André Belchior Mourão

Licenciado em Engenharia Informática

Robust facial expression analysis for affect-based interaction

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática

Orientador: Prof. Doutor João Magalhães, Prof. Auxiliar, FCT/UNL

Co-orientador: Prof. Doutor Nuno Correia, Prof. Catedrático, FCT/UNL

Júri:

Presidente: Prof. Doutor Francisco Azevedo

Arguente: Prof. Doutor Manuel Fonseca

Vogal: Prof. Doutor João Magalhães

FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

novembro, 2012

Robust facial expression analysis for affect-based interaction

Copyright © André Belchior Mourão, Faculdade de Ciências e Tecnologia,
Universidade Nova de Lisboa.
A Faculdade de Ciências e Tecnologia e a Universidade Nova de Lisboa têm o direito, perpétuo e sem limites geográficos, de arquivar e publicar esta dissertação através de exemplares impressos reproduzidos em papel ou de forma digital, ou por qualquer outro meio conhecido ou que venha a ser inventado, e de a divulgar através de repositórios científicos e de admitir a sua cópia e distribuição com objetivos educacionais ou de investigação, não comerciais, desde que seja dado crédito ao autor e editor.

Acknowledgements

To Prof. Dr. João Magalhães, my thesis advisor, for his immense support and knowledge. Without his capacity to help and motivate me during the last year, writing this thesis would have been a much harder task.

To Prof. Dr. Nuno Correia, my thesis advisor, for his help and availability throughout the process of writing this thesis.

To my colleagues from the university, especially to Ricardo Marques, Pedro Severino, Tânia Leitão, Filipe Carvalho, João Boiça, Flávio Martins, Pedro Borges and Filipa Peleja for their friendship and support in this important period in our lives.

To my long-time friends, Rui Assunção, Tiago Luís, David Santos, João Rato, Marta Martinho, Nuno Barreto and Filipe Lopes for always being there.

To my loving mother Ana Belchior, for constant support and for giving me the strength to overcome all the problems I encountered.

To my brother Miguel Mourão, for helping me relax even as the deadlines approached.

To my father Paulo Mourão, for is love and for always helping me in times of need.

Abstract

Interaction is moving towards new and more natural approaches. Human Computer Interaction (HCI) is increasingly expanding towards more modalities of human expression such as gestures, body movements and other natural interactions. In this thesis, we propose to extend existing interaction paradigms by including the face as an affect-based input.

Affective interaction methods can greatly change the way computers interact with humans; these methods can detect displays of user moods, such as frustration or engagement and adapt the experience accordingly. We have created an affect-based framework that encompasses face detection, face recognition and facial expression recognition and applied it in a computer game.

ImEmotion is a two-player game where the player who best mimics an expression wins. The game combines face detection with facial expression recognition to recognize and rate an expression in real time.

A controlled evaluation of the framework algorithms and a game trial with 46 users showed the potential of the framework and success of the usage of affect-based interaction based on facial expressions in the game. Despite the novelty of the interaction approach and the limitations of computer vision algorithms, players adapted and became competitive easily.

Resumo

Interação com computadores está a mudar em direção a abordagens mais naturais. Os paradigmas de interação pessoa-máquina estão em expansão, integrando novas expressões humanas como gestos, movimentos do corpo. Nesta tese, propomos estender os paradigmas de interação pessoa-máquina, utilizando a cara como um método de controlo.

Estas novas técnicas podem mudar a maneira como interagimos com os computadores. Um programa pode ajustar a experiência do utilizador consoante os estados afetivos detetados (i.e. frustração ou divertimento). Criamos um *framework* que incluí deteção e reconhecimento de caras e reconhecimento de expressões faciais e estudamos a sua aplicação num jogo.

O ImEmotion é um jogo para dois jogadores onde o jogador que melhor imitar uma expressão vence. O jogo combina deteção de caras e reconhecimento de expressões faciais para classificar e pontuar expressões em tempo real.

Uma avaliação controlada dos algoritmos da *framework* e um teste com 46 utilizadores revelaram o potencial da *framework* e o sucesso das expressões faciais como método de controlo do jogo. Apesar das limitações dos algoritmos de visão por computador e das técnicas utilizadas serem novidade, os jogadores adaptaram-se facilmente e tornaram-se competitivos rapidamente.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Context and motivation

Human interaction encompasses more than words. The face is a fundamental part in interaction and facial expressions can express the tone or intention of what is being said. When one meets an acquaintance, one instantly recognizes him as a friend and sees his displays of affect. This simple and day-to-day routine is beyond computer capabilities. If emotions are a key part of human interaction, why is HCI mostly ignoring this input? In this thesis we aim at delivering such functionalities as a general interaction framework. Formally, the objective of this thesis is to

research face recognition and facial expressions recognition as an enabler of affective interaction paradigms.

Affective interaction methods can greatly change the way computers interact with humans; these methods can measure displays of affect, such as frustration or engagement and adapt the experience accordingly.

Steps towards more natural interactions are gaining momentum and enabling users to interact through facial expression [1–3], gestures and other movements [4] (using depth-sensors, such as, Microsoft Kinect [5]). This trend is also observed in recommendation systems that use text sentiment analysis [6].

We argue that a computer should react to the player's facial expression as the sole input method, Figure 1, or combined with other method. For example, in a fighting game, a punch thrown with an angry face could cause more damage; a media recommendation system could alter its recommendations based on the reaction of the users to the current program.

We have created an affect-based framework that encompasses face detection, face recognition and facial expression recognition and its application in a game.

Figure 1. Players interacting with ImEmotion

1.2 Proposed framework

Our **affect-based interaction framework** opens a world of possibilities. Automatically capturing information about who is in the living room and their facial expressions provides a new level on input unobtrusiveness.

The framework was built for the living room and takes into account the limitations of that environment. The following design constraints were considered:

- **Minimal user intervention**: Traditional input methods (i.e. keyboard and mouse) are not natural control mechanisms for the living room. A remote control is useful to perform simple actions like changing channel and adjusting volume. However, complex tasks (i.e. inputting text) can become painful experiences. Therefore, we only require manual intervention of the user when there is no other alternative (i.e. inputting a name). The remaining processes are fully automatic (face detection, tracking and facial expression recognition);
- **Real time recognition**: the framework was designed to respond to the users' actions in real time. Therefore, we focused on techniques that allowed the processing of several images per second, making the framework ideal to applications where fast response is critical (i.e. games and media recommendations).

Taking into account our requirements and constrains, Figure 2 illustrates a high level view of the proposed framework divided into its main components:

- **Face recognition and tracking** is divided into the face detection and face recognition. Face detection is the component that detects faces in an image and pre-processes them for face recognition and facial expression recognition. Face

recognition assigns a name to a face and is able to learn to identify previously unknown individuals with a very small delay (under ten seconds);

- **Robust facial expression recognition** is the component that deals with the representation and fast classification of facial expression. It recognizes the expression of a face detected with the face detection algorithm and is also capable of measuring the dissimilarity between facial expressions.

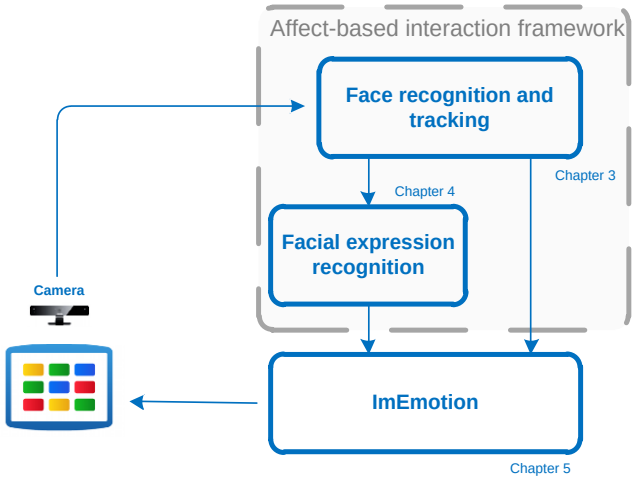

Figure 2. Proposed framework

1.3 Contributions

The contributions of this thesis are:

- **An affect-based interaction framework** that relies on complex image processing algorithms to recognize users and facial expressions in real-time. Unlike most facial analysis algorithms, our proposal handle images automatically without any user intervention;
- **ImEmotion game:** an interactive game that explores affective features measured by the affect-based interaction framework. Affect-based computer interaction still has many challenges to be researched and we believe the proposed game illustrates how such novel interaction paradigm can be embedded in computational systems. We researched robust methods of measuring displays of affect, allowing for fair competitiveness.

1.4 Organization

This thesis is organized as;

- **Chapter 2 - Background and related work**: presents background concepts, and the main algorithms for face detection, face recognition, facial expression recognition and affect-based interaction;
- **Chapter 3 - Face recognition and tracking**: presents the implementation details and evaluation of face detection, tracking and face recognition algorithms;
- **Chapter 4 – Robust facial expressions recognition**: presents the implementation details and evaluation of a robust facial expression recognition algorithm and facial expression dissimilarity measure;
- **Chapter 5 - Affective-based interaction: ImEmotion**: describes ImEmotion, a competitive game based on facial expressions built on top of our framework. It includes the game implementation details and an evaluation with real world data;
- **Chapter 6 - Conclusions**: a summary of the contributions, achievements and limitations will be discussed in this Chapter.

2

Background and related work

2.1 Face detection and recognition

Aryananda [7] worked with Kismet, an humanoid robot created by Breazeal [8] that recognizes people thorough natural interaction. Kismet created models from captured face images and compared them with the existing ones in real time. Kismet is also capable of adding new individuals during the interaction. The results shown were promising, as the robot was capable of creating recognition profiles for 4 out of 9 people, without having any misidentification or false positives.

The robot's framework possesses some similarities with our approach regarding face detection and recognition. It uses the Viola and Jones' [9] algorithm for face detection and eigenfaces with Principal Component Analysis (PCA) to project the captured faces. Our objective is to do a system focused on a living room environment, with the addition of facial expression recognition. The technologies presented are described in detail in this Chapter.

2.1.1 Viola and Jones' face detector

The Viola and Jones' [9] face detector framework consists of a group of algorithms for real time face detection. It was presented in 2001 and became one of the most cited and recognized face detection frameworks [7], [10]. The framework takes a machine learning approach to face detection, using an optimized version of the AdaBoost algorithm [11] to create a classifier based on Haar features. Their ideas are the starting point for the face detection Chapter of this thesis.

Features

The framework requires the extraction of image features. The features that are used are Haar-like features that consist of rectangular areas divided in N parts with varying

patterns. The full set used by OpenCV is shown in Figure 3. The value of the rectangle feature is the difference between the sum of the pixel values inside the rectangular regions [9]. A general method that can be used for all the features in Figure 3 is to sum of the values of the pixels of areas in black and subtract the sum of values of the pixels of the white areas.

The computation of features for images is a computational intensive process if the individual pixels values are used for each rectangular area. To accelerate the process, an integral image is used. An integral image consists of a matrix with the size of the image, where each location (x, y) contains the sum of the values of the pixels above and to the left of (x, y). The integral image is computed as

$$ii(x, y) = \sum_{x' \leq x,\ y' \leq y} i(x', y'),$$

where $ii(x, y)$ is the integral image and $i(x', y')$ is the original image.

The rectangular features are simple and are only present in vertical, horizontal or diagonal orientations. To overcome this limitation, a large set is generated, allowing a representation that is 400 times overcomplete [9]. The conjunction of the computation simplicity of the rectangular features and the usage of a very large set mitigates their apparent limitation.

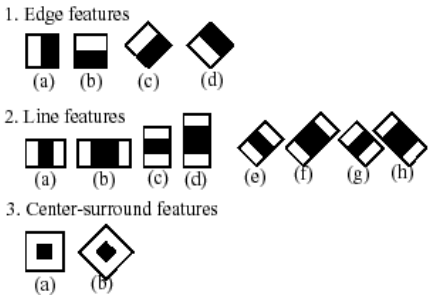

Figure 3. Haar features for each sub-window[1]

To extract the features from the images, each image is divided in subwindows of 24 x 24 pixels. Each one of these subwindows is scanned for features. Figure 3 illustrates the full set of features scanned per sub-window.

Classification algorithm

The original AdaBoost algorithm was created in 1995 by Freund and Schapire [11], as a new approach to machine learning algorithms. It combines the results of a weighted set of weak classifiers into a strong classifier [10]. The weak classifiers in this algorithm consist of simple classifiers that have only a slight correlation with the true classification (marginally better than random guessing).

[1] http://opencv.willowgarage.com/documentation/object_detection.html

The Viola and Jones' version builds on top of this algorithm, introducing some slight changes. In their version, the individual weak classifiers correspond to a single Haar feature. Given a set of Haar features, the algorithm will select the Haar features which best distinguish positive from negative examples. An example can be seen in Figure 4, where the first two features chosen by AdaBoost for an example training data are shown.

Cascade classifier

To improve the efficiency of the classification algorithm, Viola and Jones implemented a cascade of classifiers. This cascade consists in a set of AdaBoost classifiers that try to reject the less promising sub-windows of an image as fast as possible. This is achieved by feeding the sub windows to individual weak classifiers and rejecting them if they fail in any of them. Figure 5 illustrates the cascade process; $(c_1, \ldots, c_n)$ represent the individual classifiers.

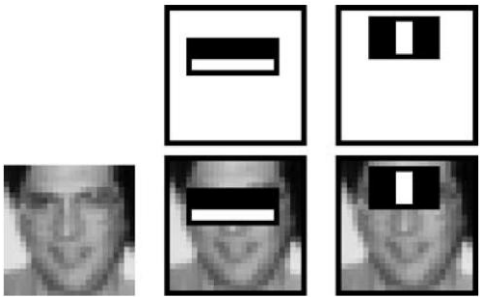

Figure 4. Example of the -fFirst two features selected by AdaBoost. Illustration taken from the work of Viola and Jones [9]

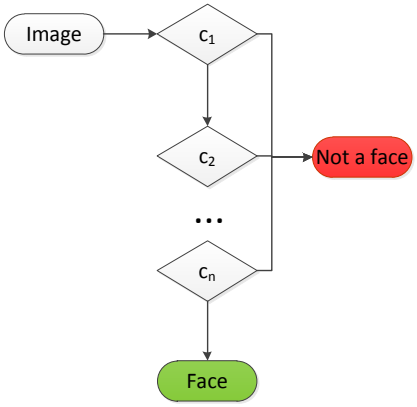

Figure 5. Cascade classifier example[2]

Initially, the tested features are simple to compute but have a high rate of false positives (about 50%). The purpose of the first stages is to eliminate a large number of negative examples with little computational effort, using the complex classifiers only deeper down the cascade.

2.1.2 Face recognition with eigenfaces

Face recognition is one of the most important components of the system. Through face recognition, we are able to recognize people in multiple sessions without manual intervention. For this step, we will focus on eigenfaces, based on the works of Turk et al. [12]. There is a lot is research in the face recognition area. The seminal works of Turk et al. [12] were the first to achieve notable results, and are still amongst the ones with top

[2] http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html

performance in recent comparative studies [13]. They use an approach based on eigenvalues and eigenvectors (called eigenfaces) to reduce dimensionality of data, retaining the significant information. The eigenfaces approach was taken in various studies [7], [14], [15] and can be combined with different projection techniques: Principal Component Analysis (PCA), Fisher Linear Discriminant (FLD) [13], [16] and Evolutionary Pursuit (EP) [13] and different similarity measurement techniques: Euclidean-,Cosine- and Mahalanobis-distance, SOM-Clustering, Fuzzy Feature Contrast (FFC) [13].

Eigenfaces

Eigenfaces [12] consist of the most representative differences between the training data and an average face. They can be seen as a set of "standardized face ingredients“: for example, a face might be composed of the average face plus 20% from eigenface 1, 35% from eigenface 2, and -12% from eigenface 3. In Figure 6, we show four eigenfaces represented as a face-like images.

The set of eigenfaces used for projection is called the face space. It is created using grayscale images of faces with the same size and scale and the eyes roughly in the same place. This is necessary because the algorithm is very sensitive to variations in size, alignment, scale and orientation. The algorithm for building the face space is the following:

- Consider a set S of k training images with width w and height h. To help with the notation, we will consider the dimensionality of an image as $m = w \times h$.
- Initially, the average face A is calculated by taking the average pixels values of all training images;
- Then, the differences between each training image $T_i \in S$ and the average A are computed into a matrix Z (size: $k \times m$).

$$Z_{i,*} = T_i - A \qquad with\ 0 < i < k$$

The vectors are then subject to Principal Component Analysis (PCA) to determine their associated eigenvalues and eigenvectors (size: $L \times L$, with $L \ll M$). The eigenvectors are called eigenfaces in this context, due to their face-like appearance (Figure 6). The PCA process is used to reduce the dimensions of the images into a manageable number. It seeks the orthonormal vectors that best represent the data distribution, using a co-variance matrix. To achieve a reduction in the dimensionality of the data, only the L vectors with highest variance are chosen, with $L \ll M$. The process used for the calculation of the eigenfaces uses the co-variance matrix and is described in the next paragraph:

1) Initially, the co-variance matrix between the all the dimensions of all images is

calculated, resulting in a matrix A with the size M^2;

2) Then, the eigenvalues are calculated by solving the following equation:

$$\det(A - \lambda I) = 0\,,$$

where λ represents the eigenvalues and I represents the identity matrix with the same dimensions as A (M^2). This equation has as many solutions (λ values) as the original number of dimensions (M^2). To achieve the desired dimension reduction, the highest L eigenvalues are chosen and the respective L eigenvectors are calculated using

$$A \cdot \lambda = V \cdot \lambda\,,$$

where λ is an eigenvalue and V an eigenvector. These eigenvectors correspond to the eigenfaces and constitute the face space.

After the construction of the face space, it is possible to project new faces into the face space. This projection is done by subtracting the new face from the average one and calculating the weights corresponding to all the eigenfaces in the face space. This projection can now be compared with other projections made in this face space using a common distance measurement like the Euclidian distance or the Mahalanobis distance.

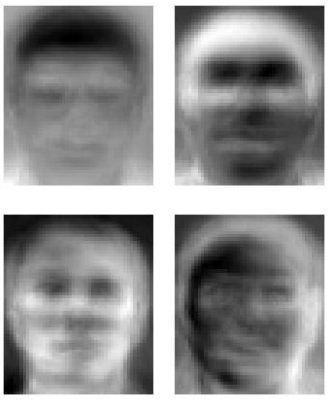

Figure 6. Example eigenfaces taken from the "Our Database of Faces" [17][3]

There are other methods for projection like Fisher Linear Discriminant (FLD) and Evolutionary Pursuit (ED), but we have chosen PCA because it achieves good results in various experiments and of ease of implementation (it is included in the OpenCV library).

[3] http://en.wikipedia.org/wiki/File:Eigenfaces.png

2.2 Facial expressions

In the words of Tian et al. "Facial expressions are the facial changes in response to a person's internal emotional states, intentions, or social communications." [18]. The relation between facial expressions and emotional states is being studied since the times of Darwin [19] and nowadays there still is a lot of research in this area [18], [20–22]. The applications of facial expression recognition range from detecting whether the subjects' eyes are opened to the detection of *Happiness* and other complex facial expressions. In this Chapter, we will discuss how to represent and recognize facial expressions from face images.

2.2.1 Representing facial expressions

Humans are able to recognize different facial expressions and infer what emotion that expression conveys. *Happiness*, *Anger*, *Surprise* are some of this emotion specific expressions and can be almost universally identified by humans [23]. But changes in facial expression can be more subtle like moving the outer section of the brows or depressing the corners of the lips. A representation is necessary to formalize the information related to the state of different face features (like mouth, nose and eyebrows). We have chosen the Facial Action Coding System (FACS) [18]. FACS primary goal was "to develop a comprehensive system which could distinguish all possible visually distinguishable facial movements" [21]. FACS is an index of Action Units (AUs). An AU is an individual action that humans are able to distinguish, that can be performed by one or more muscles of the face. There are a total of 46 AUs (see Table 1 for some examples) related to facial expressions, divided between upper face action units, lower face action units and miscellaneous actions. The upper face AUs are related to actions above and including the eyes, including movement in the brows, eyes and nose bridge. The lower face AUs are related to the lower section of the face just below the eyes, including movement in the lips, jaw, and lower part of the nose. Table 1 illustrates the upper and lower face AUs. This system is being widely used as a standard since its introduction and it combines sets of different positions in face muscles and features to determine an underlying facial expression.

To be able to make individual AUs, one must be a trained individual. It is hard to do some individual AUs voluntarily without interference from other face features. For example, it is necessary to gain control of the brow muscles to be able to raise the outer brow independently from the inner brow voluntarily. As a consequence, the individuals that make and evaluate the expressions in the datasets used for testing (The Extended Cohn-Kanade Dataset [22]) are trained to be able to make and distinguish between them in detail.

Mapping AUs into facial expressions is other issue. Complex facial expressions like *Sadness* are represented by one or more combinations of AU. *Sadness* can be represented as "AU1+4+15 or AU1+4+16" [22]. The EMFACS (Emotion FACS) system [21] was developed by the developers of FACS to map AU combinations into emotions. In Table 2 we show some AU combinations and the associated emotion. EMFACS was built under the assumption that a facial expression always conveys how the person is feeling. In this thesis we will share that assumption.

Upper Face Action Units					
AU 1	AU 2	AU 4	AU 5	AU 6	AU 7
					
Inner Brow Raiser	Outer Brow Raiser	Brow Lowerer	Upper Lid Raiser	Cheek Raiser	Lid Tightener
*AU 41	*AU 42	*AU 43	AU 44	AU 45	AU 46
					
Lid Droop	Slit	Eyes Closed	Squint	Blink	Wink
Lower Face Action Units					
AU 9	AU 10	AU 11	AU 12	AU 13	AU 14
					
Nose Wrinkler	Upper Lip Raiser	Nasolabial Deepener	Lip Corner Puller	Cheek Puffer	Dimpler
AU 15	AU 16	AU 17	AU 18	AU 20	AU 22
					
Lip Corner Depressor	Lower Lip Depressor	Chin Raiser	Lip Puckerer	Lip Stretcher	Lip Funneler
AU 23	AU 24	*AU 25	*AU 26	*AU 27	AU 28
					
Lip Tightener	Lip Pressor	Lips Part	Jaw Drop	Mouth Stretch	Lip Suck

Table 1. Upper and lower face action units. Illustration taken from the work of Tian el al. [18]

Emotion	Action Units
Happiness	6+12
Sadness	1+4+15
Surprise	1+2+5B+26
Fear	1+2+4+5+20+26
Anger	4+5+7+23+24
Disgust	9+15+16
Contempt	R12A+R14A

Table 2. Description of emotions according to facial action units (based on EMFACS [21])

2.2.2 Recognizing facial expressions

A basic structure of a facial expression recognition system was presented by Tian et al. [18] and can be described in three steps: facial acquisition, facial expression features and representation and facial expression recognition. Facial acquisition refers to the process for obtaining face images from an input image or sequence. One example that can be used to detect faces is the Viola and Jones' face detector already explained in Section 2.1.1. Other possibility would be to manually choose the face areas in an image. There are two main approaches for facial feature extraction: geometrical-based and appearance-based. Geometrical based methods rely on the shape and position of facial components like the nose, mouth, eyes and brows to create feature vectors. Appearance based methods apply contour detection filters like Gabor wavelets to the face image to extract features.

After having a method for representation, it is necessary to determine if the detected face matches can be identified as a known facial expression. Recognition can be based on two types of temporal approaches: frame-based (analyse faces individually for expression without any temporal information taken into account) and sequence-based (use several frames and their temporal information to recognize the expression).

Face detection and face orientation estimation

Face detection is the first step for facial expression recognition. The approaches taken select to mark manually the faces on an image (requires human intervention) or automatic detection using an algorithm like Viola and Jones' face detection algorithm [9].

After detection, it is necessary to estimate head orientation (to correct any rotation or yaw deviations) and align the image. Most systems require key features (i.e. eyes) to fall on a consistent location across images. A possible approach is a 3D model. In Bartlett's et al. [20] system, each frame must be fitted into a wire-mesh face using 8 manually selected features (ear lobes, lateral and nasal corners of the eyes, nose tip, and base of the centre upper teeth). In the works of Xiao et al. [24] a generic cylindrical face model is placed upon a detected image and its position is adjusted using detected 2D points. This system is capable of detecting large yaws and rotation without losing track of the face.

These approaches rely on manual input to fit the face to the model. Thus, they are not adequate for an automatic system that runs in real time. We have developed a simple alignment algorithm that uses the location of the eyes to estimate a horizontal rotation angle for the face. The full description is contained in Section 3.2.2.

Facial expressions recognition

After having a face image, it is necessary to extract the features. There are two feature types: geometric features and appearance based features. The geometric features are based on the location and shape of facial components (mouth, nose, brows and eyes). Appearance features [20] represent the contours of facial components and wrinkles. There is also the possibility of a hybrid approach that combines both features types [25].

Tian et al. [25] used geometric features to detect both permanent (i.e., lips, eyes, brows, and cheeks) and transient components (i.e., furrows). These components were used to define different states for lips (open, closed, and tightly closed), eyes (open or closed), brow and cheek and transient facial features, such as nasolabial furrows (present or absent). The features approximate position was detected on the first frame and (if necessary) corrected by the user. The system was capable of tracking the points automatically across a sequence of images. For face expression recognition, their technique was based in three-layer neural networks with one hidden layer to recognize AUs by a standard back-propagation method. They achieved a recognition rate of 93.2% with a false alarm (detection of extra AUs) rate of 2% when searching for expressions in

images containing single AUs (AU 1, AU 2, AU 4, AU 5, AU 6, and AU 7) or AU combinations such as AU 1+2, AU 1+2+4, and AU 6+7. Moriyama et al. [26] used geometrical features to detect eye key points automatically. They used the method described by Xiao et al. [24] to track the face across a sequence of images. After face tracking, the system processes the key eyes points (corners of the eyes and the middle point) to detect whether the eye is opened or closed. The system was capable of detecting eye blinks (AU 45) over time with an accuracy of 98%.

Geometrical features require the manual intervention to work. Gabor wavelets are more adequate on a system where minimal user input is key, as they don't require manual intervention. In image processing, Gabor wavelets are contour detection filters widely used for facial expression recognition. A detailed description is present in Section 4.2.2.

Tian et al. [27] made experiments with Gabor wavelets. They used Gabor filters in selected parts of the image (instead of the whole face image) and compared the results with the ones obtained in their previous study using geometric features [25]. The AUs tested were AU 41, AU 42, and AU 43 (lid dropping, slid and eyes closed). The recognition rate of these single AUs using a three-layer neural networks was 83%. When testing for other AUs, the recognition results were much worse (32% recognition rate with a false alarm rate of 32.6%), having only adequate results for AU6, AU43, and AU0 (neutral face). As a comparison, the recognition results for those AUs using geometric features were 87.6%. Combining the results from both feature types (hybrid approach), the recognition result was 92.7%, suggesting that the combination of both approaches can improve the results.

Face expression recognition is the final step: the extracted features are assigned an expression. All the studied approaches relied on a machine learning techniques. Support Vector Machines (SVM) are one of the most used approaches. A model is created from training data, through the division of feature space in the partitions that better separate the negative and positive examples.

In the basic algorithm, the divisions between examples must be linear and the algorithm only supports discrimination between positive and negative examples (two states). Several extensions were developed, allowing for other type of functions to divide spaces (i.e. quadratic) and multiclass SVM (K-SVM), that allows for a finite set of labels instead of only positive and negative labelling.

SVMs can be used to recognize expressions by themselves or in conjunction with other techniques (e.g. Hidden Markov models). Bayesian Networks can also be used for recognition either individually or in conjunction with Hidden Markov models [20].

Littlewort et al. [28] used appearance based features with Gabor wavelets in automatic facial expression (*Neutral, Anger, Disgust, Fear, Joy, Sadness, Surprise*)

recognition in real-time video. They detected faces using an AdaBoost cascade (similar to Viola and Jones'), extracted the facial features using Gabor filters and combined two approaches for recognition: multiclass decisions using SVMs and AdaBoost. They achieved good results with 91.5% recognition rate for SVMs, but the results were not adequate for real time usage. Using AdaBoost to reduce the amount of features, they achieved a 89.7% correct recognition rate and increased the speed, enabling the analysis of several hundreds of images per second.

2.3 Affective interaction

Human computer interaction is changing in new and exciting ways. In the words of Picard, affective computing, is "computing that relates to, arises from or deliberately influences emotions" [29]. Affective interaction is the area of affective computing related to input techniques. It encompasses anything from natural techniques like gestures, facial expression to advanced techniques like the interpretation of electrical signals from the brain [30] (Electroencephalography). In this Section we present some applications of affective interaction, focused mainly in facial expression and games.

Affective interaction is being applied in games. SOEmote [1] is the facial expression detection component for Everquest 2 that enables the game character to mimic the player's facial expression. The game detects the position of several facial features using a webcam and maps them into the face of the character. SOEmote also features a voice modulator integrated in the game's voice chat, allowing the player to alter its voice tone to better match the virtual character. Everquest 2 is a fantasy massively multiplayer online role playing game where players are encouraged to cooperate to defeat their enemies. SOEmotion increases immersion by placing the players' affective state in the game character. Reception by the players was mixed [31], [32]: some players praised it for the innovation while others argued that it did not add anything to the game experience.

Orvalho et. al. developed "What a Feeling", a game based on the recognition of facial expressions by the players. It was designed to help children with Autism Spectrum Disorders (ASD) to improve their ability to recognize and respond to emotions. The users must try to recognize facial expressions performed by an avatar, and the game was the ability to add new ones in real time. The game avatar is based on a facial rig that can represent *Anger*, *Disgust*, *Fear*, *Joy*, *Sadness* and *Surprise.* The therapist can manipulate the rigs using a mouse and a keyboard to create new expressions.

Facial expression is not the only affective measure incorporated in games. Paiva et al. [4] developed SenToy, a doll that recognizes emotion using motion detection sensors. Different gestures lead to the following emotions: *Anger*, *Fear*, *Surprise*, *Sadness*, *Gloating*

and *Happiness*. The doll was evaluated in two trials: in the first trial (similar to a training stage), volunteers were asked to control the emotions of a virtual character by interacting with SenToy. In the trial, Paiva et al. collected a set of actions for the emotions; for example, *Anger* was expressed by boxing with SenToy arms. In the second trial, the researchers tested the detected gestures from the first trial in a game. Users were asked to represent emotions with the doll, without prior instructions. When manipulating SenToy, the players received textual feedback with the detected emotion. The results showed that SenToy was capable of detecting some emotions very well (*Happiness*, *Sadness* and *Anger*). The remaining emotions performed worse, either because of lack of tuning in the motion detection (i.e. *Surprise*) or because the players were performing an incorrect or unexpected gesture (i.e. *Gloating* and *Fear*). The users response towards the doll was very positive, and the authors argue that affect based input methods can be effective in games.

Figure 7. "SenToy" doll. Image from [4]

Figure 8. "SOEmote" example. Image taken from the demonstration video [4]

"Emote to Win" [33] is a game where the player interacts with a Tiffany, a virtual pet snail using speech and bio signals (skin conductivity, heart rate, respiration and muscle activity). The features extracted were mapped in the Valence/Arousal scale and classified into an emotional state using a simple threshold based classifier. Tiffany responds to user actions like a pet. (i.e. if the player shouts at Tiffany, she will hide behind a bush). It is also possible for players to respond emotionally to Tiffany. For example, the player may get irritated if Tiffany refuses to appear. The authors argue that integration with Tiffany was natural, but the response of the character was not always clear to players.

2.3.1 Eliciting emotions

Psychologists use images [34] to study facial expressions and emotional response on people. International affective picture system (IAPS) [35] is a database of pictures used in the medical community to study emotional response in people. It was built by showing various images to people and measuring their emotional response.

[4] http://uk.ign.com/videos/2012/06/01/everquest-ii-soemote-tech-demo

Novel techniques are being developed to evoke emotions on people using innovative media. Wang and Marsella [36] developed a video game called Emotion Evoking Game (EEG), designed to provoke emotions on the player. The game was created to aid the development of systems that analyse emotion and facial expression. They made a small study that consisted on having a small pool of volunteers that played the version designed to provoke four different emotions (*Boredom, Surprise, Joy* and *Anger*) at specific stages throughout the duration of the gamming session. The player's face was being captured with a webcam and they were asked to fill a form at the beginning and end of the game regarding their emotional state at the key moments of the game. The video from the camera was analysed by the researchers and compared with the answers to the forms. The results were not consistent amongst emotions, producing some unexpected emotions to the programed events in the game. Brooks et al. [37] developed "Leo", a robot designed for social interaction. Leo is capable to perform and learn facial expressions and poses by natural interaction with humans. The Mimic Game [3] shows an interesting application to facial expression recognition: a 3D computerized agent mimics the facial expression and head pose of a person from a real-time video feed. The system detects the facial components and head pose using a calibrated Active Appearance Models (AAMs) [38] and maps the detected mesh into a two-dimensional emotion space. A point in this space corresponds to an expression (from EMFACS) / intensity (from neutral face to full scale expression) pair. The computerized agent receives the information regarding what is the expression and intensity and mimics the expression.

2.4 Metrics

In this Section, we describe the metrics used to evaluate our algorithms.

Face detection and alignment

The basic measures for facial detection are:

- True Positives (TP): faces correctly detected;
- True Negatives (TN): objects correctly detected as not being faces;
- False Positives (FP): objects incorrectly detected as faces;
- False Negatives (FN): faces that were not detected.

For a quick performance measurement, we use the *correct detection rate*, defined as:

$$correct\ detection\ rate = \frac{true\ positives}{total\ number\ of\ faces}$$

$$accuracy = \frac{true\ positives + \ true\ negatives}{true\ positives + \ true\ negatives + false\ positives + false\ negatives}$$

$$precision = \frac{true\ positives}{positives} = \frac{true\ positives}{true\ positives + false\ positives}$$

$Accuracy$ represents how close the algorithm is to the true value. $Precision$ represents the repeatability of the algorithm: degree to which repeated measurements show the same results over different experiments under the same conditions.

Face recognition

It is important to define the basic measures first, which work in a similar fashion to the face detection measures .Some measures used for face detection will also be used for face recognition:

- True Positives (TP): faces correctly recognized;
- True Negatives (TN): faces whose distance was above the threshold and the closest face corresponds to a different person;
- False Positives (FP): faces incorrectly labelled;
- False Negatives (FN): faces whose distance was above the threshold but the closest face corresponds to the same person.

To tune the face recognition algorithm we use the $FP/TP\ rate$:

$$FP/TP\ rate = \frac{false\ positives}{true\ positives}$$

Face recognition offers an important challenge regarding the balance between false positives and correct recognitions. The recognition of a person is accomplished by measuring the distances between a face and the ones already existing in the database. This distance must have an upper limit, above which the person will be classified as unknown. For more details about how this threshold is calculated, see Section 2.1.2.

Taking into account the threshold, other metrics can be observed, such as the $unknown\ rate$, which consists on the rate of people that the algorithm was unable to recognize (distance to the closest face higher than the upper limit):

$$unknown\ rate = \frac{false\ negatives + true\ negatives}{total\ number\ of\ faces}$$

Receiver operating characteristic curve (ROC curve) is a graphical plot that represents the performance of a binary classifier, as the threshold varies. It is obtained by plotting the $true\ positive\ rate = (true\ positives)/positives$ vs. $false\ positive\ rate = (false\ positives)/positives$. The ROC curve is useful to optimize the threshold value, as it takes into account the balance between false positives and true positives.

Facial expression recognition

In our facial expression recognition algorithm, all faces are classified with an

expression; we do not consider negative examples (expressionless faces). Therefore, the only relevant metrics are:

- True Positives (TP): facial expression correctly recognized;
- False Positives (FP): facial expression incorrectly recognized.

Working only with positive examples, the relevant metric is *precision*.

For facial expression recognition, confusion matrices are a key metric. A confusion matrix is a $K \times K$matrix, where K is the number of classes of the classifier (in our case, the seven facial expressions). In this representation, the actual class of the sample is represented by the column and the predicted class of the sample is represented by the row (Figure 9). The confusion matrix is useful to easily visualize the performance of the algorithm and the misclassifications between classes. True positives are located in the diagonal of the matrix. Therefore, it's easy to visualise errors, as they are represented by non-zero values outside the confusion matrix diagonal.

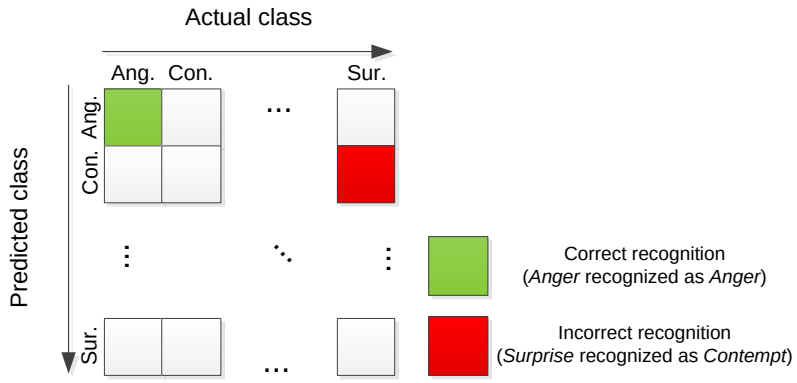

Figure 9. Example of a confusion matrix

2.5 Summary

In this Chapter, we reviewed key techniques to detect and recognize faces and analyse their facial expressions. The Viola and Jones' face detector and the eigenfaces play a fundamental role in the implemented framework.

The analysis of facial expressions is a key part of this thesis. Gabor wavelets were implemented, as they are state of the art in automatic facial expression analysis. Besides the algorithmic techniques, we also looked into the psychological and emotional aspects of facial analysis.

This leads us to the area of affective interaction. We have found that, as far as we know, there is not much research in the utilization of affective features in videogames. We discussed the approaches we deemed as being more relevant in affective based interaction.

3

Face recognition and tracking

3.1 Introduction

In this Chapter, we focus on the face recognition component of the framework introduced in Chapter 1. This component deals with the detection and recognition of who is in the living room ready to interact. It allows the recognition of people that already have a profile and the addition of unknown individuals in real time, just by capturing faces and (optionally)asking their name. Taking into account the limitations of the living room, one of the main design decisions was the minimization of user input. We only require users to input information in situations where we could not find any alternatives. The automatic recognition of who is in the living room eliminates the burden of having to manually identify everyone, thus, enabling a smoother and pervasive experience.

The face recognition component is designed to handle a real use case with multiple users. The process starts with the creation of baseline face models from previously acquired training data (**offline training phase**). In live mode, the system will capture faces of the actual users and match them to the known face models or add the unrecognized ones to the face models (**online training phase**).

Figure 10 illustrates a general view of the face recognition process. In the first step (0), we create the face models from a set of training images (offline training phase). In live mode, images acquired from a camera (1) are passed to the face detector (2) which will search the image for faces. Before running the face recognition algorithm, the detected faces must be pre-processed (3) and, for a matter of efficiency, we track the face of the detected user (4) and discard low quality faces (5). The eigenface algorithm is run to identify a named person (6). When the face is not known, the user is asked to enter her name and this data is then used to update the face models (9). This closes the loop of the face recognition process by adding new user profiles to the face dataset.

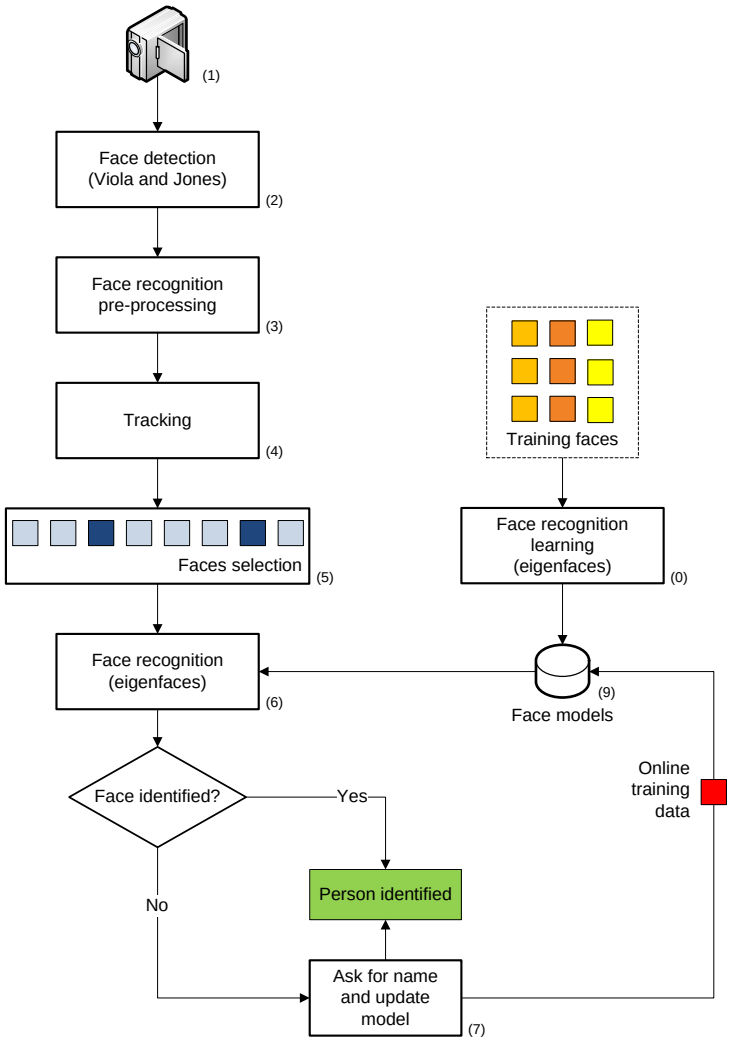

Figure 10. Detailed scheme for the online face recognition component

This framework was implemented with the OpenCV library[5], which aimed at real time computer vision. This library offers functions related to image acquisition, automatic face detection using the Viola and Jones' approach described in Chapter 2, Principal Component Analysis (essential to the eigenfaces face recognition algorithm) and nearest neighbour search with FLANN (Fast Library for Approximate Nearest Neighbors)[6].

3.2 Face detection and pre-processing

Face detection and pre-processing detects faces in images and prepares them for face recognition and facial expression recognition. The process is illustrated in Figure 11.

[5] http://opencv.willowgarage.com/wiki/
[6] http://www.cs.ubc.ca/~mariusm/index.php/FLANN/FLANN

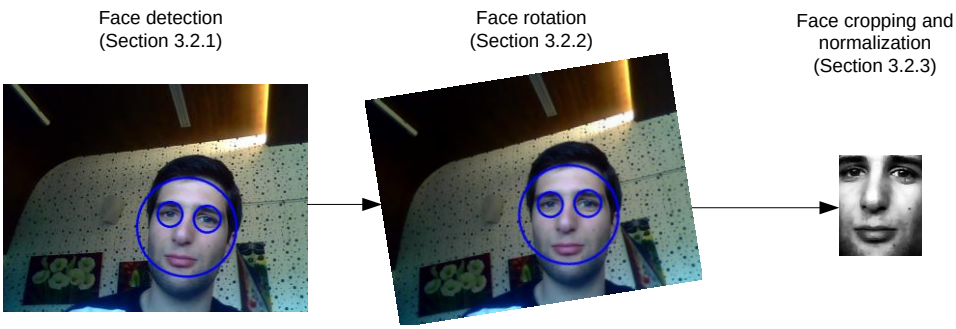

Figure 11. Face detection and pre-processing

The first image contains the detection of a face. The second image contains the face after eye detection and rotation and the third image contains the face after cropping and histogram normalization. The following Sections will detail these steps.

3.2.1 Face detection

For face detection, we use the C++ version of the OpenCV framework to implement the ideas proposed by Viola and Jones. The face detection process is described in detail in the Section 2.1.1. Initially, the algorithm loads the training data offered by the OpenCV framework. This training data consists of an XML file containing the location and values of the Haar like features. The framework offers a varied set of training data, allowing for simple detection of various body parts or face components. The algorithm scans the captured image, identifies the areas where there is a high probability of containing a face and returns a vector containing the face coordinates.

3.2.2 Face rotation

Alignment and rotation are crucial because face recognition with eigenfaces needs a correct face registration (facial features (eyes, lips) on the same reference positions). Thus, the cropped images containing the faces must go through an eye detection phase to ensure a proper alignment. This alignment assumes that eyes are on the same horizontal line.

Since the eye detection algorithm does not search for an eye pair, but for individual eye positions, we end up with multiple eye position candidates. The problem is better illustrated in Figure 12: the algorithm detected two incorrect eye positions along with the two correct ones. To solve this problem, the image is divided into four quadrants. The correct eye positions will be in the top two quadrants. When multiple eyes are detected in the same quadrant, the smallest one is selected. Figure 13 illustrates the key points behind the image rotation process. The image is rotated so that the points (x_1, x_2) and (y_1, y_2) stay in the same horizontal line.

If the algorithm is unable to detect both eyes (no candidates in one of the top

quadrants), the face image is rejected. This decision was made to ensure that only the best quality images are collected.

Figure 12. Multiple eye detection

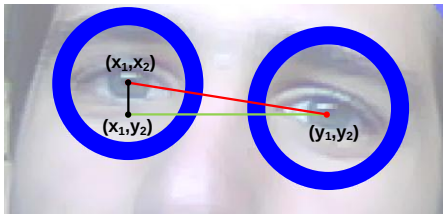

Figure 13. Preparation for image rotation

3.2.3 Face cropping and normalization

Once the face image is detected and aligned, it is necessary to crop it to the face area. The face detection algorithm does not provide an exact measurement of the face; it only returns an approximated radius. So, we decided to use the distance between the eyes as a reference scale and apply a template to crop the face pixels. The template was determined experimentally and is presented in Figure 14.

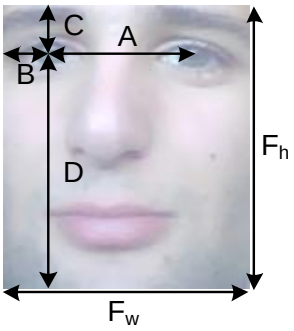

Figure 14. Face proportions

Legend:

F_w: Face width (9/5 **A**);

F_h: Face height (28/23 **F_w**);

A: The distance between the eyes

B: The distance between the eye and the vertical limit of the face (2/9 **F_w**);

C: The distance between the eye and the top limit of the face (1/4 **F_h**);

D: The distance between the eye and the bottom limit of the face (3/4 **F_h**).

Finally, we equalize the face histogram to improve the quality of images. The equalization algorithm adjusts the brightness and increases the contrast of the image to normalize images captured under different lighting conditions. For example, a greyscale image with brightness values ranging from (56-200), will be redistributed through the entire grayscale range (0-255).

As one can see in Figure 15, even after the pre-processing of the faces images, the differences between the faces of the same person are visible (i.e. alignment, scale and histogram equalization). The recognition algorithm performs better with frontal faces and aligned as the rightmost image of Figure 15.

Figure 15. Examples of captured faces after the pre-processing

3.3 Face tracking

To reduce the computational complexity of the face detection and recognition steps, we track faces across a sequence of images. It works by defining a search region where the person's face is most likely to be by examining the face's position in previous frames. The region is defined as a circle, where the centre is the last detected face centre and the radius is relative to the size of the captured image. If the face is within the search region, the user profile of previous frames will be kept for the current frame. This reduces the search space for the face detector and avoids the face recognition step.

Algorithm 1. Tracking a face across images

```
Input:          faceCenter: point in the image where face was detected
                faceImage: image of the detected face
                trackedFaces: index of tracked faces
                radius: maximum distance for tracking

Output:         trackingProfile: tracking profile

    distance <- MAX_VALUE

1.  for each profile in trackedFaces
2.      tempDistance <- L2_norm(profile.lastFaceCentre,faceCenter)
3.      if tempDistance < distance AND tempDistance < radius
            distance <- tempDistance
            trackingProfile <- profile
        endif
    endfor


4.  if trackingProfile is empty
        trackingProfile <- new trackedPerson()
    endif

5.  trackingProfile.addLocation(faceCenter)
    trackingProfile.addImage(faceImage)
```

Algorithm 1 describes the face-tracking algorithm. Every time a face is detected, we check if the centre of the detected face is inside the search region of any of the currently tracked users (Algorithm 1: step 1). A profile is retrieved if the L2 distance (Algorithm 1: step 2) is smaller than the tracking *radius* and if it is the closest tracking profile

(Algorithm 1: step 3).

If this correspondence does not exist (Algorithm 1: step 4), a new tracking profile is created. The face coordinates and the face image are then added to the corresponding tracking profile (Algorithm 1: step 5). To avoid potential errors in tracking, we compare the existing model to the training data after a certain number of images.

Figure 16. Face tracking path example

Figure 16 illustrates a face tracking example. The green curve is the trajectory of the face and the blue circle is the face tracking area (defined by *radius*) on the current image frame. Several image faces of this user were captured along that trajectory. Although tracking alone can indicate if a person is the same across a sequence of images, it must be combined with recognition to achieve our main purpose: to assign a name to that face and store this information for later usage.

3.4 Face recognition

Face recognition compares the captured face to the existing face models to determine if a detected face corresponds to a known person or if it corresponds to an unseen person. The face recognition process is divided into four steps: the offline learning step to compute the face models; the live recognition of detected faces; parameter tuning for live (real time) recognition; and the online update of the face models with new unseen faces.

3.4.1 Learning face models offline

The face recognition process relies on the eigenfaces [12]. Initially, an eigenfaces space is calculated from the training set of faces. This initial set of faces is not necessarily

related to the faces that will be captured in the future. The set is necessary to calculate the underlying data structure of face images (i.e., the eigenvectors of a matrix of faces). Our set is composed of 200 images from the "Our Database of Faces" [17].

After we have the training data according to the conditions described in Section 2.1.2 we must calculate the corresponding eigenfaces representation. The OpenCV library offers a PCA implementation to create the face space and project new faces into the created feature space (i.e. using the eigenvectors and eigenvalues).

The details of the PCA process and eigenfaces are described in detail in the Section 2.1.2. This Chapter will focus the implementation details.

Algorithm 2 describes the creation of the eigenfaces space. In the first training step, the training images are opened and projected into the face space using the PCA algorithm (Algorithm 2: step 1) to determine the faces eigenvectors. These projections are saved in a *k*-NN (*k* nearest neighbours) index for future recognition (Algorithm 2: step 2).

Algorithm 2. Creation of face space and search index.

```
Input:          f: set of face images

Output:         searchIndex: k-NN search index for face features
                eigenFaceSpace: eigenface space for face projection

    columnFaces <- transformFacesToColumns(f)

1.  faceSpace <- createEigenFacesSpace(columnFaces)

    searchIndex <- createNewKNNIndex()

    for each faceImage in the set f
       projectedFace <- faceSpace.projectInEigenFaceSpace(faceImage)
2.     searchIndex.addToIndex(projectedFace)
    endfor
```

The *k*-NN index provided by FLANN allows K nearest neighbours search with several automatically tuned indexing methods. This feature is the basis for the recognition of new faces. Figure 17 illustrates a simplified view of the feature space with only two features (*eigenface 1* and *eigenface 2*), with two users in the database (*Alice* and *Bob*) and two new detections (an *unknown user* and an *ambiguous user*).

3.4.2 Live face recognition

When the framework is running live (in real time), detection and tracking capture the best quality images for face recognition. The PCA projection transforms the image into face space features resulting in an unlabelled feature vector. To check if the face corresponds to a person already in the training data, this feature vector is compared to

the training faces (both captured faces and the trained data) by measuring the distance[7] between each of the training faces and the new projected face using the K-NN index. We identify the new feature vector according to the nearest vector from the set of labelled feature vectors (each vector corresponds to a known person). However, this distance might be too high resulting in a decision with a low confidence value. Thus, we define a threshold to decide the assignment of a label to new faces. Figure 17 shows a simplified representation of a face space, containing only 2 training faces (*Alice* and *Bob*) and 2 features (*feature 1* and *feature 2*). The *unknown face* is above the threshold for both *Alice* and *Bob*: it will be considered a new user.

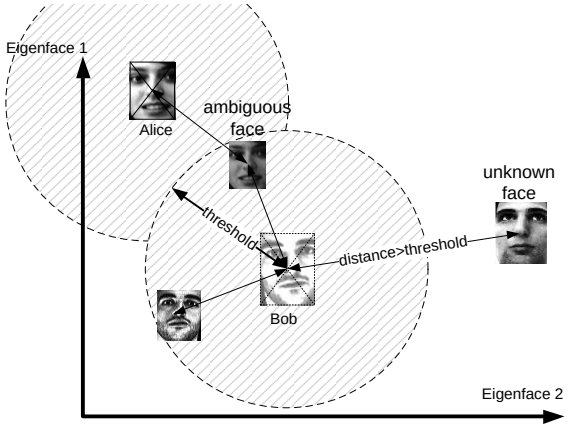

Figure 17. Simplified representation of the face feature space

Figure 17 also illustrates an ambiguous situation. The *ambiguous face* is below the threshold to both *Alice* and *Bob* leading to a situation of ambiguity. There are two possible solutions for solving the problem: we can associate the detection with the smallest minimum distance. Another possibility is to ask the user to remove the ambiguity by assigning a name to the face manually. The first solution is invisible for the user but can lead to errors. The second solution guarantees that the person is correctly identified, but requires for manual user interaction, which can be undesirable.

3.4.3 *k*-NN decision thresholds

The average distance between faces depends of the images used for training. Before we calculate the face space and the nearest neighbour index with the full training face set, it is necessary to estimate the decision threshold to determine if a face is recognised or marked as unknown. A too low threshold will discard several correct recognitions (leading to high false negatives rates), while a too high threshold will increase the rate of false positives. False positives must be avoided, because classifying *Bob* as *Alice* leads to a privacy breach, granting *Bob* access to information owned by *Alice*. Therefore, a strong

[7] The distance used currently is the Euclidean distance

requirement is to have a threshold high enough to reject the majority of false positives, at the cost of an increase in the false negatives rate. To calculate the threshold, we implemented a greedy-search strategy presented in Algorithm 3.

Algorithm 3. Calculating the threshold.

```
Input:          eigenFaceSpace: eigenface space for face projection
                searchIndex: K-NN search index for face features
                exponentIncrement: initial increment for the threshold exponent
                exponent: initial threshold exponent
                validationSet: images for algorithm validation

Output:         threshold to accept or reject recognitions

   threshold <- STARTING_THRESHOLD
   exponent <- STARTING_EXPONENT
   exponentIncrement <- STARTING_EXPONENT_INCREMENT

   for each iteration

      do
1.       exponent <- exponent + exponentIncrement
         threshold <- 2^exponent

         for each (face f, person correctPerson) in validationSet

2.          featureVector  <- EigenFaceSpace.projectInFaceSpace(f)
            searchIndex.search(featureVector, detectedPerson, distance)

3.          if (detectedPerson != correctPerson AND threshold < distance)
              trueNegativesCount++
            else if (detectedPerson != correctPerson AND threshold > distance)
              falsePositiveCount++
            else if (detectedPerson == correctPerson AND threshold < distance)
              falseNegativeCount++
            else if (detectedPerson == correctPerson AND threshold > distance)
              truePositiveCount++

          endfor
4.    while falsePositiveCount/truePositiveCount > 0.02

5.    exponent <- exponent - exponentIncrement
      exponentIncrement <- exponentIncrement * 0.9

   endfor
```

The experiment was designed to determine the ideal threshold value, using a cross-validation method. The training data is divided into three smaller sets of the same size. These smaller sets are then grouped by using two of them for training and the other for validation. All the permutations are used to calculate this threshold and their average value is chosen to ensure a robust estimate. The validation and threshold estimation process (Algorithm 3) is the following: in each round, the faces in the validation set are projected into the face space and then the closest match is found in the training set (Algorithm 3: step 2). In step 3, the algorithm checks if the face returned in the K-NN index is correct and if the distance is below or above the threshold and increases the

corresponding measure.

After looping over all test samples, the algorithm checks if the stopping criterion was met (Algorithm 3: step 4). If not, the threshold is increased exponentially (Algorithm 3: step 1) and the algorithm tests the new value for all test samples. If the stopping criterion was met, that means we increased the threshold above the optimal value. Thus, the threshold is rolled back to the value of the previous iteration and the increment that is applied to the exponent is decreased (Algorithm 3: step 5). The stopping criterion is defined by the ratio between false positives and true positives. For example, if we wish this ration to be 0.02, the criterion corresponds to

$$\frac{false\ positives}{true\ positives} > 0.02.$$

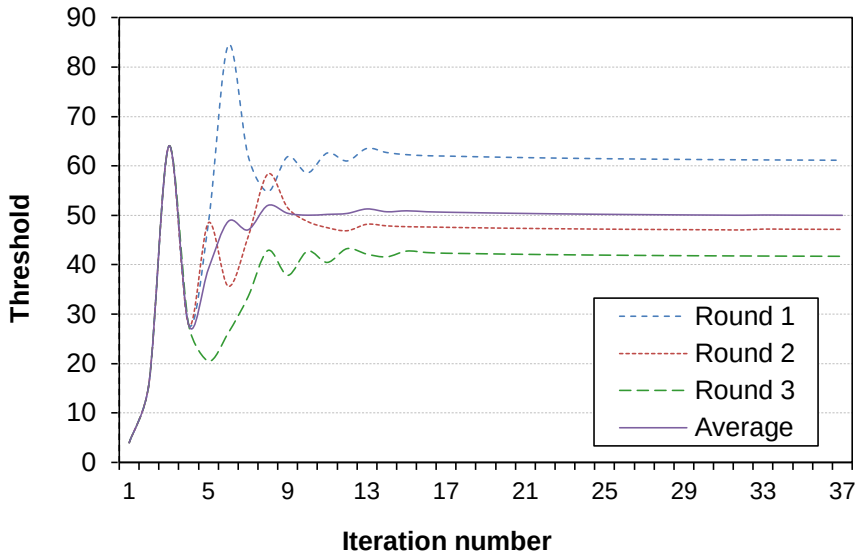

Figure 18. Threshold calculation example using exponential backoff

An example of the threshold evolution over 38 iterations is presented in Figure 18. There is a clear trend for the threshold values to converge towards a small range of values. The threshold values are presented in the Section 3.5.3.

3.4.4 Updating the face models

Face tracking detects position of faces across a sequence of images. It is necessary for the creation of sets of face images of single individuals that can be added in real time to the face models. To support the addition of new faces, the training data cannot be seen as a static set of faces. In our case, the dynamic nature of the training data bears a significant toll in keeping the eigenface space updated with the new faces while the system is running.

Figure 19. Face tracking and posterior recognition

Figure 19 contains a person who was not recognized in the training data and is being tracked (represented by a generated name *unk100*, Figure 19 left-side). After taking a certain number images, the captured faces of the new person are projected into the face space and stored in the *k*-NN index for posterior recognition (similar to the step 2 in Algorithm 2). In this example, the user was already present in the index as *Andre* (Figure 19 right-side).

3.5 Evaluation

In this Section, we evaluate the performance of the face detection and face recognition components with datasets available to the scientific community. The performance evaluation is divided into: face detection and pre-processing (Section 3.5.2) and face recognition (Section 3.5.3). Face detection and pre-processing tests evaluate the performance of the detection algorithm with the "Frontal face dataset". Face recognition tests evaluate the performance of the recognition algorithm using the "Our Database of Faces" face images dataset.

3.5.1 Data

The database used for the evaluation of the face detection component was the "Frontal face dataset", collected by Markus Weber at California Institute of Technology [39]. Sample images are in Figure 20. It consists of 447 pictures of 27 distinct subjects with a variable number of images per subject. The faces were taken from a front facing perspective at different times, with varying lighting and positions, and with different backgrounds. The face files are in JPG format, with a resolution of 896x592 and 24-bit colour bit depth. This database is adequate for face detection, because the faces are in different scales, background environments, conditions similar to the ones expected in

real world usage.

Figure 20. Sample images from the "Frontal face dataset" database.

The database used for evaluation of the face recognition algorithms was the "Our Database of Faces", created by the Olivetti Research Laboratory in Cambridge [17][40]. This database consists of 10 different pictures of 40 distinct subjects (400 face images). The face pictures were taken from a frontal perspective in different days, with varying lighting and some facial expression variation. The faces files are in the PGM format, with a resolution of 92x112 pixels and 8-bit grey levels. The database is property of AT&T Laboratories Cambridge; some samples are shown in Figure 21.

Figure 21. Sample images from the "Our Database of Faces"

3.5.2 Experiments and results: Face detection and alignment

The methodology for evaluating face detection is the following: the images go through the face detection algorithm and metrics are collected. After detection, face images go through pre-processing (see Section 3.2.2 and 3.2.3 for details) and the metrics are calculated again, for performance comparison.

The experiment for face detection was made using the Caltech faces database. The full face set composed of 447 photos containing one person per photo was chosen as testing data. Initially, the face detection algorithm (see Section 3.2.1) is executed for all the images and the measurements are taken. Then, the face pre-processing (see Section 3.2.2 and 3.2.3) is executed for all the faces detected in face detection and the new measurements are taken. The results are present in Table 3.

The face detection algorithm is able to detect more than 99% of the faces in the images at cost of 463 non-face objects being also detected (48.90% accuracy). These values of

accuracy and precision mean that the algorithm cannot be used in its current state.

	TP	TN	FP	FN	Accuracy	Precision	Correct detection rate
Face detection (FD)	445	0	463	2	48.90%	49.01%	99.55%
FD and pre-processing	433	461	2	14	98.24%	99.54%	96.87%

Table 3. Face detection results

Fortunately, the pre-processing stage is able to correctly classify those non face objects as true negatives, increasing precision to 99.54% and accuracy to 98.24%. This lowering of the false positive rate didn't affect the detection rate significantly, lowering it only by less than 3% (96.87%).

The results obtained are very positive. The performance of the algorithm with the controlled dataset yields an accuracy of 98.24%, with a minimal amount of false positives. The pre-processing stage is very important, as it nearly eliminates all false positives, improving the quality of the captured images at the same time.

3.5.3 Experiments and results: Face recognition

For face recognition, the testing methodology is the following: the "Our Database of Faces" dataset is shuffled and divided in two parts: training (70% of 400 = 280) and testing (30% of 400 = 120). The training data was further divided into three sets of faces for validation. The validation step described in Algorithm 3.

The number of eigenfaces chosen for recognition, which is equal to the number of features, was 30, in line with the state of the art [12], [13].

Our objective is to validate the training data against itself, by dividing it into three parts and using two of them for training and the other for validation (see Section 3.4.1 for the full details). The validation process returns the individual threshold values (Table 4). After validation, the testing process (similar to Algorithm 3 but with a fixed threshold) is executed with the test data and the average threshold from the rounds (results in Table 5).

Because the value of the threshold for the training data is unknown, it was determined experimentally using the exponential threshold method described in Section 3.4.1. The stopping condition tested was FP/TP > 0.02. The results from the three rounds were averaged and the algorithm was executed with different divisions in training to achieve consistent measurements and smooth curves.

	Threshold	Accuracy	Precision	Unknown rate
Round 1:	50.03	84.64%	97.26%	21.81%
Round 2:	45.04	86.43%	97.27%	21.44%
Round 3:	46.80	82.50%	97.31%	20.30%
Averages:	47.29	84.52%	97.28%	21.18%

Table 4. Face recognition results with best thresholds

The results from Table 4 are very positive. The threshold and ratios were almost uniform across the round. The average accuracy is 84.52%, with a precision of 97.28% and unknown rate of 21.18%. The precision was high, at cost of a slightly low accuracy and high amount of unrecognized faces. The validation steps were repeated with the new average threshold (47.29). The results are present in Table 5.

	Threshold	Accuracy	Precision	Unknown rate
Round 1:	47.29	85.00%	97.71%	22.14%
Round 2:	47.29	88.21%	96.92%	18.92%
Round 3:	47.29	86.07%	98.25%	18.21%
Averages:	47.29	86.43%	97.63%	19.76%

Table 5. Face recognition results with average threshold

The average threshold improved all the metrics (compared with the best individual thresholds), meaning that the multiple divisions provide a better view of the data than a single division. The final validation results show an accuracy of 86.43%, with a precision of 97.63%.

	Threshold	Accuracy	Precision	Unknown rate
Test data	47.29	89.17%	97.14%	12.5%

Table 6. Face recognition results for test data with average threshold

Finally, the test data was evaluated, using the computed threshold. The results were similar to the ones obtained with the validation data; accuracy of 89.17% and a precision of 97.14%. The unknown rate (12.5%) was lower than the value from the validation

evaluation (19.76%), because of the larger number of faces in the training data. These values are in line with our objectives: a very small amount of false positives (high precision), at cost of a slightly high number of unrecognized faces (12.5%).

The trade-off between the true positives and false positives is more visible in the ROC curve. (Figure 22). The ROC curve show a great increase until the false positive rate is about 0.1, and then the growth rate decreases significantly. Increasing the threshold increases both the true positive rate and the false positive rate and decreases the unknown rate, meaning that more faces will be recognized (either correctly or incorrectly). We think that our stopping condition was adequate for our training data, because it was able to maintain high precision (97.14%).

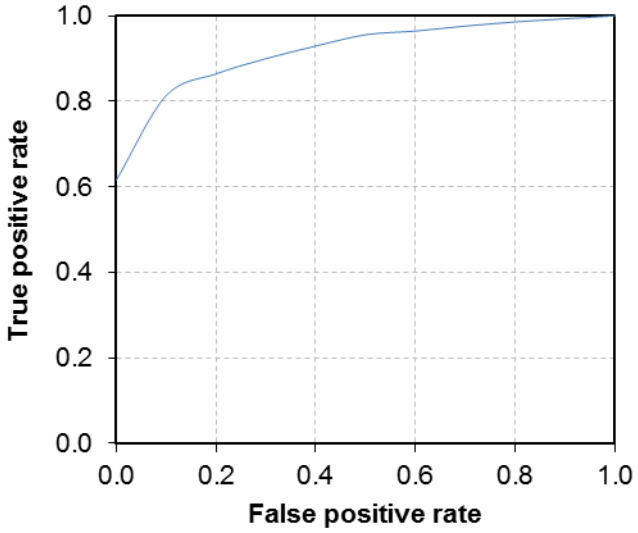

Figure 22. ROC curve for face recognition

3.6 Summary and conclusions

In this Chapter we discussed the implementation details of face detection and recognition using well established techniques. The implementation of face detection uses a set of algorithms based on the ideas of Viola and Jones'. In face pre-processing, we adjust the images of the face to make their alignment and sizes consistent, rejecting the false positives from face detection in the process. Face tracking enables us to detect the same person in a sequence of images. Through tracking, we capture various images of the same person to posterior recognition.

The face recognition component based on eigenfaces is able to recognize people from existing models and allows online insertion of new models. The evaluation of the components was very positive. The face detection algorithm achieved an accuracy of 98.24% and precision of 99.54%. The face recognition algorithm achieved an accuracy of 89.17%, with a precision of 97.14%.

4

Robust facial-expression recognition

4.1 Introduction

Facial expressions are a fundamental part in communication and humans are capable of recognizing the subtle differences between expressions. A fully automatic and robust facial expression recognition system would be useful in areas like security, communication, education and interaction. In this Chapter, we describe our implementation of such system.

4.2 Facial features extraction

A regular camera saves the images as colour pixels, resembling the way humans see the world. But this representation is not adequate for facial expression recognition by computers. Images described by coloured pixels contain a lot of unnecessary information for facial expression recognition; a simpler representation must be created.

The AUs (presented in Section 2.2.1) provide a way to represent facial expression through the position of face components and Gabor wavelets transform images into a contour representation. The contour representation clears the noise of a colour image and leaves the key information regarding the position of face components. The next Section describes how to extract the relevant information from face images for facial expression recognition.

4.2.1 Face detection and alignment

To identify the location of faces in images, we use the face detection algorithm described in Section 2.1.1. The pre-processing techniques applied for face recognition are also valid for facial expression recognition. The full process consists in face detection, rotation, cropping and histogram equalization.

4.2.2 Dictionary of Gabor filters

Using ideas already deployed in the analysis of facial expressions [28], [41], we implemented a dictionary of Gabor filters. Combinations of Gabor filters are widely applied in facial expression recognition because of their ability to detect the contours of the facial components (like eyes, nose, mouth, brows and wrinkles) and filter out most of the noise present in the image [42].

Gabor filters are edge detector filters composed by a two-dimensional wave sign weighted by an exponential decay that can be applied at a given orientation and scale. The formula for a Gabor wavelet is:

$$g(x,y) = \frac{\cos(2\pi W x) \cdot e^{\frac{x^2 - y^2}{2\sigma_x^2 \sigma_y^2}}}{2\pi\sigma_x^2\sigma_y^2}$$

This formula can be considered the "mother" Gabor function. The derivate functions with different scales (m) and orientations (θ) are created using the formula,

$$g_{m\theta}(x,y) = \mathrm{a}^{-m} \cdot g(x',y'),$$

with $x' = \mathrm{a}^{-m} \cdot (x\cos\theta + y\sin\theta)$ and $y' = \mathrm{a}^{-m} \cdot (x\sin\theta + y\cos\theta)$. The orientation of the Wavelet is given by $\theta = n\pi/K$, with K equal to the number of orientations. σ_x and σ_y are the sigmas of the Gaussian envelope and W specifies the ellipticity of the support of the Gabor function. See Manjunath and Ma [43] for a detailed discussion of the parameters. We implemented the filters using the formulas described above and using the Discrete Fourier Transformations present in the FFTW library[8].

This filter is applied to the face image to detect facial traces and contours with a set orientation and scale. Figure 23 depicts a Gabor filter with an orientation of 90º in 3D (left) and 2D (right):

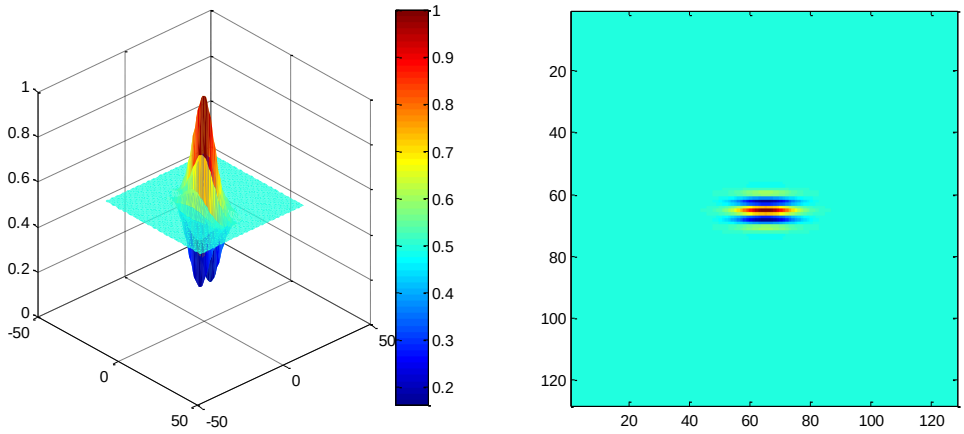

Figure 23. Gabor filter with a 90º orientation

[8] http://www.fftw.org/

Since convolution operations are more efficient in the frequency domain, both face image and Gabor filter are transformed onto the frequency domain representation using a Fast Discrete Fourier Transform [44]. In this domain, a convolution is performed to obtain the filtered image. The inverse Discrete Fourier Transform is applied to the face image to obtain the face contours in the spatial domain. The detected face contours with the orientation of the filter are represented in white. To extract information concerning the face contours and expression traces, several Gabor filters with different orientations and scales capture the different details of a facial expression.

Figure 24 illustrates the dictionary of Gabor filters at four scales and six orientations (0º, 30º, 60º, 90º, 120º and 150º) and the application of all the individual filters on a face image. The image on top is the original face image after face detection; the images on the middle correspond to the filters and the images on the bottom are the combinations of the filters with the face image. In the filtered images, the effectiveness of the different filters when capturing the different face contours is visible. Dictionaries with this configuration have been found to work well on a number of domains, namely, facial expressions recognition [42] and image retrieval [43].

4.2.3 Registration of face AU

To represent facial expressions, we have chosen the Emotion Facial Action Coding System (EMFACS) built on top of the Facial Action Coding System (FACS) [21]. EMFACS is built on the assumption that there are seven facial expressions linked to emotions: *Happiness*, *Sadness*, *Surprise*, *Fear*, *Anger*, *Disgust* and *Contempt* and a state of no expression: *Neutral*.

The previous steps towards a robust representation of facial expressions do not compute features invariant to small variations in face alignment. Since the objective is to recognize facial expressions automatically and without any human supervision, we cannot rely on approaches that manually register the position of facial key-points on an image. Examining the EMFACS data, we propose to overcome this issue with a hard-partitioning of the face image into specific regions. In EMFACS, a facial expression is represented by the position of the various face components (called Actions Units). Different expressions alter the position of face components (mouth wide open and arched eyebrows equal *Surprise*). To classify an expression, it is necessary to estimate the position of the face components and compare them to the existing expression models.

We propose to choose the key areas of the face where more changes occur between facial expressions (i.e. mouth and the brows). We also kept four rectangles that correspond to the full face divided in quarters, to increase robustness to several facial variations. Figure 25 presents an average face (created from the data from CK+) with the rectangular areas highlighted.

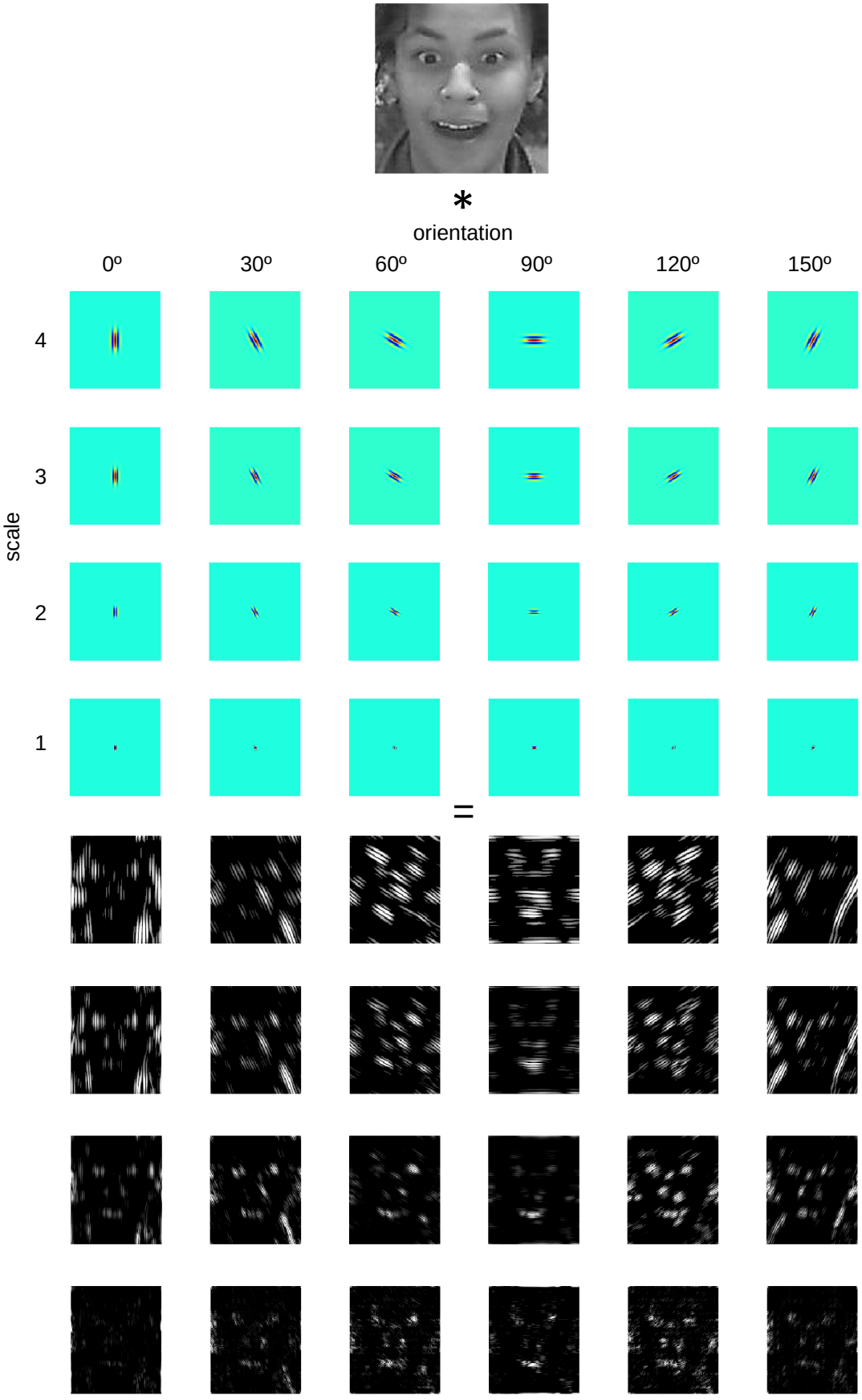

Figure 24. Appling Gabor wavelets to an image

The dictionary of Gabor filters is applied to each one of these regions to obtain the face regions contours. A facial expression is represented by a feature vector where each pair of dimensions corresponds to the mean and variance of each filter output. Since there are six regions and twenty-four filters (four scales, six orientations), the dimensionality of the robust representation is 288. Thus, a facial expression *j* is represented by the vector

$$f_j = (f_{j_1}, \dots, f_{j_{288}}).$$

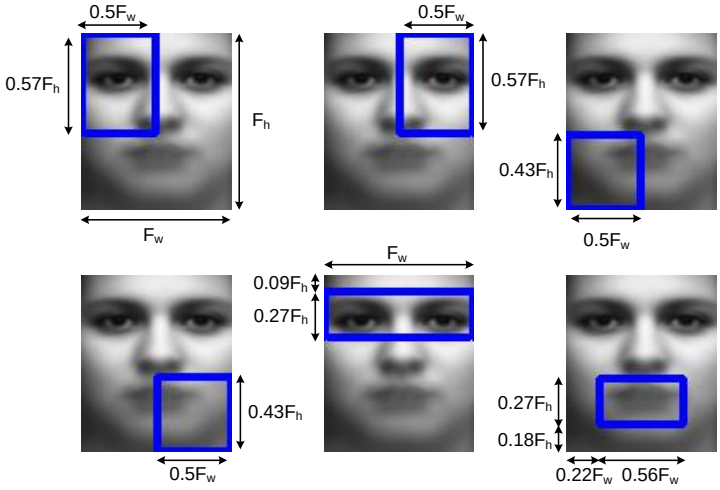

Figure 25. Chosen regions highlighted on an average face

4.3 Dissimilarity between facial expressions

We have also defined a robust metric to measure the dissimilarity between facial expressions. The feature set is still not immune to noise and variations derived from misaligned or rotated images. Therefore, two consecutive images captured from the same individual can have a large amount of features with small differences between them. If the Euclidean distance is used as the distance metric, the sum of the small individual feature distances can show an unrealistically high distance value. It is well known that some L_p norms perform better in such cases. Recently, L_0 and L_1 have become very popular for producing a stable output and being robust to features fluctuations. To compute the similarity between the two feature vectors $pf_a = (pf_{a_1}, \dots, pf_{a_{288}})$ and $pf_b = (pf_{b_1}, \dots, pf_{b_{288}})$ and the facial expression vector, we use the L_0 norm:

$$\Delta_j = L_0(pf_a, pf_b)$$

L_0 is particularly interesting since it measures the number of dimensions in a vector that are different from zero, independently of its magnitude. The L_0 norm allows small differences between features to be discarded and only counts the dissimilarity if it is

above a certain threshold. Thus, we compute the difference between the two vectors pf_a and pf_b and remove the noisy dimensions Δ_{j_i} that have a magnitude smaller than ε_0:

$$\Delta_j = (pf_a - pf_b) \text{ with } \Delta_{ji} = 0, -\varepsilon_0 < \Delta_{ji} < \varepsilon_0$$

Computing the L_0 norm of this vector (number of dimensions different from zero) produces a measure robust to fluctuations. The threshold ε_0 (determined experimentally) defines if the absolute difference between the feature values is low enough to be considered noise. This process is described as pseudo code in Algorithm 4.

Algorithm 4. Calculating the playerEmoDissim.

```
Input:          pf1: feature vector with facial expression 1
                pf2: feature vector with facial expression 2
                ε0: threshold for detecting different dimensions

Output:         dissim: dissimilarity between pf1 and pf2

Algorithm:

    dist_vect[] <- pf1 - pf2
    for i between 0 and length(dist_vect){
        if −ε0 < dist_vect[i] < ε0 {
            dist_vect[i] <- 0
        endif
    endfor
    dissim = L0_norm(dist_vect)
```

4.4 Labelling facial expression

The classification of a facial expression relies on a support vector machine (SVM). A SVM returns the class (label) of the feature vectors projected into the feature space. The original SVM algorithm only allows for two different classes, but we have used a multiclass SVM (KSVM) present in OpenCV, which allows for an unlimited number of classes. The classes correspond to the facial expressions (*Happiness*, *Sadness*…).

A SVM divides a high dimensional space through a set of hyperplanes to assign classes to areas of the feature space. The hyperplanes are designed to minimize the error of classification for the training data. Consider a training set constituted by a set of pairs $(x_i,\ y_i),\ i = 1, \dots, n$ with x being the feature vector and $x_i \in \mathcal{R}^n;\ y_i$ represents the class of the feature $y_i \in \{-1, 1\}$. The original SVM algorithm is based on the following optimization problem,

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i,$$

subject to $y_i(w \cdot x_i + b) \geq 1 - \xi_i,\quad \xi_i \geq 0.$

ξ_i is the error to minimize, w is the normal vector to the hyperplane and C is the penalty parameter of the error term. The detailed description of all the parameters is in Cortes and Vapnik [45].

In the facial expression classification, the training data must contain relevant examples for all facial expressions. We have chosen the CK+ database [22] for the training data, as it contains multiple faces from multiple people performing all the facial expressions. The dataset is described in more detail in Section 4.5.1.

4.5 Dataset

4.5.1 The Extended Cohn-Kanade Dataset (CK+)

The dataset that is the basis of the training and testing data for facial expression recognition is the Extended Cohn-Kanade Dataset (CK+). It is a comprehensive set containing 593 sequences of labelled images of faces of people performing various facial expressions.

Figure 26. Frames of the *Happiness* expression being performed

Initially, a video of a person performing a set of AUs is recorded. This performance starts with a neutral face (AU 0) and ends at peak expression. The performers are trained to be able to perform the expressions individually, without the interference of unrelated face actions. To collect the images of a facial expression being performed, several frames of the video were collected and saved to preserve the sequence of actions necessary to perform an expression.

In Figure 26, we show a sequence of images of the expression related to *Happiness*. Initially, the face is *Neutral* (expressionless). As we progress through the sequence, we can see the smile forming and the rest of the movements associated. The frequency of images per expression is: *Neutral* (*Neu.*): 593; *Angry* (*Ang.*): 45; *Contempt* (*Com.*): 18; *Disgust* (*Dis.*): 59; *Fear*: 25; *Happy* (*Hap.*): 69; *Sadness* (*Sad.*): 28; *Surprise* (*Sur.*): 83. Note that not all the sequences end at a full expression. Therefore, the number of *Neutral* images is superior to the sum of the other expression images.

4.6 Evaluation

4.6.1 Face detection per expression

We have decided that it would be interesting to test the performance of our face detector for the different facial expressions. In Table 7, we present the results of the detection of the faces for the CK+ dataset.

The average accuracy is 88.99%, without false positives. The 100% precision is due to the controlled environment in which the images from the CK+ dataset were taken (i.e. Figure 26 and Figure 27). There are also no true negatives (images without faces) in the dataset. Observing the face detection results per expression, we see that in the particular case of the *Disgust* expression, the results are quite low (accuracy: 31.7%). After analysing the detection results, we discovered that the eye detection component missed faces that required having both eyes almost closed. Figure 27 contains an example of the *Disgust* expression being performed that was not detected. There were also some examples where the face was not detected, mainly due to contrast or lightning issues.

Facial expression	TP	TN	FP	FN	Accuracy	Precision	Correct detection rate
Neutral	384	0	0	23	94.35%	100.00%	94.35%
Anger	25	0	0	6	80.65%	100.00%	80.65%
Contempt	13	0	0	0	100.00%	100.00%	100.00%
Disgust	13	0	0	28	31.71%	100.00%	31.71%
Fear	18	0	0	0	100.00%	100.00%	100.00%
Happiness	35	0	0	13	72.92%	100.00%	72.92%
Sadness	20	0	0	0	100.00%	100.00%	100.00%
Surprise	58	0	0	0	100.00%	100.00%	100.00%
Totals:	566	0	0	70	88.99%	100.00%	88.99%

Table 7. Expression-based face detection results

The results are very similar to the ones from the face detection evaluation (Section 3.5.2) with the Caltech dataset (average: 96.87%). The small loss in accuracy can be attributed to the lower performance achieved by some expressions (mainly *Disgust* and *Happiness*). For the remaining expression, different facial expressions do not affect our face detection algorithm's performance significantly.

Figure 27. Missed face detection (*Disgust* expression)

4.6.2 Facial expression recognition

The images detected in the previous experiment (Section 4.6.1) were used to evaluate facial expression recognition with the K-SVM. We opted for a balanced number of examples per expression to avoid a bias towards expressions with more examples (60 faces for *Neutral*; the remaining examples are the true positives in Table 7). In Table 9, we show the confusion matrix for the facial expression recognition on the CK+ data. Some facial expressions like *Surprise*, *Happiness* and *Sadness* were recognized with a high precision (96%, 93% and 89% respectively), while the remaining expressions attained a precision below 80%. The average precision was 79.49% (Table 8).

	Precision
CK+ data	79.49%

Table 8. Average precision

	Ang.	**Con.**	**Dis.**	**Fear**	**Hap.**	**Sad.**	**Sur.**
Ang.	**58.33**	8.33	8.33	0.00	8.33	16.67	0.00
Con.	20.00	**60.00**	0.00	20.00	0.00	0.00	0.00
Dis.	**42.86**	14.29	**42.86**	0.00	0.00	0.00	0.00
Fear	0.00	14.29	0.00	**71.43**	0.00	0.00	14.29
Hap.	6.67	0.00	0.00	0.00	**93.33**	0.00	0.00
Sad.	12.50	0.00	0.00	0.00	0.00	**87.50**	0.00
Sur.	4.17	0.00	0.00	0.00	0.00	0.00	**95.83**

Table 9. K-SVM confusion matrix for the CK+2 test data

There are a few reasons for the low recognition rate. The main reasons are the subtle differences between these facial expressions (*Anger*, *Contempt*, *Disgust* and *Fear*). The chosen rectangular features worked better with expressions with a lot of activity in facial features (*Surprise*, *Happiness)*. The amount of faces is also a significant fact. *Contempt* and

Disgust only have 13 peak expression images leading to expression models that are not adequate for all faces (Table 7 contains the amount of faces for each expression). Expressions with higher number of training faces lead to better facial expression recognition results.

Our average precision (79.49%) is lower than the ones found in some algorithms present in the state of the art (Xiao et al. [24]; Tian et al. [27]: 92.7%; Littlewort et al. [28]: 89.7%). The advantages of our system are: it's fully automatic (in opposition to Xiao et al. [24]) and detects full expressions (sets of AUs), not individual AUs (in opposition to Xiao et al. [24] and Tian et al. [27]). Littlewort et al. [28] is also fully automatic and takes into account full expressions but achieved better results. The main difference lies in the classification algorithms tested. We believe that, with further experiments with different classifiers, the precision of our face recognition algorithm we can be improved even further.

4.7 Summary

In this Chapter, we presented a fully automatic facial expression recognition system based on Gabor wavelets. The system is capable of using the faces detected in Section 3.2 to create a robust representation, unaffected by small variations in face alignment and rotation.

Our evaluation showed that the system worked very well with some expressions (*Surprise*, *Happiness* and *Sadness*), but was less capable of distinguishing between the remaining expressions. An evaluation with real world data from the game is present in the next Chapter.

We have also tested our face detection algorithm with the CK+ dataset, to check the influence of facial expression in the performance of our algorithm. We have found that only the *Disgust* expression affects the performance of our face detector significantly; the remaining expressions results are in line with previous experiments.

5

Affective-based interaction: ImEmotion

5.1 Introduction

Games and emotion are deeply linked. Games are designed to provoke a range of emotions in the players. What if those emotions were detected to help providing a better and more natural gaming experience? Steps towards enabling interaction with the player's bodies through gestures and movements are gaining momentum (Microsoft Kinect or Asus WAVI Xtion), but facial expression is a relatively unexplored interaction technique in videogames.

In this Chapter, we describe ImEmotion: an interactive game that explores the affective features measured by our affective-based interaction framework. We propose the extension of existing interaction paradigms through the inclusion of affect-based input in videogames. The components from the previous Chapters were combined with game specific algorithms to allow for fair competitiveness and smooth response.

5.2 The ImEmotion game

ImEmotion is a game that explores players' facial expressions as the mean of interaction. In ImEmotion, facial expression is both the controller and the basis of the scoring of the game.

Several challenges exist to accomplish such natural input method [46]. The most critical challenge resides on designing a robust and smooth facial expressions recognition algorithm to be able to map the facial expression into a recognized facial expression model. Facial expression recognition algorithms are not completely accurate (see Section 4.6). Thus, the design of the game must not frustrate the user and must react homogeneously, independently of the expression being performed.

With these requirements in mind, we identified the key design aspects of an affect-

based interaction framework to be the following:

1) **Affective awareness** – players must realize that the system (i.e. the game) is aware of them. This intends to triggers a mind-set to prepare them for the game interaction and the lack of a mechanical control.

2) **Affective stimuli –** each game round must face a challenge to the player. This challenge assumes the form of an emotion that must be performed. The player displays the emotion through an intentional facial expression.

3) **Affect-based interaction** – the framework must limit interaction to a well-defined set of affective user-responses. The reaction to players' affective states must be clear/unambiguous and in real-time. Weakly specified interactions will frustrate users with unrelated actions-reactions.

We have found these design principles to be fundamental to achieve an effective interaction based on facial expression.

5.2.1 Affective awareness

Players must realize that the game is aware of them and their affective behaviour. Initially the game tracks the player's faces to make them realize that they are the actual game character. In this initial step, all players are detected by the game (green square around the faces), Figure 28, and the two current players are manually selected (blue square), to avoid ambiguity. This creates an affective setting where players quickly realize that the game is aware of them and of their facial expressions.

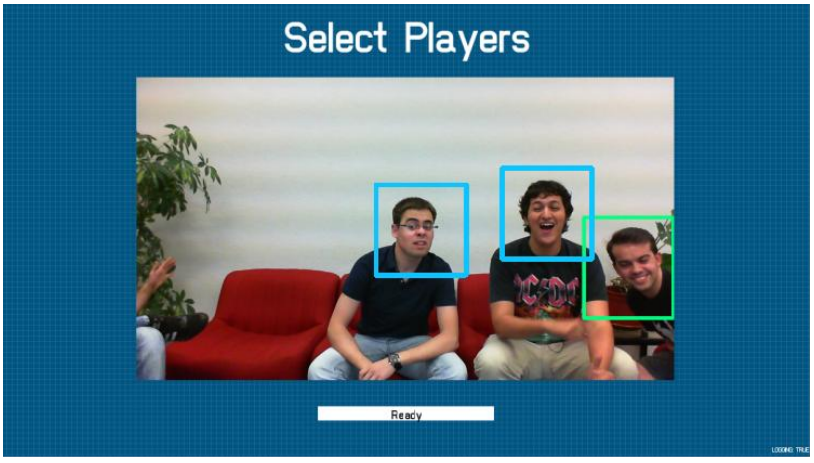

Figure 28. Players realize that the game is aware of them

This state of mind is reinforced by the fact that there is no mechanical control – the game tracks player's faces and monitors their facial expressions without mouse or keyboard interaction other that the initial player selection.

5.2.2 Affective stimuli

A specific affective context is created at the beginning of each game round posing a challenge to the player. This challenge solicits a facial expression as the game input. The player must then respond to the affective stimuli with an intentional facial expression. For the players to know what expression to perform, an elicitor is necessary. In our game, we decided to show a label to the user with the expected expression. The full set of labels is: *Anger*, *Contempt*, *Disgust*, *Fear*, *Happiness*, *Sadness* and *Surprise*. To assist users, besides the label, we present an image eliciting the intended facial expression (Figure 29).

Figure 29. Stimuli images examples

5.2.3 Playing with facial expressions

ImEmotion is two-player game where the objective is to perform a set of facial expressions. Players play simultaneously and their facial expressions are competitively scored. The player that best performs the expected expression wins a round – the player who wins more rounds wins the game.

The game is set in a room with the players seated facing a camera and a screen. The game screen (displayed in Figure 32) is composed by three main components: the player's information (Figure 32: A), the label and reaction image (Figure 32: B) and general game information (Figure 32: C). The player's information component contains the detected expression of the last detected face, the image from where it was extracted, the score obtained and the best score of the round. The facial expression stimulus component contains an image designed to help provoking a reaction and an expression label of the expected reaction. The general game information contains the game score

and the time remaining on the round. Each round consists on the following sequence:

1) A reaction image is displayed, along with the label of the expected expression. The face detection component starts searching for faces;

2) When a face is detected, the facial expression recognizer component recognizes the expression, updates the player's reaction label and compares the face to the expected one, updating the current score. The process is repeated until the timer ends;

3) At the end of the round, the best score of the round of both players is compared and the one with the best score gets one point added to the global score. If the player's best score is the same, they both get one point.

The game will continue until a set number of images are displayed. In the end, a screen with the winner is displayed (Figure 33). It contains the name of the player who won the game, a percentage that represents the average maximum score of the player and the final game score. The player's percentages will be used in the high score screen (Figure 31). The high score screen allows players to check their performance against other players and to increase competitiveness.

Both the winner screen and the high score screen were designed to increase the social and competitive aspect of the game. Players can easily compare their scores with the ones from the best players and their friends and have a personal mark to beat.

5.3 Affective interaction design

The previous section described the design aspects guiding an affective-based interaction game. In this section we detail the core algorithms behind the game: the affective framework, a facial expressions similarity algorithm and the most important element of the game: the ImEmotion scoring algorithm.

The ImEmotion algorithm is critical in the sense that it computes the game reaction to the player's affective expressions. Thus, it must offer a consistent response to all affective interactions, without favouring any particular facial expression due to algorithm accuracy issues.

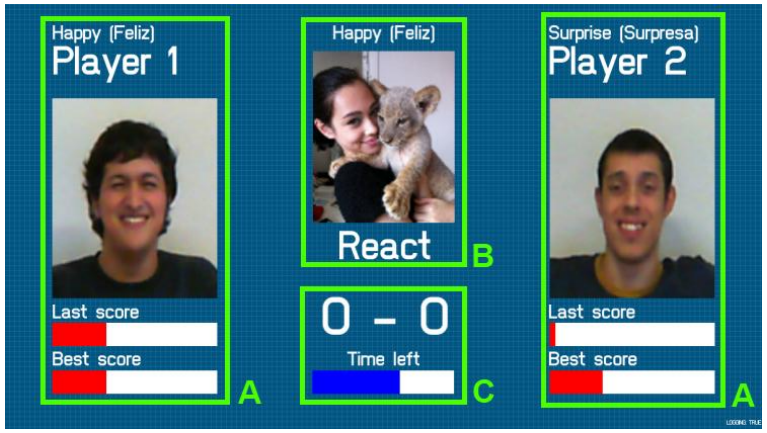

Figure 32. Two players performing the *Happiness* expression

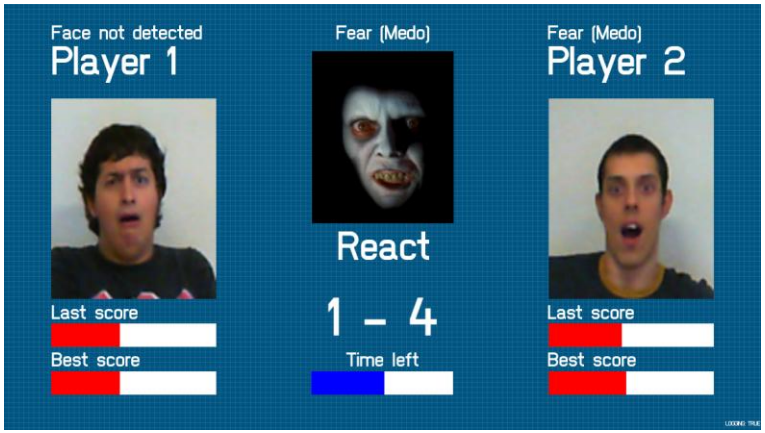

Figure 30. Two players performing to the *Fear* expression

Figure 33. Winner screen

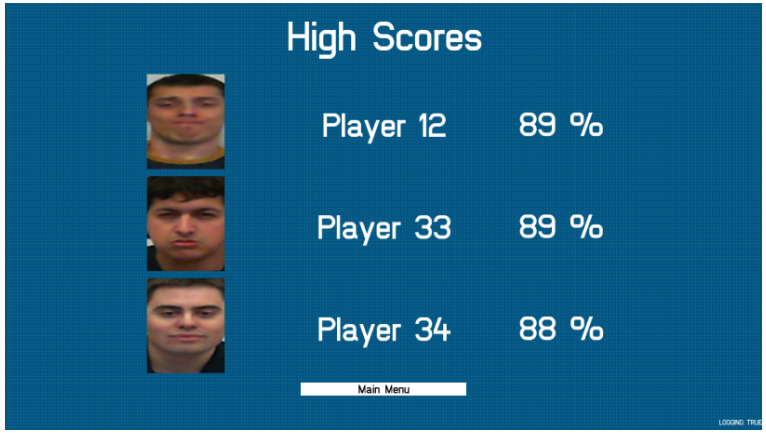

Figure 31. High score screen

5.3.1 Base face and facial expression analysis

The baseline metrics for facial expression analysis were described in Section 4.3 and 4.4:

- **playerEmoDissim:** the dissimilarity measure described in Section 4.3 can be the basis for the score of the player's facial expression. The player's face dissimilarity to an average expression face can be used to improve fairness.
- **playerEmoLabel:** the techniques described in Section 4.4 (K-SVM) are useful to tell the player if the expression is being correctly recognized; thus, the result of the K-SVM can be used as the expression label.

In the following sections, we detail how these components are used in the game.

5.3.2 ImEmotion scoring algorithm

Our goal is to devise an algorithm that offers an affective interaction with a realistic and balanced reaction to the players' expression. This is critical to keep the user responsive and immersed in the game. A significant challenge was to obtain a meaningful score that would answer players' expectations. In this section, we argue that although both expression dissimilarities and K-SVM are incomplete, a smoothed and temporally limited combination of these techniques can deliver a competitive scoring algorithm for affect-based games.

To measure the dissimilarity between a detected expression and an expected expression models, we use the dissimilarity function introduced in Chapter 4: Algorithm 4. These expected expression models are computed as the average of the feature vectors of all examples in the set Ω_j,

$$af_j = \frac{1}{|\Omega_j|} \sum_{i \in \Omega_j} f_i,$$

with $f_j = (f_{j_1}, \dots, f_{j_{288}})$. Algorithm 4 is executed with the player's face and the expected expression model as parameters and then the dissimilarity is normalized into a score between 0 and 1.

A baseline scoring algorithm would take the label from the K-SVM and the expression dissimilarity score to directly update the game interface. This approach illustrates several drawbacks and properties of the previously presented techniques:

1) Some players are detected more often than other players, giving them more chance to score.

2) The K-SVM classifier is accurate for the on-screen label of the player's face but it is not adequate for computing an expression score.

3) The expression dissimilarity is a good indicator of how close the player's expression is to average expression, but it is not adequate for providing the on-screen label.

4) Using the dissimilarity and K-SVM label separately could lead to an ambiguous situation. For example, a player's image could be labelled as *Anger* could still achieve the highest score for a *Disgust* image.

To overcome the challenges listed in the previous section, our approach was to add more game elements to increase the competitiveness of the game and remove image algorithm limitations. At the core of our approach is the prior knowledge that we know what the expected expression is. Thus, we explore that information, along with the score and the label to create improved labels and scores to design the game's scoring algorithm.

Algorithm 5. Calculating the player's score and label.

```
Input:          playerEmoLabel: label predicted by K-SVM classifier
                playerEmoDissim: score for the captured face image
                stimuliEmotion: emotion related to the image being displayed
                time: current round time

Output:         label: emotion label to display to the user
                score: score for the last image to be displayed the user
                bestScore: score of the player's best facial expression

Algorithm:

1.    if playerEmoDissim > HIGH_CONFIDENCE
2.          label <- stimuliEmotion    assumes K-SVM label
      else
3.          label <- playerEmoLabel    assumes dissimilarity label
      endif

4.    if Label == stimuliEmotion
5.          if playerEmoDissim < NOISY_EXPRESSION {
6.                score <- playerEmoDissim + rand_uniform(0, BONUS)
            else
7.                score <- playerEmoDissim
            endif

8.          score = temporalSmoothing(score, time)
9.          score = playerEmoDissim - |rand_gaussian(0, REACT)|

      endif

10.   if bestScore < score
11.         bestScore = score
      endif
```

The scoring algorithm is described in Algorithm 5. The algorithm computes three variables: the *Label* of the current facial expression, the *Score* of that facial expression, and the *BestScore* achieved by that player. *rand_uniform(a,b)* returns random numbers

between *a* and *b* that follow the uniform distribution. *rand_gaussian (a,b)* returns random numbers between *a* and *b* that follow the Gaussian distribution.

5.3.3 Fair competitiveness

The first step towards a fair game scoring scheme, is to reward the best facial expressions and not a large number of average facial expressions. This is to avoid frustrating players that are not detected as often as others. Thus, the game encourages the most acute facial expressions.

The scoring algorithm is described in Algorithm 5. The emotion label (output variable *Label*) to be shown to the player is an indicator of how the player is being interpreted by the game. This label is determined by one of two ways (Algorithm 5: steps 1 to 3):

1) If the dissimilarity between the player's expression and the current emotional stimulus (*playerEmoScore*) is above the HIGH_CONFIDENCE threshold, the label is replaced by the expression of the image shown. This ensures players receive the correct label if they achieve a high dissimilarity value and the K-SVM output is ignored.

2) If the dissimilarity value (*playerEmoScore*) is not high enough, the Label is assigned by the K-SVM classifier (step 3).

The HIGH_CONFIDENCE threshold was determined on the CK+ dataset, and it corresponds to a 95% average accuracy. Once the label has been determined, the score of the current facial expression must be calculated. If the current label is correct (Algorithm 5: step 4), but the dissimilarity is too low we assume there's too much noise (Algorithm 5: step 5), and adjust the value by adding a uniformly random value between 0 and BONUS (Algorithm 5: step 6). This solves two issues: (a) the disagreement between the dissimilarity and the K-SVM, and (b) provides a meaningful response to the user to keep him responsive.

If the player's facial expression is not recognized as the stimuli emotion (Algorithm 5: step 9), the score will be penalized by a Gaussian random value. This happens when the expression performed is not close to the expected one (both the classifiers and the dissimilarity gave results different from the expected label), so it's fair to penalize the score and show the incorrect *playerEmoLabel* (i.e., different from *stimuliEmotion*).

Finally, since in some cases the detected faces are too noisy and the visual analysis is too slow (search for faces on the image) we add a jitter to the player's last score to inform the player that the game is responding to their expressions. This is done in Algorithm 5: steps 6 and 11 when random values are added to the score.

The score of the round (*BestScore*) is the value that will determine which player wins

the round. To avoid rewarding players that are detected more often than others and to reward the best peak expression, the *BestScore* is the best score of the round, instead of a sum of individual images scores (Algorithm 5: steps 10 and 11).

5.3.4 Progressive competitiveness

To maintain the competitiveness throughout the entire round duration we impose a limit on players' score. This will allow players to adjust themselves to the camera and to train their facial expressions.

The limit is progressively removed until the score is no longer limited. It is applied in Algorithm 5: step 8. The score is capped to 30% on the first 10 seconds of the round, by multiplying the score value by 0.3. Between 10 and 20 seconds, the multiplier increases linearly from 0.3 to 1. On the last 10 seconds, the score is fully unlimited (multiplier equal to 1). Figure 34 contains a diagram of the score range per amount of time.

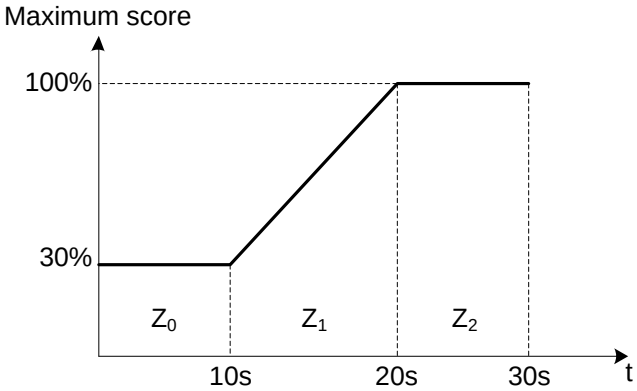

Figure 34. Time penalty to the score

5.4 Algorithms' evaluation

The scoring algorithm evaluation was conducted on a real gaming environment, where the 46 volunteers were asked to play the game for a short period of time and to answer a questionnaire about their experience. In this Section we discuss the performance of the game algorithms in this setting. Section 5.5 presents the questionnaire results.

Number of players	46
Total number of games played	29
Average nº of games played per player	1.26
Nº of images player per game (fixed)	5
Time per image (fixed)	30 seconds

Table 10. General game statistics

The purpose of the game evaluation is to assess the effectiveness of the addressed hypothesis: using affective-based interaction for competitive games.

Initially, players were briefed about the objective of the game and the interface explained. Then, they played five rounds of the game and at the end, were asked to answer the questionnaire. Players could play more than one game if desired.

The trial was conducted like a regular game session on a social environment (i.e. multiple friends watching), contrasting with the strictly controlled conditions in image capturing for most faces datasets (i.e. the CK+ dataset). In our trial, players moved their faces a lot (sometimes even stopped looking at the camera) and performed unexpected expressions (mainly *Happiness*). Table 10 summarizes the captured data statistics, each player played on average 1.26 games. We captured and classified 42,937 facial expressions, with an average of 295 facial expressions per round. This data is available for research purposes[9]. Next, we discuss the evaluation of the facial expression recognition, the game scoring algorithm and competiveness.

Facial expression	TP	TN	FP	FN	Accuracy	Precision	Correct detection rate
Anger	1599	0	0	1917	45.48%	100.00%	45.48%
Contempt	5806	0	0	3280	63.90%	100.00%	63.90%
Disgust	3267	0	0	3536	48.02%	100.00%	48.02%
Fear	2201	0	0	2022	52.12%	100.00%	52.12%
Happiness	13957	0	0	5846	70.48%	100.00%	70.48%
Sadness	4711	0	0	2602	64.42%	100.00%	64.42%
Surprise	11396	0	0	5796	66.29%	100.00%	66.29%
Average:	42937	0	0	24999	63.20%	100.00%	63.20%

Table 11. Expression-based face detection results

5.4.1 Expression-based face detection

Before running the facial expression recognition algorithm, it was necessary to detect where the players are in the camera image. Our face detection tests on the gaming environment attained an accuracy of 63.20%. This result is lower than the 89% accuracy obtained on the development dataset CK+. This difference is caused by non-frontal facial poses and heterogeneous lighting conditions. The face detector was calibrated to only work on a small area of the image where the player was. Therefore, there were no false

[9] http://search.di.fct.unl.pt

positives and every misdetection was a false negative. Nevertheless, the camera was set to 15 frames, which allowed us to comfortably process 9 images per second. Thus, each player's face was processed and its score was updated, on average, more than 4 times per second.

5.4.2 Facial expression recognition

The importance of facial expression recognition in the game is twofold: first, it must recognize the presence of the expected facial expression; second, it ought to provide players with adequate feedback about how the game is assessing their facial expression.

Table 12 and Table 13 contain the confusion matrices for the image label (*currentEmotion*) expression versus the detected expression (K-SVM and Dissimilarly Score respectively) from Section 5.3.1. Table 14 contains the confusion matrix for the label expression versus the detected expression. This value is determined by taking the displayed image label (*currentEmotion*) and comparing it to the label computed by Algorithm 5.

Figure 35 contains a visual representation of the confusion matrices from all the steps of our algorithms related to facial expressions and Table 15 contains the precision values. This table represents our different facial expression recognition experiments.

Initially, the K-SVM algorithm was tested with the CK+ data (Section 3.5.3). The K-SVM confusion matrix with the CK+ data illustrates good results for some expressions (*Happiness*, *Surprise* and *Sadness*) but is less capable of distinguishing between the remaining expressions (79.49% average precision).

When testing the K-SVM with the game trail data (Table 11), the results were considerably worse (31.70% average precision), due to the differences between the training and real world data (illumination, rotation, alignment…) and the uncertainty regarding whether the player is performing the correct expression. The value of this label corresponds to the *playerEmoLabel* from Algorithm 5.

The dissimilarity function described in Algorithm 4 was also evaluated with the condition present in Algorithm 5, step 1. The dissimilarity function is not a true facial expression recognition; it only determines if the dissimilarity between the detected expression features and the average features for the expected expression is low enough to consider the detected expression equal to the expected one. Therefore, as the condition evaluated is binary, we decided to divide the unknown values between the remaining expressions in the confusion matrix, allowing an easy comparison with the other confusion matrices. The average precision was 61.70%, ranging from 46.65% for *Surprise* to 73.30% for *Contempt*.

	Ang.	Con.	Dis.	Fear	Hap.	Sad.	Sur.
Ang.	**31.64**	2.88	0.00	7.13	27.14	3.44	27.77
Con.	**49.38**	6.92	1.69	2.62	20.29	1.43	17.67
Dis.	29.94	11.94	3.24	5.63	**31.31**	0.92	17.02
Fear	19.67	8.22	0.91	15.31	**25.12**	8.22	22.54
Hap.	17.20	7.49	0.57	9.36	**52.82**	2.56	10.00
Sad.	**38.38**	6.71	1.74	5.20	21.25	7.49	19.23
Sur.	12.65	4.84	2.88	7.78	30.77	1.26	**39.81**

Table 12. K-SVM confusion matrix for the game data (B)

	Ang.	Con.	Dis.	Fear	Hap.	Sad.	Sur.
Ang.	**52.41**	7.93	7.93	7.93	7.93	7.93	7.93
Con.	4.45	**73.30**	4.45	4.45	4.45	4.45	4.45
Dis.	5.98	5.98	**64.10**	5.98	5.98	5.98	5.98
Fear	7.47	7.47	7.47	**55.16**	7.47	7.47	7.47
Hap.	5.03	5.03	5.03	5.03	**69.82**	5.03	5.03
Sad.	5.95	5.95	5.95	5.95	5.95	**64.32**	5.95
Sur.	8.89	8.89	8.89	8.89	8.89	8.89	**46.65**

Table 13. Dissimilarity score confusion matrix (C)

	Ang.	Con.	Dis.	Fear	Hap.	Sad.	Sur.
Ang.	**54.91**	1.13	0.00	5.00	19.01	1.63	18.32
Con.	16.52	**62.50**	0.52	1.72	8.16	0.16	10.42
Dis.	9.09	4.53	**60.54**	3.61	11.39	0.52	10.32
Fear	4.41	4.63	0.82	**64.61**	8.72	2.68	14.13
Hap.	2.37	2.90	0.24	4.39	**84.89**	0.86	4.35
Sad.	7.92	3.42	1.40	3.54	10.91	**63.94**	8.87
Sur.	10.42	2.67	2.57	5.56	20.52	0.96	**57.30**

Table 14. Scoring algorithm confusion matrix (D)

The K-SVM classifier was combined with the dissimilarity function (Algorithm 4) to implement the scoring algorithm (Algorithm 5). The precision was 68.23%, and more importantly, the per-facial expression accuracy is more balanced than with the K-SVM alone (the diagonal of Algorithm 5 confusion matrix is more uniform). Moreover, the cross-interference between different facial expressions is also reduced: the confusion matrix for the K-SVM with real world data shows how the confusion between *Anger*, *Contempt* and other expressions were reduced on the game scoring algorithm. Thus, the homogeneous recognition of user's facial expression is a critical aspect to achieve a consistent affect-based interaction. Algorithm 5 achieves a less ambiguous detection of the facial expressions. This success is rooted in the two main factors: first, the dissimilarity norm L_0 is robust enough to avoid false positives, and secondly the combination of the two methods can produce better results than just the K-SVM alone.

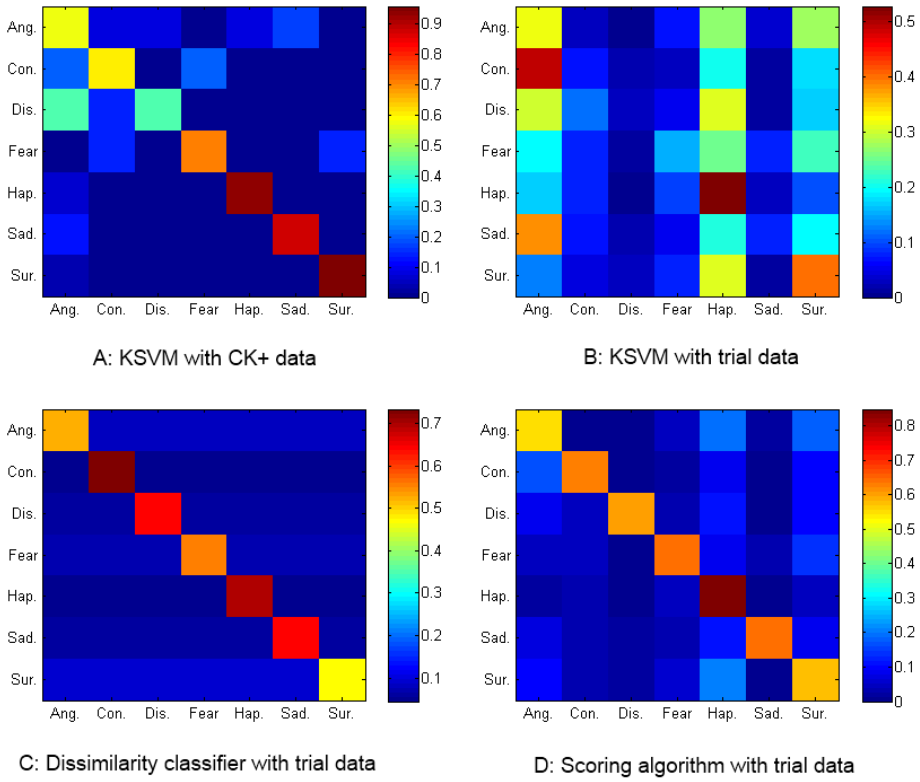

Figure 35. Confusion matrixes for the different trial. Color scales are different across graphs

Experiments:	Dataset	Precision:
K-SVM	CK+ data	79.49%
	Game trial data	31.70%
Dissimilarity classifier	Game trial data	61.70%
Scoring algorithm	Game trial data	68.23%

Table 15. Full results for facial expression recognition

The results obtained using the game data are not directly comparable to other facial expression recognition algorithms, due to the inherent uncertainty regarding the performed expression. Thus, it is hard to compare the performance of the algorithms with other studies. Nevertheless, we believe that these results are adequate for the game and make for an enjoyable game experience. The assessment of the algorithm's performance with data available to the scientific community is present in Section 4.6.2.

5.4.3 Scoring assessment

To assess the performance of the scoring algorithm, we selected the best rounds and worst rounds and compared their performances. All game rounds were divided into two categories: the best rounds where the player finished with more than 80% of final score (Figure 36), the worst rounds are the ones where the player finished with less than 80% of final score (Figure 37). This allows us the analysis of different gaming strategies.

The three curves on Figure 36 and Figure 37 the represent: (a) *Score* is the current score displayed to the player returned by Algorithm 5; (b) *BestScore* is the best score of the round returned by Algorithm 5; and (c) the *PlayerEmoDissim* is the dissimilarity between a face and an average face returned by Algorithm 4.

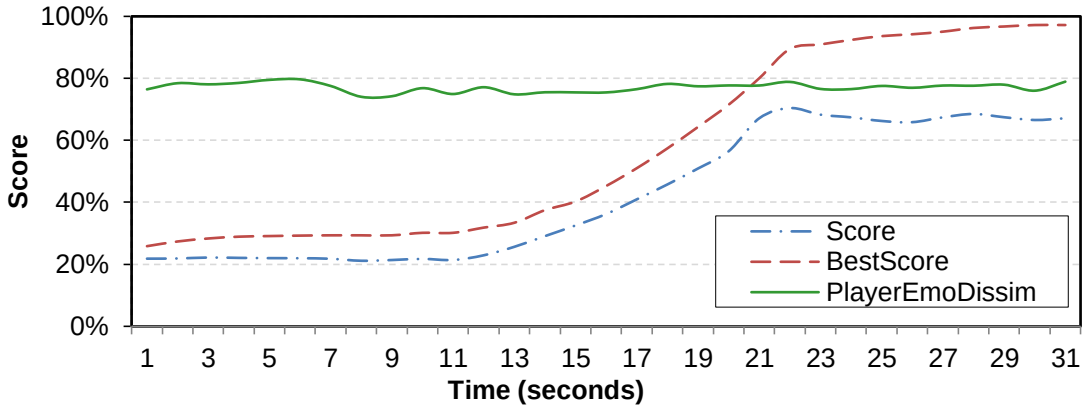

Figure 36. "Best rounds" score evolution

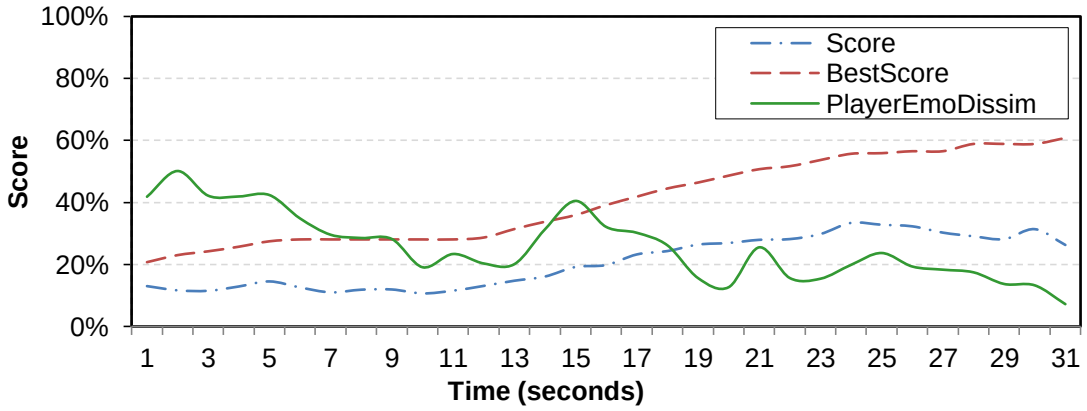

Figure 37. "Worst rounds" score evolution

In the "best rounds" graph, Figure 36, the scores evolved as expected: the *BestScore* is very similar to the time penalty graph in Figure 34, except for the unlimited zone, where the players were still able to increase the *BestScore* slowly. Other interesting fact is that *playerEmoDissim* is very consistent across the round with a value range between 74 and 80%. This value is not affected by our techniques to increase competitiveness; it is only dependent on the captured face image. Thus, its small range means that the players were able to maintain consistent expressions across the round.
In the "worst round" graph, Figure 37, the most obvious difference is the scores range and the lack of consistency of the *PlayerEmoDissim*. It varied between 7% and 50%, decreasing as the round passed. The time penalty imposed on the score is also visible in both the *Score* and *BestScore*, but their values are much lower than the ones from the "best players". The *Score* value is much smoother than the *PlayerEmoDissim* value, showing that the Scoring algorithm was capable of smoothing out the dissimilarity inconsistencies and deliver a better gaming experience (players could be performing very poorly but their score would always correspond to their "best" facial expression).

When looking at the graphs, one can observe specific trends and differences across both categories. Best players had a high *PlayerEmoDissim* throughout the round duration, meaning that they were highly competitive until the last moment of the round. The worst players showed a different trend: they would either stop performing the expected expression in the first moments or their attempts to perform the correct expressions got worse as the time passed.

5.5 User study

At the end of the game, players answered a questionnaire about their gaming experience. We took into account the heuristics from [47] that could be applied to ImEmotion when preparing the questionnaire. Besides standard questions regarding player information (gender, age, playing videogames frequency), and general gameplay (i.e. game objective, difficulties felt, enjoyment level), we also addressed ImEmotion specific aspects such as interaction novelty, enjoyment and perceived accuracy (label accuracy and score accuracy). There were also open answer questions, where the players could contribute with suggestions.

Volunteers were mostly undergraduate level students, aged between 18 and 25 years old. The gender distribution is balanced (24 male and 22 female).

5.5.1 Game design assessment

Volunteers found the game easy to understand (91% of the answers were "High" or "Very High") and enjoyed playing (91% of the answers were "High" or "Very High"),

(Figure 38). We consider that the difficulty level is adequate (not too high or too low), as the majority of the answers regarding the difficulties felt during the game were "Medium".

Regarding the image exhibition time (round duration) and number of rounds per game, Figure 39, 46% of the players wanted more images per game and 30% wanted less exhibition time per image. This is in line with what we observed: if one of the players got a very high score at the middle of the round, the competition on that round could end sooner. Thus, volunteers expected affective stimuli to be shorter and more varied.

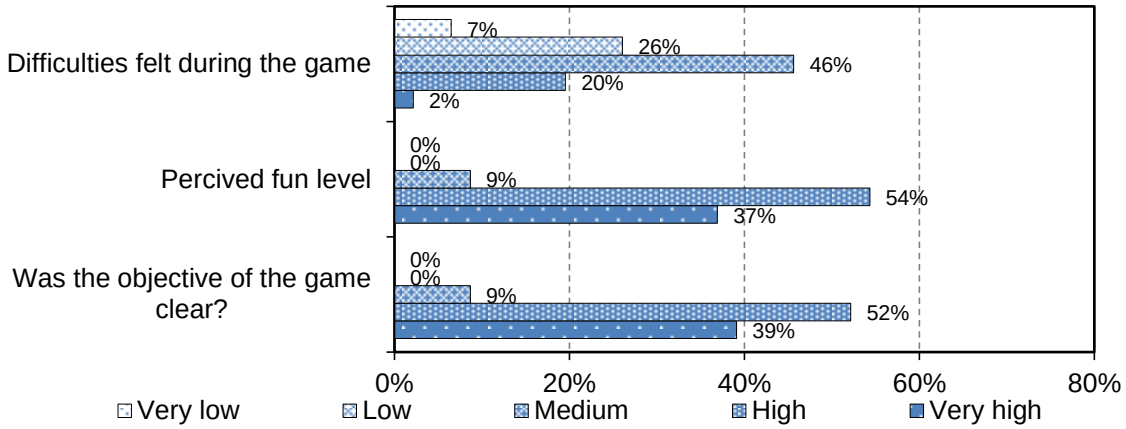

Figure 38. Game play assessment

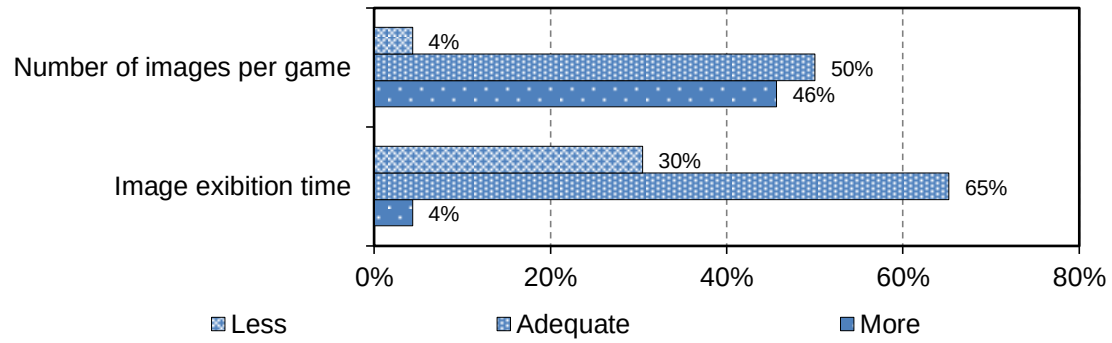

Figure 39. Round duration and images

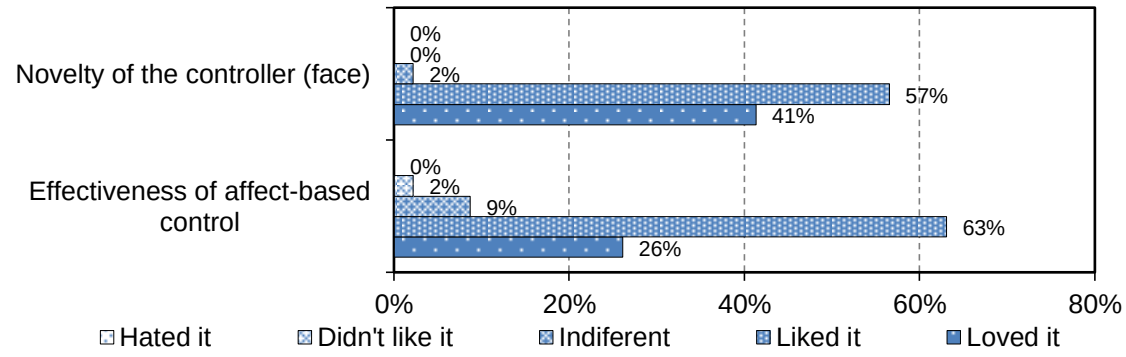

Figure 40. Affect-based interaction effectiveness

When asked what would be the adequate number of players, 69% of the players chose "Two players", while the remaining 31% chose "More than Two players" (no one chose the option "One player"). One of the possible reasons for the relatively low number of "More than Two players" answers is that the users were presented with a two player version of the game, which could lead to a bias towards that answer.

5.5.2 Affect-based interaction

The next questions focus on how players assess the various aspects of the game design (Figure 40). The large majority of players liked or loved the usage of facial expression as a controller (89%), and novelty of the controller type (98%). This is a highly positive result supporting the initial hypothesis of using the face as a game controller of the game, and supports the effectiveness of the proposed solution.

5.5.3 Perceived accuracy of affect recognition

Other critical aspect was the perceived accuracy of the score and the label by the players (Figure 41). Most of the players considered that the score was accurate most of the times (66%: 4 or over, average: 3.6), with a small reduction of when the question was about the label (43%: 4 or over, average: 3.3). This result is positive, as it shows that more than half of the players were satisfied with the label accuracy. It is important to compare these results to the algorithm's performance presented on Figure 8 and Table 5. The perceived accuracy (3.3 in 5: 66% for label) is in line with the measured accuracy (68.23%) in our formal evaluation (the confusion matrix diagonal).

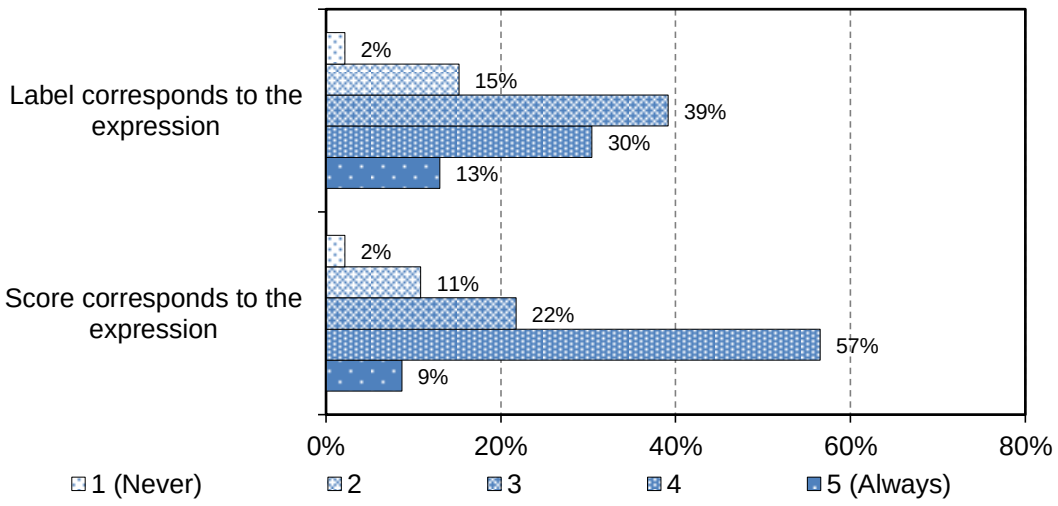

Figure 41. Accuracy perceived by the players

We also investigated the accuracy of specific expressions and measured how players felt about the different expressions and the expression specific difficulties they encountered

(Figure 42). There are three main important conclusions that can be drawn from these answers. Regarding the "Hardest expression to perform", the answers were quite distributed across expressions, except for *Contempt*. In this specific question, 35% of the players reported that the C*ontempt* was the hardest expression to perform. This result is backed up by our observations. A considerable number of players said that they did not knew how to make a "*Contempt* face", situation that did not happened as often with other expressions.

Regarding the expression whose score was least adequate, the two top results were *Contempt* (24%) and *Happiness* (20%), although the reason for the high percentage is different. In *Contempt*, there is possibly a relation with the previous question. Most players found this expression hard to perform and considered the score not to be accurate. According to our study, volunteers found the *Happiness* expression to be one of the easiest to perform, but considered the score not to be accurate. The cause for these answers is related to the classifier's good performance in detecting happy faces. Thus, the least adequate in this case, means players would get a higher score with less effort.

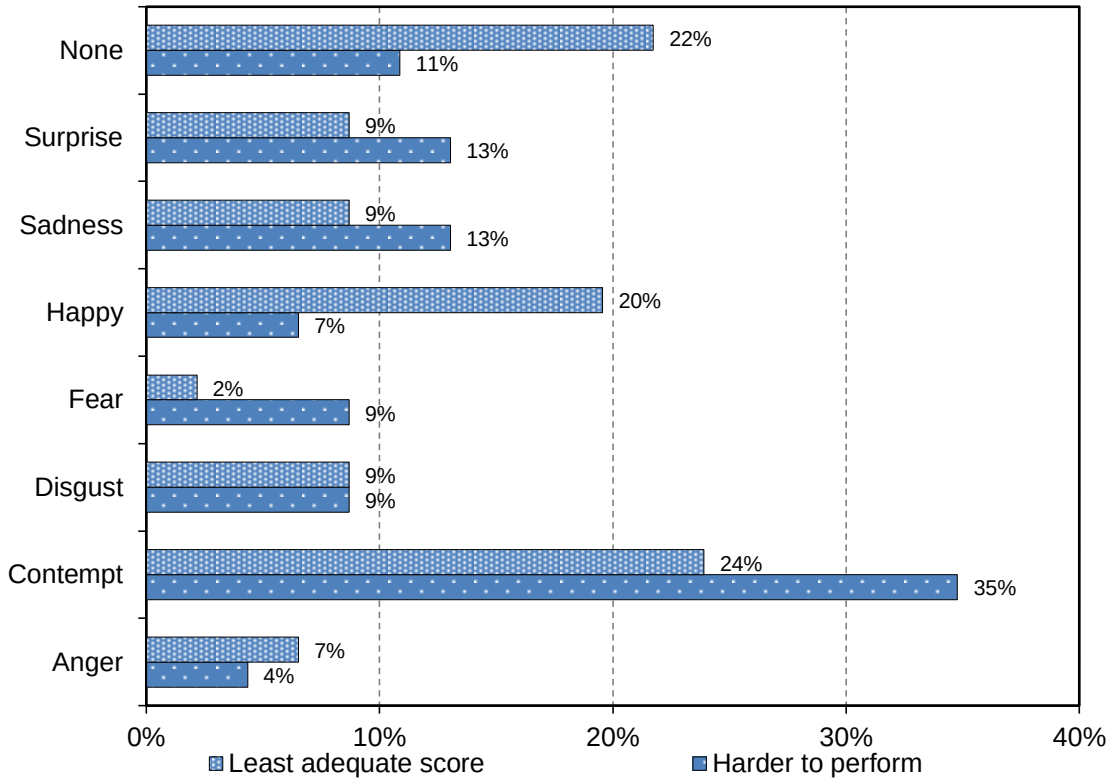

Figure 42. Expression specific assessment

5.5.4 Gaming habits

These questions were made to determine the type of games played and the frequency. The game types were divided in Casual (Facebook games, Miniclip games…), Mobile

(mobile phone and handheld consoles), Interactive (Microsoft Kinect, PlayStation Move or Nintendo Wii) and Traditional (computer or console with regular controller). We were especially interested on the gaming habits on Interactive games, as they are the ones that bear more similarities with our game.

Male volunteers spend more time playing all of the games categories, with a special prevalence of Traditional and Mobile types. 79% of female volunteers rarely or never play any type of game, with very similar results amongst the different game types. Figure 43 shows answers divided by gender and game type.

An important observation that can be drawn from the data is that Interactive gaming isn't very prevalent amongst our volunteers. Only 8% of the players play them on a weekly or more frequent basis. The percentage of players that never played this type of games is also quite high (34%). This shows that the majority of our volunteers are only used to manual controls.

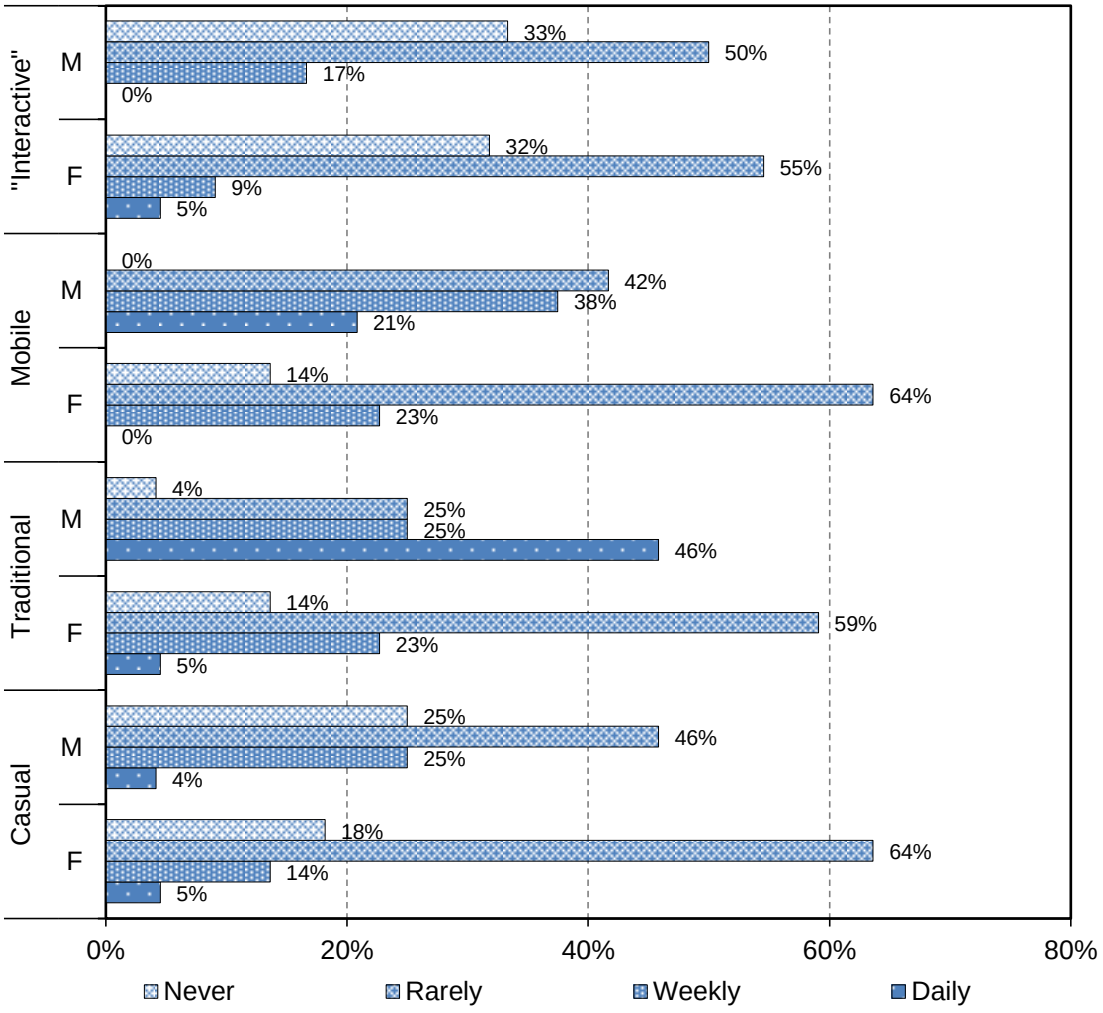

Figure 43. Game habits grouped by gender

5.6 Summary and Conclusions

In this Chapter, we proposed an emotion-based game exploring facial expressions as the sole interaction mechanism. The components of the game algorithms were detailed and exhaustively evaluated. The main conclusions to be drawn from our contributions concern four main points.

Algorithms for robust affective interaction. To deliver a consistent response to all affective interactions (without favoring any particular facial expression) existing algorithm idiosyncrasies had to be overcome. Thus, a careful selection of specific algorithms was crucial to the interaction success. An effective representation of facial expressions AU was used for computing robust facial expressions similarities based on the L_0 norm.

Facial expressions and competitiveness. Most facial expression reactions were not natural, but we considered that some of the reaction images related to *Disgust*, *Happiness* and *Surprise* really elicited the expected expression. The images related to other expressions required a more difficult reaction. In line with the previous point, the players reported that some of the expressions were hard to perform (in particular *Contempt*). The competitiveness factor of the game distorted the emotion versus facial expression link.

Gamification. Game trials illustrated how affective interaction can be successfully used as a videogame controlling mechanism. Despite the limitations of current state-of-the-art image-processing techniques, the proposed game design was able to deliver an emotion-based game. In particular, the ImEmotion scoring algorithm is the key component implementing the game competitiveness, the affective interaction. Its strength arises from a combination of several image-processing techniques, more specifically, the AUs representation, the L_0 norm and the K-SVM classifier.

Social component is key. We observed that when players came in larger groups (5 or more people), players had more fun. The higher the number of people watching, the higher would be the enjoyment of the players (bursting into laugher would be more common). Thus, the social environment allowed people to relax and explore the affective interaction more freely.

6
Conclusions

6.1 Achievements and contributions

This thesis contributes to the work tackling affective interaction in the area of human computer interaction. This developing area will make applications aware of users' affective state and allowing computers to react in more natural ways. In our user study, players reacted quite well to the affective features of ImEmotion. Thus, we believe that further research to make facial expressions analysis more robust will pave the road to novel and unobtrusive input methods.

We proposed an affect-based interaction framework and assessed its effectiveness in a computer game. We researched several computer vision methods to robustly recognize people and their facial expressions. The framework combines a **face detection algorithm** created on top of the ideas of Viola and Jones; a **face recognition algorithm** based on eigenfaces and K-NN search and an automatic **facial expression recognition**, based on Gabor wavelets and a **robust facial expression features representation.** Our results were in line with the best automatic face detection, face recognition and facial expression recognition algorithms.

The **ImEmotion** game uses facial expressions as the sole controller and promotes player's competitiveness. The game relied on the framework and proposed a scoring algorithm to overcome computer vision issues and increase competitiveness.

ImEmotion was evaluated in a **user trial** with 46 users, whose opinions were collected in a form regarding affective interaction. The images collected in the trial are available for the scientific community as a **facial expression dataset**.

Dataset. The data captured in the game trial is unique, in the sense that it captures facial expressions in a game setting. Expressions might not be related to the true emotion, but in a gaming environment, they can be considered authentic. Thus, these images are extremely useful to train new facial expression classifiers and to compare the

expressions of people with the expressions present in EMFACS. Therefore, we have decided to make the images and labels available to the scientific community[10].

The dataset is composed of 42,937 images from the seven EMFACS expressions. The images are divided by round and individual and each image contains the label detected by Algorithm 5 and the *stimuliEmotion* that corresponds to the expression that was displayed and expected for the player to perform. The labels are the ones present in the EMFACS and are coded with a single digit that corresponds to an expression.

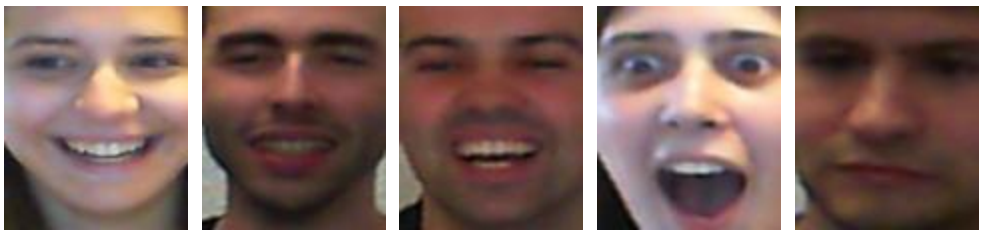

Figure 44. Face examples from the dataset

The data is labeled by the classifier. Therefore, the labels are not guaranteed to be correct and fully in line with the necessary parameters for facial expression classification from EMFACS.

6.2 Future work

Combine Active Appearance Models and Gabor Wavelets. The features we used (Gabor Wavelets) do not require any user input. However, Active Appearance Models can more effectively represent face traits by recurring to manual input. A promising research direction is the combination of these two approaches.

Integrate other types of affective stimuli. IAPS [35] provides images that are designed to elicit genuine emotions, but they are not adequate for a game environment. Audio and video stimuli can be incorporated and used as the single elicitor. For example, a movie player application will stimulate authentic facial expression when users watch comedy movies.

Interaction design with affect features in mind. Human Computer Interaction (HCI) is increasingly expanding the interaction paradigm towards more modalities of human expression such as gestures, body movements, and other natural interactions. We believe more work in this direction will help understanding how affect-based interaction can be embedded in the design of future software products.

[10] See http://search.di.fct.unl.pt for details on how to obtain the dataset

References

[1] Sony Online Entertainment, "SOEmote Guide," 2012. [Online]. Available: http://soemote.com/guide/index.vm. [Accessed: 30-Aug-2012].

[2] T. Hardware, "EverQuest 2 Receives Facial Tracking Upgrade," 2012. [Online]. Available: http://www.tomshardware.com/news/SOEmote-EverQuest-SOE-MMORPG-Live-Driver,16701.html. [Accessed: 30-Aug-2012].

[3] N. Stoiber, O. Aubault, R. Seguier, and G. Breton, "The mimic game," in *ACM SIGGRAPH 2010 Talks on - SIGGRAPH '10*, 2010, p. 1.

[4] A. Paiva, M. Costa, R. Chaves, M. Piedade, D. Mourão, D. Sobral, K. Höök, G. Andersson, and A. Bullock, "SenToy: an affective sympathetic interface," *International Journal of Human-Computer Studies*, vol. 59, no. 1–2, pp. 227–235, Jul. 2003.

[5] Microsoft, "Kinect for Xbox 360." [Online]. Available: http://www.xbox.com/pt-PT/Kinect. [Accessed: 30-Aug-2012].

[6] I. Arapakis, Y. Moshfeghi, H. Joho, R. Ren, D. Hannah, and J. M. Jose, "Integrating facial expressions into user profiling for the improvement of a multimodal recommender system," in *2009 IEEE International Conference on Multimedia and Expo*, 2009, pp. 1440–1443.

[7] L. Aryananda, "Recognizing and remembering individuals: Online and unsupervised face recognition for humanoid robot," in *Intelligent Robots and Systems*, 2002, no. October, pp. 1202–1207.

[8] C. L. /Superviso.-B. Breazeal, "Sociable machines: expressive social exchange between humans and robots," Jan. 2000.

[9] P. Viola and M. J. Jones, "Robust Real-Time Face Detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[10] R. Rojas, "AdaBoost and the Super Bowl of Classifiers A Tutorial Introduction to Adaptive Boosting," *Technical Report*, 2009.

[11] Y. Freund and R. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Computational learning theory*, vol. 904, pp. 23–37, 1995.

[12] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proceedings. 1991 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1991, pp. 586–591.

[13] J. Ruiz-del-Solar and P. Navarrete, "Eigenspace-Based Face Recognition: A Comparative Study of Different Approaches," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 35, no. 3, pp. 315–325, Aug. 2005.

[14] Y. Y. L. Yu and Zhujie, "Face recognition with eigenfaces," in *Proceedings of 1994 IEEE International Conference on Industrial Technology - ICIT '94*, 1994, pp. 434–438.

[15] E. Lizama, D. Waldoestl, and B. Nickolay, "An eigenfaces-based automatic face recognition system," in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation*, 1997, vol. 1, pp. 174–177.

[16] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711–720, Jul. 1997.

[17] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, 1994, pp. 138–142.

[18] Y. L. Tian, T. Kanade, and J. F. Cohn, "Facial expression analysis," in *Handbook of face recognition*, 1st ed., S. Z. Li and A. K. Jain, Eds. New York City, U.S.A.: Springer, 2005, pp. 247–275.

[19] C. Darwin, *The expression of the emotions in man and animals*, 2nd ed. New York City, U.S.A.: D. Appleton & Company, 1899, p. 374.

[20] M. S. Bartlett, B. Braathen, G. Littlewort-Ford, J. Hershey, I. Fasel, T. Marks, E. Smith, T. J. Sejnowski, and J. R. Movellan, "Automatic analysis of spontaneous facial behavior: A final project report," *Institute for Neural Computation MPLab TR2001*, vol. 8, 2001.

[21] P. Ekman, W. V. Friesen, and J. C. Hager, *Facial Action Coding System: A Technique for the Measurement of Facial Movement*. Palo Alto, U.S.A.: Consulting Psychologists Pres, 1978.

[22] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The Extended Cohn-Kanade Dataset (CK+): A complete dataset for action unit and emotion-specified expression," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops*, 2010, pp. 94–101.

[23] P. Ekman, "Facial expression and emotion," *The American psychologist*, vol. 48, no. 4, pp. 384–392, Apr. 1993.

[24] J. Xiao, T. Kanade, and J. F. Cohn, "Robust full motion recovery of head by dynamic templates and re-registration techniques," in *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, 2003, pp. 163–169.

[25] Y.-I. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 2, pp. 97–115, 2001.

[26] T. Moriyama, T. Kanade, J. F. Cohn, Z. Ambadar, and H. Imamura, "Automatic recognition of eye blinking in spontaneously occurring behavior," in *Proceedings. 16th International Conference on Pattern Recognition*, 2002, vol. 4, pp. 78–81.

[27] Y. Tian, T. Kanade, and J. F. Cohn, "Evaluation of Gabor-wavelet-based facial action unit recognition in image sequences of increasing complexity," in *Automatic Face and Gesture Recognition, 2002. Proceedings. Fifth IEEE International Conference on*, 2002, pp. 229–234.

[28] G. Littlewort and I. Fasel, "Fully automatic coding of basic expressions from video," *INC MPLab Technical Report*, p. 6, 2002.

[29] R. Picard, "Affective Computing," *MIT Technical Report*, no. 321, 1995.

[30] L. E. Nacke, "Wiimote vs. controller," in *Futureplay '10 Proceedings of the International Academic Conference on the Future of Game Design and Technology*, 2010, pp. 159–166.

[31] MMORPG.com, "EverQuest II: SOEmote Released." [Online]. Available: http://www.mmorpg.com/discussion2.cfm/thread/359738/. [Accessed: 30-Aug-2012].

[32] Joysitq, "The Daily Grind: Do you really want MMO innovation?" [Online]. Available: http://massively.joystiq.com/2012/08/16/the-daily-grind-do-you-really-want-mmo-innovation/. [Accessed: 30-Aug-2012].

[33] J. Kim, N. Bee, J. Wagner, and E. André, "Emote to win: Affective interactions with a computer game agent," *Lecture Notes in Informatics (LNI)*, vol. 50, pp. 159–164, 2004.

[34] P. Lang, "The emotion probe: Studies of motivation and attention.," *American psychologist*, vol. 50, no. 5, pp. 372–385, 1995.

[35] P. Lang, "International affective picture system (IAPS): Affective ratings of pictures and instruction manual," *Technical Report A-8*, 2008.

[36] N. Wang and S. Marsella, "Introducing EVG: An emotion evoking game," *Intelligent Virtual Agents*, vol. 4133/2006, pp. 282–291, 2006.

[37] A. G. Brooks, J. Gray, G. Hoffman, A. Lockerd, H. Lee, and C. Breazeal, "Robot's play: interactive games with sociable machines," *ACM Computers in Entertainment*, vol. 2, no. 3, pp. 1–10, Jul. 2004.

[38] G. J. Edwards, C. J. Taylor, and T. F. Cootes, "Interpreting face images using active appearance models," in *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, 1998, pp. 300–305.

[39] M. Weber, "Unsupervised learning of models for object recognition," California Institute of Technology, Passadena, CA, USA, 2000.

[40] F. Samaria, "Face recognition using hidden Markov models," Trinity College, University of Cambridge. U.K., 1994.

[41] M. Yeasin, B. Bullot, and R. Sharma, "Recognition of facial expressions and measurement of levels of interest from video," *IEEE Transactions on Multimedia*, vol. 8, no. 3, pp. 500–508, Jun. 2006.

[42] M. Dahmane and J. Meunier, "Continuous emotion recognition using Gabor energy filters," in *Proceedings of the 4th international conference on Affective computing and intelligent interaction*, 2011, pp. 351–358.

[43] B. S. Manjunath and W. Y. Ma, "Texture features for browsing and retrieval of image data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 8, pp. 837–842, 1996.

[44] E. O. Brigham and R. E. Morrow, "The fast Fourier transform," *IEEE Spectrum*, vol. 4, no. 12, pp. 63–70, Dec. 1967.

[45] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[46] M. Pantic, N. Sebe, J. F. Cohn, and T. Huang, "Affective multimodal human-computer interaction," in *ACM Multimedia*, 2005, p. 669.

[47] H. Desurvire, M. Caplan, and J. A. Toth, "Using heuristics to evaluate the playability of games," in *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04*, 2004, pp. 1509–1512.

Annex – User questionnaire

Questionário FaceMe

Este questionário destina-se a avaliar a sua experiência com o jogo ImEmotion, inserido na tese de Mestrado de André Mourão. Estes questionários são anónimos.

---

* Obrigatório

Idade *

Sexo *

- Masculino
- Feminino

Frequência com que joga: *

	Nunca	Raramente	Semanalmente	Diariamente
Jogos casuais (jogos de Facebook, Miniclip)				
Jogos de computador ou consola				
Plataformas móveis (telefone, consolas móveis)				
Jogos interactivos (Microsoft Kinect, Nintendo Wii, Playstation Move)				

Questões gerais de jogo*

	Muito baixo	Baixo	Médio	Alto	Muito Alto
Clareza do objectivo do jogo					
Nível de diversão do jogo					
Dificuldades sentidas durante o jogo					

Rondas *

	Menor	Adequado	Maior
Tempo de exibição das imagens			
Número de imagens exibidas			

A versão que jogou era para dois jogadores. Qual considera ser o número adequado de jogadores para este tipo de jogo? *

- 1
- 2
- Mais de 2

Sentiu-se frustrado durante o jogo? *

	1	2	3	4	5	
Nunca						Sempre

Razões para essa frustração

De que aspectos gostou no jogo? *

	Odiou	Não gostou	Indiferente	Gostou	Adorou
Controlo com expressões					
Competitividade					
Novidade do tipo de controlo					

Outros aspectos que gostou ou não gostou

Controlo por expressões faciais

A sua pontuação correspondeu à expressão efectuada? *

	1	2	3	4	5	
Nunca						Sempre

A etiqueta da sua expressão ("Anger", "Sadness",...) correspondeu à expressão efectuada? *

	1	2	3	4	5	
Nunca						Sempre

Interacção afectiva

Qual a expressão mais difícil de realizar? *

- Neutral (Neutro)
- Anger (Raiva)
- Contempt (Desprezo)
- Disgust (Nojo)
- Fear (Medo)
- Happy (Feliz)
- Sadness (Triste)
- Surprise (Surpresa)
- Nenhuma

Qual a expressão que menos correspondeu à sua pontuação? *

- Neutral (Neutro)
- Anger (Raiva)
- Contempt (Desprezo)
- Disgust (Nojo)
- Fear (Medo)
- Happy (Feliz)
- Sadness (Triste)
- Surprise (Surpresa)
- Nenhuma

Sugestões