Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMTICA

Facultat de Matemàtiques
Universitat de Barcelona

Quantitative analysis of non-verbal communication competence

Alvaro Cepero Amador

Directores: Sergio Escalera Guerrero
and Albert Clapés i Sintes
Realizat a: Departament de
Matemtica Aplicada i Anlisi. UB
Barcelona, 15 de juny de 2013

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisers Dr. Sergio Escalera and Albert Clapés for the continuous and enthusiasm support as well as their patience, motivation, and immense knowledge. Their guidance helped me in all the time of this thesis, specially during the most difficult tasks. I could not have imagined having better advisers and I really enjoyed working with them.

I would also thank the whole Egluu team (the place where I'm currently working), specially Xavi Asens, for his patience and being so flexible and understanding while I was working on this project, and to Joan Piferrer for his corrections on my catalan.

Last but not the least, I would like to thank my family: my parents Regino Cepero and Caridad Amador, for all the support and love during my lifetime and for giving my the opportunity to study in an European University, without them this project wouldn't be possible, and my sister Iliana Cepero, a worthy role model for me to follow, who has always been encouraging me.

Abstract

English

Oral communication is an escencial part of human life. Efeective communication skills are required not only professionally, but also in our daily life. Due to its importance in our daily activities it is essential to find methods that improve our communication skills. Several studies in the field of psychology on verbal and non-verbal communication has been done in the past decades, however from the point of view of Artificial Intelligence we are still at the begining of a long way to go. In this project, we propose a method that is able to extract behavioral cues from subject's presentations based on the analysis of multi modal data provided by the Kinect camera and applying different statistical classifiers that will allow the system to predict the quality of the presentation. All the experiments were took from students at Universitat de Barcelona while defending its class. works.

Resumen

La comunicación oral es una parte escencial en la vida del ser humano. Es importante adquirir habilidades comunicativas no sólo profesionalmente, sino también para nuestra vida personal, a la hora de relacionarnos con otras personas. Debido a la importacia de la comunicación oral en nuestras vidas, es primordial encontrar métodos que nos ayuden a mejorar nuestras habilidades comunicativas. Varios estudios en el campo de la psicología con respecto a la comunicación verbal y no verbal, se han llevado a cabo en las últimas décadas, sin embargo, desde el punto de vista de la Inteligencia Artificial aún estamos al comienzo de un largo camino por transitar. En este proyecto, proponemos un sistema que es capaz de extraer señales de comportamiento a partir de prensentaciones orales de diferentes personas basado en el análisis de datos multi modales sacados de la cámara Kinect y a partir de la aplicación de diferentes clasificadores estadı'sticos, que permitirán al sistema predecir la calidad de la presentación. Todos los experimentos se llevaron a cabo durante las exposiciones de estudiantes de la Universitat de Barcelona mientras realizaban la defensa de sus trabajos de clase.

Resum
La comunicació oral ès una part essencial en la vida del èsser humà. És important adquirir habilitats comunicatives no només professionalment, sinó tambè per la nostra vida personal, a l'hora de relacionar-nos amb altres persones. Degut a la importància de la comunicació oral ès primordial trobar mètodes que ens ajudin a millorar les nostres habilitats comunicatives. Diversos estudis en el camp de la psicologia respecte a la comunicació verbal i no verbal s'han dut a terme en les últims dècades, no obstant des del punt de vista de la Intel·ligència Artificial encara estem al començament d'un llarg camí per transitar. En aquest projecte, proposem un sistema que ès capa d'extreure senyals de comportament a partir de presentacions orals de diferents persones basats en l'anàlisi de dades multi-modals, trets de la cámera Kinect i a partir de l'aplicació de diferents classificadors estadístics, que permetran al sistema predir la qualitat de la presentació. Tots els experiments es van dur a terme durant les exposicions d' estudiants de la Universitat de Barcelona mentres realitzaven la defensa dels seus treballs de classe.

Índex

Capítol 1

Introduction

Man has always being puzzled by his own nature and senses. How we can communicate, how we understand the world around us or how we perceive it. In recent times it has been an important investment of resources towards the fact of making a computer see, and more important understand what it sees. Nowadays we have yet a long path to go, and maybe this could be the reason why the field of computer vision is becoming so appealing to the scientific community; because its countless applications in our daily lives. Nowadays society is demanding new kind of competences of its citizens and especially its professionals. With the implementation of the bachelors degree in the European Higher Education Area, the concept of competences became even more important in the educational field. One of the main goals of this plan is to provide specific skills and competences in the student. Oral expression and communication is among the most relevant competences in everyones life. A nationwide survey conducted in 1988 by the American Society of Training and Development and the Department of Labour found that oral communication skill were ranked within the top five skills required for potential hires.Due to the importance of communication in our daily life it is absolutely crucial to study methods to improve our communications skills and therefore learn how to express ourselves better, but first we should find methods to measure our verbal and non-verbal communication as a feedback .

The main idea behind this project is to analyse the quality of non-verbal communication in common projects defensal using a Computer Vision framework and Artificial Intelligence. Computer vision could be define as a field that includes methods for acquiring, processing, analysing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information, e.g., in the forms of decisions. In other words through Computer Vision we try to give a similar, if not better, capability to a machine or computer to interact with the environment.

1.1 Project context

1.1.1 The problem

Usually while making a presentation, several factors could affect our final grade, such as lack of preparation, nervousness, or simply our body language does not express correctly what we are trying to communicate. What we are trying to fulfil in this project is to find a way of processing all this non-verbal data and be able to perform a quantitative analysis regarding the quality of the presentation.

1.1.2 Goals

The main goals in this project is to create a solid model to analyse the raw data provided by the kinetic and translate all this information into a coherent and logical output, that is to say, a system that would be able to interpret non-verbal cues transmitted by students during an oral presentation and then predicting a grade, according to the quality of the presentation. As we stated before, the RGBD information is supplied by the kinect camera and the Kinect SDK, developed both by Microsoft in 2010. This API is not open source though, we had tried throughout this project to use open source software as far as possible.

1.1.3 State of the art

In this context, Social Signal Processing is the field of study that analyses communication signals by means of different sensors, such as microphones or cameras, and applies some kind of pattern recognition strategies. Some examples of application are social interaction analysis on small groups. According to the authors of Vinciarelli, Salamin, & Pantic (2009) psychologists have grouped all possible non-verbal behavioural cues into five major classes, though most recent approaches are based on three of them: ***gestures and postures*** (considered as the most reliable feature about people's attitude towards others), ***face and eye behaviour*** and ***vocal behaviour***. The two former kinds of communication can be evaluated as successful or not given certain criteria: knowledge transmission, richness of the language expression, discourse coherence and cohesion, and so forth. Whereas the verbal communication is often quite explicit to a human observer, non-verbal signals are relatively subtle regardless the huge amount of information they provide about the communicating subject and are not always easily separable, so as to be clearly identified and evaluated individually, but they emerge as a whole and complex behaviour. There is a vast literature in psychology studying the non-verbal component in communication act, but from the point of view of Artificial Intelligence (AI) we are at the beginning of a long way to go. While the verbal communication has been studied for many years by the Natural Language Processing field, the study of the non-verbal communication from a multi-modal social signal point of view, including visual sources, is a relatively recent field of research.

Several works have been recently performed in the Social Signal

Processing field in order to analyse the non-verbal communication of subjects in group interactions Vinciarelli, Pantic, & Bourlard (2009). Most of these works focus on audio analysis, and main goals are based on dominance, influence, and leadership recognition. In the work of Olguín et al. (2009) the authors present implementation of a platform for measuring and analysing human behaviour in organizational face-to-face settings using wearable electronic badges. The work of McCowan et al. (2005) presents the recognition of group actions in meetings modelled with HMM-based approaches from audiovisual-featured observations. Other recent approaches for dominance analysis in group interactions have been also proposed Escalera et al. (2010). The work of Pan et al. (2012) presents a Bayesian framework that models dominance skills based on audio input sources. In Sanchez-Cortes et al. (2012), the authors model a multi-modal audio-video system to recognize leadership in group interactions. The system is defined based on simple multi-modal features under controlled face-to-face interaction environments. In a similar scenario, the proposal of Marcos-Ramiro et al. (n.d.) defines multi-modal cues for the analysis of communication skills in a upper body set-up. A more general purpose approach is presented in Mohammadi & Vinciarelli (2012), where prosodic features are computed to define a set of relevant traits of subjects in oral communication settings. Very few works have been reported on the analysis of non-verbal communication as a competence skill in e-Learning scenarios. The authors of Tanaka et al. (2012) presents a system based on audio analysis from mobile devices to analyse the communicative skills and provide relevant feedback to subjects that may suffer from communication problems and some degree of autism.

1.2 Proposal

As we stated earlier, our aim throughout this project is to provide a solid tool that would be able to analyse quantitatively the non-verbal cues of a subject during an oral presentation. By the means of several machine learning methods, our system would l·learn"from a reduced subset of characteristics taken from the multi-modal data supplied by the Kinect camera, namely audio, color and depth data, thus we will try to present an automatic categorization system of presentations as e-Learning tool for evaluating the non-verbal communication competence. We will record novel data set of oral presentations, and train several classifiers based from the score defined by different professors. We will focus on defining some patterns behaviours based on the multi-modal data and select the most discriminative features.

1.3 Costs and planning

This project was conceived as a full semester project, that is to say a six month project. This project was divided in four major sections:

1. Data acquisition
2. Presentations recording
3. Feature extraction
4. Classification

Most of the work relied on the data provided by the Kinect, and considering this is a relatively new technology it is quite well documented and its community is in growing process, although most of the code is still closed, thus is comprehensible that most of time spent on this project will be focused on the study and implementation of the

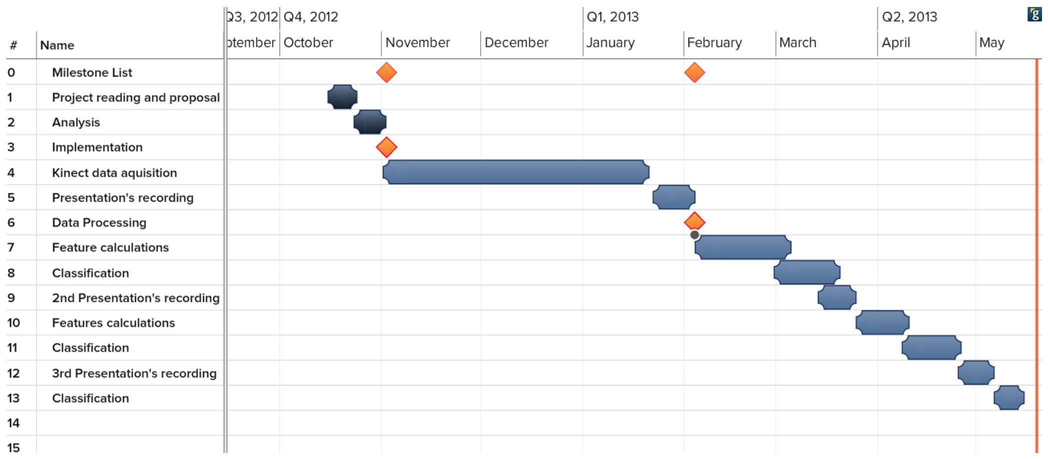


Figura 1.1: Costs and plannification gann's diagram

process of obtaining multi-modal data from the kinect. An approximate of not less than 100 hours will be necessary to the task of making the kinect work and getting the raw data from it. Once we are able to obtain significant data from the camera, the next task will be to test the software in a real environment, that is to say , on real student presentations. In general we plan to record three different sets of presentations, preferably within three different levels of studies, an approximate of 30 hours. The first group of presentations will be held at the Universitat de Barcelona between the 21 and 25 of January of 2013. Approximately 15 different students will be recorded defending their bachelor's thesis. The second group will be 8 presentations from an 8th semester Bachelors class at Universitat de Barcelona. Finally the third group would represent the highest level of education of all three, and would be approximately 12 students of a master program held at Universitat de Barcelona defending their final projects for an specific subject.

Once all the recordings are made, the next logical step will be to process all these raw data and convert each presentation into a simplified feature vector. The feature extraction method will be coded once, but would be applied for each presentation session. As it can be seen in figure 1.1 the process of feature extraction will take no more than two weeks of works, that is to say an approximate of 40 hours. The final step will be the classification, this method would require the study, research and finally an implementation using different classification methods. This task is expected to be at least one week of work or 25 hours.

1.4 Organization

The rest of the document is organized as follow. Chapter 2 introduces the different methods, hardware, programming languages, libraries and systems that we had directly applied or studied throughout this project. In chapter 3 we get into a more detailed explanation of our system, focusing primarily on parts of the implementation. In chapter 4 we will explain how was validated our model with some illustrative examples. Finally in the last chapter will summarize the whole project and results obtained as well as the proposal of the future work that can be done.

Capítol 2

Method analysis

In order to develop this project, it has been used the latest Kinect SDK developed by Microsoft, besides a set of Open Source libraries and public algorithms. In particular, OpenCV to process images at a high level. In the whole project it has been used up to three different programming languages: C++, MATLAB and python, and several libraries and resources has been studied, analysed and finally, some of them applied.

2.0.1 Programming languages

- **C++**. As the kinect is developed by Microsoft, almost all the languages from the .NET(C++, C# and VB.NET) platform are well supported. C++ and C# are among the most highly valued languages in the .NET family and there are not big advantages of one over the other, hence the election between the two came as a personal preference, and we finally opted for C++ over C# .

- **MATLAB** (**mat**rix **lab**oratory) is a numerical computing environment and fourth generation programming language and as the name suggest it is oriented to matrix manipulations, besides allows plotting of functions and data, implementation of algorithms and many other interesting features. MATLAB is wired with multiple toolbox that enhance all its power. MATLAB comes with a Statistics Toolbox that can be used to solve both supervised and unsupervised machine learning problems. Because of its flexibility when working with matrices and vectors, MATLAB is used in this project in the classification phase, specifically an interesting implementation of the Adaboost algorithm that we will explain later.

- **Python** is a general-purpose, high level programming language whose design philosophy emphasizes code readability. Python is well known for its power, simplicity and extensibility. One of its weakness is its performance when compared with other non-interpreted languages such as C or C++, but as this project was not planned to be a real time application this fact was not a real problem, besides there are implementations of third party libraries like numpy (for multi dimensional array and matrices processing) that are comparable to some C++ implementations or Pypy which is a fast compliant alternative implementation of the Python language. Another advantage of python is the number of third party libraries that exists over the internet, including *scikit-learn* (2013) or *mlpy* (2013) for machine learning analysis. Python is used in this project in the phase of extracting the characteristic feature vector of each presentation and in the classification task. It is worth mentioning that python's code can be easily embedded in C++ applications.

2.0.2 Libraries and Frameworks

- **OpenCV** (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate itself. As stated before this library is cross-platform and there are implementations for Windows, Mac OS X and GNU/Linux. The library has more than 2500 optimized algorithms, and it includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. This library is mainly used in this project to perform with a high level of efficiency operations over matrices, and to carry out some machine learning tasks as well.

- **SQLite** is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. As this project is focused on behavioural analysis, we are particularly interested in analysing the body language and the different poses that the students adopt during his/her presentation. Taking advantage of the capabilities of the kinect in providing user's world coordinates in real time, the process of saving the user's coordinate that would be later analysed was one of the first important decisions that was needed to be taken into account. We needed a tool that could be easily integrated into the application, was as fast as possible, and finally a tool that does not represent an extra layer in software nor hardware. With all these constraints our first choice was SQLite. One of the main benefits of using SQLite over other Database engines is that it does not need a standalone process with which the application communicates. Instead, the SQLite library is linked and thus becomes an integral part of the application.

- **Microsoft Kinect SDK**

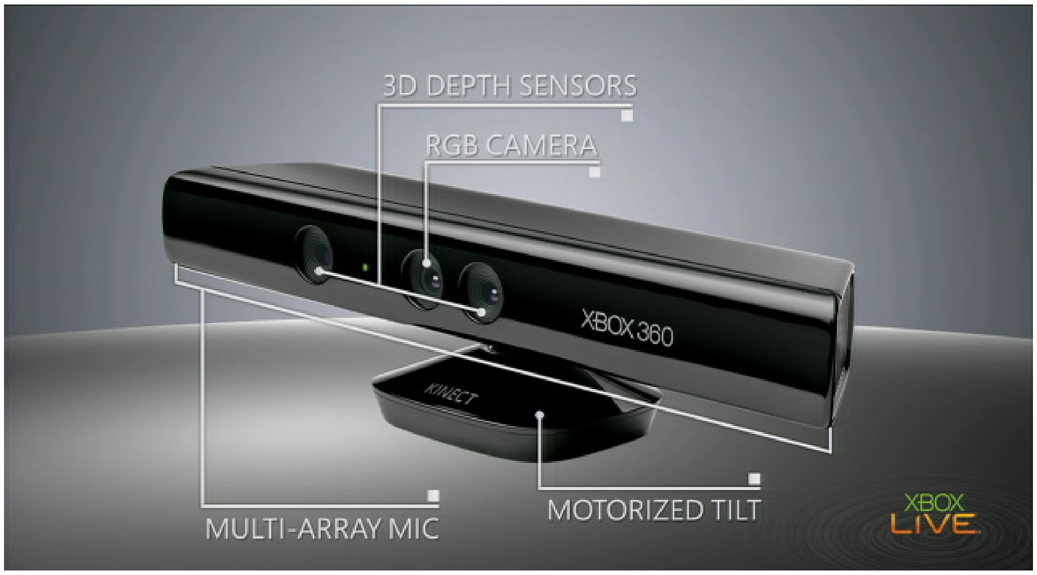


Figura 2.1: The Kinect camera

The latest version of Microsoft Kinect SDK(for Windows), allows developers to use high level languages such as C++, C#, VB.net and create applications, giving the computer ”eyes, ears”and the capacity to use them. Providing a natural interaction between humans and computers by simply gesturing and speaking. Among the main functionalities, we can find:

1. Compute the distance between objects, by the means of the infra-red camera which offer information about depth.
2. Skeletal tracking of one or two people that are in the vision angle of the camera.

3. Advanced audio processing by means of a four matrix microphones.

The Kinect contains three vital pieces that work together to detect the motion and create the physical image on the screen: an RGB colour VGA video camera, a depth sensor, and a multi-array microphone. The camera detects the red, green, and blue colour components as well as body-type and facial features. It has a pixel resolution of 640x480 and a frame rate of 30 fps. This helps in facial recognition and body recognition. The depth sensor contains a monochrome CMOS sensor and infra-red projector that help create the 3D imagery throughout the room. It also measures the distance of each point of the player's body by transmitting invisible near-infra-red light and measuring its "time of flight"after it reflects off the objects. The KinectTM infra-red sensor displays a structured/codded matrix of points through the environment.Then, each depth pixel is computed by sampling the derivative of the higher resolution infrared image taken in the infra-red camera. This value is inversely proportional to the radius of each Gaussian dot, which is linearly proportional to the actual depth. The method is based on inferring pixel label probabilities through Random Forest (RF) based. The microphone is actually an array of four microphones that can isolate the voices of the player from other background noises allowing players to use their voices as an added control feature. The kinect features a multi-array microphone that consists of four separate microphones spread out linearly at the bottom of the Kinect, with each channel processing 16-bit audio at a sampling rate of 16 kHz. By comparing when each microphone captures the same audio signal, the microphone array can be used to determine the direction from which the signal is coming, besides speech can be recognized in a large room where the speakers lips are more than a few centimetres from the microphone, in our system the distance between the speaker and the kinect is less than 3.5 meters. These components come together to detect and track 48 different points on each player's body and repeats 30 times every second.

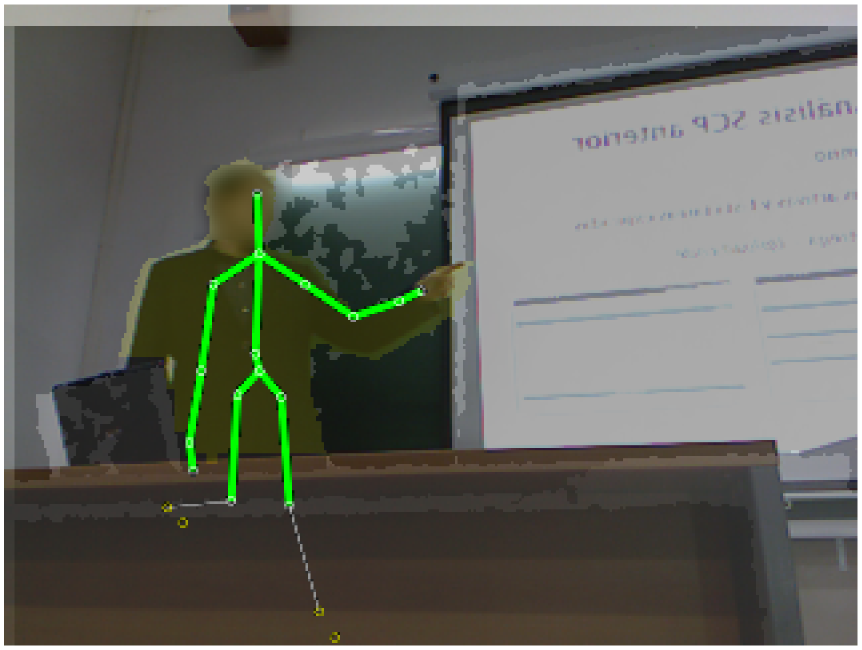


Figura 2.2: RGB-D overlayed data

In the 2.2 we can appreciate all the information provided by the kinect, in three different layers. RGB, Depth and the skeleton points.

- **NUI API** The Natural User Interface (NUI) is the core of the Kinect for Windows API. Through it, it can be accessed the audio sensor data (streamed out by the audio stream) and the color image and depth image (streamed out by the colour and depth streams). Thanks to this API, it can be carried out the skeletal tracking, and know the distance between objects and the camera. The kinect runtime also implements:

- A software pipeline that can recognize and track a human body. The depth information is converted into skeleton joints in the human body. Besides the runtime also provides a DepthPlayerIndex, which indicate, if more than one person is tracked in front of the camera, who is he/she.
- Integration with the Microsoft Speech APIs, so a speech recognition can be implemented. Also, this API makes possible to add voice commands to the application
- A tight integration with the Face Tracking SDK, which makes it possible to track human faces

Along this project all three are used, as we will explain later.

2.1 Software and hardware requirements

Unlike other Kinect libraries, the Kinect for Windows SDK, as its name suggests, only runs on Windows operating systems. Specifically, it runs on x86 and x64 versions of Windows 7. It has been shown to also work on Windows 8. Because Kinect was designed for Xbox hardware, it requires roughly similar hardware on a PC to run effectively.

2.1.1 Hardware Requirements:

- Computer with a dual-core, 2.66-GHz or faster processor
- Windows 7compatible graphics card that supports Microsoft DirectX 9.0c capabilities

- 2 GB of RAM (4 GB or RAM recommended)
- Kinect for Xbox 360 sensor
- Kinect USB power adapter

In order to execute the applications developed with the Kinect SDK, it will be necessary to use a native windows environment. This means that it would not be possible to run them in a virtual machine. The Software requirements of the Kinect SDK are:

- Microsoft Visual Studio 2010 .
- Microsoft .NET Framework 4.
- DirectX Software Development Kit.

2.1.2 Libraries

- OpenCV
- Microsoft SDK
- LibSVM
- LightSVM
- Error-Correcting Output Codes Library
- Sqlite3

2.1.3 Version Control

- Bazaar + Dropbox

2.2 Project structure

Our system will be basically focused on the gestures, postures and vocal behaviour cues. In order to analyse the behaviour of the user towards the audience we had defined a set of low and high level features. The low level features are those characteristics that are extracted directly from the Kinect SDK API, that is the RGB data, Depth and the raw audio. This system can be divided in three major modules: Data acquisition, Feature extraction and Learning, which are shown on figure 2.3. The first step is to capture the data that will be processed and further analysed. The kinect feeds in real time with information about colour, depth and sound. The colour or RGB data is used to perform the face detection and facial descriptions. The depth information is used to perform a skeleton tracking and to build a skeletal model. This model will yield the world coordinates of the user in real time. The kinect system defines 20 key points to determine a human skeleton, in our system we will focus only on the coordinates of hands, wrists, arms, elbow hip, shoulders and head. The depth information along with the RGB data is used by the kinect to provide facial description, which supplies up to 121 different points of the human face. Finally we extract the raw audio from the kinect (The next step is to extract the features from all these raw data provided by the kinect. We separate the process in two different parts: The extraction of low level features from the RGBD data which defines the face tracking system and the skeletal body model, and the processing of these low level features into high level features to build the characteristics that codifies the users behaviour. The final step is the make our system learn from this processed data by the means of different statistical classifiers.

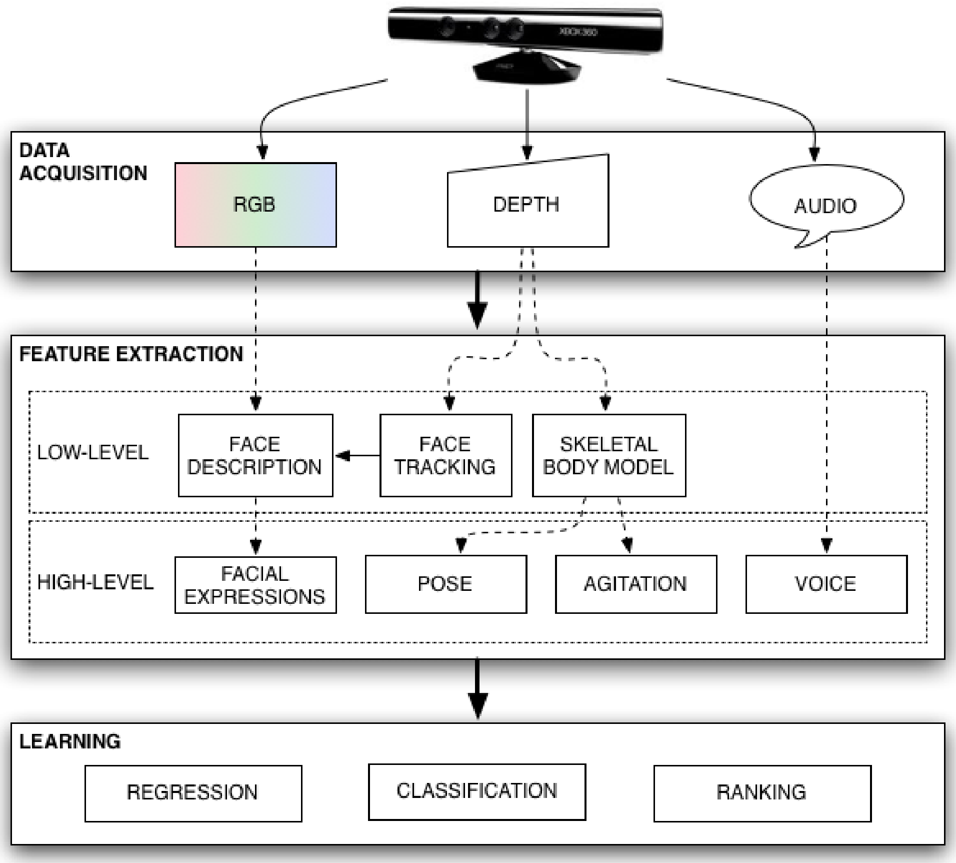


Figura 2.3: System structure

Moreover in figure 2.4 we can appreciate how the system will interact with the user. First in the data acquisition phase, the user will be recorded while making his/her presentation, then after the presentation is made the user can proceed with the feature extraction from the recent presentation. Once the feature vector is extracted comes the learning phase, where the system from the previous presentations and the new presentation will build a model that characterize the quality of the presentation. Note that the learning process would be applied until we obtain a high number of examples for our system, once this number is reached the learning process would be no longer available, since the system already learnt and it only has to predict the new presentations from the learnt model. Finally the user would be able to report all the results of the system's prediction

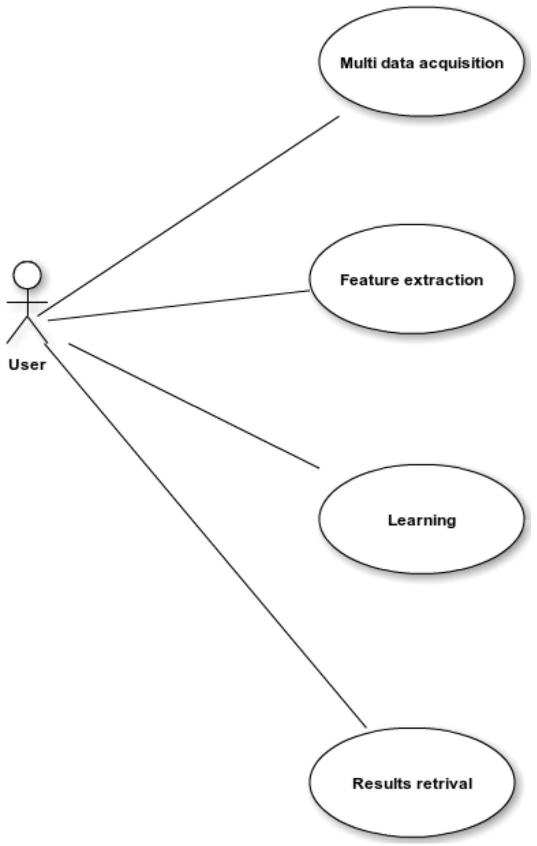
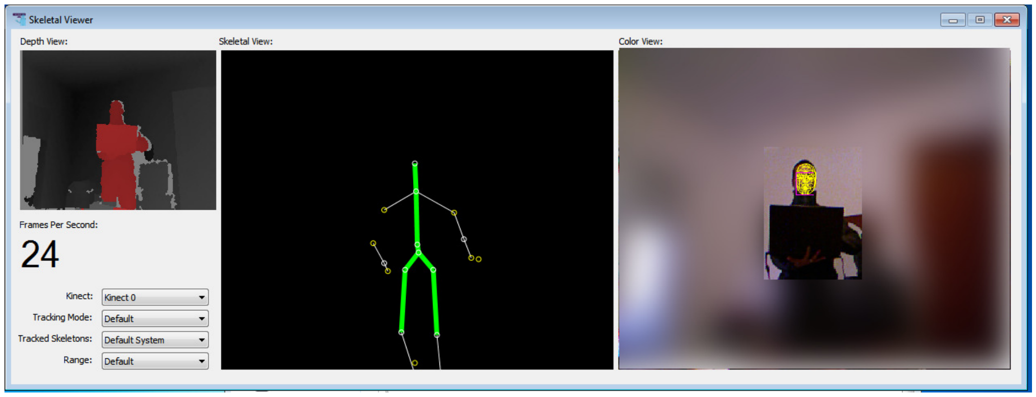


Figura 2.4: System structure



Frames Per Second: 24

Kinect: Kinect 0

Tracking Mode: Default

Tracked Skeletons: Default System

Range: Default

Figura 2.5: An screenshot from the application

2.2.1 Setting up the Kinect

As we stated earlier the first stage on this project is the data acquisition the first thing to do is to set up the kinect to record all the data that will be later processed. The first step then is to set up the kinect to record the skeletons and facial descriptors. The Skeletal Tracking is optimized to recognize users standing or sitting, and facing the Kinect; sideways poses provide some challenges regarding the part of the user that is not visible to the sensor. To be recognized, users simply need to be in front of the sensor, making sure the sensor can see their head and upper body; no specific pose or calibration action needs to be taken for a user to be tracked. Kinect field of view of the users is determined by the settings of the IR camera in default range mode, Kinect can see people standing between 0.8 meters (2.6 feet) and 4.0 meters (13.1 feet) away; users will have to be able to use their arms at that distance, suggesting a practical range of 1.2 to 3.5 meters. To this end we tried in every presentation to place the kinect camera in the same area as the tribunal, so the user will be facing the camera and never farther than 3 meters away.

2.2.2 The face tracking SDK

The Microsoft Face Tracking Software Development Kit for Kinect for Windows(Face Tracking SDK), together with the Kinect SDK, enables us to create applications that are able to track human faces. The Face Tracking SDK analyses input from the Kinect camera, and then deduces the head pose and facial expressions

The face tracking engine analyses input from a Kinect camera, deduces the head pose and facial expressions, and makes that information available to an application in real time. For example, this information can be used to render a tracked person's head position and facial expression on an avatar in a game or a communication application or to drive a natural user interface (NUI). This version of the Face Tracking SDK was designed to work with Kinect sensor so the Kinect for Windows SDK must be installed before.

Capítol 3

Design

In this section a detailed explanation of the implementation will be presented. Firstly we will present the different classes that has been designed in this project. We can appreciate clearly from figure 3.1 how the project is divided in two main blocks. The first block is in charge of dealing with the features vectors. The first block consist of three classes: ***ObjectStorage*** which will handle the task of saving and loading the feature vector into disk. The feature vectors, once they are calculated from the raw information provided by the kinect, due to its long processing time, these vector are save into local storage so we do not need to recalculate them each time. The second class ***FeatureVector*** is responsible of computing the high level features or meta characteristics. As we can see, this class contains all the necessary methods to extract the feature vector, such as *crossed arms*, *pointing*, *looking at the camera*, etc. Finally the DBWrapper class will be the layer between the application and the SQLite layer, including the necessary methods to extract the data from the database. The second block is related with the application that captures the data from the kinect. The main class named ***SkeletalViewer*** would initialize the kinect, and be listening for the events fired by the kinect whenever one kinect frame (colour, depth, skeleton) is ready. The class ***Visualize*** is in charge of displaying the data capture by the kinect into a 2D canvas in our application. Finally the ***Datarapper*** class is similar to ***DBWrapper*** explained earlier.

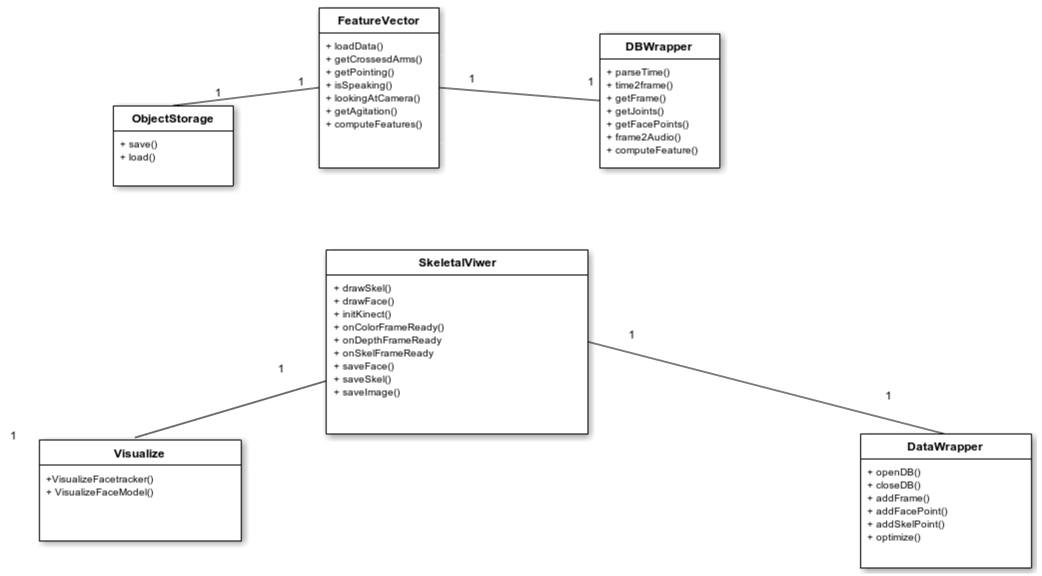


Figura 3.1: Class diagram

For this part we developed an application that collects the raw data from the kinect and then translates it into human pose and joints world coordinates. Converting from the RGB-D data into human pose is made inside the Microsoft Kinect SDK, as a black box, we only had to make use of the API provided by Microsoft which returns the world coordinate user's joint and facial descriptors. The first step is to set up the Kinect to record the data that we are interested in, to do so, we must tell to the kinect what data we want to obtain.

Kinect application development starts with the KinectSensor. This object directly represents the Kinect hardware. It is from the KinectSensor object that we access the data streams for video (colour) and depth images as well as skeleton tracking. The three main streams provided by the kinect are: the ColorImageStream, DepthImageStream and SkeletonStream The most common method of data retrieval from the sensors streams is from a set of events on the KinectSensor object. Each stream has an associated event, which fires when the stream has a frame of data available for processing. Each stream packages data in what is termed a frame. For example, the ColorFrameReady event fires when the ColorImageStream has new data. Each of the data streams (colour, depth and skeleton) return data points in different coordinates systems. First we must discover the Kinect sensor, once discovered the kinect must be initialized before it can begin producing data. First, the application must enable the streams it needs. Each stream has an Enabled method, which initializes the stream. Each stream is uniquely different and as such has settings that require configuring before enabled. The next step is determining how the application retrieves the data from the streams. The most common means is through a set of events on the KinectSensor object. There is an event for each stream (ColorFrameReady for the ColorImageStream, DepthFrameReady for the DepthImageStream, and SkeletonFrameReady for the SkeletonStream), and the AllFramesReady event, which synchronizes the frame data of all the streams so that all frames are available at once. It is worth mentioning that each event should be treated as a separate thread to work properly. Finally, the application must start the KinectSensor object by calling the Start method. Almost immediately after calling the Start method, the frame-ready events begin to fire. Although along this project we focus almost exclusively on the pose estimation based on the users world coordinates, we take advantage and save the different streams provided by the kinect, just in case we would need them in the future, so on each frame event we save to disk the raw stream to local storage (hard drive), this process might jeopardize the performance of our system by slowing down the frame rate, but as we did not want to loose any vital information that we later regret, we choose to keep them and compromise a little bit the overall performance of the system. During the presentations the frame rate average was about 14 frames per second, which was enough for our purposes.

3.0.3 Skeleton tracking

As one of our main goals was to capture the users postures, we needed to record frame by frame the Skeleton data comes from the Skeleton-Stream. Data from this stream is accessible either from events or by polling similiarily to the color and depth streams. The KinectSensor object has an event named SkeletonFrameReady, which fires each time new skeleton data becomes available. Skeleton data is also available from the AllFramesReady event. Each frame of the SkeletonStream produces a collection of Skeleton objects. Each Skeleton object contains data that describes location of skeleton and the skeletons joints. Each joint has an identity (head, shoulder, elbow, etc.) and a 3D vector.

3.0.4 Joints

Each Skeleton object has a property named Joints. This property is of type JointsCollection and contains a set of Joint structures that describe the trackable joints (head, hands, elbow and others) of a skeleton. An application references specific joints by using the indexer on the JointsCollection where the identifier is a value from the JointType enumeration. The JointsCollection is always fully populated and returns a Joint structure for any JointType even when there are no users in view.

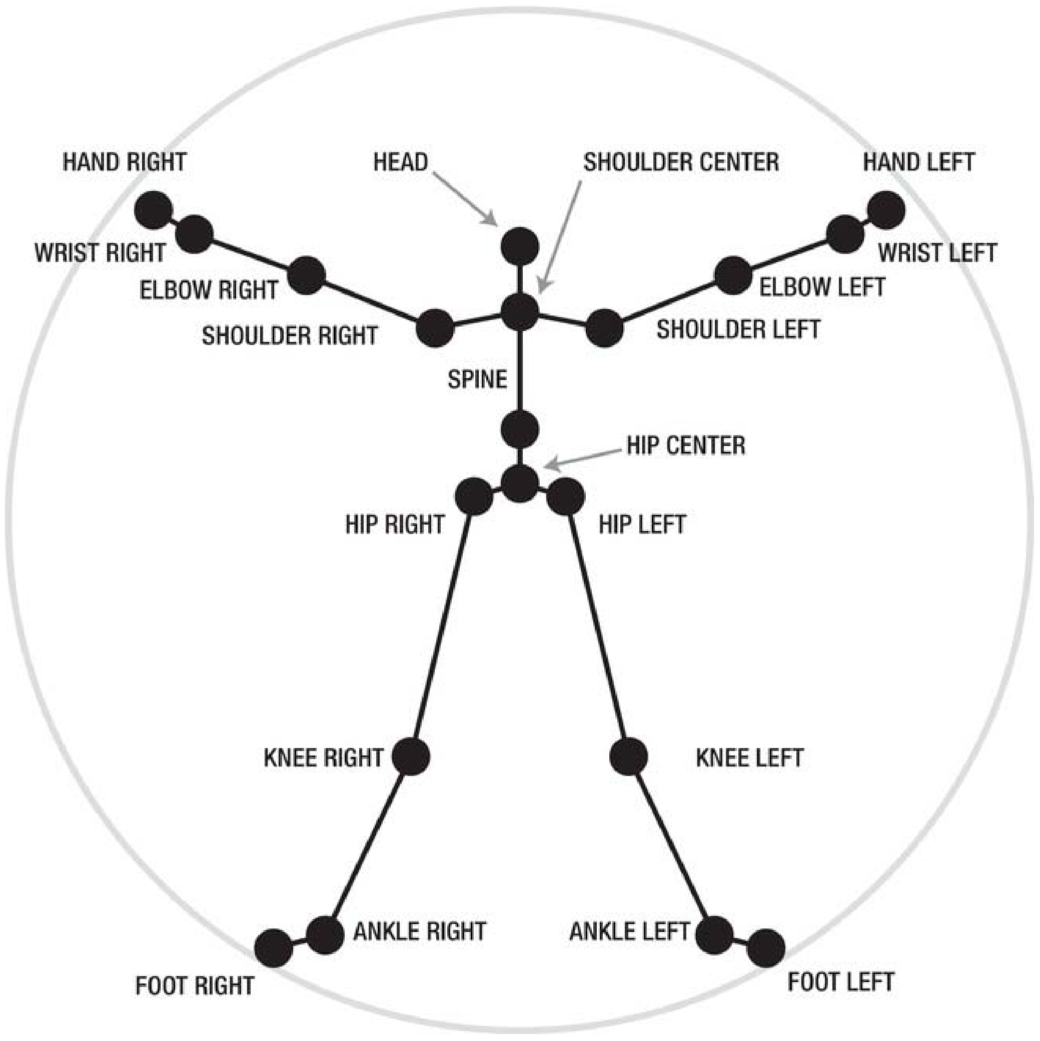


Figura 3.2: Skeleton key points

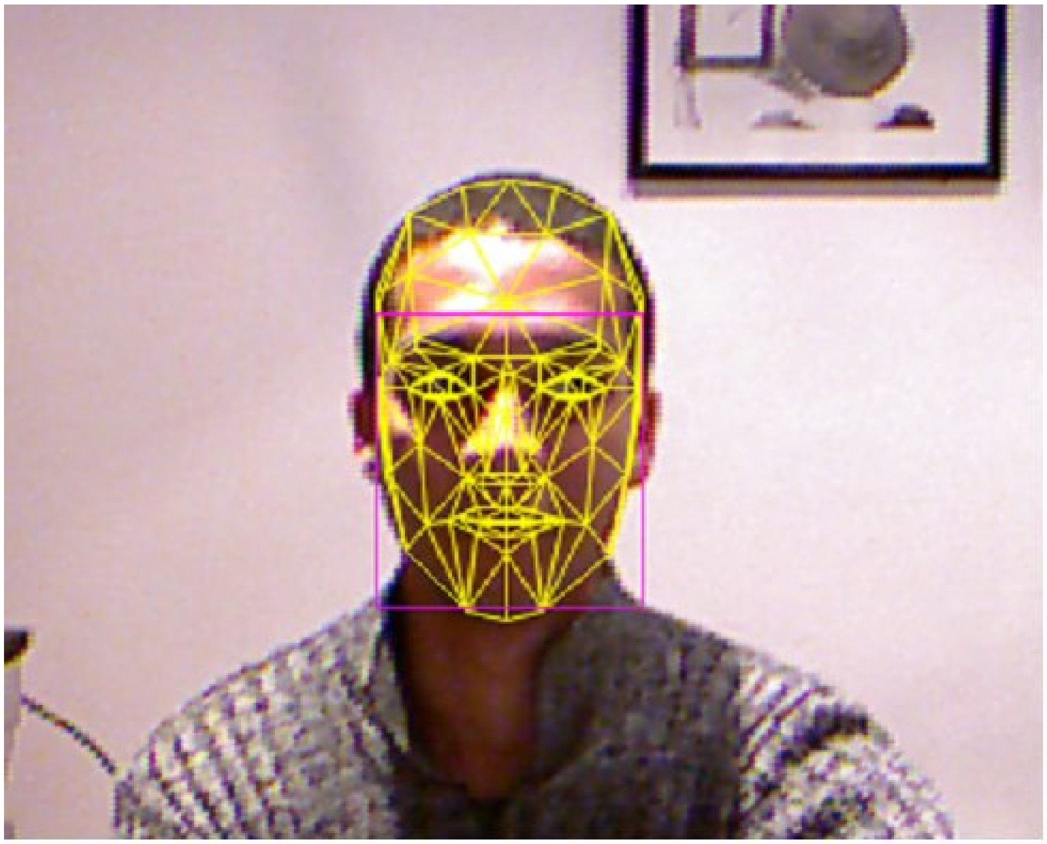


Figura 3.3: Face mask

As we presented earlier, we try to measure the user interaction using different types of data capture from the kinect, and we dedicate especial interest in the user body s world coordinate at a given time, so in each frame we save into the database the user s world coordinates. The figure[xx] shows the structure of this database. First we will save the 121 points of the user's face. Although we only compute whether the user is looking forward or not using the nose's normal vector we keep the remaining points, just in case we would need them later. The table in charge of saving the face point is called face_mask, and it is composed of six fields, three of them(x, y, z) used to indicate the world

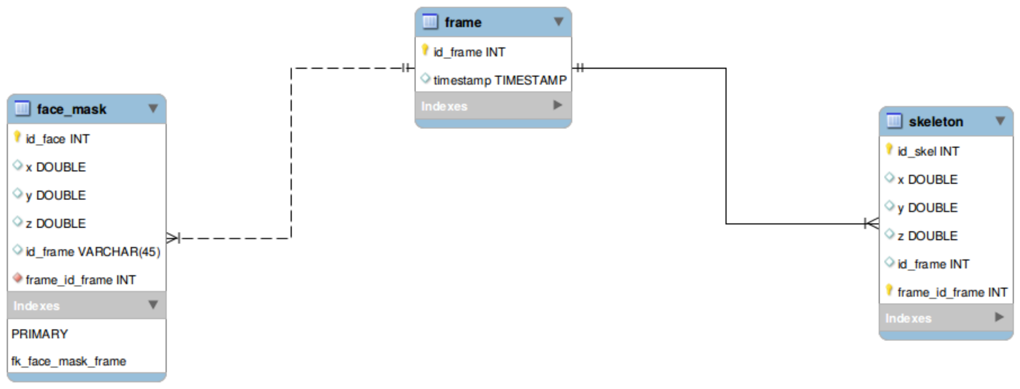


Figura 3.4: E-R Diagram

coordinate and id_frame, to represent in which frame it was saved. It is worth to say than in each frame we would have 121 entries in this table, one for each key point of the face defined by the Kinect SDK. The next table is just a relation table to store all the frames that we have stored so far, the name of this table is self-explanatory (frame), and it has only two fields, a unique number that identifies each frame and the timestamps (year , month, date, hour, minute, second and millisecond) in which the frame was stored. Finally there is the table that is charged to store the user's body coordinates, as the face_mask table we gave three fields used to indicate the world coordinate, a unique identifier and a field that relates each point with the frame it was stored. The process of storing into the database is performed in a new thread, so the application does not get affected heavily by the I/O operations that are held while writing into disk. Although using separate threads to store the data improved considerably the performance of the application, some more tweaks on the SQLite side were needed to improve even more the saving process. At the beginning, and with the default SQLite configuration, we were able to perform approximately 23 000 inserts per second which was good, but in our case every millisecond is valuable. Our first step was to use transactions when saving a skeleton or a face frame. By default SQLite will evaluate every INSERT or UPDATE statement within a unique transaction, this process is slower because it needs to load many resources each time an insert or update is carried out, to this end each time we had to save a skeleton frame or a face frame we wrapped the insert statements into a single transaction. Thanks to wrapping the whole insert process into a transaction, we double our performance, being able to store approximately 53 000 inserts per second. The next steps was to turn off the synchronous. By default SQLite will pause after issuing a OS-level write command. This guarantees that the data is written to the disk. By turning off the synchronous mode, we are instructing SQLite to simply transfer the data to the OS for writing and then continue. After this change up to 70 000 insert per seconds were possible. Finally we used an In-Memory database, so instead of working in the file, we are working entirely in RAM. The final results was an optimal insert rate of approximately 98 750 inserts per second!

3.0.5 Speech

The microphone array is the hidden gem of the Kinect sensor. The array is made up of four separate microphones spread out linearly at the bottom of the Kinect. By comparing when each microphone captures the same audio signal, the microphone array can be used to determine the direction from which the signal is coming. This technique can also be used to make the microphone array pay more attention to sound from one particular direction rather than another. Finally, algorithms can be applied to the audio streams captured from the microphone array in order to perform complex sound dampening effects to remove irrelevant background noise. All of this sophisticated interaction between Kinect hardware and Kinect SDK software allows speech commands to be used in a large room where the speakers lips are more than a few inches from the microphone.

When the Microsoft Kinect SDK is installed, the components required for speech recognition are automatically chain installed. The Kinect microphone array works on top of pre existing code libraries that have been around since Windows Vista. These pre existing components include the Voice Capture DirectX Media Object (DMO) and the Speech Recognition API (SAPI). The Voice Capture DMO is intended to provide an API for working with microphone arrays to provide functionality such as acoustic echo cancellation (AEC), automatic gain control (AGC), and noise suppression. This functionality can be found in the audio classes of the SDK. The Kinect SDK audio wrapper simplifies working with the Voice Capture DMO as well as optimizing DMO performance with the Kinect sensor. To implement speech recognition with the Kinect SDK, the following automatically installed libraries are required: the Speech Platform API, the Speech Platform SDK, and the Kinect for Windows Runtime Language Pack. The Speech Recognition API is simply the development library that allows us to develop against the built-in speech recognition capabilities of the operating system. It can be used with or without the Kinect SDK, for instance if we wanted to add speech commands to a standard desktop application that uses a microphone other than the Kinect microphone array. The Kinect for Windows Runtime Language Pack, on the other hand, is a special set of linguistic models used for interoperability between the Kinect SDK and SAPI components. Just as Kinect skeleton recognition required massive computational modelling to provide decision trees to interpret joint positions, the SAPI library requires complex modelling to aid in the interpretation of language patterns as they are received by the Kinect microphone array.

3.0.6 Capturing the data

The RGB camera, operating at 30 Hz, can push images at 640x512 pixels. The Kinect also has the option to switch the camera to high resolution, running at 15 frames per second (fps), which in reality is more like 10 fps at 1280x1024 pixels. Of course, the former is reduced slightly to match the depth camera. Firstly, when a new image frame is detected an event is triggered, and captured in a function, in our case in Nui_GotColorAlert. The next step is to capture the image stream with the function NuiImageStreamGetNextFrame which gets the next frame of data from the specified image stream. Once we have the frame image stream into memory, we proceed to first, store this image into disk the image will be saved as a 640 x 480 jpg image, then we draw the image into our application, and finally we must release the stream frame with the function NuiImageStreamReleaseFrame to avoid memory issues. The process of extracting the depth information is similar to the RGB extraction with a slight difference, which consist of separating the player index from the depth image and saving the raw data (including the player index information). The SDK assigns a number to each tracked player. The number or player index is stored in the first three bits of the depth pixel data. Each pixel is 16 bits. Bits 0 to 2 hold the player index value, and bits 3 to 15 hold the depth value. A bit mask of 7 (0000 0111) gets the player index from the depth value. The Kinect SDK defines a pair of constants focused on the player index bits. They are DepthImageFrame.PlayerIndexBitmaskWidth and DepthImageFrame.PlayerIndexBitmask. The value of the former is 3 and the latter is 7. First we separate the player index from the depth image, then the depth image is shown in our application, and the whole information (depth image and player index) is store into disk.

3.0.7 Feature extraction

Once we have all the data saved in local storage, the next step is to process all this data. So far we have all the information at our disposal, so we need to translate it into something meaningful. To this end we had define a set of meta characteristics, or put in other words, data that defines another data, in this case we are trying to transform our raw data into body language descriptors.

Before focusing on the meta characteristics, we will present first how the low level features (or raw data) are extracted. As we said earlier the world coordinates are saved into local storage using the SQLite library. As all the information regarding the users pose in a given time is storage on these databases, we need to extract all the information frame by frame.

As we stated earlier the kinect returns 20 points that defines key points of the human body. Each point is saved into the database with the frame number that it was saved, thus all the 20 points of the user in a given time will have the same frame numbers.

```
query =  SELECT id_skel , x,y,z,id_frame
                FROM skeleton
                WHERE cond_frame
                    AND (SELECT COUNT(*)
                    FROM skeleton
                    GROUP BY id_frame) = 20
```

The nine features would be detailed below:

1. **Facing towards**: Average of frames that the user is looking at the tribunal/public. To analyse whether the user is looking at the tribunal or not we use the implementation of the face tracking system provided by the Microsoft SDK, then using the nose coordinate it is computed the noses vector direction. We consider that the user is looking at the public if the angle formed between the nose and within an approximately 30 degrees range.

2. **Crossed arms**: Average of frames that the user is with his/her arms crossed. In order to know if the user is crossing hands and considering the user is always facing the tribunal. To determine if arms are crossed the x coordinate of the right hand must be

lower than the x coordinate of the spine, besides the x coordinate of the left hand must be greater than the spine and finally the difference between the right hand x coordinate and the left hand x coordinate must be greater than the half of the forearms length.

```
if hand_right.x <= hip_center.x and
   hand_right.x < hand_left and
   hand_left.x >= hip_center.x and
 (hand_left.x - hand_right.x) >= cross_thr:
            return True
        return False
```

Where cross_thr is a constant defined as the half of the user's arm length.

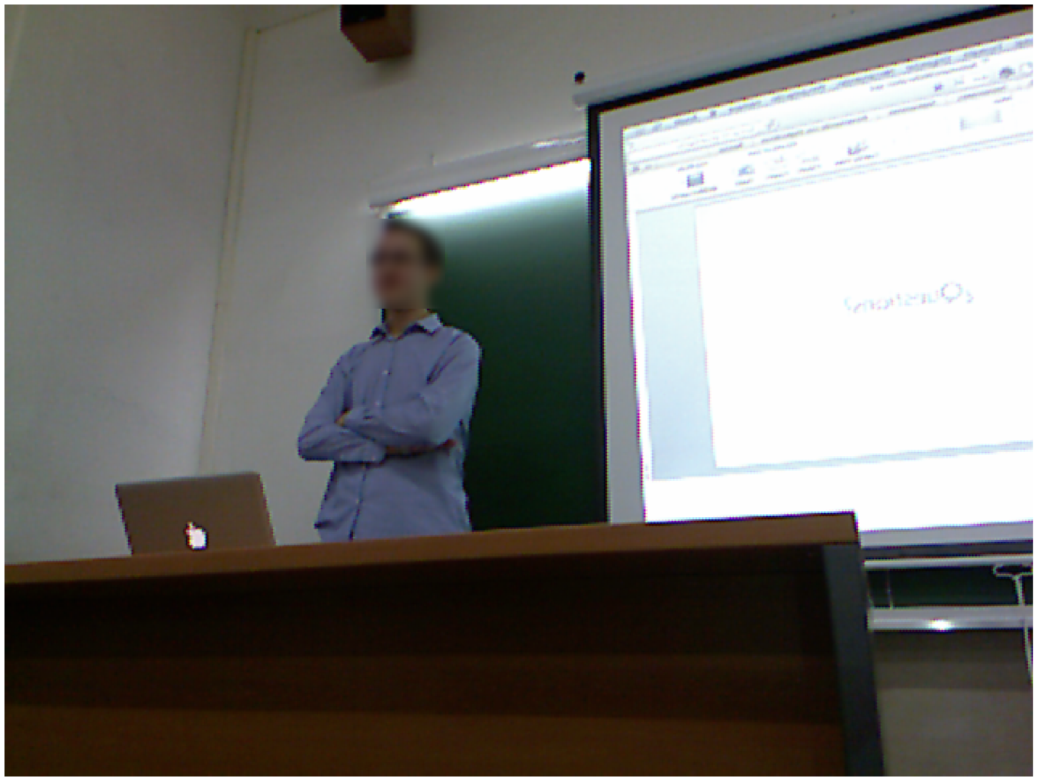


Figura 3.5: A user with his arms crossed

3. **Pointing**: Average time that the user is pointing towards the blackboard. To know whether the user is pointing or not, firstly we discard those situations where the hand is closer to the body than the elbow, then the distance between the hand and the hip is computed and then this distance is divided by the forearms length, then in order to avoid situations where the user seems to be pointing to the tribunal, we divide this distance by the difference in z-axis of both hand and hip, and finally we normalize by finding the inverse of this division. After some experiments we find out that values ranging from 0.0039 to 1 indicates that the user is pointing towards the blackboard.

```
weight = 0
hand2hip = calc_distance(hip_center, hand)
if hand2hip != 0:
  dist_weight = abs(hand2hip / arm_length)
  z_dist = abs(hand.z - hip_center.z)
  weight = 1 / (dist_weight / z_dist)
if weight < 0:
  weight = 0
return weight
  weight = 0
  if not hands_position['left_hand']:
    return weight
  hand = hand_left
  elbow = elbow_left
  if elbow.x < hand.x:
        return weight
  if arm_length_left is None:
        arm_length_left = calc_distance(elbow,
                                          hand)
  if hand.x < hip_center.x:
        weight = pointingHand(hand, hip_center
                          ,arm_length_left)
  if weight < 0:
        weight = 0
  return weight
```

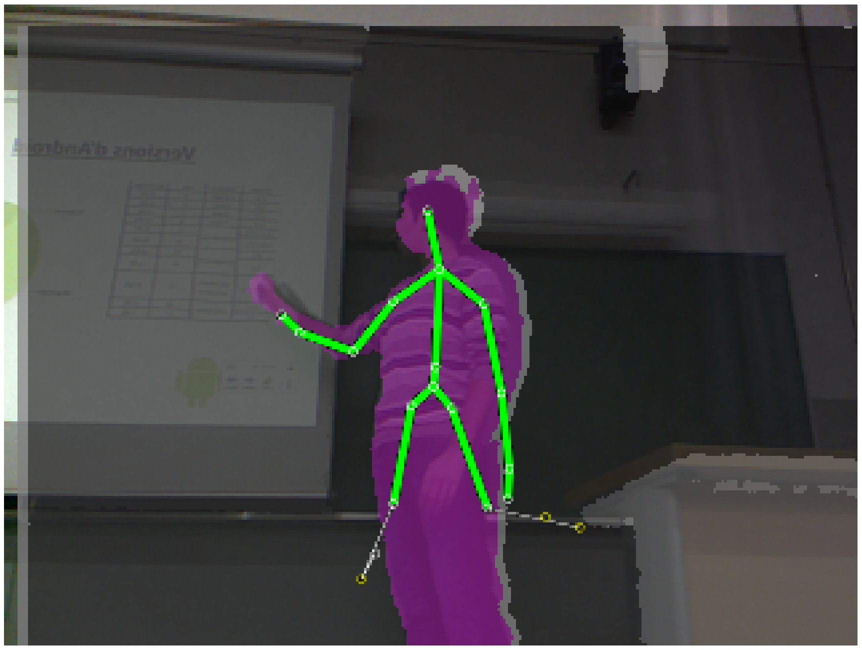


Figura 3.6: A user pointing to the blackboard

4. **Speaking**:Average time that the user is speaking. In order to compute this feature it has been used the implementation of the VAD 3 algorithm [4]. The foundations of this algorithm is based on three different features per frame. The first feature is the widely used short-term energy(E). Energy is the most common feature for speech/silence detection. The second feature is Spectral Flatness Measure(SFM).Spectral Flatness is a measure of the noisiness of spectrum. The third feature is the most dominant frequency component of the speech frame spectrum,which can be very useful in discriminating between speech and silence frames. Finally the three of these features are applied in parallel to detect the voice activity. The proposed VAD algorithm starts with framing the audio signal. First N frames are used for threshold initialization. For each incoming speech frame the three features are computed. The audio frame is marked as a

speech frame, if more than one feature of the feature values fall over the precomputed threshold.

5. **Upper agitation:** Average of the magnitude of arms, wrist and hands while hands are above the head. Namely if the users left hand or right hand is above his/her head then the magnitude is computed as the difference between frames of the distance from the wrist, hand or arm point to the hip point(taken as a reference point).

6. **Middle agitation:** Average of the magnitude of arms, wrist and hands while hands are below the head and above the hip. Namely if the users left hand or right hand is between his/her head and his/her hip then the magnitude is computed as the difference between frames of the distance from the wrist, hand or arm point to the hip point(taken as a reference point).

7. **Bottom agitation:** Average of the magnitude of arms, wrist and hands while hands are below the hip. Namely if the users left hand or right hand is below his/her hip then the magnitude is computed as the difference between frames of the distance from the wrist, hand or arm point to the hip point(taken as a reference point).

8. **Agitation while speaking:** Average of the magnitude of arms, wrist and hands while the user is speaking. To match exactly which speech frame corresponds to the image frame, we had to interpolate the sound data in order to make it the same length as the video data.

9. **Agitation while not speaking:**Average of the magnitude of arms, wrist and hands while the user is not speaking.

Magnitude of arms: Is computed as the euclidean distance between the each point and the hip, which is used as a reference point, and to know how much the user moved from one frame to the next is computing the difference between points.

Capítol 4

Results

In order to present the results, we first describe the data, settings, and evaluation measurements of the performed experiments.

4.0.8 Data

The analyzed data consists on 36 recorded videos of 13 Bachelor's Thesis presentations and 10 presentations from an 8th semester Bachelor's class at Universitat de Barcelona and 12 presentation from a master course. All the videos were recorded with a Kinect^{TM} device at 14 FPS. The videos were recorded on three different classrooms. All the videos were recorded with the user facing the tribunal. The details of the data set are shown in Table 4.1. For each presentation the vector of nine high-level communication features is computed and the score assigned by the teacher regarding the presentation quality is stored as the ground truth. Some examples of the recorded scenarios are shown in Figure 4.2.

Taula 4.1: Details of the data set

# videos	Final thesis	Class projects	Master course	Total frames
36	13	11	12	195700

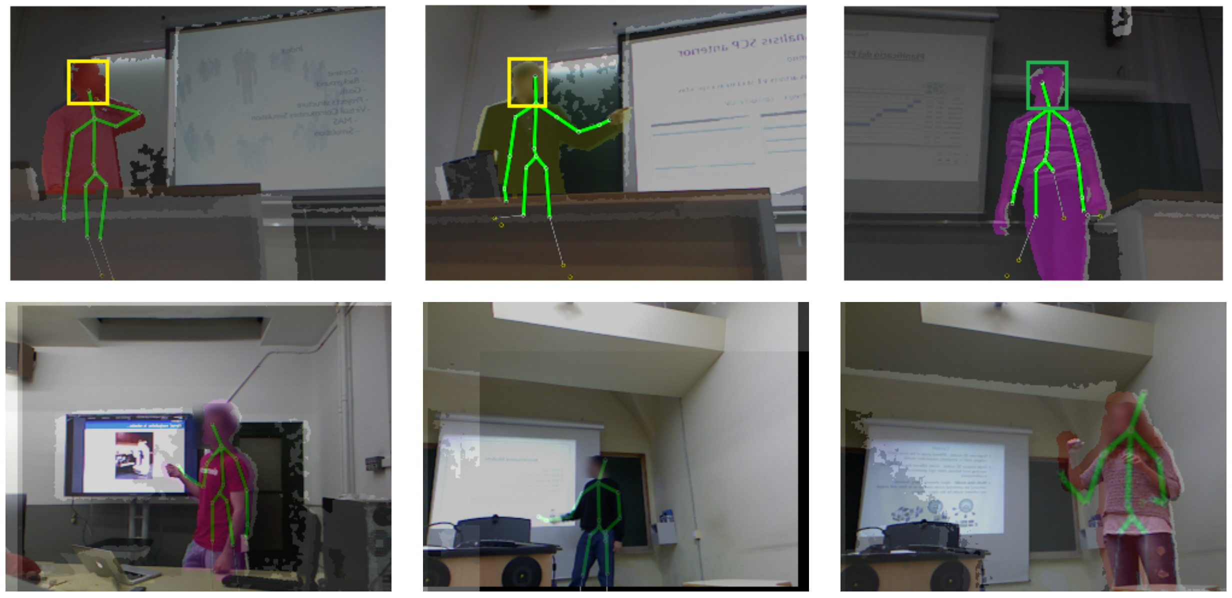


Figura 4.1: Examples of low-level features extraction. Depth and RGB aligned maps are superimposed with transparency. Color in the human body indicates user detection at pixel level using Random Forest. In this case, different colors indicate different user identifiers. Detected skeletons are drawn in green. Detected faces are also marked. Yellow color of faces indicates that the speaker is not looking at the tribunal/public and green marked face indicates that the speaker is facing towards the tribunal.

4.0.9 Settings

In order to train the the multi-modal features to be able to classify the quality of the presentations we use different classifiers. Specifically we selected Gentle Adaboost classifier Friedman et al. (n.d.) with decision stumps, Support Vector Machines with Radial Basis Function kernel (binary and one-versus-one multi-class) from *LibSVM* Chang & Lin (2011) and Ranking Support Vector Machines Joachims (2006). Adaboost is used in two ways, first to obtain a classifier which is able to separate between two differentiated groups: "good"presentations and "bad"presentations, and also as a feature selection method in order to analyse the relevance of each high-level communication indicator. We also analysed the weight assigned to the features in the case of SVM to analyse the most relevant indicators by this classifier. Moreover, SVM classifier is tested in three scenarios: binary classification, multi-class classification, ranking and regression.

4.0.10 SVM

SVM: Stands for Support Vector Machine, is a popular classification technique and consist of supervised learning models with associated learning algorithms that analyse data and recognize patterns, used for classification and regression analysis. There are several implementations of SVM algorithms. LIBSVM Chang & Lin (2011) is an integrated software for support vector classification, (C-SVC, nu-SVC), regression (epsilon-SVR, nu-SVR) and distribution estimation (one-class SVM). It supports multi-class classification, binary classification and regression. In this work we will be using the three classifications methods. SVM is considered relatively "easy"to use, although it is quite difficult to find the best parameters, so it is common that beginners who are not familiar with it often get unsatisfactory results, so, in order to use this method it is crucial to find the right set of parameters. To this end there are some tools that help us to find these parameters. First we must choose a kernel, in our case we choose RBF kernel, which is known to perform as good as linear kernel with few data samples and tends to improve when the data set grows.

There are two parameters for an RBF kernel: C and Y. It is not known beforehand which C and are best for a given problem; consequently some kind of model selection (parameter search) must be done. The goal is to identify good (C;) so that the classifier can accurately predict unknown data. To find the parameter a grid-search on C is highly recommended using cross-validation. Comparing the results between using random numbers of C and numbers obtained after a grid-search the results were improved upon a 50The best parameters found performing the grid-search are shown in Table 4.2.

4.0.11 Adaboost

short for Adaptive Boosting, is a machine learning algorithm. AdaBoost is adaptive and sensitive to noisy data and outliers, being especially useful in binary classification problems. Adaboost is used in this project within the Error Correcting Output Codes (ECOC) framework Escalera et al. (n.d.), specifically the BoostROC implementation. The BoostROC only needs that data separated int two well defined classes(for binary classification) and how many iterations we want to perform.

Taula 4.2: Methods and parameters

Method	No. Examples	Parameters
Adaboost	36	No. of max. iterations: 50
LibSVM	36	C=8 ; γ=0.5; Radial Basis Function
SVR	36	C=8192 ; γ=0.125; Radial Basis Function

4.0.12 Validation and measurements

In order to measure the generalization capability of our system we perform a leave-one-out validation model: a single observation (presentation) from the original sample is taken as the test data, and the remaining observations as the training data. This process is repeated as many times as observations we have, and the average number of hits is stored. This measurement is applied for binary and multi-class classification for different degrees of quality defined for the presentations.

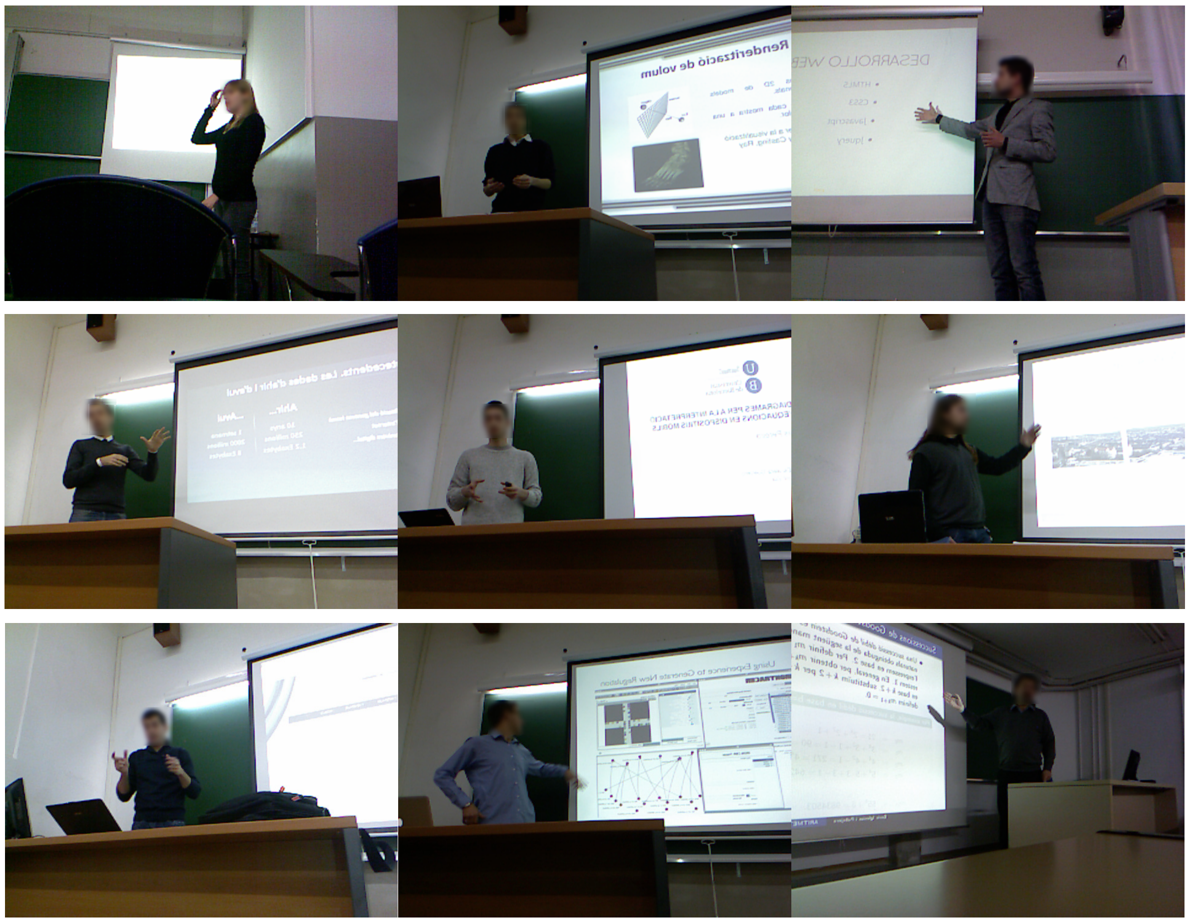


Figura 4.2: Some examples of the presentations of our data set.

In the case of ranking SVM, the error in the prediction is calculated as the ratio between by how many positions did the classifier failed predicting the correct position and the maximum number of displacement errors (prediction error). This metric is detailed in the ranking experiment section.

4.0.13 Experiments

In order to validate the proposed system, we perform five analysis: a) binary classification into "high quality" and l·low quality" presentations, b) multi-class classification into three and four categories of quality, c) analysis of feature selection relevance and classification with different feature subsets, d) raking of presentations based on quality, and e)regression analysis to determine the relationship between the different features and the final grade.

Binary classification

In order to train our model in the binary classification problem we consider a good presentation if its grade (in a scale from 6 to 10, being 10 the greatest grade possible) is greater or equal than 8.0, anything lower than 8.0 is considered as a l·low quality" presentation. First two rows of Table 4.3 show the results for binary classification using Adaboost and SVM, respectively. Best result is bolded. One can see the high recognition rate of both approaches, automatically splitting the presentations in two quality categories. In the case of SVM, the achieved accuracy is upon 80%.

Multi-class classification

In order to increase the set of possible quality categories, we designed the experiment in the multi-class case with three and four qualification groups. For the case of three groups we defined different ranges to split the presentations. The ranges of scores used to defined the three and for quality categories (namely bad, average, good, and excellent) are shown in Table 4.3. The results applying multi-class one-versus-one SVM in this case are shown in the same table. Best results are bolded. One can see that although the performance is decreased because of the increment in the number of categories, we are able to correlate in a

Taula 4.3: Binary and multi-class classification results

# classes	Method	Accuracy	'Bad'	'Average'	'Good'	'Excellent'
2	Adaboost	**75%**	6.0 - 7.9	-	8.0 - 10	-
2	SVM	**80.5%**	6.0 - 7.9	-	8.0 - 10	-
3	SVM	50%	6.0 - 7.4	7.5 - 8.4	8.5 - 10	-
3	SVM	**63.88%**	6.0 - 7.9	8.0 - 8.9	9.0 - 10	-
4	SVM	50%	6.0 - 6.9	7.0 - 7.9	8.0 - 8.9	9.0 - 10

percentage of 63.8% with the opinions of the teachers for the case of three and four qualification categories respectively.

Feature selection and relevance

We also perform feature selection and weight analysis on the high-level features selected by Adaboost and SVM classifiers in order to analyse their relevance for discriminating among groups of presentations. We focused on the binary classification problem, and for all the iterations of the leave-one-out evaluation we save the alpha weight value assigned for the selected features by Adaboost and the weights assigned by SVM. In the case of SVM the F-score method is used as a feature selection technique. F-score wei Chen (2005) is a simple technique which measures the discrimination of two sets of real numbers. These values are normalized in order to compute the percentage of relevance of each feature in relation to the rest for each classifier. Results are summarized in Table 4.4. For each classifier the four features selected with the highest score are bolded. One can see that both classifiers correlate in the relevance of different features. In particular, 'Facing towards' and 'Poiting' is selected with high scores by both classifiers. Additionally, Adaboost gives high scores to the 'Upper agitation' and 'Crossed arms', whereas SVM also assigns as relevant the 'Middle agitation' and 'Agitation while speaking' indicators. Thus, agitation indicators become relevant for both learning strategies.

Finally, in order to analyse the generalization capability of the most relevant features, we reproduced the binary classification of presentations with subsets of high-level features. Figure 4.4 shows the results. The first bars corresponds to the previous results using the complete set of nine high-level behavioural indicators. Second and last sets of bars shows the classification results of the leave-one-out experiments when classifiers only consider the subsets of four and two most relevant features based on the analysis shown in Table 4.4. Note that although the performance is reduced because of the use of a reduced feature set, we are able to correlate upon 70% of the times with the teaching scores only considering the four most discriminate features.

Taula 4.4: Percentage of relevance assigned to the high-level features by Adaboost and SVM classifiers

Feature	Meaning	Adaboost	SVM-RBF
1	Facing towards	**16.095%**	**35.020%**
2	Crossed arms	**12.99%**	1.80%
3	Pointing	**20.87%**	**25.26%**
4	Speaking	7.04%	6.14%
5	Upper agitation	**24.89%**	0.092%
6	Middle agitation	0.94%	**10.73%**
7	Bottom agitation	9.86%	6.26%
8	Agitation while speaking	6.78%	**14.13%**
9	Agitation while not speaking	0.49%	0.53%

Ranking

The goal of Rank SVM Joachims (2006) is to predict multivariate or structured outputs. In this case, we use the ground truth grade value of each presentation to generate pairwise preference constraints based on

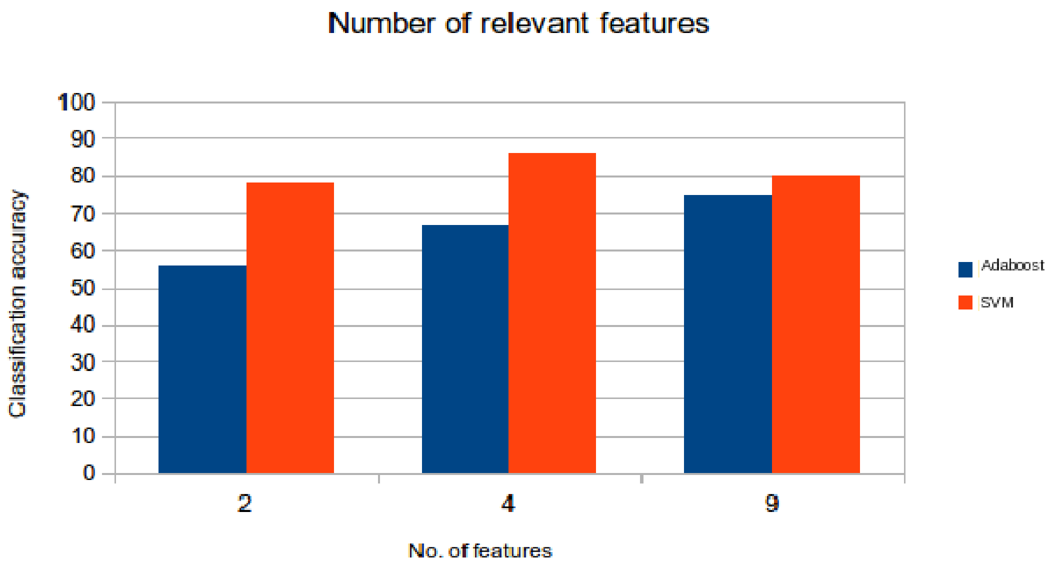


Figura 4.3: Binary classification with different feature subsets.

a training set ordered by quality of presentation in descendent order. For this experiment, we defined different number of splits of our data, namely 3, 5, and 7 fold cross-validation, so that different number of test samples are ordered. In this case, we compute the recognition error E_ϵ as the ratio in percentage between by how many positions did the classifier failed predicting the correct position and the maximum number of displacement errors, defined as follows:

$$E_\epsilon = \frac{m}{2(\sum_{i=0}^{n/2-1} N - (2i+1)) - N + n} \cdot 100,$$

where m is the number of missed positions, N is total of test samples at each iteration of a K-fold experiment, and n is the number of different scores within the test samples. Then, the classification performance ζ is defined as $\zeta = 100 - E_\epsilon$. The results of this experiments are shown in Table 4.5. One can see that for different number of $K \in \{2, 3, 5\}$, corresponding to rank at each iteration of the fold 12, 8, and 5 test samples respectively, we achieve high recognition rates, approximately in the range of 70%-80% of performance.

Taula 4.5: Ranking of presentation results

2-fold		3-fold		5-fold	
E_ϵ	ζ	E_ϵ	ζ	E_ϵ	ζ
29%	71%	18%	82%	8%	92%

Regression for grade prediction

Finally, we performed a regression analysis to estimate the relationships among variables, in our case between a dependent variable (the grade) and the independent variables(the feature vector). The estimation target is a function of the independent variables used to predict a grade. The results are shown in Figure 4.4. For each presentation we show the ground truth score and the automatically computed one by means of regression. The diference is marked with a green line. Note the high correlation among the estimations and the real score. In this case, the mean error of the leave-of-out regression evaluation is of only 0.79 points, with an standard deviation of 0.53.

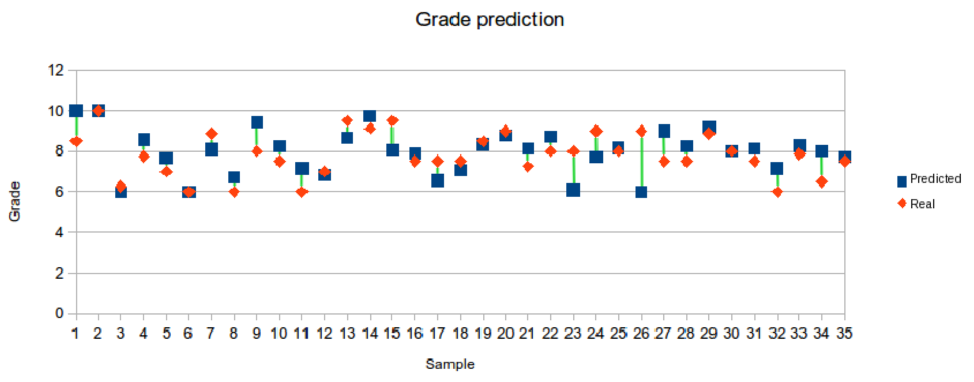


Figura 4.4: Regression target function for grade prediction

Capítol 5

Conclusion and future work

We presented an automatic system for categorization of presentations as a e-Learning tool for evaluating the non-verbal communication competence. We performed multi-modal human behaviour analysis from RGB, depth, and audio data and defined a set of high-level behaviour indicators. We recorded a novel data set of oral presentations, and based on the score defined by the experts (teachers in our case) we trained binary, multi-class, and ranking classifiers to evaluate the performance of our system. We analysed the most discriminative features that correlate to the observers opinion, and achieved classification rates upon 90% categorizing two levels of presentation quality, and upon 80% and 70% classifying the quality of the presentations in three and four groups respectively. The results of this work show the feasibility of our system to be applied as an automatic tool useful for user feedback in training scenarios, as well as for evaluation purposes. Given the reliability of our system, as a future work we plan to increase the amount of behavioural patterns including temporal constraints, more precise facial expressions, as well as to extend the number of samples so that a more precise score for each presentation could be automatically assigned. Finally, we plan to apply the methodology in real scenarios to define a useful protocol for user feedback and include the framework as a e-Learning tool in the training routine of non-verbal communication-related competences.

Bibliografia

Chang, C.-C., & Lin, C.-J. (2011). Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *2*(3), 27.

Escalera, S., Pujol, O., & Radeva, P. (n.d.). *Error-correcting ouput codes library.*

Escalera, S., Pujol, O., Radeva, P., Vitria, J., & Anguera, M. T. (2010). Automatic detection of dominance and expected interest. *EURASIP Journal on Advances in Signal Processing*, *2010*, 39.

Friedman, J., Hastie, T., & Tibshirani, R. (n.d.). Additive logistic regression: a statistical view of boosting, 1998. *URL citeseer. ist. psu. edu/friedman98additive. html*, *7*(7.1).

Joachims, T. (2006). Training linear svms in linear time. In *Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining* (pp. 217–226).

Marcos-Ramiro, A., Pizarro-Perez, D., Marron-Romera, M., Nguyen, L., & Gatica-Perez, D. (n.d.). Body communicative cue extraction for conversational analysis.

McCowan, L., Gatica-Perez, D., Bengio, S., Lathoud, G., Barnard, M., & Zhang, D. (2005). Automatic analysis of multimodal group actions in meetings. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, *27*(3), 305–317.

mlpy. (2013, June). Retrieved from `http://mlpy.sourceforge.net/`

Mohammadi, G., & Vinciarelli, A. (2012). Automatic personality perception: Prediction of trait attribution based on prosodic features. *Affective Computing, IEEE Transactions on*, *3*(3), 273-284. doi: 10.1109/T-AFFC.2012.5

Olguín, D. O., Waber, B. N., Kim, T., Mohan, A., Ara, K., & Pentland, A. (2009). Sensible organizations: Technology and methodology for automatically measuring organizational behavior. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, *39*(1), 43–55.

Pan, W., Dong, W., Cebrian, M., Kim, T., Fowler, J. H., & Pentland, A. (2012). Modeling dynamical influence in human interaction: Using data to make better inferences about influence within social systems. *Signal Processing Magazine, IEEE*, *29*(2), 77–86.

Sanchez-Cortes, D., Aran, O., Jayagopi, D. B., Mast, M. S., & Gatica-Perez, D. (2012). Emergent leaders through looking and speaking: from audio-visual data to multimodal recognition. *Journal on Multimodal User Interfaces*, 1–15.

scikit-learn. (2013, June). Retrieved from `http://scikit-learn.org/stable/`

Tanaka, H., Sakti, S., Neubig, G., Toda, T., Campbell, N., & Nakamura, S. (2012). Non-verbal cognitive skills and autistic conditions: An analysis and training tool. In *Cognitive infocommunications (coginfocom), 2012 ieee 3rd international conference on* (p. 41-46).

Vinciarelli, A., Pantic, M., & Bourlard, H. (2009, November). Social signal processing: Survey of an emerging domain. *Image Vision Comput.*, *27*(12), 1743–1759.

Vinciarelli, A., Salamin, H., & Pantic, M. (2009). Social signal processing: Understanding social interactions through nonverbal behavior analysis. In *Computer vision and pattern recognition workshops, 2009. cvpr workshops 2009. ieee computer society conference on* (pp. 42–49).

wei Chen, Y. (2005). Combining svms with various feature selection strategies. In *Taiwan university.* Springer-Verlag.