Facial Expression Recognition using Local Binary Patterns

with classification based on Support Vector Machines

AALBORG UNIVERSITY
Department of Electronic Systems
Vision, Graphics and Interactive Systems
9^{th} Semester project

Autumn 2012

Maxime Coupez
Kim-Adeline Miguel
Julia Alexandra Vigo

AALBORG UNIVERSITY

Department of Electronic Systems
Fredrik Bajers Vej 7
DK-9220 Aalborg Ø
http://es.aau.dk

Title:
Project Title

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: 1

Page Numbers: 89

Date of Completion:
December 11, 2012

Abstract:

Since the last decade, a lot of researches have been carried out about emotion recognition. The number of projects conducted in this field demonstrates the interest and the importance of systems which can recognize human mood.
In this project, an emotion recognition system is developed, using a Microsoft Kinect. This recognition is achieved in 3 steps: Face detection, extraction and classification of facial features, this structure being the usual modus operandi in emotion recognition research.
Face detection is performed using Viola-Jones' algorithm, then Local Binary Patterns (LBP) are used to extract facial features. Finally, Support Vector Machines (SVM) classify these features into six predefined emotions.
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 enough different faces to obtain a satisfying result.

Preface

This report documents the semester project entitled *Facial expression recognition using Local Binary Patterns*. 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 review 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 11, 2012

Maxime Coupez
<mcoupe12@es.aau.dk>

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

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

Abbreviations

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
PCA	Principal Component Analysiss
ROC	Receiver Operating Characteristic
SVM	Support Vector Machine

Contents

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.

Chapter 1

Motivations

A facial expression is a "visible manifestation of the effective state, cognitive activity, intent, personality, and psychopathology of a person" [8]; facial expressions represent a huge part in dialogue and interaction with other humans. Indeed, facial expressions carry more informations than speech, informations on which humans can relay for interaction. Facial expressions have a considerable effect on a listening interlocutor; in a conversation, it represents 55 percent of information received by a listener, while 38 percent are conveyed by voice intonation and the remaining 7 percent by the spoken words [21].

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 [6]. 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 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 an 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. We chose our system to be non-obtrusive, and will detail its setup further in the next section.

1.1 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 it, as seen in **Insert picture of the setting & ref to figure**. This camera provides a 640×480 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.

1.2 Facial Expression Datasets

Databases are very important for facial expression recognition system.

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.

Databases are however 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 lightning, 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 3 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.

1.2.1 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 [17].

This database contains only posed expressions. The 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 1.1. In this figure, this is a female subject displaying 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, surprised).

1.2.2 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

Figure 1.1: Example of images from JAFFE database

Neuroscience, Section of Psychology, Stockholm, Sweden [14].

The database was first developed for psychological and medical research purposes. It was created so it could 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 [14].

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) 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) [14].

An example of images contained in the database is given in Figure 1.2which represents a female subject photographed from a straight angle and displaying 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, surprised).

1.2.3 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

Figure 1.2: Example of images from KDEF database

express this emotion [28].

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 [28].

An example of images contained in the database is given in Figure 1.3, where an African female subject displays 7 different emotional expressions (neutral, happy, angry, afraid, disgusted, sad, ashamed).

Figure 1.3: Example of images from MSDFE database

Chapter 2

Facial expression recognition

After having stated the conditions and motivations of this project, we will now describe the general structure a 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 [5].

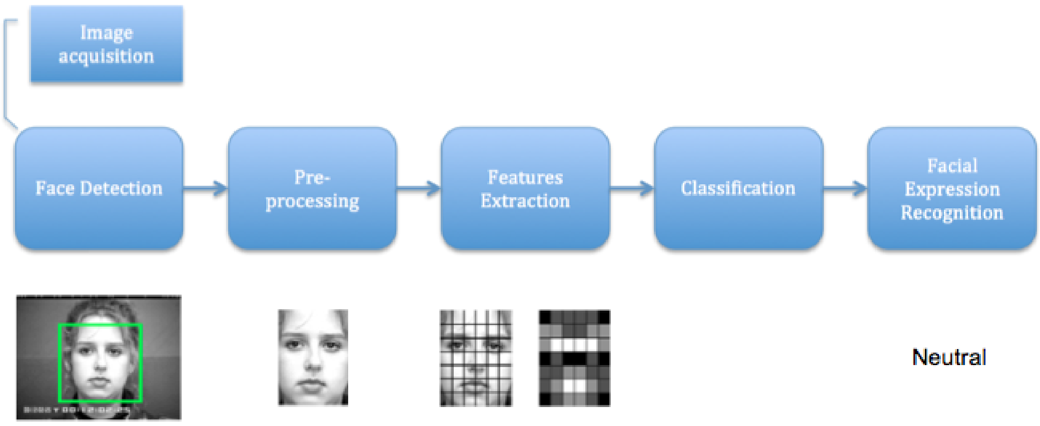

Figure 2.1: Facial Expression Recognition process

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 [5].

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" [5]. 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" [5]. There are a lot of features extraction methods. The most famous are : Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Local Binary Patterns (LBP), Hidden Markov Models (HMM), Eigenfaces, 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 Issues

2.2.1 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 [24] 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 the 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 [24].

The researches 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 express one emotion while feeling another one (for example, one subject was showing sadness while being happy) [24].

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 emotion, posed expressions databases will be used, but the result will not be close to reality.

2.2.2 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, the processing time should be taken into account. Indeed, if it is too long, the system will not be responsive enough to be labelled 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 Recognition or to improve and lighten already existing algorithms.

2.2.3 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 comparison of features, and if all the features cannot be compared because something is covering a part of the face, the 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. Some examples of images contained in this database are given by Figures 2.2 and 2.3 [18].

Figure 2.2: Example eye occlusion in the AR Face database

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 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

Figure 2.3: Example of face occlusion in the AR Face database

this problem, databases can include data with different lighting conditions. This is the case for the AR Face database and an example of the images contained in this database is given by Figure 2.4 [18]:

Figure 2.4: Example of different lighting conditions (from left to right: dark to bright) in the AR Face database

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, the 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, "Fear", from the KDEF database is given in Figure 2.5:

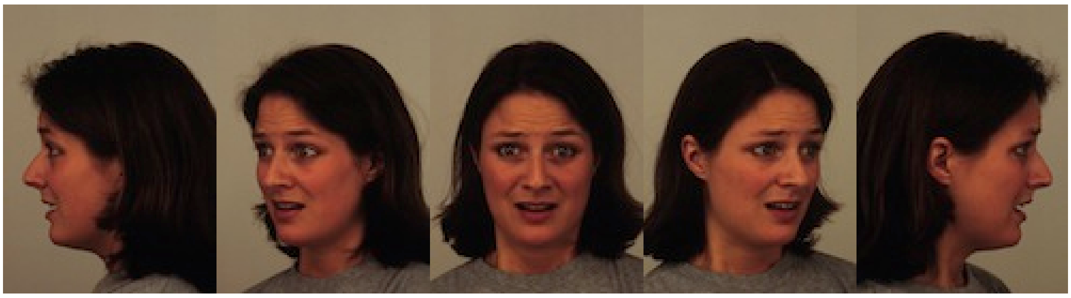

Figure 2.5: Example of different angles for the "Fear" emotion (from left to right: full left profile, half left profile, straight, half right profile, full right profile) in the KDEF database

2.3 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 the recognition process. Moreover, the intensity level of the light should be as close as possible to the one of the database. This way the lighting would not have any influence on the recognition process.

Part II

Feature detection

Contents

Before getting to the main part of this project which is Feature extraction, there is a mandatory step that is Feature 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. We will then introduce the Viola-Jones algorithm, a performant and cost-effective method for face detection and tracking.

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.

3.1 Detection

Detection is finding out if the input image or video sequence represents or contains a particular object, 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 [7].

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 [7]. For example, the output can look like Figure 3.1 [7].

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 [7]. There is an example of a binary classifier output in Figure 3.2 [7].

3.2 Classifiers

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

Figure 3.1: Example of an output of face detection

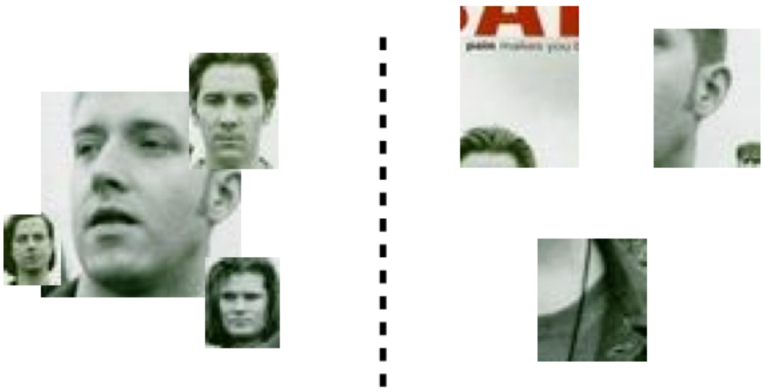

Figure 3.2: Example of a binary classifier output for face detection

distance between instances [32].

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. This method combine classifiers in "cascade". It allows to focus on the promising object-like regions by discarding the background in a very quick way [31].

4.1 Overview

The Viola-Jones algorithm works as following [7]:

- This algorithm is a "strong, binary classifier built of several weak detectors"
- "Each weak detector is a simple binary classifier"
- 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

The 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 [11].

Figure 4.1 shows some of the first Haar features in the original Viola-Jones cascade [11], while Figure 4.2 is an example from the extended set of features [7]. 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 [10].

Figure 4.1: Examples Haar features

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 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 [11]. 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 [10].

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 [10].

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.

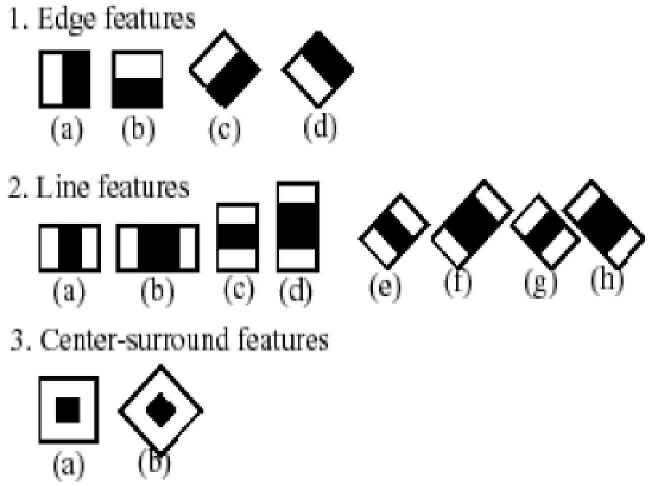

Figure 4.2: Extended set of features

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 [31].

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 [31].

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 [31].

Figure 4.3: Example of an early stage in the Haar cascade

4.3 Integral image

Viola and Jones used an intermediate representation of an image called "integral image". This integral image allows a very fast computation of rectangle features [31], 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 [11].

For a pixel P with coordinates (x, y), its value in the integral image is computer as

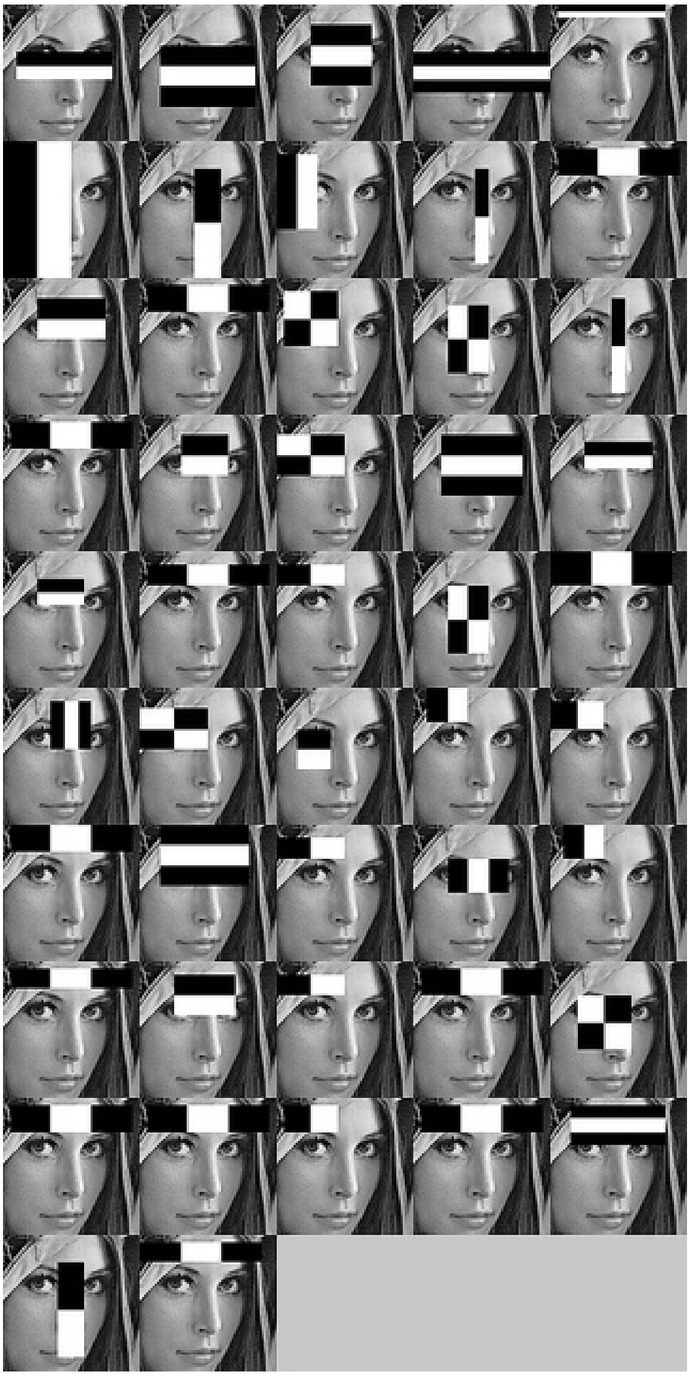

Figure 4.4: Example of a later stage in the Haar cascade

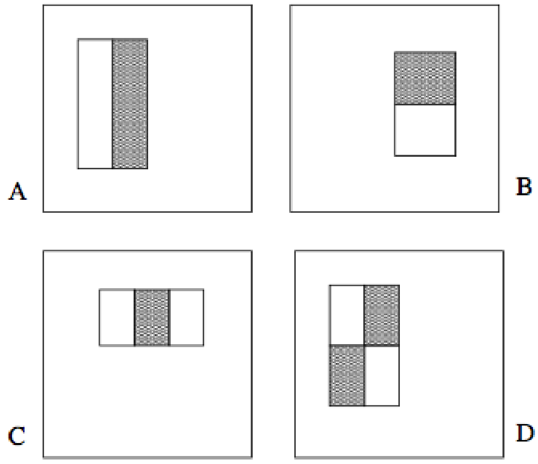

Figure 4.5: Example of the different kinds of rectangle features

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. 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 [31].

Using the following pair of recurrences:

$$s(x, y) = s(x, y - 1) + i(x, y) \tag{4.1}$$

$$ii(x, y) = ii(x - 1, y) + s(x, y) \tag{4.2}$$

(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 $4+1-(2+3)$ [31].

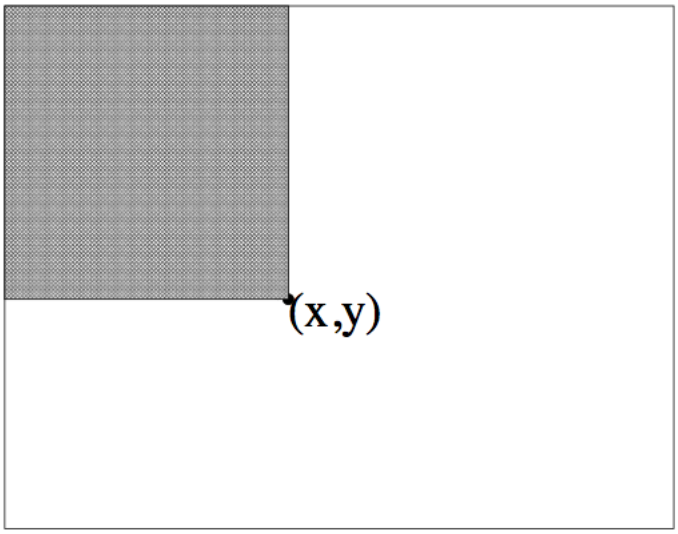

Figure 4.6: Integral image

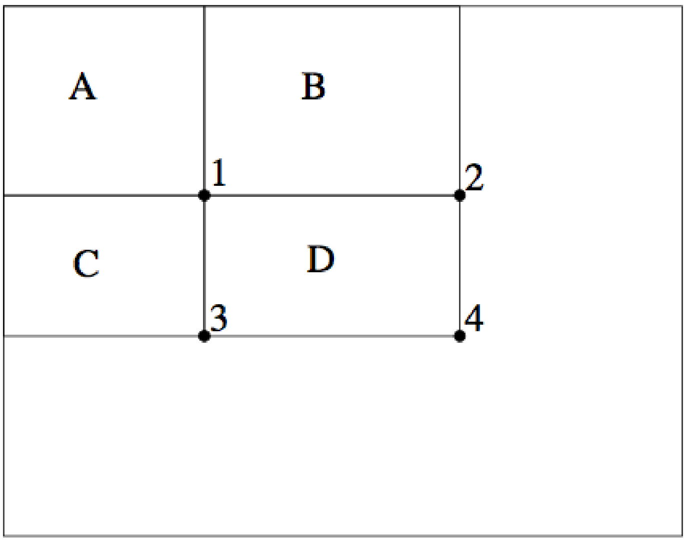

Figure 4.7: Integral image with four array references

4.4 Weak classifiers and AdaBoost

Features are extracted from a sub-region of an input image, which size is usually of 24×24 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 [26].

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 the 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 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 [11].

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 [11]. One of the difficulties and challenges for this learning algorithm is to associate large weights to efficient classifiers, and smaller weights toeach poor classifiers. In order to succeed in selecting a small group of good classifiers but with significant variety, AdaBoost is an aggressive algorithm [31].

Experiments have been conducted with a classifier built from 200 features and using AdaBoost. The classifier had a detection rate of 95%, and obtaind only 1 false positive out of 14084 negative samples from the training dataset (see figure 4.9)[31].

This experiment, points out that a 200-feature classifier is an efficient technique for object detection. It also means that a boosted classifier constructed from rectangle features is an efficient technique for object detection. Although the results of this experiment are convincing in terms of detection, they may not be performant enough for real-world tasks. This boosted classifier requires 0.7 seconds to scan a 384×288 pixel image. Regarding the computation time, it is probably faster than any other existing system. In order to improve the system so that 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; it will increase computation time [31].

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 a hence its name "Classifiers cascade". This chain is composed of a separate AdaBoost

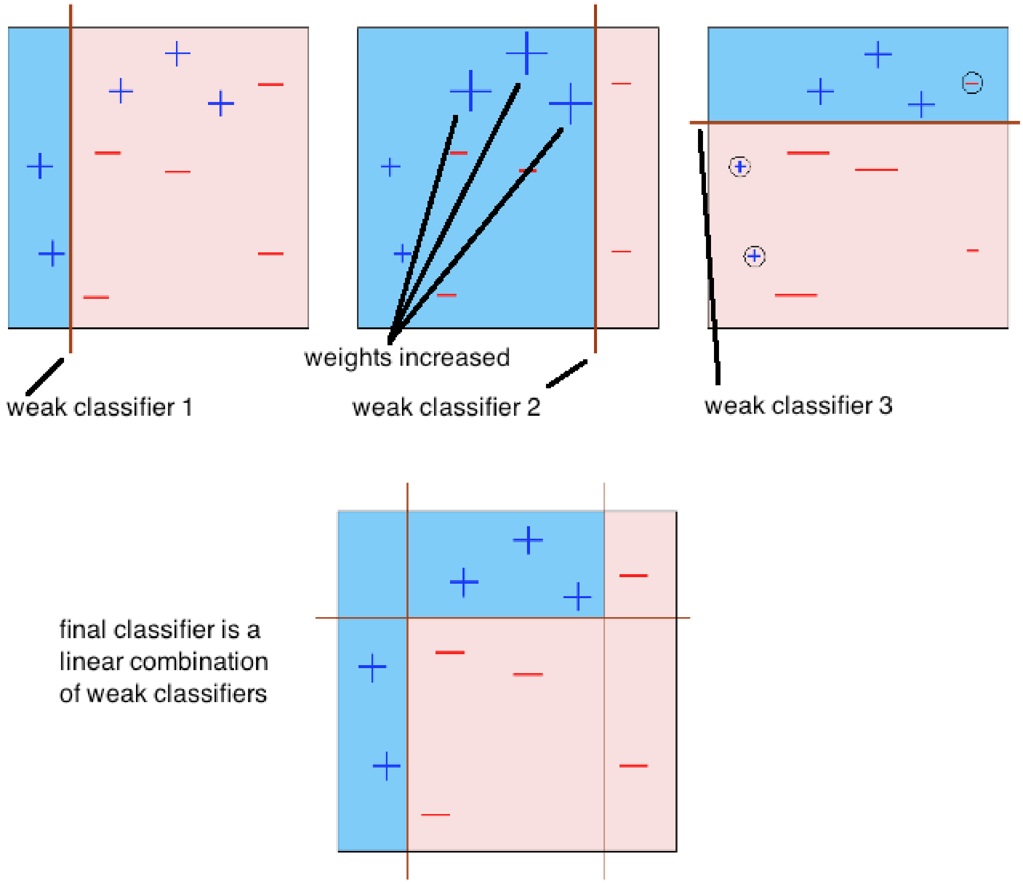

Figure 4.8: AdaBoost method

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" [11]. Using this algorithm with the classifiers cascade method allows to reduce significantly the computation time and increase significantly the detection performance [31].

Tests were conducted 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

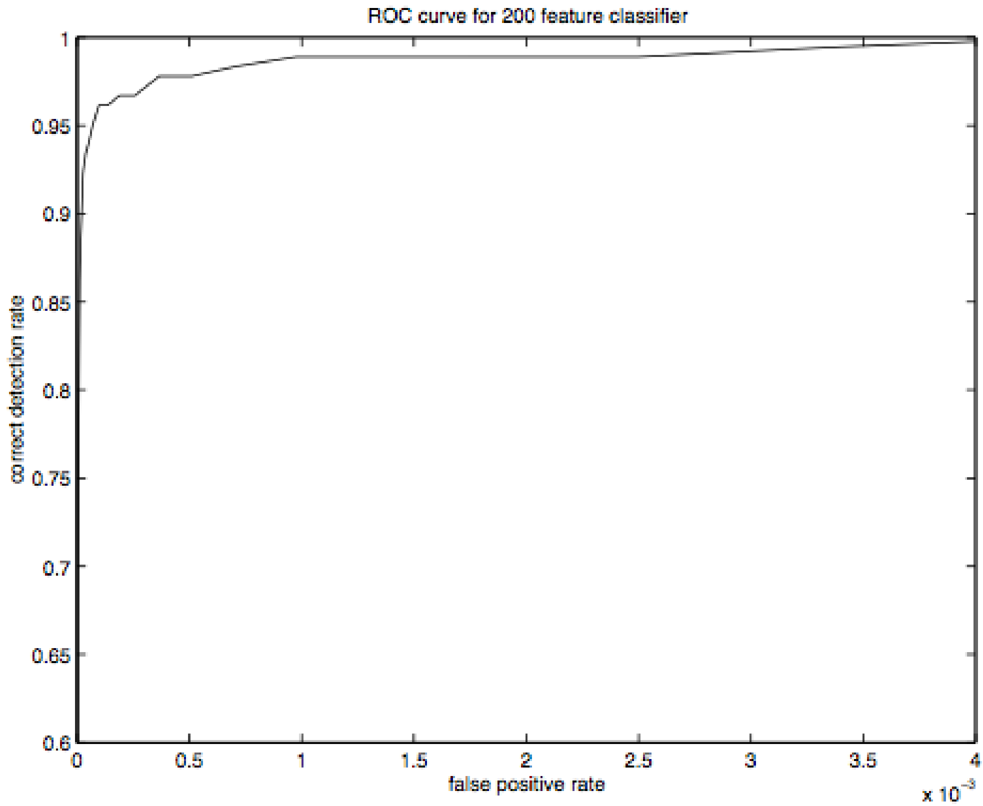

Figure 4.9: Receiver Operating Characteristic (ROC) curve for the 200 feature classifier

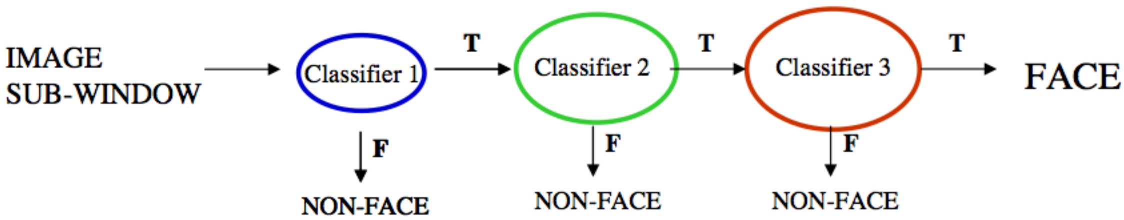

Figure 4.10: Cascade of boosted classifiers

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 [31].

The cascade method has been made in a way that there must not be false negative

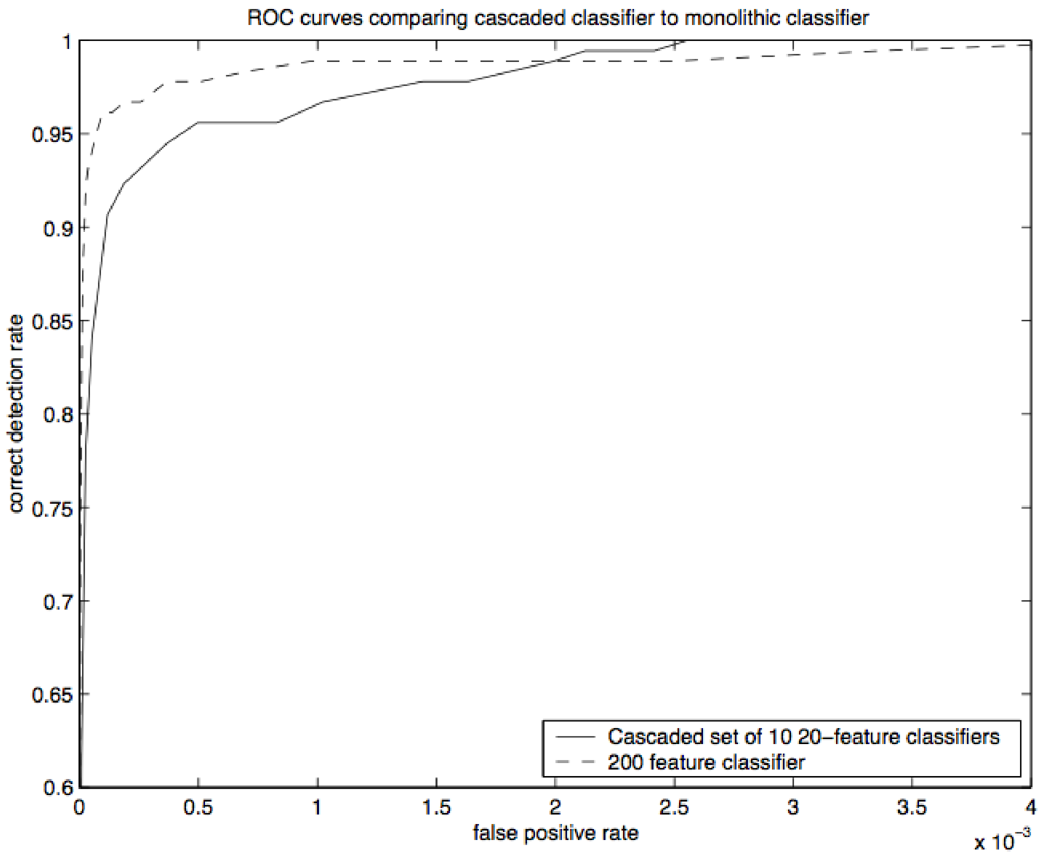

Figure 4.11: ROC curves of a 200-feature classifier and of a classifier cascade containing 10 20-feature classifiers

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 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" [11].

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 [31].

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 [11].

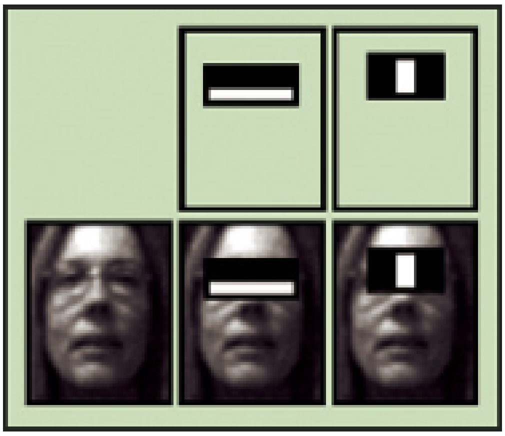

Figure 4.12: The first two Haar features in the original Viola-Jones cascade

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 [31].

Following are the different numbers about cascade classifiers (see Figure 4.13) [19]:

- 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 24×24 pixels. All these faces were chosen

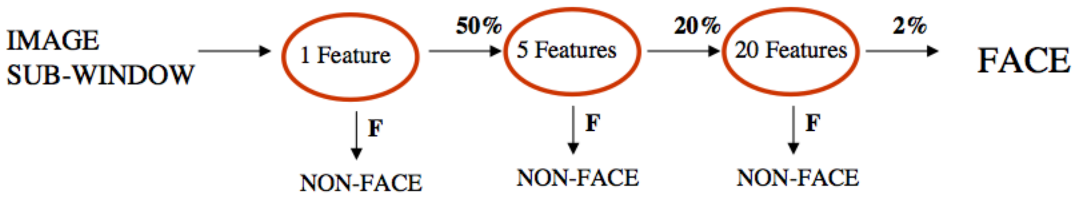

Figure 4.13: Cascade of boosted classifiers positive rate

Figure 4.14: Examples of frontal upright face images used for training

randomly on the internet. Some face examples are shown in figure 4.14 [31].

To sum up, the training set is composed of [19]:

- about 5,000 faces

    - 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)

There are usually two parts into a test set, first part being training set and second part being validation set. The training set is typically 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) [7]. For this kind of training, and with a 32 layer classifier, the total time usually exceeds several weeks [31].

Viola-Jones training stage proceeds with the following step [7]:

- With a defined number K of features (about 160,000 for a 24×24 grayscale image)
- The number L of wanted stages has to be fixed for the cascade
- Then it has to be done over and over again until the number L weak classifiers is reached:
    - With the data that has be weighted again in the previous stage
    - all the number K of weak classifiers have to be trained (the best threshold has to be find to classify in the best way the training set)
    - The best classifier has to be chosen at this stage
    - The data is then weighted again

As said previously, depending on the efficiency of a weak classifier, a weight is associated to it, depending on its classification error rate. Weak classifiers are then linearly combined depending on these weights, this operation having a huge computational cost [7].

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.

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 informations about spacial configuration of facial features, but can also encode informations about shape, texture or movement of the image's content [5].

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 already exists; the state of the art of facial expression recognition system. 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, Hidden Markov Models, Eigenfaces.

The second type of feature extraction algorithms are feature-based algorithms. These methods tend to locate important features, and build the 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 features regions, which are themselves translated into feature vectors.

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

Before developing a facial expression recognition project, it is important to know what already exists; the state of the art of facial expression recognition system. 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 *geometry-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, Hidden Markov Models, Eigenfaces.

The second type of feature extraction algorithms are geometry-based algorithms. These methods tend to locate important features, and build the 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 features regions, which are themselves translated into feature vectors.

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

5.1 Appearance-based methods

As said in the introduction, the Appearance-based method aims to extract the appearance changes of the face. In order to do that the method uses image filters that are applied to the whole face or that are applied to specific parts of the face [25]. With this kind of method, if there is any change in the lightning or the pose of the head, the 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 [9]. Its output is a new coordinate system with lower dimensions, obtained from transformed high dimensionality of data, while preserving the most important information. Since it is a statistical method, it can also be used in the classification step.

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 [9]. Like PCA, LDA can also be used as a classification algorithm.

5.1.3 Eigenfaces

Eigenfaces are a set of eigenvectors. These eigenvectors are derived from the covariance matrix of a set of images; and this in a high-dimensional vector space. The eigenvectors are ordered and each one represents the different amount of the variation among images. Characterization of the variation between face images is then possible [27]. Figure 5.1 shows of the first 28 basis vectors of Eigenfaces of the face images.

5.1.4 Hidden Markov Models (HMM)

These models are a set of statistical models used to characterize the statistical properties of a signal [23]. It can be used as a classification algorithm, and can also be developed to recognize expressions based on the "maximum likelihood decision criterion" [15].

Figure 5.1: First 28 basis vectors of Eigenfaces of the face images

5.2 Feature-based methods

As said in the introduction, the feature-based method aims to locate important facial components. The shape and the location of these facial features are extracted from an face image and then are concatenated into a features vector. This features vector represents the geometry of the face [25].

Compared to Appearance-based method, the feature-based method has equal or better results. This was recently tested by Valstar et al. [30] [29].

5.2.1 Gabor Wavelets

The most used Feature-based method is the Gabor wavelets because of their 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 5.2). These representations of face display locality and orientation performance [12]. Indeed, gabor kernels represent features of the face and can be place wherever if fits on the face and with whatever orientation. The disadvantage of this method is that it needs reliable and accurate feature detection and also tracking if necessary [25]. Another disadvantage of the Gabor feature extraction is processing time. This algorithm is very time-consuming, and dimensions of resulting vectors are very large [22].

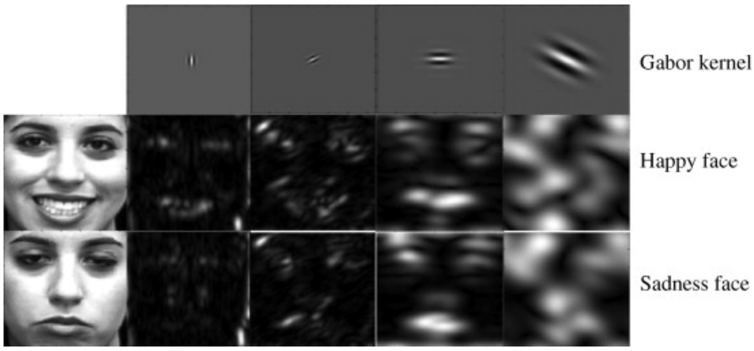

Figure 5.2: Examples of Gabor wavelets and corresponding convoluted images

5.2.2 Local Binary Patterns (LBP)

The Local Binary Pattern 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. These informations are the output of the thresholding of intensity values from the neighborhood pixels with the intensity value of the central pixel [9]. One of the advantages of this method is that it is effective even when there is change in lightning (see figure 5.3).

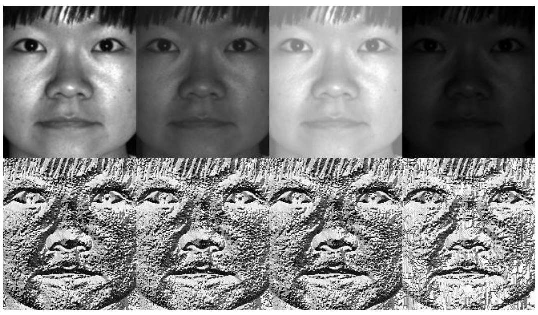

Figure 5.3: Effect of changes in lighting with LBP method

This method is more detailed in Chapter 6. This is this 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 the implementation offers choices. This method was chosen over the Gabor wavelets one because the Gabor wavelets are known to be time consuming. And this Facial Expression Recognition system is wanted to be as close as possible of real-time system.

Chapter 6

Local Binary Patterns

Some feature extractions 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 Pattern (LBP) feature extraction is a widely used a 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. [20]. 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 [13].

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 superior 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 [13].

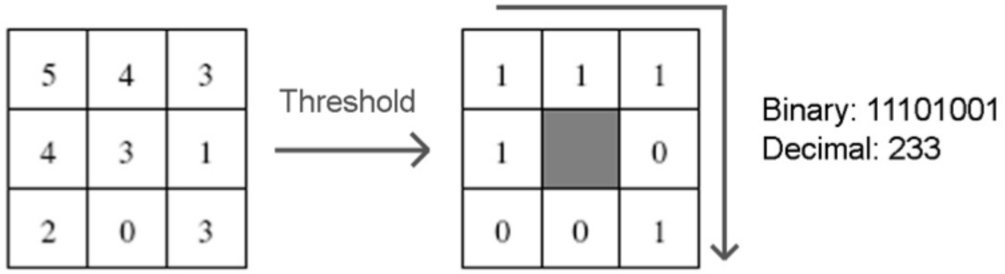

Figure 6.1: Basic LBP operator

Figure 6.2 shows an example of the LBP operator applied on a facial image from the JAFFE database [16].

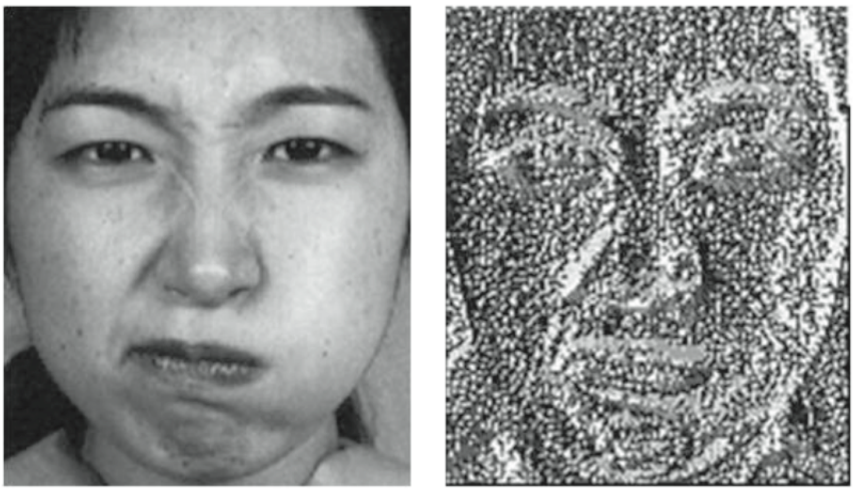

Figure 6.2: Example of the LBP operator applied on a image from the JAFFE database

6.2 Improvements

6.2.1 Circular LBP

In order to be able to use the LBP operator at different scales, a new form of the operator has been introduced. This new form is the circular LBP operator. This way, the operator can be extended to other neighbourhoods than only eight pixels. [9].

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 [9]:

- 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 pth neighbor pixel are calculated with the following formulas [13]:

$$x = x_c + R\cos(2\pi n/P) \tag{6.1}$$

$$y = y_c + R\sin(2\pi n/P) \tag{6.2}$$

Figure 6.3 shows the circular LBP operator with different numbers of neighbor pixels P and with various radius sizes [13]. For the circular LBP operator, the following notation is used: (P, R) and for a pixel p, the following notation is used: $LBP_{P,R}(p)$ [9].

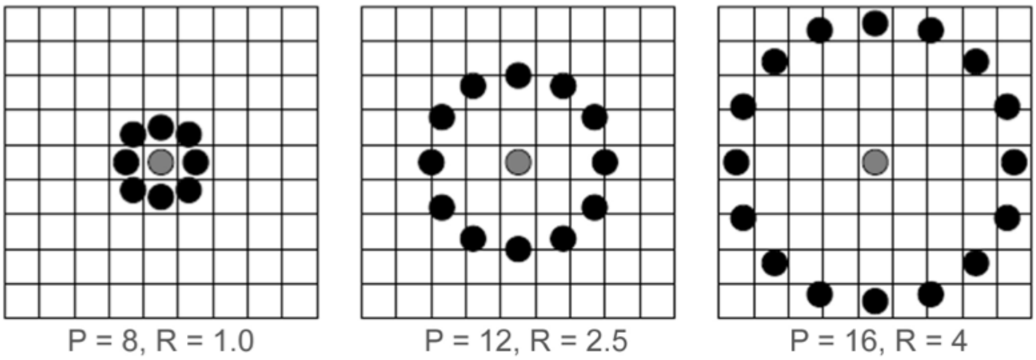

Figure 6.3: Circular LBP operator with different radius sizes R and different number of neighbor pixels P

Most of the time, when a circular LBP operator is used, coordinates of the neighbor pixels (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 [9].

The LBP operation outputs a texture, which formula is [9]:

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

With I_C, the intensity value of central pixel C, I_p for $p = 0, 1, ..., P-1$, the intensity values of the P neighbor pixels, and T, the texture in the local neighborhood of C [9].

This texture can also be calculated in an other way. The central pixel intensity value can be substracted 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 [9]:

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

Based on the work of Ojala et al. [20], $t(I_C)$ describes the overall luminance of an image, 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 [9]:

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

The above formula makes the texture description invariant against shifts in intensity values. t 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 [9]:

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

where,

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

For the pixels with an intensity value superior or equal to the one of the central pixel, the sign assigned would be 1. For the pixels with an intensity value inferior to $I_C$it would be 0 [9].

The last step is to assign a binomial weight to each sign, for the LBP operation. These weights are summed, producing an improved LBP code for central pixel C with coordinates (x_C, y_C) [9]:

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

The use of the formula above to calculate LBP still outputs a large number of patterns, which does not help reducing the data size. An other improvement to the 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 [9].

In fact, there can be only 2 or 0 bitwise transitions for a uniform LBP. Indeed, the neighborhood is circular so there cannot be 1-bitwise transition in the pattern. Indeed, if there is a bitwise transition in the neighborhood, i.e from 1 to 0, it means that there will be another bitwise transition from 0 to 1 further in the circular pattern [9].

If a pattern contains 0 bitwise transitions, it means that it is composed only of 0s or of 1s [9].

For a uniform LBP with a pattern-length of 8 bits (i.e 8 neighbors), 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 [9].

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 [9].

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 [9].

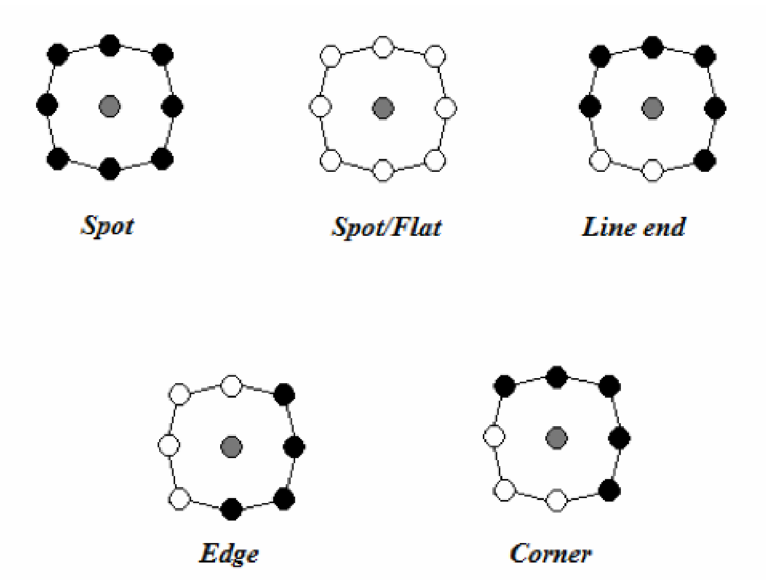

Figure 6.4: Patterns that can be detected with uniform LBP

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, pixel values obtained with the LBP operator are concatenated 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 $LBP_{8,2}^{u^2}$. Then, resulting values are gathered region-wise into different bins. For example, the LBP operator with $P = 8$ and $R = 1.0$ outputs a histogram with 59 bins: 56 bins for uniform LBP with 2 bitwise transitions, 2 bins for uniform LBP with 0 bitwise transition 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 [9].

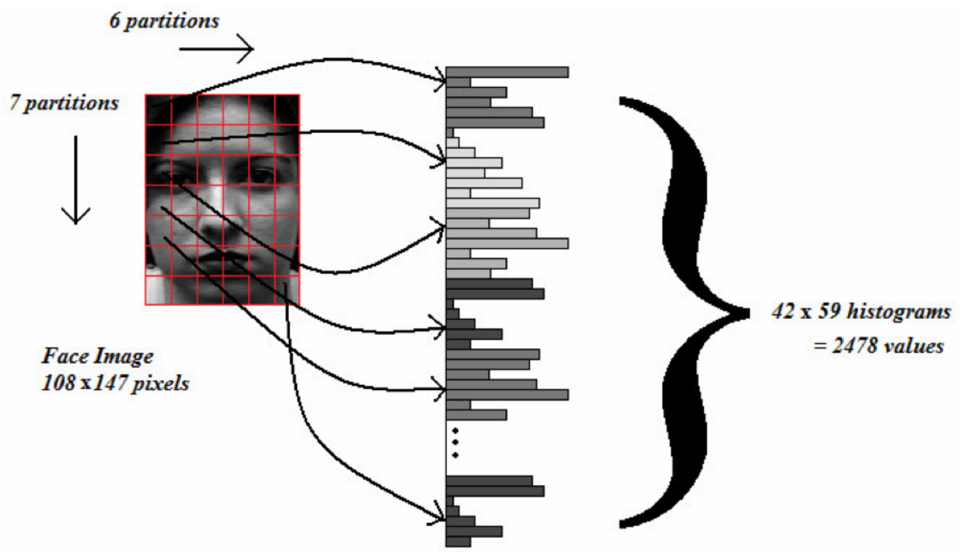

Figure 6.5: Example of a vector based on histogram computing

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 [9]. Thus, the histogram of each region is multiplied with a corresponding weight.

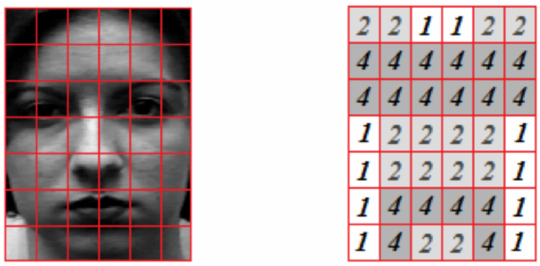

Figure 6.6: Weight assignment for each region

Part IV

Feature classification

Contents

This part will focus on the step following feature extraction, which is feature 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 last chapter will describe Support Vector Machine classification.

Chapter 7

Feature classification

Classification is done through machine learning algorithms. Machine learning, being a branch of Artificial Intelligence, aims to helps 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 so it can 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, an image of a face is a set of patterns of different sizes and shapes. Pattern recognition through machine learning can be divided into to 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 the next chapter, 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 Machine.

7.1.1 Naive Bayes classifier

The Bayesian classifier is based on Bayes theorem:

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

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

To introduce linear discrimination, it is important to understand what is classification. Classification is the process of assigning the input vector x to one of the classes C_i, $i = 1, ..., I$, hence learning the decision boundaries that separate the different classes in the input space [?]. 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

$$y(x) = w^T x + \omega_0 \tag{7.4}$$

With *weight vector* w^T and *bias* ω_0 [?]. 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.

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^{th} discriminant function will determine if a point belongs to class C_k or not. However, as shown in Figure 7.1, this combination of discriminant functions leaves an ambiguous region [?].

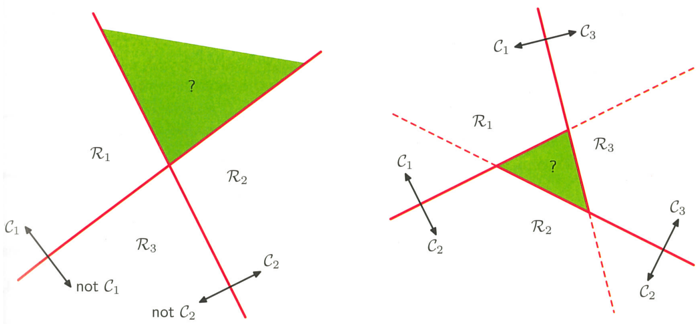

Figure 7.1: Attempting to construct a K class discriminant from a set of two class discriminants leads to ambiguous regions, shown in green. On the left is an example of *one-versus-the-rest* approach, the discriminant function designed to separate points belonging to a class C_i and points that are not. On the right is an example of *one-versus-all* approach involving 3 discriminant functions, each one separating a pair of classes classes C_i and C_j.From: Christopher M. Bishop, *Pattern Recognition and Machine Learning.* Copyright ©2006 by Springer Science.

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 an issue about ambiguous regions, as seen in Figure **??** [**?**].

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 $g(x)_i = \max_k g_k(x)$. Indeed, as shown in Figure **??**, there are no more ambiguous regions [**?**].

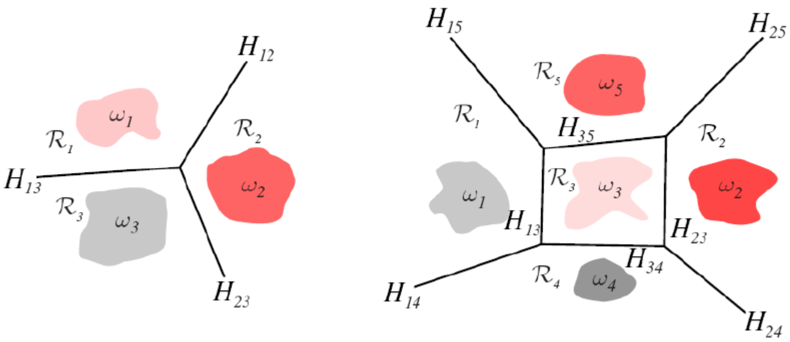

Figure 7.2: Decision boundaries produced by a linear machine for a 3-class problem and a 5-class problem. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright ©2001 by John Wiley & Sons Inc.

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 μ_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 μ_k is a minimum. In other words, equation 7.5, also called *distortion function* has to be minimized [**?**].

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

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, which result can be seen in Figure 7.3:

Initialization of reference vectors μ_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 μ_k, $k = 1..K$ **do**

$$\mu_k \leftarrow \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$

  end for
until μ_k converge

In this algorithm, the denominator $\sum_n r_{nk}$ corresponds to the number of points set to cluster k, which sets μ_k as the mean of the cluster, hence the name *K-means algorithm* [**?**].

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

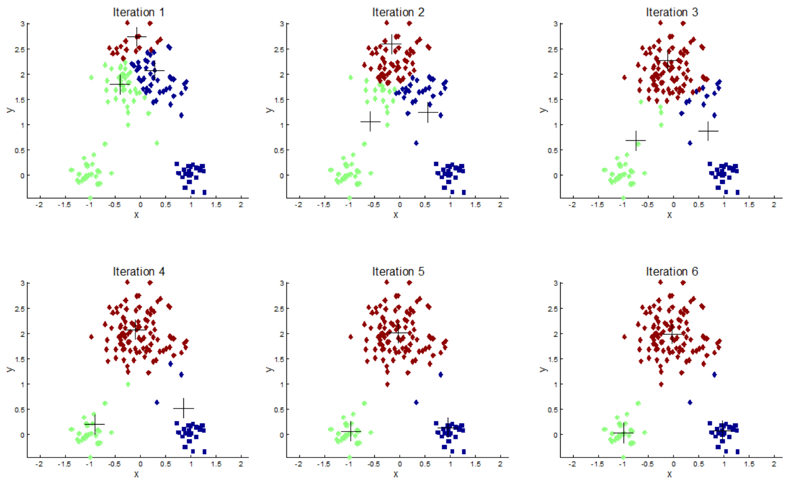

Figure 7.3: Example of application of the iterative K-means algorithm.

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 μ_k are also stored. Hence, during data reconstruction, each data point is approximated by corresponding vector μ_k. This approximation is called *vector quantization* [**?**].

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 can be written as $p(x|G_i) \sim \mathcal{N}(\mu_i, \Sigma_i)$, with mean vector μ_i and covariance matrix Σ_i, the aim is now to maximize the log likelihood function described in Equation 7.6.

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

with Φ 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 into a result comparable as the one in Figure 7.4 [**?**].

Initialize means μ_n, covariances Σ_n, mixing coefficients π_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 $\gamma(z_{nk})$:
  $\pi_k^{new} = \frac{N_k}{N}$
  $\mu_k^{new} = \frac{1}{N_k} \sum_{n=1} N\gamma(z_{nk}) x_n$
  $\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1} N\gamma(z_{nk})(x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$
  Where $N_k = \sum_{n=1} N\gamma(z_{nk})$
  Evaluate log likelihood $\mathcal{L}$
until log likelihood $\mathcal{L}$ converges

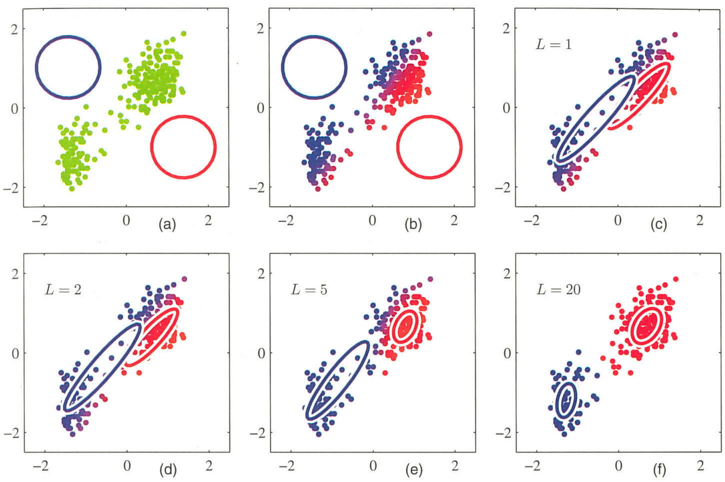

Figure 7.4: Illustration of the EM algorithm. From: Christopher M. Bishop, *Pattern Recognition and Machine Learning*. Copyright ©2006 by Springer Science.

Chapter 8

Support Vector Machine

Support Vector Machine (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 [?], 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 [?]. It can be adapted into a *relevance vector machine* to output posterior probabilities though [?], 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 the 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 $yf(x) \geq 1$, maximizing the margin is the action of maximizing $||w||^{-1}$, which is consequently equivalent to minimizing $||w||^2$ depending on this constraint. Margin maximization then requires to solve a *quadratic programming* problem under constraints, as seen in Equation 8.1.

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

8.3 Kernel function

It might however not be possible to perform this 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 8.2.

Linear kernel:	$K(x_i, x_j) = x_i^T x$	
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:	$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$	
		(8.2)

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 $\Phi(x_i)$. However, simple kernels are usually combined in order to build more complex ones.

8.4 Combining LBP and SVM

In a 2009 article, Shang and al [25] have performed facial expression recognition while using Local Binary Patterns for feature extraction, and comparing the accuracy of different kinds of classifiers: template matching, LDA and SVM. They have used images from the Cohn-Kanade database as train data, and the conclusion of their study is that classification using SVM has a high accuracy rate, as seen in Table 8.1.

Table 8.1: Recognition performance of LBP-based SVM with different kernels

	6-Class recognition (%)	7-Class recognition (%)
SVM (linear)	91.5 ± 3.1	88.1 ± 3.8
SVM (polynomial)	91.5 ± 3.1	88.1 ± 3.8
SVM (RBF)	92.6 ± 3.1	88.9 ± 3.5

Furthermore, as seen in confusion matrices 8.2 and 8.3, the accuracy for each facial expression is not the same. SVM has some difficulties especially when it comes to distinguish fear and sadness, the two facial expressions which have the lowest accuracy rates. Fear is mistaken with joy, while sadness is mistaken with anger or neutral state. Recognitions rates are however usually better for a 7-class classification, except for fear.

Since the accuracy of the system presented in this article is 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. We will describe further our implementation and results further in the report.

Table 8.2: Confusion matrix of 6-class facial expression recognition using SVM (RBF)

	Anger (%)	Disgust (%)	Fear (%)	Joy (%)	Sadness (%)	Surprise (%)
Anger	89.7	2.7	0	0	7.6	0
Disgust	0	97.5	2.5	0	0	0
Fear	0	2.0	73.0	22.0	3.0	0
Joy	0	0.4	0.7	97.9	1.0	0
Sadness	10.3	0	0.8	0.8	83.5	4.6
Surprise	0	0	1.3	0	0	98.7

Table 8.3: Confusion matrix of 7-class facial expression recognition using SVM (RBF)

	Anger (%)	Disgust (%)	Fear (%)	Joy (%)	Sadness (%)	Surprise (%)	Neutral (%)
Anger	85.0	2.7	0	0	4.8	0	7.5
Disgust	0	97.5	2.5	0	0	0	0
Fear	0	2.0	68.0	22.0	1.0	0	7.0
Joy	0	0	0.7	94.7	1.1	0	3.5
Sadness	8.6	0	0	0	69.5	2.3	19.6
Surprise	0	0	1.3	0	0	98.2	0.5
Neutral	1.6	0.4	0	1.6	6.0	0.4	90.0

Part V

Implementation

Contents

In the previous parts, algorithms of feature detection and feature extraction were studied. Method for feature classification was also examined. In this part, these algorithms and method are implemented and it is explained how it is done.

Chapter 9

Feature Detection with Viola-Jones

The feature detection part for this Facial Expression Recognition system is based on the Viola-Jones face detection algorithm. This chapter describes how this Viola-Jones algorithm is used and implemented.

9.1 Viola-Jones

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

Classifiers are trained prior to be used. Then they are loaded; one for the face and one for each region of interest. Then based on these classifiers, face and regions of interest are detected in the frames. Figure 9.1 shows an example of face detection. Some regions of interest are also detected. These regions are the mouth, the left eye and the right eye.

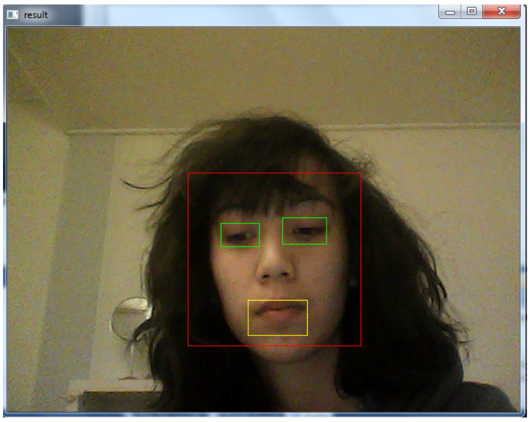

Figure 9.1: Example of face detection with Viola-Jones

Chapter 10

Feature Extraction with Local Binary Patterns

As said previously, the feature extraction method used for this Facial Expression Recognition system is the Local Binary Patterns (LBP) method. This chapter describes how this method is used and implemented.

10.1 Uniform Local Binary Patterns

A circular LBP operator, $LBP_{8,1}(p)$, is used with $P = 8$ and $R = 1.0$ to compute the pixel of the face image. The figure 10.1 shows what is obtained with this operator. The operator is also a uniform LBP operator.

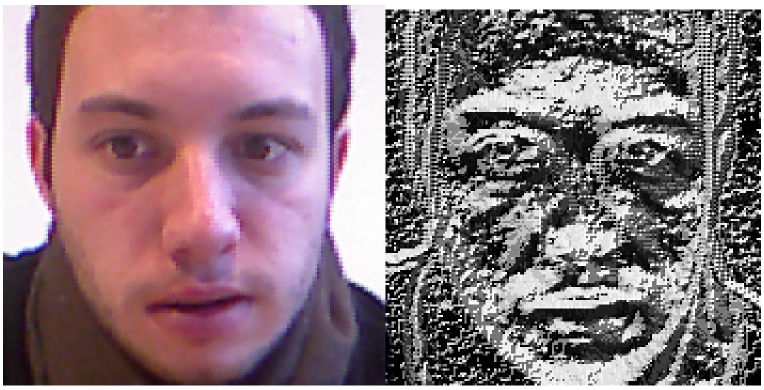

Figure 10.1: Circular LBP operator with $P = 8$ and $R = 1.0$

The face image is divided into regions. As it is commonly done, the face is a grid pattern of 6×7 (6 partitions for the columns and 7 partitions for the rows). In total the face image is divided into 42 regions. For each of these regions, the LBP operator, $LBP^{u^2}_{8,1}$, computes every pixel.

The output of this computation is the number of labelled 1s for the 8 neighbor pixels. Because a uniform LBP operator is used, the number is from 0 to 8. If the number is 9, it means that it is a non-uniform LBP. In total, the output number can have 10 different values (from 0 to 9).

10.2 Histogram computing

The numbers obtained after computing a region are concatenated into an histogram. The histogram has 10 bins; a bin for each value from 0 to 9. Each bin contains the number of pixels that have the value that correspond to that bin. For example, if in a specific region, 20 pixels, after being computed by the LBP operator are labelled with the value 7, then the bin attributed to the value 7 will have a value of 20 for this specific region. The figure 10.2 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.

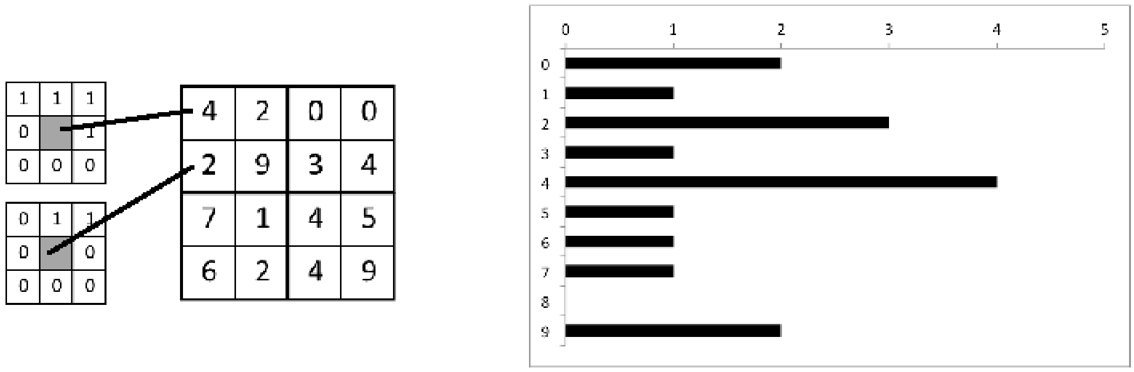

Figure 10.2: Example of Histogram computation

To obtain the feature vector for the whole image, an histogram is computed for each regions. In total, there are 42 histograms computed for the 42 regions. For the whole image, the 42 histograms have in total 420 bins; 10 bins for each histogram. The feature vector is obtained by putting the histogram side to side. The figure 10.3 shows how the feature vector is obtained with all the histograms.

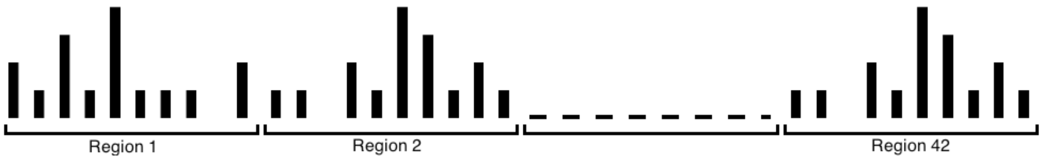

Figure 10.3: Example of a feature vector composed by the 42 histograms

Chapter 11

Feature Classification with Support Vector Machine

The feature classification part for this Facial Expression Recognition system is based on Support Vector Machine (SVM). 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, composed of the histogram 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 the SVM method.

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

$$l \qquad i : v \quad i : v \quad \ldots \quad -1 : -1$$

l stands for the label of the classes to which belongs the face image. It is represented by a number. The classes are the 6 basic emotions and the neutral one. The emotion are ordered alphabetically and the neutral is the first one. So the number corresponding to the emotions are the following:

- 0 corresponds to the *Neutral* emotion
- 1 corresponds to the *Afraid* emotion
- 2 corresponds to the *Angry* emotion
- 3 corresponds to the *Disgusted* emotion
- 4 corresponds to the *Happy* emotion
- 5 corresponds to the *Sad* emotion
- 6 corresponds to the *Surprised* emotion

i stands for the index of a bin in the feature vector. i is then 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. The values are normalized to be in the interval $[-1; 1]$. To normalize the values, the maximum v value of the whole feature vector has to be find as well as the minimum v value. The formula to have the new normalized value is the following:

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

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

$-1 : -1$ is what indicate the end of the feature vector.

The feature vector characterizes a face image; so there is only one feature vector by image.

11.2 Training and Model

The training part uses images from databases. The database used is the Karolinska Directed Emotional Faces Database (KDEF) database. All the face images are sorted by emotions. To train the data, 7 folders have to be browsed. Each folders contain frontal face images of the subjects of the KDEF database for only one emotion.

Each folder is browsed. For the first folder, the images in it are computed with LBP and a feature vector is obtained for each face image. Each feature vector is added to a file that will contain eventually all the features vectors of the face images of the training set. And then it goes to the second folder and it goes on until the last one. Based on this file, the model will be generated.

The model is trained by choosing a kernel. All four kernels available were tested to find the best one. As presented in Equation 8.2, the four kernels are the following:

- Linear:
$$K(x_i, x_j) = x_i^T x \tag{11.2}$$
- Polynomial:
$$K(x_i, x_j) = (\gamma x_i^T x_j + r)^T, \gamma > 0 \tag{11.3}$$
- Radial basis function (RBF)
$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0 \tag{11.4}$$
- Sigmoid:
$$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r) \tag{11.5}$$

For the three last kernels, a parameter can be changed and so be optimized. This parameter is the γ parameter. Each of these kernels was tested with the basic γ parameter and then was tested with the optimized γ parameter.

The cross validation method was also used. It is a method that by itself trains and tests the data at the same time to find the best parameters possible. The dataset is divided into n parts. And one part is used for testing while the other are used for training.

Chapter 12

Implementation on the Kinect

12.1 Generalities

The finality of the project is to make a software able to recognize emotions through pictures coming from a Kinect's sensor. The software is designed to run with Microsoft Windows 7 and need an available USB 2.0 port to plug the Kinect device. All of the project have been coded in C++, language enough famous to have various third party's library available for our subject purpose.
We used Microsoft Visual Studio 2010 to code and GitHub to allow our team to manage versions of the software.

12.2 Librairies

Several different libraries have been used to perform emotion recognition. One for the communication between the computer and the Kinect's 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 it's version 1.0 Beta 2.
- Opencv is a graphic library under a BSD licence, optimized for realtime processing. It have been released by intel and is actually maintained by Willow garage a robotic company.
- We are using LibSVM, which is an OpenSource library for Support Vector Machine. The version used is the 3.14 released on november 16th 2012.

12.3 Architecture

The program follow an Model-View-Controller architecture. This MVC pattern consist in 3 modules:

- The model part contain all the algorithms used.

- The view is the module which the user interact with.
- The controller have in charge to send command in order to manage each others modules.

Thanks to the language, we have been able to code in an oriented object way which make the use of MVC pattern easier. 5 classes have been created, 3 for the architecture, one for the lbp process and one to classificate datas.
Following the UML diagram of our classes:

12.4 Interactions

There is two possible interactions coming from the user:

- The user have to show a facial expression in front of the camera.
- The user have to decide when he wants the program to perform his face and emotion analysis.

The first one doesn't need an action coming from the user to the interface. Indeed, Kinect sensor can record 30 images per seconds so each expressions can be caught in the image stream. That is why the second interaction need the user to make an action (in our case, to press a button) to "block" the facial expression to make it being analyzed.

12.5 Algorithm

The entry point of the software let the 3 MVC modules to be initialized. Then it runs 3 functions through the controller:

- Initialization which loads models (for face detection and classification through SVM) and begin images capture.
- The main loop of the program which display images and wait for an actions from the user.
- Shutdown process, which delete models, release memory and close the communication with sensors.

Main loop algorithm

while the button 'q' isn't pressed do:

We block the program while any elements from the kinect is coming

We record frames from the stream handle after testing if datas are corrects

If there was not any problem we stock usefull bits from frames and we release the memory allocated for these raws data

We create an OpenCv Image type thanks to the bits

We detect the face on this new image and we define a region of interest

If button 'r' is pressed do:

We copy the image which have been recorded and displayed when the button have been pressed and we compute its histogram

We create a node for SVM thanks to the previous histogram

We perform the classification

We display the result, delete the region of interest and begin the loop again

Part VI

Evaluation

Contents

Content

Chapter 13

Results

Two kinds of results have been obtained. The first results are obtained by using a test set from the database used for the training set. The LBP operator extract features from each image of the test set, then the classification is done and this is how the first result set is obtained. The second result set is obtained with the Kinect. The Kinect gets video sequences of a subject in front of the Kinect and from these sequences, face images are extracted. Then the same process is applied to these face images and that is where comes from the second result set.

13.1 First result set

To train the model, 128 face images from the KDEF database have been used for each emotion plus the neutral state. In total, $128 \times 7 = 896$ face images have been used to train the model. To test the model, 12 face images from the KDEF database have been used for each emotion plus the neutral state. In total, $12 \times 7 = 84$ faces images has been used to test the model. For each of these 84 images, face detection was used first, then the LBP operator to extract the features and then the classification method was used.

For the classification part, the data has been trained with different kernels and different parameters. The following table resumes the results that have been obtained:

	Linear	Poly1	Poly2	RBF1	RBF2	Sigmoid1	Sigmoid2
neutral	33.33%	33.33%	25.00%	33.33%	33.33%	58,33%	41.67%
afraid	66.67%	91.67%	100.00%	75.00%	83.33%	75.00%	75.00%
angry	50.00%	50.00%	41.67%	50.00%	50.00%	66.67%	50.00%
disgusted	75.00%	83.33%	83.33%	75.00%	75.00%	75.00%	83.33%
happy	91.67%	91.67%	91.67%	91.67%	91.67%	91.67%	91.67%
sad	16.67%	16.67%	8.33%	8.33%	8.33%	8.33%	16.67%
surprised	33.33%	58.33%	50.00%	66.67%	58.33%	50.00%	75.00%
overall	52.38%	60.71%	57.14%	57.14%	57.14%	60.71%	61.90%

Poly1 stands for Polynomial and has this parameter: $D = 2$
Poly2 stands for Polynomial and has this parameter: $D = 3$
RBF1 has these parameters: $C = 8.0$ and $\gamma = 0.0078125$
RBF2 has these parameters: $C = 32.0$ and $\gamma = 0.001953125$
Sigmoid1 has these parameters: $C = 8.0$ and $\gamma = 0.0078125$
Sigmoid2 has these parameters: $C = 32.0$ and $\gamma = 0.001953125$

The same data has been trained with the same kernels and parameters but using the cross validation method (as explained in 11). All the results, compared to the one obtained without using cross validation are worse. Following is a table of both results with and without cross validation.

	with cross validation	without cross validation
Linear	53.57%	52.38%
Poly1	54.76%	60.71%
Poly2	44.05%	57.14%
RBF1	55.95%	57.14%
RBF2	50.00%	57.14%
Sigmoid1	55.95%	60.71%
Sigmoid2	61.90%	61.90%

The best accuracy percentage is for the Sigmoid2 column. It is obtained with a classification based on the Sigmoid kernel and with the following parameters for this kernel: $C = 32.0$ and $\gamma = 0.001953125$. These parameters are found by *gridsearch*, a script of the SVM library. They are the most optimized for the RBF kernel and for the Sigmoid kernel. The overall percentage of accuracy for all the emotions is 61.90%. Following is the confusion matrix:

	neutral	afraid	angry	disgusted	happy	sad	surprised	accuracy
neutral	5	4	2	0	1	0	0	41.67%
afraid	0	9	0	0	0	2	1	75.00%
angry	2	3	6	0	0	1	0	50.00%
disgusted	0	1	0	10	0	1	0	83.33%
happy	0	0	0	1	11	0	0	91.67%
sad	0	4	3	1	2	2	0	16.67%
surprised	2	1	0	0	0	0	9	75.00%

By looking at the confusion matrix, it is easy to notice that 3 facial expressions are harder to recognize than the others with this system. These 3 emotions are: angry, sad and neutral. There is a real gap between these 3 emotions and the 4 other ones (afraid, disgusted, happy and surprised). Indeed, these 3 emotions are recognized with an accuracy lower than 50%, while the 4 other ones are recognized with an accuracy higher than 70%.

< 50%	> 75%
neutral (5/12)	afraid (9/12)
angry (6/12)	disgusted (10/12)
sad (2/12)	happy (11/12)
	surprised (9/12)

The numbers that are in parenthesis represent the number of emotions well recognized over the total number of face images tested.

These 6 emotions plus the neutral state can be classified into 2 groups. Indeed, one group containing the 3 emotions hard to recognize and the second group containing the 4 emotions remaining. The 3 facial expressions *angry*, *sad* and *neutral*, are the ones that distort the less the face. Between 3 faces expressing each, one of these emotions, the differences are not clearly noticeable. The figure 13.1 shows face images from the KDEF database used in the test set, expressing these 3 facial expressions.

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

The second group contains the 4 following facial expressions *afraid*, *disgusted*, *happy* and *surprised*. These emotions distort significantly the face when they are expressed. This is why it is easier to recognize them. The figure 13.2 shows face images from the KDEF database used in the test set, expressing these 4 facial expressions. The important features carrying emotion as the mouth or the eyes are changing a lot while these 4 emotions are expressed.

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

As can be seen in the figure 13.2, for each emotion, the eyebrows are raised, or the eyes are wild opened and most of the the mouth is clearly opened. For the figure 13.1, nothing is clearly noticeable, the face is almost the same. All of this can explain the difficulties that the system encounters to differentiate the emotions *angry*, *sad* and *neutral*.

13.2 Second result set

The second result test is obtained based on the same process than the process that gives the first result set. The only difference is the input. The input is not face image from which the features are extracted; it is a video stream from the Kinect. A subject stands in face of the Kinect, the face is detected, then the features are extracted, and finally the classification is made almost in real time and the output is the name of the emotion expressed.

RESULT OBTAINED

Chapter 14

Issues

14.1 Feature extraction

14.2 Real-time

The computation of the LBP operator

Conclusion

In case you have questions, comments, suggestions or have found a bug, please do not hesitate to contact me. You can find my contact details below.

Jesper Kjær Nielsen
jkn@es.aau.dk
http://kom.aau.dk/~jkn
Niels Jernes Vej 12, A6-302
9220 Aalborg Ø

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] 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_FB_FacExprRecCVonline.pdf`, 2003.

[6] Charles Darwin. *The Expression of the Emotions in Man and Animals.* John Murray, 2. ed. edition, 1904.

[7] Fabrizio Dini. An application of viola-jones algorithm: face detection and tracking. `http://www.micc.unifi.it/dini/download/dbmm2008-Dini.pdf`, 2008.

[8] Gianluca Donato, Marian Stewart Bartlett, Joseh C. Hager, Paul Ekman, and Terrence J. Sejnowski. Classifying facial actions. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(10):974–989, 1999.

[9] Abhiram Ganesh. Evaluation of appearance based methods for facial expression recognition, 2008.

[10] Adam Harvey. Adam harvey explains viola-jones face detection. `http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html`, 2012.

[11] Robin Hewitt. How face detection works. `http://www.cognotics.com/opencv/servo_2007_series/part_2/sidebar.html`, 2007.

[12] 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.

[13] Bram K. Julsing. Face recognition with local binary patterns, 2007.

[14] Emotion Lab. Karolinska directed emotional faces (kdef). http://www.emotionlab.se/resources/kdef.

[15] Jenn-Jier James Lien. Automatic recognition of facial expressions using hidden markov models and estimation of expression intensity, 1998.

[16] 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.

[17] Michael Lyons. The japanese female facial expression (jaffe) database. http://www.kasrl.org/jaffe.html.

[18] Aleix Martinez and Robert Benavente. The ar face database. http://www-sipl.technion.ac.il/new/DataBases/Aleix%20Face%20Database.htm.

[19] University of British Columbia. The viola/jones face detector, 2001.

[20] 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.

[21] Maja Pantic and Leon J.M. Rothkrantzi. Automatic analysis of facial expressions: the state of the art. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(12):1424–1445, 2000.

[22] 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.

[23] Lawrence R. Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition, 1993.

[24] 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.

[25] 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.

[26] Padhraic Smyth. Face detection using the viola-jones method, 2007.

[27] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[28] UQUAM. The msfde. http://www.er.uqam.ca/nobel/r24700/Labo/Labo/MSEFE.html.

[29] 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.

[30] 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.

[31] Paul Viola and Michael Jones. Robust real-time object detection, 2001.

[32] Wikipedia. Statistical classification. `http://en.wikipedia.org/wiki/Classifier_(mathematics)`.

Appendix A

Appendix A name

Here is the first appendix