



# **The automatic recognition of facial expressions**

Delft University of Technology (TU Delft)

-

Electrical Engineering, Mathematics and Computer Science  
Knowledge Based Systems (EEMCS – KBS)

**Project Report, March 6th to August 31st 2006**

**Simon CHENET**

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## **Acknowledgements**

Above all, I would like to thank all researchers and students of KBS department (Knowledge Based Systems) of TU Delft for their reception and their sympathy.

I specially would like to thank Professor Leon Rothkrantz for his warm welcome, his helpfully kindness, and Datcu Dragos for his availability and his helpfulness.

Thank you also to all students, friends and family who supported me here or far away: Flavie, Anne and Xavier, Jean-Baptiste, Cornelia, Julia(s), Teva and everybody, I don't forget.

A final thank to ENSEIRB administration, which helped me for various steps, especially Ms. Evelyne Blanchard who made this stay in Netherlands possible.

## Contents

<b>1.</b>	<b>Introduction .....</b>	<b>5</b>
<b>2.</b>	<b>Background .....</b>	<b>7</b>
<b>3.</b>	<b>Related Works.....</b>	<b>11</b>
<b>3.1</b>	<b><i>Facial expressions</i> .....</b>	<b>11</b>
3.1.1	Universality of some facial expressions.....	11
3.1.2	Cultural display rules .....	12
3.1.3	Representation of facial expressions .....	13
3.1.4	Action Units: a universal representation of facial expressions.....	14
<b>3.2</b>	<b><i>Automatic facial expression recognition</i> .....</b>	<b>14</b>
3.2.1	Modeling facial expressions .....	14
3.2.2	Face detection and facial features extraction .....	16
3.2.3	Classification.....	18
<b>3.3</b>	<b><i>Information retrieval</i> .....</b>	<b>19</b>
<b>4.</b>	<b>Conceptual Design .....</b>	<b>23</b>
<b>4.1</b>	<b><i>Model design</i> .....</b>	<b>23</b>
4.1.1	Statistical Shape Model.....	24
4.1.2	Statistical Appearance Model.....	28
<b>4.2</b>	<b><i>The training set</i>.....</b>	<b>32</b>
<b>4.3</b>	<b><i>Fitting the model to a new image</i> .....</b>	<b>34</b>
4.3.1	Choice of fit function .....	35
4.3.2	Active Appearance Model.....	35
<b>4.4</b>	<b><i>Features extraction</i> .....</b>	<b>39</b>
<b>4.5</b>	<b><i>Recognizing facial expressions</i>.....</b>	<b>40</b>
4.5.1	Template-based classification.....	40
4.5.2	Artificial Neural Networks.....	41
4.5.3	Bayesian Belief Networks .....	42
<b>5.</b>	<b>System implementation.....</b>	<b>45</b>
<b>5.1</b>	<b><i>Preprocessing the data set</i>.....</b>	<b>45</b>
5.1.1	Annotating pictures .....	45
5.1.2	AAM files .....	48
<b>5.2</b>	<b><i>AAM-API: generating the model</i> .....</b>	<b>48</b>
5.2.1	Introduction .....	48
5.2.2	Model generation .....	49
5.2.3	ASF files .....	49
<b>5.3</b>	<b><i>Database conversion</i>.....</b>	<b>50</b>
5.3.1	ModelConvertor classes .....	50
5.3.2	Normalizing points .....	53
5.3.3	Writing ASF files.....	56

5.3.4	Converting pictures.....	57
<b>5.4</b>	<b>Active Appearance Search.....</b>	<b>57</b>
5.4.1	Initialization method.....	58
5.4.2	Optimization method.....	58
<b>5.5</b>	<b>Results.....</b>	<b>59</b>
5.5.1	Generating the model.....	59
5.5.2	Active Appearance Model search.....	59
<b>6</b>	<b>Conclusion.....</b>	<b>61</b>
	<b>Appendix.....</b>	<b>63</b>
<b>A.</b>	<b>Specifications.....</b>	<b>63</b>
i.	Softwares needed.....	63
ii.	System specifications.....	63
iii.	Anticipated Schedule.....	63
<b>B.</b>	<b>Example of an AAM file.....</b>	<b>64</b>
<b>C.</b>	<b>Example of an ASF file.....</b>	<b>68</b>
<b>D.</b>	<b>Configuration file used for generating the model.....</b>	<b>71</b>
<b>E.</b>	<b>AAM-API class graph.....</b>	<b>72</b>
<b>F.</b>	<b>Model results.....</b>	<b>73</b>
	<b>Bibliography.....</b>	<b>79</b>

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## 1. Introduction

People communicate with each other through spoken words and nonverbal behavior. Verbal communication is used to convey objective information, whereas nonverbal communication is used to convey subjective and affective information.

Due to a number of reasons, confusion and misunderstandings can arise when people communicate with each other.

A verbal dictionary can be used to look up the spelling of a word, sometimes the phoneme representation, the meaning in different contexts and rules of transformation.

Facial expressions play an important role in human communication. The contours of the mouth, eyes and eyebrows play an important role in classification.

The automatic recognition of facial expressions is a difficult problem because of changing light conditions, posture and occlusion. In the past several techniques have been developed such as using templates or splines to find the contours of the mouth. Or to locate special points around the contours of the mouth and use a classifier to put facial expressions in predefined classes (i.e. happiness, sadness, disgust, fear, anger and surprise).

Another method is to use vector flow in video recordings of facial expression. A promising method to classify facial expressions in still pictures is to use ideas from Viola and Jones. They select some basic features from a picture and then use a classifier to select the most promising features to classify faces in predefined classes.

The goal of the project is to design and implement an algorithm which is able to classify a grey-level picture of a front-view facial expression in predefined classes (i.e. 6 basic emotions), with no need of a human pre-processing the picture.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## 2. Background

I carried out my final project at TU Delft, in the Netherlands, within the framework of the exchange program Socrates-Erasmus.

I worked from March 6th to August 31st, 2006 within the research laboratory of EEMCS faculty of TU Delft, inside the KBS group (Knowledge Based Systems), part of the Men-Machine Interaction department.

Leon Rothkrantz, my professor at TU Delft and Datcu Dragos, as a PhD student at KBS, have supervised me. This training period had for main personal objectives:

- To train theoretically and practically my knowledge acquired during 3 years at ENSEIRB... particularly skills related to my third year course in multimedia technologies.
- To work on an international context in order to improve my English by using it both for work and everyday life.
- But also to know how research way of working is, discover the Netherlands and its culture, meeting intercultural people to learn from them.

My project was part of the whole ISFER project, and meant to be an extension of a previous project called FED.

### **ISFER: (Integrated System for Facial Expression Recognition)**

The goal for ISFER project is to develop an automated system for facial expression recognition.

Since 1992 the research in this field was done by the Knowledge Based Systems group of the TU Delft. In the course of the years a lot of working prototypes have been developed for different parts of the process of analyzing the face. Some of these prototypes are based on neural networks and fuzzy logic, while others are based on image processing techniques.

In 1996 the Human Emotion Recognition Clips Utilized Expert System (HERCULES) was developed by M. Pantic. This system performs reasoning about the emotions based on characteristics of the facial picture which can be obtained in automated way.

In 1997 the idea of the common workbench for all the programs developed so far became reality thanks to M. van Schouwen and J. Vollering. Also in this year the name Integrated System for Facial Expression Recognition (ISFER) was given in order to group all the works in this field.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## **FED, E. de Jongh, 2002: (Facial Expression Dictionary)**

The goal of this project was to develop a prototype of an online Facial Expression Dictionary, or FED for short, as a first step in the creation of a complete Nonverbal Dictionary. A complete Nonverbal Dictionary would contain information about all the ways people communicate with each other nonverbally.

Instead of words, FED contains information about facial expressions.

FED had several goals to reach:

- become available as a website
- offer the possibility to issue a nonverbal query through (multimodal) content

With FED, issuing a nonverbal query is done through uploading a picture containing a facial expression, after which the user semi-automatically determines the location of the face and the coordinates of 30 Facial Characteristics Points or FCPs. FED then determines the label of the unknown facial expression by comparing the FCP coordinates to the FCP coordinates of all entries present in the database.

Other query possibilities have been implemented as well. It is possible to look for entries in FED on facial expression label, active Action Units (an Action Unit or AU is a group of FCP correlations describing muscles movements, see section 4.1.4) or specific geometrical features. Finally, it is possible to look for entry incrementally, were the user iteratively selects the facial expression that resembles the facial expression he is looking for the closest.

The concept of FED as an online Facial Expression Dictionary was tested and found to be a viable approach. The FED system is easy to use, adapt, extend and manage. The approach taken with FED could be used to create a complete Nonverbal Dictionary.

Actually, if FED system provides a viable approach and good answers to the problem, some features were still to be possibly improved or added.

### 3D faces instead of 2D faces

Currently, the face model used by FED is Kobayashi & Hara face model (Figure 2.1).



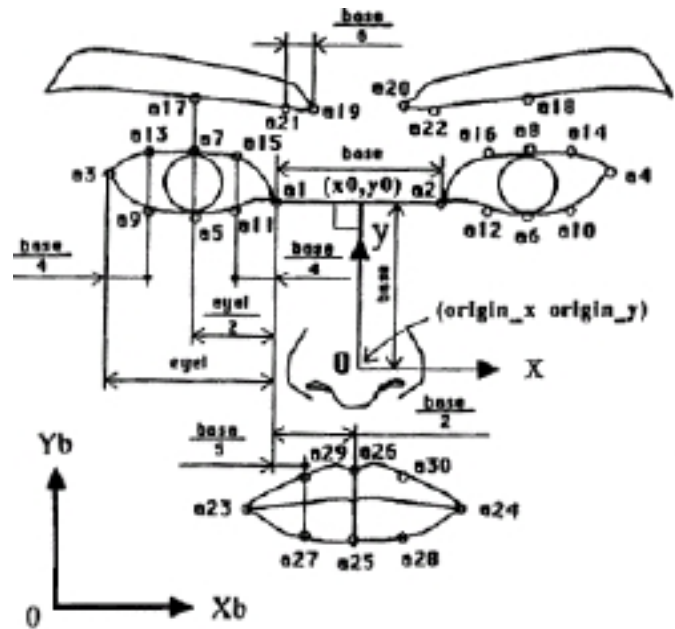


Figure 2.1: The Face Model developed by Kobayashi and Hara

Naturally, a 3D image of a facial expression would contain more information than a 2D image. It would thus be nice to introduce this concept.



Figure 2.2: The Kobayashi and Hara Face Model as it is in FED

Several methods to simulate the “3Dness” of faces exist, based on physical or statistical models, often utilizing grey-levels as additional information. Points still are computed in

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

2D-coordinates (from 2D-images), but the grey-level intensity is considered as a third dimension.

#### Fully automatic facial expression labeling

The FED system allows users to determine the label of a facial expression shown in a picture. This is accomplished semi-automatically: the user has to select a subpart of the picture containing the face and determine the positions of the 30 FCPs manually.

If the face detection and feature extraction step could be performed fully automatically, the user friendliness and performance of this type of FED query could be improved.

Image processing techniques can be used to automatically determine the location of the face. Also, the FCPs should be positioned automatically and if needed, displayed to the user, who could then manually adjust the FCPs if he thinks some are not positioned correctly.

This could save the user time, and there is no error introduced in the position of the FCPs due to incorrect positioning by the user. Of course, the algorithm that determines the positions of the FCPs would introduce an error too, but this method is still expected to be more robust.

My mission in the end was to build a good face model to recognition, easy to integrate on FED or a new facial expression recognition system, bringing improvements in order to make it fully automatic and trying to get even more satisfactory results.

## **Overview**

Chapter 3 will give an overview of the theoretic background in which the facial expression recognition is set. First, the basic psychological concepts currently accepted on facial expressions are briefly introduced. Also in this chapter is an overview of the fields of automatic facial expression, with descriptions of several methods implemented in previous works. Finally, the subject of information retrieval is briefly reviewed here.

Chapter 4 describes the conceptual design of the system, from entries (pictures) to criterions used for expression recognition, passing by the model chosen to describe pictures.

Chapter 5 describes the implementation details of the main algorithms that were used to implement the system.

Finally, chapter 6 is devoted to conclusions. This includes a discussion on to what extent the implementation goals are met and conclusions regarding the research goals based on the performance and characteristics of the system.

### 3. Related Works

#### 3.1 Facial expressions

Facial expressions play an important role in nonverbal communication. Psychological research has shown that facial expressions are an indication of someone's emotional state.

##### 3.1.1 Universality of some facial expressions

Cross-cultural psychological research on facial expressions indicates that there may be a small set of facial expressions that is universal. This was first suggested by Charles Darwin in his work *On the Origin of Species*. Psychologists Paul Ekman and Wallace Friesen (1972), and independently, Carroll Izard (1971) conducted the first methodologically sound studies, and concluded that the emotions Happiness, Anger, Sadness, Disgust, Surprise and Fear are shown and interpreted in all human cultures in the same way. It should be noted that not all social psychologists accept these conclusions (e.g., J.A. Russell, 1994). Figures 3.1 and 3.2 show one example for each of those 6 emotions.



Figure 3.1: 3 of the 6 universal expressions of facial emotion



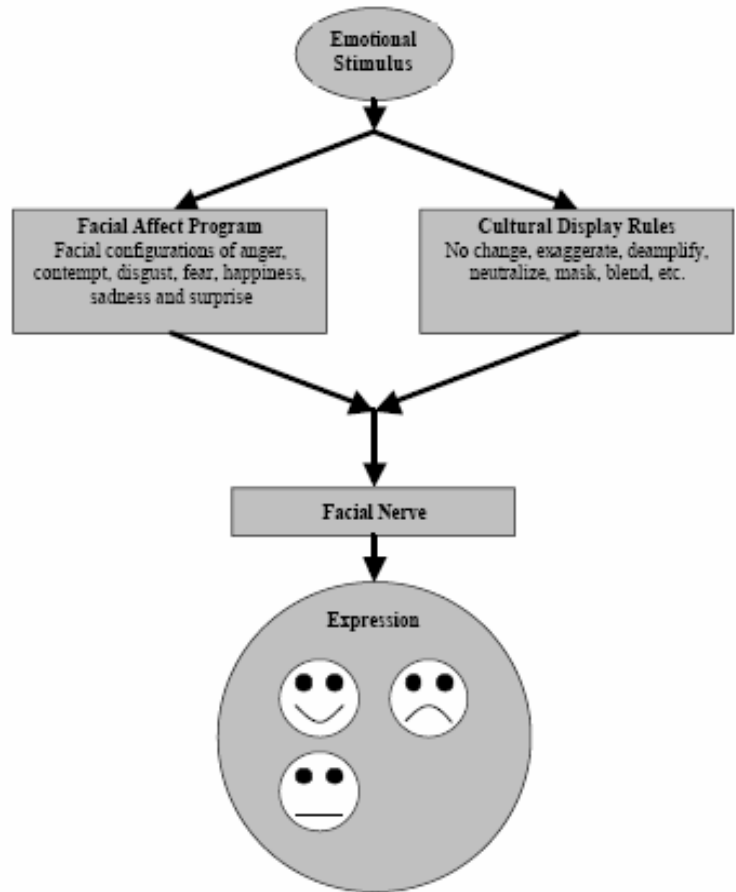
**Figure 3.2: The 3 other universal expressions of facial emotion**

Ekman and Friesen called their 6 emotions the 6 basic emotions. Even though their research suggests that all humans are born with the ability to express and recognize these 6 basic emotions, it does not imply that emotions are actually displayed, experienced and interpreted in the same way across all cultures.

### 3.1.2 Cultural display rules

Ekman and Friesen explained this discrepancy by introducing the concept of cultural display rules. Cultural display rules determine how basic universal emotions are modified in a certain culture in certain social circumstances. They performed a study where American and Japanese subjects were shown highly stressful films while their facial expressions were being monitored.

When watching the films alone, both American and Japanese subjects expressed negative feelings of disgust, anger, sadness and fear. When the experiment was done while an older, higher-status experimenter was with them in the room, the Japanese carefully hid their emotions, while the Americans continued to show their emotions. The conclusion is that facial expressions of emotion are influenced by universal, biological factors and also by culturally dependent learned display rules. Figure 3.3 gives a schematic overview of the principle of display rules. An emotional stimulus triggers an event where a universal facial expression is modified as a result of the relevant cultural display rule(s) to create the final facial expression displayed by a person.



**Figure 3.3: The neuro-cultural theory of emotional expression  
- Cultural display rules**

### 3.1.3 Representation of facial expressions

When developing a facial expression recognition system, it is important to realize that there are many possibilities that exist to represent a facial expression. Facial expressions can be represented through:

- Pictures
- Video
- Cartoons
- Smiley
- Facial characteristic points
- Active Action Units (see next section)

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

As mentioned in the introduction, the basis of a facial expression entry in the system is a picture.

### 3.1.4 Action Units: a universal representation of facial expressions

The findings on the universality of 6 basic emotions inspired researchers to try and find a way to measure facial expressions, so that emotions could be objectively measured (instead of relying on the subjective interpretation of an observer). Probably the most prominent and most used technique to emerge is the Facial Action Coding System (FACS), developed by Ekman and Friesen in 1978. In their research they define 44 so-called Action Units (AUs). Each action unit describes the movement of certain muscle(s) of the face. Every facial expression can then be described in terms of which AUs are active, i.e. which muscles are flexed and which muscles are relaxed.

## 3.2 Automatic facial expression recognition

Automatic facial expression recognition can be of importance for applications in the field of human-computer interfaces, monitoring and education. Examples are a computer system interface which gives feedback to the user depending on the emotional state of the user, or a monitoring system in the cockpit of a plane, which alerts the people in the control tower when the pilot becomes stressed.

Because of the many possible applications, research on automatic facial expression recognition has been conducted since the 1970's. The first subsection describes how facial expressions are modeled in current automatic facial expression recognition systems. The next two sections describe the three steps that have to be performed by any automatic facial expression recognition system: face detection, facial features extraction and facial expression classification.

### 3.2.1 Modeling facial expressions

Basically all of the current automatic facial expression recognition systems use one of three methods to model / represent a facial expression. The facial expression is either represented in the system as a whole (holistic representation), as a set of facial characteristic points or contours describing the eyes, eyebrows and mouth (analytic representation) or as a combination of these (hybrid approach).

An example of the analytic representation is the method of Kobayashi and Hara (1992), where the face is modeled as a set of 30 facial characteristic points.

Terzopoulos and Waters (1993) used the holistic approach and modeled the face as a 3D wire frame model with texture mappings. An example of the hybrid approach is provided by Thalmann (1998), who modeled the face as a wire frame model combined with a number of 3D facial characteristic points.

Another method for representing facial expressions, closely related to the representation through AUs, is representation of facial expressions through so-called *Facial Animator Parameters* or *FAPs*. The FAPs are based on a study of minimal perceptible actions.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

There are 68 FAPs, divided into 10 groups based on different parts of the face. Representation of facial expressions through FAPs is especially useful when trying to animate faces.

Since face is a deformable object, and in order to go further than just modeling it, looking at approaches to modeling variability is needed. The most common general approach is to allow a prototype to vary according to some physical model. Bajcsy and Kovacic (1989) described a volume model (of the brain) that also deforms elastically to generate new examples. Christensen and others (1995) described a viscous flow model of deformation which they also applied to the brain, but it is very computationally expensive. Park (1996) on one hand, and Pentland and Sclaroff (1991) on the other hand both represented the outline or surfaces of prototype objects using finite element methods and describe variability in terms of modes of vibration. Such modes are not always appropriate description of deformation and thus not very useful for recognition. This is primarily because they normally have more degrees of freedom than there are sensor measurements, so that the recovery process is under-constrained. Therefore, although heuristics such as smoothness or symmetry can be used to obtain a solution, they do not produce a stable, unique solution. Turk and Pentland (1991) used principal component analysis (PCA) to describe face images in terms of a set of basis functions, known as "eigenfaces", because they are the eigenvectors (principal components) of the training set of faces. Though valid modes of variation are learnt from the training set, and are more likely to be more appropriate than a physical model, an eigenface is not robust to shape changes, and does not deal well with variability in pose and expression, and is actually only well suited to face recognition for identification. However, the model can be matched to an image easily using correlation based methods.

3D models also had been described. Poggio and co-workers described image-based modeling techniques that make possible the creation of photo-realistic computer models of real human faces (1996). They were able to synthesize new views of a face from a set of examples views of the face. They fitted the so-called "morphable" model to an unseen view by a stochastic optimization procedure, a gradient descent algorithm (1998). This is slow, but can be robust because of the quality of synthesized images. Cootes and Taylor (1994) described a 3D model of the grey-level surface, allowing full synthesis of shape and appearance. However they had not suggested a plausible search algorithm to match their grey-level/shape combined model to a new image yet. Nastar and others (1996) described a related model of the 3D grey-level surface, combining physical and statistical modes of variation. Though they applied it to an image matching search algorithm, it requires a very good intensity and spatial smoothing initialization.

Cootes and Taylor (1995) described a shape and local grey-level appearance, using Active Shape Models (ASMs) to locate flexible objects in new images. ASM method is analytically based on points and contours.



**Figure 3.4: Example face image annotated with ASM landmarks**

Lanitis (1997) used this approach to interpret face images. Having found the shape using an ASM, the face is warped into a normalized frame, in which a model of the intensities of the shape-free face is used to interpret the image. Edwards (1997) extended this work to produce a combined model shape and grey-level appearance, but again rely on the ASM to locate faces in new images. The model used in this paper is called Active Appearance Model (AAM, Cootes, Taylor and Edwards, 1998) and is a direct further extension of this idea, using this time all the information in the combined appearance model to fit to the image; it will be described precisely in section 5.

### 3.2.2 Face detection and facial features extraction

The first step that has to be performed by an automatic facial expression recognition system is, given an input image, to determine the position of the face. Automatic detection of the position of the face is complex because of the fact that the size and orientation of the head may differ for different input images. A simple method is to assume input images where the face is visible are in frontal view. In general, methods from the field of image processing are applied to detect the face in an input image.

Detecting a face with its facial features is distinguishing the face and non-faces. This is done by using a classifier. There are different kinds of classifying methods. Some well known examples are K-Nearest Neighbours (KNN), Tree-Augmented Naive-Bayes (TAN) and Support Vector Machines (SVM). The latter is based on some rather simple ideas and provides a clear intuition of what learning from examples is about: to emphasize, starting with two classes to separate, it looks for the hyperplane with the maximum margin between the two classes, where the margin is defines as the sum of the hyperplane from the closest point of the two classes. Practical applications have already shown outstanding high performances of this classification method, like Osuna's face detection in images (1997). However, despite its success, there are some



significant and practical disadvantages in the SVM learning methodology, like being computationally expensive for datasets with thousands of entries.

Viola & Jones (2001) described an object detection method they successfully applied to face detection. Using an original image representation called Integral Image, where the value associated with a point is the sum of all the pixels above and to the left, they applied a variant of AdaBoost learning algorithm. AdaBoost is an adaptive algorithm able to boost a so-called weak learner by adjusting it step by step with weights accordingly to the global error (sum of errors) on a set of classified pictures.

Another classification method, based on the idea of the Support Vector Machine, is the Relevance Vector Machine (RVM) by Tipping (2001). RVM is a Bayesian framework for regression and classification with analogous sparsity properties as the SVM. It can be seen as a probabilistic version of SVM but without the disadvantages and simultaneously providing a number of additional advantages, including the benefits of probabilistic predictions, automatic estimation of parameters and the facility to use arbitrary basis functions, which are not necessary 'Mercer' kernels as in SVM (kernels are functions known to pre-process well a dataset such as de-noising, see Figure 3.5). Wong (2005) used RVM to detect face and extract features on a video-based recognition system.

Gaussian RBF

$$k(\mathbf{x}, \mathbf{z}) = \exp\left(\frac{-\|\mathbf{x} - \mathbf{z}\|^2}{c}\right)$$

Polynomial

$$k(\mathbf{x}, \mathbf{z}) = ((\mathbf{x}^T \mathbf{z}) + \theta)^d$$

Sigmoidal

$$k(\mathbf{x}, \mathbf{z}) = \tanh(\kappa(\mathbf{x}^T \mathbf{z}) + \theta)$$

Inverse multi-quadric

$$k(\mathbf{x}, \mathbf{z}) = \frac{1}{\sqrt{\|\mathbf{x} - \mathbf{z}\|^2 + c^2}}$$

**Figure 3.5: Mercer kernels**

The next step that has to be performed by an automatic facial expression recognition system is the automatic extraction of facial expression feature information. The method used is dependent on the representation method of the face and the kind of input images (static or dynamic). If the analytical approach is used to model the face, the relative positions and distances between the facial characteristic points are used for facial expression recognition. If the face is modelled as a whole, any data structure that describes the face as a whole (a complete 2D array of intensity values of the image, a labelled graph) can be used to represent the facial expression information. If the hybrid approach is used, some facial characteristic points usually determine the initial position of a certain template.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

The two steps can also be done at the same time. Covell (1996) demonstrated that the parameters of an eigen-feature model can be used to drive shape model points to the correct place. The AAM is an extension of this idea. Black and Yacoob (1995) used local, hand crafted models of image flow to track facial features, in which the image difference patterns corresponding to changes in each model parameter are learnt and used to modify a model estimate. However they did not attempt to model the whole face. The AAM can be thought of as a generalisation of this, in which the image difference patterns corresponding to changes in each model parameter are learnt and used to modify a model estimate.

Fast model matching algorithms have been developed in the tracking community. Gleicher (1997) described a method of tracking objects by allowing a single template to deform under a variety of transformations (affine, projective etc). He chose the parameters to minimize a sum of squares measure and essentially pre-computes derivatives of the difference vector with respect to the parameters of the transformation. Hager and Belhumeur (1998) described a similar approach, but include robust kernels and models of illumination variation.

In a parallel development Sclaroff and Isidoro (1998) have described “Active Blobs” tracking method. The approach is broadly similar in that they use image differences to drive tracking, learning the relationship between image error and parameter offset in an off-line processing stage. The main difference is that Active Blobs are derived from a single example, whereas AAMs use a training set of examples. Thus, Active Blobs use a single example as the original model template, allowing deformations consistent with low energy mesh deformations (derived using a Finite Element method). A simple polynomial model is used to allow changes in intensity across the object. AAMs learn what shape and intensity variations are valid from their training set. Sclaroff and Isidoro also suggested applying a robust kernel to the image differences.

La Cascia and others (2000) described a related approach to head tracking. They projected the face onto a cylinder (or more complex 3D face shape) and use the residual differences between the sampled data and the model texture (generated from the first frame of a sequence) to drive a tracking algorithm, with encouraging results.

### 3.2.3 Classification

The final task to be performed by an automatic facial expression recognition system is the classification of the facial expression displayed in the input image into a certain category. Successful classification is only possible if the input images are normalized in some way, so that images can be compared with each other. Normalizing all input images goes through defining the characteristic points in a coordinate system which is independent of the size of the input image and of the size of the face in the input image. Another important issue involved in the classification is defining a set of categories into which the input images are to be classified. This depends on the application domain of the facial recognition system. If the system is to be used as a tool for behavioural research, it may be desirable to determine and quantify the AUs present in the input

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

image. In other cases it may be useful to classify the input image into an emotion category, which is the purpose here.

A number of categorization mechanisms can be used. With template-based classification (like in Wong 2005), the unknown facial expression is compared with templates representing the classification categories (for example the 6/7 basic emotions). The image is classified into the category of the template to which it is closest.

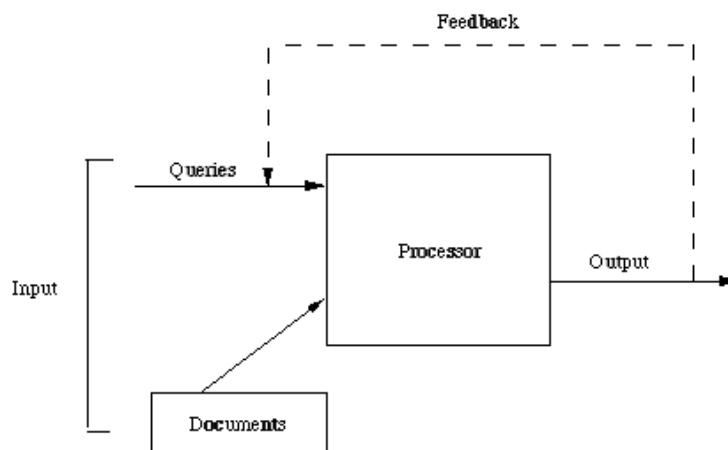
Neural networks can also be used as a classification method. Neural networks are an example of a black box approach. The neural network is trained by using a set of images that have been correctly classified as a certain emotion by a human expert. After training, the neural network can be used to correctly classify new images of which the corresponding emotions are unknown.

Another method often used for classification in facial expression recognition systems is a rule-based expert system. The expert system contains a knowledge base with information about facial expression features stored in the form of logical if-then rules. The facial features of the unknown facial expression are given as input into the knowledge base and the facial expression label is determined through logical inference. Datcu (2004) tried both latter approaches for his facial expression recognition system (Artificial Neural Network and Bayesian Belief Network), using Principal Component Analysis (PCA) for features extraction.

There are several other techniques from the field of Artificial Intelligence that can be used to implement the classification component of an automatic facial expression recognition system, such as case-based reasoning, fuzzy logic or genetic algorithms.

### **3.3 Information retrieval**

There are vast amounts of information available in (online) computer systems. The total amount of information stored worldwide increases each year. Retrieving relevant information speedy and accurately is becoming ever more difficult. Since the development of the computer, information retrieval systems have been created that aim to provide a solution. Although some advances have been made, the problem of retrieving all and only the relevant information is still largely unsolved. Figure 3.6 shows the general design of an information retrieval system.



**Figure 3.6: General design of an Information Retrieval System**

The actual implementation of an information retrieval system depends on the kind of information that is to be retrieved. The first information retrieval systems were used to search collections of text documents. Instead of parsing each document entirely, documents were usually characterized by a limited number of keywords. Advances in the field of natural language processing have made it possible to determine the relevancy of a document by looking at the entire text, but these methods are time-consuming and still far from perfect. Below an overview of the search techniques most commonly used by current information retrieval systems is given.

### Boolean search

A Boolean search strategy retrieves all information that evaluates as 'true' for the query. This formulation only makes sense if the queries are expressed in terms of index terms (or keywords) and combined by the usual logical connectives AND, OR, and NOT.

### Matching functions

A matching function calculates the degree of association between a query and a document or cluster profile. An example of a matching function is (With D the set of keywords representing the document, and Q the set representing the query):

$$M = \frac{2|D \cap Q|}{|D| + |Q|}$$

### Serial Search

With this type of search strategy, a matching function is used to calculate the degree of association between the query and a collection of documents. The documents can be

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

ranked by degree of association, or a threshold can be used to filter out unlikely matches.

#### Cluster representatives

A profile is defined for all identifiable clusters of documents in the total collection. The query is matched against these profiles, or cluster representatives. The cluster of documents corresponding to the best matching profile is returned.

#### Cluster-based retrieval

All clusters are ordered in a tree structure. The search starts by evaluating the value of a matching function for the top cluster (node 0). The search then proceeds to evaluate the matching function for the immediate descendants of the first node. This pattern repeats itself down the tree. The search is directed by a decision rule, which on the basis of comparing the values of a matching function at each stage decides which node to expand further. For example, the node with the highest matching function value could be used. A stopping rule determines when to terminate the search and retrieve a cluster. The above search strategy can be described as a top-down search strategy. It is also possible to traverse the tree through a bottom-up strategy, with the terminal nodes of the tree evaluated first.

#### Relevance feedback

This search technique uses feedback provided by the user to iteratively improve the relevancy of the returned documents. Experiments have shown that relevance feedback can be very effective.

A problem with implementing a relevance feedback system is that it is difficult for users to determine the relevance or non-relevance of a document.

Except for text, computers are also used to store multimodal information. This can include images, music, video etc. Multimodal Information Retrieval (MMIR) differs from text-based information retrieval in a number of ways. Different search techniques are used to retrieve the relevant multimodal information corresponding to a query. The most commonly used techniques are based on relevance feedback algorithms. Also, often a query can be issued using multimodal content.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## 4. Conceptual Design

As seen before, several steps have to be performed to build a system for automatic facial expression recognition. A suitable model has to be chosen, which will permit to learn the characteristics of 6 basic facial expressions. The Active Appearance Model used in this project is thus presented in this chapter.

First, the analytic representation approach of AAM is presented, showing how facial expressions are modelled, from a well-prepared set of pictures with landmarks describing face, nose, eyes, eyebrows and mouth contours. The two statistical models combined to build AAM are thus consecutively explained so as the training set preparation.

Then, the search algorithm is studied, presenting how a new “unseen” facial image is fitted to the model built from the training set, and thus approximated in its statistical representation.

Finally, the way facial expressions are recognized is explained.

### 4.1 Model design

Because faces are classes of objects which are not identical, variability has to be dealt with. This leads naturally to the idea of deformable models - models which maintain the essential characteristics of the class of objects they represent, but which can deform to fit a range of examples. There are two main characteristics such models have to possess. First, they should be general - that is, they should be capable of generating any plausible example of the class they represent. Second, and crucially, they should be specific - that is, they should only be capable of generating 'legal' examples that is to say limiting the attention of our system to plausible interpretations. In order to obtain specific models of variable objects, acquiring knowledge of how they vary is essential.

Where structures vary in shape or texture, it is possible to learn which variations are plausible and which are not. A new image can be interpreted by finding the best plausible match of the model to the image data. The advantages of such a method are that:

- It is widely applicable. The same algorithm can be applied to many different training examples, providing answers not only to facial expression recognition.
- Expert knowledge can be captured in the system in the annotation of the training examples.
- The models give a compact representation of allowable variation, but are specific enough not to allow arbitrary variation different from that seen in the training set.
- The system need make few prior assumptions about the nature of the objects being modelled, other than what it learns from the training set (for instance, there are no boundary smoothness parameters to be set).

The models described below require a user to be able to mark 'landmark' points on each of a set of training images in such a way that each landmark represents a distinguishable point present on every example image. For instance, when building the model of appearance of the eyes in face images, good landmarks would be the corners of the eye, as these would be easy to identify and mark in each image. Thus it requires that the topology of the object cannot change and that the object is not so amorphous that no distinct landmarks can be applied, which is the case with faces.

#### 4.1.1 Statistical Shape Model

Here is described the statistical model of shape which will be used to represent faces in images. The shape of an object is represented by a set of  $n$  2-dimensions points. Shape is usually defined as that quality of a configuration of points which is invariant under some transformation. In two dimensions, only the Similarity transformation (translation, rotation and scaling) will be considered. The shape of an object is not changed when it is moved, rotated or scaled.

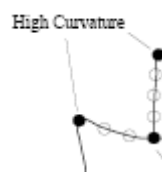
Our aim is to derive models which allow us to both analyse new shapes, and to synthesise shapes similar to those in a training set. The training set typically comes from hand annotation of a set of training images. By analysing the variations in shape over the training set, a model is built which can mimic this variation.

Much of the following will describe building models of shape under a similarity transform  $T_\theta$  (where  $\theta$  are the parameters of the transformation).

##### *Suitable Landmarks*

Good choices for landmarks are points which can be consistently located from one image to another. The simplest method for generating a training set is for a human expert to annotate each of a series of images with a set of corresponding points.

Points can be placed at clear corners of object boundaries, T-junctions between boundaries or easily located biological landmarks. However, there are rarely enough of such points to give more than a sparse description of the shape of the target object. This list would be augmented with points along boundaries.



**Figure 4.1: a T-junction**

If a shape is described  $n$  points in 2 dimensions the shape is represented by a  $2n$  element vector formed by concatenating the elements of the individual point position vectors.



ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

Thus, for a single example, the  $n$  landmark points  $\{(x_i ; y_i)\}$ , can be represented as the  $2n$  element vector,  $x$ , where:

$$x = (x_1, \dots, x_n, y_1, \dots, y_n)^T \quad (4.1)$$

Given  $s$  training examples,  $s$  such vectors  $x_j$  are generated. Before statistical analysis can be performed on these vectors it is important that the shapes represented are in the same coordinate frame in order to remove variation which could be attributable to the allowed global transformation,  $T$ .

### *Aligning the Training Set*

Though analytic solutions exist to the alignment of a set, a simple iterative approach is as follows:

1. Translate each example so that its centre of gravity is at the origin.
2. Choose one example as an initial estimate of the mean shape and scale so that  $|\bar{x}| = 1$ .
3. Record the first estimate as  $\bar{x}_0$  to define the default reference frame.
4. Align all the shapes with the current estimate of the mean shape.
5. Re-estimate mean from aligned shapes.
6. Apply constraints on the current estimate of the mean by aligning it with  $\bar{x}_0$  and scaling so that  $|\bar{x}| = 1$ .
7. If not converged, return to 4.

(Convergence is declared if the estimate of the mean does not change significantly after an iteration)

The operations allowed during the alignment will affect the shape of the final distribution. To keep the distribution compact and keep any non-linearity to a minimum, the tangent space approach is used.

Be:

$$D = \sum |x_i - \bar{x}|^2$$

The goal is to transform each shape into the tangent space to the mean so as to minimise  $D$ . The tangent space to  $x_t$  is the hyperplane of vectors normal to  $x_t$ , passing through  $x_t$ . That is to say all the vectors  $x$  such that  $(x_t - x) \cdot x_t = 0$ , or  $x \cdot x_t = 1$  if  $|x_t| = 1$ .

This preserves the linear nature of the shape variation. The simplest way to achieve this is to align the shapes with the mean, allowing scaling and rotation, then project into the tangent space by scaling  $x$  by  $1/(x \cdot \bar{x})$ .

### Modelling Shape Variation

Suppose now having  $s$  sets of points  $x_i$  which are aligned into a common coordinate frame. These vectors form a distribution in the  $2n$  dimensional space in which they live. If this distribution can be modelled, new examples can be then generated, similar to those in the original training set, and new shapes can be examined to decide whether they are plausible examples.

In particular a parameterised model is searched, with the form  $x = M(b)$ , where  $b$  is a vector of parameters of the model. Such a model can be used to generate new vectors,  $x$ 's. If the distribution of parameters  $p(b)$  can be modelled, they can be limited so that the generated  $x$ 's are similar to those in the training set. Similarly it should be possible to estimate  $p(x)$  using the model.

To simplify the problem, it is to reduce the dimensionality of the data from  $2n$  to something more manageable. An effective approach is to apply Principal Component Analysis (PCA) to the data. The data form a cloud of points in the  $2n$ -dimensions space. PCA computes the main axes of this cloud, allowing one to approximate any of the original points using a model with fewer than  $2n$  parameters. The approach is as follows:

1. Compute the mean of the data,

$$\bar{x} = \frac{1}{s} \sum_{i=1}^s x_i \quad (4.2)$$

2. Compute the covariance of the data,

$$S = \frac{1}{s-1} \sum_{i=1}^s (x_i - \bar{x})(x_i - \bar{x})^T \quad (4.3)$$

3. Compute the eigenvectors,  $\phi_i$  and corresponding eigenvalues  $\lambda_i$  of  $S$  (sorted so that  $\lambda_i \geq \lambda_{i+1}$ ).

If  $\Phi$  contains the  $t$  eigenvectors corresponding to the largest eigenvalues, then we can then approximate any of the training set,  $x$  using

$$x \approx \bar{x} + \Phi b \quad (4.4)$$

where  $\Phi = (\phi_1 | \phi_2 | \dots | \phi_t)$  and  $b$  is a  $t$  dimensional vector given by

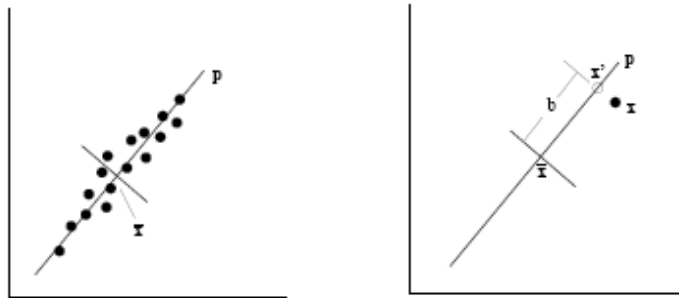
$$b = \Phi^T (x - \bar{x}) \quad (4.5)$$

The vector  $b$  defines a set of parameters of a deformable model. By varying the elements of  $b$  the shape  $x$  can vary using Equation 4.4. The variance of the  $i^{\text{th}}$  parameter,  $b_i$ , across the training set is given by  $\lambda_i$ . By applying limits of  $\pm 3\sqrt{\lambda_i}$  to the

parameter  $b_i$ , it is ensured that the shape generated is similar to those in the original training set.

The number of eigenvectors to retain,  $t$ , can be chosen so that the model represents some proportion (e.g. 98%) of the total variance of the data, or so that the residual terms can be considered noise (see next paragraph).

For instance, Figure 4.2 shows the principal axes of a 2D distribution of vectors. In this case any of the points can be approximated by the nearest point on the principal axis through the mean:  $x \approx x' = \bar{x} + bp$  where  $b$  is the distance along the axis from the mean of the closest approach to  $x$ . Thus the two dimensional data is approximated using a model with a single parameter,  $b$ . Similarly shape models controlling many hundreds of model points may need only a few parameters to approximate the examples in the original training set.



**Figure 4.2: Applying a PCA to a set of 2D vectors.  $p$  is the principal axis. Any point  $x$  can be approximated by the nearest point on the line,  $x'$**

### *Choice of Number of Modes*

The number of modes to retain,  $t$ , can be chosen in several ways. Probably the simplest is to choose  $t$  so as to explain a given proportion (e.g. 98%) of the variance exhibited in the training set.

Let  $\lambda_i$  be the eigenvalues of the covariance matrix of the training data. Each eigenvalue gives the variance of the data about the mean in the direction of the corresponding eigenvector. The total variance in the training data is the sum of all the eigenvalues,  $V_T = \sum \lambda_i$ .

Then the  $t$  largest eigenvalues are chosen such that:

$$\sum_{i=1}^t \lambda_i \geq f_v V_T \quad (4.6)$$

where  $f_v$  defines the proportion of the total variation one wishes to explain (for instance, 0.98 for 98%).

#### 4.1.2 Statistical Appearance Model

To synthesise a complete image of the face, both its shape and its texture has to be modelled. Here is described how statistical models can be built to represent both shape variation, texture variation and the correlations between them. Such models can generate photo-realistic synthetic images.

The models are generated by combining a model of shape variation with a model of the texture variations in a shape-normalised frame. “Texture” means the pattern of intensities or colours across an image patch.

Given the statistical model of shape variation, each training example can be warped into the mean shape, to obtain a “shape-free” patch (see Figure 4.4). Then a statistical model of the texture variation is built in this patch.

There will be correlations between the parameters of the shape model and those of the texture model across the training set. To take account of these a combined appearance model is built which controls both shape and texture.

The following subsections describe these steps in more detail.

##### *Statistical Texture Model*

To build a statistical model of the texture each example image is warped so that its control points match the mean shape using a piece-wise affine triangulation algorithm explained below.

Supposing an image  $I$ , the problem is to warp it so that a set of  $n$  control points  $\{x_i\}$  are mapped to new positions,  $\{x'_i\}$ . A continuous vector valued mapping function  $f$  is required, such that:

$$f(x_i) = x'_i \quad \forall i = 1 \dots n \quad (4.7)$$

Given such a function, each pixel of image  $I$  can be projected into a new image  $I'$ . In practice, in order to avoid holes and interpolation problems, it is better to find the reverse map,  $f'$ , taking  $x'_i$  into  $x_i$ . For each pixel in the target warped image  $I'$  is determined where it came from in  $I$  and fill it in. In general  $f' \neq f^{-1}$ , but it's a good approximation enough.

Note that  $f$  can always be break down into a sum:

$$f(x) = \sum_{i=1}^n f_i(x) x'_i \quad (4.8)$$

where the  $n$  continuous scalar valued functions  $f_i$  each satisfies:

$$f_i(x_j) = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases} \quad (4.9)$$

This ensures  $f(x_i) = x'_i$ .

The simplest warping function is to assume each  $f_i$  is linear in a local region and zero everywhere else.

In two dimensions, a triangulation (e.g. Delaunay) can be used to partition the convex hull of the control points into a set of triangles. To the points within each triangle can be applied the affine transformation which uniquely maps the corners of the triangle to their new positions in  $I'$ .

Suppose  $x_1$ ,  $x_2$  and  $x_3$  are three corners of such a triangle. Any internal point can be written:

$$\begin{aligned} x &= x_1 + \beta(x_2 - x_1) + \gamma(x_3 - x_1) \\ &= \alpha x_1 + \beta x_2 + \gamma x_3 \end{aligned} \quad (4.10)$$

where  $\alpha = 1 - (\beta + \gamma)$  and so  $\alpha + \beta + \gamma = 1$ . For  $x$  to be inside the triangle, the equation  $0 \leq \alpha, \beta, \gamma \leq 1$  has to be verified.

Under the affine transformation, this point simply maps to:

$$x' = f(x) = \alpha x'_1 + \beta x'_2 + \gamma x'_3 \quad (4.11)$$

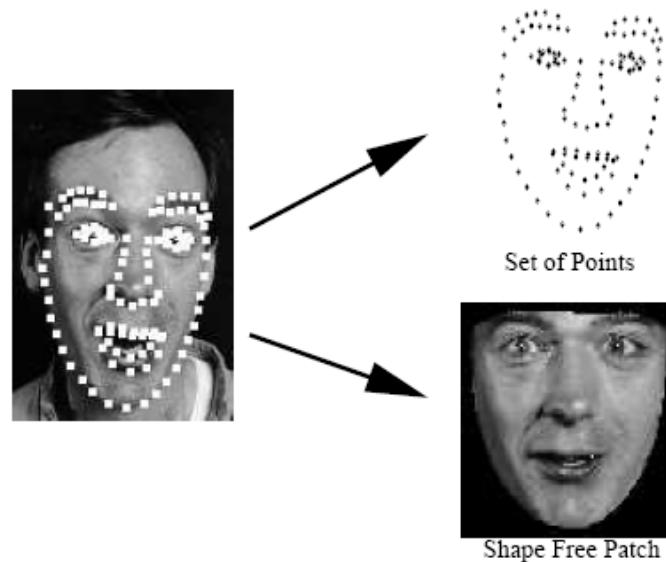
To generate a warped image each pixel  $x'$  in  $I'$  is taken, deciding which triangle it belongs to, computing the coefficients  $\alpha$ ,  $\beta$ ,  $\gamma$  giving its relative position in the triangle and using them to find the equivalent point in the original image,  $I$ . Sample is done from this point and the value into pixel  $x'$  in  $I'$  is copied.

This triangulation removes spurious texture variation due to shape differences which would occur if eigenvector decomposition on the un-normalised face patches was simply performed. Then the intensity information is sampled from the shape-normalised image over the region covered by the mean shape to form a texture vector,  $g_{im}$ . For example, Figure 4.3 shows a labelled face image, the model points and the face patch normalised into the mean shape.

The sampled patch contains little of the texture variation caused by the exaggerated expression - that is mostly taken account of by the shape.

To minimise the effect of global lighting variation, the example samples are normalised by applying a scaling,  $\alpha$ , and offset,  $\beta$ :

$$g = (g_{im} - \beta \mathbf{1}) / \alpha \quad (4.12)$$



**Figure 4.3: Each training example is split into a set of points and a 'shape-free' image patch**

The values of  $\alpha$  and  $\beta$  are chosen to best match the vector to the normalised mean. Let  $\bar{\mathbf{g}}$  be the mean of the normalised data, scaled and offset so that the sum of elements is zero and the variance of elements is unity. The values of  $\alpha$  and  $\beta$  required to normalise  $g_{im}$  are then given by:

$$\alpha = \mathbf{g}_{im} \cdot \bar{\mathbf{g}} \quad , \quad \beta = (\mathbf{g}_{im} \cdot \mathbf{1})/n \quad (4.13)$$

where  $n$  is the number of elements in the vectors.

Of course, obtaining the mean of the normalised data is then a recursive process, as the normalisation is defined in terms of the mean. A stable solution can be found by using one of the examples as the first estimate of the mean, aligning the others to it (using equations 4.12 and 4.13), re-estimating the mean and iterating.

By applying PCA to the normalised data a linear model is obtained:

$$\mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g \quad (4.14)$$

where  $\bar{\mathbf{g}}$  is the mean normalised grey-level vector,  $\mathbf{P}_g$  is a set of orthogonal modes of variation and  $\mathbf{b}_g$  is a set of grey-level parameters.

The texture in the image frame can be generated from the texture parameters  $\mathbf{b}_g$ , and the normalisation parameters  $\alpha$ ,  $\beta$ . For linearity, these are represented in a vector  $\mathbf{u} = (\alpha - 1, \beta)^T$ . In this form the identity transform is represented by the zero vector.

The texture in the image frame is then given by:

$$\mathbf{g}_{im} = T_{\mathbf{u}}(\bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g) = (1 + u_1)(\bar{\mathbf{g}} + \mathbf{P}_g \mathbf{b}_g) + u_2 \mathbf{1} \quad (4.15)$$

### *Combined Appearance Model*

The shape and texture of any example can thus be summarised by the parameter vectors  $\mathbf{b}_s$  and  $\mathbf{b}_g$ . Since there may be correlations between the shape and texture variations, a further PCA is applied to the data as follows.

For each example the concatenated vector is generated:

$$\mathbf{b} = \begin{pmatrix} \mathbf{W}_s \mathbf{b}_s \\ \mathbf{b}_g \end{pmatrix} = \begin{pmatrix} \mathbf{W}_s \mathbf{P}_s^T (\mathbf{x} - \bar{\mathbf{x}}) \\ \mathbf{P}_g^T (\mathbf{g} - \bar{\mathbf{g}}) \end{pmatrix} \quad (4.16)$$

where  $\mathbf{W}_s$  is a diagonal matrix of weights for each shape parameter, allowing for the difference in units between the shape and grey models.

Indeed, the elements of  $\mathbf{b}_s$  have units of distance, those of  $\mathbf{b}_g$  have units of intensity, so they cannot be compared directly. Because  $\mathbf{P}_g$  has orthogonal columns, varying  $\mathbf{b}_g$  by one unit moves  $\mathbf{g}$  by one unit. To make  $\mathbf{b}_s$  and  $\mathbf{b}_g$  commensurate, the effect of varying  $\mathbf{b}_s$  on the sample  $\mathbf{g}$  has to be estimated. To do this each element of  $\mathbf{b}_s$  is systematically displaced from its optimum value on each training example, and sample the image given the displaced shape. The RMS change in  $\mathbf{g}$  per unit change in shape parameter  $\mathbf{b}_g$  gives the weight  $\mathbf{W}_s$  to be applied to that parameter in equation 4.16.

A PCA is applied on these vectors, giving a further model:

$$\mathbf{b} = \mathbf{P}_c \mathbf{c} \quad (4.17)$$

where  $\mathbf{P}_c$  is the eigenvectors and  $\mathbf{c}$  is a vector of appearance parameters controlling both the shape and grey-levels of the model. Since the shape and grey-model parameters have zero mean,  $\mathbf{c}$  does too.

Note that the linear nature of the model allows us to express the shape and grey-levels directly as functions of  $\mathbf{c}$ :

$$\mathbf{x} = \bar{\mathbf{x}} + \mathbf{P}_s \mathbf{W}_s^{-1} \mathbf{P}_{cs} \mathbf{c} \quad , \quad \mathbf{g} = \bar{\mathbf{g}} + \mathbf{P}_g \mathbf{P}_{cg} \mathbf{c} \quad (4.18)$$

where:

$$\mathbf{P}_c = \begin{pmatrix} \mathbf{P}_{cs} \\ \mathbf{P}_{cg} \end{pmatrix} \quad (4.19)$$

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

Or more, to summarize, as:

$$\begin{aligned}x &= \bar{x} + Q_s c \\g &= \bar{g} + Q_g c\end{aligned}\tag{4.20}$$

where:

$$\begin{aligned}Q_s &= P_s W_s^{-1} P_{cs} \\Q_g &= P_g P_{cg}\end{aligned}\tag{4.21}$$

An example image can be synthesised for a given  $c$  by generating the shape-free grey-level image from the vector  $g$  and warping it using the control points described by  $x$ .

## 4.2 The training set

As seen previously, system's entries are grey-level pictures of front-view faces, showing one of the 6 basic emotions.

Then in order to train the model, a database of grey-level pictures showing front-view faces has to be chosen, sub-sampled (if necessary) and annotated precisely.

This section presents the Cohn-Kanade database (1999), a standard database for facial expression recognition.

### *Cohn-Kanade database pictures:*
















Subjects were 100 university students enrolled in introductory psychology classes. They ranged in age from 18 to 30 years. Sixty-five percents were female, 35% male, 85% European-American, 15% African-American or Asian.

The observation room was equipped with a chair for the subject and two Panasonic WV3230 cameras, each connected to a Panasonic S-VHS AG-7500 video recorder with a Horita synchronized time-code generator. One of the cameras was located directly in front of the subject, and the other was positioned 30 degrees to the right of the subject. Only image data from the frontal camera are concerned here.

Subjects were instructed by an experimenter to perform a series of 23 facial displays that included single action units (e.g., AU 12, or lip corners pulled obliquely, showing a smile) and combinations of action units (e.g., AU 1+2, or inner and outer brows raised). Subjects began and ended each display from a neutral face. Before performing each display, an experimenter described and modeled the desired display. Six of the displays were based on descriptions of prototypic emotions (i.e., joy, surprise, anger, fear, disgust, and sadness). These six tasks and mouth opening in the absence of other action units were coded by an expert who is certified in the use of FACS. Seventeen percent of the data were comparison coded by a second certified FACS coder. Inter-observer agreement was quantified with a coefficient characterizing the proportion of agreement above what would be expected to occur by chance. The mean coefficient for inter-observer agreement was 0.86.



Action units which are important to the communication of emotion and which occurs a minimum of 25 times in the image data base were selected for analysis. This frequency criterion ensured sufficient data for training and testing of Automated Face Analysis. When an action unit occurred in combination with other action units that may modify its appearance, the combination rather than the single action unit was the unit of analysis.

Upper Face		
AU4	AU1+4	AU1+2
		
Brows lowered and drawn together	Medial portion of the brows is raised and pulled together	Inner and outer portions of the brows are raised
AU5	AU6	AU7
		
Upper eyelids are raised	Cheeks are raised and eye opening is narrowed	Lower eyelids are raised
Lower Face		
AU25	AU26	AU27
		
Lips are relaxed and parted	Lips are relaxed and parted; mandible is lowered	Mouth is stretched open and the mandible pulled down
AU12	AU12+25	AU20+25
		
Lip corners are pulled obliquely	AU12 with mouth opening	Lips are parted and pulled back laterally
AU8+17	AU17+23+24	AU15+17
		
The infraorbital triangle and center of the upper lip are pulled upwards and the chin boss is raised (AU17)	AU17 and lips are tightened, narrowed, and pressed together	Lip corners are pulled down and chin is raised

**Figure 4.4: Facial displays studied for Automated Face Analysis**

Figure 4.4 shows the action units and action unit combinations which were selected. The action units we analyzed in three facial regions (brows, eyes, and mouth) are key components of emotion and other paralinguistic displays, and are common variables in emotions research. For instance, AU 4 is characteristic of negative emotion and mental effort, and AU 1+2 is a component of surprise. AU 6 differentiates felt smiles, also called Duchenne smiles (AU 6+12) from non-Duchenne smiles (AU 12) (Ekman et al., 1990). In

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

all three facial regions, the action units chosen are relatively difficult to discriminate because they involve subtle differences in appearance (e.g. brow narrowing due to AU 1+4 versus AU 4, eye narrowing due to AU 6 versus AU 7, three separate action unit combinations involving AU 17, and mouth widening due to AU 12 versus AU 20.).

Image sequences from neutral to target display (mean duration ~ 20 frames at 30 frames per second) were digitized automatically into 640 by 490 pixel arrays (PNG files) with 8-bit precision or grey scale values. Target displays represented a range of action unit intensities, including low, medium, and high intensity, above about 2000 pictures.

The Cohn-Kanade AU-Coded Facial Expression Database provides a valuable test-bed with which alternative approaches to automated recognition of facial expression and person identification may be tested. The database has been widely distributed for research in automated facial image analysis and serves as a test-bed and benchmark for algorithm development and testing.

The training set which was used in this project is a 400 pictures-sample of the Cohn-Kanade database, with an increased brightness applied on all pixels. It was done to make boundaries easier to see and thus facilitate the annotation process. Resulted pictures had been computed into 640x490 PNG files with 32-bit depth. Figure 4.5 shows one of those pictures.



Figure 4.5: one picture of the training set

### 4.3 Fitting the model to a new image

To interpret an image using a model, the set of parameters which best match the model to the image has to be found. This set of parameters defines the shape, position and appearance of the target face in an image, and is used for further processing, such as making measurements and classifying the object.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

There are several approaches which could be taken to match a model instance to an image, but all can be thought of as optimising a cost function. For a set of model parameters  $c$ , an instance of the model can be generated and projected into the image. This approximation can be compared with the target image, to get a fit function  $F(c)$ . The best set of parameters to interpret the object in the image is then the set which optimises this measure. For instance, if  $F(c)$  is an error measure, which tends to zero for a perfect match, the aim is to choose parameters  $c$  which minimise the error measure.

Thus, in theory all it has to be done is to choose a suitable fit function, and use a general purpose optimiser to find the minimum. The minimum is defined only by the choice of function, the model and the image, and is independent of which optimisation method is used to find it.

However, in practice, care must be taken to choose a function which can be optimised rapidly and robustly, and an optimisation method to match.

#### 4.3.1 Choice of fit function

Ideally, a fit function would represent the probability that the model parameters describe the target image object,  $P(c|I)$  (where  $I$  represents the image). Then the parameters are chosen to maximise this probability.

In the case of the appearance model described above, the varying parameters are the appearance model parameters,  $c$  and the pose parameters defining the position,  $(X_t, Y_t)$ , orientation,  $\theta$ , and scale,  $s$ , of the model in the image.

The quality of fit of an appearance model can be assessed by measuring the difference between the target image and a synthetic image generated from the model.

Given no initial knowledge of where the target face lies in an image, finding the parameters which optimise the fit is a difficult general optimisation problem.

If, however, an initial approximation to the correct solution is known, that is to say it is known roughly where the target face is in an image, due to prior processing, local optimisation techniques can be used.

Additionally, by taking advantage of the form of the fit function, it is possible to locate the optimum rapidly.

The actual Active Appearance Model searching algorithm is described in the following.

#### 4.3.2 Active Appearance Model

Here is described an algorithm which allows us to find the parameters which generates a synthetic image as close as possible to a particular target image, assuming a reasonable starting approximation (typically the mean appearance model).

##### *Overview of AAM Search*

The aim is to treat interpretation as an optimisation problem in which the difference between a new image and one synthesised by the appearance model is minimized.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

A difference vector  $\delta I$  can be defined:

$$\delta I = I_i - I_m \quad (4.21)$$

where  $I_i$  is the vector of grey-level values in the image, and  $I_m$ , is the vector of grey-level values for the current model parameters.

To locate the best match between model and image, minimising the magnitude of the difference vector  $\Delta = |\delta I|^2$  has to be done by varying the model parameters  $c$ . Since the appearance models can have many parameters, this appears at first to be a difficult high-dimensional optimisation problem. However, each attempt to match the model to a new image is actually a similar optimisation problem. A possibility is to learn something about how to solve this class of problems in advance. By providing a-priori knowledge of how to adjust the model parameters during image search, an efficient run-time algorithm is reached. In particular, the spatial pattern in  $\delta I$  encodes information about how the model parameters should be changed in order to achieve a better fit. In adopting this approach there are two parts to the problem: learning the relationship between  $\delta I$  and the error in the model parameters,  $\delta c$  and using this knowledge in an iterative algorithm for minimising  $\Delta$ .

### *Learning to Correct Model Parameters*

The appearance model has parameters  $c$ , controlling the shape and texture in the model frame according to equation 4.20.

A shape in the image frame,  $X$ , can be generated by applying a suitable transformation to a set of points  $x$ :  $X = S_t(x)$ . Typically  $S_t$  will be a similarity transformation described by a scaling,  $s$ , an in-plane rotation,  $\theta$ , and a translation  $(t_x, t_y)$ . For linearity the scaling and rotation are represented as  $(s_x, s_y)$  where  $s_x = (s \cdot \cos \theta - 1)$ ,  $s_y = s \sin \theta$ . The pose parameter vector  $t = (s_x, s_y, t_x, t_y)^T$  is then zero for the identity transformation and  $S_{t+\delta t}(x) \approx S_t(S_{\delta t}(x))$ .

The texture in the image frame is generated by applying a scaling and offset to the intensities,  $g_{im} = T_u(g) = (u_1 + 1) g_{im} + u_2 \cdot 1$ , where  $u$  is the vector of transformation parameters, defined so that  $u = 0$  is the identity transformation and  $T_{u+\delta u}(g) \approx T_u(T_{\delta u}(g))$ .

The appearance model parameters  $c$  and shape transformation parameters  $t$  define the position of the model points in the image frame,  $X$ , which gives the shape of the image patch to be represented by the model. During matching the pixels in this region of the image are sampled and gives  $g_{im}$ , and are projected into the texture model frame,  $g_s = T^{-1}(g_{im})$ . The current model texture is given by  $g_m = \bar{g} + Q_g c$ . The current difference between model and image (measured in the normalized texture frame) is thus:

$$r(p) = g_s - g_m \quad (4.22)$$

where  $p$  are the parameters of the model,  $p^T = (c^T | t^T | u^T)$ .

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

A simple scalar measure of difference is the sum of squares of elements of  $r$ ,  $E(p) = r^T r$ . A first order Taylor expansion of equation 4.22 gives:

$$r(p + \delta p) = r(p) + \frac{\partial r}{\partial p} \delta p \quad (4.23)$$

where the  $ij^{th}$  element of matrix  $\frac{\partial r}{\partial p}$  is  $\frac{dr_i}{dp_j}$ .

Suppose during matching our current residual is  $r$ . The goal is to choose  $\delta p$  so as to minimize  $|r(p + \delta p)|^2$ .

The RMS solution is obtained by equating 4.23 to zero:

$$\delta p = -Rr(p) \quad \text{where} \quad R = \left( \frac{\partial r}{\partial p}^T \frac{\partial r}{\partial p} \right)^{-1} \frac{\partial r}{\partial p}^T \quad (4.24)$$

In a standard optimization scheme it would be necessary to recalculate  $\frac{\partial r}{\partial p}$  at every step, which would be an expensive operation. However, by assuming that since it is being computed in a normalized reference frame, it can be considered approximately fixed. It

can be thus estimated once from our training set.  $\frac{\partial r}{\partial p}$  is estimated by numeric differentiation, systematically displacing each parameter from the known optimal value on typical images and computing an average over the training set.

Residuals at displacements of differing magnitudes are measured (typically up to 0.5 standard deviations of each parameter) and combined with a Gaussian kernel to smooth them:

$$\frac{dr_i}{dp_j} = \sum_k w(\delta c_{jk}) (r_i(p + \delta c_{jk}) - r_i(p)) \quad (4.25)$$

where  $w(x)$  is a suitably normalized Gaussian weighting function.

We then pre-compute  $R$  and use it in all subsequent searches with the model. Images

used in the calculation of  $\frac{\partial r}{\partial p}$  can be examples from the training set.

The best range of values of  $\delta c$ ,  $\delta t$  and  $\delta u$  to use during training is determined experimentally. Ideally it means modelling a relationship that holds over as large a range error  $\delta g$  as possible. However, the real relationship is found to be linear only over a limited range of values.

Experiments on the face model suggest that the optimum perturbation was around 0.5 standard deviations (over the training set) for each model parameter, about 10% in scale, the equivalent of 3 pixels translation and about 10% in texture scaling.

### *Iterative Model Refinement*

Given a method for predicting the correction which needs to be made in the model parameters, an iterative method can be built for solving our optimisation problem.

Given the current estimate of model parameters,  $c_o$  (initialized at 0) and the normalised image sample at the current estimate,  $g_s$ , one step of the iterative procedure is as follows:

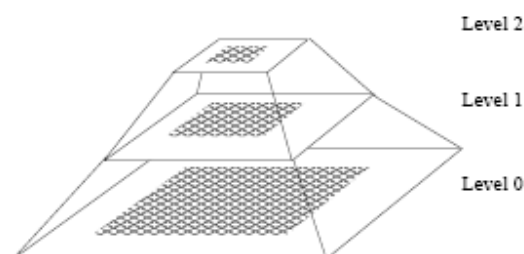
- Evaluate the error vector  $\delta g_o = g_s - g_m$
- Evaluate the current error  $E_o = |\delta g_o|^2$
- Compute the predicted displacement,  $\delta c = R\delta g_o$
- Set  $k = 1$
- Let  $c_1 = c_o - k\delta c$
- Sample the image at this new prediction, and calculate a new error vector,  $\delta g$
- If  $|\delta g_1|^2 < E_o$  then accept the new estimate  $c_1$  as the new  $c_o$
- Otherwise try at  $k = 1.5, k = 0.5, k = 0.25$  etc.

This procedure is repeated until no improvement is made to the error  $|\delta g|^2$ , and convergence is declared.

A multi-resolution implementation is used, in which is applied the above algorithm at each level, until convergence is reached, before projecting the current solution to the next level of the model. This is more efficient and can converge to the correct solution from further away than search at a single resolution.

This involves first searching for the object in a coarse image, then refining the location in a series of finer resolution images. This leads to a faster algorithm, and one which is less likely to get stuck on the wrong image structure.

For each training and test image, a Gaussian image pyramid is built. The base image (level 0) is the original image. The next image (level 1) is formed by smoothing the original then sub-sampling to obtain an image with half the number of pixels in each dimension. Subsequent levels are formed by further smoothing and sub-sampling (Figure 4.6).



**Figure 4.6: A Gaussian image pyramid is formed by repeated smoothing and sub-sampling**

## 4.4 Features extraction

Once points' coordinates are computed, expressions cannot be recognized only with those. Thus, it is needed to use some of those points to compute useful distances and angles which will provide measures in order to determine which expression is displayed. The features must be precise enough to cover every possible case. Typically, they will describe Action Units. Figure 4.7 shows some of them.

Feature	Numerical	Graphical
<i>eb_height</i> – Height of the eyebrows	$(a17.y + a19.y + a21.y) / 3.0$	
<i>eb_frowned</i> – The extent to which the eyebrows are frowned	$(a19.y - a21.y) + (a17.y - a21.y)$	
<i>e_openness</i> – Openness of the eyes	$a7.y - a5.y$	
<i>e_slanting</i> – Slanting of the eyes	$a3.y - a1.y$	
<i>m_openness</i> – Openness of the mouth	$(-(a25.y - a26.y) - (a27.y - a29.y)) / 2.0$	
<i>m_mos</i> – Measure of smile	$a23.y - (a26.y + ((a25.y - a26.y) / 2.0))$	

Figure 4.7: Some good discriminative facial features

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## 4.5 Recognizing facial expressions

Although this part of the project weren't implemented in time, there were some previously developed ideas that could have been discussed about before choosing one. This section gives some further details about those ideas quickly seen in section 3.2.3.

### 4.5.1 Template-based classification

In a template-based system, the template can be a pixel image or a feature vector obtained after processing the face images as a whole. In general, template-based techniques have limited recognition capabilities, which may be caused by the smoothing of some important individual facial details, by small misalignment of the faces, and also by large inter-personal expression differences. But they are quite simple, and have biological validity.

There are several methods of forming a template or using it.

Feature vectors of training samples with same labeled expressions can be simply averaged and the average results are used as templates. But templates can also be generated by successive training processes: the training set is divided into new training sets, each one standing for one label, and thus being composed of pictures showing the same expression for different people and/or different emotion configuration.

For instance, in the AAMs, each of the 6 expressions could be represented by a normalized shape-template, represented by a computed vector of facial characteristic points.

In addition, classification itself - and as a consequence for comparison between a new shape/appearance and templates - can be done in different ways. You can either compute an error measurement between the template's shape and the new shape (like Mean-Squared Error or MSE), or compute new features from templates (like distances and angles between precise characteristic points) then define domains or thresholds from data given by templates and finally check if the shape studied match some or all constraints given by each template. With latter approach, it is even possible not to give one and only one precise emotion label, by giving the user results with percentages. For instance, one face can show a happy smile with anger eyebrows (sadistic emotion for example), thus it would be a mistake to label it as only one of those two expressions. Then result can be like (50% happiness, 50 % anger). With a strict classifier, made to put new faces in only 6 categories, this face would have certainly been labeled as "unclassifiable", or less plausible but worse, misclassified.

Practically, templates were revealed not to be efficient enough in facial emotion classification (and precise classifications in general), but has the advantage of discriminating main expression features. As a consequence, template-based classification can be used as a coarse classification, reducing the number of dimensions of the problem from 6 (here) to 2. Then a fine classification is used in order to choose



between the 2 classes remaining, done by a simple k-nearest neighbor classifier for example.

#### 4.5.2 Artificial Neural Networks

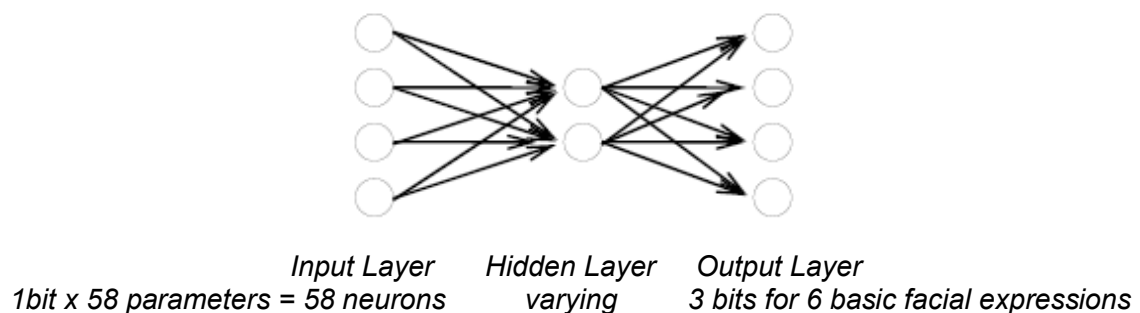
The ANN represents learning mechanisms that are inspired from the real world. The structure of such a mathematical abstraction would consist in a set of neurons presenting a certain type of organization and specific neuronal interconnections.

The Back Propagation approach of the ANN implies that the learning process takes place on the base of having learning samples for input and output patterns. In the case of learning such a system to model the mechanism of classifying facial expressions, it is required to have a set of input and output sample data.

There would be two stages; first, the training stage would make the system aware of the structure and associations of the data, then the second step would be testing. In the training step, the system would build the internal knowledge, based on the presented patterns to be learned. The knowledge of the system resides in the weights associated to the connections between the neurons.

The training of the ANN is done by presenting the network with the configuration of the input parameters and the output index of facial expression. Both kinds of data are encoded as values in the network neurons. The input parameters are encoded on the neurons grouped in the input layer of the network. In the same way, the emotion index is encoded in the neuron(s) in the output layer.

Figure 4.8 shows a three-neuron-layers-topology network. The input layer is set to handle the input data according to the type of each experiment. The data refer to the parameters of the model used for analysis.



**Figure 4.8: Structure example of an ANN**

Building knowledge in ANN is based on what follows: when the system learns a new association in the input/output space, a measure (typically the squared error) is used to give the degree of the improvement done or which still has to be done for the ANN to recognize all the samples without any mistake.

ANN is capable of storing many more patterns than the input number of dimensions and is able to acquire complex nonlinear mappings. However, it requires extremely long-time

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

training, an offline encoding, and is unable to know how to precisely generate any arbitrary mapping procedure.

#### 4.5.3 Bayesian Belief Networks

Bayesian networks are knowledge representation formalisms for reasoning under uncertainty. A Bayesian network is mathematically described as a graphical representation of the joint probability distribution for a set of discrete variables. Each network is a direct acyclic graph encoding assumptions of conditional independence. The nodes are stochastic variables and arcs are dependency between nodes.

For each variable there exists a set of values related to the conditional probability of the parameter, given its parents. The joint probability distribution of all variables is then the product of all attached conditional probabilities.

Bayesian networks are statistical techniques, which provide explanation about the inferences and influences among features and classes of a given problem.

Every expression is analyzed for determining the connections and the nature of different causal parameters. The graphical representation makes Bayesian networks a flexible tool for constructing recognition models of causal impact between events. Also, specification of probabilities is focused to very small parts of the model (a variable and its parents).

A particular use of BBN is for handling models that have causal impact of a random nature. In the context of the current project, networks have been developed to handle the changes of the human face by taking into account local and temporal behavior of associated parameters.

Having constructed the model, it is used to compute effects of information as well as interventions. That is to say that the state of some variables is fixed, and the posterior probability distributions for the remaining variables were computed.

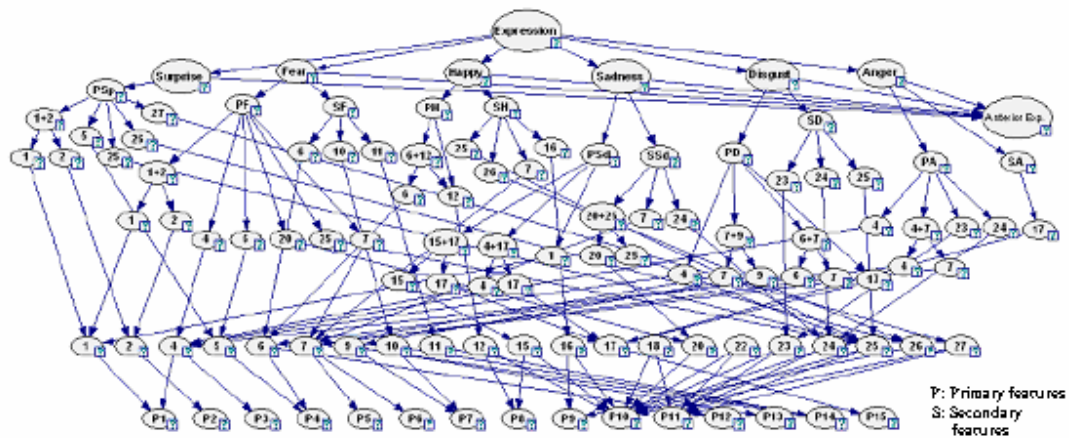
By using software of Bayesian network models construction, different Bayesian network classifier models can be generated, using the extracted given features used as the input to Bayesian network in order to verify their behavior and probabilistic influences. Some tests are performed in order to build the classifier then.

Bayesian networks were designed to explicitly encode “deep knowledge” rather than heuristics, to simplify knowledge acquisition, provide a firmer theoretical ground and foster reusability.

The idea of Bayesian networks is to build a network of causes and effects. Each event, generally speaking, can be certain or uncertain. When there is a new piece of evidence, this is transmitted to the whole network and all the beliefs are updated. The research activity in this field consists of the most efficient way of doing the calculation, using Bayesian inference, graph theory, and numerical approximations.

The BBN mechanisms are close to the natural way of human reasoning, the initial beliefs can be those of experts, avoiding the long training needed to set up neural networks for example, then learning is done by experience as soon as evidence starts to be received.

The expression recognition is done by computing the anterior probabilities for the parameters in the BBN (Figure 4.9).



**Figure 4.9: An AU-based facial expression recognition BBN**

The procedure starts by setting the probabilities of the parameters on the lowest level according to the values computed at the preprocessing stage. For each parameter, evidence is given for both static and dynamic parameters. Moreover, the evidence is set also for the parameter related to the probability of the anterior facial expression. It contains 6 states, one for each major class of expressions. The aim of the presence of the anterior expression node and that associated with the dynamic component of one given low-level parameter is to augment the inference process with temporal constraints. The structure of the network integrates parametric layers having different functional tasks. The goal of the layer containing the first features (here AUs) set and that of the low-level parameters is to detect the presence of some features in the current frame. The dependency of the parameters on AUs is determined on the criteria of influence observed on the initial database. The presence of one AU at this stage does not imply the existence of one facial expression or another. Instead, the goal of the next layer containing the AU nodes and associated dependencies is to determine the probability that one feature presents influence on a given kind of emotion. The final parametric layer consists of nodes for every emotional class. More than that, there is also one node for the current expression and another one for the one previously detected. The top node in the network represents the current expression. It has two states according to the presence and absence of any expression and stands for the final result of analysis. The absence of any expression is seen as a neutral display of the person's face on the current frame. While performing recognition, the BBN probabilities are updated in a bottom-up manner. As soon as the inference is finished and expressions are detected, the system reads the existence probabilities of all the dependent expression nodes. The most probable expression is then given by the larger value over the expression probability set.

BBN is capable of discovering causal relationships and has probabilistic semantics for fitting the stochastic nature of both the biological processes and noisy experimentation. However, it cannot deal with continuous data and have to be partly changed in order to deal with temporal expression data.



## 5. System implementation

### 5.1 Preprocessing the data set

#### 5.1.1 Annotating pictures

The training process was done with an independent application, implemented in Matlab. Seven contours were defined (see Figure 5.1):

- The face contour defines the global contour of the face, starting from the left, slightly higher than the eye-brows, going down to the chin and finishing on the other side of the face, at the same level than the first point.
- The nose contour starts from the upper-left of the nose, slightly above the eyes, and defines the nose in the whole, not taking account of nasal holes for example.
- The two eye contours both start from the inner corner of the eye, the closest to the nose, run along the upper part of the eye, reach the opposite corner, and come back to the beginning running through the bottom part. Note that those contours are not closed; the last point is not linked to the first one.
- The two eyebrows contours are defined in the same way, starting from the inner corner, reaching the opposite side by the upper part and getting back by the bottom part. Those are not closed contours either.
- Last but not least, the mouth contour starts from the inner left corner of lips, runs along the inner bottom lips, reaches the opposite corner and comes back to the first point running through the inner upper lips. Here again, this contour is not closed.

As we can see, the contours of the data pictures are not defined as shown on Figure 3.4. Instead of using two contours for both the eyes and mouth, it had been decided to take only the external contour for the eyes, and the internal contour (of lips) for the mouth. As a matter of fact, modeling the eyes circle around the iris will not bring too much additional information, as it is always keeping a round form and it often stands in the middle of the eye. It is the same for the external lips contour, which looks globally the same than the internal contour, bringing very few useful features for the future model. Additionally, the implementation has to be real-time oriented as possible as it is, and the more contours there are, the longer it will for the model to be trained or used. Figure 5.1 shows the application used for annotating data, with a picture example where contours are shown.

Its functioning is quite simple. The left frame shows all PNG files from a directory, whose complete path has been given on the top-left field by the user. Then it's possible to display each folder's picture on the main frame by left-clicking on its name. Once the picture is loaded, it's time to put contour points.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

The top-right corner presents a radio button field, enabling the successive definition of the 7 separate contours. The practical process of putting points for every 400 pictures was not completely planned, simply because at this time of the project, the model was not chosen yet. So point count change randomly between two pictures and are not equidistant at all. However, T-junctions (eyes, eyebrows and mouth corners for instance) were always marked with carefulness and precision, as those kind of points are always determinant whatever which model is chosen. As for the choice of pictures to be annotated, it was done in order to maximize variations between each two images. For instance, when two pictures of the same person display the same emotion, without significant visible changes in the way it's displayed, only one is kept.

Speaking of carefulness and precision, as Figure 5.1 proves it, global brightness of one single picture is not enough to see every points being set at the same time, even if it had been optimized in advance. This comes from some contrast differences, depending on which part of the picture is covered - shadows around the eyes and underneath the nose and the chin can be a real bother - and on which color is the contour, at a lower level. As a consequence, 2 action buttons had been added in order to increase or decrease brightness. Finally, last buttons are basic methods to be applied to points, like deleting current contour to do it over again, loading and saving every points of the picture (not only of one contour). Actually, loading a new picture automatically saves current picture points, and loads new picture points (if already existing).

In the end, practically putting points on a picture equaled to successive left-clicks on face boundaries, designing contours by running along boundaries with respect of contour path definitions seen above (points links are displayed automatically by the application). In order to be accurate, brightness has to be almost constantly adjusted during the process.

Another little but time-consuming problem is that contour boundaries are sometimes difficult to distinct precisely, especially for face contour around hairs and upper region of nose contour, and not because of brightness but simply because it is a front-view picture. Consequently, those parts of contours have to be "guessed" with trying to keep somehow a "logical" global structure for each face taken separately of course, but also to keep a coherent contour structure among the whole training set.

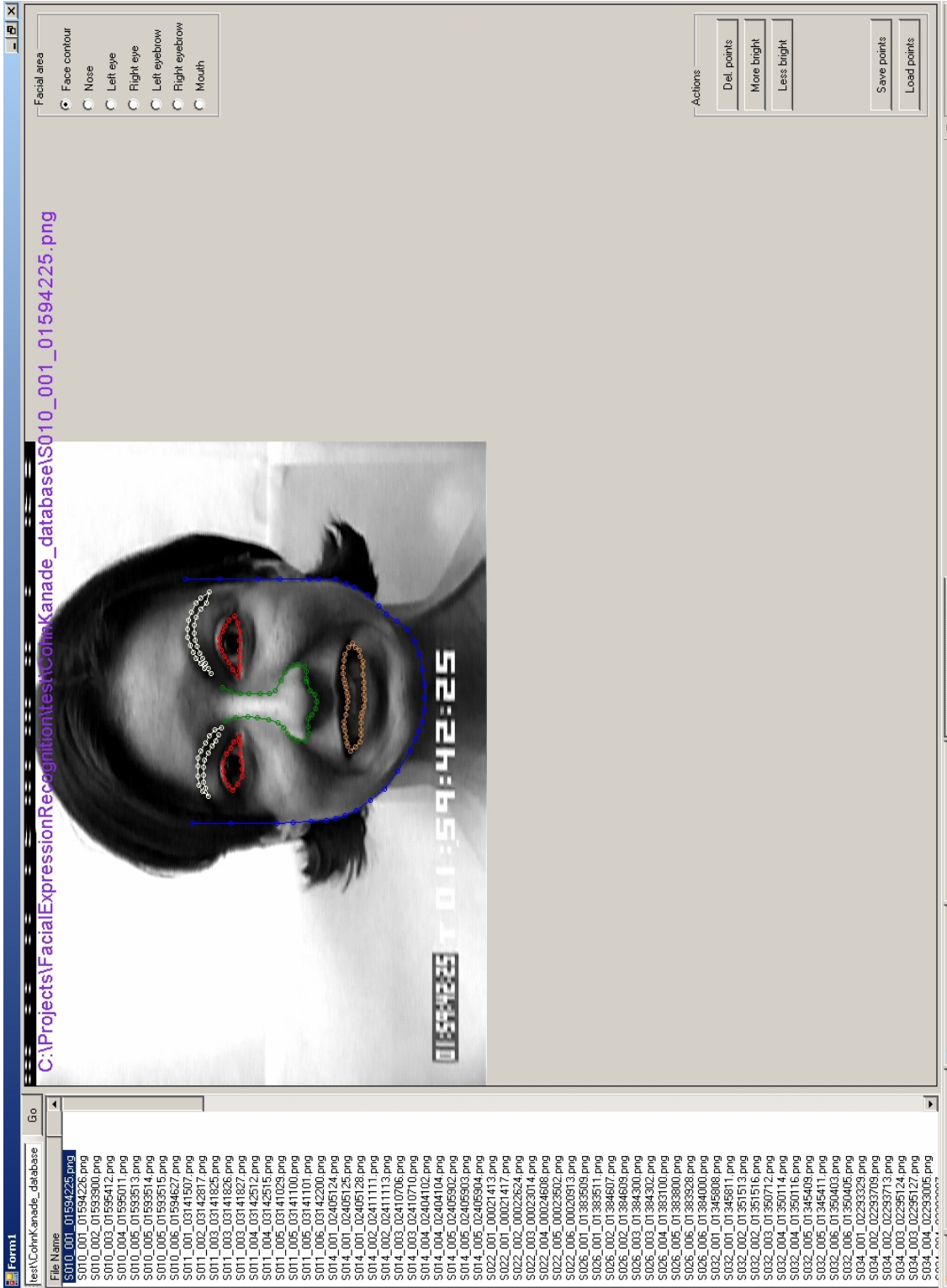


Figure 5.1: Pointer application window with example

### 5.1.2 AAM files

When saved, points' coordinates are written in a file with the AAM extension. For example, for an image called S001.PNG, once points put and saved, a text file with the name S001.PNG.AAM is created with every point coordinates written in it, grouped by contour. Plus, the quantity of points for each contour is added at the beginning of the contour's list of points. An example of an AAM file is shown at Appendix B.

## 5.2 AAM-API: generating the model

### 5.2.1 Introduction

The AAM-API is a C++ implementation of the Active Appearance Model framework. It can be used as a traditional API by linking in an AAM library, or it can be used as a precompiled command line program, e.g. as a part of an image analysis course. The complete package includes the AAMLab, various Matlab scripts for shape annotation and communication with the API, source code documentation and project files.

The AAMLab is a windows program that presents a GUI front-end to some of the functionality offered by the API. This includes running model searches, real-time visualisation of the modes of variation and training set annotation.

The software runs on the Windows platform and is partly based on the following software libraries:

- MS VisionSDK - image, vector and matrix handling etc.
- LAPACK - matrix handling, eigen and singular value decompositions etc.

Microsoft Visual C++ is required to compile or modify the source code.

#### *File Formats*

- **Image i/o:** BMP (8-bits BMP for grey-scale AAM)
- **Shape i/o:** ASF - AAM Shape File Format (see subsection 5.3.1)
- **Model I/O:** AMF - AAM Model File Format, Partly binary and partly ASCII format. (see Chapter 6 for ASCII results part)

A class graph of AAM-API can be found in Appendix E.



ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

### 5.2.2 Model generation

The model generation process was done with AAM-API's command-line *aamc*. This method makes mainly use of CAAMBuilder and CAAMModel class.

Parameters are a folder path with an image database in 8-bits BMP format, with corresponding annotations in ASF format, and a configuration file (text file) allowing tweaking the generation process. Such a configuration file can be seen in Appendix D. Additionally, all ASF files have to comport the same total number of points, or else *aamc* command-line will throw an error.

As a result, the program writes a binary AMF file, named by the user, and two text files, one characterizing the model, and one listing the pose parameters (scale, rotation and translations) for every picture of the database – of the folder parameter – and giving some statistics (mean and standard, minimal and maximal deviations).

### 5.2.3 ASF files

ASF files are text files, just as AAM files; though it's slightly more complicated, meaning they are more precisely defined. An example of such a file is shown at Appendix C.

An ASF files always starts with some comments, titling the file, and stating the creation time of the file. Comments lines start with the '#' character. Then, the number of points is written, providing an easy way to check if all ASF files present the same number of points. Indeed, the AAM-API model generation requires that all contours are defined with the same number of points, this number being the quantity of points of the future generated model.

Then all points are listed, without being grouped by contour the way as AAM files do, but displaying more information than just the two coordinates. Actually, ASF files comport seven fields useful to us:

- *Path number (integer)*: the contour number which point belongs to
- *Type (integer)*: the way the point will be displayed by the AAM-API
- *X rel* and *Y rel (double)*: the relative point coordinates; instead of stocking points' coordinates in an absolute format - that is to say the pixel coordinates, which are integers, coordinates origin being the top-left corner of the picture - points are computed into an image-size relative format. It means that those are the pixel coordinates, divided by image attributes - width for x coordinates and height for y coordinates – and thus it brings points' coordinates between 0 and 1. Since those coordinates are computed into a double format, this allows more precise calculations when generating the model
- *Point number (integer)*: starts from 0
- *Connects from* and *Connects to (integer)*: the point numbers linked to the current point on the contour, defining where the two contour's segments linked to the current point go to. If the contour is not closed, contour's first and last points are linked to themselves.

Finally, the picture's name associated to the ASF file is written at the end of the file - in the field *host image*.

## 5.3 Database conversion

In order to generate the model, and because the annotation process was done separately, before finding the AAM-API, training set's pictures and annotations text files must be converted into the right format. Several steps have to be completed: first, AAM files have to be changed to equalize the total number of points and the quantity of points defining each contour. Then those files have to be converted into ASF files, that is to say "translated" into the text file defined by the AAM-API. Finally, the 32-bits PNG pictures are converted into 8-bits BMP pictures.

### 5.3.1 ModelConvertor classes

First two classes are basic interfaces enabling to easily manipulate shape points in both AAM and ASF formats. Thus both classes' fields just correspond to the fields defined in those text files and seen in previous sections.

Since those classes will be especially used in vectors, one additional copy constructor has been defined for each of them. Indeed, copy constructors are always required to manipulate vectors, since this is the constructor which is called when simply adding a new entry – with the *push\_back* instruction for instance.

AAMPoint's and ASFPoint's fields and constructors are made public, and then it can be called by the main classes.

```
class AAMPoint {
public:
    int x;
    int y;

    AAMPoint(){ x=0 ; y=0; }
    AAMPoint(int x_p, int y_p) { x=x_p; y=y_p; }
    AAMPoint(const AAMPoint &toCopy){ x = toCopy.x; y = toCopy.y;}
};
```

```
class ASFPoint {
public:
    unsigned int PathID;
    unsigned int TypeFlag;
    double x;
    double y;
    unsigned short int ID;
    unsigned short int ConnectFrom;
    unsigned short int ConnectTo;

    ASFPoint(int path, int flag, double x_rel, double y_rel,
             int id, int cf, int ct)
        {PathID=path; TypeFlag=flag; x=x_rel; y=y_rel;
         ID=id; ConnectFrom=cf; ConnectTo=ct;}
    ASFPoint(const ASFPoint &toCopy)
        {PathID=toCopy.PathID; TypeFlag=toCopy.TypeFlag;
         x=toCopy.x; y=toCopy.y; ID=toCopy.ID;
         ConnectFrom=toCopy.ConnectFrom;
         ConnectTo=toCopy.ConnectTo;}
};
```

The two main classes give the methods for converting the database.

```
class AAMConvertor {
private:
    CString filename;
public:
    AAMConvertor(const CString &fn) { filename = fn; }
    virtual ~AAMConvertor(){}
    AAMConvertor(const AAMConvertor &aamC)
        { filename = aamC.filename; }

    const CString GetFName() const { return filename; }
    void SetFName(const CString &fn) { filename = fn; }

    bool AAM2ASF(int img_width, int img_height, int numP,
                int numFC, int numN, int numE, int numEB,
                int numPM);
private:
    AAMPoint FindNextPoint(AAMPoint p0, AAMPoint p1,
                          double currentD);
    bool AAMResize(int numberP, int numberPFC, int numberPN,
                  int numberPE, int numberPEB, int numberPM);
};
```

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

This class enables to manipulate one AAM file – which path is given by the field *filename* – and to generate the corresponding ASF file.

Since this class is then used in a vector, once again a public copy constructor is defined. The *filename* field being made private, one public accessor and modifier for this field have been defined. The main method *AAM2ASF* enabling to convert an AAM file into an ASF file is made public, in order to be used by the *ModelConvertor* class. This method uses private method *AAMResize*, which uses itself private method *FindNextPoint*.

*AAM2ASF* needs all the global model conversion parameters for the conversion process, and outputs a Boolean for errors handling, so as *AAMResize*, although it does not need image sizes. *AAMResize* set AAM files to the right size (right number of points), and use *FindNextPoint* to get the *AAMPoint* at distance *currentD* on a segment defined by two *AAMPoints*, starting with the first one.

The final class uses the later class to convert a whole directory with AAM files to corresponding ASF files. This class is associated to an *AAMConvertor* vector, with additional fields and methods.

```
class ModelConvertor : public std::vector<AAMConvertor>{
private:
    int img_width;
    int img_height;

    int nbP;
    int nbPFC;
    int nbPN;
    int nbPE;
    int nbPEB;
    int nbPM;

    int nbfiles;
public:
    ModelConvertor() { nbfiles=0; }
    virtual ~ModelConvertor(){}

    bool InitializeData(const CString &path, const CString
                        &model_param);
    bool ConvertData2ASF(const CString &path, const CString
                        &model_param);
    bool ConvertPNG2BMP(const CString &path);
};
```

This is where model conversion parameters are loaded, and computed in the eight first private fields. The two first parameters are images attributes, which have been considered to be the same through the whole training set. The six other ones are the predefined quantity of points, in a whole for *nbP*, and for each contour for the five others. Those parameters are loaded from a text file in the *InitializeData* method, which additionally initialize the public vector of *AAMConvertor*, with the names of the AAM files inside the folder determined by the parameter *path*. It also computes the number of files

added to the vector into *nbfiles*. ConvertData2ASF calls this method, before applying one by one AAM2ASF with all AAMConvertors inside the vector, using ModelConvertor fields loaded with *InitializeData*.

As for the ConvertPNG2BMP method, it uses CXImage library to simply generate 8-bits BMP files from all 32-bits PNG files of the folder *path*.

### 5.3.2 Normalizing points (AAMConvertor::AAMResize)

This method takes in parameter all the number of points for each contour, which are loaded from the *model\_parameters* text file with the ModelConvertor class.

This method reads a .PNG.AAM file, and creates an .AAM file of the same name, that is to say a new AAM file well prepared for the further training process.

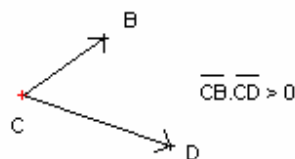
When reading through the file, every time '#' is found means that a list of points for a new contour has been reached. Depending of the contour's ID read after the '#', the number of points NP defined for the contour is initialized, accordingly to the parameters. Then the whole list of points is read, and stocked in an AAMPoint vector.

Obviously, the first and the last contour points will be kept.

At first, it was decided to use those points and especially the contour it defines to keep T-junctions, and to discard some other points until the right quantity is reached.

This is of course in the case points are more numerous than it should be in the end. Else some points are just equally added between existing points until the right number is reached.

Thus for the tough case, all points are studied in order, except for the first one and the last one. Each other point is then studied by using the two segments of which it is an extremity, as shown on Figure 5.2.



**Figure 5.2: detecting a T-junction**

With this configuration, deciding if point C is a T-junction or not can simply be defined by determining if the scalar between vectors *CB* and *CD* is positive or equals to zero. But the angle is actually not enough for deciding if a point is a T-junction or not. As a consequence, it was decided to use another prediction measure: detecting a T-junction was decided to be detecting a change in direction in x or y.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

Consequently, points are studied 3 by 3, and evolution of coordinates on those points is checked to see if there is a T-junction. For example, with Figure 5.2, y-coordinates increase from B to C and from C to D; but obviously C's x-coordinate is lower than B's whereas D's x-coordinate is higher than C's. A change of direction in x-coordinates is detected, and the T-junction is declared. Lots of cases have to be considered then, and a point – the one on the middle, C on Figure 5.2 – is not considered a T-junction if, for the 3 points:

- x-coordinates and y-coordinates increase or decrease both
- x-coordinates increase and y-coordinates decrease
- x-coordinates decrease and y-coordinates increase
- x-coordinates stay still and y-coordinates increase or decrease
- y-coordinates stay still and x-coordinates increase or decrease

Note that increasing/decreasing can mean “staying still for one of the segment and increasing/decreasing for the other one”.

Once a point is detected not to be a T-junction, it has to be deleted somehow. It is not as easy as it seems to be, since simply deleting a point make losing much contour information. Indeed, given 3 points, simply deleting the middle one and linking the extreme ones is really not a thing to do, reducing the contour precision and thus the shape approximation for the future model. In order to delete this point without losing too much precision on the contour, the following algorithm is applied:

*Let A,B,C,D,E,F,G be 7 consecutive points on a contour, where C is the current point, so neither B is the first contour point nor D is the last contour point.*

- *if  $BC \geq 2$ , then displace B on [BC] such as  $BC = |BC/2|$ , and discard C  
take F as the next point to be studied*
- *else if  $CD \geq 2$ , then displace D on [CD] such as  $CD = |CD/2|$ , and discard C  
take G as the next point to be studied*
- *else just discard C, and take D as the next point.*

Of course, if B is the first point, it won't be displaced, same thing for D if it is the last point. And it is also possible that E or F doesn't exist, in this case there is no next point, same thing if it should be D and it is the last point.

Testing if segment size is higher than two is done for checking that there will be a pixel between the two points enabling to displace B (or D) on it. Finding a point on a computer equals finding a pixel, which is why the absolute value is taken when displacing B or D.

The next point to be studied is chosen in order to avoid a bad approximation resulting from the combination of deleting two successive studied points. For instance, taking the “contour” defined by A,B,C,D,E,F,G,H, deleting C transforms it into A,B,D,E,F,G,H. If no points have been moved, this means B,C,D were close, then just deleting C do not change the contour much and B,D,E can be the next three points of the algorithm. But if B has been displaced, it should not be displaced again or else AB would become too large, plus D should not be changed either or else BD would form a combined approximation, less accurate than only one alone of course. Then E,F,G are the next

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

three points. Finally it is the same problem if D is displaced, neither D nor E should be displaced next iteration, then F,G,H are taken.

This algorithm is applied by iterating it on the whole contour, but since running through the contour only once can be not enough, the contour is read the number of times needed to reach the right contour quantity of points.

However, if this algorithm can be enough for generating the statistical shape model, it did not provide satisfactory results for the texture model, and a fortiori for the combined appearance model. This can be explained by the fact that points' locations are not necessarily the same from one picture to another. It depends on the annotation process, where changes in direction can happen at different locations. Plus, annotated contours don't have necessarily the same number of points from an image to another. As a consequence, nothing ensures that two corresponding points in two different pictures are efficiently related. Then another algorithm is applied to ensure this attribute.

The aim here is to put equally distant points on the contour, starting from the first point and ending on the last point. The total length L of the contour is computed by summing all contours' segment lengths. The distance between two consecutive points is then  $D = L/(NP-1)$ , where NP is the target number of points. The algorithm is quite simple, it is to run through the contour previously defined while putting equally distant points and deleting others. However, it is slightly more complicated when it has to be implemented because of the discreet computer's representation.

First, it cannot be done inside the vector, because the contour defined by the annotation process has to be kept as the contour to be considered, or else every time an old point is deleted, the contour changes and so as the distance D.

In addition, even if distances are computed in double, precision is not enough and pixels coordinates are still integers, so putting a point on a pixel at a certain distance from another pixel is an approximation operation. Then a threshold t has to be defined under which the right distance is said to be reached.

The algorithm iteration is then defined as:

*Let  $d=D$  stand for the distance which has to be done before putting an other point, and  $AB$  the length of the current segment  $[AB]$ , and  $C$  the next point on the contour*

- *if  $|AB-d| < t*AB$  then add  $B$  to the new contour list of points, and take  $[BC]$  as the next segment with  $d=D$*
- *else if  $AB < d$  then just take  $[BC]$  as the next segment with  $d=d-AB$*
- *else  $AB > d$  then find and add  $P$  to the new contour list of points, such as it belongs to  $[AB]$  and  $AP=d$ , then take  $[PB]$  as the new segment with  $d=D$*

The main operation is then to find P when  $AB > d$ , which is done in *FindNextPoint* method. The equation defining x-coordinate of any point P belonging to segment AB can be:  $x(P) = k*x(B) + (1-k)*x(A)$ , where k evolve between 0 and 1. It is of course similar for P's y-coordinate. Then when computing distances,  $AP=k*AB$ . Thus *FindNextPoint* algorithm is theoretically to compute  $AP=k*AB$  with all k's between 0 and 1 until  $AP=d$ . Practically of course, k is decremented starting from k=1 (or incremented with k=0), that

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

is to say that  $k$ 's accuracy is limited. Then  $k$  is initialized to  $k=1$ , and decremented by the threshold  $t$  at every step. Deciding if loop has to be stopped is then done by testing if  $k*AB \leq d$  – which means that  $k*AB \leq d \leq (k+t)*AB$ , which is equivalent to  $|k*AB-d| \leq t*AB$ . In the end, this ensures that  $AP \approx d$  while being smaller than  $d$ , thus ensuring that the right number of points will be added to the new contour list of points.

Of course, every time a point is added to the “output” list of points, the algorithm checks if the right number of points have been reached, since a priori distances between those points are a little smaller than  $D$ . This is obviously because of the algorithm, but also because pixels' coordinates are integers and not doubles. Though if the accuracy of threshold  $t$  is good enough, this test should not been verified and the entire contour studied. However, for the same reasons, it is 100% sure there will not lack points by examining the contour just once. It was decided to take  $t=0.01$  as threshold value, since it seemed to give enough satisfactory results.

Every time a contour – an AAMPoint vector – have ended to be studied, generating another AAMPoint vector defining a contour with the right properties, points' coordinates are written in the output AAM file, with the suitable structure of course. A mistake which shall not be done once points have been written on the AAM text file is forgetting to clear vectors, which avoids examining the same contour points all over again. In the end, when all contours have been examined, output is closed and is ready to be converted in ASF format. Note that if AAMConvertor *filename* was S001.PNG.AAM, it is updated to S001.AAM, since this will be the one to be converted.

### 5.3.3 Writing ASF files

(AAMConvertor::AAM2ASF)

(ModelConvertor::ConvertData2ASF)

Once contours have been normalized and written into an AAM file, it can be converted in an ASF file. It simply equals to read the AAM file contour by contour, every line coordinates being used to add an ASFPoint to a global vector. This vector will in the end be used to write the list of points into the ASF file, with the right structure, after having written the *number of model points* and before writing the *host image*.

As for ASFPoint fields, they are filled in this way:

- *PathID (Path number in ASF files)* is determined by the number read at the beginning of a contour in AAM file, minus 1 because in AAM it starts from 1 whereas in ASF it starts from 0
- *TypeFlag (Type in ASF)* is also determined by the same number, except that it does not change for both eyes on one hand, and both eyebrows on the other hand
- *x, y (X rel, Y rel)* are computed from AAM's coordinates by dividing x-coordinate by image width and y-coordinate by image height. Image attributes are given as parameters



ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

- *ID (Point number)* is simply computed with a incremental index inside the loop, starting from 0
- *ConnectFrom, ConnectTo (Connects from, Connects to)* are generally given by *ID-1* and *ID+1*, except for first and last points. For those points, it depends on if the current contour is closed or not. If the contour is closed or not is decided from *PathID* of course, since contours are always listed in the same order. If contour is not closed, first point's *ConnectFrom* field is given by current *ID*, so as the last point's *ConnectTo* field – meaning that first point and last point are linked to themselves. If contour is closed, first point's *ConnectFrom* field is given by the total number of points minus 1, and the last point's *ConnectTo* field is given by the total number of points minus the current contour number of points. Note that the total number of points correspond to the quantity of points read from AAM file, computed at the beginning of a new contour thanks to the current contour number of points given in AAM file after *PathID* number.

After the AAM file has been read and the ASFPoint vector is generated, a quick check about the total quantity of points read from the AAM file is done by comparing it with the number of model points given in parameter. Then the ASF file is actually written by using the ASFPoint vector.

One final mistake not to be done is to write the entire path for the *host image* at the end of ASF file. This “field” is just supposed to show the corresponding image's name, confirming that ASF annotations must be inside the same path than their corresponding BMP pictures for the model generation.

Finally, this method is applied to a whole path of .PNG.AAM files in *ConvertData2ASF* method by using an AAMConvertor vector. This vector is initialized in *InitializeData* thanks to a WIN32\_FIND\_DATA file descriptor and a file HANDLE enabling to look for .PNG.AAM files inside a given directory which path is given as a parameter. Then *AAM2ASF* method is called for every AAMConvertor of the vector (this method calling *AAMResize*). Additionally, once all ASF files have been generated, “temporary” AAM files (the ones which are normalized) are deleted.

#### 5.3.4 Converting pictures (*ModelConvertor::ConvertPNG2BMP*)

Similarly as in *ConvertData2ASF*, a folder which path is given as a parameter is read for searching for PNG files. Every time a PNG file is found, CXImage library is used to load it in a CXImage format. Then, it is simply saved into 8-bits BMP picture, still using CXImage library and structure.

## 5.4 Active Appearance Search

The AAM-API doesn't only provide methods for generating the model, but also for reading it of course, and more important to manipulate shapes. Consequently, starting from a new image, it provides the needed methods to initialize the model fit in order to

approximate the shape first, before optimizing it. Indeed, as seen in chapter 4, the Active Appearance Model needs a good initialization to be efficient. In this section, both initialization and optimization processes are detailed.

#### 5.4.1 Initialization method

Once the model and the picture have been correctly read from disk, the goal of searching is to find the model parameters (called  $c$  in chapter 4).

The system is meant to be integrated into a more global system where face detection will pre-process the pictures, thus giving a good initial approximation to the place where to put the shape.

So the initialization is rather simple. It requires getting the information by the face detection process, about the global position of the face – actually the coordinates of the center of the square defining face's position – and its global size – height and width of the same square.

The initial shape inserted is the reference shape, defined as the mean shape scaled to the mean size, with no rotation on it, and extracted from the model (CAAMModel *RefShape* method). Thanks to the information given by the face detection process, the reference shape can then be translated, making the center of gravity of the shape – actually the center of gravity of the points – to correspond to the center of the square; and scaled, using the face attributes (CAAMShape *COG*, *Translate* and *Scale* methods). This gives a first approximation of model parameters  $c$ .

#### 5.4.2 Optimization method

To optimize the model fit, first some constants have to be defined. As we can't be 100% sure the optimization process will converge, in case the initialization were not good enough, the optimization process will be iterated 30 times at worst. The convergence is declared when the error measure is under 0.01.

Then the initial error (and error vector) between the model and the image is computed, that is to say the pixel difference from a model instance (defined by  $c$ ) and an image. To calculate this error, first the model texture and shape model has to be generated, using  $c$  (CAAMModel *TextureInstance* and *ShapeInstance* methods). Then, the shape model is aligned on  $x$  coordinates, and texture for shape points is sampled from the image (CAAMShape *AlignTo* and CAAMModel *SampleShape* methods). Finally, the difference between the texture instance and the texture sample corresponding to shape points is computed, and its similarity measured, using the non-normalized  $L^2$  norm. Note that if the initial shape was define outside the image, the algorithm terminate here.

This process is repeated, updating pose and model parameters using the error vector and the coefficient  $k=1$ , computing the new error, and if it is inferior to the previous error, parameters are accepted as the new ones for next iteration. Of course, pose parameters are constrained in range, so they have to be checked, to be sure the model will be inside the image for example, or not to stand for a rotation of more than  $+180^\circ$  or less than  $-180^\circ$  (CAAMModel *ConstrainSearchParameters* method). Else if the new error is superior, the prediction is damped updating model parameters with  $k=0.5$ , later then  $k=0.25$ .

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

The (input/) output shape is the last one of the iteration process, even if it occurred 30 times. Thus, the final error, the number of iterations and the number of damping process needed are given as a result.

Once the shape is optimized, it can be easily written in ASF format with AAM-API methods.

## 5.5 Results

### 5.5.1 Generating the model

The methods described above were used to build the model of facial appearance. The training set is made of 400 images of faces, each labelled with 122 points around the main features.

From this, a shape model with 22 parameters were generated, a shape-free grey model with 122 parameters and a combined appearance model with only 58 parameters required to explain 95% of the observed variations. The model used about 44,000 texture samples to make up the face patch.

Model ASCII results can be found in Appendix F.

### 5.5.2 Active Appearance Model search

The optimization method was applied to the face model described above. After performing linear regression, a square error statistic ( $R^2$ ) is computed for each parameter perturbation  $\delta c$  to measure how well the displacement is “predicted” by the error vector  $\delta g$ . The average  $R^2$  value for the 58 parameters was 0.91, with a maximum of 0.98 and a minimum of 0.47.

To obtain a quantitative evaluation of the performance of the global algorithm, the model was tested on a different set of 48 labelled images.

On each test image, the model is systematically displaced from the true position by  $\pm 15$  pixels in x and y, and changed its scale by  $\pm 10\%$ . Then the AAM search is run, starting with the mean appearance model. About 7776 search configurations were run, 162 on each image, each taking an average of 3.88 seconds on a Intel Pentium 4 3 GHz with 512 Mo RAM. Of those 7776, 528 failed to converge to a satisfactory result (the mean point position error was greater than 7.2 pixels per point). Of those that did converge, the RMS error between the model centre and the target centre was (0.533; 0.56) pixels. The standard deviation of the model scale error was 12%. The mean magnitude of the final image error vector in the normalised frame relative to that of the best model fit given the marked points was 0.72 and the standard deviation was 0.11. Figures 5.3 and 5.4 show initialization and final optimization of AAM search.



Figure 5.3: AAM search initialization



Figure 5.4: Final AAM search optimization

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## 6 Conclusion

This paper showed a way to automatically extract contour points from a face. Several improvements can be done.

First and foremost, there was not enough time to complete the whole system of recognition. Then, features extraction and especially recognition process have to be implemented.

Secondly, AAM search takes an average time of nearly 4 seconds, which is too slow for a real-time oriented system. This is mainly because of the time it takes for loading the model. Then the model training set can be tweaked by defining less contour points or by decreasing its size. Another possibility is to reduce the variation percentage wanted to be explained by the model. Any case, it will reduce model's accuracy more or less, so those parameters have to be optimized such as time computation is less than 1 second while trying to keep satisfactory results in the mean time. However, some says that Active Appearance Model should not be used for real-time systems. My opinion is that if it is useless now, it could however become really interesting in couple years with better and better computation capacity.

This slowness is also the reason why multi-resolution implementation was not used, so it is also an improvement to take into account once the system is faster.

Applications for those kinds of systems are numerous. A facial expression dictionary, and then a non-verbal dictionary are the main project of KBS department. But of course other applications are obvious in video surveillance. Indeed, it could partly automate aggression detection from a video flux, instead of having an operator looking at dozen screens. If an aggressive face emotion or a scary face emotion is displayed by someone, then alert is given to the operator, which can decide what to do. Another viable application would be to listen to suspicious conversations. Thus video camera would have a long-range microphone and when detecting several suspicious, worried or badly determined persons' face emotions, it could eventually record their conversation by targeting them with the microphone.

Training into image analyzing and processing was very instructive and I'm now more skillful with speaking English and Visual C++.

My experience here accentuated my desire to work on an international context (even more on Netherlands which is so much a welcoming and understanding country), whereas I didn't feel really comfortable with research way of functioning.

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## Appendix

### A. Specifications

#### i. Softwares needed

- Windows XP
- Visual C++ with MS VisionSDK, CLAPACK and CXImage installed to compile sources

#### ii. System specifications

##### *Functional requirements*

- Fully automated way to efficiently locate faces and characterize expressions (with landmark points)
- Recognize a facial expression from that information among 6 basic emotions: fear, anger, disgust, happiness, surprise and sadness, plus neutral eventually

##### *Non-functional requirements*

- Running demonstrator of the system (displaying contours, showing search...)
- Easy to integrate

#### iii. Anticipated Schedule

- Week 1-4: Study previous related internal and external works
- Week 5-8: Find an efficient and robust model which allows to locate face contours automatically and which provides additional information than just the shape
- Week 9-16: Implement the model and test it
- Week 17-18: Find and compute the characteristics values of the front-view face for a pictures sample of the Cohn-Kanade database.
- Week 19-24: Find, implement and test the algorithm for the expression recognition
- Week 25-26: Write the report

## B. Example of an AAM file

Points for Active Appearance Model

```
#1:36  
260,177  
260,218  
260,267  
261,284  
262,303  
263,319  
265,329  
268,339  
273,351  
283,366  
294,381  
312,395  
332,408  
344,414  
352,417  
361,421  
372,423  
385,423  
398,423  
413,421  
428,418  
441,413  
453,405  
461,394  
468,383  
476,375  
487,363  
495,349  
498,340  
503,329  
503,311  
503,301  
503,270  
503,247  
503,206  
503,170  
#2:34  
362,212  
365,223  
366,236  
365,246  
363,258  
360,268  
355,274  
349,278  
344,284  
342,289
```



ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

342, 297  
 349, 301  
 356, 300  
 364, 300  
 368, 304  
 374, 308  
 381, 309  
 388, 307  
 392, 302  
 404, 298  
 401, 298  
 411, 298  
 416, 296  
 418, 289  
 417, 284  
 410, 275  
 402, 271  
 392, 265  
 389, 259  
 389, 249  
 389, 237  
 389, 228  
 391, 218  
 395, 209  
 #3:16  
 347, 227  
 341, 221  
 335, 218  
 329, 213  
 320, 210  
 312, 210  
 304, 211  
 298, 215  
 293, 220  
 299, 225  
 307, 230  
 314, 230  
 322, 229  
 329, 228  
 338, 228  
 342, 227  
 #4:18  
 406, 226  
 413, 220  
 420, 216  
 426, 211  
 432, 208  
 438, 206  
 444, 206  
 451, 209  
 456, 212  
 462, 216  
 466, 222

459,225  
451,227  
442,227  
432,227  
425,227  
417,227  
412,227  
#5:20  
355,208  
353,201  
347,195  
340,191  
332,186  
325,183  
317,183  
307,183  
298,184  
292,188  
287,194  
295,191  
301,189  
308,189  
316,189  
323,190  
330,194  
337,198  
342,202  
348,205  
#6:24  
409,197  
414,190  
419,186  
424,181  
430,178  
437,175  
445,172  
453,172  
462,173  
470,176  
479,181  
484,186  
490,195  
479,192  
473,186  
465,181  
457,179  
449,179  
442,180  
436,182  
429,185  
424,188  
419,191  
414,194

#7:39  
332, 345  
336, 350  
341, 355  
348, 356  
353, 356  
360, 356  
366, 356  
370, 357  
376, 359  
383, 360  
388, 359  
394, 359  
400, 359  
408, 358  
416, 358  
421, 358  
426, 355  
432, 354  
435, 350  
439, 347  
435, 343  
428, 339  
422, 338  
415, 337  
409, 337  
403, 338  
399, 339  
395, 340  
390, 341  
386, 342  
381, 342  
375, 341  
368, 340  
361, 338  
356, 338  
352, 339  
346, 340  
340, 341  
337, 342

### C. Example of an ASF file

```
#####
#
#   AAM Shape File - written: Tuesday July 25 - 2006 [12:33]
#
#####

#
# number of model points
#
122

#
# model points
#
# format: <path#> <type> <x rel.> <y rel.> <point#> <connects from>
<connects to> <user1> <user2> <user3>
#
0      0      0.40136676  0.36518024  0      0      1      0.00  0.00  0.00
0      0      0.39485458  0.42450437  1      0      2      0.00  0.00  0.00
0      0      0.39353967  0.48489212  2      1      3      0.00  0.00  0.00
0      0      0.39800679  0.54438909  3      2      4      0.00  0.00  0.00
0      0      0.40468706  0.60397396  4      3      5      0.00  0.00  0.00
0      0      0.41361814  0.66294780  5      4      6      0.00  0.00  0.00
0      0      0.42895674  0.71966446  6      5      7      0.00  0.00  0.00
0      0      0.45358886  0.76946325  7      6      8      0.00  0.00  0.00
0      0      0.48803194  0.80954338  8      7      9      0.00  0.00  0.00
0      0      0.52814001  0.83934570  9      8      10     0.00  0.00  0.00
0      0      0.57131780  0.85862989  10     9      11     0.00  0.00  0.00
0      0      0.61750803  0.86040945  11     10     12     0.00  0.00  0.00
0      0      0.66207799  0.84614439  12     11     13     0.00  0.00  0.00
0      0      0.70147620  0.81492163  13     12     14     0.00  0.00  0.00
0      0      0.73489080  0.77245170  14     13     15     0.00  0.00  0.00
0      0      0.76060632  0.72181013  15     14     16     0.00  0.00  0.00
0      0      0.77527809  0.66329456  16     15     17     0.00  0.00  0.00
0      0      0.78130733  0.60226972  17     16     18     0.00  0.00  0.00
0      0      0.78553550  0.54069368  18     17     19     0.00  0.00  0.00
0      0      0.78890719  0.47893551  19     18     20     0.00  0.00  0.00
0      0      0.78466697  0.41708744  20     19     21     0.00  0.00  0.00
0      0      0.77399229  0.35667215  21     20     21     0.00  0.00  0.00
1      1      0.56375361  0.43582430  22     22     23     0.00  0.00  0.00
1      1      0.56645822  0.46349902  23     22     24     0.00  0.00  0.00
1      1      0.56553918  0.49197844  24     23     25     0.00  0.00  0.00
1      1      0.56334999  0.52080165  25     24     26     0.00  0.00  0.00
1      1      0.55704943  0.54782774  26     25     27     0.00  0.00  0.00
1      1      0.54265700  0.56825953  27     26     28     0.00  0.00  0.00
1      1      0.53198451  0.59168501  28     27     29     0.00  0.00  0.00
1      1      0.53909457  0.61422601  29     28     30     0.00  0.00  0.00
1      1      0.56068051  0.61568670  30     29     31     0.00  0.00  0.00
1      1      0.58055557  0.62200749  31     30     32     0.00  0.00  0.00
```

ENSEIRB	The automatic recognition of facial expressions						TU Delft		
---------	---	--	--	--	--	--	----------	--	--

1	1	0.60166396	0.62446176	32	31	33	0.00	0.00	0.00
1	1	0.62066770	0.61584036	33	32	34	0.00	0.00	0.00
1	1	0.64136906	0.61511756	34	33	35	0.00	0.00	0.00
1	1	0.65574223	0.59671221	35	34	36	0.00	0.00	0.00
1	1	0.64915720	0.56911772	36	35	37	0.00	0.00	0.00
1	1	0.62881907	0.55390456	37	36	38	0.00	0.00	0.00
1	1	0.61767005	0.52911698	38	37	39	0.00	0.00	0.00
1	1	0.61283737	0.49957057	39	38	40	0.00	0.00	0.00
1	1	0.61042331	0.46959031	40	39	41	0.00	0.00	0.00
1	1	0.61325652	0.43937947	41	40	41	0.00	0.00	0.00
2	2	0.53796049	0.46246552	42	57	43	0.00	0.00	0.00
2	2	0.52857255	0.44940387	43	42	44	0.00	0.00	0.00
2	2	0.51722992	0.43924570	44	43	45	0.00	0.00	0.00
2	2	0.50484493	0.43283228	45	44	46	0.00	0.00	0.00
2	2	0.49137946	0.43204043	46	45	47	0.00	0.00	0.00
2	2	0.47810973	0.43264681	47	46	48	0.00	0.00	0.00
2	2	0.46601329	0.43779183	48	47	49	0.00	0.00	0.00
2	2	0.45576231	0.44655356	49	48	50	0.00	0.00	0.00
2	2	0.44869454	0.45798581	50	49	51	0.00	0.00	0.00
2	2	0.45775468	0.46472215	51	50	52	0.00	0.00	0.00
2	2	0.46880747	0.46845909	52	51	53	0.00	0.00	0.00
2	2	0.48048875	0.47146268	53	52	54	0.00	0.00	0.00
2	2	0.49235152	0.47116821	54	53	55	0.00	0.00	0.00
2	2	0.50411323	0.46952381	55	54	56	0.00	0.00	0.00
2	2	0.51588911	0.46728811	56	55	57	0.00	0.00	0.00
2	2	0.52773039	0.46513690	57	56	42	0.00	0.00	0.00
3	2	0.63623992	0.45482451	58	73	59	0.00	0.00	0.00
3	2	0.64702329	0.44244355	59	58	60	0.00	0.00	0.00
3	2	0.65895168	0.43288137	60	59	61	0.00	0.00	0.00
3	2	0.67131539	0.42505785	61	60	62	0.00	0.00	0.00
3	2	0.68485610	0.42114493	62	61	63	0.00	0.00	0.00
3	2	0.69883205	0.42098609	63	62	64	0.00	0.00	0.00
3	2	0.71261533	0.42375840	64	63	65	0.00	0.00	0.00
3	2	0.72426786	0.43199871	65	64	66	0.00	0.00	0.00
3	2	0.73393364	0.44392763	66	65	67	0.00	0.00	0.00
3	2	0.72840627	0.45663690	67	66	68	0.00	0.00	0.00
3	2	0.70504016	0.46088846	68	67	69	0.00	0.00	0.00
3	2	0.68926400	0.46183832	69	68	70	0.00	0.00	0.00
3	2	0.67335132	0.46085840	70	69	71	0.00	0.00	0.00
3	2	0.65756780	0.45721936	71	70	72	0.00	0.00	0.00
3	2	0.64295966	0.45697130	72	71	73	0.00	0.00	0.00
3	2	0.62963154	0.45751669	73	72	58	0.00	0.00	0.00
4	3	0.54516758	0.41554050	74	85	75	0.00	0.00	0.00
4	3	0.53216404	0.39037859	75	74	76	0.00	0.00	0.00
4	3	0.51000802	0.37763794	76	75	77	0.00	0.00	0.00
4	3	0.48554455	0.37354703	77	76	78	0.00	0.00	0.00
4	3	0.45999213	0.37247789	78	77	79	0.00	0.00	0.00
4	3	0.43860769	0.38598256	79	78	80	0.00	0.00	0.00
4	3	0.42658394	0.40989499	80	79	81	0.00	0.00	0.00
4	3	0.44406389	0.39987627	81	80	82	0.00	0.00	0.00
4	3	0.46662931	0.39129111	82	81	83	0.00	0.00	0.00
4	3	0.49059777	0.39290220	83	82	84	0.00	0.00	0.00
4	3	0.51234654	0.40146077	84	83	85	0.00	0.00	0.00

4	3	0.53386528	0.41041985	85	84	74	0.00	0.00	0.00
5	3	0.63508587	0.39657855	86	97	87	0.00	0.00	0.00
5	3	0.65222868	0.37045110	87	86	88	0.00	0.00	0.00
5	3	0.67559761	0.35702470	88	87	89	0.00	0.00	0.00
5	3	0.70053691	0.35210662	89	88	90	0.00	0.00	0.00
5	3	0.72602650	0.35436430	90	89	91	0.00	0.00	0.00
5	3	0.74810962	0.36895068	91	90	92	0.00	0.00	0.00
5	3	0.76186540	0.39681960	92	91	93	0.00	0.00	0.00
5	3	0.74348218	0.38942180	93	92	94	0.00	0.00	0.00
5	3	0.72006782	0.37345490	94	93	95	0.00	0.00	0.00
5	3	0.69320018	0.37346816	95	94	96	0.00	0.00	0.00
5	3	0.66800996	0.38327614	96	95	97	0.00	0.00	0.00
5	3	0.64374665	0.39353852	97	96	86	0.00	0.00	0.00
6	4	0.51570806	0.70268622	98	121	99	0.00	0.00	0.00
6	4	0.52561133	0.71544837	99	98	100	0.00	0.00	0.00
6	4	0.53768365	0.72523392	100	99	101	0.00	0.00	0.00
6	4	0.55136690	0.73173899	101	100	102	0.00	0.00	0.00
6	4	0.56540403	0.73564703	102	101	103	0.00	0.00	0.00
6	4	0.57953706	0.73891326	103	102	104	0.00	0.00	0.00
6	4	0.59424001	0.73906912	104	103	105	0.00	0.00	0.00
6	4	0.60894166	0.73844267	105	104	106	0.00	0.00	0.00
6	4	0.62356013	0.73721340	106	105	107	0.00	0.00	0.00
6	4	0.63764234	0.73456777	107	106	108	0.00	0.00	0.00
6	4	0.65172899	0.73101720	108	107	109	0.00	0.00	0.00
6	4	0.66455806	0.72223820	109	108	110	0.00	0.00	0.00
6	4	0.67685349	0.71197770	110	109	111	0.00	0.00	0.00
6	4	0.68342363	0.70097405	111	110	112	0.00	0.00	0.00
6	4	0.67235715	0.69384140	112	111	113	0.00	0.00	0.00
6	4	0.65788946	0.69010405	113	112	114	0.00	0.00	0.00
6	4	0.64260320	0.68861298	114	113	115	0.00	0.00	0.00
6	4	0.62735065	0.69027617	115	114	116	0.00	0.00	0.00
6	4	0.61156416	0.69260654	116	115	117	0.00	0.00	0.00
6	4	0.59562552	0.69363261	117	116	118	0.00	0.00	0.00
6	4	0.57955754	0.68927094	118	117	119	0.00	0.00	0.00
6	4	0.56327812	0.68840216	119	118	120	0.00	0.00	0.00
6	4	0.54746516	0.68886552	120	119	121	0.00	0.00	0.00
6	4	0.53074877	0.69226072	121	120	98	0.00	0.00	0.00

```
#
# host image
#
S010_001_01594225.BMP
```

## D. Configuration file used for generating the model

```
#####
#####
#
#   Active Appearance Model Builder Configuration File
#
#####
#####

1      # Model reduction          [1-n]   (reduction factor =
1/x)

0      # Model expansion          [0-n]   (pixels along the point
normal)

1      # Use convex hull          [0|1]   (off/on)

0      # Verbose mode             [0|1]   (off/on)

1      # Write registration movie  [0|1]   (off/on)

1      # Write variance image     [0|1]   (off/on)

1      # Produce model documentation [0|1]   (off/on)

1      # Use tangent space projection [0|1]   (off/on)

1      # Training method [ 0=PC Regression, 1=Jacobian (recommended) ]

95     # Shape model truncation (percentage [0-100], -1=parallel
analysis)

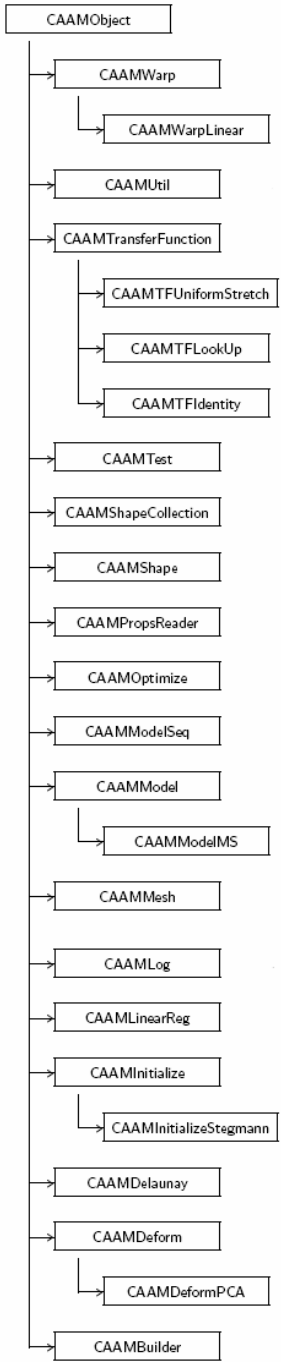
95     # Texture model truncation (percentage [0-100], -1=parallel
analysis)

95     # Combined model truncation (percentage [0-100], -2=no combined
model)

1      # Subsampling of the training set (during training) [1-n]

1      # Warping method [ 0=benchmark, 1=software, 2=hardware
(requires OpenGL) ]
```

### E. AAM-API class graph





## F. Model results

#####

Active Appearance Model File

Written : Thursday August 24 - 2006 [12:49]

Format version : 0.99

Build time : 58:03 (3482.8 secs)

Shapes : 400

Shape points : 122

Texture Bands : 1

Texture samples : 43979

Texture TF : identity

Model reduction : 1

Add Shape Extents : 0

Convex hull used : Yes

Tangent space used : Yes

Learning method : 1

Shape truncation level : 95 (variance: 0.00882/0.00928)

Texture truncation level : 95 (variance: 0.28/0.295)

Combined truncation level : 95 (variance: 0.533/0.56)

Parameters used : 58

Mean shape area : 43990.34

Combined mode variation :

1	24.45%	( 24.45%)
2	8.67%	( 33.11%)
3	6.84%	( 39.95%)
4	6.57%	( 46.52%)
5	5.32%	( 51.84%)
6	3.98%	( 55.82%)
7	3.59%	( 59.41%)
8	3.19%	( 62.60%)

9	3.08%	( 65.69%)
10	2.68%	( 68.36%)
11	2.45%	( 70.82%)
12	1.79%	( 72.60%)
13	1.64%	( 74.25%)
14	1.52%	( 75.76%)
15	1.38%	( 77.15%)
16	1.29%	( 78.44%)
17	1.05%	( 79.49%)
18	0.96%	( 80.45%)
19	0.88%	( 81.34%)
20	0.83%	( 82.16%)
21	0.81%	( 82.97%)
22	0.78%	( 83.75%)
23	0.72%	( 84.47%)
24	0.65%	( 85.11%)
25	0.62%	( 85.74%)
26	0.58%	( 86.32%)
27	0.53%	( 86.85%)
28	0.51%	( 87.36%)
29	0.47%	( 87.83%)
30	0.44%	( 88.27%)
31	0.44%	( 88.71%)
32	0.40%	( 89.11%)
33	0.39%	( 89.50%)
34	0.37%	( 89.86%)
35	0.36%	( 90.22%)
36	0.33%	( 90.55%)
37	0.31%	( 90.86%)
38	0.30%	( 91.15%)
39	0.28%	( 91.44%)
40	0.28%	( 91.71%)
41	0.26%	( 91.97%)
42	0.25%	( 92.22%)
43	0.25%	( 92.47%)
44	0.23%	( 92.70%)
45	0.21%	( 92.91%)
46	0.21%	( 93.13%)
47	0.21%	( 93.34%)
48	0.19%	( 93.53%)
49	0.19%	( 93.72%)
50	0.18%	( 93.89%)
51	0.17%	( 94.06%)
52	0.16%	( 94.23%)
53	0.16%	( 94.39%)
54	0.16%	( 94.55%)
55	0.15%	( 94.70%)
56	0.14%	( 94.84%)
57	0.14%	( 94.98%)
58	0.14%	( 95.12%)

Shape mode variation :

1	40.07%	( 40.07%)
---	--------	-----------

2	9.72%	( 49.78%)
3	6.93%	( 56.71%)
4	6.14%	( 62.85%)
5	5.50%	( 68.35%)
6	4.77%	( 73.12%)
7	4.36%	( 77.48%)
8	3.06%	( 80.55%)
9	2.50%	( 83.05%)
10	1.90%	( 84.95%)
11	1.71%	( 86.66%)
12	1.28%	( 87.94%)
13	1.08%	( 89.01%)
14	1.02%	( 90.03%)
15	0.85%	( 90.88%)
16	0.79%	( 91.67%)
17	0.68%	( 92.35%)
18	0.64%	( 92.99%)
19	0.56%	( 93.54%)
20	0.54%	( 94.09%)
21	0.51%	( 94.60%)
22	0.44%	( 95.04%)

Texture mode variation :

1	12.34%	( 12.34%)
2	10.67%	( 23.01%)
3	9.77%	( 32.78%)
4	7.90%	( 40.68%)
5	5.64%	( 46.32%)
6	4.53%	( 50.85%)
7	3.29%	( 54.14%)
8	2.81%	( 56.96%)
9	2.56%	( 59.52%)
10	1.97%	( 61.49%)
11	1.79%	( 63.28%)
12	1.64%	( 64.91%)
13	1.51%	( 66.42%)
14	1.34%	( 67.76%)
15	1.28%	( 69.04%)
16	1.18%	( 70.22%)
17	1.06%	( 71.28%)
18	0.98%	( 72.26%)
19	0.89%	( 73.15%)
20	0.78%	( 73.92%)
21	0.75%	( 74.67%)
22	0.69%	( 75.37%)
23	0.65%	( 76.01%)
24	0.62%	( 76.64%)
25	0.57%	( 77.20%)
26	0.56%	( 77.76%)
27	0.54%	( 78.30%)
28	0.52%	( 78.82%)
29	0.48%	( 79.30%)
30	0.47%	( 79.77%)

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

31	0.46%	( 80.23%)
32	0.45%	( 80.68%)
33	0.41%	( 81.09%)
34	0.40%	( 81.49%)
35	0.40%	( 81.89%)
36	0.38%	( 82.27%)
37	0.36%	( 82.63%)
38	0.35%	( 82.98%)
39	0.34%	( 83.31%)
40	0.31%	( 83.62%)
41	0.30%	( 83.92%)
42	0.30%	( 84.22%)
43	0.30%	( 84.52%)
44	0.29%	( 84.80%)
45	0.28%	( 85.08%)
46	0.26%	( 85.34%)
47	0.26%	( 85.60%)
48	0.25%	( 85.85%)
49	0.25%	( 86.10%)
50	0.23%	( 86.33%)
51	0.23%	( 86.57%)
52	0.23%	( 86.79%)
53	0.22%	( 87.01%)
54	0.22%	( 87.23%)
55	0.21%	( 87.43%)
56	0.20%	( 87.64%)
57	0.20%	( 87.83%)
58	0.20%	( 88.03%)
59	0.19%	( 88.23%)
60	0.19%	( 88.42%)
61	0.18%	( 88.60%)
62	0.17%	( 88.77%)
63	0.17%	( 88.95%)
64	0.17%	( 89.12%)
65	0.17%	( 89.29%)
66	0.16%	( 89.45%)
67	0.16%	( 89.61%)
68	0.15%	( 89.76%)
69	0.15%	( 89.92%)
70	0.15%	( 90.06%)
71	0.15%	( 90.21%)
72	0.14%	( 90.36%)
73	0.14%	( 90.50%)
74	0.14%	( 90.63%)
75	0.13%	( 90.77%)
76	0.13%	( 90.90%)
77	0.13%	( 91.03%)
78	0.13%	( 91.15%)
79	0.12%	( 91.28%)
80	0.12%	( 91.40%)
81	0.12%	( 91.52%)
82	0.12%	( 91.63%)
83	0.11%	( 91.75%)

84	0.11%	( 91.86%)
85	0.11%	( 91.97%)
86	0.11%	( 92.08%)
87	0.11%	( 92.18%)
88	0.11%	( 92.29%)
89	0.10%	( 92.39%)
90	0.10%	( 92.49%)
91	0.10%	( 92.59%)
92	0.10%	( 92.69%)
93	0.10%	( 92.79%)
94	0.09%	( 92.88%)
95	0.09%	( 92.97%)
96	0.09%	( 93.06%)
97	0.09%	( 93.16%)
98	0.09%	( 93.24%)
99	0.09%	( 93.33%)
100	0.09%	( 93.42%)
101	0.09%	( 93.51%)
102	0.08%	( 93.59%)
103	0.08%	( 93.67%)
104	0.08%	( 93.75%)
105	0.08%	( 93.83%)
106	0.08%	( 93.91%)
107	0.08%	( 93.99%)
108	0.08%	( 94.07%)
109	0.08%	( 94.14%)
110	0.07%	( 94.22%)
111	0.07%	( 94.29%)
112	0.07%	( 94.36%)
113	0.07%	( 94.43%)
114	0.07%	( 94.50%)
115	0.07%	( 94.57%)
116	0.07%	( 94.64%)
117	0.07%	( 94.71%)
118	0.07%	( 94.78%)
119	0.07%	( 94.84%)
120	0.07%	( 94.91%)
121	0.06%	( 94.97%)
122	0.06%	( 95.03%)

#####

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

## Bibliography

### External papers:

**H.Kobasyashi and F.Hara** - *Recognition of Mixed Facial Expressions by Neural Network* - IEEE International workshop on Robot and Human Communication, 381-386, 1972

**Ekman, P. and Friesen, W.** - *Facial Action Coding System* - Consulting Psychologists Press, Inc., Palo Alto California, USA, 1978

**Hara, F., Kobayashi, H.** - *A Basic Study of Dynamic Recognition of Human Facial Expressions* - Proceedings of IEEE International Workshop on Robot and Human Communication, pp.271-275, 1992-11

**Lee, Y., Terzopoulos, D., & Waters, K.** - *Constructing physics-based facial models of individuals* - Graphics Interface, Toronto, 1-8, 1993

**Thalmann, D., Kallmann, M.** - *Modeling Objects for Interaction Tasks* - Computer Animation and Simulation, 1998

**Bajcsy and Kovacic, A.** - *Multiresolution elastic matching* - Computer Graphics and Image Processing, 46:1-21, 1989

**Christensen, G. E. et al** - *Topological properties of smooth anatomic maps* - 14<sup>th</sup> Conference on Information Processing in Medical Imaging, France, pages 101-112. Kluwer Academic Publishers, 1995

**Park, J. et al** - *Deformable models with parameter functions for cardiac motion analysis from tagged MRI data* - IEEE Transactions on Medical Imaging, 15:278-289, 1996

**Pentland, A. P. and Scarloff, S.** - *Closed-form solutions for physically based modelling and recognition* - IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(7):715-729, 1991

**Poggio, T. and Ezzat, T.** - *Facial analysis and synthesis using image-based models* - 2<sup>nd</sup> International Conference on Automatic Face and Gesture Recognition 1997, pages 116-121, Killington, Vermont, 1996

**Poggio, T. and Jones, M. J.** - *Multidimensional morphable models* - 6<sup>th</sup> International Conference on Computer Vision, pages 683-688, 1998

ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

**Cootes, T. F. and Taylor, C. J.** - *Modelling object appearance using the grey-level surface* - E. Hancock, editor, 5<sup>th</sup> British Machine Vision Conference, pages 479-488, York, England, BMVA Press, Sept. 1994

**Nastar, C., Moghaddam, B. and Pentland, A.** - *Generalized image matching: Statistical learning of physically-based deformations.* - 4<sup>th</sup> European Conference on Computer Vision, volume 1, pages 589-598, Cambridge, UK, 1996

**Poggio, T. and Jones, M. J.** - *Multidimensional morphable models* - 6<sup>th</sup> International Conference on Computer Vision, pages 683-688, 1998

**Osuna, E., Freund, R. and Girosi, F.** - *Training Support Vector Machine: an Application to Face Detection* - Proceedings of CVPR'97, 1997

**Tipping, M. E.** - *Sparse Bayesian Learning and the Relevance Vector Machine* - Journal of Machine Learning Research, Vol. 1, p 211-244, 2001

**M. Covell** - *Eigen-points: Control-point location using principal component analysis* - 2nd International Conference on Automatic Face and Gesture Recognition 1997, pages 122-127, Killington, USA, 1996

**M. J. Black and Y. Yacoob** - *Recognizing facial expressions under rigid and non-rigid facial motions* - 1st International Workshop on Automatic Face and Gesture Recognition 1995, pages 12-17, Zurich, 1995

**M. Gleicher** - *Projective registration with difference decomposition* - IEEE Conference on Computer Vision and Pattern Recognition, 1997

**G. Hager and P. Belhumeur** - *Efficient region tracking with parametric models of geometry and illumination* - IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(10):1025-39, 1998

**S. Sclaroff and J. Isidoro** - *Active blobs* - 6th International Conference on Computer Vision, pages 1146-53, 1998

**M. La Cascia, S. Sclaroff, and V. Athitsos** - *Fast, reliable head tracking under varying illumination: An approach based on registration of texture mapped 3d models* - IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(4):322-336, 2000

**T.F.Cootes, G.J. Edwards, and C.J.Taylor** - *Active Appearance Models* - Proc. European Conference on Computer Vision 1998, Vol. 2, pp. 484-498, Springer, 1998.

**T.F. Cootes and C.J. Taylor** - *Statistical models of appearance for medical image analysis and computer vision* - Proc. SPIE Medical Imaging 2001

**M. B. Stegmann, B. K. Ersbøll, R. Larsen** - *FAME -- A Flexible Appearance Modelling Environment*, - IEEE Transactions on Medical Imaging, IEEE, 2003



ENSEIRB	The automatic recognition of facial expressions	TU Delft
---------	---	----------

**Cohn, J.F., Zlochower, A., Lien, J., and Kanade, T.** - *Automated face analysis by feature point tracking has high concurrent validity with manual FACS coding* - *Psychophysiology*, 36, pp.35-43

Internal papers:

**Anna Wojdel** - *Knowledge Driven Facial Modelling* - Delft University of Technology, 1998

**Edwin J. de Jongh** - *An online Facial Expression Dictionary as a first step in the creation of a complete Nonverbal Dictionary* - Delft University of Technology, 2002

**Dragos Datcu** - *Recognition of facial expressions* - Delft University of Technology, 2004

**Wai Shung Wong and William Chan** - *Automatic Facial Expression Recognition using a Sparse Learning Relevance Vector Machine* - Delft University of Technology, 2005