A System for Advanced Facial Animation

by

Kenneth D. Miller III

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science

and

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Author ....
ter Science
ay 28, 1996

Certified by ....
Alex (Sandy) Pentland
Head. Perceptual Computing Section, The Media Laboratory
hesis Supervisor

Certified by ....
Irfan A. Essa
Research Scientist. omputing Section, The Media Laboratory
Thesis Supervisor

Accepted by ....
R. Morgenthaler
Chairman, Department Committee on Graduate Theses

A System for Advanced Facial Animation

by
Kenneth D. Miller III

Submitted to the Department of Electrical Engineering and Computer Science
on May 28, 1996, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science
and
Master of Engineering

Abstract

This thesis describes a system for extracting three-dimensional models from two-dimensional images over the restricted domain of frontal images of human faces. A three-dimensional face model template is fit to a three-dimensional digitizations of a set of training subjects. The standard-form models allow different subjects to be treated in a uniform manner. The particular template model chosen is almost identical to one used for head tracking and expression extraction from video, allowing more accurate modelling of the subject.

The models are used to synthesize small face images with a series of discrete orientations, and then subjected to Karhunen-Loève expansion (principal component analysis), yielding one low-dimensional of coupled eigenimages and eigenmodes per orientation. Subsequently, face images within the range of rotations used to generate the eigenspaces can be projected onto the eigenspaces, yielding a coarse estimate of facial orientation, recovering missing data, and producing an approximate facial model.

Thesis Supervisor: Alex (Sandy) Pentland
Title: Head, Perceptual Computing Section, The Media Laboratory

Thesis Supervisor: Irfan A. Essa
Title: Research Scientist, Perceptual Computing Section, The Media Laboratory

Acknowledgements

I would like to thank Irfan A. Essa and Alex Pentland for their patience, perseverence, advice, and motivational influences during my thesis work, especially as time ran short and then ran out. Without their supervision, this thesis would not have been possible. I must also thank Sumit Basu for the informal chat sessions after which I finally understood what was really going on in the Karhunen-Loève expansion. Before I began working with them, I had no idea what I wanted to do. After prolonged exposure to the working environment they helpfully provided, and with more than a little prodding, I devised this model recovery system.

The Vision and Modeling group at the M.I.T. Media Laboratory provided computer resources, reference material, and critical data for my work. ViewPoint Data Laboratories provided the medium-complexity head model used as a template for facial reconstruction.

Finally, I am grateful for the emotional and financial support of my parents. There is almost no way I can possibly repay them for what they have given me.

Contents

List of Figures

Chapter 1

Introduction

1.1 The Problem

Recovery of three-dimensional structure from arbitrary, two-dimensional images is one of the most important open problems in the field of Machine Vision. Model recovery has a vast array of uses, as many as there are uses for human vision, including scene understanding, object tracking in threespace, and numerous others. Unfortunately, the general problem is a rather difficult one. Many an otherwise fine idea has stumbled over this hurdle. A stellar cast of researchers have spent decades on the problem, with significant, but limited, progress. The functional inverse of computer graphics, structure-from-image recovery seeks to extract three-dimensional information from two-dimensional images, recovering the information lost in the projection process.

While the problem seems trivial—we do it all the time, and think nothing of it—it is in fact one of the more difficult areas of Artificial Intelligence. The main computational snag is that is ill-posed: there are an infinity of mathematically valid solutions, as a two-dimensional image has three dimensions of information (x, y, and intensity), whereas a three-dimensional model has four (x, y, z, and intensity). Even if one makes relatively strict assumptions about material properties of objects in the scene, smoothness and integrability of surfaces, and so on, there are still many equally valid solutions. When such difficulties as noise and incomplete information appear, the intractability becomes still greater.

If the problem is so hard, then how does the human brain do it? The trouble is, the answer is usually "we don't know, really." The visual cortex is immense, comprising a good third of the brain, and has fantastically complex and generally unknown structure; this fact alone would seem to indicate that the problem is rather difficult. However, what *is* known is that the brain takes advantage of expectations, prior knowledge, and context clues. Much to the dismay and horror of those who thought machine vision would be easy, vast quantities of prior knowledge are required to solve the problem correctly, and even then are not completely sufficient: witness the panoply of optical illusions.

From an intuitive standpoint, most of the possible three-dimensional objects corresponding to a given two-dimensional image are completely implausible, requiring precise alignment of disjoint fragments with the viewer, or other peculiar arrangements. For example, through long experience, one knows how a chair looks like from a wide variety of angles. It is not a collection of countless tiny shards positioned and oriented very precisely to give an impression of "chairness"; instead, it is a rather simple structure, with a horizontal seat, a vertical back, and four vertical legs. In essence, when one looks at an image of a chair, one implicitly recalls every chair seen from every direction, placing strong constraints on the possible arrangements of matter that could produce that image.

It would then seem that the brain is capable of discerning to what "class" of object a particular image belongs. Within that class, there are certainly variations, but the class nonetheless greatly restricts the three-dimensional structure. The "shape space" within the class may even be low dimensional, and is certainly of lower dimension than the set of all possible objects. The structure recovery problem then reduces to determining to which object class the image belongs, and then estimating the "parameters" within its shape space.

1.2 The Goal

In this thesis we will reduce our scope to a somewhat narrow, yet still important, category of object: the human face. Granted that human interaction is of great importance, and that the face is the seat of communication, this is not a crippling restriction. In fact, the human face has sufficient complexity and subtlety to offer significant challenge on its own. Our brains are keenly aware of subtle nuances and details, indicating the importance of the face.

From prior experience, one knows that the human head has a very consistent shape: roughly spheroidal in form, with an ear on each side, a protruding nose in the front, indented eye sockets above, and a horizontal mouth below. All individuality lies within small perturbations of this basic structure.

Now, if we have an image of an arbitrary face, we immediately know that it has this same basic form, with a bit of variation of the details to produce the image. So, all we need to do now is estimate these details to recover the model. This "all" is, of course, not entirely trivial, but is indeed a great deal simpler than solving the general structure-from-image problem.

First and foremost, we must parameterize the possible variations. To do this, we use the statistical technique of *principal component analysis* developed by Karhunen in 1946 [1], Loève in 1955 [2], and Jolliffe in 1986 [3], and first applied to face images by Sirovich and Kirby in 1987 [4] and 1990 [5]. With this technique, we can extract the relatively small subspace of human faces hidden within the huge space of all possible structures. With a surprisingly small number of principal features, we can characterize a wide array of faces with little loss of information. The most significant eigenmodes tend to be obvious features such as lighting variations, skin color, facial hair, glasses, and other common details. Less significant modes tend to be rather obscure, having little intuitive meaning.

By projecting a face image onto this subspace, we can readily recover parameter values that describe the face. This alone is quite useful for numerous applications, including face recognition (as described by Turk and Pentland [6], and many others) and model-based compression of images, (described by Moghaddam in [7]). The values of the parameters form "clusters" that characterize particular faces.

The next step, then, is to correlate image features and model features with principal component analysis by joining the space of images with the space of models. Within this larger space, certain variations of pixel intensities and of vertex positions will tend to co-vary, and principal component analysis will pick out these correspondences. For example, if a dark region on the forehead tends to correspond with a protrusion in the model due to hair, then principal component analysis will make this connection. With a large enough sample of images and the corresponding models, we can, in essence, "train" the computer to recover three-dimensional structure from a two-dimensional image by determining which features are present or absent in a given image of a human face.

The next few chapters describe the system. Chapter 2 gives a non-technical overview of the basic operation of the model-recovery system, from model creation to principal component analysis, to model reconstruction. Chapter 3 gives a more in-depth description of the underpinnings of the system. Chapter 4 shows results from using the system on a training set of eight three-dimensional models and a test set of sixteen face images.

Chapter 2

Approach

2.1 Overview

In order to create a correspondence between facial images and facial models, we must first acquire some facial models. To this end, we utilize the three-dimensional data produced by a Cyberware Rapid 3D Color Digitizer [8]. This device sweeps a laser stripe across an object as it revolves around it, determining the distance and reflectance of each point. This yields a densely-sampled grid of range and color values in cylindrical coordinates.

In its pure form, the range data is unfortunately not as useful as one would like, at least for analysis purposes. While the full data set can be used for imaging, the massive quantity of points resulting from blind cylindrical-to-cartesian coordinate transformation makes it extremely unwieldy. The sampling density and the overall dimensions of the sampling grid are adjustable by the user, the lighting of subjects will vary, and the data set will have numerous missing points where the subject is nonreflective or steeply inclined relative to the beam. A more serious problem is that the subject of each scan is not necessarily in the same place or facing in the same direction. We must somehow regularize this information to eliminate these undesirable extraneous variations.

Atick, Griffin, and Redlich [9] solve the problem by using a database in which all the head models are of the same dimension, and completely ignore the texture map, using only range data. They also use an entire range image, 256×200, without attempting to reduce the complexity of the model.

To do this, we warp a generic three-dimensional human head model produced by View-Point Datalabs [10] to the shape of the Cyberware range data, producing a model with known structure. We then scale and orient the model so that the eyes, nose, and mouth lie in known positions when rendered. For a number of training faces, the models are rendered with a variety of small rotations to allow the model recovery to compensate for slight changes in orientation.

The set of face models and corresponding images are then subjected to principal-component analysis to determine the covariances of the images and the models. This will produce a relatively low-dimensional space of image features and the model features to which they correspond.

Once this feature space has been created, a face image can be projected onto the space to recover missing information. In this case, the missing information is the three-dimensional model corresponding to the given image.

2.2 Creating Models from Cyberware Data

Creating a standard face model from a Cyberware data set involves only a bit of work and is not especially computation-intensive. The idea is to use an appropriate model as a "template", deforming it into the proper shape. Instead of transforming the scan data into a three-dimensional object and matching the three-dimensional head template to it, we transform the three-dimensional template into a two-dimensional object and align it with the data.

First, corresponding features are found in both the scan data and a known template model. We have chosen as representative features the centers of the eyes, the tip of the nose, the left and right corners and center of the mouth, and the top, bottom, left, and right sides of the face. For a more articulate model, more features could be chosen. One can choose an arbitrary number of features, with more correspondences leading to progressively better alignment at the expense of requiring more effort.

In the Cyberware data, a variety of methods can be used to find the important features ranging in complexity from a manual search to "distance from feature-space" eigenfeature matching [7]. In any case, the feature points should be placed as accurately as possible to minimize undesirable variations in model vertex positions.

In the ViewPoint template model, the only truly practical method is to manually locate each of the ten feature points. It may well be possible to render an image of the model and use eigenfeature matching, but the synthetic nature of the model may make feature-finding difficult. In any case, the operation need only be done once.

The feature points are then brought into correspondence, and the two-dimensional template model is warped into the shape of the face of the subject. Within non-overlapping regions bounded by three feature points, the two-dimensional vertices of the template are linearly transformed to move them into place. At each vertex, the range and texture coordinates are extracted from the Cyberware data, generating a three-dimensional, texture-mapped model.

Once the model has been generated, it is scaled and rotated so that a properly-placed orthographic camera will produce an image with the feature positions in the same places as those of the 7500-image FERET database.

2.3 Creating Models from Images

Creating a standard face model from an image is somewhat more involved, and requires the creation of an eigenspace, a set of eigenfaces that encode image facial features and the corresponding model features.

A reasonably large training set of three-dimensional, texture-mapped face models are rendered as a series of "face chips", small images consisting of only the face, with the edges blended away to reduce boundary effects. To tolerate small changes in orientation of the faces to be reconstructed, each sythetic face chip is rendered with a different small rotation, generally in the range of a few tenths of a radian in any one direction. Since the laser-scanned face is already, in essence, "front lit" by the digitizer, no additiona lighting of the training models is necessary. A simple "decal" texture mapping will produce the best results.view.

These face chips, and the models which generate them, are combined into a group of training sets, one per orientation, each a matrix of column vectors. The vectors are normalized to zero mean, unit variance, the mean of all the normalized images is removed, and the Karhunen-Loève expansion is applied, extracting orthonormal eigenmodes of the face images and corresponding face models. The most significant modes, those accounting for the largest variances (that is, the ones with the largest eigenvalues) are saved for future use.

To reconstruct a model from an image, the image is projected onto each of the eigenmodes, measuring the "amount" of that particular eigenmode in the image. Once all the weights have been found, the eigenmodes are linearly combined to produce an image of the face, projected onto the eigenspace. Any missing data will be filled in, yielding a complete image and model. The eigenspace that minimizes the reconstruction error will be assumed to be the best orientation, and the model will be reconstructed from that eigenspace.

The model can then be texture-mapped with the image fairly easily, since the recovered model has the same scale as the training models, and the mapping between the training models and training images is known. The vertices of the model are projected into the image and the pixel positions extracted as texture coordinates (after appropriate scaling). Portions of the model steeply inclined with respect to the viewing direction will have reduced detail density, as the image pixels will be spread out over a larger surface area.

2.4 Putting the Model to Use

Since the models derived from three-dimensional data and the models derived from two-dimensional images have the same structural form as the generic template model, anything that uses the template model can also use the derived models.

For example, Basu uses an ellipsoid approximation in an optical-flow Kalman filter head-motion tracker [11]. More recently, he has switched to the same ViewPoint head model as we use here. While the generic head fits most faces reasonably well, a more accurate shape estimate would improve tracking accuracy by better accounting for pixel motion. His system uses a full-frontal view to achieve an initial "lock", determining head position and orientation. This frontal image can be transformed into the FERET image format required for model extraction, and a structure estimate derived. The model used by Basu's motion tracker and this model recovery system are very similar, both being derived from the same ViewPoint head model.

Similarly, the recovered head models can be used in a facial animation system similar to that of Lee, Terzopoulos, and Waters [12] when no three-dimensional scan is available. While the recovered models are of lower quality than full scans, they are better than no models at all.

With appropriate training subjects, models can be recovered with varying facial actuations as smiles, frowns, open and closed mouth, raised eyebrows, open and closed eyes, and so forth. This capability would be useful in an facial-animation-from-video system such as the one developed by Essa, Basu, Darrell, and Pentland [13], where it could provide rough estimates of facial expression to be refined by video tracking.

Photobook [14], an image database browser that classifies images by similarity, utilizes an enormous database of images. The FERET database consists of over 7500 images taken at a photography show, with the eyes in consistent locations. The photography booth used two light-emitting diodes arranged so as to be visible only when the eyes are in very specific locations in space. The subjects were instructed to take their picture only when they could see both lights. The eye positions are accurate to within one pixel, though nose and mouth positions are less accurate, as the subjects were free to rotate their heads, and there was little constraint on the distance to the camera (save the size of the booth). Using this database to generate models would produce 7500 three-dimensional models of moderate quality while only requiring a hundred or so Cyberware head scans for training purposes. Such a technique is thus an extremely cost-effective way to acquire three-dimensional models.

Chapter 3

Methodology

3.1 Overview

We shall go through the process in detail from start to finish, creating an eigenspace from a training set of human head scans, and ending with a reconstructed model from an image. The procedure involves transforming the Cyberware scan data into a useful, consistent format, rendering a series of images, performing principal component analysis on the images and models to create a set of orthonormal features, and finally projecting an image onto the feature vectors to reconstruct a model.

3.2 Creating Standardized Models from Cyberware Data

When a full set of Cyberware scan data is available, production of a useful head model is a relatively simple affair. The process basically reduces to aligning a template model to the face data and extracting the appropriate data points. While the procedure is conceptually simple, actual execution requires some attention to detail, and even then requires manual editing to achieve truly natural results.

3.2.1 Cyberware Data Preprocessing

Since the Cyberware 3D scanner uses a laser to scan the subject, extremely dark or strongly inclined features will prevent a successful digitization of some points, as insufficient light will return to the range sensor. The raw range data is thus full of holes which will throw off any attempt at modelling the subject. To patch up these dropouts, some form of interpolation must be used.

The interpolation algorithm used to repair the range data processes the array in "lexicograpical" order (left to right, top to bottom). As it sweeps through the range values, it can assume that the pixels directly to the left and directly above the current position are valid, as any holes in those locations will have already been filled in by the algorithm in an earlier step. Figure 3-1 shows a typical invalid pixel and its surrounding region. The interpolated value is taken as a weighted average of the two known pixels and the range of the first valid point below the current position ($r_{x,y+d_y}$), and the range of the first valid

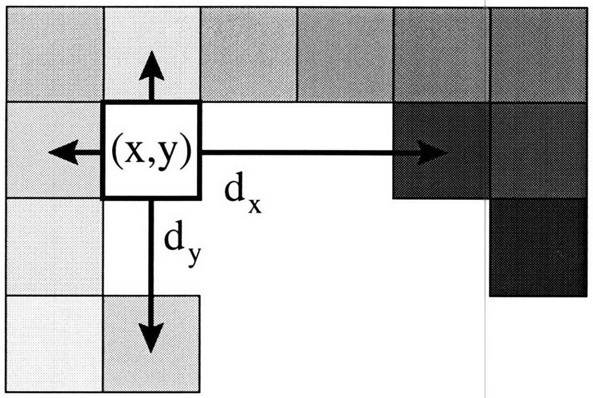

Figure 3-1: Interpolation of an invalid range by weighted average of nearby valid ranges.

point to the right ($r_{x+d_x,y}$), with each range weighted inversely proportional to distance:

$$r(x,y) = \frac{r(x-1,y) + r(x,y-1) + \frac{1}{d_x} r(x+d_x,y) + \frac{1}{d_y} r(x,y+d_y)}{2 + \frac{1}{d_x} + \frac{1}{d_y}}$$

While the results are not quit as smooth as the relaxation approach utilized by Lee, Terzopoulos, and Waters [12], this interpolation approach requires only one pass through the image, is extremely rapid for small empty regions, and produces similar results. If this approach is unsatisfactory, the results can be used as an "initial guess" to reduce the number of iterations required for relaxation.

3.2.2 ViewPoint Model Preprocessing

A three-dimensional human head model from ViewPoint Data Laboratories is used as a structural template for the face models. This model suffers from the problem that it is not just a face, but an entire head complete with neck and parts of shoulders, and is composed of several discrete parts. To simplify later analysis, the relevant region (the face) must be extracted and the separate components joined.

To this end, three clipping planes are defined, one to crop off the neck at the edge of the jaw, another to crop off the ears and back of the head, and the final one to trim the top of the head. The parameters of these clipping planes were found by trial and error, adjusting orientation and position to eliminate unwanted portions of the model. The remaining points are "uniquified" by mapping multiple points with identical (x, y, z) coordinates (within some small tolerance) onto a single point. This yields a continuous surface mesh of a bit over twelve hundred vertices, complete with surface normals. Figure 3-2 shows the resulting cropped, single-component ViewPoint model as points, lines, and as a solid shape.

3.2.3 Feature Finding

One of the more important preprocessing steps is to locate critical facial features such as the eyes, nose, mouth, and top, bottom, left, and right edges of the face. These are used to match the ViewPoint head with the Cyberware head, and later to align the Cyberware head with the Photobook image template.

The currently-defined critical feature points are, in order, the center of the left eye (as

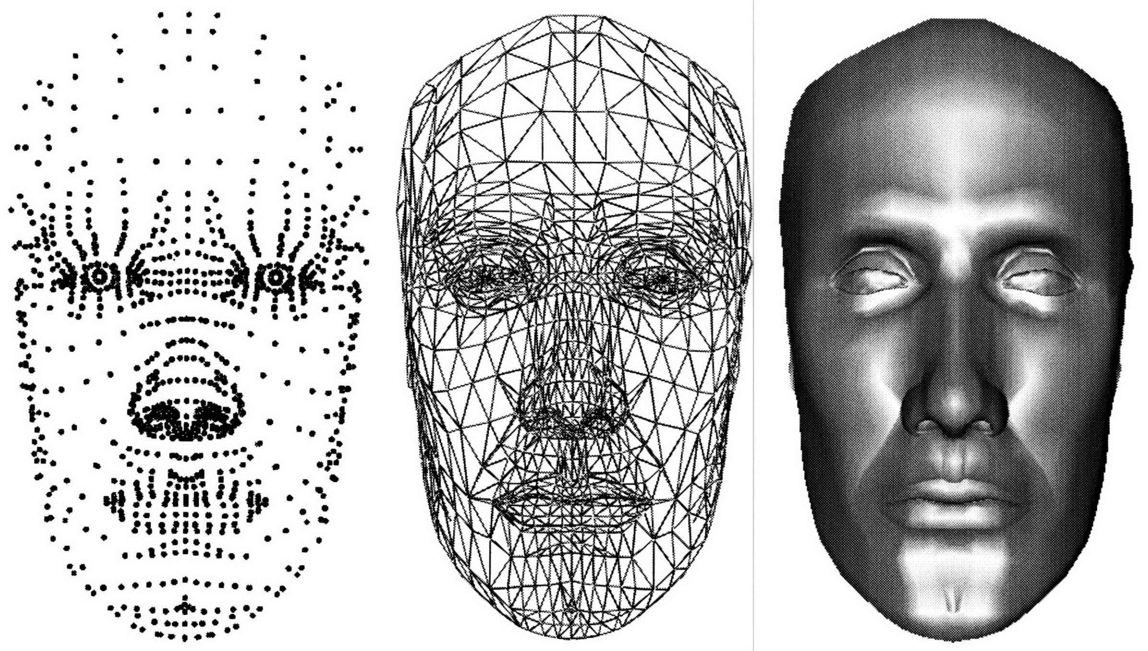

Figure 3-2: The cropped ViewPoint Datalabs head model, rendered as points, wireframe, and Phong-shaded polygons.

seen from the front), the center of the right eye (as seen from the front), the tip of the nose, the left corner of the mouth, the center of the mouth, the right corner of the mouth, the top of the forehead, the bottom of the face (where the chin meets the neck), the edge of the jawline just in front of the left ear, and the edge of the jawline just in front of the right ear. To ensure full coverage of the face, four additional points are computed from the left, right, top, and bottom points of the face to define a parallelogram (ideally a rectangle). Defining $\vec{p}_t$ as the two-dimensional point at the top of the face, $\vec{p}_b$ at the bottom, $\vec{p}_l$ at the left lide, and $\vec{p}_r$ at the right, the four corner points $\vec{p}_{tl}$, $\vec{p}_{tr}$, $\vec{p}_{bl}$, and $\vec{p}_{br}$ are:

$$\begin{aligned}
\vec{p}_{tl} &= \vec{p}_t + \frac{((\vec{p}_l - \vec{p}_t)\cdot(\vec{p}_r - \vec{p}_l))(\vec{p}_r - \vec{p}_l)}{\|\vec{p}_r - \vec{p}_l\|^2} \\
\vec{p}_{tr} &= \vec{p}_t + \frac{((\vec{p}_r - \vec{p}_t)\cdot(\vec{p}_r - \vec{p}_l))(\vec{p}_r - \vec{p}_l)}{\|\vec{p}_r - \vec{p}_l\|^2} \\
\vec{p}_{bl} &= \vec{p}_b + \frac{((\vec{p}_l - \vec{p}_b)\cdot(\vec{p}_r - \vec{p}_l))(\vec{p}_r - \vec{p}_l)}{\|\vec{p}_r - \vec{p}_l\|^2} \\
\vec{p}_{br} &= \vec{p}_b + \frac{((\vec{p}_r - \vec{p}_b)\cdot(\vec{p}_r - \vec{p}_l))(\vec{p}_r - \vec{p}_l)}{\|\vec{p}_r - \vec{p}_l\|^2}
\end{aligned}$$

For Cyberware data, an adaptation of the distance-from-feature-space technique described by Moghaddam [7] is the best way to locate the relevant points. One notable advantage of this method on Cyberware data is that structural information from the range data is available in addition to the intensity information from the texture. Since we are primarily interested in best-match position detection and not true reconstruction, we can use a relatively small number of eigenmodes (say, five to ten).

To train the feature finder, we examine a set of training faces, locate important feature positions by hand, and extract intensity and range regions centered on the feature positions.

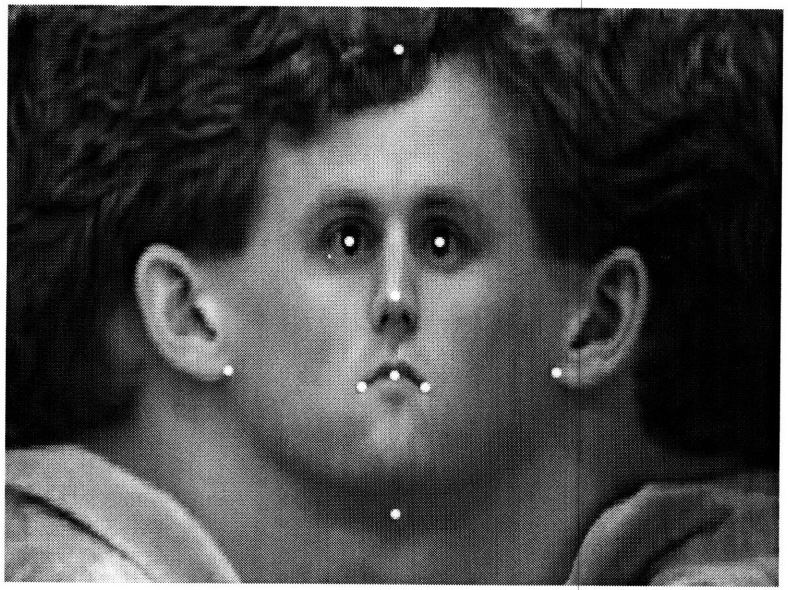

Figure 3-3: Cyberware texture data sbowing manually-extracted feature positions.

For each region, we perform principal-component analysis, saving the first few eigenmodes of both the intensity and the range images.

Once the eigenfeatures have been found, the quality of the match between an eigenfeature and a region $\vec{\Phi}$ of the image is inversely proportional to the mean square error, ϵ:

$$\epsilon = \|\vec{\Phi} - \vec{\Phi}_f\|$$

where $\vec{\Phi}_f$ the projection of $\vec{\Phi}$ onto the feature space. The position of the region producing the global minimum can be considered the position of the feature. Geometric constraints can improve the accuracy by "weeding out" obviously bogus feature positions. Figure 3-3 shows a Cyberware texture map with the positions of the features marked by white dots.

For ViewPoint data, the model is "flattened" into a two-dimensional bitmap image by transforming each vertex $\vec{v}_i$ in the model from cartesian $\vec{v}_i = (x, y, z)$ coordinates into cylindrical $\vec{v}_i' = (\theta, y)$ (discarding the range component) by the relation $\theta = \arctan(y/x)$. The resulting coordinates are rescaled into the range [0..511] and displayed as (x, y) points so that critical features can be located by hand and saved into a parameter file. This process need only be done once, but it must be done well, as all subsequent Cyberware data regularizations use the same template. Figure 3-4 shows an example template model with the main features highlighted with black circles.

3.2.4 Regularization of Face Data

In addition to the incompleteness problem mentioned in the Cyberware Data Preprocessing section, another difficulty with Cyberware scan data is that the subject is not necessarily in exactly the same location or facing in exactly the same direction. The pixel locations of important features are not consistent between subjects or even between scans of the same subject. Worse still, not all scans have the same resolution or dimensions.

To achieve meaningful analysis of the human head, the number of data points for each subject must be *exactly* the same, and corresponding features must be placed at the same

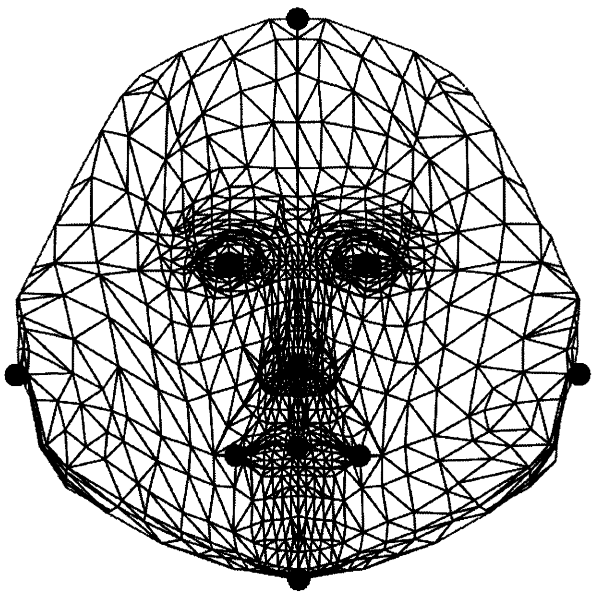

Figure 3-4: Flattened ViewPoint face model with key feature points marked.

locations in the data. To satisfy this necessity, the three-dimensional face model will be "morphed" as closely as possible into the shape of the Cyberware scan subject, both to reduce the amount of data involved from over a hundred thousand points to a few thousand, and and to enforce structural consistency and feature matching between subjects.

This is achieved by performing a two-dimensional linear transformation of the flattened ViewPoint vertices $\vec{v}_i^f$ and extracting the range at those points. Feature positions are taken in groups of three, defining non-overlapping two-dimensional triangular regions of the face to be warped from the flattened ViewPoint model to the Cyberware data. Lee, Terzopoulos, and Waters [12] use a somewhat more complex method, fitting contours of the template model to corresponding contours in the range data and then using a spring energy-minimization approach to position the remaining points.

To warp a region j from the ViewPoint template to the Cyberware data set, we define an affine transformation, T. The three corners of a source region, $\vec{s}_{j,1}$, $\vec{s}_{j,2}$, and $\vec{s}_{j,3}$, and the three corners of the corresponding destination region, $\vec{d}_{j,1}$, $\vec{d}_{j,2}$, and $\vec{d}_{j,3}$, are taken as column 3-vectors in homogeneous coordinates (with 1 as the third coordinate) and combined into two matrices, S_j and D_j:

$$S_j = \begin{pmatrix} s_{j,1,x} & s_{j,2,x} & s_{j,3,x} \\ s_{j,1,y} & s_{j,2,y} & s_{j,3,y} \\ 1 & 1 & 1 \end{pmatrix}$$

$$D_j = \begin{pmatrix} d_{j,1,x} & d_{j,2,x} & d_{j,3,x} \\ d_{j,1,y} & d_{j,2,y} & d_{j,3,y} \\ 1 & 1 & 1 \end{pmatrix}$$

The transformation matrix T_j is computed from these by right-multiplying D_j by the inverse of S_j:

$$T_j = D_j S_j^{-1}$$

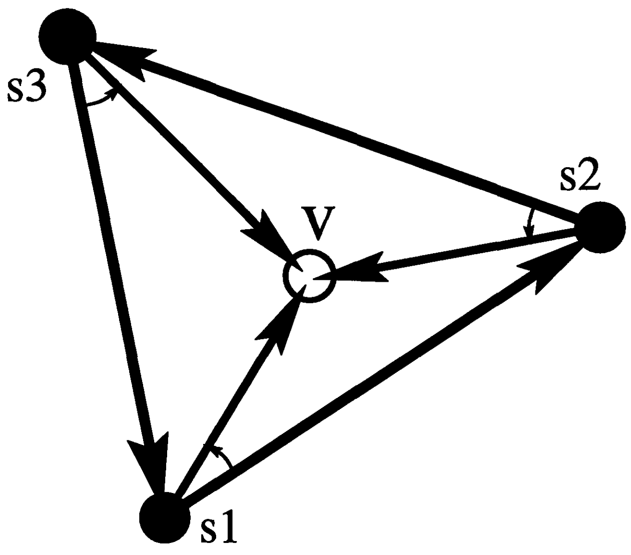

Figure 3-5: A point v inside a triangular region bounded by points s_1, s_2 and s_3.

T_j will map the corners of source region j directly onto those of destination region j, "stretching" the region like a rubber sheet, carrying all points inside the region along. How do we know if a given point $\vec{v}_i'$ is inside source region j? A very simple and efficient inside-outside test: $\vec{v}_i'$ will be within the triangular region defined by $\vec{s}_{j,1}$, $\vec{s}_{j,2}$, and $\vec{s}_{j,3}$ if and only if the cross-products $(\vec{v}_i' - \vec{s}_{j,1}) \times (\vec{s}_{j,2} - \vec{s}_{j,1})$, $(\vec{v}_i' - \vec{s}_{j,2}) \times (\vec{s}_{j,3} - \vec{s}_{j,2})$, and $(\vec{v}_i' - \vec{s}_{j,3}) \times (\vec{s}_{j,1} - \vec{s}_{j,3})$ have the same sign (see Figure 3-5). For two-dimensional data, taking the third (z) coordinate to be zero simplifies the cross-product operation to $\vec{a} \times \vec{b}$ to $a_x b_y - a_y b_x$.

Once the region containing $\vec{v}_i'$ is found, $\vec{v}_i'$ is converted to a column 3-vector in two-dimensional homogeneous coordinates and left-multiplied by T_j to produce a point $\vec{c}_i'$ in two-dimensional Cyberware head coordinates. The range at each point $\vec{c}_i'$ is sampled by taking the weighted average of the range data in a 5×5 square region centered on the point:

$$R_i = \sum_{b=-2}^{2} \sum_{a=-2}^{2} w(a,b) r(x_i - a, y_i - b)$$

where $w(a,b)$ are the weighting coefficients chosen to be a pseudo-Gaussian blurring kernel, so as to reduce the influence of any one range pixel:

$$w(-2..2,-2..2) = \frac{1}{256} \begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}$$

The resulting "smoothed" range R_i and the (x,y) location of $\vec{c}_i'$, (x_i, y_i), are combined

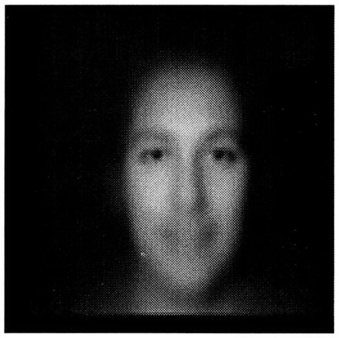

Figure 3-6: The average of 7500 Photobook FERET database images

to form a three-dimensional cartesian-coordinate Cyberware head point $\vec{c}_i$:

$$\vec{c}_i = \begin{pmatrix} -r_i \sin(\frac{\pi}{256}x_i) \\ -y \\ -r_i \cos(\frac{\pi}{256}x_i) \end{pmatrix}$$

The result is a simplified Cyberware head model with the same mesh structure as the ViewPoint model template. The texture coordinates corresponding to (x_i, y_i) can be bound to each vertex of the model, allowing the Cyberware texture data to be applied to the model. This produces a very realistic yet very compact model almost indistinguishable from a rendition using the full data set.

3.2.5 Alignment to Photobook FERET Format

The preceding procedure regularizes and greatly simplifies the data set, but does not solve the more fundamental problem of variations in position and rotation. We will use the 128×128 images from the Photobook FERET database, each with the features in approximately the same location. Examination of the average head (see figure 3-6) yields canonical positions: the left eye is at $(58.0, 59.0)$, the right eye at $(83.0, 59.0)$, the tip of the nose at $(70.5, 76.0)$, and the center of the mouth at $(70.5, 91.0)$. The nose and mouth constraints are not solid, but hold for many of the images. While there is significant variation within the data set, these feature positions best represent those of over 7500 database images.

We define three-dimensional feature position vectors $\vec{e}_l$, $\vec{e}_r$, $\vec{n}$, and $\vec{m}$ as the position of the left eye, right eye, the tip of the nose, and the center of the mouth, repectively, derived from corresponding two-dimensional feature positions by extracting the range at those points and converting to rectangular coordinates. The position vector $\vec{e}$ is taken as the midpoint between the two eye positions,

$$\vec{e} = \frac{1}{2}(\vec{e}_l + \vec{e}_r)$$

To align the model images with the average FERET face image, we set the the y component of $\vec{e} - \vec{n}$ (the vertical distance between the eyes and nose) to 17 ($= 76 - 59$), the y component of $\vec{n} - \vec{m}$ (the vertical distance between the nose and mouth) to 15 ($= 91 - 76$), and the x component of $\vec{r} - \vec{l}$ (the horizontal distance between the eyes) to 25 ($= 83 - 58$). The model derived from the Cyberware data set is in Cyberware pixel coordinates, and must be transformed to meet these requirements. In addition, to standardize model position in three-space, we translate the model so that the midpoint of the eyes, $\vec{f_e}$ is placed at the origin.

The first step in alignment is to eliminate rotational variations about the y and z axes, yielding a head that is oriented vertically, facing a viewer along the z axis. We define a plane bisecting the head defined by the three points $\vec{e}$, $\vec{n}$, and $\vec{m}$, the plane normal $\hat{n}$ being

$$\hat{n} = \frac{(\vec{e} - \vec{n}) \times (\vec{e} - \vec{m})}{\|(\vec{e} - \vec{n}) \times (\vec{e} - \vec{m})\|}$$

We then align this bisecting plane so that it intersects the viewer by finding an axis of rotation $\vec{r}$ and angle ρ that rotate the plane normal to point along the x axis. To find these, we take the cross-product of the plane normal, $\hat{n}$ with the x axis ($\hat{x} \equiv [100]^{\mathrm{T}}$). Taking advantage of the fact that the magnitude of $\vec{a} \times \vec{b}$ is $\|\vec{a}\|\|\vec{b}\| \sin \rho$:

$$\begin{aligned} \hat{r} &= \frac{\hat{n} \times \hat{x}}{\|\hat{n} \times \hat{x}\|} \\ \rho &= \arccos(\|\hat{n} \times \hat{x}\|) \end{aligned}$$

The entire model is then rotated about $\hat{r}$ by ρ.

The second step is to eliminate rotations about the x axis, by finding a rotation θ that will set the ratio of the eye-nose distance and the eye-mouth distance to 17:32. Assuming orthographic projection, the image distances between the eyes and nose and the eyes and mouth are

$$\begin{aligned} e'_y - n'_y &= (e_z - n_z) \sin \theta + (e_y - n_y) \cos \theta \\ e'_y - m'_y &= (e_z - m_z) \sin \theta + (e_y - m_y) \cos \theta \end{aligned}$$

Rearranging and solving for θ, we get an inverse-tangent formula for the proper angle of rotation about the x axis:

$$\theta = \arctan \frac{(e_y - n_y) - \frac{17}{32}(e_y - m_y)}{(e_z - n_z) - \frac{17}{32}(e_z - m_z)}$$

The final step is to normalize the size and position of the model. Scale by $24/(r'_x - l'_x)$ in the x and z directions, and by $32/(e'_y - m'_y)$ in the y direction, after rotation.

Note that this transformation places the features at positions centered around $(0, 0)$ rather than $(70, 59)$. While this is not exactly the location we specified earlier, it is a somewhat more natural position for a three-dimensional model, and judicious choice of camera position will move the face into the correct place in the image.

3.3 Generating Models from Images, Part 1

Images of faces are a highly constrained domain, being in general very similar in structure. Every human face has two eyes, a nose, and a mouth at essentially fixed locations. If these feature positions are aligned to specific points in the image, then there will be strong similarities between faces. That is, the images will not be dispersed uniformly throughout the space of all possible images, but will instead be clustered somewhere in image space and can be described by a relatively small number of dimensions. Using the Karhunen-Loève expansion (eigenvectors of the autocorrelation matrix) we can perform principal component

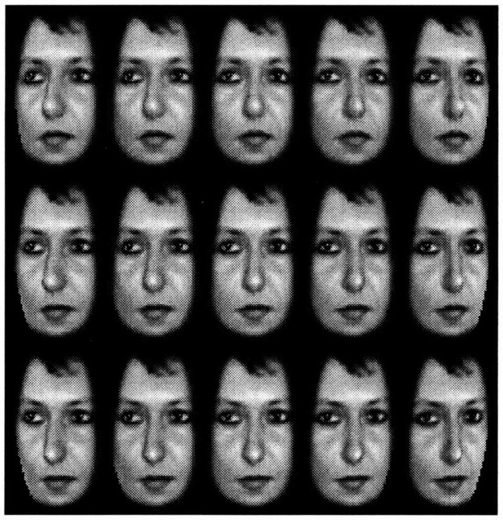

Figure 3-7: Sample training images, showing 15 standard orientations.

analysis and find the vectors that best account for the distribution. It is quite possible to define a relatively small subspace of linearly independent face components, a "face space".

We are, however, interested in a somewhat larger potential space, the image/model space correlating images and the models that generate them. By performing principal component analysis in this space, we hope to correlate image features with model features, permitting approximate reconstruction of face models from images.

3.3.1 Vector Generation

After a regularized, aligned, texture-mapped face model has been generated from a Cyberware scan, the model is rendered into a 51×88 subregion of the full 128×128 image, with $(44, 22)$ as the upper left corner. When the rendered face is masked with the alpha-channel used by Photobook for face recognition, substantial regions of the image are always black, and need not be included in the output.

At the rendering stage, a fully textured model is available, having been constructed explicitly from the Cyberware scan data. This model can rendered with any alteration desired, allowing a wide variety of details to be recovered. For example, one can apply small rotations to make reconstruction resistent to slight variations in orientation, pulling out correspondingly rotated models from the images. Similarly, different lighting directions and different facial movements (open and closed mouth, open and closed eyes, smiling, frowning, etc.) can be applied to the model, allowing similar images to be reconstructed with enhanced accuracy.

Fifteen different orientations were chosen as a reasonable set of representatives, sweeping across a five by three array of orientations spaced approximately 0.1 radians ($\approx 5.7°$) apart. This distance is a compromise between sampling density and coverage. While the images appear quite similar, the differences in the models are more significant, leading to severe feature mismatch errors if the increments are made too large. Figure 3-7 shows a typical face in each of the fifteen orientations, with horizontal rotation varying between -0.2 and $+0.2$ radians and vertical rotations varying between -0.1 and $+0.1$ radians.

Each of the synthetic face images, along with the (x, y, z) coordinates of the vertices of the model that generated it, are strung out to create M training vectors $\vec{\Gamma}_1$, $\vec{\Gamma}_2$, ...,

$\vec{\Gamma}_M$, each of length N. Images with different rotations are kept in separate training sets, leaving one image per subject per training set. This will produce a separate eigenspace for each orientation, reducing computational load and preventing unlike orientations from intermixing.

3.3.2 Face Normalization

The overall intensity of an image and the shape of the model are essentially uncorrelated. To decouple the face brightness and deformation magnitude, each synthetic face image $\vec{\Gamma}$ is normalized to zero mean, unit variance by, for each image pixel, γ_j, subtracting off the average pixel value, $\bar{\gamma}$,

$$\bar{\gamma} = \sum_{k=1}^{N} \gamma_k$$

and dividing by the square root of the sum of squares:

$$\gamma'_j = \frac{\gamma_j - \bar{\gamma}}{\sqrt{\sum_{k=1}^{N}(\gamma_k - \bar{\gamma})^2}}$$

This normalization step produces an image vector $\vec{\Gamma}'$ with all variations in overall brightness and contrast removed, leaving only salient features.

From the training set, we define the average face $\vec{\Psi}$ by

$$\vec{\Psi} = \frac{1}{M}\sum_{i=1}^{M}\vec{\Gamma}'_i$$

Subtracting the average image and average model from each of the training images and training models, we get differences $\vec{\Phi}_1$, $\vec{\Phi}_2$, ..., $\vec{\Phi}_M$,

$$\vec{\Phi}_i = \vec{\Gamma}'_i - \vec{\Psi}$$

3.3.3 Generation of the Covariance Matrix

The set of training vectors $\vec{\Phi}_i$ are then subjected to principal component analysis, which finds a set of M orthonormal vectors $\vec{u}_1$, $\vec{u}_2$, ..., $\vec{u}_M$ which best represent the distribution of the data in image/model space. The kth such vector $\vec{u}_k$ is chosen so that

$$\lambda_k = \frac{1}{M}\sum_{i=1}^{M}(\vec{u}_k^{\mathsf{T}}\vec{\Phi}_i)^2$$

is maximized, while subject to the constraint of orthonormality,

$$\vec{u}_l^{\mathsf{T}}\vec{u}_k = \delta_{lk} = \begin{cases} 1, & \text{if } l = k \\ 0, & \text{otherwise} \end{cases}$$

These quantities, $\mathbf{u}_k$ and λ_k are simply the eigenvectors and eigenvalues of the covariance matrix

$$\mathbf{C} = \frac{1}{M}\mathbf{F}\mathbf{F}^{\mathsf{T}}$$

where $\mathbf{F}$ is a matrix whose columns are the M normalized training faces,

$$\mathbf{F} = [\vec{\Phi}_1 \vec{\Phi}_2 \cdots \vec{\Phi}_M]$$

Unfortunately, $\mathbf{C}$ has the highly undesirable property of being huge: it is a square matrix with the length of each side being the length of each training vectors, N, which is the number of pixels in an image plus the number of vertices times the number of components per vertex. For even modest image sizes and simple models, diagonalization of the resulting $N \times N$ matrix is exceedingly painful, even on modern computer hardware.

The M training samples define a hyperplane of dimension $M-1$ (two points define a line, three define a plane, and so on) within the full N-dimensional space of images and models. Once the mean is subtracted out, this hyperplane will pass through the origin, requiring only $M-1$ orthogonal vectors to span it. There will thus be at most $M-1$ nonzero eigenvalues and linearly independent eigenvectors of the covariance matrix. Clearly, most of the work done in diagonalizing $\mathbf{C}$ is wasted if $M \ll N$ (which is usually the case).

The "snap-shot" method of Sirovich [5] asserts that the eigenmodes are a linear combination of the faces. Instead of grinding away at $\mathbf{C}$, we can instead define a much smaller, $M \times M$ covariance matrix $\mathbf{L}$,

$$\mathbf{L} = \mathbf{F}^{\mathsf{T}}\mathbf{F}$$

We can find the eigenvalues μ_k and eigenvectors $\vec{v}_k$ of $\mathbf{L}$ and then take appropriate linear combinations of the face vectors $\vec{\Phi}_i$ to recover the eigenvectors of $\mathbf{C}$. See the Appendix for a more detailed description of the eigenanalysis method used to find μ_k and $\vec{v}_k$.

3.3.4 Generating Eigenfaces from Eigenvalues

We now return to the subject of recovering the length-N eigenvectors of $\mathbf{C}$, the eigenfaces, from the length-M eigenvectors of $\mathbf{L}$.

Earlier we stated (without proof) that the eigenvalues $\vec{u}_1$, $\vec{u}_2$, ..., $\vec{u}_{M-1}$ of $\mathbf{C}$ could be easily recovered from the eigenvalues $\vec{v}_1$, $\vec{v}_2$, ..., $vecv_{M-1}$ of $\mathbf{L}$. Now we present the proof. Left-multiplying the equation for the kth eigenvector of $\mathbf{L}$ by $\mathbf{F}$ and regrouping,

$$\begin{aligned} (\mathbf{F}^{\mathsf{T}}\mathbf{F})\vec{v}_k &= \mu_k \vec{v}_k \\ \mathbf{F}(\mathbf{F}^{\mathsf{T}}\mathbf{F})\vec{v}_k &= \mu_k \mathbf{F}\vec{v}_k \\ (\mathbf{F}\mathbf{F}^{\mathsf{T}})(\mathbf{F}\vec{v}_k) &= \mu_k(\mathbf{F}\vec{v}_k) \end{aligned}$$

As can be seen from the last equation, the vectors $\mathbf{F}\vec{v}_k$ are the eigenvectors of $\mathbf{F}\mathbf{F}^{\mathsf{T}} = \mathbf{C}$. Thus, we can find the kth eigenvalue $\vec{u}_k$ of $\mathbf{C}$ by left-multiplying the kth eigenvalue $\vec{v}_k$ of $\mathbf{L}$ by $\mathbf{F}$. The eigenvectors $\vec{v}_1$, $\vec{v}_2$, ..., $\vec{v}_{M'}$ specify linear combinations of the training faces to construct each eigenface $\vec{u}_i$ $(i = 1, \ldots, M)$:

$$\vec{u}_i = \sum_{j=1}^{M} \vec{v}_{ij} \vec{\Phi}_j$$

The nonzero eigenvalues, λ and μ are in fact the same for both, since

$$(\mathbf{F}\mathbf{F}^{\mathsf{T}})(\mathbf{F}\vec{v}_k) = \mu_k(\mathbf{F}\vec{v}_k)$$

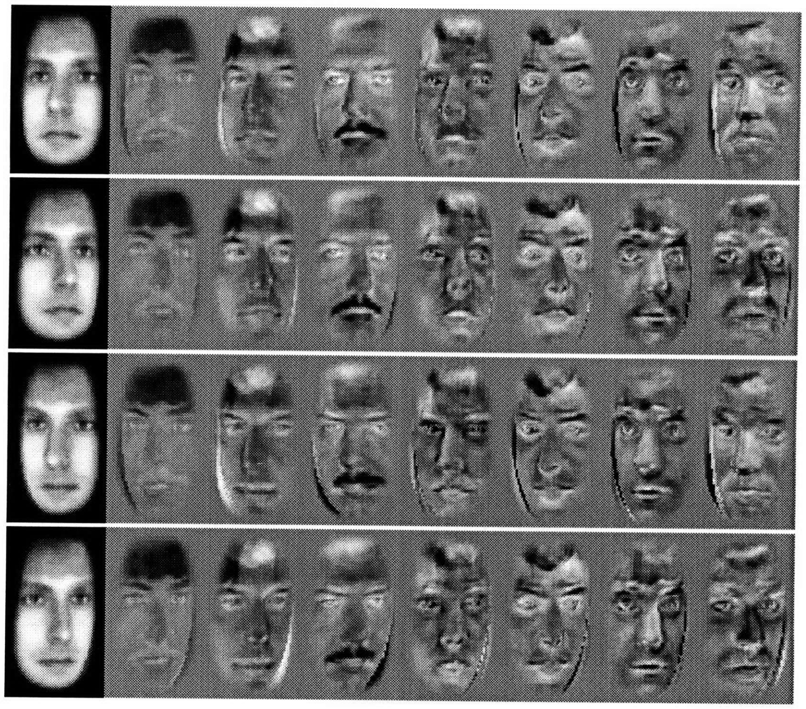

Figure 3-8: Mean and first 7 eigenimages for the four extreme rotations.

is just another way of writing

$$(\mathbf{F}\mathbf{F}^{\top})\vec{u}_k = \lambda_k \vec{u}_k$$

The magnitude of each eigenvalue gives the variance captured by the corresponding eigenvector, and provides an indication of the relative importance of that eigenvector. By selecting the eigenvectors corresponding to the largest M' eigenvalues ($M' < M$), we can reduce the dimension of the eigenspace while introducing a minimal amount of error.

To make the eigenspace $\mathbf{U} = \mathbf{F}\mathbf{V}$ orthonormal, we divide each eigenvector $\vec{u}_i$ by the square root of the corresponding eigenvalue, setting the variance to one:

$$\vec{u}_i' = \frac{\vec{u}_i}{\sqrt{\mu_i}}$$

Figure 3-8 shows the mean face, $\vec{\Psi}$, and the first $M' = 7$ eigenfaces, $\vec{u}_1$, $\vec{u}_2$, ..., $\vec{u}_7$, for the four extreme orientations, $(-0.30, +0.15)$, $(+0.30, +0.15)$, $(-0.30, -0.15)$, and $(+0.30, -0.15)$ radians in the x and y directions. Note that the general character of each eigenface remains essentially unchanged, save for rotation.

3.4 Generating Models from Images, part 2

Using the principal components derived from the training set, one can project an arbitrary Photobook-format image onto the face space to acquire a reconstruction of that image. The eigenspace is capable of estimating missing portions of an input vector. In this case, the missing portion is the three-dimensional model we seek to reconstruct from the image.

3.4.1 Projection

With an orthonormal eigenspace, we can project a Photobook-style face image vector onto the space. As preparation, the image to be projected, $\vec{\Gamma}$, is reduced to zero-mean, unit-variance by, for each element, γ_j, subtracting off the mean pixel value, $\bar{\gamma}$ and dividing by the square root of the sum of squares (as with the training data):

$$\gamma'_j = \frac{\gamma_j - \bar{\gamma}}{\sqrt{\sum_{k=1}^{N}(\gamma_k - \bar{\gamma})^2}}$$

The scaling factor and average pixel value are saved for later reconstruction, allowing the projected image to be restored to the appropriate brightness and contrast.

For each of the separate eigenspaces generated for each orientation, we subtract off the average face for that orientation, $\vec{\Psi}^d$:

$$\vec{\Phi}^d_{\text{image}} = \vec{\Gamma}'_{\text{image}} - \vec{\Psi}^d_{\text{image}}$$

Since we have no better estimate, we implicitly assume that the model portion of $\vec{\Gamma}$ is the average head model. The model portion of $\vec{\Phi}^d$ is therefore filled with zeros,

$$\vec{\Phi}^d_{\text{model}} = [000\cdots0]^{\top}$$

$\vec{\Phi}^d$ can be transformed into its eigenface components (i.e., projected onto the eigenspace of principal face components) by simply taking the dot-product with each eigenface, $\vec{u}^d_1$, $\vec{u}^d_2$, ..., $\vec{u}^d_{M'}$:

$$\omega^d_k = (\vec{\Phi}^d)^{\top}\vec{u}^d_k$$

or, in matrix form,

$$\vec{\Omega}^d = (\vec{\Phi}^d)^{\top}\mathbf{U}^d$$

where $\vec{\Omega}^d = [\omega^d_1\omega^d_2\cdots\omega^d_{M'}]^{\top}$. Each weight describes "how much" of each orthogonal eigenface is present in $\vec{\Phi}^d$.

3.4.2 Reconstruction

To reconstruct the image from eigenface basis d, we take a weighted sum of the eigenfaces to get an approximate image $\vec{\Theta}^d$, which is $\vec{\Phi}$ projected onto the eigenspace:

$$\vec{\Theta}^d = \sum_{i=1}^{M'}\omega^d_i\vec{u}^d_i$$

or, in matrix form,

$$\vec{\Theta}^d = (\vec{\Omega}^d)^{\top}\mathbf{U}^d$$

The reconstruction will not be exact, but can be quite close if the face is within the hyperplane spanned by the training set. The reconstruction process will fill in any missing data by mapping the face onto the closest (in the RMS sense) point in the eigenspace. Unlike the original face vector $\vec{\Phi}$, a reconstruction $\vec{\Theta}$ will come complete with a model.

The optimal eigenspace, $\mathbf{U}^D$, yields the minimum reconstruction error,

$$\epsilon = ||\vec{\Phi}^D - \vec{\Theta}^D|| \leq ||\vec{\Phi}^d - \vec{\Theta}^d|| \quad \text{for all } d$$

and can be safely assumed to most closely correspond to the orientation of the face in the image. The recovered model, $\vec{\Theta}^D_{\text{model}}$ will possess the appropriate rotation, bringing its features into alignment with the image.

3.4.3 Reconstitution

The optimal reconstructed face $\vec{\Theta}^D$ is must be converted into a useful format. Like the products of the vector generation phase, $\vec{\Theta}^D$ is a long column of numbers with no internal structure. The vector must be reconstituted into a human-usable form: a greyscale image and a 3D model.

The image portion, $\vec{\Theta}^D_{\text{image}}$, is trivial to transform into a human-viewable image. The pixel values are scaled by the square-root of the sum of squares that was removed during normalization, and the average pixel value is added back in. The resulting values are rounded to the nearest integer and clamped to the range [0..255], then output as 8-bit greyscale pixels with an appropriate image header.

The model portion, $\vec{\Theta}^D_{\text{model}}$ must be combined with the ViewPoint face model template to transform it from a simple list of points to a full three-dimensional model. Since the points are in exactly the same order as in the template, representing the positions of the deformed vertices of the ViewPoint model, they can be substituted for the corresponding vertices. The resulting model has the same structure as the template, but the shape of the reconstructed face.

For added realism, the original Photobook FERET image is texture-mapped onto the model. Since the training models were expressly constructed so that image pixel distances and model spatial distances are identical, finding texture coordinates consists of nothing more than extracting the x and y coordinates of each vertex, shifting it to the appropriate place in the image by moving the origin to $(70, 59)$, and scaling the coordinates into the range [0..1]. Smoothly interpolating texture values produces a model with slightly blurry features, but no pixelization. Unless feature positions are outside of the range of orientations provided in the fifteen eigenspaces, the model feature positions will correspond to within 0.05 radians in each direction.

If a large image has been reduced in size to FERET format, the original image can be used as the texture map instead of the smaller image. This will yield a much higher-resolution texture map, and a model of superior quality. If the image is large enough, the resolution can exceed that of the Cyberware scanner.

Chapter 4

Results

For testing purposes, a set of eight Cyberware head scans were converted using the aforementioned techniques. The centers of the eyes, tip of the nose, center and left and right sides of the mouth, and top, bottom, left, and right sides of the face were found by visual inspection.

The ViewPoint template model was then warped to fit the data, and selected points extracted to form the eight texture-mapped, three-dimensional face models shown in Figure 4-1. The models were then placed in fifteen slightly different orientations and rendered as 51×88 masked images and output as large composite vectors of image pixels and model vertices. Figure 4-2 shows frontal-view images of the training-set models.

Principal component analysis was performed to elucidate a linearly independent basis of the face space, yielding eigenfaces and eigenmodels. Images for model recovery were then projected onto the reduced-dimension face space, returning models. Keeping the seven eigenvectors produced yields only very small errors (mostly due to roundoff). With a larger training set, truncating to half or less will still yield satisfactory results. Figure 4-3 shows a series of in-sample images of various rotations projected onto the eigenspace and rendered. The differences are quite difficult to see, becoming apparent primarily when differencing the images. Table 4-4 shows the root-mean-square reconstruction errors for the same face over the same rotations.

For images outside the training set, the errors are significantly larger, as the eigenspaces derived from the small training set cannot reproduce the images well. Table 4-6 gives RMS errors for the test images. The projected image still vaguely resembles the original, and the model produces decent results when texture mapped with the original image. Figure 4-5 shows sixteen Photobook FERET database images projected onto the eigenspace. Due to the small training set, the reconstructed images are not of especially high quality. Some of the faces have rotations beyond the range used in the training set, leading to significantly reduced accuracy, but the range of orientations provided by the fifteen eigenspaces will capture most variations.

From the Photobook images shown in Figure 4-5, the models shown in Figure 4-8 were recovered, and the original images applied as texture maps. For comparison, the same images were texture mapped onto the undeformed template head, yielding the images shown in Figure 4-9. Whereas the eigenmodel technique managed to capture the overall shape of many of the faces, the texture-mapped template head has tremendous difficulties around the edge of the jawline. In several cases, the face image is clearly "pasted" onto the front of the face, with the edges of the jawline in the image being mapped onto the cheek of the model.

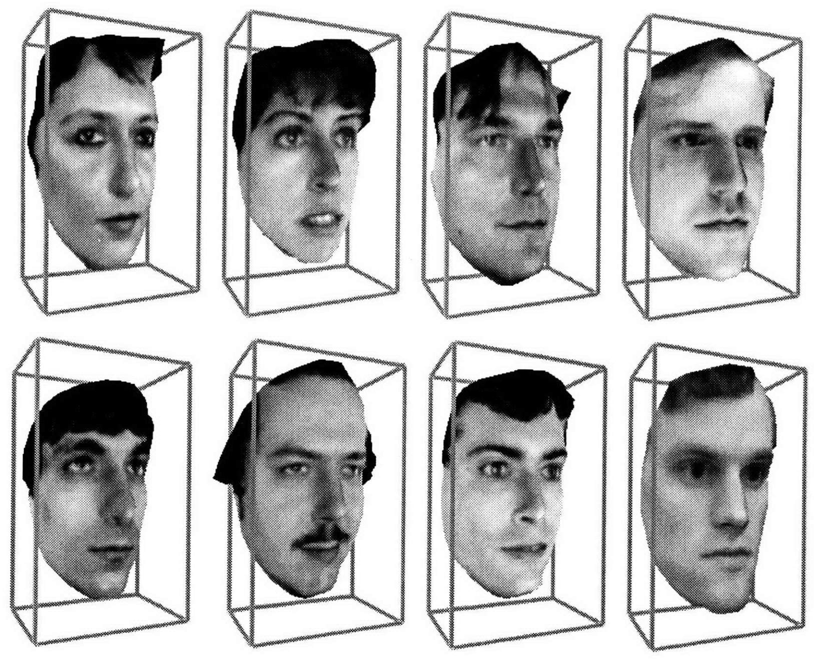
Figure 4-1: Models derived from Cyberware training set.

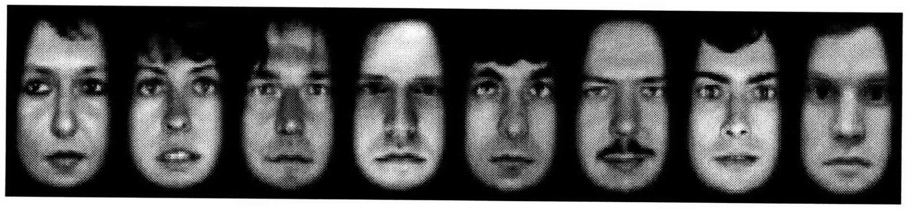
Figure 4-2: Frontal-view masked images derived from models.

Similarly, the eigenmodels capture hair and forehead shape, mapping dark hair into a slight protrusion, or mapping bald forehead into a smoothly-curved surface. In comparison, the template forehead is always the same shape regardless of the presence or absence of hair. Figure 4-7 shows two models, one derived from eigenmodels, the other derived from the template head. While the recovered models are far from perfect, they are clearly superior to the template model.

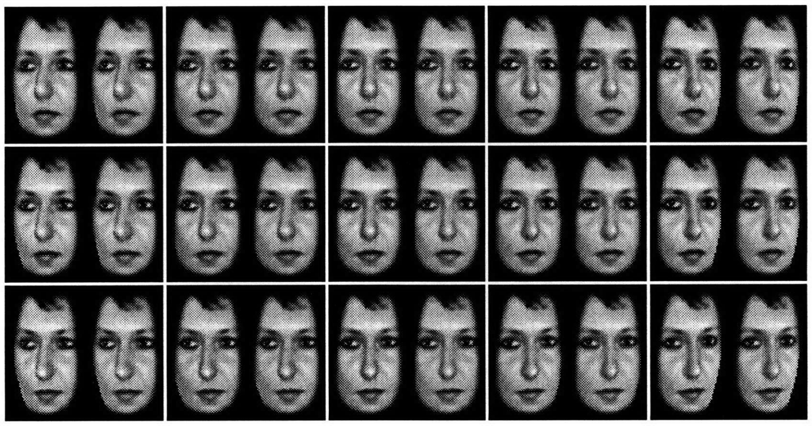

Figure 4-3: Comparison of original and projected training images for 15 standard rotations.

	-0.30	-0.15	0.00	+0.15	+0.30
+0.15	0.037073	0.037145	0.037296	0.037311	0.037332
0.00	0.037266	0.037781	0.037905	0.037975	0.037821
-0.15	0.037679	0.038078	0.038326	0.038393	0.037887

Figure 4-4: Typical in-sample RMS errors for 15 standard rotations.

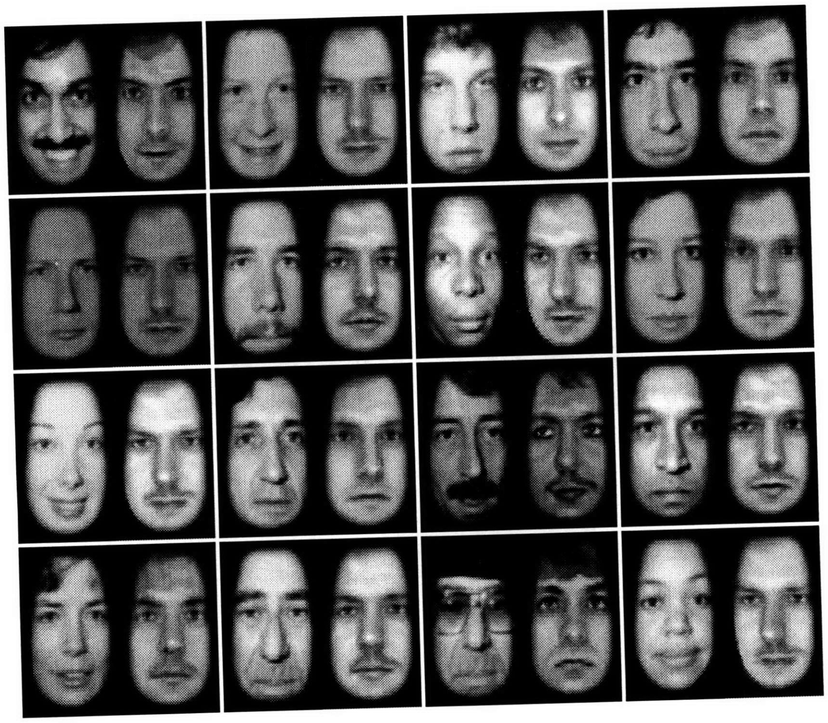

Figure 4-5: Comparison of 16 original and projected Photobook FERET database images.

0.469104	0.283834	0.377362	0.318605
0.282798	0.376462	0.478344	0.278389
0.315224	0.308800	0.528300	0.369649
0.393337	0.348274	0.411763	0.343560

Figure 4-6: RMS errors for 16 reconstructed Photobook FERET database images.

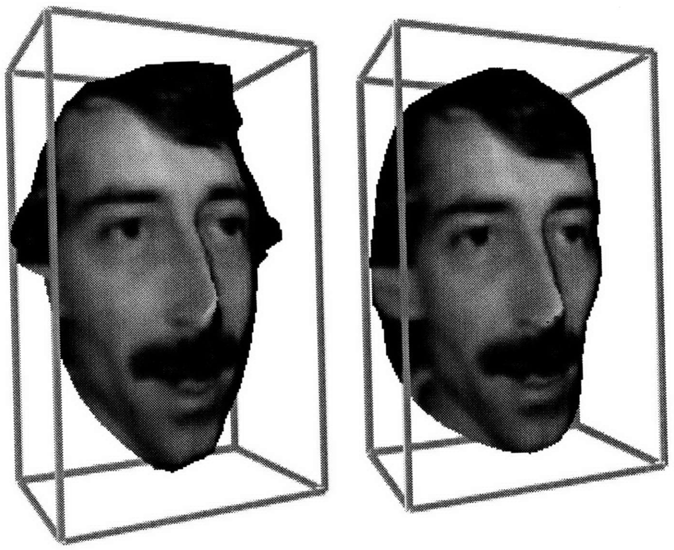

Figure 4-7: Comparison of texture-mapped recovered model and texture-mapped, undeformed template model.

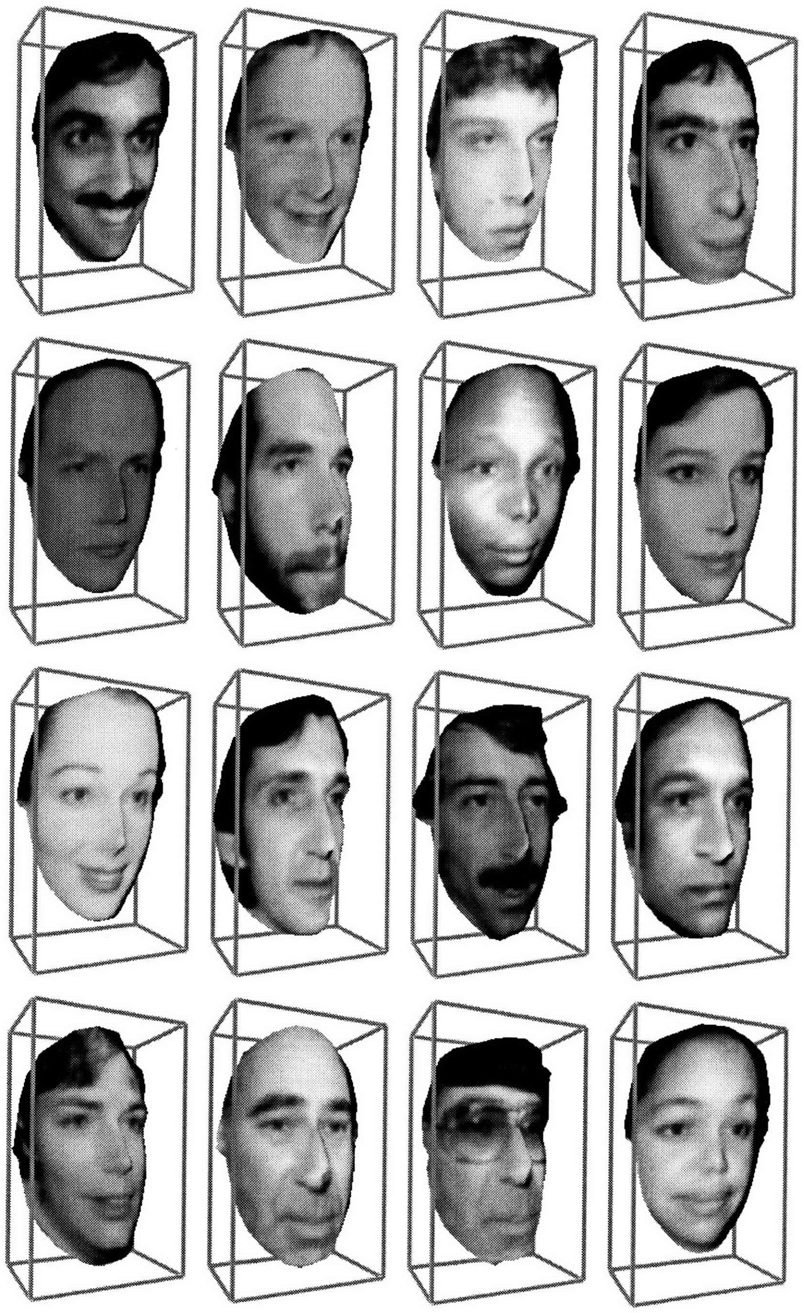

Figure 4-8: Recovered models for 16 Photobook FERET database images.

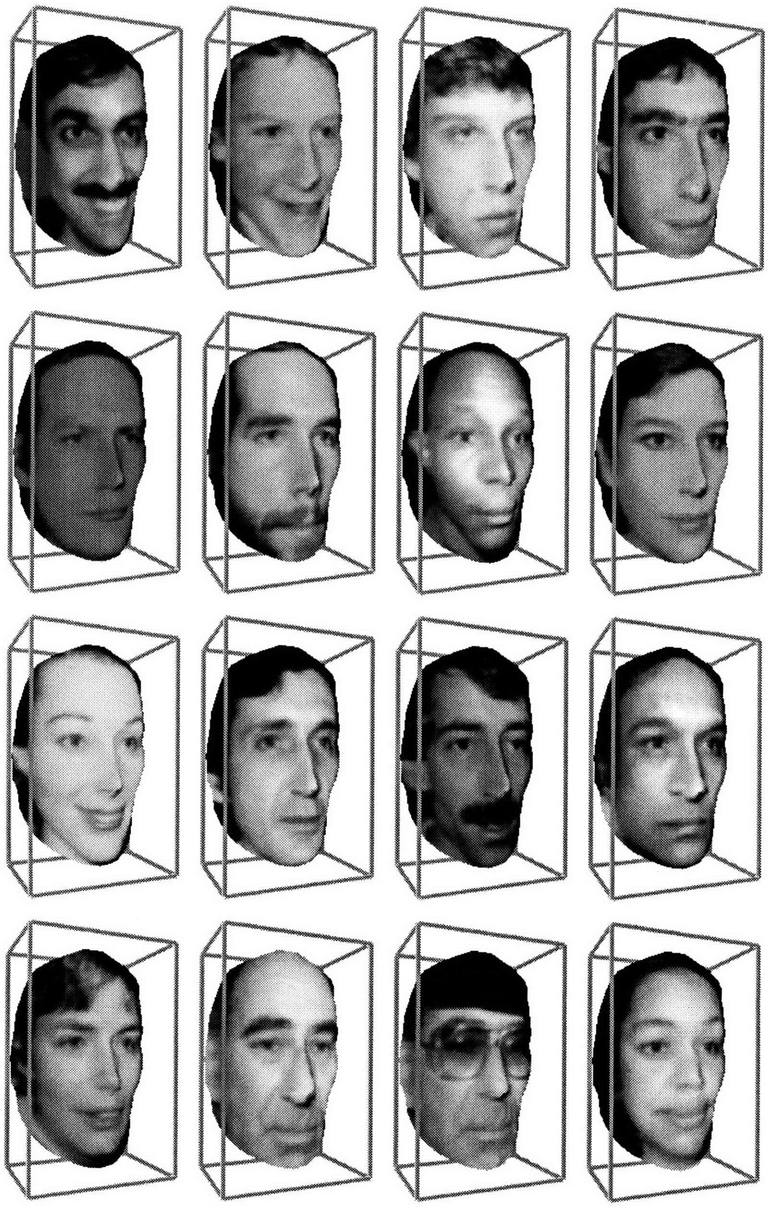

Figure 4-9: 16 Photobook FERET database images mapped onto undeformed template.

Chapter 5

Conclusion

The model recovery system has one major intrinsic benefit: since the training models can be rendered in any style or with any alteration desired to create training images, a large number of categories of face image can be produced and subsequently reconstructed from similar images with enhanced accuracy. This capability is already used to tolerate small rotations and recover rotated models. It could also be used to tolerate lighting direction changes, scaling, facial expressions, and so forth. In essence, anything done to the training models and reflected in the rendered images is something that can be recovered from an image.

As can be seen from the results in the preceding chapter, model reconstruction from images using eigenspaces yields great promise. The reconstructed models have somewhat reduced detail, but can be of a level of quality comparable to the originals, save for a few minor feature mismatches. Even with manual feature tracking and a small training set, an extremely diverse collection of faces were dimensionalized into models of acceptable quality. There are a few caveats and limitations present, but none that cannot be surmounted.

5.1 Known Problems

Several image features will cause significant mismatches between the original and reconstructed image. Fortunately, most are preventable or correctable with a little care.

The first category of mismatch comes from face images with major rotations; if the image to be reconstructed is not mostly frontal, and thus outside the range of representative orientations, the reconstruction will be quite poor. The system will in essence be trying to project an image from far outside the spanned hyperplane onto the hyperplane, yielding nonsensical results. Fortunately, a variety of techniques exist to reorient an image to frontal facing. The missing detail on the far side of the face can be approximated, yielding adequate results.

The second category of mismatch comes from significant differences in lighting. Certain features such as dark hair low down on the face are easily confused with shadow or brightness (similar but for a sign change) due to lighting. Cyberware scans are, in essence, front lit from all directions, as the scanning beam is always radial to the centerline of the subject's head. Images for reconstruction should also be front-lit, but non-frontal lighting can be corrected with intensity correction. Nastar and Pentland [15] present just such a system to manipulate lighting and reorient images. With light-reorientation, creating eigenspaces for each lighting direction is unnecessary, though possibly helpful for recovery of lighting direction.

The third category of mismatch comes from images of people with non-attached features such as glasses. Since the Cyberware scan data is in essence a two-dimensional function, it cannot represent multivalued ranges. There already exists a problem with the detail inside the nostrils, but little can be done to correct the situation except touch up the models with postprocessing. To avoid trouble, no subject should wear eyeglasses during the Cyberware scan. Images of people with glasses will not cause too much difficulty, as the non-representable detail will be mostly ignored, but the image with eyeglasses will be texturemapped onto the glasses-free model, yielding odd results (one of the test images is of a face with glasses). The eyeglasses can be added to the model later, using traditional three-dimensional modelling techniques.

5.2 Future Work

For future work, there are several significant improvements to the system that would greatly enhance its performance.

First, a training set of eight models is woefully inadequate; at least a hundred models are needed to capture common features and significant variations. The current training set is quite sparse, with insufficient coverage of face space to make generalizations. Large databases of Cyberware head scans are not difficult to find.

Second, a distance-from-feature-space feature finder needs to be added. For the eight training images, manual feature extraction was sufficient, but for several hundred, the process must be automated. While a few models must be analyzed by hand to train the feature finder, the finder can "bootstrap" itself by making an initial guess, refined by the user. The resulting location is then added to the feature training set, and the eigenfeatures are recomputed.

Third, a similar automated feature-finder should be applied to the images to be reconstructed. While the multiple eigenspaces can handle small rotations, they do not handle variations in face size or position. Repositioning and rescaling the image so that the feature position constraints are met will reduce most of the feature mismatches present with the current system.

Lighting correction and image reorientation will allow a much greater variety of images to be successfully processed for reconstruction. While altered images have missing detail, sometimes incomplete images are the best available.

In conclusion, this structure-from-image recovery system, while still nascent, holds great promise.

Bibliography

[1] Karhunen, K. (1946). *Zur spektraltheorie stochasticher.* Prozesse Ann. Acad. Sci. Fennicae, 37.

[2] Loève, M. M. (1955). *Probability Theory.* Van Nostrand, Princeton, NJ.

[3] Jolliffe, I. T. (1986). *Principal Component Analysis.* Springer Verlag, New York, NY.

[4] Sirovich, L. and Kirby, M. (1987). Low-Dimensional Procedure for the Characterization of Human Faces. *J. Optical Society of America*, a 4, 519-524.

[5] Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loeve Procedure for the Characterization of Human Faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, 103-108.

[6] Turk, M., and Pentland, A. (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience.* 1(3), 71-86.

[7] Moghaddam, B. and Pentland, A. (1995). An Automatic System for Model-Based Coding of Faces. *IEEE Data Compression Conference*, Snowbird, Utah, March 1995.

[8] Cyberware, Inc. 2110 Del Monte Avenue, Monterey, CA 93940, USA +1 408 657-1450. "http://www.cyberware.com"

[9] Atick, J., Griffin, P., and Redlich, A. N. (1995). Statistical Approach to Shape from Shading: Reconstruction of 3D Face Surfaces from Single 2D Images.

[10] ViewPoint Datalabs. 625 S. State, Orem, UT 84058, USA +1 800 DATASET. "http://www.viewpoint.com"

[11] Basu, S., Essa, I., and Pentland, A. (1996). Motion Regularization for Model-Based Head Tracking. *13th International Conference on Pattern Recognition (ICPR '96)*, Vienna, Austria, August 25-30, 1996.

[12] Lee, Y., Terzopoulos, D., Waters, K. (19??). Realistic Modeling for Facial Animation.

[13] Essa, I., Basu, S., Darrell, T. and Pentland, A. (1996). Modelling, Tracking, and Interactive Animation of Faces using Input from Video. *Proceedings of Computer Animation '96 Conference*, Geneva, Switzerland, June 1996.

[14] Pentland, A., Picard, R. W., and Sclaroff, S. (1995). Photobook: Content-Based Manipulation of Image Databases. *SPIE Storage and Retrieval Image and Video Databases II*, No. 2185, Feb 6-10, 1994 San Jose.

[15] Nastar, C., and Pentland, A. (1995) Matching and Recognition Using Deformable Intensity Surfaces. *International Symposium on Computer Vision 1995.*

[16] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. *Numerical Recipes in C: The Art of Scientific Computing, Second Edition.* Cambridge University Press, 1992.

Appendix A

Eigenanalysis

Since the training data is real (pixel intensities are real, as are vertex positions), and $\mathbf{F}^\top\mathbf{F} = (\mathbf{F}^\top\mathbf{F})^\top = \mathbf{F}^\top(\mathbf{F}^\top)^\top = \mathbf{F}^\top\mathbf{F}$, $\mathbf{L}$ is both symmetric and real. We will therefore use Jacobi method for finding the eigenvalues and eigenvectors of a symmetric real matrix presented in *Numerical recipes in C* [16] to find $\mu_1, \mu_2, \ldots, \mu_{M-1}$ and corresponding eigenvectors $\vec{v}_1$, $\vec{v}_2, \ldots, \vec{v}_{M-1}$.

The Jacobi method consists of a sequence of orthogonal similarity transformations that reduce $\mathbf{L}$ to diagonal form, to the limits of machine precision. If n such transformations are required, the series of transformations looks something like:

$$\begin{aligned} \mathbf{L} &\longrightarrow \mathbf{P}_1^{-1}\mathbf{L}\mathbf{P}_1 \\ &\longrightarrow \mathbf{P}_2^{-1}\mathbf{P}_1^{-1}\mathbf{L}\mathbf{P}_1\mathbf{P}_2 \\ &\longrightarrow \ldots \\ &\longrightarrow \mathbf{P}_n^{-1}\cdots\mathbf{P}_2^{-1}\mathbf{P}_1^{-1}\mathbf{L}\mathbf{P}_1\mathbf{P}_2\cdots\mathbf{P}_n \end{aligned}$$

Each transformation $\mathbf{P}_i$ is a *Jacobi Rotation*, a plane rotation constructed to annihilate an off-diagonal matrix element. Alas, successive transformations partially undo previous transformations, but off-diagonal elements do tend to become smaller and smaller until they vanish. Accumulating the product of the transformations, $\mathbf{V} = \mathbf{P}_1\mathbf{P}_2\cdots\mathbf{P}_n$ gives the eigenvectors, while the elements of the final diagonal matrix $\mathbf{V}^\top\mathbf{L}\mathbf{V}$ are the eigenvalues.

A Jacobi Rotation $\mathbf{P}_{pq}$ is just the identity matrix $\mathbf{I}$ with changes in rows and columns p and q:

$$p_{pp} = p_{qq} = \cos\phi$$

and

$$p_{pq} = -p_{qp} = \sin\phi.$$

This orthonormal rotation transforms $\mathbf{L}$ via

$$\mathbf{L}' = \mathbf{P}_{pq}^\top\mathbf{L}\mathbf{P}_{pq}$$

Since the only elements that differ from identity are in rows and columns p and q, only rows p and q of and columns p and q of $\mathbf{L}$ change. $\mathbf{P}_{pq}^\top\mathbf{L}$ differs from $\mathbf{L}$ only on rows p and q, while $\mathbf{L}\mathbf{P}_{pq}$ differs from $\mathbf{L}$ only in columns p and q. Multiplying out the transformation and taking advantage of $\mathbf{L}$'s symmetry, we get a set of equations (for $r \neq p$ and $r \neq q$):

$$l'_{rp} = \cos\phi\, l_{rp} - \sin\phi\, l_{rq}$$

$$\begin{aligned}
l'_{rq} &= \cos\phi \; l_{rq} + \sin\phi \; l_{rp} \\
l'_{pp} &= \cos^2\phi \; l_{pp} + \sin^2\phi \; l_{qq} - 2\sin\phi\cos\phi \; l_{pq} \\
l'_{qq} &= \sin^2\phi \; l_{pp} + \cos^2\phi \; l_{qq} + 2\sin\phi\cos\phi \; l_{pq} \\
l'_{pq} &= (\cos^2\phi \; - \sin^2\phi) l_{pq} + \sin\phi\cos\phi (l_{pp} - l_{qq})
\end{aligned}$$

The objective is to zero all the off-diagonal elements by a series of rotations. To this end, set l'_{pq} to zero, which gives an expression for ϕ,

$$\begin{aligned}
\theta &\equiv \cot 2\phi \\
&\equiv \frac{\cos^2\phi - \sin^2\phi}{2\sin\phi\cos\phi} \\
&= \frac{l_{qq} - l_{pp}}{2l_{pq}}
\end{aligned}$$

Defining $t = \sin\phi / \cos\phi$, a new expression in θ is

$$t^2 + 2t\theta - 1 = 0$$

The smaller root of this equation can be found via the quadratic formula, with the discriminant in the denominator,

$$t = \frac{\theta}{\theta^2 + |\theta|\sqrt{\theta^2 + 1}}$$

This yields relations for $\sin\phi$ and $\cos\phi$,

$$\begin{aligned}
\cos\phi &= \frac{1}{\sqrt{t^2 + 1}} \\
\sin\phi &= t\cos\phi
\end{aligned}$$

Now, since l'_{pq} is zero (we explicitly set it), the equations for the altered elements of **L** become (for $r \neq p$ and $r \neq q$)

$$\begin{aligned}
l'_{rp} &= l_{rp} - \sin\phi(l_{rq} + \tau l_{rp}) \\
l'_{rq} &= l_{rq} + \sin\phi(l_{rp} - \tau l_{rq}) \\
l'_{pp} &= l_{pp} - t l_{pq} \\
l'_{qq} &= l_{qq} + t l_{pq}
\end{aligned}$$

where

$$\tau = \tan\frac{\phi}{2} \equiv \frac{\sin\phi}{1 + \cos\phi}$$

How do we know that the method actually converges to anything? As proof that the Jacobi method does indeed yield a diagonal matrix, consider the sum of the squares of the off-diagonal elements:

$$S = \sum_{r \neq s} |l_{rs}|^2$$

The transformation implies that, since l_{pq} and, by symmetry, l_{qp} are zero,

$$S' = S - |l_{pq}|^2 - |l_{qp}|^2 = S - 2|l_{pq}|^2$$

The sum decreases monotonically, bounded below by zero, as various elements are annihilated. Since l_{pq} can be chosen arbitrarily, the sequence can be made to converge to zero.

The end product is a diagonal matrix $\mathbf{\Lambda}$ of the eigenvalues of $\mathbf{L}$, since

$$\mathbf{\Lambda} = \mathbf{V}^{\mathsf{T}}\mathbf{L}\mathbf{V}$$

where

$$\mathbf{V} = \mathbf{P}_1\mathbf{P}_2\mathbf{P}_3\cdots\mathbf{P}_n$$

with the $\mathbf{P}_i$'s being successive Jacobi rotation matrices. The columns of $\mathbf{V}$ are the eigenvectors of $\mathbf{L}$ (because $\mathbf{LV} = \mathbf{V\Lambda}$), and $\mathbf{V}$ can be computed by applying $\mathbf{V}_i = \mathbf{V}_{i-1}\mathbf{P}_i$ at each stage of computation, with the initial matrix $\mathbf{V}_0 = \mathbf{I}$. Specifically, for column r,

$$\begin{aligned} v'_{rs} &= v_{rs} \quad (s \neq p, s \neq q) \\ v'_{rp} &= \cos\phi\; v_{rp} - \sin\phi\; v_{rq} \\ v'_{rq} &= \sin\phi\; v_{rp} + \cos\phi\; v_{rq} \end{aligned}$$

In what order should the off-diagonal elements be annihilated? The original strategy used by Jacobi was to choose the largest off-diagonal element and annihilate it, but this technique is not suitable for machine calculation. Instead, the *cyclic Jacobi method* is used, where the elements are zeroed in strict order, circulating through the matrix, proceeding down the rows and across the columns.

Aside from a few accuracy- and performance-enhancing subtleties used by the Numerical Recipes algorithm, that is the essence of the Jacobi method.