MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

A.I. Memo No. 1431 November, 1993

C.B.C.L. Paper No. 80

Example Based Image Analysis and Synthesis

D. Beymer, A. Shashua and T. Poggio

Abstract

Image analysis and graphics synthesis can be achieved with learning techniques using directly image examples without physically-based, 3D models. We describe here novel techniques for the analysis and the synthesis of new grey-level (and color) images. With the first technique,

- the mapping from novel images to a vector of "pose" and "expression" parameters can be learned from a small set of example images using a function approximation technique that we call an *analysis network*;
- the inverse mapping from input "pose" and "expression" parameters to output grey-level images can be synthesized from a small set of example images and used to produce new images under real-time control using a similar learning network, called in this case a *synthesis network*.

This technique relies on (i) using a correspondence algorithm that matches corresponding pixels among pairs of grey-level images and effectively "vectorizes" them, and (ii) exploiting a class of multidimensional interpolation networks – Regularization Networks – that approximate the nonlinear mapping between the vector input and the vector output.

We also describe a second technique for analysis and synthesis of images, which can be formulated within the same theoretical framework and discuss the somewhat different implementation tradeoffs in terms of memory and computation.

As a third contribution, we introduce an approach for generating novel grey-level images from a single example image by learning the desired transformation from prototypical examples.

The three techniques described here have several applications in computer graphics, special effects, interactive multimedia and object recognition systems. The analysis network can be regarded as a passive and trainable *universal interface*, that is a control device which may be used as a generalized computer mouse, instead of "gloves", "body suits" and joy sticks. The synthesis network is an unconventional and novel approach to computer graphics. The techniques described here can be used for very low bandwidth teleconferencing and interactive simulations.

Copyright © Massachusetts Institute of Technology, 1993

This report describes research done within the Center for Biological and Computational Learning in the Department of Brain and Cognitive Sciences, and at the Artificial Intelligence Laboratory. This research was sponsored by grants from the Office of Naval Research under contracts N00014-91-J-1270 and N00014-92-J-1879. Support for CBCL is provided in part by a grant from the National Science Foundation under contract ASC-9217041. Support for the A.I. Laboratory's artificial intelligence research is provided by ONR contract N00014-91-J-4038. Tomaso Poggio is supported by the Uncas and Helen Whitaker Chair at the Whitaker College, Massachusetts Institute of Technology. David Beymer is supported by a Howard Hughes Doctoral Fellowship from the Hughes Aircraft Company. A. Shashua is supported by a McDonnell-Pew postdoctoral fellowship from the department of Brain and Cognitive Sciences.

1 Introduction

The classical *synthesis problem* of computer graphics is the problem of generating novel images corresponding to an appropriate set of "pose" control parameters. The inverse *analysis problem* – of estimating pose and expression parameters from images – is the classical problem of computer vision. In the last decade both fields of research have approached their respective problems of analysis and synthesis using intermediate physically-based models. Computer graphics has developed sophisticated 3D models and rendering techniques – effectively simulating the physics of rigid and non-rigid solid bodies and the physics of imaging. Part of computer vision has followed a parallel path: most object recognition algorithms use 3D object models and exploit the properties of geometrical and physical optics to match images to the data base of models.

In a series of recent papers we have advocated a different, simpler approach which bypasses the use of physical, 3D models (though it may be used with 3D views, see later). The approach has advantages and disadvantages. Relative to the traditional approach its main feature is to trade computation with memory. In one metaphor that aptly describes it, we speak of learning the mapping – from images to pose or from pose to images – from a set of examples, that is pairs of images and associated pose parameters. The learned mapping is then used to estimate the pose for a novel image of the same type, whereas the inverse mapping can provide a novel image for a new value of the control parameters.

1.1 Background of the General Approach

When we first introduced our technique based on Regularization Networks (Poggio and Girosi, 1990) to "solve" the analysis problem – from images to "pose" parameters – we demonstrated it only for artificial, very simple images (see Poggio and Edelman, 1990). On the synthesis problem we were able to demonstrate that our approach is effective for storing, interpolating among, and even extrapolating from professional quality, hand-drawn and colored illustrations (see Poggio and Brunelli, 1992; Librande, 1992). In the learning metaphor, the computer "learns" to draw a class of objects from a small set of examples provided by the artist, under user control. Each of the objects modeled (e.g., eyes, mouths, aircraft silhouettes, and the head of the cartoon character Garfield) were learned from a only a few digitized drawings. The sets of training images were digitized either directly, using a pen and tablet, or else from printed media using an ordinary flat-bed scanner. The $2D$ control points used in the internal representations of these objects were then obtained semi-automatically using recently developed algorithms for automatic contour tracing (Lines, pers. com.) and contour matching from a small number of manually matched points (Librande, 1992)). Poggio and Brunelli (1992) had suggested ways to extend the approach from line drawings to grey-level images and demonstrated the extension in a simple case of video images of a walking person. Their approach relied on manual correspondence established between a few key points in the example images (Chen *et al.* 1993 used the Poggio-Brunelli approach by exploiting the correspondence provided by the range data associated with their color images).

The more traditional approach to the analysis and especially the synthesis problem is based on explicit 3D models of the objects (Aizawa, Harashima and Saito, 1989; Nakaya *et al.* 1991; Choi *et al.* 1991; Li *et al.* 1993; Terzopoulos & Waters 1993, Oka *et al.* 1987; see Poggio, 1991). Parameters of a model of a generic face can be estimated from one or more images and then used to generate an accurate 3D model of the particular face which can then in turn be used to generate new images of the face. Such an approach is being developed mainly for very low bandwidth teleconferencing, an application that we are also going to describe later in this paper.

1.2 Contributions and plan of the paper

In this paper we show how to extend our previous approach from line drawing and computer simulated objects to grey-level and color images in a completely automatic way, describe some experiments with face images, discuss related theoretical results and mention various applications of the technique. The detailed plan of the paper is as follows. We introduce first our notation for computing and representing images as vectors. We then describe the *analysis network* and show examples of its performance. The *synthesis network* is introduced next with experimental examples of the synthesis of face images. Section 5 then describes several theoretical aspects of our approach. First, we summarize the Regularization Networks technique that we use for learning-from-examples; second, we introduce solutions to the correspondence problem between images that represent a key step for automatizing the whole process. Section 5 also describes how linear combination of images – in the appropriate representation – is intimately related to the Regularization Networks but can also be justified in independent ways. Section 6 describes a new technique for generating novel views from a single image of, say, a face without any 3D model but simply by "learning" the appropriate transformations from example views of a prototype face, extending a technique originally suggested by Poggio and Brunelli (see also Poggio, 1991 and Poggio and Vetter, 1992). Notice that although all our examples use face images, our techniques can be applied to images of any rigid or flexible object or collection of objects to deal with transformations of viewpoint, shape, color and texture. In the final section we discuss some of the many applications. The appendices provide details on theoretical and performance aspects.

2 Notation

Our approach (see for instance Poggio and Brunelli, 1992) is to represent images in terms of a vector of x, y of corresponding feature locations. These features can run the gamut from sparse features with semantic meaning, such as the corners of the eyes and mouth, to pixel level features that are defined by the local grey level structure of the image. In this paper we follow the latter approach. Images are vectorized by first choosing a

reference image and then finding pixel level correspondence between the two, using, for example, one of the optical flow or stereo algorithms developed in computer vision. Grey-level or color images are synthesized from the vectorized representation by using simple 2D image warping operations.

Vectorization of images at the pixel level requires the solution of a difficult correspondence problem. We found however (see also Shashua 1992b, Jones and Poggio, in preparation) that several correspondence algorithms perform satisfactorily for finding pixel level correspondence between grey level images of faces that are not too different. We will now introduce two operators for converting back and forth between the image representation and the pixel level vectorized representation.

Let img be an image that we want to convert to vector representation $\mathbf{y}$. In the pixelwise representation we are using, $\mathbf{y}$ represents pixelwise correspondences from some reference image img_{ref} to img. The **vect** operator computes these pixelwise correspondences, in our case using one of the optical flow algorithms

$$\mathbf{y} \leftarrow \mathbf{vect}(img, img_{ref}).$$

In our examples, img_{ref} might be a frontal, expressionless view of the person's face, and img will be a rotated view or a view with facial expression. Once images that represent different transformations from the reference image have been vectorized, new faces that combine these transformations can be synthesized using our learning techniques, which are of course equivalent to multivariate approximation schemes.

After computations in vectorized image space, one usually ends up with a vectorized image that one would like to view as an image. Borrowing from the computer graphics nomenclature, we call this process *rendering a vectorized image*. Because a vectorized image $\mathbf{y}$ simply characterizes feature geometry, to render it we need to sample the image texture from an example image, img_{tex}. The **rend** operator synthesizes an image with the feature geometry of $\mathbf{y}$ and the texture of img_{tex}

$$img \leftarrow \mathbf{rend}(\mathbf{y}, img_{tex}, \mathbf{y}_{tex}),$$

where $\mathbf{y}_{tex}$ is the vectorized version of img_{tex}. If img_{tex} is the same as the reference image img_{ref}, then implementing the **rend** operator is simple. In this case, a 2D warp is applied to img_{tex}, pushing the pixels in img_{tex} along the flow vectors in $\mathbf{y}$. However, in general case where img_{tex} and img_{ref} may be different, $\mathbf{y}$ must first be transformed to change the reference image to img_{tex}. How to perform this change in reference frame is explained in appendix D.

The **vect** and **rend** operators will be used in the following sections where we show the application of vectorized images in example-based image synthesis and analysis.

3 The Analysis Network: learning to estimate expression and pose parameters from grey level images

Consider the problem of learning to estimate the pose and expression parameters (including parameters associated with color or texture) of novel input images given a set of example images of a similar type. An analysis network which successfully estimates pose and expression parameters from images may be used as a trainable interface for a number of tasks such as to drive an example-based synthesis network. As we will discuss later, the "pose" parameters estimated by the network of this section could be used to encode information in teleconferencing applications and other image compression tasks. Another application is actor controlled animation, where an analysis module has examples of the (human) actor and works as a passive "body suit". The analysis parameters may be used to "direct" another person, who is represented by examples at the synthesis module.

The idea underlying the analysis network is to synthesize the mapping between a grey-level or color image and the associated "pose" parameters by using a suitable set of examples – pairs of images and associated pose parameters – and a Regularization Network for "learning" from them. The network, shown in figure 1 and explained more fully in a later section, "learns" the mapping from inputs to outputs using pairs of input/output vectors $(\mathbf{x}_i, \mathbf{y}_i)$ as examples. In this section, the outputs $\mathbf{y}_i$ are estimates of the location in pose/expression space of the input examples, which are vectorized images $\mathbf{x}_i$. When presented a new example image $\mathbf{x}$, the "trained" network will produce an estimate $\mathbf{y}$ of the pose/expression parameters. This example-based approach to estimating pose parameters for simple 3D objects was first suggested and demonstrated by Poggio and Edelman (1990, see figure 4). The input images were vectorized by establishing the correspondence between feature points. As mentioned earlier we have now used a correspondence technique that effectively vectorizes the example images – as well as the novel input image – by establishing pixelwise correspondence among them.

3.1 An example: estimating expression and pose for face images

We will describe here a specific experiment in which a Gaussian Radial Basis Function network is trained to estimate face rotation and degree of smiling from a set of four examples.

The first step is to vectorize the four examples by computing pixel-wise correspondence between them. The steps involved are:

1. Two anchor points such as the two eyes, in the case of faces, are matched manually or automatically.
2. The images are scaled and rotated using the image-plane transformation determined by the anchor points.
3. Pixelwise correspondence is found using an optical flow algorithm, such as the one described later in this paper.

Once the correspondence step is performed, each of the example images corresponds to a vector of dimensionality $q = 2 \times p \times p$, where $p \times p$ is the number of pixels in each image.

To build the network discussed in section 5.1, one must first choose a basis function G in Equation (1).

While different choices are possible, such as multiquadrics and splines, we have chosen the Gaussian function. In the next step, the "training" procedure, the network parameters of Equation (1) are estimated. In our Gaussian Radial Basis function network, the Gaussian σ's are estimated by taking 75% of the mean distance between all pairs of example inputs. The coefficients are then calculated – this is the learning stage – using the pseudoinverse solution of Equation (4).

At run time, the correspondence step must be performed again on the novel input image to establish the correspondence with the example images. The novel input is then equivalent to a vector of the same dimensionality q, which is processed by the network, providing an output which is an estimate of pose for the novel image.

Figure 2 shows four example images we used to train a network to estimate face rotation and degree of smiling. Image img_1 is used as reference by the correspondence process, so $\mathbf{x}_1 = \mathbf{0}$ and the other examples $\mathbf{x}_i$, $i = 2, 3, 4$, are represented by vectors that contain as components the coordinates of each pixel in img_1 relative to the corresponding pixel in img_i. In this case we assigned to our four examples the corners of the unit square as output values (in general we could have an arbitrary number of examples at arbitrary locations). In figure 3 we show some examples of rotation and expression estimates calculated by equation (1). As we will discuss in section 7 and as shown in figures 6 through 8, some of the applications of combining this analysis network with the synthesis network include low bandwidth teleconferencing and using a human actor to "direct" a cartoon character or another human.

4 The Synthesis Network: learning to generate a novel image as a function of input parameters

In this section we discuss an example-based technique for generating images of non-rigid 3D objects as a function of input parameters. This image synthesis task – a computer graphics task – is the inverse of the problem of image analysis – a computer vision problem – that we have discussed in the previous section.

In our approach, the input space is hand-crafted by the user to represent desired degrees of control over the images of the 3D object . After assigning example images of the object to positions in parameter space, novel (grey level or color) images are synthesized by using interpolation/approximation networks of the same type we have described for the analysis problem. In this paper we will demonstrate the technique on images of faces with input parameters such as pose, facial expression, and person identity, but other parameters may also be used.

To correctly interpolate the shape of two grey level images, we first find pixelwise correspondence as we had described above for the analysis case. Through the correspondence stage each image is associated with a vector of the x, y coordinates of corresponding pixels as in the Poggio-Brunelli technique. Once the example images are vectorized the learning stage can take place.

Following the Poggio-Brunelli and Librande-Poggio techniques, the user then

1. associates each example image to an appropriate position in the input space (for instance, pose and expression coordinates);
2. uses a multidimensional Regularization Network to synthesize the mapping between inputs and outputs.
3. displays, that is renders, the vectorized image as a grey-level or color image

The technique proposed here uses networks of the type described later (typically with $n = N$, that is Regularization Networks) as the preferred class of multidimensional approximation/interpolation technique. A specific choice of the regularization Green function provides a specific instance of regularization networks: examples are Gaussian Radial Basis Functions, Multiquadric Basis Functions and tensor products of piecewise linear splines, all of which were used in the Brunelli-Poggio and the Librande-Poggio implementations.

In summary, the synthesis network is trained on the examples $(\mathbf{y}_i, \mathbf{x}_i)$, that is the (vectorized) images and the corresponding pose vectors. After the learning stage the "trained" network can synthesize a new image $\mathbf{y}$ for each pose input $\mathbf{x}$ given to the network.

4.1 A first example: one-dimensional interpolation between two images

The simplest use of the technique is to interpolate or "morph" between two face images. This is one-dimensional interpolation in the sense that there is just one parameter – which can be thought of as *time* or *degree of morph* or *(fuzzy) identity index* – that controls the relative contribution of just two example images: thus the approximation network of figure 1 has in this case only one input and is "trained" using just two examples (an therefore has only two "hidden" units). In this example we use the pixelwise correspondences to interpolate with linear splines – that is using a regularization network with $G(x) = |x|$, see later – the image of the face. A simple cross dissolve is used to interpolate the grey levels.

Let img_1 and img_2 be the two face images to be interpolated and let x be the (input) interpolation parameter. Three examples of 1D interpolation are shown in figure 4. In the first and last the input parameter is similarity to the first face: the figure shows a "morph" between two different people. The last case is an amusing failure case where the optical flow algorithm fails to find correct correspondence. The second demonstration interpolates a change in expression and slight head rotation, given two examples of the same face. Thus, the simplest special cases of our interpolation technique is morphing between two similar images. Even just for morphing our technique is considerably more automatic than others. Notice that the use of optical flow algorithms to perform automatically one dimensional interpolation between two images is also discussed in (Bergen-Hingorani, 1990) as a method for doing frame rate conversion.

Let img_1 and img_2 be the two face images to interpolate and let x be the interpolation parameter. We then

compute the interpolated image $img_{inter}(x)$ by

$$\begin{aligned} \mathbf{y} &\leftarrow \mathbf{vect}(img_2, img_1) \\ img_{int}(x) &\leftarrow (1-x)\,\mathbf{rend}(x\,\mathbf{y}, img_1, \mathbf{0}) + \\ &\quad x\,\mathbf{rend}(x\,\mathbf{y}, img_2, \mathbf{y}) \end{aligned}$$

Note that img_1 is reproduced exactly when $x = 0$; img_2, when $x = 1$.

4.2 Multidimensional synthesis of face images

Our main result here is the first demonstration that approximation/interpolation techniques can be used to synthesize dense grey level images for a multidimensional input space.

Figure 2 shows the four examples of a face and their pose parameters in a 2-dimensional input space, the dimensions of which are designed to represent the variability present amongst the examples. The assignment of each image to a location in the input space was decided by the user. The first step is to compute correspondence of images img_2, img_3, and img_4 with respect to reference image img_1. We then use bilinear interpolation among four examples to synthesize novel images, each corresponding to a value of the 2-dimensional pose vector. The process is equivalent to using tensor product of piecewise linear splines in two dimensions (which is a regularization network with Green function $G(x, y) = |x||y|$, see Appendix A). In our demonstration, the network generates new vector images for "any" desired value of rotation and expression, the coordinates of each pixel corresponding to the coordinates of so-called output "control" point in the Poggio-Brunelli-Librande description. We have rendered the vector which is the output of the network into a grey-value image by bilinearly interpolating the grey level values between corresponding pixels of the example images. In figure 5, the example images, which are bordered in black, have been placed at the corners of the unit square in (x, y) rotation/expression space. All the other images are synthesized by the network described above.

To synthesize an image at coordinates (x, y), we first vectorize the examples using img_1 as a reference

$$\mathbf{y}_i \leftarrow \mathbf{vect}(img_i, img_1),$$

where $\mathbf{y}_1$ will be assigned the zero vector. Then we use bilinear interpolation to compute the new face geometry

$$\mathbf{y}_{int}(x, y) \leftarrow \sum_{i=1}^{4} b_i(x, y)\mathbf{y}_i,$$

where $b_i(x, y)$ is the bilinear coefficient of example i (see Appendix A)

$$\begin{aligned} b_1 &= (1-x)(1-y) & b_2 &= x(1-y) \\ b_3 &= (1-x)y & b_4 &= xy. \end{aligned}$$

Finally, we render the interpolated image img_{int} using contributions from the facial texture of all examples

$$img_{int}(x, y) \leftarrow \sum_{i=1}^{4} b_i(x, y)\mathbf{rend}(\mathbf{y}_{int}, img_i, \mathbf{y}_i).$$

Thus, given a set of example views, we can synthesize any intermediate view by using interpolation. In the examples of this section we have used a two dimensional input space. As it is clear from the theory (see examples in Librande, 1992), the input space may have any dimension, provided there is a sufficient number of examples. We have successfully generated face images in a three dimensional input space of pose-expression parameters.

Together, the analysis and synthesis networks can be used for low-bandwidth teleconferencing, effectively using the analysis parameters – that is the output of the analysis network – as a model-based image code. Before the teleconferencing session, the transmitter sends the receiver the information it needs to initialize the synthesis network – the example images and their assigned locations in pose-expression space. During the teleconferencing session, the transmitter feeds each image frame through the analysis network, transmits only the analysis parameters, and then the receiver uses the synthesis network to reconstruct the image. Figure 6 demonstrates the process for the set of example images in figure 2 and our 2D rotation-smiling parameter space (x, y).

While figure 6 uses the same set of images at both the transmitter and receiver, in general we can train the synthesis network with a *different* set of example images. For example, a stored set of examples taken while "dressed up" can used to always put a person's "best face forward" at the receiving end, even if the person is informally dressed at the transmitter. To make this work, the only requirement is that the two example sets are parameterized by the same control space. In another example, by putting a set of example images of *another* person at the receiver, the analysis/synthesis network can be used to "direct" this new person. For instance, in figure 8, the receiver is trained with the example set of figure 7, enabling the person at the transmitter to direct the person in this new example set. If we use a set of cartoon character images at the receiver, our approach for teleconferencing also becomes a method for performance-driven animation.

5 Theoretical background

The techniques described in this paper are, to a large extent, the result of an integration of two existing sets of methods:

1. How to establish automatically pixel-wise correspondence between grey level images.
2. How to approximate maps from a vector space to a vector field.

The correspondence problem is typically seen in the field of motion measurement, stereopsis and structure from motion. Recent advances in visual recognition in terms of interpolation networks (Poggio & Girosi 1990; Edelman & Poggio 1990, Brunelli and Poggio, 1991), alignment techniques (cf. Ullman 1986; Huttenlocher & Ullman 1990; Ullman & Basri 1991), and methods using geometric invariants across multiple views (cf. Mundy *et. al* 1992; Weinshall 1993; Shashua 1993b), have also put new emphasis on achieving correspondence between model images stored in memory, prior to the actual

recognition of novel instances of objects. We propose to use correspondence not only in image analysis but also in computer graphics, by providing the ability to fully register example images of an object for later synthesis of new examples. The correspondence techniques we used will be described in Section 5.2.

The second set of methods uses non-linear interpolation (or approximation) from a vector space to a vector field using generalized regularization networks (Poggio & Girosi, 1990). The basic idea, proposed by Poggio and Edelman (1990), is to view the problem of estimating pose from an object image as a non-linear mapping between 2D images (examples) and the corresponding control parameters, such as pose and facial expressions. Poggio & Brunelli (1992) suggested that the synthesis problem could be viewed in terms of the associate, inverse mapping.

The connection between this and correspondence is that the 2D example images must be fully registered with each other prior to the interpolation process. We will discuss regularization networks in Section 5.1.

In Section 5.3 we point out that regularization networks can be used to map directly vector fields onto vector fields. In other words, instead of mapping 2D images onto control parameters and then into images by an analysis-synthesis sequence of steps, we map 2D images directly onto other 2D images. We show that this may be useful in teleconferencing applications and propose connections between that mapping and recent techniques in visual recognition applied to non-rigid objects (Ullman &, Basri 1991; Poggio, 1990) and certain classes of objects called "Linear Classes" (Poggio & Vetter, 1992).

5.1 Approximation of vector fields through regularization networks

Consider the problem of approximating a vector field $\mathbf{y}(\mathbf{x})$ from a set of sparse data – the examples, which are pairs $(\mathbf{y}_i, \mathbf{x}_i)$, $i = 1 \cdots N$. Choose a *Regularization Network* or a *Generalized Regularization Network* (see Girosi, Jones and Poggio, 1993) as the approximation scheme, that is a network with one "hidden" layer and linear output units (see for instance Poggio and Girosi, 1989). Consider the case of N examples, $n \leq N$ centers, input dimensionality d and output dimensionality q. Then the approximation is

$$\mathbf{y}(\mathbf{x}) = \sum_{i=1}^{n} \mathbf{c}_i G(\mathbf{x} - \mathbf{x}_i) \tag{1}$$

with G being the chosen Green function, which may be a radial basis function, like the Gaussian, or a spline, like some tensor product spline. The equation, which is equivalent to the network of figure 1, can be rewritten in matrix notation as

$$\mathbf{y}(\mathbf{x}) = \mathbf{C}\mathbf{g}(\mathbf{x}) \tag{2}$$

where $\mathbf{g}$ is the vector with elements $g_i = G(\mathbf{x} - \mathbf{x}_i)$. Let us define as $\mathbf{G}$ the matrix of the chosen Green function evaluated at the examples, that is the matrix with elements $G_{i,j} = G(\mathbf{x}_i - \mathbf{x}_j)$. Then the "weights" $\mathbf{c}$ are "learned" from the examples by solving

$$\mathbf{Y} = \mathbf{C}\mathbf{G}. \tag{3}$$

where $\mathbf{Y}$ is defined as the matrix in which column l is the example $\mathbf{y}_l$. $\mathbf{C}$ is defined as the matrix in which row m is the vector $\mathbf{c}_m$. This means that $\mathbf{x}$ is a $d \times 1$ matrix, $\mathbf{C}$ is a $q \times n$ matrix, $\mathbf{Y}$ is a $q \times N$ matrix and $\mathbf{G}$ is a $n \times N$ matrix. Then the set of weights C is given by

$$\mathbf{C} = \mathbf{Y}\mathbf{G}^{+}. \tag{4}$$

5.2 The Correspondence Problem for Grey-level Images

While the general approximation scheme outlined above can be used to learn a variety of different mappings, in this paper we discuss mappings between images of objects and their parameters, such as pose or expression, and between images onto themselves (notice that all the techniques of this paper generalize directly from 2D views to 3D views – that is 3D models – of an object). In image analysis, the approximation techniques can be used to map input images $\mathbf{x}$ into output pose and expression vectors $\mathbf{y}$. They can also be used to synthesize the map from input pose vectors $\mathbf{x}$ to output images $\mathbf{y}$ for image synthesis. The correspondence problem is the same in both cases. Our approach critically relies on the ability to obtain correspondence between two example images.

The choice of features (components of the vectors $\mathbf{y}$) is crucial both in obtaining a smooth mapping for the regularization network, and for the type of correspondence method that is to be used. One of the most natural choices – which satisfies the requirements of smoothness of the input-output mapping – is a vectorization of the image in terms of the (x, y) locations of image features. For instance, the image of a face with n features, such as the "inner corner of the left eye", will be represented by a vector of length $2n$, which is simply the concatenation of all the features' x coordinates followed by the y coordinates. This is a representation of shape; color or grey level information can be tagged to the feature points as auxiliary information (and additional components of the $\mathbf{y}$ vector). It is important to stress that many other representations are possible and have been used. In any case a sparse x, y representation, originally suggested for object recognition (see for instance Ullman and Basri, 1989) was also used by Poggio and Brunelli (1992) for synthesis of real images in which features (in the order of 20 or so) were located and brought into correspondence manually. The same representation has been recently applied to generating images of hand-drawn colored illustrations (Librande, 1992). The feature points are much denser, on the order of hundreds, and are placed along the contours of objects. The correspondence problem is alleviated somewhat by using algorithms for automatic contour tracing and matching – only correspondences of a few points per contour need to be specified. Texture mapping was used by Poggio and Brunelli as well as by Librande to render the image from the feature points.

In this paper we suggest using the densest possible representation – one feature per pixel, originally suggested for visual recognition (Shashua, 1991). This

avoids the need of using texture mapping techniques at the expense of solving a difficult correspondence problem. Morphing in computer graphics (Beier and Neely, 1992) faces a similar correspondence problem that is typically solved with time-consuming manual intervention.

We have discovered that standard optical flow algorithms, preceded by certain normalization steps, can do a good job at automatically computing dense pixel-wise correspondences with the images we have used so far. This ability to automatically compute correspondences in grey level images makes practical the "vectorization" of a grey level image, that is the computation of a vector $\mathbf{y}$ associated to it and which describes the position of each pixel relative to a chosen reference image. There are many similar optical flow algorithms that perform in similar ways. We describe one which we have used in the experiments described here. We also refer in the rest of the paper to face images, though the technique should work for a variety of 3D objects.

The coarse-to-fine gradient-based optical flow algorithm used in the examples of this paper follows (Lucas & Kanade 1981; Bergen & Adelson 1987; Bergen & Hingorani 1990) and is applied after a normalization stage for compensating for certain image plane transformations. In our face images we chose manually the eye centers at the center of the iris (Brunelli and Poggio 1992, Yuille *et al.* 1989, Stringa 1991, Beymer 1993, describe automatic techniques for finding the eyes). This corrects for differences in scale, translation, and image-plane rotation of the two face images. An important byproduct of this normalization step is that the two images are brought closer together, making the remaining correspondence problem slightly easier. Once the images are in rough geometric alignment, the remaining displacement between the two frames is found using the a standard gradient-based optical flow method (we have also used other algorithms that do not use gradient computations with similar or better results).

If we assume the remaining displacement between the two images is sufficiently small, and that the brightness value of corresponding points does not change much, then we have the following equation, known as the "constant brightness equation" (Horn & Schunk, 1981):

$$\nabla I \cdot \boldsymbol{v} + I_t = 0, \tag{5}$$

where ∇I is the gradient at point p, and I_t is the temporal derivative at p, and $\boldsymbol{v}$ is the displacement vector viewed as a velocity vector. This equation describes a linear approximation to the change of image grey-values at p due to image motion and is obtained by a first order approximation of a Taylor series expansion of $I(x+dx, y+dy, t+dt) = I(x,y,t)$. Since the solution of this problem is underdetermined at a point – one equation for two unknowns – the additional constraint of flow smoothness is made and the solution is computed over small neighborhoods. A coarse-to-fine strategy, currently implemented by Laplacian Pyramids (Burt & Adelson, 1983), estimates displacements at the coarser levels of the pyramid and refines the estimates as finer levels are processed. This enables fast processing while still being able to find large displacements, for a large displacement at a high resolution is a small displacement at lower resolutions. More details of this particular implementation can be found in Bergen & Hingorani (1990).

5.3 Mapping between Vector Fields: The Case of Linear Combination

We have described techniques for mapping fully registered 2D images onto corresponding control parameters, such as pose and expressions of a face. The mapping can be later on used in the synthesis module to generate new images, say of a face, based on input of novel parameter settings, such as novel poses and novel facial expressions. Here we suggest an alternative technique, with potential applications to teleconferencing (see later this section, Section 7 and Appendices B and C), in which 2D images are mapped directly onto 2D images.

Continuing the derivations of Section 5.1, we note (see also Girosi, Jones and Poggio, 1993) that the vector field $\mathbf{y}$ is approximated by the network as the linear combination of the example fields $\mathbf{y}_l$, that is

$$\mathbf{y}(\mathbf{x}) = \mathbf{Y}\mathbf{G}^+\mathbf{g}(\mathbf{x}) \tag{6}$$

which can be rewritten as

$$\mathbf{y}(\mathbf{x}) = \sum_{l=1}^{N} b_l(\mathbf{x})\mathbf{y}_l \tag{7}$$

where the b_l depend on the chosen G, according to

$$\mathbf{b}(\mathbf{x}) = \mathbf{G}^+\mathbf{g}(\mathbf{x}). \tag{8}$$

This derivation readily suggests that instead of estimating $\mathbf{x}$ corresponding to the "novel" image $\mathbf{y}^d$, we can estimate the "best" coefficients $\mathbf{b}_l$ such that

$$\mathbf{y}^d = \sum_{l=1}^{N} b_l\mathbf{y}_l, \tag{9}$$

and then use the estimated b_l – that is $\mathbf{b} = \mathbf{Y}^+\mathbf{y}^d$ for reconstructing $\mathbf{y}^d$. Therefore, in cases where we are not interested in the vector $\boldsymbol{x}$ of control parameters, we can reconstruct the novel image by simply combining the available examples.

The natural application is teleconferencing, where the novel image is given at the sender site and the receiver only requires the coefficients b_l for combining the example images. In figure 10, we use the example set of figure 9 and equation (9) to write each novel image on the left directly in terms of the three vectorized example images $\mathbf{y}_2$, $\mathbf{y}_3$, and $\mathbf{y}_4$. (Since img_1 is the reference image, $\mathbf{y}_1 = \mathbf{0}$, and it makes no contribution to equation (9).) The reconstructed image, shown on the right, is synthesized by recalculating the vectorized image $\mathbf{y}_{recon}$ (using equation (9)) and then rendering it using the image texture from img_1 ($img_{recon} \leftarrow \mathbf{rend}(\mathbf{y}_{recon}, img_1, \mathbf{0})$). In the figure we show two sets of b_l coefficients since the x and y components of $\mathbf{y}^d$ are represented separately using equation (9).

This approach to teleconferencing can also be used to "direct" another person when the receiver has another example set, as we demonstrate in figure 11. Compared

to using the analysis/synthesis approach, however, there are tighter constraints coupling the transmitter and receiver examples. Since the coefficients are tied to specfic examples instead of an abstract pose-expression parameter, the examples in both the transmitter and receiver sets must be at the same locations in pose-expression space. In this sense the analysis/synthesis network is more flexible, as the abstraction provided by the input space parameterization allows transmitter and receiver examples to fall at different locations. Section 7 and Appendices B and C present more details on our two techniques for teleconferencing.

Equation 7 provides the connection between the two techniques, and we have the following result:

For any choice of the regularization network (even a Generalized Regularization Network, see Girosi, Jones and Poggio, 1993) and any choice of the Green function – including Green functions corresponding to additive splines and tensor product splines – the estimated output (vector) image is always a linear combination of example (vector) images with coefficients **b** *that depend (nonlinearly) on the input value.*

The result is valid for all networks of the type shown in figure 1, provided that a L^2 criterion is used for training.

This technique is intimately connected with two concepts in visual recognition: one is the linear combination of views of Ullman & Basri (1991), and the other is the linear class of objects proposed by Poggio & Vetter (1992). These connections, which we discuss below, are a particular case of the general result described above, but on the other hand are sharp (under certain assumptions), rather than being an approximation.

The following arguments suggest that approximating a 2D image as the linear combination of a small set of "example" images is exact for a special class of non-rigid objects provided that the views are represented as vectors of x, y coordinates of visible "features" and image projection is orthographic.

As a corollary of Ullman & Basri's linear combination result one can easily show the following two related, but different, results. First, consider objects composed of k "parts", and consider the set of images of such an object — which is defined as a result of having each part undergo an arbitrary 3D affine transformation, followed by a projection onto the image plane. The space of all orthographic images of such an object form a $3k$ dimensional space, and therefore, can be spanned by linearly combining $3k$ images of the object (x and y components separately). Note that segmentation, or decomposition of the image into parts, is not required — only that we have correspondence between the $3k$ example images. This observation was made by Basri (1990) in the context of articulated objects (objects composed of links, like scissors), but may be also valid to more natural objects.

For the second result, consider a collection of k different objects (presumably coming from the same class of objects, like the class of faces, but this is not theoretically necessary), and consider the class of objects that can be obtained by linearly combining the corresponding coordinates (in 3D) across the k objects (Vetter and Poggio 1992, refer to this as the "Linear Class of Objects"). For example, the linear class of a square and a triangle includes a trapezoid as a particular case (note that a trapezoid cannot be obtained by means of an orthographic view of a square). Next consider $3k$ "example" images of these objects — three distinct orthographic views per object. It is a simple matter to show that the set of all views of the linear class (views of the k objects, and all views of all objects generated by linear combinations of those objects) are spanned by the set of $3k$ examples.

These results express two different aspects of the same method of approximating an image by combining a small set of example images. In one aspect, we view these images as views of a non-rigid object that is assumed to be composed of a finite (small) number of rigid components. Alternatively, we may view the images as views of different objects (presumably related to each other, such as faces). For example, we may view a face as a non-rigid object, view facial expressions as a linear class of example images of the same face, or view the class of faces as a linear class. In all cases we use the same method of approximation (which is exact when the conditions above hold). Finally we note that the problem of correspondence between the example images receives different interpretations depending on how we view the transformation space of objects. For example, in the case we adopt the view of having a non-rigid object, then the problem of correspondence is in theory well defined (just as in the rigid case); in case we assume a linear class of objects, then correspondence is theoretically ill-defined because we have no longer the projections of a fixed set of points in space. In practice, however, we observed that obtaining correspondence, using the methods described in Section 5.2, between images of the same face undergoing facial expressions is relatively easy. There are many cases however in which the correspondence algorithm fails, sometime with funny effects (see figure 4). Obtaining correspondence between images of different faces is also possible but in several cases may require more interactive methods than the one used in our implementation.

On a different level, notice that the argument based on approximation does not say anything about the number of examples needed for a satisfactory performance, whereas the other argument provides such an estimate, at least in principle (an analysis of the effect of noise is still lacking). The approximation argument on the other hand does not depend on the particular view representation, as long as the mapping between the pose input and the view output is smooth (see Poggio and Brunelli, 1992 for the argument and an example). Thus not only x, y coordinates can be used but other attributes of the feature points such as color and slant as well as angles or distance between feature points and even non geometric attributes such as the overall luminance of the image. Orthographic as well as perspective projection can be dealt with. The linear combination results, on the other hand, depend on orthographic projection and especially

critically on the specific view representation, based on x, y coordinates of visible feature points.

6 Networks that generate novel grey-level images from a single example by learning class-specific transformations

Given a set of example images of an object, such as a face, we have discussed how to synthesize new images of that face for different poses and expressions, given a sufficient number of examples of that specific face. Suppose now that only one image of a specific face is available. How can we synthesize novel views of that particular face from just one "model" view? A natural idea is to use a collection of example views of another person p (or perhaps the average of a small set of people) as a prototype for representing generic face transformations. If we bring the image of a face img_{new} into correspondence with the closest prototype image, then we may be able to synthesize images of the new person as we would with the prototype from just *one view* of the new person. In general, we want to generate from one 2D view of a 3D object other views, exploiting knowledge of views of other objects of the same class.

This idea of generating "virtual" views of an object by using class-specific knowledge has been discussed before in (Poggio, 1991, see also Poggio and Vetter, 1992). In our preliminary work we have applied the method used by Poggio and Brunelli (1992) to generate a walking animation sequence of Carla from just one image of Carla and a sequence of frames of the walking "prototype" Roberto. A view $\mathbf{y}$ of Carla or Roberto is a vector of x, y coordinates of manually chosen corresponding points near the joints and limb extremities. Suppose that we have two views of the prototype, $\mathbf{y}_{ref}$ and $\mathbf{y}_p$, the second of which is a transformed version of the first, for instance a rotated view of $\mathbf{y}_{ref}$. This prototype transformation can be simply represented by the vector difference

$$\mathbf{\Delta y}_p = \mathbf{y}_p - \mathbf{y}_{ref}. \tag{10}$$

Now let $\mathbf{y}_{nov}$ be a view of object nov that appears in the same pose as $\mathbf{y}_{ref}$. Poggio and Brunelli then generate the transformed view of object nov, $\mathbf{y}_{p+nov}$, by

$$\mathbf{y}_{p+nov} = \mathbf{y}_{nov} + \mathbf{\Delta y}_p \tag{11}$$

A new grey level image is then generated by texture mapping from the original image of nov.

We have generated "virtual" views of a face by using the same technique applied to individual pixels rather than sparse control points. In figure 12, we have used the optical flow between a pair of prototype examples, img_{ref} and img_p, to "learn" face rotation and change of expression. Once a new face image img_{nov}, is brought into correspondence with img_{ref}, we can apply the prototype transformation to the new image, synthesizing img_{p+nov}.

To generate a virtual view using our notation, prototype transformations, as represented by a vectorized prototype example $\mathbf{y}_p$, need to be mapped onto a novel face. If the novel face is vectorized using the prototype reference image, producing $\mathbf{y}_{nov}$, this can be done by simply summing the two vectorized representations. Consider figure 12, where the prototype rotation transformation is represented by the vectorized img_p using img_{ref} as reference

$$\mathbf{y}_p \leftarrow \mathbf{vect}(img_p, img_{ref}).$$

Vectorizing the novel face image img_{nov} using img_{ref} as reference produces a "identity changing" transform, specifying how to deform the face geometry of the prototype into the novel face

$$\mathbf{y}_{nov} \leftarrow \mathbf{vect}(img_{nov}, img_{ref}).$$

The composition of the rotation transformation $\mathbf{y}_p$ and the identity-changing transformation $\mathbf{y}_{nov}$ is simply the sum

$$\mathbf{y}_{p+nov} \leftarrow \mathbf{y}_p + \mathbf{y}_{nov}.$$

We can interpret the last equation in terms of altering the geometry of the reference image by moving pixels around. The composite transform moves a pixel first by the prototype transform and then by the identity changing transform. Finally, we can render the composite geometry using the facial texture from the novel image

$$img_{p+nov} \leftarrow \mathbf{rend}(\mathbf{y}_{p+nov}, img_{nov}, \mathbf{y}_{nov}).$$

This produces the image shown in the lower right of figure 12.

An interesting idea to be explored in the near future is the use of the alternative technique proposed by Poggio and Vetter (1992) for generating virtual views based on the concept of linear classes, mentioned earlier.

In any case, a sufficient number of prototype transformations – which may involve shape, color, texture and other image attributes by using the appropriate features in the vectorized representation of images – should in many cases allow the generation of more than one virtual view from a single "real" view. In this way a few prototypical views may be used to generate corresponding virtual views starting from just a single image. The resulting set of *virtual* examples can then be used to train a network for either synthesis or analysis (an alternative way to achieve the same result is to use the virtual views generated by a prototypical network such a the one of figure 5 to transform the novel face appropriately). The applications of this technique are many. One of them is object recognition from a single model image. Preliminary work by one of us (Beymer, 1993, thesis proposal) has shown the feasibility of the basic principle in the case of face recognition.

7 Discussion

Two key ingredients played a major role in this paper. First, we proposed an approach that can be characterized as *memory-based* or *learning-from-examples* for both the analysis problem and for the synthesis problem of computer-graphics – instead of the classical physics-based and 3D model-based approach. Second, we have introduced optical-flow as a tool for our methods, to achieve full-correspondence between example images. In

addition, we have established a theoretical and practical connection between multi-dimensional approximation techniques and linear mappings between vector fields (linear combination of views, linear class of objects). These ingredients together were the basis for the analysis and the synthesis networks and for several applications which we have already mentioned and will outline in this section.

The problem of estimating pose parameters from the image of a 3D object is usually solved in the realm of computer vision by using appropriate 3D models or even physics-based models. For instance, pose and expression of faces can be estimated by fitting a generic model of a face to the image of a specific face (Aizawa, Harashima and Saito, 1989; Poggio, 1991). Instead, we use a short cut in the sense that we use a set of images as examples and completely avoid the problem of a physics-based model. Similarly, the classical synthesis problem of computer graphics is the problem of generating novel images corresponding to an appropriate set of control parameters such as pose, illumination and the expression of a face. The traditional approach in computer graphics involves 3D modeling and rendering techniques which effectively simulate the physics of imaging as well as rigid and non-rigid body motions. For example, the interpretation and synthesis of facial expressions would classically follow from an understanding of facial muscle groups in terms of their role in generating specific expressions and their interdependency (cf. Ekman & Friesen 1978; Ekman 1992; Essa 1993). Our alternative approach to graphics may be effective in the sense that it may provide a short-cut by trading the modeling component and computation with memory and correspondence.

Our memory-based approach to the analysis and synthesis problems may be described as a scheme that learns from examples. It may reflect the organization of human vision for object recognition (see for instance Poggio, 1990 and Poggio and Hurlbert, 1993). One may draw the same connection in the special case of interpreting facial expressions. Although the understanding of group muscles is the physically correct thing to do, it may not be the way humans interpret facial expressions.

We have seen (Section 5.3) that a regularization network can be used to map vector fields onto vector fields. Instead of using an analysis and a synthesis network we can map images onto other images directly — by exploiting the observation that the estimated output of a synthesis network is always a linear combination of the example images (vectorized). This observation makes a direct connection to two other results in the context of visual recognition: the linear combination of views (Ullman & Basri, 1991) and the linear class of objects (Vetter & Poggio, 1992). Consequently, the linear mapping between vector fields is *exact* under certain conditions that include orthographic projection, and either the group of transformations is piece-wise 3D affine, or the example set is spanning the linear class of objects represented by this set. From a practical standpoint, the mapping between vector fields introduces a scheme for synthesizing new images that does not require specifying control parameters. We introduced this scheme in the context of teleconferencing where the sender solves and sends the coefficients of the linear combination required to span the novel image from the example set of images (as shown in Appendix B.1, a combination of eigenfunctions of the example set can be used instead).

The results and schemes presented in this paper may have several related applications. We have discussed mainly two basic applications: analysis for gesture or pose estimation, synthesis for computer graphics animation and teleconferencing. Other potential applications lie in the realm of virtual reality, training systems, man-machine interfaces (including video speech synthesis), computer-aided design and animation and object recognition. For instance, the approach demonstrated with our analysis network may be developed into a trainable and rather universal control interface playing the role of a computer mouse or a body suit or a computer glove, depending on the available examples and sensors (Jones and Poggio, in preparation). In another example, if the synthesis module is trained on examples of a cartoon character or of a *different* person (see the example set of figure 7) from the analysis network, then one actor's face can be used to "direct" a cartoon character or the face of another person, as shown in figure 8. Variations on this theme may have applications in various flavors of virtual reality, videogames, and special effects.

References

[1] K. Aizawa, H. Harashima, and T. Saito. Model-based analysis synthesis image coding (MBASIC) system for a person's face. *Signal Processing: Image Communication*, 1:139–152, 1989.

[2] R. Basri. *The recognition of 3-D solid objects from 2-D images.* PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1990.

[3] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. In *SIGGRAPH '92 Proceedings*, pages 35–42, Chicago, IL, 1992.

[4] J.R. Bergen and E.H. Adelson. Hierarchical, computationally efficient motion estimation algorithm. *Journal of the Optical Society of America*, 4:35, 1987.

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

[6] David J. Beymer. Face recognition under varying pose. A.I. Memo No. 1461, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[7] R. Brunelli and T. Poggio. Hyberbf networks for real object recognition. In *Proceedings IJCAI*, Sydney, Australia, 1991.

[8] P. J. Burt and E. H. Adelson. The laplacian pyramid as a compact image code. *IEEE Transactions on Communication*, 31:532–540, 1983.

[9] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In *SIGGRAPH '93*

Proceedings, pages 279–288, Anaheim, CA, August 1993.

[10] Chang Seok Choi, Hiroshi Harashima, and Tsuyosi Takebe. Analysis and synthesis of facial expressions in knowledge-based coding of facial image sequences. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2737–2740, 1991.

[11] P. Ekman. Facial expression of emotion: An old controversy and new findings. *Phil. Transactions: Bio. Sci. (series B)*, 335:63–69, 1992.

[12] P. Ekman and W.V. Friesen. *Facial action coding system*. Consulting Psychologists Press Inc., Palo Alto, CA, 1977.

[13] I.A. Essa. Visual interpretation of facial expressions using dynamic modeling. Technical Report 235, MIT Media Laboratory, July 1993.

[14] F. Girosi, M. Jones, and T. Poggio. Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. A.I. Memo No. 1430, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993.

[15] B.K.P. Horn and B.G. Schunk. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[16] D.P. Huttenlocher and S. Ullman. Recognizing solid objects by alignment with an image. *International Journal of Computer Vision*, 5(2):195–212, 1990.

[17] H. Li, P. Roivainen, and R. Forchheimer. 3-D motion estimation in model-based facial image coding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15(6):545–555, 1993.

[18] S. Librande. Example-based character drawing. Master's thesis, M.S., Media Arts and Science Section, School of Architecture and Planning, Massachusetts Institute of Technology, Cambridge, MA, September 1992.

[19] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings IJCAI*, pages 674–679, Vancouver, 1981.

[20] J.L. Mundy, R.P. Welty, M.H. Brill, P.M. Payton, and E.B. Barrett. 3-D model alignment without computing pose. In *Proceedings Image Understanding Workshop*, pages 727–735. Morgan Kaufmann, San Mateo, CA, January 1992.

[21] H. Murase and S.K. Nayar. Learning and recognition of 3D objects from appearance. In *IEEE 2nd Qualitative Vision Workshop*, pages 39–50, New York, NY, June 1993.

[22] Yuichiro Nakaya, Yeow Chiang Chuah, and Hiroshi Harashima. Model-based/waveform hybrid coding for videotelephone images. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, pages 2741–2744, 1991.

[23] M. Oka, K. Tsutsui, A. Ohba, Y. Kurauchi, and T. Tago. Real-time manipulation of texture-mapped surfaces. *SIGGRAPH '87 Proceedings*, pages 181–188, July 1987.

[24] T. Poggio. 3D object recognition: on a result by Basri and Ullman. Technical Report # 9005–03, IRST, Povo, Italy, 1990.

[25] T. Poggio. A theory of how the brain might work. In *Cold Spring Harbor Symposia on Quantitative Biology*, pages 899–910. Cold Spring Harbor Laboratory Press, 1990.

[26] T. Poggio. 3D object recognition and prototypes: one 2D view may be sufficient. Technical Report 9107–02, I.R.S.T., Povo, Italy, July 1991.

[27] T. Poggio and R. Brunelli. A novel approach to graphics. A.I. Memo No. 1354, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[28] T. Poggio and S. Edelman. A network that learns to recognize 3D objects. *Nature*, 343:263–266, 1990.

[29] T. Poggio and F. Girosi. A theory of networks for approximation and learning. A.I. Memo No. 1140, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989.

[30] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9), September 1990.

[31] T. Poggio and A. Hurlbert. Observation on cortical mechanisms for object recognition and learning. In C. Koch and J. Davis, editors, *Large-scale Neuronal Theories of the Brain*. In press.

[32] T. Poggio and T. Vetter. Recognition and structure from one 2D model view: Observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, February 1992.

[33] T. Poggio and T. Vetter. Recognition and structure from one 2D model view: observations on prototypes, object classes and symmetries. A.I. Memo No. 1347, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1992.

[34] A. Shashua. Correspondence and affine shape from two orthographic views: Motion and Recognition. A.I. Memo No. 1327, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, December 1991.

[35] A. Shashua. *Geometry and Photometry in 3D visual recognition*. PhD thesis, M.I.T Artificial Intelligence Laboratory, AI-TR-1401, November 1992.

[36] A. Shashua. Illumination and view position in 3D visual recognition. In S.J. Hanson J.E. Moody and R.P. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 404–411. San Mateo, CA: Morgan Kaufmann Publishers, 1992. Proceedings of the fourth annual conference NIPS, Dec. 1991, Denver, CO.

[37] A. Shashua. On photometric issues in 3D visual recognition from a single 2D image. *International Journal of Computer Vision*, 1993. Submitted.

[38] A. Shashua. Projective depth: A geometric invariant for 3D reconstruction from two perspective/orthographic views and for visual recognition. In *Proceedings of the International Conference on Computer Vision*, pages 583–590, Berlin, Germany, May 1993.

[39] L. Stringa. Eyes detection for face recognition. Technical Report 9203-07, I.R.S.T., Trento, Italy, 1991.

[40] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-15(6):569–579, 1993.

[41] S. Ullman. Aligning pictorial descriptions: an approach to object recognition. *Cognition*, 32:193–254, 1989. Also: in MIT AI Memo 931, Dec. 1986.

[42] S. Ullman and R. Basri. Recognition by linear combination of models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-13:992—1006, 1991. Also in M.I.T AI Memo 1052, 1989.

[43] D. Weinshall. Model based invariants for3-D vision. *International Journal of Computer Vision*, 10(1):27–42, 1993.

[44] A.L. Yuille, D.S. Cohen, and P.W. Hallinan. Feature extraction from faces using deformable templates. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, pages 104–109, San Diego, CA, 1989.

A Regularization networks corresponding to bilinear splines

Consider $\mathbf{x} = (x, y)$ with 4 examples $\mathbf{y}_i$, $i = 1, \cdots 4$, located at the corners of the unit square. If we use $G(\mathbf{x}) = |x||y|$ as in figure 5, which corresponds to interpolation by tensor product piece-wise linear splines, then

$$\mathbf{G} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

and

$$\mathbf{G}^+ = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

Thus $b_i = \sum_{j=1}^{4} (\mathbf{G}^+)_{ij} g_j$ which implies by defining with $\mathbf{x}_i$ the x, y coordinates of example i

$$b_i(\mathbf{x}) = G(\mathbf{x} - \mathbf{x}_{5-i}) \tag{12}$$

with $i = 1, \cdots, 4$. For our choice of $G(\mathbf{x}) = |x||y|$ equation 12 gives the bilinear coefficients

$$b_i(\mathbf{x}) = |x - x_{5-i}||y - y_{5-i}|.$$

These simple relations can be easily extended to more than 2 dimensions, provided the examples are at the corners of the unit hypercube in input space.

B The learning networks and its dual representation

Let us assume that the input dimensionality is d, the network has n centers with $n \leq N$, the output dimensionality is q and there are N examples in the training set. This means that $\mathbf{x}$ is a $d \times 1$ matrix, $\mathbf{C}$ is a $q \times n$ matrix, $\mathbf{Y}$ is a $q \times N$ matrix and $\mathbf{G}$ is a $n \times N$ matrix. Consider the equations

$$\mathbf{y}(\mathbf{x}) = \mathbf{C}\mathbf{g}(\mathbf{x}) \tag{13}$$

and its dual

$$\mathbf{y}(\mathbf{x}) = \mathbf{Y}\mathbf{b}(\mathbf{x}) \tag{14}$$

Equation 13 can be regarded as a mapping from $\mathbf{x}$ to $\mathbf{y}$, whereas equation 14 may be regarded as a mapping from the space of the b coefficients into $\mathbf{y}$. The transformation from a point $\mathbf{x}$ in X space and a point b in B space is given by

$$\mathbf{b}(\mathbf{x}) = \mathbf{G}^+\mathbf{g}(\mathbf{x}). \tag{15}$$

Given $\mathbf{x}$, $\mathbf{b}$ is determined by the previous equation: the viceversa is not in general possible.

Notice that when $n = N$ each $\mathbf{y}$ is approximated by a linear combination of the N examples. When $n < N$ this is still true but we can also consider the following equation

$$\mathbf{y}(\mathbf{x}) = \mathbf{C}\mathbf{g}(\mathbf{x}) \tag{16}$$

which can be interpreted to mean that each $\mathbf{y}$ is approximated by the linear combination of n vectors – the columns of $\mathbf{C}$ – that are linear transformations of all the N examples $\mathbf{y}$ with coefficients given by the elements of $\mathbf{g}$, that is

$$\mathbf{y} = \sum_{l=1}^{n} g_l \mathbf{c}_l. \tag{17}$$

This can be compared with the previous equation 7:

$$\mathbf{y} = \sum_{l=1}^{N} b_l \mathbf{y}_l \tag{18}$$

B.1 K-L decomposition

Consider the previous equation

$$\mathbf{y} = \mathbf{Y}\mathbf{b}(\mathbf{x}) = \sum_{l=1}^{N} b_l(\mathbf{x})\mathbf{y}_l \tag{19}$$

and expand each of the columns of $\mathbf{Y}$ – that is each of the examples $\mathbf{y}_i$ – in $Q \leq N$ of the eigenfunctions of $\mathbf{Y}\mathbf{Y}^T$ – which has dimensionality $q \times q$. Thus

$$\mathbf{y}_i = \sum_{m=1}^{Q} d^i_m \mathbf{y}'_m = \mathbf{Y}'\mathbf{d}^i \tag{20}$$

and combining the last two equations

$$\mathbf{y} = \sum_{l=1}^{N} b_l(\mathbf{x})\mathbf{y}_l \sum_{m=1}^{Q} d^l_m \mathbf{y}'_m = \mathbf{Y}'\mathbf{b}'(\mathbf{x}) \tag{21}$$

where $\mathbf{Y}'$ is a $q \times Q$ matrix and $\mathbf{b}'(\mathbf{x})$ is a Q dimensional vector of the coefficients of the expansion of the vector $\mathbf{y}$ in the eigenfunctions of $\mathbf{Y}\mathbf{Y}^T$, that is

$$\mathbf{b}'(\mathbf{x}) = \mathbf{Y}'^T \mathbf{y}(\mathbf{x}) \tag{22}$$

where $\mathbf{Y}'$ is the matrix with columns the eigenvectors of $\mathbf{Y}\mathbf{Y}^T$.

Thus each output $\mathbf{y}$ *can be approximated by a linear combination of eigenfunctions of the example set* $\mathbf{y}_i$. As a consequence, a smaller set of example images may be used in the linear combination technique than the number of examples, provided they are chosen to be the most significant eigenfunctions of the correlation matrix of the example set and they are sufficient to approximate well novel images.

Of course the eigenvectors of $\mathbf{Y}\mathbf{Y}^T$ can be found in terms of the eigenvectors of $\mathbf{Y}^T\mathbf{Y}$ which may have a much lower dimensionality (for instance when the columns of $\mathbf{Y}$ are images). [1]

C A comparison of the two approaches to teleconferencing

The first approach to teleconferencing exploits the following decomposition (see equation 17) to be used at the receiver site for reconstruction of the image $\mathbf{y}$

$$\mathbf{y} = \sum_{l=1}^{n} g_l \mathbf{c}_l \tag{23}$$

The alternative technique based on the dual representation of the networks uses equation 7:

$$\mathbf{y} = \sum_{l=1}^{N} b_l \mathbf{y}_l \tag{24}$$

We have shown that the two equations are equivalent since they are two ways of rewriting the same expression. Reconstruction performance is therefore expected to be the same. There are however somewhat different computational tradeoffs offered by the two techniques. Let us denote in this appendix with q the image dimensionality and with d the dimensionality of the pose-expression space. In the first technique (equation 23) the sender has to send in batch mode and store (SBS) n d-dimensional centers plus $n \times q$ parameters c and send at run time (SR) d numbers (the $\mathbf{x}$ vector). In the second technique – equation 24 – the sender has to send in batch mode beforehand and memorize at the receiver site N q-dimensional $\mathbf{y}_l$; it has to send at run time N numbers b_i. In general $d << N$; assume for example $N = 50$, $n = 20$, $q = 10^5$, $d = 5$. Then the first technique has $SBS = n \times d + n \times q = 210^6$ and $SR = 5$ while the second technique has $SBS = N \times q = 510^6$ and $SR = 50$. This very rough estimate neglects precision issues.

Only experiments can decide the actual tradeoffs between the two techniques, if any. In terms of power and flexibility the analysis-synthesis technique seems superior: the two modules ("sender" and "receiver") are independent and could use different examples in a different number and in different positions in the space. This is not so for the linear combination technique which on the other hand may be simpler in certain cases since it does not require the assignment by the user of pose-expression coordinates to the example images (which of course could also be done automatically).

In the case of graphic applications the requirements are somewhat different. Assume $n = N$ (the vanilla RBF case). In the case of the first technique learning requires the off-line computation and storage of the q, d matrix $\mathbf{C} = \mathbf{Y}\mathbf{G}^+$; at run time the system computes the $d, 1$ vector $\mathbf{g}(\mathbf{x})$ and then $\mathbf{y} = \mathbf{C}\mathbf{g}$. In the case of the second technique learning requires the storage of the q, N matrix $\mathbf{Y}$ and the computation and storage of the N, d matrix $\mathbf{G}^+$; at run time the system needs to compute the $d, 1$ vector $\mathbf{g}(\mathbf{x})$, the $N, 1$ vector $\mathbf{b} = \mathbf{G}^+\mathbf{g}$ and finally $\mathbf{y} = \mathbf{Y}\mathbf{b}$. The first technique requires less operations at run time.

D Changing reference frames for the rend operation

During the **rend** operation,

$$img \leftarrow \mathbf{rend}(\mathbf{y}, img_{tex}, \mathbf{y}_{tex}),$$

if the reference image img_{ref} and the example image img_{tex} (used to sample facial texture) are not the same, the facial geometry vector $\mathbf{y}$ must have its reference frame changed to that of image img_{tex}. As shown in figure 13, this is accomplished by

$$\begin{aligned} \mathbf{y}' &\leftarrow \mathbf{y} - \mathbf{y}_{tex} \\ \mathbf{y}'' &\leftarrow \mathbf{warp\text{-}vect}(\mathbf{y}', \mathbf{y}_{tex}), \end{aligned}$$

where the first step subtracts out the geometry of img_{tex}, but still in the reference image of img_{ref}. The second step uses a 2D warp to translate the correspondences to the new reference image img_{tex}. Finally, the image is rendered using a 2D warp from img_{tex}

$$img \leftarrow \mathbf{warp}(img_{tex}, \mathbf{y}'').$$

[1] Notice that there is a connection between this formulation and parametric eigenspace representation for visual learning and recognition (Murase and Nayar, 1993). In particular, our $\mathbf{b}'(\mathbf{x})$ is similar to their parametrized space for pose. A significant difference is however that our approach recognizes the critical importance of correspondence between the images whereas theirs does not.

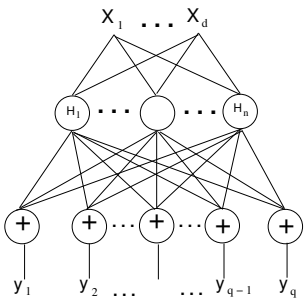

Figure 1: The most general Regularization Network with one hidden layer and vector output. In the analysis case the network will have images as inputs and pose-expression parameters as outputs; in the synthesis case inputs are pose parameters and outputs are images.

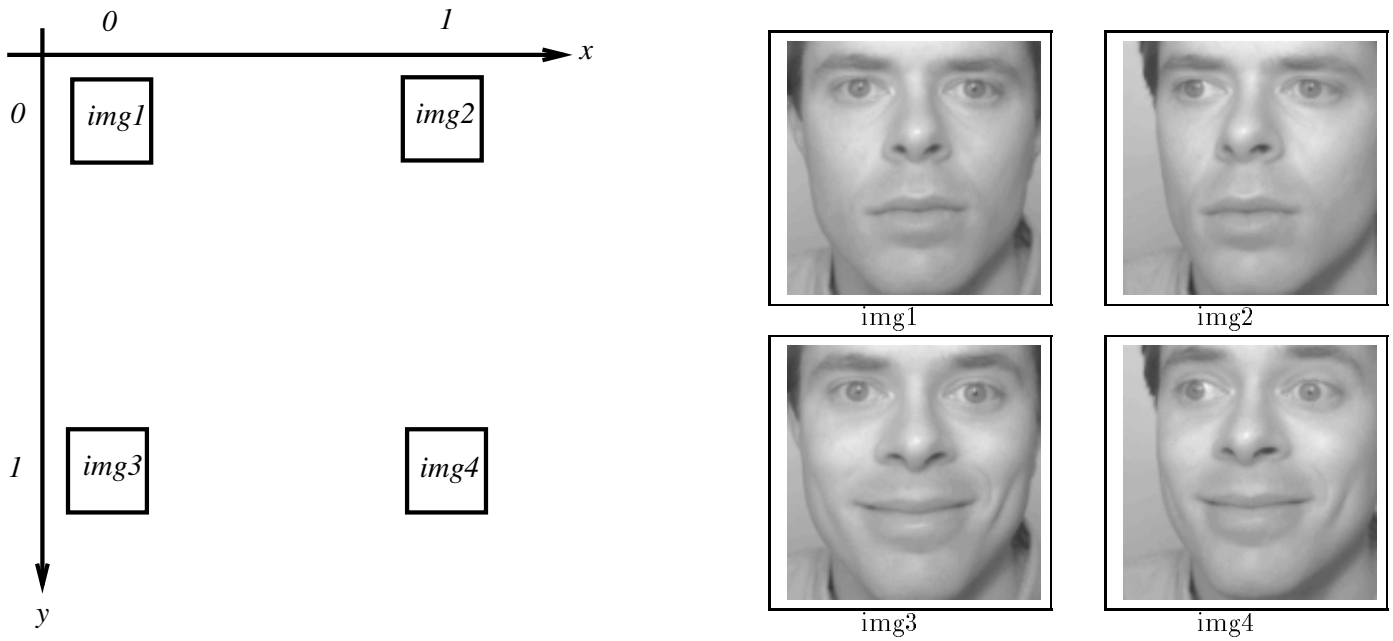

Figure 2: In our demonstrations of the example-based approach to image analysis and synthesis, the example images img_1 through img_4 are placed in a 2D rotation-expression parameter space (x, y), here at the corners of the unit square. For analysis, the network learns the mapping from images (inputs) to parameter space (output). For synthesis, we synthesize a network that learns the inverse mapping, that is the mapping from the parameter space to images.

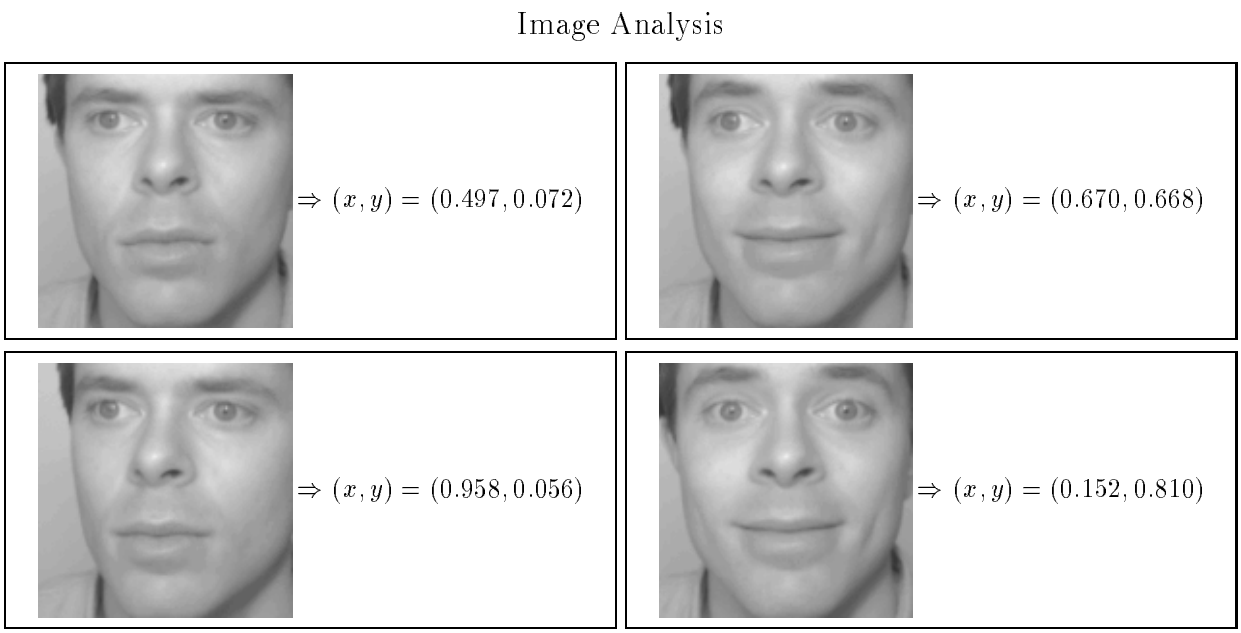

Figure 3: The analysis network trained on the four examples of figure 2 estimates the two pose/expression parameters for a novel image. The figure shows four cases of novel images – that is different from the examples in the training set – and the output of the network for each of them.

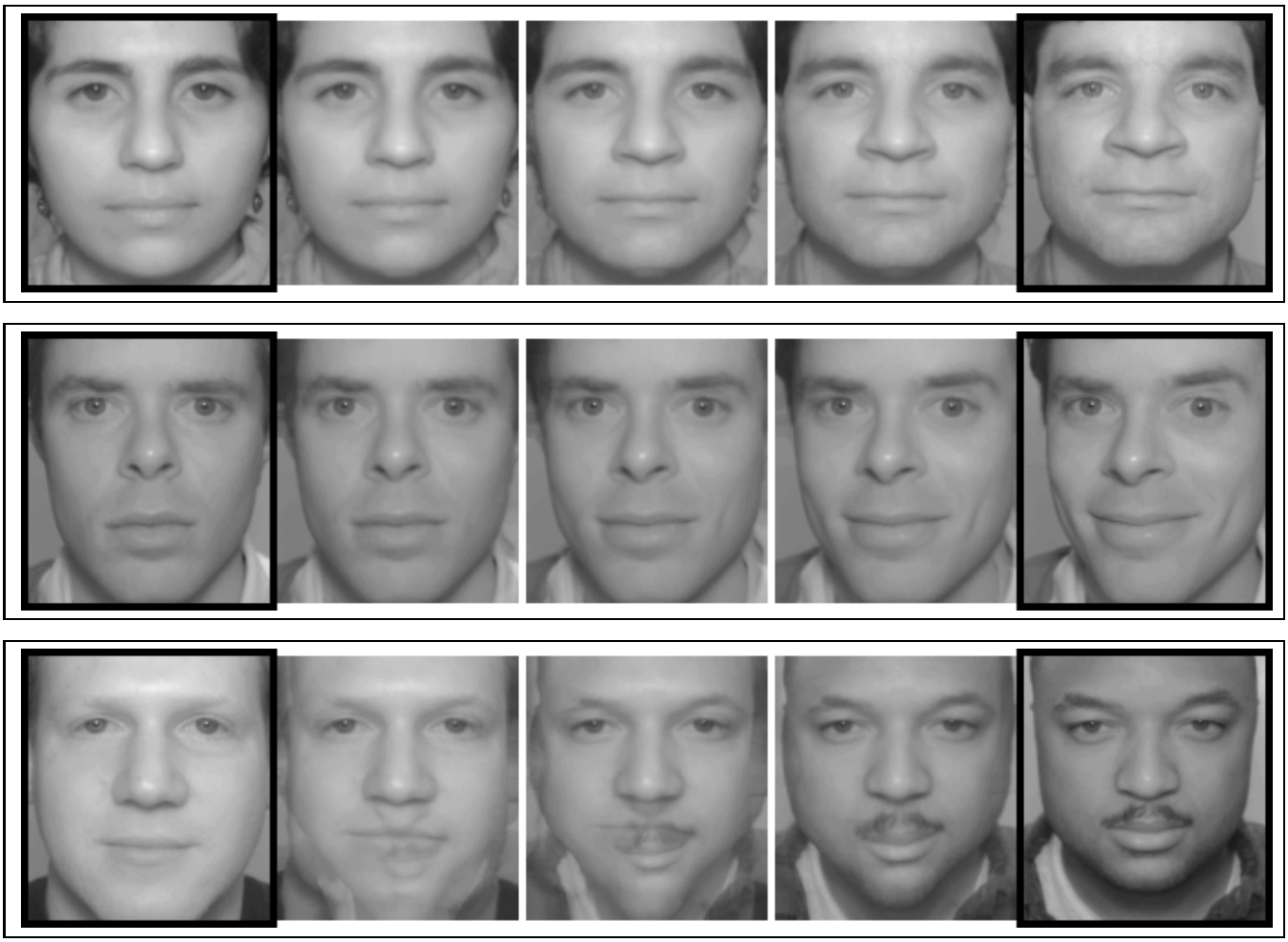

Figure 4: Three cases of 1D interpolation between the two example images bordered in black. In this case the synthesis network has one input only, two centers (i.e. hidden units) and 132×148 outputs, with each output representing the x or the y coordinate of a labeled pixel. The third case is an amusing failure case where the optical flow algorithm failed to find correct correspondences.

Figure 5: In this example of multidimensional image synthesis the input variables are two – rotation and smile – while the output of the network has as many dimensions as twice the number of pixels in the images (since it represents the x, y coordinates of each ordered pixel). The four training examples – the same shown in figure 2 – are singled out by black borders. All other images are synthesized by a regularization network with $G(\mathbf{x}) = |x||y|$, which effectively performs bilinear interpolation, after "learning" the appropriate parameters from the 4 examples.

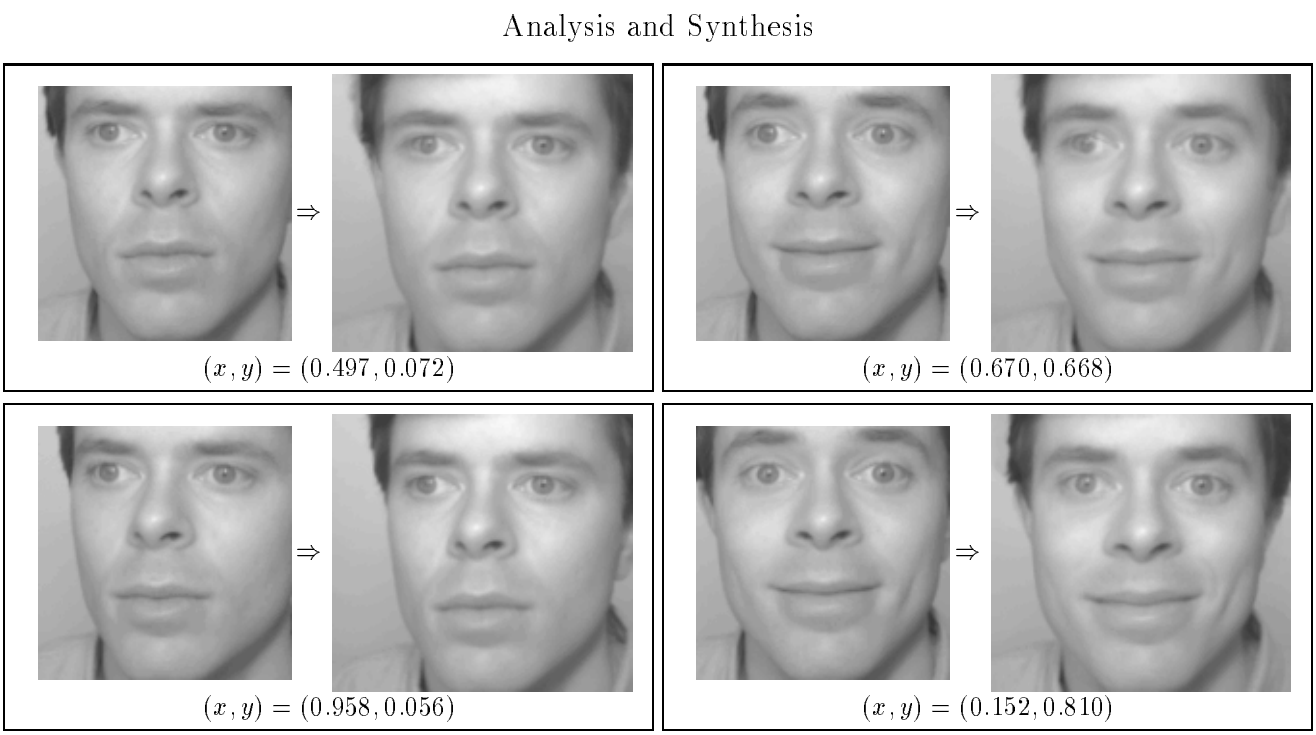

Figure 6: In each of the boxed image pairs, the novel input image on the left is fed into an analysis RBF network to estimate rotation x and expression y (as shown in figure 3). These parameters are then fed into the synthesis module of figure 5 that synthesizes the image shown on the right. This figure can be regarded as a very simple demonstration of very-low bandwidth teleconferencing: only two pose parameters need to be transmitted at run-time for each frame.

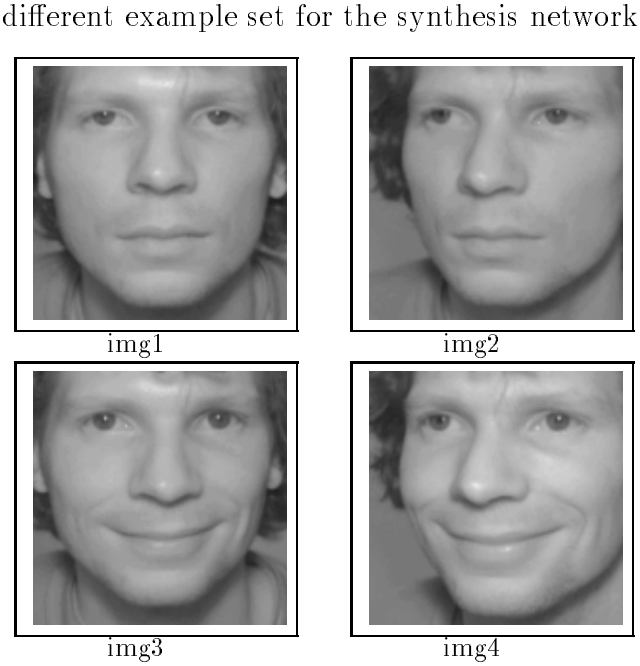

Figure 7: Another set of example images that the synthesis network uses to synthesize a face *different* from the face used by the analysis network to estimate rotation and expression.

Directing Another Person

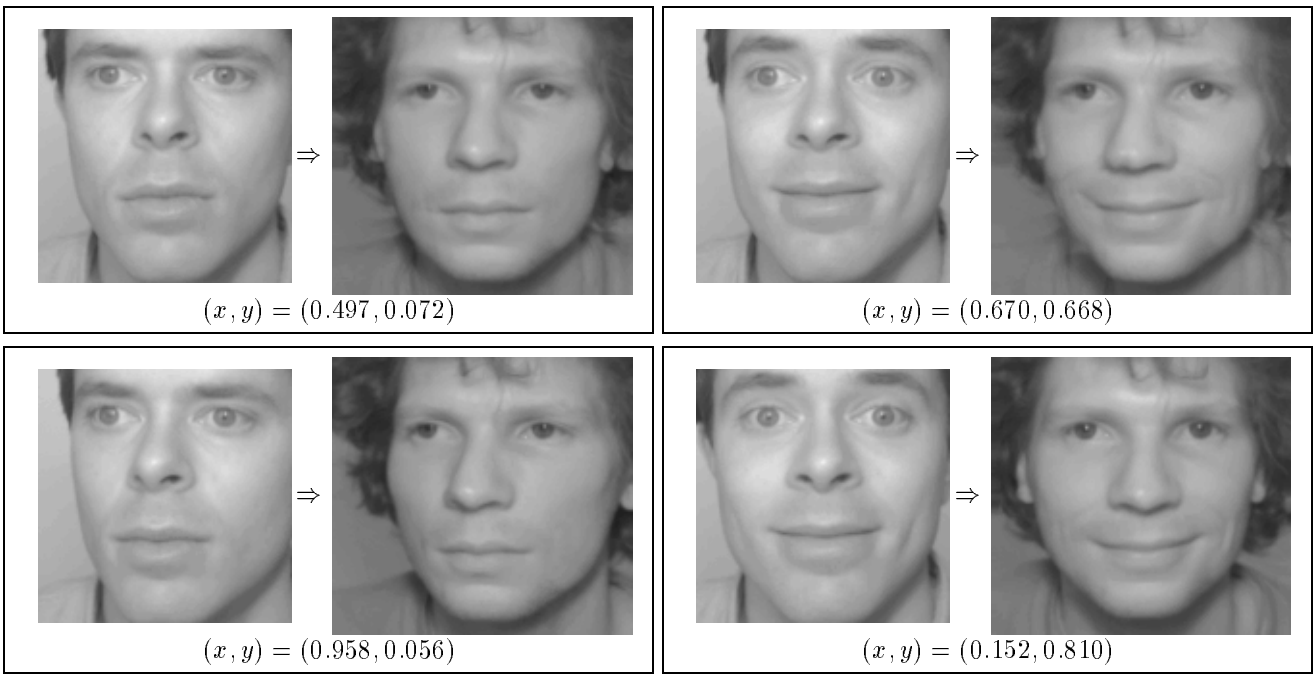

Figure 8: "Directing" another person: in each of the boxed image pairs, the input image on the left is fed to an analysis RBF network to estimate rotation x and expression y. These parameters are then fed into the synthesis network trained with the examples (see figure 7) of *another* person, which synthesizes the image shown on the right. In this way the "actor" on the left effectively "directs" the person of figure 7.

Example Set 1

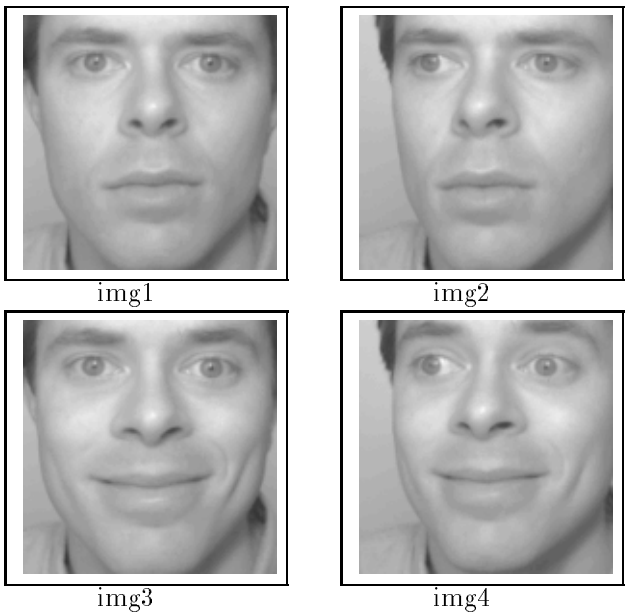

Figure 9: The set of example images at the sender and at the receiver site used by the linear combination technique demonstrated in the next figure (same images as in figure 2). In this case a pose value does not need to be assigned to each image.

Example Transmitter/Receiver Pairs

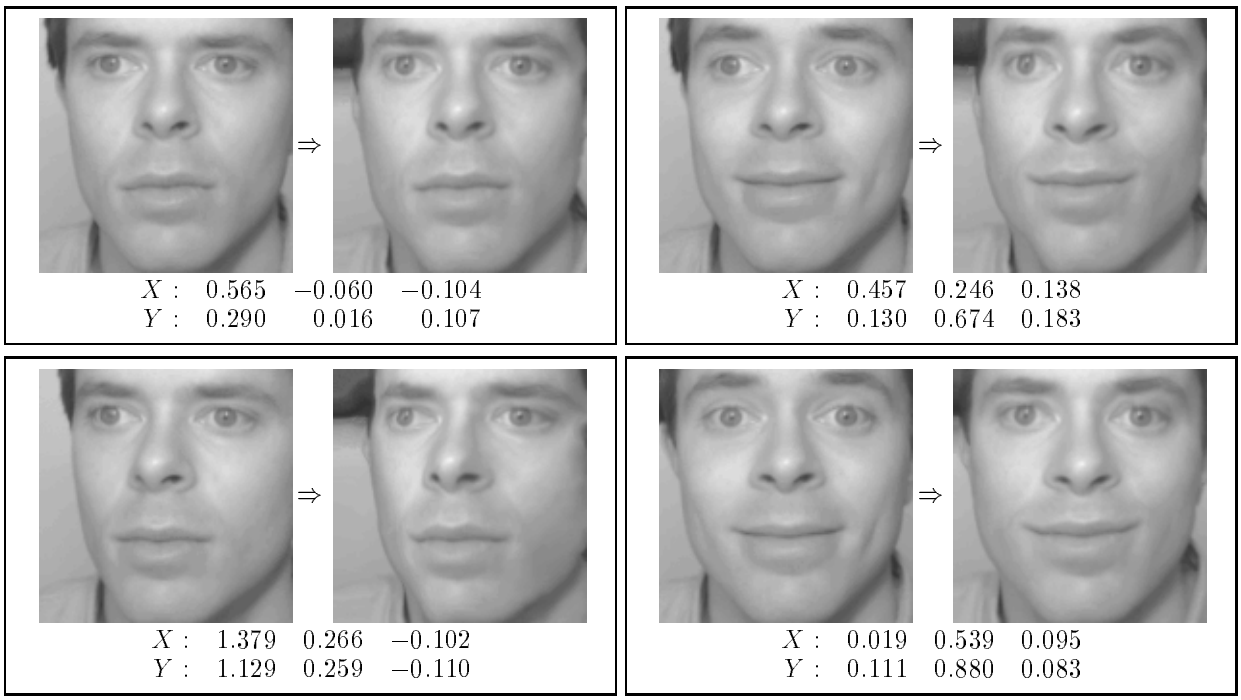

Figure 10: A novel image (left) is decomposed in terms of a linear combination of the example images of the previous figure and reconstructed using the same example images at the receiver site. Separate b_l coefficients are listed for the x and y components of the vectorized image since they are decomposed separately using equation (9).

Different Transmitter/Receiver Pairs

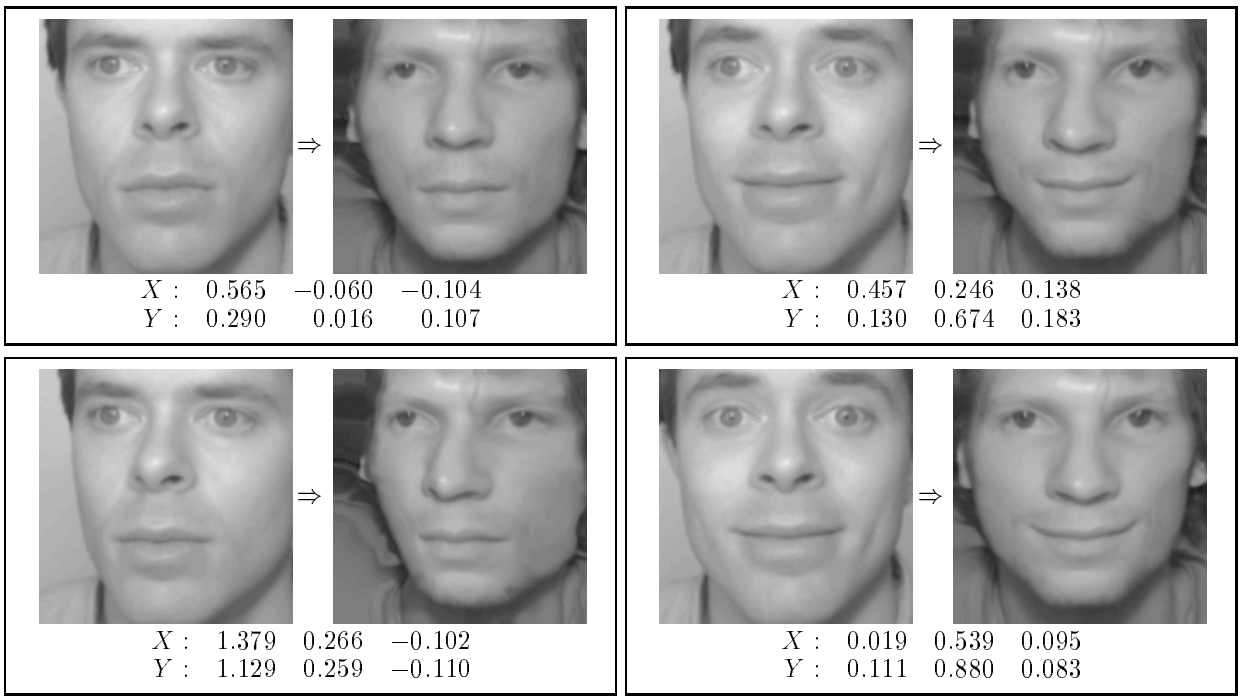

Figure 11: A novel image (left) is decomposed in terms of a linear combination of the example images of figure 9 and reconstructed at the receiver site combining with the same coefficients the example views shown in figure 7 of a *different* person.

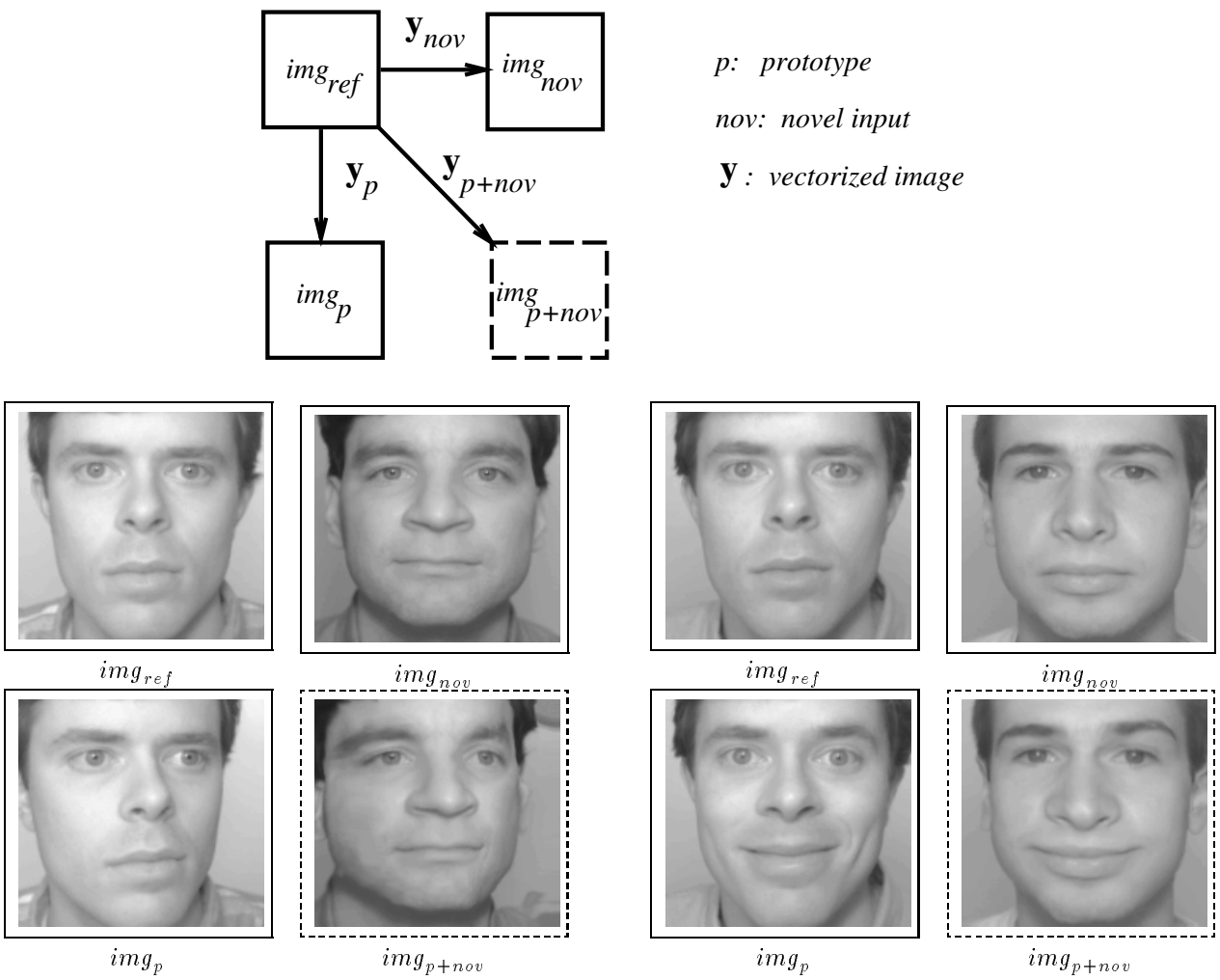

Figure 12: A face transformation is "learned" from a prototypical example transformation. Here, face rotation and smiling transformations are represented by prototypes, $\mathbf{y}_p$. $\mathbf{y}_p$ is mapped unto the new face image img_{nov} by using the correspondences specified by the flow $\mathbf{y}_{nov}$. The image img_{p+nov} is synthesized by the system.

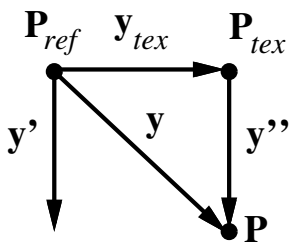

Figure 13: Changing the reference frame during the **rend** operation. When the facial geometry $\mathbf{y}$ is rendered using the image texture from img_{tex} – which is not the reference image – we must compute $\mathbf{y}$ relative to img_{tex}, producing $\mathbf{y}''$. Here, point P corresponds with P_{ref} and P_{tex} in the reference image img_{ref} and texture image img_{tex}, respectively.