Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes

by

Michael Jeffrey Jones

B.A., Computer Science, Mathematics, Rice University, 1990
S.M., Computer Science, M.I.T., 1992

Submitted to the
Department Of Electrical Engineering And Computer Science
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June, 1997

Signature of Author _____
Department of Electrical Engineering and Computer Science, May, 1997

Certified by _____
Tomaso Poggio, Thesis Supervisor

Accepted by _____
Arthur C. Smith, Department Committee on Graduate Students

Multidimensional Morphable Models: A Framework for Representing and Matching Object Classes

by

Michael Jeffrey Jones

Submitted to the Department of Electrical Engineering and Computer Science
on May 20, 1997 in partial fulfillment of the requirements
for the Degree of Doctor of Philosophy
in Electrical Engineering and Computer Science

ABSTRACT

This thesis describes a flexible model for representing images of objects of a certain class, such as faces, and introduces a new algorithm for matching the model to novel images from the class. The model and matching algorithm are very general and can be used for many image analysis tasks.

The flexible model, called a multidimensional morphable model, is learned from example images (called prototypes) of objects of a class. In the learning phase, pixelwise correspondences between a reference prototype and each of the other prototypes are first computed and then used to obtain *shape* and *texture* vectors associated with each prototype. The morphable model is then synthesized as a linear combination that spans the linear vector space determined by the prototypical shapes and textures.

We next introduce an effective stochastic gradient descent algorithm that automatically matches a model to a novel image by finding the parameters that minimize the error between the image generated by the model and the novel image. Several experiments demonstrate the robustness and the broad range of applicability of the matching algorithm and the underlying morphable model.

This approach can provide novel solutions to several computer vision tasks, including the computation of image correspondence, object verification, face recognition, optical character recognition, image synthesis and image compression.

Thesis Supervisor: Professor Tomaso Poggio
Title: Uncas and Helen Whitaker Professor, Dept. of Brain and Cognitive Sciences

ACKNOWLEDGMENTS

First and foremost my thanks to Tommy Poggio for his guidance and support throughout my 7 years as a graduate student. Tommy is the epitome of a scientist and his tireless devotion to his research continues to amaze me. I feel privileged to have worked with him.

My thanks to Eric Grimson for his guidance throughout my graduate student days. I have benefited greatly from Eric's advice and enthusiasm. The great environment that I was lucky enough to be a part of at the MIT Artificial Intelligence Lab owes much to the dedication of Eric.

Also, I thank Paul Viola for all the help he has given me during the course of my research. Paul's advice, comments and suggestions have contributed greatly to this work. His intelligence, personality and energy are a huge addition to the faculty at the AI lab.

Thomas Vetter at the Max Planck Institute in Tubingen, Germany also contributed greatly to this thesis. Our collaboration on the bootstrapping work is a very important part of this thesis. I had a number of enlightening discussions with Michael Oren about this thesis. He suggested the notation that I use in chapters 2 and 3.

I would also like to acknowledge fellow students Charles Isbell, David Beymer, Kah-Kay Sung and Tony Ezzat for helpful discussions throughout my graduate studies.

Thanks to my Magazine St roommates for all the good times while working on my thesis: Tim "Burger King" Tuttle, David "Surly" Bailey, Greg "Hubs" Hubbard, Ari "the girl next door" Drackonakis, Alvin "Sue Everybody" Chin, Josh "Outrageous" Cott, and Matissa "Ski Bum" Hollister. Thanks to Lee "Fry on" Zamir, John "Ro Cham" "Bar" Axon, and James "You are killing me" Sarvis for their friendship and "insightful" critiques of my research. Thanks also to the Posse and my ultimate teammates for providing ample avenues for thesis avoidance.

A big thanks to my parents, Jane and Jeff Jones, for nurturing the scientist in me and for just being great parents.

And finally, thanks to Sarah Sarvis for making it all worthwhile.

This thesis describes research done at the Artificial Intelligence Laboratory and within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences at the Massachusetts Institute of Technology. This research is sponsored by grants from ARPA-ONR under contract N00014-92-J-1879 and from ONR under contract N00014-93-1-0385 and by a grant from the National Science Foundation under contract ASC-9217041 (this award includes funds from ARPA provided under the HPCC program) and by a MURI grant N00014-95-1-0600. The author was supported by an AASERT grant from ONR.

Contents

Chapter 1

Introduction

The field of computer vision is concerned with the problem of how to automatically analyze images to extract information about the objects in them. For example, a typical problem in computer vision is face recognition in which the computer is given an image containing a face and the task is to identify the person. Other typical problems include facial expression recognition, object detection, object tracking in a video sequence and optical character recognition.

In order to solve such problems, it is necessary for the computer to have some knowledge about the world. In face recognition, for example, the computer needs a database containing information about what each person that it is supposed to recognize looks like. What form this knowledge should take is an open question.

One approach to representing knowledge for vision tasks is to use a model for each object or object class in which one is interested. In this case, an input image is analyzed by trying to match the model to the object(s) in the image. A successful match effectively parameterizes the image in terms of the model parameters. Therefore a successful match gives detailed information about the objects in the image. Such an approach is called a model-based approach.

Within the model-based framework there are important questions about how to represent

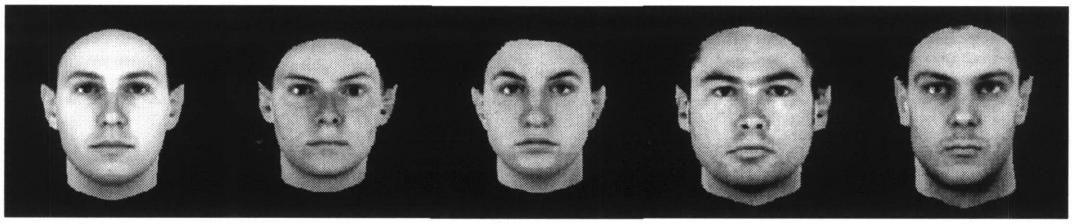


Figure 1.1: *Small example set of 5 face prototypes*

a model and how to match the model to an input image. The main contribution of this thesis is a new framework for modeling object classes and a new matching algorithm for fitting the model to a novel image. The model is called a multidimensional morphable model (or just morphable model). It uses many two-dimensional example images to specify an object class and then uses an iterative gradient descent optimization procedure to match the model to a novel image. One advantage of this model is that new object classes can be represented by giving a number of examples of the class. In addition, the matching algorithm is simple and robust and can be used for many image analysis problems including recognition, verification, tracking and compression.

In addition to the practical advantages of morphable models, there is also convincing psychophysical and even physiological evidence suggesting that the human visual system often uses strategies that have characteristics of object representations based on 2D rather than 3D models ([Edelman and Bulthoff, 1990]; [Sinha, 1995]; [Logothetis *et al.*, 1995]; [Bulthoff *et al.*, 1995]; [Pauls *et al.*, 1996]). This research suggests that 2D models may be a fruitful approach to vision.

The multidimensional morphable model is intended to model object classes as opposed to just individual objects. By object class, we mean a set of objects that are very similar but also have differences in their shape or texture. Some examples of object classes are faces, cars and human hearts.

Briefly, a multidimensional morphable model can be described as follows. First, a number

of example images (called prototypes) are provided for a particular object class. For example, figure 1.1 shows a few prototypes for the class of faces. Next the pixelwise correspondences between one of the prototypes, chosen as the reference, and each of the other prototypes is computed. The computation of correspondences may be done in an automatic way as described in chapter 8 although in some cases it may require manually specifying some corresponding points. The correspondence field for each prototype is considered to be a description of the 2-D shape for that prototype. The texture (grey level intensities) for each prototype is then obtained by normalizing it (using the correspondence field) to have the same shape as the reference image. The normalized textures are simply the prototype images warped such that they are in pixelwise correspondence with the reference image. The morphable model is then the set of all images whose shape is a linear combination of the prototype shapes and whose texture is a linear combination of the prototype textures (as explained in detail in chapters 2 and 3). Once a model is thus defined it can be used to match to novel input images in the following way. An error is defined between the novel image and the current guess for the best fitting model image. This error is then minimized by an iterative gradient descent procedure. Once a novel image is matched by a model, the resulting model parameters (the linear coefficients of the shape and texture) can be used for a variety of applications. Notice that the matching algorithm uses image *synthesis* to perform *analysis*.

The ideas which lead to the formulation of morphable models are rooted in the work of [Poggio and Vetter, 1992], [Vetter and Poggio, 1995], [Beymer *et al.*, 1993] (see also [Beymer and Poggio, 1996]), [Ullman and Basri, 1991] and [Shashua, 1992]. The "linear class" idea of [Poggio and Vetter, 1992] and [Vetter and Poggio, 1995] together with the image representation, based on pixelwise correspondence, used by [Beymer *et al.*, 1993] (see also [Beymer and Poggio, 1996]) is the main motivation for this work. Poggio and Vetter introduced the idea of linear combinations of views to define and model *classes* of objects. They were inspired in turn by the results of [Ullman and Basri, 1991] and [Shashua, 1992] who showed that linear combinations of three views of a *single* object may be used to obtain

any other views of the object (barring self-occlusion and assuming orthographic projection). Poggio and Vetter defined a linear object class as a set of 2D views of objects which cluster in a small linear subspace of $\mathcal{R}^{2n}$ where n is the number of feature points on each object. They showed that in the case of linear object classes under orthographic projection, rigid transformations can be learned exactly from a small set of examples. Furthermore, other object transformations (such as the changing expression of a face) can be approximated by linear transformations. In particular, they used their linear model to generate new *virtual* views of an object from a single (example) view. The focus of the work of [Poggio and Vetter, 1992] and [Vetter and Poggio, 1995] is image synthesis as opposed to the emphasis on image analysis in this thesis.

Much of the work in this thesis has been reported in [Jones and Poggio, 1995] which describes the shape model, [Jones and Poggio, 1996] which describes the full morphable model and [Vetter *et al.*, 1997] which describes a bootstrapping algorithm for automatically finding correspondences among the prototypes.

There are many other approaches to modeling objects and object classes. We will discuss a few of these approaches to give some perspective on what has been done before and how this thesis differs.

There have been a number of approaches that build models based on deformable curves. Such approaches attempt to model the contours or edges in an image. The work of [Kass *et al.*, 1988] on snakes is a well known and influential example. A snake is a spline which when placed on an image is attracted to features like lines and edges. They can be used for edge detection, motion tracking and stereo matching. To model a person's lips for example, a snake could be defined with the basic shape of a pair of lips. When placed on an image containing a particular pair of lips, the snake fits itself to the lip contours thus parameterizing the shape of the lips. The snake model could also be used to track the lips in a video sequence.

A similar approach known as deformable templates was formulated by [Yuille *et al.*, 1992]. Their models also consist of curves although they are more constrained than snakes in the

ways they can move. The idea is that a deformable template for an eye should only be allowed to deform in ways that still produce eyes. Their deformable templates are constructed by hand by defining an energy functional which can be a tedious process.

The work of [Blake and Isard, 1994] is another noteworthy approach which models shapes in terms of splines which are constrained in the ways they can move on an image. They use a Kalman filter based matching algorithm which can track objects (such as hands or lips) in a video sequence in real time.

The main differences with these contour-based approaches and morphable models is that the contour models do not try to capture any texture information and they are designed by hand as opposed to being defined by examples. The main advantage of such approaches is that they are relatively fast and quite adequate for modeling low level features such as the edges of an object class.

The work of [Kirby and Sirovich, 1990] and [Turk and Pentland, 1991] took the opposite approach from the contour-based work which models only shape. They attempted to model just the texture or grey-level information in an image. [Turk and Pentland, 1991] popularized the approach under the name "eigenfaces". The idea is to use many example images of faces to define a model. The model is simply a linear combination of the example images. Instead of using all of the examples in the linear combination, only the first few eigenvectors are used. These are called the eigenfaces. Pixelwise correspondences among the example images are not used and thus the model does not strictly speaking have the properties of a vector space. A new face image is matched by finding the linear combination of eigenfaces which best reconstructs the input face. The coefficients of the resulting match can then be used for recognition. This technique is fast and works well on novel faces which can be easily aligned with the example faces to correct for rotations, translations and scale. It has trouble handling different backgrounds with novel faces as well as occlusions in the input image. The morphable model which takes into account the changes in shape among the example images in addition to changes in texture is much more robust to changes in the input image such as

affine transformations and occlusion (see chapter 9).

There has been some recent work which combines both shape and texture information into a single framework. [Choi *et al.*, 1991] use example face images to build a model which consists of linear combinations of example shapes and example textures. They use a 3-D head model to help find correspondences among the example faces. The 3-D head model is first fit to an input face by using 3-D affine transformations and contour fitting in order to get the correspondences to the novel face. Then a set of linear equations is solved to find the best linear combinations of shape and texture to match the novel face. This work has many of the elements of the morphable model. The main difference is that they use a 3-D head model to help find correspondences. This undermines the advantage of using 2-D examples to define a model. They use the same technique (3-D head model) for finding correspondences to novel input images which obviates the need for a matching algorithm.

Taylor and coworkers ([Cootes and Taylor, 1992]; [Cootes and Taylor, 1994]; [Cootes *et al.*, 1992]; [Cootes *et al.*, 1994]; [Cootes *et al.*, 1993]; [Hill *et al.*, 1992]; [Lanitis *et al.*, 1995]) developed a model known as an active shape model to represent object classes. Their example-based model includes both shape and texture but fits them separately to a novel input image. Shape in their framework consists of the positions of a sparse set of feature points in each example image. These points are typically specified by hand. This results in a contour model for the shape which is fit to a novel input image. Once the shape model is fit, the input face is normalized to a canonical shape. The texture of the resulting normalized input image is then reconstructed as a linear combination of normalized example images. Unlike active shape models, morphable models use a dense correspondence field to model shape changes. Also, shape and texture are combined seamlessly into a single framework as opposed to fitting the texture after the shape has been matched. Finally, morphable models use a different matching algorithm than active shape models. However, this is the work closest in spirit to ours.

Hallinan [Hallinan, 1995] also proposed a framework for modeling faces which includes

changes to the shape as well as the texture. Hallinan does not use examples to build a shape model. Instead he uses a deformable template for the face. His main focus is on modeling illumination changes using example images. His work shows that it is practical to model illumination changes using only about a small number of example faces with various illuminations. We use a similar technique to model illumination in chapter 7.

Another paper which proposes a similar technique for modeling object classes is [Nastar *et al.*, 1996]. They treat an image as a deformable 3-D mesh and then use prototypical warps to constrain how the 3-D mesh can be deformed for a particular object class. Their technique does not incorporate the example textures (only the warps). The matching algorithm they use is also different from the one we use for morphable models.

The work of Choi *et al.*, Taylor *et al.*, Hallinan and Nastar *et al.* are similar in that their methods all *synthesize* model images in order to *analyze* novel input images. There has also been a good deal of work (especially on object recognition) which is not based on synthesizing images. These techniques generally find certain features in an input image (such as lines and corners) and use these features to index into a database of feature vectors for known objects in order to determine if the input image contains features which are very similar to features of objects in the database (see for example [Roberts, 1966], [Biederman, 1985], [Grimson, 1990], [Jacobs, 1992]). Such methods are more suited to recognizing an object from a large database of known objects than the synthesis/analysis paradigm simply because of the computational complexity of matching a large number of models to an input image as required by the synthesis/analysis approach. The synthesis/analysis approach is intended for situations in which one is concerned with only one or a few different object classes.

Chapter 2

Shape Model

Before we describe the full multidimensional morphable model in detail, we will first explain the shape model which is a major part of the morphable model. The shape model has interesting applications of its own, for example for matching contours in images and for optical character recognition.

The shape model for a particular object class is defined from a number of prototypical example images which show many of the different shapes that instances of the object class can take. Since we are only interested in the shapes for now and not the textures, we will use line drawings for the example images. Figure 2.1 shows an example of a small set of prototype images to define a class of cartoon faces.

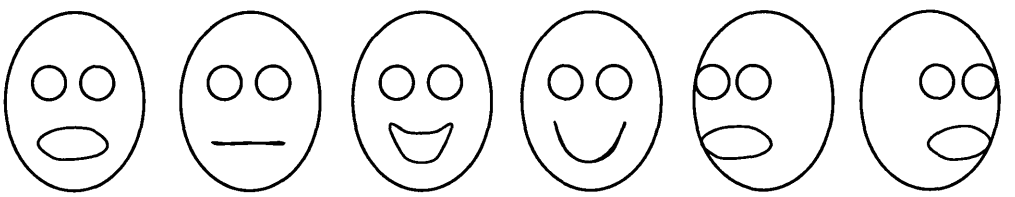


Figure 2.1: *Small example set of 6 prototypes for a class of cartoon faces*

2.1 Representing shape as a flow field

In order to represent how the shape changes among the different prototypes, we need to know how the features of the object class move from one prototype to the next. Also, we need to decide what exactly to use as "features". Instead of trying to track a sparse number of features such as corners or T junctions (which may not be invariant over the entire set of possible example images), we choose to use the densest possible feature set. We will consider every pixel in the example images to be a feature ([Beymer *et al.*, 1993], [Beymer and Poggio, 1996]). To represent shape changes we choose one of the prototypes as a *reference* prototype and compute the pixelwise correspondences from the reference prototype to each of the other prototypes. The pixelwise correspondences are stored in the computer as a flow field which is simply a matrix of x, y displacements which tell for each pixel in the reference image where it corresponds in a prototype image. There is one flow field for each prototype image (not including the reference image).

Knowing the pixelwise correspondences from the reference image to another prototype image allows one to warp the reference image onto the prototype. This means that the pixels of the reference image can be moved according to the flow field and can yield any image between the reference and prototype images. In other words, the correspondences from a reference cartoon face with an open mouth to a cartoon face with a closed mouth can be used to render cartoon faces with partially open mouths by warping according to a fraction of the flow field.

Since we are only concerned here with line drawings and since only the black pixels are important in a line drawing, one may wonder why we need a full set of correspondences which includes the white pixels. The answer is two-fold. First, the matching algorithm which is described later will use blurred line drawings which makes more of the pixels relevant. Also, when we extend these ideas to real grey-level images, all of the pixels are important and the current formulation can be used without change.

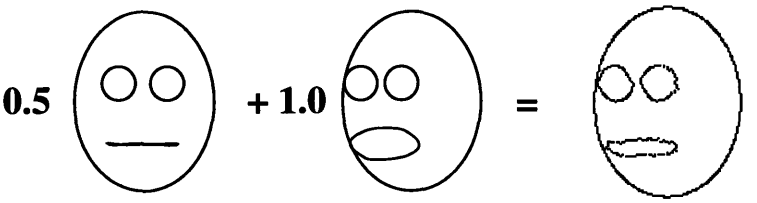


Figure 2.2: *The rightmost cartoon face was obtained as a linear combination of the examples on the left in the following way. The flow field from the reference image (not shown) to each of the two prototypes on the left was given. One half of the flow field for the first example plus the flow field for the second example was computed. This summed flow field was used to warp the reference image, yielding the cartoon face on the right. The resulting cartoon face has holes due to the warping algorithm which is why it is "rougher". This example shows that adding together half of the "closed mouth" prototype plus the "rotate left" prototype yields a cartoon face which is rotated left with a half closed mouth.*

2.2 Linear combinations of shape

To encompass all the desired shapes that a particular image from an object class may take the shape model should include more than just the images "inbetween" the reference image and one other prototype. We want to allow all linear combinations of prototypical shapes. For example, figure 2.2 shows a valid model image obtained by linearly combining two prototype shapes. We would like the model to include all such linear combinations.

2.3 Formal specification of the shape model

In this section we will formally specify the shape model. An image I is viewed as a mapping

$$I : \mathcal{R}^2 \rightarrow \mathcal{I}$$

such that $I(x, y)$ is the intensity value of point (x, y) in the image. $\mathcal{I} = [0, a]$ is the range of possible grey level values. For eight bit images, $a = 255$. (For black and white line drawings the pixels have either the value 0 or 255.) Here we are only considering grey level images, although color images could also be handled in a straightforward manner. In practice images

are discrete and bounded, but we are assuming them to be continuous and unbounded for simplicity of notation. To define a model, a set of example images called prototypes are given. We denote these prototypes as $I_0, I_1, \ldots, I_N$.

Let I_0 be the reference image. The pixelwise correspondences between I_0 and each example image are denoted by a mapping

$$S_j : \mathcal{R}^2 \rightarrow \mathcal{R}^2$$

which maps the points of I_0 onto I_j, i.e. $S_j(x, y) = (\hat{x}, \hat{y})$ where $(\hat{x}, \hat{y})$ is the point in I_j which corresponds to (x, y) in I_0. We refer to S_j as a *correspondence field* or *flow field* and also as the shape vector for I_j.

The shape model is defined as the set of images I^{model}, parameterized by $\mathbf{c} = [c_0, c_1, \ldots, c_N]$, such that

$$I^{model} \circ (\sum_{i=0}^{N} c_i S_i) = I_0 \tag{2.1}$$

where the notation $I \circ S_i$ denotes function composition, i.e. $I \circ S_i = I(S_i)$.

The summation $\sum_{i=0}^{N} c_i S_i$ describes the shape of every model image as a linear combination of the prototype shapes.

In order to allow the model to handle translations, rotations, scaling and shearing, a global affine transformation is also added. The equation for the model images can now be written

$$I^{model} \circ (A \circ \sum_{i=0}^{N} c_i S_i) = I_0 \tag{2.2}$$

where $A : \mathcal{R}^2 \rightarrow \mathcal{R}^2$ is the 2D affine transformation

$$A(x, y) = (p_0 x + p_1 y + p_2,\ p_3 x + p_4 y + p_5). \tag{2.3}$$

We assume that the coordinate system of the prototype images has its origin at the center of the image so that rotations are about the center of the image. The constraint $\sum_{i=0}^{N} c_i = 1$ is

imposed to avoid redundancy in the parameters since the affine parameters allow for changes in scale. The parameters of the model are the c_i's and p_k's.

When used as a generative model, for given values of $\mathbf{c}$ and $\mathbf{p}$, a model image is rendered by computing I^{model} in equation 2.2. For analysis the goal is, given a novel image I^{novel}, to find parameter values that generate a model image as similar as possible to I^{novel}. The next section describes how.

2.4 Matching the model

The analysis problem is the problem of matching the morphable model to a novel image. The general strategy is to define an error between the novel image and the current guess for the closest model image. We then try to minimize this error with respect to the linear coefficients $\mathbf{c}$ and the affine parameters $\mathbf{p}$. Following this strategy, we define the sum of squared differences error

$$E(\mathbf{c},\mathbf{p}) = \frac{1}{2}\sum_{x,y}[I^{novel}(x,y) - I^{model}(x,y)]^2 \tag{2.4}$$

where the sum is over all pixels (x,y) in the images, I^{novel} is the novel line drawing being matched and I^{model} is the current guess for the model line drawing. Equation 2.2 suggests to compute I^{model} working in the coordinate system of the reference image. To do this we simply apply the shape transformation (given estimated values for $\mathbf{c}$ and $\mathbf{p}$) to I^{novel} and compare it to the shape-free model, that is

$$E(\mathbf{c},\mathbf{p}) = \frac{1}{2}\sum_{x,y}[I^{novel} \circ (A \circ \sum_{i=0}^{N} c_i S_i)(x,y) - I^{model} \circ (A \circ \sum_{i=0}^{N} c_i S_i)(x,y)]^2. \tag{2.5}$$

To simplify this equation we define

$$\overline{S}(x,y) = (A \circ \sum_{i=0}^{N} c_i S_i)(x,y) \tag{2.6}$$

and also use equation 2.2 to yield

$$E(\mathbf{c},\mathbf{p}) = \frac{1}{2}\sum_{x,y}[I^{novel} \circ \overline{S}(x,y) - I_0(x,y)]^2. \tag{2.7}$$

Minimizing this error yields the model image which best fits the novel image with respect to the L_2 norm. We use here the L_2 norm but other norms may also be appropriate (e.g. robust statistics).

In order to minimize the error function any standard minimization algorithm could be used. We have chosen to use the stochastic gradient descent algorithm [Viola, 1995] because it is fast and can avoid remaining trapped in local minima. We describe this algorithm in more detail in chapter 4.

There are two important techniques which we use to improve the robustness of the matching algorithm. The first is to overcome the fact that the prototypes and input image are all black and white line drawings which means that there is little gradient information in the image. To solve this problem, we simply blur the novel image. This means we blur the black pixels onto neighboring pixels according to a gaussian distribution.

The second technique we use to improve robustness is to create a pyramid representation of the model [Burt, 1984]. This requires creating an image pyramid for each flow field and for the reference image as well as the novel image. The result is a shape model for each different level of the pyramid. The matching is then done first for the coarsest level and the resulting match is used to initialize the matching algorithm at the next higher level of resolution. This coarse-to-fine approach is explained in more detail in chapter 4.

Stochastic gradient descent requires the derivative of the error with respect to each parameter. These derivatives can be calculated straightforwardly and are given for the full morphable model in chapter 4.

One important point that may not be obvious is that the minimization algorithm does not need to deal with the problem of "holes" in the model image. A hole occurs after warping when a pixel in the model image is not mapped to (i.e. the flow field from reference to model image is not onto). The matching algorithm that we have presented does not render a full model image at each iteration to compare with the novel image (although this may be a useful way to think about what the algorithm is doing). Instead a small number of

pixels in the reference image are randomly chosen at each iteration and the guess for the corresponding point in the novel image is computed according to the linear combination of flow fields (plus the affine transformation). The corresponding point may not be integer valued in which case the gray scale value can be interpolated from neighboring pixels or the corresponding point can be rounded. In either case, the gray scale value of this point is then compared to the gray scale value of the reference image pixel to yield an error. Then the gradients with respect to each model parameter are computed for this pixel. This is done for each random pixel selected by the stochastic gradient descent algorithm.

2.5 Computing the correspondences for the prototypes

Before we show some example shape models, we should first explain how the pixelwise correspondences between the reference prototype and each of the other prototypes are obtained. Chapter 8 describes an automatic technique which uses a bootstrapping idea to compute correspondences. However, the bootstrapping algorithm does not work for all sets of prototypes. In such cases we have other semi-automatic techniques. One technique for line drawings is to use a drawing program which represents line drawings as Bezier curves. The reference image is drawn first and saved. Then to get any other prototype, the control points of the Bezier curves for the reference image are moved around. No Bezier curves are added or removed. To find the pixelwise correspondences between the reference and prototype images along the Bezier curves, one simply traces over each pair of corresponding Bezier curves (which are parameterized by time) and records pixels that occur at the same time step as corresponding. The Bezier curves are sampled at a constant velocity. This procedure yields the correspondences for the pixels along the Bezier curves. To find the other correspondences, a simple interpolation algorithm is used to estimate unknown correspondences by looking at nearby pixels whose correspondence is known.

Another technique we use for line drawings when the Bezier curve representation is not

available is to simply specify a sparse set of corresponding pixels by hand and then to interpolate to get a dense flow field as described above.

2.6 Examples

The following three examples show shape models for cartoon faces, stick figures and hand-written "2's". In all of these examples, the pixelwise correspondences were obtained by the Bezier curve technique described above.

2.6.1 Cartoon faces

Figure 2.3 shows a set of prototype images used to define a shape model for a class of cartoon faces. This model can then be fit to novel drawings of similar cartoon faces using the matching algorithm described in section 2.4. The model includes examples of the cartoon face with an open mouth, with a closed mouth, with a smile and with head rotation to the left and right. There is also an example in which the eyes are thin in the vertical axis as well as a separate example in which the eyes are thin in the horizontal axis. Together these two examples can be used to synthesize examples in which the eyes are thin in both axes which means the eyes can become very small circles. Also, note that there are some seemingly strange examples (such as one eye being higher than another). These are included because we want to use the model to match novel cartoon faces which people may draw. Since people often draw rather strange faces (perhaps to try to crash the system), we have tried to anticipate such challenges.

Figure 2.4 shows some novel cartoon faces which were used as input to the matching algorithm and the resulting best matches of the model. One can see that the matching algorithm does a good job of finding a closest fitting model image (which is a linear combination of the prototype shapes). Note the last two novel images. Both have head shapes that cannot be recontructed from the prototype images. The matching algorithm still does a good job

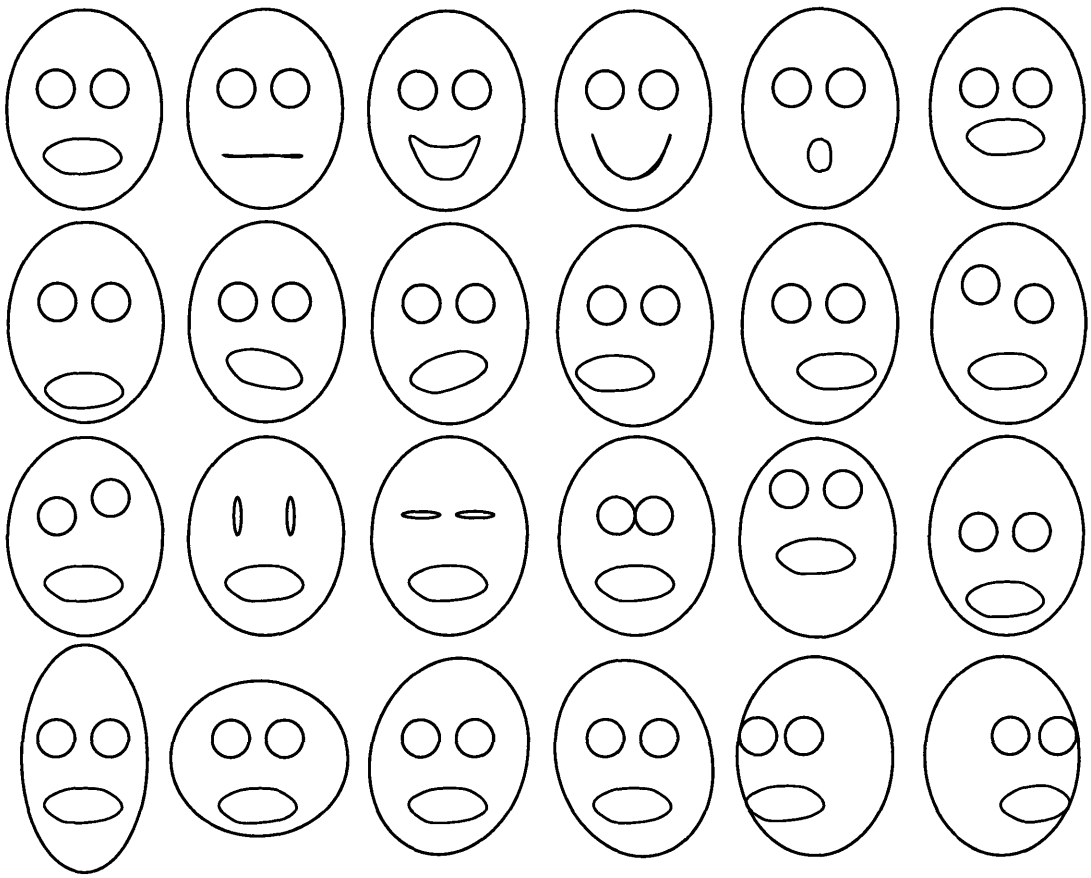


Figure 2.3: *Prototypes for a class of cartoon faces. The prototype in the upper left corner was chosen as the reference prototype. The pixelwise correspondences from the reference prototype to each of the other prototypes was computed using the Bezier curve technique described above.*

of finding the best fitting oval for each of these heads.

This example also illustrates a potential application: facial expression recognition. Although this application is much more useful with real images of faces, this cartoon example illustrates the idea. We can estimate how much smile the novel face has by simply looking at the coefficients of the model after matching. Since only the third and fourth prototypes are examples of smiling faces, the coefficients of these two prototypes tell to what degree the novel cartoon face is smiling. Similarly for open mouth (only prototypes 2 and 4 have closed mouths) and for pose (the last two prototypes deal with head rotations). These estimated

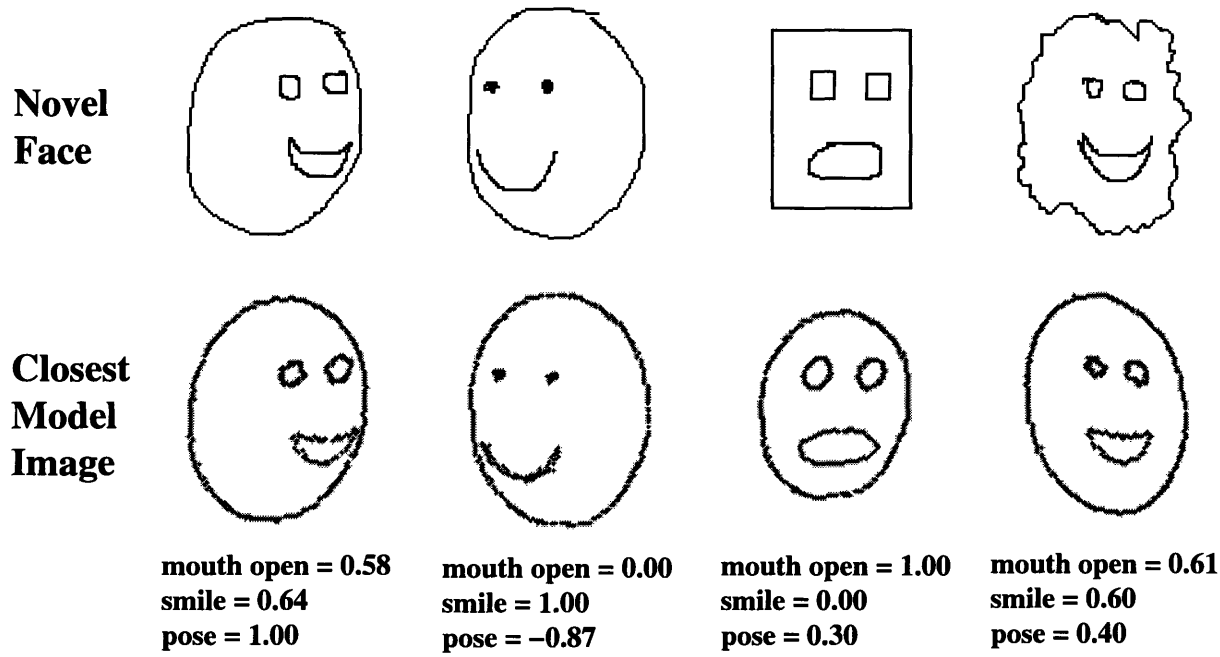


Figure 2.4: *Matches to four novel cartoon face images. An expression recognition application is illustrated by the facial expression parameters listed after each example which were computed automatically as described in the text.*

expression parameters are listed below each match in figure 2.4. They vary from 0 to 1 for the smile and open mouth parameters and from -1 (rotated left) to +1 (rotated right) for the pose parameters.

Although this example is intended mainly to illustrate the shape model and matching algorithm, it might be useful as a practical application. Suppose you have a large database of real face images with different expressions and you want to be able to retreive images based on their facial expressions. Then an interesting user interface might use a cartoon face model to index into the database of real faces. The idea is that a user could draw a cartoon face which has roughly the expression that he wants in a real face image. The matching algorithm could use the cartoon face shape model to analyze the expression parameters of the cartoon face and then use these expression parameters to index into the database and retrieve the desired real face images (which have been labelled according to their expression).

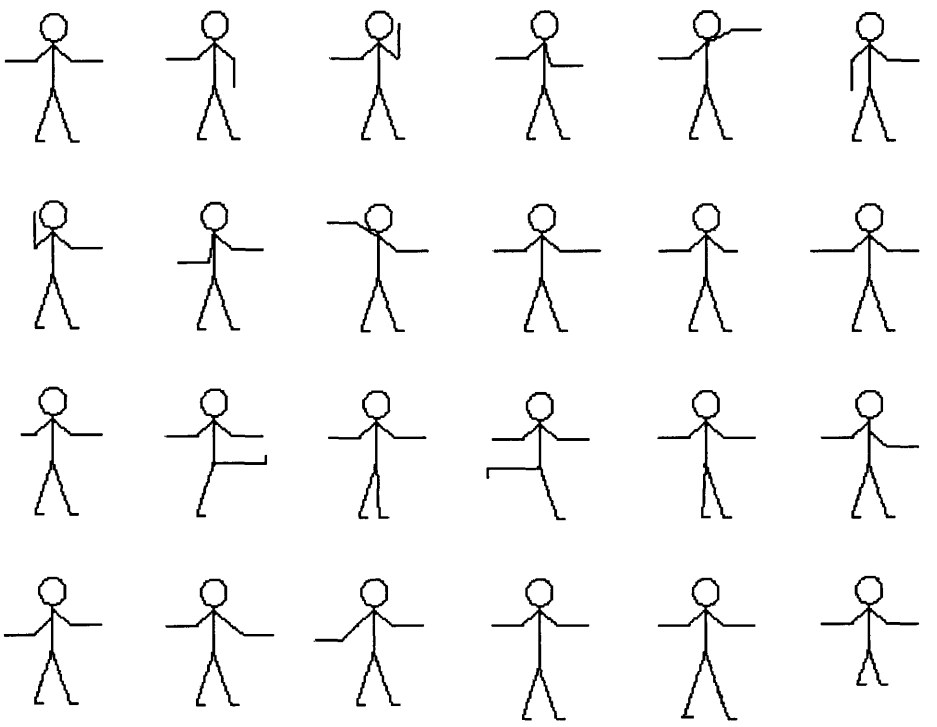


Figure 2.5: *Prototypes for a class of stick figures. The prototype in the upper left corner was chosen as the reference prototype. The pixelwise correspondences from the reference prototype to each of the other prototypes was computed using the Bezier curve technique described above.*

2.6.2 Stick figures

Figure 2.5 shows a set of prototypes which define a simple class of stick figures. The prototype in the upper left corner was chosen as the reference prototype. The pixelwise correspondences from this reference prototype to each of the other prototypes was computed using the Bezier curve technique described in section 2.5. Figure 2.6 shows four novel stick figures and the best matching model images found by the matching algorithm. The novel stick figures are matched very well by the model.

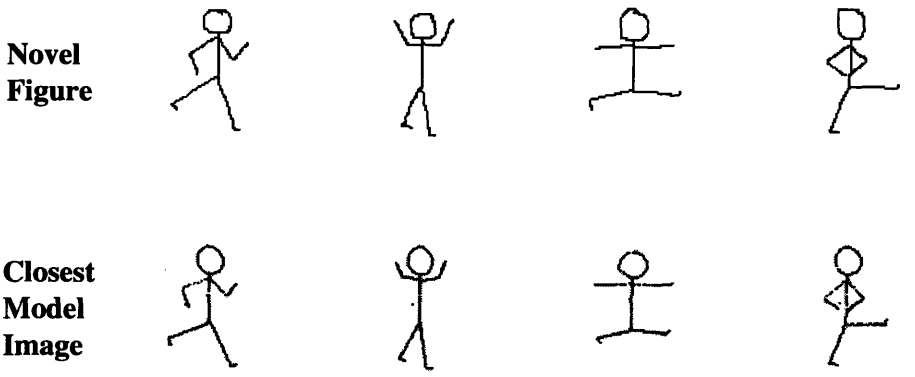


Figure 2.6: *Four examples of matching the stick figure model to novel stick figures.*

2.6.3 OCR example

Another application of the shape model is optical character recognition (OCR). The shape model can be used for OCR in the following way. A shape model for each character is created from examples. Figure 2.7 shows a small set of prototypes for the numeral "2". These prototypes were used to form a shape model for twos. For a full digit recognition application, one would need such a model for each digit. To classify an input digit, each of the digit models would be fit independently to the input. The shape model which best fit the input digit would determine which digit it is. In addition to looking at the L_2 error of the matches, one could also analyze the coefficients of each of the models after matching to determine unlikely sets of parameters and use this also to rule out some matches. This idea is explored in more detail in section 4.6. Figure 2.8 shows some example matches for the twos shape model to illustrate the idea.

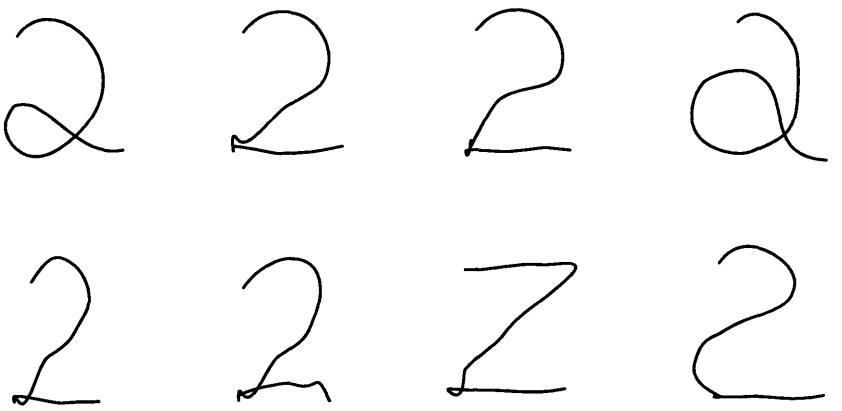


Figure 2.7: *Prototypes for a class of twos. The prototype in the upper left corner was chosen as the reference prototype. The pixelwise correspondences from the reference prototype to each of the other prototypes was computed using the Bezier curve technique described above.*

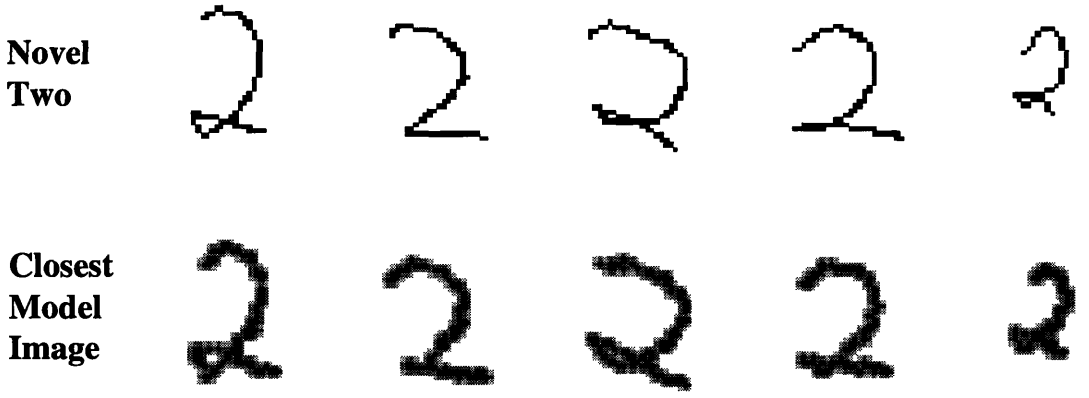


Figure 2.8: *Five examples of matching to novel twos.*

Chapter 3

Multidimensional Morphable Models

In this chapter we will formally introduce the multidimensional morphable model for representing object classes. In the next chapter we will describe a matching algorithm for matching the morphable model to a novel image and thus performing image analysis.

3.1 Vector representation

The morphable model is based on linear combinations of a specific representation of example images. In order for a linear combination of image representations to make sense, the representation must have the properties of a vector. In particular, adding two image representations together must yield another representation of an image from that object class. We argue that in order to treat the representation of images as vectors, the example images must be in pixelwise correspondence. For images considered as bitmaps, operations like addition and linear combination are not meaningful. A better way to represent images is to associate with each example image a shape vector and a texture vector (see for instance [Beymer and Poggio, 1996]).

The *shape vector* (as described in chapter 2) of an example image associates to each pixel in the reference image the displacement of the corresponding point in the example image. The *texture vector* contains for each pixel in the reference image the grey level or color

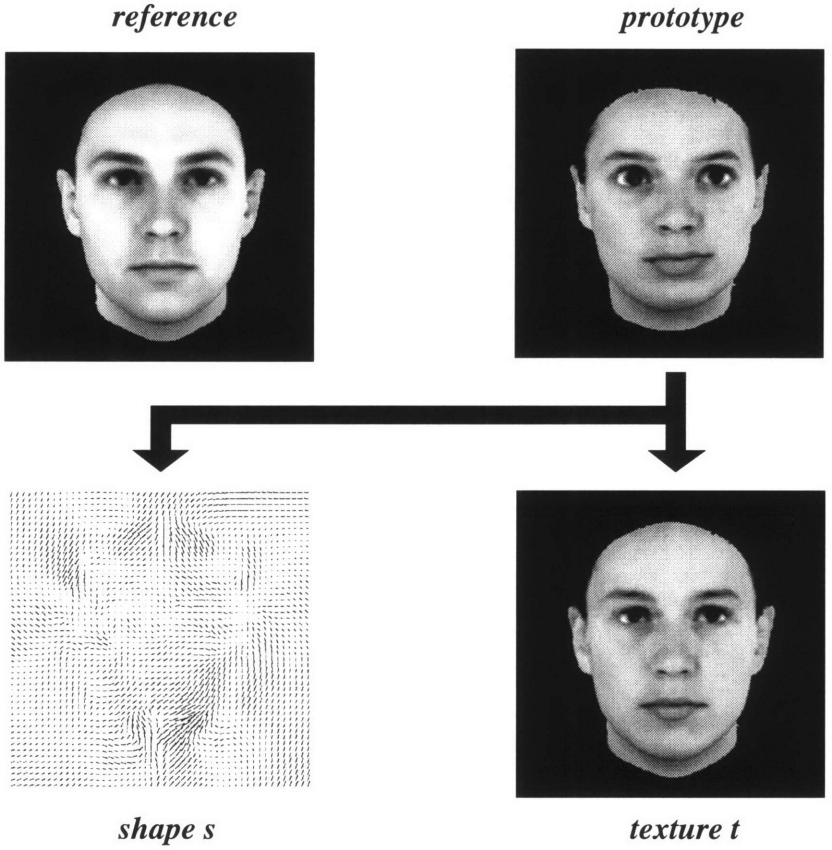


Figure 3.1: *An* $(\mathbf{s}, \mathbf{t})$ *vectorized image representation. Pixelwise correspondences are computed between image* prototype *and a standard reference image. Shape* $\mathbf{s}$ *consists of the* $(\Delta x, \Delta y)$ *displacements between each pixel in the reference image relative to its corresponding position in the prototype image. The shape* $\mathbf{s}$ *is the flow field (illustrated as a matrix of vectors) that specifies how the pixels of the* reference *image move to get to their corresponding points in the* prototype *image. The texture* $\mathbf{t}$ *is the texture of* prototype *backward warped to the shape of* reference *image.*

value for the corresponding pixel in the example image. Figure 3.1 illustrates the vectorized representation for a face prototype. We refer to the operations which associate the shape and texture vectors to an image as *vectorizing the image.* The shape vector is typically obtained by an optical flow algorithm or by the more powerful bootstrapping algorithm described in chapter 8. The texture vector is obtained by backward warping (see Appendix B) the pixels of the prototype image onto the reference shape by using the already computed shape vector.

Instead of two separate vectors we can also consider the full texture-shape vector which has dimensionality $3N$ where N is the number of pixels in the image. There is one component per pixel for the texture (assuming grey-scale images) and 2 components per pixel for the x and y displacement of the shape. The term shape vector refers to the 2D shape (not 3D!) relative to the reference image. Note that edge or contour-based approaches are a special case of this framework. If the images used are edge maps or line drawings then the shape vector may include only entries for points along an edge without the need of an explicit texture vector.

The shape and texture vectors form separate linear vector spaces with specific properties. The shape vectors resulting from different orthographic views of a single 3D object (in which features are always visible) constitute a linear vector subspace of very low dimensionality spanned by just two views ([Ullman and Basri, 1991]; see also [Poggio, 1990]). For a fixed viewpoint a specific class of objects with a similar 3D structure, such as faces, seems to induce a texture vector space of relatively low dimensionality as shown indirectly by the results of [Kirby and Sirovich, 1990] and more directly by [Lanitis *et al.*, 1995]. Using pixelwise correspondence [Vetter and Poggio, 1995] and [Beymer and Poggio, 1995] showed that a good approximation of a new face image can be obtained with as few as 50 base faces, suggesting a low dimensionality for both the shape and the texture spaces. As reviewed by [Poggio and Beymer, 1996] correspondence and the resulting vector structure underlie many of the recent view-based approaches to recognition and detection either implicitly or explicitly.

Certain special object classes (such as cuboids and symmetric objects) can be proved to be exactly linear classes (see [Poggio and Vetter, 1992]). Later we will show that there are classes of objects the images of which – for similar view angle and imaging parameters – can be represented satisfactorily as a linear combination of a relatively small number of prototype images.

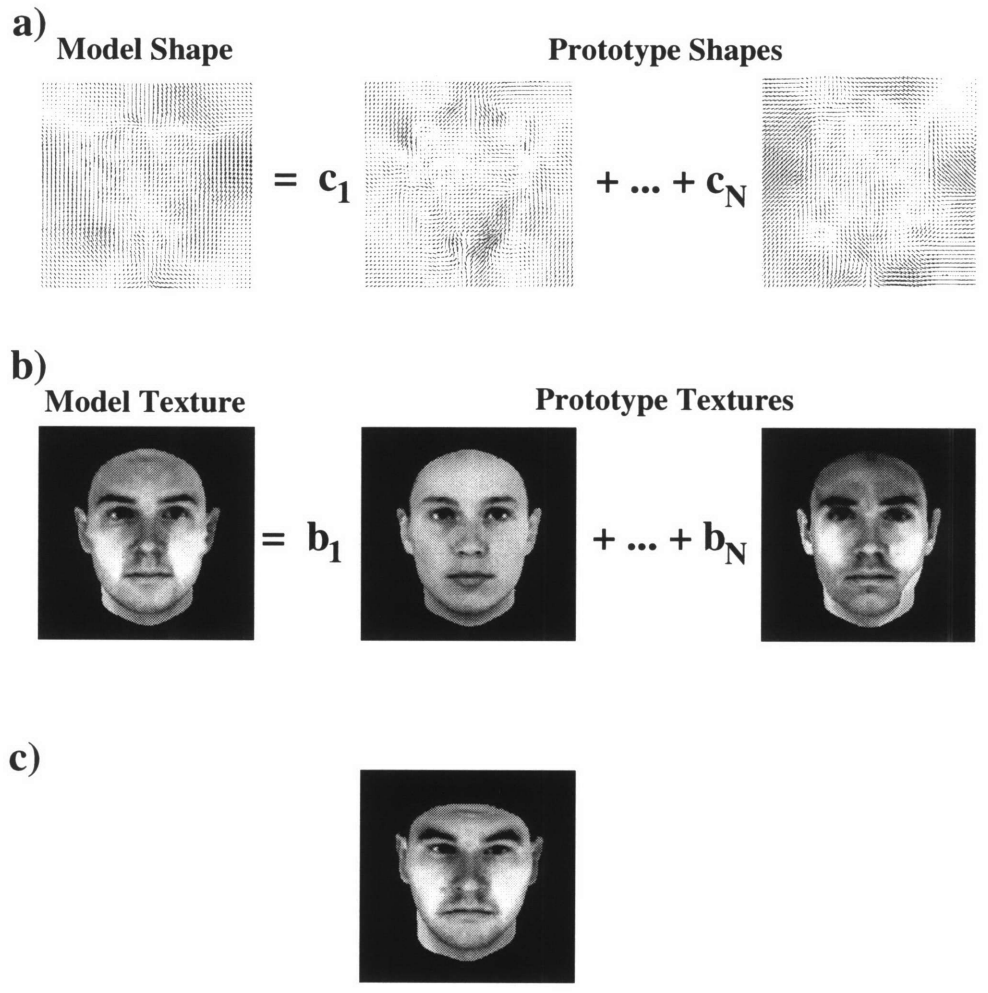


Figure 3.2: *a) The prototype shapes are linearly combined according to the coefficients* **c** *to yield a model shape. b) Similarly, the prototype textures are linearly combined according to the coefficients* **b** *to yield a model texture. c) Finally, a model image is rendered by forward warping the model shape according to the model texture.*

3.2 Linear combinations of shape and texture

Given the vectorized representations for a set of prototype images describing some object class, the associated morphable model is the set of all images whose shape (relative to the reference image) is constrained to be a linear combination of prototype shapes and whose texture is constrained to be a linear combination of the prototype textures. Figure 3.2 illustrates the model. A particular model image is synthesized by independently choosing

some setting of the linear coefficients for the shape and texture. A model shape vector (flow field) is then obtained by a weighted sum of prototype shape vectors. Likewise, a model texture vector is obtained by a weighted sum of prototype texture vectors. The model image can then be rendered by forward warping the model texture vector according to the model shape vector (see Appendix B for a simple warping algorithm).

3.3 Formal specification of the model

In this section we will formally specify our multidimensional morphable model. This formulation follows the formulation of the shape model in chapter 2 but extends it to include texture.

As before, an image I is viewed as a mapping

$$I : \mathcal{R}^2 \rightarrow \mathcal{I}$$

such that $I(x, y)$ is the intensity value of point (x, y) in the image. $\mathcal{I} = [0, a]$ is the range of possible grey level values. Let $I_0, I_1, \ldots, I_N$ be the prototype images. A naive approach might model this class of images using a linear combination of the images ([Turk and Pentland, 1991]) as follows:

$$I^{model} = \sum_{i=0}^{N} b_i I_i. \tag{3.1}$$

This approach does not result in good matches as shown in figure 3.3. The underlying reason for this method's poor performance is that the example images are not in pixelwise correspondence. Our image representation is instead based on pixelwise correspondences.

Let I_0 be the reference image. As with the shape model, the pixelwise correspondences between I_0 and each example image are denoted by a mapping

$$S_j : \mathcal{R}^2 \rightarrow \mathcal{R}^2$$

which maps the points of I_0 onto I_j, i.e. $S_j(x, y) = (\hat{x}, \hat{y})$ where $(\hat{x}, \hat{y})$ is the point in I_j which corresponds to (x, y) in I_0.

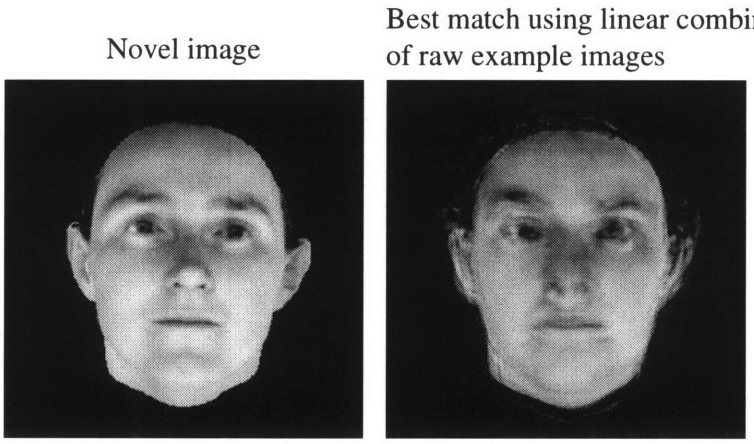


Figure 3.3: *Example of a novel image matched using a linear combination of raw example images. The example images consisted of 100 faces, which are shown in figure 5.1 - 5.3. The best match is blurry using this model because the example images are aligned and scaled but are not in pixelwise correspondence.*

We define the mapping $T_j : \mathcal{R}^2 \rightarrow \mathcal{I}$ as

$$T_j(x, y) = I_j \circ S_j(x, y) = I_j(S_j(x, y)). \tag{3.2}$$

T_j is the backward warping of image I_j onto the reference image I_0. In other words, $\{T_j\}$ is the set of shape-free prototype images – shape free in the sense that their shape is the same as the shape of the reference image. The idea of the model is to combine linearly the textures of the prototypes all warped to the shape of the reference image and therefore in correspondence with each other. The resulting texture vector can then be warped to any of the shapes defined by the linear combination of prototypical shapes.

More formally the flexible model is defined as the set of images I^{model}, parameterized by $\mathbf{b} = [b_0, b_1, \ldots, b_N], \mathbf{c} = [c_0, c_1, \ldots, c_N]$, such that

$$I^{model} \circ (\sum_{i=0}^{N} c_i S_i) = \sum_{j=0}^{N} b_j T_j. \tag{3.3}$$

The summation $\sum_{i=0}^{N} c_i S_i$ describes the shape of every model image as a linear combination of the prototype shapes. Similarly, the summation $\sum_{j=0}^{N} b_j T_j$ describes the texture of every

model image as a linear combination of the prototype textures. Note that the coefficients for the shape and texture parts of the model are independent.

Again, as we did with the shape model, we wish to allow the model to handle translations, rotations, scaling and shearing. Therefore, a global affine transformation is also added. The equation for the model images can now be written

$$I^{model} \circ (A \circ \sum_{i=0}^{N} c_i S_i) = \sum_{j=0}^{N} b_j T_j \tag{3.4}$$

where $A : \mathcal{R}^2 \rightarrow \mathcal{R}^2$ is the 2D affine transformation

$$A(x, y) = (p_0 x + p_1 y + p_2,\ p_3 x + p_4 y + p_5). \tag{3.5}$$

The two linear combinations, for shape and texture respectively, use the same set of prototype images but two different sets of coefficients. The parameters of the model are the c_i's, b_j's and p_k's.

Now that we have formally defined the multidimensional morphable model, we want to use it for image analysis by matching it to novel images. The matching algorithm is a straightforward extension of the matching algorithm for shape models, and is decribed in the next chapter.

Chapter 4

Matching a Multidimensional Morphable Model

The matching algorithm for morphable models is basically the same as the algorithm for shape models. Again, we define an error between the novel image and the current guess for the closest model image. We then minimize this error with respect to the linear coefficients $\mathbf{c}$ and $\mathbf{b}$ and the affine parameters $\mathbf{p}$.

4.1 Minimizing an error function

Following this strategy, we define the sum of squared differences error

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel}(x, y) - I^{model}(x, y)]^2 \tag{4.1}$$

where the sum is over all pixels (x, y) in the images, I^{novel} is the novel gray level image being matched and I^{model} is the current guess for the model gray level image. As with the shape model, we apply the shape transformation to the novel and model images so that we are working in the coordinate system of the reference image. This transformation yields the following error equation

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel} \circ (A \circ \sum_{i=0}^{N} c_i S_i)(x, y) - I^{model} \circ (A \circ \sum_{i=0}^{N} c_i S_i)(x, y)]^2. \tag{4.2}$$

Using the definition for $\overline{S}$ from equation 2.6 and also using equation 3.4 we simplify the error equation to

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel} \circ \overline{S}(x, y) - \sum_{j=0}^{N} b_j T_j(x, y)]^2. \tag{4.3}$$

Minimizing this error yields the model image which best fits the novel image with respect to the L_2 norm. Other norms that weight outliers less may work even better.

We have chosen to use stochastic gradient descent to minimize this error function. We describe it in more detail in section 4.2.

Figure 4.1 gives a simple outline of the matching algorithm.

Stochastic gradient descent requires the derivative of the error with respect to each parameter. The necessary derivatives are as follows:

$$\frac{\partial E}{\partial c_i} = \sum_{x,y} \left[[I^{novel} \circ \overline{S}(x, y) - \sum_{j=0}^{N} b_j T_j(x, y)] \ \frac{\partial I^{novel} \circ \overline{S}(x, y)}{\partial c_i} \right]$$

$$\frac{\partial E}{\partial b_j} = -\sum_{x,y} \left[[I^{novel} \circ \overline{S}(x, y) - \sum_{j=0}^{N} b_j T_j(x, y)] \ T_j(x, y) \right]$$

$$\frac{\partial E}{\partial p_i} = \sum_{x,y} \left[[I^{novel} \circ \overline{S}(x, y) - \sum_{j=0}^{N} b_j T_j(x, y)] \ \frac{\partial I^{novel} \circ \overline{S}(x, y)}{\partial p_i} \right]$$

$$\frac{\partial I^{novel} \circ \overline{S}(x, y)}{\partial c_i} = \left. \frac{\partial I^{novel}(x, y)}{\partial (x, y)} \right|_{\overline{S}(x,y)} \left. \frac{\partial A(x, y)}{\partial (x, y)} \right|_{\sum_{i=0}^{N} c_i S_i(x,y)} \frac{\partial \sum_{i=0}^{N} c_i S_i(x, y)}{\partial c_i}$$

These equations are derived straightforwardly from the chain rule. Recall that $A(x, y)$ is a vector function with two components, i.e. $A(x, y) = (A^x, A^y)$. Similarly $S(x, y) = (S^x, S^y)$. The matrices of partial derivatives are

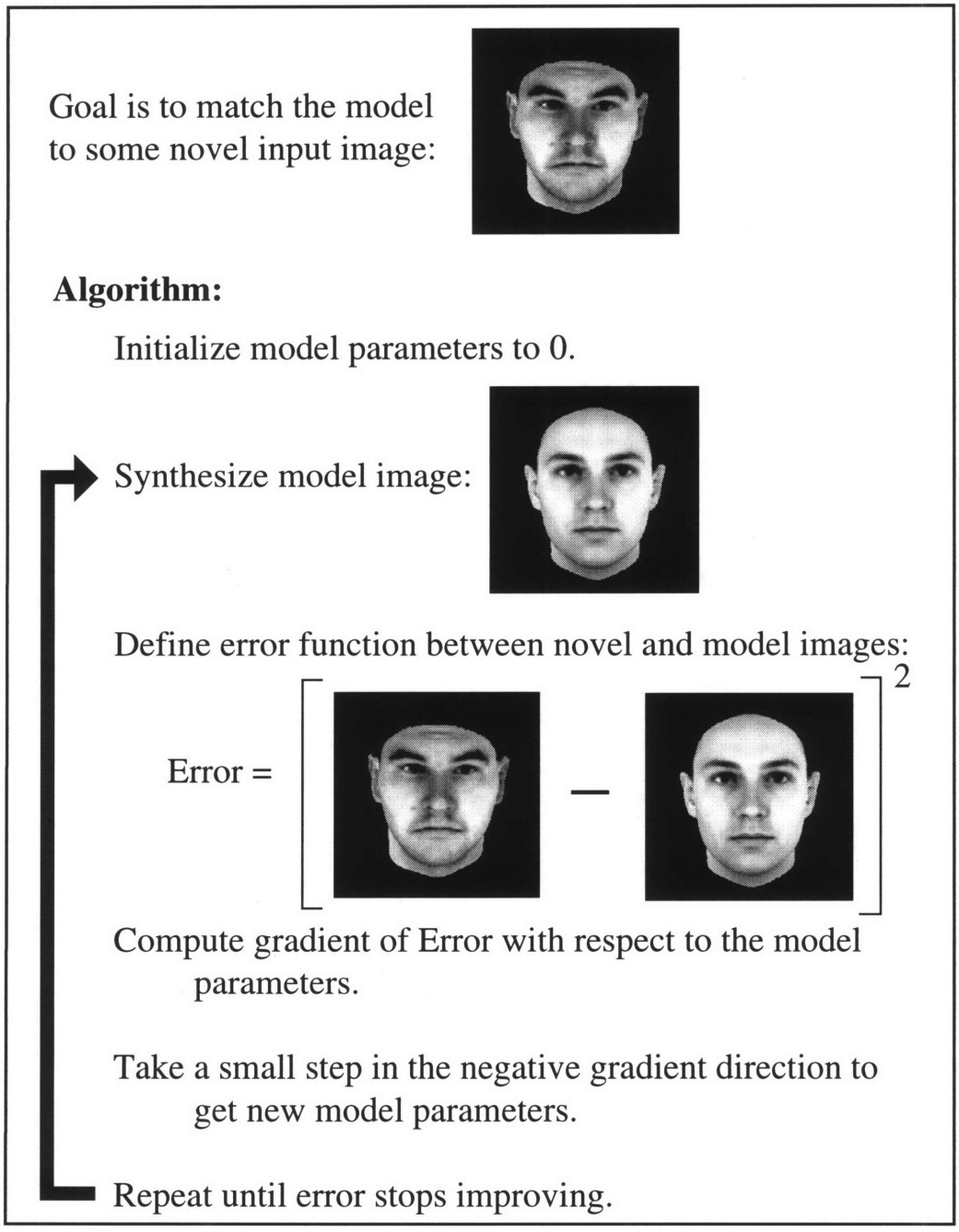


Figure 4.1: *An illustration of the matching algorithm.*

$$\left.\frac{\partial A(x,y)}{\partial(x,y)}\right|_{\sum_{i=0}^{N} c_i S_i(x,y)} = \left.\begin{bmatrix} \frac{\partial A^x}{\partial x} & \frac{\partial A^x}{\partial y} \\ \frac{\partial A^y}{\partial x} & \frac{\partial A^y}{\partial y} \end{bmatrix}\right|_{\sum_{i=0}^{N} c_i S_i(x,y)} = \begin{bmatrix} p_0 & p_1 \\ p_3 & p_4 \end{bmatrix}$$

$$\frac{\partial \sum_{i=0}^{N} c_i S_i(x,y)}{\partial c_i} = \begin{bmatrix} \frac{\partial \sum_{i=0}^{N} c_i S_i^x(x,y)}{\partial c_i} \\ \frac{\partial \sum_{i=0}^{N} c_i S_i^y(x,y)}{\partial c_i} \end{bmatrix} = \begin{bmatrix} S_i^x(x,y) - S_0^x(x,y) \\ S_i^y(x,y) - S_0^y(x,y) \end{bmatrix}$$

The $S_0(x,y)$ term in the previous derivative is due to $c_0 = 1 - \sum_{i=1}^{N} c_i$.

$$\frac{\partial I^{novel} \circ \overline{S}(x,y)}{\partial p_i} = \left.\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\right|_{\overline{S}(x,y)} \frac{\partial \overline{S}(x,y)}{\partial p_i}$$

$$\frac{\partial \overline{S}(x,y)}{\partial p_0} = \left[\sum_{i=0}^{N} c_i S_i^x(x,y),\ 0\right]^T$$

$$\frac{\partial \overline{S}(x,y)}{\partial p_1} = \left[\sum_{i=0}^{N} c_i S_i^y(x,y),\ 0\right]^T$$

$$\frac{\partial \overline{S}(x,y)}{\partial p_2} = [1,0]^T$$

$$\frac{\partial \overline{S}(x,y)}{\partial p_3} = \left[0,\ \sum_{i=0}^{N} c_i S_i^x(x,y)\right]^T$$

$$\frac{\partial \overline{S}(x,y)}{\partial p_4} = \left[0,\ \sum_{i=0}^{N} c_i S_i^y(x,y)\right]^T$$

$$\frac{\partial \overline{S}(x,y)}{\partial p_5} = [0,1]^T$$

To compute the spatial derivatives of the novel image, a finite difference approximation

may be used:

$$\left.\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\right|_{\overline{S}(x,y)} \approx \left[\tfrac{1}{2}(I^{novel}(\overline{S}(x,y)+(1,0)) - I^{novel}(\overline{S}(x,y))-(1,0)),\right.$$
$$\left.\tfrac{1}{2}(I^{novel}(\overline{S}(x,y)+(0,1)) - I^{novel}(\overline{S}(x,y)-(0,1)))\right]$$

Given these derivatives, the stochastic gradient descent algorithm can be used straightforwardly to find the optimal **c**, **p** and **b**.

4.2 Stochastic gradient descent

To minimize the error function, any standard minimization algorithm could be used. We have chosen stochastic gradient descent ([Robbins and Munroe, 1951], [Viola, 1995]) because it is fast and is less likely to get stuck in local minima. The difficulty with minimizing the error function in equation 4.3 is that the summation is over all pixels in the model image. For a 256×256 pixel image, this is 65536 gradients to calculate at each iteration. We would like to reduce this number to speed up the minimization. The idea of stochastic gradient descent is to randomly sample a small set of pixels from the image and only compute the gradient at those pixels. This gives an estimate for the true gradient. For each iteration of stochastic gradient descent, a new set of pixels is randomly selected using a uniform distribution. As Viola [Viola, 1995] discusses, this estimate for the gradient will be good enough to use for optimizing the parameters if the gradient estimate is unbiased, the parameter update rate asymptotically converges to zero and the error surface is smooth. In practice, we did not find it necessary to have the parameter update rate get smaller with time. In our experiments we typically choose only 40 points per iteration of the stochastic gradient descent. This results in a large speedup over minimization methods – such as conjugate gradient – which compute the full gradient over the whole image.

4.3 Pyramid representation

The matching algorithm requires the initial guess for the position of the model image to partially overlap the novel image. In other words, if one were to place the model image (using the current affine parameters for the amount of translation) over the novel image (which may be larger than the model image) then there should be significant overlap between the object in the model image and the object in the novel image. The amount of overlap needed is investigated in chapter 6. In order to make the matching more robust in this respect, we have implemented a coarse-to-fine pyramid approach ([Burt and Adelson, 1983]; [Burt, 1984]).

In an initialization stage, the matching algorithm creates image pyramids for the shape vectors, texture vectors and novel image with each level of the pyramid containing an image that is one fourth the size of the one below. To create an image of quarter size every other pixel is replaced by a weighted averaging of neighboring pixels (a guassian weighting is used). The flow fields (shape vectors) are also subsampled in this way by treating the x and y components separately, and in addition all the displacements in the flow field are divided by two at each level.

The minimization algorithm is first used to fit the model parameters starting at the coarsest level. The resulting parameter values are then used as the starting point at the next level. The translational affine parameters (p_2 and p_5) are multiplied by 2 as they are passed down the pyramid to account for the increased size of the images.

4.4 Pseudo code for the matching algorithm

The following pseudo code describes the matching algorithm. The model is learned once from the set of prototypes, i.e. the learning phase is only done once. The matching then takes place for each novel image to be analyzed.

LEARNING PHASE

Given the prototype images, I_j for $j = 0, \ldots, N$

1. Compute pixelwise correspondences between prototype images and the reference image I_0 using an optical flow algorithm or a semi-automatic technique: yields S_j
2. Compute texture vectors, T_j, by backward warping (Appendix B)

MATCHING PHASE

Given a novel image and a model defined by shape vectors $\{S_j\}$ and texture vectors, $\{T_j\}$

1. Create image pyramids for I^{novel}, $\{T_j\}$ and $\{S_j\}$
2. Initialize parameters $\mathbf{c}$ and $\mathbf{b}$ (typically set to zero) and $\mathbf{p} = [1, 0, 0, 0, 1, 0]$ (the identity transformation)

For each level in the pyramid beginning with the coarsest

3. Estimate the parameters $\mathbf{c}$, $\mathbf{p}$ and $\mathbf{b}$ by iterating the basic step of stochastic gradient descent either a fixed number of times or until the error stops decreasing significantly
4. Multiply the constant affine parameters p_2 and p_5 by 2
5. Go to next level in the pyramid

6. Output the parameters

4.5 Compressing the model using principal components

The prototypical shape vectors are not required to be orthogonal and in fact may not even be linearly independent. The same is true of the texture vectors. This suggests that it might be worthwhile to do a principal components analysis on the shape and texture spaces in order to compress the model.

To do this, the eigenvectors for both the shape space and texture space are computed independently. The eigenvectors of the shape space are computed by first putting each shape vector, which is a flow field typically represented as two matrices, into one big vector. One

way to do this is to simply place the elements in each column of the x displacement part of the flow field into the top half of a vector, and then place the elements in each column of the y displacement part of the flow field in the bottom half of the vector. This is done for each flow field to form N vectors. If the prototype images are say 256×256 pixels then the vectors just described will be of length 131,072. An analogous procedure is done for the texture vectors to yield N vectors of length $65,536$ in this example. A standard eigenvector algorithm can then be run separately on these vectors which represent the shape and texture spaces.

The shape space and texture space representations can then be effectively "compressed" by using only the first few eigenvectors (with largest eigenvalues) in the linear combinations of the model (both for shape and texture). So after computing the eigenvectors for the shape space, for example, they are used in place of the shape vectors in the matching algorithm. Likewise for the texture space. The only change necessary to the matching algorithm is to add in the mean shape or mean texture anytime a linear combination of shape or texture is computed in the model. This is because the mean shape and texture are subtracted from each shape and texture vector (respectively) when computing the eigenvectors.

We emphasize that the technique performs well without using eigenvectors, which however can provide additional computational efficiency.

4.6 Probabilistic model

For many applications it is necessary to have a measure of how good the fit is to a novel image after matching. One measure of this is simply the L_2 error between the novel image and the best fitting model image. We can also look at the parameters of the model to determine how likely they are. The idea is that even if the L_2 error is low, the parameters may be so unlikely (meaning, for example, that the reference image had to be very greatly distorted) that we should have low confidence that the novel image is actually a member of the class of objects we are modeling. In order to determine the likelihood of the parameters,

we need a probabilistic model of them. We develop such a model below.

The shape and texture spaces for a particular object class can be written in terms of their principal components (as described in the previous section).

Let $\mathbf{s}_{mean}$ be the mean shape vector and let $\mathbf{s}_1, \ldots, \mathbf{s}_N$ be the eigenvectors of the shape space. Let $\lambda_1, \ldots, \lambda_N$ be the eigenvalues of the shape space. Similarly, let $\mathbf{t}_{mean}$ be the mean texture and let $\mathbf{t}_1, \ldots, \mathbf{t}_N$ and $\alpha_1, \ldots, \alpha_N$ be the eigenvectors and eigenvalues (respectively) of the texture space.

Define

$$S^{model} = \sum_{i=1}^{N} c_i \mathbf{s}_i$$

and

$$T^{model} = \sum_{i=1}^{N} b_i \mathbf{t}_i$$

Now, given a set of parameters $\mathbf{c}$ and $\mathbf{b}$ found by the matching algorithm (using the eigenvector representation for the shapes and textures), we want to know their likelihood. We will assume that the model shape vector, S^{model}, for some input I^{novel} falls at a particular point along the $\mathbf{s}_i$ eigenvector with a probability that follows a Gaussian distribution. More precisely we assume there is a 1-D Gaussian distribution along each eigenvector direction with mean $\mathbf{s}_{mean}$ and variance λ_i. Hence,

$$Pr(S^{model}) = \frac{1}{\lambda_i \sqrt{2\pi}} e^{-[(S^{model} - \mathbf{s}_{mean}) \cdot \mathbf{s}_i]^2 / 2\lambda_i^2}. \tag{4.4}$$

Similarly, for the texture space we assume there is also a 1-D Gaussian distribution along each texture eigenvector with mean $\mathbf{t}_{mean}$ and variance α_i. So,

$$Pr(T^{model}) = \frac{1}{\alpha_i \sqrt{2\pi}} e^{-[(T^{model} - \mathbf{t}_{mean}) \cdot \mathbf{t}_i]^2 / 2\alpha_i^2}. \tag{4.5}$$

We will assume the distributions along different eigenvectors are independent so the total probability of a model shape is the product of the probabilities in each eigenvector direction:

$$Pr(S^{model}) = \frac{1}{\lambda_1 \lambda_2 \ldots \lambda_N \sqrt{2\pi}^N} e^{\frac{-[(S^{model} - \mathbf{s}_{mean}) \cdot \mathbf{s}_1]^2}{2\lambda_1^2} - \frac{[(S^{model} - \mathbf{s}_{mean}) \cdot \mathbf{s}_2]^2}{2\lambda_2^2} - \ldots - \frac{[(S^{model} - \mathbf{s}_{mean}) \cdot \mathbf{s}_N]^2}{2\lambda_N^2}}$$

$$= \frac{1}{\lambda_1 \lambda_2 \ldots \lambda_N \sqrt{2\pi}^N} e^{-\frac{c_1^2}{2\lambda_1^2} - \frac{c_2^2}{2\lambda_2^2} - \ldots - \frac{c_N^2}{2\lambda_N^2}} \tag{4.6}$$

since

$$\begin{aligned} (S^{model} - \mathbf{s}_{mean}) \cdot \mathbf{s}_i &= (c_1\mathbf{s}_1 + c_2\mathbf{s}_2 + \ldots + c_N\mathbf{s}_N) \cdot \mathbf{s}_i \\ &= c_i \end{aligned}$$

since $\mathbf{s}_i \cdot \mathbf{s}_j = 0$ if $i \neq j$ and 1 if $i = j$.

Similarly for the model texture,

$$Pr(T^{model}) = \frac{1}{\alpha_1 \alpha_2 \ldots \alpha_N \sqrt{2\pi}^N} e^{-\frac{b_1^2}{2\alpha_1^2} - \frac{b_2^2}{2\alpha_2^2} - \ldots - \frac{b_N^2}{2\alpha_N^2}} \tag{4.7}$$

Equations 4.6 and 4.7 can be used as a measure of confidence for any model shape and texture found by the matching algorithm. Such a measure should be used along with the L_2 error (or some other measure of distance between images) to derive an overall confidence measure.

Chapter 5

Examples of Multidimensional Morphable Models

The multidimensional morphable model was tested on three different sets of prototypes. The first two are face databases. We choose faces because they form a very nicely defined object class, and modeling faces yields a number of interesting and useful applications including recognition, tracking and expression analysis. The last example is a set of side views of cars.

5.1 Face Model #1

The prototype faces for this model were collected by Thomas Vetter and Nikolaus Troje of the Max Planck Institute in Tubingen, Germany. The images were originally rendered for psychophysical experiments [Troje and Bülthoff, 1995] under ambient illumination conditions from a database of three-dimensional human head models recorded with a laser scanner (*Cyberware*). All faces were without makeup, accessories, and facial hair. Additionally, the head hair was removed digitally (but with manual editing), via a vertical cut behind the ears. The resolution of the 8 bit grey-level images is 256-by-256 pixels. Figures 5.1, 5.2, and 5.3 show the set of face prototypes.

The face images were aligned roughly by automatically adjusting them to their two-dimensional centroid. The centroid was computed by evaluating separately the average of

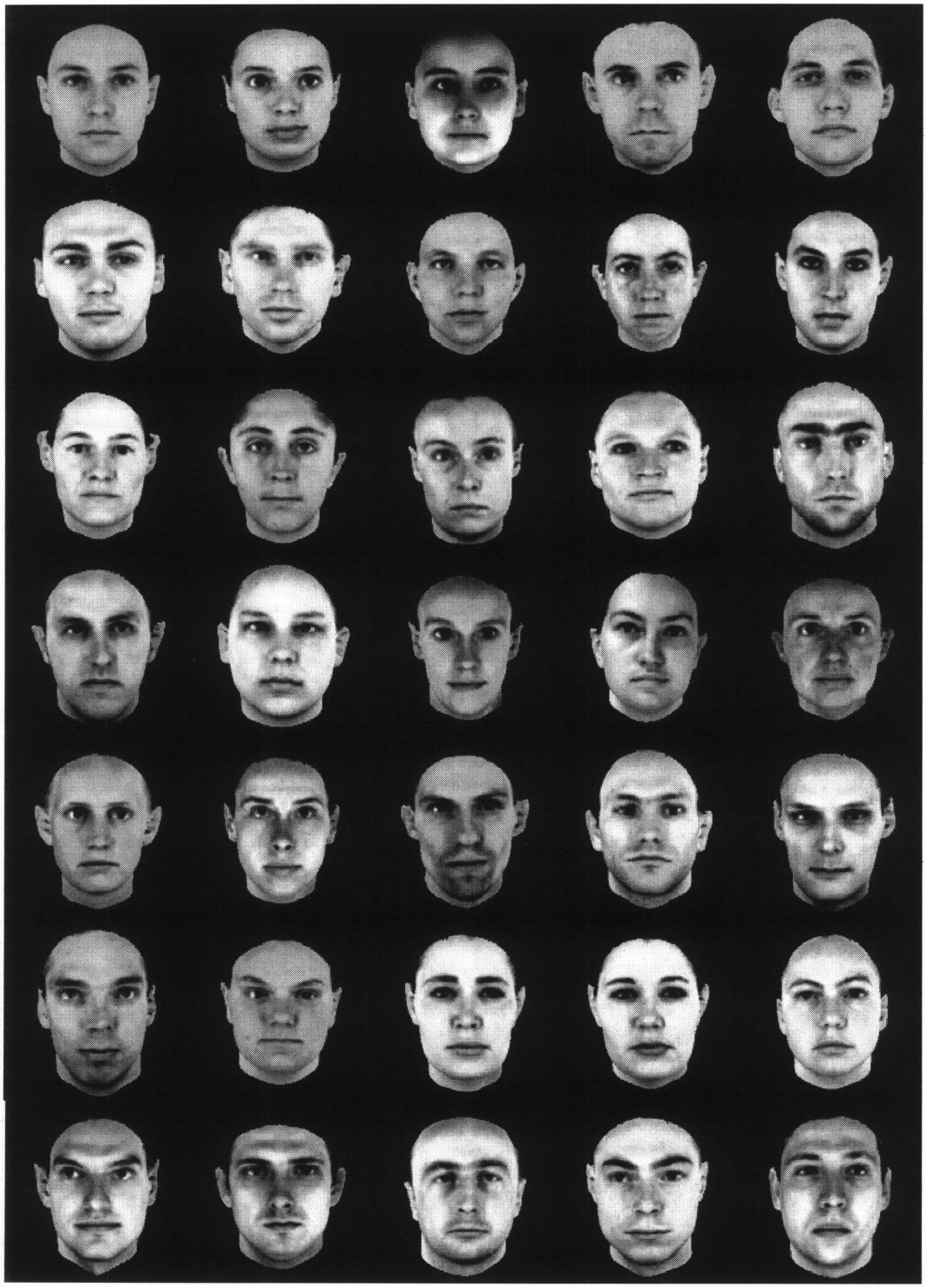


Figure 5.1: *Thirty-five of the 100 prototypes in face model #1. The prototype in the top left corner is the reference image.*

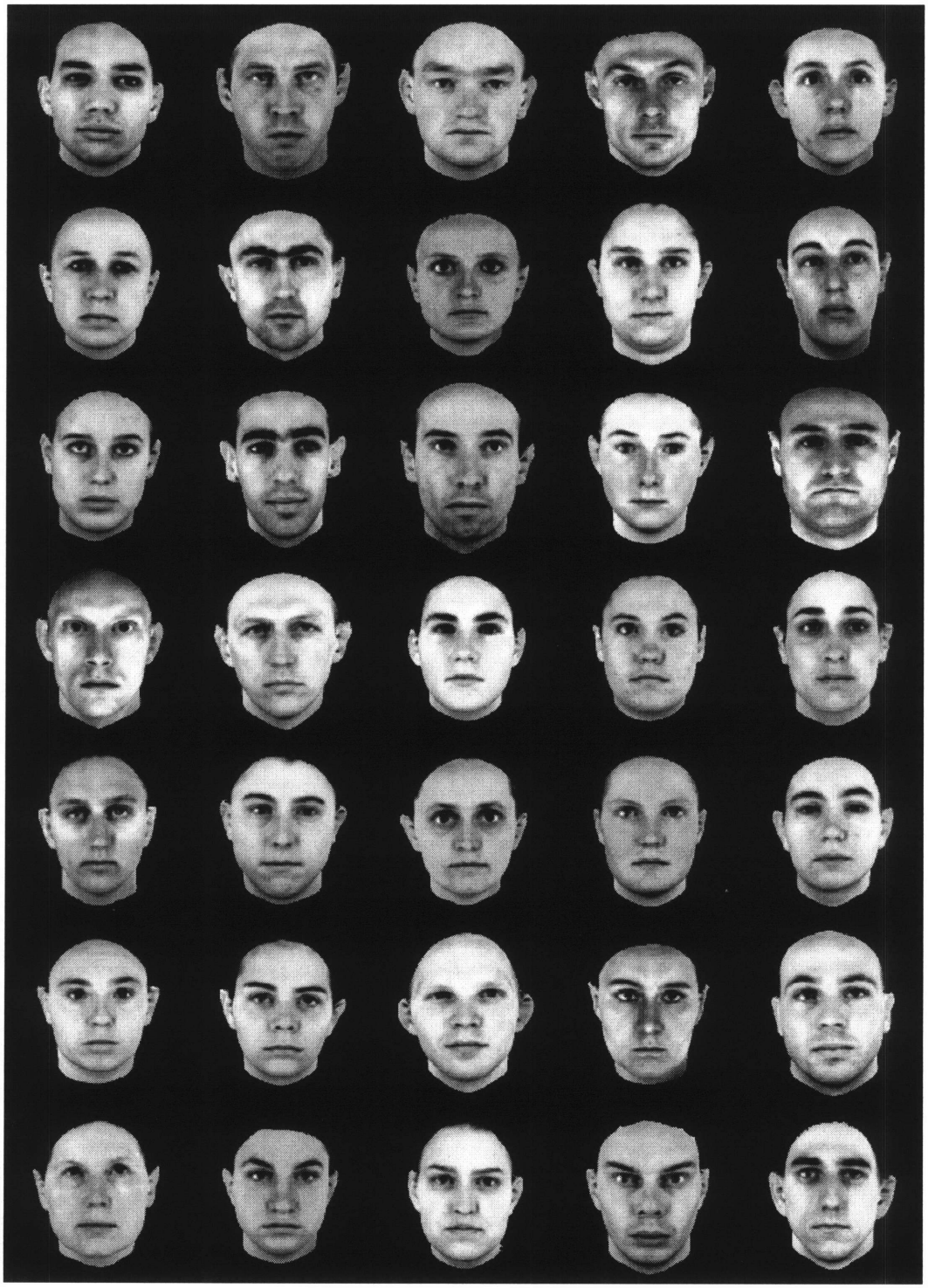


Figure 5.2: *Thirty-five of the 100 prototypes in face model #1.*

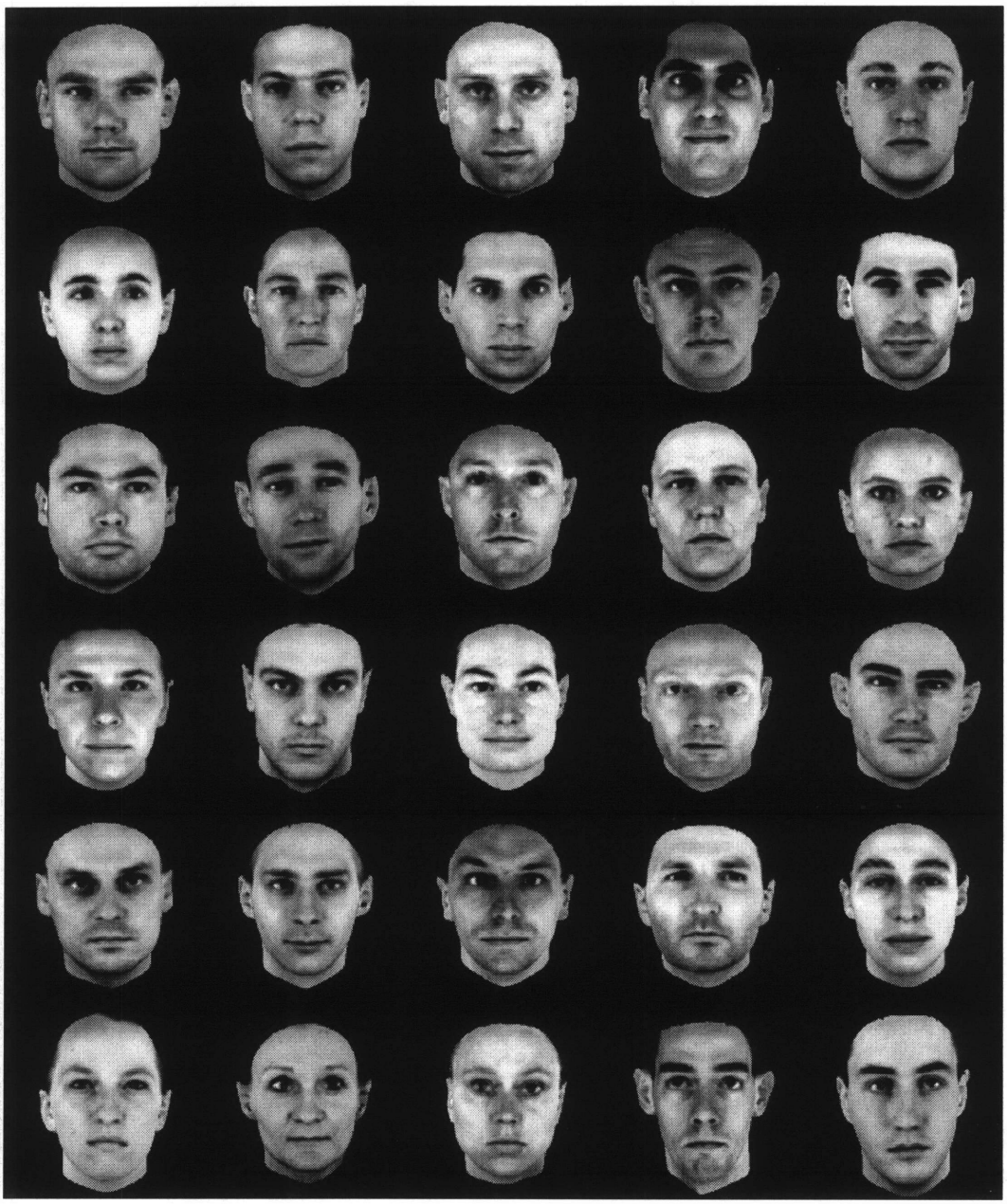


Figure 5.3: *Thirty of the 100 prototypes in face model #1.*

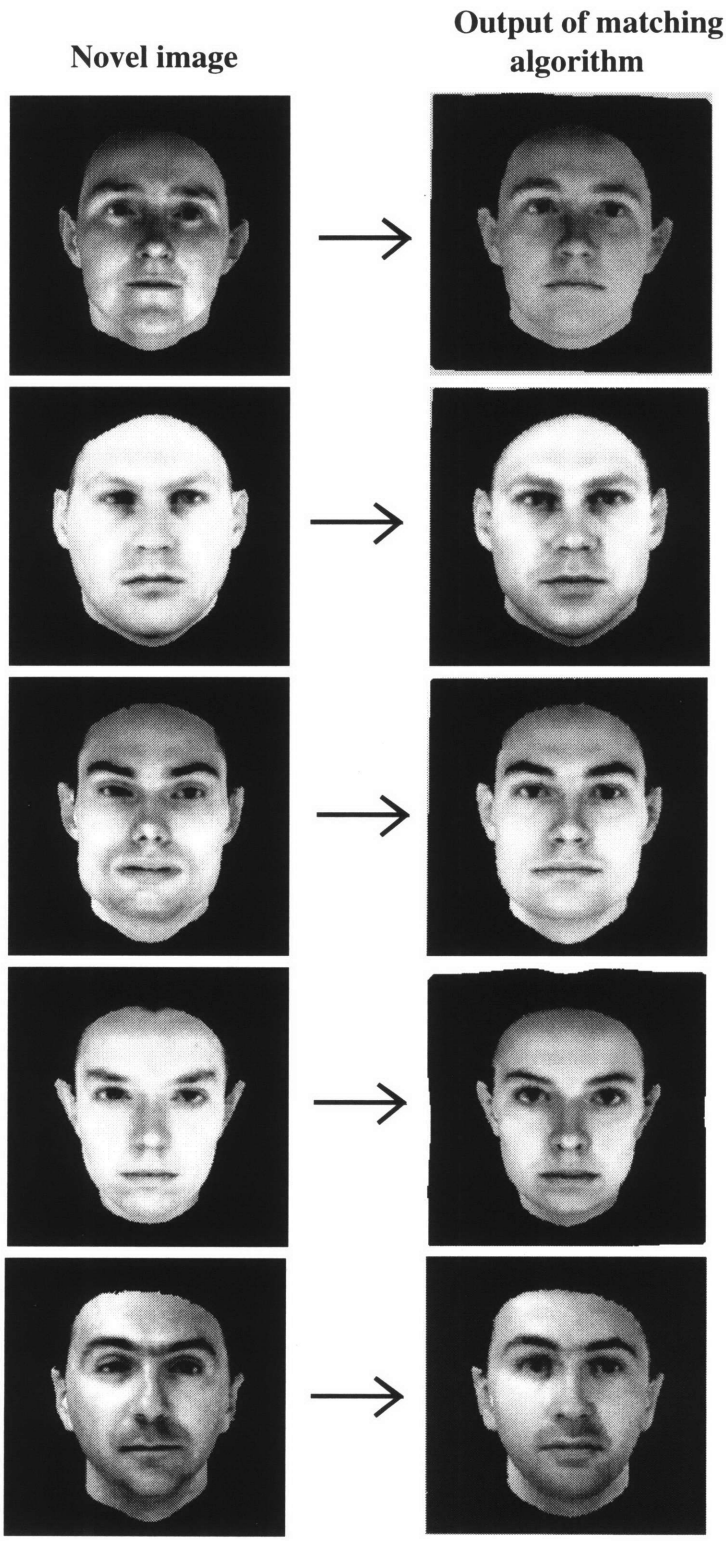


Figure 5.4: *Five examples of matching face model #1 to a novel face image.*

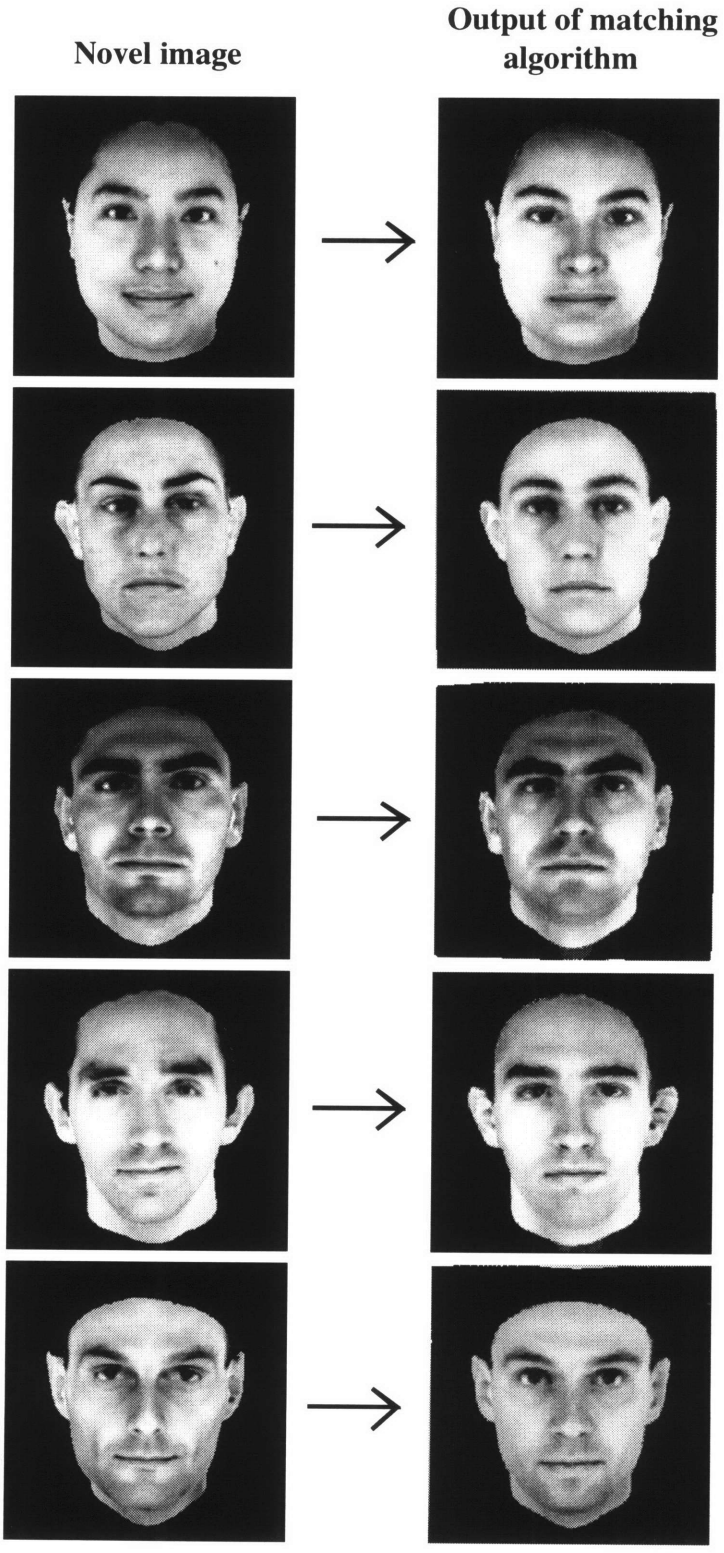


Figure 5.5: *Five more examples of matching face model #1 to a novel face image.*

all x, y coordinates of the image pixels within the face independent of intensity value.

There are 130 faces in the Vetter-Troje database. One hundred faces were used as prototypes and 30 were used to test the matching algorithm.

The pixelwise correspondences between a reference image (which in this case is the average face) and the 99 other prototype face images were computed automatically using the bootstrapping algorithm described in chapter 8.

A principal components analysis was used to compress the shape and texture spaces (independently) as discussed in section 4.5. It was found that 30 shape eigenvectors and 20 texture eigenvectors were enough to yield good reconstructions of novel faces using the matching algorithm. The matching algorithm took about 3.5 minutes to run using 30 shape and 20 texture eigenvectors, using an SGI Indy machine. There was no effort spent on trying to optimize the code or find optimal stopping conditions on the matching algorithm. The code could no doubt be speed up significantly. We used 3 pyramid levels and 8000 stochastic gradient iterations per level for the matching algorithm. Also, 40 random points were used at each iteration of the stochastic gradient algorithm.

Figures 5.4 and 5.5 shows some matches to novel faces. Overall, the novel image is matched very well. It is interesting to note that as we expected individual details are not reconstructed well. For example the first novel face in figure 5.5 contains a mole on her left cheek. This mole is not recovered in the model image. However the overall shape and texture is recovered well.

5.2 Face model #2

The prototype faces for face model #2 were collected by David Beymer at the MIT AI Lab [Beymer, 1996]. These faces are more natural in that their head hair has not been removed and some have facial hair. There are 62 faces in this database.

These images are 8 bit grey level with dimensions 184 by 226 pixels. In a preprocessing

stage they were aligned according to the center of their eyes. This was done automatically using a correlation-based eye finder.

The bootstrapping algorithm was used to find the correspondences from the average face to each of the other prototypes. The average face (topmost image in figure 5.6) was computed by the procedure described in chapter 8.

The 63 prototypes are shown in figure 5.6. Since we only had 63 images in this database (counting the average face), we used 62 to form a model and then used the remaining one to test the matching algorithm. We did this for each of the prototypes (except the reference). We used 3 pyramid levels and 8000 iterations of stochastic gradient per level. The matching algorithm took about 9 minutes to run using 61 prototypes. Again this is on an SGI Indy running unoptimized code.

Figures 5.7 shows some typical matches to novel faces. The best fitting model images shown in the figure have been cropped to remove the head hair. This is because the model correspondences in the head hair region are not accurate. This is not surprising since the correspondences between a person with short hair and a person with long hair are not well defined in the hair region. In this case we are only concerned about the quality of the matches in the facial region. As with face model #1, the matches are quite good.

5.3 Car model

To build a model of side views of cars we took pictures of 48 Matchbox cars. The car images are 256 pixels wide by 96 pixels high. The images of the prototype cars were normalized by making the distance between the center of the front and back wheels in each car equal. The centers of the wheels were identified manually. Again, we used the bootstrapping algorithm to find the correspondences among the prototypes. The bootstrapping algorithm was only able to find good correspondences to 39 of the 48 prototypes. In general, the correspondence problem for the cars is more difficult than for faces because some cars have

Figure 5.6: *The 63 prototype images of face model #2. The prototype in the top left corner is the reference image which is the average face in this case.*

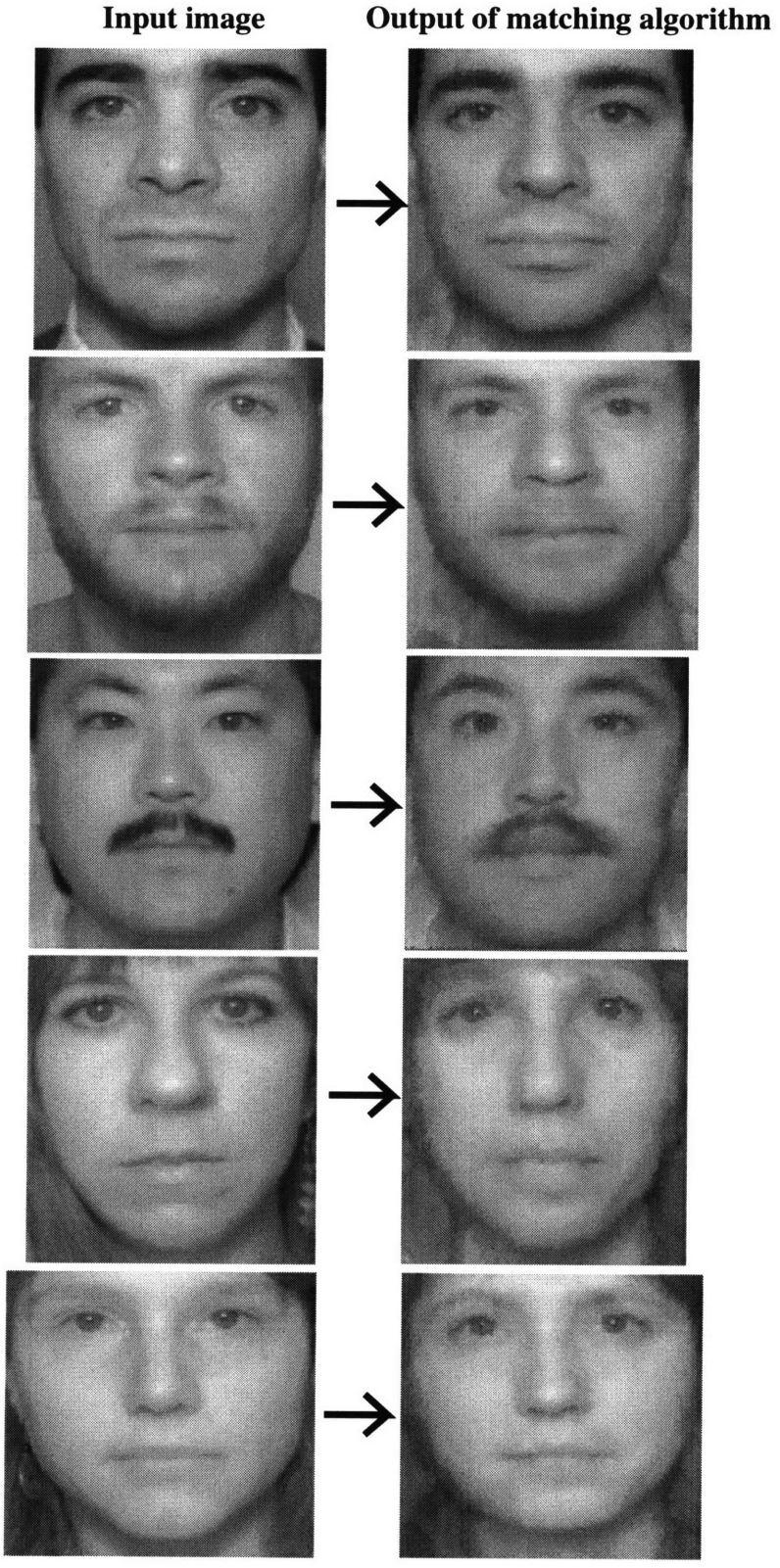


Figure 5.7: *Five examples of matching face model #2 to a novel face image.*

Figure 5.8: *The 40 prototype images of cars. The prototype in the top left corner is the reference image (and also the average image).*

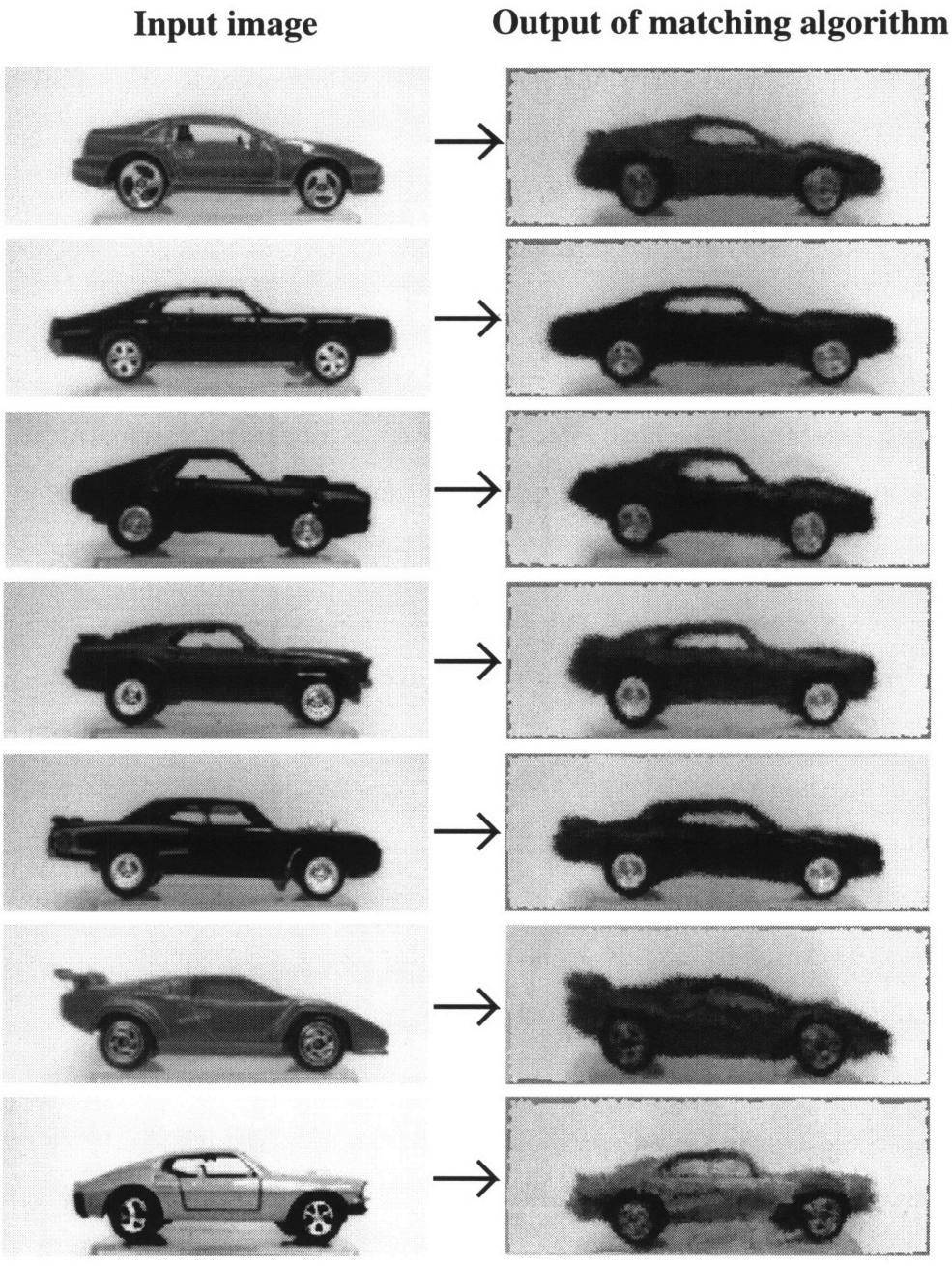


Figure 5.9: *Seven examples of matching the car model to a novel car image.*

features (such as spoilers) which are not found on other cars. Also, there is more variation among the shapes of different cars.

The goodness of a correspondence field was judged by manual inspection. Figure 5.8 shows the 39 car prototypes for which good correspondences were found plus the average image used as the reference. These prototypes were used to define a morphable model for cars. The nine cars for which good correspondences were not found are mostly cars with very different shapes from the average car and for which the correct correspondences are ambiguous.

The matching algorithm was again run with three pyramid levels. Forty points were chosen for each iteration of stochastic gradient descent. Stochastic gradient descent was run for 8000 iterations per pyramid level. The running time on an SGI Indy was about 5 minutes.

To test the model on novel cars, we used the same procedure as with face model #2, namely, one prototype was left out of the model and used as a novel image. Figure 5.9 shows the results of matching the car model to seven novel car images. The overall shape of the cars is matched well (except for the last example). However, the texture is not matched quite as well as in the case of faces. The main reason for this is that the texture of cars is much more random than with faces. There are not really any constraints on what markings can appear on a car and therefore the texture is not modeled well using linear combinations of prototypes. However, despite this problem, the matching algorithm does a reasonable job of matching the shape of the novel cars.

Figure 5.10 shows the results of matching the model to four of the car images for which bootstrapping could not find good correspondences. As expected, the matches are not quite as good as with the previous examples. However, the matches do capture the general shape of the cars adequately. To add such atypically shaped cars to the model it seems necessary to manually specify the correspondences.

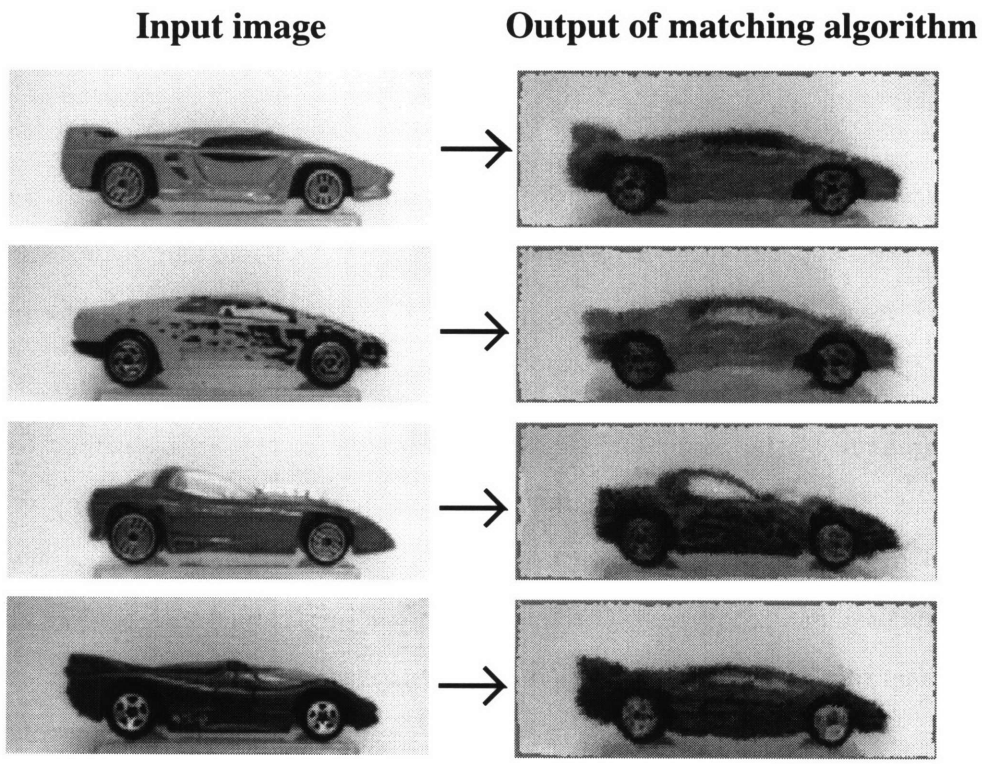


Figure 5.10: *Four examples of matching the car model to a "difficult" novel car image.*

Chapter 6

Robustness Experiments

The results shown so far have all used novel input images which have already been roughly aligned to the prototype images in terms of translation, rotation and scale. An important question to ask is how well the matching algorithm performs when the input image is not already aligned.

The performance of the matching algorithm depends on the error surface that is being optimized. Since the error surface is not convex and is different for each input image and for each set of prototypes, this question cannot be answered independently of a particular input image and set of prototypes. Our method of testing robustness is thus an empirical one in which a number of different input images with various transformations are tested to determine how well they are matched. For these experiments we used face model #1 for testing. Although the results only apply directly to these face images, a general idea of the performance of the matching algorithm can be inferred from these results.

The succeeding experiments use the following strategy. The matching algorithm described in chapter 4 is run on an input image which has been transformed using some known transformation. The L_2 error of the resulting match is then used to determine if the algorithm found a "good match." Defining a good match is not obvious. The problem is that the L_2 error will sometimes be higher for a translated, rotated or scaled input image than the original aligned image even though the synthesized best matching model image looks to

a human observer like a good match. How much worse can the L_2 error become before the best match is no longer good enough? Since the real measure we are using for what is "good enough" is how it looks to a human observer, we have choosen the threshold for the L_2 error based on the perception of a human observer. So, a good match is defined as one that has L_2 error less than or equal to 1.2 times the error for the normalized input image. It was found by visual inspection that an image with 1.2 times the error of the normalized input image still looks very similar to the normalized input image. One with more error began to look significantly different.

6.1 Robustness to translation

Figure 6.1: *Example of matching a novel face that has been translated 6 pixels in the x direction and -10 pixels in the y direction. The matching algorithm automatically translated the model image by optimizing the affine parameters for translation (as well as the other parameters of the model).*

The first set of experiments tested the robustness of the matching algorithm to changes in translation. An example of a translated face image and the resulting best match is shown in figure 6.1. Note that the affine parameters were initialized to the identity transformation, so that to match the translated image, the matching algorithm had to change the translational components of the affine parameters (p_2 and p_5) in order to align the model with the novel image. Figure 6.2 summarizes the matching performance for six novel images over a range of different translations from +20 pixels to -20 pixels in both the x and y directions. (Recall

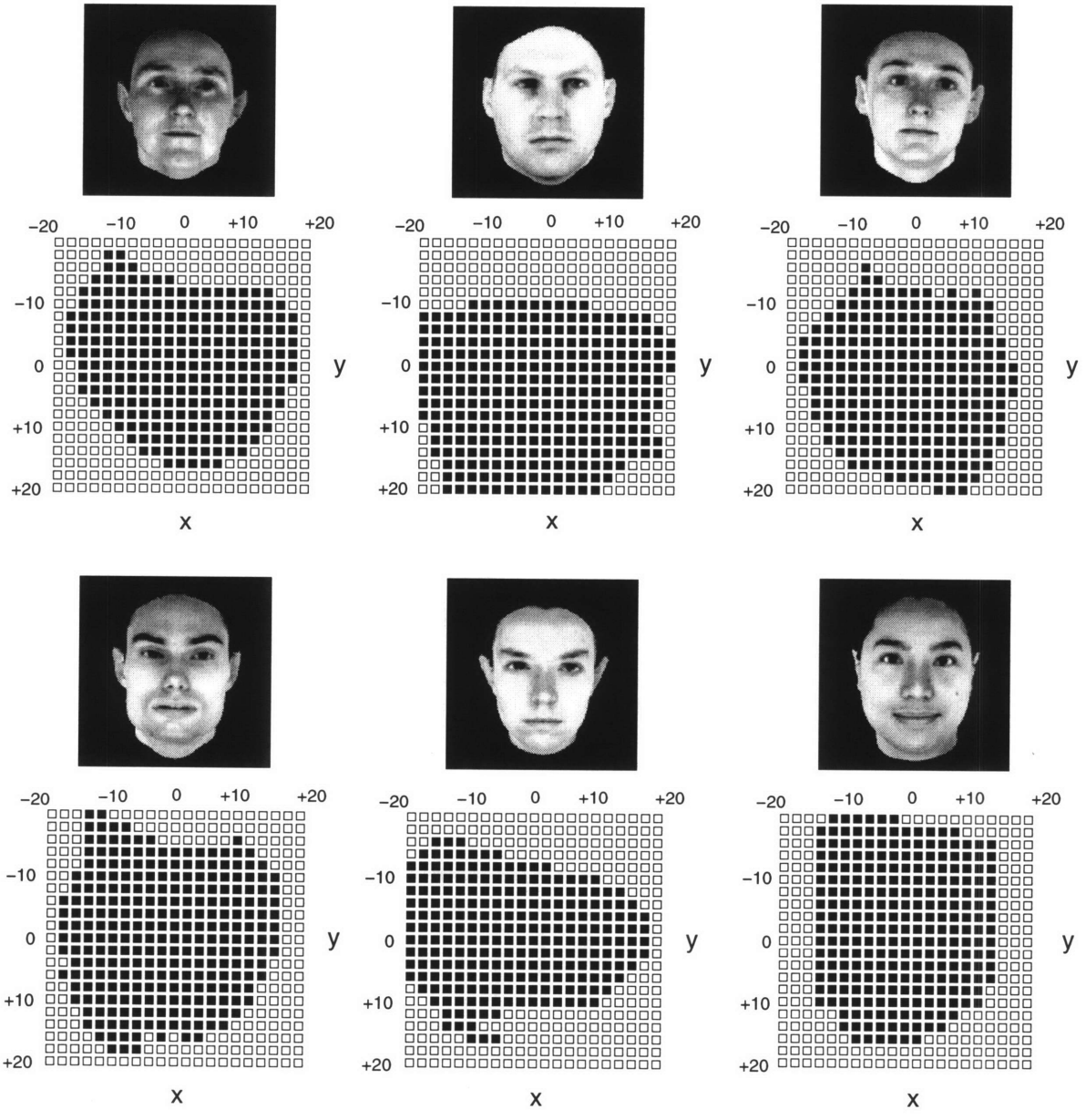


Figure 6.2: *Summary of robustness tests for translation. Each filled-in square indicates a translation for which the input image was matched well. The face above each grid is the input image before translation.*

that the images are 256×256 pixels.)

The results are different for each novel input image, but in general the matching algorithm can reliably match images that are off center by about ± 15 pixels in both the x and y directions.

6.2 Robustness to rotation

Input novel image Output after matching

Figure 6.3: *Example of matching a novel face that has been rotated 27 degrees. The matching algorithm automatically rotated the model by optimizing the affine parameters for rotation.*

Next we tested the robustness of the matching algorithm to changes in image plane rotation. An example of a rotated face and the resulting best match is shown in figure 6.3. Again, the matching algorithm must fit the rotated input image by changing the affine parameters (in addition to the other parameters of the morphable model). Figure 6.4 shows the results of matching six different input images with varying amounts of rotation. The circular graphs show the error for the different amounts of rotation tested. Each line segment radiating from the center of the circle is proportional in length to the error of the resulting match for that rotation angle. The circle indicates the cutoff for good matches which is at a distance of 1.2 times the error of the match for zero degrees rotation. In other words, any rotation angle with a line segment ending inside the circle is considered a good match.

The results show that rotations from -30 degrees to +30 degrees are generally matched well.

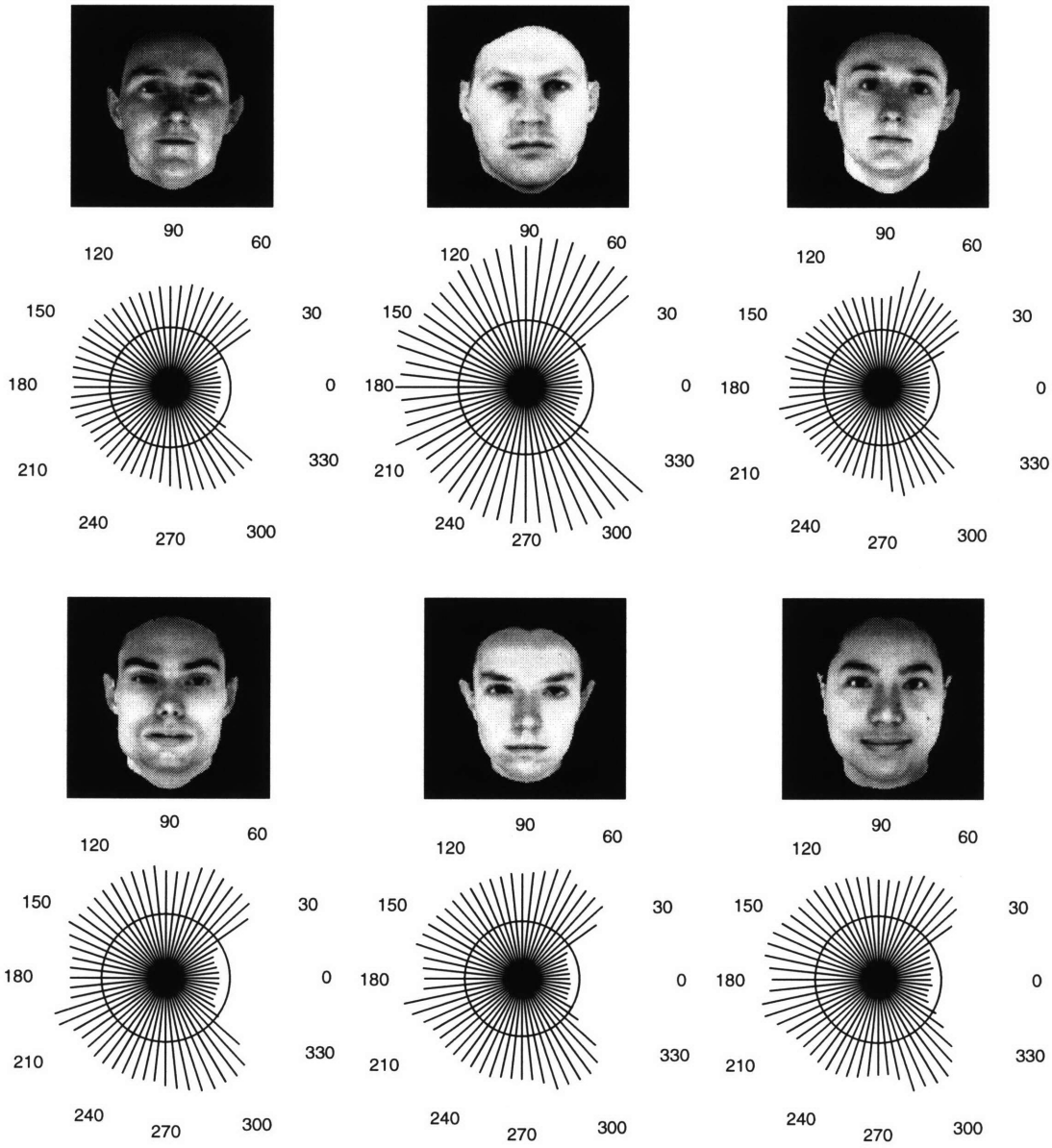
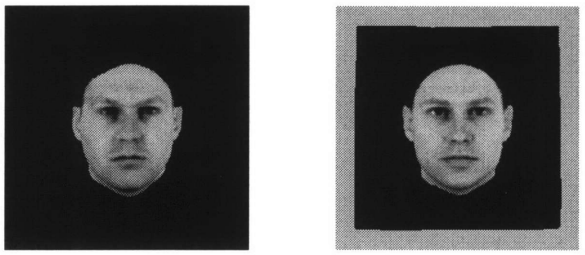


Figure 6.4: *Summary of robustness tests for rotation. Each line segment radiating from the center is proportional in length to the error of the match at that angle. Angles for which the line segment is inside the circle are considered good matches. The face above each grid is the input image before rotation.*

6.3 Robustness to scale



Input novel image Output after matching

Figure 6.5: *Example of matching a novel face that has been scaled to 0.6 times original size. The matching algorithm automatically scaled the model by optimizing the parameters for scale.*

The matching algorithm was next tested for robustness to scaling. An example of a scaled face and the resulting best match is shown in figure 6.5. It is important to realize that the information that the input face is a different size from the prototypes is not provided to the matching algorithm. The algorithm automatically discovers this as it fits the parameters of the model, including the affine parameters which allow for changes in scale. Figure 6.6 shows the results of matching six different input images over a range of different scales. Each filled-in square below the face images indicates a scale for which the matching algorithm successfully matched the input image.

The results show that the matching algorithm can reliably match input faces with scales between about .55 and 1.6 times the original size.

6.4 Robustness to occlusion

The final set of experiments test the ability of the matching algorithm to match partially occluded input images. To match occluded images we had to modify the matching algorithm slightly.

The matching algorithm picks points at random at each iteration of stochastic gradient

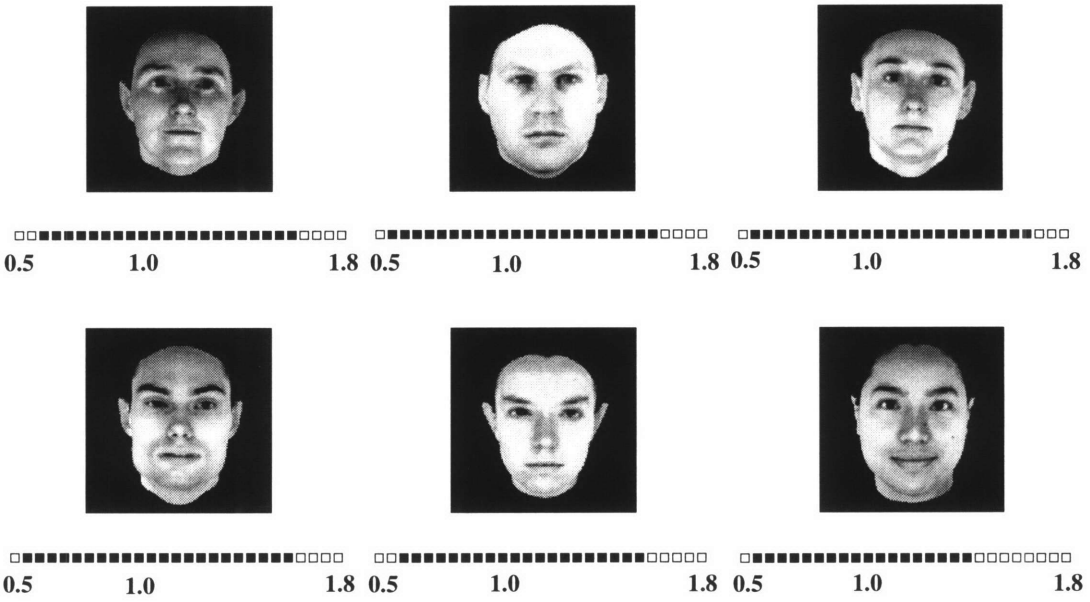


Figure 6.6: *Summary of robustness tests for scale. Each filled-in square indicates a good match at that scale.*

descent in order to estimate the gradient of the error. As long as most of these points come from non-occluded regions then the estimate for the gradient should be good enough to eventually lead to a minima. The only problem with this reasoning is that points which come from occluded regions have the highest error and thus contribute most to the gradient. Therefore, we would like to ignore points chosen from occluded regions, but unfortunately we do not know which regions these are. So, in order to prevent large errors in occluded regions from corrupting the gradient estimate too much, we modify the matching algorithm to simply throw away points with high errors. We found empirically that only the first six or so points (out of 40) with highest error were likely to come from occluded regions. So the modified matching algorithm throws out the six sampled points with highest error at each iteration of stochastic gradient.

Using this modified matching algorithm, the robustness to occlusion was tested as follows. A solid rectangle was randomly placed over the input image to cover some percentage of the face. The percentages tested were 5% through 40% at steps of 5%. The output of the

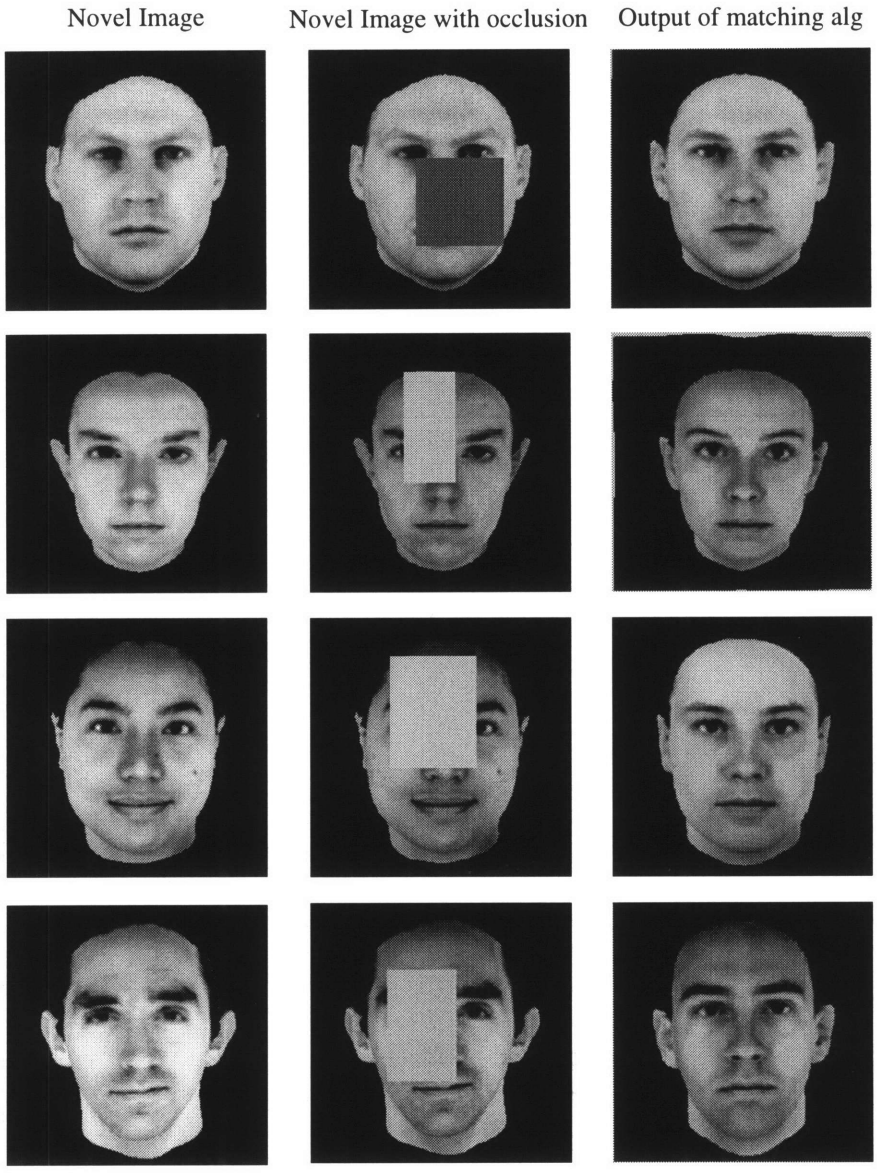


Figure 6.7: *Examples of matching partially occluded face images. The input images to the matching algorithm are in the middle column. These input images are occluded versions of the face images in the left column. The output of the matching algorithm is shown in the right column.*

matching algorithm was compared against the original unoccluded image to determine the error of the match. Figure 6.7 shows some occluded input images and the resulting best match found by the matching algorithm.

Figure 6.8 shows a graph which plots L_2 error versus percent of occlusion in the input image. The L_2 error plotted for each occlusion percentage is an average of the L_2 error over 10 different runs of the matching algorithm. On each run a new rectangle (with the appropriate size for that occlusion percentage) was randomly placed over the input image. The graph only shows the results for experiments run on one novel image, but other images produced very similar results. The graph shows that the quality of the match degrades gracefully with increasing occlusion percentage.

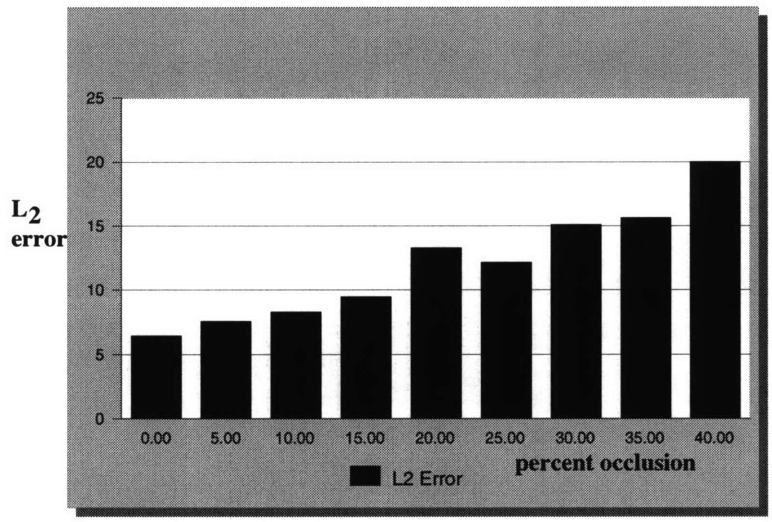


Figure 6.8: *Graph showing the gradual degradation of matching with increasing occlusion. Each error bar (except the 0% one) is averaged over 10 runs of the matching algorithm in which random occluding rectangles with the appropriate size were chosen.*

Overall, the results show that the matching algorithm always reconstructs a reasonable face despite large areas of occlusion. Facial features that are occluded are still reconstructed

in a reasonable fashion.

Chapter 7

Modeling Illumination Changes

Novel input images cannot be guaranteed to be taken under the same lighting conditions as the prototype images which define a morphable model. The models we have shown so far are not able to match images with very different lighting conditions than the prototypes. In order to make the model robust to changes in illumination we will add new prototypes which were created under different lighting conditions. This is one of the strengths of an example-based approach to computer vision - if the model does not capture some important area of image space, one can simply add examples from that area.

7.1 Adding new prototypes

In the morphable model framework, it is most efficient to add examples of just the reference image under different illuminations. We could, alternatively, add examples of all the existing prototypes under different illuminations, but this would increase the number of prototypes greatly. Instead we can add just examples of the reference image under various illuminations. Since we know the pixelwise correspondences between the reference image and each of the other prototypes, we can map the changes to the texture of the reference image due to a new lighting condition onto each prototype. Thus, we can synthesize virtual views [Beymer, 1996] of each prototype under the same lighting conditions for which we have examples for the

reference image. In addition, we can synthesize the prototypes under novel illuminations by using linear combinations of the illumination prototypes that we are given. This is discussed further below. To test this approach, we added examples taken under different illuminations to face model #1. Figure 7.1 shows the prototypes we added. These prototypes are the reference image rendered under various lighting conditions. The shape vectors for these prototypes are all the null flow field. Only the texture vectors contain new information. So the new model consists of the same prototypes as shown in figures 5.1, 5.2 and 5.3 plus the new illumination prototypes shown in figure 7.1.

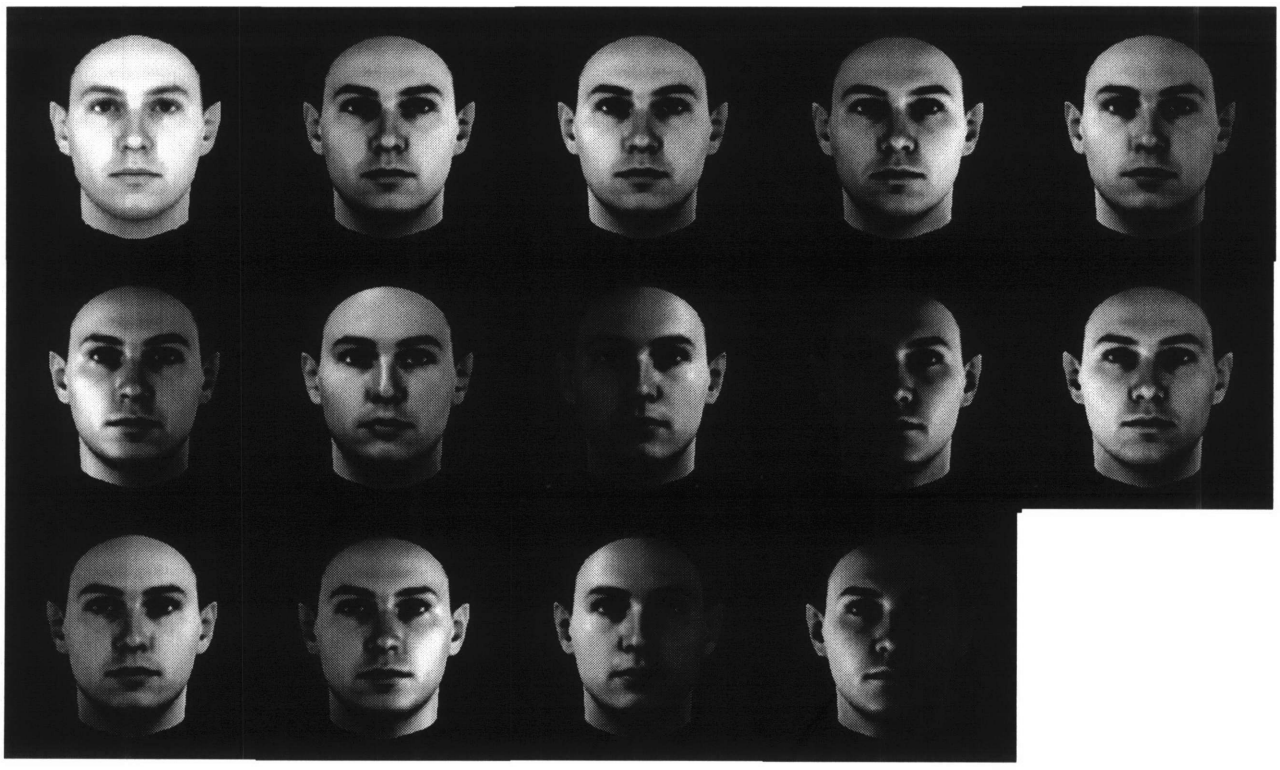


Figure 7.1: *New prototypes added to face model #1 to handle new lighting conditions. The image in the top left corner is the original reference image for comparison.*

7.2 Matching images with different illuminations

The new face model was tested on its ability to match novel images under various lighting conditions (including novel lighting conditions not included as prototypes). The results of various matches are shown in figures 7.2 and 7.3. The results show that new lighting conditions are modeled very well.

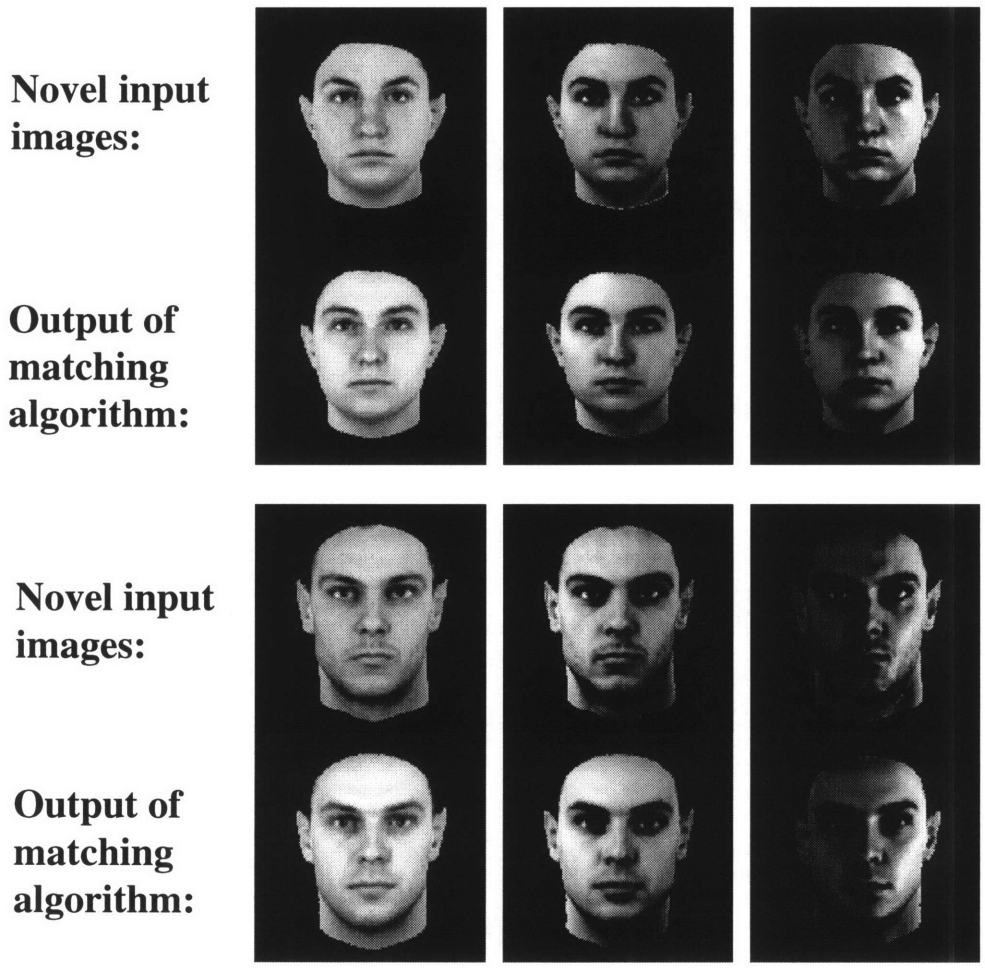


Figure 7.2: *Examples of matches by the the new face model for novel images with various illuminations. There are two rows of novel image/best match pairs. The leftmost novel image in each row is the novel image under the original, standard illumination.*

The idea of linearly combining images of an object taken under varying illumination has

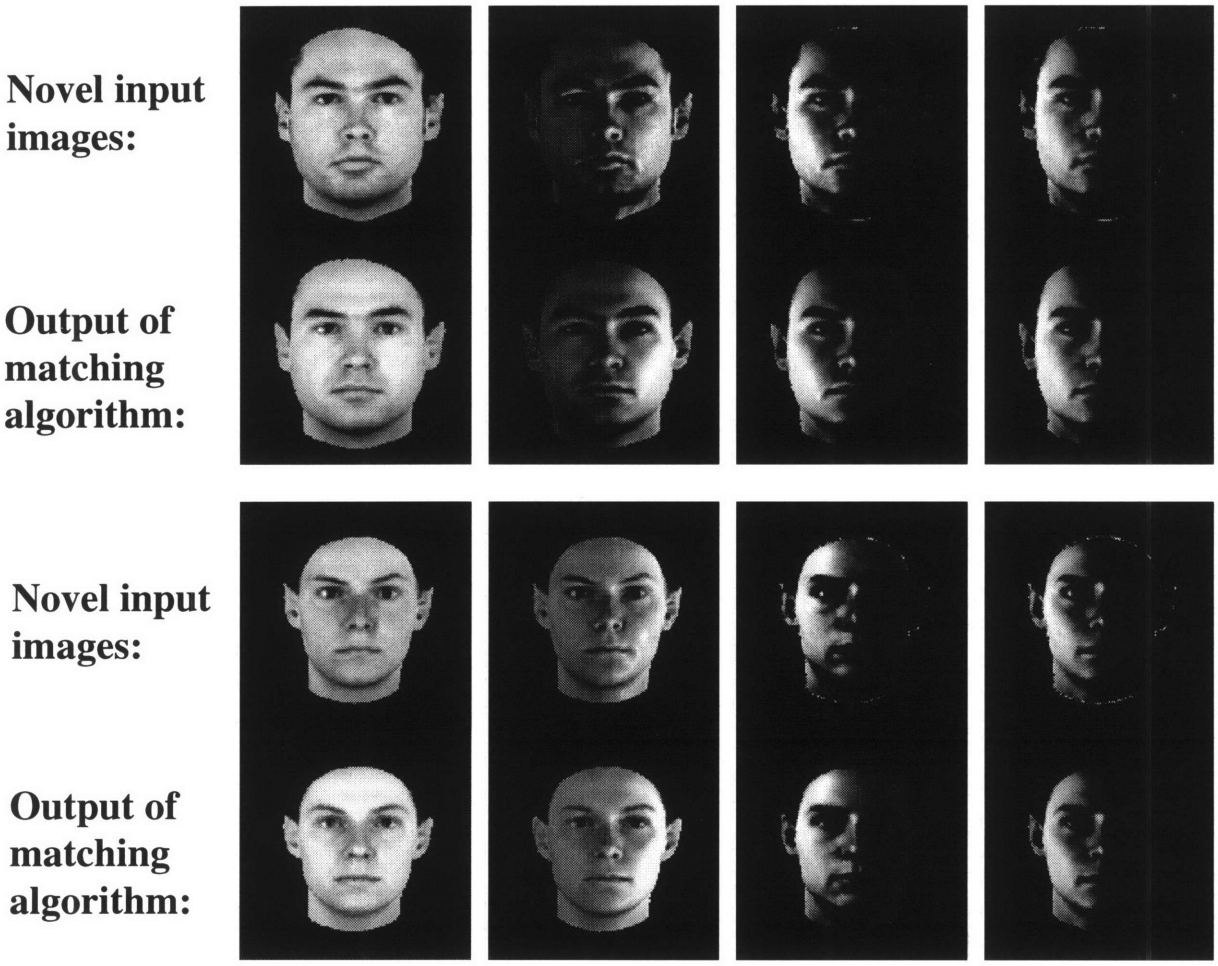


Figure 7.3: *More examples of matches by the the new face model for novel images with various illuminations. Again, the leftmost novel image in each row is the novel image under the original, standard illumination.*

also been explored by [Shashua, 1992] and [Hallinan, 1995]. Shashua showed that for Lambertian surfaces under point light sources, only three example images are needed to span the space of images of the surface under any lighting condition. Although faces are not exactly Lambertian, linear combinations of example images can still be used to approximate other illuminations. Similarly, Hallinan used example faces taken under many different lighting conditions and then found the eigenvectors of the set of images to determine the best set of illumination examples to span the space of possible images under different illuminations. Hallinan also achieved good results for modeling new faces under different illuminations using linear combinations of 5 illumination eigenvectors.

This same idea of modeling illumination changes by adding a few prototypes could also be applied to the problem of modeling changes in pose. For the face models shown, all prototypes and novel images are frontal views of faces. To handle input faces that are rotated, we could simply add examples of the reference face with different poses. Because of occlusion this will only work for small pose changes. From the work of [Beymer, 1996] we expect that changes in pose of up to 30 degrees could be handled in this way.

Chapter 8

Automatically Computing Prototype Correspondences

The distinguishing aspect of our multidimensional morphable models is that they are linear combinations of prototype shape and texture vectors and not of images (see also [Beymer and Poggio, 1996]). The prototypical images must be vectorized first, that is correspondence must be computed among them.

This is a key step and in general a difficult one. It needs to be done only once at the stage of developing the model. At run-time no further correspondence is needed. In fact matching a model to a novel image yields the correspondence to the novel image. In some cases, we computed correspondence among the prototypes with automatic techniques such as optical flow [Bergen and Hingorani, 1990]. Sometimes, however, we were forced to use interactive techniques requiring the user to specify at least some of the correspondences (see for instance [Lines, 1996]). An automatic technique that could set prototypes in correspondence would be therefore desirable even if very slow. In addition, any claim of biological plausibility would require the demonstration of such a technique.

In this chapter we describe a bootstrapping technique that seems capable of computing correspondence between prototypical images in cases in which standard optical flow algorithms fail.

8.1 Adding to a morphable model

Suppose that we have a morpable model consisting of N protoypes in correspondence. It is tempting to try to use it to compute the correspondence to a novel image of an object of the same class so that it can be added to the set of prototypes. The obvious flaw in this strategy is that if the morphable model can compute good correspondence to the new image then there is no need to add it to the morphable model since it will not increase its expressive power. If the morphable model cannot compute good correspondences, then the new prototype cannot be incorporated as such. A possible way out of this conundrum is to bootstrap the morphable model by using it together with an optical flow algorithm.

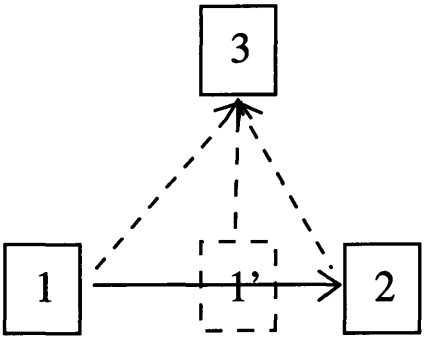


Figure 8.1: Given the morphable model provided by the combination of image 1 and image 2 (in correspondence), the goal is to find the correspondence between image 1 and the novel image 3. Our solution is to first find the linear combination of image 1 and image 2 that is closest to image 3 (this is image 1') and then find the correspondences from image 1' to image 3 using optical flow. The two flow fields can then be composed to yield the desired flow from image 1 to image 3.

8.2 The basic recursive step

Suppose that an existing morphable model is not powerful enough to match a new image and thereby find correspondence with it. The idea behind the bootstrapping algorithm is first to find rough correspondences to the novel image using the (inadequate) morphable model

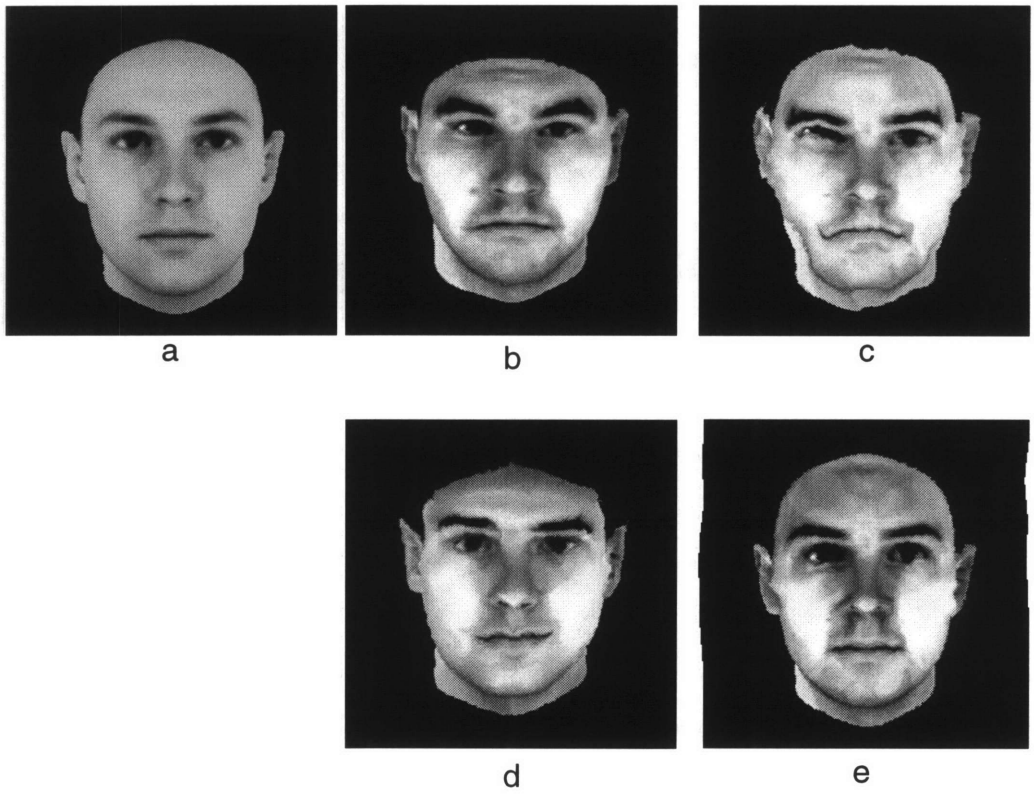


Figure 8.2: This figure shows the basic idea behind bootstrapping. Image (a) is the reference face. Image (b) is a prototype. Image (c) is the image resulting from backward warping the prototype onto the reference face using the correspondences found by an optical flow algorithm. Image (d) is the model image which best matches the prototype using a model consisting of 20 prototypical faces (which did not include image (b)). Image (e) is the image resulting from backward warping the prototype onto the reference face using the flow field which was composed from matching the face model and then running an optical flow algorithm between image (d) and image (b) to further improve the correspondences. This is the basic step of the bootstrapping algorithm.

and then to improve these correspondences by using an optical flow algorithm. This idea is illustrated in figure 8.1. In the figure, a model consisting of image 1 and image 2 (and the pixelwise correspondences between them) is first fit to image 3. Call image 1' the best fitting linear combination of images 1 and 2. The correspondences are then improved by running an optical flow algorithm between the intermediate image 1' and image 3. The two resulting flow fields are combined (as described in Appendix A) to obtain a flow field from image 1

to image 3. Notice that this technique can be regarded as a class specific regularization of optical flow, which constrains appropriately the correspondence.

8.2.1 Example

An example of our basic step is shown in figure 8.2. In this figure, an optical flow algorithm is used to find the correspondences from image (a) to image (b). The resulting correspondences are not very good as shown by image (c) which is the backward warp of image (b) according to the correspondences found by optical flow. Image (c) should have the texture of image (b) and the shape of image (a). A better way to find the correspondences to image (b) is to first fit a small model of faces to image (b), by using as a model 20 prototype face images (with known correspondences). The model was matched to image (b) as described in chapter 4. The resulting best match is shown as image (d). Next, optical flow was run between image (d) and image (b) to further improve the correspondences found by the matching algorithm. The two correspondence fields were combined (see Appendix A) to get the correspondences from image (a) to image (b). Image (e) is the backward warp of image (b) according to the final correspondence. A comparison of image (c) with image (e) shows that better correspondences are found by our basic recursive step relative to just using optical flow.

8.3 A bootstrapping algorithm for creating a morphable model

The idea of bootstrapping is to start from a small morphable model consisting of just 2 prototypical images and to increase its size (and representational power) by iterating the recursive step described above, progressively adding new images by setting them in correspondence with the model.

There are two main problems with building a morphable model. The first one is to choose the reference image, relative to which shape and texture vectors are represented. The second

is to automatically compute the correspondences even in cases in which optical flow fails.

In principle, any example image could be used as the reference image. However, small peculiarities in an image can influence strongly the matching process. Thus, an image which is close to all images is more reliable, since the computation of the correspondence is more stable for small distortions than for bigger ones. The average image of the whole data set, for which the average distance to the whole data set is by definition at minimum, is the optimal reference image. Since the correspondences between the images cannot be computed correctly in one step, the average has to be computed in an iterative procedure. Starting from an arbitrary image as the preliminary reference, a (noisy) correspondence between all other images and this reference is first computed using an optical flow algorithm. On the basis of these correspondences an average image can be computed, which now serves as a new reference image. This procedure of computing the correspondences and calculating a new average image is repeated until a stable average (vectorized) image is obtained.

The correspondence fields obtained through the optical flow algorithm from this final average image to all the examples are usually far from perfect. The bootstrapping idea is to improve the correspondences by applying iteratively the basic step described above while also increasing the expressive power of the morphable model. We could incorporate into the morphable model one new image at each timestep. The problem with this is that it is not clear how to automatically determine for which new example image the correspondences have been computed accurately. One possibility is to backward warp each example image according to the computed correspondence field. To a human observer, mistakes in the correspondence field usually lead to obvious errors in the backwarp warped face. However, the identification of these errors seems to come from people's knowledge of human faces. There does not appear to be a straightforward way to automatically identify such errors. For example, we did not find a correlation between the presence of errors (as determined by a person) and the L_2 error between the backward warped example image and the reference image.

Instead of incorporating one new example image at every iteration, we have implemented an equivalent algorithm in which the first step is to form a morphable model from the correspondences obtained from all images with optical flow. Since some of these correspondence fields are not correct and all are noisy, this algorithm uses only the most significant fields as provided by a standard PCA decomposition of the shape and the texture vectors. Instead of adding new images, the algorithm increases with successive iterations the number of principal components, ordered according to the associated eigenvalues (the allowed range of parameters of the selected principal components can also be increased with a similar effect). At each iteration a morphable model is selected and used to match each image. The optical flow algorithm estimates correspondence between the image and the approximation provided by the morphable model. This field is then added to the correspondence field implied by the matched model, giving a new correspondence field between the reference image and the example. The correspondence fields, obtained by this procedure, will finally lead to a new average image and also to new principal components which can be incorporated in an improved morphable model. Iterating this procedure with increasing expressive power of the model (by increasing the number of principal components) leads to stable correspondence fields between the reference image and the examples. The number of iterations as well as the increasing complexity of the model can be regarded as regularization parameters of this bootstrapping process.

8.3.1 Pseudo code of an efficient algorithm

1A: Selecting a reference image.

Select an arbitrary image I_i as reference image I_{ref}.
Until convergence do {
  For all I_i {
    Compute correspondence field S_i between I_{ref} and I_i using optical flow.
    Backward warp I_i onto I_{ref} using S_i to get the texture map T_i.

} end **For**

Compute average over all S_i and T_i

Forward warp $T_{average}$ using $S_{average}$ to create $I_{average}$

Convergence test: is $I_{average} - I_{ref} < limit$?

Copy $I_{average}$ to I_{ref}.

} end **Until**

1B: Computing the correspondence.

Until number n of principal components used in the morphable model is maximal {

Perform a principal component analysis on S_i and separately on T_i.

Select the first n principal components for the morphable model.

Approximate each I_i by the morphable model with I_i^{model}.

Compute correspondence field S_i' between I_i^{model} and I_i using optical flow.

Combine S_i' and S_i^{model} to yield S_i^{new}.

Backward warp I_i onto I_{ref} using S_i^{new} to get the texture map T_i.

Copy all S_i^{new} to S_i.

Increase number n of principal components used in the morphable model.

end **Until** }

8.4 Results

The method described in the previous sections was tested on two different classes of images. One class is frontal views of human faces and the second is handwritten digits.

8.4.1 Face images

The 100 face images shown in figures 5.1 through 5.3 were used to test the bootstrapping algorithm.

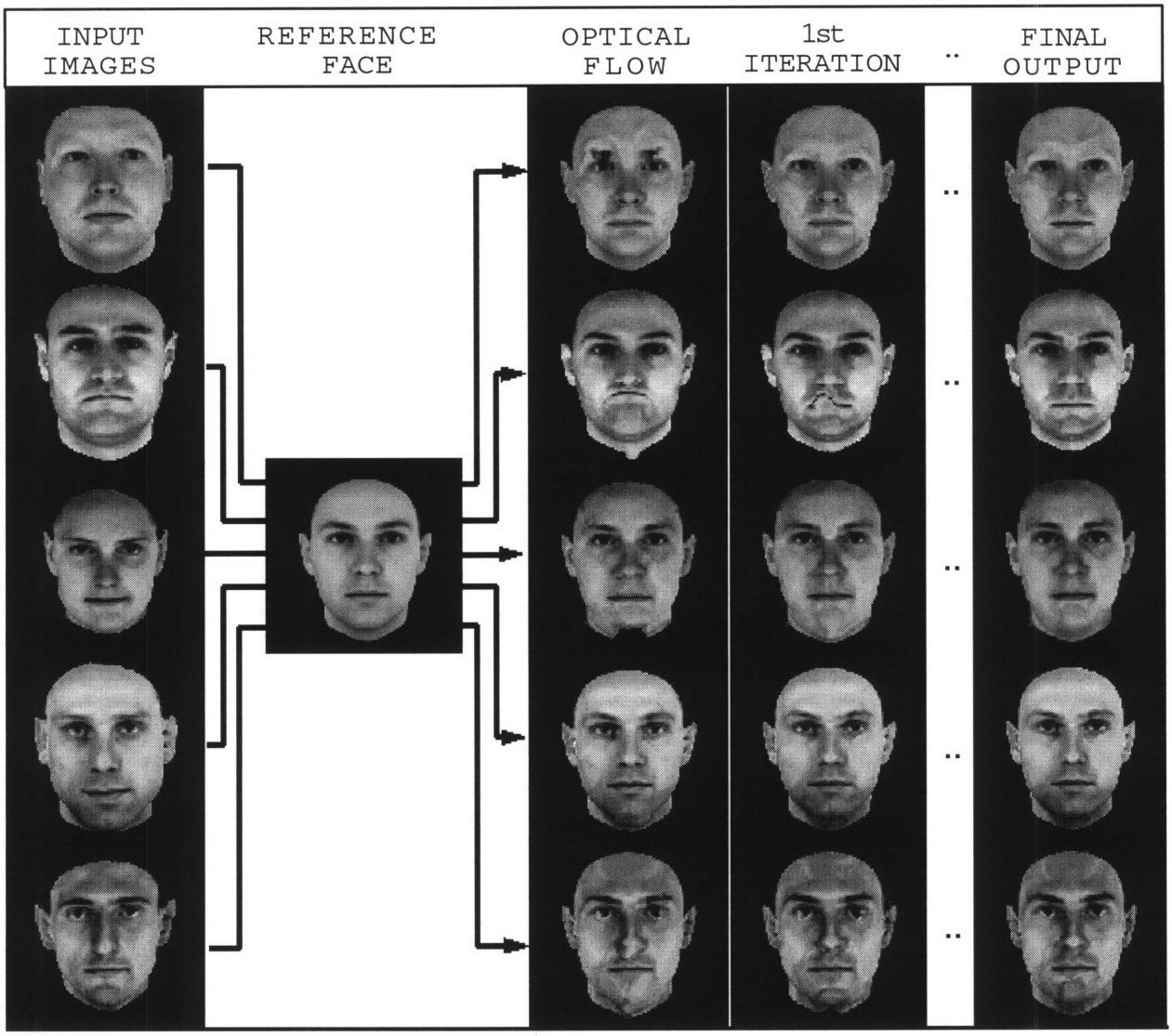


Figure 8.3: Five of the most difficult faces in our data set. The correspondence between face images (left column) and a reference face can be visualized by backward warping of the face images onto the reference image (three columns on the right). The correspondence obtained through the optical flow algorithm does not allow a correct mapping (center column). The first iteration with a morphable model consisting of two principal components already yields a significant improvement (top row). After four iterations with 10, 30 and 80 components, respectively, all correspondences were correct (right column)

Figure 8.4: The left image shows the reference face with a number of points marked with an "X". The right image shows the corresponding points on a prototype face. These correspondences were found using the automatic bootstrapping algorithm.

The method described in the previous sections was successfully applied to all face images available.

The step (1A) involving synthesis of the reference (average) image was tested for each image as a starting image in the algorithm. As a convergence criteria we used a theshold on the minimum average change of the pixel grey values. (The threshold was 0.3, whereas the grey level range was 256). The threshold was reached in every case within 5 iterations and mostly after 3. The final reference images could not be distinguished under visual inspection. One of these reference images is shown in the second column of figure 8.3; the same reference image was used for the final correspondence finding procedure.

Optical flow yields the correct correspondence between the reference image and each example image only in 80% of all cases. In the remaining cases the correspondence is partly incorrect, as shown in figure 8.3. The center column shows the images which result from backward warping the face images (left column) onto the reference image using the correspondence fields obtained through the optical flow algorithm. In the first iteration of the

correspondence finding procedure the first 2 principal components of the shape vectors (that is of the correspondence fields) and of the texture vectors are used in the morphable model. Then the correspondence field provided by matching with the morphable model is combined (as described in Appendix A) with the correspondence field obtained by the optical flow algorithm between the face image and its morphable model approximation. The backward warps using these correspondence fields are shown in the fourth column. The correspondence fields were iterated by slowly increasing the number of principal components used in the morphable model. After four iterations with 2, 10, 30 and 80 principal components respectively, the correspondence fields between the reference face and all example images did not reveal any obvious errors (right column). Figure 8.4 shows the reference image and a prototype image and a few of the corresponding points found by the bootstrapping algorithm. Each "x" on the reference image corresponds to the "x" one would expect on the prototype image.

Modeling the Test Images with 80 Principal Components		
Bootstrapping Steps	$L_2/pixels$	*Mahalanobis distance*
Optical Flow only	7.78	2.11
2 Principal Components	7.94	2.06
10 Principal Components	7.85	2.02
30 Principal Components	7.95	1.99
80 Principal Components	8.01	1.97

Table 8.1: *Results of matching a model consisting of 80 principal components on 30 test images after each step of the bootstrapping algorithm. One can see that the average Mahalanobis distance decreases after each iteration of bootstrapping implying that the model is improving. Conversely, the average L_2 error (between the reference image and the texture vector of each prototype) increases slightly.*

In a second experiment, 30 test face images were used to evaluate the ability of the morphable model to match novel faces. This experiment was used to document the improvements and changes within the morphable model during the bootstrapping. The bootstrapping algorithm was run on the training set as described earlier. After each bootstrapping step the L_2 norm between the reference texture vector and the texture vector of each example image was computed. Interestingly, the L_2 norm does not always reflect the improvement in correspondence as observed by visual inspection. The L_2 norm may even increase slightly

during the process (see Table 8.1).

After each bootstrapping step the 30 test images were approximated by the morphable model formed by the first 80 principal components obtained on the training set of 100 images. The Mahalanobis distance of each approximation to the average face image was computed. During the bootstrapping procedure the Mahalanobis distance decreased in the average over all test images as shown in table 8.1.

8.4.2 Digits

Data set and Preprocessing

The images used in these experiments were from the US postal service database (262 examples for each of the 10 digits). The original resolution of 16-by-16 pixels was increased to 32-by-32 pixels and the images were blurred with a Gaussian 5-by-5 kernel.

Evaluation

The bootstrapping algorithm was used for all 10 digits without modification. For each digit we obtained a shape model from the first 250 digits in the dataset. The reference image (average shape) is shown in the dashed boxes in figure 8.5. After computing the reference image and the initial correspondence fields with optical flow, new correspondence fields were obtained using 4 iterations of the bootstrapping algorithm. During the 4 iterations the number of principal components used in the algorithm was increased from 2 to 10, 30 and 80, respectively. Figure 8.5 shows the first 5 principal shape components of the final shape model.

The models obtained by the bootstrapping algorithm were used to match digits which where not part of the training set. In figure 8.6 new images of the digit 3 are approximated with three different models of digits. Clearly the "3" model approximates well each of the

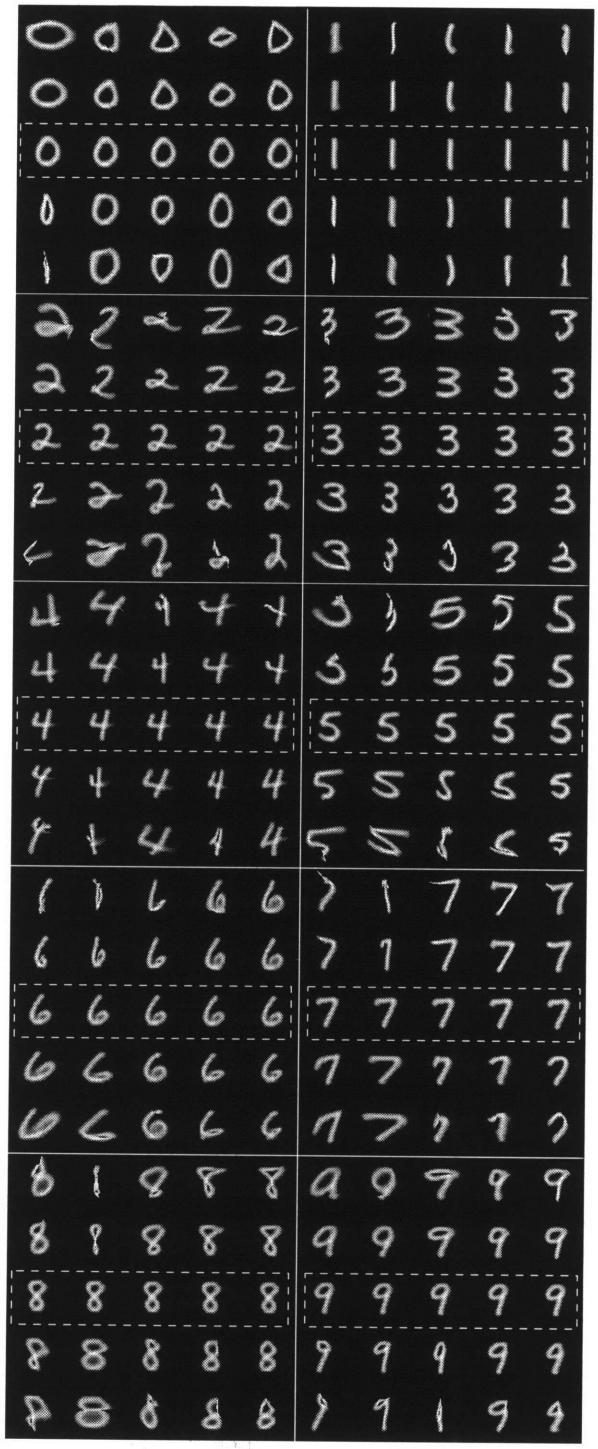


Figure 8.5: For each of the 10 digits the figure shows the first five shape eigenvectors (left to right) of the model (obtained from 250 prototypical digits). Each column displays how each shape eigenvector changes relative to the average digit (in dashed box). The range of the coefficient ranges from +5 (top) to -5 standard deviations (bottom) of each eigenvector.

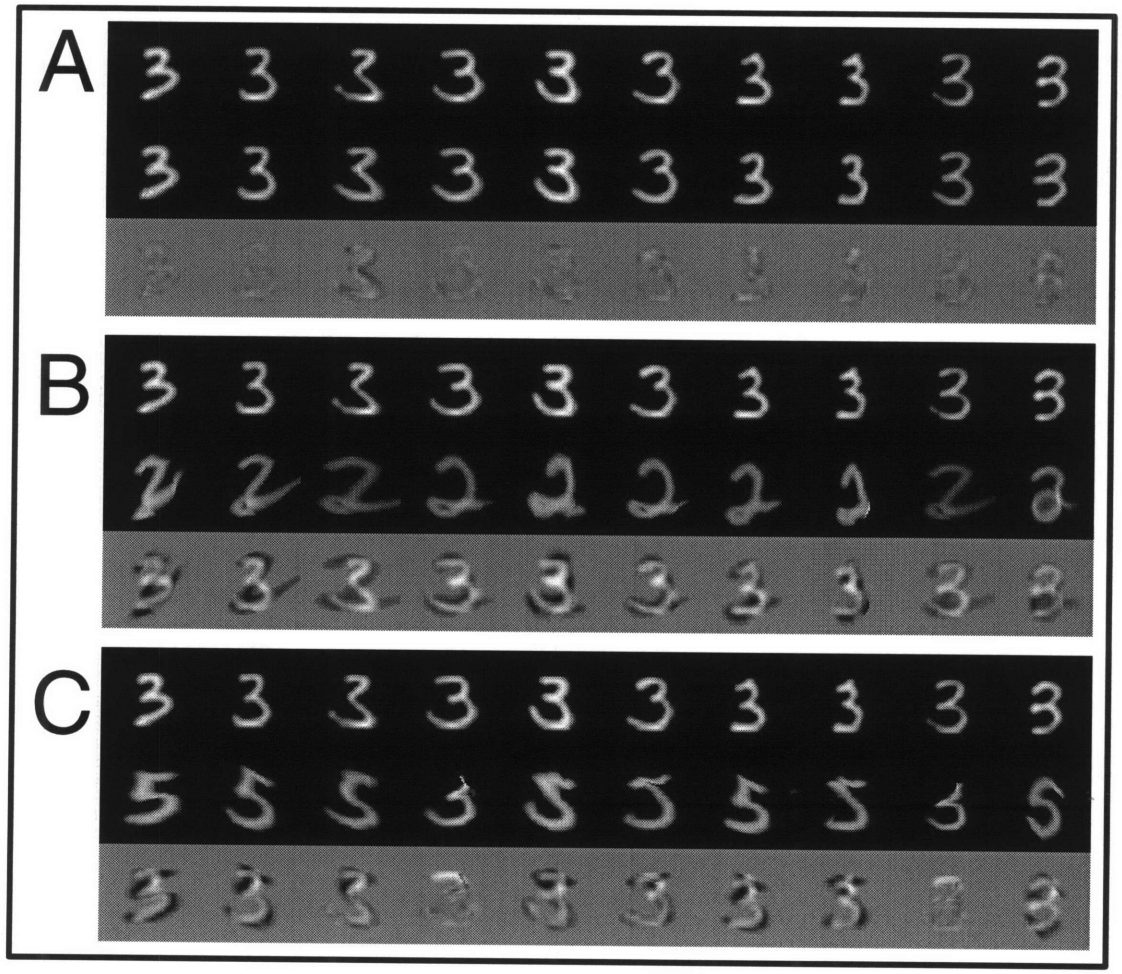


Figure 8.6: 10 examples of the digit 3 are approximated by 3 different shape models: in A a model for "3", in B for "2's" and in C for "5's". In each case the top row shows the target "3's", the center row shows the optimal approximation by the model and the third row shows the difference between the top and center row. Each model, obtained automatically by the bootstrapping procedure from 250 prototypes, consisted of the first 20 shape principal components.

new "3"'s, whereas the "5" and the "2" models provide very poor approximations. These results suggest that the digit models obtained with bootstrapping could be used successfully for recognition as well as for image compression.

8.5 Conclusions

The bootstrapping algorithm we described is not a full answer to the problem of computing correspondence between prototypes. It provides however an initial and promising solution to the very difficult problem of automatic synthesis of the morphable models from a set of prototypical examples. Notice that we have used optical flow as one part of our bootstrapping algorithm. In principle other matching techniques could be used within our bootstrapping scheme.

Chapter 9

Comparison to Eigenfaces

The eigenface approach ([Kirby and Sirovich, 1990], [Turk and Pentland, 1991]) is a well known approach in computer vision, which uses linear combinations of the raw example images which is superficially similar to morphable models. Morphable models rely instead on the linear combination of shape and texture vectors associated with the set of prototypical images. It is natural to ask how the two approaches compare. We describe here two exemplary cases.

9.1 "2's" example

To illustrate the importance of the vectorized representation and therefore of setting the prototype images in pixelwise correspondence, consider a set of prototypes for the numeral 2 shown in figure 9.1. We will first use this set of prototypical images and apply the eigenimage approach which does not use pixelwise correspondences. Note that the "2" images are in rough alignment and are normalized in scale. Figure 9.2 shows the mean image and the eigenimages derived from the set of prototypes. Note that the eigenimage "2's" are noisy and show, as expected, the ghosts of the prototype images of which they are linear combinations. Figure 9.3 shows some typical matches of the eigenvector model to novel images of "2's". The matches are not very good.

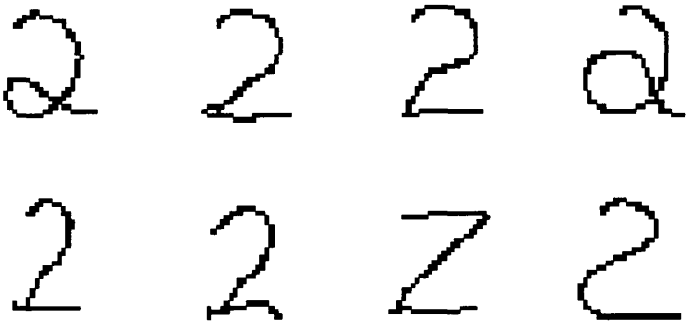


Figure 9.1: Prototype examples for a class of handwritten "2's".

Next we contrast this to the matching produced using our shape model based on pixelwise correspondences. Figure 9.4 shows the eigenvector representation for the shape vectors of the "2's". The mean image was obtained by computing the mean correspondence field from the prototype correspondence fields and then rendering it by warping the reference image according to the mean correspondence field. Similarly, the eigenvector images were obtained by finding the eigenvectors of the prototype shape vectors and then warping the reference image according to the eigenvector correspondence fields. Figure 9.5 shows the matches to novel "2" images using the morphable model. They are significantly better than in the eigenimage case without correspondences.

9.2 Face example

As another example of the better match quality obtained by our image representation, consider the case of frontal views of faces. The eigenface model of (Turk, 1991) was computed from face database #1 shown in figures 5.1 - 5.3. These faces are aligned as described in section 5.1. One hundred face prototypes were used to define the model and all 100 eigenvectors were used in the eigenface representation. The middle column of figure 9.6 shows the matches to four novel faces found using the eigenface model. The right column of figure

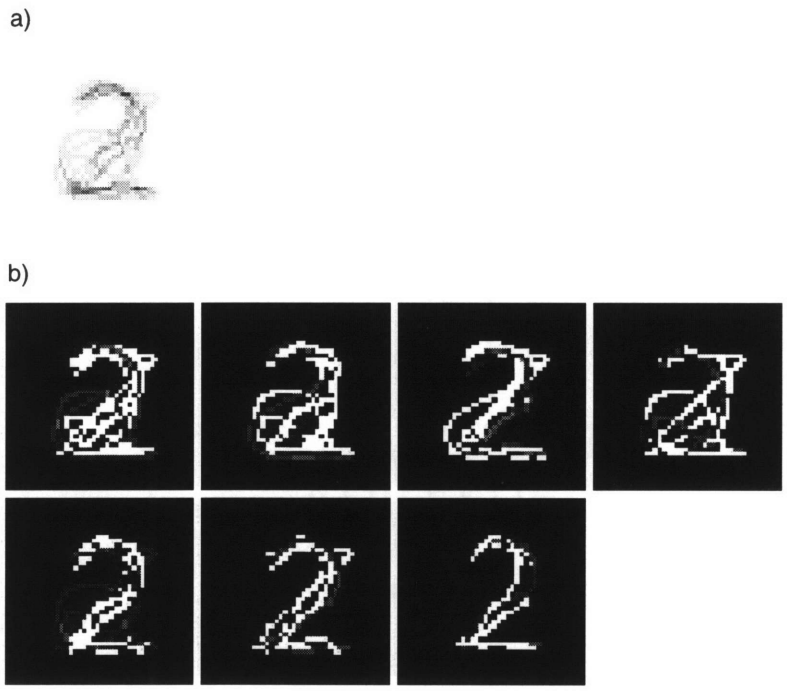


Figure 9.2: a) Mean image "2" found by averaging the prototype images of "2's". b) So-called eigenimages of "2's", that is eigenvectors computed from the set of images of prototype "2's" (without using pixelwise correspondences).

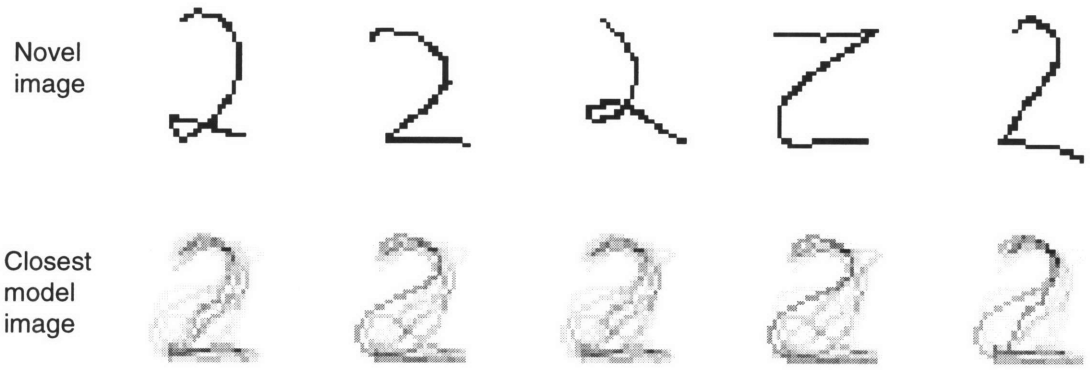


Figure 9.3: Example matches to novel "2's" using the eigenimage approach. Matching with eigenimages yields poor results.

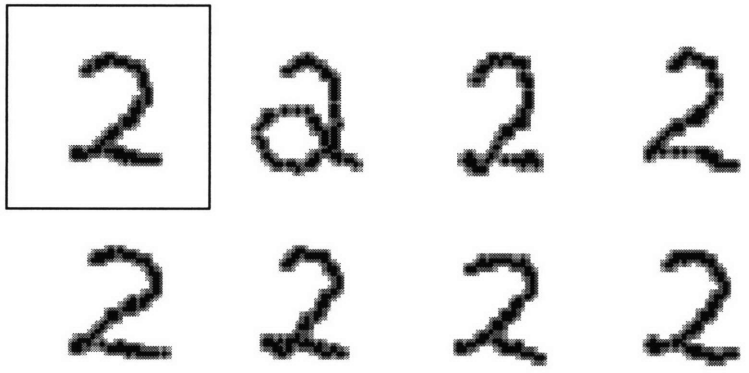


Figure 9.4: Mean "2" obtained by rendering the reference shape vector "2" which is equivalent to warping the reference "2" with the average of the prototype correspondence fields (top left in box) followed by the full set of the shape eigenvectors of the prototypes. They are rendered as images obtained as warps of the reference "2" by the eigenvector correspondence fields.

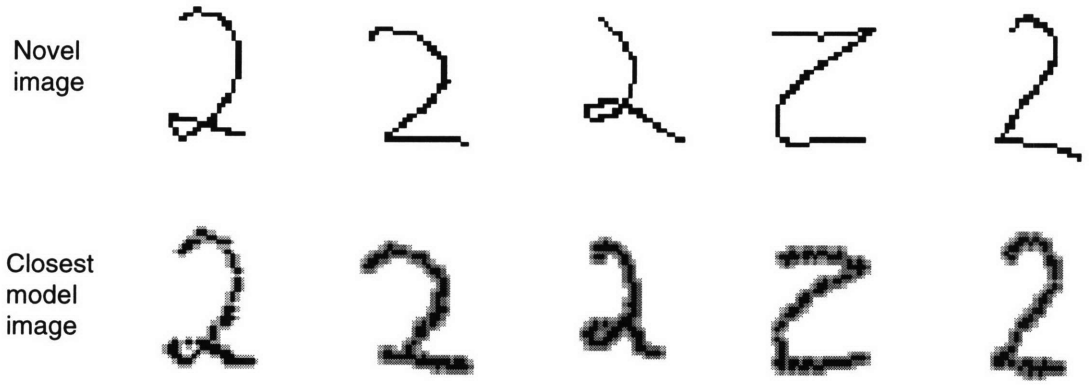


Figure 9.5: Example matches to novel "2's" using the morphable model approach. The output "2's" were blurred to fill in "holes" left after warping.

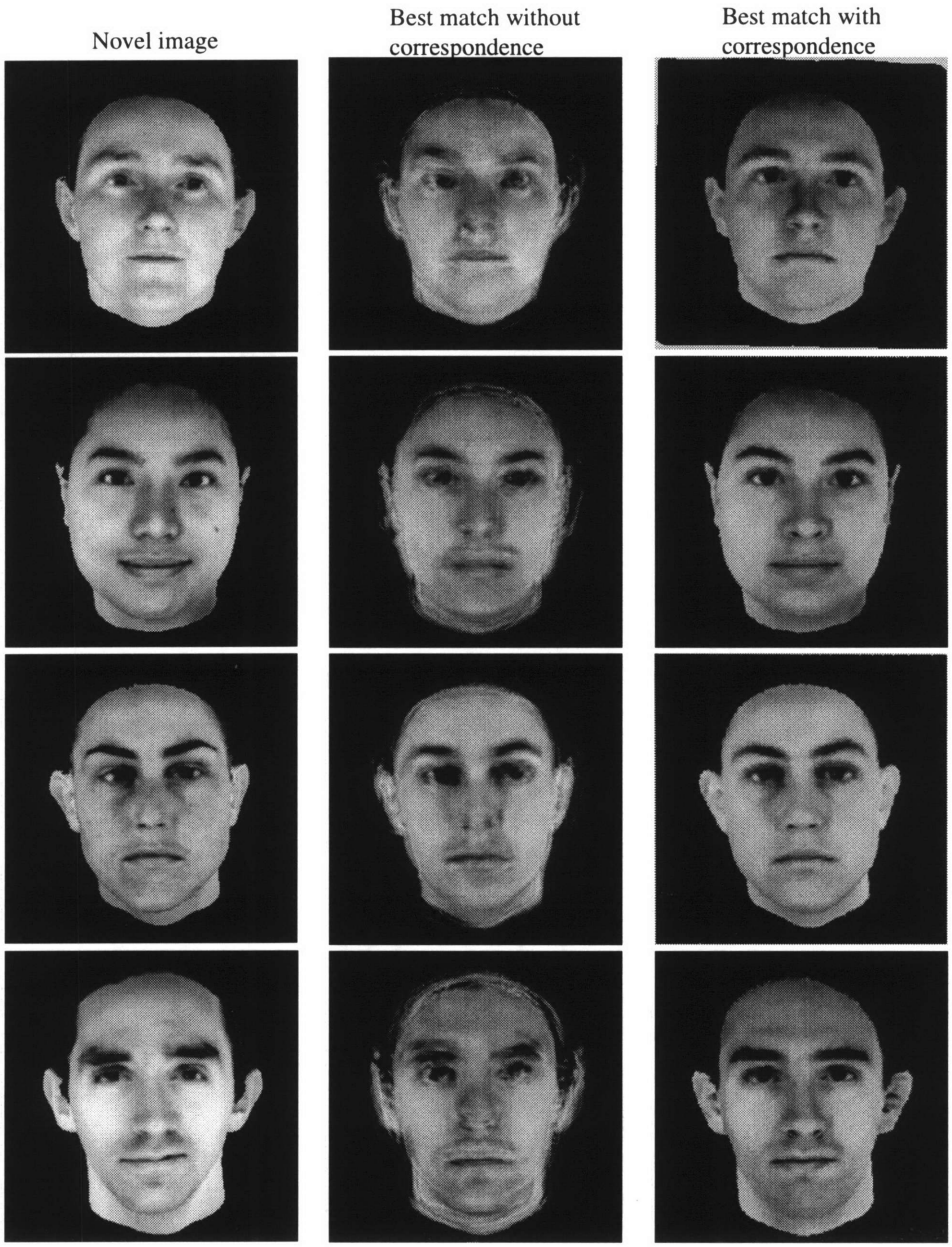


Figure 9.6: Example matches to novel faces (left) using the so-called eigenface approach (middle) – which does not use pixelwise correspondence – and using our approach (right) which uses pixelwise correspondences.

9.6 shows the best matches for the morphable model which uses pixelwise correspondences. The images produced by the eigenface approach are "fuzzy", and it is clear even from this qualitative comparison that the morphable model results in superior matching relative to the eigenface approach.

We have shown that the morphable model framework provides better matches to novel images than the eigenface approach. In addition, the eigenface approach is not robust to changes in translation, scale, rotation or occlusion whereas morphable models are.

Chapter 10

Hierarchical Morphable Models

In this chapter we will present some preliminary results on an extension of multidimensional morphable models. The idea is to split the prototype images into components (such as eyes, nose and mouth for face images) and create a model containing a hierarchy of components. At the bottom level of the hierarchy is a set of components which are themselves morphable models as defined previously. The upper levels of the hierarchy consist of linear combinations of the positions of the components.

10.1 Main idea

Figure 10.1 illustrates a hierarchical morphable model for a face with 2 levels to the hierarchy. The bottom level contains individual morphable models for the two eyes, the nose and the mouth. The next level of the hierarchy consists of linear combinations of the positions of the centers of each component in the example images. The output of the model uses these linear combinations of vectorized components and positions to synthesize an image containing all the components in the appropriate positions.

One advantage to such a model (relative to the standard morphable model) is that fewer prototypes are necessary for each component to achieve the same quality of matches. This is because each component will be less complex than the whole image and thus require fewer

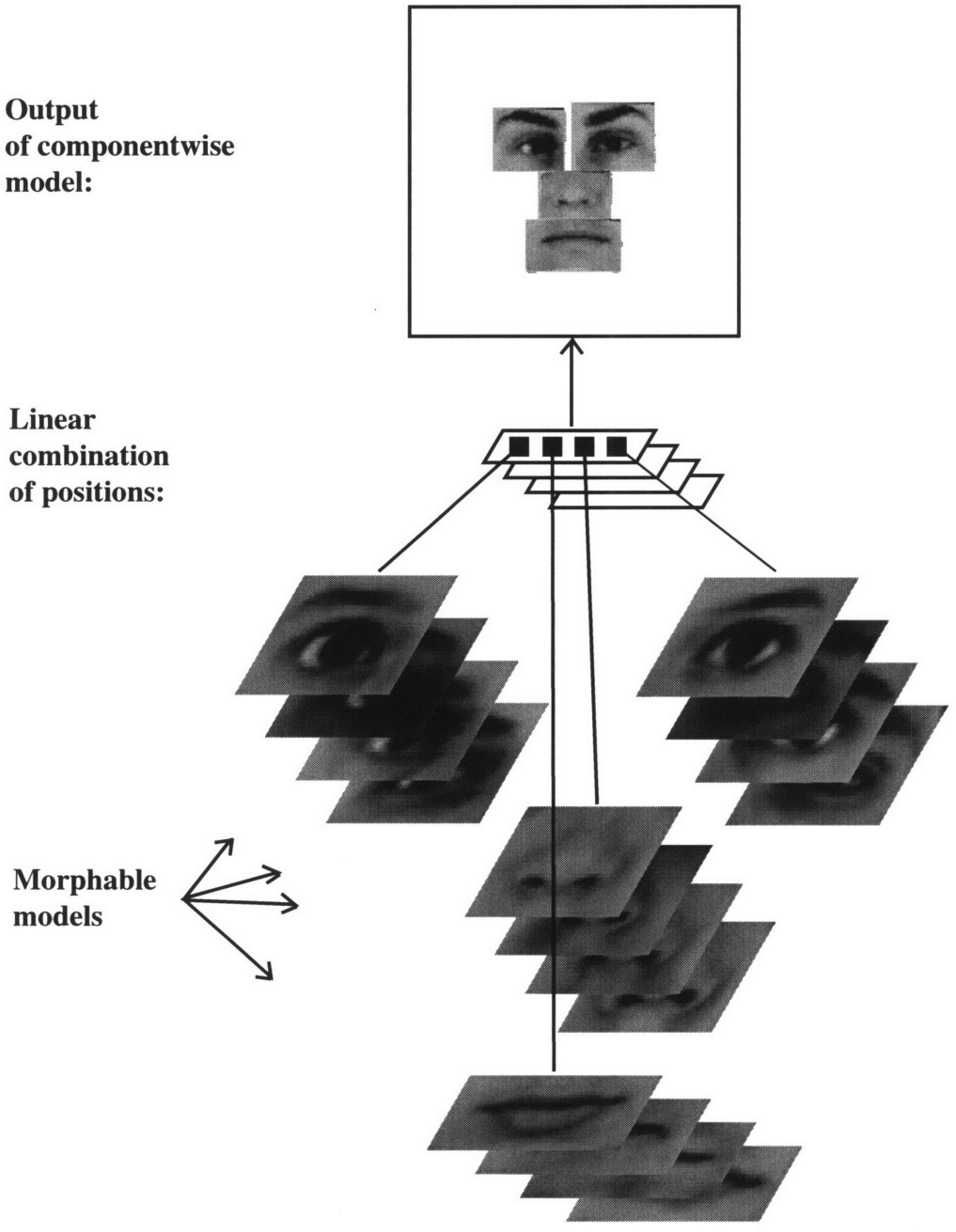


Figure 10.1: *An illustration of a hierarchical morphable model with 2 levels. The bottom level consists of component morphable models for the eyes, nose and mouth of a face. The next level consists of linear combinations of the positions of each component. A particular parameter setting yields an output face built from components.*

examples to span the possible component images. The model is also more expressive since there are more degrees of freedom. One can think of it as a piecewise linear model instead of just a linear model. Another advantage is that occlusion can be handled better using components. For example, if one component is occluded in the novel image then the other components can still be matched without being negatively affected by the occluded part of the image. With the standard morphable model, occluded images can be matched, but the quality of the match degrades with increasing amounts of occlusion (see figure 6.8).

The main disadvantage of the hierarchical model is that currently the components for each prototype image must be specified by hand. Ultimately, we would like for prototype images to be split into components automatically. Possible approaches to this problem are the work of [Bell and Sejnowski, 1995] or more brute force techniques which look for correlations in different regions of each prototype. A successful technique for automatically finding components does not need to be computationally efficient since such a procedure would only be done once to create a hierarchical morphable model. For now, however, to illustrate the idea of hierarchical models we will assume the components are specified by hand.

Another disadvantage to hierarchical models is the problem of "filling in" areas of occlusion in the novel input image. Depending on the application, however, this filling in process may not be needed. With the standard morphable model, small areas of occlusion in the input image are automatically "filled in" in the output model image by using information in the rest of the image. With a hierarchical model, this problem is harder to overcome. Since the components are independent it is not clear how to use the information for components which are not occluded to recontruct components which are. One possibility might be to fit a hierarchical model to the occluded input image and then use the parameters for the components of the hierarchical model which fit well to map to the parameters of an associated standard morphable model, and thus synthesize a whole image which fills in the occluded parts of the input image. We leave this problem for future work.

10.2 Manually specifying components

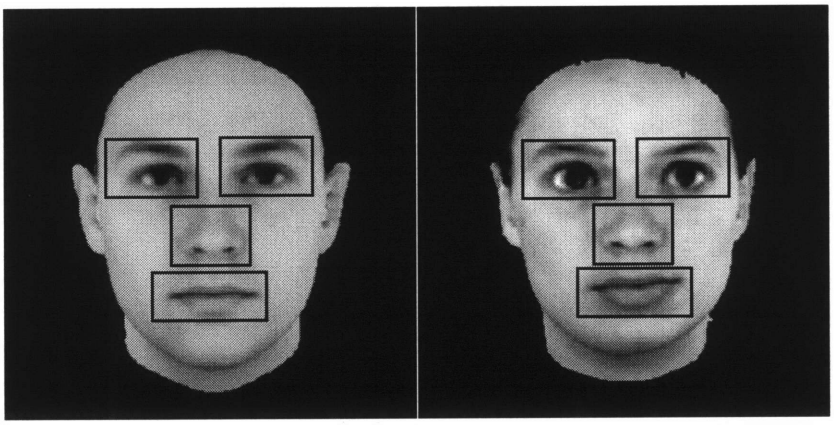


Figure 10.2: *An illustration of the manual tool used for selecting components. The face on the left is the reference face. The components are chosen once for this face. The face on the right is another prototype face. The components chosen on the reference face are specified on this prototype face.*

We are not currently addressing the problem of automatically finding components. Instead we have written a simple program which allows us to manually specify the components in each prototype by placing rectangles on each prototype to specify the region for each component. Figure 10.2 illustrates this process. Manually specifying components is not an unreasonable burden since it is only necessary in the model building phase. After this phase, the components of a novel input image are automatically found by the matching algorithm.

10.3 Formal specification

The formal model for hierarchical morphable models is similar to standard morphable models with the addition of parameters which control the position of each component.

As before we have a set of prototype images $I_0, \ldots, I_N$. I_0 is the reference image. The components are manually specified on I_0 by rectangular regions. The center of the rectangle along with the height and width give the size and position of each component. Let K be the number of components in each prototype.

Let C_k be the set of x, y coordinates of I_0 which are contained in component k, i.e. C_k is a rectangular region of I_0 defining component k.

For each prototype i, the kth component has correspondence field

$$S_i^k : \mathcal{R}^2 \rightarrow \mathcal{R}^2.$$

$S_i^k(x, y) = (\hat{x}, \hat{y})$ where x, y is a pixel in component k of the reference image and $\hat{x}, \hat{y}$ is the corresponding point in component k of prototype i. Note that there is a different set of shape vectors for each different component.

Similarly, T_i^k is the texture vector of component k for prototype i.

$$T_i^k(x, y) = I_i \circ S_i^k(x, y) \qquad \forall\, (x, y) \in C_k.$$

Finally, we add a new vector, P_i^k, to the model which is the displacement of the (x, y) position of the center of component k in prototype i (relative to its position in the reference image). In other words, P_i^k is simply a vector pointing from the center of component k in I_0 to the center of component k in I_i.

$$P_i^k = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.$$

As with shape and texture, the position of each component in any model image is constrained to be a linear combination of prototype component positions. The function $Q^k : \mathcal{R}^2 \rightarrow \mathcal{R}^2$ desribes this linear combination and is defined as

$$Q^k(x, y) = \sum_{l=1}^{M} a_l P_l^k + \begin{bmatrix} x \\ y \end{bmatrix} \tag{10.1}$$

where M is the number of prototype positions used. M is typically less then N (the number of prototypes) since the position vector space is much lower dimensional than the shape or texture space. The position vector space has dimensionality $2 \times K$ since each of the K components has 2 elements (x and y) in its position vector.

So the first pass at defining a hierarchical morphable model is all model images I^{model} such that for each component k and $(x, y) \in C_k$

$$I^{model} \circ Q^k \circ (\sum_{i=0}^{N} c_i S_i)(x, y) = \sum_{j=0}^{N} b_j T_j(x, y). \tag{10.2}$$

The element missing from the above equation is the affine transformations which allow for rotation, translation, scaling and shearing. We would like each component to have affine transformations independently as well as having a global affine transformation. This will allow an individual mouth component, for example, to rotate independently of the eyes and nose and at the same time allow the positions of all the components to translate, rotate or scale using the global affine parameters. We will add a component affine transformation, A^k, as defined before to each component model and a global affine mapping, B, to the position function, Q^k. Let

$$A^k(x, y) = (p_0^k x + p_1^k y + p_2^k,\ p_3^k x + p_4^k y + p_5^k) \tag{10.3}$$

and

$$B(x, y) = (e_0 x + e_1 y + e_2,\ e_3 x + e_4 y + e_5). \tag{10.4}$$

Now we can define the full hierarchical morphable model as the set of all images I^{model} such that for each component k and $(x, y) \in C_k$

$$I^{model} \circ B \circ Q^k \circ (A^k \circ \sum_{i=0}^{N} c_i^k S_i^k)(x, y) = \sum_{j=0}^{N} b_j^k T_j^k(x, y). \tag{10.5}$$

The mapping $(A^k \circ \sum_{i=0}^{N} c_i^k S_i^k)$ constrains the shape of component k to be a linear combination of prototype component shapes followed by a local affine transformation. The mapping $B \circ Q^k$ constrains the position of component k to be a linear combination of prototype component positions followed by a global affine transformation. The mapping $\sum_{j=0}^{N} b_j^k T_j^k$ constrains the texture of component k to be a linear combination of prototype component textures.

The parameters of the hierarchical morphable model are the shape parameters, $\mathbf{c}^k$, the texture parameters, $\mathbf{b}^k$, the position parameters $\mathbf{a}$, the component-level affine parameters

$\mathbf{p}^k$ and the global affine parameters $\mathbf{e}$. The index k goes from 1 to K (the number of components).

10.4 Matching the hierarchical morphable model

We use the same matching algorithm as before except with the equation for the new model:

$$E(\mathbf{c},\mathbf{b},\mathbf{a},\mathbf{p},\mathbf{e}) = \frac{1}{2}\sum_{k=1}^{K}\sum_{(x,y)\in C_k}\left[I^{novel}\circ B\circ Q^k\circ(A^k\circ\sum_{i=0}^{N}c_i^k S_i^k)(x,y)-\sum_{j=0}^{N}b_j^k T_j^k(x,y)\right]^2. \quad (10.6)$$

Stochastic gradient descent is used to minimize this error with respect to the model parameters. We list the necessary derivatives after first introducing some notational conveniences.

To simplify some of the notation, let

$$F^k = B\circ Q^k\circ A^k\circ\sum_{i=0}^{N}c_i^k S_i^k.$$

Also, recall that S_i^k is a vector function with two components in its output. To refer to these components separately, we will use the notation $S_i^k = (S_i^k.x, S_i^k.y)$.

$$\frac{\partial E}{\partial c_i^k} = \sum_{(x,y)\in C_k}\left[[I^{novel}\circ F^k(x,y)-\sum_{j=0}^{N}b_j^k T_j^k(x,y)]\,\frac{\partial I^{novel}\circ F^k(x,y)}{\partial c_i^k}\right]$$

$$\frac{\partial E}{\partial b_i^k} = -\sum_{(x,y)\in C_k}\left[[I^{novel}\circ F^k(x,y)-\sum_{j=0}^{N}b_j^k T_j^k(x,y)]\;T_j^k(x,y)\right]$$

$$\frac{\partial E}{\partial a_i} = \sum_{k=1}^{K}\sum_{(x,y)\in C_k}\left[[I^{novel}\circ F^k(x,y)-\sum_{j=0}^{N}b_j^k T_j^k(x,y)]\,\frac{\partial I^{novel}\circ F^k(x,y)}{\partial a_i}\right]$$

$$\frac{\partial E}{\partial p_i^k} = \sum_{(x,y)\in C_k}\left[[I^{novel}\circ F^k(x,y)-\sum_{j=0}^{N}b_j^k T_j^k(x,y)]\,\frac{\partial I^{novel}\circ F^k(x,y)}{\partial p_i^k}\right]$$

$$\frac{\partial E}{\partial e_i} = \sum_{k=1}^{K} \sum_{(x,y)\in C_k} \left[[I^{novel} \circ F^k(x,y) - \sum_{j=0}^{N} b_j^k T_j^k(x,y)] \frac{\partial I^{novel} \circ F^k(x,y)}{\partial e_i} \right]$$

$$\begin{aligned}
\frac{\partial I^{novel} \circ F^k(x,y)}{\partial c_i^k} &= \left(\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\bigg|_{B\circ Q^k\circ(A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \cdot \left(\frac{\partial B(x,y)}{\partial(x,y)}\bigg|_{Q^k\circ(A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \\
&\quad \cdot \left(\frac{\partial Q^k(x,y)}{\partial(x,y)}\bigg|_{A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \cdot \left(\frac{\partial A^k(x,y)}{\partial(x,y)}\bigg|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} \right) \\
&\quad \cdot \left(\frac{\partial \sum_{i=0}^N c_i^k S_i^k(x,y)}{\partial c_i^k} \right) \\
&= \frac{\partial I^{novel}(x,y)}{\partial(x,y)}\bigg|_{F^k(x,y)} \cdot \begin{bmatrix} e_0^k & e_1^k \\ e_3^k & e_4^k \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_0^k & p_1^k \\ p_3^k & p_4^k \end{bmatrix} \cdot \begin{bmatrix} S_i^k.x \\ S_i^k.y \end{bmatrix} \\
&= [(e_0^k p_0^k + e_1^k p_3^k) S_i^k.x + (e_0^k p_1^k + e_1^k p_4^k) S_i^k.y] \frac{\partial I^{novel}(x,y)}{\partial x}\bigg|_{F^k(x,y)} \\
&\quad + [(e_3^k p_0^k + e_4^k p_3^k) S_i^k.x + (e_3^k p_1^k + e_4^k p_4^k) S_i^k.y] \frac{\partial I^{novel}(x,y)}{\partial y}\bigg|_{F^k(x,y)}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial I^{novel} \circ F^k(x,y)}{\partial p_i^k} &= \left(\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\bigg|_{B\circ Q^k\circ(A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \cdot \left(\frac{\partial B(x,y)}{\partial(x,y)}\bigg|_{Q^k\circ(A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \\
&\quad \cdot \left(\frac{\partial Q^k(x,y)}{\partial(x,y)}\bigg|_{A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \cdot \left(\frac{\partial A^k(x,y)}{\partial p_i^k}\bigg|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} \right) \\
&= \frac{\partial I^{novel}(x,y)}{\partial(x,y)}\bigg|_{F^k(x,y)} \begin{bmatrix} e_0^k & e_1^k \\ e_3^k & e_4^k \end{bmatrix} \left(\frac{\partial A^k(x,y)}{\partial p_i^k} \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial I^{novel} \circ F^k(x,y)}{\partial a_i} &= \left(\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\bigg|_{B\circ Q^k\circ(A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \cdot \left(\frac{\partial B(x,y)}{\partial(x,y)}\bigg|_{Q^k\circ(A^k\circ\sum_{i=0}^N c_i^k S_i^k)(x,y)} \right) \\
&\quad \cdot \left(\frac{\partial Q^k(x,y)}{\partial a_i}\bigg|_{A^k\circ\sum_{i=0}^N c_i^k S_i^k(x,y)} \right)
\end{aligned}$$

$$= \left.\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\right|_{F^k(x,y)} \begin{bmatrix} e_0^k & e_1^k \\ e_3^k & e_4^k \end{bmatrix} \begin{bmatrix} p_i^k.x \\ p_i^k.y \end{bmatrix}$$

$$= (e_0^k p_i^k.x + e_1^k p_i^k.y) \left.\frac{\partial I^{novel}(x,y)}{\partial x}\right|_{F^k(x,y)} + (e_3^k p_i^k.x + e_1^k p_i^k.y) \left.\frac{\partial I^{novel}(x,y)}{\partial y}\right|_{F^k(x,y)}$$

$$\frac{\partial I^{novel} \circ F^k(x,y)}{\partial e_i} = \left(\left.\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\right|_{B \circ Q^k \circ (A^k \circ \sum_{i=0}^N c_i^k S_i^k)(x,y)}\right) \cdot \left(\left.\frac{\partial B(x,y)}{\partial e_i}\right|_{Q^k \circ (A^k \circ \sum_{i=0}^N c_i^k S_i^k)(x,y)}\right)$$

$$\left.\frac{\partial A^k(x,y)}{\partial p_0^k}\right|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} = \left[\sum_{i=0}^N c_i S_i^k.x(x,y),\ 0\right]^T$$

$$\left.\frac{\partial A^k(x,y)}{\partial p_1^k}\right|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} = \left[\sum_{i=0}^N c_i S_i^k.y(x,y),\ 0\right]^T$$

$$\left.\frac{\partial A^k(x,y)}{\partial p_2^k}\right|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} = [1,0]^T$$

$$\left.\frac{\partial A^k(x,y)}{\partial p_3^k}\right|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} = \left[0,\ \sum_{i=0}^N c_i S_i^k.x(x,y)\right]^T$$

$$\left.\frac{\partial A^k(x,y)}{\partial p_4^k}\right|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} = \left[0,\ \sum_{i=0}^N c_i S_i^k.y(x,y)\right]^T$$

$$\left.\frac{\partial A^k(x,y)}{\partial p_5^k}\right|_{\sum_{i=0}^N c_i^k S_i^k(x,y)} = [0,1]^T$$

$$\left.\frac{\partial B(x,y)}{\partial e_0}\right|_{Q^k \circ A^k \circ \sum_{i=0}^N c_i^k S_i^k(x,y)} = \left[(Q^k \circ A^k \circ \sum_{i=0}^N c_i S_i^k(x,y)).x,\ 0\right]^T$$

$$\left.\frac{\partial B(x,y)}{\partial e_1}\right|_{Q^k \circ A^k \circ \sum_{i=0}^N c_i^k S_i^k(x,y)} = \left[(Q^k \circ A^k \circ \sum_{i=0}^N c_i S_i^k(x,y)).y,\ 0\right]^T$$

$$\left.\frac{\partial B(x,y)}{\partial e_2}\right|_{Q^k \circ A^k \circ \sum_{i=0}^N c_i^k S_i^k(x,y)} = [1,0]^T$$

$$\left.\frac{\partial B(x,y)}{\partial e_3}\right|_{Q^k \circ A^k \circ \sum_{i=0}^{N} c_i^k S_i^k(x,y)} = \left[0, (Q^k \circ A^k \circ \sum_{i=0}^{N} c_i S_i^k(x,y)).x\right]^T$$

$$\left.\frac{\partial B(x,y)}{\partial e_4}\right|_{Q^k \circ A^k \circ \sum_{i=0}^{N} c_i^k S_i^k(x,y)} = \left[0, (Q^k \circ A^k \circ \sum_{i=0}^{N} c_i S_i^k(x,y)).y\right]^T$$

$$\left.\frac{\partial B(x,y)}{\partial e_5}\right|_{Q^k \circ A^k \circ \sum_{i=0}^{N} c_i^k S_i^k(x,y)} = [0,1]^T$$

To compute the spatial derivatives of the novel image, a finite difference approximation may be used:

$$\left.\frac{\partial I^{novel}(x,y)}{\partial(x,y)}\right|_{F^k(x,y)} \approx \left[\tfrac{1}{2}(I^{novel}(F^k(x,y)+(1,0)) - I^{novel}(F^k(x,y)) - (1,0)),\right.$$

$$\left.\tfrac{1}{2}(I^{novel}(F^k(x,y)+(0,1)) - I^{novel}(F^k(x,y) - (0,1)))\right].$$

10.5 Example hierarchical morphable model

As an example of a hierarchical morphable model we will use faces split into eyes, nose and mouth components as shown in figure 10.1. We use the face database shown in figures 5.1 - 5.3 as prototypes. The components were selected manually as described in section 10.2. Fifty prototypes were used for each component. We found that 50 prototypes were enough to get very good matches. A principal component analysis could be used to further reduce the number of prototypes needed, but this was not done in this example. Figure 10.3 shows 10 of the 50 prototypes used to define each component.

The correspondence fields for each component were taken from the full image correspondence fields already computed for these faces using the bootstrapping algorithm (chapter 8).

Because we are using four components in this hierachical morphable model, there are

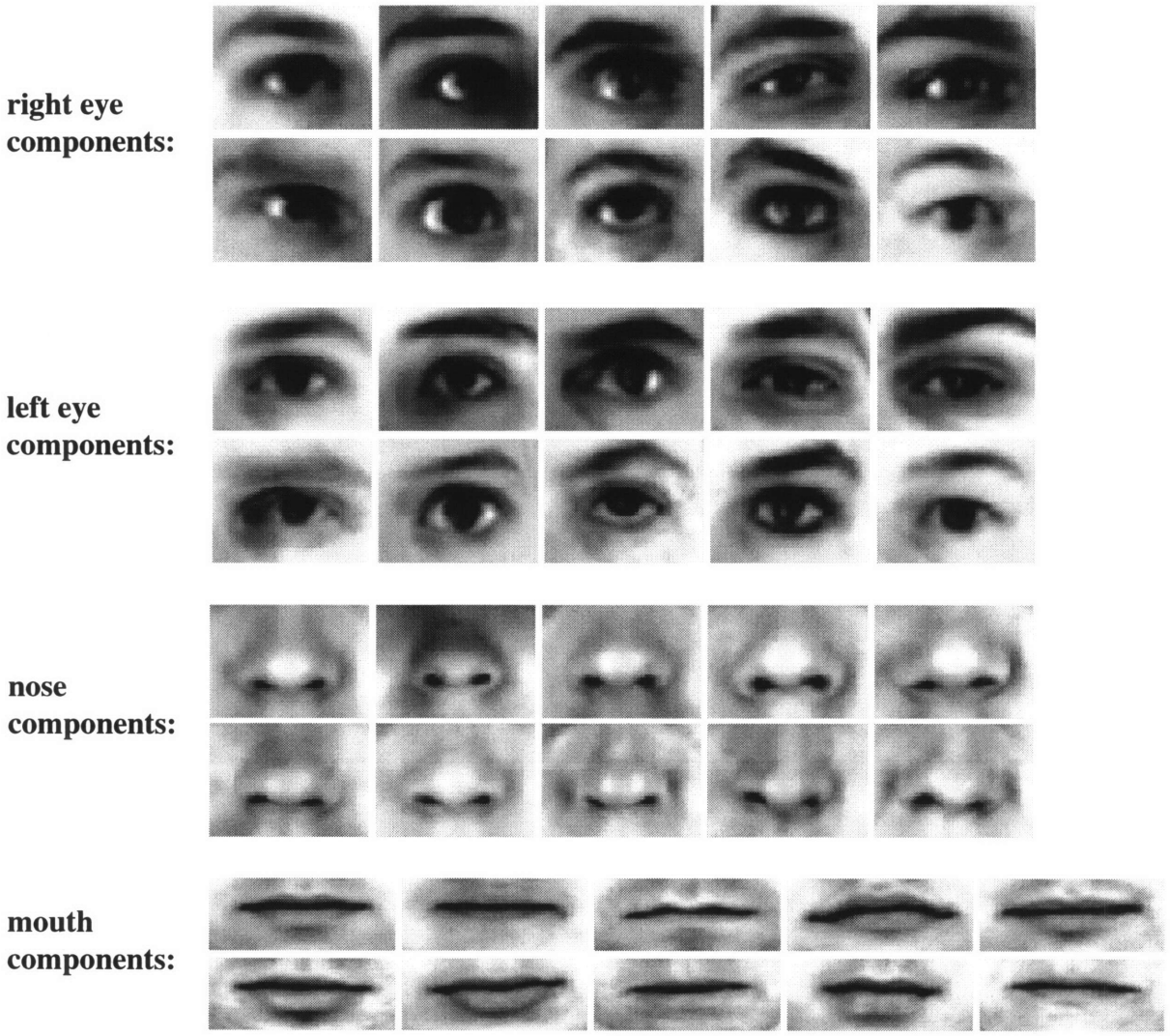


Figure 10.3: *The face prototypes were split up into 4 components. 50 prototypes were used for each component, 10 of which are shown here.*

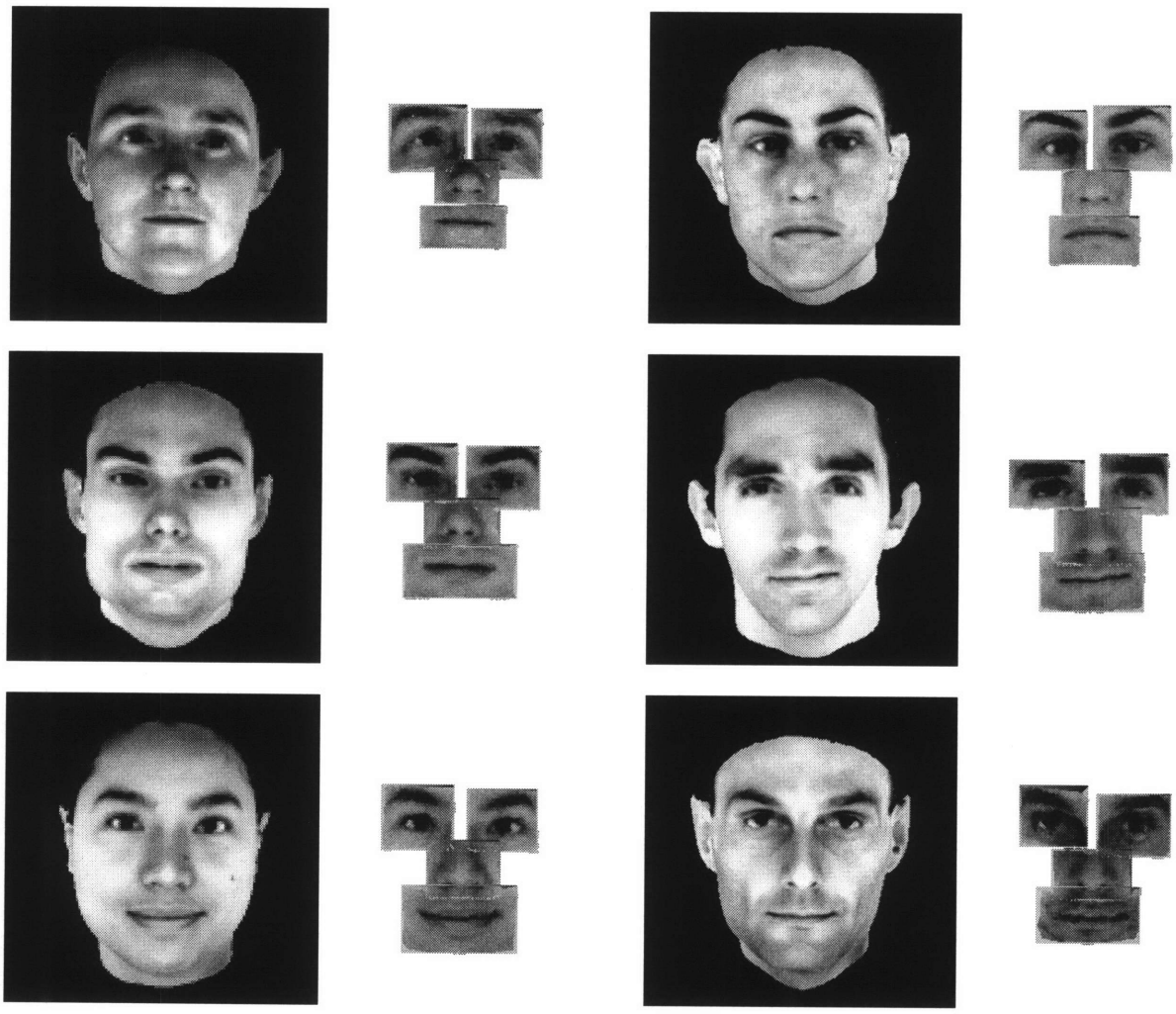


Figure 10.4: *Six example matches of the hierarchical face model to novel input images. The individual components are matched very well. Compare this figure to figures 5.4 and 5.5*

8 elements in the "position vector space". The position prototypes (P_i^k) have 2 elements $(\Delta x, \Delta y)$ for each of the four components. Therefore, 8 linearly independent position prototypes for each component would completely span the position vector space, effectively making the positions of the components unconstrained. We used the first 7 eigenvectors of the position vector space to constrain the component positions.

The stochastic gradient descent algorithm was run using just one pyramid level, 15 samples per iteration and 7000 iterations.

The results of matching the hierarchical morphable model to some novel input images are shown in figure 10.4. There are some edge artifacts around each component due to the warping algorithm used to render these images. However, by comparing these examples to the results on the standard morphable model shown in figures 5.4 and 5.5 we see that the hierarchical model finds better matches for each component.

This hierarchical morphable model was also tested on its ability to match occluded images. Figure 10.5 shows some examples of matching novel images which are occluded by solid rectangles. Figure 10.6 shows some examples of matching novel images which are occluded by cutting out pieces of other faces and pasting them over parts of the novel face. The matches are good for components which do not contain occlusion. The reconstruction, however, of components which are even partially occluded is poor. This is the main difference between the performance of the standard morphable model and the hierarchical morphable model on occluded images. The hierarchical model does a good job of matching unoccluded components even though the rest of the image may be heavily occluded. The quality of matches for the standard model degrades even in areas where there is no occlusion. On the other hand, the standard model does a better job of reconstructing areas where there is partial occlusion because it uses information from the entire image to find a match. The hierarchical model fits each component almost independently and so a partially occluded component often leaves little information to use for reconstruction of that component. This problem would of course be lessened if larger components were used so that partial occlusion

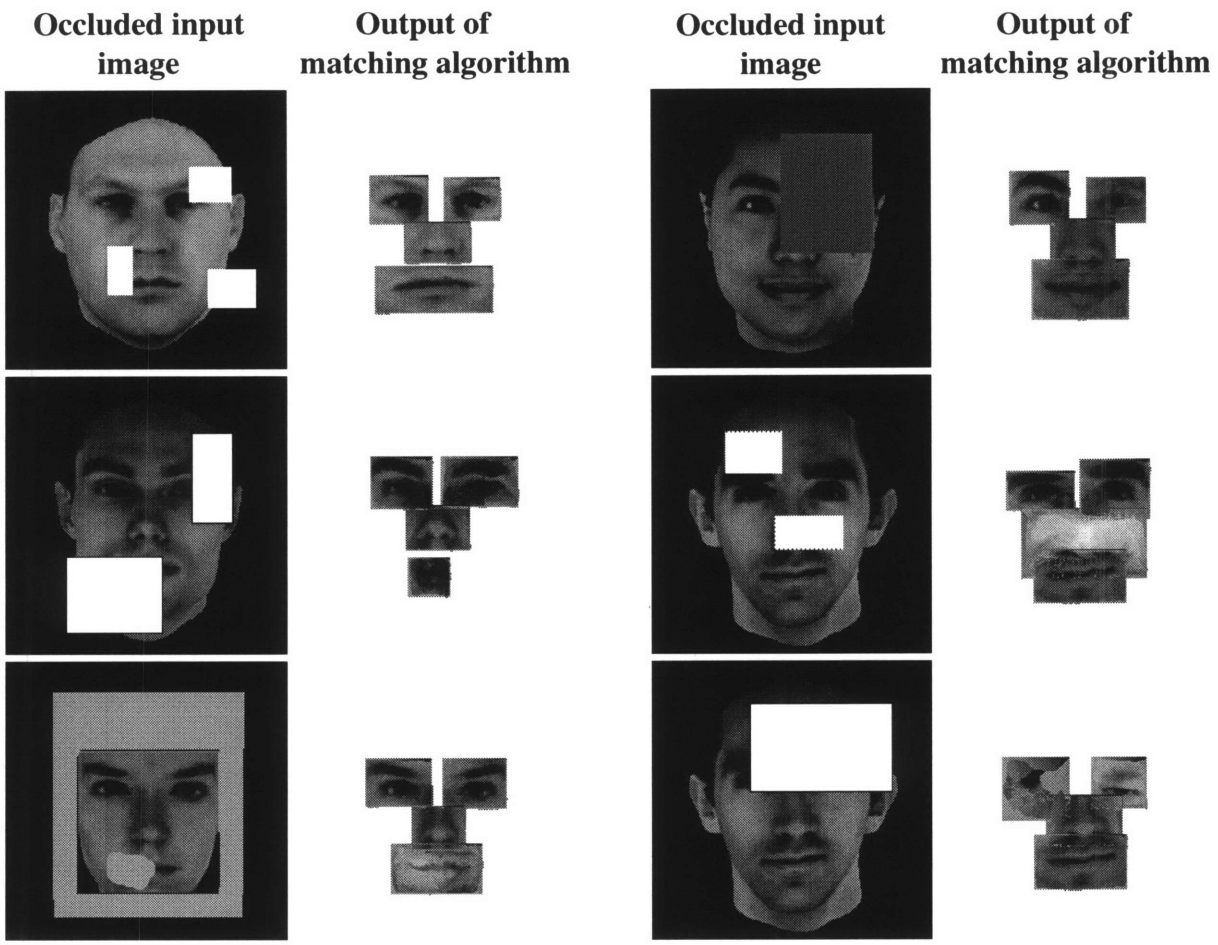


Figure 10.5: *Six examples of matching a partially occluded face using the hierarchical face model. Unoccluded components are matched well independently of occluded regions.*

might still leave plenty of unoccluded area.

The properties of the hierarchical morphable model for matching partially occluded images make it useful for object detection or verification tasks. Suppose the task is to detect a face (which may be occluded) in a cluttered scene. A hierarchical morphable model can be used to match the unoccluded parts of the face well. The well-matched components can then be used to decide if a face exists (using some heuristic such as "at least two components must have a good match"). Determining if a component is well matched can be done by simply examining the L_2 error for each component. The probability of the model parameters can also be used to determine a good match.

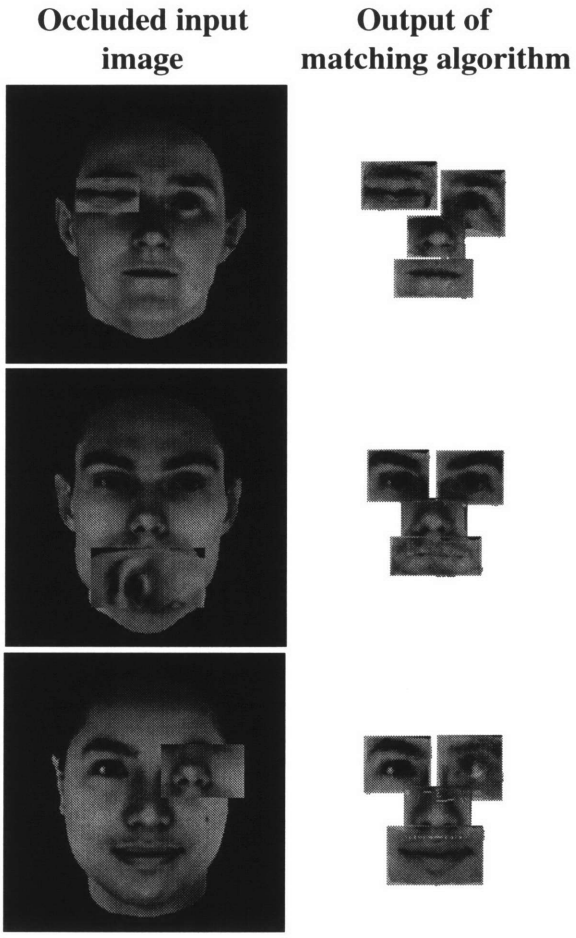


Figure 10.6: *Three more examples of matching a partially occluded face. Again the areas which are not occluded are matched very well.*

Chapter 11

Top-down approach to low-level vision

Perceptual tasks such as edge detection, image segmentation, lightness computation and estimation of 3D structure are considered to be low or mid-level vision problems and are traditionally approached in a bottom-up, generic, hard-wired way. An alternative approach is top-down, specific for object classes and example-based. In this chapter we discuss the use of multidimensional morphable models as a tool for a top-down approach to low-level vision, and we discuss its implications for human perception.

11.1 Ideal edge detection

Consider the specific problem of estimating a line drawing from a grey level image. The line drawing should ideally capture all the relevant "edges", in a way similar to an artist. As many years of work on edge detection have shown (see for a review [Marr and Hildreth, 1980, Haralick, 1980]), the problem is difficult, in part because physical edges – meant as discontinuities in 3D structure and albedo that convey information about the object's shape and identity – do not always generate intensity edges in the image. Conversely, intensity edges are often due to shading effects produced by illumination and, therefore, do not always reflect intrinsic properties of the object. Several years ago the Turing Institute in Glasgow

circulated a photograph of a face and asked fellow scientists to mark "edges" in the image. Some of the edges that were found by the subjects of these informal experiments did not correspond to any change in intensity in the picture; they corresponded to locations where the subjects knew that the 3D shape had a discontinuity, for instance the chin boundary. The traditional approach to edge detection – to use a general purpose edge detector such as a directional derivative followed by a nonlinear operation – is bound to fail in the task of producing a good line drawing, even if coupled with algorithms that attempt to fill edge gaps, using general principles such as good continuation, and collinearity [Shashua and Ullman, 1988]. A quite different approach is to exploit specific knowledge about faces in order to generate the line drawing. This approach runs contrary to the traditional wisdom in computer vision, since it assumes that object recognition may be used for edge detection – almost a complete subversion of the usual paradigm.

A possible implementation of this approach is based on casting it as a learning task. Given a set of sample prototypical (grey level) face images and the corresponding line drawings, drawn by an artist, the task is to learn the mapping that associates to a grey level image of a face its "ideal" line drawing. We discuss next how to use a morphable model to realize this general idea.

Suppose you have one morphable model built from grey level face images and a second morphable model built from the prototypical line drawings associated with each grey level face image. Then, given the image of a novel face, the approach is to estimate the parameters of the best fitting grey-level morphable model and to plug the same parameter values into the second morphable model built from the protoypical line drawings. This approach can be regarded as learning from a set of examples the mapping between a grey-level face image and its ideal line drawing. We have implemented an even simpler version of the scheme. We assume that the ideal line drawing corresponding to the average prototype is available from an artist, as shown in figure 11.1. We are using the same face database shown in figures 5.1 - 5.3. The matching of the morphable model obtained from the protoypes to a novel grey level image provides a shape vector which is a linear combination of the prototypes and

which effectively tells how to warp the average shape of the grey level prototype in order to match the shape of the novel grey level image. Because the line drawings are supported on a subset of the pixels of the corresponding grey-level images, the line drawings associated with novel images can be obtained by warping the line drawing associated with the reference prototype by using the estimated shape vector. Figure 11.2 shows a few examples of novel images (not contained in the set of our prototypical examples) and the line drawing estimated from each of them by our "ideal edge detector". To contrast this approach to a low-level gradient-based approach, figure 11.2 also shows the edges found for each face image by a Canny edge detector. Fig. 11.3 shows the ideal edge-map estimated for an input image with potentially many irrelevant edges and partial occlusion of the relevant ones. As is evident from the examples, our algorithm can detect and complete edges that do not correspond to any intensity gradients in the image and ignores those non-intrinsic to faces. The power of the algorithm derives from the high-level knowledge about faces, learned from the set of prototypical images.

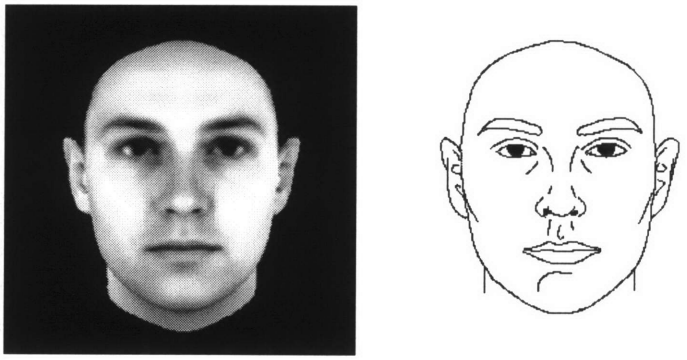


Figure 11.1: The reference face and its corresponding line drawing created by an artist.

11.2 Solving other visual tasks

Other supposedly mid-level visual tasks can be learned in a similar way. We briefly describe two of them: the generation of "virtual" views and the estimation of 3D structure from single

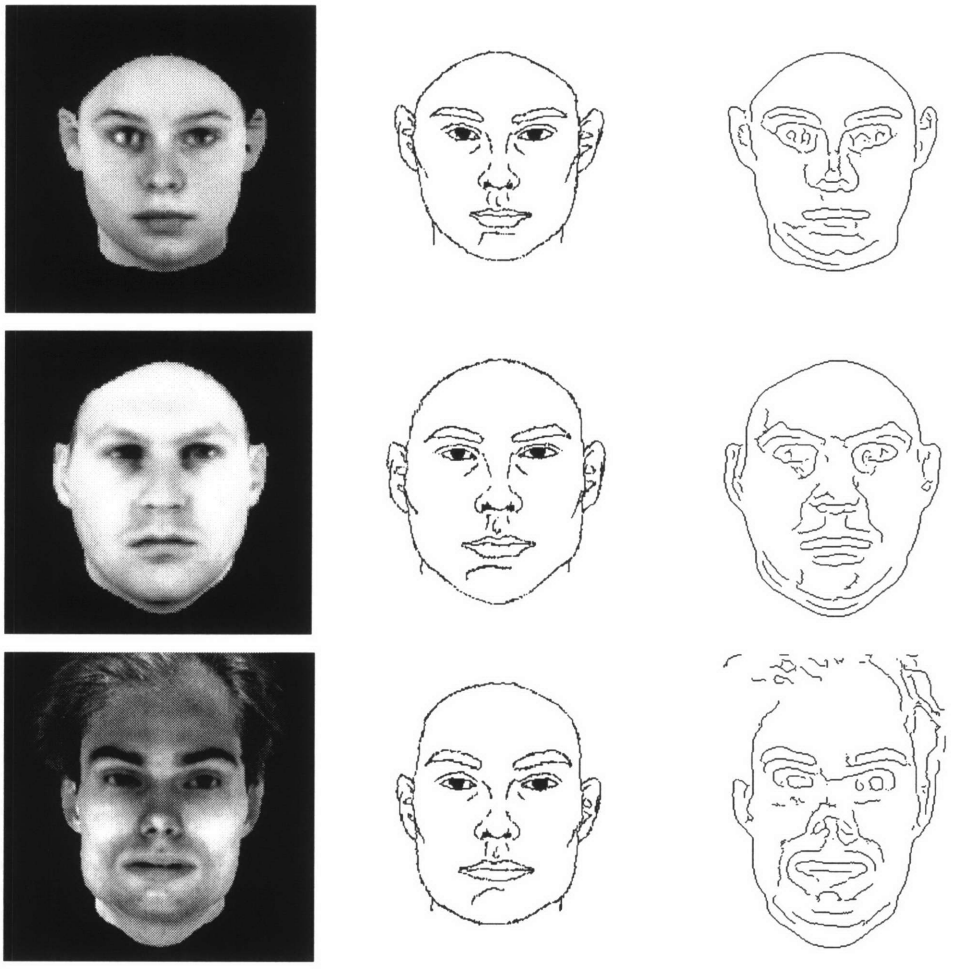


Figure 11.2: Examples of ideal edges found by the algorithm described in this chapter. The left column shows the input novel images. The middle column shows the line drawings estimated automatically by the algorithm which matches the morphable model to the novel images and then appropriately warps the ideal edges of the reference image. For comparison, the right column shows the edges found by a bottom-up edge detector (Canny, see [Canny, 1983]). Note that the ideal edges emphasize the perceptually significant features of the face much better than the Canny edges.

images.

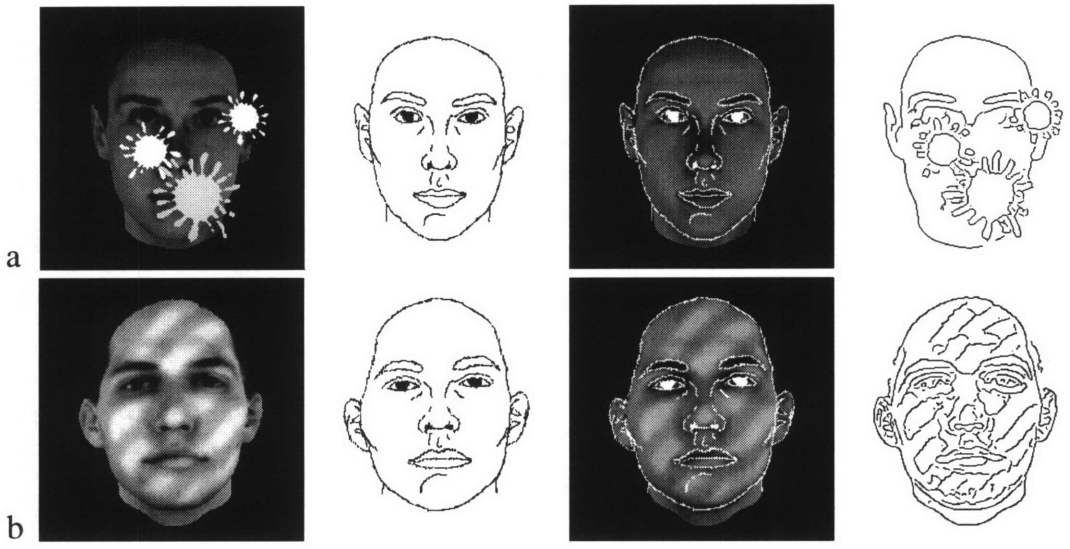


Figure 11.3: a) Example of ideal edges (second column) found for a partially occluded input face (left column). The image in the third column show these edges overlaid on the unoccluded input face. The fourth column shows the canny edge map for comparison. b) An example of an image with shadows (first column). The ideal edges found by our method are shown in column two and effectively ignore the edges produced by the shadows. The overlayed image in column three shows that these edges are accurate. Column four shows the canny edges.

Consider the case in which only one example image of an object is available instead of several example views. This may occur in object recognition tasks in which an object has to be recognized from a novel view. Vetter and Poggio ([Vetter and Poggio, 1996]) consider this problem for linear object classes. As described in chapter 1, an object belongs to a linear class if its 3D structure can be exactly described as a linear combination of the 3D structure of a small number of prototypes ([Poggio and Vetter, 1992]). A new *virtual* view of an object which belongs to a linear class can be generated exactly from a single example view, represented as a 2D shape vector, provided appropriate prototypical views of other objects in the same class are available (under orthographic projection). In this way, new views of a specific face with a different pose can be estimated and synthesized from a single view (the procedure is exact for linear classes; empirically, faces seem to be close to a linear class so

that the procedure above provides a good approximation for pose and expression). Again, this procedure can be formulated in terms of the learning metaphor in which a learning box is trained with input-output pairs of prototypical views representing each prototype in the initial and in the desired pose. Then for a new input image the system synthesizes a virtual view in the desired pose [Vetter and Poggio, 1996].

The estimation of 3D structure from a single image would proceed in a very similar way, if the image and the 3D structure of a sufficient number of prototypical objects of the same class are available ([Poggio and Vetter, 1992], [Vetter and Poggio, 1996]). In our learning box metaphor, the system, trained with pairs of prototype images as inputs (represented as 2D shape vectors) and their 3D shape as output, would effectively compute shape for novel images of the same class (compare with the somewhat different approach of [Atick *et al.*, 1995]). A similar approach may be extended to problems of color constancy and motion analysis, in which the desired information about color or motion is provided in a learning-from-examples scheme based on the use of a class-specific flexible model.

In summary, we have shown how morphable models can be used to learn to perform visual tasks in a top-down way, specific to object classes. From the point of view of a neuroscientist, our demonstrations are nothing more than plausibility proofs that the visual system can use simple learning processes to incorporate object specific knowledge and thereby learn to perform seemingly 'low-level' visual tasks in a top-down manner [Cavanagh, 1991, Mumford, 1992, Ullman, 1995]. We conjecture that perception in humans may rely on such processes to a greater extent than commonly assumed. Of course, biological vision may use bottom-up verification routines to validate the top-down "hallucination" (see [Ullman, 1995, Mumford, 1992]). A similar verification approach (top-down and bottom-up) could also be effectively used in machine vision implementations like the one described here.

Logically, our conjecture consists of two somewhat independent parts. The first one is that, at least in some cases, the visual system may solve low-level vision problems by exploiting prior information specific to the task and to the type of visual input.

The second part of the conjecture relates to how these specific algorithms may be synthesized by our visual system. The idea – which is the main point of this paper – is that visual systems may learn algorithms specific to a class of objects by associating in each "prototypical" example an "ideal" output to the input view. The "ideal" outputs may be available through other sensory modalities, sequences of images in time or even explicit instruction. The notion of what constitutes an "ideal" output corresponding to a certain class of inputs may change and evolve over time as the learning process encounters new examples. The second part of the conjecture predicts that human subjects should be able to learn to associate arbitrary outputs to input images of a certain class and to generalize from these learned associations. There is a weak and a strong form of the conjecture. The strong form is that the learning follows the morphable models algorithm we have used here in our plausibility demontration. The weak form of the conjecture leaves open the specific learning scheme. Preliminary psychophysical evidence favours the conjecture ([Sinha and Poggio, 1996]), with more experiments under way. Further work may enable us to verify whether the strong or weak form is to be preferred and in the latter case which learning scheme may be used by the visual system.

Chapter 12

Applications

There are a number of interesting applications to which the morphable model and matching algorithm can be applied. The focus of this thesis is to explain the underlying morphable model framework and to explain how it can be applied to a number of problems in computer vision. However, we leave the implementation of these applications for future work.

12.1 Example-based correspondence

The most basic application of multidimensional morphable models is to "vectorize" a novel image (Poggio and Beymer, 1996), providing dense correspondences between two images of a known class of objects. Once a novel image is approximated by the morphable model it is in correspondence with the reference image (assuming the match was good). To compute the pixelwise correspondences between two novel images, I_1^{novel} and I_2^{novel}, in the same object class, a morphable model for that class is first matched to each novel image. This gives the flow field from the reference image to I_1^{novel} and from the reference to I_2^{novel}. We want the flow field from I_1^{novel} to I_2^{novel}. To compute this, the flow field from the reference to I_1^{novel} must be reversed so that the flow field points from I_1^{novel} to the reference. Once this reversed field is computed then we can compute the flow field from I_1^{novel} to I_2^{novel} by simply combining the flow fields from I_1^{novel} to the reference and from the reference to I_2^{novel} (see

Appendix A).

By reversed field, we simply mean if the vector points from (x, y) in the reference image to (x', y') in I_1^{novel} then the reversed vector points from (x', y') in I_1^{novel} to (x, y) in the reference image. To reverse the flow field from the reference to I_1^{novel}, we must be careful. The flow field vectors which point to coordinates in I_1^{novel} will often point to non-whole number coordinates. So to compute the reversed flow field vectors from the (whole numbered) pixels of I_1^{novel} to the reference image we interpolate among the reversed vectors which fall closest to each whole numbered coordinate. This can be a tricky algorithm to implement and in practice, simply rounding the non-integral coordinates works well although with some loss in accuracy.

This pixelwise correspondence technique can deal with larger variations between two images than algorithms such as optical flow, provided that the images are of objects of a known class.

12.2 Image analysis

Custom image analysis can be implemented by mapping the linear coefficients of the model ($\mathbf{c}$ and $\mathbf{b}$ in equation 3.4) to higher level parameters such as pose or expression by using a RBF network or other learning network [Poggio *et al.*, 1993]. In other words, if some of the prototypes represent changes in pose (for example) for the object class, then the coefficients for these examples in the model can be mapped to a value representing the pose of the object. As another example, consider a set of prototypes for faces which includes faces with different expressions (smiling, frowning, angry, sad, etc), as in [Beymer *et al.*, 1993]. After matching a novel image of a face, the coefficients of the model can be used to determine if the input face is smiling, frowning, etc. We call this technique for analyzing images "analysis by synthesis" because model images are synthesized and then compared to the input image. Another possible application in the image analysis domain is lip reading. A model of lips can be built from examples. This model can then be used to track lips in a sequence of images

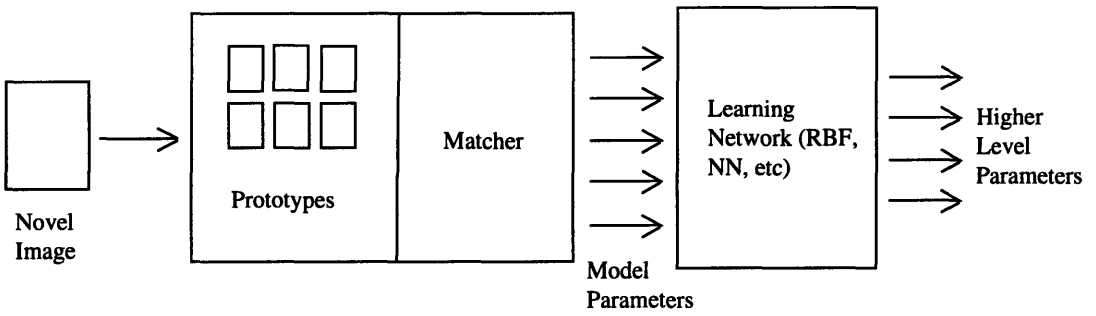


Figure 12.1: A general image analysis system. The novel image is first matched by a morphable model. The resulting model parameters become input to a learning network which has been trained to map model parameters onto higher level parameters (such as pose or expression), defined by the user.

(see also [Blake and Isard, 1994]). A mapping from model parameters to phonemes can be learned to read the lips.

Figure 12.1 illustrates a general system for image analysis which first matches a morphable model to a novel image and then maps the coefficients of the model to some higher level parameters by using a learning network such as a radial basis function.

12.3 Face recognition

A similar approach can be taken to implement a face recognition system. First a morphable model for the class of faces is built. For each person in the database, a number of different face images should be used in the model (to capture how the face can appear differently on different days). A principal components analysis of the shape and texture vectors is then done using all the prototypes. Then, the eigenvectors with significant eigenvalues are retained for the morphable model in both the shape and texture spaces. The eigenvector representation is used so that the shape vectors are orthogonal as well as the texture vectors. This eliminates the possibility of different sets of parameters yielding the same model image. Next, the original example images are matched using the eigenvector morphable model and the resulting model parameters ($\mathbf{c}$ and $\mathbf{b}$ in equation 3.4) for each person are used as input

to a standard classifier algorithm. The example images for each person form a class within the class of faces. This classifier is then used to recognize a novel face by first matching the eigenvector morphable model and using the resulting model parameters as input to the classifier.

In order to achieve invariance to lighting conditions and pose, examples images of the reference face under different illuminations and poses should be included in the example set.

12.4 Object verification

Morphable models can also be used for object verification. The verification task is to determine if a specific region of an image contains an instance of some object class or not. This entails matching the morphable model for that object class to the region in question and using the error of the final match and the likelihood of the resulting parameters (see section 4.6) to determine if the object exists there. A threshold can be set based on appropriate statistics to determine if the match is good enough for positive verification. In principal, this technique could also be used for object detection where the task is to find all instances of the object class in an image, by simply starting the matching algorithm at evenly spaced intervals in the image. However, with current hardware, this process is too slow for practical object detection applications.

12.5 Object tracking

A morphable model can be used to track a (non-rigid) object in a video sequence. Given a morphable model for the object class one is interested in tracking and a good starting point for the object's location in the initial frame of the video sequence, the matching algorithm is simply run on each frame using the parameters of the previous frame as the initial parameters. Since the object's position, shape and texture will only change slightly from frame to frame, the matching algorithm only needs to be run for a few iterations, making it fast enough to

track an object in real time.

12.6 Image compression

Another application for our morphable model framework is image compression. The idea is that novel images within a class of objects for which a model exists can be represented using only the parameters of the model. The number of bytes needed to store the model parameters is typically much smaller than the image itself. For example, consider the case of faces: given a novel face, it can be represented in terms of the morphable model described earlier. After matching, only the parameters of the model ($\mathbf{c}$, $\mathbf{p}$ and $\mathbf{b}$) need to be stored for a total of 106 bytes (assuming 50 shape and texture eigenvectors are used and only 1 byte of persision is needed). The novel face can then be synthesized from these parameters (assuming of course that the model is independently available).

12.7 Optical character recognition

As mentioned in section 2.6.3 a set of shape models (morphable models without texture vectors) can be used for optical character recognition. The idea is to first build a shape model from examples for each character that should be recognized. Then a novel character is recognized by simply matching the shape models for each character to the novel input and selecting the one with the lowest match error. The likelihood of the resulting parameters can also be used as a criterion for classifying the novel character.

Chapter 13

Conclusions

In this thesis we have described a flexible model to represent images of a class of objects and in particular we have described how to use it to analyze new images and represent them in terms of the model. Our morphable model does not need to be handcrafted but can be directly "learned" from a small set of images of prototypical objects. The key idea underlying the morphable model is a representation of images that relies on the computation of dense correspondence between images. In this representation, the set of images is endowed with the algebraic structure of a linear vector space. Our morphable model spans the space of the natural coordinates defined by the protoypes (or by an efficient linear combination of them as provided by the Karhunen-Loeve transformation).

The main contribution of this thesis is to solve the analysis problem: how to apply the morphable model for image analysis. Key to the analysis step is matching and a new matching algorithm is the main focus of this thesis. We have presented experiments which demonstrate the ability of various morphable models to match novel images for the same object class as well as experiments which demonstrate the algorithm's robustness and limits. We have also described how to learn a model from protoype images and compute the pixelwise correspondences among the prototypes automatically. Analysis coupled with synthesis offers a large number of applications of the morphable model, including recognition, image compression, correspondence and learning of visual tasks in a top-down way, specific

to object classes (e.g. estimation of contours, shape and color).

There are many directions for future research within the framework of morphable models. One obvious next step is to implement any of the applications suggested in chapter 12. Another direction for future work is to further develop the hierarchical morphable model described in chapter 10. Developing a method for automatically choosing components with each prototype would be a big step forward. Also, it would be interesting to test the hierarchical model on an object detection or verification task. Another interesting idea to pursue is combining the standard morphable model with a hierarchical model which provides more detail in important regions after first matching the whole object with the standard model. Finally, another direction for further research is to extend morphable models to three dimensional data. The prototypes could be 3D Cyberware scans (or some other 3D representation) and the matching algorithm would warp the 3D shapes to find good matches to a novel 3D shape. Also, it may be possible to match 3D shapes to 2D images by rendering a 2D image of the 3D model and comparing this to the 2D input image.

In conclusion, multidimensional morphable models provide a rich framework for approaching many important problems in computer vision, and also provide a fertile ground for future research.

Appendix A

Combining flow fields

In the bootstrapping algorithm we need to combine two flow fields which describe the pixel-wise correspondences from image I_1 to I_2 and from I_2 to I_3. Let $dx_1(x,y)$ be the x displacement part of the flow field from I_1 to I_2 and let $dy_1(x,y)$ be the y displacement part. In other words, for any whole number pixel (x,y) in image I_1, $dx_1(x,y)$ gives the displacement of that pixel in the x direction in image I_2. Likewise in the y direction for $dy_1(x,y)$. Similary, let $dx_2(x,y)$ and $dy_2(x,y)$ make up the flow field from I_2 to I_3.

The problem is to combine these flow fields to get the flow field from I_1 to I_3. We will call this new combined flow field, $dx_3(x,y)$ and $dy_3(x,y)$.

To do this we use bilinear interpolation. For every whole numbered pixel (x,y) in I_1, the coordinate $(x + dx_1(x,y), y + dy_1(x,y))$ is the corresponding point in I_2. However, $(x + dx_1(x,y), y + dy_1(x,y))$ is not necessarily whole numbered. Thus we cannot use it to index dx_2 and dy_2 directly. Instead we must bilinearly interpolate to get the value of $dx_3(x,y)$ and $dy_3(x,y)$.

$$\begin{aligned} dx_3(x,y) \;=\; & (1-\delta x)(1-\delta y)\, dx_2(\lfloor x + dx_1(x,y)\rfloor, \lfloor y + dy_1(x,y)\rfloor) + \\ & \delta x(1-\delta y)\, dx_2(\lfloor x + dx_1(x,y)\rfloor + 1, \lfloor y + dy_1(x,y)\rfloor) + \\ & (1-\delta x)\delta y\, dx_2(\lfloor x + dx_1(x,y)\rfloor, \lfloor y + dy_1(x,y)\rfloor + 1) + \end{aligned}$$

$$\delta x \delta y \; dx_2(\lfloor x + dx_1(x,y) \rfloor + 1, \lfloor y + dy_1(x,y) \rfloor + 1) \tag{A.1}$$

where $\delta x = x + dx_1(x,y) - \lfloor x + dx_1(x,y) \rfloor$ and $\delta y = y + dy_1(x,y) - \lfloor y + dy_1(x,y) \rfloor$.

Similarly for $dy_3(x,y)$:

$$\begin{aligned} dy_3(x,y) \;=\; & (1-\delta x)(1-\delta y)\; dy_2(\lfloor x + dx_1(x,y) \rfloor, \lfloor y + dy_1(x,y) \rfloor) + \\ & \delta x(1-\delta y)\; dy_2(\lfloor x + dx_1(x,y) \rfloor + 1, \lfloor y + dy_1(x,y) \rfloor) + \\ & (1-\delta x)\delta y\; dy_2(\lfloor x + dx_1(x,y) \rfloor, \lfloor y + dy_1(x,y) \rfloor + 1) + \\ & \delta x \delta y\; dy_2(\lfloor x + dx_1(x,y) \rfloor + 1, \lfloor y + dy_1(x,y) \rfloor + 1). \end{aligned} \tag{A.2}$$

Appendix B

Image warping

To render a model image given a shape vector (flow field) and a texture vector, we use a very simple forward warping algorithm. For more details on warping see [Wolberg, 1990].

B.1 Forward warping

Let the flow field be represented by $dx(x,y)$ and $dy(x,y)$, the x and y components of the flow field, respectively. Let $T(x,y)$ be the texture vector (which is simply a grey scale image). The flow field describes the displacement for each pixel, (x,y), in the image T. To obtain the forward warping of T by the flow field, dx and dy we simply move each pixel (x,y) to the new location $(x+dx(x,y), y+dy(x,y))$. This process is complicated by two problems. The first problem is that $(x+dx(x,y), y+dy(x,y))$ is not necessarily a whole number valued coordinate. The second problem is that the resulting warped image, call it $I(x,y)$, may have holes - that is some pixels in I may not be mapped to.

There are various ways to resolve the non-whole number problem. We choose the simplest and fastest since the results are satisfactory for our purposes. We simply round to the nearest whole numbered pixel. Thus,

$$I(round(x+dx(x,y)), round(y+dy(x,y))) = T(x,y)$$

where the function, round(x), simply rounds x to the nearest integer.

The problem of holes can also be solved in many ways. Again, we choose a simple and fast technique to solve this problem. Our solution is for each hole (which can be detected by simply keeping track of which pixels are not mapped to after forward warping) simply look at neighboring pixels by following an ever-widening spiral trajectory around the hole until a non-hole pixel is found (this is usually very close to the hole). Then the hole is simply assigned the grey level value of it's nearest non-hole neighbor.

B.2 Backward warping

Given a flow field $(dx(x,y), dy(x,y))$ from the reference image to a prototype image I_1, the backward warping algorithm attempts to move the pixels of I_1 to their corresponding locations in the reference image (as determined by the flow field). The only difficulty is that the flow field may not point to integer locations in I_1. To solve this we use bilinear interpolation.

So,

$$\begin{aligned} I^{bckwd-warp}(x,y) \;=\; & (1-\delta x)(1-\delta y)\, I_1(\lfloor x+dx(x,y)\rfloor, \lfloor y+dy(x,y)\rfloor) + \\ & \delta x(1-\delta y)\, I_1(\lfloor x+dx(x,y)\rfloor + 1, \lfloor y+dy(x,y)\rfloor) + \\ & (1-\delta x)\delta y\, I_1(\lfloor x+dx(x,y)\rfloor, \lfloor y+dy(x,y)\rfloor + 1) + \\ & \delta x\delta y\, I_1(\lfloor x+dx(x,y)\rfloor + 1, \lfloor y+dy(x,y)\rfloor + 1) \end{aligned} \tag{B.1}$$

where $\delta x = x + dx(x,y) - \lfloor x + dx(x,y)\rfloor$ and $\delta y = y + dy(x,y) - \lfloor y + dy(x,y)\rfloor$.

Bibliography

[Atick *et al.*, 1995] Joseph J. Atick, Paul A. Griffin, and A. Norman Redlich. Statistical approach to shape from shading: Reconstruction of 3d face surfaces from single 2d images. *submitted to Neural Computation*, 1995.

[Bell and Sejnowski, 1995] Anthony J. Bell and Terrence J. Sejnowski. An information-maximisation approach to blind separation and blind deconvolution. Institute for Neural Computation Technical Report INC-9501, UCSD, 1995.

[Bergen and Hingorani, 1990] J.R. Bergen and R. Hingorani. Hierarchical motion-based frame rate conversion. Technical report, David Sarnoff Research Center, April 1990.

[Beymer and Poggio, 1995] David Beymer and Tomaso Poggio. Face recognition from one example view. A.I. Memo 1536, MIT, 1995.

[Beymer and Poggio, 1996] David Beymer and Tomaso Poggio. Image representations for visual learning. *Science*, 272:1905–1909, June 1996.

[Beymer *et al.*, 1993] D. Beymer, A. Shashua, and T. Poggio. Example based image analysis and synthesis. A.I. Memo 1431, MIT, 1993.

[Beymer, 1996] David Beymer. *Pose-Invariant Face Recognition Using Real and Virtual Views.* PhD thesis, Massachussetts Institute of Technology, 1996.

[Biederman, 1985] I. Biederman. Human image understanding. *Computer Graphics, Vision and Image Processing*, 32:29–73, 1985.

[Blake and Isard, 1994] Andrew Blake and Michael Isard. 3d position, attitude and shape input using video tracking of hands and lips. *Computer Graphics Proceedings*, pages 185–192, 1994.

[Bulthoff *et al.*, 1995] H. H. Bulthoff, S. Y. Edelman, and M. J. Tarr. How are three-dimensional objects represented in the brain? *Cerebral Cortex*, 5(3):247–260, 1995.

[Burt and Adelson, 1983] Peter J. Burt and Edward J. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, COM-31(4):532–540, 1983.

[Burt, 1984] Peter J. Burt. The pyramid as a structure for efficient computation. In *Multi-Resolution Image Processing and Analysis*, pages 6–37. Springer-Verlag, 1984.

[Canny, 1983] J. Canny. Finding edges and lines in images. MIT A.I. Technical Report 720, MIT, 1983.

[Cavanagh, 1991] P. Cavanagh. What's up in top-down processing? In A. Gorea, editor, *Representations of Vision.* Cambridge University Press, 1991.

[Choi *et al.*, 1991] Chang Seok Choi, Toru Okazaki, Hiroshi Harashima, and Tsuyoshi Takebe. A system of analyzing and synthesizing facial images. *IEEE*, pages 2665–2668, 1991.

[Cootes and Taylor, 1992] T.F. Cootes and C.J. Taylor. Active shape models - 'smart snakes'. *British Machine Vision Conference*, pages 266–275, 1992.

[Cootes and Taylor, 1994] T.F. Cootes and C.J. Taylor. Using grey-level models to improve active shape model search. *International Conference on Pattern Recognition*, pages 63–67, 1994.

[Cootes *et al.*, 1992] T.F. Cootes, C.J. Taylor, D.H. Cooper, and J. Graham. Training models of shape from sets of examples. *British Machine Vision Conference*, pages 9–18, 1992.

[Cootes *et al.*, 1993] T.F. Cootes, C.J. Taylor, A. Lanitis, D.H. Cooper, and J. Graham. Building and using flexible models incorporating grey-level information. In *ICCV*, pages 242–246, Berlin, May 1993.

[Cootes *et al.*, 1994] T.F. Cootes, C.J. Taylor, and A. Lanitis. Multi-resolution search with active shape models. *International Conference on Pattern Recognition*, pages 610–612, 1994.

[Edelman and Bulthoff, 1990] Shimon Edelman and Heinrich Bulthoff. Viewpoint-specific representations in three dimensional object recognition. A.I. Memo 1239, MIT, 1990.

[Grimson, 1990] W.E.L Grimson. *Object Recognition by Computer: The role of geometric constraints.* MIT Press, Cambridge, MA, 1990.

[Hallinan, 1995] Peter Winthrop Hallinan. *A Deformable Model for the Recognition of Human Faces Under Arbitrary Illumination.* PhD thesis, Harvard University, 1995.

[Haralick, 1980] R. M. Haralick. Edge and region analysis for digital image data. *Computer Graphics and Image Processing*, 12:60–73, 1980.

[Hill *et al.*, 1992] A. Hill, T.F. Cootes, and C.J. Taylor. A generic system for image interpretation using flexible templates. *British Machine Vision Conference*, pages 276–285, 1992.

[Jacobs, 1992] David Jacobs. *Recognizing 3-D Objects Using 2-D Images.* PhD thesis, Massachussetts Institute of Technology, 1992.

[Jones and Poggio, 1995] Michael Jones and Tomaso Poggio. Model-based matching of line drawings by linear combinations of prototypes. In *Proceedings of the Fifth International Conference on Computer Vision*, pages 531–536, 1995.

[Jones and Poggio, 1996] Michael Jones and Tomaso Poggio. Model-based matching by linear combinations of prototypes. A.I. Memo 1583, MIT, 1996.

[Kass *et al.*, 1988] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International Journal of Computer Vision*, pages 321–331, 1988.

[Kirby and Sirovich, 1990] M. Kirby and L. Sirovich. The application of the karhunen-loeve procedure for the characterization of human faces. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 12(1):103–108, January 1990.

[Lanitis *et al.*, 1995] A. Lanitis, C.J. Taylor, and T.F. Cootes. A unified approach to coding and interpreting face images. In *ICCV*, pages 368–373, Cambridge, MA, June 1995.

[Lines, 1996] Steve Lines. The photorealistic synthesis of novel views from example images. Master's thesis, MIT, 1996.

[Logothetis *et al.*, 1995] N. Logothetis, J. Pauls, and T. Poggio. Shape Representation in the Inferior Temporal Cortex of Monkeys. *Current Biology*, 5(5):552–563, 1995.

[Marr and Hildreth, 1980] David Marr and Ellen Hildreth. Theory of edge detection. *Proc. of the Royal Society of London B*, 207:187–217, 1980.

[Mumford, 1992] D. Mumford. On the computational architecture of the neocortex. *Biological Cybernetics*, 66:241–251, 1992.

[Nastar *et al.*, 1996] Chahab Nastar, Baback Moghaddam, and Alex Pentland. Generalized image matching: Statistical learning of physically-based deformations. In *ECCV*, page to appear, Cambridge, UK, April 1996.

[Pauls *et al.*, 1996] J. Pauls, E. Bricolo, and N.K. Logothetis. Physiological evidence for viewer centered representation in the monkey. In S. Nayar and T. Poggio, editors, *Early Visual Learning.* Oxford University Press, 1996.

[Poggio and Beymer, 1996] Tomaso Poggio and David Beymer. Learning to see. *IEEE Spectrum*, pages 60–69, 1996.

[Poggio and Vetter, 1992] Tomaso Poggio and Thomas Vetter. Recognition and structure from one 2d model view: Observations on prototypes, object classes and symmetries. A.I. Memo 1347, MIT, 1992.

[Poggio *et al.*, 1993] Tomaso Poggio, Federico Girosi, and Michael Jones. From regularization to radial, tensor and additive splines. In ?, editor, *Proceedings of 1993 International Joint Conference on Neural Networks*, pages 223–227, ?, 1993. ?

[Poggio, 1990] Tomaso Poggio. A theory of how the brain might work. A.I. Memo 1253, MIT, 1990.

[Robbins and Munroe, 1951] H. Robbins and S. Munroe. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[Roberts, 1966] L. Roberts. Machine perception of three-dimensional solid objects. *Optical and Electro-optical Information Processing*, 1966.

[Shashua and Ullman, 1988] Amnon Shashua and Shimon Ullman. Structural saliency. In *Proceedings of the International Conference on Computer Vision*, pages 482–488, 1988.

[Shashua, 1992] Amnon Shashua. Projective structure from two uncalibrated images: Structure from motion and recognition. A.I. Memo 1363, MIT, 1992.

[Sinha and Poggio, 1996] Pawan Sinha and Tomaso Poggio. The role of learning in 3d form perception. *Nature (in press)*, 1996.

[Sinha, 1995] P. Sinha. *Perceiving and recognizing 3D forms.* PhD thesis, Massachussetts Institute of Technology, 1995.

[Troje and Bülthoff, 1995] N. Troje and H.H. Bülthoff. Face recognition under varying pose: The role of texture and shape. *Vision Research*, 36(12):1761–1771, 1995.

[Turk and Pentland, 1991] M.A. Turk and A.P Pentland. Face recognition using eigenfaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 586–591, 1991.

[Ullman and Basri, 1991] S. Ullman and R. Basri. Recognition by linear combinations of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:992–1006, 1991.

[Ullman, 1995] Shimon Ullman. Sequence seeking and counter streams: a model for bidirectional information flow in the visual cortex. *Cerebral Cortex*, 5:1–11, 1995.

[Vetter and Poggio, 1995] Thomas Vetter and Tomaso Poggio. Linear object classes and image synthesis from a single example image. A.I. Memo 1531, MIT, 1995.

[Vetter and Poggio, 1996] Thomas Vetter and Tomaso Poggio. Image synthesis from a single example view. *Proceedings of the European Conference on Computer Vision*, 1996.

[Vetter *et al.*, 1997] Thomas Vetter, Michael J. Jones, and Tomaso Poggio. A bootstrapping algorithm for learning linear models of object classes. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1997.

[Viola, 1995] Paul Viola. Alignment by maximization of mutual information. MIT A.I. Technical Report 1548, MIT, 1995.

[Wolberg, 1990] George Wolberg. *Digital Image Warping.* IEEE Computer Society Press, Los Alamitos, CA, 1990.

[Yuille *et al.*, 1992] Alan L. Yuille, Peter W. Hallinan, and David S. Cohen. Feature extraction from faces using deformable templates. *International Journal of Computer Vision*, 8(2):99–111, 1992.