identiques à ceux du test précédent.

7.3.1 Discussion

Toutes les prononciations des voyelles pour tous les locuteurs sont, une fois de plus, correctement classées. Cependant, on observe à quelques reprises, pour les locuteurs exclus de l'apprentissage, que le sexe du locuteur dont les prononciations servent à entraîner le modèle, qui possède la plus grande similitude avec la prononciation à classer ne correspond pas toujours au sexe du locuteur qui a prononcée cette dernière. La nature des sons à reconnaître (voyelles isolées), le vocabulaire limité ainsi que la qualité des enregistrements (environnement peu bruité) sont fort probablement les raisons d'une aussi bonne reconnaissance.

Jusqu'ici, les taux de reconnaissance obtenus en utilisant les *HMM*s dépassent les meilleurs résultats obtenus avec notre prototype (présenté au chapitre 5). Cependant, on n'utilisait qu'une prononciation pour créer nos modèles.

7.4 Modèles avec une seule prononciation

Comme pour la section 5.6, nous n'utilisons pour ce test qu'une prononciation pour créer chaque modèle (celles énumérées au tableau 5.13, à la page 78). De cette façon, on crée les dix modèles (un pour les locuteurs et un pour les locutrices pour les cinq voyelles). Sinon, ce test reste identique aux autres tests de cette partie.

7.4.1 Discussion

Il est logique de remarquer une diminution au niveau des taux de reconnaissance lorsque l'on n'utilise qu'une seule prononciation pour entraîner les modèles (tableau 7.1). Cette fois, la reconnaissance est imparfaite, mais le taux de reconnaissance global atteint 90 %. Comme pour le prototype, ce sont les voyelles « e » (75 %) et « a » (86 %) qui montrent les plus bas taux de reconnaissance pour ce test.

Pour cette expérience, le taux de reconnaissance globale du prototype (avec les meilleurs paramètres) est semblable à celui qu'on obtient avec les *HMM*s. Par contre, il ne faut pas oublier que les prononciations qui servent de modèles ont été sélectionnées en fonction du prototype. Il est fort possible qu'on obtienne de meilleurs taux de reconnaissance globale en sélectionnant d'autres prononciations comme modèles.

TABLEAU 7.1 – Résultat de la reconnaissance pour la création des modèles avec une seule prononciation des voyelles

Modèles												%
Locutrices	a	e	i	o	u	Locuteurs	a	e	i	o	u	90
fpea	10					eloa	10					86
hgia	5	4		1		mpea	10					
jlaa	10					rloa	10					
laua	1	9				sloa	10					
mtra	10					stra	10					
fpee		10				eloe		8	2			75
hgie		9		1		mpee	1	6	2	1		
jlae		10				rloe		1	1	7	1	
laue		10				sloe	4	1	5			
mtre		10				stre		10				
fpei			10			eloi			7	2	1	91
hgii			10			mpei			9		1	
jlai			10			rloi			7	3		
laui			10			sloi			10			
mtri			10			stri			8	2		
fpeo				10		eloo				10		100
hgio				10		mpeo				10		
jlao				10		rloo				10		
lauo				10		sloo				10		
mtro				10		stro				10		
fpeu					10	elou					10	98
hgiu					10	mpeu			1		9	
jlau					10	rlou					10	
lauu					10	slou				1	9	
mtru					10	stru					10	

7.5 Reconnaissance des chiffres

Pour compléter cette partie, nous avons effectué la reconnaissance vocale sur les chiffres de zéro à neuf prononcé par neuf locuteurs (quatre locutrices et cinq locuteurs) les *HMM*s et avec notre prototype.

La sélection des locuteurs se fait en fonction du taux global de reconnaissance des prononciations comme celle effectués pour les voyelles aux tests de la sous-section 5.6.1 avec notre prototype.

TABLEAU 7.2 – Liste des modèles sélectionnés pour les chiffres

Chiffre	Modèle	Taux de reconnaissance
1	locfmtr1c	93,1
	locmslo1e	93,3
2	locffpe2f	91,6
	locmrlo2f	91,6
3	locflau3h	90,8
	locmslo3c	86,9
4	locfmtr4j	92,6
	locmslo4h	94,6
5	locflau5c	89,4
	locmstr5h	84,5
6	locffpe6i	90,9
	locmmpe6f	84,6
7	locflau7f	89
	locmstr7b	85,4
8	locfmtr8h	95,3
	locmelo8h	96
9	locffpe9i	95,6
	locmelo9e	94,6
0	locffpe0a	93,6
	locmmpe0b	93,1

7.5.1 *HMM*

Comme à la section 7.4, on utilise une seule prononciation pour créer les vingt modèles (tableau 7.2) et l'apprentissage ainsi que la reconnaissance s'effectuent de la même manière que les tests précédents.

TABLEAU 7.3 – Résultat de la reconnaissance avec les *HMM*s pour les chiffres

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 51,9
fpe1					10						0
hgi1										10	0
lau1					10						0
mtr1	8									2	80
elo1									10		0
mpe1									8	2	0
rlo1									10		0
slo1	7			1						2	70
str1					10						0
											16,7
fpe2		10									100
hgi2		8								2	80
lau2		9		1							90
mtr2		9								1	90
elo2								1	9		0
mpe2										10	0
rlo2		10									100
slo2		9								1	90
str2					10						0
											61,1
fpe3			3							7	30
hgi3			5							5	50
lau3			10								100
mtr3			2							8	20
elo3			3							7	30
mpe3										10	0
rlo3			6							4	60
slo3			10								100
str3			3							7	30
											46,7

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 51,9
fpe4					10						0
hgi4				10							100
lau4					10						0
mtr4				10							100
elo4					2				8		0
mpe4				3						7	30
rlo4									2	8	0
slo4				10							100
str4					10						0
											36,7
fpe5					10						100
hgi5					7	1				2	70
lau5					10						100
mtr5					10						100
elo5					10						100
mpe5		2			5					3	50
rlo5					9				1		90
slo5					10						100
str5					10						100
											90
fpe6						10					100
hgi6						9				1	90
lau6						10					100
mtr6						10					100
elo6						10					100
mpe6						10					100
rlo6						10					100
slo6					9					1	0
str6					4	3		1		2	30
											98,6

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 51,9
fpe7					9	1				0	33,3
hgi7					2		5			50	
lau7							10			100	
mtr7					9		1			10	
elo7					8		2			20	
mpe7					1	7	1			10	
rlo7					7		1			10	
slo7					4					0	
str7							10			100	
fpe8		1			1	8				0	5,6
hgi8		10								0	
lau8		6			4					0	
mtr8		5						5		50	
elo8									10	0	
mpe8		1							6	0	
rlo8									10	0	
slo8		7							3	0	
str8		1							9	0	
fpe9						3			10	100	48,9
hgi9		2							5	0	
lau9					9				1	10	
mtr9				3					4	40	
elo9									10	100	
mpe9									9	90	
rlo9									10	100	
slo9		2		7						0	
str9					10					0	

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 51,9	
fpe0										10	100	100
hgi0										10	100	
lau0										10	100	
mtr0										10	100	
elo0										10	100	
mpe0										10	100	
rlo0										10	100	
slo0										10	100	
str0										10	100	

Le taux de reconnaissance global pour les chiffres est inférieur à 55 % et la majorité des chiffres sont mal reconnus. Les modèles du zéro et du cinq sont responsables d'une grande partie des confusions. D'ailleurs, ces modèles induisent souvent en erreurs la reconnaissance des chiffres un (16,7 %) et sept (33,3 %). D'un autre côté, le chiffre huit (5,6 %) passe plus souvent pour les chiffres neuf et deux. Par contre, le chiffre zéro est parfaitement reconnu et le chiffre cinq est très bien reconnu (90 %).

7.5.2 Prototype

Pour la reconnaissance des chiffres, on utilise 20 canaux pour notre prototype et 40 impulsions pour la création des modèles et la reconnaissance. Même si elle a généralement un effet positif sur le taux de reconnaissance, la stratégie de grouper les seuils n'est pas utilisée. Cependant, l'inhibition est conservée. Ces paramètres (figure 7.2) sont sélectionnés puisqu'ils ont fourni d'excellents résultats lors de la reconnaissance des voyelles.

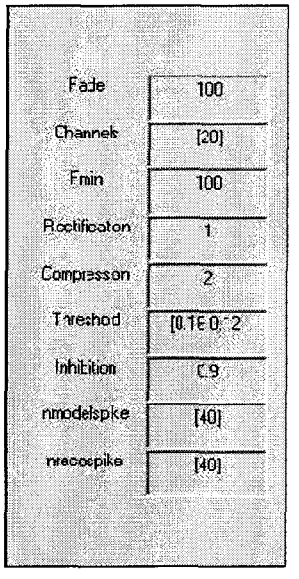


Figure 7.2 – Paramètres du traitement avec *sna* pour la reconnaissance des chiffres.

Comme pour le test précédent, on utilise deux modèles pour chaque chiffre : un pour les locuteurs et un autre pour les locutrices. Ces modèles pour la reconnaissance sont créés à partir d'une seule prononciation du chiffre (tableau 7.2).

La reconnaissance s'effectue sur les dix prononciations des chiffres par les neuf locuteurs et le taux de reconnaissance est obtenu en calculant le rapport des prononciations qui sont bien reconnues sur le nombre total de prononciations. Il n'est pas nécessaire que le modèle soit prononcé par un locuteur de même sexe que celui de la prononciation à reconnaître pour être accepté comme reconnu.

Les résultats de la reconnaissance avec notre prototype, pour ce test, sont supérieurs à ceux obtenus avec les *HMMs*. En effet, on note une hausse de plus de 10 % au taux de reconnaissance global (64,8 %). Cependant, on observe de grandes confusions pour les chiffres sept (13,3 %) et cinq (46,7 %) avec le chiffre six. On retrouve aussi une confusion importante entre les chiffres trois (64,4 %) et quatre (75,6 %). D'un autre côté, il ne semble pas y avoir de modèle problématique, comme les modèles des chiffres cinq et zéro de l'approche précédente.

7.5.3 Discussion

Le taux de reconnaissance est en général meilleur avec l'approche conventionnelle (*HMM* et coefficients cepstraux). Par contre, lorsqu'on limite la création des modèles à une seule prononciation, le prototype qui utilise le codage par ordre de rang devient aussi, sinon plus, efficace. Ce dernier résultat n'est pas surprenant puisque les *HMMs* demandent, en général, un nombre de données d'apprentissage plus important pour fournir des résultats intéressants. De plus, il faut noter que les modèles étaient choisis en fonction du prototype. Néanmoins, la sélection de ces modèles pour le prototype ne s'est pas faite de façon optimale.

TABLEAU 7.4 – Résultat de la reconnaissance avec notre prototype pour les chiffres

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 64,9
fpe1	8		2								80
hgi1	10										100
lau1	8	1								1	80
mtr1	9		1								90
elo1	10										100
mpe1	10										100
rlo1	10										100
slo1	9		1								90
str1	10										100
											93,3
fpe2		7							3		70
hgi2		8	1					1			80
lau2		10									100
mtr2		7		1				1	1		70
elo2		8	1						1		80
mpe2		10									100
rlo2		9							1		90
slo2		10									100
str2								1	7	2	0
											76,7
fpe3			8	2							80
hgi3			8	2							80
lau3	1		7	2							70
mtr3			4	6							40
elo3			6	4							60
mpe3	2		7							1	70
rlo3	2		7	1							70
slo3			9	1							90
str3	5		2					1	1	1	20
											64,4

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 64,9
fpe4			8	2							20
hgi4				10							100
lau4			2	8							80
mtr4				10							100
elo4			3	7							70
mpe4				10							100
rlo4			3	7							70
slo4			1	9							90
str4			5	5							50
											75,6
fpe5	3				6					1	60
hgi5					3	2	2			3	30
lau5	1				8	1					80
mtr5					8	2					80
elo5			1	1	1	7					10
mpe5	6			1						3	0
rlo5					6	2	2				60
slo5					2	7	1				20
str5	1				8		1				80
											46,7
fpe6						9	1				90
hgi6						7	3				70
lau6						5	5				50
mtr6						10					100
elo6						10					100
mpe6						1	2	2	1	4	10
rlo6					1	8	1				80
slo6						10					100
str6						8	1			1	80
											75,6

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 64,9
fpe7					2	6	2				20
hgi7					4	2	3	1			30
lau7		1			1	3	3		1	1	30
mtr7					1	9					0
elo7				2		8					0
mpe7										10	0
rlo7			1			9					0
slo7						10					0
str7					3	2	4			1	40
fpe8								2	8		20
hgi8								10			100
lau8								10			100
mtr8								10			100
elo8								10			100
mpe8								4	1	5	40
rlo8								9	1		90
slo8		1						9			90
str8								4	6		40
fpe9									10		100
hgi9								8	2		20
lau9		1	3						6		60
mtr9									10		100
elo9									9	1	90
mpe9									2	8	20
rlo9									1	9	10
slo9								1	4	5	40
str9									10		100

Locuteurs	Modèles 1	2	3	4	5	6	7	8	9	0	% 64,9	
fpe0							1		2	7	70	
hgi0								1	1	8	80	
lau0						1		1	1	7	70	
mtr0						2				8	80	
elo0						7		1	2		0	67,8
mpe0										10	100	
rlo0						3	1			6	60	
slo0						2				8	80	
str0									3	7	70	

D'un autre côté, le prototype réalisé n'est qu'au début de son développement alors que les chaînes de Markov sont déjà bien connues. Cette nouvelle approche présente donc un potentiel fort intéressant pour de futures recherches. D'abord, on n'a que sur-

volé l'influence des différents paramètres sur le taux de reconnaissance. Ensuite, les tests ont été limités à un petit nombre de modèles et comme l'apprentissage et la reconnaissance est simple, il serait facile d'améliorer la performance du prototype en augmentant le nombre de modèles. Enfin, suite aux développements des cartes auditives, il serait intéressant d'étendre notre stratégie de codage sur ces types de représentations.

SIXIÈME PARTIE

DISCUSSION

CHAPITRE 8

BILAN

De plus en plus de recherches s'effectuent pour développer des systèmes de reconnaissance en utilisant des réseaux de neurones. En grande partie, on tente de résoudre de cette manière les problèmes que l'on rencontre avec les approches statistiques plus conventionnelles (reconnaissance difficile dans des conditions bruités, quantité de données nécessaire pour l'apprentissage, ...).

Dans ce mémoire, on a exploré deux façons différentes d'effectuer la reconnaissance vocale à l'aide de neurones à décharges. La base de données utilisée pour les tests provient d'un projet réalisé antérieurement [11]. Elle contient notamment les prononciations des chiffres et des voyelles en français du Québec de dix locuteurs (cinq hommes et cinq femmes).

Dans une première partie, un réseau composé d'oscillateurs a été utilisé avec un cepstrogramme de quelques prononciations des chiffres. Malheureusement, il semble que ce type d'entrée ne soit pas compatible avec l'architecture présentée. De plus, l'implémentation des neurones oscillatoires dans un langage de programmation orienté événements, comme *Rodin*, risque à première vue d'être difficile.

D'un autre côté, le prototype qui a été conçu en s'inspirant du fonctionnement du système auditif possède un potentiel intéressant. En effet, à l'aide du codage par ordre de rang et en modélisant le fonctionnement de l'oreille interne par un banc de filtre et

des détecteurs de seuils, nous avons effectué efficacement la reconnaissance vocale avec un vocabulaire limité (chiffres ou voyelles). Les résultats du prototype se comparent à ceux obtenus avec des chaînes de Markov et des coefficients cepstraux lorsqu'on limite le nombre de prononciations à l'apprentissage. De plus, l'approche développée est simple et peut facilement s'implémenter dans un langage de programmation orienté événements. Cet avantage va éventuellement nous permettre une exécution plus efficace du système de reconnaissance vocale.

CHAPITRE 9

CONCLUSION

D'après les tests présentés dans ce mémoire, les efforts futurs devraient se concentrer sur le prototype qui utilise le codage par ordre de rang. En effet, ce prototype présente déjà un grand potentiel pour la reconnaissance vocale.

Quelques tests ont été effectués à travers ce document, mais il reste encore de nombreux facteurs qui n'ont pas été considérés.

Peu d'attention a été accordée à la création des modèles utilisés à l'apprentissage. Il est évident que le choix des données utilisées à l'apprentissage ainsi que la manière de l'exécuter vont avoir un impact majeur sur le taux de reconnaissance.

D'un autre côté, peu de temps a été consacré à déterminer les valeurs efficaces des seuils pour les canaux. De plus, le fonctionnement actuel de notre prototype n'accorde de l'importance qu'au début des mots. Une stratégie telle une période réfractaire suite à une décharge ou à la séparation du signal en tranche de temps devra être adoptée pour traiter tout le mot.

Il ne faut pas oublier que nous utilisons, dans le prototype, des détecteurs de seuils pour générer des impulsions. Ce modèle a d'abord été choisi pour sa simplicité. Maintenant que nous avons une meilleure idée du fonctionnement de notre prototype, il n'est plus nécessaire de se limiter à ce type de neurone. Il est fort possible que des

neurones de types intégration et décharges puissent générer des séquences d'impulsions compatibles avec notre approche. Avec un tel modèle, les canaux qui ont une activité soutenue devraient être représentés.

Enfin, suite aux développements des représentations par cartes auditives [18, 19], il pourrait être intéressant d'y étendre la stratégie présentée par notre approche.

SEPTIÈME PARTIE

ANNEXES

ANNEXE A

PREMIÈRE EXPLORATION

A.1 Code par ordre de rang

A.1.1 Objectif

Le premier test que nous avons effectué avait pour but de rapidement vérifier s'il était possible d'extraire, d'un nombre limité de mots, une caractéristique propre à chaque mot différent en utilisant le codage par ordre de rang, voir la sous-section 2.3.3 à la page 18.

A.1.2 Expérience

Pour ce test, nous avons choisi d'utiliser le logiciel *MATLAB*. Ce logiciel ne produit pas de codes rapides et optimisés. Cependant, notre expérience et sa simplicité nous permettent de rapidement créer des prototypes. Ces tests ont été effectués sur un *Pentium 4 Mobile 2 GHz* avec *MATLAB R12*.

Pour ce test simple, nous utilisons les prononciations des chiffres de un à quatre de cinq locuteurs (3 hommes et 2 femmes)[1]. Ces prononciations (figure A.1) proviennent d'une petite base de données [11] et sont enregistrées en format *wav* à 16 kHz.

En utilisant le *auditory toolbox* [23], les signaux sont filtrées par un banc de filtre de type *gammatone* de 24 canaux (100 Hz à 8 kHz). La figure A.2 présente le résultat de cette opération sur la première prononciation du chiffre 1 de la locutrice *mtr*.

Ensuite, on calcule la somme cumulée de la valeur absolue de la sortie pour chaque canal (figure A.3) et si elle dépasse un certain seuil (« un » pour cette expérience), une impulsion est générée. Pour ce test, on se limite à une seule impulsion par canal, donc un maximum de 24 impulsions par simulation.

[1] locuteurs *mpe*, *rlo*, *slo*, *hgi* et *mtr*

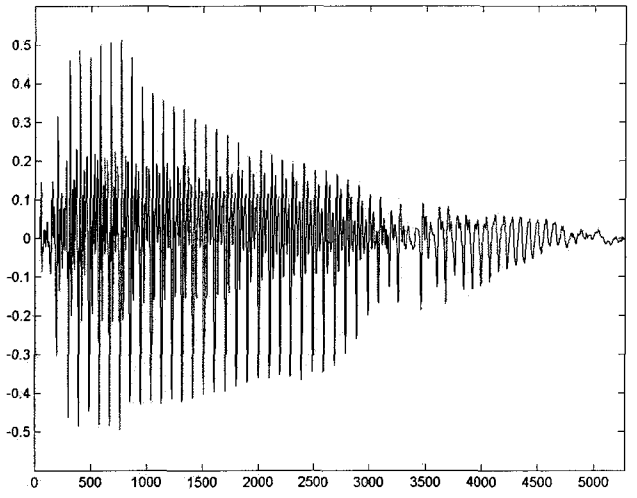


Figure A.1 – Première prononciation du chiffre 1 par la locutrice *mtr*.

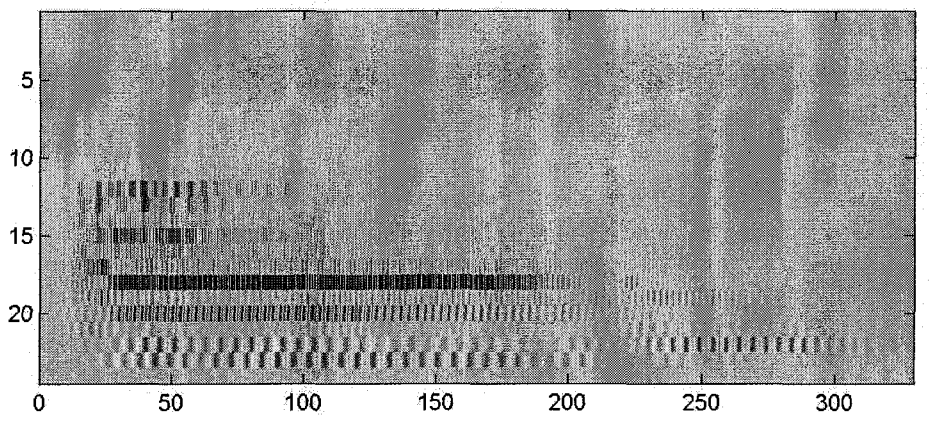


Figure A.2 – Filtrage du signal par un banc de filtre à 24 canaux.

Ces étapes constituent une modélisation très rudimentaire du comportement de la cochlée et des cellules ciliées du système auditif. De plus, l'utilisation de l'énergie (somme cumulée du signal) du canal comme indice d'excitation du neurone a été sélectionnée par souci de simplicité. Enfin, pour mieux imiter le traitement effectué par le système auditif, nous utilisons la rectification à la place de la valeur absolue pour les tests suivants.

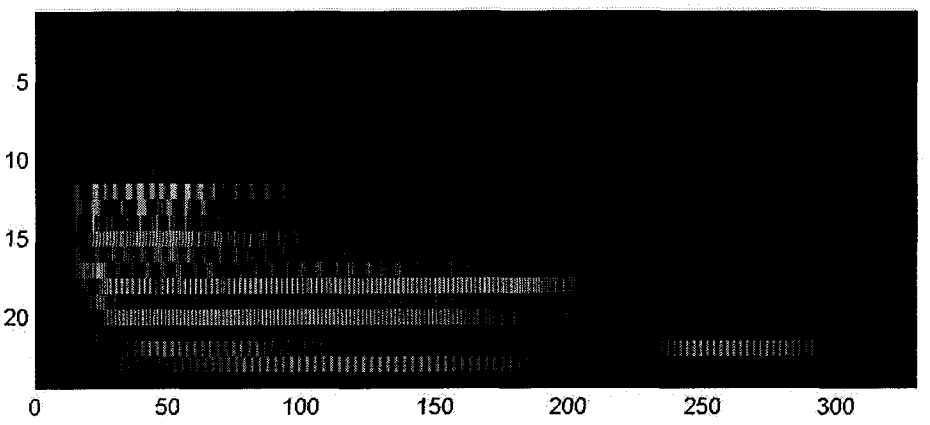


Figure A.3 – Valeur absolue des signaux pour les 24 canaux.

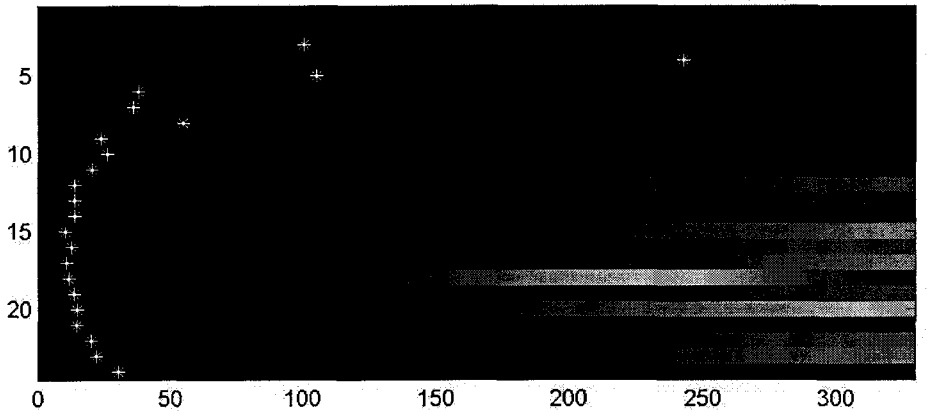


Figure A.4 – Somme cumulée et détection du seuil pour les 24 canaux.

A.1.3 Conclusion

En représentant l'ordre relatif dans lequel les impulsions sont générées, on obtient une forme relativement caractéristique des chiffres, formes qui possède une certaine indépendance au locuteur (tableaux A.1 et A.2).

On remarque que le traitement ne produit pas 24 impulsions à tous les coups. En effet, certains canaux sont peu activés et ne dépassent pas le seuil. Néanmoins, ces résultats nous ont encouragés à poursuivre avec cette approche.

TABLEAU A.1 - Représentation des impulsions générées par les dix prononciations du chiffre un par la locutrice *mtr*.

	Rang																			
Canaux	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																				
3																				1
4																				
5														2		2	1	2	1	1
6															2	2	3	1	2	
7															1	1	2	1	5	
8																		1	1	8
9														1	1	1	3	4		
10													1	2	2	3	1	1		
11	1				1					2		5	1							
12	5		3	1			1													
13	1	2	1	1	2		1	1				1								
14		3		1	2	2	1	1												
15	3	2	2		1	1	1													
16		1	1	5	1	1		1												
17		2	1	2		4	1													
18			2		2	2		1	2		1									
19					1		3	2	2	1	1									
20							2	3	1	3			1							
21								1	4	3	2									
22									1	1	5	3								
23											1	1	7	1						
24														4	4	1			1	

TABLEAU A.2 – Représentation des impulsions générées par les dix prononciations du chiffre deux par la locutrice *mtr*.

	Rang																			
Canaux	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1																				
2																				
3																				
4																				
5																		1	1	8
6									1		1	1	2		2	2			1	
7										1	3	3	1	1			1			
8											1			1	3	3	1	1		
9															1		1	1	5	2
10										2	4	2	1					1		
11										1	1		2	2	3	1				
12												1					2	5	2	
13													1			2	5	1	1	
14										1		2	1	5		1				
15			1						2	1		1	2	1	1	1				
16						1			6	3										
17		1					1	8												
18				1	1	3	5													
19	1		1	3	1	3		1												
20			1	3	5			1												
21		6	1	2			1													
22	9				1															
23		3	5	1						1										
24			1		2	3	3		1											

A.1.4 Modèle

Poursuivant dans cette direction, nous avons créé des modèles pour la reconnaissance des chiffres. Ces modèles sont simplement des vecteurs de 24 valeurs contenant l'information sur les 20 premiers canaux à générer une impulsion (pour les rangs 21 à 24, on utilise la valeur zéro).

Pour effectuer la reconnaissance, on accorde un poids k à chaque canal en fonction du modèle (dans ce test, seuls les 20 premiers canaux à décharger possèdent un poids non nul). Le canal qui devrait franchir le seuil en premier possède le plus grand poids et on va en décroissant pour les canaux suivants. Lorsqu'une impulsion est détectée dans ce canal, le poids est modifié en fonction de l'ordre dans lequel cette impulsion a été générée (inhibition). Enfin, on fait la somme de ces valeurs pour obtenir la similitude du signal avec le modèle.

$$Similitude = \sum_{k=20}^{1} k \times Inhibition^{Rang del'impulsion} \tag{A.1}$$

Ainsi, si l'ordre dans lequel les canaux de la simulation qui franchissent leur seuil est semblable à celui représenté par le modèle, la similitude va être élevée. Il reste ensuite à comparer les similitudes entre les divers modèles et le mot prononcé est celui dont le modèle possède la plus grande similitude. On peut remarquer que seuls les premiers canaux à décharger ont un véritable impact sur le pourcentage de vraisemblance. Il serait donc possible de faire la reconnaissance avec un nombre limité d'impulsions.

A.1.5 Exemple

En ordre, les vingt premières impulsions obtenues avec la première prononciation du chiffre un par la locutrice *mtr* (la figure A.4 à la page 127) proviennent des canaux : 15, 17, 18, 16, 19, 14, 12, 13, 21, 20, 22, 11, 23, 9, 10, 24, 7, 6, 8 et 3. Les poids k des canaux en ordre croissant, de 1 à 24, sont donc : 0, 0, 1, 0, 0, 3, 4, 2, 7, 6, 9, 14, 13, 15, 20, 17, 19, 18, 16, 11, 12, 10, 8 et 5.

Prenons pour la reconnaissance le même signal (*locfmtr1a.wav*) et une inhibition de $0,9$. Évidemment, les impulsions sont générées de façon identique. La similitude serait alors calculée comme suit :

$$0\times0.9^0+0\times0.9^0+1\times0.9^{19}+0\times0.9^0+0\times0.9^0+3\times0.9^{17}+...+5\times0.9^{15} = 120.94 \tag{A.2}$$

A.1.6 Reconnaissance

Pour une première reconnaissance, on utilise un modèle par mot par locuteur (pour quatre des cinq locuteurs)[2]. Pour déterminer ces 16 modèles, on assigne, par

[2] *hgi*, *mtr*, *rlo* et *slo*

ordre de génération, la position la plus probable (sur les dix prononciations) à chaque canal (tableau A.3).

TABLEAU A.3 – Modèles pour les chiffres un à quatre pour les locuteurs *hgi*, *mtr*, *rlo* et *slo*

Canaux	hgi				mtr				rlo				slo			
1																
2																
3				19												
4		18	16	14				20			20					18
5	20	16	15	9	18	20	10	13	20	20	18	20	20	11	17	13
6	19	19	17	2	16	13	2	2	19	13	16	18	16	9	8	5
7	18		19	1	19	10	13	1	16	11	15	3	15	12	11	3
8	17	17	20	3	20	15	15	3	15	12	13	1	17	14	18	11
9	16	20		7	17	17	14	10	18	15	17	4	19	20	20	17
10	12	15	18	20	14	14	7	17	17	17		14	18	19	19	19
11	6	13	14	18	7	18	1	18	14	10	19	17	14	13	16	
12	1	14	6	16	1	19	3	15	6	14	14	19	8	15	9	20
13	2	12	11	12	3	16	9	11	4	18	8	16	2	18	7	16
14	4	11	8	4	5	12	4	8	9	19	11	15	5	17	13	12
15	3	10	5	5	2	11	5	9	7	16	10	9	1	16	6	8
16	5	9	4	6	4	9	6	5	2	9	12	7	6	10	12	7
17	7	8	1	8	6	8	8	4	1	8	7	2	3	8	2	4
18	8	6	2	11	8	7	11	6	3	7	3	5	4	7	1	1
19	9	5	3	13	9	5	12	7	10	6	4	6	7	6	4	2
20	10	4	7	10	10	4	17	12	13	5	5	8	10	5	14	6
21	11	2	12	15	11	2	20	14	12	4	9	10	11	3	15	9
22	13	1	10		12	1	19	16	8	3	6	11	9	1	10	10
23	14	3	9	17	13	3	16	19	5	2	2	12	12	2	5	14
24	15	7	13		15	6	18		11	1	1	13	13	4	3	15

En observant les résultats de la reconnaissance (tableau A.4), il est surprenant de voir qu'un grand pourcentage des prononciations est classé correctement (79 %). De plus, les résultats obtenus avec le cinquième locuteur (*mpe*), non utilisés pour la création des modèles, sont de l'ordre de 90 %. Pour la stratégie qu'emploie le réseau qui utilise le codage par ordre de rang, le temps de calcul varie de façon linéaire en fonction du nombre de modèles. Le nombre de modèles n'est donc pas aussi contraignant qu'avec la majorité des systèmes de reconnaissance. Cependant, pour avoir la conscience tranquille, nous avons tenté de créer des modèles généraux pour les chiffres de un à quatre en utilisant les quatre premiers locuteurs (tableau A.5). La méthode pour obtenir ces modèles est la même méthode que décrite précédemment à la différence qu'on utilise les 40 prononciations du chiffre.

TABLEAU A.4 – Reconnaissance pour les chiffres un à quatre avec des modèles particuliers aux locuteurs.

Locuteur	Chiffre			
	1	2	3	4
hgi	100	100	40	100
mtr	100	90	30	90
rlo	70	100	80	100
slo	60	100	40	70
mpe	80	100	80	100

TABLEAU A.5 – Modèles généraux pour les chiffres un à quatre

Canaux	1	2	3	4
1				
2				
3				
4				
5	20	20	17	13
6	18	14	14	7
7	17	11	16	1
8	16	13	18	2
9	19	19	20	11
10	15	18	19	18
11	11	12	15	20
12	5	16	10	19
13	1	17	8	15
14	6	15	9	10
15	2	10	4	8
16	3	9	5	5
17	4	8	2	3
18	7	7	1	4
19	8	6	3	6
20	9	4	12	9
21	12	3	13	12
22	10	1	11	14
23	13	2	7	16
24	14	5	6	17

TABLEAU A.6 – Reconnaissance pour les chiffres un à quatre avec des modèles généraux.

Locuteur	Chiffre			
	1	2	3	4
hgi	100	100	70	100
mtr	100	90	10	90
rlo	40	100	60	100
slo	50	100	70	70
mpe	80	100	70	100

Les résultats de la reconnaissance pour ces modèles généraux (tableau A.6) sont très semblables aux résultats précédents ; on ne note qu'une légère diminution du taux de reconnaissance. Nous pouvons donc conclure, suite à ce test, qu'il doit être possible d'utiliser la stratégie du codage par ordre de rang pour réaliser un système de reconnaissance vocale. Nous pensons qu'en produisant plus d'une impulsion par canal dans la simulation, la distinction entre les mots augmenterait. Il reste à concevoir une méthode qui nous permettrait de le faire de façon logique et efficace.

ANNEXE B
PROTOTYPE

B.1 Codage par ordre de rang pour la reconnaissance vocale

Nous avons modifié l'application (*sna*), conçu par Daniel Pressnitzer, dans le but d'effectuer les différents tests présentés au chapitre 5. Les modifications concernent surtout la prise de fichiers, l'écriture des résultats, la reconnaissance et la documentation. Il est à noter que quelques modifications ont été apportées au prototype au cours du séjour à Toulouse. Il est donc possible que les résultats obtenus pour certains tests présentent de légères différences avec les résultats des tests présentés au début du chapitre sur le codage par ordre de rang.

B.1.1 Installation

Le prototype *sna* a été développé sur *MATLAB* version 6.0 et nécessite une version compatible. De plus, l'étape du filtrage pour les signaux est effectuée avec les fonctions *ERBFilterBank*, *ERBSpace* et *MakeERBFilters* [23] disponibles sur Internet[1]. Les fichiers principaux du prototype sont *ramp.m*, *sna.m*, *sna-guinit.m*, *sna-reco.m*, *sna-spike.m* et *snasigprocess.m*.

Pour démarrer l'application, on doit d'abord s'assurer que les fichiers nécessaires sont dans l'environnement de travail de *MATLAB* ; on n'a qu'à taper la commande *sna* dans la fenêtre de commandes. Si la commande fonctionne correctement, l'interface apparaîtra. Il est aussi possible d'avoir un court texte d'aide en utilisant la commande *help* suivie du nom du fichier.

B.1.2 Fichiers

ERBFilterBank traite un signal à l'aide d'un banc de filtres *gammatone*. Cette fonction ne demande qu'un vecteur en entrée et retourne les sorties dans une matrice où chaque colonne représente un canal. On fait appel à cette fonction dans *snasigprocess*.

[1]http ://rvl4.ecn.purdue.edu/ malcolm/interval/1998-010/

ERBSpace calcule une matrice de N fréquences uniformément espacées entre les hautes fréquences et les basses fréquences sur une échelle ERB. Cette fonction est nécessaire pour *MakeERBFilters*.

MakeERBFilters calcule les coefficients du banc de filtres *gammatone* (type de filtre défini par Patterson et Holdworth pour simuler la cochlée). Le résultat du calcul est retourné sous forme de matrice de coefficients du filtre où chaque colonne contient les coefficients pour un filtre d'ordre quatre. La fonction de transfert pour ces filtres partage le même dénominateur (les mêmes pôles), mais possède des numérateurs différents (des zéros différents). Enfin, ces coefficients sont rassemblés dans un vecteur qui peut être utilisé par *ERBFilterBank* pour implémenter les filtres.

ramp.m crée un effet de fondu au début et à la fin du signal que traite la fonction *snasigprocess*. Le vecteur signal est donné en entrée et le signal modifié est retourné.

sna.m est la fonction principale de l'application *sna*.

sna-guinit.m initialise l'interface de l'application *sna*. On fait référence à cette fonction au démarrage de *sna*.

sna-reco.m obtient le taux de similitude entre deux listes d'impulsions. Le taux de similitude est calculé pour les fonctions *Match* et *Automatch* dans *sna*.

sna-spike.m crée une liste d'impulsions qui possède les informations sur le canal, le seuil et la position en échantillons pour chaque impulsion. On se sert de cette fonction pour obtenir la liste des impulsions des modèles ainsi que celle pour les fichiers de la reconnaissance dans les fonctions *Model*, *Template*, *Match* et *Automatch* de *sna*.

snasigprocess.m traite le signal selon les paramètres choisis par l'utilisateur. On peut ajouter un effet de fondu, effectuer un filtrage à l'aide d'un banc de filtres *gammatone*, effectuer une rectification du signal et ajouter une compression.

B.1.3 Interface

Dans la partie supérieure droite de l'interface, un espace est réservé pour présenter « l'image » du signal traité. L'abscisse représente le temps et l'ordonnée représente les

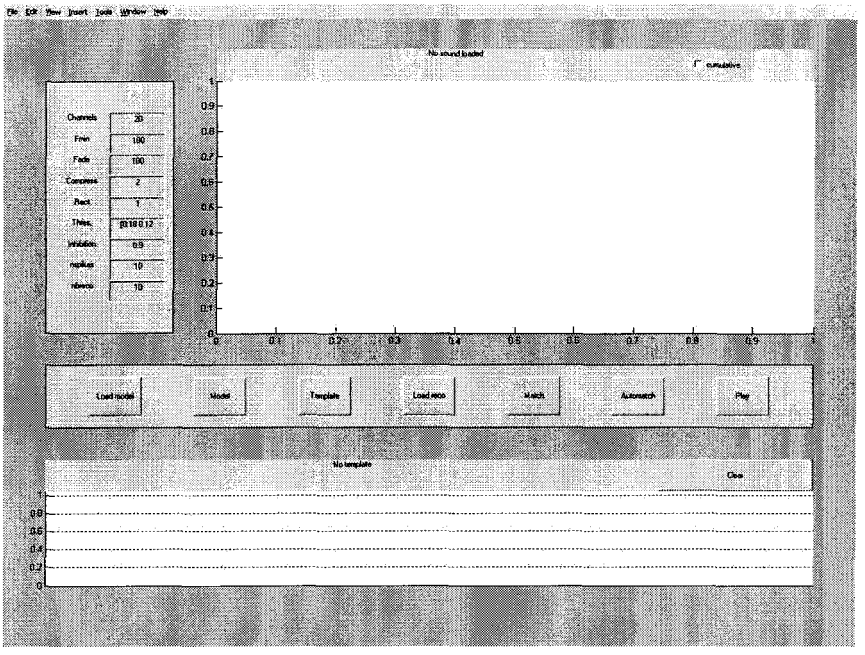


Figure B.1 – Interface.

canaux du banc de filtres. Au démarrage de l'application, cet espace est blanc et lors d'une réinitialisation, il devient vert. Pour représenter les signaux traités, on utilise une échelle de couleurs pour afficher les valeurs, les plus petites valeurs sont en bleu foncé et passent au bleu pâle, au vert, au jaune, à l'orange, au rouge jusqu'au brun par ordre croissant.

Au-dessus de l'espace de représentation, un espace est utilisé pour afficher le nom du fichier présenté ou différents avertissements. À droite de cette barre, des cases servent à activer ou à désactiver les options *mute* et *cumulative*.

Les paramètres du traitement *Fade*, *Channel*, *Fmin*, *Rectification*, *Compression*, *Threshold*, *Inhibition*, *nmodelspike* et *nrecospike* se trouvent dans un secteur situé à gauche de la fenêtre. Au démarrage, les valeurs par défaut sont affichées à droite du paramètre correspondant et l'utilisateur peut modifier ces valeurs.

Au centre de l'interface, une barre comprend sept boutons correspondant aux fonctions *Load model*, *Model*, *Template*, *Load reco*, *Match*, *Automatch* et *Play*.

Enfin, la table qui est située au bas de l'interface sert à afficher les résultats de certains tests. Au-dessus de cette table, on affiche le nom du fichier qui sert de modèle ou différents avertissements. Aussi, à l'extrême droite de cette barre, un bouton *clear* peut être activé.

B.1.4 Options

mute lorsque désactivé, cette option permet à l'application d'afficher certaines données (liste des modèles, liste des fichiers de la reconnaissance, résultat de la comparaison, etc) dans la fenêtre de commandes. Si la case est cochée par l'utilisateur, aucun message ne sera affiché hors de l'interface (excepté les erreurs).

cumulative permet d'utiliser des seuils au niveau de la somme cumulée du signal (à chaque canal) ou au niveau de l'amplitude du signal (paramètre par défaut).

B.1.5 Paramètres

Fade contient le nombre d'échantillons utilisés pour superposer un effet de fondu au début et à la fin du signal.

Channel est le nombre de canaux du banc de filtres.

Fmin représente la fréquence du filtre le plus grave.

Rectification demande une valeur binaire (zéro ou un). La rectification du signal est effectuée seulement si la valeur de ce paramètre est un.

Compression est la valeur du degré de la compression effectuée sur le signal (aucune compression n'est effectuée si la valeur est un).

Threshold contient la valeur des seuils utilisés pour le traitement.

Inhibition est le facteur d'inhibition de la reconnaissance. Une inhibition de zéro signifie que des modèles binaires sont utilisés.

nmodelspike contient le nombre d'impulsions utilisées pour créer les modèles. Par défaut, on se sert de la méthode des seuils groupés, mais en donnant un vecteur de dimension correspondant au nombre de seuils, la méthode des seuils séparés est utilisée et chaque valeur du vecteur donne le nombre d'impulsions du seuil correspondant.

Il est à noter que si la méthode des seuils groupés est utilisée, on utilise le paramètre *nmodelspike* pour les modèles et pour les fichiers de la reconnaissance. Le paramètre *nrecospike* devient, pour ces tests, inutile.

nrecospike contient le nombre d'impulsions utilisées pour la reconnaissance.

B.1.6 Fonctions

Load model nous permet de sélectionner un fichier son ou une listes de fichiers son pour le ou les modèles. Les fichiers son *wav* choisis séparément sont ajoutés à la liste des modèles. Si une liste de fichiers inscrite dans un fichier texte *txt* est sélectionnée, la liste des modèles ne contient que ceux contenus dans le fichier texte.

La liste dans le fichier texte doit avoir le format *MATLAB* d'une matrice de chaînes de caractères au nom de *listspeech*. Par exemple, *listspeech = ['cheminX fichierX.wav'; 'cheminY fichierY.wav'; ...];* . Pour le moment, il faut s'assurer que chaque nom complet possède le même nombre de caractères.

Model affiche le signal traité du modèle sélectionné. Évidemment, si aucun modèle n'existe, aucune tâche n'est effectuée. En appuyant une nouvelle fois sur *Model*, on passe au fichier suivant dans la liste des modèles.

Template affiche le signal traité du modèle sélectionné si un modèle est sélectionné. De plus, on calcule la position des seuils et on les affiche sur l'image par des caractères blancs.

Load reco fonctionne de la même manière que *Load model*, excepté qu'on effectue la sélection au niveau des fichiers utilisés uniquement pour la reconnaissance.

Match fonctionne seulement si un modèle a été créé (utilisation de *Template*) et s'il existe un fichier pour la reconnaissance (sélectionné avec *Load reco*). *Match* traite le fichier du modèle, calcule les impulsions pour créer un modèle. Ensuite, on répète ces étapes pour le fichier de la reconnaissance. Le patron des impulsions du fichier de reconnaissance est ensuite comparé avec le modèle et un taux de similitude est calculé. L'image du signal de reconnaissance traité ainsi que ses impulsions (en rose) sont affichées dans l'espace réservé à cette fin. On superpose aussi les impulsions du modèle (en blanc). Le taux de reconnaissance est alors représenté par une barre normalisée dans la table au bas de l'interface avec le nom du fichier de la reconnaissance comme

étiquette. Cette barre est de longueur « un » pour des patrons identiques au niveau du modèle et de la reconnaissance. Sinon, la longueur est proportionnelle à la similitude entre les deux patrons d'impulsions.

Si l'utilisateur active de nouveau *Match*, la reconnaissance se fait avec le fichier suivant dans la liste des fichiers de reconnaissance. Le nouveau taux de reconnaissance est ensuite affiché à droite des taux déjà calculés dans la table.

Automatch demande l'existence d'une liste de fichiers pour les modèles et d'une autre liste pour les fichiers de la reconnaissance. Si ces exigences sont satisfaites, une boîte de dialogue demande à l'utilisateur de donner le nom du fichier de résultats qui va être créé à la fin des comparaisons et qui va contenir les résultats de ces comparaisons sous forme de matrice. Pour produire les matrices de comparaisons, tous les modèles sont créés et on compare à tour de rôle chacun des fichiers de la reconnaissance. Les paramètres de chaque test sont aussi inscrits dans le fichier de résultats précédant la matrice de résultats. Il est à noter que les données inscrites dans le fichier sont ajoutées et ne suppriment pas le contenu antérieur du fichier.

Il est possible d'effectuer plus d'un test avec cette fonction. En effet, si un vecteur est utilisé pour donner la valeur du nombre de canaux, on effectue un test pour chaque valeur ainsi donnée. De plus, si on utilise la méthode des seuils groupés et qu'un vecteur est utilisé pour déterminer le nombre d'impulsions utilisées pour la reconnaissance (*nrecospike*), on effectue un test pour chaque valeur ainsi donnée. Il est à noter que pour ces tests, seul le nombre d'impulsions utilisées à la reconnaissance change (pas le nombre d'impulsions pour créer les modèles). Enfin, si un vecteur est utilisé pour le nombre de canaux ainsi que pour le nombre d'impulsions à la reconnaissance, on effectue un test pour toutes les combinaisons possibles des valeurs de ces vecteurs. Dans le cas où plus d'un test est effectué, les paramètres de ces tests ainsi que les résultats sont inscrits à la suite dans le fichier précédemment sélectionné.

Play fait jouer le fichier modèle sélectionné.

Clear réinitialise l'interface (excepté au niveau des paramètres du traitement).

BIBLIOGRAPHIE

[1] Jun-Ichi Aoe. The time-sliced paradigm-a connectionist method for continuous speech recognition. *INFORMATION SCIENCES*, (93) :133–158, 1996.

[2] Najet Arous and Noureddine Ellouze. Cooperative supervised and unsupervised learning algorithm for phoneme recognition in continuous speech and speaker-independent context. *Neurocomputing*, 51 :225–235, april 2003.

[3] Olivier Cappé. H2m : A set of matlab/octave function for the em estimation of mixtures and hidden markov models. Technical report, august 2001.

[4] P. Chandrasekaran, M. Bodruzzan, and M. Malkani. Speech recognition using pulse coupled neural networks. In *Proceedings of thirtieth Southeasthern Symposium on System Theory*, volume 3, pages 515–519, may 1998.

[5] A. Delorme and S.J. Thorpe. Face identification using one spike per neuron : resistance to image degradation. *Neural Networks*, 14(6–7) :795–803, 2001.

[6] Steeve Gagné. Analyse de parole en milieu difficile : Utilisation d'histogrammes par intervalles pour le suivi de hauteur tonale. Technical report, Chicoutimi : Université du Québec à Chicoutimi, groupe de recherche ERMETIS, august 1997.

[7] Bärbel Herrnberger, Stefan Kempf, and Günter Ehret. Basic maps in the auditory midbrain. *Biological Cybernetics*, 87(4) :231–240, october 2002.

[8] Ferrandez J.M., Rodellar V., and Gomez P. A biological hierarchical system for speech recognition. In *IWANN '99 : international world-conference on artificial and natural networks*, volume 2, pages 279–288, june 1999.

[9] G.V. Kiran and T.V. Sreenivas. A novel method of analysing and comparing responses of hearing aid algorithms using auditory time-frequency representation. In *EUROSPEECH*, pages 61–64, september 2003.

[10] Fang Liu, Yoko Yamaguchi, and Hiroshi Shimizu. Flexible vowel recognition by the generation of dynamic coherence in oscillator neural networks : speaker-independent vowel recognition. *Biological Cybernetics*, (71) :811–830, 1994.

[11] Stéphane Loiselle. Système de reconnaissance de la parole pour la commande vocale des équations mathématiques. Technical report, Chicoutimi : Université du Québec à Chicoutimi, august 2001.

[12] Wolfgang Maass and Christopher M. Bishop. *Pulsed Neural Networks.* MIT Press, 1999.

[13] David Mercier and Renaud Séguier. Spiking neurons (stanns) in speech recognition. *NNA '02*, february 2002.

[14] Brian C.J. Moore. *Psychology of hearing.* Academic Press, 1997.

[15] Christian Näger, Jan Storck, and Gustavo Deco. Speech recognition with spiking neurons and dynamic synapses : a model motivated by the human auditory pathway. *Neurocomputing*, 44–46 :937–942, 2002.

[16] David Nonnon. Système de reconnaissance intégrant une analyse perceptive : Séparation de locuteurs à l'aide d'histogrammes de passages par seuils. Technical report, Chicoutimi : Université du Québec à Chicoutimi, groupe de recherche ERMETIS, june 1997.

[17] Laurent Perrinet. *Comment déchiffrer le code impulsionnel de la Vision ? Étude du flux parallèle, asynchrone et épars dans le traitement visuel ultra-rapide.* PhD thesis, Université Paul Sabatier, 2003.

[18] Ramin Pichevar and Jean Rouat. Nonlinear speech processing with oscillatory neural networks for speaker segregation. In *EUSIPCO, Toulouse, France*, 2002.

[19] Ramin Pichevar and Jean Rouat. Cochleotopic/amtopic (cam) and cochleotopic/spectrotopic (csm) map based sound source separation using relaxation oscillatory neurons. In *IEEE Neural Networks for Signal Processing Workshop, Toulouse, France*, 2003.

[20] Joseph W. Picone. Signal modeling techniques in speech recognition. *PROCEEDINGS OF THE IEEE*, 81(9) :1215–1247, september 1993.

[21] Sumeet Sandhu and Oded Ghitza. A comparative study of mel cepstra and eih for phone classification under adverse conditions. In *ICASSP 95*, volume 1, pages 409–412, 1995.

[22] Robert Sataloff. La voix humaine. *Pour la science*, (184) :36–45, february 1993.

[23] Malcolm Slaney. Auditory toolbox version 2. Technical report, 1998.

[24] Jan Storck, Frank Jäkel, and Gustavo Deco. Temporal clustering with spiking neurons and dynamic synapses : towards technological applications. *Neural networks*, 14 :275–285, 2001.

[25] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. Spike-based strategies for rapid processing. *Neural Networks*, 14(6–7) :715–725, 2001.

[26] Simon J. Thorpe. Ultra-rapid scene categorization with a wave of spikes. In *Biologically Motivated Computer Vision*, pages 1–15, 2002.

[27] Rufin VanRullen and Simon J. Thorpe. Surfing a spike wave down the ventral stream. *Vision Research*, 42(23) :2593–2615, august 2002.

[28] DeLiang Wang and David Terman. Image segmentation based on oscillatory correlation. *Neural Computation*, 9(4) :805–836, 1997.