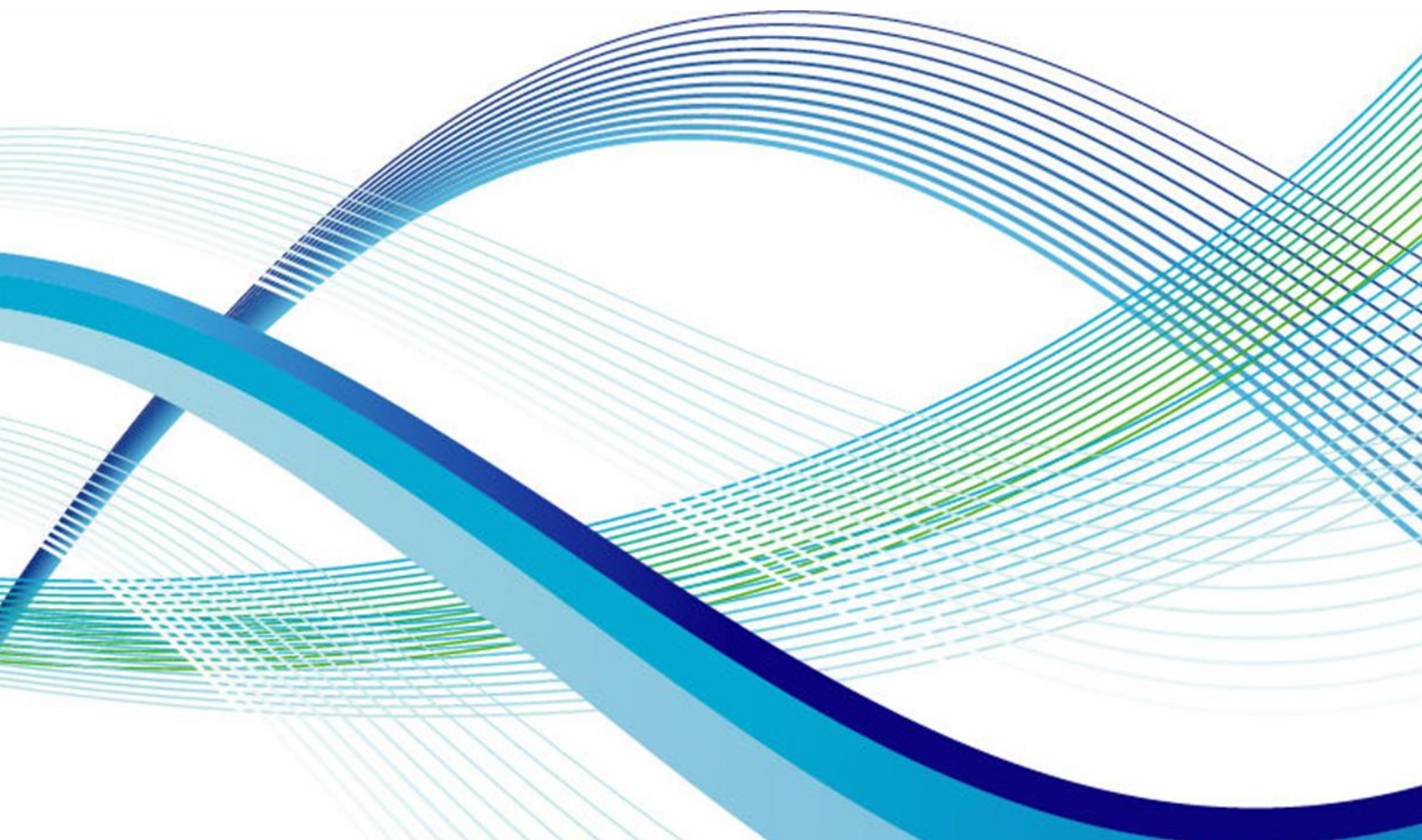


# Word recognition

---

Marie Guénon, Jean-Paul Achard,  
Jean-Dominique Favreau

24/1/2013



# Acknowledgements

---

A huge thanks to J.Leroux for agreeing to supervise and assist us  
all along this project.

## Content

---

Acknowledgements .....	2
Introduction .....	4
I. Sound processing.....	5
1.1 Sound recording.....	5
1.2 Time split.....	6
1.3 Spectrogram.....	7
Hamming window.....	7
Fast Fourier transformation.....	7
Redundancies elimination .....	8
Amplitudes weighting .....	8
Mel's scale and filter bank .....	9
Spectrogram reconstruction .....	11
II. Learning and comparison .....	13
2.1 Existing methods.....	13
Learning .....	13
Comparison: Dynamic Time Warping (DTW) .....	15
2.2 Transition to practice .....	16
DTW local.....	16
DTW Improvementss .....	17
Improvements with high and low frequencies comparison .....	18
Comparison of methods .....	19
III. Human Machine Interface .....	21
3.1 Menu.....	21
Choice of the user.....	21
Options .....	22
3.2 The game .....	23
Labyrinth.....	23
Voice directed.....	23
Conclusion .....	24
Appendices .....	25
1. Log book .....	25
2. References .....	26
3. Intended plan:.....	26
Table of figures.....	27

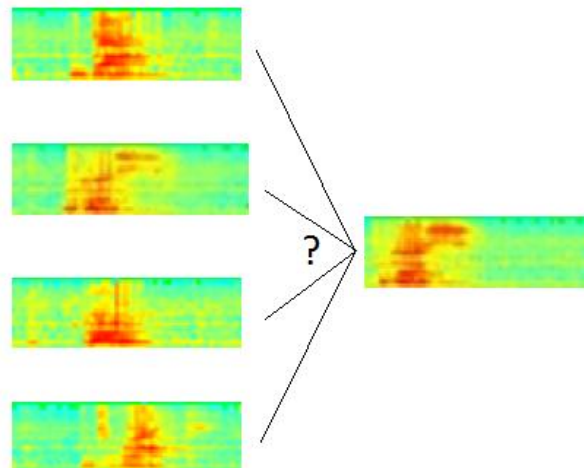
# Introduction

---

The word recognition is a very interesting area, especially the analyses aspects. It is the reason why we chose this project with a huge motivation.

This subject corresponds with our studies but it can also be used in a lot of other areas. Indeed, control by word recognition can be used in home automation system but it can also be used by disable persons who cannot use their hands to do things like playing video games or using their computer.

This project's objective is to create a game which uses word recognition by voice to run. Firstly, we will record and analyze sounds in order to compare their spectrograms and to find which one is the best, as in the example below:



Due to this comparison, we can recognize different words and use this result to control the game.



This is an illustration of our game where you control a frog in a labyrinth.

# I. Sound processing

---

## 1.1 Sound recording

In order to record the words stored in the database, we choose the microphone to use.

During the game, the sound is recorded in streaming (in continuous): we copy what the computer heard into a vector of samples.

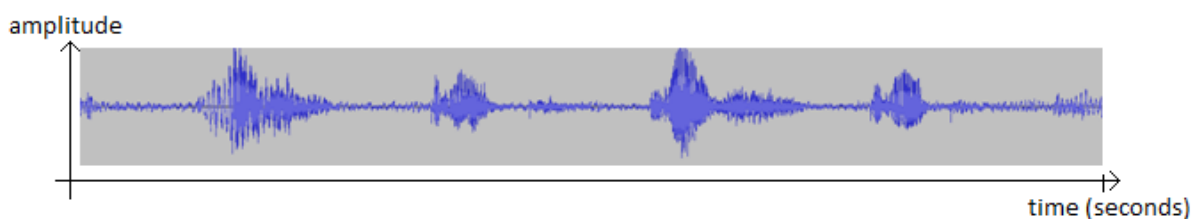


Figure 1 Example of signal: « gauche, droite, gauche, droite »

We start the word beginning detection in the first second of the signal. In the same time, we watch the vector and process it:

If there is not a word beginning in the signal, we delete the first half second of it.

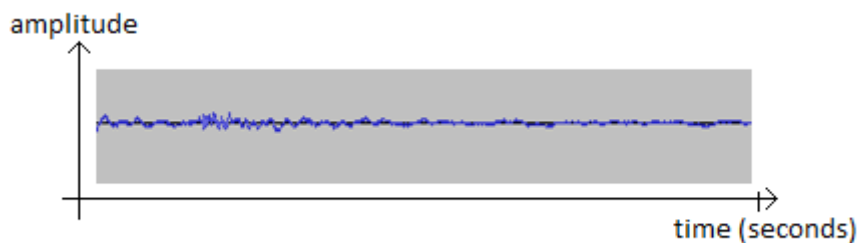


Figure 2 signal without word beginning

If there is a word beginning after 400ms, we delete a part of the signal beginning in order to have the word beginning at approximately 400ms.

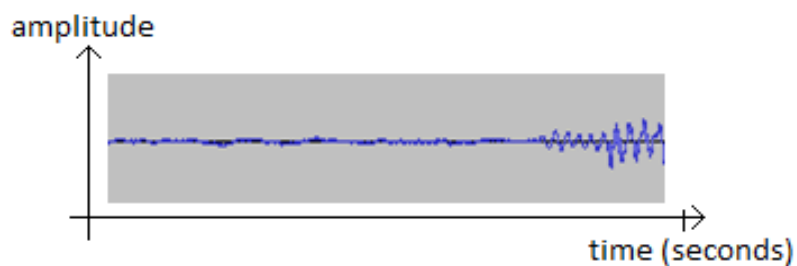


Figure 3 signal with a late word beginning

## Word recognition

Marie Guénon, Jean-Paul Achard, Jean-Dominique Favreau

---

If the word beginning has a good place, we start the processing and the recognition.

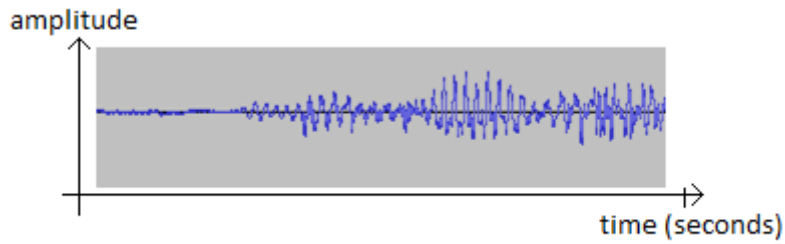


Figure 4 signal with a good word beginning

The results of the recognition are put into a FIFO queue (First Input, First Output) in order to be managed after by our game. Finally, we delete the part of the signal treated from the vector of samples.

### 1.2 Time split

Time is split in slices of approximately 30 ms:

For each time  $t$ , we take a signal's slice which extends from  $t$  to  $t + \Delta$ .

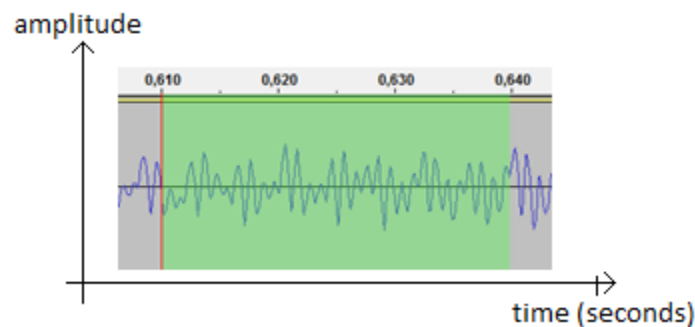


Figure 5 time split example

However, the fast Fourier transformation needs a number of samples which are a power of 2 in order to perform faster. We chose to take signal's slices of size  $n$  (with  $n$  samples), where  $n$  is a power of 2 and so that " $n \times \text{sampling time}$ " is the closest to 30ms.

## 1.3 Spectrogram

### Hamming window

Once the signal is split in slices, we apply to it a Hamming window in order to avoid big discontinuities, and so avoid inconsistent results with the Fourier transformation.

Formula of the Hamming window:

$$h(t) = \begin{cases} 0,54 - 0,46 * \cos\left(\frac{2\pi t}{T}\right) & \text{si } t \in [0, T] \\ 0 & \text{sinon} \end{cases}$$

Where T is the signal's duration of the segment studied

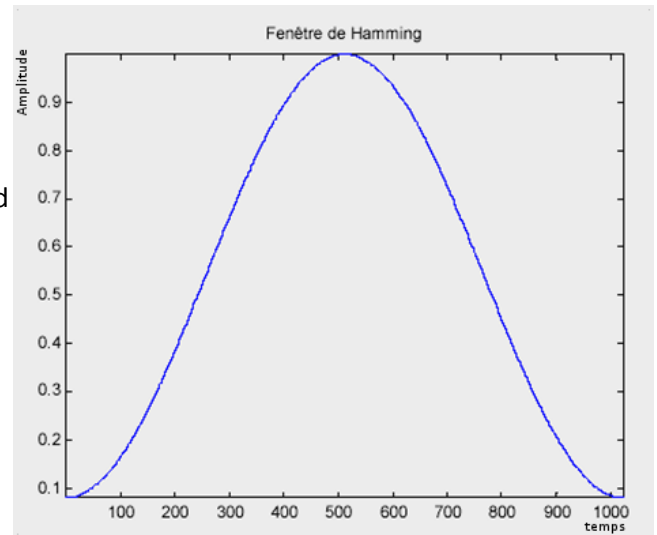


Figure 6 Hamming window

### Fast Fourier transformation

Now, we search to apply the Fourier transformation on each signal's slice:

$$X(k) = \sum_{t=0}^{T-1} x(t) \cdot e^{-2\pi j \frac{kt}{T}}$$

Where  $\begin{cases} T & \text{the signal's size} \\ x(t) & \text{the input signal at time } t \\ X(k) & \text{the value of the Fourier transformation in } k \end{cases}$

However, with an objective of speed, we seek to use the fast Fourier transformation.



Figure 7 Fourier transformation example

## Word recognition

Marie Gu  non, Jean-Paul Achard, Jean-Dominique Favreau

---

### Redundancies elimination

Any Fourier transformation is periodic and symmetrical in 0. Since it is not interesting to study the same thing twice (loose of time), we keep only a signal's half on which will be made the comparisons.

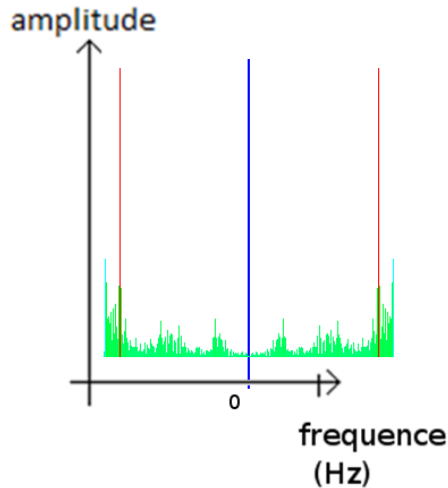


Figure 8 Fourier redundancies elimination

On figure 8, we have on the left the original signal's Fourier transformation. So we can see that this signal is symmetrical in 0 and so, we can keep only one half of it (on the left or on the right) to make our comparisons.

### Amplitudes weighting

Now we seek to minimize the importance of low frequencies. In fact, the auto-correlation has not removed the whole noise present in the recording and this noise parasitizes the low frequencies.

$$y(t) = \underbrace{\log}_{\substack{\text{logarithmic scale} \\ \text{to reduce} \\ \text{the dynamic}}} \left( 0.0000000001 + \|x(t)\|_2 * \underbrace{\frac{\pi * i^{0,6}}{T}}_{\substack{\text{weighting to minimize} \\ \text{the importance} \\ \text{of low frequencies}}} \right)$$

Where  $\left\{ \begin{array}{l} T \text{ the input signal's size} \\ x(t) \text{ the input signal at time } t \\ \|u\|_2 \text{ the function to retrieve the norm of the point } u \end{array} \right.$



### Mel's scale and filter bank

- *Mel's scale*

The Mel's scale is intended to reduce the sensibility in the high frequencies, less finely perceived by the human ear. In fact, the perception intensity of a stimulus does not linearly increase as a function of its power, but exponentially.

The Mel's scale allows moving from the frequency of the input signal (in Hz) to a frequency (in Mel) more representative of the human hearing with the following formula:

$$M = \frac{1000}{\log(2)} * \log\left(1 + \frac{F}{1000}\right)$$

Where F is the input signal frequency at the point considered.

This gives the following curve:

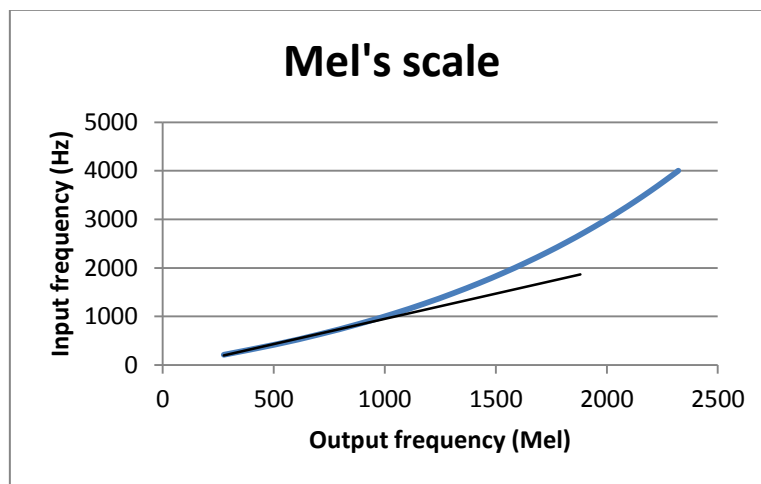


Figure 9 Mel's scale

- *Filter bank*

The filter bank reduces the number of frequencies considered in 20 possible values while respecting the Mel's scale.

Yet the use of this unit is not enough. In order to have a relative bandwidth which remains constant, the filter is built with triangular filters uniformly positioned on the Mel's scale and so, non uniformly on the frequency scale.

This gives us the following curve:

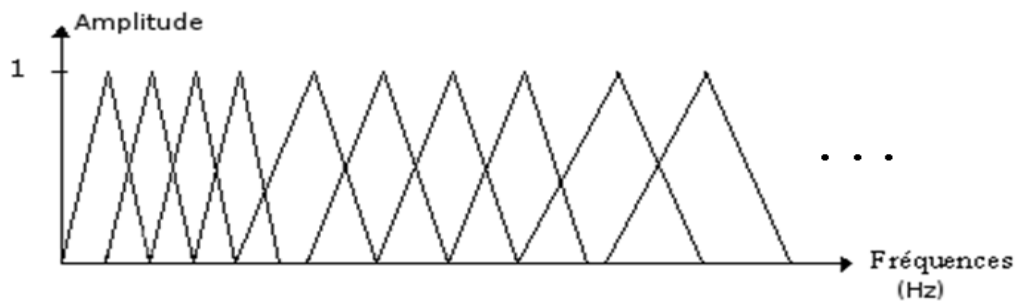


Figure 10 filter bank

We have here an example on the Mel's scale and filter bank transformation: on the left there is the signal before the transformation and on the right the signal after it.

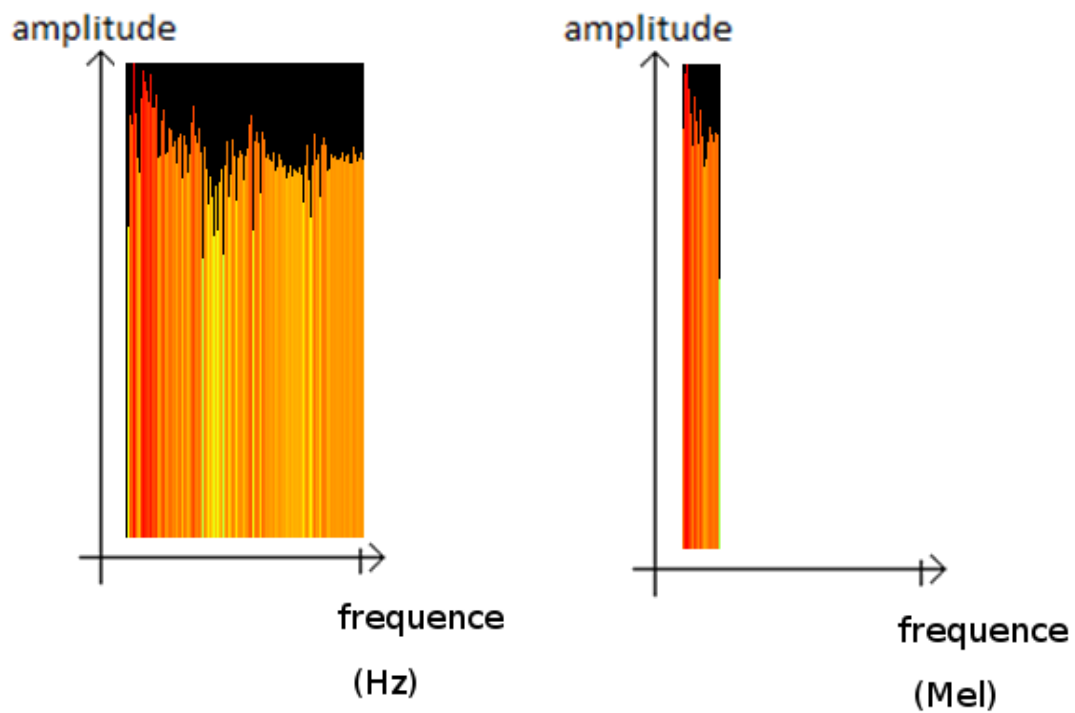


Figure 11 spectrums before and after Mel's scale

If at the beginning we have 128 frequencies, after the transformation we only have 20 frequencies as we explained it before.

## Spectrogram reconstruction

The reconstitution of the spectrogram consists to group together in a matrix the whole Fourier transformation obtained and transformed for each signal's instant.

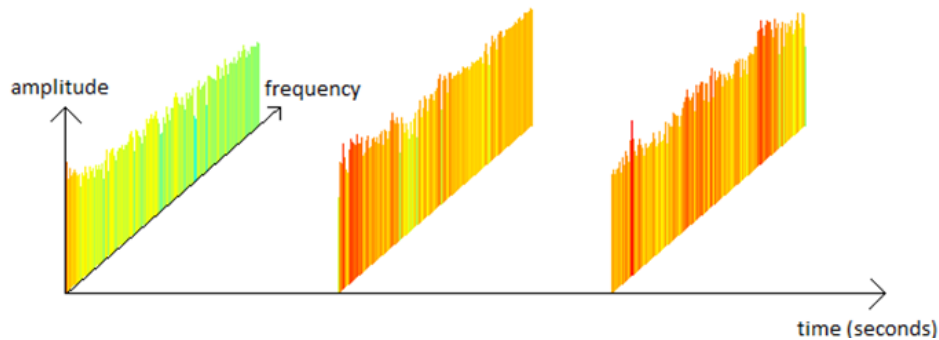


Figure 12 spectrums examples for 3 instants

The rows of the matrix represent the changes in frequency, the columns the changes in time, and the value of each box represents the amplitude of the Fourier transform on the time and frequency given.

Spectrogram obtained before filtering with the Mel's scale:

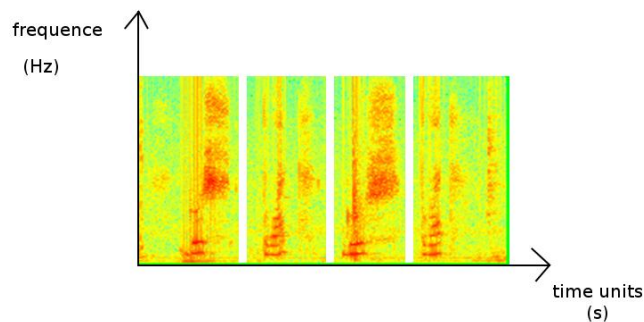


Figure 13 Fourier transformation spectrogram

The same spectrogram obtained after filtering with the Mel's scale:

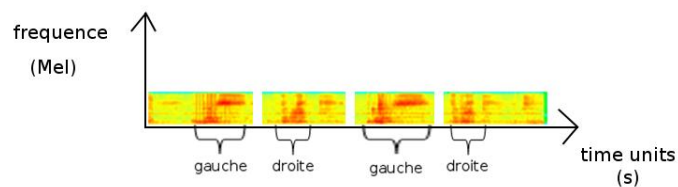


Figure 14 Mel's scale spectrogram

## Word recognition

Marie Guénon, Jean-Paul Achard, Jean-Dominique Favreau

---

We can see here the correspondence between the two spectrograms. We can also see that two identical words seem to have the same form on the spectrogram: for example, the spectrogram for “gauche” has the same high frequency characteristic on each signal. (In a blue circle on the figure 15)

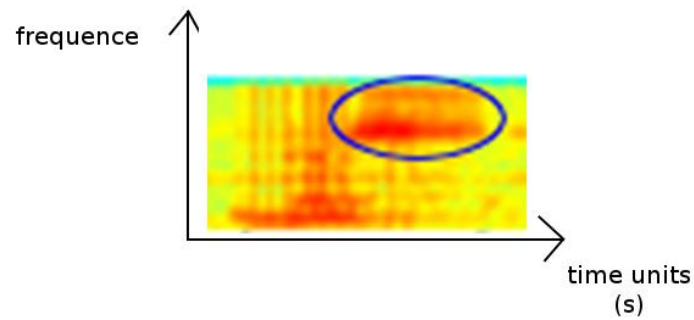


Figure 15 frequency characteristic

## II. Learning and comparison

### 2.1 Existing methods

#### Learning

##### *Artificial neuronal networks*

A neuronal network is often composed of a succession of layers. Each layer is composed of  $N_i$  neurons, having their inputs in the  $N_{i-1}$  neurons of the previous layer.

Each synapse (connection between two neurons) is associated with a synaptic weight, so the  $N_{i-1}$  are multiplied by this weight, and then summed by the neurons of level  $i$ .

Putting the layers of a neuronal network one behind the other tantamount to cascade several transformation matrices and could be reduced to a single matrix, produced by the others.

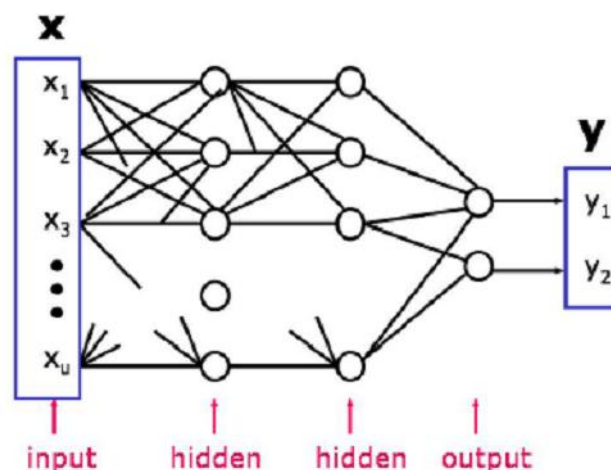


Figure 16 Artificial neuronal networks

For this learning system, the user does not need to adapt his speech and it is not necessary to train the neural network specifically to the user's voice.

However, this method requires large learning database of representative cases that will be often encountered. Moreover in the case of extreme elements, they alter the weight given to each possibility, making the division of the possibilities' space unstable.

Conclusion: In our case, we have a small vocabulary (about twenty words) based on a single-speaker (considering during a recording phase, a single person has to speak).

So, it is not necessary to use a neural network. From a computational point of view, it would be heavier to implement and it would be sufficient to create a new comparison database for each new user.

### *Hidden Markov model*

A Markov chain is an automaton which contains a number of states and it moves from one state to another one with a certain probability.

In the example bellow, we have 3 states: A, B and C. If at time  $t$ , the automaton is in state B, it can move to state C with the probability  $q$  or remain in state B with the probability  $s$ . (Note here that  $r+q=1$ ,  $s+q=1$  and  $t+?=1$ )

In our case (voice recognition), we removed the possibility of going back: time is linear and A, B, C are considered as successive states.

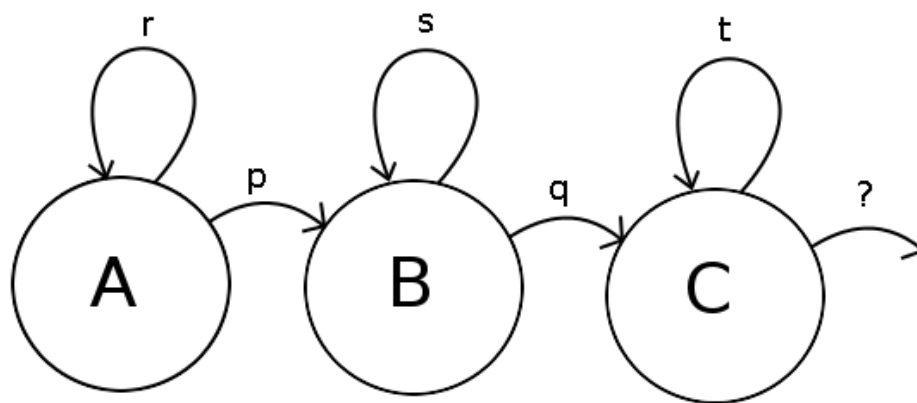


Figure 17 hidden Markov model

The issue is to determine the different transition coefficients from one state to another ( $p$ ,  $q$ ,  $r$ ,  $s$ ,  $t$ ), and then freely apply this algorithm on word recognition.

- This training can be based on the forward-backward algorithm:  
It starts by calculating the balance of probabilities "forward", which give the probability of obtaining the first  $k$  observations in a given sequence, ending in each possible state of the Markov model.
- Then it calculates the set of probabilities "back", which represents the probability of the other cases when an initial state is given.
- Both sets of probabilities can be combined to obtain the probability of being in each state at a given time during the observation of the sequence.

Conclusion: It is not interesting in this case to use the hidden Markov model, because it is too complex for the small vocabulary we are using. It is enough for us to use the Dynamic Time Warping which is a simplification of the hidden Markov model.

## Comparison: Dynamic Time Warping (DTW)

The principle of DTW is to determine for each element of a sequence, the best matching element in the other sequence.

In our case, we have to compare, using a Euclidean distance, the signal's spectrum measured with those of different words composing our vocabulary.

Comparison principle:

For each iteration we compare the  $n^{\text{th}}$  signal's spectrum measured with the reference spectrums from  $m-\Delta$  to  $m+\Delta$  and keep the  $m^{\text{th}}$  nearest spectrum.

The displacement is of the form:

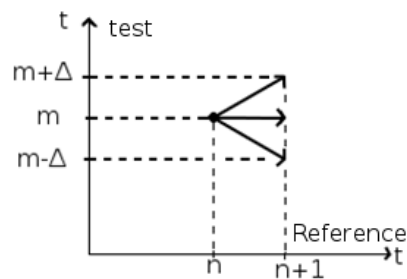


Figure 18 DTW displacement

Arrows represent the possible moves.

Every successive movement put together, we obtain the minimal path between two sequences.

Bellow an example of the shortest path between a sequence tested and the reference sequence representation:

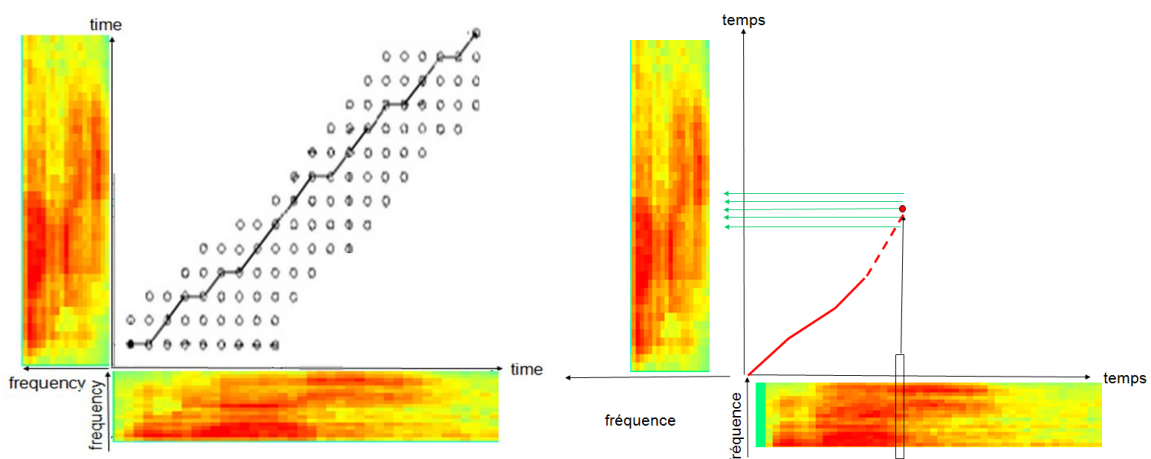


Figure 19 Minimal path examples

Once the distance with every word established, you can get the smallest of them. It corresponds to the reference word detected.

Conclusion: This method is a good way to neglect the speech speed: by its implementation, we can don't move at the same speed throw the two signals to compare. So this method considers cuts and temporal extensions of the signal that are observed on different word pronunciation.

### 2.2 Transition to practice

Due to the small vocabulary we have, after analysis, we chose to use the DTW method to compare different words.

We applied the method described above.

Note that each time that a new user wants to use our word recognition system; he has to record his own vocabulary comparison-base.

But, if a user has already used our system in the same conditions (same room or/and same noise around him), he doesn't have to record it again: it is saved in a specific folder with his user-name.

After the establishment of the comparison base, the user can begin to use our recognition system.

#### DTW local

##### *Limits*

This method has some limits due to how we implement the DTW. Indeed, in this method we have to calculate the distance between two signal's spectrums and it is possibly not the best solution to differentiate two close phonemes like, for example, /a/ and /wa/ from "bas" and "droite". Another possibility is to use the Mahalanobis distance:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T * \Sigma^{-1}(\vec{x} - \vec{y})}$$

Where  $\Sigma$  is the covariance matrix.

Another limitation of this method is studied when the vocabulary grows: the new signal is compared to all other previously studied and then we have to choose the one that is closest.

It is the reason why this method is slower for big vocabularies.

So as long as we have a small vocabulary (~20 words), this method is the best. But if our vocabulary has to grow too much, we have to change the method we use.

As we already said, words must be long and different enough so the DTW can't delete important parties, it is also important that all words to compare have approximately the same width and the same tone. Indeed, if we compare a long word to a shorter word in base, the algorithm will be soon stuck at the end of the short signal and then, the long signal will be compared some noise, so it is not a relevant way to have good results.

Another problem here is that if we have a very oscillating path (big increases or looses in the path indices), it will influence the calculated distances. It is not enough here to force the comparisons between the indices  $m$  and  $m+\Delta$ : it could jump important parts of the signal and don't compare together the interesting parts (similar parts for example).



## DTW Improvementss

### Theory: DTW global

We seek here to avoid these oscillations indexes path and have the best global solution. In order to do that, we seek to calculate on every index (step) the lower path between the reference signal and the comparison signal. (We can consider 20 steps to have the median DTW)

We use the following method from step 1 to step  $\tau-1$ .

$$\forall i \in [0, \text{step} * \Delta] \text{ table } V_{\text{dist}} = \frac{1}{\text{step}} \text{distance}(n + \text{step}, m + i) + \left(1 - \frac{1}{\text{step}}\right) * \text{minDist}[i]$$

$$\forall i \in [0, (\text{step} - 1) * \Delta] \text{ table } \text{minDist} = \min_i(V_{\text{dist}}[i, i + \Delta])$$

Where  $V_{\text{dist}}[i, i + \Delta]$  is the table  $V_{\text{dist}}$  taken only between the indices  $i$  and  $i + \Delta$

Finally, we have a table with the distances minima.

On this table, we search the minimal value and its index. It is on this last one that we will keep as minimal path.

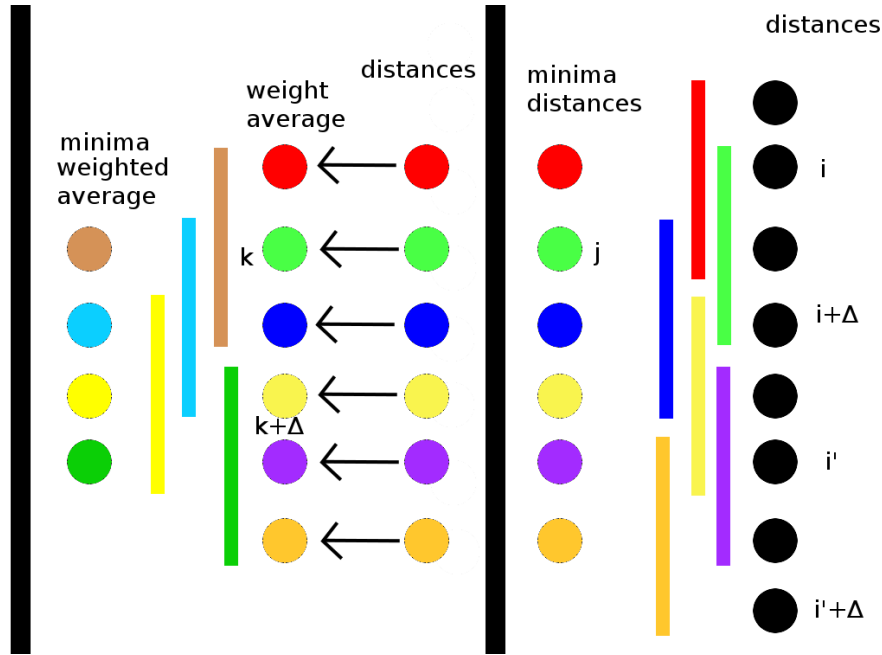


Figure 20 DTW Improvements

On figure 17, we can see an example on our method, where  $\Delta=2$ .

Initialization: we beginning by calculate the distances between the reference signal and the signal to compare (black dots). Then, we calculate the minimal distance for each interval  $[i, i+\Delta]$  because, for example, if we reach the  $j$  index, we will take the shortest path which is the minimum between  $i$  and  $i+\Delta$ .

Recurrence: we re-compute the distances in the same way. Then, we make a weighted average between the distances and the previous minima distances. Finally we calculate the minimal weighted average for each interval  $[k, k+\Delta]$

## Improvements with high and low frequencies comparison

### Theory

- Word starting time detection:

Sometimes, the user doesn't begin to talk with the sound recording beginning. So it is interesting for us to detect the beginning of the word in order to begin the sound processing at the good point.

To detect the beginning of a word, firstly we apply a Gaussian filter on the spectrogram equalized and reduced with the Mel scale in order to smooth the signal (neglect the small noise variations) and so we use the mask:

$$\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

Whereupon, we split the new spectrogram in two: the low frequencies and the high frequencies. (Remark: to improve this method, we could split the new spectrogram in more frequencies (20 like in the Mel scale) in order to gain precision)

We split the signal due to the fact that a word can begin with a sound which is not in the two parts in the same time. We apply the following operations on the whole signal, we detect the point where all frequencies are present in the first time, so it is not necessary the "real" beginning of the word.

For example, the word "should" begin with high frequencies and end with low frequencies.

Then, we calculate the distance between two successive vectors (the norm of the derivative) for each time, so we have a vector of distances.

In this vector, we locate the biggest distance and we keep every distance that is higher than 2/3 of the biggest distance.

If there is only one value which verifies that on a half of the signal and it is smaller than values kept in the other half, it is the beginning of the word. Otherwise, we calculate the average of the second norm on the 5 spectrums on the left and we do the same on the 10 spectrums on the right (by this method, we neglect shorts variations due to noises).

We keep the indices where the left average is lower than 80% of the right average. At least we consider that the beginning of the word is the smallest indices in both low and high frequencies.

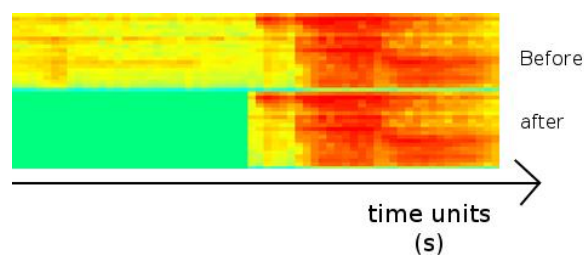


Figure 21 Beginning detection example

## Comparison of methods

In order to see if our method improvements are effective, we had to test each method on the same signal and analyze the results.

To do our tests, we chose the word “gauche” as reference and the word “droite” as testing signal.

We applied each algorithm on the testing signal and we obtain the following curves which compare the minimal path indices in function of the reference indices.

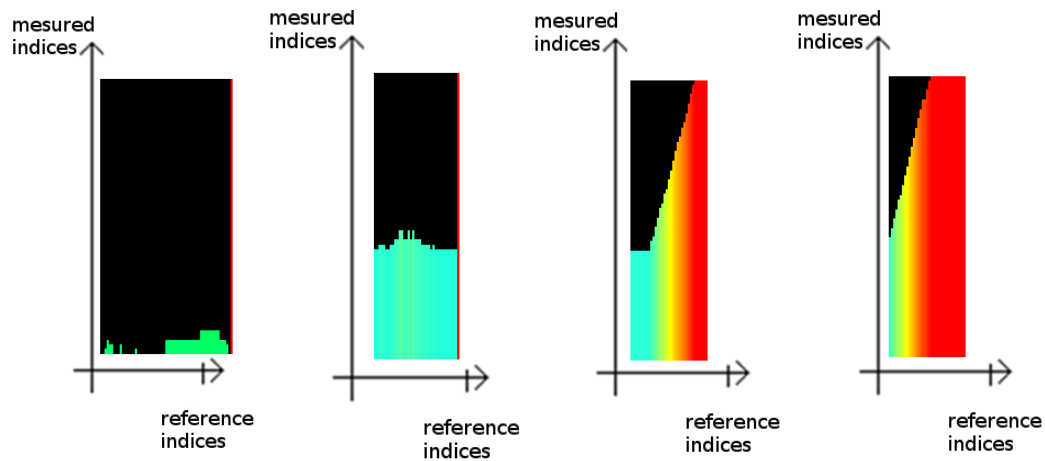


Figure 22 DTW local -- DTW local and beginning detection -- DTW median and beginning detection -- DTW global and beginning detection indices comparison

We can also study the cumulated distance between the two signals.

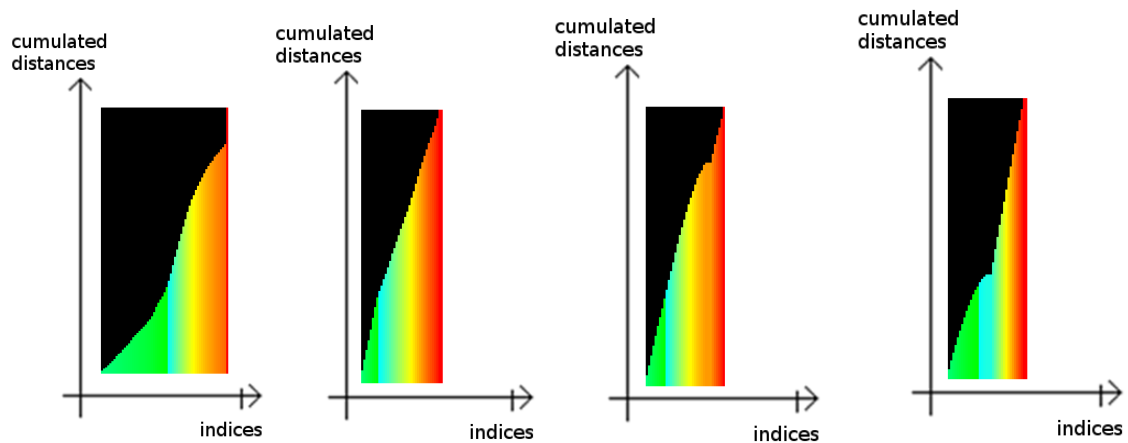


Figure 23 DTW local -- DTW local and beginning detection -- DTW median and beginning detection -- DTW global and beginning detection distances comparison

## Word recognition

Marie Guénon, Jean-Paul Achard, Jean-Dominique Favreau

We based the end of our analyzes on this results to determine which referenced word is the closest of the testing word. That's how we get the following results:

The comparisons here are based on databases where every word has two references.

Words Method	"gauche", "droite", "haut", "bas"	"Bonjour", "Hello", "Maison", "Placard"	"vacherin", "tiramisu", "moelleux", "bûche"	"Riri", "Fifi", "Loulou", "toto"
DTW local <sup>1</sup>	62,5%	80%	70%	70%
DTW local and beginning detection	62,5%	80%	70%	70%
DTW median <sup>2</sup> and beginning detection	85%	85%	98%	65%
DTW global <sup>3</sup> and beginning detection	95%	90%	98%	85%

Figure 24 comparison of methods table

We can see here that the last method is better than the others: we have better results. So we can say that choose the best global minimal path is better than choose the best local path.

However, if we have only one reference by word, the last method works better than the others.

<sup>1</sup> Cf. p14.

<sup>2</sup> Cf. p15.

<sup>3</sup> Cf. p15.

### III. Human Machine Interface

---

In order to test our word recognition system, we wanted to create a didactic and interesting way to do it. So we made a game where the gamer should control the character (here a frog) with his voice.

#### 3.1 Menu

When the game is launched, you have a first menu where you have three possible choices: exit, modify the options or choose the user you want to play.



Figure 25 first menu

#### Choice of the user

If you decided to choose the user you want to play on the previous menu and clicked on the corresponding button, you have the following screen:



Figure 26 choice of the user menu

Here, you can choose to quit this menu, and return to the previous menu. You can also choose a specific user (for example “moi”) and click on “Commencer” to launch the game.

Note that, if there is too many users, they do not all appear on the screen, you have to scroll down to see them (or scroll up to see the previous ones).

## Word recognition

Marie Guénon, Jean-Paul Achard, Jean-Dominique Favreau

---

It is also possible to choose to re-record the sound for a specific user, or to create a new user by clicking on an empty button. In these cases, you have the following menu:



Figure 27 Change user menu

On this menu, you can quit and return on the previous menu, you can change (or create) your username, you can (re-)record your words database, or you can listen to your sounds (and verify if you think that they are correct or not).

When you have finished to change your name or record your word database, you can click on “Commencer” and launch the game, but, take care, it will begin only if name is valid (not empty) and if all the sounds are recorded.

### Options

In the options menu, you can configure input and output settings like the choice of mike or the choice of the sound card drivers.

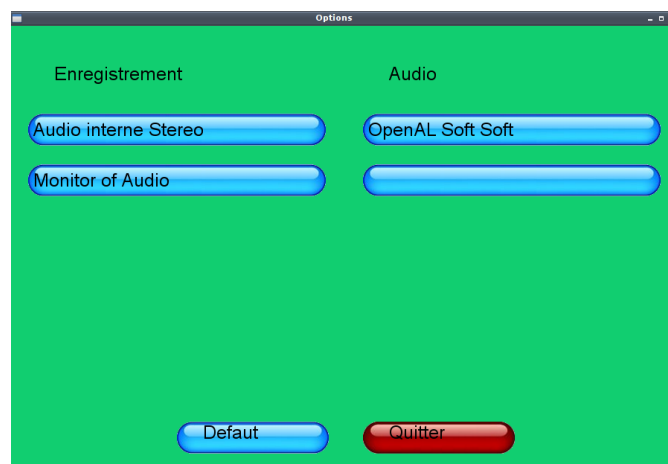



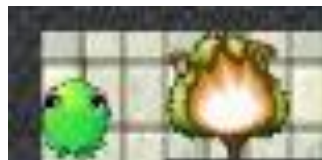
Figure 28 option menu

## Labyrinth

(represented by:  and for example here, at the bottom and in the middle) to the end



## Voice directed



# Conclusion

---

All along this project, we discovered the world of word recognition and the way to apply our knowledge on this field.

Firstly, this project has enabled us to apply our NSP course (Numeric Signal Processing, TNS): even before trying to recognize a word in a signal, we have to process it in order to keep the most interesting parts where we will apply our recognition algorithms.

On the second hand, this project has enabled us to discovered some recognition algorithms (like hidden Markov models or dynamic time warping) and apply them on word recognition and spectrograms. These algorithms were a way for us to apply our knowledge in “Infographie” (image recognition and learning algorithms) because we applied them on spectrograms which are an image representation of the signal.

Therefore, this project enabled us to deepen our knowledge and understanding of the area of words recognition. But, it also enabled us to improve our skills in C++ and multi-thread programming. Indeed, we had to manage and process several events at the same time to make our word recognition program and our game faster and efficient.

To conclude, we can say that this project was very interesting and we learned a lot of things during this period. However, our algorithm is not efficient enough to be applied on a complete vocabulary: it works only on small set of words.

So, if someone else wants to improve our word recognition, he could ameliorate the existent algorithm or change it with the hidden Markov model that is better for a large vocabulary.



# Appendices

## 1. Log book

Monday 7	<ul style="list-style-type: none"><li>•morning: github and librairies (SDL)</li><li>•after noon: tasks definition , work plan, meeting with M. Leroux</li></ul>
Tuesday 8	<ul style="list-style-type: none"><li>•morning: JP-research learning / comparison, M-repport, JD-sound recording</li><li>•after noon: JP-research learning / comparison, M-repport, JD-signal split and FFT</li></ul>
Wednesday 9	<ul style="list-style-type: none"><li>•morning: JP-research learning / comparison, M-repport, JD-Mel and spectrogram</li><li>•after noon: JP-research learning / comparison, M-repport, JD-begining DTW</li></ul>
Thursday 10	<ul style="list-style-type: none"><li>•morning: JP-research learning / comparison, M-repport and presentation, JD- endDTW and tests</li></ul>
Friday 11	<ul style="list-style-type: none"><li>•morning: JP-repport and research on partial correlation, M-repport and presentation , JD- DTW refinement</li><li>•after noon: JP-test and research on Mahalanobis, M-report and presentation, JD-tests</li></ul>
Monday 14	<ul style="list-style-type: none"><li>•morning: JP-researches on Mahalanobis distance, M-repport, JD- DTW refinement</li><li>•after noon: JP&amp;M-HMI begining, JD- DTW refinement</li></ul>
Thursday 17	<ul style="list-style-type: none"><li>•morning: JP-labyrinth and debug HMI, M-debug HMI and image analysis, JD-DTW refinement and tests</li></ul>
Friday 18	<ul style="list-style-type: none"><li>•morning: JP-debug and labyrinth creation, M-debug, report and presentation, JD-DTW refinement, word begining detection and tests</li><li>•after noon: JP-limits tests on labyrinths, M-Menu, JD-DTW and HMI integration</li></ul>
Monday 21	<ul style="list-style-type: none"><li>•morning: JP-Menu and Map , M-report and Menu, JD-DTW and HMI integration</li><li>•after noon: JP-report, M-Users menu, JD-DTW refinement</li></ul>
Tuesday 22	<ul style="list-style-type: none"><li>•morning: JP-report, M-Option menu and report, JD-DTW refinement</li><li>•after noon: JP-Winning declaration, M-report, presentation and obstacles, JD-DTW refinement</li></ul>
Wednesday 23	<ul style="list-style-type: none"><li>•morning: JP-tests, M-report and tests, JD-tests</li><li>•after noon: report and presentation</li></ul>

M = Marie / JP = Jean-Paul / JD = Jean-Dominique

## 2. References

- J. Leroux « Dynamic time warping », « HMI », TNS
- Wikipedia
- J. Mariani « Advances and trends in automatic speech recognition », p245-252, 1990
- ...

## 3. Intended plan:

- Sound processing
  - Sound recording (complete word)
  - Auto-correlation
  - Time split in slices of 20/30 ms, for all times
  - Fourier transformation for all slices → put together in a spectrogram (matrix)
- Learning (problem of Bakis / Hidden Markov/ Dynamic Time Warping)
  - Frequencies scaling: 20 significant points (Mel scale → assign importance to some frequencies)  
Not compare the 1<sup>st</sup> with the 1<sup>st</sup>, it could exist some needed translations or insignificant isolated elements.
  - Comparison: « compare most similar words », eventually if we have a great vocabulary, begin with a leak sort.  
ex : in French, 6/10  
S-I-S ← delete the similar parts  
D-I-S  
↑compare relevant part
- HMI
  - Graphic Interface: display word (and the spectrogram)
  - « Purchase » Learning : buttons « ok »/ « not ok » and we put the syllable in the database
  - Labyrinth « game »: we move with our voice  
/ !\ learning the words the gamer will use during the « game »

## Table of figures

Figure 1 Example of signal: « gauche, droite, gauche, droite » .....	5
Figure 2 signal without word beginning .....	5
Figure 3 signal with a late word beginning.....	5
Figure 4 signal with a good word beginning.....	6
Figure 5 time split example .....	6
Figure 7 Fourier transformation example .....	7
Figure 6 Hamming window.....	7
Figure 8 Fourier redundancies elimination .....	8
Figure 9 Mel's scale .....	9
Figure 10 filter bank .....	10
Figure 11 spectrums before and after Mel's scale .....	10
Figure 12 spectrogram example.....	11
Figure 13 Fourier transformation spectrogram .....	11
Figure 14 Mel's scale spectrogram.....	11
Figure 15 frequency characteristic.....	12
Figure 16 Artificial neuronal networks .....	13
Figure 17 hidden Markov model .....	14
Figure 18 DTW displacement .....	15
Figure 19 Minimal path examples .....	15
Figure 20 DTW Improvements .....	17
Figure 21 Beginning detection example .....	18
Figure 22 DTW local -- DTW local and beginning detection -- DTW median and beginning detection -- DTW global and beginning detection indices comparison .....	19
Figure 23 DTW local -- DTW local and beginning detection -- DTW median and beginning detection -- DTW global and beginning detection distances comparison .....	19
Figure 24 comparison of methods table .....	20
Figure 25 first menu .....	21
Figure 26 choice of the user menu.....	21
Figure 27 Change user menu.....	22
Figure 28 option menu .....	22
Figure 29 labyrinth example .....	23