IMAGE RECALL USING A LARGE SCALE GENERALIZED BRAIN-STATE-IN-A-BOX NEURAL NETWORK

CHEOLHWAN OH, STANISŁAW H. ŻAK

School of Electrical and Computer Engineering
Purdue University, West Lafayette
Indiana 47907–2035, USA
e-mail: `oh2@ecn.purdue.edu, zak@purdue.edu`

An image recall system using a large scale associative memory employing the generalized Brain-State-in-a-Box (gBSB) neural network model is proposed. The gBSB neural network can store binary vectors as stable equilibrium points. This property is used to store images in the gBSB memory. When a noisy image is presented as an input to the gBSB network, the gBSB net processes it to filter out the noise. The overlapping decomposition method is utilized to efficiently process images using their binary representation. Furthermore, the uniform quantization is employed to reduce the size of the data representation of the images. Simulation results for monochrome gray scale and color images are presented. Also, a hybrid gBSB-McCulloch-Pitts neural model is introduced and an image recall system is built around this neural net. Simulation results for this model are presented and compared with the results for the system employing the gBSB neural model.

Keywords: associative memory, Brain-State-in-a-Box (BSB) neural network, overlapping decomposition, image recall

1. Introduction

This paper is on using a class of nonlinear dynamical systems for image recall. A specific discrete-time dynamical system, referred to as the generalized Brain-State-in-a-Box (gBSB) neural network, is used to build an image recall system. It stores original images in the neural associative memory, and when a noisy image is presented to the system, it reconstructs the corresponding original image after the classification process of the neural network and the proposed error correction process.

The Brain-State-in-a-Box (BSB) net is a simple nonlinear autoassociative neural network that was proposed by Anderson *et al.* (1989) as a memory model based on neurophysiological considerations. The BSB model gets its name from the fact that the network trajectory is constrained to be in the hypercube $H_n = [-1, 1]^n$. The BSB model was used primarily to model effects and mechanisms seen in psychology and cognitive science (Anderson, 1995). A possible function of the BSB net is to recognize a pattern from its given noisy version. The BSB net can also be used as a pattern recognizer that employs a smooth nearness measure and generates smooth decision boundaries (Schultz, 1993). Three different generalizations of the BSB model were proposed by different research groups: Hui and Żak (1992), Golden (1993), and Anderson (1995). In particular, the network considered by Hui and Żak (1992), referred to as the generalized Brain-State-in-a-Box (gBSB), has the property that the network trajectory constrained to a hyperface of H_n is described by a lower-order gBSB type model. This interesting property helps significantly to analyze the dynamical behavior of the gBSB net. Another tool that makes the gBSB model suitable for constructing associative memory is the stability criterion of the vertices of H_n (Hui and Żak, 1992), see also (Hassoun, 1995) for a further discussion of the condition. The gBSB neural network is suitable for associative memory because it can store patterns as its stable equilibrium points. Lillo *et al.* (1994) proposed a systematic method to synthesize associative memories using the gBSB neural network. This method is used to design large scale associative memories by Oh and Żak (2002; 2003). The hybrid gBSB-McCulloch-Pitts neural network uses the same activation function as the McCulloch-Pitts neural network but the argument of the activation function is the same as in the gBSB net. Both the gBSB neural network and the the hybrid gBSB-McCulloch-Pitts net are suitable for associative memory because they can store patterns as their stable equilibrium points.

Designing associative memory using the decomposition concept has been investigated by numerous researchers (Akar and Sezer, 2001; Ikeda *et al*, 2001; Oh and Żak, 2002; 2003; Zetzsche, 1990). Ikeda *et al.* (2001) used a disjoint decomposition to design large scale associative memories. They employed the two-level network concept to alleviate the problem of spurious states caused by disjoint decomposition. Akar and Sezer (2001) used overlapping decomposition. The concept of overlapping decomposition was used earlier in the context of decentralized control of large scale dynamical systems (Ikeda and Šiljak, 1980). Akar and Sezer applied the concept of overlapping decomposition to the associative memory design, where they focused on the decomposition of the weight matrix. In their paper, the weight matrix of each decomposed sub-network must satisfy certain conditions. That is, a certain portion of each weight matrix must have all zero elements and another portion of the weight matrix of each sub-network must coincide with a certain portion of the weight matrix of a neighboring sub-network. This means that a sub-network cannot be designed separately from its neighboring sub-networks. Also, this may complicate the design process of associative memories if the number of sub-networks is large. In Oh and Żak's papers (2002; 2003), overlapping decomposition was carried out on the patterns, rather than on the weight matrix of the associative memory. This approach eliminates the restrictions on the weight matrices of sub-networks, and therefore allows designing each sub-network of associative memory independently.

In this paper, we propose an image recall system using the large scale associative memory proposed by Oh and Żak (2002; 2003) to efficiently process the images. The original images are stored in a large scale neural associative memory. When a noisy image is given as an input to the proposed image recall system, it is decomposed into sub-images and processed by neural networks and the error correction processor to reconstruct the corresponding original image. We assume that the test images are simply noisy versions of training images and they are perfectly aligned with the training images. We do not consider in this article the cases when the test images are rotated, scaled, or transformed to break the alignment with the training images.

The paper is organized as follows: In Section 2, we review the structure and stability conditions of the gBSB model and the synthesis procedure of associative memories using the gBSB neural model. Also, we introduce the hybrid gBSB-McCulloch-Pitts neural model and discuss its stability conditions briefly. In Section 3, we discuss the overlapping decomposition method that is employed to construct a large scale neural net to process large scale patterns. Also, we introduce the error correction algorithm that we employed to enhance the performance of the associative memory. In Section 4, we discuss image representation that is used in the proposed image recall system. Simulation results are presented in Section 5. Finally, conclusions are found in Section 6.

2. Overview of the Generalized Brain-State-in-a-Box (gBSB) and the Hybrid gBSB-McCulloch-Pitts Neural Models

In this paper, we use a gBSB neural model to design an image recall processor. The gBSB neural model has its root in the Brain-State-in-a-Box (BSB) neural model proposed by Anderson *et al.* in 1977 (cf. Anderson *et al.*, 1989). The dynamics of the BSB neural model are described by the difference equation

$$\boldsymbol{x}(k+1) = \boldsymbol{g}\big(\boldsymbol{x}(k) + \alpha \boldsymbol{W} x(k)\big), \tag{1}$$

with an initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$, where $\boldsymbol{x}(k) \in \mathbb{R}^n$ is the state of the BSB neural network at time k, $\alpha > 0$ is a step size, $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a symmetric weight matrix, and the activation function $\boldsymbol{g} : \mathbb{R}^n \to \mathbb{R}^n$ is defined as

$$\big(\boldsymbol{g}(\boldsymbol{x})\big)_i = \big(\mathbf{sat}(\boldsymbol{x})\big)_i \equiv \begin{cases} 1 & \text{if } x_i \geq 1, \\ x_i & \text{if } -1 < x_i < 1, \\ -1 & \text{if } x_i \leq -1. \end{cases} \tag{2}$$

The following definitions are used in the further discussion:

Definition 1. A point $\boldsymbol{x}_e$ is an *equilibrium state* of the dynamical system $\boldsymbol{x}(k+1) = \boldsymbol{T}(\boldsymbol{x}(k))$ if $\boldsymbol{x}_e = \boldsymbol{T}(\boldsymbol{x}_e)$.

We will be concerned with the equilibrium states that are vertices of the hypercube $H_n = [-1,1]^n$. That is, each equilibrium state belongs to the set $\{-1,1\}^n$.

Definition 2. An equilibrium state $\boldsymbol{x}_e$ of $\boldsymbol{x}(k+1) = \boldsymbol{T}(\boldsymbol{x}(k))$ is *super stable* if there exists $\epsilon > 0$ such that for any $\boldsymbol{y}$ satisfying $\|\boldsymbol{y} - \boldsymbol{x}_e\| < \epsilon$, we have $\boldsymbol{T}(\boldsymbol{y}) = \boldsymbol{x}_e$, where $\|\cdot\|$ may be any p-norm of a vector.

Definition 3. A *basin of attraction* of an equilibrium state of the gBSB neural model is the set of points such that the trajectory of the gBSB model emanating from any point in the set converges to the equilibrium state.

The BSB neural net can be used to construct neural associative memory, where each pattern vector is stored as a super stable equilibrium state. When a given initial state is located in the basin of attraction of a certain stored pattern, the network trajectory starting from this given initial condition converges to the pattern. The given initial state can be interpreted as a noisy version of the corresponding stored pattern. In other words, the BSB network can recall the stored pattern successfully from a noisy vector if the noisy pattern is located close enough to the desired pattern.

A generalization of the BSB neural network, referred to as the generalized Brain-State-in-a-Box (gBSB) neural model, was proposed by Hui and Żak (1992). The gBSB neural model is characterized by a nonsymmetric weight matrix and it offers more control of the extent of the basins of attraction of the equilibrium states. The dynamics of the gBSB model are represented by

$$\boldsymbol{x}(k+1) = \boldsymbol{g}\big((\boldsymbol{I}_n + \alpha \boldsymbol{W})\boldsymbol{x}(k) + \alpha \boldsymbol{b}\big), \tag{3}$$

where $\boldsymbol{g}(\boldsymbol{x}) = \mathbf{sat}(\boldsymbol{x})$ as in (2), $\boldsymbol{I}_n$ is an $n \times n$ identity matrix, $\boldsymbol{b} \in \mathbb{R}^n$ is a bias vector, and $\boldsymbol{W} \in \mathbb{R}^{n\times n}$ is a weight matrix that is not necessarily symmetric, which makes it easier to implement associative memories when using the gBSB net.

We now discuss the stability condition that we use when designing associative memory using the gBSB model. Let

$$\boldsymbol{L}(\boldsymbol{x}) = (\boldsymbol{I}_n + \alpha \boldsymbol{W})\boldsymbol{x} + \alpha \boldsymbol{b},$$

and let $(\boldsymbol{L}(\boldsymbol{x}))_i$ be the i-th component of $\boldsymbol{L}(\boldsymbol{x})$. Suppose

$$\boldsymbol{v} = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}^T \in \{-1,1\}^n,$$

that is, $\boldsymbol{v}$ is a vertex of H_n. This vertex $\boldsymbol{v}$ is an equilibrium state of the gBSB model if and only if

$$\big(\boldsymbol{L}(\boldsymbol{v})\big)_i v_i \geq 1, \quad i = 1, 2, \dots, n.$$

We can show that if

$$\big(\boldsymbol{L}(\boldsymbol{v})\big)_i v_i > 1, \quad i = 1, 2, \dots, n, \tag{4}$$

then $\boldsymbol{v}$ is a super stable equilibrium state of the gBSB neural model (Hui and Żak, 1992; Lillo *et al.*, 1994).

When the desired patterns are given, the associative memory should be able to store them as super stable equilibrium states of the gBSB network. Also, it should minimize the number of super stable equilibrium states that do not correspond to the given stored patterns. Such undesired equilibrium states are called *spurious states*. We can synthesize associative memories using the method proposed by Lillo *et al.* (1994). This design method is used by us in this paper. We now briefly present the method.

For given pattern vectors $\boldsymbol{v}^{(j)} \in \{-1,1\}^n, j = 1, 2, \dots, r$, that we wish to store, we first form a matrix $\boldsymbol{B} = [\boldsymbol{b} \;\; \boldsymbol{b} \;\; \cdots \;\; \boldsymbol{b}] \in \mathbb{R}^{n\times r}$, where

$$\boldsymbol{b} = \sum_{j=1}^{r} \epsilon_j \boldsymbol{v}^{(j)}, \quad \epsilon_j \geq 0, \quad j = 1, 2, \dots r,$$

and ϵ_j's are design parameters. Then, choose $\boldsymbol{D} \in \mathbb{R}^{n\times n}$ such that

$$d_{ii} > \sum_{k=1, k\neq i}^{n} |d_{ik}|, \quad \text{and}$$

$$d_{ii} < \sum_{k=1, k\neq i}^{n} |d_{ik}| + |b_i|, \quad i = 1, 2, \dots, n,$$

and $\boldsymbol{\Lambda} \in \mathbb{R}^{n\times n}$ such that

$$\lambda_{ii} < -\sum_{k=1, k\neq i}^{n} |\lambda_{ik}| - |b_i|, \quad i = 1, 2, \dots, n.$$

Finally, determine the weight matrix $\boldsymbol{W}$ using the formula

$$\boldsymbol{W} = (\boldsymbol{D}V - \boldsymbol{B})\boldsymbol{V}^{\dagger} + \boldsymbol{\Lambda}(\boldsymbol{I}_n - \boldsymbol{V}V^{\dagger}), \tag{5}$$

where $\boldsymbol{V} = [\; \boldsymbol{v}^{(1)} \;\; \boldsymbol{v}^{(2)} \;\; \cdots \;\; \boldsymbol{v}^{(r)} \;] \in \{-1,1\}^{n\times r}$ and $\boldsymbol{V}^{\dagger}$ is a pseudo-inverse matrix of $\boldsymbol{V}$. $\boldsymbol{W}$ and $\boldsymbol{b}$ constructed in such a way are used to implement associative memory that is guaranteed to store the given patterns as super stable vertices of the hypercube H_n. A further discussion of the above method can be found in some papers (Lillo *et al.*, 1994; Park and Park, 2000; Park *et al.*, 1999).

Next, we introduce the hybrid gBSB-McCulloch-Pitts neural network which is characterized by the same difference equation (3) with $\boldsymbol{g}$ being an activation function defined as a standard sign function,

$$\big(\boldsymbol{g}(\boldsymbol{x})\big)_i = \big(\mathbf{sign}(\boldsymbol{x})\big)_i = \begin{cases} 1 & \text{if } \; x_i > 0 \\ 0 & \text{if } \; x_i = 0 \\ -1 & \text{if } \; x_i < 0. \end{cases} \tag{6}$$

We give the stability condition that we use when designing associative memory using the above model. As before, we suppose that $\boldsymbol{v} = [\; v_1 \;\; v_2 \;\; \cdots \;\; v_n \;]^T \in \{-1,1\}^n$. Then, this vertex is a super stable equilibrium state of the above model if

$$\big(\boldsymbol{L}(\boldsymbol{v})\big)_i v_i > 0, \quad i = 1, 2, \dots, n. \tag{7}$$

It is easy to see that (4) implies (7). This means that $\boldsymbol{W}$ and $\boldsymbol{b}$ constructed for the gBSB network will also work for the hybrid gBSB-McCulloch-Pitts net.

3. Overlapping Decomposition Method and an Error Correction Algorithm for High Dimensional Patterns

3.1. Idea of the Overlapping Decomposition

The designers of most associative memories face the serious problem of the quadratic growth of the number of interconnections with the problem size. Oh and Żak (2002; 2003) attacked these difficulties by proposing a design method of associative memory as a system of decomposed neural networks. As pointed out by Ikeda *et al.* (2001), a completely decoupled modularization gives rise to spurious memory problems. In Oh and Żak's papers, the idea of overlapping decomposition was used rather than decoupled modularization. Also, to further enhance the performance of the method, an error correction algorithm was added. The error correction algorithm builds on overlapping idea; that is, the error correction algorithm arises naturally from the overlapping decomposition. The reduced size of each sub-network of the memory system may reduce the recall capacity of the memory. By adding the error correction algorithm, the recall performance of the proposed neural associative memory is enhanced.

When the gBSB network is in the recall mode, it follows from (3) that the main computational load of the network results from the multiplication of a weight matrix $\boldsymbol{W}$ and a state vector $\boldsymbol{x}$. The number of multiplications of the elements of $\boldsymbol{W}$ and $\boldsymbol{x}$ is of the order of n^2, where n is the dimension of the state vector $\boldsymbol{x}$. The same is true for the number of additions. This results in a heavy computational cost when the dimension of the pattern vector is large.

We decompose each stored pattern into sub-patterns and construct gBSB sub-networks using (5). That is, each gBSB sub-network stores the corresponding sub-patterns of the stored patterns. We employ the overlapping decomposition method and the error correction algorithm to improve the performance of neural networks. Each sub-network can be designed independently of others and the recall error may be reduced using the error correction process (Oh and Żak, 2002; 2003).

An example of overlapping decomposition of a pattern is shown in Fig. 1. This is the case when the pattern is represented as a 2-dimensional image. As we can see in Fig. 1, the original pattern is decomposed so that there exist overlapping parts between neighboring sub-patterns.

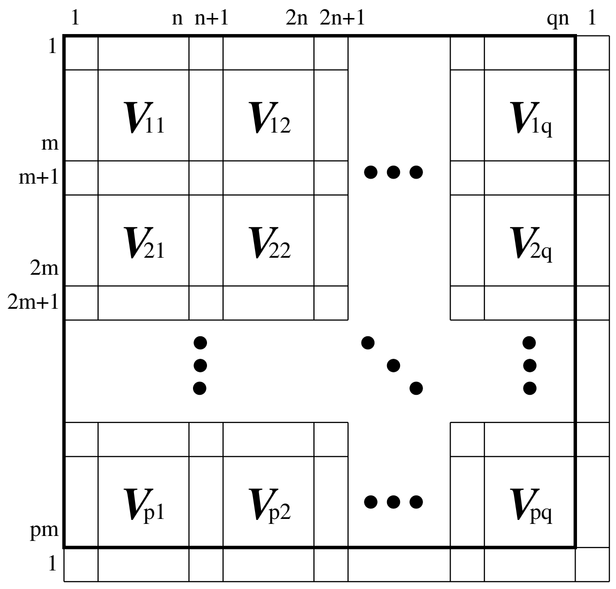


Fig. 1. Example of toroidal overlapping decomposition of a given pattern.

Also, by adopting a toroidal structure, we ensure a symmetry of the image partitions, which makes it easy to implement image processing sub-networks.

3.2. Computational Complexity of the Overlapping Decomposition Method

To investigate the computational complexity of the overlapping decomposition method, consider the pattern shown in Fig. 1. The dimension of the original pattern is $pm \times qn$. Therefore, if we used just one gBSB neural network to process this image, its weight matrix $\boldsymbol{W}$ would be a $pqmn \times pqmn$ matrix containing $(pqmn)^2$ elements. If we use an image partition shown in Figure 1, then the dimension of each sub-pattern is $(m+1) \times (n+1)$. Hence, the number of elements of the weight matrix of each sub-network is $(m+1)^2(n+1)^2$. Because there are pq weight matrices, the overall number of their elements is $pq(m+1)^2(n+1)^2$. Therefore, the ratio of the sizes of the weight matrices of sub-networks to the size of the single processing network is

$$\frac{pq(m+1)^2(n+1)^2}{(pqmn)^2} = \frac{1}{pq}\left(1+\frac{1}{m}\right)^2\left(1+\frac{1}{n}\right)^2.$$

If the sub-networks are designed so that the number of overlapping rows is m_o and the number of overlapping columns is n_o, then the above ratio becomes

$$\frac{1}{pq}\left(1+\frac{m_o}{m}\right)^2\left(1+\frac{n_o}{n}\right)^2.$$

Let

$$f(p) = \frac{1}{p}\left(1+\frac{m_o}{m}\right)^2,$$

and $pm = M$, where M is a constant. We assume that $m_o < m = M/p$, that is, the number of overlapping rows is smaller than the number of rows of the sub-pattern. Then,

$$\begin{aligned} f(p) &= \frac{1}{p}\left(1+\frac{m_o}{M}p\right)^2 \\ &= \frac{1}{p}\left(1+\frac{2m_o}{M}p+\frac{m_o^2}{M^2}p^2\right) \\ &= \frac{1}{p}+\frac{2m_o}{M}+\frac{m_o^2}{M^2}p. \end{aligned}$$

Therefore,

$$\frac{\mathrm{d}f(p)}{\mathrm{d}p} = -\frac{1}{p^2}+\frac{m_o^2}{M^2} < 0$$

because $0 < p < M/m_o$ from the above assumption. This means that $f(p)$ is a decreasing function of p in the interval $0 < p < M/m_o$. If we let

$$g(q) = \frac{1}{q}\left(1+\frac{n_o}{n}\right)^2 = \frac{1}{q}\left(1+\frac{n_o}{N}q\right)^2,$$

where $qn = N$ is a constant and $n_o < n = N/q$, $g(q)$ is a decreasing function of q in the interval $0 < q < N/n_o$. This demonstrates that the amount of computer resources occupied by the weight matrices becomes smaller as the numbers p and q grow.

Let us now consider the same decomposition as in Fig. 1 to determine the number of multiplications between the elements of $\boldsymbol{W}$ and $\boldsymbol{x}(k)$ in (3) during the recall process when only one neural network is used. Because $\boldsymbol{W}$ is a $pqmn \times pqmn$ matrix and $\boldsymbol{x}(k)$ is a $pqmn$-dimensional vector in the undecomposed network, $(pqmn)^2$ multiplications between the elements of $\boldsymbol{W}$ and $\boldsymbol{x}(k)$ are needed for a transition to the next state $\boldsymbol{x}(k+1)$. With the same reasoning, $(m+1)^2(n+1)^2$ multiplications are needed in a sub-network if we use the overlapping decomposition method to process the image. Considering that there are pq sub-networks, we get a total of $pq(m+1)^2(n+1)^2$ multiplications. Therefore, if we let N_m be the number of multiplications in the gBSB network that is not decomposed, then the total number of multiplications in the network consisting of decomposed sub-networks is reduced to

$$\frac{1}{pq}\left(1+\frac{1}{m}\right)^2\left(1+\frac{1}{n}\right)^2 N_m.$$

In the case when there are m_o overlapping rows and n_o overlapping columns, the number of multiplications is

$$\frac{1}{pq}\left(1+\frac{m_o}{m}\right)^2\left(1+\frac{n_o}{n}\right)^2 N_m.$$

This shows that the number of multiplications in the recall process of neural associative memory is also decreasing as the numbers p and q grow in the interval $0 < p < M/m_o$ and $0 < q < N/n_o$.

3.3. Reducing the Classification Error in the Image Recall Process

While the number of computations is reduced by the smaller dimensions of the decomposed sub-patterns and corresponding sub-networks, the capacity of each sub-network is lower than that of the network that is not decomposed. This may lead to a high recall error rate. To reduce this error rate, we add the error correction stage after the recall stage in order to correct possible recall errors (Oh and Żak, 2002; 2003). The overlapping decomposition plays an important role in the error correction procedure. Every sub-pattern overlaps with four neighboring sub-patterns in the decomposition of Fig. 1. After the recall process, we check the number of mismatches of overlapping portions for each sub-pattern. We record the number of overlapping portions in which mismatches occur for each sub-pattern. The number of mismatched overlapping portions is an integer from the set $\{0, 1, 2, 3, 4\}$. We first check if there are sub-patterns with no mismatches. If such a pattern is found, the algorithm is initiated by locating a marker on the above sub-pattern and then the marker moves horizontally to the next sub-pattern in the same row until a sub-pattern with mismatches is encountered. If all sub-patterns have mismatches, we select a sub-network with the minimal number of mismatches. Suppose that the marker is located on the sub-network N_{ij}, and assume that the right and the bottom portions of $\boldsymbol{V}_{ij}$ have mismatches. Note that the decomposed input pattern corresponding to the sub-network N_{ij} is denoted as $\boldsymbol{X}_{ij}$. We denote by $\boldsymbol{V}_{ij}$ the result of the recall process, see Figs. 1 and 2 for an explanation of this notation. The (n+1)-th column of $\boldsymbol{X}_{ij}$ is replaced with the first column of $\boldsymbol{V}_{i,j+1}$, and the (m+1)-th row of $\boldsymbol{X}_{ij}$ is replaced with the first row of $\boldsymbol{V}_{i+1,j}$. That is, the algorithm replaces the mismatched overlapping portions of $\boldsymbol{X}_{ij}$ with the corresponding portions of its neighboring sub-patterns $\boldsymbol{V}_{i,j+1}$, $\boldsymbol{V}_{i,j-1}$, $\boldsymbol{V}_{i+1,j}$, or $\boldsymbol{V}_{i-1,j}$, which are the results of the recall process of the corresponding sub-networks. After the replacement, the sub-network goes through the recall process again and examines the number of mismatches of the resultant sub-pattern. If the number of resultant mismatched portions is smaller than the previous one, the algorithm keeps this modified result. If the number of mismatched portions is not smaller than the previous one, the previous result $\boldsymbol{V}_{ij}$ is kept. Then, the marker moves horizontally to the next sub-network. After the marker returns to the initial sub-network, it then moves vertically to the next row and repeats the same procedure for the new row. Note that the next row of the p-th row of the pattern shown in Fig. 1 is its first row. The error correction stage is finished when the marker returns to the sub-network that the marker initially started from. We can repeat the error correction algorithm so that the sub-patterns can go through the error correction stage multiple times.

The main idea behind the error correction algorithm is to replace the incorrectly recalled elements of the sub-pattern with the ones from the neighboring sub-patterns and let the sub-pattern modified in this way go through the recall process again. If the elements from the neighboring sub-patterns are correctly recalled, it is more probable that the current sub-pattern will be recalled correctly at the error correction stage. The reason is that we might have put the sub-pattern in the basin of attraction of the training sub-pattern by replacing the overlapping elements.

In summary, the proposed neural image recall system operates as follows: Prototype images are decomposed into sub-image patterns with a toroidal overlapping decomposition structure, and the corresponding individual sub-networks are constructed using (5) independently of other sub-networks. The overlapping portions of adjacent stored sub-patterns coincide with each other if they are decomposed from the same pattern. When the noisy

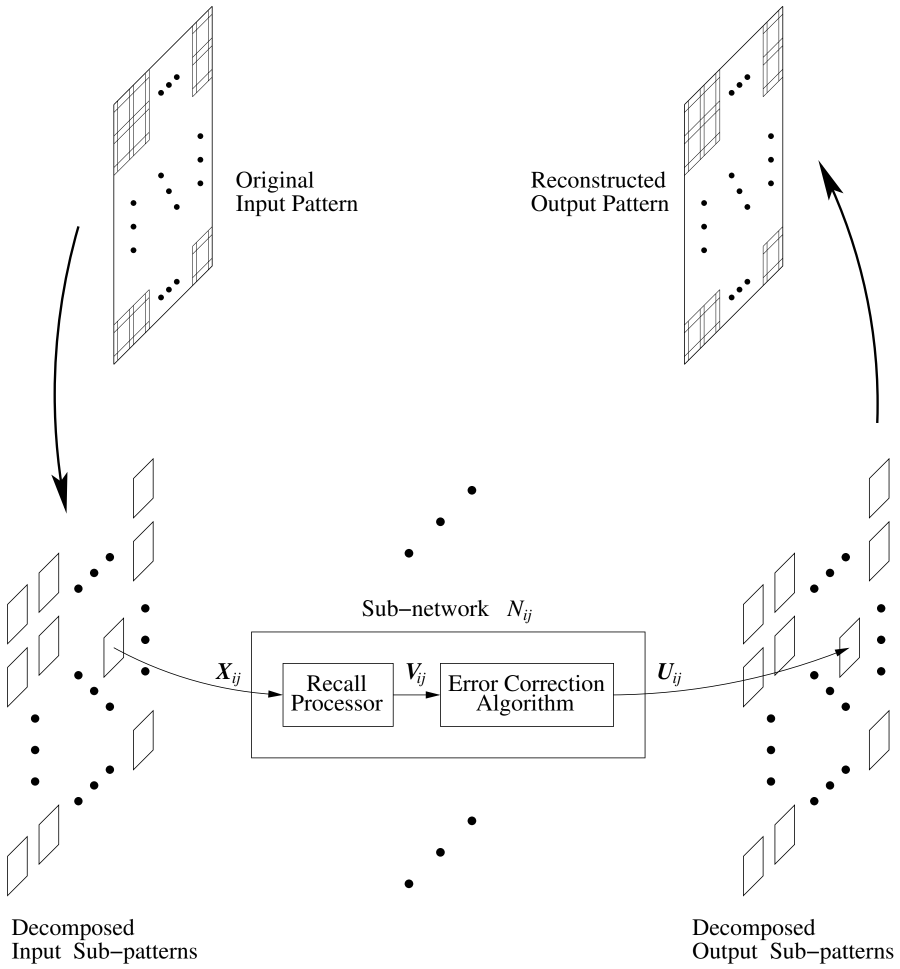


Fig. 2. Image recall procedure using gBSB neural sub-networks with overlapping decompositions and the error correction algorithm.

image is presented to the network, it is decomposed into sub-patterns. Then, each sub-pattern is assigned to a corresponding sub-network as an initial condition, and each sub-network starts processing the initial sub-pattern. After all the individual sub-networks finish processing their sub-patterns, the overlapping portions are checked if they match with adjacent sub-patterns. If the recall process is completed without recall errors, all the overlapping portions of the sub-patterns processed by the corresponding sub-networks would match with their corresponding neighboring sub-patterns. If a mismatched boundary is found between two adjacent sub-patterns, we conclude that a recall error occurred in at least one of the two neighboring sub-networks during the recall process. In other words, the network detects a recall error. Once an error is detected, the error correction algorithm described above is used to correct the recall errors. After the error correction process is finished, we combine the resultant sub-patterns to reconstruct the image. The flow of the pattern processing above is illustrated in Fig. 2.

4. Representation of an Image as a Pattern for gBSB-Based Neural Associative Memory

4.1. Image Representation

An image can be defined as a function $f(x, y)$, where x and y are spatial coordinates (Gonzalez and Wintz, 1987). For monochrome images, f is a scalar function that represents light intensity at each point. A digital monochrome image may be represented by a matrix in the 2-dimensional space, whose (i, j)-th element is $f(i, j)$, where (i, j) is a discrete spatial coordinate. Each element of the matrix is called a pixel. The pixel in digital images usually has an integer value in a finite range so that it can be represented by a finite number of binary digits. For example, if $0 \le f(i, j) \le L - 1$, then each pixel can be represented by $\lceil \log_2 L \rceil$ bits, where $\lceil \cdot \rceil$ denotes the ceiling operator. Consequently, a digital monochrome image can be represented by a matrix whose elements are binary strings of finite length.

The RGB color coordinate system is one of the most popular color coordinate systems. Based on the RGB color coordinate system, each pixel of a digital color image is represented by three components: red, green, and blue. In other words, $f(i,j)$ is a vector valued function in a digital color image, and the digital color image can be represented as three matrices, each corresponding to the red, green, or blue components of the image. Because the three components of each pixel have integer values in a finite range, we can use binary strings of finite length to represent a digital color image.

As we have seen in Section 2, gBSB-based neural associative memory can be used as a pattern classifier if the patterns are represented by vectors whose elements are -1 or $+1$. Because digital images can be represented with binary numbers, they can be used as patterns to be processed by the gBSB net. After representing each pixel as a binary number, we process each digit of the binary number as an element of a pattern vector. For example, if the monochrome image is represented by an $M \times N$ matrix and the binary representation of each pixel has k bits, the size of a pattern becomes $M \times N \times k$. Using a gBSB net, a noisy input image can be classified into one of the stored images if its vector representation is located within the basin of attraction of a stored image in the gBSB neural network memory.

In the following section, we propose using a uniform quantization to further reduce the computational load of the image recall process.

4.2. Uniform Quantization

In our image recall system, all the images to be processed are in the digitized format, that is, each pixel of an image is assumed to be already represented by a binary number with a fixed number of digits. Our goal in this section is to present a technique that reduces the number of bits required to represent a pixel. We propose to use uniform quantization of images, which can be viewed as a simple image compression method. This will lighten the computational complexity of the image recall process. In our further discussion, we use the following definition (Gray and Neuhoff, 1998):

Definition 4. A *quantizer* is a mapper defined on a set of intervals $\mathcal{S} = \{S_i; i \in \mathcal{I}\}$, where the index set $\mathcal{I}$ is ordinarily a collection of consecutive integers beginning with 0 or 1, together with a set of reconstruction levels $\mathcal{C} = \{y_i; i \in \mathcal{I}\}$, so that the overall quantizer q is defined by $q(x) = y_i$ for $x \in S_i$, which can be expressed concisely as

$$q(x) = \sum_i y_i 1_{S_i}(x),$$

where the indicator function $1_S(x)$ is 1 if $x \in S$ and 0 otherwise.

A quantizer is said to be uniform if the reconstruction levels are equispaced and the thresholds are midway between adjacent levels (Gray and Neuhoff, 1998). All intervals in the uniform quantizer have the same size except possibly the outermost intervals (Sayood, 1996).

Suppose now that we have a k-bit image and want to represent this image with $m\,(< k)$ bits per pixel using uniform quantization. The number of levels of the original image is 2^k, and that of the quantized image is 2^m. The simplest way to perform uniform quantization is to divide the range $[0, 2^k - 1]$ using 2^m intervals of the same length and assign a binary number with m digits to each interval. For example, assume that we want to quantize an 8-bit monochrome image uniformly using 2-bits-per-pixel representation. To do this, we divide the range $[0, 255]$ into 4 intervals such as $[0, 63], [64, 127], [128, 191], [192, 255]$, and assign binary numbers $00, 01, 10, 11$ to each interval.

In this paper, we used a slightly different way to quantize images. Instead of dividing the range $[0, 2^k - 1]$ into intervals of the same length, we allocated the same length to the inner intervals and we assigned half the length of them to two outermost intervals. As an example, we divide the range $[0, 255]$ into 4 intervals such as $[0, 42], [43, 127], [128, 212], [213, 255]$ and assign $00, 01, 10, 11$ to each interval. The reason why we used this scheme was because the images we used in our simulations had a lot of extreme pixel values, i.e., 0 and 255.

5. Simulation Experiments

We simulated our proposed image recall system using 150×150 gray scale images as test images. The pixels of original images were represented with 8 bits. To reduce the computational load, we carried out the uniform quantization described in Section 4.2 so that the quantized images could be represented with 6 bits per pixel. These image patterns are shown in Fig. 3. We simulated the image recall system with the gBSB neural networks and the hybrid gBSB-McCulloch-Pitts neural nets.

An example result of the image recall with gBSB neural networks is shown in Fig. 4. The input image in Fig. 4(a) is a noisy version of a stored image pattern. The noise in this image is the so-called ‘salt-and-pepper noise’ with the error probability of 0.5. In other words, each pixel might be corrupted by a noise with the probability of 0.5, and this noisy pixel is white or black with the same probability. The input image was quantized employing the same method as was used for the stored image patterns. The whole image pattern was decomposed into $100\,(10 \times 10)$ sub-patterns

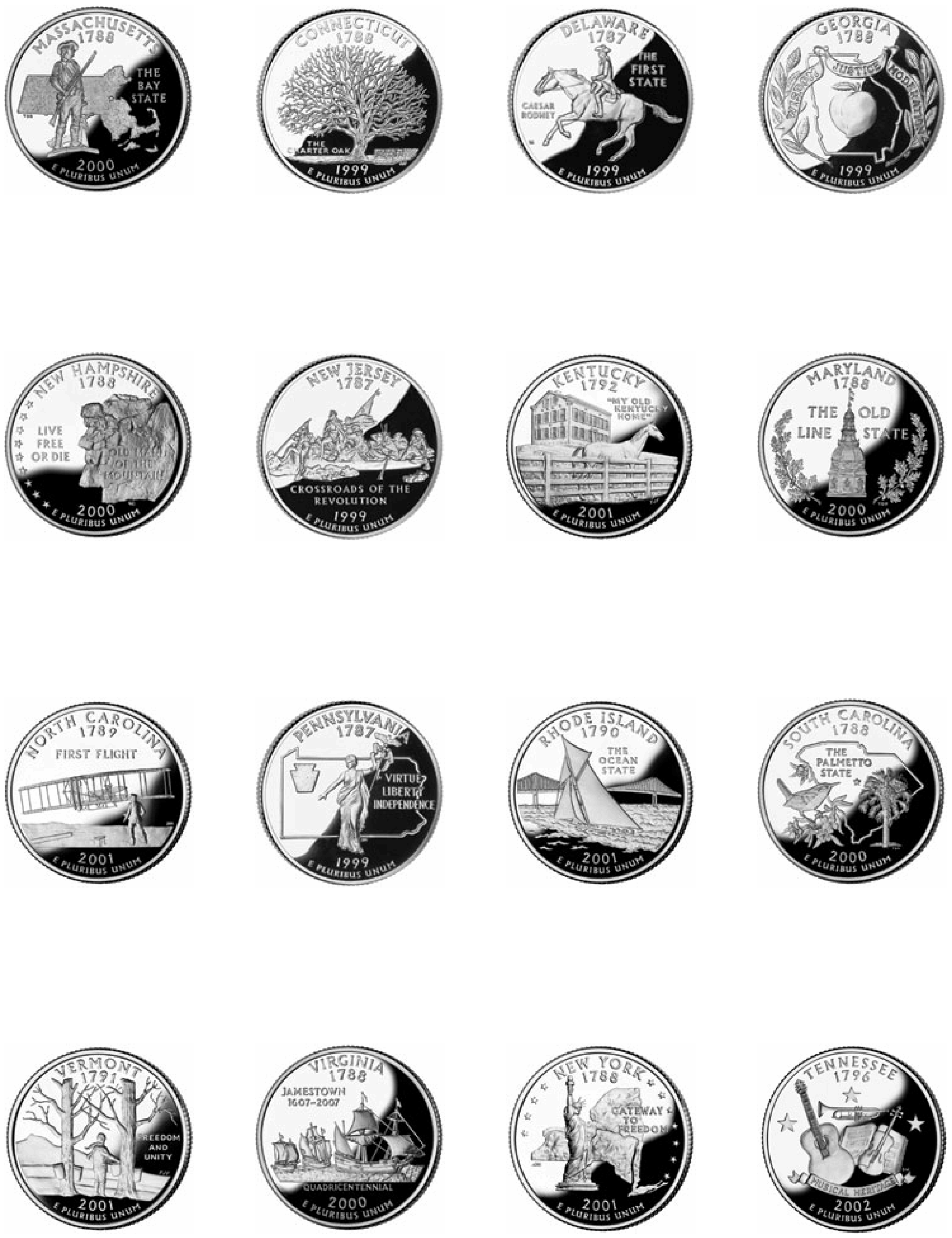
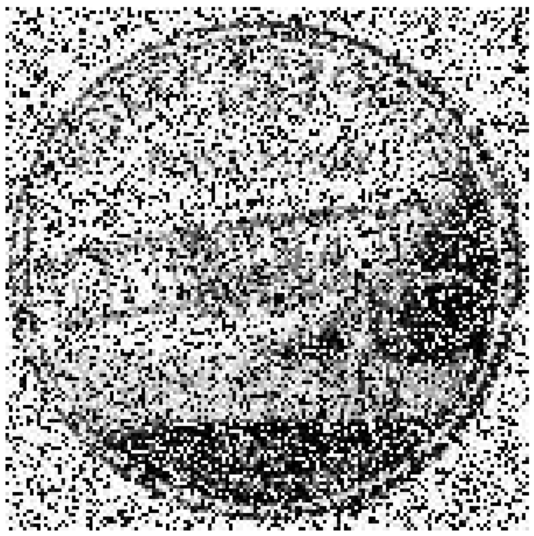
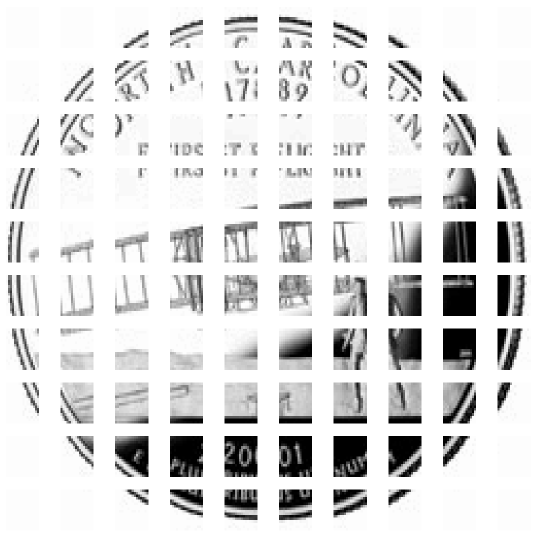


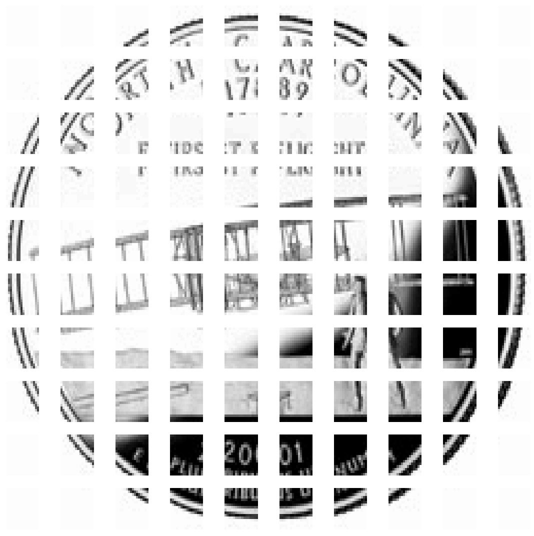
Fig. 3. Quantized prototype monochrome image patterns with 6 bits/pixel.



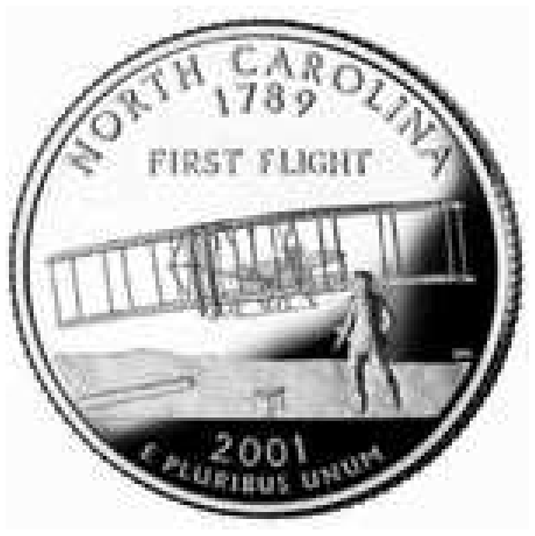
(a) Input image



(b) Result after the recall process



(c) Result after the error correction process

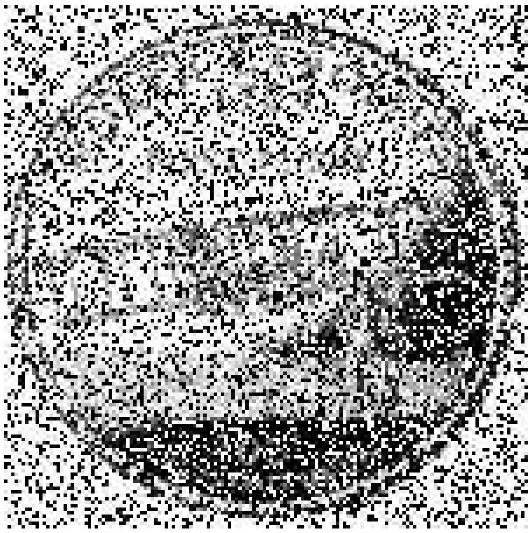


(d) Output image

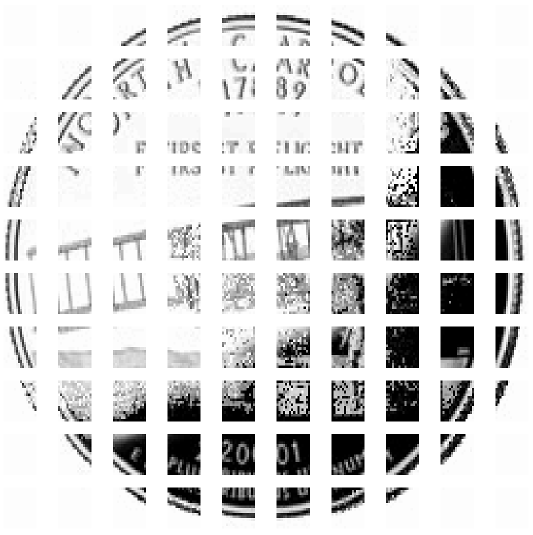
Fig. 4. Simulation result of image recall by gBSB neural networks with monochrome images.

using the overlapping decomposition method described in Section 3. Each sub-pattern went through recall process of the corresponding sub-network. The result after the recall processes of all the sub-networks is shown in Fig. 4(b). There were 5 mismatched portions between sub-patterns in this example. The next stage was the error correction process. The collection of sub-images in Fig. 4(c) is the result of the error correction process. There was no mismatched portion between these sub-patterns. Finally, the reconstructed image is shown in Fig. 4(d). In this image, there was no erroneously reconstructed pixel out of $22500\,(150 \times 150)$, i.e., no pixel in the reconstructed image had different values than the corresponding stored prototype image.

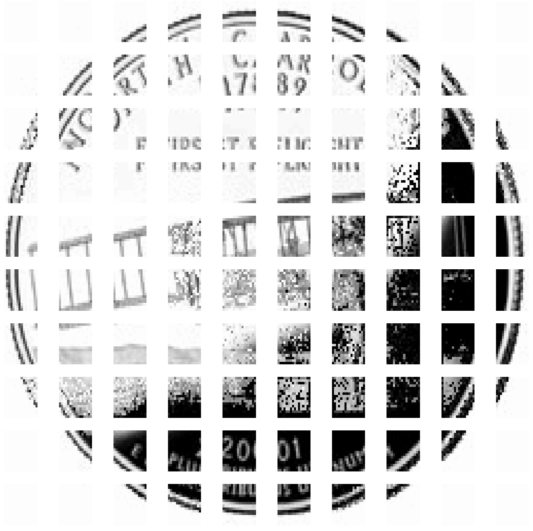
In Fig. 5, an example result of the image recall system employing the hybrid gBSB-McCulloch-Pitts neural model is shown. We used the same overlapping decomposition and the same noisy input image as in our simulation of the image recall system using the gBSB neural model, which is shown in Fig. 5(a). That is, the input image is corrupted by the salt-and-pepper noise with the error probability of 0.5, and the image was decomposed



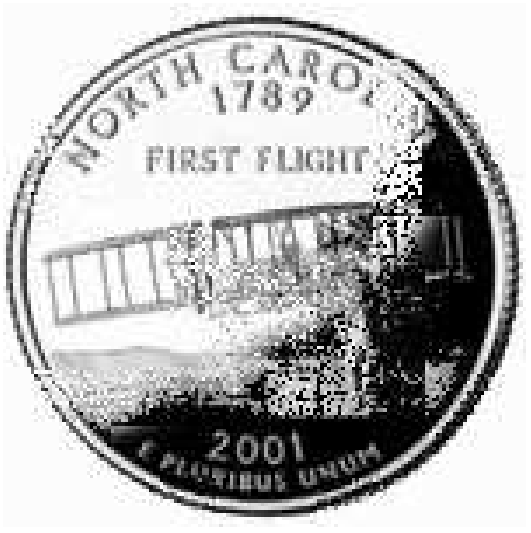
(a) Input image



(b) Result after the recall process



(c) Result after the error correction process

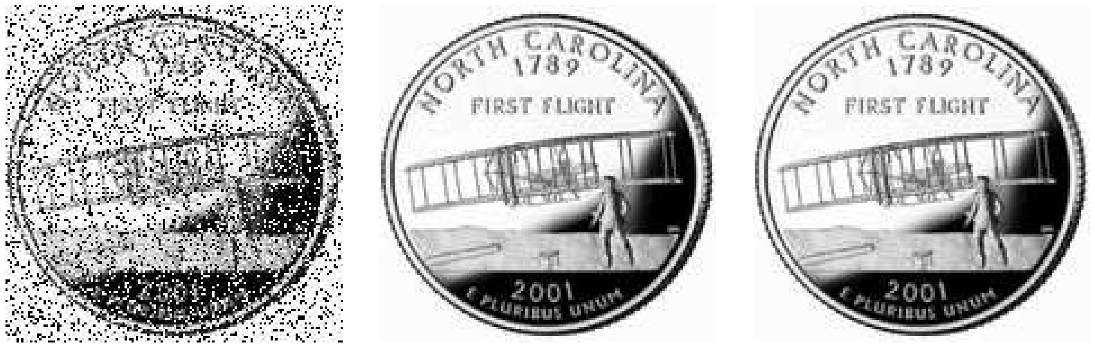


(d) Output image

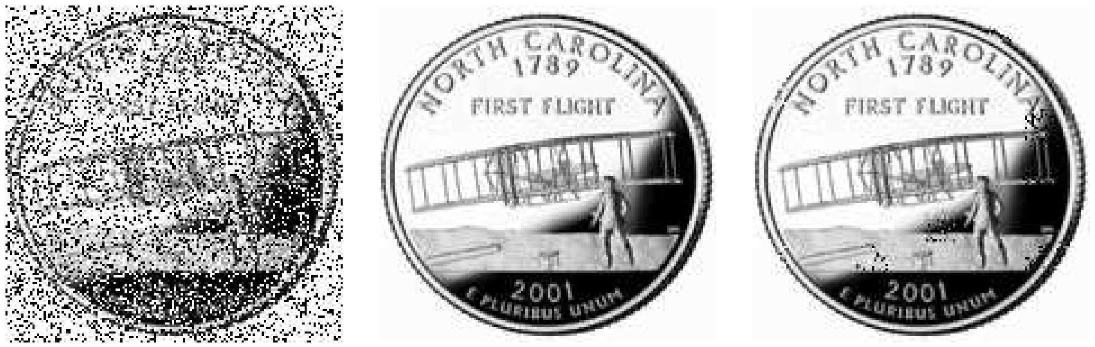
Fig. 5. Simulation result of image recall by hybrid gBSB-McCulloch-Pitts neural networks with monochrome images.

into 100 (10×10) sub-images using the overlapping decomposition method. The result of the recall processes is shown in Fig. 5(b), where there exist 87 mismatched portions between sub-images. Figure 5(c) is the result after the error correction process and there still remains 57 mismatches. The final result of image recall is shown in Fig. 5(d). In this result, 4491 pixels out of 22500 pixels have different values than the pixels of the corresponding stored prototype image, i.e., 19.96% of the whole pixels were erroneously reconstructed.

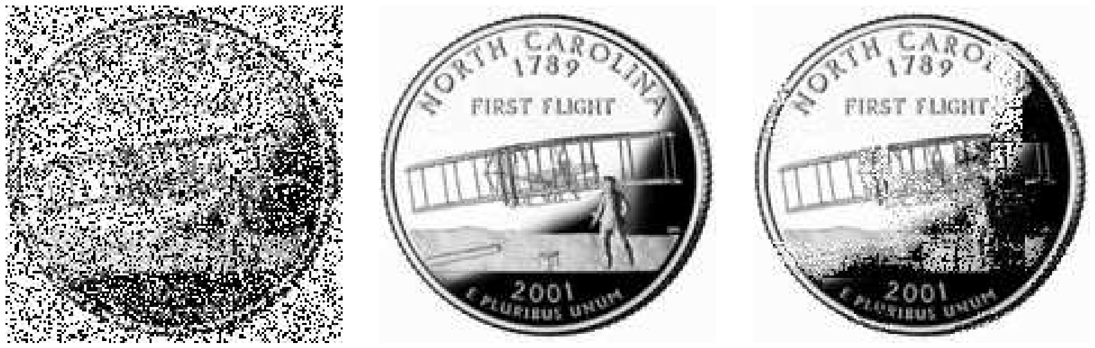
In Figs. 6 and 8(a), image recall systems using the gBSB model and the hybrid gBSB-McCulloch-Pitts model were compared with each other. The input images were corrupted by the salt-and-pepper noise with different error rates in this simulation. Also, in Figs. 7 and 8(b), the results of the simulations are shown when the input images were corrupted by the additive Gaussian noise, and the two models were compared with each other. The input images were corrupted by adding the Gaussian noise to the stored prototype image with different values of standard



(a) Input error rate: 0.3. Reconstruction error rate — gBSB model: 0, hybrid model: 0.



(b) Input error rate: 0.4. Reconstruction error rate — gBSB model: 0, hybrid model: 0.0198.

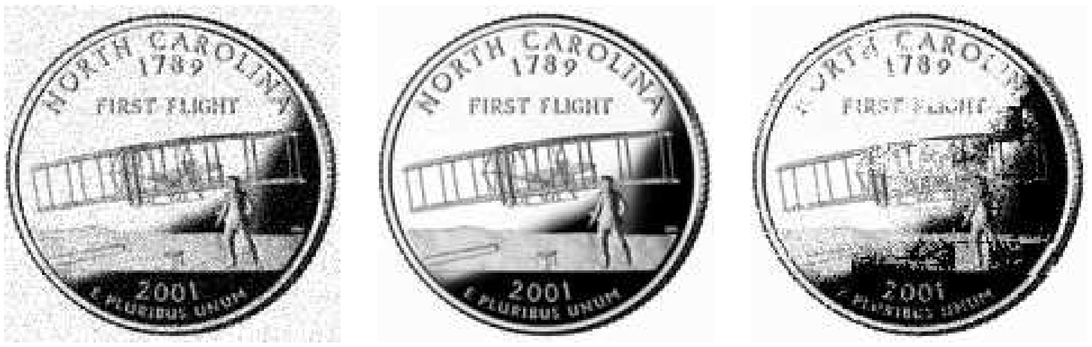


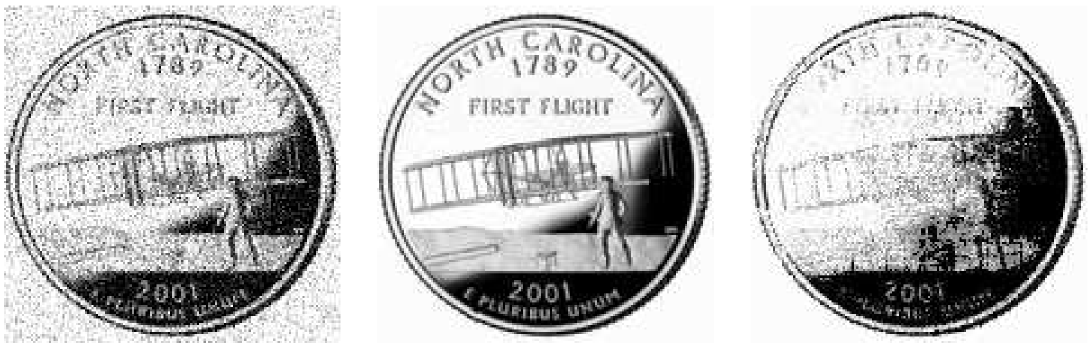
(c) Input error rate: 0.5. Reconstruction error rate — gBSB model: 0, hybrid model: 0.1996.

Fig. 6. Simulation results of image recall using the gBSB neural model and the hybrid gBSB-McCulloch-Pitts model (in the case of the salt-and-pepper input noise). Note—left: input images, center: gBSB model, right: hybrid model.

deviation that varies from 0 to 15. Each image used in this simulation had 64 gray levels because a pixel was represented by 6 bits. Therefore, for example, the standard deviation 15 means that the standard deviation of the Gaussian noise was almost a quarter of the full gray scale of the image. As we can see in these figures, the results from the system using the gBSB model were better than the ones from the system using the hybrid gBSB-McCulloch-Pitts model.

We can apply the same procedure to the recall of color images. The image patterns used in our simulation are shown in Fig. 9. The pixels in the original images were represented by 24 bits (8 bits for each of the R,G,B components) before the uniform quantization preprocessing. The image patterns in Fig. 9 are quantized versions of the original images with 6 bits per pixel (2 bits for each of the R,G,B components). An example of a simulation result is shown in Fig. 10. The image recall system was composed of gBSB neural networks in this simulation. The size of images used in this simulation was 300×200. The noisy input image in Fig. 10(a) was generated for the simulation in such a way that each of the three R,G,B ma-


(a) Standard deviation of the additive Gaussian noise in input: 5.
Reconstruction error rate — gBSB model: 4.89×10^{-4}, hybrid model: 0.456.


(b) Standard deviation of the additive Gaussian noise in input: 10.
Reconstruction error rate — gBSB model: 0.010, hybrid model: 0.577.

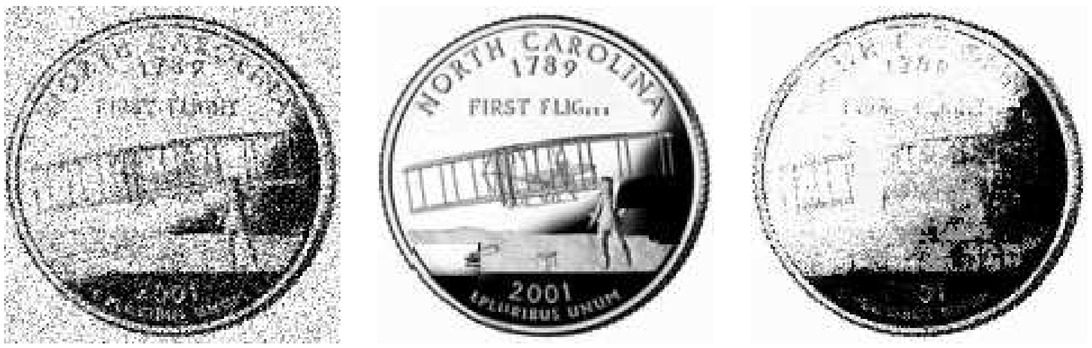
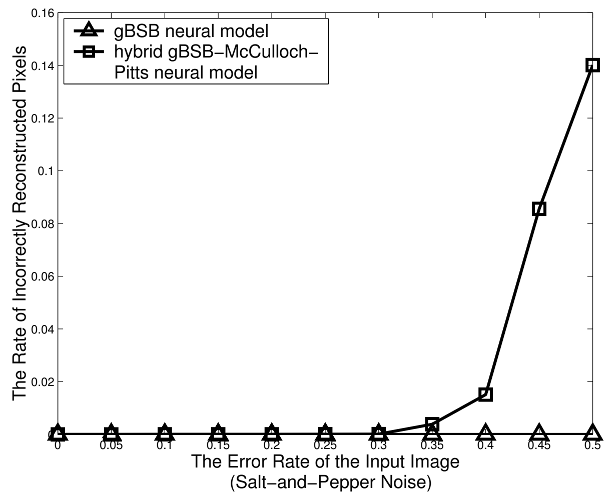

(c) Standard deviation of the additive Gaussian noise in input: 15.
Reconstruction error rate — gBSB model: 0.072, hybrid model: 0.609.

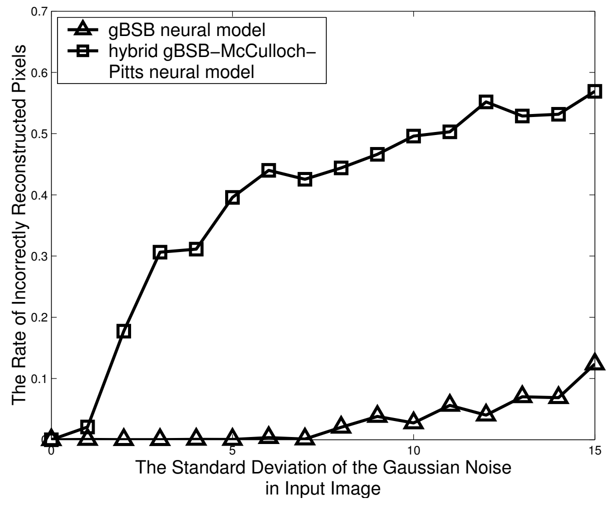
Fig. 7. Simulation results of image recall using the gBSB neural model and the hybrid gBSB-McCulloch-Pitts model (in the case of the Gaussian input noise). Note—left: input images, center: gBSB model, right: hybrid model.

trices was corrupted by the salt-and-pepper noise. The probability that each element of a pixel might be corrupted by the noise was 0.4 in this example. The patterns were decomposed into $600\,(= 30 \times 20)$ sub-patterns in this simulation. The number of mismatched portions between sub-patterns was 26 after the recall process, and it was reduced by the subsequent error correction process to 8. The final reconstructed image is shown in Fig. 10(d). The number of incorrectly reconstructed pixels was 150 out of 60000 pixels.

Remark 1. The gBSB based system outperformed the hybrid gBSB-McCulloch-Pitts network based system. An explanation for this can be found in the difference between the activation functions of the two models as given by (2) and (6), respectively. In the hybrid gBSB-McCulloch-



(a) Input noise: salt-and-pepper noise



(b) Input noise: Gaussian noise

Fig. 8. Comparison between the system employing the gBSB neural model and the system employing the hybrid gBSB-McCulloch-Pitts model.

Pitts neural network, the initial state $\boldsymbol{x}(0)$ moves to a vertex of the hypercube H_n in one step. If this vertex is a stable equilibrium state that is not a prototype vector, this results in a recall error. In other words, if the initial direction of the trajectory is towards a non-prototype stable equilibrium state, it yields a recall error. On the other hand, in the case of the gBSB neural network, the trajectory stays inside the hypercube for several time units depending on the step size α. This means that the convergence of the state in the gBSB network is less sensitive to the initial direction of the trajectory and it is possible that it changes the direction towards the corresponding stored stable equilibrium state as time passes even if the initial direction of the trajectory is towards the non-prototype stable equilibrium state.

Remark 2. The recall results of the hybrid gBSB-McCulloch-Pitts neural network based system significantly deteriorated, especially when an input image was corrupted by the Gaussian noise. This is because the binary representation of the image corrupted by the Gaussian noise might be very far in the Hamming distance sense from the stored prototype image to which the input pattern is supposed to converge. In the case of the salt-and-pepper noise, only the pixels selected with a given error probability are replaced with pixels representing white or black. In the case of the Gaussian noise, most pixel values are changed by adding or subtracting small numbers to them, and the binary representation of the modified values can be very far from the binary representation of the original pixel values. The image corrupted by the Gaussian noise may be an image that is very remote from the stored prototype image in the Hamming distance sense and may not belong to the basin of attraction of the corresponding stored prototype image.

6. Conclusions

In this paper, we described an image recall system that we constructed by employing a large scale gBSB neural network. We used the overlapping decomposition method to construct this network. This recall system works as neural associative memory that also contains the error correction subsystem to enhance the recall performance. The proposed system was able to recall the prototype images that were stored in the neural network when the noisy input images were given, even when the probability that an image pixel was corrupted with a salt-and-pepper noise was quite high, even as high as 0.5. Also, when the input image was corrupted with the Gaussian noise, the proposed system successfully reconstructed the stored prototype image unless the standard deviation of the additive Gaussian noise was too high. We built another image recall system employing the hybrid gBSB-McCulloch-Pitts neural model, and compared the performance of this system with the one using the gBSB neural model. The performance of the hybrid gBSB-McCulloch-Pitts neural network based system was not as good as the gBSB model based system, especially when the input noise was the Gaussian noise and when the input error rate of the salt-and-pepper noise was high.

In this paper, we assumed that the test images were simply noisy versions of the prototype images. If the test images were rotated, scaled, or shifted, the performance of the proposed system would most probably deteriorate

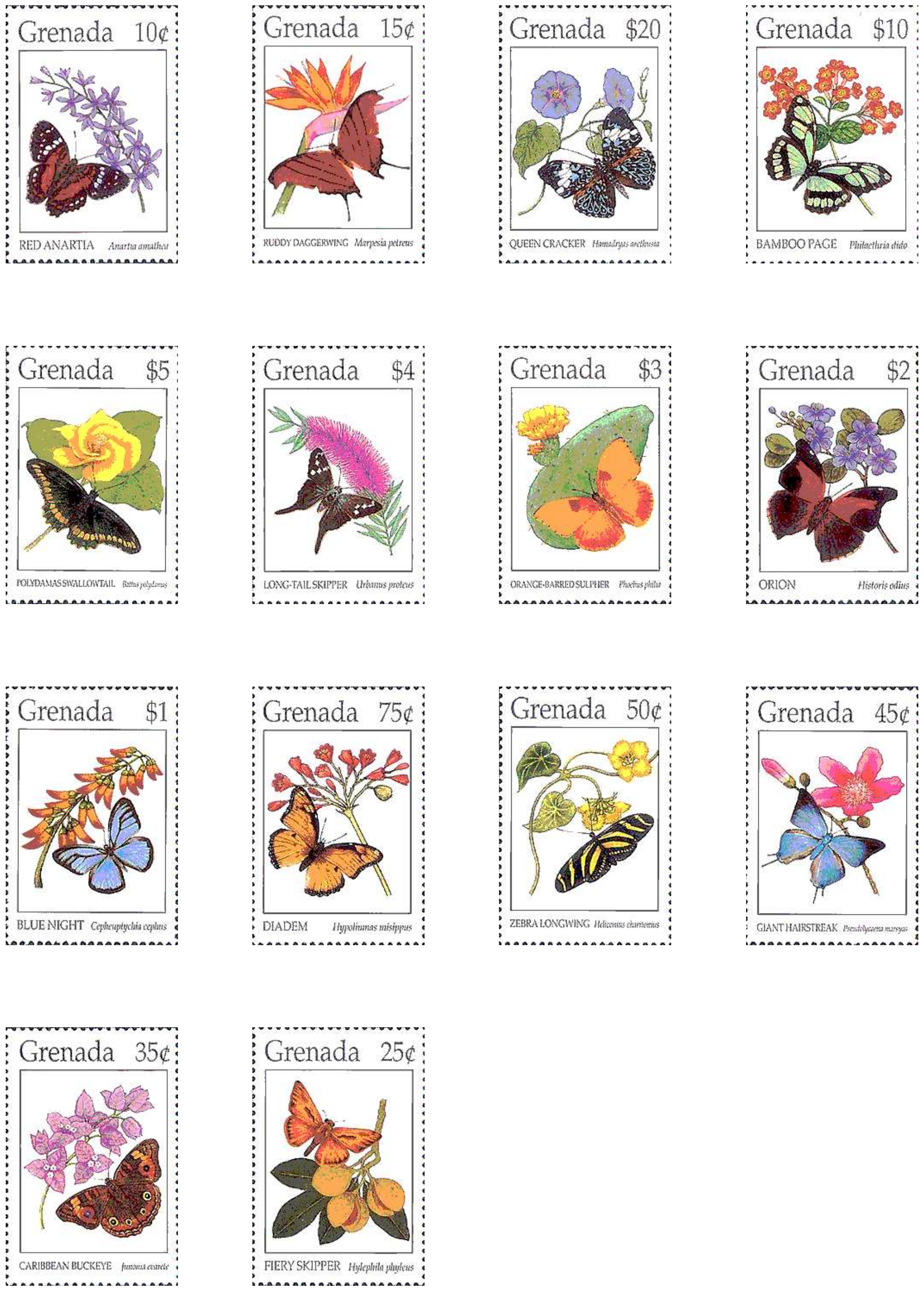
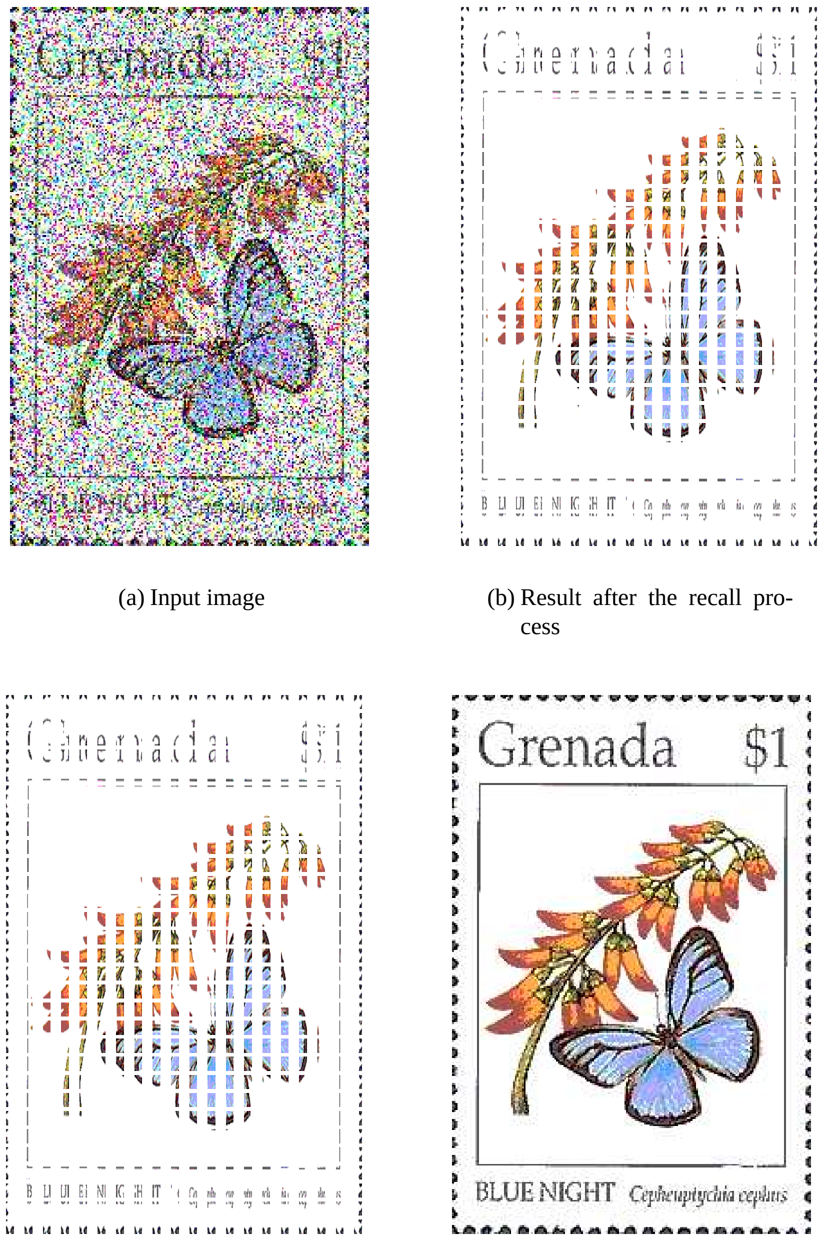

Grenada 10¢
RED ANARTIA Anartia amathea
Grenada 15¢
RUDDY DAGGERWING Marpesia petreus
Grenada $20
QUEEN CRACKER Hamadryas arethusa
Grenada $10
BAMBOO PAGE Philaethria dido
Grenada $5
POLYDAMAS SWALLOWTAIL Battus polydamas
Grenada $4
LONG-TAIL SKIPPER Urbanus proteus
Grenada $3
ORANGE-BARRED SULPHER Phoebis philea
Grenada $2
ORION Historis odius
Grenada $1
BLUE NIGHT Cepheuptychia cephus
Grenada 75¢
DIADEM Hypolimnas misippus
Grenada 50¢
ZEBRA LONGWING Heliconius charitonius
Grenada 45¢
GIANT HAIRSTREAK Pseudolycaena marsyas
Grenada 35¢
CARIBBEAN BUCKEYE Junonia evarete
Grenada 25¢
FIERY SKIPPER Hylephila phyleus

Fig. 9. Quantized prototype color image patterns with 6 bits/pixel.

significantly. Designing a system whose performance is insensitive to those kinds of transformations of test images is an open problem and it is left out for future research activities. We used binary representation of the intensity values of images. The use of different image representation methods in the proposed image recall system can be considered. Constructing an improved image representation method seems to be an interesting research topic. The uniform quantization method is used to reduce the computational load. The main reason why we used the uniform quantization is its simplicity. It would enhance the quality of quantized images if we used a quantization method that depends on the statistics of the training images. We leave this issue for future research.

Acknowledgement

We are grateful for constructive comments of the reviewers.



(a) Input image

(b) Result after the recall process

(c) Result after the error correction process

(d) Output image

Fig. 10. Simulation result of image recall using the gBSB neural model with color images.

References

Akar M. and Sezer M.E. (2001): *Associative memory design using overlapping decompositions*. — Automatica, Vol. 37, No. 4, pp. 581–587.

Anderson J.A. (1995): *An Introduction to Neural Networks*. — Cambridge: A Bradford Book, The MIT Press.

Anderson J.A., Silverstein J.W., Ritz S.A. and Jones R.S. (1989): *Distinctive features, categorical perception, probability learning: Some applications of a neural model*, In: Neurocomputing; Foundations of Research (J.A. Anderson and E. Rosenfeld, Eds.). — Cambridge, MA: The MIT Press, ch. 22, pp. 283–325, reprint from Psych. Rev. 1977, Vol. 84, pp. 413–451.

Golden R.M. (1993): *Stability and optimization analyses of the generalized Brain-State-in-a-Box neural network model*. — J. Math. Psych., Vol. 37, No. 2, pp. 282–298.

Gonzalez R.C. and Wintz P. (1987): *Digital Image Processing, 2nd Ed*. — Reading: Addison-Wesley.

Gray R.M. and Neuhoff D.L. (1998): *Quantization*. — IEEE Trans. Inf. Theory, Vol. 44, No. 6, pp. 2325–2383.

Hassoun M.H. (1995): *Fundamentals of Artificial Neural Networks*. — Cambridge: A Bradford Book, The MIT Press.

Hui S. and Żak S.H. (1992): *Dynamical analysis of the Brain-State-in-a-Box (BSB) neural models*. — IEEE Trans. Neural Netw., Vol. 3, No. 1, pp. 86–94.

Ikeda M. and Šiljak D.D. (1980): *Overlapping decompositions, expansions and contractions of dynamic systems*. — Large Scale Syst., Vol. 1, No. 1, pp. 29–38.

Ikeda N., Watta P., Artiklar M. and Hassoun M.H. (2001): *A two-level Hamming network for high performance associative memory*. — Neural Netw., Vol. 14, No. 9, pp. 1189–1200.

Lillo W.E., Miller D.C., Hui S. and Żak S.H. (1994): *Synthesis of Brain-State-in-a-Box (BSB) based associative memories*. — IEEE Trans. Neural Netw., Vol. 5, No. 5, pp. 730–737.

Oh C. and Żak S.H. (2002): *Large scale neural associative memory design*. — Przegląd Elektrotechniczny (Electrotechnical Review), Vol. 2002, No. 10, pp. 220–225.

Oh C. and Żak S.H. (2003): *Associative memory design using overlapping decomposition and generalized Brain-State-in-a-Box neural networks*. — Int. J. Neural Syst., Vol. 13, No. 3, pp. 139–153.

Park J. and Park Y. (2000): *An optimization approach to design of generalized BSB neural associative memories*. — Neural Comput., Vol. 12, No. 6, pp. 1449–1462.

Park J., Cho H. and Park D. (1999): *Design of GBSB neural associative memories using semidefinite programming*. — IEEE Trans. Neural Netw., Vol. 10, No. 4, pp. 946–950.

Sayood K. (1996): *Introduction to Data Compression*. — San Francisco: Morgan Kaufmann.

Schultz A. (1993): *Collective recall via the Brain-State-in-a-Box network*. — IEEE Trans. Neural Netw., Vol. 4, No. 4, pp. 580–587.

Zetzsche C. (1990): *Sparse coding: the link between low level vision and associative memory*, In: Parallel Processing in Neural Systems and Computers, (R. Eckmiller, G. Hartmann and G. Hauske, Eds.). — Amsterdam: Elsevier, pp. 273–276.

Received: 16 August 2004
Revised: 11 January 2005