Abderahman KRIOUILE

G2 TP2

Introduction aux Applications Informatiques

Numérisation du Son et de l'Image

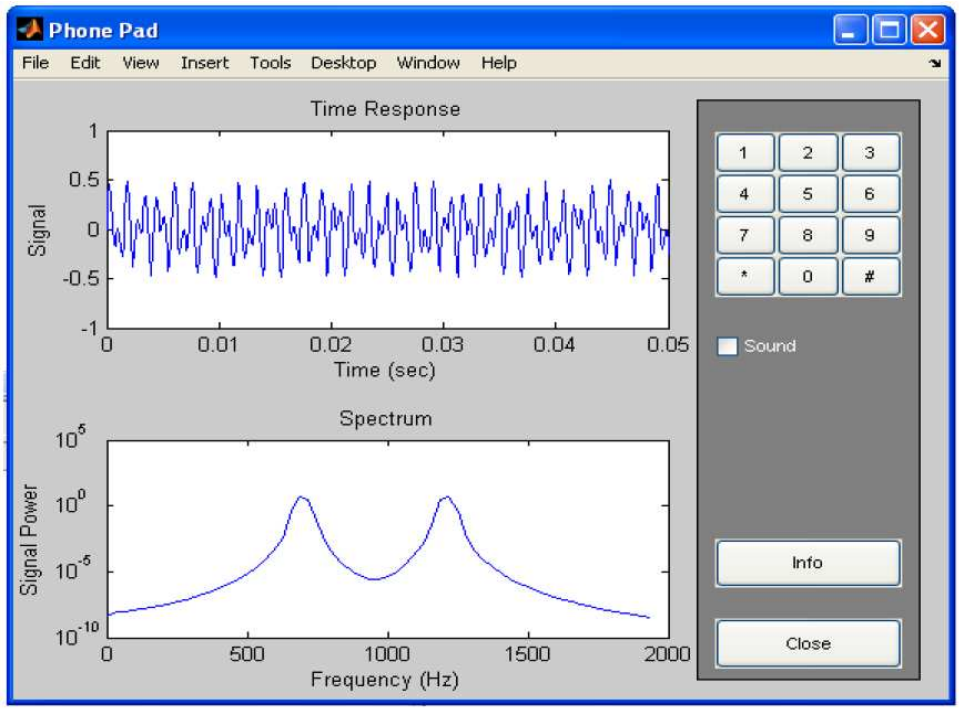
Objectifs :

Etudier la forme d'un signal de type parole et d'un signal de type image, comprendre l'influence de la fréquence d'échantillonnage, de la quantification sur un signal temporel (son) ou spatial (image) et se sensibiliser à la reconnaissance de formes.

A. Son :

L'outil phone de MATLAB permet de visualiser le signal du son produit par chaque touche du téléphone à la fois dans le domaine temporel et fréquentiel :

>> phone



>> load num_tel

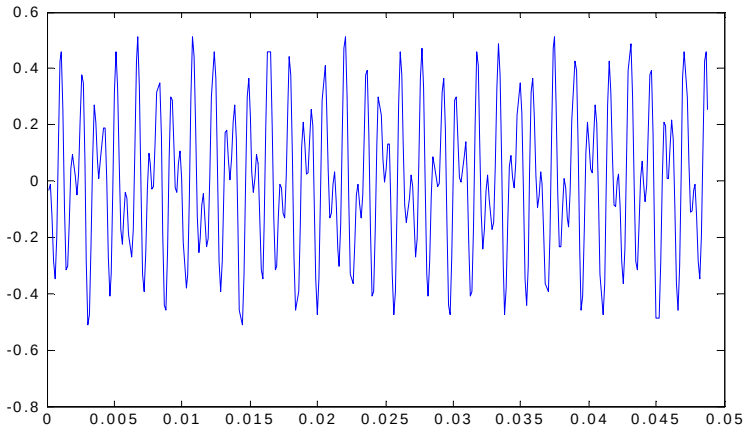
>> whos

Name	Size	Bytes	Class	Attributes
tones1	400x14	44800	double	

Représentation en fonction du temps du signal :

>> t=tones1(:,14) ;

>> plot(t,tones1(:,1))

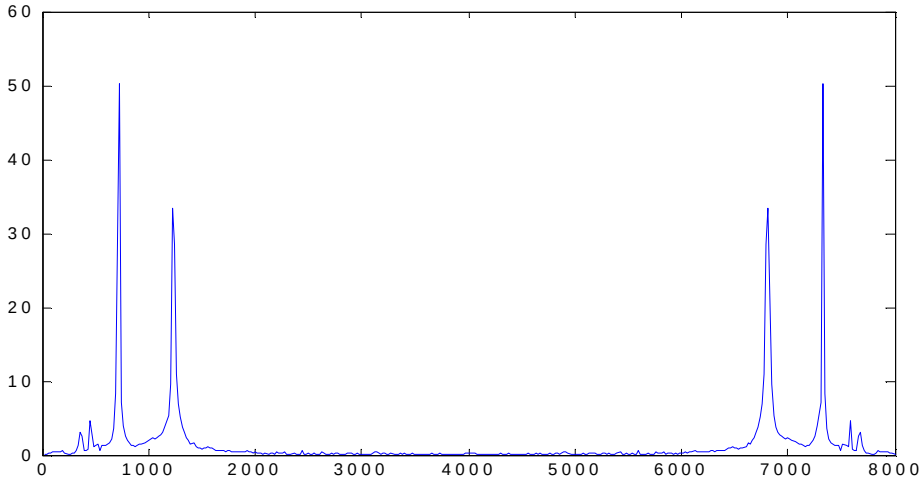


On ne peut pas distinguer les différents sons sous la forme temporelle.

Représentation en fréquence du signal :

>> t1=(t/0.05)* (400/tones1(400,14)) ;

>> plot(t1,abs(fft(tones1(:,1))))



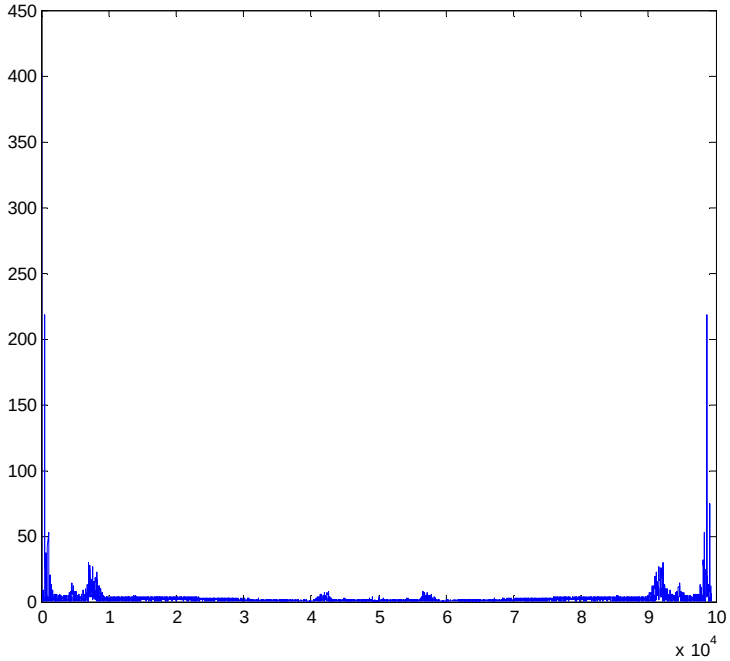
On peut reconnaitre quel chiffre a été tapé grâce aux deux fréquences marquantes de chaque signal (ici nous avons une précision de 20Hz)

- Signal 1 à 720 et 1220
- Signal 2 à 720 et 1360
- Signal 3 à 720 et 1500
- Signal 4 à 780 et 1220
- Signal 5 à 780 et 1360
- Signal 6 à 780 et 1500
- Signal 7 à 880 et 1220
- Signal 8 à 880 et 1360
- Signal 9 à 880 et 1500
- Signal 10 à 960 et 1220
- Signal 11 à 960 et 1360
- Signal 12 à 960 et 1500
- Signal 13 à 360 et 460
- Signal 14 à 0 et 0

Les sons « i » et « a » :

Avec la commande wavr_16, on charge le signal i1_44_16.wav correspondants au son « i »

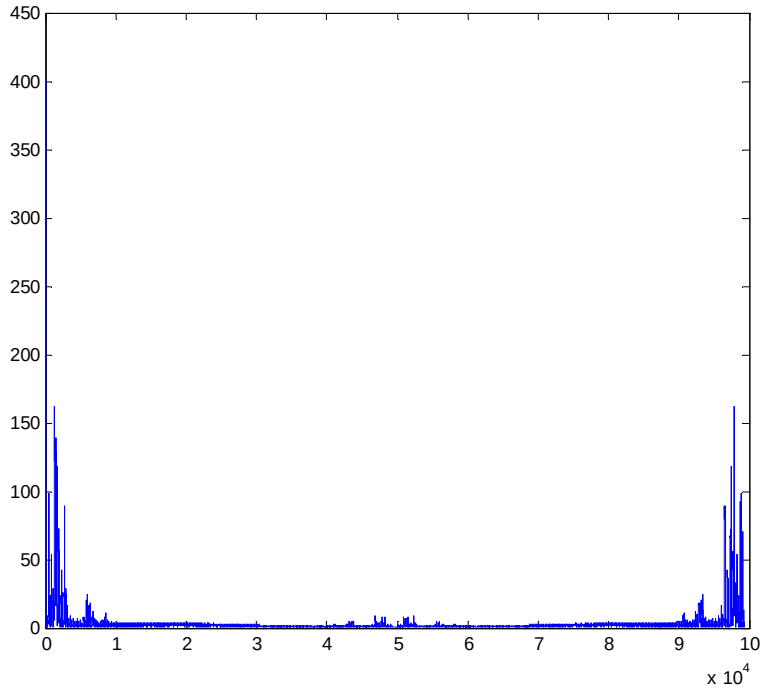
```
>> [y1 , Fe1]=wavr_16('i1_44_16.wav');
>> sound(y1, 40000)
>> plot(abs(fft(y1)))
```



Avec la commande wavr_16, on charge le signal a2_44_16.wav correspondants au son « a »

```
>> [y2 , Fe1]=wavr_16('a2_44_16.wav');
>> sound(y2, 40000)
```

```
>> plot(abs(fft(y2)))
```



On remarque que pour le son « i » il a qu'une seul fréquence marquante or pour « a » il y a plusieurs.

B. Image :

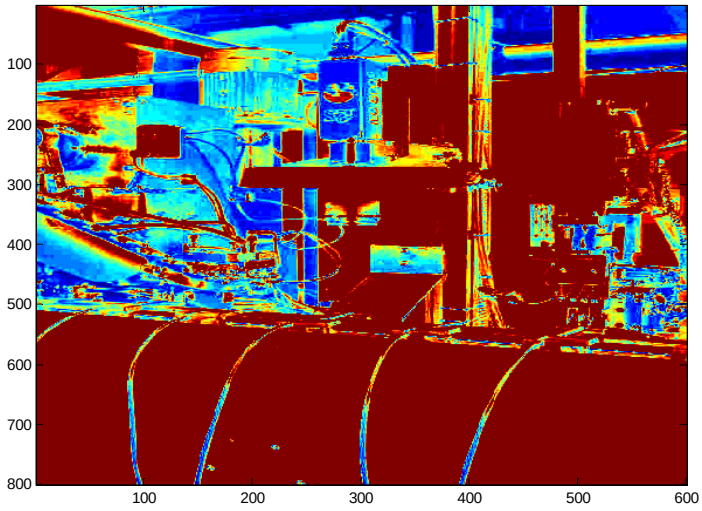
Les fichiers images eur_nb, eur_r, eur_v, eur_b représentent respectivement une image monochrome échantillonnée en 800x600 sur 256 niveaux de gris, et les trois composantes couleur (R,V,B) de la même image acquise en couleur.

On charge dabord les fichiers .mat avec la fonction load :

```
>> load eur_nb.mat
```

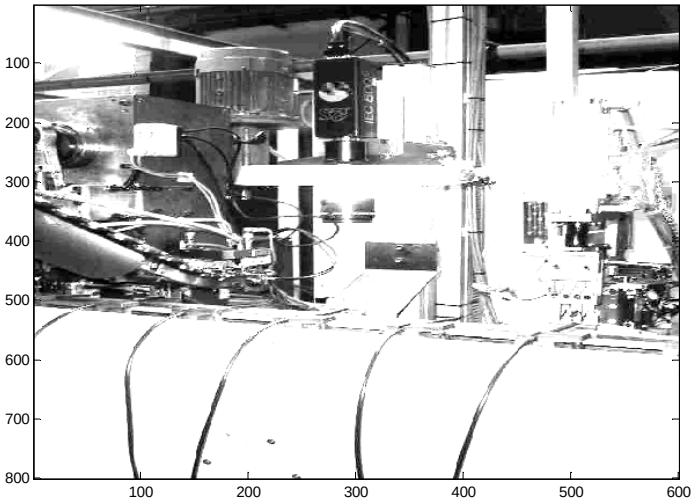
Puis on les visualise à l'aide de la fonction image :

>> image(eur_nb)



Modifiant la table couleur afin de visualiser l'image en niveau de gris :

>> colormap(gray);



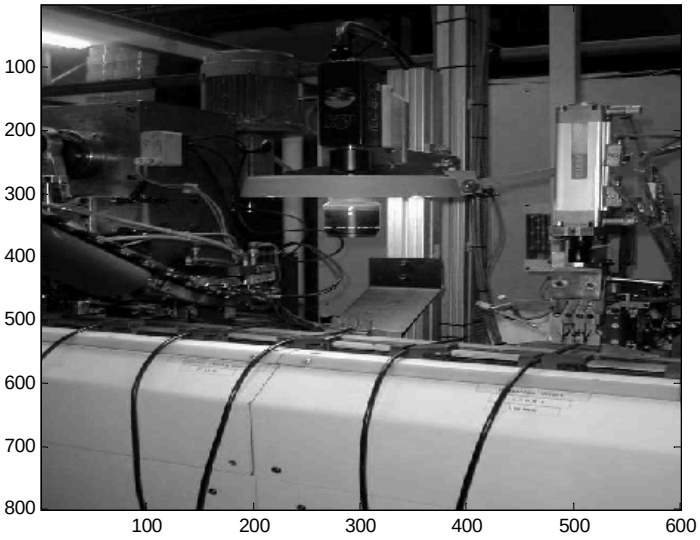
On remarque que les images sont saturées, parce que des détails sont perdu lors de la numérisation ils ne peuvent pas être récupérés, mais en revanche on peut avoir une image « normales » en procédant à une « dilatation », en divisant par 4 chaque échantillons on ajoute les deux bits manquant.

Par exemple : >> eur_v=eur_v./4;

>> image(eur_v)

>>

colormap(gray);



La relation suivante est bien vérifier :

eur_nb = 0.2989 * eur_r + 0.5566*eur_v + 0.1144*eur_b ;

Récupération de l'image a partir des composantes R,V et B :

>> I = zeros(800,600,3);

>> I(:,:,1)=eur_r; >> I(:,:,2)=eur_v; >> I(:,:,3)=eur_b;

>> I=I/64; >> image(I)