Université catholique de Louvain
Faculté des Sciences Appliquées
LABORATOIRE DE TÉLÉCOMMUNICATIONS
ET DE TÉLÉDÉTECTION

B - 1348 Louvain-la-Neuve Belgique

Mixed Reality Interactive Storytelling: Acting with Gestures and Facial Expressions

Olivier Martin

Thesis presented for the Ph.D. degree
in Applied Sciences

Jury:

Benoit Macq (TELE-UCL) - Supervisor
Thierry Dutoit (FPMS)
Jean Vanderdonckt (IAG-UCL)
Xavier Marichal (Alterface)
Michel Verleysen (DICE-UCL)
Luc Vandendorpe (DICE-UCL) - Chairman

May 2007

Contents

List of Figures

List of Tables

Remerciements

"Ceux qui réussissent sont souvent ceux qui ne se découragent pas..."
Jean-Paul Martin, 1996

Je voudrais commencer par remercier toutes les personnes que j'ai eu la chance de côtoyer lors de mon séjour à l'Université de Teesside, en Angleterre. La vie à Middlesbrough... disons que c'est vous qui l'avez rendue si agréable! Je pense surtout à *Jean-Luc, Sibo, Kimchi, Alex, Fred, Pascal,...*

Merci à *Marc Cavazza* pour son accueil chaleureux et professionnel, et pour tout ce qu'il a mis en oeuvre afin que mon séjour à Teesside soit le plus réussi possible. Merci pour tout ce que tu m'as appris, ce fut une vraie chance de travailler avec toi!

Merci à *Xavier Marichal* qui, dès le début, a cru en moi et m'a choisi pour effectuer ce travail de recherche. Tout au long de ma thèse, tu m'as guidé grâce à tes conseils avisés. Ton enthousiasme et ta passion pour le sujet furent une source constante de motivation.

Merci bien sûr à *Benoît Macq*, mon promoteur ! Benoît, tu es capitaine d'un grand navire et tu sais faire confiance à tes marins. Qui sait le trésor qu'ils découvriront demain ?... Merci pour ta confiance et ton positivisme sans faille! Malgré ton agenda surchargé, tu as toujours su garder une oreille attentive et donner les repères nécessaires pour que le bateau ne se perde jamais totalement dans le brouillard...

Merci aux "téléens" (*Kosta, Fix, Pedro, Benjamin, Jacek, Lionel,...*): vous avez rendu ma vie au labo très agréable. Une pensée spéciale pour *Fix*, avec qui j'ai partagé d'excellents moments de recherche... et de détente!

Merci à *Antoine Dejonghe*, un type bien, rusé et amical, qui m'a conseillé et soutenu tout au long de cette thèse. Merci pour ton soutien dans tous les moments de doute!

Merci à la *Région Wallonne*, qui a financé la présente recherche.

Merci à *Irene* pour cet énorme pot de café Frappé grec qui me fut bien utile dans les moments de faiblesse. Tu fus une alliée précieuse dans la réalisation de cette thèse ! Merci aussi de m'avoir aidé à corriger les dernières erreurs de ce manuscript.

Merci à *Jean Vanderdonckt, Thierry Dutoit* et *Michel Verleysen* pour tous les commentaires pertinents qui ont permis l'amélioration de ce manuscript.

Enfin, il y a tous les amis. Merci à vous tous d'avoir été à mes côtés durant ces années, pour boire un verre, écrire des équations et faire la fête (cherchez l'intrus)... Merci à vous tous ! (*Agnès, Anne-Sophie, Arash, Arthur, Astrid, Aurore, Balthazar, Benjamin, Bobby, Cap, Cédric, Daniel, Dimitri, Donatien, Florent, Franky, Fred, Gaspard, Gilles, Greg, Gorik, Isabelle, Julien d.M., Julien T., Manu, Olivier Magis, Marc, Nicolas C., Philippe, Robin, Sab, Sibo, Simon, Stéphane, Sylvie, Tanya, Tom, Vincent*, ... et tout ceux que j'oublie bien sûr !)

Puis, un Merci tout spécial à mes parents, *Jean-Paul et Christiane*, qui m'ont toujours soutenu dans ce projet de doctorat et dont l'amour inconditionnel et permanent est une source d'épanouissement depuis maintenant ... 30 ans! Merci également à ma grand-mère *Anne-Marie*, à mes beaux-parents *Michel et Jean-Claude* et à mon frère *Pascal*, sa femme *Françoise* et bien sûr... Merci à *Arnaud*, mon filleul, qui a inspiré une partie de l'introduction!

Enfin, j'en connais une qui doit se sentir oubliée ... Merci *Lilou*! A défaut d'être une muse inspiratrice, tes petits miaulements ont toujours su m'attendrir dans les moments difficiles!...

Hum... J'en connais une qui doit vraiment se sentir oubliée... Merci à toi *Eve-Laure*, merci 1000 fois! Merci d'abord de m'avoir suivi à Middlesbrough! Merci ensuite d'avoir été toujours là pour m'écouter et me soutenir de tout ton amour. Merci enfin pour tout ce bonheur que tu m'apportes au jour le jour. Merci de rendre ma vie si belle!

Abstract

This thesis aims to answer the following question : 'How could gestures and facial expressions be used to control the behavior of an interactive entertaining application?'. An answer to this question is presented and illustrated in the context of mixed reality interactive storytelling. The first part of this thesis focuses on the description of the Artificial Intelligence (AI) mechanisms that are used to model and control the behavior of the application. We present an efficient real-time hierarchical planning engine, and show how active modalities (such as intentional gestures) and passive modalities (such as facial expressions) can be integrated into the planning algorithm, in such a way that the narrative (driven by the behavior of the virtual characters inside the virtual world) can effectively evolve in accordance with user interactions. The second part of the thesis is devoted to the automatic recognition of user interactions. After briefly describing the implementation of a simple but robust rule-based gesture recognition system, the emphasis is set on facial expression recognition. A complete solution integrating state-of-the-art techniques along with original contributions is drawn. It includes face detection, facial feature extraction and analysis. The proposed approach combines statistical learning and probabilistic reasoning in order to deal with the uncertainty associated with the process of modeling facial expressions.

Résumé

Cette thèse a pour but de répondre à la question suivante: 'Comment interagir avec une application interactive au moyen de gestes et d'expressions faciales?'. Une tentative de réponse à cette question est présentée et illustrée dans le cadre de la narration interactive en réalité mixte. La première partie de cette thèse se concentre sur la description des techniques inhérentes à l'intelligence artificielle, permettant de modéliser et de contrôler en temps réel le comportement d'une application interactive. Nous présentons un algorithme efficace de planification d'actions en temps réel et montrons que les interventions de l'utilisateur, utilisant des modalités actives (tels que les gestes intentionnels) et passives (telles que les expressions faciales), peuvent être intégrées dans l'algorithme de planification de telle manière que la narration (constituée par les actions entreprises par l'ensemble des personnages peuplant le monde virtuel) puisse évoluer de manière cohérente avec les interactions entre l'utilisateur et l'application. Après avoir brièvement décrit une solution simple mais efficace au problème de la reconnaissance automatique de gestes, l'accent est mis sur la reconnaissance des expressions faciales. Une solution complète intégrant des contributions personnelles à l'état de l'art est présentée. Cette solution comprend la détection automatique du visage ainsi que l'extraction et l'analyse en temps réel de la forme des yeux, des sourcils et de la bouche pour la reconnaissance d'expressions faciales. L'approche proposée combine apprentissage statistique et raisonnement probabiliste, de manière à prendre en compte de manière efficace l'incertitude associée à la modélisation des expressions faciales.

Chapter 1

Introduction

In the past decades, video games have been evolving from very basic single-player monochromatic games to huge virtual worlds containing hundreds of thousands of human players, able to play simultaneously (*Massively Multiplayer Online Role-Playing Game* (MMORPG)). Virtual communities of players can now interact through modem-based communications using voice and text messaging to develop group strategies and make the game evolve towards new directions. Virtual worlds are populated by virtual characters, which are controlled by human players located anywhere on the planet! Figure 1.1 illustrates this evolution.

Figure 1.1: The left part of the figure shows the screen of one of the first video games, developed by Atari™ in the late seventies. The right part of the figure shows a screenshot of a popular video game 30 years later.

This extraordinary evolution is the consequence of several factors. A primarily one is the tremendous increase of the computational capabil-

ities of gaming consoles and personal computers. This explosion of the computational power had an important impact on the type of applications that were developed. Early low-complexity video-games immersed the user in simple worlds, in which the story evolves according to user actions. For these early games, only a way of *commanding* the application was needed. The user was acting as a *director* who had full control on the behavior of the application. For these types of video games, basic control pads (or joysticks) were satisfactory as games were developed in such a way that the actions were designed to correspond to specific combinations on the control pad.

Recent years have seen the emergence of new types of video games involving several players and thereby having narratives unfolding according to the behavior of several entities. More and more, the player is loosing the status of director to slip on the status of *actor*. Whereas a director has total control on the narrative, an actor has only *partial control*. Narratives of modern games thus evolve according to the actions of the characters taking part in the game.

When analyzing the communication involved in modern video games, we quickly understand that actors need new types of interfaces to play. The ideal interface should allow the user to interact with other characters as if they were humans. As a matter of fact, most of the characters are avatars which represent anthropomorphic creatures controlled by human players, so it seems quite logical that the user should be able to talk to them, as well as to use facial expressions, body gestures and vocal intonation to communicate...

It is interesting to start our exploration of the ideal interface by studying the way humans communicate between each other. At a first glance, it could be thought that humans communicate through a codified spoken language. Each of the spoken languages contains a huge set of words that can be combined to express any ideas and feelings. If we dig a little bit further however, it is obvious that humans also use other channels than spoken language to communicate. These other channels are, for instance: facial expressions, body gestures, postures and prosody.

Professor Albert Mehrabian, one of the most famous researchers in the field of non-verbal communication, argues that, on average, facial

expressions and vocal intonation account for around 93% of an effective spoken message and that the verbal component conveys a small 7% of the information[1]. Of course, this percentage depends very strongly on the type of communication that is involved but the idea is that non-verbal communication contains much more information than it is often thought.

If we think about how humans learn to communicate, we actually reach the same conclusion as Mehrabian: spoken language is not the primary channel used by humans to communicate. As a matter of fact, babies first learn to recognize facial expressions and intonations, well before being able to understand a single word. They also first learn how to express themselves through facial expressions: they smile to express positive feelings and cry to express negative ones! My godchild, Arnaud, is eight months old. Of course, he can not communicate with verbal messages yet, but sometimes, a situation arises and he looks around the people's faces, searching for information. If he sees other people smiling, he understands that the situation is funny and a big smile appears on his face, whereas he would have simply gone on staring if people would have stayed neutral.

Back to the case of human-computer applications, we will go a step further and argue that the introduction of non-verbal communication into the interface will make the application truly intelligent and that, in many cases, we cannot talk about machine intelligence without incorporating affective aspects of human communications. To illustrate this statement, let us consider a basic example that demonstrates our assumption. If we analyze the behavior of a commercial robot whose aim is to present the last commercials of a department store to clients entering the store, the robot could only be considered intelligent if he could understand whether or not the client is interested, bored or even irritated by its presence. Imagine how the appropriate reaction changes when the user is in one of these three different emotional states! The intelligence of the robot comes from its ability to adapt its behavior according to the perception of its user, and facial expressions and vocal intonation convey the affective information needed to build that perception. Thousands of similar examples can be found and in most of the cases, intelligence can only be obtained if non-verbal information is

incorporated into machine's behavior, simply because non-verbal information constitutes a major channel used by humans to convey information and that machines cannot ignore it if they want to be considered as 'intelligent'.

The considerations that have been developed so far force us to realize that *intelligent* human-machine applications will with no doubts need to incorporate facial expressions and vocal intonations in order to be able to effectively understand their human users. Major advances have been achieved in the area of non-verbal behavior recognition, but most of the existing systems are not reliable enough to be introduced in real-world applications. To a smaller degree, the same considerations apply to the synthesis of non-verbal communications. Robots or virtual characters have a lack of naturalness when they are not able to express emotions. This is a crucial aspect of today's video games where the quality of animated scenes often suffers from the absence of emotions on the characters' faces.

It is within this context that this thesis has been developed: the purpose is to show how non-verbal modalities, and more specifically facial expressions, can be integrated into practical interactive video games, using only common devices such as a video camera with a microphone input. In the scope of this thesis, we limited ourselves to image modalities and therefore only analyzed the integration of face and body gestures into an interactive application. The principles can however easily be extended to the integration of additional modalities.

1.1 Organization of the thesis

This thesis is organized in six chapters. The first chapter introduces the context that motivates the present research, along with a list of the original contributions of the thesis and the publications of the author that are related to the present research.

The second chapter presents a solution to the problem of planning in real-time the behavior of virtual actors within a mixed reality environment. It describes how such a mixed reality environment can be built, and how the behavior of the characters living in this environment

can be represented by hierarchical trees and controlled through a dedicated real-time planning algorithm. It also demonstrates how the user of the application can interact with virtual agents/objects inside the scene and thereby modify the course of the story. An interactive James Bond application illustrates how gestures and speech can be used as communication channels between the user and its application.

The third chapter is dedicated to body gesture recognition. We show how a set of semiotic gestures can be detected by a simple rule-based system, built on top of a body segmentation engine. The presented system provides real-time robust detection of a set of five 2D-gestures.

The fourth chapter focuses on the methods that have been deployed to tackle the problem of automatic extraction of relevant affective information from a video sequence. This includes face detection as well as facial feature detection and tracking. We explain that chrominance can be used as the primary discriminant feature for skin region detection. We show that, by adding shape and texture information, we can achieve an effective detection of the face contour. We then focus on facial feature detection from the extracted face image and demonstrate the effectiveness of an approach combining traditional boosted detectors with a priori statistics on the face morphology. We conclude the chapter by presenting a method based on deformable face models to achieve the real-time tracking of the facial features.

The fifth chapter concerns the facial expression recognition problem itself. The geometrical description of a set of facial features forms the input of the classifier, while the output consists of a set of six facial expressions. Among the different possible approaches to facial expression recognition, we justify the choice of statistical machine-learning techniques such as Bayesian Networks (BN) and Support Vector Machines (SVM), as the best approaches to deal with the uncertainty associated with the process of modeling emotions. We comment the probability density functions of 8 facial deformations given 6 possible emotions and present the classification results obtained with the different classification algorithms, when these 8 facial deformations are used as inputs of the classifiers.

Finally, the last chapter draws the conclusion by analyzing the presented solutions and gives an outlook for future research directions.

1.2 Original Contributions

In this section, we present the original contributions of this thesis. Figure 1.2 shows the overall system, which includes work developed in this thesis as well as external contributions. In this figure, the gray boxes represent the topics containing original contributions, while white boxes correspond to areas that do not contain significant contributions from the author.

The original contributions of this thesis are presented hereunder, in the order in which they appear in the thesis:

1. **A Hierarchical Planning Engine**, enabling the story to evolve to new directions according to user's active and passive intentions. The planning algorithm has been initially conceived at the University of Teesside, but has been completely reengineered in C++, following a modular object-oriented implementation. Unlike the Teesside implementation, the selection of the child node to explore (in the case of OR-node) is based on a heuristic function[1], which can either be deterministic or stochastic.

2. **An Affective Storytelling Scenario**. Gestures and facial expressions may be used to control the behavior of a mixed reality interactive storytelling application. A collaboration with the team of Professor Marc Cavazza from the University of Teesside already led to a first prototype, including speech and gesture modalities. We further extended this concept by incorporating affective interactions and present a first scenario of such an affect-driven story generation system.

3. **An Original Rule-Based Gesture Recognition System**, which allows robust real-time action detection is presented. Developed in C++, it takes as an input the results of a 2D body segmentation engine and recognizes five different gestures. It also provides an estimation of the distance between the user and the video camera as well as the computation of the directional vector when pointing gestures are detected. Finally, the system includes a TCP/IP

[1]Instead of predetermined cost values for each children, as in the Teesside implementation

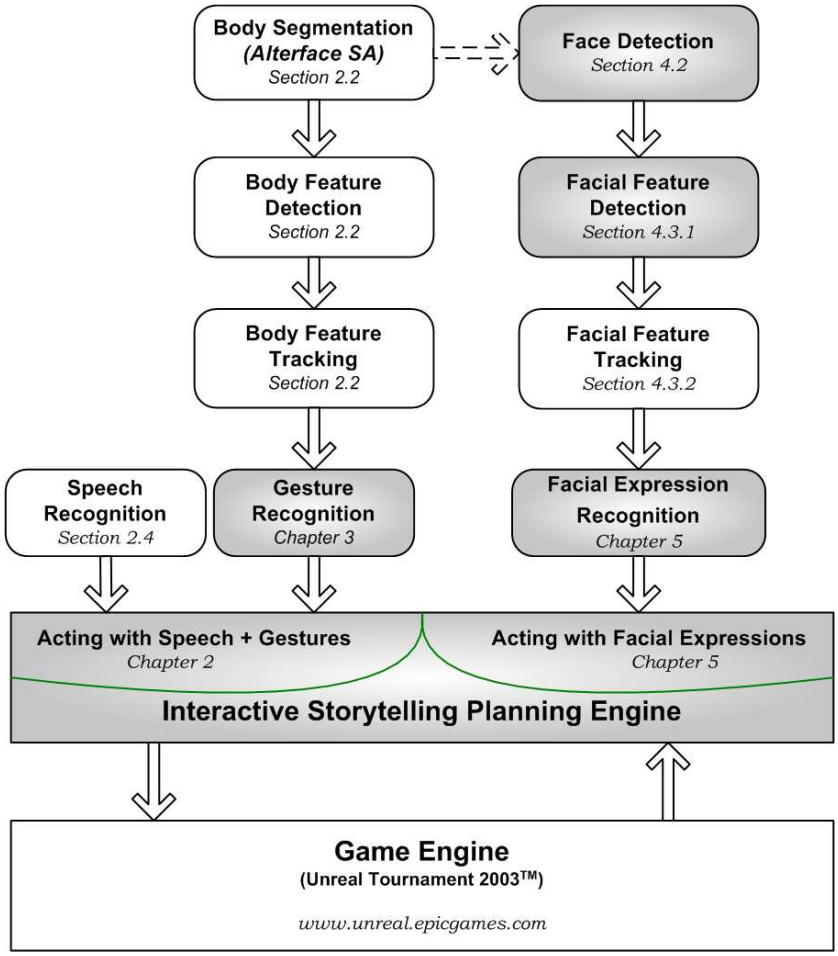


Figure 1.2: **Thesis outline**: gray boxes correspond to topics that have been investigated and which contain significant contribution from the author

.

client, which is designed to connect the system to external modules, such as a planning engine.

4. **A Multimodal Action Recognition** system. Gestures are used to remove the ambiguity in the speech signal (such as for instance

in the case of pointing gestures, in which the direction of pointing removes the ambiguity associated with the determination of the object of the interaction). Conversely, the speech signal has often been used to remove the ambiguity associated with the interpretation of the gesture, thereby enabling the same gesture to have different meanings depending on the semantic content of the speech act.

5. **A Face Detector**. The detector provides an estimation of the face contour, based on luminance and chrominance information. It provides hue-based filtering to detect potential face regions and integrates constraints on the shape of the face to be detected in an edge detection approach. It has been tested with success on a variety of faces from very different ethnicities.

6. **A Statistical Study of Facial Feature Deformations**, as induced by 6 different emotions, is presented. The relative displacements of a set of facial feature points for each of the emotions are used to build statistical models of facial expressions. The learning process does not assume the data to follow a gaussian distribution. Instead, estimations of the true probability distributions are computed and extensively commented.

7. **A Novel Algorithm for Facial Feature Detection** is introduced. It uses both histogram-based and gradient-based techniques, but only inside local search-spaces, computed statistically from a large face image database. The detection is obtained through the fusion of three independent facial feature detectors. The result of the detection algorithm is used to initialize a facial feature tracking algorithm, based on a deformable face model.

8. **Bayesian Network and Support Vector Machines for Facial Expression Recognition** are widely investigated. We justify our choice for a probabilistic approach and compare the results that were achieved on a large facial expression database.

1.3 List of Publications

This section lists the publications of the author related to the content of this thesis.

1.3.1 Journal Papers

1. O. Martin, I. Kotsia, A. Savran, A. Huerta, R. Sebbe, J. Adell, I. Pitas: **Synthesis of Expressive Facial Animation: A Multimodal Caricatural Mirror**, *Journal on Multimodal User Interface, Vol. 1 (to appear), 2007.*

2. O. Martin, F-X. Fanard, I. Kotsia, I. Pitas, B. Macq: **Facial Expression Analysis Using Statistical Models of Faces**, *submitted to IEEE Transactions on Circuits and Systems for Video Technology, 2006.*

3. M. Cavazza, O.Martin, F. Charles, S.J. Mead, X. Marichal, A. Nandi : **Multi-modal Acting in Mixed Reality Interactive Storytelling**, *IEEE Multimedia Magazine, July 2004.*

1.3.2 Conference Papers

1. F-X. Fanard, O. Martin, B. Macq: **Statistical Facial Expression Analysis for Realistic MPEG-4 Facial Animation**, *11th International Conference "Speech & Computers" (SPECOM'06), St Petersburg, Russia, 2006.*

2. O. Martin, F-X. Fanard, B. Macq: **From Feature Detection to Facial Expression Recognition: An Integrated Probabilistic Approach**, *7th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'06), Incheon, South Korea, 2006.*

3. O. Martin, I. Kotsia, B. Macq, I. Pitas: **The eNTERFACE'05 Audio-Visual Emotion Database**, *IEEE Workshop on Multimedia Database (IEEE MDDM'06), Atlanta, 2006.*

4. O. Martin, I. Kotsia, A. Savran, A. Huerta, R. Sebbe, J. Adell, B. Macq: **A Multimodal Caricatural Mirror**, *eNTERFACE'05 -*

Proceedings of the First Summer Workshop on Multimodal Interfaces, pp. 13-20, 2005.

5. F. Charles, O. Martin, M. Cavazza, S.J. Mead, X. Marichal, A. Nandi : **Compelling Experiences in Mixed Reality Interactive Storytelling**, *International Conference on Advances in Computer Entertainment Technology (ACE'04), Singapore, 2004.*

6. M. Cavazza, O. Martin, F. Charles, X. Marichal, S.J. Mead : **User Interaction in Mixed Reality Interactive Storytelling**, *The Second IEEE & ACM International Symposium on Mixed and Augmented Reality (ISMAR'03), Tokyo, Japan, 2003.*

7. M. Cavazza, O. Martin, F. Charles, S.J. Mead, X. Marichal : **User Acting in Mixed Reality Interactive Storytelling**, *International Conference on Virtual Storytelling (ICVS'03), Toulouse, France, 2003 (**Best Paper Prize**).*

8. M. Cavazza, O.Martin, F. Charles, S.J. Mead, X. Marichal : **Interacting with Virtual Agents in Mixed reality Interactive Storytelling**, *Intelligent Virtual Agents, Irsee, Germany, 2003.*

Chapter 2

Real-Time Planning of Interactive Storytelling

2.1 Introduction

Interactive storytelling is a new kind of computer entertainment. If we consider the user's perspective, it could be situated somewhere between *video games*[1] and *movies*. In a video game, the user has a role of *director*: the story evolves to follow the choices and actions undertaken by the user in a quite straightforward way. We could view video games as a media in which the user has *total control* on the story unfolding. In the opposite way, movies are a type of media that does not involve the user in the story unfolding process. The user simply watches it with no mean of influencing the narrative, thus having *no control at all* on the story unfolding. In interactive storytelling, the user is seen as *an actor* who plays a part in the story. As in this case, the narrative depends on both the behavior of the characters playing part in the story and the behavior of the user itself, we say that the user has *partial control* on the story unfolding process.

More formally, the term *interactive storytelling* was first coined by

[1] By video games, we refer here to single-player video games, in which the player *commands* the application. New generations of video games do not follow this paradigm as the behavior of the application evolves more and more according to the actions of *many* different players.

Chris Crawford, a main proponent and developer of video games. He defines interactive storytelling as: *a form of interactive entertainment in which the player plays the role of the protagonist in a dramatically rich environment*[2]. Interactive storytelling thus aims at immersing users in fantasy worlds in which they can play parts in evolving narratives that respond to their interventions.

Implementing the interactive storytelling concept thus involves mainly three computing technologies: virtual or mixed reality for creating the artificial world, artificial intelligence techniques for generating the narrative in real time and an action recognition module for understanding user interactions with the system. The three next sections of this chapter will focus on the technical challenges that are related to these aspects. We will illustrate the theoretical developments with practical examples extracted from our first Mixed Reality Interactive Storytelling (MRIS) prototype.

In this thesis, we consider as a first case an application involving one user, whose silhouette is extracted from the background and inserted within a virtual world, to create a mixed reality environment. The extracted silhouette is used for rendering purposes, but also provides information to an analysis module whose goal is to detect a set of semiotic gestures. The detected gestures are then interpreted and considered as intentional actions from the user. Beside this gesture recognition component, a speech processing module uses keyword-spotting techniques to grasp verbal information from the user's utterances. The resulting prototype is thus able to behave according to both user's gestures and speech using efficient multimodal fusion algorithms.

In a second application, we present an example of so called *affect-driven interactive storytelling* in which, by associating an emotional state vector to each character, we can create entities that behave according to their emotional state, using heuristics based on affective information. We demonstrate their use on the interactive James Bond prototype presented in section 2.4.

Eventually, the reader who is interested in the use of facial expressions as a way to impact the storyline is invited to consult chapter 5, dedicated to facial expression recognition: section 5.6 investigates the

integration of the *user*'s facial expressions, as a new modality to control the narrative.

2.2 Creating a Mixed Reality Environment

Mixed Reality (MR) is the merging of real and virtual worlds to produce a new environment where physical and digital objects can co-exist and interact. Paul Milgram situates MR as somewhere along the 'virtuality continuum' which connects completely real environments to completely virtual ones [4].

In the proposed prototype, the entire body of the user is extracted from the real environment and inserted in a virtual world, populated with virtual agents with whom the user can interact. The segmentation of the user's silhouette is performed by the Salto™ engine [2], developed by Alterface SA. The Salto™ system uses a Walsh-Hadamard transform on 4 x 4-pixel elements. Sliding the box of two pixels aside allows taking decision on 2 x 2-pixel blocks. As a result, it can segment and adequately detect the user's silhouette. Figure 2.1 gives an overview of the change-detection process with the Walsh-Hadamard transform. First, the detection module calculates the background image's Walsh-Hadamard transform. It then compares the transform's values for the current and background images. When the rate of change is higher than an established threshold, the module sets the area as foreground. As a result, we obtain an image composed of the foreground representing the user's silhouette and a background whose pixels are set to a predefined value (we used black pixels to represent the background).

After being extracted from its background, the user's silhouette is mixed with a 3D graphic model of a virtual stage including the synthetic characters taking part in the story. The resulting image is projected on a large screen facing the user, who sees his own image embedded in the virtual stage with the synthetic actors [5]. The mixed reality environment thus creates both a visual presentation of the story embedding the user and a virtual stage for the interactive story.

[2] At the time our system was developed, the engine developed by Alterface was called Transfiction™ . Since that time, it has integrated new technologies and is now called SALTO™

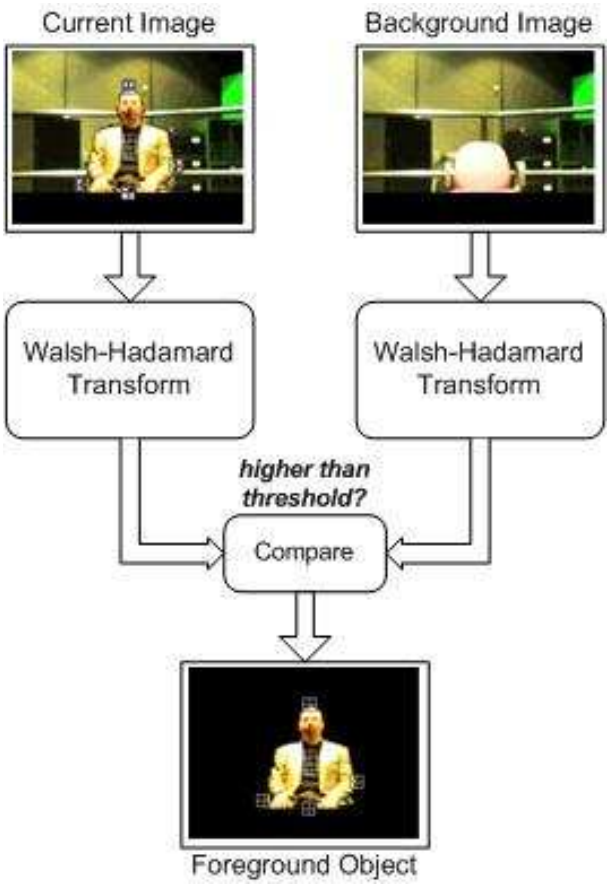


Figure 2.1: Extracting user's silhouette from background

The graphic component of the Mixed Reality world is based on a game engine, Unreal Tournament 2003™ . This engine not only performs graphic rendering and character animation but, most importantly, contains a sophisticated development environment to define interactions with objects and characters' behavior [6]. In addition, it supports the integration of external softwares, e.g. through socket-based communications.

Technically speaking, the composition of the mixed reality environment is obtained through the mixing of the video channels captured by a separate computer [7], running a DirectX™-based application. The first stage consists in isolating the user's image from its background using basic chroma-keying. The remaining stage attempts to solve the problem of occlusion by blending the user's image with the virtual environment's image using empirical depth information provided by the gesture recognition module as a user's relative distance to the video camera, and by the game engine itself for the virtual environment. Figure 2.2 depicts the overall processing: the several video image layers are composited in real time to produce the final image which is projected onto the screen facing the user.

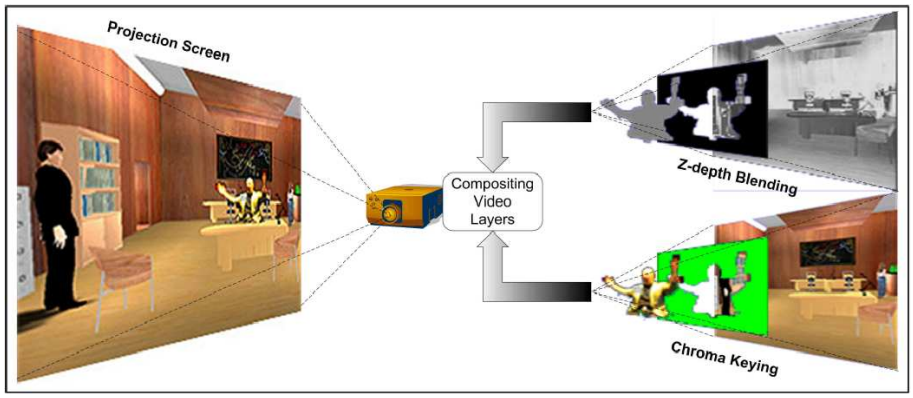


Figure 2.2: Constructing the Mixed Reality Environment

The two system components operate by sharing a normalized system of coordinates. This shared coordinate system makes possible to position the user in the virtual image, but most importantly to determine

the relations between the real user and the virtual environment. This is achieved by mapping the 2D bounding box produced by the Salto™ engine, which defines the contour of the segmented user character, to a 3D bounding cylinder in the Unreal Tournament 2003™ environment, which represents the position of the user in the virtual world (Figure 2.3) and, relying on the basic mechanisms of the engine, automatically generates low-level graphical events such as collisions and interactions with objects. The object interaction is further represented within the virtual environment by mapping the position of the leftmost and rightmost markers to a small-scale bounding cylinder, which allows for more precise collisions with objects that can be manipulated by the user within the virtual environment.

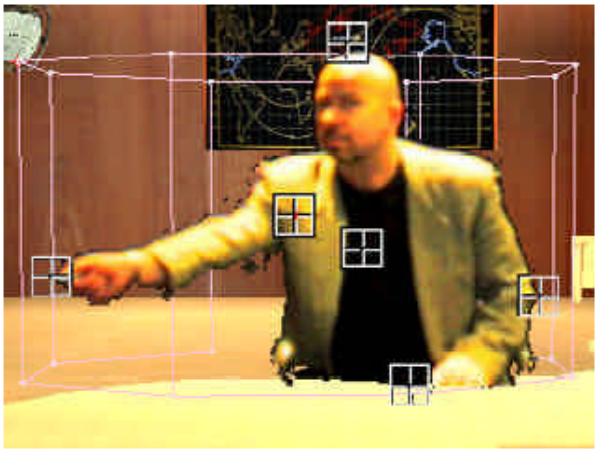


Figure 2.3: The 3D bounding cylinder determines physical interactions in the Unreal Tournament 2003™ engine

The two sub-systems communicate via TCP sockets: the image processing module, working on a separate computer sends at regular intervals to the graphic engine two different types of messages, containing updates on the user's position as well as any recognized gesture. The

position of the user is represented by five points, which are: the highest point, the lowest, the rightmost, the leftmost and the center of gravity of the segmented silhouette, as well as of the bounding box. The recognized gesture is transmitted as a code for the gesture (plus, when applicable, e.g. for pointing gestures, a 2D vector indicating the direction of pointing).

2.3 Real-Time Planning of Virtual Agents

In the context of this thesis, we will focus on character-based interactive storytelling, in which the evolution of the narrative is driven by the actions undertaken by both the agents populating the mixed reality environment and the user itself. We will focus on single-player applications, which means that the world is populated by virtual characters and one human player, whose silhouette is integrated in the virtual stage. The behavior of the virtual agents must respond to two conditions: the virtual agents must have their own role in the story in order to create an engaging and entertaining environment, and they also have to be able to react appropriately to user interventions to create interactivity.

If we consider the case of very small-scale interactive storytelling applications, the behavior of the virtual agents may be handled through a basic decision engine: the virtual characters can execute only a very small number of different actions and the choice of the action to be executed next is driven by a small set of rules. For larger scale applications however, the number of possible actions leads to an increase of the computational complexity. This makes simple rule-based decision engines intractable for real-world applications. One of the most pervasive ideas to deal with this issue is hierarchical decomposition, which reduces the complexity by segmenting the original problem into smaller individual ones. The key benefit is that, at each level of the hierarchy, the computational task is reduced to a small number of activities at the next lower level, so that the computational cost of finding the correct way to arrange those activities for the current problem is small. In the frequent cases in which high-level solutions have satisfactory low-level implementations, hierarchical methods can result in linear-time instead of exponential-time planning algorithms.

In the remaining of this section we will present the formalism that is used to represent the role of the virtual agents in a hierarchical manner. We will see that the goal of a virtual agent can be decomposed into basic executable actions and that by building sequences of these basic actions, we can create complex behaviors. The process of managing the behavior of the virtual agents can then be seen as *finding the optimal sequence of actions to achieve a predetermined goal, while simultaneously making sure that the behavior of the agents stays coherent with user's interventions*. We will show that this problem can be solved by an *online hierarchical planning* algorithm, which we will present and illustrate with two practical interactive storytelling applications.

2.3.1 Hierarchical Task Networks

To choose among the different task models, we considered the requirements imposed by the real-time constraints, inherent to interactive applications, as well as the desire to use a structure that provides the foundations for the implementation of efficient online planning algorithms. Among the available task models, a hierarchical representation, such as the Hierarchical Task Networks (HTNs), would particularly fit our needs. Even though it is by far not the most expressive of the task models [8], it is the one with the lowest complexity and it is sufficiently expressive for our applications[3]. HTNs are an application of AND/OR graphs, whose first use in user interface design was realized by Kleyn et. al [9]. HTNs were then first applied to interactive storytelling in 2002 by Cavazza et. al [10].

The HTN formalism is used to represent a problem or a goal in a hierarchical manner. The problem is represented as a tree. The root node of the tree contains the overall problem. In our case, the root node contains the definition of the goal of the agent inside the virtual world. This main goal is then decomposed into several subgoals, which are represented in the HTN by the root's children. These subgoals are further decomposed into subsubgoals and so forth. The decomposition process goes on until a primitive action level is reached. This primitive

[3]The HTN formalism is not able to handle explicitely time-varying parameters, but those can still be used by adding precondition variables, whose values are allowed to change over time

action level corresponds to actions or problems that can not be decomposed anymore. In the case of our interactive storytelling application, the primitive action level contains actions that can be directly executed by the game engine and whose outcome can be determined as SUCCESS or FAILURE.

At any level of the hierarchy (except at the primitive action level), there exist two different ways to decompose a goal into several subgoals. In the first decomposition mode, the goal can be solved using different possible alternatives. In this type of decomposition, each child represents a distinct solution and the parent node is solved if any of the children nodes is solved. This type of decomposition is called OR-decomposition, and the corresponding nodes are called *OR-nodes*. In the second type of decomposition, the goal can be achieved by solving a sequence of successive subgoals. In this case, the goal can only be solved when *all* the children nodes are solved. This type of decomposition is called AND-decomposition , and the corresponding nodes are called *AND-nodes*.

LEAVES contain primitive actions and have by definition no children. The set of leaves contained in a HTN represents the set of all possible actions that a virtual agent may undertake. It is the sequential combination of these primitive actions that can create complex behaviors, as it will be illustrated later in this chapter.

To summarize, each node of a HTN tree can be of three different types:

1. OR-nodes: The parent node is solved if at least one of the children nodes is solved.

2. AND-nodes: The parent node is solved only if all the children nodes are solved.

3. LEAVES: A node containing a primitive action

In the HTN graphical representation, AND-nodes differ from OR-nodes by the presence of a circular arc below the node, as depicted in Figure 2.4.

To illustrate our definitions with a practical example, we will consider the problem of someone who would like to get a car. Let us assume that

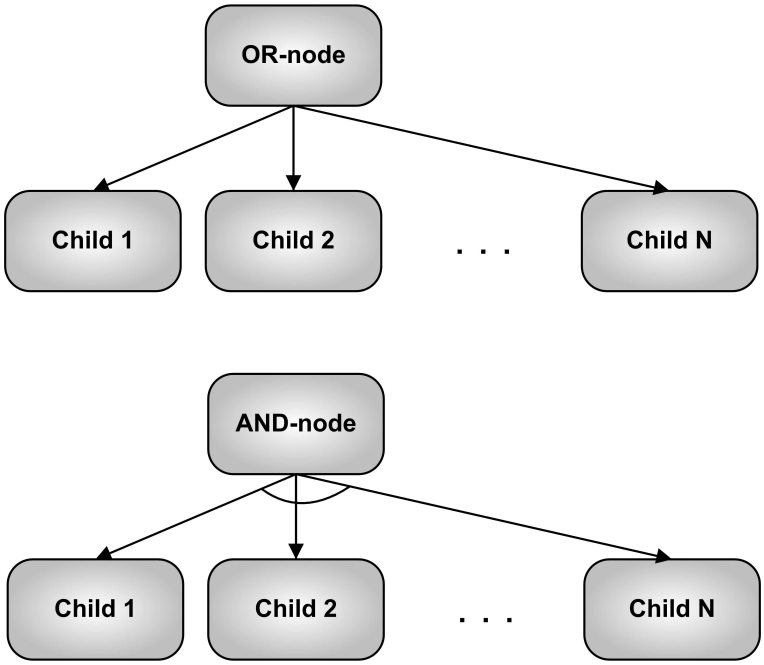


Figure 2.4: Graphical representation of OR-nodes and AND-nodes in the HTN formalism

to achieve its goal, the person may choose between buying or stealing a car. If we choose to analyze further the most honest of these options, we could decompose the buying process in a sequence of three distinct steps: find the money to buy the car, choose which car to buy and make the transaction (pay for the car). We can then just introduce two ways to get the money for the car: through cash payment or by contracting a loan. Eventually, we will consider two different types of car to choose from: used or new cars.

In the HTN formalism, the problem Get Car can be represented by an OR-node whose children are Buy Car and Steal Car. The node Buy Car can itself be decomposed using an AND-node with three children Get Money, Choose Car and Pay Car. Among these three children, the node Pay Car is a primitive action that cannot be decomposed. Following the definition of the problem as stated in the preceding paragraph, the node Get Money can be OR-decomposed into the two primitive actions: Get Cash and Get Loan, while the node Choose Car can be decomposed into Choose New Car and Choose Used Car. Figure 2.5 depicts the HTN that corresponds to our problem. As can be noticed on the figure, the HTN formalism provides a clear and compact representation of the problem.

So far, we have seen that a HTN is a compact and efficient way to represent a problem or a goal in a hierarchical fashion. The main advantage of the HTN formalism however lies elsewhere. As a matter of fact, it provides the hierarchical structure onto which a simple and efficient real time behavior planning can be built. The next section is therefore devoted to the presentation of this hierarchical planning algorithm and to its illustration with a practical example.

2.3.2 HTN Planning

The preceding section described how a complex problem can be decomposed into smaller problems using hierarchical decomposition. These smaller problems (or subproblems) can themselves be decomposed into even smaller problems and so forth, until we reach a level that corresponds to executable primitive actions.

In the particular case of character-based interactive storytelling, a

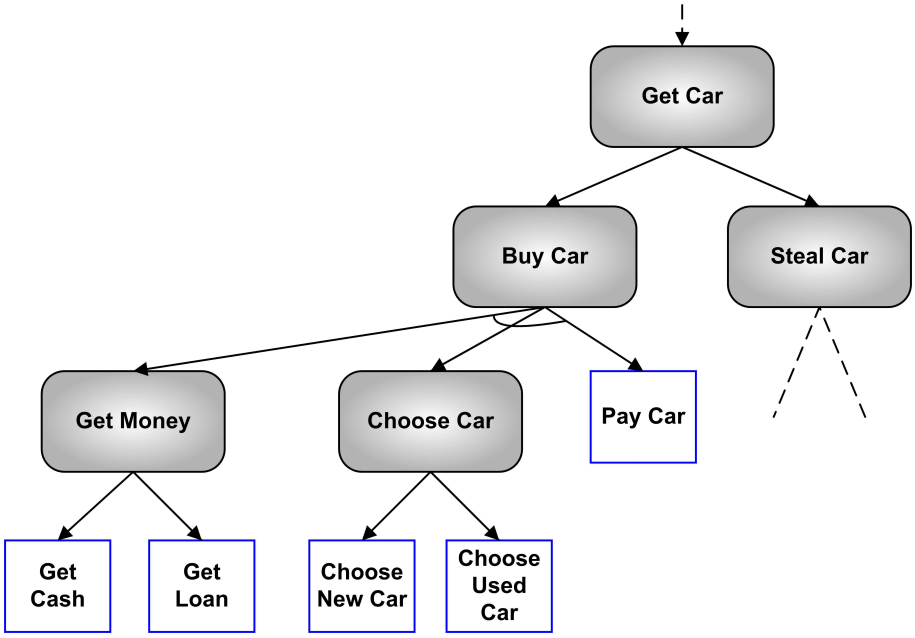


Figure 2.5: The HTN formalism provides a clear and compact representation of the problem

HTN is used to represent the role of a virtual character as a tree, whose leaves correspond to the primitive actions that can be directly executed by the character. The goal of the planning algorithm is to find the optimal sequence of primitive actions that allows the character to play its role while continuously adapting its behavior to user interactions. In order to achieve this, the planning must be done in real time, in such a way that both the outcome of primitive actions and user interventions are taken into account to decide which action should be executed next. Because user interventions will have as consequences either the modification of a node's preconditions or the modification of the outcome of an action being executed, the constraint implies that action planning and action execution should be interleaved.

The algorithm that is commonly used to search into AND/OR graphs is called the AO* algorithm, and it is detailed in depth in [11]. The version that we present in this thesis is a modified version of the original AO* algorithm, which allows action execution to be interleaved with the planning itself. As the integration of user's behavior into the planning process is the object of the next section, we will now focus on the planning algorithm itself, whose mechanisms are explained hereafter.

A HTN can be defined by a set of nodes N_i with $i \in [1, \ldots, M]$, M being the total number of nodes. For a node N, the set of all children is denoted by $Children(N)$, while its parent is represented by $Parent(N)$. To each node is associated a set of preconditions $Precond(N)$. Each node also has three possible states: $S(N) =$ SOLVED, FAILED *or* UNKNOWN and can be of three different types: $T(N) =$ OR, AND *or* LEAF. The root node R is the only node that has $Parents(N = R) = \emptyset$. Inversely, leaves are the only nodes to satisfy $Children(N) = \emptyset$.

When the application is started, the algorithm first initializes the HTN. All the nodes are set to the UNKNOWN state and the current node C is positioned on the root node: $C = R$. Once launched, the algorithm will run until the root node either gets the SOLVED status(character's goal is achieved) or FAILED status (the character has failed to achieve its goal).

The first step of the algorithm (the ***exploration*** step) consists of a *depth-first search (DFS)*: the algorithm starts from the root node ($C =$

R) and selects a path from the root node to one of the leaves of the three. This leaf corresponds to the first action to be executed by the character. To find and select a path from the root node R to a leaf L, the algorithm has to explore a certain number of nodes.

The exploration step for the current node C, $Explore(C)$, consists of the following four steps:

1. `if` $Precond(C)$=`true`: $Continue$, `else`: $\{Failed(C)$ and $Break\}$
2. `if` $T(C) ==$ LEAF: $\{Execute(C)$ and $Break\}$, `else` $Continue$
3. Select child to explore: $C = Select(Children(C))$
4. $Explore(C)$

First, the algorithm checks the preconditions associated with the node under consideration. If at least one of the preconditions is not satisfied, the node receives the status FAILED and the exploration process is interrupted. If all the preconditions are satisfied, the algorithm tests if the current node is a leaf. If this is the case, the algorithm stops its exploration step and executes the action corresponding to the current leaf node. If the current node is not a leaf, the exploration process goes on by choosing the next node to be explored next, among current node's children.

To understand how to select among a node's children which one should be explored next, we have to consider the type of node that is considered. For an OR-node, each child represents a possible solution. A cost function h thus computes the cost associated with the M_{Child} possible alternatives:

$$\forall N_i \in Children(C): \qquad h_i = h(N_i) \tag{2.1}$$

This cost function h may be any deterministic or stochastic function. Let us focus on how the $Select(Children(C))$ function chooses which child to explore next. Actually, it simply selects the node whose cost is minimal:

$$Return\ N_k \qquad with \quad h_k = \min(h_1, ..., h_{M_{Child}}) \tag{2.2}$$

For AND-nodes, the process is easier as the child to be chosen is always the first whose status is set to `unknown`. In this case, the function $Select(Children(C))$ indeed parses the children of current node C in the order in which they appear in the graphical HTN representation and returns the first child which has not been explored yet.

This concludes the presentation of the *exploration* step. We have seen how the algorithm is initialized and how, starting from the root, it selects a path that leads to a leaf, *unless* if one of the nodes encountered has its preconditions unsatisfied. At this stage, if this happens, the node is marked as FAILED and the exploration step is interrupted. We will see that as soon as a node gets a new status, the algorithm needs to *propagate* the node's new status to its parents before the exploration can resume.

As we just mentioned, if a node's preconditions are not satisfied during the exploration step, the search is interrupted, the node gets the FAILED status and this new status needs to be propagated upwards. The same happens when the exploration step reaches a primitive action: the action is executed and the result (either SUCCESS or FAILURE) changes the status of the leaf containing the executed action. In this case, the new status also needs to be propagated upwards. Let us therefore now describe the other part of the algorithm: the ***propagation*** step.

The propagation rules differ according to both the type of the parent node and the new status that has to be propagated. Four cases have to be investigated:

1. The new status is SOLVED and the parent node is an OR-node
2. The new status is FAILED and the parent node is an OR-node
3. The new status is SOLVED and the parent node is an AND-node
4. The new status is FAILED and the parent node is an AND-node

To understand how the new status will propagate upwards in the tree, we need to remember that an OR-node is solved if *one* of its children is solved. An AND-node is solved only if *all* its children are solved.

Taking this into account drives us to realize that the result is straightforward in the first and fourth case, while it is a bit more complicated in the second and third case.

In the first case, the parent node receives the SOLVED status because one of its child received the SOLVED status. In the fourth case, the parent node received the FAILED status, because as soon as one of its children receives a FAILED status, the node cannot be solved anymore. In the second case, the parent node will get the status FAILED if he does not possess at least another child whose status is UNKNOWN. If he does have a child whose status is still UNKNOWN, the parent node's status does not change and the propagation step terminates. Finally, in the third case, the parent node will get the SOLVED status only if all its other children already have the SOLVED status. In the opposite case, the parent node keeps the UNKNOWN status and the propagation step ends. To summarize, the propagation rules for a node C, in the four cases described above, are the following:

1. $S(Parent(C))=$ SOLVED
2. `if` $\forall N \in Children(Parent(C)) : S(N)=$FAILED $\Rightarrow S(Parent(C))=$ FAILED `else`: Break
3. `if` $\forall N \in Children(Parent(C)) : S(N)=$SOLVED $\Rightarrow S(Parent(C))=$ SOLVED `else`: Break
4. $S(Parent(C)) =$ FAILED

Now that we have examined both steps of the algorithm, we can understand how it works from start to end. First, we initialize the algorithm so that the current node points to the root of the HTN. Then, the exploration step is launched from the root. Two cases may happen: the preconditions of an explored node are not satisfied or the search finds a leaf, whose action is executed. In both cases, the node under consideration at the end of the search process gets a new status and this new status must be propagated upwards. When the propagation ends, the exploration step starts downwards again from where the propagation step ended. As explained right above, the propagation step only ends in the second and third case when the conditions for the parent node's

status to change are not met. The algorithm simply goes on running this way, interleaving exploration and propagation, until the root node receives the status SOLVED or FAILED.

2.4 Acting with Speech and Body Gestures: An Interactive James Bond Application

User interactions are integrated into the planning process at two different stages: when inspecting a node's preconditions and when determining an action's outcome. When the search explores a new node, it first inspects its preconditions before investigating its children. In many cases, these preconditions integrate information about the user. For instance, a particular action may require the user to be close to the character to be executable. In a similar fashion, the outcome of an action's execution often depends on the user's behavior. The success or failure of the action is determined by the analysis of the user's response through gestures, speech utterances or facial expressions. The most comprehensive way to analyze how user's interactions are dealt with is through a set of examples of such interactions.

In the remaining of this section, we will thus examine in detail two interactive applications making use of the HTN planning process described above. The first application is an interactive storytelling prototype, which allows the user to play part in the story unfolding through speech and body gestures. The second example is the scenario of a small-scale immersive video game in which the user can control the narrative with facial expressions.

A famous James Bond movie (Goldfinger) inspired the development of our first prototype of mixed reality interactive storytelling. This prototype was developed in collaboration with a team of six researchers from the University of Teesside (United Kingdom), under the supervision of Professor Marc Cavazza.

James Bond stories have narrative properties that make them good candidates for interactive storytelling experiments: for this reason, they have been used as a supporting example in the foundational work of Roland Barthes in contemporary narratology [12]. Besides, their strong

reliance on narrative stereotypes facilitates narrative control and the understanding of the role that the user is allowed to play.

2.4.1 HTN Representation of James Bond's Role

The basic storyline of our application represents the early encounter between Bond and the villain (let us call him the Professor). The objective for Bond is to acquire some essential information and then escape from the Professor's office. The actions of the user (acting as the Professor) are going to interfere with Bond's plan, altering the unfolding of the scene. Figure 2.6 depicts the HTN that represents James Bond's role in the application.

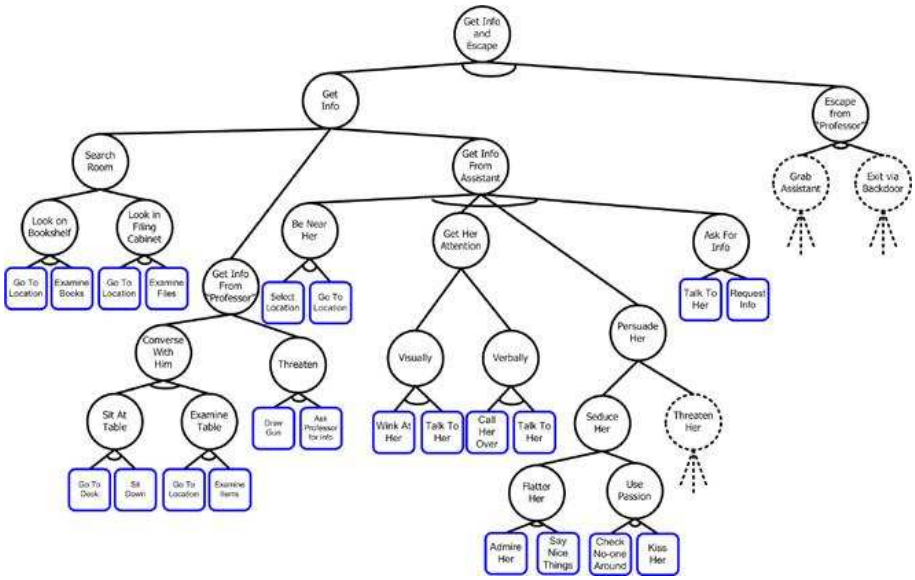


Figure 2.6: Hierarchical Task Network for James Bond

We see in Figure 2.6 that James Bond's main goal is twofold: get information and then escape from Professor. It is therefore natural to use an AND-decomposition to separate both subgoals, whose execution must be done sequentially. If we focus our attention on the GET INFO subgoal, we notice that James Bond may get the desired information

using three different approaches: by searching the room, getting the information from the Professor or from the Professor's assistant. In this case, it is the immediate surrounding of James Bond at the time the GET INFO node is explored that will serve to compute the value of the cost function associated with each of the node's child. If either the Professor or his assistant is close to James Bond in the room, he will try to get the information by discussing with the closest person whereas he will choose to search the room if neither the Professor or the assistant are in his immediate surrounding. The same kind of heuristic function is used for the GET HER ATTENTION node, when James Bond tries to get the information from the Professor's assistant: verbal modality will be used if no direct eye contact is possible (if the two characters are not facing each other) while visual modality will be chosen if both characters are facing each other. These examples demonstrate that any kind of information can be inserted in the cost functions associated with OR-nodes. In particular, it is easy to introduce randomness by inserting a stochastic component in the heuristic function, so that successive uses of the application can generate different storylines, even if user interventions remain the same.

2.4.2 Real Time Planning of James Bond's Behavior

To illustrate our discussion, let us consider a fragment of James Bond's HTN and analyze how user interactions can modify in real-time the course of the story. Figure 2.7 depicts how James Bond changes his behavior when the Professor invites him at his office. When the application is launched, James Bond follows his plan and starts searching the room for the information, which is an action that can be performed only if unnoticed by the Professor or his assistant. When the user welcomes James Bond using both a welcoming gesture and a verbal utterance such as *"Welcome Mr Bond!"*, Bond becomes aware of the professor's presence and has to abandon the intention to search the files, as the node's precondition NOT NOTICED is violated. The node SEARCH FILES receives the FAILED status, which has to be propagated upwards in the tree. As the SEARCH FILES's parent node is an AND-node, the algorithm assigns the FAILED status to the parent node and goes on its upward motion in the tree. Coming up to the SEARCH ROOM node, the propagation process

stops as it comes to an OR-node that still has a child with an UNKNOWN status. The exploration step starts again from that node by examining SEARCH ROOM's preconditions. As the NOT NOTICED precondition is again violated, the node is marked as FAILED and the propagation resumes upwards to the GET INFO node. As the GET INFO node still has two unexplored children, the heuristic function is computed for the two remaining alternatives. Because the professor is close to James Bond, the solution GET INFO FROM PROFESSOR is selected and James Bond turns around, heads toward the Professor's office and sits down on the chair.

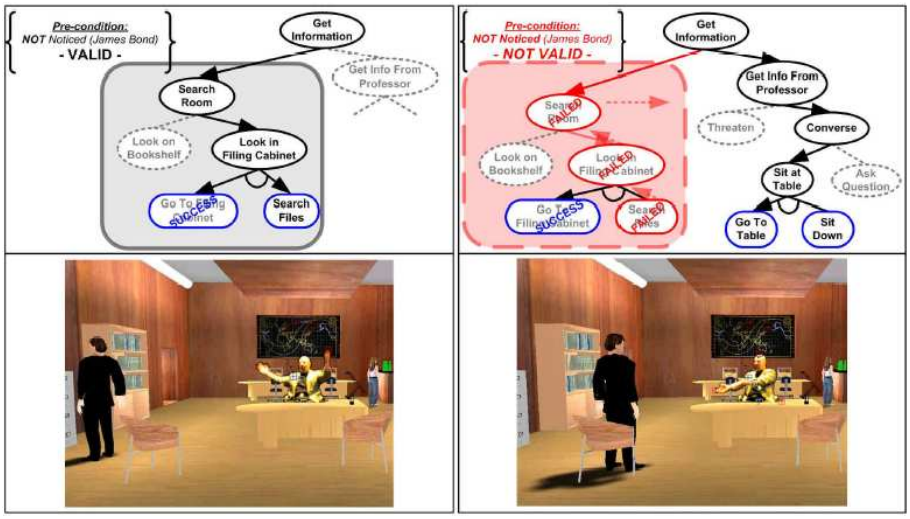


Figure 2.7: An example of user intervention. The user's greetings force James Bond to change his behavior

The fact that the user visually takes part in the story presentation obviously affects the modes of user intervention: these will have to take the form of traditional interaction between characters. As a matter of fact, the actions by which the user may interfere with the story should have a visual presentation that blends into it. In other words, the mechanisms of his normal acting should serve as a natural basis for his intervention in the storyline. These mechanisms comprise physical interactions (the direct contact between the user and virtual world enti-

ties), symbolic gestures, and speech (the latter two combining through various forms of multimodal interaction).

2.4.3 Physical Interactions

Physical interactions consist of all forms of contact between the user (or more precisely the user's embodiment through his video avatar) and virtual actors. This interaction is mediated by the low-level mechanisms managing interaction between actors in the Unreal Tournament™ engine. The user's position is associated with an empty bounding box that can generate various kinds of interaction information with the environment's objects and the virtual actors. Physical interaction is based on the use of a similar coordinate system for both the virtual world and the video information captured from the real setting. As a consequence, the user's bounding box follows the user's movements in the real world. For instance, when the user moves toward a virtual actor (which results in his bounding box colliding with that of the agent), several forms of interactions become active (such as attracting attention, shaking the agent's hand, slapping or punching the agent).

The exact form of physical interaction is triggered by the simultaneous recognition of a user's gesture and a collision between bounding boxes. This will in turn determine the action performed on a virtual actor and affect the storyline accordingly. For instance, depending on the gesture recognized (a handshake or a slap), the corresponding agent reaction is displayed (the appropriate animation being played) and the consequence of this reaction is used to update the agent's plan. The same mechanism can be used to interact with the virtual world objects, although this type of interaction has not been implemented in the prototype presented in this work.

One essential aspect of the interactions is that the system is actually tracking symbolic gestures corresponding to narrative functions, such as greetings, threatening (or responding to a threat, such as putting his hands up), offering, calling, dismissing, etc. However, gestures are mostly used in conjunction with speech recognition. The accurate recognition of a user intervention is based on i) the current stage of the plot and ii) a multimodal interpretation. For instance, the fact that the user stands up from a seated position could be interpreted as a greeting at the

very beginning of the scene, while it would be interpreted as terminating the interview should it take place at a later stage.

2.4.4 Speech Interactions

The speech recognition component is based on the Ear SDK from BabelTech™ , which is an off-the-shelf system including a development environment. The speech recognition is used in a multi-keyword spotting mode. Multi-keyword spotting consists in dynamically recognizing keywords from a predefined set from any user utterance, regardless of the other contents of that utterance. One advantage is that it can provide a robust recognition of the most relevant topics in context, without imposing constraints on the user (like the use of a specific phraseology). In other words, it means that the module matches sets of keywords with semantic concepts. By taking into consideration the context of the utterance, recognizing a single keyword is often enough to understand the semantic associated with a complex sentence.

User utterances of the kind that take place in this narrative context can be described as a specific kind of speech acts. The notion of speech act corresponds to an utterance which has a specific impact on the hearer's behavior. More importantly, there exists a very good mapping between speech acts and narrative actions such as greetings, threats, requests, denial, etc., to the point that they can constitute a direct input into the narrative representation as far as interactive storytelling is concerned.

The natural language interpretation [13] [14] is based on a template matching procedure, as the use of multi-keyword spotting precludes more complete forms of syntax-based parsing. Template-filling is based on the recognition of certain action verbs (or substantives). Once a specific template can be activated from that occurrence, it searches for keywords that can fill its remaining slots, corresponding to the action parameters (e.g. subject or object, by looking for pronouns or proper names).

To illustrate the mechanisms that are involved, let us consider a practical example extracted from our James Bond interactive storytelling prototype. The user, who plays the role of the Professor, should be

allowed to invite James Bond to sit down on a chair facing the office. Knowing that James Bond and the Professor use very formal terms to communicate, we can assume that the Professor would invite James Bond to sit down in a very polite way. This enables the use of an *invitation* template that is associated with a set of keywords such as "'*Why don't...?*"', "'*Would you...?* or "'*Please,....* Once such a template has been detected, the system searches for additional information that corresponds to the template's *attributes*. When inviting James Bond to sit down for instance, once the template *invitation* is recognized, the system will search for a verb that describes the object of the invitation and a name to determine who the invitation is dedicated to.

Although multi-keyword spotting can effectively lead to satisfying results, it may fail to detect a template or some of its attributes. To increase the robustness of the speech recognition component, it is necessary to include information coming from other modalities. The remaining of this section discusses the combination of the gesture and speech modality, as a way to improve the recognition of user's intentions.

2.4.5 Fusion of Speech and Gesture Modalities

The mechanisms governing the gesture recognition algorithm are thoroughly described in the third chapter of the present thesis. We will therefore not explain it in this section but rather present some examples of semiotic gestures that can be detected by our system (Figure 2.8) and show how these gestures may complement the information provided by the speech recognition component, thereby improving the interpretation of user interactions.

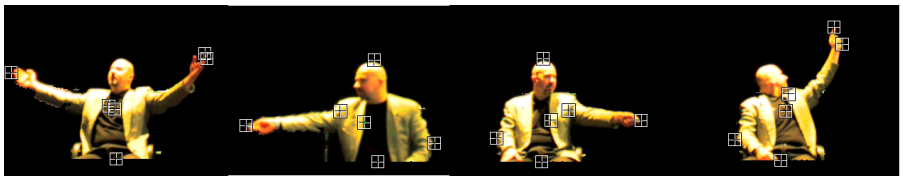


Figure 2.8: Examples of body gestures that can be detected by the gesture recognition module, described in chapter 3

In most cases, a gesture may have several possible interpretations, depending on the context in which it is produced. Figure 2.9 illustrates this assumption. On this figure, we have captured three gestures that look very similar, but were produced at three different stages of the story. Even though these three gestures look alike, their meanings are totally different from each other. The leftmost gesture corresponds to the user refusing to give the requested information to James Bond. It is a denial gesture that could correspond to the speech act "*You must be joking mister Bond!*". The second gesture is used by the user to welcome James Bond and could be linked to an utterance of the type "*Welcome, Mr. Bond!*". The third gesture is produced by the user when James Bond points his gun toward the user, threatening him to obtain the information. The user in this case responded to the threat with a sentence such as "*Shoot me and you will never have the information!*". It is obvious that these three gestures need contextual information in order for the system to understand their meaning.

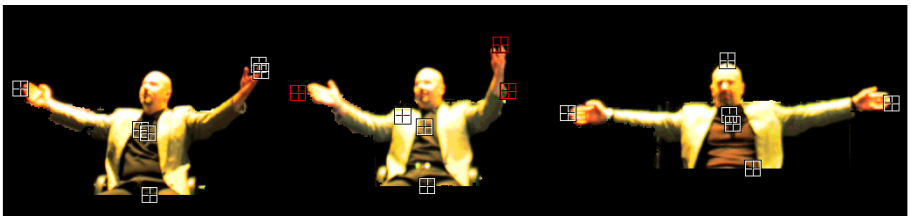


Figure 2.9: Very similar gestures may have very different meanings

To cope with these gestures having different possible meanings, we have assigned a set of interpretations to each of the gestures. The keywords that are detected in the speech act are combined with the information given by the HTN planner. The fusion of these two sources of information is then used to choose among the different possible interpretations the one that fits to the context of the speech act.

In a similar fashion, speech acts may sometimes be difficult to interpret and gestures may serve as a disambiguation factor. For instance, once the user has welcomed Bond, he can invite him to sit on a chair to start a discussion, pronouncing a sentence of the type "*Please, take a seat, I have something to tell you!*". In this case, it is possible that the

speech recognition module fails to understand the invitation. However, if the user is also pointing at the chair, it could be enough for the system to understand the whole user's intention. The gesture recognition module recognizes the pointing gesture and the game engine associates the pointed object with a set of functions that are linked to this object. As the pointed object (the chair) is only associated with a single function (SIT ON), the speech recognition component will parse again the user's utterance to detect an invitation template or the presence of a keyword associated with the action SIT ON. Once the keyword *"Please,..."* or *"seat"* is detected, the virtual character will move toward the chair designated by the user and sit down.

2.5 Affect-Driven Interactive Storytelling

In affect-driven storytelling, we allow the behavior of the virtual characters to be a consequence of **their** emotional state. This is realized by defining *affective heuristics* that are used in the exploration phase of the HTN planning algorithm when encountering OR-nodes. To illustrate with a practical example, let us define the emotional state of a virtual character by an emotional vector E:

$$E = [happy, angry, afraid] \tag{2.3}$$

In the scope of this example, we will consider three different emotions: happiness, anger and fear. Let us now assume that each emotion can receive a value ranging from -1 to +1. For each emotion, a value of +1 means that the emotion is very intense (the character is very happy, very angry or very afraid), while a value of -1 means the opposite (the character is very sad, very calm or very confident). Any value comprised between -1 and +1 may be assigned to each of the elements of E[4].

To show how affective heuristics may be used to control the behavior of the virtual characters, let us consider a fragment of the original Bond's HTN, presented in section 2.4.1. Being noticed by the Professor, James Bond chooses to get the information directly from the Professor.

[4]Some combinations should however be avoided. For instance, it is unlikely for a character to be both very happy and very angry!

The node GET INFO FROM PROFESSOR is a OR-node with two children: CONVERSE WITH HIM and THREATEN HIM. This situation is illustrated in Figure 2.10.

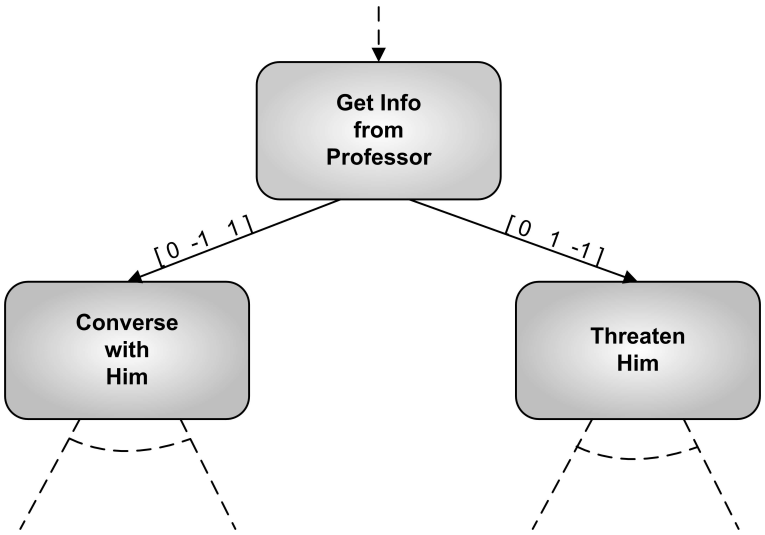


Figure 2.10: Using an internal emotional state vector to decide which node to be explored next

James Bond thus has two alternatives to obtain the information from the Professor, and the algorithm has to compute a heuristic function for each node to decide which one should be explored first. James Bond will rather choose to threaten the Professor if he is angry and not afraid, while he will probably choose to converse if he feels calm and afraid. To model this, we can assign to each alternative different *ideal* emotional vectors:

$$\text{CONVERSE}: \quad E_{Converse} = [happy = 0, angry = -1, afraid = 1] \quad (2.4)$$

$$\text{THREATEN}: \quad E_{Threaten} = [happy = 0, angry = 1, afraid = -1] \quad (2.5)$$

Let us assume that James Bond's emotional vector at that stage is:

$$E_{Bond} = [happy = 0.32, angry = -0.68, afraid = 0.22] \quad (2.6)$$

We can define a heuristic function that represents the distance between James Bond's emotional vector and the ideal emotional vectors corresponding to each alternative, using the *SSD (Sum of Squared Differences)* metric:

$$h_{child} = \sum_{i=1,...,3} (E_{child}[i] - E_{Bond}[i])^2 \tag{2.7}$$

If we compute the value of the heuristic function for each alternative, we obtain the following values:

$$h_{converse} = (0 - 0.32)^2 + (-1 + 0.68)^2 + (1 - 0.22)^2 = 0.81 \tag{2.8}$$
$$h_{threaten} = (0 - 0.32)^2 + (1 + 0.68)^2 + (-1 - 0.22)^2 = 4.41 \tag{2.9}$$

The HTN planner will inspect both values and select the child whose heuristic has the lowest value. In this case, as James Bond is rather calm and a bit afraid, he will choose to try to get the information by conversing with the Professor.

In the example depicted above, we have assigned a arbitrary value to James Bond's emotional state. In practice, the HTN should be designed in such a way that to each event is associated a modification of the character's emotional vector. For instance, if the Professor offers James Bond a drink, James Bond should become less afraid[5], while if the Professor draws a gun, James Bond should become much more afraid.

2.6 Conclusion

In this chapter, we presented a prototype of Mixed Reality Interactive Storytelling (MRIS).

First, we described how a mixed reality environment could be created, using a body segmentation engine to extract the user's silhouette from his background, a game engine to generate the virtual stage and a DirectX™ component to mix in real-time the two image layers.

[5]James Bond would drink a Martini, shaken not stirred!...and alcohol removes inhibition...

We then justified the use of a hierarchical representation of the character's behavior by arguing that hierarchical decomposition was the best way to deal with the complexity issues inherent to large-scale applications.

From the HTN representation, we derived an online planning algorithm and illustrated how it interleaves planning and execution, with a prototype based on a James Bond's episode.

We then investigated the interactions between the user and its application, starting from physical and speech interactions alone, before examining how considering both modalities together could improve the recognition of user's intentions.

Finally, we introduced an affect-driven story generation system in which the virtual characters select their behavior according to their own internal emotional state.

Chapter 3

Gesture Recognition

3.1 Introduction

In the previous chapter, we described a mixed reality interactive storytelling application, in which the user could interact with the virtual characters using a set of semiotic gestures. In this chapter, we present the gesture recognition engine itself.

An abundant literature relates research achievements in the field of human body gesture recognition and understanding. The recent development of markerless human body motion capture systems [15] [16] dramatically increased the interest for gesture-controlled interactive applications [17] [18]. The last decade has seen the emergence of mixed reality systems, in which the user can communicate with virtual entities through body gestures [19].

In the scope of this work, our goal was to build a robust system that could detect a few gestures in real-time. These gestures are then associated to a set of actions that are likely to be performed given the context of the plot. For more information about the interpretation of the gestures that will be discussed in this chapter, the reader is invited to refer to the description of the prototype presented in the preceding chapter (the James Bond interactive storytelling application, described in section 2.4) or to a similar system [19].

As gesture recognition is not the main topic of this thesis[1], we will only present a simple and straightforward solution. Our solution is simple and efficient, but it does not truly constitute an advance in the field of gesture recognition, as there exist much better techniques to treat temporal sequences of coordinates. In particular, dynamic probabilistic models such as Dynamic Bayesian Networks (DBN) constitute an elegant and efficient solution for recognizing events out of temporal sequences of observation data, especially if the duration of the events is either unknown or variable.

Built on top of a body segmentation algorithm, our gesture recognition system analyzes the temporal evolution of the coordinates of five so-called *crucial points*, situated on the user's extracted silhouette. These crucial points corresponds to the highest, the lowest, the leftmost, the rightmost and the center of gravity of the user's silhouette, as depicted on Figure 3.1.

The segmentation engine that extracts in real-time the silhouette of the user from its background is the Salto™ engine, developed by Alterface SA. We outlined the basic mechanisms of this segmentation engine in section 2.2. Therefore, we will not discuss it further more here. The reader who would like to have a deeper understanding of this segmentation algorithm is invited to refer to [20] and [21].

The gesture recognition algorithm processes sequences of coordinates, received in real-time from the segmentation engine. For each new frame, a dedicated XML interface parses the output of the segmentation engine and records the value of the crucial points' coordinates corresponding to the new frame. The set of recorded values is then used to update a queue containing the coordinates extracted from the last N frames. The information stored in the queue is then passed through a set of classifiers, each of them being responsible for the detection of a specific gesture. To each individual classifier is associated a set of rules, which have to be fulfilled for a certain percentage of the analysis time in order to activate the corresponding detection signal.

The two first gestures that we will consider are actually postures:

[1] As announced in the introduction, the focus is rather set on facial expression recognition

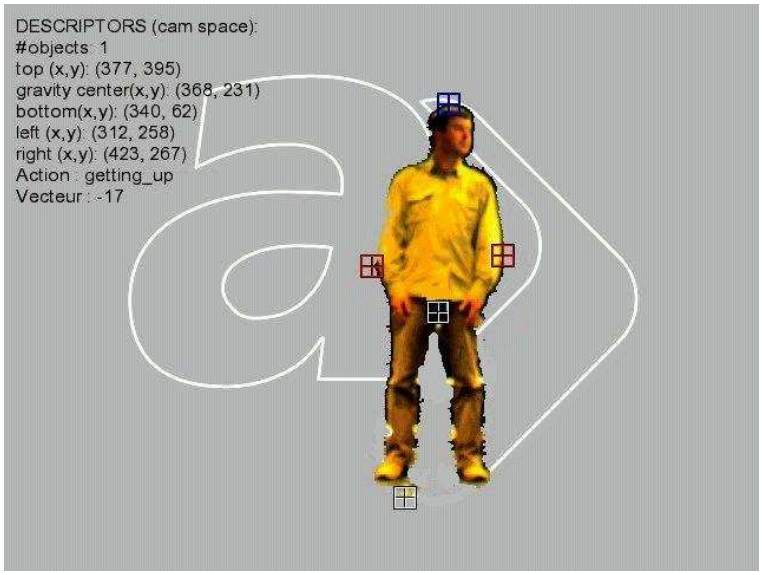


Figure 3.1: Five crucial points are defined on the user's extracted silhouette

their detection is based on the recognition of static configurations. The detection of the three last considered gestures is based on the evolution of coordinates over time. We should normally use the term 'gesture' only in the latter case, that is: when a movement is involved. For the sake of simplicity, we will however refer to gestures for all cases that will be considered.

At the time the gesture recognition system was designed, the segmentation engine was able to process up to six frames per second, on a Pentium 4 laptop computer equipped with a 2.4 GHz processor. We empirically adjusted the expected duration of the gestures to be recognized from 500 ms to one second, depending on the gesture, which means that each classifier processes a maximum of $N = 6$ frames, each of them containing the 2D-coordinates of $M = 5$ different crucial points.

As introduced earlier, each classifier applies a set of rules to the coordinates of the last N frames and outputs a string containing the name of the detected gesture if *all the rules associated with the gesture*

are satisfied simultaneously for a sufficient portion of the analysis time. This string can then be sent to an external module via socket-based communication.

In the remaining of this chapter, we will describe the rules that correspond to each gesture that the system is able to recognize. For the sake of clarity, we will adopt the following notations: the horizontal (vertical) distance between two points A and B belonging to the silhouette will be noted $d(A_x, B_x)$ ($d(A_y, B_y)$). Besides, each of the crucial point will be abbreviated as follows:

1. HG : Highest point of the silhouette
2. LW : Lowest point of the silhouette
3. LM : Leftmost point of the silhouette
4. RM : Rightmost point of the silhouette
5. CG : Center of gravity of the silhouette

3.2 Both Arms Open

The first gesture that our system is able to recognize is actually more a posture than a gesture. It is called BOTH ARMS OPEN and is depicted in Figure 3.2. It may have very different significations depending on the context in which it is produced, as illustrated in section 2.4.1.

This gesture corresponds to the user having both arms open, which means: the distance between both hands should be large (rule 1) and both hands should be around the same horizontal axis (rule 2). The gesture will be detected only if *both rules are simultaneously satisfied for a sufficient number of frames.* Mathematically speaking, the two rules could be expressed by the following relations:

$$\text{RULE 1:} \quad d(LM_x, RM_x) > T_1 \tag{3.1}$$

$$\text{RULE 2:} \quad d(LM_y, RM_y) < T_2 \tag{3.2}$$

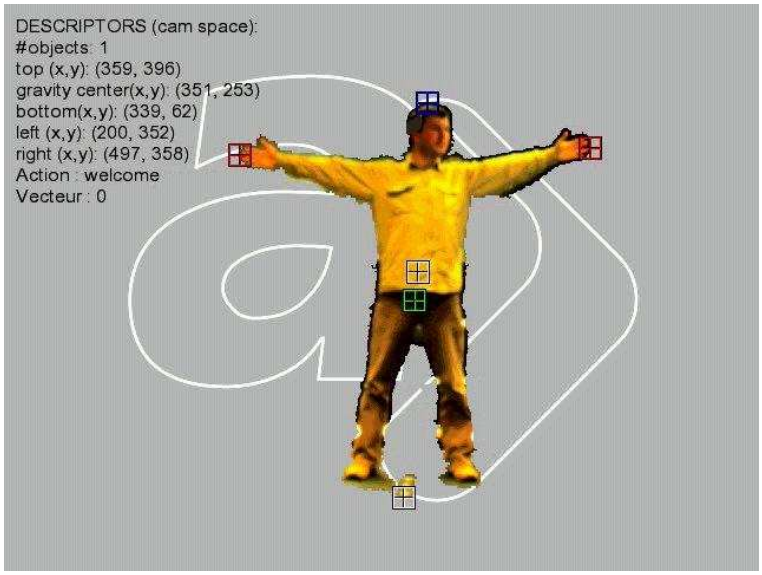


Figure 3.2: The BOTH ARMS OPEN gesture

The value of the thresholds T_1 and T_2 were determined by inspecting movies recorded from testing sessions and were adjusted to maximize the performances of the system. It should be noted that the results presented in this section concern video sequences that have *not* been used for the determination of the thresholds.

In the current prototype, the user acts at a fixed distance of the camera (a mark is placed on the ground, four meters away from the video camera). Therefore, the thresholds were empirically set as a fixed number of pixels (the resolution of the camera being fixed to 640x480). It would however be quite easy to adapt the thresholds to individual user's morphology by integrating a short calibration phase at the beginning of each new session.

Let us illustrate the detection process with a practical example. Figure 3.3 shows the evolution of the variable X for 195 successive frames, X being defined as:

$$X = d(LM_x, RM_x) - T_1 \tag{3.3}$$

Rule 1 is satisfied whenever X has a positive value. By inspecting Figure 3.3, we notice that this happens twice in the sequence: around frame 30 and around frame 95.

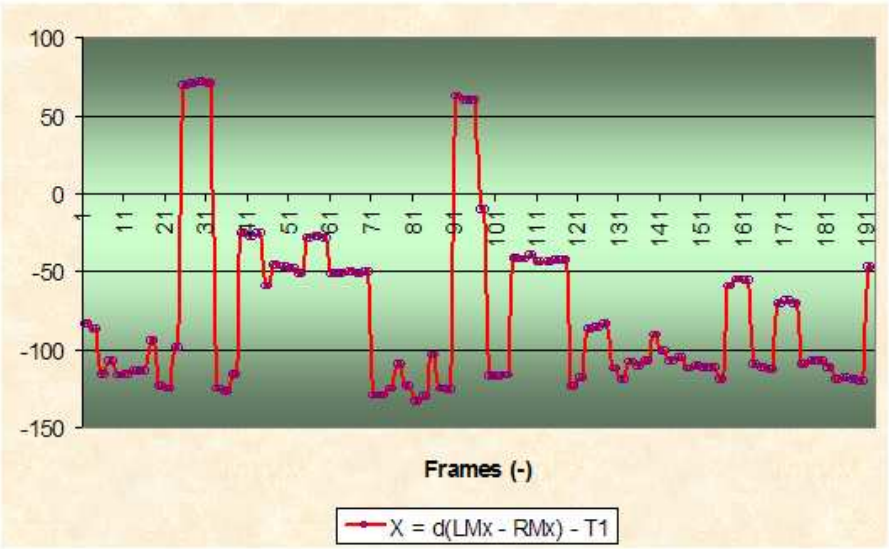


Figure 3.3: Positive values of X indicate that rule 1 is satisfied

Similarly, we can define a variable Y :

$$Y = d(LM_y, RM_y) - T_2 \tag{3.4}$$

This time, rule 2 will be satisfied for all frames for which Y has a negative value, which indicates that the vertical distance between leftmost and rightmost points is below the threshold. Figure 3.4 shows the evolution of Y for the 195 frames of our example.

As mentioned above, the gesture BOTH ARMS OPEN will be detected only if both rules are satisfied for a sufficient part of the analysis time. Two counter variables (BUFFER1 and BUFFER2) are used to count the number of frames for which each condition is fulfilled: it is incremented whenever a rule is satisfied and decremented otherwise, both buffers having a minimum value of 0 and a maximum value of 5. Once both

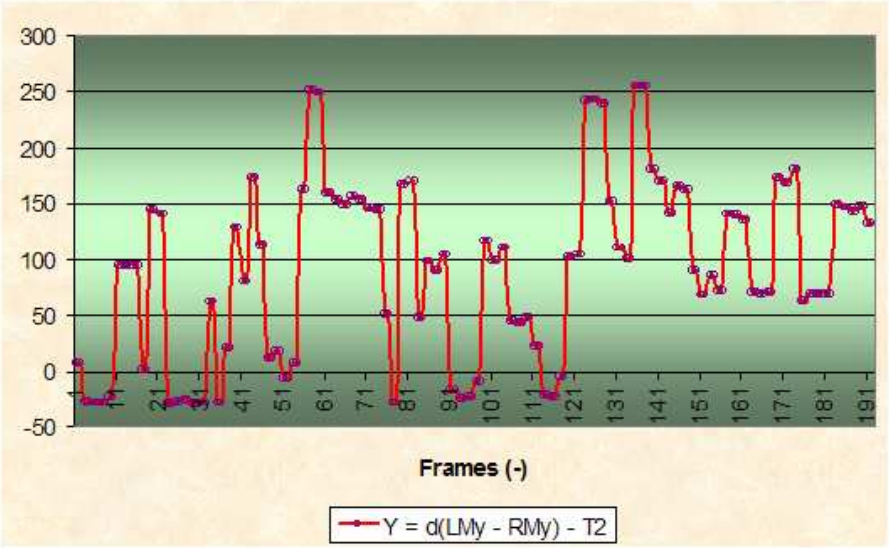


Figure 3.4: Negative values of Y indicate that rule 2 is satisfied

counters have a value that is greater than 2, the detection signal is activated. The use of such counter variables makes the recognition system more robust to segmentation errors as false alarms have to last for several successive frames to generate a false detection signal. Figure 3.5 shows the evolution of both counter variables in the example under consideration. On this figure, yellow zones represent the frames for which the detection signal has been activated.

3.3 Pointing Gestures

A POINTING gesture (as depicted in Figure 3.6) is recognized when the following configuration is encountered: one hand is pointing toward something, while the other hand remains relatively close to the center of the silhouette. The pointing gesture is thus represented by a large horizontal distance from the pointing hand to the center of gravity and a relatively small horizontal distance from the other hand to the center of gravity. We therefore have two rules to model this gesture. If we aim

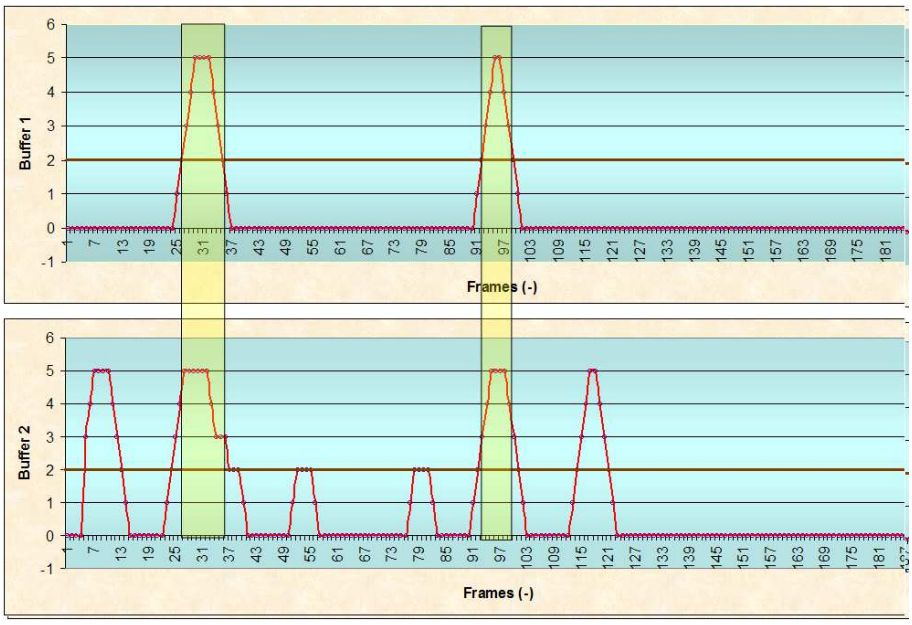


Figure 3.5: If both counter variables have a value greater than 2, a detection signal is activated, as it is the case in the yellow colored zones.

at detecting the user pointing with the left hand, the two rules are given by:

$$\text{RULE 1:} \quad d(LM_x, CG_x) > T_1 \tag{3.5}$$
$$\text{RULE 2:} \quad d(RM_x, CG_x) < T_2 \tag{3.6}$$

Inversely, if we would like to detect the user pointing with his right hand, the two rules are:

$$\text{RULE 1:} \quad d(RM_x, CG_x) > T_1 \tag{3.7}$$
$$\text{RULE 2:} \quad d(LM_x, CG_x) < T_2 \tag{3.8}$$

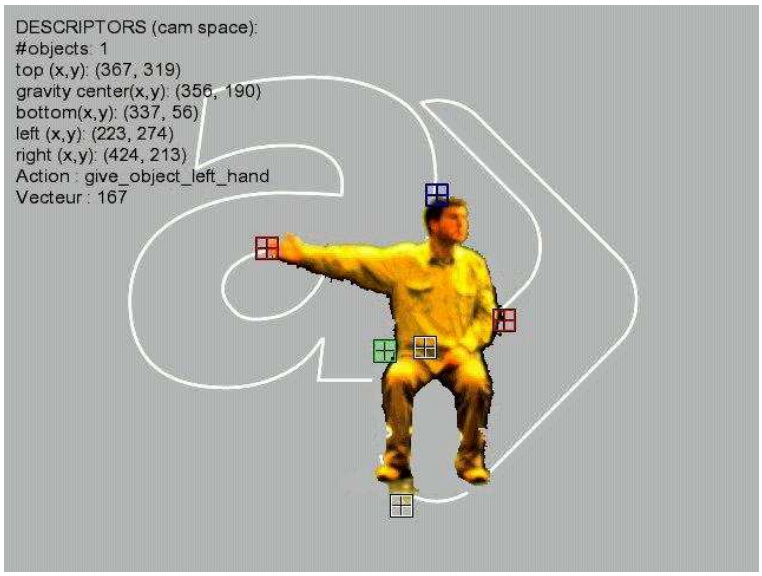


Figure 3.6: The POINTING gesture

Like in the case of the BOTH ARMS OPEN gesture, the detection signal will only be activated if both rules are satisfied for a significant part of the analysis time, in order to avoid false alarms due to sporadic errors in the segmentation process. We thus proceed in a similar way than in

the case of the BOTH ARMS OPEN gesture, using two counter variables. When both counters have a value greater than 2, the detection signal is triggered. The reader who would like to have a more detailed description of the `pointing` gesture's detection process is invited to consult the example of the BOTH ARMS OPEN gesture depicted in the last section. The procedure is identical for both gestures except for the rules that are being used, which are of course intrinsic to the gesture to be detected.

Once a POINTING gesture has been detected, the system computes a 2D-vector which represents the direction pointed by the user. To compute this vector, we build an estimate of the vertical coordinate of the shoulders, by subtracting from the highest point of the silhouette HG a fraction of the distance that separates that point from the center of gravity CG. A virtual pointing arm is built by linking the shoulder's estimate with the pointing hand. The direction of pointing is then expressed as the angle formed by the pointing arm with respect to the horizontal line, counted in the counter-clockwise direction with the 0°-angle corresponding to pointing to the right of the screen (to the left from the user's perspective). In the example depicted in Figure 3.6, the computed angle is 167°. Although the method is based on a coarse approximation of the shoulders' position, using this technique enabled us to come up with a quite reliable estimation of the direction pointed by the user.

3.4 Raise Hand, Get Up and Sit Down

In this section, we analyze the detection of three distinct gestures: GET UP, SIT DOWN and RAISE HAND. These three gestures are presented together because their detection involves the analysis of the spatio-temporal evolution of the highest point HG. The two first gestures are the opposite of each other: they correspond to the transition from the state *seated* to the state *standing up* and vice-versa, as illustrated on Figure 3.7.

The third gesture that we will discuss in this section is the RAISE HAND gesture, which can be executed both while seated or while standing up. Figure 3.8 depicts this gesture in both configurations.

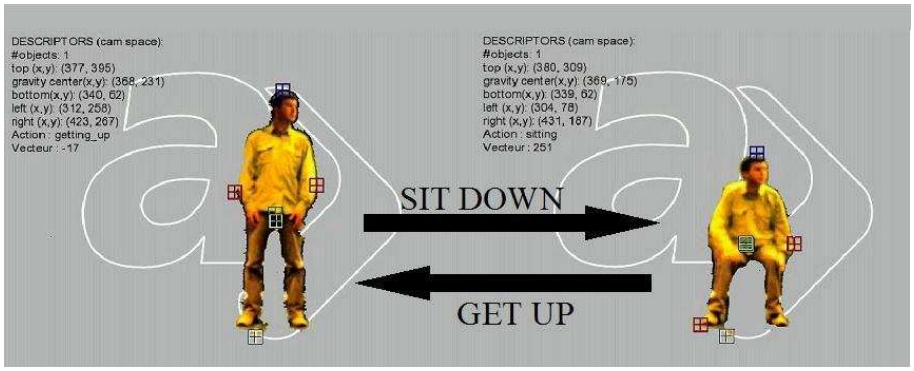


Figure 3.7: The SIT DOWN and GET UP gestures

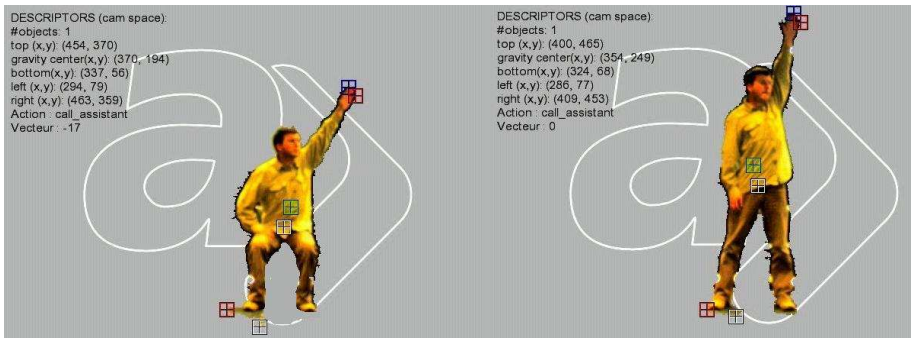


Figure 3.8: The RAISING HAND gesture may be executed both while seated or while standing up

As the gestures to be detected have an average duration smaller than one second, we examine the evolution of HG for every group of four successive frames (this corresponds to action duration of approximatively 0.66 seconds). The evolution of the coordinates of the highest point HG will be decomposed in their horizontal and vertical components:

$$\Delta HG_x(i) = HG_x(i) - HG_x(i-4) \tag{3.9}$$

$$\Delta HG_y(i) = HG_y(i) - HG_y(i-4) \tag{3.10}$$

For both the RAISE HAND and the GET UP gestures, we request the evolution of the vertical coordinate to be positive and greater than a threshold T_y, while for the SIT DOWN gesture, we require exactly the opposite: the evolution of the vertical coordinate should be negative with a value smaller than $-T_y$.

The problem with the considered criterion comes from the fact that whenever getting up **or** raising hand is involved, the vertical coordinate of both the center of gravity and the highest point increases. Inversely, it always decreases whenever the user is sitting down or lowering his hand after having raised it. To cope with this ambiguity, we have introduced two state variables: SEATED and CALLING. The SEATED variable will be set to TRUE when the gesture SIT DOWN has been detected and will remain true as long as the gesture GET UP is not detected. Similarly, the variable CALLING will be set to TRUE whenever the gesture RAISE HAND is detected and will remain true as long as the hand stays up. Adding these two state variables makes our problem easier: an increase of the vertical coordinate of HG will be automatically associated with the RAISE HAND gesture if the user is standing, while a decrease will be associated with the lowering of the raised hand if the variable CALLING is set to TRUE. However, an ambiguity remains in the case where the HG's vertical coordinate increases while the user is seated: we must have a criterion to distinguish between the user raising his hand or getting up. To treat that particular case, we observe the evolution of the horizontal coordinate of the highest point HG: if the user is raising the hand, the highest point that was located on the user's head will switch its anchor to the user's raised hand, which results in an horizontal displacement of HG. In the case of the user getting up, the highest point will remain an-

chored to the user's head and thus no significant horizontal displacement will occur.

Taking all the information into account, we can summarize the detection process by associating to each gesture the following sets of rules:

1. The GET UP gesture:

$$\text{SEATED} == \text{TRUE} \tag{3.11}$$
$$\Delta HG_y > T_y \tag{3.12}$$
$$\texttt{abs}(\Delta HG_x) < T_x \tag{3.13}$$

2. The SIT DOWN gesture

$$\text{CALLING} == \text{FALSE} \tag{3.14}$$
$$\Delta HG_y < -T_y \tag{3.15}$$

3. The RAISE HAND gesture

$$\Delta HG_y > T_y \tag{3.16}$$
$$\texttt{abs}(\Delta HG_x) > T_x \tag{3.17}$$

3.5 Depth Information

In the last chapter, we explained how a mixed reality environment could be created by inserting the extracted user's silhouette in a virtual stage. However, as the silhouette is expressed in 2D-coordinates whereas the virtual world is a 3D-environment, we need a way to estimate at which depth the user should be inserted in the virtual stage. A solution can be obtained by considering the distance between the user and the camera. As a matter of fact, it is possible to assign a 3D-bounding box to the user's silhouette, by mapping the distance between the user and the video camera to a depth in the virtual stage. As described in details in the preceding chapter, this 3D-bounding box can then be used by the graphic engine to detect events such as collisions, occlusions and interactions with virtual characters.

To estimate the distance between the user and the video camera, we calibrated the vertical coordinate of the lowest point of the user, LW_y, with the distance to be estimated. In other words, we considered a set of 13 distances ranging from 2.8m to 5.2m, using a fixed interval of twenty centimeters between successive measures. The closer the user would be from the camera, the lower the value of LW_y would be. A similar methodology could have been applied to the surface of the silhouette, expressed in pixels, for each of the considered distances. In this case, the closer the user would be from the camera, the larger the number of pixels belonging to the silhouette would become.

The result of the calibration process is depicted in Figure 3.9. Linear interpolation has been used to provide values for distances from the camera that were not considered in the calibration process.

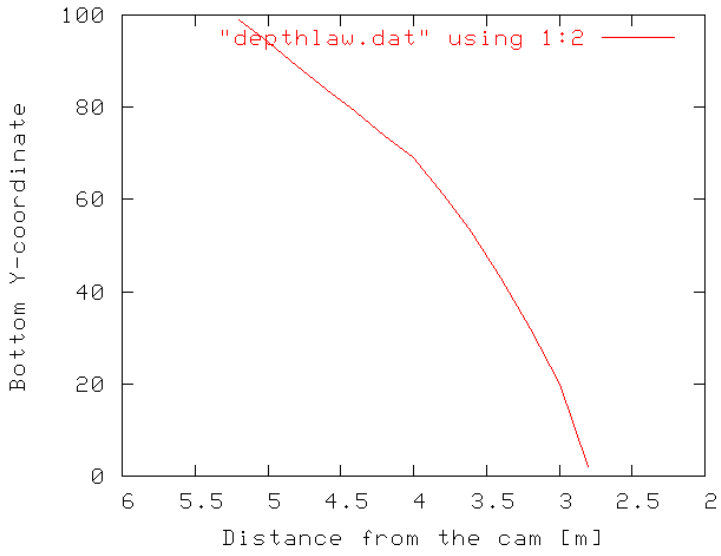


Figure 3.9: The evolution of the lowest point of the silhouette's vertical coordinate with respect to the distance from the camera

3.6 Conclusion

In this chapter, we described a simple and efficient solution to the problem of recognizing a set of five different gestures from sequences of coordinates, computed in real-time by a body segmentation engine. The implemented prototype uses rules based on distances between crucial points to detect postures, such as BOTH ARMS OPEN and POINTING gestures and rules based on the spatio-temporal evolution of these crucial points to detect gestures such as GET UP, SIT DOWN and RAISE HANDS. The solution has been designed to be robust, to work in real-time and to be able to communicate with external modules, via TCP/IP communication. It is therefore a very good solution with regards to the requirements, but we do not pretend that it constitutes the best approach in the general case of recognizing a set of gestures from sequences of body points' coordinates. As mentioned earlier in this chapter, dynamic probabilistic models, such as Dynamic Bayesian Networks, would be recommended in more elaborated systems.

Chapter 4

Affective Information Extraction

4.1 Introduction

Facial expression recognition from video sequences can be seen as a two-step process. The first step concerns the extraction of data that convey affective information, for each of the successive frames of a video sequence. The second step consists in processing these data to detect the presence or absence of specific facial expressions. As the two subproblems are quite complex and *often* considered as two separate problems, they will be treated in two distinct chapters: this chapter will discuss the extraction of affective information from video sequences whereas the next chapter will focus on the recognition of facial expressions, based on the extracted information.

The first objective to be reached is to detect and locate the face in the first frame of the video sequence. Whereas this task is fairly easy for a human being, it becomes much more difficult when the detection has to be performed automatically by a machine. In fact, it requires to think about the criteria that human beings use to recognize and locate faces in an image, and to transpose these criteria in terms of shape, texture and color information.

Once the face has been located, we must formulate its appearance in terms of facial features that carry affective information. To achieve

that, we based our approach on the results of psychology studies [22] [23] [24] [25] that have observed how emotions are expressed on human faces. These studies showed that emotions induce facial muscle actions (such as contractions and dilatations), which result in deformations of the shape of several facial features. Among them, the mouth, the eyes and the eyebrows carry a great part of the affective information that can be observed on the face. To illustrate this assumption, we invite the reader to have a look at Figure 4.1 that demonstrates that an emotion can indeed be detected by inspecting the shape of the facial features we just mentioned [26]. The second part of this chapter will therefore be devoted to the detection of these facial features and to their tracking over time.

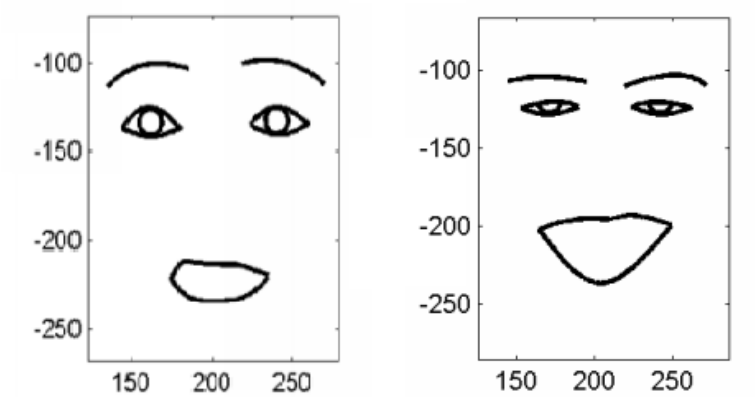


Figure 4.1: An emotion can be detected by inspecting only the contour of three facial features: the mouth, eyes and eyebrows

4.2 Automatic Face Detection

4.2.1 State of the Art

As it would be quite ambitious to present an overview of what has already been done in the field of face detection, we will only give the reader a short insight on some of the techniques that are considered to be among the most successful approaches so far.

Three main paradigms can be considered. Holistic, feature-based and model-based approaches. Holistic approaches aim to detect face regions in general images, using the appearance of the face, as learned by a machine learning algorithm. The most successful of such approaches use Support Vector Machines [27] or Neural Networks [28]. Feature-based approaches [29] recognize faces by inspecting the value of some image attributes (such as color or motion attributes) and detect regions whose attributes match those of a face. Eventually, model-based approaches use 2D or 3D face models and try to fit these models onto potential face regions.

In this thesis, we tried a hybrid approach to the problem of automatic face detection. The idea is to combine color, edge and shape information to extract an estimation of the contour of the face. We chose this approach because, in practice, the most performant face detectors are those based on a combination of different weak detectors. And, as face detection is the first step of our facial expression recognition system, it is crucial to design a robust face detection process.

After further analysis of the extracted face image, we provide a technique for automatically initializing a model-based tracking algorithm. In this section, we describe only the first of these steps: the extraction of the face contour in the first frame of the video sequence. Subsequent steps are treated in the following sections.

4.2.2 Skin-color detectors

The first idea that comes in mind when thinking about automatic face detection is that a face can be set apart from its background by using color information. The implementation of this idea in an automatic segmentation system leads to pixel-based skin-color segmentation, in which the image is divided into face and non-face regions, based only on the color of the pixel.

The color of the human skin is indeed distinctive from the color of many other natural objects: skin colors are distributed over a small area in the chrominance plane [30]. The first step of skin-color based detectors is therefore to convert the image from traditional RGB (red-green-blue) representation to a color space in which luminance and chrominance

channels are separated, such as the YCbCr or the HSV (hue-saturation-value) color spaces. In the HSV color space, skin color from all ethnicities clusters tightly in a small portion of the hue-saturation (HS)-space [31], as depicted in Figure 4.2. For this reason, we have chosen to represent images in that color space. Our goal will then be to try to separate automatically face and non-face regions, using the two components of the chrominance of a pixel as a discrimination factor.

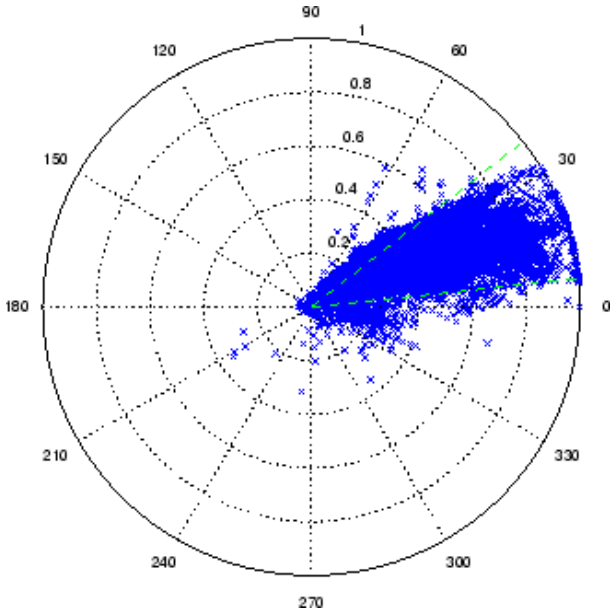


Figure 4.2: Skin pixels plotted in the HS-space. Hue is defined by the angle θ (red = 0) while saturation is represented by ρ

A RGB-image contains three channels: the color of a pixel is expressed as a mixture of red, green and blue colors. If each channel is expressed on a 8-bit intensity scale, we can express more than 16 millions of different colors by combining the relative intensity of the three channels. Let us assume that the relative intensity of each channel is noted by R, G and B and that each of these intensities may vary in the range $[0, \ldots, 1]$. In the HSV color space, we assign an angle H to the hue, which can range from 0° to 360°. Both remaining components, the

saturation S and the value V can take values in the range $[0, \ldots, 1]$. If we define by MAX the maximum of the (R,G,B) values, and MIN the minimum of these values, the transformation from the RGB space to the HSV space can be expressed by the following mapping:

$$
\begin{aligned}
H &= \begin{cases}
\texttt{undefined}, & \texttt{if} \quad MAX = MIN \\
60\frac{G-B}{MAX-MIN} + 0, & \texttt{if} \quad MAX = R \quad \texttt{and} \quad B \leq G \\
60\frac{G-B}{MAX-MIN} + 360, & \texttt{if} \quad MAX = R \quad \texttt{and} \quad B > G \\
60\frac{B-R}{MAX-MIN} + 120, & \texttt{if} \quad MAX = G \\
60\frac{R-G}{MAX-MIN} + 240, & \texttt{if} \quad MAX = B
\end{cases} \\
S &= \begin{cases}
0, & \texttt{if} \quad MAX = 0 \\
1 - \frac{MIN}{MAX}, & \texttt{otherwise}
\end{cases} \\
V &= MAX
\end{aligned}
$$

Although skin colors are contained within a relatively well delimited portion of the HS-space, a face detector based only on hue-saturation values might be inefficient due to the potential presence in the background of skin-look-alike regions. When the background does not contain such regions, detectors based on chrominance provide very satisfying results, even though it must be kept in mind that these types of detectors detect skin regions instead of face regions...

Figure 4.3 presents the detection that can be achieved by only analyzing the hue value of each pixel. In this example, two thresholds θ_{max} and θ_{min} determine the limit between skin and non-skin regions: pixels having a hue value with a θ angle between θ_{min} and θ_{max} will be considered as belonging to the face, whereas pixels with a hue value outside of that range will be classified as background pixels. In Figure 4.3, the binary image is obtained by mapping face pixels in the original image with white pixels in the binary image and background pixels in the original image with black pixels in the binary image. After inspection of

Figure 4.2, we decided to set the thresholds to $\theta_{min} = 0°$ and $\theta_{max} = 30°$. Different other threshold values have been tested on a set of 10 face images, but none gave results that outperformed those obtained with $\theta_{min} = 0°$ and $\theta_{max} = 30°$.

Figure 4.3: The result of a hue-based face detector

Even though color can be a very discriminative feature for face detection, it is better to combine color information with other sources of information to achieve a higher detection robustness and accuracy. To realize the truth of this assumption, we invite the reader to inspect Figure 4.4 that shows the type of objects that a detector based on color would provide when an edge detection algorithm is applied on the result of the detection filter.

When considering detectors based on skin color detection, it is legitimate to wonder whether or not the technique could be applicable to subjects of any ethnicities or if the parameters of the filter should be tuned to adapt to each individual ethnicity. The answer to this question is that the same filter can be applied to all subjects, because the chrominance of the skin is almost identical for all human beings. As it might be difficult to believe at first glance, we invite the reader to inspect Figure 4.5 that illustrates this assumption. On this figure, the same parameters have been used for the three images under consideration.

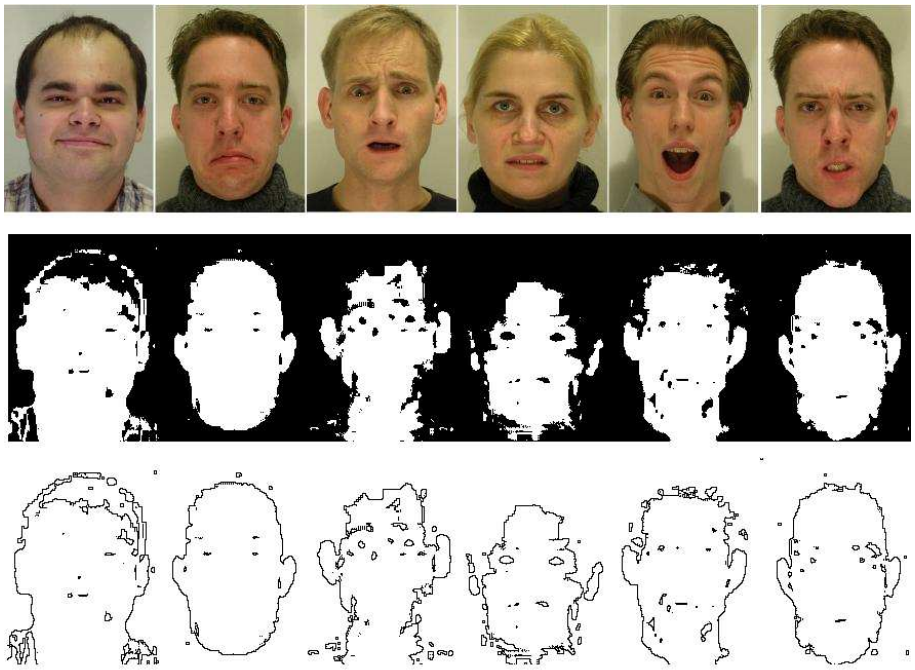


Figure 4.4: The result of a hue-based face detector. The straightforward application of an edge detection filter does not lead to a satisfying result

Figure 4.5: The filter can be used with the same parameters on all subjects, independently of their ethnicity. This figure shows the result of the filter applied on a) A Nigerian woman, b) A Japanese woman, c) An Indian woman

4.2.3 Face Contour Extraction

The techniques discussed in this section aim at extracting the face from an arbitrary image. We have seen so far that skin-color detection is a efficient approach when the background does not contain skin-look-alike regions. Pixel-based segmentation consists in classifying each pixel as belonging either to a face or a non-face region, based on the comparison of its hue value with two selected thresholds. As can be observed in Figure 4.3, many pixels lying inside the face region are not actually detected as belonging to the face because they have a hue value outside the range associated with skin pixels. Instances of such pixels are located around the eyes, the eyebrows or even to some areas of the lips. To cope with this issue and be able to localize and extract the face region from the background, we will rather estimate the face contour and consider all pixels inside that contour as belonging to the face.

The straightforward application of an edge-detection filter on potential face regions leads to images containing a large set of interlacing edges, as depicted in Figure 4.4. Since we use the extracted face contour as jump-start of the detection of the facial features, we have to maximize the robustness of the face contour extraction if we want to have the whole facial feature extraction process working properly. Face contour extraction constitutes the first step of a chain-process and we can therefore not afford to settle for the results presented in Figure 4.4. With this consideration in mind, we can admit that we would greatly benefit from the imposition of shape constraints on the extracted face contours, in order to smooth the irregular contours of Figure 4.4. As a matter of fact, if we impose the face contour to be egg-shaped, we can improve the segmentation procedure and get rid off unrealistic face contours while at the same time easing the edge classification task, which consists in determining if an edge belongs to the face contour or not.

Once a face region has been detected, we will place an ellipse around the set of pixels that are expected to belong to a face region, and search in the neighborhood of this ellipse for pixels containing a large chrominance and luminance gradient. These pixels containing high gradient values are likely to belong to the face boundary. To add regularity constraints to the extracted contour, we simply favor shapes that catch up to an ellipse, by adding stiffness constraints to potential contours. In

the remaining of this section, we will describe this procedure in detail and illustrate its two steps with a practical example.

Figure 4.6 depicts the whole procedure. Once the color filter has detected a potential face region, an ellipse embedding this potential face region is drawn. Then, the boundary of the ellipse is divided into a set of M segments of curve. The M points that separate a segment from its two adjacent neighbors will be noted by P_i with $i \in [1, ..., M]$. At each point P_i, we trace a line segment S_i perpendicularly to the curve of the ellipse at that point[1]. In the HSV color-space described in the last section, we compute the luminance gradient (∇L) as well as the hue and saturation gradients (∇Hue and ∇Sat) for each point $u_{i,j}$ of the segment S_i (j can refer to any pixel belonging to that segment). We then combine for each point $u_{i,j}$ the values of the three channels needed to compute the hybrid gradient $\nabla H_{i,j}$:

$$\nabla H_{i,j} = \frac{2\nabla L_{i,j} + 4\nabla Hue_{i,j} + 1\nabla Sat_{i,j}}{7} \tag{4.1}$$

As equation (4.1) shows, we do not assign the same weight to the different hybrid gradient components. As the hue is much more discriminative than the saturation for instance, we will assign a greater weight to the hue gradient.

The weights of each hybrid gradient component were chosen empirically, by a process of trials and errors on a set of three different pictures[2]. In another words, the weights were adjusted until the result of the detection was satisfactory for the three considered images. It should be noted that a more efficient weighting determination could be achieved by measuring the performance obtained with each individual gradient and then assigning a weight proportional to these individual performances, such as in traditional boosted detection algorithm.

For each line segment S_i, we determine the point h_i having the maximum hybrid gradient value:

[1] In the proposed implementation, instead of dividing the ellipse in segments of curve, we have divided each line of the ellipse bounding box into equally spaced intervals. The line segments are drawn along the lines linking the interval extremities with the center of the bounding box, which produces a similar result.

[2] The results presented in this section were obtained on images that *were not* used for the determination of the weights

$$\forall i \in [1, \ldots, M] : \qquad h_i = \arg\max_j \nabla H_{i,j} \tag{4.2}$$

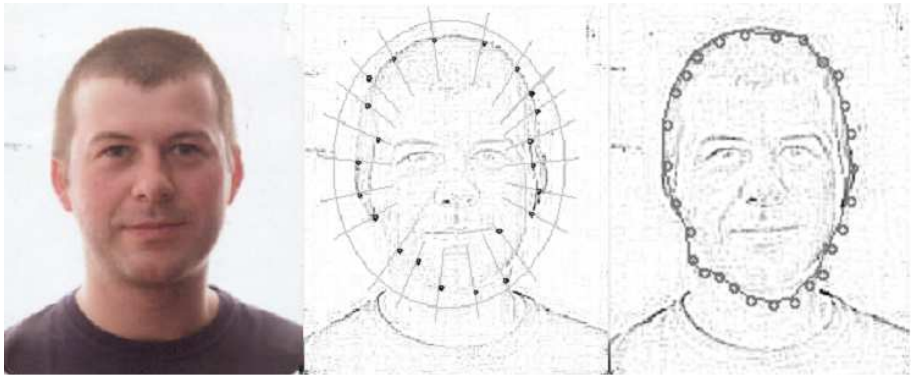


Figure 4.6: Imposing shape constraints enables to extract a reliable and precise estimation of the face contour

The procedure that we just described is illustrated in figure 4.6b: the black dots represent the maximum gradient points h_i along each segment S_i.

As the strongest edges do not always correspond to the boundaries of the face (the maximum gradient point h_i of a particular segment S_i might not correspond to a point located on the face contour), we introduce a *stiffness constraint* on the relative positions of the successive maximum gradient points h_i.

Let us define the relative position of a point h_i on a segment S_i by R_i, which is allowed to vary in the range $[0, \ldots, 1]$. For each segment S_i, a relative position $R_i = 0$ means that the maximum gradient point h_i is located at the segment extremity that is the closest to the center of the ellipse, while a relative position $R_i = 1$ implies the opposite: h_i is located at the extremity which is the furthest away from the ellipse center. Of course, all intermediate values are valid and indicate where on the segment S_i h_i is located. The stiffness constraint that we impose

is that the relative position of all the maximum gradient points h_i should satisfy the three following conditions simultaneously:

$$R_{i-1} - r \leq R_i \leq R_{i-1} + r \tag{4.3}$$

$$R_{i+1} - r \leq R_i \leq R_{i+1} + r \tag{4.4}$$

$$0 \leq R_i \leq 1 \tag{4.5}$$

where r is a margin, whose value can be adjusted to maximize the accuracy of the contour detection algorithm. In Figure 4.6c, we see that by applying a margin of $r = 0.2$, we were able to correct the detection errors of the lower-right corner of Figure 4.6b. This stiffness constraint smooths the contour to be detected, and thereby enables the system to detect a face contour, even when some of its edges are not apparent, by incorporating knowledge about the shape of the object to be detected. Once the face image has been correctly extracted, we can switch to the next step of the analysis process, which concerns the extraction of affective information from the extracted facial image.

4.3 Facial Feature Detection and Tracking

We have seen in the introduction of the present chapter that facial features such as the mouth, the eyes and the eyebrows reveal much of the affective information carried out by the facial expression of an emotion. This section therefore aims to provide a method to extract in real-time the shape of these facial features, in order to produce reliable input data for the facial expression recognition algorithm.

The starting point of our exploration is a face image of good quality. From that initial face image, we need first to locate the facial features that we are searching for. Once we know their position in the face, we need to find a way to describe their state, knowing that these states depend on the behavior of a set of underlying facial muscles [24]. As facial muscle actions have as a visual consequence the deformation of the shape of the related features, we will use shape-related measures to describe the state of the facial features. Since the contour contains all the information about the shape of a deformable object, we will focus on the

real-time tracking of feature contours. The relevance of our reasoning is illustrated in Figure 4.7 [26], which depicts the evolution of the facial feature contours when joy or surprise is experienced. Our goal is to be able to obtain and represent the information contained in this figure.

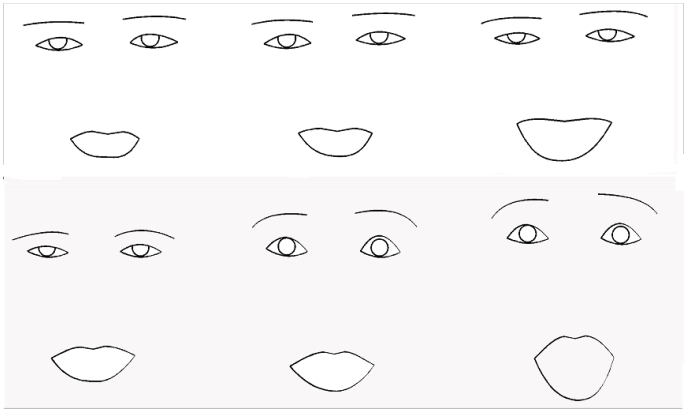


Figure 4.7: The evolution of the facial feature contours reveals the emotion experienced by the user. *Above*: The **joy** emotion. *Below*: the **surprise** emotion

.

Contour tracking is actually a two-step process. First the contours need to be detected in the first frame of the video sequence. Then, a tracking algorithm can follow their evolution in subsequent frames. The remaining of this section will thus be divided into two parts: the first part presents the feature detection process, while the second part concentrates on the tracking algorithm itself.

4.3.1 Facial Feature Detection

To achieve the detection of the facial feature contours, we must first determine in which area of the face the facial features have to be searched for. Old techniques divide the face into a set of distinct regions, each of which containing one of the facial features to be detected [32]. For instance, the lower part of the face contains the mouth, while the upper part is divided vertically into two regions, each of which containing an

eye and an eyebrow. Once the face has been divided in distinct regions, these techniques search for specific facial features in each of the defined region. This way of proceeding is not very efficient as it involves large search-spaces that might lead to both complexity and robustness issues.

The most successful approaches known so far use either Active Shape Models (ASM) or Active Appearance Models (AAM) [33] to extract facial features [34], by describing the shape (appearance) of each feature by a vector of parameters $\mathbf{x}$. In ASM, each of these vectors contains the coordinates of a set of points, whose positions describe the shape of the considered feature. In AAM on the other hand, the vector $\mathbf{x}$ contains the appearance, i.e. the texture, of the pixels belonging to the area of interest (the face). To determine the value of the elements of $\mathbf{x}$, we build a training set T containing N instances of each feature. For each feature, the mean vector $\overline{\mathbf{x}}$, computed by averaging the values of the elements of $\mathbf{x}$ over the N instances contained in T, is an estimation of the average shape (appearance) of the considered feature. Any instance of the feature may then be modeled by a weight vector $\mathbf{b}$ that contains the shape (appearance) variations of the feature with respect to the average shape (appearance):

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{P}\mathbf{b} \tag{4.6}$$

where $\mathbf{P}$ is the matrix of the first t modes of variation, p_i, corresponding to the t most significant eigenvectors in a Principal Component Decomposition of the position (appearance) variables. As the columns of $\mathbf{P}$ are orthogonal, we have $\mathbf{P}^T\ \mathbf{P} = \mathbf{I}$, and thus $\mathbf{b}$ can be computed with:

$$\mathbf{b} = \mathbf{P}^T(\mathbf{x} - \overline{\mathbf{x}}) \tag{4.7}$$

When a new face is analyzed, the system first sets the average model onto the face and then varies its parameters to better fit the image data, but only in ways that are consistent with the shapes (appearances) found in the training set, using shape (appearance) variations encoded by the N weight vectors $\mathbf{b}$.

In this thesis, we have developed a similar approach[3]. We used a

[3]The approach developed in this thesis is similar to ASM-based technique in the sense that it tries to extract feature point localization using a priori statistics computed from a database of training exemples

large facial expression database that we have manually labeled. The information that is extracted from the database allows the computation of statistics on the location and the shape of the features we aim to detect. These statistics will later be used as *a priori* information in the detection process. Boosted detection techniques may then be used within local search-spaces defined with the help of the computed statistics. Whereas the idea is very close to the ASM approach, we do not actually decompose the position variables into their principal components. Instead, we model the distribution of these variables around the mean positions by 2D Gaussian distributions, which are used to define probabilistic search-spaces. To each pixel of these search spaces is associated a probability for a specific feature point to be found, whose value is proportional to the value of the 2D distribution at that location. Before going further into the description of the statistics and their use for facial feature detection, let us first examine the data acquisition process.

Data Acquisition

As mentioned earlier, we used for our learning process a large facial expression database: the Cohn-Kanade Facial Expression Database [35]. This database contains 348 video sequences of people expressing specific emotions. Each video sequence shows the face of a subject evolving from a neutral state (in the first frame of the sequence) toward the emotion expressed at its maximal intensity (in the last frame of the sequence). For our facial feature detection problem, we considered only the first frame of each video sequence, which shows the subject in a neutral emotional state. On the other hand, to model facial expressions, we considered the comparison between the emotive state (last frame) and the neutral state (first frame).

For each considered frame, we manually labeled a set of 25 facial points, which are depicted in red in Figure 4.8. The labeling of the database was conducted by two human experts with the help of a dedicated graphical user interface (GUI), developed in Python. This interface provides a visual guide that indicates to the user which point should be marked next at each step of the labeling process. The interface also allows labeling errors to be corrected interactively, by right-clicking on the miss-marked points. Figure 4.9 shows the dedicated graphical user

interface.

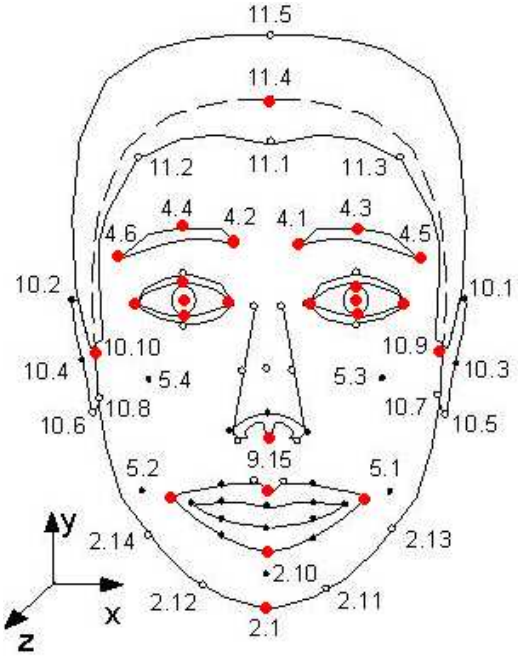


Figure 4.8: The set of 25 points that were labeled for the purpose of facial feature detection

.

Once a frame has been labeled entirely, the interface stores the coordinates of the clicked points into a file. At the end of the labeling process, we therefore have 348 files, each containing the coordinates of the 25 marked points.

Statistical Facial Feature Bounding Boxes

The data that have been extracted from the database are equivalent to the data that could have been obtained with an optimal feature detector, which never fails to provide a precise estimation of the coordinates of a set of feature points. As introduced earlier, these points can be used to compute statistics on the position and shape of the facial features to

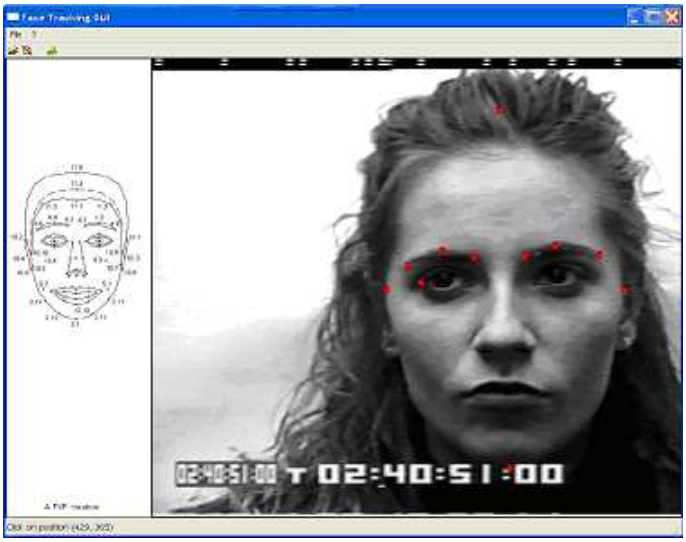


Figure 4.9: The Python graphical user interface. The visual guide is displayed on the left of the screen. Points that have already been labeled are marked with red dots

.

be detected. This information can be used as an a priori information to initialize search-spaces close to the real position of the features. Proceeding this way allows the initial problem of finding a set of features in an entire face image to be reduced to a much simpler one, as only very local search algorithms have to be employed. This can reduce drastically the computational load whereas increasing the robustness of the detection algorithms.

The proposed approach consists of computing the means and standard deviations of the position and size of the *Facial Feature Bounding Boxes* (FFBBs), from the data extracted from the database. While the means represent the most likely size and location of the bounding boxes, the standard deviations reflect the reliability of the estimations and thus indicate how far from the most likely FFBBs feature boundaries have to be searched.

For obvious normalization purposes, all the results that we will be presented are expressed relatively to the center of a normalized bounding box of the face, whose size is fixed to 200x200 (as depicted in Figure 4.10). Table 4.1 presents the average position and size of the five FFBBs that we will consider in this work, while Table 4.2 contains the data related to the standard deviations of the position and size of these FFBBs.

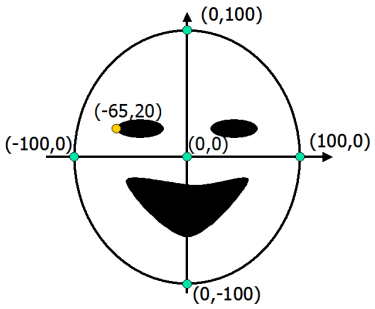


Figure 4.10: All coordinates will be expressed relatively to a normalized bounding box of the face, whose size is fixed to 200x200

.

A quick inspection of Tables 4.1 and 4.2 enables us to notice a few

Table 4.1: Average position and size of facial feature bounding boxes

MEANS	Center	Width	Height
Left Eye	(44,8)	39	8
Right Eye	(-45,8)	40	8
Left Brow	(50,23)	59	6
Right Brow	(-51,22)	59	7
Mouth	(0,-58)	73	16

Table 4.2: Standard deviations of the position and size of facial feature bounding boxes

STD DEV	Center	Width	Height
Left Eye	(5,7)	3	2
Right Eye	(5,8)	3	2
Left Brow	(5,10)	5	3
Right Brow	(5,10)	4	3
Mouth	(5,5)	5	3

interesting facts about the human face morphology. The center of the eye bounding boxes have a horizontal coordinate around -45 and +45 respectively, and a width of 40. As the overall face width is 200, we see that *the width of an eye corresponds exactly to a fifth of the entire face width*. Moreover, the distance between both eyes is also around one fifth of the face width. On the vertical scale, we notice that the eyes are slightly above the middle of the face: only 4% above the middle line, which contradicts the general belief that eyes clearly belong the the upper part of the face! We also have an indication of the reliability of these assertions: the standard deviations of the position of the eyes range from a minimum of 2% to a maximum of 4%, which also means that the position of the eyes do not vary much between distinct individuals. The eye's bounding boxes width and height standard deviations are respectively of 7.5% and 25% of the average eye's size.

The statistics concerning the eyebrows show that the variability among individuals is greater than in the case of the eyes: the standard deviations of the bounding box center's coordinates vary between 2% and 5%, which is slightly more than in the case of the eyes. The standard deviation of the size of the eyebrow's bounding boxes is around 7% for the width and 46% for the height.

The figures related to the mouth reveal that the mouth is located around the middle of the lower face half, and that the average width is around 35% of the total face width. The standard deviation of the position of the mouth is quite low (lower than 3% in each direction), while the mouth's size is associated with standard deviations of respectively 7% for the width and 19% for the height. These figures tell us that mouth shapes are relatively similar from an individual to another.

Local Search-Space Algorithms

Once the search-spaces of a particular facial feature have been defined, a variety of algorithms may be used inside these search-spaces to extract feature information. Figure 4.11 illustrates an example of such a local search-space algorithm.

In this example, the means are used to place a rectangular window close to the true mouth's bounding box, which has to be found. To find

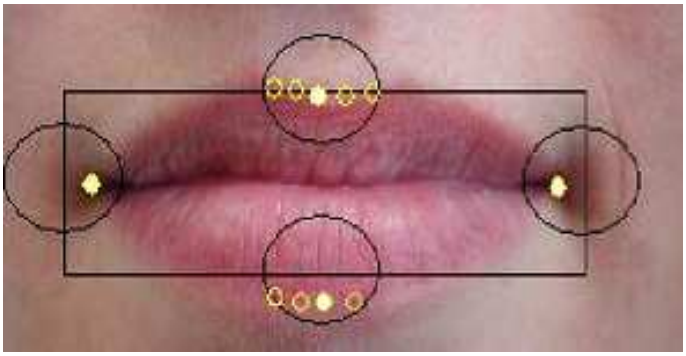


Figure 4.11: Detecting mouth boundaries, using circular search-spaces

the points defining this bounding box, elliptic search-spaces are centered on the most likely locations. Their size is proportional to the standard deviations of the mouth bounding box. Inside the elliptic search-spaces, standard algorithms, such as Harris corner detection [36] (to detect lip corners) and/or edge detection methods, such as Sobel or Canny edge detection [37] (to detect upper and lower lip boundaries) may be employed. On a similar fashion, if the method that is chosen is region-based segmentation [38], color or luminance information may be processed only inside the search-spaces.

If the algorithm used inside the search-space provides several candidates, the distance to the center of the search-space can be used as a discrimination factor, as candidates nearest to the center of the search-space are statistically more likely to be the points to be detected. This situation is illustrated in Figure 4.11 where empty yellow circles represent candidates that have been discarded. A similar methodology may be used for eye and eyebrow detection, based on the statistics presented above.

Now that we have shown how statistics can ease the problem of facial feature detection, we will illustrate the underlying ideas with a practical example involving the detection of a set of 10 feature points. We will show that satisfying results can be obtained when a priori information is added to traditional detection algorithms, at least for facial images whose resolution and contrast are sufficiently good. However, even when

our method fails to provide a correct detection of the facial feature points we are searching for, we believe that the method remains very useful to initialize ASM or AAM-based deformable face models, as illustrated in [39].

Toward a Boosted Facial Feature Detector

In this subsection, we will present the first prototype of a facial feature detection algorithm that uses the presented statistics along with luminance and gradient-based information to generate an estimation of the positions of a set of facial feature points. The search-spaces that are used in this prototype are built using a 2D Gaussian distribution $N[(\mu_x, \mu_y), (\sigma_x, \sigma_y)]$, where (μ_x, μ_y) corresponds to the most probable location of the point we aim at detecting and σ_x (σ_y) denotes its average horizontal (vertical) standard deviation. For the sake of simplicity, we have used the same value for σ_x and σ_y, so as to have circular search-spaces. Based on the results presented in Table 4.2, we chose to use a standard deviation of 5. Numerically speaking, it means that we have, for all the search-spaces that we will use in our example:

$$\sigma = \sigma_x = \sigma_y = 5 \tag{4.8}$$

Figure 4.12 illustrates the above consideration by showing the discretized luminance profile of such a Gaussian-generated search-space.

The search-spaces are thus centered on the most probable *a priori* locations (that are computed from the mean values of Table 4.1). To integrate this *a priori* information, we create an *a priori image* that will contain high luminance values for pixels close to the center of each search-space and lower values of luminance for pixels that are further away from these centers. More precisely, the center of each search-space receives in this a priori image a value of 255, which corresponds to the maximum value on a 8-bit luminance intensity scale. Pixels adjacent to these so-called central pixels receive a value that is determined by their corresponding 2D Gaussian distribution, scaled so that the maximum of the distribution receives a value of 255. The 2D Gaussian probability distribution is given by:

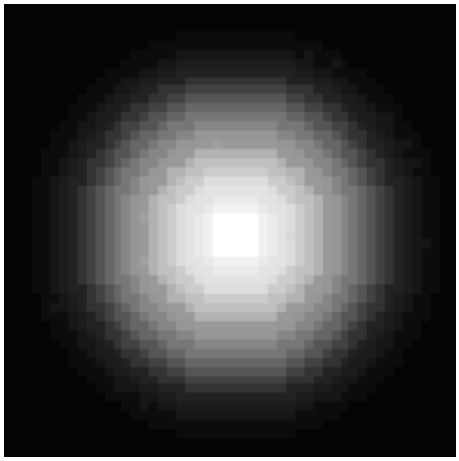


Figure 4.12: Circular Gaussian Search-Space, light values correspond to highly probable locations

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2} - \frac{2\rho xy}{\sigma_x\sigma_y}\right)\right) \quad (4.9)$$

As in our case, we consider the horizontal and vertical component of this distribution independent of each other, we assign 0 to the value of the correlation coefficient ρ, so that equation (4.9) can be rewritten in a simpler form:

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp\left(-\frac{1}{2}\left(\frac{x^2}{\sigma_x^2} + \frac{y^2}{\sigma_y^2}\right)\right) \quad (4.10)$$

The a priori image, depicting ten of these search-spaces (four of them concern the mouth bounding box and the six remaining ones are located on both eyebrows) is shown in Figure 4.13.

Sobottka et al. have shown that facial features very often correspond to regions of lower luminance [40]. They have developed an approach for facial feature detection based only on this consideration, and their results were exploited by a variety of studies. We therefore decided to include this criterion into our system and used the invert of the luminance value of each pixel as an indicator of feature belonging. Figure

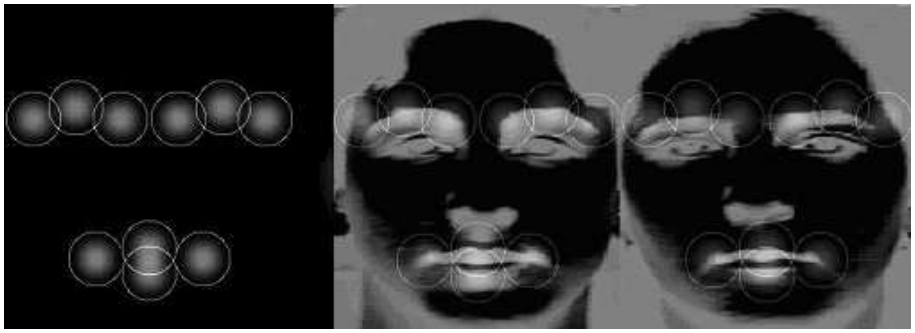


Figure 4.13: *On the left:* The set of search-spaces represents the a priori image. *On the right:* Fusing a priori image with inverted luminance image

4.14 shows the grayscale image that is obtained after equalizing the luminance histogram of the image and inverting pixel values. We see on this figure that facial features are represented by very bright pixel intensities, which reflects very low luminance values in the original image.

The right part of Figure 4.13 shows the result of the fusion of the a priori image, depicted on the left of Figure 4.13, with the original grayscale images. The fusion is made by first multiplying pixel-by-pixel both the a priori image and the original image, and then by scaling the result so that the resulting pixel values stay within the same range as the original images. We can see on this figure that all the feature points are located within their respective search-space. This has been the case for all the face images investigated so far when the search-spaces are generated with a standard deviation whose value is greater than 2% of the overall face width. It must be mentioned that the white circles corresponding to the boundaries of the individual search-spaces have been displayed on the figure to illustrate the fact that all the feature points were lying within them. In the detection algorithm, these surrounding circles are not considered. The images that result from the fusion show very bright areas around the locations of the feature points. Luminance thus brings very useful information: pixels that do not belong to a feature have generally lower values. However, as we are searching for feature boundaries, we need additional information to distinguish the

Figure 4.14: Facial features correspond to regions of lower luminance (inverted values)

pixels that lie inside the features from those which also belong to their boundaries.

It seems natural to integrate in our system the information carried by both the luminance and the chrominance gradients, as both are likely to have much larger values along the contour of the features. In this work, as we are working with a database containing only grayscale images, we will however only consider the gradient of luminance. Applying a gaussian gradient filter on the three original images leads to the result depicted in Figure 4.15. We see that, in both cases, the feature boundaries correspond to very high gradient values (dark pixels on the figure).

The output of the facial feature detector is obtained by fusing the information of the three different images that have been presented. The idea is that detected points will be those satisfying the three following conditions:

1. Being close to the *a priori* location of the feature point to be detected (a priori image)

2. Belonging to the feature (luminance image)

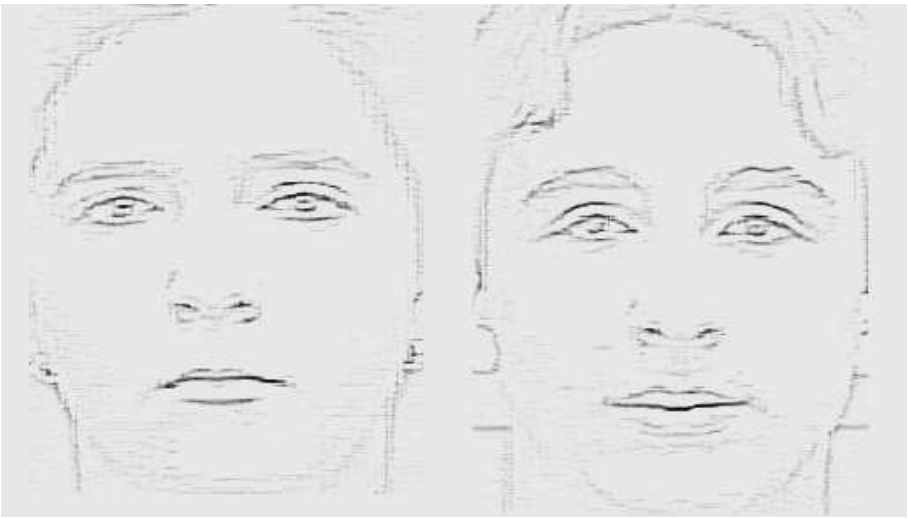


Figure 4.15: Contour profile of the three faces under consideration

3. Belonging to the boundary of the feature (gradient image)

In this first version of the prototype, we have simply multiplied the value of the pixels of an image by the values of the corresponding pixels in the two other images and normalized the result to keep the same 8-bit intensity scale. In boosted detectors, the relative weights of each source of information should be made proportional to the relative increase of performances that the individual sources carry.

The final result that was obtained with this detector on the two considered examples is shown in Figure 4.16. Red pixels represent the features point's candidates that have been detected. They correspond to the pixels having the highest value in the resulting fused image, for each of the search-spaces.

If we analyze the red pixels located around the mouth, we see that mouth corners are detected in both cases, whereas the mouth's upper and lower boundaries seem to need extra processing. We believe that the gradient histogram along a vertical segment of line would turn out to be a much more robust feature for detecting mouth's upper and lower

boundaries (in comparison with individual gradient values, as used in the results shown in Figure 4.16).

Concerning the eyebrow regions, we can notice that red pixels are sometimes associated with part of the hair. In both presented cases however, this would not be problematic in the sense that such pixels will be discarded automatically by the feature tracker that takes into account implicitly the probability of every feature deformation, and would therefore not consider such pixels as plausible candidates.

Figure 4.16: The final result of the facial feature detection algorithm on the two considered faces

4.3.2 Facial Feature Tracking

Video tracking is usually defined as the process of locating a moving object (or several ones) over time, using a video camera. Algorithms that aim to match an object in successive video frames traditionally use block-matching approaches, which consists in determining the location of a block of pixels at frame i, based on its position at frame $i-1$. Unfortunately, this approach only holds for the tracking of rigid objects, as the object appearance must remain similar for successive frames. In the case of deformable objects, the problem is harder because the shape of

the object to be tracked may change over time. The tracking algorithm must therefore detect the change in location **and** appearance.

In the particular case of facial feature tracking, the objects remain at their initial location, except for the case of eyebrows, whose vertical positions are slightly variable. The challenge is thus to capture the shape of the feature. Among the most effective approaches to track the shape of a deformable object over time are techniques based on deformable parametric templates such as *Active Shape Models* (ASM) or *Active Contour Models* (or *'snakes'*) [41].

Compared to Active Contour Models (snakes), Active Shape Models integrate prior information about the shape of the object to be detected. Rather than expecting desirable properties such as continuity and smoothness to emerge from image data, those properties are imposed from the start [42]. The approach thus integrates shape constraints in the edge tracking algorithm. In section 4.3.1, we briefly introduced the fundamental principles that constitute the ASM approach. We have seen that a new shape $\mathbf{x}$, constituted by a set of N landmarks x_i, is represented by the combination of an average shape $\overline{\mathbf{x}}$ and a weighted sum of variation modes, represented in matrix form by $\mathbf{Pb}$ where $\mathbf{b}$ is the vector of weights and $\mathbf{P}$ the matrix whose entries are the eigenvectors of $\mathbf{x}$.

The facial feature tracker that was used in this work is based on a variant of such active shape models. In the proposed version, the deformable models corresponding to the different facial features are represented by a set of masses linked to each other by a network of springs. These networks are interconnected in order to form a deformable face model. The real time deformation of this face model is driven by a set of decoupled equations, valid under the linear elasticity framework, which allows real-time tracking[4] of the facial features to be performed. The work presented in this section has been realized predominantly by Stelios Krinidis, from the team of Professor Ioannis Pitas (Aristotle University of Thessaloniki, Greece). A comprehensive description of the mathematical foundations supporting this work can be found in [43], [44], [45], [46] and [47]. In the remaining of this section, we briefly comment the principles that supported the development of this facial feature tracker.

[4]at a framerate greater than 20 frames per second

Tracking Facial Features Using Deformable Face Models

The facial feature tracking algorithm considers an object undergoing an elastic deformation as a set of masses linked by springs, where the natural length of the springs is set equal to zero, and is replaced by a set of constant equilibrium forces, which characterize the shape of the elastic structure in the absence of external forces. Modeling the problem this way enables the use of dynamic equations that are linear and decoupled for each coordinate, whatever the amplitude of the deformation [44].

The governing equations established in [43] and [44] can be applied to the problem of real-time tracking of the contour of the facial features, each of them being associated with a distinct deformable shape model. In addition to these individual shape models, there exist intrinsic relationships between the different facial features. The idea behind facial feature tracking based on deformable face models is therefore to exploit knowledge about the face morphology so as to maintain plausible spatial relationships between facial features, which results in an increase of the robustness of the tracking algorithm.

Among the different face models available in the literature, we chose to use the well-known Candide wireframe model [48] as a deformable model. Candide is a parameterized face mask specifically developed for model-based coding of human faces. A frontal and a profile view of the model is depicted in Figure 4.17.

When the tracking algorithm is launched, in the first frame of the video sequence, the grid is initially placed at the center of the face. Once the facial feature detection algorithm outputs the estimation of the positions of the ten features points that it automatically detects, the grid's position and size is adjusted by matching the positions of the points detected in the first frame of the image with the corresponding grid nodes. The resulting automatic initialization procedure is illustrated in Figure 4.18.

In Figure 4.18, only the 10 points that are detected by the automatic facial feature detection module are used to initialize the face mask. Once this initialization has been performed, the tracker uses the governing equations to update the positions of the grid nodes and thereby follow

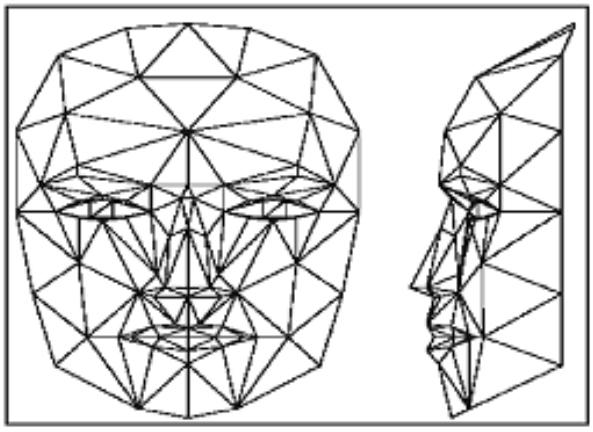


Figure 4.17: Frontal and side view of the Candide wireframe model

facial expressions appearing on user's face. Figure 4.19 shows how the face grid may deform to follow the expression of emotions.

The use of explicit equations enables the tracker to output the feature contour displacements' estimations at a frame rate of around 23 fps, which is largely enough for real-time applications. However, when the subject is speaking, the tracker often fails to remain anchored around the lip contour. It is therefore not fast enough to be employed in multimodal emotion recognition systems, in which facial expressions and speech information are combined to form a multimodal emotional vector.

The external forces that have been used in the current implementation of the tracker are only based on luminance gradient information. Although it is the most obvious feature to be used for contour tracking in gray level image, it does not constitute a robust technique for facial feature tracking. As we discuss in the conclusion of this thesis, we believe that future research directions should include a new definition of the external force. A first idea would be to include a measure of the similarity of the histogram profiles (possibly in two dimensions) around each node whose tracking has to be performed. A trade-off has however to be done between the quantity of information that is taken into account and the framerate at which the tracker should perform. With these con-

Figure 4.18: The initialization procedure with 10 feature points (4 points for the mouth, and 3 for each eyebrow)

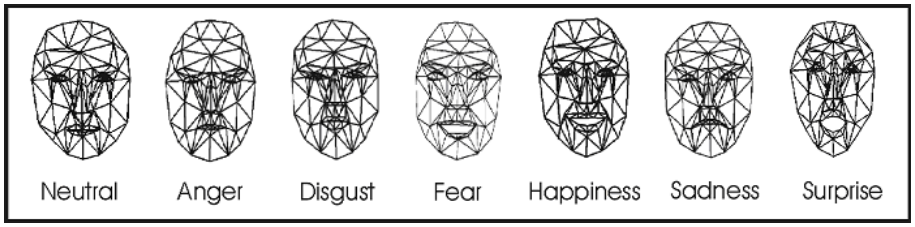


Figure 4.19: Examples of the deformed grids produced by the tracker for different facial expressions

siderations in mind, we recommend the use of a multiresolution block matching approach, such as used in the Lucas-Kanade-Tomasi tracking algorithm [49], but applied to each feature point to be tracked.

4.4 Conclusion

This chapter discussed the automatic extraction of affective information from a video sequence. We started our discussion by dividing our problem into three distinct parts: face detection, facial feature detection and facial feature tracking.

We briefly studied the problem of face detection. Among the many different approaches constituting the state of the art, we chose as a starting point color-based techniques. We showed that all skin colors belong to a relatively small portion of the hue-saturation space and that hue was a very discriminant feature. We also demonstrated that a face detector built on this criterion can work with faces from any ethnicity, without tuning the parameters of the detector. We presented several segmentation results, achieved with such a hue-based face detector, and concluded by assessing the need to include other sources of information to produce a reliable estimation of the face contour. To reach that goal, we explained how an original approach including constraints on the shape of the face as well as gradient-related contour extraction could provide a satisfying solution to the proposed problem.

Once a face image had finally been extracted, we focused our attention on the problem of facial feature detection. Our approach used a

set of statistics computed from a large facial expression database. We showed how these statistics could ease the facial feature detection by providing a priori search-spaces, thereby reducing the complexity of the original problem while increasing the robustness of the detection. We illustrated our ideas with a practical detection algorithm, which combines luminance and gradient information with statistical knowledge of the face morphology to produce an estimation of the position of a set of ten facial feature points.

We concluded this chapter by presenting the automatic initialization of a facial feature tracking algorithm, based on the detection of ten facial feature points. After having briefly defined the tracking algorithm, we explained the weaknesses of the developed tracker by assessing the need for a more global similarity measure, which requires the inclusion of another definition for the external force. Our last words suggested to include in the proposed approach a multiresolution block-matching similarity measure, such as implemented in the Lucas-Kanade-Tomasi algorithm.

Chapter 5

Facial Expression Recognition

5.1 Introduction

As we introduced earlier, the recognition of facial expressions from a video sequence can be seen as a two-step process. The first step is to extract relevant affective information from the video sequence, while the second step consists in the analysis of this information to detect facial expressions. The present chapter will be devoted to the latter of these two steps: analyze extracted information to detect the presence of facial expressions.

Several techniques are available to analyze facial expressions. Among the most popular approaches, we could mention the analysis of the displacements of features points [50] [51] or the analysis of the deformations of the shape of facial features [52]. To provide a common basis, a comprehensive representation of the set of deformations that the face may undergo has been proposed. The *Facial Action Coding System (FACS)* developed by Ekman and Friesen in the 1970s [53] determined how the contractions and dilatations of each facial muscle change the appearance of the face. They developed a manual illustrating changes of appearance on the face using written descriptions, still images and digital video examples. The units that are used to describe the facial muscle actions are called *Action Units (AUs)* and correspond to the contraction or di-

latation of one or several facial muscles.

Although it is tempting to conceive a facial expression recognition system based on the entire set of AUs, proceeding this way does not necessarily leads to a satisfying solution. The main limitation is that such a system would require a reliable estimation of *all* the positions of the feature points involved in the expression of emotions, whereas some of these points are indeed very difficult to locate for an automatic tracking device (such as points on the cheeks that are located in low-texture areas). Consequently, most of the systems that have been developed so far integrate only the AUs that are both easily detected and very discriminant, that is: the AUs that allow to distinguish without ambiguity **and** reliably an emotion that belongs to the classification system [54].

In addition to the choice of the features that will serve as inputs to the facial expression classification system, another important aspect to consider is the type of approach that will be retained. Two main paradigms can be employed: either we base our recognition on the results of previous psychological studies that have explored how emotions are expressed through facial expressions [22] [23] [24] [25], or we use learning algorithms to build statistical models of facial expressions. In this thesis, we will mainly focus on the latter approach, statistical classification, which is seen as the most efficient way to deal with uncertainty, as argued in [55].

Statistical classification can be described as a procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items and based on a training set of previously labeled items (*supervised learning*). Formally, the problem can be stated as follows: given training data $\{(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_n}, y_n)\}$, produce a classifier $h : X \rightarrow Y$, which maps an object $\mathbf{x} \in X$ to its classification label $y \in Y$.

Statistical classification algorithms are typically used in pattern recognition systems, which aim to classify data (patterns) based on either a priori knowledge or statistical information extracted from the patterns. The patterns to be classified are usually groups of measurements or observations, defining points in an appropriate multidimensional space. In the particular case of facial expression recognition, the task of the classifier h will be to predict the class label y (the facial expression) of an

input object $\mathbf{x}$ (the vector of input features), after having seen a number of training examples (i.e. pairs of input and target output), using probabilistic reasoning to form inductive rules from observed training examples. This means that the classifier will predict the class label of a previously unseen object $\mathbf{x}$ by selecting the class to which the object has the highest probability to belong to, based on a measure of the similarity of the new unseen object with the known (or learned) patterns.

Among the main approaches based on learned models of facial expressions, systems based on Neural Networks have been widely investigated. In particular, Tian et. al have used three layers Neural Networks to recognized a set of 20 AU's both in the lower and upper faces [54]. Wiskott et. al have use Neural Network after preprocessing a set of facial points using a 2D Gabor transform [56] and Lawrence et. al have developed a system combining local image sampling, a self-organizing map (SOM) Neural Network, and a convolutional Neural Network [57]. Systems based on Decision Trees and K-Nearest Neighbors were extensively tested by Sebe et. al [58], while different types of Bayesian Networks have been extensively used by Cohen et. al in [59]. The approach has been extended in [60] to handle the facial expression recognition from video sequences, using several types of Bayesian Networks. Finally, Support Vector Machines (SVMs) have been tested on the problem of facial expression recognition, leading to excellent recognition rates [61] [62].

In the scope of this thesis, we will examine the performances that can be achieved with two well-known statistical classifiers: Bayesian Networks (BN) and Support Vector Machines (SVM).

Bayesian classifiers have been chosen for their capacity to learn the relationships between variables (in this case, between the variable representing the facial expressions and those corresponding to the observation variables). They provide an intuitive and comprehensive representation of the problem (unlike Neural Networks for instance), which reveals very useful as our goal is to explore how emotions are expressed on human faces. Moreover, they are particularly suited to combine heterogeneous data, which can be very useful when combining several sources of information[1]. Finally, they are relatively easy to implement, handle

[1]such as in the case of multimodal emotion recognition

implicitely temporal aspects and can be conveniently used in real-time applications.

On the other side, SVM-based classifiers have the advantages of providing a global solution to the problem. They have proved to be one of the most promising classification techniques in a subsequent number of distinct challenges [72] [73] [74] [75]. They have the capacity to handle input data of very large dimension, which is undoubtly an important advantage in image processing problems, as the input vector might include entire blocks of pixels or even entire images, which makes them the best candidate for appearance-based facial expression recognition approaches. Finally, they are very well suited for real-time facial expression detection and the addition of new learning samples does not require the entire learning process to be relaunched, as new samples only need to be added if they become support vectors.

Once classification results have been presented and discussed, a last section investigates the use of facial expressions as a new modality to influence the narrative, in the context of Mixed Reality Interactive Storytelling . We present a short scenario that demonstrates how facial expressions, once recognized, can effectively lead to interesting entertaining interactive applications. We also briefly investigate the hardware requirements that this new modality requires in a practical implementation and the techniques that could be implemented to cope with the imperfections of the facial expression recognition system.

5.2 Data Acquisition

The data that are used to build statistical models of expressive faces were collected from the Cohn-Kanade Facial Expression database [35] using the dedicated graphical user interface presented in section 4.3.1. In the case of facial expressions, the goal is to characterize the changes of appearance of a set of facial features between the neutral and the emotive state. As the sequences contained in the database start with the first frame showing the neutral expression and evolve towards the last frame showing the emotion in its most intense form, we will represent the affective information as the difference of the feature values between the first and last frame of each sequence.

The first choice to be made is the type of affective information that will be considered. In order to be able to test a variety of different choices, we have decided to encode the coordinates of most of the Facial Animations Parameters (FAPs) related to facial expressions, following the Facial Action Coding System (FACS), described by Friesen and Ekman in [71]. More precisely, we have encoded the spatial evolution (between the neutral and the emotive state) of 28 facial points. This set of points allows to describe precisely the shape of the eyes, eyebrows and the mouth. Red dots on Figure 5.1 represent the 28 selected facial points that are related to facial expressions.

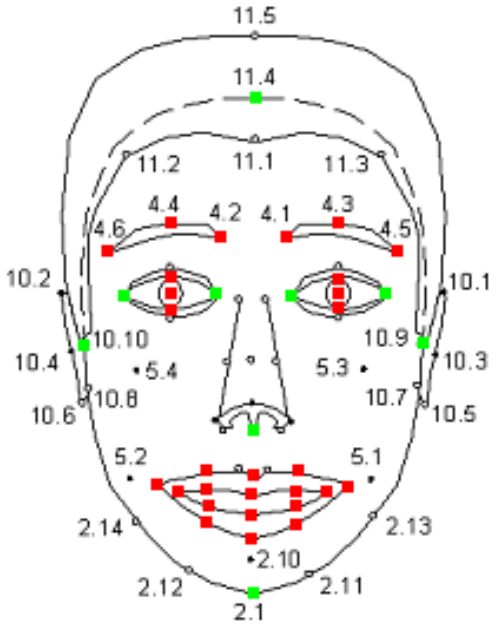


Figure 5.1: The set of 28 points that were labeled for the purpose of facial expression recognition

.

Like in the case of facial feature detection, the movements of these points have been expressed relatively to a face bounding box of size [200,200]. Figure 5.1 also contains a set of 9 points depicted in green.

These points are not involved in the representation of facial expressions. Instead, they are considered as fixed points and will be used to compute the rigid head motion between the first and last frame of the sequence (only translations have been taken into account at that stage). In addition, these points provide the position and size of the face bounding box (for face extraction and normalization purposes).

In order to be able to describe the facial features that have been used as inputs of our facial expression recognition systems, we will adopt the notations contained in Tables 5.1 and 5.2 to describe to which facial point we are referring to.

5.3 Feature Selection

Instead of computing the displacements of all the FAPs involved in the facial expression of emotions (after global head motion compensation), we decided to use as input features a set of relative facial distances. The decision to limit ourselves to a set of ten facial distances was motivated by different factors. First, these distances completely model the displacements and shapes that eyes and eyebrows can undergo, which means that the entirety of the information conveyed by the shape and configuration of the eyes and eyebrows is analyzed. We decided to consider only one distance for both the left and right eyes (eyebrows), by averaging the measures corresponding to the left and right eyes (eyebrows). Of course, doing this prevents us from detecting assymetric movements, but on the other side, it increases the robustness of the measures.

Concerning the mouth, we decided to use only three different distances: two of them analyze the horizontal and vertical movements of the mouth corners whereas the third monitors the vertical opening of the lips. We proceeded this way because in practice it is quite difficult to track the lip contour reliably as their movements are both very fast and highly non-linear. We will therefore examine the corners, which are the easiest parts of the lips and whose movements reveal much of the deformation undergone by the mouth. We will also track the distance between the upper and lower mouth boundaries, along the vertical axis dividing the face into two symmetrical regions, as the vertical opening of

Table 5.1: Notation of the points that are involved in the facial expression of emotions

Facial Point	**Notation**
Inner Left Brow	LB_{in}
Middle of Left Brow	LB_{mid}
Outer Left Brow	LB_{ext}
Inner Right Brow	RB_{in}
Middle of Right Brow	RB_{mid}
Outer Right Brow	RB_{ext}
Upper Left Eyelid	LE_{up}
Middle of Left Eye	LE_{mid}
Lower Left Eyelid	LE_{low}
Upper Right Eyelid	RE_{up}
Middle of Right Eye	RE_{mid}
Lower Right Eyelid	RE_{low}
Outer Left Mouth Corner	M_{olc}
Outer Left-Upper Mouth Point	M_{olu}
Outer Upper Mouth Point	M_{ou}
Outer Right-Upper Mouth Point	M_{oru}
Outer Right Mouth Corner	M_{orc}
Outer Right-Lower Mouth Point	M_{orl}
Outer Lower Mouth Point	M_{ol}
Outer Left-Lower Mouth Point	M_{oll}
Inner Left Mouth Corner	M_{ilc}
Inner Left-Upper Mouth Point	M_{ilu}
Inner Upper Mouth Point	M_{iu}
Inner Right-Upper Mouth Point	M_{iru}
Inner Right Mouth Corner	M_{irc}
Inner Right-Lower Mouth Point	M_{irl}
Inner Lower Mouth Point	M_{il}
Inner Left-Lower Mouth Point	M_{ill}

Table 5.2: Notation of the points that are involved in the computation of the global head motion, or for normalization purposes

Facial Points	**Notation**
Top of Skull	H_{up}
Leftmost Point of Face	H_{left}
Rightmost Point of Face	H_{right}
Bottom of Chin	H_{low}
Outer Left Eye Corner	LE_{oc}
Inner Left Eye Corner	LE_{ic}
Outer Right Eye Corner	RE_{oc}
Inner Right Eye Corner	RE_{ic}
Base of Nose	N_{low}

the mouth carries a lot of discriminant affective information. Figure 5.2 depicts the set of facial distances that constitute the facial information input vector $\mathbf{X} = D_1, \ldots, D_8$.

To summarize, the following set of equations defines the distances contained in the vector of observation variables $\mathbf{X}$. In these equations, Δx represents the difference between the value of x in the emotive and neutral state, that is: $\Delta x = x^{emo} - x^{neutral}$. Using this notation, the distances are formally defined by equations 5.1:

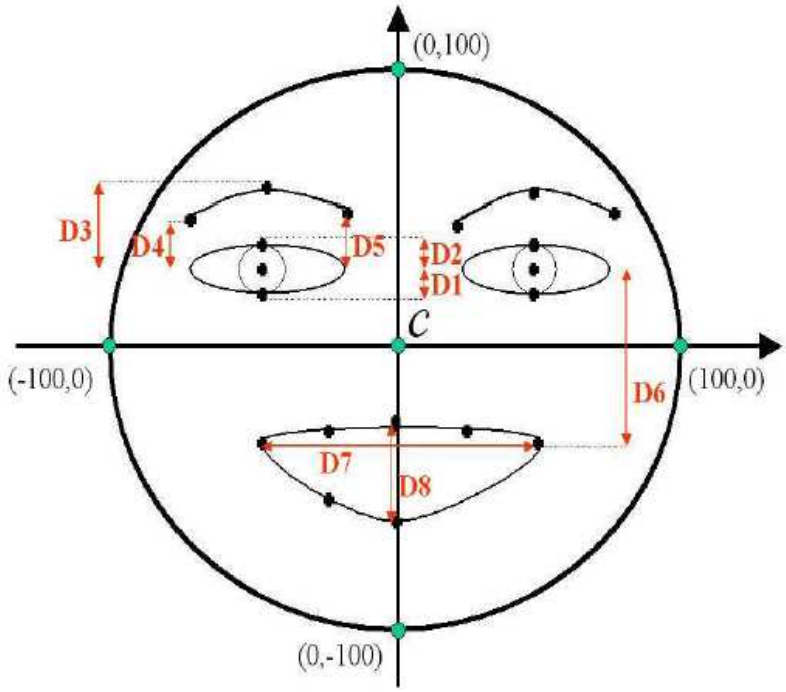


Figure 5.2: The set of facial distances that is used for facial expression analysis

$$
\begin{aligned}
D_1 &= \frac{\Delta(LE_{mid_y} - LE_{low_y}) + \Delta(RE_{mid_y} - RE_{low_y})}{2} \\
D_2 &= \frac{\Delta(LE_{up_y} - LE_{mid_y}) + \Delta(RE_{up_y} - RE_{mid_y})}{2} \\
D_3 &= \frac{\Delta(LB_{mid_y} - LE_{mid_y}) + \Delta(RB_{mid_y} - RE_{mid_y})}{2} \\
D_4 &= \frac{\Delta(LB_{ext_y} - LE_{mid_y}) + \Delta(RB_{ext_y} - RE_{mid_y})}{2} \\
D_5 &= \frac{\Delta(LB_{in_y} - LE_{mid_y}) + \Delta(RB_{in_y} - RE_{mid_y})}{2} \\
D_6 &= \frac{\Delta(LE_{mid_y} - M_{olc_y}) + \Delta(RE_{mid_y} - M_{orc_y})}{2} \\
D_7 &= \Delta(M_{olc_x} - M_{orc_y}) \\
D_8 &= \Delta(M_{ou_y} - M_{ol_y})
\end{aligned}
\tag{5.1}
$$

Facial expressions are the result of the contractions and dilatations of the facial muscles. Obviously, as the face contains tens of muscles, it is impossible to list all the combinations of motion that could produce a facial expression. Nevertheless, we need to restrict ourselves to a finite set of facial expressions that we would like to be able to recognize automatically. Most of the researchers investigating the automatic detection of facial expressions follow the taxonomy of Ekman and thus limit themselves to a set of six *universal* emotions. Those six emotions are: **joy/happiness, sadness, surprise, fear, disgust and anger**. These six emotions are considered as *universal* in the sense that any human individual will express these emotions the same way, whatever its cultural background and ethnic origin are [22]. To assess this universality property, Paul Ekman and Wallace Friesen (two of the most famous psychologists having studied the expression of emotions) have studied how humans were expressing a set of emotions. They conducted their experiments in a variety of countries and repeated the experience with people from different ethnicities and cultural backgrounds. Their conclusion was that *at least* six emotions were expressed the same way in all the cultures in which the experiments had been conducted (including

very retired villages in Papoua New Guinea!). It should be remembered that they did not claim that only six emotions were universal, but that they had been able to identify six that were. The reader interested in a more extensive taxonomy of emotions is invited to consult the description of the EARL Language [63], dedicated to the representation and annotation of emotions in technological contexts.

5.4 Bayesian Network Classifiers

In the scope of this thesis, we will only briefly present the theory supporting the Bayesian Network formalism. The reader interested in a more complete description of Bayesian networks, along with their most common applications, is invited to consult the following reference (in French) on the subject: [64]. A more pragmatic approach can be found in the following tutorials: [65] and [66]. Eventually, probably the most comprehensive survey about the basics of Bayesian Networks and the most used network structures can be found in [67].

A *Bayesian Network (BN)* (also known as Bayesian belief network or just belief network) is a probabilistic graphical model, which represents a set of variables together with a joint probability distribution containing explicit independence assumptions. More precisely, a Bayesian Network is a *directed acyclic graph* of

1. **Nodes** representing variables
2. **Arcs** representing probabilistic dependency relations among the variables and local probability distributions for each variable given the value of its parents

In addition to the graph structure, it is necessary to define the parameters of the model. For discrete variables, we can associate to each variable a *Conditional Probability Table (CPT)* that lists the probability that the child node takes each of its possible different values for each combination of values of its parents. When the variable is continuous, the two most common approaches are either to approximate its distribution by a gaussian distribution or to discretize the continuous distribution. Other possibilities include the use of another standard probability distribution, such as the Poisson or the Rayleigh distributions.

An arc from a node A to another node B implies that variable B depends directly on variable A. In this case A is called a parent of B. If for each variable X_i, $i = 1, \ldots, n$, the set of parent variables is denoted by $\textsc{Parents}(X_i)$, then the joint distribution of the variables is the product of the local distributions:

$$Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} Pr(X_i | \textsc{Parents}(X_i)) \tag{5.2}$$

This property demonstrates that Bayesian networks can represent a joint probability distribution in a very compact way. This compactness is an example of a very general property of *locally structured* systems. In a locally structured system, each subcomponent interacts directly with only a bounded number of other components, regardless of the total number of components, thereby leading to linear instead of exponential growth in complexity. To demonstrate the importance of this property, let us consider a network of n boolean variables, each being directly influence by at most k other variables, for some constant k. Each conditional probability table will then be specified by at most 2^k numbers, and the complete network can be specified by $n.2^k$ numbers. In contrast, the joint distribution contains 2^n numbers. For a network of $n = 30$ nodes, with $k = 5$, the Bayesian network will be specified by 960 numbers, while the joint probability distribution would require over a billion!

The most common structure of Bayesian network classifiers is the naïve bayesian network classifier, which is based on the assumption that all the observation variables are independent of each other. Although this assumption is unrealistic in most cases, it has been found to work remarkably well in practice, as explained in [68] and [69]. More quantitatively, an estimation of the classification error introduced by the independence assumption can be found in [70]. The key advantage of the naïve approach is twofold. First, the joint probability distribution reduces to the product of the individual probabilities:

$$Pr(X_1, \ldots, X_n) = \prod_{i=1}^{n} Pr(X_i) \tag{5.3}$$

which allows very fast inference mechanisms particularly well adapted to real-time applications. On the other hand, the discretized conditional

probability tables require much less samples in the learning set when fewer dependencies between variables are considered. For these reasons, naïve classifiers are often preferred to more complicated Bayesian networks, as the gain in complexity encompasses the loss in classification accuracy. In our particular case, given the size of the learning set, the choice is obvious: it is better to opt for the naïve approach as more complex network structures would result in poor statistical models (due to the reduced size of the learning set). Once efficient and reliable tracking techniques will be available, more complex network structures should however be investigated.

5.4.1 Naïve Bayesian Facial Expression Classification

In the case of the Naïve Bayesian Classifier, the goal is to estimate the class belonging y, from a set of observation data $\{X_i\}_{i=1,\ldots,n}$. If the class belonging is modeled as a stochastic variable Y, the classifier must compute:

$$y = \arg\max_{Y} Pr(Y|X_1, \ldots, X_n) \tag{5.4}$$

where $Pr(Y|X_1, \ldots, X_n)$ is computed using the Bayes' rule:

$$Pr(Y|X_1, \ldots, X_n) = \frac{Pr(X_1, \ldots, X_n|Y)Pr(Y)}{Pr(X_1, \ldots, X_n)} \tag{5.5}$$

As the denominator of equation 5.5 does not influence the value of y, it can be replaced by a numerical constant α. Taking into account the independence of the observation variables enables to express equation 5.5 as:

$$Pr(Y|X_1, \ldots, X_n) = \alpha \prod_{i=1}^{n} Pr(X_i|Y)Pr(Y) \tag{5.6}$$

$Pr(Y)$ can be fixed to $1/M$ where M is the total number of classes, if the probability of the object to belong to each class is supposed to be identical for all considered classes. It can also be equaled to the frequency of occurrence in the past history of events. For a particular class C, if on the last N_{tot} samples, N_C were belonging to class C, $Pr(Y = C)$ can be set to N_C/N_{tot}.

The problem of facial expression classification with naïve Bayesian classifiers thus consists merely in estimating the value of the individual conditional probabilities $Pr(X_i|Y)$. There are two different approaches to estimate these densities: either by assigning them values based on observations extracted from various psychological studies, or by learning their values through an artificial learning process[2]. As announced in the introduction of the present section, we decided to follow the latter approach, which consists in building statistical models of facial expressions from a large set of instances.

5.4.2 Learning Statistical Models of Expressive Faces

The learning process aims to build statistical models of expressive faces, using the set of distances defined by equations 5.1. In other words, we would like to analyze how the different facial expressions modify the value of the eight distances, with respect to their value in the neutral state.

The easiest approach would be to compute the mean and standard deviation of each distance for each of the considered facial expression. Once this is done, the probability density functions of the distances could be approximated by gaussian distributions, whose parameters have been computed from the training set. Unfortunately, the gaussian assumption does not hold in the case of facial expressions because an emotion can often be expressed with different expressions and intensities. An obvious example is the vertical mouth opening (D_8) when the *joy/happiness* emotion is showed: when a person smiles lightly, the lips are still pressed against each other, whereas they suddenly open widely when the intensity of the smile increases, letting the teeth appear. If half of the learning samples are taken when the person has still the lips closed and the other half when the person smiles largely, the mean value of the gaussian distribution would correspond to a vertical opening of the mouth located between the *closed lips* and the *open lips* states, and none of the smiles

[2]Actually, it is also possible to combine the two approaches in a variety of different ways. It is thus possible to consider both a priori information of experts and knowledge acquired from the learning process. The possibility to easily combine such heterogeneous sources of information is another of the advantages of Bayesian networks over other classification schemes

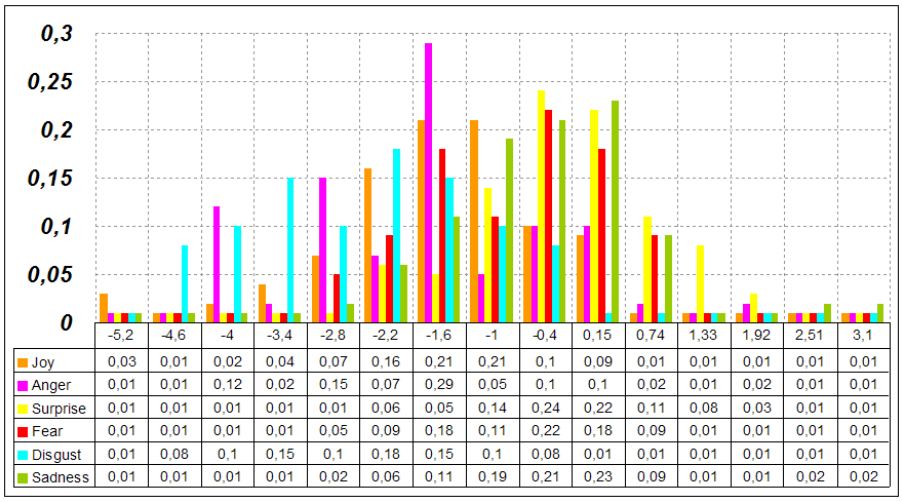
would be modeled accurately. However, it must be noticed that, in this particular case, an efficient modeling could have been achieved with a mixture of two gaussian distributions, one for the *closed lips smile* and one for the *open lips smile*. The same kind of reasoning prevails when the *fear* emotion is expressed : the mouth can either remain closed or open widely. Again, a gaussian distribution would fail to model this variety of shapes that could be encountered. For such reasons, instead of modeling with gaussian distributions, we have decided to estimate the true probability distributions by dividing the ranges of variations into a number K of intervals (*bins*) and to approximate the distributions by counting the number of occurrences N_k in each bin k with respect to the total number of occurrences N.

The number of bins K has been determined empirically to maximize the performances of the system with respect to the size of the training set (348 video sequences). Figures 5.3 to 5.10 illustrate the statistics that have been obtained with $K = 15$ (this value seems to be appropriate given the size of the training set). The experiments were also conducted with $K = 5$, $K = 10$ and $K = 20$, but the performances slightly decreased in those three other cases.

On these figures, the horizontal axis represents the difference of the input facial distances between the emotive and neutral state (as defined by equations 5.1, expressed relatively to the face bounding box of fixed dimension (see Figure 5.2). The vertical black line, drawn on each figure, indicates the point on the horizontal axis where no distance variation occurs (the emotion has provoked neither an increase nor a decrease of the underlying distance). The vertical axis corresponds to the probability that a new sample would fall into the considered bin. We have smoothed the results in the sense that we have assigned a probability of 1% (0,01) for bins that did not contain any occurences of the distance in the training set. As a matter of fact, when multiplying individual probabilities, a value of 0 would assign a zero-probability to the underlying emotion whereas it can happen that the corresponding value is plausible but had just not occured in the training set. In the following of this subsection, we will now successively examine the statistics obtained for each of the input facial distance.

The first component of the input vector is distance D_1, which mea-

sures the tension in the lower eyelids. Equations 5.1 define formally this distance. Intuitively, a decrease of D_1 reveals an increasing tension in the lower eyelids when the emotion is displayed. The first glance tells us that the curves corresponding to each emotion are quite interlaced with each other, which indicates that the discriminant factor of this feature will not be very important. The emotions that induce the largest changes in the tension in the lower eyelids are the *disgust* and *anger* emotions, which correspond in more than 90% of the cases to an increase of the lower eyelid tension. Also, it can be noticed that the statistics clearly show that most emotions actually increase the lower eyelid tension, with the exception of *sadness* and *surprise*.

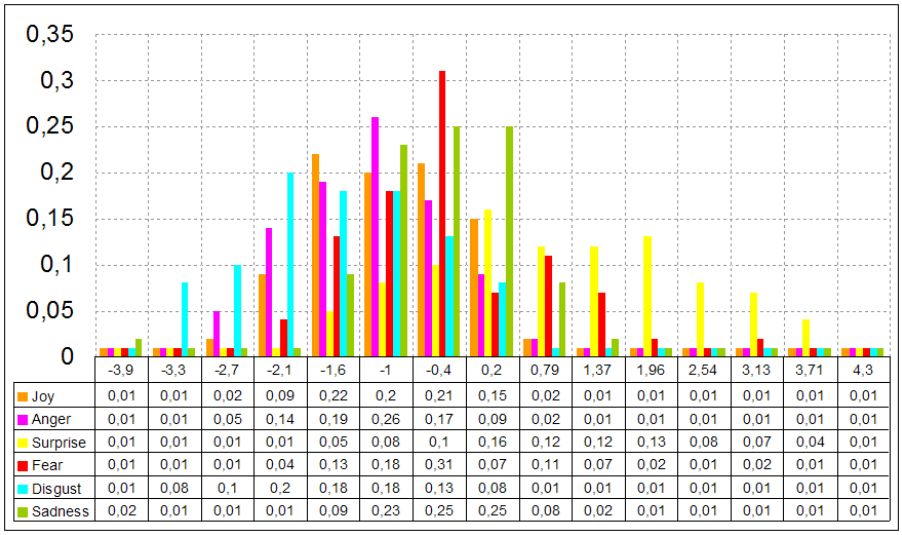


	-5,2	-4,6	-4	-3,4	-2,8	-2,2	-1,6	-1	-0,4	0,15	0,74	1,33	1,92	2,51	3,1
Joy	0,03	0,01	0,02	0,04	0,07	0,16	0,21	0,21	0,1	0,09	0,01	0,01	0,01	0,01	0,01
Anger	0,01	0,01	0,12	0,02	0,15	0,07	0,29	0,05	0,1	0,1	0,02	0,01	0,02	0,01	0,01
Surprise	0,01	0,01	0,01	0,01	0,01	0,06	0,05	0,14	0,24	0,22	0,11	0,08	0,03	0,01	0,01
Fear	0,01	0,01	0,01	0,01	0,05	0,09	0,18	0,11	0,22	0,18	0,09	0,01	0,01	0,01	0,01
Disgust	0,01	0,08	0,1	0,15	0,1	0,18	0,15	0,1	0,08	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,01	0,01	0,01	0,01	0,02	0,06	0,11	0,19	0,21	0,23	0,09	0,01	0,01	0,02	0,02

Figure 5.3: Learned statistics for facial distance D_1, which measures the tension in the lower eyelids. Negative values correspond to an increase of the tension (eyes closing)

The information brought by the second distance D_2 is somewhat redundant with the one provided by D_1, which is not surprising as movements of both upper and lower eyelids are highly correlated with each other. To illustrate this, we notice that *anger* and *disgust* also correspond to an increase in the tension in the upper eyelids. Moreover, the *joy/happiness* emotion also induces an important lowering of the upper

eyelids in most of the cases. On the other side of the axis, we notice a clear tendency of the emotion *surprise* to be accompanied by an important raise of the upper eyelid. Overall, the conclusion is the same as for the first examined distance D_1: most of the emotions seem to generate a small closing movement of the eyes. As a conclusion, the important fact that should be remembered from these two first figures is that the discriminative factor of both eye distances seems quite limited.

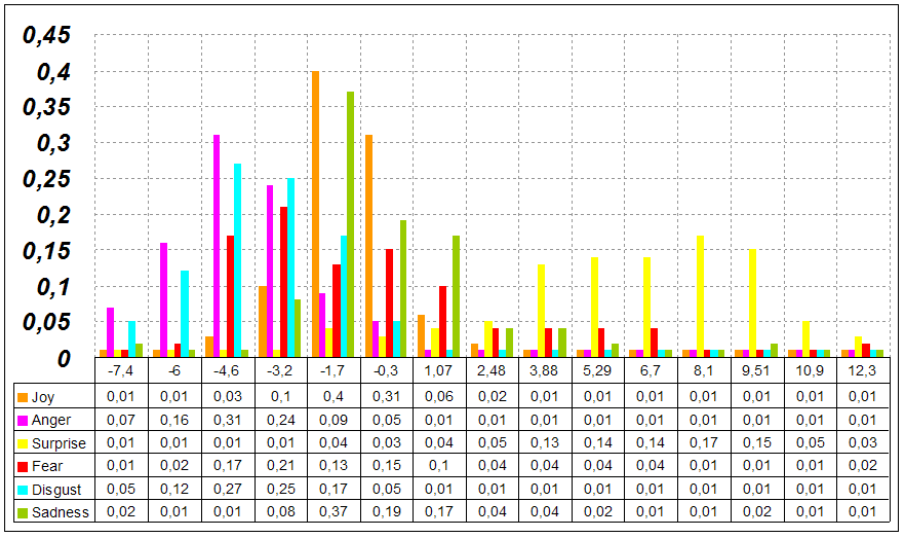


	-3,9	-3,3	-2,7	-2,1	-1,6	-1	-0,4	0,2	0,79	1,37	1,96	2,54	3,13	3,71	4,3
Joy	0,01	0,01	0,02	0,09	0,22	0,2	0,21	0,15	0,02	0,01	0,01	0,01	0,01	0,01	0,01
Anger	0,01	0,01	0,05	0,14	0,19	0,26	0,17	0,09	0,02	0,01	0,01	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,01	0,05	0,08	0,1	0,16	0,12	0,12	0,13	0,08	0,07	0,04	0,01
Fear	0,01	0,01	0,01	0,04	0,13	0,18	0,31	0,07	0,11	0,07	0,02	0,01	0,02	0,01	0,01
Disgust	0,01	0,08	0,1	0,2	0,18	0,18	0,13	0,08	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,02	0,01	0,01	0,01	0,09	0,23	0,25	0,25	0,08	0,02	0,01	0,01	0,01	0,01	0,01

Figure 5.4: Learned statistics for facial distance D_2, which measures the tension in the upper eyelid. Positive values correspond to an opening of the eye / a raise of the upper eyelids.

The third considered distance D_3 measures the vertical distance between the center of the eyebrow and the center of the eye. Positive values correspond to a raise of the center of the eyebrows. This figure brings more information than the two previous ones, in the sense that the behavior of the center of the eyebrows may be classified in three distinct classes: *fear, disgust* and *anger* induce a visible lowering of the eyebrow centers while *surprise* corresponds to an important raise of the eyebrow centers. Two emotions (*joy* and *sadness*) do not lead to any significant change of D_3, although a small lowering might be observed

in most cases.

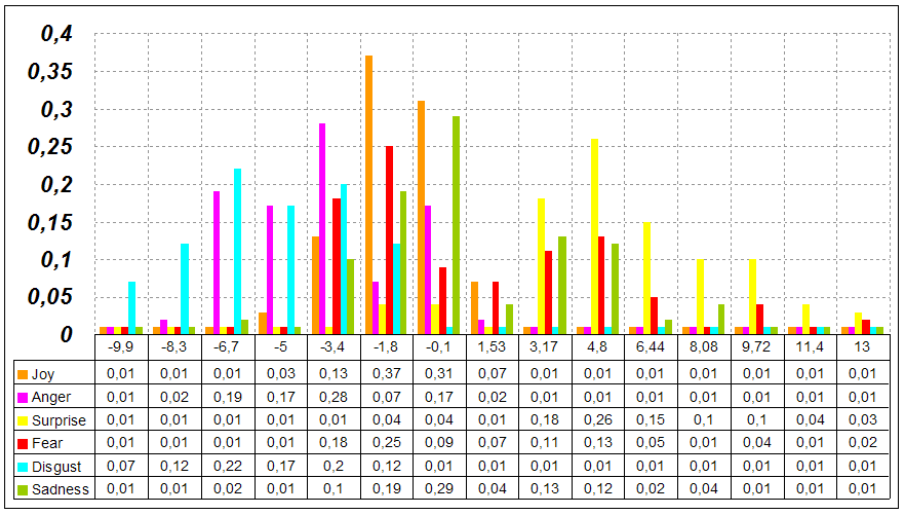


	-7,4	-6	-4,6	-3,2	-1,7	-0,3	1,07	2,48	3,88	5,29	6,7	8,1	9,51	10,9	12,3
Joy	0,01	0,01	0,03	0,1	0,4	0,31	0,06	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Anger	0,07	0,16	0,31	0,24	0,09	0,05	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,01	0,04	0,03	0,04	0,05	0,13	0,14	0,14	0,17	0,15	0,05	0,03
Fear	0,01	0,02	0,17	0,21	0,13	0,15	0,1	0,04	0,04	0,04	0,04	0,01	0,01	0,01	0,02
Disgust	0,05	0,12	0,27	0,25	0,17	0,05	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,02	0,01	0,01	0,08	0,37	0,19	0,17	0,04	0,04	0,02	0,01	0,01	0,02	0,01	0,01

Figure 5.5: Learned statistics for facial distance D_3, which measures the distance between the center of the eyebrow and the center of the eye. Positive values correspond to a raise of the center of the eyebrows.

The fourth considered input vector's element D_4 measures the vertical distance between the center of the eye and the outer eyebrow point, positive values corresponding to a raise of the outer eyebrow points. The observation of Figure 5.6 learns us that *surprise* induces an important raise of the outer eyebrow points, while *anger* and *disgust* clearly show a decrease of the vertical distance between the outer eyebrow points and the eye centers. Again, for the other emotions (*joy*,*fear* and *sadness*), the variations is less obvious, even though a slight lowering of the outer eyebrow points may often be noticed.

The fifth distance to be considered, D_5, measures the vertical distance between the inner part of the eyebrows and the center of the eyes, positive values corresponding to an increase of this distance (a raise of the inner part of the eyebrows). This distance also shows very interlaced curves. One emotion clearly generates a raise of the inner eyebrows: the *surprise* emotion. No other emotion clearly distinguishes itself from the



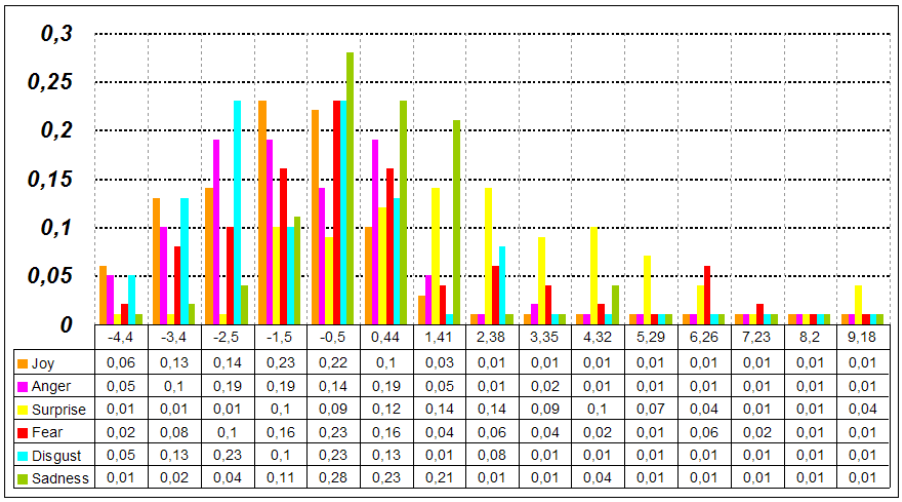
	-9,9	-8,3	-6,7	-5	-3,4	-1,8	-0,1	1,53	3,17	4,8	6,44	8,08	9,72	11,4	13
Joy	0,01	0,01	0,01	0,03	0,13	0,37	0,31	0,07	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Anger	0,01	0,02	0,19	0,17	0,28	0,07	0,17	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,01	0,01	0,04	0,04	0,01	0,18	0,26	0,15	0,1	0,1	0,04	0,03
Fear	0,01	0,01	0,01	0,01	0,18	0,25	0,09	0,07	0,11	0,13	0,05	0,01	0,04	0,01	0,02
Disgust	0,07	0,12	0,22	0,17	0,2	0,12	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,01	0,01	0,02	0,01	0,1	0,19	0,29	0,04	0,13	0,12	0,02	0,04	0,01	0,01	0,01

Figure 5.6: Learned statistics for facial distance D_4, which measures the vertical distance between the center of the eye and the outer eyebrow point. Positive values correspond to a raise of the outer eyebrow points.

others. Nevertheless, distance D_5 remains useful when used in conjunction with other distances.

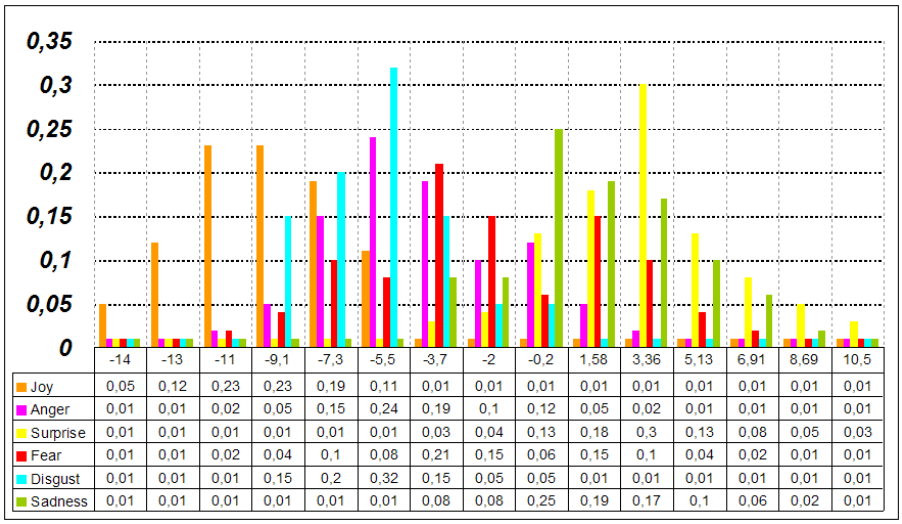
D_6 measures the vertical distance that separates mouth corners from eye corners, positive values representing a decrease of the distance (a raise of mouth corners, as eye corners are considered as fixed points). Intuitively, we are tempted to say that this feature will allow to discriminate very well *sadness* (lowering of mouth corners) from *joy/happiness* (raise of mouth corners). The observation of Figure 5.8 reveals that *joy/happiness* is indeed very well discriminated by this feature. Conversely, *sadness* is often confused with *surprise* or *fear* as these three emotions correspond to a lowering of mouth corners. Similarly, *disgust* and *anger* are often confused as both of these emotions are associated with a small raise of the mouth corners.

Distance D_7 measures the horizontal opening of the mouth, that is: the horizontal distance between both mouth corners, a positive value meaning that this distance increases when the emotion is expressed. The observation of Figure 5.9 tells us that this feature is very discriminant.



	-4,4	-3,4	-2,5	-1,5	-0,5	0,44	1,41	2,38	3,35	4,32	5,29	6,26	7,23	8,2	9,18
Joy	0,06	0,13	0,14	0,23	0,22	0,1	0,03	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Anger	0,05	0,1	0,19	0,19	0,14	0,19	0,05	0,01	0,02	0,01	0,01	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,1	0,09	0,12	0,14	0,14	0,09	0,1	0,07	0,04	0,01	0,01	0,04
Fear	0,02	0,08	0,1	0,16	0,23	0,16	0,04	0,06	0,04	0,02	0,01	0,06	0,02	0,01	0,01
Disgust	0,05	0,13	0,23	0,1	0,23	0,13	0,01	0,08	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,01	0,02	0,04	0,11	0,28	0,23	0,21	0,01	0,01	0,04	0,01	0,01	0,01	0,01	0,01

Figure 5.7: Learned statistics for facial distance D_5, which measures the vertical distance between the inner part of the eyebrows and the center of the eyes, positive values corresponding to an increase of this distance (a raise of the inner part of the eyebrows).

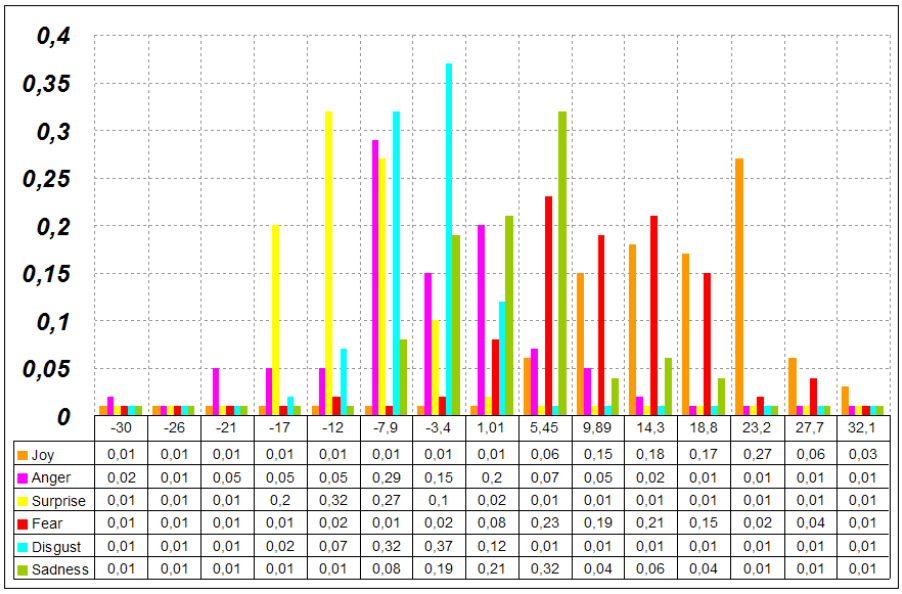


	-14	-13	-11	-9,1	-7,3	-5,5	-3,7	-2	-0,2	1,58	3,36	5,13	6,91	8,69	10,5
Joy	0,05	0,12	0,23	0,23	0,19	0,11	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Anger	0,01	0,01	0,02	0,05	0,15	0,24	0,19	0,1	0,12	0,05	0,02	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,01	0,01	0,01	0,03	0,04	0,13	0,18	0,3	0,13	0,08	0,05	0,03
Fear	0,01	0,01	0,02	0,04	0,1	0,08	0,21	0,15	0,06	0,15	0,1	0,04	0,02	0,01	0,01
Disgust	0,01	0,01	0,01	0,15	0,2	0,32	0,15	0,05	0,05	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,01	0,01	0,01	0,01	0,01	0,01	0,08	0,08	0,25	0,19	0,17	0,1	0,06	0,02	0,01

Figure 5.8: Learned statistics for facial distance D_6, which measures the vertical distance that separates mouth from eye corners, positive values representing a decrease of the distance (a raise of mouth corners)

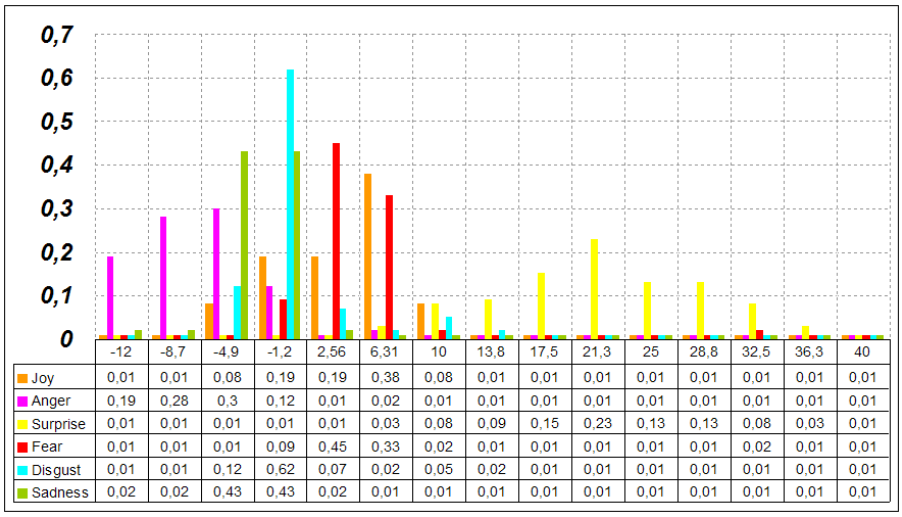
An important decrease is undoubtedly a sign of *surprise*, while an important increase can only correspond to *joy/happiness* or *fear*. When a slight decrease of the distance occurs, there's a high probability that either *anger* or *disgust* is being experienced, whereas a slight increase often reveals *sadness*.

Last but not least, the vertical opening of the mouth reveals much affective information. Distance D_8 will be positive when the mouth has a tendency to open while expressing the emotion. Negative values reveal a tendency of the lips to be pressed against each other. This is probably one of the most discriminant features for the emotions that are considered in the scope of this thesis. We see by inspecting Figure 5.10 that *surprise* and *anger* can be detected almost without ambiguity only by analyzing the value of D_8. Moreover, slight increases of D_8 are strongly associated with *fear* and *joy/happiness*, while slight decreases correspond either to *disgust* or *sadness*.



	-30	-26	-21	-17	-12	-7,9	-3,4	1,01	5,45	9,89	14,3	18,8	23,2	27,7	32,1
Joy	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,06	0,15	0,18	0,17	0,27	0,06	0,03
Anger	0,02	0,01	0,05	0,05	0,05	0,29	0,15	0,2	0,07	0,05	0,02	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,2	0,32	0,27	0,1	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Fear	0,01	0,01	0,01	0,01	0,02	0,01	0,02	0,08	0,23	0,19	0,21	0,15	0,02	0,04	0,01
Disgust	0,01	0,01	0,01	0,02	0,07	0,32	0,37	0,12	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,01	0,01	0,01	0,01	0,01	0,08	0,19	0,21	0,32	0,04	0,06	0,04	0,01	0,01	0,01

Figure 5.9: Learned statistics for facial distance D_7, which measures the horizontal opening of the mouth, that is: the horizontal distance between both mouth corners, a positive value meaning that this distance increases when the emotion is expressed.



	-12	-8,7	-4,9	-1,2	2,56	6,31	10	13,8	17,5	21,3	25	28,8	32,5	36,3	40
Joy	0,01	0,01	0,08	0,19	0,19	0,38	0,08	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Anger	0,19	0,28	0,3	0,12	0,01	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Surprise	0,01	0,01	0,01	0,01	0,01	0,03	0,08	0,09	0,15	0,23	0,13	0,13	0,08	0,03	0,01
Fear	0,01	0,01	0,01	0,09	0,45	0,33	0,02	0,01	0,01	0,01	0,01	0,01	0,02	0,01	0,01
Disgust	0,01	0,01	0,12	0,62	0,07	0,02	0,05	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01
Sadness	0,02	0,02	0,43	0,43	0,02	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01

Figure 5.10: Learned statistics for facial distance D_8, which measures the vertical opening of the mouth, with positive values occurring when the mouth opens while expressing the emotion.

5.4.3 Classification Results

In section 5.4.1, we introduced the Naïve Bayesian Classifier (NBC) and justified our choice to adopt the assumption that observation variables should be considered independent of each other. Equation 5.6 summarizes the behavior of the NBC. If we denote the emotion to be classified by E and the set of observation variables by $\{D_i\}_{i=1,\ldots,8}$, the equation becomes:

$$Pr(E|D_1,\ldots,D_8) = \alpha \prod_{i=1}^{8} Pr(D_i|E)Pr(E) \tag{5.7}$$

An interesting modification of this equation would be to replace the constant α, which is used only for normalization purpose, by a set of constants α_i (each of them being relative to one of the facial distances), which would enable each distance to have an influence on the final decision proportionnal to its discriminant factor. In this case, the equation governing the classification task would become:

$$Pr(E|D_1, \ldots, D_8) = \prod_{i=1}^{8} \alpha_i Pr(D_i|E) Pr(E) \tag{5.8}$$

Our goal is to be able to detect a set of six emotions by analyzing a set of observation variables, which are facial distances in the present case. We will therefore choose the value of E that maximizes the probability that E is detected, after having examined the value of all observation variables. The decision criterion is therefore:

$$e = \arg\max_{E} Pr(E|D_1 = d_1, \ldots, D_8 = d_8) \tag{5.9}$$

where d_i represents the value observed for variable D_i and e is the detected facial expression.

Among the three terms of the right member of equation 5.7, the term α is a normalization constant, which is adjusted so that the sum of the probabilities corresponding to the different emotions equals 1, whereas $Pr(E)$ is the *a priori* probability that a specific emotion E has to be detected. To estimate $Pr(E)$, we considered the proportion of the video samples showing emotion E with respect to the total number of video samples $N = 348$. Table 5.3 contains the number of samples corresponding to each possible value of E. To illustrate by an example, the *a priori* probability that emotion ANGER is detected is equal to $Pr(E = \text{ANGER}) = 39/346) = 11,27\%$.

Table 5.3: Number of video samples per emotion

Joy	93	Surprise	75	Disgust	37
Anger	39	Fear	52	Sadness	50

Finally, the last term to consider in equation 5.7 is the product of the individual probabilities. Those individual probabilities have been estimated from the training set and are depicted in Figures 5.3 to 5.10.

When a new value d_i is observed for a variable D_i, the algorithm first determines to which bin d_i should be assigned, based on its value.

In a second step, the probabilities $Pr(E|D_i = d_i)$ are computed for each possible value of E. Finally, once all the observation variables have known values, the individual probabilities are multiplied to give a score to each emotion, proportional to the probability that this emotion has been recognized.

The classification has been performed using a *leave-one-block-out cross validation*, with varying block sizes. In short, this protocol consists in using all the available video samples but a block of size BS to do the learning process. The BS samples that have not been used for the learning process are then used by the detection algorithm. The *cross validation* term means that we repeated the process iteratively, taking successive blocks of size BS as testing set, leaving the rest for the learning set, until the entirety of the samples had been used for detection. A reasonable block size was $BS = 3$. Indeed, using a block size of 3 enables to use 345 out of the 348 samples for the learning process (which produces almost the same result as if the maximum of 347 samples had been used for the learning) while speeding up the learning task by a factor of 3.

Table 5.4 depicts the confusion matrix obtained when considering the 8 facial distances presented above.

Table 5.4: Classification results: naïve Bayesian classifier with all the distances included in the observation model. Overall recognition rate: **81.3%**

	Joy	Ang.	Surp.	Fear	Disg.	Sadn.	Rate
Joy	**86**	1	0	3	0	3	**92.5**
Anger	0	**23**	0	1	9	6	**59.0**
Surprise	0	0	**71**	1	0	3	**94.7**
Fear	8	1	7	**32**	1	3	**61.5**
Disgust	3	7	0	0	**27**	0	**75.7**
Sadness	2	1	1	3	1	**42**	**84.0**

We see by inspecting Table 5.4 that the emotions *surprise* and *joy*

have the highest recognition rate. This is not very surprising as these two emotions are associated with very discriminant values of mouth distances. As a matter of fact, *joy* can only be confused with *fear* when watching the horizontal opening of the mouth, but if we combine this information with the measure of the vertical evolution of the mouth corners, *joy* can be detected without ambiguity. The same reasoning can be applied to the recognition of the *surprise* emotion: the observation of the vertical opening of the mouth provides enough information to detect *surprise* with a very low probability of error. *Sadness* is also associated with a recognition rate above the average, although none of the distances exhibit a very distinctive behavior when the *sadness* emotion is experienced. In this case, it is rather the combination of the information brought by the individual distances that justifies the high recognition rate: the *sadness* emotion is often associated with quite narrow distributions (the probability distributions of the distances given that *sadness* has been recognized have low variance values). For this reason, taking into account a variety of distances enables the system to detect *sadness* efficiently. We can also notice that *fear* is often confused with *surprise*, which is not surprising given the fact that human observers themselves often tend to confuse these two emotions. Moreover, these two emotions are often felt simultaneously, which gives further justifications to the lower recognition rates associated with these two emotions.

More stunning is the confusion between *fear* and *joy*, which is more difficult to explain, as both emotions are quite different from each others. The cause might be that genuine *fear* is quite difficult to elicit in an experimental environment. So it might be that subjects asked to express the *fear* emotion actually expressed something similar to a 'contorted smile'. Finally, we can observe that *disgust* and *anger* are very much confused with each other. It is interesting to observe that both emotions might be considered similar in the sense that both are negative emotions involving a strong feeling of rejection, which is expressed on the face as the facial features 'closing' to reject the stimuli (lips are tensed, eyes and eyebrows are contracted).

The recognition rate that was achieved in the presented system depends strongly on the underlying features: mouth conveys much more discriminative information than eyes and eyebrows, which appears clearly

when observing the corresponding distances' probability distributions. The question that could therefore be asked at that point is: would not there be a subset of the distances that would lead to a higher recognition rate? In other terms, the observation of the probability density functions leads us to wonder whether or not a better input vector, obtained by taking only a subset of the facial distances, could be found. If it is the case, what would be the best input vector? Two widely used techniques can be used to answer such questions: *forward selection* and *backward elimination*. Forward selection consists in measuring the performances of the system when only one single observation variable is used. The variable leading to the best recognition rate is stored in an empty input vector. Future iterations proceed similarly with the remaining variables, until the entire set of features have been added in the input vector. This way, we end up with an ordered list of the variables, ranked from the most to the least discriminative. Backward elimination proceeds exactly the opposite way: we start the process by removing each observation variable one-by-one from the entire feature set, and place in the empty input vector the variable that degraded the least the recognition rate. Future iterations proceed similarly by removing one-by-one each feature from the remaining set until the input vector contains all the feature variables. Again, we end up with an ordered list of the observation variables, ranked this time from the least to the most discriminative.

Applying forward selection and backward elimination on our facial distances led to the following observation: the two eye distances were in both cases considered as the least discriminative distances. To assess the pertinence of this observation, we decided to perform the same NBC classification, but with an input vector containing only the distances relative to the mouth and eyebrows. The observation vector $\mathbf{X}$ is thus in this case:

$$\mathbf{X} = [D_3, D_4, \ldots, D_8] \tag{5.10}$$

The results that were achieved with this modified set of observation variables are depicted in Table 5.5.

The results presented in Table 5.5 enable to conclude that considering eye distances actually ***decreases*** the performances of the system,

Table 5.5: Classification results: naïve Bayesian classifier with only the distances relative to the mouth and eyebrows. Overall recognition rate: **83.8%**

	Joy	Ang.	Surp.	Fear	Disg.	Sadn.	Rate
Joy	**88**	0	0	1	4	0	**94.6**
Anger	0	**29**	0	1	7	2	**74.4**
Surprise	1	0	**71**	1	0	3	**93.3**
Fear	9	1	2	**32**	2	4	**65.4**
Disgust	3	5	0	0	**27**	0	**78.4**
Sadness	2	4	1	2	1	**42**	**80.0**

except for the detection of the *sadness* emotion. Of course, it is also obvious that eye distance variations are much more subtle to distinguish than mouth or eyebrow distance variations. For this reason, it might be dangerous to assess the fact that distances related to the eyes do not convey as much affective information as the other facial features, because the performances might be degraded mainly due to imprecisions in the labeling process and to the introduction of a quantification error that might be important when quantifying small figures (subtle variations).

To have an estimation of the confidence interval of the given recognition rates, we measured the variation in performances that was observed when the parameters of the classification task were changed. More precisely, we measured the recognition rate for $N = 10$ and $N = 15$, with the following block sizes: $BS = 2$, $BS = 3$, $BS = 4$, $BS = 6$ and $BS = 12$. We obtained 10 different recognition rates, ranging from 77.21% to 82.69%, the average recognition rate being 79.91%. Although it is a very rough approximation, we can have an idea of the size of the confidence interval, by considering that these different rates are distributed according to a gaussian distribution of mean $\mu = 79.91\%$ and standard deviation $\sigma = 1.69\%$. The 95% confidence interval would, in this case, be approximated by:

$$I(0, 95) = [\mu - 2\sigma; \mu + 2\sigma] = [76.52\% - 83.30\%] \tag{5.11}$$

Again, we insist on the fact that this confidence interval should only considered as a rough estimation. Its purpose is to estimate the variation in performances that can be observed when the parameters of the classification task vary.

5.5 Support Vector Machines

Support Vector Machines have recently been used for a variety of pattern recognition applications including handwritten digit recognition [72], face detection [73], speaker identification [74] and text categorization[75]. In most of these cases, SVM generalization performance (i.e. error rates on test sets) either matches or is significantly better than competing techniques.

In recent years, attempts have been made to use Support Vector Machines for facial expression classification. The results obtained were among the best ever achieved, suggesting that SVM is indeed a very appropriate approach for facial expression classification [61] [62]. For these reasons, we wanted to test classifiers based on SVM with our dataset, in order to be able to compare their performances with those that were achieved with Bayesian networks.

In this section, we will develop and comment the results that were achieved with Support Vector Machines (SVMs). In appendix, the interested reader will find the main lines of the theory related to these types of statistical classifiers. This appendix was realized by synthesizing information from the following excellent references: [76], [77], [78], [79] and [80]. We decided to include these developments inside this thesis because we believe that their understanding is fundamental for whoever is interested in the results presented here.

5.5.1 Classification Results

In this section, we present the classification performances obtained with Soft-Margin Multi-Class Support Vector Machines. To guarantee an unbiased comparison with the results obtained by the Bayesian classifier presented in the preceding section, we used exactly the same data set and cross validation protocol.

As mentioned earlier in this chapter, the data set consists of 348 samples, corresponding to images of six different emotions showed at their maximal intensity. We conducted the experiments with a leave-one-block-out cross validation protocol, with a block size of 3 samples. This means that out of the 348 samples contained in the entire data set, 345 were used in model learning while the 3 remaining ones were used for testing. The experiment was conducted 116 times, so that the entire data set has been used as test set.

We used the LIBSVM implementation of the Support Vector Machines algorithms. Among the different available implementations contained in the LIBSVM Library, we chose the classical multi-class C-SVC implementation, with a variety of 7 different kernels.

To avoid the introduction of numerical issues due to large values contained in the training set, we scaled the input data so that each feature has a value range of $[-1, \ldots, +1]$. This was performed by the built-in routine of the LIBSVM implementation, called `svcscale`.

We tried to tune the parameters of the model (only γ and C) to observe the variation of the performances that could be induced. Although slight variations were indeed observed, we decided to present only the results that have been achieved with the default parameters of the model. In other words, the experiments were conducted with the following values:

$$\gamma = 1 \tag{5.12}$$

$$b = 0 \tag{5.13}$$

$$C = 1 \tag{5.14}$$

As mentioned earlier, 7 kernels were tested: the RBF kernel, the sigmoïdal kernel and polynomial kernels of different degrees. These kernels are defined by the following equations:

$$k_{RBF}(\mathbf{x}, \mathbf{y}) = \exp(-\gamma \langle \mathbf{x}, \mathbf{y} \rangle) \tag{5.15}$$

$$k_{sig}(\mathbf{x}, \mathbf{y}) = \tanh(\gamma \langle \mathbf{x}, \mathbf{y} \rangle + b) \tag{5.16}$$

$$k_{poly}(\mathbf{x}, \mathbf{y}) = (\gamma \langle \mathbf{x}, \mathbf{y} \rangle + b)^d \tag{5.17}$$

If we replace in these equations the parameters γ and C by the values that we decided to adopt, the definitions of the kernels corresponding to the results presented in this section are reduced to the following simpler forms:

$$k_{RBF}(\mathbf{x}, \mathbf{y}) = \exp(- \langle \mathbf{x}, \mathbf{y} \rangle) \tag{5.18}$$

$$k_{sig}(\mathbf{x}, \mathbf{y}) = \tanh(- \langle \mathbf{x}, \mathbf{y} \rangle) \tag{5.19}$$

$$k_{poly}(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle)^d \tag{5.20}$$

where d corresponds to the degree of the polynomial kernel.

Before presenting our results, we must agree on the fact that the choice of kernels presented above is absolutely not exhaustive. One of the most difficult aspects of manipulating SVM algorithms is to find the most appropriate kernel and the values of the parameters that maximize the performances. In the scope of this thesis, however, the focus has not been set on this aspect, and only the most popular kernels, with default parameter values have been tested. The idea behind this choice is that we do not put too much emphasis on the exact recognition rates that are presented, as those would mainly vary according to the data and the features that are being used. Our goal is to have a glimpse of the performances that can be achieved with a SVM classifier, in comparison to the performances achieved with a Naïve Bayesian classifier.

Table 5.6 presents the results that were achieved with the kernel defined by equations (5.18) to (5.20).

Compared to the results that we presented in Table 5.4, we see that classifiers based on SVMs outperform the naïve Bayesian classifier (we achieve a recognition rate of 89% with a SVM classifier instead of 81% with a Bayesian network classifier). Moreover, this higher recognition rate is achieved with five out of the seven kernels that were tested, with default parameter values. Higher recognition rates should therefore be reachable if finer tuning of the kernel parameters would be performed.

Table 5.6: Classification performances, achieved with a SVM classifier

Kernel	Degree	Recognition Rate
k_{RBF}	-	**89.91**%
k_{poly}	1	89.34%
k_{poly}	2	89.34%
k_{poly}	3	89.05%
k_{poly}	4	84.73%
k_{poly}	5	80.40%
k_{SIG}	-	78.39%

The author believes that this result should be taken with caution as it might be the case that Bayesian network classifiers would produce much better performances if the training set was subsequently larger (the probability density estimations that were computed from the database show irregularities, partly due to the limited size of the training set). This, of course, should not conflict with the observation that SVM classification seems to be a very effective approach to the problem of facial expression classification.

Finally, we would like to insist on the fact that the results presented in this section should be considered as a first trial, in other words as an invitation to dig further the use of SVM for facial expression classification. Indeed, the results are promising, even when standard parameters are used.

5.6 Integration of Facial Expressions in Mixed Reality Interactive Storytelling

In section 2.5, we have explained how the virtual characters could use their emotional state to choose between several possible solutions in order to achieve their goal. This type of interactive storytelling, in which the characters choose their actions according to their own internal emotional state was called *affect-driven interactive storytelling*.

In this section, we present an immersive interactive storytelling in

which facial expressions play a central role in the story unfolding process. This prototype should be considered as future work and therefore constitutes an illustration of the use of facial expressions as a way to enhance the entertainment of an immersive interactive application.

5.6.1 WeatherBoy: A First Scenario

One of the most difficult issues when creating such a novel type of application is to find a way to integrate the technology so as to enhance the entertaining aspects of the application. In the case of a scenario that unfolds according to facial expressions, we need to find a mapping between the real and the virtual world: facial expressions on the user's face (the real world) should generate entertaining events in the virtual world. In our application, we have chosen to associate facial expressions with the control of the weather. The player (ideally a child aged from six to twelve years old) is embodied into an avatar who has special powers: he can control the weather with his facial expressions. Let us thus call him *WeatherBoy*

So far, we have provided a mapping for four different emotions:

- *Joy* generates *sunny weather*
- *Anger* provokes *thunderstorms*
- *Sadness* brings *rain*
- *Surprise* makes the *wind* blows

The scenario is based on traditional fairy tales. Our hero, WeatherBoy, has to deliver a princess, who has been imprisoned in a dungeon by a powerful dragon. WeatherBoy will meet several people on his quest who will ask him for a specific weather: a farmer who needs rain for his field, a lord who would really like to have a sunny weather for his daughter's wedding, a sailor's wife who needs the wind to push her husband back to the shore, etc... At each of these encounters, WeatherBoy may decide whether or not he will help the people he encounters by displaying the corresponding facial expressions.

When the user is asked to express an emotion, it could be possible that the facial expression recognition system would output an uncertain

estimation. When using Bayesian Networks, this would be the case when the maximum of the *a posteriori* probabilities of the detection is low whereas in the case of the SVM classifiers, the case would be revealed by a low distance between the corresponding input sample and the separating hyperplane. In our application, we could imagine that in these cases, the weather would only partially correspond to the requested facial expression (for instance, some clouds would still be present in the sky when smile has been recognized, but with a high probability of missclassification). This would be a mean to ask the user to express more intensely or more clearly its emotion.

From the virtual character's point of view, the recognized facial expressions serve to determine whether or not their wishes have been fulfilled by the user. For instance, the farmer who needs rain to irrigate his field will have the terminal action HAVE RAIN succeeded if the user clearly shows a sad face, thereby making the weather becomes rainy. In other words, the story is well and truly impacted by the facial expressions of the user.

To win the game and deliver the princess, WeatherBoy will have to collect money from the people he helped (the lord and the farmer) and buy the magic sword with the money he earned. If he has also helped the sailor's wife, he will be told where the magic sword can be purchased. With his magic sword, WeatherBoy can then go to the dragon's den and invoke thunder to empower his sword and kill the evil dragon with a magic lightning. Such a scenario obviously invite its conceptor to also allow the use of gestures, to enable WeatherBoy to travel the virtual world or to handle the sword when facing the dragon. We thus are in presence of a multimodal interactive storytelling application, whose face and body gestures are the two main modalities[3].

On a hardware's perspective, it is necessary to opt for a two-camera system when considering a full-scale prototype. A first camera must locate and track the user silhouette as well as some body's important features such as the positions of the hands and the face. A second camera, mounted on a turret, tracks the user's face to provide a high-resolution face image to the facial recognition module.

[3]It is of course possible to add a verbal modality to further improve the application, but this is out of the scope of this section

Finally, concerning the entertaining aspects of this prototype, we know that an application must attract the attention of its user to be engaging. To be concise, it must fascinate and entertain. In other words, it must generate emotions. The interesting aspect of integrating emotions in an interactive storytelling application is that it would considerably increase the feeling of immersion. A system that can understand non-verbal communication can be very impressive for an adult, and totally magic for a child...

5.7 Conclusion

We started this chapter by justifying our choice to consider a set of facial distances as the way to represent the affective information conveyed by the face. We then focused on a set of 6 facial expressions that we aim to detect and discussed our decision to follow the trend by considering Ekman's taxonomy of emotions. We then justified the use of probabilistic classifiers, such as Bayesian networks and Support Vector Machines, as the best way to deal with the uncertainty associated with the process of modeling emotions. We concluded by briefly presenting the protocol that we used to extract meaningful data from our facial expression database.

After this introductory part, we went over the basic principles that support our definition of Bayesian network classifiers and explained our choice to consider only naïve Bayesian networks, in which all observation variables are considered to be independent from each other. Afterwards, we presented in detail the Bayesian learning algorithm that was implemented and concluded the section by commenting each of the statistical models built with our learning algorithm.

We presented the classification results achieved with the naïve Bayesian network classifier and discussed the detection performances of each individual facial expression that we aimed to recognize. We concluded our analysis by exploring feature selection/elimination techniques, as a way to derive a subset of facial distances that led to an increase of the recognition rate.

In order to benchmark the performances of our naïve Bayesian classifier, we developed and tested several classifiers, using Support Vector

Machines (SVM). We presented the classification results achieved with different kernel functions and commented the performances of the SVM classifiers. The conclusion is that the SVM classifiers outperforms the naïve Bayesian classifier, but that this result should be taken with caution, given the limited size of the training set.

Finally, we conclude the chapter by considering the use of facial expressions as a new modality to interact within an interactive storytelling application. We show how the recognition of facial expressions could be associated with events in a virtual world and how the occurence of these events may modify the course of the story. We briefly investigated the hardware requirements associated with the use of this new modality and discussed techniques to cope with the errors of the facial expression recognition module.

Chapter 6

Conclusion

The goal of this thesis was to demonstrate how interactive applications such as interactive storytelling or immersive video games could be interfaced with their user in a way that is similar to the way human beings communicate with each other. Within this scope, we investigated the problem of automatic recognition of face and body gestures, so as to provide an anthropomorphic interface enabling the occurrence of *natural* interactions between the user and the interactive storytelling application.

6.1 Definition of Objectives

We started our discussion by explaining why current human-machine interfaces are becoming obsolete for a growing number of interactive applications. Over the past decades, machine design has driven the way users should interact with the applications. As an example, today's most video games are still being played with a control pad, which requires the user to become familiar with its handling before being able to feel engaged in the application. We argue that efforts should be made to consider human-machine interfaces the other way around: the machines should adapt to their users, which are human beings[1]. It is within this

[1]During the redaction of this thesis (on December 8th, 2006), one of the most important manufacturers of video gaming consoles, released a revolutionary console, whose control pad recognizes the gestures of its user, and these gestures are used to

context that this thesis has been started: with the aim to explore so-called *natural interfaces* and their integration in interactive applications.

In the introduction, we explored the different modalities that are used by humans to communicate with each other. We argued that spoken language is far from being the only channel used by humans to express themselves, and demonstrated that a great part of the communication is indeed non-verbal, using communication channels such as voice intonation (prosody), facial expressions, body postures and gestures.

In the scope of this thesis, we chose to restrict ourselves to non-verbal modalities, and more precisely to focus on two of them: face and body gestures. Moreover, we wanted to demonstrate that face and body gestures can be used to drive the scenario of an interactive application.

6.2 Summary of Main Achievements

6.2.1 The Gesture Recognition Module

With these goals in mind, we started to conceive a simple but robust gesture recognition system, built on top of a body segmentation engine (the Salto™ system, developed by Alterface SA). In the prototype that was implemented, Salto™ provided the coordinates of five crucial points of the user's silhouette. The developed system analyzes in real time these coordinates and is able to detect a set of five different gestures. In addition, it also computes additional informations such as an estimation of the distance between the user and the video camera or the direction of pointing when a pointing gesture is detected.

As we discussed in the third chapter, we do not pretend to have really investigated deeply the problem of gesture recognition in this work. Nevertheless, the developed system was quite robust and provided a very good illustration for a first functional prototype. Its main limitations are that it is hardly scalable to handle either a larger set of gestures or gestures whose detection rules are not easy to establish. To overcome these limitations, there exist better ways to process an input data stream with the goal of recognizing temporal patterns. More specifically,

control the games. The release of this new console has a huge commercial success, which seems to be mainly due to its revolutionary interface...

a technique such as Dynamic Bayesian Networks (DBN) can be used to learn the relationships between the patterns to be detected and the observation data. It should therefore be possible to overcome some of the limitations of the existing system, as the modeling task could become much more efficient by using DBNs. As we will discuss later, it is this type of approach that has been chosen to recognize facial expressions.

6.2.2 The Online Planning Engine

Once the gesture recognition system had been developed, we added a TCP/IP module so that detected gestures could be transmitted to another module that would be responsible to react appropriately to the detected gestures. The goal was to design an online planning system, which could generate in real time a storyline integrating the detected gestures. This planning engine thus had to drive the narrative according to the interactions the user would have with the application, while keeping interactivity and entertaining aspects intact! We based our solution on the approach developed at the University of Teesside, by the team of Professor Marc Cavazza. After long discussions with the designers of the first so-called HTN planning engine, we decided to completely reengineer the planning engine to make it more efficient while at the same time adding new features, such as the possibility to use more elaborated path selection functions.

The implemented planning engine is both quite effective and expressive. Its modular C++ implementation allows real time processing, even when large plans are involved. Its representation using AND/OR nodes allow any logical function to be implemented as a combination of AND and OR nodes. Eventually, when several alternatives are possible to reach a solution (OR-nodes), the system computes the cost of each alternative and selects the path with the lowest cost. The determination of this cost can be done with any deterministic or stochastic function.

6.2.3 A Functional Prototype

The research described above was led in collaboration with a team of three other researchers from the University of Teesside, United Kingdom. The gesture recognition module was coupled with a spoken lan-

guage analysis module. These two modules were able to communicate, via socked-based communication, to a HTN-planning engine, itself connected to a famous game engine (Unreal Tournament 2003 ™‘), which was responsible for the animation of the virtual characters as well as the final rendering. The mixed reality world was realized by blending the virtual stage with the image coming from the user segmentation module, using a dedicated DirectX ™‘ application.

The implemented prototype, based on a clip of a famous James Bond movie, was presented with success to several international conferences in Artificial Intelligence and Virtual Reality. It receives the *Best Paper Prize* in ICVS'03[2] and was accepted for publication in the IEEE Multimedia Magazine in July 2004. It is still considered today as a reference in the field of Mixed Reality Interactive Storytelling (MRIS).

6.2.4 Integration of Facial Expressions

Our interactive James Bond prototype demonstrated how modalities such as gestures and spoken language could be integrated in an interactive storytelling application. Once this had been realized, it was interesting to angle the research toward a new direction, a direction that could bring even more innovation and thereby extends the capabilities of the existing prototype. We therefore decided to investigate emotions as a new modality to interact with an interactive application.

As emotions are themselves expressed through a set of different modalities (facial expressions, prosody, body gestures,...), we had to restrict ourselves and chose to work on the recognition of facial expressions, since this modality obviously presented some similarity with the body gesture modality (we examine the movement of facial features instead of the movement of body feature points). The original idea was that once a satisfying facial expression recognition system would be designed, we would work on the vocal expression of emotions (the prosody). In the final part of the thesis, we would focus the research on the answer to the following question: 'How to combine the different modalities to achieve the best recognition rate?', which is a problem known as Multimodal Fusion.

[2]International Conference on Virtual Storytelling, Toulouse, France, October 2003

The reader who has read the wholeness of this thesis would have understood that we had to reexamine our ambitions: the recognition of facial expressions, from scratch, is a very difficult task. As we explained in detail in the fourth chapter, it requires mainly three steps: face detection, facial feature tracking and facial expression recognition itself. Whereas the first and third steps are reasonable challenges, the tracking of facial features remains a very tricky problem for which very few satisfying solutions have been produced. The author believes that this latter problem could actually be the subject of a PhD thesis on its own [3]... A suspicious reader will argue that he can find on the web video sequences of facial features being tracked over time, and that this is a proof that the problem is not that complicated. Indeed, techniques such as Active Appearance Models (AAMs) do provide a solution, which can be very impressive. Although we do consider AAMs as one of the best techniques for facial feature tracking, many discussions with experts in the field learned us that *all* the impressive videos are obtained when the subject whose features are tracked had actually played part in the training of the algorithm. Often, the algorithm is even trained only with the single subject whose features are to be tracked. All agreed that when the subject did not participate to the training process, AAM-based tracking algorithms often failed to converge. Moreover, AAM-based tracking algorithms are computationally costly, which constitutes another limitation factor when our goal is to build a *person-independent facial expression recognition system for an interactive application*, which is almost a totally different problem...

Now that we have briefly stated the problem, let us review the main contributions of the author in his quest for a facial expression recognition system, along with the solutions resulting from fruitful collaborations.

The integration of facial expressions as a mean to drive an interactive application is quite similar to the integration of other modalities. We can proceed as if integrating body gestures or spoken utterances: only the modality and the detection algorithms change. The result is an interactive immersive application based on affective interactions between

[3] For instance, the excellent PhD thesis of Nicolas Eveno(INPG, Grenoble) [84], published in 2003 and whose topic is: *Lips Segmentation by Using an Analytical Deformable Model*, treats only the problem of lip tracking.

the user and the application. In order to create entertainment, it is interesting to map the detected facial expressions with semantic events in the narrative. An affect-driven storytelling prototype, built on this concept, was presented in section 5.6. In section 2.5 on the other hand, we demonstrated that affective aspects could easily be integrated in the decision functions of our HTN online planning engine, via the use of so-called *affective heuristics*. Let us now focus on the resolution of the core problem of this thesis: the recognition of facial expressions.

The problem of automatic facial expression recognition includes a number of technical challenges. First, the algorithm must be able to locate and track the face in a video sequence. Once this is achieved, it must extract the information from the face images to allow the recognition task to be performed. A choice was made to tackle all these technical challenges in this thesis, because our goal was to design a functional system, that could be robust enough to be used in industrial applications. Unfortunately, this goal turned out to be too ambitious and partnerships had to be found to be able to present a global solution that covers all the technical challenges involved in the recognition of facial expressions.

Figure 6.1 depicts the overall architecture of the facial expression analysis software. It illustrates the process that enables to obtain an estimation of the user's emotional state by detecting its face, extracting facial features and comparing the extracted features to learned models of expressive faces.

Step 1: Detection of the Face

The first problem that had to be solved was the localization of the face in the first frame of the video sequence. The most popular approaches to solve this problem are those based on a boosted combination of individual weak detectors, whose most famous implementation is the Viola-Jones detector [83]. In this thesis, we have explored the distribution of color pixels for skin and non-skin regions. We have noticed that the skin chrominance has the same value range independently of the ethnicity and origin of the subject under consideration. In other words, we claimed that all humans have approximatively the same skin color and exploited this fact, along with a priori information on the shape of the face, to conceive an original face detector.

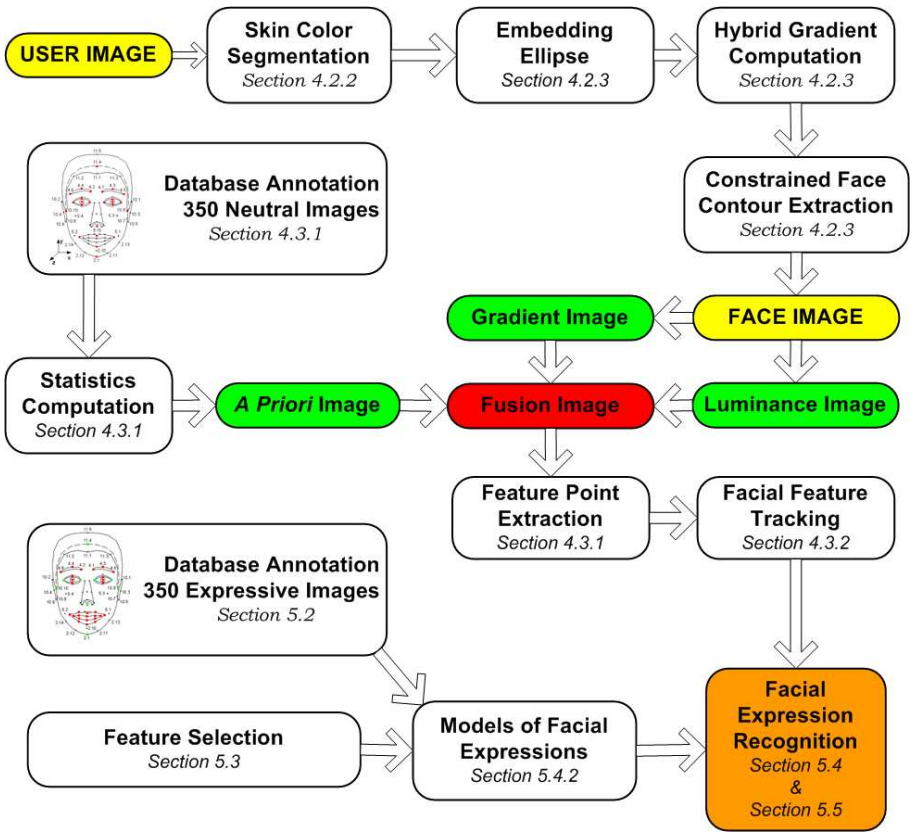


Figure 6.1: **The Facial Expression Analysis Software**: General Architecture

.

To analyze the chrominance of an image, it is necessary to map the RGB color space into a color space that separates the chrominance and luminance channels. We chose the HSV color space in which the chrominance is again divided into two components : the hue and the saturation. We showed that skin color corresponds to specific hue values and that a face detector based only on the value of hue is able to distinguish between skin and non-skin regions quite efficiently. Once potential face regions have been identified, we propose to use gradient-based filtering to detect the face contour. We combine this gradient information with a

priori knowledge of the shape of the face by adding stiffness constraints, and thereby derive an estimation of the face contour by preventing the extracted contour from moving too much aside of an egg-like shape.

Step 2: Detection of Facial Features

Once the face contour had been extracted, we aimed to search for the boundaries of the facial features: the mouth, eyebrows and eyes are features whose shape carries affective information. The main difficulty comes from the fact that these features are fast deformable objects, whose motion is both fast and highly non-linear and whose shape and texture properties vary greatly among individuals. Our approach has been driven by the conviction that a difficult problem should be divided into simpler problems in order to be solved more easily. We therefore split our problem into two parts: first, we will try to detect facial features and then, we will try to track them over time.

We started by reducing the problem of finding the boundaries of the facial features to be detected to the much simpler problem of *having an a priori idea of where in the face to look for specific points of the features to be found*. To get an estimation of the a priori location and shape of the features that we aimed to detect, we chose to manually label a large facial expression database. Once this was done, we derived from the extracted data statistics on the relative size and position of the facial features, expressed in a new coordinate system, in which positions are expressed relatively to the face bounding box. More precisely, we computed the mean and standard deviations of the position of 25 facial feature points. The means that we obtained by proceeding this way give us the *a priori* positions where we should search for the corresponding points in a new face image, while the standard deviations are an indication of the size of the search-spaces to be employed around the most probable locations, in order to have a given probability that the feature points will indeed be located inside the search-spaces that we defined.

After having showed how these statistics could ease the facial feature detection by providing local a priori search-spaces (which reduces the complexity of the original problem while increasing the robustness of the detection), we illustrated our ideas with a practical detection

algorithm. This practical example combines luminance and gradient information with statistical knowledge of the face morphology to produce an estimation of the positions of a set of ten facial feature points.

The set of detected facial feature points provides enough data to automatically initialize a facial feature tracker, based on a deformable face mask. We explained why this type of facial feature tracker was particularly efficient because it models the elastic properties of facial muscles by considering a set of facial points as a set of masses, subjected to internal and external forces. While internal forces ensure the model stays in plausible configurations, external forces tend to map the model nodes with corresponding feature points in the face image onto which it is applied. We justified why the mechanisms driving the deformation of the model should be expressed in a linear elasticity framework, so that the governing equations can be explicit and decoupled, thereby enabling real time processing. We concluded by mentioning the weaknesses of the developed tracker by assessing the need for a more global similarity measure. Among the different possibilities, we explain why a multiresolution block-matching similarity measure, such as implemented in the Lucas-Kanade-Tomasi algorithm, could be an excellent choice.

Step 3: Facial Expression Recognition

Once we consider the facial feature tracking problem solved, we can focus on the conception of a classification system to decide, at anytime, which emotion is present on the user's face. The design of such a classifier poses three main questions:

1. Which information should be taken as input of the classifier?
2. How are we going to detect the emotions ?
3. Which facial expressions are we going to detect ?

Actually, the answer to the third question was straightforward. We chose to follow Ekman's taxonomy, which defined six emotions as being universal [22], that is: expressed the same way by people from any cultural background and ethnicity. As a vast majority of the researchers in the field actually choose to do the same, most of the existing databases

are related to these six emotions and only to these six emotions. Moreover, we wanted to be able to benchmark our performances with those related in the literature. These considerations encouraged us to use the same output data vector.

Which information should be taken as input of the classifier? The answer to this question was not obvious. Many psychological studies observed how emotions are expressed on a human face. Their conclusion is that an emotion is provoked by a stimulus that induces a modification of the state of the facial muscles. The contractions (or dilatations) of these facial muscles change the appearance of the face: the eyebrows go up, the mouth opens, the eyes get almost close, etc... We will therefore take as input a *measure* of the state of the facial features. Additionally, we would like to use only information that can effectively be extracted from the face image in practical applications.

We chose to consider a set of eight facial distances because this choice satisfies our two conditions: they can be extracted without ambiguity from the face image and their values change according to the emotional state. Moreover, we compared our input data vector with the observations taken from psychological studies and came up to the conclusion that our input vector covers most of the information contained in these observations. We have opted for facial distances instead of individual facial points because we think it corresponds to a more natural way of perceiving the information, from the human point of view (we will rather say : 'The mouth has opened' than 'the lowest point of the mouth has moved downwards'). Distances are also more robust to head movements, as no motion compensation has to be performed when considering distances between two points of the same moving rigid object.

Among this set of eight facial distances, would not there exist a subset whose use would lead to better classification performances? To answer this particular question, we have tried feature selection and elimination techniques, such as forward selection and backward elimination. One of the main findings is that by using only the distances related to the mouth and the eyebrows, we could achieve a slightly better recognition rate than when eye distances were also considered. This result seems odd at first glance. We believe it is due to several factors. First, the information contained in the eyes is much more difficult to extract.

That can lead to both imprecisions in the labeling process and large quantification errors when discretizing the probability density functions corresponding to the eyes. Common sense would refute the assertion that eyes do not convey affective information, although it is what the results of our study seem to indicate. Our conclusion on that point is that it should be kept in mind that it is difficult to extract useful affective information from the eyes.

The second important observation that came out of our feature selection/elimination experience is that the most discriminative facial feature is the mouth. The shape of the mouth indeed contains enough affective information to lead to good recognition rate for several emotions, such as for surprise or happiness.

Finally, concerning the technique that we should use to detect facial expressions, two main approaches can be considered. The straightforward approach would be to derive a rule-based classification system, similar to the one developed earlier in this work for the gesture recognition system. We could build our set of rules on the observations that are related by psychological studies or edict our own rules. However, we think that the description of facial expressions in terms of facial muscle movements is not such a simple task. A smile does not simply correspond to a movement of both mouth corners: all the facial muscles are involved. Moreover, not all the people express an emotion the same way. Eventually, a person does not always express the same emotion with the same facial expression. The more we studied facial expressions, the more we were convinced that there was a high degree of **uncertainty** in the way we should model an emotion in terms of facial muscles' behavior. As the best way to deal with uncertainty is to use probabilistic reasoning [55], we decided to start with the entire input vector in consideration and to learn the relationships between an emotion and the modification of the state of the facial muscles that were induced by that emotion.

The considerations that we detailed above lead us naturally to choose Bayesian Networks (BN) as the classifier of choice because they particularly fit our needs. They provide an explicit, intuitive and understandable graphical representation of the problem. They can work in real time, and deal successfully with missing data. Last but not least, learned data can be mixed with a priori assumptions and data from several informa-

tion sources can be mixed. On the other hand, the author was attracted by Support Vector Machines (SVMs), since they are considered as one of the best techniques to classify data, when classification rules are not known. In particular, their use guarantees a global solution (unlike Neural Networks or Decision Trees for instance) to be found. The size of the input vector can grow very large as the complexity of the classifier varies according to the size of the training set, which is very useful when a reliable facial feature tracker will enable to work with much more facial features or when using models based on appearance. Eventually, both of these approaches can be easily extended with new knowledge, which is also a highly desirable property.

The fact that we were working with manually labeled data had two main advantages: it enabled the construction of reliable models, that do not suffer from segmentation errors. It also allowed to have an idea of the performances of the classifiers, independently of the performance of the upstream components. The results that we obtained with both techniques were actually quite similar, although Support Vector Machines led to a slightly better recognition rate, for most of the kernels that were tested. We invite the reader to refer to the description of the classification results for the detailed performances. The main idea to remember is that we could achieve recognition rates which are in the order of magnitude of 80% to 90%, depending on the kernel or method that was retained.

Finally, in the last section of the thesis, we investigated the integration of facial expressions in an interactive storytelling application. We presented techniques to cope with the limitations of the facial expression recognition system and discussed the basic hardware requirements to support the implementation of this technology. We concluded this thesis by explaining how, by mapping the facial expressions onto a child's face with entertaining events in a virtual world, we can create magic in a child's mind...

Appendix

6.3 Multiclass Support Vector Machines

For the sake of simplicity, we will start with the analysis of a binary classification problem whose classification result y can either be $y = -1$ or $y = +1$. Although this is not the approach that has been followed in this thesis, most N-class SVM classification problems are actually solved by combining the results of N binary SVM classifiers. Let us thus consider a training data set of l samples $\{(\mathbf{x}_i, y_i)\}_{i=1,\ldots,l}$.

The Dot Product as a Measure of Similarity

In Bayesian network formalism, the similarity between input features is reflected through their probability density functions. The choice of a similarity measure for the inputs $\mathbf{x}_i$ when these probability density functions are not known (or computed) remains a deep question that lies at the core of the field of machine learning.

One of the most intuitive ideas would be to use the *dot product* to measure the similarity between two different input vectors. Indeed, the canonical dot product $\langle \mathbf{x}, \mathbf{x}' \rangle$ computes the cosine of the angle between the vectors $\mathbf{x}$ and $\mathbf{x}$', provided that they are normalized to length 1. Moreover, it allows computation of the *length* (or *norm*) of a vector $\mathbf{x}$ as:

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} \tag{6.1}$$

Likewise, the distance between two vectors is computed as the length of the difference vector. These considerations explain the use of the

dot product as a similarity measure: all geometric relations between input variables can be characterized in the input space, whether they are expressed in terms of angles, length or distances *provided that* the input data actually exist in a dot product space. As this assumption cannot always be made, we will use a mapping function Φ to represent the input data in some dot product space $\mathcal{H}$:

$$\Phi : \mathcal{X} \mapsto \mathcal{H}, x \mapsto \mathbf{x} := \Phi(x) \tag{6.2}$$

Even if the original patterns exist in a dot product space, we may still want to consider more general similarity measures obtained by applying the mapping (6.2). In both cases, the space $\mathcal{H}$ is called the *feature space*. In that space, the dot product can be used as a similarity measure:

$$k(\mathbf{x}, \mathbf{x}') = \left\langle \Phi(\mathbf{x}), \Phi(\mathbf{x}') \right\rangle \tag{6.3}$$

The freedom to choose the mapping Φ will enable us to design a large variety of similarity measures and learning algorithms. In the remaining of this section, the function k that performs the matching between the input space and the feature space will be called the *kernel*.

Linear Support Vector Machines

Let us start with the simplest example : the case of linear machines trained on separable data. Again, we will label the data $\{\mathbf{x}_i, y_i)\}, i = 1 \dots l, y_i \in \{-1, +1\}, \mathbf{x}_i \in \mathbb{R}^n$. A classifier is said to be *linear* when the decision function $f(\mathbf{x})$ is a linear function of $\mathbf{x}$. Such a decision function can be expressed as :

$$f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b \tag{6.4}$$

where $\mathbf{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are parameters.

To decide to which class a new sample $\tilde{\mathbf{x}}$ belongs, we can decide based upon the sign of the decision function: $\tilde{y} = sign(f(\tilde{\mathbf{x}}))$. Geometrically, this is equivalent to considering a hyperplane defined by the set of points satisfying $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ and deciding to which class a new sample belongs by inspecting on which side of the hyperplane the sample $\tilde{\mathbf{x}}$ lies.

The geometric interpretation of a training set being linearly separable is the following. If the training set is linearly separable, there exists a hyperplane such as all positive (negative) samples of the training set will lie on the same side of the hyperplane:

$$\exists \mathbf{w} \in \mathbb{R}_n, b \in \mathbb{R} : y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \geq 0 \qquad \forall i = 1, \ldots, n \tag{6.5}$$

When such hyperplanes do exist, one can easily be found by applying different methods, whose first and probably most well-known is the perceptron algorithm, introduced by Rosenblatt [82]. The goal of the SVM learning algorithm is to find among the hyperplanes that separates the data into two distinct classes the one that has the largest margin on the whole training set. The *margin* introduced by an example $\mathbf{x}_i$ is defined as the Euclidean distance between this point and the closest point on the hyperplane. Taking a point $\mathbf{x}_p$ belonging to the hyperplane yields to the definition of the margin using the dot product:

$$M_i = \left\langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (\mathbf{x}_i - \mathbf{x}_p) \right\rangle \tag{6.6}$$

The margin of the training set is simply the minimum of the margins defined by the individual learning samples:

$$M = \min_{i=1,\ldots,l} M_i \tag{6.7}$$

The reason why the classification algorithm searches for the hyperplane with the largest margin is based on the heuristic that the width of the margin is inversely proportional to the probability that a new unseen example will be misclassified. This heuristic is illustrated on Figure 6.2.

Defining the hyperplane with the largest margin therefore yields the definition of two other hyperplanes H_+ and H_-, parallel to the separating hyperplane, upon which the closest positive (negative) samples of the training set will lie. With the definition of the separating hyperplane given above, it is possible that several equations correspond to the same geometrical hyperplane:

$$a \langle \mathbf{w}, \mathbf{x} \rangle + \frac{b}{a} = 0 \qquad \forall a \in \mathbb{R} \tag{6.8}$$

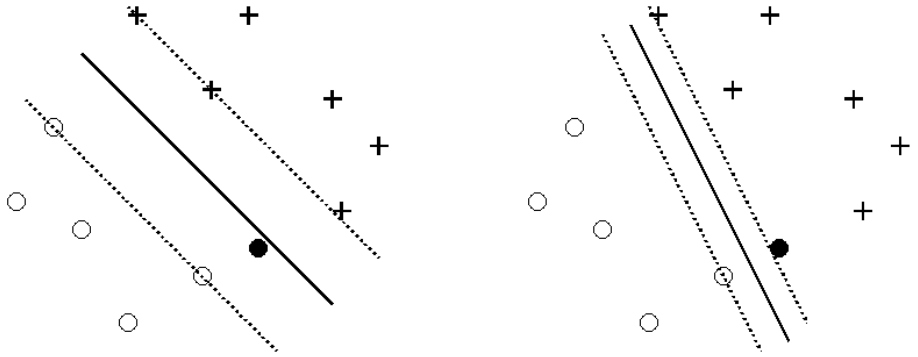


Figure 6.2: SVMs are based on the following heuristic: *the width of the margin is inversely proportional to the probability that a new unseen example will be misclassified.* We see that the black dot (representing the new unseen example) is correctly classified in the left part of the figure, in which the margin is larger. Inversely, the smaller margin in the right part of the figure leads to a classification error

.

We will therefore normalize $\mathbf{w}$ and b in such a way that the two parallel hyperplanes H_+ and H_- have the following equations:

$$H_+ : \langle \mathbf{w}, \mathbf{x} \rangle + b = +1 \tag{6.9}$$

$$H_- : \langle \mathbf{w}, \mathbf{x} \rangle + b = -1 \tag{6.10}$$

These two equations can be written in the more compact following form:

$$y_i(\langle \mathbf{w}, \mathbf{x} \rangle + b) - 1 = 0 \qquad \forall i \tag{6.11}$$

The two hyperplanes H_+ and H_- are called *canonical hyperplanes.* Combining the definition of the margin given by equation (6.6) with the definition of the canonical hyperplanes (equations (6.9) and (6.10)) leads to the value of the margin in the case of canonical hyperplanes: $\frac{1}{\|\mathbf{w}\|}$.

Those training points for which the equality (6.11) holds, and whose removal would change the solution found, are called *support vectors.* One of the advantages of the method is that only learning samples that are support vectors are used to compute the hyperplane. Consequently,

adding new learning samples requires only to check if some of these new samples become support vectors. If it is not the case, the optimization problem does have to be relaunched as we know the optimal solution will not be modified by the addition of the new learning samples.

Margin Maximization

Now that we introduced the notions of canonical hyperplanes and margin maximization, we can formulate an optimization problem whose solution would be the optimal separating hyperplane. Several formulations exist, so we will present the most used one:

$$\text{MINIMIZE} \qquad W(\mathbf{w}, b) = \frac{1}{2} \|\mathbf{w}\|^2 \tag{6.12}$$

$$\text{SO THAT} \qquad y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \geq 1 \tag{6.13}$$

The constraints (6.13) stipulate that all the samples from the training set should be correctly classified and that none of them should be located between the two canonical hyperplanes.

This constrained minimization problem can be expressed in a Lagrangian formulation. This is achieved by introducing positive Lagrange multipliers $\alpha_i, i = 1 \ldots n$, one for each of the inequality constraints (6.13). Multiplying the constraint equations (6.13) by the positive Lagrange multipliers, and subtracting the result from the objective function gives us the expression of the Lagrangian:

$$L_p \equiv \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{n} \alpha_i y_i(\langle \mathbf{x_i}, \mathbf{w} \rangle + b) + \sum_{i=1}^{n} \alpha_i \tag{6.14}$$

We must now minimize L_p with respect to $\mathbf{w}$, b, and simultaneously require that the derivatives of L_p with respect to all α_i vanish, all subject to the constraints $\alpha_i \geq 0$ (let's call this particular set of constraints C_1). This is a convex quadratic programming problem, since the objective function is itself convex, and those points which satisfy the constraints also form a convex set (any linear constraint defines a convex set, and a set of N linear constraints defines the intersection of N convex sets, which is also a convex set). This means that we can equivalently solve

the "dual" problem: *maximize* L_p, subject to the constraint that the gradient of L_p with respect to $\mathbf{w}$ and b vanishes, and also subject to the constraints that the $\alpha_i \geq 0$ (let's call *that* particular set of constraints C_2). This particular dual form of the problem is called the Wolfe dual [81]. It has the property that the maximum of L_p, subject to constraints C_2, occurs at the same value of $\mathbf{w}$, b and α, as the minimum of L_p, subject to constraints C_1. Schölkopf describes the problem as follows : Minimize L_P with respect to primal variables $\mathbf{w}$ and b, and simultaneously maximize it with respect to dual variables α_i [78].

Requiring that the gradient of L_p with respect to $\mathbf{w}$, b and α vanishes gives the conditions :

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \tag{6.15}$$

$$\sum_{i=1}^{n} \alpha_i y_i = 0 \tag{6.16}$$

Since these are equality constraints in the dual formulation, we can substitute them into equation (6.14) to obtain an expression of the dual Lagrangian, independent of both $\mathbf{w}$ and b:

$$\begin{aligned} L_D &= \tfrac{1}{2} \langle \mathbf{w}, \mathbf{w} \rangle - \textstyle\sum_{i=1}^{n} \alpha_i \left[y_i (\langle x_i, \mathbf{w} \rangle + b) - 1 \right] \\ &= \tfrac{1}{2} \textstyle\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ &\quad - \textstyle\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle + \textstyle\sum_{i=1}^{n} \alpha_i \\ &= \textstyle\sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \textstyle\sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \end{aligned} \tag{6.17}$$

The dual variables α_i can be interpreted has the *importance* that a training sample $\mathbf{x}_i$ has on the solution. A sample that lies on one of the canonical hyperplanes (a *support vector*) has $\alpha \geq 0$, whereas training samples located further from the separating hyperplane have $\alpha = 0$. Therefore, a quick inspection of equations (6.17) shows that the solution will only depend on the support vectors.

Once the expression of the dual Lagrangian has been found, the optimization problem (6.12) can be transformed into:

$$\begin{array}{ll} \text{MAXIMIZE} & L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\ \text{SO THAT} & \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \qquad\qquad \forall i = 1, \ldots, n \end{cases} \end{array} \tag{6.18}$$

Solving this optimization problem leads to the value of $\mathbf{w}$. The value of the parameter b can then be obtained by:

$$b = -\frac{\max_{y_i=-1}(\langle \mathbf{w}, \mathbf{x_i} \rangle) + \max_{y_i=+1}(\langle \mathbf{w}, \mathbf{x_i} \rangle)}{2} \tag{6.19}$$

Once $\mathbf{w}$ and b are known, the decision function is able to classify new unseen samples using:

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}, \mathbf{x_i} \rangle + b \tag{6.20}$$

Non-Linear Support Vector Machines

The technique we just described works well when the data are linearly separable. When this is not the case, the idea that first comes in mind would be to use a non-linear decision function and to apply the same kind of reasoning to determine a separating hypercurve. The problem with this reasoning is that the number of parameters needed to determine the optimal hypercurve would be very important. Moreover, we would probably introduce overfitting issues.

Instead of searching for a non-linear decision function, the idea of SVM is to map the input space towards a *feature space* in which data are linearly separable. The dimension of this feature space is generally very high, but that does not constitute a problem since the number of variables to be determined depends only on the size of the training set. (*cf.* the dual formulation of the problem, equation (6.18)). The only difference with the case of the linearly separable training set is that we work in a feature space of large dimension, induced by a kernel function k. The optimization equations are therefore identical except for the fact that the dot product is replaced by the kernel function:

$$\begin{aligned} &\text{MAXIMIZE} \quad L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \\ &\text{SO THAT} \quad \begin{cases} \sum_{i=1}^{n} \alpha_i y_i = 0 \\ \alpha_i \geq 0 \qquad\qquad \forall i = 1, \ldots, n \end{cases} \end{aligned} \tag{6.21}$$

Kernel Functions

There exists a vast theory discussing the choice and construction of kernel functions. As this theory goes far beyond the scope of this thesis, we will limit ourselves to mention that the Mercer's condition provides a sufficient and necessary condition for a function to be a kernel. A real-valued function $k(x, y)$ is said to fulfill the *Mercer's Condition* if for all square integrable functions $g(x)$ one has:

$$\int \int k(x, y) g(x) g(y) dx dy \geq 0 \tag{6.22}$$

As determining which kernel is the most appropriate for each specific problem is a task reserved for experts, we will only investigate the results that can be achieved with some of the most used kernels: the polynomial, sigmoïdal and RBF (Radial Basis Function) kernels, whose general formulations are given hereunder.

$$\text{POLYNOMIAL KERNEL} \qquad k(\mathbf{x}, \mathbf{y}) = (a * \langle \mathbf{x}, \mathbf{y} \rangle + b)^d \tag{6.23}$$

$$\text{RBF KERNEL} \qquad k(\mathbf{x}, \mathbf{y}) = \exp\Big(- \frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2} \Big) \tag{6.24}$$

$$\text{SIGMOÏDAL KERNEL} \qquad k(\mathbf{x}, \mathbf{y}) = \tanh(a * \langle \mathbf{x}, \mathbf{y} \rangle + b) \tag{6.25}$$

As equations (6.24) and (6.25) suggest, the sigmoïdal and RBF kernels map a sample to a continuous function, thereby creating a feature space of infinite dimension. The parameter σ enables to determine the width of the gaussian in the case of RBF kernels: the larger its value the more similar to distant neighbors a specific sample will be. Inversely, a very

small σ can learn any training set without errors but might perform poorly on unseen data. The same kind of reasoning may be applied for the parameter a of the sigmoïd kernel.

Soft-Margin Support Vector Machines

Even though we can find a kernel that maps the input data in a feature space in which the training set is linearly separable, it is often preferable to admit that some training data might be misclassified, due to the presence of noise in the measures or in the observation variables. Adopting this assumption can reduce overfitting issues, in which the classifier performs perfectly on training data but poorly on unseen data, because the noise on training data has also been learned by the classifier.

We could therefore, in the case of non-separable data, introduce new positive slack variables $\xi_i, i = 1 \dots l$ in the constraints to take into account classification errors in the training set (we will call these misclassified samples *outliers*). The constraints of the optimization problem could then be expressed as:

$$\langle \mathbf{x}_i, \mathbf{w} \rangle + b \geq +1 - \xi_i \quad \text{for } y_i = +1 \tag{6.26}$$

$$\langle \mathbf{x}_i, \mathbf{w} \rangle + b \leq -1 + \xi_i \quad \text{for } y_i = -1 \tag{6.27}$$

$$\xi_i \geq 0 \quad \forall i \tag{6.28}$$

Thus, for an error to occur, the corresponding ξ_i must exceed unity. We can therefore consider $\sum_i \xi_i$ as an upper bound for the number of training errors. Hence, a natural way to assign an extra cost for errors is to change the objective function to be minimized from $||\mathbf{w}||^2/2$ to $||\mathbf{w}||^2/2 + C(\sum_i \xi_i)$, where C is a parameter chosen by the user, a larger C corresponding to assigning a higher penalty to errors but also a lower propensity to maximize the margin.

The new formulation of the problem, including the possibility for a training sample to be misclassified can be expressed in the Lagrangian formulation by:

$$L_P = \frac{1}{2}||\mathbf{w}||^2 + C\sum_i \xi_i - \sum_i \alpha_i\{y_i(\langle \mathbf{x}_i, \mathbf{w}\rangle + b) - 1 + \xi_i\} - \sum_i \mu_i\xi_i \quad (6.29)$$

where the μ_i are the Lagrange multipliers introduced to enforce positivity of the ξ_i. The solution is again given by :

$$\mathbf{w} = \sum_{i=1} \alpha_i y_i \mathbf{x}_i \quad (6.30)$$

The only difference from the optimal hyperplane case is that the α_i now have an upper bound of C:

$$0 \leq \alpha_i \leq C, \quad (6.31)$$

$$\sum_i \alpha_i y_i = 0 \quad (6.32)$$

The dual Lagrangian has the same expression than in the case of a separable training set:

$$L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) \quad (6.33)$$

Solving this optimization problem leads to the global solution of the problem. Indeed, since the kernel matrix is positive definite, each local solution is also a global one, which is a major advantage compared to traditional neural network techniques.

Multiclass Support Vector Machines

Our original problem is to classify six different emotions. We will thus extend the theory previously exposed to the case of multiclass SVM. It should be noticed at this stage that there is a straight-forward approach to perform M-class classification from binary classifiers: we can construct M binary classifiers of the type *one-versus-the-rest* and choose the class which receives the highest score, before the application of the sign function. Alternatively, $M(M-1)/2$ hyperplanes could be constructed, separating each class from each other, and some voting scheme applied to perform the classification task.

Although using the binary classification scheme may reveal handy at first glance, it does not seem the most intuitive way to tackle our problem. We will therefore choose to extend the ideas developed before to the multiclass case, following Vapnik's philosophy of *directly tackling the problem.*

The first step is to extend the objective function to take as an objective the maximization of *the sum of all* the margins and the fact that an error can be the result of the confusion with any other possible class:

$$W(\mathbf{w}, \xi) = \frac{1}{2} \sum_{m=1}^{k} \|\mathbf{w_m}\|^2 + C \sum_{i=1}^{n} \sum_{m \neq y_i} \xi_i^m \tag{6.34}$$

The constraints specify that for all considered classes $i \in \{1, \ldots, k\}$, there exists a hyperplane h_i that classifies correctly the samples from the learning set according to their class label y_i, taking into account the possibility that some learning samples might be misclassified.

Expressing that all samples belonging to the class label y_i must be situated on the positive side of the hyperplane h_i (unless they are outliers in any classes other than class i) leads to the following inequality:

$$\langle \mathbf{w_{y_i}}, \mathbf{x_i} \rangle + b_{y_i} \geq 1 - \xi_i^{m=\{1,\ldots,k\} \backslash y_i} \tag{6.35}$$

Inversely, imposing samples from class label i to be located on the negative side of other classes' hyperplanes leads to:

$$\langle \mathbf{w_m}, \mathbf{x_i} \rangle + b_m \leq -1 + \xi_i^{y_i} \tag{6.36}$$

By combining equations (6.35) and (6.36), we obtain the expression of the constraints in the multiclass case:

$$\langle \mathbf{w_{y_i}}, \mathbf{x_i} \rangle + b_{y_i} \geq \langle \mathbf{w_m}, \mathbf{x_i} \rangle + b_m + 2 - \xi_i^m - \xi_i^{y_i}, \tag{6.37}$$

$$\xi_i^{y_i} \geq 0, \xi_i^m \geq 0, \quad i = 1, \ldots, n \quad m \in \{1, \ldots, k\} \backslash y_i \tag{6.38}$$

In this case the decision function becomes:

$$f(x) = \arg\max_k [\langle \mathbf{w_i}, \mathbf{x} \rangle + b_i], \quad i = 1, \ldots, k \tag{6.39}$$

Since the Lagrangian formulation of the problem in the multiclass case involves elaborated mathematical developments and does not bring any useful information to our discussion, we will not expose the entire reasoning in detail in this section. The interested reader is invited to consult [79] for the complete development of the primal and dual solution of the problem. To summarize, we find the saddle point of the Lagrangian and, by substituting the results obtained at the saddle point back into the expression of the Lagrangian, we derive the expression of the dual problem, which is again a quadratic function in terms of alpha, with linear constraints. The resolution of this dual problem leads to the final expression of the decision function, which is given by:

$$f(\mathbf{x}, \alpha) = \arg\max_{n} \Big[\sum_{i:y_i=n} A_i \left\langle \mathbf{x_i}, \mathbf{x} \right\rangle - \sum_{i:y_i \neq n} \alpha_i^n \left\langle \mathbf{x_i}, \mathbf{x} \right\rangle + b_n \Big] \tag{6.40}$$

with:

$$A_i = \sum_{m=1}^{k} \alpha_i^m \tag{6.41}$$

Bibliography

[1] A. Mehrabian and F. R. Ferris: Inference of attitudes from nonverbal communication in two channels. In *Journal of Consulting Psychology, Vol. 31, pp 248-252*, 1967.

[2] C. Crawford: Chris Crawford on Interactive Storytelling. *New Riders Games, 2004.*

[3] M. Cavazza, F. Charles, and S.J. Mead: Interacting with Virtual Characters In Interactive Storytelling. *Proc. of the First International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 318–325, ACM Press, New York, 2002.*

[4] P. Milgram and F. Kishino: A Taxonomy of Mixed Reality Visual Displays. *IEICE Transactions on Information Systems, Vol. E77-D, N12, December 1994.*

[5] M. Cavazza, O. Martin, F. Charles, S.J. Mead, X. Marichal, A. Nandi : *Multi-modal Acting in Mixed Reality Interactive Storytelling*, *IEEE Multimedia Magazine, July 2004.*

[6] J. Jacobson, Z. Hwang: Unreal Tournament for Immersive Interactive Theater. *Communications of the ACM, Vol. 45, N1, 2002.*

[7] F. Charles, M. Cavazza, S.J. Mead, O. Martin, A. Nandi, and X. Marichal: Compelling experiences in

mixed reality interactive storytelling, *International Conference on Advances in Computer Entertainment Technology (ACE'04), Singapore, 2004*

[8] Q. Limbourg, and J. Vanderdonckt: Comparing Task Models for User Interface Design, *Chapter 6, in Diaper, D., Stanton, N. (Eds.), The Handbook of Task Analysis for Human-Computer Interaction, pp. 135–154, Lawrence Erlbaum Associates, Mahwah, 2003.*

[9] M. F. Kleyn, I. Chakravarty: EDGE - a Graph Based Tool for Specifying Interaction. *ACM Symposium on User Interface Software and Technology, October 17-19, Alberta, Canada, 1988.*

[10] M. Cavazza, F. Charles, and S.J. Mead: Characters in Search of an Author: AI-based Virtual Storytelling. *International Conference on Virtual Storytelling, pp.145–154, Avignon, France, 2001.*

[11] N. J. Nilsson, N. J.: Principles of Artificial Intelligence. *Palo Alto, CA: Tioga, 1980.*

[12] R. Barthes: Introduction l'Analyse Structurale des Récits (in French). *Communications, 8, pp.1-27, 1966.*

[13] M. Cavazza, F. Charles, and S.J. Mead: Under The Influence: Using Natural Language in Interactive Storytelling. *1st International Workshop on Entertainment Computing, IFIP Conference Proceedings, 240, Kluwer, pp. 3-11, 2002.*

[14] S.J. Mead, M. Cavazza, and F. Charles: Influential Words: Natural Language in Interactive Storytelling. *10th International Conference on Human-Conputer Interaction, Crete, Greece, 2003, Vol 2., pp.741-745.*

[15] J. Duetscher, A. Blake, and I. Reid: Articulated Body Motion Capture by Annealed Particle Filtering,

IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00) - Volume 2, 2000.

[16] M. Isard, and A. Blake: Contour Tracking by Stochastic Propagation of Conditional Density, *Proceedings of the 4th European Conference on Computer Vision-Volume I - Volume I, 1996.*

[17] M. Becker , E. Kefalea, E. Mal, C. Von Der Malsburg, M. Pagel, J. Triesch, J. C. Vorbrggen, R. P. Wrtz, and S. Zadel: GripSee: A Gesture-Controlled Robot for Object Perception and Manipulation, in *Autonomous Robots, Volume 6, Number 2, pp. 203–221, April, 1999.*

[18] B. Leibe, T. Starner, W. Ribarsky, Z. Wartell, D. Krum, B. Singletary, and L. Hodges : The Perceptive Workbench: Toward Spontaneous and Natural Interaction in Semi-Immersive Virtual Environments, *IEEE Virtual Reality Conference 2000 (VR'00), 2000.*

[19] P. Maes, D. Trevor, B. Blumberg, and A. Pentland: The ALIVE system: full-body interaction with autonomous agents, in *Computer Animation '95, 1995.*

[20] X. Marichal, and T. Umeda: Real-Time Segmentation of Video Objects for Mixed-Reality Interactive Applications, *Proceedings of the "SPIE's Visual Communications and Image Processing" (VCIP 2003) International Conference, Lugano, Switzerland, 2003.*

[21] A. Nandi, and X. Marichal: Senses of Spaces through Transfiction, pp. 439-446 in *Entertainment Computing: Technologies and Applications, Proceedings of the International Workshop on Entertainment Computing, IWEC 2002.*

[22] P. Ekman: Emotion in the human face, *Cambridge University Press, New York, 1982.*

[23] P. Ekman: Facial expression of emotion: New findings, new questions. *Psychological Science, vol. 3, pp. 34–38, 1992.*

[24] P. Ekman, and W. V. Friesen: Unmasking the face. A guide to recognizing emotions from facial clues. *Englewood Cliffs, Prentice-Hall, New Jersey, 1975.*

[25] P. Ekman, and H. Oster: Facial expressions of emotion. *Annual Review of Psychology, vol. 20, pp. 527–554, 1979.*

[26] Z. Hammal, N. Eveno, A. Caplier, P.-Y. Coulon: Extraction réaliste des traits caractéristiques du visage à l'aide de modèles paramétriques adaptés. *19ème colloque sur le traitemant du signal et des images, Gretsi 2003, Paris, September 2003.*

[27] E. Osuna, R. Freund, and F. Girosi: Training Support Vector Machines: an Application to Face Detection. In *IEEE Conference on Computer Vision and Pattern Recognition, 1998.*

[28] H.A. Rowley, S. Baluja, and T. Kanade: Neural Network-Based Face Detection. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.20(1), pp. 23–38, 1998.*

[29] K. Choong Yow, and R. Cipolla: Feature-Based Human Face Detection. *Image and Vision Computing, 15(9),pp. 713–735, 1997.*

[30] B. Menser, and F. Muller: Face Detection in Colour Images Using Principal Component Analysis, *Proc. Image Processing and its Applications, 1999.*

[31] Y. Raja, S. J. McKenna, and S. Gong: Tracking and Segmenting People in Varying Lighting Conditions Using Colour. *In IEEE International Conference on Face and Gesture Recognition, pp. 228–233, Nara, Japan, 1998.*

[32] K. Sobottka, I. Pitas: Looking for Faces and Facial Features in Color Images. In *Advances in Mathematical Theory and Applications, Vol. 7, No. 1, 1997.*

[33] T. F. Cootes and C. J. Taylor: Active shape models – 'Smart Snakes', *Proc. BMVC, , pp. 266-275, Leeds, United Kingdom, 1992.*

[34] K. W. Wan, K. M. Lan, and K. C. Ng: An Accurate Active Shape Model For Facial Feature Extraction, *Pattern Recognition Letters, Vol. 26, pp. 2409–2423, 2005.*

[35] T. Kanade, J. F. Cohn, and Y. L. Tian: Comprehensive database for facial expression analysis. *Proceedings of the 4th IEEE International Conference on Automatic Face and Gesture Recognition (FG'00), pages 46–53, 2000.*

[36] C. Harris and M. Stephens: A combined corner and edge detector. *Proceedings of the 4th Alvey Vision Conference, pages 147–151, 1988.*

[37] M. Gokmen, and C. Li: Edge detection and surface reconstruction using refined regularization, *IEEE Transactionss on Pattern Analysis and Machine Intelligence, Vol. 15, pp. 492-499, 1993.*

[38] F. Leymarie and M.D. Levine: Tracking deformable objects in the plane using an active contour model. *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 15(6), pp. 617–634, 1993.*

[39] O. Martin et al.: A Multimodal Caricatural Mirror. *Proceedings of the eNTERFACE Summer Workshop on Multimodal Interfaces, pp.13-20, Mons, Belgium, July-August 2005.*

[40] K.Sobottka and I.Pitas: Segmentation and Tracking of Faces in Color Images, *in Proc. of 2nd Int. Conf.*

on Automatic Face and Gesture Recognition, pp. 236-241, Killington, Vermont, USA, 1996.

[41] M. Kass, A. Witkin, and D. Terzopoulos: Snakes: Active Contour Models. *International Journal of Computer Vision, pp. 321–331, 1987.*

[42] A. Blake, and M. Isard: Active Contours. *SpringerVerlag, 1998.*

[43] S. Krinidis, and I. Pitas: Fast free-vibration modal analysis of 2-D Physics-Based Deformable Objects. *IEEE Transactions on Image Processing 14(3), pp. 281–293, 2005.*

[44] C. Nastar, and N. Ayache: Fast Segmentation, Tracking, and Analysis of Deformable Objects. *Proceedings of the Fourth International Conference on Computer Vision, ICCV'93, Germany, 1993.*

[45] C. Nastar and N. Ayache: Frequency-based nonrigid motion analysis: Application to four dimensional medical images, *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 11, pp. 10691079, 1996.*

[46] A. Pentland and S. Sclaroff: Closed-form solutions for physically-based shape modeling and recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 13, no. 7, pp. 730742, 1991.*

[47] C. Nikou, G. Bueno, F. Heitz, and J. P. Armspach: A joint physics-based statistical deformable model for multimodal brain image analysis, *IEEE Transactions on Medical Imaging, vol. 20, no. 10, pp. 10261037, 2001.*

[48] M. Rydfalk: CANDIDE: A parameterized face. *Technical Report, Linkoping University,Sweden, 1978.*

[49] J. Y. Bouguet: Pyramidal implementation of the Lucas-Kanade feature tracker. *Intel Corporation, Microprocessor Research Labs, 1999.*

[50] Y. Yacoob, and L.S.Davis: Recognizing Human Facial Expressions From Long Image Sequences Using Optical Flow, *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 6, June 1996.*

[51] J.J. Lien, T. Kanade, J.F. Cohn, Ching-Chung Li: Automated Facial Expression Recognition Based on FACS Action Units, *3rd IEEE International Conference on Automatic Face and Gesture Recognition, Nara, Japan, 1998.*

[52] I.A. Essa, and A.P. Pentland: Facial Expression Recognition Using a Dynamic Model and Motion Energy, *5th International Conference on Computer Vision, Cambridge, MA, USA, 1995.*

[53] P. Ekman, and W.V. Friesen: Facial Action Coding System, *Palo Alto (CA): Consulting Psychologists Press, 1978.*

[54] 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, Issue 2, pp.97–115, 2001.*

[55] J. Pearl: Probabilistic Reasoning in Intelligent Systems, *Morgan-Kaufmann, 1988.*

[56] L. Wiskott, J.-M. Fellous, N. Kruger, and C. von der Malsburg: Face Recognition by Elastic Bunch Graph Matching, *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7, June 1997.*

[57] S. Lawrence, C.L. Giles, Tsoi Ah Chung, and A.D. Back: Face Recognition: a Convolutional Neural-Network Approach, *IEEE Transactions on Neural Networks, Vol. 8, Issue 1, Jan 1997.*

[58] N. Sebe, M.S. Lew, I. Cohen, Y. Sun, T. Gevers, and T.S. Huang: Authentic Facial Expression Analysis, *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, p. 517, 2004.*

[59] I. Cohen, N.Sebe, F.G. Gozman, M.C. Cirelo, and T.S. Huang: Learning Bayesian network classifiers for facial expression recognition both labeled and unlabeled data, *IEEE Conference on Computer Vision and Pattern Recognition, pp.595–601, 2003.*

[60] I. Cohen, N.Sebe, A. Garg, M.S. Lew, and T.S. Huang: Facial Expression Recognition from Video Sequences, *International conference on Multimedia and Expo, 2002.*

[61] P. Michell, and R. Al Kaliouby: Real time facial expression recognition in video using support vector machines. *Proceedings of the 5th international conference on Multimodal interfaces*, pp. 258–264, Vancouver, 2004.

[62] I. Kotsia, and I. Pitas: Real-Time Facial Expression Recognition from Image Sequences Using Support Vector Machines. *Proceedings of the International Conference on Image Processing, Geneva, 2005.*

[63] http://emotion-research.net/earl

[64] P. Naïm, P.H. Wuillemin, P. Leray, O. Pourret, and A. Backer: Réseaux Bayésiens, *2nd edition, Eyrolles, 1999.*

[65] http://www.csc.ncsu.edu/faculty/bahler/courses/csc520f02/bayes1.html

[66] http://www.cs.ubc.ca/ murphyk/Bayes/bnintro.html

[67] N. Friedman, D. Geiger, and M. Goldszmidt: Bayesian Network Classifiers. *In Journal on Machine Learning, Volume 29, No. 2-3, pp. 131–163, November 1997.*

[68] P. Domingos, and M. Pazzani: Beyond Independence: Conditions for the Optimality of Simple Bayesian Classifier. *Machine Learning, 29:103–130, 1997.*

[69] C. Elkan: Boosting and Naïve Bayesian Learning. *Technical Report CS97-557, Department of Computer Science, University of California, San Diego, 1997.*

[70] A. Garg, and D. Roth: Understanding probabilistic classifiers. *Proceedings of the European Conference on Machine Learning, pp. 179–191, 2001.*

[71] P. Ekman, and W. Friesen: Facial Action Coding System: Investigator's Guide. *Palo Alto: Consulting Psychologists Press, 1978.*

[72] A. Bellili, M. Gilloux, P. Gallinari: An MLP-SVM combination architecture for offline handwritten digit recognition. *International Journal on Document Analysis and Recognition, Volume 5, Number 4, pp. 244-252, July 2003.*

[73] E. Osuna, R. Freund, F. Girosit : Training support vector machines: an application to face detection. *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, 1997.*

[74] V. Wan, and W.M. Campbell: Support vector machines for speaker verification and identification. *Proceedings of the IEEE Signal Processing Society Workshop on Neural Networks for Signal Processing, Sydney, 2000.*

[75] T. Joachims: Text Categorization with Suport Vector Machines: Learning with Many Relevant Features. *Proceedings of the 10th European Conference on Machine Learning, pp. 137–142, 1998.*

[76] C. Burges: A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery, 2:121–167, 1998.*

[77] V. Vapnik: The Nature of Statistical Theory; *Springer Verlag, 1995.*

[78] B. Scholkopf, A. J. Smola: Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond (Adaptive Computation and Machine Learning). *MIT Press, 2002.*

[79] J. Weston, and C. Watkins: Multi-class Support Vector Machines. *Technical Report CSD-TR-98-04, 2004.*

[80] J. Callut: Implmentation Efficace des Support Vector Machines pour la Classification. *Master Thesis, ULB, 2003.*

[81] R. Fletcher: Practical Methods of Optimization. *John Wiley and Sons, Inc., 2nd Edition, 1987.*

[82] F. Rosenblatt: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review, 65(6), pp. 386–408, 1958.*

[83] P.Viola, and M.J.Jones: Robust Real-Time Face Detection. *International Journal of Computer Vision, 57(2), pp. 137–154, 2004.*

[84] N. Eveno: Lips segmentation by using an analytical deformable model. *PhD Thesis, Grenoble, 2003.*

[85] C. Tomasi, and T. Kanade: Detection and Tracking of Point Features. *Carnegie Mellon University Technical Report CMU-CS-91-132, April 1991.*

[86] http://www.similar.cc

[87] F. Charles, O. Martin, M. Cavazza, S.J. Mead, A.Nandi and X. Marichal: Compelling Experiences in Mixed Reality Interactive Storytelling. *International Conference on Advances in Computer Entertainment Technology, pp. 32-41, ACE 2004, Singapore, 2004* .