UNIVERSITÉ SAAD DAHLAB DE BLIDA

Faculté De Technologie

Département D'électronique

MÉMOIRE DE MAGISTER

Spécialité : Signaux et Systèmes

IMPLEMENTATION D'ALGORITHME DE RECONNAISSANCE DES MOTS ISOLES PAR LES CHAINES DE MARKOV CACHEES SUR CARTE DSP

Par

Ahmed BENTALAYE

Devant le jury composé de :

A.Guessoum	Professeur, USD, Blida	Président
M.Guerti	Professeur, ENSP, Alger	Examinatrice
N.Benblidia	Maître de conférences A, USD, Blida	Examinatrice
Z.Benselama	Maître de conférences A, USD, Blida	Rapporteur
F.Ykhlef	Maître de conférences B, USD, Blida	Invité

Blida, Décembre 2012

ملخص

إن استخدام الأنظمة المتحكم بها بواسطة الأوامر الصوتية يتطلب شروط استعمال مضبوطة للتحصل على نتائج مرضية، في هذا الإطار، قمنا بالتركيز في هذا البحث على تصميم نظام أوامر صوتية للتحكم في سيارة، اهتممنا بثلاثة جوانب أساسية. أولها قمنا بدراسة مختلف الطرق المنتهجة عموما في مجال التعرف الآلي على الأصوات و خصوصا على الكلمات المعزولة، حيث سمحت لنا هذه الدراسة باختيار طرق الإحصاء التي ترتك ز على السلاسل المخفية لماركوف. و من أجل جعل استجابة النظام تعمل في الوقت الحقيقي، قمنا بإدماج خوارزمية VAD (الكاشف الآلي على ا لنشاط الصوتي). وأخيرا من أجل إجراء الإختبارات من الناحية العملية قمنا بجمع هذه الخوارزميات و تحميلها على متن بطاقة المعالجة الرقمية للإشارات DSP " TMS 320 C 6713" .

RÉSUMÉ

L'utilisation des systèmes embarqués de reconnaissance automatique de la parole nécessite des conditions d'utilisation contraintes pour que ces derniers obtiennent des résultats convenables. Dans ce cadre, nous avons concentré nos études pour réaliser un système de commande vocal d'une voiture, nous nous sommes intéressés pratiquement à trois aspects. La première étant d'étudier les différents types de reconnaissances automatiques de la parole et particulièrement des mots isolés; cette étude nous a permis de choisir des méthodes statistiques sur la base des modèles de Markov cachés (Hidden Markov models). Et pour que notre système soit autonome nous avons travaillé avec l'algorithme de détection de l'activité vocale V.A.D. Et enfin, nous avons implémenté tous ces algorithmes sur une carte DSP « TMS 320 C 6713 » cela pour faire les tests sur la réalisation pratique.

ABSTRACT

The use of embedded systems of automatic speech recognition requires constraints conditions of use to get proper results. In this context, we focused our studies to develop a car voice control system; we looked at three aspects to make a very practical application. The first being to study different types of automatic voice recognition and particularly that of isolated words, this study allowed us to choose statistics methods based on the Hidden Markov models (HMM), and we have worked also with the of voice activity detection algorithm VAD. And finally we have implemented all these algorithms on a DSP card « TMS320 C 6713 » to make the tests on the practical development.

DÉDICACES

Je dédie ce modeste travail à mes chers parents qui m'ont tant donné sans se lasser.

À mon frère pour son encouragement.

À mon épouse et mes filles Achouak et Racha, pour leurs patiences, pour les nuits blanches passées loin d'elles, pour les weekends ratés, enfin pour tous les manquements que je dois acquitter.

À tous mes amis et collègues de travail.

REMERCIEMENT

Je tiens, en premier lieu, à remercier Dieu pour sa miséricorde, sa bonté, son obligeance et la faveur qu'il m'a donné afin de m'engager dans la voie de la recherche.

Ce travail aurait impossible sans l'inestimable et précieux conseil direct ou indirect de certaines personnes.

Je dois donner une mention spéciale à mon directeur du projet, Mr Z.Benselama, maître de conférences à USD de Blida, qui était très patient et m'a donné un peu de son précieux temps pour m'expliquer les objectifs de ce travail et qui progressivement l'a augmenté jusqu'à son heureux achèvement.

Sincère remerciement pour les membres du jury pour avoir évalué ce modeste travail : Mr Guessoum, Mme Guerti, Mlle Benblidia et Mr Ykhlef.

Je tiens à remercie aussi Mr M.A.Benchechali, M.Mekrazi, M.A.Makhtiche, S.Taazont, A.Rahiche, A.Bacha pour leurs aides et leurs conseils qui ont été fructueux pour ce travail.

Je remercie tous ceux qui croient que le savoir est universel et qui nous donnent tant de plaisir sur internet en nous guidant et en nous simplifiant les parcours de la recherche scientifique.

TABLE DES MATIERES

CHAPITRE III : DETECTION D'ACTIVITE VOCALE EN TEMPS REEL

CHAPITRE IV : GENERALITE ET ARCHITECTURE DES PROCESSEURS DSP

CHAPITRE V : IMPLEMENTATION PRATIQUE ET ÉVALUATION DES RESULTATS

Liste des Acronymes

ADC	Analog-Digital Converter
CCS	Code Composer Studio
CMOS	Complementary Metal Oxide Semiconductor
CMS	Suppression de la Moyenne Cepstral
DCT	Discrete Cosine Transform
DSP	Digital Signal Processor - processeur de signal numérique
DTW	Dynamic Time Warping (Alignement temporel dynamique)
EM	Expectation Maximisation
FFT	Fast Fourier Transform - Transformation de Fourier rapide
HMM	Hidden Markov Model (Modèle de Markov Caché)
HTK	Hidden Markov models Toolkit
IDCT	Inverse Discrete Cosine Transform
IDE	Integrated development environment
GMM	Gaussian Mixture Models
LDR	Light Dependent Resistor
LED	Light-Emitting Diode
LFCC	Linear Frequency Cepstral Coefficients
LPC	Linear predictive coefficients
LPCC	Linear predictive cepstral coefficients
MAC	Multyply and ACcumulate
MFCC	Mel Frequency Cepstral Coefficients
MLE	Maximum Likelihood Estimation
MLP	Multy Layer Perceptron
NPC	Neural Predictive Coding
PLP	Perceptual Linear Prediction
RAM	Random Access Memory - mémoire vive
RAP	Reconnaissance Automatique de la Parole
RASTA	RelAtive SpecTrAl
RN	Réseau de Neurone
ROM	Read Only Memory - mémoire morte
SRAP	Système de Reconnaissance Automatique de la Parole
SVM	Support Vector Machine
TAP	Traitement Automatique de la Parole
TTL	Transistor-Transistor Logic
VAD	Voice Activity Detection
ZCR	Zero Crossing Rate

LISTE DES FIGURES

LISTE DES TABLEAUX

INTRODUCTION GÉNÉRALE

La reconnaissance vocale ou reconnaissance automatique de la parole (Automatic Speech Recognition ASR) est une technique informatique, qui permet d'analyser un mot ou une phrase captée au moyen d'un microphone, pour la transcrire sous la forme d'un texte exploitable par une machine. La reconnaissance vocale, ainsi que la synthèse vocale, l'identification du locuteur ou la vérification du locuteur, sont parties des techniques de traitement de la parole. Ces techniques permettent notamment de réaliser des interfaces vocales c'est-à-dire des Interfaces Homme-Machine (IHM), où une partie de l'interaction se fait par la voix. Parmi les nombreuses applications, on peut citer les applications de dictée vocale sur PC où la difficulté tient à la taille du vocabulaire et à la longueur des phrases, mais aussi les applications téléphoniques de type serveur vocal, où la difficulté tient plutôt à la nécessité de reconnaître n'importe quelle voix dans des conditions acoustiques variables et souvent bruyantes (téléphones mobiles dans des lieux publics).

En revanche, dans le cas de ce mémoire, nous avons réalisé un système, entièrement autonome, capable de commander vocalement un automate à savoir une petite voiture.

L’élaboration d’un tel système va demander un croisement du traitement du signal numérique, le traitement du langage et la technologie des systèmes embarqués telle que les cartes DSP (Digital Signal Processor) pour notre cas nous avons opté pour Texas Instrument « TMS 320 C 6713» qui était disponible au sein de notre laboratoire.

Ce manuscrit contient cinq parties principales.

La première partie parle sur des introductions sur la reconnaissance automatique de la parole, ensuite dans la deuxième partie développe l’état de l’art où il présente les différents algorithmes et techniques d’analyse du signal vocal utilisés dans le domaine de la reconnaissance de la parole ainsi que de mots isolés, ceci est suivi, dans la troisième partie, par les détails de l’algorithme de détection d’activité vocale D.A.V afin de rendre le système autonome. La quatrième partie détaille la carte utile « TMS320 C 6713 » et on a terminé par la dernière partie qui représente l’implémentation de notre application sur ladite carte, les tests effectués, les résultats obtenus et les constatations pratiques.

CHAPITRE I

RECONNAISSANCE AUTOMATIQUE DE LA PAROLE

I.1. Introduction

La reconnaissance vocale a pour but de permettre à un utilisateur de s'adresser oralement à une machine pour des tâches diverses : transcription, commande, traduction… L'utilisation de la reconnaissance de la parole possède donc un champ d'application très vaste qui nécessite encore des travaux de recherche. L'évolution des technologies a permis l'émergence de solutions entièrement logicielles et grands publics. Cependant, l'implantation d'un système automatique de la parole dans des conditions réelles pose de nombreux problèmes tels que le bruit, l'état du locuteur, la qualité d'enregistrement, la difficulté du vocabulaire, le mode d'élocution, l'intégration du système dans un équipement.

Dans le domaine de la reconnaissance automatique de la parole, on distingue trois grands types d'applications:

- les systèmes de commandes vocales [4],[5],[6],[7],[8].
- les machines à dicter [1],[3].
- les systèmes de compréhension [2],[9],[10],[11].

I.2. Historique

La reconnaissance de la parole est une discipline récente. Depuis le début des années 1950 et les premiers travaux dans le domaine de la reconnaissance automatique de la parole, les objectifs ont bien évaluée.

Quelques dates clés :

- 1949 : étude de la déviation électrique du spot d'un oscilloscope en fonction du signal de parole [22] ;
- 1952 : reconnaissance des 10 chiffres, pour un *monolocuteur* , par un dispositif électronique câblé [23];
- 1960 : utilisation des méthodes numériques
- 1965 : reconnaissance de *phonèmes* en parole continue
- 1968 : reconnaissance de mots isolés par des systèmes implantés sur gros ordinateurs (jusqu'à 500 mots)

- 1969 : utilisation d'informations linguistiques
- 1971 : lancement du projet ARPA aux USA (15 millions de dollars) pour tester la faisabilité de la compréhension automatique de la parole continue avec des contraintes raisonnables.
- 1972 : premier appareil commercialisé de reconnaissance de mots.
- 1976 : fin du projet ARPA ; les systèmes opérationnels sont HARPY, HEARSAY I et II et HWIM.
- 1978 : commercialisation d'un système de reconnaissance à microprocesseurs sur une carte de circuits imprimés.
- 1981 : utilisation de circuits intégrés VLSI (Very Large Scale Integration) spécifiques du traitement de la parole.
- 1981 : système de reconnaissance de mots sur un circuit VLSI.
- 1983 : première mondiale de commande vocale à bord d'un avion de chasse en France.
- 1985 : commercialisation des premiers systèmes de reconnaissance de plusieurs milliers de mots.
- 1986 : lancement du projet japonais ATR de téléphone avec traduction automatique en temps réel.
- 1988 : apparition des premières machines à dicter par mots isolés [18].
- 1989 : recrudescence des modèles connexionnistes neuro-mimétiques.
- 1990 : premières véritables applications de dialogue oral homme-machine [20],[21].
- 1994 : IBM lance son premier système de reconnaissance vocale sur PC[12], [13],[14].
- 1997 : lancement de la dictée vocale en continu par IBM [15],[16],[19].
- 2000 : développement des serveurs vocaux et l'exploitation au domaine de télécommunications [25].
- 2000 : Lernout & Hauspie a acquis Dragon Systems [17].
- 2001 : Scansoft Inc a acquis tous les droits sur les produits de reconnaissance vocale Dragon Naturally Speaking.
- 2003 : Scansoft Inc le leader mondial en reconnaissance de la parole a acquis Speechworks [82].
- Les autres travaux qui suivent sont des améliorations des logiciels de la dictée vocale.

I.3. Principaux modules d'un système de RAP

Les systèmes classiques de RAP sont composés essentiellement de cinq modules (figure I.1) :

– **la paramétrisation du signal**, qui doit permettre de ne garder que les informations pertinentes de ce dernier ;

– **les modèles acoustiques**, qui doivent représenter au mieux les unités acoustiques choisies (phonèmes, diphones, mots. . .) ;

– **les modèles linguistiques**, qui doivent être une représentation la plus vraisemblable possible du langage ;

– **le dictionnaire**, qui doit contenir l'ensemble des mots que l'on souhaite pouvoir reconnaître (dans certains cas le dictionnaire peut être spécifique à une application) ;

– **le système de reconnaissance** lui-même.

Ces différentes composantes d'un système de RAP, bien que toutes nécessaires pour la reconnaissance de parole continue, sont relativement indépendantes les unes des autres.

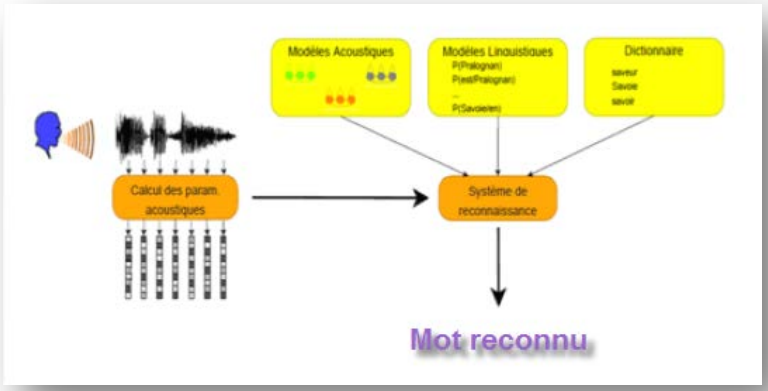
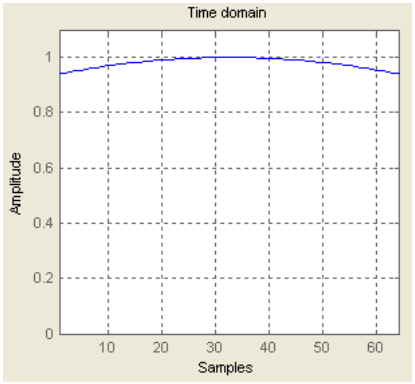


Fig.I.1 : *Schéma de principe d'un système de RAP.*

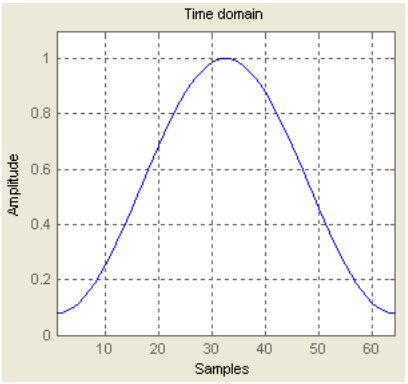
Remarque : Plusieurs extracteurs peuvent être mis en parallèle pour réaliser de la fusion de données.

Pour améliorer le signal d'entrée, on doit procéder à un **prétraitement**, c'est-à-dire à diminuer le bruit et ensuite, à extraire un vecteur caractéristique pour la classification. Il faut d'abord délimiter le début et la fin de la parole afin d'améliorer la reconnaissance.

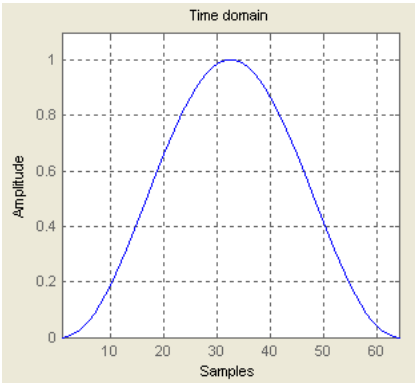
Le signal acoustique variant au cours du temps, donc il est nécessaire d'effectuer le prétraitement sur une trame de courte durée (10 à 30 ms). On peut considérer ainsi le signal comme quasi stationnaire. On peut utiliser comme fenêtre pour la récupération des trames : les fenêtres de **Hamming**, de **Hanning**, de **Kaiser**, etc. (figure I.2). L'analyse spectrale considère le signal comme périodique. Les fenêtres permettent donc de diminuer l'influence des échantillons situés au début et à la fin des trames. Ainsi, ces fenêtres possèdent des propriétés qui permettent de minimiser l'introduction d'artefact dans le spectre.



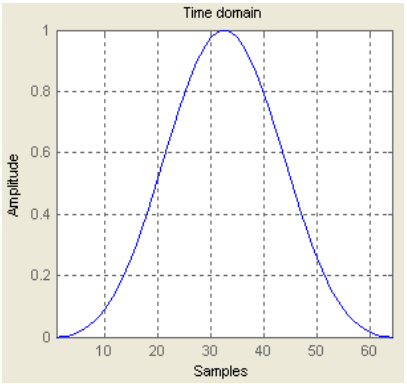
fenêtre de Kaiser



fenêtre de Hamming



fenêtre de Hanning



fenêtre de Blackman

Fig.I.2 : *Les différents types de fenêtres utilisés pour la RAP.*

Le vecteur de caractéristique est ensuite obtenu par une méthode d'analyse fréquentielle ou de modélisation explicite du signal. Ces méthodes sont décrites dans la prochaine partie. Une fois les paramètres obtenus par le traitement du signal, il est parfois nécessaire de réaliser une sélection des caractéristiques. En effet, si le vecteur obtenu possède une dimension trop grande, il est souvent intéressant de ne pas conserver toutes les caractéristiques (*Curse of dimensionality* [26] informations redondantes et non significatives, exemple : Paramètres liés à la reconnaissance du locuteur). La sélection de caractéristiques a donc deux objectifs : réduire la dimension de l'espace des données et minimiser la quantité d'incertitude. En reconnaissance de la parole, les vecteurs de caractéristiques obtenus sont souvent de longueurs variables. Cette variabilité permet de modéliser les variations des séquences des mots, l'accent ou la vitesse d'élocution, etc. Cependant, les vecteurs peuvent être de taille constante si l'on étudie l'évolution du signal.

Lorsque les formes à reconnaître ont été acquises et paramétrées de façon satisfaisante, l'étape essentielle de la reconnaissance est la classification. Il existe trois (03) grandes méthodes de classification (*Reconnaissance automatique de la parole* [27]):

- La recherche des formes similaires à la forme à reconnaître,
- La classification par méthode probabiliste,
- La construction de surface de décision dans l'espace de représentation des formes.

Dans ces trois méthodes de classification, il convient de choisir une distance adaptée. Les métriques principalement utilisées sont les distances euclidiennes et de Mahalanobis. De plus, dans les deux dernières méthodes, les formes doivent être apprises au cours d'une phase d'apprentissage complexe.

Pour être complet, un système de reconnaissance automatique de la parole peut posséder des analyseurs lexicaux, syntaxiques et sémantiques. Ces étapes sont rajoutées notamment pour des systèmes analysant la parole continue. Il améliore leur taux de reconnaissance en s'appuyant sur l'orthographe, la grammaire et le contexte de la phrase. En appliquant des probabilités basées sur la linguistique, il diminue le vocabulaire possible, améliorant ainsi la reconnaissance du prochain mot prononcé.

I.3.1. Notions de communication

La production de la parole fait référence à diverses situations qu'il est nécessaire d'expliciter. On a deux grands types de systèmes liés aux utilisateurs.

- ✓ **Systèmes mono-locuteurs (speaker dependant)** : Système utilisable par un seul locuteur, est caractérisé par la technique d'apprentissage, où une seule et même personne doit dicter un ensemble de mots, ce qui permet d'optimiser le taux de reconnaissance et d'étendre le vocabulaire utilisable.
  ***Inconvénient*,** seule la personne ayant fourni son empreinte vocale (lors de la phase d'apprentissage) peut travailler.
- ✓ **Systèmes multi-locuteurs (speaker independant)** : Système utilisable par plusieurs locuteurs qui utilise une base de données contenant des

empreintes moyennes autorisant la reconnaissance de plusieurs voix.

Inconvénient, le système n'est pas doté de capacités d'apprentissage et le nombre de mots est plus limité.

Chaque système peut être ensuite classé en fonction du signal fourni en entrée. Il existe trois types de signal d'entrée correspondant au découpage de la parole.

- **Parole continue (continuous speech)** : Système capable de reconnaître les mots prononcés dans une phrase.
- **Mots isolés (isolated word)** : Système qui nécessite que le locuteur fasse une pause entre chaque mot prononcé.
- **Mots clefs (key spotting)** : Reconnaissance de mots-clés prononcés par plusieurs locuteurs sans phase d'apprentissage.

La parole est un signal acoustique qui possède diverses propriétés. On peut considérer plusieurs unités de langage liées à la parole.

- **Parole** : Signal réel, continu, d'énergie finie, non stationnaire.
  Un signal est continu s'il est une application dans le temps continu (R+). Il est d'énergie finie si son intégral converge. Et un signal est un phénomène non stationnaire si ses propriétés statistiques changent continuellement dans le temps.
- **Vocabulaire** : Liste de mots ou d'unités de langage qui peut être reconnue par le système de reconnaissance.
- **Phonème (phoneme)** : Entité théorique qui correspond à la plus petite unité discrète que l'on peut distinguer dans une chaîne parlée. Elle peut être prononcée de façon différente en fonction du locuteur et de sa position dans le mot.
- **Allophone (Allophone)** : Réalisation sonore possible pour un phonème.

I.4. Reconnaissance de mots isolés

L'absence dans le signal vocal d'indicateurs sur les frontières de *phonèmes* et de mots constitue une difficulté majeure de la reconnaissance de la parole. De ce fait, la reconnaissance de mots prononcés artificiellement de

façon isolée (c'est-à-dire que tous les mots prononcés sont séparés par des silences de durées supérieures à quelques dixièmes de seconde) représente une simplification notable du problème [1].

I.4.1. Les techniques de reconnaissance vocale

Deux approches, l'une plus globale, et l'autre plus analytique permettent d'appréhender la reconnaissance des mots.

Dans l'approche **globale,** l'unité de base sera le plus souvent le **mot** considéré comme une entité globale, c'est à dire non décomposée. L'idée de cette méthode est de donner au système une image **acoustique** de chacun des mots qu'il devra identifier par la suite. Cette opération est faite lors de la phase d'apprentissage, où chacun des mots est prononcé une ou plusieurs fois. Cette méthode a pour avantage d'éviter les effets de coarticulation, c'est-à-dire l'influence réciproque des sons à l'intérieur des mots. Elle est cependant limitée aux petits vocabulaires prononcés par un nombre restreint de locuteurs.

L'approche **analytique,** qui tire parti de la structure **linguistique** des mots, tente de détecter et d'identifier les composantes élémentaires (phonèmes, syllabes…). Celles-ci sont les unités de base à reconnaître. Cette approche a un caractère plus général que la précédente : pour reconnaître de grands vocabulaires, il suffit d'enregistrer dans la mémoire de la machine les principales caractéristiques des unités de base.

Pour la reconnaissance de mots isolés à **grand vocabulaire,** la méthode globale ne convient plus, car la machine nécessiterait une mémoire et une puissance considérable pour respectivement stocker les images acoustiques de tous les mots du vocabulaire et comparer un mot inconnu à l'ensemble des mots du dictionnaire. Il est de plus impensable de faire dicter à l'utilisateur l'ensemble des mots que l'ordinateur a en mémoire. C'est donc la méthode **analytique** qui est utilisée : les **mots** ne sont pas mémorisés dans leur intégralité, mais traités en tant que **suite de *phonèmes*** [1].

I.4.2. Principe général de la méthode globale pour un système mono-locuteur

Le principe est le même que ce soit pour l'approche analytique ou l'approche globale, ce qui différencié ces deux méthodes est l'entité à reconnaître : pour la première il s'agit du *phonème,* pour l'autre du mot.

- La **phase d'apprentissage** : un locuteur prononce l'ensemble du vocabulaire, souvent plusieurs fois, de façon à créer en machine le **dictionnaire de références** acoustiques. Pour l'approche analytique, l'ordinateur demande à l'utilisateur d'énoncer des phrases souvent dépourvues de toute signification, mais qui présentent l'intérêt de comporter des successions de *phonèmes* bien particuliers. Pour un système multi-locuteur, cette phase n'existe pas, c'est la principale différence.

- La **phase de reconnaissance :** un locuteur (le même que précédemment, car nous sommes dans le cas d'un système *mono-locuteur)* prononce un mot du vocabulaire. Ensuite la reconnaissance du mot est un problème typique de reconnaissance de formes. Tout système de reconnaissance des formes comporte toujours les trois parties suivantes:

    - Un **capteur** permettant d'appréhender le phénomène physique considéré (dans notre cas un microphone),
    - Un étage de **paramétrisation** des formes (par exemple MFCC),
    - Un étage de **décision** chargé de classer une forme inconnue dans l'une des catégories possibles.

On retrouve ces trois étages dans un système de reconnaissance vocale, comme le montre la figure I.3 :

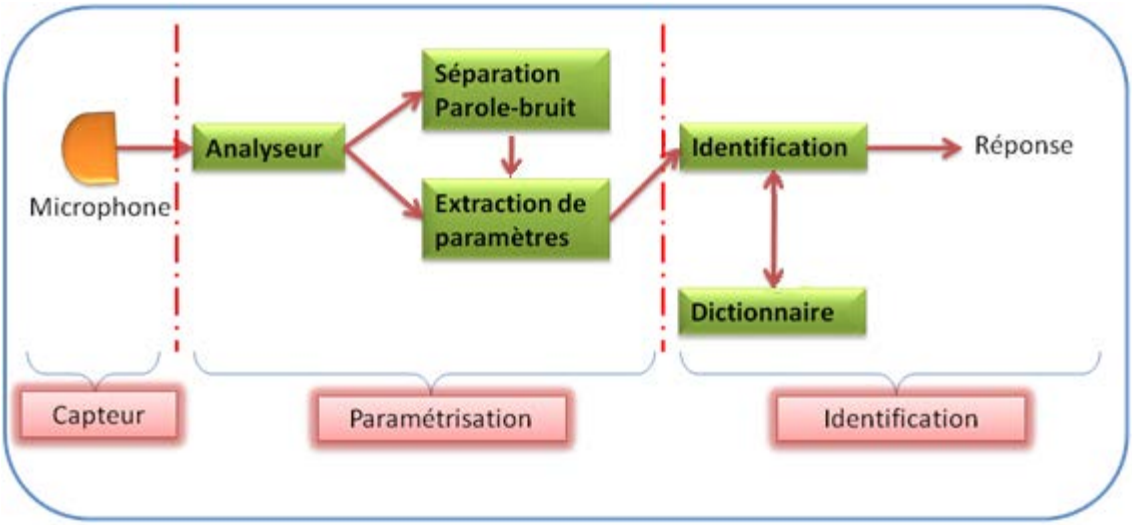


***Fig.I.3** : Phase d'identification de la parole*

I.4.3. Description des différentes phases de reconnaissance

i) Le capteur : un signal électrique est issu du **microphone** lorsque le locuteur parle.

ii) Paramétrisation du signal : cet étage, dont le rôle est d'analyser et de paramétrer le signal vocal du locuteur, consiste en un **traitement mathématique du signal.** Cette étape va être développée dans la partie suivante.

Difficulté rencontrée : comme nous sommes dans le cas de mots isolés, les frontières des mots (début et fin de mot) sont généralement déterminées en repérant les intersections de la courbe d'énergie du signal avec un ou plusieurs seuils évalués expérimentalement. Si la prise de son est effectuée dans un local bruité, le bruit de fond additionné au signal vocal peut dégrader les performances du système de reconnaissance.

iii) Prise de décision du choix du mot: Le signal vocal émis par l'utilisateur, une fois paramétré, va pouvoir être comparé aux mots du dictionnaire de référence en termes d'images acoustiques. L'algorithme de reconnaissance permet de choisir le mot le plus ressemblant, par calcul d'un taux de similitude entre le mot prononcé et les diverses références. Pour simplifier le problème, le programme va comparer le mot prononcé par le locuteur avec ceux qui sont en mémoire depuis la phase

d’apprentissage. Ce calcul n'est pas simple, même pour un locuteur unique, car les mots, donc les formes, à comparer ont des **durées et des rythmes différents.**

Une solution très efficace consiste en un **automate de Markov** qui va mettre en correspondance optimale les paramètres des deux mots. On démontre que cette méthode fournit la solution optimale du problème dans le chapitre suivant.

I.5. Conclusion

Dans ce chapitre nous avons présenté l’historique et les principales notions élémentaires qui constituent le traitement automatique du signal vocal en relation avec les domaines aussi divers que la reconnaissance, nous avons aussi décrit les différentes phases de reconnaissance.

CHAPITRE II

TECHNIQUES D'ANALYSE DU SIGNAL VOCAL

II.1. Introduction

La parole apparaît comme une variation de la pression de l'air dans l'appareil phonatoire humain [28]. Les différents traits acoustiques du signal de parole sont notamment : sa fréquence fondamentale, son énergie, son spectre. . . Chacun de ces éléments étant lui-même intimement lié à une grandeur perceptible : pitch, intensité, timbre... etc.

II.2. Paramétrisation du signal

L'objectif d'un système de paramétrisation est d'extraire les informations caractéristiques du signal de parole en éliminant au maximum les parties redondantes. Un tel système prend un signal en entrée et retourne un vecteur de paramètres (appelé indifféremment vecteur acoustique ou encore vecteur d'observations). Les vecteurs de paramètres doivent être pertinents (précis, de taille restreinte et sans redondance), discriminants (pour faciliter la reconnaissance) et robustes (aux différents bruits et/ou locuteurs).

Il existe un certain nombre d'approches pour la paramétrisation. Nous présentons ici celles utilisées le plus couramment dans la littérature :

- ✓ Paramétrisation basée sur un modèle de production de la parole, par exemple **LPC** (Linear Predictive Coefficients) [28], [29];
- ✓ Paramétrisation basée sur une analyse dans le domaine cepstral, par exemple :
  - **LPCC** (Linear Predictive Cepstral Coefficients) [30];
  - **MFCC** (Mel Frequency Cepstral Coefficient) [31];
  - **LFCC** (Linear Frequency Cepstral Coefficient)[83];
  - **PLP** (Perceptual Linear Predictive) [32];
  - **PLP-RASTA** (RelAtive SpecTrAl) [36,37] ;
  - Analyse par Ondelettes [81] ;

Il existe de nombreuses références qui comparent les systèmes de paramétrisation [40,41,42]. Au-delà des principales approches pour la paramétrisation que nous venons de présenter, d'autres peuvent être trouvées dans la littérature telles que **NPC**(Neural Predictive Coding) [43], **LSF** [44], **ZCR** (Zero Crossing Rate) [39]...etc.

II.2.1. Paramétrisation basée sur une analyse dans le domaine cepstral

Comme nous l'avons précisé ci-dessous, le signal de parole S_n est le résultat de la convolution entre un signal excitateur g_n (la glotte) et le conduit vocal b_n :

$$s_n = g_n * b_n \tag{II.1}$$

Le passage, par homomorphisme, dans un domaine où l'opérateur de convolution est transformé en opérateur d'addition permet de décorréler les contributions de la source et du conduit du signal de parole [87]. En pratique, l'utilisation de la transformée de Fourier donne les coefficients cepstraux :

$$\widetilde{s_n} = \widetilde{g_n} \text{ x } \widetilde{b_n} \tag{II.2}$$

Où $\widetilde{g_n}$et $\widetilde{b_n}$ sont les transposées dans le domaine fréquentiel de g_n et b_n.

Plusieurs méthodes permettent d'obtenir des coefficients cepstraux :

- ✓ Grâce à une récursion depuis les coefficients LPC, ce qui donne les coefficients LPCC;
- ✓ Par l'utilisation d'une FFT et d'une FFT inverse ; cette technique permet de calculer les coefficients MFCC, LFCC et PLP.

Le calcul des coefficients cepstraux (MFCC, LFCC et PLP) est souvent précédé d'une phase de préaccentuation du signal, suivie d'un fenêtrage.

II.2.1.1. La préaccentuation

Pourquoi filtrer un signal si on vaut récupérer toutes les informations ?

Il existe deux explications pour l'utilisation du module de préaccentuation [84]. Pour la première, la partie voisée du signal de la parole présente une accentuation spectrale approximative de (-20) dB par décade. Le filtre de préaccentuation permet de compenser cette accentuation avant d'analyser le spectre, ce qui améliore cette analyse. La deuxième considère que l'audition est plus sensible dans la région du spectre autour des 1 KHz. Le filtre de préaccentuation va donc amplifier cette région centrale du spectre [85].

En général le filtre de préaccentuation est de la forme :

$$x_i^{'} = x_i - \alpha * x_{i-1} \tag{II.3}$$

Avec ***0.90≤ α ≤1*** (La valeur classique de α est 0,97).

II.2.1.2. Le fenêtrage

Le découpage du signal en trames produit des discontinuités aux frontières des trames, qui se manifestent par des lobes secondaires dans le spectre ; ces effets parasites sont réduits en appliquant aux échantillons de la trame une fenêtre de pondération comme par exemple la fenêtre de Hamming [86]:

$$x_i^{''} = x_i^{'} * ham_i \tag{II.4}$$

$$\text{avec } ham_i = 0.54 - 0.46.\cos(\frac{2 * \pi * i}{N}) \tag{II.5}$$

Où N correspond à la longueur de la fenêtre.

D'autres fenêtrages sont parfois appliqués : fenêtrage de Hanning, fenêtrage de Blackman, fenêtrage de Kaiser (Chapitre I.3).

II.2.1.3. Paramètres MFCC

Afin de rapprocher l'analyse en banc de filtres de la perception humaine, les filtres ne sont généralement pas répartis de manière linéaire, mais en fonction

d'une échelle Mel. La correspondance entre une fréquence en **Hertz** et en **Mel** se calculé de la manière suivante :

$$F_{mel} = 2596 * \log(1 + \frac{F_{Hz}}{700}) \quad \text{(II.6)}$$

Intuitivement, cela revient à utiliser une échelle linéaire en basse fréquence, puis logarithmique en haute fréquence.

La chaîne complète de calculs des coefficients MFCC est définie par la figure.II.1

Généralement, seuls les 12 premiers coefficients cepstraux sont conservés et une vingtaine de filtres sont utilisés pour l'analyse en banc de filtres.

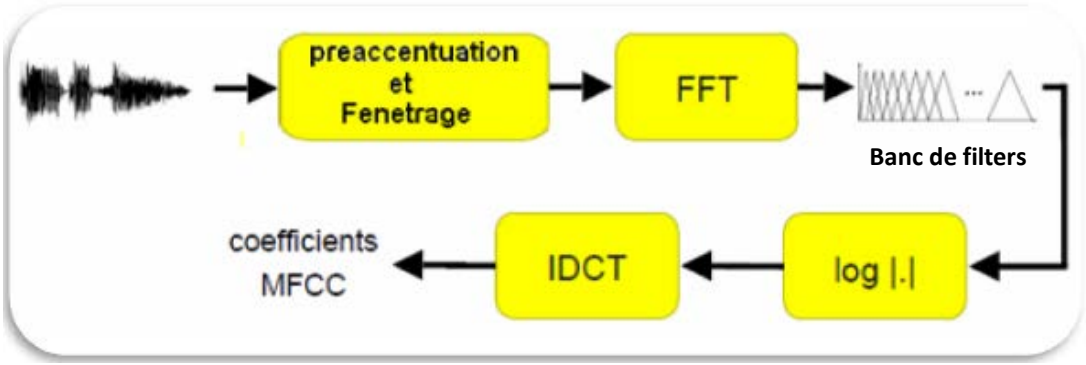


Fig.II.1 : *Chaîne de traitement pour obtenir les coefficients MFCC*

II.2.1.4. Calcul d'énergie

Généralement, l'énergie du signal est utilisée en complément des coefficients issus d'une paramétrisation basée sur une analyse dans le domaine cepstral. L'énergie correspond à la puissance du signal.

$$E_n = \sum_{n=0}^{N-1} s_n^2 \quad \text{(II.7)}$$

Le calcul de l'énergie se fait généralement sur des fenêtres glissantes de 20 ms avec un décalage de 10 ms (soient une valeur toutes les 10 ms de signal).

II.2.2. Post traitement

Nous allons détailler deux compléments communément admis :

- ✓ La suppression de la moyenne cepstral avec réduction de la variance ;
- ✓ Et les paramètres dynamiques (dérivées premières et secondes).

II.2.2.1. Suppression de la moyenne cepstral (CMS)

Généralement, après l'étape de paramétrisation proprement dite, une normalisation des paramètres est effectuée afin de rendre ces paramètres plus robustes au bruit ou au changement de canal. Cela revient généralement à soustraire la moyenne cepstrale puis à réduire la variance.

La soustraction de la moyenne cepstrale et la normalisation de la variance sont réalisées soit sur l'intégralité du fichier à décoder soit sur une fenêtre glissante.

Grâce à cette normalisation des travaux on été effectués dans [45] et [46] montrent une amélioration non négligeable des performances.

II.2.2.2. Dérivées de premier et second ordre Δ et ΔΔ

Pour enrichir la paramétrisation, les dérivées de premier et second ordre sont souvent utilisées. Cela permet d'ajouter de l'information concernant la dynamique du signal. Les coefficients Δ (dérivées du premier ordre) sont souvent estimés grâce au développement limité d'ordre 2 :

$$c_i'(t) = \frac{c_i(t+1) - c_i(t-1) + 2(c_i(t+2) - c_i(t-2))}{10} \qquad \text{(II.8)}$$

Où $c_i(t)$ correspond au $i^{ème}$ coefficient pour la trame t et c'_i(t) sa dérivée.

Les **ΔΔ** (coefficients du second ordre) sont estimés de la même manière à partir des coefficients du premier ordre.

II.3. Modèles de reconnaissance de la parole

Les premières recherches en reconnaissance automatique de la parole ont débuté à partir des années 1950. Dans les années 1960, une méthode appelée déformation temporelle dynamique (Dynamic Time Warping DTW) est apparue. Elle repose sur les travaux de Bellman 1957 [47] et reste aujourd'hui une approche importante en reconnaissance de mots isolés.

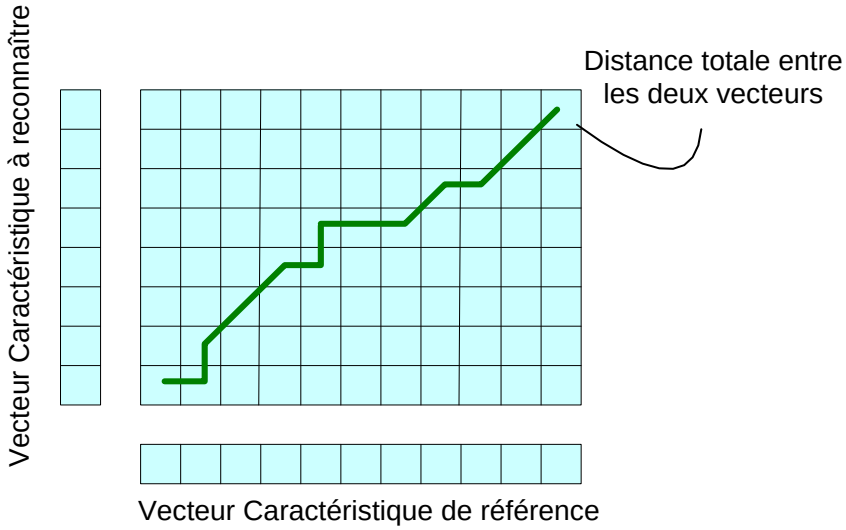


Fig.II.2 : *Comparaison élastique entre deux vecteurs caractéristiques*

Cette méthode a l'avantage d'être simple et ne nécessite que peu de données d'apprentissage. Par contre, elle est limitée par la taille du vocabulaire à reconnaître et à la reconnaissance mono-locuteur.

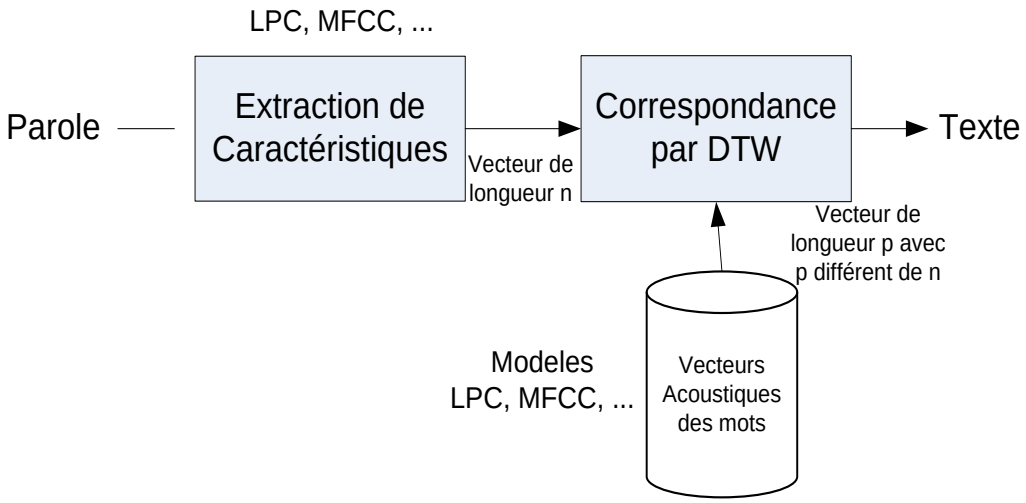


Fig.II.3 : Schéma d'un système de reconnaissance basé *sur la comparaison dynamique*

Une seconde méthode a émergé dans les années 1975, avec les travaux de Jelinek 1976[48]. Elle s'appuie sur l'utilisation des modèles de Markov cachés (Hidden Markov Models - HMM). Elle a permis de nombreuses avancées dans les domaines de la reconnaissance de la parole continue et de la reconnaissance multilocuteurs, domaines dans lesquels la DTW était peu probante.

Puis, à la fin des années 80, les réseaux de neurones sont venus offrir une nouvelle voie pour le traitement automatique de la parole [49]. Plusieurs types de réseaux de neurones coexistent :

1- Réseau de neurones multicouche ;

2- Réseau Multicouche à retard ;

3- Classifieur dynamique [56] ;

4- Réseau Récurrent [57] ;

5- Classification non supervisée : Extraction de caractéristiques

Mais, pour la reconnaissance de mots isolés, les approches les plus classiques sont le MLP (Perceptron Multi layer) et les TDNN (Time Delay Neural Network) [50] ou [51].

Il existe aussi des classifieurs binaires et statiques appelées Machine à vecteur Support (SVM) et Kernel Machines. Ils permettent de créer une surface de décision entre deux classes définies dans un même espace. Pour cela, ils construisent une frontière de décision par projection des caractéristiques provenant d'un espace d'origine dans un espace de caractéristiques de dimension supérieure (voir infini) dans le but de rendre les classes linéairement séparables [68], [59],[58],[60],[62],[63],[64].

Aussi, il y'a les systèmes experts qui sont des systèmes cherchant à reproduire l'analyse faite par des experts humains (notamment des phonéticiens dans le cadre de la reconnaissance de la parole). Ces experts procèdent généralement en deux phases : une analyse visuelle du spectrogramme suivie d'un raisonnement contextuel avec les indices notés lors de la première phase. L'objectif principal de ces systèmes est de réaliser des raisonnements logiques comparables à ceux que feraient des experts humains de ce domaine [65].

II.3.1. Modèle de Markov caché (Hidden Markov Model HMM)

C'est l'une des méthodes les plus utilisées en reconnaissance de la parole. Ce modèle combine les avantages d'une machine à état et des distributions de probabilités.

L'idée principale est de décomposer un mot en une suite de sous-unités lexicales (généralement les phonèmes) qui sont représentées par des modèles de Markov cachés.

La notation utilisée ici correspond à celle utilisée par L. Rabiner dans [52].

II.3.1.1. Principe général

La reconnaissance consiste à retrouver le mot $\tilde{m}$ parmi un ensemble de mots possibles *M (M = {m_i})*, en fonction d'une suite d'observations O (suite de vecteurs acoustiques). Cette recherche est faite par maximisation de la probabilité d'émission d'un mot en fonction de la suite d'observations :

$$\tilde{m} = Arg \max_{m} P(m/O) \tag{II.9}$$

D'après le théorème de Bayes :

$$P(m/O) = \frac{P(O/m)P(m)}{P(O)} \tag{II.10}$$

P(O) étant indépendant de *m*, on obtient :

$$\tilde{m} = Arg \max_{m} P(O/m)P(m) \tag{II.11}$$

Avec *P(m)* la probabilité d'apparition *a priori* du mot *m*, et *P(O/m)* la probabilité *a posteriori* d'émission de la séquence *O* sachant le mot *m*.

II.3.1.2. Formalisme Markovien

Un modèle de Markov (illustré par la figure II.4) est un automate fini qui change d'état à chaque unité de temps. De manière générale, on se limite aux HMM d'ordre 1, ce qui sous-entend que la possibilité d'être dans un état e_j à l'instant $t+1$ ne dépend que de l'état e_i dans lequel le système se trouvait à l'instant t (d'autres modèles existent voir [53] qui propose des HMM d'ordre 2).

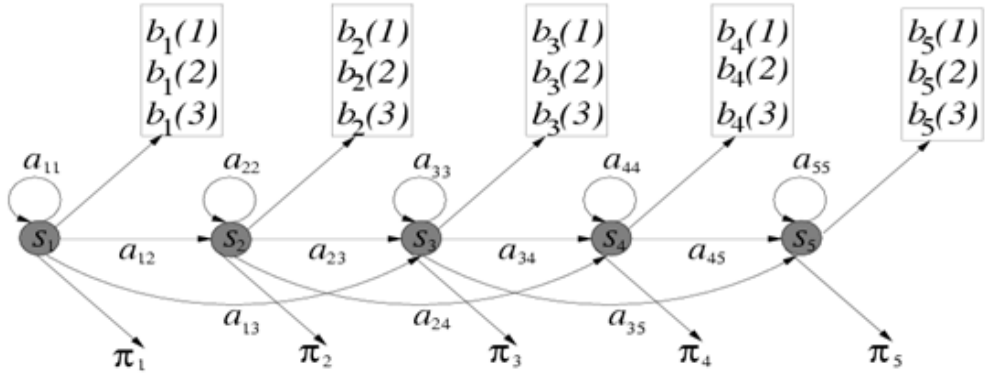


Fig.II.4 : *Exemple d'un HMM*

De tels automates sont utilisés pour modéliser les sous-unités lexicales. Chaque unité est représentée par un HMM à n états, généralement trois. A ces trois états, on ajoute un état de début d et un état de fin f qui permettront l'enchaînement des HMM.

Des HMM modélisant des mots entiers ont également été utilisés [54].

Chaque état est caractérisé par une fonction de densité de probabilité. Très souvent, ces fonctions sont des mixtures de gaussiennes (Gaussian Mixture Model : GMM). Les liaisons inter-états sont caractérisées par une matrice de transitions qui définit la probabilité de se déplacer d'un état i à un état j. Un HMM λ est donc caractérisé par :

- E : $E = \{1, 2, ..., N\}$, l'ensemble des N états du modèle ;
- A : $A = \{a_{ij}$, avec $1 \leq i, j \leq N\}$, la matrice de transition inter-états ;
- B : $B = \{bi$, avec $1 \leq i \leq N\}$, l'ensemble des fonctions de densité de probabilité associées à chacun des états.

La mise en œuvre d'un système Markovien implique la résolution de trois problèmes :

- ✓ ***L'estimation de la probabilité d'une séquence d'observations****:* pour une suite d'observations O et un HMM λ, quelle est la probabilité P(O| λ) ?
- ✓ ***Le décodage d'une séquence d'observations*** *:* pour une suite d'observations O et un HMM λ, quelle est la séquence d'états qui correspond aux observations ?
- ✓ ***L'apprentissage des paramètres du modèle*** *:* comment estimer les paramètres λ ?

Les réponses à ces trois questions est proposée dans les trois paragraphes suivants.

II.3.1.2.1. Estimation de la probabilité d'une séquence d'observations

La séquence d'observations $O = \{o_1, o_2, o_3, ..., o_t\}$, passant par la suite d'états $Q= \{q_0, q_1, q_2, ..., q_t\}$ aura pour probabilité :

$$P(O/Q,\lambda) = \prod_{t=1}^{T} P(O_t/q_t,\lambda) \qquad \text{(II.12)}$$

$$= b_{q1}(o_1)b_{q2}(o_2)b_{q3}(o_3) \dots b_{qT}(o_T)$$

En supposant que les observations sont statistiquement indépendantes. La probabilité du chemin peut, elle, être définie par :

$$P(Q/\lambda) = \pi_{q_1} \prod_{t=2}^{T} a_{q_{t-1}q_t} \qquad \text{(II.13)}$$

$$= \pi_{q_1} a_{q_1q_2} a_{q_2q_3} \dots a_{q_{T-1}q_T}$$

La probabilité conjointe des observations *O* et du chemin *Q* est donc :

$$P(O,Q/\lambda) = P(O/Q,\lambda)P(Q/\lambda) \qquad \text{(II.14)}$$

Pour l'ensemble des chemins, cela donne :

$$P(O/\lambda) = \sum_{Q} P(O/Q,\lambda)P(Q/\lambda) \quad \text{(II.15)}$$

$$= \sum_{q_1,q_2,\ldots,q_n} \pi_{q_1} \prod_{t=1}^{T} a_{q_{t-1}q_t} \quad \text{(II.16)}$$

La complexité de ce calcul est très importante, de l'ordre de $2^T * N^T$ opérations avec *T* correspondant au nombre d'observations et *N* au nombre d'états [87].

L'utilisation de l'algorithme ***Forward***, de programmation dynamique, permet de réduire ce coût de calcul tout en conservant une solution exacte. Cet algorithme comporte trois étapes :

1. initialisation :

$$\alpha_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N \quad \text{(II.17)}$$

2. itération :

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^{N} \alpha_t(i) a_{ij} \qquad 1 \le j \le N \; et \; 1 \le t \le T-1 \quad \text{(II.18)}$$

3. conclusion :

$$P(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad \text{(II.19)}$$

avec $\alpha_t(i)$ qui correspond à la probabilité conjointe d'observer la séquence $o_1,o_2,\ldots o_t$ et l'état *i* au temps *t*. Cet algorithme est d'une complexité nettement inférieure, de l'ordre de N^2T.

Il possède un pendant appelé ***backward*** qui parcourt dans l'ordre inverse. Il comporte aussi trois étapes et peut être décrit de la manière suivante :

1. initialisation :

$$\beta_T(i) = 1 \qquad 1 \leq i \leq N \tag{II.20}$$

2. itération :

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j) \quad 1 \leq i \leq N \;\; et\; t = T-1, T-2, \ldots, 1 \tag{II.21}$$

3. conclusion :

$$P(O/\lambda) = \sum_{i=1}^{N} \pi_i b_i(o_1)\, \beta_1(i) \tag{II.22}$$

II.3.1.2.2. Décodage d'une séquence d'observations

L'objectif est ici, connaissant la séquence d'observations $O = o_1, o_2, ..., o_T$ et les paramètres λ du HMM, de déterminer quel est le chemin, $S = q_1, q_2, ..., q_T$, le plus probable :

$$\tilde{S} = Arg \max_{S} P(O, S/\lambda) \tag{II.23}$$

De la même manière que pour l'évaluation de $P(O|\ \lambda)$, le test de l'ensemble des chemins possibles nécessite de l'ordre de N^T opérations. Un autre algorithme de programmation dynamique nous permet de réduire cette complexité : l'algorithme de **Viterbi** [55]. L'idée principale est d'utiliser la probabilité ($\delta_t(i)$) d'être dans l'état *i,* à l'instant *t*, pour la séquence $o_1, o_2, ..., o_t$ pour estimer la probabilité d'être dans l'état *j* avec la prochaine trame (o_{t+1}). Cette probabilité correspond au produit de $b_j(o_{t+1})$ par le max sur *i* des $\delta_t(i))a_{ij}$.

Cet algorithme récurrent est défini comme suit :

1. initialisation :

$$t = 0$$
$$r_0(m) = \pi_i b_i(o_1) \quad \text{(II.24)}$$

2. récurrence :

$$r_t(j) = \max_i r_{t-1}(i)\, a_{ij}\, b_j(o_t) \quad \text{(II.25)}$$

3. décision :

$$etat\ final = \max_i r_T(i) \quad \text{(II.26)}$$

$r_t(j)$ correspond à la probabilité maximale pour que les observations $O=o_1,o_2,....,o_t$ aient été émises par λ en suivant un chemin arrivant en j.

L'ajout d'une variable de mémorisation du chemin parcouru (suite d'états) permet, par un retour arrière, de déterminer également la séquence d'état.

II.3.1.2.3. Apprentissage des paramètres du modèle

L'apprentissage des paramètres du modèle est certainement le problème le plus complexe des trois. Ces paramètres à estimer sont les transitions entre états (a_{ij}) et les probabilités d'émissions des états ($b_i(o)$). La topologie du modèle (nombre d'états des modèles, la possibilité ou non de passer d'un état à l'autre et enfin l'alphabet des symboles) ne fait pas partie des paramètres estimés, elle est, généralement, définie *a priori*.

Classiquement, la méthode utilisée pour la ré-estimation des paramètres λ cherche à maximiser la vraisemblance (critère du maximum de vraisemblance aussi appelé MLE pour Maximum Likelihood Estimation). L'objectif est donc, possédant une certaine quantité de données O, de ré-estimer les paramètres λ tels que :

$$\tilde{\lambda} = Arg \max_{\lambda} \prod_k P(O_k/\lambda) \quad \text{(II.27)}$$

La maximisation de la vraisemblance du critère MLE est obtenue par l'utilisation de l'algorithme ***Baum-Welch*** connu aussi sous le nom de ***forward-backward***. Cet algorithme itératif converge vers un optimum local ; cependant, le temps nécessaire à cette convergence dépend directement de l'initialisation des paramètres. Cet algorithme nous assure que :

$$P\left(O/\tilde{\lambda}_n\right) \leq P(O/\tilde{\lambda}_{n+1}) \tag{II.28}$$

Définissons la grandeur ξ :

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j/O, \lambda) \tag{II.29}$$

Qui correspond à la probabilité d'aller de l'état *i* à l'instant *t* à l'état *j* à l'instant *t + 1* sachant le modèle et les observations.

Et la grandeur :

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{II.30}$$

Qui correspond à la probabilité d'être dans l'état *i* à l'instant *t* sachant les observations *O* et les paramètres *λ*.

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j/O, \lambda)$$

$$= \frac{P(q_t = i, q_{t+1} = j, O/\lambda)}{P(O/\lambda)}$$

$$= \frac{P(q_t = i, o_1, o_2, \dots, o_t/\lambda) * a_{ij}\, b_j(o_{t+1}) * P(o_{t+2}, \dots, o_T/q_{t+1} = j, \lambda)}{P(O/\lambda)} \tag{II.31}$$

Notons que $P(q_t = i, o_1, o_2, \dots, o_t/\lambda)$ correspond à $\alpha_t(i)$, défini durant la présentation de l'algorithme ***forward***. $P(o_{t+2}, \dots, o_T/q_{t+1} = j, \lambda)$ correspond au paramètre $\beta_{t+1}(j)$ de l'algorithme ***backward***. Nous obtenons :

$$\xi_t(i,j) = \frac{\alpha_t(i) * a_{ij}\, b_j(o_{t+1}) * \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i) * a_{ij}\, b_j(o_{t+1}) * \beta_{t+1}(j)} \quad \text{(II.32)}$$

La probabilité de transition entre l'état i et l'état j est ré-estimée par :

$$\tilde{a}_{ij} = \frac{esp\text{érance du nombre de trasitions des états i vers j}}{esp\text{érance du nombre de passages en i}} \quad \text{(II.33)}$$

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \text{(II.34)}$$

Il faut noter que le terme a_{ij} présent au numérateur de l'équation (II.32) assure qu'une transition non autorisée au départ reste non autorisée : a_{ij} restant à 0.

La probabilité d'être dans l'état *i* au départ (π_i) correspond à :

$$\tilde{\pi}_i = \gamma_1(i) \quad \text{(II.35)}$$

La probabilité d'émission associée à un état est définie par :

$$\tilde{b}_j(k) = \frac{esp\text{érance du nombre d}'\text{émissionsdu symbole } \upsilon_k \text{ dans l}'\text{état i}}{esp\text{érance du nombre de passages en i}} \quad \text{(II.36)}$$

$$= \frac{\sum_{t=1, o_t=\upsilon_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad \text{(II.37)}$$

II.3.1.3. Utilisation des HMM pour la reconnaissance de la parole

Pour résumer brièvement, on construit un modèle de Markov Cachés pour tous les mots. Puis, on réalise l'estimation des paramètres du modèle qui maximisent la vraisemblance des vecteurs d'apprentissages pour un mot m.

Lorsque l'on veut reconnaître un mot inconnu, on applique l'algorithme forward-backward sur tous les modèles. Le modèle qui maximise la vraisemblance correspond au mot reconnu.

Dans le cadre de la reconnaissance de la parole de mots isolés, les modèles de Markov cachés sont utilisés pour représenter le signal acoustique. Les états peuvent représenter grossièrement un son (phonème). Dans un autre ordre d'idée, les états peuvent représenter les différentes versions de prononciation d'un mot.

Les transitions représentent les différentes possibilités d'enchaîner les sons. Cette intégration de la dimension temporelle dans le modèle explique pourquoi les chaînes de Markov Cachées sont souvent utilisées dans les systèmes de reconnaissance de la parole.

Pour un vocabulaire relativement petit, contenant environ 100 mots, pour représenter un mot le modèle de Markov caché possède entre 5 et 10 états et environ 40 observations [52].

Pour de grands vocabulaires, l'association d'un modèle de Markov Caché à un mot est impossible, car le nombre de modèles et le volume d'apprentissage seraient trop importants. C'est pourquoi la reconnaissance de grands vocabulaires est toujours effectuée à partir de modèles de Markov Cachés d'unité comme le phonème, les diphones ou les triphones. L'approche fréquemment utilisée consiste à prendre un modèle à 3 états par phonème, en faisant l'hypothèse que l'état du milieu modélise la partie stationnaire du phonème et les états extérieurs modélisent la coarticulation avec les phonèmes voisins. Le nombre de phonèmes choisis est généralement de l'ordre de 35 pour le français.

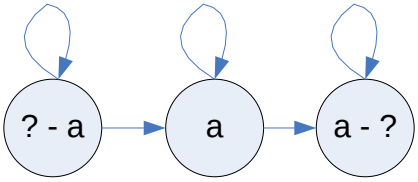


***Fig.II.5** : Représentation du phonème « a »*

Un autre modèle souvent utilisé est le triphone ou l'allophone. Il tient compte des phonèmes suivant et précédent. Le nombre d'allophones dans la langue française est d'environ 7500 et son apprentissage nécessite de grandes bases de données.

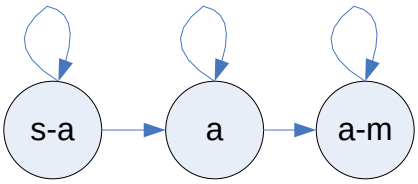


***Fig.II.6** : Représentation du triphone « s-a-m »*

La représentation d'un mot est alors obtenue par la concaténation de plusieurs modèles de Markov Cachés. Ainsi, un long apprentissage est évité et il est possible d'ajouter des nouveaux mots en considérant seulement la séquence de triphones.

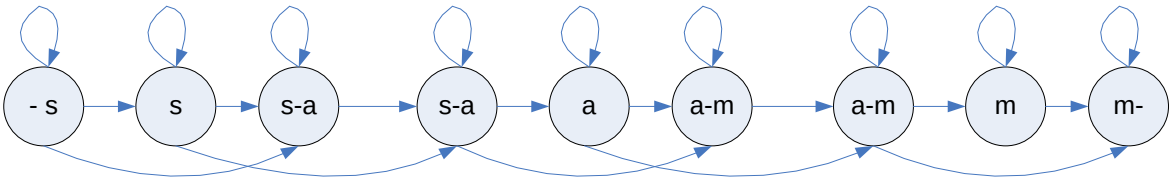


***Fig.II.7** : Représentation du mot « sam » par concaténation de phonèmes.*

II.3.2. Modèle de Mélange de lois gaussiennes

Les modèles de mélange de Gaussiennes font partis des méthodes de classification paramétrique globale. C'est-à-dire que la loi de distribution est supposée connue et l'ordre des données n'est pas significatif. Cette méthode de classification est un cas spécifique des modèles de Markov Cachés, c'est le modèle à un état.

II.3.2.1. Principe général

Elle repose sur le fait que l'on peut modéliser la distribution des données par une fonction de densité de probabilité en combinant plusieurs fonctions Gaussiennes.

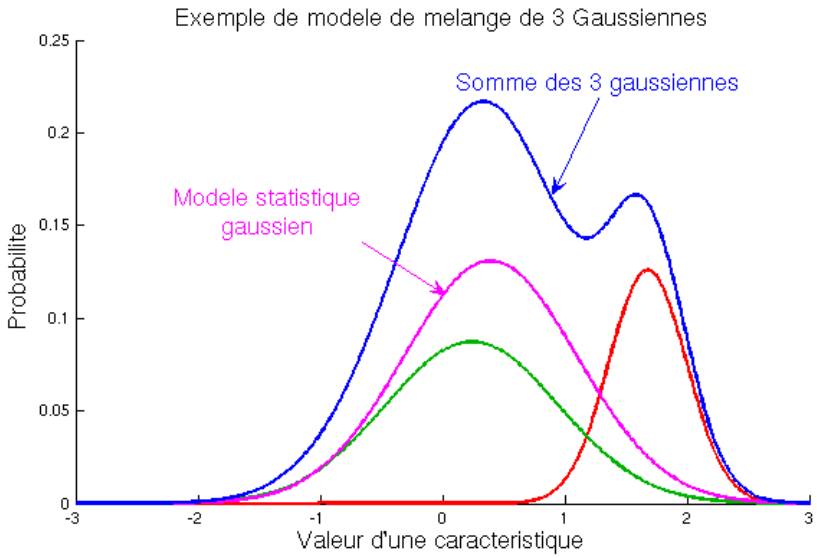


***Fig.II.8** : Modèle de mélange de Gaussiennes à 3 gaussiennes*

On considère que les données en entrées, les vecteurs de caractéristiques, sont supposées être des réalisations de variables aléatoires mutuellement indépendantes qui suivent la loi suivante :

$$f(x) = \sum_{i=1}^{M} \pi_i f_i(x) \qquad \text{(II.38)}$$

avec *M* : le nombre de composantes π_i, $i \in \{1...M\}$: les probabilités de chaque composante, et $f_i(x)$, $i \in \{1...M\}$: les densités de probabilités de chaque composante. Ainsi, on cherche à décomposer une fonction inconnue et *à priori* complexe f sur un ensemble de fonctions plus simples f_i.

On peut également décrire cette équation comme un modèle dans lequel on suppose que les données sont réparties aléatoirement (et indépendamment les

unes des autres) en *M* classes qui sont caractérisées par une distribution différente f_i.

Chaque composante f_i correspond à des lois normales multidimensionnelles. μ_i et Σ_i, $i \in \{1...M\}$, sont les vecteurs moyens et les matrices de covariances de chaque composante. La fonction f_i de densité de probabilité gaussienne multidimensionnelle est donnée par :

$$f_i(x) = \frac{1}{2\pi^{\frac{p}{2}}\sqrt{\det(\Sigma_i)}} \exp\left[-\frac{1}{2}(x-\mu_i)^t \, \Sigma_i^{-1}(x-\mu_i)\right] \qquad \text{(II.39)}$$

Où : p désigne la dimension des vecteurs.

L'estimation des paramètres du modèle se fait généralement au sens du maximum de vraisemblance. On utilise pour cela l'algorithme EM (Expectation-Maximisation).

II.3.2.2. Utilisation des GMM pour la reconnaissance de la parole

L'utilisation d'une telle méthode en reconnaissance de la parole se justifie par le fait que l'on peut répartir les vecteurs de caractéristiques en plusieurs classes qui constituent le mélange (son voisé, non voisé, ou plus finement en fonction du phonème).

Elle est utilisée dans les modèles de Markov Cachés dans le cas d'observations continues. Le modèle de Mélange de Gaussiennes est utilisé pour représenter le vecteur caractéristique à l'intérieur d'un état. La probabilité d'émission d'une observation *O* au sein d'un état *j* devient donc :

$$b_j(O) = f(x) = \sum_{i=1}^{M} \pi_i f_i(x) \qquad \text{(II.40)}$$

Concrètement, on calcule le modèle de Mélange de Gaussiennes pour chaque coefficient des vecteurs caractéristiques associés à un état. Par exemple,

le modèle de Markov Caché dans le cas d'observations continues en considérant qu'un seul coefficient peut être modélisé de cette manière :

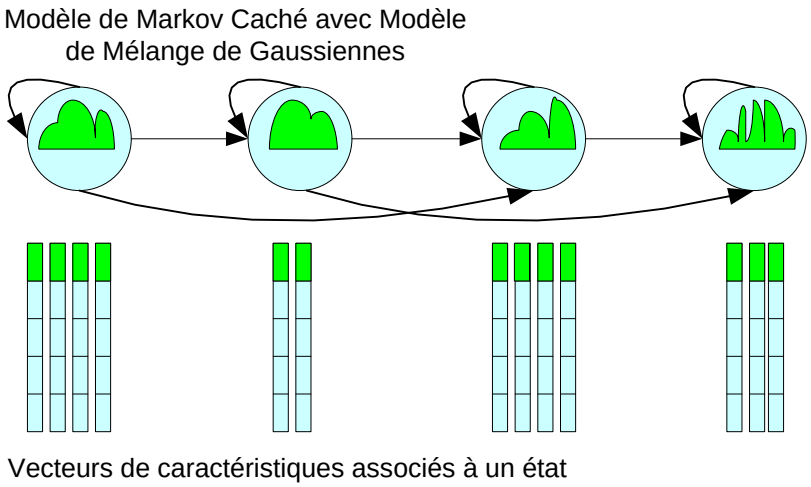


***Fig.II.9** : Modèle de Markov Caché en cas d'observations continues*

II.4. Modélisation du lexique

Les SRAP dont le système de décodage est basé sur la DTW ou sur les HMM nécessitent de posséder des modèles acoustiques. Ces modèles peuvent être des modèles d'unités sous-lexicales (phonèmes, diphones, triphones, pentaphones...) ou des modèles de mots directement.

L'inconvénient majeur de l'utilisation de modèles de mot est la complexité engendrée par l'ajout d'un mot dans le lexique. Cette approche est généralement utilisée avec un système DTW. Il convient alors de faire un alignement entre la référence et le test. La référence est le mot complet (non découpé en sous-unités). A l'inverse, les approches se servant des HMM utilisent généralement une décomposition d'un mot comme une suite de phonèmes.

Les modèles de mots sont utilisés le plus souvent dans le cadre d'application ayant un lexique non évolutif et composé de peu d'entrées. La très grande majorité des systèmes de reconnaissance de parole continue grand vocabulaire utilise des sous-unités lexicales. En effet, ces systèmes ont un lexique comportant plus de 65 000 entrées, il est donc impossible de faire prononcer toutes les entrées du lexique par un utilisateur. De plus, le partage des sous-unités lexicales entre les différentes entrées du lexique permet aussi de limiter le

nombre d'unités de base et d'augmenter, pour chaque unité, la quantité de données d'apprentissage.

II.5. Conclusion

L'extraction des paramètres acoustiques est une étape très importante dans les systèmes de reconnaissance automatique des mots isolés. Son but essentiel est d'extraire les données pertinentes à l'étape de modélisation statistique, et minimise ainsi les données redondantes et le bruit qui se présentent dans les signaux vocaux. Plusieurs expériences ont montré que les paramètres **LPCC** et **MFCC** donnent de meilleures performances aux systèmes d'identification ou de reconnaissance. Il est à noter qu'il existe d'autres paramètres acoustiques qui sont cités dans ce chapitre tels que : les paramètres **PLP, PLP-RASTA**, etc., mais dans notre travail, nous nous intéressons de plus par les paramètres **MFCC**.

Il existe plusieurs méthodes utilisées pour la reconnaissance automatique de la parole telles que la **DTW**, les réseaux de neurones **RN**, les **GMM** et les **HMM**, cette dernière est la plus utilisée et la plus efficace dans ce domaine.

CHAPITRE III
DETECTION D'ACTIVITE VOCALE EN TEMPS REEL

III.1. Introduction

La détection d'activité vocale est un élément important pour la reconnaissance vocale en temps réel, l'échec de la reconnaissance dépend essentiellement de l'implémentation d'un élément appelé VAD (*Voice Activity Detection*) celle-ci opère on isolant le signal utile (le signal de la parole) du bruit de fond en temps réel, sachant que ce dernier peu changé constamment.

Lorsqu'on parle, on a une variation de l'énergie du signal dans le domaine temporel comme il est montré sur ces deux figures **III.1** et **III.2.**

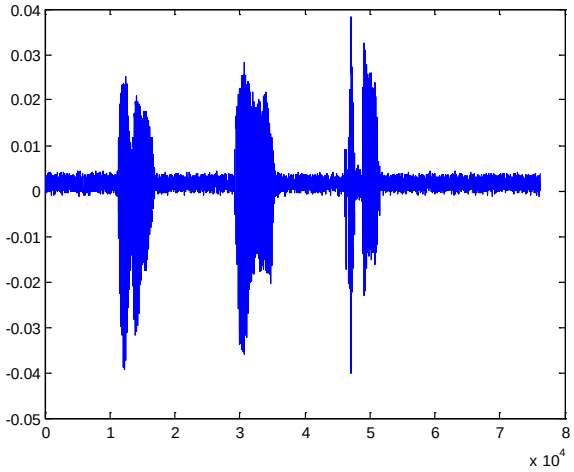


Fig. III.1 : *Bruit de fond + signal vocal*

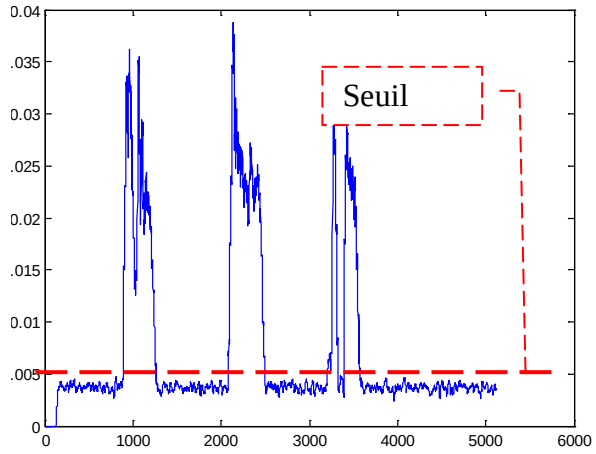


Fig. III.2 : *Énergie du signal*

Pour isoler la bande d'énergie utile représentant le mot, la première chose qui vient à l'esprit c'est de fixer un seuil pour l'énergie, dès que l'énergie du signal dépasse ce seuil, on a une Détection d'Activité Vocale en temps réel comme c'est présenté sur la figure **III.3**

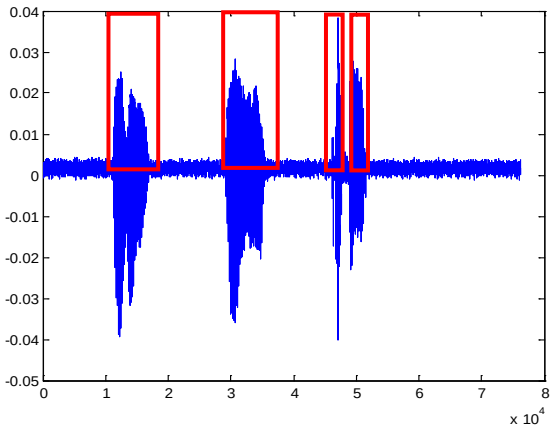
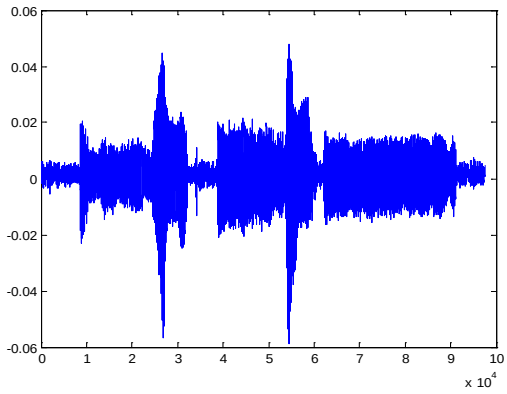


Fig. III.3 : *Fenêtrage de la parole par un seuil énergétique*

Ce qui n'est pas le cas comme pour la figure **III.5** on ne peut pas trouver des repères sur ce spectre énergétique



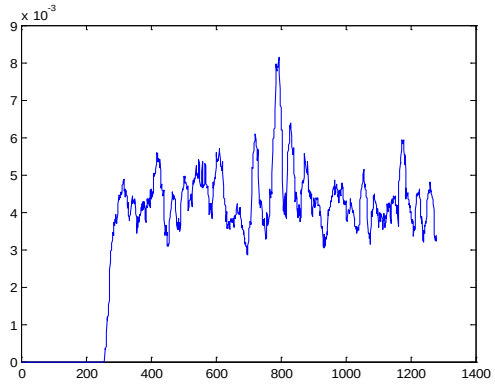


Fig. III.4 : *Bruit de fond + Parole*

Fig. III.5 : *Énergie du signal*

III.2. Détection d'activité vocale en adaptant le spectre de bruit

Pour simplifier, cette méthode se décompose en deux parties, une partie pour l'estimation statistique et l'autre pour la prise de la décision **Fig. III.6**

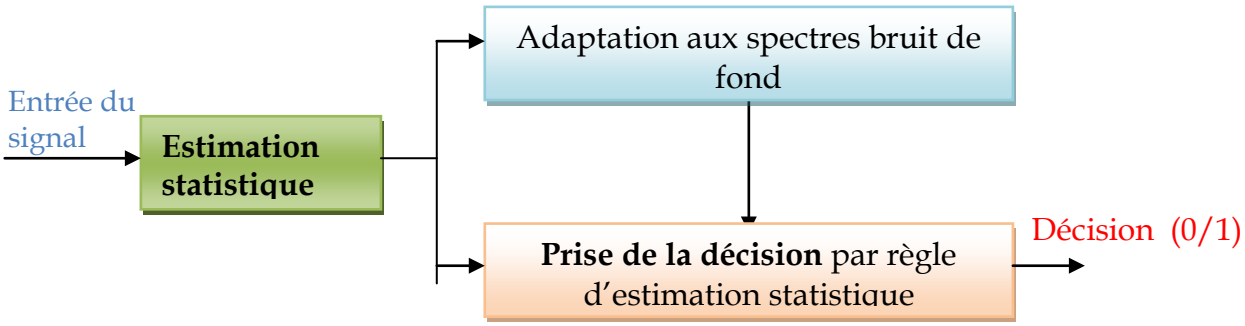


Fig. III.6 : *Déroulement global de l'algorithme de détection*

L'estimation statistique : elle a pour but de prédire bruit de fond en calculant sa moyenne par l'espérance mathématique.

Lorsqu'on prend des échantillons du signal, **la prise de décision** va estimer si ça correspond ou pas à notre espérance, dans le cas inverse c'est de la parole.

Et tout cela doit se faire dans le domaine fréquentiel pour avoir une meilleure sensibilité.

Déroulement de la détection d'hypothèse:

On va poser deux hypothèses, H_0 le bruit de fond et H_1 le bruit de fond plus parole :

Hypothèse 0 : $(H_0) : X(t)=N(t)$ (III.1)

Hypothèse 1 : $(H_1) : X(t)=N(t)+S(t)$ (III.2)

Où (N : Bruit) (S : Signal vocal)

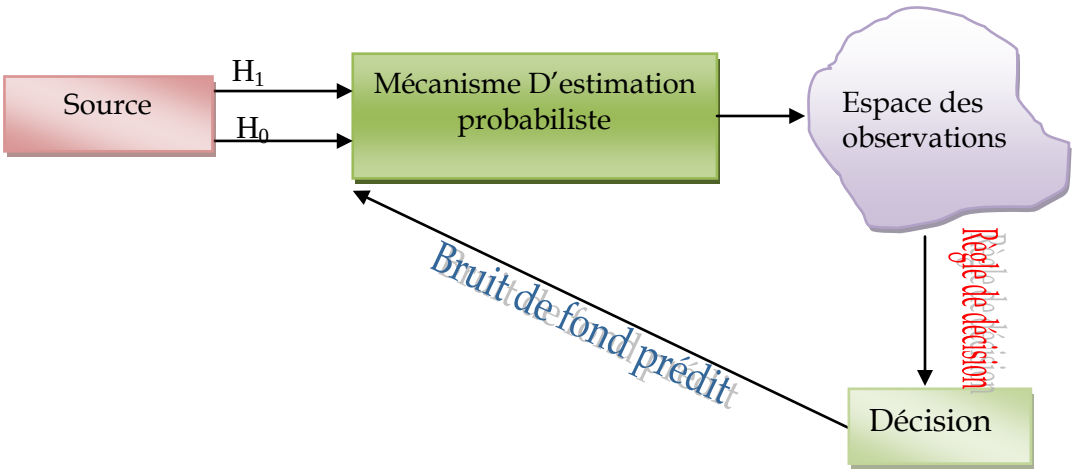


Fig. III.7 : *Déroulement de la détection d'hypothèse*

- Le **Mécanisme d'estimation probabiliste** va être estimé à l'aide du ***théorème de Bayes*** qui utilise la probabilité a priori pour estimer une hypothèse par rapport à une autre en connaissant l'une des deux hypothèses.
- L'**Espace des observations** va permettre de mieux calibrer le système précédant on calculant l'estimation maximum de probabilité en d'autres termes les meilleures approches de la variance du signal.
- La **Règle de décision** va être calculée par la vraisemblance.
- La **prédiction du bruit de fond** que l'on pourra estimer par l'espérance mathématique.

III.3. Mécanisme d'estimation probabiliste

Lors du démarrage du programme de la VAD dans notre application les premiers échantillons vont représenter le bruit de fond ce qui sera utilisé comme donnée initiale pour l'application du théorème de Bayes.

III.3.1. Le théorème de Bayes

En théorie des probabilités, le théorème de Bayes énonce des **probabilités conditionnelles** étant donné deux évènements *A* et *B*, le théorème de Bayes

permet de déterminer la probabilité de *A sachant B*, si l'on connaît les probabilités: de *A*, de *B*, de *B sachant A*.

Le théorème de Bayes permet d'écrire [69][70] :

$$P(A \mid \mathrm{B}) = \frac{\mathrm{P}(\mathrm{B} \mid \mathrm{A})\mathrm{P}(\mathrm{A})}{\mathrm{P}(\mathrm{B})} \tag{III.3}$$

Où
- P(A|B) : Loi a posteriori de A sachant B
- P(B|A) : fonction de vraisemblance de A
- P(A) : Loi a priori de A
- P(B) : Évidence (constante de normalisation)

Plus généralement, si $\{A_i\}$ est une partition de l'ensemble des possibles.

$$P(A_i \mid \mathrm{B}) = \frac{\mathrm{P}(\mathrm{B} \mid \mathrm{A}_i)\mathrm{P}(\mathrm{A}_i)}{\sum_j \mathrm{P}(\mathrm{B} \mid \mathrm{A}_j)\mathrm{P}(\mathrm{A}_j)} \tag{III.4}$$

III.3.2. Application du théorème de Bayes aux échantillons du signal

Pour notre cas, à un moment donné on a un signal bruité (bruit de fond), et à un autre moment donné on a de la parole, ce qui donne deux Hypothèses dans le domaine temporel.

- Hypothèse 0 (H_0) : X (t)=N (t)
- Hypothèse 1 (H_1) : X (t)=N (t) +S (t)

Puisqu'on va travailler avec k Fenêtre (une Fenêtre contient plusieurs échantillons à la fois), la formule devient :

- Hypothèse 0 (H_0) : $X_k=N_k$ (III.5)
- Hypothèse 1 (H_1) : $X_k=N_k+S_k$ (III.6)

Est avec ***L*** échantillon dans une Fenêtre ***k*** :

On utilisant la transforme de fourrier discret on obtient [71] :

- Hypothèse 0 (H_0) : $X_k(j\omega)=N_k(j\omega)$ (III.7)
- Hypothèse 1 (H_1) : $X_k(j\omega)=N_k(j\omega)+S_k(j\omega)$ (III.8)

On va appliquer les deux hypothèses au théorème de Bayes et on va estimer la probabilité du bruit connaissant le signal.

$$P(\mathrm{H_0} \mid \mathrm{X_k}) = \frac{\mathrm{P(X_k \mid H_0).P(H_0)}}{\mathrm{P(X_k \mid H_0).P(H_0) + P(X_k \mid H_1).P(H_1)}} \tag{III.9}$$

La formule (III.9) peut fournir les formules (III.10), (III.11), (III.12)

$$P(\mathrm{H_0} \mid \mathrm{X_k}) = \frac{\mathrm{P(X_k \mid H_0)}}{\mathrm{P(X_k \mid H_0) + P(X_k \mid H_1).\frac{P(H_1)}{P(H_0)}}} \cdot \frac{\mathrm{P(H_0)}}{\mathrm{P(H_0)}}$$

$$P(\mathrm{H_0} \mid \mathrm{X_k}) = \frac{\mathrm{P(X_k \mid H_0)}}{\mathrm{P(X_k \mid H_0) + P(X_k \mid H_1).\frac{P(H_1)}{P(H_0)}}}$$

$$P(\mathrm{H_0} \mid \mathrm{X_k}) = \frac{1}{1 + \frac{\mathrm{P(X_k \mid H_1)}}{\mathrm{P(X_k \mid H_0)}} \cdot \frac{\mathrm{P(H_1)}}{\mathrm{P(H_0)}}} \cdot \frac{\mathrm{P(X_k \mid H_0)}}{\mathrm{P(X_k \mid H_0)}}$$

$$P(\mathrm{H_0} \mid \mathrm{X_k}) = \frac{1}{1 + \frac{\mathrm{P(X_k \mid H_1)}}{\mathrm{P(X_k \mid H_0)}} \cdot \frac{\mathrm{P(H_1)}}{\mathrm{P(H_0)}}} = \frac{1}{1 + \varepsilon.\Lambda(\mathrm{k})} \tag{III.10}$$

Avec

$$\varepsilon = \frac{\mathrm{P(H_1)}}{\mathrm{P(H_0)}} \tag{III.11}$$

$$\Lambda(\mathrm{k}) = \frac{\mathrm{P(X_k \mid H_1)}}{\mathrm{P(X_k \mid H_0)}} \tag{III.12}$$

On peut remarquer que $\varepsilon, \Lambda(\mathrm{k})$ représente le rapport du signal sur le bruit (**SNR :** Signal to Noise Ratio) [71].

$$SNR_k^{a\ priori}(\omega) = \frac{\left|\mathrm{X_k}(j\omega)\right|^2}{\left|\mathrm{N_k}(j\omega)\right|^2} = \varepsilon \tag{III.13}$$

$$SNR_k^{a\ posteriori}(\omega) = \frac{\left|S_k(j\omega)\right|^2}{\left|\mathrm{N_k}(j\omega)\right|^2} = \Lambda(k) \tag{III.14}$$

ε **:** Va symboliser SNR a priori et $\Lambda(k)$: Va symboliser SNR a posteriori, ce qui résulte des équations (III.11) et (III.12) appliquée au théorème de bayes, la présence de la parole dans le signal X.

$$P(H_1 \mid X_k) = \frac{\varepsilon.\Lambda(k)}{1+\varepsilon.\Lambda(k)} \qquad \text{(III.15)}$$

À noter que :

$\Lambda(k) = \frac{P(X_k \mid H_1)}{P(X_k \mid H_0)}$: Représente le ***rapport de vraisemblance*** (LRT: likelihood ratio)[72].

$$P(H_0 \mid X_k) = \frac{1}{1+\varepsilon.\Lambda(k)} \qquad \text{(III.16)}$$

$$P(H_1 \mid X_k) = \frac{\varepsilon.\Lambda(k)}{1+\varepsilon.\Lambda(k)} \qquad \text{(III.17)}$$

Le théorème de Bayes a deux cas, un cas discret est un cas continu [70] :

- Pour le cas discret, on travaille avec les classes C_k ;
- Pour le cas continu, on travaille avec la densité de probabilité.

Alors pour notre cas, $P(H_0 \mid X_k), P(H_1 \mid X_k)$ ce sont des densités de probabilité [71][73].

Mais avec quelle densité de probabilité ?

Il y a plusieurs lois, loi binomiale, loi normale, Loi log normale, loi gamma, loi bêta…etc. Notre signal suit une loi normale (**gaussienne**) [71][73].

Les **tests de normalité** permettent de vérifier si ces données réelles suivent une loi normale ou non. Les tests de normalité sont des cas particuliers des **tests d'adéquation** (ou **tests d'ajustement**, tests permettant de comparer des distributions), appliqués à une loi normale [74]

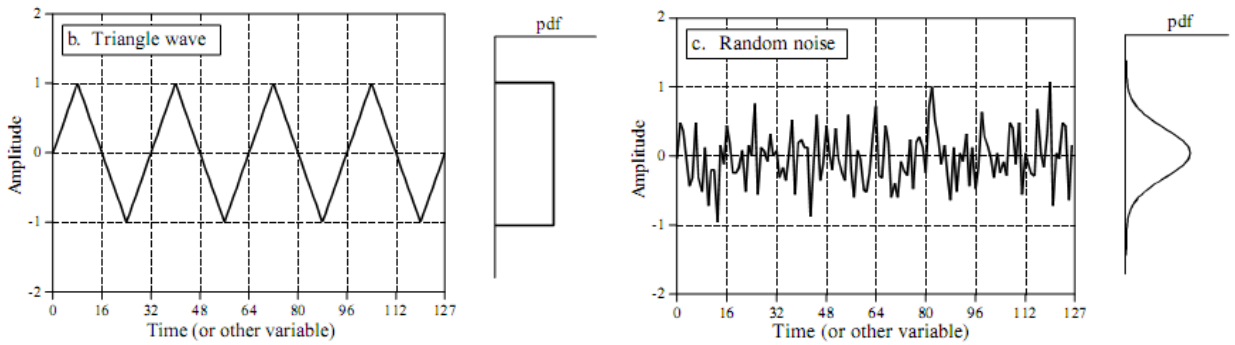


Fig.III.8 : *signal avec sa fonction de densité de probabilité (pdf)*

III.4. Espace des observations

Lorsqu'on prend plusieurs échantillons en sachant que c'est du bruit de fond, on va estimer leur variance qui représente le degré de dispersion autour de la moyenne. L'estimation maximale de la variance va donner les limites du bruit de fond qui va être appliqué à notre modèle statistique qui est la loi normale.

III.4.1. Aperçu de la loi normale

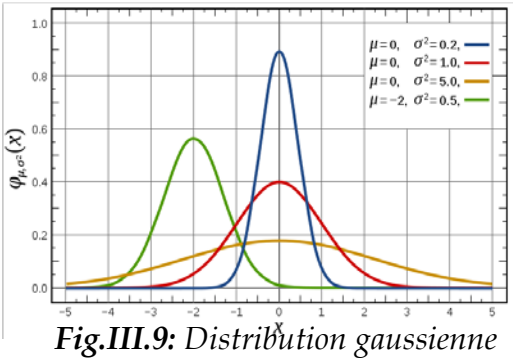


Fig.III.9: *Distribution gaussienne*

En probabilité, on dit qu'une variable aléatoire réelle *X* suit une **loi normale** (ou **loi normale gaussienne**, **loi de Laplace Gauss**) de moyenne μ et d'écart type σ strictement positif (donc de variance σ^2).

Si cette variable aléatoire réelle *X* admet pour densité de probabilité la fonction *p(x)* définie pour tout nombre réel *x*, par :

$$P(x) = \frac{1}{\sigma\sqrt{2.\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \qquad \text{(III.18)}$$

Une telle variable aléatoire est alors dite variable gaussienne. [75][76]

III.4.2. Application de la loi normale

En appliquant la loi normale aux deux hypothèses (H_0 et H_1) dans le domaine discret, nous obtenons les équations suivantes :

$$P(X \mid \mathrm{H}_0) = \prod_{k=0}^{L-1} \frac{1}{\pi.\lambda_N(k)} e^{-(\frac{|X_k|^2}{\lambda_N(k)})} \qquad \text{(III.19)}$$

$$P(X \mid \Theta, \mathrm{H}_1) = \prod_{k=0}^{L-1} \frac{1}{\pi.[\lambda_N(k) + \lambda_S(k)]} e^{-(\frac{|X_k|^2}{\lambda_N(k)+\lambda_S(k)})} \qquad \text{(III.20)}$$

Où $\Theta = \{\lambda_S(k) : k = 0,..L-1\}$ Est l'ensemble de paramètres inconnus conjoint à la densité de probabilité contenue dans l'hypothèse H_1.

$$P(X_1, X_2, ..., X_n \mid \Theta) = P(X_1 \mid \Theta).P(X_2 \mid \Theta)...P(X_n \mid \Theta) = \prod P(X_i \mid \Theta) \qquad \text{(III.21)}$$

Explication sur le paramètre inconnu conjoint à la densité de probabilité Θ

Ce paramètre est introduit pour paramétrer et calculer l'estimation maximum de probabilité $\widehat{\Theta}$ (Maximum Likelihood Estimation - **MLE**) C'est une méthode courant dans les statistiques, elle est utilisée pour ajuster un modèle statistique des données, et de fournir des estimations pour les paramètres du modèle, comme pour notre modèle Gaussienne il est paramétré sur la variance et la moyenne c'est deux valeurs doivent être fixes, et la seule variable étant X_k. L'estimation maximum de probabilité va permettre de mieux calibrer ces deux paramètres : la moyenne et la variance [77].

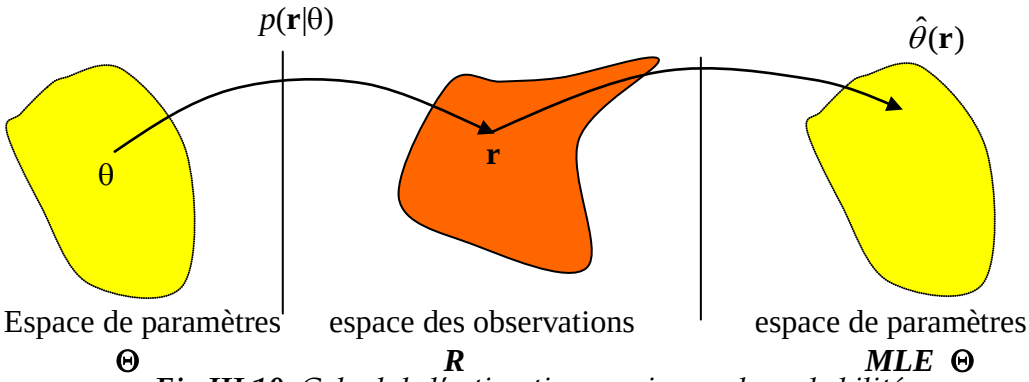


Fig.III.10: *Calcul de l'estimation maximum de probabilité*

III.4.3. Ajustement des paramètres du modèle (MLE)

L'estimation maximale représente le dérivé de la densité de probabilité par rapport à cette variable (variance), on va chercher l'estimation maximale du bruit de fond dans les échantillons, le Logarithme népérien est utilisé pour simplifier les calculs (Log Likelihood).

$$P(X \mid \Theta, \mathrm{H}_1) = \prod_{k=0}^{n-1} \frac{1}{\pi.\left[\lambda_N(k) + \lambda_S(k)\right]} e^{-\frac{|X_k|^2}{\lambda_N(k)+\lambda_S(k)}} \qquad \text{(III.22)}$$

$$P(X \mid \Theta, \mathrm{H}_1) = \frac{1}{(\pi.(\lambda_N(k) + \lambda_S(k)))^n} e^{-\sum \frac{|X_k|^2}{\lambda_N(k)+\lambda_S(k)}} \qquad \text{(III.23)}$$

$$\ln(P(X \mid \Theta, \mathrm{H}_1)) = \ln\left(\frac{1}{(\pi.(\lambda_N(k) + \lambda_S(k)))^n} e^{-\sum \frac{|X_k|^2}{\lambda_N(k)+\lambda_S(k)}}\right) \qquad \text{(III.24)}$$

Avec $$Ln(P(X \mid \Theta, \mathrm{H}_1)) = l(\theta) \qquad \text{(III.25)}$$

$$l(\theta) = -\ln(\pi.(\lambda_N(k) + \lambda_S(k)))^n + \ln\left(e^{-\sum \frac{|X_k|^2}{\lambda_N(k)+\lambda_S(k)}}\right) \qquad \text{(III.26)}$$

$$l(\theta) = -n.\ln \pi - n.\ln(\lambda_N(k) + \lambda_S(k)) - \sum \frac{|X_k|^2}{\lambda_N(k) + \lambda_S(k)} \quad \text{(III.27)}$$

$$\frac{\partial l(\theta)}{\partial \sigma_N{}^2} = 0 - n.(\frac{1}{\widehat{\lambda}_N(k) + \lambda_S(k)}) - \sum \frac{|X_k|^2}{(\widehat{\lambda}_N(k) + \lambda_S(k))^2} = 0 \quad \text{(III.28)}$$

$$-n.(\frac{1}{\lambda_S(k) + \widehat{\lambda}_N(k)}) = \sum \frac{|X_k|^2}{(\lambda_S(k) + \widehat{\lambda}_N(k))^2} \Rightarrow -n.(\lambda_S(k) + \widehat{\lambda}_N(k)) = \sum |X_k|^2 \quad \text{(III.29)}$$

$$\lambda_S(k) + \widehat{\lambda}_N(k) = \frac{\sum |X_k|^2}{n} = |X_k|^2 \quad \text{(III.30)}$$

$$\Rightarrow \widehat{\lambda}_N(k) = |X_k|^2 - \lambda_S(k) \quad \text{(III.31)}$$

Et $\widehat{\lambda}_N(k)$ est une estimation maximale de la variance (MLE) du bruit de fond.

Ce résultat va être utilisé pour le calcul du test d'hypothèse généralisé (vraisemblance généralisée) par conséquent réduit, car en utilise l'estimation maximale de la variance [77].

III.5. Règle de décision

Le rapport de vraisemblance généralisé est une approche à la résolution du problème de décision quand on ne peut pas construire un test uniformément le plus puissant.

Cette procédure consiste à associer à chaque hypothèse H_i une densité unique, parmi toutes les densités possibles, en prenant celle qui maximise la vraisemblance des données, et à l'utiliser pour construire un "rapport de vraisemblance réduit":

$$\Lambda_g(\mathbf{r}) = \frac{\max_{\theta} P(X \mid \widehat{\Theta}, \mathrm{H}_1)}{\max_{\theta} P(X \mid H_0)} \underset{H_0}{\overset{H_1}{\gtrless}} \gamma \quad \text{(III.32)}$$

La vraisemblance réduite est obtenue on substituant le résultat obtenu précédemment $\widehat{\lambda}_N(k) = |X_k|^2 - \widehat{\lambda}_S(k) \Rightarrow \widehat{\lambda}_S(k) = |X_k|^2 - \widehat{\lambda}_N(k)$ dans l'équation (III.22) $P(X \mid \Theta, \mathrm{H}_1)$ nous obtiendrons $P(X \mid \widehat{\Theta}, \mathrm{H}_1)$ par conséquent [73].

$$\Lambda_g = \frac{1}{L} \ln \frac{P(X \mid \widehat{\Theta}, \mathrm{H}_1)}{P(X \mid \mathrm{H}_0)} \quad \text{(III.33)}$$

$$\ln \Lambda(k) \underset{<}{\overset{H_1}{>}} \ln \gamma = \eta \qquad \text{(III.34)}$$
$$H_0$$

$$\Lambda_g = \frac{1}{L}\sum_{k=0}^{L-1}\left\{\frac{|X_k|^2}{\lambda_N(k)} - \ln\frac{|X_k|^2}{\lambda_N(k)} - 1\right\} \underset{<}{\overset{H1}{>}} \eta \qquad \text{(III.35)}$$
$$H0$$

$$\eta = \ln(\gamma)$$

Où γ définit un seuil pour basculer d'une hypothèse à l'autre ($0 \le \gamma \le 1$) ce qui a par conséquent aussi de mieux estimé les densités de probabilité.

$$P(\mathrm{H}_0 \mid \mathrm{X}_\mathrm{k}) = \frac{1}{1+\varepsilon.\Lambda_g(\mathrm{k})} \qquad \text{(III.36)}$$

$$P(\mathrm{H}_1 \mid \mathrm{X}_\mathrm{k}) = \frac{\varepsilon.\Lambda_g(k)}{1+\varepsilon.\Lambda_g(k)} \qquad \text{(III.37)}$$

III.6. La prédiction du bruit de fond

À une faible variation de la variance, l'espérance mathématique est égale à cette variance d'après le théorème de l'erreur quadratique moyenne (**a minimum mean square error estimator** '**MMSE**') [78].

$E[X] = \sum_j E[X \mid \mathrm{B_j}]P(\mathrm{B_j}) = \hat{\mathrm{v}}\mathrm{ar}(x)$ Estimation maximum de la variance de x, elle représente la meilleure approche pour l'estimation de Bayes.

On choisit comme variable X la variance du bruit de fond λ_N que nous allons prédire par la formule suivant :

$\hat{\lambda}_N(k) = E(\lambda_N(k) \mid \mathrm{X_k})$ Où $E(\lambda_N(k) \mid \mathrm{X_k})$ est l'espérance mathématique et $\hat{\lambda}_N(k)$ est l'estimation maximum de la variance du bruit de fond, ce qui donne pour la variance du bruit de fond :

$$\hat{\lambda}_N(k) = E(\lambda_N(k) \mid \mathrm{X_k}) \qquad \text{(III.38)}$$

$$\hat{\lambda}_N(k) = E(\lambda_N(k) \mid \mathrm{H}_0).P(\mathrm{H}_0 \mid \mathrm{X_k}) + E(\lambda_N(k) \mid \mathrm{H}_1).P(\mathrm{H}_1 \mid \mathrm{X_k}) \qquad \text{(III.39)}$$

En utilisant les résultats obtenus précédemment (III.36), (III.37) on obtient :

$$E(\lambda_N(k) \mid \mathrm{X_k}) = \frac{1}{1+\varepsilon.\Lambda_g(\mathrm{k})}.E(\lambda_N(k) \mid \mathrm{H}_0) + \frac{\varepsilon.\Lambda_g(k)}{1+\varepsilon.\Lambda_g(k)}.E(\lambda_N(k) \mid \mathrm{H}_1) \qquad \text{(III.40)}$$

$$\widehat{\lambda}_N(k) = \frac{1}{1+\varepsilon.\Lambda_g(\mathrm{k})}.E(\lambda_N(k)\,|\,\mathrm{H}_0) + \frac{\varepsilon.\Lambda_g(k)}{1+\varepsilon.\Lambda_g(k)}.E(\lambda_N(k)\,|\,\mathrm{H}_1) \qquad \text{(III.41)}$$

Cette formule va estime aux mieux le bruit de fond est de le mettre à jour, et elle va être exécutée à chaque trame ‘m’ [73], par conséquent on va remplacer :

- $\widehat{\lambda}_N(k)$ par $\widehat{\lambda}_N^{(m)}(k)$
- $E(\lambda_N(k)\,|\,\mathrm{H}_1)$ par $E(\lambda_N{}^{(m)}(k)\,|\,\mathrm{H}_1)$
- $E(\lambda_N(k)\,|\,\mathrm{H}_0)$ Par $E(\lambda_N{}^{(m)}(k)\,|\,\mathrm{H}_0)$

On obtient :

$$\widehat{\lambda}_N{}^{(m)}(k) = \frac{1}{1+\varepsilon.\Lambda_g{}^{(\mathrm{m})}(\mathrm{k})}.E(\lambda_N{}^{(m)}(k)\,|\,\mathrm{H}_0) + \frac{\varepsilon.\Lambda_g{}^{(\mathrm{m})}(k)}{1+\varepsilon.\Lambda_g{}^{(\mathrm{m})}(k)}.E(\lambda_N{}^{(m)}(k)\,|\,\mathrm{H}_1) \qquad \text{(III.42)}$$

Et lorsqu’on n’a pas de parole on peut remplacer

- $E(\lambda_N{}^{(m)}(k)\,|\,\mathrm{H}_0)$ par $\left|X_k{}^{(m)}\right|^2$
- $E(\lambda_N{}^{(m)}(k)\,|\,\mathrm{H}_1)$ par $\widehat{\lambda}_s^{(m-1)}(k)$

$$\widehat{\lambda}_N{}^{(m)}(k) = \frac{1}{1+\varepsilon.\Lambda_g{}^{(\mathrm{m})}(\mathrm{k})}.\left|X_k{}^{(m)}\right|^2 + \frac{\varepsilon.\Lambda_g{}^{(\mathrm{m})}(k)}{1+\varepsilon.\Lambda_g{}^{(\mathrm{m})}(k)}.\widehat{\lambda}_N{}^{(m-1)}(k) \qquad \text{(III.32)}$$

Est cette formule va permettre de mètre à jour la prédiction du bruit de fond dès que n’y’a pas de parole. [73]

III.7. Conclusion

L’utilisation d’une estimation statistique basée sur le théorème de Bayes a permis de mieux estimer le bruit de fond en se basant sur un pourcentage de vraisemblance, qui a permis de donner plus de souplesse au programme en s’adaptant à plusieurs types de bruit de fond, l’étude approfondie du problème a permis d’avoir plus de précision sur des échantillons variables.

CHAPITRE IV

GENERALITE ET ARCHITECTURE DES PROCESSEURS DSP

IV.1. Généralité sur les DSP (Digital Signal Processor)

Un DSP (Digital Signal Processor) est un type particulier de microprocesseur. Il se caractérise par le fait qu'il intègre un ensemble de fonctions spéciales. Ces fonctions sont destinées à le rendre particulièrement performant dans le domaine du traitement numérique du signal.

Comme un microprocesseur classique, un DSP est mis en œuvre en lui associant de la mémoire (RAM, ROM) et des périphériques. Un DSP typique a plutôt vocation à servir dans des systèmes de traitement autonomes. Il se présente donc généralement sous la forme d'un microcontrôleur intégrant, selon les marques et les gammes des constructeurs, de la mémoire, des timers, des ports séries synchrones rapides, des contrôleurs DMA, des ports d'E/S divers [79].

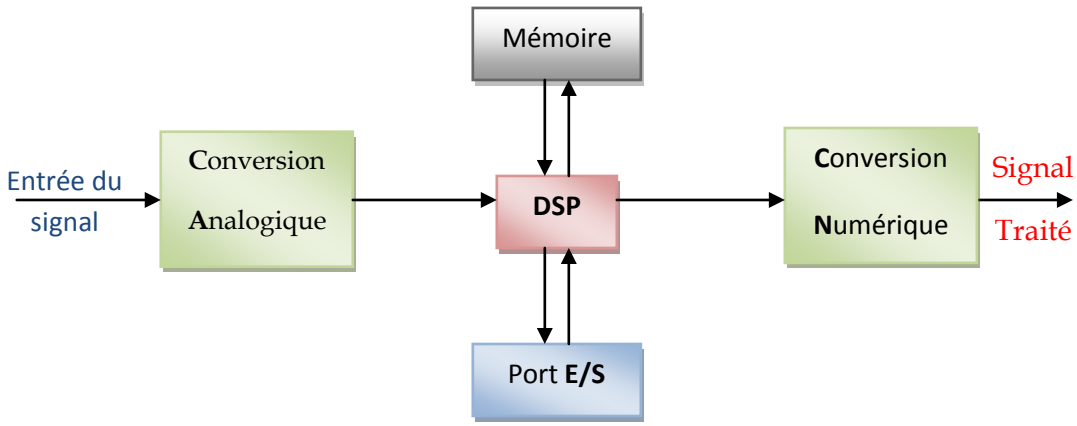


Fig.IV.1 : *Schéma global de la chaîne de traitement DSP*

IV.2. Les principales caractéristiques d'un processeur DSP

Un processeur DSP peut se distinguer d'un processeur classique par les principaux éléments suivant : l'opération MAC, l'accès à la mémoire, le pipeline.

IV.2.1. L'opération MAC

On distingue très bien l'opération **MAC** (**M**ultyply And **AC**cumulate) qui permet de calculer l'opération {A= (B*C) +D} en 1 cycle d'horloge, alors qu'un

microprocesseur classique nécessite plusieurs cycles d'horloge. Cela va, avantager le processeur DSP pour les calculs en temps réel.

IV.2.2. L'accès à la mémoire

Une autre des capacités des DSP est de réaliser plusieurs accès à la mémoire en un cycle d'horloge, ceci permet de chercher en mémoire une instruction et une donnée en même temps pour réaliser un MAC par exemple, et d'y ranger le résultat du MAC précédant simultanément. Les modes d'adressages des données sont un point particulier des DSP. Un DSP peut posséder plusieurs unités logiques de génération d'adresse, travaillant en parallèle avec la logique du cœur du DSP. Une unité logique de génération d'adresse est paramétrée une seule fois via les registres appropriés. Elle génère alors toute seule, les adresses nécessaires à l'accès des données en parallèle avec l'exécution d'une opération arithmétique.

Ceci permet non seulement de réaliser les accès mémoires simultanés en un seul cycle, mais également d'incrémenter automatiquement les adresses générées.

Ce mode d'adressage particulier, généralement appelé adressage indirect par registre avec poste (ou pré) incrément, est très utilisé pour effectuer des calculs répétitifs sur une série de données rangées séquentiellement en mémoire.

IV.2.3. Contrôle du processeur – le pipeline

Un ***pipeline*** est un élément d'un circuit électronique dans lequel les données avancent les unes derrière les autres, au rythme du signal d'horloge. Dans la microarchitecture d'un microprocesseur, c'est plus précisément l'élément dans lequel l'exécution des instructions est découpée en étages (plusieurs instructions exécutées en même temps).

L'exécution d'une instruction peut se décrire ainsi :

PFetch	Positionnement de l'adresse de la mémoire programmé et incrémentation du PC
Fetch	Recherche de l'instruction par accès à la mémoire programmé et écriture dans le registre d'instruction
Décode	Décodage de l'instruction et des adresses de l'opérande
Access	Adressage des opérandes situés en mémoire par les bus
Read	Lecture des opérandes en mémoire donnée et positionnement des adresses pour l'écriture du résultat
Exécute	Exécution de l'opération et écriture du résultat

Tab.IV.1 : *Description des opérations exécutées*

Afin de gagner du temps lors de l'exécution de séries d'instructions, il est donc nécessaire d'optimiser ces différentes étapes en les parallélisant le principe retenu est celui utilisé dans les usines de production et qui consiste à découper le travail en taches élémentaires:

Pipeline

	Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	…
Sans Pipeline	Fetch	C1				C2				C3			
	Décode		C1				C2				C3		
	Read/Write			C1				C2				C3	
	Exécute				C1				C2				C3
				Fin C1				Fin C2					Fin C3

	Cycle	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**…**
Avec pipeline	Fetch	C1	C2	C3	C4	C5	C6	C7	C8	C9	C1	C11	
	Décode		C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	…
	Read/Write			C1	C2	C3	C4	C5	C6	C7	C8	C9	…
	Exécute				C1	C2	C3	C4	C5	C6	C7	C8	…
					Fin C1		Fin C2		Fin C3				

Tab.IV.2 : *Cycle d'exécution des instructions*

- Avec le pipeline les instructions C1, C2,… se termine en même temps ;
- Sans pipeline chaque instruction doit attendre l'achèvement de la précédente.

IV.2.4. Les modes d'adressages dans les DSP

Les différents modes d'adressage utilisés par les DSP sont :

- ✓ Adressage par valeur immédiate ;
- ✓ Adressage avec une valeur absolue ;
- ✓ Adressage par accumulateur ;
- ✓ Adressage direct ;
- ✓ Adressage indirect ;
- ✓ Adressage circulaire ;
- ✓ Adressage bit-revers ;
- ✓ Adressage indirect à 2 opérandes. [79]

IV.3. Architecture des processeurs

L'architecture d'un microprocesseur est un élément important qui conditionne directement les performances d'un processeur. Il existe deux types fondamentaux de structures, dites « Von Neumann » et « Harvard », pour les DSP c'est une architecture Harvard modifiée.

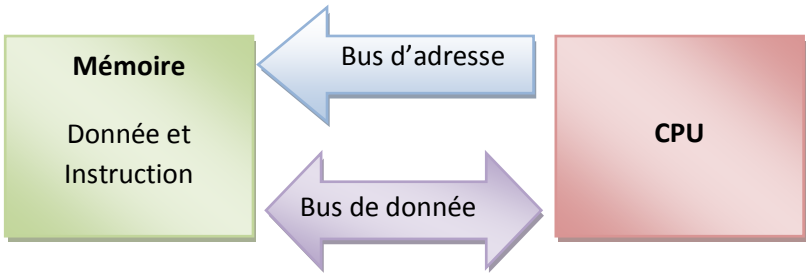


Fig.IV.2: *Architecture de type Von Neumann*

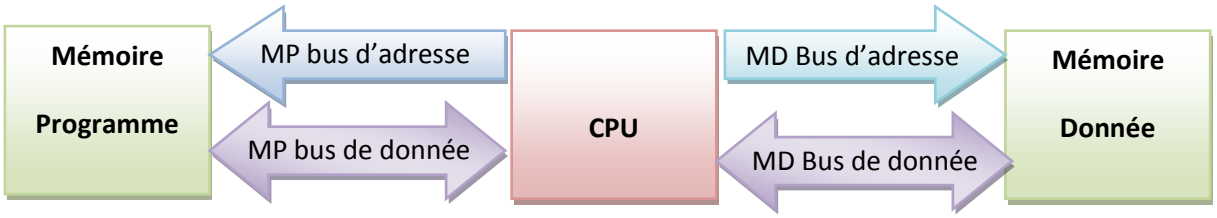


Fig.IV.3: *Architecture de type Harvard*

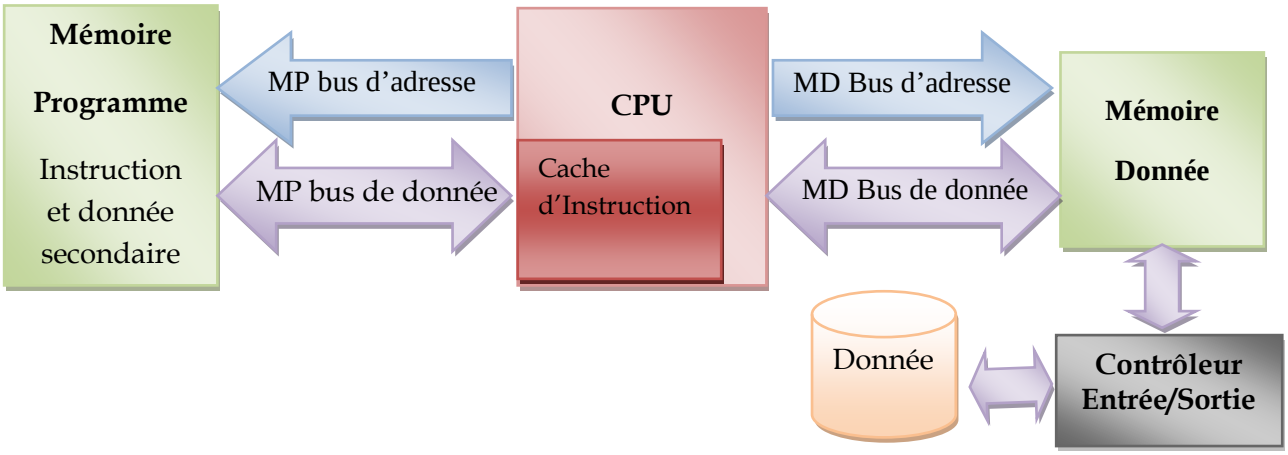


Fig.IV.4: *Super Architecture de Harvard (DSP)*

IV.3.1. Architecture de Von Neumann

Un microprocesseur basé sur une structure Von Neumann stocke les programmes et les données dans la même zone mémoire. Une instruction contient le code opératoire et l'adresse de l'opérande. Ce type de microprocesseur incorpore principalement deux unités logiques de base:

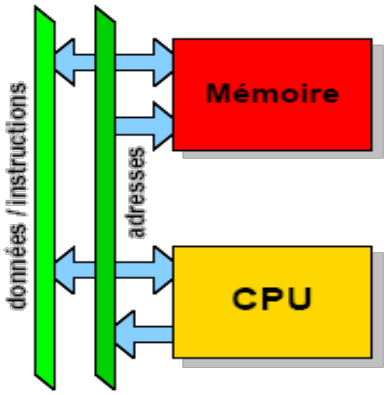


Fig.IV.5: Structure Von Neumann

- L'Unité Arithmétique et Logique, chargée de réaliser les Opérations centrales (de type multiplications, additions, soustractions, rotation, etc.),
- L'unité en charge des Entrées/Sorties, qui commande le flux de données entre le cœur du microprocesseur et les mémoires ou les ports.

IV.3.2. Architecture de Harvard

Cette structure se distingue de l'architecture Von Neumann par le fait que les Mémoires se font via un chemin distinct. Cette organisation permet de transférer une Instruction et des données simultanément, ce qui améliore les performances.

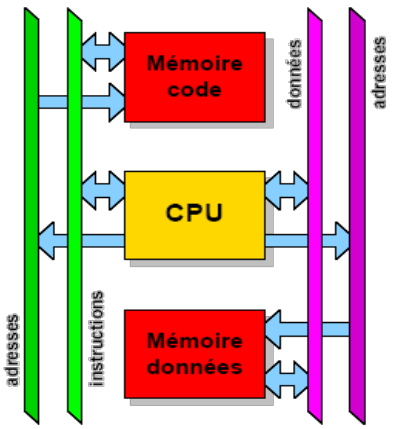


Fig.IV.6: Structure Harvard

IV.3.3. Super Architecture de Harvard (DSP)

L'architecture Harvard est plutôt utilisée dans des microprocesseurs spécialisés pour des applications en temps réels. Les DSP utilisent l'architecture dite « Structure de Harvard modifiée», qui inclus une mémoire cache d'instruction qui contient environ 32 jeux d'instructions les plus récentes, ce qui permet d'améliorer la structure de Harvard, et un contrôleur d'entrée-sortie qui permet l'échange de donné des signaux entrant et sortant.

IV.4. Les formats des données utilisées dans les DSP

Un autre point essentiel des DSP est la représentation des nombres (les données) qu'ils peuvent manipuler. Il est possible de distinguer deux familles :

- ✓ Les DSP à virgule fixe ;
- ✓ Les DSP à virgule flottante.

IV.4.1. Les DSP à virgule flottante

Les DSP à virgule flottante sont plus souples et plus faciles à programmer que les DSP à virgule fixe. Un DSP comme le TMS320C30 manipule des nombres formés avec une mantisse de 24 bits et un exposant de 8 bits (taille de la donnée en mémoire : 32 bits).

Les valeurs intermédiaires des calculs sont mémorisées dans des registres avec un format de 32 bits de mantisse et un exposant de 8 bits (taille du registre : 32 + 8 bits supplémentaires). La dynamique disponible est très grande, elle va de -1×2^{128} à 1×2^{127}, toutefois la résolution reste limitée à 24 bits au mieux. Outre les nombres fractionnaires, la très grande dynamique proposée par les DSP à virgule flottante permet de ne pas se soucier des limites des résultats calculés lors de la conception d'un programme. Cet avantage a cependant un prix, à savoir qu'un système basé sur un DSP à virgule flottante a un coût de fabrication supérieur par rapport à un système basé sur DSP à virgule fixe.

Un DSP à virgule flottante est adapté à des applications dans lesquelles :

- ✓ Les coefficients varient dans le temps (exemple : les filtres adaptatifs),
- ✓ Le signal et les coefficients ont besoin d'une grande dynamique et de précision [79].

IV.4.2. Les DSP à virgule fixe

Un DSP à virgule fixe est un peu plus compliqué à programmer qu'un DSP à virgule flottante. Dans un DSP à virgule fixe typique comme le TMS320C25, les nombres sont codés sur 16 bits (des entiers ou des fractionnaires).

Toutefois, sur ce DSP, les calculs sont effectués avec des accumulateurs de 32 bits. Lorsque les résultats doivent être stockés en mémoire, les 16 bits les moins significatifs sont perdus. Ceci permet de limiter les erreurs d'arrondis cumulatives. La précision des calculs est un point critique des DSP à virgule fixe, car le concepteur de programmes doit rester vigilant à chaque étape d'un calcul. Il doit rechercher la plus grande dynamique possible (c.-à-d. exploiter au mieux la gamme des nombres disponibles), pour conserver une bonne précision des calculs, les programmeurs contournent les limites des DSP à virgule fixe en déterminant à l'avance, et avec soin, la précision et la dynamique nécessaires (par méthode analytique ou avec des outils de simulation) pour réaliser leurs projets.

Il est également possible d'effectuer des opérations en virgule flottante dans un DSP à virgule fixe par le biais de routines logicielles adéquates. Cette approche est néanmoins pénalisante en temps d'exécution, même sur un DSP à virgule fixe très rapide.

Les DSP à virgule fixe sont les plus utilisés, car ils sont moins chers que les DSP à Virgule flottante. On les trouve dans tous les produits de grande diffusion ou le coût est un facteur important [79].

IV.5. Présentation de la carte DSP TMS320C6713DSK

Comme son nom l'indique DSP… DSK (**D**igital Signal Processor **S**tarter **K**it) c'est une carte qui permet de s'initier à la technologie DSP (traitement numérique du signal) elle a un but éducatif.

Notre carte TMS320C6713DSK est spécialement dédiée pour le traitement du son, les décodeurs audio numériques de la parole (Dolby Digital ^{TM} le son 3D…). Par la même occasion, elle dispose de plusieurs entrées analogique pour l'acquisition du signal (Mic in,line in) et de sorties analogique aussi [80].

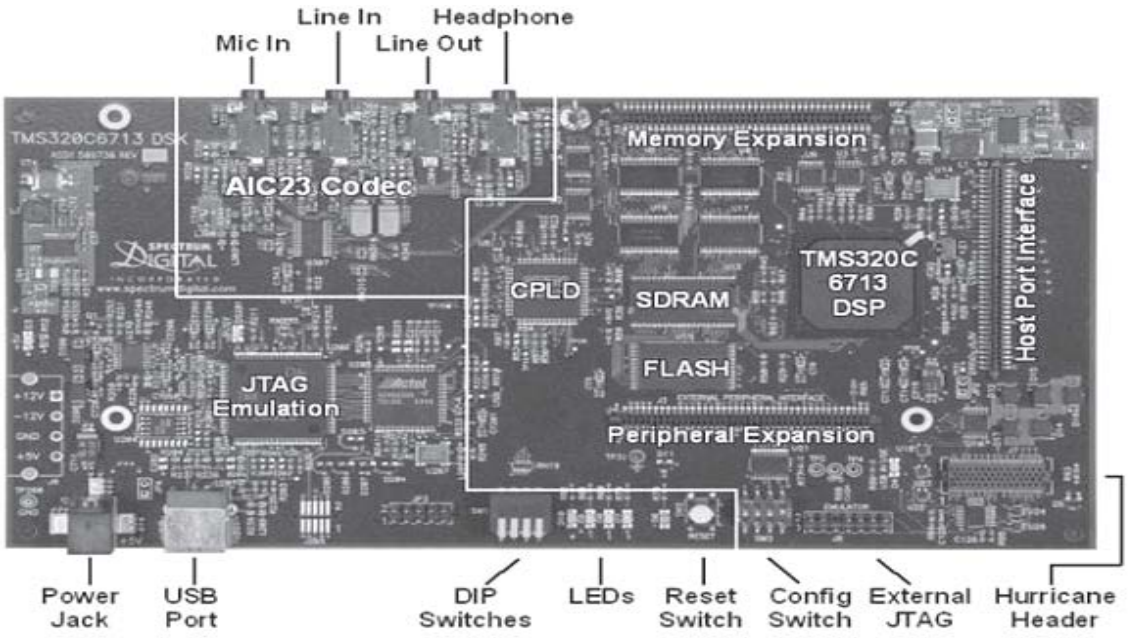


Fig.IV.7: *Photo de la carte DSP TMS320C6713*

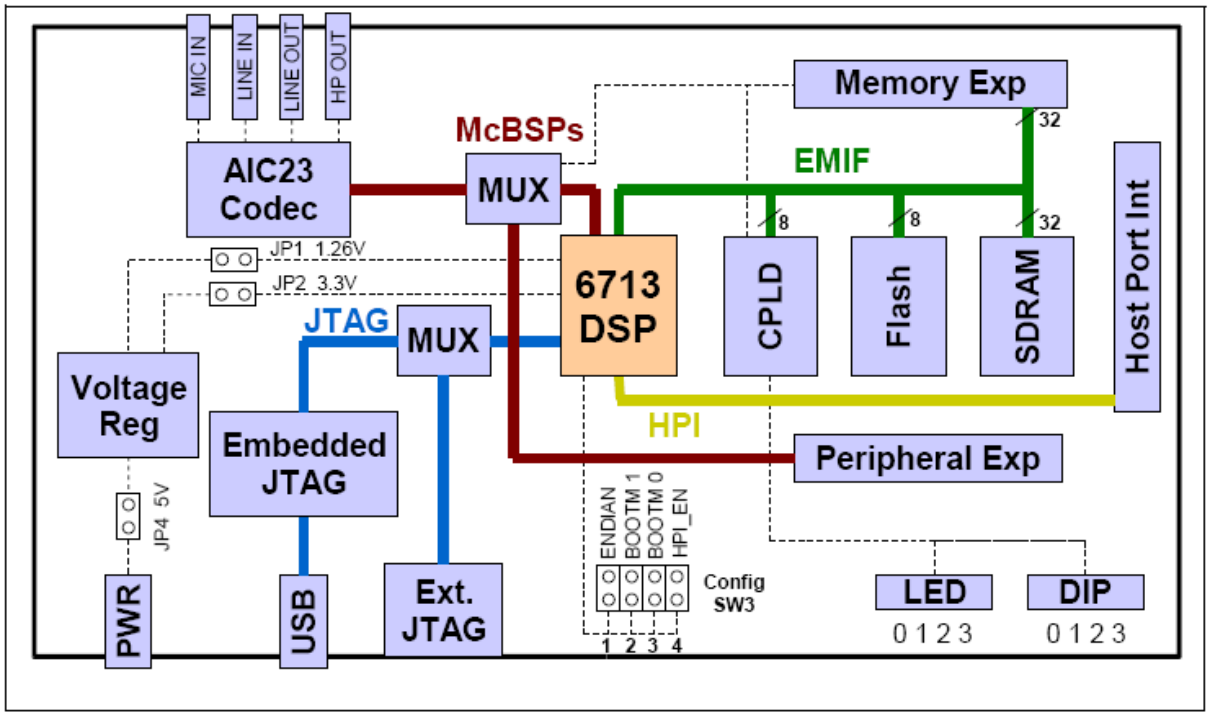


Fig. IV.8 *Schéma de la carte DSP TMS320C6713*

- Elle opère à 225 MHz d'horloge ;
- Codec (codeur/décodeur) stéréo AIC23 avec entrée Line In, sortie Line Out, et un Microphone et une sortie pour casque (écouteur) ;
- 16 Mbites de DRAM (SDRAM) ;
- 512 Kbites non volatile de mémoire Flash ;
- 4 LED accessibles par programme et un DIP switches ;
- Possibilité de configurer le boots ;

- Possibilité de la connecter à d'autre carte (HPI) ;
- Possibilité de configurer la carte à l'aide des registres implantés sur (CPLD) ;
- Interfaçage PC par USB ;
- Émulateur JTAG (**Joint Test Action Group**) il permet de tester l'état de la carte ;
- 400 MMACS (Méga MAC par second) ;
- Virgule flottante avec une architecture VLIW (Very Long Instruction Word) ;
- Donnée sur 32bit ;
- 4.4ns par cycle d'horloge ;
- 1350 MIPS (Million Opération Par Seconde) ;
- 2 Timers [80].

IV.5.2. Architecture de la carte TMS320C6713

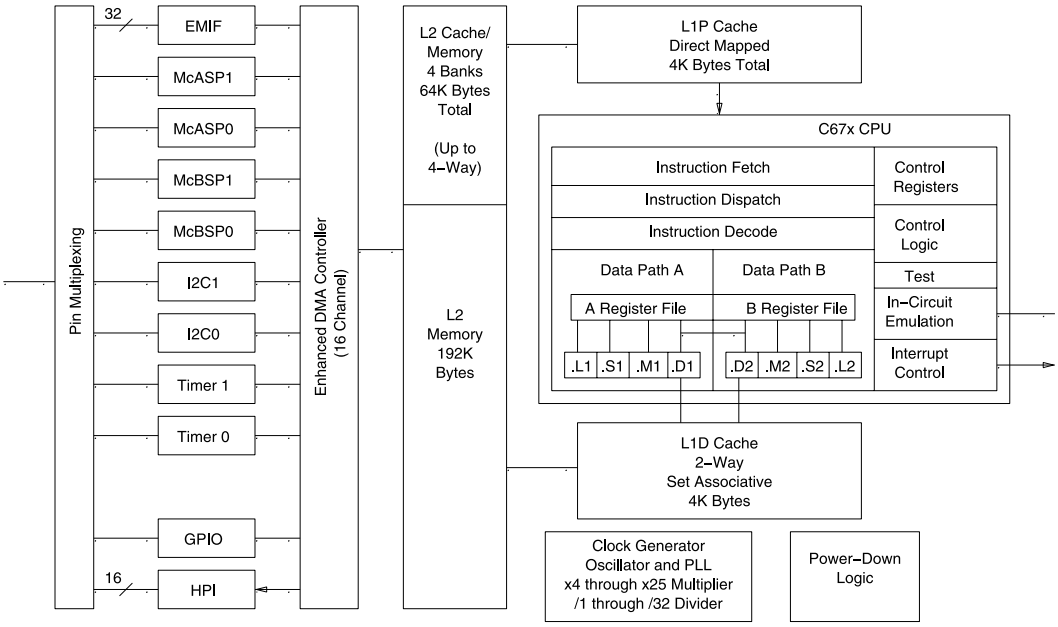


Fig.IV.9 : *Schéma bloc de la carteTMS320C6713*

- Elle dispose 4 UAL (unité arithmétique est logique) qui peuvent opérer en virgule fixe ou flottante (.L1, .L2, .S1, .S2) ;
- 2 UAL qui peuvent opérer en virgule fixe seulement (.D1, .D2) ;
- 2 Multiplicateur (.M1, .M2) qui peuvent opérer en virgule fixe ou flottante [80].

IV.5.3. Structure générale des registres

Le CPU a des registres de 32bits devisés en égalité entre A et B, le CPU a une structure Load/Store (chargée/stocker) avec la plus part des jeux d'instruction opérant sur ces registres. L'adressage de donnée s'effectue par l'unité .D1 et .D2 elle se charge de tous les transferts entre registres fichier et mémoire, elle dispose d'un seul bus de donnée connecté à tous les registres pour avoir un accès immédiat en 1 cycle d'horloge [80].

IV.6. Conclusion

Les DSP's ont bénéficié des énormes progrès en rapidité grâce au faible temps de commutation et en puissance de calcul grâce au nombre de bits des bus internes. Il est probable que dans les années prochaines, le DSP deviendra un composant indispensable, non seulement aux télécommunications, mais aussi à l'électronicien, et même à l'automaticien, au même titre que l'amplificateur opérationnel ou le microcontrôleur.

La structure d'un processeur DSP (TMS 320C6713) est optimisée au maximum pour des applications en traitement numérique du signal et surtout pour le traitement en temps réel. Elle comporte plusieurs UAL, ces dernières se divisent les taches en pipeline.

Ce processeur est idéal pour les systèmes embarqués.

CHAPITRE V

IMPLEMENTATION PRATIQUE ET ÉVALUATION DES RESULTATS

V.1. Introduction

L'implémentation d'un algorithme de traitement du signal sur un système embarqué autonome à beaucoup évolue de nos jours, la programmation se fait par des langages graphiques qui permettent de créer un programme d'une manière conceptuelle, parmi ces langages on peut citer Matlab, Scilab, LabView.

Ces logicielles peuvent communiquer avec le IDE (Integrated development environment) comme **Code Composer Studio** CCS, qui permet de générer l'exécutable sur le système embarqué « dédie aux cartes DSP [TMS] ».

Le traitement du signal vocal est un processus très complexe en termes d'extraction de l'information utile et sa reconnaissance, car il concerne l'une des problématiques les plus influentes dans cette phase. Le corpus sur lequel s'effectuent les tests de performance est fondamental étant donné que l'extraction des paramètres acoustiques est sujette aux différentes influences qui sont dues à l'enregistrement, au réglage du niveau sonore du microphone, au bruit environnant, les défauts de prononciation et la prononciation incorrecte d'un mot. Tous ces défauts influent directement sur la phase de reconnaissance.

Nous avons fait l'apprentissage sur une base de données comportant cinq (05) mots isolés, à savoir : **Avant, Arrière, Gauche, Droite et Stop**. Cela pour commander vocalement une petite voiture, les tests de reconnaissance ont été effectués sous Matlab par une base de test préenregistrée, et aussi sous Simulink pour la reconnaissance en direct.

Pour pouvoir implémenter nos algorithmes sur la carte DSP, nous avons partagé le travail sur quatre (04) grands axes:

1- Création du modèle, reconnaissance et validation des résultats **sous Matlab** ;
2- Adaptation des algorithmes au langage **Simulink** et création des fichiers exécutables;
3- Exploitation de l'exécutable sur **Code Studio Composer** CCS 3.3 ;
4- Implémentation sur la carte **DSP TMS320 C 6713**.

V.2. Description du corpus

L'un des éléments clés de toute recherche est sa base de données, dans cet ordre d'idée, nous avons essayé de développer un corpus de 5 mots isolés, chaque mot a été prononcé dix (10) fois, dans un même milieu (laboratoire) et en mode mono locuteur, avec une fréquence d'échantillonnage égale à 8000 Hz.

V.2.1. Enregistrements dans un milieu bruité

L'enregistrement du corpus a été fait dans une salle ordinaire (laboratoire), ceci afin d'inclure le bruit environnant, car nous préconisons d'utiliser notre système dans un environnement en conditions normales.

Dans ce contexte, nous avons utilisé le modèle Simulink de la Figure.V.1 afin de créer la base de données de corpus tout en délimitant les bornes du mot prononcé par la fonction VAD (voir chapitre III).

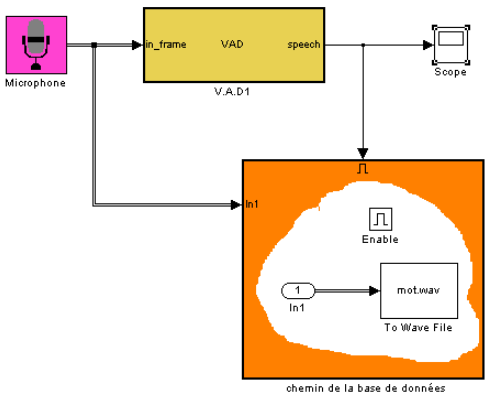


Fig.V.1 *Modèle d'enregistrement de la base des corpus*

Dès que le bloc de la détection d'activité vocale reçoit la parole, il va envoyer un signal carré pour activer le bloc suivant (Figure.V.2), où se trouve le chemin de la base de corpus.

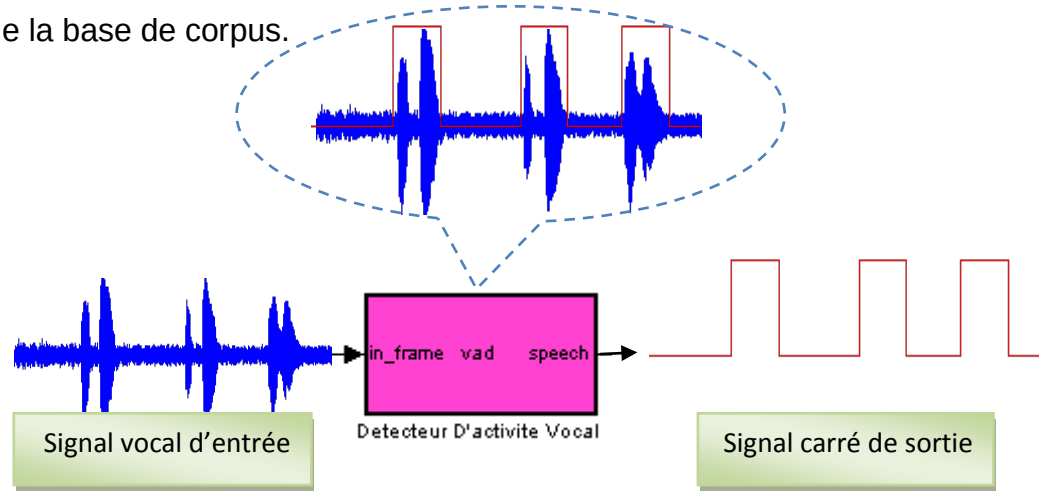


Fig.V.2 *Déroulement de la détection d'activité vocale*

V.3. Création du modèle, reconnaissance et validation des résultats sous Matlab

V.3.1. Création du modèle

La méthodologie adoptée se basé sur l'utilisation des classifieurs tel que les HMM, on utilisant les MFCC comme paramètres acoustiques robustes, afin de faire la reconnaissance de cinq (05) mots isolés, par le calcul du score maximum de vraisemblance.

Pour cela nous avons suit les étapes suivantes :

- ✓ Lecture du corpus en spécifiant le chemin de la base;
- ✓ La paramétrisation en utilisant les MFCC ;
- ✓ Modélisation des mots du corpus par les HMM ;
- ✓ Apprentissage et création du modèle HMM pour chaque mot de la base de données.

En premier lieu, nous avons effectué une modélisation du corpus. Le nombre d'états des modèles de Markov est spécifique à la longueur phonémique du mot.

Les modèles HMM sont définis par :

- La fréquence d'échantillonnage ;
- Le nombre de paramètres représentant les données ;
- Le nombre d'états ;
- Le nombre de gaussiennes modélisant les données par état ;
- Le nombre d'itérations de l'algorithme EM.

V.3.1.1. Les modèles HMM des mots de corpus

Nous avons modélisé les mots [Arrière, Avant, Gauche, Droite, Stop] avec un nombre d'HMM varie de 2 à 4 états selon le mot. Le nombre de coefficients MFCC varie entre 12 et 42, le nombre de GMM est fixé sur 4 pour chaque état, et finalement le nombre d'itérations varie entre 20 et 40.

Les paramètres globaux des modèles HMM ainsi que les matrices de transitions obtenues sont mentionnés respectivement dans les tableaux V.1 à V.18.

a/ **Modèle avec 12 MFCC :**

Le mot	Nombre d'états du HMM	Mélange de gaussiennes par état	Nombre d'itérations de l'algorithme EM	Coefficients MFCC/trame
Arrière	4	4	20	12
Avant	2	4	40	12
Gauche	4	4	20	12
Droite	4	4	20	12
Stop	3	4	20	12

Tab V.1 : *Paramètres des modèles HMM* à 12 MFCC

Transitions	État 1	État 2	État 3	État 4
État 1	0,9175	0,0318	0,0506	0,0001
État 2	0	0,9672	0,0327	0,0001
État 3	0	0	0,9559	0,0441
État 4	0	0	0	1

Tab V.2 : *Matrice de transition du modèle HMM1 du mot Arrière*

Transitions	État 1	État 2
État 1	0,9631	0,0369
État 2	0	1

Tab V.3 : *Matrice de transition du modèle HMM2 du mot Avant*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9326	0,0384	0,0289	0,0001
État 2	0	0,9647	0,0352	0,0001
État 3	0	0	0,9665	0,0335
État 4	0	0	0	1

Tab V.4 : *Matrice de transition du modèle HMM3 du mot Gauche*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9482	0,0517	0,0001	0
État 2	0	0,9289	0,0710	0,0001
État 3	0	0	0,9261	0,0739
État 4	0	0	0	1

Tab V.5 : *Matrice de transition du modèle HMM4 du mot Droite*

Transitions	État 1	État 2	État 3
État 1	0,9595	0,0404	0,0001
État 2	0	0,9687	0,0313
État 3	0	0	1

Tab V.6 : *Matrice de transition du modèle HMM5 du mot-stop*

b/ **Modèle avec 16 MFCC :**

Le mot	Nombre d'états du HMM	Mélange de gaussiennes par état	Nombre d'itérations de l'algorithme EM	Coefficients MFCC/trame
Arrière	4	4	20	16
Avant	2	4	40	16
Gauche	4	4	20	16
Droite	4	4	20	16
Stop	3	4	20	16

Tab V.7 : *Paramètres des modèles HMM à 16 MFCC*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9343	0,0657	0	0
État 2	0	0,9425	0,0575	0
État 3	0	0	0,9361	0,0639
État 4	0	0	0	1

Tab V.8 : *Matrice de transition du modèle HMM1 du mot Arrière*

Transitions	État 1	État 2
État 1	0,9636	0,0364
État 2	0	1

Tab V.9 : *Matrice de transition du modèle HMM2 du mot Avant*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9059	0,0941	0	0
État 2	0	0,8678	0,1322	0
État 3	0	0	0,8717	0,1283
État 4	0	0	0	1

Tab V.10 : *Matrice de transition du modèle HMM3 du mot Gauche*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9388	0,0612	0	0
État 2	0	0,9233	0,0767	0
État 3	0	0	0,9353	0,0647
État 4	0	0	0	1

Tab V.11 : *Matrice de transition du modèle HMM4 du mot Droite*

Transitions	État 1	État 2	État 3
État 1	0,9563	0,0437	0
État 2	0	0,9408	0,0592
État 3	0	0	1

Tab V.12 : *Matrice de transition du modèle HMM5 du mot Stop*

c/ **Modèle avec 42 MFCC :**

Le mot	Nombre d'états du HMM	Mélange de gaussiennes par état	Nombre d'itérations de l'algorithme EM	Coefficients MFCC/trame
Arrière	4	4	20	42
Avant	2	4	40	42
Gauche	4	4	20	42
Droite	4	4	20	42
Stop	3	4	20	42

Tab V.13 : *Paramètres des modèles HMM à 42 MFCC*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9397	0,0603	0	0
État 2	0	0,9376	0,0624	0
État 3	0	0	0,9363	0,0637
État 4	0	0	0	1

Tab V.14 : *Matrice de transition du modèle HMM1 du mot Arrière*

Transitions	État 1	État 2
État 1	0,9735	0,0265
État 2	0	1

Tab V.15 : *Matrice de transition du modèle HMM2 du mot Avant*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9444	0,0556	0	0
État 2	0	0,9383	0,0617	0
État 3	0	0	0,919	0,081
État 4	0	0	0	1

Tab V.16 : *Matrice de transition du modèle HMM3 du mot Gauche*

Transitions	État 1	État 2	État 3	État 4
État 1	0,9473	0,0527	0	0
État 2	0	0,9154	0,0846	0
État 3	0	0	0,9333	0,0667
État 4	0	0	0	1

Tab V.17 : *Matrice de transition du modèle HMM4 du mot Droite*

Transitions	État 1	État 2	État 3
État 1	0,9585	0,0415	0
État 2	0	0,9499	0,0501
État 3	0	0	1

Tab V.18 : *Matrice de transition du modèle HMM5 du mot Stop*

V.3.1.2. Discussions

Nous avons remarqué que l'augmentation du nombre de coefficients agit directement sur la matrice de transition et fait tendre notre HMM vers un modèle gauche- droite. Ceci n'influe pas directement sur la convergence en minimisant les itérations, mais agit plutôt sur la vitesse de convergence (temps d'une itération).

Le choix du nombre d'états HMM pour chaque mot a été fait après un nombre très important de tests d'apprentissage et de reconnaissance. Par exemple, prenons le mot ''Droite'', nous avons commencé par 4 états HMM, cela nous a donné un taux de reconnaissance égal à environ 30%. Et après que nous avons refixé le nombre sur 2 états, le taux de reconnaissance a été amélioré jusqu'à plus de 95%.

V.3.2. Reconnaissance et validation des résultats

Cette étape est une validation de notre modélisation HMM, pour cela nous avons créé une nouvelle base de données de test comportant les cinq mots, chaque mot a été prononcé dix (10) fois dans un même milieu (laboratoire) et même locuteur. Et nous allons essayer de reconnaitre les mots de la base de test avec les modèles créés auparavant.

Mots	Modèle HMM correspondant	Taux de reconnaissance		
		12 MFCC	16 MFCC	42 MFCC
Arrière	HMM 1	100%	100%	100%
Avant	HMM 2	100%	100%	100%
Gauche	HMM 3	90%	100%	100%
Droite	HMM 4	100%	100%	90%
Stop	HMM 5	100%	100%	100%
Taux de reconnaissance global		98%	100%	98%

Tab V.19 : *Taux de reconnaissance sur la base de test*

La base de test est reconnue de 98% à 100% des cas dans les trois initiations des MFCC, ceci montre qu'en termes de reconnaissance, l'approche HMM avec 16 coefficients MFCC est adéquate à notre système.

V.4. Adaptation des algorithmes au langage Simulink et création des fichiers exécutables

V.4.1.Présentation du Simulink

Simulink est un logiciel de modélisation des systèmes multi-physique en passant du « désigne » à la simulation, il renferme plusieurs outils qui permettent la génération du code pour un modèle donné, ainsi qu'il puisse modéliser des données simples ou multicanaux, et simuler des composants numériques, analogiques ou mixtes. Il peut aussi modéliser des sources de signaux et les visualiser.

V.4.2. Utilisation de Simulink

Pour pouvoir implémenter notre programme sur la carte DSP TMS320C6713, et sachant que le CCS permet de communiquer en directe avec le Simulink, nous étions obligés de passer par cette passerelle (le Simulink), pour qu'on puisse compiler les programmes Matlab en langage machine de la carte DSP (voir Figure V.3). Le raisonnement au Simulink est différent que celui du Matlab, pour cela nous avons adopté une autre stratégie de travail.

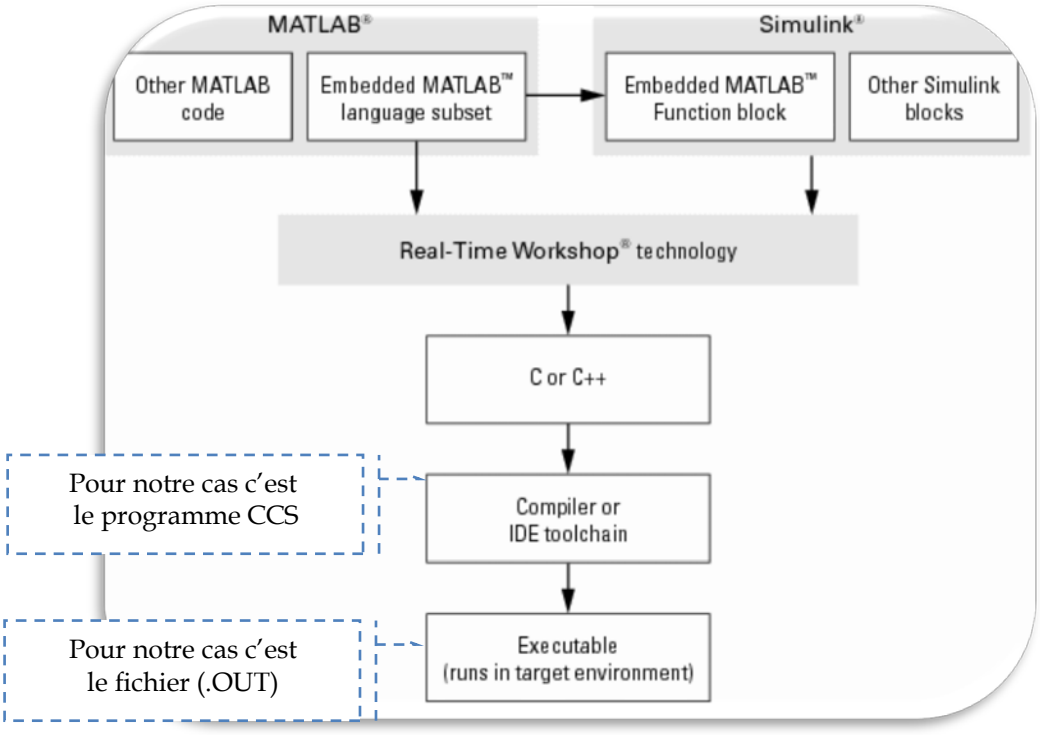


Fig. V.3. *Diagramme de génération du fichier exécutable*

V.4.3. Stratégie de travail par le Simulink

La stratégie adoptée se basé sur l'utilisation d'un algorithme de détection d'activité vocale, qui va activer, d'une part, le bloc de paramétrisation acoustique robuste MFCC, et d'autre part, le bloc de reconnaissance et de prise de décisions. Pour cela nous avons suivi les étapes suivantes :

- ✓ Détection de la parole pour **isoler et délimiter le mot à reconnaitre**, et envoyer un **signal carré d'activation** pour les autres blocs ;
- ✓ Extraction des paramètres **MFCC** ;
- ✓ Calcul de **score de vraisemblance** pour faire la reconnaissance;
- ✓ Calcul de **seuil de reconnaissance** afin de valider et accepter les résultats ;
- ✓ Codage des résultats obtenus vers un code binaire de pilotage ;
- ✓ L'envoi de la commande vers les LED's de la carte DSP.

Toutes ces étapes sont représentées dans l'organigramme de la figure V.4.

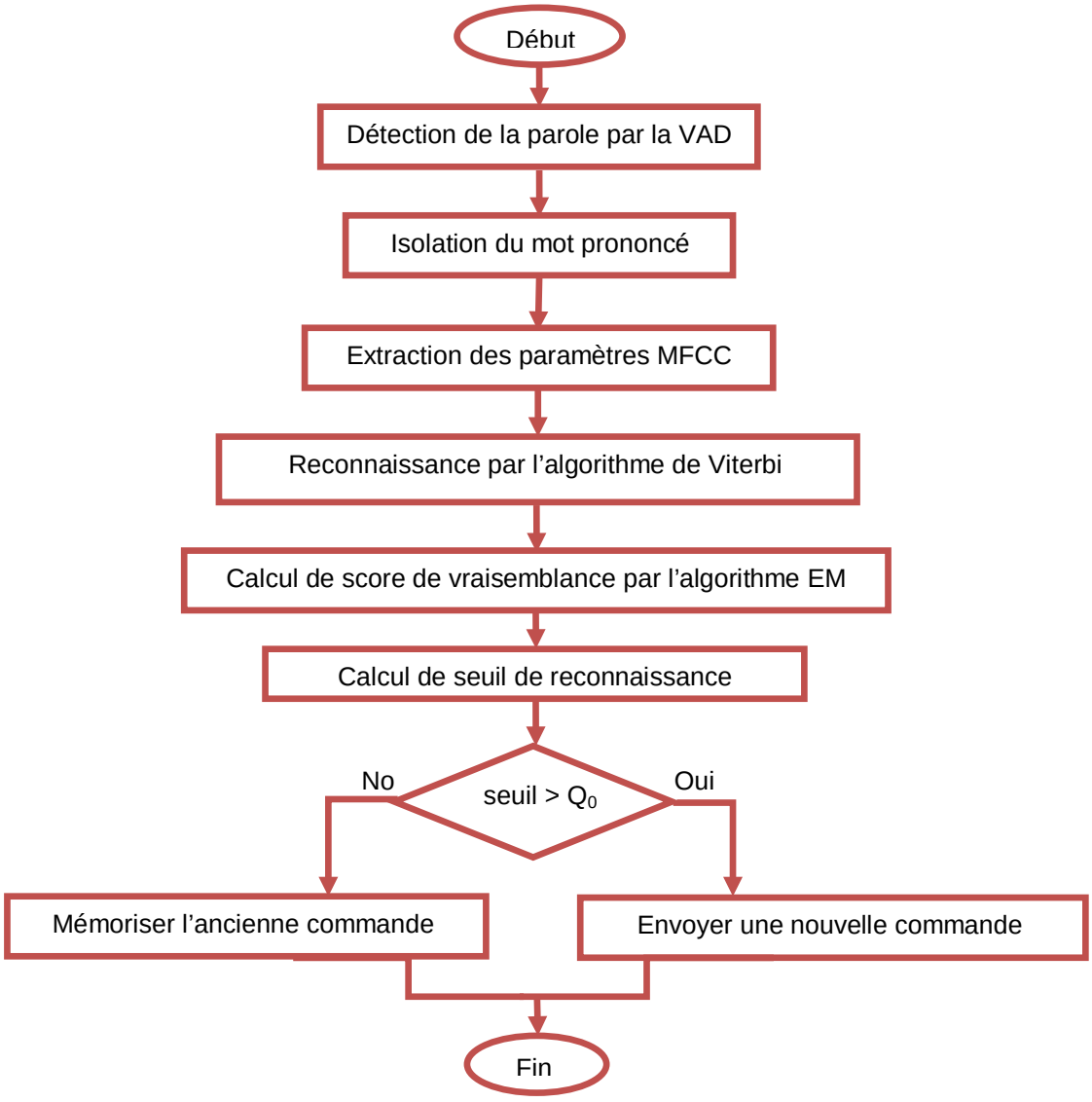


Fig.V.4 : *Organigramme de la reconnaissance par les HMM*

V.4.4. Schéma d'implémentation

L'association de Matlab, Simulink avec le CCS pour implémenter du fichier exécutable est représenté dans le schéma de la figure V.5.

Nous avons utilisé ce schéma sous Simulink pour la conception du programme qui permettra la simulation et la réalisation de l'exécutable dans la carte DSP sous CCS.

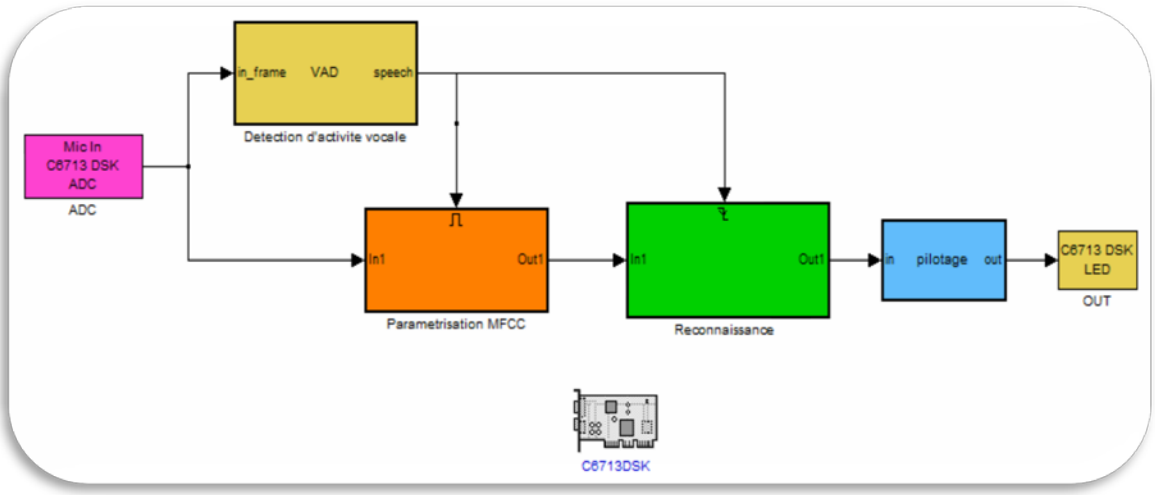
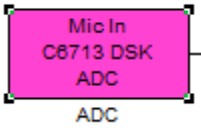


Fig.V.5 : *Schéma bloc sous Simulink*

Nous allons détailler ce schéma, bloc par bloc.

a/ Le capteur acoustique



Lorsqu'on click sur le bloc ADC, une fenêtre de configuration s'ouvre, cette dernière doit être configurée comme il est indiqué sur la Figure .V.6.

Après l'exécution, ce bloc va envoyer successivement des paquets de données de 256 échantillons avec la fréquence d'échantillonnage de 8Khz en mode mono.

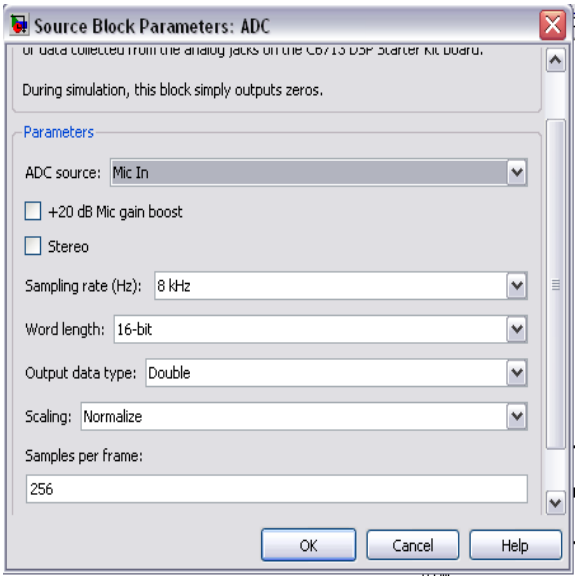
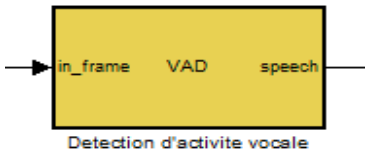


Fig.V.6 *Les paramètres du bloc ADC C6713DSK*

Les mêmes données vont être envoyées vers deux (2) blocs différents, en même temps. D'une part, le bloc de détection d'activité vocale VAD, et d'autre part, le bloc d'extraction des paramètres acoustiques MFCC.

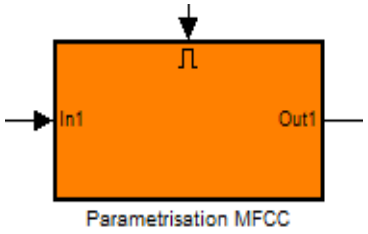
b/ Le détecteur d'activité vocale



Dès que le bloc de la détection d'activité vocale reçoit la parole, il va envoyer un signal carré de 0 et 1 pour activer les autres blocs.

c/ L'extraction des paramètres MFCC

Ce bloc dispose d'une entrée d'activation en dessus, vient directement de la VAD.



Cette entrée a le rôle suivant :

- Si la VAD = 0, ce bloc est désactivé, et les données reçues de l'entrée **In1** ne serons pas traitées.
- Et si la VAD = 1, ce bloc est activé donc il commence à coder les échantillons reçus et générer les paramètres MFCC, en accumulant les échantillons codés les uns sur les autres.

Remarque :

Après les tests effectués sous Simulink, nous avons remarqué que l'utilisation de 42 coefficients MFCC ***augmente énormément le temps de réponse du système****, et ils influencent négativement sur les résultats obtenus, cette influence due au chevauchement des données à la sortie de ce bloc. C'est pour cette raison nous avons choisi un nombre de 16 paramètres MFCC, ce choix nous a donné des bons résultats en temps réel.*

Le codage MFCC peut réduire considérablement la taille des informations à traiter, par exemple pour notre cas nous avons 256 échantillons par trame codés à 16 valeurs utiles, ces valeurs seront empilées dans une mémoire tampon le ***Delay Line*** *qui va former une matrice de 16 colonnes et un nombre fini de lignes correspond à la longueur du mot à reconnaitre.*

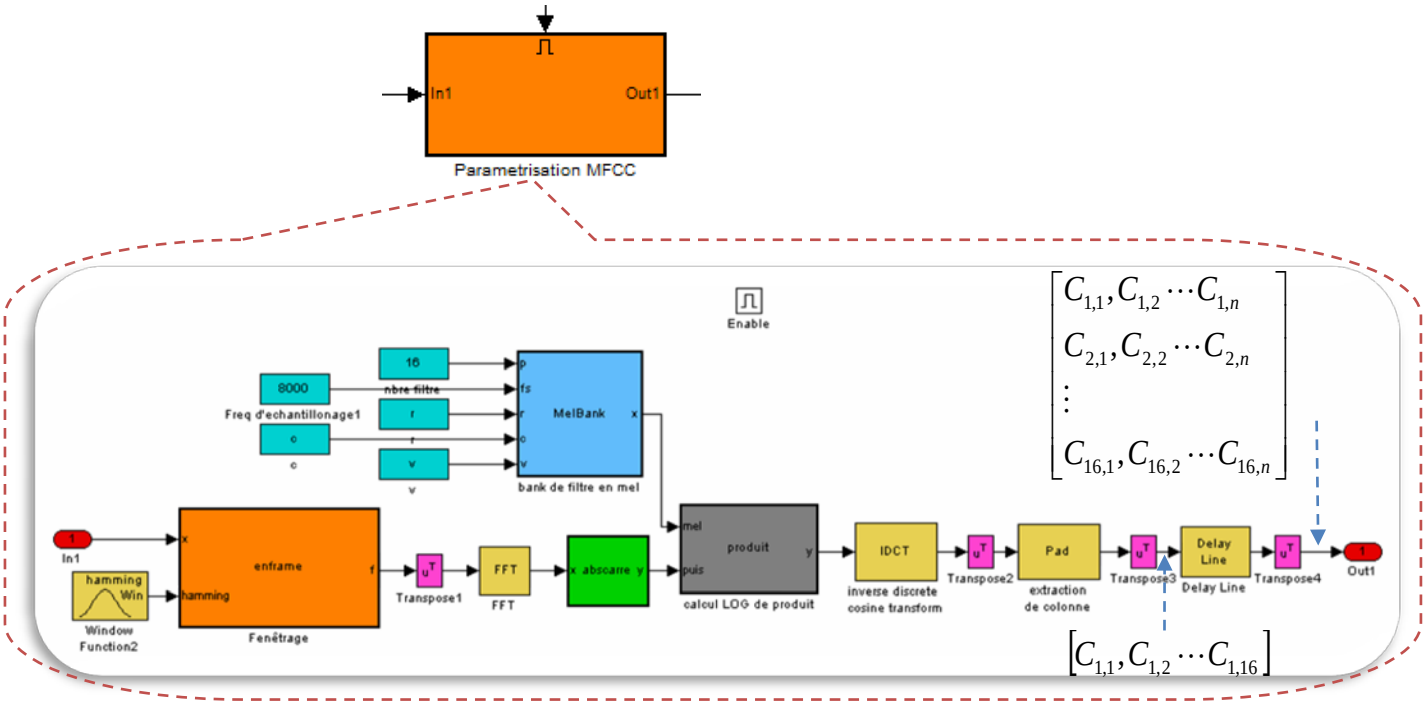
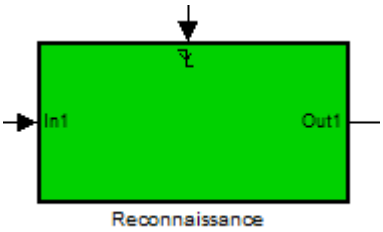


Fig.V.7 : *Les différents sous-blocs du bloc de paramétrisation MFCC*

d/ La reconnaissance



Ce bloc est activé par le front descendant du signal venait de la VAD, il fait la reconnaissance et la décision en comparant la matrice des paramètres MFCC, obtenues on temps réel, avec les modèles HMM déterminés préalablement. Cette comparaison se réalisée à l'aide de l'algorithme de Viterbi qui va calculer le meilleur chemin (appelé chemin de Viterbi) et la probabilité de vraisemblance maximale (loglikelihood).

Nous avons aussi défini un **facteur de qualité** (seuil) afin de faire une bonne reconnaissance et élimine au même temps toutes les perturbations lors de la prise de décision. Ce seuil a été calculé de la façon suivante :

$$seuil\ (\%) = \frac{(1 - \max(Loglikelihood))}{\sum_1^5 Loglikelihood} * 100 \qquad \text{(V.1)}$$

Une fois un mot est reconnu avec une bonne qualité, un numéro prédéfini correspondant à chaque mot de la base de corpus sera affiché à la sortie de ce bloc (ex : 1 : Arrière ; 2 : Avant ; 3 : Gauche ; 4 : Droite ; 5 : Stop). Ce numéro est envoyé directement vers le bloc de pilotage.

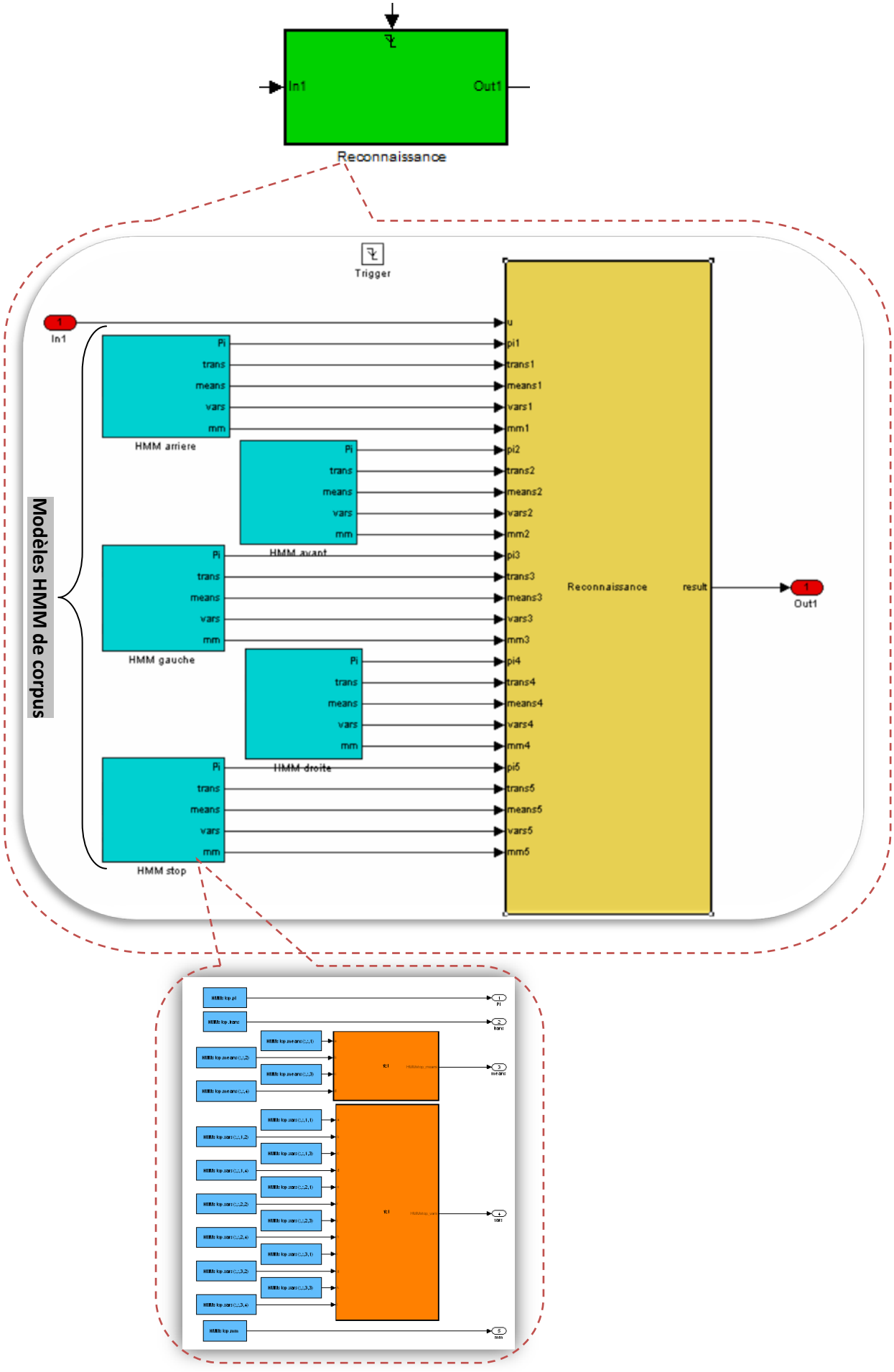


Fig.V.8 : *Les différents sous-blocs du bloc de reconnaissance*

e/ Le pilotage

Ce bloc sert à envoyer un code binaire pour commander les LED's de la carte DSP TMS320C6713 qui vont commander par la suite les moteurs de notre automate.

Dans ce contexte, nous avons prévois les commandes suivantes :

Mouvement	Commande	Nombre correspondant Affiché en Simulink
En arrière	1110	14
En avant	1101	13
Rotation à gauche	1011	11
Rotation à droite	0111	7
Arrêt (stop)	1111	15
Arrière et rotation à gauche	1010	10
Arrière et rotation à droite	0110	6
Avant et rotation à gauche	1001	9
Avant et rotation à droite	0101	5

Tab V.20 : *Les codes de pilotage*

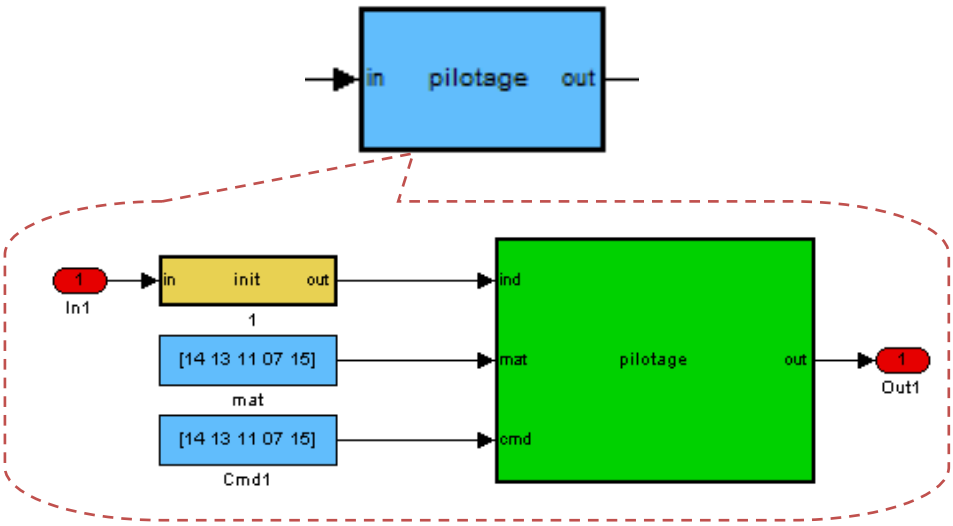
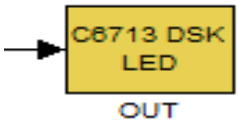


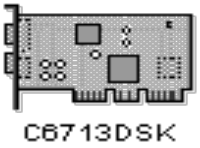
Fig.V.9 : *Les différents sous-blocs du bloc de pilotage*

e/ Les LED

Ce bloc commande directement les quatre LED's de la carte DSP TMS320C6713, il permet d'afficher en binaire un numéro décimal compris entre 0 et 15.



f/ Carte DSP



Pour pouvoir compiler le modèle Simulink vers l'exécutable de la carte DSP en étude, il doit y'avoir le bloc correspondant à cette carte avec la référence exacte écrite au dessus d'elle (dans notre cas c'est la carte TMS320C6713). Et lorsqu'on dépose ce bloc sur le modèle de Simulink, il va nous afficher un message comme illustré dans la Figure V.10, pour configurer le bios de la carte. En cliquant sur « Yes » une autre fenêtre de configuration de bios s'ouvre (**Fig.V.11)**

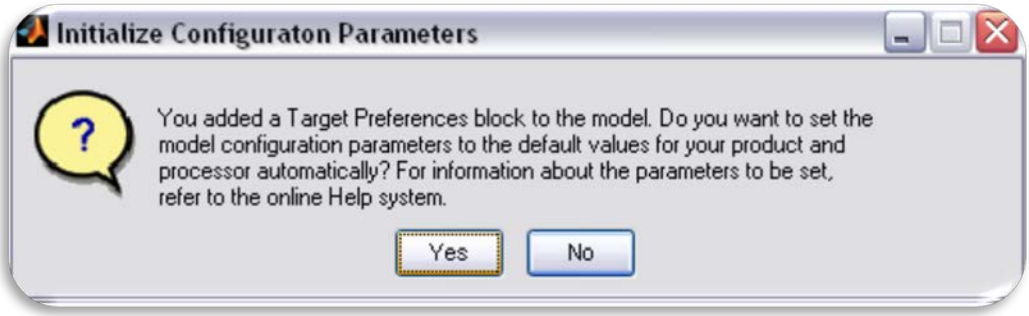


Fig.V.10 : *Fenêtre d'acceptation de configurer la carte DSP*

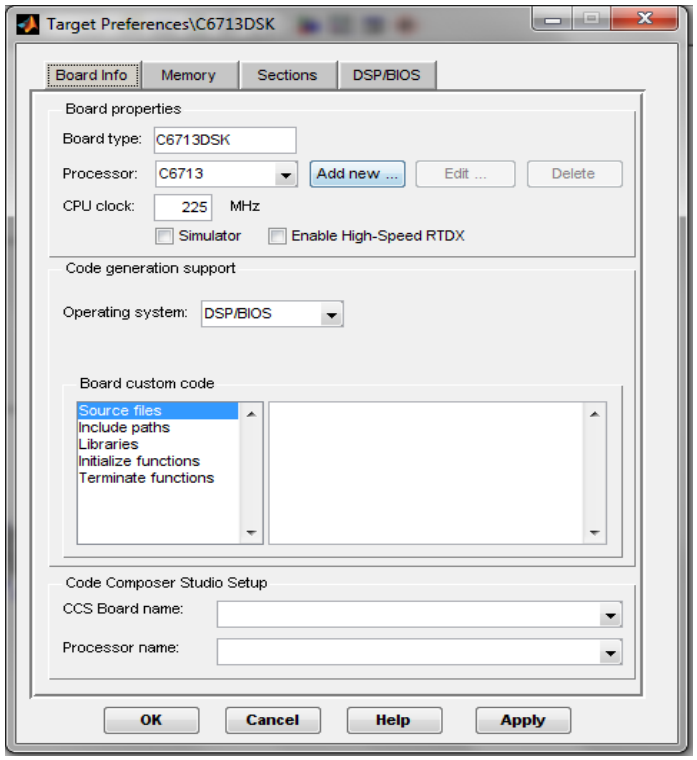


Fig.V.11 : *Fenêtre de configuration de la carte DSP TMS320C6713 DSK*

V.4.5. Ajustement des paramètres du modèle

Comme nous avons dit auparavant, le raisonnement sous Simulink est différent à celui du Matlab notamment lorsqu'on travail en temps réel. C'est pour cela le Simulink à besoin d'un ajustement spécifique.

Afin d'avoir une bonne combinaison Temps de réponse-Qualité, nous avons varié **le nombre d'itérations** et **le seuil** jusqu'à ce que nous avons trouvé des bons résultats.

V.4.5.1. Ajustement de nombre d'itération

Nous avons opté des configurations distinctes pour créer des modèles différents. Les paramètres de ces modèles et les résultats des tests obtenus, en directe et en temps réel, sont présentés dans les tableaux V.21 à V.26.

a/ Premier modèle

Le mot	Nombre d'états du HMM	Mélange de gaussiennes par état	Nombre d'itérations de l'algorithme EM	Coefficients MFCC/trame
Arrière	4	4	**10**	16
Avant	2			
Gauche	4			
Droite	4			
Stop	3			

Tab V.21 : *Paramètres du premier modèle HMM1*

Nous avons prononcé les mots à reconnaitre 30 fois en direct et en temps réel, via un microphone, dans le même milieu de travail (laboratoire) et même locuteur. Les résultats obtenus sont les suivants :

Test 1	30 fois de prononciation En direct		Taux d'erreur
	Reconnu	Non reconnu	
Arrière	30	0	**0%**
Avant	9	21	**70%**
Gauche	30	0	**0%**
Droite	16	14	**46.66%**
Stop	30	0	**0%**
Taux d'erreur global			**23.33%**

Tab V.22 : *Résultats obtenus pour le premier modèle HMM1*

On voit ici que **le taux de reconnaissance global = 76.67%.**

b/ Deuxième modèle

Le mot	Nombre d'états du HMM	Mélange de gaussiennes par état	Nombre d'itérations de l'algorithme EM	Coefficients MFCC/trame
Arrière	4	4	**15**	16
Avant	2			
Gauche	4			
Droite	4			
Stop	3			

Tab V.23 : *Paramètres de deuxième modèle HMM2*

Nous avons prononcé les mots à reconnaitre 30 fois dans les mêmes conditions du premier modèle.

Test 2	30 fois de prononciation En direct		Taux d'erreur
	Reconnu	Non reconnu	
Arrière	29	1	**3.33%**
Avant	14	16	**53.33%**
Gauche	30	0	**0%**
Droite	24	06	**20%**
Stop	30	0	**0%**
Taux d'erreur global			**15.33%**

Tab V.24 : *Résultats obtenus pour le deuxième modèle HMM2*

Pour ce modèle nous avons trouvé un **taux de reconnaissance global = 84.67%.**

c/ Troisième modèle

Le mot	Nombre d'états du HMM	Mélange de gaussiennes par état	Nombre d'itérations de l'algorithme EM	Coefficients MFCC/trame
Arrière	4	4	**20**	16
Avant	2			
Gauche	4			
Droite	4			
Stop	3			

Tab V.25 : *Paramètres de troisième modèle HMM3*

Nous avons prononcé les mots à reconnaitre 30 fois dans les mêmes conditions des modèles précédents.

Test 2	30 fois de prononciation En direct		Taux d'erreur
	Reconnu	Non reconnu	
Arrière	30	0	**0%**
Avant	30	0	**0%**
Gauche	30	0	**0%**
Droite	29	1	**3.33%**
Stop	30	0	**0**
Taux d'erreur global			**0.66%**

Tab V.26 : *Résultats obtenus pour le troisième modèle HMM3*

Le taux de reconnaissance global obtenu = 99.34%.

V.4.5.2 Détermination de seuil

Le seuil est calculé selon la probabilité de vraisemblance (Loglikelihood) trouvée pour chaque mot de la base de test (voir tableau V.27). En revanche, nous avons choisi le **seuil minimum** qu'est égal à 83.93% qui correspond le mot n°39 de la base de test (mot droite).

Mots		Résultats trouvés	Probabilité de vraisemblance calculée pour chaque modèle de Markov de corpus					Seuil
			HMMarr	HMMavt	HMMgch	HMMdrt	HMMstp	
Mot à tester n°	1	**HMMarr**	**-1527,96**	-2538,52	-2302,38	-2279,18	-3137,06	**87,03%**
Mot à tester n°	2	**HMMarr**	**-1698,30**	-2888,41	-2663,63	-2566,00	-3530,20	**87,28%**
Mot à tester n°	3	**HMMarr**	**-1523,93**	-2691,00	-2594,93	-2371,81	-3472,20	**87,96%**
Mot à tester n°	4	**HMMarr**	**-1695,59**	-2886,40	-2665,97	-2470,64	-3539,78	**87,21%**
Mot à tester n°	5	**HMMarr**	**-1607,92**	-2801,34	-2659,58	-2477,32	-3370,66	**87,55%**
Mot à tester n°	6	**HMMarr**	**-1593,70**	-2777,31	-2560,49	-2472,56	-3387,22	**87,54%**
Mot à tester n°	7	**HMMarr**	**-1662,86**	-2895,27	-2613,44	-2446,63	-3469,73	**87,29%**
Mot à tester n°	8	**HMMarr**	**-1550,35**	-2540,44	-2380,89	-2302,67	-3053,39	**86,89%**
Mot à tester n°	9	**HMMarr**	**-1523,75**	-2753,87	-2544,02	-2353,88	-3283,28	**87,77%**
Mot à tester n°	10	**HMMarr**	**-1666,40**	-2827,46	-2662,36	-2469,44	-3506,21	**87,31%**
Mot à tester n°	11	**HMMavt**	-1851,66	**-1339,97**	-1826,95	-1926,04	-2070,21	**85,14%**
Mot à tester n°	12	**HMMavt**	-1926,96	**-1259,25**	-1847,48	-1951,25	-2055,89	**86,07%**
Mot à tester n°	13	**HMMavt**	-2046,73	**-1326,97**	-1846,39	-2015,86	-1804,94	**85,32%**
Mot à tester n°	14	**HMMavt**	-2000,87	**-1294,71**	-1907,38	-2005,52	-1849,07	**85,71%**
Mot à tester n°	15	**HMMavt**	-1889,78	**-1268,45**	-1862,82	-1935,65	-2023,76	**85,88%**
Mot à tester n°	16	**HMMavt**	-2634,07	**-1996,17**	-2856,77	-2997,38	-3290,98	**85,51%**
Mot à tester n°	17	**HMMavt**	-2048,53	**-1291,18**	-1933,38	-2133,13	-2212,04	**86,58%**
Mot à tester n°	18	**HMMavt**	-1933,41	**-1269,03**	-1929,76	-2063,62	-2163,98	**86,44%**
Mot à tester n°	19	**HMMavt**	-2091,51	**-1329,60**	-1947,77	-2103,91	-2195,68	**86,25%**
Mot à tester n°	20	**HMMavt**	-2121,20	**-1350,70**	-1952,72	-2166,99	-2137,48	**86,12%**
Mot à tester n°	21	**HMMgch**	-2306,77	-1798,95	**-1405,12**	-1914,79	-1778,15	**84,73%**

Mot à tester n°	22	**HMMgch**	-2194,30	-1716,34	**-1221,70**	-1769,74	-1745,29	**85,87%**
Mot à tester n°	23	**HMMgch**	-2269,95	-1964,13	**-1409,41**	-2136,73	-2047,15	**85,66%**
Mot à tester n°	24	**HMMgch**	-2327,46	-1848,87	**-1354,58**	-2059,08	-2018,68	**85,90%**
Mot à tester n°	25	**HMMgch**	-2384,66	-1815,64	**-1312,37**	-1993,57	-1980,49	**86,17%**
Mot à tester n°	26	**HMMgch**	-2423,51	-1951,05	**-1381,14**	-2112,41	-1980,11	**85,98%**
Mot à tester n°	27	**HMMgch**	-2423,68	-1965,53	**-1347,27**	-2108,64	-2061,09	**86,40%**
Mot à tester n°	28	**HMMgch**	-2190,61	-1710,90	**-1229,04**	-1825,25	-1728,81	**85,85%**
Mot à tester n°	29	**HMMgch**	-2084,14	-1715,61	**-1256,41**	-1830,32	-1684,23	**85,34%**
Mot à tester n°	30	**HMMgch**	-1980,43	-1793,76	**-1313,02**	-1737,57	-1697,48	**84,59%**
Mot à tester n°	31	**HMMdrt**	-1772,85	-1671,17	-1623,94	**-1262,84**	-1771,76	**84,41%**
Mot à tester n°	32	**HMMdrt**	-1509,47	-1397,83	-1516,31	**-1081,61**	-1614,90	**84,81%**
Mot à tester n°	33	**HMMdrt**	-1627,07	-1616,66	-1584,70	**-1158,80**	-1695,35	**84,92%**
Mot à tester n°	34	**HMMdrt**	-1672,31	-1569,95	-1607,34	**-1244,87**	-1745,49	**84,12%**
Mot à tester n°	35	**HMMdrt**	-1784,87	-1757,30	-1701,65	**-1323,89**	-1812,58	**84,20%**
Mot à tester n°	36	**HMMdrt**	-1897,84	-1951,78	-1930,01	**-1491,28**	-2064,87	**84,03%**
Mot à tester n°	37	**HMMdrt**	-1810,22	-1729,23	-1667,28	**-1310,05**	-1888,18	**84,41%**
Mot à tester n°	38	**HMMdrt**	-1683,61	-1633,28	-1562,95	**-1226,59**	-1722,51	**84,33%**
Mot à tester n°	39	**HMMdrt**	-1833,48	-1782,47	-1733,94	**-1384,79**	-1883,33	**83,93%**
Mot à tester n°	40	**HMMdrt**	-1741,94	-1776,37	-1657,35	**-1314,92**	-1810,67	**84,16%**
Mot à tester n°	41	**HMMstp**	-3103,14	-2593,10	-2761,42	-2693,83	**-1649,53**	**87,11%**
Mot à tester n°	42	**HMMstp**	-1716,42	-1463,46	-1394,18	-1465,77	**-1147,28**	**84,04%**
Mot à tester n°	43	**HMMstp**	-1812,43	-1376,92	-1336,98	-1384,83	**-1002,20**	**85,50%**
Mot à tester n°	44	**HMMstp**	-3271,46	-2717,18	-3220,59	-3235,08	**-2063,95**	**85,77%**
Mot à tester n°	45	**HMMstp**	-3650,55	-2876,57	-3379,99	-3401,14	**-2052,24**	**86,64%**
Mot à tester n°	46	**HMMstp**	-2961,41	-2452,35	-2545,73	-2581,16	**-1620,40**	**86,68%**
Mot à tester n°	47	**HMMstp**	-3446,92	-2761,06	-2994,93	-3067,07	**-1711,65**	**87,76%**
Mot à tester n°	48	**HMMstp**	-3715,81	-3079,47	-3188,38	-3251,75	**-1999,04**	**86,88%**
Mot à tester n°	49	**HMMstp**	-3005,36	-2593,08	-2489,50	-2519,98	**-1687,27**	**86,28%**
Mot à tester n°	50	**HMMstp**	-4218,07	-3481,18	-3819,53	-3565,93	**-2007,02**	**88,26%**

Tab V.27 : *calcul du seuil via la probabilité de vraisemblance*

V.4.6. Résultats et discussion

Pour un nombre d'itérations égales à 20, nous avons trouvé un bon taux de reconnaissance global (99%). Cela vaut dire que l'algorithme ***EM (Expectation Maximization)*** *a pris tout le temps nécessaire pour converger et maximiser toutes les probabilités de vraisemblance de la séquence d'apprentissage suivant leur chemin de Viterbi (Loglikelihood).*

Après des tests effectués et afin d'avoir une bonne qualité de reconnaissance, nous avons refixé le seuil de reconnaissance en ***82,5%*** *à la place*

du minimum trouvé auparavant. Puisqu'un seuil égal à 83.93% a influencé négativement sur notre taux de reconnaissance.

Pour mettre la carte DSP dans les conditions sécuritaires, nous avons préféré d'afficher les résultats de commande sur les 4 LED's de la carte DSP. Ces commandes représentent un numéro binaire varié de 0 à 15.

V.4.7. Configuration et exécution du schéma d'implémentation

Pour compiler et exécuter notre modèle Simulink, on doit suivre les étapes suivantes :

1- Aller à ***Simulation/Configuration Parametres/Solver***, remplacer ***Stop time*** qui est à **10ms** par **inf,** cela vaut dire que notre système exécute le programme jusqu'à l'infinie.

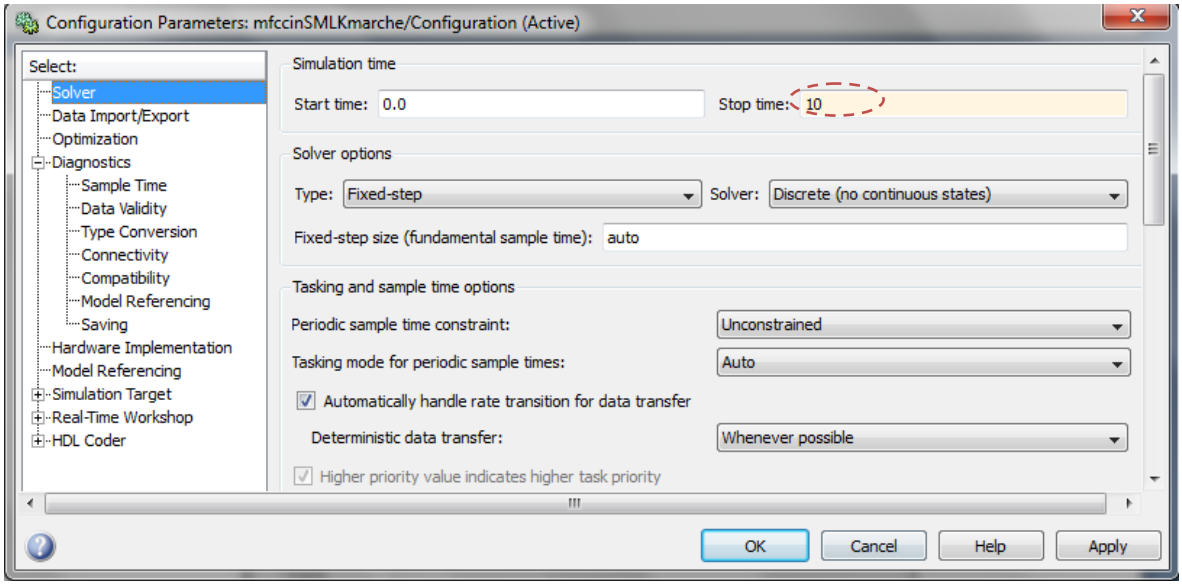


Fig.V.12 : *Configuration du Menu Solver*

2- Aller à ***Simulation/Configuration Parametres/Real-Time Workshop,*** nous voyons que le **System target file** est configuré automatiquement à **ccslink_ert.tlc,** c'est-à-dire qu'il doit générer indirectement le code pour Code Composer Studio afin de lui permettre de l'implémenter sur la carte DSP TMS320C6713DSK.

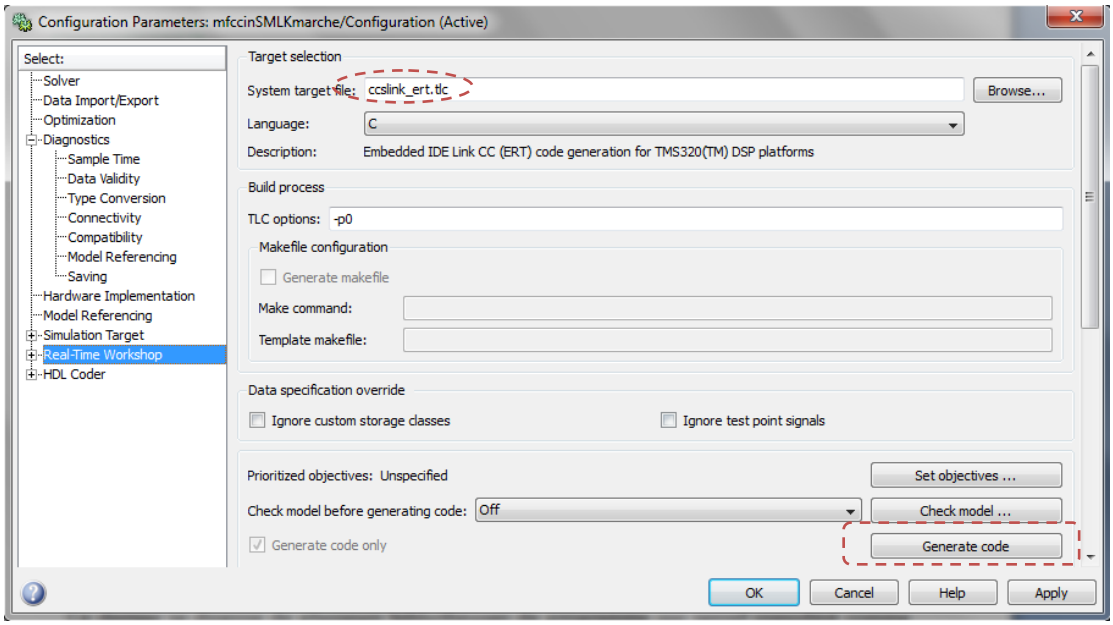


Fig.V.13 : *Configuration du Menu Real-Time Workshop*

3- Cliquer sur l'anglé **Generate code** pour générer le fichier exécutable [.out].

V.5. Exploitation de l'exécutable sur Code Studio Composer CCS 3.3

Le Code Composer Studio (CCS) est un IDE (Integrated Development Environment), ce dernier se dispose de plusieurs bibliothèques de programmes qui seront considérées comme des outils pour réaliser des applications directes.

Le CCS inclus aussi des outils pour la compilation du code en C,C++ ou en assembleur, et un linker pour générer le fichier final *.out* qui sera exécuté par la carte. Il peut aussi simuler l'exécution d'un programme en affichant sur un graphe les données traitées.

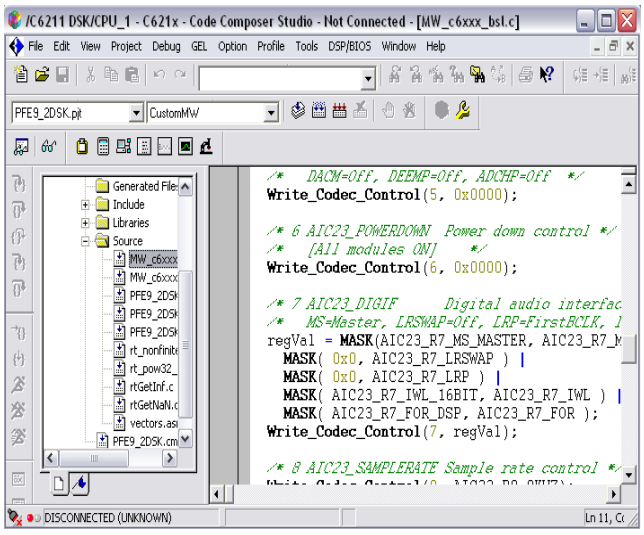


Fig.V.14 *: L'interface code composer*

Le CCS permet de déboguer le programme en temps réel, ainsi il offre une grande facilité pour construire des programmes fiables sur la carte DSP.

V.5.1. Les fonctionnalités de CCS

• Code composer studio permet de charger le programme directement sur la carte est de l'exécuter, instruction par instruction, ou bien d'exécuter jusqu'au point d'arrêt (break point) et permet aussi de visualiser les valeurs des variables de la mémoire aussi et de modifier son contenu idem pour les registres ;

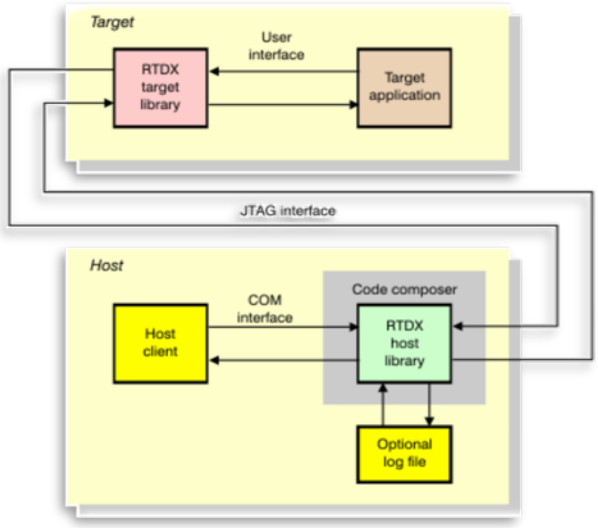


Fig.V.15 : *Schéma de communication Host Target*

• CCS permet l'échange de données entre la carte et le PC, on envoyant des données et on les reçoit dans un fichier de visualisation (probe point- Log file) en temps réel (RTDX) Real Time Data Exchange ;

• La communication entre les différends IDE, que ce soit Matlab, C/C++ et la carte DSP, est assurée par le CCS en utilisant la technologie COM « Component Object Model » comme représentée dans la figure V.15 ;

• CCS peut afficher graphiquement les donner ou le signal acquit par la carte que ce soit dans le domaine temporel ou fréquentiel ;

• CCS permet aussi d'optimiser le Code en C pour réduire le temps d'exécution.

V.5.2. Directive de compilation

Le compilateur C compile des fichiers d'extension [.C] et produit un fichier assembleur [.asm] qui sera traité suivant les étapes présentées dans la figure V.16, afin d'avoir un exécutable.

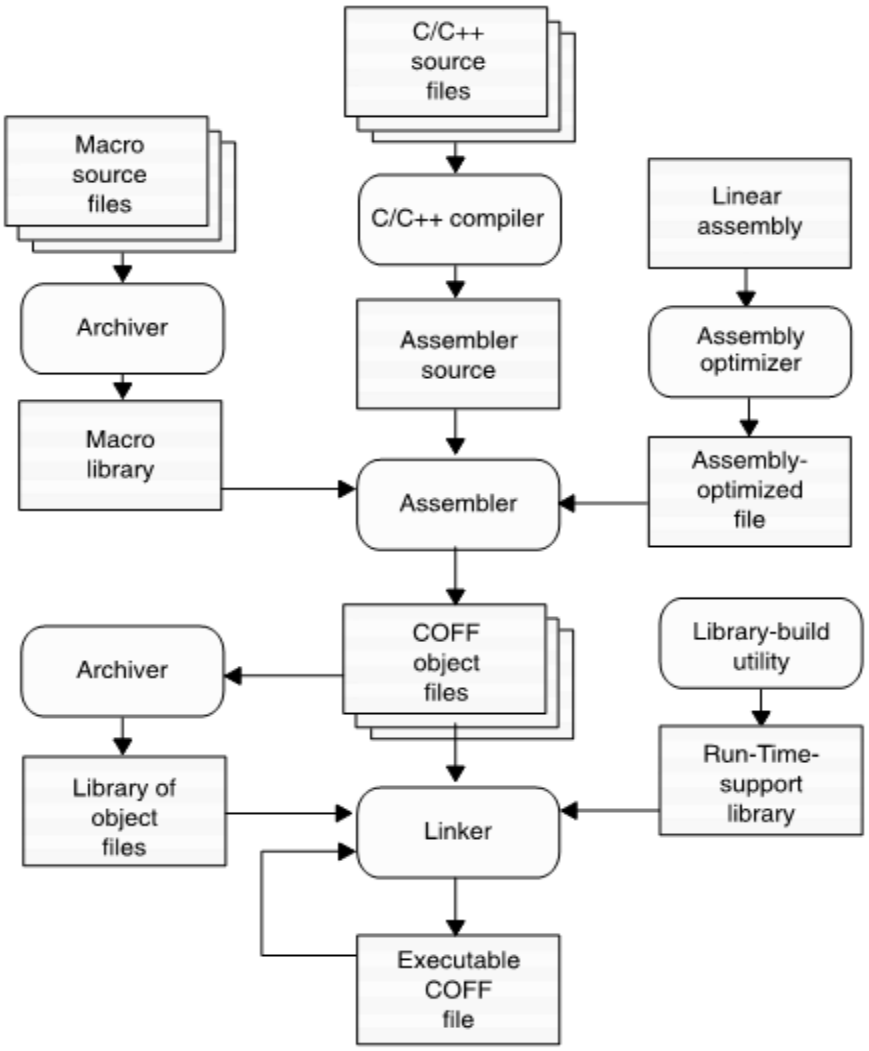


Fig.V.16 : *Diagramme de compilation*

V.6. Application pratique

Pour des raisons de sécurité, nous avons privilégié d'envoyer la commande de sortie via les LED's de la carte DSP.

Nous avons réalisé un circuit électronique qui sera déposé sur les 4 LED's de la carte DSP (Figures V.17 et V.18), ce circuit se dispose de quatre (04) capteurs optiques de type LDR, et des opto-coupleurs pour l'isolation galvanique de la carte DSP et la mannette de commande. La sortie de ces opto-coupleurs peut représenter un bouton poussoir pour les circuits logiques (TTL ou CMOS), ou on peut l'utiliser tel qu'elle est pour des applications diverses comme l'asservissement des moteurs …etc. Pour notre cas on va envoyer directement la sortie vers les interrupteurs de la mannette de commande de la voiture de test (figure.V.19).

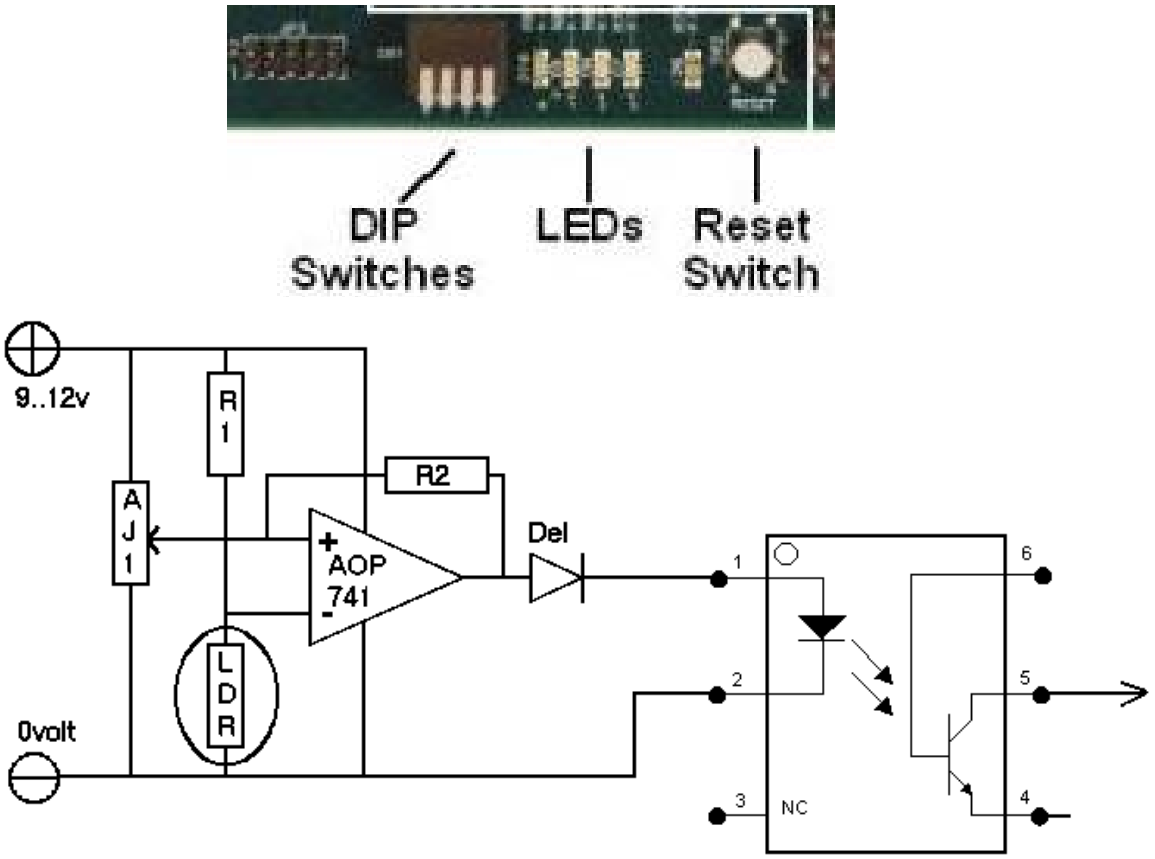


Fig.V.17 : *Schéma d'un détecteur crépusculaire + opto-coupleur*

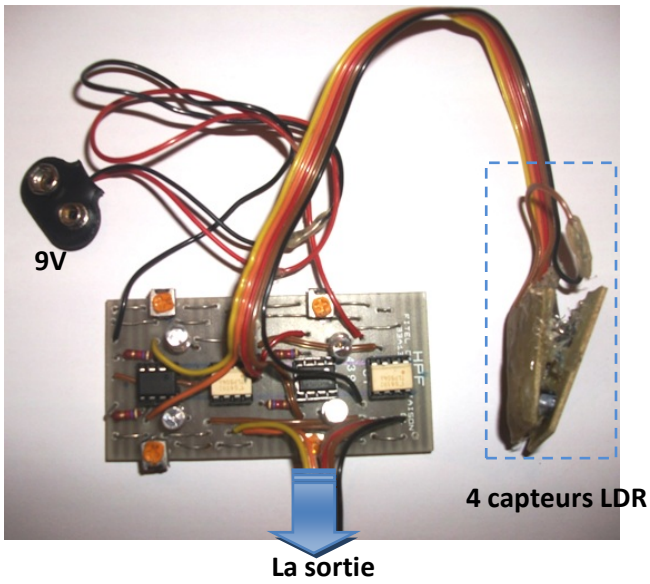


Fig.V.18 : *La carte des capteurs optiques*

Fig.V.19 : *La mannette de commande sans fil et la voiture de test*

V.7. Conclusion

L’écriture d’un tel programme directement en C sur le code composer studio peut demander un temps énorme surtout pour une application de traitement de signal, sans compter les erreurs de logique. Nous avons utilisé une plateforme (Simulink) qui va générer du code toute seule et nous a donné la possibilité de visualiser le déroulement du programme en temps réel, ça va nous faire gagner un temps considérable et par conséquent on se concentre exclusivement sur le traitement du signal.

L’implémentation de notre modèle de reconnaissance des mots isolés par les HMM sur la carte DSP TMS320C6713 nous a donné de bons résultats.

CONCLUSIONS GÉNÉRALES ET PERSPECTIVES

1. Conclusions

Les travaux menés depuis quelques années ont montré que les modèles de Markov cachées (HMM) présentent un intérêt certain pour le TAP, non seulement en reconnaissance de la parole, mais aussi en synthèse à partir du texte, en vérification du locuteur ou en acquisition du langage, essentiellement pour leur capacité d'apprentissage discriminant.

La reconnaissance automatique de la parole a connu un essor incommensurable par l'introduction de la modélisation stochastique, depuis les études établies par J. Fergusson et L. Rabiner.

La modélisation du signal vocal est régie par le choix des vecteurs acoustiques dont les performances doivent être des plus optimales, en matière de la sensitivité au bruit, de complexité de calcul, de degré de perception, etc.

L'utilisation des coefficients MFCC pour la modélisation des mots n'est pas arbitraire. L'extraction de ces coefficients se base sur la perception du signal acoustique comme l'oreille humaine. D'autre part, dans plusieurs travaux de recherche le choix s'est porté sur ces coefficients et par la suite ont montré leur capacité à la discrimination et à la bonne représentation de l'information parole et par suit ils permettent d'obtenir un taux de reconnaissance élevé.

Il est à signaler que le développement d'un système embarqué de reconnaissance de mots isolés à base des HMM est une tâche très délicate et qui nécessite beaucoup d'expérience. De nombreux problèmes se posent en effet concernant le choix de nombre des MFCC, les paramètres HMM à ajuster, le contrôle du système… etc.

Au cours de ce travail, nous avons réalisé un système autonome qui permet de commander vocalement les mouvements d'une voiture. Pour cela nous avons commencé notre travail par faire un rappel sur le signal de la parole et les principaux axes de travail, en s'intéressant de prés à la reconnaissance vocale et particulièrement à la reconnaissance de mots isolés.

Nous avons axé notre étude en premier lieu sur le développement d'un algorithme basé sur les HMM « *Hidden Markov Models* » ceci nous a permis de réaliser la reconnaissance du mot isolé avec un taux de reconnaissance très appréciable avoisinant les 99%. Par la suite, pour rendre notre système ayant une réponse en temps réel nous nous sommes intéresses à la détection du mot de commande noyé dans un milieu bruité, pour cela nous avons exploité l'algorithme de la détection d'activité vocale V.A.D qui nous a donné des résultats très satisfaisants. Et enfin pour valoriser notre travail, on s'est intéressé à la réalisation d'un système embarqué, pour cela on a étudié la carte DSP TMS 320C6713, cette dernière est dédier au traitement du signal ainsi que son logiciel Code Composer Studio qui lui est dédié.

2. Perspectives

L'objectif principal de notre travail, comme nous l'avons déjà signalé, c'était de réaliser de tel système de dialogue H-M en mode mono-locuteur avec un dictionnaire contient cinq mots isolés.

Afin de rendre ce système plus fiable, l'approche est de créer des algorithmes hybrides de reconnaissance automatique de la parole en mode multi-locuteur en utilisant un corpus plus grand, implémenté sur carte DSP ou bien une carte FPGA pour de diverses applications.

BIBLIOGRAPHIE

[1] CALLIOPE F. La parole et son traitement automatique. Masson, 1989.

[2] Guy ALMOUZNI, Traitement De La Parole, Note De Cours « EISTI ». 2009-2010

[3] Adda G., Adda-Decker M., Gauvin J.-L. et Lamel L.-F., Le système de dictée du LIMSI pour l'évaluation AUPELF'97, Journées Scientifiques et Techniques, 1997.

[4] Gagnoulet C, Jouvet D. Développements récents en reconnaissance de la parole. L'écho des recherches. CNET-ENST, N° 135, 1989.

[5] Levinson S, Rabiner L, Sondhi M. Speaker-Independant Isolated Digit Recognition Using Hidden Markov Models. ICASSP, Boston, 1993.

[6] Tubach JP, Gagnoulet C, Gauvain JL. Advances in speech recognition products from France . Conférence Speech Tech, 1989.

[7] Levin E. Word Recognition using Hidden Control Neural Architecture. ICASSP, 1990.

[8] Lippmann RP. Neural Nets for Computing. ICASSP, New-York, 1988.

[9] Lee KF, Hon HW, Reddy R. An Overview of the SPHINX Speech Recognition System . IEEE Trans on ASSP, 38 N° 1, janvier 1990.

[10] Pierrel JM. Utilisation des contraintes linguistiques en compréhension de parole continue le système Myrtille II. TSI, Vol 1, N° 5, 1982.

[11] Pérennou G. The ARIAL II Speech Recognition System In: Haton JP ed, Automatic Speech Analysis and Recognition . 1982.

[12] Averbuch A. et al. Experiments with the TANGORA 20,000 word speech recognizer . ICASSP, Dallas, 1987.

[13] Alto P, Brandetti M, Ferretti M, Maltese G, Scarci S. Experimenting Natural-Language Dictation with 20,000- Word Speech Recognizer , Proceedings of IEEE, CompEuro, Section 2, Hambourg, 1989.

[14] D'Orta P, Ferretti M, Martelli A, Melecrinis S, Scarci S, Volpi G. A Speech Recognition System for the Italian Language . ICASSP, Dallas, 1987.

[15] Cerf-Danon H, de La Noue P, Diringer L, El-Bèze M, Marcadet JC. A 20,000 words, automatic speech recognizer. Adaptation to French of the US TANGORA system , Nato 1990.

[16] Wothke K, Bandara U, Kempf J, Keppel E, Mohr K, Walch G. SPRING Speech Recognition System for German . Eurospeech, Paris, septembre 1989.

[17] Baker J. DRAGON DICTATE (TM) -30K Natural Language Speech Recognition statistical methods. Eurospecch 89 Vol 2, Septembre 1989.

[18] Fanchettc F. Speech recognition. - Talk about progress . Language Technology, N° 19.

[19] Le Breton JL. Le premier traitement de texte vocal. L'ordinateur individuel N° 98, décembre 1997.

[20] Mariani .J. Hamlet un prototype de machine à écrire à entrée vocale, Journal d'acoustique, Paris, 2 Mars 1989.

[21] Lee LS, Tseng CY, Gu HY, Liu FH, Chang CH, Hsich SH, Chen CH A Real- Time Mandarin Dictation Machine for Chinese Language with Unlimited Texts and Very Large Vocabulary . ICASSP, Albuquerque, 1990.

[22] Dreyfus-Graf J., Sonograph and sound mechanics, Journal of the Acoustical Society of America, tome 22-6, 1950.

[23] Davis K., Biddulph R. et Balashek S., Automatic recognition of spoken digits, Journal of the Acoustical Society of America, 1952.

[24] Gravier G., Bonastre J.-F., Geoffrois E., Galliano S., McTait K. et Choukri K., ESTER, une campagne d'évaluation des systèmes d'indexation automatique d'émissions radiophoniques en français, dans XXVème Journées d'Etudes sur la Parole JEP'2004, Morocco, 2004.

[25] http://membres.multimania.fr/guillaumerey/reconnaissance_historique.htm

[26] Bellman R., Adaptive Control Processes: A Guided Tour, Princeton University Press, 1961.

[27] Haton J.-P., Cerisara C., Fohr D., Laprie Y. et Smaili K., Reconnaissance automatique de la parole, Dunod, 2006.

[28] Tubach J., rédacteur, La parole et son traitement automatique, Masson, collection technique et scientifique des télécommunications, édition 1989.

[29] Tremain T.-E., The government standard Linear Predictive Coding algorithm: LPC10, Speech Technology Magazine, tome 1-2, 1982.

[30] Miet G., Towards wideband speech by narrowband speech bandwidth extension: magic effect or wideband recovery ?, Thèse de doctorat, University of Maine, 2001.

[31] Davis S.-B. et Mermelstein P., Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1980.

[32] Hermansky H., Perceptual Linear Predictive (PLP) analysis of speech, Journal of the Acoustical Society of America, 1990.

[33] Young S., Large vocabulary continuous speech recognition: A review, dans Automatic Speech Recognition and Understanding Workshop, IEEE, ASRU'1995, Snowbird, Utah, USA, 1995.

[34] Woodland P.-C. et Young S.-J., The HTK tied-state continuous speech recognizer, Proceedings of the 3rd European Conference on Speech Communication and Technology (Eurospeech'1993), Berlin, Germany, 1993.

[35] Hermansky et al, Perceptually Based Linear Predictive Analysis of Speech, 1985.

[36] Brian, K et Morgan, N. Recognizing reverberant speech with RASTA-PLP. In ICASSP, volume 2, Munich, Germany, IEEE, April 1997.

[37] Hermansky, H., Morgan, N., Bayya, A, and Kohn, P. RASTA-PLP Speech Analisys, ICSI Technical Report TR-91-069, Berkeley, California, 1991.

[38] Kedem B., Spectral analysis and discrimination by zero-crossings, Proceedings of the IEEE, 1986.

[39] Taboada J., Feijoo S., Balsa R. et Hernandez C., Explicit estimation of speech boundaries, Proceedings of the IEEE Science, Measurement and Technology, 1994.

[40] Psutka J., M¨ uller L. et Psutka J.-V., Comparison of MFCC and PLP parameterizations in the speaker independent continuous speech recognition task, Proceedings of the 7th European Conference on Speech Communication and Technology (Eurospeech'2001), Aalborg, Danemark, 2001.

[41] Chen C.-P., Bilmes J. et Ellis D.-P.-W., Speech feature smoothing for robust ASR, Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP'2005), Philadelphia, Pennsylvania, USA, 2005.

[42] Lévy C., Linarès G. et Nocera P., Comparison of several acoustic modeling techniques and decoding algorithms for embedded speech recognition systems, Workshop on DSP in Mobile and Vehicular Systems, Nagoya, Japan, 2003.

[43] Gas B., Zarader J.-L. ET Chavy C., A new approach to speech coding: the neural predictive coding, Journal of Advanced Computational Intelligence and Intelligent Informatics, 2000.

[44] Paliwal K.-K. et Atal B.-S., Efficient vector quantization of LPC parameters at 24 bits/frame, IEEE Transactions on Speech and Audio Processing, 1993.

[45] Schwartz R., Anastasakos T., Comparative experiments on large vocabulary speech recognition, Proceedings of the ARPA workshop on Human Language Technology, Princeton, NJ, USA, 1993.

[46] Chen C.-P., Bilmes J. et Kirchhoff K., Low-resource noise-robust feature post-processing on Aurora 2.0, Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP'2002), Denver, Colorado, USA, 2002.

[47] Bellman R., Dynamic programming, Princeton University Press, 1957.

[48] Jelinek F., Continuous speech recognition by statistical methods, dans Proceedings of the IEEE, 1976.

[49] Bourlard H. et Wellekens C.-J., Multi-layer perceptrons and Automatic Speech Recognition, Proceedings of the IEEE First Annual International Conference on Neural Networks, San Diego, 1987.

[50] Bourlard H. et Wellekens C.-J., Links between Markov models and multi-layer perceptrons, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1990

[51] Waibel A., Hanazawa T., Hinton G., Shikano K. et Lang K., Phoneme recognition using Time-Delay Neural Networks, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1989.

[52] Rabiner L.-R., A tutorial on Hidden Markov Models and selected applications in speech recognition, dans IEEE Transactions on Speech and Audio Processing, tome 77-2, 1989.

[53] Haton J.-F. M. A.-P., Automatic word recognition based on secondorder hidden markov models, Proceedings of the 3rd International Conference on Spoken Language Processing (ICSLP'1994), Yokohama, Japan, 1994.

[54] Lee C.-H., Juang B.-H., Soong F.-K. et Rabiner L.-R., Word recognition using whole word and sub word models, Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP'1989), Glasgow, UK, 1989.

[55] Viterbi A., Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory, 1967.

[56] Bourlard et Wellekens, Speech Pattern discrimination and multilayer Perceptrons.1989.

[57] Robinson, A recurrent error propagation network speech recognition system, 1991.

[58] Salomon, Kernel Machines for phoneme Classification, 2002.

[59] Vladimir N.Vapnik, Statistical Learning Theory, IEEE transactions on neural networks, vol. 10, no. 5, september 1999.

[60]Chih-Wei Hsu et Chih-Jen, Lin A Comparison of Methods for Multi-class Support Vector,IEEE,vol13, 2002.

[61] J. Platt, Probabilities for SV machines, MIT press, 2000.

[62] Bazzi et Katabi, Using Support Vector Machines for Spoken Digit Recognition.In International Conference on Spoken Language Processing, Beijing, China, October 2000.

[63] Woo-Yong Choi, Dosung Ahn, Sung Bum Pan, Kyo Il Chung, Yongwha Chung, Sang-Hwa Chung, SVM-Based Speaker Verification System for Match-on-Card and Its Hardware Implementation. ETRI Journal, Vol 28, N 3, 2006.

[64] Smith et Gale, Speech Recognition using SVM. MIT Press, 2002.

[65] Carbonell N., Damestoy J.-P., Fohr D., Haton J.-P. et Lonchamp F., APHODEX, design and implementation of an acoustic-phonetic decoding expert system, Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP'1986), Tokyo, Japan, 1986.

[68] Campbell et al, SVM Based speaker verification using a GMM supervector kernel and nap variability compensation. IEEE-ICASSP,Toulouse, 2006.

[69] S.E. Fienberg, When did Bayesian Inference become "Bayesian"? Bayesian Analysis, pp. 1-40, 2005.

[70] S.M. Stigler « Thomas Bayes' Bayesian Inference, » Journal of the Royal Statistical Society, Series A, 145:250-258, 1982.

[71] H.L. Van Trees, Detection, Estimation and Modulation Theory, Part I, MIT Press, Cambridge, MA, 1968.

[72] Huelsenbeck, J. P.; Crandall, K. A. "Phylogeny Estimation and Hypothesis Testing Using Maximum Likelihood". Annual Review of Ecology and Systematics 28: 437–466, 1997.

[73] Jongseo S,Wonyong S , A voice activity detector employing soft decision based noise Spectrum adaptation, School of Electrical Engineering, Seoul National University, Seoul, Korea. IEEE, 1998.

[74] R.Rakotomalala, Tests de normalité,Techniques empiriques et tests statistiques,version 2, Université Lumière Lyon, 2011.

[75] E.F, The Generation and Application of Random Numbers , Forth Dimensions Vol XVI Nos 1 & 2, Forth Interest Group, Oakland California, 1994.

[76] H.A. David, Order Statistics,Wiley series in probability and mathematical statistics, 2ème edition, pages 8-13,1981.

[77] J.Aldrich, "R. A. Fisher and the making of maximum likelihood 1912–1922". Statistical Science 12 (3): 162–176. 1997.

[78] D. Johnson, Minimum Mean Squared Error Estimators, Version 1.5, Nov 22, 2004.

[79] http://www.creatis.insa-lyon.fr/~yougz/tsi/dsp.ppt

[80] http://www.ti.com

[81] Christophe. G , Baudry. M. Etude de la paramétrisation du signal de parole à partir de représentations en ondelettes, Université de Paris 11, Orsay, France, 1995.

[82] http://www.dragon-medical-transcription.com

[83] L. Zouari-beltaifa, Vers le Temps Réel en Transcription Automatique de la Parole Grand Vocabulaire, thèse de doctorat, 2007.

[84] D.O. Shaughnessy, Speech Communication: Human and Machine. India University Press, 2001.

[85] W.J.Picone., Signal modeling techniques in speech recognition, proceeding of IEEE, vol. 81, no.9:1215-1247, 1993.

[86] Q.C.Nguyen, Reconnaissance de la parole en langue vietnamienne, thèse de doctorat, Institut National polytechnique de Grenoble, 2002.

[87] Lévy C., Modèles acoustiques compacts pour les systèmes embarqués, thèse de doctorat, l'Université d'Avignon et des Pays de Vaucluse, 2006.