MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL AND COMPUTATIONAL LEARNING

A.I. Memo No. 1583 November 1996
C.B.C.L. Paper No. 139

Model-Based Matching by Linear Combinations of Prototypes

Michael J. Jones and Tomaso Poggio

Abstract

We describe a flexible model similar to (Vetter and Poggio, 1993, 1995 and Jones and Poggio, 1995) for representing images of objects of a certain class, known a priori, such as faces, and introduce a new algorithm for matching it to a novel image and thereby perform *image analysis*. The flexible 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 the *shape* and *texture* vectors corresponding to each prototype. The flexible model is then synthesized as a linear combination that spans the linear vector space determined by the prototypical shapes and textures. In this paper we 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 flexible model. Our approach can provide novel solutions to several vision tasks, including the computation of image correspondence, object verification, image synthesis and image compression.

This paper 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 N0014-95-1-0600. Additional support is provided by ATR Audio and Visual Perception Research Laboratories, Sumitomo Metal Industries, Kodak, Daimler-Benz and Siemens AG. Tomaso Poggio is supported by the Uncas and Helen Whitaker Chair at MIT's Whitaker College.

1 Introduction

An important problem in computer vision is to model classes of objects in order to perform *image analysis* by matching the models to views of new objects of the same class, thereby parametrizing the novel image in terms of a known model. Many approaches have been proposed. Several of them represent objects using 3D models, represented in different ways (for a review, see [Besl and Jain, 1985]). Such models are typically quite sophisticated, difficult to build and hard to use for many applications – image matching in particular. A rather different approach is suggested by recent results in the science of biological vision.

There is now convincing psychophysical and even physiological evidence suggesting that the human visual system often uses strategies that have the flavor 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]). With this motivation, we have explored an approach in which object models are learned from several prototypical 2D images.

Though the idea of synthesizing a model for a class of objects, such as faces or cars, is quite attractive, it is far from clear how the model should be represented and how it can be matched to novel images. In other papers we have introduced a *flexible* model of an object class as a linear combination of example images, represented appropriately. Image representation is the key issue here because in order for a linear combination of images to make sense, they must be represented in terms of elements of a vector space. In particular, adding two image representations together must yield another element of the space. It turns out that a possible solution is to set the example images in pixelwise correspondence. In our approach, the correspondences between a reference image and the other example images are obtained in a preprocessing phase. Once the correspondences are computed, an image can be represented as a shape vector and a texture vector. The shape vector specifies how the 2D shape of the example differs from the reference image. Analogously, the texture vector specifies how the texture differs from the reference texture (here we are using the term "texture" to mean simply the pixel intensities of the image). The flexible model for an object class is thus a linear combination of the example shapes and textures which, for given values of the parameters, can be rendered into an image. The flexible model is thus a generative model which can *synthesize* new images of objects of the same linear class. But how can we use it to *analyze* novel images? In this paper we provide an answer in terms of a novel algorithm for matching the flexible model to a novel image (for a preliminary and partial version see [Jones and Poggio, 1995]).

The paper is organized as follows. First, we discuss related work. In section 3 we explain in detail our model. Section 4 describes the matching algorithm. Section 5 shows example applications and presents experiments on the robustness of the matching algorithm. Section 6 emphasizes the importance of pixelwise correspondences between prototype images and contrasts our method with other correspondence-free approaches. Next, section 7 discusses a number of applications. Extensions and future work are considered in section 8. Section 9 concludes with a summary and discussion.

2 Related Work

A preliminary version of this paper is [Jones and Poggio, 1995] which only presented applications to line drawings. In this paper, we present the full approach and its use for gray-value images.

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 [Beymer and Poggio, 1996]) is the main motivation for our 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, 1992b] 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 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. Jones and Poggio ([Jones and Poggio, 1995]) sketched a novel approach to match linear models to novel images that can be used for several visual *analysis* tasks, including recognition. In this paper we develop the approach in detail and show its performance not only on line drawings but also on gray-level images.

Among papers directly related to ours, [Beymer, 1995] addressed the problem of modeling human faces in order to generate "virtual" faces to be used in a face recognition system. A virtual view is a synthetically generated image of a face with a novel pose or expression. Beymer modeled texture the same way we do - as a linear combination of prototypical textures. Shape, however, was not constrained to be a linear combination of prototypical shapes and was instead estimated with an optical flow algorithm [Bergen and Hingorani, 1990] which was used to map each novel image to the reference face of the model. Ezzat ([Ezzat, 1996]) uses the same image representation and linear combinations of vectorized images that we use in order to build a model for synthesis and analysis of novel views of a *specific* face. [Vetter and Poggio, 1995] combined a linear model of shape and texture vectors to generate virtual views across large viewpoint changes.

Recently we have become aware of several papers dealing with various forms of the idea of linear combination of prototypical images. Choi *et. al.* (1991) were perhaps the first (see also [Poggio and Brunelli, 1992]) to suggest a model which represented face images with separate shape and texture components, using a 3D model to provide correspondences between example face images. In his study of illumination invariant recognition techniques, Hallinan ([Hallinan, 1995]) describes deformable models of a similar general flavor. The work of 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]) on active shape models is probably the closest to ours. It is based on the idea of linear combinations of prototypes to model non-rigid transformations within classes of objects. They use a very sparse set of corresponding points in their model (we use dense pixelwise correspondences). Their models only include texture as a component that is added on after the shape part of the model, which in their case is a contour, is fit to a novel image. Our image representation, relying on shape and texture vectors obtained through pixelwise correspondence, seems to be a significant extension which allows us to incorporate texture as well as shape seamlessly into a single framework.

Work which is superficially similar but is based on different representations includes [Atick *et al.*, 1995] who use 3D data taken from a Cyberware scanner and their linear combinations (without correspondence) to build a model. Turk and Pentland's [Turk and Pentland, 1991] and also Nayar and Murase's [Murase and Nayar, 1995] use of eigenimages – and therefore of a model which is a linear combination of example images – is quite different from ours since the basic representation is very different. The work of Viola [Viola, 1995] on alignment of a model with a novel image by maximization of mutual information suggested the stochastic gradient descent algorithm we use in this paper for the matching stage. Many other flexible models have been proposed. We mention here the model of Blake and Issard [Blake and Isard, 1994], which is similar in spirit to ours. Finally, a very recent paper by Nastar, Moghaddam and Pentland [Nastar *et al.*, 1996], which was published after the work described in this paper was completed, has intriguing similarities with our work, since it is based on the computation of correspondence between images, albeit with an algorithm different from the optical flow algorithm we use.

3 Modeling Classes of Objects

The work of Ullman and Basri [Ullman and Basri, 1991] and Poggio and Vetter [Poggio and Vetter, 1992] was based on a representation of images as vectors of the x, y positions of a small number of labeled features - and left open the question of how to find them reliably and accurately. Starting with [Beymer *et al.*, 1993], we have attempted to avoid the computation of sparse features while keeping as much as possible the representation of Ullman and Basri which has – at least under some conditions – the algebraic structure of a vector space. For images considered as bitmaps, on the other hand, basic vector space operations like addition and linear combination are not meaningful. We have argued therefore that a better way to represent images is to associate with each image a shape vector and a texture vector (see for a review [Poggio and Beymer, 1996]).

The *shape vector* of an example image associates to each pixel in the reference image the coordinates of the corresponding point in the example image. The *texture vector* contains for each pixel in the reference image the color or gray level value for the corresponding pixel in the example image. We refer to the operations which associate the shape and texture vectors to an image as *vectorizing the image.* Instead of two separate vectors we can also consider the full texture-shape vector which has dimensionality $N + 2N$ where N is the number of pixels in the image. 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 in the paper 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.

3.1 Formal specification of the model

In this section we will formally specify our model which we refer to as a *linear object class 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 gray level values. For eight bit images, $a = 255$. Here we are only considering gray level images, although color images could also be handled in a straightforward manner. To define a model, a set of example images called prototypes are given. We denote these prototypes as $I_0, I_1, \ldots, I_N$. A naive approach might model this class of images using a linear combination of the images ([Turk and Pentland, 1991]) as follows:

$$I^{model} = \sum_{i=0}^{N} b_i I_i. \tag{1}$$

This approach does not result in good matches as shown in figure 1. The underlying reason for this method's poor performance is that the example images are not in correspondence. Our image representation is instead based on pixelwise correspondences.

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* and interchangeably as the shape vector for the vectorized I_j. We define

$$\hat{I}_j(x, y) = I_j \circ S_j(x, y) = I_j(S_j(x, y)). \tag{2}$$

$\hat{I}_j$ is the warping of image I_j onto the reference image I_0. In other words, $\{\hat{I}_j\}$ 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

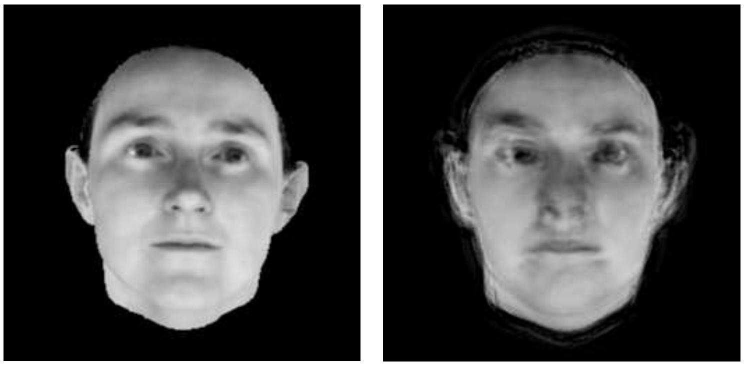

Figure 1: *Example of a novel image matched using a linear combination of raw example images. The example images consisted of 100 faces, some of which are shown in figure 2. The best match is blurry using this model because the example images are aligned and scaled but are not in pixelwise correspondence.*

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

$$I^{model} \circ (\sum_{i=0}^{N} c_i S_i) = \sum_{j=0}^{N} b_j \hat{I}_j. \tag{3}$$

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 $\sum_{j=0}^{N} b_j \hat{I}_j$ 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.

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

$$I^{model} \circ (A \circ \sum_{i=0}^{N} c_i S_i) = \sum_{j=0}^{N} b_j \hat{I}_j \tag{4}$$

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

$$A(x, y) = (p_0 x + p_1 y + p_2,\ p_3 x + p_4 y + p_5). \tag{5}$$

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 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.

When used as a generative model, for given values of **c**, **b** and **p**, the model image is rendered by computing I^{model} in equation 4. 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.

4 Matching the Model

The analysis problem is the problem of matching the flexible 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 **c** and **b** and the affine parameters **p**. Following this strategy, we define the sum of squared differences error

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel}(x, y) - I^{model}(x, y)]^2 \tag{6}$$

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. Equation 4 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 **c** and **p**) to I^{novel} and compare it to the shape-free model, that is

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel} \circ (A \circ \sum_{i=0}^{N} c_i S_i) - \sum_{j=0}^{N} b_j \hat{I}_j(x, y)]^2. \tag{7}$$

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.

The summation in equation 7 is over all pixels in the model image. 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. 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.

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

$$
\begin{aligned}
\frac{\partial E}{\partial c_i} &= \sum_{x,y}\left[[I^{novel} \circ \overline{S}(x,y) - \sum_{j=0}^{N} b_j \hat{I}_j(x,y)] \;\; \frac{\partial I^{novel} \circ \overline{S}(x,y)}{\partial c_i} \right] \\
\frac{\partial E}{\partial b_j} &= -\sum_{x,y}\left[[I^{novel} \circ \overline{S}(x,y) - \sum_{j=0}^{N} b_j \hat{I}_j(x,y)] \;\; \hat{I}_j(x,y) \right] \\
\frac{\partial E}{\partial p_i} &= \sum_{x,y}\left[[I^{novel} \circ \overline{S}(x,y) - \sum_{j=0}^{N} b_j \hat{I}_j(x,y)] \;\; \frac{\partial I^{novel} \circ \overline{S}(x,y)}{\partial p_i} \right] \\
\frac{\partial I^{novel} \circ \overline{S}(x,y)}{\partial c_i} &= \left.\frac{\partial I^{novel}(x,y)}{\partial (x,y)}\right|_{\overline{S}(x,y)} \left.\frac{\partial A(x,y)}{\partial (x,y)}\right|_{\sum_{i=0}^{N} c_i S_i(x,y)} \frac{\partial \sum_{i=0}^{N} c_i S_i(x,y)}{\partial c_i}
\end{aligned}
$$

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

$$
\begin{aligned}
\left.\frac{\partial A(x,y)}{\partial (x,y)}\right|_{\sum_{i=0}^{N} c_i S_i(x,y)} &= \left.\begin{bmatrix} \frac{\partial A^x}{\partial x} & \frac{\partial A^x}{\partial y} \\ \frac{\partial A^y}{\partial x} & \frac{\partial A^y}{\partial y} \end{bmatrix}\right|_{\sum_{i=0}^{N} c_i S_i(x,y)} = \begin{bmatrix} p_0 & p_1 \\ p_3 & p_4 \end{bmatrix} \\
\frac{\partial \sum_{i=0}^{N} c_i S_i(x,y)}{\partial c_i} &= \begin{bmatrix} \frac{\partial \sum_{i=0}^{N} c_i S_i^x(x,y)}{\partial c_i} \\ \frac{\partial \sum_{i=0}^{N} c_i S_i^y(x,y)}{\partial c_i} \end{bmatrix} = \begin{bmatrix} S_i^x(x,y) - S_0^x(x,y) \\ S_i^y(x,y) - S_0^y(x,y) \end{bmatrix}
\end{aligned}
$$

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

$$
\begin{aligned}
\frac{\partial I^{novel} \circ \overline{S}(x,y)}{\partial p_i} &= \left.\frac{\partial I^{novel}(x,y)}{\partial (x,y)}\right|_{\overline{S}(x,y)} \frac{\partial \overline{S}(x,y)}{\partial p_i} \\
\frac{\partial \overline{S}(x,y)}{\partial p_0} &= \left[\sum_{i=0}^{N} c_i S_i^x(x,y),\ 0\right]^T \\
\frac{\partial \overline{S}(x,y)}{\partial p_1} &= \left[\sum_{i=0}^{N} c_i S_i^y(x,y),\ 0\right]^T \\
\frac{\partial \overline{S}(x,y)}{\partial p_2} &= [1,0]^T \\
\frac{\partial \overline{S}(x,y)}{\partial p_3} &= \left[0,\ \sum_{i=0}^{N} c_i S_i^x(x,y)\right]^T \\
\frac{\partial \overline{S}(x,y)}{\partial p_4} &= \left[0,\ \sum_{i=0}^{N} c_i S_i^y(x,y)\right]^T \\
\frac{\partial \overline{S}(x,y)}{\partial p_5} &= [0,1]^T
\end{aligned}
$$

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

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

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

Our matching algorithm uses a coarse-to-fine approach to improve robustness ([Burt and Adelson, 1983]; [Burt, 1984]) by creating a pyramid of images with each level of the pyramid containing an image that is one fourth the size of the one below. The correspondence fields must also be subsampled, and all x and y coordinates must be divided by two at each level. The optimization 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. Section 5 shows a number of examples to illustrate the performance of the matching algorithm.

Another useful technique implemented in our model is principal components analysis. The eigenvectors for both the shape space and texture space are computed independently. 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). We emphasize that the technique performs well without using eigenvectors, which however can provide additional computational efficiency.

4.1 The matching algorithm

The following pseudo code describes the matching algorithm. The model is learned once from the set of prototypes. 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, $\hat{I}_j$

MATCHING PHASE

Given a novel image and the model S_j and texture vectors, $\hat{I}_j$

1. Create image pyramids for I^{novel}, $\hat{I}_j$ and S_j
2. Initialize parameters **c** and **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 **c**, **p** and **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

5 Examples and Results

The model described in the previous sections was tested on two different classes of objects. The first was the class of frontal views of human faces. Two different databases of faces were used. One was from Thomas Vetter and Nikolaus Troje of the Max Planck Institute in Tubingen, Germany [Troje and Bülthoff, 1995]. The other was from David Beymer, formerly of the MIT AI Lab [Beymer, 1996]. The correspondences for the Vetter-Troje face database were computed automatically by using an algorithm which relied on an optical flow algorithm implemented by Bergen and Hingorani [Bergen and Hingorani, 1990]. Of the 130 faces, 100 were successfully put in correspondence using this algorithm. The remaining 30 faces were used as novel inputs to test the matching algorithm (an automatic bootstrapping algorithm which successfully put all 130 faces in correspondence is described in Vetter, Jones and Poggio, 1996.) The Beymer face database – consisting of 62 faces – had been set into correspondence by manually specifying a number of corresponding points and then interpolating to get a dense correspondence field. The second example object class was a set of side views of automobiles for a total of 40 car images. The correspondences for this class were also computed by manually specifying a number of corresponding points and then interpolating.

The matching algorithm as described in section 4 was run with the following parameters. The number of samples randomly chosen by the stochastic gradient algorithm per iteration was 40. The stochastic gradient algorithm was run for 8000 iterations per pyramid level. Three levels were used in the image pyramids.

The running time of the matching algorithm for the Vetter-Troje faces (after compressing the shape and texture spaces by finding the principal components and using 30 shape eigenvectors and 20 texture eigenvectors) was about 3.5 minutes on an SGI Indy. The running time for the Beymer faces using 61 prototypes (with no compression) was about 9 minutes. For the cars, the running time was about 5 minutes using all 40 prototypes (with no compression).

5.1 Face model #1

To build the first model of frontal views of faces, 100 images of different people's faces were used as prototypes. These images were 256 pixels high by 256 pixels wide. Figure 2 shows 35 prototypes. The face in the upper, left corner was used as the reference face relative to which correspondences were computed for each of the other prototypes (none of the images had a beard; the hairs were covered by a tight swimming cap). We tested the ability of the model to match novel faces (none of which were among the prototypes) by using the matching algorithm described in section 4.1. Figure 3 and figure 4 show some typical examples of matches to novel faces, suggesting that indeed new face images - taken under similar illumination conditions –can be approximated well as a linear combination of our prototypes. It is likely that even better matches can be obtained by adding more prototypes to the model. It should be noted that face images taken under different lighting conditions from those used for the prototypes are not matched well, since the model cannot represent the resulting textures.

5.2 Face model #2

The second face model consisted of the Beymer face database, shown in figure 5. These images were 226 pixels high by 184 pixels wide. The face in the upper left corner was used as the reference face. The correspondences from the reference face to each of the other faces was given by manually specifying a small number of corresponding points. These faces are more difficult to put in correspondence due to the hair, beards and mustaches. The results of testing the model's ability to match novel faces are shown in figure 6. Because we only had 62 faces to work with, 61 were used in the model and one was left out so that it could be used as a novel image. Since the hair is not modelled well, the faces in figure 6 have been cropped appropriately. One can see that the matching algorithm produced good matches to the novel faces.

5.3 Cars

As another example of a linear object class model, we chose side views of cars. The car images are 96 pixels high by 256 pixels wide. Figure 7 shows 14 of the 40 cars used as prototypes in the model for the class of cars. The car in the top left of the figure was used as the reference car. The model defined by these prototypes and their correspondences was tested on its ability to match a number of novel cars. Figure 8 shows some example matches to novel cars. The novel cars are matched reasonably well, although not as sharply as the faces. This is probably due to the fact that the correspondences given for the car prototypes are not as accurate as in the case for faces. There are two reasons for this. One is the variability of appearance between cars: some cars have features that do not appear on other cars (such as spoilers). Defining correspondences for such features is ambiguous. Another reason for inaccurate correspondences is that the correspondences were given by specifying a few corresponding points by hand and then interpolating to get a dense correspondence field. This is a less accurate method than the optical flow based method used with face model #1. Unfortunately, the optical flow algorithm did not work well between the car prototypes.

5.4 Robustness of the matching algorithm

It is important to characterize how robust the matching algorithm is to translations, rotations, scaling and occlusion of the input image. 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

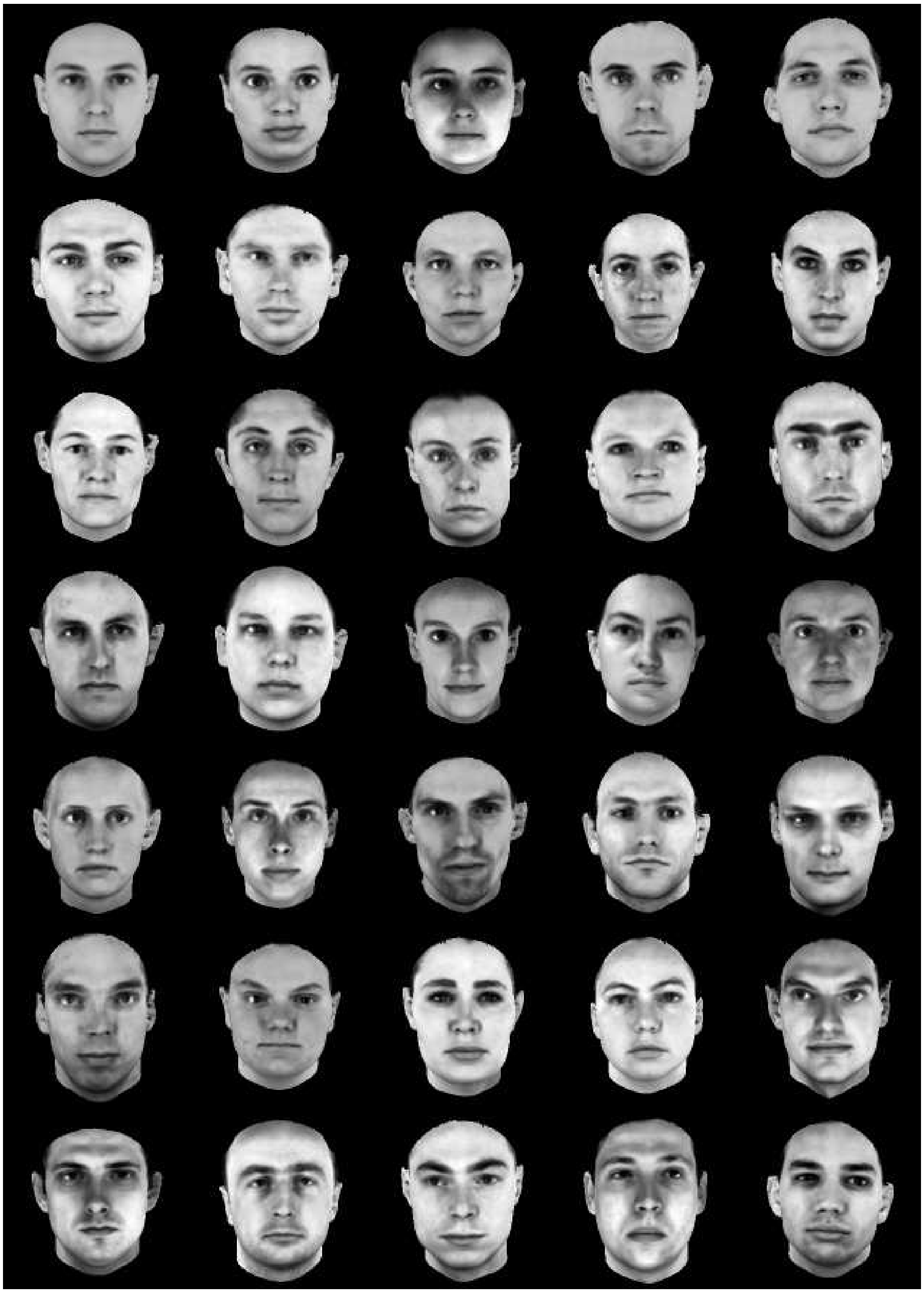

Figure 2: *Thirty-five of the 100 prototype faces used to create a model of human faces.*

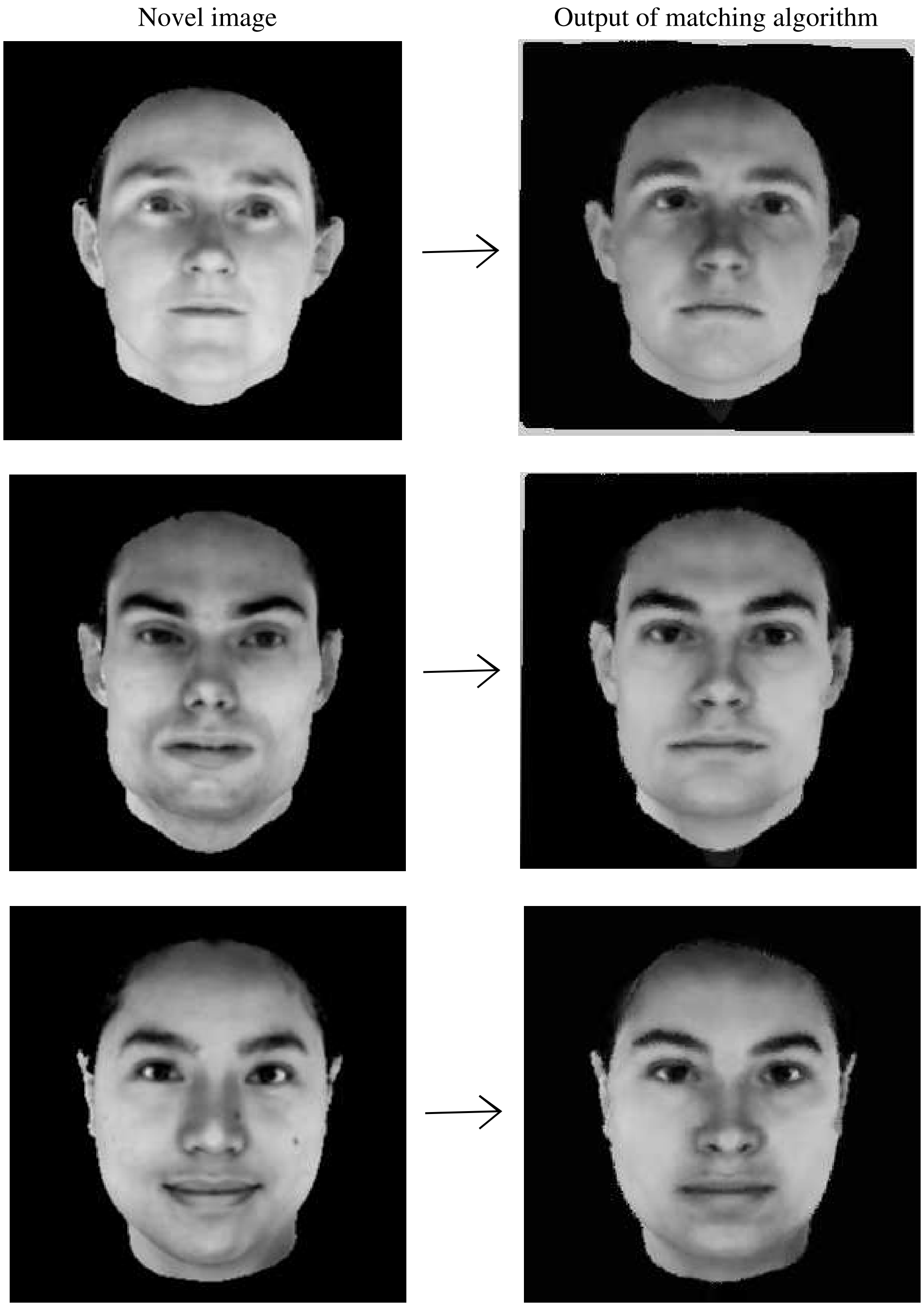

Figure 3: *Three examples of matching the face model to a novel face image.*

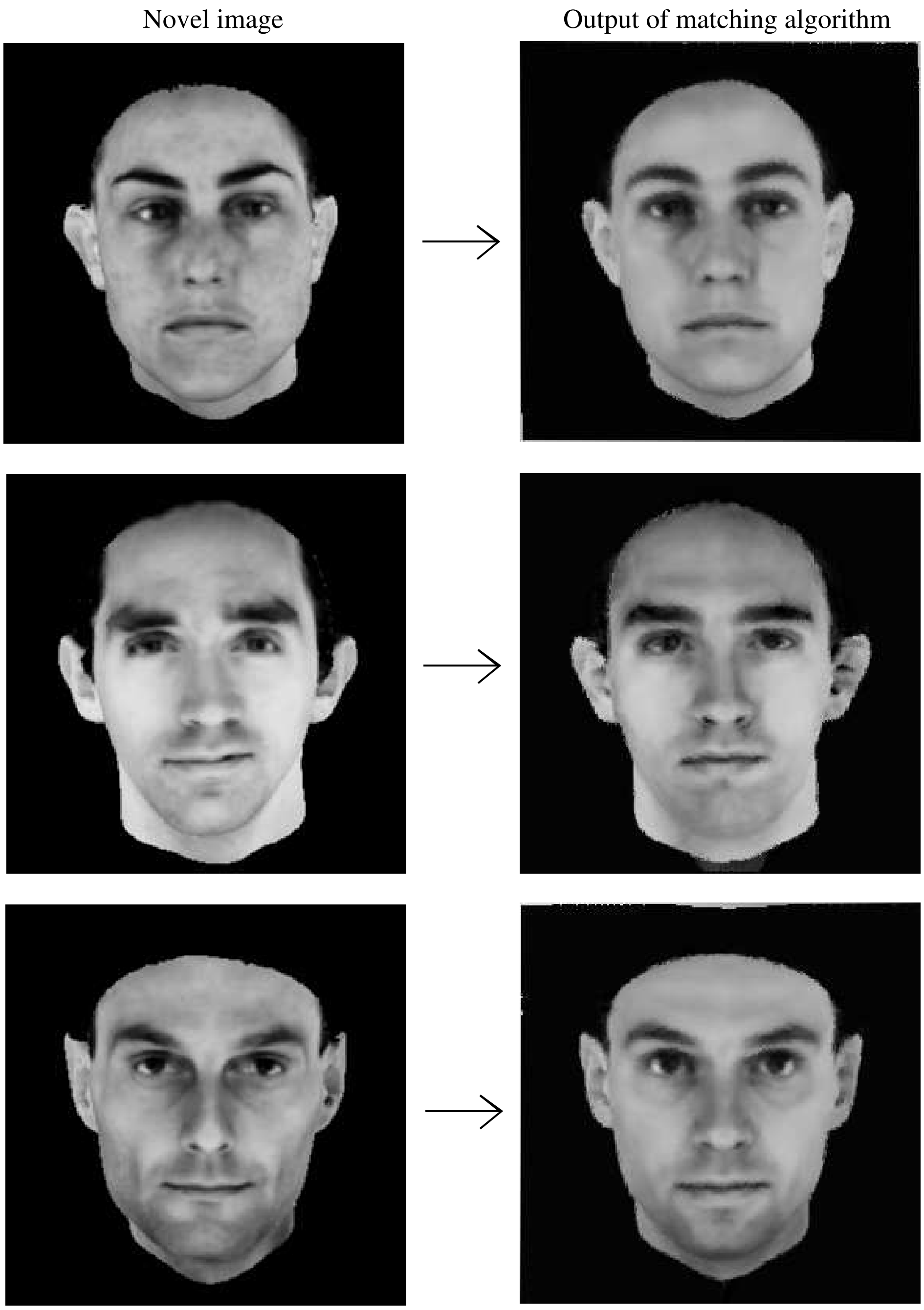

Figure 4: *Three more examples of matching the face model to a novel face image.*

Figure 5: *Database of 62 prototype faces used to create the second model of human faces.*

Figure 6: *Four examples of matching the second face model to a novel face image.*

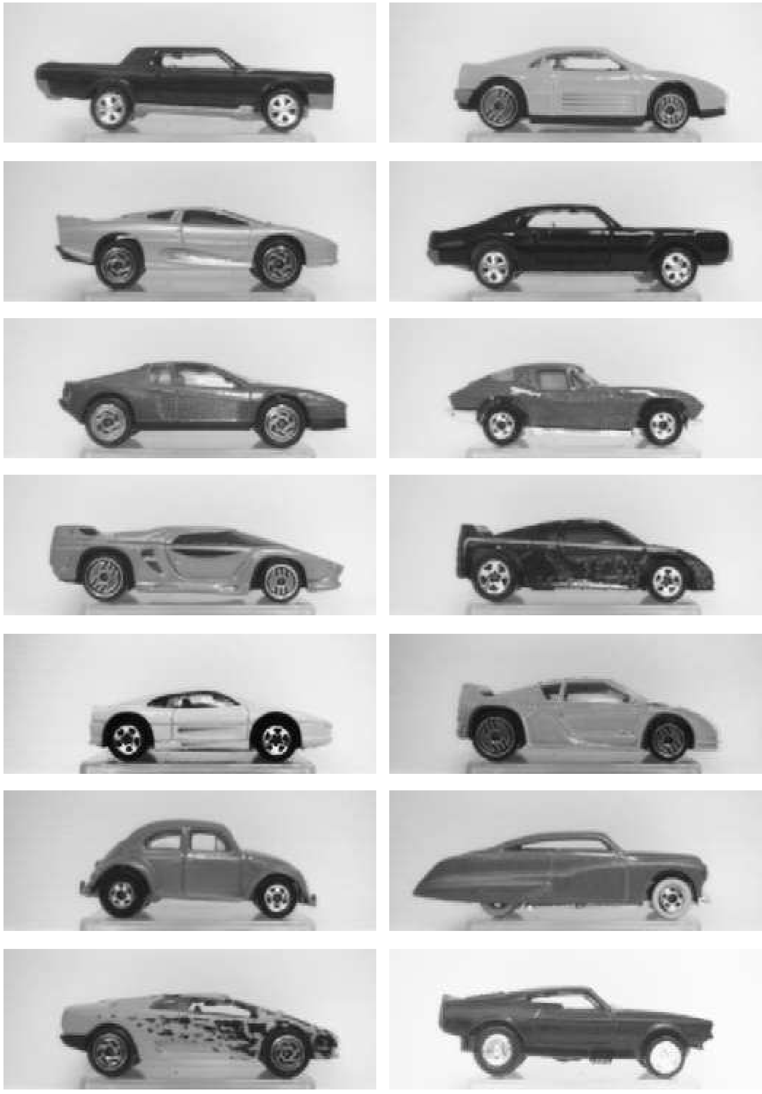

Figure 7: *Fourteen of the 40 prototype cars used to create a model of cars.*

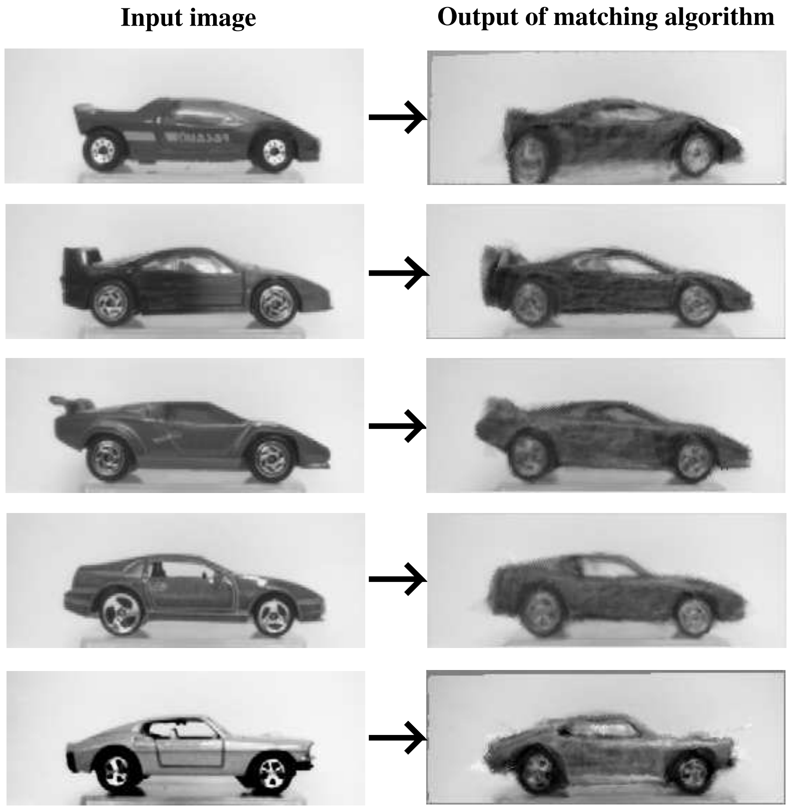

Figure 8: *Five examples of matching the car model to a novel car image, which was not among the model prototypes.*

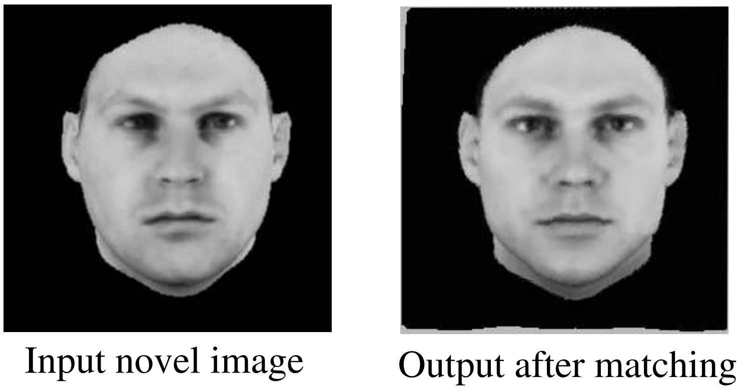

Figure 9: *Example of matching a novel face that has been translated 6 pixels in the x direction and -10 pixels in the y direction. The matching algorithm automatically translated the model image by optimizing the affine parameters for translation.*

the analysis only applies directly to these face images, a general idea of the performance of the matching algorithm can be inferred from these results.

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 9. 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 10 summarizes the matching performance for four novel images over a range of different translations from +20 pixels to -20 pixels in both the x and y directions. (Recall that the images are 256×256.) A good match is defined as one that has error less than or equal to 1.2 times the error for the untranslated input image. This definition is somewhat arbitrary, but it was found by visual inspection that an image with 1.2 times the error of the untranslated output image still looked very similar to the untranslated input image. One with more error began to look significantly different. By error, we mean the L_2 error as defined in equation 7. 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 ± 10 pixels in both the x and y directions.

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 11. Again, the matching algorithm must fit the rotated input image by changing the affine parameters. Figure 12 shows the results of matching four different input images with varying amounts of rotation. The circular graphs show the error for the different amounts of rotation tested. Each line segment is proportional in length to the error of the resulting match. 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 with a line segment inside the circle is considered a good match. The results show that rotations from -30 degrees to +30 degrees are generally matched well.

The matching algorithm was next tested for robustness to scaling. An example of a scaled

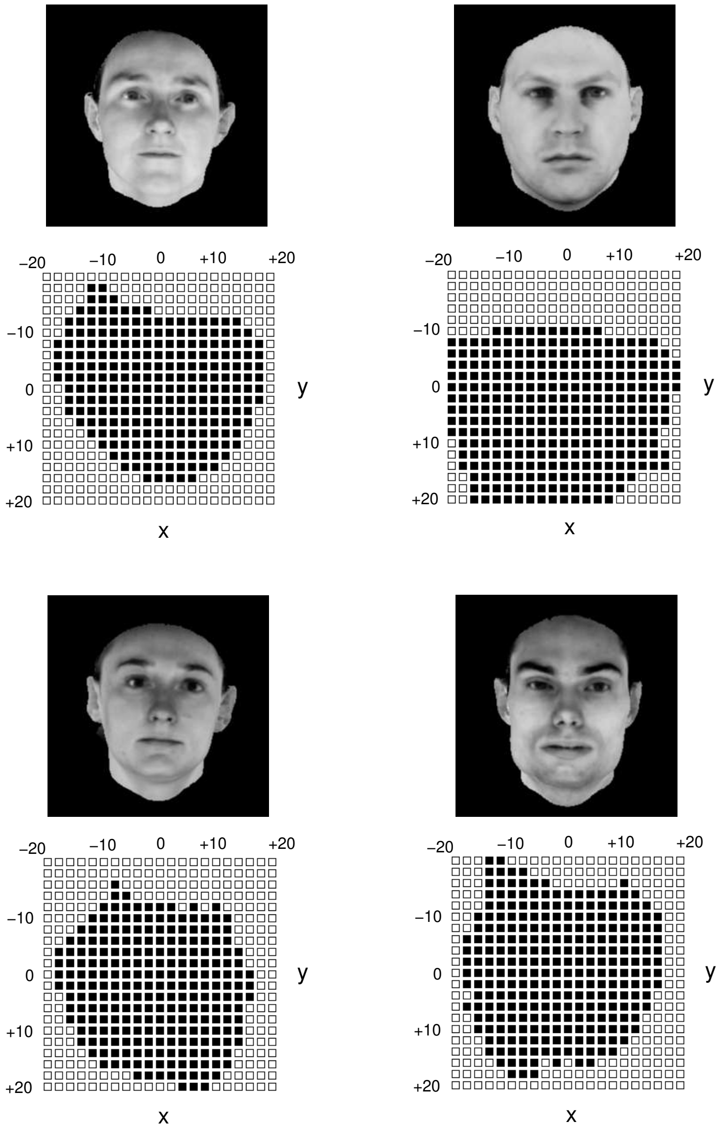

Figure 10: *Summary of robustness tests for translation. Each filled-in square indicates a translation for which the input image was matched well. The face above each grid is the input image before translation.*

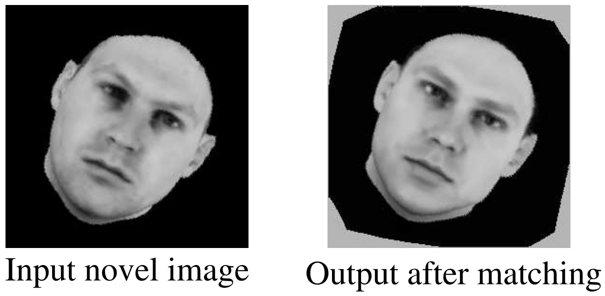

Figure 11: *Example of matching a novel face that has been rotated 27 degrees. The matching algorithm automatically rotated the model by optimizing the affine parameters for rotation.*

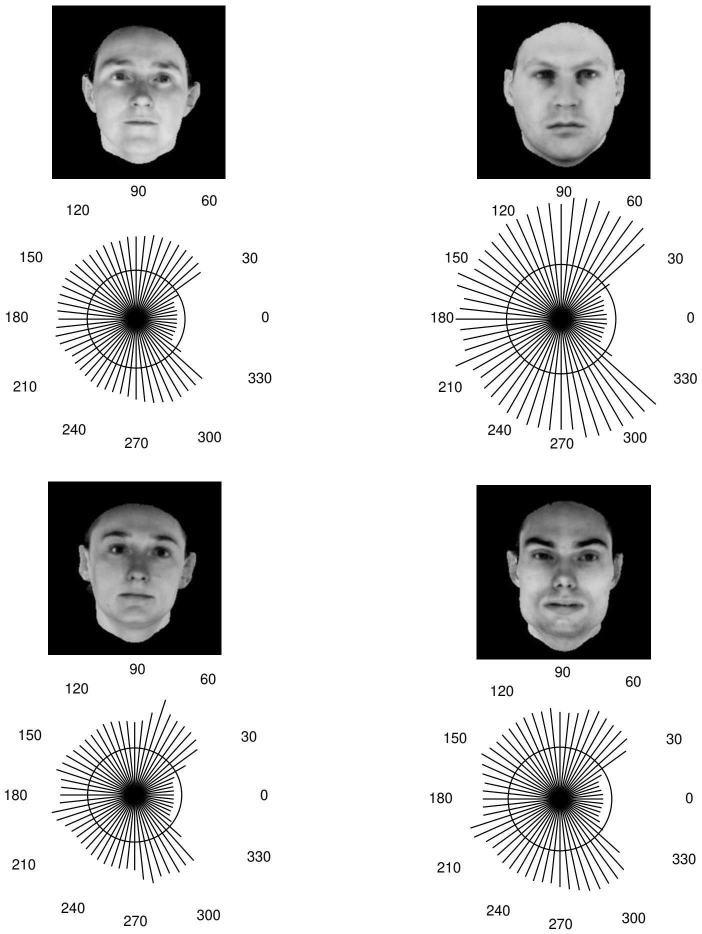

Figure 12: *Summary of robustness tests for rotation. Each line segment is proportional in length to the error of the match at that angle. Angles for which the line segment is inside the circle are considered good matches. The face above each grid is the input image before rotation.*

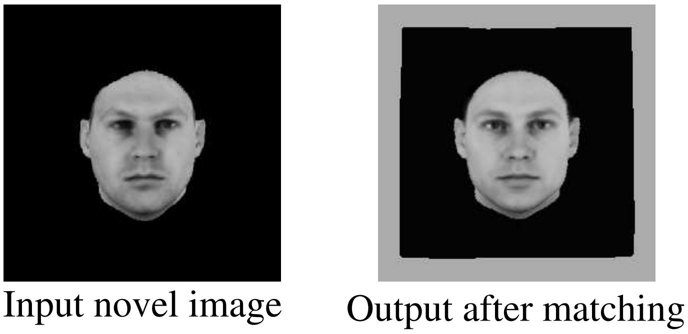

Figure 13: *Example of matching a novel face that has been scaled to 0.6 times original size. The matching algorithm automatically scaled the model by optimizing the parameters for scale.*

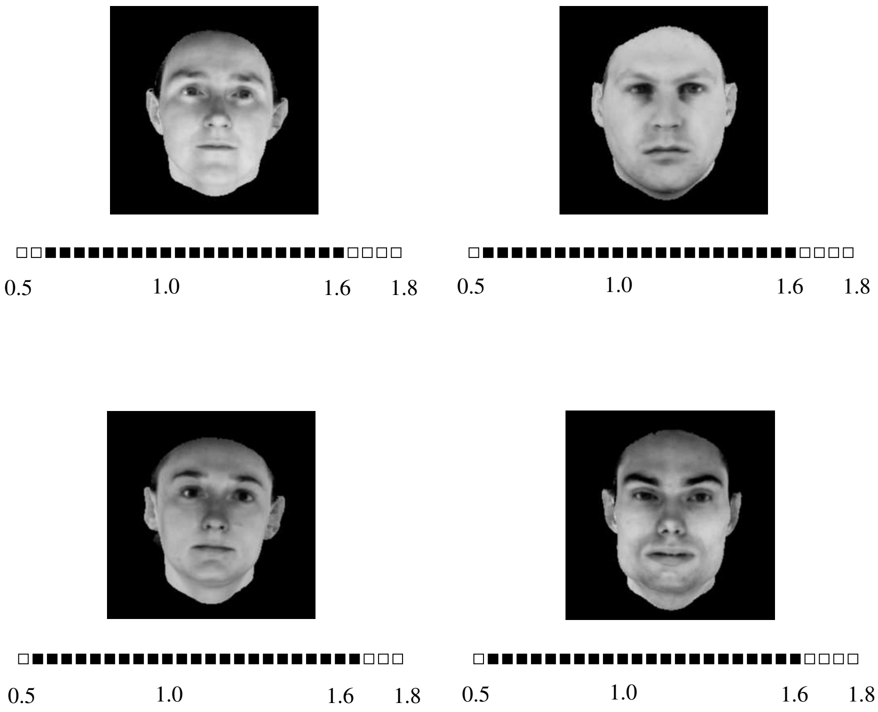

Figure 14: *Summary of robustness tests for scale. Each filled-in square indicates a good match at that scale.*

face and the resulting best match is shown in figure 13. 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 it as it fits the parameters of the model, including the affine parameters which allow for changes in scale. Figure 14 shows the results of matching four 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 scaled input image. The results show that the matching algorithm can reliably match input faces with scales between about .6 and 1.6 times the original size.

Finally, the robustness of the matching algorithm to partially occluded input images was studied. In these tests, a 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 matching algorithm was compared against the original unoccluded image to determine the error of the match. Figure 15 shows some occluded input images and the resulting best match found by the matching algorithm.

The matching algorithm used to match the occluded images is the same as in the previous experiments, with one exception. In each iteration of the stochastic gradient procedure, 40 random points are chosen at which to compute the gradient. The modification made to handle occlusions is to throw out the six points that have the highest error (defined as the difference between the corresponding points in the novel image and the current guess for the model image). The idea is that the highest errors are most likely to come from occluded regions, and we do not want to use gradient information from occluded regions in the optimization procedure. This rule was experimentally found to apply reliably for only the six or so points with the largest errors independently of the percentage of occlusion in the image.

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.

6 Comparison to eigenfaces

As we mentioned briefly in a previous section, the eigenimage approach also leads to a model which consists of linear combinations of some basic images, since each of the eigenimages is a linear combination of the image set. Our technique relies instead on the linear combination of shape and texture vectors associated with the basic set of prototypical images. It is natural to ask how the two approaches compare. We describe here two exemplary cases.

6.1 "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 16. 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 17 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

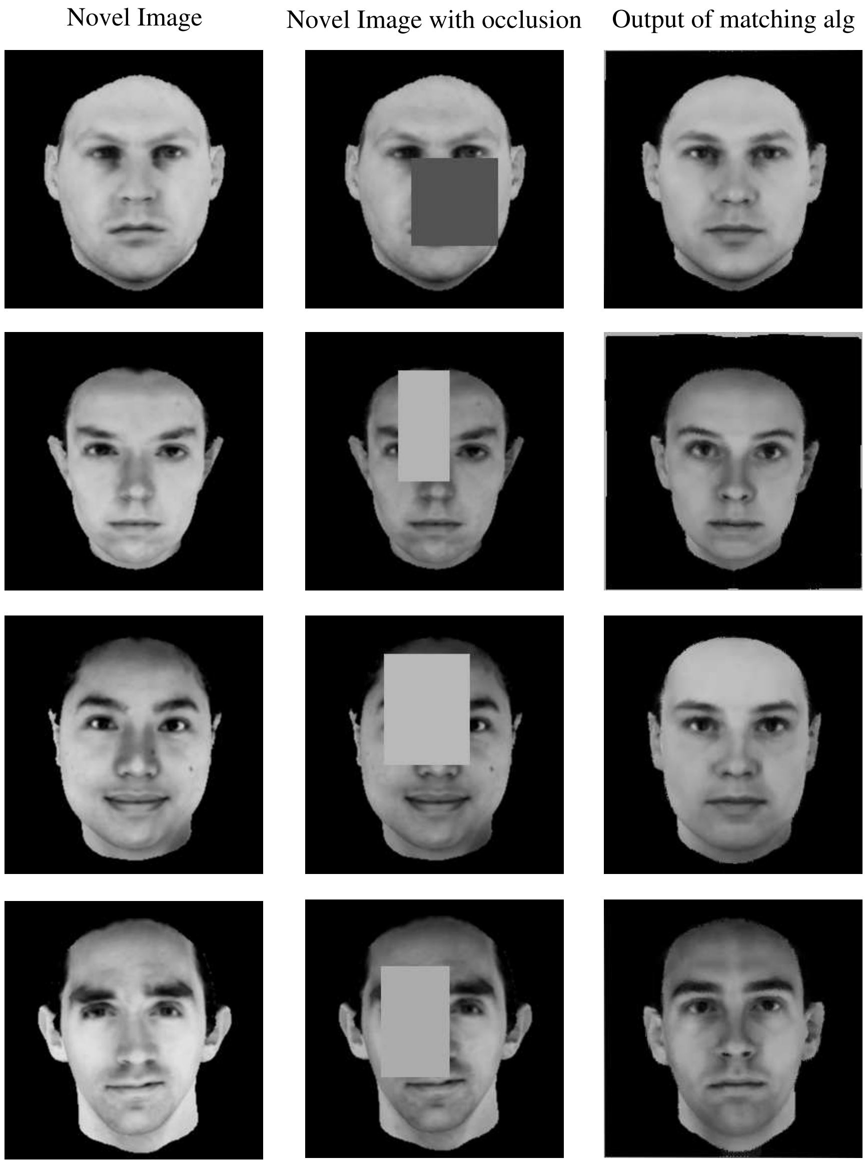

Figure 15: *Examples of matching partially occluded face images. The input images to the matching algorithm are in the middle column. These input images are occluded versions of the face images in the left column. The output of the matching algorithm is shown in the right column.*

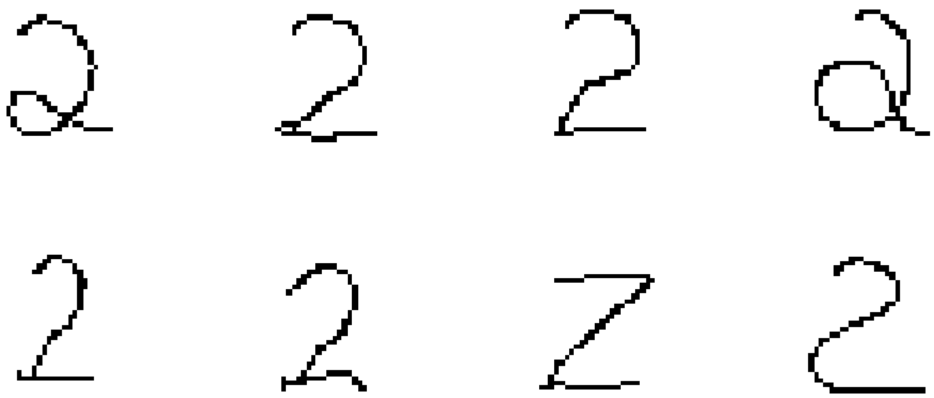

Figure 16: *Prototype examples for a class of handwritten "2's".*

a)

b)

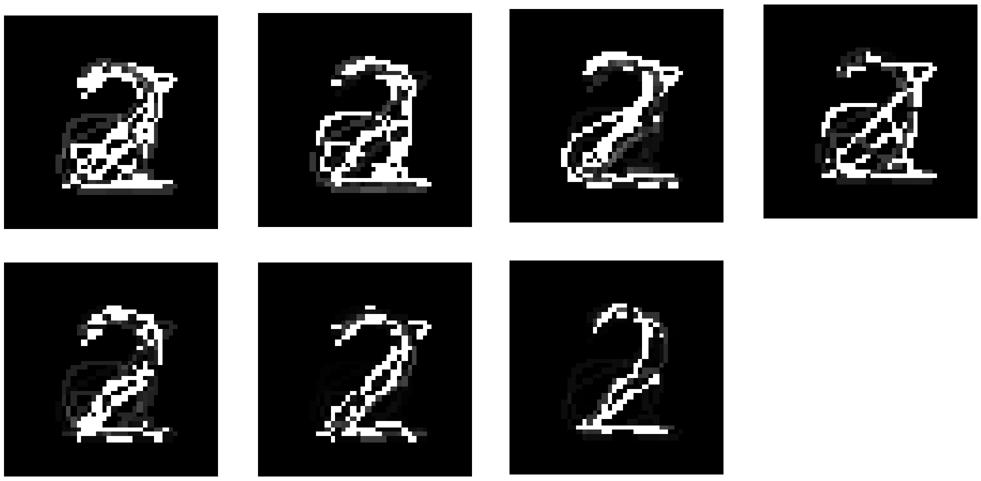

Figure 17: *a) Mean image "2" found by averaging the prototype images of "2's". b) So-called eigenimages of "2's", that is eigenvectors computed from the set of images of prototype "2's" (without using pixelwise correspondences).*

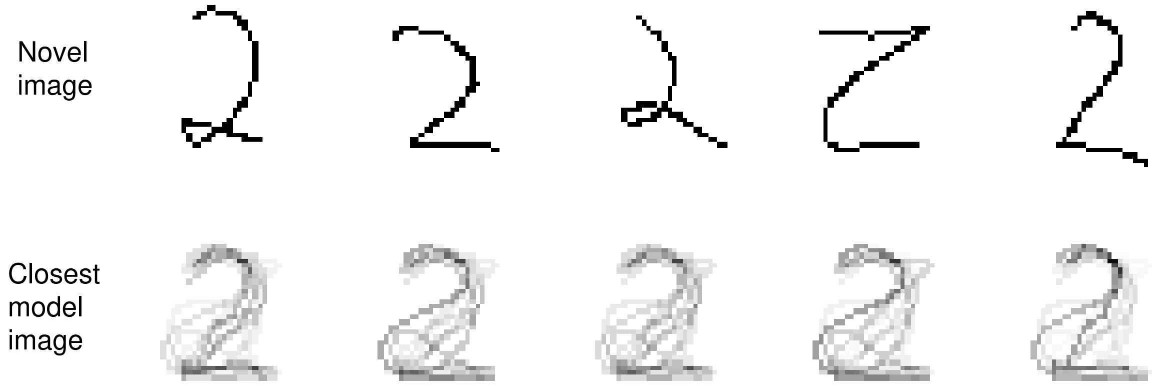

Figure 18: *Example matches to novel "2's" using the eigenimage approach. Matching with eigenimages yields poor results.*

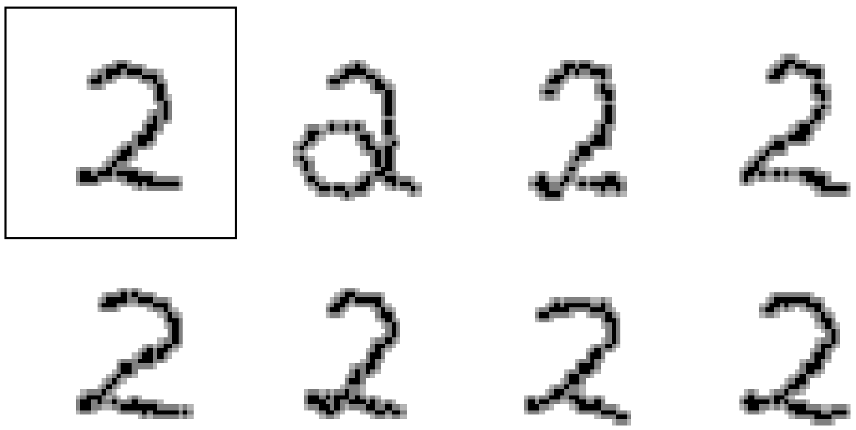

Figure 19: *Mean "2" obtained by rendering the reference shape vector "2" which is equivalent to warping the reference "2" with the average of the prototype correspondence fields (top left in box) followed by the full set of the shape eigenvectors of the prototypes. They are rendered as images obtained as warps of the reference "2" by the eigenvector correspondence fields.*

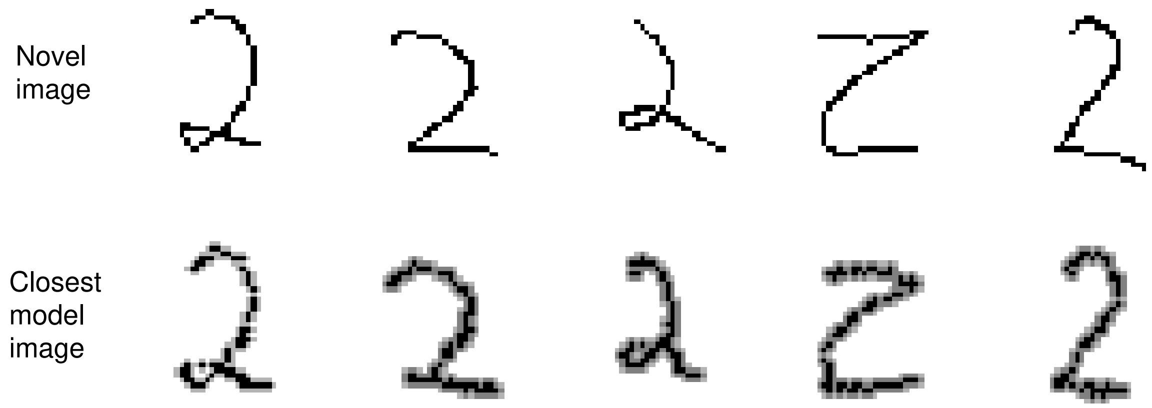

Figure 20: *Example matches to novel "2's" using the eigenvector approach with correspondences. The output "2's" were blurred to fill in "holes" left after warping.*

ghosts of the prototype images of which they are linear combinations. Figure 18 shows some typical matches of the eigenvector model to novel images of "2's". The matches are not very good. Next we contrast this to the matching produced using our linear object class model based on pixelwise correspondences. Figure 19 shows the eigenvector representation for the shape 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 20 shows the matches to novel "2" images using the linear object class model. They are significantly better than in the case without correspondences.

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 and Pentland, 1991] was computed from the face database #1 shown in figure 2. These faces are aligned using the center of the face. 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 21 shows the matches to four novel faces found using the eigenface model. The right column of figure 21 shows the best matches for the linear object class 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 linear object class model results in superior matching relative to the eigenface approach.

7 Applications

There are a number of interesting applications to which the linear object class model and matching algorithm can be applied. We discuss a few of these briefly below.

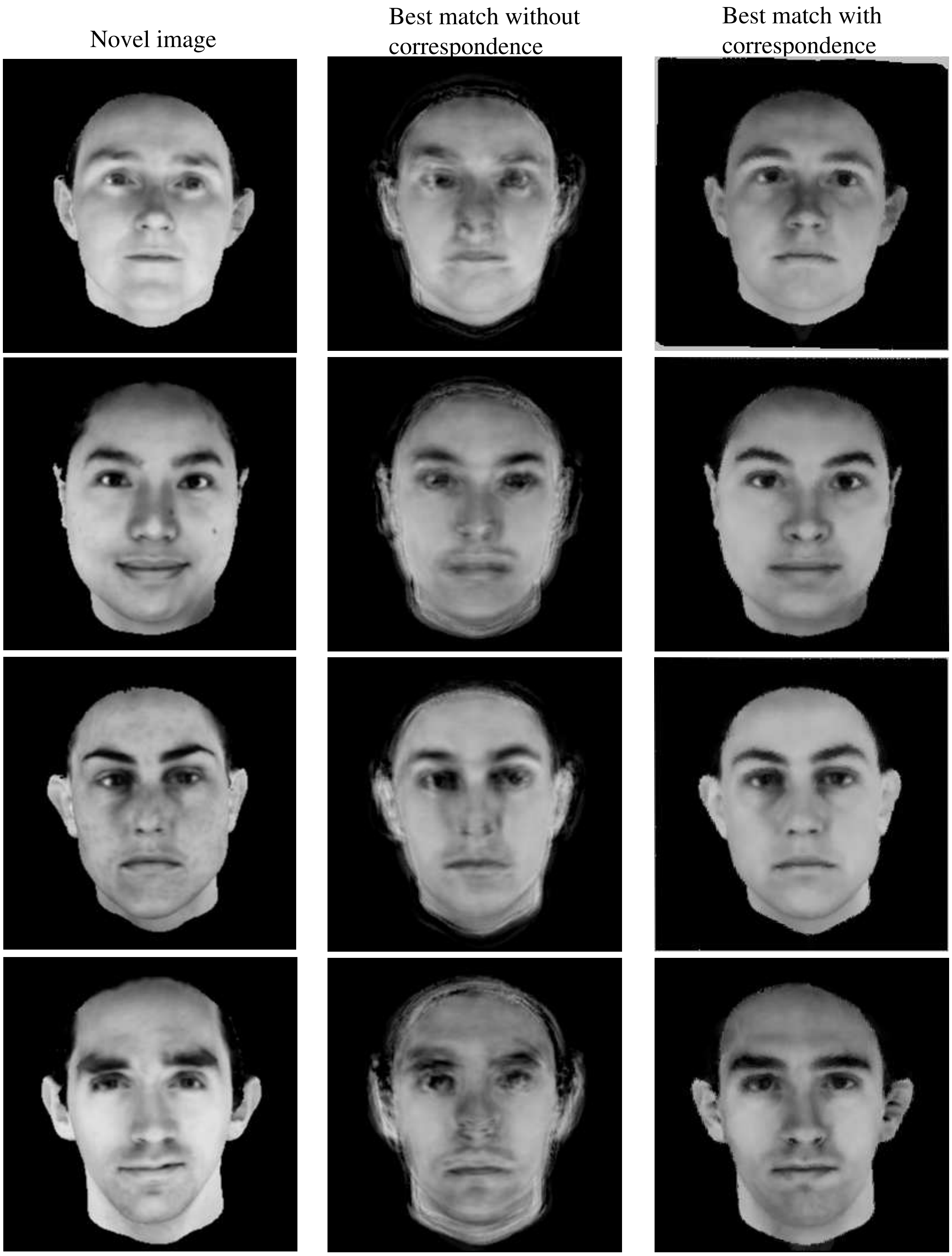

Figure 21: *Example matches to novel faces (left) using the so-called eigenface approach (middle) – which does not use pixelwise correspondence – and using our approach (right) which uses pixelwise correspondences.*

7.1 Example-based correspondence

The original application of the technique described here is to "vectorize" a novel image [Poggio and Beymer, 1996], providing dense correspondence between two images of a known class of objects. Once a novel face is approximated by the linear model it is effectively in correspondence with the model and each of the prototypes. Two novel images can be therefore set in correspondence in this way. A very similar correspondence technique was successfully used by [Beymer and Poggio, 1995] in a real-time vectorizing step needed for the generation of virtual views. This 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.

7.2 Analysis of image parameters

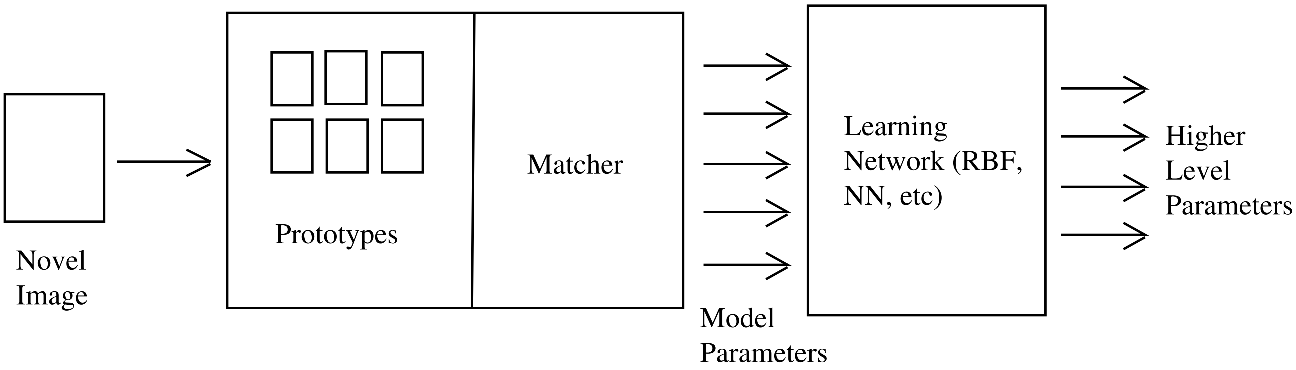

Figure 22: *A general image analysis system. The novel image is first matched by using the matching algorithm described in section 4. The resulting model parameters become input to a learning network which has been trained to map model parameters onto higher level parameters (such as pose or expression), defined by the user.*

Custom image analysis can be implemented by mapping the linear coefficients of the model ($\mathbf{c}$ and $\mathbf{b}$) to higher level parameters such as pose by using a RBF network or other learning networks. In other words, if some of the prototypes represent changes in pose 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 (see also [Blake and Isard, 1994]). A mapping from model parameters to phonemes can be learned to read the lips. Figure 22 illustrates a general system for image analysis which first matches a linear object class 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.

7.3 Top-down low-level vision

Poggio and Vetter (1992) showed that if an object's 3D shape can be represented as a linear combination of a small number of 3D prototypical shapes then the (2D) shape vector associated with the image can also be represented as a linear combination with the same coefficients (under orthographic projection and same viewpoint). This property offers the possibility of estimating 3D shape of an object of a known class from a single image. This also suggests how to synthesize virtual views (see Poggio and Vetter, 1992).

Both these tasks are examples of low-level vision tasks that are performed in a top-down, object specific, example-based way. It is conceivable that a similar approach may be valid for other low-level vision tasks, such as, for instance, finding the ideal line drawing corresponding to a natural image (Jones, Sinha, Vetter and Poggio, in preparation) or the object color or its 3D shape.

7.4 Image compression

Another application for our model-based matching algorithm 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 flexible 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 less than 100 bytes. The novel face can then be synthesized from these parameters (assuming of course that the model is independently available).

7.5 Object verification

Model-based matching can also be used for object verification. Here the task is to determine if a specific region of an image contains an instance of some object class or not. The verification task entails matching the object class model and using the error of the final match. A threshold can be set based on appropriate statistics to determine if the match is good enough for positive detection.

8 Extensions

The models we have developed so far need to be larger in order to have a greater representational power. They must be made more robust for changes in imaging parameters, in particular to changes in illumination and camera properties. We are currently investigating techniques for handling different lighting conditions by using preprocessing filtering steps and - more importantly – by adding prototypes taken under different lighting conditions ([Shashua, 1992a], [Hallinan, 1995]).

In addition, there are two major directions in which we plan to extend this work. One is to decompose a model into simpler components and thus create hierarchical models built from components. The other major extension is to address the problem of automatically finding pixelwise correspondences among the prototypes. Pixelwise correspondences give our approach much of its power, but also introduce a large burden in defining a model. A technique for automating the process of finding correspondences among the prototypes would be an extremely useful tool. We address each of these extensions in the next two sections.

8.1 Hierarchical models

The idea of hierarchical models is to divide images into components such as eyes, nose and mouth (in the case of faces) and build separate models of each component using a number of prototypes. Models of more complex object classes could then be built from simple components, possibly using several layers of components. In the previous example, a model for faces would consist of the linear combination of the example (shape) vectors containing the position of the "center" of the components (eyes, nose, mouth,...) in addition to the linear combinations of the shape and texture vectors for each example. The advantages of such a hierarchical model is that fewer prototypes would be needed since each component is simpler than the whole image, occlusions can be dealt with more easily since occluded components could simply go unmatched, and a library of generic and reusable features could possibly be built and used to represent many object classes. We are currently investigating algorithms for automatically splitting a set of images from a particular class into its component parts.

8.2 Automatically finding correspondences among prototypes

It is often difficult to compute pixelwise correspondences among the prototype images. We have used a number of techniques to do this including optical flow algorithms and semi-automatic techniques which require a user to specify a small set of corresponding points in each image. Automating this process would be a great help to model building and also to give some biological plausibility to the class of flexible models introduced here. Recently, we had promising results with a bootstrapping technique for automatically finding correspondences for images in the same object class ([Vetter *et al.*, 1996], in preparation). The idea is first to determine the pixelwise correspondence between a reference image and just one other prototype using an automatic technique such as optical flow. The resulting correspondence field can be used to build a two example model, which is then fit to each of the other prototypes and further improved by running optical flow between the matched image and the respective prototype. The prototype which is best fit is then added to the model. This process is iterated until all prototypes have been matched well. Thus, correspondences are found by iteratively building a model using the matching algorithm to get a reasonable match to one of the prototypes and then further bootstrapping the correspondences by using an optical flow algorithm. Its success is a major step towards making the approach described in the present paper truly general and powerful.

9 Conclusions

In this paper we have described a flexible model to represent images of a class of objects and in particular how to use it to analyze new images and represent them in terms of the model. Our flexible 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 flexible 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 flexible 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 paper is to solve the analysis problem: how to apply the flexible model, so far used as a generative model, for image analysis. Key to the analysis step is matching and a new matching algorithm is the main focus of this paper. We have also described in more detail than in any previous paper the model itself and how to obtain it and learn it from protoype images through pixelwise correspondence. Analysis coupled with synthesis offers a large number of new applications of the flexible 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).

Given the power and the generality of this model one may ask about its possible biological significance. This is still an open question, though it appears that some of the top-down learning tasks that this technique makes possible are indeed performed routinely by the human visual system (Sinha and Poggio, in press).

10 Acknowledgements

The authors would like to thank Thomas Vetter, Amnon Shashua, Federico Girosi, Paul Viola and Tony Ezzat for helpful comments and discussions about this work. Michael Oren suggested the notation of section 3.1 in addition to many other very useful comments.

References

[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.

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

[Besl and Jain, 1985] Paul J. Besl and Ramesh C. Jain. Three-dimensional object recognition. *Computing Surveys*, 17(1):75–145, 1985.

[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, 1995] David Beymer. Vectorizing face images by interleaving shape and texture computations. A.I. Memo 1537, MIT, 1995.

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

[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.

[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.

[Ezzat, 1996] Tony Ezzat. Example-based image analysis and synthesis for images of human faces. Master's thesis, Massachussetts Institute of Technology, 1996.

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

[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.

[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.

[Kirby and Sirovich, 1990] M. Kirby and L. Sirovich. The application of the karhunen-loeve procedure for the characterization of human faces. *IEEE*, 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.

[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.

[Murase and Nayar, 1995] Hiroshi Murase and Shree K. Nayar. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, pages 5–24, 1995.

[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 Brunelli, 1992] Tomaso Poggio and Roberto Brunelli. A novel approach to graphics. A.I. Memo 1354, MIT, 1992.

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

[Shashua, 1992a] Amnon Shashua. *Geometry and Photometry in 3D Visual Recognition.* PhD thesis, Massachussetts Institute of Technology, 1992.

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

[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.

[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 *et al.*, 1996] Thomas Vetter, Michael Jones, and Tomaso Poggio. A bootstrapping algorithm for learning linearized models of object classes. A.I. Memo submitted, MIT, 1996.

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