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

A.I. Memo No. 1600 February 1997
C.B.C.L. Paper No. 143

A Bootstrapping Algorithm for Learning Linear Models of Object Classes

Thomas Vetter, Michael J. Jones and Tomaso Poggio

Abstract

Flexible models of object classes, based on linear combinations of prototypical images, are capable of matching novel images of the same class and have been shown to be a powerful tool to solve several fundamental vision tasks such as recognition, synthesis and correspondence. The key problem in creating a specific flexible model is the computation of pixelwise correspondence between the prototypes, a task done until now in a semiautomatic way. In this paper we describe an algorithm that automatically bootstraps the correspondence between the prototypes. The algorithm – which can be used for 2D images as well as for 3D models – is shown to synthesize successfully a flexible model of frontal face images and a flexible model of handwritten digits.

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 N00014-95-1-0600. 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

In recent papers we have introduced a new type of flexible model for images of objects of a certain class. The idea is to represent images of a certain type – for instance images of frontal faces – as the linear combination of prototype images and their affine deformations. This flexible model can be used as a generative model to *synthesize* novel images of the same class. It can also be used to *analyze* novel images by estimating the model parameters via an optimization procedure. Once estimated the model can be used for indexing, for recognition, for image compression and for image correspondence.

At the very heart of our flexible models is an image representation in terms of which a linear combination of images makes sense. For a set of images to behave as vectors, they must be in pixelwise correspondence (see [Poggio and Beymer, 1996]). Our model uses pixelwise correspondence between example images and should not be confused with techniques which use linear combinations of *images* such as the so-called eigenfaces technique ([Kirby and Sirovich, 1990]; [Turk and Pentland, 1991]). 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 is represented as a *shape vector* and a *texture vector*. The shape vector specifies how the 2D shape of the example differs from a reference image and corresponds to the flow field between the two images. 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 (grey level or color values) of the image. Our flexible model for an object class is then a linear combination of the example shape and texture vectors.

The flexible model has been used for many synthesis and analysis tasks, mentioned later.

1.1 A key problem: creating the model from prototypes

The distinguishing aspect of our linear flexible models is that they are linear combinations of prototype shape and texture vectors and not of images ([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 – and in fact the model can be used to compute correspondence if necessary. In our past papers we computed correspondence between the prototypes with automatic techniques such as optical flow. 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 paper we describe a bootstrapping technique that seems capable of computing correspondence between prototypical images in cases in which standard optical flow algorithms fail.

1.2 Past work

The "linear class" idea of [Poggio and Vetter, 1992] and [Vetter and Poggio, 1995] together with the image representation used by [Beymer *et al.*, 1993] (see [Beymer and Poggio, 1996] for a review) is the main motivation behind the work of this and previous papers. Poggio and Vetter introduced the idea of linear combinations of views to define and model *classes* of objects, trying to extend 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 used the model mainly for *synthesis* tasks. In particular, for linear object classes, affine transformations can be learned exactly from a small set of examples and used to generate new, virtual views. For instance, it is possible to learn to generate a new virtual view of an object from a single example view, represented as a 2D shape vector, provided appropriate prototypical views of other objects of the same class are available (under orthographic projection). Thus, new views of a specific face with a different pose or expression can be estimated and synthesized from a single view. In a very similar way, 3D structure can be estimated from a single image if the image and the structure of a sufficient number of prototypical objects of the same class are available. More generally, flexible models of this type can underly learning of visual tasks in a top-down way, specific to object classes (Jones, Sinha, Vetter and Poggio, in preparation).

The problem of using the flexible model to *analyze* novel images was the main concern of Jones and Poggio ([Jones and Poggio, 1995, Jones and Poggio, 1996]). They introduced a novel approach to match flexible linear models to novel images that can be used for several visual analysis tasks, including recognition, image correspondence and image compression.

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. The work of Taylor and coworkers *et. al.* ([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. Many other flexible models have been proposed, such as the model of Blake and Issard [Blake and Isard, 1994].

2 Linear models

In this section we formally specify the linear object class model and describe the matching algorithm used to analyze a novel image in terms of a flexible model.

2.1 Formal specification

To write the linear object class model mathematically, we must first introduce some notation, which we summarize from [Jones and Poggio, 1996]. 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. Here we are only considering grey level images. 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* and interchangeably as the *shape vector* for the vectorized I_j. We define $I_j \circ S_j(x, y) = I_j(S_j(x, y))$. We also define

$$T_j(x, y) = I_j \circ S_j(x, y). \qquad (1)$$

T_j is the warping of image I_j onto the reference image I_0. So, $\{T_j\}$ is the set of shape-free prototype images, that is the *texture vectors*. They are shape free in the sense that their shape is the same as the shape of the reference image.

Using this notation, we are now ready to specify the model. We define the flexible model 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

$$I^{model} \circ (\sum_{i=0}^{N} c_i S_i) = \sum_{j=0}^{N} b_j T_j. \qquad (2)$$

The summation $\sum_{i=0}^{N} c_i S_i$ constrains the shape of every model image to be a linear combination of the prototype shapes. Similarly, the summation $\sum_{j=0}^{N} b_j T_j$ constrains the texture of every model image to be a linear combination of the prototype textures. Note that the coefficients for the shape and texture parts of the model are independent. This adds greater expressiveness to the model as it allows the shape of one prototype to be used along with the texture of another, for example.

To increase the flexibility of the model to handle translations, rotations, scaling and shearing, we add an affine transformation. The equation for the model images can now be written

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

where $A : \mathcal{R}^2 \to \mathcal{R}^2$ is an affine transformation parametrized by $\mathbf{p}$. Furthermore, we constrain $\sum_{i=0}^{N} c_i = 1$ in order to avoid redundancy in the parameters since the affine parameters allow for changes in scale. In the case of texture, the b_j's are not constrained to sum to 1.

Given values for $\mathbf{c}$, $\mathbf{b}$ and $\mathbf{p}$, the model image can be rendered by computing $(\hat{x}, \hat{y}) = A \circ \sum_{i=0}^{N} c_i S_i(x, y)$ and $g = \sum_{j=0}^{N} b_j T_j(x, y)$ for each (x, y) in the reference image. Then the $(\hat{x}, \hat{y})$ pixel is rendered by assigning $I^{model}(\hat{x}, \hat{y}) = g$, that is by warping the texture into the model shape.

2.2 Analysis by model matching

In the framework of this model, we can associate to each image in a class a shape vector and a texture vector. We refer to the process of analyzing an image in terms of its shape and texture vector as *vectorizing* an image.

A novel image of an object in a particular class is vectorized by matching a model of that class to the novel image. Matching means finding the best coefficients of the model so that the rendered model image most closely resembles the novel image. The general strategy for matching is to define an error function between the novel image and the current guess for the closest model image. This error is then minimized with respect to the model parameters (c_i, b_i, and p_i) by using a stochastic gradient descent algorithm. Following this strategy, we define the sum of squared differences error

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel}(x, y) - I^{model}(x, y)]^2 \tag{4}$$

where the sum is over all pixels (x, y) in the images, I^{novel} is the novel grey level image being matched and I^{model} is the current guess for the model grey level image. From equation 3 we see that in order to compute I^{model} we either have to invert the shape transformation $(A \circ \sum c_i S_i)$ or work in the coordinate system of the reference image. It is computationally more efficient to work in the coordinate system of the reference image. To do this we simply apply the shape transformation (given some estimated values for $\mathbf{c}$ and $\mathbf{p}$) to both I^{novel} and I^{model}. From equation 3, and with the notation

$$\overline{S} = (A \circ \sum_{i=0}^{N} c_i S_i). \tag{5}$$

we obtain the following error function (if we chose the L_2 norm)

$$E(\mathbf{c}, \mathbf{b}, \mathbf{p}) = \frac{1}{2} \sum_{x,y} [I^{novel} \circ \overline{S}(x, y) - \sum_{j=0}^{N} b_j T_j(x, y)]^2. \tag{6}$$

Minimizing the error yields the model image which best fits the novel image with respect to the L_2 norm. So far we have used the L_2 norm for convenience but other norms may be more appropriate (e.g. robust statistics).

In order to minimize the error function any minimization algorithm could be used. We have chosen to use the stochastic gradient descent algorithm [Viola, 1995] because it is fast and can escape from local minima.

2.3 Optical Flow

For some prototypes, the pixelwise correspondences from the reference image to the prototype can be found accurately by an optical flow algorithm. We have mostly used the multiresolution, laplacian-based, optical flow algorithm described in [Bergen and Hingorani, 1990].

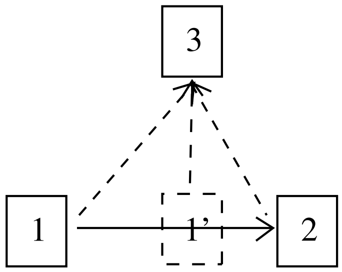

Figure 1: Given the flexible model provided by the combination of image 1 and image 2 (in correspondence), the goal is to find the correspondence between image 1 (or image 2) and the novel image 3. Our solution is to first find the linear combination of image 1 and image 2 that is closest to image 3 (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.

3 Bootstrapping the synthesis of a flexible model

Suppose that we have a flexible 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 flexible model can compute good correspondence to the new image then there is no need to add it to the flexible model since it will not increase its expressive power. If it can't, then the new prototype cannot be incorporated as such. A possible way out of this conundrum is to bootstrap the flexible model by using it together with an optical flow algorithm.

3.1 The basic recursive step: improving the flexible model with optical flow

Suppose that an existing flexible model is not powerful enough to match a new image and thereby find correspondence with it. The idea is first to find rough correspondences to the novel image using the (inadequate) flexible model object class and then to improve these correspondences by using an optical flow algorithm. This idea is illustrated in figure 1. In the figure, a model consisting of (vectorized) image 1 and image 2 (and the pixelwise correspondences between them) is first fit to image 3. Call the best fitting linear combination of images 1 and 2 image 1'. The correspondences are then improved by running an optical flow algorithm between the intermediate image 1' and image 3. Notice that this technique can be regarded as a class specific regularization of optical flow, which constrains appropriately the correspondence.

3.1.1 Example

An example of our basic step is shown in figure 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 model of faces to image (b), by using as a model a 20 prototype face images (with known correspondences). The model was matched to image (b) as described in section 2.2. 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 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

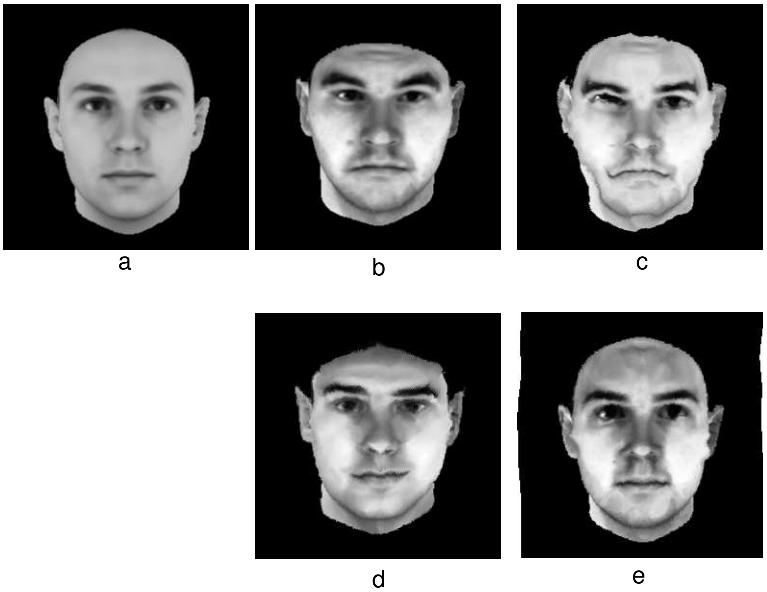

Figure 2: This figure shows the basic idea behind bootstrapping. Image (a) is the reference face. Image (b) is a prototype. Image (c) is the image resulting from backward warping the prototype onto the reference face using the correspondences found by an optical flow algorithm. Image (d) is the model 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.

correspondences are found by our basic recursive step relative to just using optical flow.

3.2 A bootstrapping algorithm for creating a flexible model

The idea of bootstrapping is to start from a small flexible model consisting of just 2 prototypical images and to increase its size (and representation 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 linear flexible 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 the matching process strongly. 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 flexible model. We could incorporate into the flexible model one new image at each timestep. Instead, we have implemented an equivalent algorithm in which the first step is to form a linear object 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 flexible model is selected and used to match each image. The optical flow algorithm estimates correspondence between the image and the approximation provided by the flexible 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 flexible 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.

3.2.1 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.
    Backwards 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 linear model is maximal {
  Perform a principal component analysis on S_i
    and separately on T_i.
  Select the first n principal components for the
    linear model.
  Approximate each I_i by the linear 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 S_i^{new}
  Backwards 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 linear model.
end **Until** }

4 Results

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

4.1 Face images

4.1.1 Data set

130 frontal images of caucasian faces were used in our experiments. The images were originally rendered for psychophysical experiments [Troje and Bülthoff, 1995] under ambient illumination conditions from a data base of three-dimensional human head models recorded with a laser scanner ($Cyberware^{TM}$). 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 grey-level images was 256-by-256 pixels and 8 bit.

Preprocessing: First the faces were segmented from the background and aligned roughly by automatically adjusting them to their two-dimensional centroid. The centroid was computed by evaluating separately the average of all x, y coordinates of the image pixels related to the face independent of their intensity value.

4.1.2 Evaluation

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

The step 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 gray value (0.3, whereas the 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 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 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 textures vectors are used in the flexible model. Then the correspondence field provided by matching with the flexible model is combined with the correspondence field obtained by the optical flow algorithm between the face image and its flexible model approximation. The backward warps using this correspondence fields are shown in the fourth column. The correspondence fields were iterated by slowly increasing the number of principal components used in the flexible model. After four iterations with 2, 10, 30 and 80 principal components, the correspondence fields between the reference face and all example images did not reveal any obvious errors (right column).

In a second experiment the same data set was split in a set of 100 training images and in a set of 30 test images. This experiment was used to document the improvements and changes within the linear model during the bootstrapping. The bootstrap 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.

After each bootstrapping step the 30 test images were approximated by the linear object model formed by the first 80 principal components obtained on the training set of 100 images. The Mahalanobis distance of each approximation

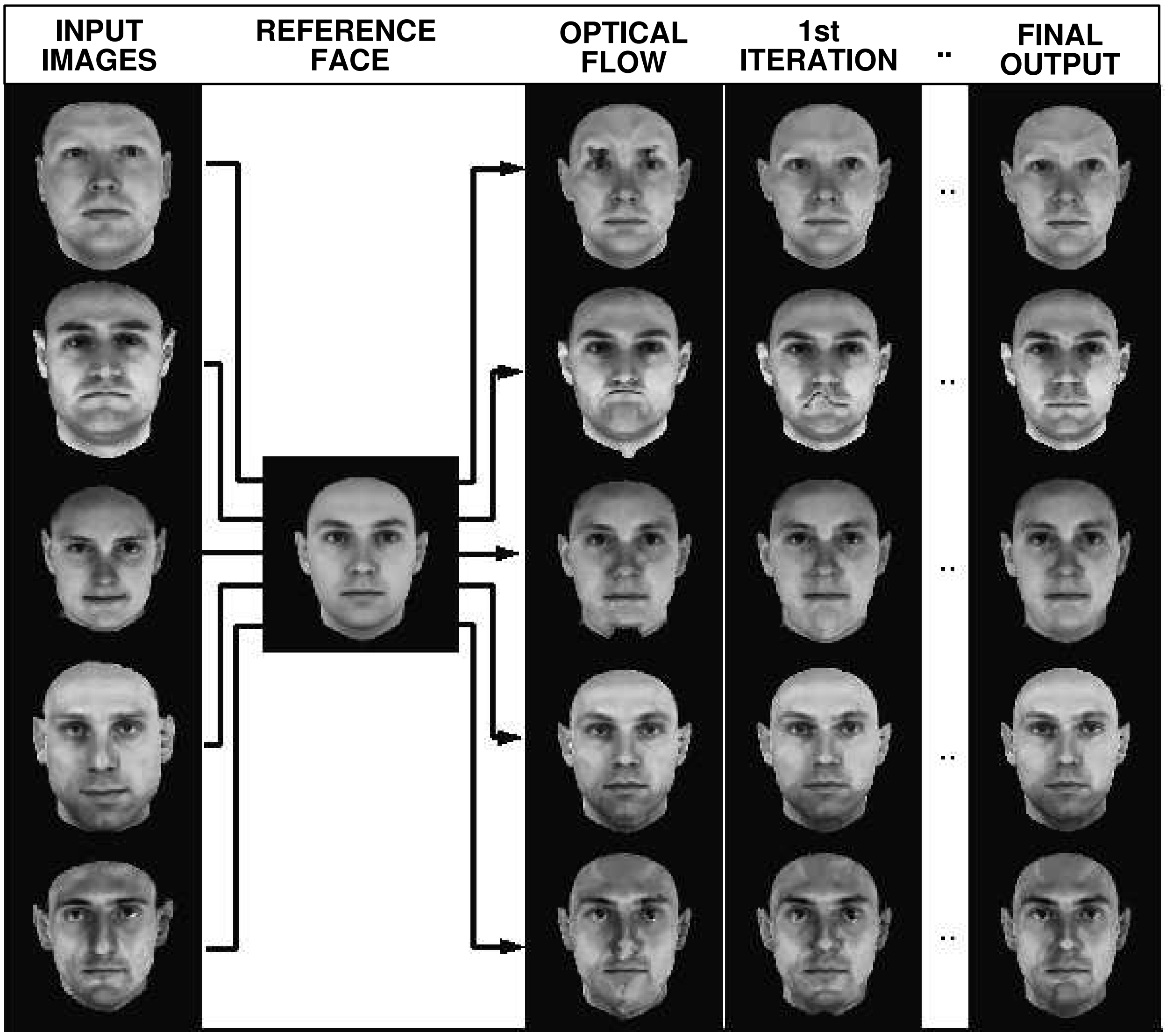

Figure 3: Five of the most difficult faces in our data set. The correspondence between face images (left column) and a reference face can be visualized by backward warping of the face images onto the reference image (three columns on the right). The correspondence obtained through the optical flow algorithm does not allow a correct mapping (center column). The first iteration with a linear flexible 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 4: 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 display 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.

to the average face image was computed. During the bootstrapping procedure the Mahalanobis distance decreased in the average over all test images.

4.2 Digits

4.2.1 Data set and Preprocessing

The images used in these experiments were from the US postal service database (262 per 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.

4.2.2 Evaluation

The bootstrapping algorithm was used for all 10 digits without modification. For each digit we obtained a linear model from the first 250 digits in the dataset. The reference image (average shape) is shown in the dashed boxes in figure 4. 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 4 shows the first 5 principal shape components of the final linear model.

The models obtained by the bootstrapping algorithm were used to match new digits which where not part of the training set. In figure 5 ten new images of the digit 3 are approximated with three different models of digits. Clearly the "3" model approximates well each of the new "3", 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.

Figure 5: 10 examples of the digit 3 are approximated by 3 different linear 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 and the first texture component.

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 flexible models from a set of prototypical examples. Notice that we have used multiresolution optical flow as one part of our bootstrapping algorithm. In principle other matching techniques could be used within our bootstrapping scheme.

References

[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, 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.

[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.

[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.

[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.

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

[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.

[Lines, 1996] Steve Lines. *The Photorealistic Synthesis of Novel Views from Example Images.* PhD thesis, MIT, 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.

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

[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.

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