Dubbing Movies: Adjusting Lip Movements to a New Sound Track

A thesis submitted in fulfillment
of the requirements for the degree of
Master of Science

by

Ran Cohen

supervised by
Prof. Shmuel Peleg
Prof. David Avnir

Institute of Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel.

1995

Contents

Abstract

Dubbing is the replacement of sound track of a movie to another language. By analyzing the original movie, and a movie of the translator speaking the target language, the lip movement of the actor can be matched to the new language. This is achieved by using a three dimensional parameterized facial model that helps analyze and track the actor's face. Texture mapping is then used to synthesize the new face at each frame. This work combines several computer graphics and computer vision techniques to produce a photo-realistic synthesized movie.

Chapter 1

Introduction

1.1 This Work

There are two common approaches to handle movies in a foreign language. One approach is to use subtitles, and another approach is to use dubbing: substituting the original sound track with a new sound track, in the local language. Both solutions have benefits and drawbacks

When subtitles are used, the viewer simultaneously hears one language and reads another language. This is a handicap in movie perception and enjoyment. When reading, the viewer can not see and consentrate in the movie, and otherwise he can not understand the foreign language. Dubbing is much more common in the western world, but raises another problem: the lips of the actors do not move according to the new sound track. Although an effort is being made by the translators to synchronize with the actors's lips, (usually by "finding the right words"), there is often a phonetic-visual gap which can sometimes be untolerable, especially for viewers that are not used to dubbing.

This work tries to manipulate the actor's lips to fit the translator's lan-

guage. Using a three-dimensional, calibrated human face model, the global head movements and lips deformation are tracked along the movie.

The initial face model is a generic, $3D$ wire-frame model. By identifying several feature points on a frontal and side views of the actor, the generic model is adjusted to fit the actor's face. After this initial calibration, the $3D$ mask tracks the actor's face in the next frame (see Chapter 3), including lips location, shape, size, and width.

The same process is performed also on a movie showing the translator. Instead of the usual audio recording of the translator, a video recording of the translator is now needed. This translator video must be synchronized with the original movie, just like a new audio track is usually synchronized in a dubbed movie.

Using the original digitized movie and the information from the tracked actor and the tracked translator, the original movie is changed, where the global head parameters (location, orientation and size) are maintained, but the lips area (lips, jaw, cheeks) are altered to fit the translator's lips. The lips fitting is done by texture mapping the actor's face in each frame on the current $3D$ face model, and setting the lips parameters of the model to those of the translator.

1.1.1 Basic Constraints and Assumptions

Since faces are three dimensional non-rigid (deformable) objects, the analysis and motion tracking of a face in a video is a not a simple task, nor is a realistic synthesis of a new face, or face expression from a given image (or images). This work overcomes these problems by combining image processing, computer vision and computer graphics techniques. But, as always is the case in these fields, there is a need for several constraints and assumptions.

A weak perspective projection is assumed for the imaging system. Even though the imaging is better described by a perspective projection, this assumption is good enough for our purposes. Weak perspective means an orthographic projection of the $3D$ scene onto the image plane, in addition to a scale factor. Under this projection model, the face can be rotated around all axes and translated only in the image plane. It can also be scaled equally in all directions. This model was chosen since it is close enough to the real world, and it simplifies the mathematical calculations on this model's transformations. Since many transformation calculations from the $3D$ (model) world to $2D$ image plane, and back to the $3D$ world are taking place, assuming weak perspective makes those calculations possible.

A high frame rate in the movie is also assumed. Under this assumption the displacement of the actor's face from one frame to another is small, and the motion field created by the actor's movements can be obtained by simple correlation.

In order for the tracking to work properly, a constrain must be given for the face of the actor to forward the camera at least 45 deg. Otherwise, difficulties are araising in both finding correlation features and later synthesizing a realistic side view of the face.

Several problems arises from the fact that the synthesized face is placed over the original face in the image, or from some lack of information (like missing teeth, when a closed mouth is to be opened). All of these problems will be discussed in time.

1.1.2 $3D$ Facial Model

In order to produce a personal face model, two steps are needed. First, a generic face model must somehow be digitized into the computer. This is

done by using a $3D$ digitizer that samples a human face and outputs the three-dimensional coordinates of points on the $3D$ face. The higher the sampling density, the better the model, but more vertices to manipulate. In the face model used in this work, there are about 290 vertices in each side of the face (duplicated around the middle axis to form symmetric face model), giving 580 vertices to manipulate. Next, the $3D$ generic model is scaled and adjusted to fit a front view of a specific person, like in [1].

Today, $3D$ facial models are mostly used in model-based image coding. Model-based coding is a new framework for image compression [1], which makes use of the $3D$ properties of an object in a scene, analyzes and transmits only the parameters for the object. Since the encoder sends only the very few required parameters, extremely low bit rate image transmission can be potentially achieved. The specific class of model based coding of human faces has many applications: low-bandwidth teleconferencing [11], videophone, animation, face recognition (in police work, for example) and many more.

After having the adjusted face model of the specific actor, texture-mapping techniques are used to synthesize a photo realistic $3D$ face. By altering the model parameters, new expressions are created.

1.1.3 FACS: Face Parameterization

A facial expression can be described by an "action unit", based on the Facial Action Coding System (FACS) by Ekman and Friesen [4]. Action units (AU) stands for a small change in the facial expression that is dependent on a small conscious activation of muscles [6]. The AU is a kind of knowledge that is expressed in parameter form. This work presents a $3D$ wire-frame face model, containing both facial shape and expression information. The shape information is stored in about 570 three-dimensional vertices and the

connections between them, forming about 510 3-4 vertex polygons. The expression information, along with more parameterizing variables of the model, are stored separately.

Using those parameters, many facial expressions can be contolled. Parameters like face color, location and size, aspect-ratios of face regions, location of specific face features and many more. These parameters will be refered to as "facial expressions".

1.2 Background

1.2.1 The Human Face

The recognition and manipulation of human faces has been a very challenging subject for much research in the recent years. Researchers in several different fields including psychology, computer graphics and computer vision, are interested in the human face. Psychologists are investigating and studying the perception of human face expressions and face recognition. Computer scientists are interested in face recognition, analysis of expressions, and synthesis of realistic images. Analysis of faces is boosted today for the next generation of visual telecommunication services [6].

The human face is capable of generating about 55,000 different and distinguishable expressions with about 30 semantic distinctions [16]. Ekman and Friesen [4] tried to quantify the facial expressions by presenting their Facial Action Coding System (FACS), in terms of about 50 controllable parameters which they called *action units* (AU) each AU involves one or more muscles and associated activation levels. Impressive animation sequences can be produced by, for example, using Parke's implementation [12] of Ekman and Friesen's FACS, which uses about 55 such action units, available as a public domain application.

This work uses an extension of Parke's model which was implemented and kindly given to us by Andrew Marriott from Curtin University of Technology in Western Australia. Wire-frame mask data and full implementation of the face parameterization is included as part of this software.

1.2.2 Parameterized Models

In his work [12] Parke describes the two basic ideas involved in the creation of parameterized graphical models: the first is the underlying concept of parameterization and the development of an appropriate set of parameters, and the second is that of graphic image synthesizers that produce images based on some pre-defined parameterization.

Parke describes parameterization as some criteria values that distinguish one individual object of some class from another. A complete set of criteria or parameters is one that allows every member of an object class to be specified by selecting appropriate parameter values. If certain members of the class can be differentiated but not uniquely described by parameter values, the parameter set is not complete.

Given the parameterization scheme, the next task is to develop a synthesis model to produce images based on the parameter values. The heart of this process is a set of algorithms, functions and data that form the parametric model. This model takes as input the parametric values specified by the animator and outputs a set of graphic primitives, such as vector and polygon descriptors. These descriptors are passed as input to the graphics routines that actually create the image of either wire-frame object, shaded (See figure 1.1) or even texture-mapped objects that gives much more realistic results.

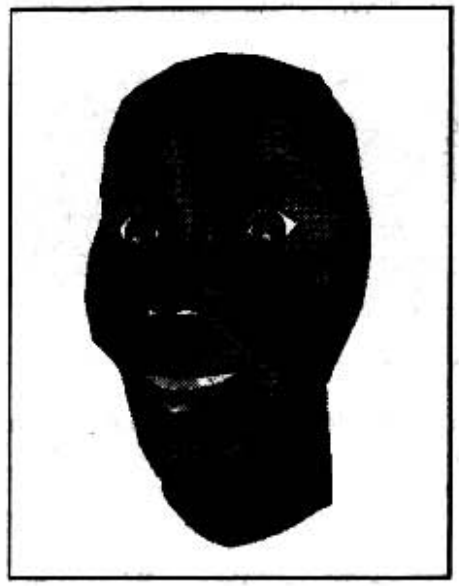

Figure 1.1: $3D$ Shaded facial model (rendered using 'fax' software)

1.2.3 Parameterized Face Model

The face parameters, as suggested by Parke, can be divided to two different sets of parameters: structure parameters and expression parameters:

1. **Structure Parameters** This set of parameters should distinguish from one person to another. The ideal set is unknown but Parke suggests a set that allows a wide variety of facial conformation within an implied limit. Those parameters include: aspect ratio of the face (height to width), color, scales of several face parts, shape and position of features inside the face, etc.

2. **Expression Parameters** Most of the expression parameters deals with the eyes and mouth. Useful expression parameters for the mouth include jaw rotation which controls mouth opening, width of the mouth, smile, position of lips, thickness of lips and the position of the mouth

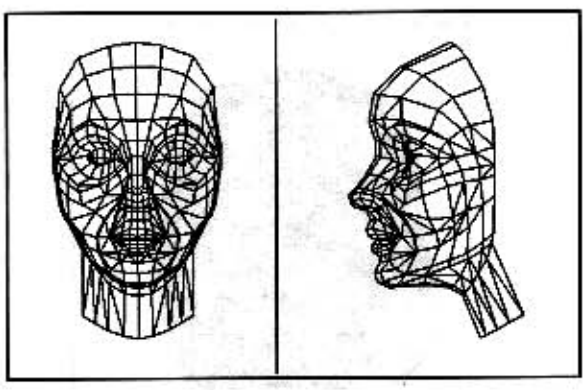

Figure 1.2: Front and side views of the wire-frame face model.

corners.

The model includes a wire-frame mask made of vertices and topology that describes the connections between those vertices. The topology creates a connected networks of polygons constructing the surfaces of the face (see figure 1.2). When the parameters are changed, the vertices change their three dimensional position but the topology remains unchanged.

In order to make the resulting synthesized face more realistic, Terzopoulos has expanded Parke's model into a hierarchical model of the face that provides natural control parameters and is efficient enough to run at interactive rate [16]. His model is decomposed into six levels of abstraction, which enclose specialized knowledge about the psychology of human facial expressions, the anatomy of facial muscle structures, the histology and biomechanics of facial tissue and facial skeleton geometry and kinematics. In short, the six levels are:

1. *Expression* : like anger, happy, etc.

2. *Control* : A subset of the FACS.

3. *Muscles* : The basic actuation mechanism of the model, as in a real face.

4. *Physics* : Physical approximation to human facial tissue, which is a lattice of point masses connected by nonlinear elastic springs.

5. *Geometry* : Nonuniform mesh of polyhedral elements whose sizes depend on the curvature of the neutral face.

6. *Images* : Graphical output.

Chapter 2

Model Adjustment

2.1 Generic Face Model

The face model used in this work (figure 2.1) is initially a symmetric model. It is composed out of 287 vertices and 256 edges for each side of the face. The vertices are divided into 4 different face components: flesh, lips, eyelash and teeth. The information is stored in 4 seperate files:

1. First vertices set, including almost all the model's vertices.
2. Second vertices set, including special values for specific vertices that needs to interpolated between two different positions.
3. Topology of the face, including all the edges between the indexed vertices (e.g. mouth corners).
4. The parameterization values. For each of the 62 existing parameters, there are range values (a minimum and a maximum values) and an initial value.

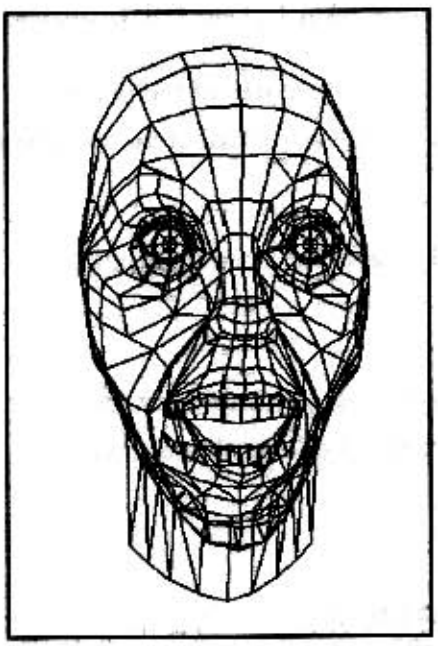

Figure 2.1: Wire-frame facial model.

The original model, with the default parameters, is being kept in some data structure, untouched. Whenever one of the parameters is altered, a new location is calculated for each of the three-dimensional vertices of the model, giving the new desired face model.

Five types of operations determine vertex position from the parameter values, as suggested by Parke [12]:

1. *Procedural construction* is used to model the eyes. Using the input parameters for the eyes like eyeball, iris and pupil size, the polygons that representing the eyes are procedurally generates.

2. *Interpolation* used for portions of the face that change shape, like forehead and the mouth. These operations use the values in the second vertices file, mentioned above.

3. *Jaw rotation* used to open the mouth around a jaw pivot axis.

4. *Scaling* controls the relative size of facial features. These operations are used for the first model adjustment to the given front and side image of the actor.

5. *Position offsets* controls the nose length, mouth corners, lip raising and lowering, etc.

For this work, several new parameters were added to the original model (like lips thickness), and other's inplementation were fixed or changed to meet the requiremts.

2.2 Initialization

The same initial generic face model is used for all actors. In order to modify the generic face model to fit a specific face, some additional information is needed. The generic model has to be translated, scaled and stretched to fit the given actor's face, from its initial position and settings (as shown in figure 2.2). In order to do so, two sets of points describing several predefined feature points on a front and a side view of the actor's face are taken as an input. Those points (see figure 2.3) were chosen to give the best information about several features and their sizes, or distances between them. Using these marks, values like forehead to chin distance, chin to nose distance and the head's width can be calculated and then compared with the real measurements on the $2D$ projection of the generic model on the image plane.

The distances on the model are measured using a world-to-screen conversion procedure that supplies the exact location of a given $3D$ coordinate on the image plane. For each feature point on the actor's images there is an appropriate vertex on the $3D$ model that its projection on the image can

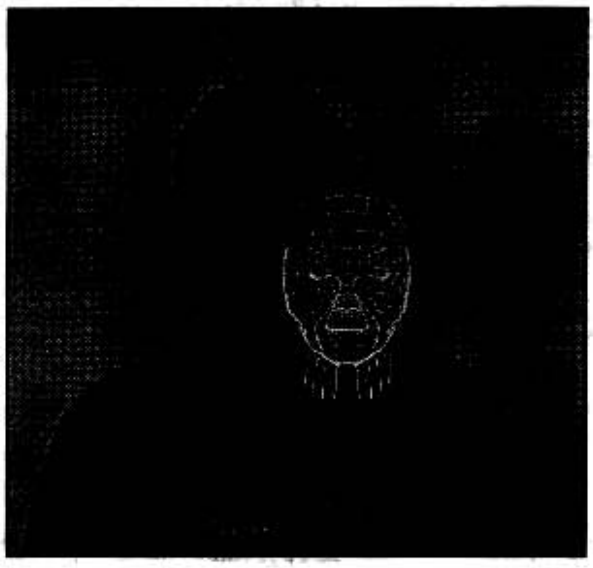

Figure 2.2: Initial settings of the face model in respect to the actor.

be found. Once the distance on the model is measured, it is compared to the distance calculated from the marked points and the model is translated, streached and scaled, using the appropriate model parameters.

Unfortunately, these steps are not enough. After applying all the necessary scaling and stretching, the model still needs to be adjusted much more accurately. Although the main features now fall in place (see figures 2.4), the model doesn't really fit on the actor's face. To fix that, a refinement stage is needed.

2.3 Refinement

The final refinement is composed of the following steps:

1. External vertices on the model are streached toward the edge of the head, in both the front and side views images.

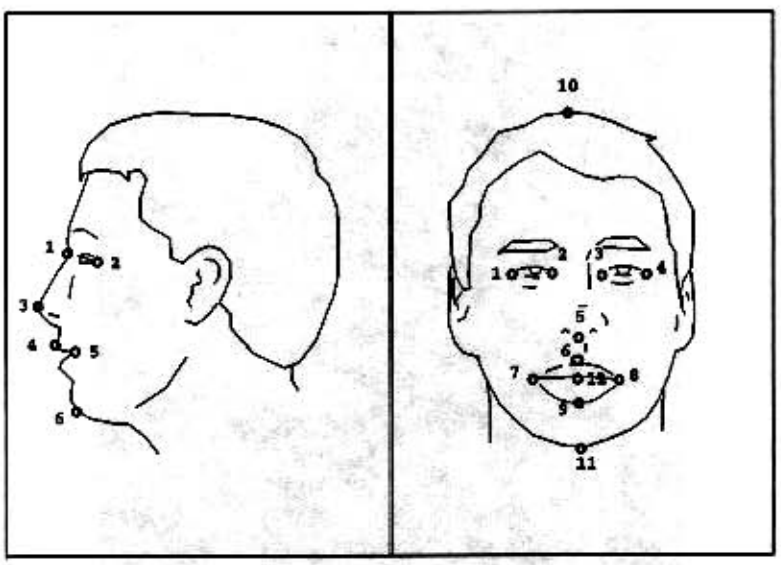

Figure 2.3: Feature points on the actor's face. These points are used for fitting the initial generic model to the actor's face in the image.

2. The mouth in the model is gently adjusted to the actor's mouth, using its specific parameters.

3. The rest of the vertices in the face model are modified according to the small set of vertices that were modified in the previous steps.

For the first two steps the Prewitt operator is used in order to detect edges in the image and adjust the $3D$ model to touch them. In the third step a $3D$ extension of a $2D$ morphing technique, explained in the following sections, is used. See figures 2.5 and 2.6 for this stage's result.

2.3.1 Edge Detection

A simple edge detector, called Prewitt operator [9] is used to locate the specific edges between the head of the actor and the background, in both the

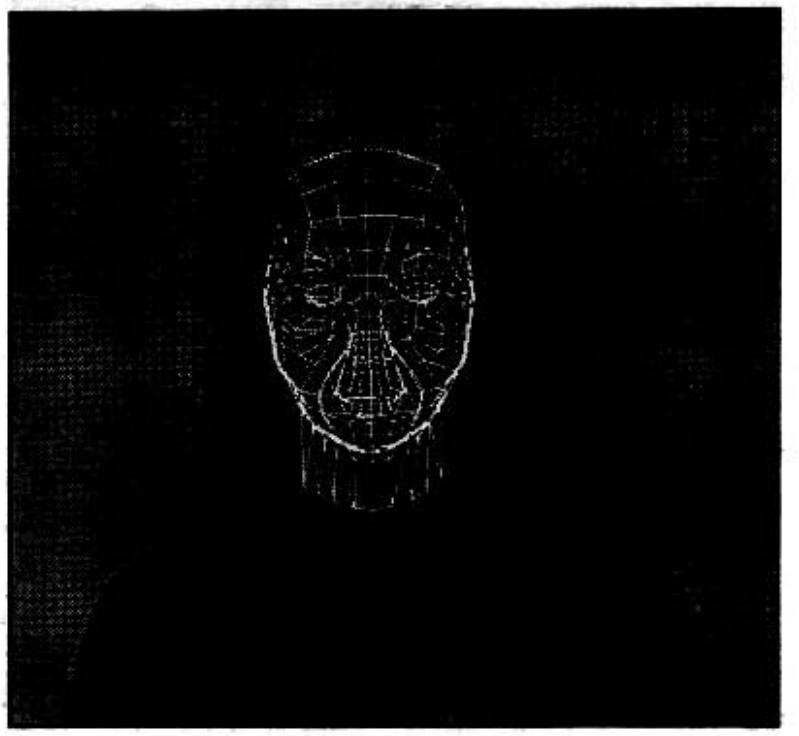

Figure 2.4: First fit of the model to the actor's face. Note that the main face features are in place.

front and the side views of the actor. For each direction there is a 3×3 mask, giving 4 different masks, (two of them are shown in figure 2.7). A search is made for the maximum correlation of this mask with the image, in a limited region around the projection of each external vertex.

In each vertex, the search is being carried out in the direction of the $2D$ projection of its $3D$ normal, as demonstrated in figure 2.8. For each pixel $p = (x, y)$ along this direction we calculate:

$$\max_{Y \in Y_1,...,Y_4} correlation(p, Y)$$

where $Y_1, ..., Y_4$ are the four Prewitt operators.

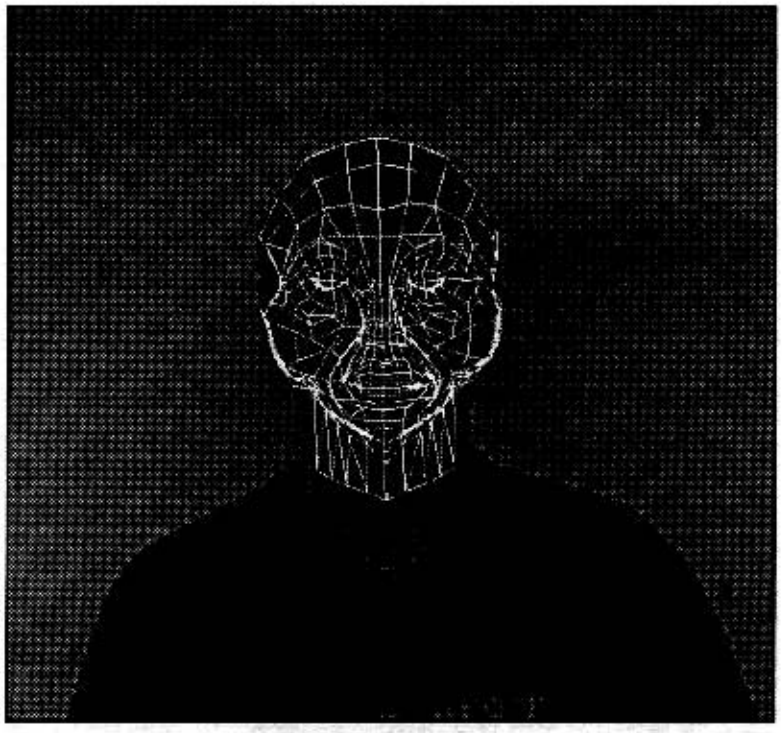

Figure 2.5: Final adjustmement of the model to the actor's face. Outer vertices were pulled towards the face edges, and mouth was asjusted to fit the actor's mouth.

2.3.2 Adjusting the mouth

Five out of the twelve feature points on the front view of the actore are located around the mouth, as shown in figure 2.3, because for dubbing purposes this is an important area. Again, the Perwitt edge detector is used to find edges between the darker area between the lips and the lips themselves. The search is done along the line between the points inside the mouth and those above and under the lips. These new points inside the lips, together with the points above and under the lips, gives an approximation for the thickness of the lips.

The four points around the mouth determine the exact jaw rotation and

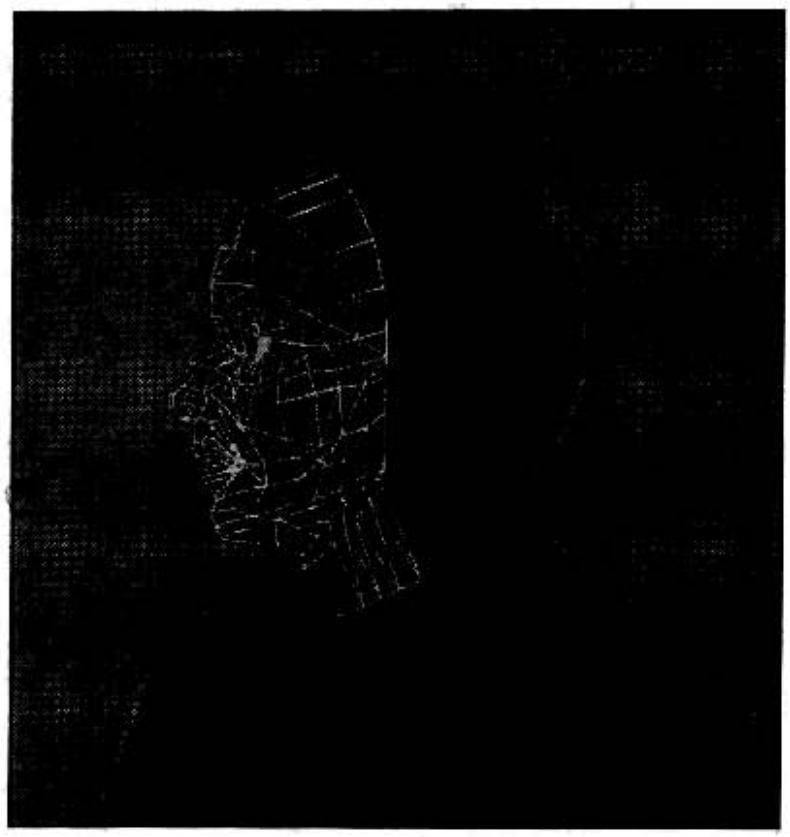

Figure 2.6: Final adjustment of the model on side view of the actor.

$$N: \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \qquad NW: \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -1 \end{bmatrix}$$

Figure 2.7: Two of the four distinct Perwitt edge detector.

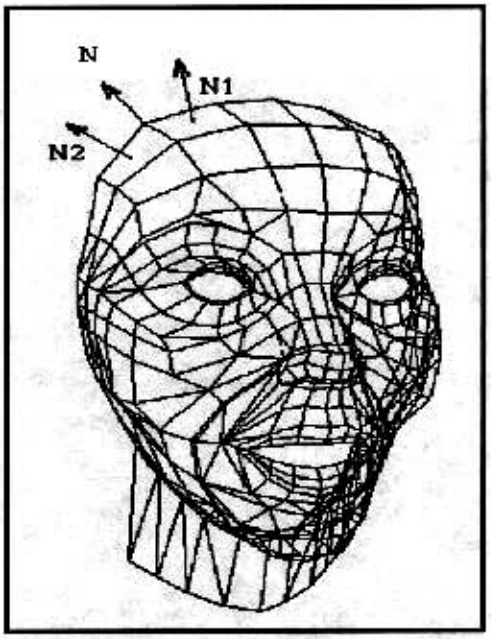

Figure 2.8: The $3D$ normal of a vertex, projected on the image plane.

mouth width, in a similar way.

2.3.3 Adjusting the Rest of the Vertices

Adjusting the mouth and the external points is insufficient, since the face model has now lost its proportions and symmetry. The rest of the vertices must now be moved with respect to the already adjusted vertices. This is done using a $3D$ generalization of a $2D$ morphing technique suggested in [3].

2.3.4 Morphing

Morphing is a new, but well known, computer graphics technique for smoothly changing from one shape to another. There are many morphing methods, most of them are based on one of three main methods which are

covered in [7] and [18].

$2D$ morphing In a previous work, a forth, new morphing method, which was presented by Beier and Neely [3] was used in order to do a similar task, but in easier conditions: all feature points are manually inserted by the animator and $2D$ morphing is performed in the mouth area, using the information taken (again, manually) from the translator's mouth. The morphing technique used there is called **field morphing**. The field morphing algorithm is described in Appendix A.

$3D$ morphing In order to adjust the rest of the model's vertices, the $2D$ field morphing algorithm described in Appendix A was generalized to work in $3D$. By combining the results of the $2D$ morphing of the both given views, the new $3D$ position of each vertex in the model can be determined. As suggested in [14], the $2D$ position $(x = x_f, y = y_f)$ of each vertex in the model is determined from the front view and $(z = z_s, y = y_s)$ from the side view. Then, the $3D$ position (x, y, z) is calculated from each vertex in the model by combining them as follows

$$\begin{aligned} x &= x_f \\ y &= \frac{(y_f + y_s)}{2} \\ z &= z_s \end{aligned} \tag{2.1}$$

All translations and scaling differences between the two views are taken into account. Several vertices are crucial to be taken from one specific view (like the lips areas in the front view, or the nose tip in the side view), so the information from the other view is disregarded.

Another major difference between the $2D$ field morphing and its $3D$ generalization is that while the $2D$ algorithm uses reverse mapping (see Appendix A), the $3D$ algorithm must use forward mapping instead. That is because the idea here is morphing $3D$ discrete vertices and not $2D$ pixel values on an image. While in the $2D$ case the resulting image can be scanned and a new value can be calculated for each pixel, in the $3D$ model there is no simple way to scan all the $3D$ region in which the model lies in. In addition, there is also no need to fill the entire $3D$ space with new vertices.

Chapter 3

Tracking

3.1 Rigid Vs. Nonrigid Motion

Given the location of corresponding feature points in two image frames, the $3D$ motion of the face has to be computed. This can be divided into two separate steps. First, estimation of the global head motion: using the assumption of rigid head motion (i.e no deformation of the head), this motion can be determined by solving a linear set of equations. The estimation of the face expression, which includes a nonrigid motion of parts of the face, is then estimated by a steepest descent minimization, to find the appropriate expression parameters that track the lips between the two frames.

3.2 Motion Model

Li et. al. [6] models the face motion as a nonrigid motion described by an affine mapping. According to Li et. al., the mapping of a point s, where $s = (x, y, z)$ to the point $s' = (x', y', z')$, which is the corresponding point in

the consecutive frame:

$$s' = R_s + T + D_s \tag{3.1}$$

where R is the rotation matrix of the face model, T is the translation matrix and D_s is the deformation matrix that corresponds to deformation caused by facial expressions.

Equation 3.1 will be solved in two steps. First, by finding the rigid motion - global head rotation and translation, and then minimizing the result for the scale factor, which is not included in the equations. After computing the rigid motion, the nonrigid facial expressions are computed.

Facial expressions are described by a 'black box' that gets the expression parameters ϕ as an input, and returns the deformation matrix D_s. The expression parameters are found by a steepest descent method to minimize the correlation function between the destination image and the altered texture-mapped model.

3.2.1 3D Motion Estimation

The rotation matrix R can be specified as three successive rotations around the $x-, y-$, and $z-axis$, by angles θ, ψ, and χ respectively, and be written as a product of these three rotations

$$R = R_z R_y R_x =$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos_\theta & -\sin_\theta & 0 \\ 0 & \sin_\theta & \cos_\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos_\psi & 0 & \sin_\psi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin_\psi & 0 & \cos_\psi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos_\chi & -\sin_\chi & 0 & 0 \\ \sin_\chi & \cos_\chi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} =$$

$$T = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

S is the scaling matrix

$$S = \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

and P is the orthographic projection matrix

$$P = \begin{bmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{-2}{f-n} & \frac{-2}{f-n} - \frac{f+n}{f-n} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

where l,r,t,b are the left, right, top and bottom corners of the view port respectively, and n and f are the near and far clipping planes. This describes the region of the $3D$ world that falls into the image plane.

The orthographic projection matrix is invertable. Therefore, P^{-1} is

$$P^{-1} = \begin{bmatrix} \frac{r-l}{2} & 0 & 0 & \frac{r+l}{2} \\ 0 & \frac{t-b}{2} & 0 & \frac{t+b}{2} \\ 0 & 0 & \frac{f-n}{-2} & \frac{f-n}{-2}\frac{f+n}{2} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} p_{11} & 0 & 0 & p_{14} \\ 0 & p_{22} & 0 & p_{24} \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8}$$

where $P_{i,j}$ are known. Multipling P^{-1} on the left side of equation (3.4) gives

$$STRV \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P^{-1} \cdot \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{3.9}$$

Since V is known, the product of V and the vertex $(X, Y, Z, 1)^T$ can be replaced with a new vertex $(X', Y', Z', 1)$

$$V \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} \tag{3.10}$$

S is first guessed to be the identity matrix I.

Combining the translation and rotation into one matrix,

$$M = TR \tag{3.11}$$

equation (3.9) becomes

$$M \cdot \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = P^{-1} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{3.12}$$

or more specifically

$$\begin{bmatrix} 1 & -\chi & \psi & T_x \\ \chi & 1 & -\theta & T_y \\ -\psi & \theta & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & 0 & 0 & p_{14} \\ 0 & p_{22} & 0 & p_{24} \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \tag{3.13}$$

where θ, ψ and χ are the rotation angles around the $x-, y-$ and $z-axis$ respectively, and T_x and T_y are the translations in the $x-$ and $y-axis$, respectively.

Opening equation (3.13) leads to

$$\begin{bmatrix} X' - Y'\chi + Z'\psi + T_x \\ X'\chi + Y' - Z'\theta + T_y \\ -X'psi + Y'\theta + Z' \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11}u + P_{14} \\ p_{22}v + P_{24} \\ p_{33}w + P_{34} \\ 1 \end{bmatrix} \tag{3.14}$$

u and v are known, but w is unknow. Since all the unknowns apear in the first two components of these vectors, the last two components will be disregarded. The matrix form of equation (3.14) becomes

$$\begin{bmatrix} 0 & Z' & -Y' & 1 & 0 \\ -Z' & 0 & X' & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \chi \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} p_{11}u + P_{14} - X' \\ p_{22}v + P_{24} - Y' \end{bmatrix} \tag{3.15}$$

At least three vertices are needed to solve equation (3.15). In order to reduce noise and make the result more robust, $n > 3$ points were chosen. In this case n was chosen to be 12. Combining n such linear equations for n points on the model, gives the following equation system

$$\begin{bmatrix} 0 & Z'_1 & -Y'_1 & 1 & 0 \\ -Z'_1 & 0 & X'_1 & 0 & 1 \\ 0 & Z'_2 & -Y'_2 & 1 & 0 \\ -Z'_2 & 0 & X'_2 & 0 & 1 \\ & & \cdot & & \\ & & \cdot & & \\ & & \cdot & & \\ 0 & Z'_n & -Y'_n & 1 & 0 \\ -Z'_n & 0 & X'_n & 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \psi \\ \chi \\ T_x \\ T_y \end{bmatrix} = \begin{bmatrix} p_{11}u_1 + P_{14} - X'_1 \\ p_{22}v_1 + P_{24} - Y'_1 \\ p_{11}u_2 + P_{14} - X'_2 \\ p_{22}v_2 + P_{24} - Y'_2 \\ \cdot \\ \cdot \\ \cdot \\ p_{11}u_n + P_{14} - X'_n \\ p_{22}v_n + P_{24} - Y'_n \end{bmatrix} \tag{3.16}$$

This linear equation system can be solved using pseudo-inverse: Writing

equation 3.16 as

$$AB = C$$

the pseudo-inverse is

$$B^* = (A^T A)^{-1} A^T C$$

This gives the required estimation vector for the $3D$ rotation and $2D$ translation of the model. S was chosen to be equal to the identity matrix I, since otherwise this equation system is not linear. In order to find the correct scaling factor, these equations were solved for different scaling values around the current scaling factor, and the result was minimized for that factor.

Choosing the Points

For the global head transformation, twelve feature points were chosen initially from the model, to be tracked throughout the entire video sequence. These feature points are located on vertices which describe a "trackable" features in the face (e.g. eye corner, nose drill). The vertices are divided into two sets of six vertices, one for each side of the face model. Since only three vertices are needed to determine the transformation, there is a big redundancy in vertices. Therefore, vertices with a weak correspondance can be thrown out of the equation.

Since there is no good way to evaluate the correspondance of a certain point, the alternative way is to through out the points that

Six specific vertices were chosen on each side of the model in order to use their $2D$ projection as the points to be tracked. Having the model is a real advantage since the location of certain trackable feature on the face, are known. At the beginning of every tracking phase, the face of the actor exactly matches the $3D$ model, and therefore the location of each and every face feature is known. Those 6×2 points were used in two iterations in

order to find the best 8 in the next frame. First, the model transformation is solved and then, by applying this transformation, the 8 best fit points are found and the system is being solved again with this subset of points. This disposes of correlation errors occuring due to noise or hidden features.

3.2.2 Expression Estimation

Unlike the rigid global motion of the head, the expression estimation can not be determined analitically, since there is a high dependancy between the different parameters. Thus, a numeric algorithm is used. The appropriate area around the mouth in both the succesive frame (target) in the video and the texture-mapping of the current frame on the face model, are compared. A steepest descent algorithm is used on the correlation function between these two images in order to estimate the lip motion.

Eight parameters acting on the mouth region are used: jaw rotation, mouth width, upper and lower lip opening, upper and lower lip thickness, and the exact left and right mouth corners.

The steepest descent algorithm details are described in appendix appendix:steepest.

Chapter 4

Producing the New Movie

This final phase combines the tracking results of both the original and the translator's movies, in order to synthesize the new movie. This movie will mostly be formed out of the original movie. The face of the actor in this movie, on the other hand, will actually be just a texture-mapped face on a three-dimentional model, synthesized to fit the lips of the translator at that particular point. The following sections describe this final phase.

4.1 Model Fitting

At each frame, the exact orientation of the model (rotation, translation and scaling) were saved in the previous phases, together with all the neccesary lips information, using the AU parameterization. Those parameters are now loaded, and applied to the model. Now, the model is located exactly on the actor's face.

4.2 Texture Mapping

When the model is in place, the current texture of the face is mapped on the $3D$ model. This synthesized image looks exactly like the original frame, no matter what the model is.

4.3 model Modification

When the model is fitted accurately to the image, the final step can be taken: the appropriate model parameters are altered according to the saved parameters from the translators movie. These paramters includes only the lips, jaw and cheeks related paramteres and not the head orientation. The actor's face which is texture-mapped on the model is modified together with the model, creating a new, synthesized frame, in which the head is in the same orientation but the lips match those of the translator. The new frame is now saved to the new movie.

It has been found experimentaly that the results are better for fewer lip parameters. Namely, There is need to use all the information gathered in the tracking phase in order to produce a photo-realistic image. Only the jaw rotation, lips width and upper and lower lips opening parameters are required. The smile parameter, together with the lip corners location and lip thickness should be dropped in order to improve the results.

Chapter 5

Applications

There is no limits to the possible applications of this work. The original goal was to match lips movement in a dubbed motion picture. This goal is probably too hard to achieve, since motion picture scenes are usually very complex, but other, simpler goals, like dubbing of commercials, can be easily achieved. In the commercial world, there are many products sold around the world. By using this work, one can take the original video commercial and produce a multi-lingual commercial, and save the neccesity to produce a new video for every language it is aimed for. Money and time are saved by replacing the production expenses with one animator working on one computer station.

Another idea is editing a movie, by altering existing scenes without having to re-shoot it again. The actor can only give the desired new scentance and his lips will be changed in the movie itself. This application has a big advantage since the lips in both movies are of the same person, probably giving better results.

Entertainment is anther field in which this work can used. For example, in musical video clips, which are becomming more and more sophisticated

today. Or putting some desired words in the mouth of a politician, etc.

Chapter 6

Implementation and Results

This application was implemented on a Silicon Grsphaics machine. The code was written in C language, the GUI used is Motif, and the graphical library is SGI's GL. The face model, including it's full parameterization is a freeware called 'fax', written by a group of Andrew Marriott, from the School of Computer Science, Curtin University of Technology in Bentley, Western Australia. Without these sources this work would have never been completed, and for that we express our deep gratitude.

6.1 The Application

We implemented a full software package for tracking and synthesizing the new movie. The application comprizes of two main programs. The first program is 'face', an automatic human face tracking program, based on the $3D$ parameterized face model. This program is a full interactive program that allows the user (animator) to manually adjust all the face parameters, and to activate the automatic tracking of both global head movements and lip deformation. The main panel (figure 6.1) controls the loaded movie, including playback and step-by-step frame control. Also, the animator can

control the various adjustements of the initial model to the actor's face, in both the front and side views of the actor. Another usefull button allows the user to apply texture-mapping of the actor's face in the $3D$ model.

A complete face control panel (figure 6.2) is supplied, for manually playing with the model, which is optionally texture-mapped.

A third panel is the tracking panel (figure 6.3) that allows an automatic tracking of both global head movement and lips, or alternatively, track each of them separately.

A zoom around the mooth area is also viewed, to ease the animator's work (figure 6.4).

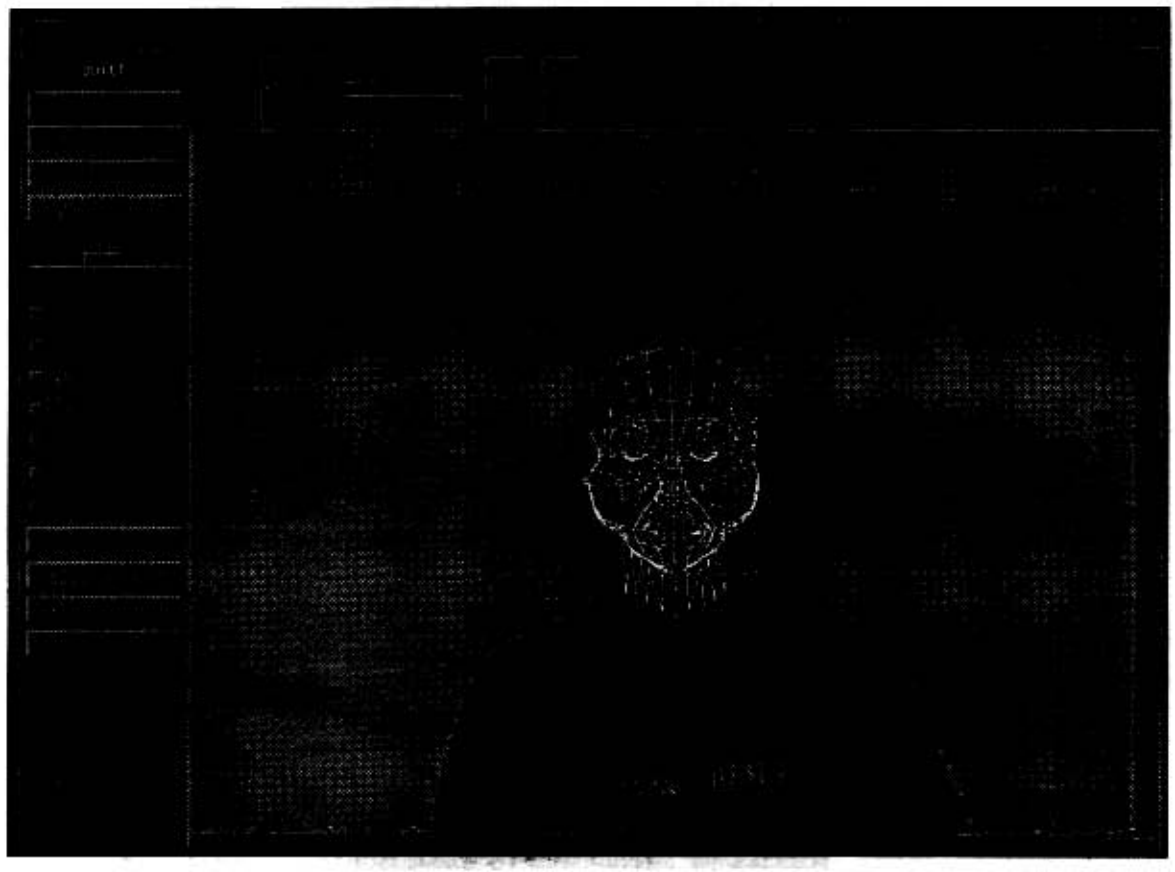

Figure 6.1: Main panel of 'face' program - an automatic human face tracking system, with interactive mode.

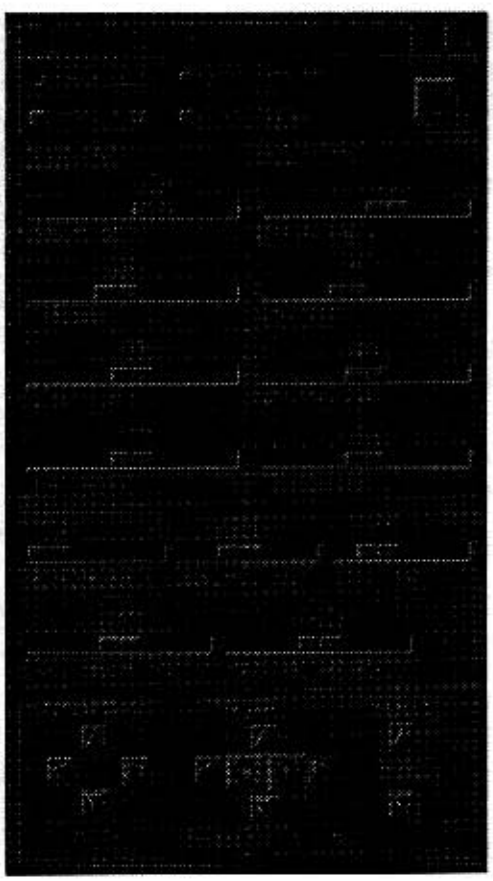

Figure 6.2: Face model control panel of the 'face' program.

Figure 6.3: Tracking panel of 'face' program. Both global head and lip movements can be automatically tracked.

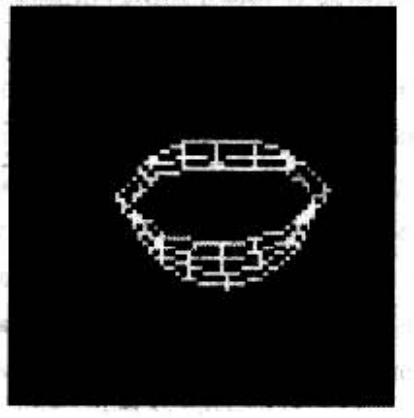

Figure 6.4: Zoom aroud the mouth, to ease the animator's work.

Appendix A

2*D* Morphing

A.1 Morphing techniques

The morphing process consists of warping two images into the same shape, and then cross-dissolving the resulting images. Cross-dissolving is usually a very simple task; the major problem is how to warp an image. All the known techniques of digital image warping are covered in Wolberg's book [18]. We describe a new method which is taken from [3], and is used by Pacific Data Images company since 1990. Under the specific face situation, the original algorithm had to be latered: instead of taking two images and warping them into an in-between image, the source image was taken and changed into the destination image, using information from a third image; in the dubbing case, the mouth of an actor in each frame of the movie was warped to fit the mouth of the translator in the appropriate frame in the translation movie. Scaling and rotation from the translator to the actor must take place first, but once this is achieved, the morphing is very similar to the one described here.

A.1.1 Reverse Mapping

The best way to perform a warp is by reverse mapping. Reverse mapping means scanning the destination image pixel by pixel and sampling the correct pixel from the source image. The most important feature of this method is that every pixel in the destination image is set to an appropriate value, while in the forward mapping. On the other hand, the source image is being scanned and each pixel is copied to the appropriate place in the destination. That way, some pixels in the destination might not get filled and would have to be interpolated.

A.1.2 The Algorithm

This method match directed lines (vectors). Lines are supplied for both the actor's and the translator's face, where each line on the actor's face has a corresponding line on the translation's face. Each pair of corresponding lines define a coordinate mapping from the destination image pixel coordinate X to the source image pixel X', where PQ is the line in the destination image and $P'Q'$ is the line in the source image (see figure A.1).

If there is only one line on each image, the following is performed: for each pixel in the destination, calculate two values: u, which is the position along the line of the projection of the point X on that line ($u \in (0,1)$ if the pixel is between P and Q, and outside the line otherwise), and v, which is the perpendicular distance in pixels from X to the line. The single line algorithm is

For each pixel X in the destination image:
find the corresponding $\mathbf{u}, \mathbf{v}$
find $\mathbf{X'}$ in the source image for that $\mathbf{u}, \mathbf{v}$
destination$(X) \leftarrow$ source$(\mathbf{X'})$

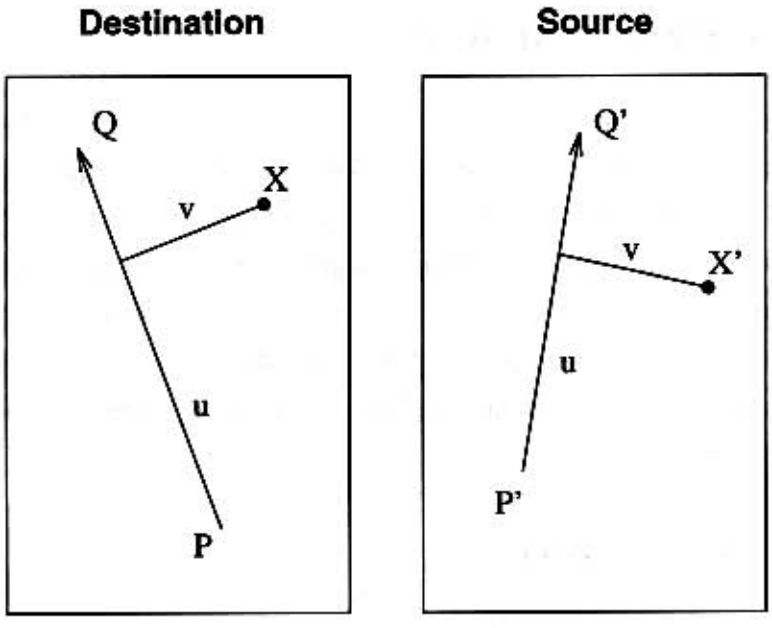

Figure A.1: Single line pair.

where:

$$u = \frac{(X - P) \cdot (Q - P)}{\|Q - P\|^2} \tag{A.1}$$

$$v = \frac{(X - P) \cdot Perp(Q - P)}{\|Q - P\|} \tag{A.2}$$

$$X' = P' + u \cdot (Q' - P') + \frac{V \cdot Perp(Q' - P')}{\|Q' - P'\|} \tag{A.3}$$

Perp(V) is the vector perpendicular and in the same length as V.

The algorithm transforms each pixel coordinate by a rotation, translation and scaling. Notice that all the pixel that are along the line in the source image are copied on top of the line in the destination image.

In the case of multiple pairs of lines we have more complex transformation. In this case, every line will influence the final value of the destination pair,

with a specific weight factor. The weight factor should be such that the line will have greater influence on nearby pixels and less on distant pixels. Additionally the weight factor should be such that the influence is greater for longer lines. The following formula is used

$$weight = \left(\frac{length^p}{a + dist}\right)^b$$

length is the length of the line, *dist* is the distance from a pixel to that line, and a, b, and p are constants that can change the affect of the lines.

When a is very close to zero, then the affect on a pixel which is on the line is infinite. Increasing a results in a smoother warping. The variable b determines how the relative strength of different lines fall off with distance. The value of p determines how the length of the line affects its weight. If p equals 1 then longer lines have a greater relative weight than shorter lines.

The multiple line algorithm is as follows

for each pixel $\mathbf{X}$ in the destination
  $\mathbf{DSUM} \leftarrow (0,0)$
  $\mathbf{weightsum} \leftarrow 0$
  for each line $\mathbf{P_iQ_i}$
    calculate $\mathbf{u}, \mathbf{v}$ based on on $\mathbf{P_iQ_i}$
    calculate $\mathbf{X_i'}$ based on $\mathbf{u}, \mathbf{v}$ and $\mathbf{P_i'Q_i'}$
    calculate displacement $\mathbf{D_i} \leftarrow \mathbf{X_i'} - \mathbf{X_i}$ for this line.
    $\mathbf{dist} \leftarrow$ shortest distance from $\mathbf{X}$ to $\mathbf{P_iQ_i}$
    $\mathbf{weight} \leftarrow (\mathbf{length^p}/(\mathbf{a} + \mathbf{dist}))^\mathbf{b}$
    $\mathbf{DSUM} \leftarrow \mathbf{DSUM} + \mathbf{D_i} \cdot \mathbf{weight}$
    $\mathbf{weightsum} \leftarrow \mathbf{weightsum} + \mathbf{weight}$
    $\mathbf{X'} \leftarrow \mathbf{X} + \mathbf{DSUM}/\mathbf{weightsum}$
  destinationImage($\mathbf{X}$) $\leftarrow$ sourceImage($\mathbf{X'}$)

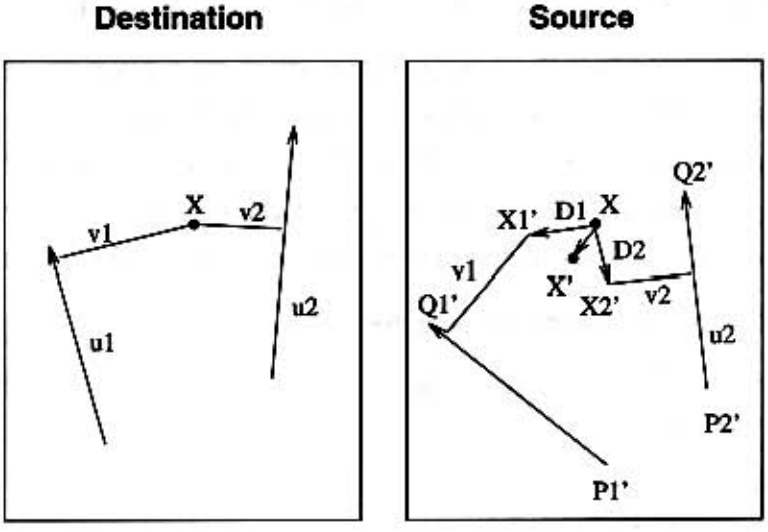

Figure A.2: Multiple line pairs.

Appendix B

The Steepest Descent Minimization Method

Given a function $f = f(x_1, x_2, ..., x_n)$, and $f'(x_i)$ for all $i = 1, ..., n$, which are the partial derivatives of f with respect to x_i the steepest descent minimization method gives us a straight forward methood to minimize f for all x_is. The general algorithm is

start with some vector $(\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_n})$
STAGE 1:
for each $\mathbf{x_i}$ do
    $\mathbf{current_f} \leftarrow f(x_1, x_2, ..., x_n)$
    $\mathbf{min_f} \leftarrow$ MAX_VALUE
    if $f'(x_i) > 0$ then $\epsilon \leftarrow -\epsilon_{x_i}$
    else, $\epsilon \leftarrow \epsilon_{x_i}$
    While $\mathbf{current_f} < \mathbf{min_f}$ do
        $\mathbf{min_f} \leftarrow \mathbf{current_f}$
        $\mathbf{x_i} \leftarrow \mathbf{x_i} + \epsilon$
        $\mathbf{current_f} \leftarrow f(x_1, x_2, ..., x_n)$
repeat STAGE 1, until f is minimized.

$$\begin{bmatrix} \cos_\psi \cos_\chi & \sin_\theta \sin_\psi \cos_\chi - \cos_\theta \sin_\chi & \cos_\theta \sin_\psi \cos_\chi + \sin_\theta \sin_\chi & 0 \\ \cos_\psi \sin_\chi & \sin_\theta \sin_\psi \sin_\chi + \cos_\theta \cos_\chi & \cos_\theta \sin_\psi \sin_\chi - \sin_\theta \cos_\chi & 0 \\ -\sin_\psi & \sin_\theta \cos_\psi & \cos_\theta \cos_\chi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.2)$$

Assuming a high frame rate, it can be assumed that the head rotation is very small between two consecutive frames [8]. Thus, $\sin_x \approx x$ is approximated for a small angle x and $\cos_x \approx 1$, the linearized version of the matrix R in equation (3.2) becomes

$$R \approx \begin{bmatrix} 1 & -\chi & \psi & 0 \\ \chi & 1 & -\theta & 0 \\ -\psi & \theta & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.3)$$

where θ is the rotation angle around the $x-axis$, ψ is the rotation angle around the $y-axis$, and χ is the rotation angle around the $z-axis$.

By substituting the original rotation matrix with the one in equation (3.3), the global head's rotation and translation can be described by the following eqaution, which are given in homogenious coordinates for the vertex $(X, Y, Z, 1)$ and its image point coordinates (u, v):

$$PSTRV \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \quad (3.4)$$

where V is the matrix that describes the model orientation (the viewing matrix), R is the unknown rotation matrix of equation (3.3) and T is the unknown translation matrix that has the following form

The general algorithm was modified to fit our case. $f(x_1, x_2, ..., x_n)$ in this case is the sum of squared differences of two sub images. The minimization of the sum of squared differences is completely equivalent to correlation. In this work, the term "correlation" is used frequently. The unknown parameters $x_1, x_2, ..., x_n$ are a sub group of the face parameters, depending on what we want to solve and minimize.

The first image is the $3D$ face model, on which the face from the current image in the movie is texture-mapped. The second image is the succesive frame in the movie which is the frame we want the model to fit on. The initial vector $(x_1, x_2, ..., x_n)$ is chosen to be the parameters vector from the previous frame, assuming the changes in the those parameters are minimal.

After applying the texture mapping of the actor's face on the model, the required model parameters can be modified, and by doing that, a new, synthesized face or expression of the actor, is created. The expression is, under some limitations, very photo-realistic. Using this fact, the synthesized face and the target actor's face in the next frame are compared. The comparision is made by minimizing the sum of squared differences of the pixel values in the required regions, by using this steepest descent method.

In order to do so, several changes are required in the above algorithm. First, the partial derivative $f'(x_i)$ must be calculated by computing the correlation between the images, at any point. We chose a specific ϵ for each parameter, depending on its range. The correlation function is calculated in both $x_i + \epsilon$ and $x_i - \epsilon$ and the direction is chosen according to the higher result. If the correlation function has reached a minimum point, the process stops. Otherwise it is continued in the derivative direction until the minimum correlation is reached. We can not prove that this algorithm converges to the correct minimum, since the exact parameterization is considered unknown.