Pictorial Information Retrieval Using the Random Neural Network

Andreas Stafylopatis, *Member, IEEE*, and Aristidis Likas, *Student Member, IEEE*

Abstract—**A technique is developed based on the use of a neural network model for performing information retrieval in a pictorial information system. The neural network provides autoassociative memory operation and allows the retrieval of stored symbolic images using erroneous or incomplete information as input. The network used is based on an adaptation of the random neural network model featuring positive and negative nodes and symmetrical behavior of positive and negative signals. The network architecture considered here has hierarchical structure and allows two level operation during learning and recall. An experimental software prototype, including an efficient graphical interface, has been implemented and tested. The performance of the system has been investigated through experiments under several schemes concerning storage and reconstruction of patterns. These schemes are either based on properties of the random network or constitute adaptations of known neural network techniques.**

Index Terms—**Associative memory, Hebbian learning, information retrieval, neural computation, pictorial information systems, random neural network.**

I. Introduction

PICTORIAL information systems have raised an increasing interest during the last decade. The role of images as a basic component of information interchange has been well established, and a great number of applications based on image processing and management have been realized, including medical diagnosis, geographic information systems, robot navigation and task planning, satellite image processing; there have also been several industrial applications [21], [22].

In general, a pictorial information system constitutes a combination of three basic components. The first component deals with image processing and the extraction of information from physical images. Typical operations provided by image processing packages include image segmentation, edge detection, thresholding, contour drawing, similarity retrieval, texture measurement, clustering, interpolation, set operations, etc. [22].

The second component is a pictorial database that is responsible for the storage and management of both the original images and the extracted information. Due to its multidimensional nature, an image incorporates a large amount of both explicit and implicit information [21]. Explicit information concerns the object entities contained in the image as well as pixel-level information such as color and brightness. Implicit information is related to the relative positions of objects in the image and to distances between objects. This implicit information is very important and results in vast amounts of data if storage in a relational database is attempted. For such reasons, traditional approaches that have been successfully applied to conventional databases cannot be directly applied to pictorial information systems. In addition, the problem of inferencing is more difficult when dealing with spatial information [19].

The third component of a pictorial information system concerns the user interface and the various mechanisms that enable the user to effectively exploit iconic information. Several high-level query languages, adapted to the needs of pictorial systems, have been developed in this direction [6], [22]. But, as pictorial databases tend to grow rapidly in size, the problem of how to get at the data becomes central. The user no longer knows the exact contents of the database nor does he have a clear sense of what should be retrieved [11]. Thus, concepts such as visualization and browsing have gained interest recently and interactive visual languages have been developed [10], [11].

As stated in [21], in order to progress in image databases from the software perspective, it is important to examine innovations in both processing and representations of images and also to devise techniques that significantly accelerate the speed of computation. Neural network technology constitutes a candidate that may offer effective solutions to many problems related to information systems [5], [13], [18]. The incorporation of neural network models as modules of pictorial information systems would provide several significant advantages. First of all, neural network models are inherently parallel and many of them can be implemented directly in hardware or can be easily simulated on massively parallel machines. Therefore, the speed of computation can be increased by several orders of magnitude. Moreover, the connection weights of a neural network constitute its long term memory. Thus, a high degree of fault-tolerance is achieved, since the damage of some connection or neurons does not lead to significant performance degradation. Finally, the most distinguished features of neural networks are related to their learning and self-organizing capabilities, as well as to their ability to automatically generalize and make inferences.

Neural networks have been successfully applied to problems related to the first component of pictorial information systems, as previously mentioned. Neural network models that perform adaptive pattern recognition, image segmentation, object classification, and other related operations have been developed

Manuscript received August 1, 1991; revised March 6, 1992. Recommended by E. Gelenbe.

The authors are with the Department of Electrical and Computer Engineering, Computer Science Division, Athens, Greece.

IEEE Log Number 9200607.

and tested with satisfactory results [24], [29]. In this paper we study the use of the *random neural network model* [16], [17] as a module of the second component. More specifically, we use it for the storage of symbolic pictures and their retrieval based on incomplete information. As will be described in the next section, symbolic images constitute an abstraction of physical images and play a very important role in pictorial information systems.

In Section II, we present the general framework of our approach, while Section III focuses on the random neural network model. In Section IV, a prototype system is described, which has been constructed for testing purposes, while some performance results are presented in Section V. Finally, the main issues addressed and conclusions drawn during this work are summarized in Section VI.

II. General Description of the Proposed Approach

A pictorial information system generally exhibits a complex hierarchical structure [6]. Several levels of abstraction may be defined, starting from the original physical images and moving towards more abstract layers containing symbolic (logical) pictures. Our approach deals with the use of a neural network model with associative memory properties for the storage and flexible retrieval of symbolic pictures.

The role of symbolic pictures in pictorial information systems is of great significance, because they allow fast and intelligent treatment of useful information. They are especially suitable for capturing the spatial knowledge embedded in images, i.e., knowledge related to the relative positions of image objects. The manipulation of physical images each time we wish to extract and exploit spatial knowledge is inefficient and time consuming. On the other hand, the maintenance of spatial information in the form of relations of a conventional database requires excessive storage space and access time.

First, in order to produce a symbolic picture from a physical one, image processing techniques are applied that result in segmentation of the physical image and recognition of the contained objects. Then objects are classified and a symbol or name is assigned to the objects of each category. In a symbolic picture, symbols are used in the places of objects taking special care to maintain their relative positions and distances. In [9], an abstraction methodology that is flexible enough to allow the representation of even complex spatial relationships is introduced. This methodology is based on orthogonal relations [7] and uses the notion of minimum enclosing rectangles to define the primitive ortho-relational subobjects into which an object is segmented. Details on this methodology can be found in [8] and [9]. An important issue is that symbolic images can be used as the basis for constructing indexes to the pictorial database. As a matter of fact, they constitute a natural coding mechanism relying on information drawn directly from the contents of the physical image. The actual symbolic image representation or some pattern derived from the symbolic image can be efficiently used as index to the database [8], [9].

Another significant point is that, given a symbolic picture, visualization of the physical image can be obtained in the form of a visual sketch composed of icons, i.e., graphical

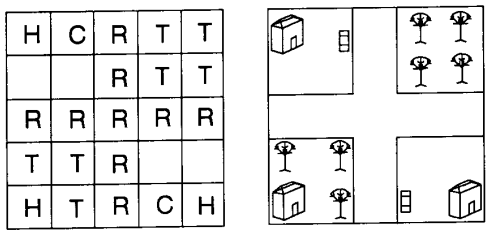


Fig. 1. (a) A symbolic image and (b) the corresponding visual sketch.

symbols representing objects. This seems easy in the case where the objects of the symbolic picture are "point objects," i.e., objects not segmented into subobjects, since it is sufficient to substitute the icon of the object for the object symbol. In a different case, visualization becomes a more difficult task and "connection rules" have to be applied to form an integrated object by connecting ortho-relational subobjects [9].

In what follows, it is convenient to consider the symbolic image as a two-dimensional array, with each array element corresponding to the symbol of an object (the null object is also assigned a special symbol). In Fig. 1, a symbolic picture is demonstrated along with the corresponding visual sketch that results from the visualization procedure. Symbols C, H, R, and T stand for car, house, road, and tree respectively.

Our objective is to examine the suitability of an artificial neural network with error correction capabilities for the retrieval of symbolic images based on incomplete information. Retrieval operations based on information that is either imprecise or vague constitute a very active research field, especially in very large and multimedia databases. Imprecision may characterize the data contained in the database, the retrieval request, or the notion of the user regarding the contents of the database [28]. The neural network approach can be used to deal with imprecision concerning the second and third of the above cases.

If the level of imprecision or noise in the retrieval request is not excessive, the neural network will eliminate imprecision or vagueness and provide the stored symbolic picture that is closest to the specifications imposed by the user. Moreover, it can be combined with visualization that is easily achieved from the symbolic picture as stated previously. Based on the result, the user can specify a new retrieval request and so on. Thus, using the neural network, we can achieve interactive browsing through a pictorial database using only symbolic pictures and without needing to perform the time consuming retrieval of physical images.

The above operation can also be useful in eliminating insignificant information in a changing environment, as, for example, in robot navigation or path planning. In such a case, navigation and planning decisions are made on the basis of an abstract picture containing the relative positions of relevant environmental objects.

In order to store symbolic images in an associative neural network, an internal representation of both objects and symbolic images must be constructed. In our approach, objects are encoded in terms of bipolar vectors (with elements in $\{-1, 1\}$).

By substituting object codes for the corresponding objects, a symbolic image can be transformed into a bipolar pattern suitable for manipulation by the network. The structure of the neural network architecture reflects the two-dimensional array representation of symbolic images, with a group of neurons (or nodes) corresponding to each array position. The number of neurons in each group is fixed and equal to the size of object codes.

When using an autoassociative neural network model for the storage and retrieval of symbolic images, some interesting singularities arise. First, an erroneous symbolic image contains errors at the object level, i.e., an existing object is missing, a nonexisting object is present, or some object appears in place of another. Due to this fact, the erroneous patterns that are presented as inputs to the neural network contain noise at the group level, i.e., wrong object codes. As will be explained in a later section, this phenomenon of noise accumulation in local regions of the pattern may affect the reconstruction capability of the network. A second peculiarity deals with the fact that patterns resulting as outputs of the network should contain legal object codes in all groups, i.e., codes that correspond to existing objects. Otherwise, the resulting output cannot be translated into a symbolic image. Because of the above properties of patterns, the autoassociative network must operate in a hierarchical fashion providing reconstruction both at the symbolic image level and at the object level. This means that, once the output pattern has been obtained through global correction, a second correction phase must be performed locally to ensure legal subpatterns.

Several neural network models exhibit autoassociative memory capabilities [23]. We have implemented the above described approach using the *random neural network model*, which will be presented in the next section.

III. The Neural Network Model

A. The Random Neural Network

The *random neural network model* has been introduced recently [14]–[16] and is characterized by the existence of positive and negative signals, which represent excitation and inhibition, respectively. The model is based on probabilistic assumptions and accepts a product form stationary probability distribution. In the implementation presented here, we have used an extension of the original random network model that exhibits associative memory capabilities [17], [25]. This extended model includes two types of nodes—positive and negative—and preserves the main properties of the original model, which considers only positive nodes. In the following, we will use the term random network to refer to the extended version, of which we will provide a brief description. Then, we will focus on the associative memory operation of the model.

Each node in the random network accumulates signals that either arrive from the outside of the network or from other nodes. External positive and negative signal arrivals to each node i are considered Poisson with rates $\Lambda(i)$ and $\lambda(i)$, respectively. A node can fire if its total signal count at a given instant of time is strictly positive. When firing, a node sends signals to other nodes or to the outside of the network. Firing occurs at random, the intervals between successive firing instants at node i following an exponential distribution with mean $1/r(i)$.

Positive and negative nodes have completely symmetrical behavior, in that only positive (negative) signals can accumulate at positive (negative) nodes. At each node, the role of signals of the opposite sign is purely suppressive, i.e., a negative (positive) signal arriving to a positive (negative) node cancels a positive (negative) signal if the node is not empty or has no effect (is lost) if the node is empty.

We distinguish positive and negative connections between nodes, which imply, respectively, that a signal leaving a node will move to another node as a signal of the same or the opposite category. More specifically, a positive signal that leaves positive node i arrives to node j as a positive signal with probability $p^+(i,j)$ and as a negative signal with probability $p^-(i,j)$. Similarly, a negative signal leaving negative node i arrives to node j as a negative signal with probability $p^+(i,j)$ and as a positive signal with probability $p^-(i,j)$. Also, a signal departs from the network upon leaving node i with probability $d(i)$. For a network with n nodes we shall have $\sum_{j=1}^{n}[p^+(i,j)+p^-(i,j)]+d(i)=1$, for $i=1,\cdots,n$.

An analogy between usual neural network representation and the model above described can be constructed [14], [16]. Each neuron is represented by a node of the random network. Considering the non-output neuron i, the parameters of the corresponding node i are chosen as follows:

$$d(i)=0 \tag{1}$$

$$r(i)p^+(i,j)=w_{ij}, \quad \text{if } w_{ij}>0 \tag{2}$$

$$r(i)p^-(i,j)=|w_{ij}|, \quad \text{if } w_{ij}<0 \tag{3}$$

where w_{ij} is the connection weight from neuron i to neuron j. Summing over all j, the firing rate $r(i)$ of the non-output node i is set:

$$r(i)=\sum_j |w_{ij}|. \tag{4}$$

Finally, for each output node i, $d(i)=1$ and some appropriate value is assigned to $r(i)$.

The above described Markovian network has product form solution; that is, the network's stationary probability distribution can be written as the product of the marginal probabilities of the state of each node. Thus, the network is seemingly composed of independent nodes, though this is obviously not the case, since in fact nodes are coupled via the circulating signals. This property was shown in [14] and [16] for the original version of the random network.

Let us consider a random network with n nodes. Also let $\hat{K}(t)$ be the state vector at time t representing the signal count at each node of the network, and $\hat{k}=(k_1,\cdots,k_n)$ be a particular value of the vector. The stationary probability distribution of the network's state is given by $p(\hat{k})=\lim_{t\to\infty}\text{Prob}[\hat{K}(t)=\hat{k}]$ whenever this limit exists. Since

both types of nodes accumulate signals of the corresponding sign only, an unsigned value is sufficient for describing the state of a node once the node sign is given.

If we denote by $\mathcal{P}$ and $\mathcal{N}$ the sets of positive and negative nodes, respectively, the flow of signals in the network can be described by the following equations:

$$\gamma^+(i) = \Lambda(i) + \gamma^+_{\text{int}}(i) = \Lambda(i) + \sum_{j\in\mathcal{P}} q_j r(j) p^+(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^-(j,i) \tag{5}$$

$$\gamma^-(i) = \lambda(i) + \gamma^-_{\text{int}}(i) = \lambda(i) + \sum_{j\in\mathcal{P}} q_j r(j) p^-(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^+(j,i) \tag{6}$$

where $\gamma^+_{\text{int}}(i)$, $\gamma^-_{\text{int}}(i)$ represent the internal arrival rates of positive and negative signals, respectively, and

$$q_i = \frac{\gamma^+(i)}{r(i)+\gamma^-(i)}, \quad i \in \mathcal{P} \tag{7}$$

$$q_i = \frac{\gamma^-(i)}{r(i)+\gamma^+(i)}, \quad i \in \mathcal{N} \tag{8}$$

It can be shown that, if a unique non-negative solution $\{\gamma^+(i), \gamma^-(i)\}$ to the above equations exists such that $q_i < 1$, then the steady-state network probability distribution has the form

$$p\left(\hat{k}\right) = \prod_{i=1}^{n} [1-q_i] q_i^{k_i}. \tag{9}$$

The proof is analogous to the corresponding proof in [14] and [16]. In the Appendix, we give the global balance equations that are satisfied by $p\left(\hat{k}\right)$.

The quantity q_i represents the steady-state probability that node i is firing. Clearly, the quantity $q_i/(1-q_i)$ is the average number of signals at node i in the steady-state. Results concerning the existence and uniqueness of the solution to the set of equations (5)–(8) can be found in [15].

Applications of the random network model (original or extended version) have been reported in the fields of image processing [3], [27], combinatorial optimization [16], and associative memory [17].

B. Autoassociative Memory Operation

Suppose that we want to store m bipolar patterns $\left\{\hat{x}^1, \cdots, \hat{x}^m\right\}$ of the form $\hat{x}^k = \left(x_1^k, \cdots, x_n^k\right)$ in a random network with n nodes. In order to determine the characteristics of the network, we first compute the connection weights using some learning rule and subsequently the parameters $p^+(i,j)$, $p^-(i,j)$, $r(i)$, following the correspondence described earlier. The description of the network is completed only after characterizing its nodes as positive or negative and specifying external arrival rates. This characterization, however, shall not be fixed for all patterns, but will depend on the input pattern that should be reconstructed during operation. If the ith element of the input pattern is equal to 1, then the ith node of the network is set positive and the arrival rates of positive and negative signals to this node are $\Lambda(i) \neq 0$ and $\lambda(i) = 0$, respectively. If the ith element of the input pattern is equal to -1, then the ith node is set negative with $\Lambda(i) = 0$ and $\lambda(i) \neq 0$.

Thus there is a distinction between the fixed part of a network (routing probabilities and firing rates), which is set *a priori* according to some construction (learning) scheme and a variable part (sign of nodes and external signal arrivals) dictated by the pattern applied as input to the network. In the following, the term "network" shall mainly be used to denote the fixed part, whereas the variable part will be expressed in terms of the input pattern.

The operation principle of the random network can be summarized as follows. After constructing the network in a way to store a number of bipolar patterns, we apply input patterns that constitute noisy versions of stored patterns and determine the sign of nodes. The solution of the nonlinear system (5)–(8) yields the quantities q_i for $i = 1, \cdots, n$. Let us define the bipolar output pattern $\hat{y} = (y_1, \cdots, y_n)$ that is initially set equal to the input pattern $\hat{x}$. Through application of a correction procedure based on information provided by the q_i values, some components of $\hat{y}$ are eventually modified and $\hat{y}$ finally constitutes the output of the network.

A first general approach for defining the correction procedure is based on the fact that the solution of the flow equations yields low and high q_i values for nodes corresponding to wrong and correct components of the input pattern, respectively. This means that the wrong components of the input pattern are expected to correspond to the lowest q_i values. The above behavior of the network has been verified experimentally and accepts a simple intuitive justification depending on the learning rule adopted.

There exists, however, no obvious mechanism for determining the exact number of components that should change their value. In fact, experimental results show that the network clearly tends to enhance the distance between q_i values that are supposed to indicate wrong and correct components, but this differentiation is not always strong enough and can vary following the input pattern. Hence, it is not possible to define a threshold value that could help us perform the discrimination.

Instead, we can successively correct the sign of components of the output pattern starting from the lowest q_i value and following an increasing q_i value order, until a good reconstruction is obtained. Since the number of wrong components is generally not known *a priori*, in order to stop the correction procedure we need a means of identifying stored patterns or, accordingly, of estimating the quality of an output pattern in case perfect reconstruction is not required. The idea is to take advantage of some characteristic of the stored patterns that allows us to recognize whether an output pattern is one of them. Moreover, reliability is enhanced if only the stored patterns exhibit this characteristic.

In conclusion, in order to implement the autoassociative memory operation described above, two issues should be addressed, which can be dealt with either in common or

separately:

- An algorithm for appropriately storing the desired patterns (learning rule);
- a scheme for correctly identifying stored patterns during recall.

The correction procedure is implemented as follows. With each one of the successively produced candidate output patterns $\hat{y}$ (in fact, with each q_i value) we associate a counter, whose value is equal to the number of nodes for which the identification criterion is not satisfied. If this value is equal to zero, then this pattern is accepted as the output. Of course, the initial vector $\hat{y}$ is first considered as a candidate output before performing any correction. Otherwise, if after n steps no such pattern has been encountered, the pattern with the minimum counter value could be taken as the output pattern. However, the correction procedure need not be pushed on to the end if no zero counter value is found. In fact, a moderate level of noise in the input pattern is generally considered; hence, the number of components that should change their sign is expected to be rather low. Therefore, the number of iterations should be bounded by a number ω that is a small percentage of the size of the network. If after ω corrections the counter is not yet zero, we accept the pattern with the minimum counter value so far. Of course, it is always possible to obtain an output that corresponds to a spurious pattern.

The identification of stored patterns considered above is based on characteristics of stored patterns that refer to the pattern as a whole, i.e., provide a criterion as to whether an output pattern is a good reconstruction or not. If, instead, we disposed of a criterion that could be applied to each node separately, thus allowing direct identification of wrong components, we could perform correction in a straightforward manner without resorting to the iterative approach. This idea, which implies a variant of the correction procedure, is investigated later in this section.

C. Identification Based on Stability

To achieve reliability in the identification procedure we can use the fact that in a Hopfield-type associative neural network, all stored patterns should be stable states of the network. This means that once the network has attained such a state, no transition out of this state can occur according to the operation of the network [20]. In other terms, all stored patterns $\{\hat{x}^1, \cdots, \hat{x}^m\}$ must satisfy the following equation [1], [26]:

$$\operatorname{sign}\left(\sum_{j=1}^{n} x_j^k w_{ji}\right) = \operatorname{sign}\left(x_i^k\right), \quad i = 1, \cdots, n, \quad k = 1, \cdots, m. \tag{10}$$

In the case where an outer-product scheme (Hebbian learning) is used for constructing the connection matrix W, the above property of the stored patterns holds with high probability, especially when the number of patterns is significantly smaller than the size of the network [2], [4], [26]. In general, we cannot guarantee that the stability property holds for every pattern $\hat{x}^k$, especially when m has a magnitude comparable to n. This property is guaranteed for every pattern in the case where all patterns $\hat{x}^k$ are orthogonal. This fact can be established with high probability when the patterns $\hat{x}^k$ are generated from a sequence of symmetric Bernoulli trials [2]. The capacity of this scheme is $m = O(n/\log n)$ [1], [26].

When the Hebbian rule is used for computing the matrix W of a random network, the stability property is satisfied by the stored patterns with high probability as stated above. We should note here that in this case the stability property must be viewed as a characteristic of stored patterns expressed mathematically by (10). This property is due to the construction scheme and is not related to the operation of the network, as is the case with the Hopfield network. Being a characteristic of stored patterns the stability property can be used for their identification. Thus, during the correction procedure, each candidate output pattern $\hat{y}$ is associated with a counter, whose value is equal to the number of indexes i that do not satisfy the equation

$$\operatorname{sign}\left(\sum_{j=1}^{n} y_j w_{ji}\right) = \operatorname{sign}(y_i). \tag{11}$$

The pattern with zero counter or minimum counter after ω corrections is taken as the output pattern.

Regarding the computational efficiency of the correction procedure, it is not difficult to verify that an efficient implementation can be obtained. In fact, at the expense of some memory space, we can initially perform the computations necessary for application of the identification criterion to the first candidate pattern and store the results in an appropriate form. Then, at each subsequent step we need only update the stored data by performing the minimum necessary amount of computation relative to the change of sign of the last corrected node.

In [30], a different construction scheme has been proposed, called the *spectral scheme*, which seems to significantly increase the capacity and reconstruction capabilities of associative neural networks. The main advantage of this scheme is that it can guarantee that each pattern $\hat{x}^k$ will be a stable state of the network provided that the m patterns are linearly independent. It seems to exhibit better performance than the outer-product scheme (capacity $m = O(n)$) and it can operate effectively under both synchronous and asynchronous modes of operation. Its main disadvantage consists of the computational cost of constructing the interconnection matrix W.

Several "spectral" formulations of the matrix W have been proposed [12], [30]. We shall make use of the following computation scheme.

We define the $m \times m$ diagonal matrix $\Lambda = dg[\lambda^1, \cdots, \lambda^m]$ and the $n \times m$ matrix $X = [\hat{x}^1, \cdots, \hat{x}^m]$. Then $W = X\Lambda(U^{\mathrm{T}}U)^{-1}U^{\mathrm{T}}$. For a constant choice of the values $\lambda^k = \lambda > 0$, $k = 1, \cdots, m$, the matrix W can be computed iteratively:

$$W[k] = W[k-1] + \frac{e^k e^{k\mathrm{T}}}{\hat{x}^{k\mathrm{T}} e^k}, \quad k = 1, \cdots, m \tag{12}$$

with $W[0] = 0$ and $e^k = (\lambda I - W[k-1])\hat{x}^k$. For this choice of λ^k, the resulting matrix W is symmetric and nonnegative definite.

Based on the spectral approach, we can construct a random network scheme as follows. The connection matrix W is calculated using the procedure described above. As a consequence, all stored patterns correspond to stable states of the corresponding neural network. This increases the reliability of the correction procedure, where we check whether a candidate output pattern $\hat{y}$ has the stability property (satisfies (11)) proceeding exactly as in the previous scheme. It can be verified that there is an improvement in performance under the spectral scheme with respect to the outer-product scheme. This happens because on the one hand the spectral scheme increases the reconstruction capability of the random network, and on the other hand it also increases the reliability during identification. A disadvantage concerns the computational cost of constructing the connection matrix.

D. Correction Based on Node Consistency

The stability property defined in the previous subsection is a characteristic of stored patterns which is closely related to the operation of the Hopfield network. In this section, we define an analogous property of the stored patterns in the context of the random neural network model. This property is based on the notion of *node consistency* defined as follows.

Definition 1: We shall say that a node of the random network is *consistent with its sign* if in the steady-state the internal arrival rate of signals it can accept is greater than the internal arrival rate of signals it rejects, namely:

$$\gamma_{\text{int}}^{+}(i) > \gamma_{\text{int}}^{-}(i), \quad i \in \mathcal{P} \tag{12}$$

$$\gamma_{\text{int}}^{-}(i) > \gamma_{\text{int}}^{+}(i), \quad i \in \mathcal{N}. \tag{13}$$

A random network is called *consistent* if all its nodes are consistent with their sign. □

Definition 2: Suppose that a bipolar pattern $\hat{x} = (x_1, \cdots, x_n)$ is presented as input to a given random network with n nodes. We shall say that the pattern $\hat{x}$ constitutes a *consistent pattern* with respect to the random network if it makes the work consistent, i.e., if it yields

$$x_i\left(\gamma_{\text{int}}^{+}(i) - \gamma_{\text{int}}^{-}(i)\right) > 0, \quad i = 1, \cdots, n\,. \tag{14}$$

□

It is important to notice the similarity between the definition of a stable pattern in the context of the Hopfield neural network model and the definition of a consistent pattern in the context of the random network model [25]. In the Hopfield model the asynchronous update algorithm is based on the stability property of stored patterns. The same principle can be applied in order to develop algorithms for the random network model that exploit the consistency property of stored patterns.

Suppose that we want to store m bipolar patterns in a random network. We claim that any learning algorithm (e.g., Hebbian, spectral) that establishes stable patterns in the context of the Hopfield neural network model also establishes consistent patterns in the context of the random neural network model (with high probability). A justification of this argument follows.

A stable pattern $\hat{y}$ in the context of the Hopfield model satisfies the following condition for each i:

$$\sum_j y_j w_{ji} I(y_j w_{ji} > 0) > \sum_j |y_j w_{ji}| I(y_j w_{ji} < 0), \quad \text{if } y_i = 1 \tag{15}$$

$$\sum_j |y_j w_{ji}| I(y_j w_{ji} < 0) > \sum_j y_j w_{ji} I(y_j w_{ji} > 0), \quad \text{if } y_i = -1 \tag{16}$$

where the indicator function $I(X)$ takes the value 0 if the logical expression X is false; otherwise, it takes the value 1.

A consistent pattern $\hat{y}$ in the context of the random network model satisfies the following condition for each i:

$$\sum_{j\in\mathcal{P}} q_j r(j) p^{+}(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^{-}(j,i) > \sum_{j\in\mathcal{P}} q_j r(j) p^{-}(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^{+}(j,i), \quad i \in \mathcal{P} \quad (\text{i.e., } y_i = 1) \tag{17}$$

$$\sum_{j\in\mathcal{P}} q_j r(j) p^{-}(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^{+}(j,i) > \sum_{j\in\mathcal{P}} q_j r(j) p^{+}(j,i) + \sum_{j\in\mathcal{N}} q_j r(j) p^{-}(j,i), \quad i \in \mathcal{N} \quad (\text{i.e., } y_i = -1)\,. \tag{18}$$

By taking into account (2)–(4), the above equations can be rewritten as follows:

$$\sum_j q_j y_j w_{ji} I(y_j w_{ji} > 0) > \sum_j q_j |y_j w_{ji}| I(y_j w_{ji} < 0), \quad \text{if } y_i = 1 \tag{19}$$

$$\sum_j q_j |y_j w_{ji}| I(y_j w_{ji} < 0) > \sum_j q_j y_j w_{ji} I(y_j w_{ji} > 0), \quad \text{if } y_i = -1. \tag{20}$$

By examining (19) and (20) we can observe that if all the q_j were equal, i.e., $q_j = q$, $j = 1, \cdots, n$, then these equations would be identical to (15) and (16). A great number of experiments has shown that, if a consistent pattern is presented as input to the random network, the resulting q_i values are close to each other. Thus, we expect that if (15) and (16) hold, then (19) and (20) will also hold with high probability.

This is the reason why we consider that the Hebbian (or spectral) rule is good enough for constructing a random network in which the stored patterns will be consistent. As expected, the ability to establish consistent patterns depends

on the relation between n and m as well as on the ability of the used learning algorithm to establish stable patterns in the context of the Hopfield neural network model.

Once a noisy version of a stored pattern is presented as input to the network, the recall procedure can be described as follows. First we solve the nonlinear system of equations that describes the steady-state behavior of the random network model, and compute the analog vector $\hat{q} = (q_1, \cdots, q_n)$. Then we recognize the erroneous nodes using the notion of consistency. We calculate the quantities $\gamma_{\text{int}}^{+}(i)$, $\gamma_{\text{int}}^{-}(i)$ for each $i = 1, \cdots, n$ and check whether condition (14) holds. In case a node i does not satisfy the above condition, there exists an error in the component i of the input pattern. Thus, we can identify erroneous nodes and reconstruct the noisy input pattern. The operation is synchronous since, once the analog vector $\hat{q}$ has been computed, all nodes are simultaneously examined for satisfaction of the consistency criterion. In this way, the effect caused on other nodes by the change of the sign of a node is neglected.

This method is very simple and provides identification and correction based on information that is local to each node of the network. The stability criterion does not feature this locality property and, therefore, implies a correction procedure based on iterative application of the criterion to whole patterns.

As indicated by experimental results, the performance of the method under Hebbian learning is comparable (perhaps slightly worse) to that of the outer-product scheme, but it is preferable in terms of speed and simplicity. In case a spectral learning algorithm is used, the performance is better, but the computational cost is increased.

IV. A Prototype Implementation

Our aim has been to develop a software module that could be integrated in a pictorial information system in order to provide retrieval capabilities based on imprecise or incomplete information. Consider that a number of symbolic images is stored in the system. Each symbolic image is composed of a number of object symbols or icons. The user of the module is provided with a graphical interface that assists him in specifying an incomplete visual sketch of the desired image. The symbolic representation of this sketch is then presented to the module that performs correction and provides the stored symbolic image that is closest to the presented sketch. A visualization of this image is displayed to the user, who either accepts the result or modifies the visual sketch and proceeds in an iterative manner until a satisfactory response is obtained.

During the above operation the user essentially composes a symbolic image based on an imprecise notion of the contents of the database. This symbolic image contains a number of visual clues that are provided by the user and describe the desired image. The system responds by returning a stored symbolic image that is close to the specifications, thus allowing the user to perform information retrieval even if the key presented to the system contains a percentage of error. This is a feature that is generally not provided by existing pictorial information systems. If the module is integrated in a real pictorial database environment, the above phase can be followed by retrieval

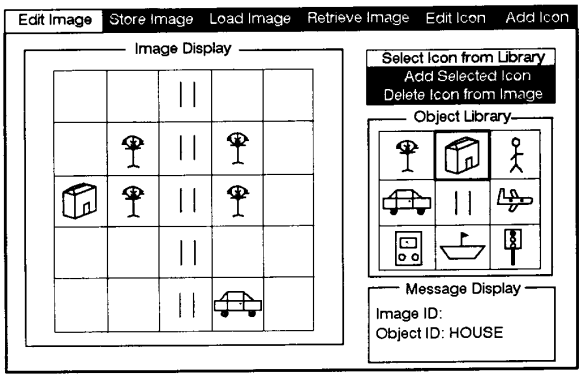


Fig. 2. The graphical interface.

of the corresponding physical image and related information using the symbolic image as the basis for indexing. The use of symbolic images for the construction of database indexes is a major advantage as already stated and provides the coupling mechanism between the neural network-based module and the database.

It is clear from the above specifications that the software module involves two basic components: a graphical interface and the underlying neural network structure. By means of menus appearing at various levels, the interface allows the user to select the necessary operations for treating symbolic images and iconic objects. Objects constitute the elementary entities that can be manipulated by the user, who is provided with a set of object editing facilities. Icons can be created and modified through simple line drawing operations in a rectangular workspace. Each new icon is associated with an internal representation and becomes part of an object library. The user can select objects from the library in order to compose symbolic images. The latter constitute the high-level data structure of the system in that they are defined in terms of elementary objects. As already mentioned, an image is represented as a $c \times c$ array of icons. The graphical workspace for manipulation of symbolic images has the form of a lattice whose contents are determined by assigning objects to cells. A symbolic image created using the above composition technique is either a new registration that is appropriately stored in the network or will be used in a retrieval operation. Symbolic images stored in the network are also stored on a secondary device in a conventional manner. This component was necessary for creating noisy input images and evaluating reconstruction during experiments. The user can index each stored image by a unique identification name, so that a specific image can also be retrieved directly from the secondary device by specifying its identification. If this loading operation is selected, the image is visualized and can be edited following the same technique as for the original creation of images. Iconic objects are also assigned identification names. Fig. 2 displays an instance of image editing operation.

The random neural network operates at low level and constitutes the kernel of the system. The network architecture corresponds to the internal representation of symbolic images. An image is represented as a bipolar pattern composed of

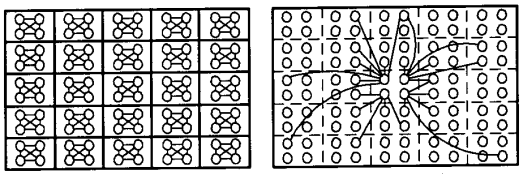


Fig. 3. The hierarchical neural network architecture. (a) Subnetworks. (b) Global network.

smaller subpatterns corresponding to iconic objects. Thus, the network contains $c \times c$ groups of p nodes, each group representing an iconic object. Objects are internally encoded by means of a p-dimensional subpattern vector. The size p of the vector as well as the exact encoding mechanism should yield the maximum possible orthogonality and independence of stored objects. In our implementation, we have used the random generation scheme based on symmetric Bernoulli trials [2], [26], [30] that ensures orthogonality with high probability. However, due to the hierarchical nature of the network architecture, the search for efficient encoding schemes is a subject of further investigation.

The operation of the network is hierarchical as far as both learning and recall are concerned. Storage of information is based on one of the learning schemes (Hebbian or spectral) described in the previous section. An advantage of the above schemes is that learning can be performed in an incremental manner, i.e., each time a new pattern is stored in the network the connection weights can be updated using only information related to that pattern and not to previously stored ones. We distinguish two types of connections: *intra-group* and *inter-group* and two different views of the random network. A first view regards the network as containing $c \times c$ disjoint *subnetworks*, where the (i, j) group of nodes $(1 \leq i, j \leq c)$ together with its intra-group connections constitutes a subnetwork that can operate in isolation. On the other hand, the total of the $c^2 p$ nodes together with the inter-group connections can be viewed as the *global network* (Fig. 3). During registration of a new image, learning involves two types of computation corresponding to the levels of hierarchical architecture.

Intra-group connections of group (i, j) contain knowledge relative to the subpatterns appearing at the (i, j) position of the stored patterns. Updating of intra-group connection weights during registration of a new image is performed as follows. For each subpattern (i, j) of the image pattern we check whether it has been already stored in the (i, j) subnetwork. This can be done by testing the subpattern for satisfaction of the identification criterion (stability or consistency depending on the scheme). If it is already stored, no action is taken; otherwise, the intra-group weights are appropriately updated. Two observations should be made concerning the above procedure. First, each subnetwork stores only the subpatterns appearing at the corresponding position considering all stored images. Second, subpatterns are memorized in a subnetwork on an equal basis, since each of them is stored once independently of the number of images containing it at that particular position. It is obvious that the computation of intra-group connection weights of all subnetworks can be carried out in parallel.

On the other hand, inter-group connection weights are computed by directly applying the learning rule to whole patterns, i.e., by acting on the global network view of the system. Clearly, this type of computation can be also performed in parallel with the computation of intra-group connection weights.

During recall, the hierarchical structure of the system implies two phases of operation. At a first step, the whole pattern is presented as input to the global network and correction is performed yielding an output pattern. This part of the recall procedure is intended to provide correction at the global level, i.e., to locate and eliminate errors related to the existence of wrong subpatterns. However, it is possible that correction at this level does not yield the exact code of a right object due to the appearance of scattered errors of limited extent. These errors are eliminated during the second phase where the output pattern is viewed as consisting of independent subpatterns. Each subpattern is then presented as input to the corresponding network that performs correction at a local level producing a legal object code. A reasonable option, which has been adopted in our implementation, is to perform local correction only on output subpatterns of the first phase, which differ from the respective subpatterns of the original input pattern, i.e., on subpatterns that have been affected by the global correction.

The output pattern obtained during recall is transformed to a symbolic image that is visualized by substituting the iconic objects for the subpattern codes. However, in spite of the correction performed during the two phases, it is always possible that an erroneous output pattern is obtained. Two types of error should be considered. A first type concerns the existence of subpatterns that do not correspond to legal object codes. In this case a special icon is displayed at the corresponding position indicating to the user that the system has encountered difficulties in correcting the error at that position. The user is thus prompted to make another trial. The second type of error concerns the case where all the subpatterns of the output pattern correspond to legal object codes, but the resulting symbolic image does not correspond to a stored one. In a case where our system were integrated in a real pictorial database, this symbolic image would provide an index that would not point to one of the physical images. If this type of error occurs, a visualization of the output pattern is displayed, along with a message indicating to the user the kind of the error. Then, two options are available. Either the user performs another trial or asks the system to repeat retrieval using the wrong output as a new input. The justification for the second option is based on the fact that gradual iterative correction has proved to be effective in many retrieval experiments. This means that the network does not eliminate all errors in one pass, but in each pass the number of errors is gradually decreased until a satisfactory output is obtained.

Our prototype software module has been implemented on a SPARC workstation under Unix and X-Windows. In the next section, we provide experimental results concerning the performance of the module.

V. Experimental Results

In order to evaluate the validity of the method, we have conducted a series of experiments that allowed us to obtain a first insight into the storage and retrieval capabilities of the hierarchical network. The symbolic images were 5×5 (i.e., $c = 5$) arrays of objects, with each object internally encoded as a bipolar vector with $p = 20$ elements. The absence of objects from an image position, i.e., the null object, was also encoded using a special vector. The artificial stored images that were used in the experiments were generated by assigning to each array position an object selected from an object library. Selection of objects was made at random following some nonuniform probability distribution, which varied for different parts of the image. This option stems from the fact that in real applications some types of objects are more likely to occupy a given image position than other ones. Also, care was taken so that the number of empty positions was always in a given range, corresponding to 20–50% of the image.

The distorted images that were presented to the module contained error at the object level. This means that distorted images were derived from the stored ones by changing the objects in some array positions. The rate of distortion (percentage of wrong objects) was 15–20%, and experiments were conducted using the schemes described in the previous section. Both the Hebbian and the spectral learning rules were used for the storage of images, while identification schemes based on both stability and consistency were examined.

Fig. 4 illustrates the success rate of the module under three different schemes concerning learning and identification. For each value of the number of stored images m, the displayed results are average values obtained as follows. For each one of the stored images, 10 retrieval experiments were performed by presenting distorted versions of that image as inputs to the system. An experiment was considered successful if perfect reconstruction of the image were achieved. For this series of experiments we used a library containing $b = 20$ iconic objects. As far as the efficiency of the different schemes is concerned, spectral learning behaves better than Hebbian, whereas the identification criteria based on stability and consistency properties are practically of equivalent performance, the stability criterion being slightly superior. We can observe that the obtained results are quite satisfactory since a success rate superior to 80% is attained under the spectral scheme for up to 60 stored symbolic images. This rate was further improved if iterative correction were applied to wrong output images, as described in the previous section, yielding almost 90% for 60 stored images.

Statistics concerning correction at the global level are shown in Fig. 5 for two different values of the size b of the object library. In all cases, the experiments were conducted as described above using the Hebbian learning rule and identification based on stability. The curves represent the percentage of error obtained at the output of the global network. The error was measured at the level of individual nodes with respect to the correct stored pattern that should be retrieved. We observe that a significant part of the job is carried out during the global correction phase. As indicated by the curves, the

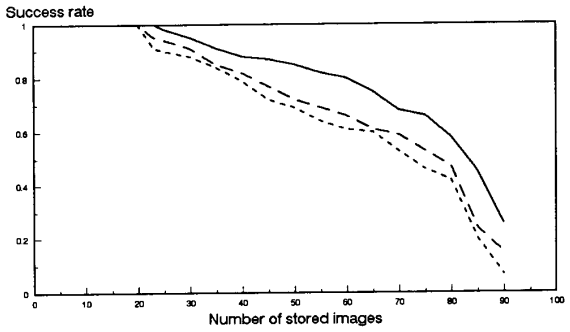


Fig. 4. Total success rate versus number of store images. (a) Solid line: Spectral learning and stability identification. (b) Dashed line: Hebbian learning and stability identification. (c) Dotted line: Hebbian learning and consistency identification.

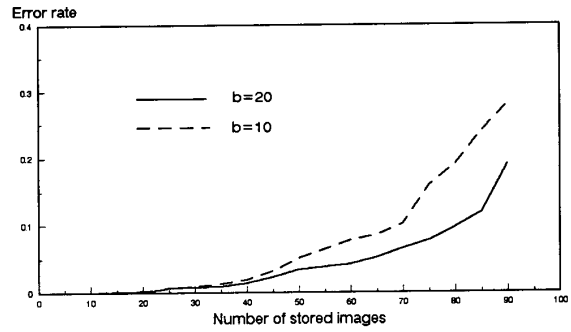


Fig. 5. Level of error at the global network output versus number of stored images.

global network exhibits a better behavior for larger values of b. This seems reasonable because, as b increases, the number of distinct objects in the image also increases leading to stored patterns with a smaller degree of correlation. The error at the output of the global network is distributed producing distortion at the input of subnetworks. The quality of the correction provided by each subnetwork is closely related to distortion appearing at its input. Thus, subnetworks play an auxiliary role and are intended to correct minor defects, the global network providing the principal correction mechanism.

Considering the overall behavior of the system we can state that experimental results validate the potential of the proposed hierarchical technique. The performance of the method is affected by capacity limitations of the network due to the learning schemes considered. As a matter of fact, the measured capacity is slightly inferior to typical values reported for other associative memories [20], which is probably due to the particular structure of stored patterns.

VI. Conclusion

In an attempt to investigate the suitability of neural network techniques for information system applications we have developed an approach providing an autoassociative memory operation during pictorial information retrieval. Our approach has been based on an extension of the random neural network model including two types of nodes with symmetrical

behavior. This symmetry seems to be well adapted to the mechanisms involved in associative memory operation. In fact, this network model seems to naturally fit the autoassociative memory paradigm, and several schemes related to the learning and recall phases can be devised.

The neural network is organized in a hierarchical manner allowing the manipulation of patterns at both the image and iconic object level. The user is provided with a simple yet efficient graphical interface that allows him to interactively perform retrieval of symbolic images using incomplete or erroneous information. The network performs error correction in two phases based on stored knowledge reflecting the structure of images at global and local levels. Experiments performed on a prototype implementation have yielded promising results.

Our experience has shown that neural networks can constitute powerful components of complex information systems providing solutions to problems that are generally difficult to solve by traditional techniques. Their advantages include tolerance to errors and massive parallelism as shown in the context of the application described in this paper. As far as associative memory operations are concerned, a major drawback is that their capacity is a relatively small percentage of their size and their performance is sensitive to algebraic properties of stored patterns. The hierarchical approach proposed here seems particularly efficient but needs further experimentation in order to deal with drawbacks and fully explore its potential. A subject of further investigation concerns the search for other learning schemes that are properly adapted to characteristics of the random neural network. Several algorithms could be considered for increasing capacity, ones that either stem from techniques developed in other contexts or can be derived based on properties of the random network. In this direction, an essential feature of the random neural network model is that the symmetrical weights property is not a necessary condition for its operation, as is the case with other associative memory paradigms.

APPENDIX
STEADY-STATE SOLUTION OF THE EXTENDED MODEL

The steady-state behavior of the system is described by the global balance equations, which can be written as follows for each state $\hat{k} = (k_1, \cdots, k_n)$:

$$
\begin{aligned}
&p(k_1,\cdots,k_n)\Big\{\sum_{i\in\mathcal{P}}\big[\Lambda(i)+\lambda(i)I(k_i>0)+r(i)\big(1-p^+(i,i)\big)I(k_i>0)\big]+\\
&\qquad\sum_{i\in\mathcal{N}}\big[\lambda(i)+\Lambda(i)I(k_i>0)+r(i)\big(1-p^+(i,i)\big)I(k_i>0)\big]\Big\}=\\
&\sum_{i\in\mathcal{P}}\Big\{p(k_1,\cdots,k_i-1,\cdots,k_n)\Lambda(i)I(k_i>0)+p(k_1,\cdots,k_i+1,\cdots,k_n)[\lambda(i)+r(i)d(i)]\Big\}+\\
&\sum_{i\in\mathcal{N}}\Big\{p(k_1,\cdots,k_i-1,\cdots,k_n)\lambda(i)I(k_i>0)+p(k_1,\cdots,k_i+1,\cdots,k_n)[\Lambda(i)+r(i)d(i)]\Big\}+\\
&\qquad\sum_{i\in\mathcal{P}}p(k_1,\cdots,k_i+2,\cdots,k_n)r(i)p^-(i,i)+\\
&\qquad\sum_{i\in\mathcal{N}}p(k_1,\cdots,k_i+2,\cdots,k_n)r(i)p^-(i,i)+\\
&\sum_{i\in\mathcal{P}}\Big\{\sum_{j\in\mathcal{P},j\neq i}p(k_1,\cdots,k_i+1,\cdots,k_j-1,\cdots,k_n)r(i)p^+(i,j)I(k_j>0)+\\
&\qquad\sum_{j\in\mathcal{P},j\neq i}p(k_1,\cdots,k_i+1,\cdots,k_j+1,\cdots,k_n)r(i)p^-(i,j)+\\
&\qquad\sum_{j\in\mathcal{P}}p(k_1,\cdots,k_i+1,\cdots,k_j,\cdots,k_n)r(i)p^-(i,j)I(k_j=0)+\\
&\qquad\sum_{j\in\mathcal{N}}p(k_1,\cdots,k_i+1,\cdots,k_j-1,\cdots,k_n)r(i)p^-(i,j)I(k_j>0)+\\
&\qquad\sum_{j\in\mathcal{N}}p(k_1,\cdots,k_i+1,\cdots,k_j+1,\cdots,k_n)r(i)p^+(i,j)+\\
&\qquad\sum_{j\in\mathcal{N}}p(k_1,\cdots,k_i+1,\cdots,k_j,\cdots,k_n)r(i)p^+(i,j)I(k_j=0)\Big\}+\\
&\sum_{i\in\mathcal{N}}\Big\{\sum_{j\in\mathcal{P}}p(k_1,\cdots,k_i+1,\cdots,k_j-1,\cdots,k_n)r(i)p^-(i,j)I(k_j>0)+
\end{aligned}
$$

$$\sum_{j\in\mathcal{P}} p(k_1,\cdots,k_i+1,\cdots,k_j+1,\cdots,k_n)r(i)p^+(i,j) +$$
$$\sum_{j\in\mathcal{P}} p(k_1,\cdots,k_i+1,\cdots,k_j,\cdots,k_n)r(i)p^+(i,j)I(k_j=0) +$$
$$\sum_{j\in\mathcal{N},j\neq i} p(k_1,\cdots,k_i+1,\cdots,k_j-1,\cdots,k_n)r(i)p^+(i,j)I(k_j>0) +$$
$$\sum_{j\in\mathcal{N},j\neq i} p(k_1,\cdots,k_i+1,\cdots,k_j+1,\cdots,k_n)r(i)p^-(i,j) +$$
$$\sum_{j\in\mathcal{N}} p(k_1,\cdots,k_i+1,\cdots,k_j,\cdots,k_n)r(i)p^-(i,j)I(k_j=0)\Bigg\}$$

where the indicator function $I(X)$ takes the value 0 if the logical expression X is false; otherwise, it takes the value 1. By using (5)–(8) it can be shown that the above equations are satisfied by the solution in (9).

REFERENCES

[1] Y. S. Abu-Mostafa and J.-M. St. Jaques, "Information capacity of the Hopfield model," *IEEE Trans. Inform. Theory*, vol. 31, pp. 461–464, July 1985.

[2] S. V. B. Aiyer, M. Niranjan, and F. Fallside, "A theoretical investigation into the performance of the Hopfield model," *IEEE Trans. Neural Networks*, vol. 1, pp. 204–215, June 1990.

[3] V. Atalay, E. Gelenbe, and N. Yalabik, "Texture generation with the random neural network model," in *Artificial Neural Networks*, Kohonen *et al.* Eds. Amsterdam: North-Holland, 1991, vol. 1, pp. 111–116.

[4] P. Baldi, "Neural networks, orientations of the hypercube, and algebraic threshold functions," *IEEE Trans. Inform. Theory*, vol. 34, pp. 523–530, May 1988.

[5] F. Biennier, J. M. Pinon, and M. Guivarch, "A connectionist method to retrieve information in hyperdocuments," in *Proc. Int. Neural Network Conf.*, vol. 1, pp. 444–448, 1990.

[6] S. K. Chang, E. Jungert, S. Levialdi, T. Tortora, and T. Ichikawa, "An image processing language with icon-assisted navigation," *IEEE Trans. Software Engineering*, vol. 11, pp. 811–819, Aug. 1985.

[7] S. K. Chang and E. Jugert, "A spatial knowledge structure for information systems using symbolic projections," in *Proc. FJCC*, 1986.

[8] S. K. Chang, Q. Y. Shi, and S. W. Yan, "Iconic indexing by 2-D strings," *IEEE Trans. Pattern. Anal. Machine Intell.*, vol. 9, pp. 484–492, July 1984.

[9] S. K. Chang, C. W. Yan, D. Dimitroff, and T. Arndt, "An intelligent image database system," *IEEE Trans. Software Engineering*, vol. 14, pp. 681–688, May 1988.

[10] S. K. Chang, M. Tauber, B. Yu, and J. S. Yu, "A visual language compiler," *IEEE Trans. Software Engineering*, vol. 15, pp. 506–525, May 1989.

[11] S. K. Chang, "A visual language compiler for information retrieval by visual reasoning," *IEEE Trans. Software Engineering*, vol. 16, pp. 1136-1149, Oct. 1990.

[12] A. Dembo, "On the capacity of associative memories with linear threshold functions," *IEEE Trans. Inform. Theory*, vol. 35, pp. 709–720, July 1989.

[13] G. Fahner, "A neural net that manages a hierarchical spatial database," in *Artificial Neural Networks*, Kohonen *et al.* Eds. vol. 1, pp. 441–446, North-Holland, 1991.

[14] E. Gelenbe, "Random neural networks with negative and positive signals and product form solution," *Neural Computation*, vol. 1, pp. 502–510, 1989.

[15] ——, "Stability of the random neural network model," *Neural Computation*, vol. 2, pp. 239–247, 1990.

[16] ——, "Theory of the random neural network model," in *Neural Networks: Advances and Applications*. Amsterdam: North-Holland, 1991, pp. 1–20.

[17] E. Gelenbe, A. Stafylopatis, and A. Likas, "Associative memory operation of the random network model," in *Artificial Neural Networks*, Kohonen *et al.*, Eds. Amsterdam: North-Holland, 1991, pp. 307–312.

[18] M. Gerso, "Information retrieval using self-organizing and heteroassociative supervised neural networks," in *Proc. Int. Neural Network Conf.*, vol. 1, pp. 361–364, 1990.

[19] A. Goodman, R. Haralick, and L. Shapiro, "Knowledge-based computer vision," *Computer*, vol. 22, pp. 43–54, Dec. 1989.

[20] J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Academy Sci.*, vol. 79, pp. 2554–2558, 1982.

[21] S. S. Iyengar and R. L. Kashyap, "Image databases," *IEEE Trans. Software Engineering*, vol. 14, pp. 608–610, May 1988.

[22] T. Joseph and A. F. Cardenas, "PICQUERY: A high level language for pictorial database management," *IEEE Trans. Software Engineering*, vol. 14, pp. 630–638, May 1988.

[23] Y. Kamp and M. Hasler, *Recursive Neural Networks for Associative Memory*. New York: Wiley, 1990.

[24] B. Kosko, *Neural Networks for Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1992.

[25] A. Likas and A. Stafylopatis, "An investigation of the analogy between the random network and the Hopfield network," in *Proc. ISCIS VI*, 1991.

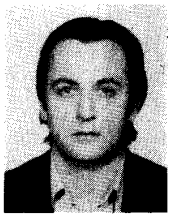
[26] R. J. McEliece, E. C. Posner, E. R. Rodemich, and S. S. Venkatesh, "The capacity of the Hopfield associative memory," *IEEE Trans. Inform. Theory*, vol. 33, pp. 461–482, July 1987.

[27] M. Mokhtari, "Reconnaissance d'Images Typées par le Réseau Neuronal Aléatoire," in *Proc. Int. Colloq. Parallel Image Processing*, June 1991.

[28] A. Motro, "Letter from the Editor," *Data Engineering*, vol. 12, pp. 2–3, 1989.

[29] Y. H. Pao, *Adaptive Pattern Recognition and Neural Networks*. Reading, MA: Addison-Wesley, 1989.

[30] S. S. Venkatesh and D. Psaltis, "Linear and logarithmic capacities in associative neural networks," *IEEE Trans. Inform. Theory*, vol. 35, pp. 558–568, May 1989.



Andreas Stafylopatis received the Diploma degree in electrical and electronics engineering in 1979 from the National Technical University of Athens and the Docteur Ingenieur degree in computer science in 1982 from the University of Paris-Sud, Orsay, France.

Since 1984 he has been with the Division of Computer Science at the National Technical University of Athens, where he is currently an Assistant Professor. His research interests include performance evaluation, information systems, parallel processing, and neural networks.

Dr. Stafylopatis is a member of the IEEE Computer Society, The Association for Computing Machinery, the European Neural Network Society, and the International Neural Network Society.

Aristidis Likas received the Diploma degree in electrical and electronics engineering in 1990 from the National Technical University of Athens, where he is currently a Ph.D. student in computer science. His research interests include parallel processing and neural networks.

Mr. Likas is a student member of the IEEE Computer Society and the Association for Computing Machinery.