

# **Fusing speech and face recognition on the AIBO ERS-7**

Delft University of Technology

Electrical Engineering, Mathematics and Computer Science  
Man-Machine Interaction



**Project Report, July 4th to September 30th 2005**

**Mickaël THOUZERY**

**Supervised by Leon Rothkrantz and Zhenke Yang**

## Table of contents

<b>Table of contents</b> .....	<b>2</b>
<b>Preface</b> .....	<b>3</b>
<b>Chapter 1 Introduction</b> .....	<b>4</b>
1.1 General Information.....	4
1.2 Project Motivation.....	4
1.3 Project Definition.....	4
1.4 Project Organisation.....	5
1.5 Report Overview.....	5
<b>Chapter 2 Purpose</b> .....	<b>6</b>
2.1 What is the AIBO?.....	6
2.2 Capabilities of the AIBO ERS-7.....	6
2.3 Fusing Audio and Face Recognition.....	7
2.4 ... Within the AIBO.....	8
2.5 Programming the AIBO.....	8
2.6 Reuse a Sound source localization software.....	9
2.7 An illustration of the scenario.....	10
<b>Chapter 3 Conception</b> .....	<b>11</b>
3.1 A common modelling for sound and image recognitions.....	11
3.2 Recognizing a face.....	12
3.2.1 Recognition with Eigenfaces.....	12
3.2.2 Detecting a face around the AIBO.....	13
3.2.3 Centering the eyes.....	14
3.3 Recognizing a sound.....	14
3.3.1 A First Approach.....	14
3.3.2 A further approach: MFCC method.....	15
<b>Chapter 4 Implementation</b> .....	<b>21</b>
4.1 Choice of a framework.....	21
4.2 Client/server architecture.....	21
4.3 An AudioFaceRecognition GUI.....	22
4.4 Merging MATLAB and C/C++ programs.....	23
4.5 Implemented Functions.....	24
4.5.1 Sound.....	24
4.5.2 Face.....	26
4.5.3 AudioFace Fusion.....	29
4.5.4 Interpretation.....	30
4.5.5 GUI Functions.....	32
4.5.6 Parametering recognition.....	33
<b>Chapter 5 Results</b> .....	<b>35</b>
<b>Chapter 6 Conclusion and Recommendations</b> .....	<b>36</b>
6.1 Conclusion.....	36
6.2 Recommendations.....	36
<b>Chapter 7 Glossary</b> .....	<b>37</b>
<b>Chapter 8 Bibliography</b> .....	<b>38</b>
<b>Appendix A</b> .....	<b>39</b>

## Preface

Above all I would like to thank all members of the Man-Machine Interaction Department of TUDelft for their reception and the likable atmosphere they create.

I would like to show my great appreciation to Professor Leon Rothkrantz and thank him for his attention, his confidence and his kindness, as well as Zhenke Yang for his immeasurable availability and help, and for all the knowledge I have learnt from him.

I also want to thank researchers and students who were somehow involved in my project, especially the entire AIBO team for their support, attention and collaboration, and Dragos Datcu for his help and availability in some points of my project.

Special thanks to all the persons who had the kindness to lend their voice and face during AIBO tests sessions.

I do not forget of course all the friends I have met in the city of Delft and everywhere in the Netherlands, for all the great moments we shared together: Maurizio, Hugo, Sarah, Quentin, Marta, Brian, Rui, Joao, Anna, Rui F., Daniel, Anders, Bjoern, Jan, Joern, Raphael, Leslie, all the people of the Ruif and the ones I forget here...

Finally, thanks to the exchange student organisation members from both universities ENSEIRB and TUDelft, namely Paul Gloess, Evelyne Blanchard, Jan de Vries and Toos Brussee-Donders.

### Abstract

Fusing visual and audio recognition on the AIBO pet dog released by Sony. The robot is communicating wireless through a TCP/IP connection to a remote computer which manages face and speech recognition of a person in its environment. Both visual and audio data are sent by the AIBO to the computer which analyses them and tells back the robot how to behave according to the result of the calculation.

### Keywords

Embedded System, Tekkotsu and Urbi Frameworks, Wireless Communication, Motions, Eigenfaces, MFCC.

## Chapter 1 Introduction

### 1.1 General Information

This project accounts for my second-year internship with my French host school ENSEIRB. It was realized at Delft University of Technology, Netherlands, within the framework of the exchange student program Socrates-Erasmus. I started it on July 4<sup>th</sup>, 2005 and finished on October 2<sup>nd</sup>, 2005. The purpose of my project was to implement face and sound recognition within the AIBO entertainment robot released by Sony.

My training period took place at the MMI (Man-Machine Interaction) research group of TUDelft. The aim of this department is to develop research around the subject of human-computer interaction of intelligent multimodal systems.

Professor Leon Rothkrantz was my supervisor for my project. Zhenke Yang, a PhD student who already had an experience of one year with the AIBO, helped me all along my stay.

Moreover, I belonged to the AIBO team which is a research group within the MMI group. It is composed of students and researchers who are involved in projects related to robotics, image and sound processing, more or less close to the AIBO.

Each week, a meeting was organised by the AIBO team to discuss the progress of each project, the prospects and gave place to announcements and questions.

### 1.2 Project Motivation

The choice of carrying out such a project in the MMI Department of TUDelft was stimulated by different aspects.

Above all, it would have put into practise the knowledge acquired during the first two years of ENSEIRB formation. The AIBO robot with all image and sound processing aspects it gathers appeared as an interesting field of work to prepare my third year at ENSEIRB in which I specialize in Numeric Communication Systems Engineering.

More specifically, this project got my interest when I contacted Prof. Rothkrantz a couple of months ago and when he proposed me some topics around the AIBO robot. The idea of merging image and sound aspects came from a suggestion of mine to test an-Eigenfaces- algorithm designed for face identification that had been developped during the previous year at ENSEIRB. But adding sound modalities was even more interesting because it was an opportunity to deal with two aspects and at the end make them interact while fully using the AIBO functionalities.

Working in an international environment was another challenge: practicing English all day long at work and in everyday life, travelling in a foreign country and meet a different culture, or better different cultures since Delft draws people from all continents and especially the Erasmus students (Portugal, Italy, Spain, Sweden, Germany, Romania,...)

### 1.3 Project Definition

The project is defined as follows:

*Design and implement a software for AIBO that uses both audio and image functionalities of the robot in order to detect and recognize persons and their identity. The model developed requires to be opened and reusable for further applications.*

## 1.4 Project Organisation

The project followed several parts:

### Study of the Background

collect and gather documentation, papers concerning the AIBO, to better apprehend its usage, the different existing programming languages and make choices according to the initial goal.

### Define and design a model

A global architecture is built up to support the audio-face recognition with the AIBO. This modelling strongly depends on the decisions take among the existing programming frameworks.

### Implementation

Put into practise the modelling adopted, following an incremental development approach.

### Evaluation of the results

Analyze in which degree the goal of the project was reached.

## 1.5 Report Overview

The present report follows this structure:

### Part 1: Introduction

Provides general information about the project, the motivations, a short definition of it and announces its structure.

### Part 2: Purpose

Presents the AIBO world and introduces the theme of fusing sound and face recognition within this robot.

### Part 3: Conception

Sets the model adopted before project execution.

### Part 4: Implementation

Describes the steps of implementation

### Part 5: Results

Details the results obtained at the final state of the project

### Part 6: Conclusion and Recommendations

Sums up the results and provides recommendations for future projects

## Chapter 2 Purpose

### 2.1 What is the AIBO?

With the growing advancement in the robotics industry, the focus has progressively shifted towards enhancing the autonomy of the robots. Autonomous robotics are now being developed to conduct various duties to make life easier for humans and even to entertain and provide its users with companionship .

The AIBO is one such example of such autonomous robot. It was introduced to the world by Sony in 1999 as a robotic pet dog. This reflected in its name, which is deduced from AI (Artificial Intelligence) and roBOt. It also means pal in Japanese. In the following five years since the original design, AIBO has been improved and its appearance has changed several times.

Initially it was impossible for individual developers to write software for AIBO because the necessary interface with the hardware was kept secret. In 2002 Sony changed this policy and released the AIBO OPEN-R Software Development Kit (SDK). The AIBO became very interesting for research purposes because it provided a complete robot platform on which people could write software and conduct experiments on at a relatively low price. The most recent version is the ERS-7, within improvement on interaction and wireless internet connection. I am working on this particular version of the AIBO.



**Figure 1: AIBO ERS-7**

### 2.2 Capabilities of the AIBO ERS-7

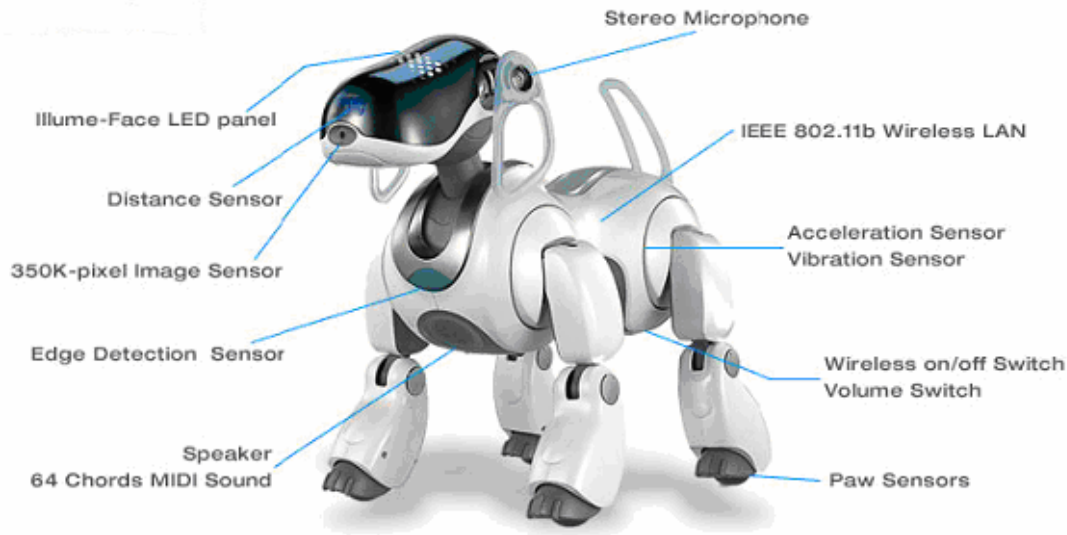
AIBO is Sony at its very best, combining its flagship technologies to conceive a fully autonomous companion to accompany and entertain man in day-to-day life. When battery life runs low, it will swiftly locate the energy station to replenish its battery. AIBO can see, hear, feel for itself and walk. Uniquely skilful, AIBO is able to connect wirelessly with other electronic devices, transmitting photos, sound files and messages. It is also possible to record movie clips.

AIBO's interaction with the outside world is ensured through its sensors: two distance sensor, various touch sensors on head, back, chin, paws, acceleration sensor, a vibration sensor video camera (a 350,000 CMOS image sensor), and stereo microphones in its ears.

In order to show reactions AIBO is equipped with speaker on its chest, LED Lights (on face, ears and back) and a series of movable parts: head, mouth, legs, ears and tail. Apart from this AIBO is also equipped with a wireless card and a blue LED to show its status. [1]

All specifications of the AIBO are provided in Appendix A.

**Figure 2** gives a description of AIBO's anatomy:



**Figure 2: AIBO Sensors and Actuators**

Regarding the topic of the project, the camera and the two microphones are of great interest.

### **Video Camera**

AIBO ERS-7 is equipped with a colour vision camera mounted above its mouth

### **Stereo Microphones**

AIBO ERS-7 is equipped with stereo microphones located either side of its head

## **2.3 Fusing Audio and Face Recognition...**

As mentioned earlier, the idea of fusing audio and face recognition appeared when I proposed to Mr Rothkrantz to port an algorithm for face recognition implemented by myself and two of my colleagues at ENSEIRB. This algorithm –named Eigenfaces, had been tested among frontal face images and had provided relevant results. Thanks to its simplicity yet good accuracy, I was a valuable and promising face recognition method. Provided with a database of fifteen persons declined on several standard facial expressions, the software could determine with accuracy if a tested image could represent a face or a non-face, a person belonging to the database-consequently giving his/her name- or foreign to it.

This truly matches what a companion pet dog must require to behave properly, that is to say reacting when someone is coming towards it, and make a selection between its masters and the unknown people.

However, Prof Rothkrantz introduced me the possibility of adding extra modalities of the AIBO in order to improve people's recognition. Indeed taking into consideration the soundscape is very

helpful in some situations when a camera is ineffective and increase the probability of a successful result. For instance in bad light conditions, the AIBO can always rely on its ears, and vice versa, if it is situated in a noisy environment then it will trust its eyes.

## 2.4 ...Within the AIBO

Working with an AIBO robot meant I had the chance to work with an state of art piece of technology, and gave the opportunity to use the software on/with an embedded system, instead of the conventional theoretical setups used in pattern recognition. This also accounted for a great challenge since it would consist in adapting the existing and future software than merely porting it on the AIBO.

The Eigenfaces is reasonably not a good method if some fundamental assumptions are not satisfied, basically the resizing of each tested picture by moving the eyes on a pre-defined position. And as a face can be localized at any point of the camera frame, the Eigenfaces program was falling through. This constraint had to be kept in mind during the conception of the future software.

## 2.5 Programming the AIBO

Three frameworks are available to program the behaviour of the AIBO:

- § OPEN-R SDK,
- § Tekkotsu
- § URBI

### OPEN-R SDK

AIBO owners can teach their pet new behaviours by reprogramming them in Sony's special *RCode* language or *Open-R SDK* for non-commercial use. AIBO Software Development Environment can create software that either executes on AIBO or executes on a PC or controls AIBO by using a wireless LAN. This SDE contains three SDKs (Software Development Kits) and a motion editor. The three SDKs are Open-R SDK, R-Code SDK, and AIBO Remote Framework. These development environments are provided free.

The Open-R SDK is a cross development environment based on gcc (C++) with which developers can make software that works on AIBO (ERS-7, ERS-210, ERS 220, ERS-210A, and ERS-220A). The R-Code SDK is an environment with which developers can execute programs written in R-Code, a scripting language, on AIBO (ERS-7). The AIBO Remote Framework is a Windows application development environment based on Visual C++, with which developers can make software that runs on Windows. The software can control AIBO (ERS-7) remotely via wireless LAN. AIBO Motion Editor can be used with the Open-R SDK, the R-Code SDK and the AIBO Remote Framework.

Open-R SDK provides a disclosed Open-R API of system layer ("level-2" interface). In this case, developers only can utilize some AIBO's functions such as:

- moving AIBO's joints
- get information from sensors
- get image from camera
- use wireless LAN (TCP/IP)

The Open-R SDK includes tools to make Open-R objects, sample programs, and memory



stick images that must be copied to a AIBO programmable memory stick  
The developers also can make motions by synchronizing with sounds (MIDI, WAV) and LED patterns in AIBO Motion Editor. [6] [16]

### **Tekkotsu**

*Tekkotsu* is a framework built on top of OPEN-R SDK[5]. This means, that in order to use Tekkotsu, the OPEN-R SDK also has to be installed. Tekkotsu offers a way to interface with WLAN. Joints, head movement camera etc. can be controlled via wireless LAN. The programming model with URBI also holds for Tekkotsu. (Tekkotsu server is also an object running on OPEN-R).

The advantage of Tekkotsu is that it offers higher level commands (instead of moving individual joints, one can issue commands like "walk"). Furthermore the Tekkotsu framework aids people who develop objects intended to work on the AIBO (with Tekkotsu) by adding a level of abstraction. So instead of having the to know the message passing details of ones object with other objects (like in the URBI) a Tekkotsu programmer can think in terms of behaviours (a term that is central in Tekkotsu programming framework tutorials). [3] [8]

### **URBI**

*Universal Robot Body Interface (URBI)* is a scripted command language used to control robots (AIBO, pioneer)[5]. It is a robot-independent API based on client/server architecture. In the OPEN-R programming model the URBI server can be viewed as another object. The developer can make use of the URBI server in two ways: via a computer through the wireless LAN using the *liburbi c++* (external client) or through direct inter-process communication using *liburbi OPEN-R* (onboard client).

In the case of an external client, the communication then takes place through a TCP/IP or IPC connection. When using URBI over TCP/IP messages are sending to the URBI object via telnet over port 54000. The URBI Object then sends the appropriate messages to the other objects on the OPEN-R system to accomplish the given command.

In the case of an onboard client, the client is also an OPEN-R object (containing *doinit()*, *dostart()*, etc.) that runs along in the OPEN-System and sends messages to the URBI server. Thus, URBI functionality can then be utilized by passing/receiving messages to/from the URBI object (which of course should also be running). [6] [7] [8]

## **2.6 Reuse a Sound source localization software...**

A possible prospect to go further in this project was to combine audio-face recognition with the localization of a sound in the bi-dimensional environment.

Indeed, a sound localization software had been developed at the MMI Department of TUDelft and successfully demonstrated at Robocup last year. It offered to the AIBO the ability to turn its head towards the source of a sound generated in a 2D horizontal plan, and this process was executed run-time. A sound recognition had also been implemented and the robot could react to specific frequencies generated by other AIBO mates.

Including such a program was interesting because it could make the AIBO react closer to a real companionship dog: instead of tracking a face and then analyzing its features, independently from

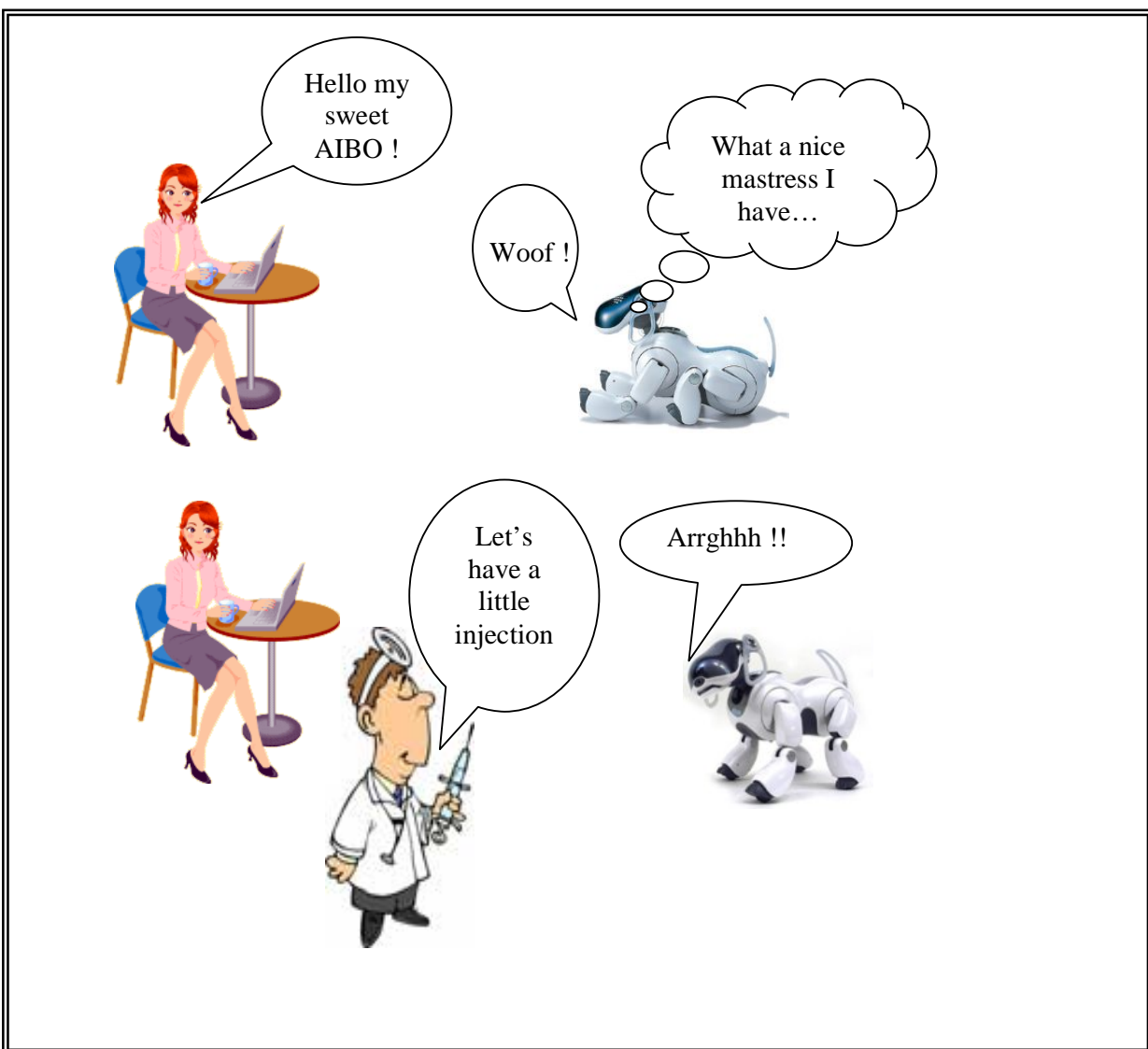
any stimulation-as assumed for audio-face recognition: it would start to look around only if something happened, basically if someone spoke.

Nevertheless, some limitations were somewhat expected: speech-and-face recognition was designed for human beings, so identification of sound had to be realized over the entire 3D environment and not only the 2D AIBO plan.

Note that this part is likely not to be treated during the internship, its realization depends on the amount of time available.

## 2.7 An illustration of the scenario

With all the aspects discussed above, the ideal scenario was as drawn in **Figure 3**:



**Figure 3** Illustration of the scenario

## Chapter 3 Conception

Taking into consideration the purpose of the research set up in the previous part, this new chapter will present and detail the technical choices and solutions made to fulfil the expectations of the project.

### 3.1 A common modelling for sound and image recognitions

Both sound and image recognitions can be classified into two modules: *feature extraction* and *feature matching*.

Feature extraction is the process that extracts a small amount of data from the face image and the voice signal that can later be used to represent each subject. A training phase is then operated on each registered subject from their relative sound and face information so that the program can build or train a reference model for each subject.

Feature matching implies the task to identify the unknown subject by comparing extracted features from his/her face and voice inputs with the ones from a set of known subjects. This module aims at indicating if the tested subject is a member of the database and if so which one he/she is. The input data is matched with stored reference model(s) and recognition decision is made, according to a certain threshold.

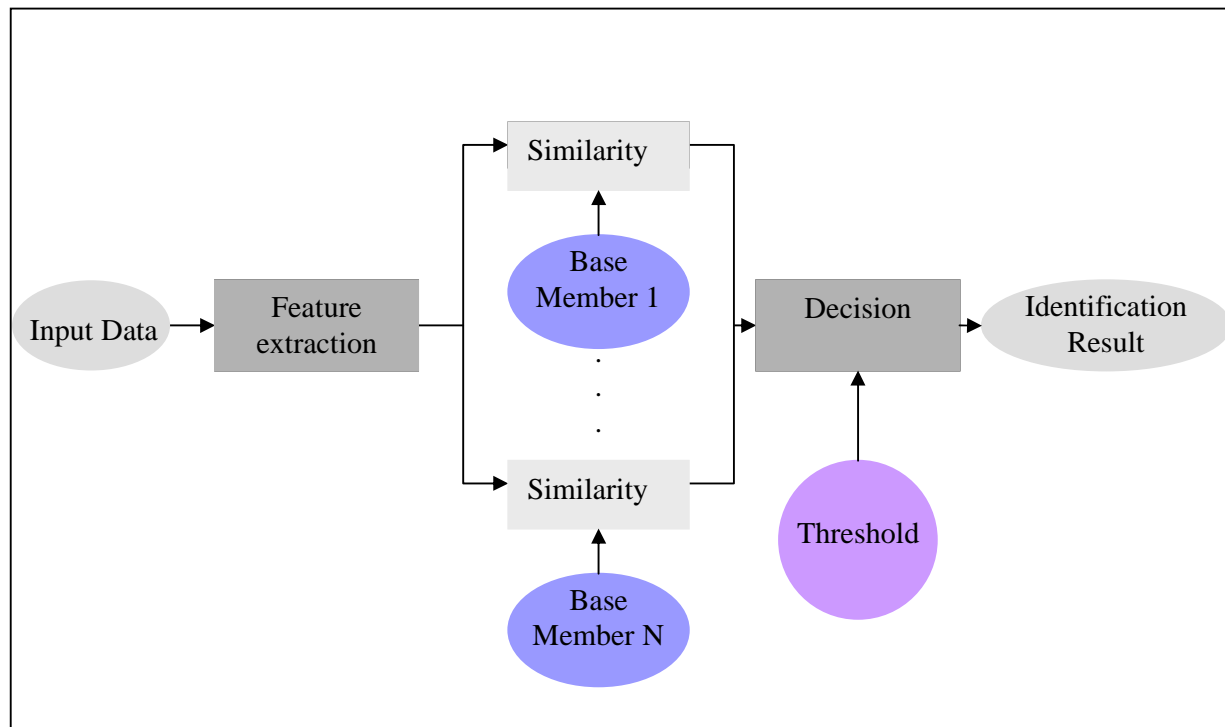


Figure 4 Model for recognition

## 3.2 Recognizing a face

### 3.2.1 Recognition with Eigenfaces

This face recognition approach was proposed by Turk and Pentland in 1991. In this technique each image is modeled as a vector whose dimension is the number of pixels of the image. Some characteristic features are nonetheless extracted thanks to a dimensional reduction method -called PCA (Principal Components Analysis).

Let us assume  $I_k$  ( $i=1..pixel\_number$ ) accounting for the image number  $k$ 's vector, therefore each face can be viewed as a group of points in a  $\mathbf{R}^{pixel\_number}$ -dimensioned space. Moreover, let us consider  $I_{ave}$  accounting for the average face of all the  $I_k$  faces.

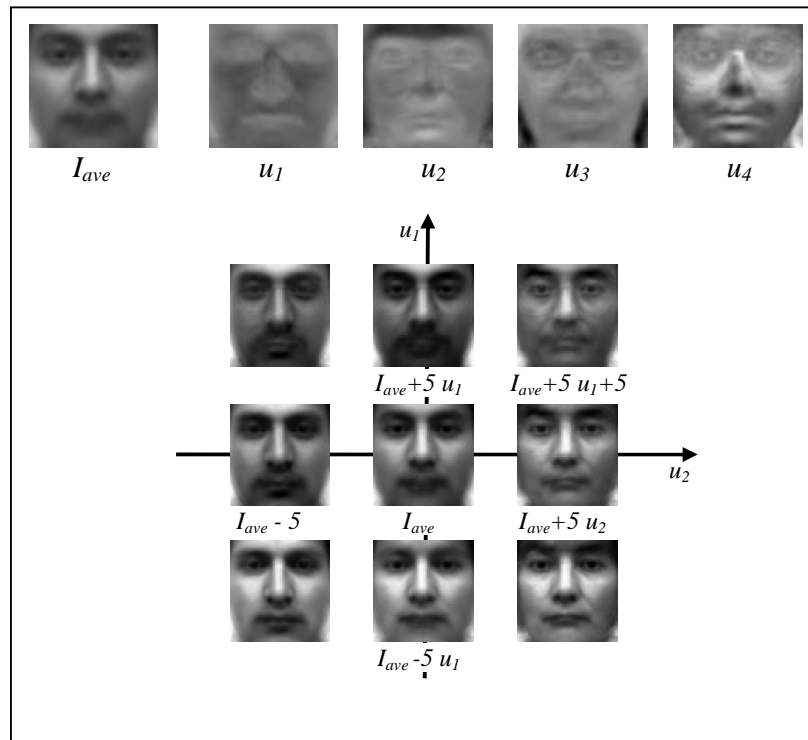
The principle of the Eigenfaces method is to model the difference between any face and the average face thanks to a limited number of images  $u_h$ , named Eigenfaces.

As a result, each image  $J$  belonging to  $\mathbf{R}^{pixel\_number}$  is expressed as the average face plus a linear combination of Eigenfaces, as follows:

$$J = I_{ave} + \sum_h c(h)u_h + \varepsilon$$

Where  $c(h)$  is the  $h^{th}$  eigenface coefficient of  $J$  and  $\varepsilon$  is the error between  $J$  and its approximation with Eigenfaces.

The eigenface method is based on the fact that the number of used Eigenfaces is much lower than the space dimension.



**Figure 5** Composition of a face with Eigenfaces

The series of Eigenfaces  $u_k$  is obtained through the PCA.

Identification –or feature matching- is then realized by comparing the distance of the projection of the tested image with the ones of the other existing images. We do understand projection by projection in the “Eigenfaces under-space”.

For more details concerning PCA and how identification is managed, please refer to [11].

Face recognition with Eigenfaces as it has been implemented imposes some serious limits when adapting it with a moving camera. The presence of a face in the frame of the AIBO’s vision does not guarantee the picture taken would represent exactly a front centred face as it is assumed in the Eigenfaces preprocessing phase. A routine to centre manually the eyes of the face on a constant position was programmed but in the case of an automatic recognition it becomes naturally inefficient.

That is why some preliminary treatments must be made before applying the algorithm. More exactly, the robot must be capable of detecting and extracting a face in the field of its camera instead of merely taking a photo. Actually an eye detection would be theoretically enough but because of the weak resolution of the camera this is more cautious to start with a face extraction, with the view to applying further an eye detection within the face obtained. In addition, if face detection is efficient, eyes’ position can hopefully be deducted according to the face frame.

### 3.2.2 Detecting a face around the AIBO

Some research around the topic of face detection had been already carried out inside of the MMI Department, where it had been made use of Intel’s Open CV library. The detection in question makes it possible to detect and track real time faces appearing in the frame of a video camera. The camera was of 640x480 resolution –instead of 208x160 for the AIBO, but faces could be detected from at least three meters, which is obviously not expected with the AIBO.

#### Intel’s Open CV library

Open CV library is released online by Intel and is intended for use, incorporation and modification by researchers, commercial software developers, government and camera vendors as reflected in the license.

It is an aid commercial uses of computer vision in human-computer interface, robotics, monitoring, biometrics and security by providing a free and open infrastructure where the distributed efforts of the vision community can be consolidated and performance optimized.

An online community organized around the Open Source Computer Vision Library has been created for sharing information, bugs, opinions, questions and answers about the library. [17]

OpenCV library is useful for many applications in image processing such as video surveillance, image retrieval and so on. With this library detection can be lead for any kinds of objects, not only faces but also hands, cars, tables, etc. But the algorithm ensuring detecton is made regardless the nature of the object. A statistical approach is chosen, using Haar-like features and a cascade of boosted tree classifiers. For each model a statistical model (classifier) is trained:

the training phase takes into account multiple instances of the object of interest-positive samples, as well as negative samples, which means samples which do not contain the object. The classifier is trained on images of fixed size, and detection is accomplished by sliding a search window of the predefined size within the image to analyze and checked whether an image region looks like the object of interest or not. To know more see please refer to [18].

A demonstration of face detection is shown on **Figure 6**:



**Figure 6 Face detection**

### 3.2.3 Centering the eyes

Since the statistical model-based method used by OpenCV works independently from the nature of the object, there is no reason that it can not satisfy an eye detection, by submitting an entire set of eye samples. However no eye detector is available in the released library, that is why the training procedure has to be done by our own. A large database of positive and samples must be picked up, because the performances of the training deeply depend on the robustness of the database -some thousands of pictures have to be gathered to guarantee a good result.

## 3.3 Recognizing a sound

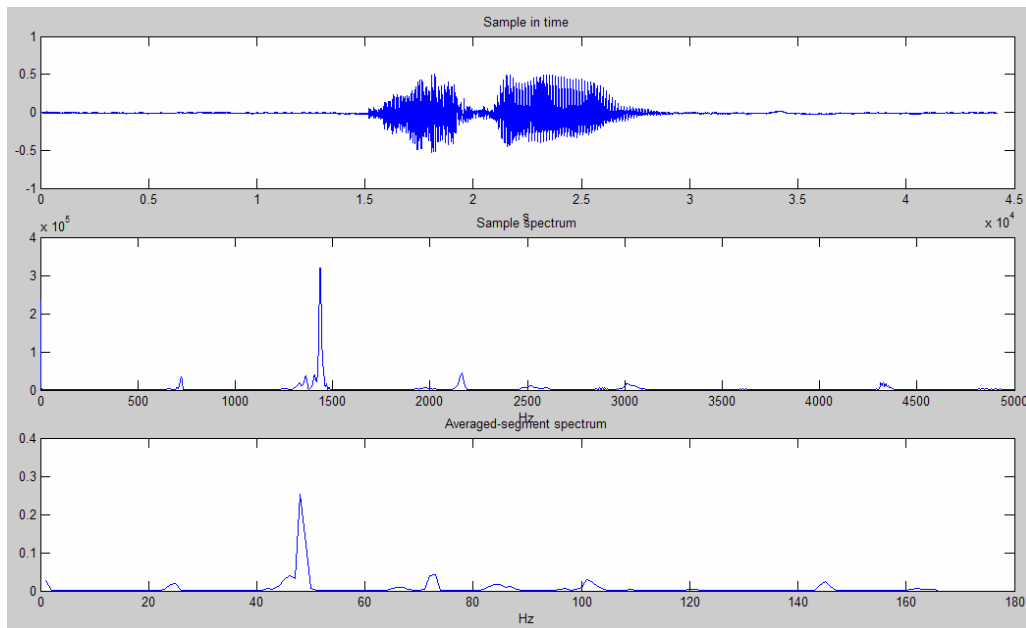
Speaker recognition methods can be divided into text-independent and text-dependent methods. In a text-independent system, speaker models capture characteristics of somebody's speech which show up irrespective of what one is saying. In a text-dependent system, on the other hand, the recognition of the speaker's identity is based on his or her speaking one or more specific phrases or keywords, like "AIBO" or "Come here!".

Without making a choice straight away on which method to adopt, we can start with simple methods and if necessary or worth shift into more complex ones.

### 3.3.1 A First Approach

A first idea is to extract the spectral characteristics of each person by calculating the average of the frequency amplitudes between regularly-spaced positions of the spectrum scale. That means a Fast Fourier Transformation is performed on the original recorded file, imaginary values converted in real ones so as to have an estimation of the average, for each segment of the spectrum. Therefore each subject's feature characteristics are described as a vector of amplitudes values for all the regions of

the spectrum. **Figure 7** plots the original sample in time, the corresponding spectrum (with fft) and the averaged-segment spectrum.



**Figure 7 Getting frequency features from a voice sample**

During the test phase a matrix multiplication is operated between two subject vectors to estimate the similarities. For instance if two voices are close one to each other, it is expected that the spectral energy will be located in the same area, which will increase the final value. Hence a distance between two subjects is obtained and usable to take decisions.

Due to the irrelevant performances of this first method (see part 4.5.1), it revealed necessary to opt for another one. Therefore the next part focuses on one of the most popular method in speech recognition, called MFCC.

### 3.3.2 A further approach: MFCC method

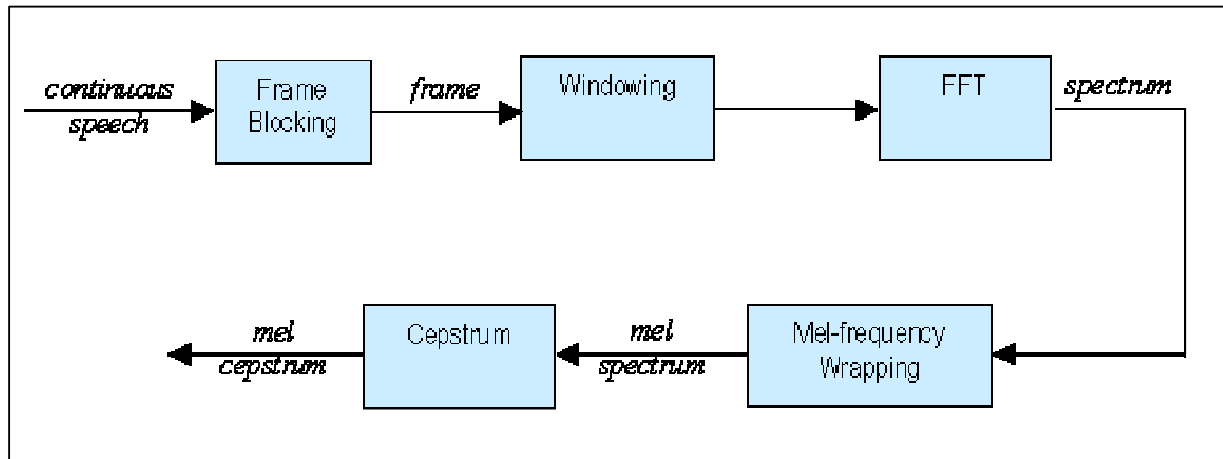
The MFCC method accounts for Mel-Frequency Cepstrum Coefficients.

#### Principle of MFCC

Quote from [12]:

*Because of the known variation of the ear's critical band-widths with frequency, filters spaced linearly at low frequencies and logarithmically at high frequencies have been used to capture the phonetically important characteristics of speech. [...] This result suggested that a compact representation would be provided by a set of mel-frequency cepstrum coefficients. These cepstrum coefficients are the result of a cosine transform of the real logarithm of the short-time energy spectrum expressed on a Mel-frequency scale. In MFCC, the main advantage is that it uses melfrequency scaling which is very approximate to the human auditory system. The basic steps are: Preprocessing and FFT, Mel-frequency scaling, Cepstrum.*

This technique has also the great advantage to be a text-independent speaker identification system in the sense that it does not care of the content of the sentence. **Figure 8** is a description of the different steps to exhibit features characteristics of an audio sample with MFCC [13].



**Figure 8** Block diagram of the MFCC processor

## Preprocessing and FFT

### Frame Blocking

In this step the continuous speech signal is blocked into frames of  $N$  samples, with adjacent frames being separated by  $M$  ( $M < N$ ). Typical values for  $N$  and  $M$  are  $N = 256$  (which is equivalent to  $\sim 30$  msec windowing and facilitate the fast radix-2 FFT) and  $M = 100$ .

### Windowing

The next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. Typically the *Hamming* window is used.

### Fast Fourier Transform (FFT)

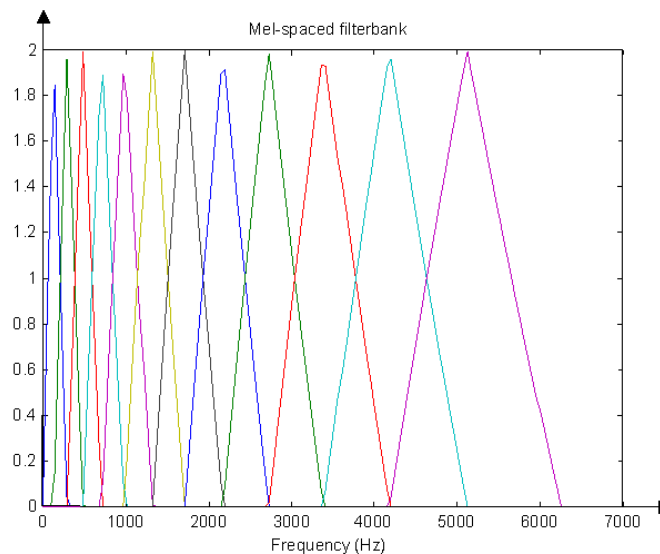
The next processing step is the Fast Fourier Transform, which converts each frame of  $N$  samples from the time domain into the frequency domain



## Mel-frequency Scaling

As mentioned above, psychophysical studies have shown that human perception of the frequency contents of sounds for speech signals does not follow a linear scale. Thus for each tone with an actual frequency,  $f$ , measured in Hz, a subjective pitch is measured on a scale called the 'mel' scale. The *mel-frequency* scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. As a reference point, the pitch of a 1 kHz tone, 40 dB above the perceptual hearing threshold, is defined as 1000 mels

One approach to simulating the subjective spectrum is to use a filter bank, spaced uniformly on the mel scale (see **Figure 9**). That filter bank has a triangular band pass frequency response, and the spacing as well as the bandwidth is determined by a constant mel frequency interval. The modified spectrum of  $S(\omega)$  thus consists of the output power of these filters when  $S(\omega)$  is the input. The number of mel spectrum coefficients,  $K$ , is typically chosen as 20.



**Figure 9** Example of a mel-spaced filterbank

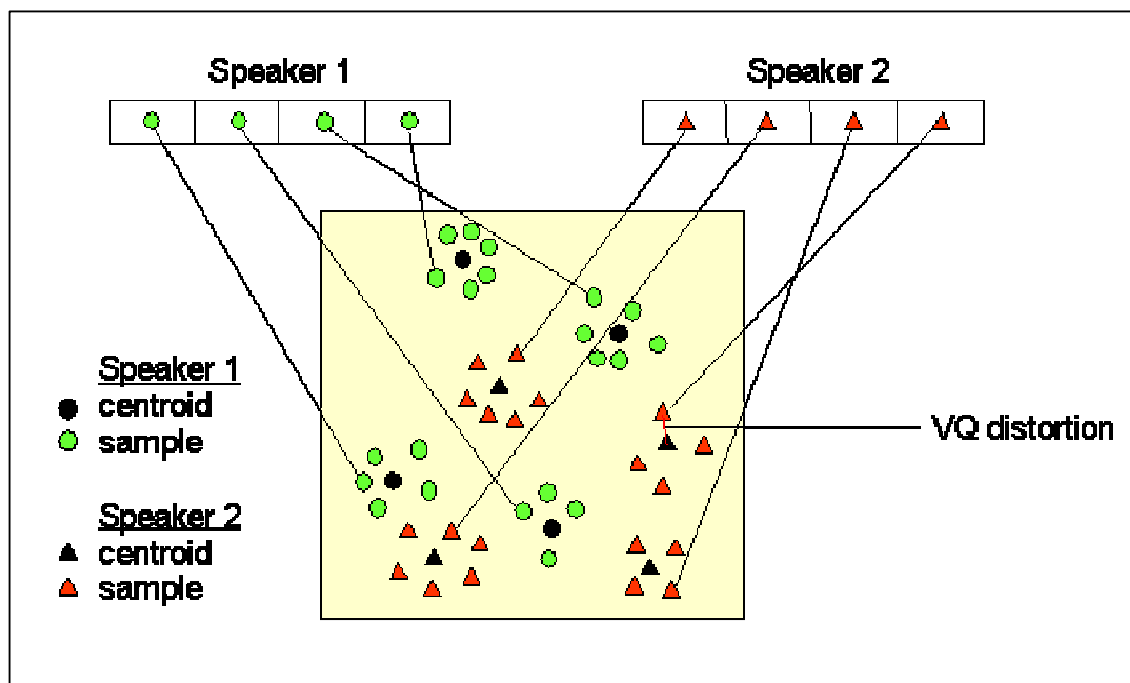
## Cepstrum

In this final step, the log mel spectrum is converted back to time. The result is called the mel frequency cepstrum coefficients (MFCC). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the mel spectrum coefficients (and so their logarithm) are real numbers, we can convert them to the time domain using the Discrete Cosine Transform (DCT).

## Feature matching

The state-of-the-art in feature matching techniques used in speaker recognition include Dynamic Time Warping (DTW), Hidden Markov Modelling (HMM), and Vector Quantization (VQ). In this project, the VQ approach will be used, due to ease of implementation and high accuracy. VQ is a process of mapping vectors from a large vector space to a finite number of regions in that space. Each region is called a *cluster* and can be represented by its centre called a *codeword*. The collection of all code words is called a *codebook*. [13]

**Figure 10** shows a conceptual diagram to illustrate this recognition process. In the figure, only two speakers and two dimensions of the acoustic space are shown. The circles refer to the acoustic vectors from the speaker 1 while the triangles are from the speaker 2. In the training phase, a speaker-specific VQ codebook is generated for each known speaker by clustering his/her training acoustic vectors. The result code words (centroids) are shown in **Figure 10** by black circles and black triangles for speaker 1 and 2, respectively. The distance from a vector to the closest codeword of a codebook is called a VQ-distortion. In the recognition phase, an input utterance of an unknown voice is "vector-quantized" using each trained codebook and the *total VQ distortion* is computed. The speaker corresponding to the VQ codebook with smallest total distortion is identified.



**Figure 10** Conceptual diagram illustrating vector quantization codebook formation

After the enrolment session, the acoustic vectors extracted from input speech of a speaker provide a set of training vectors. As described above, the next important step is to build a speaker-specific VQ codebook for this speaker using those training vectors. There is a well-known algorithm, namely LBG algorithm [Linde, Buzo and Gray, 1980], for clustering a set of  $L$  training vectors into a set of  $M$  codebook vectors. More information about LBG are provided in [13].

### 3.3 Taking decisions between both

Two models have been eventually set up to ensure recognition in both ways. Next step is to merge them to provide a final and unique answer, the one from the AIBO.

As evoked, for each kind of perception different cases of situations can occur. Of course what can happen depends also of what it is expected in the project and what can allow the techniques described above. A set of all possible events are presented here:

Speech:

- No voice heard
- Unknown voice
- Known voice but not the voice of the master
- Voice of the master







Face:

- No face detected
- A face detected but seen as a non-face by the recognition program with Eigenfaces
- Known face but not the face of the master
- Face of the master

Note that a distinction is made between different members of the database: some are considered as masters whereas the other ones are simply persons known and familiar to the pet dog but not close ones. This way of distinguishing people can be utilized even more by adopting a specific attitude for every member of the database. Notwithstanding we will stay with this first configuration.

In addition, an appreciation is made redundantly between the face detection and the face recognition methods. However the Eigenfaces algorithm will be disabled in case of a non-detection of face, and if a face is claimed to be detected and that recognition software outputs non-face as an answer, it will be decided the same.

The challenge is actually to deal with some conflicting situations and give priorities to only one output. **Figure 11** draws of all the predictable events.

Image Sound	No Face	Non-Face	Unknown Face	Known Face	Master Face
Silence	Do not react	Do not react	Bark naughtily	Happy Bark	Bark happily Moves its tail
Unknown Voice	Bark naughtily	Bark naughtily	Bark naughtily		
Known Voice	Happy Bark	Happy Bark		Bark happily Moves its tail	
Master Voice	Bark happily Moves its tail	Bark happily Moves its tail			Bark happily Moves its tail

**Figure 11** How to the AIBO can react according to what it hears and sees

In case of contradictory information, the dog can express its doubts and remain on its uncertainties. But for audio and image identification some variables measures the distance between each subject and therefore provide an estimation for a successful identification or not. For instance if the distance of a tested voice's subject is very high with the based-voice subjects, whereas the face of the same subject is very close to a person of the database, then a priority is given to face recognition's output.

## Chapter 4 Implementation

### 4.1 Choice of a framework

Choosing an OPEN-R SDK development environment could be motivated by the desire of programming some code that works on the AIBO, providing full autonomy and independency from an external computer. But this does not really fit with the requirements of this project. Due to the original software for audio/face recognition implemented with MATLAB, an auxiliary computer is required to control the AIBO. The Tekkotsu and Urbi frameworks are more adapted for this purpose and are made to facilitate remote commands on the AIBO. That is why we will focus on these two ones only.

An AIBO interface developed by researchers of the MMI Department is available to command the AIBO wirelessly. It is capable of ensuring tasks as sending commands to the AIBO and reading sensor data from it.

The following abstractions made were:

- Sensor groups (GetStatus)
- Motion control (DoAction)
- Services (audio/video) (Send/Receive)

[14]

This interface is built on top of URBI and Tekkotsu frameworks. In addition the interface hides from the user if URBI or Tekkotsu, or any other object, is used as mediator to OPEN-R objects i.e. `OvirtualRobotComm` and/or `OvirtualRobotAudioComm`.

### 4.2 Client/server architecture

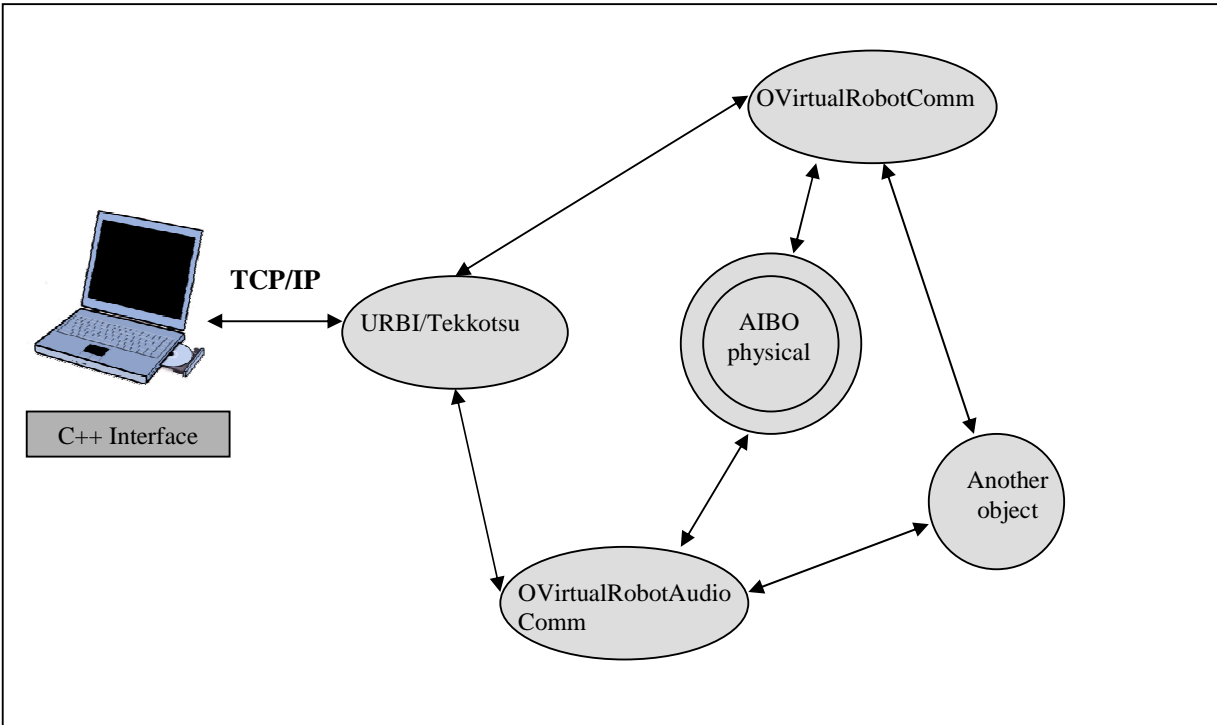
A client/server architecture involves client processes requesting service from server processes. URBI and Tekkotsu are typically client/server architecture based frameworks. It will be made use of two ranges of inter-process communication: between the remote computer client and the URBI/Tekkotsu servers onboard, and between this same server and the OPEN-R objects.

A client/server architecture involves client processes requesting service from server processes. URBI and Tekkotsu are typically client/server architecture based frameworks. It will be made use of two ranges of inter-process communication: between the remote computer client and the URBI/Tekkotsu servers onboard, and between this same server and the OPEN-R objects.

In the case of an onboard client, the client is also an OPEN-R object (containing `doinit()`, `dostart()`, etc.) that runs along in the OPEN-System and sends messages to the URBI server.

Messages are sent via a TCP/IP connection in case of an external client (computer). Communication can be made via telnet over defined ports (59000 for Tekkotsu server, 54000 for URBI server) or other ports created by the user.

However an automatic process is required in the project. Libraries available in the Urbi and Tekkotsu package are very useful because they make it possible to wrap the TCP/IP sending job in functions written in C/C++ and manage the communication process automatically.



**Figure 12 Place of the interface**

### 4.3 An AudioFaceRecognition GUI

One practical question is to know how the user will manage the AIBO remotely since it can not start to behave without some initiatives from the user. The robot must receive a message in order to switch into one specific mode, that is to say testing, learning, managing only face or speech recognition, or both of them.

A GUI previously built and used by Zhenke Yang for some experiences on the AIBO was reused and adapted to the requirements of the projects. It is more exactly a C programmed interface in which each button refers to a given C program. It is moreover possible to include some URBI or Tekkotsu routines by providing the right headers corresponding to the right libraries. Its adaptation was all the more immediate that its design is flexible and opened.

Thanks to the GUI, routines are processed not only automatically but also depending on the user's intentions.



**Figure 13 Design of the GUI**

In addition of the existing button “camera” which displays the run-time camera stream-with nonetheless a low rate level, four other buttons have been created for this project: Learn Someone, Sound Rec, Face Rec and AudioFace Rec. To have a complete overview of what they account for see paragraph 4.5.5.

#### **4.4 Merging MATLAB and C/C++ programs**

As known, the subject of the current project takes its rise with some work carried out with face identification and Eigenfaces method. The programs had been designed in MATLAB which is not proper to get along with the AIBO’s classic programming style. The programmer had to keep in mind that some conversions should have been made somehow. An alternative solution was to rewrite those scripts in C/C++. But first due to the short time “attributed” to this project, it was safer to keep MATLAB sources. In addition, including MATLAB capabilities could open a new way of working for further research projects with the AIBO.

Merging MATLAB and C/C++ is accomplished by compiling automatically MATLAB files into C files via the MATLAB compiler. It generates entire stand-alone applications and elements usable in a C environment, with no longer needing MATLAB software. Note that compiling a M-file calling other M-files will generate also these depending files. Note finally that for strange reasons C files compiled with mcc can not be assigned of arguments, as if it works with pure MATLAB functions. This statement concerns MATLAB functions called by any other m file. Even if there must be a solution to that, the problem was by-passed by creating a text file which welcomes the input values of future compiled- MATLAB files.

## 4.5 Implemented Functions

The implementation approach adopted during the project was incremental, that is to say a first working model was expected in a short deadline and some other functionalities were added on top so that the initial model was steadily improved. This assumes to swing between implementing and testing what was programmed, regularly.

Concretely, the primordial aspect to work on was linking all of the involved elements for the final process, namely Open-R, Tekkotsu, Urbi, MATLAB, ensuring a good communicational background and passing data fast and efficiently. In a second time it would consist in focusing on the sound and image processing themselves.

Notwithstanding this part will not follow the chronological evolution of the incremental implementation made but will rather take the way reverse. Indeed in the “latter” case things are defined with a certain intention but their content are implemented later. Typically the four functions of the GUI are set with the aim at executing some code, but this one is implemented further. Hence it is easier to describe implementation in the other way around.

All sound and face source codes have been written in MATLAB language and first run in a MATLAB environment. When having been sure they work –not in terms of performances but of absence of errors, they are compiled via `mcc` into C files. Only those which depend on any other M-file are compiled manually (see the previous paragraph).

### 4.5.1 Sound

Referring to the conception part, two different techniques had been thought to apply sound recognition with the AIBO.

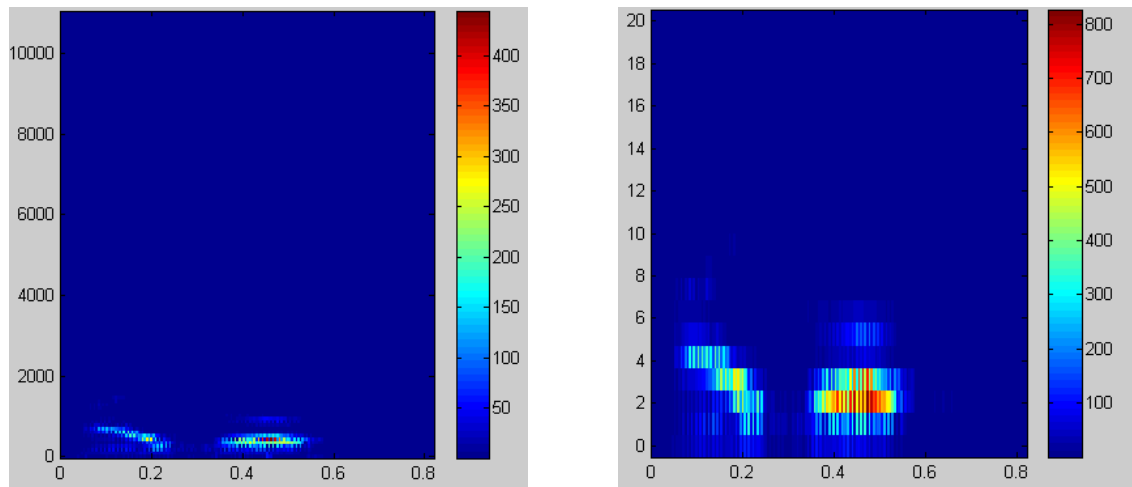
The first one is rather empiric but deserves to give a first idea of speech recognition with the AIBO. Some voice samples were recorded among a group of five persons. Nevertheless, several attempts were made on a same person and revealed a strong variation in the position of the predominant-frequency region, even in similar environment situations, using quite the same intonation of voice and pronouncing the word “AIBO”.

Some improvement occurred when speech time was stretched, for instance by pronouncing the utterance “Hello AIBO, come here”. This is understandable in so far “that/as” a long expression provides more information on someone’s voice than a simple word.

But the method reveals clearly text-dependent: performances fall through for different sentences said by all the tested people’s voices, or even distinctive intonations.

The second method called Mel-Frequency Cepstrum Coefficients (MFCC) is a more elaborated and conceptual method. The huge difference with the previous one is that it is based on the known variation of the human ear’s critical bandwidths with frequency, thus perception is more resistant to parasites and variations. To illustrate MFCC contribution over speech analysis a comparison between a non-treated sample and a mel-frequency sample is shown on **Figure 14**:





**Figure 14 Impact of the MFCC technique on a spectrum**

On **Figure 14** the first frame represents the sample without any treatment by MFCC, the second one is the MFCC transformed one.

Here is detailed the implementation phases for sound recognition with MFCC. These ones closely follow the steps defined in Chapter 3.

### Preprocessing and FFT

First of all a matrix  $M$  is created to collect all the frames blocks extracted from the original sample. A matrix of dimension  $256 \times 129$  is eventually obtained. Each column contains the values inside of a specific block. Secondly, a Hamming window is applied on each block, and finally a FFT. The FFT output values are taken and converted into absolute values to be displayed. A logarithmic scale is used to display the power spectrum; such a scale is useful to spread the spectrum and better visualize it.

### Mel-Frequency Scaling

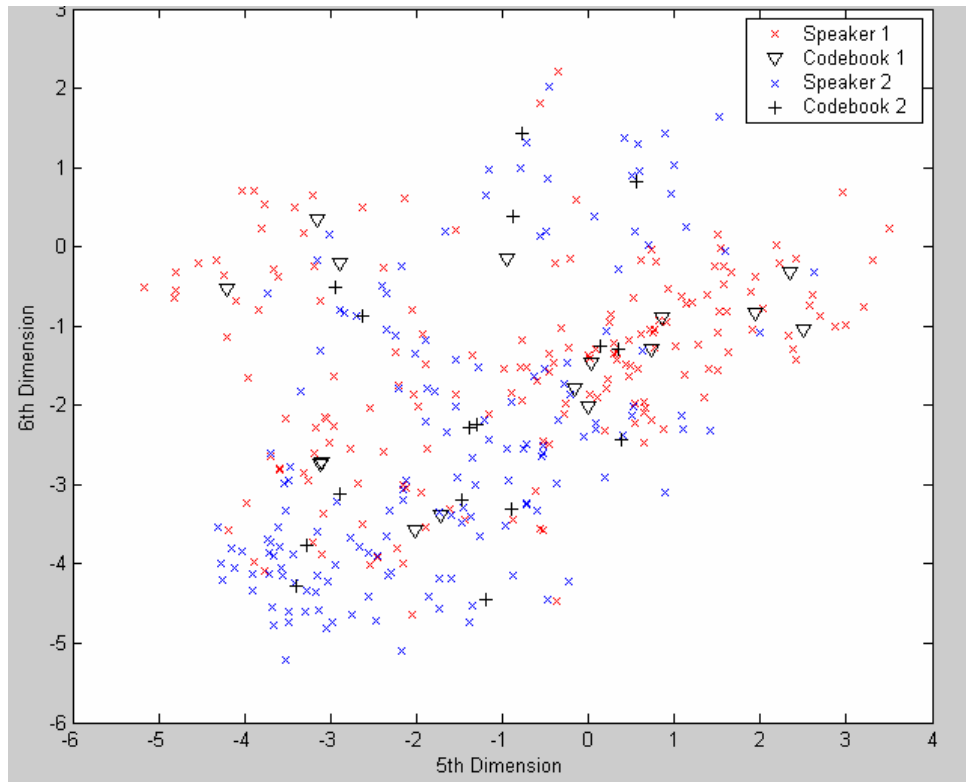
Afterwards, a filter bank is applied on the spectrum according so as to transform in into a close representation of the human ear perception. This phase was realized by reusing a function designed and released from [13], whose name is *mel\_fb*. At the end a 20-dimensioned acoustic vector is output.

Note that preprocessing, FFT and Mel-Frequency Scaling are performed by *MFCC* function.

### Clustering the training vectors

*vq\_lbg* is in charge of training a VQ codebook using the LGB algorithm mentioned in Chapter 3. *disteu* was also distributed by [13] and aims at computing the pairwise Euclidean distances between the codewords and training vectors.

To have an idea of the acoustic space how it is modeled in this way, here are two vectors from two different persons displayed for two dimensions (for instance the 5<sup>th</sup> and the 6<sup>th</sup>). **Figure 15** also displays the VQ codebooks:



**Figure 15 Representation of two dimensions**

In spite of a large common region shared by both sample points, some parts exclusively belong to one of them and will hopefully contribute to a voice features differentiation.

Test phase

The program was tested over seven voices. At the beginning it had been care of recording short samples of a same utterance (“AIBO” to change...). However half of the test was unsuccessful. But longer sounds make it significantly more efficient, in addition whatever the utterance is.

An important is nonetheless to speak continuously to give as much information as possible.

A last of four seconds was decided because of a good compromise between quantity of data and speed calculation.

In this way, the rate of successful recognition can be estimated at about 80%.

#### 4.5.2 Face

In this part we will not focus on the Eigenfaces program’s implementation since it had already been accomplished in a previous project [11]. On the contrary, what seems necessary to highlight is how the limitations of the developed Eigenfaces software are solved, namely face detection and readjustment of eyes position.

## Face detection

It was made use of OpenCV's Library. To work with it, two options are possible:

-Track and extract the face directly from the video stream of the AIBO's camera. This option implies however two restrictions: first the structure of video provided by the AIBO is not the same as the one handled by the OpenCV library hence a conversion is required. Furthermore, the video stream received on the computer via the wireless connection is of very low rate, which does not guarantee a fast detection.

-Send a series of pictures from the AIBO's camera to the PC and apply the detection algorithm. This option implies the challenge to cover the AIBO visual environment by a limited (and few) number of images, moreover the detection is tried on each picture, at least until a face is found. Somehow slackening can also occur. But a decision has to be made and since the second option seems more immediate it is will chosen.

A number of pictures has to be defined with the purpose of minimizing the speed calculation (too many ones will irrevocably slow down the detection execution) and covering the vision field of the dog.

But perhaps should it be good to define the borders of the vision field. Indeed a dog will naturally look towards him, in the front semi-circle. Furthermore a Sound localization already programmed on the AIBO is reusable: it allows an horizontal research in the plan of the AIBO's head. So basically a vertical scanning should be sufficient in this project: The dog turns its head horizontally towards a speaker and then moves its head up and down to find the corresponding face.

After some tests on the AIBO it was deemed that three pictures were enough to detect someone.

Let us remind that detection consists until now in aiming at the face by drawing a rectangle around it. But only what is inside of the rectangle is of interest for the study.

A routine was consequently written to extract the content of it: the routine itself is written under MATLAB. The program takes in arguments two values corresponding to two opposite corner points of the rectangle, output by *facedetec.c* and let in the text file coord.txt. Then it extracts the face.

Note that face detection algorithm is applied on jpg-format pictures, therefore pictures from the AIBO-which are in RGB - need to be transformed in this format. The situation is the same when the face is detected, the extracted rectangle requires a jpg-conversion for future treatments. That is what the function *convertjpg* is made for.

```
height1=pt2y-pt1y;
width1=pt2x-pt1x;
r1 = zeros(height1,width1,1);

for i = pt1y:pt2y-1
    for j = pt1x: pt2x-1
        r1(i-(pt1y-1),j-(pt1x-1),1) = s((((i-1)*208)+j)*3)/255;
        r1(i-(pt1y-1),j-(pt1x-1),3) = s((((i-1)*208)+j)*3+1)/255;
        r1(i-(pt1y-1),j-(pt1x-1),2) = s((((i-1)*208)+j)*3+2)/255;
    end
end

imwrite(r1,filejpg2,'jpg');
```

where pt1x, pt1y, pt2x, pt2y are the coordinates taken in coord.txt, r1 is the RGB image matrix received from the AIBO, s is the matrix of the new image of face extracted and converted in jpg.

When a centered face is obtained the picture still needs to be resized to normalized and fix dimensions. After some experiments, 130x130 was chosen for the centered image's dimensions. A function had been written before in the framework of the Eigenfaces project and was reused in this context with the new value of the final dimensions.

Another parameter to pass for resizing is the final eyes position.

Either we assume eyes correspond to an approximate position within the rectangle or we apply a second detection-dedicated to eyes.

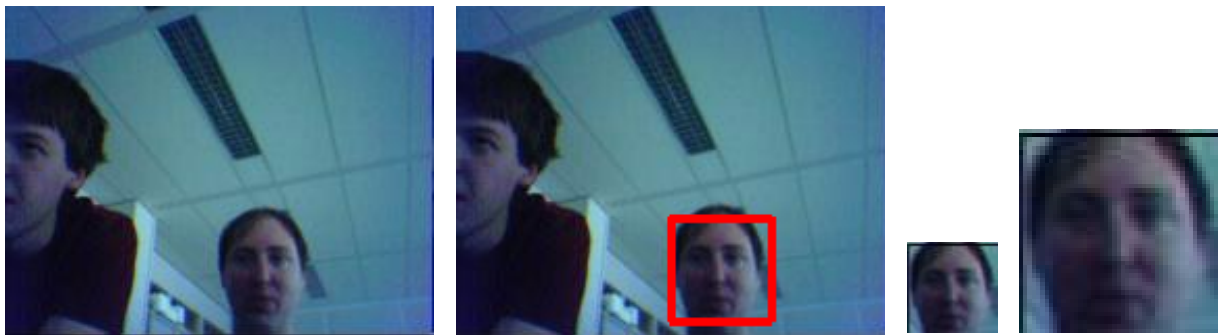
In the first option it is assumed than eyes are located as follows:

Left eye: ( $\frac{1}{4} * \text{Height}$  ,  $\frac{1}{4} * \text{Width}$  )

Right eye: ( $\frac{1}{4} * \text{Height}$  ,  $\frac{3}{4} * \text{Width}$ )

where the origin axis is the top-left corner of the frame.

The result of all the steps described above are illustrated with a series of chronological photos in **Figure 16**:



**Figure 16 Steps of face extraction**

### Eye Detection

The alternative solution to automatic eye-position resizing is to find exactly where the eyes are situated. In the same logic as face detection, let us use an eye detector. But unlike the “former/latter” case, there is officially no existing eye detector, which means it would have to be set up by our own. As introduced in Chapter 3, it is not worth reconsidering the implementation itself of detection, the only part to change is th samples on which the classifier is generated. That assumes a huge amount of photos of non-faces and faces must be gathered. Moreover the eyes' coordinates have to be specified for each image of face.

Following these requirements, it was made use of BioID Database [19] containing more than 1,500 thousands of faces in addition of 2,000 pictures of non-faces found over the Internet.

OpenCV includes a haartraining application that creates the classifier from the samples gathered. The classifier is then stored in an XML file.

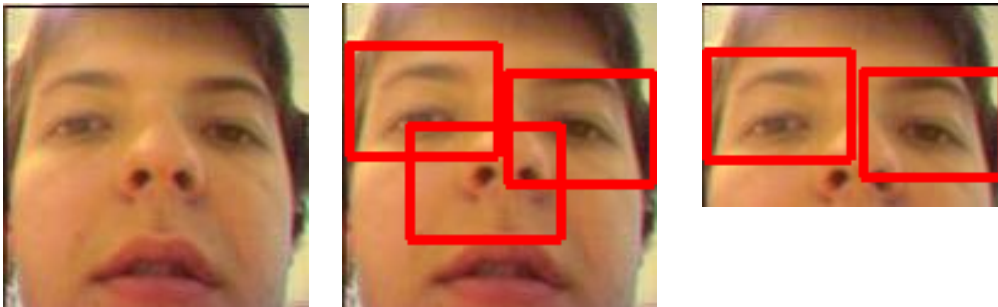
After having followed the usage scheme (by training the classifier a very reasonable number of 25 times), an eye classifier is obtained, whose size is 61 Kb. For information, all the classifiers released by OpenCV have a size of more than 1 Mb. Consequently even with getting more than three

thousand samples the classifier is very weak. This bad sign was confirmed during the test evaluation. Only sometimes the detection was relevant, otherwise it did not find anything nor find a non-eye. The solution was to add more samples but the only thing we could do was to increase the number of negative samples because we do not have in possession more face samples with eye coordinates. But the maximal size of xml file reachable was 99Kb.

However a short look on the forum of OpenCV showed that some other researchers achieved some robust eye detection, especially one classifier delivered by a member of the OpenCV community works pretty well on pictures even on medium resolution. For AIBO pictures results are more mitigated, but generally at least one eye is detected when the face is at a distance below one meter. The *eyedetect* application calls this new classifier and contains more routines implemented to take in charge different situations:

- if two eyes are found, the centres' coordinates of the two frames that localize the eyes are picked up and submitted to the resizing program (via the text file CoordEye.txt).
- if only one eye is found the other eye's position is deducted by assuming a symmetry of the face.
- if no eye is found the automatic resizing seen before is applied.

Moreover, eye detection is actually processed on half of the picture because the eye detector usually confuses eyes with other features of the lower face part, like the nose, the mouth and the chin. A practical example describes some results:



**Figure 17 Steps of eye detection**

In practise in about only 50% of cases one eye is at least discovered. The limit between achievement and failure is a question of distance to the AIBO. Moreover, even when eyes are found, the frame drawn is not exactly centred on the eye, which the frame centre does not match the eye centre. Eventually this additional eye detection was dropped, because no better eye classifier was available. But the advantage of the implementation made enables to only change the classifier and try again, without changing the codes: if someone manages to find or create a classifier sufficiently powerful for the AIBO camera, he merely replaces it.

#### **4.5.3 AudioFace Fusion**

AudioFace Fusion accounts for the gathering of two sub-programs: audio recognition and face recognition to keep modularity in our implementation design. In addition, one of them can be tested regardless the other ones. Therefore there are three parts dedicated to recognition.

There is a distinction in our implementation between what is called Recognition (see the *Rec* m-files) and Interpretation (see the *Interpret* m-files). This split concerns scripts for sound, face, and fusion between sound and face. In other words, each of these three aspects is separated on one hand in a pure execution of the recognition algorithm and on the other hand an interpretation designed by the user related to how the AIBO should interpret and react according to the results of recognition. The point of adopting such a model is that it is modular and respects a natural scheme in any perceptive system: first it receives a stimulation the dog analyzes and finds out the identity for the stimulation's origin, then it interprets how to consider it and how to react properly. The techniques for recognition presented just before belong to the first class. For the fusing part, the only thing done consisted in gathering all the output from both Sound and face recognition programs. Then interpretation is processed.

#### 4.5.4 Interpretation

Each perceptive entity contains an interpretation part, let alone when merging them. In the case of a unique way of perception decisions are made by taking into consideration the nature of the input stimulus and the background (memory, current mood) of the subject. A specific face is identified, no matter if it is known or not, but the way it is deemed -and the following reaction- is a different thing. That is why a reaction must be attributed to each case of recognition. This attribution can be arbitrary or depend on other parameters. Typically if the dog is currently in an aggressive mood, its set of reactions will be globally modified. The current implementation starts a bit with this parametered interpretation approach (see 4.5.7 Parametering recognition). It is not related to the subject's mood but its memory: it classifies among the known persons those who are its masters and the ones who are not. Therefore judgment will be different. The interpretation's output is basically a figure (to get rid of endless descriptive denominations) which represents a specific behaviour of the robot. Because of a lack of time, this part has not been developed. So what is currently available is two sound motions expressing happiness and aggressivity. Compared the huge number of figures (possible reactions) to the poor one of motions, figures refer most of the time to the same motion. Here is a demonstration of sound interpretation and how information is passed to make the AIBO act:

```

function [answer]=AudioRec_Interpret()

%answer=1-> the sound is unknown
%answer=2-> the sound is known but is not from its master
%answer=3-> the sound is from its master

[val,dist2average]=AudioRec;
[res_pic,thr1,thr2,thr3,master_vec,margin1,margin2]=parameters;

if (val==0)
    %...It does not recognize a known sound
    answer=1;
else
    nb_master=0;
    for i=1:length(master_vec)
        if (val==master_vec(i))
            nb_master=nb_master+1;
        end
    end
    if (nb_master==0)
        %...He/she is not its master(Bite!)
        answer=2;
    else
        %...He/she is its master
        answer=3;
    end
end
end

answer=num2str(answer);

```

The (previous) function *AudioRec\_Interpret* outputs a figure in the variable *answer* to specify the behaviour to adopt. This figure is reused in *AudioRec\_test.c* to indicate the behaviour to get:

```

static char * number;
static char * answer1;
answer1= (char*)getanswer1();
number="3";

if(answer1[0]=='3')
{
    behavior->GoodReaction();
}
else if(answer1[0]=='2')
{
    behavior->NaughtyReaction();
    behavior->GoodReaction();
}
else
{
    behavior->NaughtyReaction();
}

```

Interpretation should be introduced differently however for the fusion of sound and image. Of course the elements discussed previously are still true but are positioned on top of a preliminary phase: solving possible disagreements between speech and face identification.

Two functions were written to provide a single output from two conflicted inputs: *makechoice* and *makechoice2*. the first one displayed below, the second ones follows the same structure:

```
function [ans]=makechoice(dist2average,Eproj,thr1,thr2,margin1)
    estim_audiorec=abs(thr1-dist2average)/thr1;
    estim_facerec=abs(thr2-Eproj)/thr2;
    estim_audiorec=estim_audiorec+margin1;

    if(estim_audiorec>estim_facerec)
        ans=0;
    else
        ans=1;
    end
```

*makechoice* intends to give a priority between the statement that it is a face (or not) and the statement that it is a known (or unknown) sound.

Then, *makechoice2* intends to give a priority between the statement that it is a known (or unknown) face and the statement that it is a known (or unknown) sound.

#### 4.5.5 GUI Functions

These functions have the particularities to gather and combine all the parts of the previous implementation.

##### Learn someone

Is dedicated to voice and face's learning of someone. It only gathers data for further tests. In practise it first records a sound sample of four seconds before taking a series of pictures to be saved automatically in a database later useful for the Eigenfaces algorithm.

##### Sound Rec

Records a four second-sample through the AIBO stereo microphones and in a second step submits it to sound analysis. Also handles the AIBO reaction.

```
ThreadArg* data;
data = (ThreadArg *) threadarg;
tudAIBOInterface* pMyAIBO = data->AIBO();
tudAIBOSound sound;
sound.SetDuration(4000);
pMyAIBO->Receive(eMicro, &sound);
sound.WriteWav("./test/test.wav");
```

##### Face Rec



Intends to detect a face by scanning its vertical vision field with its moving head while taking continuously pictures. Then, passes them to the face detection and if necessary to face recognition (in case of an existing face). Also handles the AIBO reaction.

```
ThreadArg* data;
data = (ThreadArg *) threadarg;
tudAIBOInterface* pMyAIBO = data->AIBO();
char * filename1;
filename1 = "Picture1";
tudAIBOImage picture1;
pMyAIBO->Receive(eCamera, &picture1);
if (0 != strlen(filename1))
{
    TRACE("Writing data to file");
    FILE *fp;
    fp = fopen(filename1, "w");
    if (fp)
    {
        fwrite(picture1.GetData(), 1,
            picture1.GetSize(), fp);
        fclose(fp);
    }
}
```

### AudioFace Rec

Mixes the last two functions, first Sound Rec and then Face Rec.

Note that unlike a real animal, ears and eyes do not pick up information from the outside simultaneously. Both perceptive processes use the common Urbi framework thanks to some routines of it, Receive() for sound in one hand and Receive() for image in the other hand, consequently they are bound to share time space.

### 4.5.6 Parametering recognition

Parameters for recognition can be set and changed in parameters.txt. The point of it is that it provides a central nd top control point to lead and adapt recognition, moreover without needing to recompile systematically the m-files. Each m-file in relation with recognition have these values at their disposal.

Here is an example. The text file actually only contain the values, without the comments. These are displayed for this report only.

```
30: res_pic
length(the width is the same) of the resized picture
0.3: thr1
threshold of sound. Below it the sound is considered as unknown
30: thr2
threshold of face. Below it the face is considered as a face
16: thr3
threshold of face. Below it the face is considered as a known face
[1 2]: master_vec
vector containing the figures in the database of the master(s)
0.2: margin1
margin give an advance to sound compared to face (in the face-nonface
test)
0.2: margin2
```

## Chapter 5 Results

Some tests were realized to measure in which degree the final result matched the system specifications.

First of all, sound identification works pretty well. Considering a database of five speech samples recorded from different persons, tests reveal that the success rate is situated at about 80%. These tests cover the situations where the tested subject is unknown or known (registered in the database). But one fundamental point is the quality of the referenced-sounds. These must be long (a last of 4 seconds was chosen), and provide a good description of someone's voice characteristics. Indeed, it was noticed that when a sample from the database is poor, when the utterance is not pronounced loud then tests are of lower quality. Moreover it was found out that noise conditions must be close between the creation of the database and the test session, no matter how high is the noise level (under a certain level of course).

Finally the situation where nobody speaks, in other words speech detection, has not been treated. At least it was submitted to test, but always gave absurd results, most of the time silence is assimilated to a known person. Actually, this phenomenon appears because silence corresponds to a frequency floor with no significantly dominant components, so in the 20<sup>th</sup> dimensioned sound space modelled with MFCC technique "clusters" are localized around the centre. The distance between the sample of silence and the average of the registered voices is close.

For face identification, detection works well as well, even if the method of taking a series of pictures reveal a bit long and fastidious. The algorithm seems very robust and do not suffer from the low resolution of the camera. Sometimes the AIBO skips the face from on picture to the other. This means a short part of the face was out of the camera frame. To remove the problem the number of pictures should be increased, but the time for detection will be stretched.

Afterwards, the program with Eigenfaces was tried by recording a set of faces from five people. First of all the volunteers were tested –not on purpose- in different lights conditions, and results were not satisfying. However with a constant light level performances are much better. Note however that only five people's face are utilized. Face recognition seems more exigent than sound one, and suffers from the camera resolution. We can hardly afford a huge database because confusions would increase critically.

Finally, because of the time limitations, the software was not combined with the sound localization program. Hence the AIBO is only looking straight ahead (and in the vertical plan) to check someone's presence and identity. Nevertheless speech recognition can be processed regardless the location of the speaker.

## Chapter 6 Conclusion and Recommendations

### 6.1 Conclusion

In the final state of the project, a software dedicated to speech and face recognition within the AIBO was obtained. The recognition task is delegated to a remote computer via a wireless LAN. It is possible to make the dog learn new people -with the *learn* mode, and give them a status (currently unknown person, familiar person or master). The AIBO then uses its knowledge to react properly when meeting a new person. The attitudes of the dog are configurable. It is obvious that the more the dog knows people and the less its answers are correct, however before ten people registered results are satisfying.

### 6.2 Recommendations

In this part some prospects are suggested for future work.

#### **AIBO expressions**

AIBO expressions is a part which was not so developed in the project. The robot under-use the possibilities of the interpretation program and cover all situations with merely two kinds of expressions (happy bark and grumble). Work can be done on AIBO motions to make it react with more accuracy and complexity in its expressions.

#### **Eye detector**

As mentioned in the implantation part, the eye detector was irrelevant, first because if the face is far more than 20 cm from the camera, eyes are skipped, second because even with an achieved detection the selective frame is not properly centred on the eye. However performances deeply depend on the classifier used, and replacing the current one by a better one should solve the problem instantaneously.

#### **Run-time video detection**

A discussable point in face detection was what is more efficient in terms of fastness between applying te algorithm over a video stream or a series of static pictures. The second option was decided but without any comparison with the other one. What would be interesting is to convert the video image structure from the AIBO into the one handled by the detection algorithm so as to control the existence of faces from the AIBO cam.

#### **Sound localisation**

The current developed software do not allow the AIBO to interact with the entire 3D environment but only with the front plan. Like a real dog, the AIBO should turn its head towards someone if it hears his/her voice (and then pass to identification part).

#### **Environment variations**

Taking pictures in different light conditions tend to harm the performances and therefore should be taken in consideration somehow. A possible strategy is to make the database richer by adding pictures of a same person with different lighting degrees.

## Chapter 7 Glossary

**MFCC:** Mel Frequency Cepstrum Coefficients. One of the most popular method in the field of speech recognition research.

**mcc:** MATLAB Compiler. Converts MATLAB programs into self-contained applications and software components. Applications and components created with the Compiler do not require MATLAB to run.

**Recognition:** Process of finding out some characteristics of an entity.

**Detection:** Process of finding out the existence of an entity in a certain location.

## Chapter 8 Bibliography

- [1] <http://www.sony.net/Products/AIBO/> - Sony AIBO website
- [2] <http://www.eu.AIBO.com/> - Sony AIBO European website
- [3] <http://www-2.cs.cmu.edu/~tekkotsu/> - Tekkotsu website
- [4] François Serra, Jean-Christophe Baillie, “AIBO programming using OPEN-R SDK”, Tutorial, June 2003
- [5] <http://www.urbiforge.com/eng/> - URBI official web page
- [6] Jean-Christophe Baillie, “URBI Language Specification”, 2005.
- [7] Jean-Christophe Baillie, “URBI: A UNIVERSAL LANGUAGE FOR ROBOTIC CONTROL”, International Journal of Humanoid Robotics, 2004, World Scientific Publishing Company
- [8] <http://www.mmi.tudelft.nl/~siska/AIBO/> - MMI AIBO Team
- [9] <http://itp.nat.uni-magdeburg.de/MATLAB/toolbox/compiler/> - Tutorial for MATLAB compiler
- [10] Marlon Richert, Tijmen Roberti, Wouter de Vries, “Sound source localization using the AIBO ERS-7”, 2004
- [11] Mohamed Mouldouira, Merouane Qastalane, Mickael Thouzery, “Reconnaissance de visages par Eigenfaces”, November 2004
- [12] Li Tan and Montri Karnjanadecha, “Modified Mel-Frequency Cepstrum Coefficients”
- [13] Minh N.Do, “An automatic speaker recognition system”
- [14] MMI AIBO Team, “AIBO Interface-Specification”, January 2005
- [15] Iulia Dobai, “Personality model for companion dog”, July 2005
- [16] Silvia Oana Tanase, “Watch dog”, April 2005
- [17] <http://www.sourceforge.net/projects/opencvlibrary/>
- [18] <http://www.intel.com/technology/itj/2005/> - Learning-Based Computer Vision with Intel’s Open Source Computer Vision Library
- [19] <http://www.bioid.com/> - databases for detecting and recognizing persons with face, voice and lip movement recognition

## Appendix A

### AIBO specifications

- **CPU** 64-bit RISC Processor
- **CPU clock speed** 576 MHz
- **RAM** 64 MB
- **Program media**
  - Dedicated AIBO robot "Memory Stick™" media
- **Moveable parts** (Total 20 degrees of freedom)
  - Head - 3 DOF
  - Mouth - 1 DOF
  - Legs - 3 DOF x 4 (legs)
  - Ears - 1 DOF x 2 (ears)
  - Tail - 2 DOF
- **Input section**
  - Charging contacts
- **Setting switches**
  - Volume control switch
  - Wireless LAN switch
- **Image input** 350.000-pixel CMOS image sensor
- **Audio input** Stereo microphones
- **Audio output** Speaker 20.8mm, 500mW
- **Integrated sensors**
  - Infrared distance sensors x 2
  - Acceleration sensor
  - Vibration sensor
- **Input sensors**
  - Head sensor
  - Back sensor
  - Chin sensor
  - Paw sensors (\* 4)
- **Power consumption** Approx. 7 W (in standard mode)
- **Operating time** Approx. 1,5 hours (with fully charged ERA-7B1, in standard mode)
- **Dimensions** Approx. 180 (w) x 278 (h) x 319 (d) mm
- **Weight** Approx. 1.65 kg (including battery pack and "Memory Stick™" media)
- **Wireless LAN function** Wireless LAN module (Wi-Fi certified) Internal standard compatibility: IEEE 802.11b/IEEE 802.11 Frequency band: 2,4 GHz Wireless channels: 1 – 11 Modulation : DS-SS (IEEE 802.11 – compliant) Encryption : WEP 64 (40 bits), WEP 128 (104 bits)