MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY
and
CENTER FOR BIOLOGICAL INFORMATION PROCESSING
WHITAKER COLLEGE

A.I. Memo No. 1354 February, 1992
C.B.I.P. Paper No. 71

A Novel Approach to Graphics

Tomaso Poggio and Roberto Brunelli

Abstract

We propose a new, memory-based approach to graphic animation of 2D and 3D objects. Instead of the standard paradigm in computer graphics of 3D modeling, physical simulation and rendering we propose to use a set of 2D views to synthesize a network that generates desired images and image sequences. The approach is based on a new approximation technique which can be regarded as a scheme for learning from examples.

This memo is a slightly modified version of an unpublished manuscript (May, 1990) that represents the basis for a patent application on "Memory Based Method and Apparatus for Computer Graphics," filed by the Massachusetts Institute of Technology and the Istituto per la Ricerca Scientifica e Tecnologica, Trento, Italy, on 13 January, 1992, No. 07/819,767. The memo describes research done at I.R.S.T. in 1990. 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 Massachusetts Institute of Technology. R. Brunelli is at I.R.S.T.

1 Introduction

The paradigm that currently dominates work in graphic animation consists of 3D modeling of surfaces, physically based simulation of movements and rendering. While this approach is clearly fundamentally correct and potentially powerful, current results are still far from obtaining general purpose, realistic image sequences. In this paper we propose a parallel approach which is a short-cut to the ultimate goal of creating simulated worlds but also has interesting aspects in its own (it may be better for production of cartoons for instance).

We propose a $2\frac{1}{2}$D memory based approach to 3D graphics and animation. The main idea is to use a few views of an object such as a person to generate intermediate views, under the control of a set of parameters, such as the 3D pose of the object or the expression of a face. The process can be framed as consisting of a stage of learning from examples – in which each of the given views is associated with its parameter values (such as the pose) – followed by a stage of generalization or interpolation in which new views are generated for desired values of the parameters. The set of views used during the learning stage are real high resolution images of the object.

The learning technique we propose is that of Hyper Basis Functions (HBF), which is equivalent to the network of Fig.1. A special case of HBFs are the so called Radial Basis Functions, used in this paper, in which an unknown multivariate function is approximated by the superposition of a given number of radial functions whose centers are located on the points of the learning set (see [7] and Appendix 1).

The technique of Fig. 1 can synthesize *smooth* mappings between the input parameters such as the pose of a 3D object and its views. As shown in Appendix 2 and as it is intuitively clear, the mapping between input parameters (for instance Euler angles) and pixel grey-level or color values is not smooth in general. For this reason, we use the network of Fig.1 to produce *2D views* of the object that are not images but rather consist of the image coordinates of a set of characteristic knots, such as body junctions, face features etc., that in this paper we also call *control points.* A similar definition of 2D views has been used in the inverse problem – the problem of object recognition and pose estimation (see [4, 3]). Each view is similar to a skeleton of the object that can be used as a basis for texture mapping in order to recreate a high resolution image of the object. Thus a realistic –

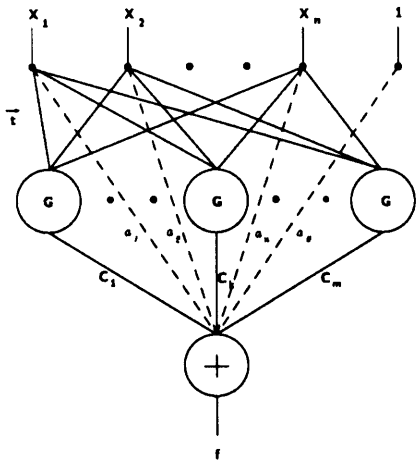

Figure 1: A network diagram of the Hyper Basis Functions technique

even if only approximate, from the point of view of physical truth – rendering can be obtained without an explicit 3D physically-based model of the object.

The space of all perspective projections can be factored into a set of *characteristic views*. The correspondence between control points configuration and characteristic views can again be learned from examples. The problem of occlusions could, in principle, be solved by the same paradigm.

The approach is said to be memory based because a set of 2D views is used for texture mapping instead of an explicit 3D model which is rendered each time. Notice that a limit case of the network of Fig. 1 (for Gaussians with σ approaching zero) is a look-up table, in which case the network will generate the view, if it exists in the training set, that exactly corresponds to the input parameters. The reason it is neither 3D nor 2D is that a set of characteristic views is more than 2D (a single perspective view) but less than a full 3D model (every perspective view is computable). We emphasize that the set of characteristic knots does not correspond to a wire-frame solid model: the control points are 2D points, defined on the image plane.

2 The memory-based approach to graphics

Let $\{C_i\}_{i \in I} \in R^2$ be a set of control points. To each different object (such as a cartoon character) or movement (such as jump, walk, run) we can associate

a map:

$$M_\alpha : \bar{x} \in R^n \rightarrow \{C_i(\bar{x})\}_{i \in I}, \alpha \in \{\text{jump}, \ldots\} \quad (1)$$

where $\bar{x} \in R^n$ represents a vector in the space of the parameters specifying the object "state" or the given movement. From each set $\{C_i\}_{i \in I}$ we can obtain another set transforming from absolute coordinates to barycentric coordinates. This new set, $\{C_i^B\}_{i \in I}$, can be used to learn the correspondence from control point configuration to characteristic views. This mapping can be seen as a simplified attitude recovery. Barycentric coordinates are used because this correspondence is intrinsic to the object, while movement of the object is relative to the environment. This map can be shared by a class of objects with a reasonable degree of approximation. Another map that can be synthesized and shared is the one giving relative depth of the different control points. This map allows the use of simple z-buffering techniques in texture mapping. In order to share these maps we must have a map from the control points of the instances of the object to those of the class prototype (and its inverse). In the following, we give some examples.

2.1 Simple examples

2.1.1 One-dimensional in-betweening

Let U_i be an element used for texture mapping (it could be a square or a triangle or more complex polygon). Our object $\mathcal{O}$ is then composed of a given number of elements and we can represent it as:

$$\mathcal{O} = \{U_i\}_{i \in I} \quad (2)$$

The animation of this object, using a particular movement map, amounts to introducing a known temporal dependence:

$$\mathcal{O}(t) = \{U_i(t)\}_{i \in I} \quad (3)$$

Each element $U_i(t)$ is computed using the map M for the given movement (each single point of unit U_i is mapped by this function giving the transformed unit). An example of such a mapping is given in Fig. 2.

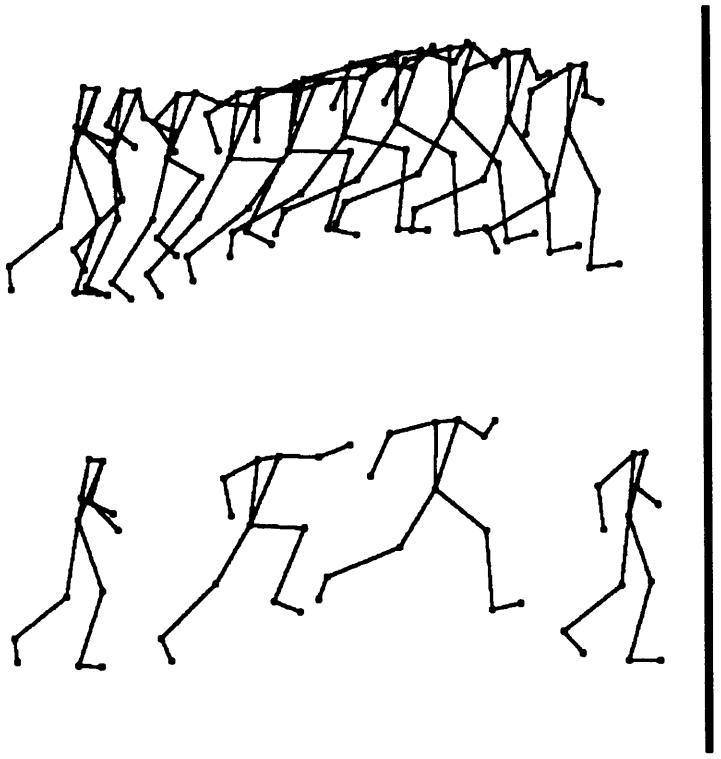

Figure 2: Jump in-betweening (lower half: frames used as examples. Upper half: in-betweening using RBF interpolation)

2.1.2 Multidimensional interpolation

A slightly more complex application, that better shows the new features of our proposal, is presented in Fig. 3. In this case the parameter space, expression and poses of a face, is bidimensional and not homogeneous. A set of control points was manually matched among five examples (appearing as framed drawings in Fig. 3) obtained by a perspective projection of a wire frame model. The resulting sets of two dimensional points were used as examples for an in-betweening RBF network. After a learning phase, the network was able to generalize and to produce realistic drawings for poses and expressions it had never seen before.

2.2 From views to grey-levels: texture mapping

The next step is texture mapping. Let us assume for simplicity that we need just one characteristic view. Let R_i be the i-th element in the characteristic view. We can consider the characteristic view as a reference frame from which texture mapping is performed. Texture mapping is fully specified, in our simplified approach, by the transformation that maps the reference element, such as R_i, into a transformed instance of the same element, such as $U_i(t)$ [1]. Let us denote with $T_i(t)$ this set of transformations (one for each element):

$$T_i(t) = T(R_i, U_i(t)) \tag{4}$$

In this way it is possible to generate desired views of the object – with the HyperBF technique – from a set of examples and to produce images from the views – with texture mapping.

2.3 One example of many obvious extensions

Suppose we have synthesized a network capable of generating views of John walking. As shown in Fig. 6 and explained above, this can be done from a set of examples, that is views. Assume that I would like to generate images of a different person – say Jayne – walking. Of course, I could repeat the

[1] An affine transformation can be used for triangular elements while a bilinear transformation is needed for quadrilaterals. For a brief review of such simple deformations see [5]

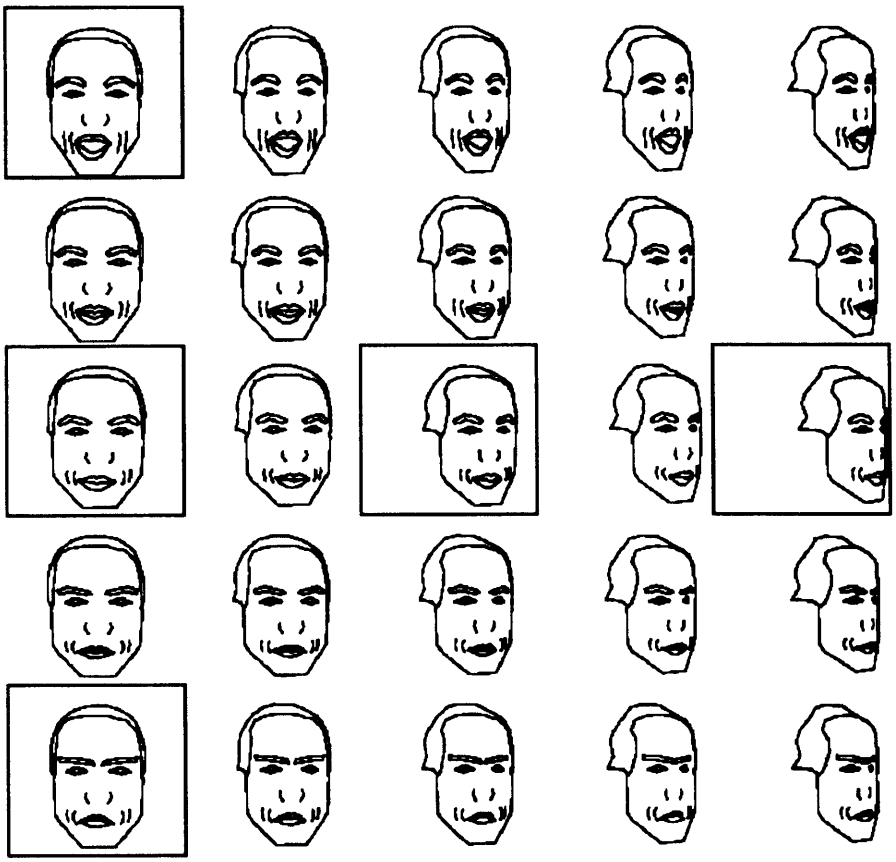

Figure 3: An example of in-betweening in a two dimensional input space of pose and expression parameters: vertical axis represent expression from *astonishment* to *anger* while horizontal axis represent rotation from 0^o to 60^o degrees. The framed drawings represent the examples used for learning: all of the remaining drawings are generated by the network. A large number of additional ones can be generated as easily.

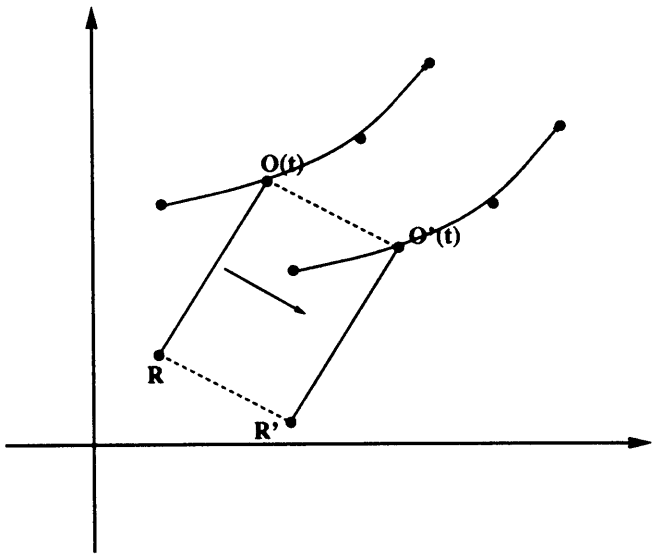

Figure 4: Parallel deformation of the prototype movement

same procedure. But shortcuts are possible. Consider the following case that exploits the network synthesized for a prototype (John) to animate another object of the same class (Jayne) with minimal additional information.

The simplest way of mapping Jayne onto an available prototype (John) is probably that of *parallel deformation.* The first step consists in transforming the control points of the reference frame of the prototype and of the new object to their baricentric coordinates. This operation allows us to separate the motion of the baricenter, which we can consider to be intrinsic to the learned movement, from the motion around it, which depends on the particular instance we map. The set of the control points is then embedded in an $2n$-dimensional space. A parallel deformation is defined by (see Fig.4):

$$\mathcal{O}'_B(t) = \mathcal{R}'_B + [\mathcal{O}_B(t) - \mathcal{R}_B(t)] \tag{5}$$

where the subscript B means that the control points are considered in their baricentric coordinates, and the objects are considered embedded in a $2n$-dimensional space (n being the number of control points). From t we can obtain the map of Jayne at time t transforming back into the set of control points and displacing the result by the baricenter of the prototype at time t

(see Fig. 5). The reason this type of mapping is called *parallel deformation* is the following. If we look at the $2n$-dimensional vectors, we see that views of Jayne are obtained adding the displacement from the reference frame of the prototype to its version at time t to the characteristic view of the instance: the deformations (i.e. the difference between the objects at time t and its characteristic view) of the prototype and of Jayne are then parallel by construction [2]. The examples show the animation of John (the prototype) (see Fig. 6) and that of Jayne (the new object) (see Fig. 7) of the same class obtained through the above steps.

A similar approach can be applied to the expression-pose space, so that a face of a person can be animated by the expression of a prototype (possibly a professional actor): possible applications to the production of cartoons, to the animation of real subjects or a mixture of the two are natural.

3 Conclusions

One of the obvious applications of these techniques is animation of cartoons characters. The ultimate goal is the synthesis of network that capture, say Donald Duck, in the sense that they can generate any desired view of Donald Duck starting from a "training set" consisting of a large set of available views of Donald Duck. Another natural application is teleconferencing. The possibility to animate an image, say of a face, using a limited number of control points reduces drastically the amount of information that needs to be transmitted. The automatic tracking of control points is possible at present, even if their automatic identification still presents a number of difficult technical problems.

4 Appendix 1: the HyperBF technique

This section describes a technique for synthesizing the approximation modules discussed above through learning from examples. We first explain how to rephrase the problem of learning from examples as a problem of approxi-

[2] A further improvement would be the modulation of the difference vector by the change in size of the protoptype vector. The resulting construction cannot be named *parallel* any longer.

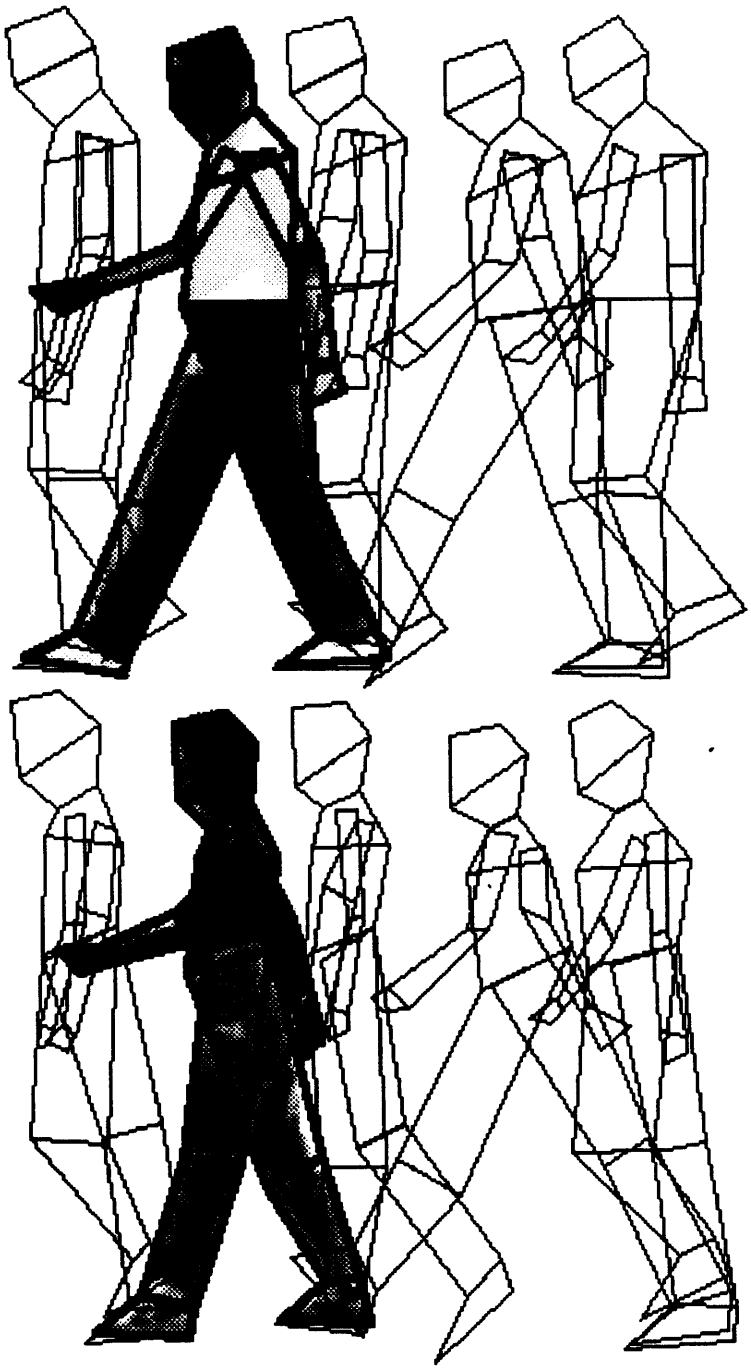

Figure 5: Motion of a new instance (down) using parallel deformation of a prototype movement (up)

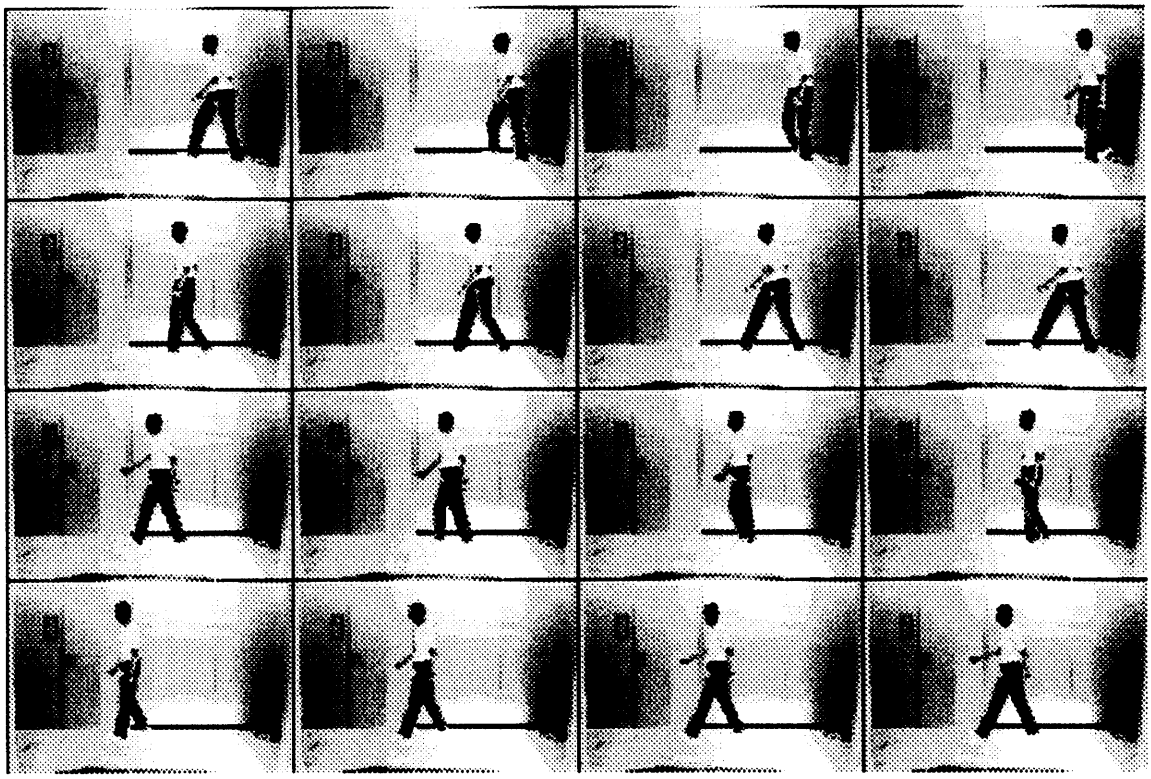

Figure 6: Prototype animation based on five sample frames: the other nine frames are synthesized through the technique of this paper.

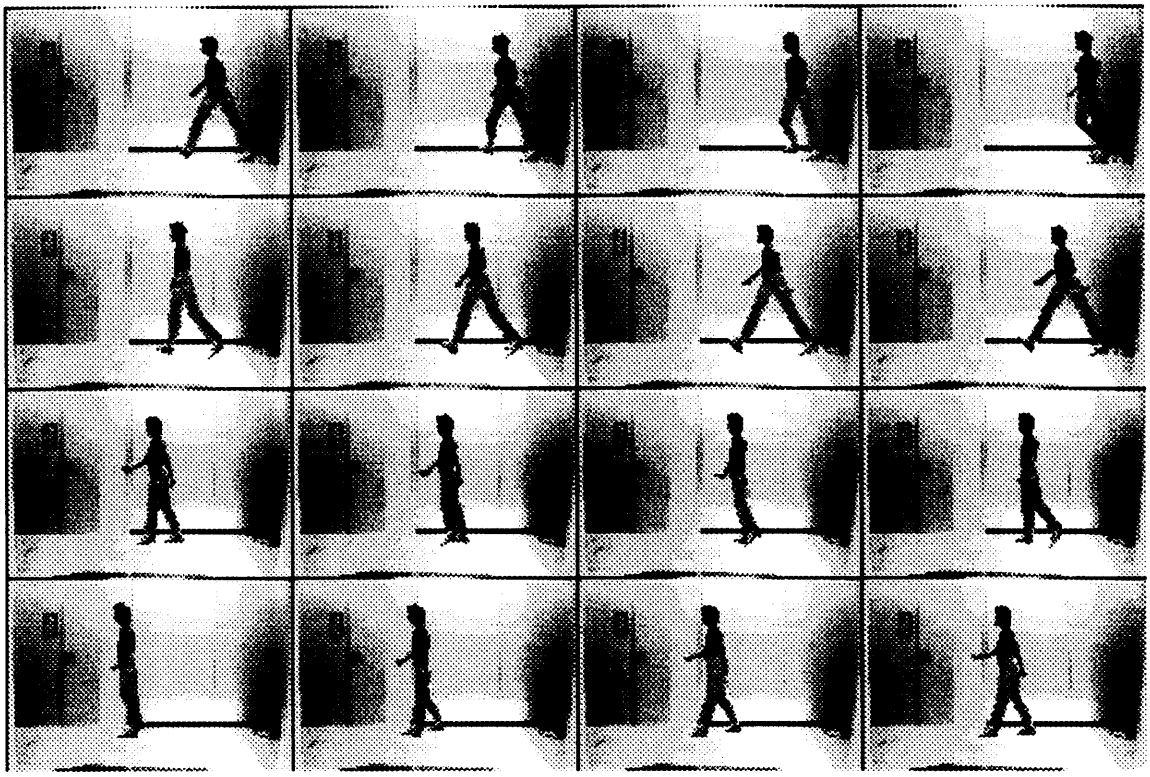

Figure 7: Animation of a new instance using the prototype learned movement (Fig. 6). All images here are synthetic apart from the first one in the sequence, which is the only real image.

mating a multivariate function. The material in this section is from Poggio and Girosi (see [10]) where more details can be found.

To illustrate the connection, let us draw an analogy between learning an input-output mapping and a standard approximation problem, 2-D surface reconstruction from sparse data points. *Learning* simply means collecting the *examples*, i.e., the input coordinates x_i, y_i and the corresponding output values at those locations, the heights of the surface d_i. *Generalization* means estimating d at locations x, y where there are no examples, i.e. no data. This requires interpolating or, more generally, approximating the surface (i.e. the function) between the data points (interpolation is the limit of approximation when there is no noise in the data). In this sense, learning is a problem of *hypersurface reconstruction* (see [11, 6]).

From this point of view, learning a smooth mapping from examples is clearly ill-posed, (see [14]) in the sense that the information in the data is not sufficient to reconstruct uniquely the mapping in regions where data are not available. In addition, the data are usually noisy. *A priori* assumptions about the mapping are needed to make the problem well-posed. One of the simplest assumptions is that the mapping is *smooth*: small changes in the inputs cause a small change in the output. Techniques that exploit smoothness constraints in order to transform an ill-posed problem into a well-posed one are well known under the term of *regularization theory* (see [14, 13, 1]). The solution to the approximation problem given by regularization theory can be expressed in terms of a class of multilayer networks that we call regularization networks or Hyper Basis Functions (see Fig. 1). The main result (see [8]) is that the regularization approach is equivalent to an expansion of the solution in terms of a certain class of functions:

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i G(\mathbf{x}; \boldsymbol{\xi}_i) + p(\mathbf{x}) \tag{6}$$

where $G(\mathbf{x})$ is one such function and the coefficients c_i satisfy a linear system of equations that depend on the N "examples", i.e. the data to be approximated. The term $p(\mathbf{x})$ is a polynomial that depends on the smoothness assumptions. In many cases it is convenient to include up to the constant and linear terms. Under relatively broad assumptions, the Green's function G is radial and therefore the approximating function becomes:

$$f(\mathbf{x}) = \sum_{i=1}^{N} c_i G(\|\mathbf{x} - \boldsymbol{\xi}_i\|^2) + p(\mathbf{x}), \tag{7}$$

which is a sum of radial functions, each with its *center* $\boldsymbol{\xi}_i$ on a distinct data point and of constant and linear terms (from the polynomial, when restricted to be of degree one). The number of radial functions, and corresponding centers, is the same as the number of examples.

The derivation in [8] shows that the type of basis functions depends on the specific *a priori* assumption of smoothness (see [12, 8]). Depending on it one obtains the Gaussian $G(r) = e^{-(\frac{r}{c})^2}$, the well known "thin plate spline" $G(r) = r^2 \ln r$, and other specific functions, radial and not. As observed by Broomhead and Lowe (see [2]) in the radial case, a superposition of functions like Eq. 6 is equivalent to a network of the type shown in Fig. 1. The interpretation of Eq. 7 is simple: in the 2D case, for instance, the surface is approximated by the superposition of, say, several two dimensional Gaussian distributions, each centered on one of the data points.

The network associated with Eq. 7 can be made more general in terms of the following extension

$$f^*(\mathbf{x}) = \sum_{\alpha=1}^{n} c_\alpha G(\|\mathbf{x} - \mathbf{t}_\alpha)\|_W^2) + p(\mathbf{x}) \tag{8}$$

where the parameters $\mathbf{t}_\alpha$, that we call "centers", and the coefficients c_α are unknown, and are in general much fewer than the data points ($n \leq N$). The norm is a *weighted norm*

$$\|\mathbf{x} - \mathbf{t}_\alpha)\|_W^2 = (\mathbf{x} - \mathbf{t}_\alpha)^T W^T W (\mathbf{x} - \mathbf{t}_\alpha) \tag{9}$$

where W is an unknown square matrix and the superscript T indicates the transpose. In the simple case of diagonal W the diagonal elements w_i assign a specific weight to each input coordinate, determining in fact the units of measure and the importance of each feature (the matrix W is especially important in cases in which the input features are of a different type and their relative importance is unknown, see [9]). Equation 8 can be implemented by the network of Fig. 1.

4.1 Learning

Iterative methods can be used to find the optimal values of the various sets of parameters, the c_α, the w_i and the $\mathbf{t}_\alpha$, that minimize an error functional on the set of examples. Steepest descent is *the* standard approach that requires calculations of derivatives. An even simpler method that does not require calculation of derivatives (suggested and found surprisingly efficient in preliminary work by Caprile and Girosi, personal communication) is to look for random changes (controlled in appropriate ways) in the parameter values that reduce the error. We define the error functional – also called energy – as

$$H[f^*] = H_{\mathbf{c},\mathbf{t},\mathbf{W}} = \sum_{i=1}^{N} (\Delta_i)^2,$$

with

$$\Delta_i \equiv y_i - f^*(\mathbf{x}) = y_i - \sum_{\alpha=1}^{n} c_\alpha G(\|\mathbf{x}_i - \mathbf{t}_\alpha\|^2_{\mathbf{W}}).$$

In the first method the values of c_α, $\mathbf{t}_\alpha$ and $\mathbf{W}$ that minimize $H[f^*]$ are regarded as the coordinates of the stable fixed point of the following dynamical system:

$$\dot{c}_\alpha = -\omega \frac{\partial H[f^*]}{\partial c_\alpha}, \quad \alpha = 1, \ldots, n$$

$$\dot{\mathbf{t}}_\alpha = -\omega \frac{\partial H[f^*]}{\partial \mathbf{t}_\alpha}, \quad \alpha = 1, \ldots, n$$

$$\dot{\mathbf{W}} = -\omega \frac{\partial H[f^*]}{\partial \mathbf{W}},$$

where ω is a parameter. The derivatives are rather complex (see Poggio and Girosi, 1990a and Notes section).

The second method is simpler: random changes in the parameters are made and accepted if $H[f^*]$ decreases. Occasionally, changes that increase $H[f^*]$ may also be accepted (similarly to Metropolis algorithm).

5 Appendix 2: A naive approach

Is it possible to learn the movement of an object looking directly at the pixel level of the images? A little thought shows that this approach is not likely to work. The most basic requirement of learning – and the one assumed in regularization techniques of which the RBF algorithm is a special case – is the smoothness of the underlying map, which allows generalization from the available examples.

The intensity at a given pixel can undergo sudden changes during the movement of the represented objects. There is no way we could *a-priori* interpolate the intensities to be displayed at different times and therefore generalize. The main reason why direct learning of the image at the pixel level does not work, stems from the discontinuos nature of the underlying map. Learning a mapping that is not sufficiently smooth is hopeless because it requires a very large number of examples. The use of a Fourier transformed image does not help: a Fourier transformed image is a linear mapping of the original image, since the Fourier operator commutes with the RBF interpolation operator. The example of Fig.8 illustrates two interesting effects of using this simple (and wrong) approach. The strip represents a wire frame cube from different viewpoints. The dark pictures are due to the lack of generalization from the available examples (uniformly distributed on the viewing sphere) while blurring and superposition of views is due to the inability to recover correctly the discontinuos intensity changes.

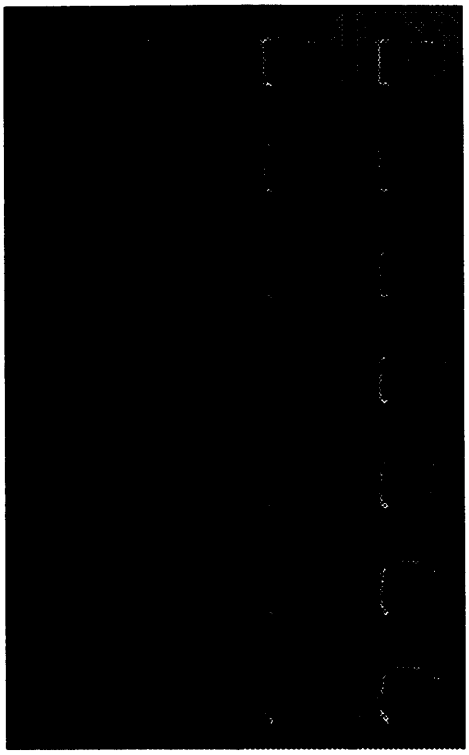

Figure 8: Direct bitmap learning. The different columns are obtained using different learning parameters: for the first three columns gaussian RBF was used with increasing sigmas (increasing generalization) while the last column was obtained using multiquadrics.

References

[1] M. Bertero, T. Poggio, and V. Torre. Ill-posed problems in early vision. *Proceedings of the IEEE*, 76:869–889, 1988.

[2] D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[3] R. Brunelli and T. Poggio. Use of rbf in real object recognition. Technical Report 9011-09, I.R.S.T, 1990.

[4] S. Edelman and T. Poggio. Bringing the grandmother back into the picture: a memory-based view of object recognition. A.I. Memo 1181,

Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1990.

[5] Z. C. LI, Y. Y. Tang, T. D. Bui, and C. Y. Suen. Shape transformation models. *Int. Jour. of Pattern Recognition and Artificial Intelligence*, Vol. 4(1):65–94, 1990.

[6] S. Omohundro. Efficient algorithms with neural network behaviour. *Complex Systems*, 1:273, 1987.

[7] T. Poggio and F. Girosi. A theory of networks for approximation and learning. Technical Report A.I. Memo No. 1140, Massachusetts Institute of Technology, 1989.

[8] 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.

[9] T. Poggio and F. Girosi. Extension of a theory of networks for approximation and learning: dimensionality reduction and clustering. A.I. Memo 1167, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1990.

[10] T. Poggio and F. Girosi. A theory of networks for learning. *Science*, 247:978–982, 1990a.

[11] T. Poggio and the staff. MIT progress in understanding images. In *Proceedings Image Understanding Workshop*, Cambridge, MA, April 1988. Morgan Kaufmann, San Mateo, CA.

[12] T. Poggio and the staff. M.I.T. progress in understanding images. In *Proceedings Image Understanding Workshop*, pages 56–74, Palo Alto, CA, May 1989. Morgan Kaufmann, San Mateo, CA.

[13] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319, 1985b.

[14] A. N. Tikhonov and V. Y. Arsenin. *Solutions of Ill-posed Problems.* W.H.Winston, Washington, D.C., 1977.