



Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Theoretische Informatik
Prof. Dr. Alex Waibel

Improving Active Appearance Models from 3D Information

with application to Facial Expression Recognition

Diplomarbeit

von

Jörg Liebelt

Betreuer:

Prof. Dr. Alex Waibel
Dr.-Ing. Rainer Stiefelhagen
Dr. Jie Yang

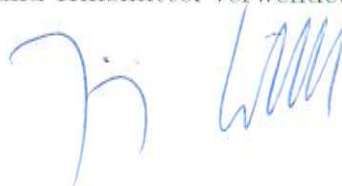
Tag der Anmeldung: 1. Mai 2005

Tag der Abgabe: 31. Oktober 2005



Ich erkläre hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Karlsruhe, den 31. Oktober 2005

A handwritten signature in blue ink, consisting of a stylized 'J' followed by a series of loops and a final 'M'.

Abstract

Existing methods for tracking facial expressions based on Active Appearance Models (AAMs) are usually confined to perfect lab conditions, trained only on a particular person and limited in detail resolution. In this work, we suggest incorporating dense 3D data obtained from a stereo camera into an existing AAM, thereby combining the advantages of fast AAM fitting and dense 3D mesh alignment. We describe a feedback circuit allowing for continuous improvement of model fitting and immediate error correction as part of a multi-level gradient descent method. Our system can then be used for precise classification of facial expressions using Support Vector Machines (SVMs).

Summary

In order to build natural and intuitive human-computer interfaces, it is of vital importance for the computer to have a certain level of knowledge on the situational context of the interaction [30]. Similar requirements exist for unsupervised behavioral analysis in healthcare applications [16]. One possible source of such contextual information is the human face. Human faces convey a vast range of information notably on emotions and the focus of attention. Moreover, the lips provide visual clues on the spoken words that help communicating in a noisy environment [32]. It is therefore necessary to develop a means of reliably identifying and tracking human faces as well as changing facial expressions.

In this work, we present a method for tracking facial expressions using a stereo camera. The focus of our approach is to improve stability and resolution of existing 2D tracking methods based on Active Appearance Models [3] by adapting the tracking results to stereo data. In particular, we aim at allowing the use of AAMs for low-resolution input images and sparsely-textured face regions. Moreover, our system achieves a processing speed of approximately 10 frames per second and allows for movements of the face deviating up to 15° from the frontal view, at a distance of 2m from the camera.

Firstly, a 2D Active Appearance Model [3] is developed using a given hand-labeled training set. The model is then initialized in a video sequence using a color-histogram based face segmentation algorithm. Subsequently, the 2D AAM is iteratively optimized using a least-squares steepest descent method to match facial deformations. An initially tracked sequence of 2D face landmarks is then used to develop a 3D model describing an equivalent set of deformations using a structure from motion factorization method [41]. Based on this 3D extension to the 2D model, we can then optimize in parallel a 3D representation of the 2D model. In a final optimization step, this 3D representation is constrained to deformations corresponding to a stereo depth map of the face that has been obtained from the stereo camera.

We conclude that precision and stability of existing 2D face tracking methods can be significantly improved by allowing for feedback of depth information into the 2D tracking process.

Résumé

Afin de pouvoir créer des interfaces homme-machine intuitives, il est d'une importance primordiale que l'ordinateur possède des informations sur l'utilisateur et sur le contexte dans lequel se déroule l'interaction. En outre, les applications médicales portant sur la surveillance de patients et sur l'interprétation psychologique de leur comportement nécessitent que l'ordinateur soit capable d'identifier les émotions des personnes sous surveillance. Le visage est la source d'informations sur l'émotion humaine la plus facilement accessible, car elle transmet des indices notamment sur la direction du regard et sur les six émotions élémentaires de la peur, du dégoût, de la joie, de la tristesse, de la colère et de la surprise. De plus, les lèvres servent à fournir des indices facilitant l'interprétation de la langue parlée dans les environnements bruyants. Par conséquent, nous souhaitons développer un moyen de segmenter et de suivre le visage humain et d'identifier les expressions faciales.

Dans le cadre de notre travail, nous présentons une méthode pour le suivi d'expressions faciales à l'aide d'une caméra stéréo. L'accent de notre approche revient à améliorer la stabilité des *Active Appearance Models* (modèles d'apparence active) en 2D en effectuant des optimisations itératives à partir des informations 3D. Cette stratégie nous permet d'employer notre système dans des conditions réalistes, notamment de travailler avec des séquences vidéo à moyenne et à basse résolution et d'analyser les régions du visage peu texturées tout en assurant une précision suffisante. De plus, notre implémentation atteint une vitesse d'environ 10 images par seconde ce qui correspond à un traitement de séquences vidéo en temps réel.

D'abord, un *Active Appearance Model* (modèle d'apparence active) en 2D est créé à partir d'une séquence d'images d'entraînement dont les correspondances entre les points caractéristiques ont été établies à la main. Ensuite, la position initiale du modèle est choisie en utilisant une méthode de segmentation par histogramme de couleur et un filtrage par profondeur stéréo. Puis, le modèle 2D est itérativement adapté aux déformations faciales à travers l'optimisation en moindres carrées par descente de gradient. En même temps, nous nous servons d'une autre séquence d'images d'entraînement et d'un algorithme de *Structure from Motion* (structure à partir de mouvement) afin de construire un modèle 3D décrivant les déformations faciales 2D d'une manière équivalente. Finalement, ce modèle 3D nous permet d'optimiser en parallèle une représentation en 3D du visage suivi et d'imposer des contraintes sur les mouvements permis du modèle 2D qui doivent correspondre aux données de profondeur obtenues par la caméra stéréo. En résumant, nous établissons un cercle de retour d'information 3D permettant l'amélioration et l'adaptation en ligne d'un modèle 2D d'un visage.

Nous concluons que la précision et la stabilité des méthodes existantes pour le suivi des déformations faciales peuvent être considérablement améliorées en utilisant des informations obtenues à l'aide d'une caméra stéréo. En outre, nous montrons comment cette méthode peut servir à classifier des expressions faciales d'une manière très efficace.

Zusammenfassung

Intuitive Mensch-Maschine-Schnittstellen sind nur möglich, wenn der Computer über ein bestimmtes Maß an Information bezüglich des Handlungszusammenhanges verfügt. Ähnliche Anforderungen stellt die unüberwachte Auswertung von Patientendaten im Rahmen von Anwendungen des Gesundheitswesens. Eine mögliche Quelle derartiger Kontextinformationen ist das menschliche Gesicht. Das Gesicht vermittelt eine grosse Anzahl von Informationen über Stimmung und Aufmerksamkeit eines Menschen. Zudem bieten die Lippenbewegungen eine Möglichkeit, visuelle Hinweise zur Verbesserung der Verständigung in geräuschbelasteten Umgebungen zu nutzen. Aus diesen Gründen ist es angebracht, einen Weg zu finden, der die zuverlässige Identifizierung und Verfolgung von menschlichen Gesichtern sowie die Analyse von sich verändernden Gesichtsausdrücken erlaubt.

Im Rahmen der vorliegenden Arbeit soll ein Ansatz zur Verfolgung von Gesichtsausdrücken mittels einer Stereokamera entwickelt werden. Ziel dieser Arbeit ist die Verbesserung der Zuverlässigkeit und die Erhöhung der Auflösung von bestehenden Methoden, die auf zweidimensionalen sogenannten *Active Appearance Models* (AAM) beruhen, indem deren Ergebnisse durch die Einbindung von Stereodaten aufgewertet werden. Insbesondere soll die Anwendung von AAMs auf Eingangsdaten mit niedriger Auflösung sowie auf gering texturierte Gesichtsregionen ermöglicht werden. Das entwickelte System erreicht eine Verarbeitungsrate von 10 Bildern pro Sekunde bei Abweichungen des Gesichts um bis zu 15° von der Frontalansicht und einem Kameraabstand von etwa 2 Metern.

Zunächst wird unter Verwendung handmarkierter Trainingsdaten ein zweidimensionales AAM erstellt. Dieses Modell wird anschliessend innerhalb einer Bildfolge initialisiert, indem das Gesicht durch Farbhistogramme und Filterung von Tiefeninformationen segmentiert wird. In der Folge wird das zweidimensionale AAM mit Hilfe eines Gradientenabstiegsverfahrens für kleinste Fehlerquadrate iterativ an die auftretenden Gesichtsausdrücke angepasst. Eine anfänglich erstellte Folge von zweidimensionalen Verfolgungsdaten eines Gesichts wird dann zur Erstellung eines dreidimensionalen Gesichtsmodells genutzt, welches das Verhalten des bestehenden zweidimensionalen Modells äquivalent beschreibt. Hierbei kommt ein sogenannter *Structure from Motion*-Algorithmus zur Anwendung. Aufbauend auf dieser dreidimensionalen Erweiterung des bestehenden Modells kann nun das gesamte Modell in einem weiteren Optimierungsschritt an die durch die Stereokamera erstellten Tiefendaten der Szene angepasst werden.

Zusammenfassend legen die Ergebnisse der vorliegenden Arbeit nahe, dass bestehende zweidimensionale Ansätze zur Gesichtsverfolgung erheblich verbessert werden können, indem Tiefeninformationen von Stereokameras in die Modellanpassung eingebunden und zur kontinuierlichen Modellverbesserung genutzt werden.

Acknowledgements

This work was conducted in Pittsburgh at the Interactive Systems Labs as a part of the French-German double-diploma program at the Universität Karlsruhe (TH) and the ENSIMAG. I am indebted to my advisors Rainer Stiefelhagen and Jie Yang for having offered me this opportunity and for constantly supporting my work. I am extremely grateful to Jing Xiao who provided me with details on his work on 2D/3D AAMs, his 3D factorization algorithm and many implementational suggestions. I would like to thank Thomas Schaaf and Celine Carraux for their help with the administration and organization of my stay at the CMU, Prof. Jacques Calmet and Marianne Genton for their support during the double-diploma program and the *CHIL* project team for allowing me to collect testing data during their meetings.

Contents

1	Motivation	1
1.1	Goal	2
1.2	Possible Fields of Application	2
1.3	Survey of Related Work	3
1.3.1	Describing facial expressions	3
1.3.2	Tracking Deformable Shapes	3
1.3.3	Shape Models from Depth Maps	3
1.4	General Outline	4
2	Registration of Deformable Shapes	5
2.1	Choice of Suitable Landmarks	5
2.2	Shape Alignment	6
2.3	Limitations	6
2.4	Modeling Shape and Texture Variation	7
3	Fitting Two-Dimensional Active Appearance Models	9
3.1	Definition	9
3.2	Project-Out ICIA	10
3.3	Initialization	11
3.4	Summary	11
4	Extending to Three Dimensions	13
4.1	Factorization	13
4.2	Extending an AAM from 2D to 3D	14
5	Using Stereo Vision	15
5.1	Obtaining and Structuring Stereo Data	15
5.2	Fitting to Stereo Data	15
5.3	Overview of Performed Fitting Steps	17

6	Experimental Results	19
6.1	Generic Example	19
6.1.1	Tools Used	19
6.1.2	Initialization	19
6.1.3	Comparison	20
6.1.4	Table of Residual Errors	22
6.2	CHIL Recordings	23
6.3	Application Example: Facial Expression Classification	25
6.3.1	Support Vector Machines	26
6.3.1.1	General Outline	26
6.3.1.2	Specifications	27
6.3.2	Results	27
6.3.2.1	Basic Emotions	27
6.3.2.2	Mouth Movements: Vowels	28
7	Conclusion and Future Work	31
7.1	Problems Encountered and Future Work	31
7.1.1	Twodimensional Model	31
7.1.2	Stereo Extension	32
7.1.3	Classification	32
7.2	Evaluation	33
A	Histogram-Based Face Segmentation	35
B	Principal Components Analysis	37
B.1	Goal	37
B.2	Method Description	37
C	Image Warping	39
C.1	Piecewise Affine Warp	39
C.2	Warp Composition	40
D	Inverse Compositional Image Alignment	41
D.1	Algorithm	41
D.2	Steepest Descent Geometry	42

E	3D Extension	45
E.1	Algorithm	45
E.2	Steepest Descent Geometry	46
F	KD-Trees	49
G	Stereo Vision	51
G.1	Stereo Geometry	51
G.2	Disparity Filtering	53
	References	55

List of Figures

1.1	Outline of this work	4
2.1	Creation of a 2D model	5
2.2	The first 6 appearance variation modes of a model	7
2.3	Examples of 2D AAM instantiation	8
3.1	Segmentation: depth-based filter, histogram-based segmentation, initial AAM position	12
5.1	Steepest Descent Geometry on the depth surface	18
5.2	Interaction between different levels of fitting	18
6.1	Ground truth testing tool GUI	20
6.2	Ground truth example: pose and scale	20
6.3	Ground truth example: rotation	21
6.4	Ground truth example: big smile	21
6.5	Ground truth example: eyes closed, initialization with 12% background outliers	22
6.6	CHIL meeting recordings: successful fitting	24
6.7	CHIL meeting recordings: unsolved issues	25
6.8	Scatterplot of pairs of classes for a 2D parameter space	26
C.1	Warping between two mesh triangles	39
D.1	Steepest descent images (contrast enhanced for visualization)	41
F.1	2D points and their 2D KD-Tree	49
G.1	Point Grey's Bumblebee stereo camera	51
G.2	Stereo vision geometry	52

List of Tables

6.1	Residual errors during generic ground truth experiments	23
6.2	Training confusion matrix for emotion expressions, precision: 86.9% .	28
6.3	Classification of emotion expressions in unseen data (user estimation), precision: 71.3%	28
6.4	Training confusion matrix for mouth movements (N=neutral), preci- sion: 87.5%	29
6.5	Classification of mouth movements in unseen data (user estimation), precision: 73%	29

1. Motivation

Developing applications that make use of information on human emotions to understand the context of human-computer interaction has been an important field of research for decades. As the face provides the most openly available source of visual clues to a person's emotional state of mind, identifying facial expressions is an important step towards the development of such applications.

While much work has been dedicated to head tracking [23], head pose and gaze estimation [30], facial expressions remain a challenging research topic due to the difficulty in describing them comprehensibly. Furthermore, applications requiring detailed high-resolution tracking information, as is the case for lipreading and facial expression recognition, cannot work reliably in just two dimensions. For example, most deformations of the mouth region result in extremely small texture changes, but cause significant variation in depth values. For lipreading applications, we need to be capable of detecting movements not only of the mouth contours, but also of the inner lip region. Stereo vision provides the most reliable and yet sufficiently fast means of integrating depth information into tracking applications.

Existing tracking methods are often confined to perfect lab conditions, trained only on a particular person and limited in detail resolution of facial features [3]. Creation of 2D or 3D facial models is usually done on a small number of different faces, resulting in highly specific representations which are difficult to generalize. Additionally, model precision is rarely acceptable due to error-prone manual labeling of training data. As a consequence, in this work we suggest an approach allowing to track deforming shapes in general and deformations of facial muscles in particular by incorporating stereo information of the scene into existing methods and thereby improving their performance significantly. Furthermore, feedback of depth information into the models as part of an on-line process allows for continuous improving of the model fitting and for immediate error correction.

Furthermore, we would like to emphasize the main difference between our stereo-based method and other model fitting methods using multi-camera environments: high-detail alignment in our approach is done on dense depth data, whereas multi-camera methods only perform sparse 2D point correspondence over several 2D im-

ages. As a consequence, we are able to achieve much more precise and robust fitting results.

In contrast to directly modeling a 3D representation of the shape, we chose to use a 2D representation as the basis of our fitting process. The main advantage of this method is that training can be done on 2D images, instead of the more difficult labeling in three dimensions. Therefore, we intend to improve the existing 2D model instead of replacing it by a more complex representation. Unlike traditional 3D mesh alignment, we use texture information explicitly for fitting, thus accessing multiple sources of scene information. Moreover, the initial 2D fitting speed makes the twodimensional approach extremely interesting as the starting point for more detailed methods. The complete sequence of minimization steps of our system still outperforms execution speed of existing direct 3D models. Similar to 2D fitting on different resolutions, known as hierarchical processing on a Gaussian image pyramid, we perform hierarchical processing that is gradually refining from a simple twodimensional representation towards a precise fit in 3D. In this processing sequence, the 2D model provides a rough initialization. The remaining improvements to the 3D model done on the highest hierarchy level, the stereo fitting, are small since the time-consuming coarse alignments have already been done on the lower fast 2D levels.

1.1 Goal

We aim at achieving stable detection of deforming shapes and tracking of a number of landmarks sufficient for detailed analysis and interpretation of the occurring deformations. Stereo-assisted tracking should be possible even in environments that are too noisy for traditional methods, and at a speed of approximately 10 frames per second.

1.2 Possible Fields of Application

More recently, the range of possible application fields for intelligent human-computer interaction was extended to including contextual information on the emotional state of mind of the user in order to dynamically adapt human-computer interfaces and thus improving efficiency [26].

Similarly, in healthcare and geriatric applications it is desirable to individualize treatment strategies to achieve better results and a more humane care environment [16]. Reliable conclusions on a patients health and treatment progress require a detailed long-term analysis of a patient's behavior. Since a continuously reducing healthcare workforce cannot provide this analysis, an unsupervised computer-based approach can be a possible remedy.

Finally, realistic creation of avatars in virtual reality environments necessitates the online adaptation of facial expressions [5]. While laser-based sensors for surface reconstruction remain slow and expensive [7], stereo-camera based facial expression analysis represents a convenient and fast solution.

1.3 Survey of Related Work

1.3.1 Describing facial expressions

A significant amount of work has been dedicated to describing facial muscle movements and evaluating their psychologic significance. The most widely used and most comprehensive classification scheme is the Facial Action Coding System FACS [8], [25]. While its universal validity over different cultures is disputed, yet it is the only objective description of facial expressions. FACS decomposes every facial expression into elementary units, each of which corresponds to the movement of one major muscle group in the human face. In order to identify a large number of FACS units, [38] proposed a face model consisting of 68 feature points. More feature points complicate hand-labeling of training data, less points reduce the practical usability.

1.3.2 Tracking Deformable Shapes

Early twodimensional shape tracking resorts to deformable contours known as snakes [18], similar techniques for 3D objects include deformable balloon models [7]. These techniques enclose an object inside a contour and define rules and constraints on the permitted shapes allowing these contours to contract around the object on the grayscale image surface similar to gradient descent. However, execution speed is usually slow and the techniques can easily fail in the presence of noisy or complex objects. Moreover, multiple objects or internal object structures have to be represented by a new set of contours.

Eigenfaces [4] and Gabor wavelets [19] decompose an image into bases representing its characteristic features. They allow identification of structures in unseen images, but are usually limited to certain poses, for example full frontal or profile views of a face.

Point Distribution Models (PDM) [10] rely on costly preprocessing steps and fail if the given brightness distribution does not exactly correspond to the training conditions.

Active Shape Models (ASM) [10], [9] are a model-based approach using only shapes as structuring elements. They deform iteratively to fit an instance of a particular shape. However, ASMs may vary only in ways seen in a training set of labeled examples and require the presence of landmark points along strong edges.

[29], [28] suggest 3D Morphable Models (3DMM) to represent facial deformations in shape and texture directly from a three-dimensional model. While this technique is robust against noise and accurate, its fitting speed is extremely slow and the construction of 3DMMs is laborious.

1.3.3 Shape Models from Depth Maps

Depth information is used in several approaches to obtain a precise 3D representation of the shapes present in a given scene. [44] use iterative mesh alignment, nonlinear subdivision and fitting, whereas [35] build face representations with adaptive meshes. In general, execution time of the subdivision algorithms is prohibitive.

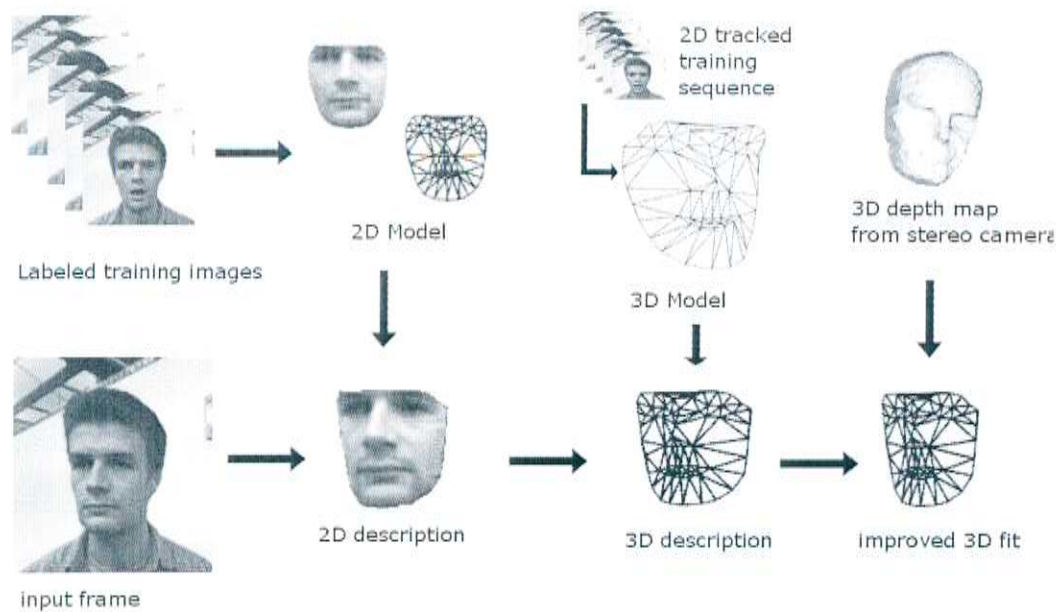


Figure 1.1: Outline of this work

1.4 General Outline

Figure 1.1 on page 4 shows how the different levels of model fitting interact. Each of the steps illustrated in figure 1.1 will be discussed in one chapter of our report:

In chapter 2, we outline how to build a 2D Active Appearance Model (AAM) capable of modeling shape deformations and pose changes. Common landmarks in training images are hand-labeled and the corresponding shapes are extracted and aligned. Using the extracted shapes, a set of 2D bases describing the deformations and a set of texture vectors representing characteristic appearance changes are derived.

Chapter 3 details how a set of 2D feature points is tracked using our 2D AAM by aligning a model instance to a sequence of input images.

In chapter 4, we explain how the tracked feature locations are then fed into a structure from motion algorithm capable of computing a set of 3D bases that equivalently describe the 2D movements as a projection of a 3D shape onto a plane. These 3D bases are used to constrain the deformations of the 2D AAM, thus preventing the creation of invalid shape instances.

Chapter 5 deals with the stereo extension. For each image, we build a KD-Tree (see appendix F for details on this space-partitioning data structure) containing depth data from the stereo camera. We can then compute the distance between the 3D shape and the nearest points in the KD-Tree. This allows us to add another constraint on the 3D shape representation, aligning the 3D shape to minimize distance to the stereo points, which in turn modifies the parameters of the initial 2D model.

Results of ground-truth and classification experiments are presented in chapter 6.

Finally, we give a conclusion in chapter 7 and analyze further improvements to our solution.

2. Registration of Deformable Shapes

The first step towards deriving a representation of shapes and their deformations is a statistical analysis of a set of training images and their subsequent registration into a common coordinate system. This task requires estimating similarity transformations between the shapes such as rotation, translation and scaling which is known as Generalized Procrustes Analysis [14] [31].

2.1 Choice of Suitable Landmarks

Feature points have to be chosen such that their presence is guaranteed in all instances of a certain shape. Moreover, in facial expression recognition feature points should be significant of the underlying facial muscle anatomy to allow for precise detection even of small deformations. In general, suitable feature points are object boundaries, junctions and biological landmarks. The face contains three so-called stable points [25], the two inner corners of the eyes and the point between the nostrils. Since these points provide a frame of reference allowing for the pose estimation, they should always be chosen as landmarks.

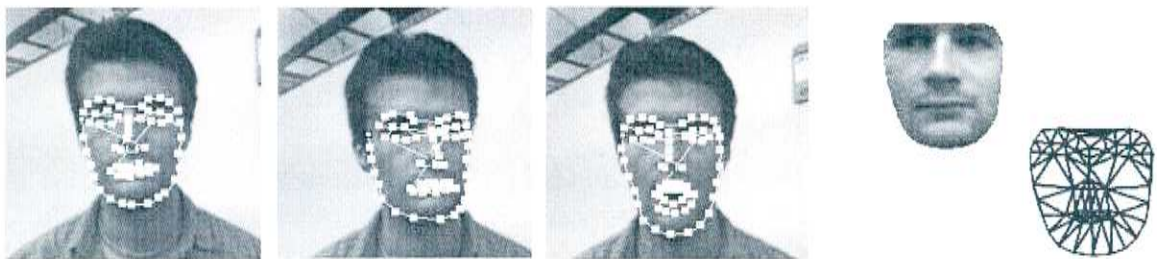


Figure 2.1: Creation of a 2D model

Figure 2.1 on page 5 uses the landmark structure suggested by [24]. We assume that landmarks over a number of images containing different instances of a shape have

been labeled by hand. The choice of the training images should be representative of the possible spectrum of shape deformations one wishes to detect. However, up to now no general rules have been established that can describe a *good* training set. Some images of an example training set and the resulting shape model are shown in figure 2.1. We chose to represent a 2D shape S as a $2n$ element vector of its labeled vertices s_i ,

$$(s_0^x, s_1^x, \dots, s_{n-1}^x, s_0^y, s_1^y, \dots, s_{n-1}^y) \quad (2.1)$$

2.2 Shape Alignment

The goal of shape alignment is to minimize the squared sum of distances of each shape with respect to the mean. When supposing that the shapes differ only in a set of rigid similarity transformations, one possible approach to shape alignment is known as Procrustes Analysis [14] [31]. It consists of the following steps of an iterative least squares minimization of the function

$$gpa(\mathcal{T}) = \underset{\mathcal{T}}{\operatorname{argmin}} \sum_{i=1}^N \|\mathcal{T}_i(S_i) - \bar{S}\|^2 \quad (2.2)$$

where \mathcal{T} is a set of similarity transformations and \bar{S} is the mean shape of the shape set:

- Translate each shape instance such that its center of gravity is at the origin.
- Choose one example as the initial estimate of the mean shape, scale it to zero-mean and use its reference frame as the default.
- Align all the remaining shapes with the current estimate of the mean shape. The allowed transformations can be described by

$$\bar{S}_i = c_i R_i S_i + T_i \quad (2.3)$$

where c_i is a scale factor, R_i is a rotation matrix and T_i is a translation.

- Re-estimate the mean from the newly aligned shapes and align it to default frame of reference.
- Reiterate until convergence is reached, measured either in terms of residual distance to the mean or as a change smaller than some ϵ .

2.3 Limitations

The above mentioned procedure treats shape deformations as gaussian noise and only considers rigid shape transformations. Such a generalization may only be acceptable when deformations are symmetric with respect to the center of gravity. [40] proposes a solution to this problem by incorporating linear shape variation into the minimization function, however in this work we consider that deformations remain small and accept a certain error during alignment. We could verify experimentally that in applications working with face shapes, the only asymmetric deformation changing the shape contour significantly is lifting one corner of the mouth. This deformation should therefore be avoided in the training data.

2.4 Modeling Shape and Texture Variation

We need a parameterized representation of the aligned shapes such that an instance s of any shape deformation can be created as a weighted sum of shape bases b_i and a mean shape s_0 using

$$s = s_0 + \sum_{i=1}^N b_i p_i \quad (2.4)$$

where p_i are parameters weighting the contribution of each shape base to the current shape instance.

One possibility to reduce dimensionality of the training shape space and to derive a simplified shape description is known as Principal Components Analysis [11] [10] and consists of exploiting cross-correlation of the shape points; see appendix B for details. As a result of this process, we obtain a set of shape bases (or shape modes). It is advisable to retain only a certain number of those shape modes in order to reduce computation complexity in the following steps. Selection of shape modes can be done such that a certain proportion (95%) of the variance exhibited in the training set can be explained.

Similarly, PCA is performed on the texture vectors contained within the aligned shapes. However, we normalize the textures for each instance by warping them such that their corresponding shapes match the mean shape. For details on image warping see appendix C. This step removes deformation-induced variation in the textures. Moreover, it is advisable to perform some brightness normalization since the AAM used in this work does not model brightness variations implicitly. As a result of this process, we obtain a parameterizable set of eigenimages describing texture variation in the training set, we will use the term *appearances* to denote its the corresponding instances A :

$$A = A_0 + \sum_{i=1}^M A_i \lambda_i \quad (2.5)$$

where A_0 is the mean appearance and the λ_i are parameters weighting each basis contribution. A set of appearance bases for a training set is shown in figure 2.2 on page 7.

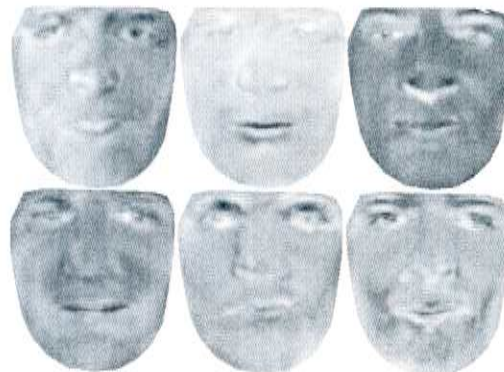


Figure 2.2: The first 6 appearance variation modes of a model

A complete instance of the model can now be created by choosing shape and appearance parameters and computing the shape and the texture that correspond to the

chosen parameters using equations (2.4) and (2.5). Since the resulting appearance instance is still in mean shape normalized form, we have to warp the appearance instance from the mean shape to the current shape instance. The following figure 2.3 on page 8 shows resulting instances of a face obtained by varying parameters of shape and texture descriptions.

While it is possible to perform another PCA on shapes and appearances at the same time and thus obtaining a combined model, we have chosen not to use such a combination. In the following steps, computation can be sped up by projecting out appearance variation as a pre-computation step, it is therefore advantageous to separate shape and appearance representation from the beginning.

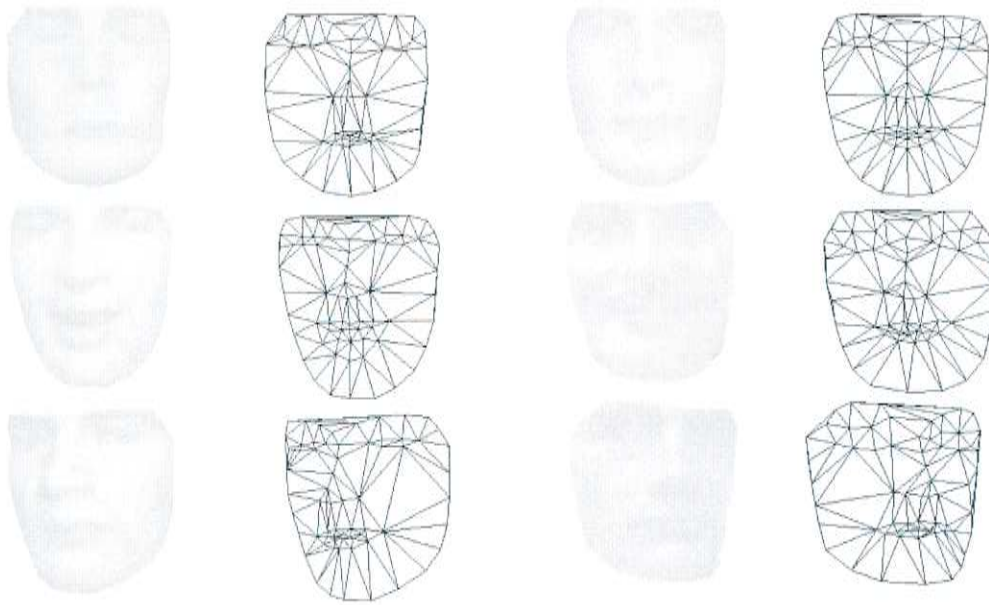


Figure 2.3: Examples of 2D AAM instantiation

3. Fitting Two-Dimensional Active Appearance Models

The Active Appearance Model (AAM) [3] is a generalization of the Active Shape Model (see [9] and chapter 1.3), but uses all the texture information in the image region covered by the target shape, rather than just that near modeled edges. Shape and appearance are iteratively modified to optimally adapt to unseen shape instances by minimizing a sum of squares difference using gradient descent. Several methods have been proposed to efficiently fit 2D AAMs [3], [1]. These methods aim at precomputing most data structures to reduce execution time during the iterations. By far the most efficient method is the project-out inverse compositional image alignment which we chose to use as the basis of our work.

3.1 Definition

An AAM can be defined by a set $(\mathbf{p}, \mathbf{S}, \lambda, \mathbf{A})$ where \mathbf{p} are the shape parameters, \mathbf{S} a set of shape eigenvectors (called *bases* below), λ a set of appearance parameters and \mathbf{A} a set of texture eigenvectors (called *appearances* below). The shape parameters and bases together define the shape warp $W(x, \mathbf{p})$ for every pixel x in an image. An AAM instance then has the form

$$I(W(x, \mathbf{p})) = A(x, \lambda) \tag{3.1}$$

where I is a 2D input image and A the appearance corresponding to the parameter set λ . Since we would like to track a shape that is not only internally deforming, but also undergoing pose changes, we need to account for possible similarity transformations of the shape

$$N(x, \mathbf{q}) = Rx + \mathbf{t} \tag{3.2}$$

where $x = (x, y)$ denotes a pixel within the shape, and $R, \mathbf{t} = (t_x, t_y)$ rotation resp. translation. Choosing the representation of the pose changes of

$$N(x, \mathbf{q}) = s_0 + \sum_{i=1}^4 b_i^* q_i \tag{3.3}$$

with the bases \mathbf{b}^*

$$\begin{aligned} b_1^* &= (x_1^0, x_2^0, \dots, x_v^0, y_1^0, y_2^0, \dots, y_v^0) = s_0 \\ b_2^* &= (-y_1^0, -y_2^0, \dots, -y_v^0, x_1^0, x_2^0, \dots, x_v^0) \\ b_3^* &= (1, 1, \dots, 1, 0, 0, \dots, 0) \\ b_4^* &= (0, 0, \dots, 0, 1, 1, \dots, 1) \end{aligned} \quad (3.4)$$

and s_0 denoting the mean shape, we obtain an equivalent representation where the parameters q_i instantiate the pose and can be used in the same way as the AAM shape parameters. The bases \mathbf{b} and \mathbf{b}^* are orthonormal since the mean shape is zero mean and all shapes have been projected into an orthogonal subspace during PCA. Since our shape alignment is sensitive to linear shape variations, orthonormality of \mathbf{b} and \mathbf{b}^* is explicitly reinforced. Given an input image $I(x)$, we now wish to minimize the sum of squares difference between an instance created with our AAM and the input image:

$$\sum_{x \in s_0} [A_0(x) + \sum_{i=1}^m \lambda_i A_i(x) - I(N(W(x, \mathbf{p}), \mathbf{q}))]^2 \quad (3.5)$$

The solution is obtained by simultaneously minimizing this difference with respect to the parameters \mathbf{p} , \mathbf{q} and λ .

3.2 Project-Out ICIA

We define the *template image* A_0 as the mean appearance warped to the mean shape. In order to compute the set of parameters that describe the current input image in the best possible way, we estimate the image warp necessary to transform the template image into the current image. When changing from one to the next input image, the template image, warped to the previous frame fit, is warped again to align with the current input image.

Traditional image alignment techniques determine this warp by solving the gradient descent problem for an additive parameter update to the current warp in order take into account pose and shape changes:

$$\begin{aligned} p_i &\leftarrow p_i + \Delta p_i \\ \lambda_i &\leftarrow \lambda_i + \Delta \lambda_i \\ q_i &\leftarrow q_i + \Delta q_i \end{aligned} \quad (3.6)$$

This technique is known as *forward additive image alignment* [2]. However, in this technique Jacobian, Hessian and steepest descent images need to be recomputed at each iteration and reduce the fitting speed.

A solution has been suggested by [1], known as *inverse compositional image alignment*. Instead of solving for an additive parameter update to the warp, we solve for a compositional update to the current warp. Moreover, the incremental warp is computed with respect to the template image and then inverted, thus allowing for a pre-computation of the Jacobian. The update now has the form

$$W(x, \mathbf{p}) \leftarrow W(x, \mathbf{p}) \circ W(x, \Delta \mathbf{p})^{-1} \quad (3.7)$$

The warp composition is detailed in appendix C. It should be noted however that the warp inversion is based on a first-order approximation of $W(x, \Delta \mathbf{p})$ to achieve a simplified warp composition

$$W(x, \Delta \mathbf{p})^{-1} = W(x, -\Delta \mathbf{p}) \quad (3.8)$$

In practice, this results in a slight smoothing of the warp at each iteration which may reduce precision.

In order to account for appearance variation, the appearance can be projected out of the minimization as suggested in [3], thus avoiding re-computation of the updated appearance variation in each iteration. The idea of this approach is to perform steepest descent in an orthogonal complement to the appearance-affected subspace to determine a set of minimizing shape parameters \mathbf{p} , which can then be used to determine the optimal appearance parameters λ as the closed form dot-product of the current error image with the appearance bases

$$\lambda_i = \sum_{x \in s_0} A_i(x) \cdot [I(N(W(x, \mathbf{p}))) - A_0(x)] \quad (3.9)$$

A detailed description of the project-out ICIA algorithm can be found in appendix D.

3.3 Initialization

The steepest descent method described above will converge towards any local minimum present in the input image. To avoid incorrect fittings in image areas possessing intensity characteristics similar to the appearance of the AAM, we need to initialize the AAM as precisely as possible.

We propose using an initialization based on the combination of a color histogram back-projection and a stereo camera-based depth segmentation. In a first step, a region corresponding to typical face colors is segmented. Its moments are estimated to obtain a first estimation of size and orientation, for details see appendix A. In parallel, the depth map corresponding to the current input image is passed through a series of preprocessing filters, removing areas with erroneous depth values or small surface. These filters are described in appendix G.

Figure 3.1 on page 12 shows an example of the segmentation performed by our implementation, using a face image. On average, the 2D AAM needs an additional 30 iterations to converge after initialization from a frontal view. However, heads turned to one side during initialization require significantly more iterations until convergence.

3.4 Summary

Based on the parameterized shape description obtained, the corresponding 2D AAM is fitted to an input image. We can now track shape landmarks in images displaying pose and shape variation.



Figure 3.1: Segmentation: depth-based filter, histogram-based segmentation, initial AAM position

4. Extending to Three Dimensions

We wish to adapt a 2D AAM to stereo data. As a consequence, we need to derive a 3D description of the tracked object using the fitted 2D shape, otherwise no comparison between stereo data and fitted shape is possible. While 2D AAMs can generate states that cannot be attained by a 3D object [41], it is possible to impose constraints such that the allowed deformations correspond to those of a 3D shape. As a byproduct of the constrained fitting, we obtain an equivalent 3D representation of the 2D model, \bar{s} . It is of particular interest to us since it will allow us to compute its distance to a stereo data surface.

4.1 Factorization

Constraining the 2D AAM to 3D deformations requires knowledge of a set of 3D bases which describe the 2D deformations in an equivalent way. Obtaining 3D information of a given scene is the topic of *Structure from Motion* research [42] [43]. Given a measurement matrix W containing the locations of corresponding landmark points in a sequence tracked by a 2D AAM, we perform factorization by singular value decomposition such that

$$W = \tilde{M}\tilde{B} \quad (4.1)$$

where \tilde{M} is a scaled projection matrix and \tilde{B} is a shape vector matrix. Since this factorization is not unique, a corrective matrix G has to be determined such that

$$\begin{aligned} M &= \tilde{M}G \\ B &= G^{-1}\tilde{B} \end{aligned} \quad (4.2)$$

[43] proposes a method to perform this step by imposing basis and rotation constraints and by taking into account degenerated cases. We have been using his method unchanged.

The scene chosen for factorization should intuitively contain all pose and shape variations one intends to track with the 3D model. We have experimentally determined that the tracked sequence used for 3D base recovery should consist of at least 100 frames displaying strong deformations to avoid singularities and allow for significant eigenvalues of W .

4.2 Extending an AAM from 2D to 3D

Given the optimally fitted 2D AAM in the input image and a set of equivalent 3D bases \bar{b}_i , we now need to derive the 3D shape corresponding to the current 2D shape. Furthermore, we would like to constrain the 2D shape such that only those deformations and pose changes are allowed which can be represented by a valid 3D shape. The constrained 3D extension described above can be achieved by performing an additional minimization. The term describing combined 2D/3D minimization now has the form

$$\sum_{x \in s_0} [A_0(x) + \sum_{i=1}^m \lambda_i A_i(x) - I(N(W(x, \mathbf{p}), \mathbf{q}))]^2 + K \cdot \left\| \underbrace{P(\bar{s}_0 + \sum_{i=1}^m \bar{p}_i \bar{b}_i)}_{=\bar{\mathbf{s}}} + \underbrace{\begin{pmatrix} o_x & \cdots & o_x \\ o_y & \cdots & o_y \end{pmatrix}}_{=\mathbf{o}} - \underbrace{N(s_0 + \sum_{i=1}^m p_i b_i; \mathbf{q})}_{=\mathbf{s}} \right\|^2 \quad (4.3)$$

where $\bar{\mathbf{s}}$ is an instance of the 3D shape, P is a scaled $3D \rightarrow 2D$ projection matrix, \mathbf{o} is an offset, and \mathbf{s} describes the current 2d shape instance after pose transformation. K is a constant, for $K \rightarrow \infty$ the constraints imposed on the 2D shape deformations and pose changes become *hard* constraints. Details on the 2D/3D extension are given in [41].

Unlike in the first minimization, Hessian and Jacobian matrices are no longer independent of the current input shape, but depend on the 2D shape and pose parameter sets. As a consequence, those matrices have to be recomputed in each iteration. However, the computation no longer needs to be performed on the whole image, but only on the vertices of the shape. Gradient descent geometries with respect to the 2D parameter sets cannot be computed in a closed form. [3] proposes a numerical estimation by determining the warp perturbations caused by small parameter changes. Analogue to the steepest descent method used for the 2D AAM, the steepest descent geometry of the 3D extension can be formulated, see appendix E for details.

We perform an initial calibration run to optimally align the internal 3D shape representation with the initial 2D fit, otherwise large differences in pose can cause 2D and 3D shapes to diverge. This is done by iterating the 3D fitting process until convergence with the 2D input shape is reached, with the only difference that during calibration iterations the feedback to the 2D shape is discarded.

In practice, the choice of the constant K is of crucial importance to tracking performance, since a small K may not be sufficient to completely suppress invalid 2D deformations, and a large K can significantly slow down and even prevent correct fitting.

5. Using Stereo Vision

Given a three-dimensional description of the Active Appearance Model, we would like to change its underlying model parameters in order to adapt the 3D model, and subsequently the 2D model, to the stereo data.

5.1 Obtaining and Structuring Stereo Data

In stereo vision, two cameras are looking at the same scene with different angles of view. The offset resulting from the different camera orientations is called disparity. Disparity is proportional to the distance of an object to the camera, greater disparity indicates that the object is nearer to the camera. Objects too far away do not yield a significant disparity and are attributed the disparity value 0. Appendix G describes stereo geometry and how to compute an object's distance to the camera using its disparity value.

In the given case, we need to efficiently calculate the distance between the 3D shape and the stereo depth map. Resolution of the depth map is typically much higher than the 3D shape resolution, it can be extremely noisy and it is likely to contain outliers. Moreover, we cannot be sure to always find depth information for a given point of the shape since stereo computation can fail due to lighting or lack of texture of a region, see appendix G for details. As a consequence, in each minimization step and for every point of the 3D shape we need to search for the stereo point closest to a given 3D shape vertex. The most efficient data structure offering spatial lookup of values is the KD-Tree, see appendix F for details. The KD-Tree is initialized with the depth values obtained by the camera, and allows for efficient searching of nearest neighbours to a given 3D point. For a set of 3D shape vertices, we need $O(v \log(n))$ to find corresponding nearest neighbours in the stereo data (where v denotes the number of vertices in the 3D shape and n is the number of points contained within the KD-Tree). In practical implementation, subsampling of the input point cloud can be performed in order to speed up generation of and search within the tree.

5.2 Fitting to Stereo Data

Analogue to 2D and 2D/3D fitting, we now wish to determine how well our 3D shape representation corresponds to the depth data obtained from the stereo camera.

Moreover, we would like to adapt the 3D shape and its underlying 2D shape to better fit the stereo data without noisy or erroneous stereo points affecting fitting quality. This goal can be achieved by introducing a third constraint minimization step. In this step, we wish to minimize

$$D = \left\| \underbrace{\left(\bar{s}_0 + \sum_{i=1}^m \bar{p}_i \bar{b}_i \right)}_{=\bar{s}} - \mathbf{c} \cdot CP_{3D}(\mathbf{c}^{-1} \underbrace{\left(\bar{s}_0 + \sum_{i=1}^m \bar{p}_i \bar{b}_i \right)}_{=\bar{s}}) \right\|^2 \quad (5.1)$$

where \bar{s} represents an instance of the 3D shape *after* applying the 3D pose transformation (we can easily compute the corresponding transformation matrix as a byproduct of the steepest descent 3D-extension described in chapter 4 and appendix E).

$CP_{3D}(\bar{s})$ is the closest-point lookup function in three dimensions providing the nearest neighbor to a given shape vertex. The closest-point lookup function CP needs to deal with outliers due to camera noise. When we assume that the initial 2D fitting is sufficiently precise, we can then define a threshold such that all pixels possessing a closest-point distance above this threshold are assigned the value 0, thus preventing erroneous depth values from influencing the fitted mesh. This threshold can either be determined as a fixed value (in practice, values around 10 pixels are reasonable), or it can be computed dynamically as a function of the average distance of the last fitted 3D mesh.

Finally, \mathbf{c} is a constant performing scale change from the camera coordinate system to the frame of reference of our 3D shape. However, \mathbf{c} can assume matrix form when a more general coordinate transformation is needed.

We chose to perform the above minimization using steepest gradient descent. The steepest descent approach to iteratively minimizing the above D can be formulated as follows:

$\bar{\mathbf{p}}_{3D}$ is the current complete 3D parameter set, and the initial value for the 3D parameter update $\Delta \bar{\mathbf{p}}_{\text{stereo}}$ is the value computed on the 3D extension level, $\Delta \bar{\mathbf{p}}_{3D}$. This value is then modified using

$$\Delta \bar{\mathbf{p}}_{\text{stereo}} = -H_{\text{stereo}}^{-1} \left(\Delta \bar{\mathbf{p}}_{3D} + G \cdot \sum_{k \in \{x,y,z\}} \sum_{i=1}^{\bar{n}} \left(\frac{\partial D_{k,i}}{\partial \bar{\mathbf{p}}} \right)^T D_{k,i}(\bar{\mathbf{p}}_{3D}) \right) \quad (5.2)$$

where each row of the Jacobi matrix of the distance function D has the form

$$\frac{\partial D_{k,i}}{\partial \bar{\mathbf{p}}} = \bar{b}_{k,i} - \frac{\partial CP_{k,i}}{\partial \bar{\mathbf{p}}} \quad (5.3)$$

where $\bar{b}_{k,i}$ denotes the components of the coordinate k in the i th 3D shape base vector \bar{b} . The contribution of the closest-point function CP is

$$\frac{\partial CP}{\partial \bar{\mathbf{p}}} = \frac{\partial CP}{\partial \bar{s}} \underbrace{\frac{\partial \bar{s}}{\partial \bar{\mathbf{p}}}}_{=\bar{\mathbf{b}}} \quad (5.4)$$

where $\bar{\mathbf{b}}$ denotes the 3D shape base vectors. Finally, the Hessian matrix is computed from

$$H_{\text{stereo}} = H_{3D,\bar{p}} + G \cdot \sum_{k \in \{x,y,z\}} \sum_{i=1}^{\bar{n}} \left[\frac{\partial D_{k,i}}{\partial \bar{\mathbf{p}}} \right]^T \left[\frac{\partial D_{k,i}}{\partial \bar{\mathbf{p}}} \right] \quad (5.5)$$

where \bar{n} denotes the number of 3D shape parameters and $H_{3D,\bar{p}}$ is the *submatrix* of the Hessian of the 3D extension step that corresponds to the 3D shape parameters (formed from $[\frac{\partial F_{i,t}}{\partial \bar{p}}]^T [\frac{\partial F_{i,t}}{\partial \bar{p}}]$).

The constant G allows weighting the contribution of the stereo fitting to the final 3D shape parameters similar to K in the 3D extension step. G is necessary to prevent erroneous stereo values from interfering with the fitting process: when using the 2D fitting as the basis of our algorithm, the stereo contribution should be limited to small mesh improvements with local impact instead of changing the global outline of the mesh. Increasing G results in extending the potential sphere of influence of the stereo fitting. We can describe this relation as *weak* resp. *strong coupling* of 2D- and 3D/stereo-fitting.

In our implementation, the above described steepest descent is performed *inside* the 3D extension step (which is described in appendix E) as follows: the new 3D shape parameter update $\Delta \bar{\mathbf{p}}_{3D}$ is fed into the above process *before* it is multiplied with the Hessian of the 3D extension H_{3D} . The modified 3D shape parameter update $\bar{\mathbf{p}}_{\text{stereo}}$ is then returned to the 3D extension together with H_{stereo} . H_{stereo} is added to the *submatrix* inside H_{3D} that corresponds to $\bar{\mathbf{p}}$. As a result, the update to the 3D parameters computed by the 3D extension is at the same time constrained to optimally fit the stereo data.

Since the result of the nearest neighbor search depends on the 3D shape, we have to reevaluate in each iteration the steepest descent geometries on the stereo data as well as on the 3D shape. The Jacobian on the stereo data, $\frac{\partial CP}{\partial \bar{\mathbf{p}}}$, can be interpreted as the influence of a small change $\Delta \bar{\mathbf{p}}$ to the 3D shape parameters on the computation of the nearest neighbour for each vertex of the 3D shape $\bar{\mathbf{s}}$. We suggest using the following approximation: we compose the shape corresponding to the current 3D parameters with a sequence of small changes, each of which has one parameter set to a small value and all other parameters set to 0. We then perform a nearest neighbor search for the resulting 3D shape and retain the change to the vertex locations. Unfortunately, this operation has to be performed at each iteration and for each 3D parameter once on all shape vertices. Figure 5.1 on page 18 illustrates the steepest descent geometry on the stereo data.

Changes to the 3D parameters on this level of the fitting process are now fed back to the 2D/3D extension computation and influence the internal 3D shape representation. This, in turn, modifies the 2D fit to better align not only with the 2D intensity image, but also with the 3D stereo data.

5.3 Overview of Performed Fitting Steps

The following figure 5.2 on page 18 summarizes once again which fitting steps have been performed and how the different levels influence each other. By varying the constants used, it is possible to tighten or loosen the strength of interaction between the different levels.

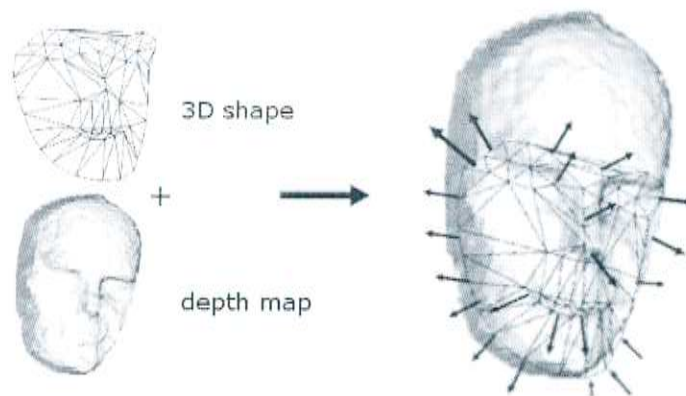


Figure 5.1: Steepest Descent Geometry on the depth surface

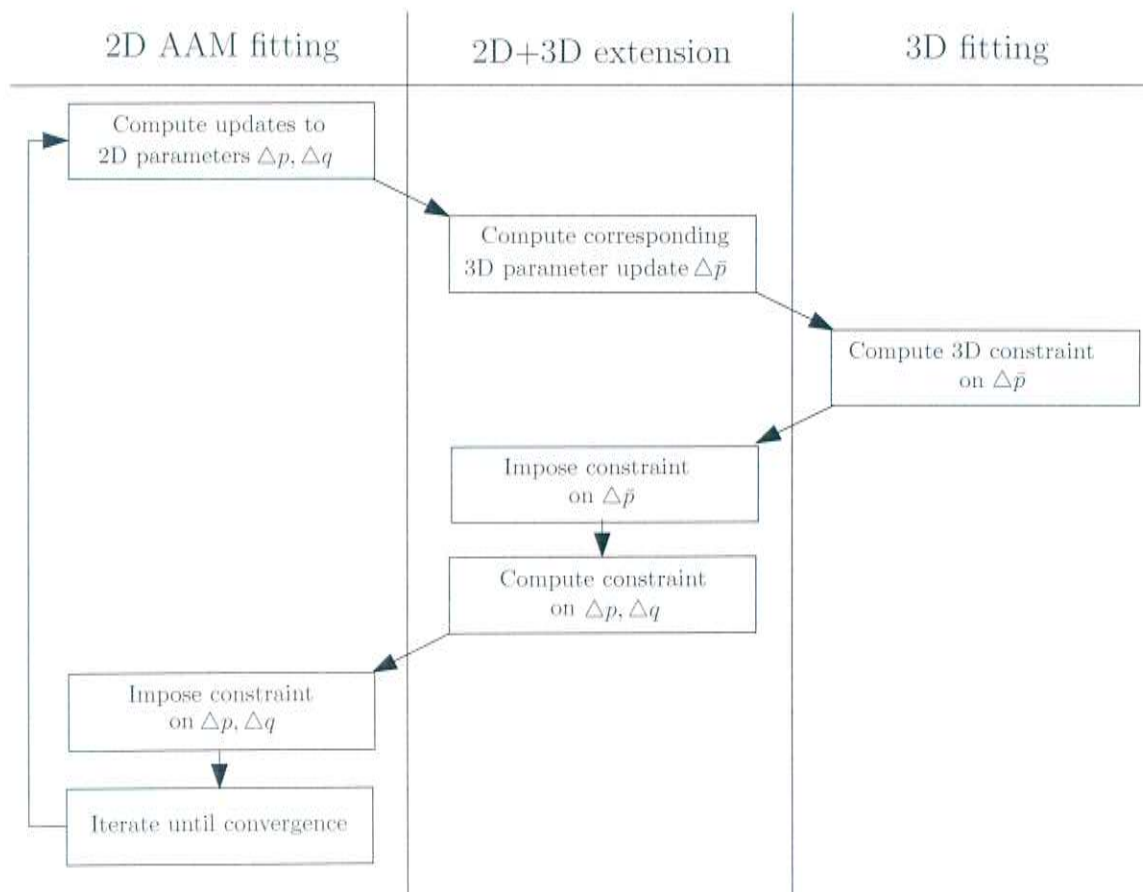


Figure 5.2: Interaction between different levels of fitting

6. Experimental Results

6.1 Generic Example

6.1.1 Tools Used

In order to test our method under well-defined, reproducible conditions, we implemented a testing tool allowing for the generation of faces using given 2D and 3D models. The user may specify the shape and pose parameters used and can thus reproduce all shape and appearance modes offered by the underlying models. The depth data is created using the depth values of the 3D model. In order to produce a smooth depth data surface between mesh vertices and around the face contours, we apply a smoothing filter and add Gaussian noise. The GUI of the corresponding tool is shown in figure 6.1 (the window on the right side). We now use the 2D image of the generated 3D face as input for our fitting algorithm. The initialization can be performed manually by the user. Since the exact shape and appearance parameters used for the creation of the generic face are known, we can easily compare them with the resulting parameter sets computed by our algorithm. Furthermore, we wish to compare fitting results of the 2D method and the 3D-extended stereo method. We therefore compute both fitted meshes and their corresponding residual errors with respect to the mesh based on the ground truth generic parameters. Errors are computed in pixels in city block distance between the 2D shape and the 2D-backprojected 3D shape. The left window of figure 6.1 shows the GUI of this tool, the two meshes are drawn in white resp. black.

6.1.2 Initialization

The images in the following paragraph 6.1.3 compare the fitting results visually. Each image contains three subimages. The leftmost image of each set shows the initialization, which is usually chosen to be the neutral average shape with a certain pose offset from the face center to test both shape and pose fitting. One has to take care to place the initial shape within the generic face, since our method relies on gradient descent on the grayscale values which is not possible on the perfectly homogeneous zero-gradient white background. Tests have shown that the fitting will

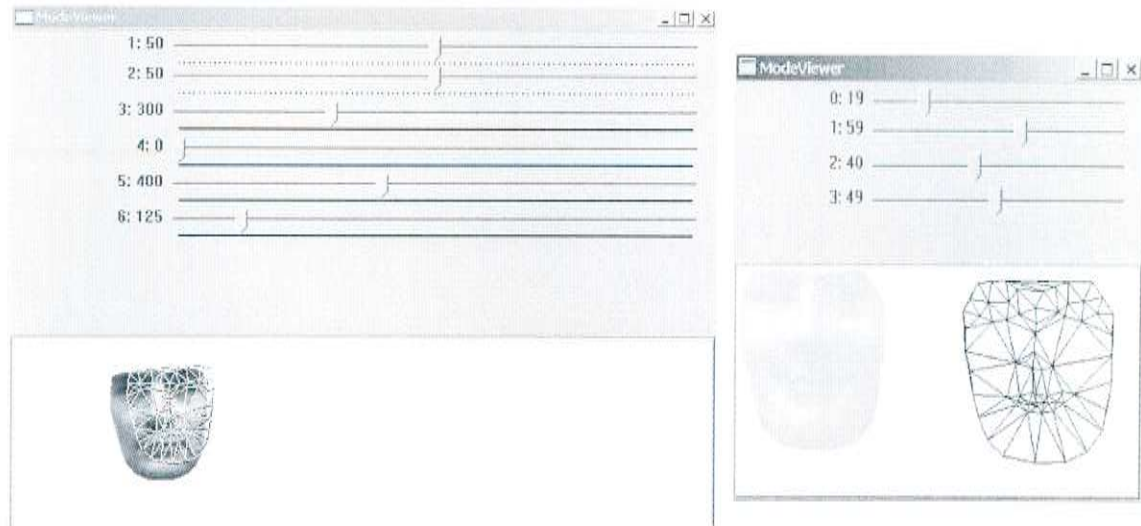


Figure 6.1: Ground truth testing tool GUI

fail if the initialization places more than 15% of shape vertices on a homogeneous background. This problem could be addressed by implementing an error function resistant to partial occlusions, as described in [15].

6.1.3 Comparison



Figure 6.2: Ground truth example; pose and scale

Figure 6.2 shows the fitting results for a generic face with neutral expression and an initial shape of 80% scale and 10% vertical and horizontal offset. 6 of the 68 shape vertices are initially placed outside of the generic face. The 2D fitted mesh is correctly aligned with the nose feature. However, fitting of the mouth, eyes and chin is not satisfactory. Its residual fitting error with respect to the generic parameters is 95 pixels (in city block distance). The 3D stereo-extended fitting results in a residual error of 69 pixels, thus reducing the error by ca. 27%. In particular the mouth region is well aligned and models the lip contours correctly. Still, the fitting of the eyelid position is not perfect. We have observed that the distance between camera and face under non-artificial ambient light results in shadows around the eyes which reduce contrast in this region significantly, thus greatly reducing the quality of the 2D model and consequently the fitting quality. For non-generic input, the stereo extension is

unable to achieve correct fitting in this area since poor contrast and sparse texture do not yield significant disparity values. We have not tested frontal lighting of the face since we deliberately intended to work in a realistic setting.

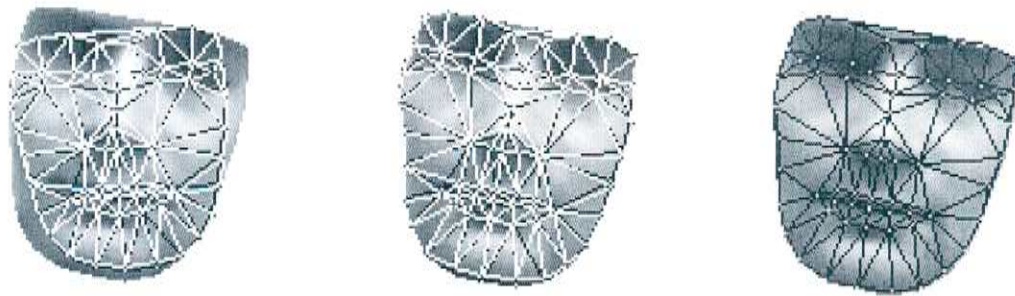


Figure 6.3: Ground truth example: rotation

Figure 6.3 shows test results for a generic face in neutral shape, but rotated by 10° to the right. 2D fitting achieves a residual error of 112 pixels, whereas 3D stereo-extended fitting converges with a residual of 78 pixels. Once again, the stereo data proves particularly useful for improving fitting of mouth and chin regions.

In general, extremely strong shape deformations (shown for example in the following figure 6.4) in conjunction with a large pose variation (for example a rotation of more than 15°) may cause our algorithm to fail: in those cases, the algorithm tends to converge to a local error minimum that does not correspond to an acceptable global fitting. It should be noted however that in realistic continuous fitting of an image sequence, pose variations as large as the one displayed in figure 6.3 hardly ever occur: even with a fitting rate as low as 5 frames per second, continuous fitting keeps inter-image differences small and initialization should be performed on a frame of the image sequence displaying a relatively neutral face in order to reduce initialization time.



Figure 6.4: Ground truth example: big smile

Figure 6.4 compares the fitting results for an extremely large shape deformation. A big smile does not only result in local deformations of the mouth, but changes the eye and external face contours as well as the texture significantly. It is therefore particularly difficult to fit, the more so as we work on a comparatively low image

resolution. For an initialization with 10% offset, 80% scale and additionally a slight 3D turn of 5°) to the right, 2D fitting converges with 191 pixels of accumulated residual error, mostly in the eye, mouth and chin region. It can be explained by the strong variation in appearance caused by the smile. By using depth information to improve the fitting, we are left with a residual error of 102 pixels. While this error is still significant, in particular in the eye region and along the external face contours, the achieved improvement is apparent. For non-generic input data, the fitting quality in these regions will typically be lower due to sparse depth data.

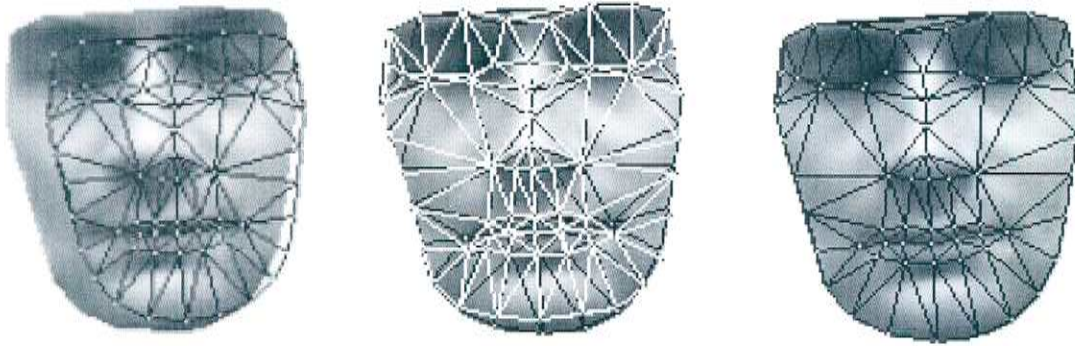


Figure 6.5: Ground truth example: eyes closed, initialization with 12% background outliers

Figure 6.5 illustrates the results of a test focussing on two aspects: the initialization contains 11% of vertices outside of the face image, and the eyes are completely closed. Both methods align well and detect the closing of the eyes. While the 2D fitting shows weaknesses in the chin and mouth regions, it does modify the shape parameters designating the closing of the eyes. However, the eye contours are not completely aligned with the closed eyelids and a residual error of 148 pixels remains. The stereo-extended fitting achieves excellent fitting of external contours and the mouth region while aligning correctly with the closed eyes, the remaining error is of 54 pixels, thus less than 1 pixel per shape vertex.

6.1.4 Table of Residual Errors

The following table 6.1 lists a series of experiments using the above described ground truth tool. All initializations have been identically chosen as a neutral shape, completely within the gray-scale generic face image, downscaled by 20% and offset to the right upper corner by 10% of the image size. The last line summarizes the average errors over all experiments. The descriptions mention the type of deformation and pose change present. For the model used in the experiments, the following 2 pose and 4 shape changes are possible: the pose change is additional to translation and scale, either a 2D rotation, a 3D head turn or both. The shape change can be a big smile (changing the complete facial appearance), closing the eyes, raising the eyebrows, opening the mouth or a combination of all 4 modes.

It should be noted that experiments 16 to 18 did achieve insufficient 2D convergence. The better 2D result for experiment 17 is simply due to a reduction in face surface due to the 3D turn. The 3D results of the combination experiments are not very satisfactory either. In general, fitting errors of more than 3 pixels per vertex are

Table 6.1: Residual errors during generic ground truth experiments

experiment			residual error [pixels]		improvement [%]
No.	pose	shape	2D	stereo- extended 3D	
1	neutral	neutral	95	69	27.4
2	neutral	smile	191	102	46.6
3	neutral	eyes	148	54	63.6
4	neutral	brows	64	58	9.4
5	neutral	mouth	141	88	37.6
6	rotation	neutral	152	108	29
7	rotation	smile	233	196	25.9
8	rotation	eyes	174	129	25.9
9	rotation	brows	180	136	24.5
10	rotation	mouth	182	144	20.9
11	turn	neutral	176	124	29.6
12	turn	smile	242	210	13.3
13	turn	eyes	185	132	28.7
14	turn	brows	197	170	13.8
15	turn	mouth	184	153	16.9
16	rotation	all 4	254	198	22.1
17	turn	all 4	248	214	13.8
18	turn+rot	all 4	304	239	21.4
			...		
average over 50 exp.			176.1	135.2	23.3

not admissible, given the low resolution of the input image. We have not yet been able to determine the cause of this behavior, but we suppose that the underlying 2D model needs to be improved when complex combined shape deformations are concerned. Still, the stereo extended fitting greatly improves combined shape deformations even under strong pose changes. The observations confirm that 3D fitting works particularly well for the mouth region, since even small mouth movements usually result in stronger changes in depth data.

6.2 CHIL Recordings

During a sequence of recorded staged meetings for the ongoing *CHIL* project [33] of the *ISL*, we collected video and depth data of several participants. The emphasis of the recordings was to test our system under realistic conditions showing fast head movements and authentic expressions, in different environment settings, and under varying camera distances and angles and thereby identifying the limitations of our current system.

Figure 6.6 on page 24 shows examples of successful fitting of the same 2D model to 3 different faces by incorporating stereo information and thus improving the fitting quality. The model has been created for the person on the left image, but our stereo extension is able to adapt it to completely different facial geometries and skin

textures and even to counter distortions due to the glasses worn by the person in the right image. During these tests, we implemented a simple color normalization since the videos have been recorded at different times and with changing artificial light. The normalization had to be adapted manually for each sequence. The results of the color normalization were acceptable, but could not prevent a high residual fitting error and slight overfitting. We conclude that color normalization is insufficient and that lighting changes need to be modeled as part of the error function.

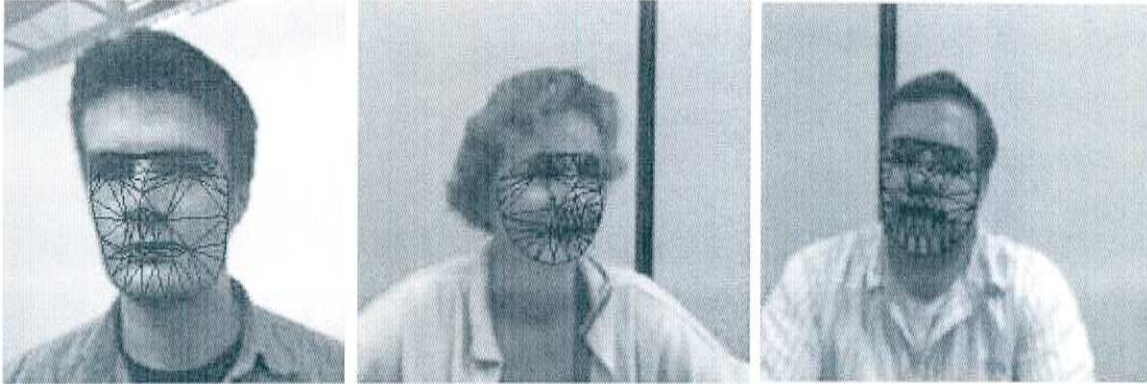


Figure 6.6: CHIL meeting recordings: successful fitting

Figure 6.7 on page 25 shows some examples for the limitations of our current system.

The image on the left illustrates failed convergence as a consequence of fast pose variation of more than 15° . The fitting algorithm accounts for changes in the feature positions correctly, but cannot adapt the pose variation (in this case a lowering of the head). This is due to the fact that translating the face results in a faster reduction of the error in the beginning of the fitting process than varying the shape parameter corresponding to a head nod. We believe that the quality of the 2D model creation, particularly the shape alignment part, has to be improved in order to address this issue.

The image in the center of figure 6.7 shows imprecisions caused by partial occlusion of more than 20% of the face surface. This problem occurred during nearly one third of the recording time. Since the 2D model error function does not explicitly deal with occlusions and the stereo data available for the occluded areas is either not present or displaying strong variations and is thus ignored, the remaining data does not allow for a sufficient fitting. Changing the 2D error function and improving stereo distance computation, for example with an enhanced iterative reweighing method, could be a possible remedy to this problem.

Finally, the image on the right of figure 6.7 shows imprecise fitting results due to an extremely low resolution of both the 2D face image and the corresponding stereo data, taken at a camera distance of $2m$. The 2D model without 3D extension usually fails to converge on face surfaces of less than 150×150 pixels. Moreover, since we have shown in chapter G.1 that at a distance of $2m$ from the camera we can expect a depth computation error of up to $5cm$, we have to concede that this problem can only be solved by increasing the stereo camera resolution of currently 640×480 pixels. However, a higher input resolution will inevitably result in a reduced fitting

speed. As a consequence, we will have to develop a more efficient 3D gradient descent method in order to compensate for the increased input resolution.



Figure 6.7: CHIL meeting recordings: unsolved issues

6.3 Application Example: Facial Expression Classification

One possible application of our system is facial expression classification. Since we use a model-based approach, the complex problem of describing a facial expression is reduced to the choice of a simple parameter vector. Depending on the complexity of the model chosen, 7 to 12 parameters are sufficient for reliably describing most elementary facial deformations and all possible pose changes. The classification task thus consists of assigning a class label to a set of model parameters for each frame of the input video.

In order to further simplify the problem, we have chosen to classify the parameter vector of each video frame independently of previous frames and parameters. Although as a consequence we lose the transition context of a given facial expression, we will show that due to the tracking precision of our 3D-extended system, we can still achieve a surprisingly precise classification. Nevertheless, we concede that context-aware classification will certainly lead to further improvements, see chapter 7.1 for suggestions of future work.

We have tested the proposed classifier on elementary facial expressions as well as mouth movements corresponding to vowels. Each two different facial expressions shown in the training input video are separated by a *neutral* intermediate expression. Whilst this might not be a realistic training setup and can therefore lead to difficulties in correctly classifying expression transitions, it greatly simplifies manual training labeling. See chapter 7.1 for a discussion of how to incorporate expression transitions into the classifier.

Figure 6.8 on page 26 shows a confusion plot comparing 4 different pairs of emotion classes (modeled after simplified FACS), confined to a twodimensional parameter space. The image on the upper left plots the *surprise*-class (plotted as stars) against the *joy*-class (plotted as rectangles), using only the parameters corresponding to eyebrow and mouth corner movement. Since separate muscle groups are creating

the corresponding expressions, the parameters describing these movement bases can be linearly separated even in input space. The other three images show class plots of sorrow/fear, surprise/fear and anger/sorrow for different parameter pairs. Since those emotions involve movements of similar muscle groups and corresponding shape bases, one can easily see that a linear separation of the data in input space is unlikely to produce satisfactory results.

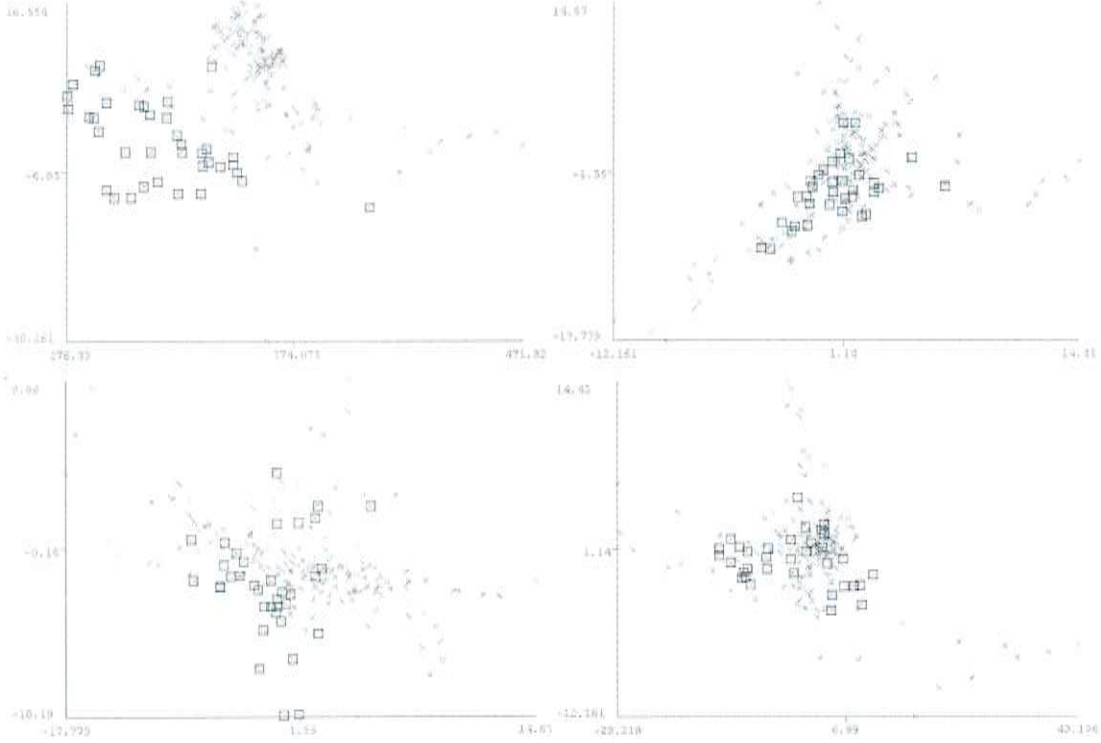


Figure 6.8: Scatterplot of pairs of classes for a 2D parameter space

6.3.1 Support Vector Machines

6.3.1.1 General Outline

Given a sequence of tracking result parameters hand-labeled with their corresponding classes, we would like to train a classifier such that the classes of unseen tracking results can be identified correctly. The classification task therefore consists of choosing a decision function capable of separating the training input into classes with as much precision as possible, i.e. by maximizing their inter-class distance (called *Maximal Margin Classifier*). If this decision function has been established such that it complies to certain conditions [37] [6], we can then suppose that its separation performance generalizes well to unseen data.

Support Vector Machines (SVM) [37] are one technique to perform this task. Assuming for example the simple two-class case, a set of l labeled training vectors of dimension n , $\mathbf{x}_i \in \mathbb{R}^n, i = 1 \dots l$ and a vector of training labels $\mathbf{y} \in \mathbb{R}^l$ with each $y_i \in \{1, -1\}$, the decision function has the form

$$y_i = h(\mathbf{x}_i, \mathbf{w}, b) = \text{sgn}(\langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b) \quad (6.1)$$

with \mathbf{w}, b denoting the parameters of a separating hyperplane in feature space. The problem can be extended to higher dimensional input data in an analogous manner.

SVMs are usually trained using some algorithm from Lagrangian optimization theory. [6] suggest using n -fold cross-validation on the training sets to improve training efficiency and to avoid overfitting of the classifier. During cross-validation, the training data is divided into n subsets of equal size, the classifier is then sequentially trained on $n - 1$ subsets and tested on the remaining subset.

When the input data cannot be separated linearly, a kernel function of the form

$$K(x, z) = \langle \phi(\mathbf{x}) \cdot \phi(\mathbf{z}) \rangle \quad (6.2)$$

with ϕ as a nonlinear mapping is used to embed the input vectors into a higher-dimensional feature space. Thus, nonlinear separation in input space is avoided and hyperplanes can be used to separate data in feature space. Several kernels have been proposed, notably linear, polynomial, radial basis and sigmoid functions.

6.3.1.2 Specifications

For our task, we have decided upon using a Radial Basis Function (RBF) as the SVM kernel. The RBF has the form

$$K(\mathbf{x}, \mathbf{z}) = e^{-\mu \|\mathbf{x} - \mathbf{z}\|}, \mu > 0 \quad (6.3)$$

All parameter values are linearly scaled to $[-1, 1]$ in order to prevent attributes in greater numeric ranges from dominating those in smaller ranges as well as to avoid numerical instabilities during computation. This scaling procedure has to be performed on training as well as on unseen testing data. Training is performed using tenfold cross-validation on a training sequence of 250 to 300 parameter state vectors, including all face deformations allowed by the chosen model in the 4 basic facial pose variations of scale, rotation, turn and nod. The classifier is then tested on a sequence of usually 2 minutes of previously unseen input video. Since our fitting algorithm is not yet able to deal with strong lighting changes, we used testing data collected immediately after the training data under the same environment conditions. Each classifier is inherently associated with one single facial expression model, since the meaning of the describing parameter set depends on the choice of the shape and appearance bases.

6.3.2 Results

We have tested our classification system on two different models. Training precision is described as a confusion matrix, while the testing results are compared to ground-truth manual labeling.

6.3.2.1 Basic Emotions

Our first model describes the 12 most frequently occurring FACS action units AU 1, 4, 5, 6, 7, 9, 10, 12, 15, 17, 20, 23 [8], its associated classifier differentiates between 6 combinations of those action units, thus recognizing surprise, joy, sorrow, fear, anger and disgust in addition to a *neutral* mean-shape, mean-appearance facial expression. The interpretation of those combinations has to remain subjective, however we have tried to use combinations already introduced in previous publications [20].

Table 6.2: Training confusion matrix for emotion expressions, precision: 86.9%

	neutral	surprise	joy	sorrow	fear	anger	disgust
neutral	62	1	2	0	0	0	0
surprise	0	17	1	2	0	0	0
joy	0	0	29	3	0	1	0
sorrow	2	0	2	19	1	0	1
fear	3	0	0	0	18	2	0
anger	1	1	0	0	2	44	1
disgust	0	0	0	3	0	3	24

Table 6.3: Classification of emotion expressions in unseen data (user estimation), precision: 71.3%

emotion	detection rate
neutral	75%
surprise	68%
joy	76%
sorrow	63%
fear	61%
anger	74%
disgust	82%
avg.	71.3%

Figure 6.2 shows the confusion matrix for training of a SVM on emotion expressions. Separation precision of 86.9% is satisfactory, although some emotions are inherently difficult to differentiate when no temporal context is provided.

Classification precision of 71% on unseen data (estimated by the test person), listed in figure 6.3, is acceptable, despite the simple context-unaware approach used. Some emotions, in particular the emotion pairs sorrow/fear and surprise/fear are inherently difficult to separate, since they are based on similar facial muscle movements, while other emotions such as disgust result in characteristic texture and shape changes making them very easy to apprehend. Improvements can be expected from incorporating more sophisticated context-aware classification, but even so we doubt that the above outlined difficult cases can be dealt with more reliably.

6.3.2.2 Mouth Movements: Vowels

Our second model describes 5 characteristic lip movements corresponding to the vowel sounds a,e,i,o and u. While training has been performed on isolated vowels, testing sequences included spoken words, albeit using exaggerated, overly expressive mouth movements. The goal of these tests was not to attempt lipreading, but to emphasize the high detail precision that our stereo-extended tracking approach is capable of achieving.

Figure 6.4 shows the confusion matrix for training data classification (N signifies a neutral mouth position). Separation of vowels o and u proved particularly difficult, but the overall training precision of 87.5% is acceptable.

Table 6.4: Training confusion matrix for mouth movements (N=neutral), precision: 87.5%

	N	a	e	i	o	u
N	13	0	0	0	0	0
a	0	24	0	0	0	0
e	0	1	13	0	0	0
i	0	0	0	9	0	0
o	0	0	0	0	17	0
u	4	0	0	0	8	15

Table 6.5: Classification of mouth movements in unseen data (user estimation), precision: 73%

vowel	detection rate
N	76%
a	81%
e	75%
i	73%
o	68%
u	65%
avg.	73%

The results for unseen data (estimated by the test person) as shown in figure 6.5 show a reasonably good average detection rate of 73%, given the absence of context information in our approach. However, the classifier failed in reliably differentiating between the vowels o and u, since the small differences in lip positions were extremely difficult to detect. Enhancing the classification and providing a different type of input data (as suggested in chapter 7.1) is likely to improve classification in this case.

7. Conclusion and Future Work

7.1 Problems Encountered and Future Work

7.1.1 Twodimensional Model

The approach to tracking deformable shapes outlined in this work has the advantage of requiring the creation of a 2D model, instead of a difficult and error-prone 3D training necessary for direct 3D tracking methods. However, even the creation of a simple twodimensional model contains a number of possible pitfalls. Whilst the 3D extension can improve fitting results of face details, the threedimensional steps of our method still rely heavily on sufficient quality of the 2D tracking as a rough initialization.

One of the most difficult parts of our work was the choice of training data for 2D model creation. As mentioned before, the deformations present in the training data are not only of crucial importance for the *type* of deformations the final 2D model will be able to track, but they also determine *how* these deformations will be represented by the resulting bases. If the training data contains only a sequence of single muscle movements (such as the raising of the eyebrows without any other simultaneous muscle action), then the resulting bases are likely to represent one of those movements per base. However, since in reality muscle actions never occur separately, but always in combination with other movements, the resulting 2D model may contain too many separate bases. This will slow down fitting speed and may cause ineffective steepest descent: supposing that some bases are strongly present in the training data, but hardly ever occur separately in reality. These bases will cause the model to converge along certain dominant gradient directions whilst other gradient directions will only contribute to a lesser degree since they are simply not present or under-represented in the training data. However, these other gradient directions might have resulted in a better overall model fitting. Further work will have to focus on establishing criteria to objectively evaluate training data quality and significance, before any improvements to the fitting process itself can be attempted.

Another crucial point was model sensitivity towards changes in lighting conditions. The presence of indoor artificial light caused significant changes in the color spectrum. When a model was created exclusively using video data under natural light

conditions, it was certain to fail under artificial light since the error between best fit and current input remained too important, thus causing the model to converge towards incorrect instances. Several authors have proposed approaches to incorporating these variations into the model itself or into the error function [15]. While these suggestions were beyond the scope of our present work, their evaluation and integration into future systems will be necessary in order to overcome the described difficulties.

Similar aspects concern shadows and occlusions, notably self-occlusions caused by large head turns. A realistic face model will have to be designed to support view changes from frontal to profile view, adapting internal model representations accordingly without requiring manual intervention. Some work has been dedicated to modeling different views separately [26], but no conclusive strategies for dealing with view transitions have emerged so far. We are of the opinion that stereo-assisted Active Appearance Models can contribute to a solution since landmark equivalences between different views can be established with greater reliability and better precision.

7.1.2 Stereo Extension

The stereo processing itself has shown several shortcomings. Among the most frequent problems encountered were disparity values varying as a function of lighting conditions and the fact that non-textured or similarly textured areas failed to yield significant disparity to allow for stable processing. For the given application task of tracking facial expressions, the areas most frequently affected were the cheeks (lack of texture) and the eye region (texture not visible due to shadows). While filtering and interpolation can counter some of these effects, we would like to test wider baseline stereo in order to compare precision in those areas particularly important for accurate facial expression detection.

The results of our test implementation are sufficient as a *proof of concept*, but the approach chosen for performing gradient descent on the stereo data is far from efficient. As part of future work on this topic, a more efficient gradient descent algorithm ought to be developed. For image alignment algorithms, [1] have shown that conventional approaches can be significantly improved by reformulating the corresponding gradient descent problem. We are convinced that a similar redesign of our stereo alignment method will speed up the fitting process, thus improving fitting quality.

7.1.3 Classification

As outlined in chapter 6.3, our support vector machine classification leaves room for improvement. We have not done an exhaustive comparison of our SVM approach with other classification methods.

Moreover, we do not use context information, but instead classify each parameter state vector independently of previous states. Clearly, by incorporating parameter states of previous video frames into the classification process of the current frame, a considerably improved classification precision can be expected.

Furthermore, it might be worthwhile using expression changes, i.e. transitions between parameter states, instead of the parameter states themselves as input data.

In our classification tests, we analyzed facial expressions that began and ended with a neutral face expression. For realistic tasks, the classifier should be able to deal with any direct transition between two expressions. Modeling these transitions as a stochastic process, for example as a Markov process, might be an interesting solution.

Alternatively, an unsupervised classification approach is likely to yield interesting results. By observing a sequence of facial deformations represented by the corresponding model parameter sets, recurring patterns of those parameters, after suitable transformation, can be clustered and automatically labeled as belonging to the same class of facial expression. As a consequence, tedious and imprecise manual training labeling would no longer be necessary. While this classification method would not obey the FACS standard, it could be intriguing to compare the automatically obtained classes and the FACS suggestions in order to determine whether the FACS *Action Units* (see [8]) are capable of reliably describing facial expressions under realistic conditions. Additionally, obtaining automatically labeled classes could be useful for online adaptation of the input model to the actual facial deformations present in previously unseen video sequences.

7.2 Evaluation

Given the results obtained during experiments, we can conclude that the proposed integration of stereo data fed back into 2D AAM creation can significantly improve precision and stability. By combining ease of model creation and tracking reliability, the described method could serve as a means of improving usability of Active Appearance Models in real-life applications. However, more work is required to tackle the remaining issues, some of which have been mentioned in section 7.1. Moreover, a unifying framework of all existing improvements to AAMs could prove extremely valuable, given the vast range of publications on this topic. The stereo extension could neatly integrate into such a framework as a means of providing quality feedback and continuous online model improvement and adaptation.

In our opinion, the most promising fields of application of the approach presented in this work are behavioral monitoring in healthcare environments [16] and lipreading as a way of improving context awareness for speech understanding [32]. In both domains, stable detection of small facial movements under realistic conditions are required, which can be achieved using our stereo extension to 2D Active Appearance Models.

A. Histogram-Based Face Segmentation

Histogram-based color segmentation is a suitable means of identifying image regions possessing characteristic colors. A manual initialization provides a set of colors characteristic of the region to be segmented, which is used to build a histogram counting the occurrence of each color in the region of interest. Using this histogram H_{roi} , the back-projection of any input image can be computed, assigning to each pixel the probability of its belonging to a region of interest

$$P(x|roi) = \frac{H_{roi}(x)}{N} \quad (\text{A.1})$$

where $x = (red, green, blue)$ is a color pixel, N the total number of pixels in the histogram and H_{roi} a function returning the number of occurrences of the color of pixel x in the region chosen for initialization. While this method is unable to track a region over a longer period of time due to changing lighting conditions, it is well suited for providing an initialization during a short period of time.

Based on the back-projection, we then derive additional information on the characteristics of the region of interest by computing the following moments of the region, where $BP(x, y)$ denotes the back-projected probability image:

- mean region size: $size_m = \sum_x \sum_y BP(x, y)$
- mean region center: $x_m = \frac{\sum_x \sum_y x BP(x, y)}{size_m}$, $y_m = \frac{\sum_x \sum_y y BP(x, y)}{size_m}$,
- mean region orientation using second moments analogously.

B. Principal Components Analysis

In the following, we give a rough outline of Principal Components Analysis (PCA, Karhunen-Loeve-Transform) with application to the problem of shape representation. Details can be found in [10], [31].

B.1 Goal

We have a set of m planar shapes (in our case 2D), each consisting of n vertices or landmarks s_i , as introduced in section (2.1):

$$\mathbf{s} = (s_0^x, s_1^x, \dots, s_{n-1}^x, s_0^y, s_1^y, \dots, s_{n-1}^y) \quad (\text{B.1})$$

Since we know that the vertices describe deformations of face landmarks who are confined to a number of possible movements of the underlying facial muscles, and since we further know that each facial muscle regroups a set of landmarks, we can assume that the locations of the moving shape vertices are correlated up to a certain degree. In order to obtain a simplified description of those vertex movements, we can therefore try to exploit the existing correlation and reduce dimensionality of the problem description.

B.2 Method Description

Given the m zero-mean shapes \mathbf{s} , their mean shape $\bar{\mathbf{s}} = \frac{1}{m} \sum_{i=1}^m s_i$ and their (symmetric) covariance matrix

$$\Sigma_s = \frac{1}{m} \sum_{i=1}^m (\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T \quad (\text{B.2})$$

we wish to achieve a linear, orthogonal transformation of the input shapes \mathbf{s} such that

$$\mathbf{r} = \Lambda \mathbf{s} \quad (\text{B.3})$$

with $\Lambda^T = \Lambda^{-1}$. The mean of \mathbf{r} is consequently $\bar{\mathbf{r}} = \Lambda\bar{\mathbf{s}}$. Covariance of \mathbf{r} follows as

$$\begin{aligned}
 \Sigma_r &= \frac{1}{m} \sum_{i=1}^m (\mathbf{r}_i - \bar{\mathbf{r}})(\mathbf{r}_i - \bar{\mathbf{r}})^T \\
 &= \frac{1}{m} \sum_{i=1}^m \Lambda(\mathbf{s}_i - \bar{\mathbf{s}})(\Lambda(\mathbf{s}_i - \bar{\mathbf{s}}))^T \\
 &= \Lambda \left(\frac{1}{m} \sum_{i=1}^m (\mathbf{s}_i - \bar{\mathbf{s}})(\mathbf{s}_i - \bar{\mathbf{s}})^T \right) \Lambda^T \\
 &= \Lambda \Sigma_s \Lambda^T
 \end{aligned} \tag{B.4}$$

Using the definition of our transformation, we have

$$\Sigma_s \Lambda^T = \Lambda^T \Sigma_r \tag{B.5}$$

If Λ^T represents the eigenvectors of Σ_s , then Σ_r has diagonal form

$$\begin{pmatrix} \lambda_0 & & & \\ & \lambda_1 & & \\ & & \dots & \\ & & & \lambda_m \end{pmatrix} \tag{B.6}$$

where the λ_i are the corresponding eigenvalues. The dimensionality of \mathbf{r} can now be reduced by omitting eigenvalues close to zero and their corresponding eigenvectors. It is also possible to truncate those dimensions which contribute less than a certain percentage to the overall model variation.

By inverting B.3 to $\mathbf{s} = \Lambda^T \mathbf{r}$ and assuming non-zero-mean shapes \mathbf{s} , we finally obtain

$$\begin{aligned}
 \mathbf{r} &= \Lambda(\mathbf{s} - \bar{\mathbf{s}}) \\
 \mathbf{s} &= \bar{\mathbf{s}} + \Lambda^T \mathbf{r}
 \end{aligned} \tag{B.7}$$

C. Image Warping

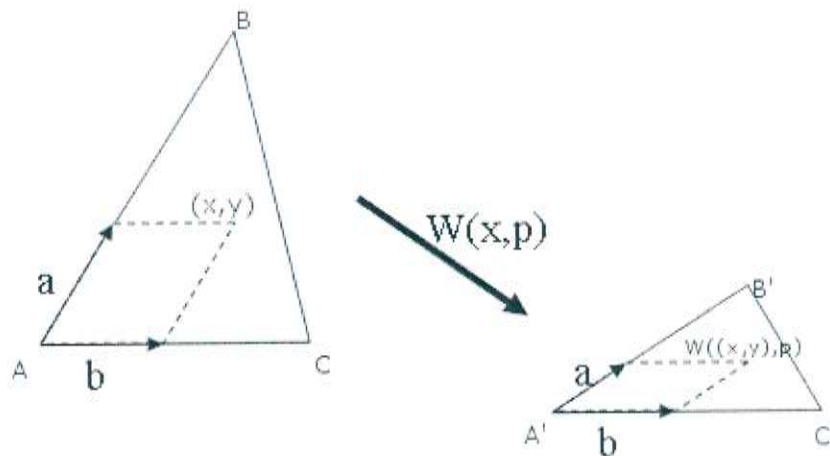


Figure C.1: Warping between two mesh triangles

C.1 Piecewise Affine Warp

Given two triangle meshes positioned at two image locations, we wish to copy the image region underneath the first mesh to the image region underneath the second mesh. Each mesh is the result of triangulating the shape vertices of a shape created from a parameter set \mathbf{p} . The location of each pixel underneath a mesh can be described relative to the *reference frame* determined by the triangle containing the pixel, as illustrated in figure C.1. Assuming the triangle origin at vertex $A(x_A, y_A)$, the pixel location (x, y) is a function of vertices $A(x_A, y_A)$, $B(x_B, y_B)$ and $C(x_C, y_C)$ as well as the relative coefficients a and b given by

$$(x, y) = (x_A, y_A) + a[(x_B, y_B) - (x_A, y_A)] + b[(x_C, y_C) - (x_A, y_A)] \quad (\text{C.1})$$

where the coefficients are given by the relative distances such that

$$a = \frac{(x - x_A)(y_C - y_A) - (y - y_A)(x_C - x_A)}{(x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A)} \quad (\text{C.2})$$

and

$$b = \frac{(y - y_A)(x_B - x_A) - (x - x_A)(y_B - y_A)}{(x_B - x_A)(y_C - y_A) - (y_B - y_A)(x_C - x_A)} \quad (\text{C.3})$$

The warp of pixel (x, y) into the second mesh triangle should conserve the relative location of the pixel within the triangle, as defined by the coefficients a and b . Therefore we can adopt the warp description in [3] as

$$W((x, y), \mathbf{p}) = (x_{A'}, y_{A'}) + a[(x_{B'}, y_{B'}) - (x_{A'}, y_{A'})] + b[(x_{C'}, y_{C'}) - (x_{A'}, y_{A'})] \quad (\text{C.4})$$

where $A'(x_{A'}, y_{A'})$, $B'(x_{B'}, y_{B'})$ and $C'(x_{C'}, y_{C'})$ are the vertices defining the second triangle.

C.2 Warp Composition

Since the image alignment methods described in section 3.2 work on the template image rather than directly on the current input image, the update to the warp parameters is relative to the *base shape* on the template image. However, the current warp $W(x, \mathbf{p})$ is relative to the *destination shape* on the current input image. In order to perform the warp composition $W(x, \mathbf{p}) \circ W(x, \Delta \mathbf{p})^{-1}$ to the new destination shape, we have to determine the warp update to the current destination shape that is equivalent to the template warp update. As soon as the new destination shape has been determined, we can invert the warp formula and extract the new warp parameters. [3] suggests computing this new destination shape by applying the warp update to each of the current destination shape vertices. Since the resulting new vertex location can lie outside its own reference triangle, the warp would be undefined. We therefore apply the warp update to all triangles containing the vertex that we wish to update and then compute the average location over all the resulting locations.

D. Inverse Compositional Image Alignment

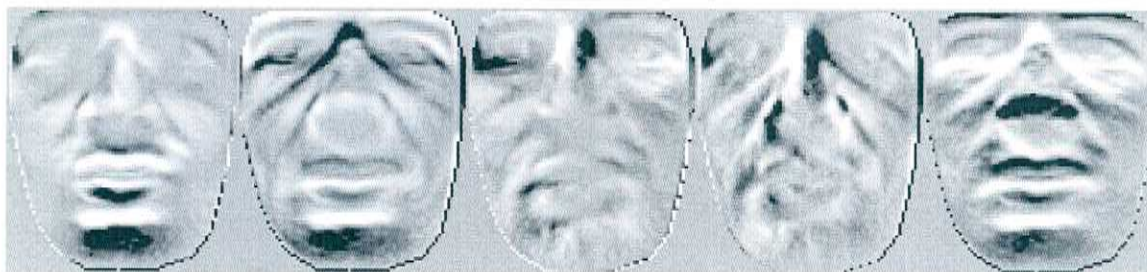


Figure D.1: Steepest descent images (contrast enhanced for visualization)

D.1 Algorithm

Using the project-out ICIA as described in section 3.2, the outline of the 2D AAM fitting algorithm now consists of the following steps, as suggested in [1]:

- **Pre-computation:**
 - Evaluation of the gradient of the template, ΔA_0 ,
 - Computation of the Jacobians for shape $\frac{\partial W}{\partial \mathbf{p}}$ and pose warp $\frac{\partial N}{\partial \mathbf{q}}$ with respect to the template image,
 - Computation of the steepest descent images for pose and shape parameters, SD , in an appearance-independent subspace (an example set of steepest descent images is shown in figure D.1 on page 41),
 - determine the Hessian matrix providing normalization of the steepest descent images.
- **Iteration:**

- perform an image warp of the current input image I using $N(W(x, \mathbf{p}), \mathbf{q})$,
- determine the error between current best fit and current input image

$$E(x) = I(N(W(x, \mathbf{p}), \mathbf{q})) - A_0(x) \quad (\text{D.1})$$

- compute the steepest descent parameter updates $\Delta \mathbf{p}$ and $\Delta \mathbf{q}$ from

$$(\Delta \mathbf{p}, \Delta \mathbf{q}) = H^{-1} \sum_{x \in s_0} (SD_i(x) \cdot E(x)) \quad (\text{D.2})$$

- compose the current and the updated warp

$$(N \circ W)(x, \mathbf{p}, \mathbf{q}) \leftarrow (N \circ W)(x, \mathbf{p}, \mathbf{q}) \circ (N \circ W)(x, \mathbf{p}, \mathbf{q})(x, \Delta \mathbf{p}, \Delta \mathbf{q})^{-1} \quad (\text{D.3})$$

There is no closed form for this step, the *approximating* solution is detailed in appendix C.

Although not required, the optimal appearance parameters λ can finally be determined using equation (3.5).

D.2 Steepest Descent Geometry

The above described algorithm requires computation of image gradients, Jacobians as part of the steepest descent images and Hessian matrices. Their computation is given below, as described in [3].

The Jacobians for pose warp $N(x, \mathbf{q})$ on the template image has the form

$$\frac{\partial N \circ W}{\partial \mathbf{q}} = \frac{\partial N}{\partial \mathbf{q}} = \sum_i \left[\frac{\partial N}{\partial x_i} \frac{\partial x_i}{\partial \mathbf{q}} + \frac{\partial N}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{q}} \right] \quad (\text{D.4})$$

where, using the warp formula C.4, we obtain

$$\frac{\partial N}{\partial x_i} = (1 - a - b, 0) \quad (\text{D.5})$$

(similar for y_i), and from the pose definition 3.3 follows

$$\frac{\partial x_i}{\partial \mathbf{q}} = (b_{x,1}^*, b_{x,2}^*, \dots, b_{x,n}^*) \quad (\text{D.6})$$

where $b_{x,i}^*$ is the x component of the pose base b_i^* (analogue for y_i). The same applies for the template shape warp $W(x, \mathbf{p})$

$$\frac{\partial N \circ W}{\partial \mathbf{p}} = \frac{\partial W}{\partial \mathbf{p}} = \sum_i \left[\frac{\partial W}{\partial x_i} \frac{\partial x_i}{\partial \mathbf{p}} + \frac{\partial W}{\partial y_i} \frac{\partial y_i}{\partial \mathbf{p}} \right] \quad (\text{D.7})$$

where, using the warp formula C.4, we obtain

$$\frac{\partial W}{\partial x_i} = (1 - a - b, 0) \quad (\text{D.8})$$

(similar for y_i), and from the shape definition 2.4 follows

$$\frac{\partial x_i}{\partial \mathbf{p}} = (b_{x,1}, b_{x,2}, \dots, b_{x,n}) \quad (\text{D.9})$$

where $b_{x,i}$ is the x component of the shape base b_i (analogue for y_i).

The steepest descent images (figure D.1 on page 41 shows an example of the steepest descent images for a simplified face model with 4 pose and only 1 shape variation modes) are computed by

$$SD_j(\mathbf{x}) = \nabla A_0 \frac{\partial W}{\partial p_j} - \underbrace{\sum_{i=1}^M \left[\sum_{x \in s_0} A_i(\mathbf{x}) \cdot \nabla A_0 \frac{\partial W}{\partial p_j} \right]}_{\text{project-out}} A_i(\mathbf{x}) \quad (\text{D.10})$$

where A_0 is the mean appearance image and the A_i are the M appearance variation modes. The last part of equation (D.10) projects the steepest descent images into the subspace orthogonal to the appearance variation, thus avoiding a continuous appearance recomputation for each iteration of the algorithm described in section D.1. The steepest descent images for the pose warp N relative to the pose parameters \mathbf{q} can be computed analogue to equation D.10.

Using the steepest descent images, each element of the Hessian matrix is then given by

$$H_{j,k} = \sum_{x \in s_0} SD_j(\mathbf{x}) \cdot SD_k(\mathbf{x}) \quad (\text{D.11})$$

E. 3D Extension

E.1 Algorithm

We follow the algorithm structure suggested by [41]. The algorithm for imposing 3D constraints on the 2D AAM requires the presence of a set of 3D bases describing the 2D model behavior equivalently. It iteratively computes updates to the 6 parameters

$$\mathbf{p}, \mathbf{q}, \bar{\mathbf{p}}, \mathbf{P}, \mathbf{o}$$

where \mathbf{p} and \mathbf{q} govern pose and shape of the 2D model, $\bar{\mathbf{p}}$ influences the equivalent 3D shape, \mathbf{P} is a 2×3 projection matrix used to determine the projection of the 3D shape on the 2D image plane, and \mathbf{o} is the 2D translation vector accounting for the correct alignment of the 2D-back-projected 3D shape with the 2D shape.

As outlined in section 4.2, the goal is the minimization of the term

$$F(\mathbf{p}, \mathbf{q}, \bar{\mathbf{p}}, \mathbf{P}, \mathbf{o}) = K \cdot \left\| \underbrace{P(\bar{s}_0 + \sum_{i=1}^m \bar{p}_i \bar{b}_i)}_{=\mathbf{s}} + \underbrace{\begin{pmatrix} o_x & \cdots & o_x \\ o_y & \cdots & o_y \end{pmatrix}}_{=\mathbf{o}} - \underbrace{N(s_0 + \sum_{i=1}^m p_i b_i; \mathbf{q})}_{=\mathbf{s}} \right\|^2 \quad (\text{E.1})$$

The algorithm intervenes in the 2D fitting process at equation D.2 by replacing it. It therefore takes as input parameters the 2D Hessian matrix H and the 2D steepest descent parameter updates *before* normalization,

$$(\Delta \mathbf{p}_{2D}, \Delta \mathbf{q}_{2D}) = \sum_{x \in s_0} (SD_i(x) \cdot E(x)) \quad (\text{E.2})$$

and returns the *constrained* and normalized 2D parameter updates $(\Delta \mathbf{p}, \Delta \mathbf{q})$. One iteration of the algorithm has the following structure:

- Compute the Jacobians used for steepest descent on the above term F for both coordinates x and y , and each shape vertex i ,

$$SD_{F,i} = \left(\frac{\partial F_i}{\partial \mathbf{p}} \mathbf{J}_{\mathbf{p}}, \frac{\partial F_i}{\partial \mathbf{q}} \mathbf{J}_{\mathbf{q}}, \frac{\partial F_i}{\partial \bar{\mathbf{p}}}, \frac{\partial F_i}{\partial \mathbf{P}}, \frac{\partial F_i}{\partial \mathbf{o}} \right) \quad (\text{E.3})$$

- Create the updated Hessian H_{3D} using the 2D Hessian H as a submatrix as outlined in the following section E.2 and compute its inverse,
- Determine the updates to all 6 parameters,
- Update the internal parameters $\bar{\mathbf{p}} \leftarrow \bar{\mathbf{p}} + \Delta\bar{\mathbf{p}}$ and $\mathbf{o} \leftarrow \mathbf{o} + \Delta\mathbf{o}$,
- Update the internal projection matrix \mathbf{P} . This matrix has 6 elements, but we have to take into consideration the orthonormality requirements by using the update described below;
- Return the updated and constrained 2D parameters \mathbf{p} and \mathbf{q} .

E.2 Steepest Descent Geometry

The Jacobians $\frac{\partial F_i}{\partial \mathbf{p}} \mathbf{J}_{\mathbf{p}}$ and $\frac{\partial F_i}{\partial \mathbf{q}} \mathbf{J}_{\mathbf{q}}$ reflect the influence of a small change to the 2D warp parameters on the 3D minimization term F . [41] suggests estimating these terms: each parameter component is in turn set to a small value, while all others are set to 0. The warp update method described in section C.2 is then used to compute the change of the current destination shape after applying this small parameter update.

The Jacobian $\frac{\partial F_i}{\partial \mathbf{o}}$ has the simple form

$$\frac{\partial F_{i,j}}{\partial \mathbf{o}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad (\text{E.4})$$

for each 2D shape vertex i . From equation 4.3 we see that

$$\frac{\partial F}{\partial \bar{\mathbf{p}}_j} = \mathbf{P} \bar{b}_j \quad (\text{E.5})$$

where \bar{b}_j is the j th 3D shape base. We are left with the Jacobian of F with respect to the projection matrix, $\frac{\partial F_i}{\partial \mathbf{P}}$, where

$$P = \omega \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \quad (\text{E.6})$$

and the two rows of P are orthonormal.

To constrain the updates of P to the allowed projection matrix structure, we use the modified small-angle-optimization suggested in [34]. The update process is split into four contributions representing matrix scale ω and small angle updates θ_x , θ_y , θ_z along the three axes. When assuming small changes, we essentially have an infinitesimal transformation of the projection matrix \mathbf{P} of the form

$$\mathbf{P} \leftarrow \mathbf{P}(\mathbf{I} + \mathbf{e}) \quad (\text{E.7})$$

where \mathbf{I} is the identity matrix and \mathbf{e} an infinitesimal update. Since our transformation is supposed to be a rotation, we require $A = (\mathbf{I} + \mathbf{e})$ to be an orthogonal rotation matrix such that $A^T = A^{-1}$. Since

$$AA^{-1} = (\mathbf{I} + \mathbf{e})(\mathbf{I} - \mathbf{e}) = \mathbf{I}^2 - \mathbf{e}^2 \approx \mathbf{I} \quad (\text{E.8})$$

we have $A^{-1} = \mathbf{I} - \mathbf{e}$. Moreover,

$$A^T = (\mathbf{I} + \mathbf{e})^T = \mathbf{I}^T + \mathbf{e}^T = \mathbf{I} + \mathbf{e}^T \quad (\text{E.9})$$

and as a consequence $\mathbf{e} = -\mathbf{e}^T$, meaning that our infinitesimal update \mathbf{e} has to be an antisymmetric matrix

$$\mathbf{e} = \begin{pmatrix} 0 & -\theta_z & \theta_y \\ \theta_z & 0 & -\theta_x \\ -\theta_y & \theta_x & 0 \end{pmatrix} \quad (\text{E.10})$$

The above conclusions lead to the following update to the projection matrix

$$\mathbf{P} \leftarrow (\omega + \Delta\omega)\mathbf{P} \begin{pmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{pmatrix} \quad (\text{E.11})$$

where the two rows of P are reorthonormalized *before* multiplication with the updated scale $(\omega + \Delta\omega)$. The corresponding terms

$$\frac{\partial F_i}{\partial \omega}, \frac{\partial F_i}{\partial \theta_x}, \frac{\partial F_i}{\partial \theta_y}, \frac{\partial F_i}{\partial \theta_z} \quad (\text{E.12})$$

are computed using

$$\frac{\partial F_i}{\partial \omega} = \frac{\partial F_i}{\partial P} \frac{\partial P}{\partial \omega} = \begin{pmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \end{pmatrix} \bar{\mathbf{s}}_i \quad (\text{E.13})$$

where $\bar{\mathbf{s}}_i$ is the i th vertex of the current 3D shape $\bar{\mathbf{s}}$; this follows from equations E.6 and E.1. For each small angle update we use the derivative of the updated projection matrix in equation E.11 with respect to each small angle update and obtain

$$\frac{\partial F_i}{\partial \theta_x} = \mathbf{P} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{pmatrix} \bar{\mathbf{s}}_i \quad (\text{E.14})$$

$$\frac{\partial F_i}{\partial \theta_y} = \mathbf{P} \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{pmatrix} \bar{\mathbf{s}}_i, \quad (\text{E.15})$$

$$\frac{\partial F_i}{\partial \theta_z} = \mathbf{P} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \bar{\mathbf{s}}_i \quad (\text{E.16})$$

Now, we can compose the 3D Hessian using the 2D Hessian passed as a parameter,

$$H_{3D} = \begin{pmatrix} H_{2D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix} + K \cdot \sum_{x,y} \sum_{i=1}^N (SD_{F,i})^T SD_{F,i}. \quad (\text{E.17})$$

Finally we compute the updates to all 6 parameter vectors using

$$\begin{pmatrix} \Delta \mathbf{p} \\ \Delta \mathbf{q} \\ \Delta \bar{\mathbf{p}} \\ \Delta \mathbf{P} \\ \Delta \mathbf{o} \end{pmatrix} = -H_{3D}^{-1} \left(\begin{pmatrix} \Delta \mathbf{p}_{2D} \\ \Delta \mathbf{q}_{2D} \\ 0 \\ 0 \\ 0 \end{pmatrix} + K \cdot \sum_{x,y} \sum_{i=1}^N (SD_{F,i})^T F(\mathbf{p}, \mathbf{q}, \bar{\mathbf{p}}, \mathbf{P}, \mathbf{o}) \right) \quad (\text{E.18})$$

where $\Delta \mathbf{P}$ consists of the 4 scale and small angle parameters as described above.

F. KD-Trees

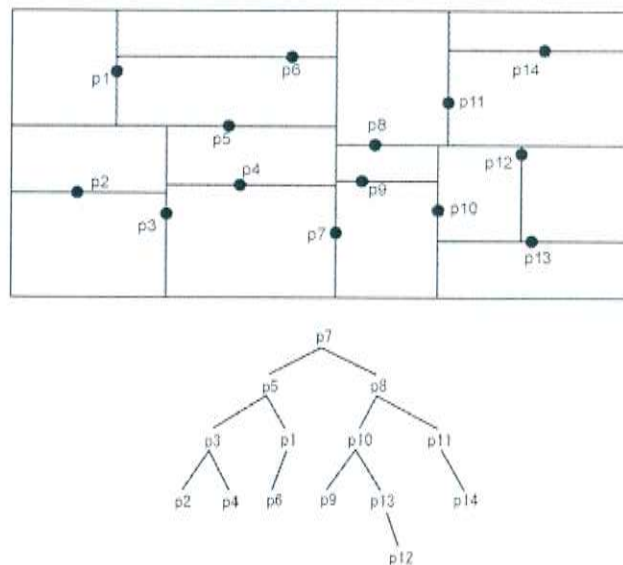


Figure F.1: 2D points and their 2D KD-Tree

Several methods aim at structuring n -dimensional data sets such that efficient lookup of nearest neighbors can be performed. Most of these methods use trees in order to exclude data not relevant for a given query as early as possible during the search.

For our stereo fitting, we have the following situation:

- **Given:** a dense 3D point cloud, typically containing around $P=5000$ elements, noisy, outliers;
- **Query input:** N 3D points (the shape vertices);
- **Output needed:** the nearest neighbor of each of the query points within the given point cloud (we could also use the mean of all neighbors within a given radius);

- **Query frequency:** a query of size N is performed per stereo fitting iteration:
 - $n_p + n_q$ **times** during computation of the Jacobian of the stereo surface (once for each small change to the n_p shape and n_q pose parameters),
 - **once** for the computation of the error function (distance between current 3D shape and current stereo point cloud).

The data structure in question will be populated once for each new input frame and does not need to offer any other functionality, neither adding nor deleting of elements; no re-balancing is required after generation of the data structure.

As a consequence, we decided to employ a KD-Tree. Figure F.1 on page 49 shows a 2D KD-Tree for the point set above. A KD-Tree is a space-partitioning data structure for organizing points in a k -dimensional space. A KD-Tree uses only splitting planes that are perpendicular to one of the coordinate system axes. In addition, every node of a KD-Tree, from the root to the leaves, stores a point. As a consequence, each splitting plane must go through one of the points in the KD-Tree. Building a KD-Tree from the P 3D points is done in $O(P \log P)$. The implementation used in our application has eliminated recursion to achieve faster processing speed.

A KD-Tree is created respecting the following two conditions:

- At each level of the tree, selection of a subtree is done as a function of *one* of the k dimensions, with each further level the dimension used for selection alternates.
- At each step, the point selected to create the separating plane is the median of the coordinates of all the points being fed into the KD-Tree.

Lookup is done in $O(\log P)$ in a balanced tree in the same way by cycling through the dimensions and selecting on each level the next subtree as a result of comparing the coordinates of the query and the node point on the current dimension. Range queries use an Euclidian distance metric.

In our implementation, KD-Tree creation and (approximate) nearest neighbour lookup rely on the *ANN* library [22].

G. Stereo Vision

G.1 Stereo Geometry

Stereo vision uses differences in viewpoint of two cameras looking at the same scene to derive depth information on the objects visible.



Figure G.1: Point Grey's Bumblebee stereo camera

In our system, we used a bumblebee stereo camera manufactured by Point Grey Research [27] as shown in figure G.1. The camera consists of two Sony ICX204 color CCD cameras at a maximum resolution of 1024x768 working at 8-10Hz. Camera baseline is 12cm, its lens focal length is 2mm resulting in a 100° horizontal field of view. The camera system does not require in-field calibration due to a pre-calibration for lens distortions and inter-camera misalignment. The manufacturer provides a comprehensive stereo processing library optimized for fast undistortion, image alignment, disparity computation and filtering which we have used in our implementation.

In the following, we will provide a rough outline of the basic stereo camera geometry.

The disparity computation itself is done by matching features in the images obtained from the left and the right camera. Typically, some edge detection is performed using a Canny or Laplacian operator and the two images are aligned to reduce overall distance between the detected image features. Lack of texture can cause the matching process to fail.

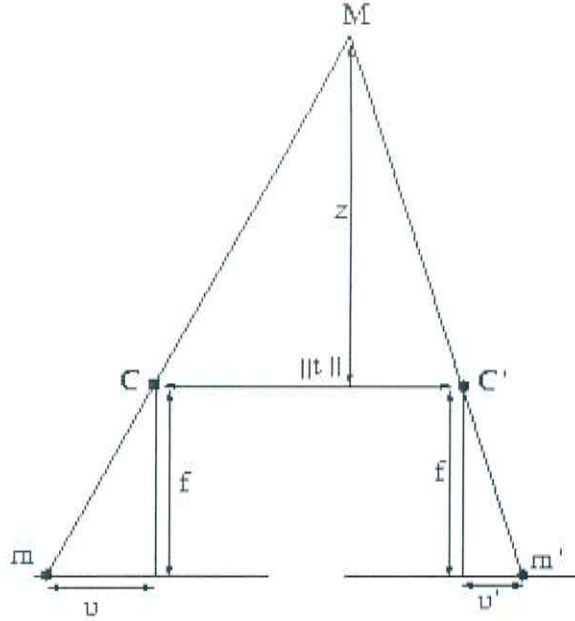


Figure G.2: Stereo vision geometry

Given the internal camera parameters focal length f (which is identical for both cameras) and baseline t (the distance between the two cameras C and C'), and supposing coplanar retinal planes, we wish to obtain the distance z of a pixel M to the camera plane. Figure G.2 illustrates the stereo camera geometry. The points $m = (u, v)$ and $m' = (u', v')$ are the (virtual) projection points of the pixel M on the retinal planes of the two cameras. We assume no vertical offset between the two cameras, as a consequence $v = v'$. The disparity associated with pixel M is

$$d = u' - u \quad (\text{G.1})$$

Using triangle similarities, we conclude that the distance z of the pixel M to the camera center can be computed by

$$z = \frac{f||t||}{d} \quad (\text{G.2})$$

After z is determined, the x and y coordinates of the pixel in the world reference system can be calculated using the projective camera equations

$$x = \frac{uz}{f}, \quad y = \frac{vz}{f} \quad (\text{G.3})$$

where u and v are the pixel locations in the 2D image projection plane.

Stereo computation accuracy is difficult to quantify, since the depth computation depends on the quality of the disparity value, and therefore on the accuracy of the image alignment method. Moreover, we cannot provide an absolute error value, since the error depends on the distance of an object to the camera. For accuracy evaluation, we therefore have to make the following two assumptions:

- In our application, we usually work with faces at an average distance of 1 meter to the camera, therefore we wish to evaluate depth accuracy at that distance.

- The manufacturer of the disparity computation library claims to achieve an average overall (matching and calibration) error of less than 0.3 pixels [27].

Assuming a given distance z and using the above mentioned internal camera parameters and equation (G.2), we obtain a disparity d for a distance z of

$$d = \frac{f\|t\|}{z} \quad (\text{G.4})$$

Supposing an overall error in depth computation of e pixels, we obtain an erroneous depth value z' of

$$z' = \frac{f\|t\|}{d + e} \quad (\text{G.5})$$

Consequently, the corresponding depth error can be computed from

$$\Delta z = \|z - z'\| = \left\| \frac{f\|t\|}{d} - \frac{f\|t\|}{d + e} \right\| \quad (\text{G.6})$$

Using typical numerical values for the internal camera parameters, $f = 218$ pixels and $\|t\| = 12\text{cm}$, and the disparity error of $e = 0.3$ pixels as provided by the manufacturer, we obtain the numerical value for the depth computation error of $\Delta z = 1.13\text{cm}$ in a distance of $z = 1\text{m}$ from the camera.

G.2 Disparity Filtering

By performing a sequence of filtering steps during disparity computation, we can exclude depth values irrelevant for our task. Furthermore, we can select a region of interest in the main 2D image based on the depth information obtained from the stereo camera. In a first step, we discard all disparity and its corresponding intensity image values based on a minimum and maximum disparity threshold. Additionally, we discard pixels not possessing sufficient texture or contrast for reliable disparity computation and we apply a morphological opening kernel to eliminate small surfaces.

References

- [1] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Proceedings of the IEEE CCVPR*, 2001.
- [2] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework, part 1 - 4. Technical report, Robotics Institute, Carnegie Mellon University, 2002–2004.
- [3] Simon Baker and Iain Matthews. Active appearance models revisited. Technical report, Robotics Institute, Carnegie Mellon University, 2004.
- [4] Peter N. Belhumeur, J.P. Hespanha, and David J. Kriegman. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19, 1997.
- [5] Jin-Xiang Chai, Jing Xiao, and Jessica Hodgins. Vision-based control of 3d facial animation. In *SIGGRAPH*. Robotics Institute, Carnegie Mellon University, 2003.
- [6] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, April 2005. <http://www.csie.ntu.edu.tw/~cjlin/libsvm/>.
- [7] Yang Chen and Gerard Medioni. Fitting a surface to 3d points using an inflating balloon model. In *Proc. IEEE CAD Based Vision Workshop*. University of Southern California, 1994.
- [8] Jeffrey F. Cohn and Paul Ekman. *Measuring Facial Action by Manual Coding, Facial EMG and Automatic Facial Image Analysis*. J.A. Harrigan, R. Rosenthal and K. Scherer, 2004. *Handbook of Nonverbal Behavior Research Methods in the Affective Sciences*.
- [9] J.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Active shape models: Their training and application. *Computer Vision and Image Understanding*, 61:38–59, January 1995.
- [10] T.F. Cootes and C.J. Taylor. Statistical models of appearance for computer vision. Technical report, Wolfson Image Analysis Unit, University of Manchester, October 2001.
- [11] I.L. Dryden and K.V. Mardia. Multivariate shape analysis. *Sankhya, The Indian Journal of Statistics*, 55:460–480, 1993.
- [12] P. Fua. Face models from uncalibrated video sequences. In *Proc. of CAPTECH*. Computer Graphics Lab, Lausanne, 1998.

- [13] P. Fua and C. Miccio. Animated heads from ordinary images. In *Proc. of CVIU*, Computer Graphics Lab, Lausanne, 1999.
- [14] J.C. Gower. Generalised procrustes analysis. *Psychometrika*, 40:33–50, 1975.
- [15] Ralph Gross, Simon Baker, and Iain Matthews. Constructing and fitting active appearance models with occlusion. In *IEEE Workshop on Face Processing in Video*, Robotics Institute, Carnegie Mellon University, 2004.
- [16] A. Hauptmann, J. Yang, and H. Wactlar. Automated analysis of nursing home observations. *IEEE Pervasive Computing*, 3:15–21, 2004.
- [17] Andreas Lanitis, Chris J. Taylor, and Timothy F. Cootes. Automatic interpretation and coding of face images using flexible models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:743–755, July 1997.
- [18] Richard Lengagne, Jean-Philippe Tarel, and Olivier Monga. From 2d images to 3d face geometry. In *Proc. of IEEE Intl. Conf. on Automatic Face Recognition*. INRIA Rocquencourt, 1996.
- [19] Chengjun Liu and Harry Wechsler. A gabor feature classifier for face recognition. In *Eighth International Conference on Computer Vision*. University of Missouri, 2001.
- [20] Philipp Michel and Rana El Kalioubry. Real time facial expression recognition in video using support vector machines. In *Fifth International Conference on Multimodal Interfaces*, 2003.
- [21] Roger Mohr and Bill Triggs. Projective geometry for image analysis. Technical report, GRAVIR, INRIA Grenoble, 1996. Tutorial given at ISPRS.
- [22] David M. Mount and Sunil Arya. *ANN: A Library for Approximate Nearest Neighbor Searching*, May 2005. <http://www.cs.umd.edu/mount/ANN/>.
- [23] Kai Nickel, Edgar Seemann, and Rainer Stiefelhagen. 3d-tracking of heads and hands for pointing gesture recognition in a human-robot interaction scenario. In *Sixth International Conference on Face and Gesture Recognition*. University of Karlsruhe, 2004.
- [24] I. S. Pandzic and R. Forchheimer. *MPEG-4 Facial Animation - the Standard, Implementation and Applications*. John Wiley and Sons, 2002.
- [25] Maja Pantic and Leon J.M. Rothkrantz. Automatic analysis of facial expressions: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1424–1445, December 2000.
- [26] Maja Pantic and Leon J.M. Rothkrantz. Towards an affect-sensitive multimodal human-computer interaction. *Proceedings of the IEEE*, 91(9):1370–1390, 2003.
- [27] Point Grey Research. *Point Grey Bumblebee Stereo Camera: Technical Documentation*, 2003. <http://ptgrey.com/products/bumblebee/bumblebee.PDF>.

- [28] Sami Romdhani, Volker Blanz, and Thomas Vetter. Face identification by fitting a 3d morphable model using linear shape and texture error functions. In *Proc. of the 6th European Conference on Computer Vision*. University of Freiburg, 2002.
- [29] Sami Romdhani and Thomas Vetter. Efficient, robust and accurate fitting of a 3d morphable model. In *Proc. of the 9th IEEE CCVPR*. University of Basel, 2003.
- [30] Edgar Seemann, Kai Nickel, and Rainer Stiefelhagen. Head pose estimation using stereo vision for human-robot interaction. In *Sixth International Conference on Face and Gesture Recognition*. University of Karlsruhe, 2004.
- [31] Mikkel B. Stegmann and David Delgado Gomez. A brief introduction to statistical shape analysis. Technical report, Informatics and Mathematical Modelling Department, Technical University of Denmark, March 2002.
- [32] R. Stiefelhagen, J. Yang, and A. Waibel. Towards unrestricted lipreading. *International Journal of Pattern Recognition and Artificial Intelligence*, 14:571–785, 2000.
- [33] Rainer Stiefelhagen, Hartwig Steusloff, and Alex Waibel. Chil - computers in the human interaction loop. In *Fifth International Workshop on Image Analysis for Multimedia Interactive Services*, April 2004.
- [34] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. In *SIGGRAPH*. Microsoft Research, 1997.
- [35] Demetri Terzopoulos and Manuela Vasilescu. Sampling and reconstruction with adaptive meshes. In *Proc. of IEEE Conf. Computer Vision and Pattern Recognition*. University of Toronto, 1991.
- [36] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams: A factorization method. *Proc. Natl. Acad. Scie. USA*, 90:9795–9802, November 1993.
- [37] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995.
- [38] Keith Waters. A muscle model for animating three-dimensional facial expression. *ACM Computer Graphics*, 21:17–24, July 1987.
- [39] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, second edition, 2005.
- [40] Jing Xiao. *Reconstruction, Registration and Modeling of Deformable Object Shapes*. PhD thesis, May 2005.
- [41] Jing Xiao, Simon Baker, Iain Matthews, and Takeo Kanade. Real-time combined 2d+3d active appearance models. In *Proceedings of the IEEE CCVPR*, 2004.
- [42] Jing Xiao, Jin-Xiang Chai, and Takeo Kanade. A closed-form solution to non-rigid shape and motion recovery. In *Proc. of the 8th European Conference on Computer Vision*. Robotics Institute, Carnegie Mellon University, May 2004.

-
- [43] Jing Xiao and Takeo Kanade. Non-rigid shape and motion recovery: Degenerate deformations. In *Proc. of the 8th European Conference on Computer Vision*. Robotics Institute, Carnegie Mellon University, May 2004.
 - [44] Chenghua Xu, Yunhong Wang, Tieniu Tan, and Long Quan. Face recognition based on a 3d mesh model. In *Proc. of Geometric Modeling and Processing*. Hong Kong University of Science and Technology, 2004.