MULTIMODAL TOOLBOX:

Analyzing gestures

Internship report of SLT Carré and SLT Levasseur
Promotion CNE Beaumont (2005-2008)

An assignment of

13274 Fiji Way
Marina Del Rey, CA 90292
United States of America

September 17th 2007 – December 6th 2007

Authors

SLT Carré David david.carre@st-cyr.terre.defense.gouv.fr
SLT Levasseur Marco marco.levasseur@st-cyr.terre.defense.gouv.fr

Supervisor

Jonathan Gratch gratch@ict.usc.edu

Institute for Creative Technologies
13274, Fiji Way
Marina Del Rey, California 90292
United States of America

Tutor

Eric Jacopin eric.jacopin@st-cyr.terre.defense.gouv.fr

Ecole Spéciale Militaire de Saint-Cyr
Division Générale de l'Enseignement et de la Recherche
56381 Guer
France

Quick Summary

Table of contents

Abstract

Rapport between people and virtual human agents is not limited to just speech. There are many non-verbal behaviors such as gestures or facial expressions that can express feelings or convey a message. One of the challenges in making an agent appear more realistic is to make his non-verbal behaviors appear more natural. To accomplish this, it is essential to find out how and when gestures are performed.

In order to determine how gestures are performed, it is necessary to assess different appearances of the same gesture and the mapping between their respective function.

To determine when gestures are performed, the key is to find relevant contextual features and their links with gestures, which will lead to the prediction of the moment they should be performed.

Finally, both of these issues can now be tackled with the provided toolbox. Preliminary results show that we have some gesture pattern. Beside, we were able, based on contextual features, to predict when the agent should nod his head. Early results appear to show the agent nods at an opportune time. Moreover, this toolbox generalizes the results to other kind of gestures than head nods, which is the goal of this study.

Introduction

The University of Southern California (USC), Institute for Creative Technologies (ICT), is a University Affiliated Research Center (UARC) sponsored by the U.S. Army.

The mission of the ICT is to build a partnership among the entertainment industry, Army, and academia with the goal of creating synthetic experiences so compelling that participants react as if they are real. The result is engaging new immersive technologies for learning, training, and operational environments.

In order to accomplish this goal, the ICT has many research departments such Speech Recognition, Graphics Animation, as well as the Virtual Humans project. The aim of the Virtual Humans project is to create realistic interactive training environments. Currently there are two scenario implementations of this project which are Stability and Support Operations - Simulation and Training (SASO-ST) and Stability and Support Operations - Extended Negotiation (SASO-EN).

One of the objectives of the Virtual Humans-Emotions group is to build rapport between people and agents. This can be enhanced by having the agent exhibit natural, human-like behaviour such as posture shifts, varying eye gazes, head shakes or head nods when responding to the human participant's actions.

Our mission here was to improve the head nods performed by the agent and to be able to predict them through a link with context features. We had to do it in order to make them happening at a suitable time and to classify them by their functions. This will give the agent a new dimension by the diversity of its gestures.

This report represents an analysis of the work during our three-month internship and documents how to use the toolbox. The report aims to expose the way took for the preparation of data for their later use. It also presents the different tools used for enhancing the head nods of the agent and gives an overview of the work that can be done to improve other non-verbal behaviours.

Chapter 1 – Overview of the project

1.1 – Introduction

1.1.1 – Goals

In order to reach the ICT's virtual humans research group objective, the agent had to give more realistic response behaviour. Initially the agent did not give the good responses at a good time and most of the time the agent seemed to be giving random feedback to the real human he had in front of him.

The main goal was to improve the feedback of the virtual human, starting with the head nods. To reach this goal, we developed a toolbox to enable to analyze the gestures in order to find out the differences between head nods and their respective function, but also with the help of context features, allowed us to find a suitable time for the head nods to happen.

This toolbox supports rapport project goals by providing tools to elicit, analyze how gestures are performed and predict when they occur.

1.1.2 – Specifications

This Toolbox had to meet several requirements:

- **Easy to use:** To facilitate the use of the Toolbox by future users, the overall system has to be well documented.
- **Adaptability:** The Toolbox has to be compatible with other kind of data. In the future, it will be used later to study other features of the agent's behavior like head-shakes or posture shifts.
- **Intuitive:** Visualization helps to give people some intuition about the results. Sometimes, it even permits to confirm them.
- **Optimized:** Some functions take a while to run, because of the size of the data used, or the complexity of the function itself. Given the number of experiences that has to be done, it is worth to improve these functions

1.1.3 – Working Language: MATLAB

The use of MATLAB (MATrix LABoratory) Software became necessary since our work was complementary to some functions developed by Dr. Morency. It is easy to create graphical results with this software. Since it was a specification of our task, it became natural to use this language.

MATLAB is also very convenient to use when importing text, arrays such as Excel sheets or even XML files. Although we previously used C++ Visual Studio for several projects, MATLAB appeared easy to handle and quick to understand with some programming bases.

The main distinction between the elements treated by this software is their structures. Indeed a grand number of elements can be considered as a part of a matrix. A grand number is also handled as a 1-by-1 vector.
Nevertheless, it appears that sometimes a gathering of many elements in the same structure is needed. The data we had to treat are often constituted by a number of elements which could be of various types. If all the data in a same structure have the same length, matrix structure can easily be used to carry these information but problems appear when these data do not have the same length. Actually one element of the data set can possess more information than another it is why we have to use the cell array structure. A cell array is a container which can carry more than one type of data. A cell array can also contain a text array as well as a matrix or complex vectors. The fact that this structure can hold different objects with different shape and size allows working with all the data types used during the non verbal feedback studies.

1.2 – Context

1.2.1 – Working group

The Virtual Humans research team encompasses several projects which are all supervised by Dr. Jonathan Gratch, Associate Director of Virtual Humans Research and Lead Research Scientist of the Emotions group.

In our research team, which was named non-verbal gesture recognition, Louis-Philippe Morency is the project Leader, so we had to report to him updates about the job done during the week. Moreover we kept him informed in case of major problems concerning the precise aim of our study, in order to avoid wildness.

1.2.2 – Coordination

Nearly every day we had a meeting with Louis-Philippe Morency to check our progress, and to reposition it on the global study. It allowed us to work on the subtasks and to keep us on track. We also had general briefings to help us anticipate possible problems we could encounter, along with moving on to other tasks if we finished earlier.

Twice a month, Dr. Gratch checked the status of our work, and we participated in a general weekly meeting with the SASO team.

1.2.3 – Working environment

In ICT, all is put together to create a favourable working environment. You can connect with your laptop to the enterprise wireless network and access the data on your workstation or on other servers you are working with. Everyone has a powerful workstation connected to the Internet. An Information Technology (IT) department is at your service if you have any problems with your machine, from installing software to solving networks problems. This is greatly appreciable when your mind is concentrated on your job, you do not want to have to deal with minor disappointments.

This workplace was definitely very enjoyable and allowed you to concentrate on your work.

1.3 – Work organisation

1.3.1 – Tasks subdivision

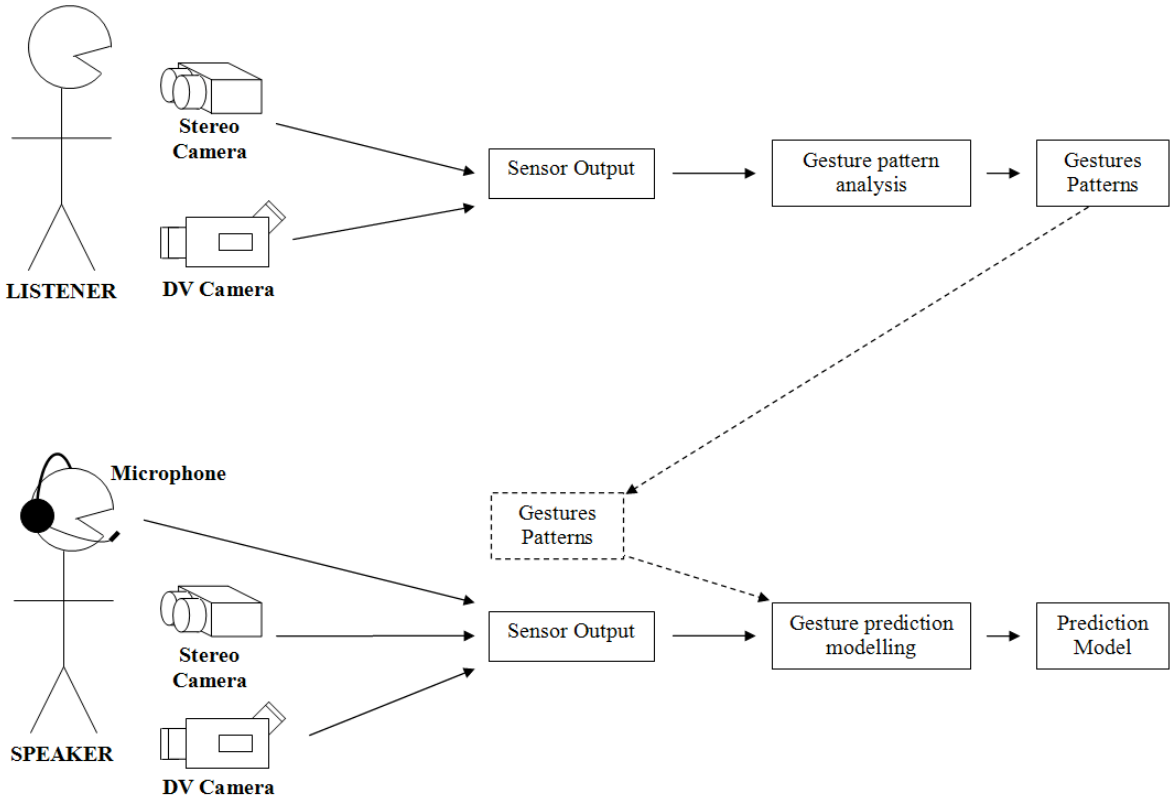
In fact, although our main goal was to give the virtual human more realistic behaviour, this task was given under two sub-tasks, that each of us tackled during our internship.

The first task was to create a gesture pattern analysis toolbox. This toolbox has to be able to identify and to classify the different types of head nods, and to identify the functions of the found types of head nods.

The second task was to create a gesture prediction modelling toolbox which has to predict and to create a model for when head nods should happen.

The link between these two tasks is illustrated below in Figure 1-1.In fact if a gesture pattern was revealed, it could greatly improve the work of the gesture prediction modelling toolbox. Indeed the gesture pattern could be used as an input for it, in order to create a model for each kind of gesture.

The dotted line shows that the use of gesture patterns as an input of the gesture modelling toolbox has not been done for the moment, but that could be a task for future research.



1.3.2 – Tasks repartition

Since SLT Levasseur worked most on Watson files, which contain the head velocities, it appeared natural that he would deal with the analysis of gesture patterns.

The gesture analysis toolbox has also the purpose of finding and selecting different types of samples in the velocity data set. These selected samples have to be encoded to allow the work of a partitioning function which gives different clusters. The clusters will be visualized thanks to MATLAB plotting functions.
Improving the Predictive toolbox involves that the different selected samples possess a type label and that the clusters allow finding different functions for these types.

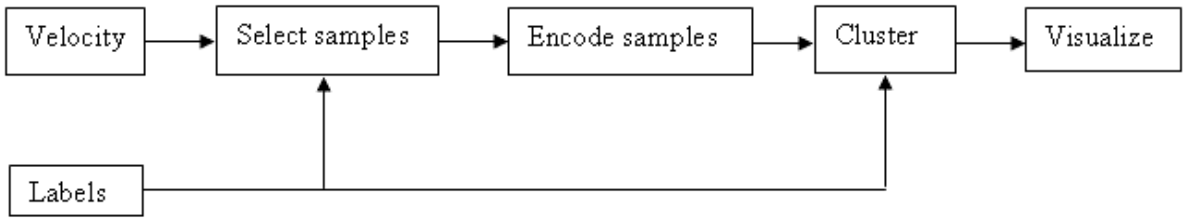


Figure 1-2: Gesture analysis toolbox components architecture

Moreover, since SLT Carre has been working on Transcriber files and started to have some intuition about when head nods should happen, it was him who tackled the Prediction toolbox part.

This toolbox represented on Figure 1-3 has to allow the selection of the features which will be relevant for the gesture modeling. These features have then to be encoded to fit as an input for training of models. In order to see and compare the results, visualization tools are finally needed.

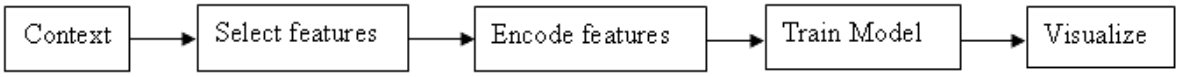


Figure 1-3: Prediction toolbox components architecture

1.3.3 – General toolbox use

The use of the Multimodal toolbox can be enlarged to the study of other non verbal behaviours such as eye gazes or head shakes. The way we built the global toolbox and the way it has to be used for future studies are resumed in Figure 1-4.

Each task implies many subtasks. First, before splitting work, a common task is to improve data, extracting them to be usable with MATLAB and organize them inside MATLAB. Moreover the creation of a ground truth gesture basis (explained in section 2.1) is necessary to keep a link between this study and real gestures.

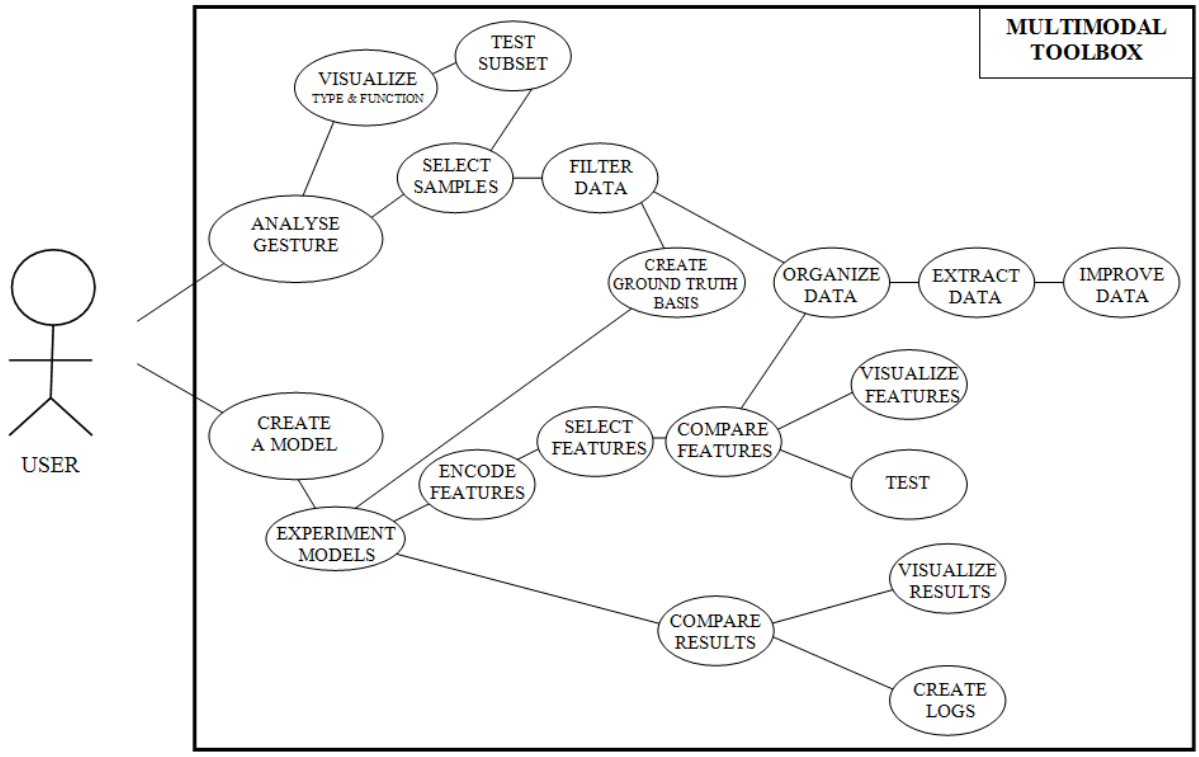


Figure 1-4 : Use case diagram of the multimodal toolbox work architecture

The upper part of the diagram, the gesture analysis toolbox, is devoted to the finding of gesture types. This action could be divided in a visualization task and in a selection task.

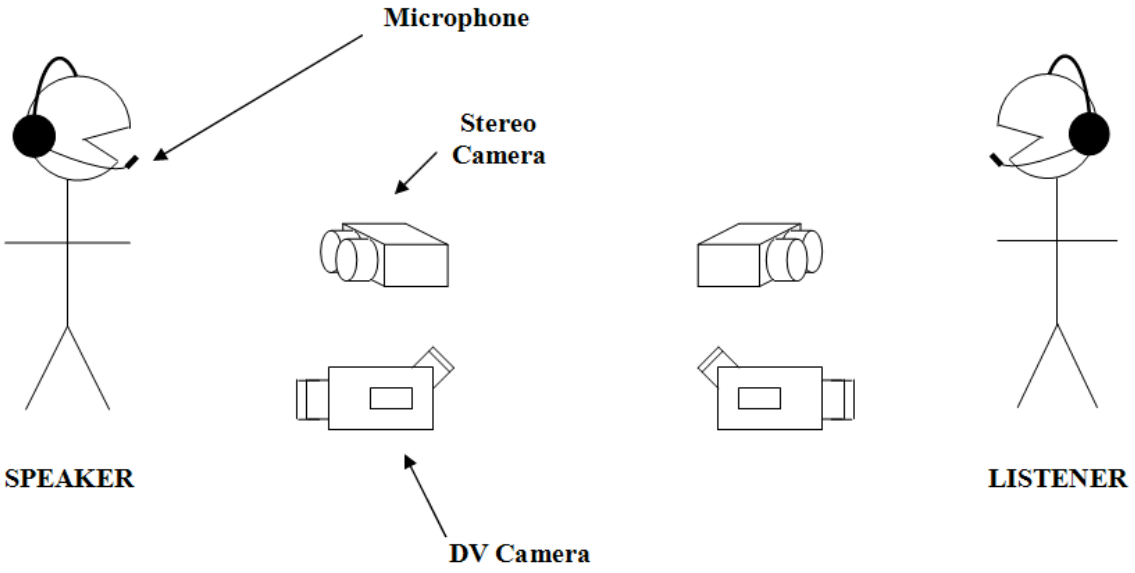
The visualization can be done after the test of different composition of elements from the data set. Nevertheless these tests as well as the analysis of the type of gestures cannot be done without a selection of the information after their filtering.
The goal of the filtering task was just to remove the interesting elements from a data set which was since organised after its extraction. By interesting we mean that these elements must be correlated with the ground truth basis.

Concerning the lower part of the diagram, the prediction toolbox, the main task could be separated in two major subtasks. First subtask is to select relevant features to be implemented in the model and their encoding. This can be done by visualizing some statistics about the features, and by testing them.
Second subtask is to experiment different models by comparing their results. In order to do that, creating visualization tools for these results, as well as creating logs of our experimentations are necessary, to be able to do these in a methodological way, and to keep a trace of previous results.

Chapter 2 – Data Acquisition

2.1 – Original user study

2.1.1 – Principles



The data used in our study come from a user study realized in October 2006. Even if there are many variants of this test (see Appendix A), we will keep our focus on the Face-to-Face test (see Figure 2-1) since this was the one during which most head nods appeared.
Principle is quite simple:

a) The Speaker watches two video dealing about sexual harassment in the workplace.

b) His goal is next to tell the story to another person as detailed as possible after a bip that tell him when to begin. This element appeared to be very important in our project since it was a way to synchronize all the feeds received during this exchange.

c) At the end the listener has to tell the story again to check he had understood well enough the story.

One of the goals of this user study was to record different body behaviour during this oral exchange. Each different recording constituted a sequence in itself.

2.1.2 – Materials

The part that was interesting for this project was the exchange one. Indeed a lot of information was recorded during this part, using many different tools:

-The microphone records in .wav files all the speech of the Speaker and the Listener. These audio files will be then used to annotate the words and the prosody.

-The camera records in .avi files all the video feed (including sound) of the Speaker from the chest to the top of the head. The same thing is done with the Listener. On the speaker side, the video will be used to annotate eye gazes, whereas on the listener side, they will be used to annotate head nods.

-The stereo camera, records in two different .avi files (Left and Right side of the binocular camera) the same part of Speaker's and Listener's body than before, thus at a rhythm of 8 frame per second. This will be used later to compute head velocities.

<table>
<tr><th colspan="2">SPEAKER</th><th colspan="2">LISTENER</th></tr>
<tr><th>Material</th><th>Information</th><th>Material</th><th>Information</th></tr>
<tr><td>- DV Camera
(*.avi files)</td><td>- Eye Gazes
ELAN</td><td>- DV Camera
(*.avi files)</td><td>- Head nod Grown Truth basis
ELAN</td></tr>
<tr><td>- Stereo Camera
(*.avi files)</td><td>- Posture Shifts
WATSON</td><td rowspan="2">- Stereo Camera
(*.avi files)</td><td rowspan="2">- Head Motion
WATSON</td></tr>
<tr><td>- Microphone
(*.wav files)</td><td>- Context features (speech, prosody)
TRANSCRIBER</td></tr>
</table>

Table 2-1: Available materials

Table 2-1 sums up the different sensors, their output, the information coming out from this data, and in capital italic, the software used to extract (Watson) or annotate (ELAN, Transcriber) these informations.

2.1.3 – Tests description

Only two of the four variants of this test happened to be interesting in the case of our study. The first interesting one was the Face-to-Face (see Figure 2-1) because it was in these sequences that occurred most of the noticed head nods. It seems natural since you give more feedback to a real human than a virtual one.

The second one was the Mediated one. In fact in this test, the only difference is that instead of having the video feed of the listener in front of him, the speaker has the agent in front of him. This means less head nods than previously for the listener because the speaker gives less facial expression to an agent. Therefore, the listener is indisposed to give feedback to the speaker which is less emotive.

2.2 – Data Processing

Before working each one on our side, we had to accomplish together some tasks in order to prepare our both studies. The raw data had to be improved and converted to a MATLAB format.

2.2.1 – Studying video files

One of the first tasks that had been given to us was to study listener's video sequences in order to find there head nods. Indeed this was crucial because all the future work will be based upon it. This is what we call our ground truth head nod basis.

The first step is to note down every head nod of each sequence. In order to realize this thing we have to watch all the video sequences separately and to make your own notes for every video. We considered that a head nod appears at each time the head makes a motion from the bottom to top or inversely. It appeared that a head nod could be mixed with other movements which complicated the study of several of them. Afterwards we had to compare the head nods we found to be sure we had not forgotten one of them. If we had any doubt about some of them we watched the video again and together to be sure of its existence. It occurred that some sequences did not include any head nods. Then, these video were dropped from our study. That reduced considerably the amount of sequences available for our task, going from 48 sequences to only 22. All the remaining video sequences are sequences which possess at least one head nod and which have been examined twice. These elements constituted the future ground truth head nod basis.

Once we agreed on the presence of head nods or not, we had to locate them temporally. The remaining part, into we could find 208 head nods, was split in two parts since it will allow us to be more efficient. SLT Levasseur studied video sequences indexed from number 2 to 97 when SLT Carre studied video sequences indexed from number 98 to 116. All the results of this work have been summarized in an Excel sheet called Summary. This survey has allowed realizing several averages about the studied video sequences since it carries information such as length of each head nod, number of head nods in a sequence and whether the head nod was made during speech or not.

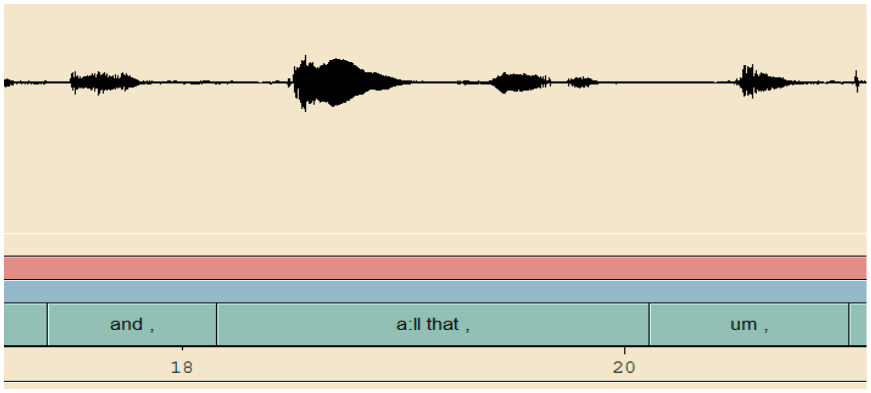
The last step was to annotate precisely (meaning to the frame) these head nods. This was a huge task, time speaking, which has been made easier by the use of annotation software called ELAN (see Appendix B). This Eudico Linguistic Annotator was meeting our requirements since it could display video feeds, go from frame to frame and supported the different video types used in the study. In

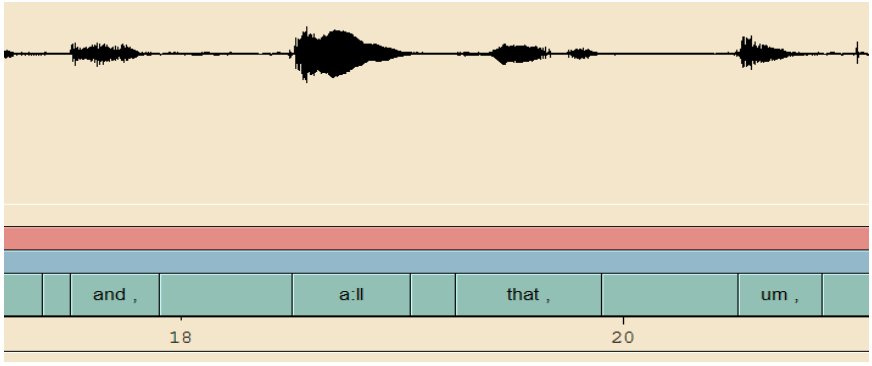
the purpose of extracting just the head nod we had to consider the moment where the head really began to move. We had already found the approximate time of it but thanks to the frame to frame tool of ELAN we were able to pick precisely the motion.

2.2.2 – Improving Transcriber annotations

To study the head nods and their context, one of the needed features is the speech of the speaker. Indeed, it was flagrant that listener's head nods came not only in response of facial expressions or body language, but also as a feedback to the story he was listening to.

All the speech being recorded by the microphone in .wav files had already been annotated by some linguists of the ICT. This was done by the help of a software named Transcriber (see Appendix B), developed by the DGA (Delegation Generale pour l'Armement).





However some key features were missing to these files. First step was to add to these 22 files the pauses they lacked. Indeed we predicted that there was a link between pauses and head nods, which will justify the 10 hours spent on adding them in these files. Moreover adding the pauses will help us to locate

temporally the words into the utterances more precisely without needing to add a timestamp to each word, which would have been a waste of time. Transcriber revealed to be quite efficient for this job since we could see the waveform of the audio file, which was very useful to detect pauses.

The second point was a lack of consistency in these files. For such study, this was a requirement, and same events happened to not be annotated the same way (see Appendix C). Jillian Gerten, the team's linguist, had to go through all these files again to be more precise with punctuation and speech annotation.

2.2.3 – Reprocess video files with Watson

The two video produced by the stereo camera (left and right views of the same person) constituted the input of software called Watson (see Appendix B). This one, developed by Louis-Philippe Morency, permits to track head's position and orientation.

The results of this tracking applied to the video sequences already existed but they were not enough accurate for the study. Dr Morency has thus adjusted the parameters of its software to heighten again the values in output.

Then, one of our tasks was to restart Watson's work on these video in order to obtain useable results for the future study.
In the purpose of realizing this work we had to create a new file for each video sequence which will contain the video on the .avi format filmed by the camera and the configuration files for WATSON. These files contain diverse information such as the name of the studied sequence, its length or the path to save future work (see Appendix B). After the creation of these different elements we had to launch again software to obtain more accurate data.
This last task was launched on different machines to allow us continuing others tasks on our machine, thus for a question of time optimization.

The processing of these video turn out different results such as files which contain the values of the velocity, the rotation (around each axis) and the index of the time, called timestamp in the rest of the report. The timestamp is the time of the recording of an action. It has not a continuous rate. Dr Morency has created software which put these entire figures directly in column in a word publisher. This last thing facilitates the treatment during the data extraction (see subsection 2.3.2). A video, where the head of the listener is framed by a square which tracks the movement, was also an output of Watson software as parameters files. The parameters files contain all the information needed to launch a Watson tracking plus all the elements describing the video sequence such as the number of frames of the video sequence.

But before going ahead, we had to verify the accuracy of this software by calculating the number of true positive rate and false positive rate that Watson found. A true positive rate means that it is detected by the software and corresponding to a ground truth head nod found previously. It can also be seen as the precision of software. The false negative rate means it is detected by Watson but is not a ground truth head nod. It is most of time considered as the error of software. To define these values we had to give a threshold value to define it there is a head nod at a certain time. The threshold is just a numeral value that the head velocity according to x has to exceed to be considered as a nod.

To verify if the accuracy of this software is really good we can not just use the output figures of the function which calculates the number of right detection and the others. We have plotted these values in a ROC (Receiver Operating Characteristic) curve (see Appendix D) which permits the study of the specificity and the sensibility of a test according to different threshold values.

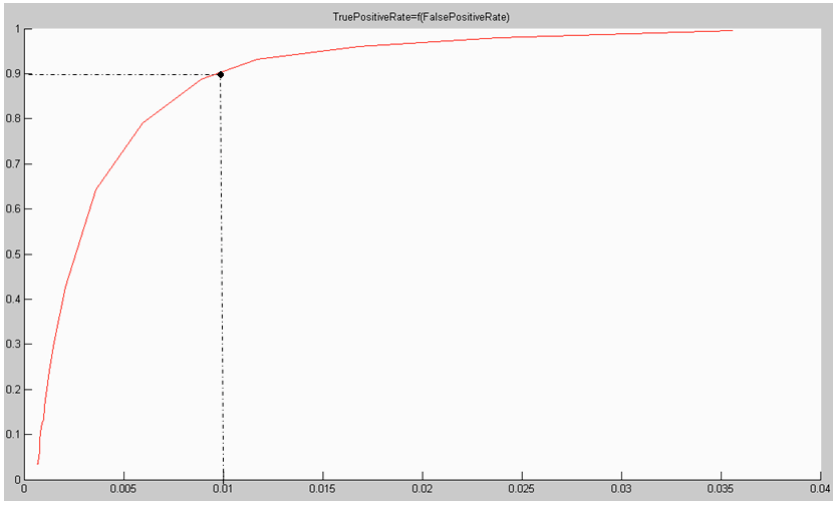


Figure 2-4: ROC curve

The X-axis represents the value of the False Positive rate whereas the Y-axis represents the True Positive rate. We plotted this curved with a threshold included between -2 and 2 with a step of 0.1.

The examination of this curve proves that Watson software is reliable and then we can really found our study on its outputs. For example the point on Figure 2-4 shows that for an accuracy of almost 90 percent software makes an error of 1 percent.

2.3 – Importing Data with MATLAB

2.3.1 – Transcriber files

In order to use words, context and pauses as features, we had first to import them from .trs files (Transcriber) to a MATLAB file, containing each utterance, their beginning and their end. It turned out that .trs files were in fact XML files making it easier to find the needed data. The data was imported as a text sheet and thanks to the markups, the data was easily found, with its associated time stamp.

First step was to locate markups that will be useful for us in order to understand where to find data. In fact there is a markup for each start of a new segment:

```
<Sync time="4.438"/>

<Sync time="4.66"/>
um ,
<Sync time="5.131"/>

<Sync time="5.519"/>
```

It contains the timestamp of this segment. Between each markup are the sentences contained in each segment (see Figure 2-3).

Final step was to organize extracted data. Two variables contained all the information: a cell array contained by sequence a matrix where begin and end timestamps of segments constituted the columns and the rows were corresponding to these segments. Another cell array contained by sequence a character array where each row corresponded to a segment.

2.3.2 – Watson files

The extraction of the data produced by software Watson constitutes the cornerstone of all the head nods study. Indeed the figures of the head velocity and of the rotation will serve for the characterisation of the different categories of head nods.

The purpose of this part was thus to product a database which can be easily used in the rest of the internship and even for other users.

MATLAB was a fair tool for this work because it permits the importation of data from publisher which product results in rows or in columns. It is why the work of Dr Morency was very important forasmuch as all the outputs of this software were directly in columns.

We focused on the x axis rotation values because it is them which are interesting for the head nods but all the other values were extracted too for a future study of head shakes or others. These entire values are contained in velocities.txt and nods.txt files which are two of the multiple output files from Watson.

The timestamp treatment consists on a transformation of these values in seconds and to bring them into a zero set bases. It was free to put these data in a cell array or not but to facilitate the future treatment per session and to be portable to the Dr Morency since existing functions we adopted the cell array.

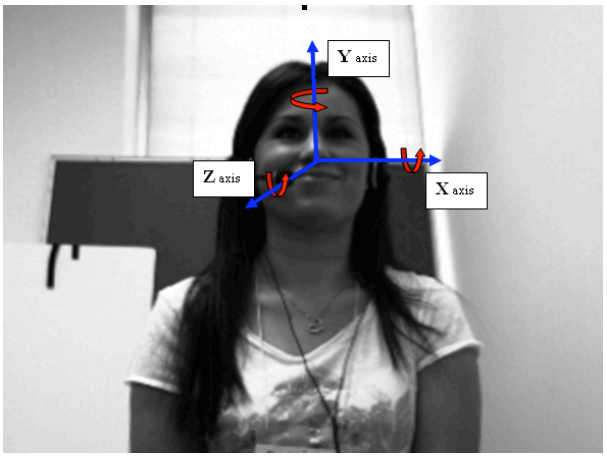


Figure 2-5: Axis references in Watson files

A point that appeared later in the study is the needed of tighten these values to the so called real head nods. The real head nods correspond to the ground truth head nods (see subsection 2.2.1). A function was thus created to find the head nods rotations values, the timestamps and the head velocity which tally with the head nods found in common.

2.3.3 – Elan files

Elan files were in fact, just as Transcriber files, XML files. However, and although it was basically markups for begin and end of annotations, we did not initially created a MATLAB function to extract these files into a MATLAB variable format. Since both of us were working on these files, we preferred to do it manually, which was faster than creating a function to do it automatically.
However it appeared later to be a wrong choice since in the next studies, the time we could have lost in creating this function will be a gain for them.
This is why, in order to keep up with our initial specifications, it has been decided at the end of the internship to fill this lack.

First step was to select automatically the files corresponding to the study.

Then, since these files were XML, we imported them as a text file. Just like Transcriber files, first step was to find out which markups could help to locate the data we were interested in. This is a sample of the syntax of markups that contains begin and end timestamps of two annotations:

```
<TIME_SLOT TIME_SLOT_ID="ts1" TIME_VALUE="21443"/>
<TIME_SLOT TIME_SLOT_ID="ts2" TIME_VALUE="22813"/>
<TIME_SLOT TIME_SLOT_ID="ts3" TIME_VALUE="26680"/>
<TIME_SLOT TIME_SLOT_ID="ts4" TIME_VALUE="27470"/>
```

In fact impair time slots correspond to begin timestamps and pair time slots correspond to end timestamps.

Last step is to organize the data extracted. An array, where rows corresponded to each annotation and columns corresponded sequence number, begin and end time of each annotation was finally created.

2.3.4 – Elvin reports

Elvin is a software that allows different programs to send messages between each other [8]. It was useful for us because it created some log files that contained speech analysis outputs. That added some context features to our study, which could be clues for detecting when a head nod should happen.

The organisation of these log files made them easier to import, since the format was consistent: Each line started with the time stamp of the action in milliseconds, and at the tenth column came the caption of the speech analysis output.

2.4 – Conclusion

Even if this part was not the trickiest, it came out to be time consuming and crucial for the remaining tasks. First the creation of a head nod ground thruth

basis served as a foundation for all the following work since it will be put in nearly all MATLAB functions created in Chapter 3 and Chapter 4.

The events cell array which came from the Transcriber files has also been used in the following parts of our study either as a words database or as a speech context.

However, we had to came back later on this part in order to fix some issues caused by the way original user study was done. Indeed all sensors did not start to record at the same time for each sequence, which forced us to align them with offsets. ELAN helped us in this task since it can be run on a “Synchronization Mode”. All offsets were then put into a variable which contained each offset by media and by sequence. The reference we took were the listener video since we started from here with the ground truth which was the basis of all this study.

At last, this first part allowed us to become familiar with MATLAB, handling every kind of variables and discovering possibilities to optimize the code. This including avoiding for loops when it was possible since MATLAB was not meant to be efficient for loops.

Chapter 3 – Gesture Analysis Toolbox

3.1 – Introduction

3.1.1 – Goal

People perform gestures with different amplitude, speed, and length. The different gestures patterns can also have different functions. So this section shows how to find these different patterns of a gesture and find out if these patterns are related to different functions.

The human being has different way to walk. Indeed we can walk more or less fast, with more or less big amplitude or frequency. All these motions can also be divided into different groups. What it means it that the same form of division can likewise be done with the head motions such as eye gazes, head shakes or head nods. We will focus more precisely on the head nods movements but the entire study could be adapted for the works on other motions.

The first goal of this part is to produce a tool that can help improve non verbal behaviour generation by the Virtual human. In fact for the moment some motion realized by the Virtual human are not always realistic. It means that the agent produces some feedbacks not always at the right time and not always as human do it. It can for example make a gesture with too high amplitude compared with that people usually do.

To obtain an improvement of the Virtual human non verbal gestures we have to search and identify different patterns of head nod motions. First of all the existence of these patterns must be underlined. We can easily ascertain that there exist several differences between the human head nods at the time of a talk. But it is harder to explain these differences.
These differences could be visual:

- The speed of the head motion can increase or decrease.
- The amplitude can be more or less high.
- The frequency can also vary during the motion

Or even be found by the composition of different factors. That means the use of mathematical tools to discover hidden correlations.

Second, we had also to find out if these patterns are related to specific functions such as:

- **Continuation**: The listener wants the speaker to keep talking and may not have understood yet all the meaning of the last spoken utterance.
- **Acknowledgement**: The listener understood the meaning of the last utterance but may not agree with the speaker. The head nod is equivalent to a spoken “ok”
- **Agreement**: The listener understood and agrees with the last spoken utterance. This head nod is equivalent to a spoken “yes”.

3.1.2 – Correlation: appearance & function

The entire work describes higher has to provide the elements which permits to correlate the diverse appearances, called gesture pattern, of the heads nods found in the video study with different functions. It will permit in the future to improve the agent behaviour because all its head gestures could be adapted to the context of the discussion and it will yield a Virtual human sensitively more human. If this task was not done the agent could realize different motions but these one would not be in correlation with the human behaviours that could cause some troubles for the speaker.

3.1.3 – Approach

The continuation of this section is divided into four different parts:

The first one will explain the different steps we took to find the right representation of the different gesture patterns. It means that a data can be considered according to different criteria and thus there are several manners to represent it.

Second we cluster all these different representation to underline the fact that the gesture patterns can be represented thanks to a typical pattern. We show there that these gesture patterns possess an average in there representation.

Third we compare the prototypical patterns found previously with the speaker speech to show the different correlations between the different gesture pattern and the fact the human is speaking. To realize the comparison we use visual tools which are implemented in MATLAB software.

Finally we create a User Study function. This one permits to find out if the patterns are related to functions. This User Study is build on the basis of a website infrastructure.

3.2 – Data representation

The purpose of this part is to organize and to analyze the data gave by Watson software to facilitate the clustering part. One of the software outputs is the velocities files which contain the entire information on the head nods. We have also built a ground truth basis and we want to find the different gesture patterns. In this purpose we try different representations such as the velocity, frequency or PCA to extract these patterns.

3.2.1 – Velocity & length

The first step is to consider the raw original information such as velocity and the head nods segmentations. We thus made different functions which can isolate the velocity values and the length of elements interesting us. We mean that we just considered the figures of the ground truth head nods. The way we extract these elements and there duration is detailed more precisely in the Data Processing part.

During this phase we encounter two main problems with the velocity data. Indeed the entire values of velocity have to be resampled and to be aligned.

Before starting all the other studies we had to bring the entire temporal values on the same basis. In fact every recording was made without some synchronization with the other. We mean that each recording, wave or video (see Chapter 2), has it own temporal beginning. To solve this problem we took all the different recording and decided to choose a unique beginning for them. We took the end of the “bip” sound played at the beginning of each sequence to realign all the recording with ELAN software.

The other point is that Watson software stores the different values of head velocity or head nods with none-constant frame rate. This problem was more important because the clustering and the use of Fast Fourier Transform explained in future parts need a constant frame rate. The way to fix this problem was thus to resample all the time line of each video sequence at the same sampling rate. The first intuition was to calculate the mean of these timestamps but in fact these terms had not a straight-line rate even in a same sequence. Thereof we had built a function which is able to resample all the sequences even if there had not an unchanging time rate. The important fact is that this function can resample these timestamps at a rate the user can choose to permit a big range of study.

3.2.2 – Frequency

To discover if there are really different classes of motion we can not just limit the study to the values given by Watson software. We had thus to consider these source of information according to a new way of regarding such as the

frequency. The calculus of the Fast Fourier Transform (FFT) gave us one of these new perspectives. [2]

The FFT is just a mean to accelerate the calculation of the Fourier Transform. In fact the complexity of the Fast Fourier Transform is on O(n ln(n)) whereas the complexity of the Fourier Transform is on O(n^2) for a sample of n elements.

$$y_p = \sum_{k=0}^{n-1} x_k \left(\cos\left(2\pi \frac{kp}{n} \right) + i \sin\left(2\pi \frac{kp}{n} \right) \right)$$

Figure 3-1: Mathematical expression of the FFT

The Fourier Transform which is given for an x input with unchanging time rate allows us to find for each frequency the value of the associated energy. To find this term we had just to consider the module of the Fourier Transform or FFT output. Figure 3-2 represents the FFT absolute value calculation of a particular head nod. It permits to visualize the difference between different gesture patterns thanks to there frequencies.

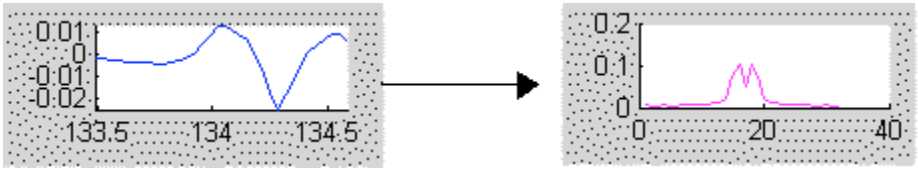


Figure 3-2 : Extraction of frequency from velocity

We can also plot this module according to the frequency to obtain the frequency spectrum of the studied sequence.

The calculation of the different head nods frequencies can permit to check if the gesture patterns are differenced by their energies or their frequency spectrum. We have to consider a window (Yellow in Figure 3-3) around the central value of the head nod because the future studies such as the clustering part and the PCA part need that the entire values have the same length. It means that the clustering can just be done on a matrix. This value was just taken at the half of each nod and we added at this time a duration chosen by the user.

We have also to make it because the dimension has to be reduced to facilitate the future clustering. It is why we consider a 32 elements vector for each head nod in the rest of the study.

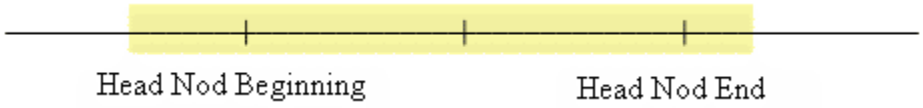


Figure 3-3: Window around the head nod

3.2.3 – Principal Component Analysis

The Principal Component Analysis (PCA) is a technique used to reduce the dimension of multidimensional data. This tool is based on applied linear algebra and uses non-parametric method to extract important information from confusing data set. It is a fine tool for making predictive models and for all the data analysis. Thus it permits to reduce a complex data set to understandable and more visible information. This technique which used linear transformation permits to find a new coordinate system. The new coordinate system will be defined by the projection direction of the greatest variance of the data for the first axis and the second greatest variance of the data set for the second axis.

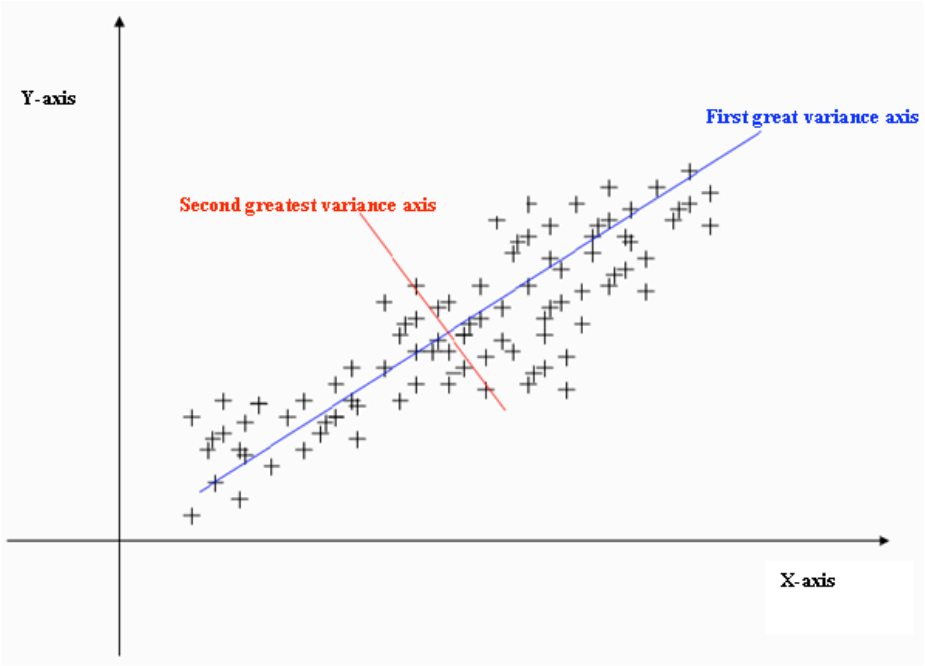


Figure 3-4: Example of points' projection on first greatest variance direction (blue) and on second greatest variance direction (red)

Find a new basis permits to keep just the values of the data set which have the more important variance. Thereof it reduces the length of the data set without losing elements which have a significant weight for the rest of the work. We used a mathematical technique called SVD (Singular Value Decomposition) to do this decomposition. The choice of this tool is justified by the fact its use is wide-spread for this sort of work and because it is since implemented in MATLAB.

- Singular Value Decomposition

The Singular Value Decomposition (SVD) allows us to do a decomposition of the information we wanted to treat. It means that SVD calculates the singular values and singular vectors of the input. The input of this function must be a matrix

where the number of rows is upper than the number of columns. For this study we had utilised the frequency of each sequence of ground truth head nods because we wanted to know if the different classes of motion are identifiable by a particular value of frequency. For example we can imagine that one of the gesture patterns is defined by a frequency of 4Hz motion at a particular moment. We considered thus a matrix per sequence which comprises the frequency value of each resampled head nod.

The SVD function applied on an m-by-n matrix gives us three different elements in output:

- A diagonal matrix of the same dimension as the input matrix. This matrix, called S in MATLAB, comports all the singular values. All the elements of this matrix are null except the diagonal.
- An m-by-m U matrix which contains the left singular vectors.
- An n-by-n V matrix which the right singular vectors.

In our case we considered a matrix which is constituted by 198 rows and 32 columns. The number 198 corresponds to ground truth head nods found previously and 32 is just the window length that can be chose by the user. So each row of the matrix is dedicated to a single head nod.

The elements of the S matrix present the singularity to be ranged in decreasing order. The first singular value has thus the most important weight in the decomposition. We used this last property to lead the rest of the study. In fact we wanted just to consider the head nods elements which are the most important. By important we mean that if we not consider the terms with a little weight, it will not greatly change the information carried by the head nods.

The elements of the V matrix constitute the coordinate of direction vector. These vectors, in red and in blue on Figure 3-4, permit to find the direction where most of the data set is scattered.

The terms in the U matrix can be defined as being the coordinates of initial elements in the new basis made by the vectors of the V matrix.

As it is explained previously the elements of the S matrix are arranged in order of importance. It becomes so possible to define a threshold of variation these elements have to reach to be considered in rest of the work. In our experiment we used the value of 95 percent because it permit to consider the most relevant terms without restrain or expand too much the study. The threshold value can have been chose differently. So in the rest of this study we interested in terms whose sum of singular values is larger than ninety five percents. We can speak about percentage after a little operation on the singular values to put them in a basis where the sum of elements matches up to hundred. After

considering the elements which reach this threshold we set, we can reduce the multi-dimensional study to a narrowed field. The goal of all these different actions was to focus the rest of work on really determining set of information. It permits also to restrain the dimension of the future clustering since the difficulty of the study rises with the size of the dimensions. This last point becomes important when we consider that the study will be adapted to the work on head shakes, eyes gazes and other human body movement.

3.3 – Clustering/Data Analysis

After the processing of the information contained in the velocities files we have to find a way to bring together elements which did not at first sight present similarities and to automate this analysis. The fact that there is some hidden information becomes from the fact we are considering a multidimensional data set. To reach this goal we chose to use a tool called clustering which is nowadays largely employed in many fields including data mining, image analysis, or pattern recognition.

3.3.1 – Goal of clustering (different appearance of head nods)

The data clustering consists on a classification of the studied elements in different subsets called clusters. It means that all the data set will be partitioned in parts which have to share common particularities. These common traits are the most often defined by a notion of distance measure [1]. Figure 3-5 shows a clustering on three clusters. The X-axis could be considered as the weight of studied persons whereas the Y-axis their height. The way of clustering can be enlarged to multidimensional elements where each dimension can represent something different such as the weight, the height or the age.

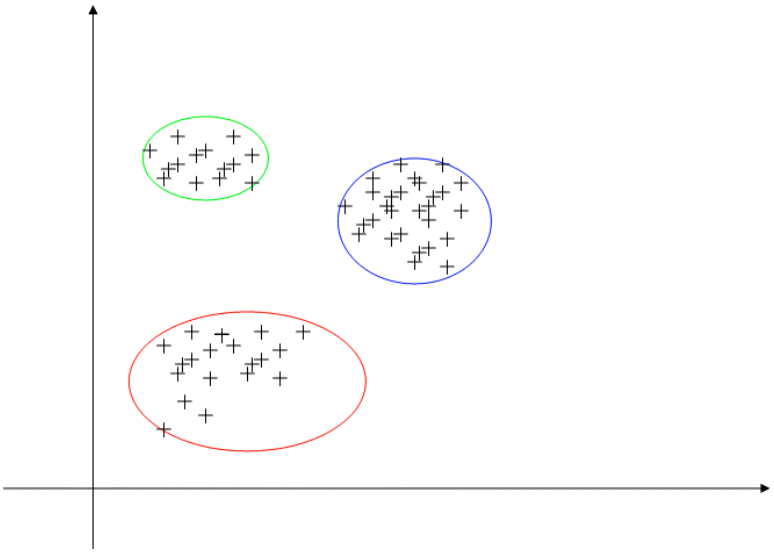


Figure 3-5: Example of clustering

There are different ways to run the data clustering algorithms. You can use hierarchical or partitional algorithms to make clustering. The hierarchical one consists to find successive clusters using subsets established previously while

the partitional algorithm checks all the parts at once. We have more precisely studied a hierarchical one called K means clustering in our work.

In fact this method of discovering “natural” groupings in data set allows us to ascertain hidden patterns for the classes of motion. We used it at the beginning with simple characteristics such as the length of each head nods but the K mean clustering algorithm studied in the following subsection allows us to study multi dimensional data set. It means that we could cluster data composed by different elements as the length, the frequency or even the m-by-n matrix studied in the FFT part.

3.3.2 – K means clustering

There are several clustering methods which could have been employed but we chose the K means algorithm.

It is used for different reasons:

- This tool is one of the most simplistic and rapid algorithm. It permits thus to work with large data base such as our 208 sequences of head nods which possess each hundred of information.
- We can rapidly find free MATLAB toolbox on the Internet. We had to do it because of the lack of time and because of our MATLAB version does not owned this toolbox. These entry-level functionalities are available in the last version of the software.
- It is possible to manually choose the number of clusters you want to study. It is important if you have already an idea about this one.

We will consider that each head nod of the data base is represented by a vector in the rest of this part. It is these vectors which constitute the m-by-n matrix where n is always the number of head nods and where m corresponds to 32 for the velocities and frequencies studies (it is the length of the window) and to almost ever to 3 (it is the number of S matrix elements needed to reach the threshold).

This algorithm functioning is grounding on the assignment of each vector to a cluster whose centroid is nearest. The centroid is the cluster’s centre and it is defined by the mean of the entire vectors which are in the subset. Its coordinates are specified by the arithmetic average of each dimension over all the information elements in the considered cluster.

The K means clustering works by steps:

- The number of clusters k is chosen. In our study we give it several values but it is able to find the optimize number of clusters. The action of giving it the number of clusters it has to create comes from the fact we

had a suspicion about the number of head nods types a human being can own.
- It randomly generates the k clusters and calculates the centroid coordinates.
- The algorithm will assign each point to the nearest centroids.
- All the new cluster centers are recomputed.
- The K means clustering repeats the two last steps until the position of each point is set.

This tool was applied on different characteristics of the head nods such as its length, its frequency. For this last study we have also created a file containing figures plotted by MATLAB. These figures representing different clusters with different sampling rate and different size of windows were made by script to automate their creation and can serve to gain time for futures works.

We even applied the K means algorithm on the output of the SVD to determine if the three prototypal head nods we guessed are really existent. We aimed to prove that the typical human head nods are composed by different values of frequency. For example a “keep going” head nod is perhaps determined by a movement composed by sixty percent of 2 Hz frequency and forty percent of 4 Hz frequency.

3.4 – Experiments

First of all we had to remember that the goal of this part is to create a toolbox which will provide the ability of future searchers to find patterns in gestures such as the head nods, head shakes or other motion in good condition. We test the gesture analysis toolbox across different cases. First we do experimentation on the velocity and the length of each nod. Then we consider the frequency found thanks to the FFT and finally we focus on the results of the PCA.

The representation of these results is done by a visual way thanks to different MATLAB functions we have created.

3.4.1 – Velocity

We had begun to visualize the most primary information given by the Watson software such as the velocity according to the speech. We did it because we believed that a type of head nod is correlated with a time in a sentence. For example an agreement head nod can appear during a pause or at the end of the sentence. To make it easier for future study we allow users to choose different ways to visualize the results. They can select if they want to consider a sole video sequence, one session of the recorded video to find out the different pattern gestures which composed this sequence (see Figure 3-6). They can also decide to consider a head nod type of pattern in all the video.

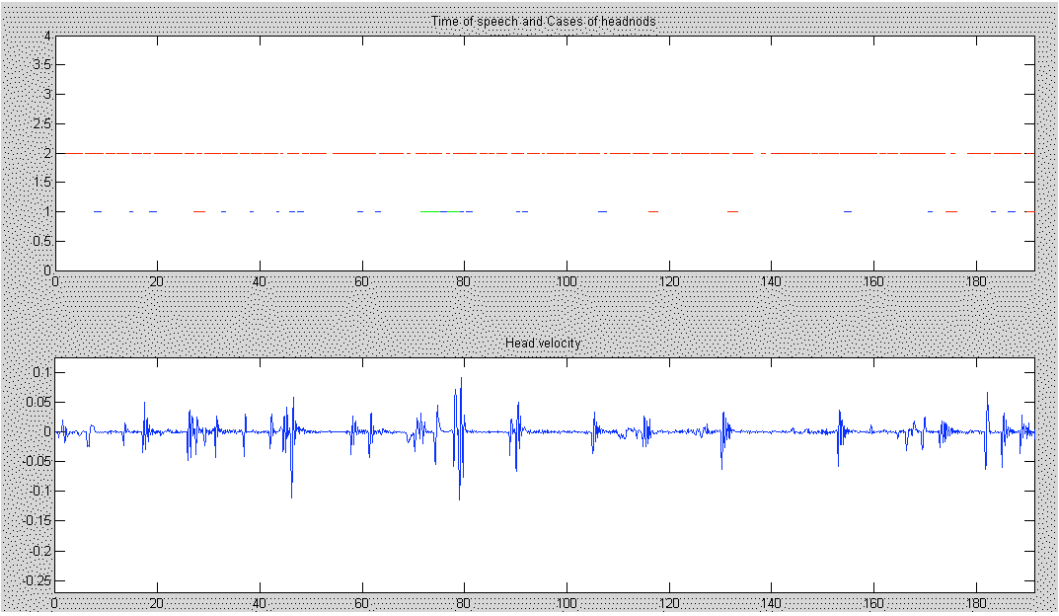


Figure 3-6: Head nod velocity, types of cluster and speech

This plotting shows a particular video session with the time of speech (in hatching red) and the three different types of head nods in the first rectangle. The

green dotted line is corresponding to the first cluster; the red dotted line is corresponding to the second cluster and the blue one to the third cluster. The second rectangle is composed by the head velocity (according to x for a nod) during the total session.

We used different number to realize the different clusters in the previous part but in the following example we will consider the clustering in three different groups.

Figure 3-7 is the representation of few head nods from the third cluster and for the second cluster. These plotting are also realised with a specific sampling rate and a size of window (see subsection 3.2.2) chose by the user. We can already see that the head velocity (in blue) is not the same for the two clusters. The speech which is red plotted is not present in the two considered clusters but just in the right one. We would so consider that there is a correlation between one type of pattern gesture and the speech. However the speech on this cluster is not present in all the cases. So at first sight it appears that there is no singular correlation between the different gesture patterns and the fact people are speaking or not. This observation incites us to consider new data given for example by the study of the frequency or by those of the SVD.

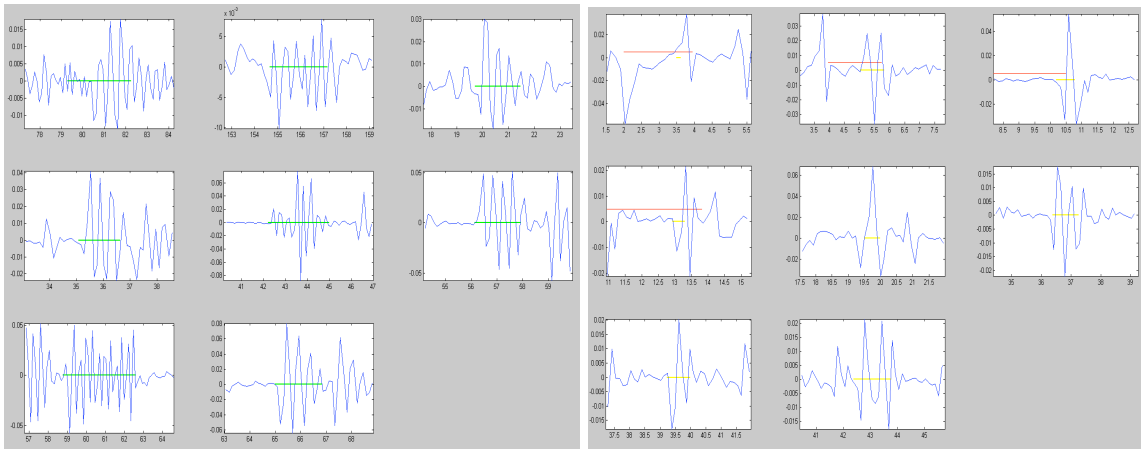


Figure 3-7: Two examples of clusters

3.4.2 – Frequency

The calculation of the FFT and its clustering would allow us to find directly what frequencies are the most significant for a type of head nod and perhaps to do a correlation between a head nod and the time of speech.

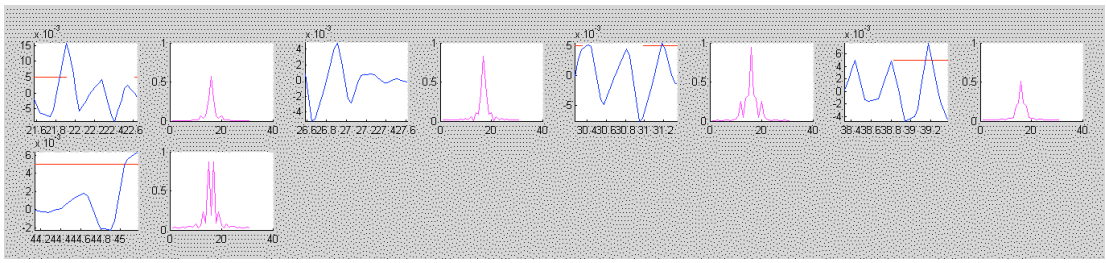


Figure 3-8: FFT, velocity and speech for cluster 1

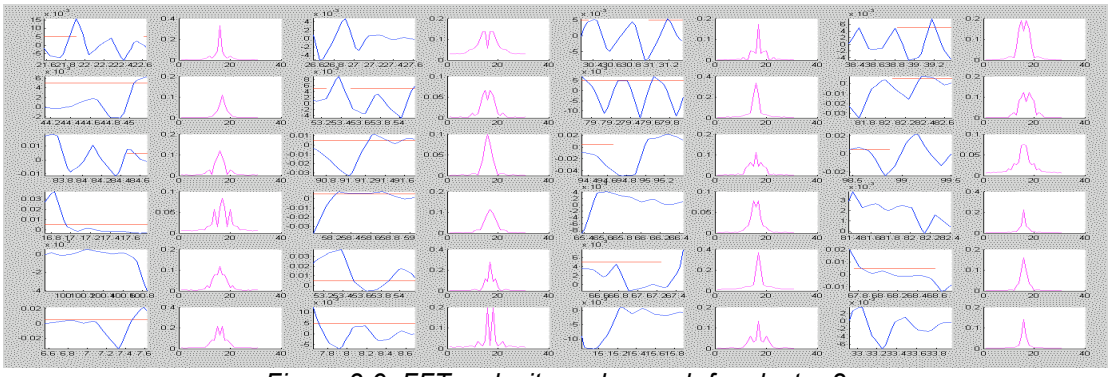


Figure 3-9: FFT, velocity and speech for cluster 2

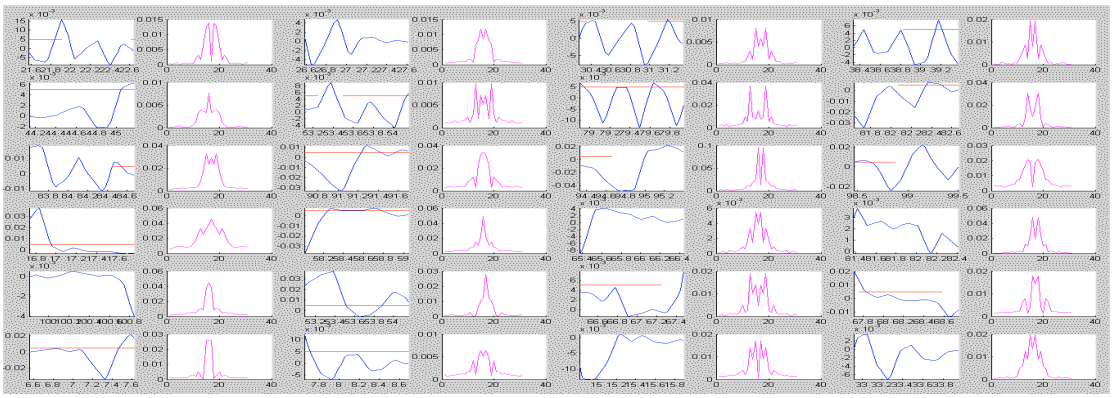


Figure 3-10: FFT, velocity and speech for cluster 3

On the three previous figures, there are the head velocity (in blue) and the speech (in red) side by side with the plotting of the FFT for each head nod of a particular cluster. These three figures permit to underline that the particular clusters seems to look different. However it is not possible to find out a correlation between the time of speech and the types of cluster. It is why we restricted again the research field and interested more precisely on the frequency elements after there decomposition in singular values.

3.4.3 – PCA

The study of the frequency SVD and its clustering permitted to find that on the average two or three frequencies are carrying 95 percent of all the information.

The plotting of each different types of gesture pattern according to their frequencies in the U bases which possess the greatest variance (see subsection 3.2.3) permit to underline that there is gathering of the types of pattern in different parts of the figure. Each gesture pattern will possess a frequency average which could define it. For example the red points which represent one type of clusters on Figure 3-11 below are all grouping together in the left bottom corner. After comes the green points represented another type of clusters and finally the blue points which are the most scattered.

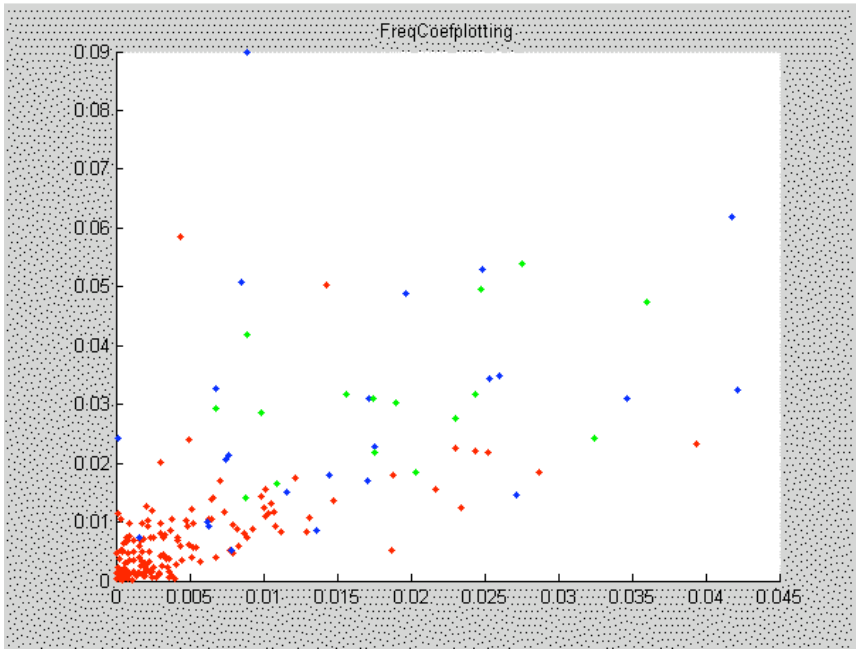


Figure 3-11: Plotting of the clusters frequencies according to U bases

One of the plotting lets us also predict that each type of cluster contained a particular frequency or the composition of a little number of them. It is particularly observable on the Figure 3-12 where is plotted for each cluster the average frequency. To plot this figure we consider for each cluster the number of the V matrix columns needed to reach the 95 percent threshold (see subsection 3.2.3) and we multiply these columns by a term called “a” which is the mean of each of these columns. The fact of considering the “a” term is done to not neglect some frequencies values which could have been crushed by the weight of other terms. In Figure 3-12 below and for this particular case the number of columns we have to consider is three. So we are plotting $a_1xV_1+a_2xV_2+a_3xV_3$ where a_1 is the mean of the V_1 column.

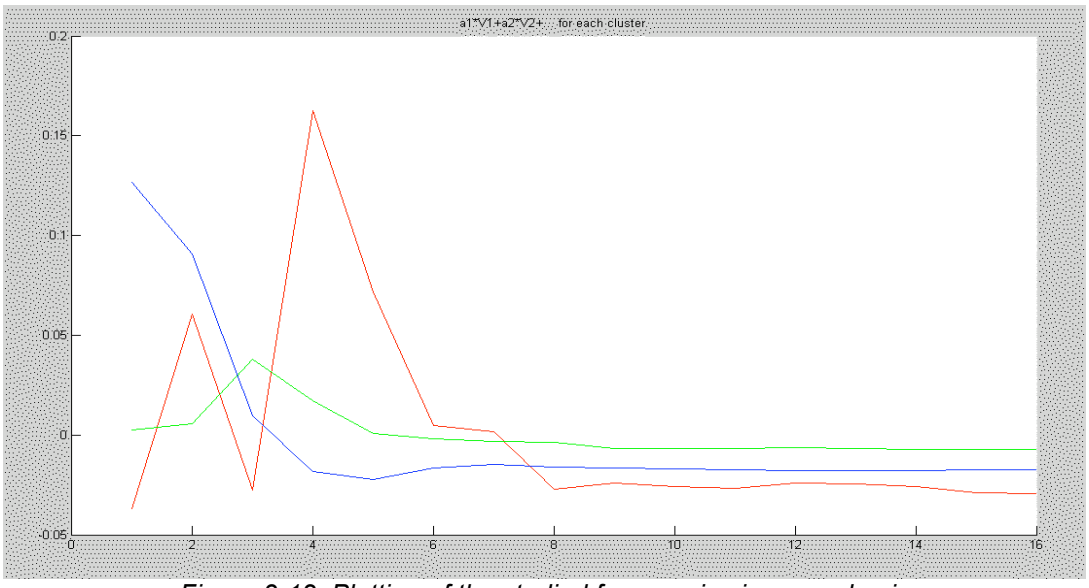


Figure 3-12: Plotting of the studied frequencies in a new basis

On Figure 3-12 we can notice that the first cluster (in red) is composed by two peaks. The first peak is around the two hertz frequency and the other around the four hertz frequency. The second cluster (in blue) seems to have a peak around the zero frequency whereas the third cluster (in green) has a little peak around the three hertz frequency.

Though this part of the work has not permitted to bring out the link between a particular type of head nod and a particular time in the speech such as the pauses or the beginning of a sentence, the last plotting permits to obtain an important result which is that each gesture patterns represented by a cluster is defined by a particular sample of frequency. So it becomes ever since possible to create a prototypical head nod for each of the different patterns

The continuation of the study is done in the shape of a user study and will permit to find out if the different patterns are related to different functions.

3.5 – User study

After the construction of the gesture analysis toolbox we have to find out if the different gesture patterns we found have a specific timing and a particular function. The prediction of the timing is realized thanks to the Prediction toolbox. Its functioning is explained more precisely in the Chapter 4. The correlation between a gesture pattern and a specific gesture is underlined thanks to a user study.

3.5.1 – Goal

The aim of this part was to create a tool for the head nods study which can be easily used by ICT external persons. This tool had different aims. First of all it has to verify the fact that the head nods possess different functions such as the agreement, understanding or “keep going” head nods. Then it can permit to show if people have the same understanding of these different behaviours. It means that we wanted to know if people have the same way to label the human features. This tool was also created to permit the study of other motions such as head shakes or eye gazes and finally, it will be used to find out the link between the patterns and the functions.

The better way to ensure that a lot of people can do the User Study is to create an online study.

3.5.2 – Data preparation

The preparation of all video sequences has to be done before the study. Preparation means that a user will not watch the video one per one and write down what he thinks about the head nods he found. Indeed the video has to be shorten otherwise the study will take too much time. It can also appear in some sequences that there are different head nods which are placed in row. So the user can forget to study one or more of them. Thus the work we had to do before beginning the creation of the web pages was to extract the chosen duration of head nod and to add a visual effect which characterises the time of nodding.

We consider 22 video sequences consist of 208 head nods. To faster the study we had to put the studier in the context of the sequence without showing him all the video. It would have been a waste of time and the watcher would have lost his concentration. We also cut all the sequences with a delay of twenty seconds before and two seconds after it. The delay before the nod has no

mathematic reason to be picked but the experimentation showed that duration under teen seconds did not permit to the studier to understand the situation and that a duration upper twenty seconds was too long to permit the study of numerous sequences. The way we do these sequences permits as well to begin the watching of the video at the time the user chooses.

We had also to bring out the studied head nod in the video. Indeed a same video sequence can own many head nods and there are some sequences which have following head nods in a short time. To solve this problem we chose to add a visual effect at each video. We preferred to add visual rather than audio effect because the speech during this time is important to understand the reason of the nod. Instead of just adding an image during the time of nodding on one of the corner of the video we chose to frame the entire screen with green line during the head nod. We preferred this way because the studier would be focused on the image during its showing rather than on the head nod.

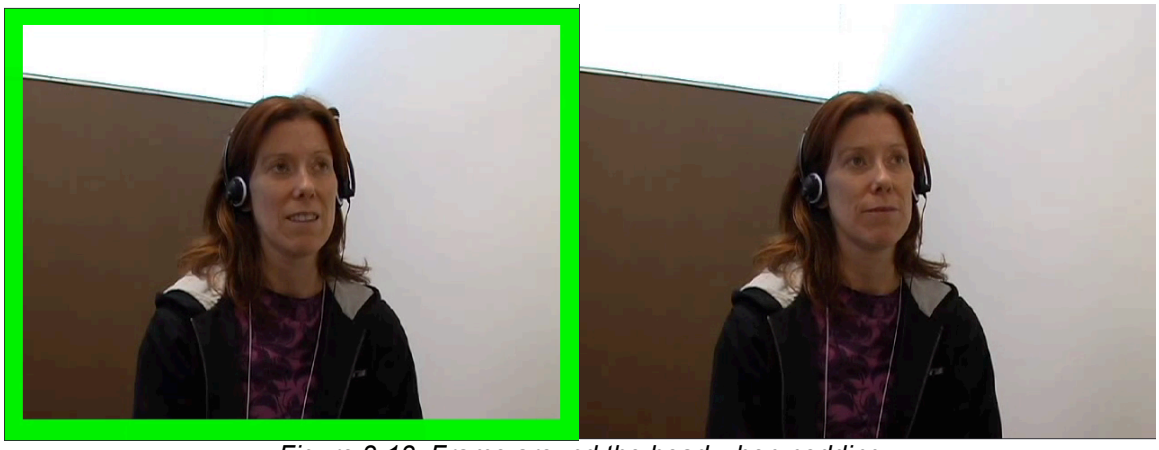


Figure 3-13: Frame around the head when nodding

To realize these actions we chose to use Virtualdub software (see Appendix B). This software already used in the data processing part was chosen because it is free software easily expendable and which is compatible with a scripting language. The last point was really interesting to create rapidly the entire sample for the future studies.

We generated likewise different scripts for the studies. A script is a file containing the number of training video sessions, the paths of these video, the number of study video sessions and the paths of these video too.

All script files are generated by a MATLAB function. The specifications to realize these scripts are multiple. In the draft version of this tool there were:

- All the paths have to be randomly chosen. We used a MATLAB function which permits to give random numbers functions of the time the application was launched.
- Each script contains 8 training video and 41 or 42 study video.

The scripts were made five per five. In each new series of script a same video can not appear twice in different study video sessions and the training video sessions can contain video which were already in a study video session but which are not ever contained in the considered script.

3.5.3 – Web pages

We have also created four different web pages in HTML/PHP. These languages were used because we had since learnt them before the internship and because there are used in other user studies create in ICT. These pages will permit to explain the goal of the study to the user, to provide a Training session and a Study session. At the end of this last session the studier has also to fill in a little survey to improve the future versions and to give new ideas about the head nods subject.

The presentation of the Training session and the Study session is the same as the one shown on Figure 3-14 below.

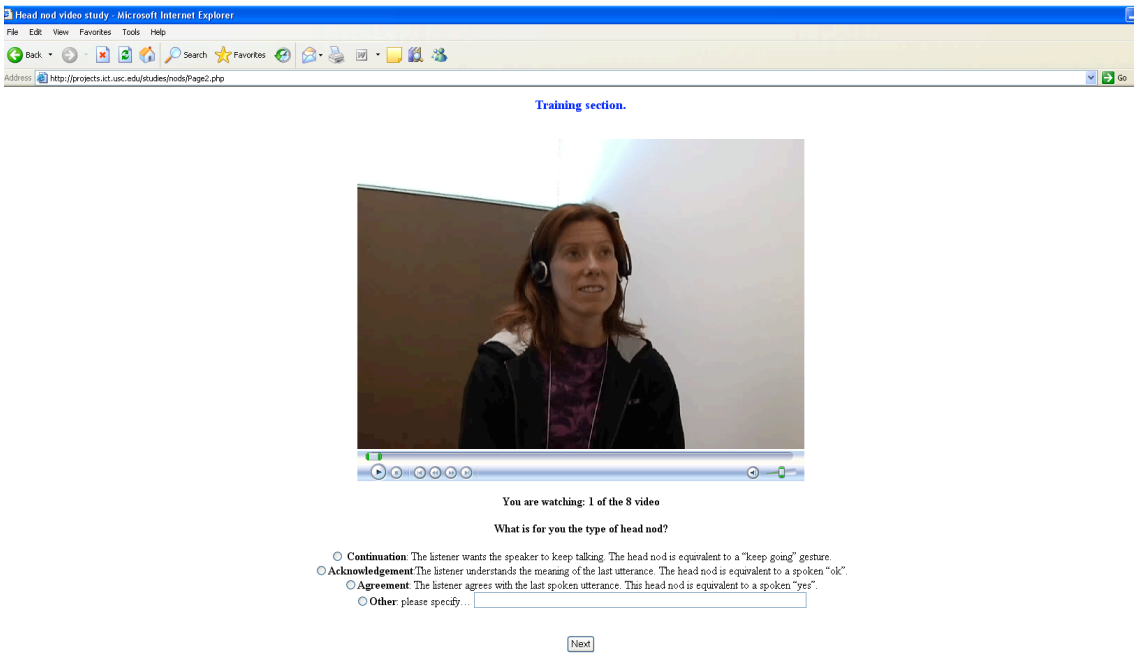


Figure 3-14: Screen shot of the Training Session

During the study the user can watch the video as much as he wants. He has also the choice to check off one of the four choices for each head nod sequence:

- **Continuation**: The listener wants the speaker to keep talking. The head nod is equivalent to a "keep going" gesture.

- **Acknowledgement**: The listener understands the meaning of the last utterance. The head nod is equivalent to a spoken “ok”.
- **Agreement**: The listener agrees with the last spoken utterance. This head nod is equivalent to a spoken “yes”.
- **Other**

The last case permits to obtain the user’s opinion via a text area. It can improve the study because it gives a new view on some head nods. If there are enough information it can also permit to find a new head nods types which was not include in the three detailed choices. All the results even those for the training session are saved in a file called Log. All the Log files are presented in column under the form:

- The time during the check of the radio button
- Session number
- Case (study or training)
- Time delay before the head (here twenty seconds)
- Video sequence number
- Head nod number in the sequence
- Head nod number in all the video
- Choice of the user (number of the checked button)
- The text if the other case is checked

To begin the study the user needs a sole session number sent by us. This number permits to log in at the beginning of the work and to remember the studier identity during the examination of his responses. Indeed all the surveys are saved in the Log file with the user session number as name. This number gives him also access to a particular script.

3.5.4 – User Study results

The User Study will provide some results on the link between the different gesture patterns and the functions of the head nods.
At the end of the internship the lack of time does not permit us to gather some results about this study. The future work on this subject will be to create several MATLAB functions which collect all the different users’ responses and compare them. These functions will permit to underline the rate of agreement among these responses and if this rate reaches a particular threshold to really do a correlation between the patterns and the functions we studied.
The other way to consider this study is in an improvement point of view because the users can also propose some functions we do not think about.

3.6 – Conclusion

The goal of this entire study was to construct a tool which can permit to analyse human non verbal feedback. We focused just on head nods to improve this first virtual human characteristic but all this study can be done with others behaviours as eye gazes or head shakes. The principal aim is to permit an enhancement of all the Virtual human has fast as possible. So if we had more time we could have begun the study of another human compartmental characteristic. Or perhaps improve even more our existent functions. It means that the processing time of a function can always be enhanced or that the graphical presentation of the different figures can also be improve.

The Analysis gesture toolbox has provides relevant results about the gesture patterns. According to this study it becomes also possible to create a prototypical head nod for each pattern thanks to the clustering of PCA results. Nevertheless the correlation between the speech and the different clusters was not revealed by the multiple figures. We have also to remind that all these results were generated by the study of just 22 video sequences. So we can think that the study of more sequences could have permit to underline more relevant results.

The User Study which is in use for finding out the correlations between the different patterns and the specific functions has not been launched before the end of the internship. It is why there different results do not appear in these pages.

We can already ask ourselves on the elements which can be improved to enhance the future studies.

After studying the primary elements given by Watson software we applied the FFT on all results. We could have used another mathematical tool as the MFCC (Mel Frequency Cepstral Coefficient) to do the head nod frequency study [4].

The way we done the clustering could likewise be changed. In fact we used the K means clustering which is one of the most rapid and primary mean to do clustering. This algorithm has also the disadvantage to not generate the same result with each run because all the clusters found in result depend on the random assignment of the beginning. This sort of clustering has to find a minimal intra-cluster variance but the result has not ever the global minimum of variance because this tool has not the means to ensure its different results. So other ways to do clustering such as Fuzzy c-means or Normalized-cut clustering can be done by future interns.

The link between the timing of a head nod presented in the next part and the different functions can also be done after the study of the future results of the User Study.

Chapter 4 – Prediction Toolbox

4.1 – Introduction

4.1.1 – Goal

In order to make the agent looking more realistic, one of the requirements is to make its gestures coming at an appropriate time. So it becomes natural that one of the branches of this study is to be able to predict when head nods are likely to happen. Some work has previously been done to attain this goal [8], but it seemed that these results could still be improved. Head nods would be the starting point of an approach that could be repeated for each kind of gesture.

4.1.2 – Approach

The approach of this prediction toolbox can be described in five steps:

- First we had to familiarize with tools L.P. Morency already created in order to be able to use them properly and to modify them if necessary. This implies to have a better understanding of machine learning techniques and related functions.

- Second, we had to select features. Indeed, a relevant subset of contextual events, called features, is needed in order to obtain good results with model training. This has been done using both visualisation tools which will give an intuition of the selection, and functions that will select automatically a subset of relevant features through all contextual events available.

- Then, we had to encode these features to be able to use them with currently available training model functions. Moreover the encoding could vary depending on the model we will use since each model has its specificities.

- Next, the results had to be compared in order to select the model which fits best with the gesture prediction. This has been done by using many visualisation tools that will be explained later in this chapter.

- At last, once the best model has been chosen, we had to confirm the improvement of previous model by running an evaluation of the new model.

4.1.3 – Available Materials

Starting point of this prediction toolbox are the previously mentioned imported in MATLAB as explained in Chapter 2. We have at our disposal the speaker words, some clues about its prosody and the gesture ground truth basis of our study. Main issue will be to select the good features that will be useful for predictions.
Moreover, most of the functions concerning machine learning techniques had already been developed by Louis-Philippe Morency and need only to be optimized to run faster, and to get better visualization of the results.

4.1.4 – Specifications

Again, being visual stayed at the top of our preoccupations. Moreover, we tried to limit to the minimum the number of functions for the user to launch for having results, from the choice of the features, to the launch of each experimentation run, this responding to the “Easy of use” specification initially mentioned in subsection 1.1.2

4.2 – Machine Learning

4.2.1 – Objective

Even you have some intuition of the events which have an impact on increasing or decreasing the likelihood that a head nod will happen, you could miss some hidden coupled features that may have an impact on the likelihood.

This is the goal of machine learning approaches. Starting from samples with events and their results, it allows you to generalize results to the different events which could happen.

4.2.2 – Generative vs. Discriminative

Machine learning has two main approaches: Generative models or discriminative models. Both of them can be used to classify the data they have in input.

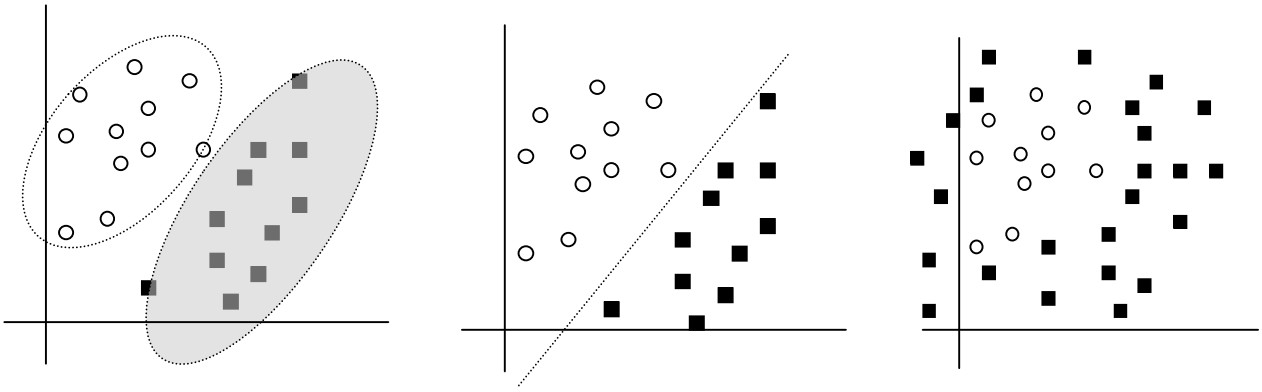


Figure 4-3: Generative approach issue

Generative models tries to create a shape that will include same type of data. These shapes are mostly Gaussians. In fact, it computes the distance to the center of each ellipse and the data will belong to the nearer with a probability proportional with this distance.

The second approach is to find the boundary between the different kinds of data. The probability of a data belonging to a type will be proportional to the shortest path to the boundary and will correspond to the data side next to the boundary.

Case showed on Figure 4-3 is an example of the limitations with generative approach. One shape is easy to find but the other is much more difficult. Indeed it would be easy to create a shape around small circles, but you would need a lot of shape mixtures to regroups small squares. However, only one line could discriminate both domains.

4.2.3 – Available Machine Learning Toolboxes

Our goal is to clarify the differences between the different models used, in order to be able to compare them. Please have a look at references for more details.

In this study many toolboxes have been used for modelling. Concerning the generative models, we used Hidden Markov Models (HMM), Hierarchical Hidden Markov Models (HHMM) whereas discriminative models were Support Vector Machines (SVM), Conditional Random Fields (CRF) and Latent-Dynamic CRF (LDCRF).

a) SVM: Although this is a discriminative model, it is non-dynamic which means it works on instantaneous observations which do not depend from previous states. This is why there is no transition matrix on this model [3].

b) CRF: It is the most basic dynamic discriminative model we used. However this model does not represent internal substructure [6].

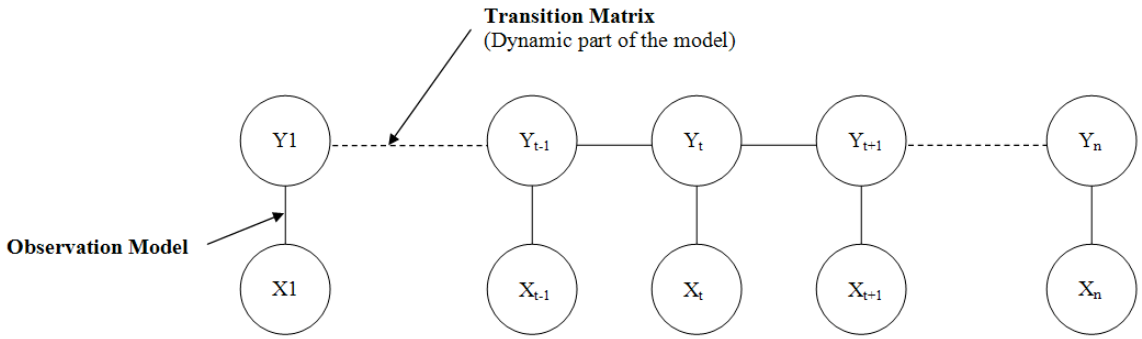


Figure 4-4: Conditional Random Fields principle

To explain it, we will use a simplified example of our work. Y_i is label that corresponds to one state at the i-th frame. In our case it would be 1 if it is a Head nod and 0 if it is not. X_i is a vector containing the features status at this same i-th frame. A feature is a numerical representation of a contextual event used to train the prediction model. Each element of the vector is a feature encoding any action from 0 if the feature is not active to 1 if it is (see subsection 4.3.3).

For the CRF model, the transition matrix, which is a link between two frames, would be a two-by-two matrix. For each state, you have the probability of either staying in this state or change. This is nearly common to each dynamic model. The part which diverges most from one model to another is the observation model, it means given the features, the way it will calculate the probability of having Y state to 0 or to 1.

c) Latent-Dynamic CRF: Latent is similar to hidden. Indeed this model adds hidden states (h on Figure 4-5) between the features (X) and the labels (Y). The

observation model is the same as CRF, and there is a deterministic link between labels and hidden states. They are called hidden because you do not know what they correspond to; you only set the number of hidden states you want to be used [7].

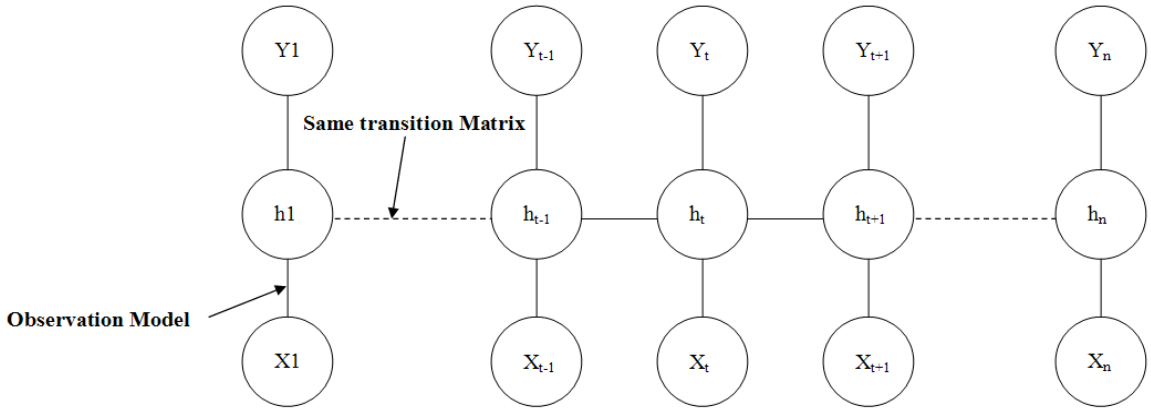


Figure 4-5: Latent-Dynamic Conditional Random Fields principle

d) HMM: The structure is the same as LDCRF since this one is inspired from the hidden states of HMM. However it keeps the old-fashioned generative model which is outperformed by CRF for activity recognition. Moreover the transition matrix is slightly different from the LDCRF one [5].

4.3 – Features

4.3.1 – Introduction

The meaning of feature as an input for modeling has been explained in the previous section. But the meaning of feature in this study has to be explained. In this case, available features are words pronounced when the speaker is retelling the story (see subsection 2.1.1), punctuation, pauses, or “Actions” logged by the Elvin system (see subsection 2.3.4).

Features are a key input in the model experimentation, so we had to take some time selecting them, with many tools.

4.3.2 – Importing Actions

The very first step was to generate a more extensive action cell array, adding to the previously created one all the Transcriber related features. But this had to be organized:

-**Unigrams**: This is just one word in itself, meaning a group of characters without spaces in it. The only exception is the character ” ‘ “ since “That’s” is saved as “Thats”. The begin and end time of each unigram is taken with the first and last character proportionally to the utterance length since the only available time data were the beginning and end of each sentence.

-Punctuation: Every character which was not a letter is considered as punctuation. The temporal situation was done with the same principle as unigrams. It was a simple way to do it, but it revealed to be efficient since each punctuation put in annotations - except ” ‘ “- reveals a context information (see Appendix C).

-**Pauses**: To add pauses, the only thing to do was to take the end time of current utterance and start time of the following one when they were not equal.

-Pairs: This meant every association we could make with two consecutive, previously created features. It could be two unigrams, unigrams with punctuation, pauses with punctuation, and unigrams with pauses.

Once all this have been done, this big cell array contained all actions in all sequences. So action{i,j} gave as a result in the first column the beginning time, and in the second column the lasting time of every iteration of the i-th action - whose name was caption{i} – in the j-th sequence.

4.3.3 – Selecting Features

A model performs best on a subset of relevant features. Moreover, too much features could cause the model to be over fit, this meaning that the model will not be general enough, this causing the model to be best performing only on the samples he was trained. This is why this section is a key point in the selected approach. Relevant feature is a feature that has an effect on the likelihood of a head nod happening. This effect could either be positive or negative.

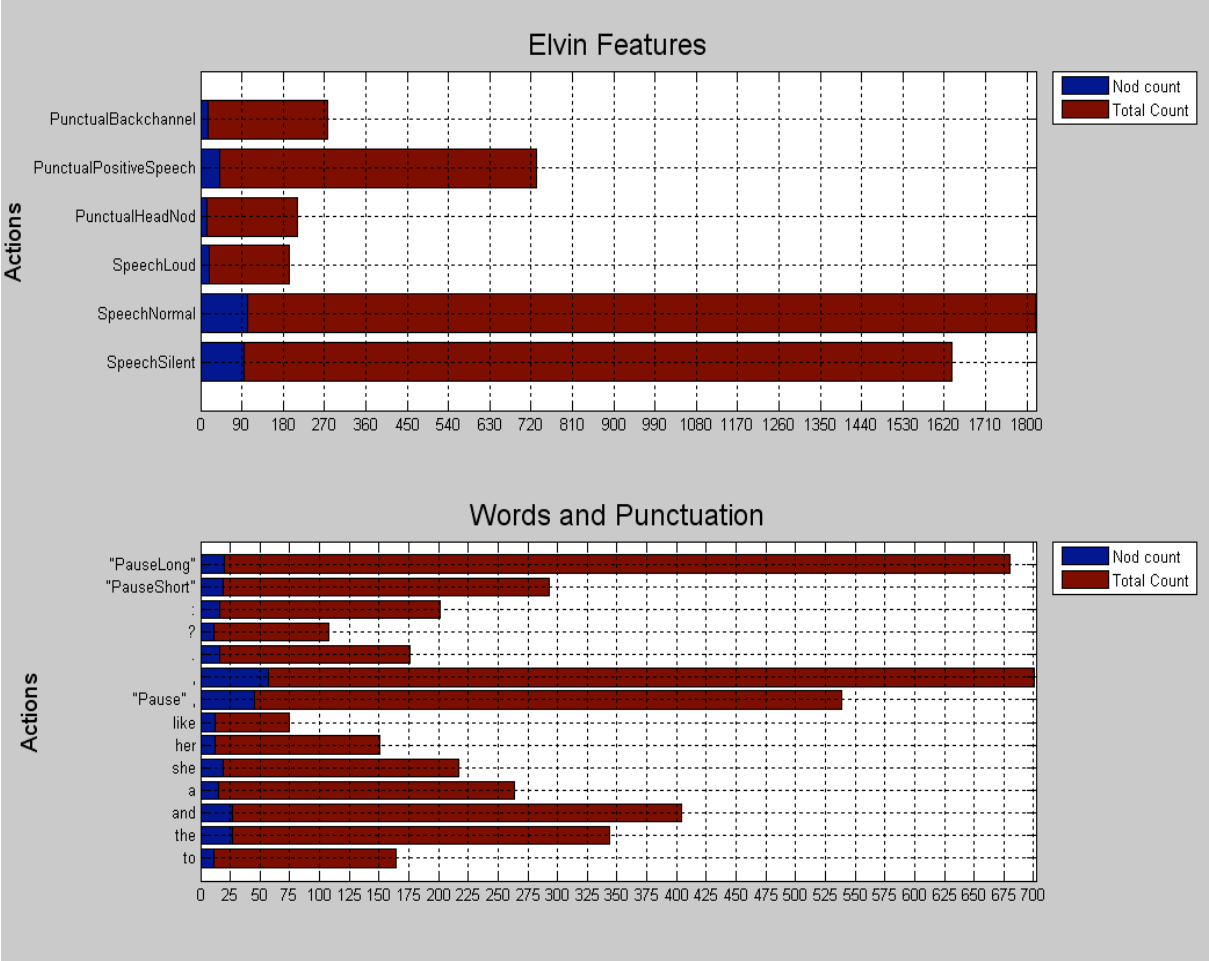


Figure 4-6: Histogram of word appearance statistics

First step is to create some visualisation tools. The two main used graphs are the one displayed on Figure 4-6 and the one displayed on Figure 4-7. The first one is a histogram showing in blue the number of times the features (words, punctuation or Elvin actions) are appearing during a time window before a head nod and in red the number of times it appears in total. As input for this function, these actions can be filtered by selecting the minimum “Nod Count” and the time window before a head nod to be taken or not into the “Nod count”.

This diagram is useful to have an idea of which features are relevant since it highlights the link between the moment they appear and the moment when appear head nods.
However a second Visualisation tool has to confirm this intuition. Indeed some of these words could have a sufficient number of appearances only because they are common words, widely used in English vocabulary.

The graph displayed on Figure 4-7 goes more deeply into the numbers shown by the previous one, since it displays the number of nod appearances in a given time window around the time the word was spoken. The goal is here to

have “peaky” features, meaning we want a maximum or minimum peak around the time that a contextual feature happens.

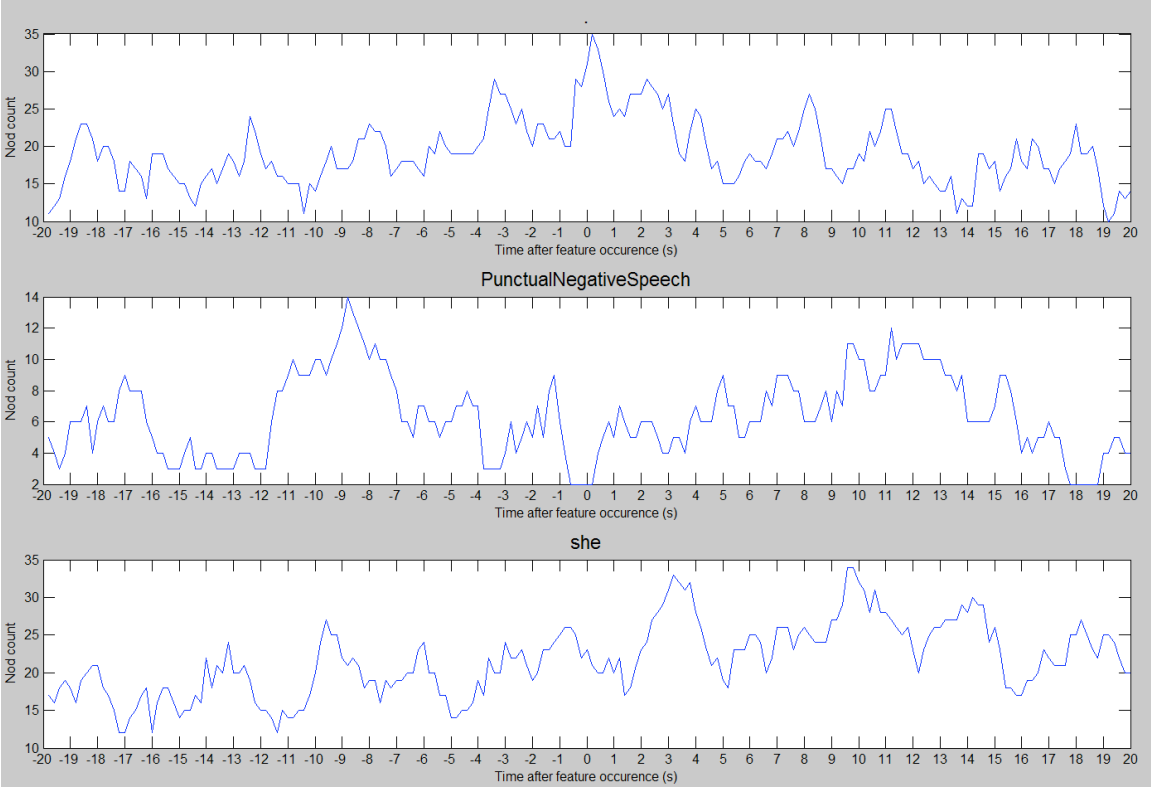


Figure 4-7: Nods appearance statistics graph

For example the top graph of Figure 4-7 confirms the intuition about falling intonation given by Figure 4-6 at the top since there is a peak reaching its maximum around the X-axis 0 value. It means that a head nod is more likely to happen around the time this contextual feature occurs than every time during this time window. Y-axis is the nods occurrence during each time segment (called bins). Moreover, a maximum peak could be as interesting as a minimum one. Indeed, a minimum peak around x=0, as displayed on the middle graph of Figure 4-7, contains the information that at the time this feature appears, it is not likely that a head nod could happen, this making our model even finer. At last this was useful to drop commonly spoken words like pronouns (lower part of Figure 4-7), which had no minimum or maximum value around the x=0 value.

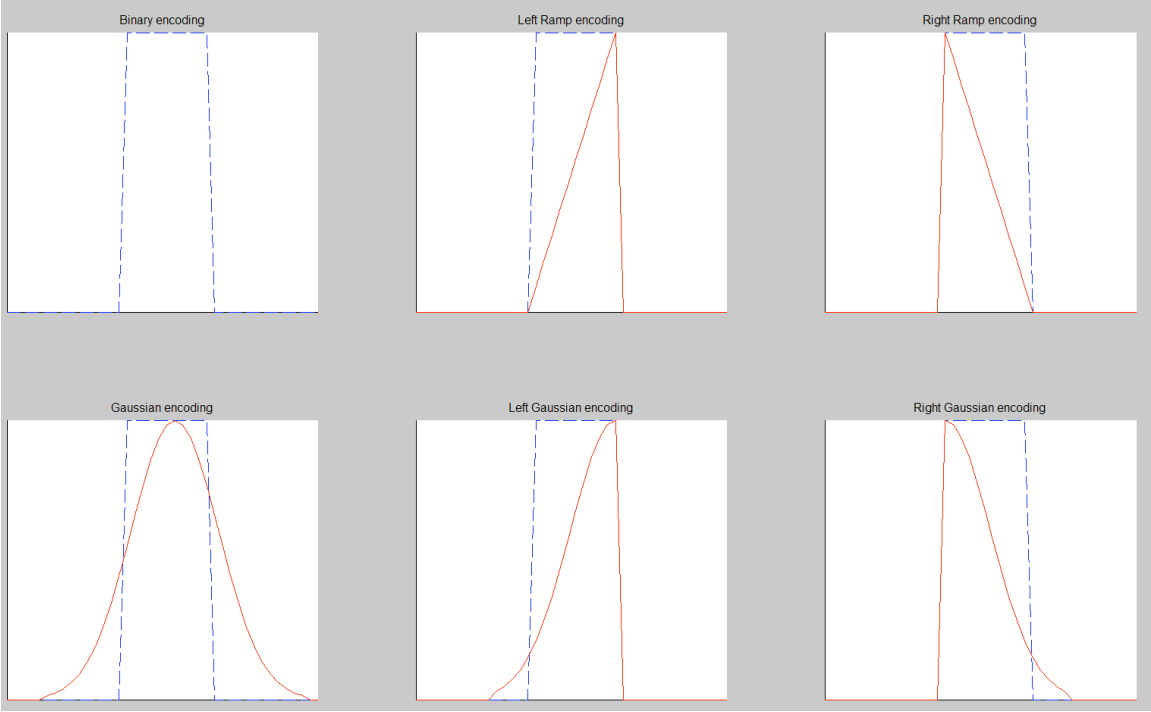
However, we had to keep close to the specifications which were to be as automatic as possible. This is why one of the functions created uses this graph, to see for every feature if it is relevant or not. This function filters automatically the actions which are more likely to have a link with head nods, using the Histogram principle and then selects the best features by the previous graph logic. This permits to have quickly a basic feature selection.
Intuition given by the histogram and automatic selection of the best features lead us to this feature selection, which is a mix of both: Long pauses, continuing

intonations alone and coupled with pauses, falling intonations alone and coupled with pauses, rising intonations alone and coupled with pauses, incomplete words, the ELVIN reports of negative speech and up gazes, but also the words “and” and “uh” are all the features that finally gave us the best results according to the visualisation of the results.
Intuition gives the first idea of what a feature selection should be, and then the automatic features selection enables to confirm or to add unplanned features to this selection.

4.3.4 – Encoding Features

In order to create features, two functions are necessary:
The first one has to convert the Actions file into the final Features file format which can be used with the modelling tools L.P. Morency already created. Indeed Actions file is a cell array where the cell line corresponds to a specific action and cell column is a specific sequence. Each cell contains a line by action appearance with its begin and end as explained in section 3.2. The features format that needs to be used in input of the prediction models is a cell by sequence where each cell contains a matrix where each row corresponds to one feature state and each column is a different frame. This implies the creation of time stamps for each sequence, in order to be able to have the corresponding time for each frame.

Model used for prediction are responsive to the sampling of their features. We use three types of encoding for them: The first encoding is the binary one, which is the simplest, and which corresponds to a 1 when the feature is active and a 0 in other cases. The second encoding is a ramp which reflects the intuition of the time needed for the listener to understand and to take into account this context feature. The last encoding chosen is a Gaussian encoding, which gives a smoother transition than the ramp. The two last can be modelled to be descending, ascending or –only for the Gaussian– symmetrical.



Moreover it appears that these different encodings can have an important impact for the result. Indeed, even if the binary encoding works best after testing for nearly every model, it appears that Gaussian encoding gives better results than binary on the HCRF model. At last it had to be tested in order to make sure that we use the best encoding for each prediction model.

4.3.5 – Conclusion

Features selection and encoding have such an important impact on the results that we had to spend most of the time comparing results obtained with different features selection and encoding. Although the automatic feature selection gave us some unplanned results which proved to be helpful, it also was not sufficient for the features selection which gave us best results, and which was a mix of these results but also some intuition concerning the features.

4.4 – Optimisation

Once all conditions have been met to launch the experimentations, time quickly became a problem since a run with LDCRF model could take up to 36 hours. So it has been decided to distribute the work charge on the new servers we had at our disposal: two computers with two quads core Xeon CPU running at 2.66 GHz. In order to do that, getting familiar with the Distributed Computing Engine was necessary and took some extra time on the planning. However, the time won by this work parallelisation will easily catch up with the time lost for learning the basis of this Toolbox.

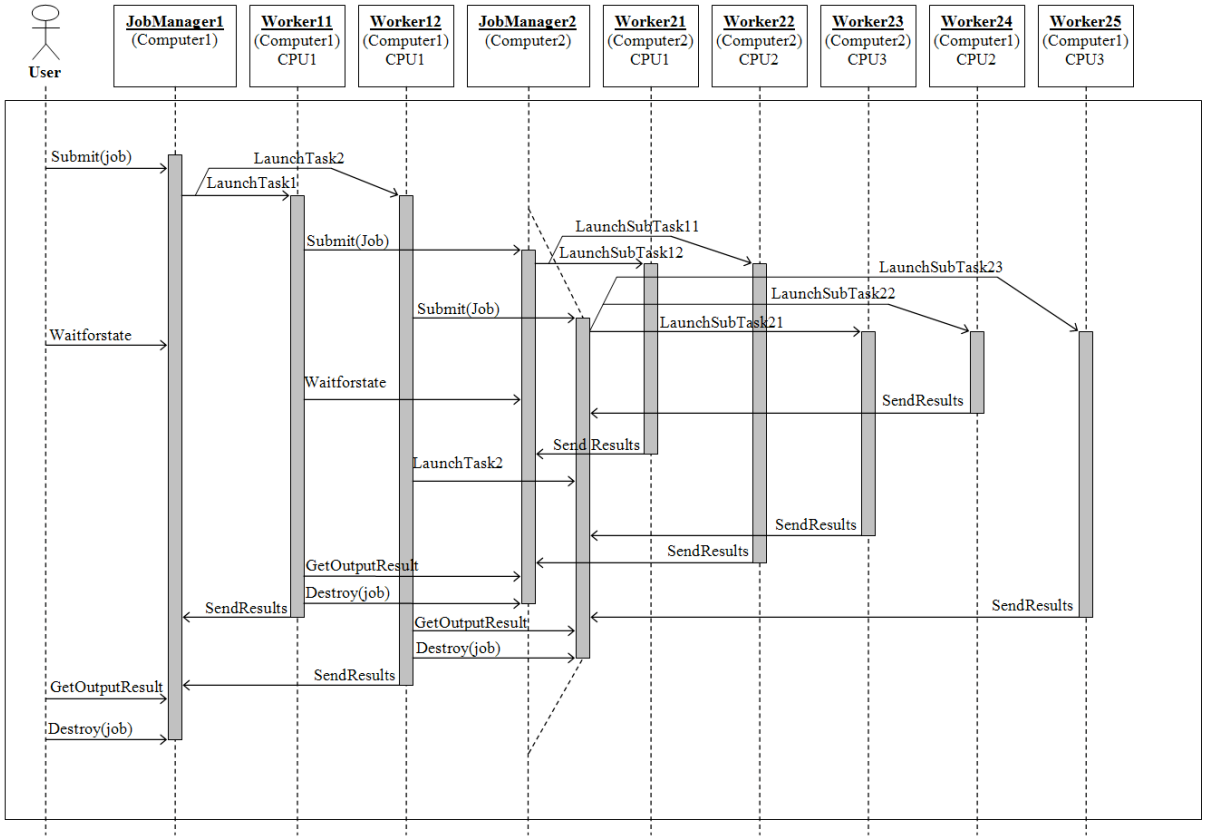


Figure 4-9: Sequence diagram of work distribution

This diagram displays the work repartition on a simplified case. In reality we had sixteen workers for the second job manager, each workers assigned to one core of the two bi-quad core machines. Workers of the first job manager were not two but three and they were all assigned to the same core since the jobs that took most computation power were the jobs given to the second job manager. The presence of multiple job managers is justified by a quick example: e.g. if you have a loop launching the same function multiple times with different parameters and in each function you have again a loop launching another function many times again. The workers associated to the first job manager are

only processing the first function until it comes to the inside loop which launches other functions which will process all the work. This is why the workers associated to the first job manager are all running on the same CPU which will be enough for such simple tasks.

It appeared to be a huge gain of time since for the LDCRF run; it took only three hours instead of 36 hours. Moreover, a script allowed launching additional runs when the current one was finished. So we could launch multiple computations during the night then see and compare the results next morning.

At last, if more machines are available, they can easily be added to the job manager, thus reducing computation time.

4.5 – Experimentation

4.5.1 – Introduction

Experimentation is the final step before deciding which model to use, with which features selection and shape. This process took a lot of time since we had to adapt our data to the functions made by Dr. Morency and to get accustomed with their mechanism.

4.5.2 – Experimentation protocol

Experimentation is a complicated process which needs some organisation; this is why a step by step experimentation protocol is needed for a better understanding:

First step is to select and create the features using the method explained in section 2.1.4. In order to do that, and since you have a wide diversity of features you can select in contextual events, you have to save all the results and their comments in a log file. This is an important point given the number of different features you will have to test. It will guide you to be able to improve your results.

Second step is to test different encoding on the same feature sample for different models. Once this will be done, you will be able to know which shape to use in function of the model you want to try. In order to do that, you have to train for each model all encodings available and compare the results using result visualisation tools to choose the encoding which fits best to this model.

Next step is to compare each model, using best encoding for each, and the feature selection that was proven to be the most relevant, with the help of first step. Visualisation tools exposed in the next section will guide you to the best model for gesture prediction.

At last you will have to confirm the improvement these results brought. By using the generateNodsScript function (see Appendix E) you will be able to create a script to generate agent's head nods using Smartbody in unreal engine [8]. Then, in using a screen capture software, you will be able to generate a movie of these head nods. ELAN in synchronization mode will allow playing it during the same time than the speaker movie to check it generates natural head nods.

4.5.3 – Results

Since one of the specifications of the subject was to have visual tools we had to create such tools to compare the results. However this was not the only reason since visual tools gives you a better intuition of the results and usually, even a better understanding of these.

The specifications of these tools were that given the graphics, you will be immediately able to decide which result is the best and why. However since some models had different outputs we had to keep portability a requirement for all the tools we will be using.

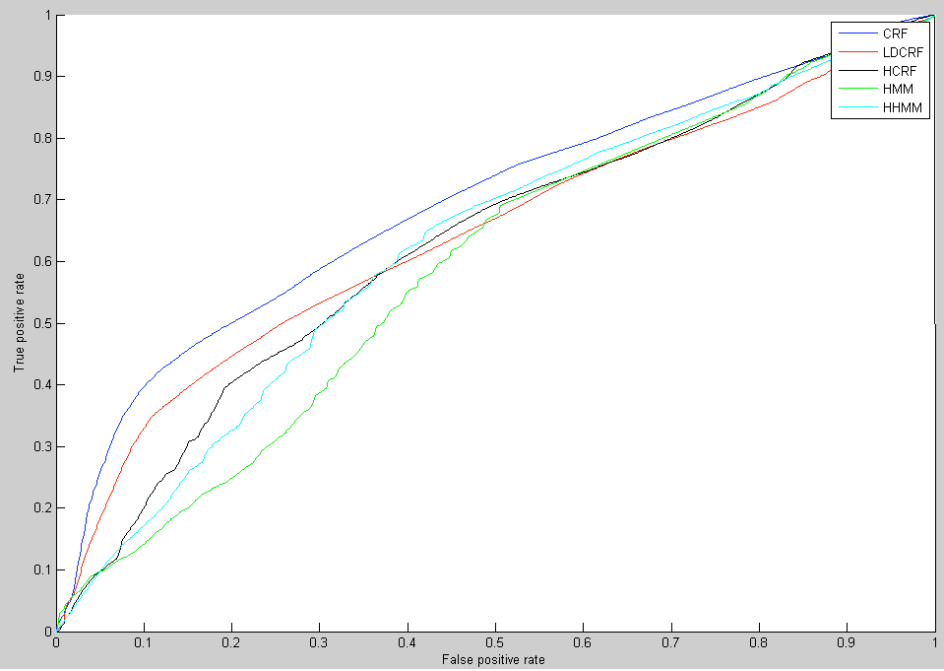


Figure 4-10: ROC curve

First, the Receiver Operating Characteristic curve (ROC), was the first visual output we will have after an experimentation run. A complete description of this curve is available in Appendix D, but the principle could be summarized in a few sentences.

In this case, it takes the likelihood, which is the raw result of this run as an input. Then it creates a vector, which will be the threshold of this likelihood and which will be 1000 samples from the minimum value of the likelihood to the maximum one. For each sample, it will test for each frame if there is a detected head nod, meaning that the likelihood is above the threshold. If there is one, it tests whether it is a real (ground truth) head nod during that time or not. If it is positive, then you have a true positive head nod, otherwise it is a false positive. For the total length it gives you a true positive rate and a false positive rate which are the points of this curve.

This curve appeared to be very helpful to compare results for different features or different encodings for a same feature. However it gave us poor feedback to compare the different models.

Indeed, since head nodding is not an exact science, it is difficult to say if you have a detected head nod which is not concurrent with a ground truth nod, that the detected one is bad. In fact the detected head nod could be well timed, but there are several factors, like personal background, that could cause the listener to not nod at this precise moment whereas a different person could nod at the same moment. This is why we tried to develop another visualisation tool which will take results directly from the raw output of model training, which is likelihood.

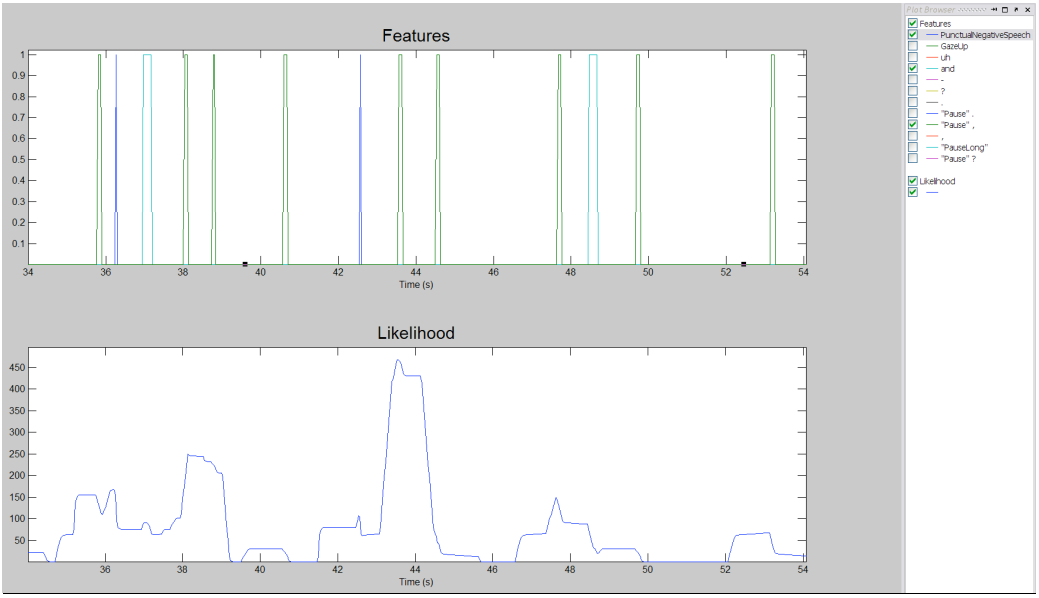


Figure 4-11: Likelihood visualization

First step was to create a visual tool which will allow to see and compare, during a specified sequence, the likelihood with the features you tested. This reveals to be very useful to confirm what the machine learning process learnt. Indeed as you can see in Figure 4-11, you can tick the different features you want to see and check their influence on the likelihood curve.

Once this was done, next step was to create a visual tool that will enable you to see when a head nod is likely to happen, according to the model selected (see Figure 4-12). The goal is here to have a maximum number of matching predictions with ground truth nods.

It comes out that the model that gives us the worst results is SVM. Indeed since it is a non dynamic model, there is no link between the current frame and the previous one. The probability of having a head nod for the current frame is only depending on the observation made at this frame. This is why we quickly abandonned this model which is really slow and which gave us no usable results.

Finally, this is HMM, in opposition to the results ROC curve displayed on Figure 4-10, which gave us the best results. Indeed, this prediction model provided most matching results with ground truth nods, plus additional predictions that needed to be played during the same time than the speaker video, in order to check they seem to be appearing at a moment which looks opportune, thus looking natural.

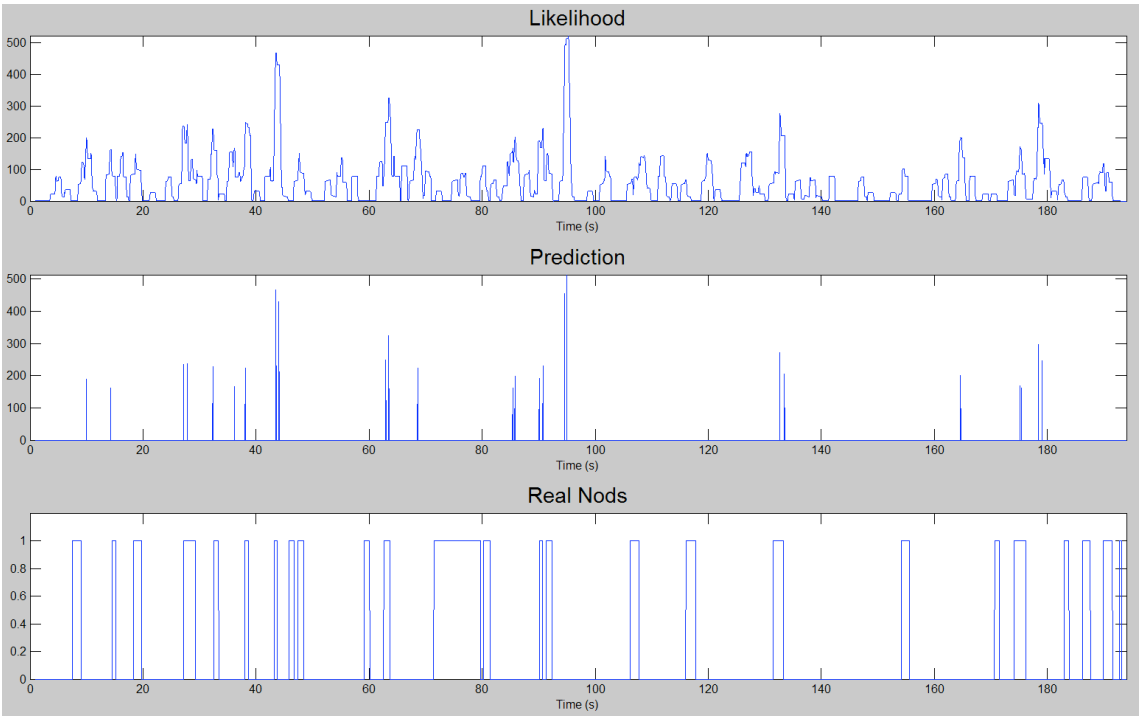


Figure 4-12: Prediction visualization

4.6 – Conclusion

At this point, the prediction toolbox allows going through all steps that leads to gesture prediction, from features selection to prediction model choice. All of this meeting requirements initially exposed.
Moreover this toolbox, originally designed to work with head nods, can be used in order to predict other kind of gesture like head shakes, posture shifts or even facial expressions.

However, some unexpected events made the delivery of this toolbox came later than originally planned. First, the time spent for training models and optimizing these functions was not thought to be this long. Furthermore, some models, like HMM required a lot of memory to be trained, which has been a barrier to our progress at some point.

Even if we did not have enough time to go trough an evaluation process which would be necessary to validate these results, tools are already created in order to start this process, described in the next Chapter. But preliminary results tend to show an improvement of head nods generated by the prediction toolbox, since they seem to appear at a time which looks opportune, and they even support comparison with original listener ones.

Chapter 5 – Future work

5.1 – Head nods classification

In order to characterize head nods we had to find the link between the function and the type of them.

The function of the head nod is the meaning of the gesture and of the motion. We had currently defined different functions of head nods such as continuation, acknowledgement or agreement.
The User study will perhaps add some new functions thanks to the text area where the different users can note those which are not in the list. So it is really by dint of this last study that the head nods functions will be defined and can be used later.

The type of the head nod is its appearance. In fact a head nod can be made rapidly or slowly, with big amplitude or with a small one. It is the clustering which permits to reveal all the different types and which can also give a label for each of them. The label is just a sort of mean for the physics features such as the amplitude or the head nod frequency. Until this point we had just studied the correlation between the speech and the type but it has not given evidential results. So the same study has to be done with another feature as the prosody for example.

The tie between the type and the function will be effective after the examination of the results of the User study. This comes from the fact that the different sorts of clusters have since been found in a previous part and that the User study has been done knowing the cluster's type of the head nod.

5.2 – Specific kind of head nod prediction

Once this work will be done, next step would be to retrain our prediction model for each kind of head nod function. This means we will have to generate a new head nod ground truth for each function. Then, when we will have found a model suitable for each one, we would be able to predict when which head nod should happen.

This task should be faster than the basic head nod prediction since we already trained the methodology. Moreover there should be less experimentation since we already know the best encoding for each feature. Probably some features should be added or removed in function of the kind of head nod. Finally the best model should be the same, since the models tend to behave in the same way with same kind of gestures, even if this has to be confirmed.

5.3 – Merging results

These two tasks achieved, we will be able to improve the agent's non verbal feedback by allowing it to do many kind of head nods. Indeed, we would be able to predict when a kind of head nod should happen and we would be able to link the function of the head nod to its motion pattern.

In fact, since motion pattern regroup information about head velocity like frequency, amplitude and duration, this would allow creating specific animations using Behaviour Markup Language (BML) for the SmartBody system, which is the link between the BML script and the animation you are able to see on the Unreal 3D engine, which is the one used for the agent's rendering.

5.4 – Portability options

Another thing that could be done to improve agent's behaviour should be to use our toolbox with other non verbal feedback. It could be eye gazes, head shakes, posture shifts or even other facial expressions.

Indeed all our work had as a specification to be portable, so the only thing people will have to do is to keep the same format for data input. However for gesture analysis toolbox, another requirement that should meet these behaviours is that they will have to be detected by Watson. For the moment, Watson is only able to give head velocities. But in order to study eye gazes, for example, people will need information about eye directions or any information linked to eye position.

This is why, for the moment, the only Virtual human's movements that could be improved would be the head ones, the other ones will depend on the gesture detection software improvements.

5.5 – Evaluation work

At this point of the work the results of the user study described in Chapter 3 are still not available. Although the entire Web pages and the different scripts are usable, the lack of time has not allowed to experiment it with external people and to study the outcome of this work.

However since most of the research work concerning displaying a movie on a webpage, saving results and creating scripts for randomly picking video for

a study has already been done, this could be adapted to evaluate the prediction toolbox.

Indeed, it would be interesting to create an evaluation of the prediction toolbox on a webpage by comparing previous head nods to the new generated ones. The test should be a selection, after seeing two video sequences of the agent's head nods during speaker speech, of the one which looks more natural than the other, and why. Moreover, a bunch of functions are already designed to create, from predictions obtained by the selected model, a script that can be used to replay the head nods in the unreal engine, using Smartbody. This allows generating video of the head nods using screen capture software. More than that, prediction toolbox already contains tools which make possible the regeneration of the original listener head nods in the unreal engine.
This could be a starting point for an evaluation process which would compare previously generated head nods, listener's ones and the newly generated nods, and should only be an adaptation of the HTML code generated for the previously mentioned User study.

Conclusion

The multimodal toolbox allows users to determine the different appearances of the same gesture and find the relevant contextual features and their link to gestures. The toolbox also enables you to predict the moment of the nodding and the way the head nod should be done. This is relevant to underline that the entire study is adaptable to other gestures such as the shakes or the eye gazes.

This toolbox also meets the requirements mentioned in section 1.1.2. Most of the results can be displayed with plots which allow a better understanding of different gesture patterns and a better intuition of relation between contextual features and head nod appearance. We created generic functions to import any type of gestures and the architecture of our toolbox can handle these new gestures. We created an extensive documentation of the toolbox (see Appendix E) so that future users can easily understand and extend the functions.

Since the toolbox is going to be used for research, we had to test several techniques to reach the same goal. This can be seen as a “code & fix” approach. Although this is inefficient for the multimodal toolbox development, this model proved to be necessary to build functions that will be a part of ongoing research.

This internship has allowed us to work in a diverse research environment and to explore how to improve the non-verbal behaviours of virtual humans. This taught us how to coordinate a large project with many subtasks. We also learned how to organize and prioritize our tasks and to be able to present the status of our work.

Finally, working in a research environment gave us the opportunity to expand our knowledge beyond engineering and exposed us to other research areas, such as psychology and language.

References

Books

[1] J.A. HARTIGAN
Clustering Algorithms
John Wiley & Sons, Inc.
1975

[2] E.O. BRIGHAM
The fast Fourier transform and its applications
Prentice-Hall, Inc.
1988

[3] V. VAPNIK
The nature of statistical learning theory
Springer
1995

Papers

[4] S. IMAI
"Cepstral analysis synthesis on the mel frequency scale"
IEEE International Conference on Acoustics, Speech, and Signal Processing
April 1983

[5] L. R. RABINER
"A tutorial on hidden markov models and selected applications in speech recognition"
Proceedings IEEE, volume 77, pages 257–286
February 1989

[6] J. LAFFERTY, A. McCALLUM, and F. PEREIRA
"Conditional random fields: probabilistic models for segmenting and labeling sequence data"
International Conference on Machine Learning
2001

[7] L.-P. MORENCY, A. QUATTONI and T. DARRELL
"Latent-Dynamic Discriminative Models for Continuous Gesture Recognition"
Proceedings IEEE, Conference on Computer Vision and Pattern Recognition
June 2007

Reports

[8] F. LAMOTHE, M. MORALES
Response Behavior
Institute for Creative Technologies
2005

[9] R.J. VAN DER WERF,
Creating Rapport with virtual humans
Institute for Creative Technologies
2006

[10] C. VERSLOOT, M. TER MAAT,
The Rapport project
Institute for Creative Technologies
2006

Electronic resources

SASO wikipedia website
http://sasowiki.ict.usc.edu/index.php/Main_Page

Transcriber official website
http://trans.sourceforge.net/

ELAN official website
http://www.lat-mpi.eu/tools/elan/

Watson official website
http://groups.csail.mit.edu/vision/vip/watson/

University of Leeds Hidden Markov Models tutorial:
http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html

Appendix

A – Tests Details

There are three different ways the original user studies were realized. What these three tests have in common is the fact that the two subjects are sat across a table. The subjects, the listener and the speaker, are in all conditions posted at the same distance from each other.

The first test, called “face-to-face”, shows the listener and the speaker who can see each other. In that case there is no screen or monitors between them.

The second and the third tests, called “responsive” and “unresponsive”, are characterised by the fact that the listener and the speaker are separated by a screen. Thus they can not see them directly but the listener can hear the speaker and see the images of him. The speaker can himself see the avatar on his monitor.

The main difference between the “responsive” and the “unresponsive” conditions is that in the “responsive” condition the avatar is controlled by the Rapport agent. The Rapport agent is a Virtual human which is able to establish contacts with a human participant thanks to non verbal behaviour feedbacks such as posture mirroring, nods or shakes. It means that in this case the speaker will speak in front of an avatar which owns human characteristics and which applies them at the good time. In the “responsive” case the avatar is able to create an impression of listening, by displaying different non verbal behaviours.
In the “unresponsive” case the avatar is directed by a pre-recorded random script. This script is thus totally independent from the speaker’s behaviours. This script has also the particularity that the avatar will do neither head nods nor head shakes but just several head turns and postures shifts. This last study permit after the comparison with the “responsive” test to check if the speaker is more active when he speaks with the Rapport agent than face to an avatar which has no real human behaviours.

B – Software

Transcriber

Transcriber is annotation software developed by the French DGA. It provides you a graphical user interface to add annotation to an audio file. Moreover it allows you to display the waveform of the audio file, to segment annotations and to create context events with an additional line. This extra feature was not used in this study. This is free software developed under GNU General Public License, and is compatible with Windows, Mac OS X and Linux platforms.

Watson

This software was developed by Louis-Philippe Morency while he was working in the Vision Interface Group at the Computer Sciences and Artificial Laboratory of the MIT (Massachusetts).

It permits to track the head's position and orientation in real time thanks to its own tracking library. This software opens the possibility of detecting the head nods and head shakes by use of a head velocities' vector. This vector is made of three values, according to each axis.

The last version of Watson software can also track with bounded drift the six degrees-of-freedom of the head for a long period and can use a monocular camera to estimate the head's orientation and position.

File's organization for each session:
Each studied session possesses its own file containing:

- An images file where all the images treated by software are stocked
- A result file containing .txt documents such as eye.txt (the eye study by Watson), nods.txt, poses.txt or velocity.txt. All these documents are made in the shape of matrix with well separated columns. It permits to easily import all these data in a MATLAB file for the future studies.
- A callib.ini document which is the document where all the configuration of software are saved (colors, brightness…)
- The output.avi file which is the same video as the input video but a square frame around the head is added.
- Three documents for the uses of Watson: ParamSeq.cfg, ParamWatson.cfg and ParamWatsonUser.cfg
- The shortcut for software running.

ELAN

This free available software is a tool witch permits to create annotations of all the video and audio data. It can also be used to edit, search or even visualize the annotation since existing. This tool was developed by the Max Planck Institute for Psycholinguistics, The Netherlands. This software permits to realize some different works such as visualize the video data as the same time as the audio data, observe the duration of an action thanks to a time line or shift in the video or in the audio file easily. ELAN has no defined numbers of annotations and owns a large set of characters that permit to realize some different work on the same page. It is also possible to watch more then one video at the same time or to choose the audio input.

Virtual Dub

Virtual dub is software licensed under the GNU General Public License. It permits to capture video and also to process them with the entire 32-bit Windows platform. This free software has batch-processing capabilities that allow processing a large numbers of files at the same time. It mains goal is to deal with the .avi files but it can also read .mpg and work with Bitmap images.

Its principal utilisation is to extract or cut some sequences in a video file such as commercial or trailers at the end of films. But it is also used to compress with some filters video sequences. You can either add image on your video (such as logo) or adjust sound files with the video.

Its scripts must be saved on .job files and are directly opened by software with “run script”. All the language can easily be found on the Internet.

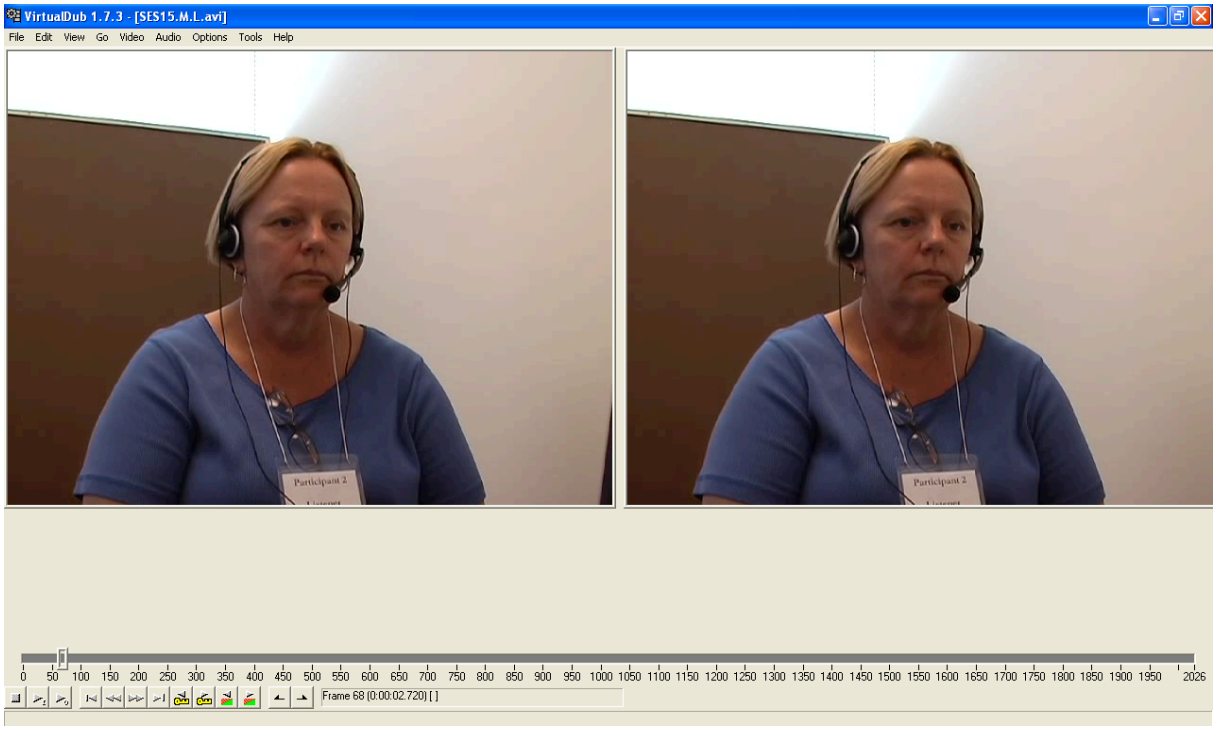


Figure B-1 Virtual dub main page

As you can see it on the bottom of Figure B-1 it gives the possibility to adjust manually the timeline. That permits to extract a sequence in a video close to the milliseconds or to the frame.

The two pictures on this figure are the original video on the right and on the left the video after transformation or extraction.

C – Transcriptions Symbols Reference

INTONATION		
,		continuing intonation
.		falling intonation
?		rising intonation
PAUSE		
/		micro pause (less than 150 milliseconds)
//		pause (___ seconds)
///		pause (___ seconds)
PRONUNCIATION		
_	ex_a_c_tly	slower or emphasized
:	i li::ke it	lengthening; the more colons, the more elongation
-	jona-	incomplete word
VOLUME		
*** ***	*be quiet*	softer speech or whisper

D – Receiver operating characteristic (ROC)

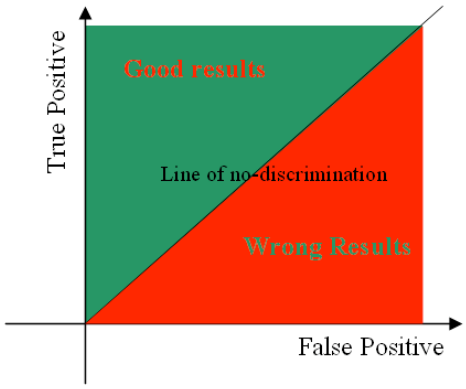
A ROC curve is the plotting of the true positive according to the false positive values of a study. The True Positive values (TP) are the values which are detected and which really appear whereas the False Positive values (FP) are the values which are not detected but which really appear. It can be resumed in a table where each condition will carry a different name.

		Actual Condition	
		Present	Absent
Test Result	Positive	Condition Present and Positive Result = True Positive	Condition Absent and Positive Result = False Positive
	Negative	Condition Present and Negative Result = False Negative	Condition Absent and Negative Result = True Negative

The TP values will be defined by the Y-axis whereas the FP values will be defined by the X-axis.

There are different zones in a ROC curves since this one is cut by a 45 degrees diagonal line from the left low corner to the right top corner. The line is a so called line of no-discrimination. So if you have points on this line you can not say if they are good or not.
All the points above the line of no-discrimination are good results whereas all the points below are wrong results.

In fact a point which is situated on the left top corner represents the best result because it means that this point own 100 percents of the sensitivity and 100 percents of the specificity.
The sensitivity is correlate with the number of TP which are found whereas the specificity is correlate with the number of FP which are not been found.



E – Detail of the API

Original Variables

All these variables can be loaded using *load 'RapportWatson06Vars.mat'* command

CaptionAction: Cell containing the caption of ELVIN actions understudy

CaptionFeats: Cell containing the caption of all used punctuations symbols understudy

Nods: Ground Truth Nods array (for each nod: Seq #, Nod start time, Nod stop time)

OffsetDing: Array containing in the first column the offset of each session for Transcriber files, and in the second column the offset of each session for Watson files. These offset are relative to the original video files recorded with the DV Camera.

Paths: Cell containing the full name of each file understudy (e.g. SES10.N.L)

PathsTranscriber: Cell containing the directory of each session understudy (e.g. 20061016_130105-10-16-2006-SES10.N)

params: Structure containing all the necessary parameters for the gesture analysis toolbox (these parameters remains in the input of all functions)

paramsData : Structure containing all the necessary parameters for the runValidateSpeech function to be launched

paramsDistributed: Structure containing all the necessary parameters for the Distributed functions to be launched

paramsNodCRF, paramsNodHCRF, paramsNodHHMM, paramsNodHMM, paramsNodLDCRF, paramsNodSVM: Structure containing necessary parameters for a specific prediction model.

Functions

Data-Processing

extraction
This function extracts the ts and the rot values for all the real head Nods

INPUT:
- realNods= Ground Truth head nods
- ts= timestamp
- rot= rot found by Watson
- params= params.delay (launch script.m)

OUTPUT:
- extractts= the ts of the realNods head
- extractrot= the rot values of the real head nods

---

extractNods
This function returns the real nods (detected by Watson and the Ground Truth one) with the timestamp

INPUT:
- rot= degree of rotation (around x...)
- ts=timestamp
- realNods= ground truth head nods= nods found by ourselves

OUTPUT:
- rot2=degree of rotation (around x...)
- ts2=timestamp
- realNods=Ground Truth Nods file(Seq #,Nod start time, Nod stop time,[True Positive Nod])

---

findSpeechNods
This function gives you which Nods are out of speech, during no speech at the beginning or no speech at the end of the Nod

INPUT:
- events=contains begin, end and index of all sentences for each session
- realNods= Grown Truth Nods file(Seq #,Nod start time, Nod stop time,True Positive Nod)

OUTPUT:
- NoSpeech=Vector which contains for each nod whether it is during speech or not
- StartNoSpeech=Vector which contains for each nod whether its start it is during speech or not
- EndNoSpeech=Vector which contains for each nod whether its end is during speech or not

---

importArrayToActions
This function converts into the actions var any feature added with ELAN and imported by readELAN

INPUT:
- Array=contains session index, begin and end of all additional feature annotated by ELAN for each session
- NameOfFeature=Name of the additional feature put in the Array var
- actions=cell array containing the actions of all sessions
- CaptionAction=cell array containing the caption of all actions

OUTPUT:
- actions=cell {actiontype, session} contains begin and end of all actions for each session
- CaptionAction=cell array containing the caption of all actions

importEventsBigramsToActions
This function extracts bigrams and their timestamps from events utterances

INPUT:

- events=contains begin, end and index of all sentences for each session
- actions=cell array containing the actions of all sessions
- utts=contains all sentences for each session
- CaptionAction=cell array containing the caption of all actions

OUTPUT:

- actions=cell {actiontype, session}contains begin and end of all actions for each session
- CaptionAction=cell array containing the caption of all actions

---

importEventsPausesToActions

This function extracts pauses and their timestamps from events utterances

INPUT:

- events=contains begin, end and index of all sentences for each session
- actions=cell array containing the actions of all sessions
- utts=contains all sentences for each session
- CaptionAction=cell array containing the caption of all actions

OUTPUT:

- actions=cell {actiontype, session}contains begin and end of all actions for each session
- CaptionAction=cell array containing the caption of all actions

---

importEventsPunctToActions

This function extracts punctuation and their timestamps from events utterances

INPUT:

- events=contains begin, end and index of all sentences for each session
- actions=cell array containing the actions of all sessions
- utts=contains all sentences for each session
- CaptionAction=cell array containing the caption of all actions
- CaptionFeat=cell array containing the caption of all used punctuations symbols

OUTPUT:

- actions=cell {actiontype, session}contains begin and end of all actions for each session
- CaptionAction=cell array containing the caption of all actions

---

importEventsToActions

This function extracts unigrams and their timestamps from events utterances

INPUT:

- events=contains begin, end and index of all sentences for each session
- actions=cell array containing the actions of all sessions
- utts=contains all sentences for each session
- CaptionAction=cell array containing the caption of all actions

OUTPUT:

- actions=cell {actiontype, session}contains begin and end of all actions for each session
- CaptionAction=cell array containing the caption of all actions

readWatsonData

This function extract data ((rotations or head nods) and timestamps) of each file from \\Sfs\data\public\RapportWatson\

INPUT:

- dirData= the files where data are. Put a "\" at the end of dirData
- Paths= file containing the path of all sessions you want to study
- file=name of the studied file ('nods' or 'velocities')

OUTPUT:

- data=degree of rotation (around x...) for 'velocities' and head nods for 'nods'
- ts=timestamp

---

readELAN

This function extracts any feature annotated with ELAN files (.eaf)

INPUT:

- dirData=the directory where .eaf files are. Put a "\" at the end of dirData
- Paths=file containing the path of all sessions you want to study

OUTPUT:

- importedarray=contains begin, end and sequence number of all annotations made in ELAN

---

readElvin

This function extracts Elvin events from Elvin logs already existing in \\Sfs\data\rapport\subject-tests\rapport-oct-2006\

INPUT:

- dirData=the directory where data are. Put a "\" at the end of dirData
- PathsTrans=cell array containing the path of all sessions you want to study
- CaptionAction=cell array containing the caption of all actions

OUTPUT:

- actions=cell {actiontype,sessionnumber}contains begin and end of all actions for each session

---

readTrans

This function extracts sentences and their timestamps of each .trs file from \\Sfs\data\rapport\subject-tests\rapport-oct-2006\

INPUT:

- dirData=the directory where data are. Put a "\" at the end of dirData
- PathsTrans=file containing the path of all sessions you want to study

OUTPUT:

- events=contains begin, end and index of all sentences for each session
- utts=contains all sentences for each session

---

shiftTimeStamp

This function removes the offset (find with the video) from the ts

INPUT:

- ts=timestamp find with readData
- offset=difference of time between head nods found by Watson and the Ground Truth one (in OffsetDing.mat)

OUTPUT:

- ts=ts-offset

window

This function extracts the ts and the rot values for all the real head Nods in a window (# of frame you choose) centered at the head nod

INPUT:

- realNods= ground truth head nods

- ts=timestamp
- rot= rot found by Watson
- params=params.srate, params.lengthWindows(launch script.m)

OUTPUT:

- windowts=the ts of the realheadnods in an window (# of frame you choose)
- windowrot=the rot values of the real head nods in an window (# of frame you choose)

---

windowafterextraction

This function extracts the ts and the rot values in an window after the extraction

INPUT:

- ts=timestamp
- rot= rot found by Watson
- params= params.srate, params.lengthWindows (launch script.m)

OUTPUT:

- windowextracts=the ts of the realheadnods in an window
- windowextractrot=the rot values of the real head nods in an window

Gesture Analysis Toolbox

Data Analysis

analyseWatsonPerformance

This function finds the Nods detected by Watson (which are between our Start Nod and End Nod), the number of Ground Truth head nods and the false detection of Watson

INPUT:

- realNods= ground truth head nods=nods found by a person
- ts=timestamp
- dataNod= head nod of the 'nods.txt' file
- thresh= number you give

OUTPUT:

- nbGestureDetected=find by Watson between our limits
- nbGestureTotal=ground truth head nod
- falseDetection= # of time Watson head nods were out of our limits
- totalFalseGesture= # of frames what are not gesture from ground truth

---

createROC

This function traces the ROC curve TruePositiveRate=f (FalsePositiveRate)

INPUT:

- rangeThresh= give the begin and the end of the study (e.g, -1:0.1:1)
- realNods= Ground Truth head nods
- ts= timestamp
- rot= degree of rotation (around x...)

Data Processing

cluster

This function makes the clustering of a matrix you give in input

INPUT:

- particularot=the rot you want to cluster (a cell array)

- params= params.clusternumber

OUTPUT:

- clusterot= a column where the # of cluster are written
- index= mask of valid gestures

---

filterR

This function filters one column

INPUT:

- data= the variable you want to study(extractrot, sampledrot, windowrot or rot)

OUTPUT:

- particularot= the variable (x, y, z) you want to keep in columns

---

removeCellMean

This function calculates the mean of each column of the cell input and subtracts it to each column.

INPUT:

- fftrot= the cell you want to remove the mean

OUTPUT:

- fftrotremoved= the input without the mean of each column

---

removeMatrixMean

This function calculates the mean of each row of the matrix input and
subtracts it to each row.

INPUT:

- freq= the matrix you want to remove the mean

OUTPUT:

- freqremoved= the input without the mean of each row

---

reajustNbCol

This functions permits to verify if each elements have the same number of terms. And it removes them which are overmuch.

INPUT:

- rot= the variable you want to study (extractrot, sampledrot, windowrot or rot)

OUTPUT:

- transrot= the variable after transformation

Resampling

resample

This function makes the resampling of values (rot,ts). It uses the function 'sampling.m' created by L.P Morency

INPUT:

- rot= the variable you want to resample (extractrot, windowrot...)
- ts= the ts you want to resample (extractts, windowts...)
- params= params.srate (launch script.m)

OUTPUT:

- sampledrot= the variables after resampling
- sampledts= the ts after resampling

Fast Fourier Transform

meancluster

This function calculates the mean of each column of particularot according to its clusters and plots these values

INPUT:

- particularot= a cell (same length of the values in column) you want to study
- clusterot= a column where the # of cluster are written
- params= params.legthWindows

OUTPUT:

- meanfft= a cell for each cluster with all the means of the columns

---

fftrequency

This function calculates the fft. So we can obtain the frequency value of the input.

INPUT:

- rot= the term you want to study (it is a cell)
- params= all the parameters of the different functions

OUTPUT:

- frequency= it is a matrix where each column represents a frequency (FE coulumn3=2Hz) and each row represents a head nod.

Principal component Analysis

clusterSVD

This function finds the # of elements of S needed to reach the threshold (called counter), after it clusters this # of column of U (coefficient of svd) and plots it with colors (depend on cluster case)

INPUT:

- S= output of svDict, singular values
- U= output of svDict, called coefficient
- params= params.clusternumber, params.eigenvect1, params.eigenvect2

OUTPUT:

- clusterSVD= a column which contains the clusters #
- counter= # of rows needed to reach the threshold (for the function ClusterSVDmean)

---

mySVD

This function calculates the Singular value decomposition

INPUT:

- frequency= the term you want to factorize (spectral theorem)

OUTPUT:

- S= contains the singular values, which can be thought of as scalar "gain controls"
- U= contains a set of orthonormal "output" basis vector directions
- V= contains a set of orthonormal "input" basis vector

Plotting

basisplotting

This function plots the coordinate of 2 bases of the V vector (you choose them in params)

INPUT:

- clusterot= from cluster function

- params= params.clusternumber, params.lengthWindows, params.eigenvect1, params.eigenvect2
- V= output of mySVD, gives direction of basis

---

freqCoefplotting
This function plots either the frequency of a basis according to the frequency of another basis or the coefficient of U matrix (see mySVD) of a basis according to the coefficient of another basis

INPUT:

- clusterot= from cluster function
- params= params.clusternumber, params.ix, params.iy
- matrix= the matrix you want to study (e.g, freq or U)

---

plotCluster
This function plots the time of speech, the type of Nods and the head velocity

INPUT:

- session= index of desired session
- events= output of the readTrans function
- ts= timestamp
- rot= degree of rotation (around x...)
- clusterrot= output of cluster

---

plotRangeNods
This function plots the time of speech, the head velocity for one case of all the head nods

INPUT:

- realNods= ground truth head nods= the nods found by ourselves
- events= output of the readTrans function
- ts= timestamp
- rot= degree of rotation (around x...)
- clusterlength(created with kMeansCluster)= must be load in current directory
- rangeNods= chose the studied HeadNods (FE 10:110-> head nod 10 to head nod 110)
- cas= the cluster (type of head nod) you want to study
- params= params.case, paramas.delay, params.column, params.row in script.m

---

plotSVDmean
This function calculates the mean of each column of U according to its clusters. And it plots a1*V1+a2*V2... according to each cluster (V1 is the column 1 of V, we consider a # of columns gave by counter and a1 is the first term of SVDmean)

INPUT:

- clusterrot= a column which contains the clusters # (apply on the mySVD output)
- counter= # of rows needed to reach the threshold
- U= output of SVDict, called coefficient
- V= contains a set of orthonormal "input" basis vector directions
- params= params.clusternumber

OUTPUT:

- SVDmean= a cell which contains all the means of the U columns (according to clusters)

plotvelocitysample (tsample, rotstudied, params)
This function plots the ts, rx (...) of the head nods found by Watson and by us (the values have to be resampled and are in a 'window')

INPUT:

- tsample= ts which have been resampled and are in the 'window'
- rotstudied= a cell array with 3 columns containing rx, ry and rz (output of extraction)
- params=params.column, params.row in script.m

---

plottingVelSpeechFFT
This function plots on the same figure velocity, speech and fft of a case of head nod

INPUT:

- clusterot= a column with the # of the cluster
- params= all the parameters in script.m
- events= output of the readTrans function
- windowextractrot= the rot values of the real head nods in an window
- windowextracts= the ts of the realheadnods in an window
- realNods= ground truth head nods
- particularot= the chosen fft (fft 'x', fft 'y'...)

index= mask of valid gestures

---

segmentplot

This function plots the time of speech, the head velocity for one cluster of head nod

INPUT:

- realNods= the Ground Truth nods
- session= index of the desired session
- events= output of the readTrans function
- ts= timestamp
- rot= degree of rotation (around x...)
- clusterrot= output of cluster
- cas= the cluster (type of head nod) you want to study
- params= params.case, params.delay in script.m

Web Pages

headnodpersession

This function gives in a column vector the number of head nods per sessions. The result is used during the Virtualdub script creation.

INPUT:

- realNods= ground truth head nods= nods found by ourselves

OUTPUT:

- numberheadnod= a matrix where each element represents the number of head nod for a session

---

scriptVirtualdubCreator

This function permits to create the scripts for Virtualdub (to extract just head nods and put a frame around the head)

INPUT:

- Paths= where all the paths of the sequences (e.g, SES 2) are
- realNods= ground truth head nods= nods found by ourselves

videoStudy

This function permits to extract the time of beginning, end and duration of the real head nods (in ms) for the Virtualdub script.

INPUT:

- Paths= where all the paths of the sequences (e.g, SES 2) are
- realNods= ground truth head nods= nods found by ourselves

OUTPUT:

- start= time of the start of the head nod (in ms)
- stop= end of the head nod from the end of the sequence (in ms)
- time= duration of the head nod

webScript
This function permits to create the scripts needed in the web page study
INPUT:
- Paths= where all the paths of the sequences (e.g, SES 2) are
- realNods= ground truth head nods= nods found by ourselves
- params= params.nbrofseq, params.nbroscript, params.delayVideo, params.nbroftrainvideo (launch script.m)

Prediction Toolbox

Select and visualize Features

computeHistogram
This function traces histograms of actions happening before Nods
INPUT:
- realNods=Ground Truth Nods array(Seq #,Nod start time, Nod stop time,[True Positive Nod])
- actions=cell array containing the actions of all sessions
- window=time before the Nod to allow the action to be taken into account
- CaptionAction=cell array containing the caption of all actions
- NbofElvinFeats=Number of Elvin features in the CaptionAction cell array
- mincount=minimum occurrence in total before nods to be taken into account

---

computeWordGraph
This function computes the number of Nods occurring during the time window before and after the words which are in wordlist. The final output is the bin index of maximum and minimum peak
INPUT:
- realNods=Ground Truth Nods array(Seq #,Nod start time, Nod stop time,[True Positive Nod])
- wordlist=cell array of all the words you want to study (e.g. {'and','"PauseLong"'})
- actions=cell array containing the actions of all sessions
- window=time before and after the words which are in wordlist to take Nods into account
- CaptionAction=cell array containing the caption of all actions
- bincount=number of subdivisions in the time window (resolution of the window)

OUTPUT:
- maxis=bin index of maximum peak
- minis=bin index of minimum peak

plotWordGraph
This function traces the number of Nods occuring during the time window before and after the words which are in wordlist
INPUT:
- realNods=Ground Truth Nods array(Seq #,Nod start time, Nod stop time,[True Positive Nod])
- wordlist=cell array of all the words you want to study
- actions=cell array containing the actions of all sessions
- window=time before and after the words which are in wordlist to take Nods into account
- CaptionAction=cell array containing the caption of all actions
- bincount= number of subdivisions in the time window (resolution of the window)

---

createTopFeatures
This function selects automatically the top features from the ones that appears most frequently before a nod

INPUT:

- nbTop=number of features you want to be selected
- actions=cell array containing the actions of all sessions
- realNods=array containing ground truth basis of the head nods
- timeStamps=Timestamps created with Create Timestamps or previously created ones
- window=time before the Nod to allow the action to be taken into account
- CaptionAction=cell array containing the caption of all actions

OUTPUT:

- features=cell array containing all the top features generated by this function
- wordlist=cell array containing the caption of associated features

---

selectFeatures
This function selects automatically the best features from top50features created previously

INPUT:

- nbFeats=number of features you want to be selected
- top50features=top features created with createTopFeatures function
- machineName=machine on which the computations will be done
- paramsData, paramsNodCRF=files containing all parameters for runValidateSpeech
- realNods=array containing ground truth basis of the head nods
- timeStamps=Timestamps created with Create Timestamps or previously created ones
- nbIterations=number of iterations for the model testing
- wordlist=cell array containing all the features contained in top50features

OUTPUT:

- features=cell array containing all the extracted features from actions
- files with the wordlist selection

---

selectFeatures2
This function selects automatically the best features from top50features created previously. This one selects using the computeWordGraph function

INPUT:

- nbFeats=number of features you want to be selected
- top50features=top features created with createTopFeatures function
- machineName=machine on which the computations will be done
- paramsData, paramsNodCRF=files containing all parameters for runValidateSpeech
- realNods=array containing ground truth basis of the head nods
- timeStamps=Timestamps created with Create Timestamps or previously created ones
- nbIterations=number of iterations for the model testing
- wordlist=cell array containing all the features contained in top50features

OUTPUT:

- features=cell array containing all the extracted features from actions files with the wordlist selection

---

selectFeaturesDistributed
This function selects automatically the best features from top50features previously, distributed computing version

INPUT:

- nbFeats=number of features you want to be selected
- top50features=top features created with createTopFeatures function
- machineName=machine on which the computations will be done
- paramsData, paramsNodCRF=files containing all parameters for runValidateSpeech
- realNods=array containing ground truth basis of the head nods

- timeStamps=Timestamps created with Create Timestamps or previously created ones
- wordlist=cell array containing all the features contained in top50features
- paramsDistributed=parameters containing all necessary info to run in distributed mode
- nbIterations=number of iterations for the model testing

OUTPUT:

- features=cell array containing all the extracted features from actions files with the wordlist selection

Encode Features

createActions

This function creates actions from elvin files and from unigrams, bigrams, punctuation and pauses of events

INPUT:

- dirData=directory where all session files containing elvin files are. Put a "\" at the end of dirData
- PathsTrans=cell array containing the path of all sessions you want to study
- events=contains begin, end and index of all sentences for each session
- utts=contains all sentences for each session
- CaptionAction=cell array containing the caption of all actions
- Gazes=array containing the annotation of all Gazes or any other annotation array imported with the readELAN function

OUTPUT:

- actions=cell {actiontype, session}contains begin and end of all actions for each session
- CaptionAction=cell array containing the caption of all actions

---

createBinaryFeatures

This function creates features from actions cell array, selecting only the items contained in the wordlist

INPUT:

- Wordlist=cell array containing all the features you want to be extracted from actions cell array
- timeStamps=Timestamps created with Create Timestamps or previously created ones
- CaptionAction=cell array containing the caption of all actions
- actions=cell array containing the actions of all sessions

OUTPUT:

- features=cell array containing all the extracted features from actions files with the wordlist selection

createFeatureShape

This function creates different shape for the features from the binary original one

Binary original one

INPUT:

- features=cell array containing all the extracted features from actions files with the wordlist selection (binary shape)
- shapetype=0 corresponds to binary (no change),1 corresponds to ramp shape and 2 corresponds to Gaussian shape
- window=number of frames you want for the Gaussian or ramp duration (put 0 to have proportional with the size of original binary feature)
- side=0 corresponds to symmetrical feature, -1 Left side and 1 Right side.

OUTPUT:

- features=gives you back the features files with the new selected shape.

Generate Models

createLabels
This function creates labels in the same format that the ones generated by the function createTrainTestAndValidationData in runValidateSpeech function

INPUT:
- actions=cell array containing the actions of all sessions
- Nods= Ground Truth Nods array (Seq #,Nod start time, Nod stop time)
- framerate =Number of frame per seconds you want to be generated

OUTPUT:
- labels=cell array where each cell is a sequence and every cell contains the same number of frames that are contained in the corresponding feature cell array

---

createTimeStamps
This function simply create timestamps for every sessions for a given length and frame rate

INPUT:
- actions=cell array containing the actions of all sessions
- framerate=number of frame per second you want for your timestamps
- Length=array containing all the length of each sequence

OUTPUT:
- timeStamps contains all timestamps by sequence where each sequence is a cell.

---

launchExperimentations
This script is only a launcher for different runs, with multiple features, models and params, in order to see put the results in a same BR variable, and to see the differences on a ROC graph

---

runValidateSpeechDistributed
This function is a distributed version of the original function developed by Louis Philippe Morency to generate (Train, test and validate) models using machine learning techniques

INPUT:
- params=configuration structure containing parameters specific to each kind of machine learning techniques you will be using(HMM,CRF,SVM...)
- paramsData = configuration structure containing parameters that are needed by this function, and non specific to the kind of technique used
- features= cell array containing the features used for the generation of this model. Same kind of cell array that the one generated by the function createFeatures
- realNods= Ground Truth Nods array (Seq #,Nod start time, Nod stop time,True Positive Nod)
- timestamps = TimeStamps corresponding to the features and labels vars
- paramsDistributed= configuration file containing the parameters required to run on a distributed mode
- nbIterations=keep empty by default

OUTPUT:
- BR=Best result of all the runs conducted by this function
- R=Every result of each run conducted by this function
- D=Data associated to the best result cell array containing details about the different runs

and how they were conducted

---

testCombinations
This function, called by trainTestAndValidateDistributed, is a distributed version of the original function developed by Louis Philippe Morency to generate (Train, test and validate) models using machine learning techniques. This one is only one part of the original trainTestAndValidate function which has been implemented in a function in order to make work distribution easier.

INPUT:
n/a

OUTPUT:
- R=Every result of each run conducted by this function

---

trainTestAndValidateDistributed
This function, called by runValidateSpeechDistributed, is a distributed version of the original function developed by Louis Philippe Morency to generate (Train, test and validate) models using machine learning techniques

INPUT:
- Data=Data containing details about the different runs and how they are going to be conducted
- params=configuration file containing parameters specific to each kind of machine learning techniques you will be using(HMM,CRF,SVM...)
- externaltag=tag corresponding to the string of the iteration number
- paramsDistributed=configuration file containing the parameters required to run on a distributed mode

OUTPUT:
- BR=Best result of all the runs conducted by this function
- R=Every result of each run conducted by this function
- bestIndex=contains the index of the best results

Visualize Results

compareResults
This function compares the likelihood of the different results to the Head Nod Ground Truth

INPUT:
- BR1=Best Results from the first experience you want to compare
- BR2=Best Results from the second experience you want to compare
- BR3=Best Results from the third experience you want to compare
- D1=Output from createTrainTestAndValidationData in the first experience
- D2=Output from createTrainTestAndValidationData in the second experience
- D3=Output from createTrainTestAndValidationData in the third experience
- timeStamps=TimeStamps corresponding to the features and labels vars
- seq=The sequence number you want to see
- labels=Cell array created with createLabels fonction
- tresh=Vector containing different thresholds you want to put on likelihood

-

---

createNodsPrediction
This function creates, with the help of a model, a certain number of head nods predictions. This function is coupled with the generateNodsScript function

INPUT:

- BR=Best Results from the model you want to use
- D=Output from createTrainTestAndValidationData in the model generation function
- seq=The sequence number you want to study
- timeStamps=Timestamps corresponding to the features and labels vars, format is the same as the output of createTimeStamps function
- nbheadnods=number of head nods you want to be generated

OUTPUT:

- probs=Represents the head nod prediction, depending of the time
- ll= Represents the likelihood of a head nod, depending of the time
- xindex=timestamp for the x axis
-

---

generateNodsScript
This function creates the script that will be used to replay the head nods generated with the function createNodsPrediction

INPUT:

- probs=first output variable of CreateNodsPrediction
- seq=The sequence number you want to study
- timeStamps=TimeStamps corresponding to the features and labels vars, the format is the same as the output of createTimeStamps function

OUTPUT:

A file will be generated in the same path that you are working on, with the following title format: yyyymmdd_hhmmss-mm-dd-yyyy-test-elvin_log.txt

---

plotLikelihood
This function plots the raw likelihood coming from a run of runValidateSpeech (Distributed or not) compared to the features appearances on a time scale

INPUT:

- BR=Best result of all the runs conducted by runValidateSpeech(Distributed)
- D=Data associated to the best result cell array containing details about the different runs and how they were conducted
- exp=the number of the iteration(i in BR{i,x})
- seq=The sequence number you want to see
- features=cell array containing the features used for the generation of this model. Same kind of cell array that the one generated by the function createFeatures
- timeStamps=third output variable of CreateNodsPrediction
- wordlist=cell array of characters containing caption of the features, in the same order they were in the input of function createFeatures

---

plotNodsPrediction
This function plots the result of the function CreateNodsPrediction

INPUT:

- probs=first output variable of CreateNodsPrediction

- ll=second output variable of CreateNodsPrediction
- timeStamps=third output variable of CreateNodsPrediction
- labels=Cell array created with createLabels function
- seq=The sequence number you want to see, this should be the same as the input of CreateNodsPrediction

---

reGenerateNodsScript

This function creates the script that will be used to replay listener's head nods contained in the realNods array

INPUT:

- realNods=Ground Truth Nods array(Seq #,Nod start time, Nod stop time, True Positive Nod)
- seq=The sequence number you want to study

OUTPUT:

A file will be generated in the same path that you are working on, with the following title format: yyyymmdd_hhmmss-mm-dd-yyyy-test-elvin_log.txt