Image Representations for Visual Learning

David Beymer and Tomaso Poggio*

Computer vision researchers are developing new approaches to object recognition and detection that are based almost directly on images and avoid the use of intermediate three-dimensional models. Many of these techniques depend on a representation of images that induces a linear vector space structure and in principle requires dense feature correspondence. This image representation allows the use of learning techniques for the analysis of images (for computer vision) as well as for the synthesis of images (for computer graphics).

The synthesis problem, the classical problem of computer graphics, can be formulated as the problem of generating novel images corresponding to an appropriate set of parameters that describe the camera viewpoint and aspects of the scene. The inverse analysis problem, that of estimating object labels as well as scene parameters from images, is the classical problem of computer vision. Since the 1980s, researchers in both fields have used intermediate, physically based models to approach their respective problems of synthesis and analysis. In computer graphics, sophisticated three-dimensional (3D) modeling and rendering techniques have been developed that effectively simulate the physics of rigid and nonrigid solid objects and the physics of imaging (*1*). Some research in 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 database of models (*2*). More recently, researchers in the two key areas of object recognition and object detection (*3*) have used a rather different approach that has been called image-based or view-based (*4–8*). In this approach, the image to be analyzed is compared directly, possibly after a simple filtering stage, with a set of example images.

The Role of Correspondence

The key underlying mathematical assumption of the image-based approach is that the images form a linear vector space (*9*). However, images are just arrays of numbers or pixels, not vectors. A set of raw images—say, of similar objects—does not have the structure of a vector space, because operations like addition do not have a well-defined meaning for raw images (*10*). In pattern recognition, a standard technique for associating a vector to an image is to derive the vector components from an ordered set of measurements on the image (*11*). This technique, however, is incompatible with image-based approaches, where vector components must correspond to pixels. A vector space structure implies that the *i*th component of all vectors in the set must refer to the same type of feature on the imaged surfaces. Strictly speaking, the use of vector space techniques in image-based approaches requires the solution of the correspondence problem: finding pixels in two or more images that represent corresponding surface features in the scene. Correspondence is a difficult problem in computer vision that can be solved for sufficiently similar images by techniques such as optical flow algorithms, which find corresponding pixels in two or more images and compute their displacement vectors (in the image plane). Correspondence transforms images into vectors by associating to feature point *i* its color and its *x*,*y* position.

In dense correspondence, each pixel corresponds to a feature point; in this context, it is useful to think of the single vector containing an ordered list of color and *x*,*y* values as two separate parts. The texture vector is an ordered list of the values (grayscale or color) of corresponding pixels relative to a reference image. The shape vector is an ordered list of the disparities, that is, the *x*,*y* locations of corresponding features relative to the reference image. We refer to the correspondence operations that associate the shape and texture vectors to an image as vectorizing the image (*12*); this process will be explained later (*13*). Edge-based or contour-based approaches also fit into this framework; if the images used are edge maps or line drawings, then the shape vector could include only entries for points along an edge. In this case, the texture vector would not be used [(*14*); see also the contour-based approach of (*15*, *16*)].

The shape and texture vectors form separate linear vector spaces with specific properties. The shape vectors resulting from different orthographic views of a 3D object (in which features are always visible) constitute a linear vector subspace of very low dimensionality [spanned by just two views; see (*4*, *17*)]. For real objects, self-occlusions and correspondence define different vector spaces or charts (*18*) for different points of view, correspondingly defining different aspects (*19*). For a fixed viewpoint, a specific class of objects (such as faces) with a similar 3D structure (*20*) seems to induce a texture vector space of relatively low dimensionality, as shown indirectly by the results of Kirby and Sirovich [(*21*), see also (*22*)], who did not use exact correspondence. Using exact correspondence, Vetter and Poggio (*23*) and Beymer and Poggio (*24*) showed that a good approximation of a new face image can be obtained with as few as 50 base faces, which suggests a low dimensionality (*25*) for both the shape and the texture spaces [see also (*16*, *26*)]. Most of the object detection and recognition schemes mentioned above do not explicitly acknowledge the need to vectorize images and instead use approximate correspondence, exploiting anchor points such as the eyes of a face, to provide texture vectors on which to run the algorithm (*6*, *22*, *27–29*). Others [(*4*, *15*, *30–32*); see also (*33*)] have assumed or used exact correspondence (relying only on shape vectors rather than texture vectors). Craw and Cameron (*18*), Lanitis *et al.* (*26*), and Beymer (*34*) have used shape and texture vectors for recognition. In any case, feature correspondence and the resulting vector structure underlie, either implicitly or explicitly, many of the recent view-based approaches to recognition and detection.

A similar situation is emerging in the field of computer graphics. In a seminal paper, Ullman and Basri [(*4*); see also the rank theorem of Tomasi and Kanade (*35*)] showed that for orthographic projection, new images can be directly synthesized from a small set of images if they are represented as shape vectors. Shashua (*36*) extended this result to the perspective case. Reprojection techniques are now being developed that synthesize new views of a scene without an explicit 3D model [see, for instance, (*37*)]. Unlike the earlier work (*4*, *27*, *38*), the most recent research focuses on visibility issues induced by camera transformations (*39*, *40*).

The authors are in the Department of Brain and Cognitive Science, Center for Biological and Computational Learning (CBCL) and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02142, USA.

*To whom correspondence should be addressed. E-mail: tp@ai.mit.edu

Learning Networks

We now describe how a vectorized image representation enables the use of still another powerful framework—learning from examples—to approach both computer vision and computer graphics and to effectively bypass physically based 3D models (*38*). This example-based approach, which implies a view-based technique, is related to but does not directly rely on the linear combination and reprojection results described earlier. In the metaphor of this article, we speak of learning the mapping—from images to pose parameters or from pose parameters to images—from a set of examples. The examples are input-output 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 synthesize a novel image for new values of the control parameters. The approach has advantages and disadvantages, and although it cannot replace the traditional 3D approach, it may be quite appropriate in some situations.

Our learning-from-examples approach can be used for both the analysis and synthesis of images of objects such as faces. For image analysis, the mapping from novel images to a vector of pose and expression parameters is computed by what we call an analysis network. For synthesis, the inverse mapping (from input pose and expression parameters to output images) can be used to synthesize new images through a similar learning network called a synthesis network. Such an example-based learning approach to vision and graphics trades memory for computation. The more traditional 3D approaches use computationally expensive rendering techniques, whereas our approach requires the memory needed to store a sufficient number of training examples.

Consider the problem of approximating a vector field $\mathbf{y}(\mathbf{z})$ from a set of sparse data, the examples, which are pairs $(\mathbf{z}_i,\mathbf{y}_i)$ $(i = 1, \ldots, N)$. Several multivariate approximation schemes are possible; here, we use specific instances of regularization networks—a class of networks with one "hidden" layer and linear output units—that encompass many standard approximation, statistical, and neural network techniques (*41*, *42*). For the case of N examples, $n \leq N$ hidden units or "centers," input dimensionality d, and output dimensionality q, the approximation is

$$\mathbf{y}(\mathbf{z}) = \sum_{i=1}^{n} \mathbf{c}_i G(\mathbf{z} - \mathbf{u}_i) \qquad (1)$$

where the $\mathbf{c}_i$ are vectors of coefficients weighting the hidden layer and G is the chosen basis function, which can be a radial basis function [such as the Gaussian (*43*)] or a spline such as a tensor-product spline. In Eq. 1, we have neglected polynomial terms needed only for G that are not positive definite. The $\mathbf{u}_i$ are d-dimensional centers of the basis functions (*44*). In the specific analysis and synthesis networks used here, we have used $n = N$. The original examples $\mathbf{z}_i$ are the centers $\mathbf{u}_i$, and the network parameters $\mathbf{c}_i$ are found by solving a linear system of Eqs. 1 over the training data.

As discussed above, the central part of our approach is to vectorize the images by representing them as separate shape and texture vectors. For computing correspondence, we have found that standard multiresolution optical flow algorithms (*31*, *45*), preceded by certain normalization steps, can automatically compute dense pixelwise matches for sufficiently similar images (*46*). After one image is defined as a reference image *ref*, the (x,y) locations of feature points (here, pixels) of a new image *img* are estimated by computing the optical flow between the two images (Fig. 1). The shape vector **s** contains the relative displacements between corresponding feature points in images *ref* and *img*. The texture vector **t** contains the image intensity (or color) values of *img* at the set of ordered feature points. The shape and texture vectors can be rendered by using **s** to warp **t**.

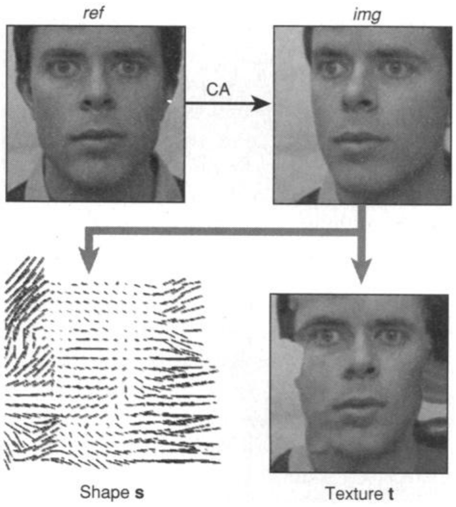

Fig. 1. An (**s**,**t**) vectorized image representation. Pixelwise correspondences are computed between image *img* and a standard reference image *ref* (CA, correspondence algorithm). Shape **s** consists of the (x,y) displacements between corresponding pixels, and texture **t** is the texture of *img* warped to the shape of *ref*. Although **t** shows the effect of self-occlusions, it contains all the information needed for *img*.

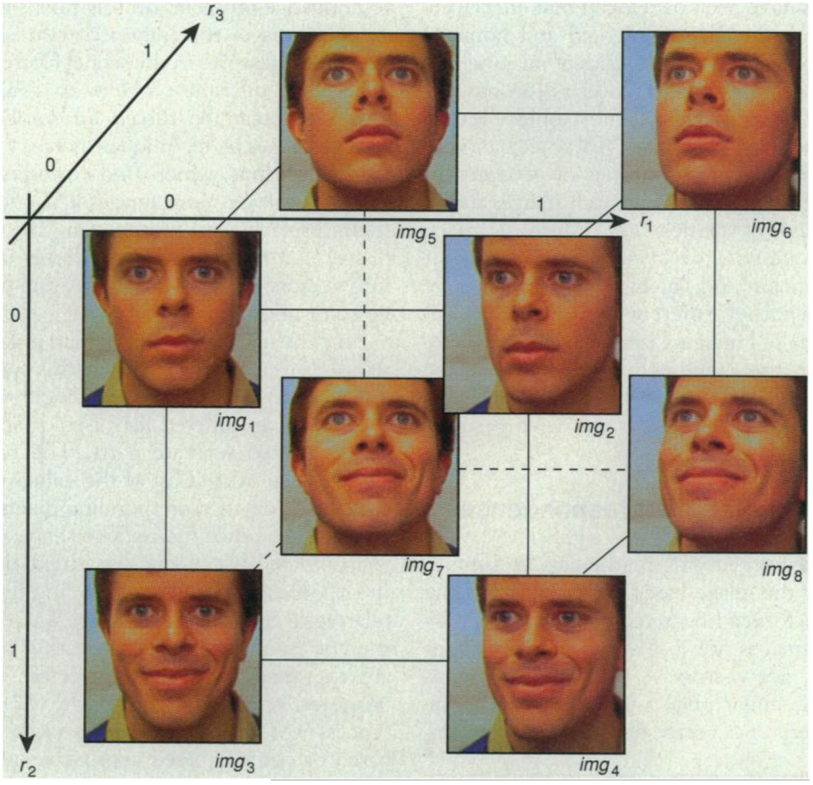

Fig. 2. In this demonstration of the example-based approach to image analysis and synthesis, the example images img_1 through img_8 are placed in a 3D rotation-expression parameter space (r_1,r_2,r_3), here at the corners of the unit cube. For analysis, the network learns the mapping from images (inputs) to parameter space (output). For synthesis, we synthesize a network that learns the inverse mapping, that is, the mapping from the parameter space to the images. The three axes of the parameter space correspond to smile, left-right rotation, and up-down rotation.

The idea underlying the analysis network is to synthesize the mapping between an image and the associated pose parameters through the use of a suitable set of examples (consisting of images and associated pose parameters). Here, we outline the learning of a mapping from input shape vectors **s** to an output pose vector **r** by means of a Gaussian radial basis function network, Eq. 1 [see also (*5, 47*)]. Figure 2 shows eight example images and their assigned locations along the corners of a unit cube in 3D pose space $\mathbf{r} = (r_1,r_2,r_3)$, where r_1 is a leftward rotation, r_2 is a smile, and r_3 is an upward rotation. Our choice of the unit cube here is arbitrary; the example images can be irregularly spaced in pose space **r**. The image representation **s** is computed by assigning img_1 as a reference image and computing optical flow with respect to the reference ($\mathbf{s}_1 = 0$). For analysis, the vector **s** only contains optical flow in rectangular boxes around the eyes, nose, and mouth (found automatically), because the flow computation is robust in these areas. The Gaussian radial basis function network has eight fixed centers, $\mathbf{u}_i = \mathbf{s}_i$. The learning procedure estimates the Gaussian widths and the $\mathbf{c}_i$ coefficients (*48*). The trained network can then evaluate the pose **r** of a new input image of the same type—for instance, an image of the same person used in the training set. Figure 3 shows results from such an analysis network. The network described here exploits only image shape information **s** but could be modified to also exploit the texture information **t**.

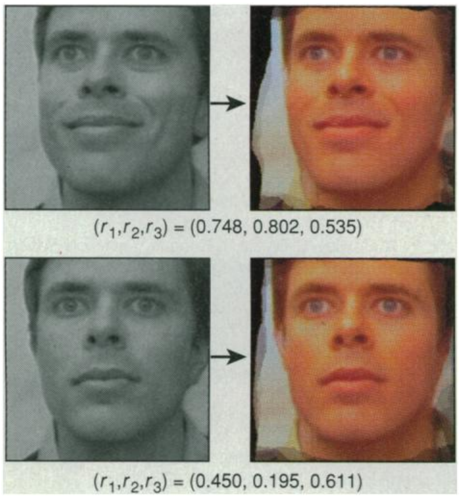
$(r_1,r_2,r_3) = (0.748, 0.802, 0.535)$
$(r_1,r_2,r_3) = (0.450, 0.195, 0.611)$

Fig. 3. In the boxed image pairs, the novel input real image (left) is fed into an analysis network to estimate left-right rotation r_1, expression r_2, and up-down rotation r_3. These parameters are then fed into the synthesis module, which synthesizes the image shown on the right. In principle, this example-based approach can achieve very high compression. Here, each frame is compressed to ~3 bytes, but the image is very limited in its range of poses and expressions. A more realistic system may require the estimation and sending of 10 to 20 parameters, plus a few others that define the image plane transformations, for a total of ~20 bytes per frame.

We now discuss an example-based technique for generating images of nonrigid 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; the input-output roles of images and pose space are reversed. In our approach, the input pose space is hand-crafted by the user to represent desired degrees of control over the images of the 3D object. After example images of the object are assigned to positions $\mathbf{r}_i$ in input pose space (Fig. 2), the example images are vectorized to create the corresponding image representation $(\mathbf{s}_i,\mathbf{t}_i)$. A regularization network is then trained on the example pairs $(\mathbf{r}_i,(\mathbf{s}_i,\mathbf{t}_i))$. Given a new pose vector **r**, the synthesis network synthesizes a new image **(s,t)** by interpolating both the shape and texture of the example images, thus performing a kind of multidimensional morphing. Such a synthesis network was used to interpolate the eight images of Fig. 2. The vectorized **(s,t)** representation for each example image was obtained automatically by computing optical flow between the example and the reference image img_1; inputs to the network are the pose parameters (r_1,r_2,r_3). The network follows Eq. 1, using tensor-product linear splines $G(r_1,r_2,r_3) = |r_1| \cdot |r_2| \cdot |r_3|$ as the basis function. Eight basis functions are used, one for each example (*38*). Finally, the network output **(s,t)** is "rendered" as an image through a 2D warping algorithm. The shape **s** represents correspondences between the texture **t** at the standard reference shape and the destination shape **s**. Forward warps [see (*49*)] can be used to locally remap pixels **t** using the set of correspondences **s**. Figure 3 shows novel images created by the synthesis network.

The analysis and synthesis networks can each be used independently in a variety of applications. The analysis network is not limited to face images and can be regarded as a trainable man-machine interface that could perform customizable gesture recognition and thereby drive a computer interface. The synthesis network is an alternative graphics and animation technique that extends to real images an approach already demonstrated by Poggio and Brunelli (*27*) and Librande (*50*) for line drawings. Applications of the analysis and synthesis networks include teleconferencing and very low bandwidth video e-mail, that is, video encapsulated in e-mail. Figure 3 shows a demonstration of the basic idea (*51*).

Large occlusions within the example images can be a problem for the analysis and synthesis networks described here, not only for the correspondence step but also for the representation of actual images. In the synthesis case, for instance, the reconstruction step may have to decide which of two pixels warped to the same image location is the visible one. Concepts such as layering (*52*), common in animation techniques, and depth ordering from image disparity (*40*) are ways to endow an image-based representation with qualitative 3D properties.

Discussion

The image analysis and image synthesis networks we have described are just one example of the many approaches that a vector

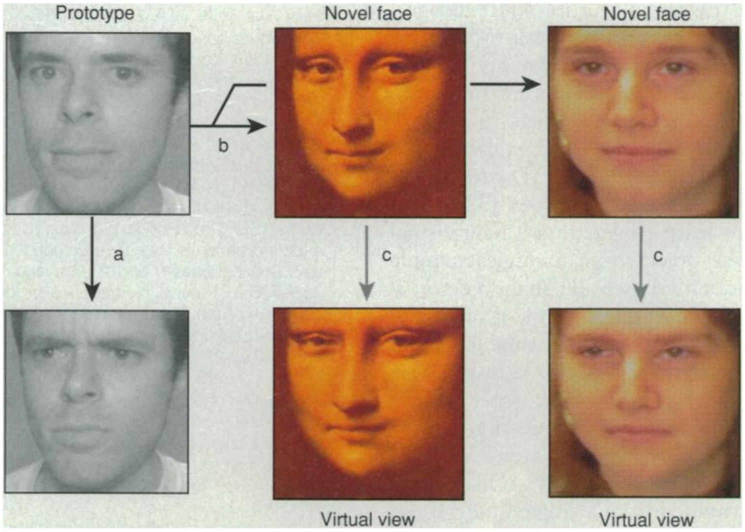

Fig. 4. One technique for generating virtual views from a single view is parallel deformation, which is simpler than the linear class approach of Vetter and Poggio (*23*). First, a 2D deformation representing a desired transformation is estimated from two prototype images in correspondence. In this example, the transformation (a) is a frown, and an optical flow algorithm is used to find feature correspondence at the pixel level. The flow is then mapped onto two novel faces (b), and the novel faces are 2D warped to synthesize virtual views (c).

representation of images makes possible. The key step in these applications is the computation of dense correspondence between images—the vectorizing step. Vectorization allows, for instance, the creation of a flexible model for a class of objects as the linear combination of prototype images and their affine deformations (*53*). This new type of flexible model can be used as a generative model to synthesize novel images of the same class. Such a model can also be used for image analysis by fitting the model parameters to an image by means of an optimization procedure (*14, 34*). The idea of this analysis-by-synthesis technique based on the linear combination of prototypes (without the use of dense correspondence) was first suggested by Taylor and co-workers (*16, 26, 30*), who have also described applications in medicine and other areas. Blake and Isard applied a similar technique to the tracking of dynamically changing images (*15, 54*).

This flexible model and its associated estimation procedure are very closely related to the synthesis network described earlier [see (*44*), Eq. 3]. The model can be used as a new example-based technique for computing correspondence (*14*); even though standard optical flow techniques work well (*55*), more robust approaches are desirable, especially when there are large occlusions (*56*). Once estimated, the parameters of the flexible model can be used for indexing and recognition and, more generally, for image analysis. They can also be used for the generation of "virtual" views and the estimation of 3D structure from single images. Consider, for instance, the case in which only one example image of an object is available; this may occur in object recognition or in analysis and synthesis tasks of the type described earlier. Vetter and Poggio (*23*) considered this problem for linear object classes. An object belongs to a linear class if its 3D structure can be described as a linear combination of the 3D structure of a small number of prototypes (*32*). It is possible to learn to generate a new virtual view of the object from a single example view, represented as a 2D shape vector, if appropriate prototypical views of other objects in the same class are available (under orthographic projection) (*57*). In this way (see also Fig. 4), new views of a specific face with a different pose or expression can be estimated and synthesized from a single view (*23, 24, 58*). Similarly, 3D structure can be estimated from a single image if the images and structures of a sufficient number of prototypical objects of the same class are available (*23, 32*). Flexible models of this type can underlie the learning of visual tasks in a top-down way, specific to object classes.

An image representation in terms of shape and texture vectors allows the use of learning and pattern recognition techniques for a variety of computer vision and computer graphics tasks (*59*). The developments described above may have implications for the key role of correspondence in biological vision (*60*). Perhaps unsurprisingly, some of the neural mechanisms that underlie object recognition in the primate cortex seem consistent with the view-based networks used in the technical applications described here (*61*) rather than with 3D model-based representations.

REFERENCES AND NOTES

1. Y. Lee, D. Terzopoulos, K. Waters, in *SIGGRAPH Proceedings*, Los Angeles, 6 to 11 August 1995 (Association for Computing Machinery, New York, 1995), pp. 55–66.
2. W. E. L. Grimson, *Object Recognition by Computer: The Role of Geometric Constraints* (MIT Press, Cambridge, MA, 1990).
3. A. Hurlbert and T. Poggio, *Nature* **321**, 651 (1986).
4. S. Ullman and R. Basri, *IEEE Trans. Pattern Anal. Machine Intell.* **13**, 992 (1991).
5. T. Poggio and S. Edelman, *Nature* **343**, 263 (1990).
6. H. Murase and S. Nayar, in *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, DC, 11 to 15 July 1993 (American Association for Artificial Intelligence Press, Menlo Park, CA, 1993), pp. 836–843.
7. D. Pomerleau, *Neural Comput.* **3**, 88 (1991).
8. T. Poggio, in *Cold Spring Harbor Symposia on Quantitative Biology* (Cold Spring Harbor Laboratory Press, Cold Spring Harbor, NY, 1990), pp. 899–910; T. Breuel, *A.I. Technical Report 1374* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1992); A. Pentland, B. Moghaddam, T. Starner, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Seattle, WA, 21 to 23 June 1994 (IEEE Computer Society Press, Los Alamitos, CA, 1994), pp. 84–91; B. A. Golomb, D. T. Lawrence, T. J. Sejnowski, in *Advances in Neural Information Processing Systems 3*, R. Lippmann, J. Moody, D. Touretzky, Eds. (Morgan Kaufmann, San Mateo, CA, 1991), pp. 572–577.
9. Techniques based on the assumption of a linear vector space include those of Murase and Nayar (*6*), who used eigenfunctions of object images; Turk and Pentland [(*22*), see also S. Akamatsu, T. Sasaki, H. Fukamachi, N. Masui, Y. Suenaga, *International Conference on Pattern Recognition*, The Hague, 30 August to 3 September 1992 (IEEE Computer Society Press, Los Alamitos, CA, 1992), pp. 217–220], who used eigenfunctions in the space of face images; and Sung and Poggio (*28*), who used the Mahalanobis distance in their face detection system [see also B. Moghaddam and A. Pentland, in (*14*), pp. 786–793; U. Fayyad, N. Weir, S. Djorgovski, in *Proceedings of the Tenth International Conference on Machine Learning*, University of Massachusetts, Amherst, 27 to 29 June 1993 (Morgan Kaufmann, San Mateo, CA, 1993), pp. 112–119].
10. A collection of objects V is a vector space if (i) for all u,v in V, $au + bv$ is in V with a,b real numbers and (ii) there is a zero vector $\mathbf{0}$ such that for all u in V, $0u = \mathbf{0}$ and $u + \mathbf{0} = u$. Other axioms omitted here ensure that addition and multiplication behave as they should.
11. Measurements such as area, perimeter, and statistical estimates of color, contrast, and other properties of images are all global features used in the past. Sparse local features can also be used; there are obvious trade-offs between feature complexity and the difficulty of the correspondence step.
12. In many object recognition tasks, correspondence is not needed over the full image. Correspondence that is limited to image patches that correspond to object components is easier and yields more robust recognition and detection schemes. In this article, discussions of full image correspondence apply to components as well.
13. The full texture-shape vector has dimensionality $3N^2 + 2N^2$, where N^2 is the number of pixels in the image (because of the three color values per pixel). Of course, it may be possible to define a sparser set of relevant feature points in the image. The term shape vector refers to the 2D shape (not 3D) relative to the reference image.
14. M. Jones and T. Poggio, in *Proceedings of the Fifth International Conference on Computer Vision*, Cambridge, MA, 20 to 23 June 1995 (IEEE Computer Society Press, Los Alamitos, CA, 1995), pp. 531–536.
15. A. Blake and M. Isard, in *SIGGRAPH Proceedings*, Orlando, FL, 24 to 29 July 1994 (Association for Computing Machinery, New York, 1994), pp. 185–192.
16. T. F. Cootes, C. J. Taylor, A. Lanitis, D. H. Cooper, J. Graham, in *Proceedings of the Fourth International Conference on Computer Vision*, Berlin, 11 to 14 May 1993 (IEEE Computer Society Press, Los Alamitos, CA, 1993), pp. 242–246.
17. T. Poggio, *Technical Report 9005-03* (Istituto per la Ricerca Scientifica e Tecnologica, Povo, Italy, 1990).
18. I. Craw and P. Cameron, in *Proceedings British Machine Vision Conference*, University of Glasgow, UK, 24 to 26 September 1991 (Springer-Verlag, London, 1991), pp. 367–370.
19. J. J. Koenderink and A. J. van Doorn, *Biol. Cybern.* **32**, 211 (1979).
20. T. Vetter, A. Hurlbert, T. Poggio, *Cereb. Cortex* **3**, 261 (1995).
21. M. Kirby and L. Sirovich, *IEEE Trans. Pattern Anal. Machine Intell.* **12**, 103 (1990).
22. M. Turk and A. Pentland, *J. Cogn. Neurosci.* **3**, 71 (1991).
23. T. Vetter and T. Poggio, *A.I. Memo 1531* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1995).
24. D. Beymer and T. Poggio, in (*14*), pp. 500–507.
25. In this framework, principal components analysis is just one of the several tools provided by the mathematical structure of a vector space, which is the actual key property here.
26. A. Lanitis, C. J. Taylor, T. F. Cootes, in (*14*), pp. 368–373.
27. T. Poggio and R. Brunelli, *A.I. Memo 1354* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1992).
28. K. K. Sung and T. Poggio, in *Proceedings of the Image Understanding Workshop*, Monterey, CA, 13 to 16 November 1994 (Morgan Kaufmann, San Mateo, CA, 1994), pp. 843–850.
29. H. A. Rowley, S. Baluja, T. Kanade, *Technical Report CMU-CS-95-158* (Carnegie Mellon University, Pittsburgh, PA, 1995); H. Murase and S. K. Nayar, *Int. J. Comput. Vision* **14**, 5 (1995).
30. T. F. Cootes and C. J. Taylor, in *Proceedings British Machine Vision Conference*, Leeds, UK, 22 to 24 September 1992 (Springer-Verlag, London, 1992), pp. 266–275.
31. A. Shashua, *A.I. Technical Report 1401* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1992); thesis, Massachusetts Institute of Technology (1992).
32. T. Poggio and T. Vetter, *A.I. Memo 1347* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1992).
33. M. Lades *et al.*, *IEEE Trans. Comput.* **42**, 300 (1993).
34. D. Beymer, *A.I. Memo 1537* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1995).
35. C. Tomasi and T. Kanade, *Int. J. Comput. Vision* **9**, 137 (1992).
36. A. Shashua, *IEEE Trans. Pattern Anal. Machine Intell.* **17**, 779 (1995).
37. S. Laveau and O. Faugeras, *Technical Report 2205* (Institut National de Recherche en Informatique et en Automatique, France, 1994); S. Avidan and A. Shashua, *Technical Report CIS-9602* (Technion, Haifa, Israel, 1996); T. Evgeniou, thesis, Massachu-

setts Institute of Technology (1996).

38. D. Beymer, A. Shashua, T. Poggio, *A.I. Memo 1431* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1993); T. Ezzat, thesis, Massachusetts Institute of Technology (1996); S. Lines, thesis, Massachusetts Institute of Technology (1996).

39. S. Seitz and C. Dyer, in *Proc. IEEE Workshop on the Representation of Visual Scenes*, Cambridge, MA, 24 June 1995 (IEEE Computer Society Press, Los Alamitos, CA, 1995), pp. 18–25; S. Chen and L. Williams, in *SIGGRAPH Proceedings*, Anaheim, CA, 1 to 6 August 1993 (Association for Computing Machinery, New York, 1993), pp. 279–288; L. McMillan and G. Bishop, in (*1*), pp. 39–46.

40. T. Werner, R. Hersch, V. Hlaváč, in (*14*), pp. 957–962.

41. T. Poggio and F. Girosi, *A.I. Memo 1140* (Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, 1989). See also T. Poggio and F. Girosi, *Science* **247**, 978 (1990).

42. F. Girosi, M. Jones, T. Poggio, *Neural Comput.* **7**, 219 (1995).

43. D. S. Broomhead and D. Lowe, *Complex Systems* **2**, 321 (1988).

44. The equation can be rewritten in matrix notation as

$$\mathbf{y}(\mathbf{z}) = \mathbf{C}\mathbf{g}(\mathbf{z}) \qquad (2)$$

where **g** is the vector with elements $g_i = G(\mathbf{z} - \mathbf{z}_i)$. If **G** is defined as the matrix with elements $G_{i,j} = G(\mathbf{z}_i - \mathbf{z}_j)$, then the "weights" **c** are "learned" from the examples by solving $\mathbf{G}\mathbf{c}_m = \mathbf{y}_m$. If **Y** is defined as the matrix in which column ℓ is the example $\mathbf{y}_\ell$ and **C** as the matrix in which row m is the vector $\mathbf{c}_m$, then the set of weights C is given by $\mathbf{C} = \mathbf{Y}\mathbf{G}^+$ ($\mathbf{G}^+$ is the pseudoinverse). The vector field **y** is approximated by the network as the linear combination of the example fields $\mathbf{y}_\ell$, that is,

$$\mathbf{y}(\mathbf{z}) = \mathbf{Y}\mathbf{G}^+\mathbf{g}(\mathbf{z}) = \sum_{\ell=1}^{N} b_\ell(\mathbf{z})\mathbf{y}_\ell \qquad (3)$$

which is the dual representation of Eq. 1. The b_ℓ depend on the chosen G, according to $\mathbf{b}(\mathbf{z}) = \mathbf{Y}\mathbf{G}^+\mathbf{g}(\mathbf{z})$. Thus, for any choice of the regularization network, the estimated output (vector) image is always a linear combination of example (vector) images with coefficients **b** that depend (nonlinearly) on the desired input value. The result is valid for all networks with one hidden layer and linear output units, provided that a L^2 criterion is used for training (*42*).

45. J. R. Bergen, P. Anandan, K. J. Hanna, R. Hingorani, in *Proceedings of the Second European Conference on Computer Vision*, Santa Margherita Ligure, Italy, 19 to 22 May 1992 (Springer-Verlag, Berlin, 1992), pp. 237–252. See also B. Lucas and T. Kanade, in *Proceedings IJCAI*, Vancouver, British Columbia, 24 to 28 August 1981 (American Association for Artificial Intelligence, Menlo Park, CA, 1981), pp. 674–679.

46. Dense correspondence can be computed at video rate by machines such as the Carnegie Mellon University video-rate stereo machine [T. Kanade, H. Kano, S. Kimura, A. Yoshida, K. Oda, in *Proceedings of the International Conference on Intelligent Robotics and Systems*, Pittsburgh, PA, 5 to 9 August 1995 (IEEE Computer Society Press, Los Alamitos, CA, 1995), pp. 95–100] or the David Sarnoff Vision Front End [P. Anandan, P. Burt, J. Pearson, in *Proceedings of the Image Understanding Workshop*, Palm Springs, CA, 12 to 15 February 1996 (Morgan Kaufmann, San Mateo, CA, 1996)].

47. S. Mukherjee and S. K. Nayar, in (*14*), pp. 794–800.

48. The width for the *i*th Gaussian, σ_i, is estimated from the average distance from $\mathbf{s}_i$ to its neighbors, and then the $\mathbf{c}_i$ are computed by solving the linear system of Eqs. 1 over the training data.

49. G. Wolberg, *Digital Image Warping* (IEEE Computer Society Press, Los Alamitos, CA, 1990).

50. S. Librande, thesis, Massachusetts Institute of Technology (1992). See also the Cartoon-o-Matic Web page, http://www.nfx.com.

51. A different approach to rendering a face with a focus on the text-to-speech component can be found in K. Waters and T. M. Levergood, *Technical Report CRL 93/4* (Cambridge Research Laboratory, Digital Equipment Corporation, Cambridge, MA, 1993).

52. J. Y. A. Yang and E. H. Adelson, *IEEE Trans. Imag. Proc.* **3**, 625 (1994).

53. Such a model constitutes a special case of the general concept of flexible templates parameterized by the coefficients of the prototypes [for different types of flexible templates, see D. J. Burr, *IEEE Trans. Pattern Anal. Machine Intell.* **3**, 708 (1981); M. A. Fischler and R. A. Eschlager, *IEEE Trans. Comput.* **C-22**, 67 (1973); U. Grenander, Y. Chow, D. M. Keenan, *Hands: A Pattern Theoretic Study of Biological Shapes* (Springer-Verlag, New York, 1991); G. E. Hinton, C. K. I. Williams, M. D. Revow, in *Advances in Neural Information Processing Systems 4*, J. Moody, S. Hanson, R. Lippmann, Eds. (Morgan Kaufmann, San Mateo, CA, 1992), pp. 512–519; A. Yuille, *Neural Comput.* **2**, 1 (1990); P. W. Hallinan, thesis, Harvard University (1995); M. Kass, A. Witkin, D. Terzopoulos, in *Proceedings of the First International Conference on Computer Vision*, London, 8 to 11 June 1987 (IEEE Computer Society Press, Washington, DC, 1987)]. The flexible model of Jones and Poggio [M. Jones and T. Poggio, in preparation; see also (*14*)] is learned from a small number of vectorized prototype images. Given prototype images $I_0, \ldots, I_{N-1}$ and their shape vectors $\mathbf{s}_i$ and texture vectors $\mathbf{t}_i$ relative to I_0, where $\mathbf{s}_i = (\Delta\mathbf{x}_i, \Delta\mathbf{y}_i)$ and $\mathbf{t}_i = I_i[\mathbf{x} + \Delta\mathbf{x}_i(x,y), \mathbf{y} + \Delta\mathbf{y}_i(x,y)]$, the model is

$$I^{\text{model}}(\hat{x},\hat{y}) = \sum_{i}^{N-1} b_i\mathbf{t}_i \qquad (4)$$

with

$$\hat{x} = x' + p_0x' + p_1y' + p_2 \qquad (5)$$

$$\hat{y} = y' + p_3x' + p_4y' + p_5 \qquad (6)$$

$$x' = x + \sum_{i=1}^{N-1} c_i\Delta\mathbf{x}_i(x,y) \qquad (7)$$

$$y' = y + \sum_{i=1}^{N-1} c_i\Delta\mathbf{y}_i(x,y) \qquad (8)$$

The two linear combinations (for shape and texture, respectively) use the same set of prototype images but two different sets of coefficients. The parameters of the model are the c_i's, b_j's and p_k's of Eqs. 4 to 8. Given a novel image, the parameters that correspond to the best fit of the model are estimated by minimizing an error measure between the novel image and the flexible model after rendering it as an image. For instance, parameters can be found that minimize the function

$$E(\mathbf{c},\mathbf{p},\mathbf{b}) = \frac{1}{2}\sum_{x,y}[I^{\text{novel}}(\hat{x},\hat{y}) - I^{\text{model}}(\hat{x},\hat{y})]^2 \qquad (9)$$

54. A. Blake, R. Curwen, A. Zisserman, *Int. J. Comput. Vision* **11**, 127 (1993).

55. Other matching techniques for object recognition are closely related to optical flow algorithms, even if the connection is not always made explicit. For instance, the functional minimized in the technique of von der Marlsburg and co-workers (*33*) is closely related to the regularization functional [T. Poggio, V. Torre, C. Koch, *Nature* **317**, 314 (1985)] used in many optical flow algorithms.

56. Optical flow techniques work between images with sufficiently low disparities. When correspondence is desired between images that are quite different, other techniques are required [P. Lipson, thesis, Massachusetts Institute of Technology (1993); I. Bachelder, thesis, Massachusetts Institute of Technology (1991); A. Witkin, D. Terzopoulos, M. Kass, *Int. J. Comput. Vision* **1**, 133 (1987)]. We do not imply that dense correspondence is always needed in vision and graphics tasks. Symbolic techniques should be used in general to guide the low-level correspondence algorithms we have used in this work.

57. This procedure is similar to training a network with input-output pairs of prototypical views representing each prototype in the initial and in the desired pose. Then, for a new input image, the network synthesizes a virtual view in the desired pose. If the network is trained with pairs of prototype images as inputs (represented as 2D shape vectors) and their 3D shape as output, it will effectively compute 3D shape for novel images of the same class [see (*23, 32*) and compare the approach of J. Atick, P. Griffin, N. Reidlich, *Network: Comput. Neural Syst.* **7**, 1 (1996)]. The linear class assumption induces a linear combination of the 2D shape vectors but not of the corresponding texture vectors, not even for Lambertian reflectance and uniform albedo (A. Yuille, personal communication).

58. An approach in the same spirit was used earlier by Pomerleau to increase the number of training examples in his system that learns to steer a vehicle from images of the road (*7*).

59. It is not surprising that a small set of corresponding images constitutes a powerful representation for recognition and graphics, because complete 3D structure can be recovered from a small number of views in correspondence (three orthographic and two perspective views are sufficient).

60. T. Poggio and S. Ullman, in preparation.

61. H. H. Bülthoff and S. Edelman, *Proc. Natl. Acad. Sci. U.S.A.* **89**, 60 (1992); N. Logothetis, J. Pauls, T. Poggio, *Curr. Biol.* **5**, 552 (1995).

62. We thank H. Bülthoff, F. Girosi, P. Dayan, M. Jordan, T. Vetter, T. Sejnowski, D. Glaser, A. Shashua, E. Grimson, M. Jones, R. Romano, and especially S. Ullman, C. Tomasi, C. Koch, A. Blake, D. Terzopoulos, and T. Kanade for reading the manuscript and for many insightful and constructive comments. Sponsored by grants from the Office of Naval Research and the Advanced Research Projects Agency under contracts N00014-93-1-0385 and N00014-92-J-1879. Support for CBCL is provided in part by grants from NSF under contract ASC-9217041, by the Multidisciplinary University Research Initiative under contract N00014-95-1-0600, and by Siemens, Daimler-Benz, Eastman Kodak, and ATR. T.P. is supported by the Uncas and Helen Whitaker Chair at Whitaker College, Massachusetts Institute of Technology. D.B. was supported in part by a Howard Hughes Doctoral Fellowship from the Hughes Aircraft Company.