RAPPORT DE TRAITEMENT DU SIGNAL :

A SPEAKER CLASSIFICATION PROJECT .

I. SIGNAL PRE-PROCESSING

Dans le code en annexe, le code implémentant cette partie est contenu dans le Script 'Preprossing.py', on y trouve deux fonctions :

1-) La fonction 'Normalize' où on charge le signale et on le normalise.

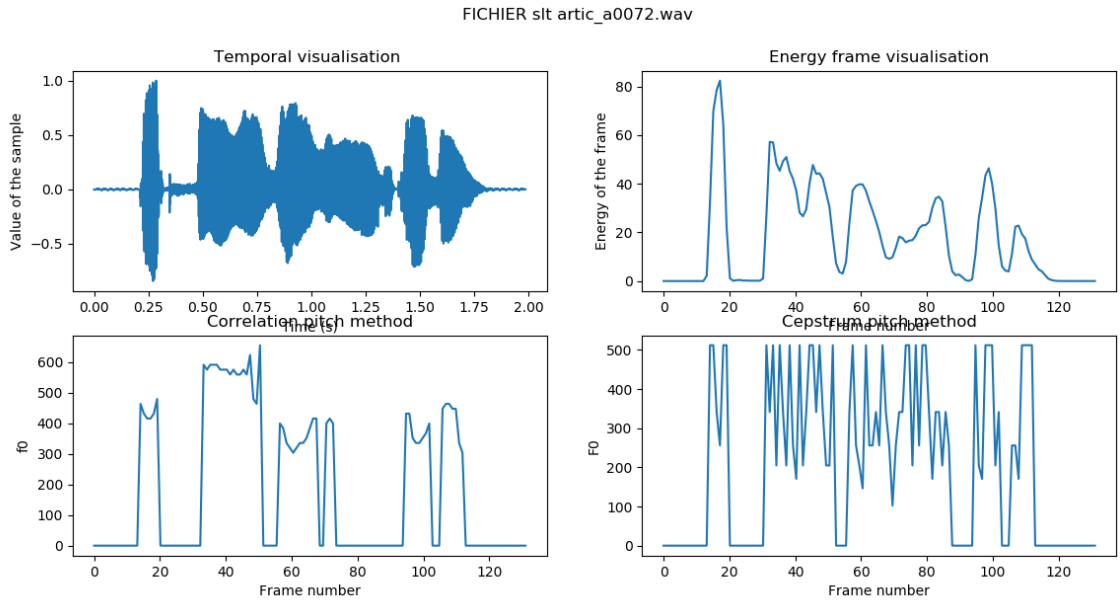
2-) La fonction 'makeframe' qui découpe le signal en plusieurs fenêtres de longueur 'width' avec un pas de 'step'.

II. FEATURES EXTRACTION ALGORITHMS

A. Signal Energy

Dans Script 'Energy.py' ont a implémenté la fonction energy qui calcul l'énergie pour les différentes voix.

B. Pitch



La fonction pitch(n) contenue dans Pitch.py permet de visualiser nos deux méthodes d'estimation du pitch, la seule vraiment fonctionnelle étant celle du cepstrum située en bas à droite du graphe. Grâce à la visualisation de

l'energie correspondant à chaque frame (graphique en haut à droite), on a estimé que notre valeur seuil devait valoir 8.

1. Autocorrelation-Based Pitch Estimation System:

La fonction corrpitch est utilisée afin de calculer l'autocorrélation d'une frame, ensuite nous trouvons l'emplacement des sommets de cette autocorrélation et faisons la différence entre les deux plus haut sommet afin de trouver la fréquence fondamentale F0 pour les sons vocaux.

Cette fonction ne semble pas très fonctionnelle puisque les valeurs de F0 ne sont pas uniquement comprises entre 80 et 500 Hz(graphique en bas à guauche).

2. Cepstrum-Based Pitch Estimation System

La fonction cepstrumpitch permet de calculer l'inverse de la transformée de fourrier du logarithme du spectre d'une frame.

Ensuite on cherche les sommets correspondants et on reprend la valeur correspondante à w avant la transformée inverse pour calculer la F0 des sons vocaux. Cette fois, nous avons pu observer très nettement que les valeurs se situent entre 80 et 500 Hz(Graphe en bas à droite), c'est cette fonction qui sera reprise afin de construire notre RuleBasedSystem.

C. Formants

Cette partie est implémenté dans le Script Formant.py et contient :

- La fonction PreEmphasis qui est notre filtre passe haut
- La fonction formant qui nous permet de calculer nos formants. Cette fonction nous sort normalement un tableau de plusieurs colonnes mais nous ne prenons que les valeurs de la colonne 1 à 3. Les graphiques si dessous représentent d'une part les valeurs que nous retourne la fonction formant et d'autre part le tableau des formants qu'on peut retrouver sur Google.

Voyelle (API)	F_1 (Hz)	F_2 (Hz)
[u]	**320**	800
[o]	**500**	1 000
[ɑ]	**700**	1 150
[a]	**1 000**	1 400
[ø]	500	**1 500**
[y]	320	**1 650**
[ɛ]	700	**1 800**
[e]	500	**2 300**
[i]	320	**3 200**

```
fomant
[[0.0, 585.5842654246368, 1035.0230807649443], [0.0,
353.69893588590014, 1531.9695248460027], [0.0, 389.5209701512976,
2330.1339669326226], [1453.4616956729142, 2296.6238280334283,
3012.9813848778685], [500.13806815770494, 1748.007993167176,
2768.0819991042904], [0.0, 413.77065768494464, 1379.075218242849],
[1618.2732463138323, 2552.933451798263, 3632.3980115514646],
[395.47855339904737, 2378.8555762224573, 2602.3940008742816],
[0.0, 445.22852743932054, 1417.660576186538]]
```

En observant ces deux graphiques on se rend que les valeurs par la fonction formant sont proches de celles attendu

D. MFCC

Le Script MFCC.py implémente cette partie.

- Le signal est d'abord filtré dans la fonction preEmphasis2
- Il passe ensuite dans la fonction hamming où on applique la fenêtre de hamming à chaque frame, les frames sont construits au début de la fonction hamming.
- On calcule ensuit la puissance de chaque frame grâce à la formule donnée dans l'énoncé.
- On applique ensuite successivement les fonction filter_banks et dct sur la puissance et le retour du filter_banks. On obtient un vecteur de MFCC dont on ne garde que les 13 premières colonnes. La figure si dessous nous donne les premières lignes du tableau des valeurs obtenues pour le mfcc.

```
[[ -48.15160219   21.42650635   27.28557124  -46.76081296
28.59777917
   -44.31992427  -26.68792339  -38.29968615   -3.60404568
22.90739992
    -5.61352292  -17.41267743   13.32750102]
 [-113.04343876  -14.73739142   14.0723344   -17.26647432
28.70313768
     5.56372042    4.07870628  -11.20822915    7.79879057
0.61843708
   -32.83925317  -28.06258824   -0.77997405]
 [ -27.83737062  -45.66829106   -7.77298952   -2.06445688
75.84142162
     7.06306915   -5.54109725  -18.40851018  -44.699694
19.0896777
   -19.53520026  -15.60157076   13.55856736]
 [ -61.17439377  -42.79500936   -5.18817714  -11.98024245
3.39515168
    -4.40191924   13.33559264   -0.5986354   -30.26683075
4.83298967
   -13.61533357  -26.75330337    8.60706973]
 [ -69.04470023   -9.88269749   -8.79162539  -51.56319173
50.74027453
   -17.78534693   -8.92492754  -12.17738438  -16.09376605
2.88949948
   -31.20482826  -13.55419274    4.23712351]
 [ -90.78680439  -10.63151022   21.93454333  -10.94298986
32.10400516
   -22.22130145  -20.96856516  -13.29752666    8.79694061
14.76159153
   -29.30698543  -30.14277039   -4.30133667]
```

E. Rule-Based System :

```
b0050
b0092
a0195
a0012
b0159
b0372
b0190
b0344
a0130
a0542
b0207
b0465
b0246
b0296
b0462
bdl mean = 127.08114701052313
stl mean = 193.91565973626973
la moyenne des F0 est : 160.49840337339643
real values are : [1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1]
rule based test found : [1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1]
1 = bdl ; 0 = stl
le taux de reconaissance est de 80.0 %
```

RuleBased.py contient tout le code correspondant. Nous réutilisons d'abord la fonction pitch contenue dans Pitch.py afin de déterminer la moyenne des Fréquences fondamentales avec un même nombre de fichier observer pour bdl et stl.

Ensuite la fonction rulebasedtest() permet de sélectionner aléatoirement un fichier .wav (stl ou bdl) afin d'en calculer la moyenne des fréquences fondamentales et de réaliser un classement (stl ou bdl) en comparant avec le seuil recueilli à la sortie de la fonction pitch. Ensuite le taux de reconnaissance est annoncé (ici sa valeur est de 80% dans cet exemple,

en général cela varie entre 70 et 100%), notre Rule Based System est donc fonctionnel.

F. Machine learning :

Nous n'avons pas eu le temps de commencer cette partie, mais plusieurs méthodes de machine learning peuvent être employées dans le cadre d'audio Recognition, les plus en vogues étants les méthodes du DeepLearning.