INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Real Time 3D Head Pose Recovery for Model Based Video Coding

by

Marius Daniel Cordea

Submitted to the Department of Electrical Engineering
in August, 1999, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering

Acknowledgements

I would like to take this opportunity to thank all the people who have been helpful to the completion of this thesis. First of all, I thank Dr. Emil Petriu my thesis supervisor, for his great guidance and support during the course of this thesis. I would like to especially thank my parents, and my brother for the support and love they have given me my whole life.

Abstract

Real Time 3D Head Pose Recovery for Model Based Video Coding

This thesis presents an Interactive Facial Animation Testbed (IFAT) for a low bit-rate videophone system based on a face (head) model. The effects of head motion and facial expressions are combined in the 3D head model. The head "*pose*" *(position, orientation, and scale estimation)* has to be accurately recovered before attempting to recover the face expressions. For the specification of the facial expressions we adopted the "muscle-based" facial model parameterization. This topology independent system incorporates 16 muscles and 10 parameters controlling mouth opening, jaw rotation, eye movement, eyelid opening, and head orientation. Two algorithms were developed for automatic head pose recovery: *2½D pose recovery* of the 3D position and 2D orientation using a 2D elliptical head model, and *3D pose recovery* of the 3D position and 3D orientation using a 3D wireframe head model. The developed global motion tracking system is meant to work in a realistic videoconferencing environment with no makeup highlighting the speaker's facial features, uncalibrated camera, unknown lighting conditions and unknown scene background. In order to validate the accuracy of the 3D head tracking system, we developed a rapid calibration technique using a sequence of images of a synthetic "standard" 3D head in lieu of real life head.

Table of Contents

List of Figures

List of Terms

3D.....Three-Dimensional

2D.....Two-Dimensional

2½D.....Two and Half-Dimensional

pose.....Position Orientation and Scale Estimation

MBVC.....Model Based Video Coding

IFAT.....Interactive Facial Animation Testbed

AU.....Action Unit

FACS.....Facial Action Coding System

HSV.....Hue-Saturation-Value

RGB.....Red-Green-Blue

ROI.....Region-of-Interest

PDF..... Probability Density Function

EM.....Expectation-Maximization

MOG.....Mixture of Gaussian

LKF.....Linear Kalman Filter

EKF.....Extended Kalman Filter

IEKF.....Iterated Extended Kalman Filter

SFM.....Structure-from-Motion

RMS.....Root-Mean-Square

COP.....Center of Projection

Chapter 1

Introduction

New applications in communications, multimedia, and digital television require highly efficient, robust and flexible digital video compression and coding techniques, enabling efficient interchange and distribution of visual information. Multimedia applications range from desktop videoconferencing to interactive entertainment networks providing video-on-demand, video games, and teleshopping. The integration of motion video in multimedia environments is technologically one of the most demanding tasks, due to the high data rates and real-time constraints. The bit-rate required to encode a full motion VHS- quality video signal has decreased from 20 Mbit/s around 1980 to well below 1 Mbit/s today [Gir96]. For head-and-shoulders scenes, typical for videoconferencing applications, rates can be substantially lower. In recent years, several video coding standards such as H.261, H.263, MPEG-1, and MPEG-2 have been introduced. They address mainly the compression of generic video data for digital storage and communication services. These video coders utilize the statistics of the video signal without knowledge of the semantic content of the frames and can therefore be used for arbitrary scenes [Eis99]. Higher coding efficiency can be expected from model-based video coders when the semantic information of the scene can be exploited. However, this type of coders is restricted to scenes that can be composed of objects that are known by the decoders.

1.1. Motivation

In today's multimedia systems a key issue is the efficient storage, transmission and manipulation of image sequences and 3D scenes. The upcoming MPEG-4 Video and Synthetic/Natural Hybrid Coding (SNHC) standard will cover the emerging content-based image coding and image access as well as animation of 3D models. The limited rate budget available makes the motion estimation for very-low-bit-rate coding more important. Envisioned data rates between 8 and 64 Kbps enable the transmission over mobile and fixed telephone channels as well as over the

Internet. By using this new technique, bit rates as low as 14.4 or even 9.6 Kbps will be efficient for future videoconferencing. Amongst the advantages of the very-low-bit rate coding is the use of Public Switched Telephone Networks (PSTN) or mobile channels as transmission media. The general public has access to this form of transmission media.

Model-based video coding (MBVC), also known as *knowledge-based video coding* [Aiz89], [Kis89], has recently emerged as a very low bit rate video compression method suitable for videoconferencing and videophone applications. The MBVC increases coding efficiency by using knowledge about the scene content and describing the real world geometry by 3D model objects (e.g. 3D face models). Additionally, object-specific functionalities like the animation of facial expressions should be provided. In a videoconferencing scenario, the principle of this compression is to generate a parametric model of the image seen at the emission end and to transmit only the characteristic parameters describing how the model changes in time. These change parameters are then used to animate the model of the image recovered at the reception end. In Figure 1.1 is presented the structural diagram of the one-way channel of the model-based videophone system [Cor98]. Each image is analyzed and transformed in a 3D-wireframe model. Using *a priori* knowledge about the images to be transmitted, a polygonal mesh geometric model of a generic human face is fitted/molded on the live image of the particular person (Figure 1.2) as seen by the video camera at the emission end. In order to convey a natural-look feeling the 3D modeling should be able to recover the color and texture of the underlying skin in each elementary polygon of the wireframe. The emitting videophone sends the complete set of data on the model geometry, color and texture to the receiving end only once. After that, it needs only to transmit the description of changes in the image model, which can result in a data transmission rate as low as 1 Kbps [Gir94], without reducing the resolution of the original image.

The real time recovery of the 3D pose of the face/head for model based low-bandwidth video coding is a central element in the emerging videoconferencing and videophone systems [Aiz89], [Aiz95], [Sch96], [Kis89], [Ohz96].

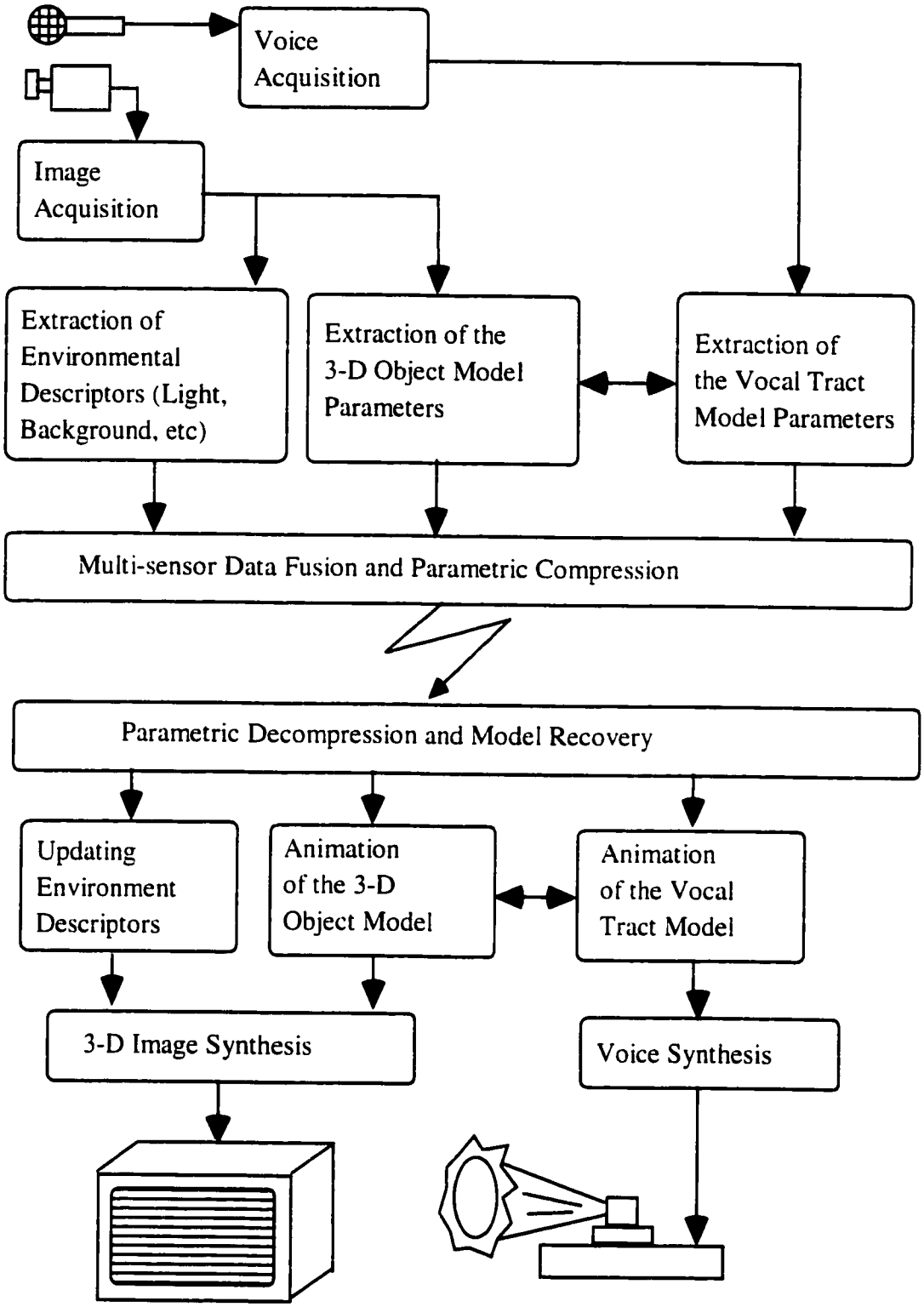


Fig. 1.1: Structural diagram of a one way videoconferencing system [Cor98].

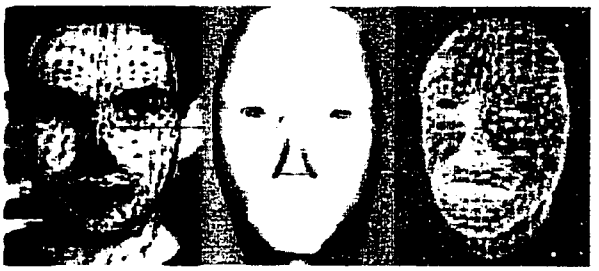


Fig. 1.2: The 3-D generic polygonal mesh face model (on the right) is molded by Marius' face image (on the left) to generate the 3-D polygonal model (in the center) to be used for parametric transmission and animation.

1.2. The Model-Based Video Coding Architecture

Traditionally the MBVC research has concentrated into three areas:

(i) *Head Modeling*
(ii) *Face Detection*
(iii) *Motion Estimation*

Computer generated 3D models have extensively been used for the synthesis of naturally looking images and video. These models should have a limited number of vertices but enough details in order to distinguish facial features such as the lips or eyebrows. Rendering of animated video sequences using 3D models, usually needs 3 separate sets of data [Har98]:

(i) data describing the basic shape of the 3D object, e.g. a wireframe or a spline surface;
(ii) data describing the surface brightness and color (the texture);
(iii) data describing the temporal variation of the model (the animation parameters);

The first two data sets describing the shape and texture are usually considered together. A generic human face is modeled as a 3-D polygonal mesh. Computer graphics or acquired image texture is mapped onto the mesh to obtain a natural looking appearance.

For the specification of the facial expressions, we adopted the "muscle-based" facial model parameterization, which is not topology specific and can be controlled by a small number of parameters [Wat87]. This model incorporates three types of muscles: linear, sphincter, and sheet. Anatomical characteristics of the facial muscles and tissues are implemented as geometric distortion functions that describe the displacement of a predefined set of control points on the facial surface.

The first important step in a full automatic MBVC system is the identification and location of the face in first image frames. The developed method uses a stochastic facial color model and the elliptical structure of the human head. After separating the head from the background clutter, an ellipse is fitted to mark the boundary between the head region and the background. The next motion estimation steps encompass global 3D-motion recovery, local motion estimation, expression and emotion analysis, and video-audio synchronization etc. The considerable effort that is requested is nevertheless justified since the recovery of the correct motion parameters is the essential to the achievement of the ultra low bit rates promised by MBVC. Image coding algorithms like the ITU-R Rec. H.261 and H.263 as well as MPEG-4 Video describe changes in an image sequence by 2D motion parallel to the image plane.

The goal of this thesis is to develop an Interactive Facial animation Testbed (IFAT) for a low bit-rate MBVC videoconferencing system. The problem is technologically difficult, as 3D motion parameters have to be extracted from a 2D-video sequence. This can be done by tracking facial features or by using the optical flow in successive frames showing different points of view of the same 3D scene. In both cases the knowledge of the 3D scene and imaging process could be used to increase the accuracy and efficiency of the implementation. In the past several years, work has concentrated mainly on to recovering both 3D-head pose (rigid motion) [Aza93], [Lih93], [Jeb96], [Bas96], [Fuk93] and facial expressions (non-rigid motion) [Lih94], [Bla95], [Ess95], [Nur98] from a sequence of images of performer's face. The effects of head motion and facial expressions are combined in these images, so it is crucial to successfully separate the rigid from the non-rigid motion of the head (problem known as "pose/expression separation"). The head pose has to be accurately computed before attempting to recover the expressions.

This thesis discusses two algorithms for automatic head pose recovery using model-based approaches. The head pose is recovered by using a tracking module along with an estimator module. These two tracking algorithms implementing the MBVC motion estimation are:

(i) *2½D tracking*, when four head motion parameters are recovered, namely 3D position and orientation in 2D image plane. This method uses a 2D elliptical head model, two matching algorithms: a region- and an edge-based, and a Linear Kalman Filter (LKF) estimator.

(ii) *3D tracking*, when all six rigid motion parameters are recovered, namely 3D translations and 3D rotations. This method uses a 3D wireframe head model, a correlation-based matching algorithm, and an Extended Kalman Filter (EKF) estimator.

In both cases the input to the system is a 2D-video sequence of the performer's head-and-shoulders. The output is the trajectories of tracked facial features, as well as an estimate of the *2½D* or *3D* motion of the head. The recovered motion parameters are rendered in the *Head Modeling* section (Figure 1.3). A major consideration in MBVC is the estimation of camera parameters such as the focal length from image sequences. The conventional approach assumes that the camera model and parameters are assigned *a priori* and the camera system is fixed. Nonetheless, in a real system, the camera parameters are usually unknown and the camera may change its room, pan, and rotation. Another element recovered by our *3D tracking* method, besides the 3D relative motion object-camera, is the perspective camera focal length. Our global motion tracking system is meant to work in a realistic videoconferencing environment, without makeup highlighting the speaker's facial features, and with an uncalibrated camera, unknown lighting conditions, and unknown scene background.

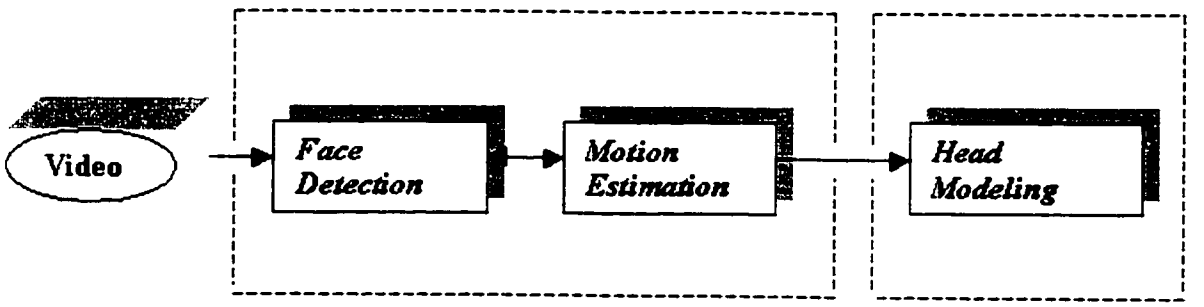


Fig. 1.3: The high-level Model-Based Video Coding architecture.

1.3. Contributions

This thesis represents a continuation of our work on model-based virtual environments [Mas97], [Cor98], [Pet99a], [Pet99b], [Geo99], [Spo99], addressing technologically difficult issues of real time tracking for head pose recovery in MBVC. The specific contributions are:

(i) real time efficient integration of muscle based 3D-head modeling and animation with face detection and tracking (Chapters 2 and 3);

(ii) real time implementation of computer vision algorithms for 2½D and 3D rigid head tracking (Chapters 4 and 5);

The application is implemented in an object-oriented, "bottom-up" fashion, combining machine vision, image processing, computer graphics and mathematical modeling fields, into a complex and challenging framework.

1.4. Thesis Outline

Chapter 2 of the thesis covers the head modeling section of the MBVC and presents the MBVC framework developed in the thesis. Chapter 3 describes a method for localization and recognition of human faces from cluttered color images. Since the recovery of the correct head motion parameters is crucial for MBVC, most of this thesis work addresses MBVC motion estimation in Chapter 4 and Chapter 5. Chapter 4 describes the *2½D-tracking* algorithm, using a LKF to smooth and predict the motion parameters. Chapter 5 employs an EKF and a Newtonian physical model, used to perform *3D head tracking* in real time. The proposed algorithm uses the generic mesh model to initialize the structure parameters of the EKF, through an interactive procedure. In order to validate the accuracy of the 3D head tracking system, we developed a rapid calibration technique using a sequence of images of a synthetic "standard" 3D head in lieu of real life head. The thesis concludes with a summary of the relative merits and shortcomings of the developed IFAT system approach and with a sketch of future research directions.

Chapter 2

Face Modeling and Animation

This chapter introduces the basics of the human facial modeling and presents the object oriented facial animation framework developed and used in the thesis.

2.1. Introduction

Facial modeling has been an active area of research in computer graphics for more than two decades. It benefits from and can contribute to the larger field of human body modeling. Facial modeling is also relevant to other fields such as medicine, and communications. It is recognized as a challenging project that requires a multidisciplinary effort. A facial model captures with a given degree of accuracy the form and function of a face, whether human or otherwise, in a way that makes the model useful for specific applications [Pel94]. State-of-the-art facial models for computer animation attempt to represent the geometry, photometry, deformation, motion, etc., of the various features associated with the face, as well as with the rest of the head and neck.

Applications

The range of applications that uses models of the human face is wide, including telecommunications, e-commerce, education, entertainment, medicine, computer-based interactive services etc. The amount of detail embedded into the face model varies from application to application.

- ***Telecommunication:*** This is the research area of the present project. Facial models are developed for use in video-telephony and teleconferencing, usually over low-bandwidth

channels. A realistic model of the human face consisting of a suitable selected set of facial features (parameters) is recovered at the emission end. This relatively reduced set of facial parameters is then transmitted to the receiving end, where a synthetic 3D image of the human face is reconstructed. During the transmission, the animation of the speaker's face is controlled by the facial parameters. This communication technique has the notable advantage of requiring a very low bit-rate.

- ***Education:*** Facial expression could play a significant role in an educational environment. For example, a face model that captures the expressions and anatomy of the human face may be used in telelearning applications over low bit-rate channels.
- ***Entertainment:*** In a complex environment computer generated characters capture facial expressions and human emotion of a "live performer". Prerecorded scripts can control the synthesis of requested facial expressions. One other example will be pre-animated video messages (i.e. E-mail).
- ***Medicine:*** "Preoperative simulation of corrective plastic surgery and dental treatment are of great interest to both practitioners and patients alike. Such applications demand precise models of particular individuals based on the bone and soft tissue of the head. A computerized system, which incorporates an anatomically complete model of the head and face, would provide surgeons with the capability to plan, and even rehearse, complex operations without undertaking costly and potentially dangerous exploratory surgery [Pel94]." Computer models help psychology in facial recognition and nonverbal communication areas.

2.2. Facial Animation Techniques

Facial expressions play a major role in the communication among people. Face image serves to identify a person, reflect individual's emotional state, allowing to other people to relate with that person. Therefore, the control mechanism used to stimulate facial expressions plays an important role in facial animation framework.

There are four fundamental approaches to facial animation [Pel94]:

(i) shape-interpolation

(ii) key-node parameterization

(iii) muscle-based animation

(iv) physically-based animation

2.2.1. Shape Interpolation

This facial animation technique originates from the early work of Parke [Par72]. The process interpolates among a set of 3D key facial postures obtained by different computer vision methods (stereo-photogrammetry, structured light, 3D-digitizers etc.). This set of facial postures has the property of topological equivalence of the wireframe facial models. Once a complete set is derived, then in between interpolation of the (x, y, z) vertex coordinates can be computed. Parke's algorithm generates the frames between two chosen key-pose expressions. This approach requires complete specification of the model geometry for each facial expression, the result being a very labor-intense technique.
DeGraf and Wahrman [Deg90] used puppetry together with shape interpolation to animate expressive characters. Unfortunately, this technique is limited to the matrix of facial postures, which prompted the development of parameterization schemes.

2.2.2. Key-Node Parameterization

In his 1974 PhD dissertation [Par74], Parke pioneered the idea of the animation of facial expressions based on a parametric control algorithm. This parameterization reduces the face model to a small set of control parameters that can be "hard-wired" into a particular facial geometry. These parameters are only loosely based on the dynamics of the face. For example, the expression on the eyebrow from surprise involves the manipulation of five or six vertices of the facial geometry. Parke proposed two major categories of facial parameters [Par74]: conformation parameters, which control the structure of a face, and expression parameters, which control its emotional content. Interpolating between key-node positions updates the node

positions after a change of parameter values. Other operations such as rotation, scaling, and translations of facial nodes are also needed by parametric controls.

2.2.3. Physically-Based Animation

Physically based animation attempts to represent the shape and dynamic changes of a face by modeling the underlying properties of facial tissue and muscle actions. Most of these models are based on mass-spring meshes and spring lattices, with muscle actions approximated by a variety of force functions [Pla81], [Ter90].

Platt and Balder [Pla81] proposed a muscle-based animation technique using a multi-domain model of the human face consisting of skin nodes, muscle nodes, and bones nodes. In order to animate this face model, they used a physically based elastic behavior connecting 3D facial nodes by springs. Applying forces to this elastic skin mesh via the underlying muscles generates a facial expression. For example, when a single muscle fiber contracts, a force is applied to a muscle point in the direction of its bone point. The force is then reflected along all arcs adjacent to the point. These reflected forces are then applied to their corresponding adjacent points. In this fashion, a force is propagated out from the initiating point across the face. Based on earlier successful applications of physics-based modeling of deformable objects in computer graphics [Ter88], [Ter89], Terzopoulos and Waters extended this method to the animation of human faces [Ter90], [Wat92]. They proposed a deformable face using facial muscles and forward dynamic skin models based on the biological structure of human skin [Ter90]. The biomechanical skin model consists of three spring-mass layers: an epidermal layer, a dermal layer, and a fatty layer. The dynamics of facial skin deformation are simulated with the Lagrange equations of motion with elastic properties of the skin incorporated. Furthermore, with each of these layers simulating the non-linear stress-strain and volume preservation properties of a real skin patch, their face model easily and naturally generated facial skin creases, like furrows, or wrinkles.

Guenter [Gun92] used a 3D finite-element face model to animate facial expressions. According to Guenter, his work took 20 minutes to generate 4 facial expressions, using specialized

hardware. Physically based models are computationally expensive and difficult to control with force-based functions.

2.2.4. Muscle-Based Animation

Waters studied a simplified version of the physically based animation using only facial muscles for the activation of the neighboring nodes in a facial mesh [Wat87]. Similar to Platt and Balder [Pla81], Waters used a simple spring mass model for the muscle simulation. The muscles have vector properties and are independent of the underlying bone structure. This results in a model independent of the facial topology. Reeves of Pixar Inc. adopted Waters' work in facial animation [Ree90]. Reeves and his colleagues developed the "Tin Toy" animation in which the computer animated actor "Billy" performed facial expressions using Waters' muscle-control scheme. The Waters' muscle-based approach has also been extended to B-spline surfaces [Ste95].

We adopted in this thesis the Waters' method because of its distinct advantages:

(i) independence of particular facial geometry;

(ii) yields a reduced set of facial parameters (muscle activation values) convenient for low-bit rate communication;

(iii) low computation time;

A well-known muscle-based coding system, largely used for the facial animation field is the Facial Action Coding System (FACS).

Facial Action Coding System (FACS)

Two sign communication psychologists, Ekman and Friesen, in 1977 [Ekm86] developed the anatomically oriented FACS. Their system describes complex facial expressions in terms of basic facial muscle actions. Ekman and Friesen reduced the 50.000 distinguishable expression space to a comprehensive system, which could distinguish all possible visually facial expressions by using only 44 Action Units (AU's). The AU refers to a basic visual facial

expressions by using only 44 Action Units (AU's). The AU refers to a basic visual facial movement, which can not decomposed into smaller units. One or more muscles control the AU's. Complex facial expressions can be obtained by combining different AU's.

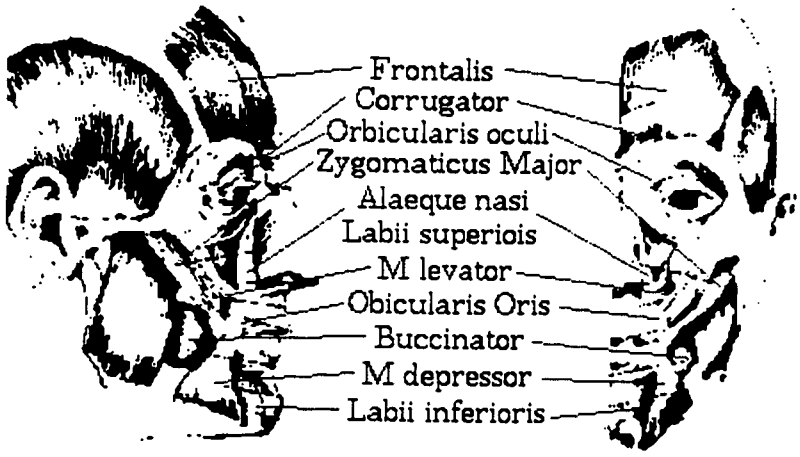


Fig. 2.1: Facial muscles [Wat87].

In Waters' scheme, muscles can be described as vectors with direction and magnitude in two and three dimensions. The direction is headed to the attachment point on the bone, and the displacement of the deformation depends on the muscle elastic constant and the tension of the muscle contraction. The model mimics at a simple level the action of three primary muscle groups of the face (Figure 2.1):

(i) *Linear muscle*, such as the *Zygomatic Major*

(ii) *Sphincter muscle*, such as the *Obicularis Oculi*

(iii) *Sheet muscle*, such as the *Frontalis Major.*

The predefined surface points associated with these three categories of muscles are such that the areas of influence are fan-shaped, rectangular and elliptical respectively. The deformations that exist in the surface are handled by displacing the surface points to produce adequate bump on the surface and result from low level muscle contraction. The resultant displacement $\|pp'\|$ of an arbitrary surface node p to p' is expressed by the following function:

$$p' = f(k, \beta_1, ..., \beta_n, p) \qquad (2.1)$$

where k is a spring constant and $\beta_1,...,\beta_n$ are parameters dependent upon muscle type.

The Linear Muscle

The linear muscle computes the deformation of the adjacent tissues represented by the node p. For the displacement of an arbitrary mesh node p (Figure 2.2a) from the position p to a new position p' within the circle segment $\|v_1 p_m p_n\|$ towards v_1 along the vector (pp'), the expression presented below is employed.

$$p' = p + akr\frac{pv_1}{\|pv_1\|} \qquad (2.2)$$

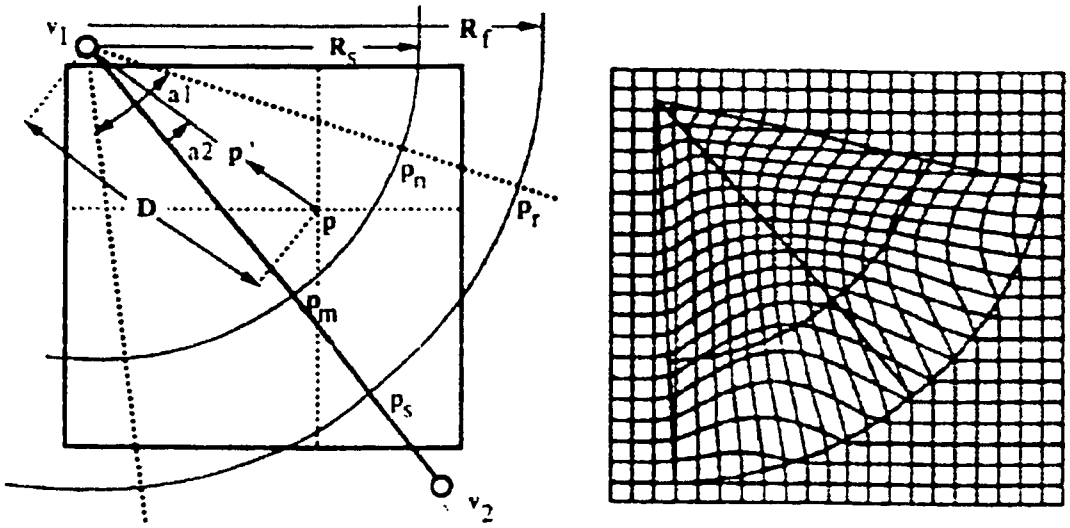
The new location p' is a function of an angular displacement parameter

$$a = cos(\mu), \qquad (2.3)$$

where μ is the angle between the vectors (v_1, v_2) and (v_1, p), D is $\|v_1 - p\|$, r is a radial displacement parameter:

$$r = \begin{cases} \cos\left(\frac{1-D}{R_s}\right) & , p \in sector[v_1 p_n p_m v_1] \\ \cos\left(\frac{D-R_s}{R_f - R_s}\right) & , p \in sector[p_n p_r p_s p_m] \end{cases} \qquad (2.4)$$

and k is a constant representing the elasticity of skin. When varying the elasticity constant, the mesh will deform toward the root of the vector producing the following visual effect (Figure 2.2b).



(a) (b)

Fig. 2.2: Linear muscle zone of influence (a) and the effect of a contraction (b) [Wat87].

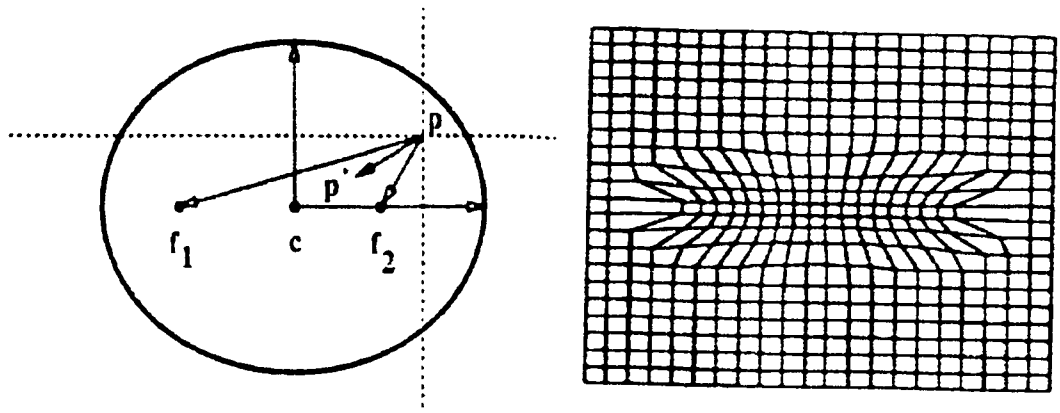
The Sphincter Muscle

The sphincter muscle contracts around an imaginary central point *c*. This muscle is represented by an elliptical shape with minor and major axis as parameters (Figure 2.3a). The displacement equation takes the form:

$$p' = f(k, L_x, L_y, p) \tag{2.5}$$

$$f = 1 - \frac{\sqrt{L_y^{\,2} p_x^{\,2} + L_x^{\,2} p_y^{\,2}}}{L_x L_y} \tag{2.6}$$

where k is the muscle spring constant, L_x, L_y, are the semi-major and minor axis of the ellipse.



(a) (b)

Fig. 2.3: Sphincter muscle zone of influence (a) and the effect of a contraction (b) [Wat87].

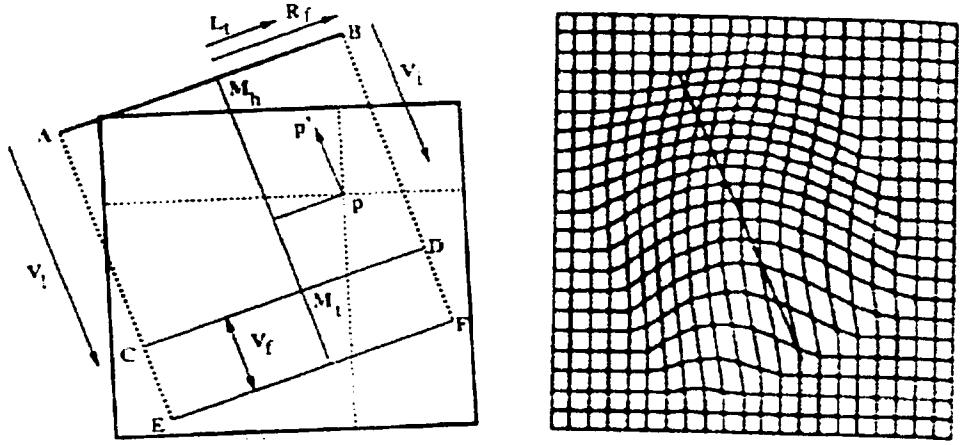
The Sheet Muscle

Sheet muscle consists of series of parallel fibers in which lie in flat region. An example is the *Frontalis Major* muscle on the forehead, which is primarily involved with the raising of the eyebrows [Wat87]. The displacement will be parallel to the central muscle vector. The new position p' of a node that moves from an initial position p is:

$$p' = f(k, d, p) \tag{2.7}$$

where k is the muscle spring constant and

$$d = \begin{cases} \cos\left(1 - \dfrac{L_t}{R_f}\right) & , p \in \text{sector}[ABCD] \\ \cos\left(1 - \dfrac{L_t}{R_f}\left(\dfrac{V_i}{V_l} + V_f\right)\right) & , p \in \text{sector}[CDFE] \end{cases} \tag{2.8}$$



(a) (b)

Fig. 2.4: Sheet muscle zone of influence (a) and the effect of a contraction (b) [Wat87].

In all cases a cosine function $k = f(\cos x)$ approximate the elastic properties of the skin. Replacing the cosine function with a nonlinear interpolant, or power function, allows simulating the skin aging with lower elasticity.

Facial expressions are sometimes created by the cooperation of different muscles with overlapping areas of influence rather than by just a single muscle. Since the muscles and their areas of influence are predefined in the model, the maximum factor by which the muscle can contract without displacing the surface points to absurd position is normalized to unity (the muscle in its relaxed state has a null contraction factor). The muscle contraction represents a sufficient set of parameters for the generation of any facial expression. By transmitting the reduced set of the muscle (or joint) contraction values, we avoid transmitting the larger set mesh point displacements (positions). This results in a compression of the model data to be transmitted.

A basic animation framework using muscle-based model can be described as follows:

- Muscles are described as belonging to one of three primary groups: linear, sphincter, and sheet. Each group describes a specific geometric deformation function emulating muscle action on skin tissue.
- The muscles control parameters are:

(i) location on bone and skin

(ii) zones of influence, and

(iii) type of contraction profile (linear, non-linear).

Each muscle functions independently.

- Muscles are then mapped to Action Units (AU's) in FACS. Consequently, orchestrating a proper sequence of AU muscle contractions can create desired facial expressions. Muscle control parameters are then scripted into a keyframe animation system. At each frame in the sequence, muscles are contracted that in turn deform the facial geometry. The resulting geometry is rendered in accordance with viewpoint, light source, and skin reflectance information.

2.3. An Interactive Facial Animation Implementation

The 3D facial model animation (at the receiver end) requires a major implementation effort. Our "Interactive Facial Animation Testbed" (IFAT) was implemented in a modular fashion, in order to facilitate experimentation with a variety of animation methods. When the animation system is used in Videophone mode, deformation parameters are received automatically. The IFAT consists of two main modules (Figure 2.5):

(i) *Animation/Rendering Module*

(ii) *Modeling/Control Module*

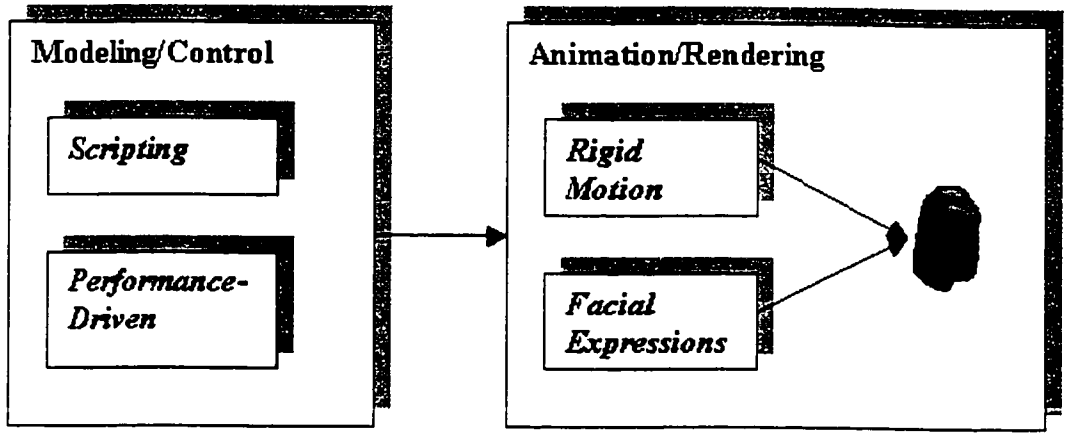


Fig. 2.5: The architecture of the Interactive Facial Animation Testbed.

2.3.1. Animation/Rendering Module

A "baseline" state of the facial model corresponds to a face in a neutral position, where simulated muscles are mapped to the wireframe model in a relaxed state (Figure 2.6). During the animation, a series of muscle induced deformations are applied to the facial model points in the baseline position, after which the face is rendered.

Eyes are constructed from spheres with iris and pupil. The eyes rotate simultaneously in their coordinate system. Eyelids are wireframe models, which can blink automatically during the animation.

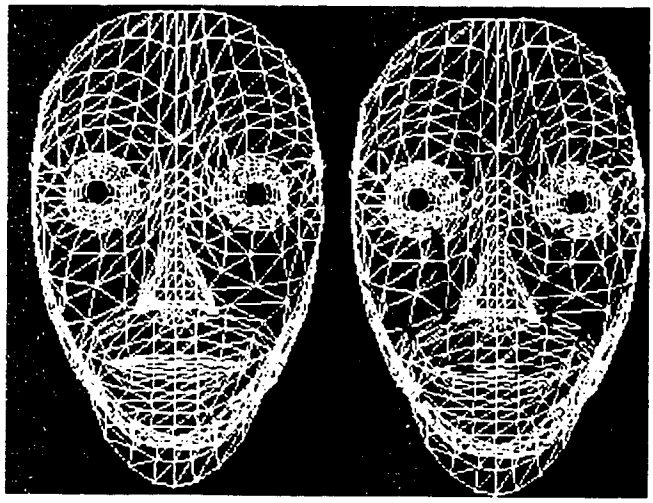


Fig. 2.6: Mapping the muscles onto the wireframe model.

Two types of "controllers" control the whole facial deformation at the low level:

(i) *Muscle* controllers: 16 muscle parameters using *muscle-based* model.
(ii) *Jaw, Eyelids*, and *Eyes* controllers: 3 parameters using *key-node parameterization* model.

It was shown [Wat87] that a reduced set of parameters: 8 pairs of facial muscles (16 muscles) plus the jaw, eyelids, and eyes motion parameters, mapped to AU contractions, can fairly cover an extensive space of human facial expressions. The animation module can also control the rigid motion of the human head, through basic transformations (translation, rotation and scaling) of the 3D-head frame (Figure 2.5). The object-oriented implementation of the system permits to add new controllers (muscle-based or parametric-based) in the future.

2.3.2. Modeling/Control Module

The Modeling/Control Module contains two blocks that allows for a textual scripting description of the animation program or for a gesture-driven animation:

(i) scripting block
(ii) gesture/performance block

Scripting

The Scripting block was conceived as an open interface, which enables different animation modules to be easily added. Currently the scripting block has a low-level implementation, which allows generating facial expressions in two modes:

(i) Constructing specific facial expressions (Figure 2.7), by combining AU muscle activations (muscle low level programming).

(ii) Selection of one of the six primary facial expressions that communicate anger, disgust, fear, happiness, sadness and surprise. These are obtained by combining predefined AU's (Figure 2.8) (macro level expression programming).

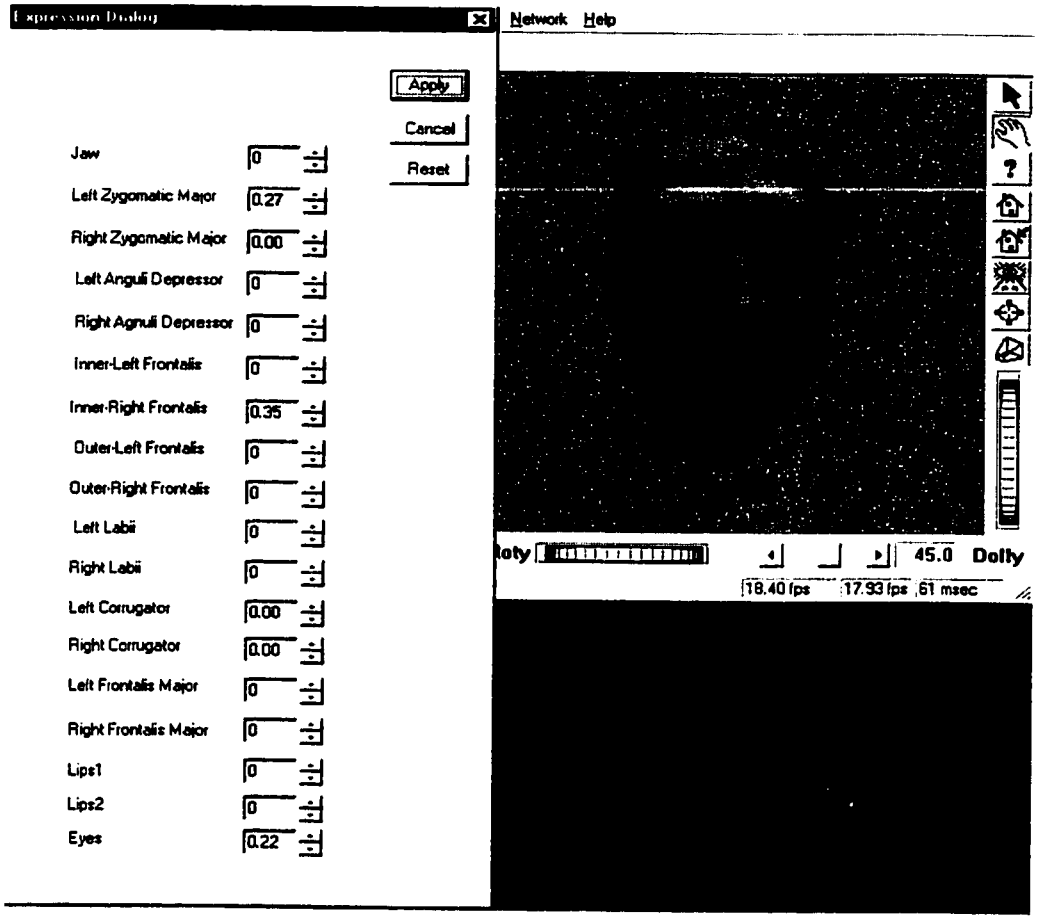


Fig. 2.7: Muscle level setting (programming) of a specific expression.

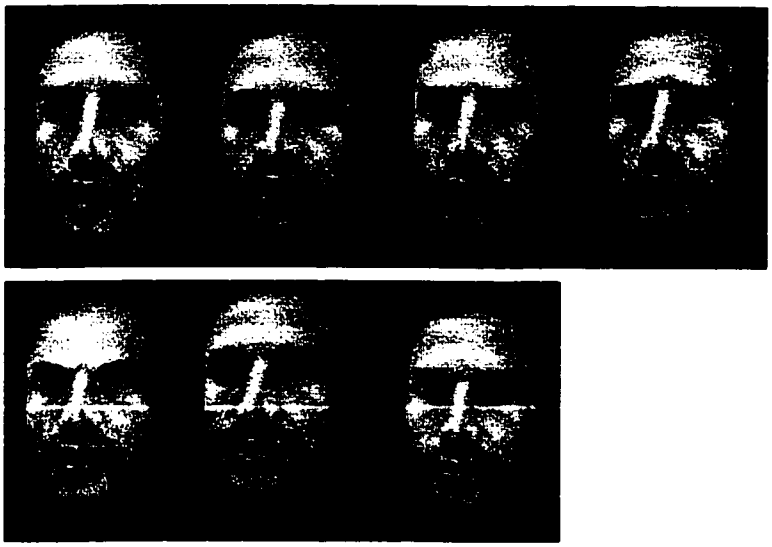


Fig. 2.8: Fundamental (macro level) expressions generated by our framework (surprise, disgust, fear, sadness, anger, happiness, and neutral positions).

This low-level parameterizations provide a basis for a future high level of programming/scripting techniques:

- *Speech-Driven Level:* animating directly from a speech soundtrack
- *Behavior-Driven Level:* expressing the desired actions in terms of "behaviors"
- *Story-Driven Level:* (children stories like) based on major activities such as: story understanding, stage direction, and action generation.

Performance-Driven Animation

Generally, Performance-Driven animation involves measuring real human actions to drive synthetic characters. Data used to drive the animation comes from interactive input devices such as: data gloves, instrumented body suits, vision-based motion system, and other various sensors. Since our goal was to construct a real-time, low-cost videoconferencing system, we implemented the performance-driven control as a vision-based module, which is detailed in the next chapters (3,4, and 5).

Chapter 3

Face Detection

Although a lot of work has already been done in this research area [Sob96], [Pen94], [Pog91], [Mck97], [Jie98], [Bur89], [Bru93], the recognition of human faces out of a scene with cluttered background is still an unsolved problem that requires further investigation. This chapter describes one basic function of the Performance-Driven block (Figure 2.5): the real-time detection of the human face at the emission end of our IFAT system. The algorithm has the ability to detect faces under different scale, and frontal orientation.

3.1. Introduction

There are many applications that require real time algorithms for face localization and recognition as for instance MBVC, security systems, "mug shot" matching [Che95]. Because of variations in illumination, background, orientation and facial expressions, the problem is very complex.
The first step in the head pose recovery from a sequence of images is the identification and location of the face in first image frames.

Face detection is a difficult problem for three main reasons:

(i) Although most faces have similar structures with the facial features arranged in roughly the same spatial configuration, there is still a large variation due to non-rigidity and textural differences. There are significant geometrical or textural differences even between images of the same person's face due to changes in expression and facial makeup. Because of that, traditional fixed template matching techniques, which work well for rigid and articulated non-deformable objects do not perform adequately for human face detection.

(ii) Face detection is also difficult because cosmetic features, such as glasses or a moustache, can either be present or totally absent from a face. Furthermore, the features, when present,

can cloud out other relevant facial features. All this adds more variability to the range of permissible face patterns that traditional face detection systems can handle.

(iii) Face detection can be further complicated by unpredictable imaging conditions in an unconstrained environment. Because faces are essentially 3D structures, any change in the light distribution can change significant shadows on a particular face, resulting in an increased variability of the observable 2D facial patterns.

Several methods using texture, depth, shape, color or intensity information were developed over the years for the detection of facial regions and features [Sob96], [Pen94], [Mck97]. Some approaches first identify relevant facial features, and then decide that these actually represent or not a face. Other approaches identify an image area as being a regional face descriptor based on color, shape, motion etc. Some applications may require only to identify, localize and track face areas. Other applications require facial feature recognition that will be used for motion tracking. The main difference between face and feature tracking is that the face is considered as a rigid object, while the face features as deformable patterns. In the following sections we present techniques employed for face and feature detection [Bey93], [Pog91]:

- *Feature-Based* detection including:
  - (i) *pictorial* approach, using *correlation templates*
  - (ii) *parameterized model* approach, using *deformable templates*
  - (iii) *grey level interest operators,* using *spatial image invariants.*
- *Color-Based* detection

3.2. Techniques for Face Detection

Facial feature detection could be reduced to the problem of localization of some major facial features such as the eyes, nose, mouth, and face outline.

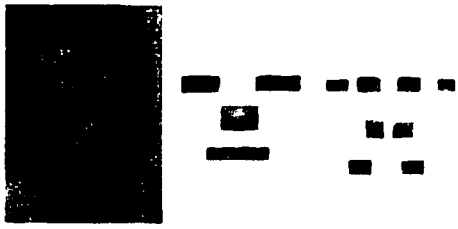
3.2.1. Feature-Based

Pictorial approach using *correlation templates*

In the pictorial approach, generic pixel-based models of facial features are matched against the image. In the case of the template-based systems, correlation or the matched filter methods are used to measure the difference between feature models and image candidates. The generic feature nodes are templates of the major facial features [Bic91], [Bur89], [Bru93]. In the case of neural networks matched filters, the templates are implicitly coded as weights of hidden layer neurons [Vin92]. In this approach implicit feature templates are not explicitly defined but learned from "feature" and "non-feature" examples.

In many practical applications the face patterns are too varied and can not be modeled by fixed correlation templates. In such a case a sub-features detection is used [Bru93]. The idea is to use the invariance property of the facial sub-features (e.g. corners of the eye) to describe a full feature (e.g. the eye) by few fixed templates (Figure 3.1). While a full template matching could fail to detect and track a non-rigid feature (e.g. mouth), the sub-feature matching can succeed. Further, the method can even infer the presence of faces by analyzing the spatial relationship between the detected sub-features [Pog91].

A related approach to correlation templates is that of view-based eigen-templates [Pen94]. This approach assumes that the set of all possible face patterns lies into a small sub-space of a multi dimensional feature space. As a first step, the dimensionality of the feature space is reduced to a new reference frame that still adequately describes the set of models. Usually, these models are coming from a training set of faces and/or features. The new reference frame is constructed as a linear combination of few uncorrelated principal components (basic features) or eigenvectors. These principal components provide enough resolution to variation that can occur between face/features in any image of the set. The largest eigenvectors account for variations like those due to illumination changes and predominant differences in facial appearance between individuals [Pog91]. This face/feature recognition is based on the measurement of the "Distance From Faces/Features Space" (DFFS) at each pixel of the observed image. If the resulted value is below a certain threshold, the pixel is labeled as "face/feature".



(a) (b) (c)

Fig. 3.1 Full templates (b) extracted from the Marius' image (a), and their sub-features (c).

Parameterized model approach using *deformable templates*

Deformable templates are similar in principle to classical correlation templates, except that the former has some built-in non-rigidity component. To find faces in an image with a deformable template, one essentially has to fit the template to different parts of the image and threshold the output for matches. One deformable template method uses hand constructed parameterized curves and surfaces to model the non-rigid elements of faces and facial sub-features, such as: eyes, nose and lips [Yui92]. These parameterized curves and surfaces are fixed elastically to a global template frame to allow for minor positional variations between facial features. The matching process attempts to align the template with one or more pre-processed versions of the image, such as the peak, valley and edge maps. An energy functional will constraint the alignment by attracting the parameterized curves and surfaces to corresponding image features, while penalizing "deformation stress" in the template. The best-fit configuration is found by minimizing the energy functional, and the minimum value also serves as a closeness measure for the match.

Grey level interest operators using *spatial image invariants*

Image-invariant techniques assume that even though faces may vary, there are spatial image relationships, common and possibly unique to all faces. Face detection requires compiling image invariant and checking for positive occurrences of this invariant at all image candidate locations. One image-invariant algorithm is based on a set of observed brightness invariant between different parts of the human face [Sin94]. The idea is that while illumination variations can significantly alter brightness levels at different parts of a face, the local brightness distribution remains largely

unchanged. For example, the eye regions of a face are almost always darker than the cheeks and the forehead, except possibly under some very unlikely lighting conditions. Similarly, the bridge of the nose is always brighter than the two anking eye regions. The algorithm encodes the observed brightness as a ratio template, which is used as a pattern recognition parameter. A ratio template is a coarse spatial template of a face with a few appropriately chosen sub-regions, roughly corresponding to key facial features. The brightness constraints between facial parts are captured by an appropriate set of pairwise brighter-darker relationships between corresponding sub-regions. An image pattern matches the template if it satisfies all the pairwise brighter-darker relationships. The symmetry operator is a powerful pattern recognition tool for finding facial features in arbitrary natural scenes without a priori knowledge of the world. This method is robust even if the face isn't directly facing the camera, the position changes, or the background isn't uniform [Jeb96].

3.2.2. Color-Based

Color has been used in machine vision systems for image segmentation [Wla94], [Sob96] and face recognition [Fin92], [Swa91]. Using color descriptors has several advantages over the other face features:

- Lower computational time, allowing real-time performance on standard hardware.
- Orientation invariant under certain lighting conditions, robustness under partial occlusion, rotation in depth, and scale changes.

Due to these significant advantages, we chose a color-based face detection technique. A detailed description of this technique is presented in Section 3.3. Color information (skin and hair color), edge information (intensity and sign), and multiple active contour models are employed in [Yok96] to extract the contours of facial features (eyes, nose, mouth, etc.), and their feature points.

3.2.3. Other Face Detection Techniques

[Luo96] describes a method for automatically modeling the human face providing a realistic 3D structure and texture description of a specific face for model-based facial images coding. Boundaries of the face, mouth, and eyes are located first using active contour models ("snakes")

guided by artificial neural networks. Then, a 3D general wire-frame face model (WFM) is fitted to a 2D image, using important features (descriptors) extracted from the image. Finally, colored face texture is mapped to the 3D model. Renders [Ren96] presents a method for an automatic location of facial features (eyes, nose and mouth) in image sequences using a neural network approach. The sought features are modeled as structural assemblies of sub-features. An upright frontal-face neural network-based detection system is described in [Row98]. Here, a retinally connected neural network examines small windows of an image and decides whether each window contains a face and an supervisor system arbitrates between multiple networks to improve performance over a single network. [San98] proposes an algorithm for face extraction from a color image, where the lip and skin color pixels in an image are extracted first, on the basis of statistical probability techniques. McKenna et al. [Mck96] integrates motion-based tracking with model-based face detection to produce segmented face sequences from complex scenes containing several people. The motion of moving image contours was estimated using temporal convolution and a temporally consistent map of moving objects was maintained, and faces were detected using a neural network. The symmetry characteristic of the human visage is used in [Den97] to extract a set of significant features for the detection task. This method has a very good success rate in the identification of facial elements. It also has the advantage of a fast computation that makes it suitable for real-time applications like personal identification and communications [Den97].

3.3. A Color-Based Face Detection Implementation

This section presents a color- and shape-based regional descriptor algorithm that we have implemented for real-time face detection.

A major difficulty with using color in computer vision applications is the color value variation due to lighting changes. This is particularly apparent in RGB (red, green, blue) space. Intensity is distributed throughout all three parameters, rendering color values highly sensitive to scene brightness. A simple approach to color constancy is to use either the:

- HSV color space which consists of hue angle (H), color saturation (S) and brightness (V) or,
- "rg" (red, green) normalized, known as "pure" colors in absence of brightness.

3.3.1. Color Space Transformation

The most popular color space used for digital images is the RGB. According to the tristimulus theory of Wyszecki and Stiles [Wys82] color can be represented by three basic components, separated by three separate color filters S_x, $x = R, G, B$, applied to any light radiance $E(\lambda)$:

$$\begin{aligned} R &= \int_{\lambda} E(\lambda) S_R(\lambda) d\lambda, \\ G &= \int_{\lambda} E(\lambda) S_G(\lambda) d\lambda, \\ B &= \int_{\lambda} E(\lambda) S_B(\lambda) d\lambda \end{aligned} \tag{3.1}$$

In the RGB space, the *(R, G, B)* parameters of pixel $p(R,G,B)$ represent the brightness of the fundamental colors. When the fundamental colors *R, G, B* of two pixels, $p_1(R_1,G_1,B_1)$ and $p_2(R_2,G_2,B_2)$, are proportional, i.e.,

$$\frac{R_1}{R_2} = \frac{G_1}{G_2} = \frac{B_1}{B_2} \tag{3.2}$$

they have the same color but different brightness. The human visual system adapts to different brightness and various illumination sources such that a perception of color constancy is maintained within a wide range of environmental lighting conditions [Bra93][Jie98]. Therefore it is possible for us to remove brightness from the skin-color representation, while preserving an accurate, but low dimensional color information. Since the brightness is not important for characterizing skin colors, under the normal lighting condition, we can represent skin-color in the normalized chromatic color ***rg*** space, thus reducing the dependence on changing in space lighting. Chromatic colors ***r, g*** are defined by a normalization process:

$$\begin{aligned} r &= R/(R+G+B), \\ g &= G/(R+G+B) \end{aligned} \tag{3.3}$$

Having $r + g + b = 1$, the blue color will be redundant.

There also are many color models based on the human color perception. Some models use hue, saturation, and intensity (HSV) components (Figure 3.2). The HSV model is related to the RGB model by the following formulas:

$$V = (R + G + B) / 3, \qquad (3.4)$$
$$S = 1 - min(R, G, B) / V,$$
$$H = arctan(\sqrt{3}(G - B), 2R - G - B).$$

where *arctan (y, x)* gives the angle between horizontal axis and the line *(0,0)-(x,y)*.

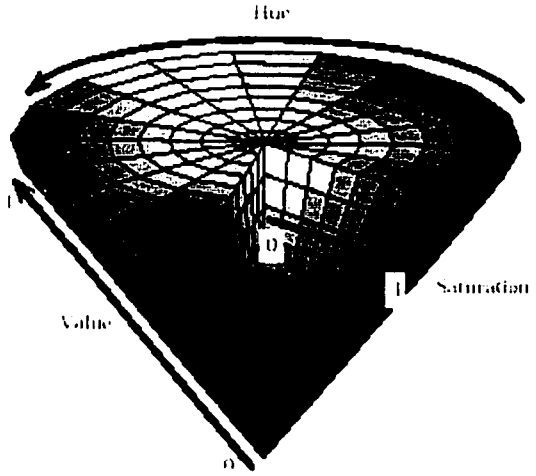


Fig. 3.2 Hue-Saturation-Value Space [Mat96].

3.3.2. Stochastic Color Skin-Model

Recently, more methods have been proposed for color-based detection and tracking of skin-colored objects: [Wre98] for body, [Mck97], [Jie98] for face, and [Nur98], [Kac97] for lips. The system developed by Wren et al. [Wre98] enabled tracking of people, in controlled environments, with static cameras. Each pixel in an image belongs to a spatial and color vector map. The feature vectors were clustered in feature blobs, with a collection of blobs representing a person. In this case

the tracking was limited to people with homogeneously colored regions and a constant background. Human face was modeled as an adaptive normal distribution in *HS* [Mck97], or *rg* [Jie98] space. During tracking, the models can cope with changes in lighting conditions. The *HS* model was used in [Nur98] and [Kac97] to detect and track the lip motion. The Bayesian inference will classify a pixel as "lip" or "face", where the face was used as a background of the lip model.

The skin color distribution of people with different skin colors forms a compact cluster, with a regular shape in *rg* (or *HS*)-chromatic color space. Our experimental 3D-histogram (Figure 3.3) shows that the human face colors of different people under different lighting conditions in the *rg* (*HS*) color space have similar Gaussian distributions. The experiment leads to the idea of modeling a set of human faces as a *Mixture of Gaussian* (MOG) distributions in the 2D normalized color space (*rg* or *HS*).

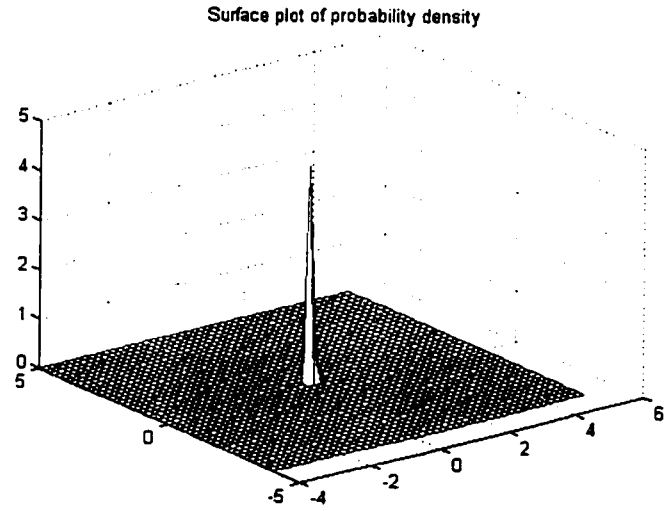


Fig. 3.3 Skin-Color Distribution in rg-Space

Mixture Modelling

The modeling can be reduced to the problem of identifying the parameters of a *Probability Density Function PDF*, $p(x)$, based on a finite number of samples x_i from that distribution. There are three basic approaches to density estimation [Bis95]:

- *Parametric* methods used when the density function is *a priori* known. The model will contain several parameters, which are then optimized by fitting the model to the data set. The disadvantage is that the chosen parametric function might not represent too well the actual density.
- *Non-parametric* methods used when the density function is not known. The shape of the density function will be determined by the experimental data set. The drawback in this case is that the number of model parameters grows with the data set.
- *Semi-parametric* methods, will mix the best of both the above estimations by allowing a general class of functions, with a number of adaptive parameters independent of the data set, which can lead to building a more flexible models. A particular form of such a density function is called a *mixture model* [Bis95]. Fitting the model to the data set means finding its parameters by training, based on maximum likelihood. The algorithm that is generally used for learning the parameters is the *Expectation-Maximization (EM)* algorithm of Dempster et al. [Dem77].

Our detection system employs a "semi-parametric" method for color density estimation, namely a *Gaussian mixture model*. The conditional density for a pixel **x**, belonging to an object **O** is linearly modeled as a mixture with **m** component densities:

$$p(x \mid O) = \sum_{j=1}^{m} p(x \mid j) P(j) \qquad (3.5)$$

where $p(x|j)$ corresponds to the prior probability that pixel x was generated by component j, and $P(j)$ represents the mixing parameter, also called *weight* or *prior*, with

$$\sum_{j=1}^{m} P(j) = 1 \qquad (3.6)$$

Each mixture component is a Gaussian with mean μ and covariance matrix Σ, i.e. in the case of a 2D colour space (rg or HS):

$$p(x|j) = \frac{1}{2\pi|\Sigma_j|^{\frac{1}{2}}} e^{-\frac{1}{2}(x-\mu_j)^T \Sigma_j^{-1}(x-\mu_j)} \tag{3.7}$$

Expectation-Maximization (EM) algorithm was used for fitting such a mixture to a data set represented by multiple skin color samples from face images.

Offline Model Training and Testing

We have developed a face mixture model, using the EM algorithm. This EM algorithm provides off-line maximum-likelihood estimates of both mixture priors and the underlying Gaussian parameters of equation (3.7). The *"Mixture"* program was developed for *PDF* model training on large sets of skin samples and then the resulted model was validated on single images. When modeling an object in the two-dimensional *rg* space as a "full covariance" mixture of $m(w, \mu_r, \mu_g, \Sigma)$ components, six Gaussian parameters and the prior need to be estimated for each component of the mixture model.

$$\mu_r = \frac{1}{N}\sum_{i=1}^{N} r_i \quad ,$$

$$\mu_g = \frac{1}{N}\sum_{i=1}^{N} g_i \quad , \tag{3.8}$$

$$\Sigma = \begin{bmatrix} \sigma_{rr} & \sigma_{rg} \\ \sigma_{gr} & \sigma_{gg} \end{bmatrix} ,$$

where μ_r and μ_g represent the *r*, *g* mean values of the training set, Σ is the covariance matrix of *r*, *g* pixels, *N* is the number of pixels of the training set.

Figure 3.4 summarizes the mixture parameter estimation algorithm [Dem77], [Kac97].

Step 1. **Select** M initial cluster centers, $m_1, \ldots, m_M$

Step 2. **Initialize** the estimates using *K-means* clustering,

For k = 1...M

$$w_k^{[0]} = \frac{N_k}{N}$$

$$\mu_k^{[0]} = \frac{1}{N_k} \sum_{x \in X_k} (x - m_k)$$

$$\Sigma_k^{[0]} = \frac{1}{N_k} \sum_{x \in X_k} (x - m_k)(x - m_k)^T$$

where N is the total number of pixels, and N_k is the number of pixels belonging to k-th cluster.

Step 3. **Iterate** for $i = 1 \ldots nr\_of\_iterations$

EM, k = 1...M

$$w_k^{[i]} = \frac{1}{N} \sum_{x \in X} p^{[i-1]}(k \mid x)$$

$$\mu_k^{[i]} = \frac{N}{w_k^i} \sum_{x \in X} p^{[i-1]}(k \mid x) x$$

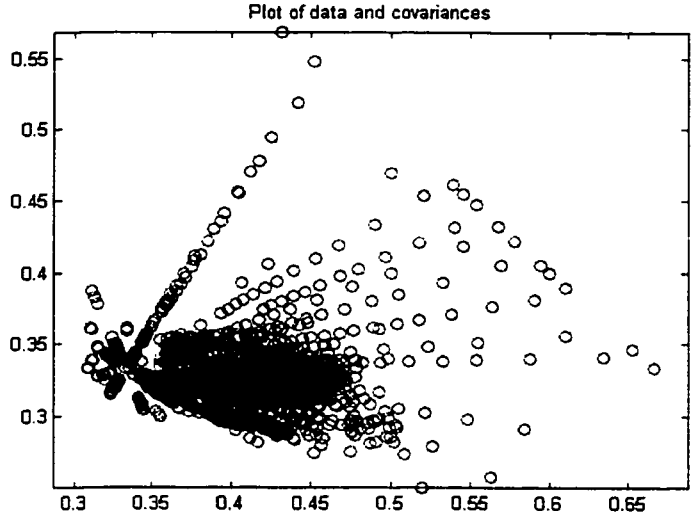
$$\Sigma_k^{[i]} = \frac{N}{w_k^i} \sum_{x \in X} p^{[i-1]}(k \mid x)(x - \mu_k)(x - \mu_k)^T$$

Until $\mu_k < \varepsilon$ for each mixture component M.

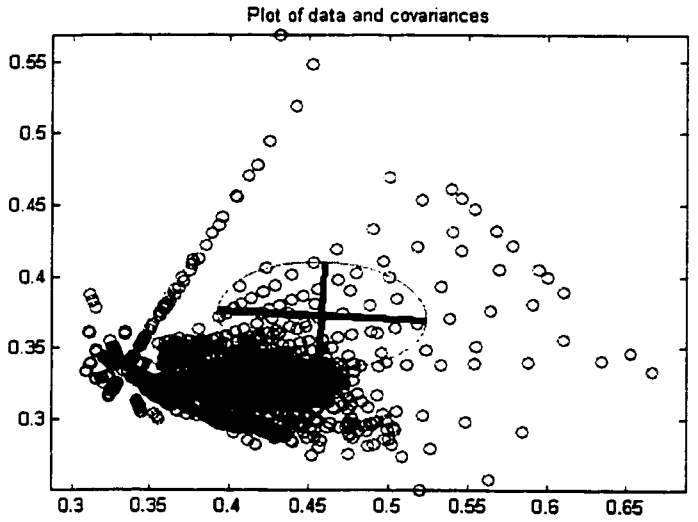
Fig. 3.4: EM learning of mixture parameters in the rg-space [Den77].

We are using the distribution based on a training set of 40 images of faces from different races (African American, Asian and Caucasian). In order to obtain a limited level of intensity variance, we experimented with face color modeled in *rg*-space, and *HS*-space, where pixels with very high or very low intensities were discarded. Usually, it is sufficient to model the face as a single "full covariance" or even a "diagonal covariance" Gaussian. Figure 3.5 present the fitting process of a

mixture model to the data set, using a single or multi component model with "full" or "diagonal" covariance matrices.



(a)



(b)

(c)

Fig. 3.5: Fitting mixture parameters on face color data distribution

(a) One Gaussian with " diagonal" covariance .

(b) Three Gaussian with "full" covariance.

(c) Three Gaussian with "full" covariance ("Mixture" output).

As shown in Figure 3.5(a), one "diagonal" Gaussian can fairly model the face color span in *rg*-space. The *EM* iteration number was chosen equal to 50 for all cases. At the end of training the mixture parameters are saved, for future use as input into detection and tracking phases.

The developed *"Mixture"* software can test, as illustrated in Figure 3.6, the mixture model on single images from the training or test set. The testing is done on "pixel-by-pixel" basis at the mouse cursor point, or "full image" basis, where the whole image is taken in consideration. Figure 3.6 shows a mixture model test on snapshots taken from live image sequences. The framework status bar indicates the pointed pixel position and its face classification probability. Note the "face-like" color of some of the background objects, and the capability of the trained model to discriminate between face and other object colors.

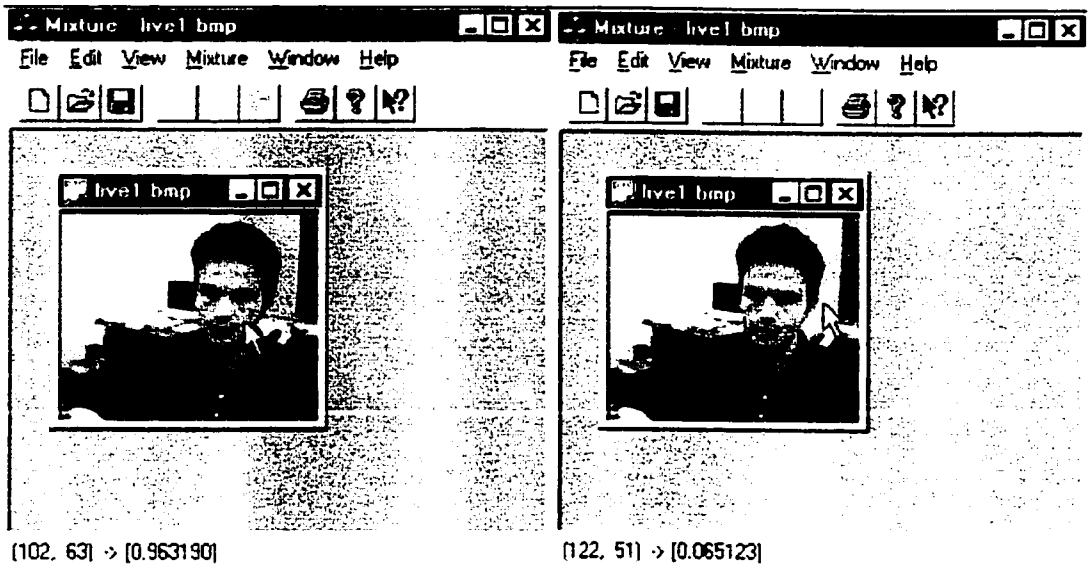


Fig. 3.6: Testing Process. "Face" and "Non-Face" pixel classifications.

3.3.3. Non-Stochastic Color Skin-Model

Under certain lighting conditions it is sufficient to consider specific hue and saturation values as discriminating color information for the segmentation of skin-like regions at the detection stage. The hue and saturation domains, which describes the human skin color, can be defined or estimated a priori and used subsequently as reference for any skin color. In our case we have chosen the parameters as follows: $S_{min} = 0.23$, $S_{max} = 0.68$, $H_{min} = 0°$ and $H_{max} = 50°$ [Sob96]. These values represent a hexagon sector in hue-saturation map as shown in Figure 3.7.

Face segmentation tests validated the correctness of this non-stochastic model. However, the stochastic skin model offers a much better generality allowing for instance to model the face color with two Gaussian components, one capturing the indoor light, the other the outdoor light influence in skin color.

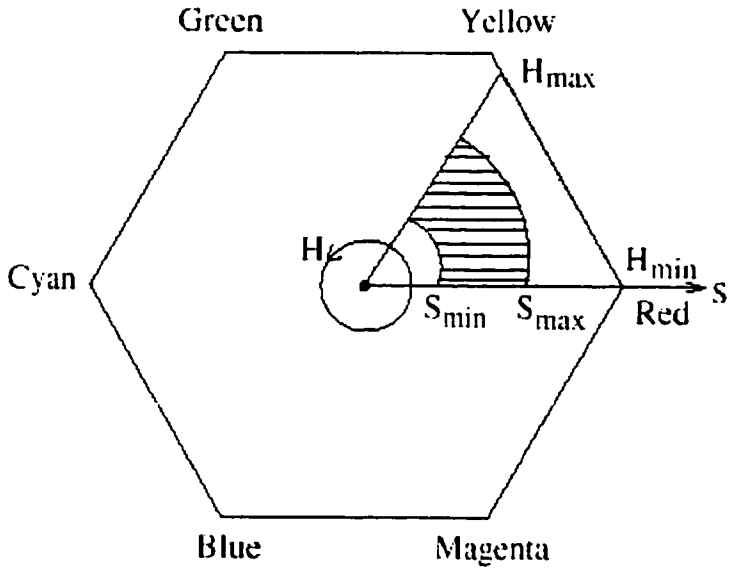


Fig. 3.7: Skin segmentation in HS-space [Sob96].

3.3.4. Face Shape Evaluation

Once the first frame of the sequence is segmented based on color information, the face detection algorithm will search for faces based on shape. The oval shape of a face can be approximated as an elliptical outline. The performed detection can be edge- [Ele95], [Bir97] or region-based [Sob96], [Wre97]. The region-based algorithm has the advantage of being more robust against noise and changes in illumination.

To find faces in images we are searching for elliptical components based on region growing algorithm applied at a coarse resolution of the segmented image. Normally, we obtain one large connected area for the face, a smaller one for the hands and small connected components in the background (Figure 3.8a). Each *connected component CC* is approximated by its best-fit ellipse based on moments [Jai89], [Sob96]. An ellipse is defined by its center (center of gravity) (μ_x, μ_y), length a and b of its minor and major axis (ellipse size), and its orientation θ.

$$\mu_x = \frac{1}{N}\sum_{(x,y)\in CC} x$$

$$\mu_y = \frac{1}{N}\sum_{(x,y)\in CC} y$$

$$\theta = \frac{1}{2}\arctan\left(\frac{2\mu_{1,1}}{\mu_{2,0}-\mu_{0,2}}\right) \qquad (3.9)$$

$$a = \left(\frac{4}{\pi}\right)^{\frac{1}{4}}\left[\frac{(I_{\max})^3}{I_{\min}}\right]^{\frac{1}{8}}$$

$$b = \left(\frac{4}{\pi}\right)^{\frac{1}{4}}\left[\frac{(I_{\min})^3}{I_{\max}}\right]^{\frac{1}{8}}$$

where N represents the number of pixels of the connected component CC, and I_{min}, I_{max} denote the least and the greatest moments of inertia of an ellipse with orientation θ.

$$I_{\min} = \sum_{(x,y)\in CC}\left[(x-\mu_x)\cos\theta-(y-\mu_y)\sin\theta\right]^2$$

$$I_{\max} = \sum_{(x,y)\in CC}\left[(x-\mu_x)\sin\theta-(y-\mu_y)\cos\theta\right]^2 \qquad (3.10)$$

At this stage we have several face candidates, and the next logical step will be the selection of the real face, by assessing how well the connected component CC is approximated by its best-fit ellipse E. Many researchers choose the face as being the largest "skin-colored" blob (Figure 3.8b) [Wra97], [Mck97], [Nur98]. In this case if the background contains "face-colored" and "face-sized" objects, the algorithm could yield false face detection. Besides the color constraint, our detection module uses the shape size and geometrical constraints.

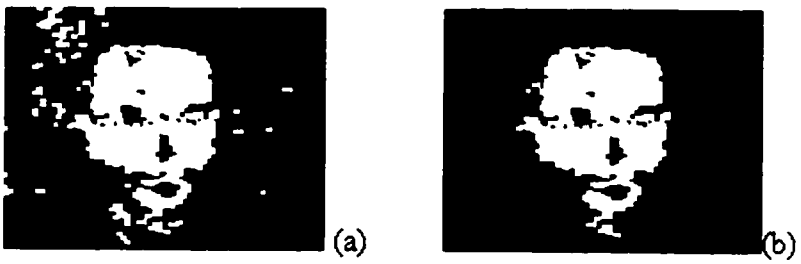


Fig. 3.8: The selection process of a face as the largest blob.

In order to increase the computational speed, we use offline pre-computed elliptical outlines. We call them "matching-ellipses", as opposed to the "detection-ellipses" computed as described above. The shape evaluation stage of the detection algorithm is illustrated in Figure 3.9. The *Step 5* of the algorithm represents the initialization stage of the 2½D tracking module.

Step 1. **Detect** face candidates, and **fit** "detection ellipses"

$E_{d1}, E_{d2}, \ldots E_{dn};$

Step 2. **Select** among candidates, based on shape size:

$L_1 \leq (minor\ axis)_i \leq L_2,\ i = 1,\ldots n$

$\Rightarrow E_{d1}, E_{d2}, \ldots E_{dk},\ k \leq n;$

Step 3. **Replace** "detection ellipses" with "matching ellipses"

$E_{ti} = E_{di};\ i = 1,\ldots,k;$ *(Figure3.10)*

Step 4. **For each** E_{ti},

evaluate a measure function

L_i("skin-inside", "non-skin-outside"); $i=1,\ldots k;$

Step 5. **If** $L_i \geq treshold;$

Then

select $face = E_{ti};$

Fig. 3.9: Shape Evaluation Algorithm.

Fig. 3.10: Fitting a matching ellipse on detected face.

Chapter 4

Tracking for 2½D Head Pose Recovery

This chapter describes the tracking function of the Performance-Driven block (see Figure 2.5) of the IFAT system. The developed technique extracts the 3D position and planar orientation of the head.

4.1. Introduction

To detect and locate a human face, the system will process the image, identifying relevant features, and then use these features to recognize and determine the location of the face. Tracking finds and locates the relevant facial features in a sequence of images. Tracking should allow estimating the motion while locating the face. There are three main tracking techniques [Hor86], [Sha95]:

(i) *Feature-based* methods, which extract image features and track their movement from frame to frame.

(ii) *Optical-flow* methods, which use spatial and temporal partial derivatives to estimate the image flow at each location in the image.

(iii) *Correlation-based* methods, which use similarity in brightness patterns to determine the motion vectors.

4.1.1. Feature-based tracking

An image feature is a low level image descriptor that can be classified into three main types:

(i) region-based
(ii) edge-based
(iii) point-based

Regions (or "blobs") correspond to smooth surface patches, or loci of zero-dimensional changes [Sha95]. Reliable tracking of such regions is often difficult, since minor changes between frames (due to image noise or motion) can lead to very different segmentation in consecutive frames [Wil93], [Ros91].

Edges are loci of one-dimensional change [Sha95], located where the change in intensity is significant in one direction. They are generally detected by finding either maxima in the first image derivative [Can86], or by zero-crossing in the Laplacian of the Gaussian of the image [Mar82]. Until the recent advent of *Energy-based Deformable Models* [Yui89], and *Active Contour Models* [Kass88], arbitrarily curving edges were difficult to describe and track. These models have been used to improve adaptive tracking by requiring the template to vary slowly or smoothly. However, the models necessitate high resolution of the input data, manual initialization and do not help to reason about occlusion events.

Point features are distinctive object points in the image, which can be easily tracked. Common point features include local maxima of directional variance [Mov80], maximum curvature points along edges [Med86], inflection points, zeros and discontinuities of curvature [Bri90]. Point features that are loci of two-dimensional intensity changes are named *"corners"* or *second-order features*. Corners impose more constraints on the motion parameters than edges, so that the optical flow field can be fully recovered. Figure 4.1 shows corners detected on image of a human face, where facial features are rounded

and the skin surface is smooth. The detected corners do often correspond to salient facial features such as nostrils, corners of eyes, mouth endpoints, tips of eyebrows. The eyes, mouth, and nostrils can be robustly localized even when the head moves, and the mouth and eyes open and close.

Trackers based on feature points are increasingly used in computer vision applications [Set87], [Hwa89], [Tom94]. However, in a scene where objects move erratically, the noisy image data and spatial and temporal sub-sampling can make motion and acceleration estimation difficult.

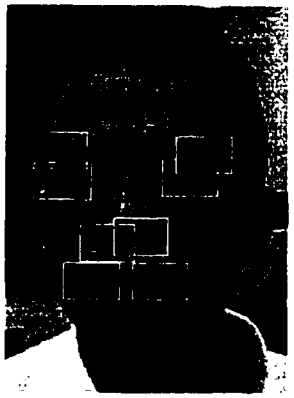


Fig. 4.1: Salient facial features in Marius' face image.

4.1.2. Optical-flow tracking

Optical flow is a computational approximation of the projection of a 3D-velocity field into image plane. Algorithms for recovering optical flow [Hor89] are based on a set of assumptions about the world which, by necessity, are simplifications and hence may be violated in practice. One assumption is the brightness constancy, which is violated when motion boundaries, shadows, or specular reflections are present. Motion boundaries also violate the common assumption that the optical flow varies smoothly. Violations such as these result in gross measurement errors. Moreover, the extraction of the optical flow from an image sequence is a highly computational task.

4.1.3. Correlation-based tracking

Correlation is a popular method for tracking objects [Web81], [Hag96], [Jeb96]. It uses the sum of the absolute differences between template and search area pixel intensities as a difference measure. On the negative side, the correlation tracking methods are sensitive to changes in overall illumination changes between frames of the sequence. If the average intensity of the search area becomes very different from the average intensity of the template then the difference measure will be too high and false matches occur. Moreover, it fails when the tracked features deform or change over time. Simple auto-adaptive templates can adjust to non-rigid object changes, but often they tend to drift off the region of interest without an underlying recursive estimation algorithm.

Visual events such as changes in the background, foreground occlusion, ambient light changes, and unanticipated motion of the camera or the target, concur to produce undetected, irrecoverable failure in many existing visual trackers. Robustness is a characteristic of intelligent behavior and a desirable property of any tracking system. Although there have been attempts to make trackers more robust with respect to certain disturbances [Kos95], [Toy95], little has been done toward developing an effective, general tracking framework for dealing with these problems. Robust, reliable visual tracking of an object in a complex environment will require the integration of several different visual modules, each using a different criterion and each employing different assumptions about the incoming images. The modules must be selected so that when one module fails another can come to aid, or take over.

4.2. 2½D Head Tracking

In order to increase its robustness, our system combines several basic tracking methods in a unified framework used to compute the 2½D motion parameters of a person head.
We have used both the face color and contour matching techniques to track the 2D projection of the head. In order to find head models convenient for tracking, we exploit two geometrical properties: rigidity and symmetry. The inherent rigidity of the head is

quite a convenient property for tracking. The rough symmetry about the vertical axis passing through its center results in an approximate constant 2D-head projection into the image plane. This projection can be modeled by an ellipse with a fixed aspect ratio of 1.2, and a variable orientation, as shown in Figure 4.2.

Fig. 4.2: The 2D elliptical model used for face (head) tracking.

The 2D elliptical head model has four parameters: 2D location (x, y), the size σ, and the orientation γ around z-axis (normal to the image plane). The elliptical model space is defined as a four-dimensional vector $S = (x, y, \sigma, \gamma)$.

On the first image the head is assumed to be at some canonical position. The initial ellipse size σ_0 is computed for the initial frame and stored. Then on subsequent frames, *depth* estimates t_z are recovered by computing the instantaneous size of the model σ_{curr} and comparing it with σ_0:

$$t_z = \frac{\sigma_{curr}}{\sigma_0} \qquad (4.0)$$

The t_z value will get smaller as the target approaches the camera, and larger as it moves away. Thus it provides an estimate of the relative distance of the target from the camera. The recovered 3D position is given by (x, y, t_z), and the planar orientation by γ. The recovered motion parameters are propagated to the *Animation/Rendering* block of the IFAT system, which renders a new posture of the 3D-model (Figure 4.3).

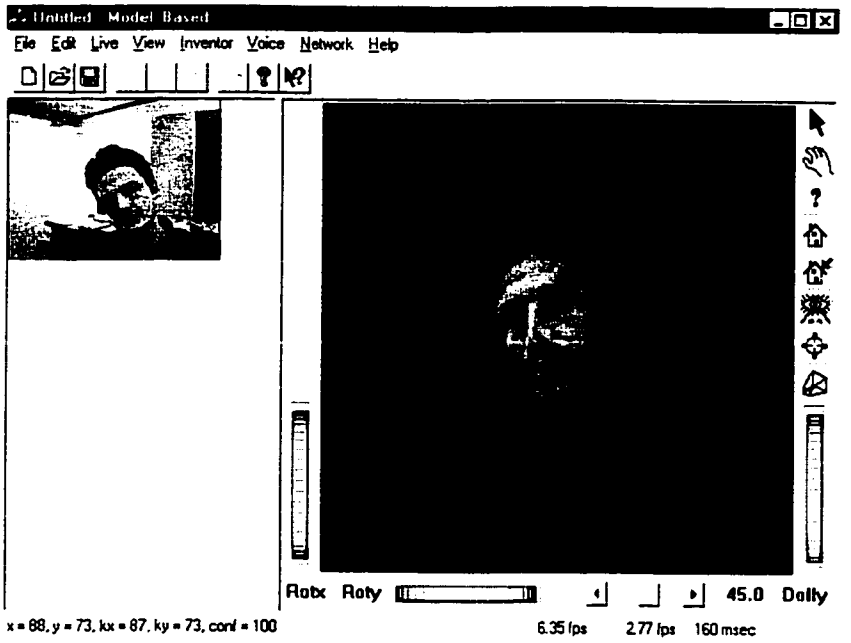


Fig 4.2: Tracking the head rotation about z-axis.

Once the head is detected, an elliptical outline is fitted to the head contour. Every time a new image becomes available, the tracker will try to fit the ellipse model from the previous image in such a way to best approximate the position of the head in the new image. Essentially, tracking consists of an update of the ellipse state $S = (x, y, \sigma, \gamma)$, to provide a best model match for the head in the new image. The state is updated by a hypothesize-and-test procedure [Fie97], [Bir98] in which the goodness of the match is dependent upon the intensity gradients around the object's boundary and the color of the object's interior:

$$S = \arg\max_{S_i \in S} \{\overline{\varphi}_g(\sigma_i, \gamma_i) + \overline{\varphi}_c(\sigma_i, \gamma_i)\} \tag{4.1}$$

where $\overline{\varphi}_g(\sigma_i, \gamma_i), \overline{\varphi}_c(\sigma_i, \gamma_i)$ are the matching scores based on intensity gradients and object color, respectively.

The ellipse's state S is continually updated by local searches combining the output of two "orthogonal" modules of the 2½D tracking system:

(i) *Contour Gradient Module:* dealing with the intensity gradient around the ellipse's perimeter.

(ii) *Face Color Module:* dealing with color information inside the elliptical model ("face-color inside" and "no-face-color outside" the ellipse).

These two modules will complement each other, leading to a robust, real-time system that is able to accurately track a person's head allowing full 360-degree out-of-plane rotation, short severe occlusion, arbitrary camera movement, and multiple moving people in the background. The steps of the tracking algorithm are shown in Figure 4.4.

```
Start the face search at estimated location.

For each pixel of the search area
        For each size of the ellipse
                For each orientation of the ellipse
                        Best fit the elliptical model
                End
        End
End
```

Fig. 4.4: The face search loop.

The state space is built offline to improve the tracker speed, and it is made up of an ellipse array with size and orientation within certain range (Figure 4.5).

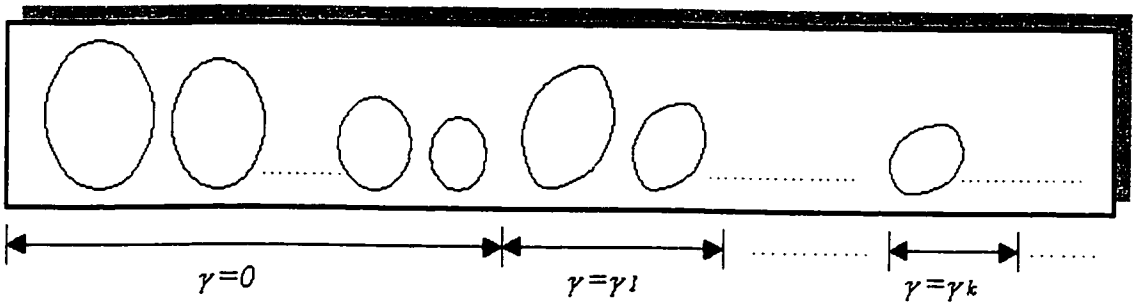


Fig. 4.5: The ellipse's state space

For every new frame the tracker will search along the x and y axes with a span of ±6 pixels, ±1 pixel for ellipse's size, and ±30° for orientation γ. These values can be optimized when using a recursive estimation technique. A *Linear Kalman Filter* (LKF) predicting velocity and acceleration is used to improve the behavior of the tracker because it removes any restriction on the maximum lateral velocity and acceleration of the subject. Each new search starts from the estimated location, size, and orientation based on the LKF prediction.

Adaptive Sub-sampling

In order to improve speed, our tracker employs a coarse-to-fine position-searching algorithm, by sub-sampling the image. Once an image area at location (x, y) is checked, the next area is chosen starting at location $(x+c, y)$. When the end of the scan line has been reached, then the area starting at $(x, y+c)$ is checked. Then the procedure is repeated by setting $c = abs(c/2)$, until $c = 1$. Determining the best sub-sampling constant, c, is important but non-trivial. Sub-sampling affects both the accuracy and the speed of the tracking. If c is too large then the position estimates will be too noisy. If c is too small then too much time will be taken up searching each image area. In our algorithm we start with $c = 3$.

4.2.1. Contour Gradient Module

The "region-of-interest" (ROI) of the input image is preprocessed by firstly extracting the gray information at each ROI pixel:

$$I = \frac{R+G+B}{3}, \qquad (4.2)$$

where I is the intensity and (R,G,B) the pixel color information.
The resulting gray image is smoothed by convolving with a [5x5] Gaussian filter (Figure 4.6a). Then a rough approximation to the gradient magnitude is obtained by convolving the image with a Sobel edge detector filter (Figure 4.6b). We have found that thresholding the gradient (Figure 4.6c), helps to reduce the sensitivity to contrast. Without this threshold, the ellipse model could be attracted to really strong gradients on the background rather than mediocre gradients around the head.

(a) (b) (c)

Fig. 4.6: Preprocessing step of the Gradient Module.

A widely used figure of merit of the goodness of match is the normalized sum of the gradients around the perimeter of the tracking ellipse:

$$\varphi_g(\sigma,\gamma) = \frac{1}{N_{\sigma,\gamma}} \sum_{i=1}^{N_{\sigma,\gamma}} n_{\sigma,\gamma}(i) \cdot g_{\sigma,\gamma}(i) \qquad (4.3)$$

where:

- $n_{\sigma,\gamma}(i)$ is the is the unit vector normal to the ellipse contour at pixel i,
- $g_{\sigma,\gamma}(i)$ is the intensity gradient at perimeter pixel i of the ellipse with state S,
- $N_{\sigma,\gamma}$ is the number of pixels on the perimeter of an ellipse with size σ, and orientation γ.
- $(\cdot)$ denotes the dot product.

The hypothesize-and-test paradigm allows all of the data to be examined before a decision is made.

4.2.2. Face Color Module

The Color Module uses a Mixture of Gaussian (MOG) model to label only the pixels belonging to the ROI (i.e. the face) and then tries to best fit the elliptical model on the "face-labeled" area. The ellipse fitting is done using a normalized log-likelihood maximization technique. The normalized log-likelihood, L, of the color data, $X(t)$, observed from the object at time t is given by:

$$L = \frac{1}{N} \sum_{x \in X_t} \log p(x \mid O), \tag{4.4}$$

where N is the number of pixels in the interior of the ellipse. L is a measure of the color tracker confidence:

$$\varphi_c(\sigma, \gamma) = L, \tag{4.5}$$

Maximizing the log-likelihood is equivalent to minimizing the error: $E = -L$. L is evaluated at each time frame step. A sudden, large drop in the value of L is as a sign that the tracker has lost the face.

The Adaptive MOG Model

Color appearance is often unstable due to changes in both background and foreground lighting. These changes very often lead to a "lost target" event. The color constancy problem has been addressed mainly through the formulation of physics-based models of light formation [For88]. In order to compensate for changes in lighting conditions, we adopt a statistical approach in which color distributions are estimated over time. Under the assumption that lighting conditions change smoothly over time, the MOG model is adapted to reflect the changing appearance of the object being tracked [Bis95], [Mck97]. During successful tracking dynamic adaptation is performed if the normalized log-likelihood $L_t > T$ (fixed threshold), for the frame t. When $L_t \leq T$, a new set of pixels, x, is sampled from the face and used to update the MOG model.

Let $\xi_t = \sum_{x \in X_n} p(j \mid x)$ denote the sum of the posterior probabilities of data in frame t,

where by Bayes rule:

$$p(j \mid x) = \frac{p(x \mid j)w(j)}{p(x \mid O)} \tag{4.6}$$

The parameters are first estimated for each mixture component $\boldsymbol{m}$ (w, μ, Σ), using only the new data, x, from frame t:

$$\begin{aligned} \mu_t &= \frac{\sum p(x \mid j)x}{\xi_t} \\ \Sigma_t &= \frac{\sum p(j \mid x)(x-\mu_{t-1})^T(x-\mu_{t-1})}{\xi_t} \\ w_t &= \frac{\xi_t}{N_t} \end{aligned} \tag{4.7}$$

where N_t denotes the number of pixels in the new data set and all summations are done over the tracked face bounding box. The mixture model components then have their

parameters $m(w, \mu, \Sigma)$, updated using weighted sums of the previous recursive estimates, $m(w_{t-1}, \mu_{t-1}, \Sigma_{t-1})$, estimates based on the new data, $m(w_t, \mu_t, \Sigma_t)$, and estimates based on the old data, $m(w_{t-1-L}, \mu_{t-1-L}, \Sigma_{t-1-L})$.

$$\begin{aligned}
\mu_t &= \mu_{t-1} + \frac{\xi_t}{\Delta_t}(\mu_t - \mu_{t-1}) - \frac{\xi_{t-1-L}}{\Delta_t}(\mu_{t-1-L} - \mu_{t-1}) \\
\Sigma_t &= \Sigma_{t-1} + \frac{\xi_t}{\Delta_t}(\Sigma_t - \Sigma_{t-1}) - \frac{\xi_{t-1-L}}{\Delta_t}(\Sigma_{t-1-L} - \Sigma_{t-1}) \\
w_t &= w_{t-1} + \frac{N_t}{\Omega_t}(w_t - w_{t-1}) - \frac{N_{t-1-L}}{\Omega_t}(w_{t-1-L} - w_{t-1})
\end{aligned} \tag{4.8}$$

where:

$$\begin{aligned}
\Delta_t &= \sum_{i=t-L}^{t} \xi(i) \\
\Omega_t &= \sum_{i=t-L}^{t} N(i)
\end{aligned} \tag{4.9}$$

Adaptation is suspended in case of tracking failure, and restarted when the tracker regains the target. In order to increase the real time performance over, tracking system has the following two notable characteristics:

- The input image will be used entirely only for detection at the starting time or when the target is lost. In order to improve the speed, the tracker uses a ROI limited to the surrounding bounding box of the tracked face. Hence the processing time has greatly decreased, especially for high-resolution images. Even if the tracker confidence drops below a chosen threshold, we still consider a "tracking mode" if the target is re-aquisitioned in the next four frames. The system will automatically switch from current ROI to the whole image rectangle, in order to allow for the search of the lost target within a larger area (Figure 4.7). Another advantage is accuracy. A smaller

search area is less likely to contain other objects similar to the face model, so the tracking process is less likely to be misled.

- The color-tracker confidence is split in two terms, one representing the model's interior color (i.e. the face) ϕ_{int}, and the other the ellipse's exterior color ϕ_{ext}:

$$\varphi_c(\sigma,\gamma) = \phi_{int}(\sigma,\gamma) + \phi_{ext}(\sigma+1,\gamma) = L(\sigma,\gamma) + L(\sigma+1,\gamma) \qquad (4.10)$$

The second term will check the environment color along the perimeter of the next-in-size ellipse. The confidence will increase if we don't find a "face" outside the current ellipse. The "face-inside/non face-outside" technique improves accuracy by allowing the outline contour to inflate up to the displayed face boundary (Figure 4.8). The maximum size of the ellipse is obtained, by searching for "skin" inside, "non-skin" outside. Another and maybe the most important advantage of using an elliptical shape for tracking (and detection) is that skin-like background regions do not attract the tracker.

Fig. 4.7: The switching aspect of the tracking process, in case of a locked target and lost target after a quick move.

Fig. 4.8: Tracker without and with "face-inside/non face-outside" technique.

4.3. Linear Kalman Filter (LKF) for 2½D Tracking

The measurement values obtained by tracking are quite naturally corrupted by noise, resulting in an unstable tracking behavior. The face ellipse will be jumpy and easily lose the locked target. For instance in the case of a sequence with multiple faces the ellipse will jump from one face to another during scene motion, without smooth following the initial target as expected. Localization errors in the face tracking propagate to the recovered pose parameters. When used for synthesis, applying these pose computations to a 3D-head model results in jerky movements, of the animated head. In order to overcome this inconvenience we use an optimal discrete LKF to process the measurements of the tracking parameters for each frame. The continuous linear imaging process is sampled at discrete time intervals by grabbing images at a constant time interval. These images are then sequentially analyzed using a LKF to determine the motion trajectory of the face within a determined error range.

The LKF is a recursive procedure that consists of two stages: time updates (or prediction) and measurement updates (or correction). At each iteration, the filter provides an optimal estimate of the current state using the current input measurement, and produces an estimate of the future state using the underlying state model.The values, which we want to smooth and predict independently, are the tracker state parameters. The tracker will employ a LKF as a recursive motion prediction tool, for the recovery of the 2½D head pose parameters.

4.3.1. LKF Basics

The LKF estimates the state $s \in R^n$ of a discrete-time process that is governed by the linear first-order difference equation:

$$s(k+1) = A(k)s(k) + w(k) \qquad (4.11)$$

This process is sampled and quantified at discrete time intervals k, producing the set of experimental measurement data:

$$m(k) = H(k)s(k) + v(k) \qquad (4.12)$$

where:

- $w(k)$ and $v(k)$ are the random variables that represent the process and measurement noise. We assumed them to be white noises, independent and having normal probability distributions N:

$$p(w) \sim N(0,Q) \qquad (4.13)$$

$$p(v) \sim N(0,R) \qquad (4.14)$$

- The n-by-n matrix A is the state transition matrix.
- The m-by-n matrix H in the measurement transition matrix.
- n is the number of state parameters, and m is the number of measurements.
- Q and R are covariance matrices.

The filter's task is to provide a "best" linear estimate (filtered value) $\hat{s}(k)$, and a "best" predicted value $\hat{s}(k \mid k-1)$ of the state $s(k)$. "Best" means estimators that minimize the mean square error of each state component simultaneously.

$$E[s_i(k) - \hat{s}_i(k)]^2, \qquad i = 1...n \qquad (4.15)$$

4.3.2. Filtering and Prediction

The equations for the LKF could be separated in two groups: *time update* equations and *measurement update* equations.

- *the time update* equations are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the *a priori* estimates for the next time step.
- *the measurement update* equations are responsible for the feedback, i.e. for incorporating a new measurement into the *a priori* estimate to obtain an improved *a posteriori* estimate.

The *time update* equations can also be thought of as *predictor* equations, while the *measurement update* equations can be thought of as *corrector* equations. Indeed the final estimation algorithm resembles that of a *predictor-corrector* algorithm for solving numerical problems as shown below in Figure 4.9. The LKF has ability to adjust its own parameters according to the current confidence in the accuracy of the state parameters.

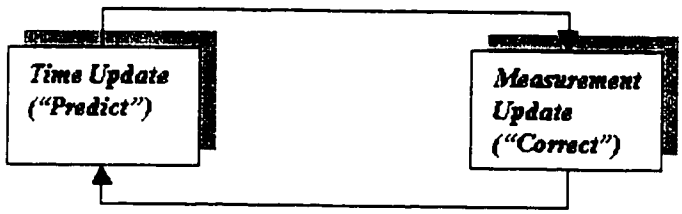


Fig. 4.9: The Kalman Filter Loop.

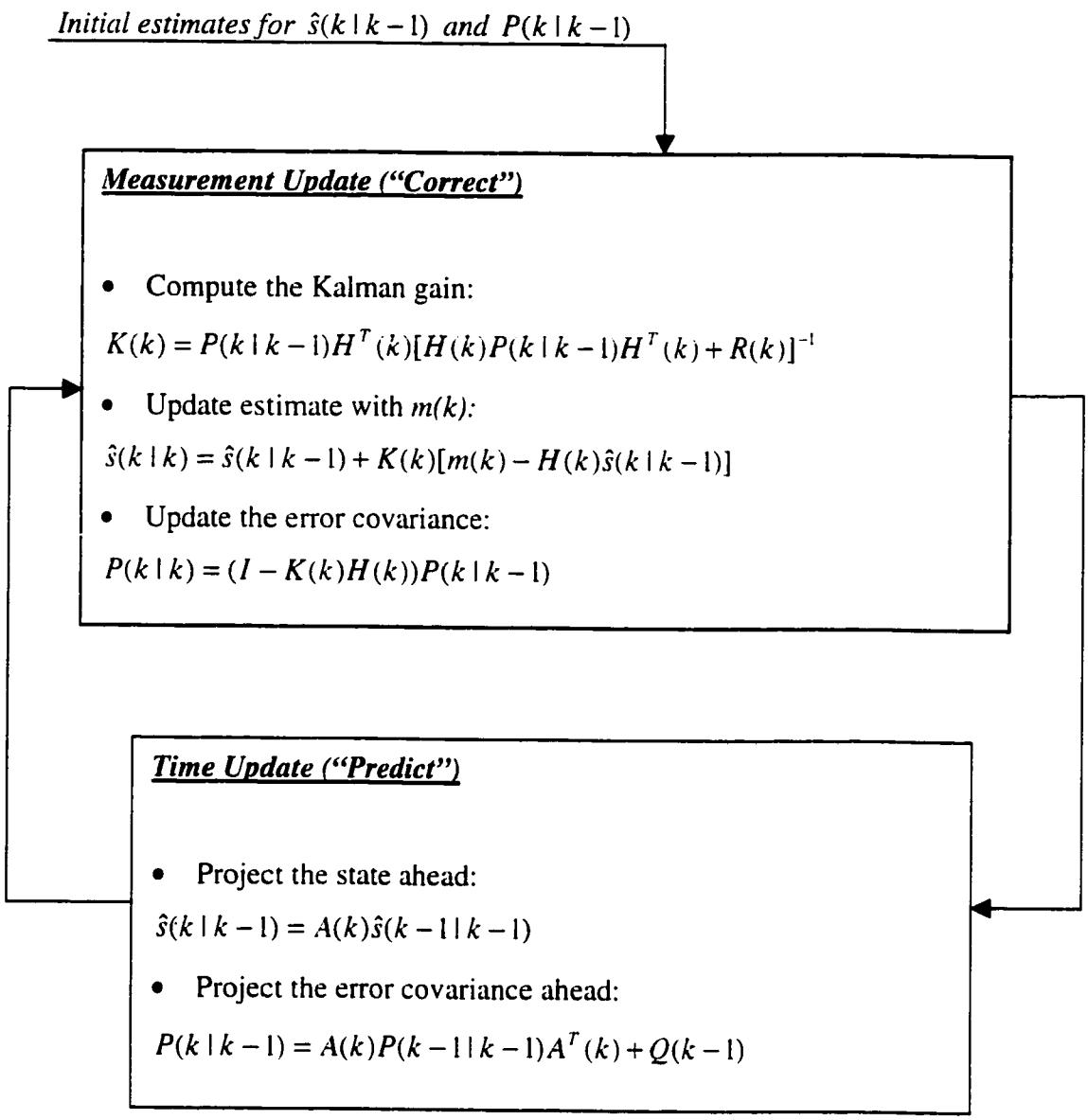


Fig. 4.10: Update Equations of the Linear Kalman Filter

The first *measurement update equation* computes the *Kalman gain* (*blending factor*), $K(k)$ (*n-by-m* matrix), which determines the ratio of the error between predicted and measured parameters that will be used to update the state vector.

The second *measurement update equation* uses the measured samples $m(k)$, to generate the *a posteriori* state estimate $\hat{s}(k \mid k)$.

The $m(k) - H(k)\hat{s}(k \mid k-1)$ term, called *residual*, shows the difference between the predicted measurement $H(k)\hat{s}(k \mid k-1)$ and the actual measurement $m(k)$. A residual of zero means that the two are in complete agreement.

In the third *measurement update equation* the portion of the gain matrix that is associated with elements obtained from the image (i.e., face model position and velocity) is subtracted from an identity matrix I to yield a proportion of non-gain. This is multiplied with the estimated error covariance to produce an updated error covariance that reflects the remaining uncertainty about the state parameters.

The prediction of the state parameter values and error at the next time instance is achieved by the *time update equation* equations. The motion model $A(k)$ matrix allows new estimates to be derived by applying an appropriate function to the current parameters and error, as discussed further in next section. We are using a linear motion model, reflecting the behavior of an object moving at constant velocity.

The $H(k)$ matrix is used to convert the state parameters and their associated errors to the same basis as the measurement data. In our case, for images captured orthogonal to the direction of face movement, the state parameters are held in the same coordinate system as the measurement data, affording a one-to-one correspondence. Therefore $H(k)$ remains constant throughout, which allows the extraction of only those elements of the parameters or error that equate to measurements from the image. If a perspective transformation is needed to align the image sequences with the state parameters of the model, $H(k)$ should be updated at each iteration of the Kalman filter loop. This reflects the non-linear changes between the image measurement and the state parameters.

4.3.3. Motion and Measurement Models

LKF techniques can be used to track objects robustly from measurements of position, motion and shape. The way the problem is formulated depends on the measurements that

can be made and the results that are required. For instance, if one is interested in positional coordinates, scale and orientation, each of these can have a value and a first and second derivative with respect to time. This could be formulated as a single twelve dimensional state vector $s = (x_c, y_c, \sigma, \theta, \dot{x}_c, \dot{y}_c, \dot{\sigma}, \dot{\theta}, \ddot{x}_c, \ddot{y}_c, \ddot{\sigma}, \ddot{\theta})$ with a corresponding state transition matrix $A(k)$. Ideally, we want to smooth and predict the entire ellipse state vector, including its position, scale and orientation. The most important element of the ellipse state vector represents its position, since each search starts at the estimated face location. Estimating scale and orientation could improve the speed of the tracking algorithm, but the tracking success (correctness and speed) depends most on the predicted search location. Hence, we chose the state vector to model only the motion of the tracker position in x-y plane,

$$s = (x_c, y_c, \dot{x}_c, \dot{y}_c, \ddot{x}_c, \ddot{y}_c).$$

Our dynamic system describes both coordinates of the center of the tracker by its position, velocity and acceleration. Using Newton dynamics, the state equation (4.11) becomes:

$$\begin{pmatrix} x_c \\ y_c \\ \dot{x}_c \\ \dot{y}_c \\ \ddot{x}_c \\ \ddot{y}_c \end{pmatrix}_{k+1} = \begin{pmatrix} 1 & 0 & \Delta\tau & 0 & \frac{\Delta t^2}{2} & 0 \\ 0 & 1 & 0 & \Delta t & 0 & \frac{\Delta t^2}{2} \\ 0 & 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_c \\ y_c \\ \dot{x}_c \\ \dot{y}_c \\ \ddot{x}_c \\ \ddot{y}_c \end{pmatrix}_k + w(k) \qquad (4.16)$$

This state transition equation represents the familiar equation of motion of a body moving with constant acceleration. The measurement model of the ellipse center (x_c, y_c) is the measurement vector $m_k(x_c, y_c)$ as per (4.12). The measurement transition

matrix $H(k)$ is chosen to enable conversion of the state parameters to the same basis as the measurement data. Since only position data is extracted $H(k) = [I,0]^T$, where I is the identity matrix. This removes the velocity values from the state vector to facilitate comparison with the measurements of position.

LKF Initialization

Setting initial parameters of these two models is rather crucial for the LKF performance. The LKF must be initialized prior to the recursive update stage.
The $R(k)$ matrix is initialized with the covariance of the estimated measurement error. It is assumed that the error of the center position of the tracker is Gaussian distributed with a standard error equivalent to one pixel displacement

The initial face tracker position initializes the state vector prediction $\hat{s}(k \mid k-1)$ used as input in the *measurement update module* (Figure 4.10). The initialization of the error covariance matrix $P(k \mid k-1)$ is crucial for the update stage, dictating how quickly the algorithm will converge on parameters within an acceptable error. Brown et al. [Brn83] suggest if there is no initial predicted estimate, $P(k \mid k-1)$ should be initialized with the covariance of the first measurement. Another initialization technique [Hos95] uses the error from the first prediction and the associated measurement, after processing three images. Face tracker positions in the first two images are located then from this information forward prediction is made. This prediction is compared with the measured position in the third image to determine an initial error value. It is important to initialize $P(k \mid k-1)$ to values high enough to allow influence from measurements taken from the early images in the sequence so that the filter can self adjust according to the input data. More on description and analysis on tuning the LKF can be found in

[Pal94]. The actual inter-frame time is continuously measured and used as Δt in equation (4.16).

LKF Update

The update stage is an iterative process generating estimates of the next state parameters (position, velocity, and acceleration) and the associated error covariance matrix.

Figure 4.11 illustrates the structure of a tracking algorithm, which integrates a LKF.

REPEAT

Step1 **Start** the target search in image based on filter estimate.

Step2 **Locate** the target in image using the tracking algorithm.

Step3 **Use** located position in Kalman filter ($m(k)$ measurement vector).

UNTIL the end of the image sequence.

Fig. 4.11: Kalman filter tracking algorithm.

Since the search area should be as small as possible, yet be large enough to accommodate face motion variations, the size of this area is adjusted according to the most recent deviation between the real and predicted trajectories. The search area is centered at the estimated face location. We start the face search within an area of $[(wt/30)\times(ht/30)]$, where *wt* and *ht* are the width, respectively the height of the captured image. The search window boundaries can be adjusted using the error covariance [Hos95], or the difference "predicted-measured" position values. When the deviation of the "predicted-measured" position is small, the search area is only a little bit larger than the initial area. As the deviation increases, the expansion factors modify the search window and the ROI containing the face. The image geometry parameters mentioned in Figure 4.12 are:

$$\Delta x_c = \sqrt{P(k \mid k-1)[x_{err}]} \qquad (4.17)$$

$$\Delta y_c = \sqrt{P(k \mid k-1)[y_{err}]}$$

$$x_{low} = \hat{x}_c(k \mid k-1) - \Delta x_c$$

$$x_{high} = \hat{x}_c(k \mid k-1) + \Delta x_c \qquad (4.18)$$

$$y_{low} = \hat{y}_c(k \mid k-1) - \Delta y_c$$

$$y_{high} = \hat{y}_c(k \mid k-1) + \Delta y_c$$

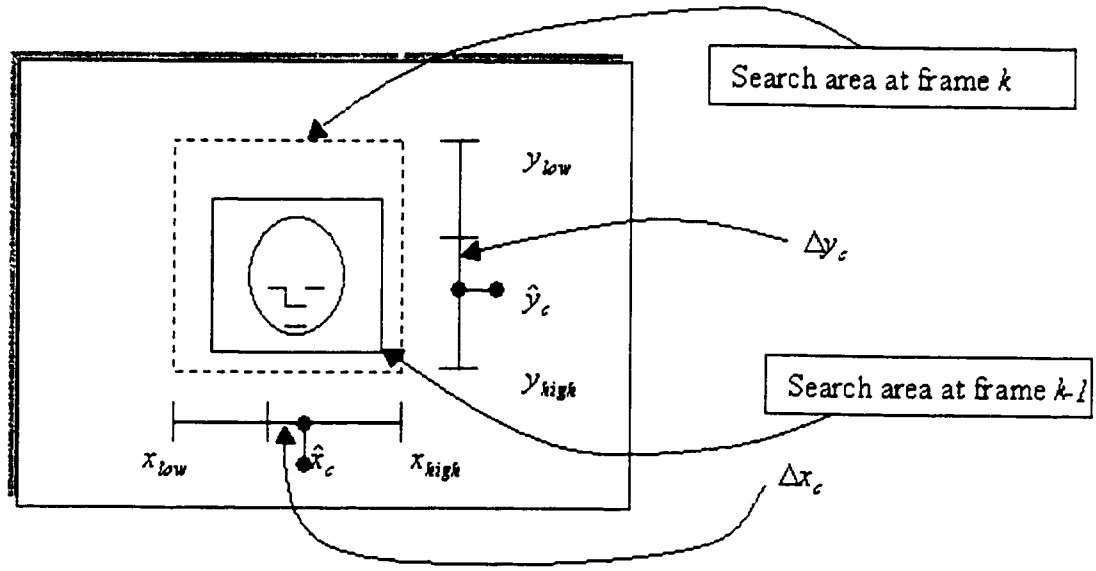


Fig. 4.12: Adjusting the search area

Figure 4.13 shows the estimated vs. measured position of the elliptical tracker and its confidence, for a sequence of 200 frames, containing a moving face. The face is detected in first 5 frames, then successfully tracked in next frames.

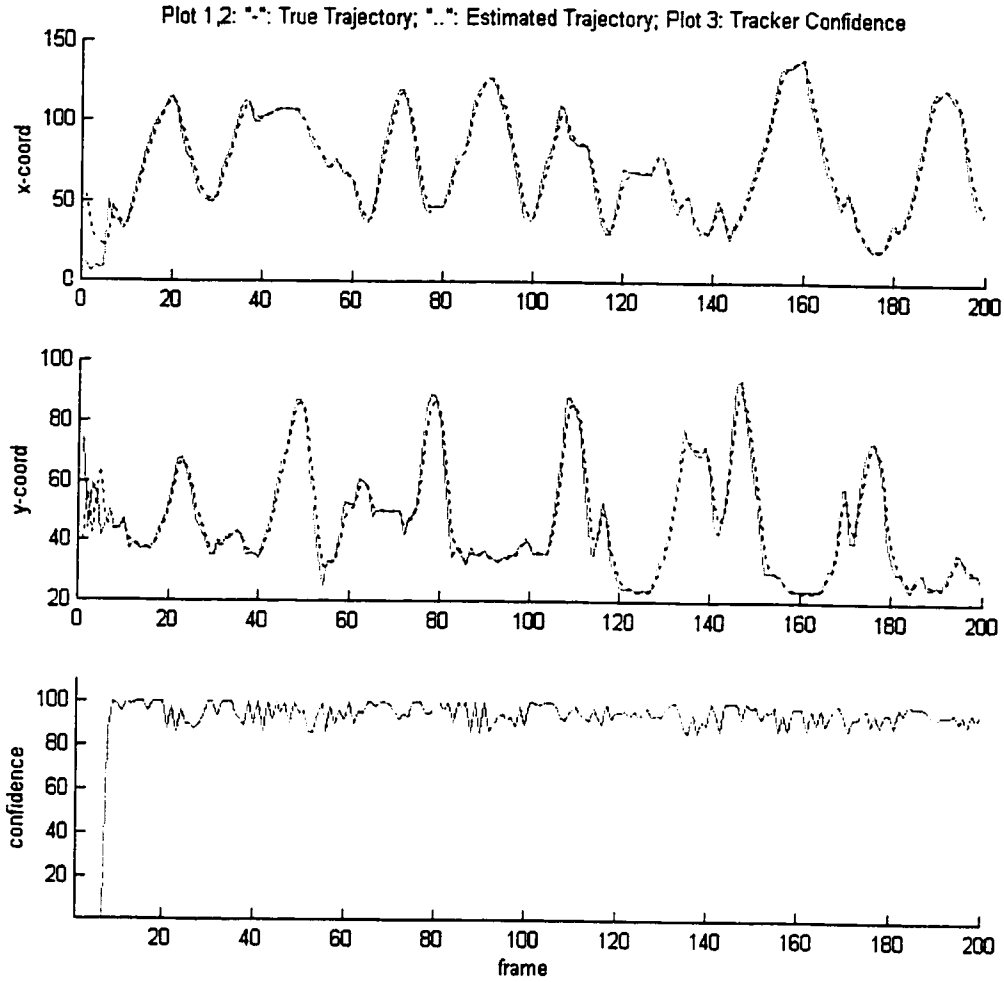


Fig. 4.13: Tracking a moving face in an image sequence.

Chapter 5

Tracking for 3D Head Pose Recovery

Inferring 3D object characteristics from 2D measurements is a complex task, which is far from being resolved. The general problem of recovering 3D position parameters from 2D images could be solved using different 2D views of the 3D objects. If these 2D images are taken at the same time the problem is solved by *stereovision* [Fau92], [Har92], or *trifocal tensor* [Shs95], [Har94]. Another approach [Jeb99] using monocular 2D images of moving objects is known as *Structure-From-Motion* (SFM). Following we give a short SFM background, and then present our SFM implementation of the 3D head tracking task.

5.1. Structure-From-Motion (SFM)

Given 2D-object images the SFM problem aims to recover:

- the 3D object coordinates
- the relative 3D camera- object motion
- camera geometry (camera calibration)

There are many papers reporting on different SFM applications:

- *3D Model Reconstruction,* where 3D object-coordinates, thus 3D models are recovered from 2D images [Deb96], [Jeb99]. In this case SFM is an alternative to well-known methods such as 3D laser scanning, structured light, depth from shading etc.
- *3D Motion Recovery,* where 3D motion parameters are recovered and used to drive 3D characters/avatars in virtual environments. These 3D avatars will mimic live actions of the performing actors. Azarbayejani [Aza93] *et al.* present a visually controlled framework where virtual 3D face models are animated from a camera-

based head tracking. Kutulakos [Kut98] presents an augmented virtual reality environment where 3D objects are super-imposed (enhanced/augmented reality) in the real scene in real time. Such techniques are currently being integrated into standard computer graphics software for use in film, video, games, interactive media, industrial design and visualization.

- *Camera calibration* that recovers the external and internal camera geometric parameters. The external parameters represent the camera position in real world coordinates and internal parameters include camera variables such as focal length. Park *et al.* [Par94] describe a method for estimating camera parameters from video sequences for applications to model-based video systems, using feature correspondence and a least-square estimator.

5.1.1. Perspective Camera Model

Most of SFM techniques start by assuming a perspective projection model (Figure 5.1), often referred as a pinhole camera, which reflects the natural process of the image acquisition.

The 3D feature points are projected in the image plane through the projection rays rooted at the *center of projection (COP)*, located inside the physical camera. The origin of camera coordinate system often is chosen as *COP*. The focal length f is the distance from *COP* to the image plane along the *principal axis* (or *optical axis*). The *principal axis* is traditionally aligned with the z-axis. The projection of the *COP* along the *principal axis* is called the *principal point* $p_0(u_0, v_0)$. Considering $p_0(u_0, v_0)$ as the center of the image plane, and applying Thales theorem we obtain the perspective projection equation.

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{f}{Z}\begin{pmatrix} X \\ Y \end{pmatrix} \qquad (5.1)$$

where $p(X, Y, Z)$ is a 3D object feature point represented in camera coordinate reference frame.

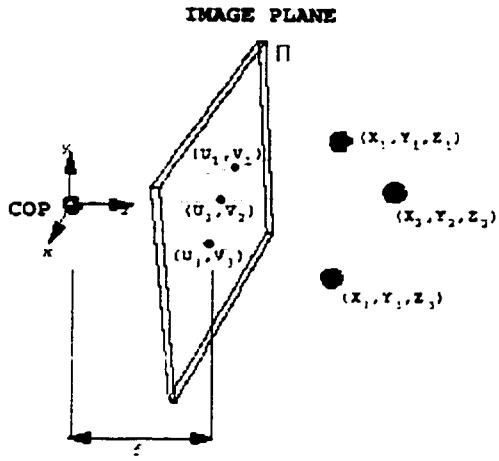


Fig. 5.1: Perspective camera model [Jeb99].

This simplified model taking into account only the most important internal parameter (*focal length*) is often sufficient in machine vision application modeling.

The translation motion is modeled as a 3D position of the object reference frame relative to the current camera reference frame:

$$T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}, \tag{5.2}$$

with t_x, t_y components corresponding to directions parallel to image plane, and t_z the depth of the object, respectively.

The rotation is modeled as a composite matrix obtained by the multiplication of the three elementary rotation-matrices accounting for the pitch, yaw, and roll of the object frame.

$$R = R_\alpha R_\beta R_\gamma = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}, \tag{5.3}$$

where α, β, γ (pitch, yaw, and roll respectively) denote the three rotation angles (Euler angles) of the object about the x, y, and z-axis, respectively. The rotation matrices for the Euler angles are:

$$R_\alpha = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix},$$

$$R_\beta = \begin{pmatrix} \cos\beta & 0 & -\sin\beta \\ 0 & 1 & 0 \\ \sin\beta & 0 & \cos\beta \end{pmatrix}, \qquad (5.4)$$

$$R_\gamma = \begin{pmatrix} \cos\gamma & \sin\gamma & 0 \\ -\sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

Multiplying these matrices finally yields the elements of the composite rotation matrix R (5.3):

$$\begin{aligned}
r_{11} &= \cos\gamma\cos\beta, \\
r_{12} &= \sin\gamma\cos\beta, \\
r_{13} &= -\sin\beta, \\
r_{21} &= -\sin\gamma\cos\alpha + \cos\gamma\sin\beta\sin\alpha, \\
r_{22} &= \cos\gamma\cos\alpha + \sin\alpha\sin\beta\sin\gamma, \\
r_{23} &= \sin\alpha\sin\gamma + \cos\gamma\sin\beta\cos\alpha, \\
r_{31} &= \sin\gamma\sin\alpha + \cos\gamma\sin\beta\cos\alpha, \\
r_{32} &= -\cos\gamma\sin\alpha + \sin\gamma\sin\beta\cos\alpha, \\
r_{33} &= \cos\beta\cos\alpha.
\end{aligned} \qquad (5.5)$$

5.1.2. Solving the SFM problem

The SFM problem assumes no prior knowledge about the 3D model and motion, and camera calibration. SFM aims to recover these 3D parameters from 2D observations over a sequence of images. The SFM framework consists of two main modules (Figure 5.2):

- *Tracking module*, delivering the 2D point measurements $p_i(u_i, v_i)$ of the tracked features, where $i=1,...,m$, and m is the number of measurement points.
- *Estimator module* (for the estimation of 3D geometry and motion), delivering a state vector

$$s = (t_x, t_y, t_z, \alpha, \beta, \lambda, f, X_i, Y_i, Z_i) \qquad (5.6)$$

where:

- $(t_x, t_y, t_z, \alpha, \beta, \lambda)$ are the six 3D camera/object relative motion, namely translation and rotation;
- f is the camera focal length;
- $P_i(X_i, Y_i, Z_i)$ is the object geometry, where $i=1,...,m$, and m is the number of tracked features.

The SFM framework represents rigid motion recovery function of the Performance-Driven block (Figure 2.5). The recovered motion parameters are propagated to the Animation/Rendering module to animate the 3D-head model as described in Chapter 2.

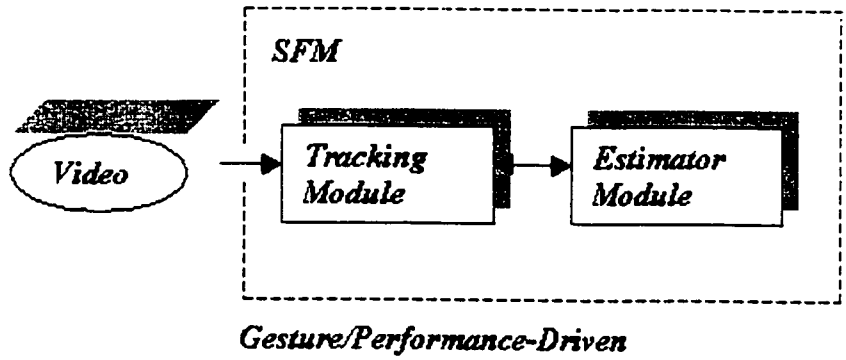


Fig. 5.2: The SFM component of the Performance-Driven block of our IFAT system.

There are two known matching techniques, *feature-based* and *optical flow*, which SFM can use to infer the 3D information from 2D observations.

As discussed in Chapter 4, features are object components that can easily be identified: edges, corners or other outstanding points.

The *Feature-based* tracking first determine such locations in two frames and then match features between frames. This technique consists of three steps:

(i) *Extraction* of a set of salient two-dimensional features from a set of monocular frames *(implemented in the Tracking Module).*

(ii) *Tracking* the found features from one frame to another prior to motion computation, problem recognized as very difficult. Some constraints such as rigidity are exploited in matching algorithms *(implemented in the Tracking Module).*

(iii) *Motion Estimation* is the last step of this approach. Also, the task is not trivial since the relation between the projections and 3D scene objects is nonlinear and the input data is always corrupted by noise. The reconstruction of the objects characteristics practically requests an available *a priori* information (a 3D virtual model of a real object, in our case, the 3D-head model) *(implemented in the Estimator Module).*

The *Optical flow* technique can be divided into two steps:

(i) *Computation* of the apparent velocity field (or so-called optical flow) from the temporal changes in gray levels in the image plane *(implemented in the Tracking Module).*

(ii) *Inferring* the motion parameters and structure of objects in 3D space from optical flow by using additional constraints *(implemented in the Estimator Module).*

As reported in the literature [Hee92], [Zha92], optical flow techniques are unstable and unreliable since the derivatives enhance the noise level, and the real velocity will be different from the apparent velocity. Feature-matching and optical flow are generally computed from two sequential frames in an image sequence. Since both algorithms introduce errors, the information extracted from only two frames is not very accurate. Applying a two-frame algorithm to sequential pairs of images does not solve the noise reduction problem. While it can be theoretically possible to combat noise by using long sequence analysis, this is practically difficult because of the delay it introduces between the real and virtual world motions.

There are two ways in which multiple frame measurements can be integrated in a SFM framework:

(i) *Global* or *Batch* methods, where data from a finite number of n frames $(n > 2)$ is collected and used as an input for the SFM algorithm. While this approach produces accurate result for the given set of frames, it requires storage of large amounts of data and doesn't produce any result until all frames have been processed.

(ii) *Recursive* or *Iterative* methods, where a current estimate of the quantities of interest is maintained after every acquired frame and updated each time a new frame becomes available. In this case accuracy and convergence/stability may become an issue for the SFM algorithm. However, because no expensive data storage is required and an estimate is available at every time instant, these methods are well suited for real-time applications.

Due to the continuous nature of visual information acquisition and the large amounts of data involved, we decided to use a recursive estimation technique.

Due to the perspective camera model, SFM is a nonlinear problem. Attempts were made to reduce the inherent complexity of solving it linearly. A popular technique is the factorization method proposed by Tomasi and Kanade [Tom92]. They are using an orthographic camera model, which neglects the camera focal length and simplifies the problem by setting $u = X$, and $v = Y$ (Figure 5.3). They assume a beforehand known image sequence. The feature points are fed in a "*features* x *frames*" matrix, then the 3D parameters are extracted using singular value decomposition and linear operators. Unfortunately, the "batch" technique and the orthographic camera model prohibit the application of this method to real life scenarios.

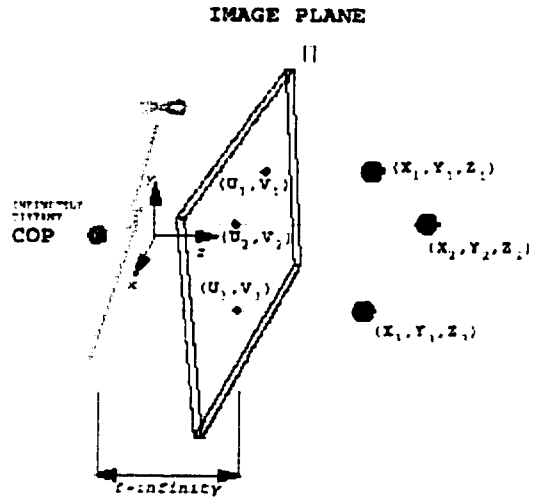


Fig. 5.3: Orthographic camera model.

Many nonlinear frameworks can be related to the classic *Relative Orientation* problem proposed by Horn [Hor86], [Hor90], using the perspective projection camera model. Structure and motion are recovered by minimizing a nonlinear cost function for a two-frame recursive technique. The solution doesn't recover the camera internal geometry. Lowe offers a solution [Low87] to the SFM problem, starting with a certain guess (initial solution) for the unknown scene parameters, including the six pose parameters, possibly

some intrinsic camera parameters such as the focal length *f*. Using a perspective camera model, the algorithm re-project the *a priori* known 3D scene model into the image plane, and then compute, a vector of errors *d* between the positions, orientations and apparent angles of the re-projected features, and the corresponding actual measurements. In [Kum89], [Sze93], and [Wen93] the SFM solution is found through the Levenberg-Marquardt nonlinear minimization algorithm.

5.2. Nonlinear Recursive SFM Solution

The problem can be formulated as a parameter estimation problem:
"Given a number of noisy measurements of 2D-tracker positions, we have to optimally recover the SFM components of equation (5.6)". Least-squares techniques have been popular to solve such type of problems, in computer vision. Unfortunately, these techniques need a priori knowledge of the 3D object models and are not able to handle gross and systematic errors, and correlation in measurements.
These unrealistic assumptions prevent least-squares methods from converging to an acceptable solution. A robust alternative to the least square methods is the Extended Kalman Filter (EKF), for the following reasons [Zha92]:

(i) EKF explicitly considers the observation uncertainties.
(ii) EKF is fed with measurement recursively.
(iii) EKF is a simple and robust solution to parameter estimation problems.

The literature on Kalman Filter-based recursive algorithms for recovering 3D structure and/or motion from a monocular sequence of images dates back to the early eighties [Gen82]. Extensive work has been done on three main areas of the problem:

(i) recovery of motion with known structure [Aza93], [Bro86], [Gen82], [Gen92]
(ii) recovery of structure with known motion [Mat89], [She92],
(iii) recovery of motion and structure simultaneously [Aza95], [Jeb96], [Bro91], [Hee90], [Hee91].

Gennery [Gen82], [Gen92] considers the problem of tracking 3D objects of known shape, and recovering object motion and position with respect to the observer. His state vector therefore consists of 12 parameters (6 configuration and 6 velocity). For tracking he uses both point and line features.

Matthies et al. [Mat89] make the assumption of known motion, and demonstrate their scheme in the special case of translational motion both with a feature-based scene representation and with a dense depth-map. Recently Shekhar and Chellappa have successfully tested a similar approach, again assuming known motion [She92].

Heel [Hee90], [Hee91] proposed a filter formulation for the recovery of both structure and motion, with further regularization of the depth map. The formulation is similar to the one by Matthies et al. [Mat89], but the motion is now estimated at each step with a least-squares procedure from the estimated structure. The best estimate of motion is used to produce an estimate for structure, which is used to obtain the new estimate of motion. In Heel's algorithm the motion parameters are not included in the state of the filter, hence motion temporal coherence cannot be exploited.

We have adapted the SFM approach of Azarbayejani and Pentland [Aza95] to recursively recover the 3D structure, 3D motion and perspective camera geometry from feature correspondences over a sequence of 2D images. To speed up the calculations we are using a motion model that simplifies the Jacobian. EKF is used to solve the SFM problem resulting in an accurate, stable and real time solution. This EKF takes in consideration the non-linear aspect of mapping. We use a perspective camera model to reflect the mapping between the 3D world and its projection. In the next sections we present three EFK based techniques, used to recover motion (5.3.6 and 5.3.7), and structure and motion (5.3.8). All methods also recover the camera focal length.

5.2.1. EFK Basics

As described in Chapter 4 the Linear Kalman Filter (LKF) addresses the general problem of estimating the state of a discrete-time controlled process governed by a *linear* stochastic difference-equation. When the process to be estimated is non-linear, one has to use a non-linear estimation technique (EKF) to solve the SFM problem. Essentially the EFK is applied to nonlinear systems with additive white noise. The EFK continually updates a linearization around the previous state estimate, starting with an initial guess. The EFK linearizes about the current mean and covariance values. The degree of accuracy of this linearization depends on the initial guess and the evolution of disturbances. The formalism assumes that the system-noise and measurement-noise processes are uncorrelated and both are Gaussian white-noise sequences. Similarly to a Taylor series, the filter linearizes the estimation around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the case of non-linear relationships.

A non-linear dynamic system can be represented by the following equations:

$$s(k+1) = f(s(k), w(k)) \tag{5.3}$$

$$m(k) = h(s(k), v(k)) \tag{5.4}$$

where:

- $w(k)$ and $v(k)$ are the random variables that represent the process and measurement noise. We assumed them to be independent of each other, white, and with normal probability distributions.

  $$p(w) \sim N(0, Q) \tag{5.5}$$

  $$p(v) \sim N(0, R) \tag{5.6}$$

- the *non-linear* function $f(\bullet)$ in the difference equation (5.3) relates the state at time step k to the state at step $k+1$. It includes as parameter the zero-mean process noise $w(k)$

- the *non-linear* function $h(\bullet)$ in the measurement equation (5.4) relates the state $s(k)$ to the measurement $m(k)$.

After linearization, the state and measurement equations (5.3) and (5.4) can be expressed as [Wel98]:

$$s(k+1) = As(k) + \xi(k) \qquad (5.7)$$

$$m(k) = Hs(k) + \eta(k) \qquad (5.8)$$

where:

- A is the Jacobian matrix of partial derivatives of $f(\bullet)$ with respect to s
- H is the Jacobian matrix of partial derivatives of $h(\bullet)$ with respect to s
- $\xi(k)$ and $\eta(k)$ are independent random variables having zero mean and covariance matrices: WQW^T and VRV^T, with Q and R from (5.5) and (5.6). Here, W is the Jacobian matrix of partial derivatives of $f(\bullet)$ with respect to $w(k)$, and V is the Jacobian matrix of partial derivatives of $h(\bullet)$ with respect to $v(k)$.

The observations are the 2D feature coordinates (u,v) which are concatenated into a measurement vector $m(k)$ at each time step. The observation vector is the back-projection of the s state vector containing the scene 3D structure, the relative 3D camera-scene motion, and the camera internal geometry, namely the focal length.
The s to m mapping is nonlinear ($H(s)$ varies with s), and corrupted by some noise encoded in the time-varying covariance matrix $R(k)$. Large variances $R(k)$ represent lost 2D features during the tracking process. The system evolution matrix A is based on first order Newtonian dynamics in 3D and assumed time invariant. The 3D structure, motion and the focal length are constrained through the A matrix, which sets a linear dependency between the past and current values. The internal state contains a random Gaussian noise $\xi(k)$, representing the variations of the internal state.

The recursive approach captures both the cause-effect and the dynamic nature of the SFM problem, offering also a probabilistic framework for uncertainty representation. In this chapter we present a solution to the SFM problem allowing recovering the 3D motion of the user head, and the focal length of the camera. The state vector $s(translation, rotation, velocity, focal\_length)$ contains the relative 3D camera-object translation, rotation and their velocities, and camera focal length. The $m(u_1, v_1, u_2, v_2, ..., u_N, v_N)$ measurement vector is the concatenation of the 2D feature screen coordinates.

5.2.2. Filtering and Prediction

The "filtering and prediction" relates to the state estimate, where $\hat{s}(k \mid k)$ is the state estimate after a measurement ("filtered estimate") and $\hat{s}(k \mid k-1)$ is the state estimate after a time update ("predicted estimate").

The EKF recursively estimates the state vector at each frame using the measurement vector $m(k)$, the state prediction $\hat{s}(k \mid k-1)$ and the state prediction error covariance matrix $P(k \mid k\text{-}1)$. The basic operation of the EKF is similar to the discrete LKF (Figure 4.9). In this case, since we use a perspective transformation, $H(k)$ is updated at each iteration of the EKF loop in order to reflect the nonlinear mapping between the state parameters and the image measurement. At the time step k the computational EKF cycle proceeds as shown in Figure 5.4:

Step 1. **Input** a new measurement vector $m(k)$.

Step 2. **Update** the measurement-system Jacobian matrix:

$$H(k) = \left(\frac{\partial h(k)}{\partial s} \right)_{s=\hat{s}(k|k-1)}$$

Step 3. **Compute** the system error covariance matrix:

$$P(k \mid k-1) = A(k)P(k-1 \mid k-1)A^T(k) + W(k-1)Q(k-1)W^T(k-1)$$

Step 4. **Compute** the gain matrix:

$$K(k) = P(k \mid k-1)H^T(k)[H(k)P(k \mid k-1)H^T(k) + V(k)R(k)V^T(k)]^{-1}$$

Step 5. **Update** the system error covariance matrix:

$$P(k \mid k) = (I - K(k)H(k))P(k \mid k-1)$$

Step 6. **Predict** the state vector:

$$\hat{s}(k \mid k-1) = f(\hat{s}(k-1 \mid k-1),0)$$

Step 7. **Update** the state vector:

$$\hat{s}(k \mid k) = \hat{s}(k \mid k-1) + K(k)[m(k) - h(\hat{s}(k \mid k-1),0)]$$

Step 8. **Store** for the next cycle the updated state vector and system error covariance matrix:

$$\hat{s}(k \mid k) \xrightarrow{step+} \hat{s}(k-1 \mid k-1)$$

$$P(k \mid k) \xrightarrow{step+} P(k-1 \mid k-1)$$

Fig. 5.4: The EKF algorithm steps.

EFK equations suggest that the filter behavior agree with our human intuition. The gain matrix can be expressed [Zha92] in the form:

$$K(k) = P(k \mid k)H^T(k)R^{-1}(k), \qquad (5.9)$$

Hence, the Kalman gain is "proportional" to the uncertainty in the estimate and "inversely proportional" to the measurement uncertainty. Having defined the residual as $r = m(k) - h(\hat{s}(k \mid k-1),0)$, the expression in (5.9) reflects two scenarios:

- very uncertain *measurement* and relatively precise *state estimate*. In this case the residual r is generated by the noise and the state estimate needs little change.
- relatively precise *measurement* and very uncertain *state estimate*. In this case the residual r mirrors the state estimate errors and the state estimate needs a strong correction.

Inverting $P(k|k)$ and replacing $K(k)$ in (5.9), we obtain the covariance matrix equation:

$$P^{-1}(k \mid k) = P^{-1}(k \mid k-1) + H^T(k)R^{-1}(k)H(k), \qquad (5.10)$$

We notice the "proportionality" between P and R covariance matrices. It also can be seen that a very precise measurement ($R(k)$ is small) will considerably reduce the estimation error by decreasing the estimation variance.

5.2.3. Motion and Measurement Models

The EKF requires a physical dynamic model of the motion and a measurement model relating image feature locations to motion parameters. Additionally, because this approach recovers the motion without structure, a representation of the object (user's head) is required. In this section we present the chosen motion and measurement models and their resultant equations (5.11) and (5.16).

The Motion Model

The dynamic model is a discrete-time Newtonian physical model of a rigid body motion. moving with constant velocity. Our state vector $s(t_x, t_y, t_z, \omega_x, \omega_y, \omega_z, f, \dot{t}_x, \dot{t}_y, \dot{t}_z, \dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z)$ consists of 13 elements grouped as follows: the relative camera-object translation (t_x, t_y, t_z), the small inter-frame rotation

$(\omega_x, \omega_y, \omega_z)$, the camera focal length f, the translational velocity $(\dot{t}_x, \dot{t}_y, \dot{t}_z)$, and the rotational velocity $(\dot{\omega}_x, \dot{\omega}_y, \dot{\omega}_z)$. The state equation (5.3) could be written as:

$$\begin{pmatrix} t_i \\ \omega_i \\ f \\ \dot{t}_i \\ \dot{\omega}_i \end{pmatrix}_{k+1} = \begin{pmatrix} I & 0 & 0 & I\Delta\tau & 0 \\ 0 & I & 0 & 0 & I\Delta\tau \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{pmatrix} \begin{pmatrix} t_i \\ \omega_i \\ f \\ \dot{t}_i \\ \dot{\omega}_i \end{pmatrix}_k + \xi(k) \qquad (5.11)$$

where $i = x, y, z$ is the index of the coordinate axes of the camera reference frame, I is the identity matrix and $\Delta\tau$ is the inter-frame time.

The Measurement Model

The measurement model relates the state vector s to the 2D-image location (u_k, v_k) of each image feature point, p_k. The point $p_k(X_k, Y_k, Z_k)$ of the object reference frame becomes the point $p_{ck}(X_{ck}, Y_{ck}, Z_{ck})$ of the camera reference frame, where:

$$\begin{pmatrix} X_{ck} \\ Y_{ck} \\ Z_{ck} \end{pmatrix} = T(t_x, t_y, t_z) + R(\alpha, \beta, \gamma) \begin{pmatrix} X_k \\ Y_k \\ Z_k \end{pmatrix} \quad , k=1,\ldots,N, \qquad (5.12)$$

where T and R represent the object (or camera) translation and rotation matrices, and N is the number of points.

The observed perspective projection is given by:

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = \frac{f}{Z_{ck}} \begin{pmatrix} X_{ck} \\ Y_{ck} \end{pmatrix} \quad , k=1,\ldots,N, \qquad (5.13)$$

where f is the camera focal length. The above equation is valid when considering the camera system origin in center of projection (*COP*). If the system origin is fixed at the image plane, the above equation transforms to:

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = \frac{1}{1+\beta Z_{ck}} \begin{pmatrix} X_{ck} \\ Y_{ck} \end{pmatrix} \quad , k=1,\ldots,N, \tag{5.14}$$

where $\beta = \frac{1}{f}$ is the inverse focal length. The equivalence can be easily verified, by replacing Z_{ck} in (5.13) with $f + Z_{ck}$ (see Figure 5.1). The choice of the motion and projection equation will reflect in the calculation difficulty of the H Jacobian, at each frame step.

At each filter cycle we have to calculate the partial derivatives of u and v with respect to each of the unknown parameters. Lowe [Low87] proposed a reparameterization of the projection equations, to simplify the calculation by expressing "the translations in terms of the camera coordinate system rather than model coordinates". We need the above partial derivatives to compose the H Jacobian matrix. In this case the measurement equation (5.4) will take the following form:

$$\begin{pmatrix} X_{ck} \\ Y_{ck} \\ Z_{ck} \end{pmatrix} = R(\alpha,\beta,\gamma) \begin{pmatrix} X_k \\ Y_k \\ Z_k \end{pmatrix} \tag{5.15}$$

$$\begin{pmatrix} u_k \\ v_k \end{pmatrix} = \begin{pmatrix} \frac{f}{Z_{ck}+t_z} X_{ck} + t_x \\ \frac{f}{Z_{ck}+t_z} Y_{ck} + t_y \end{pmatrix} \tag{5.16}$$

In this new parameterization, t_x and t_y simply specify the location of the object on the image plane and t_z specifies the distance of the object from the camera". We used the above model in our EKF framework.

The EKF minimization requires the calculation of the first partial derivative (Jacobian) of each measurement relative to each state variable:

$$H_{i,j} = \left(\frac{\partial h_i(s)}{\partial s_j} \right), \qquad i=1,...2N, \qquad j=1,...,M \qquad (5.17)$$

where s is the M-component state vector, and k is the index of the u or v feature coordinates in the observation array $m_k(u_1, v_1, u_2, v_2, ..., u_N, v_N)$.

The magnitudes of the approximate measurement matrix elements reflect the amount of information about each state contained in data obtained at the current point in the object trajectory [Bro86]. We start the calculation of the Jacobian H from the above nonlinear, but differentiable and well-behaved equations (5.15) and (5.16). In the Lowe's model, the object's coordinates in the camera reference frame, X_{ck}, Y_{ck}, and Z_{ck} (5.15) and (5.16) are defined:

$$(X_{ck}, Y_{ck}, Z_{ck}) = (r \cos\omega_z, r \sin\omega_z, z), \qquad (5.18)$$

where r is the distance of the point projection in x-y plane. Table 5.1 gives a summary of the partial derivatives of the X_{ck}, Y_{ck}, and Z_{ck} with respect to the small rotation angles ω_x, ω_y and ω_z.

	∂X_{ck}	∂Y_{ck}	∂Z_k
$\partial\omega_x$	0	$-Z_{ck}$	Y_{ck}
$\partial\omega_y$	Z_{ck}	0	$-X_{ck}$
$\partial\omega_z$	$-Y_{ck}$	X_{ck}	0

Table 5.1: Partial derivatives of structure parameters with respect to the small inter-frame rotations.

Usually, it is difficult to calculate the partial derivatives for the standard form of the projection equation (4.11), (4.12) [Aza95], [Jeb99]. Lowe's [Low87] reparametrization (5.15), (5.16) greatly simplifies the Jacobian calculation. Considering p a SFM state vector parameter, and using the derivation chain rule, we have:

$$\frac{\partial u}{\partial p} = \frac{f}{Z_{ck} + t_z}\frac{\partial X_{ck}}{\partial p} - \frac{fX_{ck}}{(Z_{ck} + t_z)^2}\frac{\partial Z_{ck}}{\partial p} \tag{5.20}$$

For simplicity, we substitute:

$$d = \frac{1}{Z_{ck} + t_z} \tag{5.21}$$

Hence:

$$\frac{\partial u}{\partial p} = fd\left(\frac{\partial X_{ck}}{\partial p} - dX_{ck}\frac{\partial Z_{ck}}{\partial p}\right) \tag{5.22}$$

and similarly,

$$\frac{\partial v}{\partial p} = fd\left(\frac{\partial Y_{ck}}{\partial p} - dY_{ck}\frac{\partial Z_{ck}}{\partial p}\right) \tag{5.23}$$

If we substitute p in (5.15) and (5.16) with the state vector components, and use the Table 5.1 information, we get the following two groups of equations, (5.24) and (5.25) (the Jacobian components):

(i) Horizontal measurements u_k: (4.24)

$$\frac{\partial u}{\partial t_x} = 1$$

$$\frac{\partial u}{\partial t_y} = 0$$

$$\frac{\partial u}{\partial t_z} = -fd^2 X_{ck}$$

$$\frac{\partial u}{\partial \omega_x} = -fd^2 X_{ck} Y_{ck}$$

$$\frac{\partial u}{\partial \omega_y} = fd(Z_{ck} + dX_{ck}^2)$$

$$\frac{\partial u}{\partial \omega_z} = -fdY_{ck}$$

$$\frac{\partial u}{\partial f} = dX_{ck}$$

(ii) ***Vertical measurements*** v_k**:** (4.25)

$$\frac{\partial v}{\partial t_x} = 0$$

$$\frac{\partial v}{\partial t_y} = 1$$

$$\frac{\partial v}{\partial t_z} = -fd^2 Y_{ck}$$

$$\frac{\partial v}{\partial \omega_x} = -fd(Z_{ck} + dY_{ck}^2)$$

$$\frac{\partial v}{\partial \omega_y} = fd^2 X_{ck} Y_{ck}$$

$$\frac{\partial v}{\partial \omega_z} = fdX_{ck}$$

$$\frac{\partial v}{\partial f} = dY_{ck}$$

When N points are tracked, there are $2N$ measurements (coordinates of point projections) at each frame and 7 parameters to be recovered (six motion parameters plus camera focal length). Both motion and focal length are over-determined at each frame when $2N > 7$, which happens when $N \geq 4$, i.e. when tracking 4 or more points. When camera parameters are known beforehand, we need $N \geq 3$ points to recover the 3D motion.

We employ a three-parameter incremental rotation $(\omega_x, \omega_y, \omega_z)$, similar to that used in [Bro86] and [Aza95] to estimate inter-frame rotation. The $\omega_x, \omega_y, \omega_z$ parameters are known as incremental Euler angles. Unbiasing (i.e. centering about the zero mean) these incremental Euler angles, avoids over-parameterization. Being approximately independent, they can be used reliably in system linearization [Jeb99]. An incremental rotation quaternion can be computed from these three parameters:

$$\delta q = \left(\sqrt{1-\eta}, \frac{\omega_x}{2}, \frac{\omega_y}{2}, \frac{\omega_z}{2} \right) \tag{5.26}$$

$$\eta = \frac{\omega_x^2 + \omega_y^2 + \omega_z^2}{4} \tag{5.27}$$

The incremental rotation computed at each frame step is combined into a global quaternion vector (q_0, q_1, q_2, q_3) used in the *EKF* linearization process and rotation of the 3D-model [Sho94]. A *3x3*-rotation matrix *R* can be generated using unit quaternions:

$$R = \begin{pmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1 q_2 - q_0 q_3) & 2(q_1 q_3 + q_0 q_2) \\ 2(q_1 q_2 + q_0 q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_2 q_3 + q_0 q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{pmatrix} \tag{5.28}$$

Figure 5.5 illustrates the integration of this small-rotation algorithm in the recovery of the global rotation of the 3D head model.

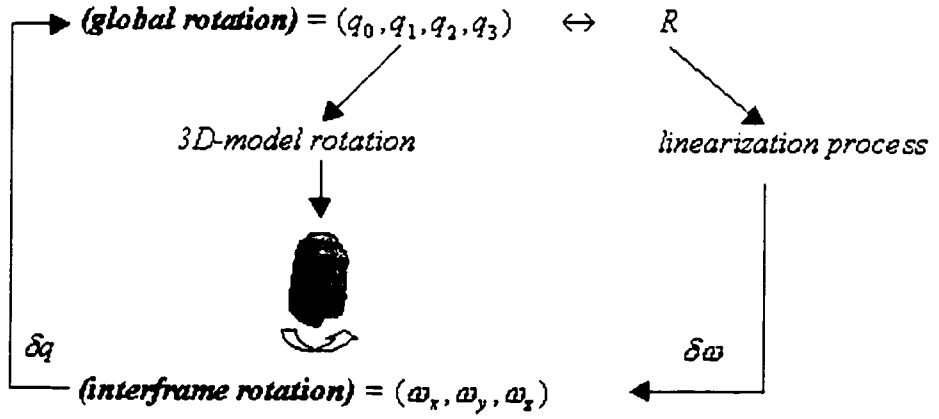


Fig. 5.5: The rotational process in 3D-pose recovery algorithm.

EKF Initialization

The 3D-model provides the initial structure parameters (X_i, Y_i, Z_i) of the Kalman filter. Each 2D-feature point (u_i, v_i) corresponds to a structure point $p_i(X_i, Y_i, Z_i)$. As shown

in Figure 5.6 these (u_i, v_i) points are obtained by intersecting the 2D image plane with a ray rooted in the camera's center of projection COP and aiming to the 3D structure point on the head model.

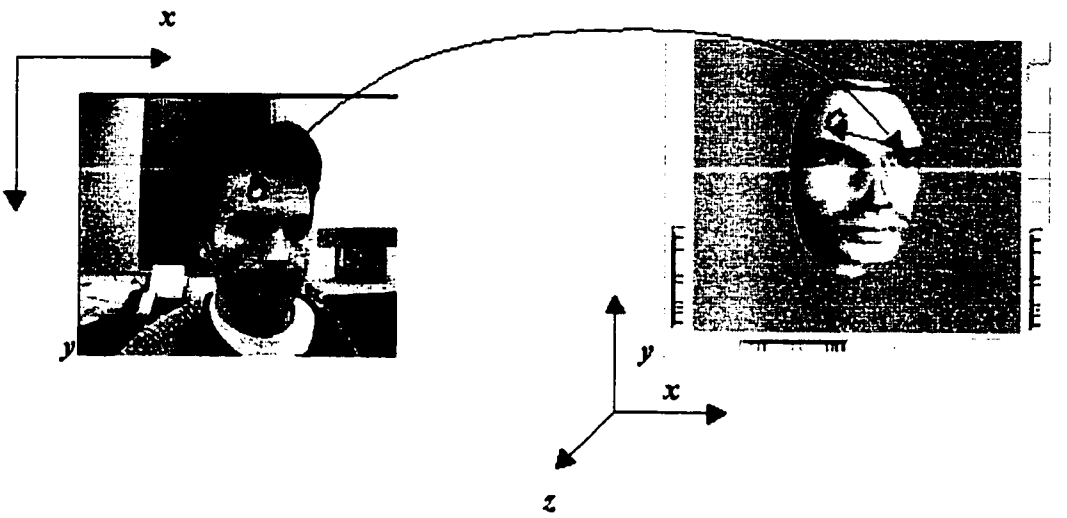


Fig. 5.6: Identical point selection process on Marius' image and the corresponding 3D model projection.

The typical point identification problem of the 3D pose recovery from 2D images is solved in our case by identifying corresponding points in both the 2D live image of the subject and the 3D model of the subject's head. The relation between the normalized coordinate (x_{n_live}, y_{n_live}) of the selected 2D live image point and the 2D perspective screen projection of the same feature point identified on the 3D head model (x_{n_mdl}, y_{n_mdl}) is:

$$x_{n\_live} = 2\frac{x_{live}}{live\_Width} - 1 = x_{n\_mdl}$$

$$y_{n\_live} = 1 - 2\frac{y_{live}}{live\_Height} = y_{n\_mdl}$$

where:

- *live_Width, live_Height* are the dimensions of the "live" Windows area,

- (x_{live}, y_{live}) is the 2D location of the live image feature point.
- (x_{n_mdl}, y_{n_mdl}) is the 2D screen projection of the 3D model point.

In order to aid the point identification process, we are using an augmented reality technique by projecting in the 2D live image the 3D mesh used to model the head. A multiple "point identification" procedure using this augmented reality technique is summarized in Figure 5.7. At this development stage it is still up to the user to arrange the scale matching between the live face image and the projected mesh. This is done manually either by adjusting the camera focus or by adjusting the scale of the projected 3D head image.

The steps of the EKF initialization algorithm are as follows:

Step 1.

The user positions his/her face at the center of the screen, and adjust the matching of the live image and the projected mesh , so that the projected mesh covers the entire facial region.

Step 2.

Left mouse click on every rigid feature point of interest on the live image. An automatic program function takes care to properly align the selected live feature point to a vertex of the projected mesh.

Step 3.

Right mouse click anywhere on the active Windows "live" image triggers the tracking process (by "booting the EKF module).

Fig. 5.7: The EKF "boot" algorithm.

EKF Update

The EKF update stage is illustrated in Figure 5.9. At each iteration, the EKF computes an estimate of the rigid 3D motion that must probably correspond to the motion of the 2D live image. We employ the Kanade-Lucas-Tomasi (KLT) [Tom94] 2D-gradient feature tracking method, which robustly performs the tracking reinforced by the EFK estimation output. An estimate of orientation and camera focal length is found at each step. After the 3D-motion and focal length are recovered, a perspective transformation will project feature points back onto the image to determine an estimated position of the 2D feature trackers. At the next frame in the sequence a 2D tracking is performed starting at this 2D estimated position. The current matching coordinates of tracked features are fed back into the Kalman as the observation vector, and the loop continues. The feedback from EFK is used to update the 3D-model pose parameters, i.e. provides the 3D head tracking information.

5.2.4. An Improvement of the EKF

The EKF success rate depends on a good guess for starting conditions. Linearizing a nonlinear model results in relatively small errors, which can be neglected for estimate accuracy less than 10% [For87]. If the current estimate $\hat{s}(k \mid k-1)$ differs too much from the real one, the first-order approximation in the Taylor extension of the EKF state equations will yield a wrong output and the estimates for the next step will also differ from the true ones. One method to reduce the non-linearity effects is to iteratively apply the *Iterated Extended Kalman Filter* (*IEKF*). The IEKF can be applied locally to a data measurement set "by redefining the nominal trajectory and relinearizing the measurement equation" [Zha92], with good results especially when the measurement function has a strong nonlinear character. The performance is due to the fact that when $\hat{s}(k \mid k)$ is computed after measurement incorporation, this value can be a better state estimate than $\hat{s}(k \mid k-1)$ for evaluating $H(k)$ in the measurement update equations. Then the state estimate could be computed iteratively. This results in replacing the measurement update relations by setting $\hat{s}_0(k) = \hat{s}(k \mid k-1)$ and iterating in Kalman gain and state estimation

update relations. The iteration is stopped when two consecutive state vector estimates differ by less then a pre-selected threshold. Figure 5.8 illustrates the IEKF algorithm.

Step 1. **Set** $\hat{s}_0(k) = \hat{s}(k \mid k-1)$.

Step 2. **Iterate** on Jacobian, Kalman gain and state estimate equations:

$$H_n(k) = \left(\frac{\partial h(k)}{\partial s} \right)_{s=\hat{s}(k|k-1)_n}$$

$$K_n(k) = P(k \mid k-1) H_n{}^T(k) [H(k)_n P(k \mid k-1) H_n{}^T(k) + V(k) R(k) V^T(k)]^{-1}$$

$$\hat{s}_{n+1}(k \mid k) = \hat{s}_n(k \mid k-1) + K_n(k)[m(k) - h(\hat{s}_n(k \mid k-1), 0)],$$

where $n = 0, 1, 2, \ldots$

until $\hat{s}_{n+1}(k \mid k) - \hat{s}_n(k \mid k-1) < ITER\_THRESHOLD$

Step 3. **Update** the system error covariance matrix:

$$P_{n+1}(k \mid k) = (I - K_{n+1}(k) H_{n+1}(k)) P(k \mid k-1)$$

Fig. 5.8: The main steps in Iterated Extended Kalman Filter method.

However, the IEKF should be applied only when the improvement is obvious for a small number of iterations. The reason is the increase of the computational cost (each iteration needs to invert a "*mesurement_size x mesurement_size*" matrix), which reduces in turn the real-time performance. The method could be applied globally (batch mode) to the whole-observed data [Zha92]. When the state evolves over time, the IEKF requires the back propagation of the state estimate, limiting once more its use, due to the inserted delay.

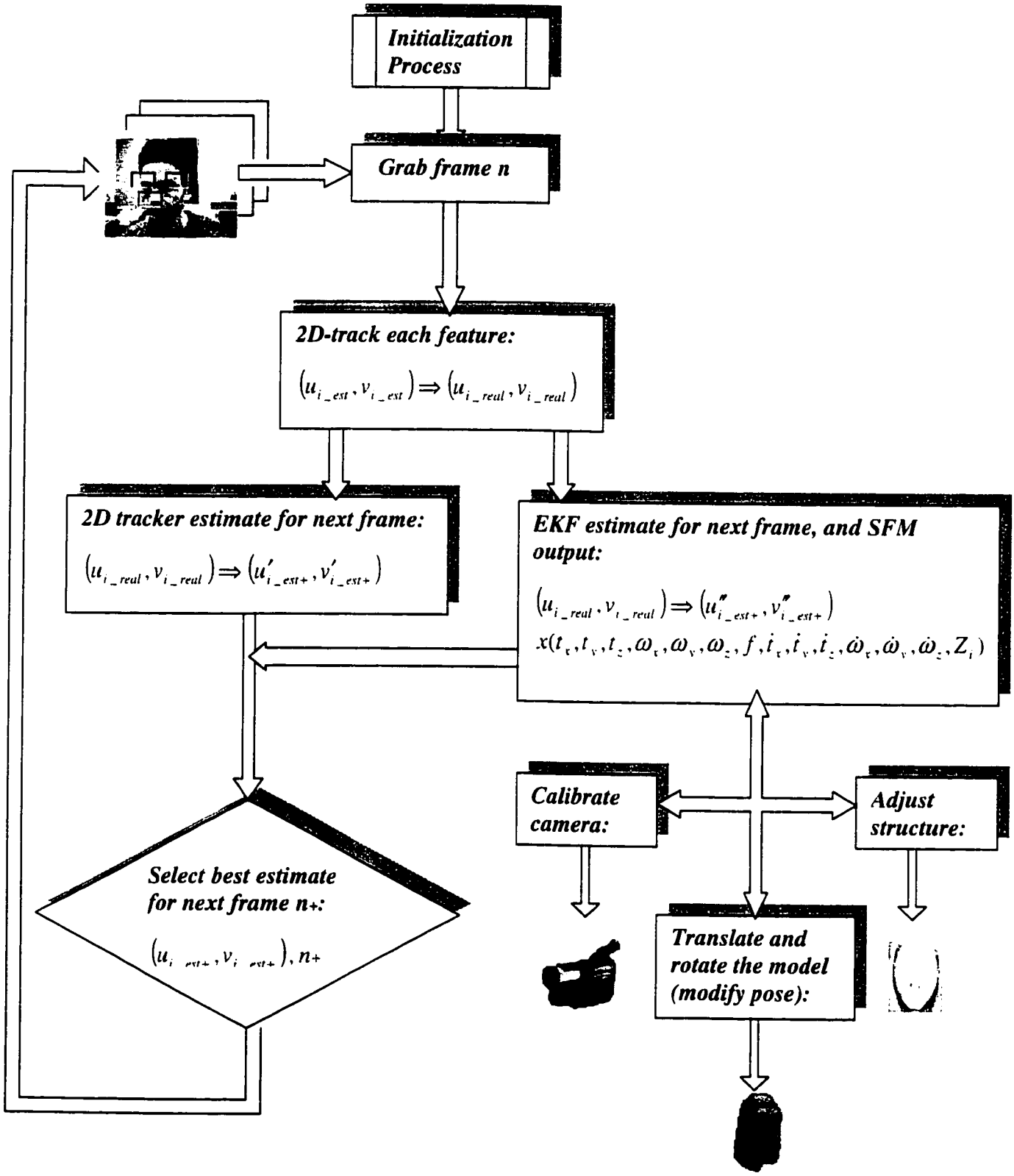


Fig. 5.9: Continuous 3D pose recovery using Extended Kalman Filter.

5.3. Calibration

In order to validate the accuracy of our 3D-head tracking system we developed a rapid calibration technique. A previously recorded sequence of 2D images representing 3D head model poses, is played as "live" image, and tracked with our EKF and IEKF frameworks. The estimated motion values $(t_x^e, t_y^e, t_z^e, \theta_x^e, \theta_y^e, \theta_z^e)$ are compared with the measured motion values $(t_x^m, t_y^m, t_z^m, \theta_x^m, \theta_y^m, \theta_z^m)$ of the synthetic image sequence. In the above representation (t_x, t_y, t_z) is the 3D position, and $(\theta_x, \theta_y, \theta_z)$ is the 3D orientation of the head. The resulted errors show the effect of both human-aided 3D/2D point-identification and 3D tracking. We minimized the errors by fine tuning the initialization process of the EKF. The following figures illustrate different aspects of the calibration process:

(i) Figure 5.10 shows the flowchart of a calibration process using a computer generated sequence pilot.

(ii) Figure 5.11 shows the 3D/2D point-identification process: mesh fitting (a), and point selection (b).

(iii) Figure 5.12 shows the 3D-tracking of a synthetic head during the calibration process.

(iv) Figure 5.13 shows the *recovered* vs. *real* 3D-orientation for a calibrated sequence.

We have found experimentally in one case that the RMS difference between true and recovered rotation angles is 3.035 degrees when tracking 4 points, and 3.002 degrees when tracking 5 points. These statistics are comparable to the Polhemus sensor accuracy [Aza95] indicating that the vision estimate is as least as accurate as the Polhemus sensor.

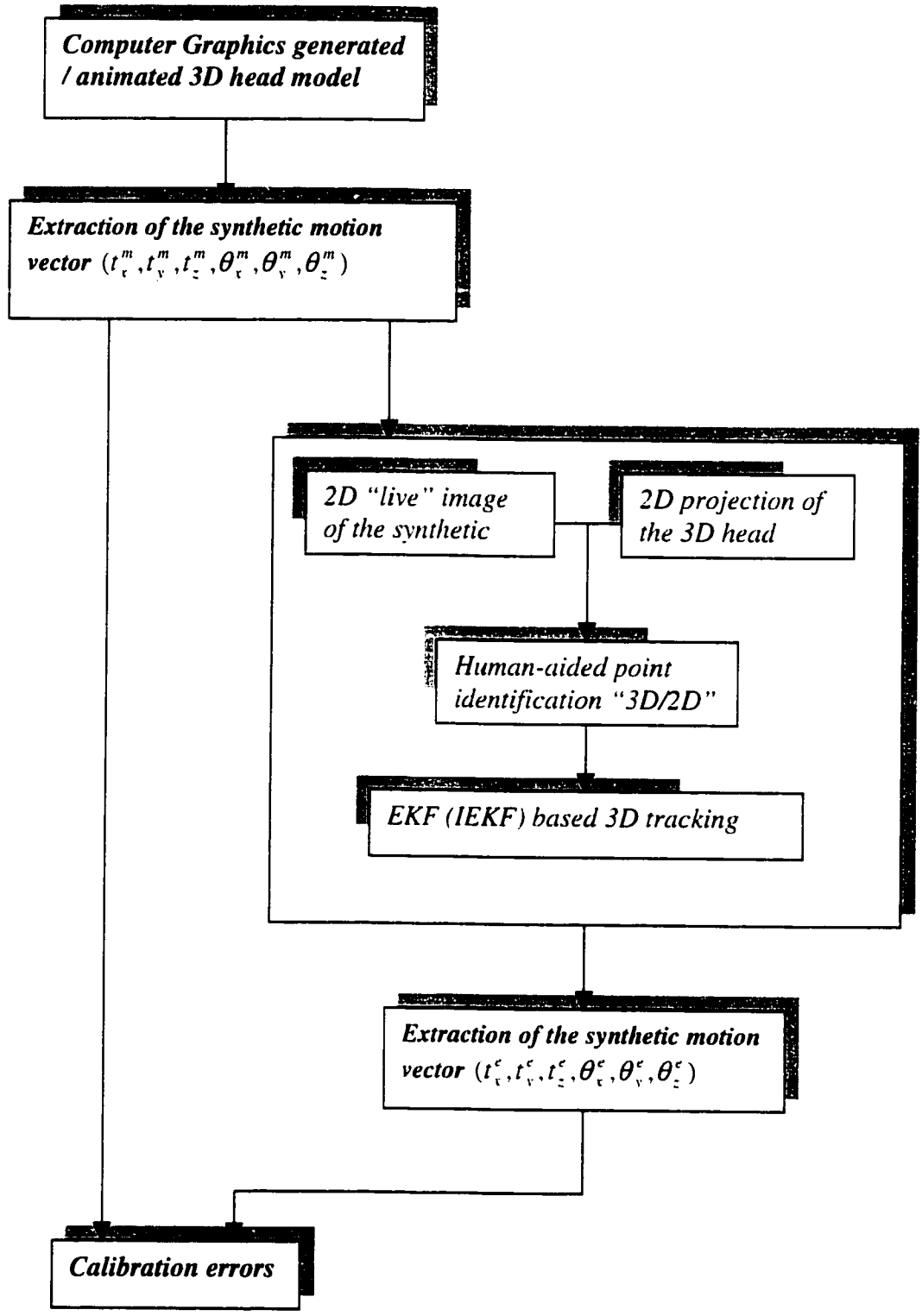


Fig. 5.10: Calibration flowchart.

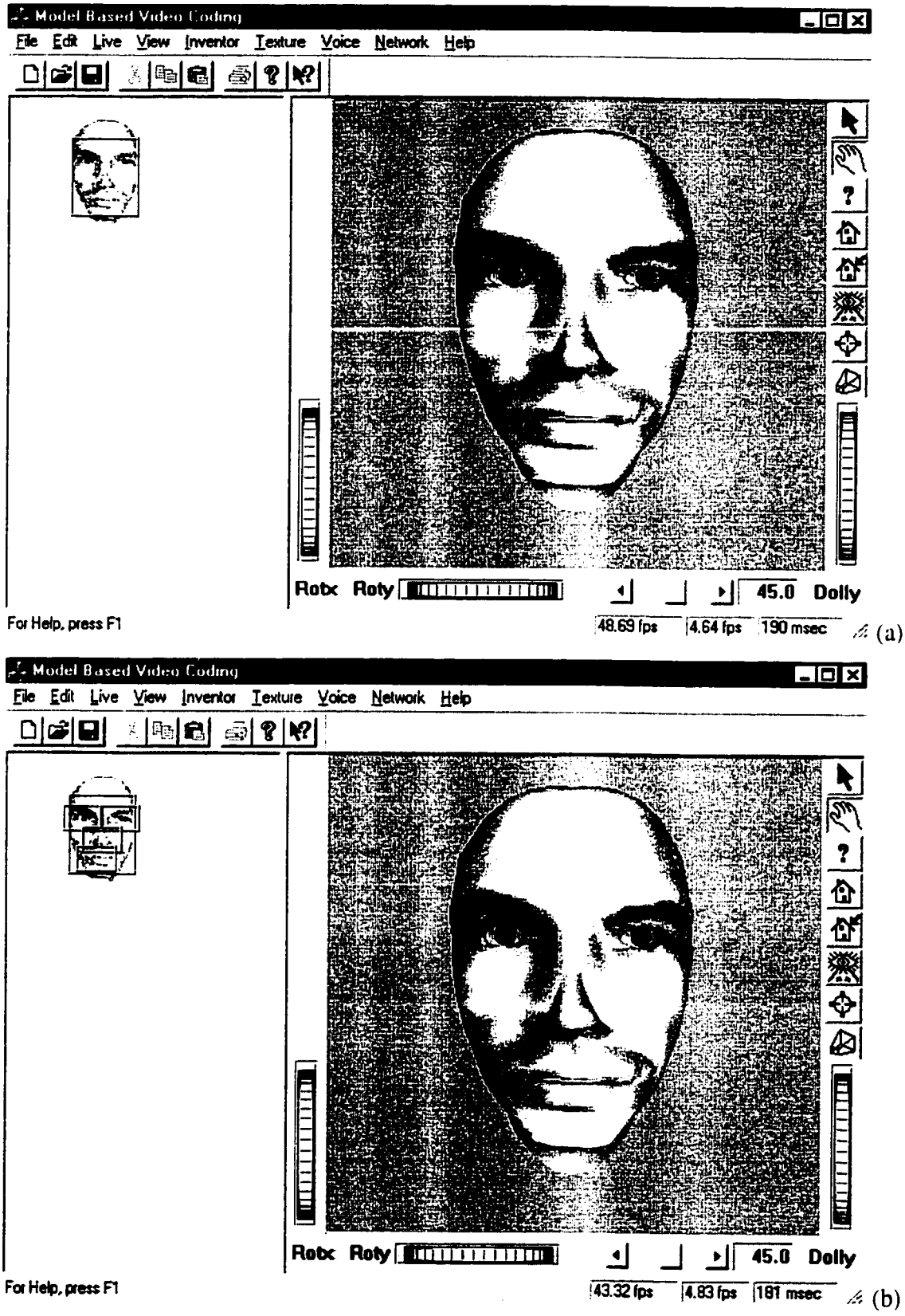
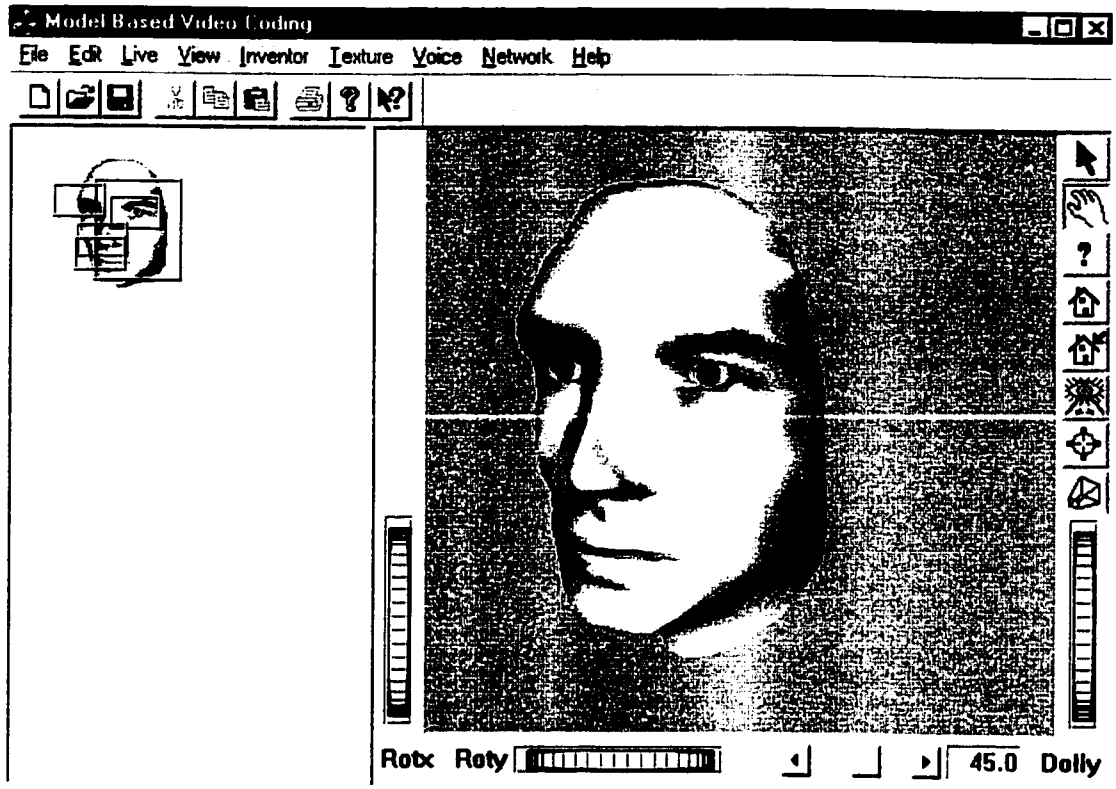
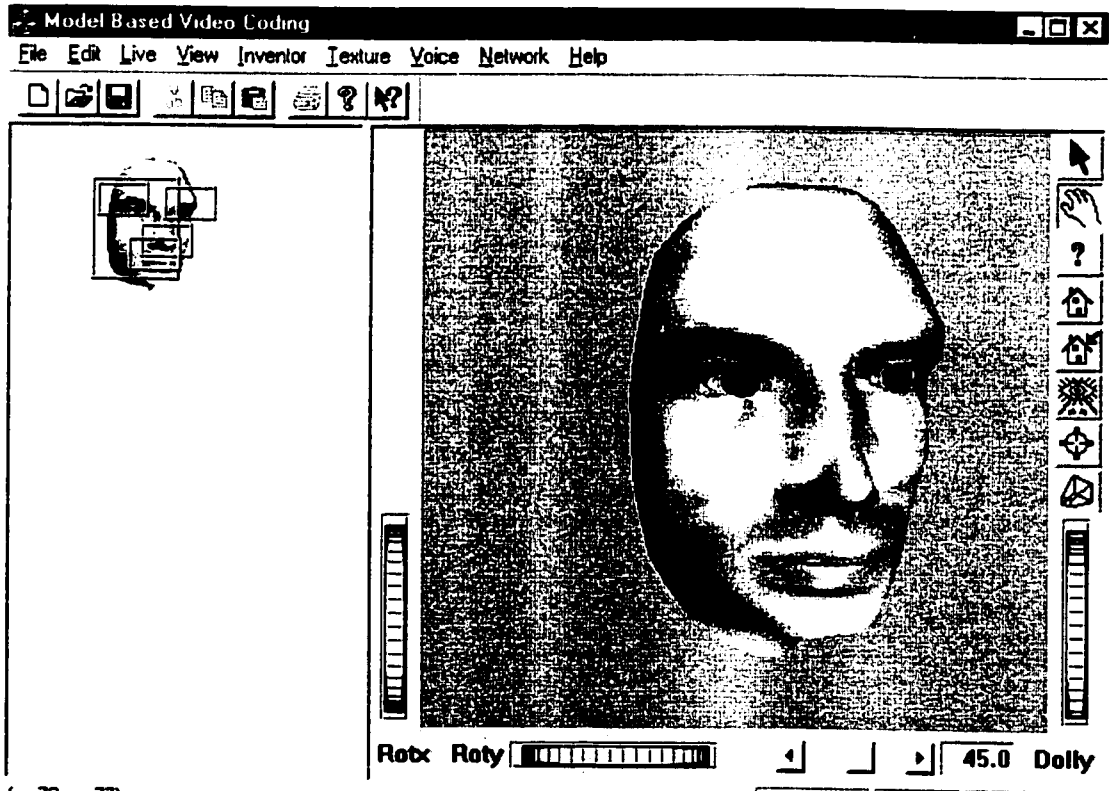


Fig. 5.11: (a) Mesh fitting and (b) point selection in the initial frame.


Model Based Video Coding
File Edit Live View Inventor Texture Voice Network Help
Rotx Roty
45.0 Dolly
(x= 70, y= 77)
30.71 fps
4.05 fps
357 msec

(a)


Model Based Video Coding
File Edit Live View Inventor Texture Voice Network Help
Rotx Roty
45.0 Dolly
(x= 72, y= 77)
39.89 fps
4.19 fps
350 msec

(b)

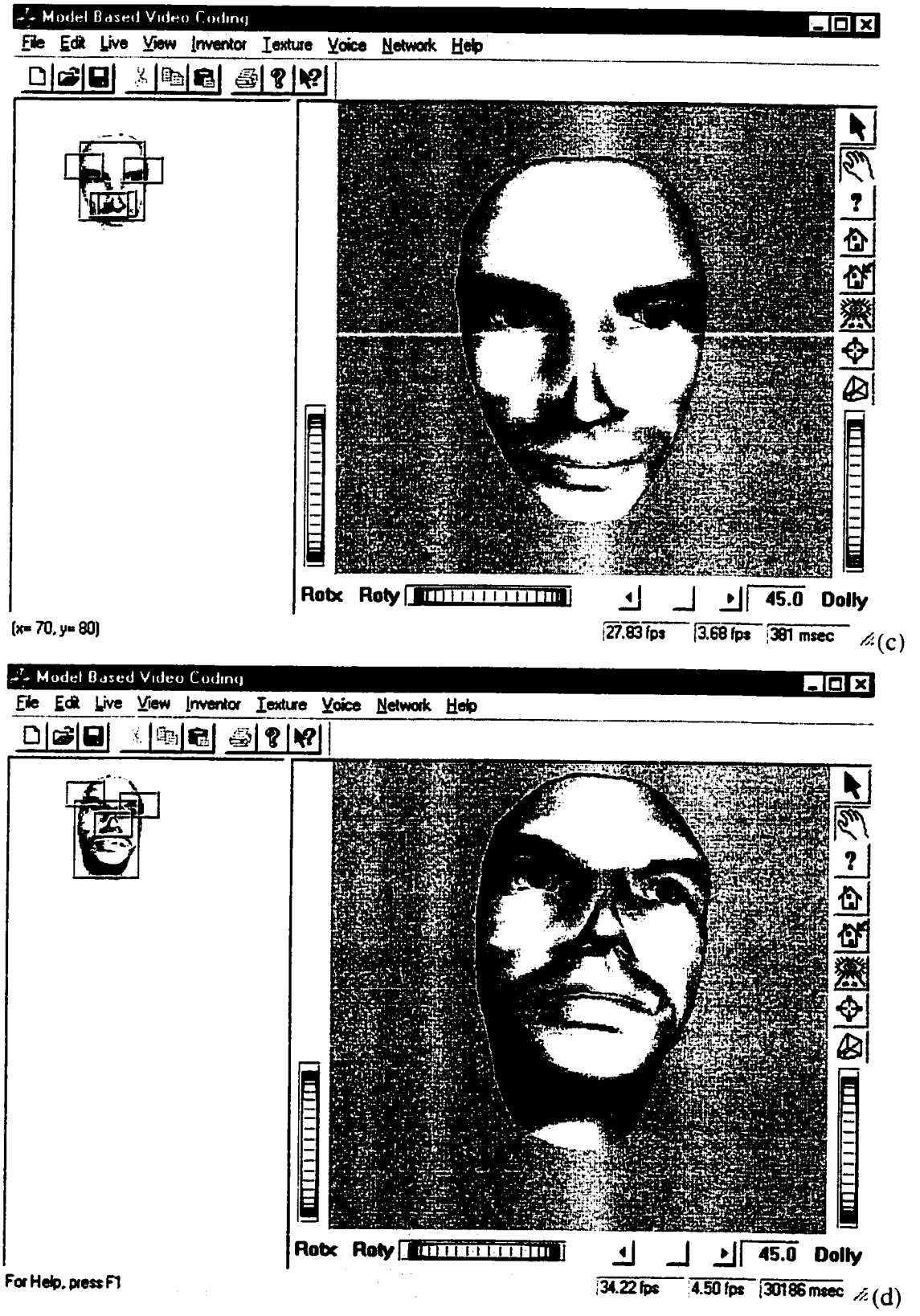
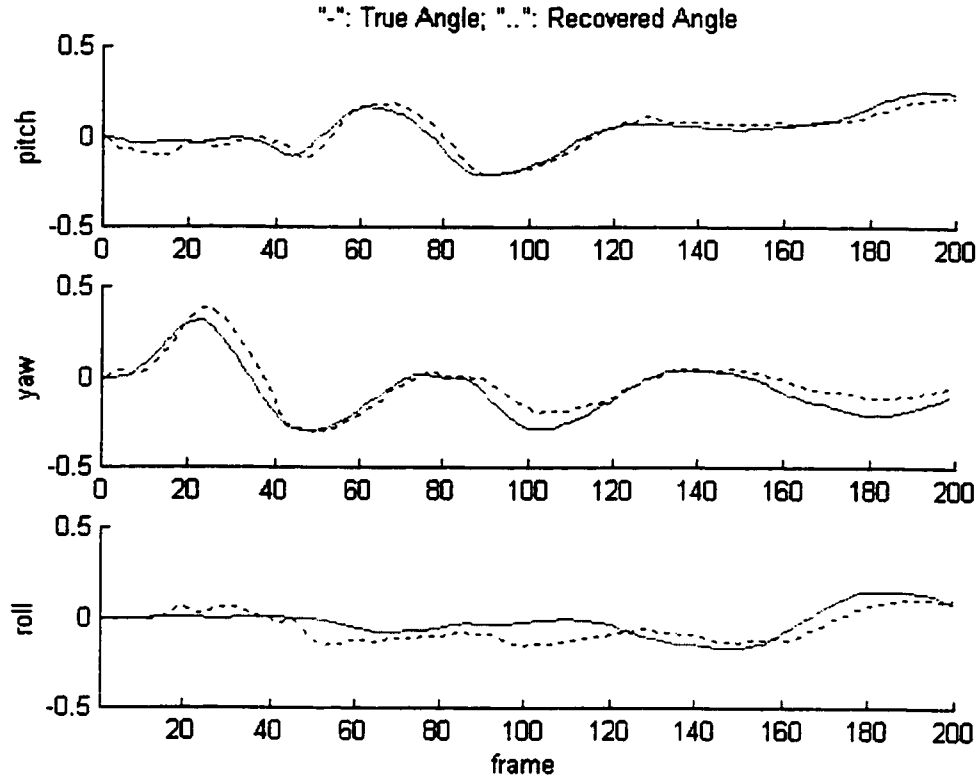
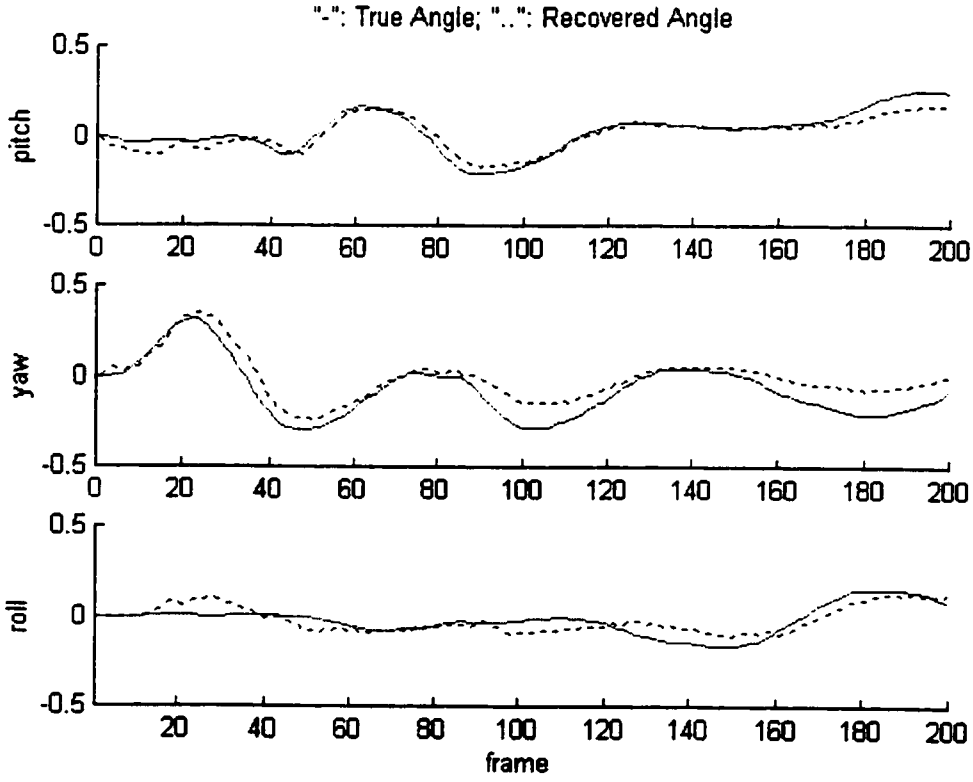


Fig. 5.12: Calibration: (a) frame 25, (b) frame 100, (c) frame 150, (d) frame 200 during tracking.



(a)



(b)

Fig. 5.13: True and recovered rotation angles: EKF-4 points (a) and EKF-5 points (b).

Chapter 6

Conclusions and Further Development

6.1. Conclusions

This thesis presents development issues of an Interactive Facial Animation Testbed (IFAT), for an experimental low bit-rate model-based videoconferencing system [Mas97], [Cor98], [Geo99], [Spo99]. For the specification of the facial expressions we adopted the "muscle-based" facial model parameterization, which is topology independent. Facial expressions are described using the FACS, which allows controlling the movements of specific facial muscles. An array of Controller classes is used for the low-level control of all facial (muscles, jaw, mouth, and eyes) deformation. Currently, the system incorporates 16 muscles and 10 parameters controlling mouth opening, jaw rotation, eye movement, eyelid opening, and head orientation (Chapter 2). A real-time face detection algorithm using a stochastic skin-color model was developed (Chapter 3). Two algorithms were developed for automatic head pose recovery using model-based approaches:

(i) *2½D pose recovery* of the 3D position and 2D orientation, using a 2D elliptical head model (Chapter 4);

(ii) *3D pose recovery* of the 3D position and 3D orientation, using a 3D wireframe head model (Chapter 5);

In order to validate the accuracy of the 3D head tracking system, we developed a rapid calibration technique using a sequence of images of a synthetic "standard" 3D head in lieu of real life head. The discussed model-based videoconferencing concept promises a substantial reduction of the bit rate needed for data transmission at the cost of an equally substantial additional data processing. However the IFAT techniques can be applied for purposes that reach beyond efficient

compression, such as computer-based film making, developing new user interfaces, virtual prototyping environments [Pet99a] and [Pet99b], identification systems, and teaching aids.

6.1.1. Implementation issues

For a computer to "understand" live images and recover meaningful parameters in (almost) real-time is not a trivial task. Most of applications apply geometric and heuristic constraints or use high-end hardware in solving the vision problem. Our goal was to implement MBVC system, which provides real-time performance using as much as possible commercial hardware and software. Our software was developed on Win32 machines and Visual C++ 5.0 as an object-oriented framework. The IFAT *Animation/Rendering* module uses Open Inventor library (www.tgs.com). The IFAT *Performance-Driven* control module uses Microsoft's *Vision SDK 1.0* (www.research.microsoft.com) and Intel's *IPL* (www.intel.com/vtune/perflibst/) libraries. The live data acquisition was performed with the help of *Vision SDK 1.0* library, which supports VFW (Video-For-Windows) and Matrox-Meteor frame grabbers.

6.2. Future Work

Although both 2½D and 3D tracking frameworks we developed perform well, there are a number of pending outstanding issues as outlined below:

Structure Recovery

Object structure is defined as the coordinates of the object feature points in the object-centered coordinate frame. In [Aza95], [Jeb96] the 3D head tracking method recovers not only the motion, but also the object geometry (structure). These positions are constant in time due to the rigidity assumption. It was shown [Aza93] that a recovered structure can help the EKF convergence even when using a 3D model for motion recovery. In EKF extended formulation, one structure parameter can be estimated for each feature point along with the six motion parameters, and camera focal length. All motion and structure parameters could in principle be uniquely recovered using EKF from any image sequence, when $N \geq 7$, where N is the number of tracked

points. An augmented IFAT (with $N \geq 7$) could then provide small depth "corrections" of the recovered 3D-head model.

Adaptive EKF

We have used in our algorithm a time-invariant noise covariance matrix $R(k)$. An improved SFM solution can be obtained by using an adaptive EKF that uses a dynamically varying $R(k)$ matrix that changes with the arrival of new observation vectors to model the confidence of the new data. Thus, if a feature has a very high residual, the filter should trust its spatial information less and focus on other feature tracks. In addition, if a tracked feature is lost or occluded, its correlation window will have a very high residual error and the EKF should essentially ignore its contribution to the estimation. By dynamically changing $R(k)$ using the values of the residuals of the 2D-trackers, it becomes possible to assign a weight on the observations they provide and obtain a more robust estimate of the internal state.

Feature Detection and Model Adaptation

In the case of MBVC system, the input images are synthesized at the receiver end from geometric descriptions of the objects in the scene. Exact and real-time fitting the shape of the generic model to the face seen at the emission end has obvious practical advantages. The model adaptation requires detection of several facial features in first face images. The wireframe mesh could be automatically fitted on the 2D face using detected facial features as anchor points and elastically deforming the rest of the geometry. Such a model will provide better initial depth parameters to the EKF used to recover SFM. The depth deformation will be performed after obtaining the true depth positions of the anchor points during the 3D tracking. Detecting facial features will assure also an automatic re-initialization in the case of the 3D tracking failure.

Expression Recovery and Speech Synchronization

The goal of the developed IFAT framework is to recover both head-pose and facial expressions from the performer facial movement. The effects of head motion and facial expressions will be

non-linear coupled in a 2D-video sequence, so that a successful MBVC method will have to separate the rigid from the non-rigid motion. As a first and important step toward the proposed goal, this thesis resolves the rigid motion recovery problem, leaving as a further development the integration of the facial expression recovery in our unified framework.

The "speech-driven level" which allows animating the face model directly from a speech soundtrack is particularly interesting for videoconferencing and videophone applications.

Bibliography

[Aiz89]K. Aizawa, H. Harashima, and T. Saito, *Model-based analysis synthesis image coding for a person's face,* Image Communication, vol. 1, no. 2, pp.139-152, 1989.

[Aiz95] K. Aizawa, T.S. Huang, *Model Based Image Coding: Advanced Video Coding Techniques for very Low Bit-Rate Applications*, Proc. IEEE, vol. 3, No. 2, Feb. 1995.

[Ara98] Helder Araujo, Rodrigo L. Carceroni, Christopher M. Brown. *A Fully Projective Formulation to Improve the Accuracy of Lowe's Pose-Estimation Algorithm.* University of Rochester Computer Science Department Rochester, NY - 14627 – USA.

[Aza93] Ali Azarbayejani, Thad Starner, Bradley Horowitz, and Alex Pentland. *Visually controlled graphics.* IEEE Trans. Pattern Analysis and Machine Intelligence, 15(6): pages 602-605, June 1993.

[Aza95] A. Azarbayejani and A Pentland. *Recursive estimation of motion, structure, and focal length.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 17(6), 1995.

[Bas96] Sumit Basu, Ifran Essa, and Alex Pentland. *Motion regularization for model-based head tracking.* Technical Report 362, MIT Media Laboratory Perceptual Computing Section, Cambridge, MA, August 1996.

[Bey93] David J. Beymer, *Face Recognition Under Varying Pose.* C.B.C.L. Paper No. 89, MIT AI Laboratory, Dec. 1993.

[Bic91] Martin Bichsel. *Strategies of Robust Object Recognition for the Automatic Identification of Human Faces.* PhD thesis, ETH, Zurich, 1991.

[Bir97] Stan Birchfield, *An Elliptical Head Tracker.* Computer Science Department Stanford University, Stanford, CA 94305

[Bir98] Stan Birchfield, *Elliptical Head Tracking Using Intensity Gradients and Color Histograms.*Computer Science Department, Stanford University, Stanford, CA 94305

[Bis95] C. Bishop, *Neural Networks for Pattern Recognition.* Oxford University Press, 1995.

[Bla92] M. J. Black. *Robust Incremental Optical Flow.* PhD thesis, Yale Univeristy, New Haven, CT, 1992. Research Report YALEU/DCS/RR-923.

[Bla95] M. Black and Y. Yacoob, *Tracking and recognizing rigid and non-rigid facial motions using local parametric models of image motion,* In *Proceedings ICCV '95*, pages 374–381, 1995.

[Boz94] S. M. Bozic. *Digital and Kalman Filtering.* Addison-Wesley, 2nd edition, 1994.

[Bra93] D.H. Brainard, B.A. Wandell, and E.-J. Chichilnisky. *Color constancy: from physics to appearance.* Current Directions in Psychological Science, Vol. 2, No. 5, pages 165-170, 1993.

[Bri90] T. Brint and M. Brady, "Stereo matching of curves", Image and Vision Computing, v8, 1990, pages 50-56.

[Bro86] Ted J. Broida and Rama Chellappa. *Estimation of object motion parameters from noisy images.* IEEE Trans. Pattern Analysis and Machine Intelligence, 8(1): pages 90-99, January 1986.

[Bro91] T. Broida and R. Chellappa. *Estimating the kinematics and structure of a rigid object from a sequence of monocular frames.* IEEE Trans. Pattern Anal. Mach. Intell., 1991.

[Brn83] Brown, R. G. and Hwang, P. Y., *Introduction to Random Signals and Applied Kalman Filtering.* John Wiley & Sons, New York, 1983.

[Bru93] R. Brunelli and T. Poggio. *Face Recognition: Features versus Templates.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 1993.
Journal of the Optical Society of America, pages 519-524, March 1987.

[Bur89] Peter J. Burt. *Multiresolution techniques for image representation, analysis, and 'smart' transmission.* In SPIE Vol. 1199, Visual Communications and Image Processing IV, pages 2-15, 1989.

[Can86] J. F. Canny, *A computational approach to edge detection,* IEEE Trans Pattern Anal. Mach. Intel., v6, 1986, pages 679-698.

[Che95] R. Chelleppa, C. L. Wilson and S. Sirohey. *Human and machine recognition of faces: A survey.* Proceedings of the IEEE, vol. 83, no. 5, May 1995, pages 705-740.

[Cor98] **Marius Cordea**, Emil M. Petriu, Dorina C. Petriu, *Object-Oriented Face Animation For Model-Based Video Compression Applications*, School of Information Technology and Engineering University of Ottawa, Ottawa, Ont., Canada, and Dept. of Systems and Computer Engineering Carleton University, Ottawa, Ont., Canada. ICT98-Porto Caras, Greece, June 1998.

[Dem77] A.P. Dempster, N.M. Laird, and D.B Rubin, *Maximum likelihood from incomplete data via de **em** algorithm.* Journal of the Royal Statistical Society, 39-B: 1-38, 1977.

[Deb96] P.E. Debevec, C.J. Taylor, and J. Malik. *Modeling and rendering architecture from photographs.* In SIGGRAPH '96, August 1996.

[Deg90] B. deGraf and M. Wahrman. *Notes on human facial animation.* In SIGGRAPH Course Notes 26, pages 13-15. ACM SIGGRAPH, 1990.

[Den97] De Natale Francesco, Giusto Daniele, Maccioni Fabrizio *Symmetry-based approach to facial features' extraction.* International Conference on Digital Signal Processing, DSP v2 1997. IEEE, Piscataway, NJ, USA. 97TH8306, pages 521-525.

[Dre81] L. Dreschler and H.-H. Nagel. *Volumetric model and 3d trajectory of a moving car derived from monocular tv frame sequences of a street scene.* IJCAI, 692-697, 1981.

[Eis99] P. Eisert, T. Wiegand and B. Girod, *"Rate-Distortion Efficient Video Compression Using a 3-D Head Model"*, ICIP99, Kobe, October 1999.

[Ekm86] P. Ekman and W. Friesen. *Manual for the Facial Action Coding System.* Consulting Psychologists Press, Palo Alto, 1986.

[Ele95] A. Eleftheriadis and A. Jacquin. *Automatic face location, detection and tracking for model-assisted coding of video teleconferencing sequences at low bit-rates.* Signal Processing:Image Communication, vol. 7, no. 3, July 1995, pages 231-248.

[Ess95] I. Essa and A. Pentland, *Facial expression recognition using a dynamic model and motion energy,* In *Proceedings ICCV '95*, pages 360–367, 1995.

[Fau92] O. Faugeras. *What can be seen in three dimensions from an uncalibrated stereo rig?* In Proceedings of the 2nd European Conference on Computer Vision, pages 563-578, Santa Margherita Ligure, Italy, 1992. Springer-Verlag.

[Fie97] P. Fieguth and D. Terzopoulos. *Color-based tracking of heads and other mobile objects at video frame rates.* In *Proc. of the IEEE CVPR*, pages 21–27, 1997.

[Fin92] G.D. Finlayson, *Color object recognition.* Master's thesis, Simon Fraser Univ., 1992.

[Fol90] J. Foley, A. van Damm, S. Feiner, and J. Hughes. *Computer Graphics: Principles and Practice.* 2nd Edition. Addison-Wesley, Reading, MA, 1990.

[For87] W. Forstner. *Reliability analysis of parameter estimation in linear models with applications to measurements in computer vision.* CVGIP 40, pages 273-310 (1987).

[Fra92] Steve Franks. Facial animation system. Internet, 1992.

[For88] D. A. Forsyth. *Colour Constancy and its Applications in Machine Vision.* PhD thesis. University of Oxford, 1988.

[Fuk93] T. Fukuhara and T. Murakami. *3-d motion estimation of human head for model-based image coding.* IEEE Proceedings, 140(1): 26-35, February 1993.

[Gee96] Gee Andrew, Cipolla Roberto, *Fast visual tracking by temporal consensus.* Image and Vision Computing v 14 n 2 Mar 1996. pages 105-114.

[Gel96] A. Gelb, *Applied Optimal Estimation*, M.I.T. Press, 1996.

[Gen82] D.B. Gennery. *Tracking known 3-dimensional object.* In *Proc. AAAI 2nd Natl. Conf. Artif. Intell.*, pages 13–17, Pittsburg, PA, 1982.

[Gen92] D.B. Gennery. *Visual tracking of known 3-dimensional object. Int. J. of Computer Vision*, 7(3), pages 243–270, 1992.

[Geo99] N.D. Georganas, E. Petriu, **M. Cordea**, D. Ionescu, *Distributed Virtual Environments for Training and Telecollaboration*, Proc. IMTC/98, IEEE Instrum. Meas. Technol. Conf., pages 1847-1850, Venice, Italy, May 1999.

[Gir94] B. Girod, *Image sequence coding using 3D scene models*, In Symposium on Visual Communications and Image Processing 94, Sept. 1994.

[Gir96]B. Girod, K. Ben Younes, R. Bernstein, P. Eisert, N. Färber, F. Hartung, U. Horn, E. Steinbach, K. Stuhlmüller, T. Wiegand, *"Recent Advances in Video Compression"*, Proceedings ISCAS 1996, pages 580-583, Atlanta, March 1996.

[Gon95] S.Gong, A.Psarrou, I.Katsoulis, P.Palavouzis, *Tracking and Recognition of Face Sequences*, Department of Computer Science, Queen Mary and Westfield College, University of London, and School of Computer Science, University of Westminster, London UK, 1995.

[Gun92] Brian Guenter, *A system for simulating human facial expression.* In State of the Art in Computer Animation, pages 191-202. Springer-Verlag, 1992.

[Hag96] G.D. Hager and P.N. Buelhumeur, *"Real-Time Tracking of Image Regions with Changes in Geometry and Illumination"*, IEEE Conference on Computer Vision and Pattern Recognition, pages 403-410, 1996.

[Har92] R. Hartley, R. Gupta, and T. Chang. *Stereo from uncalibrated cameras.* In Proceedings of the Conference on Computer Vision and Pattern Recognition, pages 761-764, Urbana-Champaign, Illinois, 1992.

[Har94] R. Hartley. *Lines and points in three views - an integrated approach.* In Proceedings of the ARPA IU Workshop. DARPA, Morgan Kaufmann, 1994.

[Har98] F. Hartung, P. Eisert and B. Girod, *Digital Watermarking of 3D Head Model Animation Parameters*, 10th IMDSP Workshop 98, pages 231-234, Alpbach, Austria, July 1998.

[Hee90] J. Heel. *Direct estimation of structure and motion from multiple frames.* AI Memo 1190, MIT AI Lab, March 1990.

[Hee91] J. Heel. *Temporal integration of 3-d surface reconstruction.* To appear on IEEE trans. PAMI, special issue on the interpretation of 3-D scenes., March 1991.

[Hor86] Berthold Klaus Paul Horn. *Robot Vision.* MIT Press, 1986.

[Hor90] Berthold Klaus Paul Horn. *Relative orientation.* International Journal of Computer Vision, 4(1): pages 59-78, January 1990.

[Hos95] Robin Hosie, *Predicting Ball Trajectories in Uncontrolled Environments*, MBSc. Thesis, 1995, Curtin University of Technology.

[Hua81] T. S. Huang. *Image Sequence Analysis.* Springer-Verlag, USA, 1981.

[Hua96] T. Huang , J. Stroming y , Y. Kang , and R. Lopez. *Advances in Very Low Bit Rate Video Coding in North America.* IEICE Trans Communications, Vol. E00, No.1 Jan. 1996.

[Hwa89] V.S.S. Hwang. *Tracking feature points in time-varying images using an opportunistic selection approach.* PR, 22(3): 247-256, 1989.

[ISO97] ISO/IEC 14496-2, *Coding of Audio-Visual Objects: Visual (MPEG-4 video)*, Committee Draft, October 1997.

[Jai89] A. K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[Jeb96] Tony Jebara and Alex Pentland, *Parameterized Structure from Motion for 3D Adaptive feedback Tracking of Faces*, Media Laboratory, MIT Cambridge, MA 02139, November 1996.

[Jeb99] Tony Jebara, Ali Azarbayejani and Alex Pentland. *3D Structure from 2D Motion.* MIT Media Laboratory, Cambridge MA, 02139, 1999.

[Jie98] Jie Yang Alex Waibel, *A Real-Time Face Tracker*, School of Computer Science Carnegie Mellon University, Pittsburgh, PA 15213.

[Kac97] R. Kaciuc and A. Blake, *Accurate, Real-Time, Unadorned Lip Tracking*, Dept. of Engineering Science, University of Oxford, UK, 1997.

[Kas88] M. Kass, A. Witkin and D. Terzopoulos, *"Snakes: active contour models"*, International Journal of Computer Vision", v1, 1988, pages 321-331.

[Kis89] F. Kishino, K. Yamashita, *Communication with realistic sensation applied to a teleconferencing system, IEICE Tech. Rep. IE89-35*, 1989.

[Kit80] L. Kitchen and A. Rosenfeld. *Gray-level corner detection.* TR, U. of Maryland, 1980.

[Kos95] A. Kosaka and G. Nakazawa. *Vision-based motion tracking of rigid objects using prediction of uncertainties,* In Proc. 1995 IEEE Conf. on Rob. and Autom., pages 2637{2644,Nagoya, Japan, May 1995.

[Kum89] Ratnam V. Raja Kumar, Arun Tirumalai, and Ramesh C. Jain. *A non-linear optimization algorithm for the estimation of structure and motion parameters.* In Proc. IEEE Conf. on Computer Vision and Pattern Recognition, pages 136-143, June 1989.

[Kut98] K. Kutulakos. *Altering reality through interactive image and video manipulation.* In Computer Vision for Virtual Reality Based Human Communications.CVVRHC'98, Bombay, India, January 1998. IEEE Computer Society.

[Lih93] Haibo Li, Pertti Roivainen, and Robert Forchheimer. *3-d motion estimation in model-based facial image coding.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 15(6): 545-555, June 1993.

[Lih94] Haibo Li, Pertti Roivainen, and Robert Forchheimer. *Two-view facial movement estimation.* IEEE Transactions on Circuits and Systems for Video Technology, 4(3): 276-287, June 1994.

[Lop96] R. L. Lopez, *Head and Feature Tracking for Model Based Video Coding.* University of Illinois at Urbana-Champaign, 1996.

[Low87] David G. Lowe. Three-dimensional object recognition from single two-dimensional images. Artificial Intelligence, 31(3): pages 355-395, March 1987.

[Low91] David G. Lowe. *Fitting parameterized three-dimensional models to images.* IEEE Trans. on Pattern Analysis and Machine Intelligence, 13(5): pages 441-450, May 1991.

[Luo96] Luo Suhuai and King Robin, *Automatic human face modeling in model-based facial image coding.* Proceedings of the Australian and New Zealand Conference on Intelligent Information Systems 1996, 96TH8234, pages 174-177.

[Mar79] D. Marr, T. Poggio, and S. Ullman. *Bandpass channels, zero-crossings, and early visual information processing.* JOSA, 69:914-916, 1979.

[Mar82] D. Marr, *"Vision"* W. H. Freeman & Company, USA, 1982.

[Mas97] M.H. Assaf, **M. Cordea**, M. Bondy, C.M. Nafornita, E. Petriu, H.J.W. Spoelder, *Image/Voice Modeling and Synchronization for Model-Based Video-Telephony*, Proc. ET&VS-IM/97 IEEE Workshop on Emergent Technol. and Virtual Systems for Instrum. Meas., pages 165-171, Niagara Falls, Ont., 1997.

[Mat89] L. Matthies, R. Szelisky, and T. Kanade. *Kalman filter-based algorithms for estimating depth from image sequences.* Int. J. of computer vision, 1989.

[Med86] G. Medioni and Y. Yasumoto, *Corner Detection and curve representation using cubic B-splines*, IEEE International Conf. on Robotics and Automation, 1986, pages764-769.

[Mck96] McKenna Stephen, Gong Shaogang, *Tracking faces.* Queen Mary and Westfield College, London, Engl. In Proceedings of the International Conference on Automatic Face and Gesture Recognition 1996. IEEE, Los Alamitos, CA, USA, 96TB100079. pages 271-276

[Mck97] Stephen J. McKenna, Yogesh Raja and Shaogang Gong, *Object Tracking using Adaptive Colour Mixture Models.* Dept. of Computer Science, Queen Mary and Westfield College, London.

[Moh96] R. Mohr and B. Triggs. *Projective geometry for image analysis.*Technical report, International Society for Photogrammetry and Remote Sensing, Vienna Congress, July 1996. WG III/2 Tutorial.

[Mos94] Y. Moses, Y. Adini, and S. Ullman. Face recognition: the problem of compensating for changes in illumination direction. In Third European Conf. on Computer Vision, pages 286-296, 1994.

[Mov80] H. Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* PhD,Stanford U., 1980.

[Muf96] Mu Fenghao, Li Haibo, Forchheimer Robert, *Automatic extraction of human facial features.* Linkoping Univ, Linkoping, Sweden. In Signal Processing: Image Communication v 8 n 4 May 1996, pages 309-326.

[Nak92] Yuichiro Nakaya and Hirshi Harashima. *Model-based/waveform hybrid coding for low-rate transmission of facial images.* IEICE Transactions on Communications: Special Issue on Video Coding and Its Applications, E75-B(5):377-384, May 1992.

[Nur98] Nuria Oliver, Alex Pentland Francois Berard, *LAFTER: Lips and Face Real Time Tracker with Facial Expression Recognition.* M.I.T. Vision and Modeling Group, CLIPS-IMAG, BP 53, 1998.

[Ohz96] H. Ohzu, K. Habara, *Behind the scenes of Virtual reality: Vision and motion*, Proc. IEEE, vol. 84, No. 5, May 1996.

[Pal94] P. Palavouzis. *Head Tracking for Face Recognition.* Master's thesis, Department of Computer Science, QMW, London, 1994.

[Par72] F. Parke. *Computer generated animation of faces.* In ACM National Conference, pages 451-457. ACM, 1972.

[Par74] Parke, F. I. *A Parametric Model for Human Faces.* Phd Thesis, University of Utah, 1974, UTEC-CSS-75-074.

[Par96] F. I. Parke, K. Waters, *Computer Facial Animation*, A. K. Peters, Ltd., 1996.

[Par94] J. Park, N. Yagi, K. Enami, K. Aizawa. *Estimation of Camera Parameters from Image Sequence for Model-Based Video Coding.* In IEEE Transaction on Circuits and Systems for Video Technology, 4(3), pages 288-296, June 1994.

[Pel94] Catherine Pelachaud, Norman I. Badler and Marie-Luce Viaud "*Final Report to NSF of the Standards for Facial Animation Workshop*" University of Pennsylvania, School of Engineering and Applied Science, Computer and Information Science Department, Philadelphia, PA 19104-6389

[Pen94] A. Pentland, B. Moghaddam, and T. Starner. *View-Based and Modular Eigenspaces for Face Recognition.* In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition, pages 84-91, June 1994.

[Pet99a] E. Petriu, **M. Cordea**, D.C. Petriu, *Virtual Prototyping Tools for Electronic Design Automation*, IEEE Instrum. Meas. Mag., Vol. 2, No. 2, pages 28-31, 1999.

[Pet99b] E. Petriu, **M. Cordea**, D.C. Petriu, Lou McNamee, "Modelling Issues in Virtual Prototyping Environments," Proc. VIMS'99, IEEE Workshop on Virtual and Intelligent Measurement Systems, pages 1-5, Venice, Italy, May 1999.

[Pla81] Platt, S. M. and Badler, N. I. *Animating Facial Expressions.* Computer Graphics. 15(3): 245, 1981.

[Pog91] Kah-Kay Sung and Tomaso Poggio, *Example-based Learning for View-based Human Face Detection.* A.I. Memo No. 1521 December, 1994, C.B.C.L. Paper No. 112, MIT AI and CBCL Laboratories

[Ree90] B. Reeves. *Facial animation in tin toy.* SIGGRAPH'90 Course 26, Facial Animation, pages 90-106, 1990.

[Ren96] Reinders M.J.T, Koch R.W.C., Gerbrands J.J., *Locating facial features in image sequences using neural networks.* Delft Univ of Technology, Delft, Neth. Proceedings of

the International Conference on Automatic Face and Gesture Recognition 1996. IEEE, Los Alamitos, CA, USA, 96TB100079, pages 230-235.

[Row98] Rowley Henry, Baluja Shumeet, Kanade Takeo, *Neural network-based face detection.* In IEEE Transactions on Pattern Analysis and Machine Intelligence v 20 n 1 Jan 1998. pages 23-38

[San98] Sanger Demas, Miyake Yoichi, Haneishi Hideaki, Tsumura Norimichi. *Algorithm for face extraction based on lip detection.* In Journal of Imaging Science and Technology v 41 n 1 Jan-Feb 1997. pages 71-80.

[Sch96] R. Schaphorst, *Videoconferencing and Videotelephony: Technology and Standards*, Artech House, Inc., 1996.

[Sed86] Sederberg, T.W. and Parry, S. R. *Tree-form Deformation of Solid Geometry Models.* Computer Graphics, (SIGGRAPH 1986) 20(4): 151, 1986.

[Sha95] Larry S. Shapiro. *Affine analysis of image sequences.* PhD Thesis. Sharp Lab. of Europe, Oxford, U.K., 1995.

[Shs95] A. Shashua and M. Werman. *On the trilinear tensor of three perspective views and its underlying geomtry.* In International Conference on Computer Vision, 1995.

[Sho94] Ken Shoemake. *Quaternions.* Department of Computer and Information Science University of Pennsylvania Philadelphia, PA 19104.

[She92] C. Shekhar and R. Chellappa. *Passive ranging using a moving camera.* J. of Robotics S. vol. 9, 1992.

[Sin94] P. Sinha. *Object Recognition via Image Invariants. A Case Study.* In Investigative Ophthalmology and Visual Science, volume 35, pages 1735-1740, Sarasota, Florida, May 1994.

[Smi92] S. Smith. *Feature Based Image Sequence Understanding.* PhD. Thesis, Dept. Engineering Science, Oxford University, 1992.

[Sob96] K. Sobottka and I. Pitas, *A novel method for automatic face segmentation,* Department of Informatics, University of Thessaloniki 540 06, Greece, 1996.

[Set87] I.K. Sethi and R. Jain. *Finding trajectories of feature points in a monocular image sequence*, PAMI, 2:574-581, 1987.

[Spo99] H.J.W. Spoelder, E. Petriu, T. Whalen, D.C. Petriu, **M. Cordea**, *Knowledge-Based Animation of Articulated Anthropomorphic Models for Virtual Reality Applications*, Proc. IMTC/98, IEEE Instrum. Meas. Technol. Conf., pp. 690-695, Venice, Italy, May 1999.

[Ste95] R. Steimmetz, K. Nahrstedt, *Multimedia: Computing, Communication and Applications*, Prentice Hall, 1995.

[Sze93] Richard Szeliski and Sing Bing Kang. *Recovering 3d shape and motion from image streams using non-linear least squares.* In IEEE Conference on Computer Vision and Pattern Recognition, pages 752-753, Los Alamitos, CA, June 1993. IEEE Computer Society, IEEE Computer Society Press. (New York).

[Swa91] M. J. Swain and D. H. Ballard, *Colour indexing*. IJCV, pages 11-32, 1991.

[Tek94] Gozde Bozdagi, Murat Tekalp, and Levent Onural. *3-d motion estimation and wireframe adpatation including photometrics for model-based coding of facial image sequences.* IEEE Transactions on Circuits and Systems for Video Technology, 4(3): 246-256, June 1994.

[Ter88] D. Terzopoulos and K. Fleischer, *Deformable models*, The Visual Computer, 4(6), pages 306-331, December 1988.

[Ter89] D. Terzopoulos, J. Platt, and K. Fleischer. *Heating and melting deformable models*, In Proceedings of Graphics Interface '89, pages 219-226, June 1989.

[Ter90] D. Terzopoulos and K. Waters, *Physically-based facial modeling, analysis, and animation.* Visualization and Computer Animation, pages 73-80, 1990.

[Tha88] Magnenat-Thalmann, N. N.E. Primeau and D. Thalmann. *Abstract Muscle Action Procedures for Human Face Animation*. Visual Computers, 3(5): 290, 1988.

[Toy95] K. Toyama and G. D. Hager. *Tracker fusion for robustness in visual feature tracking*, In SPIE Int'l Sym. Intel. Sys. and Adv. Manufacturing, volume 2589, Philadelphia, PA,October 1995.

[Tom92] Carlo Tomasi and Takeo Kanade. *Shape and motion from image streams under orthography: a factorization method.* International Journal of Computer Vision, 9(2): pages 137-154, November 1992.

[Tom94] Jianbo Shi, and Carlo Tomasi, *Good Features to Track*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR94) Seattle, June 1994.

[Vin92] J.M. Vincent, J.B. Waite, and D.J. Myers. *Location of feature points in images using neural networks.* BT Technology Journal, 10(3): 7-15, July 1992.

[Yok96] Yokoyama Taro, Wu Haiyuan, Yachida Masahiko, *Automatic detection of facial feature points and contours.* Robot and Human Communication - Proceedings of the IEEE International Workshop 1996. IEEE, Piscataway, NJ, USA, 96TH8179 p. pages 335-340

[Yui92] A. Yuille, P. Hallinan, and D. Cohen. *Feature Extraction from Faces using Deformable Templates.* International Journal of Computer Vision, 8(2):99-111, 1992.

[Zha92] Z. Zhang and O. Faugeras, *3D Dynamic Scene Analysis*, Springer Series in Information Sciences, 1992.

[Wat81] Waters, K. *A Muscle Model for Animating 3D Facial Expressions.* Computer Graphics. 21(4): 17, July 1987.

[Wat92] Waters, K. *Modelling Three-Dimensional Facial Expressions.* Shlumberger Laboratory for Computer Science, Austin, TX in *Processing Images of Faces* by Bruce V. and Burton M., New Jersey, 1992.

[Web81] J.A. Webb and J.K. Aggarwal. *Visually interpreting the motion of objects in space.* In Computer, 14:40-46, 1981.

[Wel98] Greg Welch, Gary Bishop, *An Introduction to the Kalman Filter.* TR95-041 Oct. 1998, University of North Carolina at Chapel Hill.

[Wen93] Juyang Weng, Narendra Ahuja, and Thomas S. Huang. *Optimal motion and structure estimation.* IEEE Trans. Pattern Analysis and Machine Intelligence, 15(9): pages 864-884, September 1993.

[Wil93] C. S. Wiles, "Image Sequence Understanding". First year report, Dept. Eng. Sc., Oxford University, 1993.

[Wla94] Wladyslaw Skarbek and Andreas Koschan, *Colour Image Segmentation. A Survey.* Institute of Computer Science Polish Academy of Sciences, Institute for Technical Informatics, Technical University of Berlin, October 1994

[Wre97] C.R. Wren, A. Azarbayejani, T. Darrell, and A.P. Pentland, *Pfinder:real-time tracking of the human body.* IEEE PAMI, vol. 19, no. 7, pages 780-785, 1997.

[Wuh96] Wu Haiyuan, Chen Qian, Yachida Masahiko, *Detecting human face in color images.* Proceedings of the IEEE International Conference on Systems, Man and Cybernetics v 3 1996. IEEE, Piscataway, NJ, USA,96CH35929. pages 2232-2237

[Wys82] Wyszecki G. and Stiles W. S. *Color science,* Wiley, New York, 1982.

[Xug90] G. Xu, H. Agawa, Y. Nagashima, F. Kishino, Y. Kobayashi, *Three dimensional face modeling for virtual space teleconferencing systems*, IEICE Trans., vol. E73, no. 10, 1990.