Biophysical Journal Volume 66 June 1994 1804–1814

Pattern Recognition and Classification of Images of Biological Macromolecules using Artificial Neural Networks

R. Marabini and J. M. Carazo
Centro Nacional de Biotecnología (CSIC), Universidad Autónoma, Campus de Canto Blanco, 28049 Madrid, Spain

ABSTRACT The goal of this work was to analyze an image data set and to detect the structural variability within this set. Two algorithms for pattern recognition based on neural networks are presented, one that performs an unsupervised classification (the self-organizing map) and the other a supervised classification (the learning vector quantization). The approach has a direct impact in current strategies for structural determination from electron microscopic images of biological macromolecules. In this work we performed a classification of both aligned but heterogeneous image data sets as well as basically homogeneous but otherwise rotationally misaligned image populations, in the latter case completely avoiding the typical reference dependency of correlation-based alignment methods. A number of examples on chaperonins are presented. The approach is computationally fast and robust with respect to noise. Programs are available through ftp.

INTRODUCTION

The range of problems associated with detecting and recognizing patterns is vast. It is therefore not surprising that the number of approaches that have been proposed over the years is large. Focusing the attention on the field of electron microscopy of biological macromolecules, the problem we address here may be described by the following sequence of events: given a sample containing either a biochemically homogeneous population of macromolecules or a heterogeneous one, obtain electron microscope images of these specimens and extract the maximum amount of relevant two-dimensional and/or three-dimensional biological information by means of image processing operations performed on the whole set of images (for a review, see Frank, 1990). Images are in general very noisy, so the processing techniques have to be very robust.

Two main questions related to pattern classification appear when addressing this type of problem. The first one is the assessment of the homogeneity of the image data set (note that a biochemically homogeneous population does not necessarily render a homogeneous set of images, since different two-dimensional views of the same three-dimensional structure may exist). If a number of different image classes are detected at this stage, it is then necessary to devise methods to classify the original data set into these classes. This step is usually carried out by tools such as multivariate statistical analysis (MSA) or the related principal component analysis (PCA) (for a review, see Frank et al., 1988b).

The second main problem appears after this classification, since now the classes must be aligned in such a way that an averaging process performed over the images within each class renders new information with an enhanced signal-to-noise ratio. The alignment is usually carried out by cross-correlation with respect to a reference image. Unfortunately, a clear-cut separation between these two problems apparently does not exist.

The difficulty that arises when trying to separate the problem of classification from the one of alignment is that differences between images may be due either to genuine differences or to positional factors, such as rotation and translation. Ideally, we would prefer to classify an image data set without considering the latter as genuine differences. In principle, methods to define a description of an image independent of its relative position are possible, but they do not seem to work for general cases involving very noisy images (Schatz and Van Heel, 1990; Frank et al., 1992; Marabini and Carazo, signal proc., in press). It is therefore necessary to first "align" the images before classification; however, aligning is only well defined within a homogeneous data set. Clearly, the problem is recursive and the single practical approach to date is to iterate between alignment with respect to some reference, an always potentially problematic step (Boekema et al., 1986), and a posteriori classification.

Topics of recognition and classification have been previously treated in this context by van Heel (1986), using hierarchical ascendant classification; Frank et al. (1988a), using a hybrid *k*-means and ascendant classification approach; Wagenknecht et al. (1989) and Carazo et al. (1989), using directly MSA to analyze rather continuous structural changes; and Carazo et al. (1990), using fuzzy sets, among other authors (for a review, see Frank, 1990).

The approach we propose in this work is conceptually different to previous ones in that it uses principles of neural networks to accomplish the classification task (for a review of neuronal networks see Lippmann, 1987). In essence, we use a particular type of neural network known as the self-organizing map (Kohonen, 1990) and we show that this method works remarkably well when confronted with a variety of tasks, from the classification of a heterogeneous image data set after a first cycle of refinement, revealing subtle structural variations, to the direct reference-free alignment of a homogeneous set of particle images through a

Received for publication 22 October 1993 and in final form 10 March 1994.
Address reprint requests to Carazo, J. M., Centro Nacional de Biotecnología (CSIC), Universidad Autónoma, Campus de Canto Blanco, 28049 Madrid, Spain. Tel.: 34-1-585-4510, 34-1-585-4543; Fax: 34-1-585-4506; E-mail: carazo@samba.cnb.uam.es.

0006-3495/94/06/1804/11 $2.00

classification-based angular assignment. The method is very fast, taking only a few minutes on a modern workstation to classify hundreds of images.

The use of neural networks is increasing in all those areas where, as in image processing, many solutions can be simultaneously tested and large computer resources are needed. The intrinsic parallel processing of the data and the ability to adapt makes the neural network potentially faster and more robust than the classical sequential process. Furthermore, the self-organizing maps are not parametric and need very little a priori knowledge.

The algorithm we present here possesses a number of interesting properties that may solve some of the problems encountered when following the procedures described above. As we will show, if the image data set is homogeneous, then the method is able to directly analyze sets of images that have not been rotationally aligned, by performing a classification based on the particle rotation angle, therefore eliminating the reference problem. Alternatively, if the population of images is heterogeneous but they are known to be correctly aligned, then the algorithm concentrates on genuine particle differences by performing a classification without any prior data reduction step.

THEORETICAL BACKGROUND

The self-organizing algorithm maps a set of n-dimensional vectors onto a two-dimensional array of nodes in such a way that vectors projected onto adjacent nodes are more similar than vectors projected onto distant ones. Note that this algorithm does not work like the usual n-dimensional onto two-dimensional mapping programs (Radermacher and Frank, 1985) in the sense that the emphasis is not on approaching the mathematical distances between the vectors but in plotting similar vectors close to one another in the output space. In the same way, the self-organizing algorithm could be interpreted as a nonlinear projection of the probability density function of the n-dimensional input onto an output array of nodes. Accordingly, distances in the output space must be interpreted with caution, because they are not directly related to metric distances in the n-dimensional space.

The mapping is obtained iteratively by a sequence of steps. Each step requires the presentation of an input vector to the array of output nodes (each node has assigned a certain vector that we will refer in the following as "code vectors"). The input vector is used as argument to an activation function that estimates the similitude between the input and the code vectors. Finally, the most similar code vector as well as its neighborhood are adjusted such as to improve its response to any other similar input vector.

Following the description above, the input of the self-organizing map is the set of our experimentally obtained images, while the output is a two-dimensional array of nodes, with an image assigned to each of them. These assigned images start with some arbitrary value but, as the algorithm proceeds, will tend to approximate the probability density distribution of the input data (Kohonen, 1990). In this way, we always have two pieces of information within this scheme: one is provided directly by the $\mathbf{m}_j$'s (the code vectors), and the other is the final assignment of each input data to a given code vector, which results in a classification of the data set.

Kohonen (1990) showed that this algorithm has some interesting properties. First, it represents most faithfully those dimensions of the input space along which the variance in the sequence of inputs $\mathbf{x}_i$ is most pronounced. These will often correspond to the most important features of the inputs. Second, it tries to preserve continuity, i.e., it maps similar inputs $\mathbf{x}_i$ to neighboring locations. Finally, it reflects differences in the sampling density of the input space in a natural way: regions from which inputs have occurred more frequently are mapped onto larger domains and therefore with better resolution.

The basic idea of the self-organizing algorithm is as follows.

Step 0: Assume a sequence of input vectors $\mathbf{x}(t)$ with a probability density function δ, where t is the "time" coordinate. The input vectors $\mathbf{x}(t)$ are images randomly selected from our experimentally obtained images that are sequentially presented to the algorithm. Note that the expression $\mathbf{x}(t)$ does not refer to a change in the images themselves, but to a temporal evolution of the input data. In our application, this evolution refers to the sequential presentation of input vectors to the network. In this way, $\mathbf{x}(0)(\mathbf{x}(t), t = 0)$ refers to the first image presented to the network, $\mathbf{x}(1)(\mathbf{x}(t), t = 1)$ refers to the second image, etc.

Step 1: Assume a set of code vectors $\mathbf{m}_j(t)$ assigned to the output nodes. Suppose that $\mathbf{m}_j(t)$ has been initialized in some way.

Step 2: Each vector $\mathbf{x}(t)$ is now presented to the output set of code vectors at each successive instant of time.

Step 3: $\mathbf{x}(t)$ is compared simultaneously with each $\mathbf{m}_j(t)$, for example using a distance d_j. Note that although the distance d_j is a critical factor in the mapping algorithm, the results will not reflect the mathematical distance in the n-dimensional space between input images.

Step 4: The best matching $\mathbf{m}_j(t)$ is then selected.

Step 5: The selected $node_j$, that will be called $node_c$, is used to set a center around which a neighborhood N of nodes is defined by some function $h(r, t)$ that decreases with r and t, where r is the distance between nodes in the output plane (this means that the value of h will be higher the closer to $node_c$ we are, and that the neighborhood decreases in size as the algorithm proceeds). All the code vectors within this neighborhood will now be modified in such a way that they will tend to $\mathbf{x}(t)$ even more closely. Obviously, since $h(r, t)$ decreases with the radius r from $node_c$, nodes closer to $node_c$ will be more strongly modified. To make the whole procedure more stable, another function $\alpha(t)$ is also introduced. The action of this $\alpha(t)$ is to set an upper limit to the magnitude of the changes allowed at each presentation, and it decreases with t. $\alpha(t)$ takes values between 0 and 1; for $\alpha(t)$ close to 0 the magnitude of the changes will be very small, while for $\alpha(t)$ close to 1 the change will be so big as to almost

force the code vector $\mathbf{m}_c(t)$ to become equal to $\mathbf{x}(t)$. $\alpha(t)$ and $h(r, t)$ control the speed of the code vector updating. $\alpha(t)$ fixes an upper limit to the modification of $\mathbf{m}_j(t)$ in each step, avoiding the formation of locally but not globally ordered regions in the output plane. On the other hand, $h(r, t)$ is a sigmoidal function that incorporates a lateral inhibition acting over nodes that are some distance apart, causing the nearest nodes to be more strongly updated. The inhibition increases with the time.

Step 6: The procedure continues at step 2 for each $\mathbf{x}(t)$ until convergence.

See Table 1 for a summary of the procedure.

Mathematically, the self-organizing map algorithm can be described as a Markov process (Ritter and Schulten, 1988) whose states are the code vectors $\mathbf{m}_j$ and the transitions, which are described in step 5 (Table 1), are determined by the probability density distribution δ of the input $\mathbf{x}(t)$. The necessary and sufficient conditions to guarantee the convergence of the process are:

$$\lim_{t\to\infty} \int_0^t \alpha(t)\, dt = \infty \qquad \lim_{t\to\infty} \alpha(t) = 0$$

A full discussion on convergence and topological properties of the self-organizing map can be found in the work of Cottrell and Fort (1986) and Ritter and Schulten (1986, 1988).

When the number of steps is finite, as in all the real cases, the convergence is not fully guaranteed. Ritter and Schulten (1986) have reported the appearance of two different kinds of instabilities. The first type is related to the final position of the code vectors on the outplane, while the second one affects the code vector features themselves.

TABLE 1 Self-organizing map algorithm

Step no.	Process
0	Convert the images I_i into vectors $\mathbf{x}(t)$.
1	Initialize the $\mathbf{m}_j$ code vectors. Initialize the code vectors $\mathbf{m}_j$ associated with each output node to random values.
2	Present one input vector $\mathbf{x}(t)$.
3	Compute distance to all nodes at time t. Compute the distance d_j between the input vector $\mathbf{x}(t)$ and each output node j using $d_j = \|\mathbf{x}(t) - \mathbf{m}_j(t)\|$
4	Select output node with minimum distance. Select the output node as that node associated with the code vectors $\mathbf{m}_j(t)$ that minimizes d_j.
5	Update code vectors for node j and its neighborhood. $\mathbf{m}_j$ are updated for node j and all nodes in a neighborhood N whose radius is defined by some function $h(r, t)$ decreasing with r and t. $\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \alpha(t)h(r, t)(\mathbf{x}(t) - \mathbf{m}_j(t))$ where $\alpha(t)$ is a decreasing function that controls the magnitude of the changes with the time ($0 < \alpha(t) < 1$) and $h(r, t)$ is a sigmoidal function that controls the neighborhood of the node j in which the code vectors are updated.
6	Repeat (go to step 2) until process converges.

As to the first type of instability, it should be noted that for practical applications these smooth, fluctuating distributions over a large spatial scale are usually not very disturbing, as one is interested mainly in preserving the correct neighborhood relationships along the most important feature dimensions. As to the second type, the fluctuations follow the variance of the input data. If this variance exceeds some critical value, the associated nodes become unstable and a reconfiguration can occur. The calculation of this critical value is theoretically provided, for some very simple cases, by Ritter and Schulten (1986). In essence, and translating the mathematical ideas behind this possible source of instability into simple terms, the theory is telling us that if the variance among the images is very large or the number of images is very small, then the algorithm would be unstable, which is a logical result.

Centering our attention on the real data applications to electron microscopy to be presented later in this work, we have never encountered any convergence problems of this second class when analyzing the data sets used here, a fact that indicates that there is a good trade-off between variability and number of images in typical electron microscopic applications.

We note that the projection of the input data onto the output plane does not require any a priori knowledge of the number of classes present in the input data. It is, however, obvious that a decision on the total number and spatial distribution of output cells has to be done before hand and that, since the code vectors tend to follow the probability density distribution δ of the input data, there is a relationship between the number of output code vectors and both the number of input images and δ. Still, and as we note in the section presenting the examples, simple decisions on the number of code vectors have been sufficient to perform the studies presented in this work.

Self-organizing maps could be presented as a "kind" of k-means clustering algorithm, in the sense that this is considered its most similar "classical" method (Lippmann, 1987). There are, however, clear differences between the two approaches, such as the notion of neighborhood code vector updating, that is not present in k-means, and the fact that in self-organizing nets each input vector is presented only once (if the number of samples is greater than the number of iterations), while in k-means algorithms each image is presented once for each iteration.

Learning vector quantization (Kohonen, 1990), a variant of the described algorithm, has been used to implement a supervised pattern classification tool; by supervised we mean that a set of already classified vectors is necessary.

This latter approach is different from the one described before; instead of trying to approximate the code vectors to the input vectors (or their probability function), we wish to obtain vectors that effectively represent each class and produce optimal decisions. This second method has two distinct stages.

First is the training procedure, in which a set of classified vectors is needed. We present the classified input vectors $\mathbf{x}(t)$

to the network as in the previous algorithm, but the updating is done in a different way. First, no neighborhood is taken into account. Second, if the input vector $\mathbf{x}(t)$ is well classified, then the code vector $\mathbf{m}_j$ is updated to match it more closely, while if the classification fails, then $\mathbf{m}_j$ is updated in such a way that the distance d_j is increased.

The updating strategy is basically the same one as in the previous case of unsupervised classification in that code vector $\mathbf{m}_j$ is modified by a quantity proportional to the difference between the node and the last assigned input, i.e., to the term $(\mathbf{x}(t) - \mathbf{m}_j(t))$. However, there are two important differences in the exact way of proceeding. The first one is that no neighborhood is taken in account and that, therefore, the function $h(r, t)$ is not considered. The second one is that the quantity $(\mathbf{x}(t) - \mathbf{m}_j(t))$ is added or subtracted to the vector $\mathbf{m}_j$ depending on whether the classification was considered correct or incorrect. If correct, this term is added and the updated code vector $\mathbf{m}_j$ becomes more similar to the assigned input $\mathbf{x}(t)$. If incorrect, then this term is subtracted and $\mathbf{m}_j$ becomes less similar to $\mathbf{x}(t)$ than it was before.

Second is the classification itself, which is accomplished by calculating the distance between the unclassified vectors and the tuned code vectors and selecting that vector (that class) which is the closest in some defined way.

It can be proved that the described strategy gives an optimum classifier in the Bayesian sense (Kohonen, 1990). The training procedure is summarized in Table 2.

HOW THE SELF-ORGANIZING ALGORITHM WORKS

We have defined mathematically how the self-organizing map deals with general input vectors $\mathbf{x}(t)$. We now present a more conceptual explanation of how the algorithm behaves when confronted with a set of different images, acting as $\mathbf{x}(t)$ vectors. We are mainly interested in showing, through a simple example, how the output code vectors evolve and how images are assigned to these vectors through their evolution.

TABLE 2 Learning vector quantization training algorithm

Step no.	Training procedure
0	Convert the classified images I_i into vectors $\mathbf{x}(t)$.
1	Initialize the $\mathbf{m}_j$ vectors. Initialize the code vectors $\mathbf{m}_j$ associated with each output node to an initial value (for example, to some of the already classified images).
2	Present one input vector.
3	Compute distance to all nodes at time t. Compute the distance d_j between the input vector $\mathbf{x}(t)$ and each output node j using $d_j = \|\mathbf{x}(t) - \mathbf{m}_j(t)\|$.
4	Select output node with minimum distance. Select the output node as that node associated with the code vector $\mathbf{m}_j(t)$ that minimizes d_j.
5	Update code vectors in node j. $\mathbf{m}_j$ are updated in the following way: $\mathbf{m}_j(t+1) = \mathbf{m}_j(t) + \alpha(t)(\mathbf{x}(t) - \mathbf{m}_j(t))$ if $\mathbf{x}$ is correctly classified $\mathbf{m}_j(t+1) = \mathbf{m}_j(t) - \alpha(t)(\mathbf{x}(t) - \mathbf{m}_j(t))$ if $\mathbf{x}$ is incorrectly classified $\mathbf{m}_k(t+1) = \mathbf{m}_k(t)$ if k is different from j. where $\alpha(t)$ is a decreasing function that controls the magnitude of the changes with time ($0 < \alpha(t) < 1$).
6	Repeat (go to step 2) until the process converges.

The case study we are going to use in this section corresponds to the classification of a set of lateral views obtained from a biochemically pure preparation of the cytoplasmic chaperonin Tailless Complex Polypeptide-1 side view (see example 2 for a full description of the data set). The heterogeneity within the processed population of images comes from the fact that, intentionally, the original images have been centered but not rotationally aligned. The evolution of the code vectors at different iterations is shown in Fig. 1.

Fig. 1, *1* presents the code vectors before any tuning. These initial code vectors have been calculated using the following equation:

$$CV_{i,j} = \min(I_j) + (\max(I_j) - \min(I_j))*\text{rand\_number}_j \quad (1)$$

that is, the value of pixel i of code vector j is a random number lying in the interval $[\min(I_j), \max(I_j)]$, where I_j refers to an image randomly selected from our data set that is different for each code vector.

It is clear that the initial code vectors generated in this way are all different, and that it is almost impossible to recognize any characteristics of the images under analysis in them. Other forms of code vector initialization have been tested, but we have not found any difference in the final results of the algorithm.

At any rate, there is always one code vector that has minimum "distance" from a certain view of the particles under analysis. Let us assume that this code vector is the encircled one (Fig. 1, *1*). After some iterations, this code vector and its neighborhood are constrained to resemble this view more closely (Fig. 1, *2*).

The code vectors surrounding code vector *a8* are not subject to similar forces, because while the upper right corner (code vector *a4*) is going to be changed to match mainly vectors with this particular view, the lower left (code vector *a12*) is surrounded by other code vectors that are not well defined, so it is not going to be modified to resemble only a single view. What finally happens in this step is that *a4* will be better defined than *a12* and its surroundings.

The immediate consequence (Fig. 1, *3*) is that the code vector *a12* is going to receive less and less influence from the particular view that began the classification because its distance to this view is larger than the distance between this same specimen view and code vector *a4*. Furthermore, after this initial modification, *a4* and *a12* are going to attract different views, which in this case means rotated views, immediately forcing *a12* and *a4* to rotate in opposite directions. Of course, *a12* influences its neighborhood just as *a8* did.

Fig. 1, *4* shows how the best defined code vectors migrate toward the corners, or borders, of the output plane. At the same time that this migration occurs, the code vectors rotate following the mechanism described in Fig. 1, *3*.

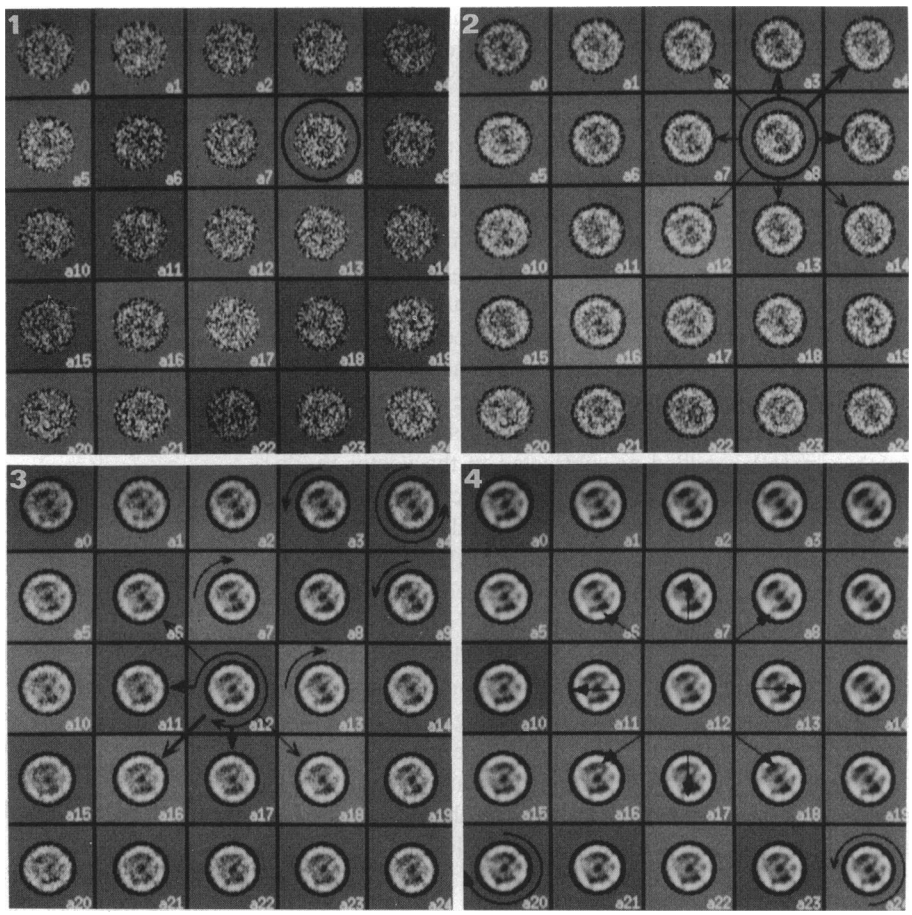

FIGURE 1 Code vector evolution during the classification procedure of a set of 404 images corresponding to side views of the chaperonin TCP-1 complex that have been centered but not rotationally aligned. In all the cases presented in this work, images were of dimensions 64 × 64 pixels and were analyzed within a circle of radius 22 pixels. The labels attached to each code vector are their unique image names used within our image processing system. (*1*) Code vectors before any tuning. The circled particle (labeled as *a8*) is going to minimize the distance to a particular view. (*2*) Code vectors after 50 iterations. The upper right area begins to match a particular view. (*3*) Code vectors after 100 iterations. The code vectors start to differentiate. (*4*) Code vectors after 150 iterations. The best matching code vectors migrate to the border of the output plane and complete their differentiation.

It is a central result in self-organizing theory that the output code vectors tend to follow the probability density distribution δ of the input data (Kohonen, 1990). It is therefore possible in those cases in which δ presents well defined peaks (for example, when the sample under analysis presents two clear clusters) that the meaningful code vectors lie in opposite corners of the map and the rest of the output map remains mainly empty.

FIELDS OF APPLICATION

We propose to use the self-organizing maps mainly for two purposes: after a translational and rotational alignment of macromolecules, and after a translational (but not rotational) alignment of macromolecules.

Translational and rotational alignment of macromolecules

The classification focuses on intrinsic differences among the images. In a homogeneous population only minor changes are to be anticipated, usually related to small deformations of the particles or inhomogeneous staining (example 2, Fig. 3). On the other hand, if we have a heterogeneous population of views or specimens, then the self-organizing map approach would separate the different classes (example 3). Furthermore, in populations with impurities, self-organizing maps help to distinguish between the specimens and the impurities.

Translational (but not rotational) alignment of macromolecules

If we start from a basically homogeneous data set that is not rotationally aligned, then the classification will group images according to their relative orientation (examples 1 and 2). It is therefore possible to use the code vectors as an intermediate step toward reference-free rotational alignment, and even to use one of the code vectors as a reference if a reference is needed.

The second algorithm performs a supervised classification, that is, a set of preclassified vectors is needed. Although we have not used this approach for any real application in our laboratory, we believe that it might be useful for some special cases. To provide the reader with an example of how this other algorithm works, we will present in example 4 a real data application of the method.

EXAMPLES

In this section we illustrate the use of the self-organizing map in addressing four different classification problems. The

following protocol has been followed from the algorithmic point of view. 1) A hexagonal topology has always been used for the output plane (output nodes are represented on a square grid just for display purpose). 2) The total number of nodes ranged from 25 to 100. 3) Two rounds of particle presentation were applied, the first one containing 1000 presentations and the second one 10,000 (as the total number of original images was about 400, presentations were iterated over the data set. That is, each image was presented 10,000/400 = 24 times, approximately). 4) Values of $\alpha(0) = 0.05$ and $\alpha(0) = 0.01$ were used in the two rounds of presentations, respectively. The explicit form of the function $\alpha(t)$ was:

$$\alpha = \alpha(0)\left(1 - \frac{n.iterations - t}{n.iterations}\right)$$

where *n.iterations* is the number of iterations. 5) The neighborhood function $h(r, t)$ had a Gaussian profile, starting as the whole output plane in the first round and about half of it in the second round. The explicit form of $h(r, t)$ is:

$$h(r, t) = \exp\left(\frac{\|r_j - r_c\|}{2\sigma(t)^2}\right)$$

where

$$\sigma(t) = 1 + (R_{max} - 1)\left(\frac{n.iterations - t}{n.iterations}\right)$$

n.iterations is the number of iterations; R_{max} is the initial neighborhood radius; and $\|r_j - r_c\|$ is the distance between the node j and the node i. 6) In all cases, images were of dimensions 64 × 64 pixels, and they were analyzed within a circular mask of radius 22 pixels.

We should note that typical computing times were from 5 to 10 min on a Silicon Graphics Indigo 4000 workstation and that the general behavior of the method changed very little when different choices of α and neighborhood were used.

The first two examples involve samples from which homogeneous image sets were obtained and preprocessed. The preprocessing amounted to a translational, but not rotational, alignment of the particles. The goal is to demonstrate that the method is able to correctly order the different images as a function of their rotational angle.

Example 3 deals with the case of a biochemically heterogeneous sample giving rise to a set of images showing different views. In this case, the images were translationally and rotationally centered and the goal was to differentiate among the biochemically distinct classes.

Finally, example 4 uses the learning vector quantization algorithm to automatically classify the images obtained from the biochemically heterogeneous sample already used in the previous example. An initial classification had to be performed to provide the algorithm with the needed preclassified training set.

Example 1

The specimen used is *B. subtilis* GroEL (Carrascosa et al., 1990). GroEL is a member of the so-called chaperonins, a family of proteins that plays an active role in the translation, folding, and assembly of polypeptides (for a review, see Gatenby and Ellis, 1990). A total of 307 translationally but not rotationally aligned images corresponding to the top view of the aggregate have been used in this example for classification. Fig. 2 summarizes the results. Figure 2, *1* shows input data.

Fig. 2, *2* and *3* show the final ("tuned") code vectors, which effectively represent all the possible different views of a rotationally unaligned set of particles (in this type of analysis the meaningful code vectors are located at the periphery of the output map). Fig. 2, *4* shows some of the actual images assigned to each code vector.

As a comparison with another already well established approach in this field, we also present the first four eigenimages from the PCA performed over the input data set (Fig. 2, *5*). While PCA is able to detect a trend of variability which is indicative of the presence of a sevenfold symmetry, no explicit classification is performed at this step. The self-organizing map, by contrast, directly group the images according to their relative angles and renders code vectors that can be very easily interpreted.

Example 2

The aim of this second example is to show the versatility of the method when applied to another specimen with a symmetry less evident than in the previous case.

The specimen used is the TCP-1 complex (side view) (Gao et al., 1993; Marco et al., manuscript submitted). The TCP-1 complex is a cytoplasmic chaperonin that contains the t-complex polypeptide TCP-1 among several other related polypeptides.

The general image processing approach, as well as the layout of the corresponding figure (Fig. 3) presenting the results, are basically the same as in the previous example. A total of 404 images were translationally but not rotationally aligned before being used as input to the self-organizing map.

As before, while it is very difficult to discern any pattern in the input images (a representative gallery is shown in Fig. 3, *1*), the tuned code vectors at the borders of the map clearly show different rotated versions of the TCP-1 complex side view (Fig. 3, *2* and *3*). Besides the ensemble information provided by the code vectors themselves, we also have the list of individual images assigned to each vector (see gallery in Fig. 3, *4*). We have therefore assigned rotational angles to the input images directly through this classification scheme.

The first four eigenimages from PCA are also presented in Fig. 3, *5*, resulting in basically the same situation as the one shown in the previous example.

A question raised is how the method would perform if applied in the following three-step procedure. First, a cycle of rotational angle assignment by classification is carried out (Fig. 3, *1–4*). Second, the angle so found is applied to each input image. Third, a second cycle of classification on the already aligned images is accomplished.

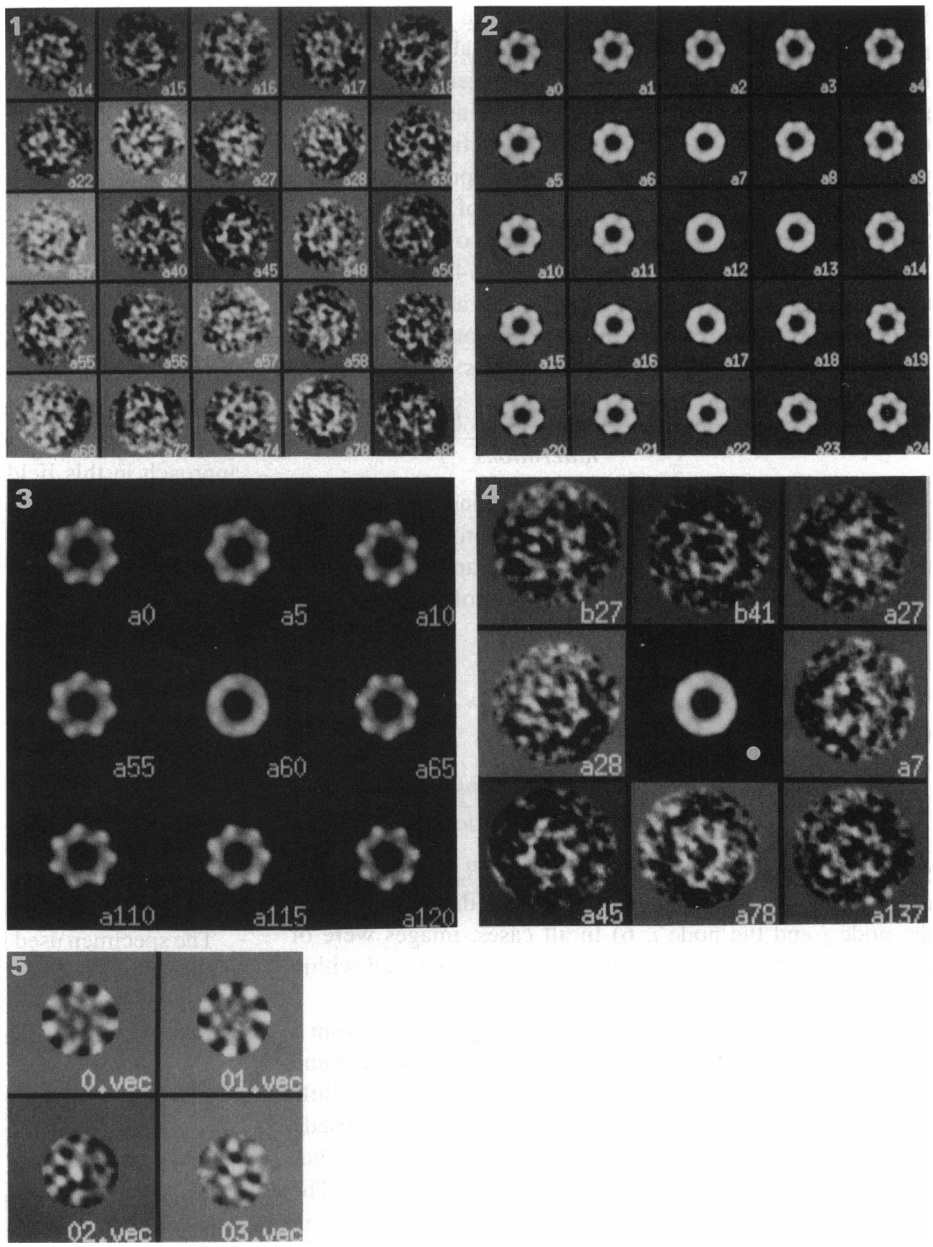

FIGURE 2 Unsupervised classification of *B. subtilis* GroEL chaperonin (*top view*). A total of 307 translationally but not rotationally aligned images have been used. The external diameter of the particles in this characteristic view is ~14.5 nm. (*1*) Gallery of particles translationally but not rotationally aligned. (*2*) Tuned code vectors showing that the main difference between the particles is a rotational misregistration. (*3*) Tuned code vectors, detail. (*4*) An image randomly chosen among those assigned to the code vectors shown in Fig. 2, *4*. Note that there were no images assigned to the central code vector *a60* and that, therefore, the image that is shown at the center marked with a dot is not an original image but the code vector *a60* itself. (*5*) First four eigenimages obtained by PCA. The first two suggest a sevenfold symmetry.

The results of such a double classification scheme is presented in Fig. 3, *6*, clearly proving that the algorithm now classifies images according to subtle variations, such as differences in contrast at the left side of the view.

Example 3

In this example we analyze a data set formed by an already translationally and rotationally aligned set of 420 images, corresponding to top views obtained from a biochemically heterogeneous sample containing the TCP-1 complex (Gatenby and Ellis, 1990; Gao et al., 1993) and the TCP-1/actin binary complex. In this case, the aim is to detect this biochemical variability within the aligned set of images.

That this data set does not lend itself to an obvious classification is evident from the gallery of images presented in Fig. 4, *1*. However, the tuned code vectors shown in Fig. 4, *2* clearly present a pattern of variability at the center of the particle, providing a direct interpretation of the global variability of the data set. The final assignment of the input images to the code vectors is presented in Fig. 4, *3*, which is actually a classification of the input data set.

In this particular case a PCA approach, while useful, was more difficult to interpret than the self-organizing map (Fig. 4, *4*). In essence, the first eigenimage detected a variability at the center of the particle, but even in this image, and certainly in the other three eigenimages shown here, the pattern of variability is rather complex.

Example 4

The purpose of this example is to test the learning vector quantization algorithm for supervised classification. We

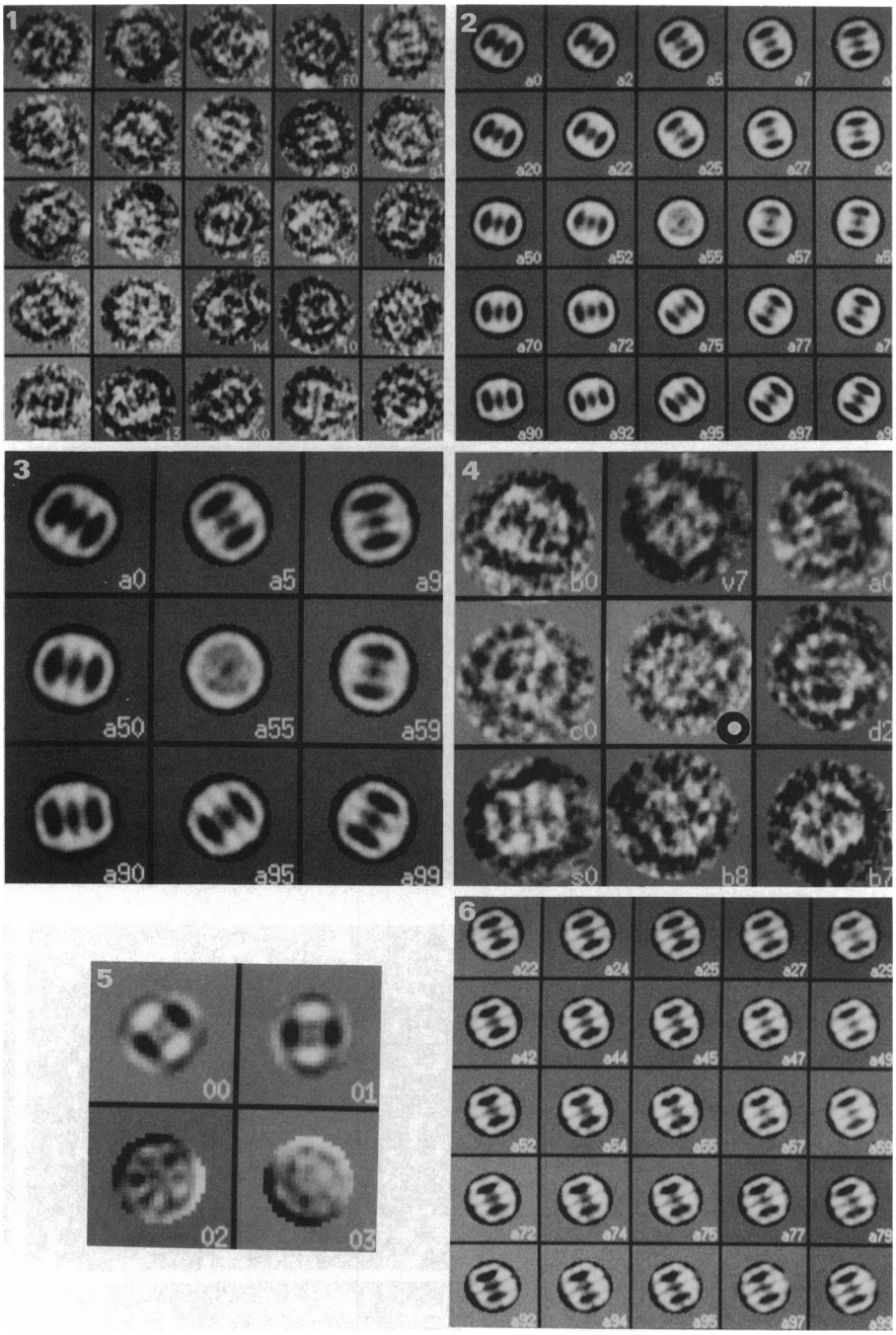

FIGURE 3 Unsupervised classification of TCP-1-complex (*side view*). A total of 407 translationally but not rotationally aligned images have been used. The maximum dimensions of the particles in this view are ~15.3 × 16.3 nm. (*1*) Gallery of particles. (*2*) Tuned code vectors. (*3*) Tuned code vectors (detail). (*4*) An image randomly chosen among those assigned to the code vectors shown in *4*. Note that the image marked with a dot, originally assigned to the central code vector *a55*, is not a lateral view but a top one. (*5*) First four eigenimages obtained by principal component analysis. (*6*) Code vectors of the same initial image set after rotational alignment.

have used the same images as in example 3. By visual inspection, approximately 200 particles were classified, 100 as TCP-1 alone and the other 100 as TCP-1/actin (the remainder of the 420 particles are not easily classified). This manually classified data set was then divided into two groups of equal size, each one containing 50 particles of the two classes. The first group was then used to tune the code vectors (Fig. 5, *3*), while the other one was used as the test data set. The accuracy of the subsequent classification of the test data set was 92–96%, depending on the choice upon the tuning subsets.

Figs. 5, *1* and *2*, are galleries of TCP-1 with and without actin. Fig. 5, *3* shows the tuned code vectors. From this output vector map it is evident that virtually all the influence has been received by only one code vector of each class (*a3*/*a13*).

DISCUSSION

In this work we present the application of a particular type of neural network, known as the self-organizing map, to a number of interesting problems in the field of macromolecular structure determination by electron microscopy and image processing. More precisely, we address those situations in which crystals are not available and the structural determination has to be based on the analysis of a large

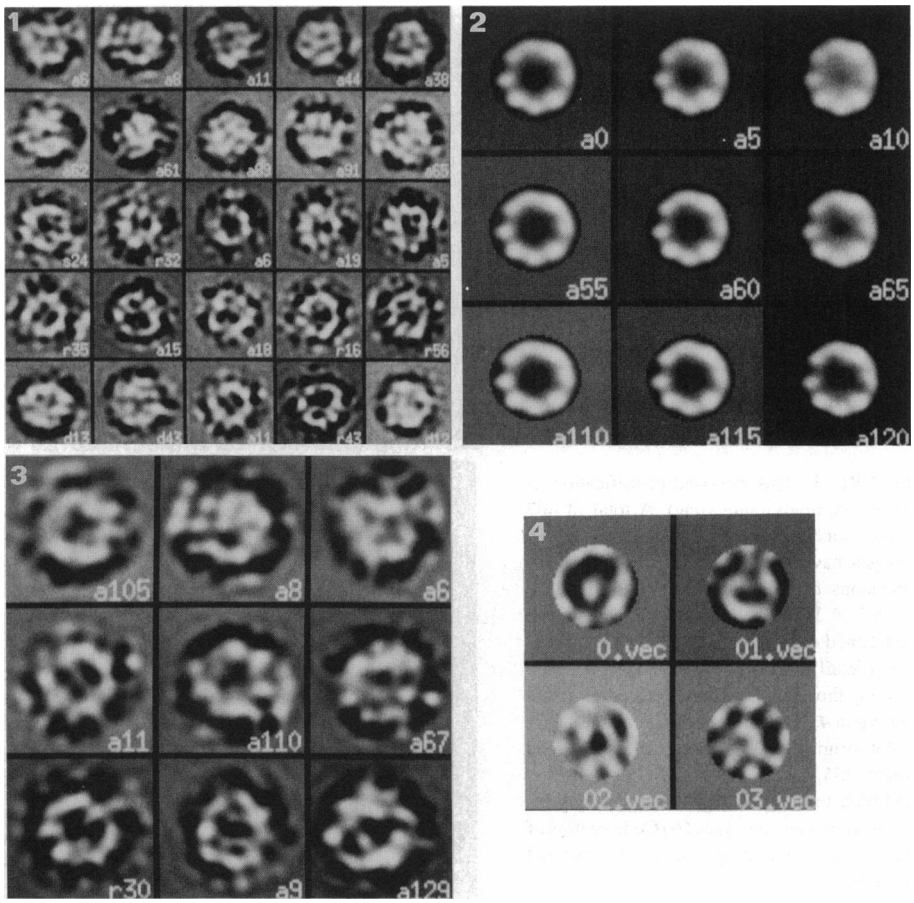

FIGURE 4 Unsupervised classification of TCP-1 and TCP-1/actin complexes (*top view*) performed from a set of 420 images. The external diameter of the particles in this characteristic view is ~16.5 nm. (*1*) Gallery of particles translationally and rotationally aligned. (*2*) Tuned code vectors (detail). (*3*) An image randomly chosen among those assigned to the code vectors shown in *2*. (*4*) First four eigenimages obtained by principal component analysis. The first eigenimage, labeled as 0.vec, showed a clear variability located at the center of the particle, but the general pattern was difficult to interpret.

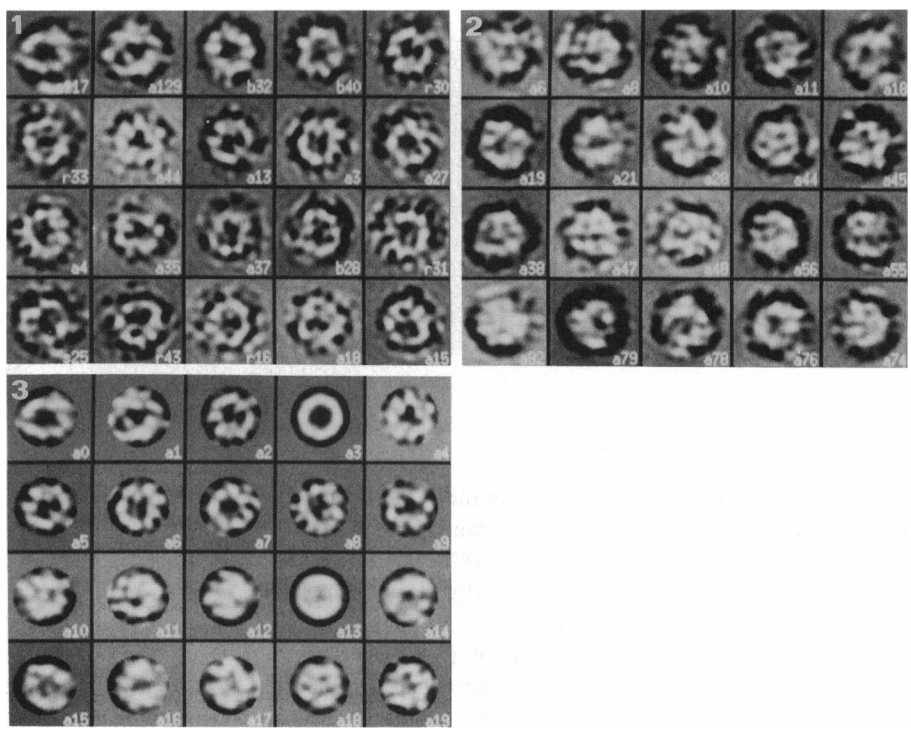

FIGURE 5 Supervised classification of TCP-1 complex and TCP-1/actin complex (*top view*) performed from a total set of 420 translationally and rotationally aligned images of 64 × 64 pixels. (*1*) Gallery of TCP-1/actin complex particles. (*2*) Gallery of TCP-1 complex particles. (*3*) Tuned code vectors. Notice that code vectors *a3* and *a13* have received most of the influence during the tuning process.

collection of images of isolated particles. Recent examples, such as the resolution of the three-dimensional structure of the *Escherichia coli* ribosome (Frank et al., 1991) or of adenovirus (Stewart et al., 1991), are good examples of the biological potentials of this type of study.

The need for powerful classification tools is clearly recognized as a key problem, and much work has been devoted to this problem in recent years (for a review, see Frank, 1990). We present in this work a totally new approach to this problem that is based on a particular type of neural network

and that is able to perform, in a rather simple way, many of the main classification tasks.

We have used as examples a number of studies involving chaperonins, in particular GroEL and TCP-1, to provide the reader with a sense of the wide range of problems that can be addressed with the approach we propose. On structural biology grounds, we would like to further comment on the results obtained when analyzing top views of the TCP-1 and TCP-1/actin binary complexes (example 3). The methodology we propose in this work is able to provide a very simple, clear, and direct piece of information on the location of the unfolded actin within the TCP-1 complex. It is evident from an inspection of the code vectors shown in Fig. 4 that the location of the actin is in the central channel of the aggregate, suggesting a number of interesting hypotheses (a more in-depth analysis is presented by Marco et al. in an accompanying paper).

We have shown that if we start from a basically homogeneous population of views, then the neural network-based approach is able to classify directly a set of rotationally misaligned images into groups related to their relative orientation. Alternatively, if the population of views is potentially heterogeneous but the images are known to be correctly aligned, a classification centered on more intrinsic differences among the particles may be performed. The application to negatively stained samples of GroEL and TCP-1 indicate that the method works remarkably well in a variety of cases.

Placing this new work in the context of the methodology commonly used in image processing, it is clear that it is particularly valuable when addressing two vital steps, one being the elucidation of the underlying ensemble structure from a large collection of homogeneous misaligned images, and the other the detection of different views among a collection of aligned input images. As far as the analysis of a set of misaligned images is concerned, this methodology offers a way to perform a classification based on similarities in the relative orientations of the views without any prior angular correlation-based alignment. On the detection of structural differences, the approach does not need an a priori knowledge of the number of clusters present in the starting population.

Comparing this approach to others previously proposed in this field for structure-based classification, we should mainly comment on MSA. MSA is a very good method for data reduction, capable of extracting the main patterns of variability within a given data set and, in many cases, enabling us to understand the patterns of heterogeneity present in our image data set, especially if coupled with some further classification step (for a review, see Frank, 1990). MSA has also been recently used to perform an orientation classification (Dube et al., 1993). The action of the neural network method we propose here is quite different from MSA in that it directly performs a classification (the assignment to specific code vectors) while at the same time it offers a direct visualization of the pattern of variability through an inspection of the code vectors themselves. The algorithm is also very fast, since typical computer times for the analysis of 400 particles within a circle of radius 22 were on the order of 5–10 min on a modern workstation. The classification process of our neural network approach is also very easy to understand through inspection of the code vectors.

Taking into account the properties of the new approach discussed above, we propose to use the self-organizing map as a new routine tool for image classification that may, if needed, be complemented by other tools, such as MSA. Considerations on the speed of the neural network approach, the simplicity and directness of the output results, and its robustness against noise, make it a valuable tool.

Author's note—The programs used to perform this work, user's guide, and a complete example are available by anonymous ftp and gopher at gopher.cnb.uam.es.

The authors thank S. Marco for careful testing of the programs and J. J. Merelo for introducing us to the world of neural networks. We also wish to thank Dr. T. Kohonen for making his programs available by anonymous ftp. GroEL images were kindly provided by Dr. J. L. Carracosa's group, and TCP-1 images by Dr. N. Cowan's and Dr. J. L. Carrascosa's groups. Comments from Dr. J. Frank and Dr. P. Penczek on earlier versions of this work are also acknowledged and greatly appreciated.

This work was supported in part by Grant PB91-0910 from Plan General de Promoción del Conocimiento (Dirección General de Investigación Científica y Técnica, DGICYT, Spain). R. Marabini holds a CSIC predoctoral fellowship.

REFERENCES

Boekema E. J., J. A. Berden, and M. van Heel. 1986. Structure of mitochondrial F_1-ATPase studied by electron microscopy and image processing. *Biochim. Biophys. Acta.* 851:353–360.

Carazo, J. M., F. F. Rivera, E. L. Zapata, M. Radermacher, and J. Frank. 1990. Fuzzy sets-based classification of electron microscopy images of biological macromolecules with an application to ribosomal particles. *J. Microsc.* 157:187–203.

Carazo, J. M., T. Wagenknecht, and J. Frank. 1989. Variations of the 3D structure of the *E. coli* ribosome in the range of overlap views. an application of the methods of multicone and local single-cone three-dimensional reconstruction. *Biophys. J.* 55:465–477.

Carrascosa, J. L., G. Abella, S. Marco, and J. M. Carazo. 1990. Three-dimensional reconstruction of the 7-folded form of the *B. subtilis* GroEL chaperonin. *J. Struct. Biol.* 104:2–8.

Cottrell, M., and J. C. Fort. 1986. A stochastic model of retinotopy: a self organizing process. *Biol. Cybern.* 53:405–411.

Dube P., P. Tavares, R. Lurz, and M. van Heel. 1993. The portal protein of bacteriophage SPP1: a DNA pump with 13-fold symmetry. *EMBO J.* 12:1303–1309.

Frank, J. 1990. Classification of macromolecular assemblies studied as single particles. *Q. Rev. Biophys.* 23:281–329.

Frank, J., J. P. Betraudiere, J. M. Carazo, A. Verschoor, and T. Wagenknecht, 1988a. Classification of images of biomolecular assemblies: a study of ribosomes and ribosomal subunits of *Escherichia coli*. *J. Microsc.* 150:99–115.

Frank J., P. Penczek, R. Grassucci, and S. Srivastava. 1991. Three-dimensional reconstruction of the 70S *Escherichia coli* ribosome in ice: the distribution of ribosomal RNA. *J. Cell Biol.* 115:597–605.

Frank J., P. Penczek, and W. Liu. 1992. Alignment, classification, and 3-D reconstruction of single particles embedded in ice. *In* Proceedings of the 10th Pfefferkorn Conference on Signal Processing: Scanning Microscopy Supplement. P. W. Hawkes and W. O. Saxton, editors. Scanning Microscopy International.

Frank J., M. Radermacher, T. Wagenknecht, and A. Verschoor. 1988b. Studying ribosome structure by electron microscopy and computer-image processing. *Methods Enzymol.* 164:3–35.

Gao Y., I. E. Vainberg, R. L. Chow, and N. J. Cowan. 1993. Two cofactors and cytoplasmic chaperonin are required for the folding of alpha-tubulin and beta-tubulin. *Mol. Cell. Biol.* 13:2478–2485.

Gatenby, A., and R. J. Ellis. 1990. Chaperone function: the assembly of

ribulose biphosphate carboxylase-oxygenase. *Annu. Rev. Cell Biol.* 6:125–149.

Kohonen, T. 1990. The self organizing map. *Proc. IEEE.* 78 (9):1464–1480.

Lippmann, R. P. 1987. An introduction to computing with neuronal nets. *IEEE ASSP Magazine.* 4–22.

Radermacher, M., and J. Frank. 1985. Use of nonlinear mapping in multivariate image analysis of molecule projections. *Ultramicroscopy.* 17:117–126. (Erratum, *Ultramicroscopy* 19:75 (1986)).

Ritter, H., and K. Schulten. 1986. On the stationary state of Kohonen's self-organizing sensory mapping. *Biol. Cybern.* 54:99–106

Ritter H., and K. Schulten. 1988. Convergence properties of Kohonen's topology conserving maps: fluctuations, stability and dimension selection. *Biol. Cybern.* 60:59–71.

Schatz, M., and M. van Heel, 1990. Invariant classification of molecular views in electron micrographs. *Ultramicroscopy.* 32:255–264.

Stewart, P. L., R. M. Burnett, M. Cyrklaff, and S. D. Fuller. 1991. Image reconstruction reveals the complex molecular organization of adenovirus. *Cell.* 67:145–154.

van Heel, M. 1986. Finding the characteristic views of macromolecules in extremely noisy electron micrographs. *In* Pattern Recognition in Practice. Vol. II. E. S. Gelsema and L. N. Kanal, editors. Elsevier North-Holland, New York. 291–299.

Wagenknecht, T., J. M. Carazo, M. Radermacher, and J. Frank. 1989. Three-dimensional reconstruction of the ribosome from *E. coli. Biophys. J.* 55: 455–464.