

# Facial Expression Recognition Using Local Binary Patterns and Support Vector Machines



**AALBORG UNIVERSITY**  
DENMARK

Department of Electronic Systems  
Vision, Graphics and Interactive Systems  
9<sup>th</sup> Semester project

Autumn 2012  
Maxime Coupez  
Kim-Adeline Miguel  
Julia Alexandra Vigo



**Title:**

Facial Expression Recognition Using Local Binary Patterns and Support Vector Machines

**Theme:**

Interactive Systems

**Project Period:**

Fall Semester 2012

**Project Group:**

12gr942

**Participant(s):**

Maxime Coupez  
Kim-Adeline Miguel  
Julia Alexandra Vigo

**Supervisor(s):**

Zheng-Hua Tan

**Copies:** 2

**Page Numbers:** 95

**Date of Completion:**

December 19, 2012

**Abstract:**

Since the last decade, a lot of research has been carried out in the area of emotion recognition. The number of projects conducted in this field demonstrates the interest and the importance of systems that can recognize human emotion.

In this project, a facial emotion recognition system is developed, using a Microsoft Kinect as the sensor. This recognition is achieved in three steps: Face detection, facial features extraction and classification.

Face detection is performed using Viola-Jones' algorithm, Local Binary Patterns (LBP) method is used to extract facial features, and Support Vector Machines (SVM) classify these features into six predefined emotions plus the neutral state.

The system is implemented to run on a computer using a Kinect and works for one person in front of it. The classifier is trained with the Karolinska Directed Emotional Faces database, which includes a high number of face images to obtain a good result.

*The content of this report is freely available, but publication (with reference) may only be pursued due to agreement with the author.*



# Preface

This report documents the semester project entitled *Facial Expression Recognition Using Local Binary Patterns and Support Vector Machines*. The project was carried out during the 9th semester of specialization *Vision, Graphics, and Interactive Systems* under the Department of Electronic Systems at Aalborg University in Autumn 2012.

The report is divided into five parts plus appendices: *Introduction*, *Feature Detection*, *Feature Classification*, *Implementation* and *Evaluation*. The first part reviews the general structure of a facial expression recognition system and its main issues, and concludes with a state of the art of existing systems. Analysis of possible solutions and design of our system are contained in the following two parts, and the fourth part describes our implementation. The last part evaluates the performance and accuracy of our system and concludes on the project as a whole.

References to secondary literature sources are made using the syntax [number]. The number refers to the alphabetically sorted bibliography found at the end of the report, just before the appendices.

We would like to thank our supervisor at Aalborg University Zheng-Hua Tan for supporting us in this challenging project.

A CD is attached to this report which includes:

- Source code of the developed program.
- PDF file of this report.

Aalborg University, December 19, 2012

---

Maxime Coupez  
<mcoupe12@es.aau.dk>

---

Kim-Adeline Miguel  
<kmigue12@es.aau.dk>

---

Julia Alexandra Vigo  
<jvigo12@es.aau.dk>



# Abbreviations

AU	Action Units
CK	Cohn-Kanade
DLBP	Dominant Local Binary Patterns
HMM	Hidden Markov Models
JAFFE	Japanese Female Facial Expression
KDEF	Karolinska Directed Emotional Faces
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
MSFDE	Montreal Set of Facial Displays of Emotion
MVC	Model-View-Controller
NN	Neural Network
PCA	Principal Component Analysis
RBF	Radial Basis Function
ROC	Receiver Operating Characteristic
ROI	Region Of Interest
SIFT	Scale Invariant Feature Transform
SVM	Support Vector Machines



# Contents

<b>Preface</b>	<b>v</b>
<b>Abbreviations</b>	<b>vii</b>
<b>I Introduction</b>	<b>2</b>
<b>1 Motivations</b>	<b>4</b>
<b>2 Facial expression recognition</b>	<b>6</b>
2.1 General structure . . . . .	6
2.2 Environment Setup . . . . .	8
2.3 Facial Expression Datasets . . . . .	8
2.4 Issues . . . . .	13
2.5 Requirements . . . . .	17
<b>II Face detection</b>	<b>19</b>
<b>3 Face detection</b>	<b>21</b>
<b>4 Viola-Jones Face detection</b>	<b>23</b>
4.1 Overview . . . . .	23
4.2 Haar features . . . . .	23
4.3 Integral image . . . . .	28
4.4 Weak classifiers and AdaBoost . . . . .	29
4.5 Classifiers cascade . . . . .	32
4.6 Test set and training . . . . .	34
<b>III Feature extraction</b>	<b>37</b>
<b>5 Feature extraction</b>	<b>39</b>
5.1 Appearance-based methods . . . . .	40
5.2 Feature-based methods . . . . .	40
<b>6 Local Binary Patterns</b>	<b>43</b>
6.1 Overview . . . . .	43
6.2 State of the art . . . . .	44

6.3 Facial Expression Recognition based on LBP . . . . .	48
6.4 Histogram computing . . . . .	48
<b>IV Classification</b>	<b>50</b>
<b>7 Classification</b>	<b>52</b>
7.1 Supervised learning . . . . .	53
7.2 Unsupervised learning . . . . .	54
<b>8 Support Vector Machines</b>	<b>58</b>
8.1 Overview . . . . .	58
8.2 Margin maximization . . . . .	59
8.3 Kernel function . . . . .	60
8.4 Combining LBP and SVM . . . . .	61
<b>V Implementation</b>	<b>63</b>
<b>9 Face detection with Viola-Jones</b>	<b>65</b>
9.1 Viola-Jones . . . . .	65
<b>10 Feature extraction with Local Binary Patterns</b>	<b>66</b>
10.1 Uniform Local Binary Patterns . . . . .	66
10.2 Histogram computing . . . . .	66
<b>11 Classification with Support Vector Machines</b>	<b>68</b>
11.1 Histogram concatenation . . . . .	68
11.2 Training and Model . . . . .	69
<b>12 Implementation on the Kinect</b>	<b>71</b>
12.1 Generalities . . . . .	71
12.2 Librairies . . . . .	71
12.3 Architecture . . . . .	72
12.4 Interactions . . . . .	72
12.5 Algorithm . . . . .	73
<b>VI Evaluation</b>	<b>75</b>
<b>13 Results</b>	<b>77</b>
13.1 First result set . . . . .	77
13.2 Second result set . . . . .	81
13.3 Additional results . . . . .	82

<b>14 Issues</b>	<b>85</b>
14.1 Feature extraction . . . . .	85
14.2 Real-time . . . . .	85
14.3 Issues with the implementation on the Kinect . . . . .	86
14.4 Training dataset . . . . .	87
<b>15 Conclusion</b>	<b>88</b>
15.1 Theoretical framework . . . . .	88
15.2 Results . . . . .	88
15.3 Future work . . . . .	89
<b>Bibliography</b>	<b>91</b>
<b>Appendix: Source Code</b>	<b>95</b>

**Part I**

**Introduction**

# Contents

*The main motive of this project is to understand facial expression recognition systems and their applications. A review of the architecture of such systems will be done, along with a state of the art of already existing algorithms. After this study, issues coming along with this kind of recognition system will be studied. In the last part, the requirements of this project will be formulated.*

<b>1</b>	<b>Motivations</b>	<b>4</b>
<b>2</b>	<b>Facial expression recognition</b>	<b>6</b>
2.1	General structure . . . . .	6
2.2	Environment Setup . . . . .	8
2.3	Facial Expression Datasets . . . . .	8
2.4	Issues . . . . .	13
2.5	Requirements . . . . .	17

# Chapter 1

## Motivations

A facial expression is a "visible manifestation of the effective state, cognitive activity, intent, personality, and psychopathology of a person" [10]; facial expressions represent a huge part in dialogue and interaction with other humans. Indeed, facial expressions carry more information than speech, information on which humans can rely for interaction. Facial expressions have a considerable effect on a listening interlocutor [27].

Albert Mehrabian found that facial expressions represents 55% of information received by a listener, while 38% of information received comes from voice intonation and the remaining 7% by the spoken words. To find these numbers, Albert Mehrabian conducted an experiment based only on the word *maybe*. Three female subjects were recorded and taped while pronouncing this word with different intonations to express different attitudes addressed to an imaginary listener. The different attitudes was, for example, *like*, *dislike* and *neutral*. 17 female subjects were then asked to guess the attitudes expressed by the first three subjects. The second experiment was to take pictures of the three female subjects expressing the different attitudes. Then the 17 female subjects had to guess again the attitudes of the three female subjects, this time while hearing the recordings and seeing the pictures at the same time. The output of these experiments was that the facial part carried more information than the vocal one with a ration of  $\frac{3}{2}$ . It corresponds to the 55% and the 38% mentioned previously [27]. This percentage can be however applied for only certain specific situations, since in some situations speech is more important than facial cues.

Since antiquity, researchers have been interested in emotion, and more particularly in emotion recognition. One of the most important studies on facial expression analysis impacting on modern day science of automatic facial expression recognition is the work carried out by Charles Darwin [4]. In 1872, Darwin wrote a book that established general expression principles, expression means and expression description for both humans and animals [8]. He also classified various kinds of expressions. This can be considered as the beginning of facial expression recognition.

Nowadays, with the emergence of new technologies and computers, research is now focused on computer-based automatic facial expression recognition. Because facial expressions are major factors in human interaction, this research field will improve the domain of Human-Machine Interaction. Indeed, emotion recognition will enable

computers to be more responsive to users' emotions, and allow interactions to become more and more realistic.

Another domain where facial expression recognition is an important issue is robotics. With the advances made in robotics, robots tend to mimic human emotion and react as human-like as possible, especially for humanoid robots. Indeed, since robots are being more and more present in our daily lives, they need to understand and recognize human emotions.

A lot of applications in the robotics field have already been created. For example, Bartlett et al. have successfully used their face expression recognition system to develop a character that is animated and that mirrors the expressions of the user (called CU Animate) [3]. They have also been successful in deploying the recognition system on Sony's Aibo Robot and ATR's RoboVie [3]. Another interesting application has been demonstrated by Anderson and McOwen, called "EmotiChat" [2]. It is a regular chatroom, except the fact that their facial expression recognition system is connected to the chat and convert the users' facial expressions into emoticons. Because facial expression recognition systems' robustness and reliability are constantly increasing, lots of innovative applications will appear.

There are also various other domains where emotion recognition can be used: Telecommunications, behavioural science, video games, animations, psychiatry, automobile safety, affect-sensitive music jukeboxes and televisions, educational software, etc [4].

This project focuses on facial expression recognition from a video stream. Indeed, facial expression recognition can be performed *statically* on input images, or *dynamically* on video sequences. Systems can also be *obtrusive*, or *non-obtrusive*, the former based on a device mounted on the user's head or body, therefore following each of his movements and perform facial expression recognition without much losses, while the latter can encounter difficulties if the user is not properly situated. However, non-obtrusive systems allow more natural user interactions. The system described in this report was chosen to be non-obtrusive, and its setup will be detailed further in the next section.

# Chapter 2

## Facial expression recognition

After having stated the conditions and motivations of this project, we will now describe the general structure of a facial expression recognition system. It can indeed be roughly summed up as classification applied to a pre-processed image. An overview of pre-processing steps will be done in this chapter, while feature extraction and classification will be more detailed respectively in Parts 5 and 7. Following sections will be about issues raised by facial expression recognition systems, and key requirements these systems have to meet in order to be considered acceptable.

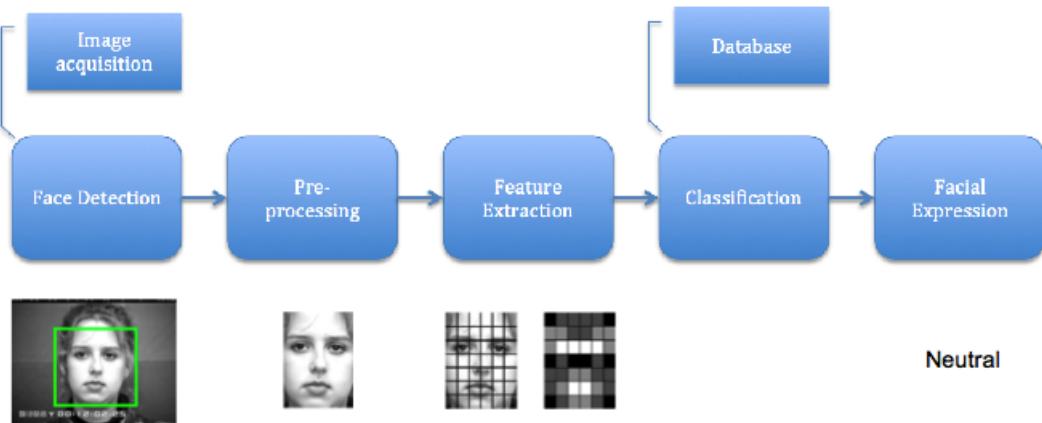
### 2.1 General structure

Facial expression recognition is a system enabling an automatic recognition of emotions displayed by a human face. Facial expression recognition can be image or video-based; it can also be computed in real-time if needed. Researchers usually try to recognize emotions out of static images. It can also be achieved real-time on video streams: While the person displays his/her emotions, the facial expression recognition system analyses the video, and detect the displayed emotion.

In both cases, facial expression recognition process is structured as in Figure 2.1.

#### 2.1.1 Image Acquisition

The first step is *Image Acquisition*. Images used for facial expression recognition can be static images or image sequences, with the latter giving more informations about the displayed expression, i.e steps in muscles movement. About static images, facial expression recognition systems usually take 2D greyscale images as inputs. We can however expect future systems to use color images; first because of the increasing affordability of technologies and devices capable of capturing images or image sequences; then because colors can give more information on emotions, for example blushing [7].



**Figure 2.1:** Facial expression recognition process

### 2.1.2 Face Detection

Second step is *Face Detection*. Indeed, in a static image and even more in an images sequence, this is an obvious need. Once the face has been detected, all other non-relevant information can be deleted. This step could hence be included in the next step, which is *Pre-processing*, but because of its importance it can be considered as a step in itself. If working with image sequences or video streams, the face has to be detected and tracked. One of the most used and famous detection and tracking algorithm is the Viola-Jones algorithm, which will be explained further in Chapter 4. This algorithm can be trained to detect all kind of objects, but is mostly used for face detection.

### 2.1.3 Pre-processing

Third step is *Pre-processing*, where image processing algorithms are applied to the image in order to prepare it for the next step. Pre-processing is usually about noise removal, normalization against the variation of pixel position or brightness, segmentation, location or tracking of parts of the face. Transformation, scaling and rotation of the head in the image or image sequence have an effect on emotion recognition. In order to solve this problem, the image can be geometrically standardized, with the eyes generally used as reference points [7].

#### 2.1.4 Feature Extraction

Once the image has gone through the *Pre-processing* step, the next one is *Feature Extraction*. In this step, data is converted "into a higher representation of shape, motion, color, texture, and spatial configuration of the face or its components" [7]. One of the main goals of this step is to reduce the dimensionality of the input data. The reduction procedure should retain "essential information possessing high discrimination power and high stability" [7]. There are a lot of features extraction methods, i.e Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Local Binary Patterns (LBP), Gabor Wavelets. This step will be detailed further in Part 5. The extracted data is then used in the *Classification* step.

#### 2.1.5 Classification

The *Classification* step is the culminating point of the facial expression recognition process. There are many kinds of classification algorithms, some of them can even be used in the feature extraction part, as it will be detailed in Part 5. This step takes into input a model previously trained with pre-processed data, and test data made of feature vectors extracted from the image we want to label. Feature vectors from pre-processed data and test data have to be obtained using the same feature extraction algorithm. The chosen classifier then outputs a value corresponding to the label of the class the picture belongs to.

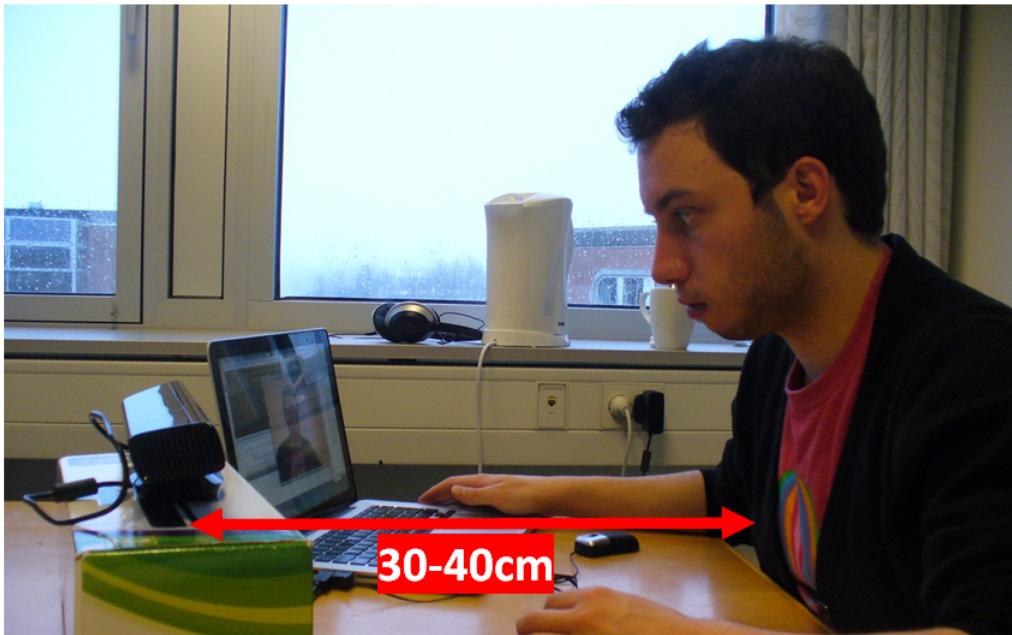
## 2.2 Environment Setup

Our system will use the camera embedded into a Microsoft Kinect to record the user's video input. We will consider a casual use of the camera, the user sitting in front of the computer, the camera being next to him or her, as seen in Figure 2.2. The distance between the user and the camera sensor should be around 30-40 cm to match the size of the faces of the training. This camera provides a  $640 \times 480$  pixels frame resolution, while recording at 30 FPS.

For development and training purposes we will use some pre-existing emotion datasets, in order to validate the efficiency of the system before testing it in real conditions.

## 2.3 Facial Expression Datasets

Databases are very important for facial expression recognition system.



**Figure 2.2:** Facial expression recognition system setup

Using the same databases as in previous studies allows performance and accuracy comparisons between new implementations and previously obtained results. Since most studies draw their results on the same databases, it is then relatively easy to compare them and choose a database suitable for our system.

Moreover, databases are difficult to build. Indeed, it has to be obtained following a meticulous procedure while being exhaustive so it can be considered as representative. The majority of actual databases use *posed* expressions rather than *spontaneous* ones, this choice having a major influence on facial expression recognition systems. This explains why some databases are updated, and now integrate spontaneous expressions. Even with this transition from posed expressions to spontaneous expressions, there are other requirements that should be met to have a standardized database. Its content should be of different resolutions and scales, and should also contain exposition under different conditions, i.e changes in lighting, occlusions or different head angles [4].

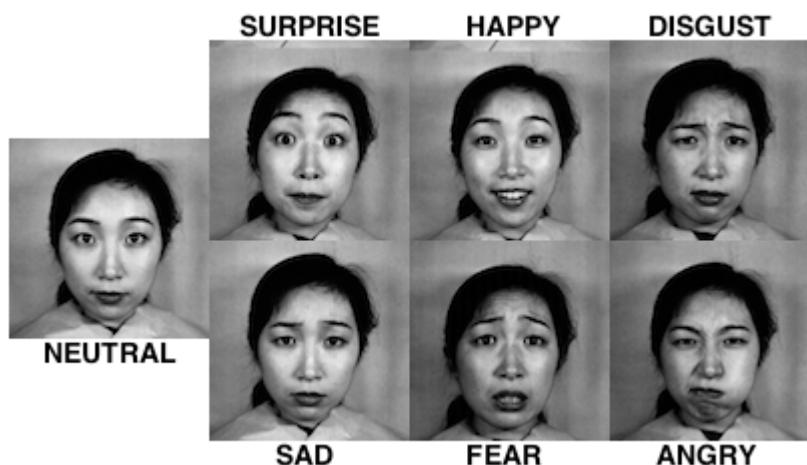
Constructing a database is then a tedious task because of all these requirements to meet. Consequently, most studies are based on already existing datasets. The 4 datasets described afterwards are popular and freely available facial expression datasets which have been used a numerous amount of times in the past few years. Our system will then be trained and tested with one or several of these databases.

### 2.3.1 Japanese Female Facial Expression Database (JAFFE)

This database contains 213 images of 7 facial expressions (6 basic facial expressions: happy, angry, afraid, disgusted, sad, surprised, and 1 neutral facial expression). Each expression has been photographed three or four times. Each image has been rated on 6 emotion adjectives by 60 Japanese subjects. All images come from 10 Japanese female models. The database was planned and assembled by Miyuki Kamachi, Michael Lyons, and Jiro Gyoba [23].

This database contains only posed expressions. Photos have been taken under strict and controlled conditions: similar lighting and hair tied so there is no facial occlusion [4].

An example of images contained in the database is given by Figure 2.3. In this figure, this is a female subject displaying 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, surprised).



**Figure 2.3:** Example of images from JAFFE database

### 2.3.2 Karolinska Directed Emotional Faces Database (KDEF)

The Karolinska Directed Emotional Faces (KDEF) contains 4900 pictures of human facial expressions. The material was developed in 1998 by Daniel Lundqvist, Anders

Flykt and Professor Arne Ohman at Karolinska Institutet, Department of Clinical Neuroscience, Section of Psychology, Stockholm, Sweden [20].

The database was first developed for psychological and medical research purposes. It was created so it could be used in perception, attention, emotion, memory and backward masking experiments. This database has also been built under controlled conditions. Indeed, researchers tried to maintain constant and soft lighting for each subject. Furthermore, they made their subjects wear the same grey T-shirts, and used a grid to center the participants faces while they were shot. This grid was also used to place eyes and mouths at the same position in fixed image coordinates during scanning [20].

The database contains 70 individuals (35 males and 35 females), from 20 to 30 years, each one displaying 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, surprised). Each expression has been photographed twice for each subject from 5 different angles (-90, -45, 0, +45, +90 degrees: i.e. full left profile, half left profile, straight, half right profile, full right profile) [20].

An example of images contained in the database is given in Figure 2.4 which represents a female subject photographed from a straight angle and displaying 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, surprised).



**Figure 2.4:** Example of images from KDEF database

### 2.3.3 Montreal Set of Facial Displays of Emotion Database (MSFDE)

This database contains facial expressions of European, Asian, and African subjects, from both genders. Each expression was created by directly asking the subject to express this emotion [32].

The database contains expressions of happiness, sadness, anger, fear, disgust, and embarrassment, along with a neutral facial expression. All expressions have been photographed at 5 different levels of intensity [32].

An example of images contained in the database is given in Figure 2.5, where an African female subject displays 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, ashamed).



**Figure 2.5:** Example of images from MSFDE database

### 2.3.4 Cohn-Kanade AU-Coded Facial Expression Database (CK)

The Cohn-Kanade AU-Coded Facial Expression Database was created in 2000 for research purposes in the area of automatic facial image analysis. Two versions of this database are at the moment published [19].

The first version is the original release of the database. It is composed of 486 image sequences of 97 subjects. The database is FACS coded. For each sequence, the initial image is the neutral expression and the final image is a peak expression. For each emotion, a label is given, which refers to the facial expression that is supposed to be expressed but not necessarily the one that is really expressed. This is the difference between the first and second version, since the second version uses validated emotion

labels, so the expressed emotion is the same as the labeled one [19].

The second version called *CK+*, is composed of both posed and spontaneous expressions. Concerning posed expressions, the database has an increased number of image sequences of 22%, and contains images from more subjects (27% augmentation). An example of images contained in the database is given in Figure 2.6, where a female subject expresses the *disgusted* emotion. Figure 2.6 shows all the steps of an image sequence for the *disgusted* emotion [19].



**Figure 2.6:** Example of images from CK database (starting from the left top corner to the right bottom corner) (©Jeffrey Cohn)

## 2.4 Issues

### 2.4.1 Datasets

Databases can be a source of issues. As said previously, databases should meet a number of requirements in order to be as exhaustive and efficient as possible.

In order to build a facial expression recognition system efficient in live conditions, it should be able to recognize spontaneous expressions rather than posed expressions. Indeed, spontaneous expressions are closer to reality than posed expressions, the latter being exaggerated to facilitate their labelling and recognition. While creating a database of spontaneous expressions, Sebe and al [29] made some observations of the major problems they encountered [4]:

- The same emotions can be expressed at different intensities by different subjects;
- As soon as the subject is aware of being photographed and studied, the authenticity of the emotion is lost;
- Because of laboratory conditions, even if the subject is not aware of being photographed or recorded, the subject is not encouraged to display spontaneous expressions.

In order to overcome these problems, they came up with a method. Their solution was to record facial expressions with a camera hidden in a video kiosk displaying emotion inducing videos. Subjects were notified of the recording after it was done, and were asked a permission to use recorded sequences for research studies. The subjects then explained which emotions they felt and expressed, their replies being documented even if it did not match the recorded expressions [29].

The researchers found that a wide range of expressions are hard to induce, particularly fear and sadness. They also found that spontaneous expressions could be misleading: some subjects expressed one emotion while feeling another one (for example, one subject was showing sadness while being happy) [29].

In a nutshell, databases bring some issues that can affect the authenticity of the recognition system. It depends of the type of expressions: spontaneous or posed expressions. If the system aims to recognize facial expressions of people unaware of it, spontaneous expressions databases will be used but, as seen previously, it can lead to authenticity issues. If the system aims to recognize facial expressions of people asked to express certain emotions, posed expressions databases will be used, but the result will not be close to reality.

#### 2.4.2 Real-time

The primary goal of the facial expression recognition system described in this paper is to perform real-time facial expression recognition. As a real-time application, processing time should be taken into account. Indeed, if it is too long, the system will not be responsive enough to be categorized as real-time.

This is one of the challenges of this kind of system, because processing is usually really heavy no matter which algorithm is used. Most facial expression recognition applications should be working in real-time conditions, for example in robotics or in surveillance. A solution could be to find new algorithms for facial expression recog-

nition or to improve and lighten already existing algorithms.

### 2.4.3 Conditions

Another one of the challenges of this kind of system is to be independent of recording conditions. It means that the recognition should not be disturbed for example by occlusions, or difference in the lighting, or even by the angle between the face and the camera. These examples cover almost all conditions that can change during the recording, and have an influence on the recognition system.

#### Occlusion

"Occlusion" defines all elements covering the face, partly or entirely. For example, a beard, a scarf masking the bottom of the face, glasses or bangs. By hiding a part of the face, these occlusions can affect the recognition. Indeed, facial expression recognition systems are based on features comparison, and if all features cannot be compared because something is covering a part of the face, recognition accuracy is affected. In order to compensate for this problem, some databases includes data with occluded faces, for example with beards, glasses or scarves. This is the case for the AR Face database [24]. Some examples of images as the ones that could be find in this database are given by Figures 2.7 and 2.8.



**Figure 2.7:** Example eye occlusion as the AR Face database would have

#### Lighting

As for occlusion, lighting is an element that can affect recognition effectiveness. With different lighting conditions from those in the database, recognition will be less efficient. All conditions different from those recorded in the database on which the



**Figure 2.8:** Example of face occlusion as the AR Face database would have

recognition process is based will impact the system. If images are brighter or darker than reference data, some details can disappear, or some features will not be recognized as well as if the conditions were identical. In order to compensate for this problem, databases can include data with different lighting conditions. This is the case for the AR Face database [24] and an example of images as the one that could be contained in this database is given by Figure 2.9.



**Figure 2.9:** Example of different lighting conditions (from left to right: dark to bright) as the AR Face database would have

## Angle

Head angle is one of the most impacting conditions during recognition. Indeed, some features can be occluded or disappear if the head does not properly face the camera. This issue also rises if the system is not tuned to work with head angles other than straight profile. For example, with a profile angle, one eye, half of the nose and of the mouth disappear. If the database does not contain samples with profile faces, recognition will fail. However, if the database contains images from straight angle as well as from different profile angles, recognition will be possible. An example of different images taken from different angles for one emotion *neutral* is given in Figure 2.10:



**Figure 2.10:** Example of different angles for the *neutral* state (from left to right: full left profile, half left profile, straight, half right profile, full right profile)

## 2.5 Requirements

Based on everything said previously, our facial expression recognition system can be defined by some requirements prior to its implementation. Additional requirements may be defined further. Here are the requirements already defined :

- Able to recognize basic emotions : As explained before, facial expression recognition systems are able to recognize 6 basic emotions and the neutral state. This system should be able to do the same: recognize the 6 basic emotion that are *Happiness*, *Fear*, *Surprise*, *Disgust*, *Sadness*, *Anger* and the *Neutral* state.
- Able to work in real-time : This system should be able to recognize facial expressions in real-time. It means that it can recognize expressions based on video sequences. It also means that the algorithm for feature extraction has to be carefully optimized so it is able to compute facial features in a decent amount of time.
- Recognition from straight angle of the face : This system should be able to recognize facial expression from a straight angle of the face. It means that the system should be able to detect faces in front of the camera lens, and recognize expressions in these faces. It might not be able to recognize emotions on a face on a profile or half-profile angle.
- Recognition with no occlusion : This system should be able to recognize emotions with no occlusion on the subject's face. It means that the face should not

be covered in any way: no glasses, no beard or no scarf. The face should also be complete, not cut and not masked.

- Recognition with no changes in lighting : This system should be able to recognize emotions under constant lighting conditions during recognition process. Moreover, the intensity level of the light should be as close as possible to the one of the database. This way lighting would not have any influence on the recognition process.

**Part II**

**Face detection**

# Contents

*Before getting to the main part of this project which is Feature extraction, there is a mandatory step that is Face detection. In order to avoid as much computation and processing as possible, only parts containing regions of interest for a facial expression recognition system have to be processed. It consequently means that detection of interesting features has to be performed beforehand, which is the goal of face detection. This part will explain how face detection works in general. Then Viola-Jones algorithm, a performant and cost-effective method for face detection and tracking, will be introduced.*

<b>3</b>	<b>Face detection</b>	<b>21</b>
<b>4</b>	<b>Viola-Jones Face detection</b>	<b>23</b>
4.1	Overview	23
4.2	Haar features	23
4.3	Integral image	28
4.4	Weak classifiers and AdaBoost	29
4.5	Classifiers cascade	32
4.6	Test set and training	34

# Chapter 3

## Face detection

Face detection is the first step following image acquisition, and is compulsory for facial expression recognition. At the end of this step, only useful information will remain. It will not only reduce processing time during the feature extraction step, but will also improve the classifier by discarding useless information.

Detection is about finding out if the input image or video sequence represents or contains a particular object, and is usually followed by recognition. In our system, facial expression recognition will be performed. However, depending on the recognition, an additional tracking step can be needed. Tracking consists in following a moving target along the images of a video sequence [9].

An object detector can be defined as a "black box" taking an image as input. Its input depends on the type of detector, whether it is high level or low level. At a high level the output can be considered as an annotated image saying where the object of interest appears [9]. For example, the output can look like Figure 3.1 [9].



**Figure 3.1:** Example of an output of face detection [9]

At a low level, the output is not an annotated image anymore. The core of the object detector is a basic component saying if an instance of the object of interest is contained in a certain region or sub-region of the original image. This kind of detection is performed by a binary classifier [9]. There is an example of a binary classifier output in Figure 3.2 [9].



**Figure 3.2:** Example of a binary classifier output for face detection [9]

Classification aims to solve the problem of identifying in a set of categories or sub-populations to which a new observation belongs. It is based on a training set of data that contains instances whose category affiliations are known. Data can be labeled based on some measures of inherent similarity; for example, vectors representing the distance between instances [36].

# Chapter 4

## Viola-Jones Face detection

Viola-Jones algorithm is "a visual object detection framework that is capable of processing images extremely rapidly while achieving high detection rates". 3 main points characterize this algorithm. The first one is the use of what is called an "integral image", which is a new representation of the image allowing the features to be computed very quickly. Second point is the use of a learning algorithm based on AdaBoost, which builds efficient classifiers. The third and last point is the use of a method to combine classifiers in cascade. It allows to focus on the promising object-like regions by discarding the background in a very quick way [35].

### 4.1 Overview

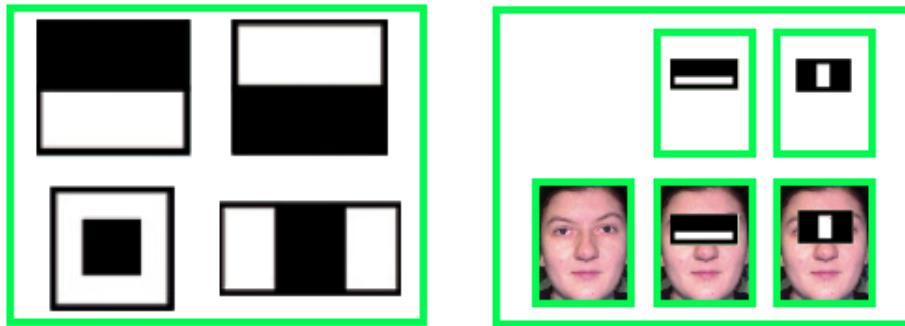
The Viola-Jones algorithm works as following [9]:

- This algorithm is a "strong, binary classifier built of several weak detectors"
- During the learning part, a cascade of weak classifiers is used and trained in order to reach the desired hit/miss rate using AdaBoost learning algorithm
- The input image is then divided into several rectangular sub-regions in order to detect objects. Each sub-region is computed by the cascade of classifiers;
- To classify a sub-region as positive, it has to pass all stages of the cascade
- The algorithm iterates over the 2 last steps with sub-regions of different sizes.

### 4.2 Haar features

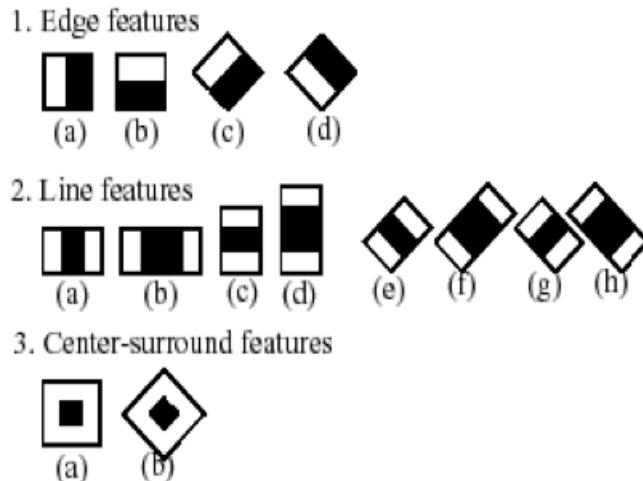
Features used by Viola and Jones are based on Haar wavelets, which are single wavelength square waves. A square wave is composed of a high interval and a low interval. In a two dimensional space, a square wave is represented by a pair of adjacent rectangles: a light one and a dark one. True Haar wavelets are however not used in rectangle combinations for visual object detection. There are better suited features for this task, which are Haar-like, hence the name of Haar (or Haar-like) features,

instead of Haar wavelets [16].



**Figure 4.1:** Example of Haar features

Figure 4.1 shows some of the first Haar features in the original Viola-Jones cascade [16], while Figure 4.2 is an example from the extended set of features [9]. Figure 4.3 is an example of an early stage in the Haar cascade. In these examples, each pair of black and white rectangles represents a feature. These features are used by the algorithm to detect regions of interest in the input image [14].



**Figure 4.2:** Extended set of features [9]

Basic subtraction is used to detect the presence of Haar features. It consists of subtracting the average pixel value of the dark region from the average pixel value of

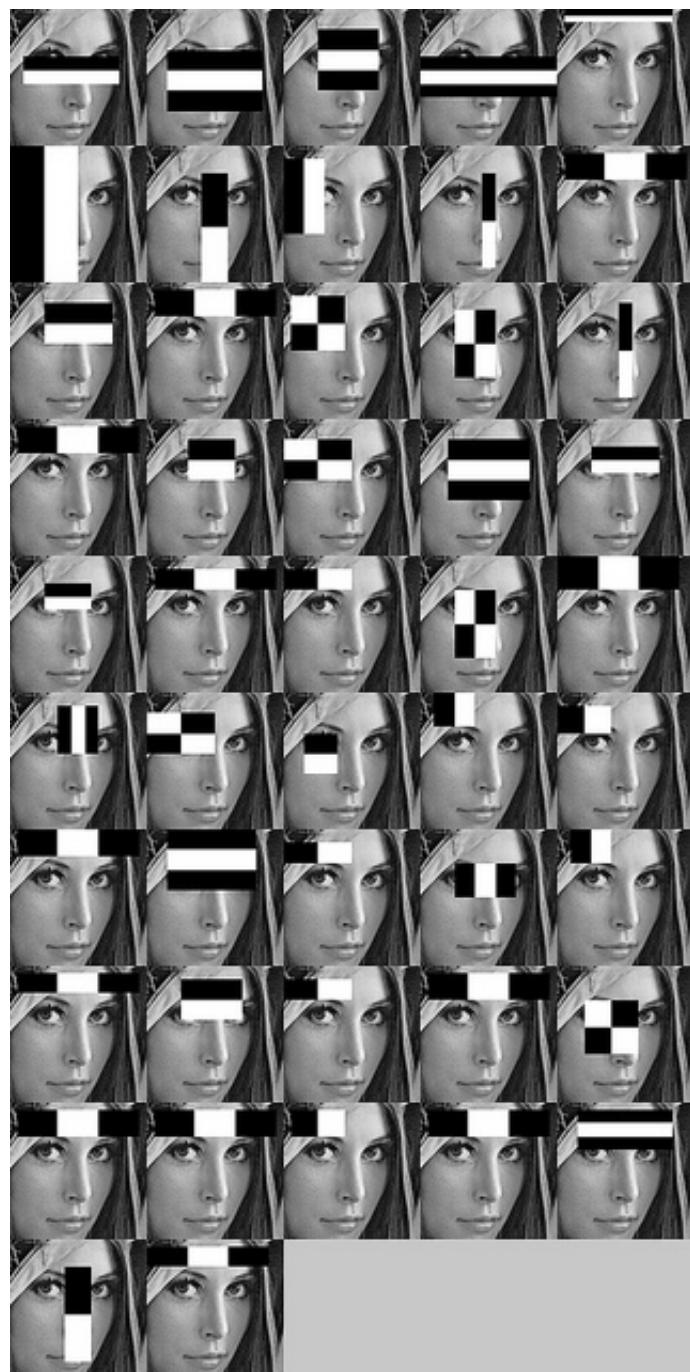


**Figure 4.3:** Example of an early stage in the Haar cascade [14]

the light region. There is then a thresholding of the result, where the feature is considered present if it is above the threshold. If the outcome is positive, the current stage is validated, and the feature can go on the next stage [16]. There are about 20 to 30 different stages in order to detect the presence of Haar features. The first stage is a very coarse scan of the image, while following stages are more finely tuned and harder to pass [14].

For example, Figure 4.4 shows a later stage in the Haar cascade. Compared to an earlier stage as in Figure 4.3, there are many more patterns of black and white rectangles that need to match the input image [14].

There are 3 kinds of features used by Viola-Jones' algorithm: a two-rectangle feature, a three-rectangle feature and a four-rectangle feature, as seen in Figure 4.5. Indeed, images (A) and (B) show the two-rectangle features, image (C) shows the three-rectangle feature, and image (D) shows the four-rectangle feature [35].

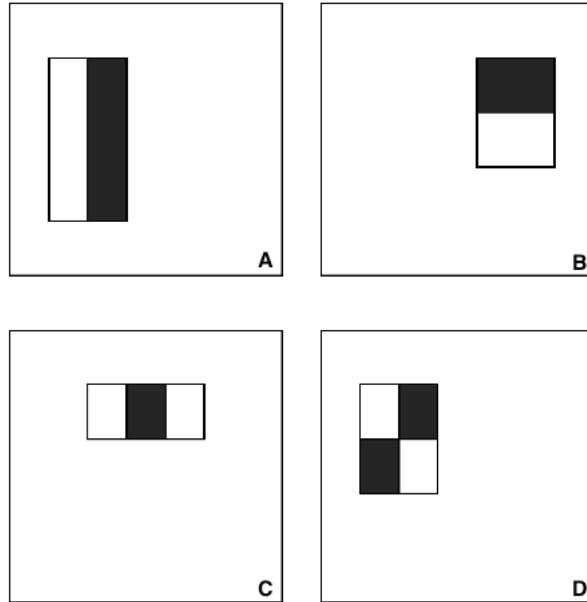


**Figure 4.4:** Example of a later stage in the Haar cascade [14]

For two-rectangle features, the output value is computed by the difference between the sum of the pixels being in the two rectangular regions. The 2 regions are identical, with same size and same shape, and they are horizontally or vertically adjacent.

The three-rectangle feature value is calculated by the sum of the pixels of the two outside rectangles subtracted from the sum of the pixels in the central rectangle.

The last kind of feature is the four-rectangle feature, which value consists in the difference between the diagonal pairs of rectangles [35].



**Figure 4.5:** Example of the different kinds of rectangle features

Viola and Jones admit that rectangle features can be considered as primitive features. In contrast with other features, rectangle features are quite coarse, even though they are sensitive to the presence of edges, bars and other simple image structure. It however appears that a rich image representation is provided by this set of rectangle features, and this representation supports effective learning. Compared to the great computational efficiency provided by rectangle features, their limited flexibility is then not much of a problem [35].

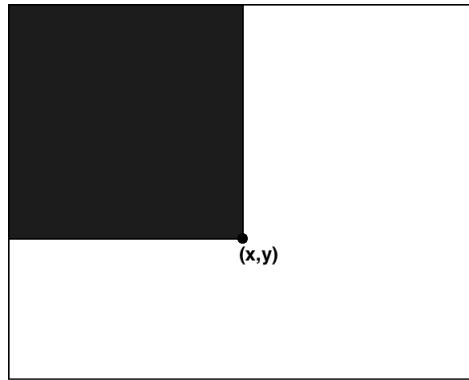
### 4.3 Integral image

Viola-Jones algorithm uses an intermediate representation of an image called "integral image". This integral image allows a very fast computation of rectangle features [35], which then influences the determination of the presence or absence of hundreds of Haar features. In general, adding small units together is similar to integrating them; here, the small units are pixel intensity values. The integral value of a pixel can hence be computed by summing values from pixels above it and on its left. Integration of an entire image consequently starts at the top left corner and goes through all the image to the down right corner [16].

For a pixel  $P$  with coordinates  $(x, y)$ , its value in the integral image is computed as follows:

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

where  $ii(x, y)$  is the integral image and  $i(x, y)$  is the input image intensity value function [35]. In Figure 4.6, the value of the integral image at point  $(x, y)$  is represented as the sum of all pixels above and on its left.



**Figure 4.6:** Integral image

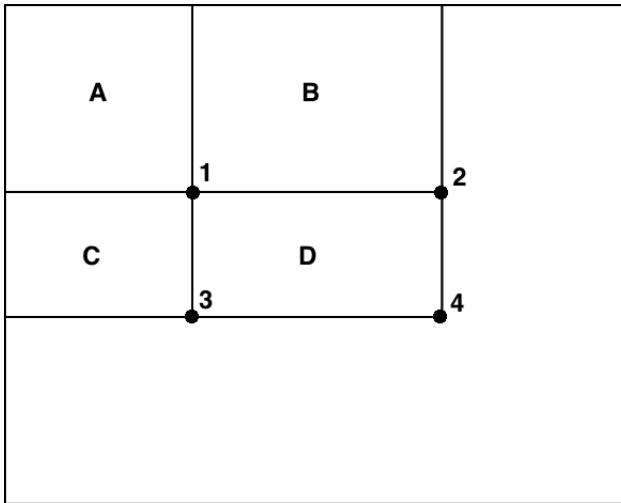
Using the following pair of recurrences:

$$s(x, y) = s(x, y - 1) + i(x, y) \quad (4.1)$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \quad (4.2)$$

(where  $s(x, y)$  is the cumulative row sum,  $s(x, -1) = 0$ , and  $ii(-1, y) = 0$ ) the integral image can be computed in only one pass over the original image.

In Figure 4.7, the sum of the pixels in rectangle D can be calculated with four array references, thanks to the integral image. Value of the integral image at point 1 is the sum of pixels in rectangle A. Value at point 2 is  $A + B$ , at point 3 is  $A + C$ , and at point 4 is  $A + B + C + D$ . The sum in D can then be computed as  $4 + 1 - (2 + 3)$ .



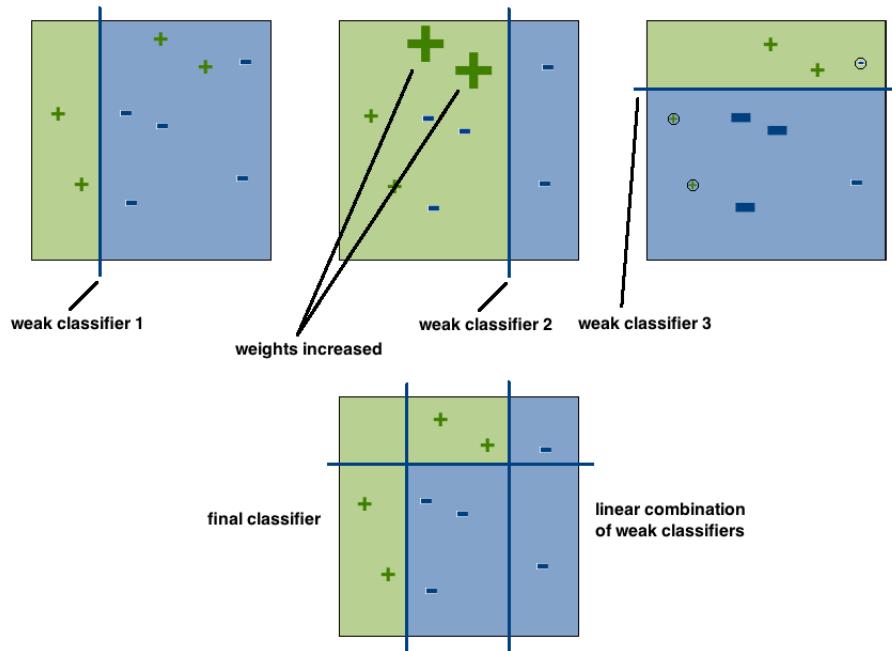
**Figure 4.7:** Integral image with four array references

## 4.4 Weak classifiers and AdaBoost

Features are extracted from a sub-region of an input image, which size is usually of  $24 \times 24$  pixels. Each feature type is moved and scaled across the entire input image, which means that there are about 160,000 possible combinations to process in a 24 pixel by 24 pixel sub-region [31].

AdaBoost is a machine-learning method used by Viola and Jones in order to select specific Haar features to use. It is also used to set the threshold levels. This method is based on the statement that combination of many weak classifiers forms a strong one. They are called weak classifiers because their accuracy and efficiency are only a bit above random guessing, which is not particularly good. The purpose of using so many weak classifiers is to get a right answer with a higher rate of success. This algorithm relies on the verified hypothesis that if each weak classifier pushes the final answer a little bit in the right direction each time, it means the correct answer will be obtained in the end [16].

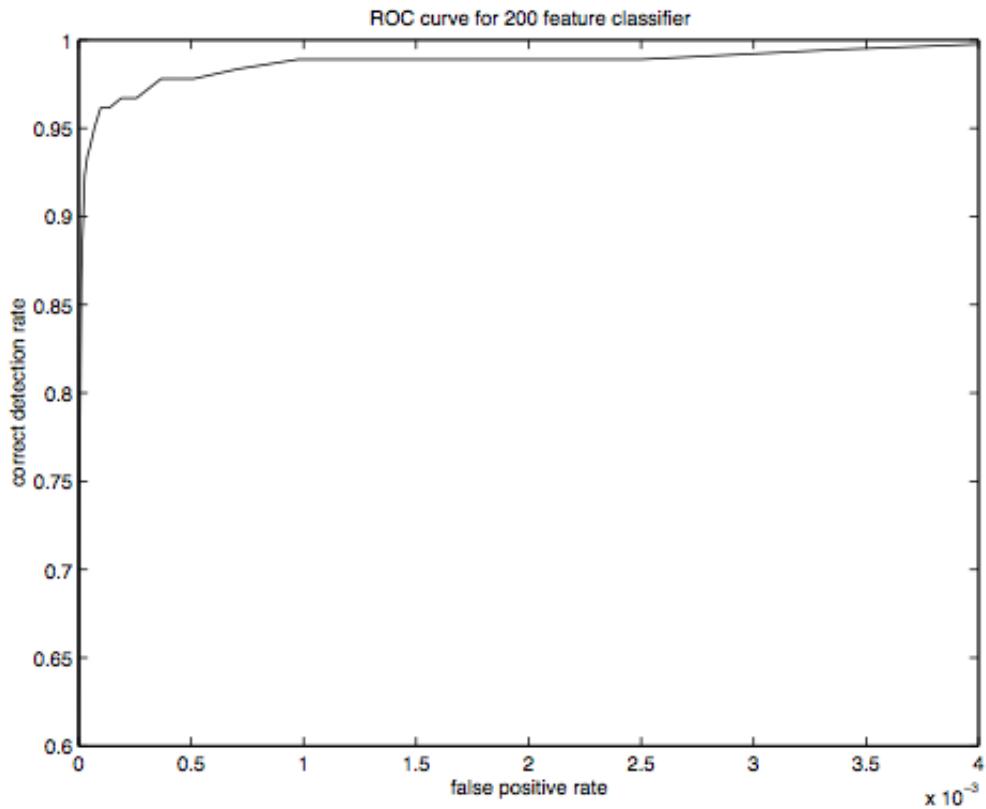
AdaBoost works the following way: it chooses a set of weak classifiers that are going to be combined and assigns a weight to each classifier (see Figure 4.8). The result of this weighted combination is a strong classifier [16]. One of the difficulties and challenges for this learning algorithm is to associate large weights to efficient classifiers, and smaller weights to poor classifiers. In order to succeed in selecting a small group of good classifiers but with significant variety, AdaBoost is an aggressive algorithm [35].



**Figure 4.8:** AdaBoost method

In [35], Viola and Jones conduct experiments with a classifier built from 200 features and using AdaBoost. This experiment was a first step in their study, since they compared it afterwards with further experiments. The classifier had a detection rate of 95%, and obtained only 1 false positive out of 14084 negative samples from the training dataset (see figure 4.9). Afterwards, Viola and Jones conducted another experiment using a *classifiers cascade*, and compared the latter results with the ones described here.

Their first experiment pointed out that a 200-feature classifier is an efficient technique for face detection. It also meant that a boosted classifier constructed from rectangle features is an efficient technique for face detection. Although the results

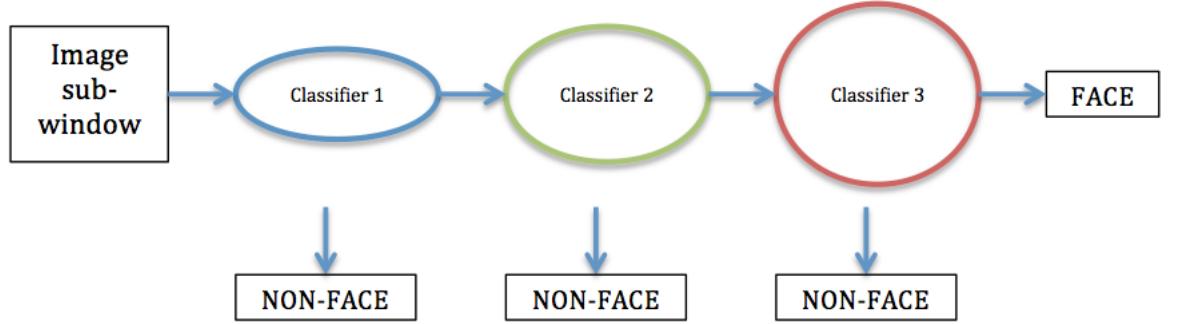


**Figure 4.9:** Receiver Operating Characteristic (ROC) curve for the 200 feature classifier [35]

of this experiment were convincing in terms of detection, they may not be performant enough for everyday applications. This boosted classifier required 0.7 seconds to scan a  $384 \times 288$  pixel image. Regarding computation time, it is probably faster than any other existing system. In order to improve the system so it will perform well in real-world conditions, detection performance must be improved. The most straightforward method to achieve that is to add features to the classifier. Doing that will immediately decrease the speed of this system while increasing computation time. This is why they implemented a method called *classifiers cascade* in order to improve the computation time without decreasing the accuracy of the algorithm [35].

## 4.5 Classifiers cascade

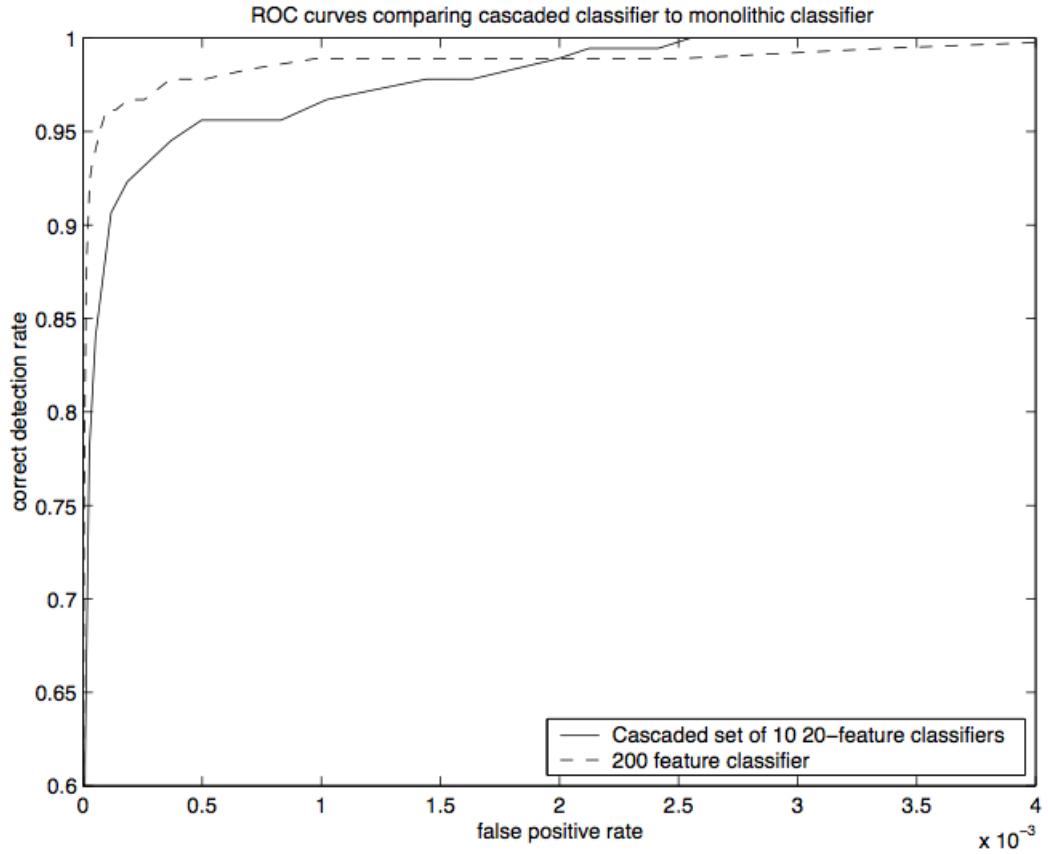
What Viola and Jones did to classify image regions and sub-regions in an efficient way is to combine AdaBoost classifiers as a filter chain. It is constructed as a cascade, hence its name *Classifiers cascade*. This chain is composed of a separate AdaBoost classifier for each filter, which has a fairly small number of weak classifiers. As in figure 4.10, the classifier cascade represents a chain of filters. If an image sub-region passes successfully all cascade steps, it is labelled as *Face*, otherwise it is classified as *Not Face* [16]. Using this algorithm with the classifiers cascade method allows to reduce significantly the computation time and increase significantly the detection performance [35].



**Figure 4.10:** Cascade of boosted classifiers

In [35], Viola and Jones conduct tests to see if the cascade method was feasible. Two simple detectors were trained, one of them being a 200-feature classifier and the other one being a cascade of 10 20-feature classifiers. Figure 4.11 gives the ROC curves comparing the two classifiers' performance. Between the two classifiers, regarding their accuracy, the difference is little and not significant. On the other hand, regarding their speed, the difference is large and significant, the classifier cascade being about 10 times faster. This is because as soon as the first stage, most negative samples are discarded, so they will not be evaluated ever again afterwards [35].

The cascade method has been made in a way that there must not be false negative labeling, meaning that no face should be classified as *Not Face*. The assistance threshold has been set low for each level, low enough for the training set to pass all or almost all face examples. All training images that passed previous stages are classified by filters trained to do it for each level. A region is immediately classified as *Not Face* as soon as it failed at one filter. If one of these filters succeeds to pass



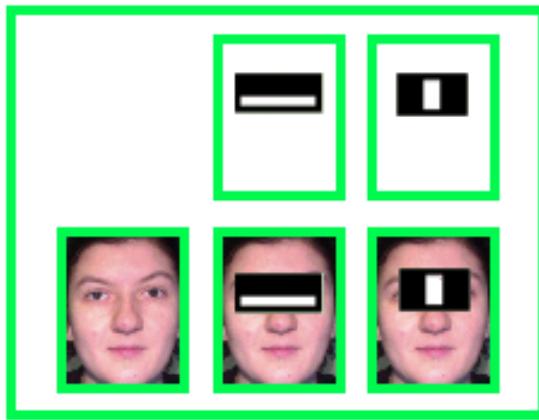
**Figure 4.11:** ROC curves of a 200-feature classifier and of a classifier cascade containing 10 20-feature classifiers [35]

a region, then it is up to the next filter in the chain. If a region succeed to pass through all the filters present in the chain, then it can be classified as *Face* [16].

The key point of this method is to construct smaller, more efficient, boosted classifiers. They will then detect nearly all positive instances while rejecting a lot of negative sub-regions. Before using complex classifiers to achieve low false positive rates, simple classifiers are called to reject most sub-regions [35].

The order of the filters in the cascade is not random; it is based on the importance weighting assigned by AdaBoost. Heavily weighed are called early, so they eliminate non-face sub-regions as soon as possible. Figure 4.12 shows the first 2 features from Viola-Jones cascade, applied to a face. The first feature used is the one with the eye region being darker than the cheek region. The second feature used is the one with

the eyes region being darker than the bridge of the nose [16].



**Figure 4.12:** The first two Haar features in the original Viola-Jones cascade

The structure of the cascade in itself means that within any single image, most sub-regions are negative. The cascade hence tries to reject as many negatives as possible, starting at the first stage. Indeed, when a positive instance occurs, it will trigger the evaluation of all the classifiers of the cascade, which can be time consuming [35].

Following are the different numbers about cascade classifiers that Viola and Jones obtained with their algorithm (see Figure 4.13) [25]:

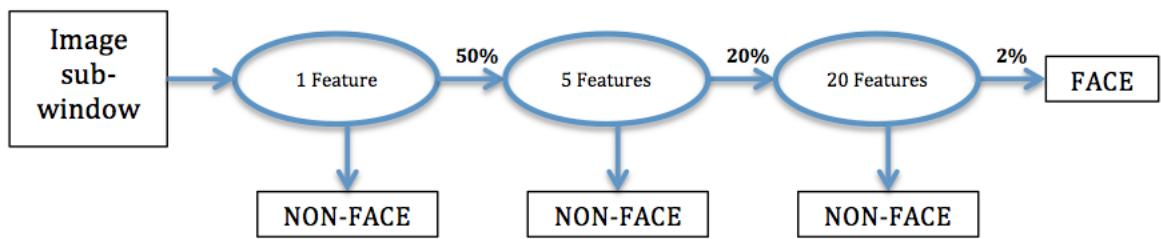
- 1-feature classifier: 100% detection rate and 50% false positive rate
- 5-features classifier: 100% detection rate and 40% false positive rate
- 20-features classifier: 100% detection rate and 10% false positive rate

## 4.6 Test set and training

The training set is made of about 5000 hand-labelled images of faces. All faces are scaled and have the same resolution of  $24 \times 24$  pixels. All these faces were chosen randomly on the internet. Some face examples are shown in figure 4.14 [35].

To sum up, the training set is composed of [25]:

- about 5,000 faces



**Figure 4.13:** Cascade of boosted classifiers positive rate



**Figure 4.14:** Examples of frontal upright face images used for training [35]

- All frontal
- 300 million non faces sub-regions

- from 9,400 non-face images
- Normalized faces
  - Scale, translation
- Many variations
  - Between individuals
  - Lightning
  - Pose (rotation of the head)

When testing their algorithm, Viola and Jones typically had a training set made of about 5,000 positives samples (faces) and 10,000 negative samples (non-faces, usually non-face sub-regions chosen from non-face images) [9]. For this kind of training, and with a 32 layer classifier, the total time usually exceeded several weeks [35].

Viola-Jones training stage proceeded with the following steps [9]:

- With a defined number  $K$  of features (about 160,000 for a  $24 \times 24$  grayscale image)
- The number  $L$  of wanted stages had to be fixed for the cascade
- Then it has to be iterated until number  $L$  weak classifiers was reached:
  - With data that had been weighted again in the previous stage
  - all  $K$  of weak classifiers had to be trained (the best threshold had to be found to classify the training set in the best way)
  - The best classifier had to be chosen at this stage
  - The data was then weighted again

As said previously, depending on the efficiency of a weak classifier, a weight was associated to it, depending on its classification error rate. Weak classifiers were then linearly combined depending on these weights, this operation having a huge computational cost [9].

# **Part III**

## **Feature extraction**

# Contents

*This part will focus on the step following face detection, which is feature extraction. Chapter 5 will present the main issues on feature extraction, and usual feature extraction methods, while the following chapter will focus on the Local Binary Patterns algorithm.*

<b>5 Feature extraction</b>	<b>39</b>
5.1 Appearance-based methods	40
5.2 Feature-based methods	40
<b>6 Local Binary Patterns</b>	<b>43</b>
6.1 Overview	43
6.2 State of the art	44
6.3 Facial Expression Recognition based on LBP	48
6.4 Histogram computing	48

# Chapter 5

## Feature extraction

After having detected the face, for example using Viola-Jones algorithm, it is necessary to perform feature extraction in order to process data before classification. This step takes an image as input, and extracts vectors characterizing its main features. In the case of a facial expression recognition system, resulting vectors contain information about spatial configuration of facial features, but can also encode information about shape, texture or movement of the image's content [7].

There are however many kinds of algorithms outputting features vectors, and the choice of an effective one depends on many criteria. These feature extraction methods can generally be ranked among 2 categories: they can either be *appearance-based* or *feature-based*, depending on the way they extract feature vectors. The aim of this chapter is to explain the differences between these 2 types of feature extraction methods, and provide some examples from each category.

Before developing a facial expression recognition project, it is important to know what the state of the art of facial expression recognition systems. In this chapter, an overview will be given of the existing systems before to decide on a feature extraction system for the project.

Two main categories of feature extraction algorithms can be distinguished : *appearance-based* or *feature-based*. The first ones are algorithms that try to find basic vectors characterizing the whole picture, usually by a dimensionality reduction method. These algorithms lead to a simplification of the dataset, while retaining the main characteristics of the picture. However, these methods have to be carefully parametrized, so they do not encounter the "curse of dimensionality", which is about processing high-dimensional data.

Examples of appearance-based methods : Principal Component Analysis, Linear Discriminant Analysis.

The second type of feature extraction algorithms are feature-based algorithms. These methods tend to locate important features, and build feature vectors depending on those regions of interest. The key point of these methods is that the face is not a global structure anymore. Indeed, it has been summarized in a set of feature regions,

which are themselves translated into feature vectors.

Examples of feature-based methods : Gabor Wavelets, Local Binary Patterns.

## 5.1 Appearance-based methods

As said in the introduction, appearance-based methods aim to extract appearance changes of the face. In order to do that, methods use image filters applied to the whole face or applied to specific parts of the face [30]. With this kind of methods, if there are any changes in lighting or head pose, recognition will be less effective.

### 5.1.1 Principal Component Analysis (PCA)

This is a statistical method; one of the most used in linear algebra. PCA is mainly used to reduce high dimensionality of data and to obtain the most important information out of it. PCA computes a covariance matrix and a set of values called the eigenvalues and eigenvectors from the original data [12]. Its output is a new coordinate system with lower dimensions, obtained from transformed high dimensionality of data, while preserving the most important information.

### 5.1.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis is also a statistical method, used to classify a set of objects into groups. It is done by looking at a specific set of features describing the objects. LDA as PCA are used to establish a linear relationship between the dimensions of the data. LDA uses this relationship to model the differences into classes, while PCA does not take any differences into account in the linear relationship. The idea behind LDA is to perform a linear transformation on the data to obtain a lower dimensional set of features [12].

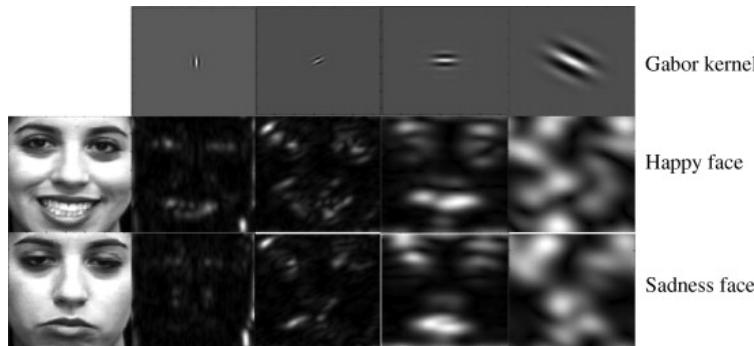
## 5.2 Feature-based methods

As said in the introduction, feature-based methods aim to locate important facial components. Shape and location of these facial features are extracted from a face image and are then concatenated into a feature vector representing the geometry of the face [30].

Compared to appearance-based methods, feature-based methods have equal or better results. This was recently tested by Valstar et al. [34] [33].

### 5.2.1 Gabor Wavelets

A feature-based method generally used is the Gabor wavelets because of its performance. Gabor filters are applied in order to extract a set of Gabor wavelet coefficients. Filter responses are obtained when Gabor filters are convolved with face image (see Figure 5.1). These representations of face display locality and orientation performance [17]. Indeed, Gabor kernels represent features of the face and can be placed wherever it fits on the face and with whatever orientation. A disadvantage of this method is that it needs reliable and accurate feature detection and tracking if necessary [30]. Another disadvantage of Gabor feature extraction is processing time. This algorithm is very time-consuming, and dimensions of resulting vectors are very large [28].



**Figure 5.1:** Examples of Gabor wavelets and corresponding convoluted images

### 5.2.2 Local Binary Patterns (LBP)

Local Binary Patterns is a feature-based method. Its first application was to describe texture and shape of an image by extracting informations from the neighborhood of a central pixel [26]. This information is the output of the thresholding of intensity values from neighborhood pixels with intensity value of the central pixel [12]. One of the advantages of this method is that it is effective even when there is change in lighting (see Figure 5.2).



**Figure 5.2:** Effect of changes in lighting with LBP method

This method is more detailed in Chapter 6. This is the method that is used in this particular Facial Expression Recognition system. The choice has been made on this method because it is particularly effective and its basic implementation can be modified in many ways to improve feature extraction. This method was chosen over Gabor wavelets because they are known to be time consuming, and this facial expression recognition system is wanted to be as close as possible to a real-time system.

# Chapter 6

## Local Binary Patterns

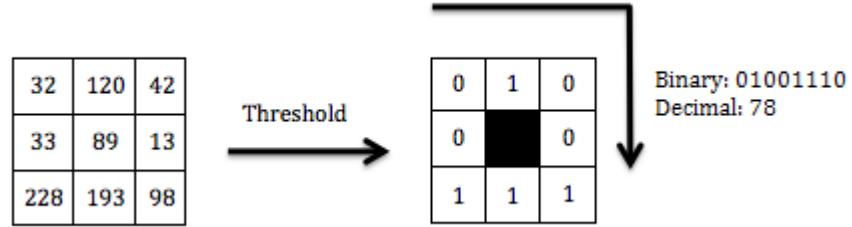
Some feature extraction methods are widely used and studied to characterize and describe the face, such as Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). These methods describe the whole image, and are not very efficient if there are changes in lighting or head pose. That is why some researchers turned to local descriptors. These local descriptors describe the face by characterizing parts of the face depending on their importance. The Local Binary Patterns (LBP) feature extraction is a widely used local descriptor [1].

The LBP operator is known to be an effective texture descriptor because it efficiently describes micro-patterns. Since a face can be seen as a composition of micro-patterns, it is logical to use this texture descriptor. It has been introduced in 1996 by Ojala et al. [26]. This operator has a lot of advantages, one of them being its highly discriminative rate. Other advantages are its invariance to gray-level changes and its computation efficiency, which makes it suitable for image analysis but may not be efficient enough for real-time analysis [1].

### 6.1 Overview

Globally, a gray-scale image of a face is divided into small regions. LBP histograms are extracted from each of those small regions. These histograms are then concatenated into one feature vector describing the image [18].

The LBP operator works with one central pixel and its eight neighbour pixels. It is a basic binary thresholding between the central pixel and its neighbours. The threshold is set as the intensity value of the central pixel. Then, for each neighbour pixel, if its value is greater or equal to the threshold, a value of 1 is assigned, otherwise a 0. When all neighbours have been processed, a LBP code for the central pixel can be obtained by concatenating all values of its eight neighbours into a binary code. For a better comprehension, a human readable, decimal value can be computed from the binary one. Figure 6.1 shows an example of the LBP process for a pixel and its eight neighbor pixels [18].



**Figure 6.1:** Basic LBP operator

Figure 6.2 shows an example of the LBP operator applied on a facial image from the JAFFE database [22].



**Figure 6.2:** Example of the LBP operator applied on a image from the JAFFE database

## 6.2 State of the art

### 6.2.1 Circular LBP

In order to be able to use the LBP operator at different scales, a particular form of the operator has been introduced, which is circular LBP operator. This way, the operator can be extended to other neighbourhoods with different sizes [12].

This operator still compares the intensity value of the central pixel with its neighbors, but neighbors pixels are now calculated on a circle as follows [12]:

- P represents the number of sampling points (neighbor pixels)

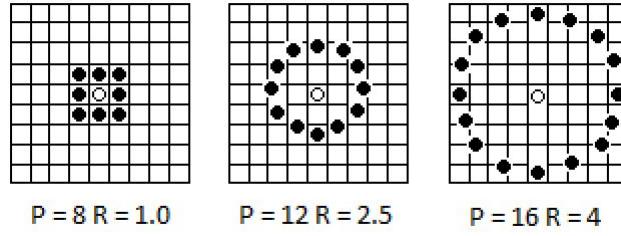
- C represents the central pixel with coordinates  $(x_c, y_c)$
- R represents the distance between each neighbor pixel and the central one

Coordinates of the  $p^{th}$  neighbor pixel are calculated with the following formulas [18]:

$$x = x_c + R \cos(2\pi p/P) \quad (6.1)$$

$$y = y_c + R \sin(2\pi p/P) \quad (6.2)$$

Figure 6.3 shows the circular LBP operator with different numbers of neighbor pixels  $P$  and with various radius sizes [18]. For the circular LBP operator, the following notation is used:  $(P, R)$  and for a pixel  $p$ , the following notation is used:  $LBP_{P,R}(p)$  [12].



**Figure 6.3:** Circular LBP operator with different radius sizes  $R$  and different number of neighbor pixels  $P$

Most of the time, when a circular LBP operator is used, coordinates of the  $p^{th}$  neighbor pixel (calculated with the formulas given above) may not land exactly on a pixel. In this case, using a bilinear interpolation of neighbor pixels intensity values is recommended. For example, the circular operator with  $P = 8$  and  $R = 1.0$  is similar to the basic LBP operator; the only difference being during the calculation, if neighbor pixels do not land exactly onto single pixels, these pixels have to be interpolated first [12].

The LBP operation outputs a texture, which formula is [12]:

$$T = t(I_C, I_0, I_1, \dots, I_{P-1}) \quad (6.3)$$

where  $t(I_P)$  represents the luminance for a specific pixel  $P$ .

With  $I_C$ , the intensity value of central pixel C,  $I_p$  for  $p = 0, 1, \dots, P - 1$ , the intensity values of the P neighbor pixels, and  $T$ , the texture in the local neighborhood of  $C$  [12].

This texture can also be calculated in another way. The central pixel intensity value can be subtracted from the computed neighbourhood values. Output texture  $T$  becomes the combination of differences between intensity values of the neighbor pixels and central pixel, and of the intensity value of the central pixel  $I_C$ , resulting in the following formula [12]:

$$T = t(I_C, I_0 - I_C, I_1 - I_C, \dots, I_{P-1} - I_C) \quad (6.4)$$

Based on the work of Ojala et al. [26],  $t(I_C)$  describes the overall luminance of the central pixel, which is not related to the local texture of the image. Consequently, relevant information for texture analysis are not provided by this value. It can then be discarded without affecting the texture description [12]:

$$T = t(I_0 - I_C, I_1 - I_C, \dots, I_{P-1} - I_C) \quad (6.5)$$

The above formula makes the texture description invariant against shifts in intensity values. It however does not make it invariant against scaling of intensity values. To obtain a texture description invariant to these scalings, only the signs of the differences are taken into account, and not the difference in itself. The new formula is as follows [12]:

$$T = t(s(I_0 - I_C), s(I_1 - I_C), \dots, s(I_{P-1} - I_C)) \quad (6.6)$$

where

$$s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases} \quad (6.7)$$

For pixels with an intensity value superior or equal to the one of the central pixel, the sign assigned would be 1. For pixels with an intensity value inferior to  $I_C$  it would be 0 [12].

The last step is to assign a binomial weight to each sign, for the LBP operation. These weights are summed, producing an imporved LBP code for central pixel  $C$

with coordinates  $(x_C, y_C)$  [12]:

$$LBP_{P,R}(x_C, y_C) = \sum_{p=0}^{P-1} s(I_p - I_C)2^p \quad (6.8)$$

The use of the formula above to calculate LBP still outputs a large number of patterns, which does not help reducing the data size. Another modification of the basic LBP operator, called uniform LBP, can then be used to efficiently reduce the data size. This kind of LBP will be explained in the subsection below.

### 6.2.2 Uniform LBP

A Local Binary Pattern can be called uniform only if it contains 2 or less bitwise transitions. A bitwise transition is a transition from 0 to 1 or from 1 to 0 [12].

In fact, there can be only 2 or 0 bitwise transitions for a uniform LBP. Indeed, the neighbourhood is circular so there cannot be 1-bitwise transition in the pattern. If there is a bitwise transition in the neighbourhood, i.e from 1 to 0, it means that there will be another bitwise transition from 0 to 1 further in the circular pattern [12].

If a pattern contains 0 bitwise transitions, it means that it is composed only of 0s or of 1s [12].

For a uniform LBP with a pattern-length of 8 bits (i.e 8 neighbours), containing 2 bitwise transitions, with  $P$  neighbor pixels, there are  $P(P-1)$  possible combinations. So there are  $8(8-1) + 2 = 58$  possible patterns ("+2" is for the 2 patterns with 0 bitwise transitions. The uniform LBP has the following notation:

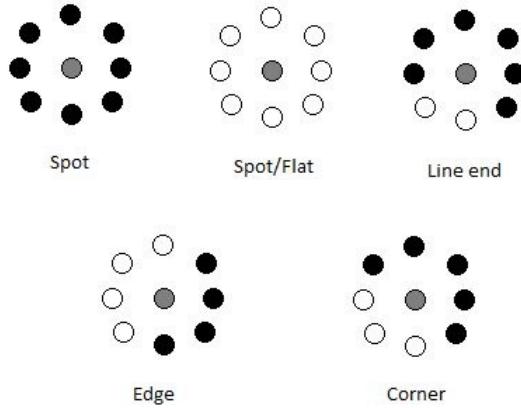
$$LBP_{P,R}^{u^2}$$

with  $P$  numbers of neighbouring pixels and radius size  $R$  [12].

There are 2 main advantages coming along with the uniform LBP. The first one is the reduction in range of possible patterns. With the uniform LBP operator, there are 58 possible combinations as seen above, whereas with the basic LBP operator, there are  $2^8 = 256$  possible combinations for the same bit-length. The use of uniform LBP then leads to less computation [12].

The second advantage is that, even though there is a reduction of dimensionality, the patterns remain discriminative between various structural features. These structural features detected by the uniform LBP are the spot, the spot/flat, the line end, the

edge and the corner, as shown in Figure 6.4 [12].



**Figure 6.4:** Patterns that can be detected with uniform LBP

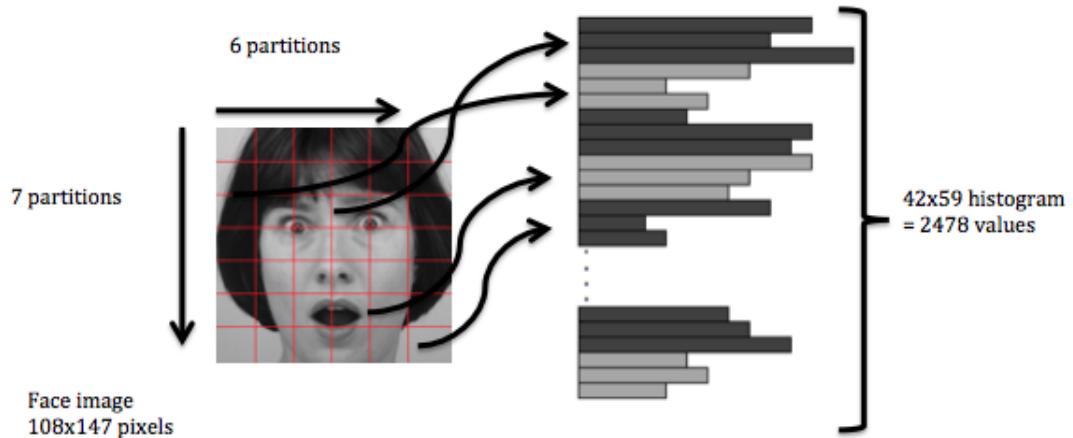
### 6.3 Facial Expression Recognition based on LBP

The LBP operator can be used to extract features for a facial expression recognition system. The main idea is to obtain a set of features vectors from the train data, using LBP. After training the classifiers with these feature vectors, test data will then undergo the same LBP feature extraction before classification.

### 6.4 Histogram computing

To obtain these features vectors mentioned above, the number of different pixel values obtained with the LBP operator are partitioned into an histogram, which will give feature vectors.

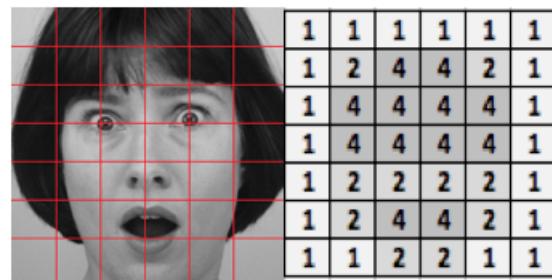
To compute this histogram, the face has to be partitioned into regions first. Most of the time, it is partitioned as following: 42 regions, 7 rows and 6 columns. It matches parts of the face yielding important features for emotion recognition. Face dimensions for all tested images should be kept constant. Then for all pixels of the face, their LBP value is computed using  $LBP_{8,1}^u$ . Then, resulting values are gathered region-wise into different bins. For example, the LBP operator with  $P = 8$  and



**Figure 6.5:** Example of a vector based on histogram computing

$R = 1.0$  outputs a histogram with 59 bins: 56 bins for uniform LBP with 2 bitwise transitions, 2 bins for uniform LBP with 0 bitwise transitions and the remaining bin for non-uniform patterns). All these bins are concatenated column-wise; as it can be seen in Figure 6.5. The final vector obtained with this histogram is a vector of  $42 \times 59 = 2478$  rows, the histogram being computed for each region partitioned on the sample image [12].

The regions obtained when the sample image is partitioned do not have the same contribution to the expressed emotion. For example, eyes bring more information about an emotion than cheeks. For each region, a weight is then assigned as in Figure 6.6 [12]. Thus, the histogram of each region is multiplied with a corresponding weight.



**Figure 6.6:** Weight assignment for each region

## Part IV

# Classification

# Contents

*This part will focus on the step following feature extraction, which is classification. Classification will be introduced in Chapter 7, with a general overview of the classification problem, along with a presentation of some classification algorithms. The next chapter will describe Support Vector Machines classification.*

<b>7 Classification</b>	<b>52</b>
7.1 Supervised learning . . . . .	53
7.2 Unsupervised learning . . . . .	54
<b>8 Support Vector Machines</b>	<b>58</b>
8.1 Overview . . . . .	58
8.2 Margin maximization . . . . .	59
8.3 Kernel function . . . . .	60
8.4 Combining LBP and SVM . . . . .	61

# Chapter 7

## Classification

Classification is done through machine learning algorithms. Machine learning, being a branch of Artificial Intelligence, aims to help AI systems improve their performances by learning from their environment. Indeed, the knowledge necessary to build a robust and intelligent system can not always be built-in or explained to it. The solution to this problem is to make the system learn this knowledge through examples and apply it to similar situations. Thus it will be able to perform relevant actions without the need of human intervention.

More specifically, machine learning can be described as a way to develop and implement algorithms taking empirical data as input, and processing these values in order to find links between them. The output will then be used by the system to compute the appropriate action or behaviour. In order to achieve that, the system needs to learn key characteristics from a training dataset given as example or obtained through past experience. It will then study this observable data, and build a model based on it. The system will then use this model to infer actions depending on new data it will get as input.

However, machine learning is not only about computing a database and relying on it for every possible situation. In a changing environment, the system needs to know how to learn from these changes and adapt itself.

There are many kinds of machine learning algorithms, with their own specificities and level of abstraction. In this chapter we will focus on algorithms behaving like functions and performing pattern recognition. Indeed, a facial image is a set of patterns of different sizes and shapes. Pattern recognition through machine learning can be divided into two main categories of algorithms: supervised learning and unsupervised learning. Those two categories will be described further in this chapter, along with examples.

This chapter serves as an introduction for Chapter 8, which is about Support Vector Machines, a specific kind of supervised learning algorithm which will be used in our system.

## 7.1 Supervised learning

Supervised learning is the task of providing labelled input data to the algorithm, also called *train data*. The main point is that the model is only defined by the observable data it gets for training. It does not make any assumption about underlying, latent variables that could interfere with this observable data. It can then search for patterns and relations between the data points, and build a model fitting these relations before classifying test data.

Classification can be divided in two steps, first step being training, and second step being prediction. There are many kinds of supervised learning algorithms, the basic one being a naive Bayes classifier, which will be detailed afterwards. An other example of algorithm is Linear Discriminant Analysis (LDA). Furthermore, the next chapter will focus on an other supervised learning algorithm: Support Vector Machines.

### 7.1.1 Naive Bayes classifier

The Bayesian classifier is based on Bayes theorem:

$$\begin{aligned} \text{sum rule: } \quad p(X) &= \sum_Y p(X, Y) \\ \text{product rule: } \quad p(X, Y) &= p(Y|X)p(X) \end{aligned} \tag{7.1}$$

With these rules, it is possible to compute the *posterior probability* of an event  $C_i$  given observable data  $x$ , using formula 7.2.

$$p(C_i|x) = \frac{p(x|C_i) \times p(C_i)}{p(x)} \tag{7.2}$$

With:

- $p(x|C_i)$  the probability of observed data  $x$  given  $C_i$ , which can also be called the *likelihood function of  $C_i$* ;
- $p(C_i)$  the prior probability of  $C_i$ ;
- $p(x)$  the probability of observable data  $x$ .

When combined with Bayes theorem, equation 7.2 becomes:

$$p(C_i|x) = \frac{p(x|C_i) \times p(C_i)}{\sum_{k=1}^K p(C_k) \times p(x|C_k)} \tag{7.3}$$

The classifier output will then be class  $C_i$  which meets condition  $p(C_i|x) = \max_k p(C_k|x)$  (the likelihood function has to be the highest among all classes  $C_k$ ).

### 7.1.2 Linear Discrimination

Since classification is the process of assigning input vector  $x$  to one of the classes  $C_i$ ,  $i = 1, \dots, I$ , its key point is about learning the decision boundaries that separate the different classes in the input space [5]. If the input dataset is *linearly separable*, decision boundaries can then be described by linear functions of the input vector  $x$ .

For example, the linear discriminant function for a 2-class problem will look like

$$y(x) = w^T x + \omega_0 \quad (7.4)$$

With *weight vector*  $w^T$  and *bias*  $\omega_0$  [5]. The input vector  $x$  is then assigned to class  $C_1$  if  $y(x) = 0$ , otherwise it will be classified as belonging to class  $C_2$ .

For problems with a number of classes  $K > 2$ , a *one-versus-the-rest* approach can be used, where  $K-1$  two-way discriminant functions are used: the  $k^{\text{th}}$  discriminant function will determine if a point belongs to class  $C_k$  or not. However, this combination of discriminant functions leaves an ambiguous region [5].

An other way to solve a multi-class problem could be to use  $\frac{K(K-1)}{2}$  discriminant functions, which is the *one-versus-all* approach. However, the resulting  $K$ -class discriminant also has the same ambiguous region problem as the one-versus-the-rest approach [5].

A  $K$ -class discriminant which does not lead to an ambiguous region problem is to use  $K$  linear discriminant functions  $g(x)_k$ ,  $k = 1..K$ , and to assign an input vector  $x$  to a class  $C_i$  if  $g(x)_i = \max_k g_k(x)$  [5].

## 7.2 Unsupervised learning

Unlike supervised learning, the classification algorithm is not fed with train data in the case of unsupervised learning. It will be given a set of observable data, and its goal is to group data the smartest way possible, and by itself. Furthermore, in unsupervised learning, the concept of *class* is not applicable any more; the term *cluster* being preferred.

The key point of unsupervised algorithms is data clustering. In most common unsupervised algorithms such as K-means, the algorithm can get a hint of the number

of clusters it has to find. It then proceeds, usually in an iterative way, to find the latent variables related to the data. Indeed, it is assumed that all observable data is governed by latent variables which can be organized in different levels.

Besides K-means, an other common unsupervised algorithm is the Mixture Models algorithm, which will also be described in the following subsections.

### 7.2.1 K-Means

K-means clustering relies on a set of  $k$  reference vectors  $\mu_k$ ,  $k = 1..K$ , also called *prototype vectors*. These vectors represent the centres of the  $K$  clusters. This clustering method has two goals: first, it has to find how the clusters are shaped and which data lies in which cluster; secondly, the set of vectors  $\{\mu_k\}$  has to be determined while verifying the following condition: for each data point  $x_n$ , the sum of squares of the distances between  $x_n$  and its closest vector  $\mu_k$  is a minimum. In other words, equation 7.5, also called *distortion function* has to be minimized [5].

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|x_n - \mu_k\|^2 \quad (7.5)$$

With

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

This can be achieved through the following iterative algorithm:

```

Initialization of reference vectors  $\mu_k$ ,  $k = 1..K$ 
repeat
  for all  $x_n \in \chi$  do
     $r_{nk} \leftarrow \begin{cases} 1 & \text{if } k = \arg \min_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$ 
  end for
  for all  $\mu_k$ ,  $k = 1..K$  do
     $\mu_k \leftarrow \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$ 
  end for
until  $\mu_k$  converge

```

In this algorithm, denominator  $\sum_n r_{nk}$  corresponds to the number of points assigned

to cluster  $k$ , which sets  $\mu_k$  as the mean of the cluster, hence the name *K-means algorithm* [5].

This algorithm can also be used in *lossy data compression*, where input data can be reconstructed with minor errors, in contrary to *lossless data compression*. If the K-means algorithm is applied, then for each data point  $x_n$  the only value stored is the cluster  $k$  it belongs to. Apart from the data points, values of cluster centres  $\mu_k$  are also stored. Hence, during data reconstruction, each data point is approximated by corresponding vector  $\mu_k$ . This approximation is called *vector quantization* [5].

### 7.2.2 Mixture of Gaussians

While the K-means algorithm will try to find reference vectors describing the different clusters, the mixture models algorithm will rather model the underlying distribution of the clusters, usually by a Gaussian probability density function. Each cluster is then represented by a Gaussian distribution, and the aim of the algorithm is to find the best parameters for the latent variables governing these distributions. It can be achieved by finding parameters maximizing likelihood  $p(x) = p(x|G_i)p(G_i)$ , with:

- $G_i$ : clusters
- $p(G_i)$ : prior probability (mixture proportion)
- $p(x|G_i)$ : component density

Since a Gaussian mixture is roughly similar to  $p(x|G_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$ , with mean vector  $\mu_i$  and covariance matrix  $\Sigma_i$ , the goal is now to maximize the log likelihood function described in Equation 7.6.

$$\begin{aligned} \mathcal{L}(\Phi|\chi) &= \ln \prod_n p(x^n|\Phi) \\ &= \sum_{n=1}^N \left\{ \sum_{k=1}^K \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\} \end{aligned} \quad (7.6)$$

with  $\Phi$  representing mixing coefficients, including prior probabilities and sufficient statistics of component densities.

The maximum likelihood estimation can then be obtained through the *Expectation-Maximization algorithm* for Gaussian Mixture Models (EM), which converges and fits a mixture model onto each cluster [5].

### Expectation-Maximization algorithm:

Initialize means  $\mu_n$ , covariances  $\Sigma_n$ , mixing coefficients  $\pi_n$  and compute initial value of the log likelihood

**repeat**

**Expectation step:** Evaluate the expected value of the latent variable using current parameter values:

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j \mathcal{N}(x_n | \mu_j, \Sigma_j)}$$

**Maximization step:** Re-estimate the parameters using  $\gamma(z_{nk})$ :

$$\pi_k^{new} = \frac{N_k}{N}$$

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N_k} N \gamma(z_{nk}) x_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N_k} N \gamma(z_{nk}) (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$\text{Where } N_k = \sum_{n=1}^{N_k} N \gamma(z_{nk})$$

    Evaluate log likelihood  $\mathcal{L}$

**until**  $\mathcal{L}$  converges

# Chapter 8

## Support Vector Machines

Support Vector Machines (SVM) is a binary linear classifier belonging to the supervised learning algorithms category. It has been proven to be very efficient for classification, regression and novelty detection problems [5], which makes it suitable for facial expression recognition.

This chapter will first provide an overview of how classification is performed using SVM. It will then focus and describe the key points behind this classifier, namely *margin maximization* and *kernel function*. The chapter will conclude with the review of a research paper detailing facial expression recognition using LBP for feature extraction and SVM for classification.

### 8.1 Overview

SVM is originally a binary classifier, which means it has a *one-vs-all* approach. It is a decision machine, so its output is not a posterior probability, but rather a class label [5]. It can be adapted into a *relevance vector machine* to output posterior probabilities though [5], but this alternative will not be detailed in our report.

SVM is also a linear classifier, such as LDA. For a two-class problem, it will linearly separate the train data by finding the optimal hyperplane between these 2 classes. This hyperplane is defined as being as far as possible from both classes. *Margin maximization* is then applied to optimize the distance between the two classes and the hyperplane, as it will be explained in Section 8.2.

The name "Support Vector" originates from this margin maximization feature. Indeed, the distance between the classes and the separating hyperplane is computed regarding to the closest data points from both classes. These particular data points, lying on the margins edges, are the support vectors. Some properties are associated to these vectors, such as non-zero Lagrange multipliers (see Section 8.2), which builds the classification algorithm.

For a multi-class problem, however, this linear separation is not possible anymore.

The dataset has to be mapped into an other space, where it can be linearly separated. This is what *kernel functions* are used for, as described in Section 8.3.

## 8.2 Margin maximization

As introduced in Section 8.1, margin maximization for a two-class linear problem starts by finding the separating hyperplane between these two classes. A linear classification model of the form  $f(x) = w(x) + b$  can be inferred, with  $w$  being the normal to the hyperplane and  $b$  the bias. The hyperplane can then be characterized by  $w(x) + b = 0$ .

Margin is defined as the distance between the closest point of the class to the hyperplane, and that hyperplane, which can also be written as  $d(x) = \frac{|w(x)+b|}{\|w\|}$ . Since a data point  $(x_i, y_i)$  is correctly classified if  $y_i f(x_i) \geq 1(1)$ , maximizing the margin is the action of maximizing  $\|w\|^{-1}$ , which is consequently equivalent to minimizing  $\|w\|^2$  depending on constraint (1). Margin maximization then requires to solve a *quadratic programming* problem under constraints, as seen in Equation 8.1.

$$\begin{cases} \min \frac{1}{2} \|w\|^2 \\ \forall i, y_i \cdot f(x) \geq 1 \end{cases} \quad (8.1)$$

In order to solve this problem, positive *Lagrangian coefficients*  $\alpha_i$ ,  $i = 1..N$  are introduced for each constraint present in Equation 8.1. It outputs a Lagrangian function of  $w$  and  $b$ :

$$L_P(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^N \alpha_i (y_i (w \cdot x_i + b) - 1) \quad (8.2)$$

With  $\alpha = (\alpha_1..\alpha_N)^T$ . When setting  $\frac{\delta L}{\delta w} = 0$  and  $\frac{\delta L}{\delta b} = 0$  in order to minimize  $L_p$  with respect to  $w$  and  $b$ , two conditions can be obtained

$$w = \sum_{i=1}^N N \alpha_i y_i x_i \quad (8.3)$$

and

$$\sum_{i=1}^N N \alpha_i y_i = 0. \quad (8.4)$$

When substituting these conditions into Equation 8.2, parameters  $w$  and  $b$  disappear. Thus the resulting function is Equation 8.5

$$L_D = \sum_{i=1}^N N\alpha_i - \frac{1}{2} \sum_{i=1}^N N \sum_{j=1}^N N\alpha_i\alpha_j y_i y_j (x_i \cdot x_j) \quad (8.5)$$

This function is called *dual representation* of the margin maximization problem. In order to solve this problem, one can either minimize  $L_P$  in Equation 8.2 or maximize  $L_D$  in Equation 8.5.

### 8.3 Kernel function

It might however not be possible to perform linear separation with more classes. Indeed, data might be overlapping, and thus it will not be a linear problem anymore. The solution to overcome this problem is to map the non-linear dataset from its input space into a higher feature space using a function  $\Phi(x)$ , and perform margin maximization and classification in this higher space.

In order to achieve this mapping, a *kernel function* of the form  $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$  is applied to the dataset. This kernel represents an inner product in the feature space. There are four kernels available, which are described in Equation 8.6.

Linear kernel:	$K(x_i, x_j) = x_i^T x_j$
Polynomial kernel:	$K(x_i, x_j) = (\gamma x_i^T x_j + r)^T, \gamma > 0$
Radial Basis Function (Gaussian) kernel:	$K(x_i, x_j) = \exp(-\gamma \ x_i - x_j\ ^2), \gamma > 0$
Sigmoid kernel:	$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

(8.6)

The main advantage of using a kernel function is that there is no need to define or calculate  $\Phi(x_i)$ , only  $\Phi(x_i)^T \Phi(x_j)$ . We hence do not know the true form of  $\Phi(x_i)$ . Despite having an incomplete knowledge of  $K(x_i, x_j)$ , the four kernels described above can be used as they are, or be combined in order to build more complex kernels.

## 8.4 Combining LBP and SVM

In a 2009 article, Shang and al [30] have performed facial expression recognition while using Local Binary Patterns for feature extraction, and compared the accuracy of different kinds of classifiers: template matching, LDA and SVM. The template-matching classifier was a nearest-neighbour classifier based on *Chi square statistic* ( $\chi^2$ ) dissimilarity measure. Secondly, train data for LDA was first mapped into a PCA subspace. Even though the dimensionality of the data was reduced, 98% of the information was kept [30]. It was then combined with a Neural Network (NN) classifier. About SVM, since it is a binary classifier, multi-class classification was achieved through a *one-versus-the-rest* approach. It is the same as for K-means, as presented in Chapter 7, Section 7.2. They also used 10-fold cross-validation, and performed grid search to find optimized parameters. Images from the Cohn-Kanade database were used as train and test data.

The output of their study is that classification using SVM has a higher accuracy rate than the two other methods. Template matching yielded 84.5% of accuracy when performing 6-class recognition, and 79.1% for 7-class recognition. SVM outperformed largely this method by having an accuracy rate between 91.6% and 92.5% for 6-class recognition depending on the kernel used, and between 88.1% and 88.9% for 7-class recognition [30]. When comparing LDA and SVM, since LDA was trained on data mapped into another subspace, SVM classification was also performed on the same subspace. The combination of LDA and NN had accuracy rates of 79.2% and 73.4%, respectively for 6-class and 7-class recognition. It can already be inferred that LDA+NN performed worse than template matching when provided with appropriate parameters and data. Then, with SVM trained on the same subspace and with the linear kernel, the accuracy rate was 87.7% (6-class) and 80.2% (7-class) [30].

They also used a linear programming classification method which was presented in 2005 [11], but it will not be detailed in this report. The general outcome is that SVM outperforms other classification methods, which makes it very efficient for facial expression recognition.

This study also compared the efficiency of Local Binary Patterns and Gabor wavelets, when combined with SVM. Gabor wavelets yielded accuracy rates generally 2% lower than LBP-based feature extraction, no matter which SVM kernel is used. Indeed, LBP feature extraction had an accuracy rate between 91.5% and 92.6% for 6-class recognition, while Gabor wavelets feature extraction resulted in accuracy rates ranging from 89.4% to 89.8% [30]. For 7-class recognition, Gabor wavelets accuracy ranged between 86.6% and 86.8%, while LBP-based feature extraction had an accuracy rate between 88.1% and 88.9% [30]. The conclusion is that the combination of LBP and SVM, when provided with suitable parameters, has a very high accuracy

rate for facial expression recognition.

However, the accuracy for each facial expression varied a lot. When used with RBF kernel, SVM had some difficulties especially when it had to distinguish fear and sadness, the two facial expressions which had the lowest accuracy rates (6-class recognition: 73.0% and 83.5% respectively; 7-class recognition: 68.0% and 69.5% respectively). Indeed, expression recognition accuracy ranged from 88.9% (anger) to 98.7% (surprise) in 6-class recognition, and from 85.0% (anger) to 98.2% (surprise) for 7-class recognition.

Since the accuracy of the system presented in this article was very high, we chose to implement a similar system. Indeed, we are performing facial expression recognition using LBP for feature extraction, and SVM classification. We however did not use the Cohn-Kanade database as train data because we did not get it on time. We will describe our implementation and results further in the report.

**Part V**

**Implementation**

# Contents

*In the previous parts, algorithms for face detection and feature extraction were studied. A method for classification was also examined. This part will describe and explain how these algorithms and methods are implemented.*

<b>9 Face detection with Viola-Jones</b>	<b>65</b>
9.1 Viola-Jones . . . . .	65
<b>10 Feature extraction with Local Binary Patterns</b>	<b>66</b>
10.1 Uniform Local Binary Patterns . . . . .	66
10.2 Histogram computing . . . . .	66
<b>11 Classification with Support Vector Machines</b>	<b>68</b>
11.1 Histogram concatenation . . . . .	68
11.2 Training and Model . . . . .	69
<b>12 Implementation on the Kinect</b>	<b>71</b>
12.1 Generalities . . . . .	71
12.2 Librairies . . . . .	71
12.3 Architecture . . . . .	72
12.4 Interactions . . . . .	72
12.5 Algorithm . . . . .	73

# Chapter 9

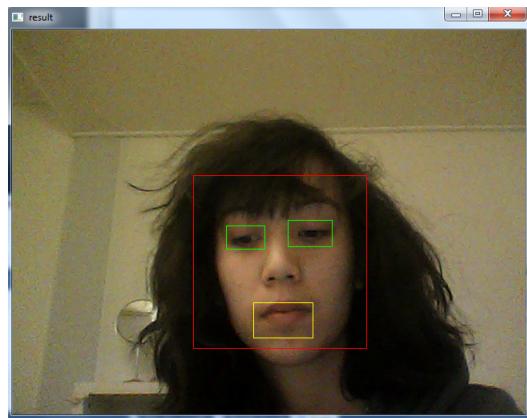
## Face detection with Viola-Jones

The face detection part for this facial expression recognition system is based on Viola-Jones face detection algorithm. This chapter describes how this Viola-Jones algorithm is used and implemented.

### 9.1 Viola-Jones

To perform Viola-Jones face detection, video sequences of face are obtained with a webcam or the Kinect. The algorithm is then able to detect face and regions of interest in the frames, which are nose, left eye, right eye and mouth.

Classifiers are trained prior to be used. They are then loaded with a model; one for the face and one for each ROI. Then, based on these classifiers, face and ROI are detected in the frames. Figure 9.1 shows an example of face and ROI detection. In this case of ROI detection, separate classifiers have been used for each eye.



**Figure 9.1:** Example of face and ROI detection with Viola-Jones

Detected faces are extracted with a size of  $200 \times 200$  pixels to discard as much background as possible, so they only contain the face and part of surrounding hair.

# Chapter 10

## Feature extraction with Local Binary Patterns

As said previously, the Local Binary Patterns (LBP) method is used as feature extraction method for this facial expression recognition system. This chapter describes how this method is used and implemented in our system.

### 10.1 Uniform Local Binary Patterns

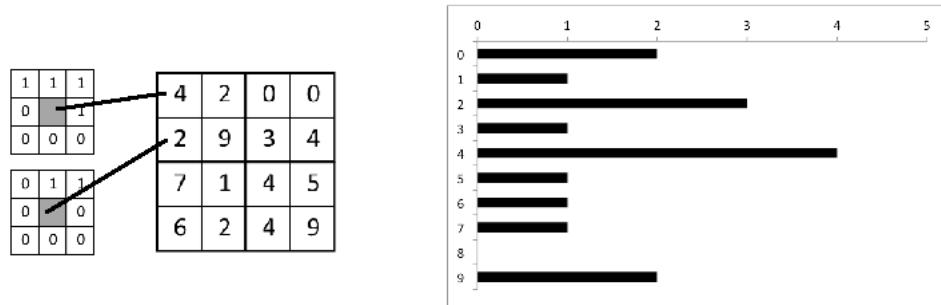
A circular invariant uniform LBP operator,  $LBP_{8,1}(p)$ , is used with a number of neighbours  $P = 8$  and radius  $R = 1.0$  to compute pixel values of the image. The face image is divided into regions. As it is commonly done, the face is a grid pattern of  $7 \times 6$  (7 partitions for the rows and 6 partitions for the columns). In total the face image is divided into 42 regions. For each of these regions, the LBP operator,  $LBP_{8,1}^{u^2}$ , computes every pixel.

The output of this computation is the number of labelled 1s in the 8-pixels neighbourhood because circular invariant uniform LBP operator is used. Because a uniform LBP operator is used, values range from 0 to 8. There can also be an output of 9 for non-uniform LBP patterns. To sum up, the output number can have 10 different values (from 0 to 9).

### 10.2 Histogram computing

After obtaining all pixel values from a region, they are concatenated into a 10-bin histogram, one for each value from 0 to 9. Each bin contains the number of pixels having the value corresponding to that bin. For example, if in a specific region, 20 pixels are labelled with value 7 after being computed by the LBP operator, then the bin attributed to value 7 will have a value of 20 for this specific region. Figure 10.1 shows two pixels with their eight neighbor pixels, a region containing these pixels with the values obtained with the LBP operator and an histogram computed for this

region.



**Figure 10.1:** Example of Histogram computation

To obtain the feature vector for the whole image, an histogram is computed for each region. Thus, 42 histograms are computed for the 42 regions. Since each histogram has 10 bins, the concatenation of all histograms outputs a feature vector of 420 points. Figure 10.2 shows how the feature vector is obtained with all the histograms.



**Figure 10.2:** Example of a feature vector composed by the 42 histograms

# Chapter 11

## Classification with Support Vector Machines

The classification part for this facial expression recognition system uses a Support Vector Machines (SVM) classifier. This chapter describes how SVM is used and implemented

### 11.1 Histogram concatenation

The output of the feature extraction part using Local Binary Patterns (LBP) is a feature vector characterizing the whole image, made of histograms characterizing each region of the face image. In order to proceed to the classification part, the feature vector has to be concatenated to be readable by SVM functions.

For the training part, the feature vector has to be in the following form:

$$l \quad i : v \quad i : v \quad \dots \quad -1 : -1$$

$l$  stands for the label of the class the image belongs to. A label is a number associated to the 6 emotions, plus neutral state. These emotions have been ordered alphabetically and bear a number between 1 and 6, the neutral state having value 0. Values corresponding to the emotions are then assigned as following:

- *Neutral* state: 0
- *Afraid* state: 1
- *Angry* state: 2
- *Disgusted* state: 3
- *Happy* state: 4
- *Sad* state: 5
- *Surprised* state: 6

$i$  stands for the index of a bin in the feature vector, hence a number from 1 to 420 (420 bins for the feature vector).

$v$  stands for the value that the bin has at a given index  $i$ . These values are normalized to fit in interval  $[-1; 1]$  so small values will not be overwhelmed by higher values. To normalize the values, the maximum value  $max$  of the whole feature vector has to be found as well as the minimum value  $min$ . To normalize the feature vector, each value has to be processed with the following formula:

$$new\_v = -1 + (1 - (-1)) \times \frac{v - min}{max - min} \quad (11.1)$$

where,

- $v$ , the value that the bin has at a given index
- $new\_v$ , the new normalized value
- $-1$ , the minimum of the normalized interval
- $1$ , the maximum of the normalized interval
- $min$ , the minimum  $v$  value of the whole feature vector
- $max$ , the maximum  $v$  value of the whole feature vector

The feature vector has to be ended by the pair  $-1 : -1$ .

## 11.2 Training and Model

The training part uses images from databases. The database used is the Karolinska Directed Emotional Faces Database (KDEF) database, as described in Chapter 1. All face images are sorted by emotions. To compute feature vectors for train data, 7 folders have to be browsed, one for each emotion.

While browsing all folders, a LBP feature vector is computed for each image, and added to a file, which will eventually contain all feature vectors of the training set. The SVM model will then be trained using these feature vectors.

All four kernels available were tested to find the best one. As presented in Equation 8.6, the four kernels are the following:

- Linear:

$$K(x_i, x_j) = x_i^T x_j \quad (11.2)$$

- Polynomial:

$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^T, \gamma > 0 \quad (11.3)$$

- Radial basis function (RBF)

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \quad (11.4)$$

- Sigmoid:

$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \quad (11.5)$$

For the three last kernels, the  $\gamma$  parameter can be changed and optimized. Each of these kernels was tested with the default  $\gamma$  parameter, and then with an optimized one. The optimized  $\gamma$  parameter has been computed using an optimization tool.

For the polynomial kernel, the  $D$  parameter represents the dimension of the polynomial. It has to be chosen. Once it is chosen the  $\gamma$  is found automatically.

For the RBF and the sigmoid kernel, two parameters has to be chosen: the cache parameter and the  $\gamma$  parameter. These parameters can be found by using *gridsearch*, a script of the SVM library. They are the most optimized for the RBF kernel and for the sigmoid kernel.

The cross validation method was also used. It is a method that trains and tests the data at the same time by itself in order to find the best parameters possible. It starts by dividing the dataset into  $n$  parts, and one part is used for testing while the other are used for training. The cross validation method has  $n$  training/testing cycles, each cycle using a different testing set.

# Chapter 12

## Implementation on the Kinect

### 12.1 Generalities

The purpose of the project is to implement a system able to recognize emotions through images coming from a Kinect camera sensor. The system is designed to run with Microsoft Windows 7 and need an available USB 2.0 port to plug the Kinect device. The project has been coded in C++, which allows us to include useful third-party libraries in our project.

We used Microsoft Visual Studio 2010 to code and GitHub as a source control management system to facilitate parallel development of features.

### 12.2 Librairies

Several different libraries have been used to perform facial expression recognition in our system. One for the communication between the computer and Kinect sensors, one for image processing and another one for classification.

- The library used for intercommunication with sensors is the Software Development Kit released by Microsoft for their Kinect for XBox in its version 1.0 Beta 2.
- OpenCV (Open Source Computer Vision Library) is a graphic library under BSD licence, optimized for real-time image processing. It have been released by Intel and is actually maintained by Willow Garage, a robotic company.
- We are using LibSVM to perform classification, which is an OpenSource library for Support Vector Machines [6]. The version used is the 3.14 released on November 16, 2012.

## 12.3 Architecture

The program follows a Model-View-Controller architecture. This MVC pattern consists in 3 modules:

- The model part contains all used algorithms;
- The view enables user interaction with the system by displaying human-readable information;
- The controller sends commands in order to manage the other modules.

Since we use an object-oriented language, the use of the MVC pattern is easier. Five classes have been created: 3 for the architecture, one for LBP processing, and the remaining one for classification, as seen in Figure 12.1.

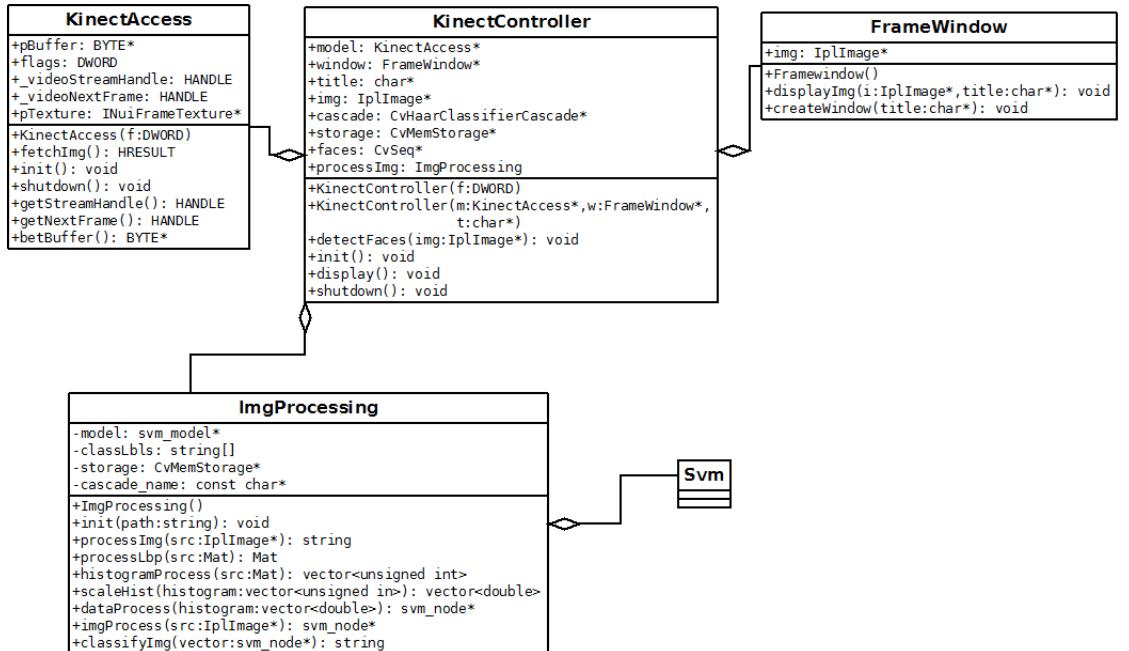


Figure 12.1: UML diagram describing our implementation of a facial expression recognition system

## 12.4 Interactions

There are two possible ways for the user to interact with the system:

- The user displays a facial expression in front of the camera;
- The user decides when he or she wants the program to perform facial expression recognition.

The first possibility does not need the user to be active. Indeed, the Kinect camera sensor can record 30 images per second, so each expression can be caught in the image stream. That is why the second option needs to be triggered by the user (in our case, press a button) to "capture" the facial expression and start the analysis process.

The Figure 12.2 shows what the subject sees on the computer. The subject expressed the *angry* emotion and it was well recognized.

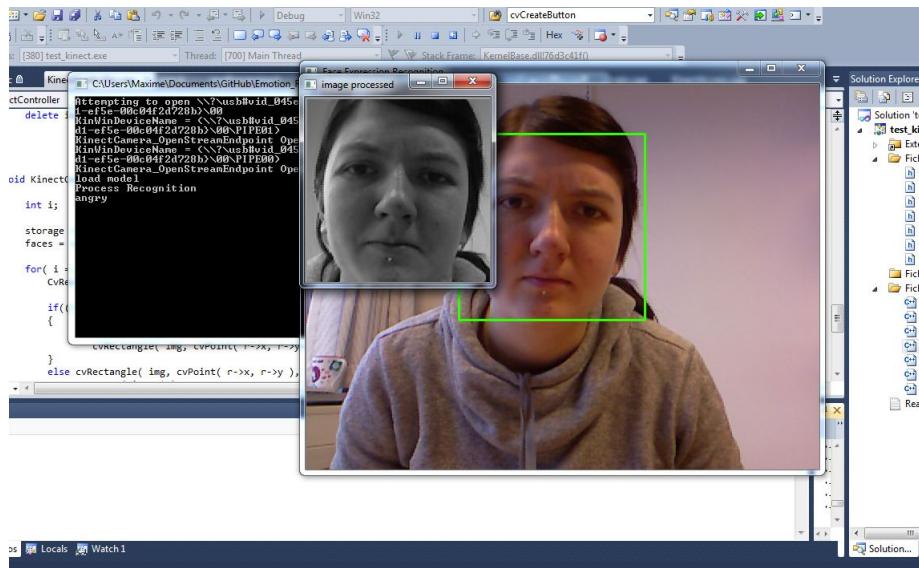


Figure 12.2: Interface on the computer

## 12.5 Algorithm

The entry point of the software first initializes the 3 MVC modules. It then runs 3 functions through the controller:

- Initialization which loads models (for face detection and classification through SVM) and begins images capture;

- Main loop of the program which displays images and waits for actions from the user;
- Shutdown process, which deletes models, releases memory and closes communication with sensors.

### Main loop algorithm

```

while button 'q' is not pressed do
    wait for single object from videoNextFrame handle
    result ← fetchImage(videoStream handle)
    if result == success then
        Texture ← Image.texture
        LockedRectangle ← Texture.lockedRectangle
        if LockedRectangle.bits ≠ 0 then
            buffer ← LockedRectangle.bits
            unlockRectangle()
            releaseImageFrame(videoStream handle)
            openCvImage ← buffer
            face ← detectFace(openCvImage)
            roi ← setRegionOfInterest(face)
            if button 'r' is pressed then
                openCvImage2 ← openCvImage
                greyscImage ← convertToGrayscale(openCvImage2)
                featVector ← lbpProcess(greyscImage)
                label ← classify(featVector)
                display(label)
            end if
            delete(roi)
        end if
    end if
end while

```

**Part VI**

**Evaluation**

# Contents

*This part is two-folded: its first chapter describes the results obtained through this facial expression recognition system, while Chapter 14 focuses on the main issues faced when implementing this system. A concluding chapter sums up the framework and methods used in this facial expression recognition system, the results, and explore some possible improvements.*

<b>13 Results</b>	<b>77</b>
13.1 First result set . . . . .	77
13.2 Second result set . . . . .	81
13.3 Additional results . . . . .	82
<b>14 Issues</b>	<b>85</b>
14.1 Feature extraction . . . . .	85
14.2 Real-time . . . . .	85
14.3 Issues with the implementation on the Kinect . . . . .	86
14.4 Training dataset . . . . .	87
<b>15 Conclusion</b>	<b>88</b>
15.1 Theoretical framework . . . . .	88
15.2 Results . . . . .	88
15.3 Future work . . . . .	89

# Chapter 13

## Results

Two kinds of results have been obtained. First results have been obtained by extracting a test set from the KDEF database, and test it against other subjects from the database. Feature extraction is done by LBP, followed by classification using SVM. The second set of results is obtained with the Kinect in real-time conditions, while the entire KDEF database is used for training. The Kinect gets video sequences of a subject in front of it, his or her face being extracted from these sequences using Viola-Jones algorithm. Then the same feature extraction and classification process is applied to these images.

### 13.1 First result set

To train the model, 128 face images from the KDEF database have been used for each emotion, plus neutral state. In total,  $128 \times 7 = 896$  face images have been used as train data. To test the system, 12 face images from the KDEF database have been used for each emotion, plus neutral state. In total,  $12 \times 7 = 84$  faces images have been used as test data. For each of these 84 images, face detection was performed first, then the uniform LBP operator extract its features, and then classification is performed using SVM.

The model has been trained with different kernels and different parameters. The outcome of these different processes are summed up in Table 13.1.

*Poly1* stands for Polynomial and has degree parameter:  $D = 2$  and  $\gamma = 0.001953125$

*Poly2* stands for Polynomial and has degree parameter:  $D = 3$  and  $\gamma = 0.001953125$

*RBF1* has cache and  $\gamma$  parameters:  $C = 128$  and  $\gamma = 0.0078125$

*RBF2* has cache and  $\gamma$  parameters:  $C = 8192$  and  $\gamma = 0.00048828125$

*Sigmoid1* has cache and  $\gamma$  parameters:  $C = 128$  and  $\gamma = 0.0078125$

*Sigmoid2* has cache and  $\gamma$  parameters:  $C = 8192$  and  $\gamma = 0.00048828125$

As said in Chapter 11, parameters for RBF and sigmoid kernels are found by the *gridsearch* script of LIBSVM. For the polynomial kernel, as said in Chapter 11, degree parameter  $D$  has to be chosen, otherwise the default value is  $D = 3$ . *Poly2* has default  $D$  value, while *Poly1* has  $D = 2$ . It has been chosen because if dimension is greater than 3 then there an *overfitting* issue. It means that the model yields good

**Table 13.1:** Results with different kernels and different parameters

	Linear	Poly1	Poly2	RBF1	RBF2	Sigmoid1	Sigmoid2
neutral	66.00%	<b>83.33%</b>	83.33%	83.33%	66.67%	8.33%	66.67%
afraid	58.33%	<b>83.33%</b>	75.00%	66.67%	66.67%	66.67%	58.33%
angry	41.67%	<b>41.67%</b>	33.33%	50.00%	41.67%	16.67%	41.67%
disgusted	58.33%	<b>75.00%</b>	75.00%	50.00%	50.00%	16.67%	58.33%
happy	100.00%	<b>100.00%</b>	91.67%	91.67%	100.00%	66.67%	100.00%
sad	8.33%	<b>8.33%</b>	0.00%	8.33%	8.33%	8.33%	8.33%
surprised	66.67%	<b>75.00%</b>	75.00%	66.67%	66.67%	25.00%	66.67%
<b>overall</b>	<b>57.14%</b>	<b>66.67%</b>	<b>61.90%</b>	<b>59.52%</b>	<b>57.14%</b>	<b>29.76%</b>	<b>57.14%</b>

results, not because it has been trained with suitable parameters, but because it fits the test data too much, and will have erroneous results if given other data.

A model has also been trained with same kernels and parameters but using cross-validation (as explained in Chapter 11). Results obtained with a model using cross-validation are compared to those with a model trained normally, this comparison being shown in Table 13.2. All results are inferior or equal to those without cross-validation, except for the sigmoid kernel tuned with parameters cache and  $\gamma$  equals to  $C = 8192$  and  $\gamma = 0.00048828125$ .

**Table 13.2:** Results with and without cross validation

	with cross validation	without cross validation
Linear	57.14%	57.14%
Poly1	55.95%	66.67%
Poly2	47.62%	61.90%
RBF1	52.52%	59.52%
RBF2	55.95%	57.14%
Sigmoid1	16.67%	29.76%
Sigmoid2	63.10%	57.14%

The best accuracy percentage is for a model trained with Poly1 kernel parameters, with an overall accuracy rate of 66.67%. It is obtained with a classification based on the polynomial kernel and with following parameters:  $D = 2$  and  $\gamma = 0.001953125$ . Table 13.3 represents the confusion matrix for this kernel.

By looking at the confusion matrix, it is easy to notice that 2 facial expressions are harder to recognize than others with this system: *angry* and *sad*. There is a great difference between these 2 emotions and the 5 other ones (afraid, disgusted, happy,

**Table 13.3:** Confusion matrix

	neutral	afraid	angry	disgusted	happy	sad	surprised	accuracy
neutral	<b>10</b>	0	0	0	0	2	0	83.33%
afraid	0	<b>10</b>	1	0	0	0	1	83.33%
angry	4	0	<b>5</b>	0	0	3	0	41.67%
disgusted	1	0	0	<b>9</b>	1	1	0	75.00%
happy	0	0	0	0	<b>12</b>	0	0	100.00%
sad	5	1	2	2	1	<b>1</b>	0	8.33%
surprised	0	3	0	0	0	0	<b>9</b>	75.00%

neutral and surprised). Indeed, these 2 emotions are recognized with an accuracy lower than 50%, while the 5 other ones are recognized with an accuracy equal or greater than 75%, as summed up in Table 13.4. Furthermore, the recognition accuracy for the *sad* expression is much lower than random guess, which really contrasts with the *happy* facial expression, the latter reaching a 100% accuracy.

**Table 13.4:** Recognition accuracy of the six basic emotions and of the neutral state

< 50%	> 75%
angry (5/12)	afraid (10/12)
sad (1/12)	disgusted (9/12) happy (12/12) surprised (9/12) neutral (10/12)

The numbers in parenthesis represent the number of faces correctly classified over the total number of face images tested.

These 6 emotions plus neutral state can be categorized into 2 groups. Indeed, one group containing the 2 emotions hard to recognize and another group containing the 5 remaining emotions.

The first group contains the 5 facial expressions *afraid*, *disgusted*, *happy*, *surprised* and the *neutral* state. These emotions distort significantly the face when they are expressed, except for neutral state, which is the basic face of the subject when he or she does not express an emotion. This is why it is easier to recognize them. Figure 13.1 shows face images from the KDEF database used in the test set expressing these 5 facial expressions. Important features carrying emotion as mouth or eyes are changing a lot while these 5 emotions are expressed.



**Figure 13.1:** Face images from the KDEF database used in the test set

As it can be seen in Figure 13.1, for each emotion, eyebrows are raised, eyes are widely opened, and the mouth has a distinct shape, whereas in Figure 13.2 there are no differences as visible as in Figure 13.1. These variations of intensity might explain why it is easier for the system to differentiate them.

The second group contains the 2 remaining facial expressions. The 2 facial expressions *angry* and *sad*, are the ones that distort the less the face. For a same subject expressing these 2 different emotions, as in Figure 13.2, the differences are not clearly noticeable. This might explain why the system struggles when trying to differentiate *angry* and *sad* emotional states and mistakes it with neutral state.



**Figure 13.2:** Face images from the KDEF database used in the test set

The *sad* emotion is the hardest to recognize. On 12 test faces, it has been well recognized only once. It is mostly mistaken for *neutral* state. As it can be seen in Figure 13.2, there are no clear variations between *neutral* state and *sad* emotion, only the eyebrows are lightly frowned.

The *angry* emotion is also hard to recognize. On 12 test faces, it has been well recognized only 5 times. It is mostly mistaken for *neutral* state or for *angry* emotion. As it can be seen in Figure 13.2, there are no bold variations between *neutral* state,

*sad* emotion and *angry* emotion. Only the eyebrows slightly change, as well as the eyes. The mouth is almost identical between the three face images in Figure 13.2

## 13.2 Second result set

The second result test is processed in the same way as the first data set, the only difference being the input. Indeed, it is not performed on static images extracted from the KDEF database anymore; images used for testing are extracted from the video stream coming from the Kinect. A subject stands in front of the Kinect, his or her face is detected and extracted, then features are computed, and finally classification is performed. It runs almost in real-time, and the process outputs the name of the emotion expressed.

### 13.2.1 With face images from the KDEF database

Because results do not have a good accuracy at first sight, another method was tested. Face images from the KDEF database are given as input directly to our program, bypassing face detection using the Kinect. Ten face images are randomly chosen among all face images present in the KDEF database to be tested for each emotion.

The model used is the one that gave the best results in the first test set: polynomial kernel with following parameters:  $D = 2$  and  $\gamma = 0.001953125$ . Table 13.5 represents the confusion matrix for this kernel with these parameters.

**Table 13.5:** Confusion matrix

	neutral	afraid	angry	disgusted	happy	sad	surprised	accuracy
neutral	<b>6</b>	1	2	0	0	1	0	60.00%
afraid	0	<b>4</b>	0	3	0	<b>3</b>	0	40.00%
angry	2	1	<b>6</b>	1	0	0	0	60.00%
disgusted	0	1	0	<b>7</b>	1	1	0	70.00%
happy	0	0	0	0	<b>10</b>	0	0	100.00%
sad	1	3	2	1	1	<b>2</b>	0	20.00%
surprised	0	0	0	0	0	0	<b>10</b>	100.00%

Overall accuracy rate is 64.28%, which is lower than the one obtained previously. Obtained results are however consistent with those found in the first result set.

### 13.2.2 With subjects

For this test set, we used face images from the JAFFE database and displayed them in front of the Kinect. Ten face images has been chosen randomly from this databases for each emotion.

As for the precedent part, the model used is the polynomial kernel with following parameters:  $D = 2$  and  $\gamma = 0.001953125$ . Table 13.6 represents the confusion matrix for this kernel and these test conditions.

**Table 13.6:** Confusion matrix

	neutral	afraid	angry	disgusted	happy	sad	surprised	accuracy
neutral	<b>6</b>	0	0	0	0	4	0	60.00%
afraid	2	<b>1</b>	1	0	0	6	0	10.00%
angry	3	0	<b>0</b>	0	0	7	0	0.00%
disgusted	1	0	1	<b>0</b>	0	8	0	0.00%
happy	4	<b>3</b>	0	0	<b>0</b>	3	0	0.00%
sad	4	0	0	0	0	<b>5</b>	1	50.00%
surprised	5	0	1	0	0	4	<b>0</b>	0.00%

The overall accuracy rate is 15,71% which is really low compared to the previous result, and even lower than the one obtained in the first result set of 66.67%.

## 13.3 Additional results

One way to obtain better results with the LBP operator for this system is to add weights for each of the 42 regions of the face, as seen in Chapter 6. The LBP operator used in this system is already far from the basic LBP operator; it is a uniform circular invariant LBP operator. Even though results obtained with this operator are quite good with an accuracy of 66.67%, there is still room for improvement of feature extraction. Weighting the regions of the face will have an impact on the computed histogram, hence on the resulting feature vector ready for classification.

The face image is divided in 42 regions (7 rows  $\times$  6 columns), as seen in Chapter 6, and mentioned in [12]. The weights are however not applied in the same way as in [12], but rather as in Figure 6.6. Figure 13.3 shows an example of the division into regions of face images from the KDEF database that this system uses. Thus, border regions are less important than those containing the ROI of the face (eyes, nose, mouth). Figure 13.3 shows an example of the division into regions of face images

from the KDEF database used by this system.



**Figure 13.3:** Example of division into regions of face images from the KDEF database

Results were obtained using the same process as for the first set result. The results with weights applied are summed up in Table 13.7. The model has also been trained with cross-validation.

**Table 13.7:** Results with different kernels and different parameters

	Linear	<b>Poly1</b>	Poly2	RBF1	RBF2	Sigmoid1	Sigmoid2
neutral	75.00%	75.00%	91,67%	<b>91,67%</b>	75.00%	100.00%	66.67%
afraid	91,67%	83.33%	100.00%	<b>100.00%</b>	83.33%	33.33%	91,67%
angry	58.33%	58.33%	41.67%	<b>66.67%</b>	58.33%	50.00%	50.00%
disgusted	41.67%	58.33%	50.00%	<b>50.00%</b>	50.00%	75.00%	41.67%
happy	100.00%	100.00%	91.67%	<b>100.00%</b>	100.00%	91.67%	91.67%
sad	8.33%	8.33%	8.33%	<b>8.33%</b>	8.33%	8.33%	8.33%
surprised	75.00%	75.00%	75.00%	<b>83.33%</b>	75.00%	66.67%	75.00%
<b>overall</b>	<b>64.29%</b>	<b>65.48%</b>	<b>65.48%</b>	<b>71.42%</b>	<b>64.29%</b>	<b>59.52%</b>	<b>60.71%</b>

*Poly1* stands for Polynomial and has degree parameter:  $D = 2$  and  $\gamma = 0.001953125$

*Poly2* stands for Polynomial and has degree parameter:  $D = 3$  and  $\gamma = 0.001953125$

*RBF1* has cache and  $\gamma$  parameters:  $C = 128$  and  $\gamma = 0.0078125$

*RBF2* has cache and  $\gamma$  parameters:  $C = 8192$  and  $\gamma = 0.00048828125$

*Sigmoid1* has cache and  $\gamma$  parameters:  $C = 128$  and  $\gamma = 0.0078125$

*Sigmoid2* has cache and  $\gamma$  parameters:  $C = 8192$  and  $\gamma = 0.00048828125$

A model has also been trained with same kernels and parameters but without cross-validation. Comparison has been made between the model with cross validation and the one without cross validation as it is shown in Table 13.8. Results are consistent whether cross-validation is used or not, and in contrary to non-weighed LBP, best results are achieved using cross-validation.

The best accuracy percentage is for a model trained with RBF1 kernel parameters

**Table 13.8:** Results with and without cross validation

	with cross validation	without cross validation
Linear	64.29%	66.67%
Poly1	65.48%	65.48%
Poly2	65.48%	65.48%
RBF1	71.42%	63.10%
RBF2	64.29%	67.86%
Sigmoid1	59.52%	60.71%
Sigmoid2	60.71%	67.86%

and with cross-validation, with an overall accuracy rate of 71.42%. It is obtained with classification based on the RBF kernel and with parameters  $C = 128$  and  $\gamma = 0.0078125$ . Table 13.9 represents the confusion matrix for this kernel.

**Table 13.9:** Confusion matrix

	neutral	afraid	angry	disgusted	happy	sad	surprised	accuracy
neutral	<b>11</b>	0	0	0	0	1	0	91.67%
afraid	0	<b>12</b>	0	0	0	0	0	100.00%
angry	2	0	<b>8</b>	0	1	1	0	66.67%
disgusted	1	2	1	<b>6</b>	2	0	0	50.00%
happy	0	0	0	0	<b>12</b>	0	0	100.00%
sad	2	8	1	0	0	<b>1</b>	0	8.33%
surprised	0	2	0	0	0	0	<b>10</b>	83.33%

As for the first test result, two emotions are hardly recognized by our system: *sad* emotion and *disgusted* emotion (see Table 13.10), this time not the *angry* emotion.

**Table 13.10:** Recognition accuracy of the six basic emotions and neutral state

$\leq 50\%$	$> 66.67\%$
disgusted (6/12) sad (1/12)	afraid (12/12) angry (8/12) happy (12/12) surprised (10/12) neutral (11/12)

# Chapter 14

## Issues

### 14.1 Feature extraction

As seen in Chapter 6, there are many existing ways to improve the basic LBP operator. Indeed, using a circular LBP, detecting uniform patterns or assigning weights to different regions of the image can improve facial expression recognition and yield better results than the basic LBP operator. These improvements however need more computation time, which can lead to a slower system.

The LBP operator used in this system already has some improvements; it is a uniform circular LBP operator, with a radius  $R = 1$ , which is equivalent to a regular LBP operator. Results obtained with this operator are quite good, with an accuracy of 66.67%. This accuracy rate can be improved by weighting each region of the face, or applying a different radius in order to detect other kinds of patterns. This system has however trouble recognizing some of the 7 emotions, especially the *sad* one.

This feature extraction method was chosen because the LBP operator gives good results and has a high discriminative power. Moreover, the basic LBP operator is rather easy to implement, and can be improved in many ways afterwards. But there are most likely other methods that can output equal or higher accuracy rates than LBP. There are also certainly other more optimized methods using less computation time than LBP. Even if the LBP operator seems to be a good compromise between computation time and results, other methods can be implemented to compare the performance of each algorithm with the same test conditions.

### 14.2 Real-time

Computation of feature vector for an image using LBP takes about 1-4 seconds, depending on the computer, which as not expected. Indeed, the computation time was expected to be a matter of milliseconds. This time consuming process can explain why this facial expression recognition system does not really work in real-time, and also why the subject has to trigger the system. With the Kinect, 30 frames are

received per second. The system should hence need 33.3 milliseconds per frame maximum ( $\frac{1}{30} = 0.0333s$ ) to be able to work in real-time and have the time to process each frame.

The basic LBP operator is quite simple, so a system based on this feature extraction method should be able to work in real-time. Using the circular LBP operator adds some computation time; but the use of the uniform LBP allows reducing the computation time by only considering uniform patterns and not non-uniform ones.

Because all frames cannot be processed, the system was adapted to work with video sequences and almost in real-time. The subject stands in front of the Kinect and express an emotion among all recognized ones (6 basics plus neutral one). When the subject thinks the expressed emotion is recognizable, he or she can click on the interface to trigger the process with a screenshot of the facial expression he or she had at that time. This frame is processed and classified and the output given to the subject is the name of the emotion that he expressed. When working in real-time conditions, the output is given about 2 seconds after the subject's click.

### 14.3 Issues with the implementation on the Kinect

At the beginning of the project the system was designed to work in real-time. First versions of the system were implemented with a linear structure, which fetches all images from the Kinect video stream. Since the implementation of LBP feature extraction into the system, the system needed a few minutes to process the video stream. The Kinect's camera sends 30 images per second, but the following LBP processing was slowing down the whole system to a critical point. The main architecture of the system then had to be changed, the main issue at this point being to retain the key points of our implementation while allowing a smoother processing.

This was done by modifying the way images are chosen and processed. Indeed, image retrieval from the Kinect is made through blocking functions, in order to wait for data coming from the camera sensor. This synchronous approach is not compatible with a simple thread management, which is the way our system is implemented. A solution to bypass this problem was to introduce more user interaction, which is materialized by the triggering of a key to start the facial expression recognition process.

The second issue with the Kinect is directly linked to the previous one. Indeed, since a key has to be triggered to start the process, a graphical user interface was contemplated, using GTK or Qt. However, it involved a lot of third-party libraries with their own event loops, which conflicted with the Kinect image retrieval event loop. The system consequently has no GUI, and the output is displayed in the console. It

might be possible to provide the system with a more user-friendly interface, but it is not implemented in this facial expression recognition system.

## 14.4 Training dataset

The KDEF database is a large database, with many different subjects photographed in many different positions. It is indeed sufficient for most facial expression recognition systems. However, some issues can be raised concerning this database. First, its subjects are all Caucasian, which can lead to erroneous results when testing the system with people from non-Caucasian origins.

A second point about spontaneous can be mentioned. Indeed, this database contains subjects displaying posed expressions, which binds the system to recognize posed expressions. If this facial expression recognition system is intended to be used with spontaneous expressions, it will have to be trained with a more appropriate dataset.

# Chapter 15

## Conclusion

### 15.1 Theoretical framework

The theoretical framework on which this system is based relies on three major points: face detection, feature extraction and classification. Face detection is achieved through Viola-Jones face detection algorithm. This cascade of AdaBoost classifiers based on Haar-like features results in a strong binary classifier, composed of a chain of small weak ones. Moreover, the use of integral images enables real-time detection.

Feature extraction is performed using Local Binary Patterns, a simple but efficient texture descriptor encoding changes in micro-patterns. It outputs histograms which bin indexes are intensity values of the pixels, and bin sizes the number of pixels having this value. This texture descriptor can be used for facial feature extraction, and can be modified in order to detect patterns of various sizes and shapes, but also to reduce histogram size, hence improving computation time.

Thirdly, Support Vector Machine is used as classifier. This binary linear classifier, belonging to the supervised learning category, is based on *margin maximization* and *kernel functions*. Indeed, since it is a linear classifier, data has to be mapped into a higher space for the classifier to perform correctly. Four classic kernels can be used, and even combined if needed. When compared to other classification methods, SVM yields better results, which makes it a suitable choice for facial expression recognition using Local Binary Patterns.

### 15.2 Results

Different results are obtained based on the use of the Kinect or not.

Results obtained by using face images of the KDEF database as test set are quite good, with an accuracy rate of 66.67%. There is nevertheless an issue with the *sad* emotion, which stands out because of its bad recognition accuracy. This emotion is mostly misclassified as *neutral* state. Indeed these two emotions are quite simi-

lar, almost identical when it comes to feature extraction, hence the low accuracy rate.

The *angry* emotion also has a below average recognition accuracy, but it is not as significant as for the *sad* emotion. The reason behind this result is the same as for the *sad* emotion. Since an *angry* face is more distorted than a *sad* face it is more easily classified correctly. The accuracy rate is thus of 41.67% for the *angry* expression, instead of 8.33% for *sad*.

Concerning results obtained with the Kinect, the accuracy rate is significantly lower.

## 15.3 Future work

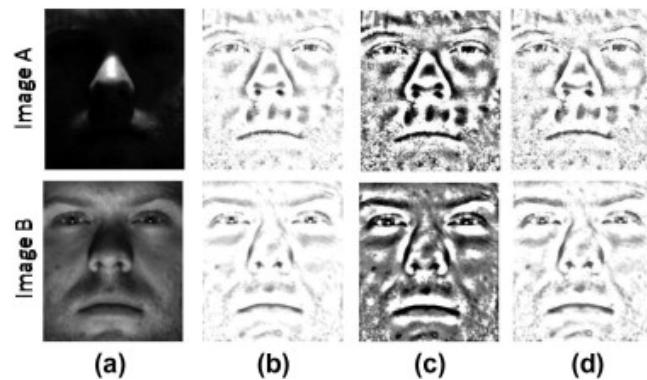
### 15.3.1 Local Binary Patterns

One way to improve the accuracy would be to use a LBP operator with a larger scale, hence modify the radius of the circular operator; for example,  $LBP_{12,2.5}^{u^2}$  or  $LBP_{16,4.0}^{u^2}$  (with  $P = 12$  and  $R = 2.5$  or with  $P = 16$  and  $R = 4.0$ ). This implies to use the bilinear interpolation because sampling points do not fall exactly on pixels, as seen in Chapter 6. However, using bilinear interpolation is more computationally expensive. A good compromise has to be found between computation time and accuracy rate.

### 15.3.2 Combination of feature extraction methods

To improve the accuracy of LBP feature extraction method, another feature extraction method can be used and combined with it.

For example, a method has been proposed by Liao et al. [21], where they combine LBP and Gabor filter. It is called Dominant Local Binary Patterns (DLBP), and is robust against change of lighting, image rotation and image noise. It works by using the most recurrent patterns of the LBP method to obtain more information on the texture. It also uses the Gabor method to add global texture information to one already obtained by LBP. It works based on the circularly symmetric Gabor filter responses [21]. Figure 15.1 contains two face images of the YaleB face database, and shows the robustness of the combination against change in lighting. Image (a) is the original image with different lighting conditions, and image (b) is the preprocessed image with the Gabor wavelets. Image (c) is the same image, mapped with the LBP operator. Finally, image (d) is the preprocessed image with the combination of the two methods [13].



**Figure 15.1:** 2 face images of the YaleB face database (a) when processed with Gabor wavelets (b), LBP (c), and Gabor wavelets+LBP (d)[13]

LBP has also been combined with the Scale Invariant Feature Transform descriptor (SIFT), which is a ROI descriptor. This descriptor is robust against image rotation, image translations, scaling and lighting variations. Heikkila et al. introduced a combination of the SIFT descriptor with the LBP operator [15].

### 15.3.3 Use of the Cohn-Kanade Database

The initial idea was to use the CK database for our facial expression recognition system. There has however been a problem of time; indeed, between the moment when the permission of using the database was asked and the moment we have been allowed to access it, almost three months passed. The KDEF database was used instead.

It would yet be interesting to run our system on this database because it is one of the most complete databases, and its image sequences are also of high quality. In the second version of the database, spontaneous expressions have been added, which might help resolve the accuracy issue met by the system when it runs in real-time with the Kinect.

# Bibliography

- [1] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(12):2037–2041, 2006.
- [2] Keith Anderson and Peter W. McOwan. A real-time automated system for recognition of human facial expressions. *IEEE Trans. Systems, Man, and Cybernetics Part B*, 36(1):96–105, 2006.
- [3] Marian Stewart Bartlett, Gwen Littlewort, Ian Fasel, and Javier R. Movellan. Real time face detection and facial expression recognition: Development and application to human computer interaction. *Proc. CVPR Workshop on Computer Vision and Pattern Recognition for Human-Computer Interaction*, 5:53, 2003.
- [4] Vinay Bettadapura. Face expression recognition and analysis: The state of the art. *Tech Report*, 2012.
- [5] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer Science, 2006.
- [6] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [7] Claude C. Chibelushi and Fabrice Bourel. Facial expression recognition: A brief tutorial overview. [http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL\\_COPIES/CHIBELUSHI1/CCC\\_FBFacExprRecCVonline.pdf](http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/CHIBELUSHI1/CCC_FBFacExprRecCVonline.pdf), 2003.
- [8] Charles Darwin. *The Expression of the Emotions in Man and Animals*. John Murray, 2. ed. edition, 1904.
- [9] Fabrizio Dini. An application of viola-jones algorithm: face detection and tracking. <http://www.micc.unifi.it/dini/download/dbmm2008-Dini.pdf>, 2008.
- [10] Gianluca Donato, Marian Stewart Bartlett, Joseph C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(10):974–989, 1999.
- [11] Xiao Feng, Matti Pietikainen, and Abdenour Hadid. Facial expression recognition with local binary patterns and linear programming. *Pattern Recognition and Image Analysis*, 15(2):546–548, 2005.

- [12] Abhiram Ganesh. Evaluation of appearance based methods for facial expression recognition. *Technische Universiteit Eindhoven*, 2008.
- [13] Yi Zheng Goh, Andrew Beng Jin Teoh, and Michael Kah Ong Goh. Wavelet local binary patterns fusion as illuminated facial image preprocessing for face verification. *Expert Systems with Applications*, 38(4):3959–3972, 2011.
- [14] Adam Harvey. Adam harvey explains viola-jones face detection. [http://www.cognotics.com/opencv/servo\\_2007\\_series/part\\_2/sidebar.html](http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html), 2012.
- [15] Marko Heikkila, Matti Pietikainen, and Cordelia Schmid. Description of interest regions with local binary patterns. *Journal Pattern Recognition*, 42(3):425–436, 2009.
- [16] Robin Hewitt. How face detection works. [http://www.cognotics.com/opencv/servo\\_2007\\_series/part\\_2/sidebar.html](http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html), 2007.
- [17] Yousra Ben Jemaa and Sana Khanfir. Automatic local gabor features extraction for face recognition. *International Journal of Computer Science and Information Security*, 3(1), 2009.
- [18] Bram K. Julsing. Face recognition with local binary patterns, 2007.
- [19] Takeo Kanade, Jeffrey F. Cohn, and Yingli Tian. Comprehensive database for facial expression analysis. *Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 46–53, 2000.
- [20] Emotion Lab. Karolinska directed emotional faces (kdef). <http://www.emotionlab.se/resources/kdef>.
- [21] S Liao, Max W. K. Law, , and Albert C. S. Chung. Dominant local binary patterns for texture classification. *IEEE transactions on image processing*, 18(5):1107–1118, 2009.
- [22] Weifeng Liu, Yanjiang Wang, and Shujuan Li. Lbp feature extraction for facial expression recognition. *Journal of Information and Computational Science*, 8(3):412–421, 2011.
- [23] Michael Lyons. The japanese female facial expression (jaffe) database. <http://www.kasrl.org/jaffe.html>.
- [24] Aleix Martinez and Robert Benavente. The ar face database. <http://www-sip1.technion.ac.il/new/DataBases/Aleix%20Face%20Database.htm>.
- [25] University of British Columbia. The viola/jones face detector, 2001.
- [26] Timo Ojala, Matti Pietikainen, and David Harwood. A comparative study of texture measures with classification based on feature distributions. *Pattern Recognition*, 29:51–59, 1996.

- [27] Maja Pantic and Leon J.M. Rothkrantz. Automatic analysis of facial expressions: the state of the art. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12):1424–1445, 2000.
- [28] Lekshmi V Praseeda and M Sasikumar. Facial expression recognition from global and a combination of local features. *IETE Tech Rev*, 26(1):41–46, 2009.
- [29] N Sebe, M S Lew, Y Sun, I Cohen, T Gevers, and T S Huang. Authentic facial expression analysis. *Image and Vision Computing*, 25:1856–1863, 2007.
- [30] Caifeng Shan, Shaogang Gong, and Peter W. McOwan. Facial expression recognition based on local binary patterns: A comprehensive study. *Image and Vision Computing*, 27:803–816, 2009.
- [31] Padhraic Smyth. Face detection using the viola-jones method. *University of California, Irvine*, 2007.
- [32] UQUAM. The msfde. <http://www.er.uqam.ca/nobel/r24700/Labo/Labo/MSEFE.html>.
- [33] M F. Valstar and M. Pantic. Fully automatic facial action unit detection and temporal analysis. *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, page 149, 2006.
- [34] M F. Valstar, I. Patras, and M. Pantic. Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data. *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 3:76–84, 2005.
- [35] Paul Viola and Michael Jones. Robust real-time object detection. *Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling*, 2001.
- [36] Wikipedia. Statistical classification. [http://en.wikipedia.org/wiki/Classifier\\_\(mathematics\)](http://en.wikipedia.org/wiki/Classifier_(mathematics)).



# Appendix: Source Code

The full source code for the program is included on the CD-ROM attached here.