Introduction

Le Master Recherche *Signal, Image, Acoustique et Optimisation* comprend un stage de 5 mois minimum en entreprise ou laboratoire pour y effectuer un travail orienté « recherche ». Conformément aux particularités du parcours « Signal et Acoustique » que j'ai choisi pour le Mastère, et en raison de mon vif intérêt pour l'Acoustique Musicale et la Synthèse sonore, je me suis naturellement tourné vers l'Institut de Recherche et Coordination Acoustique/Musique pour ma recherche. C'est ainsi que j'ai été admis dans l'équipe Analyse et Synthèse de Sons de l'IRCAM, dirigée par Mr. Xavier Rodet, pour y effectuer sous sa direction un stage du 1er février 2005 au 15 juillet 2005 en synthèse de la parole, domaine qui m'intéresse tout particulièrement.

Le stage concernait le projet *Talkapillar* de synthèse de parole par concaténation d'unités diphoniques et la tâche principale qui m'était assignée était l'optimisation de la fonction de sélection des unités. Néanmoins, étant impliqué globalement dans ce projet, j'ai également participé à d'autres travaux liés à ma tâche principale. Ce document présente un rapport de ces différentes activités de recherche et développement sur le système Talkapillar.

Dans une première partie vous trouverez une vue d'ensemble du stage, notamment sur les conditions de travail et le contenu effectif. Puis, je vous présenterai une brève introduction à la synthèse concaténative afin d'introduire les caractéristiques du projet Talkapillar. Ensuite je vous exposerai dans deux parties séparées les travaux que j'ai effectués, en partant de la problématique et en arrivant aux résultats effectifs. Enfin le bilan du stage conclura ce rapport afin de déterminer ce que ce stage m'a apporté et ce que j'ai pu à apporter à l'IRCAM.

I. Présentation du stage

Après une brève introduction aux activités de l'IRCAM, je présenterai l'organisation adoptée durant ce stage avant d'exposer son contenu effectif.

1. L'IRCAM

L'*Institut de Recherche et Coordination Acoustique/Musique* a été initié en 1969 par Georges Pompidou qui en confie la direction au compositeur et chef d'orchestre Pierre Boulez. L'Ircam, associé au centre Georges Pompidou et placé sous la tutelle du Ministère de la Culture et de la Communication, devient dès lors un centre dédié à la recherche et la création musicale contemporaine, ou se rencontrent art et science pour élargir l'instrumentarium et renouveler le langage musical.

L'Ircam regroupe près de 90 scientifiques menant des recherches fondamentales sur les apports des mathématiques, de l'acoustique, de l'informatique et de la physique à la création musicale. C'est le plus gros centre de recherche scientifique au monde entièrement dédié aux technologies pour la création musicale.

Le propre de l'Ircam est de susciter l'accueil et la coordination de points de vue scientifiques variés sur le phénomène musical, issus de la physique (acoustique, mécanique …), du traitement du signal, de l'informatique, de la psychologie cognitive, de la musicologie. Le mode d'organisation mis en oeuvre à cet effet repose sur une répartition thématique des activités entre équipes spécialisées, chacune d'elles intégrant l'ensemble de la chaîne de travaux correspondant à son domaine : recherche, développement logiciel, contrats et projets extérieurs, diffusion.

Parmi elles, l'équipe Analyse/Synthèse, dirigée par Xavier Rodet, effectue des activités de recherche et de développement autour de l'analyse, la transformation et la synthèse des signaux sonores. L'analyse de sons comprend les méthodes permettant l'extraction ou la structuration automatique de divers types d'informations provenant du signal, comme la fréquence fondamentale ou les évolutions spectrales déterminant la hauteur et le timbre du son perçu. Des informations non strictement musicales sont également prises en compte et intéressent des domaines tels que l'acoustique industrielle, le design sonore et le multimédia.

Les techniques de transformation et la synthèse des sons sont d'abord conçues pour répondre aux demandes des musiciens pour la création de nouveaux sons et de nouvelles musiques. Les travaux de l'équipe, issus de la coordination des activités des chercheurs et des développeurs, résultent en la création d'outils informatiques à interfaces graphiques adaptées aux utilisateurs (compositeurs, ingénieurs du son, amateurs…).

En particulier, de nombreux travaux effectués ou en cours au sein de l'équipe ont trait au signal vocal :

- son *analyse*, par exemple l'extraction d'information sur la prosodie du Français (travaux réalisés actuellement par Gregory Beller dans le cadre d'une thèse)
- sa *transformation*, en particulier extraction de l'identité de la voix d'un locuteur pour l'appliquer à un autre locuteur (thèse de Fernando Villavicencio)
- sa *synthèse*. C'est dans ce dernier registre que j'ai été impliqué lors de mon stage.

2. Contexte de travail

L'équipe Analyse/Synthèse de l'Ircam était alors composée d'une quinzaine de scientifiques, dont 2 personnes travaillant également sur la synthèse de parole avec Talkapillar :

- Thomas Hueber, élève ingénieur CPE Lyon, effectuant une année de césure à l'Ircam de septembre 2004 à septembre 2005. Il travaille sur le développement de Talkapillar.
- Grégory Beller, en stage de Mastère Recherche ATIAM de l'IRCAM d'avril 2005 à août 2005. Ce stage se prolongera par une thèse dans le domaine de la musicalité de la voix et de la prosodie en septembre 2005. Il a participé préalablement au développement de Talkapillar et se sert de cet outil pour sa thèse.

Mr. Rodet supervisait le projet, donnait les orientations, et m'encadrait également d'un point de vue pédagogique et scientifique.

Concernant les conditions de travail, j'ai été muni dès le départ d'un PC relié au réseau de l'IRCAM et disposant du système opératoire Linux et de la distribution Fedora Core 2, muni de tous les accessoires et software nécessaires à mon travail (compte réseau et Internet, carte son, casque …). Le langage de programmation Matlab était à la base du projet. Je disposais également d'une grosse base de donnée gérée par PostGreSQL8.

Les horaires de travail des chercheurs étant assez décalés par rapport à ceux que je pratique habituellement, j'ai eu à trouver un compromis avec les horaires de mes collègues (commençant et finissant régulièrement tard) tout en travaillant le plus possible le matin. J'ai donc adopté les horaires de travail 9h30-18h30 avec une pause d'une heure et quelques adaptations en fonction du travail et des réunions.

A propos de ma méthode de travail, elle a cumulé les caractéristiques d'une organisation de type « recherche » (autonomie) avec celles d'un travail de « développement » en équipe. En outre, je rapportais régulièrement l'avancé de mes travaux et de mes réflexions à Mr. Rodet.

En marge de mon travail sur Talkapillar, j'ai également eu la chance de pouvoir assister et participer à un grand nombre de conférences et séminaires variés dans le cadre de l'équipe ou de l'IRCAM en général:

- Présentation des travaux de l'équipe à mon arrivée
- Conférences publiques du Cycle voix proposées par l'Ircam.
- Séminaire sur les applications vocales développées par l'équipe à l'usage des compositeurs
- Création d'un projet RIAM (réseau pour la Recherche et l'Innovation en Audiovisuel et Multimédia) en partenariat avec France Télécom et des partenaires en doublage de films, conception de jeux vidéo…
- Démonstrations du système Talkapillar

3. Contenu du stage

Un système de synthèse de la parole à partir du texte par corpus et sélection d'unités est développé dans l'équipe. La synthèse utilise un très large corpus de parole structuré dans une base de données PotsgreSQL. Pour synthétiser une phrase, le système sélectionne dans ce corpus des unités (phones, diphones, syllabes, etc.) qui, concaténées, donnent la parole voulue.

La qualité du résultat dépendait en grande partie de la sélection des unités optimales en fonction du contexte. La qualité d'une sélection est mesurée par une fonction de coût dépendant de différents aspects pondérés par des "poids". Mon principal travail a alors été de trouver ces poids pour que la fonction de coût corresponde à la qualité perçue de la parole synthétisée.

En outre, l'un des importants facteurs de qualité est que la base contienne des unités bien connues et bien découpées. Mon premier travail a été de travailler sur la segmentation des unités car c'était à mon arrivée l'un des principal défauts du système.

Cette première tâche s'est étalée sur les deux premiers mois avant que je me focalise sur l'apprentissage. Cependant j'ai également pris une part active au développement global de Talkapillar, à l'élaboration où la modification de certains modules. Ce qui suit reflète mon travail avec une optique chronologique.

II. La synthèse vocale par concaténation d'unités

1. Etat de l'art en Synthèse à partir du texte

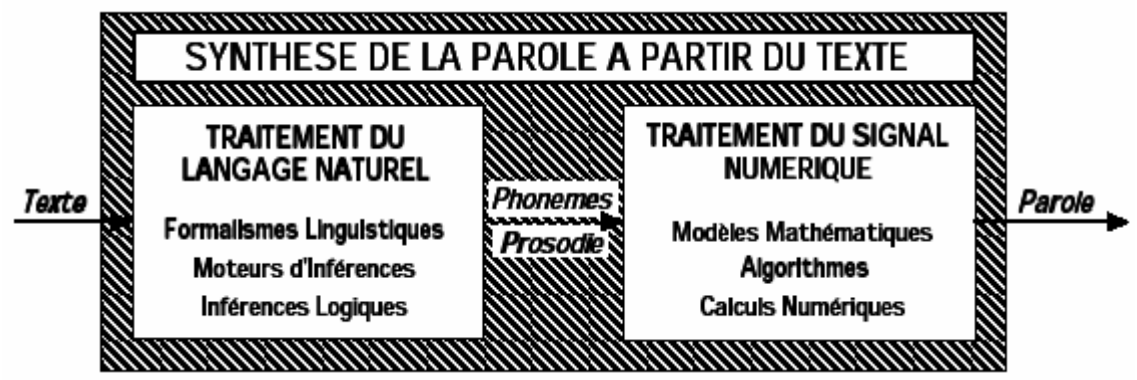
Un système Text-To-Speech (TTS) permet la prononciation artificielle d'une phrase à partir de sa transcription littéraire seulement, sans avoir été préalablement prononcée. Cette technique permet donc, pour la synthèse, de contourner la complexité que pourrait représenter une modélisation du conduit vocal. [Flanagan, 1972] présente les différentes techniques mécanique, puis électronique et informatique alors en vue pour la synthèse vocale, abandonnées depuis.

En effet, le signal vocal est le résultat d'un processus complexe de stimulation du conduit vocal (et parfois nasal), produisant 2 types de sons dont la combinaison donne naissance au langage :

- Les voyelles : Elles sont produites grâce à la vibration des cordes vocales (voisement) et à la résonance du conduit vocal dont les 3 premières fréquences sont appelées « formants »
- Les consonnes : Contrairement aux voyelles, elles ne sont pas exclusivement voisées et ne sont pas nécessairement réalisées avec une configuration stable du conduit vocal (fricatives produites par un flux d'air turbulent au niveau d'une constriction du conduit vocal, plosives réalisées par une fermeture puis ouverture du conduit vocal…).

Un mot est donc l'enchaînement d'un grand nombre de processus physiques complexes très difficiles à modéliser par un système de synthèse (mécanique des fluides, non-linéarité du comportement des tissus muqueux …).

Un système TTS va se baser sur un autre procédé de création de voix, sans aucune analogie avec le procédé naturel. Son fonctionnement se découpe en deux blocs :



- un bloc de traitement du langage naturel, capable de produire la transcription phonétique de la phrase à lire et d'y associer une intonation et un rythme naturels
- un module de traitement du signal qui transforme cette information symbolique en signal de parole.

Ce dernier module, homologue de l'appareil phonatoire, doit prendre en compte des contraintes articulatoires, car on sait depuis longtemps que les transitions phonétiques contribuent plus à l'intelligibilité du signal vocal que les zones stables des unités phonétiques, les *phonèmes*. Cette considération a ouvert la voix à deux types de « philosophies » TTS :

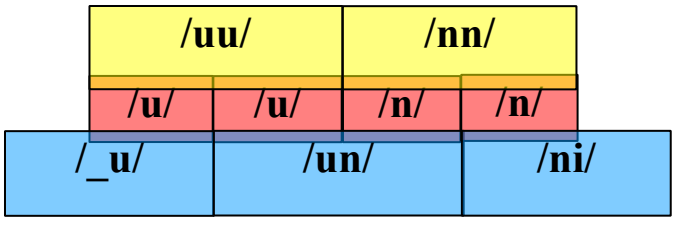
- *Explicite,* sous la forme d'une série de règles décrivant formellement l'influence des phones les uns sur les autres
  → La **synthèse par règles** consiste à analyser une grosse quantité d'enregistrement de diverses combinaisons de phonèmes afin de généraliser des comportements entre eux. Une phrase à prononcer va donc déclencher une suite de paramètres, conformément à ces règles, qui seront transmises à un calculateur. Ce procédé a par la suite laissé la place au suivant, plus performant et facile à implémenter.

- *Implicite,* en enregistrant des exemples d'unités phonétiques dans une base de données de segments de parole, et en les utilisant tels quels comme unités de parole.
  → La **synthèse par concaténation** qui ne nécessite pour une synthèse fluide qu'une étape de concaténation d'unités enregistrées dans une grande base de données, sans règles. Ce type de synthèse peut donc reposer sur la concaténation de diverses unités phonétiques, comme le montre l'exemple suivant pour le mot « *unité* »:

Phones: /uu/ /nn/ /ii/ /tt/ /éé/

Semi-phones: /u/ /u/ /n/ /n/ /i/ /i/ ...

Diphones: /_u/ /un/ /ni/ /it/ /té/ /é_/



Or, comme nous l'avons vu précédemment, c'est la nature transitoire des phonèmes entre eux qui apporte le plus de clarté à l'élocution, d'où une utilisation répandue de la **concaténation de diphones,** sur laquelle a porté principalement mon travail, bien que Talkapillar puisse également effectuer de la concaténation de phones et de semi-phones et sans connaissance de la nature des signaux concaténés entre eux. C'est d'ailleurs pour cette raison que le système Talkapillar est basé sur un système de synthèse musicale par concaténation, comme nous allons le voir maintenant.

2. Le projet TALKAPILLAR

Le projet Talkapillar est l'adaptation à la synthèse de la parole du système *Caterpillar* de synthèse musicale concaténative, issu de la thèse de Diemo Schwarz [Schwarz, 2004]. Le but à atteindre est la synthèse de texte avec une élocution donnée et différentes émotions naturelles, puisque extraites d'un gros volume d'échantillons de voix naturelles. Nous verrons par la suite le procédé utilisé pour extraire non seulement des unités « optimales » pour une phrase donnée, mais également une prosodie optimale.

Au départ le projet devait servir à utiliser un enregistrement du discours prononcé par Jean Cocteau lors de son entrée à l'Académie Française pour reconstituer sa voix et s'en servir comme voix-off pour le récit de sa vie dans un documentaire lui étant consacré. Par extension, les travaux menés dans le cadre de Talkapillar ne se sont pas bornés uniquement à cet objectif, mais se sont ouverts à toutes les possibilités que le produit fini pourrait représenter pour des compositeurs. En effet, de nombreux musiciens professionnels sont intéressés par les applications que pourrait avoir un système vocal capable de sélectionner des unités selon des critères non seulement d'intelligibilité mais également de musicalité.

En raison des spécificités du signal de parole cette migration a nécessité des changements fondamentaux dans le système :

- Le premier objectif d'un synthétiseur de parole est la clarté de l'élocution, l'intelligibilité des enchaînements de signal afin de faire comprendre une information. Cette notion est absente en synthèse musicale où le critère prépondérant est esthétique.
- La base de signal de parole nécessite l'ajout de nouveaux critères pour caractériser chacune des unités, par rapport au signal de musique. En effet, il faut incorporer tous les aspects grammaticaux, sémantiques, en plus de tous les descripteurs utilisés pour la musique.

Le système Talkapillar repose sur la création puis la segmentation d'une base de parole continue en de multiples unités acoustiques propres à la voix. La synthèse d'une voix se fait ensuite par sélection optimale d'unités de ce corpus avant de les concaténer pour donner le signal de parole artificiel.

La qualité de la synthèse finale dépend donc de deux principaux aspects, qui ont chacun donné lieu à un travail séparé lors de mon stage :

- La qualité de la base de parole : elle doit être parfaitement découpée en unités phonétiques décrites, sans aucune erreur, avec le plus possibles de caractéristiques
- L'optimalité de la synthèse : une phrase en sortie du système ne doit comporter aucune erreur de sélection et présenter le meilleur résultat possible sur l'ensemble de la base à partir des données de l'utilisateur (phrase à synthétiser, prosodie …)

III. Optimisation de la segmentation de la base de parole

1. Le corpus

A. L'enregistrement

Etant donné que la synthèse finale est une concaténation d'éléments de cet enregistrement initial, il est primordial que sa qualité soit parfaite, sans artefacts non désirés et perceptibles à la concaténation.

L'enregistrement initial de Jean Cocteau, base de notre travail, n'étant pas d'assez bonne qualité (réverbération, bruits ambiants, qualité du support audio…), il a été décidé de commencer le travail sur un enregistrement « propre » pour ne pas gêner l'élaboration des algorithmes. Le corpus sur lequel j'ai travaillé est un enregistrement de ce discours prononcé par Xavier Rodet dans la chambre anéchoïque de l'Ircam. Cette pièce permet un rapport signal sur bruit optimal et une qualité CD grâce à une réverbération quasiment nulle.

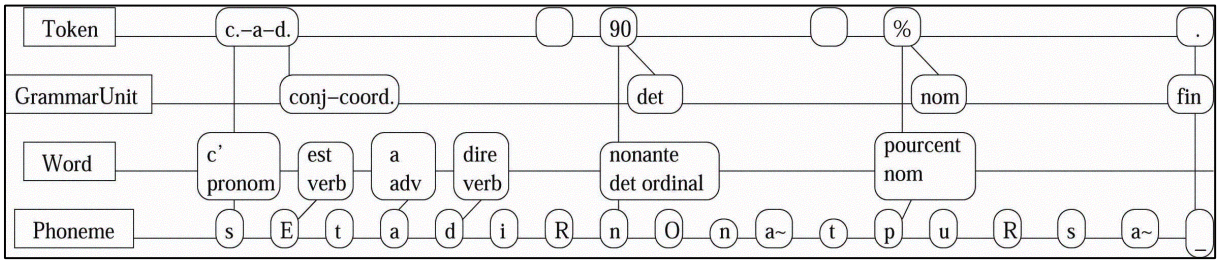
Le résultat est un enregistrement de 1H30, correspondant à 1153 phrases, pas forcément au sens grammatical du terme mais en tant que segments cohérent issus du texte original, ceci afin de favoriser des phrases courtes plus facilement exploitables.

La prochaine étape consiste à faire coïncider l'audio enregistré avec les phrases transcrites. Pour cela, il est tout d'abord nécessaire de faire intervenir le module de traitement du langage naturel (cf. II 1.) pour extraire des données des phrases transcrites et faire le lien avec le discours.

B. Analyse textuelle du Français par EULER

L'analyse du langage naturel, c'est-à-dire de la phrase textuelle en elle même, est effectuée par le module Euler [Bagein, 2001]] développé à l'Université Polytechnique de Mons. Il fourni un grand nombre d'informations sur la phrase traitée, sans aucune utilisation de dictionnaire. Cette analyse est effectuée sur la base de règles établies sur la langue française et appliquée à une chaîne de caractère seule. Il en résulte un grand nombre de **descripteurs symboliques** pour chacune des unités constituant la phrase.

L'analyse d'Euler repose sur la création de modules MLC (*Multi Layers Containers*), en extrayant séparément des données syntaxiques, grammaticales, linguistiques et phonétiques à partir d'une succession de chaînes de caractères séparées par des espaces. En plus d'une connaissance approfondie de la langue française, ce système gère également la reconnaissance et l'analyse de nombreuses représentations symboliques. Voici un exemple de constitution d'une MLC :



A chaque occurrence d'un mot est associé une valeur grammaticale, linguistique puis phonétique, sous la forme d'une suite de caractère tirée du code phonétique X-SAMPA, compréhensible par le reste du programme car n'utilisant que des caractères informatiques usuels.

Le lecteur pourra se référer à l'Annexe 1 pour y trouver un exemple de fichier présentant la transcription en X-SAMPA et les informations retournées par Euler après l'analyse d'une phrase donnée :

Ces informations sont :

- Décomposition de la phrase en entités linguistiques avec une description grammaticale (nature, temps des verbes, ponctuation…)
- Décompositions de la phrase en unités phonétiques : phones (unités phonétique de base), semi-phones (« moitié » de phone), diphones (correspondant à la transition entre deux phones, combinaison de deux semi-phones consécutifs issus de deux phones consécutifs).
- Extraction de données prosodiques symboliques à partir de l'analyse des **syllabes** à l'échelle de la phrase (accentuations, structure de la phrase, intonations …)

Cette analyse permet d'extraire de chaque phrase 2 types d'unités utilisées par la suite pour la segmentation et la sélection :

- Des unités segmentales de type semiphone, phone ou diphone munies de descripteurs symboliques issus leur nature et de leur contexte textuel
- Des unités supra segmentales appelées **groupes prosodiques,** représentatives de la « musicalité » [Beller, 2005] de la phrase entière ou d'un morceau de cette phrase. Il permet de créer des unités descripteurs prosodiques de longueurs variables qui permettront par la suite d'ajouter une description acoustique aux unités segmentales.

2. Segmentation automatique du corpus

Une fois le texte découpé en unités, il faut faire correspondre l'audio avec l'analyse textuelle en indiquant clairement à quel endroit de l'audio se situent les frontières entre unités segmentales : c'est la **segmentation**. De soudains changements dans le spectre du signal ou dans son amplitude sont souvent le signe de frontières de segments. Néanmoins, ces frontières ne sont pas des indices fiables à cause du phénomène de coarticulation (influence des phonèmes les uns sur les autres).

Une meilleure possibilité consisterait à découper manuellement l'ensemble du discours en associant à chaque fichier audio un autre fichier contenant des marques temporelles aux endroits précis de ces transitions. Même si cette solution permettrait un découpage avec très peu d'erreurs, elle n'est pas applicable dans le cas de notre corpus qui comporte 1153 phrases dans autant de fichier audio. Une méthode de segmentation automatique est donc utilisée dans le projet, présentée dans ce qui suit.

A. Principe

Il s'agit ici de comparer chaque unité segmentale d'une phrase à une bibliothèque d'unités préenregistrées, grâce à une mesure de ressemblance acoustique et perceptive.

En plus du discours de Jean Cocteau, il a donc été enregistré 3 autres fichiers audio, appelés *Bootstrap*, par Mr. Rodet prononçant l'ensemble des diphones possibles en Français, c'est-à-dire toutes les combinaisons de phones :

- Consonne/Voyelle (ex. : « popop » pour les diphones /po/ et /op/)
- Double consonne (ex. : « bléblébl » pour le diphone /bl/)
- Double voyelle (ex. : « iadia » pour le diphone /ia/)

Nous avons ensuite découpé ces fichier audio manuellement comme spécifié précédemment, en ajoutant des marques temporelles grâce au logiciel Xspect développé à

l'Ircam. Cette étape est nécessaire pour s'assurer d'avoir un Bootstrap parfaitement découpé et avoir des correspondances parfaites entre chaque diphone et le son associé.

Pour la segmentation de chaque phrase du corpus, appelée *phrase cible*, on la synthétise en concaténant les diphones correspondant du Bootstrap à partir de son analyse textuelle préalable : pour chaque diphone de cette phrase l'audio correspondant du Bootstrap est utilisé. On obtient alors une deuxième occurrence de la phrase de très mauvaise qualité mais avec un découpage connu et précis : c'est la *phrase source*. La segmentation automatique consiste alors à faire correspondre les deux fichiers audio afin de déterminer les frontières entre diphones dans la phrase originale.

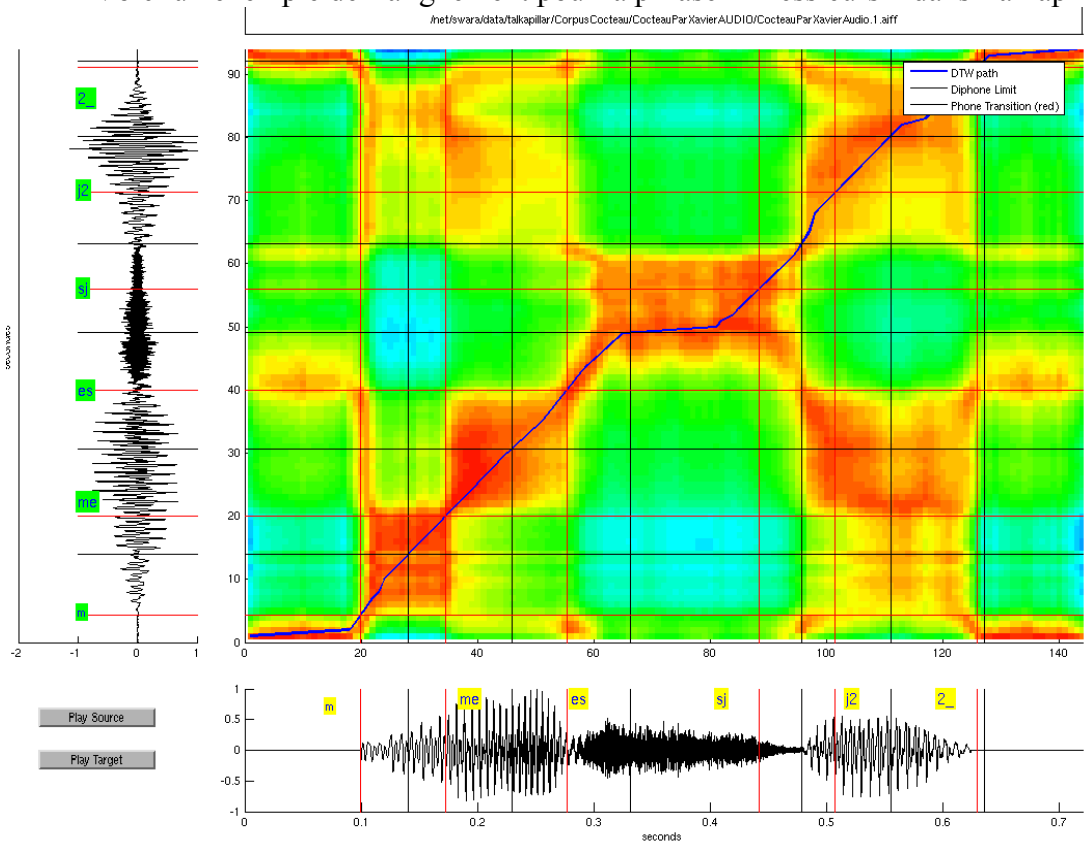
Or, il est impossible pour un locuteur de prononcer une phrase deux fois exactement de la même manière ce qui implique de grandes variations de longueur et il en est de même entre ces deux phrases à faire correspondre. La segmentation passe donc par une étape d'**alignement** temporel entre les deux phrases, problème courant en reconnaissance de la parole.

B. Alignement temporel de deux phrases

Afin de pouvoir comparer deux segments de parole, il est nécessaire d'adopter une représentation qui prend en compte les spécificités de ce type de signal. La plus couramment utilisée et celle qui donne les meilleurs résultats s'appelle paramétrisation MFCC pour *Mel-Frequency Cepstral Coefficients* (voir annexe 2)

Le problème est alors ramené à la comparaison, par alignement, de deux séquences vectorielles de longueurs différentes, représentant chacune un segment de parole. Dans Talkapillar, ceci est effectué par l'algorithme de *programmation dynamique* ou *Dynamic Time Warping* (DTW) , c'est-à-dire par déformation temporelle linéaire en « forçant » la correspondance des deux séquences par déformation temporelle (voir annexe 3).

Voici un exemple de l'alignement pour la phrase « Messieurs » dans Talkapillar :

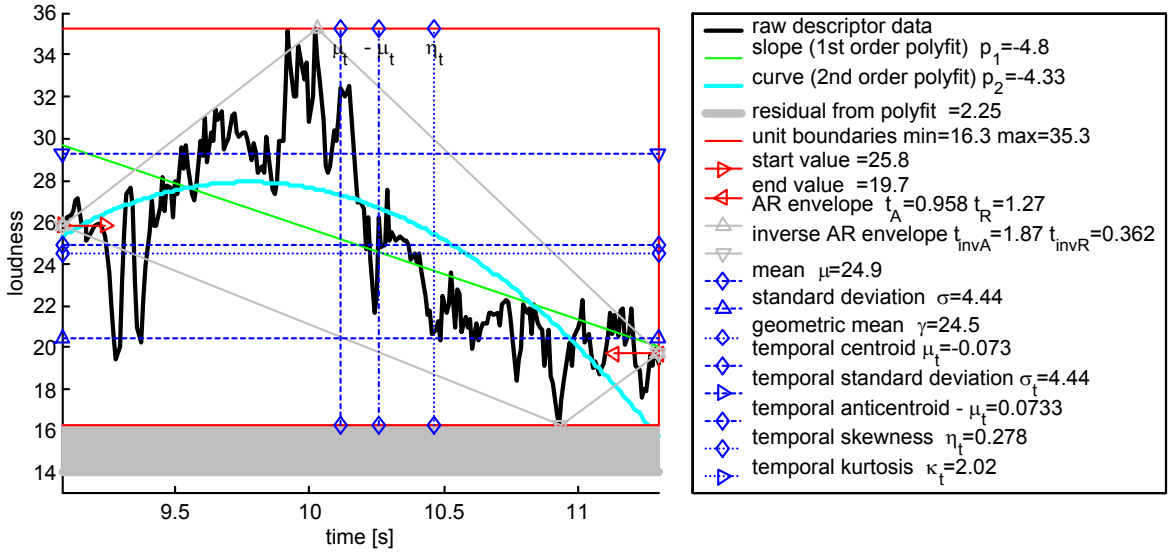


La phrase cible (prononcée par le locuteur) apparaît horizontalement, la source (référence obtenue par concaténation d'unités) verticalement. Les couleurs correspondent à une échelle de similarité cepstrale entre les deux signaux : pour les trames rouges les deux signaux sont similaires, puis plus on se rapproche du bleu plus les signaux sont dissemblables. Le chemin optimal affiché en bleu passe effectivement par les zones les plus rouges. Quand le chemin croise une délimitation entre deux unités sources (c'est-à-dire une ligne horizontale noire ou rouge) alors cela correspond à une frontière entre deux unité de la phrase cible. Le découpage qui s'ensuit délimite les diphones, entre deux marques noires, les phones entre marques rouges et les semiphones entre deux marques adjacentes quelconques.

3. Analyse & Importation

Une fois le corpus segmenté, chaque phrase audio est analysée puis les **descripteurs dynamiques** obtenus sont associés aux diphones correspondant. Pour chaque descripteur il existe plusieurs valeurs, appelées *characteristic values*, et calculées sur le diphone. Chacune des unités est alors importées dans la base de données avec l'intégralité de ces valeurs. Pour la sélection, seules les caractéristiques demandées seront chargées localement.

Voici un exemple d'analyse sur la courbe de f0 (hauteur) pour une unité de type diphone :



Toutes les unités sont indexées dans la base de données ainsi que tous les descripteurs qui lui ont été associés. La manipulation de la base de données, ainsi que l'élaboration de requêtes complexes, nécessite une interface complète de dialogue avec le serveur pgsql. C'est à cet effet que Diemo Scharz a implémenté le DBI. C'est une interface entre Matlab et la base de données de Talkapillar, pour manipuler les données de la base, hors processus d'importation ou de synthèse. Des actions telles que la création de corpus, la suppression d'un fichier et des unités qui lui font référence, l'obtention d'informations sur une unité etc..., sont grandement facilitées par cette interface.

4. Travail effectué

A l'écoute des synthèses fournies par Talkpapillar au début de mon stage, il s'est avéré qu'une des raisons du relativement médiocre résultat final (en plus de la nécessité d'optimiser l'algorithme de sélection, principale mission de mon stage), était la présence d'unités aberrantes dans la phrase synthétisée, c'est-à-dire dont la transcription textuelle et l'audio associé ne correspondaient pas. Cela est dû au fait que la sélection des unités, comme nous le verrons un peu plus loin, se fait exclusivement sur les critères symboliques associés à chaque diphone et jamais sur un critère d'intelligibilité ou de ressemblance acoustique. Ainsi, si une unité est mal décrite temporellement, elle pourra quand même être sélectionnée par l'algorithme qui n'a aucune indication sur cette possibilité d'erreur lors de l'étape d'alignement. La seule description connue à ce niveau est symbolique et basée uniquement sur l'analyse textuelle, puis synchronisée avec un alignement admis comme parfait. J'ai donc commencé mon stage en remédiant à cette lacune.

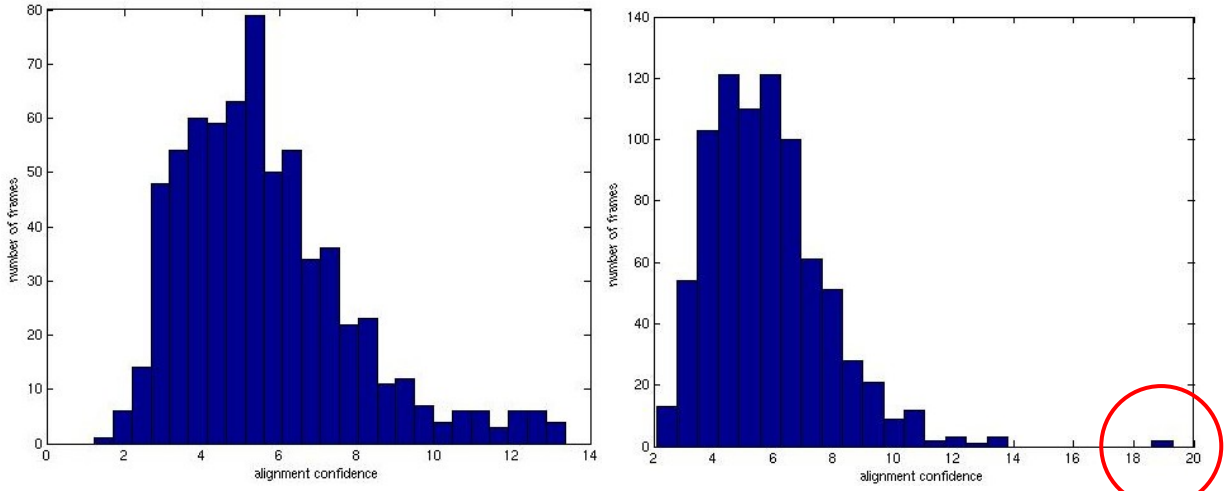
Il y a erreur de segmentation si le texte prononcé ne correspond pas exactement à la transcription phonétique d'Euler. Il s'ensuit qu'un diphone est différent dans la phrase prononcée est celle issue de la transcription (ou absent dans l'une ou l'autre). Les principales raisons que j'ai trouvées sont présentées en page suivante:

- Les liaisons ne sont pas effectuées de la même manière, cette règle dépendant beaucoup d'un locuteur à l'autre
- Erreurs dans la transcription phonétique. Euler travaillant sans dictionnaire, certaines règles sont mal utilisées. Certaines aberrations ont également été remarquées.
- La transcription d'Euler est celle d'un débit courant, rapide, ne convenant pas à un discours avec certaines pauses, de trop importants ralentissements. Ainsi certains « e » finaux qui peuvent avoir été prononcées n'apparaîtront jamais dans la transcription et seront assimilés aux unités adjacentes.
- Quelques erreurs dues au fait que Euler a été développé à partir du Français parlé en Belgique.

Afin d'éviter un réenregistrement du texte, j'ai introduit un indicateur pour chaque phrase décrivant l'efficacité de l'algorithme DTW à reconnaître la séquence de diphone présente dans la phrase source dans la phrase cible. Pour cela, j'ai associé à chaque abscisse temporelle dans la phrase cible la valeur de la distance euclidienne entre les vecteurs de MFCC après déformation temporelle par DTW. Une trop grande valeur indique une différence spectrale notable entre les deux diphones sensés être identique, et pourrait correspondre alors à une grande probabilité de mauvais alignement. Si plusieurs valeurs de cette distance correspondent à une seule abscisse temporelle de la cible, ce qui arrive quand l'algorithme s'écarte d'une diagonale pour aller chercher des unités selon la direction horizontale, alors seule la valeur maximale est retenue. En effet, puisque l'on travaille sur des probabilités d'erreur, une grande valeur sur une horizontale correspond à un chemin choisi impliquant une grande distance spectrale à une abscisse donnée.

Pour me rendre compte des informations que peut apporter un tel indicateur sur une phrase, j'ai tracé la distribution des trames temporelles de chaque phrase du corpus par rapport à ces distances. L'analyse des ces histogrammes a alors pu mettre en évidence l'existence de 2 types de distribution, selon l'existence ou non d'erreur d'alignement.

Voici deux exemples d'une phrase « bien portante » (à gauche) et d'une phrase comportant une erreur d'alignement due à une liaison non prononcée par le locuteur (à droite) :



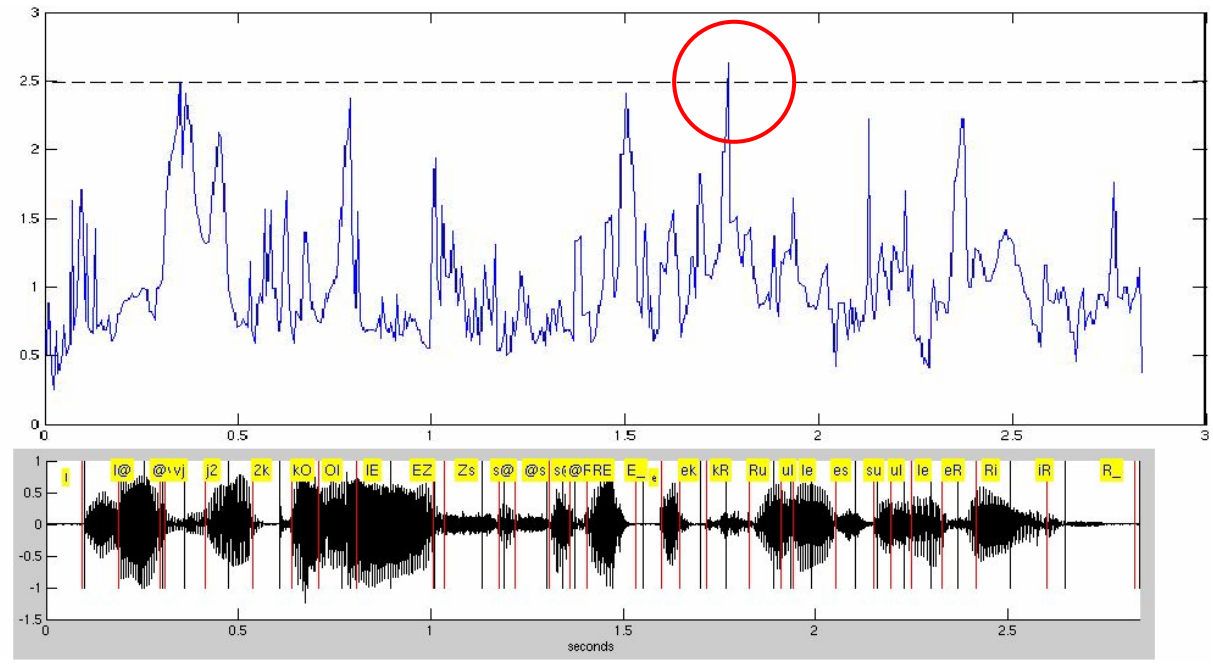
Cette représentation met en évidence le fait qu'un seuil bien choisi permettrait d'écarter les trames à risque puis les unités correspondantes. Une fois normalisé sur l'ensemble du corpus, cet indicateur permet donc de localiser les emplacements possibles d'erreurs.

Une première application que j'ai fait de cet indicateurs a été un script *Xverif.sh* permettant de corriger manuellement le texte afin de le faire « coller » à ce qui a été prononcé.

Pour chaque phrase contenant des grandes valeurs de ce score d'alignement, les fichiers audio (source et cible) s'ouvrent avec des marqueurs aux emplacements de distances au dessus d'un seuil déterminé empiriquement, permettant ainsi de vérifier la raison de l'erreur et d'apporter des modifications adéquates au texte original.

Le résultat de cette « adaptation » du texte à abouti à la création d'un nouveau corpus texte, utilisé comme référence pour la suite du projet.

Cette première étape n'ayant pas donné de résultats suffisants ce score a alors été intégré à la base de donnée comme descripteur de chacune des unités (cf. IV.1) . On peut alors associer à chaque fichier son du corpus, une courbe de confiance sur les unités :

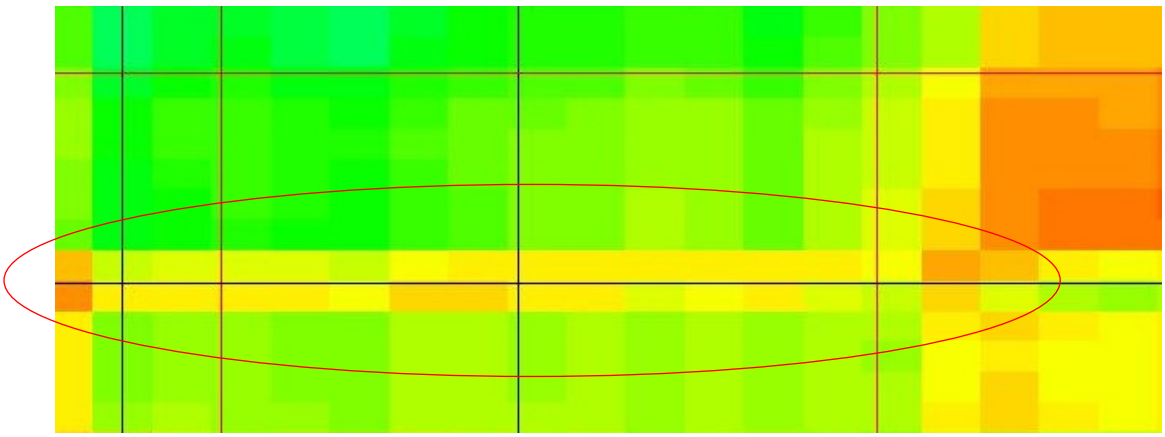


Avant l'étape de sélection, un tri sur l'ensemble des diphones est alors effectué pour rejeter les unités comportant une valeur maximale trop importante de ce descripteur (c'est-à-dire la valeur la plus importante pour chacune des trames de signal correspondant à cette phrase). Les valeurs écartées ne correspondent pas seulement à des erreurs grossières mais également à des prononciations trop éloignées de la prononciation de référence enregistrée dans le bootstrap.

Cet estimateur a permit de bien meilleurs résultats de synthèse. Néanmoins, cette méthode entraîne deux principaux inconvénients :

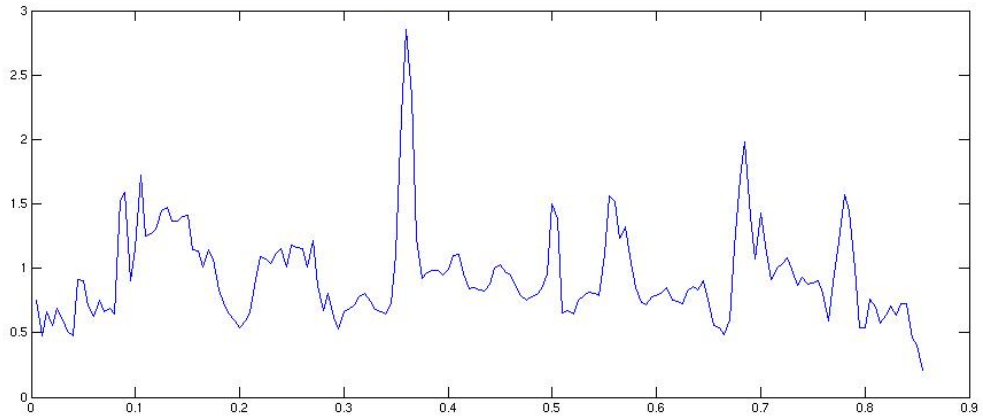
- Le premier est que certaines unités seront forcément retirées de la sélection pour un seuil donné, car elles ont un mauvais score d'alignement, sans que cela n'implique forcément une mauvaise synthèse si ces unités venaient à être choisies. Heureusement, cela reste assez minoritaire et un très grand nombre d'unités dans la base permet cette concession. En effet, sur l'ensemble du corpus de 1153 phrases, on dispose sans pré filtrage de 31733 diphones et après l'utilisation d'un seuil normalisé sur le corpus nous n'en utilisons par exemple que 28458, ce qui permet tout de même de disposer de plusieurs candidats pour chaque type de diphones.

- Le deuxième inconvénient était, au contraire, que certaines unités n'ont pas été filtrées alors que le résultat final d'une synthèse permettait de se rendre compte qu'elles avaient été mal découpées ou prononcées pendant l'alignement, introduisant un résultat avec des sonorités « fantômes ». Ceci est dû au fait que la valeur du score pour ces unités ne ressort pas de la courbe d'erreur globale. Après étude, j'ai déterminé que ce problème était généré par une trop faible granularité de la programmation dynamique. Si deux unités à aligner sont de toute évidence dissemblables à cause d'une erreur de prononciation, mais que sur la longueur d'une trame leurs spectres sont très semblables, alors l'algorithme DTW peut passer par le très étroit couloir dans la matrice de distances locales, et générer un score d'alignement local faible, comme le montre la figure suivante :

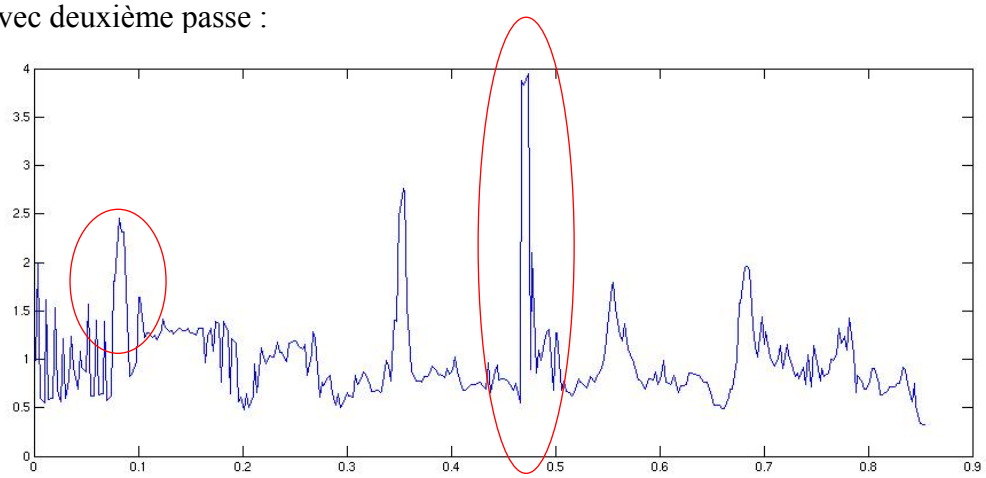


La solution retenue a été d' « étoffer » le descripteur d'alignement en effectuant une deuxième passe de DTW avec une granularité plus grossière afin de faire ressortir un comportement plus global à l'échelle d'un diphone. Pour chaque trame, la valeur maximale entre ces deux analyses est retenue. Les résultats obtenus pour une phrase avec une erreur de prononciation non détectée initialement sont présentés ci-dessous :

- Avec une seule passe :



- Avec deuxième passe :



On peut ainsi voir qu'une très vraisemblable erreur en raison d'une grande valeur a pu passer inaperçue au départ.

Cette méthode a donné de très bons résultats en ne conduisant à des phrases synthétiques qu'avec des unités bien segmentées. Je pouvais alors commencer le travail d'optimisation de la sélection.

IV. Optimisation de la sélection d'unités pour la synthèse

Il s'agit de la partie principale de mon stage, celle qui m'a à l'origine été assignée. Il s'agit d' « apprendre » automatiquement au programme la meilleure manière de choisir des unités en fonction des spécificités de la base de parole et de la description utilisée.

Après avoir introduit le procédé de synthèse d'une phrase, je présenterais l'état du problème, ma contribution, mes résultats, avant d'aborder les perspectives à venir.

1. La description des unités

Lorsque le système doit synthétiser une phrase donnée par une chaîne de caractère, celle-ci est d'abord découpée en unités, c'est-à-dire en une suite de diphones ordonnés: ce sont les **unités cibles** qu'il faut synthétiser au mieux à partir des unités présentes dans la base et résultant d'une analyse préalable, les **unités sources**.

A. Unités sources

Elles correspondent chacune à une entrée dans la base de données PostgreSQL et sont caractérisées par un grand nombre de descripteurs que l'on peut regrouper en deux catégories :

- Les descripteurs symboliques sont extraits de l'analyse textuelle de la phrase par le module Euler. Ils apportent des informations sur le diphone telles que sa position dans le mot dont il est issu, la place de ce mot dans la phrase, si c'est le premier mot de la phrase etc.… Ces données sont adaptées en fonction de l'alignement avant d'être associées aux unités correspondantes.

- Les descripteurs acoustiques sont extraits d'une analyse directe sur les fichiers audio et sont ensuite associés aux diphones correspondants en fonction de la segmentation. Le descripteur acoustique le plus significatif est celui de la fréquence fondamentale que nous appellerons « f0 ». Elle correspond à la mélodie de la phrase parlée (dans le cas de phonèmes voisés) et se situe pour une voix d'homme autour de 150 Hertz. La courbe de f0 ainsi que l'énergie ou un indicateur d'apériodicité (pour les plosives ou les fricatives par exemple) sont obtenus par l'algorithme YIN [Cheveigné, 2002] puis associée à chaque unité de la phrase (phone, semi-phone,diphone). A l'échelle d'un diphone, on dispose donc de la moyenne ou du maximum du tronçon de cette courbe, mais également d'indicateurs plus complets comme les polynômes d'interpolation de Legendre.

B. Unités cibles

Elles correspondent à une description précise des unités à sélectionner le mieux possible parmi celles du corpus. C'est pour cela que les choix des descripteurs symboliques et acoustiques sont primordiaux pour l'efficacité d'une synthèse.

Elles sont d'abord extraites de la phrase cible à synthétiser par le module Euler. On obtient ainsi une description symbolique de chacun des diphones en fonction d'informations grammaticales, sans information acoustique.

Pour ajouter une description prosodique à ces cibles, le système utilise une sélection de groupes prosodiques parmi ceux déjà présents et intégralement connus dans la base de parole :

- On ajoute à chaque fichier source les données prosodiques symboliques d'Euler auquel on associe les marqueurs temporels issus de la segmentation. On peut alors retrouver des descriptions acoustiques telles que l'évolution réelle de f0 durant l'unité comme décrit précédemment.
- A chaque groupe prosodique de la phrase cible on associe les descripteurs prosodiques issus de l'analyse du texte seul par Euler.
- On effectue une recherche de l'unité prosodique optimale dans la base, grâce à l'algorithme décris ci après
- On ajoute aux unités cibles les valeurs acoustiques correspondant aux groupes sélectionnés

Ainsi, avant même d'effectuer la synthèse on a pu ajouter aux unités cibles une description acoustique correspondant à l'intonation de la phrase cible si elle avait été prononcée par le locuteur.

Au final, elles doivent être décrites avec un nombre suffisant de descripteurs bien choisis, car ce sont autant de critère de sélection pour la synthèse.

C. Les données

Structure des unités dans Matlab

Une étape préalable à la sélection est le chargement des données pour l'ensemble des unités sources et cibles depuis la base pgSQL vers la machine locale sous la forme de la structure Matlab définie en annexe. Elle contient la description numérique de chaque unité en fonction des descripteurs spécifiés par l'utilisateur ainsi que la manière dont ils sont décrits :

- *name* : nom du descripteur
- *chval* (characteristic value) : moyen utilisé pour décrire ce descripteur sur la durée du diphone (moyenne, valeurs min ou max, analyse spectrale…)
- *weight* : poids de ce descripteur dans la minimisation de la fonction de coût (voir après)
- *func* : fonction utilisée pour évaluer la distance entre deux unités pour ce descripteur

Avant d'être traitées par l'algorithme de sélection, les unités sont donc rapatriées depuis la base sur l'ordinateur local sous la forme d'un tableau de structures :

`cid: 2281`	Index du corpus utilisé dans la base
`corpus: 'Xavier'`	Nom du corpus
`data: [28262x40 double]`	Toutes les données pour les descripteurs demandés
`uid: [28262x1 double]`	Index des unités rapatriées
`mean: [1x40 double]`	Moyenne de tous les descripteurs
`std: [1x40 double]`	Déviation standard des descripteurs
`ftid: [40x1 double]`	Index des descripteurs dans la base
`chval: {1x40 cell}`	Valeur caractéristique du descripteur
`N: 28262`	Nombres d'unités (diphones) dans la structure
`Nt: 18`	Nombre de critères de cible
`Nc: 22`	Nombre de critère de concaténation

Le champ data contient une matrice de taille [nb d'unités]×[nb de descripteurs] regroupant toutes les valeurs des descripteurs pour toutes les unités.

Fonctions distances

Nous avons vu qu'il existait une grande variété de données pour comparer deux unités en fonction des descripteurs et du type de valeur choisi. Il existe deux types de données :

Des données absolues

Elles décrivent une propriété fondamentale de l'unité concernée est sont extraites de la description statiques des unités par Euler. Il s'agit le plus souvent d'une position absolue (première ou dernière unité, du mot, de la phrase…) ou de l'appartenance à une unité linguistique de la base (mot lexical, mot grammatical …).

En critère de cible pour la sélection, pour comparer deux unités sur ce type de critère, il faut utiliser une distance binaire, qui ne peut avoir que deux valeurs : nulle si les valeurs sont égales et égale à 1 sinon.

En critère de concaténation, il s'agit de mesurer la continuité de ces données d'une unité à l'autre. Cependant, il a fallu créer de nouvelles fonctions spécifiques à certains descripteurs. Par exemple le critère de précédence entre deux unités a nécessité l'écriture d'une fonction comparant l'index de l'unité naturellement précédente d'une unité « droite » avec l'index d'une unité « gauche ». Si ces deux unités sont adjacentes dans la base, alors la distance sera nulle.

Des données algébriques

Elles peuvent être extraites de l'analyse des descripteurs dynamiques (f0, énergie, débit…) ou bien de certaines valeurs fournies par Euler (position absolue dans la phrase, dans le mot …). La distance euclidienne est utilisée le plus souvent pour comparer ces valeurs.

Etant donné qu'il existe un grand nombre de descripteurs différents avec des ordres de grandeur et des variations très différentes, il est nécessaire de normaliser les données sur l'ensemble du corpus. Cela permet d'éviter des distorsions entre les différentes distances traitées par l'algorithme de sélection.

Si x est la *characteristic value* d'un descripteur pour une unité donnée, alors la valeur normalisée x' est donnée par :

$$x' = \frac{x - \mu}{\sigma}$$

où μ est la moyenne et σ est la déviation standard de cette valeur caractéristique sur l'ensemble du corpus.

La distance euclidienne d sur un descripteur donné entre une unité source x_u et une unité cible x_t se ramène à :

$$d = \sqrt{\left(x_u^{'} - x_t^{'}\right)^2}$$

Par la suite, c'est le carré de cette distance qui sera utilisé pour la sélection où l'entraînement, afin d'éviter des problèmes liés à la racine carrée, en particulier pour les problèmes d'optimisation.

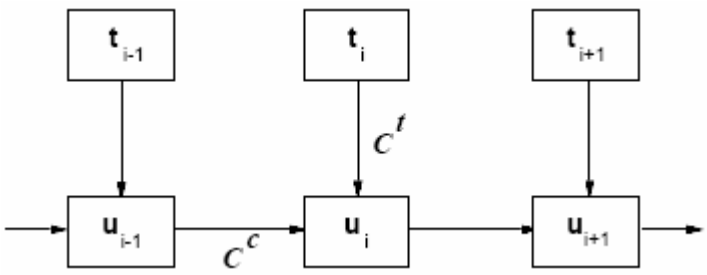
Comme pour les données absolues j'ai également dû créer une fonction pour comparer ces unités à la concaténation c'est-à-dire, au niveau d'un joint, la dernière valeur d'un descripteur de l'unité de gauche avec la première valeur de cette unité pour l'unité de droite. Par exemple pour la hauteur, il faut faire la différence entre la characteristic value *endval* du descripteur f_0 de l'unité u_{i-1}, et la characteristic value *startval* du descripteur f_0 de l'unité u_i.

2. La sélection des diphones

A. Coût cible et coût de concaténation

L'algorithme de sélection utilisé dans Talkapillar est basé sur la minimisation d'une **fonction de coût** global [Black, 1996]. Il s'agit de trouver le chemin optimal parmi les unités candidates permettant de trouver la séquence d'unités u_i correspondant le mieux à une séquence d'unités cibles t_τ par le biais de deux fonctions de coût, chacune faisant appel à un jeu de descripteurs propres (cf. 1.C) :

- Une fonction de coût cible mesurant la similarité entre deux unités u_i et t_τ
- Une fonction de coût de concaténation reposant sur la qualité de concaténation entre deux unités consécutives u_i et u_{i-1}.



Le chemin optimal est obtenu par l'algorithme de Viterbi dans le réseau des unités de la base. Il s'appuie sur une pondération des différents descripteurs, c'est-à-dire associer à chaque distance symbolique entre descripteur une importance relative dans la sélection.

Le coût de cible

Le coût de cible C^t mesure une similarité perceptuelle indépendante entre une unité source candidate u_i et une unité cible t_τ . Elle est donnée par la somme des différences entre les *p* descripteurs de ces unités, évaluées puis pondérées par une fonction de distance C_k^t et une valeur de poids propre ω_k^t :

$$C^t(u_i, t_\tau) = \sum_{k=1}^{p} \omega_k^t C_k^t(u_i, t_\tau)$$

Le coût de concaténation

Le coût de concaténation C^c correspond à une mesure de discontinuités entre toute paire d'unités (u_i, u_j) de la base. De la même manière, il est donné par la somme pondérée des q sous coûts C_k^c introduits par chaque descripteur de concaténation :

$$C^c(u_i, u_j) = \sum_{k=1}^{q} \omega_k^c C_k^c(u_i, u_j)$$

Le coût total

Dans le cas de la synthèse d'une phrase contenant *n* diphones, on cherche à minimiser la distance entre une séquence cible $t^n = (t_1, \ldots, t_n)$ et une séquence source $u^n = (u_1, \ldots, u_n)$. Le problème est ici ramené à la minimisation coût du coût global :

$$C(t^n, u^n) = Wt \sum_{i=1}^{n} \omega_i^t C_i^t(t_i, u_i) + Wc \sum_{j=2}^{n} \omega_j^c C_j^c(u_{j-1}, u_j)$$

ou *Wt* et *Wc* sont deux poids globaux permettant de régler l'importance relative d'un coût global sur l'autre.

La procédure de sélection consiste donc à trouver la séquence d'unités :

$$\overline{u}^n = \min_{u_1, \ldots, u_n} C(t^n, u^n)$$

B. L'algorithme de sélection

Le but de cette étape est de choisir la meilleure unité possible afin d'effectuer le moins possible de transformation sur la voix obtenue. Dans le cas de Talkapillar, aucune transformation audio n'a été envisagée après la synthèse.

Considérons la base de parole comme une interconnexion de diphones liés entre eux par des distances cibles et concaténative. On peut alors imaginer cette base comme un ensemble d'états, chacun correspondant à un diphone source, auxquels on associe :

- Une probabilité d'occupation b_{ij} (probabilité pour la $j^{ième}$ cible d'occuper le $i^{ème}$ état)
- Une probabilité de transition a_{ij} (passage de l'état i à l'état j)

Cette représentation suffit à représenter la base car tous les diphones sont potentiellement consécutifs, le réseau est donc intégralement connecté. L'algorithme de sélection doit trouver le chemin le moins coûteux dans ce réseau qui correspond à la cible. En utilisant le coût cible $\omega^t C^t$ comme probabilité d'occupation et le coût de concaténation $\omega^c C^c$ comme probabilité de transition, on peut utiliser l'**algorithme de Viterbi** [Viterbi, 1967]. Il s'agit, pour chacune des unités cibles, de ne garder séquentiellement que le plus court chemin trouvé précédemment. Pour chaque nouvelle unité cible t_τ , on étend le tableau du chemin ψ pour chaque candidat par une colonne τ en gardant uniquement le chemin minimal (indice de la ligne) :

pour $1 \le j \le N$

$a_{j1} = C^t(t_1, u_j)$

pour $2 \le \tau \le T$

pour $1 \le j \le N$

$b_{j\tau} = \omega^t C^t(t_\tau, u_j)$

$a_{j\tau} = \min_{1 \le k \le N} (a_{k,\tau-1} + \omega^c C^c(u_k, u_j) + b_{j\tau})$

$\psi_{j\tau} = k_{\min}$

Le décodage du chemin ψ pour trouver la séquence d'unités s_τ est trouvée en commençant pas la dernière unité *k* puis en remontant dans les indices :

$k = \underset{1 \le j \le N}{\operatorname{argmin}}(a_{jT})$

pour $T \ge \tau \ge 1$

$s_\tau = u_k$

$k = \psi_{\tau k}$

3. La sélection des groupes prosodiques

Comme nous l'avons vu précédemment, une étape de sélection d'unités prosodiques est effectuée afin d'ajouter une description acoustique aux diphones cibles. Le procédé de sélection est similaire à celui de la sélection de diphones, c'est-à-dire :

- Analyse par Euler du texte à synthétiser pour en extraire une division en groupes prosodiques
- Pour chacun de ces groupes, construction d'une unité prosodique cible comportant des informations symboliques sur la prosodie fournies par Euler:
  - → L'index de l'unité dans la phrase
  - → Les tons des accents finaux de début (qui appartient à l'unité précédente) et de fin d'unité
  - → Le nombre de syllabes neutres, inaccentuées dans l'unité etc.…
- Sélection de la meilleure unité prosodique parmi celles extraites des analyses du corpus : une analyse textuelle par Euler, une analyse acoustique par YIN et un alignement par DTW. La sélection se fait par le même algorithme de Viterbi que pour la sélection des unités, on a donc minimisation de 2 fonctions de coût :
  - → coût de distance à la cible: les tons des premier et dernier accents doivent correspondre, pondération ajustée selon le nombre de syllabes, fonction de la position de l'unité dans la phrase
  - → coût de concaténation : dans l'état actuel des choses, seule une continuité de hauteur à la jonction de deux groupes prosodiques permet d'estimer une bonne concaténation.
- Attribution des valeurs des descripteurs acoustiques issues de cette unité prosodique aux diphones cibles englobés dans le groupe prosodique cible.

Ce procédé utilise le même type de structure de sélection (cf. 1.C) et nécessite donc également un entraînement pour obtenir automatiquement la pondération des différents critères de sélection.

4. Travail effectué

A. Plan

Il existe plusieurs classes de méthode pour sélectionner des unités de parole dans une base de donnée :

- *Arbre de classification* : résultant d'une thèse de doctorat déposée par Robert Ed. Donovan à Cambridge en 1996, il s'agit d'organiser la base de données de façon à pouvoir choisir rapidement l'unité requise, à partir de ses critères linguistiques. Le plus souvent, il s'agit d'une classification en arbre, effectuée une fois pour toutes, lors de la conception du synthétiseur. La sélection d'unités ne se fait qu'entre classes dont les contextes sont adéquats par opposition à une sélection globale.
- *Fonction de coût* : Issue des travaux de Black, Hunt, et Campbell (1996) et utilisée principalement par AT&T (US) et ATR (Japon), cette méthode procède par minimisation dynamique d'une fonction de coût, estimée à partir de la phrase à produire (et de ses caractéristiques linguistiques) et des phrases enregistrées dans une base de données (ces phrases étant elles-mêmes analysées en fonction des mêmes critères linguistiques que la phrase à produire). Cette approche est la base de Caterpillar et Talkapillar.

Quand j'ai intégré le projet Talkapillar, l'algorithme de sélection des unités était déjà conçu pour minimiser une fonction de coût, j'ai donc ciblé mes recherches dans ce sens là.

J'ai commencé cette partie principale de mon stage en consultant dans la littérature scientifique les différentes méthodes utilisées pour l'entraînement d'un tel système de sélection. [Black, 1996] dégage deux classes de méthodes pour un tel système :

- **Recherche dans l'espace des poids** : il s'agit de tester un grand nombre de combinaison de poids, de mesurer une différence de qualité entre une phrase synthétisée par rapport à une référence, et réajuster les poids en fonction. Cette méthode telle quelle est beaucoup trop lourde à implémenter, mais on peut utiliser un algorithme d'optimisation pour mettre à jour automatiquement les poids en fonction des résultats, au lieu d'essayer un grand nombre de combinaisons.
- **Régression linéaire multiple** : elle consiste à faire « coller » les variations d'une distance objective entre deux unités à la fonction de coût. Cette méthode est fondamentalement différente de la précédente car elle implique un entraînement séparé des critères de cible et de concaténation.

B. Sélection des unités

Méthode de régression

Méthode 1 : Régression linaire multiple

La méthode de régression linéaire m'a paru plus naturelle et intuitive, j'ai donc commencé par l'implémenter pour me familiariser assez rapidement avec le système complexe de sélection. C'était aussi un moyen de fournir le plus rapidement possible un premier jeu de poids et de descripteurs aux autres membres de l'équipe.

En effet, depuis le début du projet et jusqu'à ce que je l'intègre, l'assignation des ces poids se faisait manuellement. Il fallait tout d'abord choisir les descripteurs qui semblaient judicieux pour la description des unités, tester un jeu de valeurs choisies instinctivement, écouter le résultat obtenu après synthèse d'une phrase quelconque, puis corriger ces valeurs manuellement etc.…Les résultats, bien que progressant à chaque fois en raison du facteur humain de l'ajustement des poids, étaient très passables, pour une grosse quantité de travail.

Le principe est le suivant : si l'on dispose d'une **distance objective** entre deux phrases, permettant d'en évaluer automatiquement la similarité perceptive et d'élocution,alors on peut en faire le lien avec une description symbolique judicieuse. Si l'on dispose d'un grand nombre d'association distance symbolique/distance objective, alors on peut envisager une régression linéaire sur chacun des descripteurs : c'est la régression linéaire multiple. Les cœfficients obtenus sont alors utilisés comme poids pour la sélection. En effet, ce problème de régression peut se ramener à l'obtention d'un vecteur ω tel que :

$$od(u_i, u_j) \propto \sum_{f=1}^{N_f} \omega(f) d_f(u_i, u_j)$$

- u_i et u_j sont 2 unités à comparer
- od est la fonction distance objective
- N_f est le nombre de descripteurs
- d_f est la distance symbolique pour le $i^{ème}$ descripteur,

Ainsi une différence d_i entre deux unités par rapport au $i^{ème}$ critère de cible ou de concaténation se traduira sur le coût total par une augmentation égale à $\omega_i d_i$.

C'est cette méthode de base que j'ai appliquée à la sélection des critères cibles, avant d'en imaginer une extension aux critères de concaténation.

Critères de cibles

Etant donné une phrase *s* comportant *T* diphones, on va réaliser l'entraînement sur chacun de ces diphones séparément. Pour chacun des ces diphones, on calcule la distance symbolique avec chacun des représentants de ce diphone dans la base. On obtient alors la matrice suivante :

$$X_\tau^t = \begin{pmatrix} d_1(u_1^\tau) & \dots & d_{N_t}(u_1^\tau) \\ \vdots & \ddots & \vdots \\ d_1(u_{n_\tau}^\tau) & \cdots & d_{N_t}(u_{n_\tau}^\tau) \end{pmatrix}, \tau \in [1, T]$$

- N_t est le nombre de critères de cibles
- $d_k(.)$ est la fonction de distance pour le k^{ième} descripteur, avec l'unité cible
- n_t est le nombre de diphones candidats (même symbole que la cible)
- u_j^τ est le j^{ième} candidat.

Pour la fonction de distance objective, j'ai utilisé la distance euclidienne sur les cœfficients cepstraux. Je commence par aligner temporellement, en utilisant l'algorithme DTW (voir annexe 3), les deux signaux correspondants aux diphones cible et candidats et j'utilise comme score pour cette association la somme des distance locales sur chacune des trames d'avancement du DTW. En effet cette fonction sera nulle pour la comparaison de deux fois le même signal (même nombre de trames, mêmes valeurs de cepstre sur ces trames), et une forte différence de spectre correspondant à une forte différence d'élocution impliquera une augmentation de cette valeur. On aboutit alors à un vecteur comprenant *n* valeurs de distance objective avec l'unité cible.

Par analogie, on dispose alors d'une bibliothèque de *n* « expériences », chacune caractérisée par N_t « paramètres » et ayant engendré *n* « résultats ». On effectue alors une régression linéaire sur tous les paramètres et les cœfficients obtenus sont utilisés comme poids pour la sélection.

Cette expérience est répétée pour chacun des diphones présent dans la phrase d'entraînement, et on obtient ainsi *T* séries de N_t valeurs de coefficients.

Critères de concaténation

A propos des critères de concaténation, la littérature ne traitait ce problème qu'en considérant qu'il ne se ramenait qu'à des sauts de f0 et d'énergie et se traitait à part, avec un réglage manuel. J'ai alors étendu la méthode décrite ci-dessus à l'entraînement des critères de concaténation.

Ce qui nous intéresse ici est l'influence des différences de critères de concaténation entre deux unités sur la qualité du joint sonore. On est alors ramenés au problème :

$$od(u_{i-1}, u_i) \propto \sum_{f=1}^{N_f} \omega_i^c(f) d_f(u_{i-1}, u_i)$$

Pour la phrase *s* comportant *T* diphones, on va réaliser *T* - *1*entrainements pour chacun des joints sonores. On calcule ensuite la distance symbolique entre toutes les combinaisons des deux diphones concernés dans la base la base.

On obtient alors la matrice suivante :

$$X_\tau^c = \begin{pmatrix} d_1(u_1^\tau, u_1^{\tau+1}) & \cdots & d_{N_c}(u_1^\tau, u_1^{\tau+1}) \\ \vdots & \ddots & \vdots \\ d_1(u_1^\tau, u_{n_{\tau+1}}^{\tau+1}) & \cdots & d_{N_c}(u_1^\tau, u_{n_{\tau+1}}^{\tau+1}) \\ \vdots & \vdots & \vdots \\ d_1(u_j^\tau, u_k^{\tau+1}) & \cdots & d_{N_c}(u_j^\tau, u_k^{\tau+1}) \\ \vdots & \vdots & \vdots \\ d_1(u_{n_\tau}^\tau, u_{n_{\tau+1}}^{\tau+1}) & \cdots & d_{N_c}(u_{n_\tau}^\tau, u_{n_{\tau+1}}^{\tau+1}) \end{pmatrix}, \tau \in [1, T]$$

- N_c est le nombre de critères de concaténations
- $d_k(.,.)$ est la fonction de concaténation pour le k^{ième} descripteur entre deux unités source

Pour la distance objective, on utilise la même que pour les critères de cibles, mais en comparant l'audio créé en concaténant directement les deux diphones candidats avec l'audio cible, découpé exactement aux limites des ces deux diphones.

La régression multiple permet de la manière d'obtenir un jeu de poids pour la concaténation. En revanche il est nécessaire d'équilibrer ces critères par rapport aux critères de cible. En effet, l'entraînement ayant eu lieu séparément pour ces deux types de poids, il faut régler la valeur de $\frac{W_t}{W_c}$ (cf. IV.3.C) :

- Soit en normalisant chaque jeu de poids (division par la valeur de poids maximale)
- Soit en synthétisant une même phrase avec différentes valeurs de ce coefficient multiplicatif et en ne gardant que la meilleure combinaison.

On obtient alors *T - 1* jeux de N_c coefficients pour la concaténation.

Limites

J'ai moyenné l'ensemble de tous les jeux de sur la phrase pour n'obtenir plus que N_t + N_c valeurs à exploiter. Une autre solution aurait été de garder un grand jeu de poids pour l'ensemble du corpus, et associer un jeu de poids en fonction du contexte lexical du diphone : un jeu de poids pour chaque diphone du français, ou pour chaque catégorie de phones (plosives, liquides, voyelles …). Mais il nous a semblé plus judicieux de caractériser un corpus par un seul jeu de poids même si l'on doit fortement augmenter le temps de calcul en entraînant également sur un grand nombre de phrases.

Le problème de cette méthode a été que certains des coefficients obtenus étaient forcément négatifs. Pour obtenir des poids j'ai alors dû retraiter ces données pour obtenir uniquement des valeurs positives. En effet, une valeur négative pour un descripteur signifierait qu'une dissemblance entre deux diphones par rapports à celui-ci entraînerait une diminution de la distance globale entre ces deux diphones. Il s'agit donc d'une limite du modèle utilisé. Néanmoins, après avoir constaté que les quelques cœfficients négatifs était de moindre valeur que les positifs, j'ai extrait des poids de ces valeurs en assignant un poids nul aux coefficients de valeur faible (donc incluant les valeurs négatives) et leur valeur aux

cœfficients plus élevés. Bien évidemment cette méthode détruit la balance entre descripteurs, mais a quand même permis des progrès notables car les descripteurs les plus importants pour la synthèse ressortait des valeurs. Finalement, il s'est avéré que le modèle ne correspondait pas et je suis donc passé à une autre formulation du problème.

Méthode 2 : Régression non linéaire

Par la suite, tout en gardant la formulation régressive du problème telle qu'elle est définie ci dessus, j'ai utilisé une formulation classique de problème d'optimisation avec contrainte sur les valeurs des poids. L'obtention des vecteurs de poids ω^t et ω^c peut alors être calculée par la résolution du problème suivant :

- Pour les critères de cibles :

$$\begin{cases} \min\limits_{\omega^t \in \mathbb{R}^{N_t}} \left\{ f^t(\omega^t) \right\} \\ \text{tel que } \forall k \in [1; N_t],\ \omega^t(k) \geq 0 \\ \text{avec } f^t(x) = \sum\limits_{i=1}^{n_\tau} \left(\left[\sum\limits_{k=1}^{N_t} \omega_k^t \left(d_k^t(u_i^\tau) \right)^2 \right] - \text{od}^t(u_i^\tau) \right)^2 \end{cases}$$

- Pour les critères de concaténation :

$$\begin{cases} \min\limits_{\omega^c \in \mathbb{R}^{N_c}} \left\{ f^c(\omega^c) \right\} \\ \text{tel que } \forall k \in [1; N^c],\ \omega^c(k) \geq 0 \\ \text{avec } f^c(\omega^c) = \sum\limits_{j=1}^{n_{\tau+1}} \sum\limits_{i=1}^{n_\tau} \left(\left[\sum\limits_{k=1}^{Nc} \omega_k^c \left(d_k^c(u_i^\tau, u_j^{\tau+1}) \right)^2 \right] - \text{od}^c(u_i^\tau, u_j^{\tau+1}) \right)^2 \end{cases}$$

Optimisation globale

Face à la faible qualité des synthèses obtenues par cette méthode - du moins pour la sélection des unités cibles, nous verrons qu'elle marchera pour la prosodie - j'ai également implémenté un algorithme d'entraînement fonctionnant sur l'optimisation globale de tous les poids intervenant dans la sélection, c'est-à-dire pour les critères de cibles et les critères de concaténation, comme s'il s'agissait d'un seul jeu de poids comportant $N_t + N_c$ valeurs.

J'ai également pris en compte la nécessité de choisir automatiquement les descripteurs cohérents pour la synthèse. En effet, on dispose en tout de 83 descripteurs parmi lesquels on choisi ceux que l'on estime être les plus représentatifs d'une unité ou d'un joint. Un algorithme d'optimisation globale devrait être capable d'évaluer la contribution de chaque descripteur dans la qualité finale et d'ajuster la valeur du poids associé en fonction.

Pour cela j'ai conservé comme mesure objective de la qualité d'une synthèse sa ressemblance avec la phrase naturellement prononcée par le locuteur. Il s'agit donc de résoudre le problème suivant :

$$\begin{cases} \min\limits_{\omega \in \mathbb{R}^{N_t+N_c}} \left\{ od(s) \right\} \\ \text{avec } \begin{cases} \forall k \in [1; N_t + N_c], \omega(k) \geq 0 \\ \omega_{\text{init}} = 0_{\mathbb{R}^{N_t+N_c}} \end{cases} \end{cases}$$

C'est l'algorithme à direction de descente en gradient *line-search* qui permet la convergence la plus rapide vers une solution à ce problème. Le programme d'entraînement va donc :

- Rechercher dans le corpus les données syntaxique et acoustiques de la phrase
- Définir les unités cibles correspondantes
- Charger tous les diphones de la base en excluant ceux de la phrase concernée
- TANT QUE la phrase synthétisée n'est pas assez bonne
  - synthétiser la phrase cible
  - calculer la distance objective avec la phrase naturelle
  - adapter les poids

  FIN

Si nous prenons comme exemple la structure définie en annexe 5, voici un exemple de valeurs obtenues pour une phrase donnée et des groupes prosodiques sélectionnés préalablement, pour 18 critères de cible et 22 critères de concaténation :

```
W =
  Columns 1 through 6
     145.32       3.8993       3.1102        5.895       3.6712            0
  Columns 7 through 12
    0.48684        4.729       0.2097       1.2197      0.34718            0
  Columns 13 through 18
          0       2.7661       2.2943       4.8063      0.99556       1.4259
  Columns 19 through 24
          0            0            0            0            0            0
  Columns 25 through 30
          0            0            0            0            0            0
  Columns 31 through 36
          0       2.3055            0            0            0            0
  Columns 37 through 40
          0            0            0            0
```

Sur ces résultats obtenus après approximativement 15 minutes de calcul, on peut observer que le critère prépondérant pour la sélection d'une unité est le symbole du diphone correspondant, et que pour les critères de concaténation, seul le saut de hauteur dû au joint a été retenu comme critère utile pour une bonne sélection, ce qui parait cohérent.

Les synthèses réalisées alors sont celles qui ont donnés les meilleurs résultats, d'après mon jugement personnel et celui de l'ensemble des personnes de l'équipe. Cependant les phrases synthétiques comportaient des intonation non naturelles où incohérentes à l'échelle de le phrase entière. Cela était causé par une mauvaise description acoustique des cibles, et donc par une mauvaise sélection des groupes prosodiques.

Pour m'en assurer, j'ai réalisé des phrases synthétiques du corpus (comme décrit préalablement) mais en incorporant aux cibles correspondantes, en plus de l'analyse syntaxique la description acoustique des phrases naturelles. Cela revient au cas où :

- les groupes prosodiques auraient sélectionnés les unités parfaites (car exactement ceux correspondant au texte saisi).
- les descripteurs acoustiques correspondant à ces groupes prosodiques idéaux ont correctement été ajoutés à la description des unités cibles.

Puis j'ai synthétisé ces phrases à partir des unités du corpus, auquel on a retiré cette phrase et les unités mal alignées. Les résultats obtenus ont alors été les meilleurs depuis le début, évidemment avec une prosodie naturelle, mais des unités très bien choisies, la qualité globale étant essentiellement limitée par la faible qualité du corpus (voir 5.).

J'ai également synthétisé des phrases du corpus mais uniquement à partir du texte, sans aucune description manuelle de la cible. L'algorithme va alors par lui-même chercher les unités correspondant à cette phrase parmi celles du corpus et reconstituer le phrase. Les seules unités naturelles non présentes dans la phrase finale ont été filtrées à cause de leur mauvais score d'alignement (cf. III).

Dans toute la suite de mon travail j'ai utilisé les poids cibles précédents comme poids optimaux pour la sélection des diphones.

C. Sélection prosodique

Les groupes prosodiques, de la même manière que les diphones, phones ou semi-phones, sont importés dans la base de donnée sous forme d'unités avec une description symbolique et acoustique. La structure d'un groupe prosodique dans Matlab est donc la même que pour un diphone. On peut donc envisager d'utiliser les mêmes algorithmes.

Tout d'abord, il a donc fallu définir une « distance objective prosodique » qui mesurerait la similarité entre unités prosodiques. J'ai utilisé pour cela la différence de courbe de hauteur car c'est celle qui caractérise le plus la « musicalité » de la voix. Les groupes prosodiques ont tous des longueurs différentes, il faut donc d'abord aligner les courbes cibles et sources par l'algorithme DTW puis effectuer la distance euclidienne.

En outre, même si l'on peut imaginer des correspondances de groupes prosodiques un à un, on ne dispose pas de critères pour la bonne concaténation de plusieurs groupes prosodiques. En effet, un saut minime de f0 ne garantie pas que deux unités prosodiques concaténées sont objectivement cohérentes entre elles, par exemple si le début d'une phrase est neutre et la fin interrogative avec une virgule pour séparer les deux.

J'ai donc utilisé la méthode de régression non linéaire qui réduit le problème à l'échelle d'un ou deux groupes prosodiques. Il s'agit pratiquement du même algorithme, mais n'ayant pas de symbole à associer à un groupe prosodique, j'ai effectué un tri sur la longueur des groupes prosodiques avant de les évaluer : les unités ayant une longueur trop différente n'ont aucune chance d'être sélectionnée. Pour la concaténation, la distance objective utilisée est la différence alignée entre la courbe obtenue par concaténation des courbes de hauteur des 2 unités candidates, et la section de la courbe de hauteur de la phrase naturelle correspondant aux deux groupes prosodiques adjacents.

Les résultats obtenus sont assez concluants sans être optimaux, la qualité de la synthèse s'en est trouvée beaucoup améliorée.

Afin d'améliorer encore les résultats de convergence, j'ai utilisé une mise à jour BFGS en fournissant le gradient de la fonction à minimiser :

- pour les critères de cible :

$$\begin{cases} g^t(\omega^t) = \left[\frac{\partial f^t(\omega^t)}{\partial \omega_1^t} \cdots \quad \frac{\partial f^t(\omega^t)}{\partial \omega_f^t} \cdots \quad \frac{\partial f^t(\omega^t)}{\partial \omega_{N_t}^t} \right] \\ \text{avec } \forall f \in [1; N_t], \ \frac{\partial f^t(\omega^t)}{\partial \omega_f^t} = \sum_{i=1}^{n_\tau} 2 \cdot \left(\left[\sum_{k=1}^{N_t} \omega_k^t \left(d_k^t(u_i^\tau) \right)^2 \right] - \mathrm{od}^t(u_i^\tau) \right) \cdot \left(d_f^t(u_i^\tau) \right)^2 \end{cases}$$

- pour les critères de concaténation :

$$\begin{cases} g^c(\omega^c) = \left[\frac{\partial f^c(\omega^c)}{\partial \omega_1^c} \cdots \quad \frac{\partial f^c(\omega^c)}{\partial \omega_f^c} \cdots \quad \frac{\partial f^c(\omega^c)}{\partial \omega_{Nc}^c} \right] \\ \text{avec } \forall f \in [1; N_c], \ \frac{\partial f^c(\omega^c)}{\partial \omega_f^c} = \sum_{j=1}^{n_{\tau+1}} \sum_{i=1}^{n_\tau} 2 \cdot \left(\left[\sum_{k=1}^{Nc} \omega_k^c \left(d_k^c(u_i^\tau, u_j^{\tau+1}) \right)^2 \right] - \mathrm{od}^c(u_i^\tau, u_j^{\tau+1}) \right) \cdot \left(d_f^c(u_i^\tau, u_j^{\tau+1}) \right)^2 \end{cases}$$

L'algorithme de Quasi-Newton permet alors une convergence même à partir d'un point initial très éloigné de la solution optimale, et les résultats obtenus sont encore meilleurs.

5. Perspectives

Les phrases que l'on peut désormais synthétiser avec le système sont de bien meilleure qualité pour la plupart mais encore loin de résultats acceptables dans un but commercial. Dans cette partie nous allons développer les aspects à améliorer dans Talkapillar :

A. La base de données

L'un des premiers défauts de notre corpus est dû à son contenu textuel. Le but premier de Talkapillar étant de resynthétiser la voix de Jean Cocteau, ce texte a été utilisé par défaut alors qu'il n'apparaît en fait pas adapté au développement de Talkapillar.

- Il n'est pas équilibré phonétiquement, donc tous les diphones de la langue française ne sont pas représentés équitablement voire pas du tout pour certains. Or la substitution d'un diphone absent par un autre lors de la sélection, selon des critères qui n'ont rien à voir avec le symbole de ce diphone, décrédibilise complètement l'intelligibilité et le naturel de la phrase. D'autres corpus ont déjà été importé mais uniquement pour compléter la pauvreté de ce texte et ne suffisent pas en lui-même.
- Il possède de nombreux noms propres et étrangers qui engendrent un grand nombre d'erreurs avec le module d'analyse textuelle et les différences de prononciation
- Beaucoup de phrases sont beaucoup trop longues et forcent le locuteur à adapter son rythme et sa respiration en ajoutant par exemple des pauses là où Euler n'en met pas.

De plus il reste de gros progrès à faire sur la segmentation automatique de ce texte car dans bon nombre de synthèses, pour certains joints, on sent l'influence de la coarticulation d'un phone de l'unité naturellement précédente sur celle qui a été sélectionnée. Ce phénomène provoque l'impression d'entendre une « deuxième voix » qui se superpose à la synthèse même si les unités sont correctement sélectionnées. Cela est dû à une mauvaise délimitation des phonèmes par l'alignement, ce qui rejoint le problème récurent en synthèse concaténative : concrètement, comment peut on caractériser un diphone ?

En outre, on peut aussi noter que notre base de parole est réduite par rapport à d'autres travaux dans ce domaine (France Télécom travaille avec plus de 10 heures de parole pour de très bons résultats, nous travaillions avec 1h30).

B. La sélection

Comme nous l'avons vu précédemment, la méthode de régression a permit de mettre en évidence des critères de préférences pour les descripteurs très différents d'une unité cible à l'autre. Les similarités sont de deux types :

- En rapport avec des catégories syntaxiques. Par exemple, avec les premières et dernières unités d'une phrase, la solution optimale convergera vers un jeu de poids

mettant en avant les critères absolus de position dans la phrase (lastinsentence, *firstinword* …), alors que les unités en milieu de mot par exemple vont faire émerger le descripteur associé à la position relative (*posinword, posinsent, ..*)

- En rapport avec des catégories linguistiques. Etant donné les grandes différences qu'il peut y avoir entre deux types de phones (par exemple voisées/plosives), j'ai pu observer des regroupements en fonction de ces classes.

L'un des prochains objectifs que me fixe pour la fin de ce stage est donc de mettre en place une procédure de sélection qui charge pour chaque unité cible un jeu de poids en fonction de son appartenance. Dans l'idéal, il s'agirait d'entraîner l'algorithme sur l'ensemble du corpus pour obtenir un jeu de poids pour chaque diphone, mais le temps de calcul est encore trop important. En revanche on peut concevoir des regroupements en classes de diphones.

C. La distance objective

Pour la comparaison de deux phrases on utilise la distance euclidienne des MFCC sur les phrases alignées, mais cette fonction de rend pas assez compte des pertes de qualités de la synthèse dues à une concaténation catastrophique. Elle est en revanche très efficace pour le mesure de similarité sur les parties stables des diphones. Or c'est la perception des sauts qui est la principale limite de notre système

Une possible extension serait donc de définir une fonction objective qui rende mieux compte des problèmes de concaténation. On pourrait ainsi utiliser pour les critères de concaténation la courbe de f0 comme pour les groupes prosodiques, et ne mesurer que la hauteur des sauts à la concaténation. Il est à noter que le saut de f0 est également un des critères de concaténation est que l'utilisation de cette fonction objective de concaténation convergerait toujours vers un poids unique sur ce descripteur

V. Bilan

Tout d'abord ce stage s'est parfaitement inscrit dans le prolongement de mon Master Recherche puisqu'il m'a fait mettre en pratique les cours qui ont été abordés pendant 5 mois et en particulier les modules d'Acoustique, de Traitement & Synthèse de la Parole et d'Optimisation. Ces enseignements préalables m'ont été très utiles pour aborder le stage avec une bonne connaissance préalable de ces domaines, et j'ai ainsi pu me plonger directement dans mon travail sans une première étape de mise à niveau. Ce fut particulièrement utile car j'ai pu dès le départ me familiariser avec ce projet complexe et entamer mes recherches rapidement.

Concernant le contexte de ce stage, j'en suis particulièrement satisfait car ce travail était à la fois un projet de recherche à la pointe de ce qui se fait en synthèse de la parole mais également un important travail de développement, car dans l'idéal ce projet devrait devenir un outil pour les compositeurs. Ce fut particulièrement enrichissant d'aborder mon stage sous ces deux aspects conjoints.

A propos des résultats, j'ai eu la chance de pouvoir percevoir les évolutions de la qualité du travail du début à la fin de mon stage, et me rendre compte que ma contribution à ce projet a apporté de bonnes avancées au projet. La qualité des phrases synthétisées à la fin de mon stage est très encourageante et de loin meilleures à celles issues du système à mon arrivée. De plus, l'utilisation future d'un nouveau corpus, mieux choisi et plus long, permettra, j'en suis sûr, d'offrir des résultats parfaitement exploitables dans un contexte artistique voire commercial.

D'un point de vue personnel, ce stage m'a permit de concrétiser le vif intérêt pour le traitement et la synthèse de la parole, à la suite des cours du Master Recherche SIAO. J'ai trouvé également particulièrement enrichissant le fait de travailler à l'Ircam, dans un contexte artistique et scientifique unique au monde. Ces 6 mois au sein de l'équipe Analyse/Synthèse m'ont apporté une grande ouverture d'esprit sur la synthèse sonore et l'Informatique musicale en général.

En conclusion je dirais donc que ce stage a été une réussite, qu'il m'a donné l'opportunité de développer de très bonnes facultés d'autonomie en recherche, d'adaptation et de développement sur un projet. Il m'a également permit de consolider mes connaissances en développement sur Matlab, ma maîtrise du système Linux, ainsi que diverses autres compétences comme la gestion de projet d'équipe par CVS.

VI. Annexes

1. Références

[Bagein, 2001]] Michel Bagein, Thierry Dutoit, Nawfal Tounsi, Fabrice Malfrère, Alain Ruelle, and Dominique Wynsberghe. *Le projet EULER, Vers une synthèse de parole générique et multilingue*. Traitement automatique des langues, 42(1), 2001.

[Beller, 2005] Grégory Beller. *La musicalité de la voix parlée*. Maîtrise de musique, Université Paris 8, Paris, 2005.

[Black, 1996] Hunt, A. et Black, A. W. (1996). *Unit selection in a concatenative speech synthesis system using a large speech database*. in Proceedings of the 21th International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 373-376, Atlanta, GA, USA.

[Blouin, 2003] Christophe Blouin. *Sélection des unités pour la synthèse vocale par concaténation*.PhD thesis, Université Paris 11 Orsay, 2003.

[Cheveigné, 2002] Alain de Cheveigné and Hideki Kawahara. *YIN, a Fundamental Frequency Estimator for Speech and Music*. Journal of the Acoustical Society of America (JASA), 111:1917–1930, 2002.

[Flanagan, 1972] Flanagan, J. L., ed. (1972). *The Synthesis of Speech,* Scientific American.

[Schwarz, 2004] Schwarz D. (2004). *Data-driven concatenative sound synthesis*. PhD thesis, Université Paris 6.

[Viterbi, 1967] Viterbi, A. J. (1967, April). *Error bounds for convolutional codes and an asymptotically optimal decoding algorithm*. IEEE Transactions on Information Theory IT-13, 260-269.

2. Paramétrisation MFCC

A. Le cepstre

Le signal de la parole est porteur de différentes informations, dont le fondamental, qui ne sont pas toutes nécessaires à la compréhension du message. Ce signal est le résultat d'une multiplication du spectre d'entrée par la réponse fréquentielle du conduit vocal. Dans ce cas, tout le contenu spectral du signal source, modifié par le conduit vocal, ne peut pas être éliminé par filtrage, puisque précisément cette opération de filtrage est adaptée à l'extraction d'information localisée en fréquence.

Une alternative consiste à utiliser la méthode « cepstrale ». En effet, celle-ci est basée sur l'opérateur logarithme qui transforme un produit en addition, ce qui permet alors de séparer la source du conduit vocal. Le cepstre est défini comme la Transformée de Fourier inverse du module d'un spectre exprimé en échelle logarithmique :

$$y_c(k) = TF^{-1}\left[Log_{10} \left| Y(f) \right|^2 \right]$$

- *c* indique le domaine cepstral, appelé aussi domaine « pseudo-temporel »
- *k* la variable cepstrale exprimée en « quéfrence » et homogène à un temps.

Après la séparation des modèles du conduit vocal et de la source (Fig.1a), on procède à une opération de « liftrage » (Fig.1b) pour éliminer toutes les quéfrences dépassant un certain seuil (Fig. 2a, 2b, 2c). Le retour au domaine fréquentiel s'effectue par une FFT dont on calcule ensuite l'exponentielle (Fig.1c). On obtient alors un spectre lissé constitué de l'information formantique (Fig.2).

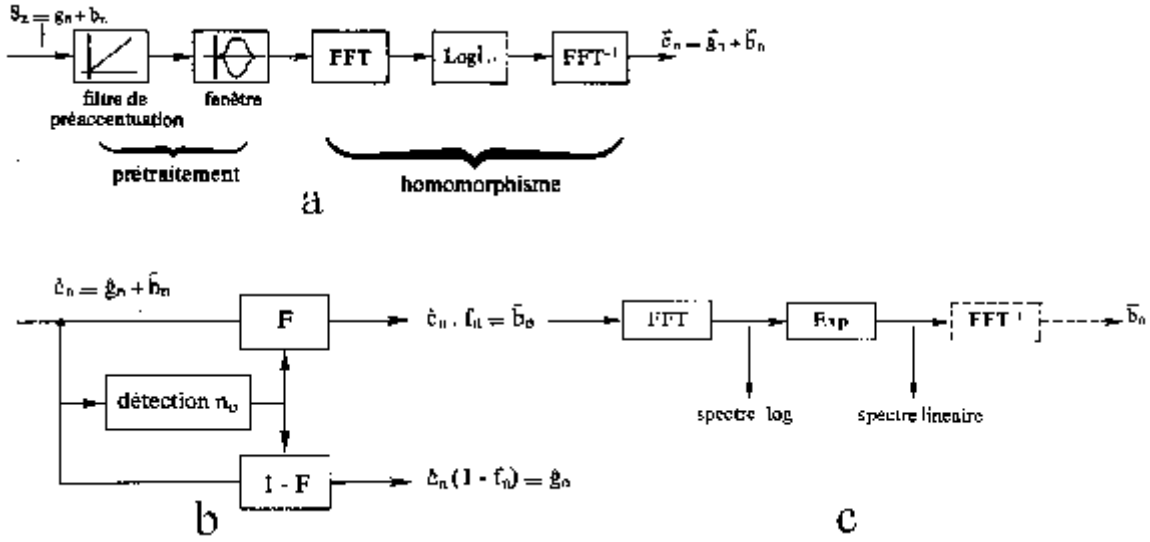


Figure 1

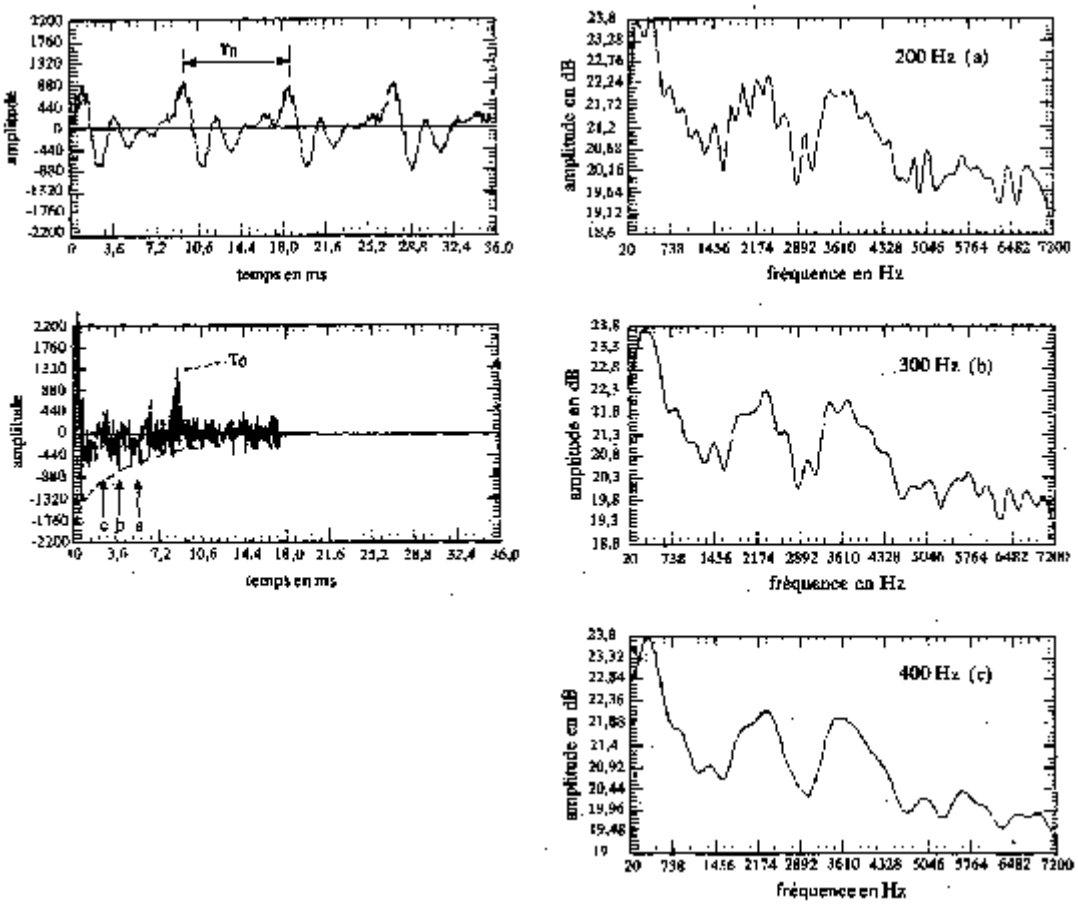


Figure 2

B. L'analyse MFCC

Après 500Hz, l'oreille perçoit moins d'une octave pour un doublement de la fréquence. Des expériences psycho acoustiques ont alors permis d'établir la loi qui relie la fréquence et la hauteur perçue : l'échelle des Mels où le « Mel » est une unité représentative de la hauteur perçue d'un son (Fig.6). Par exemple l'octave d'un son de 2000Hz (1800 Mels) sonnera l'octave supérieure à 4600Hz (3600 Mels) au lieu de 4000Hz.

L'analyse MFCC (*Mel Frequency Cepstral Coefficients*) consiste en l'évaluation de Coefficients « Cepstraux » à partir d'une répartition Fréquentielle selon cette échelle. Mais il faut noter, dans l'analyse MFCC décrite ci-dessous, que les coefficients cepstraux obtenus ne correspondent pas exactement à la définition du cepstre défini précédemment.

Cette technique est composée dans un premier temps d'une analyse spectrale. L'étendue dynamique du spectre de puissance ainsi obtenu permet sa compression logarithmique afin de s'accorder avec la perception d'intensité de l'oreille humaine.

Pour finir, une transformée discrète en cosinus (DCT) est appliquée afin d'obtenir les N_c **coefficients MFCC** :

$$\forall k \in [1, N_c], C(k) = \sum f(i) \cdot \cos(\frac{2\pi ik}{N} + 0,5)$$

où *f(i)* est la $i^{ème}$ des N_f sorties log du banc de filtres.

3. Dynamic Time Warping

Lorsqu'un locuteur prononce deux fois une même phrase, les spectrogrammes, vecteurs précédemment obtenus, ne seront jamais exactement les mêmes. Il y aura des différences non linéaires dans le temps (rythme) qui nécessitent « **d'aligner les axes temporels** ».

Le problème de l'alignement temporel entre une phrase inconnue et une référence peut se résoudre simplement en explorant tous les chemins possibles. Malheureusement, le nombre de chemins possibles croit exponentiellement avec la longueur des signaux à comparer. Toutefois, le problème peut être résolu de manière efficace par un algorithme de comparaison dynamique qui va mettre en correspondance optimale les échelles temporelles des deux mots.
Les deux signaux sont représentés par deux suites de vecteurs de dimension *n* :

- un vecteur issu de la paramétrisation et appartenant au signal cible : $\forall i \in [1, I], X_i \in \mathbb{R}^n$
- un vecteur appartenant au signal source : $\forall j \in [1, J], Y_j \in \mathbb{R}^n$

Soit $d(i,j)$ la « distance » euclidienne :

$$\text{Si } X_i = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ et } \mathrm{Y}_j = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \text{ alors } d(i,j) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

La distance $d(i,j)$ représente la distance entre le spectre de la référence et le spectre du test aux instants *i* et *j*. En calculant cette distance pour toutes les combinaisons d'instants *i* et *j*, on construit la *matrice des distances locales.*

Soit $g(i,j)$ la distance cumulée sur l'ensemble de la phrase, calculée à partir de la matrice des distances locales en respectant les propriétés de monotonie et d'évolution lente du signal étudié. C'est à dire, les seuls chemins valides arrivant au point (i,j) viennent des points (i-1,j), (i-1,j-1) ou (i,j-1).

$$g(i,j) = \min \begin{cases} g(i-1,j) + d(i,j) \\ g(i-1,j-1) + k \cdot d(i,j) \\ g(i,j-1) + d(i,j) \end{cases}$$

k est un coefficient multiplicatif appliqué au chemin diagonal, habituellement égal à 2 car correspondant au plus grand saut de spectre habituellement, mais que nous prendrons égal à 1 dans le cas de l'alignement des MFCC, pour favoriser justement ces diagonales dans le choix du chemin optimal.

Soit *G* la distance normalisée entre les deux signaux : $G = \dfrac{g(I,J)}{I+J}$

La méthode de la comparaison dynamique consiste à choisir, parmi tous les chemins physiquement possibles, la référence pour laquelle la distance totale G est la plus faible et qui représente le chemin le plus court. La ressemblance idéale se traduit donc par une **diagonale**.

4. Exemple de MLC

Analyse textuelle générée pour la phrase « Voici mon rapport »

```
'Token'
      '.','PUNCTUATION'; 0; 'GrammarUnit' 0
      'Voici','WORD'; 1; 'GrammarUnit' 1
      ' ','WHITE'; 2
      'mon','WORD'; 3; 'GrammarUnit' 2
      ' ','WHITE'; 4
      'rapport','WORD'; 5; 'GrammarUnit' 3
      '.','PUNCTUATION'; 6; 'GrammarUnit' 4

'Word'
      '.','ENDPUNCT','UNSTRESSED';0; 'GrammarUnit' 0 ; 'Phoneme' 0 ; 'Syllable' 0
      'voici','PREP','UNSTRESSED';1; 'GrammarUnit' 1 ; 'Phoneme' 1 ; 'Syllable' 1 2
      'mon','DETPOS','UNSTRESSED';2; 'GrammarUnit' 2 ; 'Phoneme' 6 ; 'Syllable' 3
      'rapport','NOUN','PRIMARYSTRESS';3;'GrammarUnit' 3;'Phoneme' 8;'Syllable' 4 5
      '.','ENDPUNCT',''; 4; 'GrammarUnit' 4 ; 'Phoneme' 13 ; 'Syllable' 6

'GrammarUnit'
      'ENDPUNCT','ENDPUNCT'; 0; 'Word' 0 ; 'Token' 0
      'PREP','PREP'; 1; 'Word' 1 ; 'Token' 1
      'DETPOS','DETPOS'; 2; 'Word' 2 ; 'Token' 3
      'NOUN','NOUN'; 3; 'Word' 3 ; 'Token' 5
      'ENDPUNCT','ENDPUNCT'; 4; 'Word' 4 ; 'Token' 6

'Phoneme'
      '_',200,'SILENCE','UNVOICED','','','NUCLEUS',''; 0; 'Word' 0 ; 'Syllable' 0
      'v',78,'FRICATIVE','UNVOICED','','','ONSET',''; 1; 'Word' 1 ; 'Syllable' 1
      'w',65,'GLIDE','UNVOICED','','','CODA',''; 2; 'Syllable' 1
      'a',83,'ORALVOWEL','VOICED','','','NUCLEUS',''; 3; 'Syllable' 1
      's',90,'FRICATIVE','UNVOICED','','','ONSET',''; 4; 'Syllable' 2
      'i',78,'ORALVOWEL','VOICED','','','NUCLEUS',''; 5; 'Syllable' 2
      'm',76,'NASAL','VOICED','','','ONSET',''; 6; 'Word' 2 ; 'Syllable' 3
      'o~',111,'NASALVOWEL','VOICED','','','NUCLEUS',''; 7; 'Syllable' 3
      'R',53,'LIQUID','UNVOICED','','','ONSET',''; 8; 'Word' 3 ; 'Syllable' 4
      'a',83,'ORALVOWEL','VOICED','','','NUCLEUS',''; 9; 'Syllable' 4
      'p',96,'UNVOICEDPLOSIVE','UNVOICED','','','ONSET',''; 10; 'Syllable' 5
      'O',94,'ORALVOWEL','VOICED','','','NUCLEUS',''; 11; 'Syllable' 5
      'R',53,'LIQUID','UNVOICED','','','CODA',''; 12; 'Syllable' 5
      '_',200,'SILENCE','UNVOICED','','','NUCLEUS',''; 13; 'Word' 4 ; 'Syllable' 6

'Syllable'
      '_','SIL','UNDEFINED','P1',200,1; 0; 'Word' 0 ; 'Phoneme' 0
      'vwa','CV','NA','l',226,3; 1; 'Word' 1 ; 'Phoneme' 1 2 3
      'si','CV','NA','l',168,2; 2; 'Word' 1 ; 'Phoneme' 4 5
      'mo~','CV','NA','l',187,2; 3; 'Word' 2 ; 'Phoneme' 6 7
      'Ra','CV','NA','l',136,2; 4; 'Word' 3 ; 'Phoneme' 8 9
      'pOR','CVC','AF','L-L-',243,3; 5; 'Word' 3 ; 'Phoneme' 10 11 12
      '_','SIL','UNDEFINED','P1',200,1; 6; 'Word' 4 ; 'Phoneme' 13
```

5. Structure Matlab des diphones

```
diph = struct( 'target_cost_func', 'distance_euclidean', ...
               'target_feature',...
                                  struct(...
                      'name',{   Feature_Types,       'lexicalword',...
                                 'phoneticword',      'phoneticsyllable',...
                                 'firstinsent',       'firstinword',...
                                 'firstinsyll',       'lastinsent',...
                                 'lastinword',        'lastinsyll',...
                                 'toneofaccent',      'accentofsyllable',...
                                 'grammaticalnature',...
                                 'f0',                'normalizedflow',...
                                 'posinsent',        'posinword',...
                                 'posinsyll',...
                                 }, ...
                      'chval',{  'mean',     'mean',...
                                 'mean',     'mean',...
                                 'mean',     'mean',...
                                 'mean',     'mean',...
                                 'mean',     'mean',...
                                 'mean', ...
                                 'mean',     'mean',...
                                 'mean',     'mean',...
                                 'mean',     'mean',...
                                             'mean',...
                        }, ...
                      'weight',{500,       33.324,...
                                14.537,    0.31733,...
                                9.0865,    17.462  ,...
                                11.283,    10.279  ,...
                                4.0615,    28.54 ,...
                                33.89,     18.983, ...
                                5,...
                                0 ,         0,...
                                21.424,     2.6905 ,...
                                0,...
                                   }, ...
                      'func',{   @distance_binary, @distance_binary,...
                                 @distance_binary, @distance_binary,...
                                 @distance_binary, @distance_binary,...
                                 @distance_binary, @distance_binary,...
                                 @distance_binary, @distance_binary,...
                                 @distance_binary, @distance_binary,...
                                 @distance_binary, ...
                                 [],               [],...
                                 [],               [],...
                                 [],...
                                      } ...
                    ), ...
                'Wt', 1, ...
                'concat_cost_func', 'concat_euclidean', ...
                'concat_feature',...
                                    struct(...
                      'name',{   'lexicalword',...
                                 'phoneticword',       'phoneticsyllable',...
                                 'firstinsent',        'firstinword',...
                                 'firstinsyll',        'lastinsent',...
                                 'lastinword',         'lastinsyll',...
                                 'toneofaccent',       'accentofsyllable',...
                                 'grammaticalnature',  'f0',...
```

```
                'f0',                 'f0',...
                'flow',               'normalizedflow',...
                'posinsent',          'posinword',...
                'posinsyll',          'prevunit',...
                'unituid',...
                   }, ...
    'chval',{  'mean',...
                'mean',     'mean',...
                'mean',     'mean',...
                'mean',     'mean',...
                'mean',     'mean',...
                'mean',     'mean',...
                'mean',     'mean',...
                'startval', 'endval',...
                'mean',     'mean',...
                'mean',     'mean',...
                'mean',     'mean',...
                'mean',...
      }, ...
    'weight',{0,...
                0 ,    0 ,...
                0 ,    0 ,...
                0 ,    0 ,...
                0 ,    0 ,...
                0,     0,...
                0,     0   ,...
                0,     0,...
                0,     0  ,...
                0 ,    0 ,...
                0 ,    58, ...
                0,...
                   }, ...
    'func',{   @concat_binary,...
                @concat_binary, @concat_binary,...
                @concat_binary, @concat_binary,...
                @concat_binary, @concat_binary,...
                @concat_binary, @concat_binary,...
                @concat_binary, @concat_binary,...
                @concat_binary, [],...
                @concat_acoustic,@concat_null,...
                [],               [],...
                [],               [],...
                [],              @concat_prevunit,...
                @concat_null,...
                     } ...
     ), ...
'Wc',1 , ...
'corpusmean', 0, ...
'corpusstd',  1,  ...
'beamwidth',  50000, ...
'unittypes',  {{}},  ...
'preselection', preselection,...
'loudnesslimit', []  );
```