# Acknowledgments

Once upon the time in a faraway forest there was a rabbit writing something on a computer .A wolf came and asked him what is he doing.'I am working at my final project', replied the rabbit.'And what is the subject?', asked the wolf.'Rabbit the strongest animal in the forest', said the rabbit.'It can not be', argued the wolf, 'come in the bushes and I will show you'.After a couple of minutes, the wolf came out of the bushes, almost beaten to death.The rabbit followed him with the bear explaining 'It is not the subject that matters, but the mentor'.

Can you imagine how lucky would be a poor rabbit having two bears, as mentors?...or perhaps you can realize what that would mean for the wolf questioning the rabbit...

Well that's me... I got lucky having two great supervisors like Dorin Bocu and Leon Rothkrantz ...or Leon Rothkrantz and Dorin Bocu.

So, I pay my thanks to both of them.

# Contents

# List of Figures

# List of Tables

# Overview

This paper has been divided in three major parts:

**Facial expressions analysis** This is the first part of my diploma paper, and it aims to be a survey of the past work in the field of facial expressions analysis.It might be a starting point for those who want to become more familiar with face detection, facial expressions information extraction or facial expressions classification.

**Modeling FPDImSys** In the beginning, I was naive enough to believe that I could model and than implement a system who's goal was supposed to be :interpretation of the facial expressions...so, I put some effort in modeling such a system. Finally, the results of modeling and implementation parts, are only suited for : extraction of facial data, which is a subgoal of the initial one.
You can find here, a description of the steps I've taken so far,naming:choosen methodology,domain modeling and use case modeling.
For modelling the system I have been working with ObjectiF.

**Implementation details regarding the system** This part of the thesis gives details concerning the implementation of the application.To accomplish this task, I've been using **Java** language, and as an editor I've been working with **Borland JBuilder 4 Enterprise**. I will try to cover the most important programming details, the ones which ensure the functional side of the application.This application will have a user guide and it might be a good idea to read the user guide first[1].

[1] the user guide is attached to the application.You can get connected to the online help, if you press the help button residing on the toolbar

# Part I

# Facial expressions technology -survey

# Introduction

*People are seldom interested in how results are achieved .Most of the time they are just interested in the results themselves.This is because we assume that everybody thinks/works in a similar way, that is, the same way as we do.*

The human face plays an important role in our day by day life.It is the shelter of several organs like mouth, eyes, ears, nose which accomplish biological tasks, allowing the bearer to taste , see, hear and smell.We can figure out the importance of the human face(biological importance), if we just notice, that only one human sense:touch, is not perceived having the aid of the face. Besides these biological functions, the face provides a number of roles needed for our social activities and interpersonal communication.The human face mediates person identification, attitudinal/emotional state and lip-reading.

When communicating with other humans we use many forms of communication. The most obvious is speech, a form of verbal communication. However, this is often accompanied or substituted by other methods known as non-verbal forms of communication. In particular, there is the tone with which someone says something as well as the facial and bodily expressions that accompany it. Indeed, when it comes to guessing how someone has reacted to something or how they are feeling, expression analysis more accurate. If we consider the usual response to the question "Are you OK?" the statement "I'm fine", we can see how little words we use to communicate our emotional state.

There is a lot of research about facial expressions, most of all carried out by specialists in psychology.A prove of facial expressions's importance has been given by Mehrabian.[2].Whether a person feels liked or disliked during a 'face-to-face' communication, depends on spoken words only for 7%, the voice intonation contributes for 38% and the facial expression for an amazing 55%.

In the section to come will emphasize some issues concerning the importance of facial expressions stated by Paul Eckman[3] and Terrence J. Sejnowski.

[2]Professor Mehrabian is known for his pioneering work in the field of nonverbal communication (body language). His experiments helped identify nonverbal and subtle ways in which one conveys like-dislike, power and leadership, discomfort and insecurity, social attractiveness, or persuasiveness

[3]Paul Ekman is Professor of Psychology, in the Department of Psychiatry at the University of California Medical School, San Francisco

## 1.1 Background about the importance of facial expressions

Facial expressions provide information about:

- affective state,including both emotions such as fear, anger, enjoyment, surprise, sadness, disgust, and more enduring moods such as euphoria, dysphoria, or irritableness;
- cognitive activity, such as perplexity, concentration, or boredom;
- temperament and personality, including such traits as hostility, sociability or shyness;
- truthfulness, including the leakage of concealed emotions, and clues as to when the information provided in words about plans or actions is false;
- psychopathology, including not only diagnostic information relevant to depression, mania, schizophrenia, and other less severe disorders, but also information relevant to monitoring response to treatment.

...and some issues about the importance:

- In basic research on the brain, facial expressions can identify when specific mental processes are occurring, periods that can be examined with new imaging technologies (e.g., Magnetic Resonance Imaging or Positron Emission Tomography) too expensive for continuous monitoring. Facial expressions also hold promise for applied medical research, for example, in identifying the role of emotions and moods in coronary artery disease.
- In education, the teacher's facial expressions influence whether the pupils learn and the pupil's facial expressions can inform the teacher of the need to adjust the instructional message.
- In criminal justice contexts, facial expressions play a crucial role in establishing or detracting from credibility.
- In business, facial expressions are important in negotiations and personnel decisions.
- In medicine, facial expressions can be useful in studies of the autonomic nervous system and the psychological state of the patient.
- In international relations, analysis of facial expressions of world leaders can be used for assessing reliability and change in emotional state.
- In man-machine interactions, facial expressions could provide a way to communicate basic information about needs and demands to computers.

## 1.2 Facial expressions and computers

Automating facial expression analysis could bring facial expressions into man-machine interaction as a new modality to make this interaction more efficient. However, facial

expressions are not always unambiguously interpreted even by competent humans. Computers can do increasingly amazing things, but they are not magic.Faces are highly complex patterns that often differ in only subtle ways, and that it can be impossible for man or machine to compare or interpret images when there are differences in lighting or camera angle.

The information which we use all the time when communicating with others is currently not available to a computer. At present we interact with computers either via the keyboard or with the aid of a mouse. The interaction is one way and in the form of a command from the user to the computer. However, it would be more desirable if the computer could:

- initiate courses of action which are dependent on the mood of the user in this case the facial expression would be used as a condition for undertaking the action

- suggest and implement commands which the user would approve of but has not yet specified facial expression could be used here to monitor the users response to the action

The two ideas above could be a part of an ideal human computer interface(HCI).HCI multi-modal systems are widely thought to become the "forth generation" of computing and information technology.Such a system should be able to adapt itself to the context of use e.g who the user is? where he is? what is he doing ? how is he feeling? Four japanese researchers pointed out that human interpretation of interpersonal face-to-face communication provides an ideal model for designing a multi-modal human-computer interface.

The main characteristics of human communication are:multiplicity and multi-modality of communication channels. Examples of human communication channels [4] are:auditory channel that carries the speech, auditory channel that carries vocal intonation, visual channel that carries the facial expressions and visual channel that carries the body movements. The sense of sight, hearing and touch are examples of modalities. When different communication channels are used during face-to-face communication, human interaction becomes more flexible and robust.Failure of one channel is recovered by another channel and a message from one channel can be explained by another channel.As a result, communication becomes highly flexible and robust.

Until now, most of the studies, treat the ways of communication separately.Examples of such software systems which could provide a human like interaction with the user are:speech recognition systems, detection and interpretation of facial expression systems or recognition of body movements systems.

[4] communication medium

## 1.3 Software systems dealing with representations of human face

Face recognition and facial expression recognition are different tasks to accomplish, but both of them are part of the so called 'face technology' .Although, not dealing explicitly with expressions of human face, face recognition worths attention and one reason for that is the number of face recognition systems referred while surfing the web.

The first step in tackling face recognition and facial expressions analysis is face detection.

If the next step performed after detecting face and extracting information about facial features, is to identify the face (matching the face in the image with the known faces in a possible database) than these three steps will complete a recognition system. Typical sources of images for use in facial recognition include video camera signals and pre-existing photos such as those in driver's license databases.

Two well know face recognition systems are FaceIt and FACEFinder,developed by Visionics and Viisage.

Visionics Corp. of Jersey City,specializes in systems that use cameras linked to computers to scan faces and automatically compare them with photographs stored in a database. The company gained attention after its FaceIt [**?**] system was installed by Tampa police to match images caught by dozens of camera is against mug shots of well known criminals. "FaceIt recognizes faces in a distance ,in crowd and at a glance",the company officials said."These could be shoplifters,known terrorists or criminals, VIP guests or costumers, expected visitors generally classified as friends or foes". FaceIt will automatically detect human presence, locate and track faces, extract face images, perform identification by matching against a database of people.Fundamental to any face recognition system is the way in which faces are coded.FaceIt uses Local Feature Analysis (LFA) [5].Under optimal conditions, the company said, the error rate for matches is less than 1%.The company said its software is able to account for changes in lightening, facial hair and aging.The accuracy, however, depends on the clarity of both the photos in database and the image s being captured and researched, so that gloomy conditions could lower the accuracy.Mach rates also could fall if the face is recorded at an odd angle.

Viisage's (Visionics competitors)[**?**] FaceFINDER products meet the demands for a modern surveillance identification solution. Face recognition provides the non-intrusive and discreet capability for surveillance applications.Casinos are finding this product useful for scanning their properties for known cheats.International airports use FaceFINDER to search for terrorists and banks are using facial recognition to help identify ATM users ,eliminating the need for passwords and PIN's. Over the past three years, Viisage received about $800,000 as a part of a project managed by the Justice Department to develop a way to automatically search Internet pornography sites for images of missing and exploited children.

[5] LFA is a mathematical technique developed by co-founders of Visionics Corporation

In the paragraphs above, it is summarized the activity of two important companies currently working in face recognition's field.If you have interest in other companies work, you could follow the link [**?**]

Developing software systems for facial expressions analysis is a subject of research, most of it carried out by universities, or better say by departments of universities (like Knowledge Based Systems Department of TUDelft) or research institutes affiliated to universities (like The Robotics Institute at Carnegie Mellon University in USA).

In the chapters to come, will be described the necessary steps that have to be taken in order to accomplish facial expressions analysis.

# Face detection

Early efforts in the face detection have been dated back as early as the beginnings of the 1970's where simple anthropometric techniques were used.Despite of the problems, the growth of research remained stagnant until the 1990s, when practical face recognition systems become reality.

Face detection is a necessary first-step in facial expressions analysis (as well as in face recognition), with the purpose of localizing and extracting the face region from the background.

The human face has a high degree of variability in its appearance, which makes face detection a difficult problem in computer vision. The scale and the orientation of the face may vary from image to image.If the photographs are taken with a fixed camera, faces in the image may have different sizes and been taken at different angles due to the movements of the observed person.Therefore, is difficult to search for fixed patterns in those images.The presence of noise and occlusion makes this problem even more difficult. As is not a trivial problem, when it has to be accomplish by computers, face detection received consider-able attention among researchers.

And,we are talking about such a simple problem for the human brain...

Humans detect facial patterns by casual inspection of the scene.We detect faces effortlessly in a wide range of conditions,under bad lightening condition or from a great distance. The details of sharp features(e.g eyeballs,eyelashes) become less proeminent.The presence of features and their geometrical relationship with each other appears to be more important. There is currently very little evidence to prove the mechanism used by peoples in locating the face.It seems like there are special units to detect faces or this task is done very efficiently, probably using massively parallel computation. Another characteristic of human visual system is that a face is perceived as a hole unit, not as a collection of facial features.When a face is partially occluded (by a hand for example) we perceive the hole face as if we fill in the missing parts.This is very difficult to achieve by a computer.

If our visual system is very robust,we shall see what is important for computers to detect human faces.

One of the issues in detecting the minimum size for a pattern to be recognized as a face. Figure 2.1 shows a human face (of Sophia Loren ) in different spatial resolutions.A

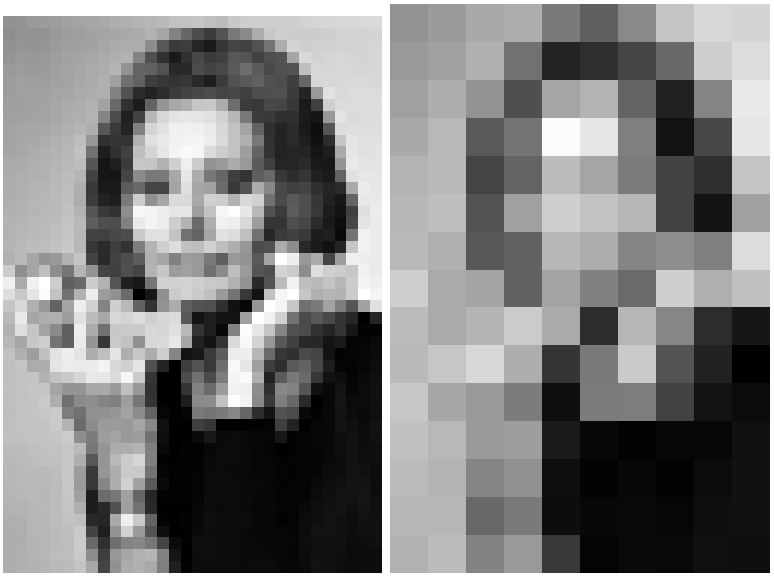


Figure 2.1: Human Face at Different Spatial Resolution

face can be detected easily in the first image(32x47).However,in the 10x15 image there is little resemblance to a face .It was suggested that a spatial resolution of 100-200 bits per face would be enough for detecting a face but not for identifying the person being represented.

A related issues is the minimum gray scale resolution necessary for face detection.Figure 2.2 ,shows the same face at two different gray scale resolutions:8bpp the first picture,1bpp the second picture [6].In all cases,the presence of a face is immediately noticed.It suggests that for detection (not identification) only 1bpp[7] images may be sufficient if the pattern has a good spatial resolution.

The first step in detecting a face in the images to locate it.

**Comment** If we are talking about static images,then the process of extracting the face is referred as localizing the face and its features in the scene.In the case of facial image sequences, this process is referred to as tracking the face and its features in the scene.

In most of the real life situations, however, the location of the face is not known a priori.The first step therefore, is to determine if the scene has any faces. In some instances, the condition under which the photograph is obtained is controlled[8].Hence,the location of the face in the scene can be easily determined.

[6] bits per pixel
[7] bpp=bit per pixel
[8] e.g the mugshots taken by the police

Figure 2.2: Human Face at Two Gray Resolution

Face detection techniques can be effectively organized into two categories distinguished by their different approach to utilizing face knowledge. One category is termed the *feature-based approach* and can also be referred as *analytical approach*. Typically, in these techniques face detection tasks are accomplished by manipulating distance, angles, and area measurements of the visual features derived from the scene. The other category is known as *image based approach* or *holistic approach*. Unlike feature-based approach, these kind of techniques are not using feature derivation and analysis.

## 2.1 The analytic way used for face detection

The development of the feature-based approach can be further divided into three areas. Given a typical face detection problem in locating a face in a cluttered[9] scene, low-level analysis first deals with the segmentation of visual features using pixel properties such as gray-scale and color. Because of the low-level nature, features generated from this analysis are ambiguous. In feature analysis, visual features are organized into a more global concept of face and facial features using information of face geometry. Through feature analysis, feature ambiguities are reduced and locations of the face and facial features are determined. The next group involves the use of active shape models. An example of active contour model could be the snake, proposed in the late 1980s, which has been developed for the purpose of complex and nonrigid feature extraction such as eye pupil and lips.

[9]cluttered=filled or scattered with a disorderly accumulation of objects

### 2.1.1 Low level analysis

#### Edges

As the most primitive feature in computer vision applications, edge representation was applied in the earliest face detection. The work was based on analyzing line drawings of the faces from photographs, aiming to locate facial features. Edges are places in the image with strong intensity contrast. Since edges often occur at image locations representing object boundaries, edge detection is extensively used in image segmentation when we want to divide the image into areas corresponding to different objects. Representing an image by its edges has the further advantage that the amount of data is reduced significantly while retaining most of the image information.The position of the edge can be estimated with the maximum of the 1st derivative or with the zero-crossing of the 2nd derivative. There are several operators for detecting edges like:Sobel , Roberts Cross, Canny Edge

Figure 2.3: The result of applying Sobel operator for edge detection

Detector,etc. The result of applying Sobel operator for the first image in figure 2.3 can be seen in the second image : Edge-based techniques have also been applied to detecting glasses in facial images.

#### Gray information

Besides edge details, the gray information within a face can also be used as features. Facial features such as eyebrows, pupils, and lips appear generally darker than their surrounding

facial regions. This property can be exploited to differentiate various facial parts. Several recent facial feature extraction algorithms search for local gray minima within segmented facial regions. In these algorithms, the input images are first enhanced by contrast stretching[10]to improve the quality of local dark patches and thereby make detection easier. The extraction of dark regions is achieved by low-level gray-scale thresholding[11].

Another possible technique is to explore the gray-scale behavior of faces in mosaic (pyramid) images. When the resolution of a face image is reduced gradually by averaging, macroscopic features of the face will disappear. At low resolution, face region will become uniform. Based on this observation, Yang[**?**] proposed a hierarchical face detection framework. Starting at low resolution images, face candidates are established by a set of rules that search for uniform regions. The face candidates are then verified by the existence of prominent facial features using local minima at higher resolutions. This method is now incorporated into a face detection system.

### Color

While gray information provides the basic representation for image features, color is a more powerful means of discerning object appearance. Two shapes of similar gray information might appear very differently in a color space. It was found that different human skin color gives rise to a tight cluster [12]in color spaces even when faces of difference races are considered. This means color composition of human skin differs little across individuals. One of the most widely used color models is RGB representation in which different colors are defined by combinations of red, green, and blue primary color components. Since the main variation in skin appearance is largely due to luminance change (brightness), normalized RGB colors are generally preferred, so that the effect of luminance can be filtered out. The normalized colors can be derived from the original RGB components as follows:

$$r = R/(R+G+B)$$
$$g = G/(R+G+B)$$
$$b = B/(R+G+B)$$

From the above equations can been that $r + g + b = 1$. The normalized colors can be effectively represented using only $r$ and $g$ values as $b$ can be obtained by noting $b = 1 - r - g$. In skin color analysis, a color histogram[13] based on $r$ and $g$ (red and green

[10]often called normalization,is a simple image enhancement technique that attempts to improve the contrast in an image by 'stretching' the range of intensity values it contains, to span a desired range of values.This is done by applying a linear scaling function to the image pixel values

[11]Thresholding often provides an easy and convenient way to perform a segmentation on the basis of the different intensities or colors in the foreground and background regions of an image.The input to a thresholding operation is typically a grayscale or color image. In the simplest implementation, the output is a binary image(black and white image) representing the segmentation

[12]cluster= group of the same or similar elements gathered or occurring closely together

[13]histogram =is a graph showing the number of pixels in an image at each different intensity value found in that image

are more present in skin color then blue) shows that face color occupies a small cluster in the histogram. By comparing color information of a pixel with respect to the $r$ and $g$ values of the face cluster, the likelihood of the pixel belonging to a flesh region of the face can be deduced.

ISFER[14] uses an algorithm for finding head contour based on HSV [15] color model.Analysis of 120 full-face images of different people results in the conclusion that the Hue of the face seldom exceeds the interval of [-40,60].These experimental results also yield the fact that the range of Hue never exceeds 40 for the images of a single face ,irrespective of change in lighting conditions.The algorithm analyzes the vertical and horizontal histogram of the input image and then the face is extracted as the biggest object in the scene having the hue in the defined range. Before applying the algorithm based on HVS, the outer boundaries of the face are determined, by analyzing the image histogram.ISFER is able to detects frontal or side views of the human face.

### 2.1.2 Feature analysis

Features generated from low-level analysis are likely to be ambiguous. For instance,in locating facial regions using a skin color model, background objects of similar color can also be detected. This is a classical problem which can be solved by higher level feature analysis. In many face detection techniques, the knowledge of face geometry has been employed to characterize and verify various features from their ambiguous state. One possible approach involves sequential feature searching strategies based on the relative positioning of individual facial features. The confidence of a feature existence is enhanced by the detection of nearby features.

#### Feature searching

Feature searching techniques begin with the determination of prominent facial features. The detection of the prominent features then allows for the existence of other less prominent features to be hypothesized using anthropometric[16] measurements of face geometry. For instance, a small area on top of a larger area in a head and shoulder sequence implies a “face on top of shoulder” scenario, and a pair of dark regions found in the face area increase the confidence of a face existence.A pair of eyes is the most commonly applied reference feature due to its distinct side-by-side appearance.

An algorithm was proposed to illustrate this technique. The algorithm starts by hypothesizing the top of a head and then a searching algorithm scans downward to find an eye-plane which appears to have a sudden increase in edge densities (measured by the

[14]Integrated System for Facial Expression Recognition is an expert system developed by Department of Knowledge Based Systems from Delft University of Technology

[15]HSV Hue Saturation Value

[16]anthropometry=the study of human body measurement for use in anthropological classification and comparison

ratio of black to white along the horizontal planes). The length between the top and the eye-plane is then used as a reference length. Using this reference length, a flexible facial template covering features such as the eyes and the mouth is initialized on the input image. The initial shape of the template is obtained by using anthropometric length with respect to the reference length , obtained from the modeling of 42 frontal faces in a database. The flexible template is then adjusted to the final feature positions according to a fine-tuning algorithm that employs an edge-based cost function. The algorithm is reported to have an 82% accuracy (out of 30 images in the same database) in detecting all facial features from quasi-frontal head and shoulder faces on a plain background. Although the algorithm manages to detect features of various races since it does not rely on gray and color information, it fails to detect features correctly if the face image contains eyeglasses and hair covering the forehead.

Besides feature searching there is another possible approach in the field of feature analysis and that is constellation analysis. Using more robust modeling methods like statistical analysis,face detection research address this problem by grouping facial features in face-like constellations.This is a way to solve problems like occluded or missing facial parts.

## 2.2 The holistic approach for detecting the face

Although some of the recent feature-based attempts have improved the ability to cope with the unpredictability, most are still limited to head and shoulder and quasi-frontal faces. There is still a need for techniques that can perform in more hostile scenarios, such as detecting multiple faces with complex backgrounds. This requirement has inspired a new research area in which face detection is treated as a pattern recognition problem. Unlike feature-based approach which makes use of a priori knowledge about the face,pattern-based techniques eliminate the potential error due to the incomplete or inaccurate face knowledge. The basic approach in recognizing face patterns is via a training procedure which classifies examples into face and non-face prototype classes. Comparison between these classes and a 2D intensity array[17] extracted from an input image allows the decision of face existence to be made. The simplest image-based approaches rely on template matching, but these approaches do not perform as well as the more complex like Principal Component Analysis(PCA) or neural networks.

### Eigenface technology

Eigenface is a technology patented at MIT which is often referred as Principal Component Analysis(PCA).The main idea of the principal component analysis is to find the vectors that account for the distribution of face image within the entire image space. In the eigenface technique we must have a training set of grayscale images and we have to compute

[17] matrix representing the gray-scale image

the eigenvectors of the covariance matrix(for the training set). These eigenvectors can be thought of ,as a set of features that together characterize the variation between face images. When the eigenvectors are displayed, they look like ghostly faces,and are termed eigenfaces.

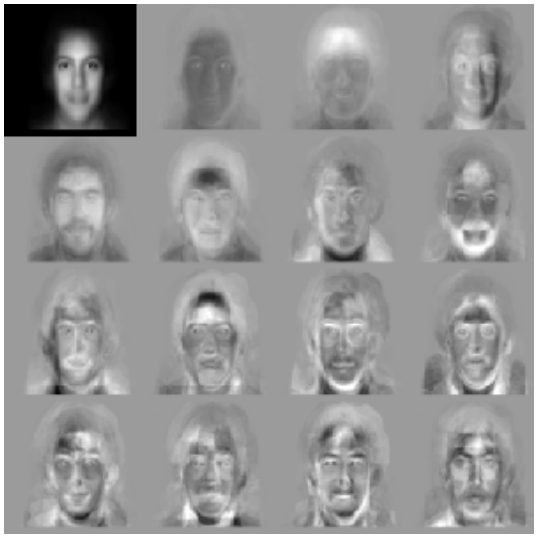


Figure 2.4: Standard Eigenfaces

The eigenfaces can be linearly combined to reconstruct any image in the training set exactly. Let's see how we can compute eigenfaces... If we consider face images of size $N$ by $N$, then these images can be thought of as a vector of dimension $N^2$. Let $T_1, T_2, ...., T_M$ be the training set of face images. The average face is defined by:

$$\psi = \frac{1}{M} \sum T_i, i = 1, M$$

Each face differs from the average face by the vector

$$\phi_i = T_i - \psi$$

. The covariance matrix has:

$$C = \frac{1}{M} \sum \phi_i \phi_i^T, i = 1, M$$

a dimension of $N^2$ by $N^2$. Solving the equation:

$$Cu_i = \lambda_i u_i$$

we can find out the eigenvalues $\lambda_i$ and eigenvectors $u_i$.These vectors define the subspace of face images,which we call "face space".Each vector is of length $N^2$,describes an $N$ by $N$ image,and is a linear combination of original face images. Eigenface technique is used especially for recognition systems provided with large databases of human faces, but it can

also be applied for face detection.To accomplish face detection, is computed the distance between the mean adjusted input image:

$$\phi = T - \psi$$

and its projection onto the face space:

$$\phi_f = \sum w_k u_k, k \epsilon M^{'}$$

$.M^{'}$ is a subspace of the eigenvectors space(those with the largest eigenvalues) and $w_k$ describes the contribution of each eigenvectors in representing the input image. We can deduce if the image is "face-like", if the DFFS(Distance From Face Space) is compared with a certain threshold. Eigenface technique has been tested by Ifran Essa and Alex Pentland, on a database of 7,562 images of 3,000 people of both sexes, ranged in age and ethnicity, having varying head position and facial hair.It worths saying, that the the tests were taken for images with complex backgrounds, having the human face in frontal or cvasifrontal views.Eigenface works not only in dealing with static images but also with image sequences.

### Neural networks

Neural networks are parallel distributed information processing structures, inspired more or less by the types of computational structures found in the brain, enabling computers to learn from experience. Such networks comprise processing elements, known as "units", which are analogous to neurons. These are classed as input units, hidden units, and output units. One unit connected to another implies that activity of one unit directly influences the activity of the other; the propensity of activity in one unit to induce or inhibit activity in the other is called the "weight" of the connection between these units. Networks learn by modifying these connection strengths or "weights" To obtain results when working with a neural network, one must accomplish to main tasks:training and testing.

Training a neural network for the face detection task is challenging because of the difficulty in characterizing prototypical "nonface" images. Unlike face recognition, in which the decision to be made is between different faces, the two classes to be distinguished in face detection are "images containing faces" and "images not containing faces". It is easy to get a representative sample of images which contain faces, but much harder to get a representative sample of those which do not.The size of the training set for the second class can grow quickly.

A solution for face detection based on neural networks has been revealed by Henry A. Rowley and Takeo Kanade[**?**].Their solution avoids the problem of using a huge training set for nonfaces by selectively adding images to the training set as training progresses.

# Facial expressions information extraction

After the presence of the face has been detected in the observed scene, the next step is to extract information about the encountered facial expression in an automatic way.If this step can not be performed in an automatic way, then a fully automatic system for analyzing facial expressions can not be developed.

Facial data extraction, in fact, the *methods* suited for this task, are dependent on the chosen *model* for representing the face, as well as on the facial image.
There are two main approaches for modeling, representing faces:

- analytic face representations
- holistic approach

Being given a particular face model, the extraction of face information without losing much of that information, is complicated by several factors.Two of them are:presesence of things [18] which can obscure the facial features and variation in size and orientation of the face in the image.

If we know that face models are divided in two main categories, then we should know that, the methods used for extracting data, are also grouped in two classes, naming:

- feature-based methods
- template-based methods

In the sections to come, will be referred some of the models employed in each of the above methods.The models to be presented are valid for static images.

## 3.1 Feature-based methods

Feature based methods localize the features of an analytical(geometrical) face model.

[18] glasses,facial hair

**Comment** *Model features*, utilized when modeling the face, are different from *facial features*, prominent features of the face like: eyes, eyebrows, etc.

The experiments carried out by psychology professors(J.N.Bassili,V.Bruce) suggested that the visual properties of the face, could be made clear by describing the movements of points belonging to the facial features (eyebrows,eyes and mouth) and then by analyzing the relationship between those movements.This approach determined the researchers to define visual properties of facial expressions based on points.

### 3.1.1 Kobayashi's point based model

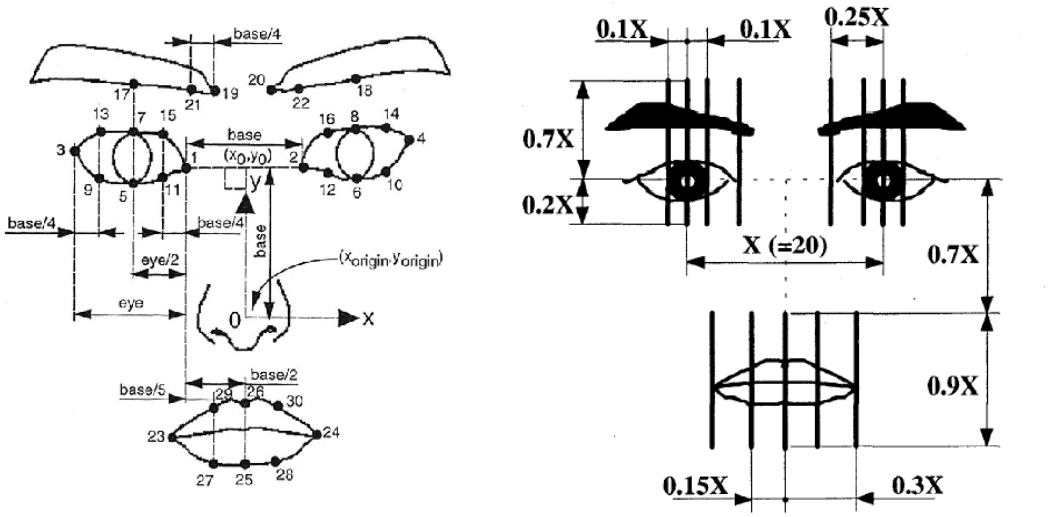



Figure 3.5: Kobayashi 's model

One of the well known face models based on points, was developed by Hiroshi Kobayashi and Furio Hara [19].In their early work, the two mentioned researchers proposed a geometric face model of 30FCP(Face Characteristic Points), as can be seen in the left image of the figure3.5. The FCPs are localized on the eyes,eyebrows and mouth, because more than 90% of the expressions conveyed by the human face are produced by the mentioned above facial organs.

It is obvious that this face model is suited just for the frontal views of the human face. In their later work, Hara and Kobayasi, developed a similar face model that can be seen in the right side of the figure 3.5. Having this model, they will perform analysis of the brightness distributions along 13 vertical lines, crossing some of the FCPs.

Why did they choose vertical lines?

[19] two japanese researchers interested in facial expressions analysis and its correlations with robots world

Because the borders lines of facial organs against face skin usually exists in horizontal direction and the change in facial expression appears in up or down movement of the border lines.

In order to compensate the difference in size of each subject's face,they normalized the face image as to the distance **X** (shown in the right image of the figure 3.5) between right and left pupils, to become 20 pixels, by using affine transform.The length of a vertical line is determined by empirical methods, to be 18 pixels.So, 234 pixels(18x13) stand for facial information, when using this facial model. The real-time system that was developed for recognizing the six basic emotions, using this model, deals with images taken from 1m,distance from the camera and human subjects with no facial hair or glasses.

### 3.1.2 ISFER face model

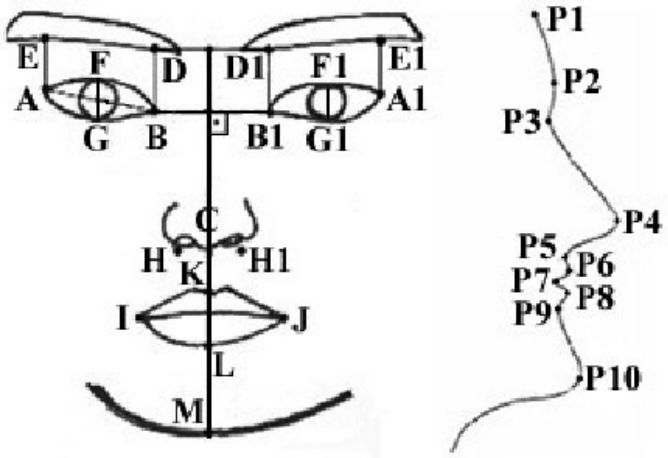



Figure 3.6: Facial points of the frontal-view face model and the side-view face model

Another analytic face model was developed for the use of ISFER.In the next paragraph you can find details about this model which is visible in figure 3.6 .

ISFER is using a point-based model composed of two 2D facial views, the frontal and the side view. The frontal-view face model is composed of 30 features. From these, 25 features are defined in correspondence with a set of 19 facial points(Fig.3.6) and the rest are some specific shapes of the mouth and chin.

ISFER's face frontal model describes facial activity in terms of AUs [20].The AUs linguistical description is given by FACS and more details about FACS and AUs can be found in the next chapter28. AUs are coded with respect to model features(examples can be seen below).

[20]Action Units

| AU | FACS description | Mapped on moedel |
|---|---|---|
| 1 | Raised inner brow | increased f1 and f2 |
| 2 | Raised outer brow | increased f1 or f2 |
| 6 | Raised cheek | increased f21 or f22 |
| 12 | Mouth corners pulled up | decreased f12, decreased f13, increased f14, increased f15 |

Table 3.1: Representation of AUs(not all of them)

where :

| AU | FACS description |
|---|---|
| f1 | Angle BAD |
| f2 | Angle B1A1D1 |
| f12 | Distance IB |
| f13 | Distance JB1 |
| f14 | Distance CI |
| f15 | Distance CJ |
| f21 | Image intensity in circle (r(0.5AB), C(A)) left from line (A, E) |
| f22 | Image intensity in circle (r(0.5A1B1), C(A1)) right from line(A1, E1) |

Table 3.2: Some of the features of the frontal view model

The utilized side-view face model consists of 10 profile points, which correspond with the peaks and valleys of the curvature of the profile contour function (Fig.3.6 ). To localize the contours of the prominent facial features and then extract the model features in an input dual-view, Pantic and Rothkrantz apply multiple feature detectors for each prominent facial feature (eyebrows, eyes, nose, mouth, and profile). For example, to localize the eyes they use methods like snakes and neural networks. Then, the best of the acquired (redundant) results is chosen. This is done based on both, the knowledge about the facial anatomy (used to check the correctness of the result of a certain detector) and the confidence in the performance of a specific detector (assigned to it based on its testing results). The performance of the detection scheme was tested on 496 dual views. Human observers in 89 percent approved, when visually inspected the achieved localization of the facial features. The system deals merely with images of faces without facial hair or glasses.

The source of the information given above is *Automatic Analysis of Facial Expressions:The State of the Art*[**?**].

## 3.2 Template-based methods

### 3.2.1 Active appearance model

One of the template based methods is Active Appearance Model(AAM) developed by Cootes, Edwards, Taylor.This model is defined by another two models naming:statistical model of the shape and statistical model of gray-level appearance. To build their model

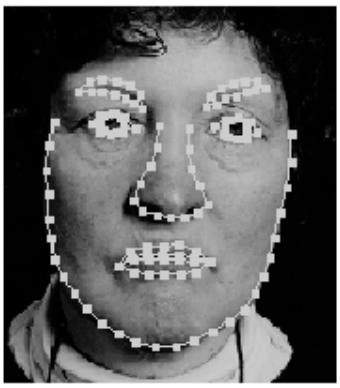


Figure 3.7: Example of face image labelled with 122 landmark points

they used facial images (from a 88 training images) that were manually labelled with 122 points localized around the facial features, as can be seen in figure3.2.1.To generate a statistical model of the shape variation, they aligned all training images in an approximate sense (this is done by translation, scaling and rotation) to ensure that they correspond as closely as possible and applied PCA(Principal Component Analysis) to get a mean shape.The gray-level information for each training image is extracted according to the mean shape.PCA is applied to obtain the mean normalized gray-level vector, and once more to obtain a 80D vector of appearance parameters, controlling both, the shape and the gray levels of the model.

That is how the AAM face model was constructed.But how to fit this model to an input image?
The answer for this question is another complex task described by the authors in *Active Appearance Models*[**?**]. It seems like, this method it' s suited for images representing faces without facial hair or glasses and the landmarks have to be manually labelled before the fitting procedure.

### 3.2.2 Labelled graphs and Bunch graphs

Another model which defines holistic views of the face is the **labelled graph**, or a more general face representation could be obtained with the **face bunch graph**.

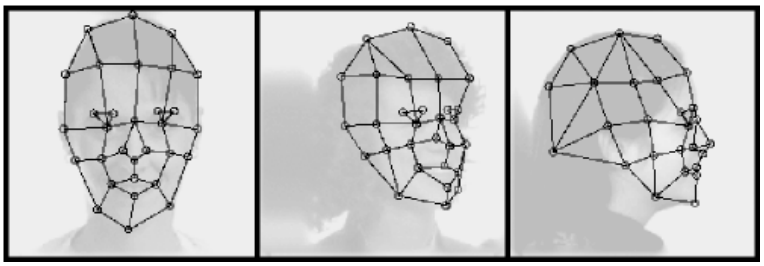


Figure 3.8: Graphs for faces in different views

A labelled graph is a set of nodes connected by edges;nodes are labelled with **jets**[21], and edges with distances.The geometry of an object[22] is encoded by edges, while the gray value distribution is encoded by the jets.

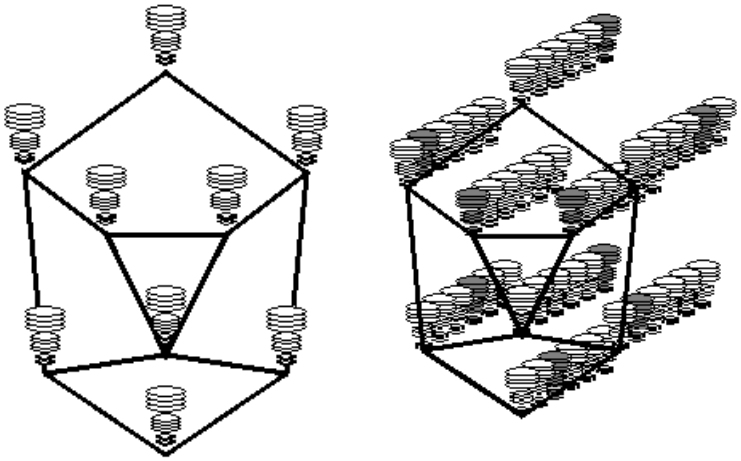


Figure 3.9: Face labelled graph(left)and face bunch graph(right)

**Note** Single labelled graphs(Fig22) can only represent 2-D views and not a 3-D structure of objects;a 3-D information can be processed by integrating several graphs.

While individual objects,such as faces, can be represented by simple labelled graphs, an object class requires a more comprehensive representation in order to account for all kind of variations within the class. Faces, for example, may have a beard or glasses, may have different expressions, or may be of different age, sex, or race. The Face Bunch Graph has a stack-like structure and combines graphs of individual sample faces. It is

[21]a jet is a condense and robust representation of a local gray value distribution, based on Gabor filter applied to the input image

[22]labelled graphs can be used not only for face representations, but for a more general object representation

crucial that the individual graphs all have the same structure and that the nodes refer to the same fiducial points. All jets referring to the same fiducial point [23], e.g. all left-eye jets, are bundled together in a bunch, from which one can select any jet as an alternative description. The left-eye bunch might contain a male eye, a female eye, both closed or open, etc. Each fiducial point is represented by such a set of alternatives and from each bunch any jet can be selected independently of the jets selected from the other bunches. That provides full combinatorial power of this representation and makes it so general even if constituted from few graphs only.

The figure22 shows a sketch of a face bunch graph. Each of the nine nodes is labelled with a bunch of six jets. From each bunch one particular jet has been selected, indicated as grey. The actual selection depends on the situation, e.g. the face onto which the face bunch graph is matched. Though constructed from six sample faces only, this bunch graph can potentially represent 6**9=10077696 different faces.

**Elastic graph matching** is the method to compare graphs with images and to generate new graphs. In its simplest version a single labelled graph is matched onto an image. A labelled graph has a set of jets arranged in a particular spatial order. A corresponding set of jets can be selected from the Gabor filtered image. The image jets initially have the same relative spatial arrangement as the graph jets, and each image jet corresponds to one graph jet. The similarity of the graph with the image then is simply the average jet similarity between image and graph jets.

In order to increase the similarity one allows the graph to translate, scale and distort to some extent, resulting in a different selection of image jets. The distortion and scaling is limited by a penalty term in the matching cost function. The image jet selection which leads to the highest similarity with the graph is used to generate a new graph.

When a bunch graph is used for matching, the procedure gets only a little bit more complicated. Beside selecting different image locations the graph similarity is also maximized by selecting the best fitting jet in each bunch. This is done independently of the other bunches to take full advantage of the combinatorics of the bunch graph representation.

[23] point of reference

# Facial expression classification

While the human mechanisms for face detection are very robust, the same is not the case for interpretation of facial expressions. It is often very difficult to determine the exact nature of the expression on a person's face. According to Bassili [24], a trained observer can correctly classify faces showing six basic emotions with an average of 87 percent. This ratio varies depending on several factors: the familiarity with the face, the familiarity with the personality of the observed person and the general experience with different types of expressions.Another factor which influences humans, when interpreting the face is the situation dependent nature of facial expressions.Among the type of messages conveyed by facial expressions are:emotions, emblems[25], illustrators[26], regulators[27].These kind of messages make classifying of facial expressions a difficult task to accomplish even when made by humans.

To classify facial expressions is the last step to be taken when analyzing encountered facial expressions.The bases for performing an automatic classification have been established by specialists in psychology and neuropsychology.

A fundamental issue when classifying a certain facial expression is to define a set of categories we want to deal with. Facial expressions can be classified in various ways:

- in terms of facial actions that cause an expression
- in terms of some non-prototyping expressions such as "raised eyebrows"
- in terms of some prototypic expressions such as emotional expressions

The Facial Action Coding System(FACS)28is probably the most known study of facial activity.It is a system that has been developed to facilitate objective measurement of facial activity.FACS provides a linguistic description of all possible,visually detectable, facial changes in terms of so-called *Action Units*.

Most of the studies on automated expression analysis perform an emotional classification .The most known and the most commonly used study on emotional classification of

[24]psychology professor with research interests focused on cognitive processes in the perception of people
[25]culture specific communicator such as wink
[26]actions accompanying and highlighting speech, such as raised brows
[27]nonverbal conversational mediators, such as smile or nods

facial expressions is Paul Eckman's study.Eckman[28] defined six such categories, referred to as the *basic emotions*:happiness,sadness,anger,fear,surprise and disgust. He described each basic emotion in terms of a facial actions that uniquely characterize that emotion.In the past years,many questions arose around this study.Two possible reasons are:

- It is not at all certain that each facial expression able to be displayed on the face can be classified under the six basic emotion categories
- Ekman's description is of the six prototypic basic emotions is linguistic, thus ambiguous)

Nevertheless, most of the studies on vision-based facial expression analysis rely on Eckman‘s emotional categorization of facial expressions.

**More about FACS**

The Facial Action Coding System was developed by Eckman & Friesen in 1978.The task was accomplished by determining how the contraction of each facial muscle (singly and in combination with other muscles) changes the appearance of the face. Videotapes of more than 5000 different combinations of muscular actions were examined to determine the specific changes in appearance which occurred and how to best differentiate one from another. It was not possible to reliably distinguish which specific muscle had acted to produce the lowering of the eyebrow and the drawing of the eyebrows together, and therefore the three muscles involved in these changes in appearance were combined into one specific Action Unit (AU). Likewise, the muscles involved in opening the lips have also been combined. Measurement with FACS is done in terms of Action Units rather than muscular units for two reasons. First, for a few changes in appearance, more than one muscle has been combined into a single AU, as described above. Second, FACS separates into two AUs the activity of the frontalis muscle, because the inner and outer portion of this muscle can act independently, producing different changes in appearance. There are 44 AUs which account for changes in facial expression, and 12 AUs which describe changes in gaze direction and head orientation. Coders spend approximately 100 hours learning FACS. A FACS coder "dissects" an observed expression, decomposing it into the specific AUs which produced the movement. The coder repeatedly views records of behavior in slowed and stopped motion to determine which AU or combination of AUs best account for the observed changes.

Automating FACS would make it widely accessible as a research tool in the behavioral science,but there are few attempts to automate FACS.

Independent of the used classification categories, the methods to accomplish the named task are divided into three main classes:template based methods, rule-based methods and neural-network-based methods.

[28]Paul Ekman is Professor of Psychology, in the Department of Psychiatry at the University of California Medical School, San Francisco

## 4.1 Template-based methods

If a template-based classification method is applied, the encountered facial expression is compared to the templates defined for each expression category. The best match decides the category of the shown expression. In general, it is difficult to achieve a template-based quantified recognition of a nonprototypic facial expression. There are infinitely a lot of combinations of different facial actions and their intensities that should be modelled with a finite set of templates. The problem becomes even more difficult due to the fact that everybody has his/her own maximal intensity of displaying a certain facial action.

To achieve expression classification into one of the six basic plus "neutral" emotion categories, Hong [29] made the assumption that two persons who look alike have a similar way for showing the same expression. First, they fit a labelled graph to an input facial image. Then, the best matching person, whose personalized gallery is available, is found by applying the method of elastic graph matching. The personalized galleries of nine people have been utilized, where each gallery contained 28 images (four images per expression). The personalized gallery of the best matching person is used to make the judgement on the category of the observed expression. The method has been tested on images of 25 subjects. The achieved recognition rate was 89% in the case of the familiar subjects and 73% in the case of unknown persons. As indicated by Hong, the availability of the personalized galleries of more individuals would probably increase the system's performance. The time necessary for performing a full analysis of an incoming facial image is about eight seconds.

**Note** This method, like all the other methods surveyed here are applicable just for static images.

## 4.2 Rule-based methods

The rule-based classification methods, classify the examined facial expression into the basic emotion categories based on the previously encoded facial actions. The prototypic expressions, which characterize the emotion categories, are first described in terms of facial actions. Then, the shown expression, described in terms of facial actions, is compared to the prototypic expressions defined for each of the emotion categories and classified in the optimal fitting category.

The method used by ISFER achieves automatic facial action coding from an input facial dual-view in few steps. First, a multidetector processing of the system performs automatic detection of the facial features in the examined facial image(see subsection 3.1.2). From the localized contours of the facial features, the model features (Fig. 3.6) are extracted. Then, the difference is calculated between the currently detected model features and the same features detected in an expressionless face of the same person.

[29] H.Hong, H. Neven, and C. von der Malsburg, "Online Facial Expression Recognition Based on Personalized Galleries"

Each basic emotion has an encoded description acquired from the linguistic description given by Eckman.Having the description of face's expression in terms of AUs, the classification can be accomplished by comparing the AUs coded description of the shown expression to each AUs coded description of a particular emotion category.For example, happiness is described below, in terms of AUs.

| **Expression** | **AUs coded description** |
|---|---|
| Happiness | 6 + 12 + 16 + (25 or 26) |

Table 4.3: Rule to describe happiness for "pure" emotional classification

This is a rule for recognizing a "pure" emotional expression, but a facial expression is rarely classified in just one of the six basic categories, naming:happiness, sadness, anger, disgust, fear or surprise.Therefore ISFER offers the possibility of giving a weighted classification of a particular expression.The illustrating rules for classifying happiness in terms of weighted emotional expressions are given below. The rules utilized by ISFER to

| **Emotion** | **AUs** | **Classified as** |
|---|---|---|
| Happiness | 6 | 98% happiness |
| Happiness | 12 + 16 + (25/26) | 100% happiness |
| Happiness | 12 | 100% happiness |
| Happiness | 12 + (25/26) | 100% happiness |

Table 4.4: Rules describing happiness for weighted emotional classification

classify facial expressions are validated by certified FACS coders.

The overall performance of ISFER automatic emotional classification of facial expressions has been tested on a set of 265 dual-view images.It worth saying that ISFER encodes 29 AUs(from 44 possible AUs) and the recognition ratio has been reported to be 91%. The dual-views used for testing of the system have been recorded under constant illumination and the subjects were both sexes and ranged in age(22-33) and ethnicity(European, South American, Asian).None of the subjects had a moustache, a beard, or wear glasses.

More information about ISFER can be found in *Expert system for automatic analysis of facial expressions*[**?**] and *Facial Expression Analysis by Computational Intelligence Techniques*[**?**]

# Part II

# Modeling FPDImSys

# Thoughts on methodology

I am going to try finding some answers for the questions below...

1. What is a methodology?
2. What is the use of involving a methodology when developing a software product?
3. What methodology I'm going to use for developing my system?

## 5.1 What is a methodology?

I am pretty satisfied with the definition of the software methodology, I've found in The Free On-line Dictionary of Computing...

**Methodology** An organized, documented set of procedures and guidelines for one or more phases of the software life cycle, such as analysis or design. Many methodologies include a diagramming notation for documenting the results of the procedure; a step-by-step "cookbook" approach for carrying out the procedure; and an objective (ideally quantified) set of criteria for determining whether the results of the procedure are of acceptable quality.

## 5.2 What is the use of involving a methodology when developing a software product?

The benefits of using a software methodology are surely visible when a complex soft system is being developed by several teams.There is really need to standardize the efforts and promoting reuse and consistency between project teams. There is almost always a need to maintain a system or to improve it for future versions, if we don't want it to become quickly shelfware.Software development methodologies are helping when appears the need to accomplish a task like this.

## 5.3 What methodology I'm going to use for developing my system?

What would you think, if I would say that I don't want to use a well known methodology to develop a software system? I see an alternative...

- You would think I'm nuts
- You would think that I am a genius person, possessing a brilliant mind capable to bring out a new methodology

...if you knew me, you would consider the first way of the alternative to be more viable.

But I still don't want to use a well known methodology like Rumbaugh's OMT, or it could be better to say that I don't want to use OMT entirely.I like the object model from OMT, but I don't know how to enhance and validate objects, discovered early in the development process through the map of concepts.As I am a beginner, that's an important stage. I would like a methodology closed to the user's view of the system, and RUP seems to be suited, but...is too large and complex, although it unifies 'best practices' of the software development techniques. What I like about RUP is the use case driven approach:is driving objects models from use cases.

**Note** Ivar Jacobson describes what use case driven means, when defining OOSE/Objectory methodology: 'When we wish to change the system behavior, we remodel the appropriate actor and use case.The hole system architecture will be controlled by what the users wish to do with the system.As we have traceability through all models, we will be able to modify the system from new requirements.We ask the users what they want to change (which use case) and see directly where these changes should be made in the other models.'

Now, I know several things about the methodology that I am willing to use:

- it has to be suited for object-oriented software development
- it has to be use case driven like RUP ,OOSE/Objectory and should not be too complex...so I can finish my work in about 20 years for now on.

Let's take a look on...

## 5.4 ICONIX Unified Object Modeling approach

Doug Rosenberg the initiator of this methodology[**?**] said : 'The key difference between this book and the amigos forthcoming book about the Rational Unified Process(RUP)

involves "altitude". I see the brilliant methodologists who created the UML as operating at, say, 30,000 feet above the earth,while I'm more like 1,000 feet off the ground, helping people take the great OO ideas and use them every day to build great software system ' The ICONIX approach is integrating the best elements of the Booch,Rumbaugh and Jacobson approaches, like RUP does but at a higher level.

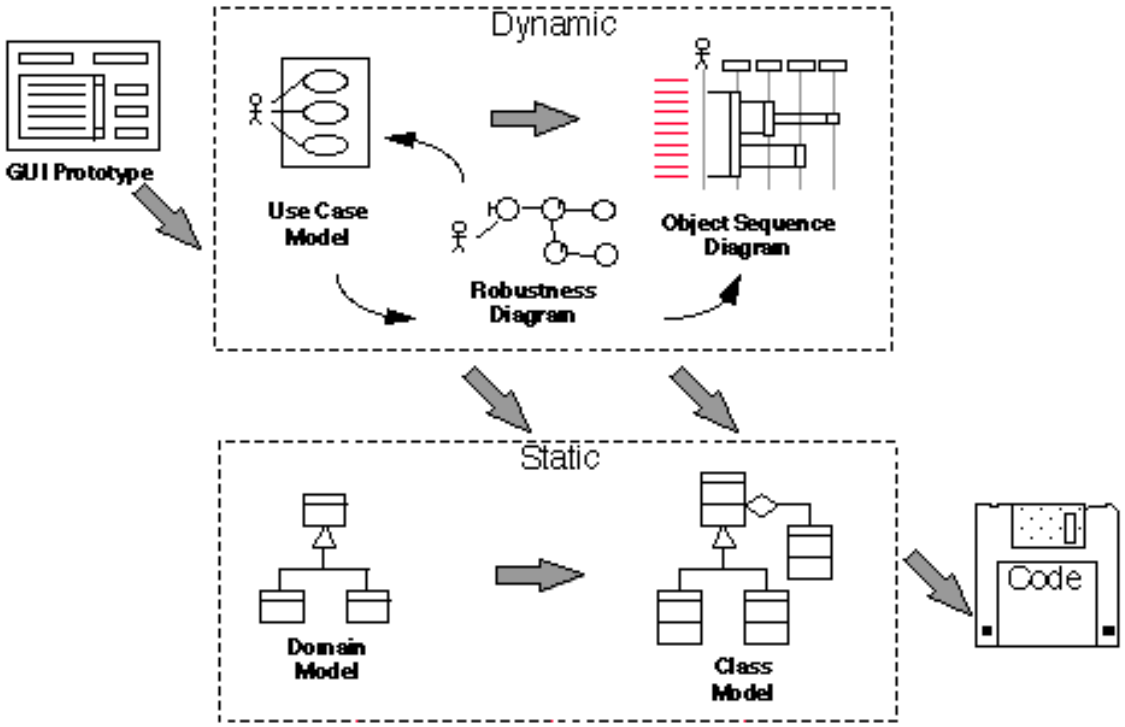



Figure 5.10: The ICONIX approach, showing three amigos contributions

The reason for choosing ICONIX for modeling is that I like the overall approach, but I found also some useful advices in the ObjectiF's documentation, concerning use case tracing, finding actors, etc. The conclusion is that I want to use ICONIX to guide my modeling efforts, not to become dogmatic about this approach ...I am not even sure that I could.

Table 5.5: Questions and answers

| | **OOSE/Objectory** | **OMT** | **Booch** | **UML Tool/Technique** |
|---|---|---|---|---|
| Users and user action? | * | | | Use cases |
| "Real world" objects? | | * | | High-level(coarsely grained) class diagrams |
| Objects for each use case? | * | | | Robustness analysis |
| Object interactions? | * | | * | Sequence and collaboration diagrams |
| Real time control issues? | * | * | * | State diagrams |
| How to build? | | | * | Low-level(finely grained) class diagrams |

# Domain modeling

The term **problem domain** refers to the area that encompasses real-world things and concepts related to the problem that the system is being designed to solve. **Domain modeling** is the task of discovering 'objects'(classes,actually) that represent those things and concepts.

What about ...

## 6.1 Requirements analysis

...first...

The goal of requirements analysis is to identify the user's needs and wishes for the system being developed.

The wanted system should be able to find an automated or a semiautomated way, to classify a human expression(that conveys a special feeling) in one of the six basic emotion categories defined by Eckman[30]: surprise, fear, disgust, anger, happiness and sadness, or to label the expression as being unknown.

As the output of the desired system is almost clear(at least to me), the input is just referred as 'a human expression'.But which is the source of the human expression? The response of this question points to the real input of the system which is*a profile image of a face* representing an expression.

We should not wait for miracles provided by such a system. There are psychological evidence to say that classification of lateral views of facial expressions seems even for human an unusually, difficult job.The overall recognition of lateral view is about 61% and the recognition of the frontal view is significantly higher.

The system is expected to give a reliable classification of our input image...but, with reference to what?

We should provide the system with another facial image ,a neutral ,expressionless view of the same face.The system will be using the neutral view, as a reference point in

[30]Paul Ekman is Professor of Psychology, in the Department of Psychiatry at the University of California Medical School, San Francisco

determining an appropriate category the facial expression image.

**Note** ObjectiF [31], the tool I am currently working with, has a template which generates a MSWord document in order to specify requirements.This template is based on IEEE Recommended Practice for Software Requirements Specifications. It has been normed as ANSI IEEE Std 830-1993. It may be a good idea to see, this document filled in, for the system I'm willing to model.

The goal of the system is to classify a expression view of the face profile. Although, few words are used to describe the main objective of the system, is not a trivial problem to obtain results(good results).

The main goal is stated, but there are several subsidiary goals of the system. Let's try to break down, to decompose the problem:

- First step: Detect face's profile
- Second step: Extract important points localized on profile line
- Third step: Give an interpretation(analyze) the contour of the profile expression
- Forth step: Classify emotionally the facial expression

The point is to show that, it might be no need for a classification to be accomplished, if the user's wishes go as far as the first or the second step go.

In the following paragraph, I will try to explain the steps mentioned above (except the last one), referring to their inputs and outputs.

**Note** Only the first step, face profile detection, has a 'special' input.The input for each of the following steps is represented by the output of the preceding step.

### Detect face's profile

The input for this first step is a image of the face profile and the output is an array of points representing the profile line.Two of the constraints concerning the input image are:

1. The image should be 8-bit gray scale image
2. The head is straight and turned to the left

[31] a powerful tool which supports object-oriented and component-based software development.For more information see http://www.microtool.de

### Extract important points localized on profile line

The output of this step is a reduced array of points belonging to the profile contour.These points are termed as extreme points ,or fiducial points of the face curvature.

### Give an interpretation(analysis) the contour of the profile expression

The output is a set of AU-s which produced the facial expression. Some additional information referring the steps above will be added when allocate functional requirements[32] to the use cases.

Do you still remember the title of this chapter?It was Domain Modeling... It seems like a 'grammatical inspection' of the requirements, is suited for this modeling stage [**?**][**?**].

## 6.1.1 'Grammatical inspection' technique

The purpose of the named technique is to discover concepts.A concept is a way to represent 'things' from the real world.A concept could be an object or an event.Labels are used to denote a concept e.g.'car' is a label for an object which has four wheels.

This technique is for sure a part of the conceptual perspective, one of the three perspective in the software development process stated by Martin Flower."A conceptual model [33] should be drawn with little or no regard for the software that might implement it"

**Note** I have not studied Martin Flower's book *UML Distiled*, but I found a brief description about the perspectives (conceptual, specification, implementation) in Alan Shalloway's book[**?**].

Highlighting nous and noun phrases in the requirements specifications,is a good way to identify concepts which could become objects or attributes.

Said and done...

- user
-profile
-*side* view
-*facial* image
-system
-*fiducial* points
-*profile* line

[32]functional requirements describes a law that governs a unit of behavior(described by a use case)
[33]represents the concepts in the domain under study

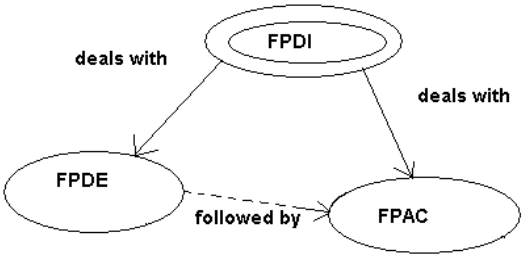



Figure 6.11: High level concepts map

-interpretation
-analysis
-AU
-*basic* emotions
-surprise
-fear
-disgust
-anger
-happiness
-sadness
-*neutral* profile
-*expression* profile.

**Note** We won't consider 'user' being a concept, because the user is an actor for the system being developed.
'System' is a too ambiguous to be considered a concept.
I consider 'interpretation' and 'analysis' being synonyms, so in the map of concepts will appear just one of them to eliminate the redundancy.
The words in italic are attributes.

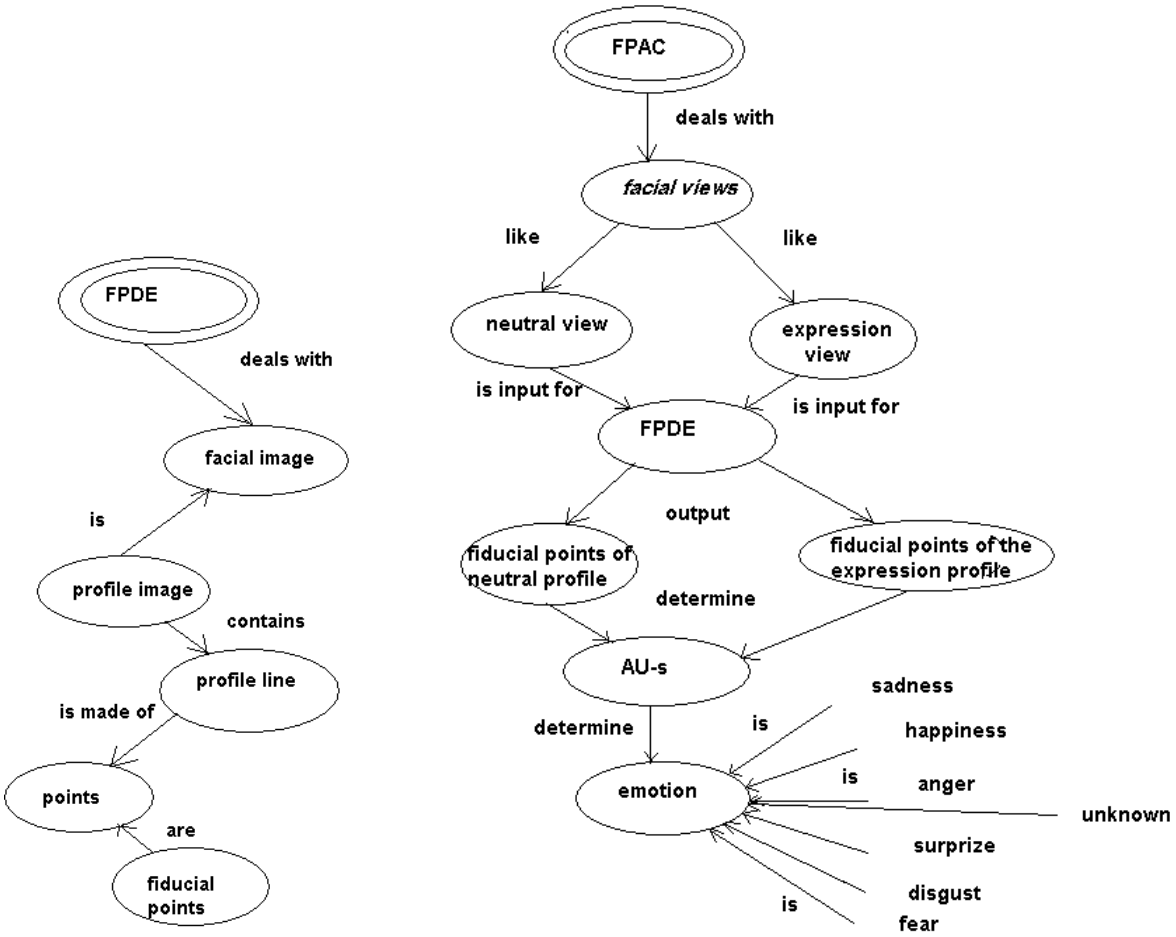
In figure 6.11 we can see the functionality that the system can provide[34].

I decided to group together two subordinate goals like **face profile detection** and **extracting fiducial points of the face curvature** because *only* this part of system's functionality is exposed to the user, if the input consists of *one* profile image.My reasoning was similar concerning the next two subsidiary objectives.

There are some rules in helping us to relate concepts map to classes and relations between classes:

- concepts are nothing more than 'baby' classes

[34]a list of acronyms used can be found in the requirement specification document




- links labelled with verbs like 'move','take','display', suggest future object operations
- links labelled with verbs like 'is a/an', 'could be', 'like', suggest inheritance relationships between objects
- links like 'is composed of', 'is a part of', determine aggregation relationships

More about concepts, concepts map, links between concepts and their importance for OOA, can be found the second part of the book 'Proiectare si Implementare Software'[**?**].

The class diagram presented in figure 6.12 is the first try to put together objects from the domain model.I guess it is obvious that this diagram is suitable just for face detection and extraction of the fiducial points.The detection of the face is the responsibility of an object called ProcessImage.This diagram will be refined during the following modeling stages.

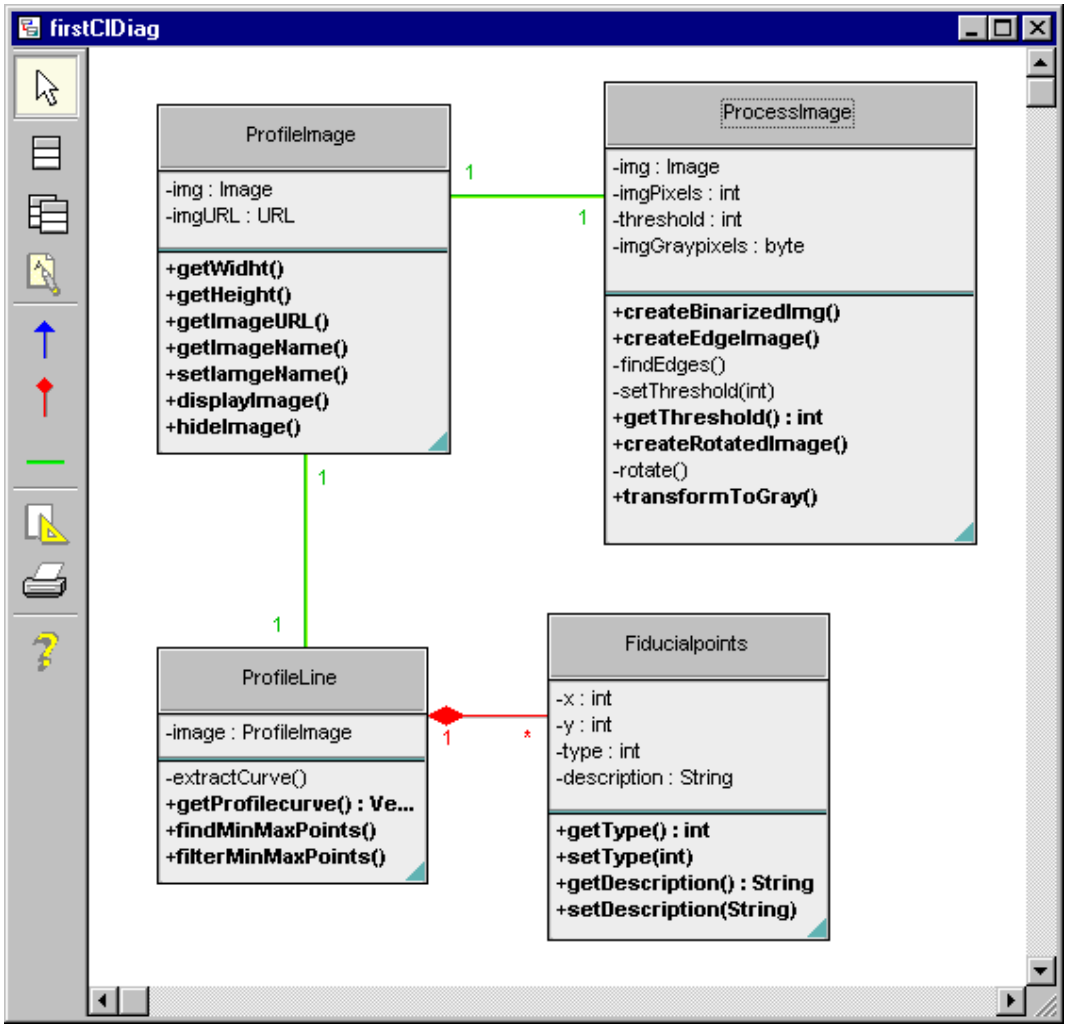



Figure 6.12: First class diagram

# Use Case Modeling

In this activity type, the costumer's of future user's expectations are established with regards to system's functionality.

**A Use Case** is a sequence of actions, a unit of behavior, that an actor performs within a system to achieve a particular goal.

**An actor** is a user of a system, including human users and other systems.

**Use case diagrams** are made of use cases and actors.This is a way to describe the functionality of a system and the users. Use case diagrams should be easy to understand because they represent user's view of the system.

Keep in mind that use cases satisfy only functional requirements [35], and functional requirements can be satisfied by one or more use cases.Functional requirements can be seen as laws which governs a behavior.

## 7.1 Looking for use cases

In one of the books I am currently reading[**?**], the advise is to work inward from a GUI(Graphical User Interface) to identify use cases.That means to write a rough user manual which will be used to draw a 'rapid' prototype of the graphical user interface without degenerating into extended GUI design. This approach can be accomplish using scenarios.

**Note** ObjectiF allows users to bring out dialogs and scenarios, after use cases and use case diagrams have been created.

Although, I have written a rough user manual, I have chosen to follow another way to find use cases.This approach is exposed in ObjectiF's documentation.

I have to follow some steps...

[35] there are many different types of requirements:functional requirements, data requirements, performance requirements, test requirements

- Find actors
- Describe actors
- Find use cases
- Describe use cases
- Create use case diagrams
- Make the use case's user interface

### 7.1.1 Find actors

We are advised to ask ourselves the following questions:

- Who will use the application system's main functions,for whom it is being developed?(This question is target at finding primary actors)
- Who will use the application to daily fulfill his tasks?(This question leads to active actors)

The answer for both questions above is **casual user**.

There are five more questions to answer, but neither I found a relevant answer for them, nor I consider them suited for this system.

### 7.1.2 Describe actors

I tried to accomplish this task using ObjectiF's template for describing actors.

### 7.1.3 Find use cases

Knowing that an actor always initiate a use case and the result of the use case is a particular value needed by the actor, we are provided with another set of questions to help us finding a use case.

- What are actors tasks?

-loading profile images into the system
-initiate face detection, face classification procedure, etc

- Which function are therefore expected from the application system, to help the user fulfill their tasks?

-provide a set of images(a database of images will be nice)
-a GUI easy to handle

- Which information must application system provide to the actor?

-array representing profile points
-array representing fiducial points
-list of AU-s involved in showing an expression
-classification results

**Note** I followed the naming conventions for use cases.Each use case should have a function oriented name consisting in a present-tens verb and nouns,following the pattern:*Actor* wants to *use case name*.

Founded use cases:Process image, Extract fiducial points, Classify image, Determine AU-s and Compute distances.

### 7.1.4 Describe use cases

I tried to describe use cases using ObjectiF's templates.And now...few words about the discovered use cases...

Initially, I had also a 'Detect face' use case.That would be a useful use case for the systems which are dealing with more complex methods for face detection.In my system 'Process image' will accomplish the mentioned task.

#### Why 'Process image'?

...'Preprocess Image' could be a better name

An image, almost always need preprocessing.'Process image'is not only desired but necessary, for an enhanced system, which could deal with images taken under various range of conditions.

#### Extract fiducial points

If I were to give a more general name to this particular stage of the process, I could have called it 'Extracting facial data(information)'.So, it's easy to notice that the current title contains the word 'points', which is a real clue in identifying the model being used to extract facial data.

The purpose of a face model is to define a mechanism for automatic extraction of facial data(in this case fiducial points), from facial images, and establish a relation between AU-s and the information extracted.

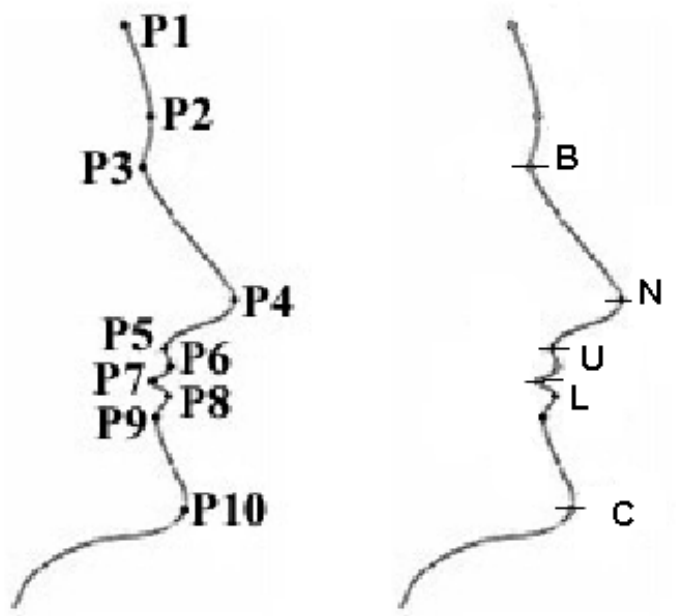



Figure 7.13: 2D Point Based Models

It was pointed out by several researchers(in psychology field of emotion recognition) that, a 2D point based model7.1.4 is accurate enough and provides sufficient information to classify a human expression.It seems like, such a 2D point based model resembles to to the model used by human observers when judging a facial expression.

**2D Profile Models**

Before defining a certain model, we have to answer a question:What kind of system will use this model? As facial recognition is a different task from facial expression classification, some of the information needed for face identification e.g length of the nose, is considered noise in facial expression categorization.

There are several models defined for face profile views of the face like Harmon's and Galaton's models7.13, but they are used for face identification systems. I am willing to use a side view model presented in the left side of the figure7.13, which was defined for the use of ISFER[36], an expert system which was developed by the Department of Knowledge Based Systems of Delft University of Technology.

A description of face profile characteristic points of this model can be found in Apendix:A I have to make two more comments regarding this particular use-case. The first one...
I consider that between the user and this use case should exist a 'communicates relation-

[36]Integrated System for Facial Expression Recognition performs recognition and emotional classification of human facial expression

ship' because the functionality provided, is the best that can be done if the user loads into system just one profile image. The second one...
The functionality of this use case gives the reason of considering the tool to be developed, automated or semiautomated. A semiautomated tool could offer the user a possibility to mark out fiducial points and then go on with analysis.

### Determine AU's

The aim of the 'Determine Au's' use case is to convert the obtained face geometry, into a set of AU's activated when showing the expression. After extracting fiducial points, the next step is to compute some distances (and maybe measure some angles) between characteristic points of the defined face model.

But how can we decide which distances are to be computed?

We can make such a decision if there is a rule to encode each AU, which can leave visible results on a face profile.A table containing this set of rules can be found in Apendix:A. Using the face model defined by KBS Department, from a total of 44Au-s defined by FACS,20 can be uniquely described. All the rules for representing AU-s in terms of a face model, have been generated from the linguistic description given by FACS.

**Note** Facial expressions represent a visible consequence of facial muscle activity.FACS represents a system that describes facial expressions in terms of codes of the facial muscles involvement.The activation of the muscles is described by the system as the activation of 44 Action Units(AUs).Each AU corresponds with the contraction produced by one or a group of related muscles.Activation of an AU is described in terms of facial appearance change i.e. change of facial features such as eyebrows, eyes and mouth caused by the activity of the underlying muscle(s).With FACS all visually distinguishable facial movements can be described as AUs-codes.On the other hand, FACS has been developed for human observers.So,neither facial muscle activity nor AU's codes can be extracted from an image.

### Classify image

...actually, the purpose is to classify the expression conveyed by the image.

If we want to classify a facial expression, we have to know the AUs involved in the shown expression, but that is not enough.We have to know, also, the description of each basic emotion in terms of AUs . The linguistic descriptions of the six basic emotional expressions given by Eckman, is the base for generating the AUs description as can be seen in one table in Appendix:A.

Two important questions regarding the facial model being developed and the rules being used for classifying is:How accurate is this face model? How reliable are the rules representing the AUs? To eliminate (as much as it can be) the subjectiveness when

interpreting a facial expression, the rules were given to FACS coders in order to validate their correctness. The average of the correct recognition ratio is 86% in the case of utilizing the rules, and 87% is the average ratio that can be obtained by a *trained* human observer.

## 7.2 Create use case diagrams

With ObjectiF's help I have made two use case diagrams, for the wanted system and a more general one, for a perspective view of a larger system.You can see two of the diagrams, below:

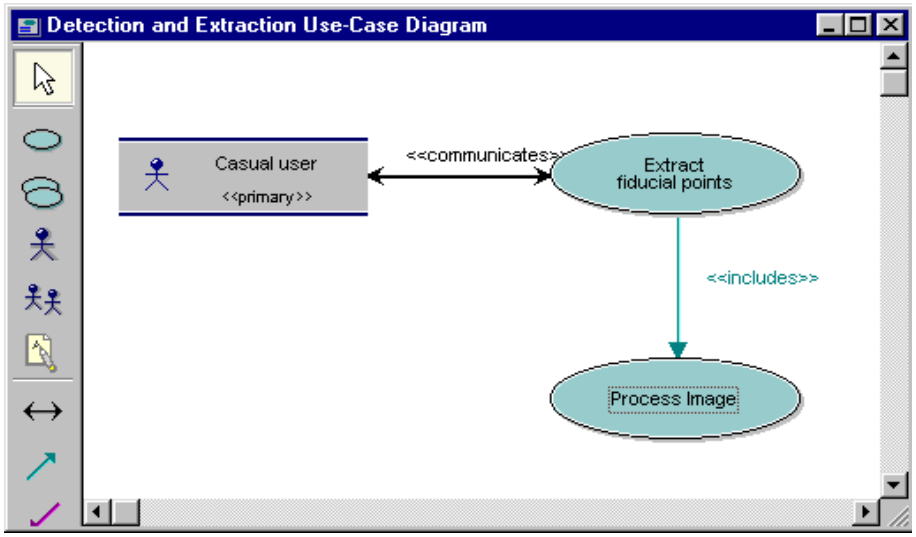



Figure 7.14: Detection and Extraction Use-Case Diagram

## 7.3 Make the use case's user interface

I generated some scenarios for the use cases, but I don't consider them to be a realistic representation of the future user's interface.

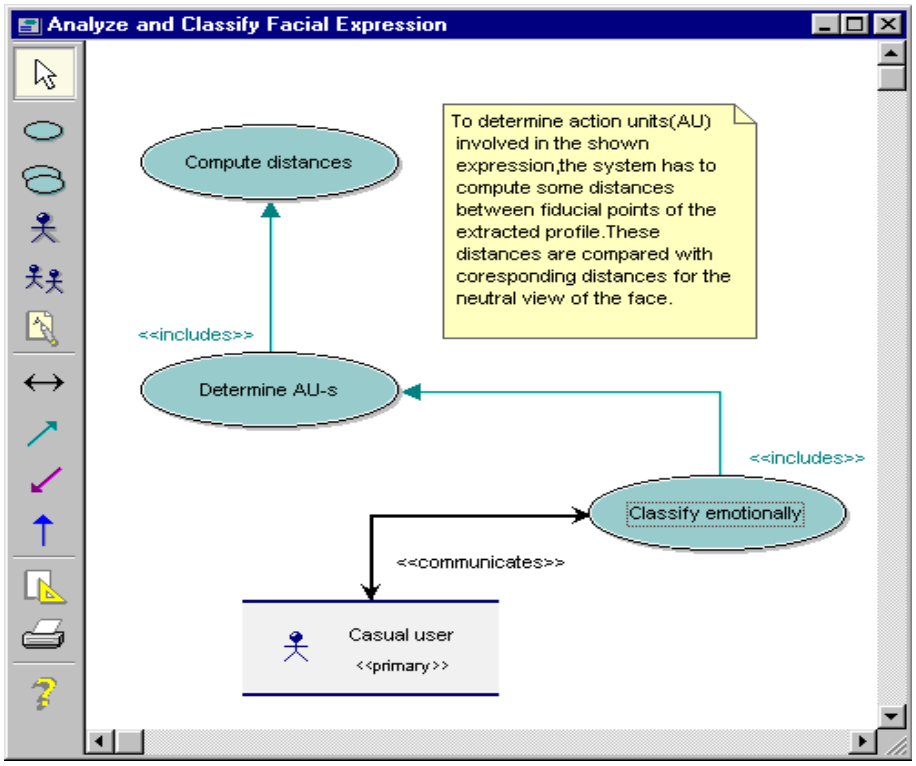



Figure 7.15: Analyze and Classify Facial Expression

# Robustness analysis

Regarding my modeling efforts, the next step to be taken is **robustness analysis**. Ivar Jacobson introduced the concept of **robustness analysis** to the world of OO in 1991.It involve analyzing the narrative text of each of your use cases and identifying a first-guess set of objects that will participate in the use case, then classifying these objects into the following three types of objects:

| | | |
|---|---|---|
| **Boundary objects** | are objects in the new system which with the actors will be interacting.These frequently include windows, screens, dialogs, and menus. |  |
| **Entity objects** | which are usually objects from the domain model.These kinds of objects should store data,fetch data, and perform different kinds of computations that don't change very often. |  |
| **Control objects** | embody much of the application logic.They serve as a connecting tissue between the users and the stored data. |  |

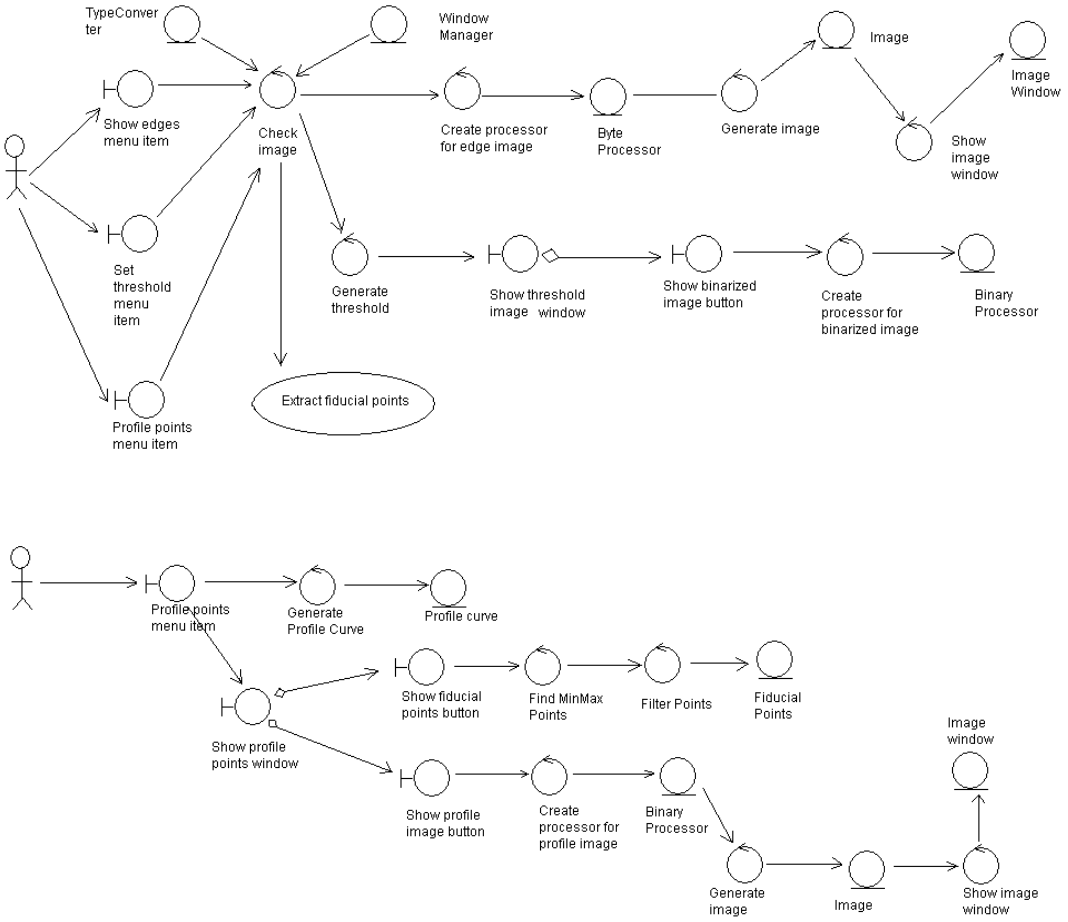
The robustness diagrams are associated to use cases.You can find below the robustness diagrams for 'Process image' and 'Extract fiducial points' use cases.

The robustness analysis is said to be important because it covers, or better say it fills a gap between *analysis*-the *what* and *design*-the *how*. Robustness analysis is not a part of RUP[37], but is an active part of ICONIX.Doug Rosenberg gives some arguments in favor of robustness analysis, and three of the reasons that seemed to work for me are:

**Sanity check** Helps you to make sure your use case text is correct and it doesn't specify behavior that is unreasonable for the set of objects you have to work with.

**Completeness check** Helps you to make sure those use cases address all the necessary

[37]Rational Unified Process




alternate courses of action.

**Ongoing discovery of objects** You may have missed some objets during domain modeling

If you still have in mind the initial diagram of classes 6.12, and you look at the robustness diagrams above then you will notice, there are some new entity objects which are not present in the initial domain model.

I decided that it is better to have a object called ***ImageWindow*** which will be responsible for displaying an object called ***Image***.If the modeled system is supposed to process and analyze more than one image in the same time, then we definitely need a ***WindowManager*** object.
Using just one kind of object to process images does not seems to be such a great idea, as long as I want to process different kinds of images.If I remember well, I have already mentioned that the input images for the system will be graylevel images and you may think that will be enough to have just one image processor, but that is not true, because

after binarizing the image, I won't need an image processor for graylevel images, I am more likely to need an object called ***BinaryProcessor***.Not to mention that will be nice to have a colored image(having a ***ColorProcessor*** object associated) as an input and to process the image at least as a graylevel image(having a ***ByteProcessor*** object associated).If we want to convert the colored image to a gray image, we will need for sure an object called ***TypeConverter***. The necessity of different kinds of image processors is once more confirmed, if we have the knowledge that an image can be encoded in various ways(for example we can have 24bit images and as well as 8bit images)

**Note** I discovered the necessity of some objects will coding. I suppose that is not so bad...I mean, not to finish modeling part before implementing, as I remember an advise which sounds like:'Analyze a little, design a little, implement a little'

# Interaction modeling

Without performing *interaction modelling* we may never know how the desired system might function.

**Interaction modeling** is the phase in which you build the threads that waves the objects together and enable us to start seeing how your new system will perform useful behavior.

You may think of interaction modeling as representing the changing of the guard between analysis and design.This is the point at which preliminary designs, coming from robustness analysis are driven to fully detailed object-oriented design.

Interaction modeling has three main goals:

- Allocate behavior among boundary, entity and control objects
- Show the detailed interaction that occur over time among the objects associated with each of your use cases
- Finalize the distribution operation among classes

I tried[38] to achieve this *dynamic* modeling by drawing sequence diagrams(the major work product of design).

The sequence diagram for **Extract fiducial points** use case [39], can be seen below:

...and, an activity diagram, which captures actions and results of those actions for the same use case:

[38]the interaction modeling I have performed is far from being complete
[39]sequence diagrams are attached to use cases

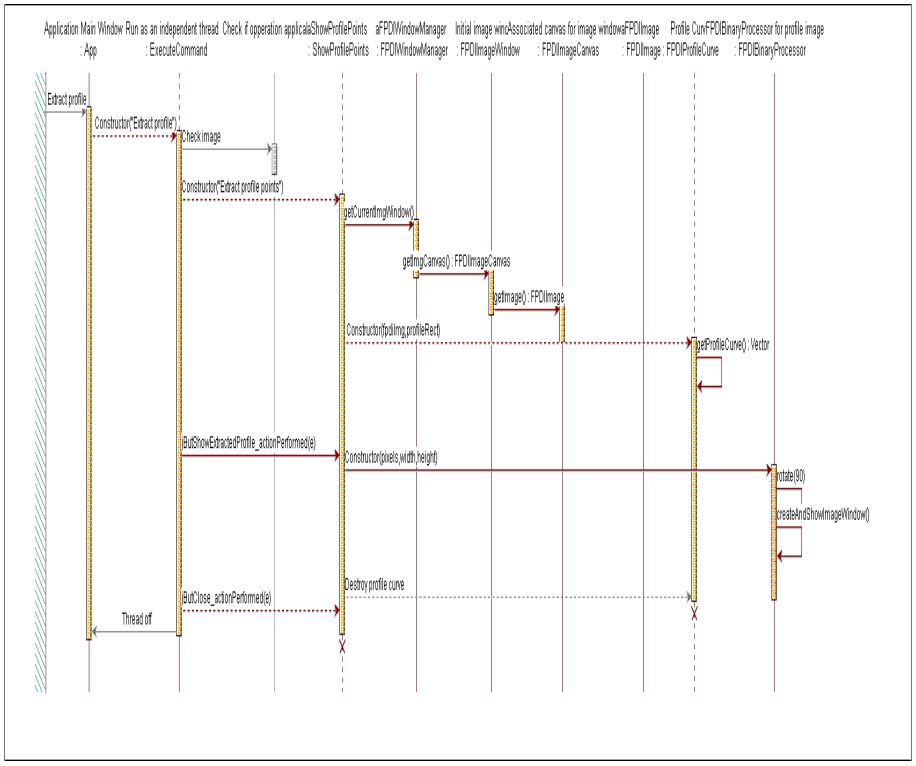



Figure 9.16: Extracting fiducial points sequence diagram

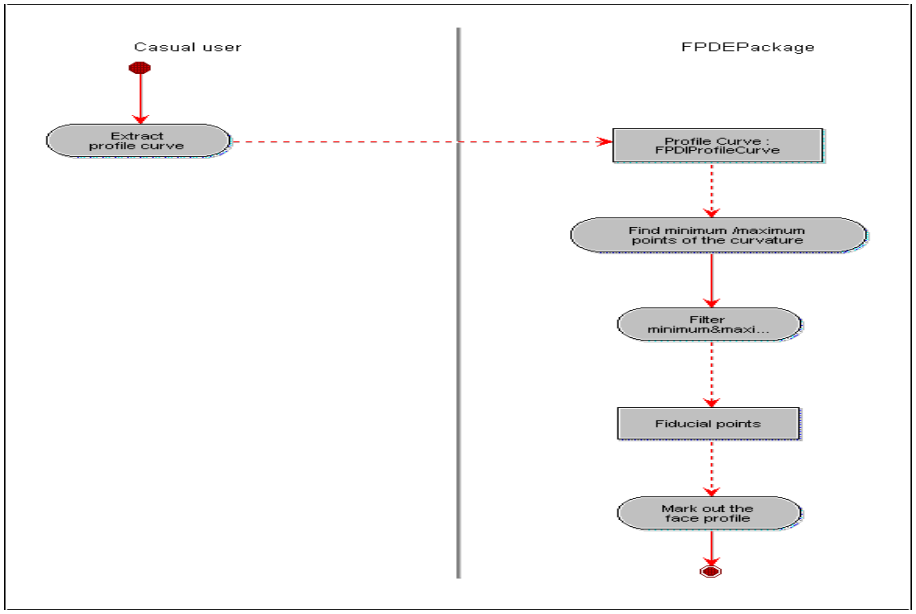



Figure 9.17: Extracting fiducial points activity diagram

# Part III

# Implementation details

# Class diagrams

Class diagrams are also a result of modeling the system.As I have only those class diagrams, suited for the implementation part, meaning detection of the face profile and extraction of fiducial points, I decided to integrate them here.

You will see below two important diagrams.There is one more that (worths at leat to mention it) shows how windows and canvases[40] work together.

Let's see now the diagrams I told you about:

As you can see in the figure10.18 we need different classes for processing the image:**FPDIByteProcessor**- to process gray scale images- and **FPDIColorProcessor**- for processing color images.Actually, only gray scale images will be processed by the system and RGB-images are converted to gray images using **TypeConverter class**.
**FPDIImageData**' s main role is to store an array of pixels;this class is also responsible of grabbing the pixels.

Something about extracting fiducial points diagram...

An **FPDIImage** object has associated an **FPDIProfileCurve** which is composed of **FPDIFiducialPoints**.**FPDIProfileCurve** is in duty with finding minimum and maximum point of the profile curve; this class is also responsible for filtering those extreme points to find the fiducial points of the profile curve.

[40]objects used for drawing operations

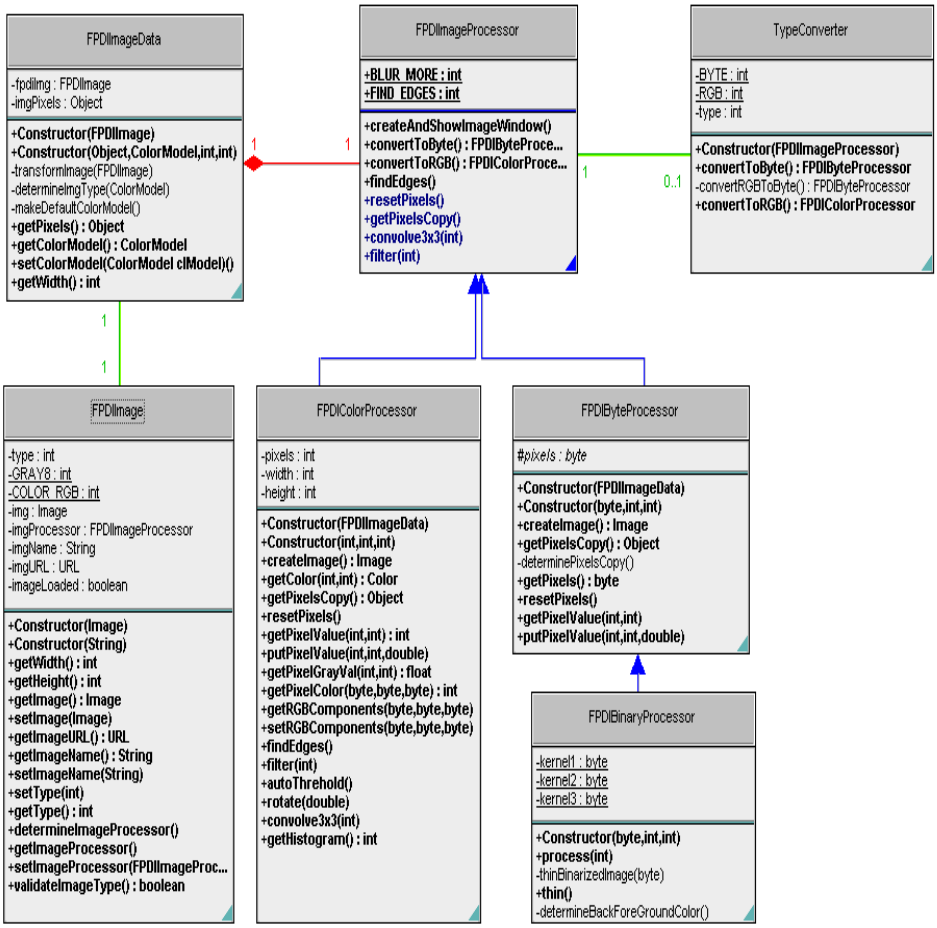



Figure 10.18: Processing image classes

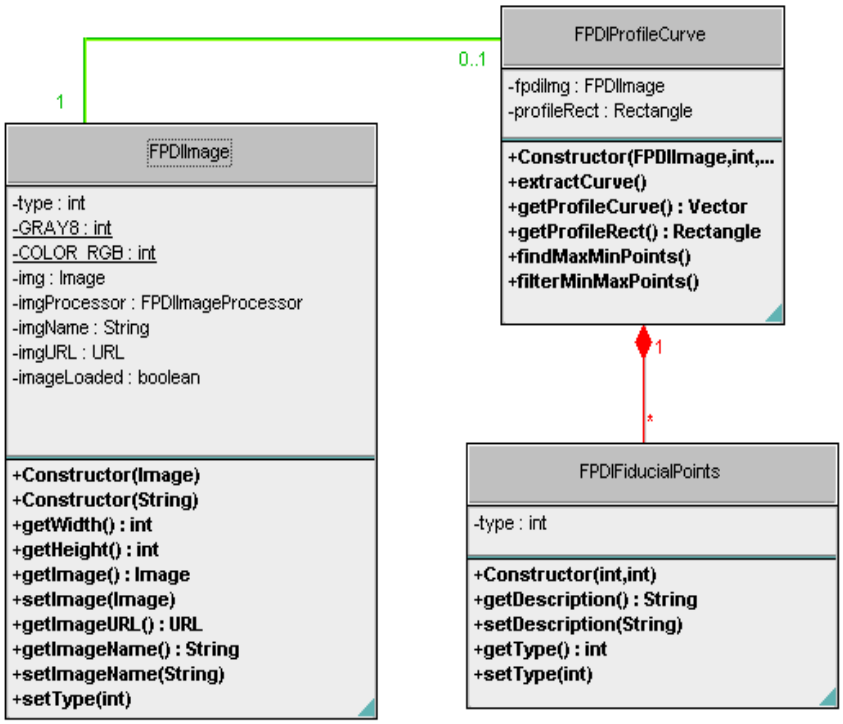



Figure 10.19: Extracting fiducial points classes

# Connecting to a database of profile photos

First of all I have to say that you can find the code which solves this problem, in the file named **OpenDB.java**[41]

Second of all, the database is not really a database, it's just a table.You can see the structure of this table(although it's not important)in the code listed below, when using the SELECT statement.

The table was created using **JDataStore** a tool which is coming with JBuider4's installation kit .You can use the named tool by choosing **Tools** from the main menu and then **JDataStore Explorer**. I say, it's useless to explain how a database can be created because I found the help pages concerning the subject to be satisfactory enough.

Just one more remark...in fact it's a paragraph taken from JBuilder's help:

> Remember that although the JDataStore Explorer can execute queries and save changes to the dataset back to server tables, it has no application -specific logic. The data is presented in just a simple tabular display. Data entry support and data validation that is usually provided by code in a data module won't be performed. If you edit data in the Explorer, you won't see any edit masks or pick lists, you won't be prevented from entering values that violate minimum or maximum constraints, and you might be able to leave required fields empty or violate referential integrity constraints. (Constraints defined in the server are enforced, but not until you attempt to save your changes.) The Explorer is useful for working with test data or making small, careful changes to production data, but it's not a substitute for a data module written with the specific integrity requirements of its datasets in mind.

Now it is the moment for some code...

[41] you can find this source file in the **UserInterface-¿src** subfolder of the application's main folder named **test2**

```
private void getConnected(String dbName)
{
/*
 * The ConnectionDescriptor object stores properties related to connecting
 * to a SQL database.
 * Its main properties are:
 *                          connectionURL (of the database)
 *                          userName
 *                          password
 *                          driver
 */
  ConnectionDescriptor conDescriptor=
                    new ConnectionDescriptor("jdbc:borland:dslocal:"+dbName);
  conDescriptor.setDriver("com.borland.datastore.jdbc.DataStoreDriver");
  conDescriptor.setPromptPassword(true);

  /* asociate the descriptor to the object representing the database FPDImSysDB */
  FPDImSysDB.setConnection(conDescriptor);

  /*
   * creating a query on the table tImages.You can see in the SELECT statement
   * the fields of the table
   */
  try{
   queryImages.setQuery(new QueryDescriptor(
                                                FPDImSysDB,
                                                "SELECT \"tImages\".\"ImgID\","+
                                                       "\"tImages\".\"Name\","+
                                                       "\"tImages\".\"NoseOrient\","+
                                                       "\"tImages\".\"Image\","+
                                                       "\"tImages\".\"faceRectX\","+
                                                       "\"tImages\".\"faceRectY\","+
                                                       "\"tImages\".\"faceRectWidth\","+
                                                       "\"tImages\".\"faceRectHeight\""+
                                                "FROM\"tImages\"",
                                                 null,
                                                 true,
                                                 Load.ALL)
                           );
    /* executing the query */
    queryImages.executeQuery();
    manageDataSet(queryImages);
  }catch(Exception ex){};
}
```

To display the table and the the images, I've been working with **DataExpress** and **dbSwing** components.An image is stored in the database like a stream.When the user decides to open a desired image the following code it's executed.

```
InputStream in =queryImages.getInputStream("Image");
   in.reset();

   ByteArrayOutputStream out = new ByteArrayOutputStream();

   int c;
   while ((c = in.read()) != -1)  out.write(c);

   Image img = Toolkit.getDefaultToolkit().createImage(out.toByteArray());
   out.close();
   in.close();
```

It worths mentioning that you can add an image to the database, only if you have write permissions.There is an account that has write rights: **admin** account.If you are logged on as a **user** than you can only open an image.

# Grabbing pixels from the input image

If we want to process an image we need to have further knowledge about that image, because it is impossible to work with an **Image** object without knowing its type, as well as every pixel's color and location.

**PixelGrabber** class from ***java.awt.image*** package helps us to transform an image into a two dimensional array of pixels:

```
private void transformImage(FPDIImage fpdiImg)
  {
     Image img=fpdiImg.getImage();
     int imgWidth=fpdiImg.getWidth();
     int imgHeight=fpdiImg.getHeight();

     imgPixels=new short[imgWidth*imgHeight];
     PixelGrabber pg=new PixelGrabber(img,0,0,imgWidth,imgHeight,false);
     try {
           pg.grabPixels();
           this.clModel=pg.getColorModel();
           this.determineImgType(clModel);
        }catch (InterruptedException e)
        {
           System.err.println("Interrupted waiting for pixels! "
                              +e.getMessage());
        }
    if ((pg.getStatus() & ImageObserver.ABORT) != 0)
    {
         System.err.println("image fetch aborted or errored");
         System.exit(-1);
    }
    else
         imgPixels=pg.getPixels();
   }
```

# Converting a RGB-color image to a gray scale image

The purpose of this chapter is to make one thing very clear:***the application can process only 8-bit gray scale images***, so if you want to work with colored images (RGB-color images), the application has to **convert** them.

When an image window [42]is displayed you will get as well an information message telling you that, the image has to be converted if you want to perform any processing steps like: binarizing the image or edge detection, etc.

If you disagree the message ***This image type can not be processed*** will appear every time you try to do something.

Immediately after a new image window appears on the desktop, the image is exposed to a validation procedure:

```
/* if the image is RGB-color, it will be converted and then displayed */
if (fpdiImg.validateImageType())
      FPDIImageProcesor imgProc=fpdiImg.getImageProcessor();
      imgProc.createAndShowImageWindow(fpdiImg.getImageName());
```

The image processor is turned to a byte processor:

```
/** Converts this processor to a ByteProcessor. */
  public  FPDIByteProcessor convertToByte()
  {
      TypeConverter tc = new TypeConverter(this);
      return tc.convertToByte();
  }
```

,and the following two methods of the **TypeConverter** object are executed:

[42]an instance of **FPDIImageWindow**

```
/** Converts processor to a ByteProcessor. */
   public FPDIByteProcessor convertToByte()
   {
       switch (type) {
           case BYTE:
               return (FPDIByteProcessor)ip;
           case RGB:
               return convertRGBToByte();
           default:
               return null;
       }
   }

   /** Converts a ColorProcessor to a ByteProcessor. */
  private FPDIByteProcessor convertRGBToByte()
  {
      int width=ip.fpdiImgData.getWidth();
      int height=ip.fpdiImgData.getHeight();
      byte[] pixels8=new byte[width*height];

      for (int x=0; x<width; x++)
         for (int y=0; y<height; y++)
           pixels8[y*width+x]=
             (byte)((int)((FPDIColorProcessor)ip).getPixelGrayVal(x,y)& 0xff);

      int w=ip.fpdiImgData.getWidth();
      int h=ip.fpdiImgData.getHeight();
      return
      new FPDIByteProcessor(pixels8,w,h);
  }
```

# Detecting edges using Sobel operator

Edges are pixels where the brightness function changes abruptly, and are often used to find image boundaries.Edge detection can be achieved using first order derivative filters, like Sobel edge detector.

The approximation of first derivative of the function [43] relies on central differences. If we use just the following two convolution kernels [44],

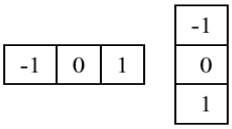



Figure 14.20: Prewitt kernels

we will get a noisy image, therefore the following two kernels are used

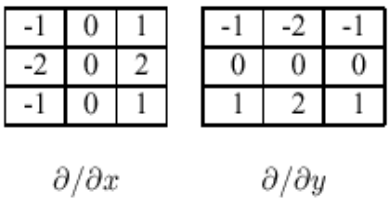



Figure 14.21: Sobel kernels

The code lines which perform edge detection are listed below.This procedure is a part of the ***FPDIByteProcessor.java*** file.

```
public void findEdges()
```

[43] an image is a function

[44] convolution is a simple mathematical operation which provides a way of 'multiplying together' two arrays of numbers;one array is a part of the image and the other is a small matrix called kernel

```
{
     filter(FIND_EDGES);
}

public void filter(int type)
{
     int p1, p2, p3, p4, p5, p6, p7, p8, p9;
     int offset, sum1, sum2, sum=0;

     int rowOffset =width;

        /* there is a need for a copy of  imgGrayPixels because
         * we need the 'old' values not the ones already computed
         */

     byte[] imgPixelsCopy=(byte[])this.getPixelsCopy();

     for (int i=1; i<height-1; i++)
     {
       offset = 1 +i*width;

       p1 = 0;
       p2 = imgPixelsCopy[offset-rowOffset-1] & 0xff;
       p3 = imgPixelsCopy[offset-rowOffset] & 0xff;
       p4 = 0;
       p5 = imgPixelsCopy[offset-1] & 0xff;
       p6 = imgPixelsCopy[offset] & 0xff;
       p7 = 0;
       p8 = imgPixelsCopy[offset+rowOffset-1] & 0xff;
       p9 = imgPixelsCopy[offset+rowOffset] & 0xff;

       for (int x=1; x<width-1; x++)
       {
           p1 = p2;
           p2 = p3;
           p3 = imgPixelsCopy[offset-rowOffset+1] & 0xff;
           p4 = p5;
           p5 = p6;
           p6 = imgPixelsCopy[offset+1] & 0xff;
           p7 = p8;
           p8 = p9;
           p9 = imgPixelsCopy[offset+rowOffset+1] & 0xff;

          switch (type)
```

```
        {
        case BLUR_MORE:
                sum=(p1+p2+p3+p4+p5+p6+p7+p8+p9)/9;
                break;
        case FIND_EDGES:
                sum1= p1 + 2*p2 + p3 - p7 - 2*p8 - p9;
                sum2= p1  + 2*p4 + p7 - p3 - 2*p6 - p9;
                sum= (int)Math.sqrt(sum1*sum1 + sum2*sum2);
                if (sum>255) sum=255;
         }

         pixels[offset++] =(byte)sum;
      } /*end -second for*/
    } /*end -first for*/
}
```

# How to binarize an image?

The result of binarizing an image,is a new image which has only two color pixels:**black** and **white**.

Gray level thresholding is the simplest segmentation process.Its main goal is to divide an image into parts that have a strong correlation with objects or areas of the real world contained in the image.

$$g(i,j) = \begin{cases} 1 & : \quad f(i,j) >= T \\ 0 & : \quad f(i,j) < T \end{cases}$$

Where $f$ is the initial image and $g$ represents the image after threshold value $T$ was applied.

So, if we want to obtain a binarized image we will have to determine the threshold automatically. This task was performed with using the next algorithm:

## 15.1 Iterative(optimal) threshold selection

1. Assuming no knowledge about the exact location of objects, consider as a first approximation that the four corners of the image contain background pixels only and the remainder contains object pixels.

2. At step $t$, compute $\mu_B^t$ and $\mu_O^t$ as the mean background and object grey level respectively,where segmentation into background and objects at step $t$ is defined by the threshold value $\mathsf{T}^t$ determined in the previous step.

$$\mu_B^t = \frac{\sum_{(i,j)\epsilon background} f(i,j)}{NO_{background-pixels}}$$

$$\mu_O^t = \frac{\sum_{(i,j)\epsilon objects} f(i,j)}{NO_{objects-pixels}}$$

3. Set

$$\mathsf{T}^{t+1} = \frac{\mu_B^t + \mu_O^t}{2}$$

$\mathsf{T}^t + 1$ now provides an updated background/object distinction

4. If $\mathsf{T}^{t+1} = \mathsf{T}^{t}$,halt;otherwise return to (2)

This algorithm is implemented in **FPDIByteProcessor.java**.The source code which solves this problem is...

```
public void autoThreshold()
{
  threshold(getAutoThreshold());
}

public int getAutoThreshold()
{
   int backSum=0,backNo=0;
   int objSum=0,objNo=0;
   double threshold=0;

   double backMean=0,objMean=0;
    for (int i=0;i<height;i++)
      for (int j=0;j<width;j++)
      {
         if ((i==0) && ((j==0)||(j==width-1)))
              backSum=backSum+this.getPixelValue(j,i);
         else
         if ((i==height-1)&& ((j==0)||(j==width-1)))
              backSum=backSum+this.getPixelValue(j,i);
         else
              objSum=objSum+this.getPixelValue(j,i);
      }

    backMean=backSum/4.0;
    objMean=objSum/(height*width-4.0);

    threshold=(objMean+backMean)/2.0;

    double new_threshold=threshold;

 do{
      backSum=0;
      backNo=0;

      objSum=0;
      objNo=0;
```

```
      threshold=new_threshold;

      for (int i=0;i<height;i++)
        for (int j=0;j<width;j++)
          if (this.getPixelValue(j,i)<=threshold)
                 {
                     backSum=backSum+this.getPixelValue(j,i);
                     backNo++;
                  }
             else
                  {
                     objSum=objSum+this.getPixelValue(j,i);
                     objNo++;
                  }
        objMean=objSum/(double)objNo;
        backMean=backSum/(double)backNo;

        new_threshold=(objMean+backMean)/2.0;

   }while(new_threshold!=threshold);

   System.out.println("threshold "+threshold);
   return (int)threshold;
}

public void threshold(int level)
{
  for (int y=0; y<height; y++)
     for (int x=0; x<width; x++)
        if (getPixelValue(x,y)<= level) this.putPixelValue(x,y,255);
           else  this.putPixelValue(x,y,0);

}
```

# Zooming,scrolling and rotating an image

These tools are necessary when working with images, because they help the user and the programmer, as well, to see, analyze, the image structure after processing the image.

## 16.1 Zooming the image

In this section only the ***zoomIn*** procedure will be presented, because the ***zoomOut*** procedure it is very similar.

```
protected void zoomIn(int x,int y)
 {
  /* you can enlarge the image no more than 16 times*/
     if (magnification>=16) return;
     double newMag = getHigherZoomLevel(magnification);

     if (newMag==1)
     {
           unzoom();
           return;
     }

     int newWidth = (int)(dstWidth*newMag/magnification);
     int newHeight = (int)(dstHeight*newMag/magnification);

     Dimension screen = Toolkit.getDefaultToolkit().getScreenSize();
     Point loc =canvContainer.getLocationOnScreen();

     if ((loc.x+50+newWidth)<screen.width && (loc.y+50+newHeight)<screen.height)
     {
         /* the image window can not be larger than the screen.If we
          * can increase the size of the window, we will do that
```

```
          */
         setDrawingSize(newWidth, newHeight);
         canvContainer.pack();
     }
     else
     {
        /* if we can't make the window larger we will show just the
         *  part of the image which has the mouse pointer in its centrum
         */
           int w = (int)Math.round(dstWidth/newMag);
           int h = (int)Math.round(dstHeight/newMag);
           /* mouse pointer coordinates with reference to srcRect*/
           x = srcRect.x+x/(int)this.magnification;
           y = srcRect.y+y/(int)this.magnification;

           Rectangle r = new Rectangle(x-w/2, y-h/2, w, h);
           /* if there is a need srcRect is adjusted*/
           if (r.x<0) r.x = 0;
           if (r.y<0) r.y = 0;
           if (r.x+w>iWidth)  r.x=iWidth-w;
           if (r.y+h>iHeight) r.y=iHeight-h;

           this.srcRect = r;

      }
     setMagnification(newMag);
     repaint();
}

protected static final double[]
   zoomLevels={1/16.0,1/12.0,1/8.0,1/6.0,1/4.0,1/3.0,1/2.0,
               1.0,2.0,3.0,4.0,6.0,8.0,12.0,16.0};

static double getHigherZoomLevel(double currentMag)
{
  double newMag = 16.0;
  for (int i = zoomLevels.length - 1; i >= 0; i--)
  {
     if (zoomLevels[i] > currentMag) newMag = zoomLevels[i];
     else   break;
   }
   return newMag;
}
```

If the magnifier tool was choosen from the toolbar and the left mouse button was clicked(on the image) then the image is magnified and the canvas repainted.

```
JToggleButton jButMag=
          (JToggleButton)App.jToolBar.getComponentAtIndex(App.MAGNIFY);
     if (jButMag.isSelected())
     {
          //left mouse button down
        if ((flags & (InputEvent.BUTTON1_MASK))==InputEvent.BUTTON1_MASK)
                  this.zoomIn(x,y);
          //right mouse button down
        if ((flags & (InputEvent.BUTTON3_MASK))==InputEvent.BUTTON3_MASK)
                   this.zoomOut(x,y);
     }
```

```
public void paint(Graphics g)
{
 Image img=fpdiImg.getImage();
 if (img!=null)
 /* draws a part of the image (the rectangle srcRect)
  * by scaling it to fit in the destination rectangle
  *(0,0,dstWidth,dstHeight)
  */
  g.drawImage(img, 0, 0, dstWidth,dstHeight,srcRect.x, srcRect.y,
              srcRect.x+srcRect.width, srcRect.y+srcRect.height, this);

}
```

## 16.2 Scroll an image

Having the code for scrolling and additional information regarding it, I guess it is not problem to understand it.

```
public void mouseDragged(MouseEvent e)
{
   int x=e.getX();
   int y=e.getY();
   JToggleButton jButScroll=
            (JToggleButton)App.jToolBar.getComponentAtIndex(App.SCROLL);
    /* if the scroll button is selected then scroll(x) procedure is caled
```

```
      * and the canvas is repainted
      */
    if (jButScroll.isSelected())
    {
       scroll(x,y);
       repaint();
    }
 }

  protected void scroll(int sx, int sy)
 {
  /* the current point while dragging-with reference to srcRect */

    int x = srcRect.x + (int)(sx/magnification);
    int y = srcRect.y + (int)(sy/magnification);

  /* (xMoseStart,yMouseStart) coordinates of the mouse point when
   * the mouse is for the first time pressed having the scroll
   * button enabled
   */

   /* srcRect's width and height are not modified;just the left corner
    * coordinates of srcRect are changed
    */
    int newx = srcRect.x+ (xMouseStart-x);
    int newy = srcRect.y + (yMouseStart-y);

    if (newx<0) newx = 0;
    if (newy<0) newy = 0;
    if ((newx+srcRect.width)>iWidth) newx =iWidth-srcRect.width;
    if ((newy+srcRect.height)>iHeight) newy =iHeight-srcRect.height;
    srcRect.x = newx;
    srcRect.y = newy;
  }
```

You will find the code for zooming and scrolling an image in **FPDICanvas** and **FPDIImageCanvas**. Zooming and scrolling an image are in fact, performed for the image ***canvas***.

## 16.3 Rotate an image

If you try to rotate an image you will see that it's impossible(the rotate buttons are disabled), unless the image is a binary image[45]. Unlike scroll or zoom, the image rotation affects every pixel of the input image.As the operation is valid only for binary images, you can find the code for rotation in **FPDIBinaryProcessor.java**.

```
public void rotate(double angle,Point center)
 {
        byte bgColor=(byte)getBackColor();

        byte[] pixels2 = (byte[])getPixelsCopy();
        double centerX = center.x;
        double centerY = center.y;

        double cos = Math.cos(angle);
        double sin = Math.sin(angle);
        double xs, ys;
        int index, ixs, iys;

        for (int y=0; y<height; y++)
        {
            index = y*width;
            for (int x=0; x<width; x++)
            {
             /* rotating the point (x,y) with reference to (centerX,centerY)*/
                xs=(x-centerX)*cos-(y-centerY)*sin+centerX;
                ys=(x-centerX)*sin+(y-centerY)*cos+centerY;
                if ((xs>=-0.01) I\&I\& (xs<width) I\&I\&
                    (ys>=-0.01) I\&I\& (ys<height))
                {
                   ixs = (int)(xs+0.5);
                   iys = (int)(ys+0.5);
                   if (ixs>=width) ixs = width - 1;
                   if (iys>=height) iys = height -1;
                   pixels[index++] = pixels2[width*iys+ixs];
                }
                else  pixels[index++]=bgColor ;
            }
        }

    }
```

[45]each pixel of the image is black or white

It worths mentioning that rotating the image is useful only for normalizing[46] the position of the head in the image.The tip of the nose is found as the first foreground pixel from the left[47].If the head is skewed, the nose might not be the left most foreground pixel[48] and it's necessary to normalize the face position,by **rotation**.

**Note** If you play to much with rotation tools, you will get a distorted image and that's because pixel's coordinates can **only** be of integer type.

[46] making the head straight

[47] a priori knowledge about the head position is used

[48] instead of detecting the tip of the nose, the tip of the chin or the hair could be labelled as being the nose

# Profile line and fiducial points

First of all I have to mention that, **the profile line** and **the profile curve** have the same meaning.

The profile line was found by scanning the image, from the left side to the right side[49] and each foreground pixel encountered while scanning the image from top down,is part of the profile curve.The face is first normalized and the line between

Having this profile curve extracted, we managed to reduce substantially, the amount of the data to be processed for finding the fiducial points [50].

The next step that has to be taken, is finding extreme points of the profile curve:

```
private void findMaxMinPoints()
{
 FPDIBinaryProcessor binProc=
        (FPDIBinaryProcessor)fpdiImg.getImageProcessor();
 int indOfChin=getIndexProfileVect(binProc.getChinPoint());
 for (int i=0;i<indOfChin;i++)
 {
    int li=i-neighbour;
    int ls=i+neighbour;
    int nrMin=0;
    int nrMax=0;
    Point pi=(Point)profileVector.elementAt(i);
    if (binProc.getNosePoint().equals(pi))
    {
        types.add("min");
        minmax.add(pi);
        continue;
    }
    if ((li>0)&&(ls<indOfChin))
       for (int j=li+1;j<ls;j++)
           if (j!=i)
```

[49] the nose is turned to left

[50] the image is a bidimensional vector, and the profile line is just a vector

```
            {
               Point currPoint=(Point)profileVector.elementAt(j);
               if (currPoint.x>=pi.x) nrMin++;
               if (currPoint.x<=pi.x) nrMax++;
            }
     if (nrMin==ls-li-2) {
                            types.add("min");
                            minmax.add(pi);
                          }
     else
     if (nrMax==ls-li-2) {
                            types.add("max");
                            minmax.add(pi);
                         }

     }
   if (indOfChin!=-1)
   {
    Point pi=(Point)profileVector.elementAt(indOfChin);
    types.add("min");
    minmax.add(pi);
   }
  }
```

As the profile line is a function, these extreme points are local minima and maxima points for a defined neighborhood.

After taking this step, we have too many potential fiducial points and we need to filter them using the rule:*starting with the nose and going up and then down, we ought to have each minimum point followed by a maximum point and each maximum point followed by a minimum point.*

You can find the code which performs this task in the file named **FPDIProfile-Curve.java**.

# Attaching a professional help to the application

The **JavaHelp** system consists of a fully featured, highly extensible specification and an implementation of that specification written entirely in the Java language. The implementation, based on the Java Foundation Classes (JFC), provides a very simple interface that allows application developers and authors to quickly and easily add online help to their applications.

There is no purpose to explain what should be done to add a professional help to your application, as long as the **JavaHelp API** and its documentation can be downloaded from: http://www.sun.com/products/javahelp

I can say only that a help system is based on ***xml*** and ***html*** files.The help viewer is enabled, when the user clicks the help button from the toolbar.The code which performs the above named task follows:

```
private void getHelp()
{
   try {

          /* loading the helpset */
         ClassLoader cl = ExecuteCommand.class.getClassLoader();
         URL url = HelpSet.findHelpSet(cl, helpsetName);

         mainHS = new HelpSet(cl, url);
      } catch (Exception ee)
        {
         System.out.println ("Help Set "+helpsetName+" not found"
                             +ee.getMessage());
         return;
        }
        catch (ExceptionInInitializerError ex)
        {
```

```
         System.err.println("initialization error:");
         ex.getException().printStackTrace();
        }
        /* mainHB's type is HelpBroker;The HelpBroker is the default
         * presentation of a HelpSet. A HelpBroker can be asked to show
         * a given Navigational View, and can display a given ID (help topic).
         */
        mainHB = mainHS.createHelpBroker();

        JButton bHelp=(JButton)App.jToolBar.getComponentAtIndex(App.HELP);
        bHelp.addActionListener(new CSH.DisplayHelpFromSource(mainHB));
  }
```

# Conclusions

A better understanding of the facial expressions analysis could be achieved, when trying to go deeper and find details concerning facial expressions, when trying to expand our horizons with respect to facial expressions analysis.

The aim of the first part of my final project was to be an overall view of the facial expressions analysis.There are summarized some of the methods involved in each phase of the analysis naming:face detection, facial information extraction and facial expressions classification.In fact as you could see, there are only methods suited for static face images and face models which fit 2D views of the face. The possible use of the face technology is stated in the first chapter, as well as its outcome benefits when interacting with computer's world.

Actually, just the second part and the third part of this paper are really the results of my work and now I can say that more sophisticated mathematical models are needed to perform a reliable extraction of facial data(in this case, fiducial points) as long as these results are fundamental to succeed in classifying an expression of the face profile.

This is an end and also a beginning(especially for me), if you think it is possible to have friendly machines(computers,robots) capable to carry out various task, but also to express some warm feelings such as kindness or sympathy to human beings.

# Appendix A

# Fiducial points and Action Units

Table A.1: Facial points of the side-view

| Point | Description |
|---|---|
| P1 | *Top of forehead*- the beginning of the forehead line. This point corresponds to the point where the hair started |
| P2 | *Eyebrow* - the point where the eyebrow contacts the profile |
| P3 | *Valley below the eyebrow* - the point with maximum curvature in front of the eye |
| P4 | *Tip of the nose* - the point with maximum curvature on the top of the nose |
| P5 | *Valley bellow the nose* - the point of maximal curvature of the line below the nostrils |
| P6 | *Upper lip* - the point with maximum curvature of the upper lip |
| P7 | *Valley between lips* - the point with maximum curvature between the upper and lower lips |
| P8 | *Lower lip* - the point with maximum curvature of the lower lip |
| P9 | *Valley below the lower lip* - the point in the valley with maximum curvature |
| P20 | *Tip of the chin* - the point with maximum curvature in the chin |

Table A.2: Representation of AUs with the defined side-view model

| AU | Movement of profile points |
|---|---|
| 1 | P2 upwards,curvature between P1 and P2 contains a row of peaks and valleys |
| 1+2 | P2 upwards,curvature between P1 and P2 remains without local extremities |
| 4 | P2 downwards,curvature between P2 and P3 is not increased |
| 8 | Distance P5-P6 increased,P6 outwards, P8 outwards,curvatures between P6 and P8 is more straight and angular,distance P8-P10 increased |
| 9 | Curvature between P2 and P3 increased |
| 10 | P6 upwards and outwards,distance P5-P6 decreased,curvature between P2 and P3 is not increased |
| 12 | Distance P5-P6 decreased,P6 inwards, P8 inwards,distance P6-P8 increased |
| 13 | Distance P5-P6 decreased,P6 inwards, P8 inwards,distance P6-P8 remains same |
| 15 | Distance P5-P6 increased,curvature between P5-P6 is not increased,P6 downwards,P8 downwards,distance P6-P8 not decreased |
| 16 | P8 downwards and outwards,distance P8-P10 decreased |
| 17 | P10 inwards |
| 18 | P6 outwards,P8 outwards,curvature between P6 and P8 is not angular |
| 19 | Tongue showed, curvature between P6 and P8 contains two valley and a peak |
| 20 | Distance P5-P6 increased, curvature between P5 and P6 is not increased,P6 inwards,P8 inwards,distance between P6-P8 not decreased |
| 23 | Distance P5-P6 increased, curvature between P5 and P6 is not increased,P6 downwards and inwards,P8 upwards and inwards, distance between P6-P8 decreased but it is ¿0 and ¿threshold1 |
| 24 | Distance P5-P6 increased, curvature between P5 and P6 is not increased,P6 downwards and inwards,P8 upwards and inwards, distance between P6-P8 decreased but it is ¿0 and ¡threshold1 |
| 25 | Distance P6-P8 is increased,distance P4-P10¡threshold2 |
| 26 | Distance P4-P10 between threshold2 and threshold3 |
| 27 | Distance P4-P10¿threshold3 |
| 28 | Points P6 and P8 are absent |
| 28b | Point P8 is absent |
| 28t | Point P6 is absent |
| 36t | Bulge above the upper lip produced by the tongue,curvature between P5 and P6 is increased |
| 36b | Bulge under the lower lip produced by the tongue,P9 is absent |

Table A.3: Description of Ekmans basic emotions

| **Emotion** | **Facial Feature** | **Action Unit** |
|---|---|---|
| Surprise | Eyebrows are raised | AU1,AU2 |
| | Upper eyelid is raised slightly | AU5 |
| | Jaws dropped; mouth opened | AU26 |
| Fear | Eyebrows are raised high and pulled together | AU1, AU2, AU4 |
| | Upper eyelid is raised; lower eyelid tightened | AU5, AU7 |
| | Mouth dropped open, lips are stretched horizontally | AU20, AU26 |
| Disgust | Nose is wrinkled so that lips are parted | AU9 |
| | Upper lid is raised | AU10 |
| | Lower lip is pulled down | AU16 |
| | Tongue is moved forward in the mouth and can be put out | |
| Anger | Eyebrows are pulled together down | AU4 |
| | Upper eyelid is raised AU5 Lower eyelid is tightened, but upper eyelid is kept raised | AU7 |
| | Lips are tightened, pushed up and pressed together | AU23,AU24 |
| Happiness | Cheeks are raised | AU6 |
| | Corner of the lips are raised | AU12 |
| Sadness | Inner corners of eyebrows are raised and pulled down | AU1,AU4 |
| | Corners of the lips are pulled down | AU15 |
| | Cheeks are raised;lip corners are pulled down | AU6, AU12 |