



UNIVERSITÉ
LAVAL

DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUE

FACULTÉ DES SCIENCES ET DE GÉNIE

UNIVERSITÉ LAVAL

1065, AVENUE DE LA MÉDECINE

QUÉBEC (QUÉBEC) G1V 0A6

TÉLÉPHONE : +1 (418) 656-2984

TÉLÉCOPIEUR : +1 (418) 656-3159

COURRIER ÉLECTRONIQUE : GEL@GEL.ULAVAL.CA

VOCAL ACTIVITY DETECTION ALGORITHM MODULE COMMANDE DE LA PAROLE

DEVELOPMENT AND INTEGRATION OF A VOICE COMMAND FOR USERS
WITH SPEECH DISABILITIES (DYSTROPHY, PARALYSIS, ETC.) ON THE
JACO ROBOTIC ARM

SOUS LA TUTELLE DE BENOIT GOSSELIN

ET ALEXANDRE CAMPEAU-LECOURS



DÉPARTEMENT DE GÉNIE ÉLECTRIQUE ET DE GÉNIE INFORMATIQUES
FACULTÉ DES SCIENCES ET DE GÉNIE
UNIVERSITÉ LAVAL

Table des matières

1	Introduction	2
2	Notions théoriques	3
2.1	Traitement à court-terme du signal vocal	3
2.2	Méthodes dans le domaine temporel	3
2.2.1	Énergie à court terme	3
2.2.2	Taux de passage par zéro (Zero Crossing Rate, ZRC)	3
2.3	Domaine fréquentiel	4
2.3.1	Entropie Spectrale	4
2.3.1.1	Travaux de Shen J, Hung J et Lee J [10]	4
2.3.1.2	L'entropie spectrale	4
2.3.1.3	Limitation	5
2.3.2	Spectral Centroid	5
2.3.3	Mel Feature Cepstral Coefficients (MFCC)	5
2.3.3.1	Etape pour déterminer les MFCCs	5
3	Travaux réalisés	6
3.1	Fonctionnement général de l'algorithme	6
3.2	La distance ou corrélation entre des vecteurs de bruit de fond et celui de chaque trame	7
3.3	Détermination des seuils	7
3.3.1	Seuil pour les MFCCs	8
3.3.2	Seuil pour l'entropie spectrale	8
3.3.3	Segmentation et correction des labels	9
3.3.3.1	Correction des "uns" et "zéros" isolés	9
3.3.3.2	Segmentation	9
3.3.3.3	Post traitement des segments	10
3.4	Conclusion	11
4	Tests et comparaisons	12
4.1	Présentation des algorithmes évalués	12
4.2	Robustesse au bruit	12
4.2.1	Données des tests	13
4.2.2	Signal avec présence de voix et peu ou pas de bruit (SNR élevé)	13
4.2.2.1	Algorithme utilisant le Spectral centroid et le STE(Short time Energy)	14
4.2.2.2	Algorithme utilisant l'entropie spectrale	14
4.2.2.3	Notre algorithme	15

TABLE DES MATIÈRES

4.2.3	Signal avec un SNR de $-30dB$	16
4.2.3.1	Algorithme utilisant le Spectral centroid et le STE(Short time Energy)	17
4.2.3.2	Algorithme utilisant l'entropie spectrale	17
4.2.3.3	Retour à notre algorithme	18
4.2.4	Signal avec un musique de fond - SNR -13dB	19
4.2.4.1	Algorithme utilisant le Spectral centroid et le STE(Short time Energy)	19
4.2.4.2	Algorithme utilisant l'entropie spectrale	20
4.2.4.3	Notre algorithme	21
4.2.5	Tableau récapitulatif	23
4.2.6	Conclusion	23
4.3	Performances	25
4.3.1	Déroulement	25
4.3.2	Résultat	25
	25
5	Conclusion	27
	Références	28

Table des figures

1	Représentation temporelle du signal utilisé	13
2	Détection d'activité vocale utilisant le centroïde spectrale et l'énergie à court terme : (a) Energie à court terme , centroïde spectrale et leur seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"	14
3	Détection d'activité vocale utilisant l'entropie spectrale : (a) Entropie spectrale et son seuil de détection du Signal vocale ; (b) Segmentation finale du Signal "Mode un deux"	15
4	Notre algorithme de détection d'activité vocale : (a) Distance entre le vecteur de coefficient spectraux de chaque trame et le vecteur de coefficient de bruit de fond ; Distance entre la valeur de l'entropie spectrale de chaque trame et la valeur de l'entropie correspondant au bruit de fond ; (b) Segmentation initiale et finale du Signal "Mode un deux"	16
5	Signal avec bruit blanc gaussien	16
6	Algorithme utilisant le centroïde spectral et l'énergie à court terme : (a) Energie à court terme , centroïde spectrale et leur seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"	17
7	Algorithme utilisant l'entropie spectrale : (a) Entropie spectrale et son seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"	18
8	Notre algorithme de détection d'activité vocale : (a) Distance entre le vecteur de coefficient spectraux de chaque trame et le vecteur de coefficient de bruit de fond ; Distance entre la valeur de l'entropie spectrale de chaque trame et la valeur de l'entropie correspondant au bruit de fond ; (b) Segmentation initiale et finale du Signal "Mode un deux"	19
9	Détection d'activité vocale utilisant le centroïde spectrale et l'énergie à court terme : (a) Energie à court terme , centroïde spectrale et leur seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"	20
10	Algorithme utilisant l'entropie spectrale : (a) Entropie spectrale et son seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"	21
11	Notre algorithme de détection d'activité vocale : (a) Distance entre le vecteur de coefficient spectraux de chaque trame et le vecteur de coefficient de bruit de fond ; Distance entre la valeur de l'entropie spectrale de chaque trame et la valeur de l'entropie correspondant au bruit de fond ; (b) Segmentation initiale et finale du Signal "Mode un deux"	22

Liste des tableaux

1	tableau récapitulatif des segmentations de notre algorithme et ceux de références . . .	23
---	---	----

LISTE DES TABLEAUX

2 Tableau résumant les performances des algorithmes 25

Résumé

L'objectif de ce rapport est de confronter les performances de notre algorithme de Détection d'Activité de la Voix dans une trame audio (DAV ou Vocal Activity Detection en anglais). Le DAV est un algorithme qui vise à extraire d'une trame audio tous les tronçons où il y a présence de voix et améliore considérablement le processus de reconnaissance de la parole. En effet, en ne se concentrant que sur les tronçons de voix, on ne cherche pas à reconnaître des silences entre des mots mais que des mots, phonèmes etc. (cela dépend de la manière dont est réalisé l'algorithme de reconnaissance de la parole).

Actuellement, plusieurs méthodes sont utilisées pour réaliser cette opération de détection de flux de parole dans différents domaines (temporels et fréquentiels). Le domaine utilisé va bien sûr impacter sur la rapidité de l'algorithme. Le domaine temporel ne nécessitant pas de calcul complexe, il rendra les algorithmes plus rapide néanmoins il sera plus sensible au bruit. Le domaine fréquentiel est quant à lui plus robuste au bruit mais est plus gourmand en calcul. Les méthodes les plus connues dans le domaine temporel sont l'énergie à court terme combinée aux taux de passage à zéro (Short time Energy and Zero Crossing Rate) et dans le domaine fréquentiel, l'entropie (Spectral Entropy), le Spectral Centroid (qui est une sorte d'indicateur du "centre de masse" du spectre) mais aussi les Mel Feature Cepstrum Coefficients (MFCCs).

Notre parti pris a été de choisir de travailler dans le domaine fréquentiel essentiellement afin d'avoir un algorithme très robuste au bruit et de combiné l'entropie au MFCCs. De plus, même si mes deux méthodes sont très peu sensible au bruit, elles perdent un peu leur immunité à partir d'un certain SNR (Signal Noise Ratio ou Rapport de Signal à Bruit en français). De fait, afin de pouvoir travailler dans des environnements assujettis à des SNR très faible (la présence de bruit est considérable), nous avons donc généré une fonction d'initialisation qui récupère l'empreinte du bruit environnant et s'en sert de vecteurs modèles. Les vecteurs extraits des deux méthodes sont donc comparés à ces modèles de bruit par distance euclidienne ou par corrélation pour chaque fenêtre d'analyse. Les vecteurs modèles sont mis à jour à chaque nouvelle fenêtre pour suivre les variations du signal.

L'étude comparative de mon modèle se fera par confrontation à trois modèles et avec des contraintes de bruits.

1 Introduction

Ce rapport s'organise de la façon suivante. Tout d'abord quelques notions théoriques nécessaires à la bonne compréhension du sujet seront introduites. Un état de l'art non exhaustif de la détection de l'activité vocale sera présenté. Ensuite, les différents travaux réalisés seront décrits. Ces travaux reposent essentiellement sur les Mel Feature Cepstrum Coefficient (MFCC) et l'entropie spectrale, dans l'objectif de rejeter toutes les trames audio non utiles. Les performances du VAD des méthodes proposées seront exposées puis comparées à des méthodes de références. Ce sera alors l'occasion de soulever des difficultés liées à nos différentes méthodes mais aussi des améliorations qui pourraient y être éventuellement apportés.

2 Notions théoriques

2.1 Traitement à court-terme du signal vocal

On peut assez facilement constater que la forme d'onde d'un signal vocal met en évidence son caractère non stationnaire. Étant donné son caractère non stationnaire, une étude à long terme ou globale est généralement inefficace. L'hypothèse la plus utilisée dans le traitement de la parole est le fait que les propriétés du signal vocal changent lentement dans le temps [5]. Cette hypothèse conduit à un traitement à court terme. De plus, l'analyse à court terme permet de respecter le caractère temps réel que nous souhaitons donner à notre algorithme par la suite. On évite de générer de la latence.

2.2 Méthodes dans le domaine temporel

Les méthodes ci-dessous sont une liste non exhaustive des méthodes temporelles utilisées. Elles sont néanmoins celles le plus couramment utilisées. Les définitions de chaque méthode est extraite d'un article [8].

2.2.1 Énergie à court terme

L'énergie court-terme est un paramètre qui reflète les variations d'amplitude dans le signal vocal. Elle fut un de premiers paramètres utilisés dans la détection d'activité vocale. Elle permet de classer les trames voisées et non voisées. Elle est définie de la manière suivante :

$$E_n = \sum_{k=n-N+1}^n [x(k) \cdot w(n-k)]^2 \quad (1)$$

où $w(n-k)$ est la fenêtre d'analyse, n est l'échantillon sur lequel est centré la fenêtre d'analyse et N la taille de la fenêtre d'analyse.

2.2.2 Taux de passage par zéro (Zero Crossing Rate, ZRC)

Le ZRC compte le nombre de passage par zéro du signal. Les trames de voisées [1] ont un taux de passage à zéro faible alors l'inverse des trames non voisées qui ont un ZRC élevé. On peut définir le ZRC de la manière suivante :

$$Z_n = \sum_{m=-\infty}^{infy} |sgn[x(m)] - sgn[x(m-1)]| w(n-m) \quad (2)$$

où

$$sgn[x(n)] = \begin{cases} 1 & x(n) \geq 0 \\ -1 & x(n) < 0 \end{cases} \quad (3)$$

et

$$w(n) = \begin{cases} 1/(2N) & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

2.3 Domaine fréquentiel

2.3.1 Entropie Spectrale

Nous ne pouvons pas parler d'entropie spectrale sans parler auparavant de l'entropie introduite par Claude Shannon dans sa théorie de l'information. Cette dernière permet de mesurer la quantité d'information contenue dans un signal aléatoire. Elle est définie de la manière suivante :

$$H(x) = - \sum_k p(x_k) \cdot \log_2[P(x_k)] \quad (5)$$

où $x = x_{k0 \leq k \leq N-1}$ est une série temporelle, fréquentielle ou autre et où $P(k)$ est la probabilité d'un certain état x_k .

2.3.1.1 Travaux de Shen J, Hung J et Lee J [10]

Shen J., Hung J. et Lee J. ont été les premiers à utiliser l'entropie dans le cadre de la détection de la parole. Ils ont démontré que l'entropie d'un flux de parole diffère de celui d'un flux sans parole. En effet, leur étude a montré que la structure de la voix reflète d'une organisation que l'on ne retrouve pas dans la structure spectrale du silence.

2.3.1.2 L'entropie spectrale

L'entropie de Shannon mesurant la quantité d'information contenue dans un signal aléatoire, il est plus d'usage d'utiliser l'entropie spectrale : la structure harmonique d'un segment de voix n'apparaissant que dans son spectrogramme. L'entropie spectrale est définie tout d'abord par la transformée de Fourier à court terme (Short time Fourier Transform) :

$$S(k, l) = \sum_{n=1}^N h(n) s(n-l) \exp \frac{-j2\pi kn}{N} \text{ avec } 0 \leq k \leq K-1 \text{ et } K = N \quad (6)$$

$S(k, l)$ représente l'amplitude de la k^{ime} composante fréquentielle, pour la l^{ime} trame d'analyse. $s(n)$ est l'amplitude du signal au temps n . N , le nombre de points considérés pour la transformée de Fourier. $h(n)$ est la fenêtre d'analyse (généralement, une fenêtre de hamming).

Energie spectrale On définit l'énergie spectrale de la manière suivante :

$$S_{energy}(k, l) = |S(k, l)|^2 \text{ avec } 1 \leq k \leq \frac{N}{2} \quad (7)$$

P(k,l) La probabilité associée à chaque composante spectrale est obtenue en normalisant :

$$P(k, l) = \frac{S_{energy}(k, l)}{\sum_{i=1}^{\frac{N}{2}} S_{energy}(i, l)} \text{ avec } 1 \leq i \leq \frac{N}{2} \text{ et } \sum_i P(i, l) = 1 \text{ pour tout } l. \quad (8)$$

Entropie Spectrale L'entropie spectrale s'appuie sur l'entropie de Shannon :

$$H(l) = \sum_{i=1}^{\frac{N}{2}} P(i, l) \cdot \log_2 |P(i, l)| \quad (9)$$

Il est plus souvent d'usage de considérer l'opposé de l'entropie pour obtenir des profils analogues à ceux de l'intensité.

2.3.1.3 Limitation

Une des limitations de l'entropie spectrale exposé par Ouzounov [4] est que l'entropie d'une trame sans parole bruitée par un bruit blanc coloré peut-être équivalente à celle d'une trame avec parole mais bruitée.

2.3.2 Spectral Centroid

Le "centroïde spectral" C_i de la i^{ime} trame est défini comme le centre de gravité de son spectre.

$$C_i = \frac{\sum_{k=1}^N (k+1) X_i(k)}{\sum_{k=1}^N X_i(k)} \quad (10)$$

$X_i(k)$ avec $k = 1, \dots, N$ sont les coefficients de la i^{ime} courte trame de la transformée de Fourier discrète (DFT). N est la longueur de la trame. Le centroïde spectral est une mesure de la position spectrale où des valeurs élevées correspondent à des sons plus brillants.

Ces explications sont issues de l'article de Theodoros Giannakopoulos [7]. Son algorithme et son étude seront utilisés par la suite pour comparer les résultats de notre algorithme.

2.3.3 Mel Feature Cepstral Coefficients (MFCC)

Les explications qui suivent sont issues et pour la plupart inspirées du site *practicalcryptography.com* [2] pour leur clarté et leur compréhension.

Les Mel Features Cepstral Coefficients (MFCCs) sont des caractéristiques majoritairement utilisées en reconnaissance de parole automatique. Les MFCCs ont été introduites par Davis and Mermelstein dans les années 80, et ont été "l'état de l'art" depuis. Avant la mise en place des MFCC, Les coefficients de prédiction linéaire (LPC) et les coefficients cepstraux de prédiction linéaire (LPCC) constituaient la caractéristique principale de reconnaissance automatique de la parole (ASR), en particulier avec les classificateurs HMM (Hidden Markov Model) .

2.3.3.1 Etape pour déterminer les MFCCs

Pour déterminer les MFCCs, il faut commencer, comme tout au long de notre étude, par sectionner le signal en courte trame se recouvrant les unes aux autres (overlapping). Pour chaque trame, on calcule l'estimation du périodogramme du spectre de puissance (en d'autres termes, on regarde la distribution de notre périodogramme). Ensuite, on applique un banc de filtres Mel aux spectres de puissances et on somme de l'énergie dans chaque filtre. On prend le logarithme de toutes les énergies du banc de filtre. Puis la transformée en cosinus inverse (DCT). On ne garde que les 12 premiers coefficients en omettant le premier que l'on remplace par le Log de l'énergie.

3 Travaux réalisés

Cette partie va aborder du travail réalisé sur l'algorithme de détection vocale (VAD). Une grande partie est inspirée de méthodes et modèles existants. En ce qui concerne le modèle de bruit de fond avec mise à jour à chaque trame, il est extrait des travaux Hongzhi Wang et Yuchao Xu, Meijing Li [12]. J'ai donc transposé leur travaux à l'entropie afin de la rendre plus robuste au bruit. Nous avons parler d'une des limitations de l'entropie un peu plus haut que nous avons cherché à repousser par cette méthode.

3.1 Fonctionnement général de l'algorithme

Avant de rentrer dans des explications plus approfondis sur les méthodes utilisées ainsi que leurs outils, nous allons expliquer globalement le fonctionnement de notre algorithme.

Tout d'abord, une première fonction *recorder.m* [3] permet l'acquisition de donnée par la carte son de mon laptop et de sauvegarder des données dans des vecteurs (préalablement aloués). Deux acquisition sont réalisées :

- Acquisition du bruit environnant
- Acquisition d'une trame de voix durant 3s (le temps est ici arbitraire. D'autant plus, que par la suite nous chercherons à être en temps réel).

Une fois ces deux acquisitions réalisées, nous utilisons celle de bruit de fond afin de générer un vecteur $MFCC_{s_{noise}}$. Pour cela, nous "découpons" le signal acquis en plusieurs trames de 15ms avec un recouvrement de 10ms (donc un décalage de 5ms) afin de respecter l'hypothèse du caractère stationnaire exprimé un peu plus haut. Ces trames (ou fenêtres) sont ensuite envoyées à un algorithme : *short_time_Fourier_transform.m* pour réaliser une transformée de Fourier (l'appellation de la fonction est un peu faussée car nous sommes déjà dans un cas de court terme). Enfin, on extrait les Mel Feature Cepstral Coefficients du signal de bruit de fond (toutes les trames) que nous moyennons. Ce vecteur est donc la moyenne des coefficients cepstraux du bruit (en considérant que seule du bruit de fond est présent dans cette acquisition). On opère de manière similaire pour l'entropie, en calculant la valeur moyenne de l'entropie pour le signal de bruit de fond (on réalise la moyenne sur l'ensemble des trames).

Pour l'acquisition d'une trame de voix, on réalise aussi le "découpage" en trame avec chevauchement. Pour chaque trame, on calcule l'entropie et les coefficients cepstaux. On compare le vecteur de coefficients cepstraux et la valeur de l'entropie de chaque trame à respectivement, le vecteur de bruit de fond de la moyenne des coefficients cepstraux et à la valeur moyenne du bruit de fond de l'entropie. On utilise pour cela, respectivement, la corrélation et la distance euclidienne.

On applique ensuite des seuils pour chacune des deux variables évoluant au cours du temps afin d'être le plus pertinent possible. Chaque trame qui est au dessus du seuil est labellisée d'un "un" sinon elle l'est d'un "zéro". Le vecteur contenant les labels est ensuite envoyé à une fonction permettant de corriger les valeurs aberrantes (par exemple un "un" perdu au milieu de "zéro" et vis-versa. L'algorithme corrige jusqu'à plusieurs paquets de valeurs aberrantes en fonction des paramètres qu'on lui rentre en argument). Pour cela, on s'appuie du poids des labels au nombres paires entourant chaque label considéré.

Une fois l'opération de correction de label réalisée, nous n'avons plus qu'à extraire les segments (en ms et non en trame comme nous avons précédemment) en considérant nos labels de "un" comme de la voix et les autres comme du silence, du bruit et des parasites sur le signal. Enfin, un dernier algorithme trie les segments récupérés. En effet, dans la littérature, chaque phonème dure un certain temps (selon la langue, le plus petit dure un 20ms et le plus long, souvent voisé, dure 100ms). Dans notre cas, on suppose que notre algorithme reconnaît un ensemble de phonème, soit une syllabe. Dans ce cas, on considère donc qu'en dessous de 90ms, on rejette le segment considéré. (LITTÉRATURE -REF ?)

Finalité Nous obtenons donc de l'acquisition d'un signal de trois secondes (choix arbitraires), un ensemble de segments contenant de la parole dans la majeure partie des cas (en effet, des bruits parasites peuvent-être présents). Nous concluons sur les manières d'améliorer notre algorithme notamment sur ses performances comparées à ceux de la littérature (on travaille avec des transformées de Fourier sur 257points¹ pour chaque trame de 15ms. La complexité des calculs est donc élevée). Nous allons développer par la suite les méthodes utilisées pour les différentes phases de l'algorithme de Détection d'activité vocale (on rappelle qu'il est composé d'un ensemble de sous-algorithmes).

3.2 La distance ou corrélation entre des vecteurs de bruit de fond et celui de chaque trame

L'article de Hongzhi Wang et Yuchao Xu, Meijing Li [12] propose d'extraire les Mel Features Cepstral Coefficients (MFCCs) d'un signal vocal pour chaque trame en considérant les dix (10) premières trames comme des trames de bruit de fond. En réalisant la moyenne de ces vecteurs, on obtient un vecteur de $MFCC_{bruit}$. Ce vecteur va permettre de réaliser la Corrélation des MFCC de chaque nouvelle trame avec lui même (dans l'article, ils utilisent aussi la distance euclidienne, qui se révèle être moins performante). Le vecteur de $MFCC_{bruit}$ est mis à jour à chaque trame :

$$\underline{cno} = \underline{c}.p + (1 - p).\underline{c} \quad (11)$$

où \underline{cno} est le vecteur de $MFCC_{bruit}$, \underline{c} sont les coefficients de chaque nouvelle trame et p est proche de 1. Le fait que p doit-être proche de 1 permet de garantir que le vecteur de MFCC bruité reste constitué de bruit de fond principalement malgré l'introduction de coefficients de parole à chaque nouvelle trame (on minimise cette introduction).

Nous nous sommes donc appuyés de ces travaux pour construire notre algorithme de détection vocale. Nous avons aussi appliqué le principe de distance euclidienne à l'entropie spectrale, toujours dans le souci de rendre notre algorithme le plus robuste au bruit.

3.3 Détermination des seuils

La détermination des seuils a été la partie la plus difficile à mettre en oeuvre, non pas que les algorithmes sont complexes, mais qu'une mauvaise méthode entraîne la suppression des trames voulues ou des ajouts. Ma première idée fut d'utiliser la distribution des valeurs obtenues. Cette distribution étant obtenue après avoir supprimé toutes les valeurs au-dessus de la moyenne du signal. Le but étant de récupérer la plus petites variations possibles du signal après analyse (Entropie et/ou MFCCs). Néanmoins, cette méthode bien qu'efficace était constante au court du temps ou du moins sur l'analyse

1. Il s'agit de la puissance de 2 la plus proche de la longueur de nos fenêtres de travail en échantillons

d'un buffer d'échantillon. Il y avait donc des pertes et l'introduction de bruit après seuillage. La deuxième idée fut de trouver une méthode adaptée au chacun des paramètres utilisés.

3.3.1 Seuil pour les MFCCs

Concernant les MFCCs, je me suis appuyé d'une fonction sigmoïde mise à jour à chaque fenêtre (après une lecture sur quelques articles sur les réseaux de neurones). La sigmoïde a été améliorée afin d'avoir pour $x = 0$, une valeur proche de la valeur maximale du bruit (contrairement à la sigmoïde habituelle qui a pour valeur en $x = 0$, 0,5). Cette valeur maximale du bruit est généralement proche de zéro de part la méthode utilisée pour déterminer la distance entre des trames de bruit de fond et la trame en cours (la distance d'un vecteur de bruit de fond à un autre vecteur de bruit donne généralement un résultat faible).

3.3.2 Seuil pour l'entropie spectrale

La fonction de seuillage concernant l'entropie spectrale est quant à elle très standard du fait que les trames de parole sont très différentes de celle où il y a des silences ou du bruit. Pour cela, nous utilisons les 20 premières trames comme référence pour réaliser la valeur du seuil à l'instant 0. On considère la moyenne de ces 20 trames et on y ajoutant trois (3) fois l'écart type de ces 20 trames :

$$th_{noise} = \frac{1}{N} \sum_{k=1}^N x_k + 3 \cdot \sqrt{\frac{1}{N-1} \sum_{i=1}^N |x_i - \mu|} \quad (12)$$

où μ représente la moyenne du signal considéré. En effet, on considère que sur les 20 premières trames ne représentent que bruit. En y ajoutant trois fois l'écart-type, on s'assure d'être au-dessus du bruit de fond dans sa majeure partie.

Pour avoir un seuil pertinent et précis, on réalise une mise à jour ce seuil. La mise à jour est effective dès que l'on a 10 trames à traiter. En effet, nous n'avons pas de pertinence à utiliser la moyenne et l'écart type sur une trame. Dix trames nous semblaient convenable pour pouvoir exploiter ces deux indices pour une mise à jour correcte de notre seuil. Les étapes de la mise à jour sont les suivantes :

- On collecte dix nouvelles trames
- On réalise l'opération suivante :

$$th_{new} = \frac{1}{N} \sum_{k=1}^N x_k + 3 \cdot \sqrt{\frac{1}{N-1} \sum_{i=1}^N |x_i - \mu|} \quad (13)$$

- On met à jour la valeur du seuil pour les dix prochaines trames de la manière suivante (la mise à jour est similaire à celle du vecteur bruit pour l'entropie et les MFCCs) :

$$th = p \cdot th_{noise} + (1 - p) th_{new} \quad (14)$$

Comme durant la mise à jour du vecteur bruit, on cherche à avoir une valeur de p proche de 1 (un) afin que la mise à jour ne soit pas polluée par une trame de voix. Cela rendrait notre seuil inutile car il ne considérerait plus les trames de voix ou alors celle avec des intensités plus élevées que celle considérée.

Amélioration possible du seuil On pourrait, tout comme le seuil des MFCCs, considérer une fonction sigmoïde particulière pour calculer le seuil optimal à chaque trame. Pour le moment, celle-là n'a pas été développée car les résultats étaient satisfaisants.

3.3.3 Segmentation et correction des labels

Nous en avons un peu parlé plus haut, une fois nos seuils identifiés, il faut labéliser chaque trame. Chaque label est soit un "1" soit un "0" ("0" correspondant à une trame sous le seuil). On réalise cette opération pour les deux paramètres que nous utilisons. Nous combinons nos résultats obtenus pour labéliser chaque trame (nous avons donc un "et" logique entre les deux vecteurs de labels obtenus).

3.3.3.1 Correction des "uns" et "zéros" isolés

Notre vecteur de labels récupéré, ils arrivent que des "zéros" et des "uns" se retrouvent isolés dans une combinaison, respectivement, de "uns" et des "zéros". Afin de corriger ces "erreurs", on va chercher à évaluer le poids des labels entourant le label considéré. En soit, on balaye tous les labels afin de tous les vérifier. Les poids des labels entourant chaque label doit-être paire pour éviter des générer des nouvelles erreurs ou déplacer des labels (on a donc un nombre impaire des labels à évaluer pour trancher sur la valeur du label que l'on considère). On évalue donc le nombre de "un" et de "zéro", ce qui détermine si notre label est un "un" ou un "zéro". Mes propos ci-dessus n'étant pas forcément claire, illustrons mes propos avec un exemple :

1. Considérons un vecteur de labels : [1 1 0 0 1 1 1] et poids de 3 labels de part et d'autres du label que l'on évalue. On cherche à savoir si les "zéros" de ce vecteur de "uns" ne sont pas des erreurs. On se positionne sur le premier "zéro", on prend 3 labels à gauche et à droite (les premières valeurs n'ayant pas ou peu de labels sur leur gauche, on ne considère que le poids à d'autres en y intégrant au fur et à mesure les labels à gauche). On évalue donc le vecteur : [1 1 0 0 1 1]. On constate qu'il y a 4 "un" pour 2 "zéro". Le label "zéro" considéré est en faite un "un" (car il y a plus de poids de "un" que de "zéro"). Finalement, en évaluant l'ensemble du vecteur de label, on a : [1 1 1 1 1 1 1]. Notre vecteur est donc corrigé.

Aucune littérature n'a été utilisée pour traiter de ce problème, simplement des constats sur un grand nombre d'exemples. Le résultat obtenu est concluant, même si des améliorations sont tout à fait envisageables. Notamment sur le poids à considérer qui pourrait ne pas être fixe par exemple. On pourrait considérer que le poids devient plus grand quand, pendant un lapse de labels, ces derniers sont inchangeants. De fait, on pourrait corriger des "zéros" ou "uns" isolés sur un plus grand poids.

3.3.3.2 Segmentation

² Une fois le vecteur de labels corrigé, il nous reste plus qu'à extraire les segments où il y a de l'activité vocale. Pour cela, on considère un "head" et "tail". Le "head" caractérise le premier label "un" qui précède un "zéro" du vecteur et le "tail", le dernier "un" qui précède un groupe de "un" et est suivi d'un "zéro". Il peut y avoir plusieurs "head" et "tail" sur le signal acquisitionné. Le "head" est le premier marqueur, le "tail" lui n'apparaît dès lors qu'un "head" est marqué sur le vecteur de labels. On extrait pour chaque "head" et "tail", leurs limites (la valeur de leur échantillon). Par exemple :

2. L'idée s'appuie sur le principe des bufferisation que l'on utilise couramment en temps réel pour stocker nos données dont l'on réalise l'acquisition. Notamment en ce qui concerne les appellations "head" et "tail"

1. On considère un vecteur de label : $[0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0\ 0]$ (**nb** : la taille du vecteur est plus conséquente dans l'algorithme). On balaye toutes les valeurs du vecteur jusqu'au premier "head" (un "un" qui est précédé d'un "zéro"). On place donc le "head" sur le "un" qui représente le cinquième échantillon de notre vecteur ($[0\ 0\ 0\ 0\ \color{red}{1}\ 1\ 1\ 1\ 0\ 0\ 0]$, ici en rouge). Puis on continue l'analyse mais cette fois-ci pour trouver le "tail". Ici le "tail" est le "un" qui représente le huitième échantillon du vecteur ($[0\ 0\ 0\ 0\ 1\ 1\ 1\ \color{blue}{1}\ 0\ 0\ 0]$, ici en bleu). Le "head" et le "tail" délimite donc le segment suivant : $[0\ 0\ 0\ 0\ \color{red}{1}\ \color{red}{1}\ \color{red}{1}\ \color{red}{1}\ 0\ 0\ 0]$.

Ensuite, le "head" et le "tail" renvoient leurs limites (à savoir la valeur de leur échantillon). On obtient donc :

$$\begin{bmatrix} Limits(head) \\ Limits(tail) \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}$$

Pour extraire notre vecteur, il nous reste plus qu'à réaliser la conversion : Trame \Leftrightarrow Samples. Pour cela, on utilise la formule suivante pour calculer l'échantillon du "head" :

$$Sample = Limits(head).step_{ms} \quad (15)$$

où $step_{ms}$ représente la durée en ms de la fenêtre d'étude en ms moins le recouvrement que l'on considère en ms . Dans notre cas, elle est de $5ms$. Pour calculer l'échantillon du "tail", l'opération est similaire, mais il faut veiller à rajouter la longueur en ms de la fenêtre d'étude :

$$Sample = Limits(head).step_{ms} + window_{ms} \quad (16)$$

où $window_{ms}$ est la fenêtre d'étude considérée.

Les échantillons des "heads" et "tails" sont renvoyés dans un vecteur que l'on concatène³ avec ceux déjà retournés. On obtient finalement une matrice. On n'a plus qu'à extraire les segments en s'appuyant de la matrice précédemment trouvée pour l'ensemble du signal.

3.3.3.3 Post traitement des segments

Nous avons déjà commencé à développer l'idée un peu plus haut, nous allons continuer ici. Ce qu'il faut retenir c'est que malgré notre correction des labels un peu plus haut, des erreurs subsistent. Vous nous direz : "Mais alors quel fut l'avantage de corriger les labels un peu plus haut?". Tout simplement, la fonction de correction des labels permet de corriger des labels isolés. Néanmoins, elle ne prend pas en compte le fait que les segments doivent-être d'une certaine taille pour pouvoir être assimilé à une activité vocale. En effet, la durée est quelque chose que nous n'avions pas évoqué, mais c'est un paramètre important à prendre en compte. Un phonème peut durer entre $20ms$ et $100ms$, une syllabe $100ms$ à minima. Aucune contrainte n'a été faite sur la longueur (en sample) de nos segments. Il se peut que ceux-ci ne durent qu'une centaine d'échantillons (*i.e* moins de $5ms$ à $16\ 000Hz$). Il faut donc les rejeter car ne caractérise par une activité vocale. Pour cela, nous avons choisi de rejeter tous les segments représentant moins de $50ms$ en estimant qu'on ne peut avoir une activité vocale dans ce laps de temps. Les autres segments sont quant à eux conservés.

3. Mots anglais : enchainement des vecteurs en soit



3.4 Conclusion

Après avoir discuté des nos choix méthodiques et les algorithmes mis en oeuvre, nous allons maintenant discuter des performances de notre algorithme de détection d'activité vocale en le comparant à des algorithmes.

4 Tests et comparaisons

Dans cette partie, nous allons comparer notre algorithme à ceux de la littérature. La plupart des algorithmes ont été récupérés sur la plate-forme collaborative de Matlab en se basant sur leur cotation mais aussi le respect de la littérature.

4.1 Présentation des algorithmes évalués

Nous allons évaluer différents algorithmes à savoir :

- Un algorithme basé sur la détection de l'énergie et du taux de passage par zéro. Les seuils sont fixés de manière manuelle dans une plage prédéfinie [11]. (à venir ...)
- Un algorithme basé sur la détection de l'énergie ainsi que le centroïde spectrale ("centre de gravité").[6] Les seuils sont définis par détection des maxima locaux de leur histogramme. Pour des explications un peu plus précise, nous vous invitons à lire son article[7].
- Un algorithme basé sur la détection de l'entropie spectrale (dont nous avons parlé un peu plus haut). Le seuil est défini par une formule liant la moyenne à l'écart-type. Cet algorithme a été réalisé par mes soins en s'inspirant des plusieurs articles. [9][10]. Cet algorithme utilise des fonctions communes à notre algorithme.

Plusieurs tests seront opérés :

- Performances : Rapidité de la détection de l'algorithme.
- Robustesse au bruit : nous chercherons à ajouter progressivement du bruit dans nos tests en baissant le SNR (Signal Noise Ratio ⁴).

Nous savons que nous partons dès le départ avec un désavantage. En effet, nous utilisons que des méthodes fréquentielles qui ont une complexité de calcul élevé. De plus, notre algorithme n'est pas optimisé car nous répétons certaines opérations plusieurs fois, notamment la plus gourmande, la Discrete Fourier Transform ⁵.

4.2 Robustesse au bruit

Notre première tâche avant de parler de performances en terme de temps d'exécution est de tester leur robustesse face aux bruits. En d'autres termes, sachant que nous allons travailler dans des environnements différents, il faut veiller à ce que l'environnement de travail n'empiète pas sur la reconnaissance vocale. Pour cela, nous allons réaliser plusieurs test sur un fichier audio contenant de la voix. On veillera dans un premier à ce qu'il est pas ou peu de bruit de fond. Puis nous en rajouterons progressivement. Enfin, le test ultime sera dans un environnement musical, nous mettrons en fond, une trame musicale (disons d'un style "métal" ⁶). Le but de chaque algorithme est sur cinq tests ⁷ d'avoir le plus de trame avec présence vocale et le moins avec du bruit environnant.

4. Taux du rapport signal à bruit

5. Transformée de Fourier Discrète

6. Le style métal ne représente par forcément le style englobant du morceau que nous choisirons. Néanmoins, nous vous donnons le style métal pour que vous puissiez vous représenter le bruit généré par ce style de musique

7. Le nombre de tests est pris aléatoirement et de manière impaire

4.2.1 Données des tests

Les données suivantes seront utilisées pour tous les test. En cas de changement, nous les rappellerons :

1. Fréquence d'échantillonnage : 16 kHz
2. Quantification du signal : 16 bits
3. Taille des fenêtres : 15 ms
4. Décalage entre les fenêtres : 5 ms

4.2.2 Signal avec présence de voix et peu ou pas de bruit (SNR élevé)

Pour commencer, nous allons illustrer le signal que nous allons utiliser. Il est assez simpliste, dure environ 5 s et dans un premier temps, le SNR est très élevé. La figure ci-dessous (FIGURE 1) est la représentation temporelle du signal avec des marqueurs indiquant la présence des mots "mode", "un" et "deux". À titre d'information, le signal est échantillonné à 16 kHz et quantifié sur 16 bits .

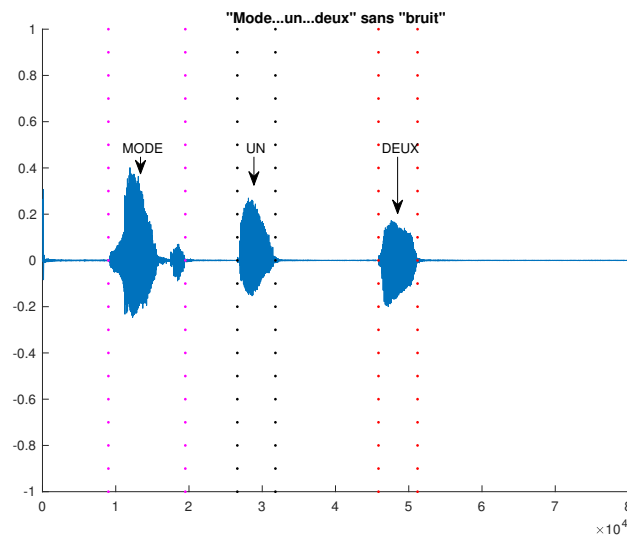


FIGURE 1 – Représentation temporelle du signal utilisé

Nous allons maintenant tester ce signal avec l'algorithme que nous proposons et ceux que nous avons soumis à titre comparatif.



4.2.2.1 Algorithme utilisant le Spectral centroid et le STE(Short time Energy)

Nous allons exposer les résultats de la segmentation [FIGURE 2 - b] et les courbes de l'entropie et son seuil de détection [FIGURE 2 - a] ci-contre. Les figures de l'algorithme n'étant pas forcément concrète sur le découpage (sauf si l'on sait que la partie grisée est la partie rejetée et les parties bordeaux sont celles conservées), nous avons utilisé les données retournées par l'algorithme pour vous présenter une figure similaire à FIGURE 1.

Etant donné que le découpage des segments n'était pas le même que celui présenté, nous avons labellisé les segments pour plus de compréhension. Comme vous pouvez le constater, le mot "mode" a été décomposé en syllabes. Cette décomposition est sans doute dû au fenêtrage et au décalage très courts

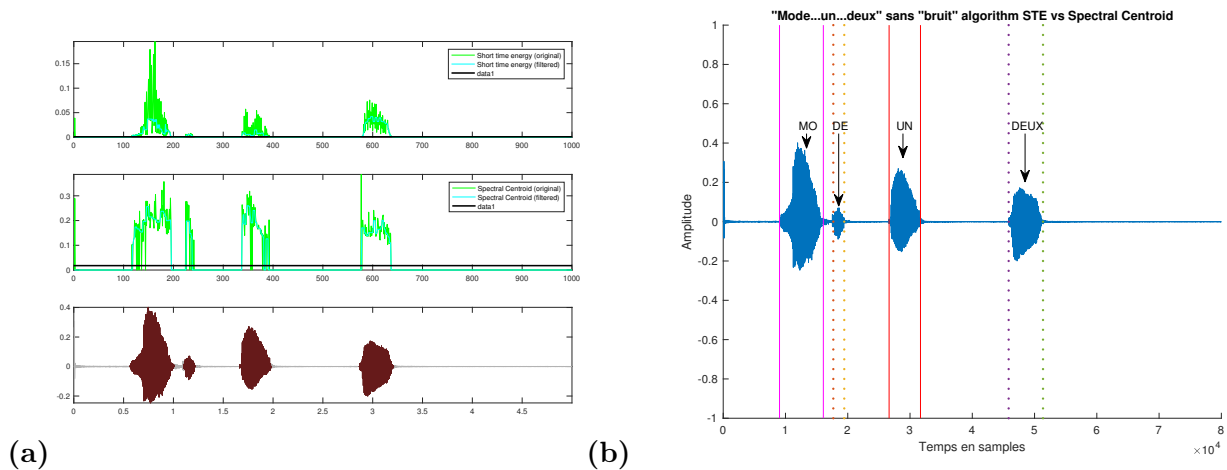


FIGURE 2 – Détection d'activité vocale utilisant le centroïde spectral et l'énergie à court terme : (a) Energie à court terme , centroïde spectrale et leur seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"

4.2.2.2 Algorithme utilisant l'entropie spectrale

En utilisant l'entropie spectrale avec un seuil définit ainsi :

$$threshold = \frac{1}{N} \sum_{k=1}^N x_k + 3 \cdot \sqrt{\frac{1}{N-1} \sum_{i=1}^N |x_i - \mu|} \quad (17)$$

où N désigne le nombre d'éléments considérés (dans notre cas, les 10 premières trames supposés être du bruit).

On peut aisément constater [FIGURE 3 - a] qu'il n'y a aucune erreur de segmentation même si les segments sont plus larges que ceux l'algorithme précédent. Nous observons aussi que l'entropie spectrale dessine parfaitement l'activité vocale de manière distincte. [FIGURE 3 - b]

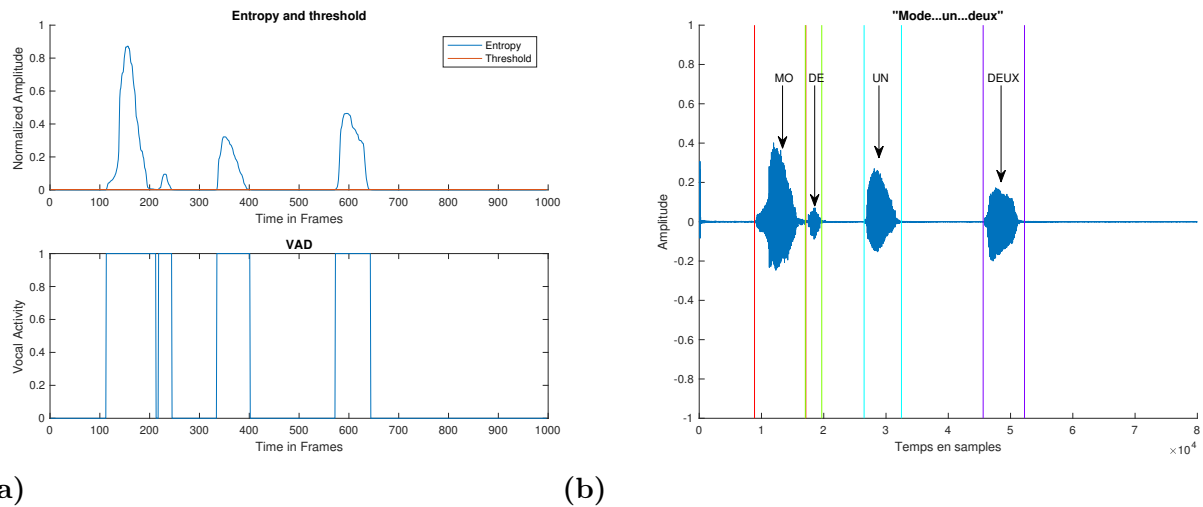


FIGURE 3 – Détection d'activité vocale utilisant l'entropie spectrale : (a) Entropie spectrale et son seuil de détection du Signal vocal ; (b) Segmentation finale du Signal "Mode un deux"

4.2.2.3 Notre algorithme

Nous avons évoqué un peu plus haut, le fait que nous utilisons un signal dit de bruit de fond. Dans notre cas, nous allons utiliser la fin du signal (où il n'y a aucune présence vocale à des fins de vecteur de bruit). Dans les conditions normales d'utilisation, ce dernier aurait été directement enregistré en amont du traitement. Notre signal servant de référence est réalisé de manière à avoir 1 seconde d'acquisition (on peut bien sûr le réduire si besoin est). Les courbes suivantes sont les résultats de nos différents paramètres utilisés. Les paramètres sont accompagnés de leur seuil [FIGURE 4 - a]. La dernière courbe symbolise notre vecteur de labels.

Comme précédemment, nous allons symboliser les différents morceaux comme nous l'avons fait tout au long du rapport afin d'en simplifier la lecture. Néanmoins, vous pouvez constater une détection de segments est très fiable [FIGURE 4 - b]. Nos résultats sont concordants avec ceux précédents [FIGURE 1], [FIGURE 3 - a], [FIGURE 2 - b]. Dans un premier lieu, on pourrait croire que notre algorithme, bien plus complexe en terme de calculs, n'apporte rien à la littérature existante sur le sujet, si ce n'est d'être plus lent. Cependant, nous allons par la suite constater qu'il se révèle plus robuste au bruit que ceux que nous présentons.

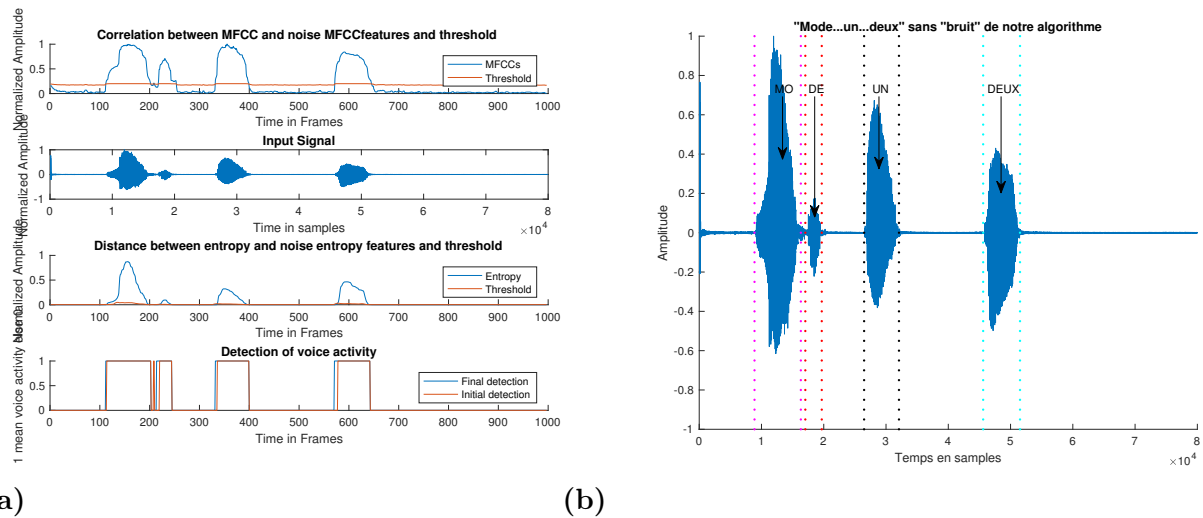


FIGURE 4 – Notre algorithme de détection d'activité vocale : (a) Distance entre le vecteur de coefficient spectraux de chaque trame et le vecteur de coefficient de bruit de fond ; Distance entre la valeur de l'entropie spectrale de chaque trame et la valeur de l'entropie correspondant au bruit de fond ; (b) Segmentation initiale et finale du Signal "Mode un deux"

4.2.3 Signal avec un SNR de $-30dB$

Nous remarquons que dans des conditions optimales, l'ensemble des algorithmes est capable de détecter les présences vocales du signal. Nous allons donc rajouter du bruit pour voir comment les algorithmes se comportent avec différents SNR⁸. Tous les SNR sont calculés à l'aide de la fonction Matlab *snr.m*. Le bruit est un bruit blanc gaussien généré à l'aide de la fonction *wgn.m* de Matlab. La FIGURE 1 est représenté dans un environnement bruyant symbolisant celui que nous utiliserons pour la suite de notre étude de cas [FIGURE 5].

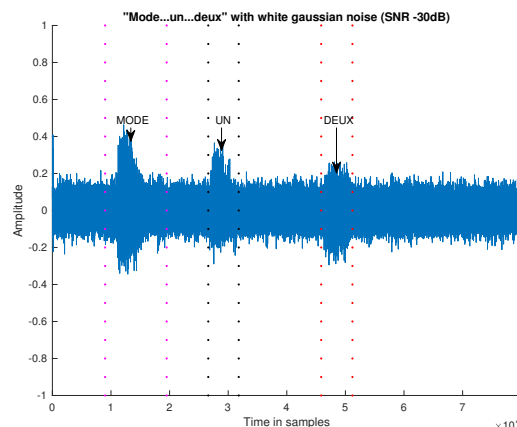


FIGURE 5 – Signal avec bruit blanc gaussien

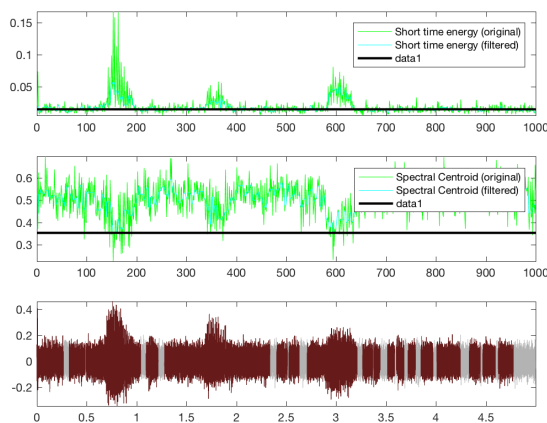
8. Signal Noise Ratio : Rapport de Signal à Bruit



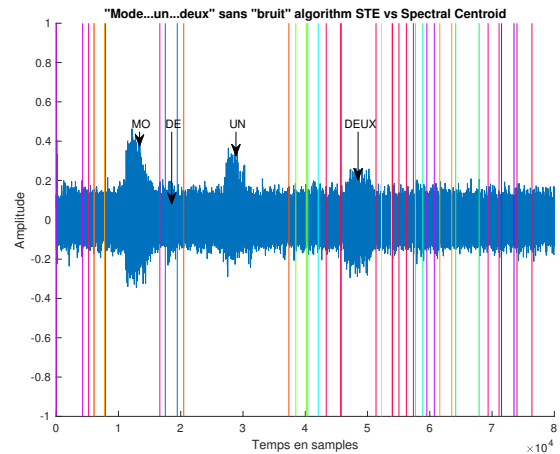
4.2.3.1 Algorithme utilisant le Spectral centroid et le STE(Short time Energy)

Nous pouvons nous rendre compte que la segmentation du signal (signal bien évidemment bruité) est plus difficile [FIGURE 6 - b]. Plus difficile non pas dans le sens que l'algorithme a du mal à détecter mais qu'il détecte le bruit environnant en plus des segments de voix.

Il est vrai qu'il y a beaucoup de couleurs différentes pour représenter les différents segments détectés [FIGURE 6 - b], rendant l'exploitation de la figure compliquée. Afin d'y voir un peu plus clair, la [FIGURE 6 - a] en plus d'afficher les courbes de l'énergie et du centroïde, trace aussi la segmentation obtenue [courbe 3]. Les parties grisées représentent les segments rejetés et celles bordeaux, les segments conservés (présence de voix).



(a)



(b)

FIGURE 6 – Algorithme utilisant le centroïde spectral et l'énergie à court terme : (a) Energie à court terme , centroïde spectrale et leur seuil de détection; (b) Segmentation finale du Signal "Mode un deux"

4.2.3.2 Algorithme utilisant l'entropie spectrale

Le résultat diffère légèrement par rapport à la trame audio sans bruit [FIGURE 3]. En effet, les segments sont moins étendus [FIGURE 7 - a] mais aussi, on voit que le phonème "de" s'est légèrement effacé (il est très étroit. Un de nos algorithmes de correction de segments l'a supprimé mais il devrait être présent. Cela montre aussi l'équilibre à avoir dans nos algorithmes. Une correction sera sans doute à amener pour avoir des résultats plus précis). On peut souligner que l'entropie spectrale est assez efficace dans un environnement bruyant/bruité [FIGURE 7 - b]. Les segments de voix sont dessinés franchement même s'il on constate un léger offset comparé à l'entropie spectrale dans un environnement non bruyant [FIGURE 7 - a].

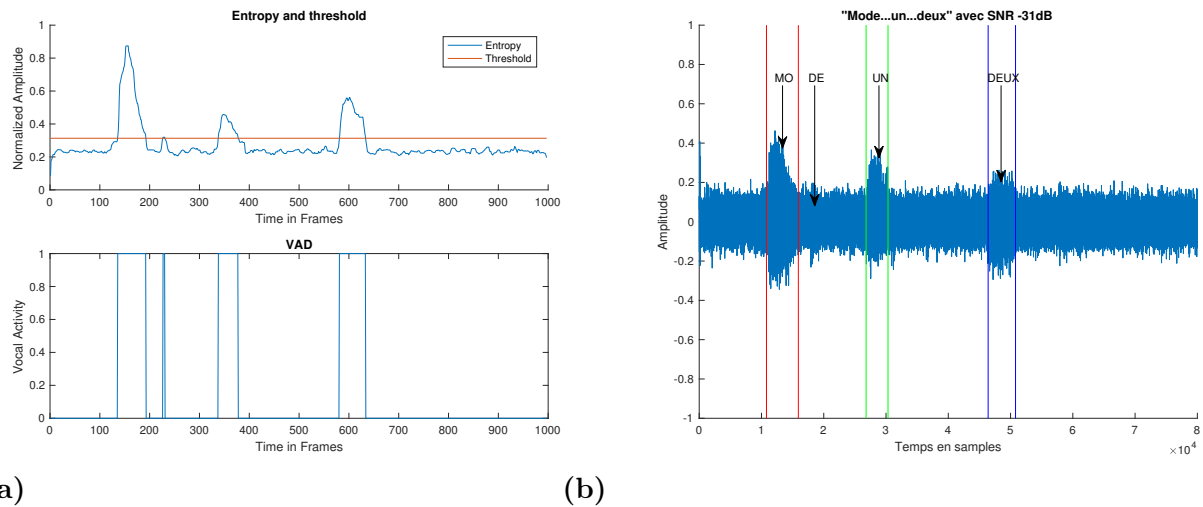


FIGURE 7 – Algorithme utilisant l'entropie spectrale : (a) Entropie spectrale et son seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"

4.2.3.3 Retour à notre algorithme

Le bruit de fond est cette fois-ci un vecteur de bruit blanc gaussien d'une durée d'une seconde. Bien-sûr, il n'est en aucun cas un des tronçons de celui utilisé en addition à notre morceau.

Vous pouvez constater que nous n'avons aucun segment introduit par du bruit [FIGURE 8 - b], néanmoins, la syllabe "de" du mot "mode" n'a pas été détectée. La question est pourquoi ? Pour cela, nous allons analyser les courbes de détection [FIGURE 8 - a]. On remarque sur la courbe "Distance between entropy and noise entropy features and threshold", de fait que le seuil soit fixe, la syllabe "de" n'a pas été détectée. C'est sans doute une amélioration que nous pourrions apporter par la suite à notre algorithme. De plus, sur la courbe "correlation between MFCC and noise MFCCs features", notre algorithme calculant la distance entre le vecteur bruit et le vecteur de MFCC que l'on calcule à chaque instant est un peu instable au début, malgré que l'on note la présence de "pics". Pour conclure sur cette simulation, nous pouvons déjà argumenter que notre algorithme est robuste au bruit même si on engendre une perte de syllabe dite "non plosives". Quelques améliorations et modifications pourraient assurer la détection de cette syllabe néanmoins. On reste satisfait de notre détection. En effet, un algorithme de détection d'activité vocale ne travaillant jamais seul, il rendra les algorithmes de reconnaissance de la parole plus performants. Le gain de temps à ne pas traiter du "bruit" est donc presque nul pour notre algorithme comparé à celui utilisant l'énergie à court terme.

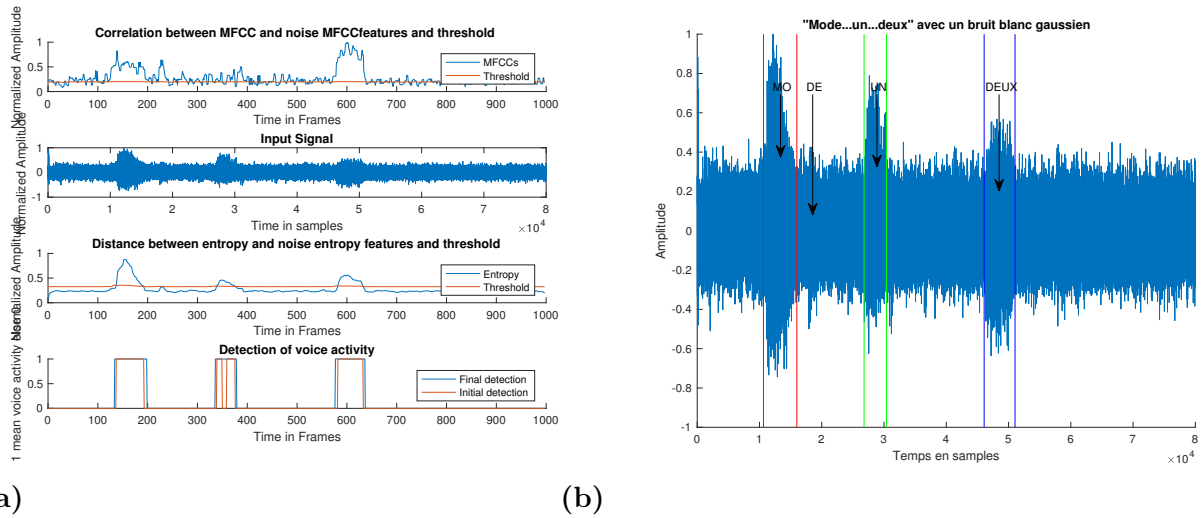


FIGURE 8 – Notre algorithme de détection d'activité vocale : (a) Distance entre le vecteur de coefficient spectraux de chaque trame et le vecteur de coefficient de bruit de fond ; Distance entre la valeur de l'entropie spectrale de chaque trame et la valeur de l'entropie correspondant au bruit de fond ; (b) Segmentation initiale et finale du Signal "Mode un deux"

4.2.4 Signal avec un musique de fond - SNR -13dB

Nous aimons un peu la difficulté, alors nous allons utiliser, comme nous l'avons énoncé dans l'introduction un peu plus, une musique de style "métal"⁹. La musique utilisée est **Rammvier** de Rammstein¹⁰. Le choix de cette musique n'est pas arbitraire et pas seulement par goût. Le spectre de cette musique est assez densément rempli dans la bande passante que nous étudions. De plus, la musique est bien sûr remplie de parole (nous avons donc la présence d'une voix parasite). Le but de cette dernière expérience est donc de vérifier que notre algorithme est robuste contre le bruit musical et les voix dans des trames musicales (dans le cas général, on ne peut empêcher la détection de la voix si elle est bien plus présente que celui du locuteur).

La musique prélevée n'est néanmoins pas à une fréquence d'échantillonnage de 16 kHz . Il faut donc sous-échantillonner les morceaux choisis. Pour cela, afin d'éviter tout repliement du signal, on commence par appliqué un passe-bas à 8 kHz (théorème de Shannon). Puis, une fois cette opération réalisée, nous pouvons sous-échantillonner nos morceaux et les ajouter à notre "morceau".

4.2.4.1 Algorithme utilisant le Spectral centroid et le STE(Short time Energy)

De part la confusion générée par la segmentation de l'algorithme [FIGURE 9 - b], nous allons aussi nous appuyer des courbes permettant la détection des segments ainsi que des parties grisées et bordeaux traduisant notre segmentation [FIGURE 9 - a].

9. La définition de style différant entre les individus, la musique n'est peut-être pas représentative du style "métal" mais d'un style en déclinant.

10. Pour nos tests nous avons simplement utilisé un échantillon de 5s et une autre seconde pour notre vecteur bruit.

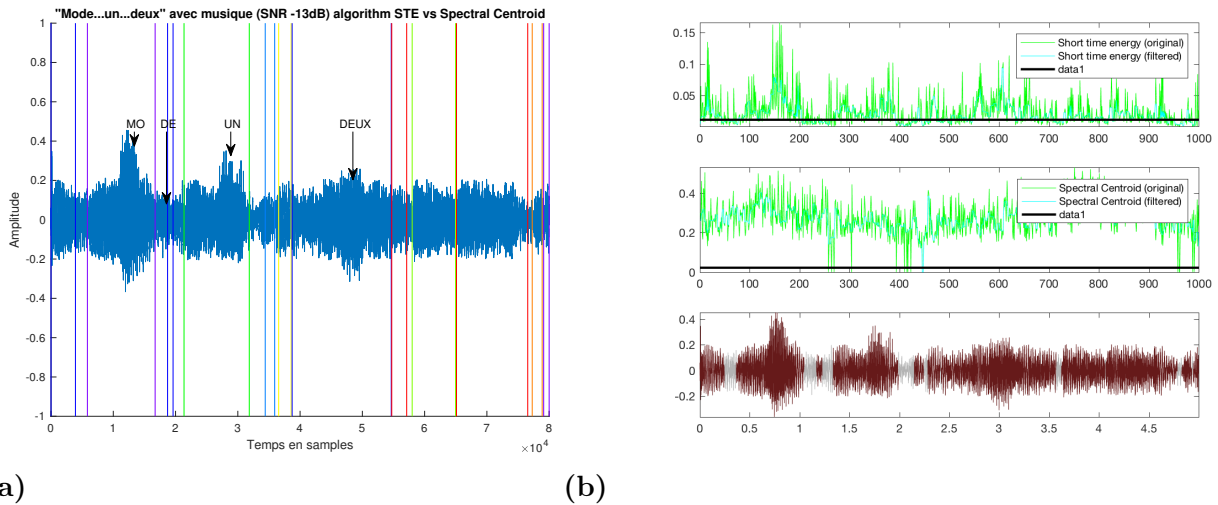


FIGURE 9 – Détection d'activité vocale utilisant le centroïde spectral et l'énergie à court terme : (a) Énergie à court terme, centroïde spectral et leur seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"

On remarque, comme précédemment [FIGURE 6 - b], que dès qu'il y a un SNR (taux de signal à bruit) trop faible, l'algorithme renvoie beaucoup de segments où il n'y a que de "bruit" et omet certains segments de voix, notamment "de" (du mot "mode"). On peut donc conclure que l'algorithme est peu robuste au bruit.

4.2.4.2 Algorithme utilisant l'entropie spectrale

Nous l'avons souligné un peu plus haut, nous utilisons des algorithmes propres à notre algorithme pour afficher la segmentation (notamment sont découpage). De fait les petits segments sont rejetés (du moins ceux qui ne respectent pas la condition sur leur longueur). Le résultat expose la détection d'un seul segment [FIGURE 10 - b]. Néanmoins, on peut constater sur la courbe de l'entropie spectrale et son seuil de détection [FIGURE 10 - a] détecte d'autres segments notamment le mot "deux", pas entièrement mais partiellement tout comme un bout du mot "un" (bien que ce soit un phonème de fin). De plus, en analysant les résultats que nous proposons [FIGURE 10 - a ; Courbe 1], l'entropie spectrale est assez perturbée par un environnement musical comparé à un environnement bruyant [FIGURE 7 - a ; Courbe 1] où on avait seulement un offset. Ici, des dents de scies font leur apparition, signe que l'entropie ne devient plus assez robuste et précise. Le seuil utilisé traditionnellement expose ses limites ici. En effet, le bruit de fond étant élevé et c'est avec ce bruit de fond qu'il définit son niveau. Des segments sont donc omis de fait.

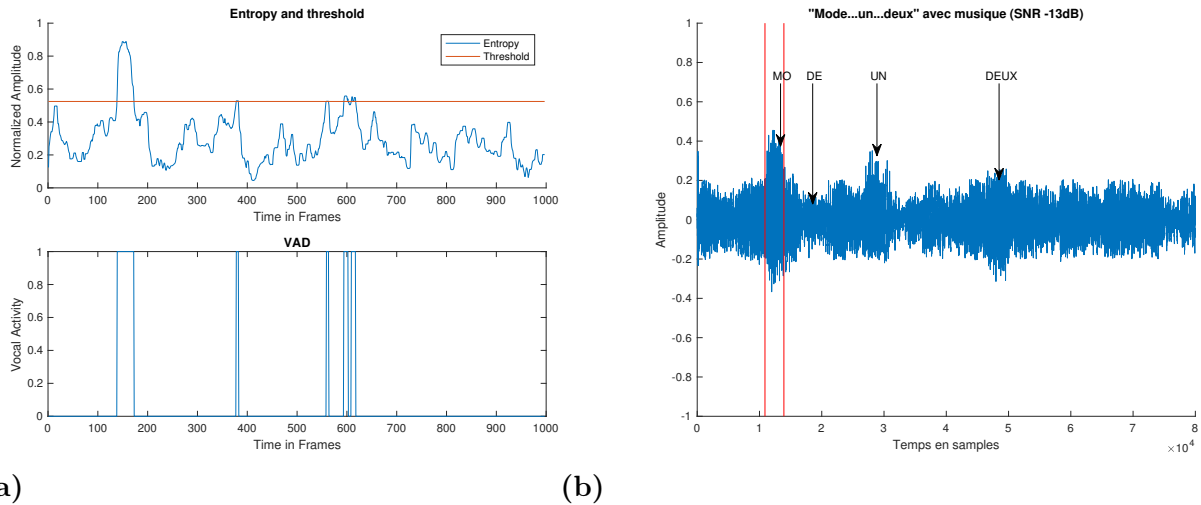
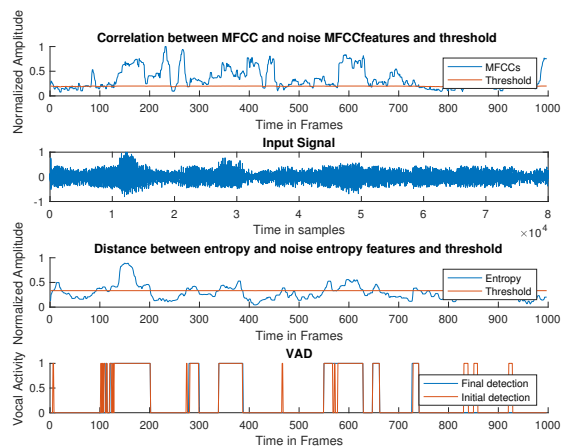


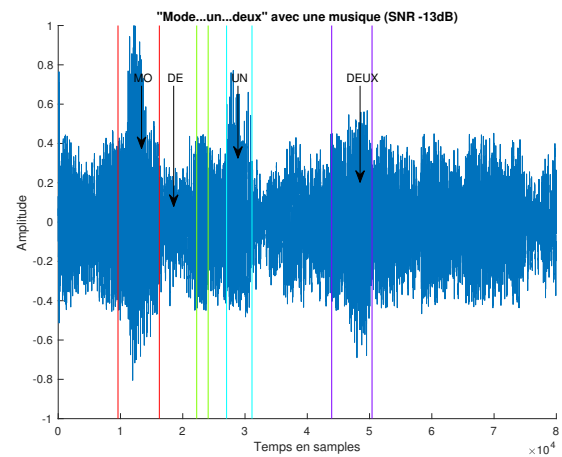
FIGURE 10 – Algorithme utilisant l'entropie spectrale : (a) Entropie spectrale et son seuil de détection ; (b) Segmentation finale du Signal "Mode un deux"

4.2.4.3 Notre algorithme

Nous avons toujours le même problème que précédemment [FIGURE 8 - b], le segment "de" n'est pas détecté et nous introduisons même un segment de bruit [FIGURE 11 - b]. De plus, on se rend compte de l'importance de traiter les segments avec une des fonctions de notre algorithme que nous avons exposé ci-dessus. Sans cette fonction déterminant si la longueur du segments est suffisantes, nous aurions introduit beaucoup de segments de bruits [FIGURE 11 - a; Courbe 4]. En effet, même si la corrélation entre le vecteur de coefficients cepstraux de chaque trame avec celui contenant les coefficients de bruit de fond reste assez robuste à un environnement musicale, l'entropie se retrouver elle un peu moins précise [FIGURE 11 - a; Courbe 3]. D'après l'article de Hongzhi Wang et Yuchao Xu, Meijing Li [12], la corrélation serait plus précise que la distance euclidienne que nous utilisons pour l'entropie. Des modifications seraient donc à faire pour utiliser l'entropie. Nous pourrions ainsi vérifier si cette méthode rendrait plus précis ce critère que nous utilisons (distance euclidienne entre la valeur de l'entropie de chaque trame et celle du bruit de fond). En comparaison si l'on exploite notre première critère de détection [FIGURE 11 - a; Courbe 1], on remarque celui-ci à seulement introduit un léger offset et reste globalement précis.



(a)



(b)

FIGURE 11 – Notre algorithme de détection d'activité vocale : (a) Distance entre le vecteur de coefficient spectraux de chaque trame et le vecteur de coefficient de bruit de fond ; Distance entre la valeur de l'entropie spectrale de chaque trame et la valeur de l'entropie correspondant au bruit de fond ; (b) Segmentation initiale et finale du Signal "Mode un deux"

4.2.5 Tableau récapitulatif

Nous allons résumer l'ensemble des résultats de segmentation précédents [TABLE 1] dans le but d'avoir des résultats plus concrets. Les appellations suivantes ont pour signification :

- NSD : Nombre de segments détectés
- NSV : Nombre de segments contenant une activité vocale
- NSB : Nombre de segments contenant du bruit de fond
- DO : Détection optimale [FIGURE 1]. La mention "Oui" signifiera que la segmentation est optimale sinon la mention "Non" apparaîtra.
- TE : Taux d'erreurs :

$$Taux_{erreur} = \frac{NSB}{NSD} \quad (18)$$

- TR : Taux de réussite :

$$Taux_{reussite} = \frac{NSV}{NSD} \quad (19)$$

- TP : Taux de précision :

$$Taux_{precision} = \frac{TR - TE}{100} \quad (20)$$

- SC : Spectral centroid
- STE : Short time energy

Algorithme	Test	NSD	NSV	NSB	DO	TE	TR	TP
Notre algorithme	Sans bruit	4	4	0	"Oui"	0%	100%	1
	SNR 30 dB	3	3	0	"Oui"	0%	75%	0.75
	SNR 13 dB	4	3	1	"Oui"	25%	75%	0.5
Entropie Spectrale	Sans bruit	4	4	0	"Oui"	0%	100%	1
	SNR 30 dB	3	3	0	"Oui"	0%	75%	0.75
	SNR 13 dB	1	1	0	"Non"	0%	25%	0.25
SC et STE	Sans bruit	4	4	0	"Oui"	0%	100%	1
	SNR 30 dB	22	4	18	"Non"	81%	100%	0.19
	SNR 13 dB	12	3	9	"Non"	75%	75%	0

TABLE 1 – tableau récapitulatif des segmentations de notre algorithme et ceux de références

4.2.6 Conclusion

On peut conclure cette partie concernant la robustesse aux bruits en énonçant que notre algorithme est très robuste au bruit comparé à ceux présentés. On voit très rapidement que le bruit "parasite" la détection de segments de voix. Ce n'est pas le cas pour notre algorithme même si des phonèmes ne sont pas détectés. Il faut donc par la suite veiller à bien superviser notre algorithme de reconnaissance de la parole¹¹. En d'autres termes, il devra considérer des segments manquants selon le bruit qu'il détecte ou non. De plus, comme notre algorithme détecte des segments de voix (à quelques erreurs près), nos algorithmes en aval pourront donner des résultats plus précis. En effet, ils ne seront pas pollués par des données erronées (ou contenant de "bruit" qui n'ont aucune utilités).

11. On parle ici de l'algorithme global et non le VAD présenté ici.



Après avoir veillé à respecter le cahier des charges concernant la robustesse au bruit, il s'agit de nous intéresser aux performances de notre algorithme.

4.3 Performances

Nous en avons parlé un peu plus haut, mais outre la robustesse, il faut être performant. Rien ne sert d'avoir un algorithme qui fait peu d'erreur mais qui prend cent fois plus de temps. Il est donc important de comparer la vitesse d'exécution des algorithmes proposés au notre. Bien-sûr, on sait d'or-et-déjà que notre algorithme est plus lent que ceux présentés. Comment le savoir avant même de réaliser les tests de chronos ? Tout simplement car nos méthodes n'emploient que des caractéristiques fréquentielles. De part la complexité des calculs engendrés, nous n'aurons donc de meilleurs temps d'exécution. Néanmoins, si nous sommes en dessous de vingt fois leur temps, nous pourrions considérer notre algorithme performant au vue de sa robustesse au bruit. Nous générons en effet moins de segments parasites qui ralentiront le traitement par la suite. Enfin, nous savons que notre algorithme est grandement améliorable notamment en terme de répétition. Nous avons, après analyse, constaté que nous répétions certaines opérations, notamment les transformées de Fourier Discrète plusieurs fois (quatre fois dans notre cas, deux pour le calcul de l'entropie, deux autres pour les MFCCs). Nous pourrions gagner un temps considérable. L'algorithme évoluera à cet effet lorsque nous l'implémenterons en C++.

4.3.1 Déroulement

Pour quantifier les performances des algorithmes que nous confrontons au notre, nous avons choisi d'utiliser les fonctions "tic" et "toc" sur Matlab[®]. Nous avons réaliser plusieurs fois les algorithmes afin d'en relever le temps moyen d'exécution. Ils seront résumés dans le tableau un peu plus bas. On considère le temps de départ ("tic") après que l'on est acquis le signal que l'on souhaite étudier. Le temps de fin ("toc") est positionné à la fin de l'exécution de la dernière fois des algorithmes. De plus, le temps d'exécution est calculé pour les différents cas proposés ci-dessus.

4.3.2 Résultat

Nous nous tâcherons de commenter les résultats après les avoir exposés ci-dessous.

Algorithme	Temps moyen	Temps min	Temps max	Ratio
Notre algorithme	0.802	0.7708	0.8803	41
Algorithme utilisant l'entropie spectrale	0.65	0.58	0.71	33
Algorithme utilisant le Spectral centroid et le STE	0.0195	0.0184	0.0216	1

TABLE 2 – Tableau résumant les performances des algorithmes

Nous ne sommes pas peu fier des performances de notre algorithme même si le temps d'exécution pour des trames de 5s est assez conséquent (40 fois plus lent que le plus rapide). Néanmoins, nous pouvons au moins réduire ce temps de moitié si ce n'est plus. Enfin, nous venons de mener une opération de détection sur une trame en temps non-réel. Peut-être est-il plus performant en temps réel ? Nous veillerons lors de notre implémentation en C++ à réduire le temps d'exécution de l'algorithme que nous vous avons présenté. Nous pensons qu'une réduction au tiers serait un bel enjeu. Cela permettrait de se qualifier de performant au vue de la complexité des calculs que nous employons. Une optimisation temps réel est aussi possible.

L'algorithme utilisant l'entropie spectrale utilisant certaines de nos fonctions dont la transformée de Fourier à court terme, son temps d'exécution se trouve sans doute élevé à cet effet. Il faudra donc optimiser cette fonction. Pour information, le spectral centroid utilise aussi une transformée de Fourier (*fft.m* sur Matlab[®]) néanmoins elle semble plus optimisée que la notre.

5 Conclusion

Nous pouvons conclure d'or-et-déjà que notre algorithme se révèle très robuste au bruit et même accompagné d'un fond musical avec un SNR très faible (moins de 13dB). Comparé à ce que la littérature nous sommes en effet bien plus performant sur cet aspect là. Cependant, en terme de rapidité d'exécution, nous avons des résultats plutôt médiocre bien qu'avec une optimisation du code, nous pourrions avoir de meilleurs résultats. Il faudra donc veiller par la suite à optimiser notre code afin qu'il est soit et robuste et rapide. Nous allons aussi par la suite réalisé un seuil pour l'entropie à l'aide d'une sigmoïde pour gagner en performances de détection.

Références

- [1] Les sons voisés et non voisés. http://www.clg-camus-miramas.ac-aix-marseille.fr/spip/sites/www.clg-camus-miramas/spip/IMG/pdf/Les_sons_voises_et_non.pdf. [Online; Accessed : 2017-03-27].
- [2] Mel frequency cepstral coefficient (mfcc) tutorial. <http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/>. [Online; Accessed : 2017-03-15].
- [3] Record and play audio. https://www.mathworks.com/help/matlab/import_export/record-and-play-audio.html. [Online; Accessed : 2017-02-04].
- [4] Ouzounov .A. Robust features for speech detection a comparative study. *In Int. Conference on Computer System and Technology*, 2005.
- [5] Patrick Durand and René Durand. Les tomates tueuses. *Le beau journal*, page 24, jan 2007.
- [6] Theodoros Giannakopoulos. Silence removal in speech signals. <https://www.mathworks.com/matlabcentral/fileexchange/28826-silence-removal-in-speech-signals>. [Online; Accessed : 2017-03-27].
- [7] Theodoros Giannakopoulos. A method for silence removal and segmentation of speech signals, implemented in matlab. 2009.
- [8] Faran Awais Butt Madiha Jalil and Ahmed Malik. Short-time energy, magnitude, zero crossing rate and autocorrelation measurement for discriminating voiced and unvoiced segments of speech signals. *978-1-4673-5613-8©2013 IEEE*, Mai 2013.
- [9] Philippe Renevey and Andrzej Drygajlo. Entropy based voice activity detection in very noisy conditions. *In Int. Conference on Computer System and Technology*. [Online; Accessed : 2017-03-27].
- [10] Hung J. Shen J and Lee J. Robust entropy-based endpoint detection for speech recognition in noisy environments. *In Fifth International conference on spoken Language Processing*, 1998.
- [11] Prof. Lawrence Rabiner (Rutgers University, Prof. Ronald Schafer (Stanford University) Kirty Vedula University of California, Santa Barbara), and Siva Yedithi (Rutgers University). Endpoint detector. <https://www.mathworks.com/matlabcentral/fileexchange/45314-endpoint-detector1>. [Online; Accessed : 2017-03-27].
- [12] Hongzhi Wang and Meijing Li Yuchao Xu. Study on the mfcc similarity-based voice activity detection algorithm. *Artificial Intelligence, Management Science and Electronic Commerce (AIMSEC), 2011 2nd International Conference on*, 2011.