FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Avatar Control by Automatically Detected Face Interest Points

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Miroslav Byrtus
Matrikelnummer 1328167

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dr. Horst Eidenberger

Wien, 20. Dezember 2015

Miroslav Byrtus	Horst Eidenberger

FAKULTÄT FÜR !NFORMATIK

Faculty of Informatics

Avatar Control by Automatically Detected Face Interest Points

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics

by

Miroslav Byrtus
Registration Number 1328167

to the Faculty of Informatics

at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dr. Horst Eidenberger

Vienna, 20th December, 2015

Miroslav Byrtus | Horst Eidenberger

Erklärung zur Verfassung der Arbeit

Miroslav Byrtus
Vorgartenstraße 67, 1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 20. Dezember 2015

Miroslav Byrtus

Danksagung

Vielen Dank für eure Unterstützung. Insbesondere möchte ich Herrn Ao.Univ.Prof. Dr. Horst Eidenberger für die engagierte und professionelle Betreuung danken. Des weiteren möchte ich mich bei den geduldigen Probandinnen und Probanden für die Zusammenarbeit bedanken.

Ganz besonders möchte ich mich bei meiner Familie bedanken, die mir mein Studium und diese Diplomarbeit durch ihre liebevolle und aufrichtige Unterstützung ermöglicht haben.

Danke!

Acknowledgements

Many thanks for your support. I would especially like to thank Ao. Univ. Prof. Dr. Horst Eidenberger for his dedicated and professional support. I would also like to thank the participants of the evaluation for their patience and cooperation.

Additionally, I especially want to thank my family, who facilitated my studies and this thesis through their loving and sincere support.

Thank you!

Kurzfassung

Das Entwerfen von Systemen, die fähig sind mit den Menschen zu interagieren, ist eine aufwändige Aufgabe. Ein wichtiger Aspekt bei diesem Problem ist, die menschlichen Emotionen zu verstehen und auf diese auf menschliche Weise zu reagieren. Kompliziert ist auch die Tatsache, dass Menschen selber Probleme haben, die Emotionen richtig zu erkennen.

Derzeit gibt es zahlreiche robuste und gut funktionierende Systeme, die Gesichter erkennen, Augen, Nase und Mund lokalisieren können. Hier fehlt aber die sogenannte Meta-Information in Form einer ausführlichen Beschreibung des Gesichts, die ein tieferes Verständnis des Ausdrucks bringen kann. Diese Information sollte nicht unterschätzt werden, denn die Gesichtsausdrücke beinhalten eine große Menge von Information der non-verbalen Kommunikation. Die Gesichtsausdrücke spielen in der menschlichen Kommunikation eine wichtige Rolle, denn eine große Informationsmenge wird auch durch die non-verbale Kommunikation übermittelt.

Ein System, das die menschlichen Emotionen automatisch erkennen kann, wäre für Bereiche wie Human-Computer Interaktion, Psychologie, Soziologie etc. hilfreich. Ein solches System würde eine automatische Analyse von Stress-, Höhenangst und Aggressivität ermöglichen. Außerdem würde es auch für die Überwachung nutzbar sein.

Das Ziel dieses Projektes ist es, ein robustes System zu entwerfen und zu implementieren, das die Emotionen in Gesichtern erkennen und analysieren kann. Das System wird voll automatisiert, sodass der Benutzer keine weiteren Einstellungen tätigen muss, um das System zum Laufen zu bringen.

Die erwartete Ausgabe ist eine textuelle Beschreibung der Emotion. Die analysierten Gesichtsparameter werden zudem weiter an die Animationskomponente geschickt, wo die erkannte Emotion in der Form einer Animation nachgespielt und angezeigt wird.

Abstract

Designing systems that are able to interact with people is a complex process. An important aspect of this problem is understanding human emotions and responding to them in a human way. The fact that people themselves often have problems in recognizing emotions properly makes the task even more difficult.

There are currently numerous robust and well-functioning systems that can recognize human faces, and locate the eyes, nose and mouth. However, these systems miss the so-called meta-information in the form of a detailed description of the face, which can lead to a deeper understanding of facial expressions. This information should not be underestimated, since facial expressions contain a large amount of non-verbal information. Facial expressions are important in human communication because much information is transmitted through non-verbal communication.

A system that can automatically detect human emotion would be useful in areas such as human-computer interaction, psychology, sociology and other areas. Such a system would enable automated analysis of stress, vertigo or aggression levels. Moreover, it would also be useful in monitoring public spaces, resulting in higher security.

The aim of this project is to design and implement a robust system that can recognize and analyze emotions from human faces. The system should be fully automated so that the user does not need to setup any parameters in order to make the system run correctly.

The expected output is the textual description of the emotion. The analyzed face parameters are also forwarded to the animation component, where the facial expression is animated on an avatar.

Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1 Idea

The goal of this thesis is to propose a system for deep visual face analysis. As the proof of concept, a prototype application for recognizing and animating user facial emotions in real-time was implemented.

This ushers in a new level of image analysis that can exist alongside advanced scene analysis, which is also used, for example, for an emotion-driven human-computer-interface. Due to its sequential input, the content and responses can be adapted over time according to the user's emotional responses. Some actions can be also automatized, or at least predicted according to the user's emotions.

As a side effect of emotion recognition, the user's facial data can be used for an avatar animation. Since the face analysis precisely describes the user's face, it can also replicate the emotion precisely. In this way the user's body language can be credibly animated, while the user's real face remains anonymous.

Even though the system is to be developed and tested on Europeans only, the idea is to create a robust solution that will work with any race, age or gender without limitations.

Assuming minimal lighting conditions are met, the proposed solution will be able to work fluently on a common personal computer with no need for special hardware. The user will be captured by a web-cam and the data will be analyzed with algorithms that even modern mid-end laptops will be able to handle.

1.2 Motivation

The motivation for this work is to enable the computer to understand and respond to a human-specific expression such as facial emotion. With this capability, programs will be able to reach a new dimension of semantic information.

Emotional expressions happen spontaneously over time. In most cases people do not think explicitly about what triggers in them an honest reaction. If a machine could evaluate facial expressions automatically, the user's response time could be decreased in some scenarios with less user actions required.

Body language is an important part of interpersonal communication, especially the expressions of the face and hands. A system capable of analyzing the user's facial emotion and animating it on an avatar enables the use of this kind of body language over the Internet. Thereby enriching the expressive power of virtual communication without sacrificing the user's anonymity.

A novel human-computer-interface is another aspect of the motivation. Currently the usual setting comprises the user sitting in the front of the computer and seeing some content to which he or she is able to respond to, for example sending a response with the keyboard. Between these two points, the user reacts spontaneously, and facially, to the content. The proposal comprising this thesis could also be used for this kind of task.

1.3 Challenges

The most challenging part of this work is to describe the face so precisely that the individual facial parts, the eyes, mouth etc., can be analyzed separately according to their form and position relative to other facial parts. Such analysis requires a system of face interest points to be designed, describing the form and the position of every face feature of an appropriately fitted image.

There are several published approaches and methods for the extensive description of human faces, but there is as yet no ideal approach for the particular task of emotion analysis. None of the published algorithms and approaches are sufficiently precise or applicable for real-time application, a fact that makes the choice of approach a challenging task. The more precise the method, the higher the computational power needed. This work therefore searches for the perfect ratio between precision and performance.

It also searches for existing methods, algorithms and libraries that can be combined together in order to achieve some of the functionality. These are listed and compared in the following chapters. Following this the most suitable combination will be chosen and the rest of the functionality will be implemented.

Another technical challenge is to use only open-source libraries, which can be adapted to the functionality required for this thesis. The preferred programming language is C/C++, so that communication between the recognizer and animator in the final product can be implemented optimally without any limitations. Even if the final product were to be compiled for usage under the Microsoft Windows operating system, the source code ought to be platform-independent.

1.4 Overview

This thesis is organized in two parts; research and implementation. The former investigates approaches that could potentially be used in the prototype. The precision and computational complexity of the chosen methods are described and compared with each other.

In the latter approach, we chose the best method evaluated and based the prototype implementation on it. The prototype was also tested and evaluated among a small group of people. The project was described, results discussed and future work was proposed.

The description of the prototype also consists of multiple sections. Firstly the user's face has to be captured, localized and described so precisely, that the positioning and form of certain face features can be further analyzed. In order to achieve this a system of significant points on the face had to be designed. These points will then be located on the face to create a face map that helps to analyze the face further.

The prototype will only work correctly if a set of minimum requirements is met. The computer needs to be able to access the web-cam, which has to capture the whole frontal face of the user. The face has to be captured in a way the visual information is fine enough to distinguish clearly the individual face features.

Assuming that the minimal requirements are met, the significant points mapping is applied and fitted onto the visual image. This mapping is then analyzed further and compared to a set of pre-trained emotions, predicting the actual emotional expression of the perceived face. This is executed in a loop over time, so that a real-time sequential emotion prediction is reached. From this mapping and precision prediction an avatar face with textual description is rendered. With the mapping precise enough, processed and applied to the 3D model properly, an almost real emotional expression can be simulated.

CHAPTER 2

State of the Art

Since this thesis is organized in two parts, the *State of the Art* chapter is also organized accordingly. Firstly, methods related to the emotion recognition part are described, and then the animation part commences.

Currently, thanks to the rapidly growing Web and IT technologies, many algorithms and libraries are published that relate to this topic in some way. There are different approaches for locating the face, finding and matching patterns and shapes, and many approaches to machine learning. To attain the functionality required in this thesis a combination of these methods will be needed, so that the chapter deals with all of them.

- In the first step, technology for locating a user's face is required. The simplest way to locate objects in visual images is to use visual trackers. However, for human faces this is not the best way for accomplishing such a task, for two reasons. First, having trackers on the face would be unnatural and would not represent a real environment. Second, to be able to track the face so thoroughly that the emotion could be recognized, dozens of trackers on a small face-sized surface would be needed. This would lead to an unrealistic setting that would not find relevance in the real world. For these reasons tracker-less methods for locating and describing faces based on image processing approaches were researched and used. Section 2.1 focuses on this topic more closely.

- The next problem in this thesis concerns the points of interests of a human face. The location of the face itself would not be sufficient for deeper analysis, such as recognizing emotions. This is why the face needs to be located more thoroughly and with greater detail - information about locations and the shapes of particular face features are needed. For such thorough, detailed facial description a system of significant points needs to be defined in order to clearly define the position and shape of particular face features on the image. There are multiple systems using different positioning and point counts, and which are described in more detail in the following sections.

- With such a system of points the fitted point map can be analyzed and compared to pre-defined maps of emotions. How to pre-define a configuration and how to compare and match them to a current one is analyzed in the following sections. Most approaches do use annotated face databases for training a model that is able to compare the current map of points to the trained one, and to tell which trained one is most similar to the current one. This training-classifying method is closest to that used by humans for understanding perceived content. Since the task is to enable machines to recognize a human facial expression in a human way, this method is the most appropriate. If implemented correctly, the prototype will be able to recognize facial emotions in much the same way that humans do.

- There are two ways of classifying emotions: static and dynamic. The static method takes a static image of a human face and recognizes the expression on it. With a model trained properly, a simple static image contains enough information for understanding the emotion on a face. Conversely, dynamic video content contains more information than the static one. According to a study by C. Soladié, H. Salam, C. Pelachaud, N. Stoiber and R. Séguier [SSP$^+$12] the duration of a smile is also important when recognizing an expression. Short smiles can have different meanings than longer ones. In dynamic video recordings, the last facial expression can even be compared to previous expressions, and the difference measured. According to this, differences and differences to the emotionless expression, the strength of an expression can be also measured.

2.1 Face Analysis

This section concerns approaches to facial analysis, which consist of localizing a frontal human face and matching its shape to a model. The section is divided into two subsections: Face Recognition and Face Analysis; these two tasks differ in several aspects and there are special methods peculiar to each. The methods are now described and compared, as is choosing the implementation that will best fit our needs. These were then tested and combined in order to create the most optimized base for the prototype solution.

2.1.1 Face Recognition

The first step of the image processing required for reaching the final functionality of the prototype is to localize the face. This is in order to determine whether there is a recognizable face captured suitable for further processing. If not, then either there is no one in the front of the camera, or the lighting conditions are insufficient for face recognition. In either case the process should be stopped with a warning, since the system cannot work properly unless there is a recognizable face on the image.

Assuming that the bounding rectangle will be returned from the face recognition method, the visual information can be reduced by removing any unnecessary part of the image from the bounding rectangle. In this way, the process can zoom-in and focus solely on the face. This brings a further optimization level to the whole process.

There are several methods for detecting objects in images. One reliable and currently widely used method was introduced by P. Viola and M. Jones in 2001, [VJ01].

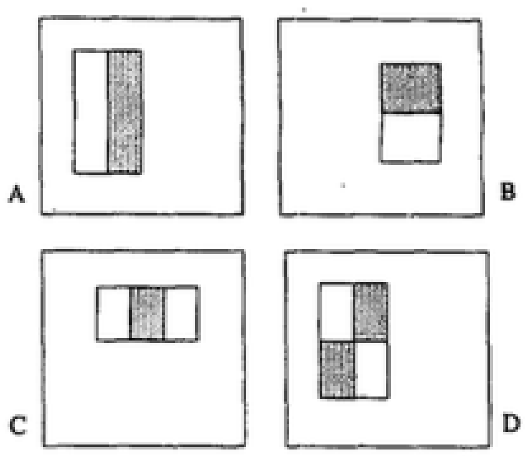

Figure 2.1: Simple rectangle features used in the work [VJ01].

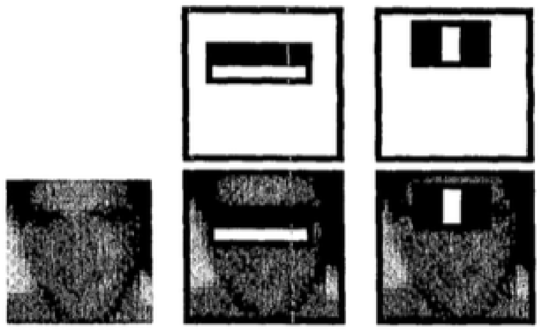

Figure 2.2: Features selected by AdaBoost. The first feature notices the lightness difference between eyes and the cheeks. The second notices the difference in the eye region caused by the nose being between the eyes. [VJ01]

This method was improved in 2002 by the work of R. Lienhart and J. Maydt [LM02], where the Haar-like features were extended by rotating into four edge features, eight line features and two center-surround features, shown in Figure 2.3.

As a result, a reliable and robust method was created, which was also directly implemented in the OpenCV library [70], making it a highly suitable method for this thesis.

[70]http://opencv.org/, 21.10.2015

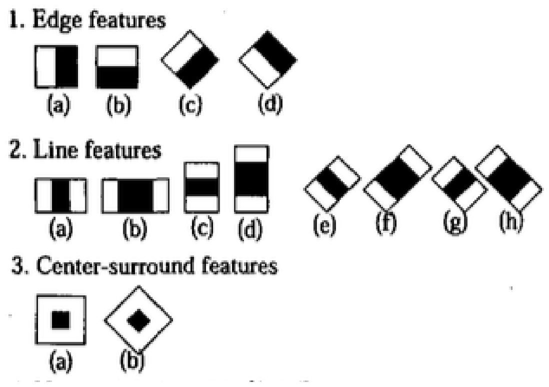

Figure 2.3: Rotated features introduced in the publication [LM02].

2.1.2 Face Shape

In the previous Section 2.1.1, the face was only localized with a bounding rectangle returned. We are thus still limited to pure visual data of the user's face. It would clearly not be the most optimized way to analyze emotion using only visual content. This is why the present section discusses how to reduce the amount of data while describing the face far more comprehensively. In general, for understanding emotion, the shapes and positions of particular face features are sufficient. Therefore so the goal of this section is to find a method that can provide us with this data.

To predict emotions directly from images of human faces would not be impossible, but still far from an optimal way of solving the task. There is too much information that is unrelated to the emotion, which would make it complicated, computationally expensive and not robust enough. Since shapes and the positions of face features are enough to define an emotion, the shape information needs to be extracted from the visual information.

Thus a more comprehensive description of a the shape of a face is needed. In this case this means shape descriptions and localizations of particular face features, such as the mouth, eyes and eyebrows. To understand human facial emotional expressions, the positions and the shapes of organs are needed. This is the reason why a biometric analysis needs to be performed, and for this a set of reference points is required. The problem with suggesting such set of points is that they need to be positioned in such a way that, if localized correctly, they would clearly indicate the positions and the shapes of the facial organs, enabling the recognition of emotion. Additionally, the number of reference points is also an important aspect, hence all important information should be captured by as few points as possible.

2.1.3 Active Appearance Models

Active Appearance Models (AAM) comprises a special image processing approach, which deals with deformable objects. It is able to be trained for a specific object that can vary in form. With such an AAM model, the fitting algorithm comes into play. It can fit the trained model to the new, unseen image. Parameters applied to the trained model in order to match the new image are returned. These parameters describe the actual form of the object.

The method consists of two parts, the parametric model and the fitting algorithm. With the fitting algorithm, the parametric model can be fitted to any unseen image that allows for deeper analysis and description.

Since this method also stores the appearance of the object, the visual information can be combined with the matching shape. However, for the appearance only, gray-level information is stored [CET98]. As a result of this the AAM can also generate synthetic images from the model [SRU+10].

The AAM is a general-purpose tool that can be trained for any kind of object, not only the human face. Having the shape of the whole object and its interior parts, a deep object analysis can begin, resulting in an extensive object description. The shape information comprises a set of parameters to be applied to the model in order to fit the new image. With the resulting description, information concerning deformation of the face and its parts can be collected and used further. This is why this approach is so interesting and relevant to this thesis. Since a human face is a deformable object, the AAM can be trained for it, and returns exactly what is expected - the actual shape of the human face that is required for further analysis and emotion recognition. Such a description of form is useful not only for the emotion recognition of the human face, but can also be used in medicine, where particular deformations of cells or organs can be used to indicate a disease.

There are already published papers that deal with the analysis of human facial expressions. For example, the publications of Dhall [Dha13] and Garcíia Bueno et. al [GBGFMB12] deal with expression analysis using AAM on human faces. The latter, the work of Garcíia Bueno et. al, also uses AAM to locate face features, which are then classified using *neural evolution based on neutral networks and differential evolution algorithm* [GBGFMB12] in order to predict a user's emotion from facial expressions.

Studies by Ashraf et. al [ALC+07] and Hammal et. al [HC12] have shown the use of AAM in the task of recognizing pain in human facial expressions. Pain is one of the expressions that can also be recognized and measured from pure visual content. These publications were used as proof that this method is suitable for the problem of recognizing emotions from facial expressions. Other expressions can also be recognized using AAMs, such as happiness, fear, sadness, and others. Further to this, in 2013 Datcu et. al [DCLR13] proposed a solution showing the use of AAM for heart rate analysis derived from facial features.

In previous paragraphs the AAM approach was suggested as an ideal solution for preparing data for the final emotion analysis. The method is robust, fast, and offers a low failure rate, but it also has its drawbacks. The method encounters heavy problems when images are partially occluded. If important features are occluded the fitting algorithm becomes lost and fails to match the other features. Thus if the object is at least partially occluded then the algorithm does not usually fit the model. When working with human faces especially, this problem is a major one. Here occlusion often appears, particularly with people wearing glasses, with beards, long hair or make-up. If seen by a human it does not at first appear to be occlusion, because for humans it is already common. However, the machine sees it as occlusion, since the spectacles frame or the beard occludes the face features. Even heavy make-up can cause a disorder in the appearance of the face that can lead to incorrectly located features. In these cases the algorithm cannot recognize the object correctly, leading to an incorrect match. The machine cannot handle all these variations and the special exceptions during the training phase.

Nevertheless attempts are made to avoid occlusion and to ensure the method is robust. In 2010, Storer et. al [SRU$^+$10] proposed in their article a robust AAM fitting strategy as an improvement to counter occlusions. The idea was based on and inspired by the publications of Nguyen et. al [NLEDlT08] and Xiaodong Jia et. al [JG10]. Both publications deal with the usual facial occlusions. The former proposed a method for image-based shaving [SRU$^+$10], which removes beards from images of human faces using an image-processing approach. Beards can occlude up to a third of the face and can also visually change the shape of the jaw. The latter study, dealt with occlusion caused by spectacles [JG10], which is also a common occlusion in a real world. These are two of the most common occlusions of a human face, and comprises a good basis for such enhancement.

As mentioned above, AAM consists of a model and a fitting algorithm. Since the model has to be trained for the object of interest, the training stage is also mentioned here. For this annotated images of objects of interest are needed. The annotations are points of interest that will create the contour of whole objects and its parts if connected in the correct way. For facial expression recognition, annotated faces with different expressions are needed. How many points are required and on which positions is the topic of the next section. There are systems that use different amounts and different positions for these annotations. In general they are very similar, but still slightly different, according the kind of data sought. See the following figures 2.4, 2.5, 2.6.

The exact flow was explained in the work of Kohli et. al [KPG11]. The shape is defined by landmarks localized on the object. The trained model contains the mean shape and mean appearance of the object. This mean shape is then used by the fitting algorithm, which applies parameters to the mean shape when trying to fit the new image. A similar approach is used for extracting the appearance part. However, this is based on the normalized mean grey-level appearance instead of the shape. Collected shape and

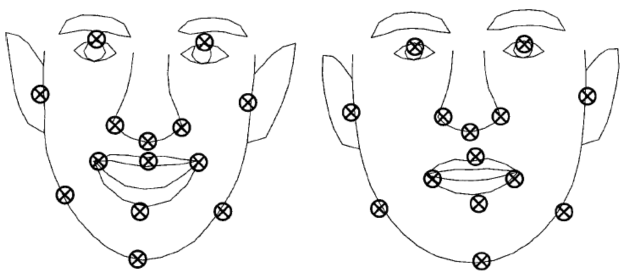

Figure 2.4: A minimal landmark configuration that is enough to match the facial shape, but is not enough for deeper analysis, such as emotion recognition [LTC97].

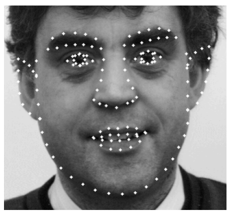

Figure 2.5: Another configuration where all face parts are annotated in detail, so that the shape can be analyzed [LTC97]. This setting should be good enough to recognize emotion from the fitted model.

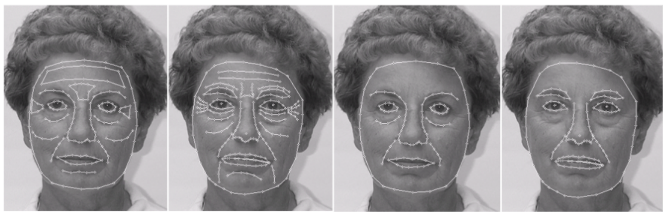

Figure 2.6: A set of different annotation schemes [SRP10] with different levels of details. All the scheme details are enough for an emotional recognition task, with different levels of detail and precision.

grey-level parameters are combined, resulting in the shape and appearance model of the newly fitted object.

2.1.4 Active Shape Models

Active Shape Models (ASM) are the predecessors of Active Appearance Models. They differ in that no appearance part is obtained. Since the appearance part is avoided, there is less data to be processed, which makes the fitting algorithm run faster. On the other hand, according to the study of Cootes et. al [CET98], there is less data to be processed, making this approach less robust than the AAM.

The first examples of usage were introduced in 1995 by T.F. Cootes et. al [CTCG95], where Active Shape Models were used for fitting the shapes of resistors, hands and hearts. In this study the process of aligning the training set was demonstrated. They invariably took a pair of shapes and compared them to each other, trying to match one shape to another by translating, rotating and scaling it so that it fitted the second shape as exactly as possible. The difference between the two shapes was calculated as the sum of the distances between point locations. The bigger the difference, the smaller the weight assigned to the particular shape.

The Active Shape Model also lacks the possibility of synthesizing new, untrained images from trained ones, as the AAM can. Although this difference makes the method less robust, it also makes it faster, since there is less data to be processed. It depends upon the task itself whether computational costs are more important than precision and robustness. If neither synthesis nor the appearance part is needed, then it is in the developer's hands to decide which method to choose. As in this thesis, the appearance part is not required at all, although the robustness and the precision of the fitting algorithm is essential for the task. This is why libraries for both Active Shape and Appearance Models were tested and compared, so that the best method can be chosen.

Since the synthesis of new faces is useless for emotion recognition, Active Shape Models have the advantage of speed here. The proof of concept needs to run in real-time, making speed an important requirement. On the other hand, Active Appearance Models still remain more robust and precise than Active Shape Models, which is an advantage in every task. The actual testing on hardware with trained AAM and ASM models shows the comparison, which is concluded in next sections.

2.1.5 Face Databases

Since we use a classifier that needs to be trained, some training data is needed. Training data is clearly an essential part of the whole classification process. The quality of results depends not just on the training algorithm but on the quality and amount of training data. The better the training, the bigger the chance of avoiding classification mismatches.

We need to train two models for the prototype, so we also need two different training data sources. First one will be used in the Active Shape or Appearance Model. This will be important for properly matching the shape to the face. Since third party libraries will be used for ASM/AAM, there is a probability that there already are some pre-built models of human faces that work well.

For building a proper ASM/AAM model, thousands of faces need to be trained. For such purposes there are face databases published on the Internet. In order to test the solution thoroughly several were tested in the implementation part. Since human faces differ intensively, face databases also differ in many aspects. Besides the differences in human faces, they also differ in the angle of the captured faces, image quality, annotations, and some of them even capture faces multiple times with diverse facial expressions. The training algorithm of the ASMs and AAMs has a special requirement for training images of the objects of interest. It expects annotated images as input. Usually annotations are stored in separate files that describe the positions of the face interest points. When choosing or creating a face database, the face interest points system should also be considered. The count and the location of points needs to be declared before the training process actually starts. Since annotating images manually takes up too much time, we sought for an already annotated face database with a fitting annotation system. Additionally a small test database was created for testing the fitting algorithm on one's own face with our own annotation system. Testing the model with the same face that the model was trained with is the best case scenario, although this would not happen in a real setting.

The following databases were tested and used in this thesis: *BioID* [80], *IMM* [81], *FRANCK* [82] and *MUCT* [83]. Additionally, as mentioned above, a small local database was also created and tested, containing five to ten annotated images of one face. Since this would not be sufficient input for general usage, it was tested on the same face as in the database, and used for testing and debugging purposes only.

The second database needed is a database of emotional expressions linked to the shape ASM/AAM model. With this model trained, the emotion can be classified. In comparison to the first type of face database, this only contains additional information about the captured expression. Since there is no face database containing this labeling, it had to be created manually. There are face databases capturing the same person with different emotional expressions, but the labeling still needs to be done manually.

Face databases are usually very large and the training process takes a lot of time. In this case the search for a compact face database and a fast training algorithm is not necessarily needed. The training process is executed only once in the beginning of the whole process in order to prepare the model for the prototype that will be reused with every prototype run. The face database is not deployed with the prototype; only the model is. Because of this, the hundreds of megabytes and execution time that is counted in minutes or even in hours are not significant. The important thing is the model size, since this has to be loaded with every program start and used in real-time. We focused on this aspect during the implementation phase. Since no serious problems are expected and it is too complex a matter to pre-calculate the size and the behavior of the model, it was tested directly while implementing the prototype.

2.1.6 Face Description

In this section we will examine the topic of describing human faces. The goal of describing human faces is to extract contextual information from purely visual representations of

[80]https://www.bioid.com/About/BioID-Face-Database, 21.10.2015
[81]http://www.imm.dtu.dk/ aam/datasets/datasets.html, 21.10.2015
[82]http://personalpages.manchester.ac.uk/staff/timothy.f.cootes/data/talking_face/talking_face.html, 21.10.2015
[83]http://www.milbo.org/muct/, 21.10.2015

the human face. There is much information obtainable from the face itself, such as a person's race, gender or approximate age. Further, images of faces can be compared for matching pictures of the same person, so that even identity can be tested by a facial description.

The work of Kohli et. al in 2011 [KPG11] describes an approach for age estimation from human faces. Here Active Appearance Models were used in combination with Dissimilarity Based Classifiers [KPG11]. For this task the appearance component is important when differing between a child and an adult. In this task the AAM is the clear choice since with Active Shape Models the solution would miss the appearance component. Conversely, for an emotion expression description, the shape of face features is the key, and the appearance component is not essential in such a task.

Another paper [DCLR13] proposed a method for measuring the heart rate from a facial image. Here the Active Appearance Model was used to recognize small movements and shape deformations of facial regions. A system consisting of 66 face interest points was used in this work (See Figure 2.7).

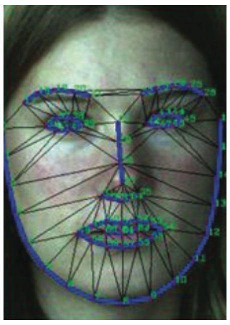

Figure 2.7: The 66 points of interest system used in [DCLR13].

In addition to the specific characteristics of race, gender, age or identity, more abstract contextual information can also be extracted from the image, such as a description of the emotional facial expression. This is much more complicated than the recognition of the specific characteristics, since the meaning of an expression can be misunderstood even by a human. However, recognizing the main emotional expressions in a robust, trustworthy way should still be possible. This aspect is the topic of this present work,

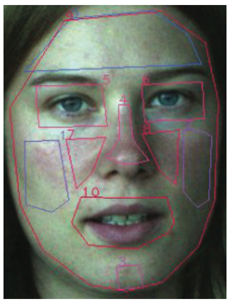
Figure 2.8: Facial regions of interest for measuring the heart rate from the face, introduced in [DCLR13].

and is described and implemented in the scope of the thesis.

We propose to work with the positions and movements of face features directly, since they indicate the expression in the most accurate way. For this, the suitable annotation (Interest Points System) for finding contours of all the important face features had to be found, and then specially adapted for facial expression analysis. This is described in the following chapter called Emotion Description.

For extracting the abstract contextual information from the face, the shape information is needed. For tasks such as race, gender and identity recognition, the shape is not very important, or even of no use. However, for tasks like emotion recognition it is essential and contains all the anticipated information needed for further processing. As described in previous sections, Active Shape- and Appearance Models are useful for obtaining shape information, although identity recognition is not possible from shape only. This means that when extracting the pure shape information from the visual information, identifying the human becomes impossible. Since the user remains unidentifiable after extracting the shape from the image, the method can also be used in communications in which the user can use his or her facial expressions while remaining anonymous.

Another domain of use is Visual Speech Recognition, in which the shape of the mouth can be used for recognizing what people are saying. If the shape of the mouth is described precisely, the differences between the pronunciations of various vowel and consonant sounds can be trained and classified.

Emotion Description

This subsection deals with the idea of describing emotional expressions from an image of a human face. As mentioned above, the basis for the recognition is the shape obtained from the image. This shape is then analyzed and fitted to a pre-trained model of facial expression shapes. The fitting process can be performed in various ways. One approach could be to compare static shapes as a whole directly with the pre-trained face models. Another would be to segment the face shape into facial parts of interest and to fit them separately, combining the results for an overall expression description.

A quite different approach would be to use the differences between several recent shapes in a video stream instead of simply matching static shapes. Shapes can be saved in a buffer containing a number of recent expressions, where the shape and time differences can be compared. This seems like an approach containing much richer information, but it should also be considered whether the additional information would really have any advantages over working with static images. Since emotion is recognizable from static photos of humans, it will be considered of little or no use to work also with the differences over time.

For listed approaches, several tools are required. First, a method for extracting the facial shape and the contours is required. An example has already been described in the above sections, and Active Shape and Appearance Model is a suitable solution. This step is essential for reducing the data from a huge amount of visual data to a much smaller amount of face shape data, which can be processed much faster and easier. Second, a classifier that can fit the shape to the emotion is needed. The shape data can be easily classified with any classifier since it is merely a small set of two dimensional coordinates that every classifier should be able to handle well.

Facial Action Coding System

The Facial Action Coding System (FACS) is a system describing facial expressions with *Action Units (AU)*, introduced by Ekman and Friesen in 1978 [Ekm78] [84]. Action Units used in the work are facial muscles that cause facial movements, resulting in facial expressions.

The Facial Action Coding System was developed to work primarily with video sequences. This system was developed especially for psychologists to observe human facial expressions. The idea was to mark Action Units manually on the video images instead of using an automated tool for fitting AU to the image. An extended system of 46 distinct action units was developed in the FACS article published by Bartlett, Marian Stewart, et al. [BVS+96] in 1996.

Our solution is inspired by this system. Instead of using Action Units based on face muscles, the Action Units in this thesis were based directly on face features visible to the human eye, such as the mouth, the eyebrows etc. Additionally, not all the Action Units proposed in the publication were used; the proof of concept comprises working with the six basic face emotion expressions (See Figure 2.11).

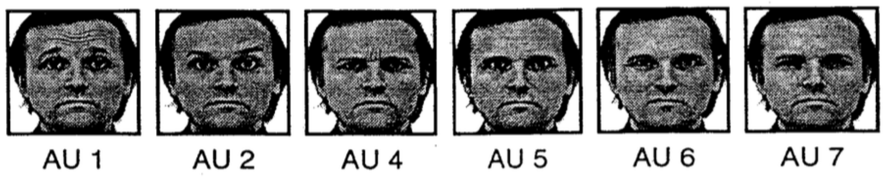

Figure 2.9: Example of six action units used in the study of Bartlett, Marian Stewart, et al. : AU 1 - inner brow raiser, AU 2 - outer brow raiser, AU 4 - brow lower, AU 5 - upper lid raiser, AU 6 - cheek raiser, AU 7 - lid tightener [BVS+96]

The problem of emotion expression recognition is the fact that even people often misunderstand human expressions. There are many slightly differing expressions that cannot always be classified with confidence. Using facial muscles as Action Units is

[84]http://ocw.mit.edu/courses/brain-and-cognitive-sciences/9-00sc-introduction-to-psychology-fall-2011/emotion-motivation/discussion-emotion, 21.10.2015

anatomically precise and correct, but unusable for our proof of concept. That is why we used different AU, as mentioned above (See Figure 2.10).

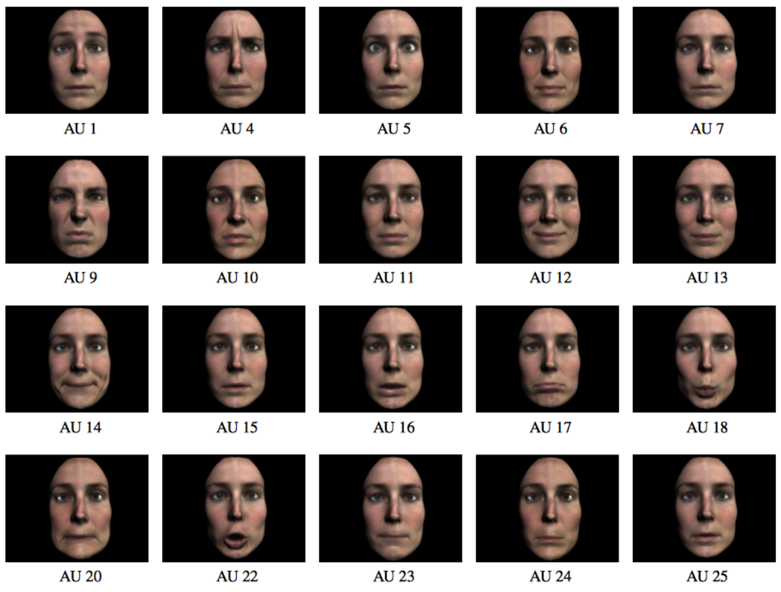

Figure 2.10: 25 Action units of the extended Facial Action Coding System introduced in the work of Cosker et. al in 2010 [CKH10]. Every Action Unit is a muscle that is used for facial expression. Some of the expressions shown can barely be differentiated from each other by the human eye, which makes it unusable for the proof of concept.

Even though the FACS contains multiple action units that can be combined, and offers a huge amount of different expressions with a high level of detail, in this thesis a simplified version (Figure 2.11) was used in both facial analysis (emotion recognition) and animation.

2.1.7 Classification Methods

In previous sections we discussed ways of describing the face and defining emotions. That was the preparation for the present section, in which methods for classifying the emotions are described. Since the facial image as pure visual information has already been pre-processed and the shape of approximately 60 points was extracted, the emotion can be now classified directly from the shape. The small number of points reduces the

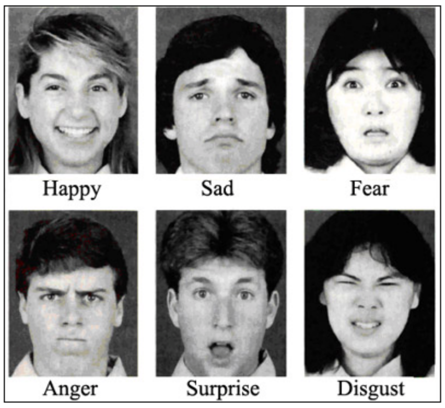

Figure 2.11: Six basic expressions described by Ekman and Friesen in 1971 [FACS Ekman][84]. These six basic expressions were used in the system designed within the scope of this thesis.

computational cost of the classification process. The simplicity of the shape model helps the classifier to work fast and to be sufficiently robust.

Since the classifiers need to be trained in order to be able to classify the input, both the training and the evaluation phase is discussed in this section. For the training process, the above mentioned face databases were used. Before choosing the right classifier for the prototype, both the training and the evaluation processes of particular classifiers were tested and observed.

In the training process the size of the trained model is the most important aspect to observe and to compare, since the model needs to be loaded each time the classifier is to be initialized. In our proof of concept, this happens each time the application starts. The bigger the size of the trained model is, the longer it takes to load each time in the evaluation phase. The computational costs of the training phase are negligible, since the time and complexity of the training algorithm does not affect the evaluation algorithm and has to be executed only while building the system. It clearly needs to be executed several times while testing, but the time spent on developing and testing was not considered a drawback.

Conversely, the computational costs are assumed to be one of the most important aspects of the evaluating algorithm, which is executed for each frame, making speed essential for an application running in real-time. In addition to the speed of the algorithm, the low failure rate is also considered to be a main aspect. These two factors need to meet our requirements, since a classifier is useless if it is either running at heavy computational costs or returning results with a high failure rate.

There are several machine learning methods already implemented in the OpenCV library, making it easy to test and compare different classifiers within one library. According to the *OpenCV Documentation* [85], the following classifiers are available in the *Machine Learning Library (MLL)*: Statistical Models, Normal Bayes Classifier, K-Nearest Neighbors, Support Vector Machines, Decision Trees, Boosting, Gradient Boosted Trees, Random Trees, Extremely Randomized Trees, Expectation Maximization, Neural Networks and MLData.

Despite the fact that the Support Vector Machines classifier was designed to work with and to predict two classes only, there was some progress in the implementation resulting in a multiple class classifier available in the OpenCV library. The OpenCV is based on the implementation of the C/C++ LibSVM library developed by Chang et. al [CL11].

Since we are classifying the shape model consisting of 66 points and containing X and Y coordinates only, the prediction itself is simple. This is why we sought for the simplest and fastest classifier with the smallest trained model, rather than a large and complicated classifier that would be too much for such a small quantity of data that was already prepared in previous steps. For these reasons the Support Vector Machines classifier was chosen.

Face Segmentation

As a subtopic of the classification section, face segmentation describes how to apply the classifier to the face shape. Having the classifier implementation ready for use, a proper way of training and classifying objects needs to be found. One of the decisions that needs

[85] http://docs.opencv.org/modules/ml/doc/ml.html, 21.10.2015

to be made is whether to train and classify the face shape as a whole, or to segment the face into face areas first and classify them separately.

In a real scenario humans recognize emotion from the overall facial expression. This is why the classification of the face shape as a whole is the obvious first approach. But is such classification also possible with a machine-learning approach? Classifying the overall shape should be straightforward, since not only the face features matter, but also the relative positions and the angle between them. It must also be considered that the ASM/AAM approach does not work 100% of the time. Sometimes, if the shape is not matched correctly at a particular place, some segments can corrupt the classification of the emotion: for example if the mouth indicates happiness while the eyebrows indicate sadness. If working with a non-segmented face shape, such small inconsistencies should not corrupt the overall facial analysis. That makes the choice of classifying the whole face as one object more appropriate than segmenting it into smaller pieces.

Some face features, for example the mouth, can often reveal emotion alone: the happy and the sad emotions can be recognized with the mouth only, with the other features hidden. For recognition of an angry emotion, however, the mouth alone is not enough. The mouth shape of an angry emotion is not specific enough; it can be observed with the same shape in other emotions. In this case the shape of eyebrows is also needed to reveal emotion. This is why, if a segmented face is used for the recognition, a system for describing and understanding the relations between shapes and the positions amongst the face features is needed. This would make the process of classifying emotion more complicated, less robust and slower, and would not be the ideal solution. For this reason also the face will be classified as a whole. The reliability of the result was observed and is described in the following sections.

Cultural differences should also be considered when thinking about classifying facial expressions. From a European perspective, the mouth is the most significant when it comes to recognizing facial expression, while in the Asian culture the eyes have greater significance. However, when classifying the face as a whole, the overall expression should - despite this difference - result in the same emotion in both cultures. When classifying segmented faces, a priority should be assigned to each feature in order to reflect the human way of understanding facial expressions.

2.2 Face Animation

Assuming that the face was analyzed and the emotion classified, the collected data should also be visualized in a useful way. The simplest way of outputting the data is the textual description that can be shown on the screen next to the face. Another solution might be to adjust the environment according to the recognized emotion, such as lighter colouring, or something similar. Nevertheless, for this thesis a more advanced approach is appropriate. As the proof of concept the recognized emotion was visualized on an animated avatar. Naturally, humans recognize emotion expression directly in the face, making it ideal for human interaction. As a side effect, when animating expressions on an avatar the identity of the user remains anonymous. A perfect solution for both human and machine is the combination of textual representation and animated avatar, in which the machine can work with the former and the human with the latter.

The problem with animating an avatar is the fact that the human face is a deformable object, which differs significantly from user to user. The chosen level of detail needs to be complex enough to be able to visualize the emotion in a realistic way and simple enough so that the avatar can match the face of any user. There are several animation methods listed below, but the main discussion is concerned with the differences between two and three dimensional animation, since these technologies differ substantially. The advantages and disadvantages of one or the other methods are listed below.

2.2.1 2D Animation

The easiest way to animate emotion in this thesis was to directly visualize the debug information of either the fitted Active Shape Model or the fitted Active Appearance Model. Since this model contains enough information to recognize emotion, fitted points can be drawn as a visualization of the face and the emotion should be still recognizable by the human eye. However, this is not a real animation of a human expression since the level of detail is as low as possible. Nevertheless, this approach was used for debugging and testing because of its simplicity and low data quantity, which makes it perfect for such usage.

One method for animating the data adequately could be to use *Scalable Vector Graphics (SVG)*. The visual information here is saved as a set of vectors instead of a mesh of pixels. Thus with the shape of the face recognized from the facial analysis, the vectors that build the SVG image can be easily adapted by translation or rotation in order to simulate the mapped shape. In other image formats, for example BMP format, in which the visual data is saved as set of pixels having visual information for each pixel, any deformation is too complex, making this format unusable for the task.

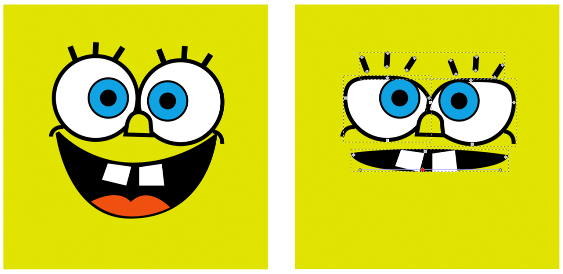

Figure 2.12: A picture of *Sponge Bob* in the SVG Format. On the right-hand side is the deformed face. The deformation was not applied to the graphics directly but to the pivot points, which were moved and the angles adjusted. The result of this change applied to the SVG file is a deformation simulating an angry expression.

Flash is an alternative for animating deformable objects, with *ActionScript* scripting the animation. However, in comparison with SVG Flash is too complicated for use. We would need a special development environment to support external applications affecting objects inside of the environment, while the SVG file format is open and can be adapted as a plain text file.

2.2.2 3D Animation

Animations in 3D are much more complex than those in 2D. They contain more details, can be seen from multiple viewpoints, the lighting can be adjusted and set, and much more. All these details create more realism, but this also has a disadvantage in the form of computational costs. A 3D animation also has different requirements, different approaches and different environments to work with.

To be able to implement such a system, a working 3D editor is needed. This editor would need to be capable of loading rigged 3D models and of offering an API with which the 3D model can be accessed and manipulated from outside. In order to animate the expression precisely and in an optimized way, a rigged model is used. With this there is no need to deform the mesh directly; all that is required is translation and rotation of the joints to deform the face in the desired way.

The *Unity Editor* meets all the requirements of a suitable 3D editor. It can load rigged models in different file formats, such as *.blend* or *.max*. However, the rig information is not always imported correctly. This can happen with both file formats, so we can assume that the problem is on the Unity site. However, importing rigged *.blend* models tends to fail less often. The import functionality is essential here, since the Unity Editor does not offer the environment for advanced 3D modeling. Therefore the only way is to create the avatar with an external modeling tool such as *Blender* or *3DS MAX* and export it for import into the *Unity Editor*. If modeling in 3DS MAX there is a special plug-in for this program called *Morph-O-Matic*, made especially for 3D rigs and animations of faces. However, the library is not free, and since *3DS MAX* does not offer the functionality required for our prototype listed above, the model would need to be exported into the Unity Editor. Unfortunately, there is a loss of quality while exporting and importing, defeating the purpose of using this library for the task.

A further consideration when choosing an environment is the capability of the API offered by the editor. Since the proof of concept for analyzing facial expressions was written in C++, an interface for cooperation between the Editor and C++ applications is essential for the prototype. The amount of shared data is not huge, only around 60 coordinates as well as emotion flags to be transferred. However, since this needs to occur in each frame, the interface should be able to operate at sufficient speed.

One important aspect when animating facial emotions is the *Neutral Face Expression*.

This is the user's facial expression without any emotion shown, and with all the muscles in their relaxed position. This expression is important, especially in the animation process. Every human has different facial proportions that need to be fitted to a general model that is the same for every face. For this reason the animation model needs to be calibrated for the actual user's facial proportions. This has to be done with the user's emotionally blank facial expression so that no emotion will be traced. We will take the distance between the eyes and the eyebrows for an example. Since the animation works with the shape of the features and the distances between them, these need to be calibrated. If user A has a one centimeter distance between the eyes and the eyebrows and an user B has a two centimeter distance, then the emotion animated on the same model would appear as if user B shows some emotion even in his or her neutral expression. This is why these distances need to be normalized for the animation model. Following this it will be clear whether the eyebrow is really raised or not. The same needs to be done with, for example, the mouth, where the calibration of the mouth's width is essential for differentiating between the neutral expression and a smile.

In this thesis dealing with animation with a high level of detail was anticipated. This is why the 3D animation of an avatar was chosen. The avatar is also a 3D model of a human head with face deformed over time in order to simulate the emotion. The deformation is based on normalized positions of face features obtained from the ASM/AAM model, and normalized according to the neutral expression and applied to the head model. This approach results in the most realistic animation of all the methods listed in this section.

2.3 Related Work

Before commencing on this thesis, some related work was researched in order to gain an overview of existing approaches. Experiments and attempts were observed in various publications and the experiences from their successes and failures were gathered. The publications did not have to be associated with our topic directly, but they did need to use at least some of the methods that, with some modifications, can be adapted to our needs and used for our purposes. The parts of publications listed in this section have served as an inspiration for building the basic functionality of the prototype.

Earlier publications show different approaches in how to deal with topics similar to our own. The work published by Michael P. and El Kaliouby R. [MEK03] (2003) deals with the classification of six basic expressions. 22 facial features were located and classified with Support Vector Machines. Unfortunately, they only attained a total accuracy of 60.7% when training and testing with six users.

Another approach was used by Azcarate, A. et. al in their work [AHvdSV05] in 2005. The Face Detector based on the Haar Classifier and extended with Piecewise Bezier volume deformation was used for detecting faces and locating face features. In the second stage Naive-Bayes and TreeAugmentedNaive-Bayes classifiers were used for emotion classification. The Cohn-Kanade face database containing seven expressions (six basic expressions and one neutral expression) was used as a resource for training and testing the algorithm. In this work approximately 65% accuracy was attained in tests, where for the test set, samples of other people than for the training set were used.

In 2007, Datcu D. et. al published an interesting article [DR07] using Active Appearance Models as a basis for emotional expression analysis. This approach can extract important face features that can be analyzed further. The work used Support Vector Machines for classifying expressions of still pictures and video sequences. In the case of video sequences the attained accuracy was 79.62-88.67%.

The publication of Ari I. et. al [AUA08] (2009) aimed at using the face feature analysis in sign language, showing yet another method for a face interest points system. Face features were located with an Active Shape Model Tracker using 116 points, and classified with Support Vector Machines. In addition to frequently used Support Vector Machines, there are also other methods for classifying face features fitted to an Active Shape Model. The article by Milborrow S. and F. Nicolls [MN14b] proposed, in 2014, a solution with SIFT descriptors matched by the Multivariate Adaptive Regression Spline (MARS). However, this work focuses on the computational efficiency rather than the failure rate.

Nedkov S. and Dimov D. used in their work [ND13] Action Units from the Facial Action Coding System introduced by Ekman P. and Friesen W. V. [Ekm78]. The facial dynamics of emotional expressions were analyzed and classified with a *Linear Discriminant Analysis*. Their experiments showed 75% precision of the classification results.

A different method was introduced in the work of Sun X. et. al in 2009 [SRDW09], where the visual muscle activity was analyzed by *Vector Flows*. The muscle activity was then classified according to the six basic facial emotions proposed by Ekman P. [Ekm78]. For the classification process, the *Bayesian Network* was trained on the Cohn-Kanade Face Database. For locating face features Active Appearance Models were used, and the classification of Action Units reached the overall rate of 90%.

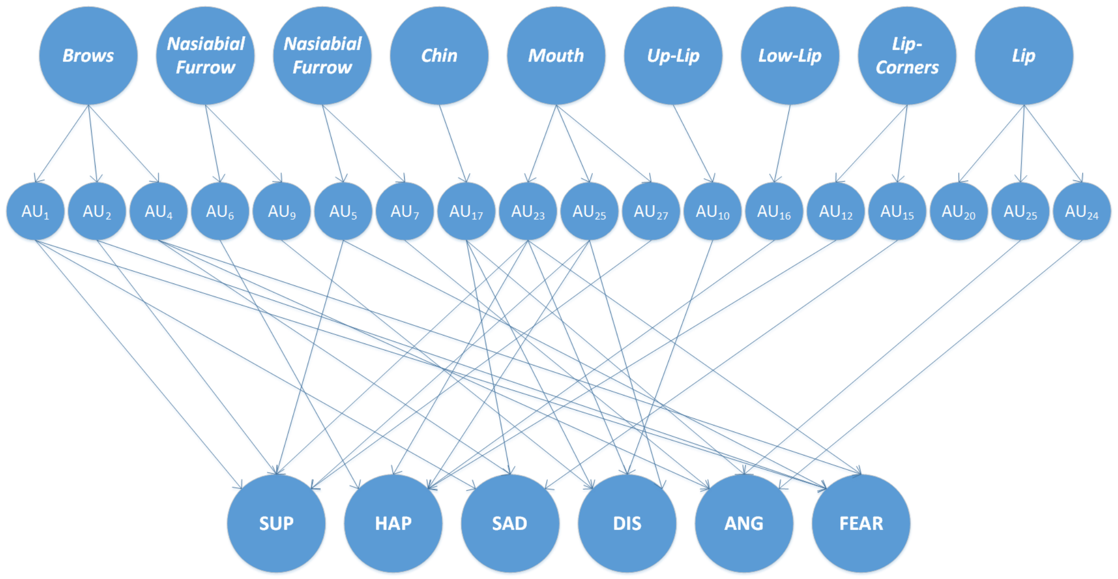

Figure 2.13: 3-layered Bayesian Model used in the classification method in the publication [SRDW09].

The publications listed above offer a comprehensive overview of possible approaches for dealing with the problem of the emotional analysis of human facial expressions. Each method has its benefits but also drawbacks, which is why no perfect solution has been found. Nevertheless, these articles have helped to build the basis of this thesis, since the approaches and experiments described in these publications influenced the choice of methods to use.

CHAPTER 3

Project Description

This chapter describes the functional part of the project. In addition to the goals and the anticipated, the structure and implementation process is also listed and described here.

3.1 Goals

The main goal of this project is to implement a robust system capable of recognizing emotion in the human face. The captured image of the user's face is analyzed, the facial expression animated on an avatar and a textual description of the recognized emotion is visualized.

The following goals were set:

1. *Robustness.* The prototype needs to be robust and resilient against poor input and conditions. If the minimal lighting conditions are not met, then no face is recognized and there is no output. The user's input possibilities are limited to facial expressions and head positions only, which is why there is no possibility of inputting the wrong data an crashing the system. However, as mentioned in previous chapters, the face will be recognized properly only if it is facing the camera directly. Otherwise there is a high probability of mismatching the face.

2. *Fully Working Prototype.* The goal is to implement a fully working prototype that proves the concept by showing a working solution to the problem. The whole product was developed in a way that allows any other developer to easily adapt the code to his or her needs. In this way the prototype can be enhanced, or it can serve as a testing ground for methods used in the implementation.

3. *Performance.* Another goal was to reach optimal performance, making the prototype viable on commonly used laptops and computers without high computational power, and using only common web-cam devices.

4. *Automation.* The system also needs to be easy to use and fully automated, so that the user does not have to set up any specialized settings or run parameters in order to make the product work properly.

3.2 Challenges

The project faces several challenges:

1. *Face Interest Point System.* The first complex problem is the system of face interest points. This needs to be defined well so that the emotion can be optimally read from the system of points, and an algorithm that fits the points to the facial images is also required. Optimal count and location of points is also important, enabling it to run in real time while matching the face with sufficient precision. The shapes of human faces differ extensively, so the system must be able to fit a wide range of faces properly. During facial expressions the shape of the face deforms greatly, so the algorithm needs to be able to match the face in various deformations.

2. *Expression Interpretation.* The second challenge is the problem of recognizing emotion from facial expressions captured by the web-cam. Besides technical complexity, the diversity in how people understand facial expressions is also a significant aspect. There are many facial expressions, the meaning of which can be understood differently by different people, because emotion is not always self-evident

and is contingent upon personal experience. People from different cultures especially, or people of different ages, understand certain facial expressions differently to others. If there were emotions present in the face that could not be unambiguously recognized by other humans, then it would be impossible to build a system of explanation for all emotions that works perfectly all the time.

3. *Libraries.* Since one of the main non-functional requirements for the prototype was that it should be able to run on a personal computer without any special hardware or computational power needed, we had to build an optimized prototype precise enough on the one hand but operating fast enough on the other. The problem with libraries is that they are either very precise but not able to run in real time on a common computer, or they are fast with low computational costs but are not robust enough and therefore unreliable.

We had to overcome all the challenges listed above in order to create a valuable prototype.

3.3 System Design

3.3.1 Overview

Before the implementation started, we searched for robust, well working libraries that can be used in the prototype. Multiple implementations of different methods were found, and all were tested, the results being described in Section 4. From these tests we selected the most suitable libraries that work well and used them in the implementation.

The following diagram (Figure 3.1) shows how particular libraries and modules work together and how the input data flows through them. The user's face is first captured by the web-cam, which is controlled by the OpenCV library. The captured image is then sent to the external ASM/AAM Library, where the ASM/AAM Model is fitted to the user's face. In our prototype we used the FaceTracker ASM Library. The fitted model is required in both the emotion classification and the rendering part; therefore it is sent to both of them. For memory saving purposes we do not create a copy of the model, but rather the model is first used in the classification code and then reused for avatar rendering. These parts do not influence each other, and they are therefore split in the diagram.

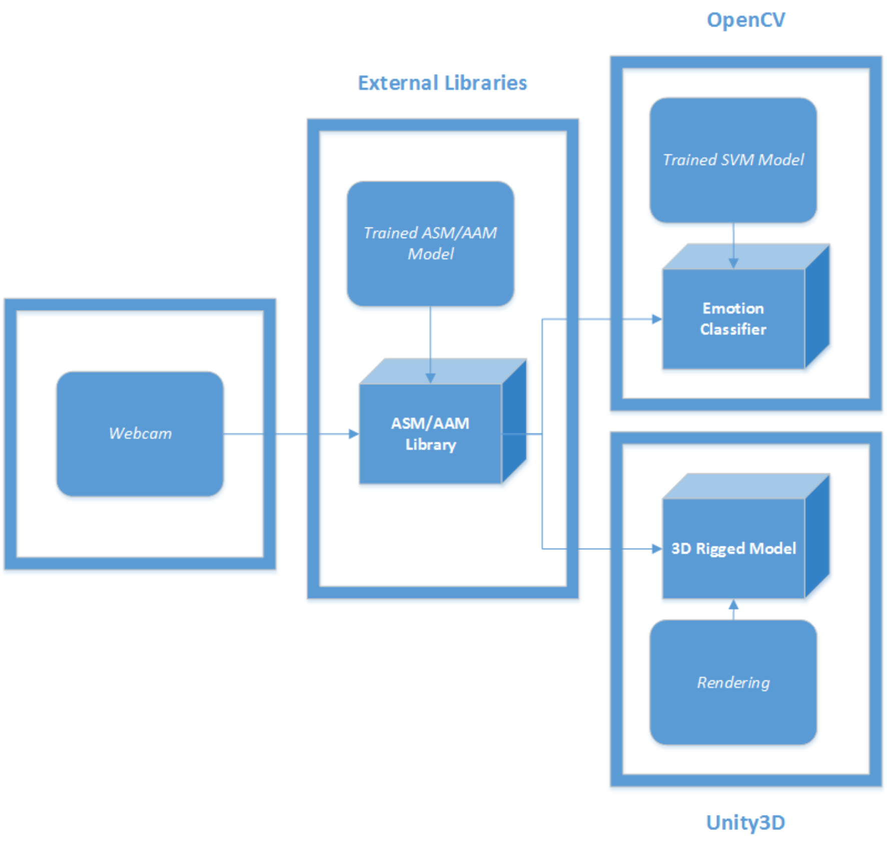

Figure 3.1: The logical parts of the project divided into stages, showing which libraries and tools were used in each stage.

In order to enable the application to analyze the input data, we need to train and prepare two models: one for fitting the points of interest to the image and one for the emotion analysis. The training code we used for building both of these models was provided by the libraries used in the implementation. There was no need to build our own algorithms to be able to train the models, as some libraries are equipped with fully working pre-trained models that can be used directly. Nevertheless, a face database is needed for the training phase, as already mentioned. In this thesis we tested several of them when implementing the prototype, and details can be found in Section 4. As a result of the training phase an ASM/AAM model for fitting face features and a SVM model for classifying emotions were generated. These models are loaded into the runtime and used in a loop over time, which is used for processing the input data.

We were constantly testing the system while implementing or trying out new approaches in order to see rapid results. We were able to create automated tests for emotion classification only, preparing the images with captured emotional expressions, together with a label of the emotion so that the results could be compared algorithmically. Testing the animation is more complicated since it strongly depends on the user's feeling, which cannot be tested automatically. This is why we tested it by observing users using the prototype. We tested it while implementing the prototype for refining, and when the project was finished we also performed tests of the final product among a small group of people. The final test results are described in Section 5.

The following sections describe the input and the output of the prototype.

3.3.2 Input

As soon as the project was prepared it could begin to read and analyze images of human faces. The user's face is captured by the web-cam and the visual information is processed further. The fitting algorithm of an ASM/AAM library is implemented, fitting the face features trained in the model to the captured image. The facial features define the shape of the face, so that by having them localized the shape analysis can begin. The trained SVM model is loaded by the SVM classifier implemented in OpenCV, based on the libSVM. The implementation works robustly, but there are still some minimal requirements yet to be met. The lighting conditions need to be sufficient to recognize the face with its features on the captured image. The user must sit in front of the web-cam in such way that his or her whole face is captured. If the face is only partially present or captured only from a side view, the emotion cannot be recognized reliably, since in such cases both the fitting algorithm and the emotion classifier become stuck at their minimum.

3.3.3 Output

Having the input data processed and the results available, the prototype needs to present the data to the user. The collected data from the input analysis are visualized on a 3D avatar and a textual description of the classified facial expression is displayed. The 3D avatar represents the facial expression of the user and is adjusted so that the avatar's face features correspond to the user's, creating a mirror effect of the user's facial expressions. This visualization is useful only when the program is running in real-time. However, the textual description of the classified facial expression displayed alongside the face does not require to be updated in every frame in order to work and to look realistic, hence some latency here is acceptable.

CHAPTER 4

Implementation

4.1 Overview

This chapter describes the overall implementation of the prototype, including the process of testing and comparing technologies and libraries before the implementation could commence. Problems and their solutions are also dealt with in this chapter. A graphical structure of the project can be seen in the following flowchart (Figure 4.1), and a detailed technical view of the communication between individual parts can be found at the end of the chapter.

4.2 Face analysis

Deep facial analysis is the main problem of this thesis. The first challenge was to define and localize those face features important for emotion recognition. There are various libraries that can describe the shape of a deformable object that are also suitable for the task of describing the human face. An efficient tool for describing the shape of the human face and localizing its features is essential for facial expression analysis. To find the optimal library for this task is also challenging, since published libraries are mainly either fast but imprecise, or work precisely but are too slow for real-time usage.

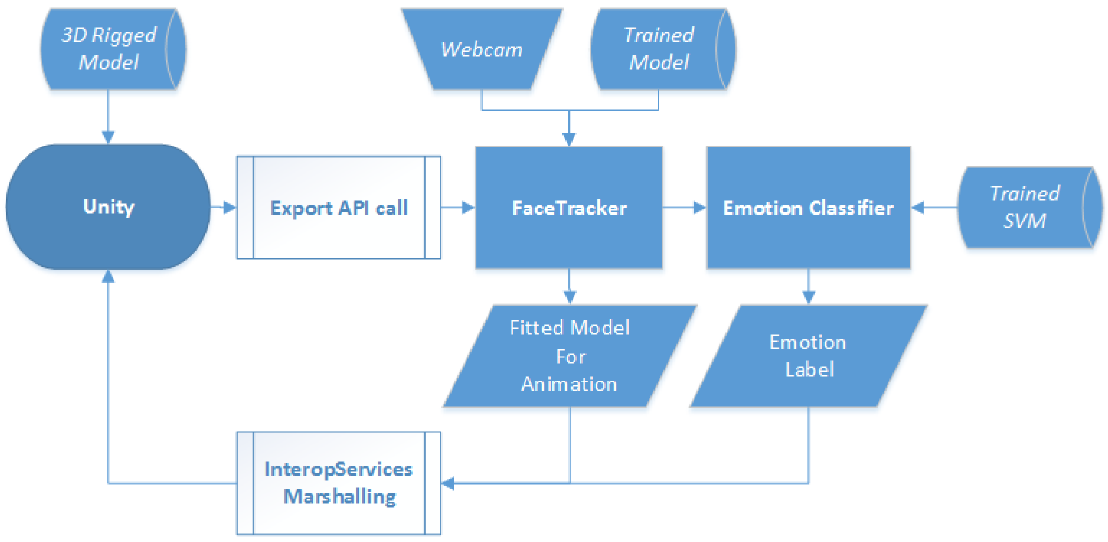

Figure 4.1: Flowchart showing how the data is processed in the prototype. The starting point is the Unity box on the left side. It requests data from the C++ program containing the FaceTracker library trough *Export API (application programming interface) call*. The FaceTracker library loads the pre-trained ASM model and fits it to the image captured by the web-cam. The C++ program that is hosting FaceTracker then sends the image further on to the Emotion Classifier, which classifies the emotion of the face in the image and returns the label of the recognized emotion. Both the fitted ASM model and the emotion label are then sent back to Unity, where the data is visualized.

However, there are many published libraries that use different algorithms, different face points systems and different face databases. Since we require a fast and precise working library, we selected the best methods available and tested them. The results of these tests are analyzed in this chapter.

There are two main aspects to be considered concerning facial analysis.

- The first is the shape of the face features, since, for example, the shape of the mouth or the eyelids are important in emotion recognition. In this thesis shapes are trained and classified by Support Vector Machines. We first trained an SVM model of facial feature shapes that could be used for classifying new shapes by comparing them to the pre-trained shapes of emotions.

- The second aspect are the locations of face features, because not only the shape

> of the features is important but also the angle and distance to other features have their significance. Here we can take eyebrows as an example. Eyebrows do not substantially change their form; usually it is the angle and position that is significant. Anger, surprise and fear depend upon heavy eyebrow positions and angles. Raised or lowered eyebrows also have their particular meanings.

However, since the user can freely move the face and thus scale or rotate it, the location of the eyebrows cannot be used directly. The face needs to either be normalized and centered or some of the pivots need to be used. We have chosen to implement a solution using pivots. For measuring raised or lowered eyebrows the absolute position cannot be used and so the relative position to a pivot is needed. Ideally, the pivot is a static non-deformable feature. We have chosen the nose in the implementation, since it is located in the centre of the face and only small deformations and movements are possible. When measuring the relative positions of the features on a freely movable face, scaling should also be considered. Assuming that the user is looking directly at the camera, he or she can still move forwards and backwards. If moving towards the camera, the overall face becomes larger and so do relative distances. This is why we took multiple pivots and recalculated the relative position of the facial features according to the distances between them.

A Support Vector Machine was trained and used for classifying localized face features. The problem here was again the user's movement, where the face is rotated and scaled, which changes the positions of the face features (See Figure 4.2). The classifier cannot handle such position changes itself, so a solution for this problem had to be found. We took three pivot face features that are not deformable and do not change their positions. As mentioned above, the first was the nose, a static feature in the centre of the face. The left and right cheekbones were chosen as second and third references. According to the relative positions of these references, the rotation and scaling of the face can be calculated, and this enables the rotation and the scaling of the face back to the initial setup.

To make the classification process even more robust we ignored contour features,

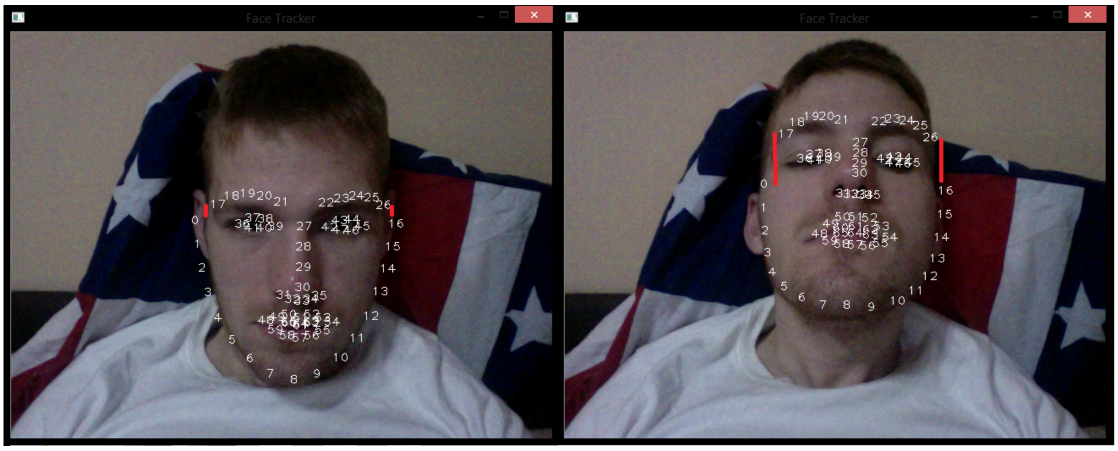

Figure 4.2: This figure shows the repositioning of face features when capturing the user's face with the same expression from different angles.

which are unimportant for emotion classification. As result only the eyebrows, eyes, nose and mouth approach more closely to the classification. The chin and the cheeks were omitted in the classification phase since they do not contain information important for emotion recognition. They nevertheless still need to be localized since they are important points for the normalization of the model. With this approach we attained an optimized and robust way of classifying emotion.

4.2.1 Static vs. dynamic analysis

The method of processing the input images is an aspect that also should be considered. Input images can be either processed independently, one after other, or a certain relation between separate inputs in the stream captured by the web-cam could be investigated. As already mentioned, emotion can be recognized either statically or dynamically. The study [SSP$^+$12] examined whether the duration of smiling is also significant. However, we were analyzing the static input when recognizing emotions in our prototype. Measuring the duration of a happy expression was beyond the scope of this thesis, although but it would be easy to extend the prototype to embrace this functionality in future work if somebody should wish to research this topic. Nevertheless, the deeper analysis of emotion remains more pertinent to psychologists than to ourselves.

Some libraries tested in the scope of this thesis also use facial tracking over time as a performance enhancer. This is a clever approach that results in improved performance, while not losing precision. FaceTracker, the library used in the final version of the prototype, works in this way. The library is unusably slow when it loses its tracking and starts searching for the face. As soon as it locates a face, the performance increases rapidly. This is one of the reasons why we have chosen this library.

4.2.2 ASM / AAM Libraries

For implementing the prototype, we have chosen to use a library that implements ASM/AAM instead of writing our own implementation. There are a lot of different ASM and AAM implementations that could be used for the prototype. Before we began work on the implementation we tested the existing libraries to see if they are usable for our purpose, and compared them so that we could choose the best one. The main properties that we tested are performance, precision, models offered by the library, the complexity of integration in our project, and the complexity of the training process. An open-source solution was advantageous, since the code of an open-source solution can be directly adapted to our needs.

We have tested following ASM and AAM libraries:

DeMoLib

DeMoLib [116] is a complex implementation of Active Shape and Appearance Models created as the PhD project of Dr. Jason Saragih and supervised by Dr. Roland Goecke. It was later extended by other students of the University of Canberra. It is a cross-platform library written in C++, the sources of which are available for non-profit academic purposes only.

AAM implementation is based on the work of [CET98] and offers various fitting algorithms, such as *Di-linear*, *Project-out inverse compositional*, *Simultaneous inverse com-*

[116] http://staff.estem-uc.edu.au/roland/research/demolib-home/, 21.10.2015

positional, *Robust inverse compositional*, *2D+3D inverse compositional*, *Original fixed Jacobian*, *Linear iterative discriminative* and *Nonlinear iterative discriminative.*

The library also implements Active Shape Models. However, we did not test these since the required *connectivity file* could be found neither in the library nor in any of the face databases. There is no description of how the file structure should look so we were unable to create the file.

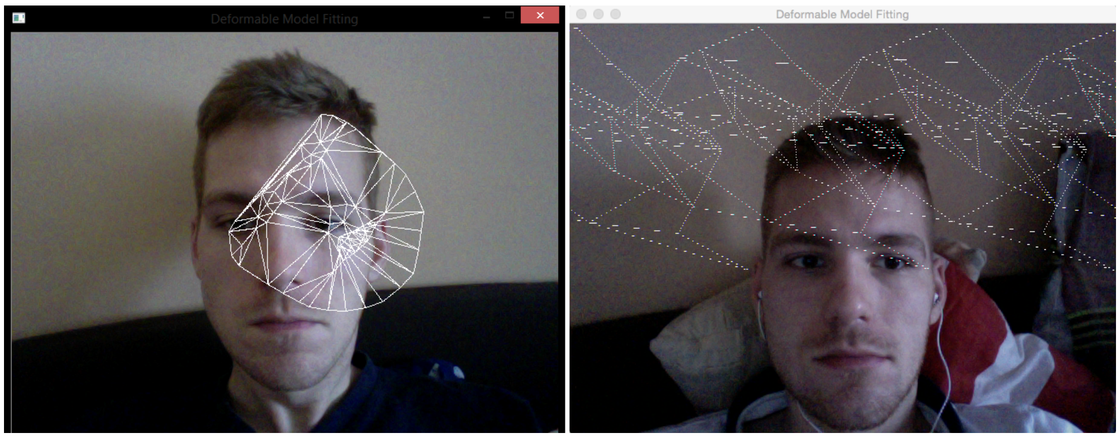

Figure 4.3: This figure shows the results of testing the *Di-linear AAM* fitting. Other fitting algorithms implemented in this library were tested on both Windows and OSX operating systems, attaining similar results.

The implemented fitting algorithm is fast, but the results of the testing showed that the library is not stable and does not work precisely enough (See Figure 4.3). Therefore it was not used in the prototype.

Vision Open Statistical Models

The Vision Open Statistical Models (VOSM) [100] library is another cross-platform, open-source implementation of Active Shape and Appearance Models. VOSM offers *1D profile ASM*, *2D profile ASM*, *Direct Local texture contrained (LTC) ASM*, *Basic AAM*, *Inverse compositional image alignment AAM (ICIA)* and *Inverse additive image alignment AAM (IAIA)*. There are also settings and run parameters prepared for training models on

[100]http://www.visionopen.com/downloads/open-source-software/vosm/, 21.10.2015

BioID, *IMM*, *FRANCK*, *AGING* [104], *XM2VTS* [105], *EMOUNT* [106] and *JIAPei* [107] face databases in the library. Even our own face database can be used with this library, which makes wide training options possible. However, the problem with this library is its performance. The implementation runs on high computational costs, making it unable to run in real-time on a standard personal computer. This makes it unusable for our prototype.

Face Tracker

FaceTracker [115] is a library that implements Active Shape Models and uses them for human face tracking. It was written by Dr. Jason Saragih, the author of the above mentioned DeMoLib library. In comparison to the DeMoLib library, FaceTracker is much smaller, since it only offers one implementation of ASM. The library is optimized, runs well in real-time and contains precise pre-trained models that are about 1MB large, which makes it a perfect candidate for our prototype. The testing showed that the library is stable and provides precise results (See Figure 4.4).

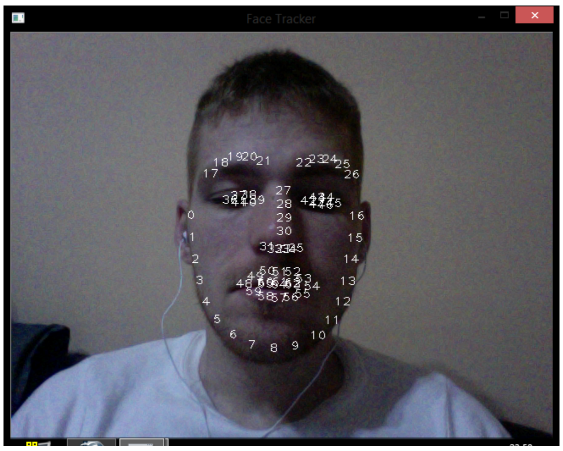

Figure 4.4: This screen-shot shows the exactness of the fitting algorithm implemented in the FaceTracker library. All features were located correctly.

[104]http://agingmind.utdallas.edu/facedb, 21.10.2015
[105]http://www.ee.surrey.ac.uk/CVSSP/xm2vtsdb/, 21.10.2015
[106]http://www.emount.com.cn/
[107]http://visionopen.com/cv/databases/JIAPEI/, 21.10.2015
[115]http://jsaragih.org/, 21.10.2015

As an optimization, the library continuously tracks the user's face. As soon as the face becomes lost the library attempts to re-initialize the tracking algorithm. We recognized the re-initialization in the prototype and used it for re-initialization of the animation part.

The only drawback with this library is the missing training part. It was also implemented by Dr. Jason Saragih, but the source code is no longer available since it is the property of the *Carnegie Mellon University* [108]. Because of this there is no possibility to train our own models. However, the pre-trained model works as desired and so we tolerate this drawback. There is just one small problem with fitting a face to a mouth that is wide open, but this is an unnatural expression and therefore not a serious problem.

AAM / ASM Library

Yao Wei published in his Github repository [109] his implementations of Active Shape and Appearance Models, providing source-code for both training and evaluating.

It was interesting to observe these libraries working. The precision of the fitting algorithm is not bad, but still not accurate enough. There is also one serious problem. When working with dynamic images the fitted model shakes from frame to frame, even when the face does not move or change. With each fitting loop, the model is slightly different, and this causes a shaking effect (See Figure 4.5). This makes the library unusable for the avatar-rendering part, since it would damage the whole animation.

STASM

STASM [110] is a C++ library based on the Active Shape Model introduced by Tim Cootes. The library was developed and introduced by S. Milborrow and F. Nicolls [MN14a]. However, this library was considered for use with static images only. We adjusted the library to work with dynamic input from web-cam, but the fitting algorithm is too slow for use in a real-time application.

[108] http://www.cmu.edu/index.shtml, 21.10.2015
[109] https://github.com/greatyao, 21.10.2015
[110] http://www.milbo.users.sonic.net/stasm/, 21.10.2015

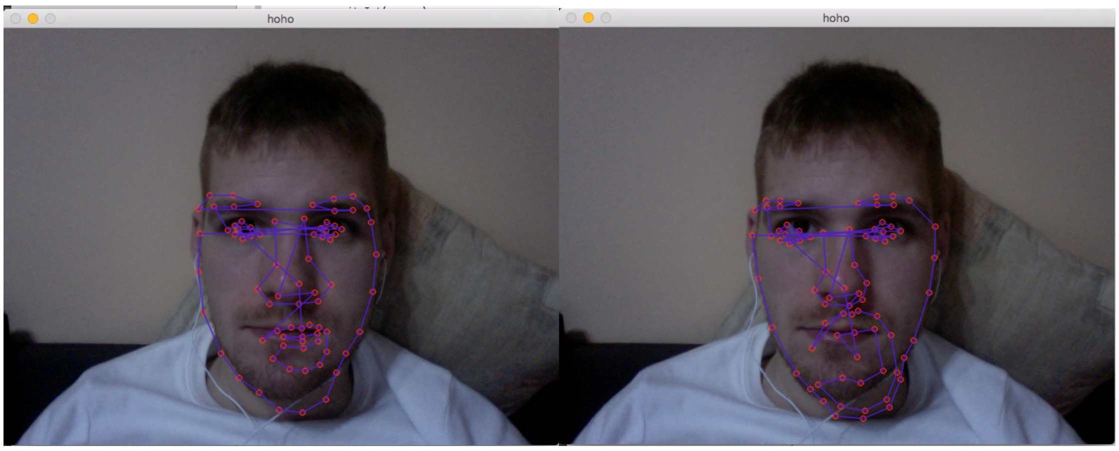

Figure 4.5: Two frames with the same face pose but with different results of the fitting process. Such differences happen with each loop, causing strong shaking of the model, making the library unusable for our prototype.

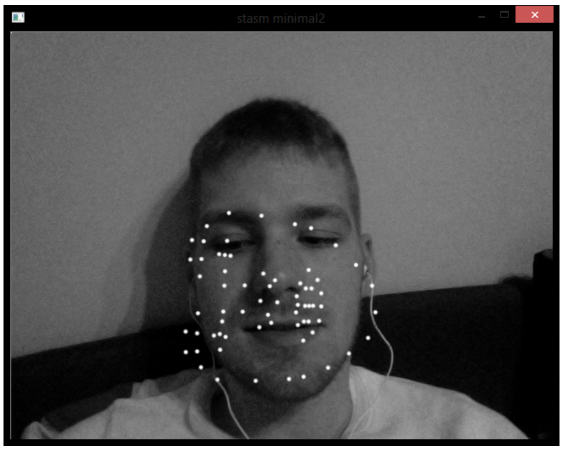

Figure 4.6: STASM library testing. It is impossible to use this library in a real-time application running on common hardware. The precision of the fitting algorithm implemented in this library is also too low. Notice the inaccuracy of the located features.

Others

We also wanted to test the *Candide* [111] and *AAM-API* [112] libraries, but the outdated code and lack of good documentation made the compilation of these libraries impractical.

[111] http://www.icg.isy.liu.se/candide/, 21.10.2015
[112] http://www.imm.dtu.dk/ aam/aamapi/, 21.10.2015

4.3 Face rendering

As already mentioned, a 3D avatar is animated as a visualization of the output. For this purpose we used the Unity3D Engine with C# scripts for handling the avatar animation. The Unity3D Editor provides options for importing *Blender's* [113] .blend file format. There are several forums offering many freely downloadable 3D models in .blend format, some of which are already rigged. The advantages of using ready-rigged 3D models were described in Section 2.2.2. However, the import of Blender files into the Unity3D Editor does not always handle the rigs well, causing many models to fail to be imported correctly. We finally located two models that could be imported correctly and we used them both while implementing the prototype.

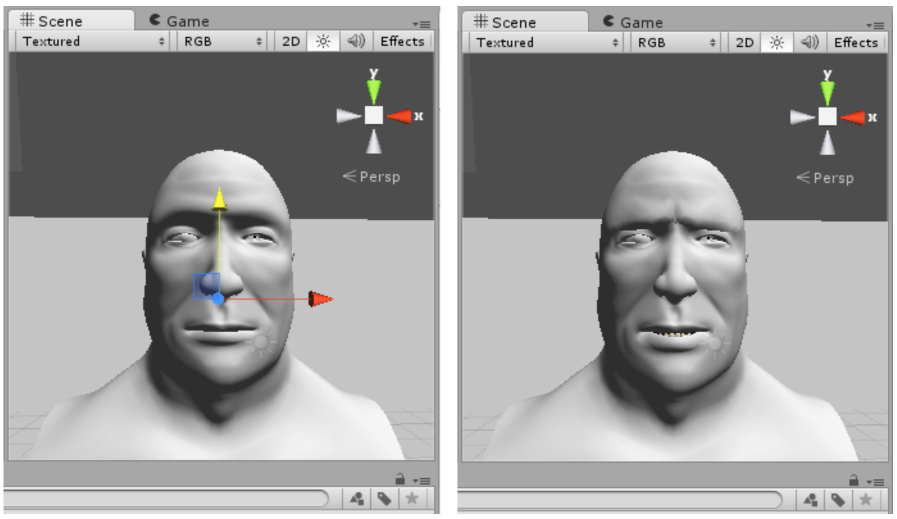

Figure 4.7: The *Holmen advance head rig* is a 3D rigged model downloaded in the .blend file format. It is a highly detailed model with spectacular texture. However, the texture cannot be seen on this screen-shot since the Unity3D Editor was not able to import it.

As mentioned in Figure 4.7 [101] and 4.8 [102], the Blender models lose their quality when imported to the Unity3D Editor (See Figure 4.9).

[113]http://www.blender.org, 21.10.2015
[101]http://www.blendswap.com/blends/view/48717, 21.10.2015
[102]http://www.blendswap.com/blends/view/2389, 21.10.2015

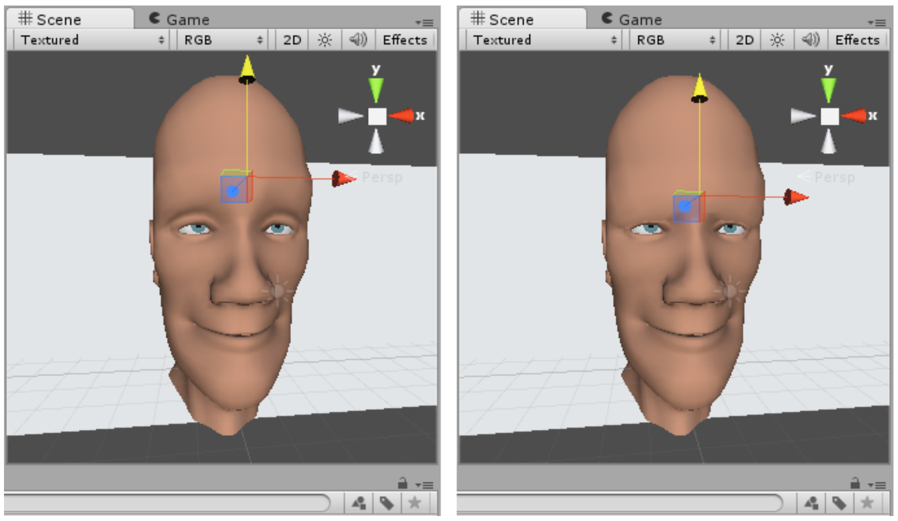

Figure 4.8: The quality of this *Holmen Face Rig* model is the same as the quality of the previous one shown in the Figure 4.7, but the superiority of the loaded material makes the model look more realistic. Both models are equipped with facial muscle joints, making them suitable for our prototype.

Since the joints of different rigs are also positioned differently to match the particular model correctly, our script settings also have to be adjusted when changing the 3D model. The Unity3D Editor offers a user interface for setting parameters for C# scripts used in the project. This makes the adjustment of script variables to the new 3D rigged model easier, with no need to change the source.

When working with a purely non-rigged 3D model, animating facial expressions becomes too difficult. In this case the mesh needs to be deformed manually, which means to synchronize and transform multiple vertices into the desired deformation. Since this approach would be too complex, we worked with rigged models (See Figure 4.10). We found that a suitable approach was to use a rig for deforming the mesh in order to deform the face and simulate a facial expression.

A system for mapping the coordinates of face features onto the 3D model had to be implemented. The input from ASM/AAM is a list of coordinates of the fitted model, which have to be converted before applying them to the 3D model rig. The conversion was implemented in a C# script in Unity3D, which first maps features to joints and

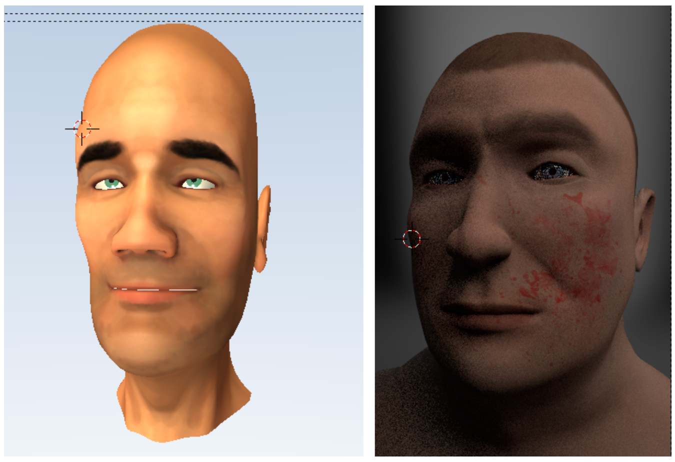

Figure 4.9: A screenshot of 3D models shown in Figures 4.7 and 4.8, opened in Blender before being imported into the Unity3D Editor. Notice the texture difference. However, we wanted to use the model in a real-time application made in Unity3D, which would be too complicated to render in such level of detail on commonly used hardware.

then recalculates the translations that need to be applied to particular joints. Since the ASM/AAM model has different proportions than the 3D model, the ASM/AAM feature locations cannot be applied directly but have to be recalculated to joint translations, causing feature shifts on the 3D model. We calculate the ratio between the ASM/AAM model and 3D model proportions at the initial stage, capturing the neutral expression. We first calculate the shifts between the current and neutral feature locations in the ASM/AAM model, which are then applied to the 3D model joints in neutral positions. If the user's face becomes lost, both the ASM/AAM library and the 3D model neutral expressions are re-initialized to avoid any mismatch between the neutral expressions of the ASM/AAM and the 3D model.

Sometimes, when the captured face moves too fast, FaceTracker returns incorrect data, causing a shift drift on avatar's joints. In such cases the system should be re-initialized in the same way as it is in case of losing the user's face. However, it is much more complicated recognizing this occurrence than recognizing the face loss itself, since FaceTracker is not aware of delivering incorrect results. This is an open problem of the implemented prototype.

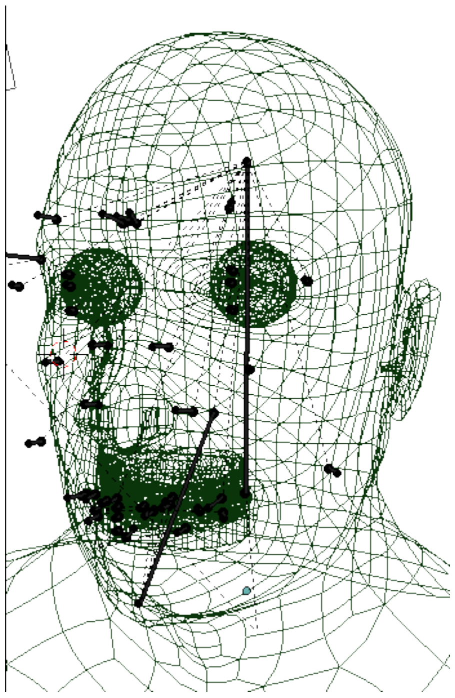

Figure 4.10: The armature of the *Holmen Advanced Rig* shows how joints can be modeled in the Blender modeling tool.

4.4 Technical Details

In order to enable things to work together, the interfaces for communication between libraries and tools had to be implemented. The most complicated part was the data exchange between the OpenCV program written in C++ and the Unity3D Project with scripts written in C#. For this purpose we used *Export API* in the C++ application in order to be able to use program methods from other applications. With *Export API* we are able to run the program as a plug-in in the Unity3D. The Unity3D project needs the data from SVM classifier and features located in the ASM/AAM library, so that the recognized emotion can be shown and the features applied to the avatar. C# scripts written in MonoDevelop were used for retrieving data from the plug-in and applying them to the avatar. The data exchange between C# Unity3D scripts and C++ plug-in was done by Marshalling. Since the *OpenCV::Mat* data structure is not implicitly supported, data could not be transfered directly. The C# script sends two float arrays to the plug-in, which the OpenCV matrix is partitioned into. These filled arrays are then reassembled back in the C# script (See Figure 4.11).

In addition to the *OpenCV::Mat* representation of the fitted ASM/AAM model, the classified emotion also needs to be transferred. Here only a predefined label is transferred, which lowers the data amounts sent between the plug-in and Unity3D project. The *showCamera* boolean parameter is a flag for turning debug options on and off.

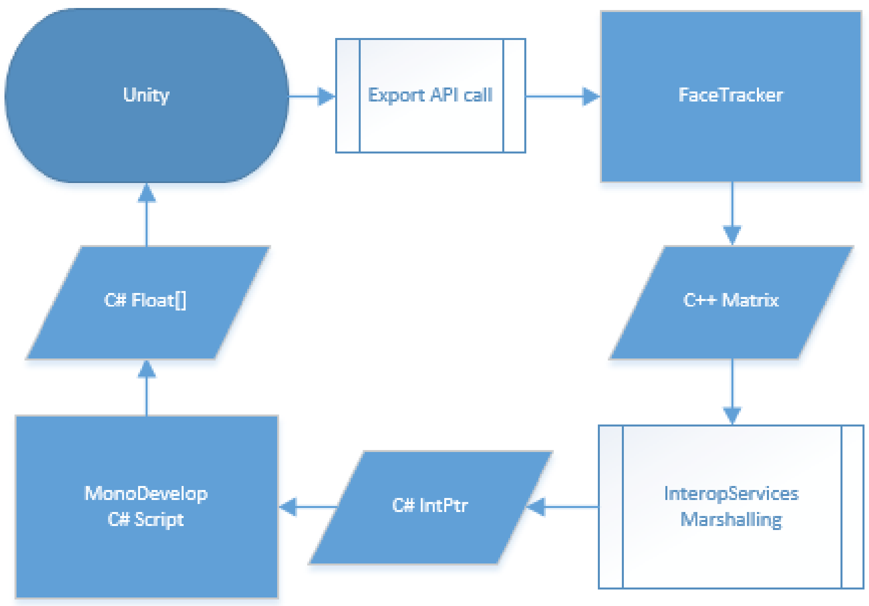

Figure 4.11: Data flow chart showing the data flow between the Unity C# script and C++ plug-in.

```
EXPORT_API int getShape(
  float** pointsX, float** pointsY, // marshalling of a OpenCV::Mat
  int* classified_label, // classified emotion label
  bool showCamera)
```

The C# script can call the above method as follows:

```
[DllImport("FACE_TRACKER")]
 private static extern int getShape(
        ref IntPtr pointsX, ref IntPtr pointsY,
        ref int classified_label,
        bool showCamera);
```

All of the libraries used were downloaded in a form of source code that first had to be built. For this we used the *CMake* [114] tool. As a build target we used Microsoft Visual Studio 2010 or 2013 when working on Windows, and Xcode 6.X when working on OSX. These environments were used to build the dynamic library for use as a plug-in in the Unity3D project. As mentioned above, the Unity3D project was developed in the Unity3D Editor and scripts used in this project were written in the MonoDevelop Editor, since these two editors cooperate well.

One of the goals of this thesis was to write prototype that does not need any special hardware to run fluently in real-time. Common integrated or USB web-cams turned out to be satisfactory, even for capturing the user's face.

[114] http://www.cmake.org/, 21.10.2015

CHAPTER 5

Evaluation

5.1 Approach

After finishing the implementation phase, the evaluation began. We evaluated the implemented prototype in both a quantitative and qualitative manner.

We focused on two main goals in the evaluation phase.

- We wanted to evaluate the emotion classification part to find out how precisely it works and how it behaves with various faces.
- The second goal was to evaluate the animation part. We wanted to find out how accurately the animation matches users' expressions and whether it animates diverse faces well.

The evaluation phase was divided into two parts. We evaluated the emotion recognition part and the animation part separately, since they had to be processed in different ways.

- The results of the emotion classification were processed quantitatively. The individuals tested alternated their facial expressions, informing us whether the recognized emotion matched their intention. Counting correctly classified facial expressions gave us an overview of how well the prototype works.

- Secondly we wanted to know how people perceive the animation of their expressions on the given avatar. Here individuals were asked to rate the avatar animation with points from 1 to 5; the higher the score the better the animation. We did not adjust the parameters of the avatar to individual faces since we wanted to check the basic prototype without fine tuning.

We executed the evaluation of the prototype as described above on a sample of 10 individuals. We wanted participants to be as different as possible, and so we therefore created three age groups, both male and female. The first group consisted of participants of around 20, the second of around 30-40 and the last, around 60. Three female and one male participant took part in the first group, one female and two males in the second group and two females and one male in the third. Their tasks were to keep switching between three different facial expressions: happy, sad and fearful, until they become familiar with the prototype.

In addition to this, after the testing we discussed with the participants their overall feelings concerning the prototype. We wanted to know if they found it useful, smooth and precise enough, and whether they had any ideas for improvements. We also observed the precision of the ASM fitting algorithm in the debug mode while participants were testing the prototype, in order to see if the model matches properly. As soon as the model mismatched an individual's face, we stopped him or her and reinitialized the model by covering the web-cam for a second.

The results are provided in the Section 5.2.

5.2 Results

The results shown in Table 5.1 (graphically illustrated in Figure 5.1) show that emotion recognition works well. However, the emotion is not recognized correctly all of the time; Figure 5.3 for example shows the incorrectly recognized happy expression of Participant H. Here, the happy expression was recognized as the neutral one. The participant did not look directly into the web-cam and was captured at an angle, which can influence the

Expression	HAPPY	SAD	FEAR
Participant A	correct	correct	correct
Participant B	incorrect	correct	incorrect
Participant C	correct	correct	correct
Participant D (without glasses)	incorrect	correct	correct
Participant E	correct	correct	correct
Participant F	correct	correct	correct
Participant G	incorrect	correct	incorrect
Participant H	incorrect	correct	correct
Participant I (without glasses)	correct	correct	correct
Participant J	correct	correct	correct

Table 5.1: The results of the facial expression categorization evaluation. Tests were marked as incorrect if the emotion was not recognized correctly on the participant's natural happy/sad/fearful expression, and if the participant had to make additional effort in order for the emotion to be classified properly.

Animation Rating	HAPPY	SAD	FEAR
Participant A	5	4	2
Participant B	4	5	3
Participant C	3	4	3
Participant D	5	5	4
Participant E	4	4.5	3.5
Participant F	4	5	4
Participant G	4	4	3
Participant H	5	5	4
Participant I	5	5	5
Participant J	5	5	4

Table 5.2: Avatar animation scores based on participants' perceptions. 1: bad - 5: excellent. Participant A gave a low rating of the fearful animation since she was not able to fit the expression properly at all (See Figure 5.5).

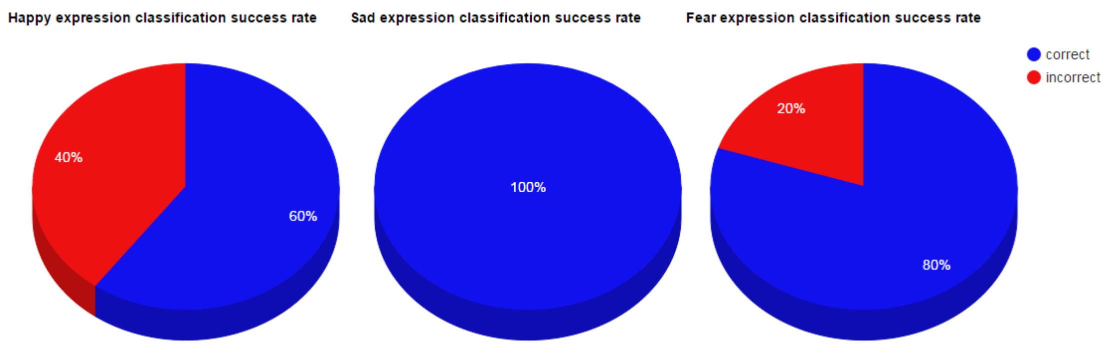

Figure 5.1: The results of emotion classification.

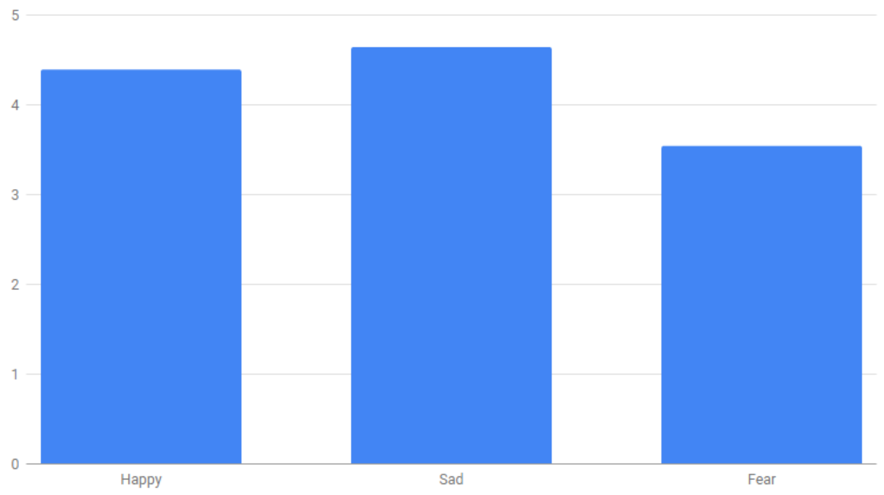

Figure 5.2: Overall scores of the animation's rating (See Table 5.2)

classification. At the beginning Participant E was unable to achieve correct expression classification correctly at all. It transpired that the problem was the distance between him and the web-cam, which apparently negatively influenced the classification. He was classified correctly as soon as he moved closer to the camera.

Another problem with emotion classification was caused by the incorrectly matched ASM model. Participant D and I were wearing spectacles, and this interfered with FaceTracker, so they removed them. This ASM/AAM fitting drawback was described in Section 2.1.3. As also described in Section 2.1.3, the ASM/AAM fitting algorithm encounters problems when fitting a face with a beard correctly (See Figure 5.4). This occurred with Participant F, who was sporting a beard. Despite FaceTracker not matching the face well, the emotion classifier was able to classify the emotion correctly.

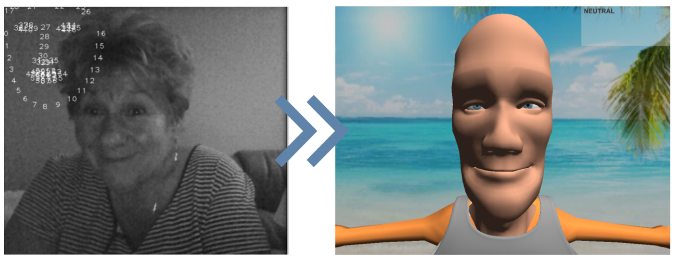

Figure 5.3: Participant H: The happy expression classified incorrectly as the neutral expression. The ASM debug points were matched correctly, but intentionally moved to the left top corner so that the real expression can be seen clearly.

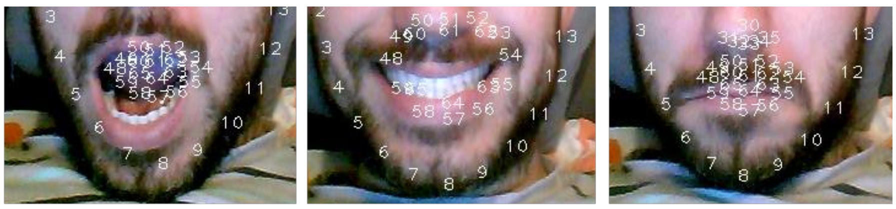

Figure 5.4: Figure showing how FaceTracker fits the face of Participant F, who sports a beard.

Table 5.2 (graphically illustrated in Figure 5.2) shows participants' ratings of the animation. The animation of the fearful expression gained a lower rating, since the rig of the avatar's mouth is not flexible enough to simulate a mouth wide open (See Figure 5.5).

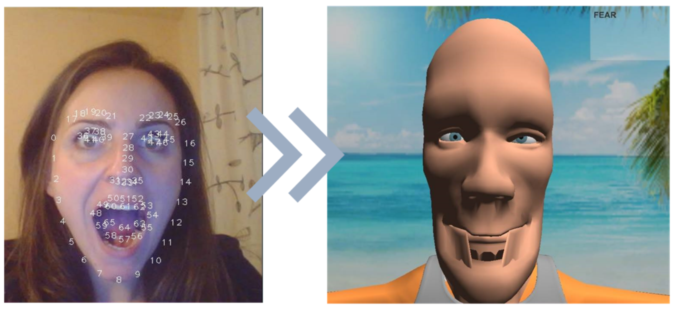

Figure 5.5: Participant A: Inaccuracy in the animation of the avatar's fearful animation.

During the free discussion participants complained about losing the face tracking and the need to re-initialize the model by covering the web-cam for a second. This is a known issue of FaceTracker that does not happen often and strongly depends on lighting conditions. Many participants noted that the sensitivity of the emotion classifier was too low. They became detained in the sad classification until it unnaturally over-expressed a different emotion. This was caused by the smaller training sample with large differences between emotions, and can be easily corrected by training the emotion classifier with a larger training sample of facial expressions.

We noticed that participants tended to move and rotate the head while working with the computer. Both rotating and moving heads affects the prototype in some way; head movements affect the avatar's animation, and rotation affects the emotion classification. We also observed that participants tended to move the head towards or away from web-cam noticeably when expressing fear. Since this is natural behavior in a real setting, it should be considered before the prototype could be practically used.

Finally, all the participants said that it was fun to play with the prototype and that they had not tried anything like this before. They are doubtful about its usage in chat-like applications, while the idea to automatically analyze the user's reaction when watching advertisements was considered a potentially useful replacement for questionnaires in cases where the user is aware of being observed.

CHAPTER 6

Conclusion

In this work we managed to build a fully working prototype application for recognizing and animating user facial emotions in real-time. We compared state-of-the-art technologies and chose the most suitable. We first focused on the recognition of face features and considered possible ways of solving the problem. Active Shape and Active Appearance Models were chosen for this task. We tested and compared published ASM/AAM implementations, in which FaceTracker was considered the most suitable library for implementing ASM. Secondly, with face features recognized, we focused on expression analysis. We used Support Vector Machines for the facial expression classification and also created a training interface that was used for building the SVM model. The final step was to animate the facial expression on an avatar. Since this part was designed in Unity Engine, we used the face features recognition and expression classification part as a plug-in in the Unity project. Here, face features were animated on an avatar, and a textual description of the classified emotion was also displayed.

The evaluation showed that both emotion recognition and avatar animation parts work satisfactorily. However, certain drawbacks such as a lower sensitivity of the emotion classifier and a low flexibility of avatar's mouth emerged, and improvements could be made to these in future work. The following list summarizes the drawbacks and suggests improvements:

- Evaluation participants found the recognition not sensitive enough. In order to become classified by the prototype correctly, they had to show their expressions more intensely than they would do in real-life scenarios. In the case of one participant the sensitivity problem was corrected by moving closer to the camera.

- We observed that users tended to move their heads frequently, affecting both emotion classification and avatar animation in a negative way. For future work the movement and the rotation could be recognized and dealt with so that the system is no longer influenced in this way.

- Another problem that we observed while testing is that sometimes the FaceTracker library is unable to match the model correctly to the face. We re-initialized the model automatically in code when FaceTracker lost the face entirely. In the case of mismatching the face, we were not able to automatically recognize the failure and to re-initialize the model. This issue should also be dealt with in future work.

- Evaluation participants were sometimes not satisfied with the avatar's mouth animation. The chosen avatar does not have a mouth flexible enough for opening wide. This could be corrected either by adjusting the avatar being used or using another one with a higher level of detail.

These are enhancements to the built prototype that could enable it to work more precisely and to look better. However, with these enhancements in place the prototype would become ready for production. For future work we recommend training the SVM model with a larger training sample in order to render the emotion classification both more precise and more sensitive. The avatar could also be enhanced by rigging it more precisely and better optimizing it to the ASM model.

For testing the prototype, both the source code[120] and Windows executable[121] are provided. Despite the executable being provided for the Windows operating system only, the platform-independent source code can also be built on Linux and OSX.

[120]https://github.com/mirobyrtus/fips/tree/master/source, 10.11.2015

[121]https://github.com/mirobyrtus/fips/tree/master/executable/stabilized, 10.11.2015

Bibliography

[AHvdSV05] Aitor Azcarate, Felix Hageloh, Koen van de Sande, and Roberto Valenti. Automatic facial emotion recognition. *Universiteit van Amsterdam*, 2005.

[ALC$^+$07] Ahmed Bilal Ashraf, Simon Lucey, Jeffrey F. Cohn, Tsuhan Chen, Zara Ambadar, Ken Prkachin, Patty Solomon, and Barry J. Theobald. The painful face: Pain expression recognition using active appearance models. In *Proceedings of the 9th International Conference on Multimodal Interfaces*, ICMI '07, pages 9–14, New York, NY, USA, 2007. ACM.

[AUA08] I. Ari, A. Uyar, and L. Akarun. Facial feature tracking and expression recognition for sign language. In *Computer and Information Sciences, 2008. ISCIS '08. 23rd International Symposium on*, pages 1–6, Oct 2008.

[BVS$^+$96] Marian Stewart Bartlett, Paul A. Viola, Terrence J. Sejnowski, Beatrice A. Golomb, Jan Larsen, Joseph C. Hager, and Paul Ekman. Classifying facial action. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 823–829. MIT Press, 1996.

[CET98] Timothy F. Cootes, Gareth J. Edwards, and Christopher J. Taylor. Active appearance models. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 484–498. Springer, 1998.

[CKH10] Darren Cosker, Eva Krumhuber, and Adrian Hilton. Perception of linear and nonlinear motion properties using a facs validated 3d facial model.

In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization*, APGV '10, pages 101–108, New York, NY, USA, 2010. ACM.

[CL11] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[CTCG95] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models-their training and application. *Computer Vision and Image Understanding*, 61(1):38 – 59, 1995.

[DCLR13] Dragos Datcu, Marina Cidota, Stephan Lukosch, and Leon Rothkrantz. Noncontact automatic heart rate analysis in visible spectrum by specific face regions. In *Proceedings of the 14th International Conference on Computer Systems and Technologies*, CompSysTech '13, pages 120–127, New York, NY, USA, 2013. ACM.

[Dha13] Abhinav Dhall. Expression analysis in the wild: From individual to groups. In *Proceedings of the 3rd ACM Conference on International Conference on Multimedia Retrieval*, ICMR '13, pages 325–328, New York, NY, USA, 2013. ACM.

[DR07] Dragoş Datcu and Léon Rothkrantz. Facial expression recognition in still pictures and videos using active appearance models: A comparison approach. In *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, CompSysTech '07, pages 112:1–112:6, New York, NY, USA, 2007. ACM.

[Ekm78] W. V. Friesen Ekman, P. The facial action coding system: A technique for measurement of facial movement. Consulting Psychologists Press, 1978.

[GBGFMB12] Jorge Garcíia Bueno, Miguel González-Fierro, Luis Moreno, and Carlos Balaguer. Facial gesture recognition using active appearance models based on neural evolution. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 133–134, New York, NY, USA, 2012. ACM.

[HC12] Zakia Hammal and Jeffrey F. Cohn. Automatic detection of pain intensity. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 47–52, New York, NY, USA, 2012. ACM.

[JG10] Xiaodong Jia and Jiangling Guo. Eyeglasses removal from facial image based on phase congruency. In *Image and Signal Processing (CISP), 2010 3rd International Congress on*, volume 4, pages 1859–1862, Oct 2010.

[KPG11] Sharad Kohli, Surya Prakash, and Phalguni Gupta. Age estimation using active appearance models and ensemble of classifiers with dissimilarity-based classification. In *Proceedings of the 7th International Conference on Advanced Intelligent Computing*, ICIC'11, pages 327–334, Berlin, Heidelberg, 2011. Springer-Verlag.

[LM02] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900–I–903 vol.1, 2002.

[LTC97] A. Lanitis, C.J. Taylor, and T.F. Cootes. Automatic interpretation and coding of face images using flexible models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(7):743–756, Jul 1997.

[MEK03] Philipp Michel and Rana El Kaliouby. Real time facial expression recognition in video using support vector machines. In *Proceedings of the 5th International Conference on Multimodal Interfaces*, ICMI '03, pages 258–264, New York, NY, USA, 2003. ACM.

[MN14a] S. Milborrow and F. Nicolls. Active Shape Models with SIFT Descriptors and MARS. *VISAPP*, 2014.

[MN14b] Stephen Milborrow and Fred Nicolls. Active shape models with sift descriptors and mars. *VISAPP*, 1(2):5, 2014.

[ND13] Svetoslav Nedkov and Dimo Dimov. Emotion recognition by face dynamics. In *Proceedings of the 14th International Conference on Computer Systems and Technologies*, CompSysTech '13, pages 128–136, New York, NY, USA, 2013. ACM.

[NLEDlT08] Minh Hoai Nguyen, Jean-Francois Lalonde, Alexei A Efros, and Fernando De la Torre. Image-based shaving. *Robotics Institute*, page 141, 2008.

[SRDW09] Xiaofan Sun, Leon Rothkrantz, Dragos Datcu, and Pascal Wiggers. A bayesian approach to recognise facial expressions using vector flows. In *Proceedings of the International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing*, CompSysTech '09, pages 28:1–28:6, New York, NY, USA, 2009. ACM.

[SRP10] Amrutha Sethuram, Karl Ricanek, and Eric Patterson. A comparative study of active appearance model annotation schemes for the face. In *Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '10, pages 367–374, New York, NY, USA, 2010. ACM.

[SRU$^+$10] Markus Storer, PeterM. Roth, Martin Urschler, Horst Bischof, and JosefA. Birchbauer. Efficient robust active appearance model fitting. In AlpeshKumar Ranchordas, JoãoMadeiras Pereira, HélderJ. Araújo, and JoãoManuelR.S. Tavares, editors, *Computer Vision, Imaging and Computer Graphics. Theory and Applications*, volume 68 of *Communications in Computer and Information Science*, pages 229–241. Springer Berlin Heidelberg, 2010.

[SSP$^+$12] Catherine Soladié, Hanan Salam, Catherine Pelachaud, Nicolas Stoiber, and Renaud Séguier. A multimodal fuzzy inference system using a continuous facial expression representation for emotion detection. In *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ICMI '12, pages 493–500, New York, NY, USA, 2012. ACM.

[VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001.