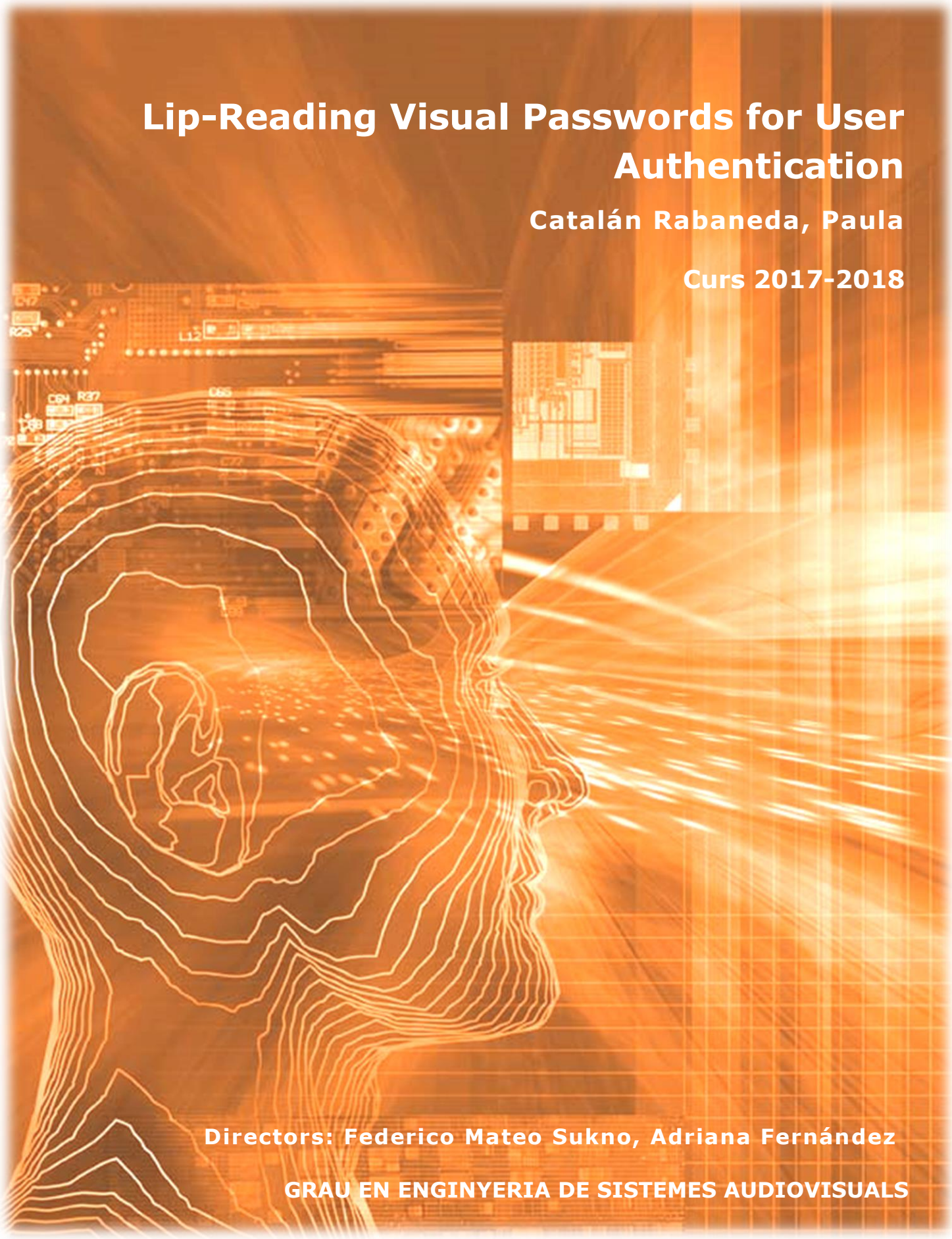
Lip-Reading Visual Passwords for User Authentication

Catalán Rabaneda, Paula

Curs 2017-2018



Directors: Federico Mateo Sukno, Adriana Fernández

GRAU EN ENGINYERIA DE SISTEMES AUDIOVISUALS

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Lip-Reading Visual Passwords for User Authentication

Paula Catalán Rabaneda

TREBALL DE FINAL DE GRAU UPF / ANY 2017-18

TUTORS DEL TREBALL

Federico Mateo Sukno, Adriana Fernández,
Cognitive Media Technologies Research Group

A la meva família

Acknowledgements

First of all, I would like to express my gratitude to Adriana Fernández and Federico Sukno for their guidance during the course of the project and for teaching an unknown field to me, without their knowledge and determination this project could not have been possible. Also, I would like to give credit to all the teachers that have cared for my educational career and shared their passion since primary schooling.

Thanks a lot to the friends I have shared library, class and Gutenberg square hours with, thanks for your support during these 4 years of continuous overcoming, for every coffee and beer.

I wish to thank the support of my best friends and, lucky for me, also flatmates, during this experience in Barcelona, I'm impressed of how we have changed and matured at an individual level but strengthened our relationship day after day.

Lastly, I would like to say to my family that I love you and thanks for your unconditional help, support and love. I have a wonderful family and I feel it in every call and every weekend that we have spent together.

Abstract

The important role that technologies have in our day-to-day life makes us aware that security is in the spotlight. Biometric-based authentication methods provide a safer alternative to traditional PIN passwords since they evaluate who you are and not what you know. We propose a visual password for user authentication based on automatic lip-reading. The result of a lip-reading system is a double password, in one hand, the word detection, and in the other, the speaker recognition. The approach we develop in order to achieve this double recognition is based on Hidden Markov Models and Gaussian Mixture Models, the features chosen as the input of the system are Discrete Cosine Transform coefficients of a spoken word video frames. The results obtained in this project show an approximate 15% of general recognition error. However, most of this error comes from word recognition, which results to be suboptimal, while speaker recognition is successfully achieved with errors below 4.23%.

Resum

Degut al paper que juguen les tecnologies en el nostre dia a dia, cada cop estem més conscienciats pel què fa la seguretat en les nostres dades personals. Això ens porta a la cerca de mètodes d'autentificació d'usuaris alternatius i més segurs a les tradicionals contrasenyes PIN, com l'autentificació basada en paràmetres biomètrics, que avaluen qui és l'usuari enlloc de què sap. En aquest projecte es proposa un a contrasenya visual per a la identificació d'usuaris basada en la lectura de llavis automàtica. El resultat d'aquest tipus de contrasenya visual és una doble clau d'accés, d'una banda, la detecció de paraula, i per l'altra, el reconeixement d'usuari. El sistema que es proposa està basat en Models Ocults de Markov i Models de Mescles Gaussianes, i com a característiques dels vídeos dels usuaris i paraules, entrada al sistema, els coeficients de la Transformada Discreta de Fourier dels diferents fotogrames. Els resultats obtinguts revelen un error en general del 15%. Tot i així, la gran majoria d'aquest error és deguda al reconeixement de paraula, ja que s'ha mostrat insuficient, mentre que el reconeixement d'usuari ha demostrat ser molt eficaç amb un error del 4.23%.

Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

User authentication technologies are the ones in charge of providing security to our personal data and access to our virtual identity. These uses are reason enough to place value and attention to these technologies. Recently, several systems have integrated novel technologies such as iPhone X which can be accessed by using Face ID, a Face Recognition system applied to authenticate users. Likewise, nearly every smartphone that we carry in our pockets has a fingerprint system to recognise its owner. With these two types of authentication systems we obtain answer to the question *Who are you?*, a more valuable information than the answer to *What do you know?* from the traditional passwords and PIN numbers.

The evolution on user authentication is a natural result of technology and society′s growth. Nowadays, society demands amenities and speed as well as security. At this point, an important role is played by biometric-based passwords, such as recognition of face, iris or fingerprints. While the latter option is more widespread, its disadvantages are the need of a specific device to accomplish the recognition and its vulnerability, because they can be hacked through a method called spoofing that consists of using a fake finger to simulate the user one [Cao and Jain, 2016].

The idea of studying Visual Passwords for user authentication comes from these vulnerabilities and disadvantages and gives motivation to find a new way of recognizing users. In contrast to other biometric-based passwords, one thing in favour of visual passwords is the fact that only a camera is needed, given that they are based on recognizing only visual signals, and hacking like the fingerprint case is possible but much more unlikely.

This project is focused on a visual password, specifically, lip-reading. The term lip-

reading refers to recognizing the spoken words using visual speech information such as lip movements, that is why it is also called Automatic Lip-Reading (ALR) or Visual Speech Recognition (VSR) [Puviarasan and Palanivel, 2011]. It is a simultaneous method (biometric authentication combined with a password authentication). Moreover, not only a mouth is visually different than another, but each of us verbalizes in our unique way. We aim to investigate the use of lip-reading as a system to authenticate users in a double way; firstly, because of its biometric characteristics, related to the user, and secondly, recognizing the word spoken, oriented as a traditional password.

In order to achieve this objective, we will try to construct a speaker dependent automatic lip-reading system doing an extensive analysis from the data features to the chosen algorithm and its whole set of parameters: the approach that we expose is based on Hidden Markov Models and Gaussian Mixture Models (HMM-GMM) and on Discrete Cosine Transform (DCT) coefficients as features of the recognition.

HMM-GMM is a popular method for speech recognition applications, mainly due to its ability to handle variable-length input sequences and the ability to capture the temporal variability of speech, as well as the relationship between speech features (input) and the set of phones that form the word (equivalent to hidden states). In addition to these reasons, the flexibility of the resulting recognition system where one can easily change the size, type, or architecture of the models to suit particular words, sounds etc., and the ease of implementation of the overall recognition system [Rabiner and Juang, 1992] are valuable aspects of HMM-GMM. Also, the demonstrated efficacy within the several past years studies is remarkable. As far as the DCT coefficients of the video frames as input features, the ease of extraction from the images and its energy compaction property make them a good input option.

Related work shows the wide range of possibilities for constructing a model to achieve a good VSR system, from the data that will be the input of the system to the model itself.

Regarding to features used for describing lip motion, there are generally two kinds of them. [Zhao et al., 2010] The first one are local features, which include lip contours and specific reference points at a local level of the lips image. The second type of features are the ones taken observing the whole mouth at once (and not focusing on specific points of it), this group includes image transforms such as DCT. While the local kind of features need a very accurate tracking and a high-quality image, [Zhao et al., 2010], [Yuan et al., 2017], general features can be used in applications where circumstances are more inconstant, which are more related to our visual pass-

word proposal [Hassanat, 2014], [Lee and Myung, 2017].

In the history of Speech Recognition, a great variety of systems have been studied. Although in general, the main approaches are Hidden Markov Models and Artificial Neural Networks [Sujatha and Krishnan, 2012], [Zhao et al., 2010], there are interesting systems to be mentioned like Support Vector Machines [Lee and Myung, 2017], KNN method, used in the recognition step, [Hassanat, 2014] and systems based on image processing techniques and filtering [Desai and Gupta, 2012].

As already pointed out, HMM-GMM has been proved to be a very good choice for audio speech recognition (ASR), and so, to audio-visual speech recognition (AVSR) [Nefian et al., 2002]. After these conclusions, VSR studies have applied it too, obtaining, with the same background, a silent password that can lead us to a more secure solution in terms of daily use, situation in which is more difficult that our password is understood by others if we verbalise it with no sound at all.

Chapter 2

THEORY

In this section, we will explain the most relevant theoretical concepts for the project. Firstly, Discrete Cosine Transform as the descriptor of the video frames, input to our system. Secondly, Hidden Markov Models and Gaussian Mixture Models, the two principal models of the project because they are the basis of the recognition; together, they form HMM-GMM system and perform a temporal modelling of word sequences.

2.1 Discrete Cosine Transform

2.1.1 Image Processing and Transformations

Working with image inputs requires image processing, which can basically be summed up in three points:

- Image importing or acquisition
- Image manipulation, analysis and transformation
- Image exporting, which can consist of an altered image, an analysis of the image or a set of features of it

And the goals of image processing can be summarized in these points:

- Image representation and modelling
- Image enhancement
- Image restoration or image filtering
- Image compression

In our case, we will process video frames in order to achieve a representation or modelling of it, and use this representation as the input to our system.

Specially for compression and representation goals, image transformations are a keystone. Image transformations can go from simple operations, like rotations, to image conversions from one representation to another, for instance, Discrete Fourier Transform and Discrete Cosine Transform, that convert a spatial representation of an image to its representation in the frequency domain.

In the next subsections, the focus will be on Discrete Cosine Transform (DCT) because it is the transformation that will give us the input to our system.

2.1.2 Discrete Cosine Transform

The DCT represents an image in the frequency domain through a series of coefficients that are result of separating the image in parts of differing frequencies.

The DCT equation computes the i, j_{th} entry of the DCT of an image:

$$D(i,j) = \frac{1}{\sqrt{2N}} \cdot C(i) \cdot C(j) \cdot \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} p(x,y) \cdot cos[\frac{(2x+1) \cdot i\pi}{2N}] \cdot cos[\frac{(2y+1) \cdot j\pi}{2N}] \tag{2.1}$$

where

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{if } u = 0. \\ 1, & \text{if } u > 0. \end{cases} \tag{2.2}$$

and $D(i,j)$ is the resulting DCT coefficients, N is the image size and $p(x,y)$ is the x, y^{th} element of the image in the spatial domain represented by p.

2.1.3 DCT Coefficients and Energy Compaction Property

One of the properties of Discrete Cosine Transform is the capability of redistributing signal energy into a small number of transform coefficients.

In fact, before talking about Energy Compaction, it is important to understand the coefficients distribution in the DCT output matrix: The coefficients that represent low

frequencies are the upper-left corner ones and the coefficients in the bottom-right corner are the ones that carry high-frequency information.

Moreover, the kind of information that each type of frequency is not the same. In one hand, the low frequencies correspond to low varying information in the image, namely, the image background and the general appearance of the objects. In the other hand, and following the same idea, the high-frequency information is the one that defines the little details and remarks the edges between objects or inside objects themselves.
From these concepts, we can affirm that for visually recognizing the image reducing the number of coefficients we use (compressing), we need to keep a priority in the upper-left ones, as they carry the basic information.

In relation to these concepts, Energy Compaction Property of the DCT states that in the first coefficients there is almost all the Energy (or information) of the image. So, by using this rule, we can find the optimal number of coefficients for which the Energy does not increase that much if we keep using further coefficients, having a balance between Energy and Coefficients number.

2.1.4 DCT Reconstruction and Error Measures

The reconstruction of a DCT image consists on the application of the Inverse Discrete Cosine Transform. As its name suggests, it does the inverse transformation as DCT, goes from DCT coefficients to the spatial representation of an image.

Related to what we have been commenting in the previous section, what we can do is to reconstruct with only a part of the coefficients and have a good representation of the original image but dealing with less data.

Given a reconstruction, we can calculate how good it is compared to the original. Actually, there are many ways of doing it and given the circumstances, a way will be better than another.

Said so, in this project we use the Relative Error Norm (L^2):

$$L^2 = \frac{\sqrt{(x_r - x_g)^2}}{\sqrt{x_g^2}} \tag{2.3}$$

where x_g is the original image and x_r is the reconstructed image.

2.2 Hidden Markov Models

2.2.1 Introduction

While solving engineering problems, such as signal compression, prediction, reconstruction or analysis and understanding, all types of real signals need to be modelled. We can think of signals from a set of temperature measurements to speech signals or characters from a finite alphabet. Therefore, signal modelling allows us to represent a signal via some model parameters, and consequently, to make simulations of its behaviour and use it for a certain application.

There are two types of models, the deterministic ones, that define a signal by some known specific properties of it (e.g., a sine wave modelled by its amplitude and frequency), and the statistical or stochastic models, that characterize a signal based on its statistical properties, which includes Gaussian processes and Hidden Markov Models.

In this section, we explain the theory underlying HMM and how it can be used to process visual speech signals. A Hidden Markov Model is a tool for representing probability distributions over sequences of observations. In this model, it is considered that a given observation at a certain time is produced by a stochastic process. This stochastic process is defined by a state that cannot be observed, that is why it is called a hidden state. Another important assumption on this model is that each state at a certain time only depends on its predecessor state, the other previous states are forbidden.

The explanation of the theory is possible thanks principally to following the work of Valery A. Petrushin and Lawrence R. Rabiner in [Rabiner, 1989] and [Petrushin, 2000]. In addition, [Degirmenci, 2014] and [Collins, 2013] have been followed for more specific details.

2.2.2 Elements of an HMM

A Hidden Markov Model describes a system in which a sequence of observations O = O_1,O_2, O_3,...O_T that have happened at a time t = 1, 2, 3, ..., T, is represented as a hidden state sequence Q = S_1,S_2, S_3,...S_N, and this state sequence is determined by statistical parameters. We aim to know and use the hidden state sequences to perform a classification.

Thus, the necessary elements to construct an HMM are the following:

1. **N, the number of states in the model**

We denote individual states as S=S_1, S_2,...,S_N, and we refer to the state at time t as q_t.

2. **M, the number of distinct observation symbols per state**

   We denote the individual symbols as V=V_1, V_2,...,V_M.

3. **A=a_{ij}, the state transition probability distribution**

   where a_{ij} = P[q_{t+1} = S_j|q_j = S_i], $1 \leq i, j \leq N$.

4. **B = b_j, the observation symbol probability distribution in state**

   where $b_j(k) = P[V_k \, at \, t \mid q_t = S_j]$, $1 \leq j \leq N$, $1 \leq k \leq M$.

5. **π = π_i, the initial state distribution**

   where $\pi_i = P[q_1 = S_j]$, $1 \leq i \leq N$

An HMM model is defined as $\lambda = (A, B, \pi)$.

2.2.3 The Three Basic Problems in HMM Modeling

There are three main problems that an HMM solves and each of them are useful for different real-world applications. These issues are the following:

1. **The evaluation problem** Given a model and a sequence of observations, which is the probability that the observations have been produced by the model, $P(O \mid \lambda)$?

2. **Uncovering the hidden part of the model** Which is the *correct* sequence of states that has generated a certain sequence of observations?

3. **Optimization of the model parameters** Which are the parameters of λ that make the model represent in the best way a given sequence of observations?

Solution to the first problem

To know how accurate is a model for a given set of observations, we need to calculate $P(O \mid \lambda)$. From the concept or definition of this probability, we should consider every possible state sequence, that leads us to the following formulation:

$$P(O \mid \lambda) = \sum P(O \mid Q, \lambda) \cdot P(Q \mid \lambda) \tag{2.4}$$

So, we define this probability as the sum of probabilities of all possible sequences of states, and for each of them, we have to take into account two different probabilities:

- The probability that the sequence of observations has been produced by the state sequence and the model.

- The probability that the state sequence has been produced by the model

This way of calculating $P(O \mid \lambda)$ is very inefficient, due to the fact that there are N^T possible state sequences.
The alternative we have is to apply a procedure called the forward-backward procedure.

The forward-backward procedure is a dynamic programming algorithm. This type of algorithm solves big problems by dividing them into simpler ones that involve simpler calculations, storing all the results to use them the moment they are necessary. Seen the definition of our problem above, it is understandable the need of a method like this one.

The forward-backward procedure:

Consider the forward variable $\alpha_t(i)$ as the probability of the partial observation sequence $O = O_1, O_2, ..., O_t$, and the state S_i at time t (all the state sequences that end with the state S_i), given a model λ:

$$\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t = S_i \mid \lambda) \tag{2.5}$$

To obtain the value of $\alpha_t(i)$, the following recursive algorithm is used:

1. **Initialization** The forward probabilities are initialized as the joint probability of state S_i and initial observation O_1:

$$\alpha_1(i) = \pi_i \cdot b_i(O_1), 1 \leq i \leq N \tag{2.6}$$

2. **Induction** Recursive calculation of $\alpha_t(j)$

$$\alpha_{t+1}(j) = [\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}] \cdot b_j(O_{t+1}), 1 \leq t \leq T-1 \tag{2.7}$$

3. **Termination** Final calculation of $P(O \mid \lambda)$ as the sum of the terminal forward variables result of the previous step $\alpha_T(i)$

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \tag{2.8}$$

The same way we have done with $\alpha_t(i)$, we also consider a backward variable $\beta_t(i)$, as the probability of the partial observation sequence from t+1 to the end, given that the state at time t is S_i (being considered all the sequences that start with this state) and the model λ.

$$\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T, q_T = S_i \mid \lambda) \tag{2.9}$$

To obtain the value of $\beta_t(i)$, the following recursive algorithm is used:

1. **Initialization** Value arbitrarily set to 1:

$$\beta_T(i) = 1, 1 \leq i \leq N \tag{2.10}$$

2. **Induction** Recursive calculation of $\beta_T(i)$

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j), t = T-1, T-2, ..., 1, 1 \leq i \leq N \tag{2.11}$$

3. **Termination** Final calculation of $P(O \mid \lambda)$

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i)\beta_T(i) \tag{2.12}$$

Solution to the second problem

Solving the second problem implies to find the "optimal" state sequence. Unlike the previous case, there is not an exact solution for it.

An option is to choose the states q_t that are individually most likely, maximizing the expected number of correct individual states. However, we do not take into account the state transitions. As the transitions between states are important, the solution is to find the single best sequence of states that maximizes $P(Q \mid O, \lambda)$, equivalent to $P(Q, O \mid \lambda)$. To do so, Viterbi Algorithm is applied.

Viterbi algorithm:

Given an observation sequence $O = O_1, O_2, ..., O_T$, to find the single state sequence $Q = q_1, q_2, ..., q_T$ that best fits the observations, we first need to define the quantity:

$$\delta_t(i) = max_{q_1, q_2, ..., q_{t-1}} P[q_1, q_2, ..., q_t = i, O_1, O_2, ..., O_t \mid \lambda] \tag{2.13}$$

So, $\delta_t(i)$ is the highest probability along a single state path, at time t, which accounts for the first t observations and ends in state S_i.

We also define $\psi_t(j)$, an array to keep track of the arguments which maximized δ_t for each t and j.

These are the algorithm steps:

1. **Initialization**

$$\delta_1(i) = \pi_1 \cdot \beta_i(O_1), 1 \leq i \leq N \tag{2.14}$$

$$\psi_1(i) = 0 \tag{2.15}$$

2. **Recursion**

$$\delta_t(j) = max_{1 \leq i \leq N}[\delta_{t-1}(i) \cdot a_{ij}] \cdot b_j(O_t), \tag{2.16}$$

$$\psi_t(j) = argmax_{1 \leq i \leq N}[\delta_{t-1}(i) \cdot a_{ij}], \tag{2.17}$$

$$2 \leq t \leq T, 1 \leq j \leq N$$

3. **Termination**

$$p* = max_{1 \leq i \leq N}[\delta_T(i)] \tag{2.18}$$

$$q_T* = argmax_{1 \leq i \leq N}[\delta_T(i)] \tag{2.19}$$

4. **State sequence backtracking**

$$q*_t = \psi_{t+1}(q*_{t+1}), t = T-1, T-2, ..., 1 \tag{2.20}$$

Solution to the third problem

The solution to this problem leads us to the construction of the model. Hence, we need to determine a method to adjust the model parameters (A,B, π) in order to maximize the probability of the observations given the model (defined by this parameters).
The most popular way of solving this problem is using an iterative procedure called Baum-Welch method (also called Expectation-Maximization EM), it is a method that reestimates the model parameters through iterative update and improvement.

[Levinson et al., 1983] has been followed in order to structure this part of the explanation.

Baum-Welch method:

Firstly, we need to define $\xi_t(i,j)$, this variable is the probability of, given a model and an observation sequence, being in state S_i at time t, and state S_j at time $t+1$:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O, \lambda) \tag{2.21}$$

In fact, we can express this probability in terms of $\alpha_t(i)$ and $\beta_t(j)$, forward and backward variables respectively (being these calculated in the solution to the first problem):

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_j(O_{t+1}) \cdot \beta_{t+1}(j)}{P(O \mid \lambda)} \tag{2.22}$$

Then, we define:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j) \tag{2.23}$$

If we sum these two variables, we get:

$\sum_{t=1}^{T-1} \gamma_t(i)$ = expected number of transitions from S_i

$\sum_{t=1}^{T-1} \xi_t(i,j)$ = expected number of transitions from S_i to S_j

From these concepts, we can now formulate re-estimation equations for π, A and B:

$\bar{\pi}_i = \gamma_1(i)$: expected frequency (number of times) in state S_i at time t (t = 1)

$$\bar{A} = \bar{a}_{ij} = \frac{\sum_{t=1}^{T} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} = \frac{\text{expected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions from } S_i}$$

$$\bar{B} = b_j\bar{(k)} = \frac{\sum_{t=1,\, s.t\, O_t = V_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} = \frac{\text{expected number of times in } S_j \text{ and observing symbol } V_k}{\text{expected number of times in } S_j}$$

As Baum-Welch is an iterative algorithm, after the re-estimation of the parameters, $P(O \mid \lambda)$ is calculated.

Being $\lambda = (A, B, \pi)$ the initial model and $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ the re-estimated model, λ would be or equal or less likely that $\bar{\lambda}$ in the sense that $P(O \mid \lambda) \leq P(O \mid \bar{\lambda})$. This probability increases by iteratively use $\bar{\lambda}$ in place of λ and repeat the re-estimation until some limiting point is reached. The resulting model is called the maximum likelihood HMM.

2.3 Gaussian Mixture Models

For this section, [Reynolds, 2015] work has been followed.

2.3.1 Mixture Models and Parameter Estimation

Let X_1, X_2 . . . , X_m be a sample from a distribution P with density p. The goal of density estimation is to estimate p. This estimation will depend, in large part, on the complexity of P.

Given the situation in which we are dealing with complex data and we do not know its probability distribution, a model to describe this data (do its estimation) is needed. The most popular models for it are Mixture Models.

A Mixture Model or Mixture Density is a more flexible form of density estimation, made up of a linear combination densities:

$$p(x) = \sum_{j=1}^{M} p(x \mid j) \cdot P(j) \tag{2.24}$$

where $p(x \mid j)$ are the component densities and $P(j)$ are the mixing parameters.

The difficulty in constructing a good representation of the observed data is knowing which mixture components should be responsible for what data. To ease this process, the most used methods are Expectation Maximization (EM) algorithm and different classification and clustering algorithms.

2.3.2 Gaussian Mixture Models

The Gaussian Mixture Model is the most important mixture model, and its particularity is that the component densities are Gaussians, such that:

$$p(x \mid j) = N_j(x; \mu_j; \sigma_j^2), \tag{2.25}$$

each component Gaussian has mean μ_j and spherical covariance $\Sigma = \sigma^2$, and, given the previous definition of a Mixture Model, we could formulate the following:

$$p(x) = \sum_{j=1}^{M} N_j(x; \mu_j; \sigma_j^2) \cdot P(j) \tag{2.26}$$

This formulation gives a general idea of what a Gaussian Mixture Model is, but still something is missing. In fact, the output of a Gaussian Mixture Model is a weighted

sum of the gaussian mixture components. So, a more accurate definition of the model would be:

$$p(x) = \sum_{j=1}^{M} w_j \cdot N_j(x; \mu_j; \sigma_j^2) \cdot P(j) \tag{2.27}$$

where w_j are the weights of each gaussian component.

So, to consider GMM is generated, parameters w_j, μ_j and σ_j^2 need to be well estimated.

2.4 Hidden Markov Model and Gaussian Mixture Model

After introducing both models, Hidden Markov Models and Gaussian Mixture Models, it is important to understand why our system can be described as a combination of them, and in which point do they come together in practice. To refer to this combination, the abbreviation HMM-GMM is commonly used.

The purpose of this section is not to repeat the above said, but to establish the relationship between these two models, focusing more in our own application.

2.4.1 HMM-GMM model construction

Literally from [Yu and Deng, 2015], a GMM-HMM is a statistical model that describes two dependent random processes, an observable process and a hidden Markov process. The observation sequence is assumed to be *generated* by each hidden state according to a Gaussian mixture distribution.

To develop the system, we follow the solution to the third HMM problem, adjusting $\lambda = (A, B, \pi)$ for a fixed number of states N and a fixed number of gaussians determining a GMM, M. This latter variable is determining the number of gaussians that will form each state by a Gaussian Mixture Model.

We will, therefore, have a different Gaussian Mixture Model to represent each one of the states and we will train them during the training stage using Expectation-Maximization, while the other variables will be also being learned.

From this explanation, and following the previous definitions and formulations (in specific, equation 2.27), we define:

$$S_i = \sum_{j=1}^{M} w_j \cdot N_j(x; \mu_j; \sigma_j^2) \cdot P(j), 1 \leq i \leq N, 1 \leq j \leq M \tag{2.28}$$

Given 2.28, formulations of A, B and π (λ), α, β and ξ can be set out the following way:

- $a_{ij} = P[q_{t+1} \in \ S_j \mid q_j \in \ S_i], 1 \leq i, j \leq N$
- $b_j(k) = P[V_k \, at \, t \mid q_t \in \ S_j], 1 \leq j \leq N, 1 \leq k \leq M$
- $\pi_i = P[q_1 \in \ S_j], 1 \leq i \leq N$
- $\alpha_t(i) = P(O_1, O_2, ..., O_t, q_t \in \ S_i \mid \lambda)$
- $\beta_t(i) = P(O_{t+1}, O_{t+2}, ..., O_T, q_T \in \ S_i \mid \lambda)$
- $\xi_t(i, j) = P(q_t \in \ S_i, q_{t+1} \in \ S_j \mid O, \lambda)$

Chapter 3

DEVELOPMENT TOOLS AND DATA RESOURCES

It would not have been possible to carry out this research project without the following series of software tools and a complete database:

Database

The database that we have used in our experiments with the system is called GRID database, which includes 1000 utterances of a series of words from a 51 words vocabulary for 34 different speakers [Cooke et al., 2006].

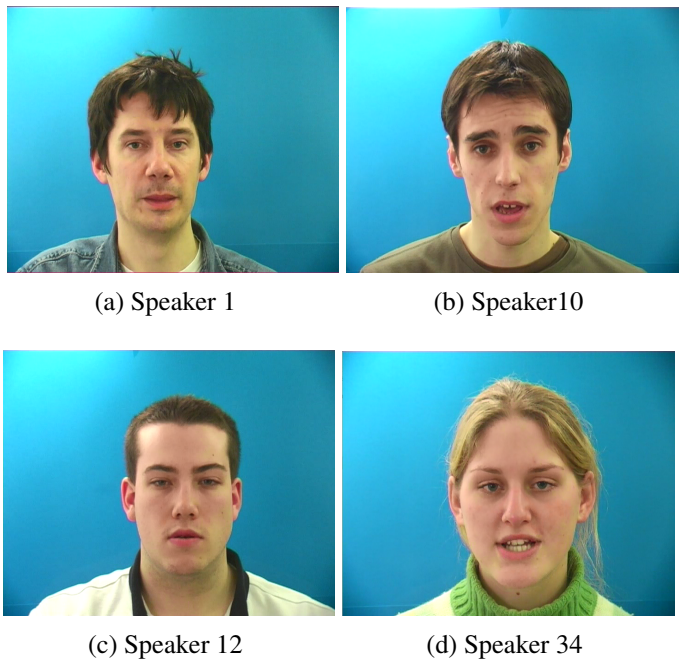
So, for each speaker, we have 1000 videos of a sentence made up with words from the vocabulary and with the structure *action, colour, preposition, letter, digit, adverb*. For instance, *bin blue at f two soon*. Being the words in each category:

- **Actions:** Bin, Lay, Place, Set
- **Colours:** Blue, Green, Red, White
- **Prepositions:** At, By, In, With
- **Letters:** All the letters of the alphabet
- **Digits:** One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Zero
- **Adverbs:** Again, Now, Please, Soon

Being the most repeated words the actions, colours, prepositions and adverbs. Each of them is repeated 250 times for each speaker. In addition, there is a file for each video that specifies the start and end audio sample for each word said in the sentence.

As for the technical specifications, videos frame rate is 25 frames/second, and their codification is MPEG-1, with a resolution of 720x576 pixels. The audio frequency rate is 44000 samples/second.

In our case, we had to adapt this database to our needs, from working only with the mouth part of every frame, to select only the frames of each word separately, because we worked word by word, not with the word sentences.



(a) Speaker 1 (b) Speaker10

(c) Speaker 12 (d) Speaker 34

Figure 3.1: Examples of video frames of different speakers

Matlab

All the computational processes of the project have been carried out using the software environment Matlab (R2017b).

External Code

Apart from the code developed by us, there are two code sources that have been of crucial importance for the project:

HMM-GMM model code: The code of the model is the core part of the project. To develop it, we have used a Toolbox called *GMM-HMM (multiple Gaussian) for isolated words recognition* from the Dayeh University's EE Department, by Lee-Min Lee and

Hoang-Hiep Le [Le, 2017].

This code is prepared for recognizing words using HMM-GMM, but the input to the model are coefficients from audio, that is why we have adapted it to turn the input into visual coefficients. Therefore, this code is the basis of our implementation of the model, but we have added functions designed to our own needs and made some adaptions for the same reason.

Face tracker: We don't focus on face tracking although we need to perform it in order to find the lip region and trim the frames around it. So, we are able to do the face tracking thanks to an external tracker developed by Oriol Martinez, researcher in UPF Department of Information and Communication Technologies (DTIC).

Internally developed code

The code developed specially for this project goes from adapting database videos to our needs to adapting the model code for the same reason. This process implies developing new functions and changing some parts and measures of the external code.

- **Lip-region sequences generation functions**

  Process described in section 4.2.1 is internally developed such that, given the mouth rectangle points, each of the frames is trimmed and all videos are saved as a group of frames (.mat format) that contains only lip-region of the original frames.

- **Feature extraction functions**

  - Given a word and a speaker, all lip-region sequences from that speaker where that word appears are selected.
  - For each of that selected sequences, information on its alignment file is read and processed (Equation 4.1).
  - With the processed values for each of the selected sequences, the frames that correspond to that word are separated to form new independent sequences.
  - For each of that new word lip-region sequences, DCT coefficients are extracted and, given a number of coefficients, that number of coefficients of the top-left part of the frame is kept.

- **Data grouping and splitting functions**

- **Data grouping**

  Given a characteristic requirement for the group of sequences, and a set of sequences, the output is a structure with the sequences that meet the requirements, prepared as the input to the system. The requirements can be, for instance, to group all sequences of a word in exception of a particular speaker (as in section 5.3) or to group all words from an speaker in exception of a particular word (as in section 5.2). The most basic example is to group all sequences of a particular word and a particular speaker, important for model construction.

- **Cross-Validation Data splitting**

  Given the training and validation percentages of data and a group of sequences previously grouped together, different sets of sequences are generated in order to perform every training and test following Cross-Validation indicated number of folds.

- **HMM-GMM system adapted functions**

  Generally, the adaptions have been in both stages, training and testing.

  - **Adaptations on accepted data input**

    Code was prepared for audio speech recognition, and so different data and default parameters needed to be changed. Also, and because we wanted to work with sequences of different lengths, we worked with structures and had to change some functions in order to match data structure type.

  - **Automation adaptations**

    As experiments (sections 5.2, 5.3, 5.4) were performed for different words, speakers, and parameters, automation was very necessary to save time mainly. So, all this part was added in accordance with our project needs.

- **HMM-GMM system developed functions**

  - **EER calculation function**

    EER is used in the project as an Error Measure and as a reference to estimate threshold, explained in section 4.3.2.

    Being Viterbi's output accepted, rejected and total amount of sequences, EER calculation function was constructed with 4.2 and 4.3.

  - **FAR Confusion matrix calculation**

    FAR is used in the project as an Error Measure. Explaned in section 4.3.2.

The same way we needed to calculate EER, a function was developed to calculate confusion matrices and at the same time, analyse output values (detecting maximum and minimum confusions, for example).

Chapter 4

SYSTEM'S OUTLINE

The recognition of words and users through lip-reading is the main goal of this project, having videos of different users saying different words (specified in the previous section) and a HMM-GMM model that performs the recognition.

This gives us the following general view:

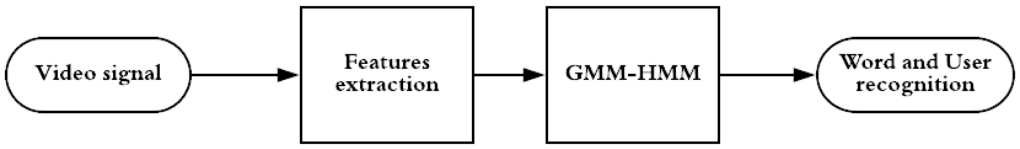


Figure 4.1: General project process diagram

It gets more complex when we go into detail about the two principal process blocks:

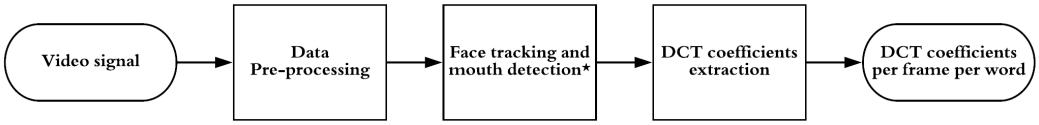


Figure 4.2: Features extraction process diagram[1]

[1]Face tracking is not part of the project, we do have the video signals without the mouth detection processing at first, but the face tracking and mouth detection is entirely performed by the external code mentioned in the previous section

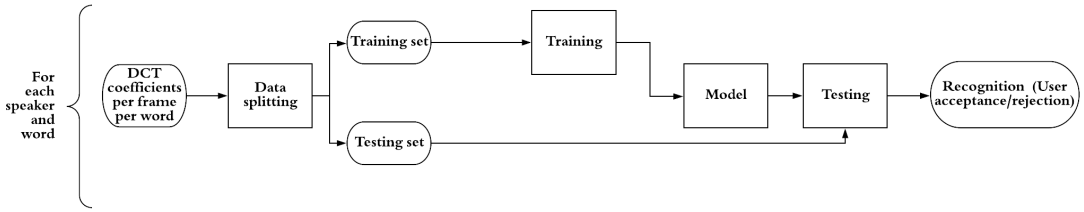


Figure 4.3: GMM-HMM project's system diagram

4.1 Data Pre-processing

As said in section 3, when introducing the database characteristics, each video contains a sequence of words.

For our application, we need to have sequences of frames for each word and not for arbitrary sentences. Every video comes with an information file called Alignment file; in order to perform the splitting, we will need to use this file that is structured like this:

0	t_1	silence
t_2	t_3	action
t_4	t_5	colour
t_6	t_7	preposition
t_8	t_9	letter
t_{10}	t_{11}	digit
t_{12}	t_{13}	adverb
t_{14}	t_{final}	silence

Table 4.1: Alignment file table

where t_i are the values of audio samples in which the word starts and ends respectively.

For example:

0	12500	sil
12500	21000	set
21000	27750	blue
27750	29000	at
29000	32750	a
32750	40500	four
40500	49250	now
49250	74500	sil

Table 4.2: Example of an alignment file structure table

So, for each file, we need to convert these audio sample values to frame values, this calculation can be done thanks to having the technical specifications of the videos. The conversion is the following:

$$f_i = t_i[audiosamples] \cdot \frac{1[second]}{44000[audiosamples]} \cdot \frac{25[frames]}{1[second]} \tag{4.1}$$

where f_i are the frame values equivalent to the audio samples t_i

Finally, for each word, every video in which it appears is split with these f_i values that indicate the first and last frame of the word in the video, and all sequences from the whole set of videos for a specific word are kept together in a structure variable.

Figure 4.4: Frame sequence of Speaker 34 saying ”Place”

4.2 Feature Extraction

4.2.1 Face Tracking and Mouth Detection

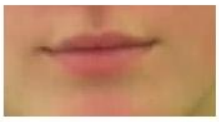
The feature extraction process starts with the videos and the face points extracted by the tracker. Then, these points are processed to define the lip region of each frame, in such a way that we obtain 4 points around the mouth that define a rectangle on each

frame.

The next step is to cut each frame by its rectangle sides, getting as a result the same sequence of frames but this time of the mouth only. (Illustrated in Figure 4.5)

(a) Face landmarks of a frame

(b) Mouth region rectangle of a frame



(c) Mouth snipped frame

Figure 4.5: Face Tracking and Mouth Detection procedure representation

4.2.2 DCT Coefficients Extraction

At this point, the only thing that is left is the DCT transformation of the frames and keep some of the coefficients as input to the system. The procedure is simple:

1. Images conversion to grey-scale

2. Application of the 2D-DCT to the grey-scale images (formulation in section 2.1.2)

3. Selection of how many coefficients are we going to keep.

4. Resize the coefficeints matrix to vector form.

4.3 HMM-GMM

4.3.1 Model construction

This section is dedicated to the learning process of the HMM-GMM system. It is important to recall that the input sequences will correspond to a word said by one user; in consequence, the model will correspond to that word and that user.

For a better understanding of the internal procedure, the following pseudo-code is provided:

Algorithm 1 Training stage pseudocode

```
1:  Input: Training set sequences
2:  Initialization
3:  for i = 1: number of splittings do
4:      for j = 1: number of iterations do
5:          for each input sequence do
6:              Forward-backward procedure
7:              Likelihood calculation
8:              Re-estimation of model parameters
9:          end for
10:     end for
11:     for each state do
12:         GMM Splitting
13:     end for
14: end for
15: Output: Models (one for each iteration)
```

Due to for loops 3 and 4, many models for the same data are generated every time the function is run. Number of splittings refers to number of Gaussians (M) of GMM, and number of iterations refers to the iterations for which model is built for every number of gaussians, as model construction is iterative.

So, having the for loop 3, we do not have to decide the exact number of gaussians that are going to be used to build the model, but a range of them, easing mainly the testing process. And this is why step 12 is GMM Splitting, which is a function that recalculates GMM in order to add another Gaussian to the GMM.

Also, it is significant that cross-validation has been carried out in all experiments (section 5.1), so the training for each model will be done at least two times (case in

which 50% of data is distributed in each set).

4.3.2 Threshold estimation

As our system is a recognizer and its application is a password, we can think of it as a classifier. In order to classify between Right data (correct user and word) and Wrong data (either incorrect user or word or both), we need to estimate a threshold.

To estimate the best threshold for that word and user, we compare, given the model, the output of Viterbi's function of the correct sequences and the incorrect sequences. Meaning by correct and incorrect sequences:

- **Correct sequences:** Testing set of sequences left from training having into account Cross-Validation.

- **Incorrect sequences:** All the sequences of other words and/or speakers, depending on the experiment planning.

We do so by calculating False Acceptance Rate (FAR), False Rejection Rate (FRR) and Equal Error Rate (EER).

$$FAR = \frac{FP}{N} \tag{4.2}$$

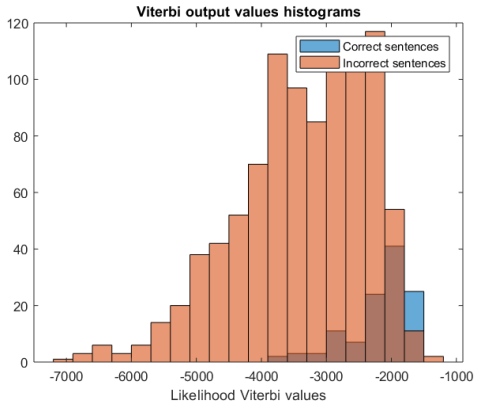
$$FRR = \frac{FN}{P} \tag{4.3}$$

where FP are the False Positives (incorrect data that has been detected as correct), FN are the False Negatives (correct data that has been detected as incorrect), N is the total amount of Incorrect data and P the total amount of correct data.

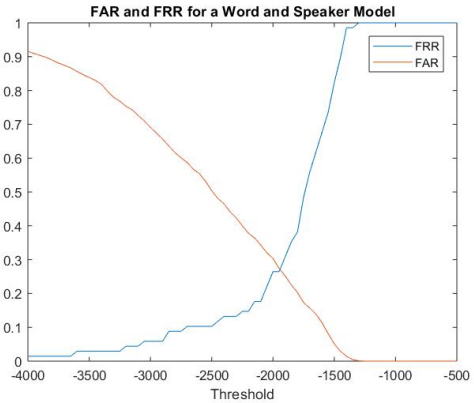
Equal Error Rate depends on these two values because is the point in which FAR and FRR are equal.

So, what we need to do is to calculate FAR and FRR for a set of thresholds and find the one in which FRR and FAR are equal. To use as a reference, EER = 0.5 is the EER result for random data.

It is remarkable that the values resulting from Viterbi's function are values between the interval $[-Inf, 0]$, where 0 is more likely to be correct and -Inf is not likely to be.



(a) Histograms of Viterbi's output likelihood from sequences of both sets (in blue, the same speaker and word sequences as the model and in orange, other words said by the same speaker (incorrect) sequences) for an specific model



(b) Example of FAR and FRR for EER and Threshold estimation of a word of a speaker vs. other words of the same speaker

Figure 4.6: Examples of graphics from Viterbi's function results

In the case of the figure 4.6b, the EER is 0.2767 and the Threshold fixed for that model would be -1955.

Chapter 5

EXPERIMENTS AND RESULTS

In this chapter some experiments with the system are exposed in order to understand its behaviour in relation to the different variables that affect it. Also, results of word and user recognition are collected. What we expect from these experiments is to evaluate the system's performance generally and specifically for words and users.

5.1 Variable Analysis

In order to perform the experiments, we need to plan the management of the different variables that have an effect on the model. These are the parameters that we need to have in mind:

- Training data amount
- Number of DCT coefficients per frame
- Number of states N
- Number of Gaussians M
- Words and speakers chosen

Training data

Directly from the database we should have 250 sequences of each word. Instead, we have less sequences due to data length restriction, video problems or tracker failures.

Data length restriction consists of discarding short sequences that are considered outliers and so, can corrupt the system. We have set the restriction such a way that length need to be bigger than the mean of the sequences minus 1 to be accepted.

We do this because we need to have always in mind the application of the system and limitations of the project; the fact that we have a word in, for instance, three frames is something unusual and that in controllable circumstances would not happen. A high frame rate would be a priority and database videos would not be the entrance of the system but a real-time recording, so, it is important to recall the current framework and do a certain adaptation to get close to the real application.

For this reason, we apply length restriction and, as a consequence, we do not have exactly the same amount of data for all the words. Approximately, there are between 180 and 225 sequences for word and speaker.

These approximately 200 sequences are split in training and testing sets.

The reference training data amount will be fixed in 50% of the whole set of that word and speaker. This way, we get to perform 2-fold cross-validation and still have a reasonable amount of data.

Number of DCT Coefficients per frame

To fix the number of coefficients per frame we have used as a reference the L^2 value (section 2.1.4, equation 2.3).

Lip region frames are very uniform. Therefore, DCT coefficients (and consequently, error reconstruction calculation) have been extracted after subtracting the mean value to the original image.

(a) Original lip-region frame

(b) Resulting lip-region frame from subtraction

Figure 5.1: Example of frame and the result of subtracting its mean to the pixel values

After this procedure, the number of coefficients set to enter to the system is the one that corresponds to the 80% of the energy, in terms of error measure, $L^2 = 0.2$.

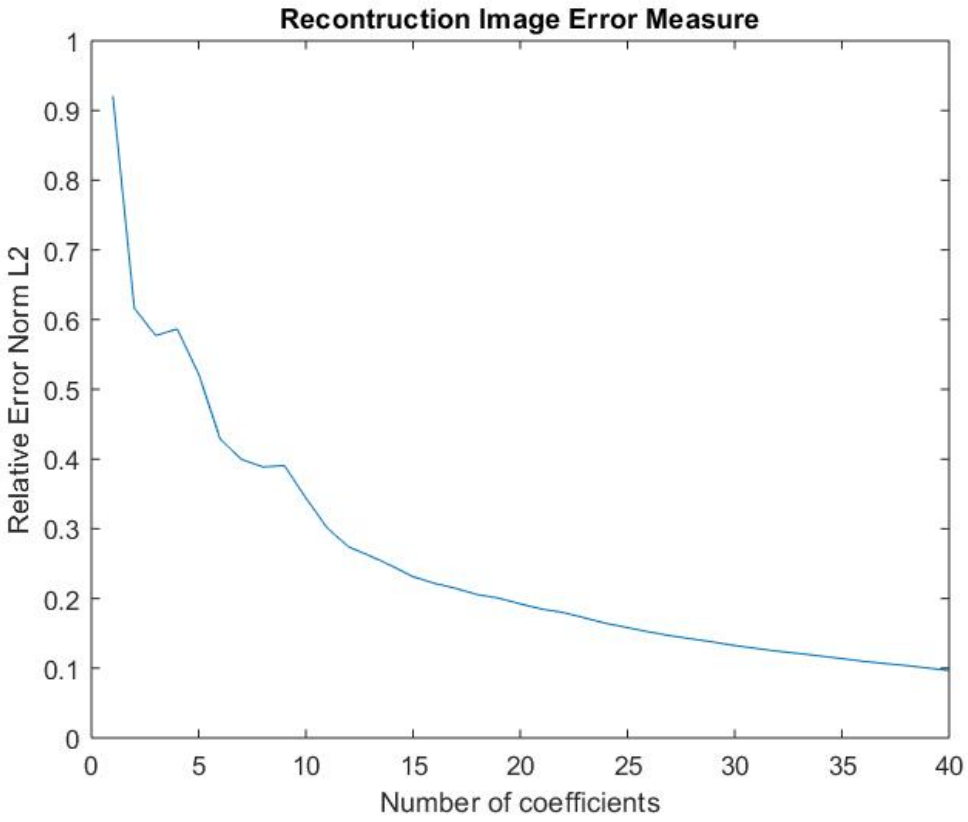


Figure 5.2: L^2 for all frames of all sequences of a word of speaker 33

where, the number of coefficients refer to the square side in pixels so that 5 is equivalent to 25 coefficients, 10 to 100 coefficients, etc. From the figure, we can see that the equivalent to 0.2 of error are 18x18 coefficients[2] with $L^2 = 0.2058$.[3]

Number of Gaussians M

M defines how many gaussians will form each GMM that define each state. In the case of recognising words, very short sequences between 3 and 10 frames approximately, a high M value was not optimal. M has been set to 2 in all the experiments, as in preliminary tests (that are not included in this project) a very low variability was shown if M was set in values between 2 and 4 gaussians.

[2]The lip-region frames size is 40x70 pixels

[3]This measure has been done with the other speakers and frames and the values are very similar, due to the uniformity of the whole database

Other parameters

As for the parameters left to comment, there is no relevant information to be given before the experiments. Our aim is investigate them through the tests.[4]

5.2 1st Experiment: Unipersonal Word Recognition

Objective:

Perform and analyse the word recognition between words said by the same user. We expect to detect which number of states works better for each word and which words make each model confuse the most. Also, we want to evaluate the system's results when recognizing words, extracting enough conclusive values to consider if we have a good approach for the application of a Visual Password.

Description:

Given a set of models (specified in the parameters section) with their optimal threshold for each word of a user, the experiment consists of comparing each word of a user with the other words of the same users individually and calculate FAR for the optimal threshold of that model.

Parameters:

This experiment will be carried out for 5 different users.

[4]Every paramater will be specified at the beggining of each experiment

Parameters	Information
Training Data Amount	50% of the sequences of each word
Number of DCT coefficients	324 coefficients (18x18)
Number of states N	From 1 to 8
Number of Gaussians M	2 (specified in section 5.1)
Speakers	5 speakers (tagged as 1,2,10,12,34 in GRID Database)
Words	Again, Bin, Blue, Green, Lay, Now, Place, Please, Red, Set, Soon, White

Table 5.1: 1^{st} Experiment Parameters table

Results:

The results of the first experiment are exposed in this section. It is important to say that, unless we are looking at the results of a speaker specifically, the results are the mean of the ones from various speakers for which the experiment has been carried out.

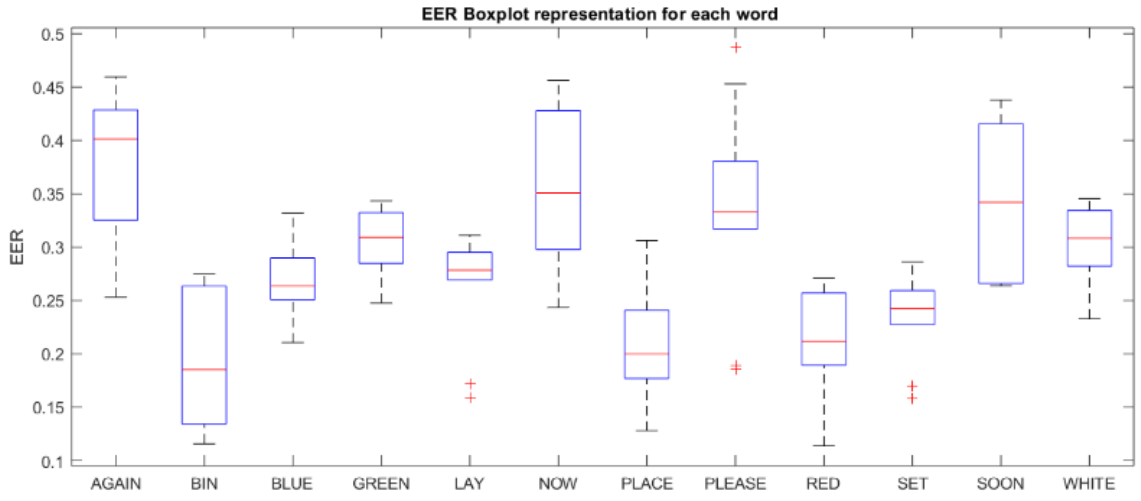


Figure 5.3: Boxplot representation of EER for each word model against other words from the same user

This figure gives a general view of EER values for each model word against other

words of the same user. We can see how all the values that we obtain are less than 0.5 and bigger than 0.1.

We can also observe that the highest EERs correspond to Again, Now Please, and Soon, all of them form the adverbs group, and they are the last ones of the sentences said by the users.

The models with the smallest EER values are Bin, Place and Red and the more variable results are Again, Bin, Now and Soon.

	AGAIN	**BIN**	**BLUE**	**GREEN**	**LAY**	**NOW**	**PLACE**	**PLEASE**	**RED**	**SET**	**SOON**	**WHITE**
AGAIN		0	0.8	7.8	22.6	31.5	5.2	54.5	0	20	56.4	9.8
BIN	4.7		78.9	32.9	14.3	9	21.3	0.2	67.4	10.8	5.7	34
BLUE	12.4	49.5		41.1	21.4	19.8	32.9	5.7	55.2	15.8	14.8	42.6
GREEN	21.3	27.9	45		31.1	24.5	45.5	8.5	29.6	28.8	28.2	68.5
LAY	10.2	11.7	10.1	27.8		14	29.4	19.4	8	36.3	11.8	26.1
NOW	36.8	0.5	3.6	10.4	11.7		14.2	26.6	0.2	14.2	43.2	15.2
PLACE	26.5	30.1	35.7	54.5	58.6	34.8		21.8	12.9	45.7	33.8	58.4
PLEASE	22.7	0	0	0	12.5	9.3	0.5		0	8.4	32.7	0
RED	1.9	53.5	67.2	28.7	9.9	4.1	9.6	1.4		3	1.9	28
SET	36.2	14.7	21.5	40.8	67.1	25.8	49.7	27.8	3.9		39.7	40.1
SOON	41.1	0	0.6	3.3	19.8	28.1	3.2	52.2	0	17.3		8
WHITE	19.8	25.4	40.8	59.7	28.3	22.8	41.6	9.6	25.7	23.6	25	

Table 5.2: FAR Confusion Matrix between different words of the same user (%)

In table 5.2 we can see False Acceptance Rate for each model against each particular word.[5,6] The columns of the table represent the model words and the rows represent the testing words. For a better analysis, the following figure is a visual representation of this table.
It is necessary to specify that both Figure 5.3 and table 5.2 contain the best model (in terms in number of states) for each word, information about this parameter will be provided later in this same section.

[5]Where each column represent a word model

[6]The exposed values in the table are the mean of the 5 speakers values

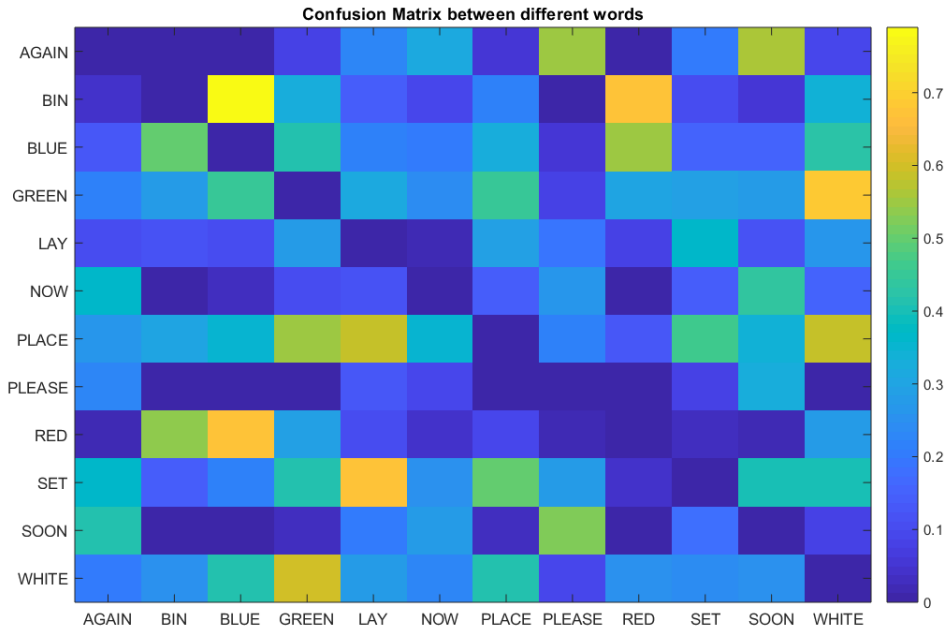


Figure 5.4: FAR Confusion Matrix Between different words of the same user (normalized values)

We can observe that the word models confusion matrix is considerably symmetric, this indicates that words are confusing in pairs, that is to say, if a model word confuses a lot with a particular word, the model of that confusing word has a high FAR for the same previous word. For example, we can see how Red model is confusing with Bin and Blue the most (0.67 and 0.55 of False Acceptance Rate respectively), if we then look at Bin model, we can see how Red FAR is 0.53 and Blue FAR is 0.49. The same happens with Blue one, where higher FARs are against Bin, 0.79, and Red, 0.67.

For a better understanding of the particular word confusions, more relevant information of each word is presented in the following table:

	Speaker 1			Speaker 2			Speaker 10			Speaker 12			Speaker 34		
	L	N	EER	L	N	EER	L	N	EER	L	N	EER	L	N	EER
AGAIN	6	6	26.3	9	7	41.2	6	5	43.2	7	7	33.0	8	7	42.4
BIN	4	1	27.4	4	4	25.7	4	4	17.6	5	5	13.3	5	2	13.4
BLUE	5	2	33.1	4	1	26.9	4	4	27.2	5	5	22.1	6	5	25.6
GREEN	4	4	30.4	5	5	34.1	5	5	27.7	6	3	31.7	6	6	28.4
LAY	8	6	29.4	5	5	28.4	9	7	26.4	6	6	16.5	6	5	29.9
NOW	5	5	30	9	7	42.1	5	5	35	7	7	24.9	8	7	44
PLACE	5	5	23.7	5	5	29.8	5	5	20	6	6	13.8	6	6	17.7
PLEASE	7	6	32	8	7	32.4	6	6	18.7	9	8	43.5	10	8	40
RED	4	3	25.9	4	3	20	3	1	12.3	5	5	26.1	5	4	20.4
SET	7	6	26.5	5	5	26.3	5	5	24.7	6	6	16.4	7	7	23.3
SOON	7	6	26.4	8	7	34.2	5	5	43.3	7	7	27.4	11	8	38.1
WHITE	4	3	30.8	5	5	32.2	5	5	34.2	6	6	27.8	6	6	29.2

Table 5.3: Mean sequence length (L), Best number of states (N) and EER (%) for each word and speaker

Through table 5.3 we can establish more relations between different data, and help us to build more accurate conclusions.

We can see how each speaker contributed differently in terms of EER, while for some words the results are significantly similar, there are words more variable (that is visible in boxplots, figure 5.3). Also, we observe that the maximum number of states that is chosen as the best result for some words is 8 (for Speaker's 34 Please and Soon). Testing has been actually done until 9 states, but any model word of any speaker worked with that amount of states. So, number of states is limited, too many states is not the best option but neither too few, as the general number oscillated between 3 and 6 states approximately.

Moreover, a general tendency in the data shown in the table is the relation between mean length of data and number of states. It is discernible that for short sequences like Bin or Red (which mean length goes from 3 frames to 5 for all speakers) best number of states go from 1 to 4, and the other way round, when mean length goes from 7 to 11, the best number of states is 7 or 8. The same happens with intermediate values like 5 or 6.

If we focus in EER ≥ 0.4, we get the following list:

- **AGAIN Speaker 2:** Mean length 9, N = 7
- **AGAIN Speaker 10:** Mean length 6, N = 5

- **AGAIN Speaker 34:** Mean length 8, N = 7
- **NOW Speaker 2:** Mean length 9, N = 7
- **NOW Speaker 34:** Mean length 8, N = 7
- **PLEASE Speaker 12:** Mean length 9, N = 8
- **PLEASE Speaker 34:** Mean length 10, N = 8
- **SOON Speaker 10:** Mean length 5, N = 5

The first impression from the list is that the words are the 4 adverbs as commented from the box plot. Also, there is a majority of N $\geq$ 7. So, we can associate the bigger Errors to large sequences and great number of states, but also to the fact that there are the words that the speakers always say at the end of the sentences, what could lead us to think of a problem related to the database or to the use we make of it.

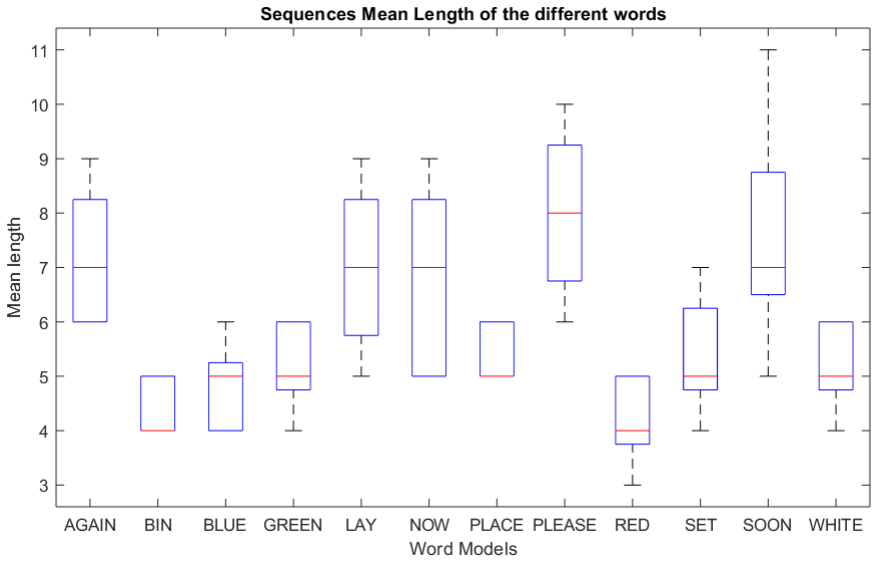


Figure 5.5: Mean Length of word sequences of all speakers Box plots

In relation to the confusion matrix commented above (figure 5.4), it is noticeable that confusion occurs between some words and that it is reciprocal. We can remind some of these confusion cases and extract some conclusions with the visual help of figure 5.5.

For instance, in the previous analysis we have identified the confusion of Bin, Blue and Red between each other, we can see how these 3 words are the ones with smaller mean length. The same happens with the words Green, Set, Place and White all of them confuse between each other the most and they have all the mean length of 5.

As a conclusion, we can say that results show that confusion is influenced by the

length of the words more than for some kind of mouth movement similarity and that large sequences generate worse models.

5.3 2nd Experiment: Multipersonal Word Recognition I

Objective:

Perform and analyse word recognition between different users comparing a word model with sequences of the same word said by different users. We expect to evaluate the system's results at recognising users, extracting enough conclusive values to consider if we have a good approach for the application of a Visual Password.

Description:

Given a set of models (specified in parameters section) with their optimal threshold of each word of a user, the experiment consists of comparing each word said by a user with the same word said by other users.

Parameters:

Parameters	**Information**
Training Data Amount	50% of the sequences of each word
Number of DCT coefficients	324 coefficients (18x18)
Number of states N	From 1 to 8
Number of Gaussians M	2 (specified in section 5.1)
Speakers	6 speakers (tagged as 1,2,10,12,33,34 in GRID Database), Speaker 33 only for testing
Words	Again, Bin, Blue, Green, Lay, Now, Place, Please, Red, Set, Soon, White

Table 5.4: 2^{nd} Experiment Parameters table

Results:

The most general way of observing the results of the models is through the EERs. In this case, and because we are analysing the user recognition, we expose EER values for word model and speaker. In addition, EER values are the ones corresponding to the best model in terms of number of states in every case.

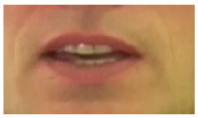
	Speaker 1	Speaker 2	Speaker 10	Speaker 12	Speaker 34
AGAIN	2.8	2.1	8	4	7.3
BIN	3.2	1.6	4.6	4.4	7.6
BLUE	2.6	0.8	7.7	3.9	7.9
GREEN	2.9	0.7	1.3	5.2	3.2
LAY	5.3	0.3	1	0	8.1
NOW	1.6	5.4	5.1	0.8	11.4
PLACE	4.3	1.2	1.7	2.4	5.5
PLEASE	5	1.9	0	6.1	18.6
RED	1.9	1.4	1.5	0.7	3.9
SET	6.5	1.4	1.1	3.8	7.9
SOON	2.4	0.9	0	6.2	19.3
WHITE	1.7	1.2	5.8	0.8	18.2

Table 5.5: EER (%) for each model word and speaker against the same word said by the other speakers

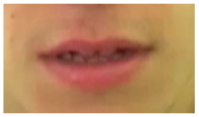
The first impression of the values from the table 5.5 is that errors are much lower than the ones obtained in Experiment 1 (Figure 5.3 and Table 5.3). In fact, values are very low, so user recognition is very well executed as far as we can observe. Broadly speaking, speaker recognition is better than word recognition.

The next step, given the obtained data, is to see if there is a pattern between user confusions.

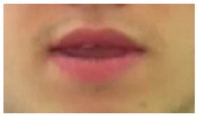
(a) Speaker 1 Mouth Region



(b) Speaker 2 Mouth Region

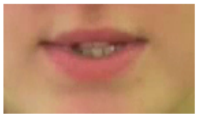


(c) Speaker 10 Mouth Region



(d) Speaker 12 Mouth Region

(e) Speaker 33 Mouth Region



(f) Speaker 34 Mouth Region

Figure 5.6: Example of mouth region frames of each speaker

In order to know if there is more confusion between specific speakers and to make a better judgement, lip-region images of each of them are shown in Figure 5.6. In addition to the visual information, point out that Speakers 1,2,10 and 12 are males and Speakers 33 and 34 are females. Also, and remembering the previous results where length sequences were important, say that relevant information is given in table 5.3.

	Speaker 1	Speaker 2	Speaker 10	Speaker 12	Speaker 34
Speaker 1		3.4	11.8	13	1.2
Speaker 2	0.14		0	0	1.3
Speaker 10	5.9	1		6	0
Speaker 12	12.9	3.3	0		8
Speaker 34	0	0	0	0	
Speaker 33	0	0	0	0	22.2

Table 5.6: Mean Confusion FAR (%) values between different speakers of all word models

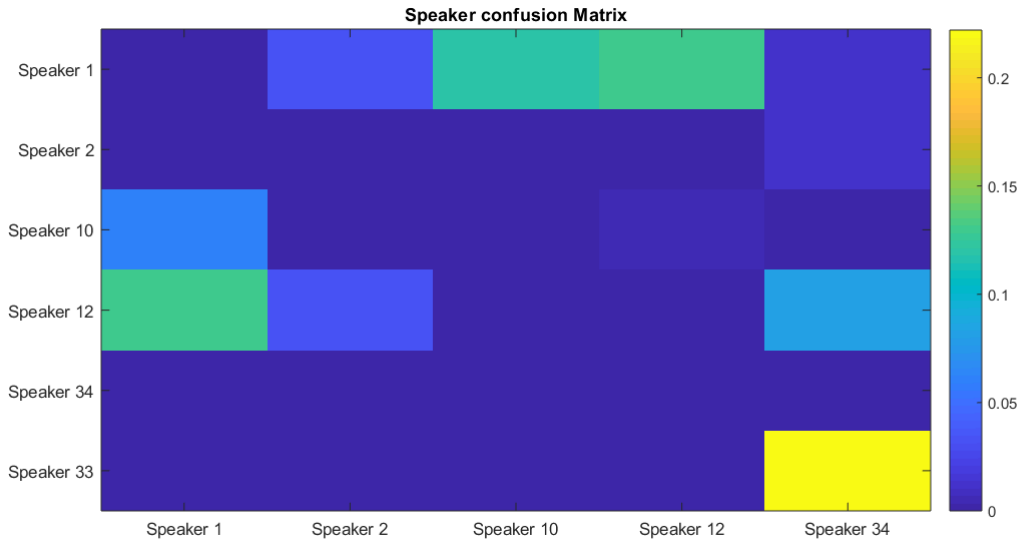


Figure 5.7: Speaker Confusion Matrix (normalised values)

Like in the previous experiment, in 5.6 and 5.7, columns represent the Models and Rows the tests. It is notable that we have constructed models for Speakers 1, 2, 10, 12 and 34 but not for 33, so there is one more row in this case, for Speaker 33 tests.

We can see how matrix confusion between users is symmetric. Speaker 1 confuses the most with Speaker 12 and Speaker 12 confuses the most with Speaker 1. It is observable that Speakers 1, 10 and 12 have more confusions between them. While Speaker 2 is the one with lower values, Speaker 34 is the one with the higher FAR value against Speaker 33. It is significant that none of the other speakers confuses with Speaker 33 or Speaker 34. To link these values with user information, Speakers 1,10 and 12 are all males and Speakers 33 and 34 are females.

So, the general conclusions are that user recognition is very good in terms of Equal Error Rate. As for particular confusions, they have a gender tendency and are recipro-

cal. Also, although the visual conclusions are considerably subjective, Speaker 2 has the most different mouth among the other speakers, with thinner lips, so we can attribute confusion to image factors but it is not disposable to think of temporal factors and verbalisation.

In relation to the number of states, results show that in this case it is not as decisive as in 1st experiment, considering that in a 60% of the cases, EER varies less than a 5% between models for the same word and speaker with different number of states. As for the other 40%, in any case varies more than a 20%.

Even so, in the following table the number of state for each model is shown:

	Speaker 1	Speaker 2	Speaker 10	Speaker 12	Speaker 34
AGAIN	6	7	4	7	7
BIN	2	1	4	5	5
BLUE	4	1	4	5	5
GREEN	4	5	5	5	6
LAY	6	2	7	7	6
NOW	5	7	5	5	7
PLACE	5	3	3	3	4
PLEASE	6	3	4	4	8
RED	4	2	1	1	5
SET	6	5	5	5	6
SOON	6	7	5	5	8
WHITE	3	5	5	5	6

Table 5.7: Best number of states for each speaker and word model for user recognition

As a reference, the 73% of the states that best worked for each model were the same amount as in the 1st Experiment (Table 5.3).

The conclusion regarding the number of states is that in order to recognise users is not as decisive as to recognise words, but the 73% of coincident numbers with the previous experiment gives the idea that there actually is a number of states more suitable for each word.

It is important to keep in mind the application of the system, for that application, both word and user are going to be recognised with the same amount of states; this case is contemplated in section 5.4.

5.4 3rd Experiment: Multipersonal Word Recognition II

Objective:

Perform and analyse the word and user recognition at the same time, with the same threshold and number of states for both uses.

Description:

Setting the parameters of the models from Experiment 1, test the models with data from the same user (other words) and the same word said by other users at the same time, this way, and because the experiment is more general, extract more final conclusions, recreating a complete situation of the application.

Parameters:

Parameters	**Information**
Training Data Amount	50% of the sequences of each word
Number of DCT coefficients	324 coefficients (18x18)
Number of states N	From 1 to 8
Number of Gaussians M	2 (specified in section 5.1)
Speakers	6 speakers (tagged as 1,2,10,12,33,34 in GRID Database), Speaker 33 only for testing
Words	Again, Bin, Blue, Green, Lay, Now, Place, Please, Red, Set, Soon, White
Thresholds	Estimated thresholds for Experiment 1

Table 5.8: 3^{rd} Experiment Parameters table

Results:

In order to recreate a more realistic application testing, 3^{rd} Experiment consists of testing user and word recognition with the same models, consequently, we need to set

more parameters than in previous experiments, like number of states and threshold. Given the results of both different experiments, the conditions for this experiments are the same as in experiment 1. We need to see how these changes affect user recognition.

	Speaker 1	Speaker 2	Speaker 10	Speaker 12	Speaker 34
AGAIN	13.4	23	21.7	10	15.3
BIN	13.5	13.4	8.1	6.9	17.7
BLUE	10.5	14.2	12.5	11.4	10.3
GREEN	15.9	18.5	14.1	20.3	15.1
LAY	15.7	16.3	15.3	9.8	10.8
NOW	16.2	19	19.2	17.7	21.4
PLACE	12.1	17.7	11.8	6.7	10.1
PLEASE	17.1	24.6	9.3	21.2	19.2
RED	12.6	11.5	7.2	14.6	20.1
SET	14.7	14.3	12.1	8.9	12.9
SOON	13.5	20.1	22.7	18.3	17.4
WHITE	16.5	19.3	17.8	12.4	13.3

Table 5.9: FAR (%) of each speaker and word model against other words and speakers jointly

In table 5.9, general results for each model are shown, being together as testing data either incorrect speaker data and incorrect word data from the same speaker. General FAR under these circumstances are between 20% and 10% approximately. In fact, mean general value for all models and speakers is 15% of False Acceptance Rate (always considering a little percentage of error).

Although general numbers are a good sample of values to extract conclusions, it is interesting to break down these percentages into the different cases commented during all the project, user recognition and word recognition. The values that compose these values are the following:

	Speaker 1		Speaker 2		Speaker 10		Speaker 12		Speaker 34	
	Same user	Other users	Same user	Other users	Same user	Other users	Same users	Other user	Same user	Other users
AGAIN	25.6	<0.1	38.9	<0.1	42.6	<0.1	32.1	<0.1	42.5	<0.1
BIN	27.8	<0.1	26.1	<0.1	21.7	<0.1	13.2	<0.1	11.4	12.6
BLUE	32	<0.1	27.4	<0.1	25.4	0.2	23	<0.1	25.1	4
GREEN	31.5	<0.1	33.1	<0.1	28.4	<0.1	33.2	9	23.4	0.2
LAY	28.2	0.1	27.4	<0.1	23.9	<0.1	15	<0.1	24.7	0.7
NOW	30.1	<0.1	41.7	<0.1	34.7	<0.1	16.2	<0.1	43.9	<0.1
PLACE	23.1	<0.1	30.9	<0.1	18.5	<0.1	13.2	<0.1	17.7	0.2
PLEASE	32.3	<0.1	31.8	<0.1	19.2	<0.1	42	<0.1	40.5	<0.1
RED	28.1	<0.1	20.3	<0.1	13.2	<0.1	25.2	<0.1	22	13
SET	28.5	0.1	24.9	<0.1	24	<0.1	15.9	<0.1	23.1	<0.1
SOON	26.3	<0.1	34.3	<0.1	43.3	<0.1	18	0.1	36.2	<0.1
WHITE	30.7	<0.1	33.7	<0.1	34.0	<0.1	25.3	<0.1	24.6	1

Table 5.10: FAR (%) of each speaker and word model against other words said by the same speaker and the same word said by other speakers

Comparing with table 5.5 where user recognition values are generally low, we show in table 5.10 that in most of the cases False Acceptance Rate between users is less than 0.1 or very low. So, applying favourable word recognition conditions works for user recognition reducing even more values obtained in Experiment 2.

It is observable that Speaker 34 is performing recognition worse than the other Speakers but it is difficult to attribute this fact to something in specific, because a lot of factors have a role in the recognition.

So, focusing in the application of the system, a visual password, user recognition is good in terms of FAR, but word recognition is much worse than what it should for a use like the one we aimed for.

Chapter 6

CONCLUSIONS

The objective of the project was to evaluate the proposed approach performing lip-reading and analyse the results at the time of recognising speakers and words. It is at this point, when we need to consider the feasibility of our system to be a Visual Password with a practical application to protect our virtual identity.
A system based on HMM-GMM and DCT coefficients, intended for the performance of visual speech recognition, has been disclosed within the scope of the project, hence has allowed us to explore in its architecture and variables and to set out some experiments to test if it was a good approach for the proposed application, a double password. Although the study could have included more tests involving more changes in parameters, we have set values for some of the variables to work in a common basis between experiments. Based on this premise, 3 experiments have been outlined; experiments involved testing, and before carrying them out, 60 models have been constructed, one for each word (12 words) for every speaker (5 speakers).

As 1^{st} and 2^{nd} experiments were more focused on exploring a specific type of recognition (word and speaker respectively), 3^{rd} experiment was a general one. Given the results of the previous ones, a more realistic situation was set up and gave a general evaluation of both types of recognition at the same time. The most general conclusion from them is that speaker recognition is good, but word recognition is not that positive.

Results show that we have a good approach at the time of recognising users, with a general user recognition Equal Error Rate of 4.23% in Experiment 2 (Section 5.3). As for word recognition, results are not as conclusive, being the general word recognition Equal Error Rate of a 26.9%.

So, the purposed approach could be a user identifier but the word recognition mis-

takes are too high to consider that we have achieved to read the lips of the speakers. Moreover, conclusions at a technical level in relation to the system have been drawn too, thanks to the above mentioned experiments.

As for word recognition, it is remarkable the importance of the duration of the word, translated to our parameters, the input sequence length, in the final result. It is shown that shorter words seem to construct better models. In addition, and strengthening the previous idea, the fact that there is a direct relation between number of states and sequence length is one of the conclusions, and that, for a large number of states (from 9 states specifically) the model fails at the time of building and, consequently, testing. The worst results in word recognition are the ones that involve long input sequences models.

Another observation has been the reciprocity in confusions in both cases, words and speakers (1^{st} and 2^{nd} experiment). Specially this fact could lead us to further studies, as well as dealing with the limit of the number of states.

In order to keep testing our system, it would be interesting to try on recognition with another database, and that way discard possible errors coming from this source, like the fact that the 4 word models that failed the most were situated in the same position in the sentences and the fact that every sequence had a considerably low number of frames.

We need to keep in mind that this is a project and that limitations have been present; a real application would be in a different situation and every factor would need an extensive evaluation. For the moment, in relation to the scope of this project, the conclusion is that a good user authentication is possible and FAR values support this statement, in contrast, a double password has not a solid basis because of the word recognition. However, further studies are exposed from this project to keep investigating in the field.

Bibliography

[Cao and Jain, 2016] Cao, K. and Jain, A. K. (2016). Hacking mobile phones using 2d printed fingerprints.

[Collins, 2013] Collins, M. (2013). The forward-backward algorithm. *Columbia Columbia Univ*.

[Cooke et al., 2006] Cooke, M., Barker, J., Cunningham, S., and Shao, X. (2006). An audio-visual corpus for speech perception and automatic speech recognition (l). 120:2421–4.

[Degirmenci, 2014] Degirmenci, A. (2014). Introduction to hidden markov models. *Harvard University*, pages 1–5.

[Desai and Gupta, 2012] Desai, P. and Gupta, A. (2012). Automated lip reading technique for password authentication.

[Hassanat, 2014] Hassanat, A. B. (2014). Visual passwords using automatic lip reading. *CoRR*, abs/1409.0924.

[Le, 2017] Le, H.-H. (2017). Gmm-hmm (multiple gaussian) for isolated words recognition.

[Lee and Myung, 2017] Lee, D. and Myung, K. (2017). Read my lips, login to the virtual world. In *2017 IEEE International Conference on Consumer Electronics (ICCE)*, pages 434–435.

[Levinson et al., 1983] Levinson, S. E., Rabiner, L. R., and Sondhi, M. M. (1983). An introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *The Bell System Technical Journal*, 62(4):1035–1074.

[Nefian et al., 2002] Nefian, A. V., Liang, L., Pi, X., Xiaoxiang, L., Mao, C., and Murphy, K. (2002). A coupled hmm for audio-visual speech recognition. In *2002 IEEE*

International Conference on Acoustics, Speech, and Signal Processing, volume 2, pages II–2013–II–2016.

[Petrushin, 2000] Petrushin, V. A. (2000). Hidden markov models: Fundamentals and applications. In *Online Symposium for Electronics Engineer*.

[Puviarasan and Palanivel, 2011] Puviarasan, N. and Palanivel, S. (2011). Lip reading of hearing impaired persons using hmm. *Expert Systems with Applications*, 38(4):4477 – 4481.

[Rabiner, 1989] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

[Rabiner and Juang, 1992] Rabiner, L. R. and Juang, B. H. (1992). Hidden markov models for speech recognition — strengths and limitations. In Laface, P. and De Mori, R., editors, *Speech Recognition and Understanding*, pages 3–29, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Reynolds, 2015] Reynolds, D. (2015). *Gaussian Mixture Models*, pages 827–832. Springer US, Boston, MA.

[Sujatha and Krishnan, 2012] Sujatha, P. and Krishnan, M. R. (2012). Lip feature extraction for visual speech recognition using hidden markov model. In *2012 International Conference on Computing, Communication and Applications*, pages 1–5.

[Yu and Deng, 2015] Yu, D. and Deng, L. (2015). Hidden markov models and the variants. In *Automatic Speech Recognition*, pages 23–54. Springer.

[Yuan et al., 2017] Yuan, Y., Zhao, J., Xi, W., Qian, C., Zhang, X., and Wang, Z. (2017). Salm: Smartphone-based identity authentication using lip motion characteristics. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8.

[Zhao et al., 2010] Zhao, G., Huang, X., Gizatdinova, Y., and Pietikäinen, M. (2010). Combining dynamic texture and structural features for speaker identification. In *Proceedings of the 2nd ACM workshop on Multimedia in forensics, security and intelligence*, pages 93–98. ACM.