Reconnaissance de phonèmes par analyse formantique dans le cas de transitions voyelle-consonne.

Introduction.

Dans le but d'améliorer la communication homme-machine, on s'est assez vite intéressé, dans la deuxième moitié du vingtième siècle, au problème de la reconnaissance vocale. En 1959 est mis au point un système monolocuteur de reconnaissance des chiffres. Mais la difficulté du problème dans sa généralité a été sous-estimée: si un programme ARPA avait été lancé dans les années soixante-dix aux États-Unis, il faut attendre 1997 pour qu'IBM propose le premier logiciel de dictée vocale, et nombreux sont encore les écueils inhérents à la reconnaissance vocale: un processus complexe de la production de la parole à la détermination de l'orthographe, le bruit de fond, des différences parfois importantes suivant les accents et les dialectes au sein d'une même langue, l'incapacité de traiter le bilinguisme, situation commune dans certains pays, ainsi que des besoins très importants en calcul, rendant difficile la reconnaissance en temps réel.

Nous considérons ici un cas particulier du problème, en nous intéressant uniquement aux transitions entre voyelles non nasalisées et consonnes sonantes, et en nous arrêtant au niveau de la reconnaissance du phonème. Nous proposons un algorithme de séparation des phonèmes, basé sur la différence entre les niveaux d'énergie entre consonnes et voyelles, ainsi qu'un algorithme de reconnaissance par analyse des formants, qui sont autant d'occasions de s'intéresser à la nature de la voix humaine et de relier l'aspect physique à la phonologie, étude des éléments linguistiquement discriminants de la voix.

I. Éléments de phonétique et de phonologie.

1. Le son et l'audition humaine

Le son est une onde qui se propage dans un milieu matériel sous la forme de faibles variations de pression. Il est perceptible à l'oreille humaine lorsque sa fréquence se situe globalement entre 20 Hz et 20 kHz au maximum. Toutefois, on estime que l'information phonétique se trouve en-dessous de 10 kHz. Dans le cas d'une communication téléphonique, les fréquences au-dessus de 3,5 kHz sont coupées, mais la conversation reste intelligible, certains phonèmes très aigus, comme [S], étant toutefois mal rendus.

Une caractéristique assez importante de l'audition humaine, connue sous le nom de « loi d'Ohm » dans le domaine de l'acoustique, est que notre oreille n'est pas sensible à la phase du signal, ce qui justifie que nous ne nous intéressions qu'au module dans les applications qui suivent.

2. La voix humaine

La voix humaine résulte de l'interaction entre l'air expiré et les divers organes phonatoires sur

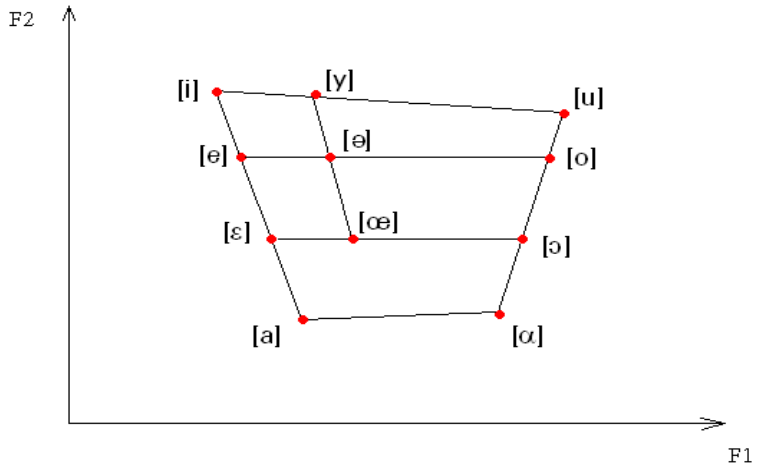
son parcours. Dans le cas de phonèmes voisés, le son est initialement produit par la vibration des cordes vocales. Il est ensuite traité de manière différente suivant les cavités par lesquelles il passe, le pharynx et la bouche principalement. Ces cavités agissent comme des résonateurs qui renforcent pour certains phonèmes les fréquences correspondant à leur fréquence de résonance. Ces fréquences renforcées sont appelées formants, et ce sont eux que les phonologistes essaient de repérer dans un spectrogramme pour reconnaître les phonèmes prononcés. Il est intéressant de noter que la cavité nasale, elle, atténue les formants des phonèmes dits nasalisés, et c'est pourquoi on parle de phénomène d'anti-résonance.

Ces organes phonatoires n'étant pas figés, leur forme varie en fonction du mode d'articulation propre à un phonème particulier, et leurs fréquences de résonances aussi, d'où l'apparition de formants différents pour chaque phonème. En pratique, on considère les trois ou quatre premiers formants, les deux premiers étant les plus liés au mode d'articulation. Dans cette étude, on a décidé d'en retenir quatre, notés F1, F2, F3 et F4.

3. Phonétique et phonologie

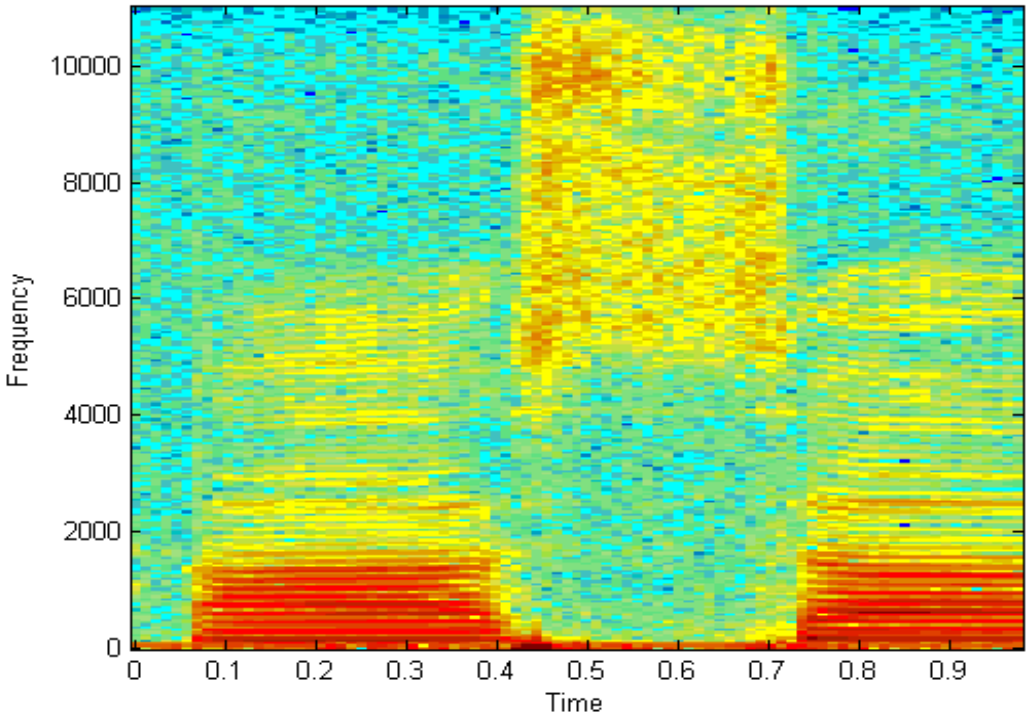
La phonétique est l'étude des phonèmes, ces derniers étant souvent définis comme « la plus petite unité phonique distinctive ». Par exemple, l'opposition des mots « pain » et « bain » montre que [p] et [b] sont des phonèmes. On peut les classer en plusieurs classes et sous-classes, le premier niveau étant celui des voyelles et des consonnes.

Les voyelles sont des sons qui ont une durée assez longue, caractérisés par un libre écoulement et une grande constance de leurs propriétés fréquentielles dans le temps: en l'absence d'éléments prosodiques trop marqués, les formants sont horizontaux sur un spectrogramme. On classe souvent les voyelles au moyen du trapèze vocalique, reproduit ci-dessous, qui représente leur position dans le plan défini par les deux premiers formants et traduit la position de la langue lors de l'articulation (voyelles postérieures ou antérieures, ouvertes ou fermées):



Les consonnes sont des phonèmes qui, lors de leur articulation, rencontrent un obstacle (lèvres pour les voyelles labiales, dents pour les dentales, fermeture du palais pour [k], etc.). Elles sont généralement beaucoup plus courtes que les consonnes et beaucoup plus variables dans le temps. Elles peuvent être bruyantes ou sonantes. Ce n'est que dans ce dernier cas qu'elle présentent des formants.

La figure ci-dessous représente le spectrogramme de la syllabe [asa] et illustre quelques-uns des concepts formulés précédemment. On perçoit très clairement l'horizontalité des formants de la voyelle [a] (même si sur cette image non traitée, ils ont difficiles à distinguer d'autres bandes fréquentielles « parasites »). La consonne [s] est un bruit très aigu situé au-dessus de 5 kHz, clairement exempt de formants.



II.Transformée de Fourier

La transformée de Fourier permet de réaliser une analyse temps-fréquence avec une résolution suffisante pour le signal vocal, quasi-stationnaire sur des intervalles de l'ordre de 10 à 100 ms.

1. Transformée de Fourier

On se place dans l'ensemble des fonctions de carré intégrable $L^2(\mathbb{R})$, qui est un préhilbertien, et.on considère la famille orthogonale des sinusoïdes $\{e_\omega = t \to \mathrm{e}^{i\omega t} / \omega \in \mathbb{R}\}$ (bien sûr, physiquement, on se limite aux pulsations positives). La transformée de Fourier F (ou TF(f)) d'une fonction f permet de réaliser une projection sur l'espace vectoriel engendré par les sinusoïdes, en calculant tout d'abord la composante de la projection sur chaque sinusoïde, donnée par le produit scalaire, et en n'oubliant pas la conjugaison complexe par rapport à la seconde place:

$$F(\omega) = \langle f \mid e_\omega \rangle = \int_{-\infty}^{+\infty} f(t)\mathrm{e}^{-i\omega t}\,dt$$

Dans notre étude, cette transformée ne nous servira jamais à reconstruire le signal, mais de manière plus qualitative à examiner les contributions de chaque plage de fréquence dans le signal vocal en étudiant le spectrogramme, qui est en fait une représentation tridimensionnelle indiquant la magnitude (module de F(ω)) dans le plan temps-fréquence. (D'après la loi d'Ohm, l'oreille n'est pas sensible à la phase du signal auditif.)

Toutefois, il faut adapter cette représentation continue au cas discret, qui est celui du calcul numérique.

2. Transformée de Fourier discrète

Le signal est représenté par des échantillons prélevés uniformément dans le temps:

$$\{f(n) / n \in [\![0, N-1]\!]\}$$

On en fait un signal de période N afin d'éviter les effets de bord. La transformée de Fourier discrète est alors donnée par:

$$F(k) = \sum_{n=0}^{N-1} f(n) \mathrm{e}^{\frac{-2\,i\pi k n}{N}}$$

La fréquence d'observation est proportionnelle au facteur $\frac{k}{N}$, pour $k \in [0, \frac{N}{2}]$: si f_e est la fréquence d'échantillonnage, alors $f = f_e \cdot \frac{k}{N}$. Il faut noter que cette transformée est redondante, car l'information obtenue est concentrée dans la moitié de l'intervalle $[\![0, N-1]\!]$. En effet:

$$\forall k \in \mathbb{Z},\ F(N-k) = \sum_{n=0}^{N-1} f(n) \mathrm{e}^{\frac{2\,i\pi k n}{N}} \mathrm{e}^{-2\,i\pi n} = \sum_{n=0}^{N-1} f(n) \mathrm{e}^{\frac{2\,i\pi k n}{N}} = \overline{F(k)}$$

et donc le module est le même, la phase étant simplement de signe opposé. Si l'on veut analyser le signal sur une plage de 10 kHz, il faut donc prendre une fréquence d'échantillonnage de 20 kHz.

Dans la pratique, on implémente la transformée de Fourier rapide (FFT, pour Fast Fourier Transform) qui a une complexité en $O(N\log_2(N))$ au lieu de $O(N^2)$ par calcul direct et qui tire parti de cette redondance pour optimiser le calcul.

3. Transformée de Fourier et convolution

En représentation continue, la convolution entre deux fonctions est définie par:

$$\forall x \in \mathbb{R},\ (f * g)(x) = \int_{-\infty}^{+\infty} f(t)\, g(x-t)\, dt$$

Par changement de variable en $u = x - t$, l'opérateur de convolution est commutatif. Il est de plus associatif:

$$((f * g) * h)(x) = \int_{-\infty}^{\infty} f * g(x-y)\, h(y)\, dy$$

$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x-y-t)\, g(t)\, h(y)\, dt\, dy$$

$$= \int_{-\infty}^{+\infty} g(t) \left(\int_{-\infty}^{\infty} f(x-t-y)\, h(y)\, dy \right) dt$$

$$= \int_{-\infty}^{+\infty} g(t)(f * h)(x-t)\, dt$$

$$= (g * (f * h))(x)$$

et donc par commutativité on a $(f * g) * h = (f * h) * g$.

On va prouver que la transformée de Fourier de la convolée de deux fonctions est le produit usuel des transformées de Fourier. De la définition découle l'égalité: $F(\omega) = (f * e_\omega)(0)$. De plus, on a: $\forall x \in \mathbb{R},\ (f * e_\omega)(x) = F(\omega) \times e_\omega(x)$ car:

$$(f * e_\omega)(x) = \int_{-\infty}^{+\infty} f(t) \mathrm{e}^{i\omega(x-t)} = \mathrm{e}^{i\omega x} \int_{-\infty}^{+\infty} f(t) \mathrm{e}^{-i\omega t} = F(\omega)\, e_\omega(x)$$

Finalement, la transformée de Fourier de $h = f * g$ est donc, en utilisant successivement la première propriété, l'associativité puis la seconde propriété:

$H = (e_\omega * f * g)(0) = ((e_\omega * f) * g)(0) = F(\omega) \times (e_\omega * g)(0) = F(\omega) \times G(\omega)$, soit:

$$TF(f * g) = F \times G \quad (1)$$

Ce résultat est également valable en représentation discrète. Il nous servira en analyse formantique du fait de la modélisation adoptée pour le signal de voix.

4. Problème du fenêtrage

En pratique, non seulement le signal s est discret, mais de plus le temps d'observation 2τ est fini. On observe donc la convolée du signal par la fonction porte:

$$\Pi : t \to 1 \ \ si\, t \in [-\tau, \tau]$$
$$0 \ \ sinon$$

D'après (1), la transformée de Fourier de $s * \Pi$ est le produit des transformées, celle de la fonction porte étant:

$$\int_{-\infty}^{+\infty} \Pi(t) e^{-i\omega t}\, dt = \int_{-\tau}^{\tau} e^{-i\omega t}\, dt = 2\tau\, sinc(\omega\tau)$$

Le lobe central de ce sinus cardinal a une largeur de $\frac{2}{\tau}$. En conséquence, lorsque le temps d'observation tend vers zéro, le spectre s'élargit. Pour résoudre ce problème, on utilise des fenêtres dont l'énergie est concentrée en zéro, ce qui limite ce phénomène. Dans nos travaux, nous avons utilisé la fenêtre de Hamming.

III.Analyse formantique

Les algorithmes proposés, dont le code source figure en annexe, ont été implémentés en MATLAB, un logiciel de calcul numérique, sur un ordinateur personnel cadencé à 2,4 GHz sous Microsoft Windows. La fréquence d'échantillonnage est de 22050 Hz, ce qui permet d'observer le signal sur une plage de plus de 10 kHz (nécessaire pour reconnaître le phonème [s], dont l'énergie est concentrée entre 6 et 8 kHz, mais finalement sa reconnaissance n'a pas été implémentée).

1. Problème du décodage acoustico-phonétique

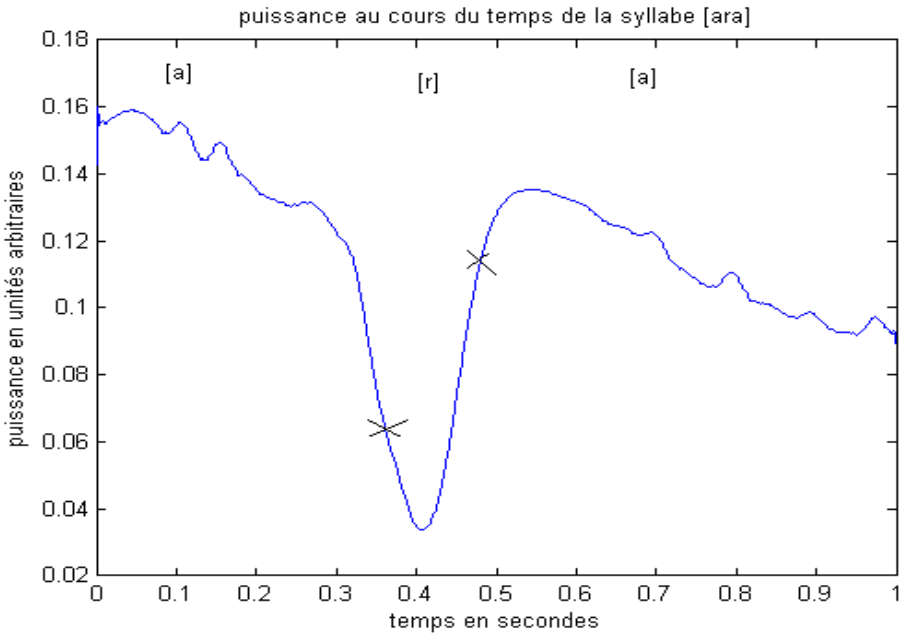
Le signal de voix étant continu, il n'est pas évident de reconnaître dans le signal enregistré les différents éléments linguistiques que sont les mots, les syllabes et les phonèmes. C'est ce problème que l'on appelle le décodage acoustico-phonétique. Nous avons ici mis en oeuvre un procédé permettant de localiser les transitions entre voyelles et consonnes. Les syllabes correspondant à ce cas sont qualifiés d'ouvertes. Elles sont très majoritaires dans certaines langues, telles les langues polynésiennes ou le japonais dans une moindre mesure, mais ce n'est pas le cas dans les langues latines où deux consonnes peuvent souvent se suivre, ou pire, dans les langues slaves, tel le tchèque. Ce n'est donc qu'une solution partielle au problème, d'autant plus que les mots à analyser doivent avoir été au préalable débarrassés des silences initial et final et que le nombre φ de phonèmes à détecter doit être fourni au programme de reconnaissance.

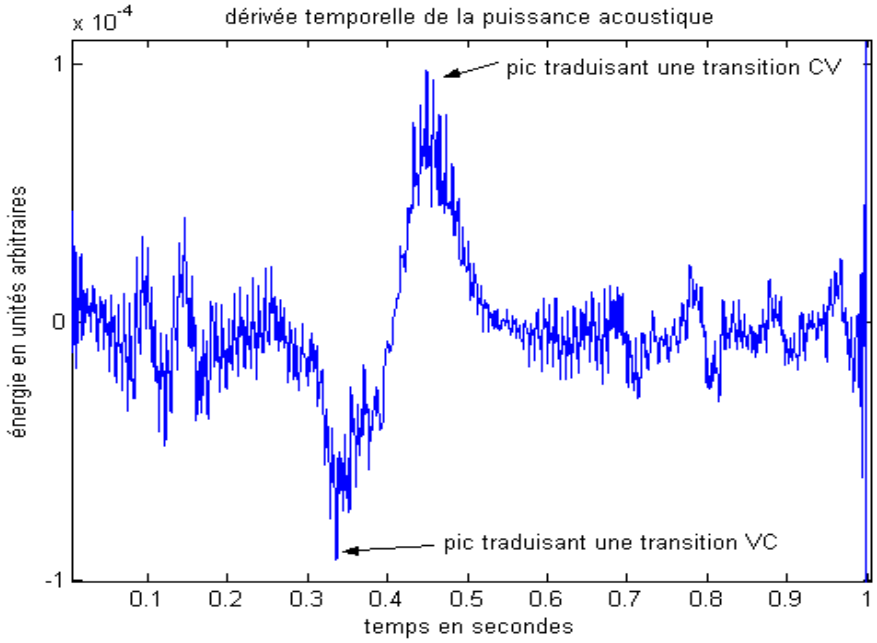
2. Détection des transitions CV et VC (consonne-voyelle et voyelle-consonne)

Pour éliminer au maximum le bruit de fond, on procède tout d'abord à un filtrage numérique. Une analyse préalable ayant montré que, dans les conditions de l'expérience, le bruit de fond est surtout localisé en dessous de 100 Hz (et principalement à 50 Hz, fréquence à laquelle tournent les ventilateurs de l'ordinateur), on invoque tout d'abord la procédure coupe100 qui calcule la FFT du signal, coupe les harmoniques en-dessous de 100 Hz (en tenant compte de la redondance) et renvoie

la FFT inverse.

L'algorithme proposé (implémenté par la fonction separe) est basé sur le fait que la puissance sonore des consonnes est généralement bien inférieure à celle des voyelles. Nous commençons par calculer la moyenne quadratique en chaque point sur une fenêtre suffisamment large pour être supérieure à la période éventuelle du phonème (qui à ce stade n'est évidemment pas connue). Le signal est alors dérivé, après avoir été lissé pour éviter l'apparition de pics dus à de simples fluctuations, puis la dérivée est elle-même lissée afin d'être plus facile à traiter. Par la suite, on recherche les (φ-1) pics les plus importants en valeur absolue, qui correspondent donc aux varitions les plus brutales (On exclut le bord, sur lequel se produisent des phénomènes aberrants). Si ce pic est négatif, la puissance a subi une forte baisse, et donc on est en présence d'une transition VC, dans le cas contraire il s'agit d'une transition CV. Les graphiques ci-dessous illustrent notre méthode pour la syllabe [ara]:





On estime de manière empirique (après d'une série d'expériences), que la transition est située à un instant dans le voisinage du pic où, par rapport au phonème précédent, le signal a perdu ou gagné une puissance égale aux deux tiers de l'écart absolu de puissance entre les deux phonèmes. Sur le graphe, les transitions sont marquées par des croix.

La procédure sépare renvoie les instants où s'effectuent les transitions au sein de l'enregistrement ainsi qu'une variable « mode » qui vaut 'c' si l'enregistrement commence par une consonne, 'v' dans le cas d'une voyelle.

3. Analyse formantique

Nous avons choisi de reconnaître par analyse formantique les voyelles non nasalisées [i], [e], [ɛ], [a], [u], [o], [ɔ], [ɑ], [y], [ø], [œ] et [ə] ainsi que les sonantes impulsionnelles [m], [n], [ɲ], et continues [l] et [r], laissant de côté les semi-consonnes [w] et [j], qui sont également des sonantes continues, mais qui prêtent trop à confusion pour le type de reconnaissance effectué.

Avant de reconnaître les formants, il faut d'abord effectuer un traitement visant à les révéler de manière plus marquée.

En effet, on constate que la magnitude décroît dans le spectrogramme à raison de 6 dB par octave. Pour pallier à cette atténuation, qui rend difficile la détection des derniers formants, on accentue le signal de voix v(n) en calculant la grandeur $v'(n) = v(n) - \alpha v(n-1)$ avec $v'(0) = v(0)$. Plus α est grand, plus la magnitude est réhaussée en haute fréquence. Dans notre expérience, nous avons choisi $\alpha = 2$.

En outre, les formants sont masqués par d'autres effets. Le plus souvent, le signal de voix v(t) est modélisé par le produit de convolution entre le signal de source e(t) dû à la vibration des cordes vocales, et le signal h(t) dû aux différents résonateurs. Le traitement homomorphique permet, à l'issue du calcul d'une grandeur appelée « cepstre », d'éliminer l'influence de la source. Pour effectuer la déconvolution, on découpe le signal en morceaux sur lesquels on applique la transformée de Fourier fenêtrée. On a alors $V(\omega) = E(\omega) H(\omega)$, et on prend le logarithme du module (la phase n'a aucune importance) pour passer à une représentation additive: $\ln|V(\omega)| = \ln|E(\omega)| + \ln|H(\omega)|$. On effectue alors une transormée inverse, ce qui donne le cepstre: $c(t') = FT^{-1}(\ln|V(\omega)|) = FT^{-1}(\ln|E(\omega)|) + FT^{-1}(\ln|H(\omega)|)$. Le cepstre s'exprime dans une grandeur appartenant à une sorte d'échelle temporelle déformée par le logarithme que l'on appelle « quéfrence ». La contribution de la source correspond aux hautes quéfrences, et celle du conduit aux basses quéfrences. Il faut donc déterminer la quéfrence de coupure à partir de laquelle on n'a plus que la contribution de la source. En pratique, il suffit de trouver la quéfrence à partir de laquelle le cepstre ne comprend plus de pics et reste relativement bas. Cette opération s'appelle le liftrage. Par la suite, on peut éventuellement recalculer h(t): $h(t) \approx FT^{-1}(\exp(C(\omega')))$, mais pour obtenir le spectrogramme « corrigé », on se contente de $H(\omega) \approx \exp(C(\omega'))$.

Ce traitement est mis en oeuvre par les modules accentue et deconvol.

Une fois le spectre corrigé obtenu, on cherche alors chaque formant dans une bande de fréquence précisée le plus finement possible afin d'avoir suffisamment de chances de trouver le formant, qui correspond à la magnitude moyenne la plus élevée. Cette opération est réalisée par le module formants. Des estimations de valeurs formantiques (en Hertz) pour une voix d'homme sont données ci-dessous. Pour chaque formant on a indiqué les valeurs extrémales. Il convient de se rappeler que ces valeurs peuvent varier grandement en fonction de l'individu et au sein même du discours.

voyelle	F1	F2	F3	F4
[i]	**250**	**2250**	**2980**	3280
[e]	420	2050	2630	3340
[ɛ]	590	1770	2580	**3480**
[a]	**760**	1450	2590	3280
[u]	290	**750**	2300	3080
[o]	360	770	2530	3200
[ɔ]	520	1070	2510	3310
[ɑ]	710	1230	2700	3700
[y]	**250**	1750	**2160**	**3060**
[ø]	350	1350	2250	3170
[œ]	500	1330	2370	3310
[ə]	570	1560	2560	3450

consonne	F1	F2	F3	F4
[m]	**300**	1300	**2300**	2770
[n]	350	1050	**2300**	**3470**
[ɲ]	**360**	**1000**	2400	3300
[l]	**360**	**1700**	**2500**	3300
[r]	550	1300	**2300**	**2700**

Lorsqu'on a déterminé les valeurs des formants, les procédures reconnaissance_voyelles et reconnaissance_consonnes vont calculer la distance formantique avec chacune des valeurs en mémoire. La distance formantique n'est autre qu'une distance euclidienne; entre les phonèmes A et B, elle vaut $d_{A,B} = \sqrt{\sum_{i=1}^{4} (F_A^i - F_B^i)^2}$.

4. Fonctionnement global de l'algorithme de reconnaissance.

La reconnaissance est orchestrée par la procédure reconnaissance. Cette dernière prend en argument l'enregistrement, sans silences de début et de fin puis demande à separe les positions des phonèmes et quel type de phonème (consonne/voyelle) est en première position. Grâce à ces données, elle peut selon le cas envoyer le phonème à reconnaissance_voyelles ou à reconnaissance_consonnes, qui agissent comme décrit ci-dessus, puis renvoie le numéro que l'on a décidé d'associer à chaque phonème. La séquence de ces numéros est alors renvoyée par reconnaissance à l'utilisateur.

Évidemment, avant de lancer reconnaissance, il faut être passé par une phase d'apprentissage, réalisée par les scripts apprentissage_voyelles et apprentissage_consonnes. Trois échantillons sont pris pour chaque phonème, afin d'optimiser la reconnaissance par la suite. Les données sont alors stockées sous forme de tableau dans le répertoire de travail (qui doit être le répertoire courant de MATLAB), sous les noms de formants_voyelles et formants_sonantes.

IV.Discussion sur les résultats de ce traitement

1. Séparation

L'algorithme de séparation des phonèmes fonctionne de manière très satisfaisante du moment que:

- les silences de début et de fin aient été supprimés;
- le locuteur ne souffle pas sur le microphone durant l'enregistrement, ce qui porte le signal au niveau de saturation et se traduit par des pics très importants que l'algorithme prend pour une transition;
- on se limite à six phonèmes maximum (au-delà, l'algorithme ne parvient plus à distinguer les varitions de puissance entre deux phonèmes et les fluctuations).

2. Reconnaissance des voyelles

Le programme de reconnaissance n'est en définitive pas suffisamment élaboré pour discriminer sans faille chacune des voyelles. En particulier, il est malgré tout relativement dépendant de la hauteur des voyelles gardées en mémoire; toutefois, une fois que l'on se maintient à cette hauteur, la reconnaissance se fait parfaitement.

Les confusions sont cependant loin d'être aléatoires et ont ont un lien avec la position relative des voyelles sur le trapèze vocalique:

- Le groupe [a]/[ɑ] est parfaitement reconnu, du fait de sa position particulière à la pointe inférieure du triangle vocalique. On n'en demandera pas plus dans le cas d'un locuteur parlant une variété de français dans laquelle la diférence entre ces deux phonèmes n'est pas très claire.
- [e] est parfois confondu avec [ə], ce qui est cohérent car la seule chose qui les différencie sur le triangle vocalique est leur degré d'ouverture
- [o] et [u] le sont aussi pour les mêmes raisons. Si leur proximité ne semble pas claire de prime abord, il n'est que de penser que, au sein des langues latines, par exemple en occitan, en catalan et en portugais (mais aussi en français: penser à la querelle entre « oïstes » et « ouïstes » en moyen-français), un /o/ affaibli se prononce justement [u].
- [ɔ] et [ɛ] sont beaucoup plus rarement confondus respectivement avec [a], toujours pour des raisons de proximité vocalique.
- Le groupe [ə], [ø], [œ] est très bien reconnu. Ne différenciant pas très clairement ces phonèmes entre eux, je ne saurais me prononcer sur leur reconnaissance.
- Enfin, *last but not least*, la confusion entre [i] et [y] est très intéressante: un espagnol apprenant le français ne pourra pas (du moins de prime abord) articuler [y], qu'il remplacera instinctivement par [i]. De même, un catalan du sud apprenant l'occitan (une langue pourtant très semblable à la sienne), risque de se faire entendre dire « Parles occitan coma la gènt de Perpinhan »...

3. Reconnaissance des consonnes sonantes

La reconnaissance des sonantes fonctionne hélas beaucoup mois bien. Cela doit être principalement dû au fait que notre algorithme cherche formants horizontaux quand en réalité ils peuvent varier dans le temps dans le cas des consonnes, et ce, de plus, en fonction des voyelles précédant et suivant les consonnes, les effets de transition étant alors beaucoup plus importants que chez les voyelles, qui durent bien plus longtemps. On pourrait suivre de manière plus précise les trajectoires formantiques, mais alors il faudrait implémenter des algorithmes de comparaison plus complexes et plus coûteux en temps de calcul.

Si certaines consonnes sont cependant assez bien reconnues, comme [n] et [r], (à hauteur d'environ 50 %) on a affaire aux confusions suivantes:

- [m] et [ɲ] sont presque systématiquement pris pour [n]. Cela a encore une cohérence, car ce sont là les trois consonnes nasales du français, dont [n] est peut-être plus intense.
- [l] est souvent pris pour [r]: cette fois-ci, c'est difficilement explicable.

Conclusion

Cette approche de la reconnaissance vocale nous a permis de cerner certaines caractéristiques phonétiques et de mettre en oeuvre des solutions pour les exploiter. Si le logiciel, à l'arrivée, ne permet pas d'identifier avec précision chaque phonème, il met en évidence la proximité d'un bon nombre d'entre eux. Pour obtenir de meilleurs résultats, plusieurs voies sont envisageables: une meilleure détermination des paramètres du problème (coefficient d'accentuation, quéfrence de coupure du cepstre...) ainsi qu'une analyse plus précise des trajectoires et des transitions formantiques et une meilleure prise en compte de la prosodie dans le discours.

Annexe 1: Alphabet Phonétique International (API).

Pour chacun des phonèmes utilisés, nous donnons un exemple afin d'aider à la prononciation:

[i]	p**i**e	[ø]	p**eu**
[e]	th**é**	[œ]	b**oeu**f
[ɛ]	f**ai**t	[ə]	d**e**
[a]	l**a**	[m]	a**m**i
[u]	b**ou**e	[n]	u**n**i
[o]	sc**eau**	[ɲ]	oi**gn**on
[ɔ]	b**o**sse	[l]	**îl**ot
[α]	p**â**te	[r]	ha**r**as
[y]	r**u**e		

Annexe 2: code source

microphone:

```
function son = micro(t,fs);

% prend en argument la durée de l'enregistrement et la fréquence
% d'échantillonnage, renvoie l'enregistrement

AI = analoginput('winsound');
chan = addchannel (AI,1);
duree = t;
set(AI,'SampleRate',fs)
vraiefs = get(AI,'SampleRate'); % vraie fréquence d'échantillonnage
set(AI,'SamplesPerTrigger',duree*vraiefs)
set(AI,'TriggerType','Manual')
blocksize = get(AI,'SamplesPerTrigger');
Fs = vraiefs;

% Réaliser l'acquisition
start(AI)
Trigger(AI)
data = getdata(AI);
```

apprentissage_consonnes:

```
clc;
fs=22050;
load bandes_sonantes;
bandes = bandes_sonantes;

load ConsApp;
[n,k]=size(ConsApp);
n=floor(n/3);
formants_sonantes=zeros(n,4,3);

disp('***Reconnaissance des formants***')
disp(['Vous allez devoir prononcer chaque groupe VCV pendant une seconde en laissant les
voyelles déborder sur le temps d enregistrement']);
disp('Ctrl-pause pour interrompre le processus');
fprintf('\n');

for i=1:n
   for c=1:3
      en_cours=1;
      while en_cours
         input(['Prononcez la syllabe ' ConsApp(3*(i-1)+c,:)]);
         disp('enregistrement...');
         syl=microphone(1,fs);
         disp('OK');
         fprintf('\n');
```

```
        % Extraction de la consonne sonante
        S=separe(syl,3);
        if (S(2)-S(1))*1000/fs >= 34 % La sonante doit au moins durer 33 ms
           en_cours=0;
        else
           disp('cet enregistrement s''est mal effectué');
        end;
     end;
     consonne=syl(S(1):S(2));
     %Analyse formantique de la consonne sonante
     formants_sonantes(i,:,c)=formants(coupe100(consonne),bandes);
  end;
end;

save formants_sonantes formants_sonantes;
```

apprentissage_voyelles:

```
clc;
fs=22050;
n_essai=3;
load bandes_voyelles;
bandes=bandes_voyelles;

load voyelles;
[n,k]=size(voyelles);
formants_voyelles=zeros(n,4,n_essai);

disp('***Reconnaissance des formants***')
disp(['Vous allez devoir prononcer chaque voyelle pendant une seconde et prononcer la série
plusieurs fois de suite']);
disp('Ctrl-pause pour interrompre le processus');
disp('Commencez à parler avant d appuyer sur entrée');
fprintf('\n');

for i=1:n_essai
  for v=1:n
     input(['Prononcez la voyelle ' voyelles(v,:)]);
     disp('enregistrement...');
     voy=coupe100(microphone(1,fs));
     disp('OK');
     fprintf('\n');
     formants_voyelles(v,:,i)=formants(voy,bandes);
  end;
end;

save formants_voyelles formants_voyelles;
```

coupe100

```
function pur=coupe100(X);
```

```
fs=22050;
N=length(X);
Y=fft(X);
Y(round(101*N/fs):round(N-101*N/fs))=0;
pur=X-real(ifft(Y)); %ifft renvoie des nombres au format « complexe »
```

reconnaissance

```
function suite_phonemes=reconnaissance(X,fi);

%renvoie la suite des numéros correspondant aux phomèmes étudiés
%module « chef d'orchestre »

suite_phonemes=zeros(1,fi);
enregistrement=coupe100(X); %suppression du bruit
n=length(enregistrement);

[L,mode]=separe(enregistrement,fi); %mode: commence par une voyelle ou une consonne
L=[1,L,n]; %position des phonemes

for i=1:2:fi %reconnaissance du groupe représenté par le premier phonème
   phoneme=enregistrement(L(i):L(i+1));
   if mode=='v'
      suite_phonemes(i)=reconnaissance_voyelles(phoneme);
   else
      suite_phonemes(i)=reconnaissance_consonnes(phoneme);
   end;
end;

for i=2:2:fi % Au tour du second groupe
   phoneme=enregistrement(L(i):L(i+1));
   if mode=='c'
      suite_phonemes(i)=reconnaissance_voyelles(phoneme);
   else
      suite_phonemes(i)=reconnaissance_consonnes(phoneme);
   end;
end;
```

separe

```
function [positions,mode]=separe(X,fi);

%separe(X,fi):
%arguments:
%-X: séquence à analyser (les silences de début et de fin doivent
%avoir été enlevés)
%-fi: nombre de phonèmes que contient la séquence
%renvoie un vecteur contenant les positions des transitions entre phonèmes
%ainsi que la variable mode, qui vaut 'c' si la séquence commence par une
%consonne, 'v' sinon

positions=zeros((fi-1),3);
```

```
N=length(X);
fs=22050; % fréquence d'échantillonnage

%calcule et lisse le vecteur puissance acoustique et sa dérivée
fenetre_puissance=500; %moyennage sur 500 points soit environ 22 ms
fenetre_lissage=400; %lissage sur 400 points soit environ 18 ms
P=zeros(1,N); %initialise le vecteur puissance
for i=1:N
   tranche=X((max(1,i-fenetre_puissance/2)):(min(N,i+fenetre_puissance/2)));
   P(i)=sqrt(mean(tranche.^2));
end;
P=smooth(P,fenetre_lissage);
D=smooth(diff(P),fenetre_lissage);

%recherche des fi-1 pics de la dérivée
separation_non_effectuee=1;
while separation_non_effectuee
   Dplus=abs(D);
   N_D=length(Dplus);
   bord=200; %prévient les effets de bord
   Dplus(1:bord)=0;%(dérivées artificiellement grandes aux extrémités)
   Dplus(N_D-bord:N_D)=0;
   for i=1:(fi-1)%recherche des pics
      [val,pos]=max(Dplus);
      positions(i,2)=pos;     %positions est une matrice (fi-1) lignes, 3 colonnes Ci
      signe_pic=sign(D(pos)); %C2 correspond aux pics, C1 et C3 aux zéros à gauche et à
droite
      indexmin=pos;
      while (signe_pic*D(indexmin)>0) && (indexmin>1)
         indexmin=indexmin-1;
      end;
      positions(i,1)=indexmin;
      indexmax=pos;
      while (signe_pic*D(indexmax)>0) && (indexmax<N_D)
         indexmax=indexmax+1;
      end;
      positions(i,3)=indexmax;
      Dplus(indexmin:indexmax)=0; % suppression de ce pic, on passe au suivant
   end;
   %tri par ordre croissant selon la seconde colonne du vecteur positions
   for i=1:(fi-2)
      [val,pos]=min(positions((i:(fi-1)),2));
      pos=pos+i-1; % pos est une position relative !
      echange=positions(i,:);
      positions(i,:)=positions(pos,:);
      positions(pos,:)=echange;
   end;
   %Ara cal fer el balanç...
   if D(positions(1,2))*D(positions(2,2)) < 0
      separation_non_effectuee = 0;
   else
      %Suppression du pic trouble-fete
      pic_gauche=positions(1,2);
```

```
        pic_droit=positions(2,2);
        [val,point_critique]=min(Dplus(pic_gauche:pic_droit));
        point_critique = pic_gauche + point_critique;
        D(1:point_critique)=0;
    end;
end;

% affinage des positions selon une "règle des tiers"

if D(positions(1,2)) < 0
    mode='v';
else
    mode='c';
end;
positions_exactes=[];
for i=1:(fi-1)
    gauche=positions(i,1);
    milieu=positions(i,2);
    droite=positions(i,3);
    hauteur=abs(P(droite)-P(gauche));
    if D(milieu) < 0
        P_tiers=P(droite)+hauteur/3;
    else
        P_tiers=P(droite)-hauteur/3;
    end;
    [val,pos]=min(abs(P(gauche:droite)-P_tiers));
    positions_exactes(i)=pos+gauche-1;
end;

positions=positions_exactes;
```

reconnaissance_voyelles

```
function RangVoyelle=reconnaissance_voyelles(voyelle);

% Prend l'enregistrement d'une voyelle en argument, renvoie son numéro

load formants_voyelles;
Nvoy=length(formants_voyelles); % Nombre de voyelles à considérer (plus grande dimension
de formants_voyelles)
load bandes_voyelles;
bandes=bandes_voyelles;

%Calcul des 4 premiers formants
F=formants(voyelle,bandes);

%Calcul des distances avec toutes les formants en mémoire au moyen de la
%distance euclidienne
D=zeros(3,Nvoy); %matrice contenant les distances
for i=1:Nvoy
    for j=1:3
        D(j,i)=sqrt(sum((F-formants_voyelles(i,:,j)).^2));
```

```
   end;
end;

Candidats=zeros(3,2);

for i=1:3
   [dmin,RangVoyelle]=min(D(i,:));
   Candidats(i,1)=dmin;
   Candidats(i,2)=RangVoyelle;
end;

[dminimini,posultime]=min(Candidats(:,1));
Candidats
RangVoyelle=Candidats(posultime,2);
```

reconnaissance_consonnes

```
function RangConsonne=reconnaissance_consonnes(consonne);

fs=22050;
fmax=fs/2;
load formants_sonantes;
load bandes_sonantes;
bandes=bandes_sonantes;
Nson=length(formants_sonantes);

F=formants(consonne,bandes);

% Distance euclidienne avec les autres sonantes
D=zeros(3,Nson); %vecteur contenant les distances
for i=1:Nson
   for j=1:3
      D(j,i)=sqrt(sum((F-formants_sonantes(i,:,j)).^2));
   end;
end;

%recherche du meilleur candidat
Candidats=zeros(3,2);
for i=1:3
   [dmin,RangConsonne]=min(D(i,:));
   Candidats(i,1)=dmin;
   Candidats(i,2)=RangConsonne;
end;
[dminimini,posultime]=min(Candidats(:,1));
RangConsonne=Candidats(posultime,2);

Candidats+12
RangConsonne=RangConsonne+12; %décalage par rapport aux voyelles
```

accentue

```
function X=accentue(x);
```

```
alpha=2;
n=length(x);
X(1)=x(1);
X(2:n)=x(2:n)-alpha*x(1:n-1);
```

deconvol

```
function [Y,taillefft]=deconvol(signal,fs,fenetre,n0)
%SPECT(signal,fs,fenetre)
%   Calcule le spectrogramme en utilisant une fenêtre de Hamming.
%   -signal: signal d'entrée (vecteur colonne)
%   -fs: fréquence d'échantillonnage du signal
%   -fenetre: taille de la fenêtre en ms

nbfram=size(signal);
nbfram=nbfram(1);

frameparfenetre=round(fs*fenetre/1000); % nombre de frames dans une fenêtre
recoupframe=round(frameparfenetre/2); % nombre de frames pour le redécoupage
hamming=0.54-0.46*cos(2*pi*(0:frameparfenetre-1)'/(frameparfenetre-1));

% Trouve la puissance de 2 supérieure à nbfram pour le calcul de la fft
fftsize=2.^ceil(log2(frameparfenetre));

sgrang=2:round(fftsize/2)-1;

Y=zeros(round(fftsize/2)-2,1);

for i=1:(frameparfenetre-recoupframe):(nbfram-frameparfenetre);
   sfen=signal(i:min(i+frameparfenetre-1,end)); % partie du signal dans la fenêtre
   sfen=(accentue(sfen'))';
   sfen=sfen.*hamming; % on applique une fenêtre de Hamming
   sfen=ifft(log(abs(fft(sfen,fftsize))),fftsize);
   sfen(1)=[];
   sfen(n0+1:end-n0)=0;
   xf=abs(exp(fft(sfen,fftsize)));
   Y=[Y xf(sgrang)]; % On ajoute la fft court terme au spectrogramme
end

Y=Y(:,2:end); % On supprime la première colonne de zéros.
```

Formants

```
function F=formants(phoneme,bandes);

%renvoie les fréquences des formants F1, F2, F3 et F4
%dans le vecteur F.
phon=phoneme(1:3:end); %sous-échantillonnage pour profiter pleinement de la résolution
                       %fréquentielle aux basses fréquences
F=zeros(1,4);
fs=22050/3; %fréquence d'échantillonnage
```

```
fmax=fs/2; %fréquence maximale observable après une fft

%bandes dans lesquelles chercher les formants (en Hertz)
%bandes=[235 775;675 2265;2145 2995;3045 3495];

%calcul du spectrogramme du phonème
Tfenetre=23; %fenetre de 23 ms pour la fft
quefrence=floor(1000*length(phon)/fs); %Quéfrence de coupure (sic)
S=deconvol(phon,fs,Tfenetre,quefrence);
magnitude=mean(abs(S')); % moyenne à chaque fréquence
Nf=length(magnitude);

%équivalent des bornes selon l'indexation de la matrice
B=min(round(bandes*Nf/fmax),Nf);


%recherche des formants

i=1;
while i <= 4
   b1=B(i,1);
   b2=B(i,2);
   [val,numpos]=max(magnitude(b1:b2));
   numpos=numpos+b1-1;
   F(i)=round(numpos*fmax/Nf);
   if (i==2) && ((F(2)-F(1)) <= 410) %distance minimale entre F1 et F2
      if F(2) <= 763              %ce sont ceux qui posent le plus problème
         F(1)=F(2);
      end;
      B(2,1)=b1+1;
      i=1;
   end;
   i=i+1;
end;
```

Constantes utilisées par le programme:

bandes_sonantes et bandes_voyelles: matrices à 4 lignes et 2 colonnes contenant les bandes fréquentielles dans lesquelles chercher les formants. Ces valeurs sont ajustées directement par l'utilisateur au moyen de l'interface de MATLAB. Un réajustement est en effet nécessaire si l'on veut reconnaître une voix féminine.

Voyelles et ConsApp: vecteurs colonnes contenant les chaînes de caractères à afficher lors des procédures d'apprentissage.

formants_voyelles et formants_sonantes: Tableau tridimensionnel contenant trois échantillons pour chaque formant, généré automatiquement par le programme à l'issue des procédures d'apprentissage.