Chapter 18
Facial Action Tracking

Jörgen Ahlberg and Igor S. Pandzic

18.1 Introduction

The problem of facial action tracking has been a subject of intensive research in the last decade. Mostly, this has been with such applications in mind as face animation, facial expression analysis, and human-computer interfaces. In order to create a face animation from video, that is, to capture the facial action (facial motion, facial expression) in a video stream, and then animate a face model (depicting the same or another face), a number of steps have to be performed. Naturally, the face has to be detected, and then some kind of model has to be fitted to the face. This can be done by aligning and deforming a 2D or 3D model to the image, or by localizing a number of facial landmarks. Commonly, these two are combined. The result must in either case be expressed as a set of parameters that the face model in the receiving end (the face model to be animated) can interpret.

Depending on the application, the model and its parameterization can be simple (e.g., just an oval shape) or complex (e.g., thousands of polygons in layers simulating bone and layers of skin and muscles). We usually wish to control appearance, structure, and motion of the model with a small number of parameters, chosen so as to best represent the variability likely to occur in the application. We discriminate here between rigid face/head tracking and tracking of facial action. The former is typically employed to robustly track the faces under large pose variations, using a rigid face/head model (that can be quite non-face specific, e.g., a cylinder). The latter here refers to tracking of facial action and facial features, such as lip and eyebrow

J. Ahlberg (✉)
Division of Information Systems, Swedish Defence Research Agency (FOI), P.O. Box 1165, 583 34 Linköping, Sweden
e-mail: jorahl@foi.se

I.S. Pandzic
Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia
e-mail: igor.pandzic@fer.hr

S.Z. Li, A.K. Jain (eds.), *Handbook of Face Recognition*,
DOI 10.1007/978-0-85729-932-1_18,

motion. The treatment in this chapter is limited to tracking of facial action (which, by necessity, includes the head tracking).

The parameterization can be dependent or independent on the model. Examples of model independent parameterizations are MPEG-4 Face Animation Parameters (see Sect. 18.2.3) and FACS Action Units (see Sect. 18.2.2). These parameterizations can be implemented on virtually any face model, but leaves freedom to the model and its implementation regarding the final result of the animation. If the transmitting side wants to have full control of the result, more dense parameterizations (controlling the motion of every vertex in the receiving face model) are used. Such parameterizations can be implemented by MPEG-4 Facial Animation Tables (FAT), morph target coefficients, or simply by transmitting all vertex coordinates at each time step.

Then, the face and its landmark/features/deformations must be tracked through an image (video) sequence. Tracking of faces has received significant attention for quite some time but is still not a completely solved problem.

18.1.1 Previous Work

A plethora of face trackers are available in the literatures, and only a few of them can be mentioned here. They differ in how they model the face, how they track changes from one frame to the next, if and how changes in illumination and structure are handled, if they are susceptible to drift, and if real-time performance is possible. The presentation here is limited to monocular systems (in contrast to stereo-vision) and 3D tracking.

18.1.1.1 Rigid Face/Head Tracking

Malciu and Prêteux [36] used an ellipsoidal (alternatively an ad hoc Fourier synthesized) textured wireframe model and minimized the registration error and/or used the optical flow to estimate the 3D pose. LaCascia et al. [31] used a cylindrical face model with a parameterized texture being a linear combination of texture warping templates and orthogonal illumination templates. The 3D head pose was derived by registering the texture map captured from the new frame with the model texture. Stable tracking was achieved via regularized, weighted least-squares minimization of the registration error.

Basu et al. [7] used the Structure from Motion algorithm by Azerbayejani and Pentland [6] for 3D head tracking, refined and extended by Jebara and Pentland [26] and Ström [56] (see below). Later rigid head trackers include notably the works by Xiao et al. [64] and Morency et al. [40].

18.1.1.2 Facial Action Tracking

In the 1990s, there were many approaches to non-rigid face tracking. Li et al. [33] estimated face motion in a simple 3D model by a combination of prediction and a model-based least-squares solution to the optical flow constraint equation. A render-feedback loop was used to combat drift. Eisert and Girod [16] determined a set of animation parameters based on the MPEG-4 Facial Animation standard (see Sect. 18.2.3) from two successive video frames using a hierarchical optical flow based method. Tao et al. [58] derived the 3D head pose from 2D-to-3D feature correspondences. The tracking of nonrigid facial features such as the eyebrows and the mouth was achieved via a probabilistic network approach. Pighin et al. [50] derived the face position (rigid motion) and facial expression (nonrigid motion) using a continuous optimization technique. The face model was based on a set of 3D face models.

In this century, DeCarlo and Metaxas [11] used a sophisticated face model parameterized in a set of deformations. Rigid and nonrigid motion was tracked by integrating optical flow constraints and edge-based forces, thereby preventing drift. Wiles et al. [63] tracked a set of hyperpatches (i.e., representations of surface patches invariant to motion and changing lighting).

Gokturk et al. [22] developed a two-stage approach for 3D tracking of pose and deformations. The first stage learns the possible deformations of 3D faces by tracking stereo data. The second stage simultaneously tracks the pose and deformation of the face in the monocular image sequence using an optical flow formulation associated with the tracked features. A simple face model using 19 feature points was utilized.

As mentioned, Ström [56] used an Extended Kalman Filter (EKF) and Structure from Motion to follow the 3D rigid motion of the head. Ingemars and Ahlberg extended the tracker to include facial action [24]. Ingemars and Ahlberg combined two sparse texture models, based on the first frame and (dynamically) on the previous frame respectively, in order to get accurate tracking and no drift. Lefèvre and Odobez used a similar idea, but separated the texture models more, and used Nelder–Mead optimization [42] instead of an EKF (see Sect. 18.4).

As follow-ups to the introduction of Active Appearance Models, there were several appearance-based tracking approaches. Ahlberg and Forchheimer [4, 5] represented the face using a deformable wireframe model with a statistical texture. A simplified Active Appearance Model was used to minimize the registration error. Because the model allows deformation, rigid and non-rigid motions are tracked. Dornaika and Ahlberg [12, 14] extended the tracker with a step based on random sampling and consensus to improve the rigid 3D pose estimation. Fanelli and Fratarcangeli [18] followed the same basic strategy, but exploited the Inverse Compositional Algorithm by Matthews and Baker [37]. Zhou et al. [65] and Dornaika and Davoine [15] combined appearance models with a particle filter for improved 3D pose estimation.

18.1.2 Outline

This chapter explains the basics of parametric face models used for face and facial action tracking as well as fundamental strategies and methodologies for tracking. A few tracking algorithms serving as pedagogical examples are described in more detail. The chapter is organized as follows: In Sect. 18.2 parametric face modeling is described. Various strategies for tracking are discussed in Sect. 18.3, and a few tracker examples are described in Sects. 18.4–18.6. In Sect. 18.6.3 some examples of commercially available tracking systems are give.

18.2 Parametric Face Modeling

There are many ways to parameterize and model the appearance and behavior of the human face. The choice depends on, among other things, the application, the available resources, and the display device. Statistical models for analyzing and synthesizing facial images provide a way to model the 2D appearance of a face. Here, other modeling techniques for different purposes are mentioned as well.

What all models have in common is that a compact representation (few parameters) describing a wide variety of facial images is desirable. The parameter sets can vary considerably depending on the variability being modeled. The many kinds of variability being modeled/parameterized include the following.

- *Three-dimensional motion and pose*—the dynamic, 3D position and rotation of the head. Nonrigid face/head tracking involves estimating these parameters for each frame in the video sequence.
- *Facial action*—facial feature motion such as lip and eyebrow motion. Estimated by nonrigid tracking.
- *Shape and feature configuration*—the shape of the head, face and the facial features (e.g., mouth, eyes). This could be estimated by some alignment or facial landmark localization methods.
- *Illumination*—the variability in appearance due to different lighting conditions.
- *Texture and color*—the image pattern describing the skin.
- *Expression*—muscular synthesis of emotions making the face look, for example, happy or sad.

For a head tracker, the purpose is typically to extract the 3D motion parameters and be invariant to all other parameters. Whereas, for example, a user interface being sensitive to the mood of the user would need a model extracting the expression parameters, and a recognition system should typically be invariant to all but the shape and texture parameters.

18.2.1 Eigenfaces

Statistical texture models in the form of *eigenfaces* [30, 53, 60] have been popular for facial image analysis. The basic idea is that a training set of facial images are collected and registered, each image is reshaped into a vector, and a principal component analysis (PCA) is performed on the training set. The principal components are called eigenfaces. A facial image (in vector form), $\mathbf{x}$, can then be approximated by a linear combination, $\hat{\mathbf{x}}$, of these eigenfaces, that is,

$$\mathbf{x} \approx \hat{\mathbf{x}} = \bar{\mathbf{x}} + \boldsymbol{\Phi}_x \xi, \tag{18.1}$$

where $\bar{\mathbf{x}}$ is the average of the training set, $\boldsymbol{\Phi}_x = (\boldsymbol{\phi}_1 | \boldsymbol{\phi}_2 | \ldots | \boldsymbol{\phi}_t)$ contains the eigenfaces, and ξ is a vector of weights or eigenface parameters. The parameters minimizing $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$ are given by

$$\xi = \boldsymbol{\Phi}_x^{\mathrm{T}} (\mathbf{x} - \bar{\mathbf{x}}). \tag{18.2}$$

Commonly, some kind of image normalization is performed prior to eigenface computation.

The space spanned by the eigenfaces is called the *face subspace*. Unfortunately, the manifold of facial images has a highly nonlinear structure and is thus not well modeled by a linear subspace. Linear and nonlinear techniques are available in the literature and often used for face recognition. For face tracking, it has been more popular to linearize the face manifold by warping the facial images to a standard pose and/or shape, thereby creating *shape-free* [10], *geometrically normalized* [55], or *shape-normalized* images and eigenfaces (texture templates, texture modes) that can be warped to any face shape or texture-mapped onto a wireframe face model.

18.2.2 Facial Action Coding System

During the 1960s and 1970s, a system for parameterizing minimal facial actions was developed by psychologists trying to analyze facial expressions. The system was called the *Facial Action Coding System* (FACS) [17] and describes each facial expression as a combination of around 50 *Action Units* (AUs). Each AU represents the activation of one facial muscle.

The FACS has been a popular tool not only for psychology studies but also for computerized facial modeling (an example is given in Sect. 18.2.5). There are also other models available in the literatures, for example, Park and Waters [49] described modeling skin and muscles in detail, which falls outside the scope of this chapter.

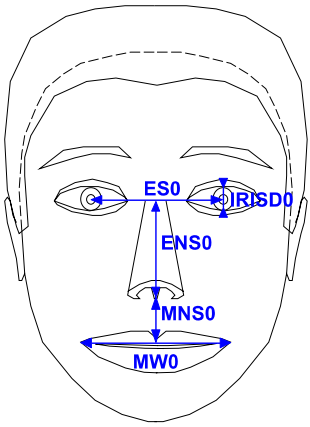

Fig. 18.1 Face Animation Parameter Units (FAPU)

18.2.3 MPEG-4 Facial Animation

MPEG-4, since 1999 an international standard for coding and representation of audiovisual objects, contains definitions of face model parameters [41, 47].

There are two sets of parameters: *Facial Definition Parameters* (FDPs), which describe the static appearance of the head, and *Facial Animation Parameters* (FAPs), which describe the dynamics.

MPEG-4 defines 66 low-level FAPs and two high-level FAPs. The low-level FAPs are based on the study of minimal facial actions and are closely related to muscle actions. They represent a complete set of basic facial actions, and therefore allow the representation of most natural facial expressions. Exaggerated values permit the definition of actions that are normally not possible for humans, but could be desirable for cartoon-like characters.

All low-level FAPs are expressed in terms of the *Face Animation Parameter Units* (FAPUs), illustrated in Fig. 18.1. These units are defined in order to allow interpretation of the FAPs on any face model in a consistent way, producing reasonable results in terms of expression and speech pronunciation. They correspond to distances between key facial features and are defined in terms of distances between the MPEG-4 facial Feature Points (FPs, see Fig. 18.2). For each FAP it is defined on which FP it acts, in which direction it moves, and which FAPU is used as the unit for its movement. For example, FAP no. 3, `open_jaw`, moves the Feature Point 2.1 (bottom of the chin) downwards and is expressed in MNS (mouth-nose separation) units. The MNS unit is defined as the distance between the nose and the mouth (see Fig. 18.1) divided by 1024. Therefore, in this example, a value of 512 for the FAP no. 3 means that the bottom of the chin moves down by half of the mouth-nose separation. The division by 1024 is introduced in order to have the units sufficiently small that FAPs can be represented in integer numbers.

The two high-level FAPs are `expression` and `viseme`. `expression` can contain two out of a predefined list of six basic expressions: joy, sadness, anger, fear, disgust and surprise. Intensity values allow to blend the two expressions. Similarly,

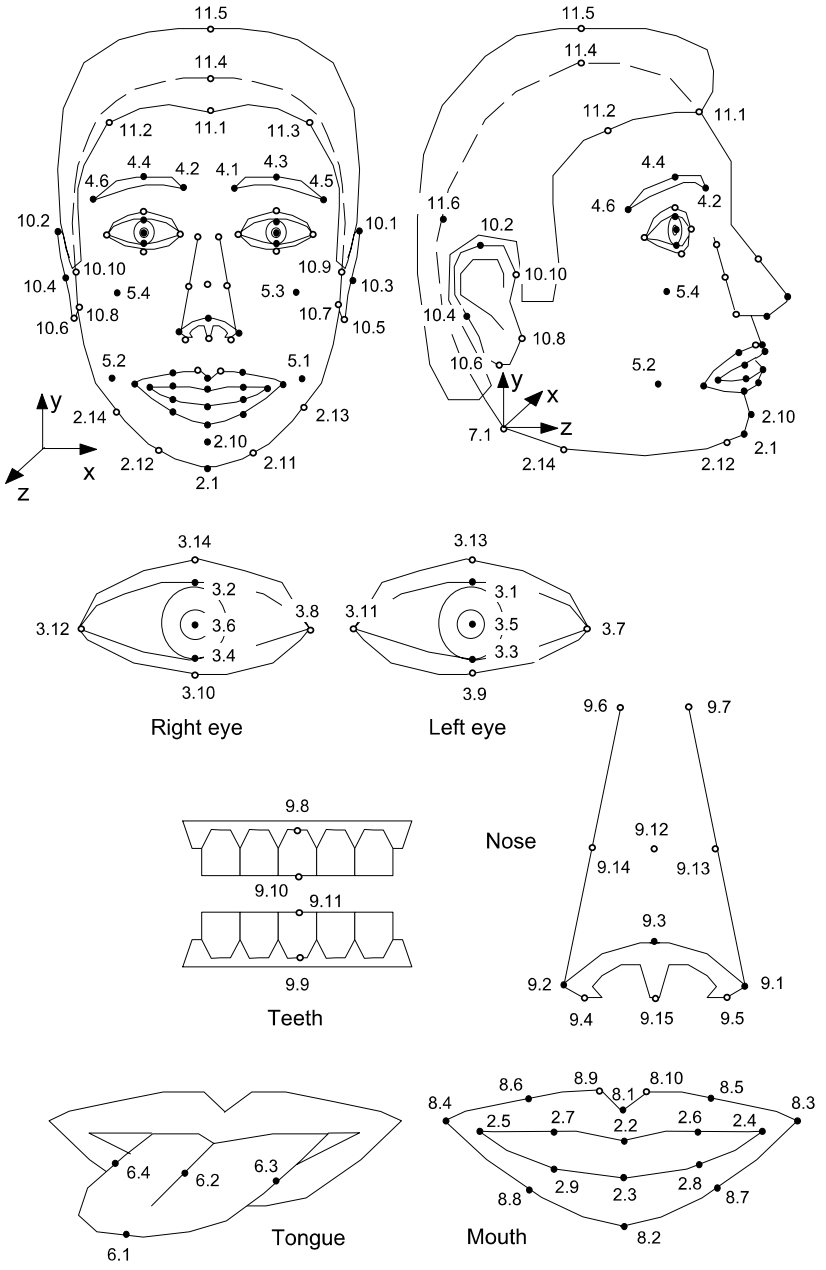

Fig. 18.2 Facial Feature Points (FP)

`viseme` can contain two out of a predefined list of 14 visemes, and a blending factor to blend between them.

The neutral position of the face (when all FAPs are 0) is defined as follows:

- The coordinate system is right-handed; head axes are parallel to the world axes.
- Gaze is in the direction of Z axis.
- All face muscles are relaxed.
- Eyelids are tangent to the iris.
- The pupil is one third of IRISD0.
- Lips are in contact—the line of the lips is horizontal and at the same height of lip corners.
- The mouth is closed and the upper teeth touch the lower ones.
- The tongue is flat, horizontal with the tip of tongue touching the boundary between upper and lower teeth (feature point 6.1 touching 9.11, see Fig. 18.2).

All FAPs are expressed as displacements from the positions defined in the neutral face.

The FDPs describe the static shape of the face by the 3D coordinates of each feature point and the texture as an image with the corresponding texture coordinates.

18.2.4 Computer Graphics Models

When synthesizing faces using computer graphics (for user interfaces [25], web applications [45], computer games, or special effects in movies), the most common model is a *wireframe model* or a *polygonal mesh*. The face is then described as a set of vertices connected with lines forming polygons (usually triangles). The polygons are shaded or texture-mapped, and illumination is added. The texture could be parameterized or fixed—in the latter case facial appearance is changed by moving the vertices only. To achieve life-like animation of the face, a large number (thousands) of vertices and polygons are commonly used. Each vertex can move in three dimensions, so the model requires a large number of degrees of freedom. To reduce this number, some kind of parameterization is needed.

A commonly adopted solution is to create a set of *morph targets* and blend between them. A morph target is a predefined set of vertex positions, where each morph target represents, for example, a facial expression or a viseme. Thus, the model shape is defined by the morph targets $\mathbf{A}$ and controlled by the parameter vector α

$$\begin{aligned} \mathbf{g} &= \mathbf{A}\alpha, \\ \textstyle\sum \alpha_i &= 1. \end{aligned} \tag{18.3}$$

The $3N$-dimensional vector $\mathbf{g}$ contains the 3D coordinates of the N vertices; the columns of the $3N \times M$-matrix $\mathbf{A}$ contain the M morph targets; and α contains the M morph target coefficients. To limit the required computational complexity, most α_i values are usually zero.

Morph targets are usually manually created by artists. This is a time consuming process so automatic methods have been devised to copy a set of morph targets from one face model to another [20, 43, 46].

To render the model on the screen, we need to find a correspondence from the model coordinates to image pixels. The projection model (see Sect. 18.2.6), which is not defined by the face model, defines a function $P(\cdot)$ that projects the vertices on the image plane

$$(u, v) = P(x, y, z). \tag{18.4}$$

The image coordinate system is typically defined over the range $[-1, 1] \times [-1, 1]$ or $[0, w-1] \times [0, h-1]$, where (w, h) is the image dimensions in pixels.

To texturize the model, each vertex is associated with a (prestored) texture coordinate (s, t) defined on the unit square. Using some interpolating scheme (e.g., piecewise affine transformations), we can find a correspondence from any point (x, y, z) on the model surface to a point (s, t) in the texture image and a point (u, v) in the rendered image. Texture mapping is executed by copying (interpolated) pixel values from the texture $\mathbf{I}_t(s, t)$ to the rendered image of the model $\mathbf{I}_m(u, v)$. We call the coordinate transform $T_{\mathbf{u}}(\cdot)$, and thus

$$\mathbf{I}_m(u, v) = T_{\mathbf{u}}\big[\mathbf{I}_t(s, t)\big]. \tag{18.5}$$

While morphing is probably the most commonly used facial animation technique, it is not the only one. Skinning, or bone-based animation, is the process of applying one or more transformation matrices, called bones, to the vertices of a polygon mesh in order to obtain a smooth deformation, for example, when animating joints such as elbow or shoulder. Each bone has a weight that determines its influence on the vertex, and the final position of the vertex is the weighted sum of the results of all applied transformations. While the main purpose of skinning is typically body animation, it has been applied very successfully to facial animation. Unlike body animation, where the configuration of bones resembles the anatomical human skeleton, for facial animation the bones rig is completely artificial and bears almost no resemblance to human skull. Good starting references for skinning are [29, 39]. There are numerous other approaches to facial animation, ranging from ones based on direct modeling of observed motion [48], to pseudo muscle models [35] and various degrees of physical simulation of bone, muscle and tissue dynamics [27, 59, 61].

More on computerized facial animation can be found in [47, 49]. Texture mapping is treated in [23].

18.2.5 Candide: A Simple Wireframe Face Model

Candide is a simple face model that has been a popular research tool for many years. It was originally created by Rydfalk [52] and later extended by Welsh [62] to cover the entire head (Candide-2) and by Ahlberg [2, 3] to correspond better to

MPEG-4 facial animation (Candide-3). The simplicity of the model makes it a good pedagogic example.

Candide is a wireframe model with 113 vertices connected by lines forming 184 triangular surfaces. The geometry (shape, structure) is determined by the 3D coordinates of the vertices in a model-centered coordinate system (x, y, z). To modify the geometry, Candide-1 and Candide-2 implement a set of Action Units from FACS. Each AU is implemented as a list of vertex displacements, an *Action Unit Vector* (AUV), describing the change in face geometry when the Action Unit is fully activated. The geometry is thus parameterized as

$$\mathbf{g}(\alpha) = \bar{\mathbf{g}} + \boldsymbol{\Phi}_a \alpha, \quad 0 \leq \alpha_i \leq 1 \tag{18.6}$$

where the resulting vector $\mathbf{g}$ contains the (x, y, z) coordinates of the vertices of the model, $\bar{\mathbf{g}}$ is the standard shape of the model, and the columns of $\boldsymbol{\Phi}_a$ are the AUVs. The α_i values are the Action Unit activation levels.

Candide-3 is parameterized slightly different than the previous versions, generalizing the AUVs to animation modes (implementing AUs or FAPs) and adding shape modes. The parameterization is

$$\mathbf{g}(\alpha, \sigma) = \bar{\mathbf{g}} + \boldsymbol{\Phi}_a \alpha + \boldsymbol{\Phi}_s \sigma. \tag{18.7}$$

The difference between α and σ is that the shape parameters control the static deformations that cause individuals to differ from each other. The animation parameters control the dynamic deformations due to facial action.

This kind of linear model is, in different variations, a common way to model facial geometry. For example, PCA found a matrix that described 2D shape and animation modes combined, Gokturk et al. [22] estimated 3D animation modes using a stereo-vision system, and Caunce et al. [9] created a shape model from models adapted to profile and frontal photos.

To change the pose, the model coordinates are rotated, scaled and translated so that

$$\mathbf{g}(\alpha, \sigma, \pi) = s\mathbf{R}\mathbf{g}(\alpha, \sigma) + \mathbf{t} \tag{18.8}$$

where π contains the six pose/global motion parameters plus a scaling factor.

18.2.6 Projection Models

The function $(u, v) = P(x, y, z)$, above, is a general projection model representing the camera. There are various projection models from which to chose, each with a set of parameters that may be known (calibrated camera) or unknown (uncalibrated camera). In most applications, the camera is assumed to be at least partly calibrated. We stress that only simple cases are treated here, neglecting such camera parameters as skewness and rotation. For more details, consult a computer vision textbook like [54].

- The simplest projection model is the *orthographic projection*—basically just throwing away the z-coordinate. Parameters are pixel size (a_u, a_v) and principal point (c_u, c_v). The projection can be written

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_u & 0 & 0 & c_u \\ 0 & a_v & 0 & c_v \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{18.9}$$

- The most common camera model is the *perspective projection*, which can be expressed as

$$\begin{cases} (u', v', w') = \mathbf{P}(x, y, z, 1)^{\mathrm{T}}, \\ (u, v) = (u'/w', v'/w') \end{cases} \tag{18.10}$$

where

$$\mathbf{P} = \begin{pmatrix} a_u & 0 & 0 & c_x \\ 0 & a_v & 0 & c_y \\ 0 & 0 & 1/f & c_z \end{pmatrix}, \tag{18.11}$$

(c_x, c_y, c_z) is the focus of expansion (FOE), and f is the focal length of the camera; (a_u, a_v) determines the pixel size. Commonly, a simple expression for $\mathbf{P}$ is obtained where $(c_x, c_y, c_z) = \mathbf{0}$ and $a_u = a_v = 1$ are used. In this case, (18.10) is simply

$$(u, v) = \left(f\frac{x}{z}, f\frac{y}{z} \right). \tag{18.12}$$

- The *weak perspective projection* is an approximation of the perspective projection suitable for an object where the internal depth variation is small compared to the distance z_{ref} from the camera to the object.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} a_u/z_{\mathrm{ref}} & 0 & 0 & c_x \\ 0 & a_v/z_{\mathrm{ref}} & 0 & c_y \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}. \tag{18.13}$$

18.3 Tracking Strategies

A face tracking system estimates the rigid or non-rigid motion of a face through a sequence of image frames. In the following, we discuss the two-frame situation, where we have an estimation of the model parameters $\hat{\mathbf{p}}_{k-1}$ in the old frame, and the system should estimate the parameters $\hat{\mathbf{p}}_k$ in the new frame (i.e., how to transform the model from the old frame to the new frame).

As mentioned in the introduction, we discuss only monocular tracking here, that is, we disregard stereo vision systems.

18.3.1 Motion-Based vs. Model-Based Tracking

Tracking systems can be said to be either *motion-based* or *model-based*, also referred to as *feed-forward* or *feed-back* motion estimation. A *motion-based* tracker estimates the displacements of pixels (or blocks of pixels) from one frame to another. The displacements might be estimated using optical flow methods (giving a dense optical flow field), block-based motion estimation methods (giving a sparse field but using less computational power), or motion estimation in a few image patches only (giving a few motion vectors only but at very low computational cost).

The estimated motion field is used to compute the motion of the object model using, for example, least-squares, Kalman filtering, or some optimization method. The motion estimation in such a method is consequently dependent on the pixels in two frames; the object model is used only for transforming the 2D motion vectors to 3D object model motion. The problem with such methods is the *drifting* or the *long sequence motion problem*. A tracker of this kind accumulates motion errors and eventually loses track of the face. Examples of such trackers can be found in the literature [7, 8, 51].

A *model-based* tracker, on the other hand, uses a model of the object's appearance and tries to change the object model's pose (and possibly shape) parameters to fit the new frame. The motion estimation is thus dependent on the object model and the new frame—the old frame is not regarded except for constraining the search space. Such a tracker does not suffer from drifting; instead, problems arise when the model is not strong or flexible enough to cope with the situation in the new frame. Trackers of this kind can be found in certain articles [5, 31, 33, 56]. Other trackers [11, 13, 22, 36] are motion-based but add various model-based constraints to improve performance and combat drift.

18.3.2 Model-Based Tracking: First Frame Models vs. Pre-trained Models

In general, the word *model* refers to any prior knowledge about the 3D structure, the 3D motion/dynamics, and the 2D facial appearance. One of the main issues when designing a model-based tracker is the appearance model. An obvious approach is to capture a reference image of the object from the first frame of the sequence. The image could then be geometrically transformed according to the estimated motion parameters, so one can compensate for changes in scale and rotation (and possibly nonrigid motion). Because the image is captured, the appearance model is deterministic, object-specific, and (potentially) accurate. Thus, trackers of this kind can be precise, and systems working in real time have been demonstrated [22, 32, 56, 63].

A drawback with such a first frame model is the lack of flexibility—it is difficult to generalize from one sample only. This can cause problems with changing appearance due to variations in illumination, facial expression, and other factors. Another

drawback is that the initialization is critical; if the captured image was for some reason not representative for the sequence (due to partial occlusion, facial expression, or illumination) or simply not the correct image (i.e., if the object model was not correctly aligned in the first frame) the tracker does not work well. Such problems can usually be solved by manual interaction but may be hard to automate.

Note that the model could be renewed continuously (sometimes called an *online* model), so the model always is based on the previous frame. In this way the problems with flexibility are reduced, but the tracker is then motion-based and might drift.

Another property is that the tracker does not need to know what it is tracking. This could be an advantage—the tracker can track different kinds of objects—or a disadvantage. A relevant example is when the goal is to extract some higher level information from a human face, such as facial expression or lip motion. In that case we need a tracker that identifies and tracks specific facial features (e.g., lip contours or feature points).

A different approach is a *pre-trained model-based tracker*. Here, the appearance model relies on previously captured images combined with knowledge of which parts or positions of the images correspond to the various facial features. When the model is transformed to fit the new frame, we thus obtain information about the estimated positions of those specific facial features.

The appearance model may be person specific or general. A specific model could, for example, be trained on a database containing images of one person only, resulting in an accurate model for this person. It could cope, to some degree, with the illumination and expression changes present in the database. A more general appearance model could be trained on a database containing many faces in different illuminations and with different facial expressions. Such a model would have a better chance to enable successful tracking of a previously unseen face in a new environment, whereas a specific appearance model presumably would result in better performance on the person and environment for which it was trained. Trackers using pre-trained models of appearance can be found in the literature [5, 31].

18.3.3 Appearance-Based vs. Feature-Based Tracking

An *appearance-based* or *featureless* or *generative model-based* tracker matches a model of the entire facial appearance with the input image, trying to exploit all available information in the model as well as the image. Generally, we can express this as follows:

Assume a parametric generative face model and an input image $\mathbf{I}$ of a face from which we want to estimate a set of parameters. The parameters to be extracted should form a subset of parameter set controlling the model. Given a vector $\mathbf{p}$ with N parameters, the face model can generate an image $\mathbf{I}_m(\mathbf{p})$. The principle of analysis-by-synthesis then states that the best estimates of the facial parameters are the ones

minimizing the distance between the generated image and the input image

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \delta\left[\mathbf{I}, \mathbf{I}_m(\mathbf{p})\right] \tag{18.14}$$

for some distance measure $\delta(\cdot)$.

The problem of finding the optimal parameters is a high-dimensional (N dimensions) search problem and thus of high computational complexity. By using clever search heuristics, we can reduce the search time. The trackers described in [5, 31, 33, 36] are appearance-based.

A *feature-based* tracker, on the other hand, chooses a few facial features that are, supposedly, easily and robustly tracked. Features such as color, specific points or patches, and edges can be used. Typically, a tracker based on feature points tries to estimate the 2D position of a set of points and from these points to compute the 3D pose of the face. Feature-based trackers can be found in [11, 12, 26, 56, 63].

In the following sections, we describe three trackers found in the literature. They represent the classes mentioned above.

18.4 Feature-Based Tracking Example

The tracker described in this section tracks a set of feature points in an image sequence and uses the 2D measurements to calculate the 3D structure and motion of the face and the facial features. The tracker is based on the structure from motion (SfM) algorithm by Azerbayejani and Pentland [6]. The (rigid) face tracker was then developed by Jebara and Pentland [26] and further by Ström et al. [56, 57]. The tracker was later extended as to handle non-rigid motion, and thus track facial action by Ingemars and Ahlberg [24].

With the terminology above, it is the first frame model-based and feature-based tracker. We stress that the presentation here is somewhat simplified.

18.4.1 Face Model Parameterization

The head tracker designed by Jebara and Pentland [26] estimated a model as a set of points with no surface. Ström et al. [57] extended the system to include a 3D wireframe face model (Candide). A set of feature points are placed on the surface of the model, not necessarily coinciding with the model vertices. The 3D face model enables the system to predict the surface angle relative to the camera as well as self-occlusion. Thus, the tracker can predict when some measurements should not be trusted.

18.4.1.1 Pose Parameterization

The pose in the kth frame is parameterized with three rotation angles (r_x, r_y, r_z), three translation parameters (t_x, t_y, t_z), and the inverse focal length $\phi = 1/f$ of the camera.[1]

Azerbayejani and Pentland [6] chose to use a perspective projection model where the origin of the 3D coordinate system is placed in the center of the image plane instead of at the focal point, that is, the FOE is set to (0, 0, 1) (see Sect. 18.2.6). This projection model has several advantages; for example, there is only one unknown parameter per feature point (as becomes apparent below).

Thus, the 2D (projected) screen coordinates are computed as

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{1 + z\phi} \begin{pmatrix} x \\ y \end{pmatrix}. \tag{18.15}$$

18.4.1.2 Structure Parameterization

The structure of the face is represented by the image coordinates (u_0, v_0) and the depth values z_0 of the feature points in the first frame. If the depths z_0 are known, the 3D coordinates of the feature points can be computed for the first frame as

$$\begin{pmatrix} x_0 \\ y_0 \end{pmatrix} = (1 + z_0\phi) \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} \tag{18.16}$$

and for all following frames as

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \mathbf{R} \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} \tag{18.17}$$

where $\mathbf{R}$ is the rotation matrix created. For clarity of presentation, the frame indices on all the parameters are omitted.

All put together, the model parameter vector is

$$\mathbf{p} = (t_x, t_y, t_z, r_x, r_y, r_z, \phi, z_1, \ldots, z_N)^{\mathrm{T}} \tag{18.18}$$

where N is the number of feature points and r_x, r_y, r_z are used to update $\mathbf{R}$. Combining (18.16), (18.17), and (18.15), we get a function from the parameter vector to screen coordinates

$$(u_1, v_1, \ldots, u_N, v_N)^{\mathrm{T}} = h_k(\mathbf{p}). \tag{18.19}$$

Note that the parameter vector has $N + 7$ degrees of freedom, and that we get $2N$ values if we measure the image coordinates for each feature point. Thus, the problem of estimating the parameter vector from image coordinates is overconstrained when $N > 7$.

[1] In practice, the z-translation should be parameterized by $\zeta = t_z\phi$ instead of t_z for stability reasons.

18.4.2 Parameter Estimation Using an Extended Kalman Filters

A Kalman filter is used to estimate the dynamic changes of a state vector of which a function can be observed. When the function is nonlinear, we must use an extended Kalman filter (EKF). The literature on Kalman filtering is plentiful [21, 28, 54, 55], so we only summarize the approach here.

In our case, the state vector is the model parameter vector $\mathbf{p}$ and we observe, for each frame, the screen coordinates $\mathbf{u}_k = h_k(\mathbf{p}_k)$. Because we cannot measure the screen coordinates exactly, measurement noise $\mathbf{v}_k$ is added as well. We can summarize the dynamics of the system as

$$\begin{cases} \mathbf{p}_{k+1} = f_k(\mathbf{p}_k) + \mathbf{w}_k, & \mathbf{w}_k \sim N(\mathbf{0}, \mathbf{W}_k), \\ \hat{\mathbf{u}}_k = h_k(\mathbf{p}_k) + \mathbf{v}_k, & \mathbf{v}_k \sim N(\mathbf{0}, \mathbf{V}_k) \end{cases} \tag{18.20}$$

where $f_k(\cdot)$ is the dynamics function, $h_k(\cdot)$ is the measurement function, and $\mathbf{w}$ and $\mathbf{v}$ are zero-mean Gaussian random variables with known covariances $\mathbf{W}_k$ and $\mathbf{V}_k$.

The job for the EKF is to estimate the state vector $\mathbf{p}_k$ given the measurement vector $\hat{\mathbf{u}}_k$ and the previous estimate $\hat{\mathbf{p}}_{k-1}$. Choosing the trivial dynamics function $f_k(\mathbf{p}_k) = \mathbf{p}_k$, the state estimate is updated using

$$\hat{\mathbf{p}}_k = \hat{\mathbf{p}}_{k-1} + \mathbf{K}_k\big[\hat{\mathbf{u}}_k - h_k(\hat{\mathbf{p}}_{k-1})\big] \tag{18.21}$$

where $\mathbf{K}_k$ is the Kalman gain matrix. It is updated every time step depending on $f_k(\cdot)$, $h_k(\cdot)$, $\mathbf{W}_k$, and $\mathbf{V}_k$. The covariances are given by initial assumptions or estimated during the tracking.

18.4.3 Tracking Process

The tracker must be initialized, manually or by using a face detection algorithm. The model reference texture is captured from the first frame, and feature points are automatically extracted. To select feature points that could be reliably tracked, points where the determinant of the Hessian

$$\det(H) = \begin{vmatrix} \mathbf{I}_{xx}(x, y) & \mathbf{I}_{xy}(x, y) \\ \mathbf{I}_{xy}(x, y) & \mathbf{I}_{yy}(x, y) \end{vmatrix} \tag{18.22}$$

is large are used. The function `cvGoodFeaturesToTrack` in OpenCV [44] implements a few variations. The determinant is weighted with the cosine of the angle between the model surface normal and the camera direction. The number of feature points to select is limited only by the available computational power and the real-time requirements.

The initial feature point depth values $(z_1, \ldots, z_N)$ are given by the 3D face model. Then, for each frame, the model is rendered using the current model parameters. Around each feature point, a small patch is extracted from the rendered image.

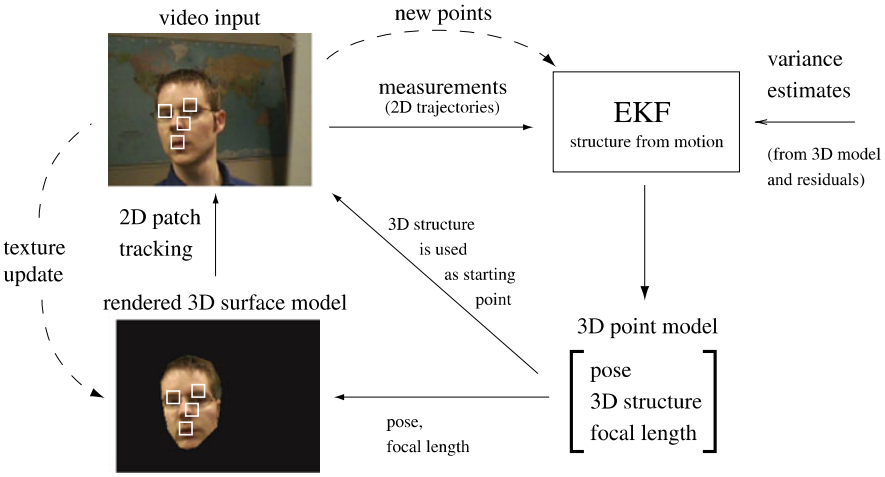

Fig. 18.3 Patches from the rendered image (*lower left*) are matched with the incoming video. The two-dimensional feature point positions are fed to the EKF, which estimates the pose information needed to render the next model view. For clarity, only 4 of 24 patches are shown. Illustration courtesy of J. Ström

The patches are matched with the new frame using a zero-mean normalized cross correlation. The best match, with subpixel precision, for each feature point is collected in the measurement vector $\hat{\mathbf{u}}_k$ and fed into the EKF update equation (18.21).

Using the face model and the values from the normalized template matching, the measurement noise covariance matrix can be estimated making the EKF rely on some measurements more than others. Note that this also tells the EKF in which directions in the image the measurements are reliable. For example, a feature point on an edge (e.g., the mouth outline) can reliably be placed in the direction perpendicular to the edge but less reliably along the edge. The system is illustrated in Fig. 18.3.

18.4.4 Tracking of Facial Action

Ingemars and Ahlberg [24] extended the tracker to track facial action as well. A set of states was added to the state vector corresponding to the animation parameters of Candide-3. In order to be able to track these, there must be feature points within the area that is influenced by each estimated animation parameter, and the number of feature points must be higher. However, the automatic feature point selection can still be used, since the observation function can be calculated online by interpolation of the known observation functions of the face model vertices.

In order to be both accurate and to handle drift, the tracker combines feature point patches from the first frame of the sequence with patches dynamically extracted from the previous frame. Examples are shown in Fig. 18.4.

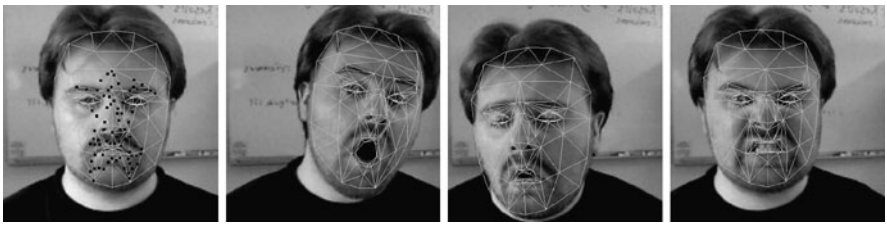

Fig. 18.4 Tracking example, feature-based tracking. The *leftmost image* shows the automatically extracted tracking points. Images courtesy of N. Ingemars

18.5 Appearance-Based Tracking Example

In this section, we describe a statistical model-based and appearance-based tracker estimating the 3D pose and deformations of the face. It is based on the Active Appearance Models (AAMs) described in Chap. 5. The tracker was first presented by Ahlberg [1], and was later improved in various ways as described in Sect. 18.5.4.

To use the AAM search algorithm, we must first decide on a geometry for the face model, its parameterization of shape and texture, and how we use the model for warping images. Here, we use the face model Candide-3 described in Sect. 18.2.5.

18.5.1 Face Model Parameterization

18.5.1.1 Geometry Parameterization

The geometry (structure) $\mathbf{g}(\sigma, \alpha)$ of the Candide model is parameterized according to (18.7). There are several techniques to estimate the shape parameters σ (e.g., an AAM search or facial landmark localization method). When adapting a model to a video sequence, the shape parameters should be changed only in the first frame(s)—the head shape does not vary during a conversation—whereas the pose and animation parameters naturally change at each frame. Thus, during the tracking process we can assume that σ is fixed (and known), and let the shape depend on α only

$$\begin{cases} \bar{\mathbf{g}}_\sigma = \bar{\mathbf{g}} + \boldsymbol{\Phi}_s \sigma, \\ \mathbf{g}_\sigma(\alpha) = \bar{\mathbf{g}}_\sigma + \boldsymbol{\Phi}_a \alpha. \end{cases} \tag{18.23}$$

18.5.1.2 Pose Parameterization

To perform global motion (i.e., pose change), we need six parameters plus a scaling factor according to (18.8). Since the Candide model is defined only up to scale, we can adopt the weak perspective projection and combine scale and z-translation in one parameter. Thus, using the 3D translation vector $\mathbf{t} = (t_x, t_y, t_z)^{\mathrm{T}}$ and the three Euler angles for the rotation matrix $\mathbf{R}$, our pose parameter vector is

$$\pi = (r_x, r_y, r_z, , t_x, t_y, t_z)^{\mathrm{T}}. \tag{18.24}$$

18.5.1.3 Texture Parameterization

We use a statistical model of the texture and control the texture with a small set of texture parameters ξ. The model texture vector $\hat{\mathbf{x}}$ is generated according to (18.1) (see Sect. 18.2.1). The synthesized texture vector $\hat{\mathbf{x}}$ has for each element a corresponding (s,t) coordinate and is equivalent to the texture image $\mathbf{I}_t(s,t)$: the relation is just lexicographic ordering, $\hat{\mathbf{x}} = L[\mathbf{I}_t(s,t)]$. $\mathbf{I}_t(s,t)$ is mapped on the wireframe model to create the generated image $\mathbf{I}_m(u,v)$ according to (18.5).

The entire appearance of the model can now be controlled using the parameters $(\xi, \pi, \alpha, \sigma)$. However, as we assume that the shape σ is fixed during the tracking session, and the texture ξ depends on the other parameters, the parameter vector we optimize below is

$$\mathbf{p} = \left(\pi^{\mathrm{T}}, \alpha^{\mathrm{T}}\right)^{\mathrm{T}}. \tag{18.25}$$

18.5.2 Tracking Process

The tracker should find the optimal adaptation of the model to each frame in a sequence as described in Sect. 18.3.3. That is, we wish to find the parameter vector $\mathbf{p}_k^*$ that minimizes the distance between image $\mathbf{I}_m$ generated by the model and each frame $\mathbf{I}_k$. Here, an iterative solution is presented, and as an initial value of $\mathbf{p}$ we use $\hat{\mathbf{p}}_{k-1}$ (i.e., the estimated parameter vector from the previous frame).

Instead of directly comparing the generated image $\mathbf{I}_m(u,v)$ to the input image $\mathbf{I}_k(u,v)$, we back-project the input image to the model's parametric surface coordinate system (s,t) using the inverse of the texture mapping transform $T_{\mathbf{u}}$

$$\mathbf{I}_{\mathbf{u}(\mathbf{p})}(s,t) = T^{-1}_{\mathbf{u}(\mathbf{p})}\left[\mathbf{I}_k(u,v)\right], \tag{18.26}$$

$$\mathbf{x}(\mathbf{p}) = L\left[\mathbf{I}_{\mathbf{u}(\mathbf{p})}(s,t)\right]. \tag{18.27}$$

We then compare the normalized input image vector $\mathbf{x}(\mathbf{p})$ to the generated model texture vector $\hat{\mathbf{x}}(\mathbf{p})$. $\hat{\mathbf{x}}(\mathbf{p})$ is generated in the face subspace as closely as possible to $\mathbf{x}(\mathbf{p})$ (see (18.1)), and we compute a residual image $\mathbf{r}(\mathbf{p}) = \mathbf{x}(\mathbf{p}) - \hat{\mathbf{x}}(\mathbf{p})$. The process from input image to residual, is illustrated in Fig. 18.5.

As the distance measures according to (18.14), we use the squared norm of the residual image

$$\delta\left(\mathbf{I}_k, \mathbf{I}_m(\mathbf{p})\right) = \left\|\mathbf{r}(\mathbf{p})\right\|^2. \tag{18.28}$$

From the residual image, we also compute the update vector

$$\Delta\mathbf{p} = -\mathbf{U}\mathbf{r}(\mathbf{p}) \tag{18.29}$$

where $\mathbf{U} = (\frac{\delta\mathbf{r}}{\delta\mathbf{p}})^{\dagger}$ is the precomputed active appearance update matrix (i.e., the pseudo-inverse of the estimated gradient matrix $\frac{\delta\mathbf{r}}{\delta\mathbf{p}}$). It is created by numeric differentiation, systematically displacing each parameter and computing an average over the training set.

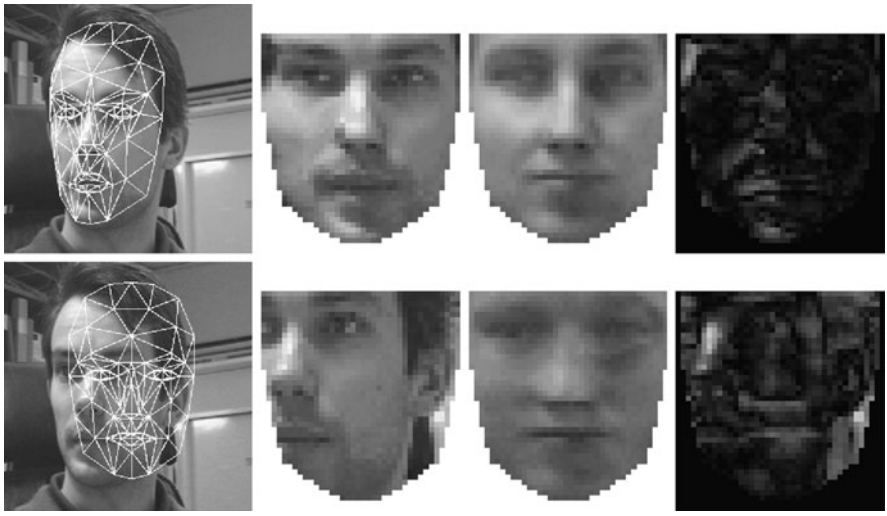

Fig. 18.5 Analysis-synthesis process. A good and a bad model adaptation. The more similar the normalized image and its approximation is, the better the model adaptation is

We then compute the estimated new parameter vector as

$$\hat{\mathbf{p}}_k = \mathbf{p} + \Delta\mathbf{p}. \tag{18.30}$$

In most cases, the model fitting is improved if the entire process is iterated a few times.

18.5.3 Tracking Example

To illustrate, a video sequence of a previously unseen person was used and the Candide-3 model was manually adapted to the first frame of the sequence by changing the pose parameters and the static shape parameters (recall that the shape parameter vector σ is assumed to be known). The model parameter vector $\mathbf{p}$ was then iteratively optimized for each frame, as is shown in Fig. 18.6.

Note that this, quite simple, tracker needs some more development in order to be robust to varying illumination, strong facial expressions, and large head motion. Moreover, it is very much depending on the training data.

Fig. 18.6 Tracking example, appearance-based tracking. Every tenth frame shown

18.5.4 Improvements

In 2002, Dornaika and Ahlberg [12, 13] introduced a feature/appearance-based tracker. The head pose is estimated using a RANdom SAmpling Consensus (RANSAC) technique [19] combined with a texture consistency measure to avoid drifting. Once the 3D head pose π is estimated, the facial animation parameters α can be estimated using the scheme described in Sect. 18.5.

In 2004, Matthews and Baker [37] proposed the Inverse Compositional Algorithm which allows for an accurate and fast fitting, actually reversing the roles of the input image and the template in the well-known, but slow, Lucas–Kanade Image Alignment algorithm [34]. Supposing that the appearance will not change much among different frames, it can be "projected out" from the search space. Fanelli and Fratarvangeli [18] incorporated the Inverse Compositional Algorithm in an AAM-based tracker to improve robustness.

Zhou et al. [65] and Dornaika and Davoine [15] combined appearance models with a particle filter for improved 3D pose estimation.

18.6 Fused Trackers

18.6.1 Combining Motion- and Model-Based Tracking

In order to exploit the advantages of the various tracking strategies mentioned above, modern state-of-the-art trackers often combine them. For example, Lefèvre and Odobez [32] used the Candide model and two sets of feature points. The first set is called the "trained set", that is, a set of feature points trained from the first frame of the video sequence. The second set is the "adaptive set", that is continuously adapted during tracking. Using only one of these sets (the first corresponding to first frame model-based and the second to motion-based, according to the terminology used here) results in a certain kinds of tracking failures. While tracking using the adaptive method (motion-based tracking) is more precise, it is also prone to drift. Lefèvre and Odobez devised a hybrid method exploiting the advantages of both the adaptive

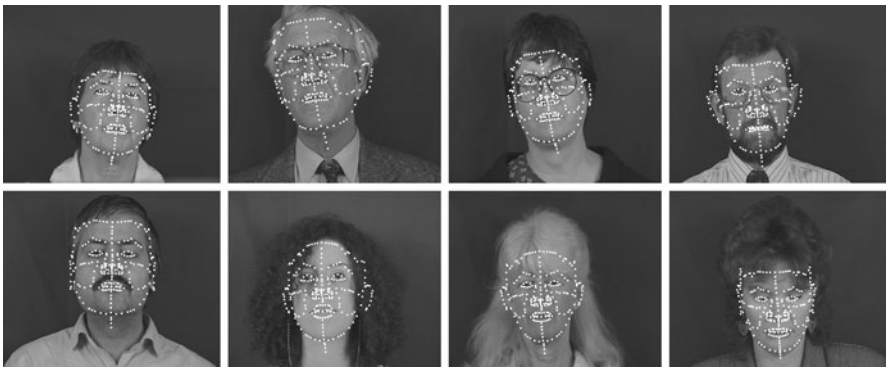

Fig. 18.7 Face model alignment examples. Images courtesy of A. Caunce

and the trained method. Compared to the method by Ingemars and Ahlberg [24] described above, Lefèvre and Odobez made a more thorough investigation of how to exploit the two methods. Another major difference is the choice of methods for 3D pose estimation from 2D measurements (EKF and Nelder–Mead downhill simplex optimization [42], respectively). Notably, Lefèvre and Odobez demonstrated stable tracking under varying lighting conditions.

18.6.2 Combining Appearance- and Feature-Based Tracking

Another recent development was published by Caunce et al. [9], combining the feature-based and appearance-based approaches. In the tradition of the Manchester group (where the AAMs originated), the work was started with thorough statistical model-building of shape and texture of the face. Using profile and frontal photographies, a 3D shape model was adapted to each training subject and a PCA performed. A set of texture patches were extracted from the mean texture, and used to adapt the 3D model to a new image in a two stage process. The first stage consists of the rigid body adaptation (pose, translation) and the second step the facial shape (restrained by the pretrained model). Moreover, a scale hierarchy was used.

Thus, this method is primarily aimed at facial landmark localization/face model alignment, but is used (by Caunce et al.) for tracking of facial action as well by letting the parameter estimated from the previous frame serving as the initial estimation. Example results are shown in Fig. 18.7 using XM2VTSDB [38] imagery.

18.6.3 Commercially Available Trackers

Naturally, there are a number of commercial products providing head, face and facial action tracking. Below, some of these are mentioned.

One of the best commercial facial trackers was developed in the 1990s by a US company called Eyematic. The company went bankrupt and the technology was taken over by another company which subsequently sold it to Google.

The Australian company Seeing Machines is offering a face tracker API called faceAPI that tracks head motion, lips and eyebrows.

Visage Technologies (Sweden) provides a statistical appearance-based tracker as well as an feature-based tracker in an API called visage|SDK.

The Japanese company Omron developed a suite of face sensing technology called OKAO Vision that includes face detection, tracking of the face, lips, eyebrows and eyes, as well as estimation of attributes such as gender, ethnicity and age.

Image Metrics (UK) provides high-end performance-based facial animation service to film and game producers based on their in-house facial tracking algorithms.

Mova (US) uses a radically different idea in order to precisely capture high-density facial surface data using phosphorescent makeup and fluorescent lights.

Other companies with face tracking products include Genemation (UK), OKI (Japan), SeeStorm (Russia) and Visual Recognition (Netherlands). Face and facial feature tracking is also used in end-user products such as Fix8 by Mobinex (US) and the software bundled with certain Logitech webcams.

18.7 Conclusions

We have described some strategies for tracking and distinguished between model- and motion-based tracking as well as between appearance- and feature-based tracking. Whereas motion-based trackers may suffer from drifting, model-based trackers do not have that problem. Appearance- and feature-based trackers follow different basic principles and have different characteristics.

Two trackers have been described, one feature-based and one appearance-based. Both trackers are all model-based and thus do not suffer from drifting. Improvements found in the literature are discussed for both trackers.

Trackers combining the described tracking strategies, presumably representing the state-of-the-art, have been described, and commercially available tackers have been briefly mentioned.

References

1. Ahlberg, J.: An active model for facial feature tracking. In: Proceedings of the International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS), Tampere, Finland, pp. 63–67 (2001)
2. Ahlberg, J.: Candide 3—an updated parameterized face. Technical Report LiTH-ISY-R-2326, Linköping University, Sweden (2001)
3. Ahlberg, J.: Model-based coding—extraction, coding and evaluation of face model parameters. PhD thesis, Linköping University, Sweden (2002)